CARL
VON
OSSIETZKY
universität OLDENBURG

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Leakage Models for High Level Power Estimation

Dissertation zur Erlangung des Grades eines
Doktors der Naturwissenschaften

vorgelegt von

Dipl.-Phys. Domenik Helms

25.5.2009

October 27, 2009

Contents

Preface

When working in the well established field of register transfer (RT) level modelling, one will soon find out, that doing RT level leakage estimation is by no means a simple modification of timing, area or dynamic power modelling. There are several factors, making leakage macro modelling a unique and completely new challenge:

Leakage is barely state dependent from an RT level view. Instead, dynamic parameters highly influencing leakage are temperature ϑ (dominating subthreshold leakage) and supply voltage V_{DD} (having highest influence on gate and pn-junction leakage). Process variations also are a new challenge for RT models. All leakage currents are highly dependent on variation of certain process parameters [1, 2].

Especially the high sensibility to magnitude and distribution of lowest level technological parameters prohibits a simple function-like 'parameters-in estimation-out' model interface. When being used for design space exploration, in-depth information like spread of channel doping concentration may not be available. Instead, RT level leakage models have to integrate parameter distribution prediction and estimation of the resulting leakage currents.

Low leakage power management techniques, as adaptive body biasing (ABB) or power gating have large impact on leakage currents. But in contrast to direct leakage reduction techniques as high-k oxides, they have a variable influence on the leakage being controllable from outside an RT component and thus cannot be abstracted within the model. A leakage model supporting leakage power management will thus have to regard these techniques as input parameters.

Parameter variation as well as power management techniques will not just affect the leakage current of a component, but also its timing. For a single transistor, the subthreshold leakage, its body potential controlled by ABB, and its operation speed all are highly correlated. Thus a holistic leakage model regarding parameter distribution and power management techniques will have to communicate and share information with an adequate delay model. Only this way, the power-delay correlation, being characteristically for sub-100nm systems and having largest impact on yield, can be described accurately.

This thesis will motivate, develop, and evaluate an RT level macro modelling methodology satisfying all these demands. The estimation framework is embedded into the commercial behavioural-to-RT synthesiser tool PowerOpt, where the model predictions guide the synthesis of a SystemC/C++ system specification. The framework consists of the model itself, a floorplan based temperature and voltage drop model, and a variation engine. This engine is translating high level metrics into parameter variation measures, which are then used for leakage and performance prediction. This way, the high correlations between leakage and performance [3] as well as the thermo-electrical coupling and electro-electrical coupling [4] which are characteristic for today's devices can be accurately described and thus also automatically improved by various optimisation techniques.

Abstract

Leakage currents are one major concern when designing sub-100nm CMOS devices, making design for leakage at all stages of the design process mandatory. Early leakage optimisation requires early leakage prediction, and for electronic system level design, this means estimation capabilities at RT level or above. Existing models either offer very accurate, but also very slow simulation at transistor level (such as BSIM or PSP), or faster modelling at gate level, disregarding relevant parameters (such as the Liberty library). The goal of this thesis is to develop RT level leakage macro models, being even faster than recent gate level models, while regarding all relevant parameters and thus being just slightly less accurate than transistor level models.

The main contributions of this work are

- A powerful, yet fast single transistor leakage model, which can be characterised using industrial standard models.
- To cope with the complexity, this model is abstracted in layers, first towards gate models, then RT hard macros, and finally RT soft macros, while explicitly or implicitly preserving parameter influences.
- All relevant parameters (such as varying process parameters or temperature) are forwarded through all abstraction layers and can still be explicitly regarded at system level.

In this thesis, such a respective estimation framework is proposed, describing the subthreshold, gate, and junction leakage of industrial 90nm, 65nm, and 45nm devices. The models are characterised using BSIM compact models and a Monte Carlo process variation description. Each varying BSIM parameter can be described; as an example of use, channel length, oxide thickness and channel doping are regarded together with the temperature, supply voltage and body voltage. The final macro model needs less than a hundred parameters to capture the leakage behaviour of an entire family of RT components. Compared to SPICE/BSIM, a model prediction is computed up to a million times faster for large RT components, and is within $3.6\% - 6.9\%$ standard deviation (depending on the analysed technology) over a wide range of operating conditions and process variation settings.

1 Introduction

The first major power motivated design-flow change became necessary in the mid-1980s. The large short circuit currents flowing through each NMOS gate at logic *0* where about to exceed a power density of $10W/cm^2$. With the advent of CMOS, the dominant short circuit currents were reduced by orders of magnitude for the cost of nearly twice as much transistors being needed for the same logic. Besides the remaining short-circuit currents barely appearing when switching, CMOS power consumption is dominated by capacitance charging currents.

Despite the large reduction of the overall power consumption in CMOS, the pace of Moore's Law soon led to unacceptably high power densities again. In order to prevent exceeding $10W/cm^2$, the second major power motivated design-flow change was turning away from constant voltage scaling towards constant field scaling. Gordon Moore said, that the MOSFET world is a $5V$ world, and that changing the voltage means that the world would have to change. Accordingly, the world changed.

Scaling technology with constant fields is triggering a chain of subsequent parameter modifications: The threshold voltage needs to be reduced to keep performance high, the oxide thickness needs to be scaled down even faster than the general scaling factor to keep the gate-capacitance high, thus subthreshold swing low, thus $I_{\text{on}}/I_{\text{off}}$ ratio high, thus the performance high again.

By consequently scaling technology down further, we now reached a state in the mid-2000s, where parasitic leakage effects do not only dominate the power budget but also become a concern as important as the performance which used to be the most important scaling concern since the advent of MOSFETs. According to the ITRS roadmap [5], the top challenges towards the $32nm$ technology node are controlling leakage while increasing performance: On the one hand, the channel needs to be controlled more efficiently by the gate, increasing the $I_{\text{on}}/I_{\text{off}}$ ratio, thus keeping channel leakage low with rising performance[1]. On the other hand, the gate capacitance needs to maintain high while scaling technology down without a further reduction of the oxide thickness, which is already at its physical limit[2].

According to Kaushik Roy's 2003 prediction [7], but in contrast to the most famous leakage chart [8], the ITRS 2006 update predicts, that dynamic power and channel leakage will approximately break even below $45nm$ devices as presented in Figure 1.1. With gate current exploding for conventional bulk CMOS, high performance high-k gate dielectric devices need to be introduced soon. Fully depleted ultra-thin body MOSFETs (UTB) in 2010 will reduce the subthreshold leakage as the thin

[1] This can be done by introduction of multi-gate devices as SOI back-gates, FinFETs, and all-around-gates.

[2] A high gate capacitance is important for the subthreshold slope thus for a high $I_{\text{on}}/I_{\text{off}}$ ratio. A further reduction of the oxide thickness would boost the tunnelling currents, already dominating recent technology. The introduction of completely new dielectrics and metal gates, reducing the parasitic capacitance loss, is the only solution known.

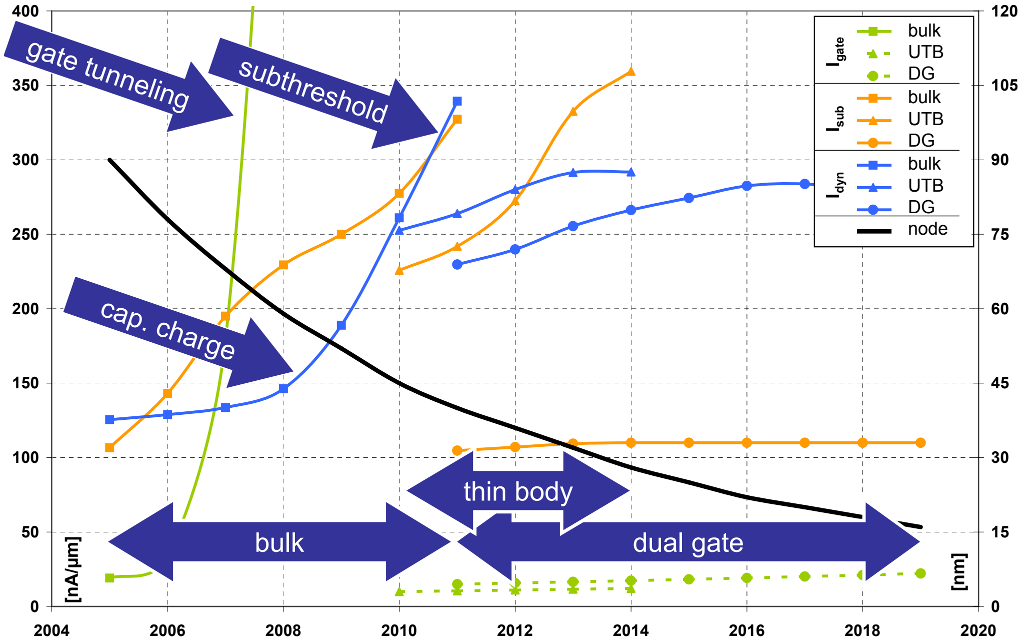


Figure 1.1: High performance logic ITRS prognosis [6] of the gate and channel leakage versus dynamic charge current and node size.

channels can effectively be switched (see Section 1.4.1). The introduction of dual gate devices will bring significant improvements to the system's leakage currents (see Figure 1.1). Accordingly, leakage currents will not be the *red brick wall* to stop Moore's law, but will also not be removed for good by a new technology for the next logic generations. Instead, leakage will remain one top concern (not *the* top concern) in CMOS design.

The leakage modelling framework, being the main focus of this thesis, enables accurate leakage estimation of RT netlists. Accurate estimation, early in the design process, is the key enabler for powerful design-space exploration. Using the high degrees of freedom and the fast design iteration of early design phases is always the key for improving design parameters. Thus, by enabling early leakage estimation, this work can contribute to leakage optimisation.

Terminology:

Throughout this work, the terms instance, system, component, and device are used in a well defined way:

- The term **device** is used only for single transistors or smallest circuits of transistors (less than a logic gate).
- A **component** always refers to an RT component of at least tens of gates and not more than some thousand gates.

- A **system** is a large integrated circuit implemented on its own piece of silicon.
- An **instance** is one explicit realisation of a system, component, or device. Example: A device instance is a certain transistor physically or theoretically built in silicon, underlying explicit variation influences. The exact geometry of an instance is fixed and known.

Model parameters are distinguished as follows:

- Static parameters (*StPa*) means parameters, that are fixed, once a system instance was fabricated, such as channel length, oxide thickness, or doping concentration.
- In contrast, dynamic parameters (*DyPa*) can change after production, such as supply voltage (V_{DD}), body voltage (V_{BB}), temperature, or input data.

Remark for the reader:

The entire Chapter 1 contains necessary background to enable the understanding of the later chapters for a reader not perfectly aware of leakage physics, leakage modelling, and design for leakage. My own work is only redundantly mentioned here as a contribution to the state of the art. From Chapter 2 on, I will explicitly refer to the respective sections of Chapter 1, wherever needed. A reader who is familiar with these leakage topics should be able to directly proceed from Chapter 2 on.

1.1 State of the art leakage modelling

This section will survey the state of the art in leakage modelling. First, related work sorted by research group is presented. Then, my two approaches, the bottom-up and the top-down approach are described. As this thesis employs the bottom-up approach as the core for leakage estimation, it will only be briefly reviewed here and completely described from Chapter 2 on.

1.1.1 Motivation of my work

In 2000, [9] presented a first useful system level complexity based leakage estimation methodology. This separation approach divides leakage current into transistor count n, supply V_{DD}, leakage per device I_{dev} and a constant k as

$$I_{\text{leak}} = k \cdot n \cdot V_{\text{DD}} \cdot I_{\text{dev}}, \tag{1.1}$$

where k is a design complexity metric. Even though this complexity based approach is far too simplistic, it motivated my work: For the top-down model, data dependency is introduced to the parameter k. With this data dependency, k accurately separates from the other parameters for the Predictive Technology Model (PTM) $70nm$ technology.

An extension to the elementary approach [9], proposed by [10], where the complexity based leakage computation was distributed into NMOS and PMOS part as

$$I_{\text{leak}} = k_{\text{NMOS}} \cdot n_{\text{NMOS}} \cdot I(V_{\text{DD}}, T)_{\text{dev NMOS}} + k_{\text{PMOS}} \cdot n_{\text{PMOS}} \cdot I(V_{\text{DD}}, T)_{\text{dev PMOS}}, \tag{1.2}$$

was the initial motivation for the bottom-up model, and thus for parts of this thesis. The results of [10] have been implemented into the first published leakage estimation tool, called HotLeakage.

[11] computes subthreshold leakage variation under intra-die variation; regarding gate length, oxide thickness, and channel doping. A parameter sensitivity analysis enables estimation of the effect of a variation on the average leakage. This work motivated the variation engine used in this thesis, focusing on inter-die modelling, handling the effect of intra-die variation by an increased average only.

1.1.2 Other groups

Chenming Hu Group, University of California, Berkeley, TSMC

By introducing the Berkeley Simulator (BSIM) [12,13], the Predictive Technology Model (PTM) [14], and a set of reference MOSFET modelcards ranging from $180nm$ to $22nm$, Berkeley enabled evaluation of leakage estimation and optimisation approaches even without the need for expensive silicon production or intimate knowledge of confidential industrial technology data. In [15], an automated modelcard generation methodology is presented, facilitating the description of existing technology in BSIM.

Dennis Sylvester Group, University of Michigan, IBM

In [16] the sensitivity analysis of [11] is extended to regarding intra- and inter-die variation of the printed gate length. This idea was refined in [17]. The characterisation based model can accurately predict the probability density function of the leakage distribution.

[18] reported gate level state dependent leakage estimation by identifying all possible leakage states of a transistor. Instead, in [19], a static data-dependency analysis predicts the average (data independent) leakage with a general model using the leakage sensitivity to an arbitrary parameter.

Vivek De Group, University of California Santa Barbara, Intel

In [20, 21], thermally dependent leakage estimation was combined with a chip-wide temperature prediction, thus also regarding the electro-thermal back-coupling introduced by the subthreshold current's thermal dependence. [22] analyses the impact of threshold voltage variations in order to handle intra-die process variations.

A complete high level leakage model was presented by [23], regarding all PTV variations, thus all parameters except for the state. Even though they (Intel Hillsboro) may have a holistic model, no technical details are revealed, their work, as published, is not reporting absolute leakage estimates, but only analyses the influence of all relevant parameters.

Kaushik Roy Group, Purdue University

[24] presents a subthreshold leakage estimation methodology for computing lower and upper bounds by regarding the stacking effect. [25] analyses the leakage distribution for the one and two input

gates reporting a substantial state dependency. There are many approaches at this level enabling accurate leakage current prediction if all relevant parameters are exactly known per device. But some parameters are not accurately predictable on the lower levels. The effect of parameter variations on leakage was analytically investigated in [7]. This approach analyses the most important sources of leakage and predicts their distribution for $90nm$, $50nm$, and $25nm$.

In order to combine different $DyPa$, an iterative approach is presented by [4], accurately modelling dynamic power and leakage power by regarding the interaction between temperature, supply voltage, and power consumption. The authors introduce a thermal system model handling the electro-thermal coupling, as well as a supply-grid model handling the electro-electro coupling introduced by IR drops.

1.1.3 Empirical RT bottom-up model

This model [26] consists of four abstraction layers: Physical transistor model, state dependent gate model, simulation based RT hard-macro, and abstracted RT soft macro.

Physical transistor model

The transistor models capture the technologies dependence on $DyPa$ and the dependence on $StPa$. As only a few models for transistors in typical states (NMOS & PMOS, non-conducting & conducting) are needed and the width of the transistor enters the leakage almost linear, a tabular interpolation model is sufficient here. The effect of statistical variation (intra-die variation) is assumed to be known at characterisation time and invariant for one technology. Thus, the effect of intra-die variation is covered implicitly by the model.

State dependent gate model

The gate level models use the transistor models, trying to explain the leakage of each gate in each state as a linear combination of the reference transistors. Only few characterisation data is needed, and only a small number of model parameters has to be stored. This model stage assumes, that all transistors of the gate underlie exactly the same inter-die variation.

Simulation based RT hard-macro

For model characterisation, zero delay simulation is used to gather the state of each gate inside a given gate-netlist. Exploiting the easy mathematical structure of the gate level model, an entire netlist with known input vector can be described using the gate level model structure. As the compression is an accurate mathematical transformation, this step is error free as long as there is no significant gradient of the $DyPa$ or $StPa$. An entire RT component with constant temperature, voltage and inter-die variation, and with fixed input vector can be modelled as a single super-gate.

Abstracted RT soft macro

Having a separate model for each RT component at each bitwidth and for each possible input vector is obviously prohibitive. Thus, for each RT component, only samples with different size and data are generated. Then, size and data are abstracted resulting in a single model per component and architecture. That means, that two models are needed for a ripple and a look ahead adder, but one model can describe all bitwidths for one of these components under each possible input vector.

1.1.4 Analytical RT top-down model

The analytical model starts with the complexity based separation approach (Equation 1.1) from [9] and introduces data dependency and a characterisation methodology. Correction terms describing threshold voltage variation, an absolute shift due to inter-die and a deviation due to intra-die, are developed for the current per device $I_{\text{dev}}(\mu_{\text{Vth}}, \sigma_{\text{Vth}})$. Analysis shows, that thermal dependencies do not accurately separate.

In the bottom-up model, the *DyPa* temperature, supply voltage, and body voltage, and the *StPa* of channel length, oxide thickness, and channel doping are explicit input parameters. In this analytical model [27], only supply voltage, temperature and component state directly enter the model. Bulk voltage, deviating due to ABB indirectly enters the model by modifying the effective threshold voltage. Modelling a huge number of transistors in one model, random process variations (intra-die variations) only enter due to the non-linearity of the $I_{\text{leak}}\left(V_{\text{th}}\right)$ relation which is captured inside the models as presented in [11, 16, 22].

Other work

[28] regards the distribution of variations by presenting a statistical leakage and delay model on gate level, thus enabling optimisation of the gate size to minimise leakage power.

In [29] a methodology is presented, estimating the probability density function of the leakage current due to process parameter variation, enabling accurate yield prediction, by introduction of a intra-die gradient model.

Best to my knowledge, the only public approaches enabling PTV and state dependent RT level leakage analysis are my empirical bottom-up [26] and my analytical top-down approach [27]. The strengths of the analytical model are easy model characterisation and very fast model evaluation. The advantages of the empirical model are higher modelling accuracy and more direct model parameters.

1.2 Leakage sources in MOS transistors

Subthreshold leakage was the first leakage effect, significantly influencing the power budget of MOS transistors. But as transistor scaling reaches atomic dimensions, several other physical leakage current mechanisms, negligible in macroscopic devices, exponentially grow in importance with each new technology generation. In this section, all known physical leakage mechanisms are presented (rf. Figure 1.2), and in the Section 1.3, the influence to outer parameters is discussed.

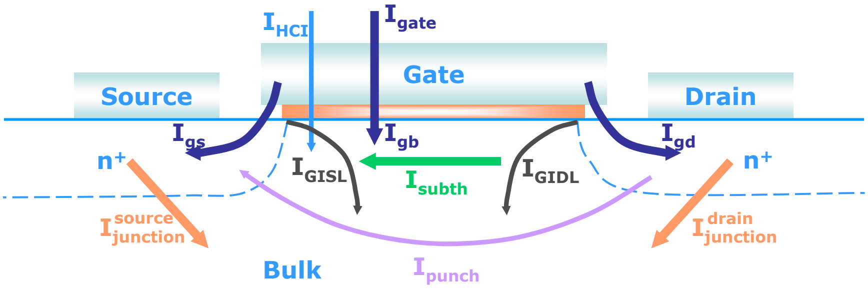


Figure 1.2: Leakage currents of an NMOS transistor: The drain-source leakage for a locking transistor is dominated by subthreshold leakage I_{subth}. The gate leakage can be divided into the major effect of gate tunnelling I_{gate} and hot-carrier injection I_{HCI}. Junction leakage I_{junction} consists of the well known diode currents which can be boosted by gate induced drain leakage I_{GIDL}.

1.2.1 Drain-source leakage

This section will explain the subthreshold current by an observation of the channel current's gate voltage dependency, also presenting gate induced drain leakage (GIDL) and drain induced barrier lowering (DIBL). Afterwards, the analytical description of subthreshold current and threshold voltage are presented and discussed. Finally, the punchthrough effect is presented.

Subthreshold leakage

The dominating drain-source leakage is the subthreshold leakage [1], I_{subth} in Figure 1.2. The occurrence of subthreshold leakage can be easily understood, when analysing the $log\,(I_{\text{ds}}(V_{\text{gate}}))$ behaviour, as presented in Figure 1.3. Shown by graph A, the working current I_{on} flowing at high gate voltage decreases almost linearly in the semi-logarithmic plot, indicating exponential decrease of the channel current itself. The gradient $S = \Delta V_{\text{gate}}/\Delta(log_{10}I/I_0)$ is called subthreshold slope. For the $130nm$ PTM technology analysed here, the subthreshold slope is $S = 80mV/dec$. The fundamental limit for the slope of planar, single-gated CMOS is $S \geq 63mV/dec$ at $T = 300K$ as shown by [30]. As a result of the finite slope, even for $V_{\text{gate}} = 0$, a small subthreshold current from drain to source remains.

Comparing graph A and B in Figure 1.3, the difference between the behaviour of the $130nm$ and $45nm$ technology, besides a higher I_{on}, is a much lower threshold voltage. The subthreshold slope is still at $S = 80mV/dec$, close to the fundamental limit. Obviously, each threshold voltage reduction of $80mV$ leads to a factor ten increase in the off-current.

Graph B was obtained, using a small drain source potential of $V_{\text{ds}} = 100mV$, just enough to generate a channel current. When increasing the potential to $V_{\text{ds}} = 1.0V$ or $V_{\text{ds}} = 2.0V$, the $45nm$

technology graph changes (Graph C): Of course, I_{on} rises as a direct result of the higher potential. But additionally, the threshold-voltage seems to become lower with rising V_{ds}. This effect is called drain induced barrier lowering (DIBL), and will be discussed later in this section. For the $45nm$ technology, a higher V_{ds} will not only increase I_{on} but also reduce the $I_{\text{on}}/I_{\text{off}}$ ratio, thus boosting the subthreshold leakage.

Graphs A-C are currents at the source contact on an NMOS device. When obtaining the current at drain, the drain-bulk leakage is observed, too. Presented in graph D, is the gate induced drain leakage (GIDL) effect. Depending on the drain-gate potential V_{dg}, a current, larger than the subthreshold current may flow from drain to bulk. This junction leakage effect is detailed in Section 1.2.3.

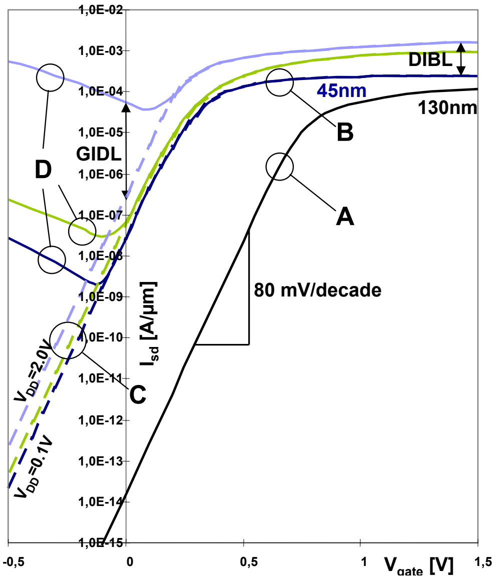


Figure 1.3: BSIM simulation of the channel current with respect to the gate voltage. **A:** Conventional behaviour of the $130nm$ technology. Below the threshold voltage, the channel current is reduced by a factor of ten per $80mV$ gate voltage. For $V_{gate} = 0$, the remaining subthreshold voltage is $10fA/\mu m$ **B:** For the $45nm$ technology, a reduced threshold voltage leads to a significantly larger subthreshold current of $30nA/\mu m$. **C:** For significant supply voltages, the drain induced barrier lowering (DIBL) influences the threshold voltage resulting in $300nA/\mu m$ off-current. **D:** The gate induced drain leakage (GIDL) is increasing the drain-bulk current, exponentially depending on the gate-drain potential.

Analytical description of the subthreshold current

The subthreshold current's analytical expression, according to [7] is

$$I_{\text{sub}} \propto \frac{W}{L} {V_{\text{T}}}^2 e^{\frac{V_{\text{gs}}-V_{\text{th}}}{SV_{\text{T}}}} \left(1 - e^{-\frac{V_{\text{ds}}}{V_{\text{T}}}}\right), \tag{1.3}$$

where L and W are the channel length and width, and $V_{\mathrm{T}} = k_B T/q^-$ is the thermal voltage, a voltage made proportional to the absolute temperature T by multiplication with Boltzmann's constant k_B and the elemental charge q^- [3]. At room temperature $T = 300K \Rightarrow V_{\mathrm{T}} \approx 25mV$. Simplified, to the most relevant parameters for completely discharged gate $V_g = 0V$ and for $V_{\mathrm{ds}} >> V_{\mathrm{T}}$, Equation 1.3 simplifies to

$$I_{\mathrm{sub}} = \alpha \frac{W}{L} T^2 e^{-\beta V_{\mathrm{th}}/T}. \tag{1.4}$$

The subthreshold leakage depends exponentially on the temperature and the threshold voltage. For the development of the model presented in Chapter 2, it is important to note, that the transistor width enters the subthreshold equation almost linearly besides a minor $V_{\mathrm{th}}(W)$ contribution, while the channel length dependence, especially the $V_{\mathrm{th}}(L)$ dependency is very complex.

Analytical description of the threshold voltage

The threshold voltage is developed in detail in the BSIM manual [31]. When being carefully simplified [32, 33], only regarding *DyPa* which may change at run time, and significant *StPa* which vary at production time, the complex threshold voltage equation can be reduced to

$$\begin{aligned}
V_{\mathrm{th}} &= V_{\mathrm{FB}} + \Phi_S(T) && (1.5)\\
&+ \gamma\left(\sqrt{\Phi_S + V_{\mathrm{bs}}} - \sqrt{\Phi_S}\right) && (1.6)\\
&- \frac{(V_{\mathrm{bi}}(T) - \Phi_S) + V_{\mathrm{ds}}/2}{\cosh(L/l_c) - 1} && (1.7)\\
&+ \alpha(V_{\mathrm{bs}}) \frac{\Phi_S T_{\mathrm{ox}}}{W + \Delta W} && (1.8)\\
&- k_{\mathrm{retro}} V_{\mathrm{bs}} + k_{\mathrm{halo}}(L)\sqrt{\Phi_S} + \Delta_{\mathrm{DITS}}(V_{\mathrm{ds}}, T) && (1.9)
\end{aligned}$$

where 1.5 computes the threshold voltage without substrate bias for a long channel device. Line 1.6 describes the body effect. The body effect results in an increase of the threshold voltage if $V_{bs} > 0$, thus if source is more positively charged than the body. This effect is important for transistor stacks, where the intermediate source-drain contact may be charged up and it is exploited by body-biasing (Section 1.4.4), where the body potential is deliberately biased. Term 1.7 introduces the drain induced barrier lowering effect as described in [12], and as shown in Figure 1.3 C: A high V_{ds} and a small L in relation to a characteristical l_c both induce a reduction of the effective threshold voltage. For extremely short channel lengths this effect dominates the threshold voltage. The term 1.8 describes the occurrence of a parasitic transistor with increased threshold voltage, formed at both ends of the printed transistor, thus increasing the average threshold. This term is the only minor contribution, introducing nonlinearity to the leakage transistor-width dependency for small-width devices. The three terms of Equation 1.9 describe the effect of non-uniform doping, as detailed in Section 1.4.1. From a mathematical point, these terms just introduce first order corrections to the

[3] The thermal voltage's equation has no deeper meaning, but is just the result of the deliberate choice of units in the MKSA (meter, kilogramme, second, ampere) system. Temperature and energy are naturally equivalent ($E = k_B T$), and for electrons, voltage also equals energy ($E = V \cdot q^- \Rightarrow V = E/q^- = k_B T/q^-$).

threshold varying terms above. The first term describing retrograde well is the counterpart to the body effect by reducing threshold with V_{bs} dependence. The second term from halo doping increases threshold voltage with sinking channel length, and the third term, describing drain induced threshold shift reduces the V_{ds} dependency of the DIBL-effect.

Punchthrough

Besides the subthreshold effect, which is usually dominating the drain-source leakage, the subsurface drain-induced barrier lowering effect (Punchthrough, I_{punch} in Figure 1.2) may occur at high V_{ds}. Due to the drain source potential, the drain and source region may be pushed closer to each other. Especially in short channel devices, drain and source may merge, which is then called a punchthrough. Recent transistors employ well engineering, to control the channel's extensions, but the doping concentration towards the bulk is usually lower. In such devices punchthrough occurs deeply below the surface. Majority carriers cross the source junction and may be collected by the drain, increasing the drain source leakage and effectively reducing the subthreshold slope. Halo doping (rf. Sec 1.4.1) reduces the effect of punchthrough. Punchthrough is not relevant as a source of power consumption, but has a technological impact, as it limits the maximal V_{DD}. Punchthrough is thus relevant for I/O cells and certain design techniques such as super-cutoff.

1.2.2 Gate leakage

The elemental principle of MOSFET devices is the isolation of the gate contact. In contrast to the subthreshold leakage, which is a necessary result of the exponential, but not infinite reduction of the working current, a significant current through the gate isolator should classically not be expected. Nevertheless, a vast fraction of the total leakage of sub-100nm systems is due to gate currents (see Figure 1.1). In this section, gate tunnelling, the major source of gate leakage will be described. The principles of tunnelling are briefly depicted followed by an approximative analysis of the tunnelling. Finally, the four mechanisms of hot carrier injection are presented.

Gate tunnelling

The dominant source of gate currents is gate tunnelling, which can again be subdivided by the current path, the origin of the carriers or the shape of the potential to be passed [1] (see Figure 1.4).

- From an electrical point of view, the possible current paths are directly through the oxide to source I_{gso} or drain I_{gdo}, or into the channel I_{gc}. From the channel, the current can then flow to source I_{gcs}, to drain I_{gds} or to bulk I_{gb}. Figure 1.4 presents these five possible gate-tunnelling current paths.
- From a solid-state physics point of view, the carriers passing the oxide may either be electrons or holes and they may either tunnel from the conduction-band or the valence-band. For all four, the basic tunnelling equations are identical. A difference between the tunnelling probabilities

of electrons and holes occurs because of the exponential decrease of the transition probability with higher oxide thickness. The probability also sensibly depends on the effective mass (which is larger for holes than for electrons) and the carrier energy, which is higher for carriers from the conduction-band.

- From a quantum mechanical point of view, tunnelling can be distinguished by the shape of the barrier. Depending on the gate and channel potentials, carriers may either completely traverse the entire oxide thickness through a nearly rectangular potential barrier (*direct tunnelling*) or tunnel into the oxide's conduction band through a triangular barrier of reduced thickness $T_{\mathrm{ox,eff}}$, called *Fowler-Nordheim tunnelling* (FN tunnelling) (see Figure 1.4). FN tunnelling will occur as soon as V_{ox}, the voltage drop across the oxide exceeds at least the barrier height for electrons of $\Phi_{ox} = 3.3V$. For typical sub-$100nm$ devices in normal operation conditions, FN tunnelling is negligible, but for flash memories, working at $> 10V$ in write and erase mode, the shallow barrier shape and the reduced tunnelling distance enables tunnelling even through very high oxide.

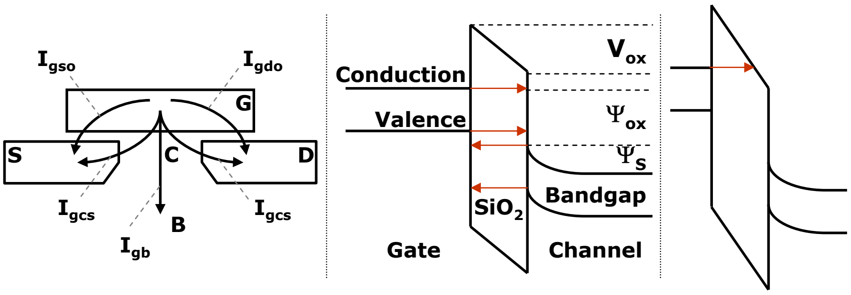


Figure 1.4: **Left:** Current paths in MOSFET transistors. **Middle:** Direct tunnelling of electrons and holes from conduction and valence band. **Right:** Fowler-Nordheim tunnelling from the gate's conduction-band to the oxide's conduction-band for extreme V_{ox}.

Tunnel effect in MOSFET transistors

The tunnel effect is a well known quantum-mechanical phenomenon: The Schrödinger equation $H\Phi = E\Phi$, where the Hamiltonian H contains the second-order spatial deviation, as well as the underlying potential V directly prohibits an infinite second-order spatial deviation (an infinitely sharp bend) of the wave-function Φ, even if the potential itself is not continuous, as it would result in an infinite energy density (see Figure 1.5). In case of a potential barrier, this means, that the wave-function of a free electron exponentially decreases if its energy is lower than the barrier height, but is not zero outside the barrier. A quantum-mechanical particle has always a small probability of passing a barrier, higher than its total energy. The particle is never *inside* the barrier, it is not traversing the barrier, it may just occur outside the barrier.

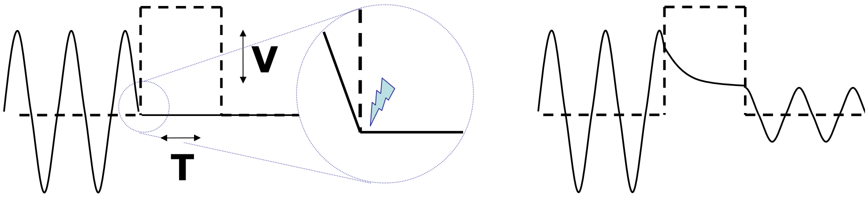


Figure 1.5: The tunnel effect. **Left**: There is no differentiable transition between the wavefunction of a free electron and a zero probability. **Right**: As the Schrödinger equation demands differentiable wavefunctions, there is a finite transmission probability.

For the gate-oxide-channel intersection, the shape of the conduction-bands of the semiconductor-isolator-semiconductor stack is very close to a rectangular barrier (see Figure 1.4), thus the transition probability can be approximately described by standard quantum-mechanical equations [34]. In [31], the tunnelling current to all three terminals is developed. In a simplified form, it results as

$$I_{gx} = k_x W L_{eff,x} T_{ox}^{-2} V_{gx} V_Y e^{-\beta_x T_{ox} \left(\alpha_{0,x} + \alpha_{1,x} V_{ox} + \alpha_{2,x} V_{ox}^2 \right)}, \tag{1.10}$$

where x is the respective terminal (source, drain or bulk), k_x, $\alpha_{i,x}$, and β_x are constants, depending on the terminal x. Gate-source and gate-drain leakage are described identically. As tunnelling probability gives the charge transfer per electron trying to pass the barrier, it has to be multiplied with an area to compute current from current density. Gate tunnelling always occurs over the entire width W of a transistor, but for gate-drain and gate-source leakage only over a small fraction of the channel length, denoted $L_{eff,x}$. Another asymmetry between gate-source/drain and gate-bulk leakage is V_Y, which is V_{gx} for $x =$ source/drain but V_T (thermal voltage) for $x =$ gate. Usually $V_T << V_{gx}$ for $x =$ source/drain, thus tunnelling to bulk is much smaller and temperature dependent.

The behaviour as described by Equation 1.10 can be observed, when plotting the gate current I_{gX} versus the respective voltage V_{gX} for different oxide thicknesses and temperatures as presented in Figure 1.6: Gate leakage is exponentially depending on oxide thickness as well as potential difference. Gate to bulk tunnelling is almost two orders of magnitude smaller than gate to source or gate to drain leakage and only gate to bulk leakage is showing a significant thermal dependency.

Hot carrier effects

In comparison to other isolators, silicon oxide, the recent MOSFET gate dielectric, has a comparatively small bandgap of $3.3eV$ @ $300K$. As soon as a carrier inside the channel manages to obtain this energy, it can enter the oxide's conduction band and carry a current towards the gate without the exponential decrease per oxide thickness. Especially in short channel devices, the electric fields may become large enough to accelerate the carriers so that they become *hot*, what means that their kinetic energy exceeds the oxide bandgap. Holes from the valence band additionally need to overwhelm the silicon bandgap ($1.12eV$ @ $300K$) and have a larger effective mass. Thus hot carrier injection is much stronger in NMOS devices.

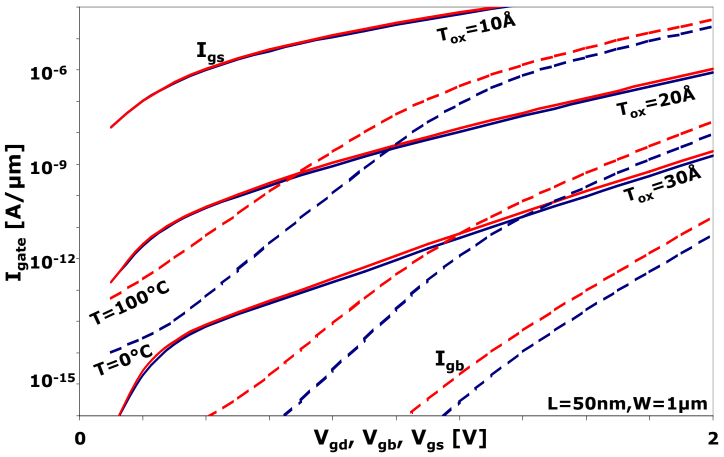


Figure 1.6: Gate leakage of the $45nm$ predictive technology model NMOS device [35]. As source and drain are symmetrical, I_{gs} and I_{gd} are identical. Gate source and gate bulk leakage were obtained for two temperatures and three oxide thicknesses.

According to [36], hot carrier effects can be sub-divided into four effects:

- **Drain avalanche hot carrier injection (DAHC):** When switching, for $V_{ds} > V_{gs} > 0$ (this condition is typically reached when switching), a high field occurs at the drain. The accelerated carriers collide with the silicon lattice, generating electron-hole pairs (*impact ionisation*). The resulting electron-hole pairs may have sufficient energy to becoming *hot*, thus reaching the oxide conduction band.

- **Channel hot electron injection (CHE):** If drain and channel potential both are significantly higher than source, carriers traversing the channel are attracted by the gate potential. Some of them may reach the channel-oxide junction before reaching drain.

- **Substrate hot electron injection (SHE):** In case of extreme biasing of the substrate, carriers from the substrate are accelerated towards the gate if V_{gb} is large. As they move upwards, they gain further energy in the higher field close to the surface.

- **Secondary generated hot electrons injection (SGHE):** The SGHE equals the DAHC effect. But additionally, while the majority carriers move towards the gate oxide, the minority carriers move towards the substrate creating a field which is driving the majority carriers towards the gate.

1.2.3 Junction leakage

This section will present the mechanisms enabling drain-bulk and source-bulk leakage. Classical diode currents: electron drift, diffusion, and electron-hole generation are presented. Afterwards, a third mechanism called band to band tunnelling (BTBT), emerging at high doping gradients is described.

Finally, it will be explained how gate induced drain leakage (GIDL) can amplify the BTBT to a significant contributor.

Diode currents

Equation 1.11 (Shockley ideal diode equation) gives the current of a pn-junction of an ideal diode due to drift (under the electric field V_{app}), diffusion and thermal recombination generation. The proportionality factor I_0 itself is proportional to the junction area, the inverse doping concentration $1/N$ and the temperature to the power of three.

$$I^{pn} = I_0 \left(\exp \left(\frac{V_{app}}{n \cdot V_T} \right) - 1 \right) \tag{1.11}$$

Additional junction currents can be carried by electron hole generation (Shockley-Read-Hall currents). Electron-hole pairs are generated, they drift due to the electric field and recombine elsewhere, thus carrying a current. Electron hole generation depends on the carrier concentration and on the recombination rate (the inverse carrier lifetime).

For transistors with heavily doped p and n regions, band to band tunnelling currents may occur, as soon as the potential drop across the junction gets larger than the bandgap (see Figure 1.7 left). Electrons from the valence band of the p-region can directly tunnel to the n-side's conduction band as this is energetically lower due to the field (the band gap is smaller than the potential difference between p- and n-side). This effect is analytically described in [1].

All these effects of drift, diffusion, recombination and band-to-band-tunnelling only carry marginal currents in the order of several pico-amperes per micrometer transistor width in recent and upcoming technologies. Both, the gate leakage and source drain leakage are usually orders of magnitude larger (see Figure 1.7 right).

Gate induced drain leakage (GIDL)

Having very steep doping gradients, electrons from the valence band of the p-doped side can directly tunnel through the bandgap into the conduction band of the n-doped side. Energetically, this effect can occur as soon as the potential difference across the pn-junction (external field plus build-in potential) is larger than the band gap. Practically, the pn-junction doping gradient is not steep enough for significant band-to-band-tunnelling (BTBT). The tunnelling effect is extremely dependent on the distance, an electron has to tunnel through; and even in recent $45nm$ technologies, the width, in which the potential across the pn-junction exceeds the bandgap, is too large for significant tunnelling.

If the gate voltage is low and the drain voltage high, an extreme field is set up across the gate oxide. For the drain pn-junction directly below the gate oxide, this high field reshapes the bands and leads to a reduction of the tunnelling distance and thus an extreme increase of the tunnelling probability. As can be seen in Figure 1.7, below a V_{dg} of approximately $1V$, the junction leakage is clearly dominated by the thermally dependent drift, diffusion and electron-hole generation currents.

Above, the BTBT dominates the overall current showing a high oxide field (potential drop and oxide thickness) dependence.

This effect of gate induced drain leakage also influences the off current of a transistor, as presented in Figure 1.3 graph D. From a physical view, gate induced source leakage (GISL) is absolutely equivalent, but technically, gate-source voltage is usually not in a GISL condition[4].

Besides GIDL, also backward body biasing is able to boost the BTBT leakage to a really significant contribution. If the bulk potential is negative ($V_{DB} > V_{DD}$), the band shift at the pn-junction becomes smaller due to the additionally applied voltage. Thus the tunnelling distance reduces and the tunnelling probability exponentially increases.

[7] gives an excellent analytical description and prediction of the three dominant sources of leakage (subthreshold off-current, gate tunnelling and gate induced drain leakage).

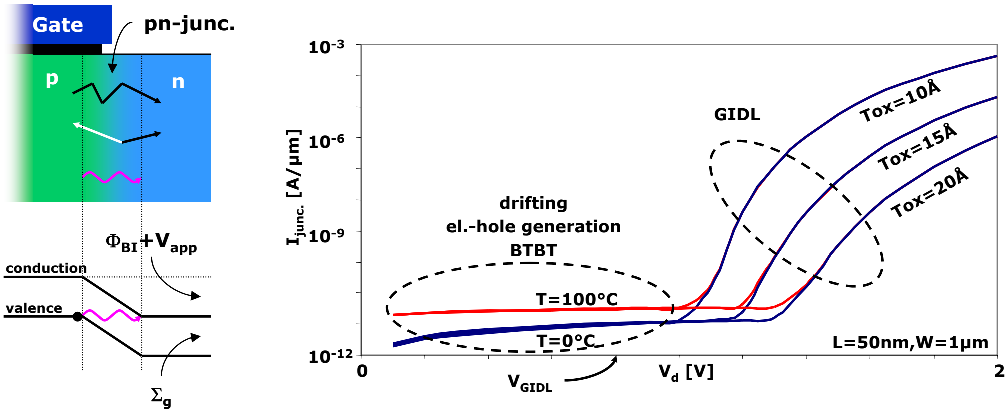


Figure 1.7: **Left:** Junction currents are carried by drift, diffusion, electron-hole generation and band-to-band-tunnelling (BTBT). **Right:** Gate induced drain leakage (GIDL) can increase the BTBT current to a significant part of the overall leakage. With shrinking oxide thickness, this effect rises exponentially.

1.3 Leakage factors

In this section, the factors which are influencing the various leakage currents, and which thus have to be regarded for the estimation are described. At first, the most important transistor geometry parameters (*StPa*) are reviewed. Leakage currents are extremely sensible to a variation of these parameters resulting in a non-trivial distribution from transistor to transistor up to a die-wise and even fab-wise spread. The variation dependence of the leakage currents and the correlation towards the performance are analysed in the following section.

[4] When a locking transistor is in parallel to a conducting one, the locking transistor may be subjected to both, GIDL and GISL if drain voltage is high. The smallest gate, where GISL may occur is the $!(a|b\&c)$ OAI21 gate in the a-NMOS, when $a = 0, b = 1, c = 0$ if the c-NMOS is closer to GND in the circuit.

Afterwards, the macroscopic leakage factors ($DyPa$) temperature and voltage are discussed. The final part presents the data dependence of system components from a high abstraction level view.

1.3.1 Variation parameters

As discussed in Section 1.2, for recent technologies and for the upcoming ones at least down to the $32nm$ node, the dominating leakage mechanisms are subthreshold leakage, gate tunnelling, and gate induced drain leakage (GIDL). Among the transistor parameters, which are known to diverge due to process variation, three core parameters [11] can be identified, having significant impact on one of the dominating leakage sources.

While being discussed, the various influences are also classified by their effect from a transistor and the system view. If an effect results in a parameter deviation which is identical for all transistors in a certain region, this effect, and its variation is called globally correlated. If each transistor is affected individually the variation is called local. Local variations are subdivided into deterministic and statistic variations. A local deterministic variation of an transistor A will result in an identical deviation for the transistor A in each system instance, but will have no influence on transistors in the proximity of transistor A within one instance (summarised in Table 1.1).

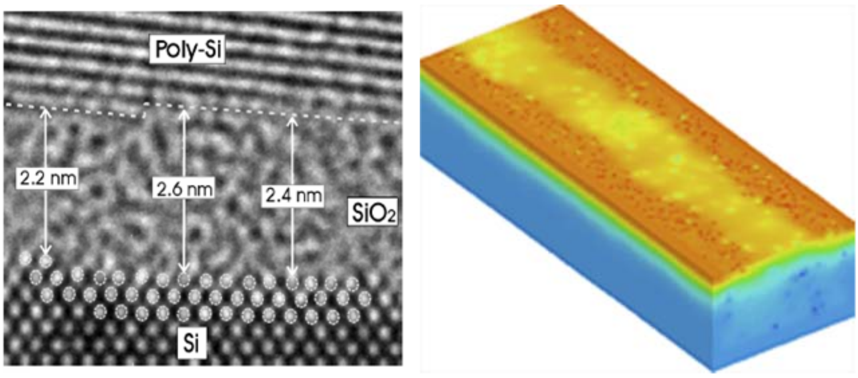


Figure 1.8: **Left:** Oxide thickness variation: Oxide layers of recent technologies are only a few molecule layers thick. Single atom mesh errors lead to significant oxide thickness variation. (C. Visweswariah) **Right:** Visualisation of the effect of line edge roughness and doping concentration variation on an NMOS transistor. (A. Asenov)

Channel length

There are several reasons for a variation of the transistor's channel lengths, each leading to a characteristic distribution. In the following, three variation mechanisms and their resulting influence on the channel length are discussed. A mask-misalignment can result in a deterministic, global variation with an identical delta for all transistors of an instance but a varying influence per system. Interference effects in the optical proximity corrected (OPC) sub-wavelength lithography can lead to a local deterministic geometry (especially channel length) variation. This variation is identical for the same transistor in each system instance. Line edge roughness (LER) is the result of a certain granularity of materials as the interconnect metal, the poly-silicon, or the photo-resist, which is used for production (rf. Figure 1.8). Due to this natural granularity of matter, all produced structures show a roughness in the order of several nanometres [37]. With shrinking device sizes, the line edge roughness gains influence, leading to a completely statistical local distribution of the channel length of each transistor.

As detailed in Equation 1.7, a variation of the channel length has a direct influence on the threshold voltage due to the drain induced barrier lowering (DIBL) effect. Thus both, subthreshold leakage and device performance are modified. All kinds of correlation presented above can also be found within the subthreshold leakage and performance distribution. As both variations result from only one parameter, they show a strong correlation.

Oxide thickness

For the $65nm$ technology node, the ITRS reports an oxide thickness of about $1.1nm$[5] for planar bulk. Silicon oxide molecules of the gate oxide have a diameter of $0.36nm$. Thus, the oxide layer is in average 3.5 molecules thick. This directly results in two sources of variability. At first, 3.5 layers can of course be only an average, as molecules always come in integer quantities. Thus, the oxide layer thickness will at least vary between three and four layers, resulting in ±14% variation. Additionally, in such small dimensions, single atom effects can dominate the resulting oxide thickness. As can be seen in Figure 1.8, a single atom layer stair in the silicon crystal leads to statistical per transistor oxide thickness variations.

One typical source of deterministic global oxide thickness variation occurs during production of the oxide: Several prepared wafers are mounted on an oxidation boat and then heated up in an oxygen rich environment. The thickness of the resulting oxide layer is a function of the temperature and the oxygen concentration. Due to the large dimensions of the wafers and their small distance towards each other, the central part of each wafer is typically a little colder resulting in a smaller oxide thickness for the wafer centre.

As can be seen in Equation 1.10, the gate tunnelling probability is exponentially depending on the thickness of the oxide. A single molecule ($0.36nm$) difference in oxide thickness can easily lead to a factor ten variation in gate tunnelling. This extreme sensible dependency makes oxide variations significant, even though the oxide of recent MOS devices is controlled with nearly atomistic accuracy. Each oxide thickness variation will also lead to a subthreshold leakage variation because threshold voltage and subthreshold slope both depend on the oxide thickness.

[5] An equivalent oxide thickness of $1.1nm$ for oxi-nitride gates results in a mechanical thickness of $1.1nm \cdot k_{SiNO}/k_{SiO_2} = 1.1nm \cdot 4.2/3.9 \approx 1.2nm$.

Channel doping

The channel doping is realized by randomly *shooting* doping ions into the silicon lattice. The channel doping concentration developed from $\approx 0.5 \cdot 10^{18}/cm^3$ for the $180nm$ technology up to $\approx 2 \cdot 10^{18}/cm^3$ for the $45nm$ technology by a factor four over these five technology generations [35, 38]. In the same time, the active channel volume scaled down roughly by a factor of 60. A $45nm$ technology generation channel of $L_{active} = 20nm$, $X_{dep} = 30nm$, and $W_{typ} = 100nm$ has a channel volume of $60000nm^3 = 6 \cdot 10^{-17}cm^3$ and has thus 120 doping atoms in average.

With dopant counts around 100, the transistors will show significant fluctuation. The exact number of dopants is statistically deviating per transistor resulting in a concentration variation and thus a per transistor threshold voltage variation. With even lower dopant counts, the exact position of each doping atom will also contribute to the variation of the threshold of each transistor (rf. Figure 1.9).

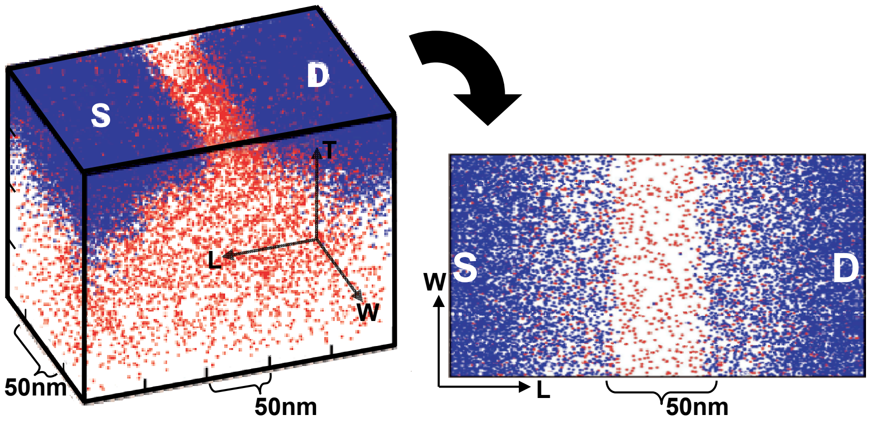


Figure 1.9: Distribution of doping atoms in a $45nm$ transistor. Especially the channel doping atoms show a high local variability of the doping distribution. [39]

In [40], the threshold voltage of $50nm$ devices was computed using a 3D atomistic model. For a doping concentration of $2 \cdot 10^{18}/cm^3$, the threshold voltage shows a nearly Gaussian distribution with $255mV$ mean and $43mV$ standard deviation due to random deviations of the dopant count and position. This threshold distribution has two major effects:

On the one hand, the average subthreshold leakage is increased. Assuming a subthreshold slope of $80mV$ per decade, a threshold variation of $+43mV$ will lead to a factor three reduction in leakage and with $-43mV$, the leakage is increased by a factor of three. As the average of $3x$ and $x/3$ is larger than x, the average leakage rises. More accurately in this example, due to a Gaussian distribution

$$p\left(V_{th}\right) = \left(2\pi\sigma^2\right)^{-1/2} \exp\left(-\frac{\left(V_{th} - \mu\right)^2}{2\sigma^2}\right), \tag{1.12}$$

the expectation value $E\left(I\left(V_{th}\right)\right)$ of the leakage current

$$I\left(V_{th}\right) = I_0 \exp\left(-V_{th}/n(T)\right) \tag{1.13}$$

will result as

$$E\left(I\left(V_{th}\right)\right)=\int_{-\infty}^{+\infty} I\left(V_{th}\right)\cdot p\left(V_{th}\right)\,dV_{th}=I\left(V_{th}\right)\cdot\exp\left(\frac{\sigma^2}{2n^2}\right). \tag{1.14}$$

Assuming $n(T=300K)=80mV/\ln(10)\approx 35mV$, this effect results in a factor 2.15 increase of the average subthreshold leakage just because of the deviation.

On the other hand, a $17\% = 43mV/255mV$ standard deviation of the threshold voltage will lead to a 6 sigma probability of having zero threshold voltage, thus a transistor failure. For an SRAM memory of $1MB$ size without redundant error correction circuitry, this will result in a yield loss of roughly 4% [41]. The yield loss rises to 30% for 5.5 sigma, and 98% for 5 sigma.

Besides the random dopant fluctuation, there can be global, deterministic deviation due to any disturbance (especially temperature and timing deviations during doping) of the doping mechanism leading to a change in average dopant count. Finally, the well proximity effect introduces a source of local, deterministic variation of the doping concentration. The well proximity effect is an artefact, resulting from using high-energy implanters to produce retrograde well (see Section 1.4.1) structures. In order to form a retrograde well, doping atoms are accelerated getting high enough energy to pass the channel in order to form a high doping concentration below this channel. The dopants may be scattered at the edges of the photo-resist, loosing energy and thus increasing the doping concentration inside (not below) the channel. For each transistor, the threshold voltage is thus deterministically deviating with the proximity to other structures [42].

A deviation of the doping concentration has a direct influence on the transistors threshold voltage and thus on the subthreshold leakage. Additionally, the gate induced drain leakage (GIDL) - being a tunnel effect - is extremely depending on the doping gradient. Thus, the doping concentration also influences the junction leakage (see Section 1.2.3).

correlation	channel length	oxide thickness	channel doping
global deterministic	mask misalignment	wafer oxidation	timing deviation
local deterministic	OPC artefacts	-	well proximity
local statistic	LER	single layer effects	random dopant fluct.

Table 1.1: Taxonomy of correlation for the three transistor parameters with highest influence on leakage. Different correlation types will have to be treated separately for the purpose of leakage estimation. Besides a second order effect, local statistic variations will average out from a system level view. In contrast, global deterministic variations will not average out per system instance and have to be explicitly regarded. Global variations will lead to a distribution of the overall system leakage and performance. Local deterministic variations will lead to an absolutely deterministic deviation for all systems and can thus be treated as an absolute shift of certain design metrics.

1.3.2 Variation dependence

A common classification of the effects described above, which is frequently used in literature, is by their variability from a system, wafer, and even fab level view [43]: Depending on the exact cause of a global deterministic variation effect, it can be

- **fab-to-fab variation:** Integrated device manufacturers producing the same design in more than one site report different factory averages for system parameters (typically due to differences in their production devices).
- **lot-to-lot variation:** All wafers in a lot see the same variation due to process variations (as timing, temperature, chemical concentration)
- **wafer-to-wafer variation:** Due to variations when processing a single wafer (as coating or polishing) the entire wafer varies
- **die-to-die variations (centre-to-edge variations):** Parameters have a deterministic radial deviation depending on the distance form the wafer centre. For instance, the oxidation process leads to such a typical radial variation across the wafer. Die-to-die variation may also occur in non-radial shape, such as gradients.
- **inter-component correlation:** Two variables do not vary independently, but show a certain correlation. For instance, IBM [44] reports a high correlation between low-field mobility ($U0$) and the DIBL factor ($ETA0 = \Delta V_{th}/\Delta V_{DD}$), while the mobility and the drawn channel length are nearly completely independent from each other.
- **within-die gradients:** Spatial parameter variations on a wafer with such a short range (or high amplitude), that each die sees parameter gradients.
- **random intra-die variation:** Concerning the taxonomy from Section 1.3.1, all items above belong to the deterministic global variations even though some of the effects are completely statistical[6]. All local, statistical per transistor variations are regarded as random intra-die variations. The remaining class of the Section 1.3.1 taxonomy, the local deterministic variations cannot be seen in this scheme as they will always result in a deterministic deviation for all systems on all wafers from each fab. Instead, deterministic local variations will cause a deterministic *simulated versus measured* deviation.

One of the first reports on variation of leakage and performance was done in 2000 [45], quantifying the impact of variations. For the $70nm$ technology, Intel reported an off-current deviating from $1nA/\mu m$ to $1\mu A/\mu m$. Intel discovered a nearly perfect correlation between the drive current and the logarithmic off-current (see Figure 1.10). This correlation can be explained by a gate length variation resulting in a threshold voltage variation. Drive current (and thus later system performance) depends

[6] A process timing deviation of a lot for instance will result in a random lot-to-lot variation from a factory view, but from a transistor view, it results in a deterministic global deviation of all transistors in a system.

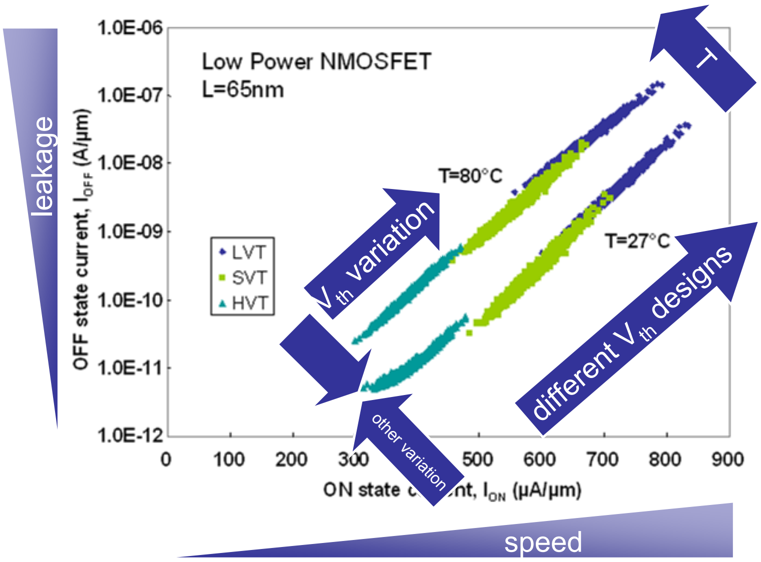


Figure 1.10: Correlation between performance and leakage. Besides the strong temperature and threshold voltage dependency, the devices of one technology at one temperature show a nearly perfect linear correlation between performance and logarithm of the leakage current.

almost anti-proportionally on the threshold voltage while the subthreshold currents is decreased exponentially with a rising threshold voltage [3]. In a later work on the $90nm$ technology under production, Intel reported a reduced leakage variability of still a factor of 20 [23].

1.3.3 Thermal dependence

All leakage effects, which base on solid-state physics, as subthreshold leakage and diode currents, need a certain thermal activation, and thus are extremely thermally dependent. In contrast, leakage effects, being caused by the tunnel-effect, as gate leakage and gate induced drain leakage (GIDL) show nearly no thermal dependency.

As detailed in Section 1.2.1, especially the subthreshold current is significantly depending on the temperature. Using

$$I_{ds}\left(V_g\right) = I_0 \cdot e^{-\alpha \frac{V_{th}}{T}} \tag{1.15}$$

and assuming, that the subthreshold slope is $80mV$ at $350K$ temperature[7], α numerically results as $\alpha \approx 10K/mV$. Such a device would see twice the leakage, when increasing the temperature by roughly $38K$ for a $V_{th} = 250mV$ device and $26K$ for a $V_{th} = 350mV$ device. The subthreshold slope is already very close to its fundamental theoretical limit of $63mV$ per decade [30], and the working temperatures will also remain between $300K$ to $400K$, thus the thermal variability mainly depends on the threshold voltage for all bulk CMOS devices.

[7] meaning, that the channel current is reduced by a factor of ten for each $80mV$, the gate voltage is below the threshold voltage

The temperature has a strong influence on the leakage. Vice versa, all leakage currents finally dissipate to thermal energy. This effect is called electro-thermal coupling. Electro-thermal coupling will increase the leakage power because the warmer system suffers from more leakage and warms up gaining even more leakage. In the worst case, when doubling the leakage increases the temperature more than $26K$ (for the example of a $350mV$ threshold), the leakage is increased by more than a factor two, resulting in a runaway-situation.

In terms of prediction of the electro-thermal coupling, [21] suggests a chip-wide package dependent thermal model in a feedback loop with a thermally dependent leakage model. [46] presented an improved thermal model, showing how sensibly temperature computation depends on small deviations as the thickness of the thermal interface layer or the thermal resistance of the package.

Intel reported [23] a junction temperature distribution with a difference of $50K$ between the hottest and the coolest place on their recent microprocessors. That means, that the subthreshold leakage will be approximately four times higher at the hot spots. For high accuracy in a leakage estimation framework, such an enormous effect cannot be ignored. A reliable leakage model has always also to predict the thermal distribution, and not only the average junction temperature.

An interesting application of the thermal behaviour of the leakage was discussed in [47]: Obviously, when rising the temperature increases the leakage exponentially, actively cooling the systems will reduce the subthreshold leakage by orders of magnitude. While reducing the leakage by cooling, the electron mobility [48] which behaves like

$$\mu_0 \left(T\right) = \mu_0 \left(T^*\right) \cdot \left(\frac{T^*}{T}\right)^{1+\epsilon}, \tag{1.16}$$

where ϵ is between 0.1 and 0.5 for all PTM technologies, is increased. As a result, active cooling increases the performance[8] (which depends on the electron mobility) while reducing the subthreshold leakage, at the same time.

1.3.4 Voltage dependence

Besides the direct, and obvious dependence of the leakage power on the supply voltage as $P_{leak} = I_{leak} \cdot V_{\mathrm{DD}}$, the leakage current itself also depends on the supply voltage due to several effects (see Figure 1.11):

- With the drain induced barrier lowering (DIBL) effect, the threshold voltage is biased proportionally to the supply voltage (see Equation 1.7). An increased supply voltage $V_{\mathrm{DD}} = V_{DD}^0 + \Delta_V$ will reduce the threshold voltage to $V_{th}^0 - \beta_{DIBL}\left(L_{ch}\right) \cdot \Delta_V$. The lower threshold voltage will then lead to a subthreshold current of

$$I_{sub}\left(V_{\mathrm{DD}}\right) = I_{sub}\left(V_{DD}^0\right) \cdot e^{\Delta_V \cdot \alpha_{slope} \cdot \beta_{DIBL}/T}. \tag{1.17}$$

  With lower channel length, β_{DIBL} increases, making the DIBL effect, and thus the leakage voltage dependency more important for smaller technology nodes. For the predictive technology

[8] For $65nm$ and below, the temperature-performance dependencies are more complex, as discussed in Section 2.6

model, the factor $\alpha_{slope} \cdot \beta_{DIBL}/T$ at $T = 350K$ computes to $1.44V^{-1}$ for $65nm$ and $2.01V^{-1}$ for $45nm$[9]. A V_{DD} increase of $192mV$ ($65nm$) and $137mV$ ($45nm$) will thus result in a 10% leakage current increase due to DIBL.

- The gate oxide carrier tunnelling probability exponentially depends on the field across the oxide, and the gate current depends linearly on this probability and quadratically on the gate voltage (see Equation 1.10). As a result, the gate leakage is extremely voltage dependent as presented in Figure 1.6. For the $45nm$ technology, an increase of roughly $V_{\mathrm{DD}} + 100mV$ results in a doubled gate leakage, nearly independently form the transistor geometry, especially the oxide thickness.

- The gate induced drain leakage (GIDL) effect can be explained by a V_{dg} dependent thinning of the pn-junction width and a resulting increased band to band tunnelling (BTBT) probability. BTBT is like each tunnel-effect extremely tunnel distance dependent. Thus, a V_{DD} increase will result in a higher potential and a smaller tunnel distance. Increasing V_{DD} from $1.2V$ to $1.3V$ will increase the junction current by a factor 82 ($45nm$) and 211 ($65nm$).

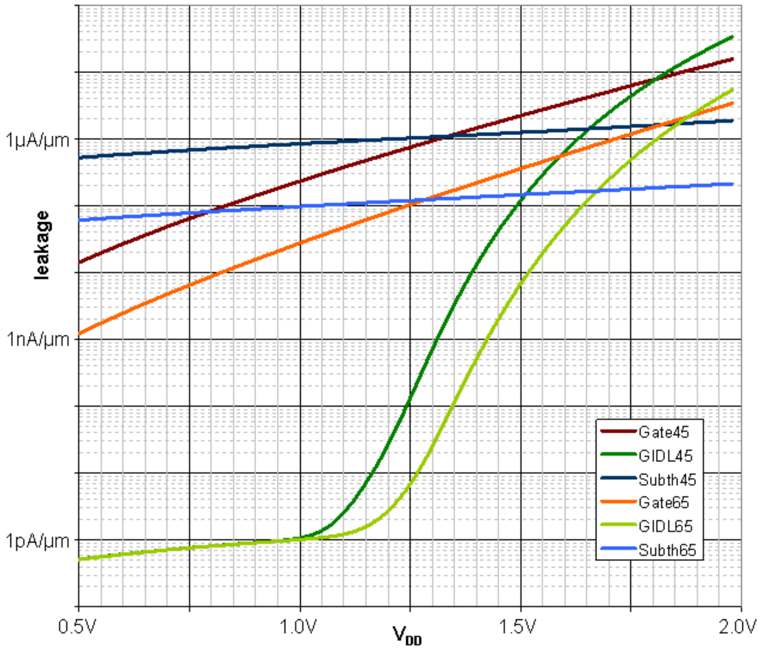


Figure 1.11: Simulation of the subthreshold leakage, gate tunnelling, and gate induced drain leakage for the PTM $45nm$ and $65nm$ technology at $T = 350K$. The subthreshold leakage has the highest impact for low V_{DD} but the lowest increase with rising voltage, and vice versa for the GIDL effect.

[9] The DIBL factor β_{DIBL} is $50mV/V$ for the $65nm$ and $70mV/V$ for the $45nm$ technology.

In terms of leakage estimation, the high leakage sensibility on supply voltage variations makes very accurate prediction of the voltage level reaching the gates necessary, which may diverge from the nominal supply voltage by static voltage drops (and ground bounces) in the middle of the die and by dynamic voltage drops due to a finite RC delay of the supply lines. Intel [23] reported in 2003 a ±7% variation of the supply voltage and in 2008 [49] ±5%. The back-coupling between leakage, increasing the IR-drop and IR-drop, reducing the leakage is called electro-electrical coupling. In [4], a combined approach is presented to compute the temperature and supply voltage map and to feed the leakage and dynamic power models with this data, thus accurately regarding electro-thermal and electo-electrical coupling.

1.3.5 Data dependence

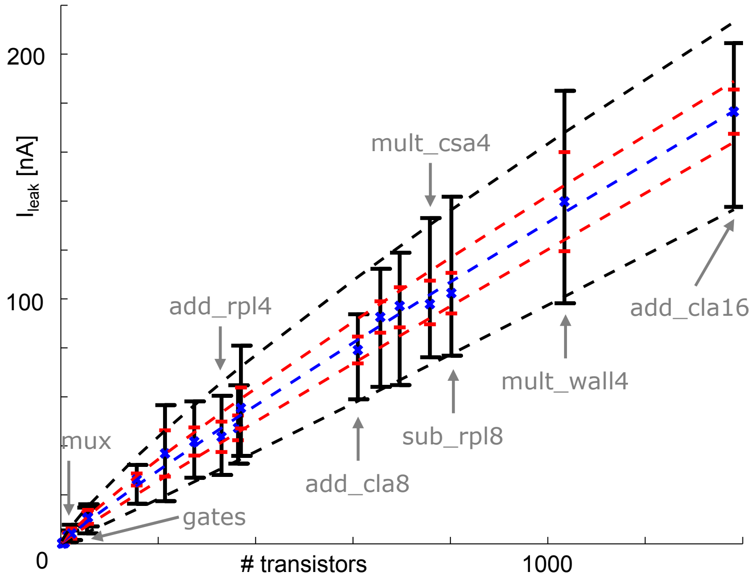


Figure 1.12: Data dependency of RT components with respect to the number of transistors obtained by brute force simulation. **Black line**: absolute minimum and maximum leakage, **Blue cross**: average leakage over all states, **Red ticks**: 1 σ deviation

A single gate in the 70nm PTM technology can have more than a factor 100 variation in leakage depending on the state, as presented in Table 1.2. Nevertheless, among all leakage factors, discussed in this Section 1.3, the data dependence of entire RT components has the lowest impact on the leakage distribution. Figure 1.12 presents the data dependence of single gates and entire RT components. Different structures have been synthesised and converted into SPICE netlists using the 70nm PTM transistors (see Section 4.3). This older analysis was done using the gate analysis flow as described in [27]. While there is obviously an almost linear behaviour between the transistor count and the average leakage, the minimum leakage grows over-linearly and the maximum leakage under-linearly. As a result, the relative min-max distance is reduced with increased number of transistors. The six transistor NOR3 has an $I_{max}/I_{min} = 112$ and a $\sigma_I/\mu_I = 228\%$ but an average 1000 transistor component only has an $I_{max}/I_{min} = 1.7$ and a $\sigma_I/\mu_I = 8.1\%$. Other researchers came to similar results for larger circuits [24], analysing minimum and maximum leakage for the ISCAS85 benchmarks: The min-max relation was between 1.48 for the $C880$ circuit and 1.12 for the $C7552$ circuit.

abc	000	001	010	011	100	101	110	111
$V_1[mV]$	1200	1200	1200	1200	229	1161	1164	1183
$V_2[mV]$	1200	1200	205	1161	205	1161	170	1144
$I_{sub}[pA]$	15620	880	760	240	750	240	230	140

Table 1.2: State dependent subthreshold leakage of a NOR3 gate in $70nm$ PTM transistors and $W/L = 3$ for NMOS and $W/L = 6$ for PMOS. V_1 is the potential between PMOS a and b transistor, V_2 is the potential between PMOS b and c transistor influencing the body effect. In the 000 state, the NMOS transistors are leaking, the 010 and 100 state show lower leakage than the 001 state due to the body effect.

Even though, the influence of the data is very weak at RT level, for a bottom-up modelling approach[10], the extreme data dependency of single gates must be regarded on gate level and can be abstracted when moving towards the RT level modelling. Besides this work, also the Groups of Dennis Sylvester [18, 19] and Kaushik Roy [25] base their estimation approaches on a per transistor or per gate data dependent description.

1.4 Leakage optimisation methodologies

In sub-$100nm$ technologies, leakage related problems are based on the standby current's exponential dependency on both, parameter scaling and parameter variation. Countermeasures, reducing the static current and its spread can be classified into three categories: First, improved devices can reduce leakage sources by orders of magnitude. Secondly, performance and leakage can be traded off for conventional technologies using slow technology for uncritical circuits. And thirdly, power management implements and controls circuits, where leakage and performance can be tuned at runtime. Section 1.4.1 will thus focus on design of new devices, Section 1.4.2 reviews leakage avoidance techniques on gate level, and Section 1.4.3 presents techniques above circuit level. Finally, Section 1.4.4 reviews leakage management techniques and Section 1.4.5 dedicated memory techniques.

1.4.1 Device level

Well Engineering

A sub-$100nm$ device with a uniformly doped channel would be subjected to severe short channel effects, making subthreshold leakage vary with geometry and field variations by several orders of mag-

[10] As will be detailed later, the model being presented here is built in a bottom up manner.

nitude. Thus, recent devices employ well engineering (non uniform doping) to flatten the $V_{\mathrm{th}}(L_{\mathrm{ch}})$, $V_{\mathrm{th}}(V_{\mathrm{ds}})$, and $V_{\mathrm{th}}(V_{\mathrm{bs}})$ dependency, as described in Equation 1.9 (see Figure 1.13 for an overview). Retrograde well (increased doping concentration below the channel towards the substrate) introduces a negative V_{bs} dependent threshold bias, reducing the body effect [31]. Halo doping (increased doping concentration at the channel's pn-junctions) introduces V_{th}-rollup, slightly *increasing* the threshold at low channel length, and thus reducing the DIBL effect for short channel devices. Pocket implant (increased doping concentration at the pn-junction, shielding source and drain) is reducing punchthrough probability at high V_{ds}. Shallow source/drain extensions (SDE) (shallow extensions of source and drain below the gate and shielded by the pocket implant) improve the series source drain resistance of the devices. The driving strength of these devices is higher without increasing short channel effects. Well engineering as described here, is well known and available in recent technologies since 2002.

Mobility Enhancement

In strained silicon devices, the silicon lattice is stretched or squeezed to increase carrier mobility, thus increasing the on current without influencing the off current. By the introduction of strained silicon, either performance can be increased or leakage can be reduced. Strained silicon devices became available 2004. Next improvements in carrier mobility will be obtained by strained silicon in insulator (SSOI), expected for the end of this decade and unstrained silicon-germanium on insulator (SGOI) [50] expected much later (for the $< 22nm$ technologies) [51].

High-k Dielectric

As high C_{ox} is crucial for a high $I_{\mathrm{on}}/I_{\mathrm{off}}$ ratio at low threshold voltage, but lowering T_{ox} is limited due to tunnelling, higher insulator dielectricity is needed. Currently, dielectricity is slightly increased from 3.9 for SiO_2 to 4.2 for oxi-nitride ($SiNO$). [52] presents an analysis on tunnelling probabilities in SiO_2 and $SiNO$. The introduction of real high-k insulator is one of the largest changes ever made in MOSFET production. Instead of introducing single atoms to the crystalline Silicon grid, a completely new Hafnium (HfO_2) structure with a different lattice size is epitaxial grown over the channel. The poly-silicon gate can now be replaced by a metal gate. A poly-silicon gate has the additional disadvantage of forming a parasitic thickness enlargement by a charge shift at the gate-insulator interface, reducing the capacitance without reducing tunnelling probability. Hafnium based high-k devices are available since 2008.

UTB Fully Depleted SOI

In conventional SOI, the transistor is formed in a top-layer which is isolated by a buried oxide layer from the substrate. When scaling down transistor dimensions, also the active SOI layer is reduced in thickness. Ultra thin body (UTB) SOI denotes devices having less than $20nm$ SOI layer thickness. In fully depleted SOI devices, the SOI layer is thinner than the depletion width X_d [53]. One of the main disadvantages of UTB devices, a high source/drain resistance, can be reduced by elevated source/drain structures created layer by layer by selective epitaxial growth (see Figure 1.13). UTB devices have several advantages compared to bulk CMOS devices. At first, they work with an undoped channel, thus random dopant fluctuation (see Section 1.3.1) one of the most severe variation effects for $45nm$ devices and below, is removed for good. Additionally, the electrostatic channel control is much higher. [53] reports subthreshold slopes below $70mV$ per decade for a dual gate UTB device. Thus, the subthreshold leakage will be significantly lower than in bulk CMOS devices of comparable performance. And finally, UTB devices are free of bulk effects as latch-up and GIDL leakage (this is a property of all SOI devices).

Dual Gate

In order to further increase the electro-static control, the channel can be controlled from two or even more sides to approach the fundamental slope limit. In FinFET devices, the channel stands upright (like a fin), and is controlled from the front and back side. If the channel width (now height) has approximately the same size as the length, a third gate atop of the channel can be implemented.

The most recent Omega FinFETs [54] implement a gate almost all around the channel (with an Ω shaped gate).

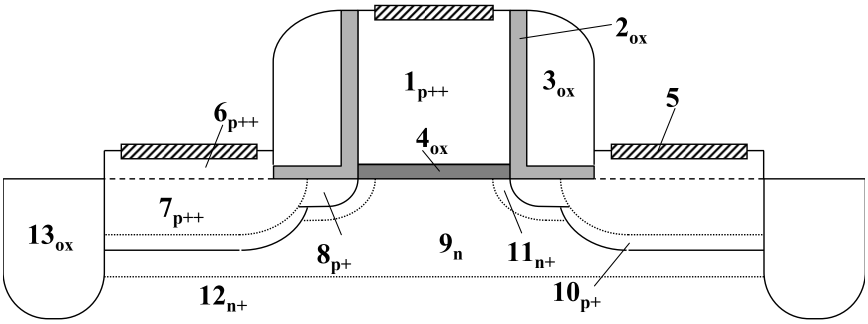


Figure 1.13: MOSFET structure: 1) poly-silicon gate 2) buffer oxide layer 3) gate sidewall isolation 4) gate oxide 5) metal drain contact 6) elevated source 7) source 8) shallow source extension 9) channel 10) halo doping 11) pocket implant 12) retrograde well 13) field oxide (FOX)

1.4.2 Gate level

In this section, only performance neutral transformations of a gate netlist are regarded. Ideas, assuming a slight performance degradation are always outperformed by transistor techniques. A single percent acceptable slowdown can be easily translated into at least 10% cumulative subthreshold leakage and gate tunnelling reduction[11]. As a consequence, there are only a few really different promising ideas how to reduce the leakage at gate level:

[11] Example for the result of oxide thickness scaling on an industrial $65nm$ technology.

Available slack of the gate netlist can be exploited in various ways: In each approach, a gate, not being on the critical path, is replaced by a logically equivalent, slower but less leaky other gate. The most common approach is gate sizing: A reduction of the gate's driving strength will reduce leakage and area. As area reduction is a well known optimisation goal, gate sizing should be already nearly perfectly exploited using any standard synthesis tool. Thus, all other approaches will be compared to an optimally sized netlist.

Instead of reducing the gate size for non-critical gates, or even after deliberately up-sizing gates to increase the available slack [55], leakage currents can be reduced by increasing the gate's threshold voltage. The threshold voltage is usually varied by scaling the channel length, oxide thickness, or channel doping [56]. This idea can be slightly improved by adding mixed V_{th} cells to the technology, which contain low and high threshold transistors [57]. Instead of reducing the data-path, also slower sequentials can be used reducing leakage [58]. In [55], also the influence of clustered voltage scaling on leakage power is analysed.

By accurately regarding the influence of process variation, [28] can improve the gate sizing itself, as gate sizing is traditionally done for the nominal leakage, and not the average one due to variations.

In order to compare all these techniques, Table 1.3 collects their optimisation results for three arbitrarily chosen ISCAS85 benchmarks.

An interesting industrial overview on the effectiveness of gate level optimisation techniques as V_{DD} scaling, L_{ch} scaling, stack forcing (duplicating one gate input to force the body effect), and reverse body biasing (see Section 1.4.4) is presented by Intel [61].

1.4.3 Techniques above gate level

At gate level and below, most optimisation approaches are specific modifications, which will definitely (or most likely) improve the leakage (eventually for the cost of other metrics). Above gate level, optimisation typically means modification that changes (but not necessarily improves) a certain metric. In these cases, optimisation is done by changing the circuit, determining the impact of the change and doing/undoing changes with some heuristics. Thus, high level leakage optimisation (besides few special cases) will require high level leakage estimation. Due to this problem, high level leakage optimisation ideas are rare and are always a byproduct of high level leakage estimation research.

The first work on high level leakage estimation and optimisation is from 2000 [9], where a static power model for architects is presented. The authors propose several optimisations, based on their estimation. Two proposed ideas are reducing the supply voltage to trade off performance and power (leakage as well as dynamic), and reducing the number of devices (again reducing the performance for a leakage reduction). For these optimisations, no leakage estimation is needed; less transistors at lower voltage will consume less leakage power. Besides these two trivial techniques, the authors propose to use more efficient circuits: Several design alternatives are logically equivalent but differ a lot in leakage. The design efficiency, which is an abstract metric they use within their models, can be used to quantify the efficiency of a design alternative.

optimisation	C880	C6288	C7552	remarks
Sirisantana et al [56] (250nm)				
Dual N_{dep}	83.4	39.2	83.7	
Dual L_{ch}	57.6	23.8	56.6	P_{dyn} increased by $3 - 15\%$
Dual T_{ox}	81.4	37.4	76.4	P_{dyn} reduced by $4 - 22\%$
Srivastava et al [55] (130nm)				
Clustered V scaling (CVS)	20.6	1.1	30.2	P_{dyn} reduced by $15 - 33\%$
Dual V_{th} (DVT)	20.6	1.0	36.4	P_{dyn} reduced by $12 - 46\%$
CVS + DVT + W adaption	44.1	20.3	36.6	P_{dyn} reduced by $19 - 51\%$
Sultania et al [59] (100nm)				
Dual T_{ox} + input swapping	92.2	74.0	96.0	Mainly gate leakage reduction
Pin reordering	7.4	2.9	6.2	No technology requirements
Sill et al [57] (65nm)				
Dual V_{th}	65.5	39.8	75.1	Optimal L_{ch}, T_{ox}, & N_{dep} scaling
Mixed V_{th}	69.1	49.3	77.5	Optimal per transistor V_{th} library
Bhardwaj et al [60] (130nm)				
Improved gate sizing	5.2	12.2	11.6	Regards variation influences

Table 1.3: Optimisation results for three ISCAS85 benchmarks from five representative gate level leakage optimisation techniques. All techniques compare to the optimal sized (not regarding variation) reference circuits. The second to fourth column gives the percental optimisation against this reference (e.g. 75% reduction means a reduction of a factor four).

Based on their synthesis based leakage estimation approach, [62, 63] propose an improved binding and allocation algorithm heuristically reducing leakage during behavioural synthesis. The main idea of their work is to increase resources idle times to make later power management more efficient. Their heuristic can be improved by mapping the problem to the knapsack problem [64] and by introducing a tabu search based optimiser [65]. In [66], they present a solution for pipelined components.

In [67], the binding is optimised to reduce leakage by reducing the peak temperature. The mandatory logic operations per unit time imply a certain minimum dynamic power. As a +1 Kelvin temperature deviation will increase the leakage by a higher magnitude, than a −1 Kelvin temperature reduction will reduce it, the leakage will be the smaller, the more homogenous the workload and thus the temperature is distributed over the system.

Using the high level estimation methodology developed in this thesis, first optimisation results were obtained by my colleagues and students [68, 69]. By simply letting the *simulated annealing* based floorplan and binding heuristic know the consequences (in terms of total power) of design modifications, the optimal trade-off between binding for dynamic power, floorplanning for interconnect power, and binding and floorplanning for low leakage (low thermal distribution) is found automatically.

Additionally, the binding [68] and scheduling [69] can be adapted to improve the gain of later power management. In [68], a cost matrix for each predecessor-successor pair of operations is generated[12], and a path, close to the optimal one is determined. In [69], the constraints from the CDFG and the effect of binding to the leakage power are translated into a linear programming form and then solved using a standard integer linear programming (ILP) solver, resulting in a schedule and binding nearly optimally supporting component-wise power management.

1.4.4 Leakage management

In order to trade of leakage power for performance at run-time, the conceptually most simple way is dynamic voltage scaling (DVS). As soon as the available performance is not needed, the supply voltage can be reduced to save leakage power.

Besides DVS, among the known solutions for leakage power reduction, the most promising ones above the technology level are body biasing and power gating, both reviewed in [70]. Both techniques have to be implemented on low levels of abstraction, but controlled from system level.

Even though body biasing is less powerful in reducing leakage it has the advantage of leaving the component functional all the time. Body biasing offers a tunable trade-off between leakage and performance. State assignment has the least benefit, but also the least implementation overhead. In contrast to the latter ones, it can be implemented in each technology.

ABB

The principle of body biasing is to exploit the $V_{\mathrm{th}}(V_{\mathrm{body}})$ dependence (Equation 1.6) by forcing a positive or negative body effect. This way, global deterministic (inter-die) variations can be significantly reduced. Forward body biasing (FBB) means increasing the body voltage for NMOS (or decreasing it for PMOS) to force a negative body effect, thus a reduction of the threshold voltage and in consequence a faster, but more leaking device. Reverse body biasing (RBB) does the exact opposite: NMOS body voltage is reduces below $0V$ or PMOS body voltage is increased above V_{DD} and the transistors are slowed down, saving leakage.

Forward and backward biasing both have fundamental application limits. For FBB, the body voltage must be far below the $0.65V$ limit (for silicon), to avoid body-source diode currents. [71] suggests a $450mV$ FBB voltage limit, to have sufficient robustness against noise influences. For RBB, the threshold voltage and thus the performance and subthreshold leakage are constantly reduced. But with extreme back-biasing, the drain-bulk junction becomes thinner due to the higher V_{db} potential. A thinner junction exponentially increases the band-to-band tunnelling probability (see Section 1.2.3). Below a minimum back-bias, the performance is still reduced, but the total leakage is increased due to the BTBT leakage. Figure 1.14 summarises the leakage behaviour with varying body potential. The minimal RBB voltage is technology dependent, and can even be at positive voltages [72].

Several ideas have been proposed on how to take advantage of body biasing. [73] showed, that a performance neutral *sweetspot* between supply voltage and body voltage can be found, in which the

[12] under the assumption of power gating with a transition energy and time, thus with a certain break even time

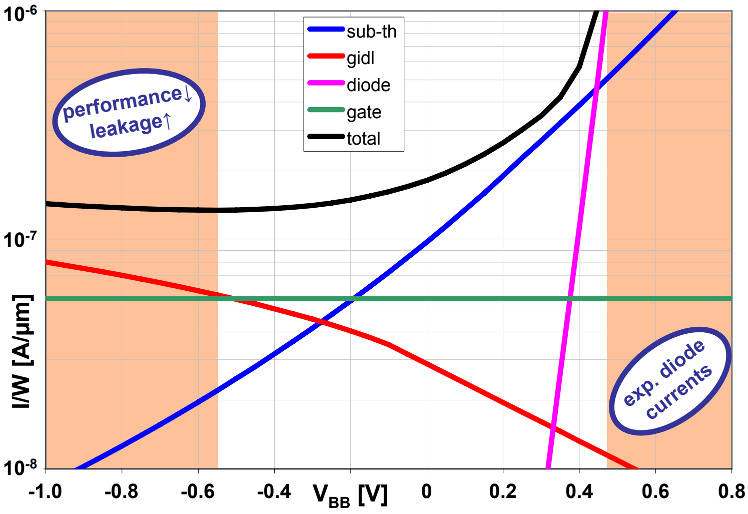


Figure 1.14: Subthreshold current, gate induced barrier lowering, bulk-source diode current, gate tunnelling, and total leakage over body voltage for a 65nm NMOS.

system operates with the same performance but with reduced power consumption. [74] came up with the idea of exploiting the positive correlation between power and performance under process variation by applying a body bias fixed per device after manufacturing and thus reducing the yield loss. [75] proposed a method dynamically assigning optimal ABB voltage to keep the leakage minimal, while the performance is just high enough. This method can save a lot of leakage power if the system is not within the worst case design corner, it was designed for. In [23], Intel evaluated adaptive bidirectional biasing (RBB or FBB controlled on chip at runtime). Their results (Table 1.4), show that at least for the 150nm technology ABB can significantly reduce the process variation.

technique	parametric yield loss	slow bin	fast bin
no ABB	53%	46%	1%
ABB	0%	68%	32%
multi ABB	0%	4%	96%

Table 1.4: Binning population of an Intel 150nm test-chip [23] enabling bidirectional body bias control for NMOS and PMOS *ABB*. The *multi ABB* technique enables different body voltages for different corners of the die reducing the deterministic within-die variation (rf. Section 1.3.1). All systems are sorted into two useful bins (slow and fast). Systems, slower than the slow bin contribute to the parametric yield loss. Without ABB, 53% of all instances were lost, and just 1% entered the fast bin.

A V_{th} hopping scheme, in which the non critical part of the design is high V_{th} by design and the critical part can be switched between high and low V_{th} depending on the workload is introduced by Sakurai [76] for real time systems and [77] for low power processors.

In order to improve the body biasing benefit at gate level, [78] presents a clustering approach also reducing the effect of within-die variation by clustering gates to body bias domains, called within-die ABB. [79] proposes a *larger than* V_{DD} forward biasing scheme, were for the $22nm$ node the NMOS and PMOS body is simply switched, and for smaller nodes an NMOS bias $> V_{\mathrm{DD}}$ is applied. The authors showed that both, delay and leakage sensitivity towards variation can be reduced and the thermal stability can be increased. This observation is surely true, but just valid, if the supply voltage in $22nm$ is below $650mV$ and devices are still bulk CMOS, as ABB can generally not be used in UTB or multi-gate devices. [72] proposes a hardware device (a modified current mirror) approximating the minimal body voltage at runtime and thus reducing the effect of inter-die variability as well as thermal variability for ultra low power devices.

A first combined V_{DD} and ABB variation approach was introduced by [80]. They developed simple power and delay models, which are accurate enough to guide a task-wise dynamic V_{DD}, ABB and frequency scaling approach, thus exploiting the system-level slack.

My recent research in this area focuses on high level techniques maximising the variation reduction [81] by carefully integrating body biasing decisions to the earliest stages of system synthesis. It exploits the available register-to-register slack by classifying components of the design into two or more domains (ABB-islands). For each ABB-island an individual *sweetspot* between supply and body voltage can be found, maximising the leakage reduction without reducing the performance. Due to the high additional interconnect required for supplying a second V_{DD}, the supply voltage must not be optimised per component but per block, leading to V_{DD}-islands. The *sweetspot* per group of components is a trade-off between supply voltage and body voltage. Thus the optimal time for planning the ABB-islands and V_{DD}-islands is during synthesis and coarse RT level floorplanning (see Section 3.5 for details).

In [82], a fine grained (gate-wise) ABB application is proposed. For a large number of explicit instances of deterministic, as well as statistic variations, the gate-wise optimal body voltage is determined. Afterwards, a correlation matrix M is set up describing at M_{ij}, how likely it is, that gate i and gate j have comparatively similar optimal body potential. Then, a clustering of the matrix M into a user-constrained number of groups is done heuristically, limiting the number of ABB domains, while trying to keep the optimisation potential high. The promising results of this work are presented in Table 1.5.

Power gating

The idea of power gating is conceptually simple: A sleep signal, which is 0 if the component is idle is used to lock a sleep transistor, which is in series to the logic, usually towards ground. By selecting sleep transistors with a low source-drain leakage, the overall leakage of the gate can be reduced by nearly two orders of magnitude, as soon as the gate is in its sleep state. Even though the principles of power gating are well known, practical aspects are in the scope of recent research [83]: Power gating techniques can be generally divided into fine-grain and coarse-grain [84]. Coarse grain power gating uses a virtual ground (or virtual supply) line for large CMOS structures, even entire RT components.

	leakage reduction[%]		delay reduction[%]	
Technique	μ	95% percentile	μ	95% percentile
1 cluster	4	36	1	8
2 cluster	37	59	0	8
3 cluster	43	64	0	8
4 cluster	47	66	0	8

Table 1.5: Gain of the fine-grain ABB clustering approach [82] compared to a dual V_{th} optimisation. With only one cluster, thus using conventional ABB, the mean leakage just as low as for the dual V_{th} approach, but the 95% percentile is significantly lower. The introduction of a second ABB domain reduces the average leakage by 37%. The gain of using more clusters is diminishing.

Virtual ground and real ground are connected by a large number of wide NMOS sleep transistors. Fine-grain power gated logic uses a sleep transistor for each single gate. The sleep transistor is integrated into the layout of each standard cell.

Fine grain power gating is easy to use and well supported by traditional design flows, as only the standard cell library changes and the sleep signal has to be delivered. The area overhead is high, as the sleep transistor of each gate has to be large enough to deliver the peak current of its logic. Because the peak current of a group of gates is usually much lower than the sum of their peaks, coarse grain sleep transistors are significantly smaller [85]. But coarse grain power gating generates a large number of technical problems. The size of the sleep transistors has to be carefully chosen [86], as large transistors increase sleep-state leakage and sleep-transition cost and small transistors reduce the performance.

The state retention after wake up was intensively researched recently. Voltage anchors [87, 88] can maintain the state but need a 3rd, not gated ground or supply rail. Alternatively, sequential gates can be kept in an intermediate mode which is lowered in leakage but weakly on to maintain the state [89]. In [90], a virtual ground/supply rail clamp (VRC) technique is presented, where diodes in parallel to the gating transistors ensure a certain retention voltage for sequentials as well as data paths.

In NMOS gated devices, after a macroscopic ($T_{\mathrm{slp}} > 1ms$) sleep period, all gate outputs are close to V_{DD}, due to the comparatively low resistance of the PMOS logic towards supply and the high resistance of the gating NMOS towards ground (for PMOS gating vice versa). On wake up, the sleep transistors conduct as well as all pulldown networks (as all fan-in signals are at V_{DD}). Thus, on wakeup, all gates discharge. All nodes but the first one, which is driven from somewhere outside, now see a logic 0 at their inputs and start to charge up again. A large amount of short circuit current will flow[13], as all nodes are in transition between 0 and 1 for a very long wakeup time. Especially in coarse grain devices, the large discharge current, together with additional short circuit current will

[13] [91] presents a peak current prediction for power gated system.

stress the virtual ground. If this ground bounce is high enough, it may hazard other logic decreasing speed and noise immunity [92].

Several solutions to the ground bounce problem have been recently proposed. [89] employs a set of parallel sleep transistors with increasing size. At wake-up, only the smallest one is immediately turned on, limiting the first current peak. After a short time, the next larger one is conducting and so on. Alternatively, all sleep transistors are *slowly* turned on by a staircase signal. Both techniques reduce ground bounce for the cost of a long wake-up time penalty. Alternatively, a logic netlist with an initial fine grain setting can be clustered to coarse-groups preventing short circuit and limiting maximum wakeup current as presented in [92]. In this work, the logic is clustered into groups with respect to a maximum current constraint. Groups are chosen in a way, that all cells can be safely discharged without generating short circuit at the fan-out. First, all gates having a logic 0 at the output are turned on again, closing the pull-down network of their fan-out gates. After all logic 0's are restored, the gates having logic 1 output can be safely turned on again. By carefully scheduling the wakeup time for each cluster, the authors can minimise the overall charge as well as the peak current.

Another approach, proposed by [93] is to alternate NMOS and PMOS gating in coarse grain logic blocks. Before wakeup, a pass transistor connecting virtual supply and virtual ground is activated to recycle 50% of the charge stored inside the gated logic. Finally, short circuit currents can be completely avoided by an input driven sleep-transistor sizing [92]: Assuming that the logic-state is fixed, PMOS or NMOS gating is used, always supporting the locking logic and fixing the output value[14].

An additional challenge of coarse grain gating is the overhead of NMOS (or PMOS) transistors and their placing. In conventional standard-cell rows, the ratio between NMOS and PMOS is fixed and the large additional NMOS gating transistors will leave a lot of PMOS area unused. [94] proposes to add *NMOS only* rows for the gating transistors between the logic rows.

In [95], several implementations are evaluated using a combination of body biasing and power gating: A workload dependent zero-bias or forward-bias scheme was able to save 47% leakage power compared to a static $450mV$ FBB core. The area increased by 8%. When the system is idle, additional power can be saved by power gating. Standard power gating reduced the leakage by a factor 37 for the cost of 2.3% performance and 6% area. Combining both techniques by using FBB for the gating transistors increased the benefit to a factor 64 leakage reduction with reduced performance costs of 1.8% and an area increase of 8%.

State assignment

Minimum leakage vector (MLV) assignment means exploitation of the data dependency by identification of the least leaking input vector and assignment of this MLV on idle. By performing state assignment in the feedback gate of the fan-in sequential (refer Figure 1.15), state assignment can be made performance neutrally, and the dynamic power for switching to the MLV is the only cost. While

[14] A logic with a 1 output is NMOS gated, a 0 is PMOS gated.

the data dependency of a single gate in $90nm$ and above can be larger than an order of magnitude due to the body effect, in $65nm$, the increasing gate tunnelling and reduced body effect reduce the data dependency to a factor of $2-3$. Entire RT components tend to even have a much smaller data dependency of ±2.4% (ISCAS C499) to ±39% (16bit CLA Incrementer) [96]. For $65nm$ and below, simple MLV assignment seems to be a technique free of cost but also nearly free of benefit.

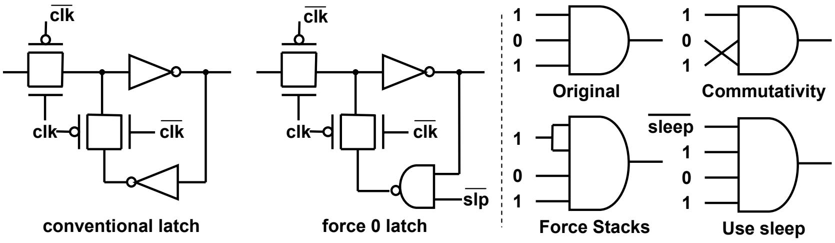


Figure 1.15: Left: Per input bit, only two transistors with low driving strength are needed to force an idle state. Right: Commutativity, stack forcing and internal sleep states are used as long as the critical path remains unchanged.

By a critical path neutral reorganisation of the gate netlist, [96][15] shows, that an RT component designed to save as much leakage as possible in one certain state, can nevertheless save 51% (ISCAS C2688) up to 91% (8bit CLA subtracter) for the cost of less than 5% additional area (refer Figure 1.15).

Earliest results on minimum/maximum current estimation due to different input states and determination of the best/worst state have been motivated by IDDQ testing [24, 97]. [98] presented special force 0 and force 1 latches for MLV with low area and power overhead. In [99], an ILP based best/worst state search approach is presented, reducing the average leakage by 20.3% on average, only by an assignment of the best state. Applying the best of 10,000 randomly chosen vectors would only reduce leakage by 9.1%.

The first approach aiming to improve data dependent leakage reduction is [52], who combined MLV and pin reordering and achieved a reduction of 27% subthreshold leakage and 45% gate leakage on average, using a branch and bound approach. [100] then showed, that adding control points, can reduce the leakage in the best state by additional 15% compared to simple MLV application. Control points can be implemented by choosing gates inside the circuit, incrementing their number of inputs, and assigning sleep signal to the new input pin so that the logic function remains unchanged when not in sleep (see Figure 1.15). By improving the optimisation heuristics, [101] could reduce the leakage in sleep by 49% versus MLV.

1.4.5 Memory techniques

In contrast to the dynamic energy, being unavoidable cost for each transition, leakage currents are constantly draining power independently of a component's activity. Thus, for memories, where only

[15] This work was done by my student using the gate level leakage and delay models of this thesis.

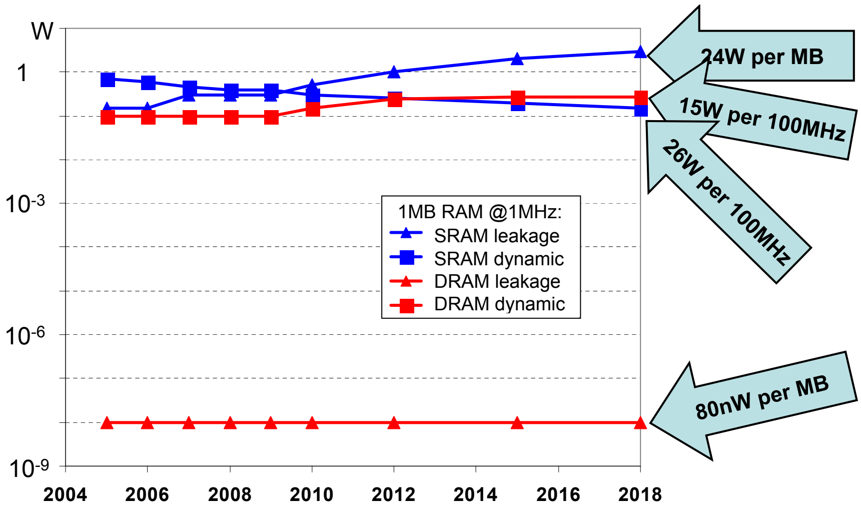


Figure 1.16: Leakage power and dynamic energy per access for an SRAM and DRAM memory. The values base on confidential industry predictions and have been anonymised, maintaining the right relations and orders of magnitude. To visualise the meaning of the values, the 2018's predictions have been translated into Watts per MB for leakage and Watts per 100MHz operation speed (under burst access).

a few transistors work per operation, most power dissipation is caused by leakage. Vice versa, as memories dominate the chip area of many systems, most power dissipation due to leakage occurs in memories. Because of the high benefit of memory leakage reduction and due to the high regularity, there have been several dedicated memory leakage optimisation approaches reported in literature. As only SRAMs suffer from severe leakage[16] (see Figure 1.16), all optimisation techniques refer to SRAM optimisation, which typically means cache, not main memory optimisation.

A 6T SRAM bit-cell is a circuit, consisting of four NMOS and two PMOS transistors, which is most of its time (if not in R/W) in one of two possible, symmetrical states. The leakage in both states (0 and 1) follows the same principles as for data-path gates as visualised in Figure 1.17. Thus, most transistor level and some gate level leakage optimisation techniques can also be applied for SRAM leakage.

All memory leakage optimisation approaches, reported so far, can be classified into one of these two categories:

- Improved 6T cells sacrificing area, performance or read-write stability for a reduced per bit leakage.
- State loosing implementations as power gating or implementations making a cell temporarily unavailable (remaining their state) as drowsy bits. These techniques have to be controlled

[16] The refresh energy of DRAMs is assigned to dynamic energy, even though the charge loss, making refresh necessary is due to leakage

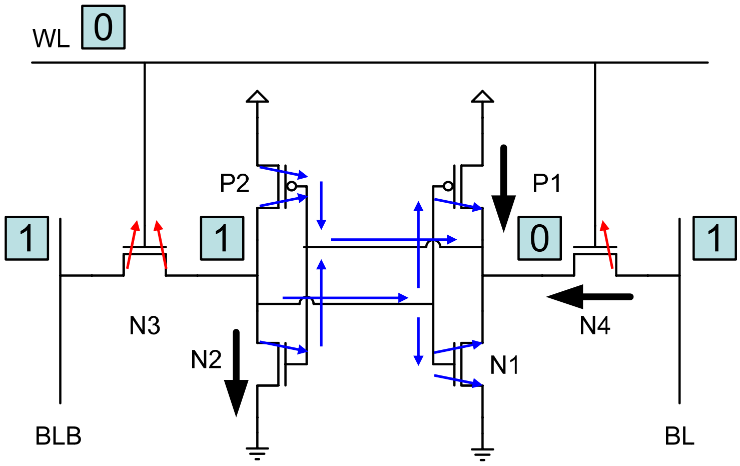


Figure 1.17: Leakage paths in SRAM cells. Black arrows indicate subthreshold leakage, blue ones gate tunnelling and red ones bitline-to-wordline leakage by gate tunnelling.

having a higher level view than single bit cells, as leakage power reduction has to be treated of against the time and energy costs of state retention and/or wake-up.

[102] proposes exploiting the fact, that caches have to store much more 0's than 1's (70% of all bits in cache store a 0 in average over 15 benchmarks [102]). Thus optimising a cell for the 0 state only can trade off costs and use. [103] presents an overview of seven asymmetric dual V_{th} cells. Among the 6^2 different possible assignments of high- and low threshold to the six transistors of a cell, [103] identified several interesting ones, trading off leakage in the 0 and/or the 1 state for performance and R/W stability. The most promising candidates are then optimised by upscaling some transistors to re-improve performance or stability.

High level SRAM leakage optimisation techniques either employ power gating or drowsy memories. Equally to data-path components, power gating for 6T cells is implemented with an additional (usually high threshold) gating transistor cutting power supply (typically implemented as NMOS gating, due to the higher driving strength per unit area of NMOS). Drowsy memories are realized by switching the supply voltage of a memory line or even a larger structure from nominal supply voltage to a state retention voltage. A 6T cell powered with state retention voltage would loose its state at a read access. Thus, drowsy memory structures have to be powered up again before they can be accessed.

For both techniques, power gating and drowsy memories, a higher level power management has to trade off the cost of wake up and state retention from lower memory hierarchy levels with the leakage savings. [104] proposes a *cache decay* technique, where each cache line is powered down after a fixed (but large) number of cycles without access. After this timeout, the cache line is marked as invalid and has to be read from the main memory if it is needed again. Instead of implementing one counter per cache line, generating a large hardware overhead, [104] employs one counter as a frequency divider. Each cache line then only needs a small counter to count up to the large decay threshold. As the exact number of cycles is irrelevant, [105] proposes an a analogue RC circuit, determining the time to the cache decay. Each access charges the capacitance, which is slowly discharged over a high resistance. A level converter and two pass transistors translate the analogue signal to a drowsy- or zero-voltage driving the ground line resulting in drowsy or awake memory.

In [106], a dynamically resizable I-cache is proposed. By counting the miss- and hit rate, the best I-cache size for the actual workload is determined at runtime. All cache lines, that are currently not needed are power gated. In contrast, [107] proposes a *frequent value cache* (FV-cache) design. Values, frequently occurring in data caches (values as 0, 1, or 0xffffffff), are coded to the MSB part of a codeword, the LSB part is disabled, using power gating, and a table translates frequent value codes. [108] improves this FV-cache design to an STV-cache (*spatio-temporal value cache*). An STV-cache can code frequent values and double occurrences of non-frequent values within one cache line into the MSB bits.

2 Description of the model

This chapter will present, how accurate and fast RT level soft-macros[1] for leakage prediction can be generated. The models are built in a strict bottom up manner, from transistors to RT components. The structure of this chapter follows this bottom up concept:

After the discussion of the model properties in Section 2.1 and a definition of the required inputs and desired outputs of the model in Section 2.2, the model will be presented in three stages from transistor level (Section 2.3) over gate level (Section 2.4) up to RT level (Section 2.5). In Section 2.6, the connection to an existing delay model is discussed, and Section 2.7 presents how optimisation techniques such as power gating can be included into the model framework.

The entire model as outlined in [26] has been developed and evaluated for the $90nm$ technology. After discussion with Infineon and ST Microelectronics, and according to a reevaluation with a $65nm$ industrial technology [109], several updates have been made. The initial model will be presented first as it intuitively presents the principal modelling idea. Problems of the initial solution are discussed then, and the final concept including necessary updates is presented afterwards.

2.1 Problem statement

The ultimate goal of this thesis is to be able to predict and optimise the leakage of entire digital systems from their specification as good as possible regarding all important influences and all correlations to other system metrics. The system level synthesis and estimation tool PowerOpt already offers several prerequisites [110]:

PowerOpt reads in a system specification in C / C++ / SystemC specification and generates a control and data flow graph (CDFG) describing the hardware parts of the system. This CDFG is then synthesised into an RT netlist minimising a cost function

$$f_{\text{target}} = P_{\text{comp}} + P_{\text{local icn}} + P_{\text{global icn}} + P_{\text{clock}} + \alpha \cdot A_{design} + \beta \cdot t_{\text{crit}}, \tag{2.1}$$

where P_{comp} is the dynamic power of the RT components, which can be optimised by reducing the activity while binding the allocated operations to components. $P_{\text{local icn}}$ and $P_{\text{global icn}}$ is the dynamic power of the local (inside RT components) and global interconnect. Interconnect power is automatically reduced while reducing the component's activity; additionally, $P_{\text{local icn}}$ is influenced by the shape and $P_{\text{global icn}}$ by the position of the components, optimised when floorplanning the system. P_{clock} is the power cost for clock distribution (floorplan dependent), A_{design} is the expected overall

[1] A single model that can describe a class of RT components. Here, one architecture but all bitwidths.

area (dominated by the component's allocation) and t_{crit} is the RT level critical path (in clock-cycles) through the entire RT netlist (scheduling dependent). A framework optimising all the synthesis steps of scheduling, binding, allocation, coarse floorplanning, and clock and bus routing was developed by Ansgar Stammermann and me[2] [111].

Both goals, leakage estimation and optimisation, can be achieved from here by the introduction of RT level leakage models for RT components. When optimising the system, its leakage power can be added to the cost function (Equation 2.1), and the heuristic search engine will try to also minimise the leakage power. Having found the optimal system, the leakage models can be used again to generate a leakage estimate. Even though the heuristic optimiser was designed to optimise the dynamic power, it can also influence the leakage (mainly by reducing the component count and the components size at allocation and module selection). All binding and floorplan optimisation is not really influencing the leakage. In [69, 96, 112], three updates[3] of the optimisation engine are presented, introducing dedicated leakage optimisation capability for the optimisation heuristic [69, 96, 112].

Thus, the ultimate goals can both be solved by introducing accurate leakage models for RT components. The basic idea of these models, which will be detailed in the next sections, is as follows:

- As gates consist of transistors, the behaviour of the leakage of a logic gate on all parameters somehow results from the behaviour of single transistors on these parameters.

- Even though the potentials at each terminal of each transistor and the transistor geometry depend on the logic gate and the state of this gate, the leakage of each logic gate in each state is described by an equivalent circuit, composed by only a few parallel connected reference transistors with variable width. Evaluation (Section 5.3.2) shows less than 4% standard deviation between the leakage of the gate and its equivalent circuit.

- As the transistor width has a near perfect linear influence, this composition mathematically equals a linear combination of the reference transistor's leakage current functions.

- When accounting the leakage currents of the gates in a clever way (as described in Section 2.4.2), there is no fan-in or fan-out interdependency between the gates of an RT component. The leakage of the gates simply sums up.

- This method conserves the linearity of the RT level model. An RT component with known netlist and known input data can be described as a sum (= linear combination) of gates. As the gates are linear combinations of transistors, an entire RT component can be described as a *super gate*, thus again as a linear combination of the reference transistors.

- A component is modelled by only eight scaling parameters. These parameters only change smoothly when changing input data or bitwidth of the component.

[2] That work is not part of this thesis

[3] developed by three of my former students

For all technologies, from generic $180nm$ PTM technology [38] over an industrial $90nm$ technology [113] up to the recent $45nm$ technology [114], evaluation proved, that this basic assumptions are valid.

2.2 Parameter selection

In Section 1.2, the three most dominant sources of leakage current have been identified as subthreshold current, gate to source/drain tunnelling, and pn-junction band-to-band-tunnelling amplified by gate induced drain leakage (initially, this selection was motivated by [7]). Each of these currents is meant to be determined by the model described here. In this section, the input parameters which have to be regarded by the model are identified.

The possible input parameters can be separated into three classes: $DyPa$, random $StPa$, and deterministic $StPa$. $DyPa$ are parameters, which can vary during runtime (e.g. temperature or voltages). $StPa$ are fixed for a single instance of the system, but can vary between different instances. For example, the printed channel length may diverge for transistors at different locations on the same chip (local statistical $StPa$), or all transistors on one chip have the same deviation compared to the nominal parameter value (global deterministic $StPa$).

For the $DyPa$, the choice of the first three parameters is quite obvious and reported frequently in literature [11, 23]:

- Of course, the temperature has to enter the model as the subthreshold leakage is highly temperature dependent, doubling every $20-40$ Kelvin temperature increase (rf. Section 1.3.3).

- Supply voltage has to be regarded, as it increases subthreshold current by increasing the DIBL effect, it increases gate leakage by exponentially increasing the tunnelling probability, and it increases the junction leakage by boosting BTBT (rf. Section 1.3.4).

- Finally, as one main purpose of the model is computing the effect of adaptive body biasing, the body voltage, applied to the NMOS and PMOS substrates also has to enter the model.

There is a fourth $DyPa$, entering the model, the input data. For the purpose of gate level modelling, data dependency is usually regarded by model splitting (one model for each input vector) as presented in [19]. But for RT components, model splitting per input vector is prohibitive. Thus, the fourth $DyPa$, chosen is:

- Signal probability as a measure of the number of zeros and ones at the primary inputs and (due to correlation) also at the internal nodes. As presented in Section 1.3.5, the input vector has only a medium influence (below 10% standard deviation) on the leakage of large structures. Thus, selection of the signal probability is more than sufficient to describe this effect.

2.2.1 Global deterministic die-to-die variation

After discussions with Infineon and ST Microelectronics, it turned out that each integrated device manufacturer uses its own, highly confidential process parameter variation model.[4] In order to not disclose these models, a set of three obvious [10] *StPa* is chosen in this work as representative of the confidential real parameter set. Each modelling step, described throughout this work is adaptable to other and/or more process parameters.

To analyse the accuracy of the model, the following three parameters describing process variation where chosen:

- The printed channel length L_{ch} was chosen as it is known to be not easy to control and because it has a huge impact on the subthreshold leakage.
- The oxide thickness T_{ox} is a well controlled parameter, but as the smallest change can lead to substantial gate leakage variations, it also enters the model. In recent technologies, the gate leakage is increased by a factor of 10 with an oxide thickness variation of 2.5Å, which is approximately one atom-layer. The oxide thickness also has a huge impact on the gate induced drain leakage (refer Section 1.2.3).
- Finally, also the channel doping N_{ch} is included, exponentially influencing the junction-leakage, as well as the threshold voltage and thus the threshold current.

The global variation from instance to instance of these parameters is typically neither Gaussian distributed, nor uncorrelated between the parameters, nor locally stable within one instance (parameter gradients). A sophisticated variation engine, which will be presented in Section 3.2 cares for modelling all global *StPa* from instance to instance and the gradients[5] within an instance.

2.2.2 Local statistical within-die variation

For a single model evaluation with given parameter mean values, a random Gaussian variance, describing the local statistic (intra-die) variation is assumed.

As presented by [16] and [22], regarding the second order effect of the local variation on the expectation value of the leakage is mandatory for accurately modelling leakage. Depending on the variance of the input parameters, the expectation value of the leakage on an entire system can be up to 40% higher [16] than the nominal leakage (the leakage of a system with nominal *StPa*). Thus, as local statistical *StPa*, I chose the (mathematical) variance σ for the parameters chosen above as global *StPa*.

[4] IBM is best to my knowledge the only manufacturer reporting their process variation model [44] employing five process parameters: long channel threshold *VT0*, drawn channel length *LD*, short channel effect coefficient *THETA*, low field mobility *U0*, and DIBL factor *ETA*.

[5] As presented by [3] or alternatively [29], it would also be possible, to describe the effect *StPa* gradients from a system level view. But as local variations of temperature and voltage require a local (per RT component) model evaluation methodology, the gradients of *StPa* can easily be modelled by the same methodology.

From a physical point of view, an independent list of varying parameters would be desirable, but this cannot be handled by the model due to structural limitations. If a certain parameter does only show local (no global) variations, it can be described by adding this parameter to the list of global *StPa* but assuming a zero global variation. This *workaround* has no negative impact on the accuracy but a slight impact on the complexity of the model. Each parameter, varying locally, globally or both has to enter the model as a full parameter.

2.2.3 Local deterministic within-die variation

Regarding the variation taxonomy from Table 1.1, local deterministic *StPa* (the same device shows an absolute deviation for each system instance) do not enter the model explicitly. Nevertheless, if local deterministic variation data would be available when characterising an RT component, its effect will automatically be supported by the model, as there is no variability due to deterministic static variation from an RT component point of view.

2.3 Simulation of reference transistors

As the entire model is built in a *bottom up* way, it is made abstraction level by abstraction level. Compact models are used as initial input to the model, thus this first layer of the model has to capture the behaviour of single transistors on all relevant parameters. In order to analyse, how the transistor model has to be constructed, to ensure maximum model quality, the basic idea is developed straight forward in Section 2.3.1. Then, flaws of this initial model are analysed and the model is refined accordingly in Section 2.3.2 and Section 2.3.3.

2.3.1 The initial model

According to the parameters identified above (Section 2.2), the initial version of the transistor model distinguished between NMOS and PMOS transistors. As a single transistor can be in only two input states (0 and 1), the data dependency was included by a model splitting, leading to four transistor models: locking NMOS, where subthreshold and junction currents dominate the leakage (NMOS0), conducting NMOS where only gate leakage occurs (NMOS1), locking PMOS (PMOS1), and conducting PMOS (PMOS0). The idea of describing transistor leakage for each valid state was motivated by the work of IBM [18] who presented the twelve states, a transistor can be in. In [22], the individual leakage estimation for NMOS and PMOS was first suggested. In that work, the locking transistors NMOS0 and PMOS1 were separated into four drain/source states 00,01,10, and 11; while for the conducting transistors drain and source obviously had to have the same potential i.e. 00 or 11.

In my work, for each of the four transistors (called references, or reference transistors), the further modelling was done for each combination of the *DyPa* temperature, supply voltage, and body voltage[6], sampled over a sufficient range.

For each of these points (NMOS or PMOS, 0 or 1, specific sampling values for temperature, V_{DD}, and V_{BB}), the leakage was simulated for a transistor with $1\mu m$ width and nominal values of channel length, oxide thickness, and channel doping. Then, for each *StPa*, 2 additional simulations were done for a positive and a negative parameter variation:

$$\begin{gathered}
I_{c,t,v,b}\left(L_{ch}, T_{ox}, N_{dep}\right) \\
I_{c,t,v,b}\left(L_{ch}-\Delta_L, T_{ox}, N_{dep}\right)\ ,\ I_{c,t,v,b}\left(L_{ch}+\Delta_L, T_{ox}, N_{dep}\right) \\
I_{c,t,v,b}\left(L_{ch}, T_{ox}-\Delta_T, N_{dep}\right)\ ,\ I_{c,t,v,b}\left(L_{ch}, T_{ox}+\Delta_T, N_{dep}\right) \\
I_{c,t,v,b}\left(L_{ch}, T_{ox}, N_{dep}-\Delta_N\right)\ ,\ I_{c,t,v,b}\left(L_{ch}, T_{ox}, N_{dep}+\Delta_N\right) \\
c \in \{\mathrm{NMOS0}, \mathrm{NMOS1}, \mathrm{PMOS1}, \mathrm{PMOS0}\}\,,\ t \in \{300K, 320K, \ldots, 400K\} \\
v \in \{0.8V, 0.9V, \ldots, 1.4V\}\,,\ b \in \{-0.5V, -0.4V, \ldots, 0.5V\}
\end{gathered}$$

For a later use, these currents were interpolated for temperature and voltage values between the sampling points resulting in seven parameter dependent currents I_0, I_{L-}, I_{L+}, ... In order to predict the effect of inter-die variation, a separability of the parameters was assumed:

$$I(L,T,N) = I(L_0,T_0,N_0) + f(L-L_0) + g(T-T_0) + h(N-N_0). \tag{2.2}$$

Using the respective gradient (for instance $f(L-L_0) = (L-L_0)/\Delta_L \cdot (I_0 - I_{L-})$ if $L < L_0$ and $f(L-L_0) = (L-L_0)/\Delta_L \cdot (I_0 - I_{L+})$ if $L > L_0$), enabled a first order prediction of the inter-die variation.

In order to compute the effect of intra-die variation, also statistical independence between the *StPa* had to be assumed:

$$p(L,T,N) = p(L) \cdot p(T) \cdot p(N) \tag{2.3}$$

Combining Equations 2.2 and 2.3 resulted in an expectation value

$$E\left(I(L,T,N)\right) = I_0 + E(f) + E(g) + E(h) \tag{2.4}$$

For each parameter, the additional current due to an increased expectation value was then computed by a Taylor series characterised by the three SPICE-simulations. For each *StPa* P, the leakage dependence $I(P)$ was approximated as

$$I(P) = I_0 + (P-P_0) \cdot \frac{I_{+\Delta} - I_{-\Delta}}{2\Delta} + (P-P_0)^2 \cdot \frac{I_{-\Delta} - 2I_0 + I_{+\Delta}}{2\Delta^2} + O(P^3) \tag{2.5}$$

Assuming a Gaussian distribution around the mean P_0 with a variation σ_P, the expectation value results as

$$E\left(I(p)\right) = \frac{1}{\sigma_P\sqrt{2\pi}} \int_{-\infty}^{\infty} dP\, I(P) \cdot e^{-(P-P_0)^2/(2\sigma_P^2)} = I_0 + \sigma_P^2 \cdot \frac{I_{-\Delta} - 2I_0 + I_{+\Delta}}{2\Delta^2} + O(\sigma_P^4/\Delta^4) \tag{2.6}$$

[6] Drain, source and gate voltage result from the circuit description and the supply voltage. For NMOS0, $V_{gs} = 0$ and $V_{ds} = V_{DD}$; for NMOS1, $V_{gs} = V_{DD}$ but $V_{ds} = 0$ to avoid short circuit in channel.

Due to the symmetry in P_0, the odd polynomials disappear, and the remaining error is of the fourth order.

As described above, all reference transistors were modelled for a transistor width of $1\mu m$. The choice of this comparatively large width has several important advantages: At first, it nearly completely prevents the occurrence of the narrow width effect (see Equation 1.8). Thus the threshold voltage is independent of the width and thus the width enters all leakage sources in a linear way. For instance, a transistor of $2\mu m$ width will have nearly exactly twice the leakage[7]. If the transistor is linearly scaled, as it will be done in Section 2.4, the scaling value can be interpreted as the *effective* transistor width in μm.

2.3.2 Introduction of the semi-analytical model

Even though the straight forward model choice is already sufficient for abstracting it to RT level, it suffers from several flaws introducing huge errors, especially for recent technologies, dominated by process variation.

The leakage is simulated at nominal parameters and at two further sampling points for each varying *StPa*. Thus, the initial transistor model is effectively regarding a second order Taylor series of the parameter dependence neglecting cross-correlation between parameters and higher order terms. For the $90nm$ technology, this leads to a moderate underestimation of the effects of variation for high deviations from the nominal value. For a more recent $65nm$ technology [109], the distribution of *StPa* is wider, increasing the likeliness of two or more parameters significantly varying at the same time. Additionally, the process sensibility to parameter variations is higher so that this simple parameter model would lead to unacceptably large errors.

The input to the transistor model is the *DyPa* and the mean and the standard-deviation of the *StPa*. The desired model output is the expectation value of the leakage, with respect to the *DyPa*, and evaluated at mean *StPa* values shifted by inter-die. Thus, it would be most desirable to have an accurate analytical description of at least the *StPa*.

The BSIM simulations are based on equations which are in principal known from the BSIM manual, but they are much too complex to be integrated into a high level model. The BSIM library itself is nothing, but a program evaluating the transistor equations. And using BSIM immediately shows that the computational effort is too high for being employed for high level modelling. Thus, I decided to generate analytical descriptions of the transistors, as accurate as possible describing the transistor behaviour but having a very low complexity.

In order to simplify the BSIM transistor equations, some very helpful assumptions can be made: The first one is, that an analytical description of the *DyPa* is not needed as no expectation value over the *DyPa* has to be computed. Thus it will be enough to develop small analytical models for different settings of voltages and temperature and to regard temperature, V_{DD}, and V_{BB} as constant for each model. Secondly, it can be exploited, that only a few reference transistors need to be modelled, for

[7] Due to the narrow width effect, this relation does not hold for $100nm$ and $200nm$ width.

which the circuit description is exactly known. As for gate, as well as channel and junction leakage, the parameters nearly completely enter as an exponent, the accuracy can be improved by modelling the leakage's logarithm instead of its current[8]. Finally, the model of the locking transistors is split into two models, one describing the gate tunnelling and one describing the subthreshold leakage. In combination these assumptions will ease up the problem as follows:

Simplify the analytical description of the leakage current's exponent under the assumption that the temperature, all voltages and all other BSIM parameters are constants.

The plan is to finally repeat this simplification for the sampling points of the *DyPa*, hoping, that for parameter values, between the exact *DyPa* sampling points, some kind of interpolation (not necessarily in the strict mathematical sense) is possible. An accurate, straight-forward Taylor development needs at least 45 polynomials to describe the leakage with sufficient error measures as

$$\ln\left(I(L,T,N)\right) \approx \sum_{i=0}^{4}\sum_{j=0}^{2}\sum_{k=0}^{2} \alpha_{i,j,k}(\vartheta, V_{DD}, V_{BB}) \cdot L^i T^j N^k. \tag{2.7}$$

This equation can accurately predict the leakage behaviour, as long as the α-values for a certain combination of *DyPa* are exactly known. Unfortunately, it turns out, that the α-values behave chaotically due to the much too large degree of freedom in 45 free (and under-determined) parameters. Even setting insignificant α-values to zero is no solution, as the estimation accuracy becomes unacceptable low before the parameter assignment becomes stable (and well defined).

The solution to this problem is to use an extended Laurent-development[9], as especially the channel length parameter enters several terms in the denominator and in square roots. The general form of 9^{th} order

$$\ln\left(I(L,T,N)\right) \approx \sum_{i=-4}^{4}\sum_{j=-4}^{4}\sum_{k=-4}^{4} \alpha_{i,j,k}(\vartheta, V_{DD}, V_{BB}) \cdot L^{i/2} T^{j/2} N^{k/2} \tag{2.8}$$

seems to be far more complex having $9^3 = 729$ free parameters. But as functions like $1/x$ or $\sqrt{x}$ are directly available and do not have to be developed by a high order Taylor series, this equation can easily be reduced to just three or four significant terms without a huge loss of accuracy (see evaluation in Section 5.3.1). Additionally, the remaining α-values are not only mathematically continuously, but also nearly linear, when being characterised the right way (will be detailed in Section 5.1). Even determining these values at three sampling points only in each *DyPa* direction (resulting 27 sets of αs per reference transistor), the accuracy is at about 5% standard deviation for the channel leakage reference and at 3% for the gate leakage (ref. Table 5.2).

[8] For some reference transistors, a sum of leakage currents, all behaving exponentially, has to be modelled. Using the logarithm of the function $f(x) = \alpha\exp(\beta x) + \gamma\exp(\delta x)$ will nevertheless result in a much smoother behaviour than the function f itself. The proof follows this idea: Without loss of generality assume $\beta < \delta$ and compute $\lim_{x\to\pm\infty} d/dx \ln(f)$, which is β for negative and δ for positive limes. Then compute $d^2ln(f)/dx^2$ and show, that the slope develops monotonically from β to δ

[9] Laurent-development means also using polynomials with negative exponent together with the usual Taylor polynomials. Extended Laurent development means using all polynomials $P^{i/2}$ with $i \in \mathbb{Z}$.

The improvements of this semi-analytical model are visualised in Figure 2.1 and Figure 2.2. When only describing the channel length dependency, the new model nearly perfectly matches the BSIM data. Cross correlations between the parameters slightly reduce the overall accuracy.

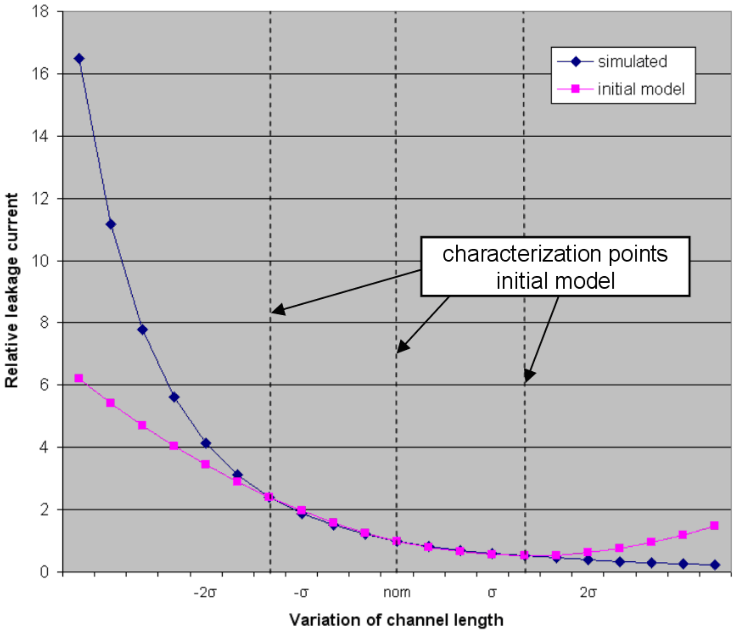


Figure 2.1: Initial, sampling based parameter model, implementing second order Taylor approximation describing the variation dependency. For the sake of simplicity, only the channel length dependency is displayed here. For a technology with a small variation spread and the low parameter dependency, the quadratic approximation is sufficient.

2.3.3 Choice of the reference transistors

As described above (Section 2.3.1), the initial model started with a straight-forward choice of four reference transistors; NMOS and PMOS; on and off. The semi-analytical model introduced in Section 2.3.2 made a split of the locking transistors into subthreshold part and gate-tunnelling part necessary. As the evaluation of the initial model showed large errors in the gate-level layer for stacked transistors (e.g. NAND2@00) a further reference, consisting of two locking transistors was introduced. The final model consists of eight references: **NC**, the channel (subthreshold) leakage of a locking NMOS device, **N0** and **N1**, the gate tunnelling of an NMOS transistor with 0 and 1 at the gate respectively, and **NS**, a stack of two locking NMOS transistors. For PMOS, these four references are defined respectively (**PC**, **P0**, **P1**, and **PS**)

As described in Section 1.2.3, for the 45nm and smaller technology, the leakage mechanism of gate induced drain leakage starts to significantly contribute to the overall leakage budget. Nevertheless, there is no need for a third independent reference transistor class besides subthreshold and

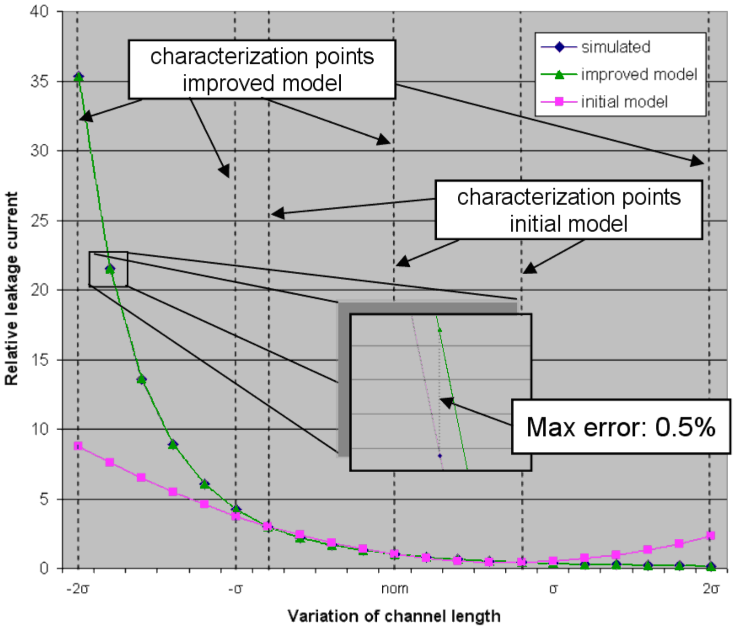


Figure 2.2: For a technology with high variation, quadratic approximation will lead to a huge underestimation. The extended Laurent-development is compact and can accurately describe the real behaviour.

gate leakage to also model cases of gates, in which for instance the gate leakage is high, but the junction leakage is low. This is because the occurrence of the GIDL effect is highly correlated to the subthreshold leakage.

The GIDL and thus high BTBT leakage always occurs together with the subthreshold leakage, thus both effects can be described by the same reference transistor. This phenomenon is easy to understand: For an NMOS transistor, GIDL will occur, if the gate-source voltage is below the drain-source voltage. That means for the references, that the drain is at V_{DD} and the gate and source are at ground. This is exactly the scenario, under which subthreshold leakage occurs: The transistor is locking and there is a potential between drain and source, which can induce a subthreshold current through the locking channel. Thus, it is sufficient to update the reference transistor's definition a little bit (see Figure 2.3)[10].

Instead of determining the subthreshold leakage alone, all channel models (**NC**, **PC**, **NS**, and **PS**) describe the sum of subthreshold and GIDL leakage. The gate leakage circuits for conducting transistors (**N1** and **P0**) do not have to be updated because there is no GIDL in a conducting transistor, as $V_{GS} = V_{\mathrm{DD}}$, and thus the pn-junction is either unchanged (for $V_{DS} = V_{\mathrm{DD}}$) or even

[10] For all technologies, analysed so far, combining subthreshold leakage and GIDL to one reference (and thus to one semi-analytical model), had no negative impact on the modelling accuracy. If, for a new technology, the accuracy of the combined leakage would be too low, the number of reference transistors could easily be increased by two (introducing new references, exclusively describing GIDL in NMOS and PMOS).

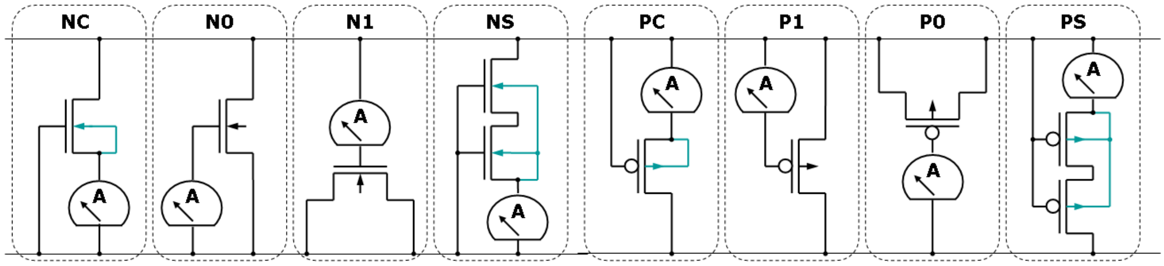


Figure 2.3: Selection of the eight references of the transistor model. The **NC**, **PC**, **NS**, and **PS** models describe NMOS and PMOS subthreshold leakage of single transistors and stacked transistors respectively. As the occurrence of gate induced drain leakage and subthreshold leakage are correlated, the GIDL effect can be included by determining the sum of both. The remaining four references **N0**, **P0**, **N1**, and **P1** describe the gate tunnelling for NMOS and PMOS transistors with gate at 0 and 1 respectively.

shaped less steep (for $V_{\mathrm{ds}} = 0$). The gate leakage parts of the locking transistor models (**N0** and **P1**) also do not have to be changed, as the GIDL, boosted BTBT leakage in a locking transistor, is already captured by the subthreshold leakage models for locking transistors (**NC**, **PC**, **NS**, **PS**).

Remark:

Subthreshold leakage and gate leakage are not correlated, i.e. they may occur independently. As subthreshold leakage and junction leakage are perfectly correlated, it is obvious, that gate and junction leakage cannot be correlated. Thus it is mandatory to have a junction leakage free gate leakage model.

2.4 Gate models

This section will present, how the gates of a technology library can be accurately modelled in terms of leakage, using the reference transistors developed in Section 2.3. As gate level models are not the final goal of this work, but just the next step in a bottom-up modelling approach, the gate models have to be simple enough to enable further abstraction towards RT level, and accurate enough to ensure a high accuracy of the final RT models.

This section is structured as follows: In 2.4.1 the core idea of the gate level model layer is motivated by exactly that *gedanken-experiment*, which initially led me to the entire idea behind this thesis. Section 2.4.2 then defines what exactly is the leakage of one gate. An elaborated definition is needed to simplify the later abstraction to RT level. Section 2.4.3 then presents the mathematical structure of the abstraction from transistor to gate level and in 2.4.4, the preliminary results are recapitulated.

2.4.1 Motivation

A gate is nothing but a structure of transistors and interconnect. As leakage does not occur in interconnect, from a leakage point of view, a gate is nothing but a number of transistors. The only relevant properties of the gate are the proximity of the transistors and the limited number of conditions, its transistors can be in:

- Proximity means, that a gate is a very compact structure, and it can be assumed, that even shortest ranged correlation will identically influence all transistors of a gate.
- Limited number of conditions means, that the special properties of CMOS gates cause, that all its transistors can only be in a very low number of electrical input conditions, depending on the gate's state. A NAND2 gate for instance consists of four transistors. But due to the fact, that they are combined to a gate, these transistors can only be in a very few number of conditions. The source of both PMOS transistors will always be at V_{DD}, and depending on the state, their drains will either be at ground or at V_{DD}. As the fanin is also a CMOS gate, the inputs of all transistors is either ground or V_{DD} and so on. As long as the gate is not switching[11], the gate and its four transistors can only reach four states.

Now, assuming that the collection of the reference transistors was chosen well, the idea is that the entire leakage behaviour of the gate's transistors can be explained by the right choice of reference transistors.

Example:

Let PMOS leakage double each $20K$ in temperature and NMOS leakage double each $40K$. If it is only known that the leakage of a component is $10\mu A$ at $300K$ it cannot be identified, how much of this leakage was due to NMOS or PMOS. Thus the leakage at $340K$ cannot be computed. Pure NMOS would double and pure PMOS would rise by a factor of four, thus the component's current will definitely be in between $(20\mu A \leq I \leq 40\mu A)$. If the component's current at a second temperature can be determined (e.g. $I(380K) = 76\mu A$) and assuming, that total current is a sum of NMOS and PMOS leakage currents, this example can be solved:

$$I(T) = n \cdot 2^{(T-300K)/20K} + p \cdot 2^{(T-300K)/40K} \tag{2.9}$$

$$I(300K) = 10\mu A \cap I(380K) = 76\mu A \tag{2.10}$$

$$\Rightarrow n = 3\mu A \cap p = 7\mu A \tag{2.11}$$

and thus also estimate $I(340K) = 26\mu A$.

This basic idea (separability of the physical behaviour of a gate) is not new, but was implicitly introduced by [9] in a very rough way. In [10, 115], an NMOS-PMOS separation was introduced to

[11] All energy consumed while switching is regarded in the existing dynamic and short circuit models, and will not be regarded here.

the basic separation approach, ending up with a leakage model, being very close to the example in Equation 2.9.

My gate model goes much further, as it is a high-dimensional extension of the example given above. The core assumption of the gate model is, that the leakage of each gate can be described as a linear combination of the leakage of the reference transistors presented in Section 2.3.3. For each gate which is modelled, a SPICE-netlist is set up and a series of SPICE evaluations at various combinations of the *DyPa* is performed. Afterwards, a linear parameter regression is fitting the scaling parameters of the reference transistors to the simulation results of the complete gate. The scaling parameters then can be interpreted as the effective widths of the reference transistors.

In Table 2.1, regression results of the initial model, employing only four reference transistors (NMOS0 describes all leakage currents flowing in an NMOS gate with a 0 at its gate; NMOS1, PMOS0, and PMOS1 are defined respectively) are presented. In this older version, which is more clear to analyse, than the recent eight reference transistor version, the outcome of the regression can be easily understood (In the evaluation in Section 5.3.2, the widths of the real eight transistor model are discussed):

	Industrial 90nm technology				PTM 65nm technology			
	NMOS0	NMOS1	PMOS1	PMOS0	NMOS0	NMOS1	PMOS1	PMOS0
Inv real [nm]	535		727		195		390	
Regr @0 [nm]	557	232	-53	-78.1k	187	-0.05	0.08	389
Regr @1 [nm]	28	797	669	-94.0k	-0.10	187	384	6.14
NOR2 real [nm]	535/492		727/668		204/197		456/441	
Regr @00 [nm]	1062	216	-91	-46.3k	384	-0.02	1.06	886
Regr @01 [nm]	155	-1628	-1627	1070	-2.36	196	452	425
Regr @10 [nm]	51	794	794	-85.0k	16.1	185	408	88.8
Regr @11 [nm]	121	-197	-191	625k	-2.54	364	117	405

Table 2.1: Effective widths resulting from the linear parameter regression for an industrial 90nm technology [113] and the 65nm PTM technology [116] for an INV-gate and a NOR2-gate. NMOS0 is the locking NMOS transistor, NMOS1 the conducting one, PMOS1 is locking and PMOS0 is conducting.

Discussion of the NMOS1 and PMOS0 results in 90nm:

The first observation (which is consistent for all gates in 90nm) is that the regression of the conducting transistors NMOS1 and PMOS0 results in values with absolutely no physical meaning. In contrast, for the 65nm technology especially the NMOS1 regression results correlate well with the physical parameters.

This phenomenon was analysed, and is well understood: The regression only tries to minimise the total model error and not to find best length correlation with physical meaning. For the 90nm technology, the NMOS1 leakage (dominated by gate leakage) is 20 times smaller than the NMOS0 leakage (dominated by subthreshold). PMOS gate leakage is again orders of magnitude smaller, as the effective mass of holes is larger than that of electrons and tunnelling exponentially depends on the mass of the tunnelling particles. The regression thus uses the vectors of NMOS1 and PMOS0 in order to correct the NMOS0 and PMOS1 fitting instead of trying to fit the NMOS1 and PMOS0 behaviour. In order to evaluate this assumption, the NMOS1 and PMOS0 vectors were manually set to 0, with the result, that total error was only marginally higher.

For the $90nm$, the entire model could be very simple by ignoring gate leakage and thus limiting it to two references. But the $65nm$ evaluation shows that as soon as gate leakage becomes important, reference transistors describing the gate leakage are needed. The regression for 65nm automatically determines equivalent lengths here:

Discussion of the regression results:

Without having stacked transistors, this approach is obviously correct and the scaling parameters obtained represent the physical width of the transistors (see Table 2.1, inverter). For instance, the $65nm$ inverter with its $195nm$ NMOS and $390nm$ PMOS transistor with a 0 at the input can be modelled best by $187nm$ NMOS0 and $389nm$ PMOS0 respectively, a quite obvious result. Instead, for more complex structures, the scaling parameters do not need to have a physical meaning.

The two parallel NMOS transistors of the NOR2 gate in Table 2.1 behave like a single transistor of double width in terms of leakage, if both inputs lock or both conduct. In the mixed case, there is no NMOS0 contribution, as the conducting NMOS1 transistor short-circuits the subthreshold leakage dominated NMOS0 contribution. The gate-leakage dominated NMOS1 is in the order of the physical width of the conducting transistor.

The PMOS part behaves similarly for the 00 case. As PMOS0 is gate-leakage dominated, the PMOS branch is equivalent to a single PMOS0 transistor of double width. In the 11 state, the widths of both transistors do not add, as current passes them in series. Instead, due to the stacking effect (see Section 1.2.1), the effective width is far below the $220nm$, that would be expected for resistors.

2.4.2 Definition of the leakage of a gate

When developing gate wise leakage models, it has to be carefully defined, which leakage current is assigned to which gate. In the first place, this may sound obvious, and indeed there is a trivial definition, but as will be presented, this trivial solution introduces major problems when trying to combine gate models to RT component models. The trivial solution would be:

Each leaking electron is assigned to the gate, where it entered the circuit through the supply.

This definition would be simple and sufficient for a single gate, but problems arise, as soon as several gates are combined to a larger structure: With the trivial definition, the leakage current of a gate depends on its fanin and fanout.

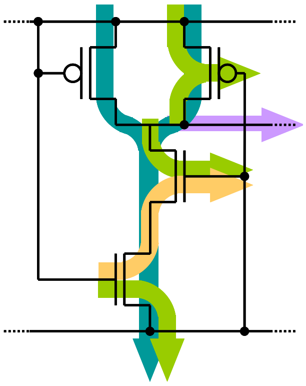


Figure 2.4: Current paths of gate tunnelling and subthreshold through a NAND gate.

In Figure 2.4, the currents going into the lower NMOS gate physically stem from a fanin gate at logic one and are thus flowing though a chain of conducting PMOS transistors outside the logic gate. The gate tunnelling flowing out of the NAND through the upper NMOS gate will flow to ground through one or more conducting fanin NMOS gates, again outside the logic gate. The current flowing out of the gate's output is a result of the fanout's gate leakage. If all currents would be assigned to the gate where they drain off, the total leakage would be easy to determine by simulation, but hard to model, as the leakage of a gate then would depend on the states of all fanin and the fanout gates. This effect will introduce significant errors and is hard to account for [117,118].

All these problems can be avoided, if the assignment of leakage currents to the gates is made more carefully:

Assign the leakage currents to the gate where they are caused, not where they drain off the supply.

Thus the following definitions for a single logic gate are used:

- All absolute current values, flowing through a transistor's gate oxide (in or out) sum up to the tunnelling current of the logic gate, the transistor belongs to.
- All absolute current values flowing into or out of the substrate sum up to the junction leakage of the gate.
- The current flowing from supply, bulk, and all inputs to ground, bulk, and all inputs when the gate is static, is the sum of gate tunnelling, junction leakage and subthreshold leakage of the gate.

Even though these definitions seem to be obvious somehow, they are very important when modelling currents of larger structures, based on the simulation results of single gates. These definitions enable local (gate-wise) computation of the leakage currents and avoid forgetting currents or accounting them twice.

According to the definition above, when determining the gate's currents, one current meter is added to the bulk connections for PMOS and one for NMOS, and one current meter is added to the gate connection per input (for PMOS and NMOS together) as presented in Figure 2.5.

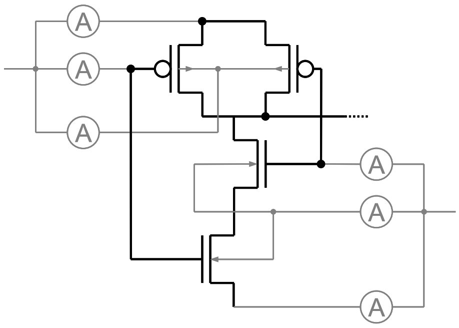


Figure 2.5: In order to obtain a component's gate tunnelling, junction leakage, and subthreshold leakage individually, each input gets one ampere meter, and the NMOS and PMOS bulk get one ampere meter each.

Loading effect:

The loading effect causes, that the gate tunnelling of a large number of fanout gates can affect the output voltage of the driving gate. Some sources [119] report a severe impact on the leakage among gates. As the individual treatment of each gate is conceptually mandatory, the loading effect cannot be regarded in this thesis. In contrast to the analysis of [119], using the PTM technique with the highest gate tunnelling, the $45nm$ oxi-nitride (low-k), shows, that a gate driving ten other gates of the same size looses or gains $1-2mV$ output voltage due to the ratio of the pullup or pulldown on current and the gate tunnelling of the fanout gates. The slightly reduced V_{ds} will not significantly affect the driver itself ($1-2\%$ lower subthreshold leakage), but the weak output voltage will avoid that the driven fanout gates are completely locking. For the $1-2mV$ effect of ten gates load, this will result in a subthreshold leakage increase in all fanout gates of $3-6\%$, and for techniques with lower gate leakage as $\geq 65nm$ or high-k technologies, this effect is even lower.

Nevertheless, in extreme cases, the loading effect may cause a significant estimation error in extreme cases: If a $45nm$ inverter has to drive 50 other inverters, the output voltage will be reduced by $36mV$ or increased by $59mV$. All fanout gates now see $3-6$ times higher subthreshold leakage. As such extreme ratios are extremely rare, the overall estimation error due to the loading effect will again be low.

2.4.3 Improved regression

My first approach, combining the contradictory requirements of being as accurate as the complex transistor models, but small and easy characterisable, and easy to abstract at the same time, is presented in [26]: Using linear regression, each gate is described as a linear combination of four reference transistors; NMOS and PMOS, conducting and locking. While this model performs well for 45nm, 65nm, and 90nm predictive technologies, the first evaluation of a real world's industrial 65nm technology [109] results in large errors, as shown in Figure 2.6.

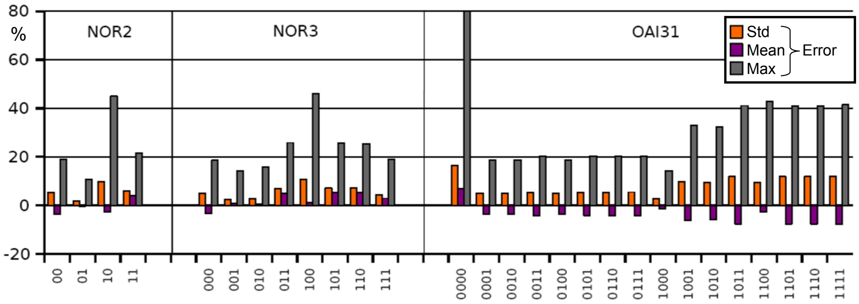


Figure 2.6: Standard deviation, mean and maximum of the relative (percentaged) errors of three gates from an industrial 65nm technology [109] modelled using the a straightforward model.

For the 65nm PTM technology the errors are correlated to the state in which the gate is: The more locking transistors (consecutive zeros for NAND and ones for NOR) are stacked, the higher the mapping error becomes. This behaviour is easy to understand: With each locking transistor at its source, the behaviour of a transistor 'A' inside a gate is controlled more by the body effect. As transistors under source-body potential difference are not characterised, the description of the transistor 'A' becomes more inaccurate. This is because the underlying assumption of the overall gate mapping model is that each transistor inside a gate is approximately in one of the states of the reference transistors. Due to the introduction of the stacked transistor references (**NS** and **PS**, see Figure 2.3), this problem was efficiently reduced.

Nevertheless, the evaluation of a real industry technology shows huge errors: Especially the deviating mean relative error ('mean' in Figure 2.6) which is expected to be very low for regression based models leads to the right assumption: As presented in Figure 2.7, the reason for these higher errors is the higher observable span. Only varying the voltages, the leakage of a gate can change the leakage by a factor of ≈ 50. Temperature introduces a further factor above ten and variations more than factor ≈ 100 [12].

Now, conventional parameter regression was used for the initial model, the gate mapping algorithm tried to minimise the sum of the squared errors. Thus, a 50% deviation at a low leakage model region was nearly completely ignored for the sake of increasing the absolute accuracy in the high leakage regions.

The obvious solution of implementing an algorithm optimising linear parameters with respect to the relative errors is not used. Parameter regression is a well established mathematical algorithm, which can be mapped to a simple geometrical problem and thus be explained by a matrix inversion.

[12] A factor ≈ 30 is expected due to inter-die correlations, but atop of this, random intra-die variation can again increase this deviation.

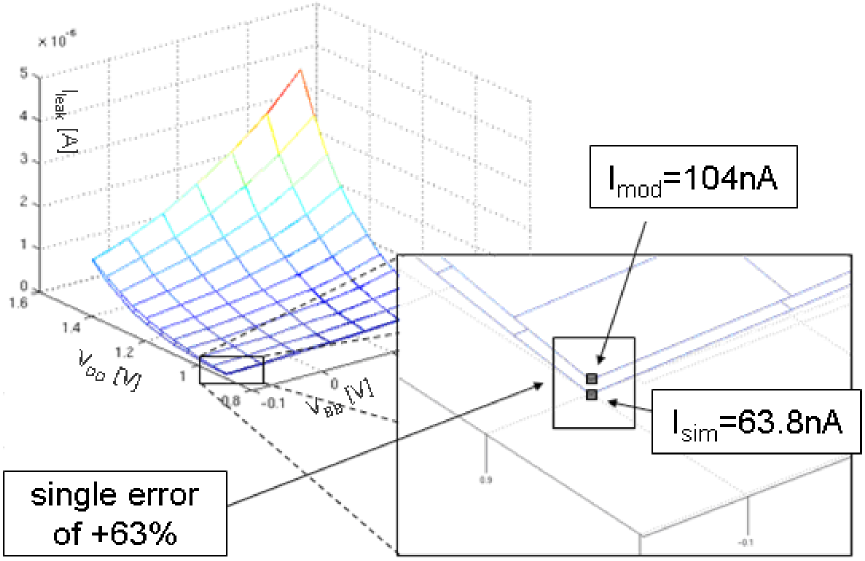


Figure 2.7: The increased difference between the lowest and the highest overall leakage current causes instability when applying parameter regression. Small values are considered less than higher ones resulting in a huge error.

For matrix inversions, very powerful algorithms having low time complexity and an integrated control of the numerical stability exist.

Instead, the mapping problem is reformulated, to be solvable by conventional linear parameter regression: With $f(\vec{x}_j)$ being the simulated gate leakage, $\vec{x}_j$ being the parameters ($DyPa$, and $StPa$) and $t_i(\vec{x}_j)$ being the reference transistors, the following equation has to be solved:

$$\text{Optimize}\,\alpha_i\,\text{so that}\sum_j\left(\frac{f(\vec{x}_j)-\sum_i\alpha_i\cdot t_i(\vec{x}_j)}{f(\vec{x}_j)}\right)^2\text{is minimal}\tag{2.12}$$

Regression just offers:

$$\text{Optimize}\,\alpha_i\,\text{so that}\sum_j\left(f(\vec{x}_j)-\sum_i\alpha_i\cdot t_i(\vec{x}_j)\right)^2\text{is minimal}\tag{2.13}$$

By reformulating the problem to:

$$\text{Optimize}\,\alpha_i\,\text{so that}\sum_j\left(\ln\left(f(\vec{x}_j)\right)-\sum_i\alpha_i\cdot\ln\left(t_i(\vec{x}_j)\right)\right)^2\text{is minimal}\tag{2.14}$$

The final model prediction $g(\vec{x}_j)$ is

$$g(\vec{x}_j)=e^{\sum_i\alpha_i\cdot\ln(t_i(\vec{x}_j))}=\prod_i t_i(\vec{x}_j)^{\alpha_i}\tag{2.15}$$

The logarithm, introduced to the problem formulation leads to the effect, that a relative deviation $g=f\cdot\delta$ will lead to an absolute shift in the cost function, thus the regular regression can be used:

$$\ln(g)-\ln(f)=\ln(f\cdot\delta)-\ln(f)=(\ln(f)+\ln(\delta))-\ln(f)=\ln(\delta)\tag{2.16}$$

As presented in Figure 2.8, the changed cost function improves maximum error and standard deviation of the relative error. Both errors now again are highest for large stacks of locking transistors.

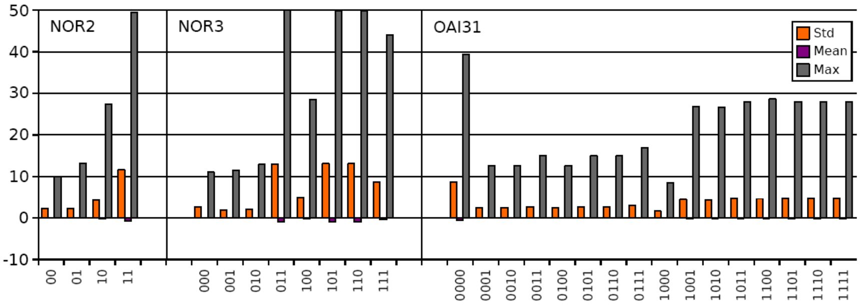


Figure 2.8: Standard deviation, mean and maximum errors of three gates from a $65nm$ industrial technology. The initial reference transistors are still used, but this time, the scaling parameters are obtained for the exponents of the simulated values to weight small values as much as large ones. Maximum errors are reduced about 30%, the mean relative errors remain at less than 1%.

2.4.4 Conclusion

The outcome of the gate leakage model stage is as follows: For a given technology library, each gate is either directly simulated (if transistor data in SPICE is available) or constructed by analysis of the gate properties as will be described in Section 4.3. Using the improved regression, the input-dependent leakage simulation results of the gates are translated into eight scaling parameters.

Having available the accurate description of the eight references and the eight scaling parameters per gate per state, each logic gate of a technology's leakage currents can be predicted.

2.5 RT level model

The RT level model is the final abstraction layer of the bottom up leakage model. As will be presented in Section 2.5.1, due to the good preparation at the transistor and gate level stages, it is comparatively simple, to combine the gate level models to hard models. In order to obtain more general soft models, both, the explicit data dependency, as well as the explicit bitwidth dependency have to be abstracted as described in Section 2.5.2. The resulting soft models are already the final RT level leakage models. Section 2.5.3 gives an outlook and discusses limitations of the final model and potential extensions.

2.5.1 Hard model generation

The task described here is to predict the leakage currents of an RT component having available

- a netlist of gates, representing an RT component in Verilog format

- the technology library of the used gates in Liberty (.lib) format
- the state of each primary input
- the physical operation conditions, such as supply voltage, temperature, body bias potential at NMOS and PMOS well
- the absolute and concrete value of deterministic process variations (assuming, that gradients within the component are negligible)
- the statistics of uncorrelated process variation
- and a gate level leakage model for each gate within the netlist as described in Section 2.4.

The hard model generation starts with parsing the Verilog netlist of gates and translating it into a directed graph, which does not necessarily have to be acyclic. For each node (i.e. gate), the Liberty library entry is parsed to obtain its Boolean logic table. Assigning the logic states to the primary inputs of the component, it is straight-forward to obtain zero-delay simulation results for the component, as long as it is not sequential (as long as its graph is acyclic), by simply evaluating each gate's logic table as soon as all input states are known. The treatment of sequential circuits is discussed later.

Having available the input state of each gate, the gate-level models can directly be evaluated. Now, the linearity of the gate-models is exploited. Instead of a gate-wise linear combination of the leakage currents

$$I_{tot} = \sum_{i}^{\text{all gates}} \alpha_{NC}(i) \cdot I_{NC}(L,T,N,\vartheta,V_{\text{DD}},V_{\text{BB}},i) + \alpha_{N0}(i) \cdot I_{NC}(L,T,N,\vartheta,V_{\text{DD}},V_{\text{BB}},i) + \ldots + \alpha_{PS}(i) \cdot I_{PS}(L,T,N,\vartheta,V_{\text{DD}},V_{\text{BB}},i) \quad (2.17)$$

it is assumed, that all of the six *DyPa* and *StPa* are constant for all the gates within one RT component, or at least, that the deviations are negligible. As in Equation 2.17, the leakage of the references is no longer depending on the summation variable, the associativity of the function can be exploited resulting in

$$I_{tot} = I_{NC} \cdot \sum_i \alpha_{NC}(i) + I_{NC} \cdot \sum_i \alpha_{N0}(i) + \ldots + I_{PS} \cdot \sum_i \alpha_{PS}(i) \quad (2.18)$$

where

$$I_{NC} = I_{NC}(L_n,T_n,N_n,\vartheta,V_{\text{DD}},V_{BB,n})\,,\ I_{N0} = I_{N0}(L_n,T_n,N_n,\vartheta,V_{\text{DD}},V_{BB,n})\,,\ \ldots\,,\ I_{PS} = I_{PS}(L_p,T_p,N_p,\vartheta,V_{\text{DD}},V_{BB,p}) \quad (2.19)$$

Each of the eight references is independent of the gates, but the variables (at least the body voltage) are not independent of the reference. Usually, the body voltage of NMOS and PMOS can be controlled independently (in contrast to the supply voltage or the temperature). The channel length, oxide thickness, and channel doping are also typically not equal for NMOS and PMOS[13].

As a result, an entire RT component can be described after zero-delay simulation as a *super-gate* with much larger scaling parameters (the effective width of the references for a 16 bit multiplier can be in the order of a millimetre, see Figures 5.6 and 5.7). The scaling parameters are completely independent from the currents of the reference transistors, thus the architecture and input dependency completely separates from the physical and process dependency.

Discussion of sequential circuits

The entire zero-delay simulation framework was developed as a proof of concept, in order to show, that each netlist's leakage (sequential or not) can be accurately predicted, if the state of each gate is known. The problem is, that for sequential circuits, the determination of the state of the gates may be complicated or dependent on the history or even impossible.

In order to not exclude sequentials just because of the lack of simulation capability, a straightforward modification of the simulator was made, enabling simulation of a large class of sequential circuits:

Sequentials are identified easily: All primary inputs are assigned to the respective gate inputs, and after each data assignment, it is checked, if the gate output is now determined. The logic result is then assigned to the fanout gates. After this simulation terminates, all acyclic and all determinable cyclic gates are evaluated (see Figure 2.9). One indeterminable gate is arbitrarily chosen and its sequence is cut introducing a virtual input and output. The user has to constrain a value to the virtual input[14] and the simulation continues. Eventually additional, nested sequences are found and solved. Finally, it is checked, that the virtual outputs equal the respective virtual inputs. If not, the user is warned that a non-stable sequential circuit (such as a phase locked loop PLL) is found.

In any case, the leakage computation can be done. For stable sequentials, the inner states can be abstracted away by ignoring the virtual inputs during abstraction (see Section 2.5.2)

My colleague Marko Hoyer developed in his diploma thesis [120] a framework, called RTSim (see Figure 5.2), which realizes all the concepts, mentioned here. The RTSim also contains a much more elaborated gate simulator, which is not described here, as it is conceptually not my, but his work.

2.5.2 Data and bitwidth abstraction

The modelling flow till here enables leakage prediction of entire RT components, as long as their Verilog description is available and the exact input data is known. A preliminary result of each modelling run is the computation of the cumulative scaling parameters (the total effective widths) for the eight reference parameters.

It is the task of the data and bitwidth abstraction part to find a method computing these eight scaling parameters from abstract component metrics as bitwidth or input data signal statistics. Time

[13] In PMOS, the oxide is usually slightly thicker and the doping concentration is much lower.

[14] the *user* in the later characterisation flow is another program

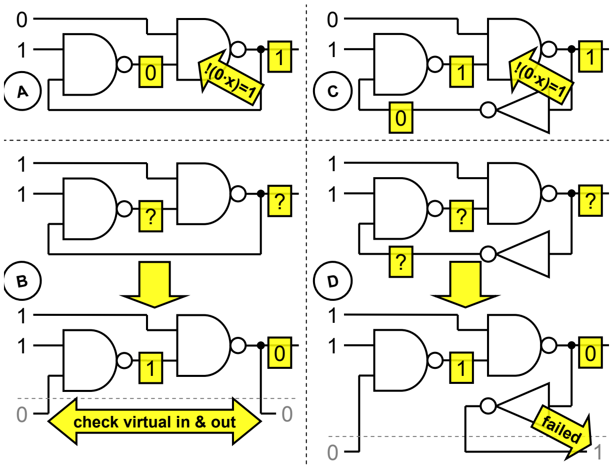


Figure 2.9: **A:** Sequential circuit w/o history dependence due to state. **B:** Sequential circuit with history dependence. One arbitrary node is cut, generating a virtual input and output pair. During simulation, it is checked, that this pair is consistent. **C:** Depending on the input state, this sequential circuit may be unstable. Nevertheless, it passes simulation, if the sequence is not active. **D:** For the same circuit with other input data, the check of virtual in- and output fails, indicating, that the sequence is not stable. Leakage estimation can nevertheless continue with this temporary value, but the simulator throws a user warning.

consuming gate level simulation or even having the exact gate-level netlists will then no longer be needed. Not till then, the models are really RT level, as for the hard model, gate level detail information (simulation and netlist) is always needed[15].

In order to find and evaluate such a method, computing the parameters, several RT components are synthesised in a large range of different bitwidths and then simulated with input data of different input characteristics. For each of these simulation runs, not the total leakage, but the eight scaling parameters (which are fortunately perfectly independent of *DyPa* and *StPa*) are stored separately. Figures 5.6 and 5.7 on page 125 show examples of the behaviour of the scaling parameters. These main observations can be made for all components:

- Even though the state dependency of an entire RT components is known to be just around $10 - 15\%$ standard deviation, the data dependency of the single scaling parameters is much higher. Data dependency has to be definitely regarded. This is not contradictory: Typical gates show a significant change in leakage only for one special state (e.g. in the 000 state, a NAND3 gate is leaking much less, than in all other states). Due to the high correlation, an RT component can usually not keep all its gates in such an extreme state. In contrast, the scaling parameters are significantly diverging for each state of the gate. For instance, the N1 leakage

[15] Note, that the resulting models are still technology dependent; they are free from any gate level details, but valid just for the technology, they have been characterised for.

(gate leakage NMOS when gate is at V_{DD}) is twice as high for a NAND3 gate in 110 state that in 101 state and nearly 0 in 011 state.

- The scaling parameters show a high correlation to the number of ones and zeros at the primary inputs, thus the signal probability p is a promising abstraction.
- The behaviour of the parameters on p is usually not linear and the maximum is neither at $p = 0$ nor at $p = 1$.
- After several trials, it turned out, that a quadratic approximation shows sufficient results: A function $\alpha_2 \cdot p^2 + \alpha_1 \cdot p + \alpha_0$ is adequate to describe the data dependency for each bitwidth at each component. Further data abstraction is not necessary.
- The α_0 to α_2 parameters are similar for components with similar bitwidth and develop slowly and smoothly with bitwidth. Thus, the function $\alpha_2(\mathrm{bw}) \cdot p^2 + \alpha_1(\mathrm{bw}) \cdot p + \alpha_0(\mathrm{bw})$ is adequate to describe the data dependency over all bitwidths.
- For one fixed signal probability, the parameters scale approximately linearly with the bitwidth for components, with only one input (as incrementers) and components, where only one bitwidth can be specified (as adders).
- For components with two inputs, where two bitwidths can be separately specified (such as multipliers), the parameters scale with the product of the bitwidths.
- This leads to the (natural) assumption, that the parameters linearly depend on the area of the design. This leads to the following approach: $\alpha_i(\mathrm{A}, \mathrm{B}) = \beta_N + \beta_A \cdot A + \beta_B \cdot B + \beta_S \cdot A \cdot B$, where A and B are the bitwidths of the inputs A and B. For components with only one or three inputs, this equation is modified accordingly.

Due to these observations, the following resulting equation was chosen to describe the combined data and bitwidth behaviour

$$\begin{aligned}
S^i &= \alpha^i_{0N} + \alpha^i_{1N} \cdot p + \alpha^i_{2N} \cdot p^2 \;+\; \alpha^i_{0A} \cdot A + \alpha^i_{1A} \cdot pA + \alpha^i_{2A} \cdot p^2 A \\
&+ \alpha^i_{0B} \cdot B + \alpha^i_{1B} \cdot pB + \alpha^i_{2B} \cdot p^2 B \;+\; \alpha^i_{0S} \cdot AB + \alpha^i_{1S} \cdot pAB + \alpha^i_{2S} \cdot p^2 AB \\
i &\in \{\mathrm{NC}, \mathrm{N0}, \mathrm{N1}, \mathrm{NS}, \mathrm{PC}, \mathrm{P0}, \mathrm{P1}, \mathrm{PS}\}
\end{aligned}$$

where the twelve α parameters are determined by characterisation and the S^i are the eight scaling parameters of the component.

Remarks

The virtual inputs, generated during simulation to enable sequential, history dependent circuit simulation are controlled while test data generation, but ignored during abstraction. That way, the history dependency of such components is abstracted away as good as possible.

2.5.3 Limitations and potential extensions

The model as presented is capable of predicting the leakage currents of single RT components. Entire systems can be estimated by synthesising them to RT level and then using the RT models component for component. The main limitation, why the modelling itself cannot be performed further to higher levels ending up with models for IP-blocks, memory-arrays, processors and controllers, or even entire systems is correlation. In order to proceed from Equation 2.17, which is mathematically equivalent to the gate level models to Equation 2.18, where each component can be described as a *super gate*, it had to be assumed, that both, the *DyPa*, as well as the *StPa* [16], are perfectly constant for all gates in one RT component.

Each *StPa* or *DyPa* variation introduces an additional error to the model. For gradients (linear local variation of a parameter), this effect is of second order (see Equation 2.22); and for small die areas in the size of RT components, it is valid to linearly approximate the variations. But for larger areas, allocating significant die area, the parameters may show an exotic behaviour which is by far not linear[17].

Assuming, that due to correlation of the *StPa*, these parameters in a small section (an RT component) of size $X \cdot Y$ are not constant, but behave like

$$p(x,y) = p_{ctr} + \Delta_x \cdot \frac{x}{X/2} + \Delta_y \cdot \frac{y}{Y/2}\,,\; x \in [-X/2, X/2]\,,\; y \in [-Y/2, Y/2] \tag{2.20}$$

where the parameter has an exponential influence to the result (leakage current) as $I = e^{\alpha p}$, the error introduced by using the average (centre) parameter value p_{ctr} for all parameters is

$$E\left(I\left(p\left(x,y\right)\right)\right) - E\left(I\left(p_{ctr}\right)\right) = \frac{1}{XY}\int_{-X/2}^{X/2} dx \int_{-Y/2}^{Y/2} dy e^{\alpha p(x,y)} - e^{\alpha p_{ctr}} \tag{2.21}$$

$$= e^{\alpha p_{ctr}} \cdot \left(\frac{1}{XY}\int_{-X/2}^{X/2} dx e^{2\alpha\Delta_x x/X} \int_{-Y/2}^{Y/2} dy e^{2\alpha\Delta_x y/Y} - 1 \right)$$

$$= e^{\alpha p_{ctr}} \cdot \left(\frac{\sinh(\alpha\Delta_x)\sinh(\alpha\Delta_y)}{\alpha^2\Delta_x\Delta_y} - 1 \right)$$

$$= e^{\alpha p_{ctr}} \cdot \left(\frac{(\alpha\Delta_x + 1/6(\alpha\Delta_x)^3 + \ldots)(\alpha\Delta_y + 1/6(\alpha\Delta_y)^3 + \ldots)}{\alpha^2\Delta_x\Delta_y} - 1 \right)$$

$$= e^{\alpha p_{ctr}} \cdot \left(\alpha^2/6(\Delta_x^2 + \Delta_y^2) + \mathcal{O}((\alpha\Delta)^4) \right) \tag{2.22}$$

For moderate gradients, Equation 2.22 is very small. If for instance $\alpha\Delta_x = \alpha\Delta_y = 0.1$ is chosen, the upper right corner has 49.1% higher leakage than the lower left corner. Nevertheless, as long as the variations are linear, the real expectation value is only 0.33% higher than the function value in the centre.

[16] This assumption refers to correlated parameter variations, even the small ranged ones. Fully uncorrelated variation are regarded independently.

[17] Such as thermal hot spots or circular parameter variations. Nevertheless, a small section of a heat-map with a hotspot or of a circular parameter variation is almost linear.

To conclude, correlated variations with intra-die range are the only limitation to the model framework. Only components for which the variation can be assumed to be either marginal, or perfectly linear can be described with one single model. Larger structures have to be broken down into smaller components before simulation. For homogeneous structures as memory arrays, a tile-based application is also possible.

2.6 Delay modelling

In sub-100nm technologies, the total leakage of a system and its critical path delay are strictly correlated, as both (leakage and delay) depend on variable parameters as threshold voltage, supply- and body-voltage, and temperature. Even though the focus of this work is on leakage modelling, this strong correlation requires, that leakage and delay are regarded simultaneously.

Thus, this section summarises, how an RT level delay model, providing the according delay predictions to each leakage prediction, can be developed. The delay model was developed in parallel to the leakage model in collaboration with my colleague Marko Hoyer, and is not my conceptual work. Nevertheless, it is briefly outlined here to describe the integration into the entire framework. In Section 3.2, a variation engine will be presented, which uses the leakage model described above and the delay model described here.

2.6.1 Bottom up delay modelling

In order to generate a delay model, in a first step, a standard inverter of the respective technology is generated in SPICE and its rise and fall delays are simulated under several parameter conditions as varying *StPa* and *DyPa*, as well as input slew and output load. The delays are simply saved in a table.

The gate level abstraction [121,122] bases on the assumption, that rise and fall times of each input pin for each gate in each state can be described by an affine projection of the delays from an inverter (rf. Figure 2.10). By selecting the worst-case delay for all states of the other inputs the resulting model is a per pin projection of the inverter delay as

$$D_i^{gate} = k_i \cdot (D_{inv} + d_i) \cdot (C_{FO} + c_i) \cdot \left(T^2 + t_i\right) \tag{2.23}$$

where D_i^{gate} is the worst delay for pin i for all states of the other pins, C_{FO} and T are the gate's fanout load and temperature and D_{inv} is the inverter delay model capturing process variation, slew, load, and temperature behaviour in a table. All other parameters are determined by characterisation.

In order to obtain this characterisation data, rise and fall time of each input from each gate in the library are simulated in SPICE. The input slope dependency can at gate level be avoided by letting an inverter produce a standard slope, which is driving the gate under test.

The result is a delay model for a technology library with implicit (inside the reference inverter) dependency on *DyPa*, *StPa*, and fanout load and explicit dependency on fanout load and temperature.

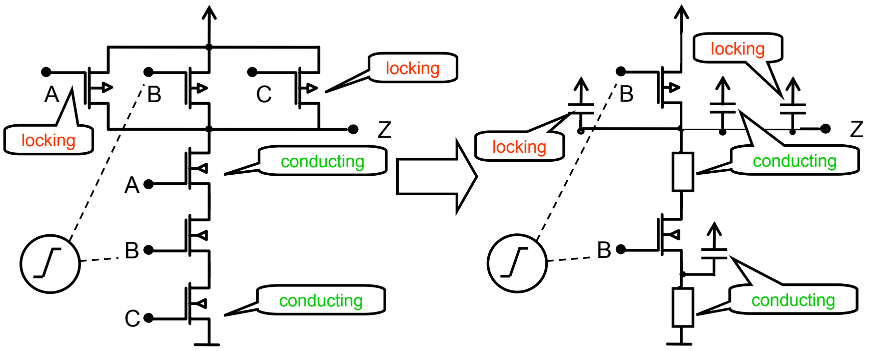


Figure 2.10: When switching a CMOS transistor pair inside a gate, all other transistors behave like capacitances when locking and like resistances followed by capacitances when conducting. Exactly knowing the inverter dependency on *StPa*, temperature and fanout load, all gates can be modelled under PTV variation after a characterisation based mapping onto the inverter.

In equivalence to the leakage model, most physical parameter dependencies are hidden inside the lowest level model. For the delay model, not eight transistors but a single standard inverter is this reference.

2.6.2 Temperature dependent critical paths

If an entire RT netlist's delay has to be predicted, for each gate, the output load C_{FO} is known, and a reduced delay model

$$\begin{aligned} D_i^{gate} &= k_i' \cdot (D_{inv} + d_i) \cdot \left(T^2 + t_i\right) && (2.24) \\ &= f_0\left(D_{inv}\right) + f_1\left(D_{inv}\right) \cdot T^2 && (2.25) \end{aligned}$$

can be assigned to each gate. Within the zero delay simulation framework, described in Section 2.5.1, in addition to the simulation data, the f_i parameters can be summed up over each path. Both f_i parameter can have their worst case on a different paths. Thus, for each parameter, the maximum of all paths is chosen. In the following, it will be motivated, why the delay model is not directly generated for the critical path:

Not all gates show a proportional performance degradation with rising temperature. Instead, in recent 65nm technologies, very complex effects have been reported [123]. The two dominant parameters influencing the temperature-performance behaviour are carrier mobility which is decreasing, and thus reducing performance with higher temperatures; and threshold voltage which is decreasing and thus increasing performance with higher temperatures. In pre-100nm technologies, carrier mobility clearly dominated the performance and hotter devices used to be always slower. But with technology scaling, the influence of the thermal threshold voltage behaviour becomes more important resulting in devices which are faster, the hotter they are and even non-monotonous devices which are slowest at an intermediate temperature and faster when hotter or colder.

When combining the per gate models to RT components, the effects mentioned above can lead to different timing for different temperatures for each path, and the resulting models have to handle critical paths alternating with temperature. By selecting the worst case f_0 and f_1 for (potentially) different paths the following behaviour can be achieved: If the critical path is for all temperatures the same, both f_is will be from this critical path, resulting in a conventional delay modelling. If there are two paths, where one is the slowest at low temperatures, and one is the slowest at high temperatures, the f_0 is taken from the first path and the f_1 is taken from the latter path, and the entire delay model is slightly overestimating (but never underestimating) the delay. For all RT components evaluated [121], the delay can be estimated regarding all PTV effects with 1.8-4.6% standard deviation against a Monte-Carlo transistor level evaluation.

If now, in a final step, both f_i can be predicted by an equation, depending on the component's bitwidth, an RT level soft macro model for delay is obtained.

2.6.3 Statistical statical timing analysis

Both models, for delay and leakage base on a low level reference, capturing the dependency on the physical parameters. If varying parameters are entering these references, global deterministic parameter variations can be directly described. Uncorrelated variations cannot be described by the models.

For the leakage model, this problem is fixed by regarding the influence of the variation on the expectation value of the prediction. It is exploited, that the total leakage is the sum of a large number of instances. In contrast, the delay is the sum of only a few gates on the critical path, and if there is more than one critical path, only the worst path is relevant:

$$I_{leak} = \sum_{\textbf{many gates}}^{N} I_{gate} = N \cdot E\left(I_{gate}\right) \tag{2.26}$$

$$t_{crit} = \max_{\textbf{many paths}}^{M} \sum_{\textbf{few gates}}^{n} t_{gate} \tag{2.27}$$

A statistical variation of the leakage will always average out, but with a high probability, there will be some paths, for which accidently most gates show a correlated delay deviation. This behaviour cannot be captured by computing the expectation value. Instead, a statistical statical timing analysis (SSTA) will have to be integrated into the framework. The integration of an SSTA into the delay model, and especially its abstraction into a soft macro model is subject of ongoing research in my group. So far, the effect of statistical variation on the delay of an RT component cannot be predicted.

2.7 Model splitting

Among the leakage optimisation techniques (Section 1.4), that have to be controlled from a RT level or higher view, the most promising ones are surely power gating, body biasing, voltage scaling, and input vector control. The related *DyPa* as body voltage, are explicit model parameters, and at least the *best* state for each RT component (will be presented in Section 5.1.4) can be explicitly regarded. The only important optimisation technique, not supported by the framework, as described so far, is power gating.

Power gating modelling can be introduced by a model splitting approach. This idea is conceptually very simple. Each RT component is described by three instead of one model; the first model describes the RT component if no power gating is implemented, the second model describes the same component, if power gating is implemented, but not used (i.e. the component is active and its power gates are conducting), the last model describes this component, if in power gated condition. If additionally, the energy consumed for going to sleep and for waking up, and the wakeup latency is stored, an RT component under power gating can in principle be described. The idea of modelling leakage control techniques by a model splitting and an overhead accounting was already reported in [10], where it is used for the leakage estimation engine of the HotLeakage tool.

The basic model splitting and overhead cost assumption are visualised in Figure 2.11. Additionally, the area, and the component delay are modelled twice; once for the component without power-gating and once for the component with power gating.

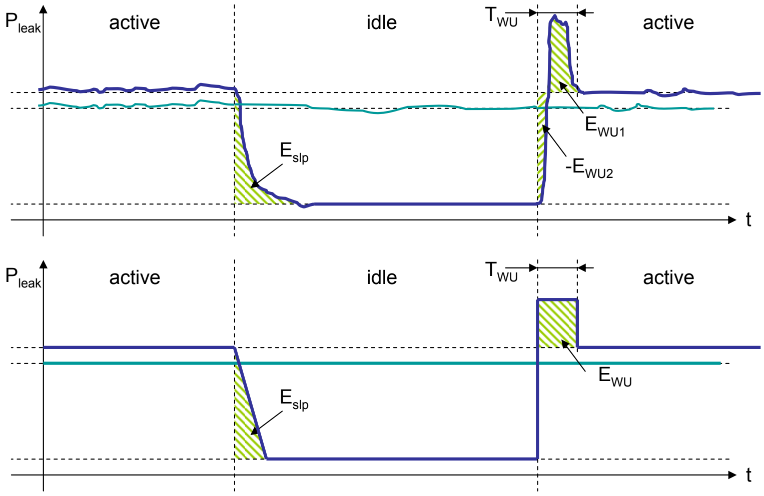


Figure 2.11: Visualisation of the power gating model: For simulated data (top figure), it can be observed, that the availability of power gating increases the leakage power in active mode. As soon as power gating can be activated (idle phase of the component), the leakage power can be significantly reduced. In order to model this behaviour, the energy overhead for falling asleep and waking up again, as well as the wakeup delay are simulated. For falling asleep, the energy cost is linear decreasingly distributed over the time $E_{slp}/2(P_{active} - P_{idle})$. The wakeup energy results in a power consumption of $P_{active} + E_{WU}/T_{WU}$.

Even though it is conceptually simple to integrate power gating by model splitting, from a technical point of view, modelling power gating is not simple at all. The following work was developed by my colleague Sven Rosinger and is neither my implementational, nor my conceptual work. It is mentioned here simply to pronounce, that each of the relevant leakage optimisation techniques can be described.

When trying to describe the effect of power gating, the problems arise from technical details. All relevant metrics:

- the extra power cost for power gating, when the component is active,
- the idle power,
- the sleep and wakeup energy,
- the wakeup delay

are depending on technical details as

- the driving strength (i.e. width of the gating transistor),
- the gating transistors threshold voltage,
- the power gating architecture (NMOS vs. PMOS, single vs. double cutoff, regular vs. super cutoff).

even if the user of such a model (being a user of PowerOpt, thus working at system level) might be able to specify technical implementation details, especially the width of the gating transistor should not be arbitrarily chosen, but optimised for the component and to design constraints.

The power gating model (described in [86]), will thus have to start with a width prediction according to performance constraints. Then all relevant metrics have to be computed depending on this width. Finally, the model can be used as described in Figure 2.11

3 Parameter determination

The essential difference between the well known dynamic power and leakage RT macro models is, that function like modelling is no longer enough for leakage estimation. Dynamic power models, once characterised, need information like input bitwidth, input bit vectors, clock frequency, or supply voltage. All these parameters are available at RT level. In contrast, the leakage models developed in Chapter 2 need detailed information on parameters which are far out of the scope of an RT level designer, like oxide thickness distribution, back-bias, or V_{DD}-noise. Thus, an RT level leakage modelling methodology needs to have additional functionality determining these parameters from abstract metrics available at RT level.

In this chapter, the PowerOpt framework is briefly presented. Afterwards, the modules developed for parameter determination are presented in detail. Section 3.2 presents the variation engine, in Sections 3.3 and 3.4, the thermal and power grid computation are described. In order to get body bias values, ABB optimisation has to be planned as described in Section 3.5.

Having all variation data and other important parameters available, the leakage and delay models can be used to predict the power and performance distribution, enabling a forecast of the functional yield as presented in Section 3.6.

3.1 Framework overview

To simplify the understanding of the next sections, this section gives an overview over the PowerOpt framework and the modules, developed to support the model.

The modelling methodology has been developed to be used inside the PowerOpt framework (Figure 3.1), which can read in SystemC description of a complete system and transform it into a CDFG (control data flow graph), which is the most general way of representing an algorithm. Based on this CDFG, each operation is mapped to an operator (binding and allocation), to a time (scheduling), and to a place (coarse RT floorplanning). Recent development in this area is presented in [111]. After these steps, the position of each RT component is known.

By simulating the algorithm with user-defined application data, data-traces for all *operations* can be obtained. Regarding binding and scheduling data-traces for each *operator* (i.e. RT component) are generated. This enables RT level estimation of the dynamic and short-circuit power depending on the input transitions and the supply voltage[1] [124]. The dynamic and short circuit power consumption,

[1] Even though the overall modelling approach will regard further parameters, data and supply voltage is sufficient here, as dynamic and short-circuit power are nearly independent from temperature, body-voltage, or variation.

together with the position of each component can be combined with package information to compute an initial thermal and V_{DD}-drop map. The variation engine generates simulation instances with a varying inter-die and a characteristic intra-die variation. Having the body-bias, optimised for the system, initial leakage power and delay estimates can be computed and reiterated.

After the initial leakage power computation, the temperature (Section 3.3) and IR-drop (Section 3.4) maps must be re-iterated, to accurately regard the effect of electro-thermal coupling [4]. The ABB-optimisation methodology (Section 3.5) adapts to the results of this inner-most iteration. Finally, the entire behavioural synthesis may by iteratively updated, trying to optimise total power and yield.

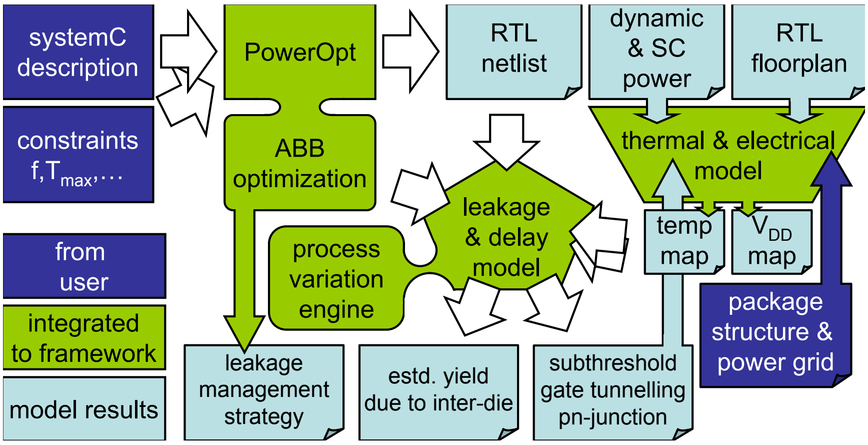


Figure 3.1: Model application flow: A SystemC description of the design is fed into the synthesiser, generating RT netlist together with estimates of the dynamic and short circuit power and a coarse floorplan. For each component of the RT netlist, the temperature and V_{DD} are computed, regarding power and floorplan. Together with inter-die and intra-die variation data, the leakage [26] and delay [121] models can predict estimates of the systems total power and the performance.

3.2 Variation engine

In sub-100nm technologies, leakage and delay show a high variation with a nearly perfect correlation due to global deterministic *StPa* variation as shown in Figure 1.10 on page 21. On one hand, this correlation has a significant influence on the parametric yield[2]. On the other hand, there are various optimisation methodologies exploiting the correlation by saving power for the fast systems and speeding up the low leakage systems. To accurately predict yield and to support these adaptive management techniques, it is mandatory to accurately regard the correlation of leakage and delay.

[2] Example: Let 70% of all systems meet its power budget, and 50% be fast enough. Without correlation, one could expect a parametric yield of $70\% \cdot 50\% = 35\%$. But if leakage and delay are positively correlated, there is not a single system, which is too slow and has to high power. Thus, the parametric yield is $100\% - 30\% - 50\% = 20\%$.

3.2.1 Requirement specification

The intention of the leakage and delay prediction methodologies presented in Chapter 2 is as follows: Both, leakage and delay are described as accurate as possible with respect to all relevant parameters, especially global deterministic *StPa*. If now both models are evaluated for a large number of realistic *StPa* samples, a large number of leakage-delay prediction samples can be obtained, automatically carrying the right correlation. In this section, the methodology, how the models can be fed with samples of realistic *StPa* values will be presented.

A first intuitive variation description was done by simply assuming a Gaussian distribution of all *StPa*. This straight-forward understanding of variation was motivated by IBM work [3, 17], in which three forms of variation (global deterministic, local statistic, and over-die gradients) are identified. Also other researchers [29] presented simple estimation approaches relying on a first order dependency analysis of the leakage and the delay on the *StPa* (called parameter sensitivity). In each of these approaches, the three forms of variation were assumed to be uncorrelated for each parameter[3], and additionally, the correlation among the parameters was ignored[4]. In contrast, [44] presents recent IBM research results, showing that the relevant *StPa* are typically not Gaussian distributed. Additionally, the IBM technology shows a more or less strong correlation among each pair of *StPa*.

Besides having more sophisticated and accurate leakage and delay estimation methodologies, regarding more influences, the exact treatment of the correlation between *StPa* and the correlation between local variation and global variation is the core advantage compared to the proposed approaches from IBM / University Michigan [3, 17] and University Minnesota [29].

3.2.2 Engine description

To cover all the requirements, mentioned above, the variation engine was designed as presented in Figure 3.2. A generating methodology from industry is evaluated inside a Monte Carlo loop resulting in a number of global *StPa* samples, which is large enough to also capture a sufficient number of very unlikely parameter combinations. This is necessary to have a stable prediction also for a later yield prediction close to 100%[5]. For each sample, the local statistic parameter variation is determined if it is not independent from the magnitude of the global parameter variations. If the distribution of over-die gradients also depends on the recent global variation, these values are stored for later use.

Each sample of global parameters and (possibly individual) local parameters is then evaluated by both, the leakage and the delay model of a single inverter, resulting in a large number of leakage-delay samples representing the influence on inter- and intra-die variation. This step uses the assumption, that a parameter combination increasing the leakage of a single inverter will approximately have the

[3] That means for instance, that the magnitude of the local channel length variation on one die is assumed to be independent from the global variation of the channel length on exactly this die.

[4] That means for instance, that the probability of a certain channel doping deviation is meant to be independent from the recent channel length variation.

[5] In order to see the difference between 98% and 99% of all devices meeting their constraints, a significant number of samples has to lie between 98% and 99%.

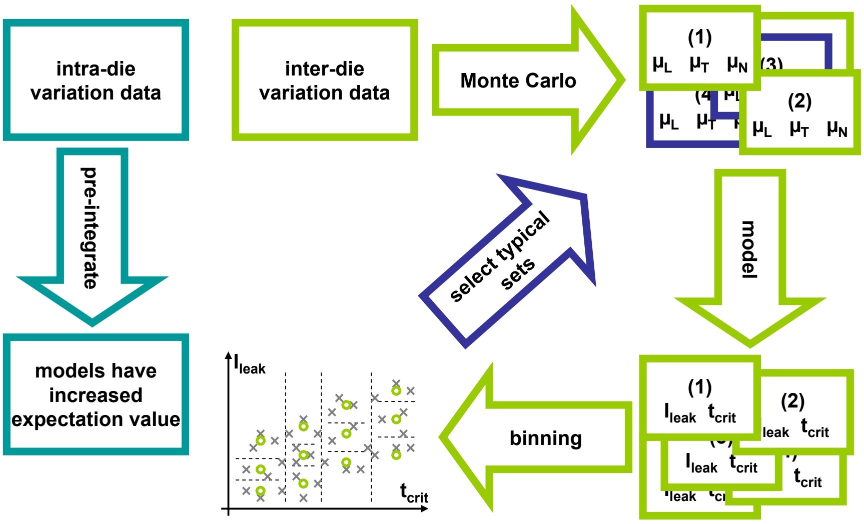


Figure 3.2: Visualisation of the process variation engine. The industrial methodology generating global parameter variation is evaluated in a Monte Carlo loop and evaluated in nanoseconds by the models. The samples are binned by delay and sub-binned regarding leakage. For each sub-bin, containing the same number of samples, one typical instance is chosen, its parameters are back-tracked and stored in a small list of typical *StPa*.

same effect on the leakage of a large RT structure.

All leakage-delay samples are then binned into equi-probable delay classes. This means that all N samples are assigned to one of D classes, so that finally, each class is populated by N/D samples and that the delay of each sample in class i is smaller than the delay of each sample in class $i+1$ for all $i \in [0, \ldots, D-2]$. Inside each class, the remaining N/D samples are binned again (called sub-binning) into L sub-bins with respect to their leakage in the same way as described above. As a result, $D \cdot L$ classes are obtained, each populated by $N/(D \cdot L)$ samples.

For all samples inside a certain delay bin and leakage sub-bin, the average delay and leakage are computed. For each sample, the distance to the average is computed as the geometric average of the relative deviations as follows:

$$I_{\text{mean}} = \frac{D \cdot L}{N} \cdot \sum_{j=0}^{N/(D \cdot L)-1} I_j \tag{3.1}$$

$$T_{\text{mean}} = \frac{D \cdot L}{N} \cdot \sum_{j=0}^{N/(D \cdot L)-1} T_j \tag{3.2}$$

$$\text{dist}_j^2 = \frac{(I_j - I_{\text{mean}})^2}{(I_{\text{mean}})^2} + \frac{(D_j - D_{\text{mean}})^2}{(D_{\text{mean}})^2} \tag{3.3}$$

Finally, for each sub-bin, the sample with the smallest distance is chosen, and its parameters (global deterministic, local statistic and over-die gradient) are stored in a table. At model evaluation time, the entire system is evaluated multiple times (Section 3.5 details different possible application of this evaluation) for each parameter set stored in this table. As each result can be assumed to be as likely as each other, because it represents the same number of Monte-Carlo samples of the real technology, the parametric yield can be determined by simply counting the number of evaluations meeting and not meeting the required specifications. In the conclusions of this chapter, the variation engine is presented embedded into the entire parameter determination and model evaluation flow (Figure 3.8).

3.2.3 Remarks

The effect, that local deterministic variations depend on the recent global values (e.g. the larger the channel length for all instances, the smaller the random dopant variation is per transistor) is accurately regarded.

If over-die parameter gradients also depend on the recent global variation, this effect is regarded by storing the gradients for the chosen samples and using them for model evaluation, but the samples in one sub-bin were chosen to end up with the same overall leakage and delay. Each instance may have totally different gradients. If for example, in the worst delay bin, best leakage sub-bin, the sample with the lowest distance has an extreme gradient, this corner case may become even worse due to a high over-die gradient. Thus, the correlation between global variation and parameter gradients is just statistically covered.

3.3 Thermal model

The temperature of the silicon surface is exponentially influencing the subthreshold leakage. As presented in Section 1.3.3 a transistor with $250mV$ threshold voltage will double its subthreshold leakage each $38K$ thermal increase. With $350mV$ threshold voltage, subthreshold leakage is much lower, but depends even more on temperature, doubling each $26K$.

Now, the temperature of the silicon surface depends on the ambient temperature outside the system, the cooling efficiency, the thermal resistance of the package and, most importantly here, the power consumption inside the system. That means, that the higher the leakage power is, the higher the temperature becomes and vice versa. This effect is known as electro-thermal coupling.

3.3.1 Electro-thermal coupling

Example:

Let a system at ambient temperature consume a total power of $1W$ and the subthreshold leakage at $V_{th} = 250mV$ contribute with 50% to the power. Let the efficiency of the cooling (thermal resistance of the package plus thermal interface material plus cooling device) be $20mW/K$, which means, that with each Kelvin temperature difference, the cooling device can get rid of a thermal energy of $20mJ$ each second. Now the system will heat up to $50K$ over ambient temperature, reaching an equilibrium between converted and transfered thermal energy. With a $50K$ increase in temperature, the leakage is increased by a factor 2.5 and the total power is at $1.75W$. Now a $1.75W$ power will increase the system to $87K$, and at $87K$, the total power becomes $2.95W$, and so on.

In this extreme case, thanks to the electro-thermal coupling, the system enters a runaway situation where an increase of one Kelvin increases the leakage so much, that the temperature is increased by more than one Kelvin. The same example with a better cooling device of $50mW/K$ will initially result in a temperature of $20K$ over ambient, and then due to the electro-thermal coupling increase to $24.4K$ and then finally converge to approximately $26.1K$, which is reached after six iterations to an accuracy of $0.01K$ and after ten iterations to an accuracy of $10^{-5}K$. Even for thermally stable systems, the electro-thermal coupling leads to an increase of the final temperature and thus the leakage.

Also the system performance is thermally dependent, but with a much weaker slope of $10-30\%$ for a $100K$ temperature difference. In $90nm$ systems and above, a warmer system used to always be a slower system, but in $65nm$ and below, the thermal dependency becomes much more complex: As described in [125], the majority carrier mobility and the threshold voltage are both reduced with rising temperature. While the mobility decrease is reducing performance, the threshold decrease is increasing it. At $90nm$ and above, the mobility reduction clearly dominates and the worst performance is at maximum temperature. At $45nm$ and below, it is vice versa, and all gates become faster, the warmer they are. In between, at $65nm$ it depends on the gate (especially on the threshold voltage and the supply voltage), if the delay increases, decreases or even behaves non-monotonically with temperature.

3.3.2 Related work

[23] reported a 50 Kelvin temperature difference between the warmest and the coldest spot at the silicon surface of a large high performance device. Even with much smaller thermal gradients, the extreme thermal dependency of the subthreshold leakage makes an accurate spatial computation of the temperature mandatory.

Several publications have been made on on-chip heat determination. In [126], the physical equations for thermal energy, thermal flux, thermal resistance and thermal capacitance are solved in the Fourier-space for the frequencies of the thermal distribution. [21] ignores the spatial distribution of the heat, but finds the saturation temperature (increased by the electro-thermal coupling) by a self consistent analytical approach. [46] presents a full 3D finite element based approach alternately computing thermal gradients, thermal flux, and temperature.

The alternate direction implicit approach [127] uses the fact, that the differential equation describing the thermal distribution (see Equation 3.4) has a one dimensional closed (analytical) solution. The basic idea is to compute the equilibrium state for each of the three spatial dimensions one after the other, and then restart with the first dimension. That way, the numerical integration can be completely avoided.

3.3.3 Choice of a thermal model

The approaches described above have been evaluated by my student Patrick Knocke with the result, that none of the approaches converges fast enough to be used in a framework (PowerOpt), in which the thermal profile has to be computed for thousands of solutions under test.

Thus, I developed a solution exploiting available symmetries to combine speed and accuracy (superposition based approach, see below). In order to evaluate the accuracy of this approach, a finite elements method, directly solving the physical equations of thermodynamics was implemented, too.

Plain finite elements:

The entire active silicon surface and all layers of interconnect, package, thermal interface material, heat spreader, and heat sink are separated into cuboids (i.e. finite elements). For each finite element, the spatio-temporal differential equation of thermodynamics

$$\rho C_T \frac{\partial}{\partial t} T(x,y,z,t) = \vec{\nabla}\left(k(x,y,z)\,\vec{\nabla} T(x,y,z,t)\right) + P(x,y,z,t) \tag{3.4}$$

is solved. In this equation, $\rho C_T \left[\frac{J}{Km^3}\right]$ is the spatial thermal capacity and $k(x,y,z)\left[\frac{W}{Km}\right]$ is the thermal conductivity of the material of the finite element. $P\left[\frac{W}{m^3}\right]$ is the power density of the finite elements, and $\vec{\nabla} = (\partial/\partial x, \partial/\partial y, \partial/\partial z)$ the Nabla operator. Given an initial condition of the temperature distribution $T(x,y,z,t=0)$ Equation 3.4 can now be numerically solved for small discrete time steps.

As a steady state solution $P(x,y,z,t) = P(x,y,z)$ is sufficient, a distribution

$$0 = \vec{\nabla}\left(k(x,y,z)\,\vec{\nabla} T(x,y,z)\right) + P(x,y,z) \tag{3.5}$$

is needed. This solution is obtained by iteratively computing the temperature of each finite element under the (faulty) assumption, that all neighbour cells have already converged.

In order to improve simulation speed, the time-steps for simulation are not always equal, but are updated each step and is determined by the finite element with the worst convergency property. Nevertheless, the convergency of this method is very bad. It is just used for evaluation, as its results are reliable (in contrast to the approaches below, which introduces a certain amount of error).

Superposition based approach:

This alternative exploits not only the homogeneity in time (steady state solution)[6], but also a homogeneity in the z direction. The system just consists of layers (in x-y direction) of homogenous material, and has only a single active layer. The system ends in z direction with an infinite ambient body which is always constant in temperature. The temperature just needs to be known within the active layer (see Figure 3.3).

Even though there is no general closed form solution, Equation 3.4 can be solved in the symmetry described above and using the linearity of the equation by a superposition approach:

[6] A time dependent version will be developed at Section 3.3.4

At first, Equation 3.4 is solved in steady state ($\frac{\partial}{\partial t}T = 0$) for an infinite die area with a singular heat source (without loss of generality at $x = y = z = 0$, thus $P(x, y, z) = P_0 \cdot \delta(x) \cdot \delta(y) \cdot \delta(z)^7$).

$$0 = \vec{\nabla}\left(k(z)\,\vec{\nabla}T(x, y, z)\right) + P_0\delta(x)\delta(y)\delta(z) \tag{3.6}$$

For a given $k(z)$, Equation 3.6 can be numerically solved for $P_0 = 1W^8$ and an ambient temperature $T_A = 0$. Let $T^*(x, y)$ be the solution of Equation 3.6; the distribution of a real system with a continuous heat source at the $z = 0$ plane (active layer) can then be super-composed using the singular solution $T^*(x, y)$. The active layer is therefore represented by a grid of singular sampling points, representing the total power distribution of the active layer

$$P_{active} = P(x, y, z = 0) \approx \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} P_{i,j} \cdot \delta(x - i \cdot \Delta_x) \cdot \delta(y - j \cdot \Delta_y) \tag{3.7}$$

where $\Delta_x = X_{tot}/M$ and $\Delta_y = Y_{tot}/N$ are the spatial distance of the sampling points in x and y direction; X_{tot} and Y_{tot} are the size of the active layer in x and y direction; and

$$P_{i,j} = \int_{i\cdot\Delta_x}^{(i+1)\cdot\Delta_x} \partial x \int_{j\cdot\Delta_y}^{(j+1)\cdot\Delta_y} \partial y P(x, y, z = 0) \tag{3.8}$$

is the overall power from the respective active layer tile. Due to the linearity of Equation 3.4, the sum of the singular solutions, shifted to the according position ($x = i \cdot \Delta_x$ and $y = j \cdot \Delta_y$) and scaled to $P_{i,j}$ exactly solves the differential equation under the limitations mentioned above. As thermodynamics just depend on temperature differences, the ambient temperature can be added to the final result.

Reflection at the x and y edges

In order to use the super-composed solution, a correction has to be made: The methodology as described above solves the thermal distribution for an infinite die with a finite active area. A minor amount of energy is lost into the virtual surroundings of the die. In order to guarantee energy conservation inside the system, an intuitive solution is found: The thermal energy outside the die area is folded back (reflected) onto the active area. This solution is motivated by optical and electrodynamical physic and guarantees not only energy conservation, but also a gradient free connection to the border [9].

3.3.4 Final adaptations to the model

The superposition approach can compute thermal profiles of systems under certain symmetries nearly 1000 times faster than a finite elements approach. With the finite elements, the effect of electrothermal coupling (see Section 3.3.1) can be easily regarded by simply updating the power density

[7] The delta function $\delta(x) = (2\pi)^{-1}\int \partial k e^{ikx}$ has two important properties, used in this work: $\delta(x) = 0$ if $x \neq 0$ and $\int \partial x \delta(x) = 1$

[8] It is a property of the δ function, that P_0 changed from W/m^3 to W.

[9] Under the assumption, that no energy passes the border, the thermal profile must be flat ($\partial T/\partial x = 0$) in the proximity of the border.

after each iteration step. For the superposition approach, a static power density is mandatory. Thus, electro-thermal coupling has to be regarded after thermal mapping by an outer iteration (leakage is computed at ambient temperature, then temperature is computed with this leakage power, then leakage power is updated with respect to the recent temperature, and so on).

As presented in Section 3.3.1, for typical systems such an iteration needs three to six steps, which leaves a speedup of more than a factor 200[10] for the superposition approach regarding coupling. The final iterative solution, computing electro-thermal coupling is combined with the V_{DD} variation model, and will thus be presented at the end of Section 3.4.

The methodology as described above is sufficient for systems in a steady condition (as a constantly working ASIC component). As soon as this component can run in more than one mode (such as active and idle), it is no longer valid to assume, that the power density can be averaged over time. Instead, in order to describe a warming up phase after idle or a cooling down phase after active, a *thermal memory layer* TML can be introduced to the model between the system and ambience. If such a layer is described by a homogenous material with a finite thermal capacitance and a finite thermal resistance towards ambient but with a zero thermal resistance in x and y direction, the thermal behaviour (inclusively cool down and warm up) of a system in different operation modes can be described by computing the thermal profile for each steady state and exploit the fact, that the superposition approach computes the temperature relatively to the ambient temperature.

In a successive simulation over time, all energy from the system accumulates in the TML and slowly dissipates to the ambience. The system below is now computed relatively to the TML. Without expensive recomputation of the thermal profile, a cool down or warm up can be simulated. Figure 3.3 presents all limitations and adaptations presented in this section.

3.4 V_{DD} variation model

There are two effects, influencing the nominal voltage reaching each single gate with a relevance for leakage power estimation. The more important one is called IR drop: All gates within one power rail draw a certain amount of current from supply to ground. As the power and supply rails have a finite resistance, a small portion of the supply-ground voltage drops over these resistances. As a result, the effective voltage, each gate *sees* is reduced, the further away, the gate is from the power stripes[11]. The on-chip IR drop typically reduces the supply voltage for the worst gate by -5% of the nominal V_{DD} [128].

The second effect is called voltage drop: An abrupt change of the system's power (thus current) demand will lead to a damped oscillation of the supply voltage due to the resistance and capacitance of the entire supply interconnect. Voltage drops can lead to a significant reduction of the supply voltage of $\pm(10-15)\%$, but only for a very short time in the order of nanoseconds [23]. The IR-drops

[10] only the superposition has to be iterated, and the singular solution can be reused

[11] Power stripes are very thick metal interconnects, perpendicular to the supply and ground rails. The power stripes connect the rails with the pads of the external power supply.

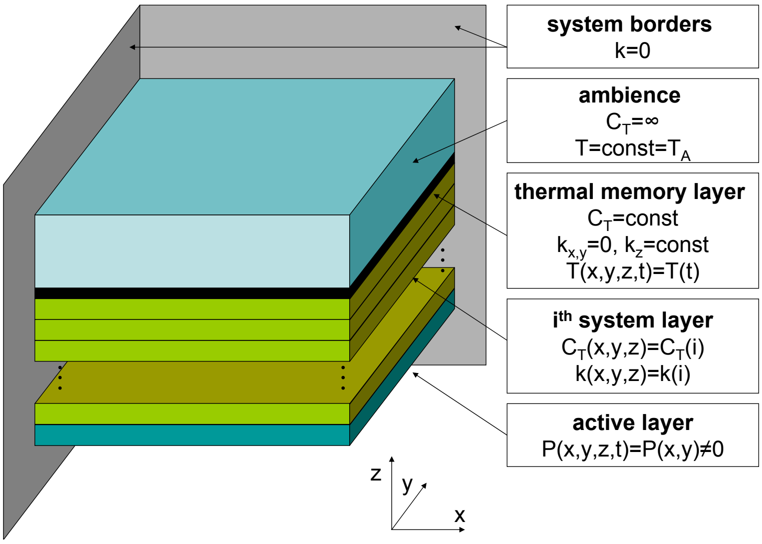


Figure 3.3: Visualisation of the system symmetries, exploited to enable fast but accurate thermal simulation. The ambient layer is always at a user constraint constant temperature. The thermal memory layer TML has a temperature which is depending on the total energy history of the system. The TML introduces a rough time dependency. The active layer is the only layer with a non-zero power density. The system borders in x and y direction do not conduct thermal energy at all.

have a significant influence on the overall leakage power, while the voltage drops can be ignored [4], as they occur over a short time only and have positive and negative deviations that will almost even out.

3.4.1 IR drop modelling

The IR drop prediction starts with an RT level floorplan of the system. For each component, the dynamic and static power dissipation is computed. Assuming a homogenous distribution of the power over each single RT component[12], a power density map can be generated. The power density map is then rasterised to a user constrained granularity. The default granularity here is the height of a standard cell row.

Now, the user can constrain the number and position of power stripes, the default is two stripe at the leftmost and rightmost position of the system[13]. It is assumed, that there is no IR drop in the power stripes, the pads, and the supply interconnect outside the chip. Thus, nominal V_{DD} refers to

[12] The PowerOpt tool works at system level and synthesises the system down to RT level. There is no deeper (gate level) look into the system available. Thus, a homogenous power distribution per component is the most accurate available power density map.

[13] In PowerOpt, as in many other EDA tools, the system is always presented in a way, that the standard cells form lines from left to right, and that the power stripes are perpendicular from top to bottom.

V_{DD} in the power stripes, not at the DC/DC converter, or the power pins.

Each segment of the power density map (by default a quadratic area of standard cell height), drains a current $I_{x,y} = dP_{x,y} \cdot A/V_{x,y}$, where $dP_{x,y}$ the power density of the segment (x, y), A the area of each segment and $V_{x,y}$ the voltage level at this segment ($V_{x,y}$ equals the nominal voltage at start of the computation).

One standard cell row (*or several consecutive rows*) is modelled by two resistor chains (see Figure 3.4), representing the supply and ground rail (*or rails*). All resistors have the same resistance, and between each resistor pair, there is a constant current source representing the standard cells according to the power map. At both ends of the chain, a voltage source with the nominal supply voltage represents the power stripe connection (Figure 3.4). The resistor chains can remain open ended, if the last power stripe is not at the end of the standard cell row. In order to compute this network with two voltage and N current sources, at first the current direction at each node has to be identified: From both voltage sources, the current will flow into the system, and there will be exactly one current source j, which gets current from both sides (refer Figure 3.4). In order to find this current source j, the potential at each node is computed from the left and right side, for each possible j.

Example:

If only source 1 is supplied from the left, and all other sources from the right side, the current I_1 will result in a voltage drop over the first resistance of $\phi^L(j = 1) = V_{nom} - r \cdot I_1$. If sources 1 and 2 are supplied from the left, then $I_1 + I_2$ will result in a voltage drop over the first resistance and I_2 will also result in a drop over the second resistance, thus $\phi^L(j = 2) = V_{nom} - r \cdot (I_1 + I_2) - r \cdot I_2$. If now for instance 35% of the current from source 3 is also supplied from the left side, then ϕ^L becomes a continuous function in j: $\phi^L(j = 2.35) = V_{nom} - r \cdot (I_1 + I_2 + 0.35 \cdot I_3) - r \cdot (I_2 + 0.35 \cdot I_3) - r \cdot 0.35 \cdot I_3$. In general with $j = n + x$, $n \in \mathcal{N}$ and $0 \leq x < 1$ for the potential from left and right:

$$\phi^L(n + x) = V_{nom} - r \cdot \sum_{i=1}^{n} i \cdot I_i - r \cdot x \cdot (n + 1) \cdot I_{n+1} \tag{3.9}$$

$$\phi^R(n + x) = V_{nom} - r \cdot \sum_{i=1}^{N-n-1} i \cdot I_{N-i+1} - r \cdot (1 - x) \cdot (N - n) \cdot I_{n+1} \tag{3.10}$$

For the wanted node which is supplied partly from the left and partly from the right, the voltage drop from left and right have to be exactly identical, thus $\phi^L(j) = \phi^R(j)$ gives the node n and the ratio x to how much the current I_n is supplied from the left. Now, the current source I_n can be split into two parts and the network can be separated into two networks with just one voltage source, for which the voltage drops are easy to compute[14]. If more than two power stripes have to be computed, or if the power stripes are not at the left-most and right-most position, the electrical circuit can be

[14] Remark: A by-product of the V_{DD} computation is a computation of the currents. Together with the thermal map, all necessary information is available to describe electro-migration effects in the supply interconnect. Electro-migration is one major contributor to degradation effects, which is my recent research focus.

directly separated at each voltage source. Between each pair of voltage sources, the algorithm as described above can be used.

3.4.2 Iterative computation of leakage, temperature, and supply voltage

In order to regard the effect, that a lower voltage will result in a lower leakage, and thus a lower power density, and thus a lower IR drop, which will increase the leakage again, ... , the IR drop computation has to be integrated into the iterative computation of the electro thermal coupling presented in Section 3.3. This idea was already proposed by [4], and can be used in this work exactly as proposed there:

At first, leakage and dynamic power are computed at nominal voltage and initial temperature. The resulting power density map is then used to compute the thermal map and the IR drop map of the system. For each RT component, the average temperature and power is computed. Dynamic power is (nearly) independent of the temperature, but not of the voltage. Thus both, the static and the dynamic power are reevaluated and so on.

Second order error correction:

Building the average temperature and voltage for each RT component will result in a second order error, as the power is not linearly depending on temperature and voltage, thus a +1 Kelvin deviation will result in a larger power increase than the decrease caused by a −1 Kelvin deviation. The error caused this way (Equation 3.11) can be avoided, by evaluating the power models for each sample of the power map, and then averaging all model predictions (see Equation 3.12):

$$P_{single} = \frac{1}{XY}\left(\sum_{x,y} V_{DD}(x,y)\right) \cdot I_{model}\left(\frac{1}{XY}\sum_{x,y} V_{DD}(x,y), \frac{1}{XY}\sum_{x,y} \vartheta(x,y)\right) \tag{3.11}$$

$$P_{sample} = \frac{1}{XY}\sum_{x,y} \left\{V_{DD}(x,y) \cdot I_{model}\left(V_{DD}(x,y), \vartheta(x,y)\right)\right\} \tag{3.12}$$

3.5 Body and supply voltage optimisation

The leakage models support a component wise variation of the supply and body voltage. In order to guide the user exploring the ABB and DVS alternatives, several design alternatives are developed here.

ABB as well as DVS offer a trade off between power and performance which can be exploited at design time to reduce power [73], or after manufacturing to increase yield [75]. ABB and DVS can be used at runtime to increase thermal stability, yield and reliability and to reduce power as presented in [72, 129]. In order to support each of these ideas, the optimisation time within the system estimation loops has to be carefully chosen. As presented in Figure 3.5, the inner-most loop of the estimation steps through each cycle, performing cycle wise data-accurate power estimation. If *run time* adaption of the body bias is enabled, the ABB and DVS optimisation engine can update

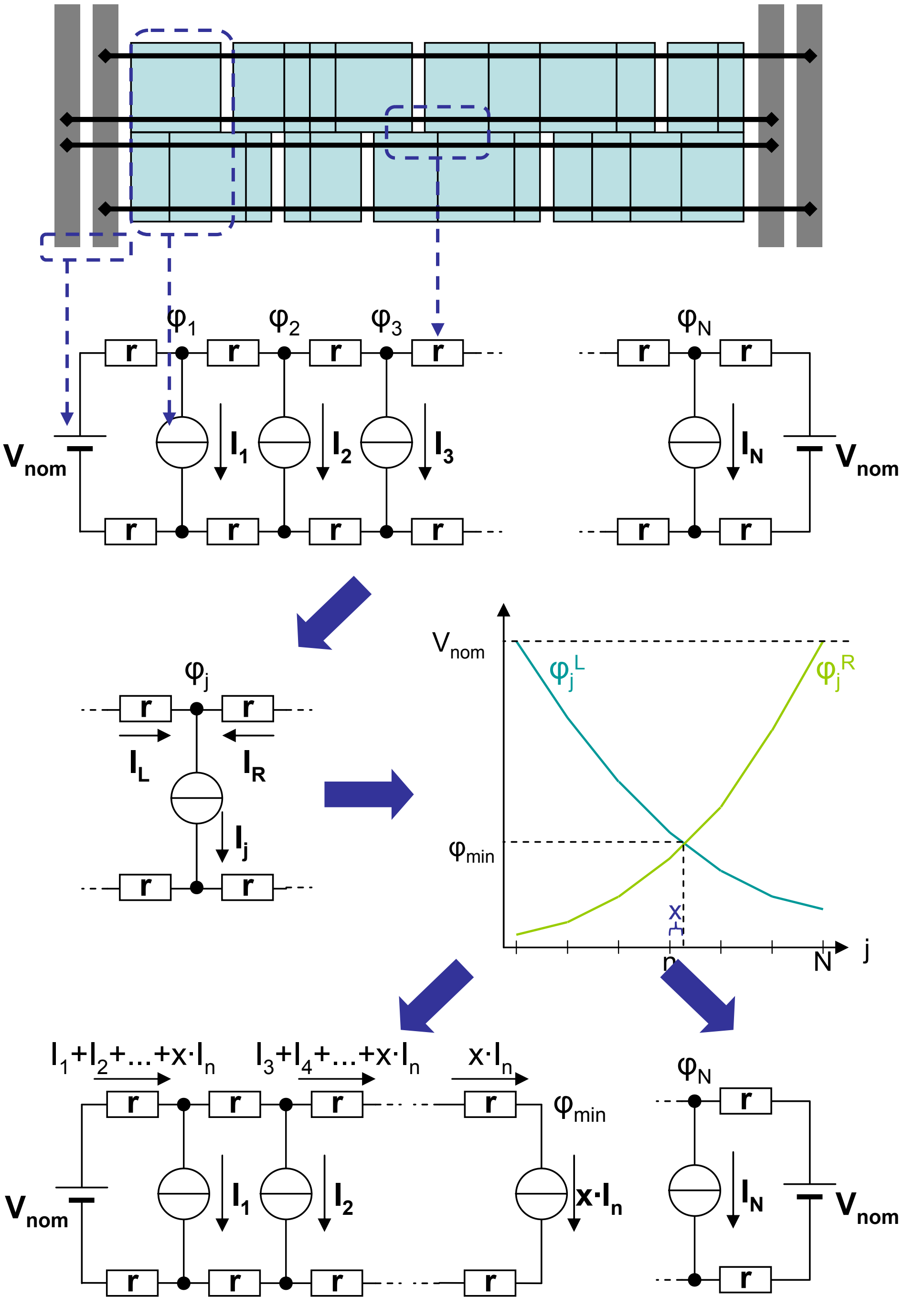


Figure 3.4: IR drop computation: One or more cell rows are described by an equivalent circuit with constant current sources. The node j, which is supplied from the left and right side can be found by computing the potential level ϕ_j for each j from the left and the right side. Now, the equivalent circuit can be separated into two simple circuits, for which the potentials are easy to compute.

both voltages at each cycle. This inner-most loop is executed for each inter-die instance, generated by the variation engine (see Sec. 3.2). If *test time* biasing is enabled, the ABB and DVS optimisation takes place for each variation instance. If only *design time* ABB and DVS optimisation is enabled, the voltages are optimised after high level synthesis inside the outer-most iteration loop.

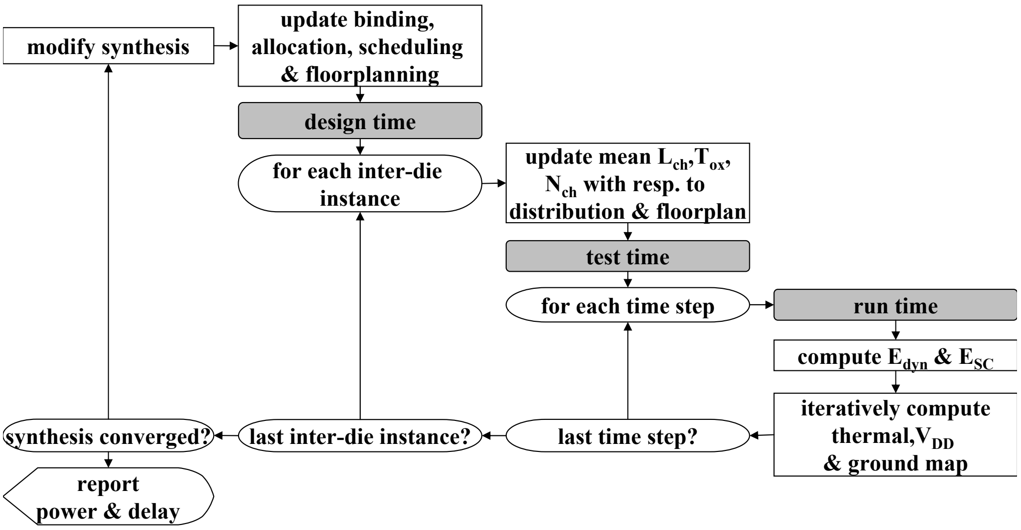

modify synthesis
update binding, allocation, scheduling & floorplanning
design time
for each inter-die instance
update mean L_{ch},T_{ox}, N_{ch} with resp. to distribution & floorplan
test time
for each time step
run time
compute E_{dyn} & E_{SC}
iteratively compute thermal,V_{DD} & ground map
last time step?
last inter-die instance?
synthesis converged?
report power & delay

Figure 3.5: The PowerOpt framework can modify each step of the behavioural synthesis having available accurate power, performance and yield prediction. For one synthesis result, several instances with different variation are generated. For each instance, the power for each cycle is computed. The three *levels* of body-bias and supply voltage control (design-time, test-time, run-time) are supported by moving the ABB/DVS-optimisation engine to the adequate iteration loop.

3.5.1 Optimisation schemes

The optimisation framework also supports several multiple V_{BB} and V_{DD} ideas [81], which will be presented in the following (ref. Figure 3.6):

Optimum

Optimum refers to the energy-savings being theoretically achievable by body biasing without voltage scaling. All components share the same supply voltage but have individual $V_{\mathrm{BB}N}$ and $V_{\mathrm{BB}P}$ values.

The optimal body biasing is obtained by varying each gate's body bias and the global supply voltage to minimise the power while keeping the performance nominal. For n RT components, this is a $2n+1$ dimensional optimisation problem $\overrightarrow{v}$ with one constraint delay $(\overrightarrow{v}) = \mathrm{delay}_{\mathrm{nom}}$. As the resulting $2n$ dimensional solution plane $P\left(\overrightarrow{v}|_{\mathrm{delay=nom}}\right)$ has only one minimum, a simple gradient approach can find the best solution.

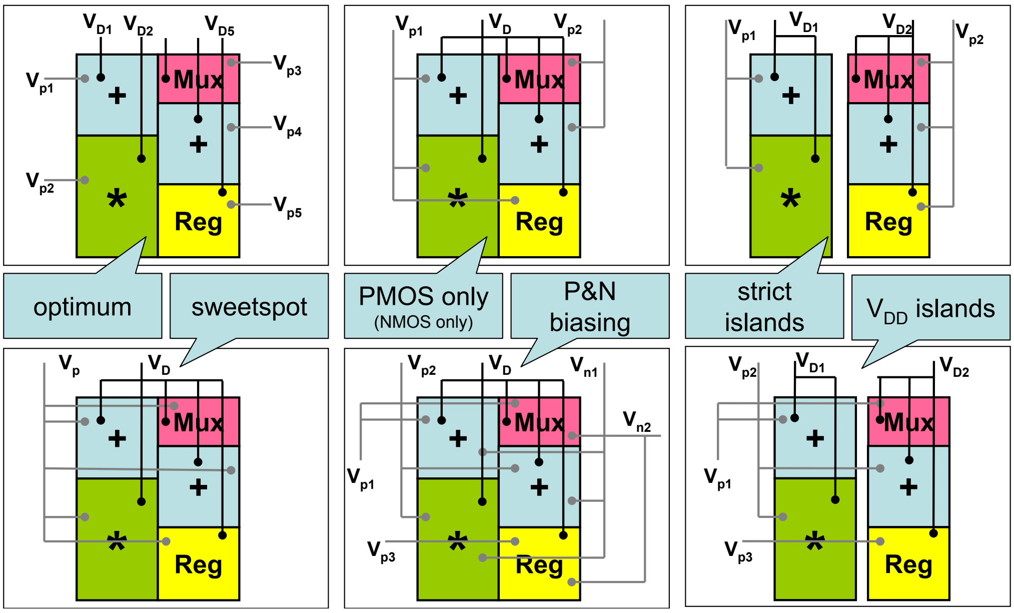


Figure 3.6: ABB and DVS optimisation schemes implemented. **Optimum:** Each RT component has its optimal supply and body voltage. **Sweetspot:** V_{DD} and V_{BB} are tuned once for all components. **PMOS only:** All components share the same supply and NMOS bias, but PMOS bias is optimised for groups of components. **P&N biasing:** Single V_{DD} but groups for NMOS and PMOS V_{BB}. **Strict islands:** System is locally divided into islands. Sweetspot for each island. V_{DD} **islands:** Local islands for V_{DD} and logical groups for V_{BB}.

The prove, that $P\left(\overrightarrow{v}|_{\mathrm{delay=nom}}\right)$ has only one global and no other local minima follows this idea: If one body voltage is changed and the supply voltage is adapted to let the overall design still meet its timing, there is one power optimal solution and the power rises monotonically on both sides of this minimum. Thus the gradient $\partial v_i / \partial P\left(\overrightarrow{v}\right) = 0$ exactly ones for each dimension i. By definition, a high dimensional minimum is a vector, where the gradients in each dimension are 0. Thus the solution space has exactly one minimum, the global one.

For the optimisation evaluation, each component can also have an individual V_{DD}, called **Opt-Full**.

Sweetspot

Following the idea of [73], body voltage and supply voltage are tuned to minimise total power while meeting a worst case delay[15]. This corresponds to a system, where the V_{DD} and both V_{BB} are chosen for the total system. Sweetspot is equivalent to P&N-Body-Biasing-1, as well as Strict-Islands-1 and V_{DD}-Islands-1 (see below).

[15] in order to have a performance constrained yield-loss of less than 5% for the **fixed method**

PMOS-only-x

An alternative ABB implementation having the lowest possible implementation overhead. PMOS-only is applicable in conventional dual-well systems, where the well is p-doped, NMOS transistors are build directly into this well, and thus all NMOS transistors share the same well. In contrast, each PMOS transistor has its own n-doped well isolated by trench isolation and can be charged individually.

A spatial grouping algorithm is applied, in which the user can specify how many different PMOS body voltages he can afford to supply (defined by x in the method name). SPICE simulations show that the interconnect supplying the body bias voltages does not have to carry large currents, thus additional interconnect cost routing each body bias across the system is acceptable. Groups of RT components are identified having the same body bias voltage, but do not necessarily have to be placed locally together (called a group in contrast to an island).

The NMOS bodies in this approach share one common $V_{\mathrm{BB}N}$. As the optimal $V_{\mathrm{BB}N}$ is typically very close to ground potential, $V_{\mathrm{BB}N} = 0$ is selected in order to save a voltage generator.

NMOS-only-x is the equivalent implementation but for n-doped dual well technologies.

P&N-Body-Biasing-x

This techniques is applicable, when having available a triple-well technology where both, NMOS and PMOS body can be individually controlled. The value x again holds the user-specified number of groups (not islands, see above) in which the design is separated. The NMOS and PMOS voltage for each group of components can be individually chosen. All components share one supply voltage. For this approach and PMOS-only-x, the separation into groups, the selection of the V_{BB} voltages for each group, and the selection of the global V_{DD} are done as follows:

Based on the Optimum solution, a solution is generated with respect to a user-constrained number of V_{BB} voltages. An increase of the body voltage will speed up the component but increase the power. For each component, the power cost is computed for increasing the V_{BB} to the next higher occurrence in the design (called a shift). Among all possible shifts, the cheapest in terms of energy is chosen. The number of individual body voltages decreases by one. Shifting is iterated until the number of different body voltages equals the user-constraint. This heuristic does not find the optimal but at least a good solution in a reasonable time. A low computational complexity is essential, as the heuristic has to be repeated several times (rf. Figure 3.5).

Strict-Islands-x

As soon, as there is more than one supply voltage, it is not enough to classify RT components to logical groups. Instead, the high additional interconnect cost for routing a second V_{DD} requires locally grouping the components in the floorplan to rectangular areas, called islands. Moving the components of a logical group on the die will have drastic influence on the data interconnect delay and energy. Thus, the supply voltage clustering of RT components has to be done best when generating the floorplan.

Inside PowerOpt, coarse RT floorplanning is done the same time as binding and allocation is. In order to gain the highest energy reduction, supply- and body-voltage island-building methodology is added to the floorplanning heuristic which is used by the binding and allocation heuristic [111]. The outcome is a design, in which the binding supports the optimal floorplan trade-off between short interconnect and a useful separation of the component into voltage domains, together with a floorplan, realizing this trade off.

The existing floorplanning heuristic works by representing the system as a slicing floorplan tree, a number of moves connecting different solutions, an undo-methodology for each move, and a simulated annealing methodology which randomly performs a move and probably undoes it if a cost function is not improved. The cost function itself consists of the interconnect energy, the interconnect delay on the critical path, and the total chip area, thus on all design metrics which change when the floorplan is changed. Implementation details can be found in [111, 130].

As the slicing floorplan always creates rectangular regions, the floorplanning heuristic has to be updated only in two ways: At first, the x highest nodes in the slicing tree become special voltage island nodes. A voltage island node is not movable and all children of this node inherit the same supply and body voltage. Secondly, new terms are added to the cost function which used to be invariant to floorplan changes, but may change with supply and body voltage (e.g. the component leakage).

The position of an RT component within the floorplanning tree determines its position and thus the length of its interconnect. Now it also determines the supply and body voltage of each component. The slicing tree is randomly changed, and after each change, each voltage island is optimised to its individual sweetspot (see above). After position, interconnect length, V_{DD}, and V_{BB} is determined, the component and interconnect power and the floorplan area are computed and added building the cost function[16].

V_{DD}-Islands-x

In order to limit the interconnect overhead, supply voltage domains have to be locally clustered. But for body-bias voltages local clustering is not needed. Thus, in this final approach, clustering of x voltage islands is combined with x ABB-groups which are built independently to the islands.

The algorithm for V_{DD}-Islands-x is a modification of Strict-Islands-x. The position in the floorplan here determines the V_{DD} island. The voltage level of each island is set by two additional floorplan moves increasing and decreasing V_{DD} of all components on an island. The body voltages are heuristically fixed before evaluation of the cost function as follows: Knowing the supply voltage of each component, the optimum V_{BB} assignment is found as described in Optimum. Then, the number of V_{BB} groups is reduced to x by the shifting algorithm described at P&N-Body-Biasing-x.

[16] The delay does no longer have to enter the cost function, as the sweetspot methodology always cares for the performance constraint.

In contrast to the Optimum, applying the shifting algorithm once, I had to limit the computational effort for the heuristic search of ABB domains, as this heuristic has to be repeated for each move in the search algorithm finding best supply clustering. This approach should have the highest savings as it is the superset of all other methodologies proposed. But due to the high complexity of the problem, the solutions obtained here are not optimal and are always outperformed by the (theoretically) less effective but easy solvable Strict-Islands-x algorithm. For the sake of completeness, the results are nevertheless given in the evaluation section.

3.5.2 Application examples

% vs. Reference	FDCT	JPEG1	JPEG2	JPEG3	Wvlet1	Wvlet2	Wvlet3	Wrap2	Wrap3	Avg
Optim-Full	36.6	23.3	27.2	24.7	53.0	54.3	49.0	20.1	20.6	34.3
Sweetspot	24.8	8.7	8.7	15.8	37.6	37.8	38.1	4.6	4.9	20.1
PMOS-only-1	23.1	8.7	8.6	15.3	34.2	32.9	33.0	1.0	1.5	17.6
PMOS-only-2	23.1	8.7	8.6	15.3	34.2	32.9	33.0	0.9	1.3	17.5
PMOS-only-3	23.1	8.7	10.5	15.3	34.2	32.9	33.0	1.0	1.4	17.8
NMOS-only-1	0.0	0.5	0.2	0.1	0.0	0.0	0.0	3.3	3.3	0.8
NMOS-only-2	3.6	1.3	3.5	1.8	2.7	1.8	1.6	4.2	4.1	2.7
NMOS-only-3	3.6	2.9	7.5	1.8	2.7	1.8	1.6	4.3	4.3	3.4
P&N-BodyBias-2	26.2	9.1	10.0	16.0	36.9	37.4	37.5	4.6	4.7	20.3
P&N-BodyBias-3	26.2	10.1	16.3	16.0	36.9	37.4	37.5	5.0	5.2	21.2
Strict-Islands-2	24.8	8.7	17.7	17.5	37.9	38.0	38.3	11.0	11.6	22.8
Strict-Islands-3	25.1	8.7	19.0	22.1	49.2	38.3	38.3	10.8	11.4	24.8
V_{DD}-Islands-2	20.5	-17.7	11.2	16.0	36.3	34.6	36.8	-2.7	-2.1	14.8
V_{DD}-Islands-3	25.5	-1.5	11.7	0.5	23.1	37.5	37.3	-2.3	-1.7	14.5

Table 3.1: Percentaged improvement in terms of total energy reduction of the proposed methodologies against the reference implementation. Depending on the hardware overhead and the design, total energy consumption may be reduced by over 50%. The theoretical maximum is at 34% on average.

Nine benchmarks have been applied: FDCT is a 32bit cosine transformation. JPEG1 and JPEG2 are the row and column routines of a JPEG codec, JPEG3 is the main task. Wvlet1-3 are three different implementations of a wavelet algorithm and Wrap2 and Wrap3 are memory wrapping subroutines used in Wvlet2 and Wvlet3. The runtime of the algorithms was limited to 15-30 minutes (depending on the size of the algorithm) which is enough to compare 800.000 different binding, floorplan and island partitioning solutions. If the body bias is fixed at design time (**fixed method**), the power for all instances is averaged. Individual values are obtained for various embedded system processes and presented in Table 3.1.

PMOS-only-1 is a very cheap and powerful trade-off. But having available the body biasing hardware further interesting implementations can be found. For systems having a high saving potential (e.g. the wavelet tasks), the Strict-Island method with three ABB and V_{DD}-islands is very close to the optimal solution. As discussed above, the V_{DD}-islands-x method performs worse than expected due to its high complexity.

For one prototypical example, the average power, if the body voltages are fixed after manufacturing (**adaptive method**) is computed as follows: Using the variation engine, the average power and performance distributions are computed. For the **fixed method**, the bias voltages are fixed under the constraint, that a user-specified percentage of 95% has to meet its timing. For all designs in the distribution, the same body voltages are used for power estimation.

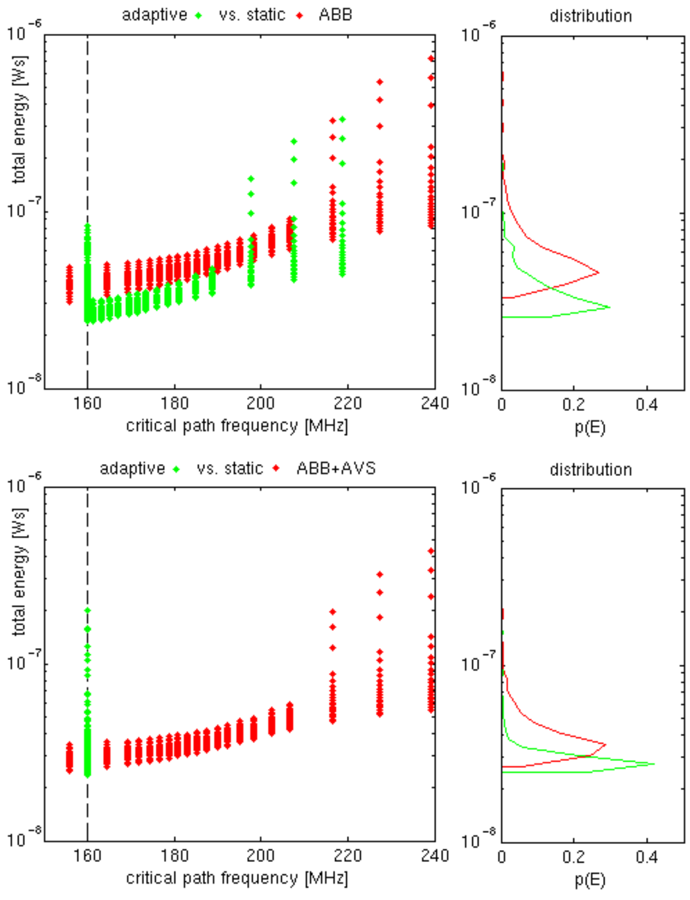


Figure 3.7: Power-performance distribution due to inter-die variation. The framework prediction shows the effect of body-biasing if voltage levels are optimised at design time (red) and fixed for all systems and if voltage levels can be adapted after manufacturing (green). The top part is computed for a user given supply voltage of $1V$. The bottom part compares fixed-sweetspot methodology with adaptive-sweetspot, where supply and body voltage are adapted after manufacturing.

For the **adaptive method**, the body voltages are adapted for each variation instance to let each system meet its own timing constraint. Thus, especially the fast but power hungry systems will be strongly back-biased and thus will increase the benefit of body biasing. Figure 3.7 shows the distribution of power and performance due to inter-die variation of the channel length and oxide thickness variation. In the top part, the supply voltage was fixed, and the body bias was optimised once for all systems or individually for each system instance. Adapting body-voltages alone can fix all timing problems of the slow process corner and reduce the leakage spread nearly by factor two. Additional power savings in the fast process corner cannot be exploited because increasing junction leakage is limiting reverse body biasing [72].

When V_{DD} can also be adapted, all slack due to process variation is exploited. In the bottom part of Figure 3.7, the static and adaptive sweetspot methodology are compared. All systems of the **adaptive method** finally exactly meet their timing constraint. The slowest systems now operate with V_{DD} around nominal $1V$ and strong forward body bias. The fastest systems have V_{DD} at their minimum of $0.7V$ and are at high reverse bias.

The PowerOpt user can now compare different realisations explicitly for the system under test and trade-off its benefits (power yield, variation) against its costs (technology, additional hardware).

3.6 Conclusion

Having available the leakage model described in Section 2.3 till Section 2.5 with the model splitting for power gating and minimal leakage vector (Section 2.7), the delay model (Section 2.6), the variation (Section 3.2), temperature (Section 3.3), and voltage model (Section 3.4), and the ABB and DVS optimisation (Section 3.5), all these components can be combined to an iterative yield prediction flow for entire systems.

For one design solution (synthesised and floorplanned), the estimation flow (see Figure 3.8) begins with selecting one specific variation sample from the variation engine, and is repeated for each variation sample:

For each sample of the variation engine, for each RT component of the system, the exact value of parameter variation[17] is determined. If adaptive supply and/or body voltage scaling is enabled, the voltages for each component are optimised (with the restrictions described in Section 3.5). Now for each component, the leakage and critical path delay can be predicted ('no PG' in Figure 3.8). If power gating and/or minimum leakage vector are enabled, the leakage for being in MLV, for being in power gated mode ('PG on'), for having power gating available, but not in use ('PG off'), and the overhead of a power gating transition are computed instead. The clock cycles, where power gating or MLV are applied can be optimised (see Section 1.4.3) and the total power and worst delay can be computed accordingly.

After this flow is performed for each variation sample, the expected distribution of the systems power and performance can be determined. By comparing it to user constraint power and performance

[17] may depend on the component's position on the die due to parameter gradients

bounds, the parametrical yield can be predicted. Such a resulting distribution was already presented in Figure 3.7.

Now, all metrics as functional yield, average delay, average leakage power, dynamic power, and area can enter a cost function. A heuristic search engine can now modify the synthesis and/or the coarse floorplan, and obtain a new cost function for this new solution. By selectively accepting or rejecting these modifications[18], the cost function, and thus its design metrics can be improved.

The user can decide himself, which cost function has to be minimised, but the energy and cost constrained ones are obviously very promising:

$$f_{energy} = E_{dyn} + T_{total} \cdot P_{leak} + \alpha_A \cdot A$$

$$f_{cost} = A/(Y_{param.} \cdot Y_{func.}) = A/(Y_{param.} \cdot Y_{norm}^{A/\mu m})$$

where E_{dyn} is the dynamic energy consumed for the entire runtime T_{total}, P_{leak} is the average leakage power, A the die area and α_A the tradeoff factor, which describes, how much area can be sacrificed per energy optimisation. $Y_{param.}$ is the parametric yield and $Y_{func.} = Y_{norm}{}^{A/A_0}$ the area dependent functional yield under the assumption that a one micrometer design has a yield of Y_{norm}.

Best to my knowledge, the only related work, doing similar parametric yield predictions due to inter-die variations is [17], which is much simpler, just regarding subthreshold under Gaussian variation of the threshold voltage and channel length as well as gate leakage under variation of the oxide thickness. Parameter gradients, temperature and voltage distribution and design for leakage and variability techniques cannot be regarded.

[18] PowerOpt has a very sophisticated heuristic for selecting the best modification and for the decision whether to accept or reject a modification.

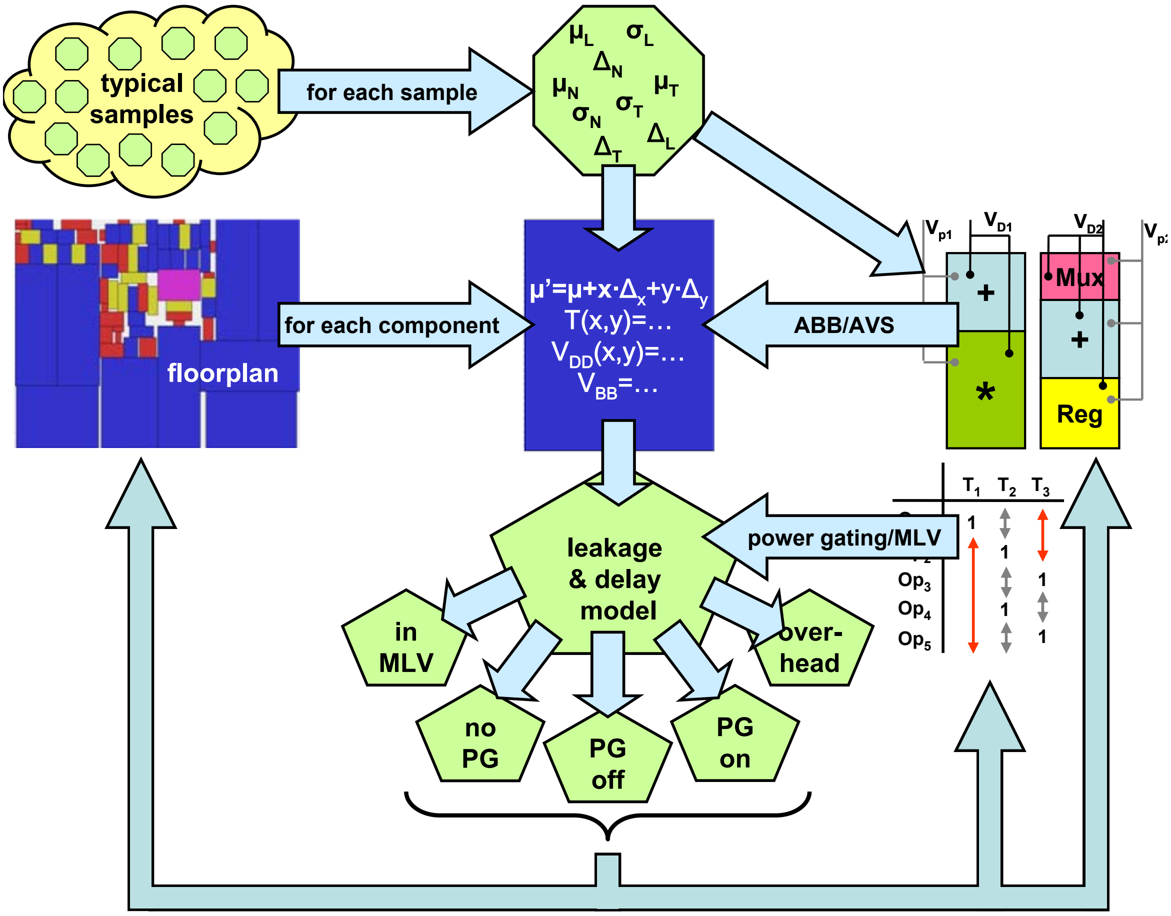


Figure 3.8: Visualisation of the parameter determination and model evaluation framework: For each sample, representing a typical variation class, the parameter means μ, variation σ, and gradients Δ are determined. The ABB/AVS optimisation engine plans a V_{DD} and V_{BB} scheme. Based on a floorplan of the system, for each component, the gradients contribute to the mean, and a thermal and electrical model determines local temperature and voltage. Afterwards, a power management engine controls the power down times of each component. Depending on the power down state, one of four models is chosen: No power management implemented, component is in minimum leakage vector, power gating is implemented but the component is active, PG is implemented and used. Finally, the transition overhead for switching between power states is added. The power result is then fed back to all stages as the thermal mapping or the optimisation steps.

4 Simulation environment

The leakage models (Section 2.3 till 2.5), being main content of this thesis, as well as the delay models which are connected to the main model to describe leakage delay correlations (Section 2.6), both rely on transistor level simulation. Real silicon measurement is not used in this thesis. Obviously, producing $45nm$ test-chips for evaluation of this thesis is by far too expensive. Additionally, these technologies are subjected to process variation, resulting in huge measurement variation. A single silicon measurement can thus not be used for a meaningful evaluation. On the other hand, the industrial compact models are constantly compared with and tuned for a large series of silicon measurements by the industry themselves. The simulation result of a well maintained industrial compact model is thus far more accurate, than a single silicon measurement as it describes the general technology behaviour rather than an individual variation instance [131].

As all model characterisation as well as the final evaluation base on transistor level simulation, available industrial transistor simulators and compact models (single transistor models) are reviewed in Section 4.1. The academic SPICE simulator and its compact model BSIM are then presented in Section 4.2. Section 4.3 discusses the gate analysis flow, automatically generating SPICE netlists from the Liberty library (*.lib*) descriptions. Finally, Section 4.4 presents a methodology generating SPICE netlists for entire RT components.

4.1 Commercial transistor simulators

Besides Berkeley's SPICE, which is a free academic analogue simulator, the most prominent industrial transistor simulators are the commercial SPICE derivatives, and the free PSP model from Penn State and Phillips. Annex A presents a market analysis of commercial transistor simulators.

4.1.1 Phillips PSP

The Penn State Phillips (PSP) model [132] is a result of merging two surface potential based compact models, the *SP* model from Pennsylvania State University and the *MM11* model from Phillips. The PSP model is called surface potential based, because transistor simulation starts by solving a simplified Poisson equation of the surface potential at the channel/oxide interface [132]. Potential based compact models are regarded as the most accurate transistor description methodology available.

The PSP model distinguishes between a global and a local parameter set. Before each transistor evaluation, the global parameter set is converted into a local parameter set for this transistor instance.

The core model of PSP is then divided into the extrinsic and the intrinsic model. In the intrinsic model, the channel current and the charges of the transistor's terminals are computed. The extrinsic model then adds outer influences as gate tunnelling and substrate currents. Two additional modules, the *JUNCAP2* junction model and the non-quasi static *NQS* model increase the modelling accuracy (see Figure 4.1).

Especially due to the JUNCAP2 model, accurately describing gate induced drain leakage, the PSP model is meant to be the most accurate publicly available compact model. Since September 2006, the PSP model is the official[1] industrial standard compact model.

Phillips developed an analogue simulator, called *pstar*, which (best to my knowledge) is only used in-house, and is not commercially available.

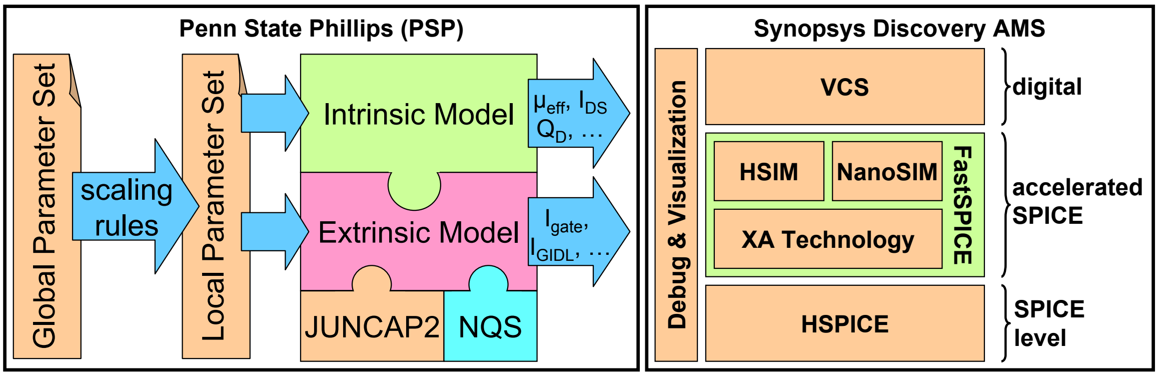


Figure 4.1: **Left:** Structure of the PSP model: For each transistor evaluation, local parameters are computed according to scaling rules. The compact modelling is separated into an intrinsic model, computing inner properties as channel current, or terminal charges, and an extrinsic model computing effects as gate leakage. The JUNCAP2 model is the most accurate model, describing gate induced drain leakage effects. **Right:** Synopsys Discovery AMS tools. HSPICE is a commercial SPICE derivative, NanoSIM simulates SPICE netlists with a slightly reduced accuracy but with a high performance. HSIM is a hierarchical SPICE simulator and VCS is the Synopsys logic level simulator.

4.1.2 Synopsys Discovery AMS

Discovery AMS is a SPICE based tool-suite for transistor level simulation. With Discovery AMS, Synopsys offers various simulators from the SPICE equivalent HSPICE to the less accurate, but very fast fastSPICE tools (refer Figure 4.1). HSPICE uses an extended SPICE syntax with additional useful features as parameter definition via parameters or temperature or geometry range analysis. As HSPICE follows the SPICE syntax rules much stricter than SPICE itself, SPICE scripts are typically neither up- nor downward compatible. Converting SPICE into HSPICE scripts can be done

[1] according to the Compact Model Council

by simply regarding upper-case and lower-case letters and terminal symbols[2]. The fastSPICE tool NanoSIM is a less accurate but much faster implementation which is able to deal with large netlists in a reasonable time. HSIM is a hierarchical SPICE derivative exploiting a regularity of the design (if any). Before simulation, repetitive pattern in the netlist are identified, and simulation is done once for all instances. HSIM is thus extremely efficient for memory simulation.

The compact model, used by all these simulators is BSIM level 49 (and higher) which is a BSIM3v3 clone with some additional HSPICE parameters.

4.2 SPICE & BSIM

When I started developing the leakage models in 2003, BSIM was the only available compact model, for which industrial characterisation data was available, as will be detailed in Section 4.2.3. For model generation, characterisation, and evaluation, a single RT component computation is the maximal demand. Thus, the obvious choice for a transistor netlist simulation framework (analogue simulator + compact model + modelcard) was the free SPICE (*Simulation Program with Integrated Circuit Emphasis*) and BSIM (*Berkeley Simulator*) and BPTM (*Berkeley Predictive Technology Model*).

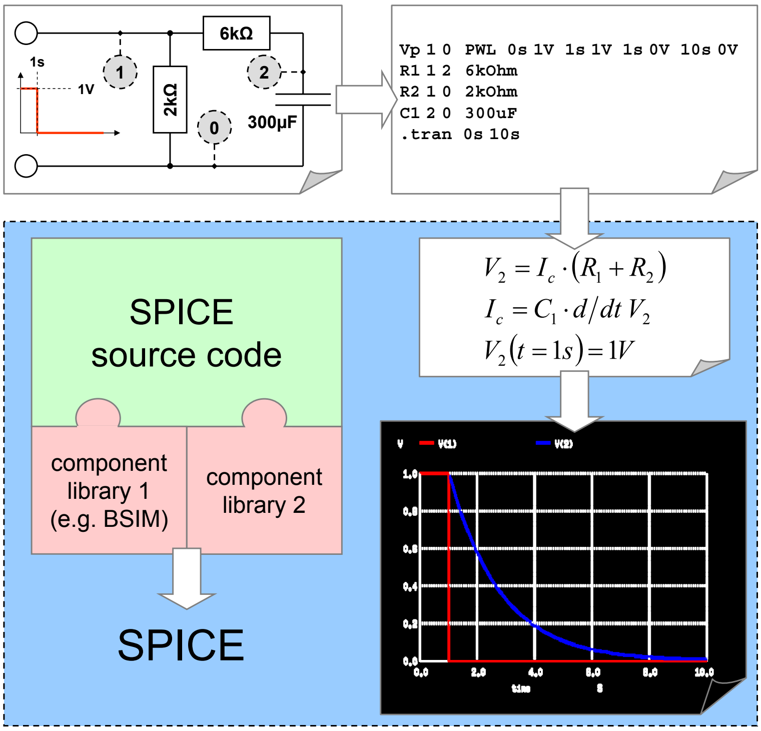
The simulation engine of SPICE is nothing else but a powerful numerical solver of differential equations. SPICE does not need these equations in an analytical form, but extracts them from an electrical circuit. In terms of differential equations, SPICE offers linear (resistors, conductors, ...), nonlinear (diodes, transistors, ...), and time dependent (voltage and current sources) components. The built-in nonlinear models are relatively simple, thus more complex components (as sub-100nm MOSFET devices) must be described by including respective component libraries to the C-source of the SPICE executable. The numerical solutions can then be processed using built-in analysis methodologies. A typical SPICE flow is presented in Figure 4.2.

4.2.1 Simulating in SPICE

The voltage over time $V_i\,(t)$, defined as the potential difference to the special node 0, of each circuit node is available as a variable in SPICE after simulation. As there is a one-to-one correspondence between voltages and nodes[3], all voltages can be accessed by using the **V(node id)** command. Each potential difference between two nodes can thus be easily computed by subtracting their respective voltages $V_ij\,(t) = V_j\,(t) - V_i\,(t)$. In contrast, the currents $I_j\,(t)$ of each branch are not generally stored, as there is no one-to-one correspondence to a certain circuit structure. Currents are only stored for voltage sources (by definition) and for certain components with more than two terminals (by implementation). Thus, the simplest way of simulating currents is by including a $0V$ DC voltage source into the branch, where a current has to be simulated.

[2] Example: In SPICE, the symbols = ( ) ... are all treated the same way. Thus 'Tox=1.6e-9' or 'Tox(1.6e-9)' or even 'Tox)1.6e-9(' are all evaluated correctly in SPICE, but not in HSPICE.

[3] each node has a unique voltage, and the voltage is identical for all terminals connected to a node



```
Vp 1 0  PWL 0s 1V 1s 1V 1s 0V 10s 0V
R1 1 2  6kOhm
R2 1 0  2kOhm
C1 2 0  300uF
.tran 0s 10s
```

$$V_2 = I_c \cdot (R_1 + R_2)$$
$$I_c = C_1 \cdot d/dt\, V_2$$
$$V_2(t = 1s) = 1V$$

Figure 4.2: SPICE flow example: SPICE translates circuits, described as transistor netlists into differential equations, and solves them numerically. Non-standard component (as recent MOSFET transistors) can be described by including component-libraries. BSIM is a SPICE library.

For later model building, also other electrical properties as charges, power dissipation, energy consumption and switching delay have to be simulated. In order to realize such simulations, I implemented a program, called SCI (SPICE circuit integrator), which can read in a SPICE netlist, and call SPICE in non-interactive mode. For the charge, power, and energy computation of one user specified $0V$ source, the three vectors $t(id)$, $V(id)$, and $I(id)$ are gathered. For each time t^* with $t(j) < t^* < t(j+1)$, the voltage $V(t^*)$ and the current $I(t^*)$ are linearly interpolated and the resulting metrics are numerically computed as

$$Q = \sum_j (t(j+1) - t(j-1))/2 \cdot I(j) \tag{4.1}$$

$$E = \sum_j (t(j+1) - t(j-1))/2 \cdot V(j) \cdot I(j) \tag{4.2}$$

$$P = E/(t(N-1) - t(0)). \tag{4.3}$$

In order to determine delays (typically switching delays of gates), the SCI user can specify a node, a voltage threshold V^d, and a transition direction (rising or falling) for two nodes (input and output). The SCI program then finds each id j with $V(j) < V^d < V(j+1)$ (and vice versa for falling transitions) and interpolates the intersection time

$$t^d = \frac{\left(V(j+1) - V^d\right) \cdot t(j) + \left(V^d - V(j)\right) \cdot t(j+1)}{V(j+1) - V(j)}. \tag{4.4}$$

As long as all detected events occur consecutively for the input and output node, SCI returns t_i^d, t_o^d, and $t_o^d - t_i^d$. In standard cell libraries, the delay of a gate is defined as the time between an input transition with a standard slope passing a certain threshold and the output transition, driving a standard load passing its threshold. Typical delay metrics are 50-50 delay from $V_{DD}/2$ to $V_{DD}/2$; or 10-90 delay from $0.9 \cdot V_{DD}$ to $0.9 \cdot V_{DD}$ for a rising output and from $0.1 \cdot V_{DD}$ to $0.1 \cdot V_{DD}$ for a falling output.

4.2.2 BSIM computation flow, version differences & capabilities

As mentioned, there are two *main-stream* compact modelling methodologies, surface potential based models (e.g. PSP) and threshold based models. BSIM, the Berkeley simulator is a threshold based model.

BSIM compact modelling works as follows: After determination of the transistor geometry parameters as oxide thickness, effective channel length, or effective channel width, BSIM compact modelling starts with the determination of the threshold voltage [13, 133]. Equations 1.5 to 1.9 in Chapter 1 show a simplified version of the BSIM threshold voltage: The threshold voltage of a uniformly doped long channel transistor considering the body effect is determined and then corrected, describing the influences of non-uniform doping, short channel effects, and narrow width effects [14]. Based on the threshold voltage, the channel charge and the drain-source current can be computed [134].

With version 4.00 (March 2000) the two major other leakage effects gate tunnelling and gate induced drain leakage, were introduced. Both effects are computed independently from the V_{th} and I_{ds} computation and are superposed to the model. Up to version 4.21 (October 2001), besides minor bug-fixes and the introduction of gate induced source leakage, analogue to GIDL, two new parameters (XL and XW) were introduced. These parameters can model mask and etch effects reducing the effective channel length and width. Version 4.40 (March 2004) introduced the first realistic gate tunnelling model for all occurrences of carrier tunnelling (gate-edge side-wall, STI-edge side-wall and bottom junction).

With Version 4.50 (September 2005) several variation parameters describing threshold voltage variation and well proximity effect (see Section 1.3.1) were introduced. This version already has the full functionality, used in this thesis. Nevertheless, the recent Version 4.61 (May 2007) is employed for all numerical results, as it brings some accuracy fixes and enables description of high-k oxides and metal gate devices.

4.2.3 Modelcards

Over the last years, the availability of BSIM characterisation data *modelcards* improved significantly. Nowadays, the Nano-CMOS generator [135] can either generate completely generic modelcards for a given technology node, or can generate customised modelcards for a known technology. The modelcard's transistors will then have parametrisable major parameters like threshold voltage or source-drain resistance. Modelcards, generated by Nano-CMOS are always compatible with the most recent BSIM version available. Nano-CMOS is an integral part of the gate analysis flow, described in the following Section 4.3, and will be detailed there.

Most SPICE simulation leading to this thesis was done before Nano-CMOS was available, using the pre-designed *Berkeley Predictive Technology Model* modelcards. For each new technology, Berkeley developed a modelcard for the currently actual BSIM version. As a result, the BPTM modelcard for the 130nm technology was characterised for BSIM3v3, the 45nm file needed BSIM4.4.0 to run and the 65nm technology was characterised twice[4].

For each technology node, a dedicated SPICE version with the right BSIM library included would have been necessary. In order to be able to do direct comparison between different technologies, I developed a conversion tool, called simConv, being able to convert each modelcard from versions

- BSIM3v3.x
- BSIM4.x, level 14 (real BSIM)
- BSIM4.x, level 49 and 53 (HSPICE)

to version 4.5.0[5]. With simConv, a SPICE executable with the actual BSIM library can interpret all BPTM cards for each technology node. SimConv can work in three modes, which are briefly outlined in Annex B. For each conversion, the output format is BSIM level 14, version B450.

4.3 Gate construction by property analysis

The simulation environment as described in this chapter needs a BSIM description of the transistors and a SPICE netlist for each standard cell of the technology. As such intimate technology information is not always available, a gate property analysis flow as described here can be used to generate both, transistor and gate description from a technology description in the *.lib* format and some basic technology data. A Liberty library (*.lib*) technology description (throughout this chapter called a library) is also needed for the characterisation of all other model parts (dynamic power, area, timing) of the Component Data Base (CDB) model, which is the base of the PowerOpt estimation. Thus the library description is usually available.

4.3.1 Transistor description

A full BSIM characterisation card consists of 752 single values. The Berkeley device group developed a characterisation flow [136] to determine these values. Re-engineering all these values from the few numerical values of the library is definitely prohibitive.

[4] once called 70nm and made for BSIM3v3.5 and once called 65nm and made for a BSIM4 version

[5] I had to update from the initial version converting to BSIM4.4.0, but the BSIM4.5.0 modelcards runs stable with each higher version, since no parameters were updated from 4.5.0 to 4.6.1.

Instead, the *Nano-CMOS* online generator (available at [135], described at [15]) is used to obtain a BSIM modelcard. For Bulk NMOS and PMOS devices from $32nm$ to $130nm$ node size, this tool can generate modelcards based on only five elemental technology properties, which have to be user specified. The threshold voltage, supply voltage and oxide thickness are usually available, the effective channel length can - if unknown - be computed from the drawn channel length minus twice a technology dependent length offset due to mask and etch (dL in Table 4.1). For the drain to source resistance, technology dependent default values can be used, if the real value is not known. These technology dependent default values are summarised in Table 4.1.

	NMOS			PMOS		
node	$T_{ox}[nm]$	$dL[nm]$	$R_{ds}[\Omega]$	$T_{ox}[nm]$	$dL[nm]$	$R_{ds}[\Omega]$
$130nm$	1.6	40	200	1.6	40	240
$90nm$	1.4	27	180	1.4	27	200
$65nm$	1.2	20	165	1.2	20	165
$45nm$	1.1	14	155	1.1	14	155
$32nm$	1.0	10	150	1.0	10	150

Table 4.1: Technology dependent default values for the transistor characterisation [135]. Assuming, that at least, supply voltage, threshold voltage, and node size are known, a modelcard can be generated.

4.3.2 Standard gate description

Having available both, the library and the BSIM transistor description, the analysis of the gates can be performed as a series of five steps. The main purpose of a library is to describe the technology's timing; leakage currents in recent library files are usually not reliable and only roughly characterised. Thus, all steps are based on timing values as rise- and fall-time, and input capacitance. The leakage values specified in the library are not regarded for the gate construction.

While the transistor description requires manual user interaction, the following steps are completely automated. After the user specified the filename of the library, this library is parsed and SPICE descriptions for all gates are automatically generated.

Step 1: Standard inverter generation

At first, a CMOS inverter, with $W_N nm$ NMOS and $W_P nm$ PMOS transistor width is implemented in SPICE having the standard fanout load and a standard fanin slope, as specified in the library documentation. W_N and W_P now are tuned to exactly meet the rise and fall times of a standard inverter (driving strength 1) from the library. Using the input capacitance C_{inv1} from the library, the ratios $\alpha_N = W_N/C_{inv1}$ and $\alpha_P = W_P/C_{inv1}$ are computed and stored.

Step 2: Enumeration of all equivalent gates

For each logic gate in the library, all CMOS circuits implementing it's functionality are identified. Not regarding driver cells and driving strengths, an n input cell has at least 2^{n-1} implementations, which are logically equivalent, but electrically different. All these cells are represented as non-commutative binary trees with $n-1$ nodes which either are NMOS parallel and PMOS serial (denoted as $+$) or NMOS serial and PMOS parallel (denoted as $*$)[6]. By definition, inside the serial branch, the left operand is closer to the gate's output. Each leaf represents one of it's n inputs. All possible n input binary trees are created and logically simulated. The result is stored and compared with a simulation of the logic function description inside the library.

Example: A logic cell *AOI21* is described as $!\,(a\&b|c)$. Simulation of all eight states gives the target vector 10101000. For each of the six permutations of a,b, and c, there are eight different binary tree shapes which are in reverse polish notation:

```
ab+c+  abc++  ab+c*  abc+*  ab*c+  abc*+  ab*c*  abc**
```

By not allowing two identical consecutive operators ++ and $**$, or equivalently no right child of a node having the same operator as the node itself, all redundant gates are removed. Among the remaining 36 binary trees, exactly four match the target vector[7]:

```
ab*c+  ba*c+  cab*+  cba*+
```

Using the rule, that the left operand is closer to the logic output in case of serial circuits, the four trees translate into the four circuits presented in Figure 4.3.

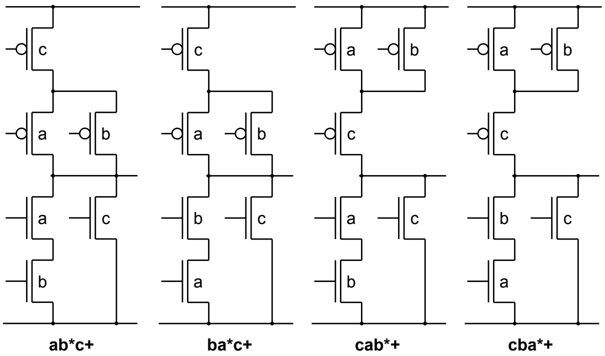


Figure 4.3: Among the 36 different binary trees, these four have the same logical target vector as the *OAI21* gate.

[6] For the sake of simplicity, the meaning of $+$ and $*$, which are usually equivalent to | and & are redefined: $ab+$ now means for NMOS two transistors in parallel and for PMOS two transistors in series with a closer to the gate's output ($*$ vice versa). Thus $ab+$ and $ba+$ both implement the NOR2 function, denoted $!(a\&b)$.

[7] Among the 36 valid circuits are six *NAND3* and six *NOR3* gates as well as four *AOI21* and four *OAI21* gates for each of the three permutations $!(a\&b|c)$, $!(a\&c|b)$ and $!(b\&c|a)$

Step 3: Add drivers

Step 2 is sufficient to find all possible single gate CMOS cells. Real technologies employ a variety of special design styles deviating from single gate CMOS. The most frequent one is a consecutive driver cell, which is supported in four flavours (rf. Figure 4.4):

- **Inverter driver:** All possible CMOS cells (as described in Step 2), having a CMOS inverter at their fanout. With this driver, cells as *AND* and *OR* are realizable. In the binary tree, this design alternative is represented by an additional flag at the trunk node. The target vector is simply inverted. The reverse polish notation of this alternative starts with a $'$. Thus $'ab*$ is one implementation of $a\&b$.

- **Buffer driver:** All CMOS cells having two inverters (a buffer) at their fanout. Logically, this makes no difference, but electrically, this may make sense for high driving strengths. In the binary tree, this design alternative is also represented by a flag at the trunk node. The reverse polish notation gets a leading ". For instance, $"ab + c + d+$ may be an efficient realisation for a *NOR4* with high driving strength.

- **NAND driver:** All possible CMOS cells followed by a *NAND2* gate. The other *NAND2* input is an additional input. Inside the binary tree, this option is realized by allowing the head node to be of the type nand-driver as long as the right child is a leaf (means basic input). For the reverse polish notation, this node is denoted as $/$. For instance $ab + c/$ implements a *NOR2* gate followed by a *NAND2* gate and results in the target vector 10111111.

- **NOR driver:** As NAND drivers. Denoted by a $-$. $ab + c-$ means two consecutive *NOR2* gates as one standard cell. The target vector in this case is 00101010.

Step 4: Compare gates

After all possible implementations are found, these candidates are implemented in SPICE. The width of each transistor W_N^i and W_p^i is roughly predetermined using the input capacitance C^i of the respective input i from the library and the scaling parameters from Step 1 as $W_N^i = \alpha_N \cdot C^i$ and $W_P^i = \alpha_P \cdot C^i$. The idea of this assumption is, that there is always a typical width ratio between an NMOS and a PMOS transistor of one input. For inner inputs (resulting from the drivers), the widths of the single load standard inverter are used in the first place, as no input capacitance is available.

Now the rise and fall timing of all inputs driving the standard load and having the standard input slope is determined. The timing values are normalised (so that they sum up to 1). Now, this relative timing profile is compared to the normalised timing from the library. The implementation having the best fit is chosen as being the most likely implementation of this gate.

The reason, why the timing is normalised is because of the so far undetermined inner inputs from the drivers. Upscaling these transistors will speed up the entire component, but will leave the relative timing profile unchanged.

Step 5: Fine-tune the widths

Having chosen the most likely implementation, the transistor widths are tuned by a gradient search to exactly meet the timing. In case of a buffer driver, the first inverter keeps minimal sizing. After the timing is exactly met, the driving transistors are up-scaled while down-scaling the primary input transistors as long as the total width is minimised.

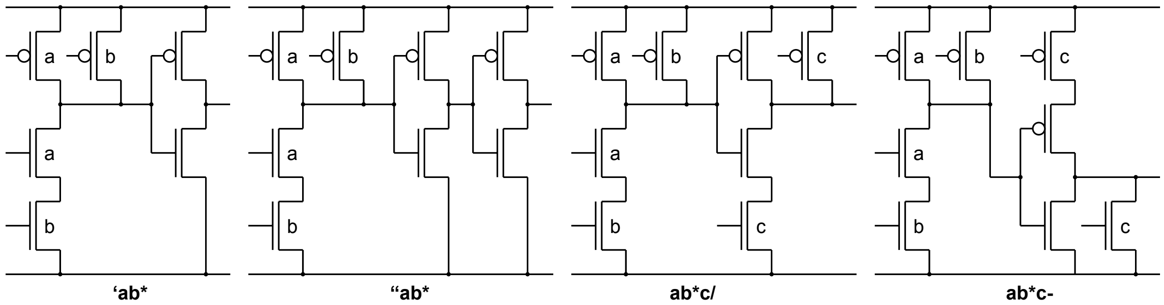


Figure 4.4: In addition to regular CMOS, also inverter-, buffer-, nand-, and nor-driven CMOS cells are supported within the gate analysis flow.

4.4 Generation of RT components in SPICE

This section will discuss, how SPICE netlists for entire RT components can be automatically generated. The flow presented here bases on the assumption that a SPICE description of the entire library and a compact model of the transistors that have to be analysed are available. Recent technologies may already offer this information; alternatively it can be constructed from the gate level *.lib* library description as presented in Section 4.3.

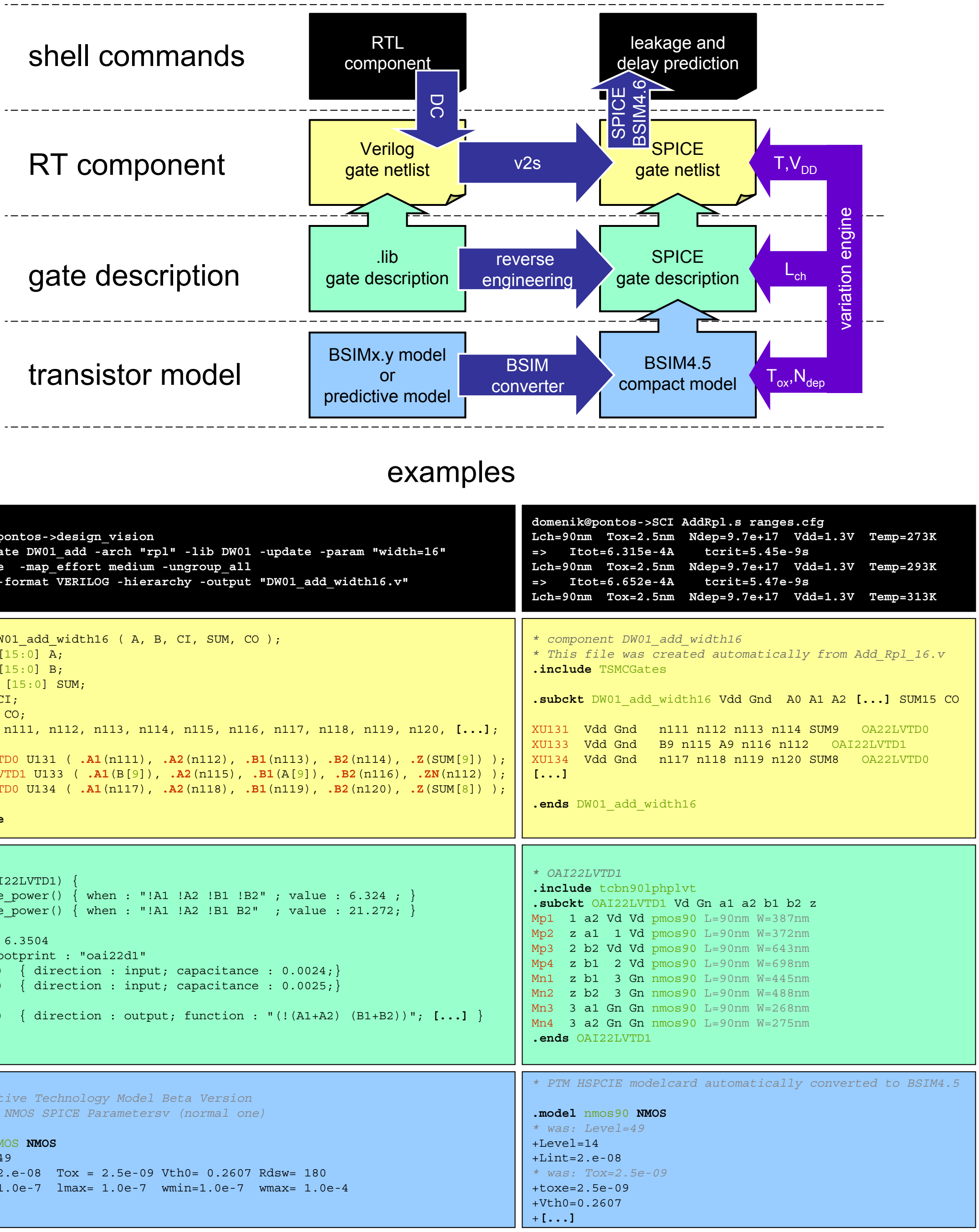
Even though simulating entire RT components, accurately regarding global PTV variation, is extremely time consuming, it needs to be done for a few samples for evaluation purposes: The full evaluation of the RT models was done layer by layer, first comparing the abstraction to gate level versus single gate SPICE simulation and then comparing the abstraction to RT versus the own gate model. Nevertheless some single point simulations for fixed PTV variation conditions neglecting local variation can be obtained within hours per simulation and can increase the confidence of the entire evaluation.

The component generation starts with a commercial RT level synthesis tool (see Figure 4.5). Using this tool, a Verilog description of any RT component can be obtained, and then converted into the SPICE format (Section 4.4). If not available, the gate's SPICE description can be obtained by an analysis of the *.lib* library description (Section 4.3). If an appropriate compact model is not available, it can be generated from a generic or old versioned one (Annex B). Before simulating the component's leakage and delay (Section 4.2.1), a variation engine can generate PTV variations (Section 3.2)

In this work, Synopsys Design Compiler (*DC*) was used. Using the *DC* startup script, some important settings can be made, enabling and simplifying the later SPICE conversion. Certain library gates for which no SPICE description is available (the I/O cells for instance) need to be excluded from the synthesis. *DC* needs to be instructed to avoid tristate gates and hierarchical designs, as both concepts make a conversion more complex. Synthesising an RT component (or even an entire logic function specified in an algorithmic HDL), a flat (no inner sub-functions) gate netlist in Verilog, just consisting of gates known in SPICE, can be obtained.

The self-developed tool *v2s* can now easily convert this Verilog netlist into a SPICE function. In a first step, the Verilog header is analysed and converted into a SPICE sub-circuit. The main task here, except for a generic replacement of the header data, is the conversion of the interconnect. Connections as supply and ground, and optionally N/P-MOS body or sleep-signal have to be added. The busses have to be explicitly initiated bit by bit. In a second step, each line of the Verilog body, instantiating one logic cell has to be converted to SPICE by reordering the statements from Verilog to SPICE syntax and additionally connecting the cell to the special lines mentioned above (supply, ground, etc.).

The result is a SPICE circuit instantiating the SPICE description of each gate of the component and connecting them together in the right way.



```
domenik@pontos->design_vision
->elaborate DW01_add -arch "rpl" -lib DW01 -update -param "width=16"
->compile  -map_effort medium -ungroup_all
->write -format VERILOG -hierarchy -output "DW01_add_width16.v"
```

```
domenik@pontos->SCI AddRpl.s ranges.cfg
Lch=90nm  Tox=2.5nm  Ndep=9.7e+17  Vdd=1.3V  Temp=273K
=>   Itot=6.315e-4A    tcrit=5.45e-9s
Lch=90nm  Tox=2.5nm  Ndep=9.7e+17  Vdd=1.3V  Temp=293K
=>   Itot=6.652e-4A    tcrit=5.47e-9s
Lch=90nm  Tox=2.5nm  Ndep=9.7e+17  Vdd=1.3V  Temp=313K
```

```
module DW01_add_width16 ( A, B, CI, SUM, CO );
  input [15:0] A;
  input [15:0] B;
  output [15:0] SUM;
  input CI;
  output CO;
  wire   n111, n112, n113, n114, n115, n116, n117, n118, n119, n120, [...];

  OA22LVTD0 U131 ( .A1(n111), .A2(n112), .B1(n113), .B2(n114), .Z(SUM[9]) );
  OAI22LVTD1 U133 ( .A1(B[9]), .A2(n115), .B1(A[9]), .B2(n116), .ZN(n112) );
  OA22LVTD0 U134 ( .A1(n117), .A2(n118), .B1(n119), .B2(n120), .Z(SUM[8]) );
  [...]
endmodule
```

```
* component DW01_add_width16
* This file was created automatically from Add_Rpl_16.v
.include TSMCGates

.subckt DW01_add_width16 Vdd Gnd  A0 A1 A2 [...] SUM15 CO

XU131  Vdd Gnd   n111 n112 n113 n114 SUM9   OA22LVTD0
XU133  Vdd Gnd   B9 n115 A9 n116 n112   OAI22LVTD1
XU134  Vdd Gnd   n117 n118 n119 n120 SUM8   OA22LVTD0
[...]

.ends DW01_add_width16
```

```
cell (OAI22LVTD1) {
  leakage_power() { when : "!A1 !A2 !B1 !B2" ; value : 6.324 ; }
  leakage_power() { when : "!A1 !A2 !B1 B2"  ; value : 21.272; }
  [...]
  area : 6.3504
  cell_footprint : "oai22d1"
  pin(A1)  { direction : input; capacitance : 0.0024;}
  pin(A2)  { direction : input; capacitance : 0.0025;}
  [...]
  pin(ZN)  { direction : output; function : "(!(A1+A2) (B1+B2))"; [...] }
}
```

```
* OAI22LVTD1
.include tcbn90lphplvt
.subckt OAI22LVTD1 Vd Gn a1 a2 b1 b2 z
Mp1  1 a2 Vd Vd pmos90 L=90nm W=387nm
Mp2  z a1  1 Vd pmos90 L=90nm W=372nm
Mp3  2 b2 Vd Vd pmos90 L=90nm W=643nm
Mp4  z b1  2 Vd pmos90 L=90nm W=698nm
Mn1  z b1  3 Gn nmos90 L=90nm W=445nm
Mn2  z b2  3 Gn nmos90 L=90nm W=488nm
Mn3  3 a1 Gn Gn nmos90 L=90nm W=268nm
Mn4  3 a2 Gn Gn nmos90 L=90nm W=275nm
.ends OAI22LVTD1
```

```
* Predictive Technology Model Beta Version
* 0.09um NMOS SPICE Parametersv (normal one)

.model NMOS NMOS
+Level= 49
+Lint = 2.e-08  Tox = 2.5e-09 Vth0= 0.2607 Rdsw= 180
+lmin = 1.0e-7  lmax= 1.0e-7  wmin=1.0e-7  wmax= 1.0e-4
+[...]
```

```
* PTM HSPCIE modelcard automatically converted to BSIM4.5

.model nmos90 NMOS
* was: Level=49
+Level=14
+Lint=2.e-08
* was: Tox=2.5e-09
+toxe=2.5e-09
+Vth0=0.2607
+[...]
```

Figure 4.5: **Top**: flow overview, **bottom**: script examples.

5 Experimental assessment

In this chapter, the tool flow necessary for automated model characterisation is presented in Section 5.1. As the model application is performance critical, Section 5.2 discusses an efficient model representation. Finally, in Section 5.3, the evaluation of the sub-models is presented.

5.1 Experimental set-up

The entire model consists of four bottom-up layers: reference transistors, state dependent gate models, RT hard macros and RT soft macros. Each modelling layer needs simulation data for characterisation. Sections 5.1.1 to 5.1.3 will outline the characterisation flow layer by layer. Finally, Section 5.1.4 presents an overview over the entire model building flow.

5.1.1 Reference transistor characterisation

As described in Section 2.3, each reference transistor's leakage is described as an extended Laurent series of the *StPa*, but the *DyPa* enter the equation by an interpolation of the scaling parameters $\alpha_{i,j,k}$ (see Equation 2.8). This transistor model relies on the fact that only very few α's need to be non-zero and that these few α's are easy to interpolate for different *DyPa*.

Most of the times, it turned out, that choosing the right α's[1] for only one central set of *DyPa* and determining only these α's for all other *DyPa* combination resulted in a well defined and smoothly interpolatable model. But for an automatic characterisation methodology, *'most of the times'* is not sufficient. Thus, in this section, a methodology is described, always resulting in a smooth transistor model.

Data acquisition

The characterisation data for each reference transistor was obtained using a C-program *transistorMod*, that can automatically generate SPICE test benches controlling all relevant parameters (process as well as dynamic), and being able to generate each of the eight reference transistors. All the *DyPa*, as well as the channel length can be easily set in SPICE, all other *StPa* can be controlled by generating a copy of the BSIM model card, identifying and modifying the *StPa*, and including this copy into the test bench. After the test bench is written out, *transistorMod* initiates a system call starting SPICE

[1] At one set of *DyPa* (e.g. $\vartheta = 60°C$, $V_{\mathrm{DD}} = 1V$, $V_{\mathrm{BB}} = 0$) the α's which are non-zero are chosen, and at all other parameter sets, only these α's are determined.

in batch mode. Using the SCI tool (see Section 4.2.1), analysing the SPICE raw-file, for one set of parameters, all relevant currents can be simulated.

The entire methodology is encapsuled into a nested loop iterating over a complete set of parameters (*StPa* and *DyPa*). A characterisation run simulating seven samples of each *StPa* and seven for each *DyPa* in each combination for each of the eight references needs $7^6 \cdot 8 = 941192$ SPICE runs, and thus 26 hours of computation time on a $3GHz$ dual core CPU[2]. As a result, a tabulator separated text file, listing the simulated currents for each parameter combination, is generated. The high amount of simulations is only needed for an accurate evaluation of the accuracy of the model (a good model should accurately meet each point and not only fit to two or three sampling points per dimension). For later model use, three samples per dimension, resulting in less than ten minutes computation time is absolutely sufficient.

Polynomial fitting

The problem of the polynomial fitting is to find for each reference transistor a function

$$f\left(\alpha_i(\vartheta, V_{DD}, V_{BB}), L_{ch}, T_{ox}, N_{dep}\right),$$

that can describe the behaviour of the leakage for each fixed set of *DyPa*, and for which the α parameters behave smoothly (interpolatable) between different *DyPa*. This problem was solved as follows:

The solution space was slightly reduced from the general form in Equation 2.8 to be limited to four terms, of which one is the constant one.

$$I(L,T,N) = f\left(\vec{v}, \alpha_i(\vartheta, V_{DD}, VBB)\right) = \exp\left(\alpha_0 + \alpha_1 L^{v_0} T^{v_1} N^{v_2} + \ldots + \alpha_2 L^{v_6} T^{v_7} N^{v_8} \alpha\right) \qquad (5.1)$$

with $v_i \in [-2, -1.5, -1, -0.5, 0, 0, 5, 1, 1.5, 2]$. This results theoretically in $9^9 = 3.87 \cdot 10^8$ possible polynomials.

The *best* solution is defined as follows: For all sets of *DyPa* the samples are divided into regression points and evaluation points. Regression points are the 27 combination of [second lowest/medium/second highest] values of [temperature/supply voltage/body voltage]. Given a $\vec{v}$ under test, for each regression point, the α's are determined by regression to let the function $f(\vec{v})$ have the least square error. For all evaluation points, the α's are not computed by regression, but by linear inter- and extrapolation of the regression points[3]. The mean square error (MSE) is computed for one of the reference transistors for all combinations of *StPa* for each of the *DyPa* sets. Pure regression (without evaluation points) would just lead to the least square errors. The *smoothness* of the αs can not be achieved, by regression only. The method above evaluates the mean square error of interpolated αs and thus automatically chooses smooth αs.

[2] In order to exploit dual core and higher parallelism, the program can be given a start and end sample. The simple call *transistorMod* would have needed 52 hours, calling *transistorMod(1,470596)* and *transistorMod(470596,941192)* independently in two consoles exploited the full parallelism.

[3] using tri-linear inter- and extrapolation

A brute force search of the best $\vec{v}$ will need $3.87 \cdot 10^8$ MSE computations, each requiring 27 regressions of nine α parameters to 7^3 target values. Even on the fastest machines, each regression needed several seconds, thus a brute force solution will be way to expensive (hundreds of years). Instead, a tabu-search was used to heuristically identify a *good* solution instead of the *best* solution. Among all possible heuristics for such classes of problems (gradient search, simulated annealing, integer linear programming, ...), I chose tabu-search for the following reasons:

- Neighbour relations between solutions can be easily identified. Two solutions $\vec{v}$ and $\vec{w}$ are neighbours, if $|\vec{v} - \vec{w}| = 0.5$. If two solutions are neighbours, their behaviour is comparatively similar (only one term in one dimension differs).

- As the solution of neighbours always have very similar cost functions, heuristics, that start at a specific solution and move to a neighbour, compare new and old target function and then keep or reject the last move, usually perform well.

- The solution space has a comparatively small number of dimensions (9). No point has more than 18 neighbours. That means, that probability of finding a local minimum (a point where all neighbours are worse) is relatively high. This disqualifies gradient search (deepest descend).

- The evaluation of a single solution is very expensive. Heuristics as simulated annealing can avoid local minima, for the cost of a very slow convergence[4].

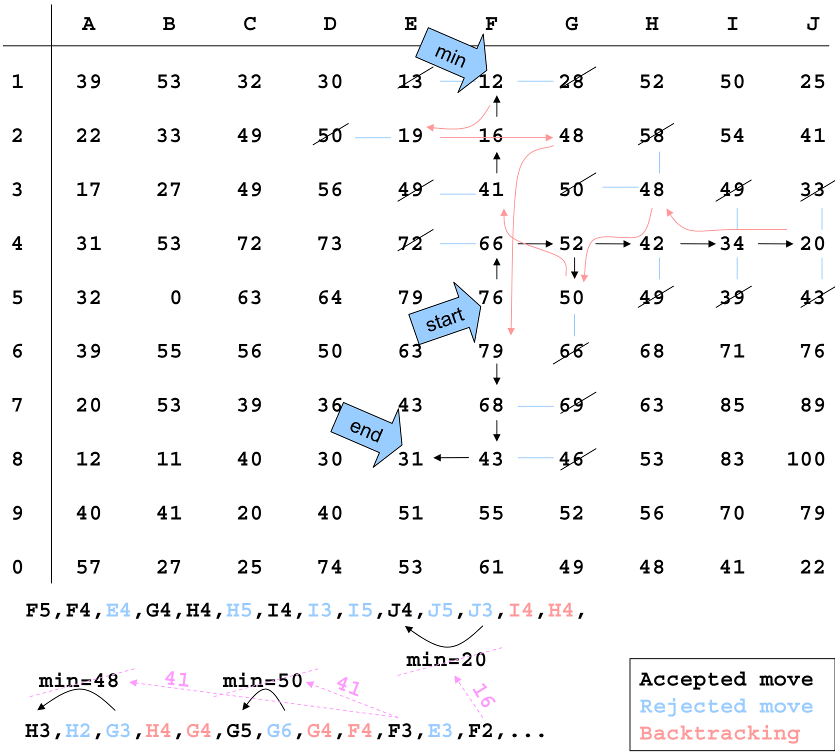
A good choice is thus a fast random descend (undo move if cost function became worse), combined with a large tabu-list and a systematic restart option: The algorithm starts with a random solution. As long as the recent solution is far away from a local minimum, it will find a better solution after trying a few neighbours. The tabu list will avoid trying the same solution twice. The closer the solution approaches to a local minimum, the more neighbours have to be evaluated, thus the heuristic equals more and more a deepest gradient search and will definitely find this local optimum.

A local minimum can be easily identified, as it is the first solution, for which all neighbours are in the tabu-list. The heuristic will save this point and then backtrack, trying to fill the basin of the local minimum with tabu-states. The local minimum is marked as such in the tabu-list. As soon as a better solution than one of the marked minima is found, these minima are unmarked. As soon as a remaining marked minimum has to be removed from the end of the tabu-list, the algorithm restarts with an other solution. The restart solution is not randomly chosen, but the point in the vector space is chosen, having the largest possible distance to all former start-points and to all former real minima[5].

The concept for this search was invented by me, the algorithm itself was implemented in Matlab by my student Marita Blank. Figure 5.1 visualises its basic idea.

[4] There is a proof, that the probability of finding the global minimum instead of a local one rises to 1 when the *cool-down* rate goes to 0. But with a cool-down rate going to 0, the runtime goes to infinity [137].

[5] A real minimum is a minimum that could not be unmarked before the end of the tabu-list



	A	B	C	D	E	F	G	H	I	J
1	39	53	32	30	~~13~~	12	~~28~~	52	50	25
2	22	33	49	~~50~~	19	16	48	~~58~~	54	41
3	17	27	49	56	~~49~~	41	~~50~~	48	~~49~~	~~33~~
4	31	53	72	73	~~72~~	66	52	42	34	20
5	32	0	63	64	79	76	50	~~49~~	~~39~~	~~43~~
6	39	55	56	50	63	79	~~66~~	68	71	76
7	20	53	39	36	43	68	~~69~~	63	85	89
8	12	11	40	30	31	43	~~46~~	53	83	100
9	40	41	20	40	51	55	52	56	70	79
0	57	27	25	74	53	61	49	48	41	22

min
start
end
F5,F4,E4,G4,H4,H5,I4,I3,I5,J4,J5,J3,I4,H4,
min=20
min=48
41
min=50
41
16
H3,H2,G3,H4,G4,G5,G6,G4,F4,F3,E3,F2,...
Accepted move
Rejected move
Backtracking

Figure 5.1: Two dimensional visualisation of the nine dimensional tabu-list descend with backtracking: Starting from F5, a first local minimum is found in J4 and the trace is backtracked to a free neighbour of H4. After reaching F2, for which the target value is 16, all higher valued minima are unmarked. In F1, a new local minimum is found. After a backtracking, the algorithm reaches E8 and the recent minimum F1, was not unmarked and reaches the end of the tabu-list (ten entries in the example). Now a new start point is chosen as far away from F5 and F1 as possible. The algorithm restarts from A1 and converges soon to the global minimum B5 (not visualised).

5.1.2 Gate model characterisation

After having the reference transistors available, a SPICE description of each gate in the technology library is needed. Either, this data is delivered by the technology vendor (who used these SPICE models to characterise the technology library), or it can be obtained, analysing the gate's properties as described in Section 4.3. A script, developed by Marko Hoyer now automatically processes all gates in all driving strengths[6]. For each gate, the respective *.lib* section is parsed to obtain necessary data (for instance the boolean logic function, the rise/fall delays, or the input capacitances). Similarly to the *transistorMod* tool for transistor characterisation, each gate under each input vector is now simulated under 125 different combinations of the *DyPa*. Depending on the size of the technology, this

[6] The user may constrain the gates to be processed, e.g. to exclude special cells as pad-cells.

characterisation may need up to one day on a single machine. Again, the high number of sampling points is just used for an accurate evaluation. A later user (who can rely on the accuracy) will just need to have 10-20 samples, reducing the workload to two hours, which can again be reduced by parallelisation. After the SPICE data for each input vector is available, it is written into a tabulator separated text-file and a prepared Matlab script is started via system call. The Matlab script performs a linear regression and saves the result matrix (eight scaling parameters per gate per input state) to disc.

Remark:

All data necessary for the delay model can be automatically obtained in parallel to the leakage characterisation. For each input pin, with a rising and falling input flank, and under each combination of the other pins in which the gate is sensible for a transition of the pin under test, the delays, as specified in the library[7] is determined by simulation.

5.1.3 RT level model characterisation

The only characterisation data needed for a single RT hard macro (i.e. a model for a given netlist and input vector) is the Verilog description of the component, which can be obtained as described in Section 2.5 using a commercial synthesis tool. When passing the Verilog file and an input vector to the RT level simulator RTSim (developed by Marko Hoyer), the state of each gate is determined by zero delay simulation (see Section 2.5.1) and the gate level models can be evaluated. By summing up the scaling parameters for each gate, these eight parameters describing the leakage of this component can be obtained.

In order to characterise the final RT soft models, the RTChar tool (developed by me) initiates for each component family (e.g. ripple adders) the synthesis of several components with varying bitwidth and input data. RTChar has to distinguish single input (e.g. incrementer), linear (e.g. adder), and dual input (e.g. multiplier) components. A linear component has two inputs, but both have to have the same bitwidth. Each component synthesised is then simulated using RTSim with several input vectors generated by RTChar.

All simulation data from RTSim (eight scaling parameters per simulation) are then passed to a linear regression (implemented as Matlab batch file) via system call, which returns the 96 regression parameters needed for each component family (Section 2.5.2 discusses the meaning of the scaling parameters).

5.1.4 Conclusion

The entire characterisation engine consist of seven single executables:

[7] In each library, I had available, the gate delay is defined as the 50-50 delay. The time between the input and the output passes $50\% V_{DD}$

- the characterisation data generator (simulates the eight references under varying parameters and stores the results inside a huge table)
- the transistor model generator (reads the table from the characterisation data generator and heuristically optimises an analytical function for a smooth regression of all simulation results)
- the gate model generator (reads in the *.lib* description of the technology and the schematic if available and finds the best scaling parameters describing the behaviour of this gate under all input data)
- the RT component synthesiser (synthesises RT components mapped to the *.lib*. I used the commercial tool DesignVision from Synopsys here)
- the RT component simulator (simulates the gate level Verilog description and composes the RT hard macro from it)
- the RT component abstractor (abstracts several hard macros at different input sizes to one RT soft macro)
- the PowerOpt model generator (translates all data to the PowerOpt component data base format)

Figure 5.2 shows the leakage and delay model generation embedded into the entire modelling flow:

Transistors:

At first, the technologies BSIM file (either the specific is available or a generic is used) is analysed by using it simulating the eight reference transistors (Section 2.3). Then, all gates of the technology library are either constructed by a gate analysis (Section 4.3) or they are already available as a SPICE library. The leakage (Section 2.4) and delay (Section 2.6) model builder performs a few SPICE simulations using the BSIM file and the SPICE library to map the leakage to the reference transistors and the delay to the standard inverter. The delay and leakage scaling parameters, together with area and wire-load data, directly parsed from the technology library are stored as the characterisation data.

Gates:

Each RT component is synthesised from the designware library, using the DesignVision to map the target library generating a Verilog netlist. The RT level characteriser (Section 2.5.2) controls the architecture and bitwidth of the components to be designed. Based on the netlist and using the fast delay and leakage models, the best state of the system (the minimum leakage vector) is identified, and (if requested) a performance neutral optimisation improving the best state can be done (Section 1.4.4).

Components:

As described in Section 2.5.1, an RT level leakage model can be build from a Verilog netlist by the tool RTSim. The RTChar tool (Section 2.5.2) controls which RT components are designed, what is the minimum leakage vector of the design, and which input data is used for the RTSim simulation. The result is one RT soft model for each class of RT components with abstract bitwidth and input data dependence and a dedicated *best state*[8] and power gating sub-model (with bitwidth dependency only). As described in Section 2.6, RTChar also generates a bitwidth dependent delay model.

Systems:

For all relevant RT components, the characterisation data and model routines are represented as a dynamic link library (DLL) in the PowerOpt component data base (CDB) format. Inside PowerOpt, the models can be included at runtime enabling system level estimation and optimisation.

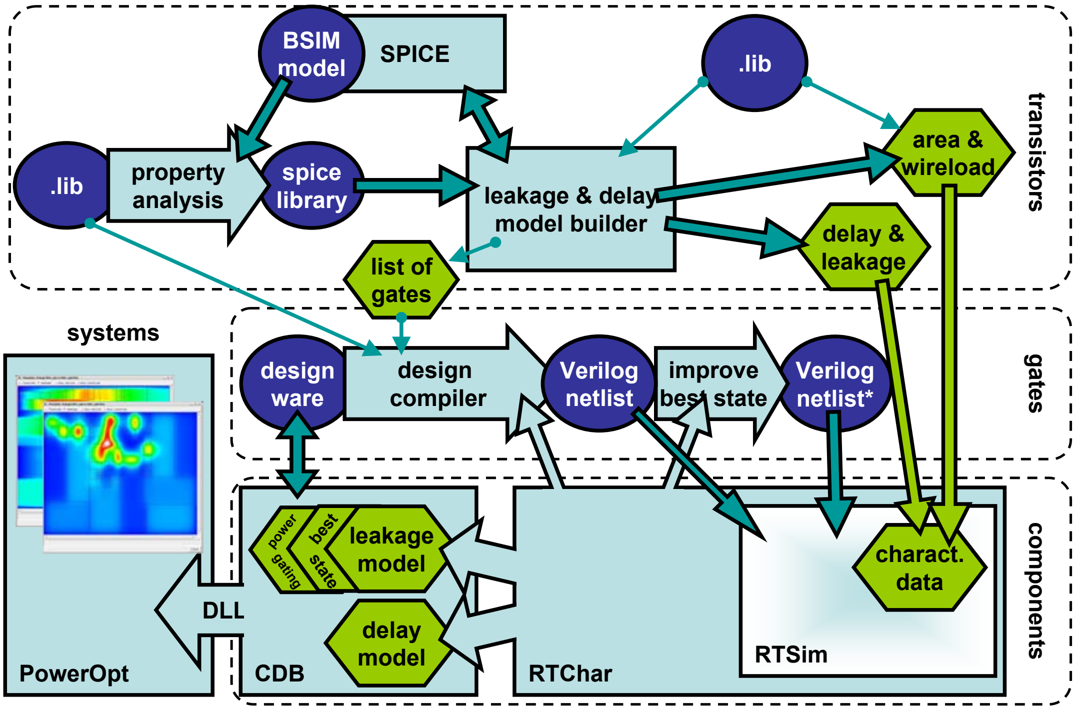


Figure 5.2: See Section 5.1.4 for details

After all updates have been implemented, the characterisation process (without the visualisation and evaluation of the single steps) takes approximately 24 hours characterisation time on a dual processor 3GHz x86 machine.

[8] Within RTChar, the MLV state 1.4.4 is identified heuristically [96] and a simplified data-independent model for a component family in MLV is generated.

5.2 Model representation and interface

The entire leakage model was developed for being used inside the PowerOpt tool. In order to find the best solution, thousands of designs have to be compared, each consisting of hundreds of RT components. Each system is simulated at system level resulting in a simulation trace of thousands of input pattern per component. The variation engine initiates hundreds of system instances. Thus, 10^9 to 10^{11} single model evaluations have to be done in a comparatively short time (minutes to hours). A powerful model representation and a model interface, detecting and avoiding redundant computations is thus mandatory.

5.2.1 Transistor level

For the transistor layer of the model, the representation is quite simple. The main reasons are on one hand, that there is not so much data to be organised, and on the other hand, that effects as correlation do not matter from a single transistor view. Each of the eight reference transistors is described by a simple function with nine independent exponents and four scaling parameters resulting from a piecewise tri-linear interpolation.

$$I_r = \exp\left(\alpha_{0,r} + \sum_{i=1}^{3} \alpha_{i,r} \cdot L_{ch}^{l_{i,r}} T_{ox}^{t_{i,r}} N_{dep}^{n_{i,r}}\right) \tag{5.2}$$

$$\begin{aligned}\alpha_{i,r} &= \beta_\vartheta \beta_{DD} \beta_{BB} \cdot f_{i,r}(\vartheta^<, V_{DD}^<, V_{BB}^<) \\ &+ (1-\beta_\vartheta)\beta_{DD}\beta_{BB} \cdot f_{i,r}(\vartheta^>, V_{DD}^<, V_{BB}^<) \\ &+ \beta_\vartheta(1-\beta_{DD})\beta_{BB} \cdot f_{i,r}(\vartheta^<, V_{DD}^>, V_{BB}^<) \\ &+ \ldots\end{aligned}$$

$$\beta_\vartheta = \frac{\vartheta^> - \vartheta}{\vartheta^> - \vartheta^<} \ ; \ \beta_{DD} = \frac{V_{DD}^> - V_{DD}}{V_{DD}^> - V_{DD}^<} \ ; \ \beta_{BB} = \ldots$$

where for instance $\vartheta^<$ is the largest sampled temperature below the parametrised temperature ϑ and $\vartheta^>$ is the next larger temperature. If ϑ is smaller than the smallest sampled value, $\vartheta^<$ is the smallest available temperature in the table. For $\vartheta < \vartheta^<$ and $\vartheta > \vartheta^>$ the β_ϑ becomes negative and Equation 5.2 is automatically extrapolating.

More challenging was to adapt the model interface to speed up the typical use-case, which is access from the gate and RT level layers of the model. Both model layers have the following access pattern:

1. If one reference transistor is accessed for a combination of *DyPa* and *StPa*, usually the other seven references will also be accessed directly afterwards with the same parameter settings (called a full access).

2. After a full access, it is likely that only *StPa* change before the next full access. The user (a higher level model) knows this, and can give the information to the interface.

3. It may also occur frequently, that only the *StPa* or only *DyPa* or even no parameter changes between two full accesses, but the user does not know (typically, if more than one user accesses the models).

In order to avoid redundant recomputation, the model is implemented as follows:

- The constructor function gets one pointer to a float array of size 873 where sequentially, the nine sampling values $\vartheta^1, \vartheta^2, \dots, V_{\mathrm{BB}}^3$, and the 864 function values for all eight reference transistors r, for all four scaling factors i and for all 27 combinations of the sampling values $(\vartheta^1, V_{\mathrm{DD}}^1, V_{\mathrm{BB}}^1), (\vartheta^2, V_{\mathrm{DD}}^1, V_{\mathrm{BB}}^1), \dots$ are stored. The constructor also sets up several cache data, as described below. As all relevant model data is initialised by a single float array, the initialisation speed is significantly improved. This is relevant, as new models are constructed and destructed frequently in PowerOpt.

- The *coloriseDynamic* function takes as input the three *DyPa*. The last set of inputs and the last result of the function is cached, so if the function is called consecutively with the same parameters, no reevaluation is needed (to speed up access pattern number 3). The $\vartheta^<, \vartheta^>, V_{DD}^<, \dots$ parameters and the $\beta_\vartheta, \beta_{DD}, \beta_{BB}$ parameters are computed once for all transistors. Now, for each transistor $r \in [0; 7]$ and four terms $i \in [0; 3]$, the function values $f_{i,r}(\vartheta^<, V_{DD}^<, V_{BB}^<), f_{i,r}(\vartheta^>, V_{DD}^<, V_{BB}^<), \dots$ are read from a five dimensional table $(8x4x3x3x3)$[9], resulting in 32 $\alpha_{r,i}$ values after interpolation, which are stored. If the user knows that consecutive estimates for the same *DyPa* are needed, the *coloriseDynamic* function has to be called just once before the first estimation (to speed up access pattern number 2).

- The *coloriseProcess* function takes as input the three *StPa*. The same caching mechanism described above is used here again. This function precomputes the three polynomials $L_{ch}^{l_{i,r}} T_{ox}^{t_{i,r}} N_{dep}^{n_{i,r}}$ per transistor.

- The *estimateFast* function takes as input the transistor identifier r and returns the exponent of the scalar product of the parameter and the exponent vector (according to Equation 5.2). This speeds up the access pattern number 1.

- The *estimateFastArray* function takes no input and directly returns an array of eight transistor estimates. This is a further speedup for access pattern number 1, as it avoids seven function calls and some array accesses can be improved.

- Finally, there is an *estimateFull* function which is taking all six *DyPa* and *StPa* and the transistor identifier and forces a full reevaluation.

The *coloriseDynamic* function including the caching check needs 349 arithmetic, logical, read, and write operations. The *coloriseDynamic* function needs up to three square root computations and

[9] The indices of the last three dimensions can be precomputed once for all transistors and terms.

(depending on the exponents)[10] up to 63 operations. For the *estimateFast* function, 14 operations and one exponential function are needed. Thus for one full access, the colorise methodology and the clever exponent computation reduces the number of operations from 3408 operations, and 80 exponential functions to 524 operations and four exponential functions which is more than seven times faster. With more and more accesses for the same set of *DyPa*, this speedup even improves.

5.2.2 Gate level

In contrast to the transistor model, the gate level model is just an intermediate result, which is used for RT level model generation. Single gate estimates are not needed in PowerOpt. Thus, it was not optimised for high performance, but implemented straight forward. When initialising, the C++ class *gateModelHandler* parses a configuration file, in which for each gate the following data is stored:

- the full logic name, as it will occur in a Verilog netlist
- the number of inputs n
- the full logic names of the inputs of the gate as they will occur in a Verilog netlist
- the logic function as a string of 2^n zeros and ones (e.g. NAND2 = *1110*)
- the eight scaling parameters per input vector as a tabulator and newline separated array (eight entries per line, 2^n lines)
- some delay relevant data extracted from the *.lib* (e.g. input capacitance per pin)
- the representation of the delay model

During configuration, all data from the configuration file is used to initialise a new instance of the class *gateModel*, in which all data is stored in a straight forward way. After configuration, the *gateModelHandler* can return the handle to a *gateModel* when getting its logic name by a simple string comparison. Global values for the dynamic and variation parameters can be directly set in *gateModelHandler*, for each *gateModel* this settings can be overwritten for evaluation purposes (e.g. analysing the effect of parameter gradients).

The main leakage related functions of *gateModel*, are

- *setInput*, where each input (by logic name or by integer identifier) can be set to 0 or 1,
- *getParameters*, which returns the eight scaling parameters for the input vector set by *setInput*
- *getLeakage*, which colorises the transistor models to the recent *DyPa* and *StPa* if necessary, requests the transistor estimation using the *estimateFastArray* function, and returns the linear combination of scaling parameters and transistor estimation.

[10] The square root of each input parameter is computed on demand and cached. Terms as $x^{-1.5}$ are then computed as $\sqrt{x}/(x \cdot x)$.

5.2.3 RT level

The characterisation methodology as described in Section 5.1 is implemented into the program RTChar (see Figure 5.2). RTChar is a central element of the entire model building flow as it controls the generation of Verilog characterisation netlists and the simulation of these netlists under various data. After regression of the characterisation data to the targeted function, it automatically generates C++ code segments which can be included into the respective PowerOpt CDB models (see Figure 5.2). After model characterisation, the CDB library is compiled to a dynamic library, which can be linked at runtime to the PowerOpt tool.

These code segments contain all the $8 \cdot 12$ parameters, which are sufficient to describe an entire family of RT components under different data and they look as follows:

```
// this function was automatically generated by RTChar
// manual changes will be lost after next characterisation
float CdbComponent::getScalingParameter
       ( parameterType ID, float pX, int bwA, int bwB )
{
  switch( ID )
  {
  case NC:
    return ( 0.023*pX + 0.325 )*pX + 1.627
        +  (( 0.543*pX + 1.643 )*pX + 6.437 )*float(bwA)
        + ((( 0.475*pX + 1.774 )*pX + 6.534
        +    ( 0.091*pX + 0.187 )*pX + 0.393 )*float(bwA) )*float(bwB);
  case NO:
    return ...
  }
}
```

5.2.4 Overview

In PowerOpt, all relevant information for each RT component family is stored in an instance of the C++ class CdbComponent. All data enters the CDB in form of C++ code fragments or entire functions (as shown above). After characterisation of all metrics (dynamic power, area, delay, leakage power), the code fragments are merged (actualised by the C++ include command) and compiled to a dynamic library.

The description of the process variation, which is assumed to be an inherent property of a technology, is not included to each component, but instead a new class *CdbVariationEngine* is generated. *CdbVariationEngine* consist of a template (identically for each technology), in which all relevant methodology is implemented (see Section 3.2) and a custom initialisation function, which is automatically generated after characterisation and included during compilation of the CDB.

A model request from PowerOpt starts with a colorisation of the component, where the architecture (component family) and bitwidth are specified. These data are stable for the entire lifetime of the model, thus several intermediate results can be precomputed. After colorisation, the input data from the behavioural simulation is fed into the model. For each new input vector, the hamming distance is computed in order to compute the dynamic power. For leakage estimation, the scaling parameters for this bitwidth and signal probability (*getScalingParameters*) are computed. These scaling parameters are directly multiplied with the time, that has passed between two input pattern. Each of the eight scaling parameter - time products can be summed up over the simulation time. After the last input vector was fed, the scaling parameter - time products are divided by the total time, resulting in the effective average scaling parameter.

Explanation:

As the RT level model is not linear in signal probability, it is not possible to precompute the *average signal probability* in advance. On the other hand, the scaling parameters themselves cannot be simply averaged, because the total leakage depends on how long the component was in a low or high leakage state.

After the component was colorised and fed with the simulation trace, all estimations (as dynamic power) can be requested (in fact, delay and area estimates does not even need the input trace and are available directly after characterisation). For the leakage and delay estimation, the *DyPa* and the delay bin and power sub-bin (see Section 3.2) have to be specified to access the functions *getDelay* and *getLeakage*. Both binning parameters are translated by the variation engine into process typical *StPa*. Having available all six parameters (*DyPa* and *StPa*), the eight reference parameters, as well as the delay reference can be evaluated. The linear combination of the effective scaling parameters with the references gives the average leakage current of the component.

If *DyPa* change when the input trace is fed into the system (for instance due to dynamic voltage scaling), the scaling parameter - time products have to be directly translated into leakage values. As not effective width, but a width time product enters the transistor model, the result is not in the unit Ampere, but Coulomb. These charges for each part of the simulation trace, where the *DyPa* are constant, can be multiplied with the recent voltage, resulting in an energy and then be summed up.

The following pseudo code illustrates the dependencies described above:

```
colorize( architecture, bitwidth )
{
  Area = computeArea
  Delay = computeNominalDelay // no variation

  // global parameters
  DynamicEnergy = 0
  ParameterTimeProduct[0-7] = 0
  LeakageEnergy = 0
  Lch = Tox = Ndp = <invalid>
  Temp = Vdd = Vbb = <invalid>
}
setVariation{ delayBin, powerBin }
{
  Lch = variationEngine.getLength( delayBin, powerBin )
  Tox = variationEngine.getOxideT( delayBin, powerBin )
  Ndp = variationEngine.getDoping( delayBin, powerBin )
  transistors.colorizeProcess(Lch,Tox,Ndp)
}
feed( inputVec , time )
{
  DynamicEnergy += computeDynamicE( inputVec, lastInputVec )
  for t = 0 to 7
  {
    LeakageParameterTimeProduct[t] += (time-lastTime)
          * getScalingParameter( bitwidth , lastInputVec )
  }
  lastInputVec = inputVec
  lastTime = time
}
setDynamicParameters(newTemp, newVdd, newVbb)
{
  refTrans[0-7] = transistors.estimateFastArray()
  for t = 0 to 7
  {
    LeakageEnergy += Vdd * ParameterTimeProduct[t] * refTrans[t]
    ParameterTimeProduct[t] = 0
  }
  Temp = newTemp
  Vdd = newVdd
  Vbb = newVbb
  transistors.colorizeDynamic(Temp,Vdd,Vbb)
}
getLeakageEnergy
{
  // force refresh of leakage Energy
  setDynamicParameters(Temp,Vdd,Vbb)
  return LeakageEnergy
}
```

5.3 Evaluation

Besides the core idea of a bottom up leakage model, this work consists of several secondary models for parameter determination: The thermal (Section 3.3) and IR-drop (Section 3.4) mapping, the body voltage and nominal V_{DD} island optimisation (Section 3.5) and the variation engine (Section 3.2).

After the evaluation of the leakage model itself, which is presented in the following sections, the thermal model will be evaluated against the direct finite elements implementation of the thermodynamical differential equations, and the IR-drop model will be compared to SPICE simulations, also performing a time-discrete numerical solution of the respective electrical differential equations. The body and nominal supply voltage optimisation does not have to be evaluated in terms of prediction accuracy, as it does not predict or estimate, but select the body voltage and the nominal supply voltage.

In terms of prediction accuracy, the variation engine equals a Monte Carlo based multiple point estimation, thus it can be expected, that the accuracy of a leakage estimation under variation is as good as the leakage model itself. Nevertheless, a meaningful evaluation of the variation engine predicting the leakage (and performance) behaviour of a large number of instances underlying a process variation behaviour, which is characterised by silicon measurement of a real industrial technology or even a direct comparison between model and silicon would be very desirable. Unfortunately, such an evaluation is not possible, mainly due to intellectual property problems: An exact description of the behaviour of the *StPa* of an industrial technology is highly confidential, as it allows a direct conclusion to the technology and quality of the process.

For the remainder, the respective evaluation is presented below as follows: The semi-analytical transistor models, as well as the gate models are evaluated directly against SPICE simulation results. For the RT level data and bitwidth abstraction, the additional error of the respective abstraction step is analysed in comparison of the gate level model. Finally, the resulting RT soft macro model is again compared to SPICE, but for the sake of computational complexity not in a full range evaluation, but only for a number of single point comparisons.

5.3.1 Transistor models

The transistor model evaluation can be made straight forward, as each model prediction can easily be also obtained in SPICE. Three technologies are analysed, the 90*nm* and 65*nm* PTM bulk CMOS, as well as the 45*nm* high-k metal gate PTM. For each of the eight reference transistors, characterisation data is obtained for combinations of the following input parameters (Table 5.1):

Using this characterisation data, the models are generated as described in Section 2.3. The semi-analytical terms, obtained by the model generator are presented in Table 5.3. Then for nearly 100000

	$45nm$		$65nm$		$90nm$	
Parameter	min	max	min	max	min	max
$\vartheta[C]$	40	100	40	100	40	100
$V_{DD}[mV]$	800	1100	900	1200	1000	1300
$V_{BB}[mV]$	-150	225	-150	225	-150	225
$L_{ch}[nm]$	41.0	49.1	59.2	70.9	81.9	98.1
$T_{ox}^{N}[\text{Å}]$	8.19	9.81	16.8	20.2	18.7	22.3
$T_{ox}^{P}[\text{Å}]$	8.37	10.0	17.7	21.3	19.6	23.4
$N_{dep}^{N}[10^{18}/cm^{3}]$	5.92	7.08	2.31	2.77	1.77	2.11
$N_{dep}^{P}[10^{18}/cm^{3}]$	2.55	3.05	1.70	2.04	1.30	1.56

Table 5.1: Parameter ranges for transistor model characterisation. Oxide thickness and channel doping are given individually for NMOS and PMOS.

single points, each model prediction is compared to an according SPICE simulation. The model accuracy is visualised in Annex C and the resulting standard deviations and maximum errors are reported in Table 5.2.

As can be seen in Annex C, the absolute deviation between simulation and model is always between one and three orders of magnitude smaller, than the simulation data itself. Due to the characterisation procedure, optimising for lowest relative error, the largest errors usually occur for the largest simulation values. The standard deviation is always below 4% in $45nm$, 6.5% in $65nm$ and 7.5% in $90nm$, and the worst case single point relative error is at 15%/35%/37% for the three technologies. These extreme errors occur always for corner points, where the deviation in most dimensions is at its maximum or minimum.

The errors for the references dominated by gate leakage (N0,N1,P0,P1) are significantly lower, as the behaviour of the simulation data is far simpler. Most of these references are not really six dimensional functions, but rather two dimensional ones, as only oxide thickness and supply voltage have a significant influence on the simulation data. Table C.1 presents the model dependency of these transistors to the other four dimensions, which is either perfectly independent, or only enters with a minor contribution in the order of pico-amps.

5.3.2 Gate models

As described in Section 2.4, each gate in the technology library in each input state is modelled as a linear combination of the eight reference transistors. The scaling parameters can be determined by just a few simulation results in just some input parameter dimensions, as they are independent of all the *DyPa* and *StPa*. As the references always refer to a transistor of $1\mu m$ width, these eight scaling parameters can be interpreted as effective width in μm.

	45nm		65nm		90nm	
	NMOS	**P**MOS	**N**MOS	**P**MOS	**N**MOS	**P**MOS
standard deviation [%]						
Channel	3.87	2.42	5.77	6.40	4.86	7.36
0-input	0.88	1.08	0.82	0.95	0.68	2.42
1-input	1.15	0.98	1.31	0.84	1.02	0.69
Stacked	3.24	2.35	3.76	5.81	3.31	4.92
maximum error [%]						
Channel	15.51	10.10	26.47	20.37	23.09	30.83
0-input	1.64	4.17	1.48	5.68	1.22	7.18
1-input	2.85	2.30	2.94	1.62	2.04	1.24
Stacked	8.70	6.49	14.08	34.23	14.94	37.23

Table 5.2: Standard deviation of the semi-analytical model for the eight reference transistors. Evaluation compares the model to BSIM simulation for the 90nm and 65nm PTM bulk CMOS and the 45nm high-k metal gate PTM technology. *DyPa* variation range is described in Section 5.3.1. Highest errors are obtained for simultaneous and extreme deviations of all *StPa* (which is very unlikely to happen)

Simulation description

In order to evaluate the error, introduced by the linear regression, three technologies are analysed:

- the 90nm TSMC general purpose, low threshold technology, characterised for typical case (tcbn90gthplvttc)
- the 65nm STM medium threshold technology (CORE65LPSVT)
- a 45nm generic technology using the 45nm metal gate/high-k PTM transistors and the same circuits as the 65nm technology with scaled transistor widths (at evaluation time, OFFIS had no real 45nm technology available.)

As the *.lib* format does not support process variation, all recent technologies are characterised in several process corners like fastest, slowest or typical. The models presented in this thesis can explicitly regard process variations. Thus for all technologies analysed, the typical case is the best choice as it can be used to determine the average parameter values.

For evaluation, the following gates are taken from the SPICE description of the library:

- NAND and NOR with two, three, and four inputs
- the AOI and OAI gates in 21, 211, and 22 size

AOI (AND-OR-Inverted) means that the inputs are combined in groups of AND gates and their results are subjected to a NOR gate. AOI gates can be build by a single layer CMOS structure

$45nm$	
NC	$\alpha_0 + \alpha_1 \cdot L^{0,5}T^{-0,5} + \alpha_2 \cdot T^{-0,5} + \alpha_3 \cdot L^{-2}$
N0	$\alpha_0 + \alpha_1 \cdot T^{-1} + \alpha_2 \cdot T^{0,5} + \alpha_3 \cdot T^{1,5}N^{0,5}$
N1	$\alpha_0 + \alpha_1 \cdot T + \alpha_2 \cdot L^{-0,5} + \alpha_3 \cdot T^{0,5}$
NS	$\alpha_0 + \alpha_1 \cdot T + \alpha_2 \cdot L^{-2}T^{0,5} + \alpha_3 \cdot LTN^{0,5}$
PC	$\alpha_0 + \alpha_1 \cdot L^{-2}T^{0,5} + \alpha_2 \cdot L^{-1,5}T^{0,5} + \alpha_3 \cdot L^{-2}T^{0,5}N^{0,5}$
P0	$\alpha_0 + \alpha_1 \cdot L^{-0,5} + \alpha_2 \cdot T^{0,5} + \alpha_3 \cdot T$
P1	$\alpha_0 + \alpha_1 \cdot T^{-2}N^{-1} + \alpha_2 \cdot T^{-0,5}N^{-0,5} + \alpha_3 \cdot T^{2}N^{0,5}$
PS	$\alpha_0 + \alpha_1 \cdot L^{-1}T^{0,5} + \alpha_2 \cdot L^{-2}T^{0,5} + \alpha_3 \cdot L^{2}T^{0,5}N^{-0,5}$
$65nm$	
NC	$\alpha_0 + \alpha_1 \cdot L^{-1}T^{0,5} + \alpha_2 \cdot L^{-1,5}T^{0,5} + \alpha_3 \cdot L^{-2}T^{0,5}$
N0	$\alpha_0 + \alpha_1 \cdot L^{0,5}T^{2}N^{0,5} + \alpha_2 \cdot T^{-0,5} + \alpha_3 \cdot T^{0,5}$
N1	$\alpha_0 + \alpha_1 \cdot L^{0,5} + \alpha_2 \cdot T^{-0,5} + \alpha_3 \cdot T^{-1}$
NS	$\alpha_0 + \alpha_1 \cdot L^{-2} + \alpha_2 \cdot L^{-1,5} + \alpha_3 \cdot L^{-1}T^{1,5}N^{0,5}$
PC	$\alpha_0 + \alpha_1 \cdot L^{1,5} + \alpha_2 \cdot L^{-2}T^{-1,5}N^{-0,5} + \alpha_3 \cdot L^{-2}T^{0,5}$
P0	$\alpha_0 + \alpha_1 \cdot L^{0,5}T^{-0,5}N^{-0,5} + \alpha_2 \cdot T^{-0,5} + \alpha_3 \cdot T^{0,5}$
P1	$\alpha_0 + \alpha_1 \cdot T^{-1} + \alpha_2 \cdot T^{-0,5} + \alpha_3 \cdot T^{1,5}N^{0,5}$
PS	$\alpha_0 + \alpha_1 \cdot L^{-2} + \alpha_2 \cdot L^{0,5} + \alpha_3 \cdot L^{-1,5}T^{2}N^{0,5}$
$90nm$	
NC	$\alpha_0 + \alpha_1 \cdot L^{2}T^{-0,5}N^{-0,5} + \alpha_2 \cdot L^{2}T^{-0,5}N^{-1} + \alpha_3 \cdot L^{-2}$
N0	$\alpha_0 + \alpha_1 \cdot T^{0,5} + \alpha_2 \cdot T^{-0,5} + \alpha_3 \cdot L^{0,5}T^{2}N^{0,5}$
N1	$\alpha_0 + \alpha_1 \cdot T^{0,5} + \alpha_2 \cdot T + \alpha_3 \cdot L^{-0,5}$
NS	$\alpha_0 + \alpha_1 \cdot L^{-2}T^{0,5} + \alpha_2 \cdot L^{-1,5}T^{0,5} + \alpha_3 \cdot L^{2}T^{0,5}N^{-0,5}$
PC	$\alpha_0 + \alpha_1 \cdot L^{-1}TN^{0,5} + \alpha_2 \cdot L^{1,5}T^{1,5}N + \alpha_3 \cdot L^{-1,5}TN^{0,5}$
P0	$\alpha_0 + \alpha_1 \cdot T^{0,5} + \alpha_2 \cdot T + \alpha_3 \cdot L^{0,5}$
P1	$\alpha_0 + \alpha_1 \cdot T^{0,5} + \alpha_2 \cdot T^{-0,5} + \alpha_3 \cdot L^{0,5}T^{2}N^{0,5}$
PS	$\alpha_0 + \alpha_1 \cdot L^{-1,5}T^{0,5} + \alpha_2 \cdot L^{-2}T^{0,5} + \alpha_3 \cdot L^{2}T^{0,5}N^{-0,5}$

Table 5.3: Automatically determined transistor models for the $90nm$ and $65nm$ PTM bulk CMOS and the $45nm$ high-k metal gate PTM technology. For each reference transistor, the leakage dependency to the global variation of the channel length L, the oxide thickness T and the channel doping N is determined analytically. The dependency to the temperature and voltages is captured within the α's.

with just two transistors per input; OAI gates behave respectively. The AOI22 gate for instance implements $!(a_1 \cdot a_2 + b_1 \cdot b_2)$. For each of the gates the implementation with driving strength 1 is taken.

Using SPICE, each of these twelve gates in each of the three technologies is simulated in each input state with each combination of these parameters:

- The channel length, oxide thickness, and channel doping are individually varied in seven steps each from 91% to 109% of the nominal value.
- The temperature is simulated in steps of $12K$ from $313K$ (approximately $40°C$) to $373K$ (approximately $100°C$)
- The supply voltage is varied from $1.0V$ to $1.3V$ for $90nm$, from $0.9V$ to $1.1V$ for $65nm$ and from $0.8V$ to $1.0V$ for $45nm$.
- The body voltage is varied from $-0.15V$ to $+0.225V$ for NMOS and from $V_{\text{DD}} - 0.225V$ to $V_{\text{DD}} + 0.15V$ for PMOS

Overall, $3.2 \cdot 10^7$ single SPICE simulations had to be made which lasts 40 hours on a single core $3GHz$ machine but can be perfectly distributed to a computing cluster.

Gate characterisation:

For each gate in each state in each technology, a few simulations with nominal *StPa* and nominal body voltage, but in each combination of supply voltage and temperature (36 sampling points) are chosen. The eight reference transistors are evaluated exactly under these 36 conditions. Then, a linear regression is used to explain the sampling points of the gates as good as possible with the eight reference transistors. The resulting scaling parameters (effective widths) are assumed to be fixed from now on.

Evaluation:

Now, for each of the 74000 sampling points for each combination of *DyPa* and *StPa*, the transistor models are evaluated and linearly combined according to the effective widths, that have been determined using only the 36 samples. For each gate in each state and each technology, the standard deviation ($\text{std}(model - real)/\text{mean}(real)$) and maximum error ($\max(|model - real|)/\text{mean}(real)$) are computed. Figures 5.3, 5.4, and 5.5 show all the results, and Table 5.4 summarises the average error, standard deviation as well as maximum error, occurring for all gates in all input states analysed.

Evaluation results

Even though an error of 2% to 10% standard deviation is more than acceptable to predict the leakage of gates, that can vary by four or five orders of magnitude with their input parameters, the major remaining source of inaccuracy can be identified:

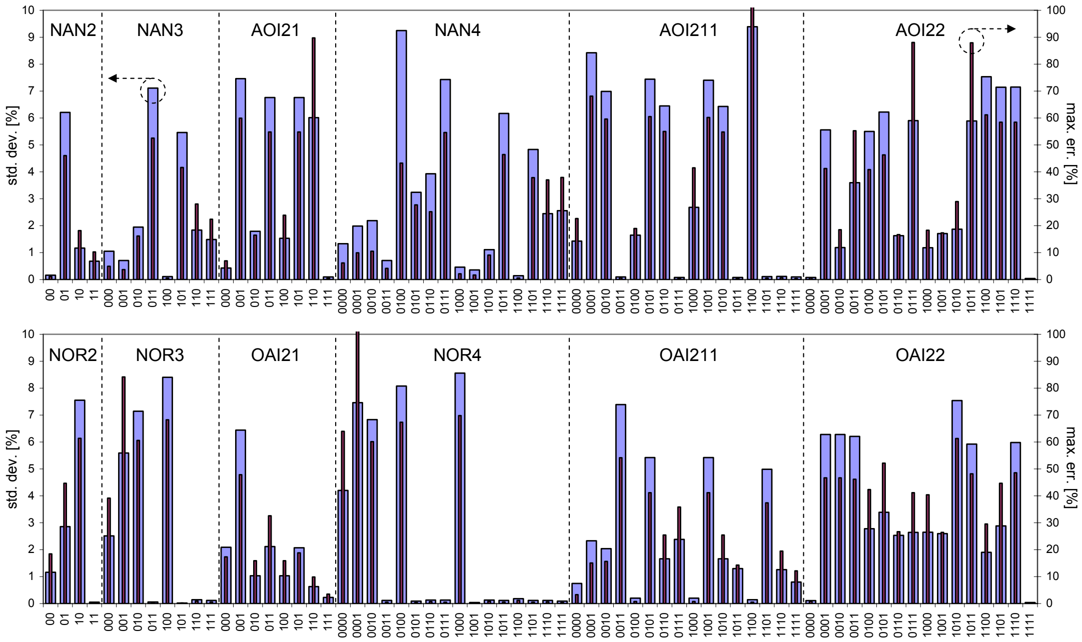


Figure 5.3: Standard deviation and maximum error for the $90nm$ TSMC low threshold technology. Highest errors are introduced for gates with a stack of transistors with conducting and locking transistors, where a conducting transistor faces towards output (such as NAND2@01 or NOR2@10).

Highest errors do always occur for transistor stacks, in which the transistor, closest to the output is conducting, but the entire (pullup- or pulldown-) network is locking. This effect can be understood analysing the NAND2 gate in 10 state (the N1 transistor closer to the output is at V_{DD}, the other one (N2) at ground): In this case, the output is at V_{DD} due to the conducting P2 transistor and the internal node between N1 source and N2 drain is charged up. While the source of N1 is increased, the gate-source voltage is reduced. As soon as V_{gs} goes below V_{th}, the N1 transistor locks and the intermediate node saturates at $V_{\mathrm{DD}} - V_{\mathrm{th}}$. This lets N2 enter a condition, which is not covered by the eight references, in which gate, drain, and source voltage is always either ground or V_{DD}.

It was evaluated, that using two additional reference transistors with a locking-conducting PMOS or NMOS stack will reduce these errors to approximately halve the magnitude for these cases, but the two additional references did not behave well in the transistor modelling step.

The $65nm$ technology showed the highest errors. This effect is most likely caused on the one hand by the gate leakage[11], which is harder to capture at gate level with linear regression, and on the other hand, by the industrial compact model showing much more detail, that the predictive one, used for $45nm$.

[11] the gate leakage is maximal for the $65nm$ technology. The $90nm$ transistors have thicker oxide by design, and the $45nm$ transistors have much thicker oxide because of the high-k material employed here.

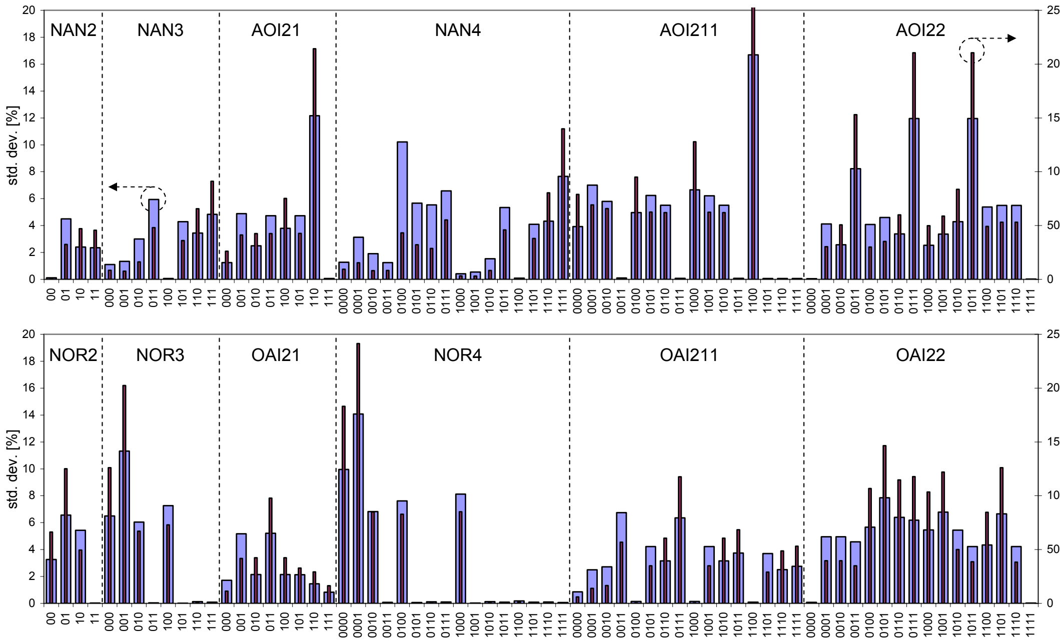


Figure 5.4: Standard deviation and maximum error for the 65nm STM standard threshold, low power technology. Due to the much higher gate leakage, the errors are around 10% standard deviation.

technology	ASD	MEPG	ME
90nm	3.0%	29.3%	134%
65nm	3.8%	51.3%	274%
45nm	0.6%	4.9%	15.3%

Table 5.4: ASD: Average standard deviation, MEPG: Maximum error per gate (averaged over all states), ME: maximal error for all gates and all input states evaluated

Discussion of the effective widths

For most small gates, the results of the regression (see Table 5.5) are meaningful, for larger gates, the regression optimises all eight references and the effective widths do no longer correlate with the physical widths of the transistors of the gate. For instance, for the NAND gate with both inputs low, the NMOS part can be described by an NS reference scaled down to the average width of the NMOS channels. The additional N0 tunnelling has only halve the width of the tunnelling NMOS transistor, as the reference is tunnelling from drain and source to gate, and the transistor here sees only drain to gate leakage. The PMOS part is conducting and thus does not see any subthreshold leakage, as indicated by the low effective widths for PC and PS. The P0 reference has approximately the sum of both transistor lengths.

For the NAND2 gate with the NMOS closer to the output (A) low and the other one high, the

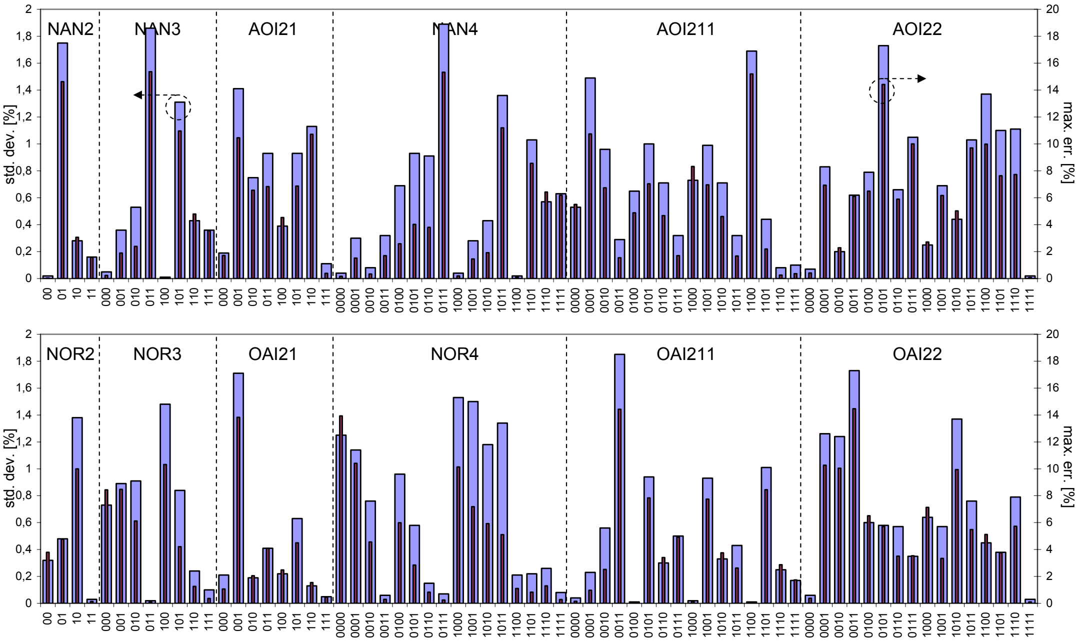


Figure 5.5: Standard deviation and maximum error for the 45*nm* high-k metal gate predictive technology using scaled 65*nm* circuits. For all technology sizes, the PTM devices are much easier to model that real industry devices.

NMOS subthreshold leakage is represented by a linear combination of the NC and the NS reference, and the gate tunnelling is described by a large N0 contribution minus a smaller N1 contribution. As the regression optimises for the lowest error and not the highest interpretability of the parameters, the values do not completely behave as expected.

For this one example, the NAND gate with 01 input, I compared the accuracy between the regression result and the *meaningful* values, by manually resetting the NAND2@01 parameters to $NC = 344.5nm$ (only the upper NMOS locks), $N0 = 344.5nm/2$ (only gate-drain tunnelling expected), $N1 = 366nm/2$ (only gate-source tunnelling expected), $NS = 0$ (the lower NMOS should only tunnel and the upper NMOS is nearly in typical NC condition), $PC = 0$ (subthreshold leakage of the locking PMOS is short-circuited by the open one), $P0 = 468.2nm$ (the conducting PMOS is in typical P0 condition), $P1 = 0$ (all terminals of the locking PMOS are at V_{DD}), and $PS = 0$ (no stack in PMOS). In contrast to the model obtained by regression, which had a standard deviation of 4.49% and a maximum error of 32.55%, this meaningful, but experimental model still achieved a standard deviation of 24.6% and 182.3% maximum error. Both models in this example, finally used a scaling based gate description, and both models have an acceptable accuracy. Nevertheless, developing the scaling parameters by regression, as proposed in this thesis, leads to slightly better estimation results, than developing such models directly from layout information.

Gate	A	B	NC	N0	N1	NS	PC	P0	P1	PS
			[nm]	[nm]	[nm]	[nm]	[nm]	[nm]	[nm]	[nm]
NAN2	0	0	0	183	0	345	0	1075	57	1
	0	1	123	1787	-462	366	-37	-74k6	10k4	1836
	1	0	321	2058	-271	-1482	18	-10k4	-41k3	6967
	1	1	-56	4455	-722	-3929	963	-11k2	-118k	16k6
physical			A=344.5nm B=374.5nm				A=468.2nm B=508.9nm			
NOR2	0	0	604	4870	-1466	-2890	39	-66k6	-57k2	13k9
	0	1	-91	6013	-1631	-4975	465	-27k7	-138k	21k6
	1	0	51	2614	-333	-1320	37	-134k	41k6	6444
	1	1	0	1	711	0	0	-428	558	501
physical			A=344.5nm B=374.5nm				A=468.2nm B=508.9nm			

Table 5.5: Examples for the effective widths resulting from the linear regression. All values are obtained for a 65nm technology. Due to IP protection, all values had to be scaled. All conclusions drawn here are also valid for the confidential industrial values. The results are discussed in Section 5.3.2

Conclusion

The gate modelling approach is very accurate and the resulting models are very simple, as eight float values per gate and per state are enough to describe each gate's complex dependence on the *StPa* and *DyPa*.

The result of the evaluation, especially the maximum error strictly depends on the range of the input parameters. The higher the model range (the quotient between largest and smallest value to be modelled), the higher are standard deviation and maximum error.

In an older version with just four references, for nearly each gate the effective widths were meaningful (interpretable). With eight references, meaningful parameters perform acceptable, but the regression usually finds strange (even negative) effective widths, which can better explain the gate's behaviour.

5.3.3 RT component models

The transformation from a gate model to an RT hard model is mathematically exact, thus the error of the RT hard macros can be assumed to be in the order of single gates. In principle, the relative standard deviation should even be smaller, as the total component leakage sums up for each gate, but the absolute error of independent sources just rises with the square root of the number of gates. However, as the RT hard-macro is just a by-product on the way from gate to RT level, it is not especially evaluated here.

In order to evaluate the accuracy of the soft model, various RT component families such as rip-

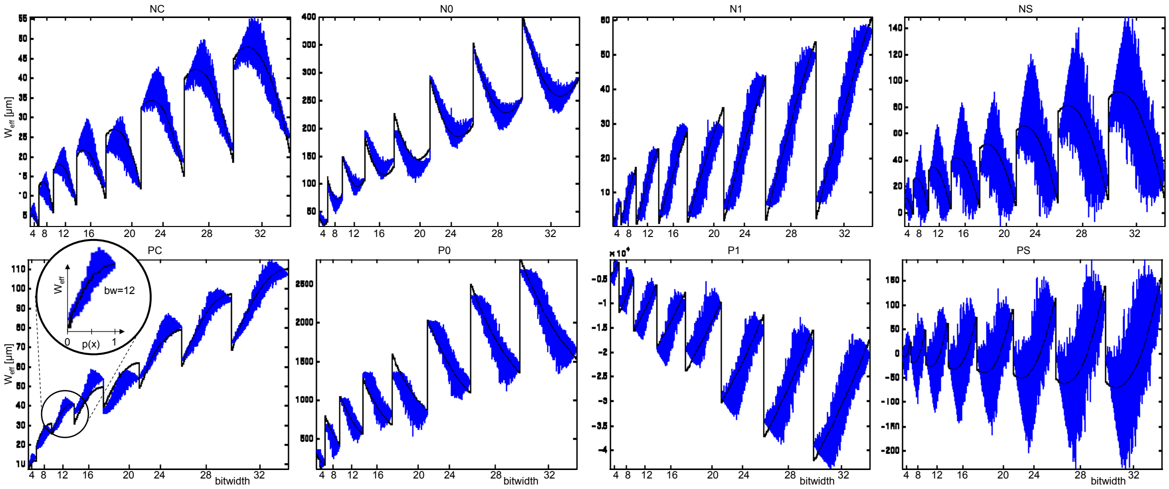


Figure 5.6: Ripple adder: Visualisation of the effective widths by data accurate, gate-wise simulation and by soft modelling regarding the bitwidth and signal probability only. Within each bitwidth, the signal probability rises from left to right.

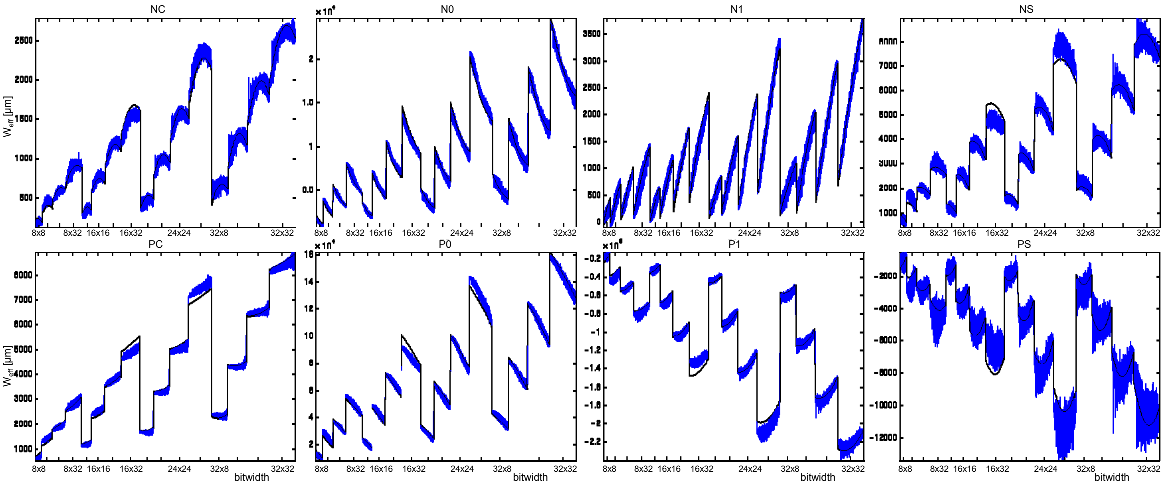


Figure 5.7: CSA Multiplier: Visualisation of the effective widths of simulation and model.

ple adders or look ahead decrementers, are synthesised, using Synopsys DesignVision and a 90nm technology, and simulated with a huge number of stimuli for each possible input probability. The evaluation of the 90nm technology only is sufficient here, as the abstraction steps from gate level to soft RT models are technology independent. For each case, a hard macro is generated. Now, the effective widths (i.e. the scaling parameters of the reference transistors) of all hard macros of one component family are described as a function of the components bitwidths and the signal probability according to the RT soft model, described in Section 2.5.2. Using these predicted widths, the total leakage can be computed and compared to the hard models. Table 5.6 lists the standard deviations of the overall leakage estimation and the computation of each effective width. Figures 5.6 and 5.7 visualise the errors in effective width for two example components. Figure 5.8 then tries to visualise the resulting error by plotting the model prediction over the simulation result.

Several observations can be made:

- The look ahead adder (AddCla) and incrementer (IncCla) show a much higher error than the other components in Table 5.6. Such an observation has already been made by other researchers and also by me for earlier models (such as the dynamic power, area, or timing models) and is not a weakness of the leakage model. Instead, the high error is a result of the inconsistent synthesis results of the commercial synthesis tool. When computing relevant metrics as critical path, or total component area as a function of the component bitwidth, it can be observed, that the synthesis results show an unsteady, and sometimes even non-monotonic behaviour[12].
- Especially the P1 and PS scaling widths behave strangely for most components, having negative values and triple-digit percentaged errors. This can be explained as follows: In the technology analysed, PMOS subthreshold leakage is just halve as high as NMOS subthreshold leakage and PMOS gate leakage is more than 100 times smaller than NMOS gate leakage. For the PS, the stack of two locking transistors, boosted by the body effect additionally reduces the leakage by more than a factor 20 in PMOS (10 in NMOS). Thus the resulting effective widths of P1 or PS are much more sensible to small numerical disturbances and thus much harder to predict (see in Figure 5.6 and 5.7, that the deviation of the widths within one class is significantly larger).
- For the overall leakage, the resulting error measures are extremely good, compared for instance with the dynamic power model. For the dynamic model (where I used to work at before doing leakage estimation), typical standard deviations are between 15 and 25% (over 35% for CLA components).

5.3.4 RT level evaluation against SPICE

The sections above presented a seamless evaluation chain from SPICE simulations to entire RT components. Using the law of Gaussian error distribution, the error for entire RT components

[12] For instance it can be observed, that a 43 bit CLA adder, optimised for critical path is slower and even larger than the 44 bit component. As a 44 bit component can perfectly fulfil all functions of the 43bit component, this non-monotonic behaviour is a contradictory artefact of the synthesiser.

Component	$\sigma(I)$	$\sigma(NC)$	$\sigma(N0)$	$\sigma(N1)$	$\sigma(NS)$	$\sigma(PC)$	$\sigma(P0)$	$\sigma(P1)$	$\sigma(PS)$
AddCla	21.12%	27.7%	19.5%	47.4%	33.5%	21.0%	25.4%	-28.8%	-43.9%
AddRpl	5.01%	8.4%	5.8%	13.3%	29.1%	5.8%	7.9%	-11.0%	-694.1%
DecCla	9.69%	10.4%	10.9%	-12.3%	10.6%	9.1%	10.7%	-10.5%	-10.6%
IncCla	12.24%	15.6%	9.3%	-14.9%	14.5%	11.6%	11.8%	-13.3%	-17.6%
MultCsa	3.79%	5.1%	3.9%	7.4%	6.9%	4.0%	4.3%	-5.1%	-13.5%
SubCla	6.05%	10.0%	8.2%	34.0%	12.7%	6.7%	9.8%	-11.0%	-15.9%
SubRpl	5.03%	9.8%	7.5%	22.1%	66.2%	5.6%	9.1%	-15.3%	259.1%

Table 5.6: Standard deviation of the RT soft model for a $90nm$ technology: $\sigma(I)$ is the standard deviation of the RT soft model against the explicit gate-wise simulation. The other eight σ's give the standard deviation of each single reference width computation. As the scaling parameters of the most relevant references (like NC) are more accurate, than for the minor references (like P1), the resulting error in total leakage is better than the average width estimation error.

against SPICE can be predicted as follows:

$$I_{RT} = \sum_{i=1}^{8} \alpha_i I_i \tag{5.3}$$

$$\Rightarrow \sigma_{IRT} = \sqrt{\sum_{j=1}^{8} \left(\left(\sigma_{\alpha j} \frac{\partial}{\partial \alpha_j} \sum_{i=1}^{8} \alpha_i I_i \right)^2 + \left(\sigma_{Ij} \frac{\partial}{\partial I_j} \sum_{i=1}^{8} \alpha_i I_i \right)^2 \right)} \tag{5.4}$$

$$= \sqrt{\sum_{j=1}^{8} \left((\sigma_{\alpha j} I_j)^2 + (\alpha_j \sigma_{Ij})^2 \right)} \tag{5.5}$$

where α_i is the RT level scaling parameter for reference transistor i; I_i is the respective reference current and σ the standard deviation. Using the respective values for the transistor and RT level error, the standard deviation of the entire model can be expected to be at 5.0% in $90nm$, 6.9% in $65nm$, and 3.6% in $45nm$[13].

Nevertheless, an evaluation of the model at its highest level, directly against SPICE simulations will further increase the confidence in its correctness. A single computation of an RT component of significant size can easily need hours, large components can even completely fail to converge in SPICE. A reliable simulation of global and local process variation under the influence of different *DyPa* sets will require thousands or even millions of single SPICE simulations.

In order to cope with the extreme numerical complexity, the full evaluation of the RT model against SPICE is reduced in order to at least enable some single point evaluations: A very small component, the 8 bit ripple adder is chosen for evaluation, as it can be simulated comparatively fast in SPICE, and just the $45nm$ PTM technology is analysed. The *DyPa* are limited to 125 different settings including all extreme cases of highest and lowest temperature, supply, and body voltage. For each of

[13] computed for the AddRpl component

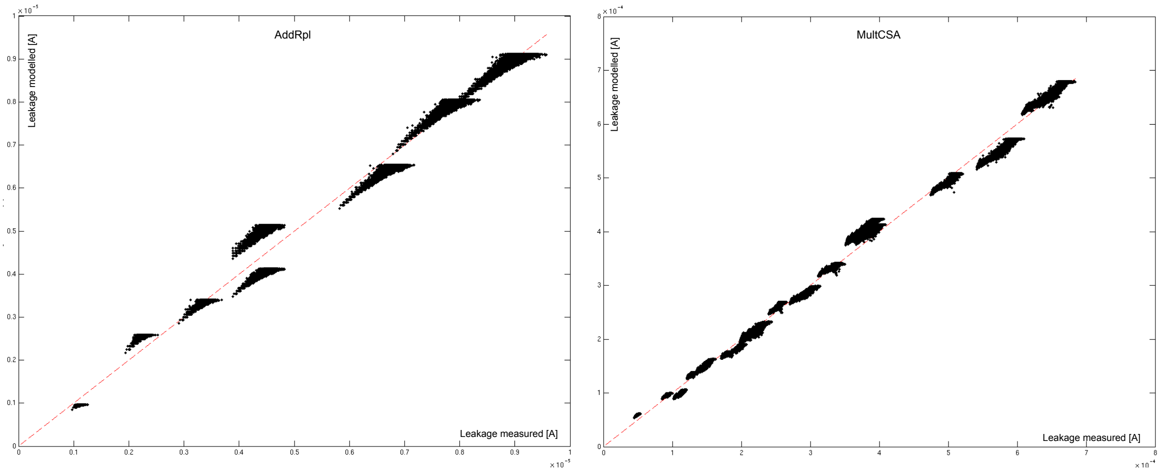


Figure 5.8: **Left:** Ripple adder **Right:** CSA Multiplier leakage as a result of the soft macro over leakage that is obtained by transistor level simulation

these settings, only the *typical* process corner, where the mean of all *StPa* is at their nominal values is analysed. Each single process parameter follows a local Gaussian distribution. The results are visualised in Figure 5.9. The deviation at all sets show, that the total error is never far above 10% (worst case is for maximum temperature, minimal supply and maximum forward biasing).

Then, only under nominal supply, average temperature and zero body biasing, the analysis is repeated for over 100 different settings of global parameter variation. Again, each transistor underlies its individual Gaussian local variation around the mean. The results are visualised in Figure 5.10. Worst single errors here range up to 25% (worst case for lowest channel doping and smallest oxide thickness for small, but not smallest channel lengths).

Remark:

The statement, that RT components evaluate in hours and even do not converge was true for the resources I had available in 2007 when I finished the research for this work. Nowadays, using recent HSPICE versions, it is easily possible, to do single point estimation of average sized components (thousands of transistors) in minutes. Thus, a full evaluation of my models against SPICE would be possible today. The first analyses leading to this work were made in Berkeley SPICE, being freely available, and then Berkeley SPICE was tightly integrated into the entire characterisation flow.

The missing direct evaluation (short of the punctual comparison presented above) is in my opinion a flaw of this work. But as all characterisation methodology is tightly integrated to Berkeley SPICE, the cost for a full evaluation is very high. Additionally, even the worst case error for completely correlated error sources, resulting in a sum of all errors of all model stages will be surely below 20 percent error, and thus still acceptably low for the later model purpose of system level power prediction.

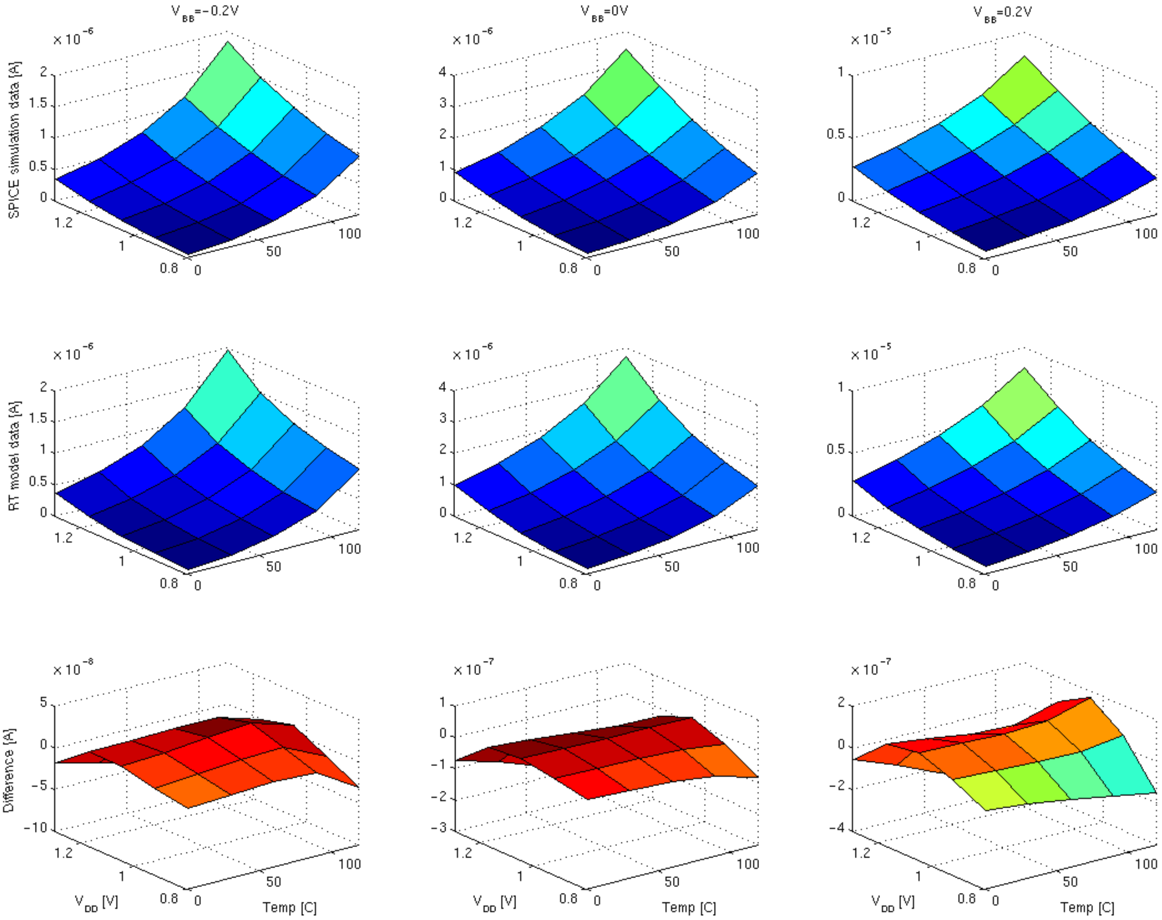


Figure 5.9: Visualisation of the total leakage of an 8 bit ripple adder with all inputs 0. *StPa* are at their nominal values, each diagram shows leakage with respect to supply voltage and temperature. From left to right column, the body bias rises. Top row shows SPICE simulation result, middle row shows the model output and bottom row shows the difference between SPICE and model.For $\vartheta < 40°C$ and $V_{\mathrm{BB}} < -150mV$, the model has to extrapolate.

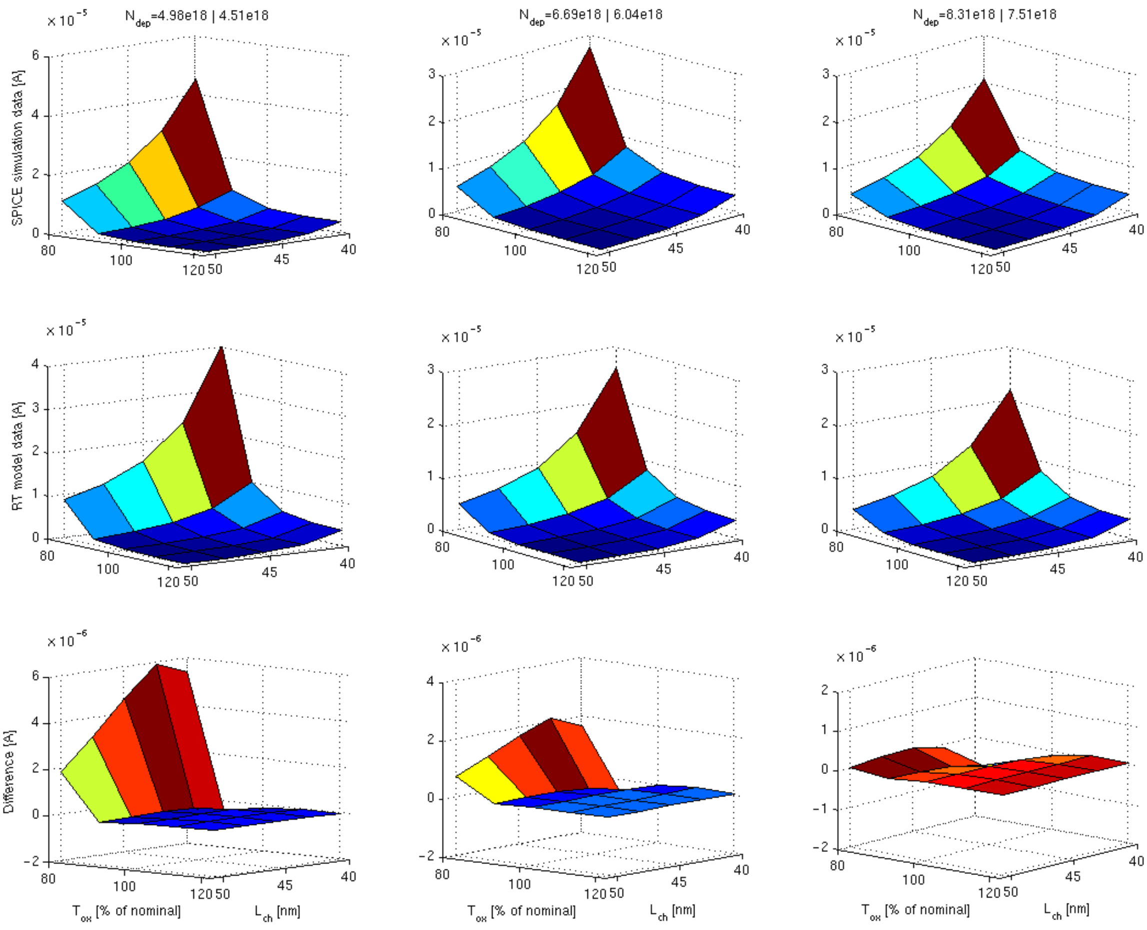


Figure 5.10: Visualisation of the total leakage of the 8 bit ripple adder with all inputs 0. *DyPa* are at their nominal values, each diagram shows leakage with respect to oxide thickness and channel length. From left to right column, the channel doping rises. Top row shows SPICE simulation result, middle row shows the model output and bottom row shows the difference between SPICE and model.

5.3.5 Thermal model

In Figure 5.11, all three solutions, described in Section 3.3, the iterative, the plain superposition, and the superposition with reflection are visualised and compared. To simplify the visualisation, only a one-dimensional cross-section through the active layer is shown.

As can be seen, the results for the finite elements method and the superposition method are almost identical. This evaluation was made for a system exactly fulfilling the symmetries, needed by the superposition approach (single active layer, system consists of homogeneous layers in x-y direction, ambient temperature does not change, no time dependency of the thermal profile, ...).

A system with $1600 \cdot 1600$ finite elements in x-y direction and ten elements in z direction needed 80 hours computation time in Matlab on a 2x3GHz x86 system. The pre-computation of a singular solution requires 3.2 minutes (when exploiting the circular symmetry) and has to be done just once for a given chip geometry. The superposition of the final solution needed 2.4 minutes and the reflection at the borders less than a second.

The superposition approach can compute thermal profiles of systems under certain symmetries nearly 1000 times faster than a finite elements approach. With the finite elements, the effect of electro-thermal coupling can be easily regarded by simply updating the power density after each iteration step. For the superposition approach, a static power density is mandatory. Thus, electro-thermal coupling has to be regarded after thermal mapping by an outer iteration (leakage is computed at ambient temperature, then temperature is computed with this leakage power, then leakage power is updated with respect to the recent temperature, and so on).

As presented in Section 3.3.1, for typical systems such an iteration needs three to six steps, which leaves a speed-up of more than a factor 200[14] for the superposition approach regarding coupling. The main drawback of this thermal model does not directly arise from the model itself, but from the fact, that PowerOpt computes complex systems function-wise, and not once for the entire system. While the energy reflection assumption leads to a nearly perfect match between iterative and superposition approach at borders; between tasks with extremely different average energy density, it leads to local errors, which need to be regarded.

Extremely different average energy density among different parts (functions) of a system typically results from differences in the workload over time or from temporary idleness. Thus, in Figure 5.12, a system consisting of three functions (each consisting of several RT components) is analysed. One of the functions (with the lowest x-coordinates), is in a low activity mode, the function with the highest x coordinates is completely idle, and the centre one is at its maximum activity.

Again, only a one-dimensional cross-section through the active layer at $y = 1000nm$ is shown. As inherently required by the PowerOpt tool, the thermal modelling needs to be performed per system part and merged together afterwards, resulting in an estimation error at the task borders. For evaluation, the finite elements approach was applied to the entire system. In the system evaluated, the maximum error was $10.3K$, the standard deviation $1.74K$ and the average temperature was $0.36K$

[14] only the superposition has to be iterated, and the singular solution can be reused

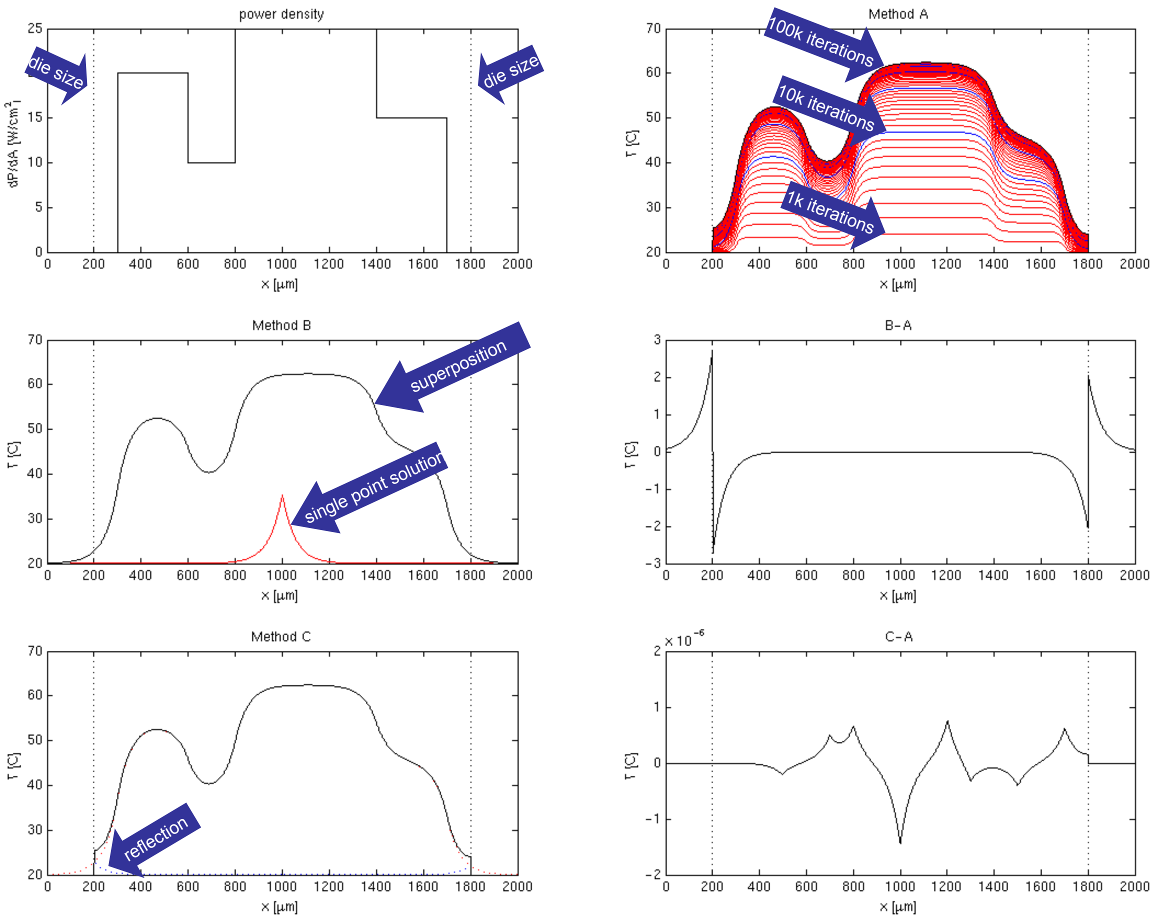


Figure 5.11: Evaluation of the thermal model. Under the power density distribution (top left), a finite elements approach (Method A) converges to a solution (top right). Each red line represents the result after 1000 more iterations, each blue line after 10000 iterations. After 100000 iterations, the computation converged (top black line). The middle left figure shows the superposition method (Method B). A solution for a singular heat source is found (here: power for a $1(\mu m)^2$ die area, magnified x 1000 in temperature for visualisation). The thermal distribution can be super-composed from this solution. The middle right figure shows, that huge computation errors occur, as Method A assumes no energy transfer at the borders and Method B assumes an infinite die size. The bottom right figure shows the correction: The thermal distribution into the virtual die area is folded (reflected) back into the die. The difference (bottom right) to the accurate Method A now is marginal.

higher for the superposition than for the finite elements approach.

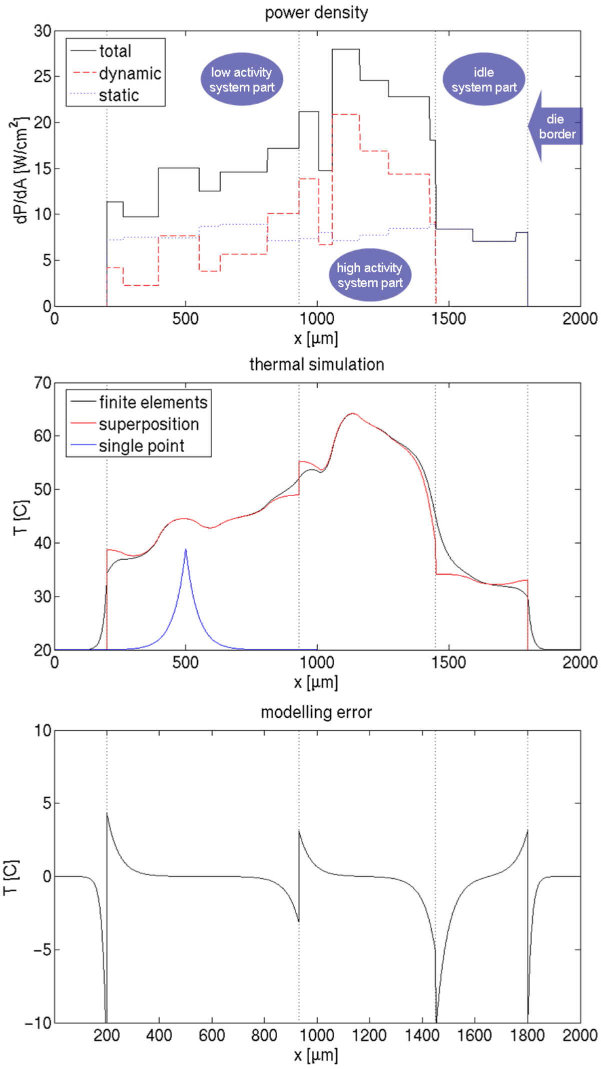


Figure 5.12: Evaluation of the thermal model for a cross-section through the die. The superposition approach composes task wise solutions based on a single point solution. At the task borders, the temperature prediction diverges from the accurate approach.

5.3.6 IR-drop model

In order to evaluate the IR-drop model, a single standard cell row in $65nm$ with $1V$ nominal V_{DD} is completely simulated using SPICE. Figure 5.15 (left) shows the drain to drain voltage for eight gates equidistantly chosen from the row. Due to the RC properties of the supply grid and the CMOS gates, each voltage shows a high frequency oscillation.

If the total power, averaged over time, is lumped into equivalent current sources, all noise effects

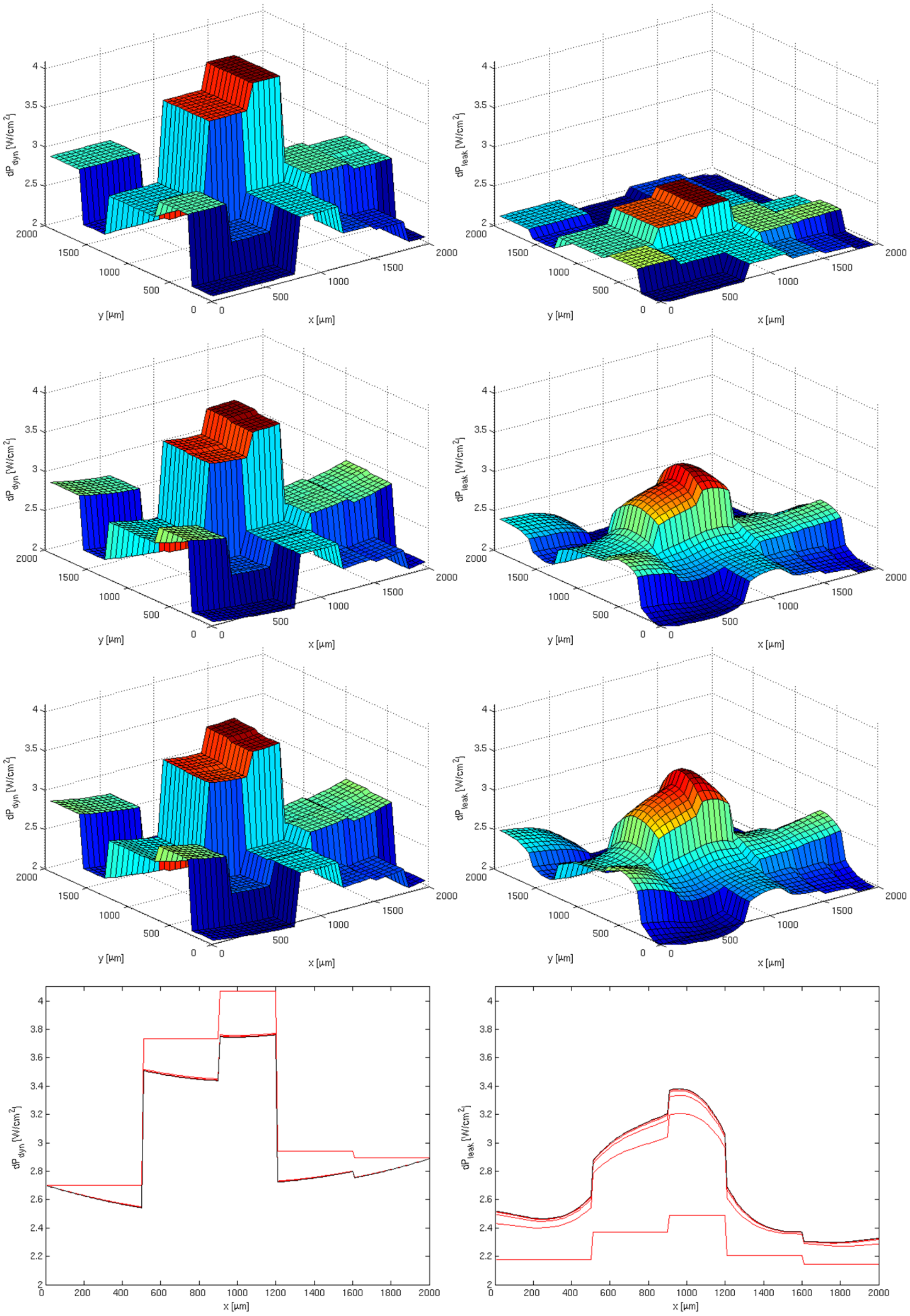


Figure 5.13: Dynamic (left) and static (right) power maps. First row: before temperature and voltage mapping. Second row: After first temperature and voltage mapping. Third row: Final result after ten iterations. Fourth row: All iterations for $y = 1000\mu m$.

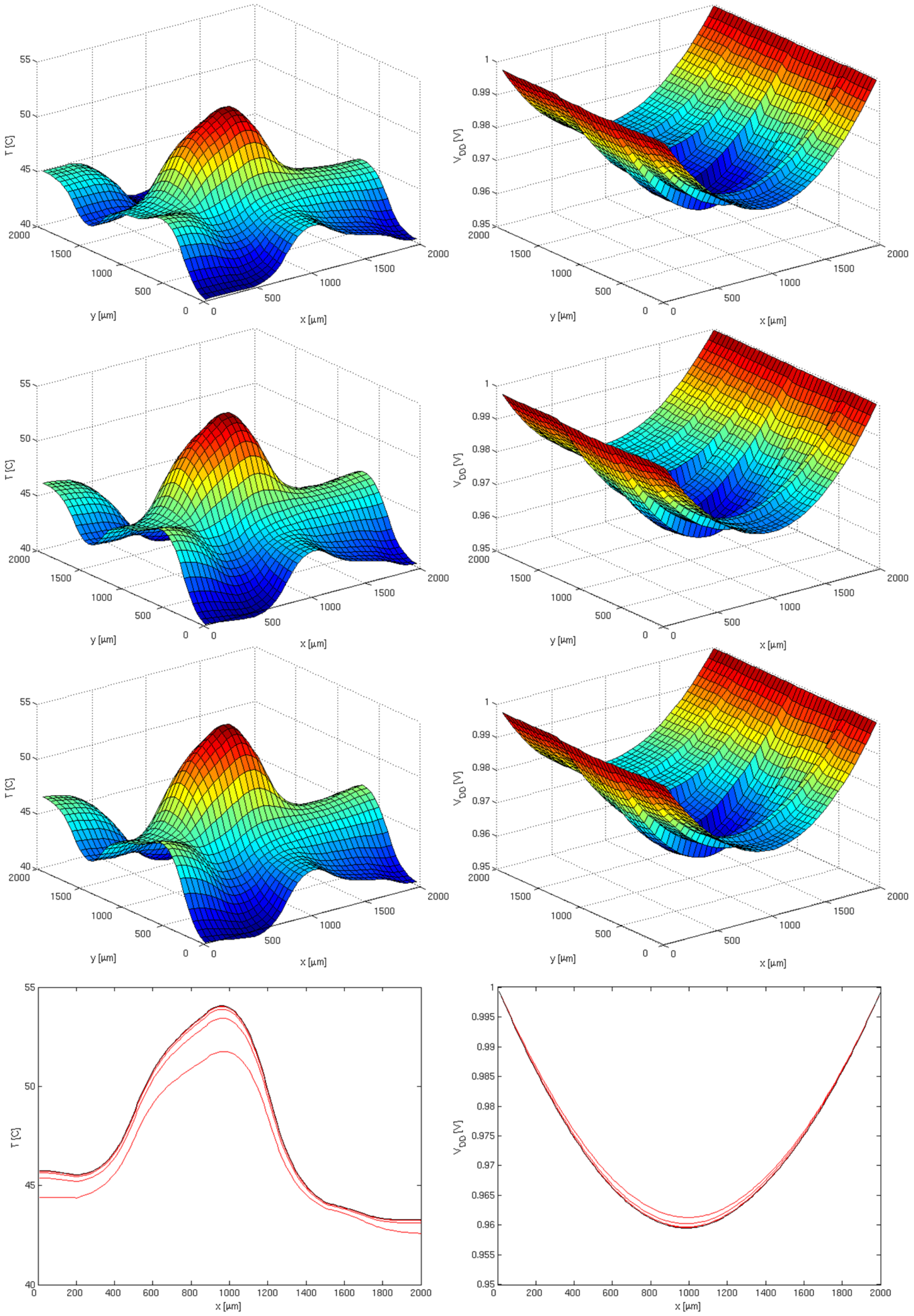


Figure 5.14: Temperature (left) and voltage (right) maps. First row: Initial computation. Second row: After first iteration of the power maps. Third row: Final result after ten iterations. Fourth row: All iterations for $y = 1000\mu m$.

except for the IR-drop are averaged out. Depending on how exact the average per gate power is estimated, the supply voltage of this equivalent circuit can equal the average voltage on the left side up to numerical precession, Figure 5.15 (right).

For this equivalent circuit, a closed form solution exists, as described in Section 3.4, exactly solving the equivalent circuit in $\mathcal{O}(n)$ time, where n is the number of voltage sampling points in the system.

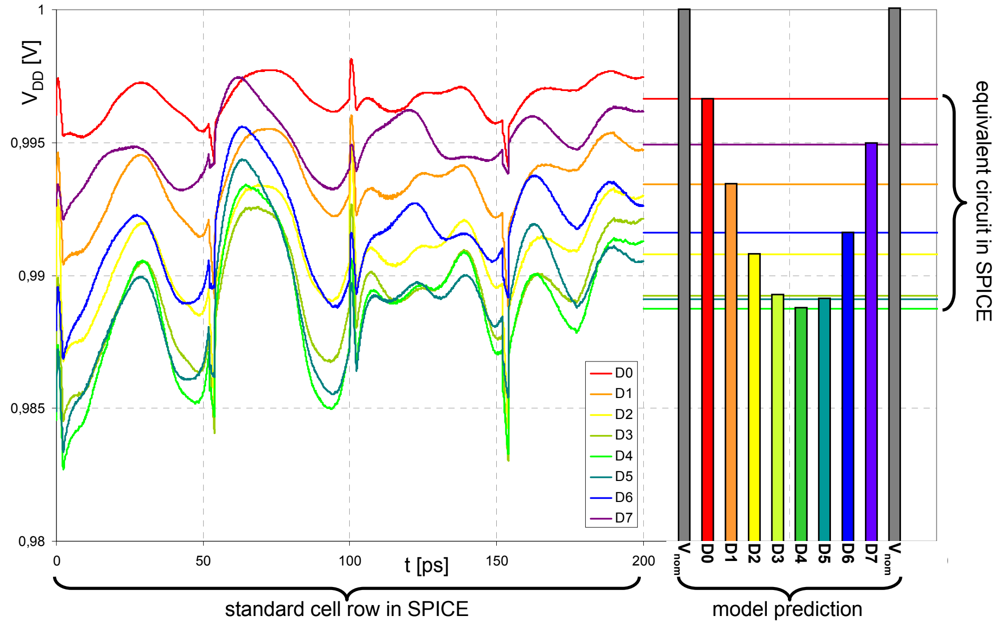


Figure 5.15: Evaluation of the IR-drop model. **Left:** A real standard cell row shows high frequency oscillations besides the IR drop. **Right:** An equivalent circuit in SPICE just shows the IR drop behaviour, averaging over other oscillations. The analytical model perfectly describes this equivalent circuit.

Figures 5.13 and 5.14 visualise the behaviour of the sample based combined voltage and temperature mapping. The system bases on a generic floorplan of size $2 \cdot 2mm^2$. The initial dynamic and leakage power for each RT component is computed at $1V$ supply voltage and $300K \approx 27°C$ temperature. Based on this, the total power density is sampled each $10\mu m$ in x and y direction (Figure 5.13, first row). From the power map, the initial temperature and IR-drop map can be computed (Figure 5.14, first row). Now, the power mapping can be repeated: For each $10 \cdot 10\mu m^2$ sample, the dynamic and leakage power models are evaluated. For visualisation purposes, not the expectation value of the power per component, but its individual per sample contribution is displayed (Figure 5.13, second row) and reiterated in the temperature and voltage model (Figure 5.14, first row). The third row of both figures shows the final result after ten iterations[15], and the last row shows the convergence at a cross-cut at $y = 1000\mu m$.

[15] In a working, not visualisation mode, only four iterations would have been necessary, ten is chosen to show, that the computation really converged.

6 Conclusion

In only five years, leakage developed from an academic corner phenomenon to a central problem of embedded system design. In sub $90nm$ designs, the leakage power is already exceeding the dynamic power. Today, leakage currents can only be kept under control, because they are regarded as a top priority criterion in technology design. For recent EDA tools, the process, temperature and voltage variations, as well as the electro-thermal and electro-electrical back coupling to temperature and supply are the key problems regarding both, power and performance. Today, the design of systems under variation still relies on corner cases and back-coupling is regarded by worst case assumptions. The availability of a fast and early model holistically regarding static and dynamic power, performance and back-coupling under variation and under the influence of optimisation techniques for leakage and/or variability will significantly improve the entire design for variability process.

This thesis contains a complete framework, integrated into the PowerOpt synthesis tool, that enables early leakage prediction already at the specification level, accurately regarding the effect of PTV variations, as well as the correlation between power and performance. Before the first synthesis steps, the system's behaviour in terms of power and performance and thus the expected parametric yield can be predicted, and various design techniques improving behaviour and yield can be tested and traded off. Until now, several promising design optimisations, especially those that have to be applied at high level, but implemented at low level of abstraction can hardly be applied due to a missing EDA support. I think that my contribution here is one further step towards the perfect design for variability.

Even though most of the research is from 2003, in literature, there is even today no other high level approach reported, that is as accurate or complete as this one. Best to my knowledge, all relevant impacts are regarded. The entire model development is blackboxable and bases only on industry standards such as the BSIM compact model for the technology information and the Liberty library (.lib) format for the circuit information. Process variation, which has no standardised description, enters as a Monte Carlo list. The high degree of automatisation and the use of industry standards will simplify the integration of the leakage estimation engine into existing design flows.

Due to the modelling layers, new technologies should be easy to integrate. Even for the radically new technologies, expected to coming up with the $32nm$ node, the ultra thin body devices and the FinFETs, only the transistor model layer will have to be updated or replaced. The model prediction for each entire RT component is around 10% standard deviation against SPICE simulations, and a model evaluation of the final model just needs some hundred operations and can thus be performed in less than a microsecond. In contrast to the accuracy, which finally was much higher than the minimum demand of a macro model, the high model evaluation performance is absolutely necessary, as the PowerOpt tool iteratively compares thousands of design alternatives and needs to perform up to millions of single model evaluations per design alternative.

Modelling leakage currents was a completely new, and very interesting challenge. The way from data dependent modelling, which is still a standard description (e.g. in the Liberty library format, or the PowerOpt dynamic power models) towards a physical property modelling needed completely new ideas and methodologies, but it also offered completely new opportunities: The parametric yield was integrated very naturally, as a result of the per component variation of power and performance, and a maximum constraint for the entire system; and the distribution maps of relevant system metrics as temperature, voltage level, and current flow will ease up a possible integration of new, upcoming effects as reliability issues.

6.1 Outlook

The model as described here is the base of the entire system analysis methodology and is constantly updated and improved within several research projects. Recently, there are three major directions, in which the model is beeing improved: Support of new technologies, increase of the model abstraction level, and integration of degradation effects.

This work is complete in terms of leakage estimation for bulk CMOS devices from $90nm$ down to $45nm$. The $32nm$ node is expected to introduce a series of significant changes. Instead of bulk CMOS, these devices may be implemented as fully depleted ultra thin body devices or with vertical channels as FinFETs. Both implementations will definitely have high dielectricity oxides and are both implemented as silicon on insulator. Gate and junction leakage will be significantly reduced, potentially leading to much simpler transistor level models. But in contrast, while approaching molecular dimensions and exceeding the resolution of light by nearly an order of magnitude, these devices will be subjected to completely new problems, especially granularity limitations (e.g. line edge roughness or random dopant effect), high range optical correlation (introducing a new class of parameter correlations), a high parameter degradation rate, and a high sensibility to outer influences (as single event upsets). I expect only minor changes for the leakage model itself, but the parameter determination framework (especially the variation engine) will have to integrate a number of new effects. Additionally, with the rising importance of local variation, a completely new delay model including a statistical statical timing analysis will be unavoidable.

Additionally, this framework will be augmented by a tile-based model for large components, where process and temperature gradients cannot be ignored. This will not just be beneficial for the model accuracy, but will also enable the leakage estimation of much larger systems as processors, IP components and memories. This will help to leverage the abstraction layer to a component communication centric one, called bus-cycle accurate basic blocks (B^3). Each algorithmic basic block, just interrupted by control statements and external communication, has to be modelled by a single leakage power model evaluation. Leakage models will have to be generated for dedicated hardware, as well as for software instructions and for special IP such as memories. The challenge here is that these models need to be nearly as accurate as the available macro models, but cannot be much more detailed than *complexity based* models; they will have to be simple enough to enable estimation for entire platforms.

In order to integrate degradation effects, such as electro-migration, hot carrier injection, or negative bias instability to the framework, a flow will be developed, in which the reliability is interpreted as a function of the system parameters in time. Functional yield is then simply the reliability at time $t = 0$. Based on the different operation conditions per instance due to varying system parameters, these parameters develop differently in time. The two top challenges are at first the detailed understanding and modelling of the degradation mechanisms at transistor level, and then the description of the degradation, that influences the system in a timescale of months, but is influenced by the system behaviour, which has to be simulated in a timescale of nanoseconds.

A Market overview

Besides the Synopsys simulators, clearly dominating the market (see Table A.1), also other major EDA vendors offer their SPICE equivalent and performance improved fastSPICE versions:

- **Cadence Virtuoso:** Cadence offers two transistor simulators, SPECTRE, a SPICE version and UltraSim, a fastSPICE derivative. Both tools support BSIM3, BSIM4 and PSP compact models.
- **Mentor Graphics ADVance MS:** The accurate (SPICE like) analogue transistor simulator from Mentor Graphics is called Eldo, and their fastSPICE tool is called Mach TA. Both tools read in standard SPICE netlists.
- **Magma FineSim:** The simulators, developed by Magma are called *FineSim SPICE* (SPICE like), and *FineSim Pro* (fastSPICE). Both FineSim tools read SPICE syntax and support the BSIM3, BSIM4, and Phillips MM9 (PSP predecessor) compact models. Magma distinguishes between SPICE and fastSPICE, and additionally hyperSPICE. HyperSPICE sacrifices more accuracy, for an even higher performance increase.
- **SimuCAD:** SimuCAD is a small EDA vendor, which developed a compact model, called HiSIM. HiSIM, being a surface-based model, has a higher simulation accuracy than threshold based BSIM compact models. Together with this compact model, SimuCAD offers the SPICE level tool *SmartSPICE.* SmartSPICE-HiSIM is the only commercially available surface-potential compact model, with exclusive analogue simulator, SimuCAD has a comparatively large market share for SPICE like simulators. Currently, SimuCAD offers no performance improved fastSPICE tool.

Vendor	SPICE like	market share	fastSPICE	market share	comp. model
Cadence	SPECTRE, PSPICE	40%	UltraSim	13%	BSIM,PSP
Magma	FineSim SPICE	NA	FineSim Pro	NA	BSIM
Mentor	ELDO	37%	Mach TA	5%	BSIM,PSP
Philips	pstar	in-house	NA	NA	PSP
SimuCAD	SmartSPICE	5%	-	-	HiSIM
Synopsys	HSPICE	58%	NanoSIM/HSIM	59%/56%	BSIM / PSP

Table A.1: Market shares [138] of SPICE like and improved performance tools.

B Modelcard conversion

BSIM3 to BSIM4 conversion

Non quasi static model: The split selectors *TRNQSMOD* and *ACNQSMOD* for transient and AC non quasi static modelling have to be set to the old value *NQSMOD*. The Elmore constant *ELM* and its binning has to be removed as not needed by BSIM4.

Flicker and thermal noise model: The model selectors *FNOIMOD* and *TNOIMOD* have to be recomputed. As some old models are no longer supported, the respective noise model is disabled in these cases and a user warning is generated.

Non uniform doping model: The values for non uniform doping concentration have to be escaped (set to zero values) to reflect a homogeneous channel doping concentration of *NCH* (channel doping parameter of the BSIM3 model).

Mobility model: From BSIM3 to BSIM4, a new mobility model was introduced (number 2 in BSIM4) and two old models are no longer supported (number 0 and 2 in BSIM3). Depending on the exact version of the BSIM file, the model identifier has to be adapted and a user warning is generated in case of an unsupported mobility model.

Capacitance model: Capacitance model 1 is changed from BSIM3 to BSIM4. The old parameters seem to work, but the user is warned to manually check correctness. The models 2 and 3 in BSIM3 can be accurately modelled by the BSIM4 capacitance models 1 and 2 respectively.

Upgrading BSIM4

Junction diode parameters: Several symmetrical parameters have to be duplicated to describe a symmetrical device in the (potentially) asymmetrical model introduced with version 4.5.0.

Capacitance model: As for the junction diode, all symmetrical parameters for source and drain capacitance have to be duplicated (Example: $CKAPPA \rightarrow CKAPPAS{=}CKAPPAD$).

Limiting diode current: The parameter *ijth* is now represented by four parameters for source, drain and for forward and reverse.

HSPICE-BSIM to BSIM4 conversion

Several parameters need to be slightly modified. For instance, *TREF* in HSPICE-BSIM (level 49 and level 53) has to be renamed to *TNOM* in BSIM4. Other parameters have to be made asymmetrically (for source and drain) (as $CJGATE \rightarrow CJSWS{=}CJSWD$).

Other parameters have to be removed. For instance all parameters controlling the HSPICE MOSFET binning process (*BINFLAG*, *LMIN*, *LMAX*, ...).

C Transistor model visualisation

The figures depicted in Annex C try to visualise the model errors at transistor level. As the transistor models are six dimensional functions, it is very hard to generate full visualisation of all aspects. Instead, for the subthreshold leakage references (NC,NS,PC,PS) the behaviour is visualised for the three dimensional subspace (V_{DD}, V_{BB}, ϑ) for nominal *StPa* and the three dimensional subspace (L_{ch},T_{ox},N_{dep}) for nominal *DyPa*. For the gate leakage references (N0,N1,P0,P1), the leakage is *nearly* independent of four of the six dimensions, thus, they are omitted for the sake of clarity[1]. Instead, Table C.1 lists the maximal deviations for the gate leakage references for the other four dimensions.

All visualisations for the subthreshold leakage references are organised as follows: In the upper halve (*DyPa*), each single diagram plots the current on the z-axis over the body voltage on the x-axis and the temperature on the y-axis. For the three columns of each diagram the supply voltage is continously increased. The first row shows simulation data from SPICE, the second row the model data, and the last row the absolute difference between both. The bottom halve (*StPa*) is structured accordingly. Here, the current on the z-axis is plotted over the oxide thickness on the x-axis and the channel length on the y-axis for increasing values of channel doping from left to right.

The visualisation of the gate leakage references is organised as follows: For each of the four references, the current on the z-axis is plotted over the supply voltage on the x-axis and the oxide thickness on the y-axis. Top diagram is SPICE simulation data, middle diagram is model data and bottom diagram is absolute difference between both. The deviation in all other four dimensions is reported in Table C.1.

As can be seen, gate leakage of locking transistors is perfectly independent of other input parameters than T_{ox} and V_{DD}, which are the two major parameters influencing the tunnel effect. All other minor dependencies (the gate leakage in the order of tens of nA for $90nm$, to several hundreds of nA for $45nm$) result from a second order effect, where electrons tunnel from the gate to the channel and flow from there to source or drain.

[1] Of course, these dimensions are only omitted for the visualisation, not for the modelling itself

Reference	N0	N1	P0	P1	N0	N1	P0	P1	N0	N1	P0	P1
Technology		$45nm$				$65nm$				$90nm$		
$\sigma(L_{ch})[pA]$	0	32320	34220	0	0	1308	0	0	0	850	0	0
$\sigma(N_{dep})[pA]$	0	60	50	0	0	10	0	0	0	10	0	0
$\sigma(\vartheta)[pA]$	0	6090	6900	0	0	234	0	0	0	173	0	0
$\sigma(V_{BB})[pA]$	0	800	3020	0	0	16	0	0	0	17	0	0

Table C.1: Most gate leakage reference transistors do not show a significant decency to other parameters but the oxide thickness and supply voltage. The table shows standard deviation of the SPICE simulation data when varying one parameter by ±10%. Even though these dimensions are regarded within the models themselves, they will be ignored for the following visualisation.

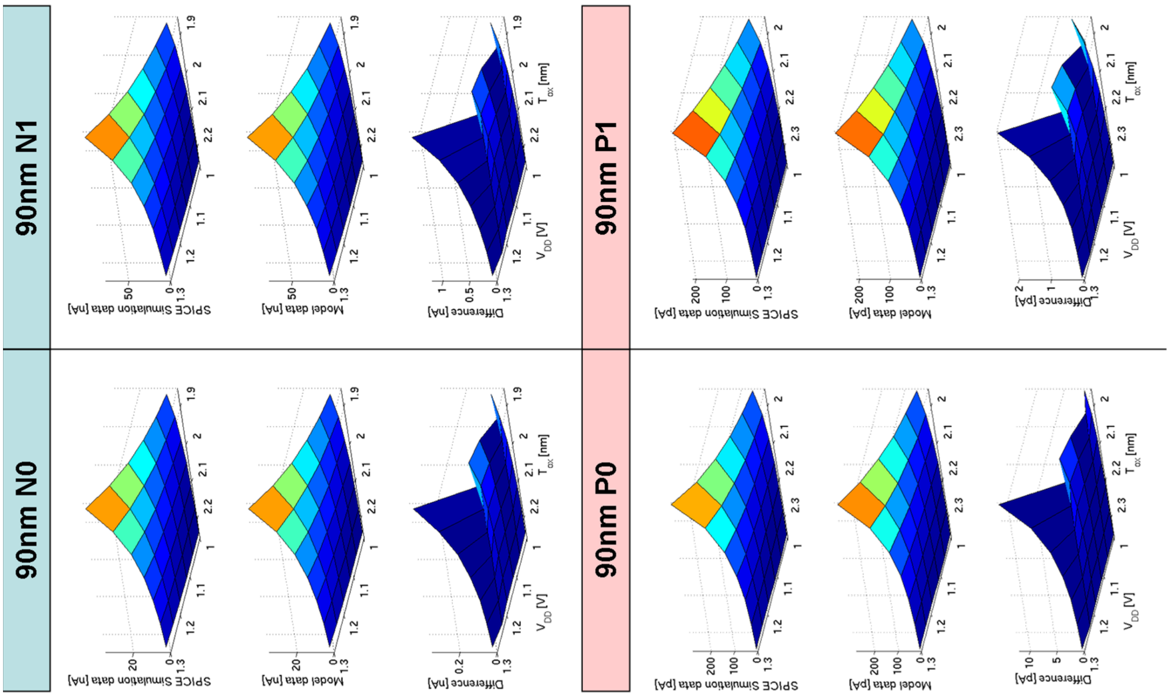


Figure C.1: All four gate leakage references in $90nm$.

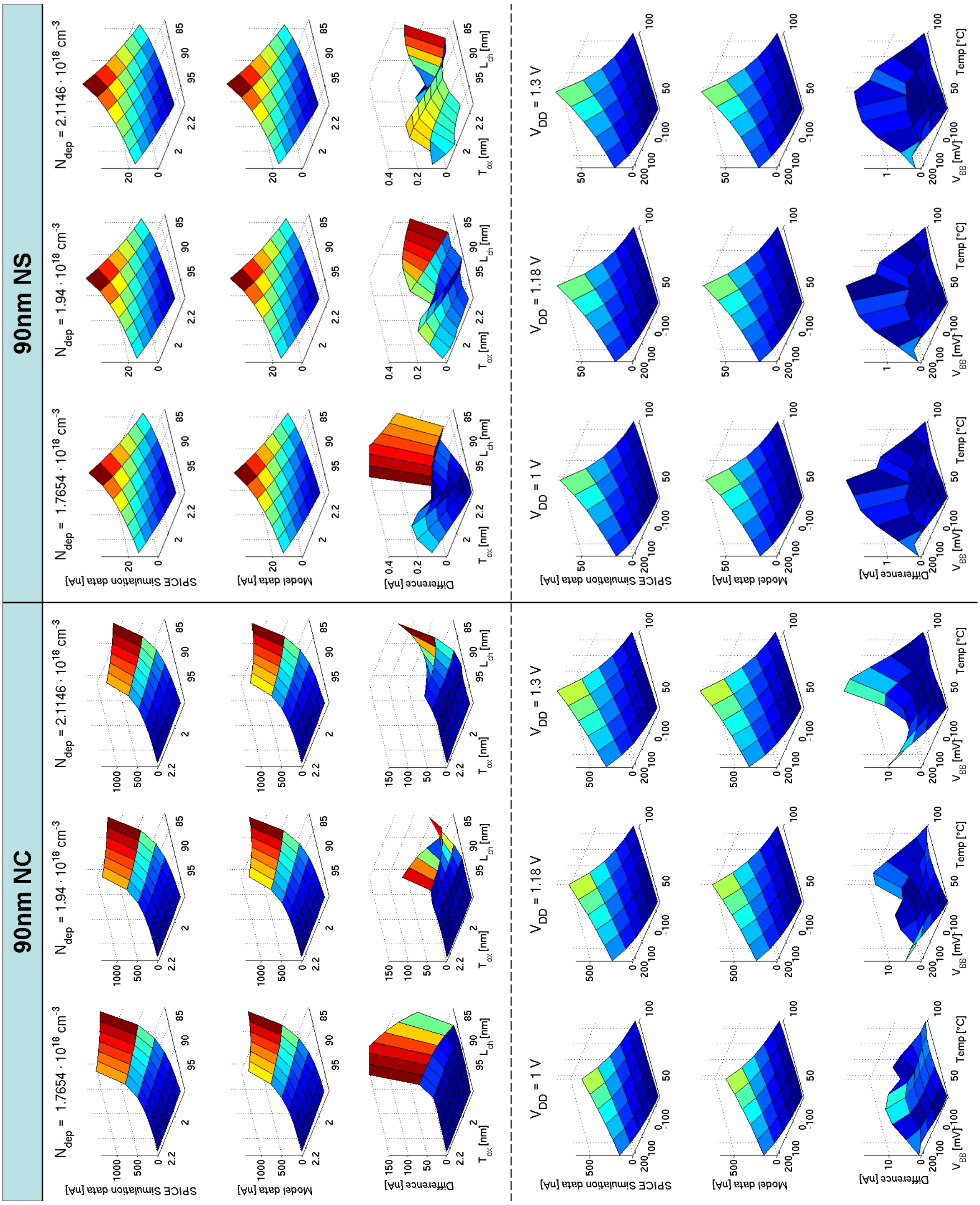


Figure C.2: NC and NS reference in $90nm$

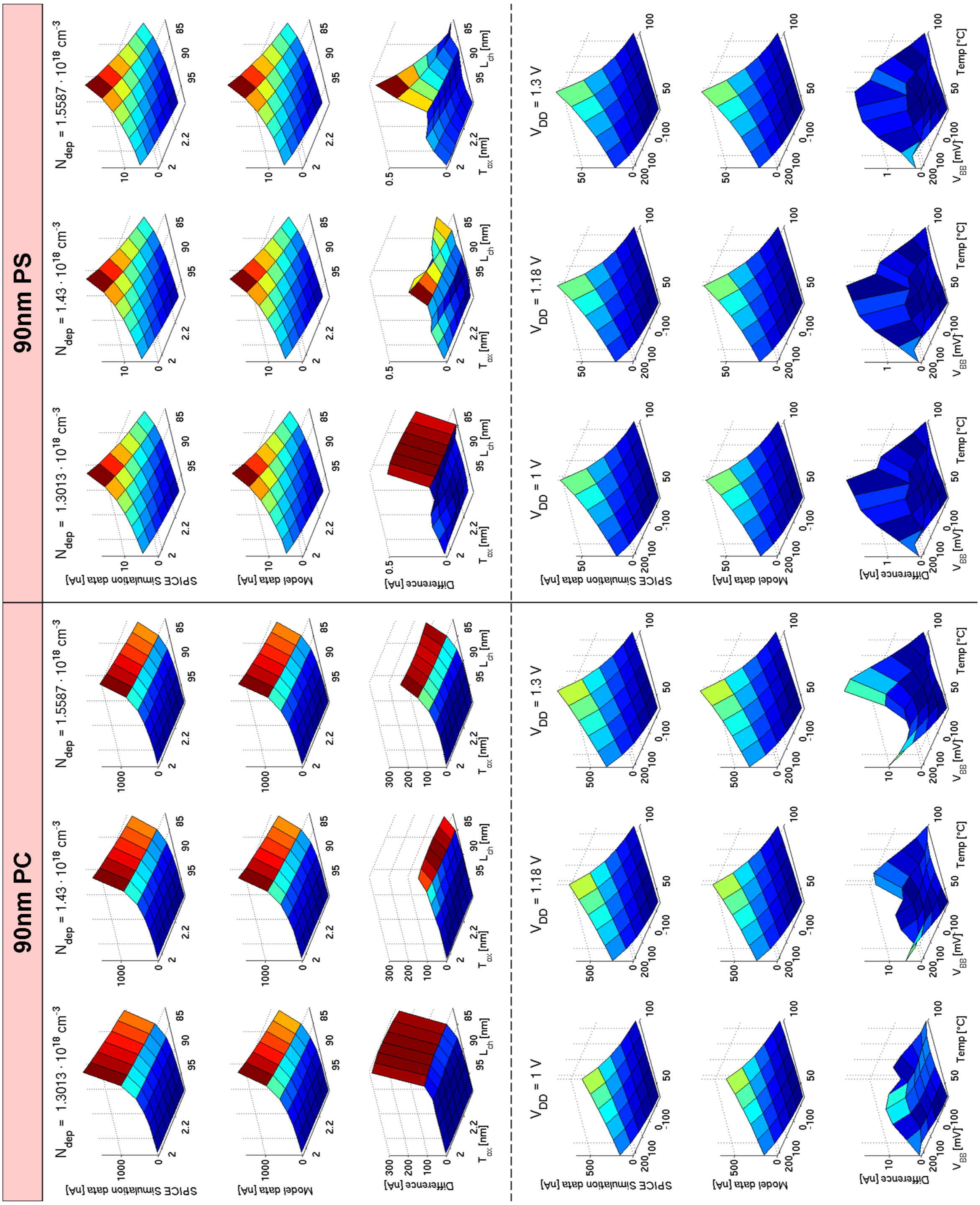


Figure C.3: PC and PS reference in 90nm

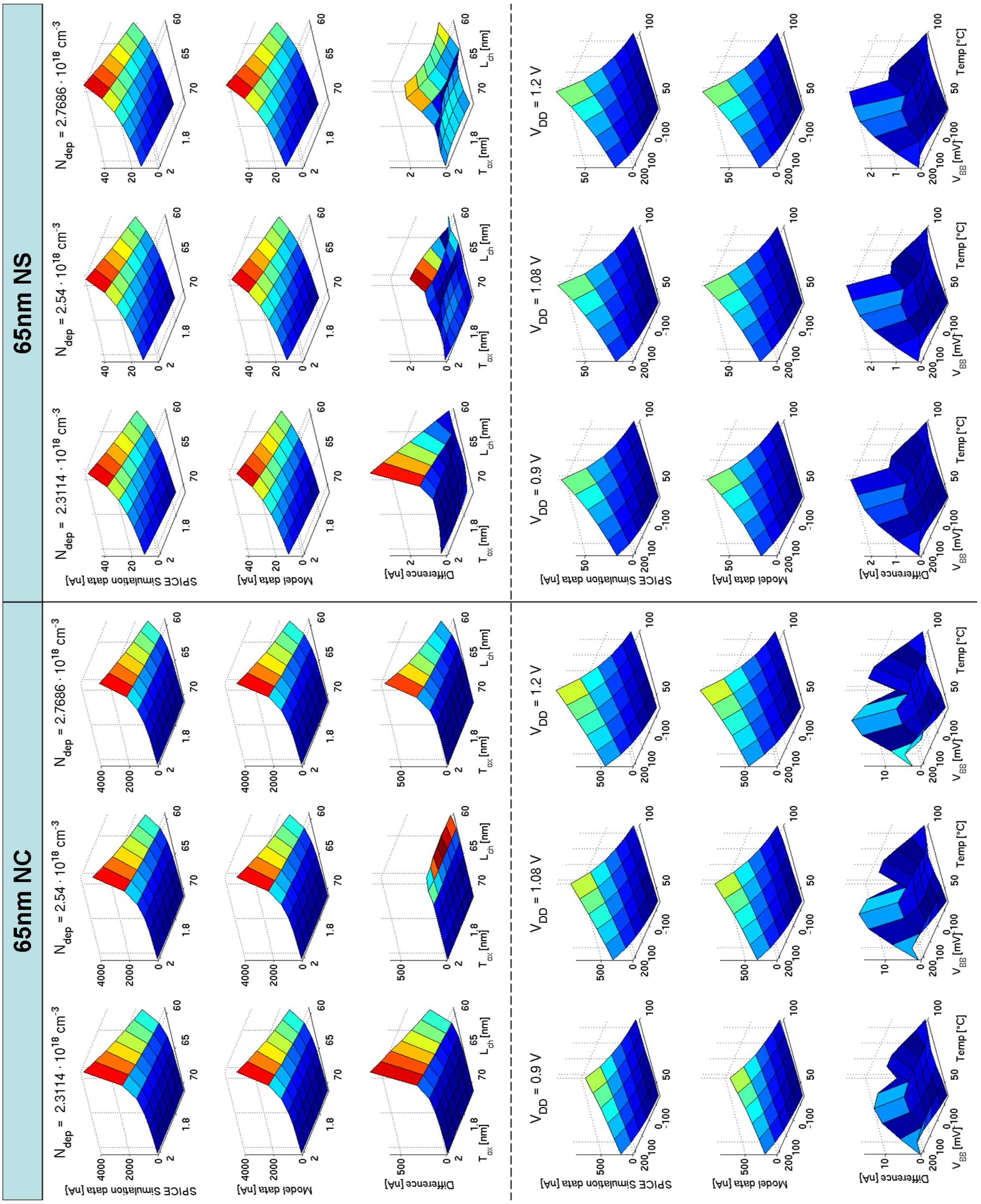


Figure C.4: NC and NS reference in $65nm$

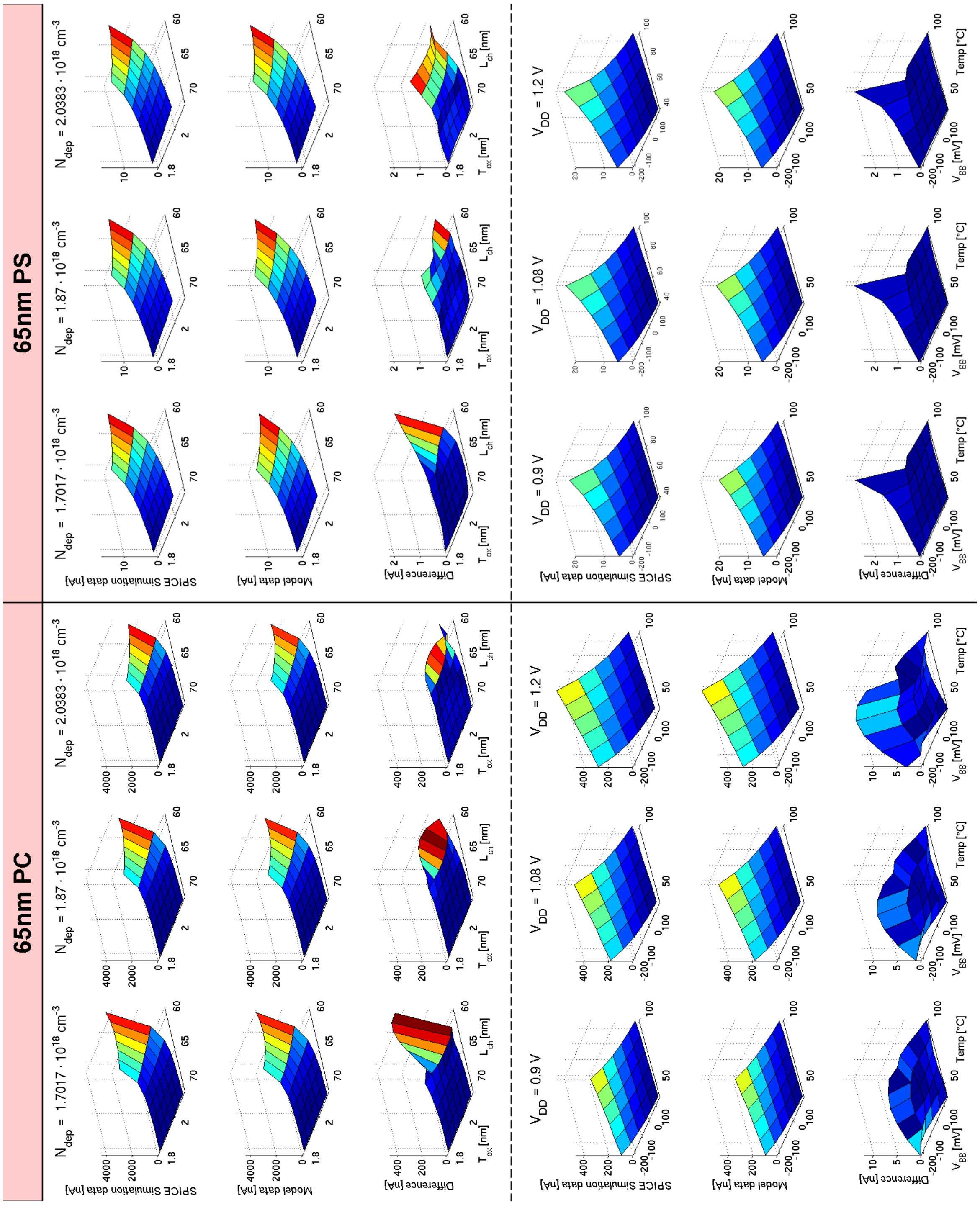


Figure C.5: PC and PS reference in 65nm

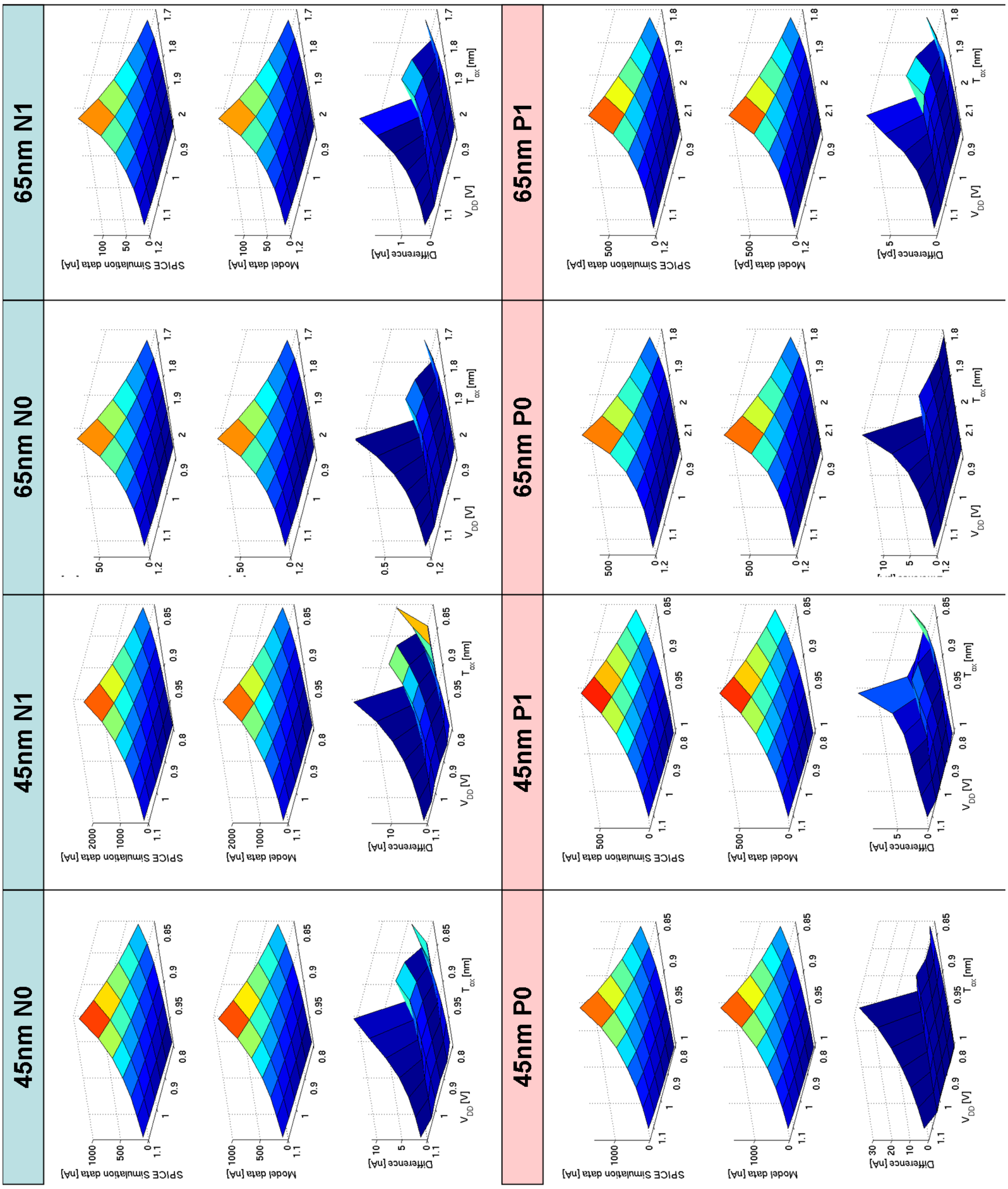


Figure C.6: All four gate leakage references in $45nm$ and $65nm$

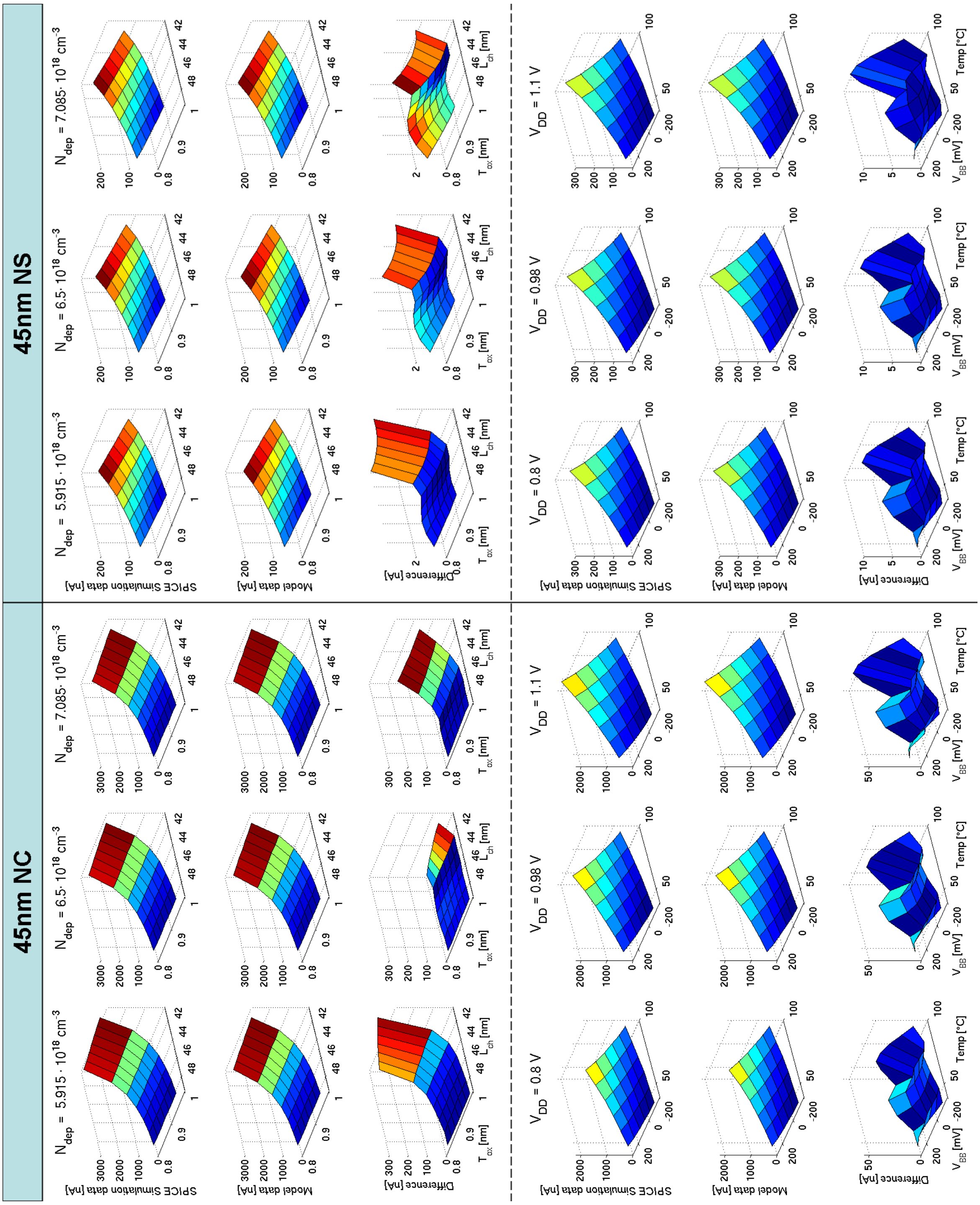


Figure C.7: NC and NS references in $45nm$

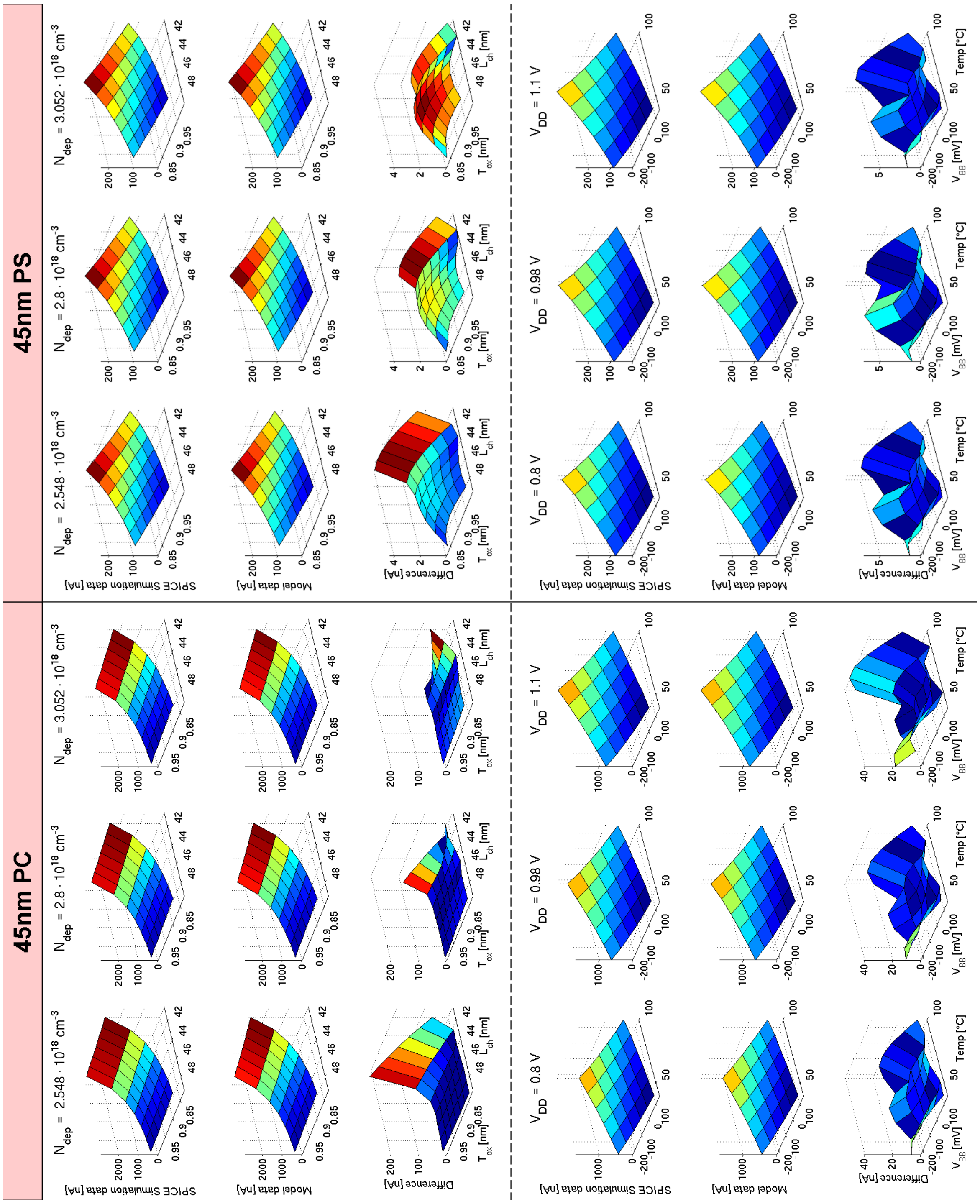


Figure C.8: PC and PS reference in $45nm$

Bibliography

[1] Kaushik Roy, Saibal Mukhopadhyay, and Hamid Mahmoodi-Meimand. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. In *Proceedings of the IEEE*, pages 305–327, February 2003.

[2] Domenik Helms, Ali Keshavarzi, and Wolfgang Nebel. Embedded tutorial: Leakage currents in nanometer CMOS. In *Intl. Symposium on Low Power Electronics and Design*, October 2006.

[3] Ashish Srivastava, Dennis Sylvester, Saumil Shah, David Blaauw, Kanak Agarwal, and Stephen Director. Accurate and efficient gate-level parametric yield estimation considering correlated variations in leakage power and performance. In *42nd Design Automation Conf.*, pages 535–540, June 2005.

[4] Haihua Su, Frank Liu, Anirudh Devgan, Emrah Acar, and Sani Nassif. Full chip leakage-estimation considering power supply and temperature variations. In *Intl. Symposium on Low Power Electronics and Design*, pages 78–83, August 2003.

[5] ITRS. Process integration, devices, and structures. Technical report, ITRS, 2006 Update. Table 39.

[6] ITRS. Process integration, devices, and structures. Technical report, ITRS, 2006 Update. Table 40a.

[7] Saibal Mukhopadhyay and Kaushik Roy. Modeling and estimation of total leakage current in nano-scaled-CMOS devices considering the effect of parameter variation. In *Intl. Symposium on Low Power Electronics and Design*, pages 172–175, August 2003.

[8] Nam Sung Kim, Todd Austin, David Blaauw, Trevor Mudge, Krisztián Flautner, Jie S. Hu, Mary Jane Irwin, Mahmut Kandemir, and Vijaykrishnan Narayanan. Leakage current: Moore's law meets static power. *IEEE Computer*, pages 68–75, December 2003.

[9] J. Adam Butts and Gurindar S. Sohi. A static power model for architects. In *Intl. Symposium on Microarchitecture*, pages 191–201, 2000.

[10] Yan Zhang, Dharmesh Parikh, Karthik Sankaranarayanan, Kevin Skadron, and Mircea Stan. Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects, March 2003.

[11] Ashish Srivastava, Robert Bai, David Blaauw, and Dennis Sylvester. Modeling and analysis of leakage power considering within-die process variations. In *Intl. Symposium on Low Power Electronics and Design*, pages 64–67, August 2002.

[12] Zhi-Hong Liu, Chenming Hu, Jian-Hui Huang, Tung-Yi Chan, Min-Chie Jeng, Pink K Ko, and Y C Cheng. Threshold voltage model for deep-submicrometer MOSFETs. *IEEE Trans. on Electron Devices*, 40:86–95, January 1993.

[13] Chenming Hu. BSIM model for circuit design using advanced technologies. *Symposium on VLSI Circuits Digest of Technical Papers*, pages 5–6, 2001.

[14] Yuhua Cheng, Min-Chie Jeng, Zhihong Liu, Kai Chen, Mansun Chan, Chenming Hu, and Ping Keung Kox. An investigation on the robustness, accuracy and simulation performance of a physics-based deep-submicronmeter BSIM model for analog/digital circuit simulation. In *IEEE Custom Integrated Circuits Conf.*, pages 321–324, 1996.

[15] Yu Cao, M Orshansky, T Sato, Dennis Sylvester, and Chenming Hu. Spice up your MOSFET modelling. *IEEE Circuits and Devices Magazine*, 17:17–23, July 2003.

[16] Rajeev Rao, Ashish Srivastava, David Blaauw, and Dennis Sylvester. Statistical estimation of leakage current considering inter- and intra-die process variation. In *Intl. Symposium on Low Power Electronics and Design*, pages 84–89, August 2003.

[17] Rajeev R. Rao, David Blaauw, Dennis Sylvester, and Anirudh Devgan. Modeling and analysis of parametric yield under power and performance constraints. *IEEE Design & Test of Computers*, 22:376–385, July 2005.

[18] Rahul M Rao, Jeffrey L Burns, Anirudh Devgan, and Richard B Brown. Efficient techniques for gate leakage estimation. In *Intl. Symposium on Low Power Electronics and Design*, pages 100–103, August 2003.

[19] Emrah Acar, Anirudh Devgan, Rahul Rao, Ying Liu, Haihua Su, Sani Nassif, and Jeffrey Burns. Leakage and leakage sensitivity computation for combinational circuits. In *Intl. Symposium on Low Power Electronics and Design*, pages 25–27, August 2003.

[20] Weiping Liao, Fei Li, and Lei He. Microarchitecture level power and thermal simulation considering temperature dependent leakage model. In *Intl. Symposium on Low Power Electronics and Design*, pages 211–216, August 2003.

[21] Kaustav Banerjee, Sheng-Chih Lin, Ali Keshavarzi, Siva Narendra, and Vivek De. A self-consistent junction temperature estimation methodology for nanometer scale ICs with implications for performance and thermal management. In *IEEE Intl. Electron Devices Meeting*, pages 36.7.1–36.7.4, December 2003.

[22] Siva Narendra, Vivek De, Shekhar Borkar, Dimitri Antoniadis, and Anantha Chandrakasan. Full-chip sub-threshold leakage power prediction model for sub-0.18μm CMOS. In *Intl. Symposium on Low power Electronics and Design*, pages 19–23, August 2002.

[23] Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De. Parameter variations and impact on circuits and microarchitecture. In *40th Design Automation Conf.*, pages 338–342, June 2003.

[24] Zhanping Chen, Mark Johnson, Liqiong Wei, and Kaushik Roy. Estimation of standby leakage power in CMOS circuits considering accurate modeling of transistor stacks. In *Intl. Symposium on Low Power Electronics and Design*, pages 239–244, August 1998.

[25] Saibal Mukhopadhyay, Arijit Raychowdhury, and Kaushik Roy. Accurate estimation of total leakage current in scaled CMOS logic circuits based on compact current modeling. In *40th Design Automation Conf.*, pages 169–174, June 2003.

[26] Domenik Helms, Marko Hoyer, and Wolfgang Nebel. Accurate PTV, state, and ABB aware RTL blackbox modeling of subthreshold, gate, and PN-junction leakage. In *16th Intl. Workshop on Power and Timing Modeling, Optimization and Simulation*, pages 56–65, September 2006.

[27] Domenik Helms, Günter Ehmen, and Wolfgang Nebel. Analysis and modeling of subthreshold leakage of RT components under PTV and state variation. In *Intl. Symposium on Low Power Electronics and Design*, pages 220–225, October 2006.

[28] Sarvesh Bhardwaj and Sarma B K Vrudhula. Leakage minimization of nano-scale circuits in the presence of systematic and random variations. In *42nd Design Automation Conf.*, pages 541–546, June 2005.

[29] Hongliang Chang and Sachin S. Sapatnekar. Full-chip analysis of leakage power under process variations, including spatial correlations. In *42nd Design Automation Conf.*, pages 523–528, 2005.

[30] Arijit Raychowdhury, Xuanyao Fong, Qikai Chen, and Kaushik Roy. Analysis of super cut-off transistors for ultralow power digital logic circuits. In *Intl. Symposium on Low Power Electronics and Design*, pages 2–7, October 2006.

[31] Mohan V. Dunga, Wenwei Yang, Xuemei Xi, Ali M. Niknejad, and Chenming Hu. BSIM4.6.1 MOSFET model - user's manual. Technical report, Department of Electrical Engineering and Computer Sciences University of California, Berkeley, 2007.

[32] Domenik Helms. Leakage power modeling, estimation and optimization. In *5th Marlow Workshop*, pages 1–40, April 2005.

[33] Domenik Helms, Eike Schmidt, and Wolfgang Nebel. Leakage in CMOS circuits - an introduction. In *14th Intl. Workshop on Power and Timing Modeling, Optimization and Simulation*, pages 13–35, September 2004.

[34] James E Draper. Quantal tunneling through a rectangular barrier using $|\psi|^2$ and flux. *American Journal of Physics*, 48:749–751, 1980.

[35] Nanoscale Integration and Modeling Group. 45nm BSIM4 modelcard for bulk CMOS v1.0 http://www.eas.asu.edu/∼ptm/, February 2006.

[36] Silicon Far East. Hot carrier effects http://www.siliconfareast.com/hotcarriers.htm. Technical report, siliconfareast forum, 2004.

[37] A. R. Brown, G. Roy, and A. Asenov. Poly-si gate related variability in decananometre MOSFETs with conventional architecture. *IEEE Trans. Electron Dev.*, 54(11):3056–3063, 2007.

[38] Nanoscale Integration and Modeling Group. 180nm BSIM4 modelcard for bulk CMOS v0.0 http://www.eas.asu.edu/∼ptm/, May 2001.

[39] Campbell Millar, David Reid, Scott Roy, and Asen Asenov. Accurate statistical description of random dopant-induced threshold voltage variability. *IEEE Electron Device Letters*, 29:946–948, August 2008.

[40] Asen Asenov. Random dopant threshold voltage fluctuations in 50nm epitaxial channel MOSFETs: A 3D 'atomistic' simulation study. In *28rd Conf. on European Solid-State Devices*, pages 300–303, September 1998.

[41] Steve Furber. Dealing with variability in modern circuit design. In *Intl. Conf. on CMOS Variability*, pages 1–14, October 2007.

[42] Mohan V. Dunga, Wenwei Yang, Xuemei Xi, Ali M. Niknejad, and Chenming Hu. BSIM4.6.1 MOSFET model - Chapter 15: Well proximity effect model. Technical report, Department of Electrical Engineering and Computer Sciences University of California, Berkeley, 2007.

[43] Hans Stork. The search for perfection in scaling. In *Intl. Symposium for Testing and Failure Analysis*, November 2005.

[44] Diana Marculescu and Sani Nassif. Design variability: Challenges and solutions at microarchitecture-architecture level. In *Half-day Tutorial; Design, Automation and Test in Europe*, March 2008.

[45] S Tyagi et al. A 130nm generation logic technology featuring 70nm transistors, dual Vt transistors and 6 layers of Cu interconnects. In *Electron Devices Meeting*, pages 567–570, 2000.

[46] Wei Huang, Eric Humenay, Kevin Skadron, and Mircea R Stan. The need for a fullchip and package thermal model for thermally optimized IC designs. In *Intl. Symposium on Low-Power Electronics and Design*, pages 245–250, August 2005.

[47] A P Jacob, T Myrberg, O Nur, M Willander, P Lundgren, E Ö Sveinbjörnsson, L L Ye, A Thölen, and M Caymax. Cryogenic performance of ultrathin oxide MOS capacitors with in situ doped p^+ poly-$\mathrm{Si}_{1-x}\mathrm{Ge}_x$ and poly-Si gate materials. *Semiconductor Sci. Technol.*, 17:942–946, July 2002.

[48] Mohan V. Dunga, Wenwei Yang, Xuemei Xi, Ali M. Niknejad, and Chenming Hu. BSIM4.6.1 MOSFET model - Chapter 13: Temperature dependence model. Technical report, Department of Electrical Engineering and Computer Sciences University of California, Berkeley, 2007.

[49] Srikanth Balasubramanian. Power delivery for high performance microprocessors. In *Intl. Symposium on Low Power Electronics and Design*, page 239, October 2008.

[50] Gianni Taraschi, Arthur J. Pitera, and Eugene A. Fitzgerald. Strained Si, SiGe, and Ge on-insulator: review of wafer bonding fabrication techniques. *Solid-State Electronics*, 48:1297–1305, April 2004.

[51] ITRS. Process integration, devices, and structures. Technical report, ITRS, 2007 Edition. Table PIDS1b.

[52] Dongwoo Lee, Wesley Kwong, David Blaauw, and Dennis Sylvester. Analysis and minimization techniques for total leakage considering gate oxide leakage. In *40th Design Automation Conf.*, pages 175– 180, June 2003.

[53] Daniel Connelly, Paul Clifton, Carl Faulkner, and D.E. Grupp. Ultra-thin-body fully depleted SOI metal source/drain n-MOSFETs and ITRS low-standby-power targets through 2018. In *IEEE Intl. Electron Devices Meeting*, pages 972–975, December 2005.

[54] Yiming Li and Chien-Sung Lu. Characteristic comparison of sram cells with 20 nm planar mosfet, omega finfet and nanowire finfet. In *IEEE Conf. on Nanotechnology*, pages 339–342, June 2006.

[55] Ashish Srivastava, Dennis Sylvester, and David Blaauw. Power minimization using simultaneous gate sizing, dual-Vdd and dual-Vth assignment. In *41st Design Automation Conf.*, pages 783–787, June 2004.

[56] Naran Sirisantana, Liqiong Wei, and Kaushik Roy. High-performance low-power CMOS circuits using multiple channellength and multiple oxide thickness. In *Intl. Conf. on Computer Design*, pages 227–232, September 2000.

[57] Frank Sill, Frank Grassert, and Dirk Timmermann. Total leakage power optimization with improved mixed gates. In *18th annual symposium on Integrated circuits and system design*, pages 154–159, September 2005.

[58] Jun Seomun, Jaehyun Kim, and Youngsoo Shin. Skewed flip-flop transformation for minimizing leakage in sequential circuits. In *44th Design Automation Conf.*, pages 103–106, June 2007.

[59] Anup K. Sultania, Dennis Sylvester, and Sachin S. Sapatnekar. Gate oxide leakage and delay tradeoffs for dual-Tox circuits. *IEEE transactions on very large scale integration VLSI systems*, 13:1362–1375, December 2005.

[60] Sarvesh Bhardwaj and Sarma Vrudhula. Leakage minimization of digital circuits using gate sizing in the presence of process variations. *IEEE Trans. on Computer Aided Design and Systems*, 27:445–455, March 2008.

[61] Bhaskar Chatterjee, Manoj Sachdev, Steven Hsu, Ram Krishnamurthy, and Shekhar Borkar. Effectiveness and scaling trends of leakage control techniques for sub-130nm CMOS technologies. In *Intl. Symposium on Low Power Electronics and Design*, pages 122–127, August 2003.

[62] Chandramouli Gopalakrishnan and Srinivas Katkoori. Behavioral synthesis of datapaths with low leakage power. *IEEE Intl. Symposium on Circuits and Systems*, 4:699–702, 2002.

[63] Chandramouli Gopalakrishnan and Srinivas Katkoori. Resource allocation and binding approach for low leakage power. In *16th Intl. Conf. on VLSI Design*, pages 297–302, January 2003.

[64] Chandramouli Gopalakrishnan and Srinivas Katkoori. Knapbind: An area-efficient binding algorithm for low-leakage datapaths. In *21st Intl. Conf. on Computer Design*, pages 430–435, October 2003.

[65] Chandramouli Gopalakrishnan and Srinivas Katkoori. Tabu search based behavioral synthesis of low leakage datapaths. In *IEEE Computer society Annual Symposium on VLSI*, pages 260–261, February 2004.

[66] Ranganath Gopalan, Chandramouli Gopalakrishnan, and Srinivas Katkoori. Leakage power driven behavioral synthesis of pipelined datapaths. In *IEEE Computer Society Annual Symposium on VLSI,*, pages 167–172, May 2005.

[67] Rajarshi Mukherjee, Seda Ogrenci Memik, and Gokhan Memik. Peak temperature control and leakage reduction during binding in high level synthesis. In *Intl. Symposium on Low Power Electronics and Design*, pages 251–256, August 2005.

[68] Sven Rosinger and Wolfgang Nebel. Power management aware resource allocation and binding reducing dynamic and static power. In *Will be submitted to 19th Intl. Workshop on Power and Timing Modeling, Optimization and Simulation*, 2009.

[69] Kiril Schröder. Development of a leakage reducing heuristic, optimizing the scheduling at the RT synthesis (Original title: Entwicklung einer Heuristik zur Reduktion der Leckströme durch Optimierung des Schedulings vor der RT-Synthese), April 2008.

[70] Domenik Helms. System level optimization of static power consumption in nano-CMOS circuits. In *14th Intl. Conf. Mixed Design of Integrated Circuits and Systems*, pages 169–171, June 2007.

[71] Siva Narendra et al. $1.1V$ $1GHz$ communications router with on-chip body bias in $150nm$ CMOS. In *IEEE Intl. Solid-State Circuits Conf.*, page 16.4, 2002.

[72] Cassondra Neau and Kaushik Roy. Optimal body bias selection for leakage improvement and process compensation over different technology generations. In *Intl. Symposium on Low Power Electronics and Design*, pages 116–121, August 2003.

[73] Maurice Meijer, Francesco Pessolano, and J. Pineda de Gyvez. Technology exploration for adaptive power and frequency scaling in $90nm$ CMOS. In *Intl. Symposium on Low Power Electronics and Design*, pages 14–19, September 2004.

[74] Ali Keshavarzi, James W. Tschanz, Siva Narendra, Vivek De, W. Robert Daasch, Kaushik Roy, Manoj Sachdev, and Charles F. Hawkins. Leakage and process variation effects in current testing on future CMOS circuits. *IEEE Design and Test of Computers*, 19:36–43, September 2002.

[75] James Tschanz, James Kao, Siva Narendra, Raj Nair, Dimitri Antoniadis, Anantha Chandrakasan, and Vivek De. Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. *IEEE Journal of Solid State Circuits*, 37:422–423, February 2002.

[76] Seongsoo Lee, Seungjun Lee, and Takayasu Sakurai. Energy-constrained VDD hopping scheme with run-time power estimation for low-power real-time VLSI systems. *Journal of Circuits, Systems, and Computers*, 11:601–620, February 2002.

[77] Koichi Nose, Masayuki Hirabayashi, Hiroshi Kawaguchi, Seongsoo Lee, and Takayasu Sakurai. VTH-hopping scheme to reduce subthreshold leakage for low-power processors. *IEEE journal of solid-state circuits*, 37:413–419, March 2002.

[78] James Tschanz, Keith Bowman, and Vivek De. Variation-tolerant circuits: Circuit solutions and techniques. In *42nd Design Automation Conf.*, pages 762–773, June 2005.

[79] Hari Ananthan, Chris H. Kim, and Kaushik Roy. Larger-than-VDD forward body bias in sub-0.5v nanoscale CMOS. In *Intl. Symposium on Low Power Electronics and Design*, pages 8–13, September 2004.

[80] Le Yan, Jiong Luo, and Niraj K. Jha. Joint dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems. *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, 24:1030–1041, July 2005.

[81] Domenik Helms, Olaf Meyer, Marko Hoyer, and Wolfgang Nebel. Voltage and ABB island optimization in high level synthesis. In *Intl. Symposium on Low Power Electronics and Design*, pages 153–158, August 2007.

[82] Sarvesh Kulkarni, Dennis Sylvester, and David Blaauw. Design-time optimization of post-silicon tuned circuits using adaptive body biasing. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 27:481–493, March 2008.

[83] Domenik Helms and Wolfgang Nebel. Logic design techniques for 65 to 45nm and below for reducing total energy and solving technology variations problems. In *14th IEEE Intl. Conf. on Electronics, Circuits and Systems*, December 2007.

[84] Vishal Khandelwal and Ankur Srivastava. Leakage control through fine-grained placement and sizing of sleep transistors. In *IEEE Trans. on Computer Aided Design of Integrated Circuits*, 2007.

[85] Ashoka Visweswara Sathanur, Luca Benini, Alberto Macii, Enrico Macii, and Massimo Poncino. Multiple power-gating domain (multi-vgnd) architecture for improved leakage power reduction. In *Intl. Symposium on Low Power Electronics and Design*, pages 51–56, October 2008.

[86] Sven Rosinger, Domenik Helms, and Wolfgang Nebel. RTL power modeling and estimation of sleep transistor based power gating. In *17th Intl. Workshop on Power and Timing Modeling, Optimization and Simulation*, pages 278–287, September 2007.

[87] Andrea Calimera, Antonio Pullini, Ashoka Visweswara Sathanur, Luca Benini, Alberto Macii, Enrico Macii, and Massimo Poncino. Design of a family of sleep transistor cells for a clustered power-gating flow in 65nm technology. In *17th Great Lakes Symposium on VLSI*, pages 501–503, March 2007.

[88] Hyung-Ock Kim, Youngsoo Shin, Hyuk Kim, and Iksoo Eo. Physical design methodology of power gating circuits for standard-cell-based design. In *43rd Design Automation Conf.*, pages 109–112, July 2006.

[89] Kim Suhwan, S. V. Kosonocky, D. R. Knebel, and K. Stawiasz. Experimental measurement of a novel power gating structure with intermediate power saving mode. In *Intl. Symposium on Low Power Electronics and Design*, pages 20–25, September 2004.

[90] Weiping Liao, Joseph M. Basile, and Lei He. Leakage power modeling and reduction with data retention. In *Intl. Conf. on Computer Aided Design*, pages 714–719, November 2002.

[91] Fei Li and Lei He. Maximum current estimation considering power gating. In *Intl. Symposium on Physical Design*, pages 106–111, April 2001.

[92] Afshin Abdollahi, Farzan Fallah, and Massoud Pedram. A robust power gating structure and power mode transition strategy for MTCMOS design. In *IEEE Trans. on Very Large Scale Integration VLSI Systems*, pages 80–89, January 2007.

[93] Ehsan Pakbaznia, Farzan Fallah, and Massoud Pedram. Charge recycling in MTCMOS circuits: concept and analysis. In *43rd Design Automation Conf.*, pages 97–102, July 2006.

[94] M. Anis, S. Areibi, and M. Elmasry. Design and optimization of multithreshold CMOS (MTCMOS) circuits. In *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 1324–1342, 2003.

[95] James W. Tschanz, Siva G. Narendra, Yibin Ye, Bradley A. Bloechel, Shekhar Borkar, and Vivek De. Dynamic sleep transistor and body bias for active leakage power control of microprocessors. *IEEE journal of solid-state circuits*, 38:1838–1845, November 2003.

[96] Patrick Knocke. Development of gate-level heuristics for idle time leakage-current reduction (Original title: Entwicklung von Gatterebenen-Heuuristiken zur Optimierung des Leckstroms im Ruhezustand), April 2007.

[97] Antoni Ferré and Joan Figueras. Leakage power bounds in CMOS digital technologies. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 21:731–738, June 2002.

[98] J.P. Halter and F. Najm. A gate-level leakage power reduction method for ultra-low-power CMOS circuits. In *IEEE Custom Integrated Circuits Conf.*, pages 475–478, 1997.

[99] Feng Gao and John P. Hayes. Exact and heuristic approaches to input vector control for leakage power reduction. In *Intl. Conf. on Computer Aided Design*, pages 527–532, November 2004.

[100] Lin Yuan and Gang Qu. Enhanced leakage reduction technique by gate replacement. In *42nd Design Automation Conf.*, pages 47–50, June 2005.

[101] Lei Cheng, Liang Deng, Deming Chen, and Martin D.F. Wong. A fast simultaneous input vector generation and gate replacement algorithm for leakage power reduction. In *43rd Design Automation Conf.*, pages 117–120, July 2006.

[102] Navid Azizi, Farid N. Najm, and Andreas Moshovos. TR-01–01–02: Asymmetric-cell caches: Exploiting bit value biases to reduce leakage power in deep-submicron, high-performance caches. Technical report, ECE Department, University Toronto, ON, Canada, 2002.

[103] Navid Azizi, Farid N. Najm, and Andreas Moshovos. Low-leakage asymmetric-cell SRAM. *IEEE Trans. on Very Large Scale Integration VLSI Systems,*, 11:701–715, August 2003.

[104] Stefanos Kaxiras, Zhigang Hu, and Margaret Martonosi. Cache decay: Exploiting generational behavior to reduce cache leakage power. In *Intl. Symposium on Computer Architecture*, pages 240–251, July 2001.

[105] Chris H. Kim and Kaushik Roy. Dynamic Vt SRAM: A leakage tolerant cache memory for low voltage microprocessors. In *Intl. Symposium on Low power Electronics and Design*, pages 251–254, August 2002.

[106] Se-Hyun Yang, Babak Falsafi, Michael D. Powell, Kaushik Roy, and T. N. Vijaykumar. An integrated circuit/architecture approach to reducing leakage in deep-submicron high-performance I-caches. In *7th Intl. Symposium on High-Performance Computer Architecture*, page 147, 2001.

[107] Jun Yang and Rajiv Gupta. Energy efficient frequent value data cache design. In *Intl. Symposium on Microarchitecture*, pages 197–207, 2002.

[108] Kimish Patel, Luca Benini, Enrico Macii, and Massimo Poncino. STV-cache: A leakage energy-efficient architecture for data caches. In *16th Great Lakes Symposium on VLSI*, pages 404–409, April 2006.

[109] France STMicrolectronics, FTM Crolles. STM 65*nm* designkit, internal reference documentation. Technical report, STMicrolectronics, September 2005.

[110] ChipVision. PowerOpt - interactive creation of RTL code from ESL IP for low-power applications, August 2008.

[111] Ansgar Stammermann, Domenik Helms, Milan Schulte, Arne Schulz, and Wolfgang Nebel. Binding, allocation and floorplanning in low power high-level synthesis. In *Intl. Conf. on Computer Aided Design*, pages 544–550, November 2003.

[112] Olaf Meyer. Development of heuristical methods optimizing leakage by body biasing (Original title: Entwicklung von heuristischen Verfahren zur Optimierung der Leakage durch Body Bisasing Verfahren), April 2007.

[113] France STMicrolectronics, FTM Crolles. STM 90*nm* designkit, internal reference documentation. Technical report, STMicrolectronics, September 2005.

[114] France STMicrolectronics, FTM Crolles. STM 45*nm* designkit, C45-DRM revB, internal reference documentation. Technical report, STMicrolectronics, April 2007.

[115] Dharmesh Parikh, Yan Zhang, Karthik Sankaranarayanan, Kevin Skadron, and Mircea Stan. Comparison of state-preserving vs. non-state-preserving leakage control in caches, 2003.

[116] Nanoscale Integration and Modeling Group. 65*nm* BSIM4 modelcard for bulk CMOS v1.0 http://www.eas.asu.edu/∼ptm/, February 2006.

[117] Kyung Ki Kim, Yong-Bin Kim, Minsu Choi, and Nohpill Park. Leakage minimization technique for nanoscale CMOS VLSI. *IEEE Design & Test*, 24:322–330, July 2007.

[118] Joan Figueras. CLEAN workshop on leakage estimation and optimization, 2006.

[119] Saibal Mukhopadhyay, Swarup Bhunia, and Kaushik Roy. Modeling and analysis of loading effect in leakage of nano-scaled bulk-cmos logic circuits. In *Design, Automation and Test in Europe*, pages 224–229, March 2005.

[120] Marko Hoyer. Modelling leakage in sub-100*nm* CMOS technologies on gate level (original title: Gate-level Modellierung von Leckströmen in sub-100*nm* CMOS Technologien.), March 2006.

[121] Marko Hoyer, Domenik Helms, and Wolfgang Nebel. Modeling the impact of high level leakage optimization techniques on the delay of RT-components. In *17th Intl. Workshop on Power and Timing Modeling, Optimization and Simulation*, pages 171–180, September 2007.

[122] Domenik Helms, Marko Hoyer, Sven Rosinger, and Wolfgang Nebel. RT level makro modelling of leakage and delay under realistic PTV variation. In *Workshop on Low Power Design Impact on Test and Reliability*, May 2008.

[123] Andrea Calimera, Enrico Macii, Massimo Poncino, and R. Iris Bahar. Temperature-insensitive synthesis using multi-Vt libraries. In *18th Great Lakes Symposium on VLSI*, pages 5–10, May 2008.

[124] Domenik Helms, Eike Schmidt, Arne Schulz, Ansgar Stammermann, and Wolfgang Nebel. An improved power macro-model for arithmetic datapath components. In *12th Intl. Workshop on Power and Timing Modeling, Optimization and Simulation*, pages 16–24, September 2002.

[125] Ranjith Kumar and Volkan Kursun. Reversed temperature-dependent propagation delay characteristics in nanometer CMOS circuits. *IEEE Trans. on Circuits and Systems II: Express Briefs*, 53:1078–1082, October 2006.

[126] Herming Chiueh, Jeffrey Draper, Louis Luh, and Jr. John Choma. A novel model for on-chip heat dissipation. In *IEEE Asia-Pacific Conf. on Circuits and Systems*, pages 779–782, November 1998.

[127] Ting-Yuan Wang, Yu-Min Lee, and Ching-Ping Chen. 3D thermal-ADI - an effcient chip-level transient thermal simulator. In *ISPD*, April 2003.

[128] Michael Solka and Jonathan Young. Whitepaper: A practical methodology for calculating acceptable IR drop targets in advanced VDSM design. Technical report, Synopsys, April 2006.

[129] Jim Tschanz, Siva Narendra, Ali Keshavarazi, and Vivek De. Techniques to minimize variation impacts on microprocessor performance and power. In *IEEE Intl. Symposium on Circuits and Systems*, pages 9–12, May 2005.

[130] D. F. Wong and C. L. Liu. A new algorithm for floorplan design. In *23rd Design Automation Conf.*, pages 101–107, June 1986.

[131] Michel Harrand. private conversation, Grenoble 2006.

[132] Gennady Gildenblat, Xin Li, Weimin Wu, Hailing Wang, Amit Jha, Ronald van Langevelde, Geert D. J. Smit, Andries J. Scholten, and Dirk B. M. Klaassen. PSP: An advanced surface-potential-based MOSFET model for circuit simulation. *IEEE Trans. on Electron Devices*, 53:1979–1993, September 2006.

[133] Yu Cao, Takashi Sato, Michael Orshansky, Dennis Sylvester, and Chenming Hu. New paradigm of predictive MOSFET and interconnect modeling for early circuit simulation. In *IEEE Custom Integrated Circuits Conf.*, pages 201–204, May 2000.

[134] Xuemei (Jane) Xi, Jin He, Mohan Dunga, Chung-Hsun Lin, Babak Heydari, Hui Wan, Mansun Chan, Ali M. Niknejad, and Chenming Hu. The next generation BSIM for sub-100nm mixed-signal circuit simulation. In *IEEE Custom Integrated Circuits Conf.*, pages 13–16, October 2004.

[135] Yu Cao, Wei Zhao, and Chi-Chao Wang. Predictive technology model nano-CMOS online characterization http://www.eas.asu.edu/∼ptm/. Technical report, Nanoscale Integration and Modelling Group, Arizona State University, 2007.

[136] Mohan V. Dunga, Wenwei Yang, Xuemei Xi, Ali M. Niknejad, and Chenming Hu. BSIM4.6.1 MOSFET model - Chapter 14: Parameter extraction methodology. Technical report, Department of Electrical Engineering and Computer Sciences University of California, Berkeley, 2007.

[137] V Granville, M Krivanek, and J P Rasson. Simulated annealing: A proof of convergence. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16:652–656, June 1994.

[138] John Cooley. Deepchip homepage http://www.deepchip.com/items/snug05-18.html. Technical report, 2005.