Zur Homepage der Dissertation

Carl von Ossietzky Universität Oldenburg
Fachbereich Informatik

Dissertation

Power Modeling of Embedded Memories

Eike Schmidt

March 25, 2003

zur Erlangung des Grades eines Doktors
der Ingenieurswissenschaften

Gutachter/Supervisor:	Professor Dr. Wolfgang Nebel
Zweitgutachter/Reviewer:	Professor Dr. Norbert Wehn
Tag der Disputation:	10. Februar 2003

To my family

Abstract

After Moore's Law the number of transistors on an integrated circuit doubles every 18 months. New circuits are furthermore clocked with increasing frequencies. This development not only leads to an increase of the available functionality, but also to a rise of the electrical power consumption of these systems.

The power consumption of integrated circuits is problematic in two respects: on one hand the power must be fed into the system, on the other hand the heat produced on the chip must be dissipated. An increased power consumption consequently leads to reduced battery lifetimes and increased energy costs. Today already 80% of the office related power consumption in the USA stems from computers. The heat development of integrated circuits reduces their reliability and lifetime. The required cooling measures (ceramic packages, heat fins, fans, etc.) furthermore increase the system cost.

For the development of low power systems it is necessary to estimate and consider the power consumption already in the early stages of design. For such estimates models of the system blocks are required.

This thesis describes a methodology for the generation of models of the power consumption of embedded memories. Memories have a special importance among the system blocks as it is forecast that more than 90% of the area of newly developed systems will be occupied by memory within ten years. In addition to the requirements on models, like accuracy, speed and mathematically closed form the presented approach specifically adresses the requirements on the modeling procedure. These are mainly the protection of intellectual property and low modeling costs.

As a key point of the approach a method for the generation of nonlinear (signomial) models from empirical data is presented. This regression based method allows the generation of piecewise model functions, which can be used for optimisation through geometric programs. The presented models reduce the prediction error by up to 95% compared to existing approaches.

Zusammenfassung

Nach Moore's Law verdoppelt sich die Zahl der auf einem Computerchip integrierbaren Transistoren alle 18 Monate. Neue Schaltungen werden darüber hinaus mit immer größeren Geschwindigkeiten betrieben. Diese Entwicklung führt nicht nur zu der gewünschten Zunahme an verfügbarer Funktionalität, sondern auch zum Anstieg der elektrischen Leistungsaufnahme dieser Systeme.

Die Leistungsaufnahme Integrierter Schaltungen ist aus zwei Blickwinkeln problematisch: Zum einen muss die Leistung dem System zugeführt, zum andere die entstehende Wärme abgeführt werden. Eine erhöhte Leistungsaufnahme führt daher zu sinkenden Batterie- und Akkubetriebszeiten und erhöhten Energiekosten. Schon jetzt entfallen 80% der in den USA in Büroumgebungen verbrauchten Energie auf Computersysteme. Die Wärmeentwicklung von Integrierten Schaltungen reduziert ihre Verlässigkeit und Lebensdauer. Die notwendigen Kühlungsmaßnahmen (Keramikgehäuse, Kühlkörper, Lüfter, etc.) erhöhen zudem die Systemkosten.

Zur Entwicklung von verlustleistungsarmen Systemen ist es notwendig, die Leistungsaufnahme bereits früh im Entwurf abzuschätzen und in Entwurfsentscheidungen einzubeziehen. Für solche Abschätzungen sind Modelle der Strukturblöcke notwendig.

Diese Arbeit beschreibt eine Methodik für die Erstellung von Modellen der Leistungsaufnahme von eingebetteten Speichern. Speichern kommt unter den Strukturblöcken eine besondere Bedeutung zu, da sie in 10 Jahren voraussichtlich über 90% der Fläche neuentwickelter Schaltungen einnehmen werden. Neben den Anforderungen an die Modelle, wie Genauigkeit, Geschwindigkeit und mathematischer Geschlossenheit berücksichtigt der vorgestellte Ansatz insbesondere Anforderungen an den Modellierungsprozess selbst. Dies sind hauptsächlich der Schutz intellektuellen Eigentums, sowie die Gewährleistung niedriger Modellierungskosten.

Als Kernpunkt der Methodik wird ein Verfahren zur Generierung nichtlinearer (signomialer) Modelle aus empirischen Daten vorgestellt. Das regressionsbasierte Verfahren erlaubt die Erzeugung stückweiser Modellfunktionen, welche über geometrische Programme optimierbar sind. Die vorgestellten Modelle reduzieren den Vorhersagefehler um bis zu 95% verglichen mit bisherigen Ansätzen.

Acknowledgements

First of all I would like to thank my supervisor Prof. Dr. Wolfgang Nebel for his trust and support. He created the fruitful scientific environment that I was glad to share. I would also like to thank Prof. Dr. Norbert Wehn for taking the time to review this document.

Let me express my gratitude to my colleagues at the OFFIS, the University and Chip Vision Design Systems.. Special thanks to Dr. Gerd von Cölln, Dr. Lars Kruse, Ansgar Stammermann and Domenik Helms for many discussions and insightful comments. And all the other people that shared their opinions and advice, like project partners and the people at the Philips Natlab: thank you.

On a personal note I would like to say thank you to my family for the unwavering support and the vivid interest during all these years. A special thank you to Nicole for giving me the space, the support and the love to finish this work with many extra hours. I could not have done this without you. My apologies to the friends that I neglected too long.

Last but not least my gratitude goes to my employer, the Kuratorium OFFIS e.V., for giving me the opportunity to participate in the EU projects PEOPLE and POET in which the foundations of the presented work were conceived.

Contents

1 Introduction

This chapter introduces and motivates the concept of memory power estimation and briefly describes the scope and contributions of this thesis.

1.1 Power Consumption

Power consumption has become an important aspect in the design of computation intensive integrated circuits (IC's). Today it influences the feasibility and cost of designs and entire technologies such as third generation mobile phones [37]. Consequently it has become a new dimension in the design cost space complementing the traditional factors performance, area and testability. Power consumption affects the design and use of IC's in several ways. The different power related issues can be connected to two main aspects: a) the energy consumed on the chip has to be provided and b) the energy is turned into heat within the chip.

Table 1.1: Short and long term forecast of ASIC power consumption [65, 66].

Year	2001	2002	2003	2004	2005	2006
Feature size/nm	150	130	107	90	80	70
MTransistors/chip	714	899	910	1020	1286	1620
Max clock /GHz	1.7	2.3	3.1	4.0	5.2	5.6
Min voltage/V	1.2	1.2	1.1	1. 1	1.0	1.0
Power (high)/W	130	140	150	160	170	180
Power (battery)/W	2.4	2.6	2.8	3.2	3.2	3.5

Year	2007	2010	2013	2016
Feature size/nm	65	45	32	22
MTransistors/chip	2041	4081	8163	16326
Max clock /GHz	6.7	11.5	19.3	28.8
Min voltage/V	0.9	0.8	0.7	0.6
Power (high)/W	190	218	251	288
Power (battery)/W	3.5	3.0	3.0	3.0

The necessity to provide power is a sensitive issue especially for mobile systems which are fed by batteries. Battery lifetime as well as weight and volume influence product feasibility and customer acceptance. Good examples for this category of applications are mobile phones, laptops and personal digital assistants. For stationary systems energy cost is a factor that causes rising concern. According to estimates done by the U.S. Environmental Protection Agency 80% of the total office equipment electricity is due to computers [94]. Power consumption does affect circuit reliability. High consumption also means high on-chip currents. These in turn impact signal integrity negatively (e.g. resistive voltage drops, ground bounce) and trigger failure mechanisms, such as electro-migration and hot-electron degradation [120].

IC's do not perform work in the mechanical sense. Instead all the energy fed into the chips is turned into heat. Some modern ICs reach a heat emission of $30W/cm^2$ and more. It equals the heat energy radiated by a 1200°C hot material and ten times the heat density of an electrical stove [75]. High chip temperatures, however, can reduce the system reliability. Every 10°C increase in temperature roughly

doubles the failure rate of components [112]. Cooling measures are hence commonly used to dissipate the excess heat. These measures range from ceramic chip packages and heat fins, over fans to liquid cooling. The choice of such cooling can drastically influence production costs. Aggressive cooling using fans may furthermore affect customer acceptance, e.g. in home hifi and video.

The power consumption will be an even more pressing problem in future: with increasing chip integration and operating frequency the power demands of systems are expected to rise further (cf. table 1.1). As the supply voltage will drop in future technologies, the on-chip currents will increase even faster.

1.2 Memories

Memories play an important role in integrated circuits as they are used for both, intermediate information storage as well as inter-block communication. The pace at which requirements of storage bandwidth and storage capacity in systems increased has been higher than the progress in memory circuits. This has rendered memory a bottleneck in many systems and made it a technology driver in the past decades [110, 111].

Table 1.2: Forecast of fraction of ASIC die area used for memory [65].

Year	1999	2002	2005	2008	2011	2014
%area new logic	64	32	16	8	4	2
%area reused logic	16	16	13	9	6	4
%are memory	20	52	71	83	90	94

In considering the power consumption of systems, memories are a crucial aspect as they can be responsible for a major part of the total system dissipation. In the signal processing domain a share of up to 80% of the total power has been attributed to the memory subsystem [157]. It is generally expected that the fraction of the total system taken by memories will further increase (cf. table 1.2). The major reason for this is that embedded memory has a relatively small design cost per area in terms of manpower and time to market. Another reason remarkably is related to power consumption: since the per area heat development of memories is lower than that of logic, memory can be used to add functionality with a smaller impact on the total heat.

1.3 IC Design

The design of integrated circuits is a complex process spanning several levels of abstraction. A *macro based top-down* design flow is state of the art in most domains and will be assumed throughout this thesis. This flow employs stepwise refinement and the mapping of functionality onto macro primitives. An exemplary flow of this kind is described in the following [23, 114]: the flow starts with a *system specification* that is divided into tasks. During hardware/software partitioning it is decided which of these tasks to implement in hard- respectively software. Processor cores for the execution of the software are subsequently allocated and communication mechanisms are defined, resulting in a *behavioral level* specification.

Algorithms are used to describe the circuit on behavioral level. No architecture is yet fixed for the hardware blocks and timing is described for the outside behavior only. In the design flow the algorithmic descriptions may undergo several transformation iterations (algorithm selection, optimizations). Subsequently the behavioral level synthesis steps scheduling, allocation and binding are performed producing a structural register transfer level description (architecture). This description is a net list of macro blocks (e.g. functional units like adders, multipliers, etc.) and registers. The timing is cycle accurate, values are expressed as bit-vectors.

The RT net list is transformed into a gate-level net list during logic synthesis. This consists of the substitution of RT level blocks by their gate level net lists, controller synthesis, logic optimization, retiming and technology mapping. The gate-level representation is a net list of cells, i.e. primitives of the target technology. Timing is accurate, effects of placement and routing, however, can only be estimated

(e.g. wire-load models). Finally the gate net list is transformed into a layout. Layout synthesis consists mainly of the steps floorplanning, placement and signal/clock/power routing. The result is a geometric description of material layers on the die.

1.4 Design Automation

In reaction to the technology developments mentioned above systems on chip have increased in complexity at dramatic speed. Due to this complexity and the importance of time-to-market, iterative design cycles have become expensive. This situation has pushed the development of technologies to reduce and shorten design cycles, namely re-use methodologies and electronic design automation. Two intertwined key technologies in design automation are synthesis and estimation. Synthesis refers to the automated execution of a design step subject to user defined constraints. Estimation seeks to predict the properties of the synthesis results for a given design description. For the reasons mentioned there is a strong trend towards entering an automated design flow as early as possible [114](see also table 1.3). The higher the level of design, however, the more important is estimation: as the path down to layout gets longer, the length of the feedback cycle increases and transparency of synthesis results is reduced. Hence design space exploration greatly benefits by estimation approaches that allow to prune the design space without synthesis. For power reduction it is important to take the power aspect into consideration as early as possible in the design flow because the optimization potential sharply increases with the abstraction level (see figure 1.1 left). Consequently it is clearly desirable to use the speed-up of high level estimation technology to enable power oriented design space exploration (see figure 1.1 right). Furthermore, since synthesis inherently means the solving of an optimization problem, including the power aspect into this optimization can, although not yet state of the art in commercial tools, significantly reduce power without manual intervention [95].

Table 1.3: Behavioral Synthesis Revenue (Millions of Dollars) [131].

Year	1998	1999	2000	2001	2002	2003	2004	2005
Revenue	13.1	10.5	7.8	9.4	11.2	34.5	58.0	79.8

1.5 Models

Models are a mathematical representation of the properties, e.g. the power consumption, of (circuit) primitives obtained by abstraction. They form the basis for both power estimation and optimization/synthesis. To incorporate memory power into the design space exploration, models for a wide variety of types and brands of memories are needed. To suite a high-level design flow as sketched above, models must fulfill

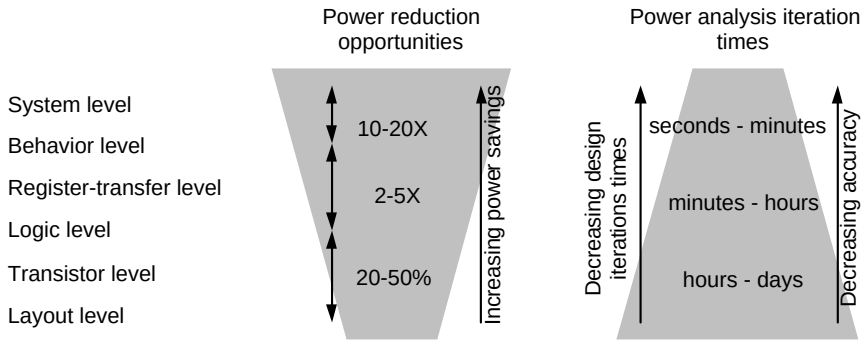


Figure 1.1: Possible power reduction and design iteration times on different levels of abstraction [114].

the speed, accuracy and pragmatic requirements of different abstraction levels. These requirements are discussed in more detail in Section 3.3.

1.6 Scope and Main Contributions

Power estimation and optimization at high levels of abstraction (e.g. RT level and higher) have been a focus of research at the research and development institute OFFIS and the Carl-von-Ossietzky University in Oldenburg in recent years [78, 79, 81, 80, 82, 70, 71]. The behavioral level power estimation tool suite ORINOCO has been developed as a result of several research projects [3, 107, 108, 132].

This thesis complements the previous work by incorporating memory power estimation into the ORINOCO estimator. The main contributions are:

- A discussion of the memory power modeling problem.
- The definition of requirements on high-level power models.
- A new power modeling methodology for embedded memories, widely automating the modeling process.
- A tool which implements the modeling flow. This tool has in the meantime matured to a commercial product marketed as part of the ORINOCO tool suite.
- An interface concept for the integration of (memory) power models into the estimation/optimization tool ORINOCO DALE.
- The evaluation of the methodology and tool using several practical examples.

1.7 Chapter Overview

The rest of this thesis is organized as follows: The next chapter briefly describes the physical aspects of power consumption as well as the fundamental structures of embedded memories. Memory optimizations are presented as an application field of models. Chapter 3 contains an introduction to statistical modeling. It concludes with a catalogue of requirements on memory power models. In chapter 4 related work on the topic is discussed. Chapter 5 presents the new memory power modeling methodology. The subsequent chapter discusses the implementation of the modeling flow in the prototype tool ORINOCO BEACH and the integration of (memory) models in an estimation tool. In chapter 7 the tool and flow is evaluated using several practical examples. The thesis closes with a short summary and conclusion in chapter 8.

2 Power Consumption in Memory Circuits

This chapter begins with a description the physical origins of power dissipation and its effects. This description is followed by a short introduction to semiconductor memories [110, 111, 52].

2.1 Sources of power dissipation

This section gives an overview of the physical sources of power dissipation. It is assumed that the reader is familiar with the basics of electronics, transistors and circuit design.

Four sources of power dissipation can be identified in MOS circuits:

$$P = P_{switching} + P_{short-circuit} + P_{leakage} + P_{static} \tag{2.1}$$

$P_{switching}$ refers to the power consumed for the charging/discharing of internal nodes. $P_{short-circuit}$ is the power related to direct paths from V_{DD} to ground that occur while a gate is switching. $P_{leakage}$ is due to currents flowing across off transistors. P_{static} is consumed by permanent power ground paths that exist for example in pseudo NMOS circuits. In memories such paths are often found in sense amplifiers, voltage and back-bias generators.

Though its share is expected to decrease, *switching power* is still the dominant factor (about 80 % of the total in logic circuits [120]). The energy drawn from the power supply to charge a node of capacitance C_L by V_{swing} can be computed as:

$$E = \frac{1}{2} \cdot C_L \cdot V_{DD} \cdot V_{swing} \tag{2.2}$$

With α the average number of transitions per clock cycle and f_{clk} the clock frequency $P_{switching}$ now becomes:

$$P_{switching} = 1/2 \cdot \alpha \cdot C_L \cdot V_{DD} \cdot V_{swing} \cdot f_{clk} \tag{2.3}$$

Assuming full voltage swing (e.g. $V_{swing} = V_{DD}$) this equation shows that switching power is quadratically dependent on the supply voltage and linearly on the switching activity α and the load capacitance C_L. The load capacitance has three components: $C_L = C_{gate} + C_{diffusion} + C_{interconnect}$. Capacitances between the gate and source, drain and bulk are lumped into one capacitance C_{gate} to ground. Similarly the diffusion capacitance between drain and bulk form $C_{diffusion}$. $C_{interconnect}$ comprises of the capacitance of the interconnect wire to ground and the cross-capacitances to other wires. It depends on the three dimensional geometry of the circuit layout.

Unlike the switching power, *short-circuit* power is dependent on the slope of the input signal. For a symmetric CMOS inverter the average short-circuit current can be derived as (see [120] [26]):

$$I_{mean} = \alpha \cdot \frac{\beta}{12 \cdot V_{DD}} \cdot (V_{DD} - 2V_t)^3 \cdot \tau \cdot f_{clk} \tag{2.4}$$

where β is the transistor strength, V_t the threshold voltage and τ the signal transition time. For more complex gates the relationship is not as easily computed. Still the equation shows that leakage power scales linearly with the signal transition times. In well designed logic circuits, short-circuit power can be kept below 10 % of the total power [26]. Short-circuit currents do normally not occur in dynamic circuits: the pre-charge is turned off before the evaluation phase, preventing a conductive path. Memories in some cases form an exception to this. Due to very tight timing margins pre-charge and evaluation phase may

slightly overlap (e.g. for bit-line pre-charge and sensing). [96] show that in future low-threshold technology short circuit-power can take up to 20% of the total power consumption.

Leakage power mainly stems from two mechanisms: reverse bias diode leakage and sub-threshold leakage. Reverse bias diode currents occur when the drain-bulk voltage of an off transistor becomes negative and a reversed diode is formed. The current is given by:

$$I_{diode} = I_{saturation} \cdot (e^{\frac{V}{V_{thermal}}} - 1) \tag{2.5}$$

where $I_{saturation}$ is the reverse saturation current, V is the drain-bulk voltage and $V_{thermal} = k_B T/q$ (with k_B the Bolzman-constant, T temperature and q the elementary charge) the thermal voltage $V_{thermal}(300° K) = 0.026V$. The diode current is strongly dependent on diode voltage and temperature. Its contribution to the overall power consumption is usually small, though. For a 1 million transistor chip, assuming an average drain area of $10\mu m^2$ [26] report leakage currents in the order of $25\mu A$ at $25°C$.

The subthreshold current of a MOS transistor is given as:

$$I_{substhreshold} = K \cdot e^{\frac{(V_{gs}-V_t)}{(nV_t)}}(1 - e^{-\frac{V_{ds}}{V_t}}) \tag{2.6}$$

where K is a technology dependent parameter, V_{gs} and V_{ds} the gate-source and drain-source voltages, V_t the threshold voltage and n a parameter depending on the oxide thickness and the channel depletion width. The subthreshold current is due to carrier diffusion between source and drain. It occurs when the gate-source voltage is above the weak inversion point but below the threshold voltage. Subthreshold currents have traditionally been negligible. They however depend exponentially on the threshold voltage. At the transistor threshold ($V_{gs} = V_t$) approximately $0.14\mu A/\mu m$ gate width is dissipated [26].

In summary the dynamic currents ($P_{switching}$ and $P_{short-circuit}$) presently dominate total power consumption of systems. In memories however, where large parts of the circuit are idle for most of the time and aggressive low voltage and high performance techniques are used $P_{leakage}$ already is considerable. It is expected that leakage effects will exceed the capacitive currents during operation around the 2GBit memory generation [68].

2.2 Classification of memories

Storage media can be divided into mechanical (e.g. punch cards), moving media (e.g. hard drives) and solid state [110]. The group of solid state memories contains the well established semiconductor memories and the ferroelectric memories, which are just now turning into marketable products. As semiconductor memories classify MOS (Metal On Silicon), bipolar and CCD (Charge Coupled Devices). The major types of MOS memories are: dynamic random access memory (DRAM), static random access memory (SRAM), read-only memory (ROM), erasable programmable ROM (EPROM), electrically erasable programmable ROM (EEPROM) including flash RAM and register file.

The following properties are important for the system environment and are thus used to qualify and select memories:

1. *Speed.* The speed is defined by the clock frequency, read and write timing, latencies and bandwidth.

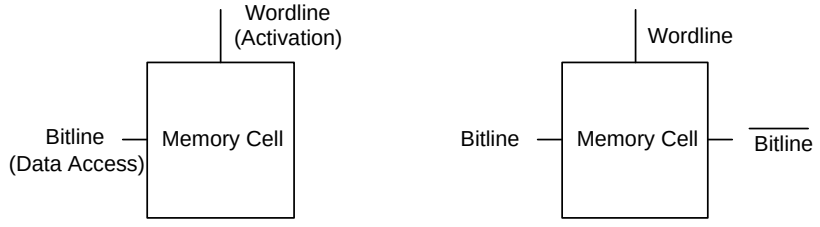


Figure 2.1: Single ended and double ended memory cells.

2. *Density.* The storage capacity per chip area depends mainly on the type of memory cell and technology used.
3. *(Re)programmability.* Defines the expected number of writes possible during the memories lifetime.
4. *Volatility.* Volatile memories loose their content on power off.
5. *External organization.* External organization refers to the ways a memory can be accessed.

For a characterization of the memory types mentioned above with respect to these categories see table 2.1.

Memories can also be classified according to their role in the system context: Storage elements directly in the data path are termed *foreground memories* in contrast to centralized off-data path storage named *background memory*. *Off-chip memories* are distinguished from *on-chip* or *embedded memories*. While the former come in discrete chip packages, the latter are integrated into the "system on chip" using augmented CMOS fabrication processes. A final distinction stems from data organization: As a compromise between cost (in terms of timing, area and power) and performance memory is usually organized in a *memory hierarchy* of several levels: Starting with small high-performance memories at the data path, higher levels comprise larger, but slower memories. *Cache memories* specifically support the use in hierarchies by dedicated hardware handling its interaction with the next higher memory. *Scratch pad memory* refers to a small and fast on chip RAM with no correspondence in higher levels of the hierarchy and usually directly under application control [98].

2.3 Block architecture

The core of MOS memories are the memory cells holding the stored bit values (see figure 2.1). A *single ended* memory cell has one data in/output line, the more common *double ended* cells have two complementary in/outputs. Each cell further has a select input. Although the internal cell structures differ between memory types, the cells themselves are always arranged in two dimensional arrays (see figure 2.2). All select inputs are connected to *word lines* that run over the array horizontally. *Bit lines* cross the array vertically and are connected to the cell data in-/outputs. Each cell consequently has a horizontal ("column", "bit") and a vertical ("row", "word" or "x") address. Modern memory commonly consist of a number Z of such arrays or *banks*. The B bits belonging to a single data word are stored horizontally, i.e. with identical word but different column address. Each row holds Y such B bit words and thus $Y \cdot B$ columns/bits. A bank consequently has a capacity of $X \cdot Y$ words ($X \cdot Y \cdot B$ bits) and the total memory $X \cdot Y \cdot Z$.

Figure 2.3 shows a typical memory block structure. Thus structure and the functionality of its blocks can be best explained following a memory access cycle: The cycle begins by the entering of the different address parts (row, column and bank) into the memories *address buffers*. Subsequently the addresses are decoded using $n \rightarrow 2^n$ *address decoders* (where n is the number of address lines). The decoded bank address is used to trigger the selected bank. One single word line is then activated according to the row address connecting all cells of that row to their respective bit lines. Now the *read/write circuitry* can access the cells using the bit lines. This block in turn communicates with the outside world via a *read/write bus* running over all memory columns horizontally and respective *data input/output buffers*. The access

Table 2.1: Characteristics of MOS memory types [110].

Characteristic	DRAM	SRAM	ROM	EPROM	EEPROM
relative cell size	1.5	4-6	1.0	1.5	3-4
volatile	yes	yes	no	no	no
reprogrammable	yes	yes	no	no	yes
write speed	30ns	5ns	-	30min	10 ms
read speed	30ns	5ns	50ns	80ns	120ns
#erase/write	∞	∞	0		500,000

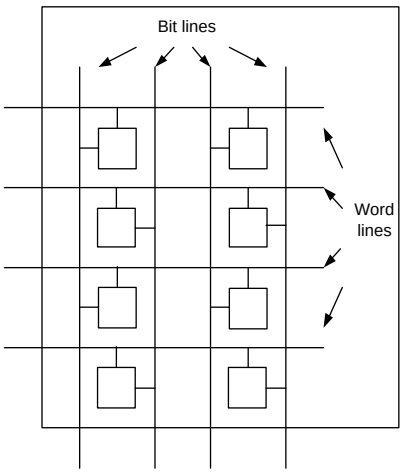


Figure 2.2: Memory array containing single ended cells.

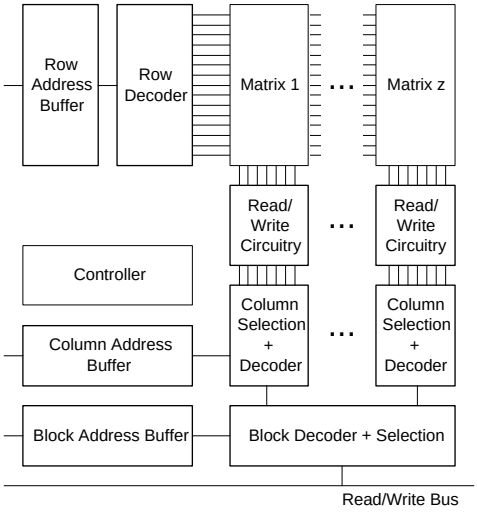


Figure 2.3: Typical memory block structure.

sequences are coordinated by a control unit. In *synchronous memories* this unit is triggered by the clock, in *asynchronous memories* by changes at the address/data inputs.

2.3.1 Address Logic

The address decoders are usually decoupled from the address inputs using address buffers. These buffers reduce the input load and keep the addresses stable during the memory cycle. While latches can fulfill these requirements, in high performance "registered" memories registers may take their place offering more flexibility in the memory I/O timing [111].

During read the data input buffers are disconnected from the internal bus using tri-states. Similarly the data output unit also often incorporates a tristate buffer to decouple the memory from an external data bus. Speed and area are the main requirements on address decoders. Commonly they are static (see figure 2.4) or dynamic CMOS (figure 2.5), possibly cascaded into two or more stages. Power consumption of decoders are usually dominated by the output drivers. The output loads of the row, column and bank

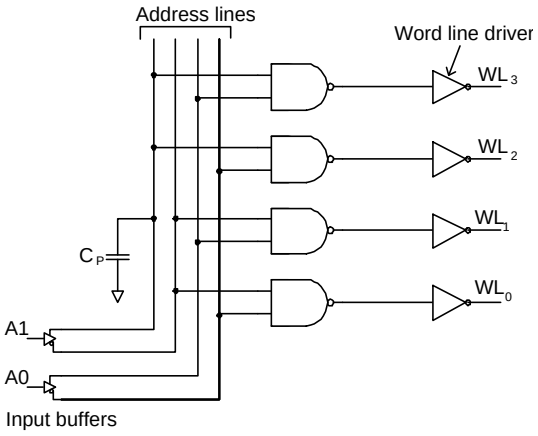


Figure 2.4: Statical Row Decoder.

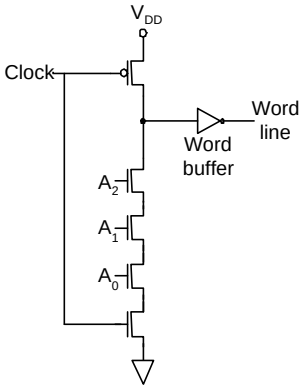


Figure 2.5: Dynamic Row Decoder.

decoders are the row lines, the column select circuitry and the bank activation circuitry respectively. Since the circuitry is symmetrical in respect to the address, power does also not vary significantly with address.

The column select circuitry selects one out of the Y horizontal words. It usually employs pass transistors to connect the wanted columns to the sensing circuitry. The column decoders have no drivers and so only their input load contributes to the power consumption.

2.3.2 Read/Write Circuitry

Due to the limited driving capability of memory cells and comparatively high bit line capacitancies the voltage level changes observed at the bit lines during read are slow and possibly not full swing. *Sense amplifiers* are therefore used to accelerate the read process: in the most simple case the amplification is performed by inverter type buffers. For higher performance *differential amplifiers* are used. These circuits amplify the voltage between the two complementary bit lines or, for single ended memories, between the bit line and a reference line. Several variations of these circuits exist (e.g. figure 2.6). Power considerations are complicated by the fact that sense amps are not full swing static CMOS. As they have a considerable DC current, they are often used in pulsed mode. To ensure secure operation inputs and/or outputs are usually equalized by shorting them during idle periods. Sense amps may be cascaded to achieve sufficient amplifier gain.

For write accesses it is necessary to drive the bit lines plus the inputs of all deselected cells and one selected cell. Simple buffers are used to perform this task.

2.3.3 Auxiliary Circuitry

Voltage generators are often used in memory circuits. A half-voltage generator enables $V_{DD}/2$ bit-line precharge. It usually has a DC current, so that pulsed mode is recommended. Charge pumps are employed for boosted voltage generators that supply levels above V_{DD} for driving the word lines in DRAMs. Back bias generators use the same technique to avoid forward biasing of junctions (electrons are injected into the substrate leading to dynamic circuit problems and reduced refresh times in DRAMs). Leakage control also requires additional voltages [68].

2.4 Types of Memories

2.4.1 Static RAM

The static RAM is based on bistable transistor flip-flop cells. The most common type of SRAM cell is depicted in figure 2.7. It consist of two cross coupled inverters that are decoupled from a pair of complementary bit lines by access transistors. Variations of this scheme use passive loads, single bit-lines or multiple ports.

During operation the cell is in one of two stable states, where points A and B have complementary logic levels (A=1, B=0 or A=0,B=1). No refresh is needed and data is retained as long as the supply voltage is on. On read or write the cell is connected to the bit lines by activation of the pass transistors through the word line. The bit line connected to the '1' node is then pulled high, the one connect to the '0' node is pulled low. Sense amplifiers are used to speed up the reading process by amplifying the voltage difference between the two bit lines. SRAM read is a non-destructive process as the logic state of the cell remains unchanged. For writing into a cell, the bit lines are charged to their target values and forced to stay there while the cell is activated.

The main advantages of static RAM over dynamic RAM are higher speed, no need for refresh and lower data retention current. Its main drawback is its considerably larger size.

2.4.2 Dynamic RAM

The DRAM memory cell consists of a storage capacitance C_s detached from its single bit line by an access transistor. This transistor is in turn controlled by the word line (see 2.8). The charge stored in

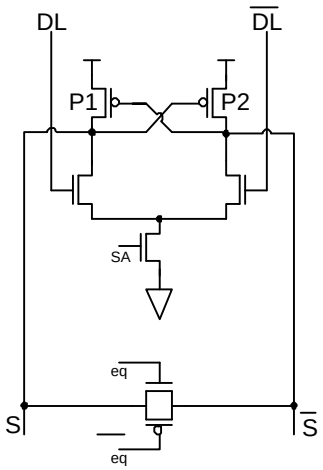


Figure 2.6: Sense amplifier.

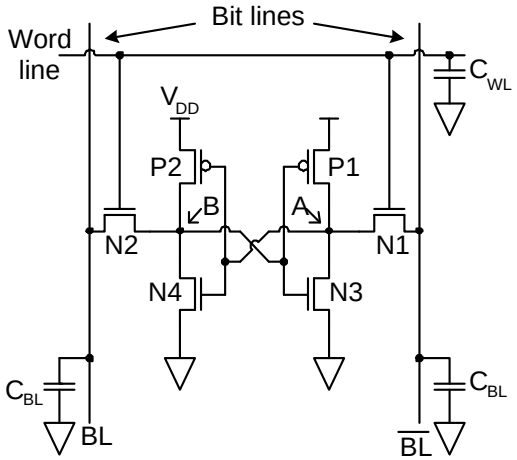


Figure 2.7: Static RAM Cell.

the capacitance determines the logic value of the cell. Since it leaks away over time it is necessary to refresh the state of the cells in regular intervals. For reading the cell, it is connected to the bit line (usually pre-charged to $V_{DD}/2$) by activating the access transistor. The cell capacitance then charges or discharges the bit line producing a small voltage difference V_{swing}:

$$V_{swing} = (V_{MC} - V_{BL}) \cdot \frac{C_s}{C_{BL} + C_s} \tag{2.7}$$

where V_{MC} and V_{BL} are memory and bit-line voltage and C_{BL} is the bit-line capacitance. Sense amplifiers are used to produce a stable logic value from the small voltage difference. Since the charge state of the cell converges to that of the bit line during read, reading is inherently a destructive process. As a consequence sense amps must amplify the signal to full swing *on the bit lines* in order to write the original state back into the cell during read. It is necessary to have one amp per column, because all columns of the selected row are activated and not just the ones read. Writing happens in the same way as in SRAMs by charging the bit lines to the target value and forcing it there during cell activation.

Each row has to be refreshed in regular intervals to ensure data retention. Refresh is similar to reading from a row and can be controlled by external or internal controllers.

2.4.3 ROM

The content of ROMs is already defined during mask fabrication and not changeable afterwards. Two types of ROM arrays exist (see figures 2.9,2.10): the parallel NOR and the series NAND ROM. NOR ROMs are similar in structure to DRAMs: The word lines drive access transistors, that connect the bit

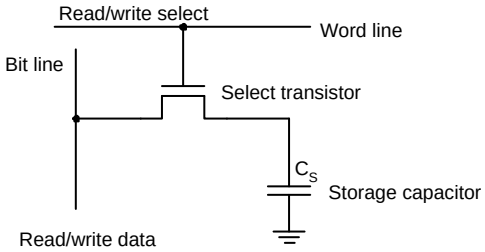


Figure 2.8: Dynamic RAM Cell.

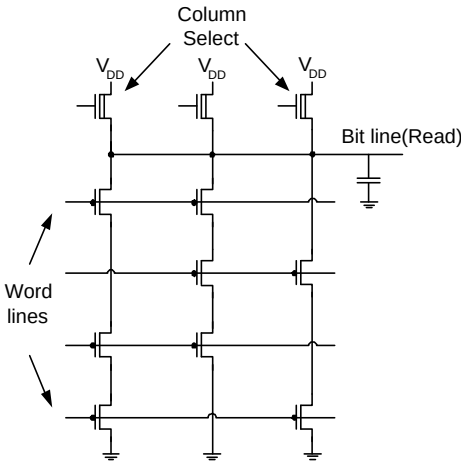


Figure 2.9: NAND ROM Array.

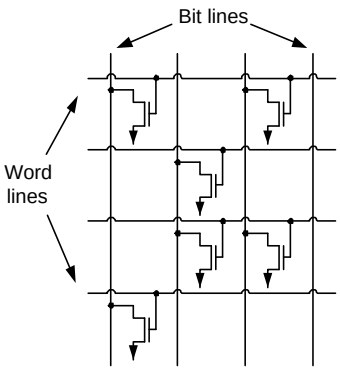


Figure 2.10: NOR ROM Array.

lines to ground. During programming, these transistors are disabled for row/column combinations that contain a zero (or one respectively). Between accesses the bit lines are pre-charged. During read they are then discharged only if a working access transistor is present. In NAND ROMs the columns are intersected by pass transistors that are controlled by the word lines. These transistors are replaced by a permanent connection for row/column combinations that represent a zero (or one). For operation one end of the columns is connected to V_{DD} while the other is connected to ground. During read the pass transistors of the not selected rows are activated, while the one of the selected row, if working, is deactivated. The column can now only charge if no connection to ground exists, i.e. a blocking transistor is present in the row/column.

While NOR arrays are larger than NAND arrays they have the advantage of being generally faster and programmable later in the fabrication process.

2.4.4 Caches

Caches are motivated by a mismatch in speed capability of processors and memories. They are buffers that hold selected information from larger and usually slower ('main') memory. Therefore they represent a compromise between cost effectiveness and performance. The selection of cache contents implemented in different caching strategies is based on the assumptions of spatial and temporal correlation of memory accesses. A cache memory is a special (SRAM) memory architecture:

A direct-mapped cache consists of several memories: a data RAM, a tag RAM and possibly a status RAM. The data RAM contains several *lines*. Each line is a sequence of main memory contents. The position of the line within the data RAM is the *index*. The index of a line usually coincides with the final bits of its main memory address. The index of a certain main memory line is therefore uniquely determined. This does not hold vice versa: several memory lines can be mapped to one index. The tag RAM therefore stores the address of origin of each cache line (minus the index part).

A cache read access therefore consists of the following steps: select the index part of the access address. Read the tag RAM at index. Compare access and tag address to find out wether the cache line is from the right part of the main memory. If the cache line matches ('cache hit'), read the cache line from the data RAM. If it does not match ('cache miss'), remove the current content and read in the correct cache lines from main memory. *N-way set associative caches* consists of N tag and data memories. A specific line can be loaded into any of these. Consequently, all of them must be searched in parallel during a read access.

When introducing redundant information in form of a cache it must be ensured that both, cache and memory stay coherent. This becomes important, whenever the cache is written. The two most common

techniques to handle this situation are write-through and copy-back. In the write-through strategy each write is not only performed to the cache, but also to the main memory. When copy-back is used, data is modified only in the cache. That makes it necessary to copy the cache line back to main memory when it is removed from the cache. This copying must however only be conducted for lines that were changed, or *dirty*. Copy-back caches therefore store a flag for each cache line, indicating whether it is clean or dirty. These flags are contained in the status RAM, which is often merged with the tag RAM.

2.4.5 Register Files

While single registers have the advantage of speed and easy integration into the data path, timing and area scale sub-optimally with the number of registers used. This is the main reason for using dedicated memories as described previously. Registers files are a compromise between dedicated memories and registers. They consist of ensembles of registers that share access lines and the address logic [38]. Modern register files do also employ RAM cell technology. Register files are often used where storage capacity of intermediate size is required (i.e. a couple of words).

Figure 2.11 shows a simple flip-flop based register file of depth (i.e. number of words) 4 and a bit-width of 1. The content is stored in the D-flip-flops visible in the center. As each flip-flop permanently drives its content to the outputs only a multiplexer is needed for the read access. This multiplexer simply selects the desired flip-flops output. With each clock the flip-flop captures the input. Therefore the output must be fed back to the input during idle cycles. Multiplexors select between this output value and new write data for a write access. The multiplexors are controlled by the write enable and the respective decoded write address. The decoded write addresses are supplied by a decoder block. This block is not necessary for the read address because the decoding is performed by the output multiplexer.

As the flip-flops are active even during idle cycles, this type of register files has a considerable standby power. During asynchronous read accesses the circuit is virtually idle. As all flip-flops drive their content at the outputs continuously, the only active parts of the circuitry are the output multiplexors. This leads to very small power consumption during read (about one order of magnitude smaller than write in the register file evaluated in chapter 7). The bottom side of this is, that the decoding units of the multiplexors take the majority of the power. These are complicated to model, as the input lines become highly correlated within the circuitry.

2.4.6 Trends in Memory Development

The traditional design goals area, performance and reliability as well as the newer goals like testability and power consumption have led to a high amount of diversity in modern memory designs. Speed optimizations usually target the I/O interface (e.g. access modes), the internal organization (e.g. banking, segmentation) and the sensing speed. Area optimizations concentrate on the compaction of the memory array layout. The main themes of cost reduction are testability (e.g. build-in self-test), redundancy and repair. Low power design tries to reduce the supply voltage while keeping leakage currents at bay. The number of techniques and architectures is so immense, that the subject cannot be exhaustively treated here (e.g. [60, 61, 145, 146, 1, 139]).

Overview articles and recent developments can be found in the following papers: [69, 68] describe circuit technology trends for low-power and low-voltage RAMs. A collection of papers on low-power SRAMs and DRAMs including [69] can be found in [27]. [35] investigate low power techniques for ROMs. Segmentation, special precharge techniques and data encodings are proposed. [62] give an overview of high-speed DRAM architectures, like DDR SDRAM, virtual channel SDRAM, RAMBUS and SLDRAM. [5] analyze the sizing of different decoders for performance and power. [55] compare divided bit line, pulsed word line and isolate bit line SRAM architectures. [28] present ROM modules using four-phase high-speed precharge/discharge dynamic CMOS logic. [2] investigate the problem of bit line leakage currents and propose a compensation scheme. [51] compare different dual-V_T architectures. [149] investigate current mode sense amplifiers and introduce an augmented architecture. [76] suggest a current-mode SRAM writing scheme for low-power. [140] present a state-of-the-art ultra low voltage high-performance embedded DRAM macro.

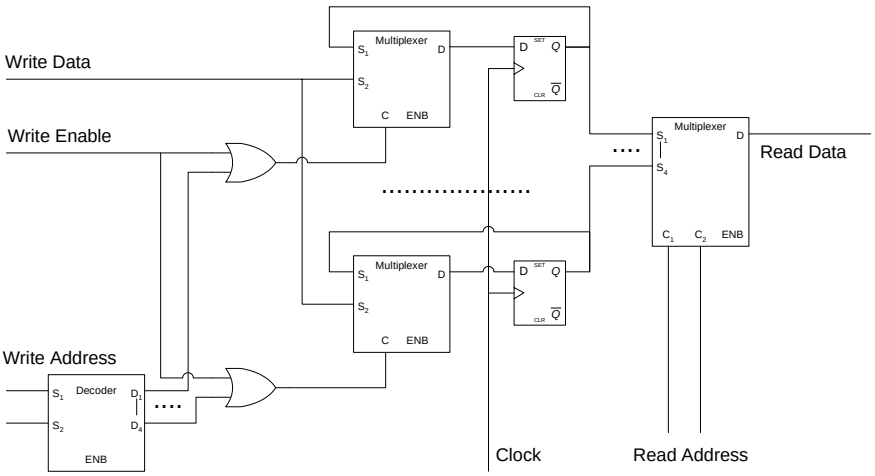


Figure 2.11: Register file with 4 words of 1 bit.

The high number of techniques published make clear that it is potentially very difficult, if not impossible, to find a single power model that can represent all state-of-the-art memories.

2.5 Off-chip versus Embedded Memory

Traditionally memory and logic have been situated on separate chips (off-chip memory). Process technology has developed independently for the two types of circuitry to optimize the performance of the respective functionality. Memory technology focussed on capacitor concepts (DRAM), noise sensitivity and yield optimization, while logic technology emphasized high transistor performance and large numbers of interconnect layers. In recent years very large scale integration has made it possible to integrate memory and logic blocks (*embedded memory*). Designers now have the option to use on and/or off-chip storage.

2.5.1 Off-chip Memories

The advantages of using off-chip memories are [74, 147, 53, 105]:

- *Expansion:* Memory sizes are easily expandable (by replacement with a bigger chip).
- *Test:* Discrete memory chips are easier to test.
- *Yield:* Separating the system in several chips increase overall yield.
- *Technology:* The logic part of system can be fabricated in a pure logic technology.
- *Fabs:* More fabs are available (since no embedded memory process is required).
- *Power:* Memory power is not a contribution to logic chip power. By separating memory and logic the per chip power consumption is reduced. Note that the overall power consumption is increased by such a separation.

Off-chip memories are highly optimized for performance and/or capacity. Only a few different sizes and options are available. Each memory constitutes an independent design. Due to the high market pressure, off-chip memories rarely undergo technology scaling. New technology generations instead bring augmented designs and increased capacity. The designs of different vendors differ significantly. As the power is relatively independent of the actual memory size it is very difficult to generate a size dependent model [24]. Power modeling should therefore be performed on a per instance basis.

2.5.2 Embedded Memory

Several embedded memory technologies exist today constituting different compromises between logic and memory technology. Embedded memory is becoming an increasingly important aspect of design [159] due to the following advantages [74, 147, 53, 105]:

- *Power Consumption:* As large board wire capacitive loads are avoided, it is possible to save several factors of energy.
- *Bandwidth:* By far higher bandwidth and *fill frequency* (frequency with which the complete memory can be filled) are achievable.
- *Sizes:* The memory sizes can be customized to the individual needs.
- *Granularity:* The memories can be better distributed along the circuit. In this way memory can be placed where it is needed.
- *Cost:* The system cost is reduced due to smaller chip count and less pins. On the other hand more expensive packaging may be needed as chip power increases. Furthermore several voltages might be required.
- *Electromagnetic Interference (EMI):* Since less data has to be transferred less EMI is produced resulting into an increase of reliability.

Embedded memories are usually generated by the memory vendors using a memory compiler. This software is fed with the desired size parameters and in turn generates the layout by automatically placing (normally handcrafted) leaf cells. Since memory designs are considered sensitive IP by the vendors, layouts usually are not disseminated. Users are instead supplied with a different type of memory compiler. This 'user compiler' assembles abstract memory views to enable the user to perform the design steps without knowledge of the netlist/layout. Among these views may be power and timing specifications, behavioral simulation models and possibly EDA tool specific abstracts (Synopsys DB files, LEF, DEF etc.). In the design houses design space exploration as well as the complete design to layout is performed solely based on these abstract memory views. Only at the waver fab is the memory layout supplied by the memory vendor integrated into the layout of the surrounding logic. Power values delivered by user compilers are often only rough specs. The inaccuracy of such specs is obscured by the fact, that more accurate information is at no stage of the design available to the users. This thesis therefore presents a methodology for the generation of accurate power models for embedded memories.

2.6 Memory Optimization

Memories can have a major impact on cost and performance of embedded systems. With the fast development of processing units, the processor-memory performance gap still widens. This puts memory more and more into the focus of system optimization efforts. Traditionally, optimization was based on a single processor centric multi purpose system view (e.g. classical PC architecture). This has brought forth the cache based memory hierarchy. A cache hierarchy is however only a compromise dealing with the fact that the application and workload of the system varies and is not known in advance. In the embedded systems domain much more can be gained: As the application running on the system is known at design time, the memory subsystem can be tailored to suit the expected profile of code and data, reducing the costs. As a consequence embedded systems contain a wide variety of customized non-traditional memory architectures. Not only the memory system is part of the optimization process. As the code of the application is available and can be modified, it is also a target of memory cost optimizations. The combination of both targets, the memory subsystem architecture and the application, makes available an optimization space with huge potential in cost savings. [21] for example report a factor 8.0 saving in the energy consumption of an MPEG 4 video motion estimation kernel.

Optimizing an application and its implementation for low memory power is a problem far too complex to be solved with a single isolated technique. A variety of approaches has therefore evolved which focus on

different aspects and are often used in combination. An integrated low-power memory subsystem design methodology is only just evolving [24].

Platform-independent source to source transformations usually form the first step in memory power aware design. These transformations have the following goals:

1. *Elimination of unnecessary storage.* The code of many applications contains data transfer statements (e.g. assignments) that are not needed for its functionality. These assignments inflate the optimization problem and are therefore removed.

2. *Increase of data locality.* Similar to hardware goods the reduction of time between production and consumption of data results in reduced storage cost (temporal locality). Increasing spatial and spatiotemporal locality enables more effective memory architectures (e.g. caches).

3. *Increase of Regularity.* Increasing the regularity of the code fascilitates subsequent optimizations. More specifically, balancing the rate of production and consumption of data values makes it possible to fold these actions and thus heavily increase temporal locality.

4. *Unearthing of parallelism.* Parallelism that is inherently in the code is often hidden, e.g. by loop structures. Increasing the visible parallelism increases the optimization potential of later steps.

5. *Increase of data reuse.* When using a memory hierarchy, introducing value copies into the code explicitly can prepare the mapping of such copies onto the levels of the hierarchy.

The above goals are usually pursued by loop and dataflow transformations. Most of these transformations originate from the domain of (parallelizing) compilers. Platform-independent optimizations usually do not reduce the storage cost themselves. Instead they increase the potential of the subsequent platform-dependent optimizations.

Platform dependent-optimizations seek to find the optimal memory architecture in terms of power, area and performance for the given application. This means deciding on how many memories to use of which type (allocation) and also where to store which data (binding) and in which order to perform the accesses (scheduling). Due to its complexity, the problem is broken into sub-problems many of which are solved heuristically. A short overview of standard problems is given in the following (see [101] for a more detailed discussion).

In order to reduce the area used for registers, register allocation tries to minimize the amount of registers needed to store a given set of variables. Register allocation begins with a lifetime analysis of the variables. Based on the lifetimes a conflict graph can be built, containing a conflict for every variable pair with overlapping lifetimes. An optimal allocation is now a coloring of the graph with a minimal chromatic number. Alternatively the problem can be cast into a network flow formulation [158]. As this is a standard problem, a variety of approaches has evolved. [133] gives a good overview of existing techniques.

When separate registers are replaced by register files or memory modules, interconnect costs are reduced. On the other hand the problem formulation above is in this case complicated by the fact that the number of parallel accesses is now bounded by the available memory ports. Several ILP based approaches have been proposed to solve this problem. So far, however, the considered costs only consist of the plain numbers of ports and memories and do not include power considerations.

When two data values are accessed in the same cycle, they can be mapped onto the same memory only, if it has multiple ports. As multi-port memories are expensive, the scheduling of accesses has great impact on the system cost. To obtain a global cost optimum it would be necessary to consider scheduling and memory allocation and assignment simultaneously. Unfortunately, this is intractable. Traditional design approaches either assume that the allocation is fixed before scheduling or they perform the allocation after the scheduling. Recognizing the importance of memory accesses more recent approaches try to minimize the memory bandwidth during the scheduling phase. The most elaborate technique is proposed by the IMEC institute [156]: a pre-scheduling step is performed on the memory accesses. This scheduling produces a partial ordering of the memory accesses minimizing the number parallel memory transfers. This serves as input both, to the data path scheduling and the memory allocation and assignment.

For a given schedule the structure of the memory subsystem is defined by allocation and assignment. Allocation determines the number, type and port configuration used, while the assignment decides where data values are mapped. Both aspects influence each other and the cost of the solution. Memories containing more data are bigger and therefore consume more area and energy. They have a tendency to waste part of their capacity, as words of different bit-widths are stored. Monolithic approaches with one big memory are hence sub-optimal in area as well as power. Scattering the data over many memories on the other hand increases the relative cost of peripheral memory logic (e.g. address decoders) and interconnect, leading again to increased cost. The optimum architecture for power as well as area therefore lies in between these extremes. The minima are unfortunately usually not the same. A number of approaches has been suggested to solve the allocation and assignment problem for different specifications and target architectures. Many of these use or could use energy consumption models as part of their cost functions.

The memory packing problem concerns the question of how to realize a logical memory (as seen be the designer) by physical memories under given constraints. Here it may be necessary to split logical memories, when no physical memories with the required size and performance are available. In the MemPacker utility this problem is solved by a branch-and-bound approach [73]. The MeSA algorithm uses a clustering approach, grouping behavioral arrays into the same array [115]. [125] extend this idea to two dimensional clustering, mapping several values onto separate bits of the same memory word. The approach uses simulated annealing for optimization. [12] suggest application specific memories for the power optimization of embedded systems. They apply the analytical power model of [72] to optimally slice SRAM memories. A later publication extends this approach to the integration of a complete back-end flow [11]. [100] suggests customizing the number of memory banks to the application. This involves solving the problem of assigning arrays to banks. The HIMALAIA tool developped at IMEC combines the pre-scheduling described above and an automated memory allocation and assignment step [24, 142]. [129] presents an alternative approach. The APEX algorithm described in [47] first performs a clustering of access patterns and then explores memory architectures by mapping the patterns heuristically. For assessing the power cost, a cache/main memory energy ratio is used. [126, 127] and [154, 153] investigate memory optimizations and data-flow transformations for of MAP Turbo Coders. Multiprocessor systems are the target of [88]. Integer linear programming is used here to minimize the access costs. The increasing importance of leakage is recognized by the approach of [67]. They map basic blocks into instruction memories with different supply and threshold voltages applying a greedy heuristic based on execution count and size. They present their own analytical model (in essence scaling with $\sqrt{\#words}$). The memory subsystem power minimization problem gains another dimension, when power/performance tradeoffs are considered. In a system consisting of several tasks this makes it possible to perform an energy aware cycle budgeting. [22] presents a tool based on HIMALAIA, that produces cycle budget vs. power curves.

As was mentioned earlier, the size of memories has impact on both, area and power consumption. An effective way to minimize the memory size requirement is to map several data values to the same address space. This in-place mapping can be used between arrays or among values in one array. Of course a liveness analysis is necessary to prevent conflicts. This analysis is far from trivial for arrays. For a more in-depth discussion of this technique see [24].

Bus encoding is a general power technique trying to minimize the switching activity on system busses. Because large busses usually occur in the memory subsystem, bus encoding is often used in the memory context. An overview of techniques can be found in [99,101]. Another possibility to reduce the address bus switching activity is to change the data organization in memory. [98] compare row-major, column-major and tile-based accesses and find that tile-based organization leads to a 63% reduction in transition count.

In more traditional multi-purpose memory hierarchies, cache development is still the main driving force. Recent approaches that consider the energy consumption are the following: [42] explore sub-banking, multiple-line buffers and bit-line segmentation. They perform SPICE simulations. [63] use a simplified version of the analytical power model of [134] to evaluate way-predictive caches in comparison to phased caches. [9] suggest L0 caches and evaluate their results using the analytical power model of [151]. [103,104] apply an analytical model proposed by [128] to show the efficiency of a mixed hardware/software technique to reduce the required number of tag bits in loops. [64] present a technique to reduce the number of cache tag comparisons within loops. [43,44] try to optimize a cache based system by considering cache and power consumption simultaneously. A macro-model is built for the number of cache misses to avoid repeated

simulation. Apart from cache tradeoffs and innovative cache architectures a cache-friendly layout of data in memory and the application of small dedicated scratch pad memories have been proposed [98].

The preceding paragraphs have documented that memory (power) optimizations can drastically reduce the power consumption of embedded systems. They are therefore able to push the boundary of cost and feasibility. The savings are achieved by one or several of a multitude of techniques. Many of these require models for the power consumption of memories. Other, not yet power aware optimizations could be augmented to use such models for even bigger savings. Supporting tools and integrated methodologies for the memory optimization problem are currently evolving. These tools, techniques and methods will be limited by the availability and accuracy of memory power models.

3 Modeling

This chapter introduces the notion of modeling and explains basic statistical concepts used throughout this thesis. It closes with a discussion of the requirements on the models and modeling process for application in the context described in chapter 1.

3.1 Introduction

Models are a means to (approximately) describe specific aspects of a complex reality. They are abstractions of *prototypes*, defined subsets of the real world. Mathematical models describe quantitative properties of the prototypes and their relationships by mathematical equations. They are clearly defined and easily communicated as well as analyzed using mathematical formalisms. When abstraction is applied or the underlying mechanisms and relationships are not fully known, models will be subject to errors. These may concern the form and/or the numerical values used. Statistical techniques should accompany the model building (*statistical model building*) in this case to quantify expected errors.

Different approaches to model generation may be taken: the *conceptual approach*, also known as model synthesis, is based purely on the theory of the area of application. The model arises from the theory more or less naturally, while remaining degrees of freedom may be exploited to achieve favorable model structures [45]. In the *empirical approach*, as the other extreme, the model is generated from an analysis of empirical data without exploiting theoretical knowledge of underlying relationships. The *eclectic approach* seeks to combine conceptual and empirical elements.

Model building usually consists of the following stages:

1. *Data Abstraction.* In many cases the observed quantities do not serve as model variables directly. They merely undergo a transformation step to obtain a derived, more suitable, variable (e.g. computation of Hamming distance from vector pairs). Data abstraction consists of the identification of potential variables and their relationship to observables.
2. *Data Acquisition.* Data from the field of application is needed for the parameter estimation and the validation. The availability, controllability and cost of such data has a large impact on the modeling process.
3. *Model Identification.* The selection of a suitable mathematical representation. This representation describes the principle relationships and still contains abstract parameters instead of concrete numerical values.
4. *Fitting by estimation.* Appropriate values for the parameters are chosen to minimize a selected error measure. By this parameter estimation the model is fitted to the data. This stage moves from the general to the specific numerical form.
5. *Validation.* The process of comparison of the model with the observed world is called validation. Statistical techniques are use for the assessment of the model's properties, its consistency and suitability.
6. *Application.* Application is the ultimate purpose for which the model is required. Pragmatic considerations influence all other stages.
7. *Iteration.* The concept of statistical modeling contains the computation and analysis of errors throughout the modeling process. Should these errors proof to be unacceptably high at any stage, the modeling process must be re-iterated.

It has to be noted that these stages are usually not adopted in strict sequential order.

3.2 Statistical Basics

The modeling approaches presented in this thesis make extensive use of statistical techniques. This section gives a short introduction into the fundamental principles and terminology used.

3.2.1 Random Variables

Statistical models are assumed to be subject to random disturbances. Random quantities such as these disturbances are described by random variables. A formal mathematical definition of random variables and the concept of "probability" is technically complicated (for in-depth discussion see [46, 102]). The intuitive meaning in this reduced context is however quite straight-forward: a random variable V is a variable that can randomly take on any value v in its domain (write: $V = v$). The probability is the likeliness of a specific outcome (write: $P(V = v)$). Let now V be a continuous random variable. Then $F : \mathbb{R} \longrightarrow [0, 1], F(x) = P(V \leq v)$, i.e. the probability that V is less or equal v, is the *distribution function* and $f(x) = \frac{\partial F(x)}{\partial x}$ is the *probability density function* (in short pdf). The *mean value* or *expectation* is then defined as:

$$E(V) := \int_{v:f(v)>0} vf(v) \tag{3.1}$$

The *variance* is given as:

$$\sigma^2 = var(V) = E((V - E(V))^2) \tag{3.2}$$

V is called *normally distributed* if:

$$f(v) = \frac{1}{\sigma\sqrt{2\pi}} e^{(v-E(v))^2/2\sigma^2} \tag{3.3}$$

Let V_i be a number of n independent variables. Then the *sample mean* is:

$$\bar{v} := \frac{1}{n} \sum_i v_i \tag{3.4}$$

Likewise the *sample variance*:

$$s^2 := \frac{1}{n-1} \sum_i (v_i - \bar{v})^2 \tag{3.5}$$

It can be shown that if the V_i are independent random variables with the same mean $E(V)$ and variance σ^2 and $n \to \infty$ the sample mean and variance approximates the distribution mean and variance, i.e.

$$E(\bar{v}) = E(V) \tag{3.6}$$

$$E(s^2) = \sigma^2 \tag{3.7}$$

Furthermore if the V_i are normally distributed or n large, then $\bar{v}$ is (approximately) normally distributed.

3.2.2 Experimental Designs

An experimental design is a subset of combinations of values of the random variables used for data acquisition. Let $levels(x_i)$ the set of all values variable x_i takes in a set of experiments. An experimental design is then

$$design \subseteq \prod_i levels(x_i) \tag{3.8}$$

where $\prod$ stands for the cartesian product. In case of equality, i.e. $design = \prod_i levels(x_i)$, the design is called *factorial* and the x_i *factors*.

Example 1. Let the variables be x_1 and x_2. Then $D = \{(1,2),(1,3),(4,2),(4,3)\}$ is a factorial design with $levels(x_1) = \{1,4\}$ and $levels(x_2) = \{2,3\}$.

Obviously the choice of the experimental design has a high impact on the quality of the resulting model. However, even if the experiments are conducted specifically for the modeling, some of the variables may not be directly under control. This problem can occur for example due to the data abstraction (see section 5.1).

3.2.3 Regression

Linear regression is a common statistical method for model fitting. Consider the following situation: a *dependent variable* or *response* y, is to be modeled by a linear relationship that involves the k *independent* or *regressor* variables x_i and $k+1$ unknown parameters β_i. Let $\boldsymbol{y}$ and $\boldsymbol{x_i}$ be the vectors of the n observed values of the respective variables. With $\boldsymbol{\varepsilon}$ the vector of errors:

$$\boldsymbol{y} = \left(\sum_{i=1}^{k} \boldsymbol{x_i} \cdot \beta_i\right) + \beta_0 + \boldsymbol{\varepsilon} \tag{3.9}$$

Model fitting now is the task of estimating appropriate *regression coefficients* $\hat{\beta}_i$ to minimize the errors or *residuals* $\boldsymbol{\varepsilon} = \boldsymbol{y} - \hat{\boldsymbol{y}}$ (where $\hat{\boldsymbol{y}}$ is obtained by substituting $\hat{\beta}_i$ in eqn. 3.9). Let $\mathbf{X}$ be the matrix formed by the vertical concatenation n-element 1-vector and the $\mathbf{x}_i$ and $\boldsymbol{\beta}$ be the vector of the β_i. Then the matrix formulation is:

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{3.10}$$

The *residual* or *error sum of squares* is defined as:

$$SSE = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = (\boldsymbol{y} - \boldsymbol{X\beta})'(\boldsymbol{y} - \boldsymbol{X\beta}) \tag{3.11}$$

The so called *least squares estimate* of $\hat{\boldsymbol{\beta}}$ which has minimum SSE can be obtained as follows: Thinking of $\mathbf{X}$ as a linear projection matrix $\mathbf{X}\boldsymbol{\beta}$ defines a subspace of $\mathbb{R}^n$ (*expectation plain* [7]). The closest point $\hat{\boldsymbol{\beta}}$ to $\boldsymbol{y}$ has the property that the remaining error vector is orthogonal to this expectation plain, i.e.:

$$\mathbf{X}'\boldsymbol{\varepsilon} = \mathbf{X}'(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}) = \mathbf{0} \tag{3.12}$$

The least squares estimate follows directly:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{y} \tag{3.13}$$

The least squares estimate can be easily calculated using computers. It also has a number of desirable statistical properties given the standard assumptions on all errors. Let ε the random variable of the errors, with ε_i arbitrary observations:

1. $E(\varepsilon) = 0$.
2. $var(\varepsilon) = const = \sigma^2$.
3. The ε_i are independent.
4. The ε_i are normally distributed.
5. The random variable ε and the regressor variable x_i do not influence each other.

See [7], pp. 23-26 for a discussion of these assumptions. Taking them as true the following properties can be guaranteed:

1. The least squares estimator $\hat{\beta}$ is normally distributed.
2. $E(\hat{\beta}) = \beta$. That is to say the estimator is unbiased.

3. $\hat{\beta}$ has the highest probability of all estimators (maximum likelihood).
4. Even if the errors are not normally distributed it follows from the Gauss-Markov theorem that $\hat{\beta}$ has the smallest variance of all linear unbiased estimators.

Other types of regression
The assumptions for linear regression can also be written as:

$$\boldsymbol{\eta} = \left(\sum_{i=1}^{k} \boldsymbol{x_i} \cdot \beta_i \right) + \beta_0 + \boldsymbol{\varepsilon} \tag{3.14}$$

with

1. y_i normally distributed with mean μ_i and variance σ^2
2. $\mu_i = \eta_i$.

If any single aspect of this formulation is changed (e.g. a different distribution is assumed), a *generalized linear model* results. *Robust regression* is a technique specifically addressing distributions with a high amount of outliers ("heavy tailed" distributions). *Ridge Regression* is used for models with interdependent regressor variables. *Linearizable models* are models that are not linear, but that can be transformed into a linear model.

Nonlinear regression models are of the general form $\boldsymbol{y} = f(\mathbf{X}, \boldsymbol{\theta}) + \boldsymbol{z}$, where at least one of the derivatives of the expectation function with respect to the parameters $\boldsymbol{\theta}$ depends on at least one of the parameters.

3.2.4 Interval Estimation

When using regression models for prediction it is often of interest what errors to expect. Regression theory gives the answer in terms of confidence intervals, i.e. intervals around the point estimate $\hat{y}_0$ in wich the true response lies with $1 - \alpha$ probability. The $1 - \alpha$ *prediction interval* at the point $\boldsymbol{x}_0$ can be computed as:

$$\hat{y}_0 \pm t_{\frac{\alpha}{2}, n-k} \cdot \sqrt{\frac{SSE}{n-k} \cdot \left(1 + \boldsymbol{x_0}'(\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{x_0}\right)} \tag{3.15}$$

Here $t_{\frac{\alpha}{2}, n-k}$ is the upper $\alpha/2$ quantile of the Students t-distribution with $n - k$ degrees of freedom.

3.2.5 Significance test

During model building and evaluation (e.g. during variable selection, see below) it can be necessary to assess the statistical significance of a term $\beta_{t+1} \cdot x_{t+1}$ (i.e. its "importance") in a given regression model. This statistical significance is expressed by the level of confidence in the associated coefficient β_{t+1} being different from zero. The level of confidence is determined by testing the following statistical hypothesis:

$$H_0 : \quad \beta_{t+1} = 0 \tag{3.16}$$

The probability of rejecting this hypothesis expresses the confidence in β_{t+1} differing from zero. The level of confidence is of course dependent on the rest of the model. Let therefore Φ_1 an arbitrary model with t variables. Let Φ_2 the same model expanded by $\beta_{t+1} \cdot x_{t+1}$. Under these circumstances the confidence is determined using analysis of variance methods through the "partial F-test":

$$F_0 = \frac{SSR_2 - SSR_1}{MSE_2} \tag{3.17}$$

where SSR_i is the regression sum of square and MSE_i is the mean square error for model Φ_i. F_0 follows the Fisher F-distribution, so that we can reject H_0 with probability $1 - p$ if $F_0 > F_{p,1,n-t}$. The same partial F-test can be applied either when considering to include new variables into existing models or when considering to remove variables from models [92].

3.2.6 Interpolation

Interpolation is another parameter estimation technique. In contrast to regression it is a locally applied technique: the model response is not determined by the complete data set, but only by the data points in the neighborhood of the desired point. Less complex models than for regression are generally used. This has mainly two reasons: a) an ill specified model has less impact and b) less data points are available to fix the parameters of the model. When smoothness is not required (e.g. spline interpolation), *linear interpolation* is usually employed. Linear interpolation can be formulated as follows:

Let $\boldsymbol{p_i}$ the i-th row of matrix $\mathbf{X}$, i.e. the x vector of the point of the i-th observation y_i and $\boldsymbol{p}$ the point to be predicted. Let N denote the set of indices of a given set of adjacent points. Then the goal is to find a function:

$$\hat{y} = f(\{\boldsymbol{p_i}, i \in N\}) \tag{3.18}$$

The main problem here is to identify a *meaningful* set of adjacent points so that f becomes simple, especially when the dimensionality of the $\boldsymbol{p_i}$, i.e. the number of variables, is high. Note that since this neighbor identification has to be performed at model application, it is a performance critical step. Practical solutions for this issue pose restrictions on the $\boldsymbol{p_i}$.

For evaluation purposes a state of the art linear interpolation algorithm is adapted [119]. To tackle the neighbor identification problem this algorithm is restricted to factorial designs. Based on that simplifying restriction appropriate adjacent points can be found as follows:

For every one of k factors x_i with value $\tilde{x_i}$ find the adjacent levels, i.e. $l_i \in levels(x_i)$ (lower) with $\forall m \in levels(x_i) : m \leq l_i \vee m > \tilde{x_i}$ and $u_i \in levels(x_i)$ (upper) with $\forall m \in levels(x_i) : m < \tilde{x_i} \vee m \geq u_i$. When the sets $levels(x_i)$ are ordered, l_i and u_i can be obtained by simple search algorithms. All possible combinations of l_i and u_i for every factor now represent points $\boldsymbol{p_j}$ that form a hypercube around $\boldsymbol{x}$ (see figure 3.1). Finding the appropriate points among the $\boldsymbol{p_j}$ means finding a hyper-tetrahedra among the $\boldsymbol{p_j}$ adequately surrounding $\boldsymbol{x}$. Rovatti et al. show that this problem can be solved by sorting: First, the values of $\tilde{x_i}$ are normalized with respect to l_i and u_i ($xnorm_i = \frac{\tilde{x_i} - l_i}{u_i - l_i}$). Then the $xnorm_i$ are sorted. Eventually the set of points is generated as follows: start with the "lower left corner", e.g. $(l_1, \cdots, l_k)$. For the next point change the dimension i with lowest $xnorm_i$ to u_i, e.g. $(l_1, \cdots, u_i, \cdots, l_k)$. Now continue with the next bigger $xnorm_j$ and so on. Repeat until $k+1$ points are generated.

Example 2. Consider the following three-dimensional factorial interpolation table:

		$x_1 = 0$			$x_1 = 4$		
		x_2					
		1	3	5	1	3	5
x_3	0	0.2	0.4	0.6	1.2	1.4	1.6
	3	0.3	0.5	0.7	1.3	1.5	1.7
	6	0.4	0.6	0.8	1.4	1.6	1.8
	9	0.5	0.7	0.9	1.5	1.7	1.9

Let $x = (3, 4, 5)^t$ (see also figure 3.1). Then $l_1 = 0, u_1 = 4, l_2 = 3, u_2 = 5, l_3 = 3$ and $u_3 = 6$. The normalization produces $xnorm_1 = \frac{3}{4}, xnorm_2 = \frac{1}{2}$ and $xnorm_3 = \frac{2}{3}$. The first point generated is then

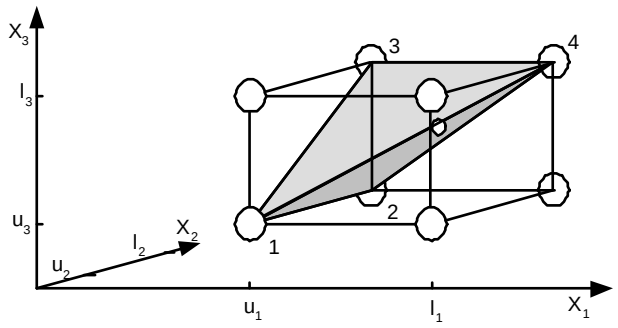


Figure 3.1: Identification of adjacent points for interpolation.

the left lower corner: $(0,3,3)^t$. Now, as $xnorm_2$ is the smallest normalized value, dimension 2 is changed first: $(0,5,3)^t$. Now dimension 3 contains the next smaller $xnorm$: $(0,5,6)^t$. Finally dimension 1 is updated: $(4,5,6)^t$. The points generated by this procedure subsequently become the reference points for the interpolation.

Since it is based on sorting, this approach has $n \cdot \log n$ complexity in the number of dimensions. Rovatti et al. prove that this is the optimum [119], i.e. no faster algorithm is possible. For this thesis the algorithm was implemented in C, using binary search for the identification of adjacent levels and quicksort for the sorting.

The previous paragraphs have shown, that only by restricting the allowed designs the multi-dimensional interpolation problem becomes efficiently solvable. For the practical application, however, this restriction is problematic in several aspects:

1. A varying grid size cannot be realized effectively: if a high density of levels (fine grid) of a variable x_i is necessary for specific levels of the remaining variables, then grid points must be introduced for *all* of their levels. Variation of grid size is necessary to stabilize accuracy over the model space.

2. The number of sample points can only be chosen in large steps. This is a direct consequence of 1. Let k_i be the number of levels of variable x_i then in-/decreasing the number of levels in x_j by one changes the number of points by $\prod_{i \neq j} k_i$.

3. Existing data can only be reused if it fits into a grid configuration.

3.2.7 Error measures

For quantitative evaluation of models the following error measures are employed:

1. *Mean Square Error* (MSE).
   The MSE obtained from the regression above by setting $MSE = \frac{SSE}{n-2}$ is an unbiased estimator of the variance of the regression coefficients $\hat{\sigma} := MSE$. Should the model be ill-specified, however, the MSE is inflated by the squared bias.

2. *Root Means Square Error (RMS).*
   The square root of the MSE is an estimator for the variance var, i.e. the squared standard deviation. For the sake of intuition the RMS error is usually given as fraction of the mean response $\frac{RMS}{\bar{y}}$. This approach will be followed throughout this thesis.

3. *The coefficient of multiple determination* R^2.
   Let the *observation sum of squares* $Syy = \sum_{i=1}^{n}(y_i - \bar{y})^2$. Then R^2 is defined as: $R^2 = 1 - \frac{SSE}{Syy}$. It expresses how much of the variability in y can be explained by the (linear) regression model. R^2 has to be interpreted with caution as it is merely a measure of linearity and not of fit [45]. Thus a large value does not necessarily mean that the regression model will be an accurate predictor [92].

4. *Max Absolute Relative Error* (XARE).
   The $XARE = max(abs(\varepsilon_i/y_i))$ is the worst case relative error.

5. *Mean Absolute Relative Error* (MARE).
   The $MARE = \frac{1}{n}\sum_{i=1}^{n} abs(\varepsilon_i/y_i)$ is an estimator for the expected relative error of a randomly drawn instance.

6. *Mean Relative Error* (MRE).
   The $MRE = \frac{1}{n}\sum_{i=1}^{n}(\varepsilon_i/y_i)$ is an estimator for the expected relative error of the sum of a large number of randomly drawn instances.

3.2.8 Cross Validation

During model generation the models and their parameters are specifically chosen to minimize selected errors with respect to a set of observations. The errors of the resulting model with respect to these observations will therefore tend to underestimate errors encountered with other data. To address this issue the suggestion of Hjorth et al. [57] is followed here: In a technique called *cross validation* the total set of data is split into a *regression set* and a *validation set*. Only the regression set is used for the model building process, while the error measures are computed for both sets separately. The regression set errors give information about the ability of the model building process to fit a model to the observed data (regression set). The validation set errors document the predictive performance. If both errors differ significantly this is an indicator for the regression set being either to small or ill chosen.

3.3 Requirements

For rating and comparison of modeling methods, both existing and newly suggested, the requirements need to be clearly stated. For reaching the eventual goal of supplying memory models to high level power estimation and optimization two key demands have to be met: the models must be suitable for application at high levels of abstraction and the modeling procedure must be widely applicable. More concrete the requirements can be divided into those referring to the models and those referring to the modeling procedure itself.

3.3.1 Requirements on the models

1. *Accuracy.* The predictive quality, expressed in the measures described above, is the ultimate goal. Model building often has to trade-off between absolute and relative accuracy: While absolute accuracy is important for comparing the memory subsystem with other components, relative accuracy is valuable in evaluating different memory architectures.

   From modeling techniques for other circuit aspects at RT and higher levels maximum average errors in the area of 20% are reported for size scalable models [49, 15, 155, 89]. The same accuracy is therefore required for memory power models.

2. *Speed.* The time consumed for calculating the model response must be reasonable in the context of the surrounding application. The impact highly depends on the number of model evaluations. Data pattern dependent models for example must be evaluated for each pattern. Data independent models in contrast only have to be re-evaluated on architecture changes. Optimization algorithms often exhibit a high iteration count, thus putting more weight on the speed requirement.

3. *Abstraction.* At high levels of design, e.g. behavioral level, circuit descriptions are abstract. Timing is for example only loosely fixed (i.e. only in terms of algorithm inherent precedence and designer specified constraints). Models on these high levels must consequently cope with abstract information about their environment. They must also be abstract enough themselves to guarantee sufficient speed.

4. *Flexibility.* On-chip memories have a high flexibility in size and available options. As a consequence memory parameters have become a dimension of design. To support high-level estimation and optimization, memory models must adapt quickly to varying parameters. This requirement differs from traditional design flows were (memory) power estimation is only conducted after the definition of a fixed architecture.

5. *Analytical form.* A model given in analytical form, i.e. as a set of closed equations, is advantageous in two respects: a) Legibility. An analysis of trends and relationships is facilitated. b) Optimization. Analytical optimization techniques are only applicable on closed form representations. These techniques however often excel purely numerical approaches.

6. *IP protection.* Memory designs are highly sensible intellectual property. Models must therefore not allow a reverse-engineering of IP.

3.3.2 Requirements on the Modeling Process

These requirements are often underrated. For the goal of this work however, the promotion of memory power modeling, they play a central role.

1. *IP protection.* As mentioned above IP protection is of prime importance. This does not only apply to the resulting models but also to the modeling procedure itself. Since in-depth information about the memory designs is required for the modeling, memory vendors can only accept it to be conducted by trusted instances. Often that means it can only be done by the vendor himself. Modeling approaches must respect the need for IP protection, or they will not be used in practice.

2. *Cost.* Modeling cost is the second major hurdle in the spread of memory power modeling methodologies. As mentioned above modeling must most often be conducted be the vendor. To be adopted, methodologies must keep the cost for vendors as low as possible. Main cost factors are manpower, modeling competence, third party licenses and compute power.

3. *Speed.* The time taken for modeling influences the overall costs and the "time to model". It must be reasonable in the context of the complete flow. Trading off modeling speed for model application speed can sometimes be useful, since in contrast to model application modeling is only performed once.

4. *Fit into Flow.* A natural fit into the memory generation flow reduces the organizational complexity and the effort needed.

3.3.3 Specific Requirements of Memory Power Modeling

The nature of the application, i.e. the modeling of the power consumption of memories, has influence on the specific modeling capabilities required. The structures of memory circuits (cf. section 2.2) combined with the physical nature of power consumption (summarized in section 2.1) motivate specific requirements on models. The issues mentioned in the following can be also found in the evaluation examples (see section 7):

1. *Multi-dimensionality.* The number of relevant memory parameters changes with the type of design. It can well exceed the three to four parameters usually found in data path power models.

2. *Additivity.* The currents flowing through all gates of the memory circuit add up to form the total power consumption. Furthermore the estimation error can be seen as additive. These two observations motivate an additive model structure.

3. *Continuity.* The influence of memory parameters on the power consumption can be approximated as quasi-continuous over wide ranges. This is an important precondition for an effective approximation by functions (see Taylor approximations). For size related parameters assuming continuity is sensible as small changes in the parameters should result in small changes in the design. Sporadic jump discontinuities may however occur, e.g. due to changes of the memory organization at certain size boundaries. These cases must be detected and modeled. Furthermore size parameters are often quantized, i.e. they are rounded up to the multiples of a factor. Restricting the parameter values to these 'real' values solves this issue. Configuration parameters are usually of non-continuous influence, but have only few possible values. It is therefore sensible to build a sub-model per option (cf. section 5.1). Data parameters usually consist of bit-vectors or bit-vector pairs. Abstraction function like the Hamming distance are used to transform them onto a quasi-continuous scale (see section 5.1).

4. *Nonlinearity.* Nonlinearities in the parameters occur in memories in manifold ways: Firstly the influence of the supply voltage on the power is often assumed to be quadratic following equation 2.3. This however is not fully correct, as the switched capacitance also varies with V_{DD} due to timing variations. Furthermore eqn. 2.3 assumes full swing. Yet memories do often contain reduced swing logic. Nonlinearities can furthermore be found in the sense amplifiers, which often work in current mode and in the address decoding logic that has a logarithmic (or exponential) component by its nature (cf. [151]). Nonlinearity is often neglected to make models computationally more tractable. This thesis advocates that significant accuracy is sacrificed by this.

Model representations have a considerable impact on the performance with respect to the above requirements. They are discussed in section 4.5. Standards for the realization of these representations in software are considered an implementation aspect and therefore discussed in section 6.3.1.

4 Related Work

This chapter discusses modeling concepts and related work in memory power modeling. For the sake of clarity modeling techniques are divided into conceptual, eclectic and empirical.

4.1 Conceptual Modeling

In conceptual approaches the structure of the modeled instances is assumed to be completely transparent. Models are derived by abstraction from the detailed implementation. The lowest degree of abstraction in conceptual modeling constitutes complete net-list extraction. In this automated technique a net-list of capacitances, transistors and resistors and inductances is computed from the geometry of the layout. The resulting circuit level models can be evaluated using circuit level simulators like SPICE. They are the most accurate simulation models available (errors within several percent compared to measurements). Unfortunately complete extraction and simulation is only feasible for very small memory instances.

A consequent step towards more abstract models is made by critical path models [141]: the circuit level net list is reduced to contain only the active parts plus a set of dummy cells that represented the collected capacitances of the inactive parts. This approach is quite effective since memories consist of a high repetition of only few basic blocks. Memory compilers that assemble memory layouts from a set of leaf-cells can easily be modified to generate such models [97]. Critical path models are the most detailed simulation models available for all memory sizes. They are therefore commonly used at providers' sites for timing and functionality checks. While accuracies within a few percent of full extracted net-lists can be achieved, the main disadvantage is computational complexity: simulation takes from minutes to hours per memory cycle. [25] report a time requirement of 30 CPU hours for the delay vs. power optimization of an SRAM through iterative HSPICE simulation. Another drawback of critical path models is that scaling the memory instance involves re-simulation.

A distinct advantage of circuit level models is their capability to reflect leakage effects (see section 2.1). More abstract models are based on an analysis of the switched capacitance, i.e. $N \cdot C$, where N is the (average) number of a switches per cycle and C the capacitance. With their restriction to capacitive currents these models (called *analytical models* in the following) ignore leakage. The model of [85] uses the NMOS transistor gate capacitance C_{tr} as technology dependent base unit. The power consumption of an SRAM memory cell is then for example computed as:

$$P_{memcell} = \frac{2^k}{2}(c_{int} \cdot l_{column} + 2^{n-k} \cdot C_{tr}) \cdot V_{DD} \cdot V_{swing} \tag{4.1}$$

with 2^n the memory capacity, 2^k the number of columns, c_{int} the per unit interconnect capacitance and l_{column} the length of a column. [39] use a more detailed capacitance model. They give the example of the word line capacitance:

$$C_{wordline} = C_{DifDrv} + 2^n \cdot x \cdot C_{IntPoly2W} + 2^n \cdot 2 \cdot C_{gMin} \tag{4.2}$$

where n is the number of column address lines , 2^n the number of cells per row/word line, x the horizontal size of a single memory cell, $C_{DiffDrv}$ the diffusion capacitance of a medium sized word line driver, $C_{IntPoly2W}$ the interconnect capacitance for a poly line 2λ wide, and C_{gMin} the average gate capacitance in a minimum transistor.

[4] investigate the effect of size and technology scaling on power, area and delay of SRAMs. Unfortunately they give only vage information about their energy model: The decoder energy is estimated via the

collected critical path capacitances. Bit and word line energy is computed via $C \cdot V_{DD} \cdot 2 \cdot V_{swing}$ (with C the respective capacitances). From circuit level simulations the sense amp energy is approximated by $\frac{12fj}{\lambda} \cdot w_s$ (λ half minimum feature size, w_s size of the sense-amp).

In principle switched capacitance models could be augmented by leakage models at the cost of additional analysis complexity. One way to achieve this could be a leakage characterization as proposed in [40].

Cache Models

As already mentioned in section 2.6 cache optimization has been a very active area in the recent past. Noticeably only a very small number of energy models are used for a wide range of optimizations: [134] propose the following cache model (static logic):

$$\begin{aligned} E_{cache} &= E_{decoding\_path} + E_{cell\_array} + E_{I/O\_path} \\ E_{decoding\_path} &= \alpha \cdot Addr\_bus\_bs \\ E_{cell\_array} &= \beta \cdot Word\_line\_size \cdot Bit\_line\_size \cdot Bit\_line\_sb \\ E_{I/O\_path} &= \gamma \cdot (Addr\_pad\_bs + Data\_pad\_bs) \end{aligned} \tag{4.3}$$

where:

$Addr\_bus\_bsr$	no of bit switches on address busses per instruction
$Word\_line\_size$	no of memory cells in a word line
Bit_line_size	no of memory cells in a bit line
Bit_line_sb	no of switching bit lines per instruction
$Addr\_pad\_bs$	no of bit switches on address pads per instr.
$Data\_pad\_bs$	no of bit switches on data pads per instr.
α, β, γ	Technology and implementation dependent constants

Using this model they compare caches with varying associativity and line size. [72] present a more detailed analytical model for m-way set-associative caches. The effective capacitances of SRAM cells are assumed as

$$\begin{aligned} C_{bit,pr} &= N_{rows} \cdot (0.5 \cdot C_{d,Q1} + C_{bit}) \\ C_{bit,r/w} &= N_{rows} \cdot (0.5 \cdot C_{d,Q1} + C_{bit}) + C_{d,Qp} + C_{d,Qpa} \\ C_{wordline} &= N_{columns} \cdot (2 \cdot C_{g,Q1} + C_{wordwire}) \end{aligned} \tag{4.4}$$

where $C_{bit,pr}$, $C_{bit,r/w}$ and $C_{wordline}$ are the effective load capacitances during precharging and read/write to a cell and the capacitive load of the word line driver. $C_{d,Q1}$, $C_{d,Qp}$ and $C_{d,Qpa}$ are the drain capacitances of the access transistors, the equalization transistor and the precharge enable transistors ($C_{g,X}$ are the respective gate capacitances). C_{bit} is the bit line capacitance over the height of one cell, $C_{wordwire}$ the word line capacitance over the width of one cell. For CA accesses the energy consumption is then computed as:

$$\begin{aligned} E_{cache} &= E_{bit} + E_{word} + E_{output} + E_{ainput} \\ E_{bit} &= 0.5 \cdot V_{DD}^2 \cdot \big(N_{bit,pr} \cdot C_{bit,pr} + N_{bit,w} \cdot C_{bit,r/w} + N_{bit,r} \cdot C_{bit,r/w} \\ &\quad + N_{columns} \cdot CA \cdot (C_{g,Qpa} + C_{g,Qpb} + C_{g,Qp})\big) \\ E_{word} &= V_{DD}^2 \cdot CA \cdot N_{columns} \cdot (2 \cdot C_{g,Q1} + C_{wordwire}) \\ E_{output} &= E_{aoutput} + E_{doutput} \\ E_{aoutput} &= 0.5 \cdot V_{DD}^2 \cdot (N_{out,a2m} \cdot C_{out,a2m} + N_{out,a2c} \cdot N_{out,a2c}) \\ E_{doutput} &= 0.5 \cdot V_{DD}^2 \cdot (N_{out,d2m} \cdot C_{out,d2m} + N_{out,d2c} \cdot N_{out,d2c}) \\ E_{ainput} &= 0.5 \cdot V_{DD}^2 \cdot N_{ainput} \cdot ((m+1) \cdot 2 \cdot N_{rows} \cdot C_{in,dec} + C_{awire}) \end{aligned} \tag{4.5}$$

with:

E_{cache}	total cache energy
E_{bit},E_{word}	energy dissipated in the bit/word lines
E_{input},E_{output}	dissipation due to input/output transitions
$N_{bit,pr}, N_{bit,w}, N_{bit,r}$	total number of bit line transitions due to precharge, write and read
$C_{out,a2m}, C_{out,d2m}$	the capacitive load of the address/data to main memory lines
$N_{out,a2m}, N_{out,d2m}$	the respective numbers of transistions
$C_{out,a2c}, C_{out,d2c}$	load of the CPU side interconnect
$N_{out,a2c}, N_{out,d2c}$	the respective numbers of transistions
N_{ainput}	address input transitions
$C_{in,ded}$	gate capacitance of 1. level decoder
C_{awire}	internal wire capacitance of the addressing

Of these variables the capacitances are determined using process models (see [151]). The transition counts are derived in [72] for several different configurations.

At the same conference [56] extend the model of [134] by the notion of cache misses:

$$\begin{aligned} E_{cache} =& E_{decoding\_path} + (hit\_rate) \cdot (E_{cell\_array} + E_{I/O\_path}) \\ &+ (1 - hit\_rate) \cdot (E_{main\_mem} \cdot CacheMemRatio) \end{aligned} \tag{4.6}$$

with E_{main_mem} the base energy consumed by a main memory and $CacheMemRatio$ the ratio of power consumption of a main memory access to an on-chip cache access. This model however handles the off-chip accesses erroneously. Consequently [128] propose a corrected model:

$$\begin{aligned} E_{cache} =& E_{decoding\_path} + (hit\_rate) \cdot E_{cell\_array} \\ &+ (1 - hit\_rate) \cdot (E_{cell\_array} + E_{I/O\_path} + E_{main\_mem}) \end{aligned} \tag{4.7}$$

The model is used for an exhaustive search over on-chip memory, cache and line size as well as associativity and data organization. The same authors present a more accurate model which combines [72] and [134] in [129]. The goal is to reach the accuracy of [72] and the simplicity of [134].

$$\begin{aligned} E_{cache} &= E_{decoding\_path} + E_{cell\_array} + E_{I/O\_path} + E_{main\_mem} \\ E_{decoding\_path} &= \alpha \cdot Addr\_bus\_bs \cdot (C/L) \\ E_{cell\_array} &= \beta \cdot Word\_line\_size \cdot (Bit\_line\_size + 4.8) \cdot (N_{hit} + N_{miss}) \\ E_{I/O\_path} &= \gamma \cdot (Addr\_pad\_bs + Data\_pad\_bs \cdot 8L) \\ E_{main\_mem} &= E_{I/O\_path} + Em * 8L * N_{miss} \end{aligned} \tag{4.8}$$

where C is the cache size, L the cache line size and N_{hit}, N_{miss} the number of hits and misses. This model is reported to be accurate with respect to [72] within 0.38%. [84] propose the following Cache energy model:

$$\begin{aligned} E_{cache} &= 0.5 \cdot V_{DD}^2 \cdot (CA \cdot C_{bit,rd} + CA \cdot C_{word} \\ &\quad + a \cdot C_{bit,wr} + b \cdot C_{dec} + c \cdot C_{od}) \\ C_{bit,rd} &= N_{bitl} \cdot N_{rows} \cdot (C_{SRAM,pr} + C_{SRAM,rd}) \\ &\quad + N_{cols} \cdot C_{pr\_logic} \\ C_{word} &= N_{cols} \cdot C_{word,gate} \end{aligned} \tag{4.9}$$

where:

$a \cdot C_{bit,wr}$	eff. switched capacitance for writing a bit
$b \cdot C_{dec}$	eff. switched capacitance for dedoding
$c \cdot C_{od}$	eff. switched capacitance of the output
N_{bitl}	no. of bitlines
N_{rows}	no. of rows
N_{cols}	no. of columns

The cache model is used in a comprehensive framework for the hardware/software energy optimization in embedded systems. [160] present an analytical model for register files. Only the read access model is presented here:

$$\begin{aligned}
E_{regfile,read} &= E_{wl,read} + E_{bl,read} + E_{SA} + E_{SA,ctrl} + E_{precharge,ctrl} \\
E_{wl,read} &= V_{DD}^2 \cdot N_{bits} \left(C_{gate} \cdot W_{pass,r} + W_{cell} \cdot C_{metal}\right) \\
E_{bl,read} &= V_{DD} \cdot M_{margin} \cdot V_{sense} \cdot C_{bl,read} \cdot N_{bits} \\
C_{bl,read} &= N_{reg} \cdot \left(C_{metal} \cdot H_{cell} + C_{drain} \cdot W_{pass,r}\right) \\
E_{SA} &= \frac{1}{8} \cdot V_{DD} \cdot T_{period} \cdot I_{dsat} \\
E_{SA,ctrl} &= V_{DD}^2 \cdot N_{bits} \cdot \left(C_{gate} \cdot W_{SA,ctrl} + W_{cell} \cdot C_{metal}\right) \\
E_{precharge,ctrl} &= V_{DD}^2 \cdot N_{bits} \cdot \left(\frac{C_{bl,read}}{40} + W_{cell} \cdot C_{metal}\right)
\end{aligned} \tag{4.10}$$

where:

C_{gate}, C_{drain}	gate/drain capacitance per unit width
$W_{pass,r}$	width of the cell read pass transistor
W_{cell}, H_{cell}	cell width/height
C_{metal}	per unit metal layer capacitance
M_{margin}	length of the safety margin for the word line pulse
V_{sense}	bit line swing for sensing
N_{reg}	number of registers
T_{period}	length of sensing period
I_{dsat}	transistor saturation current
$W_{SA,ctrl}$	width of sense amp control transistor

Different architectural techniques like current-direction sensing, differential sensing, low-swing write and port-priority selection are discussed using this model. A model heavily used is the CACTI model proposed by [151]. It has been developed and enhanced over a period of 8 years and is available as open source software. A fixed cache architecture template is here decomposed into a large number of simple equivalent *RC* circuits. Resistance and capacitance of these circuits are related to geometry (e.g. transistor width) and technology parameters. The capacitance driven by the address decoder drivers is for example given as:

$$\begin{aligned}
C_{eq} = {} & draincap_p(W_{decdrivp}, 1) + draincap_n(W_{decdriven}, 1) \\
& + 4 \cdot N_{dwl} \cdot N_{dbl} \cdot gatecap(W_{dec3to8n} + W_{dec3to8p}, 10) \\
& + 2 \cdot B \cdot A \cdot N_{dbl} \cdot N_{spd} \cdot C_{wordmetal}
\end{aligned} \tag{4.11}$$

where:

W_x	width of transistor x
$draincap_{p/n}(W, k)$	drain capacitance of k stacked P/NMOS transistors of width W
$gatecap(W, k)$	gate capacitance of k stacked PMOS or NMOS transistors of width W
N_{dwl}, N_{dbl}	number of horizontal/vertical cache array tiles
N_{spd}	number of cache lines per row
B	block size
A	associativity
$C_{wordmetal}$	metal wire capacitance per bit width

As might be anticipated from this example the model is very accurate (errors below 10% are reported)

and scalable with size and associativity, but also very large (involving roughly 100 parameters, therefore not printed fully here). Unfortunately it cannot be easily adapted to other cache designs (e.g. types of decoders).

Accuracy is often taken for granted when applying analytical models. Consequently few analytical models are carefully evaluated. Accuracy is however *not* a guaranteed property of analytical models. [39] report an average error of 60%. [72] admit an overestimation of up to 30%, especially with optimized architectures.

4.2 Empirical Modeling

Empirical modeling treats the internal structure of modeled instances as completely hidden. The models are built from observations of the instance behavior using statistical techniques. The observations are gathered by measurement or low level (e.g. circuit or gate level) power estimation. Leakage related power consumption can be modeled if the low level estimate contains this aspect.

4.2.1 Interpolation Techniques

Interpolation based techniques have been mainly proposed for combinational logic circuits: [48] compares different lookup tables and suggest a three dimensional table depending on the average input signal probability (P_{in}), the average input transition density (D_{in}) and the average output transition density (D_{out}). [6] add to this approach by proposing an adaptive characterization technique and a two-stage interpolation scheme. [70] proposes a 4-dimensional table using the Hamming-distances and signal distances at both inputs. While [70] is already scalable with respect to size parameters, [48] is extended in [17] to have this flexibility. [17] report average errors of 6.4%. [50] propose an unscalable 4-dimensional lookup table of P_{in}, D_{in}, D_{out} and the average spacial correlation $SC_{ij} = P\{x_i \wedge x_j = 1\}$ with average errors of 6%.

4.2.2 Regression Techniques

Regression for memory modeling has been proposed independently by [30] and [32]. The models of [30] are instance based and not scalable with size parameters. The modeling procedure is based on the ideas of [10] (see below): A linear regression tree is built as previously suggested for data path components. In contrast to [10] a software neural network is then used to select the candidate variable on which to split the model. The results of the clustering are mostly trivial however: clock, write enable, chip select and output enable are reported to be of major influence on the power consumption. This is however rather obvious even for non-experts. Errors of 6-8% are presented. [32] also perform linear regression to built separate models for the structural parts of memories, depending on size parameters and control inputs. They apply stepwise regression (see section 5.3.3) for selecting the model variables. Since some of the relationships cannot be adequately expressed using linear dependencies, they perform manual pre-transformations on some input variables.

Several regression approaches have also been suggested for the high-level power modeling of data path components. [10] propose a simple linear regression model: the binary variables i_n and o_n are 1 whenever an input/output transition occurred during subsequent cycles. The fitted model is

$$P = \sum_n c_n \cdot i_n + \sum_m c'_m \cdot o_m + c_0 + \varepsilon \tag{4.12}$$

To increase the accuracy of this model, tree regression is then applied: the model is split along one of the binary variables to obtain a piecewise model. This splitting can be performed iteratively, resulting in a tree-like structure of models. The splitting criterion in this case is the maximum variance σ^2.

Basing on the same models [15] propose an *in situ* characterized model: a behavioral simulation and a gate-level simulation is run in parallel. As long as the components are not fully characterized, the gate-level simulator is used to gather samples for a regression model. Once enough data is generated the simulation switches to the faster behavioral simulator. An iterative type of linear regression called least

mean squares (LMS) is used to effectively perform the regression on the increasing amount of data. Other such iterative regression techniques are described in [93]. The root mean square errors reported are below 50%.

[155] propose a cycle accurate, non-scalable power model for RT-level components. They introduce *transition variables*: all possible switching events on a single bit primary input line are coded into binary triples $[a\,b\,c]$. An exact functional relation between the cycle power and the transition variables is derived. This relation is however exponential in the order of input correlations considered. The function is therefore reduced by omitting the higher correlations. To obtain further improvements, the population of input vectors is stratified. A piecewise model is the result. Eventually stepwise linear regression is applied to reduce the number of variables. Relative errors of up to 20% are reported with models involving up to 15 variables (ISCAS benchmarks) .

[13] fit polynomials of third degree for combinational logic circuits. The parameters used are P_{in}, D_{in}, D_{out} (see above), plus the spacial correlation metric S_{in} and the temporal correlation metric T_{in}. An average absolute error of below 2% was shown (selected subset of ISCA-85 benchmarks).

Linear (5 terms) and quadratic (15 terms) polynomials of the variables $P_{in}, D_{in}, SC_{in}, D_{out}$ (SC: spacial correlation coefficient) are used in [49]. RLS regression is performed to allow automatic online characterization. They report average errors of 18.08% (ISCAS-89 benchmarks).

In addition to the standard techniques interpolation and regression other empirical methods have been suggested for power analysis: [29] for example use the concept of power sensitivity: the derivatives of the power consumption with respect to the primary input activity/probability (power sensitivities) are computed during an average power analysis. These derivatives can then be used for data sensitive power estimation. The work of [89] uses clustering: the complete set of switching events is partitioned into subsets that have similar power consumption. A table containing one energy value per cluster is then built with average errors of 10% - 15%.

4.3 Eclectic Modeling

Eclectic models are a mixture between conceptual and empirical models in that they rely on some information on the internal structure and on some empirical data. Often the structural information is used to propose mathematical models of which the variables are fitted using empirical data. A very fundamental approach of this kind for digital logic including memories was [83], where effective capacitance coefficients were fitted to an abstract architecture model. The approach of [39] is hybrid in a different way: after an analysis of existing memory modeling approaches it is suggested to model some structural parts of the memories analytically and some empirically. The error is reported to drop from 60% to 12 %.

The author also has performed eclectic modeling on a Philips embedded ROM [121,123]: in a combination of conceptual analysis and circuit level simulation an instance based data dependent linear regression model was developed. The very detailed model (see Appendix A) is characterized using circuit simulation of critical path models with a predefined stimulus set (30-90 CPU minutes). Mean errors are below 5% for the memory cycle and below 15% for the (asynchronous) address changes. The main lessons learned from this modeling project were:

1. Results are not self-evident. The conceptual analysis requires significant human resources.

2. Integrating the characterization into the memory generation flow requires considerable effort. In this case, for example, the critical path simulation models had to be partly re-written.

3. The models are not easily retargetable to other memories even from Philips.

4. The modeling is IP critical. The document [121] remains confidential for that reason. So is part of the motivation of the ROM model.

4.4 Discussion

When analyzing the described approaches in the light of the requirements formulated here (see section 3.3), the following observations can be made:

Among the conceptual models full and reduced net list models are not sufficiently abstract for high-level application. They are too complex and thus slow. Size scaling is often a problem too. Through their abstraction, analytical models solve the complexity problem. They are furthermore often size scalable and have the form of a set of equations making them fit for optimization algorithms. However, while the extraction of fixed layouts to gate level net lists can, within limits, be automated, the feasibility of such an automation could so far not been shown for analytical models. This also means that accuracy depends very much on the skill and effort of the involved persons. Significant errors can be the result [39]. Furthermore it was shown that 'one size fits all' assumptions lead to big errors [72]: the analytical models must fit the underlying architecture. Re-modeling might therefore be necessary with every architectural change.

In summary the biggest drawbacks of the conceptual modeling, in view of the problem to be solved, are tied to conceptual modeling processes itself: as in-depth knowledge of the circuitry is necessary, and will consequently be reflected in the models, IP protection is difficult. Modeling costs in terms of manpower, competence and "time to model" tend to be high for the same reasons. Today's analytical models neglect leakage currents. Yet, as was mentioned in section 2.1, these currents will get more and more important. As a consequence future memory models will have to include leakage models at the cost of additional complexity.

The major advantage of empirical modeling is that in-depth knowledge of the application is not required. This fact opens up potential for automation. While empirical modeling cannot be automated completely in the general case, the claim of this thesis is that for the specific task of memory power modeling it can be done to a high extend. Not feeding a high amount of expertise into the modeling process furthermore has the advantage of IP protection: neither must the modeling be outsourced nor does any detail enter the model that is not observable from the outside. Abstraction is a further model property that is natural to empirical modeling.

The drawbacks of empirical modeling are the following: As there is no guarantee that models obtained only from empirical data are 'right', accuracy is always a potential issue. Furthermore the demand for flexibility results in models with a high number of dimensions posing a problem to some empirical model representations (e.g. [59]). Among the different representations, only regression models have a closed analytical form. Computational demands can be a further problem.

As a principle drawback eclectic methods do share the need for expert knowledge with the analytical approaches. The exact properties of the range between the empirical and analytical models and depend on the combination of techniques used.

This thesis advocates that the necessity for expert intervention with its impact on modeling costs, time and IP protection is such a severe drawback that approaches with this requirement have to be ruled out. Empirical techniques are therefore the natural remaining candidates for the modeling task.

4.5 Empirical methods

Empirical modeling approaches combine three aspects that are not completely separable: learning, representation and application. The *learning* strategy describes how a model is built from the observations. The model *representation* determines how the model is stored whereas the *application* refers to the way forecasts can be drawn from the model. In the following the main streams of empirical modeling are described. The respective techniques are closely related to machine learning.

1. Table lookup methods store the empirical data directly in a table. Interpolation is used to obtain intermediate values (see section 3.2.6).

2. Decision diagrams hold the empirical data in a tree structure. Tree pruning can be used to increase the compactness of the representation [59].

3. Neural networks try to imitate the function and structure of the (human) brain. They are able to partition high-dimensional event sets by hyper planes. Neural networks are "trained" on the required response by adapting the interconnect structure and internal weighing of stimuli of the neurons [34].

4. Genetic algorithms are a generic optimization technique inspired by biology. It can work on different model representations. Instead of single models genetic algorithms optimize complete sets ("populations") of models. The fitness of each model is determined by evaluating it on the characterized data. The fittest models are then selected and allowed to "reproduce": they are copied and small changes ("mutations") are made. Evaluation, reproduction and mutation are repeated for several generations.

5. Regression optimizes the parameters of given functions as described in section 3.2.3 [92].

Table 4.1: Properties of empirical modeling techniques.

	Table	Decision Diagram	Neural Network	Genetic Alg.	Regression
Compactness	-	-	-	?	+
Analytical Form	-	-	-	?	+
Application Speed	o	o	-	?	+
Generation Speed	+	+	-	-	o
Required Data	o	o	-	-	o

Table 4.1 compares general properties of the presented techniques. Decision diagrams and neural networks have a tendency to lack compactness. They are not able to produce models in analytical form. Neural networks have slow learning convergence resulting in a high requirement of observation data. The properties of genetic algorithms depend strongly on the parameters of the algorithm and the model representation used. Nevertheless they also need a considerable amount of training data.

Taking together all properties, regression modeling seems to be the most promising paradigm for the requirements of this thesis. The potential weaknesses of regression techniques were identified previously as accuracy (out-of-sample problem) and computational complexity of the model building. Furthermore it was stated above that the advantageous properties of the modeling process is connected to its automation. In the next chapters a highly automated regression based modeling methodology is demonstrated. This methodology features modest computational complexity and high accuracy.

5 Embedded Memory Modeling Methodology

In the previous chapters the problem of memory power modeling has been motivated, requirements have been formulated and other existing work was introduced and discussed. This chapter now presents a new memory power modeling methodology based on statistical concepts. The aim of these techniques is to allow users to perform memory model building that are neither experts in statistics nor in the internals of memory circuits, while meeting the requirements defined in section 3.3.

Figure 5.1 depicts the modeling flow detailed below. This flow is orientated at the stages of statistical modeling described in section 3.1. Its stages are data abstraction, data acquisition (experimental design and characterization), identification and fitting, validation and iteration.

5.1 Data Abstraction

The data abstraction step, i.e. identification of candidate variables and derivation of such candidate variables from observables, usually precedes data acquisition and model identification as it impacts both of them. As this step is heavily dependent on the specific environment, i.e. the memory generator, the type and manufacturer of memory and the simulator/power estimator, it can not be completely automatized or formalized. On the other hand it is fairly straight forward in the context of embedded memories as will be described in this section.

The response variable is clearly identified as the energy or power consumption. Usually it can be obtained during data acquisition as energy/power per access or average current. The remaining candidate variables describe input variables to the data acquisition process as well as the modeling: Let V the set of all variables, $V_{instance} \subseteq V$ the set of variables referring to properties of the instances in question. $V_{instance}$ usually contains size related variables (e.g. number rows, number of columns) as well as configuration related ones (e.g. { 'output enable' , 'no output enable' }). Candidates for both types of variables can be found among the parameters of the generation process, i.e. memory compiler. Size parameters are usually non-negative integers and can therefore be directly used as model variables. The scope of configuration parameters is in most cases an enumeration with a small number of members. For the sake of our modeling, configuration parameters are converted into a set of binary variables using a 'one hot encoding'.

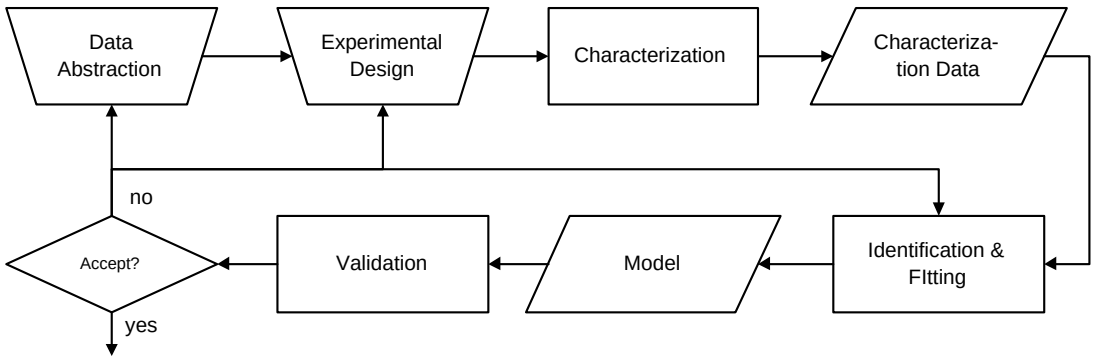


Figure 5.1: Flow diagram of the proposed modeling methodology.

Example 3. Let a variable $v \in \{member_1, \cdots, member_n\}$. We now define n binary variables $v_i \in \{0, 1\}$ with: $v_i = 1 \Leftrightarrow v = member_i$.

Variables V_{data} for modeling the data impact on the power consumption are less straight-forward as they must be abstracting: The data streams consist of pairs of bit vectors. For vectors of n bit 4^n such pairs exist. It is therefore necessary to identify functions f_a that project pairs of bit vectors onto a numerical scale according to their power impact. As switching events cause power consumption the most often used abstraction functions relate to the differences between vectors. They are: the Hamming distance, the number of rising/falling bits and the weight (number of ones in a vector).

Abstraction for address related variables $V_{address}$ may be even more difficult as it is likely that the address bits are highly structurally correlated in the memory. In case of such structural correlations functions like Hamming distance and weight, which abstract from the bit position, are not adequate. Without knowledge of the internal structure of the decoders abstraction can become very difficult.

Including V_{data} or even $V_{address}$ into the modeling procedure will severely increase the data requirement. The impact of data and addresses on the total memory internal power, however, is usually relatively small. For example the maximum data related deviation for the SRAM investigated in [121] was below 20 %, the address related effect below 1 %. These effects are normally heavily dominated by the energy consumed for data and address bus driving. A quick analysis separating the data/address influences (by reading/writing varying data to/from identical addresses and vice versa) allows an initial assessment of the impact. Depending on the accuracy requirements and available resources it then has to be carefully decided whether to include V_{data} or $V_{address}$.

The ranges of all candidate variables together form a space (*candidate space*), where every point corresponds to an assignment. Not all points in this space have to be valid however, as certain combinations of variable values may not be legal.

5.2 Data Acquisition

Once candidate variables are identified the data acquisition is performed by synthesizing selected memory instances, simulating them with given data and estimating power based on the activity data protocolled during simulation. The first subsection will explain how to decide which instance to select and which activity to simulate. The second subsection will describe the flows used to generate data in different scenarios.

5.2.1 Experimental Design

The experimental design consists of defining the sets of instances to be synthesized and the address/data streams for simulation (cf. section 3.2.2). Each instance/address/data combination corresponds to exactly one point in the candidate space (*sample point*). Note that the inverse does generally not hold, as the candidate variables do not uniquely define the address/data stream and the instance. In experimental design it is important to adequately cover the candidate space with sample points. Regions lacking observation points have to be avoided.

In this thesis factorial designs (cf. section 3.2.2) are employed as an easily implementable means to obtain adequate covers of the parameter candidate space (see section 3.2.2). In factorial designs a number k_i of levels $k_{i,j}$ is selected for each candidate variable v_i. All combinations of levels of all candidate variables are then adopted as sample points. The result is a rectangular grid structure in the candidate space comprising of $\prod_i k_i$ points.

Restrictions and complications arise from the implementation of the design: Since $V_{instance}$ variables map directly to the parameters of the generator they can be directly controlled and are consequently unproblematic. The degrees of freedom not covered by $V_{instance}$ have to be fixed to obtain unique instances. Usually there exist natural defaults defined by the generator or the application.

The data and address related variables abstract from the concrete bit vector streams as described above. Hence they are not directly controllable in the characterization process. For data acquisition it is therefore necessary to generate data/address streams that correspond to the values of the data related variables (e.g.

with given hamming distance, etc.) by applying the inverted abstraction functions f_a^{-1}. As mentioned above these are usually not unique. The problem to be solved is now to synthesize a data stream that consists of a sequence of vector pairs that each lie in the inverted abstraction f_a^{-1} of all variables in V_{data}. Furthermore it is desirable to perform the synthesis in a random fashion to avoid accidental biasing. A documentation of approaches to this problem is beyond the scope of this thesis.

In cases where the data acquired in an initial modeling run do not produce sufficient accuracy, further data acquisition runs may become necessary. In the evaluation phase of the first run the accuracy problem can usually be tied to a specific region in the candidate plane. The new design can now be specifically tailored to remedy the problem. This can be done for example by another factorial design that has a high density of levels in the problematic area. Note that the union of two factorial designs is in general not a factorial design (figure 5.2). This highlights the advantage of the technique proposed here, which is not limited to factorial designs.

5.2.2 Characterization

for all instances **do**
  generate memory circuit level net list
  generate testbench in SPICE/PStar
  for all $V_{data}, V_{address}$ combinations **do**
    synthesize data/address streams
    perform circuit level simulation/estimation
    extract power value
  end for
end for

Algorithm 1: Characterization based on circuit-level power analysis.

Characterization refers to the conducting of low level estimation to obtain data for the model building. The characterization flow in this thesis encompasses three possible scenarios: circuit-level estimation, gate-level estimation and simple feed back from the memory generator. While circuit level estimation is the most accurate, it is not always possible due to limitations in time or access to the respective net lists. In many cases not even a gate-level net-list will be available to users (IP problem). In these cases the fall back solution is to use the estimates provided by the user version of the memory compiler.

The circuit-level estimation based characterization flow (see algorithm 1) begins with the generation of the respective memory instance as circuit-level net-list (e.g. SPICE, PowerMill, PStar [113, 58, 135, 130]). Subsequently a testbench is synthesized that feeds the circuit with the address/data streams and tracks its power consumption. Next address and data streams according to the properties defined by the sample point are generated. The circuit and testbench are then simulated for all such data streams. Finally the power value of each simulation is extracted from the simulation protocols.

Gate-level estimation based characterization is similar (see algorithm 2). The memory is generated as gate-level (firm macro) or RT-level (soft macro) net list. Logic synthesis and technology mapping are performed as necessary. The resulting technology mapped net list is subsequently written out in HDL

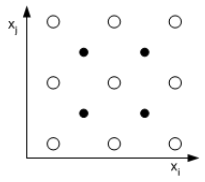


Figure 5.2: The union of two factorial designs is in general not a factorial design.

```
for all instances do
  generate memory
  perform logic synthesis / technology mapping
  generate testbench in HDL
  for all V_data, V_address combinations do
    synthesize data/address streams
    simulate memory to obtain switching activity
    perform gate-level analysis
    extract power value
  end for
end for
```

Algorithm 2: Characterization based on gate-level power analysis.

(e.g. VHDL or Verilog) source code. Timing information is written into an SDF (Standard Delay Format) file. A matching testbench is also generated in HDL source code. It is then compiled together with the memory net list for simulation. In the simulation step an HDL simulator is used to track the switching activity within the circuit during the processing of the generated address/data streams. The simulator reads the SDF file for timing information. The protocol of switching activity is obtained by using LPI extensions of the simulators, which write standardized switching activity files (SAIF (Switching Activity Interchange Format) or VCD (Value Change Dump)). The switching information is subsequently used by gate-level power estimators such as Synopsys PowerCompiler® [136] or WattWatcher Gate® to obtain the final power consumption estimate.

```
for all instances do
  generate memory
  extract power value
end for
```

Algorithm 3: Characterization based on generator estimate.

As a last resort the characterization can be performed using the customer available memory generator only. When called to generate customer views like HDL descriptions the generator also provides a power estimate. These power estimates are usually address/data independent. Furthermore they often constitute only rough approximations. The flow of this scenario is simple (see algorithm 3): the generator is called for every instance and the power value is extracted from its report.

Certainly this scenario is the least attractive one. One might even ask why not to activate the memory compiler directly from the application instead of first generating a model with the proposed methodology. A couple of reasons stand against this: invoking the compiler is still orders of magnitude slower then evaluating the model. The compiler does not have to be available at the site of application. Furthermore it only provides point estimates. The models in contrast have analytical form with the advantages mentioned before.

5.3 Identification and Fitting

In the proposed approach model identification and fitting go hand in hand. As no other information is available, searching for an adequate model requires repeated fitting and error analysis. Defining a family of tentative models has two conflicting objectives: the family is to be kept general enough to express all relationships occurring and at the same time concrete enough to allow an effective selection and fitting. The problems to be solved are the selection of a minimal set of variables, the determination of their mutual interactions and the identification a mathematical expression for their influence on the result.

The next subsection presents a family of mathematical models called signomials. These models are especially suitable for memory power estimation and optimization (see below). The following two subsections describe the selection and fitting based on this model family: The first one explains the identification of a mathematical relationship between candidate variables and the response, second one details the variable selection process.

5.3.1 Signomial Models

The choice of a family of models represents a compromise between model expressiveness, efficient generation and applicability as expressed in the requirements in section 3.3. This thesis advocates that for the task of memory modeling *signomials* represent a nearly optimal compromise. In the remainder of this section an effective generation technique will be described. Section 5.6 documents the suitability of signomials for optimization tasks. Chapter 7 shows that the models generated with the presented technique outperform existing ones. Signomials are defined as follows:

Let t variables x_i, $\forall x_i : x_i \in \mathbb{R}^+$. A *monomial* is then

$$\beta_i \cdot x_{l_{i,1}}^{\alpha_{i,1}} \cdot \ldots \cdot x_{l_{i,p_i}}^{\alpha_{i,p_i}} =: \beta_i \cdot \xi_i \left(\mathbf{X}, \boldsymbol{\alpha_i}\right) \tag{5.1}$$

where $\beta_i, \alpha_{i,j} \in \mathbb{R}$, $l_{i,j} \in \mathbb{N}$, $\boldsymbol{l_i} = [l_{i,1}, \ldots, l_{i,p_i}]'$ a vector containing an ordered subset of p_i indices of the original variables, formally:

$$\forall 1 \leq j \neq h \leq p_i : 1 \leq l_{i,j} \neq l_{i,h} \leq t \tag{5.2}$$

The vector $\boldsymbol{\alpha_i}$ contains one exponent for each variable designated by the elements of $\boldsymbol{l_i}$. In other words: each term is a product of a subset of the variables with an individual real valued exponent. In the strict mathematical sense the monomial is the product $\beta_i \cdot \xi_i$. For the sake of simplicity the term monomial (or *term*) will in the remainder of this work also be used for only the function ξ_i.

Given the monomials a *signomial* can now be defined as a sum of monomials:

$$f(\boldsymbol{\xi}(\mathbf{X}, \boldsymbol{\alpha}_i), \boldsymbol{\beta}) = \left(\sum_{i=1}^{k} \beta_i \cdot \xi_i(\mathbf{X}, \boldsymbol{\alpha}_i) \right) + \beta_0 \tag{5.3}$$

Signomials with the restriction $\beta_i \in \mathbb{R}^+$ are called *posynomials*. Posynomials, as will be shown later, take an important role in nonlinear optimization.

Compare the signomials with simple multi-linear regression models (repeated here for convenience):

$$f(\mathbf{X}, \boldsymbol{\beta}) = \left(\sum_{i=1}^{k} \beta_i \cdot \boldsymbol{x}_i \right) + \beta_0 \tag{5.4}$$

Both models have similar outwardly structure and both models are linear in the parameters β_i. Therefore signomials can like multi-linear functions be fitted using linear regression (provided the $\alpha_{i,j}$ are fixed).

For the fitting process the *extended signomials* are defined by redefining the zero exponent x^0 in this context:

$$x_q^0 := \ln x_q \tag{5.5}$$

To see why this is appropriate, let:

$$x^\lambda = e^{\lambda \ln x} = 1 + \lambda \ln x + \frac{1}{2}\lambda^2 (\ln x)^2 + \cdots \tag{5.6}$$

leading to

$$\lim_{\lambda \to 0} \left(\frac{x^\lambda - 1}{\lambda} \right) = \ln x \tag{5.7}$$

Note that the bracketed term on the left side of the equation is a linear transformation of x^λ. The last equation basically expresses that for small λ the relationship between x^λ and $\ln x$ is approximately linear [19].

To summarize: the parameters of signomials are the coefficients β_i, the exponents $\boldsymbol{\alpha}_i$ and the subsets $\boldsymbol{l}_i$. Each of the monomials ξ_i can be viewed as a transformation of the original variables. f has an outer structure similar to an ordinary multi-linear model.

Continuous nonlinearities can under most circumstance be well linearized using simple power transformations. Due to the real valued exponents α_i a wide range of relationships can be expressed. Among these are all polynomials x^n, reciprocal polynomials $1/x^n$, roots $\sqrt[n]{x}$ and logarithms $\log_n x$. Together with the possibility to build interaction terms this gives a very broad modeling space.

Piecewise Modeling

Piecewise nonlinear relationships can be easily expressed. A simple example: assume we would like to split a model at the cut line $x_i = \varphi$, e.g.:

$$y = \begin{cases} \beta \cdot x_j^{\alpha} & x_i < \varphi \\ \beta' \cdot x_j^{\alpha'} & x_i >= \varphi \end{cases} \tag{5.8}$$

We can now define a pair of new artificial discriminator variables:

$$x_d = \begin{cases} 1 & x_i < \varphi \\ 0 & otherwise \end{cases} \tag{5.9}$$

$$x'_d = 1 - x_d \tag{5.10}$$

With this variable the piecewise model can be written as

$$\begin{aligned} y &= \beta \cdot x_j^{\alpha} \cdot x_d \\ &+ \beta' \cdot x_j^{\alpha'} \cdot x'_d \end{aligned} \tag{5.11}$$

and thus be expressed by the presented model family.

Matrix notation

The model coefficients can be written in a *matrix notation* that is more convenient for computation: for a model defined as above this notation is given by a real plus a boolean valued $k \times t$ matrix:

$$\mathbf{M}_{i,j} = \begin{cases} \alpha_{i,q} & \exists q : l_{i,q} = j \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

$$\mathbf{B}_{i,j} = \begin{cases} 1 & \exists q : l_{i,q} = j \\ 0 & \text{otherwise} \end{cases} \tag{5.13}$$

Note that the second matrix is necessary because the $\alpha_{i,j}$ may become zero.

5.3.2 Variable transformations

As was stated in [87] research on fitting procedures ('response surface modeling') has concentrated on polynomials rather than on signomials. Therefore a new technique is presented in the following subsections to effectively fit signomials to given observations.

This subsection deals with the first part of fitting these models: fitting the $\alpha_{i,j}$. For this purpose the model structure defined by the subsets $\boldsymbol{l}_i$ is assumed to be fixed. Fitting eqn. 5.3 then still means determining appropriate transformations $\alpha_{i,j}$ and linear coefficients β_i. To tackle this problem an adaptive transformation of the independent variables is applied. This iterative algorithm extends the work of Box and Tidwell to handle signomials [20, 19].

Let without loss of generality $\alpha_{i,j}^{(0)} = 1$ the initial guess for the exponents and $\boldsymbol{\xi}^{(0)}$ with $\boldsymbol{\xi}^{(0)} = \boldsymbol{\xi}(\mathbf{X}, \boldsymbol{\alpha}^{(0)})$ the respective monomials. To improve the exponents, a first order Taylor expansion is performed around

the initial guess

$$\begin{aligned} f(\boldsymbol{\xi},\boldsymbol{\beta}) &= f\left(\boldsymbol{\xi}^{(0)},\boldsymbol{\beta}\right) \\ &+ \sum_{i=1}^{k}\sum_{j=1}^{p_i}\left(\alpha_{i,j}-\alpha_{i,j}^{(0)}\right)\left\{\frac{\partial f(\boldsymbol{\xi},\boldsymbol{\beta})}{\partial \alpha_{i,j}}\right\}_{\substack{\boldsymbol{\xi}^{(0)}\\ \boldsymbol{\alpha}_i^{(0)}}} \end{aligned} \tag{5.14}$$

now the derivative is split:

$$\left\{\frac{\partial f(\boldsymbol{\xi},\boldsymbol{\beta})}{\partial \alpha_{i,j}}\right\}_{\substack{\boldsymbol{\xi}^{(0)}\\ \boldsymbol{\alpha}_i^{(0)}}} = \left\{\frac{\partial f(\boldsymbol{\xi},\boldsymbol{\beta})}{\partial \xi_i}\right\}_{\boldsymbol{\xi}^{(0)}} \left\{\frac{\partial \xi_i}{\partial \alpha_{i,j}}\right\}_{\boldsymbol{\alpha}_i^{(0)}} \tag{5.15}$$

the latter can be computed directly (see eqn. 5.1)

$$\left\{\frac{\partial \xi_i}{\partial \alpha_{i,j}}\right\}_{\boldsymbol{\alpha}_i^{(0)}} = x_{l_{i,1}}^{\alpha_{i,1}^{(0)}} \cdot \cdots \cdot x_{l_{i,p_i}}^{\alpha_{i,p_i}^{(0)}} \cdot \ln x_{l_{i,j}} \tag{5.16}$$

and (cf. eqn. 5.3)

$$\left\{\frac{\partial f(\boldsymbol{\xi},\boldsymbol{\beta})}{\partial \xi_i}\right\} = \beta_i \tag{5.17}$$

The "real" β_i are not known however, since they are dependent on the $\alpha_{i,j}$, which are not yet correct. Hence an estimator $\boldsymbol{b}$ for $\boldsymbol{\beta}$ is computed by a preliminary fit using the initial guess for the alphas

$$\boldsymbol{y} = \boldsymbol{f}\left(\boldsymbol{\xi^{(0)}}, \boldsymbol{b}\right) + \boldsymbol{\varepsilon} \tag{5.18}$$

With these preliminary considerations the Box-Tidwell algorithm (alg. 4) can be explained: During the algorithm the $\chi_{i,j}$ will contain the $x_{l_{i,j}}^{\alpha_{i,j}}$ with the current α. As the initial guess for all exponents is 1 (line 2), it is sufficient to copy the $x_{l_{i,j}}$ into the $\chi_{i,j}$ at the start (line 3). In the subsequent step the estimator $\boldsymbol{b}$ for $\boldsymbol{\beta}$ is determined through regression (line 5). The same happens with the estimator $c_{i,j}$ for $(\alpha_{i,j}-\alpha_{i,j}^{(0)})\cdot\beta_i$ (line 6). The new approximations of alpha can then be computed (line 7). Finally the iteration counter is increased and the χ are adapted to the new exponents (line 16+17). The algorithm is repeated until either the solution has stabilized (the maximum change in $\alpha_{i,j}$ is below the threshold $\alpha_{threshold}$) or a maximum number of iterations i_{max} is reached (line 18). The iteration threshold becomes necessary for over-determined models (see section 5.3.5) that can make the algorithm fail to converge. These models also make it necessary to force the exponents to stay within safe bounds, i.e. regions where over/underflows do not occur. This is achieved by the statements in line 8-15. The choice of values for i_{max}, α_{max} and α_{min} is discussed in section 5.3.5.

Remarks

This transformation differs from the often suggested variance stabilizing transformations in that it is performed on the input variables instead of the response. The transformation can be seen as a combination of regression and Gauss-Newton optimization [7]:

The Gauss-Newton technique allows the iterative fitting of a function provided its derivatives are computable. This is achieved by a first order Taylor expansion of all parameters ψ_i (compare equation 5.14):

$$f(\boldsymbol{\psi}) = f(\boldsymbol{\psi}^{(0)}) + \sum_i \delta_i \frac{\partial f(X,\boldsymbol{\psi})}{\partial \psi_i} \tag{5.21}$$

The *Gauss increments* $\boldsymbol{\delta}$ is then computed so as to minimize a given error function (in our case the mean square error). Finally the parameter estimate ψ is corrected by the gauss increment ($\boldsymbol{\psi'} = \boldsymbol{\psi} + \boldsymbol{\delta}$).

In the Box-Tidwell method described above only the exponents $\alpha_{i,j}$ are treated as variable parameters with the Gauss-Newton method. The parameters β are fitted separately using simple regression (cf. eqn. 5.18). The computation of the Gauss increments is also performed by regression. With the seperation described in equation 5.15 the $\frac{\partial f(X,\boldsymbol{\psi})}{\partial \psi_i}$ are in turn computed using the estimates for β.

Input: $\mathbf{X}$: matrix of observations, $\boldsymbol{l}_i$ model structure
Output: $\alpha_{i,j}$ adapted exponents

$i \leftarrow 0$
$\alpha_{i,j} \leftarrow 1$
$\chi_{i,j} \leftarrow x_{l_{i,j}}$
repeat
  fit the following equation to obtain the b´s as an estimate of the $\beta's$:

$$\boldsymbol{y} = \boldsymbol{f}\left(\boldsymbol{\xi^{(0)}}, \boldsymbol{b}\right) + \boldsymbol{\varepsilon} \tag{5.19}$$

  fit the following equation to acquire the estimates $c_{i,j}$ for $(\alpha_{i,j} - \alpha_{i,j}^{(0)}) \cdot \beta_i$:

$$\boldsymbol{y} = \boldsymbol{f}\left(\boldsymbol{\xi^{(0)}}, \boldsymbol{\beta}\right) + \sum_{\boldsymbol{i=1}}^{\boldsymbol{k}} \sum_{\boldsymbol{j=1}}^{\boldsymbol{p_i}} \boldsymbol{c_{i,j}} \cdot \boldsymbol{\xi_i^{(0)}} \cdot \ln \boldsymbol{x_{l_{i,j}}} + \boldsymbol{\varepsilon} \tag{5.20}$$

  obtain the corrected α by setting: $\alpha_{i,j} = \frac{c_{i,j}}{b_i} + \alpha_{i,j}^{(0)}$
  for all $\alpha_{i,j}$ **do**
    if $\alpha_{i,j} > \alpha_{max}$ **then**
      $\alpha_{i,j} \leftarrow \alpha_{max}$
    end if
    if $\alpha_{i,j} < \alpha_{min}$ **then**
      $\alpha_{i,j} \leftarrow \alpha_{min}$
    end if
  end for
  $i \leftarrow i + 1$
  $\chi_{i,j} \leftarrow x_{l_{i,j}}^{\alpha_{i,j}}$
until $\max \frac{c_{i,j}}{b_i} < \alpha_{threshold}$ **or** $i > i_{max}$

Algorithm 4: Box-Tidwell regression for signomials.

5.3.3 Variable Selection

In the previous section it was demonstrated how the exponents are optimized for a given model structure (set of $\boldsymbol{l}_i$). This leaves to be solved the problem of identifying the $\boldsymbol{l}_i$. Identifying the variables involved in a statistical model is a problem of regression analysis called *variable selection*. Since exhaustive search is often infeasible due to the number of possibilities ($\sum_{i=1}^{n} 2^{ik}$ in this case), other techniques have been proposed (see [92] for an overview). For this thesis *stepwise regression* is used: Stepwise regression is based on the F test as a measure of statistical significance (cf. section 3.2.5). Note that for the signomial models the test described in section 3.2.5 is performed on the monomials $\beta_i \cdot \xi_i$ instead of simple variables $\beta_i \cdot x_i$. As stepwise regression is an iterative technique, it integrates well with the other techniques described.

The stepwise regression algorithm proceeds as follows: start with an empty model and add the most significant variable/ term (highest F_0) until significance drops below a probability threshold ("forward step"). After each inclusion of a new variable check all variables already in the model whether they have become obsolete ("backward step"). Stop when neither forward step nor backward step changes the model. The algorithm is described in more detail in the next sub-section.

5.3.4 Model Generation Algorithm

In the preceding section signomial models were introduced. Subsequently the selection and transformation of variables were described. This section now integrates these fragments in describing the complete model generation algorithm. The key aspect is the interplay of selection and transformation. The algorithm is show in listing 5:

The model generation starts with a model containing only a constant (line 1). It is repeated until convergence is reached (line 2). The "forward selection" begins in line 3: every monomial in the model (l. 4) is in turn expanded by a variable not already in the model (l. 5-7). The new monomial is added to the previous model to build a new one (l. 8). Then the exponents are adapted for the new model using the technique described above (l. 9). Now the statistical significance of the added term is tested by performing the F-test on the previous model and the expanded one (l. 10). If the new term is the most significant candidate so far, it is stored (l. 11-13). If the best candidate over all evaluated models is above the significance threshold, it is made the new model (l. 17-19). Convergence can only occur if no term was added (l. 21).

The backward step goes through all monomials of the models (l.24). It removes the respective monomial from the model and performs the adaption of exponents followed by the significance test (l. 25-27). When the F-value obtained is the smallest encountered so far, it is stored (l. 28-31). Should the least significant monomial be under the significance threshold it is removed from the model. The algorithm does only converge if neither the forward nor the backward step does change the model (l. 33-36).

Example 4. Let the variables be x, y and b. Let further $n = 100$. The algorithm starts with an "empty" model containing only a constant c (represented as $M = B = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$). The following new monomials are considered in the first iteration: $c \cdot x, c \cdot y, c \cdot b$. Each corresponding new model is then optimized using the Box-Tidwell method and rated using the F-test. Suppose the best model turns out to be $11.4 \cdot y^{0.46}$ (represented as $M = \begin{pmatrix} 0 & 0.46 & 0 \end{pmatrix}$ $B = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$. Note that factors like 11.4 are not part of the matrices as they can always be recovered by a simple regression run), with F_max= 3.5. Since the F-value is above the threshold $F(0.9, 1, t + 2) = 2.76$, the new monomial will be included. This ends the forward step. The backward step is not able to remove a monomial in this iteration. The next forward step then considers $c \cdot x, c \cdot b, c \cdot x \cdot y$ and $c \cdot y \cdot b$ as new terms.

Remarks

Note that in this algorithm the search space is drastically restricted by considering as new monomials only monomials already in the model, expanded by a single additional independent variable. Through this measure the complexity of each forward step is reduced from 2^v to $k \cdot v$ where k is the number of terms and v the number candidate variables. The following reasoning motivates the restriction made: Suppose the underlying relationship is of the form $c_2 \cdot x_i \cdot x_j$. For the sake of simplicity the current model only

1: $converged \leftarrow false, M \leftarrow 0, B \leftarrow 0, t \leftarrow 0$
2: **while not** *converged* **do**
3: $F_{max} \leftarrow 0$ { "forward step" }
4: **for all** rows i of M/B **do**
5: **for all** columns j, $B(i,j) = 0$ **do**
6: $m \leftarrow M(i,), b \leftarrow B(i,)$
7: $m(j) \leftarrow 1, b(j) \leftarrow 1$
8: $M' \leftarrow \left(\frac{M}{m}\right), B' \leftarrow \left(\frac{B}{b}\right)$
9: $(B', M') \leftarrow box\_tidwell(B', M')$
10: $F_0 \leftarrow F\_test(M, B, M', B')$
11: **if** $F_0 > F_{max}$ **then**
12: $F_{max} \leftarrow F_0$
13: $M_{max} \leftarrow M', B_{max} \leftarrow B'$
14: **end if**
15: **end for**
16: **end for**
17: **if** $F_{max} > F(p, 1, n-t)$ **then**
18: $M \leftarrow M_{max}, B \leftarrow B_{max}$
19: $t \leftarrow t + 1$
20: **else**
21: $converged \leftarrow true$
22: **end if**
23: $F_{min} \leftarrow F(p, 1, n-t) + 1$ {"backward step"}
24: **for all** rows i of M/B **do**
25: $(M', B') \leftarrow remove\_row(i, M, B)$
26: $(B', M') \leftarrow box\_tidwell(B', M')$
27: $F_0 \leftarrow F\_test(M', B', M, B)$
28: **if** $F_0 < F_{min}$ **then**
29: $F_{min} \leftarrow F_0$
30: $M_{min} \leftarrow M', B_{min} \leftarrow B'$
31: **end if**
32: **end for**
33: **if** $F_{min} < F(p, 1, n-t)$ **then**
34: $M \leftarrow M_{min}, B \leftarrow B_{min}, t \leftarrow t - 1$
35: $converged \leftarrow false$
36: **end if**
37: **end while**

Algorithm 5: Iterative model generation.

contains a constant c_1. The MSE is:

$$MSE = mean((c_2 \cdot x_i \cdot x_j - c_1)^2) \tag{5.22}$$

By including a term $c_2 \cdot \bar{x}_i \cdot x_j$ or $c_2 \cdot \bar{x}_j \cdot x_i$ the error will reduce unless the means are zero:

$$MSE = mean((c_2 \cdot (x_i - \bar{x}_i) \cdot x_j - c_1')^2) \quad \text{or} \tag{5.23}$$

$$MSE = mean((c_2 \cdot x_i \cdot (x_j - \bar{x}_j) - c_1')^2) \tag{5.24}$$

There is therefore a high probability that one of these monomials will enter the model. In the next forward step the interaction term $x_i \cdot x_j$ will be considered. Note that as $\forall x_i : x_i \in \mathbb{R}^+$, $\bar{x}_i = \bar{x}_j = 0$ would mean that both are zero variables and thus meaningless.

5.3.5 Convergence and performance

Convergence

The algorithm contains two main loops: the iterative variable transformation as inner and the stepwise regression as outer loop. It is a commonly accepted fact, that the stepwise regression algorithm in the version described here is cycle free and since the number of possible models is finite it is consequently terminating (see [90] for proof). For the inner loop, the situation is more complex: in the forward step new candidates are included in the model. If these candidates are superfluous they can render the model over-specified. As a consequence the model coefficients are no longer well determined. In these cases the quotient c/b in algorithm 4 can become numerically unstable. This problem was tackled by limiting the number of iterations (see also section 5.3.2). In the evaluation chapter an upper bound $i_{max} = 30$ is chosen. This is motivated by the distribution of iterations required for termination (see figure 5.3): either the algorithm terminates within the first 30 iterations, or it takes very long to terminate. To avoid arithmetical overflows and underflows, the exponents are furthermore constraint to an empirically selected region. For the evaluation chapter $\alpha_{min} = -30$ and $\alpha_{max} = +3$ are chosen.

Note that these measures are not critical to the algorithm: if the model including the new term is over-specified, the new term does obviously not improve the model any further. The regression step will therefore reject this model with high probability.

Complexity

Asymptotical complexity considerations are a complicated issue for iterative algorithms as presented here: the sequence in which candidate models are tested cannot be exactly predicted and therefore no candidate can be ruled out a priori. The theoretical worst case is consequently exponential, as is the total number of possible models. In practical cases the nested structure of the candidate models leads to much lower effort. The parameter transformation algorithm has been chosen for its quick convergence. It reaches the upper bound of 30 iterations (involving 60 regression steps) only for over-specified models.

The overhead of introducing an additional input variable is dependent on its involvement in the model: in best case (for runtime) the new variable does not enter the model. The overhead is linear in this case because one additional candidate variable per model term is tested in each forward step.

5.3.6 Relation to other work

The techniques of this chapter have similarities to the work of Coumeri and Thomas [32] in also using stepwise regression for variable selection. Their technique however is restricted to linear models of the candidate variables. The approach proposed in [32] has three remaining drawbacks that are overcome be the method presented here:

1. A structural decomposition of the memory circuit is performed before modeling. This is a severe problem for characterization as observations of all separate components must be generated. Often the low level simulation models do not allow this.
2. A manual pre-transformation of variables is necessary, violating the notion of black box modeling.

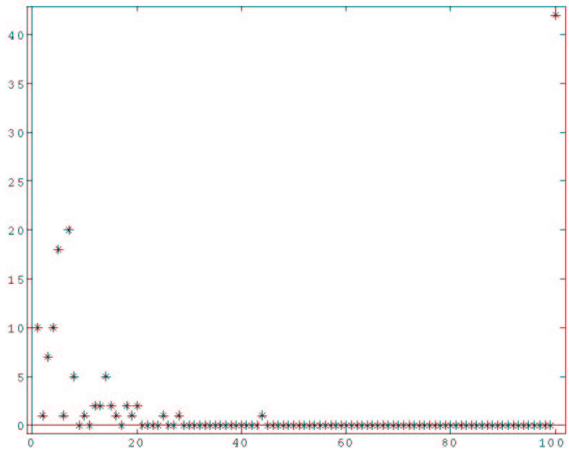


Figure 5.3: Typical histogram of the iteration count during the variable transformation. Iteration counts exceeding 100 are listed under 100.

3. Only linear dependencies can be expressed. As mentioned above linear relationships cannot express all dependencies within memories adequately.

In the evaluation section both approaches are compared against each other.

5.4 Validation

Whenever empirical modeling is performed, validation is extremely important. Therefore the validation of the generated models is the consequent next step in the proposed methodology. Two questions must be answered:

1. How well does the model fit the observations?
2. How well does the model serve as a predictor?

Although the second question can be considered the ultima ratio, both, the first and the second are not completely separable: a model that only fits past observations but has no predictive quality is clearly useless. Yet the quality of prediction can itself only be *predicted* from the model and observations made before or during model building. In other words the quality of fit to past observations determines the confidence in predictions. The formal relationship is described below.

From an operative point of view validation is also strongly connected to a third aspect: If it does not fit well, what is the problem? This question, like the other two, can be investigated by analyzing the quality of fit. The next subsection describes mathematical measures for such an analysis. Visual analysis, a technique that accompanies the mathematical measures especially for problem diagnosis, is described in the next subsection. Finally, criteria for an of acceptance models are discussed.

5.4.1 Mathematical Measures for Quality of Fit

Statistical *criteria of fit* can be divided into three families [45]:

1. *Discrepancy* between fitted model and data.
2. *Explanatory ability.* A measure that reflects how well the data is explained by the model. Likelihood is such a measure. For linear models with normally distributed errors likelihood is equivalent to the mean square error [45].

3. *Shape matching.* The quality of fit is assessed by the comparison of shapes of the model and the sampled data. This criterium is particularly used for such models that assume certain statistical distributions. As we have no distributional model, this criterion is not applicable here.

Based on the assumptions about the nature of errors described in section 3.2.3 regression analysis supplies criteria of the first two families: The root mean square error (see section 3.2.7) serves as measure of discrepancy as well as likelihood. It can be used as estimator for the variance of the errors of predictions made from the model. With this information it is possible to compute a confidence interval for predictions as described in section 3.2.4. This computation demonstrates the connection between the two issues mentioned above: The MSE, a traditional criterium of fit, determines the confidence in predictions made from the model.

The coefficient of multiple determination R^2 gives further information about the fit: low values of R^2 suggest that the model was not adequately linearized. High values on the other hand are not per se a guaranty of good fit. The relative error measures provide more intuitive criteria, but are often misleading when an optimization of the absolute error is performed.

It must however be remembered that all these error measures do not formally document the predictive qualities but merely give testimony about the sampled data. If such data, i.e. the cross validation set, is carefully chosen, it is intuitive that the errors observed later in application are similar. This is however not formally guaranteed.

In the proposed methodology the following criteria of fit are computed (cf. 3.2.7): MSE, standard deviation, R^2, XARE, MARE, MRE.

5.4.2 Visual Inspection

Visual inspection is common practice in statistics. It relies on unsurpassed human capability to intuitively assess the quality of fit or detect hidden relationships. In the context of the presented methodology it is often helpful for problem diagnosis when the mathematical error measures are unacceptably high. In these cases visual inspection can provide hints on the nature of the problem. Two common graphical representations are suggested here: a) a plot of both, the model and the data, against one or more variables and b) a plot of the residuals against one or more variables.

Data vs. Model plot

Two different aspects can be investigated with this plot: a) the fit and b) whether the relationship between candidate variables and response is well defined. The graph gives a good impression about the overall fit (see figure 5.4). For diagnosis, regions of sudden changes in the observations (jumps, or changes of slope) can be identified. It is then possible to analyze whether the model adequately reflects these changes. Regions of increased errors can be found and attributed to either a region of change as described or a lack of sample points.

If the relationship between candidate variables and response is not well defined this is indicated by a high degree of scattering of the response. It is an indicator that the data abstraction failed: as some aspects of the underlying relationship were not or not adequately recognized they have a quasi-random influence on the response in the plot.

Residuals vs. Variable plot

This plot can be used to identify inadequacies in the model: If a model is correct, the remaining errors are randomly distributed with constant variance. Therefore, for every variable a plot of the residuals against the variable should show a random distribution with constant variance. Note that this also applies for candidate variables not in the model. If such a plot shows a strong correlation of variable and error this is an indicator that the variable is not adequately represented in the model.

The principle of both graphs are fairly intuitive even for users not versed in statistics. The biggest limitation of this method is its restriction to three dimensions, meaning that only two variables can be plotted independently. This can in some cases lead to plots that are counterintuitive at first glance (see example 5). When assessing such a plot it must always be kept in mind that it represents only a section of a multidimensional graph.

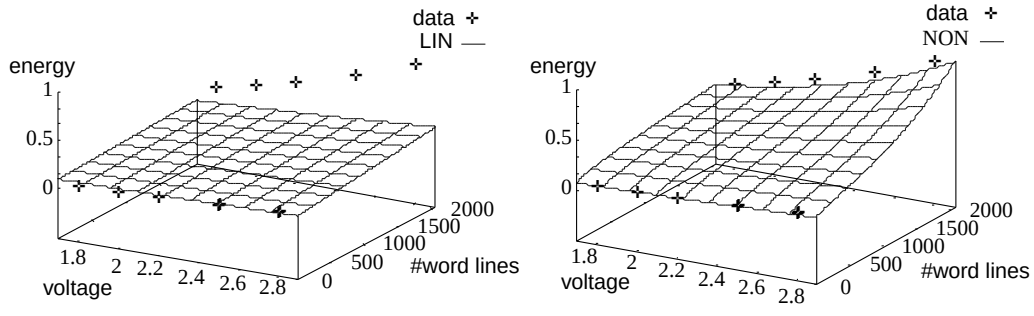


Figure 5.4: The restriction to three dimensions sometimes results in counter intuitive model plots (left). Nonlinear models can often approximate the response surface better (right).

Example 5. See figure 5.4. It clear seems as if the modeling could easily have performed better. Yet the graph only shows a small section of the multidimensional graph. Increasing the local accuracy here would have lead to a worse overall fit.

Note that all candidate variables should be investigated and not just the ones included in the model.

5.4.3 Criteria of Acceptance

The above techniques can be used to compare different models. For the modeling process that is by its nature iterative, it is also necessary to define criteria when to stop and accept the current model. As the presented methodology involves users with limited modeling experience, these criteria should be easily evaluated.

Suitable criteria of acceptance are bounds on one or several measures of fit described above. Values for such bounds must be chosen with the application context in mind (see section 3.3). It is therefore sensible that such bounds are not defined by the memory modeler but an expert in the eventual application.

Example 6. In section 3.3 the accuracy requirement of 20% maximum average error (RMS error as well as MARE) was set as this is the accuracy of models for other circuit aspects at this level of abstraction. Should a modelling run turn out to produce errors above this threshold, additional modelling effort (e.g. recharacterization) would be necessary. In industrial environments this would require a cost vs. accuracy tradeoff.

This example shows, that a tradeoff between accuracy and modeling cost may become necessary. Such a tradeoff is complicated by the fact that the accuracy gains of additional modeling effort are not known a priori.

5.5 Iteration

In the previous section possible criteria of acceptance were discussed. If a model does not fulfill these criteria it is necessary to re-iterate the modeling procedure (cf. section 3.1). This methodology suggests three different iterative steps: (Re-)defining candidate variables, acquiring more data, regenerating the model.

(Re-)defining candidate variables

Conjectures about missed aspects in the previous modeling run can trigger this feedback loop. Such conjectures could be new tentative abstraction functions f_a, the necessity of piecewise modeling or the introduction of completely new candidate. In a newly preformed data abstraction new candidate variables are defined to reflect the speculations.

In some cases the levels of the new candidate variable resulting from the original experimental design do not adequately cover the candidate space. In these cases it is necessary to repeat the experimental design

and data acquisition step (cf. section 5.2.1) before model generation. Otherwise the model generation can be performed based on the existing data.

Acquiring more data
When the previous model is close in form to the observations but lacks local or global accuracy, this can usually be remedied by (locally) increasing the density of sample points. To do so a new experimental design is set up and characterization is restarted, followed by another model generation run. Note that one of the strengths of the presented methodology is, that no restrictions are posed on the new experimental design. It can therefore be chosen independently of the initial design and deal optimally with the accuracy problem.

Restart at model generation
The model generation procedure is affected by a number of parameters. Among these are for example the upper and lower bounds for the exponents, or the weight function used. In some cases it can therefore be sensible to re-execute the model generation with modified parameters (cf. section 6.2.2).

Selecting measures for improvement
As described above, the validation, especially the visual examination, produces insight into shortcomings of the current model. The selection of an appropriate measure has not been automated, but here is a list of miss-specifications and measures:

1. *Jump discontinuities.* When the trend of the observations exhibits sudden changes that are not adequately reproduced by the model, piecewise modeling is indicated (cf. 5.3.1).

2. *Lack of Data.* When the form of the model seems appropriate but accuracy is still not satisfying acquiring more data is the appropriate step. New data can be acquired locally or globally.

3. *Inappropriate Transformation.* If the residuals vs. variable plot shows strong correlations this means, that the variable has not been appropriately transformed. As described above, power transformations are performed by the algorithm automatically. If a different transformation f_a is underlying, this has to be found manually however.

4. *Non-multiplicative Interaction Terms.* The presented algorithm can find interaction terms only if they are multiplicative. Other interactions cannot be automatically detected. Non-multiplicative interaction terms lead to a high degree of scattering in the variable vs. residual plot. If such an interaction is suspected, it should be included among the candidate variables.

5. *Missing variables.* Missing variables lead to globally increased errors. Including missing variables during data abstraction will obviously solve the problem. The most difficult problem consist in identifying the missing variable. It should be noted, that quite commonly not all variables that could improve the model are included. Most often this happens for cost reasons as generating the necessary observations might be exceedingly expensive.

5.5.1 Piecewise Modeling

Size scalable macros may change the internal configuration at certain size boundaries. This leads to jump discontinuities in the observed power trends. Figure 5.5A exemplifies such a situation for the most simple case (only one input dimension and linear regression). Discontinuities such as these can best be handled by piecewise modeling: the original model is split into two independent ones along a defined line, i.e. the line of discontinuity (figure 5.5B). As described in section 5.3.1 piecewise modeling can be represented in the suggested model family by introducing discriminator variables. The biggest problem in applying piecewise modeling therefore remains the identification of the necessary cut lines (also called *nodes*). In this section a simple technique is proposed for the detection of co-axial nodes, i.e. lines of the form $x_i = \varphi$ (e.g. the vertical line in figure 5.5B and D). Many other forms of cuts, including interactions of variables, are imaginable. The restriction to co-axial cuts was made here to limit the search space. As the behavior of a model is unknown between the levels, it is sensible to furthermore restrict the positions of possible nodes

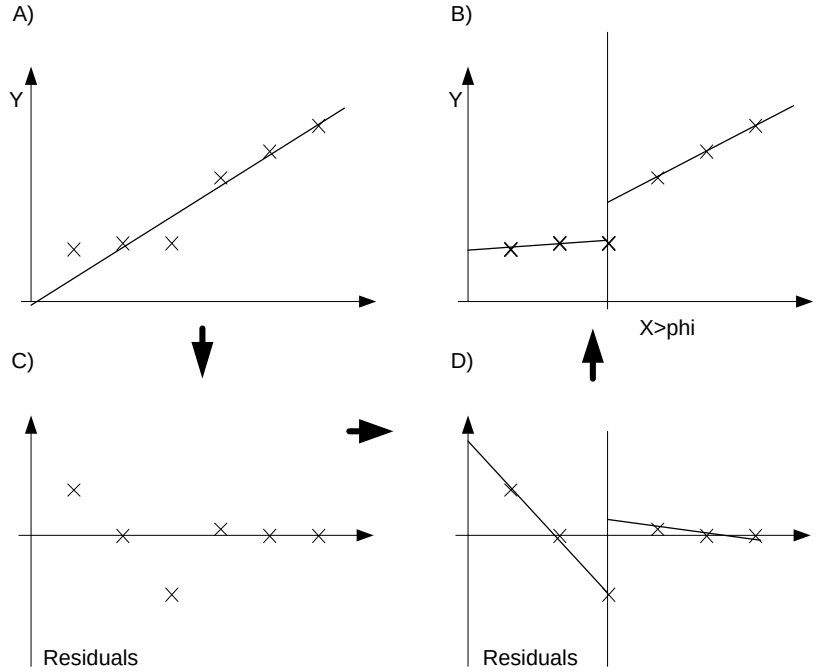


Figure 5.5: The steps of node insertion: A) Initial Regression, B) Model including node, C) Analysis of Residuals, D) Regression on residuals.

for each variable to its levels. The number of candidates is then the sum of the levels of all variables. This sum is bounded from above by the product of the number of observations n and the number of variables k. Even in complex modeling situations it is possible to enumerate all candidates and evaluate them.

Performing a complete model generation with every candidate would be too time consuming. Therefore the following scheme for comparing the candidate nodes was applied: Suppose an initial model has been fitted to the data (5.5A). A co-axial discontinuity, such as a jump or sudden change in slope at a specific value of a variable x_i, is then detectable in the residuals vs. variable plot of x_i (5.5C). The key idea of the evaluation of cut-line candidates is to perform a one-dimensional piecewise linear regression *on the residuals* (5.5D) and let the mean square error serve as its quality measure.

Let $\boldsymbol{r}$ be the residuals of an initial modeling run and x_i the variable under investigation with the levels $levels(x_i)$. Let $\varphi \in levels(x_i)$. Now regress this linear model:

$$\boldsymbol{r} = \beta_0(\boldsymbol{x_i} < \varphi) + \beta_1 \boldsymbol{x_i}(\boldsymbol{x_i} < \varphi) + \beta_2(\boldsymbol{x_i} \geq \varphi) + \beta_3 \boldsymbol{x_i}(\boldsymbol{x_i} \geq \varphi) \tag{5.25}$$

The mean square error of this regression can be seen as a first order approximation of the remaining error after the introduction of the respective cut line. Based on this approximation a candidate variable/level pair for the cut can be identified as one that produces small errors when applying eqn. 5.25.

Limits of eqn. 5.25 become clear when iterating model generation and node insertion: the regression model does not take into account the interaction of discriminator variables. This is best explained by an example:

Example 7. Let a hidden relationship:

$$y = \begin{cases} 0.1 \cdot x_1 + 0.2 \cdot x_2 + 2 & \text{if } x_1 \geq 6, x_2 \geq 6 \\ 0.1 \cdot x_1 + 0.2 \cdot x_2 & \text{otherwise} \end{cases} \tag{5.26}$$

Figure 5.6 shows the graph of this function. The discontinuities at the value 6 on both axis is clearly visible. The figure also shows a piecewise linear model fitted to the relationship. This *pre-cut* model has

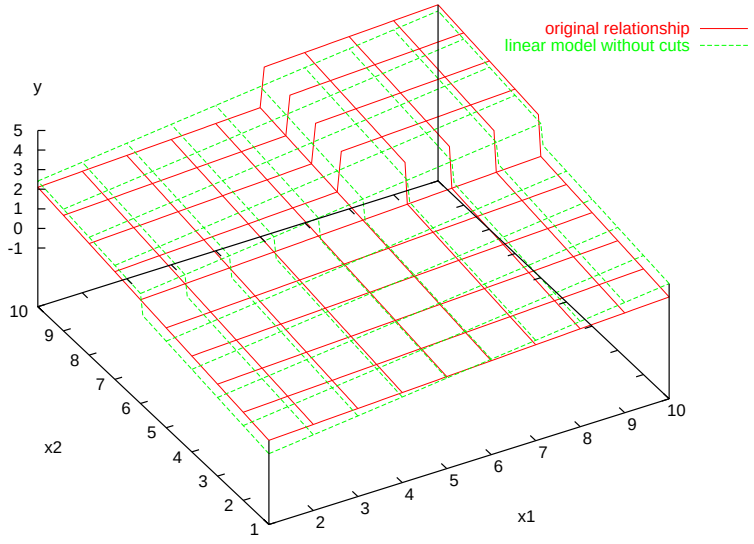


Figure 5.6: A two dimensional relationship and its best linear approximation with one cut in x_2.

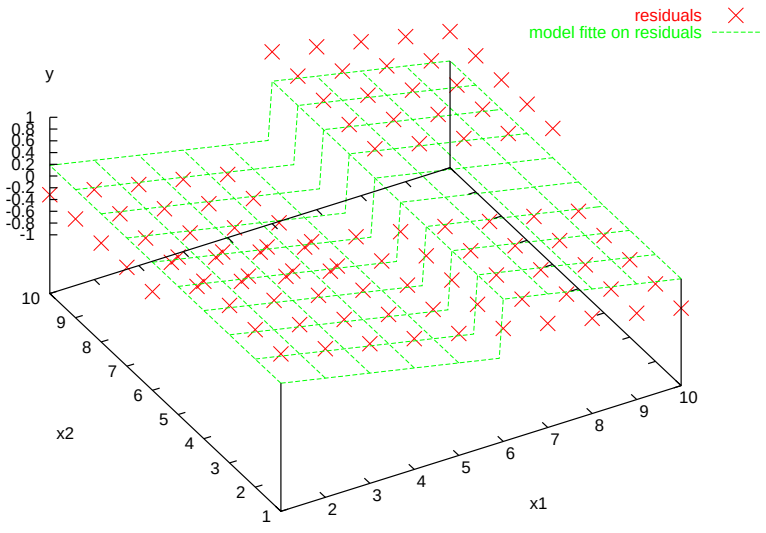


Figure 5.7: Residuals of the model shown in figure 5.6 and a piecewise linear model of these residuals with cut in x_1.

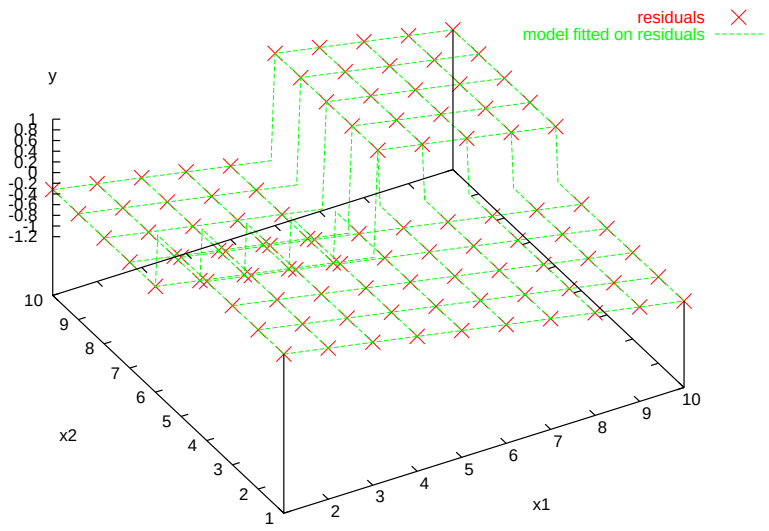


Figure 5.8: Residuals of the model shown in figure 5.6 and a piecewise linear model on these residuals using an interaction of the two cuts.

a node along $x_2 = 6$ which we assume was found in the previous cut insertion run. The pre-cut model is the point of departure for the insertion of the next node. Figure 5.7 shows its residuals. According to the technique suggested above a piecewise linear model of the form described in eqn. 5.25 is fitted to these residuals (x_1 is the variable under investigation here). This *residual* model is also shown in the figure.

An analysis of the residuals and their model reveals discrepancies: The residuals are clearly divided into four quadrants along the lines $x_1 = 6$ and $x_2 = 6$. Since the residual model contains only one node, it cannot adequately represent these quadrants. A significant error is the result. Does that mean that introducing the discriminator variables $x_{d_2} = (x_1 < 6)$ and $x_{d'_2} = (x_1 \geq 6)$ would not have a big benefit? No. In fact it would reduce the error to zero. This is the case as the discriminator variables $x_{d_1} = (x_2 < 6)$ and $x_{d'_1} = (x_2 \geq 6)$ are already in the pre-cut model. By combining the two new and the two old variables through the interaction terms $x_{d_1} \cdot x_{d_2}$, $x_{d_1} \cdot x_{d'_2}$, $x_{d'_1} \cdot x_{d_2}$ and $x_{d'_1} \cdot x_{d'_2}$. all four quadrants could be handled separately. In the case of eqn. 5.26 this would result in zero error.

The observation just described means that in cases of repeated cuts the error fitting eqn. 5.25 to the residuals is not a good predictor because it neglects interactions of nodes. A solution to this problem can also be demonstrated on the example above: by including the node $x_2 = 6$, that was already in the pre-cut model, into the residual model, the residual model can effectively forecast the impact of the interactions of the nodes (see figure 5.8). This leads to the more general requirement: the model on the residuals should contain all nodes already in the pre-cut model. A respective regression model can be generated by the method described in algorithm 6 (the '$\circ$' symbol stands for a multiplicative concatenation of terms).

Unfortunately the number of pieces in a piecewise model grows worst case exponentially. For algorithm 6 this means that the models D can become very big. Models with many terms however cause several problems ("overfitting"). A natural measure is therefore to restrict the number of considered variables by a maximum u. In case this maximum is exceeded, two strategies are possible:

1. Only the first u nodes are considered for every new cut. Later nodes are ignored. The legitimation for this approach is that the nodes entering the models first have the biggest impact.

2. When considering i nodes, $i > u$, all $\binom{i}{u}$ possible combinations of u nodes are tested. This is the more thorough approach, but time complexity can limit its applicability as the number of cases again grows exponentially.

Input: $(x_i < \varphi), x_i \in V, \varphi \in levels(x_i)$ the cut under investigation.
$C \subset \{(x < \varphi) | x \in V, \varphi \in levels(x)\}$
Output: D is set of all summands in the model
$D = \{(x < \varphi), x(x < \varphi), (x \geq \varphi), x(x \geq \varphi)\}$
for all $(c < \gamma) \in C$ **do**
$D_{tmp} \leftarrow \emptyset$
for all $d \in D$ **do**
$D_{tmp} \leftarrow D_{tmp} \cup (c < \gamma) \circ d$
$D_{tmp} \leftarrow D_{tmp} \cup (c \geq \gamma) \circ d$
end for
$D \leftarrow D_{tmp}$
end for

Algorithm 6: Generation of regression models for cut searches. The '$\circ$' denotes concatenations.

Algorithm 7 exemplifies the second solution. Note that the first solution can be emulated by substituting the second innermost loop by $W = C$. The input C must only contain the first u nodes in that case.

Input: $C \subset \{(x_i < \varphi) | x_i \in V, \varphi \in levels(x_i)\}$ set of previous nodes
initialmse: mean square error of previous model
Output: *bestcut* is candidate node for separating the previous model
bestmse $\leftarrow$ *initialmse*
bestcut $\leftarrow \emptyset$
for all $x \in V$ **do**
bestmse(x) $\leftarrow$ *initialmse*
for all $W \subseteq C, |W| = u$ **or** $W = C$ **do**
for all $\varphi \in levels(x)$ **do**
built regression model with W and $(x < \varphi)$ using algorithm 6
compute MSE by regression
if $MSE >$ *bestmse*(x) **then**
bestmse(x) $\leftarrow MSE$
bestlevel(x) $\leftarrow \varphi$
end if
end for
end for
if *bestmse*$(x) >$ *bestmse* **then**
bestmse $\leftarrow$ *bestmse*(x)
bestcut $\leftarrow (x <$ *bestlevel*$(x))$
end if
end for

Algorithm 7: Suggestion of co-axial cut lines for piecewise modeling. Testing all u element cut subsets.

Algorithm 8 describes the complete interactive interplay of model generation and search for nodes. This algorithm is also implemented in the modeling tool (see next chapter). Two stopping criteria are used: Failure to include a suggested discriminator variable and suggestion of a node already in the model. Both criteria indicate that the last model cannot be significantly improved anymore. The highest number of cuts inserted into a model presented within this thesis was 10 (cf. chapter 7).

Iteratively slicing the model space tends to render models with decreasing errors but increasing complexity. A trade-off between these two therefore has to be performed. For this purpose algorithm 8 protocols the accuracy and complexity of all intermediate models. The selection is left to the user. The decision can be taken based on the errors in relation to given bounds, the visually perceived adequacy, and the

remaining degrees of freedom of the error.

Input: set V of candidate variables, y response variable with n observations
Output: sequence of models and statistical error measures
repeat
perform model generation as documented in algorithm 5
perform validation by computing statistical error measures
protocol current model and errors
identify tentative partitioning using algorithm 7
until the last partitioning was not used in the model **or** the suggested partitioning is already in the model

Algorithm 8: Model generation including iterative partitioning for piecewise modeling.

One drawback of the model segmentation must be mentioned: segmentation reduces the region of definition of the model. Let x a regressor variable, l_{i-1} and l_i the $i-1$-th respectively i-th level of x. If now the model is cut at the node $x < l_i$, the highest level in the 'lower' segment is l_{i-1} and the lowest level in the 'higher' segment is l_i. Although the lower segment, by its defintion, extends up to l_i no observation is available above l_{i-1}. The behavior of the lower segment model is therefore in reality undefined between l_{i-1} and l_i. Predictions in that region may have random errors.

5.6 Optimization

As mentioned in the introduction optimization as a major application for memory models. A simple example of optimization is the determination of the power optimal aspect ratio (height x vs. width y) for a memory of given size. Mathematically the global optimum of a cost function $P(x, y)$ is sought:

$$V_{mem} := \quad \text{minimize} \qquad P(x, y) \tag{5.27}$$

$$\text{subject to} \qquad x \cdot y > size \tag{5.28}$$

To support optimization, models must be in a mathematical form that allows to efficiently compute their global optima subject to constraints. This is not a trivial requirement: the global optimum of general nonlinear functions cannot be effectively computed even if they are given as closed form equations [18,152]. In such cases, exhaustive search is the only way to obtain at least a near optimal solution. This section describes how global optimization *can* be efficiently performed for the family of mathematical models chosen here. By doing so an integrated methodology from the gathering of observations to the solving of optimization problems becomes available. This circumstance in turn justifies the choice of signomials as mathematical models.

Signomials (or more specifically posynomials) are among the most general sub-classes of nonlinear functions for which an effective global optimization technique exists. The respective technique is called *geometric programming*. Because of their high optimization potential, signomials and posynomials have in recent years been used in many fields, including the optimization of analog circuit properties (e.g. [54,87]). So far most signomial/posynomial models have been produced analytically, i.e. by circuit analysis. [87] itself states that "even the one-time user effort required to derive the analytic expressions of performance metrics is a barrier to the widespread use of techniques such as ours."

The combination of empirical regression analysis and posynomials for hardware optimization has only been published very recently [33]. In [33] second order polynomial regression is performed on the performance of analog circuits. The resulting model is subsequently transformed into a posynomial. Geometric programming is then applied to obtain an optimal sizing.

In the remainder of this section it is briefly described how geometric programming can be used to solve signomial optimization problems. First however a simpler, albeit less compact, representation of monomials and signomials is introduced. This representation is only valid for non-extended signomials

(cf. section 5.3.1). The basic idea of the new representation is to include *all* variables in *all* monomials. If a monomial does not really depend on a specific variable, the variable's exponent is set to zero, effectively turning the variable into a constant one (as $x^0 = 1$). Consequently instead of writing:

$$\beta_i \cdot x_{l_{i,1}}^{\alpha_{i,1}} \cdot \ldots \cdot x_{l_{i,p_i}}^{\alpha_{i,p_i}} \tag{5.29}$$

now it is simply written

$$\beta_i \cdot x_{i,1}^{\alpha'_{i,1}} \cdot \ldots \cdot x_{i,m}^{\alpha'_{i,m}} \tag{5.30}$$

where:

$$\alpha'_{i,j} = \begin{cases} \alpha_{i,r} & \text{if} \quad \exists l_{i,r} : l_{i,r} = j \\ 0 & \text{otherwise} \end{cases} \tag{5.31}$$

Note that this writing can also represent the constant offset (by setting all α's to zero). The *primal posynomial geometric program* is now given as:

$$\begin{aligned} V_{GP} := \quad & \text{minimize} && g_0(t) && && (5.32)\\ & \text{subject to} && g_k(t) \leq 1, && k = 1, 2, \ldots, p && (5.33)\\ & && t_i > 0, && i = 1, 2, \ldots, m && (5.34) \end{aligned}$$

where

$$g_0(t) = \sum_{j=1}^{n_0} \beta_j \cdot t_{j,1}^{\alpha_{j,1}} \cdot \ldots \cdot t_{j,m}^{\alpha_{j,m}} \tag{5.35}$$

$$g_k(t) = \sum_{j=n_{k-1}+1}^{n_k} \beta_j \cdot t_{j,1}^{\alpha_{j,1}} \cdot \ldots \cdot t_{j,m}^{\alpha_{j,m}} \tag{5.36}$$

Here, $g_0(t)$ is called the *objective function* and $g_k(t)$ the *kth constraint function*. Each constraint function consists of $(n_k - n_{k-1})$ monomials and the total number of monomials is n_p. In the terms of this thesis $g_0(t)$ is the power model, whereas the $g_k(t)$ contain additional constraints (e.g. $x \cdot y > size$ in the example above). The primal program can be solved by solving the dual. The *dual posynomial geometric program* is:

$$\begin{aligned} V_{GP} := \quad & \text{maximize} && \prod_{j=1}^{n_p} (\beta_j / x_j)^{x_j} \prod_{k=1}^{p} \lambda_k^{\lambda_k} && && (5.37)\\ & \text{subject to} && \sum_{j=1}^{n_0} x_j = 1, && && (5.38)\\ & && \sum_{j=1}^{n_p} x_j \cdot \alpha_{i,j} = 0, && i = 1, 2, \ldots, m && (5.39)\\ & && x_j \geq 0, && i = 1, 2, \ldots, n_p && (5.40) \end{aligned}$$

where

$$\lambda_k = \sum_{j=n_{k-1}+1}^{n_k} x_j, \quad k = 1, 2, \ldots, p. \tag{5.41}$$

For a primal geometric program (GP) having m independent variables (t_i), p constraints and n_p monomials the dual has n_p non-negative variables (x_j) in m+1 linear equations. The degree of difficulty a of GP is therefore defined as:

$$\text{degree of difficulty} = n_p - m - 1 \tag{5.42}$$

The interesting point about the dual problem is that its objective function can be turned into a *convex* function by taking the negative of the logarithm:

$$F(x) = \sum_{j=1}^{n_0} x_j \cdot \ln\left(\frac{x_j}{c_j}\right) + \sum_{k=1}^{p} \sum_{j=n_{k-1}+1}^{n_k} x_j \cdot \ln\left(\frac{x_j}{c_j \lambda_k}\right) \tag{5.43}$$

By this transformation the dual problem becomes an astonishingly simple linearly constraint convex programming problem

$$\text{minimize} \qquad F(\boldsymbol{x}) \tag{5.44}$$

$$\text{subject to} \qquad \mathbf{A}\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \geq 0 \tag{5.45}$$

The coefficient Matrix $\mathbf{A}$ is

$$\begin{bmatrix} 1 & \cdots & 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \alpha_{1,1} & \cdots & \alpha_{1,n_0} & \alpha_{1,n_0+1} & \cdots & \alpha_{1,n_1} & \cdots & \alpha_{1,n_{p-1}+1} & \cdots & \alpha_{1,n_p} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \alpha_{m,1} & \cdots & \alpha_{m,n_0} & \alpha_{m,n_0+1} & \cdots & \alpha_{m,n_1} & \cdots & \alpha_{m,n_{p-1}+1} & \cdots & \alpha_{m,n_p} \end{bmatrix} \tag{5.46}$$

and

$$\boldsymbol{b}^t = [1, 0, \ldots, 0] \in \mathbb{R}^{m+1} \tag{5.47}$$

This problem can be solved by existing techniques (e.g. *interior-point algorithms*). Their detailed description is beyond the scope of this thesis and can be found for example in [77]. Software implementations of thesis algorithms are available [31].

The geometric program described in eqn. 5.32 does not incorporate equality constraints. These can however be emulated by inequality constraints. Let

$$h_k(t) = \beta_j \cdot t_1^{\alpha_{1,j}} \cdot \ldots \cdot t_m^{\alpha_{m,j}} = 1 \tag{5.48}$$

This can be written as:

$$h_k(t) = \beta_j \cdot t_1^{\alpha_{1,j}} \cdot \ldots \cdot t_m^{\alpha_{m,j}} \leq 1 \tag{5.49}$$

$$\frac{1}{h_k(t)} = \frac{1}{\beta_j} \cdot t_1^{-\alpha_{1,j}} \cdot \ldots \cdot t_m^{-\alpha_{m,j}} \leq 1 \tag{5.50}$$

Note that while in theory this makes the interior-point algorithm unstable, the algorithm still works fine in most cases [77].

Unfortunately, signomial optimization problems are not directly solvable via geometric programming. To solve them nevertheless, signomials can be approximated by posynomials. This is achieved by a sequence of preprocessing steps that substitute all negative coefficients β_i. This preprocessing is technically rather complex and will therefore not be described here. For more detail see [8].

In summary this section has shown that the signomial functions generated by the presented model building methodology are well suited for optimization. With geometric programming, a broad body of optimization theory and software exists.

5.7 Rounding

The real valued exponents of signomials have one principle disadvantage: as the real valued power function is much more complicated to compute than the integer valued one, the computation of general signomials is significantly slower than that of polynomials. Polynomial models on the other hand are potentially less accurate than signomials.

In principle it is possible to obtain polynomials by simply rounding the signomials resulting from above. The exponents are rounded to integer values, or $\pm\frac{1}{2}$ (square roots). This simple rounding however would mean sacrificing accuracy: the structure of the optimal polynomial model is likely to differ from the

structure of the signomial one. To be able to produce polynomial models as well as signomials within the flow presented above a different modification was therefore made to algorithm 5: Instead of rounding only the final model, the monomials are rounded after each exponent adaption (*box_tidwell*) step. By this measure the computation of statistical significance for the in-/exclusion of variables does take into account the effects of rounding. The exponent rounding scheme just described has been implemented as an option in the model generation tool described in the next chapter.

5.8 Summary

In this chapter a general empirical modeling technique for memories was proposed. Circuit and gate level simulation as well as compiler estimates can be used for data acquisition. The model generation relies on the key ideas of signomial models, input variable transformation (linearization) and multi-linear regression. Model validation is conducted using statistical error measures and visual inspection. The modeling flow contains a number of possible iterative loops for the improvement of models. Piecewise modeling is explicitly supported by a co-axial cut search algorithm. Emphasize was laid on an extensive automation of the modeling process. Remaining manual steps were designed so as to require as little knowledge as possible in the areas of statistics and model building. It was shown that the mathematical form is one of the most general forms, for which effective optimization techniques exist and therefore the models are well suited for the application in optimization.

6 Implementation

As already mentioned in the introduction the modeling flow has been implemented as proof of concept in the prototype tool ORINOCO BEACH. Furthermore the application of the resulting models has been realized as part of the power estimator ORINOCO DALE. Both tools are part of the behavioral level power estimation suite ORINOCO. This chapter will briefly document the aspect of the implementation: section 6.1 gives an overview of the ORINOCO functionality, section 6.2 addresses the model generation tool BEACH while section 6.3 explains the integration of models into the estimator DALE.

6.1 ORINOCO Tool Suite

ORINOCO is a behavioral level power estimation and optimization tool suite comprising of three tools: the estimator/optimizer DALE and the modeling tools RIO and BEACH. ORINOCO can be used on C/C++ as well as VHDL descriptions. The functionality of ORINOCO can be best explained following the tool flow (see figure 6.1): at the beginning the source code description is compiled with an ORINOCO specific C++ compiler (see [1] in figure). This modified GNU compiler generates a control/dataflow file[2] (proprietary format) and instruments the application by introducing additional statements into the program's object code[3]. Subsequently the instrumented program is executed with typical input data streams provided by the user[4]. During execution the added statements produce a background file containing an activity profile, i.e. a representation of the data streams flowing through the application. With these preliminaries the estimator DALE is started. DALE reads the control/dataflow file and builds an internal control/dataflow graph-like representation[5]. Next the activity data is read in and annotated to the respective nodes of the graph[6]. The tool is now ready for the estimation itself. The estimation[7] forecasts an RT-level architecture resulting from the algorithm and computes the activity of each functional unit based on this architecture and the activity profile. Then power models for the different structural components are applied to obtain the power estimate. These steps are repeated and folded to be able to optimize the forecast architecture for power. Once the process is finished DALE produces several graphical and textual views of the power estimation results as well as the optimized architecture.

The purpose of RIO and BEACH is to provide models of RT-level blocks to the estimator. RIO focusses on combinational data path components like adders and multipliers. It uses a fixed type of data abstraction and a combination of interpolation and regression [70, 71, 143] (see also section 4.2.1). BEACH as a black box modeling tool primarily serves the modeling of memories within ORINOCO. Evaluation results (cf. chapter 7) encourage the use for other components, like IP blocks.

6.2 ORINOCO BEACH

As the different algorithms involved in the model generation have already been thoroughly described, this section will not go into detail about the implemented code. Instead, section 6.2.1 only describes the implementation framework, i.e. languages and tools used. Section 6.2.2 briefly documents the usage and tool flow of BEACH.

6.2.1 Implementation Framework

The body of BEACH is implemented in C++ and the script language Tcl/Tk: The Tcl/Tk interpreter has been extended using its C procedure interface [148,109] to recognize the commands of the application.

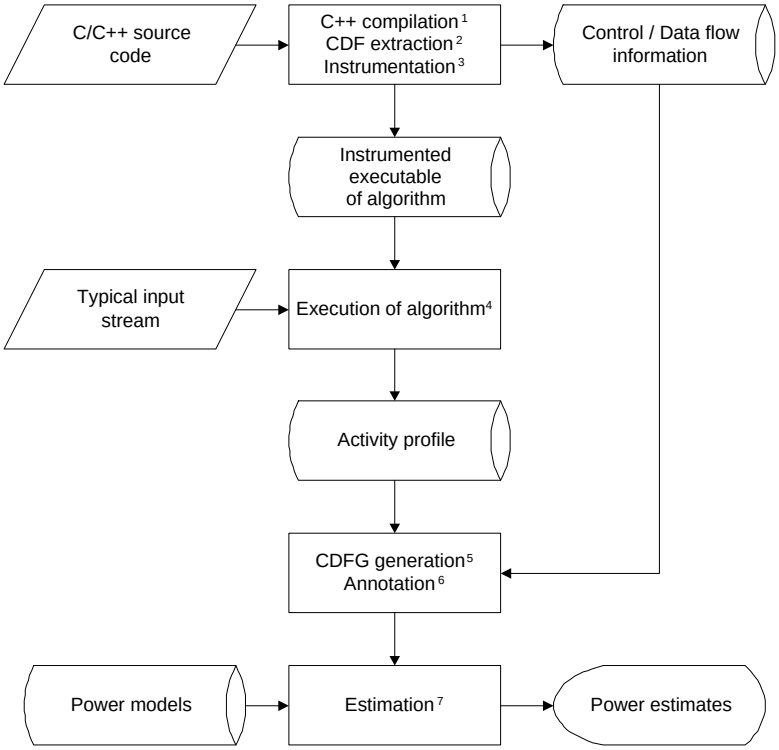


Figure 6.1: ORINOCO power estimation flow.

The graphical user interface has been completely implemented in Tcl and Tk. This makes it possible to operate the tool with graphical support in an interactive fashion as well as in batch mode. To limit the necessary implementation effort, a couple of public domain tools were integrated: For the statistical computations BEACH relies on the statistics software Macanova, developed at the statistics institute of the University of Minnesota [14]. Macanova offers base functionality for regression, test statistics and the manipulation of data structures. It is accessed via a file interface. For the visualization of modeling results BEACH uses Gnuplot [150]. This de facto standard tool is controlled by exporting data and command files. Last but not least some file processing functionality is performed by the use of GNU AWK, a script language dedicated to the processing of text files [116].

6.2.2 Tool Flow

The section describes the tool flow of ORINOCO BEACH (see figure 6.2) [122]. The modeling flow is represented in the tool as a set of five so called 'views': *Characterization*, *Data*, *Terms*, *Variables* and *Model*. Each view reflects an aspect of the modeling and is manipulated on a different screen. The flow is executed by proceeding through the views linearly.

The *characterization view* is an editor that allows the manipulation of characterization scripts. A simple language has been developed (see appendix B) to specify the experimental design. A parameterizable command string is executed for every experiment allowing for example to call a memory generator and subsequently postprocess its output. While specifying the experimental design it is possible to already plan for re-iterations of the characterizations, should they be necessary. A special FOR-statement allows the definition of sequences of decreasing step size. The initial characterization is performed with the first step size; every new execution then automatically selects the next smaller step size. In this way the resolution of the characterization can be gradually increased.

When the characterization command is issued, the characterization script is executed and all its outputs to the standard output are redirected into a data view. Eventually the *data view* contains the acquired

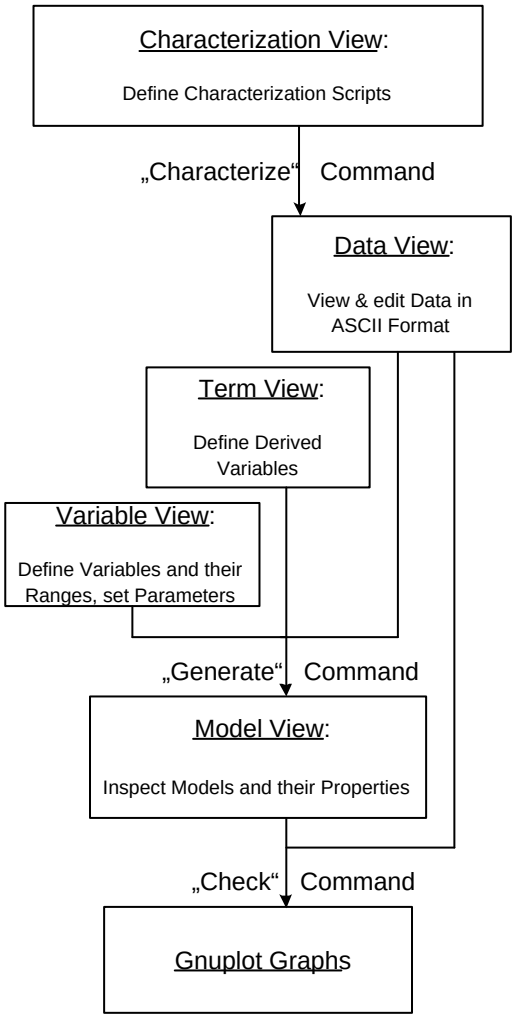


Figure 6.2: ORINOCO BEACH tool flow.

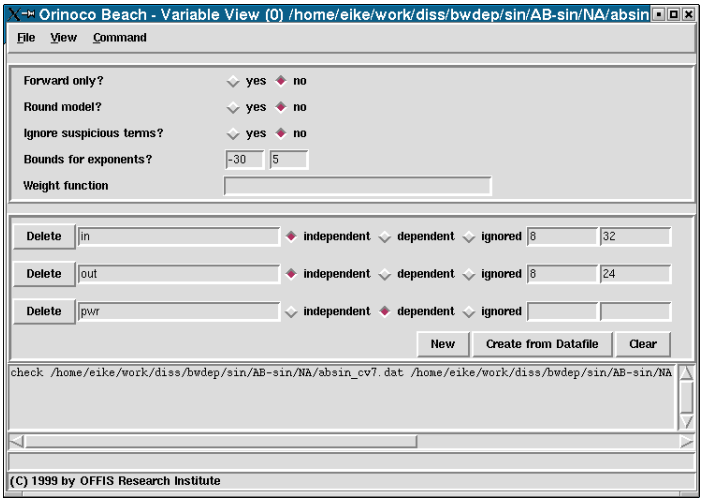


Figure 6.3: ORINOCO BEACH variable view.

data in a simple, ASCII table format. The tool flow can also be directly entered by reading such a table into BEACH. In the data representation names are given to the individual columns of data. Furthermore the cross validation and regression data sets are selected here. BEACH supports random, equidistant and manual selection.

Artificial variables derived from the acquired data are defined in the *terms view*. Expressions for the computations are given in Macanova syntax. Parameters of the model generation and validation can be set in the *variable view* (see figure 6.3). The parameters are:

1. *Variables*. The independent and dependent variables are declared here. Additionally a range can be set for each variable defining the model region plotted during visual inspection.
2. *Weight Function*. To perform weighted regression a weight function can be entered. This allows for example to minimize relative instead of absolute errors.
3. *Lower and upper bounds for exponents*. As described in 5.3.5 bounds must be set to the exponents in the linearization procedure to ensure numerical stability.
4. *Rounding*. This option allows the exponents to be rounded producing model expressions that are more easily interpreted and evaluated. The bottom side of this measure is a potential loss of accuracy.
5. *Forward only*. In the variable selection process it is possible to omit the backward elimination. The resulting pure forward selection is an alternative variable selection method documented in literature [45].

After both, the artificial variables have been defined and the parameters have been set, the generate command can be issued to build a model from the data. Once the generation has finished, the model expressions, all error measures and the computation time used can be investigated in the *model view* (see figure 6.4). The models are also stored in C format. The check command may now be issued to plot the model and the acquired data against each other. The visible section is defined in the variables view (see above).

Based on the diagnosis of the errors, the different feedback loops discussed in section 5.5 may now be initiated, returning to the characterization, data, terms or variables view.

6.3 Model Integration

The practical application of models is an important aspect of modeling (see section 3.1). This section highlights three central topics in the implementation and integration of the models: the choice of model representation, the assignment of models to source code operators and the dynamic interaction between estimator and model. The initial concepts of the integration of the power models into the ORINOCO DALE estimator were drafted together with G. von Cölln [143] and are developed further in this thesis.

6.3.1 Model Representation

The choice of the software representation of the power models has a high impact on the estimation efficiency and flexibility. In ORINOCO power models are described using the C++ language. Models are C++ classes inherited from a common base class. This base class specifies the interface between estimator and models. In case of BEACH the model equations are directly generated in C++ syntax. The model interface is open: the interface class and its documentation as well as a set of templates are distributed with the estimator. This enables users to supply their own models. The libraries of C++ represent a natural model library concept: The separate model classes are compiled and linked into dynamic libraries. One or several of these dynamic libraries are then loaded into the estimator at runtime. This approach ensures the necessary power of expression, programming infrastructure (tools and documentation) and runtime efficiency.

As an alternative model description language the *Advanced Library Format* (ALF) has evolved over recent years. This meta-language is an attempt to unify all model description forms into a common

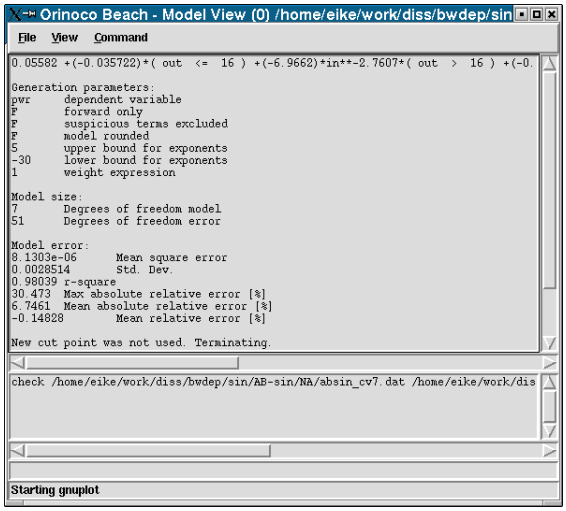


Figure 6.4: ORINOCO BEACH model view.

standard: "The fundamental purpose of ALF is to serve as the primary database for all third-party applications of ASIC cells. In other words, it is an elaborate and formalized version of the databook" [118]. ALF 2.0 is an approved Accellera standard since 2001 and approaches standardization by the IEEE (as draft standard P1603). ALF offers modeling capabilities for functionality as well as electrical (*arithmetic models*) and physical properties (*geometric models*) of technology cells and circuit sub-blocks. It is intended to be used in design planning (RTL partitioning, floor plan, pin assignment) and design implementation (synthesis, as well as layout, timing, power and signal integrity optimization) [117]. A power macro model for a memory could for example have the following form:

```
CELL my_memory {
    PIN[0:2] Addr { DIRECTION = input; }
    PIN[1:4] Dout { DIRECTION = output; }
    VECTOR (?! Addr -> ?! Dout ) {
        ENERGY {
            HEADER {
                SWITCHING_BITS sba { PIN=Addr; }
                SWITCHING_BITS sbd { PIN=Dout; }
            } EQUATION {
                0.4 + 0.2*LOG(sba) + sbd
}   }   }   }
```

ALF was not initially chosen as a model interface for ORINOCO as it formerly had a low-level (i.e. technology cell oriented) focus. In version 2.0 still not all operators necessary for RTL data stream analysis (e.g. signal distance) are incorporated. Furthermore at the time of decision the standard was still floating and no parser was available for the language. Last but not least the models are in textual format that needs to be interpreted by tools at the cost of performance.

With the standard now nearing completion and the availability of a public domain ALF parser since the end of 2001, a possible future scenario could involve an ALF to ORINOCO translator: ALF models would be processed by this tool to produce ORINOCO compatible model source code, which could then be compiled and linked into ORINOCO. This strategy would combine advantages of the ALF standard with those of the ORINOCO model representation.

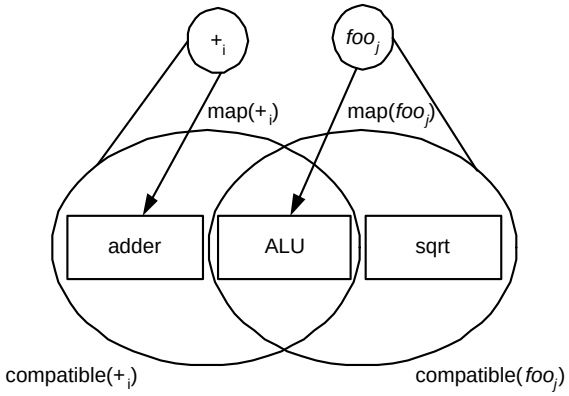


Figure 6.5: Assignment of model to operator.

6.3.2 Operator-Model Relationships

In the DALE estimator each source code operator (e.g. '+', '*', but also function calls 'call foo') is mapped onto an estimation model (e.g. model of a ripple adder). An important aspect in the integration of models is to define the relationship between operators and the models. Put in simple words the question is: for a certain operator which model is suitable to estimate its properties? Let $O = \{o_i\}$ the occurrences of the operators to be estimated. O can contain standard operators, such as additions, but also function calls that abstract more complex operations (e.g. fft-filter). Let $M = \{m_i\}$ the available models. The suitable models can then be described by defining a function $compatible : O \rightarrow \mathbf{P}M$ that assigns the set of compatible models to each operator occurrence (see figure 6.5). From the set of compatible models eventually one model has to be selected for the estimation. This is expressed by the mapping function $map : O \rightarrow M$.

For the operator/model assignment defining *compatible* is the key problem: while it is intuitive that the operator '+' can be mapped to a model 'adder', it is for example not clear, whether the same applies for the model 'ALU'. Furthermore even if we 'know' what the ALU can do, it remains unclear whether foo can be mapped to it. While in the former scenario the semantics of 'ALU' is unclear, the same is the case for the function call to foo in the latter case. As a further complication neither the operator occurrences O nor the available models M are statically known: O varies with the circuit under investigation, M might change by the addition or removal of estimation models/libraries. The relationship between O and M can therefore not be statically fixed.

The approach chosen in ORINOCO to fix these problems is the introduction of an additional level of entities between operators and models. This level represents logic functionalities. Let $S = \{s_i\}$ the set of logic functionalities (see figure 6.6). Let $op_f : O \rightarrow S$ the definition of logic functionality for each operation and $mod_f : M \rightarrow \mathbf{P}S$ the functions that each model is able to perform. Then $compatible = mod_f^{-1} \circ op_f$. The advantage of introducing the s_i is that operators and models can rendezvous on them: mod_f is defined by the model designers, who know the logic functionalities the models can perform. If a new model m_i is generated that performs a formerly non-existent functionality s_{new} (e.g. IP modeling), this new functionality can be implicitly declared by including it in $mod_f(m_i)$. The available functionalities in the estimator is then defined by the models loaded M_{loaded} through $S_{available} = \bigcup mod_f(M_{loaded})$. Conversely functionalities can be assigned to the operators freely. Formerly nonexistent functionalities are again implicitly declared. The set of required models is consequently: $S_{required} = op_f(O)$. Only if all required functionalities are covered by the loaded models, i.e. $S_{required} \subseteq S_{available}$, the mapping may be performed as described above. Otherwise a warning is produced and the respective operators are ignored in the estimation.

Example 8. The user assigns the functionality 'square root' to the call of function foo in the source code. 'square root' therefore becomes a member of the required functionalities 'square root' $\in S_{required}$. Supposed among the models is a CORDIC element. This element has 'square root' among its functionalities (square root $\in mod_f$(CORDIC)), but also 'vector-length', 'arg' and others. Once the library containing

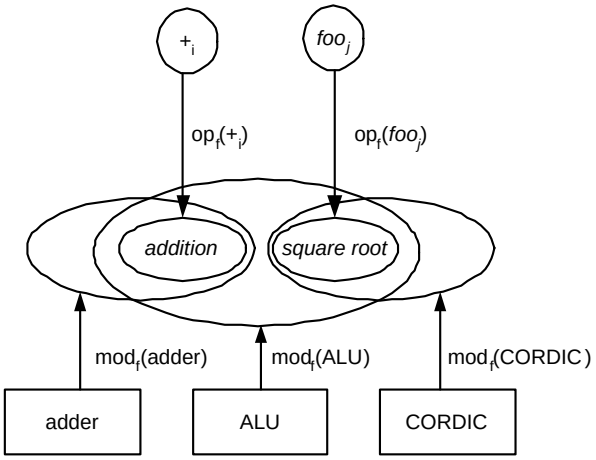


Figure 6.6: Defining the set of compatible models by the introduction of logic functionalities.

the CORDIC model is loaded, 'square root' is also in $S_{available}$. The call to foo may therefore be estimated and CORDIC is among the compatible models (CORDIC $\in$ $compatible(foo)$).

Introducing S as described allows to define the sets of compatible models extensible. The method relies on a consensus between the person generating the model and the user of the estimator. This consensus can be supported by appropriate documentation.

6.3.3 Dynamic Model/Estimator Interaction

The model integration concept puts emphasize on the dynamic addition and removal of models as well as a high degree of model autonomy. These aspects as well as the application of the integration concepts described above, are best documented by explaining the run-time model/estimator interaction.

Point of reference for the models within the estimator is the *model manager*, a globally static object (see figure 6.7). The model manager maintains the *model catalogue* that keeps track of all loaded models (M_{loaded}) and the *catalogue of functionalities* containing all available as well as the required functionalities ($S_{available} \cup S_{required}$). For each model functionality pair this catalogue furthermore tracks the parameter mapping (cf. $M_{static}, M_{dynamic}$). On the side of the models each model class is represented by a *model maintenance object* (MMO). The MMO is a locally static object, which handles all class specific functionality and information, e.g. the instantiation of model objects.

A typical model library life-cycle now looks as follows:

1. Loading of dynamic model library
2. Automatic instantiation of one model maintenance object (MMO) for each model class (as locally static objects).
3. Registration of the model class at the estimator model catalogue by the MMO.
4. Registration of the MMOs at each functionality their models are able to perform. Implicit registration of all new logic functionalities.
5. Provision of model information as guidance for the mapping phase.
6. Instantiation of models through their MMOs.
7. Customization of the models static properties as defined by estimated source through O_{static}.
8. Repeated calls to estimation methods.
9. Destruction of model objects.

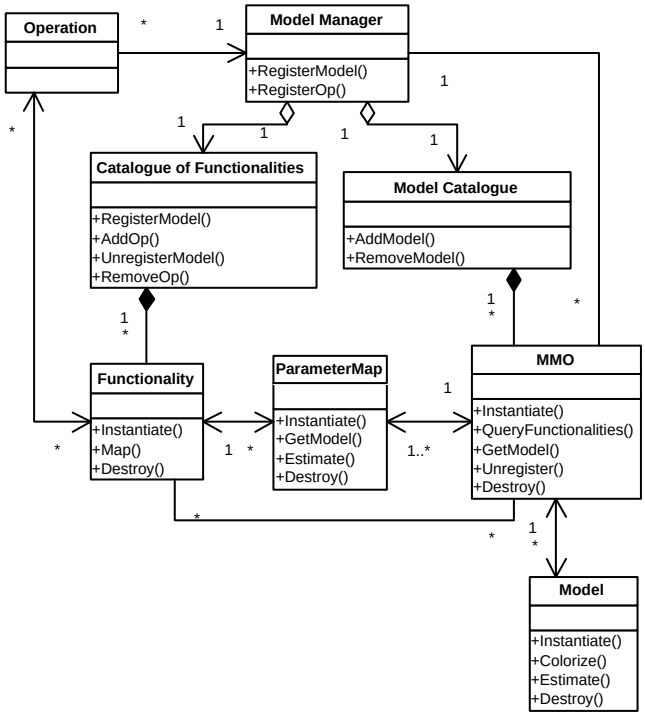


Figure 6.7: UML Class Diagram of the estimator/model interface [41].

10. Unloading of dynamic library. Implicit destruction of the MMOs. The MMOs in turn unregister the models from the estimator model catalogue and the functionality catalogue during their destruction.

The communication of the different objects involved is shown in the sequence diagram 6.8.

6.4 Summary

This chapter briefly described the ORINOCO tool suite, the implementation framework of ORINOCO BEACH, its tool flow and usage. Furthermore it was shown that a flexible and at the same effective integration of models into an estimation/optimization tool is not a trivial task. Three interacting concepts were described in detail: the model representation, the assignment of models to operators and the dynamic life-cycle of estimator models.

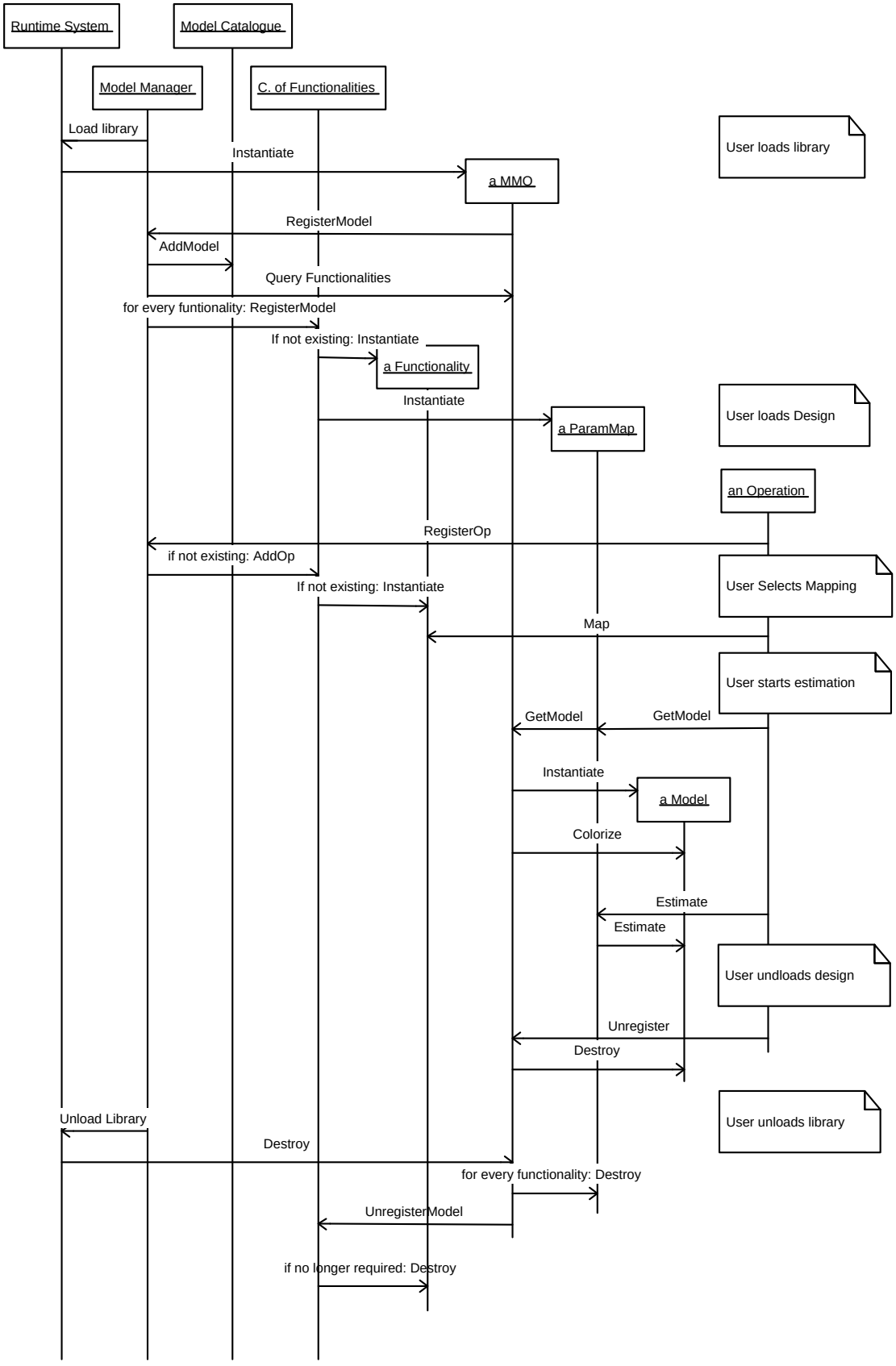


Figure 6.8: UML sequence diagram of the estimator/model interface [41].

7 Evaluation

This chapter contains the evaluation of the techniques presented in chapter 5. Three different application cases corresponding to the different characterization flows (cf. algorithms 1 to 3) are shown. In a fourth case the application of the proposed methodology on other domains of power modeling is exemplified. Unfortunately the availability of technology information was limited, so that neither DRAMs, caches nor flash memories could be investigated. The demonstrators are:

1. *Register files from the Synopsys DesignWare® library.* The characterization was performed by synthesizing the DesignWare soft-macros and performing gate-level simulations.
2. *Philips embedded SRAM, high-speed and low-power ROM.* In this case the characterization was conducted using circuit-level simulation of critical-path models. The experimental design could not be chosen freely as the majority of data sprang from existing Philips in-house simulations.
3. *LSI embedded SRAM.* The data in this case was obtained from estimates of the LSI memory compiler.
4. *Data-path soft-macros from the Synopsys DesignWare®.* To document the applicability of the introduced techniques in related fields, the size dependency of the power consumption of data-path macros is shown here. The data is again generated by gate-level synthesis and simulation based gate-level power estimation.

The complete evaluation was performed using the implemented tool described in chapter 6. Three different model generation approaches are compared: linear interpolation, linear regression and the nonlinear technique proposed here. The representative for linear regression is the approach of Coumeri and Thomas. As this technique is a subset of the algorithm presented here, it can be effectively emulated using the BEACH tool by deactivating the power transformation steps. Note that following the notion of black box modeling we apply the linear regression to the complete circuitry instead of its structural parts as originally proposed in [32]. For both regression approaches two types of models were built: one minimizing the mean square error (or, equivalently, the standard deviation) and one minimizing the mean relative error. The second case could be produced by setting a weight function in BEACH (cf. 6.2.2). The piecewise modeling by node insertion is orthogonal to linear and nonlinear regression and was hence applied to both types of modeling. The interpolation was implemented following the suggestion of Rovatti et al., which is shown in [119] to be time optimal (cf. section 3.2.6). The implementation of both the table and the interpolation was conducted using C.

7.1 Register Files

The section documents modeling of the cell as well as interconnect energy consumption of register files given as soft macros from the Synopsys DesignWare®.

Data Abstraction

The standard register file "DW_ram_r_w_s_dff" is a flip-flop based one read, one write port register file [138]. It has the following inputs: clock, reset, read address, write address, write enable and chip select. The only output is the read data. The read operation of this macro is asynchronous and not sensible to the chip select, in other words changing the read address always initiates a read action. Write is synchronous and dependent on the activation of both, the chip select and the write enable. From this information four working modes can be deduced:

1. *Idle clocking.* The register file is not accessed but clocked.
2. *Read access.* By changing the read address an asynchronous read is initiated. This is independent from the clock.
3. *Write access.* New write address and data is applied. The chip select and write enable are activated. The write is then started by the rising clock signal. The write address is in this case different from the read address so that the new content is not visible at the output.
4. *Write through access.* Identical to write access except that write and read address are identical meaning, that the value written becomes immediately visible at the output.

These four working modes can be simulated using suitable short sequences of input vectors (see Appendix C). The degrees of freedom for these simulations are:

- Instance parameters: number of stored words (*depth*) and *bitwidth*
- The address prior to the observed access $addr_{n-1}$ and during the observed access $addr_n$.
- The data input vector prior to the observed access $data_{n-1}$ and during the observed access $data_n$.

As mentioned in 5.1 data and address vectors must be abstracted into scalar variables for model building. Weight and Hamming distance are used for register file modeling as previously suggested. For the address inputs, the *bit-wise Hamming distance* is computed:

$$d_i := a_{n-1}[i] \; XOR \; a_n[i] \tag{7.1}$$

This finer grain measure can better handle spacial correlations introduced by the address decoders (cf. 2.3.1,A). Weight and (ordinary) Hamming distances are normalized by dividing them by the length of the parameter bit vectors. The complete set of candidate variables consists of address, data and instance related variables:

$$\begin{aligned}
V &= V_{address} \cup V_{data} \cup V_{instance} && (7.2)\\
V_{address} &= \{d_i(addr_{n-1}, addr_n) | 1 \leq i \leq bitwidth(address)\} \\
&\quad \cup \{weight(addr_{n-1}), weight(addr_n)\} && (7.3)\\
V_{data} &= \{d(data_{n-1}, data_n), \\
&\quad weight(data_{n-1}), weight(data_n)\} && (7.4)\\
V_{instance} &= \{depth, bitwidth\} && (7.5)
\end{aligned}$$

Experimental Design

From an analysis of circuit designs the required range of size parameters was determined as $4 \leq depth, bitwidth \leq 32$. The ranges of all other variables follow naturally:

$$d_i() \in \{0, 1\}, d(), weight() \in (0, 1)$$

The characterization set of instances was defined as a factorial design:

$$\begin{aligned}
Designs = \{(depth, bitwidth) | \quad & 4 \leq depth, bitwidth \leq 32, \\
& depth, bitwidth \text{ even}\}
\end{aligned} \tag{7.6}$$

Address and data parameters were obtained by randomly selecting values for $addr_{n-1}$, $addr_n$, $data_{n-1}$ and $data_n$ within their ranges. This does mean that the values of the abstracted functions (e.g. Hamming distances) are not equally distributed. They are instead distributed according to their probability during the application of the model, provided the data and address streams in the application are not significantly correlated. The latter assumption does not have to be true, but without knowledge of the application context it is the most sensible one.

Characterization

The characterization flow is depicted in figure 7.1. The major steps are explained in the following:

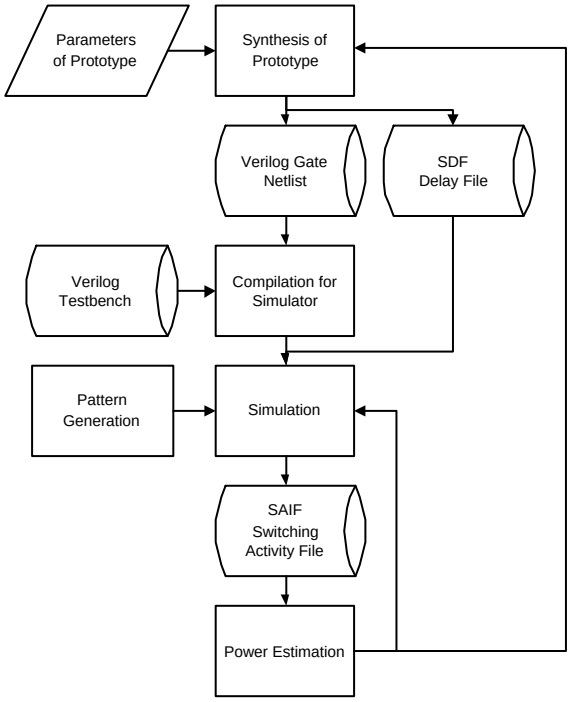


Figure 7.1: Characterization flow for register files.

1. *Synthesis of prototype.* A register file soft macro is instantiated according to the selected depth and bit width and synthesized into a LSI 'lcbg11p' technology gate level net list using the Synopsys DesignCompiler® [86, 137]. The net list is exported as Verilog file. Delay information is exported as SDF (Standard delay format) file [36].
2. *Compilation for simulation.* The register file net list is compiled for the simulation using Mentor Modelsim® [91]. A Verilog testbench is also compiled. The size parameters of the testbench are adapted using the "define" functionality so that the testbench source code remains unchanged.
3. *Generation of stimuli vectors.* $addr_{n-1}, addr_n, data_{n-1}$ and $data_n$ are generated randomly. Four stimuli files according to the four working modes are generated. Weights and Hamming distances are computed. A Perl script was written for this purpose [144].
4. *Simulation.* The design is simulated with the four different stimuli files. The simulator's programming interface is used to protocol all switching activity during these simulations and write this information out as SAIF (switching activity interchange format) files (see [136], chapter 5).
5. *Gate level power estimation.* Using the SAIF file and the gate level net list Synopsys PowerCompiler® computes a power estimate [136]. This estimate is stored in a table together with the values of the candidate variables.

As motivated above, steps 3 to 5 are executed repeatedly in order to obtain varying values for data and address parameters. The complete flow is repeated for each element of *Designs*. Simulating and estimating the 240 instances eqn. 7.6 in the four working modes with 100 vector pairs 100 vector pairs each took 610 hours of CPU time on SPARC ULTRA servers running Solaris.

Evaluation

Based on the obtained characterization data the model generation itself is evaluated. For this purpose the data is split into regression set and validation set (cf. 3.2.8). The following 1066 data sets (equivalent

to 20 hours of characterization) were used for regression:

$$depth, bitwidth = 4 \cdot i, 1 \leq i \leq 8 \tag{7.7}$$

$$awgt1, awgr2, ahd, dwgt1, dwgt2, dhd = \frac{1}{4} \cdot i, 0 \leq i \leq 4 \tag{7.8}$$

The remaining 22934 data set were reserved for the cross validation.

Interpolation table size is an issue with this application. With 8 input variables of 15 ($depth, bitwidth$), 11 ($awgt1, awgt2, ahd$) and 69 ($dwgt1, dwgt2, dhd$) levels a full interpolation table would have nearly 100 billion (10^{11}) entries. The main reason for this phenomenon is the grid requirement, i.e. the requirement that all combinations of levels must be present. In the register file application many combinations of variables cannot occur. The difference in the data weights can for example not exceed the hamming distance ($dwgt2 - dwgt1 \leq dhd$). While such constraining equations are easily derived, there is no straight-forward way of using this knowledge to reduce the interpolation table size.

Table 7.1: Cross validation errors for the "DW_ram_r_w_s_dff" register file model.

Type	RMS	R^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Write: Cell Power								
I	44.9	0.251	277.0	51.1	-50.8	126328	-	-
NA	0.8	1.000	5.8	0.8	-0.2	1134	1843	14
NAr	0.8	1.000	5.9	0.8	-0.3	832	366	9
LA	0.8	1.000	5.9	0.8	-0.3	44	49	14
NR	0.9	1.000	5.1	0.8	-0.1	785	418	9
NRr	0.9	1.000	5.6	0.8	-0.2	963	305	9
LR	1.0	1.000	5.7	0.8	-0.4	44	46	12
Write: Wire Power								
I	34.7	0.321	279.4	50.0	-49.6	684055	-	-
NA	2.2	0.999	120.5	5.0	-0.9	9197	504	10
NAr	3.0	1.037	71.4	4.9	0.9	2452	607	11
LA	3.5	0.996	37.5	5.2	-0.4	218	25	12
NR	6.6	0.988	32.8	5.5	-1.0	9854	4265	13
NRr	2.8	1.010	40.2	4.5	-1.0	1664	822	10
LR	4.8	0.993	35.3	7.0	-4.6	262	39	14
Write Though: Cell Power								
I	43.5	0.231	283.7	51.2	-50.8	687018	-	-
NA	1.7	0.999	8.3	1.4	-0.7	7927	438	9
NAr	1.6	1.014	9.5	1.5	-0.8	1883	571	10
LA	1.6	0.999	10.1	1.4	-0.6	349	40	15
NR	1.9	0.999	10.5	1.6	-0.5	9854	536	9
NRr	2.0	0.999	12.8	1.9	-1.4	2015	268	8
LR	2.0	0.999	13.5	1.7	-0.9	218	49	15
Write Through: Wire Power								
I	34.8	0.315	280.2	50.2	-49.7	684647	-	-
NA	2.2	0.999	100.7	4.9	-1.3	10160	1894	12
NAr	2.8	0.953	48.7	4.1	0.1	2540	416	10
LA	3.3	0.997	33.8	4.7	-0.2	218	15	10
NR	11.0	0.965	21.9	6.6	0.3	6525	178	7
NRr	3.4	0.999	51.3	4.7	-2.2	1927	668	11
LR	4.9	0.993	34.5	6.8	-4.4	218	22	11
Idle Clocking: Cell Power								
I	43.6	0.254	271.4	50.6	-50.4	686647	-	-
NA	0.0	1.000	0.0	0.0	0.0	6876	903	10
NAr	0.0	1.000	0.0	0.0	0.0	657	57	5
LA	0.0	1.000	0.0	0.0	0.0	131	6	4
NR	0.0	1.000	0.0	0.0	0.0	3898	167	7
NRr	0.0	1.000	0.0	0.0	0.0	482	62	5
LR	0.0	1.000	0.0	0.0	0.0	131	6	4
Idle Clocking: Wire Power								
I	33.9	0.350	305.8	51.4	-51.0	685460	-	-
NA	2.0	0.999	62.0	3.8	0.1	8846	1001	12

Table 7.1: (continued)

Type	RMS	R^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
NAr	3.1	0.997	59.3	4.7	0.4	2190	492	10
LA	3.1	0.997	41.7	4.8	-0.1	218	31	14
NR	6.9	0.986	62.9	5.7	-0.9	7489	502	10
NRr	6.2	0.883	37.4	6.9	-4.4	1445	134	7
LR	3.9	0.996	30.5	6.2	-3.3	174	23	11

For this evaluation the table size was reduce by normalizing the input variables and selecting as table entries the grid also used as regression set (eqn. 7.7,7.8). Through these measures the number of levels is reduced to 8 ($depth, bitwidth$), 11 ($awgt1, awgt2, ahd$) and 5 ($dwgt1, dwgt2, dhd$) and the total size of the table to $9 \cdot 10^6$ entries.

Table 7.1 shows the errors of the 22934 cross validation data sets and the modeling parameters obtained for the write, write through and clock working modes of the register file separated into cell and wire power. The table is organized as follows:

1. *Type*. This column contains the model type: I, NA, NAr, NR, NRr, LA or LR. Here 'N' stands for nonlinear, 'L' for linear, 'A' for minimization of absolute, 'R' for minimization of relative error and 'I' for interpolation. The suffix 'r' stands for the models generated using the exponent rounding scheme described in section 5.7.
2. *RMS*. The root means square error.
3. R^2. Coefficient of multiple determination.
4. *XARE*. Maximum absolute relative error.
5. *MARE*. Mean absolute relative error.
6. *MRE*. Mean relative error.
7. t_{eval}. Time for one evaluation in nanoseconds. Measured on a 933 Mhz Pentium 3 Mobility, 512 Mb SDRAM, running Debian Linux.
8. t_{gen}. Time for model generation in seconds. Measured on a 933 Mhz Pentium 3 Mobility, 512 Mb SDRAM, running Debian Linux. This time does not include characterization.
9. *Size*. The number of monomials ξ_i in the model.

In the interpolation case no values are presented for the columns t_{gen} and *Size* as these categories are not applicable.

Table 7.2: Cross validation errors for the read accesses of the "DW_ram_r_w_s_dff" register file model.

Type	RMS	R^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Read: Cell Power								
I	174.9	-	3525.5	414.5	353.6	267045	-	-
NA	15.9	0.974	356.5	18.4	0.8	2103	106000	23
NAr	15.9	0.974	454.3	18.3	0.1	1534	24527	16
LA	14.0	0.980	265.6	14.7	1.9	114	3452	38
NR	14.4	0.979	422.3	14.8	-0.2	2103	83027	50
NRr	17.1	0.970	313.9	17.4	-1.8	4205	88263	49
LR	15.7	0.975	217.8	14.8	-2.8	170	8945	50
Read: Wire Power								
I	254.1	-	4234.4	422.9	360.3	242045	-	-
NA	14.0	0.977	439.7	16.1	4.9	1193	12073	13
NAr	15.8	0.971	2081.2	24.1	-1.5	2103	9636	21
LA	13.5	0.979	319.6	16.5	10.7	114	2400	36
NR	14.3	0.976	529.5	15.2	4.5	1875	91874	50
NRr	12.7	0.981	229.0	13.7	7.0	1477	98367	26
LR	13.9	0.977	247.6	13.5	7.8	114	7374	50

As mentioned in section 2.4.5 asynchronous read accesses constitute an especially challenging modeling problem due to intense spacial correlation. In order to achieve the desired accuracy nevertheless the bit-wise Hamming distances were additionaly included among the candidate variables for the read cases. By adding these variables a bigger regression set became necessary. This was diagnosed by an extreme discrepancy between regression and cross validation errors using the old set. See table 7.3 for the comparison (note that during cross validation errors can become so high that r^2 is not any more well defined. This is indicated in this and the following tables by '-'). Consequently, to enlarge the regression set the restriction 7.8 on the regression subset was dropped. By this measure the regression set increased to 6400 values, equivalent to roughly 40 hours of characterization. The cross validation subset decreased to 17600 values. The results are shown in table 7.2.

Table 7.3: Comparison of regression and cross validation errors for the read accesses to the register file (NA) using the regression set described in eqn. 7.8.

Type	RMS	r^2	XARE	MARE	ARE
Regr.	2,1	0.997	284.0	9.1	-1.4
CV	216,7	-	11449.0	25.8	7.6

Results

Most prominent among the results is the bad performance of the interpolation (see also figure 7.2). This is due to the table density problems described above. With MAR errors around 50% (and even up to 423% in the read cases) this technique is not suitable in its presented form.

Both, the pure linear and nonlinear methods show very similar errors. With MAREs below 7% in all but the read cases both types of regression behave well. Remembering that the error to be minimized in the 'A' cases is the root mean square error and in the 'R' case the mean relative absolute error, the nonlinear regression has a better optimization performance than the linear one. The MAREs for read mode are higher, but stay below 20%. Here, the linear modeling has a slight accuracy advantage.

With the sole exception of the trivial cell power during clocking, the nonlinear models are more compact than the linear ones, i.e. they draw a comparable accuracy out of less model terms. This is due to the higher expressive capabilities of the nonlinear models. The tendency of the maximum relative errors of the nonlinear models to be higher than that of the linear ones is, maybe counterintuitively, a result of the same circumstance: as the targets of the optimization are the root mean square or the mean absolute relative error, it is worth while to reduce the mean error at the cost of an increased maximum relative error. The nonlinear method can, due to its increased flexibility, do this more effectively, than the linear one. This effect is especially pronounced when optmizing for absolute errors: Large relative errors for small values of the dependent variable often mean only small absolute errors. A minimization of the absolute error therefore will tend produce large relative errors for small values of the dependent variable (see figure 7.3).

Interestingly, for the multidimensional models shown here interpolation is by far the slowest model representation (see column t_{eval}). Nonlinear regression outperforms interpolation by a factor of at least 5. Linear regression models can again be up to a factor of 50 quicker. This is due to the use of the computationally more demanding power functions. The model generation times of the nonlinear models are by far higher than those of the linear ones (column t_{gen}). With a maximum of 71 minutes for write/clock and 30 hours for read accesses the time requirements are nevertheless acceptable.

As expected the rounded nonlinear models ('NAr' and 'NRr') range between the linear and the nonlinear ones in accuracy as well as performance in most cases. While the rounded nonlinear approach can in practice be a good compromise between performance and accuracy, it does not shed any additional light into this investigation. It will therefore not be pursued here any further.

7.2 Philips Embedded Memories

This section describes the modeling of three types of Philips embedded memories: a single-ported SRAM ('SRAM'), a low-power ROM ('LPROM') and a high-speed ROM ('HSROM').

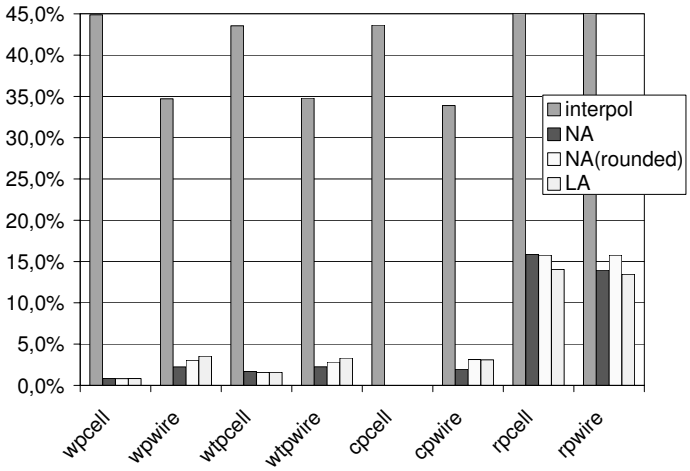


Figure 7.2: Absolute errors of the register file models. The interpolation error exceeds the scale for read accesses (175%/255%).

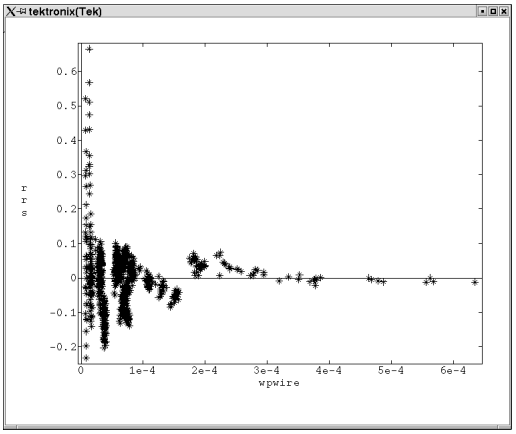


Figure 7.3: Relative residuals as a function of the dependent variable. Minimizing absolute errors can mean high relative errors for low values of the dependent variable.

Table 7.4: SRAM (126 samples), see section 7.1 for description of the columns.

Type	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Initial								
I	0.8	1.000	6.6	1.2	-0.7	8267	-	-
NA	1.6	1.000	9.4	2.0	-0.7	4514	6	7.25
LA	4.7	0.997	25.5	7.3	-1.4	74	1	7.0
NR	4.4	0.995	6.0	1.7	-0.1	3919	4	6.25
LR	6.1	0.994	9.5	2.9	-0.9	85	1	7.0
Best								
I	0.8	1.000	6.6	1.2	-0.7	8267	-	-
NA	0.7	1.000	3.4	0.9	-0.1	6762	1552	13.0
LA	1.0	1.000	4.8	1.1	-0.3	979	118	15.5
NR	1.5	1.000	2.2	0.8	0.0	6450	444	10.25
LR	0.9	1.000	1.4	0.4	0.0	291	3660	32.5

Data Abstraction

The Philips simulation models for characterization did not allow a free selection of address and data. As a consequence none of the models presented here does reflect the influence of these aspects. The remaining candidate variables were related to instance and environmental parameters (cf. section 2.3):

- x: the number of word lines.
- y: the number of words stored in parallel columns.
- z: the number of blocks or banks (only applicable for LPROM).
- b: the bit width.
- vdd: the supply voltage.

The response variable was the average current through the V_{DD} terminal per access. Unfortunately, the current obtained for the SRAM was averaged over read as well as write accesses so that read and write could not be modeled separately.

Experimental Design

The model generation was performed based on data already existing at Philips. Therefore it was not possible to influence the experimental design. The existing design did not show a high degree of regularity. This circumstance especially put interpolation to a test (see below). A major drawback of the existing design was that only little data was available for other supply voltages than the nominal 2.5V.

Characterization

The characterization was performed using PStar (Spice like) circuit level simulation. The simulated critical path models were obtained from the Philips in-house memory compiler. The compiler assembled the simulation models from net list descriptions of leaf cells extracted from the layout. The models were tested by Philips to be accurate within a few percent of fully extracted net lists. Each simulation of the ROMs consisting of 4 cycles took about 20 minutes of CPU time on HP PA-RISC 7200 CPUs. The SRAM simulations took longer by roughly a factor of 2.

Evaluation

Due to the lack of regularity of the experimental design no natural separation of the characterization data into regression and validation set existed. 20% of the total data was therefore selected randomly for the regression set. This procedure was repeated four times. Results presented are averaged over these.

Results

Table 7.4 shows the results for the static RAM modeled as a function of the parameters x, y and b. The characterization data set consisted of 126 samples, of which 24 were selected for the cross validation.

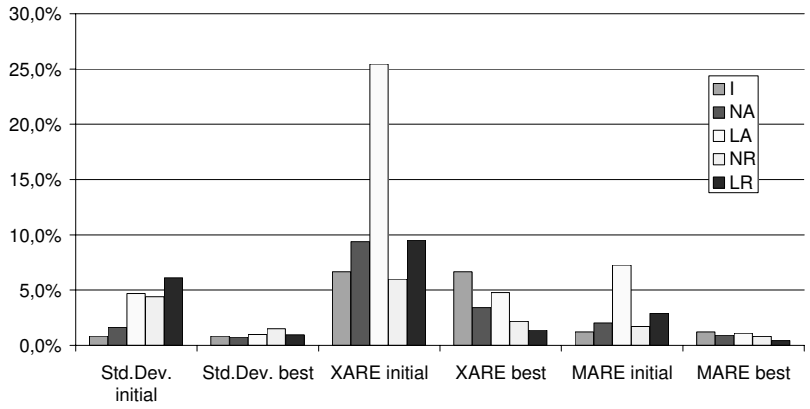


Figure 7.4: Absolute and relative errors of the SRAM model.

Similar to the register files, different cases (NA, LA, NR, LR) are compared using various measures (cf. section 7.1). Here the results are averaged over the four randomly selected cross validation sets. Furthermore the algorithm for automatic selection of cuts for piecewise modeling was applied (cf. section 5.5.1). Table 7.4 shows the models without cuts under 'Initial' and best model obtained under 'Best'. As best absolute models ('A') those with the lowest standard deviation were selected. The best relative ('R') models were those with the lowest MARE. The complete data for all models can be found in appendix D.1.

We can see that all models except the 'Initial/LA' perform very well (see figure 7.4). Worst case relative errors below 10% are acceptable even for gate level application. The generation takes only a couple of seconds for the initial models. Here the standard deviation of the 'LA' model is a factor of two higher than that of the 'NA' case. Roughly the same relation holds for the relative errors in the models optimized for relative accuracy. The accuracy gain by the introduction of piecewise modeling is more pronounced for the linear regression model. This is due to the fact that the iterative introduction of cuts terminates later in the linear than in the nonlinear case (cf. appendix D.1). A byproduct of this effect is, that the 'Best' linear models tend to consist of more monomials, than the nonlinear ones. Expressed the other way around, the nonlinear modeling is able to produce adequate models with less terms. The more complex computation of the power functions causes a factor of up to 65 in the evaluation times of linear and nonlinear models. The interpolation takes even longer than the nonlinear regression. It has slightly bigger errors than the regression models of the 'Best' category.

Table 7.5: High speed ROM (127 samples) .

Type	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Initial								
I	18.2	0.949	43.1	4.4	1.4	14296	-	-
NA	1.5	1.000	25.8	3.7	0.7	6417	55	7.25
LA	2.4	0.999	29.0	5.3	-1.6	167	2	8.0
NR	2.1	0.999	6.5	1.9	0.4	5760	11	6.5
LR	3.6	0.998	6.1	2.4	0.2	198	2	9.75
Best								
I	18.2	0.949	43.1	4.4	1.4	14296	-	-
NA	1.3	1.000	19.9	3.3	0.4	8781	441	9.75
LA	1.5	1.000	11.9	2.2	-0.2	2021	831	22.0
NR	2.3	0.999	4.9	1.5	0.1	6427	68	6.75
LR	1.6	1.000	4.4	1.2	0.2	2281	1800	24.5

The modeling data for the high speed ROM is displayed in table 7.5 (see appendix D.2 for complete data). In addition to the variables for the SRAM the 127 original data samples also cover different supply

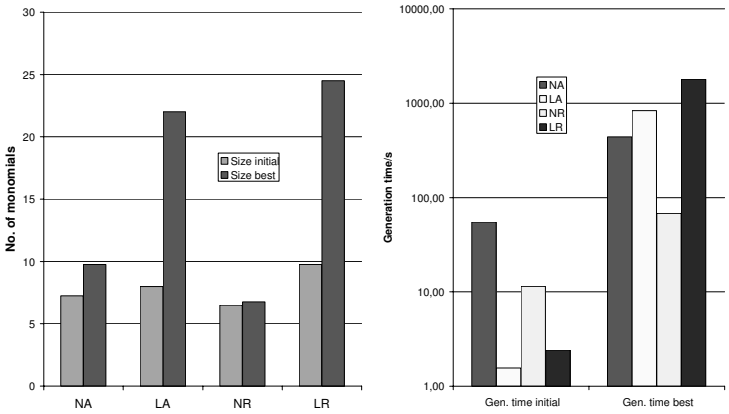


Figure 7.5: Size of models (left) and generation time (right) of the high-speed ROMs.

voltages. In order not to put the linear models at a disadvantage, the squared supply voltage was used as variable to modeling. The results are quite similar to the SRAM. The interpolation has the highest errors of all. With XAREs below 30 % and MAREs below 6 % all regression models are acceptable. The linear approach outperforms the nonlinear one only for the best relative case, but this at the price of a more than three times bigger model (see figure 7.5). Apart from this case the nonlinear technique delivers more compact as well as accurate models. The model generation time stays below 1800 seconds in all cases, the linear piecewise models taking longest.

Table 7.6 documents the results for the low power ROM (complete data in appendix D.3). This memory architecture has a more complicated structure, that is reflected by an additional parameter z (number of blocks). The advantage of the nonlinear models is even more pronounced in this more complicated situation (see figure 7.6). Among the initial models only the nonlinear ones are acceptable. As usual the improvement of linear models through the piecewise modeling is bigger, but the nonlinear ones stay superior.

Table 7.6: low power ROM including voltages (242 samples).

Type	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Initial								
I	14.5	0.983	108.6	12.9	0.4	17136	-	-
NA	2.7	0.999	29.8	5.0	0.7	12321	297	12.5
LA	15.8	0.980	72.0	21.1	-13.7	182	5	9.25
NR	5.5	0.997	28.7	5.3	-1.0	10182	165	10.5
LR	18.1	0.973	46.5	12.6	-3.3	220	8	11.75
Best								
I	14.5	0.983	108.6	12.9	0.4	17136	-	-
NA	2.4	1.000	27.6	4.4	1.2	16264	5166	17.75
LA	5.8	0.997	37.8	7.9	1.0	1759	421	19.5
NR	2.8	0.999	19.2	3.2	-0.6	14838	1160	13.5
LR	8.2	0.994	21.4	3.7	0.0	1642	202	19.75

7.3 LSI Embedded Memories

While the preceding models were based on low level power estimates the model in this section was generated from the estimate directly produced by a memory generator as a consumer view. The memory under investigation was a single ported SRAM in the LSI 'lcbg11p' technology called 'm11_111ha'.

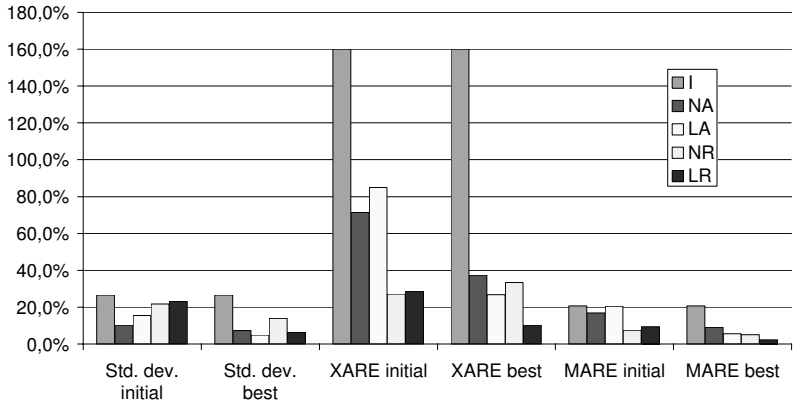


Figure 7.6: Absolute and relative errors of the low-power ROM models.

Data Abstraction

The memory generator delivered only data and address independent power estimates. The candidate variables were therefore restricted to the instance parameters $V = V_{instance} = \{x, b\}$.

Experimental Design

The parameter range for this memory is given by the generator as: $256 \leq x \leq 8192, 4 \leq b \leq 80$. For the characterization a factorial design was again chosen:

$$Design = \{(x, b)| \quad x = 64 \cdot i, 4 \leq i \leq 128 \\ b = 4 \cdot j, 1 \leq j \leq 20\} \tag{7.9}$$

Characterization

The memories were characterized by calling the memory compiler for every instance combination and redirecting its output into a table.

Evaluation

Similar to the evaluation of the Philips memories 24 % of the total of 1500 characterized instances were randomly chosen for the cross validation set. This process was repeated 4 times.

Table 7.7: Cross validation errors for the LSI m11_111ha embedded SRAM.

Type	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Initial Model								
I	0.1	1.000	1.0	0.0	0.0	19629	-	-
NA	1.9	0.998	17.1	2.0	0.2	1352	22	3
LA	2.5	0.997	15.8	2.6	-0.6	70	22	4
NR	2.2	0.998	7.3	1.7	0.0	1366	22	3
LR	3.0	0.996	7.8	2.1	-0.1	49	23	4
Best Model								
I	0.1	1.000	1.0	0.0	0.0	19629	-	-
NA	1.2	0.999	14.1	1.4	0.1	2367	317	8.25
LA	0.1	1.000	2.9	0.1	0.0	4265	9707	41.25
NR	1.3	0.999	6.2	1.0	0.0	4160	877	11.75
LR	0.1	1.000	2.3	0.1	0.0	4202	27019	41.75

Results

In this example (cf. table 7.7) the interpolation approach performed exceptionally well (XARE = 1.0 %, MARE = 0.0 %). But the other approaches did also show very good results (max XARE = 17.1 %, max

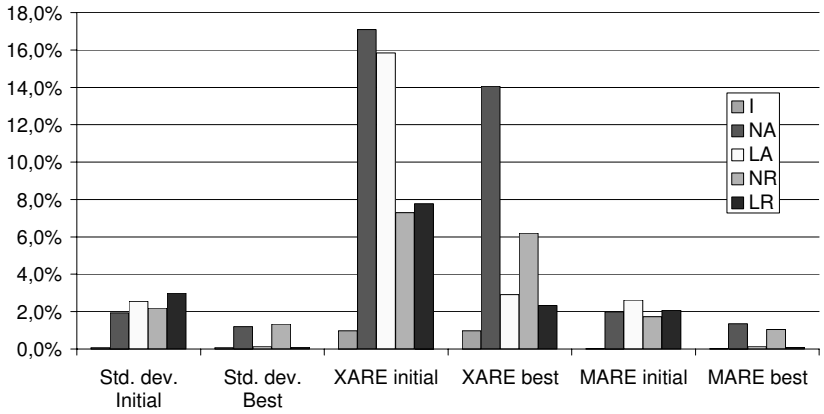


Figure 7.7: Absolute and relative errors of the m11_111ha SRAM models.

MARE = 2.6 %) with only 3-4 terms (see figure 7.7). This suggests that the underlying model of the memory generator is a very simple one. The piecewise modeling for linear regression terminates only after 6-9 cuts, resulting in very accurate, but also very large models ($Size > 40$). This is also reflected in long generation times.

7.4 DesignWare data path components

In the previous sections several memory modeling cases were exemplified. This section shows the application of the proposed modeling methodology in a different field of power modeling: the generation of power complexity functions for data path macros [124]. It can be seen as a fingerpost to a broader application domain of the developed methodology and tool. Three different components will be presented: a finite impulse response (FIR) filter, a sine module and a wallace tree multiplier [106].

Power Complexity

Like memories combinational data path components have a power consumption depending on both, instance (size) parameters and processed data. To reduce the modeling complexity, several approaches suggest to decompose the models into two sub-models: a data dependent 'activity' sub-model δ and a size dependent 'complexity' or 'capacitance' sub-model γ [83, 16, 70]. As motivated by applying equation 2.3 globally, i.e.

$$P_{avg} = 1/2 \cdot \alpha_{avg} \cdot C_{avg} \cdot V_{DD} \cdot V_{swing} \cdot f_{clk} \tag{7.10}$$

both sub-models are then assumed to have a multiplicative relationship

$$P(data, size) = \delta(data) \cdot \gamma(size) \tag{7.11}$$

While concrete ideas exist on how to generate δ, finding γ is a problem not solved for the general case. Bogliolo [16] states the relation between power consumption and size cannot be described by a general analytical formula, since it is dependent on the functionality of the macro. In this section the introduced black box modeling technique is advocated as a means to solve this problem. The approach will be evaluated using the enhanced Hamming distance model [70].

Table 7.8: Cross validation errors for the power complexity of the FIR Filter component. The new column 'M' contains the sequence number of the model (i.e. $M - 1$ is the number of nodes).

Type	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
I	1	27.4	0.903	45.1	5.2	-4.9	1678	-	-
NA	1	42.4	0.768	348.1	114.5	36.7	459	0	2
	2	0.9	1.000	7.3	1.8	0.2	482	1	3
LA	1	41.8	0.775	369.9	125.4	35.5	43	0	2
	2	0.9	1.000	1.1	0.5	-0.1	44	0	3
	3	1.6	1.000	2.6	0.5	0.1	81	1	5
NR	1	75.2	0.272	82.0	35.4	-30.0	457	0	2
	2	3.8	0.998	36.0	7.3	2.6	773	1	4
	3	4.1	0.998	42.6	10.6	1.9	63	1	3
LR	1	115.7	-0.723	87.1	52.3	-39.8	43	0	2
	2	0.9	1.000	1.4	0.7	-0.1	43	0	3
	3	0.9	1.000	1.2	0.4	0.0	80	1	5

For building a complexity model data acquisition has to be performed first. Let n instances be chosen to adequately cover the targeted instance range. Let $size_i$ denote the instance parameter values of instance i. For each instance a data stream ds_i is generated respecting the following property:

$$\forall 1 \leq ds_i, ds_j \leq n : \delta(ds_i) = \delta(ds_j) \tag{7.12}$$

Data for data path macros is generated analog to the data for the register file macros: selected instances are synthesized to gate level and simulated with the appropriate data stream. A gate level power estimation is

then performed based on the collected switching activity. Let $\vec{y}$ be the vector of low level power estimates for the n instances. Applying equation 7.12 we can then deduce γ from the following relationship:

$$\vec{y} = P(\vec{d}, \vec{size}) = const \cdot \gamma(\vec{size}) \tag{7.13}$$

For more detail about the generation of power complexity functions see [124].

Table 7.9: Cross validation errors for the power complexity of the sine component.

Type	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Intial Model								
I	25.9	0.878	36.6	8.7	-4.2	4291	-	-
NA	32.5	0.806	43.0	14.7	-2.6	1982	0	3
LA	37.7	0.740	135.2	23.8	-0.8	109	0	3
NR	46.5	0.603	56.4	16.4	-9.4	1982	0	3
LR	15.8	0.451	57.2	26.2	-18.6	91	0	2
Best Model								
I	25.9	0.878	36.6	8.7	-4.2	4291	-	-
NA	12.1	0.973	44.4	9.9	-2.3	1927	16	7
LA	23.2	0.901	157.7	16.3	1.5	91	0	3
NR	46.5	0.603	56.4	16.4	-9.4	1982	0	3
LR	15.8	0.451	57.2	26.2	-18.6	91	0	2

Table 7.10: Cross validation errors for the power complexity of the sine component II.

Type	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Intial Model								
I	19.5	0.927	74.5	10.4	2.7	4473	-	-
NA	29.4	0.844	68.9	19.9	4.6	1455	0	3
LA	35.5	0.773	169.0	30.7	11.6	55	1	3
NR	33.4	0.799	43.2	14.3	-1.5	2000	1	4
LR	45.5	0.627	74.4	26.2	-5.5	55	0	2
Best Model								
I	19.5	0.927	74.5	10.4	2.7	4473	-	-
NA	9.3	0.984	31.3	6.3	1.1	1527	24	7
LA	10.3	0.981	45.8	6.9	1.9	836	58	12
NR	11.6	0.976	53.1	8.2	-0.1	1891	67	8
LR	18.5	0.938	29.8	8.9	0.1	182	3	6

Data Abstraction

As the address and data dependencies are separated into δ, only size parameters remain to be modeled. These are the internal bit width of the FIR filter, the input and output bit widths of the sine module and the two input bit widths of the multiplier.

Experimental Design and Evaluation

For the FIR a 100 tap component was chosen as a complex example. The coefficients where chosen randomly. The internal bit width was varied between 4 and 40 bits in steps of two bits (19 instances). From these instances each bit width divisible by 4 was selected for regression (10 instances).

$$Design_{FIR} = \{bw \mid bw = 2 \cdot i, 2 \leq i \leq 20\} \tag{7.14}$$

$$RegrSet_{FIR} = \{bw | bw = 4 \cdot i, 2 \leq i \leq 20\} \tag{7.15}$$

Of the sine module 117 instances were characterized in the factorial design described by:

$$Design_{Sine} = \{\ (bw_{in}, bw_{out}) | bw_{in} = 2 \cdot i, 4 \leq i \leq 16, \\ bw_{out} = 2 \cdot j, 4 \leq j \leq 12\ \} \tag{7.16}$$

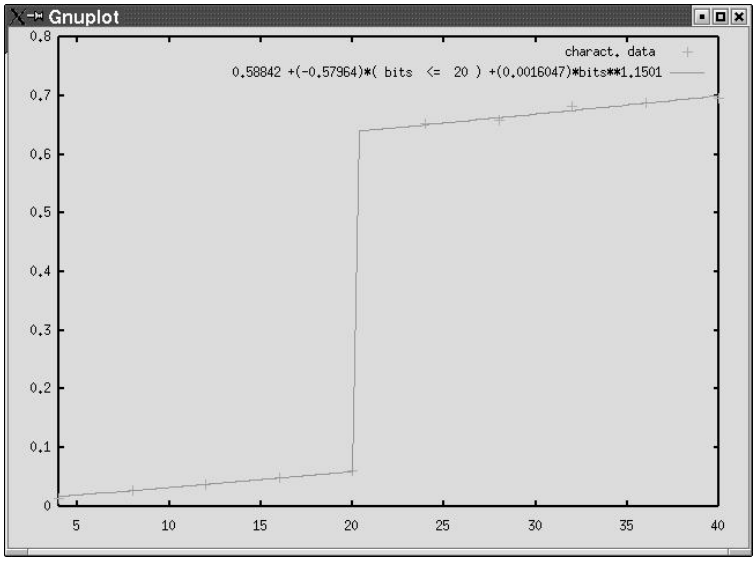


Figure 7.8: Regression data and NA model (M = 2) for the FIR component (BEACH screenshot).

Two Regression sets were defined (35 and 58 samples):

$$RegrSet_{Sine_1} = \{\ (bw_{in}, bw_{out}) | bw_{in} = 4 \cdot i, 2 \leq i \leq, 8 \\ bw_{out} = 4 \cdot j, 2 \leq j \leq 11\ \} \quad (7.17)$$

$$RegrSet_{Sine_2} = \{\ (bw_{in}, bw_{out}) | bw_{in} = 2 \cdot i, 4 \leq i \leq 16, \\ bw_{out} = 2 \cdot j, 4 \leq j \leq 22, \\ bw_{in} = 4 \cdot k\ \dot{\vee}\ bw_{out} = 4 \cdot k, k \in N\ \} \quad (7.18)$$

The wallace tree multiplier was characterized for all combinations of bitwidths between 4 and 32, respectively 4 and 26:

$$Design_{Mult} = \{(bw_a, bw_b) | 4 \leq bw_a \leq 32, 4 \leq bw_b \leq 26\} \quad (7.19)$$

Of these 667 instances all those with two even bit widhts were selected for regression (180 instances).

$$RegrSet_{Mult} = \{\ (bw_a, bw_b) | bw_a = 2 \cdot i, 2 \leq i \leq 16, \\ bw_b = 2 \cdot j, 2 \leq j \leq 13\ \} \quad (7.20)$$

Characterization

The characterization was performed by synthesis and gate level simulation similarly to the register files (cf. section 7.1). In this case, however, only the input stream ds_i had to be simulated for every instance i.

Results

Table 7.8 shows the errors of all models for the FIR unit. Note that, as the complete set of models are given instead of only the initial and best, a new column 'M' has been added to the table. 'M' refers to the number of the model in a sequence of cuts (during automatic piecewise modeling). A model with $M = i$ consequently contains $i - 1$ automatically generated cuts (see section 5.5.1). Remarkable among the results is that the initial models of all categories exhibit significant errors. A look at the data shows the reason for this phenomenon (cf. figure 7.8): the power of the FIR unit explodes by roughly one order

Table 7.11: Cross validation errors for the power complexity of the wallace tree multiplier component.

Type	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Intial Model								
I	17.0	0.933	65.7	12.3	0.6	3621	-	-
NA	14.5	0.951	114.6	12.5	-0.8	1235	0	3
LA	14.6	0.950	71.2	12.2	2.6	62	1	3
NR	15.6	0.944	50.0	11.2	-2.9	1296	1	3
LR	15.5	0.944	48.6	11.2	-2.9	82	1	4
Best Model								
I	17.0	0.933	65.7	12.3	0.6	3621	-	-
NA	14.5	0.951	114.6	12.5	-0.8	1235	0	3
LA	14.6	0.950	71.2	12.2	2.6	62	1	3
NR	15.7	0.943	47.5	10.8	-3.5	2428	7	5
LR	15.5	0.944	48.6	11.2	-2.9	82	1	4

of magnitude at 20 bits. The automatic piecewise modeling identifies this discontinuity and splits the model into two segments as can be seen in the figure. Through this segmentation the errors are kept reasonable. It can also be seen that the nonlinear regression is less effective in this case. This has two reasons: the underlying relationship is nearly perfectly linear and the regression set is with 10 elements very small. The cross validation results for sine component are shown in table 7.9 and 7.10. The nonlinear method again shows a slight advantage over both linear types of models. It can be observed that for the first regression set (table 7.9) segmentation is not able to significantly improve the model accuracy. The second regression set is much better in this respect. The reason for this phenomenon can be found in the nature of the piecewise modeling applied: as was explained in section 5.5.1 introducing nodes reduces the region of definition of the model by the range between the levels adjacent to the node. By the choice of sample points in the second regression set the distance between levels is halved with respect to the first set. Furthermore there is no level in the remaining cross validation set, that does not occur in at least one sample point. This change allows a much better convergence of the node insertion. In table 7.11 the errors for the wallace tree multiplier are displayed. In this example all three modeling approaches have a very similar performance. Node insertion although not significantly reducing error measures, leads to intuitively more adequate models (see figure 7.10).

7.5 Discussion and Summary

In the preceding sections the newly proposed modeling methodology was evaluated on a set of practical examples. Model accuracy, speed and size as well as the model generation performance were analyzed in

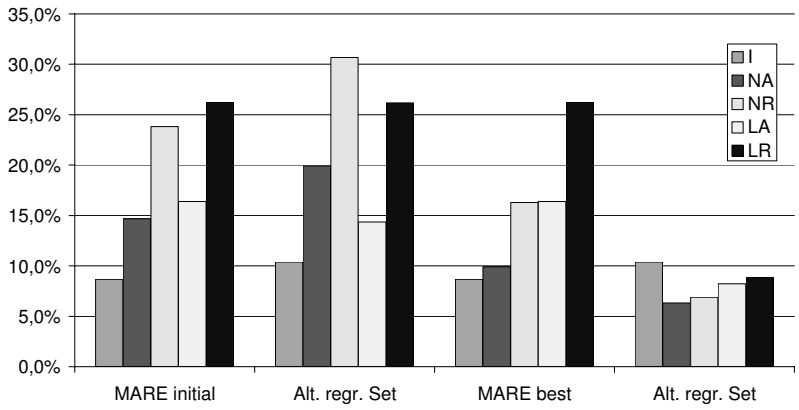
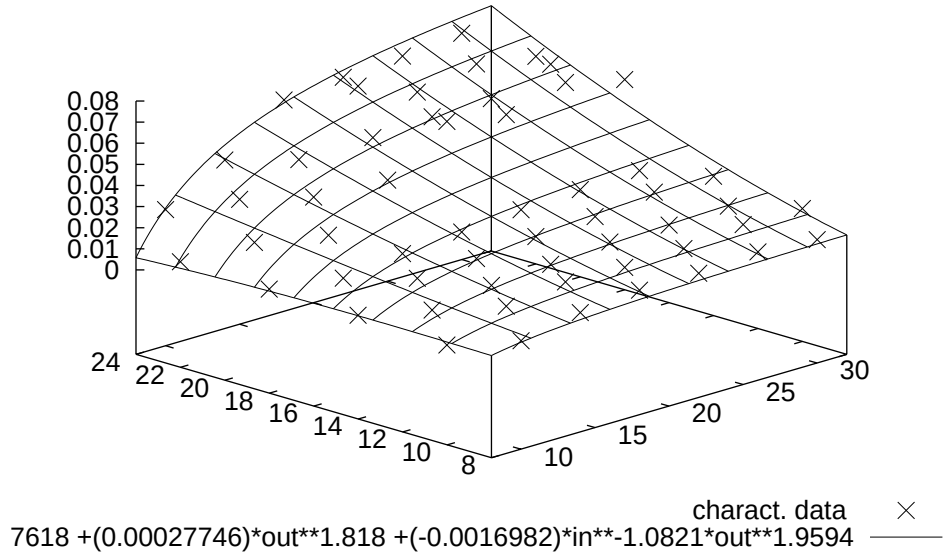


Figure 7.9: Initial and best relative errors for $RegrSet_{Sine_1}$ and alternative regression set $RegrSet_{Sine_2}$.



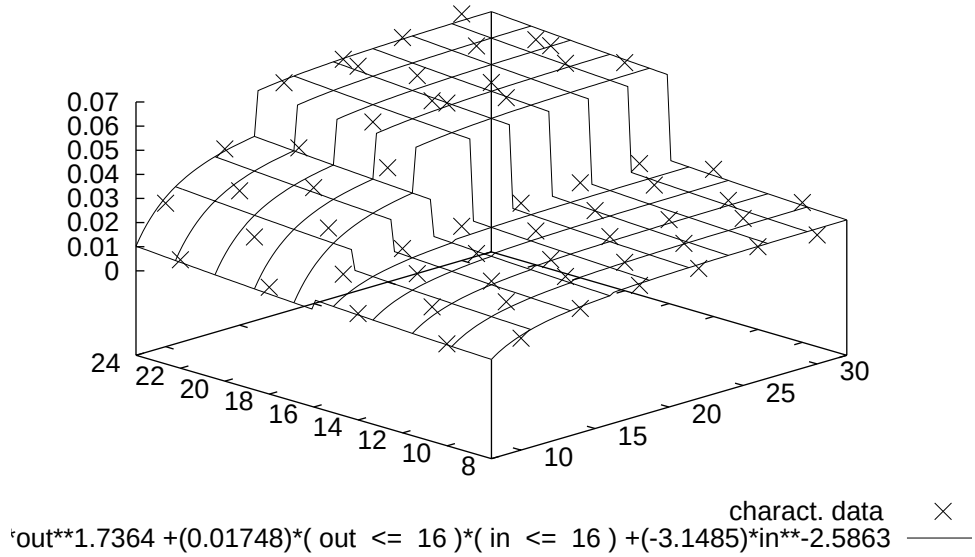


Figure 7.10: Nonlinear absolute models for the wallace tree multiplier: initial (top) and final (bottom).

reference to the requirements defined in section 3.3. The results regarding the most important metric - accuracy - are again summarized in table 7.12.

Both relative and absolute error values show, that the new technique achieves excellent accuracy. With RMS errors below 3% and MAR errors below 7.5% the models for memories are in the same accuracy range as circuit level models. Only the read accesses for registers have higher errors (15.9%/15.2%). All errors stay clearly within the accuracy requirements defined for RT and behavioral level application in section 3.3.

Comparing the accuracy of the nonlinear regression with linear regression and interpolation, the following observations can be made: While in simple cases interpolation slightly outperforms the nonlinear technique (see Table 7.7), it quickly looses ground with increasing complexity (e.g. MAR errors above 50% in table 7.1). The nonlinear technique proposed in this work and linear regression are often close in accuracy. In some cases however the nonlinear approach clearly outperforms the linear one. For example the absolute errors of the linear models are nearly a factor of 3 higher for the Philips SRAM and nearly a factor of 6 for the low power ROM.

The errors for piecewise modeling by node insertion (see section 5.5.1) are listed in table 7.13. They show clearly, that piecewise modeling is able to reduce the remaining error significantly: as an extreme case the absolute error for LSI SRAM (linear model) is reduced by a factor of 20. In general the linear models benefit more from the node insertion than the nonlinear ones.

The errors for the data path macros are summarized in table 7.14. Here, as described above, discontinuities can have a drastic impact. Using node insertion errors below 15% can however be reached for all components.

In addition to accuracy nonlinear models have one further advantage: compactness. Figure 7.11 shows the average model size for the different regression based memory models. It can be seen that the nonlinear models are generally more compact than the linear ones. The NA model for the LSI RAM is more than a factor 4 smaller than the LA one. It is intuitive that accuracy and compactness come at the price of an increased model generation time. Consider therefore the average modeling times visualized in figure 7.12. It can be observed that contrary to expectation the nonlinear technique is not generally slower, than the linear one, when node insertion is performed. While it is indeed slower for register files and LP ROM it outperforms the linear approach in the HS ROM and LSI RAM examples. This phenomenon can be explained by the faster convergence of the node insertion procedure and the smaller models produced. These effects can outweigh the additional effort of the exponent adaption. The maximum average CPU time is 7.5 hours. Taking together the 8 individual sub-models the complete nonlinear register file model does require 50 hours. While these times might seem long at first glance three things should be kept in mind when considering the connected costs: i) in contrast to analytical modeling virtually no human intervention is required. ii) no other tool licenses are blocked during the model generation. iii) the modeling is only performed once for each memory.

Figure 7.13 finally contains the average evaluation times. The table size problem of the interpolation inflates the lookup times for the register files beyond acceptable limits. But also for the other cases the regression models outperform interpolation. The nonlinear models clearly loose here due to the complicated computation of real valued power function.

Table 7.12: Optimum absolute (RMS) and relative (MAR) errors of all three compared methods.

	Nonlinear		Linear		Interpolation	
Name	RMS	MARE	RMS	MARE	RMS	MARE
Regfile	2.2	6.6	3.5	7.0	43.6	51.4
Regfile(read)	15.9	15.2	14.0	14.8	254.1	422.9
SRAM	1.6	1.7	4.7	2.9	0.8	1.2
HSROM	1.5	1.9	2.4	2.4	18.2	4.4
LPROM	2.7	5.3	15.8	12.6	14.5	12.9
LSI SRAM	1.9	1.7	2.5	2.1	0.1	0.0

Table 7.13: Optimum absolute (RMS) and relative (MAR) errors after node insertion.

	Nonlinear		Linear	
Name	RMS	MARE	RMS	MARE
SRAM	0.7	0.8	1.0	0.4
HSROM	1.3	1.5	1.5	1.2
LPROM	2.4	3.2	5.8	3.7
LSI SRAM	1.2	1.0	0.1	0.1

Table 7.14: Optimum absolute (RMS) and relative (MAR) errors before and after node insertion.

	Nonlinear		Linear		Interpolation	
Name	RMS	MARE	RMS	MARE	RMS	MARE
FIR filter	42.4	35.4	41.8	52.3	27.4	5.2
Sine	29.4	14.3	35.5	26.2	19.5	10.4
Multiplier	14.5	11.2	14.6	11.2	17.0	12.3
FIR filter	0.9	7.3	0.9	0.4		
Sine	9.3	8.2	10.3	8.9		
Multiplier	14.5	10.8	14.6	11.2		

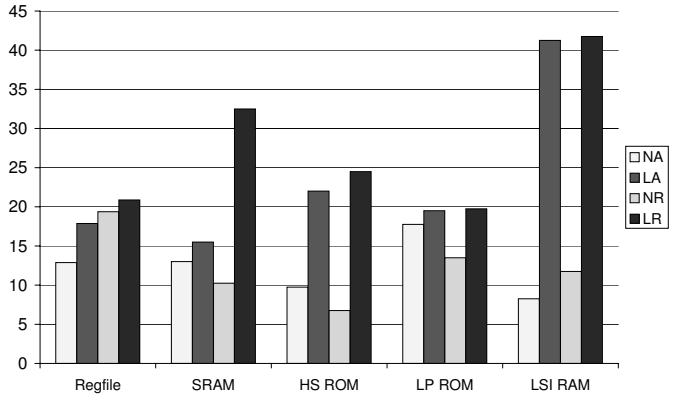


Figure 7.11: Average size (in number of monomials) for the different memory models (using node insertion).

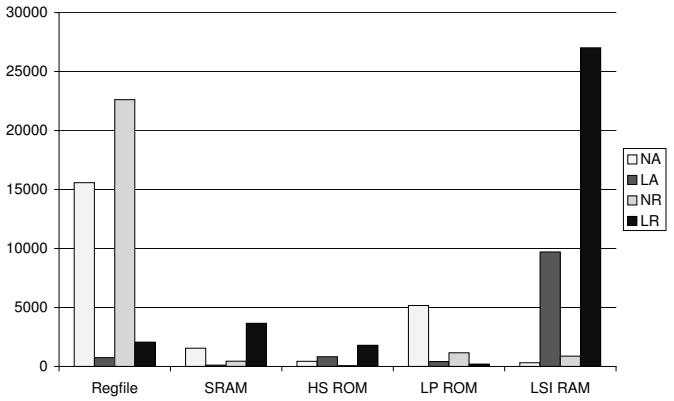


Figure 7.12: Average generation time in seconds for the different memory models (using node insertion).

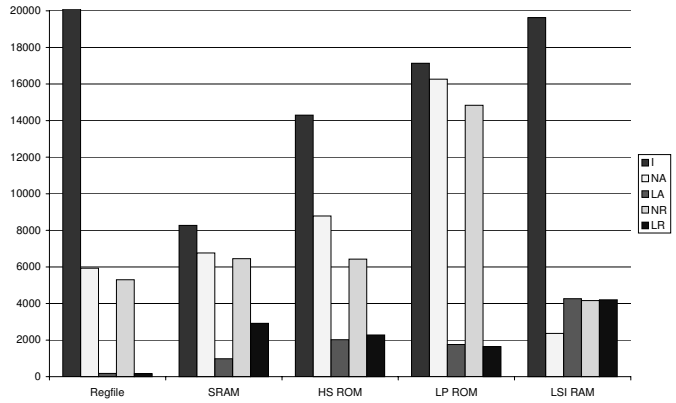


Figure 7.13: Average evaluation time in nanoseconds for the different memory models. The interpolation time of the register files exceeds the scale with 508 ms.

Summary

In summary the evaluation has shown that nonlinear modeling technique and the node insertion technique are able to significantly increase the accuracy and compactness compared to existing memory modeling methods. For memories as well as for the presented data path components the errors stay well below the defined limits.

The increased capabilities come at the price of a mathematically more complex model. This causes an increase in model evaluation time. A general increase in model generation time can however not be observed.

Fundamental limitations lie in the nature of the technique: The presented signomial models are among the most general functions for which effective optimization strategies exist. Still the expressiveness of signomials is limited. Not all kinds of relationships can be accurately approximated. One of the most simple examples of relationships that cannot be expressed is $y = (x_i + x_j)^\alpha$.

A second principle limitation is the requirement of observation data. All empirical modeling methods share this requirement of observations as their sole basis for model inference. In principle, the bigger the modeling freedom, the more data is required. The results above show that the nonlinear technique presented here works stable with the same amounts of data as the linear one. Nevertheless the more detailed a relationship must be modeled, the more characterization data is required. The generation effort of such data eventually is part of the whole modeling effort.

Unfortunately the available technology information for this evaluation was strictly limited. Therefore it was not possible to exemplify the modeling of DRAMs, flash RAMs or caches. DRAMs and flash RAMs are in their principle structures quite similar to the memories described above. It is therefore sensible to assume that the presented technique can also be successfully applied on these memories. In section 2.4.4 it was shown that the core of cache memories consists of two DRAM memory arrays. Analytical cache models concentrate on these memories (see section 4.1), which should also be easily modeled using the technique described here. It remains to be seen in how far surrounding logic, like tag comparison, poses an additional problem to the modeling.

8 Summary and Conclusion

Summary

In this thesis the issue of power modeling for semiconductor memories was elaborated. The relevance of this issue was documented and the physical effects leading to and the complications arising from power consumption were briefly described. To outline the modeling problem, an overview over the basic memory structures as well as statistical modeling techniques was given. A catalogue of requirements was generated on the background of high-level estimation and optimization applications. Existing approaches to memory power modeling were analyzed and discussed with respect to these requirements. A new modeling methodology was then proposed to remedy the flaws of existing approaches. Core of this methodology is an iterative algorithm that combines variable selection, variable transformation and model fitting for signomial functions as well as an automated node insertion procedure for piecewise modeling. For a wide applicability emphasis was laid on keeping the procedure as automatic as possible and making the interactive steps simple. It was shown that the family of signomial functions is – although nonlinear – well suited for optimization tasks.

Subsequent to the introduction of the new methodology its implementation in the tool ORINOCO BEACH and the integration of existing models in the power estimator DALE were documented. To validate the presented approach, several applications were exemplified: the modeling of three embedded memories of Philips based on critical-path model simulation, a family of LSI embedded memories based on memory compiler estimates and register files of Synopsys DesignWare® based on synthesis and gate-level simulation. In all cases results of the new methodology were compared to linear regression modeling and linear interpolation.

Conclusion

This thesis contributes the following aspects:

1. *A new family of analytical models (signomial models) for memory modeling.* These models have not been suggested for memory modeling before. They are among the most general families of mathematical functions for which effective optimization techniques exist (see section 5.6). More specifically, the nonlinear relationships occurring in memory designs can be effectively described using these models. The presented signomial models reduce the error up to a factor of six compared to linear models (see section 7.5). The absolute errors as well as the mean absolute relative errors are below 16% in all cases. The models are therefore well suited for application at RT level and higher abstraction levels (see requirements in section 3.3). They are generally more compact than linear models.

2. *A set of algorithms for the generation of signomial models.* The iterative algorithm presented in section 5.3.4 is an efficient search procedure combining variable selection and fitting. Its fast convergence allows model generation times that are comparable to that of linear models (see figure 7.12). Maximum average generation time are 6.5 hours. The node insertion algorithm for piecewise modeling (section 5.5.1) extends the model generation process and allows a reduction of model errors up to a factor of 20 (see section 7.5). In combination the techniques are well suited even for the power complexity modeling of data path components: Absolute and mean absolute relative errors are below 15%. The rounding procedure described in section 5.7 is a further extension that allows intermediate solutions between linear and nonlinear models in terms of accuracy and evaluation time (see section 7.1).

3. *A tool implementation of the generation algorithms.* The software tool BEACH described in section 6.2 implements the model generation as well as support for data acquisition and model analysis. It is currently marketed as one of three tools of the ORINOCO tool suite (see section 6.1). The evaluation in this thesis was conducted entirely using BEACH.

4. *A modeling methodology.* As motivated in chapter 3 model building is a complex iterative process. Chapter 5 describes a modeling methodology centered around the algorithms mentioned above. It deals with the aspects of data abstraction, data acquisition, variable selection and fitting, validation and measures in case of lacking accuracy. Emphasize was laid upon the applicability of this method by (trained) non-experts. The goal of this methodology is to reduce the overall modeling costs (see section 3.3). The methodology is exemplified in the evaluation in chapter 7.

5. *A concept for the integration of models into an estimation tool.* As described in chapter 3 the application of models is an integral part of the model building process. The performance of a power estimation or optimization application not only depends on the properties of the models, but also on their effective integration into the application. For these reasons a concept for the integration and application of the generated models within a power estimation tool was proposed (see section 6.3). The most important features of this concept are performance and flexibility. The proposed concept has been implemented and is now part of the behavioral power estimator ORINOCO DALE.

In summary the presented models, tool and methodology clearly meet the requirements defined in section 3.3. Therefore they serve as enabling technology for high-level memory power estimation and optimization.

Outlook

Due to cost and time-to-market requirements intellectual property components will continue to gain an increasing market share. At the same time design will be conducted at ever higher levels of abstraction to keep the pace of the technology development. Making available models for properties of IP at the design level will for this reason be an increasingly important as well as complicated and hence expensive task. The contribution of this thesis is to provide high-level power modeling of embedded memories at acceptable cost. As exemplified in section 7.4 the same methodology can also be applied to other kinds of IP modeling. The presented techniques could therefore serve as basis to approach the more general problem of IP property modeling.

A ROM Instance Model

This chapter describes the ROM model developed previously by the author [121, 123].

A.1 Introduction

Let $MEM(a)$ denote the content of address a, $w_n = MEM(a(n))$ and $w_{n-1} = MEM(a(n-1))$ the data word read in cycle n and $n-1$ and $w'_n = MEM(a(n)\ XOR\ 01)$. Let OE be the state of the output enable (active low). The variables S_x respectively O_x are slope and offset of linear approximations. They are model coefficients that must be fitted by a characterization procedure. The function # specifies the number respective bit events in two consecutive vectors. $\#_{01}(i,j)$ for example denotes the number of rising bits between vector i and j. The norm || denotes the *weight* of the respective data word, i.e. the number of ones contained.

The total power consumption consists of: the power consumed during the memory cycle, during address changes and during output (de-)activation:

$$P_{total} = P_{cycle} + P_{address} + P_{output} \tag{A.1}$$

A.2 Cycle Related Power

Cycle-synchronous power can further be sub-divided into the sensing and the z bus related parts and the rest.

$$P_{cycle}(w_n, w'_n) = P_{sensing}(w_n, w'_n) + P_{driving}(w_{n-1}, w_n, OE) + P_{static} \tag{A.2}$$

The sensing related contribution can again be subdivided into bit-line pre-charge, the "corespondent address", and the sense amplifiers

$$P_{sensing}(w_n, w'_n) = P_{bpre}(w_n) + P_{caddr}(w'_n) + P_{senseamps}(w_n) \tag{A.3}$$

The dependencies are as follows:

$$P_{bpre}(w_n) = S_{bpre} \cdot |w_n| + O_{bpre} \tag{A.4}$$

$$P_{caddr}(w'_n) = S_{caddr} \cdot |w'_n| + O_{caddr} \tag{A.5}$$

$$P_{senseamps}(w_n) = S_{senseamps} \cdot |w_n| + O_{senseamps} \tag{A.6}$$

Should the bit invert be active for a specific column, the inverted bit value must be used in these equations. No other alteration of the model is necessary. The driving related contribution comprises of the power consumed by the z bus and the output driving.

$$P_{driving}(w_{n-1}, w_n, OE) = P_{zbus}(w_{n-1}, w_n) + P_{outCycle}(w_{n-1}, w_n, OE) \tag{A.7}$$

with

$$\begin{aligned} P_{zbus}(w_{n-1}, w_n) = {} & S_{zdrive01} \cdot \#_{01}(w_{n-1}, w_n) \\ & + S_{zdrive10} \cdot \#_{10}(w_{n-1}, w_n) \\ & + S_{zdrive11} \cdot \#_{11}(w_{n-1}, w_n) \\ & + O_{driving} \end{aligned} \tag{A.8}$$

$$P_{outCycle}(w_{n-1}, w_n, OE) =$$
$$\begin{cases} S_{oca01} \cdot \#_{01}(w_{n-1}, w_n) + S_{oca10} \cdot \#_{10}(w_{n-1}, w_n) + O_{oca} & \text{if } OE = 0 \\ S_{ocd01} \cdot \#_{01}(w_{n-1}, w_n) + S_{ocd10} \cdot \#_{10}(w_{n-1}, w_n) + O_{ocd} & \text{if } OE = 1 \end{cases}$$

And last but not least:

$$P_{static} = O_{static} \tag{A.9}$$

A.3 Output (De-)Activation Related Power

The power related to the (de-)activation of the output is represented as follows: let w_o denote the old value at the output and w_n the content on the z bus (i.e. the last value read). Then the asynchronous output driving power is given as:

$$P_{output}(w_o, w_n, OE) = \begin{cases} P_{oact}(w_o, w_n) & \text{OE falling} \\ P_{odeact}(w_n) & \text{OE rising} \\ 0 & \text{otherwise} \end{cases} \tag{A.10}$$

with:

$$\begin{aligned} P_{oact}(w_o, w_n) = {} & S_{oact01} \cdot \#_{01}(w_o, w_n) \\ & + S_{oact10} \cdot \#_{10}(w_o, w_n) \\ & + S_{oact11} \cdot \#_{11}(w_o, w_n) \\ & + O_{oact} \end{aligned} \tag{A.11}$$

and:

$$P_{odeact}(w_o) = S_{deoact} \cdot |w_o| + O_{odeact} \tag{A.12}$$

A.4 Address Change Related Power

The address words are separated into the bit groups X, Y and Z according to their involvement in the row, column or block addressing. Let Xi, Yi and Zi denote respective i-th bit.

$$P_{address} = P_{X0} + P_{Xrest} + P_{Y0} + P_{Xrest} + P_Z \tag{A.13}$$

$$P_{X0} = \begin{cases} C_{X0r} & \text{X0 rising} \\ C_{X0f} & \text{X0 falling} \\ 0 & \text{otherwise} \end{cases} \tag{A.14}$$

$$P_{Y0} = \begin{cases} C_{Y0r} & \text{Y0 rising} \\ C_{Y0f} & \text{Y0 falling} \\ 0 & \text{otherwise} \end{cases} \tag{A.15}$$

$$P_Z = \sum_i P_{Z_i} \tag{A.16}$$

$$P_{Z_i} = \begin{cases} C_{Zir} & \text{Zi rising} \\ C_{Zif} & \text{Zi falling} \\ 0 & \text{otherwise} \end{cases} \tag{A.17}$$

$$P_{Yrest} = P_{Ydrv}(yaddr) + \sum_i P_{Y_i} \tag{A.18}$$

where *yaddr* is the y address. Like above:

$$P_{Yi} = \begin{cases} C_{Yir} & \text{Yi rising} \\ C_{Yif} & \text{Yi falling} \\ 0 & \text{otherwise} \end{cases} \tag{A.19}$$

and:

$$P_{Ydrv}(yaddr) = f_{Ydrv}(yaddr) \tag{A.20}$$

Let

$$b_i(xaddr) = xaddr AND' 11' \cdot 2^{2 \cdot i} \tag{A.21}$$

denote the output of the i-th x pre-decoder block at x address $xaddr$. Then:

$$P_{Xrest} = P_{Xdrv}(xaddr) + \sum_i P_{X_i} \tag{A.22}$$

$$P_{Xi} = \begin{cases} C_{Xir} & \text{Xi rising} \\ C_{Xif} & \text{Xi falling} \\ 0 & \text{otherwise} \end{cases} \tag{A.23}$$

$$P_{Xdrv} = \textstyle\sum_i f_{Xdrv}(i, b_i(xaddr)) \tag{A.24}$$

Here f_{Xdrv} is a function of the decoder block number and the output of the respective decoder block.

The number of coefficients for this extremely detailed type of address change model is high, but bearable. Let Xrest, Yrest and Z denote the number bits of the respective address part. Then the number of coefficients is:

$$NrCoeffs = 2 + 2 \cdot Xrest + 2 \cdot Xrest + 2 + 2 \cdot Yrest + 2 \cdot^{Y} rest + 2 \cdot Z \tag{A.25}$$

and:

$$max(NrCoeffs) = 2 + 2 \cdot 10 + 2 \cdot 10 + 2 + 2 \cdot 3 + 2 \cdot^{3} + 2 \cdot 6 = 60 \tag{A.26}$$

B Characterization Metafile Format

B.1 Introduction

As mention previously ORINOCO BEACH features a simple characterization description language. This limited language serves the concise specification of the experimental design. The language itself as well as its interpreter are based on the script language AWK [116]. In the remainder of this section an extended Backus-Naur-Form (EBNF) of the language is presented. Subsequently the functionality is briefly summarized and an example is given.

B.1.1 EBNF-description of the metafile

Here is a pseudo EBNF-description of the metafile. For easier notation the basic tokens Number, String and Variable are defined using regular expressions.

```
MetaFile = VarsSection {<NEWLINE>}
[BeginSection {<NEWLINE>}]
[ViewBeforeSection {<NEWLINE>}]
ExecuteSection {<NEWLINE>}
[ViewAfterSection {<NEWLINE>}]
[EndSection{<NEWLINE>}].

VarsSection = "[vars]" <NEWLINE>
{ Variable " = "
( "select " { String }+ |
  "toggle " { String }+ |
  "file " String |
  "for " Number " to " Number
      [ " step " { Number }+ ] ) <NEWLINE>}+.

BeginSection =
 "[begin]" <NEWLINE> AwkStringExpression <NEWLINE>.

ViewBeforeSection =
"[viewbefore]" <NEWLINE> AwkStringExpression <NEWLINE>.

ExecuteSection =
"[execute]" AwkStringExpression <NEWLINE>.

ViewAfterSection =
"[viewafter]" <NEWLINE> AwkStringExpression <NEWLINE>.

EndSection =
"[end]" <NEWLINE> AwkStringExpression <NEWLINE>.

AwkStringExpression =
```

```
[EmbracedString] (<variable> |
<variable>{(‘ ‘|EmbracedString) <variable>})
[EmbracedString].

EmbracedString = ’"’ String ’"’.

Number = "\-?[0-9]+(\.[0-9]+)?".
String = "[!#%&’*-,0-:<>@-Z\\]".
Variable = "[a-zA-Z][!#%&’*-,0-:<>@-Z\\]*".
```

Comments

- AwkStringExpressions contain everything that gawk evaluates to an expression that is valid in the form ”printf(AwkStringExpression)”. Variables, strings and even whole functions may occur within an AwkStringExpression (e.g. in the [execute]-section: ”echo ” 2 ^ a ” ” e).
- Hash and space must be preceded by the escape character “ ” in Strings. Precision of numbers is granted up to six digits.
- <NEWLINE> denotes the newline character. No other characters may follow.
- The AwkStringExpression in the ExecuteSection must evaluate to a valid shell-command.

B.1.2 Summary of the Functionality

The metafile is divided into separate chapters by the keywords: [vars], [begin], [viewbefore], [execute], [viewafter] and [end]. Of these only the two core sections [vars] and [execute] are mandatory. In the [vars] region the parameters of the characterization are defined as variables. Different generators describe which values the may be taken by which variable. Both the “select” and the “toggle” generator make the variable toggle through all strings given. In contrast to “select”, toggle also encompasses the empty string. The “file” generator makes the variable assume all lines in the specified file. The “for” generator has the usual semantics, with one exception: the statement allows to specify a sequence of step size. For the first characterization, the first step size is taken. With every new execution of the characterization, the respective next step size in the sequence is then used. Samples from previous runs are automatically identified and omitted from re-characterization.

The [execute] section contains the shell string that is executed for every combination of variable values specified in the [vars] section. The standard output of the executed shell command is redirected into the characterization table. It must be a single line.

The remaining sections only serve the purpose of formatting the characterization table: the strings defined in the [begin] and [end] sections appear in the table before respectively after the lines containing the characterization data. In every line of the table the content of the [viewbefore] string precedes the output of the executed shell command and the [viewafter] string follows it.

Example 9. The following small example file is shipped with BEACH:

```
[vars]
x1 = for 1 to 5 step 2
x2 = for 1 to 10 step 5 2 1
x3 = for 5 to 10 step 5
x4 = for 10 to 1 step -3
x5 = for 10 to 20 step 10 5

[begin]
"\#LABELS x1 x2 x3 x4 x5 pwr"

[viewbefore]
x1" \t"x2" \t"x3" \t"x4" \t"x5" \t"

[execute]
```

```
"gawk -v x1="x1" -v x2="x2" -v x3="x3" -v x4="x4"  -v x5="x5" \
 'BEGIN {print 1.7*log(x1)*x2 +0.4*x3*x3 +1*x5*x1 }'"
```

Performing two characterization runs with this metafile leads to the following characterization table:

```
# Run 2
#LABELS x1 x2 x3 x4 x5 pwr
1       1       5       10      10      20
1       1       5       10      15      25
1       1       5       10      20      30
1       1       5       7       10      20
1       1       5       7       15      25
1       1       5       7       20      30
1       1       5       4       10      20
1       1       5       4       15      25
1       1       5       4       20      30
...
5       9       10      7       20      164.624
5       9       10      4       10      114.624
5       9       10      4       15      139.624
5       9       10      4       20      164.624
5       9       10      1       10      114.624
5       9       10      1       15      139.624
5       9       10      1       20      164.624
```

C Pattern sequences for Register File Characterization

This chapter contains the patterns used for the gate level simulation of register files. As mentioned in section 7.1 four working modes where distinguished: idle, read, write and write-through. Separate pattern sequences were generated for these four cases. All sequences however can be expressed as functions of four common variables: $addr_1, addr_2, dat_1$ and dat_2. Here $addr_1$ and $addr_2$ are legal addresses for the instance under investigation and dat_1 and dat_2 are respective bit vectors. $addr_0$ and dat_0 furthermore represent the address and data word zero with appropriate bit width.

The tables C.1 to C.4 show the pattern sequences. The column 'pat' contains the number of the respective pattern. 'raddr', 'waddr' and 'wdata' list the read addresses, write addresses and write data using the number encoding defined above. The 'rst','cs','we' and 'clk' columns display the reset, chip select, write enable and clock signal. Note that the first three signals are active low. The horizontal line in the table separates the initialization phase from the power estimation. The power consumption is only computed for the patterns below the line.

Idle Clocking

For the characterization of the idle clocking power the circuit is reset (patterns 1–4). Then $addr_2$ is initialized with dat_1 (5–8). Reading this value once completes the initialization (9–12). In the part relevant for estimation the same read is performed again. Therefore no change whatsoever occurs at the register file's in- and outputs except the clocking.

Read Access

Here $addr_1$ is initialized with dat_1 and $addr_2$ with dat_2 (patterns 5–12). Then dat_1 is read from $addr_1$ (13–16). In the estimation phase (17–20) dat_2 is read from $addr_2$.

Write Access

dat_1 is written to $addr_2$. Then an arbitrary value is read from $addr_1$. During estimation dat_2 is written to $addr_2$, while the read address is $addr_1$. The content of $addr_2$ is therefore changed from dat_1 to dat_2 but dat_2 does not become visible at the output.

Write-through Access

The write-through access patterns are similar to the write case. Here, however, write and read address are identical during the last access so that the newly read word becomes immediately visible at the output.

Table C.1: Patterns for the characterization of register files: Idle Clocking.

pat	raddr	waddr	wdata	rst	cs	we	clk
1	0	0	1	0	1	1	0
2	0	0	1	0	1	1	1
3	0	0	1	0	1	1	1
4	0	0	1	0	1	1	0
5	1	2	1	1	0	0	0
6	1	2	1	1	0	0	1
7	1	2	1	1	0	0	1
8	1	2	1	1	0	0	0
9	2	1	2	1	1	1	0
10	2	1	2	1	1	1	1
11	2	1	2	1	1	1	1
12	2	1	2	1	1	1	0

Table C.1: (continued)

pat	raddr	waddr	wdata	rst	cs	we	clk
13	2	1	2	1	1	1	0
14	2	1	2	1	1	1	1
15	2	1	2	1	1	1	1
16	2	1	2	1	1	1	0

Table C.2: Patterns for the characterization of register files: Read Access.

pat	raddr	waddr	wdata	rst	cs	we	clk
1	0	0	1	0	1	1	0
2	0	0	1	0	1	1	1
3	0	0	1	0	1	1	1
4	0	0	1	0	1	1	0
5	0	1	1	1	0	0	0
6	0	1	1	1	0	0	1
7	0	1	1	1	0	0	1
8	0	1	1	1	0	0	0
9	0	2	2	1	0	0	0
10	0	2	2	1	0	0	1
11	0	2	2	1	0	0	1
12	0	2	2	1	0	0	0
13	1	0	2	1	1	1	0
14	1	0	2	1	1	1	1
15	1	0	2	1	1	1	1
16	1	0	2	1	1	1	0
17	1	0	2	1	1	1	0
18	2	0	2	1	1	1	0
19	2	0	2	1	1	1	0
20	2	0	2	1	1	1	0

Table C.3: Patterns for the characterization of register files: Write Access.

pat	raddr	waddr	wdata	rst	cs	we	clk
1	0	0	1	0	1	1	0
2	0	0	1	0	1	1	1
3	0	0	1	0	1	1	1
4	0	0	1	0	1	1	0
5	2	2	1	1	0	0	0
6	2	2	1	1	0	0	1
7	2	2	1	1	0	0	1
8	2	2	1	1	0	0	0
9	2	1	1	1	1	1	0
10	2	1	1	1	1	1	1
11	2	1	1	1	1	1	1
12	2	1	1	1	1	1	0
13	2	2	2	1	0	0	0
14	2	2	2	1	0	0	1
15	2	2	2	1	0	0	1
16	2	2	2	1	0	0	0

Table C.4: Patterns for the characterization of register files: Write-through Access.

pat	raddr	waddr	wdata	rst	cs	we	clk
1	0	0	1	0	1	1	0
2	0	0	1	0	1	1	1
3	0	0	1	0	1	1	1
4	0	0	1	0	1	1	0
5	1	2	1	1	0	0	0
6	1	2	1	1	0	0	1
7	1	2	1	1	0	0	1

Table C.4: (continued)

pat	raddr	waddr	wdata	rst	cs	we	clk
8	1	2	1	1	0	0	0
9	1	1	1	1	1	1	0
10	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1
12	1	1	1	1	1	1	0
13	1	2	2	1	0	0	0
14	1	2	2	1	0	0	1
15	1	2	2	1	0	0	1
16	1	2	2	1	0	0	0

D Experimental Data

This appendix contains the complete cross validation information of the evaluation examples described in chapter 7. The tables shown here are organized identically to those in evaluation chapter (for description see section 7.1) with the following exceptions: The 'type' information is moved into a headline for all models of the respective type. Consequently the 'type' column has been replaced by two columns 'C' and 'M': When a model is tested with several cross validation data sets, 'C' contains the number of the respective data set. In case the model building was performed using automatic insertion of cuts for piecewise regression, 'M' contains the number of the model in the modeling sequence. M = 1 stands for a model without cuts (shown as 'initial model' in chapter 7, while a model with M = i contains $i - 1$ cuts.

D.1 Philips SRAM

Table D.1: Complete cross validation data for the Philips SRAM.

C	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Interpolation									
1	1	0.8	1.000	4.6	0.8	-0.3	7455	-	-
2	1	0.8	1.000	9.3	1.2	-0.6	8667	-	-
3	1	0.8	1.000	8.1	1.6	-1.3	8263	-	-
4	1	0.9	1.000	4.6	1.2	-0.5	8684	-	-
Nonlinear Regression, Absolute Errors									
1	1	1.6	0.999	9.2	1.6	-1.0	4583	4	7
	2	1.2	1.000	3.8	1.3	0.1	5417	43	8
	3	1.1	1.000	3.5	0.9	-0.2	6167	102	8
	4	1.0	1.000	3.1	0.9	-0.3	6333	235	9
	5	1.0	1.000	3.2	0.9	-0.3	7167	497	10
2	1	1.6	1.000	7.6	1.8	-0.1	4333	5	7
	2	1.0	1.000	5.6	0.9	-0.1	5583	47	10
	3	0.8	1.000	5.0	1.0	-0.2	5250	123	10
	4	0.7	1.000	5.0	1.1	-0.2	6500	391	13
	5	0.7	1.000	4.1	0.9	-0.1	6875	793	13
	6	0.4	1.000	4.9	0.8	-0.2	6958	1456	13
3	1	1.5	1.000	8.5	2.1	-0.5	4348	6	7
	2	0.8	1.000	4.0	1.4	0.0	6130	51	10
	3	1.0	1.000	8.1	1.8	-0.7	6652	188	12
	4	0.8	1.000	3.5	1.3	0.0	6000	363	11
	5	0.7	1.000	3.6	1.2	-0.1	6130	738	13
	6	3.9	0.998	48.4	4.6	-3.9	8783	1450	13
4	1	1.8	1.000	12.3	2.7	-1.3	4792	7	8
	2	1.0	1.000	8.0	1.6	-0.3	7458	95	12
	3	2.5	0.999	9.5	1.4	0.2	8375	247	12
	4	0.9	1.000	4.2	1.1	-0.2	6833	574	13
	5	2.7	0.999	4.5	1.0	0.2	6750	1093	13
	6	1.0	1.000	4.7	1.1	0.0	8542	2161	14
	7	0.8	1.000	1.9	0.8	0.3	6792	3516	16
Linear Regression, Absolute Errors									
1	1	4.3	0.996	21.1	5.5	0.3	42	1	7
	2	2.7	0.998	13.1	3.2	-1.8	417	4	10
	3	1.0	1.000	3.6	1.1	-0.3	875	19	14
	4	1.0	1.000	3.4	1.0	0.6	1000	56	16

Table D.1: (continued)

C	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
	5	1.1	1.000	3.6	1.1	0.4	1042	121	16
	6	1.1	1.000	5.2	1.2	0.5	917	204	16
	7	1.2	1.000	3.2	1.1	0.5	1000	317	16
2	1	5.0	0.996	24.9	7.3	-2.9	83	1	7
	2	2.0	0.999	12.2	2.8	-1.2	458	5	11
	3	1.0	1.000	12.4	2.4	-0.4	792	22	14
	4	1.0	1.000	4.2	1.5	-0.6	917	65	16
	5	0.9	1.000	8.5	1.1	-0.3	1000	122	16
	6	0.9	1.000	3.4	1.0	0.1	917	205	16
	7	0.9	1.000	8.1	1.4	-0.3	1000	316	16
	8	0.8	1.000	5.7	1.1	0.0	1000	462	16
	9	0.9	1.000	3.9	1.0	0.0	1083	647	16
3	1	5.2	0.996	32.4	9.4	0.9	130	1	7
	2	1.8	1.000	9.3	2.7	-0.6	870	8	13
	3	1.6	1.000	7.9	1.8	0.3	957	36	15
	4	1.4	1.000	7.2	2.0	-0.2	1087	79	16
	5	0.9	1.000	4.6	1.4	-0.6	1043	137	16
	6	0.8	1.000	4.1	1.2	-0.4	1000	219	16
4	1	4.2	0.998	23.5	6.9	-3.9	42	1	7
	2	2.5	0.999	8.6	2.9	-1.7	125	3	9
	3	1.5	1.000	6.2	1.7	-0.9	792	18	14
	4	1.4	1.000	3.0	1.0	-0.1	1000	54	16
	5	1.2	1.000	2.9	1.0	0.0	1042	111	16
	6	1.2	1.000	2.9	1.0	0.0	1000	192	16
Nonlinear Regression, Relative Errors									
1	1	1.4	1.000	2.9	1.1	0.4	3917	3	6
	2	1.4	1.000	2.6	0.9	0.3	4958	24	7
	3	1.3	1.000	2.8	1.0	0.2	5875	127	9
2	1	1.4	1.000	2.3	1.0	0.0	4458	5	7
	2	1.9	1.000	2.7	1.0	-0.1	5667	74	9
	3	1.2	1.000	1.7	0.8	-0.1	5792	181	9
	4	1.0	1.000	1.2	0.5	0.1	9375	1642	14
3	1	1.6	1.000	2.9	1.5	0.3	4174	5	7
	2	1.2	1.000	2.4	1.0	0.3	5174	39	9
	3	1.9	1.000	2.9	1.3	-0.1	9130	311	13
	4	2.1	0.999	9.4	1.3	0.7	5696	476	10
4	1	13.3	0.982	15.9	3.2	-1.1	3125	2	5
	2	2.5	0.999	2.5	0.8	-0.4	6292	70	11
Linear Regression, Relative Errors									
1	1	5.6	0.993	9.5	2.5	-0.5	42	1	7
	2	5.4	0.993	9.1	2.1	0.3	125	3	9
	3	4.5	0.995	7.7	1.6	0.2	458	12	11
	4	1.6	0.999	4.0	1.0	0.1	792	49	15
	5	1.2	1.000	1.8	0.8	0.2	833	142	16
	6	1.0	1.000	2.0	0.7	0.0	2000	749	27
	7	0.7	1.000	1.7	0.4	0.1	3750	3016	40
	8	0.6	1.000	1.1	0.4	0.0	3333	5262	39
	9	0.8	1.000	1.6	0.5	0.2	3292	7931	36
2	1	7.5	0.992	10.0	3.5	-1.6	83	1	7
	2	8.0	0.991	11.2	2.5	-1.2	83	3	9
	3	6.8	0.994	9.8	1.7	-0.9	458	12	11
	4	1.4	1.000	4.2	0.9	-0.3	875	53	16
	5	0.8	1.000	1.8	0.5	-0.1	1583	230	22
	6	0.7	1.000	1.2	0.4	0.0	1958	629	26
	7	1.0	1.000	1.3	0.4	-0.1	3042	1893	36
	8	1.3	1.000	1.9	0.5	-0.2	2875	3065	32
3	1	5.7	0.996	9.5	3.0	-0.7	130	1	7
	2	5.6	0.996	9.3	1.8	0.4	174	3	9
	3	4.7	0.997	7.7	1.6	0.1	435	12	11
	4	1.4	1.000	2.1	0.7	0.2	913	50	16
	5	1.4	1.000	1.7	0.5	0.0	1826	230	23
	6	1.5	1.000	1.8	0.5	0.0	2826	737	29

Table D.1: (continued)

C	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
	7	0.7	1.000	1.9	0.5	0.0	4304	3434	42
	8	0.5	1.000	2.1	0.5	-0.1	3261	5410	36
4	1	5.7	0.997	9.2	2.6	-0.7	83	1	7
	2	5.6	0.997	9.1	2.0	0.2	83	4	10
	3	4.8	0.998	7.8	1.6	0.2	375	14	12
	4	2.7	0.999	3.5	1.1	0.0	833	51	15
	5	1.4	1.000	2.1	0.9	0.1	1250	177	19
	6	1.6	1.000	1.7	0.5	0.0	1708	497	23
	7	1.4	1.000	2.5	0.7	0.1	2625	1156	28
	8	1.2	1.000	1.7	0.4	0.1	2708	2181	28
	9	1.1	1.000	3.6	0.8	-0.4	3125	4108	33
	10	1.0	1.000	1.4	0.4	0.1	3542	8012	36
	11	1.5	1.000	2.0	0.4	0.0	5208	14276	44

D.2 Philips High Speed ROM

Table D.2: Complete cross validation data for the Philips High Speed ROM including voltages.

C	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Interpolation									
1		32.9	0.902	56.8	4.7	4.2	12350	-	-
2		1.4	1.000	3.2	1.3	0.5	17286	-	-
3		3.3	0.999	11.7	2.1	2.0	15250	-	-
4		35.4	0.897	100.8	9.3	-1.3	12300	-	-
Nonlinear Regression, Absolute Errors									
1	1	1.3	1.000	18.1	4.4	0.3	6167	14	7
	2	1.4	1.000	20.5	4.8	-0.3	6958	79	7
	3	1.3	1.000	17.3	4.0	0.0	7417	165	8
	4	1.3	1.000	10.1	2.6	1.1	9417	1140	14
2	1	2.3	0.999	7.7	2.1	-0.6	6000	14	7
	2	2.2	0.999	7.0	1.9	-0.6	7167	177	9
	3	2.0	1.000	6.8	2.0	-0.6	7000	372	11
	4	1.9	1.000	7.3	2.2	-0.3	11750	1035	14
3	1	1.3	1.000	59.6	5.7	2.2	7292	38	8
	2	1.2	1.000	29.5	3.9	0.8	7125	135	8
4	1	1.1	1.000	17.8	2.5	0.9	6208	154	7
	2	1.1	1.000	30.9	3.0	-1.0	6583	249	9
	3	1.0	1.000	25.5	3.1	1.2	8833	429	9
	4	1.1	1.000	11.7	2.2	0.4	9333	697	9
Linear Regression, Absolute Errors									
1	1	2.0	1.000	26.6	6.9	-5.2	167	1	8
	2	1.4	1.000	21.0	5.5	-4.2	250	7	10
	3	1.3	1.000	18.4	4.2	-1.2	750	24	12
	4	0.8	1.000	13.0	2.8	0.2	1417	174	18
	5	1.0	1.000	14.7	2.1	0.6	2500	618	27
	5	1.2	1.000	8.4	2.8	0.7	2292	1133	24
	6	1.2	1.000	36.4	4.5	0.9	2917	2375	30
	7	1.3	1.000	7.7	2.1	0.5	2958	3991	28
	8	1.3	1.000	6.9	2.3	0.1	3000	5601	30
	9	1.3	1.000	14.3	3.2	0.1	2792	7566	30
2	1	3.6	0.999	23.7	4.7	-2.9	167	2	8
	2	4.3	0.998	12.6	3.0	-1.2	625	11	11
	3	3.1	0.999	10.3	2.5	-0.8	917	38	14
3	1	1.8	1.000	41.5	5.8	2.1	167	1	8
	2	1.5	1.000	29.3	3.9	1.0	708	11	12
	3	1.3	1.000	22.5	3.1	0.3	1000	52	14
	4	2.3	0.999	15.5	2.3	0.8	2292	277	25
	5	2.3	0.999	34.0	3.3	-1.4	2125	599	23

Table D.2: (continued)

C	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
	6	2.0	1.000	16.9	1.8	0.9	2958	1448	30
	7	0.9	1.000	9.8	1.3	0.7	3042	2485	30
4	1	2.2	1.000	24.1	3.8	-0.6	167	2	8
	2	2.1	1.000	19.1	3.3	-0.3	667	12	11
	3	2.0	1.000	20.7	3.6	0.2	500	24	10
	4	1.4	1.000	10.1	2.1	-0.4	2125	210	23
	5	1.2	1.000	14.7	2.1	-0.9	2708	628	26
	6	1.3	1.000	10.9	2.2	-1.0	3167	1169	27
	7	1.4	1.000	8.1	1.4	0.0	3000	2158	30
	8	1.3	1.000	6.1	1.7	0.3	3167	3486	30
Nonlinear Regression, Relative Errors									
1	1	1.7	1.000	4.5	1.8	0.2	5292	9	6
2	1	3.4	0.999	6.5	2.1	0.1	5542	8	6
	2	10.4	0.989	15.9	5.9	-0.7	6167	22	5
3	1	1.6	1.000	7.3	1.7	0.8	5292	9	6
	2	1.9	1.000	6.5	2.5	0.2	5708	51	6
	3	1.8	1.000	3.6	0.8	0.1	7750	161	7
4	1	1.7	1.000	7.8	1.9	0.7	6917	21	8
	2	2.4	0.999	4.9	1.3	0.2	7125	94	8
	3	2.2	1.000	5.3	1.3	0.2	8542	251	9
Linear Regression, Relative Errors									
1	1	3.0	0.999	4.7	2.2	0.1	208	3	11
	2	2.5	0.999	5.5	2.1	0.0	708	12	12
	3	3.0	0.999	7.6	2.0	0.3	917	35	14
	4	1.6	1.000	4.7	1.4	0.2	1542	127	19
	5	1.6	1.000	7.5	1.3	0.5	2250	464	25
	6	1.5	1.000	6.4	1.3	0.4	2333	1029	27
2	1	5.8	0.996	9.7	3.1	-0.8	167	2	8
	2	4.9	0.997	9.8	2.8	-0.6	875	12	12
	3	5.5	0.997	11.7	2.7	-0.6	1250	51	16
	4	4.9	0.997	9.9	2.2	-0.5	1333	129	16
	5	2.9	0.999	8.0	1.6	-0.4	1958	318	22
	6	2.8	0.999	4.6	1.2	-0.2	1667	672	22
	7	2.3	0.999	3.9	1.1	-0.2	2250	1218	23
3	1	2.8	0.999	4.9	2.5	1.6	208	2	10
	2	2.2	0.999	5.2	2.2	1.1	750	12	13
	3	2.3	0.999	5.0	2.0	0.6	792	33	14
	4	1.1	1.000	3.6	1.2	0.5	1333	134	18
	5	2.7	0.999	3.6	1.5	0.4	2625	616	27
	6	2.0	1.000	3.4	1.3	0.5	2542	1187	27
	7	1.6	1.000	2.6	1.2	0.4	3083	2201	30
4	1	2.7	0.999	5.1	2.0	0.1	208	2	10
	2	2.0	1.000	4.1	1.5	-0.2	708	13	13
	3	2.9	0.999	5.6	1.7	-0.3	1083	56	15
	4	2.7	0.999	4.5	1.6	0.1	1833	152	20
	5	2.0	1.000	3.2	1.3	0.2	2292	482	24
	6	1.8	1.000	3.5	1.2	0.2	3375	1399	30
	7	1.7	1.000	3.6	1.1	0.3	3333	2339	30
	8	1.6	1.000	3.6	1.0	-0.1	3083	3470	30
	9	1.5	1.000	3.5	1.0	0.1	3208	4818	30

D.3 Philips Low Power ROM

Table D.3: Complete cross validation data for the Philips Low Power ROM including voltages.

C	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Interpolation									
1	1	14.8	0.979	128.6	17.4	-5.3	17051	-	-
2	1	17.2	0.979	101.2	13.4	0.6	17105	-	-
3	1	14.3	0.988	113.0	10.2	3.8	18457	-	-
4	1	11.5	0.988	91.6	10.7	2.4	15929	-	-
Nonlinear Regression, Absolute Errors									
1	1	3.0	0.999	22.7	4.4	0.3	14745	297	14
	2	3.6	0.999	20.7	4.4	0.1	11043	577	11
2	1	2.9	0.999	24.5	4.7	1.6	10872	136	10
	2	2.9	0.999	21.2	4.7	1.5	11830	494	11
	3	2.7	1.000	22.8	4.6	1.0	14936	2032	16
	4	2.7	1.000	36.3	4.2	0.4	14787	4125	16
	5	2.5	1.000	27.1	4.1	1.5	15894	9440	20
	6	4.7	0.998	40.5	4.4	1.8	17872	13990	16
3	1	2.9	0.999	51.5	6.2	-0.2	13283	544	15
	2	2.2	1.000	42.2	5.0	1.2	16543	1978	18
	3	2.2	1.000	49.7	6.2	1.1	15826	3333	15
4	1	2.3	1.000	20.4	4.6	1.0	10383	210	11
	2	2.3	1.000	51.0	5.5	2.3	14660	1194	16
	3	2.3	1.000	28.1	5.9	0.9	15596	3126	18
	4	2.6	0.999	21.9	4.0	0.1	12617	3845	10
	5	2.0	1.000	18.2	4.0	1.9	17872	8948	19
Linear Absolute									
1	1	8.8	0.993	88.6	26.0	-20.3	170	5	10
	2	6.3	0.996	73.7	13.7	-5.5	1064	45	15
	3	5.5	0.997	40.7	5.3	-1.5	1511	114	18
	4	6.2	0.996	66.1	9.2	-5.2	1553	226	18
	5	5.4	0.997	39.2	8.2	0.5	1596	398	18
	6	7.2	0.995	47.4	8.5	-2.1	1489	577	18
2	1	18.3	0.977	65.0	21.0	-12.2	170	5	9
	2	11.5	0.991	28.7	7.0	-0.9	1106	67	16
	3	8.2	0.995	47.4	8.3	-2.0	1830	192	20
	4	8.0	0.996	44.1	11.5	2.8	1468	304	20
3	1	18.9	0.975	65.5	20.7	-13.5	196	4	8
	2	11.7	0.990	16.8	6.2	0.3	1370	55	17
	3	9.0	0.994	19.2	6.1	0.5	1565	169	19
	4	6.6	0.997	49.2	7.9	-0.3	1587	290	20
	5	3.6	0.999	27.0	3.7	0.5	1696	481	20
	6	5.5	0.998	48.1	7.6	2.8	2087	702	20
4	1	17.1	0.974	69.0	16.7	-8.6	191	4	10
	2	12.8	0.985	26.9	6.3	-0.7	1447	54	17
	3	12.1	0.987	39.7	7.9	-2.6	1809	135	20
	4	8.3	0.994	34.7	7.9	0.8	1681	306	20
	5	6.2	0.997	40.7	8.0	0.1	2277	501	20
	6	7.1	0.995	55.3	11.8	0.0	1915	735	20
Nonlinear Relative									
1	1	2.9	0.999	28.9	4.2	-2.1	11766	180	11
	2	2.3	1.000	21.8	3.6	-1.0	13681	535	12
	3	17.3	0.973	45.4	10.4	-3.4	11149	2050	13
	4	14.8	0.980	42.9	13.3	0.0	7340	2558	9
2	1	5.0	0.998	14.4	4.1	-0.5	11362	231	13
	2	5.2	0.998	14.0	4.4	-0.3	10745	762	12
3	1	12.3	0.989	50.5	10.5	-1.2	6196	43	7
	2	14.2	0.986	45.0	9.7	-0.7	7174	202	8
	3	12.3	0.989	49.0	11.0	-0.5	6913	398	8
	4	1.9	1.000	18.1	2.9	-0.4	15630	1493	12
	5	2.4	1.000	19.4	3.0	-0.5	18435	4957	15
4	1	1.7	1.000	21.1	2.4	-0.3	11404	205	11

Table D.3: (continued)

C	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
	2	1.6	1.000	23.6	2.4	-0.3	14383	709	13
	3	1.8	1.000	22.4	2.2	-0.4	18681	2380	17
	4	1.9	1.000	21.9	2.4	0.0	14745	3707	12
	5	2.1	1.000	20.6	2.3	-0.1	14915	5433	14
	6	2.3	1.000	20.2	2.3	-0.3	13149	7109	13
Linear Relative									
1	1	10.2	0.990	71.3	15.2	-6.9	191	7	10
	2	3.1	0.999	36.4	3.9	-1.1	1362	62	19
	3	7.1	0.995	36.5	5.0	-1.2	1574	163	18
2	1	20.6	0.971	31.3	11.6	-1.1	213	8	13
	2	12.9	0.988	12.9	4.4	1.0	1723	61	20
	3	10.9	0.992	15.6	3.7	0.5	1553	149	20
	4	10.2	0.993	14.9	3.7	0.9	1830	299	20
	5	10.6	0.992	15.6	3.8	0.8	1830	511	20
	6	9.1	0.994	14.7	4.2	0.2	1979	754	20
	7	8.4	0.995	21.8	4.5	1.1	1936	1171	20
3	1	21.8	0.967	44.2	11.5	-4.8	283	10	13
	2	13.3	0.988	13.3	4.0	0.0	1326	51	20
	3	11.2	0.991	12.7	3.5	0.2	1457	123	20
	4	9.5	0.994	10.3	3.0	-0.1	1739	288	20
	5	8.9	0.994	14.2	3.2	0.3	1696	523	20
	6	11.5	0.991	24.1	5.3	0.9	2065	785	20
	7	8.3	0.995	18.1	4.8	0.0	1891	1083	20
4	1	19.8	0.965	39.3	12.1	-0.3	191	7	11
	2	13.1	0.985	22.5	4.7	-0.4	1404	63	20
	3	9.8	0.992	24.0	4.4	0.2	1638	161	20
	4	8.9	0.993	27.1	4.7	0.0	1809	303	20
	5	8.0	0.994	27.5	5.4	-0.7	1851	504	20
	6	13.9	0.983	30.2	5.8	0.7	2064	729	20

D.4 LSI m11_111ha Embedded SRAM

Table D.4: Complete cross validation data for LSI m11_111ha Embedded SRAM.

C	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Interpolation									
1		0.1	1.000	1.0	0.0	0.0	42297	-	-
2		0.1	1.000	0.6	0.0	0.0	11541	-	-
3		0.1	1.000	1.0	0.0	0.0	12157	-	-
4		0.1	1.000	1.3	0.0	0.0	12521	-	-
Nonlinear Regression, Absolute Errors									
1	1	1.9	0.999	19.2	2.0	0.2	1344	21	3
	2	1.3	0.999	20.5	1.5	0.2	2409	61	6
	3	1.3	0.999	13.3	1.4	0.1	1429	95	5
	4	1.3	0.999	13.1	1.4	0.1	1457	150	6
	5	1.3	0.999	11.9	1.3	0.1	2465	347	9
2	1	1.9	0.998	16.6	2.0	0.2	1372	22	3
	2	1.3	0.999	13.6	1.4	0.0	1429	57	6
	3	1.2	0.999	16.5	1.4	0.1	1457	103	6
	4	1.2	0.999	13.4	1.2	0.0	1933	232	9
	5	1.0	1.000	13.9	1.2	0.1	1933	469	9
3	1	1.9	0.998	15.3	2.0	0.2	1316	22	3
	2	1.3	0.999	17.0	1.5	0.2	3165	73	7
	3	1.3	0.999	16.0	1.5	0.1	1429	109	5
	4	1.3	0.999	14.5	1.4	0.1	1457	165	6
	5	1.3	0.999	14.5	1.4	0.1	1401	238	6
4	1	1.9	0.998	17.3	2.0	0.2	1372	22	3
	2	1.3	0.999	15.6	1.4	0.2	3613	75	8

Table D.4: (continued)

C	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
	3	1.2	0.999	13.5	1.4	0.2	1429	129	6
	4	1.1	0.999	11.8	1.3	0.1	1260	227	8
	5	1.1	0.999	13.5	1.4	0.1	1905	377	8
Linear Regression, Absolute Errors									
1	1	2.5	0.997	15.1	2.6	-0.5	56	22	4
	2	1.7	0.999	15.8	2.0	-0.6	308	42	7
	3	0.6	1.000	5.1	0.6	0.0	896	122	15
	4	0.4	1.000	5.1	0.4	0.0	1625	443	21
	5	0.3	1.000	6.1	0.4	0.0	2521	1164	26
	6	0.2	1.000	4.7	0.1	0.0	2885	2586	30
	7	0.1	1.000	4.8	0.1	0.0	3529	5379	36
2	1	2.5	0.997	16.4	2.6	-0.6	84	22	4
	2	1.7	0.999	17.3	2.1	-0.6	336	42	7
	3	0.6	1.000	4.6	0.6	0.0	1204	117	16
	4	0.5	1.000	4.1	0.5	0.0	2129	425	22
	5	0.3	1.000	4.1	0.4	0.0	2185	1367	23
	6	0.2	1.000	2.5	0.1	0.0	3137	2842	32
	7	0.1	1.000	2.9	0.1	0.0	3754	5921	36
	8	0.1	1.000	1.3	0.1	0.0	4874	12099	45
3	1	2.6	0.997	16.2	2.7	-0.6	56	21	4
	2	1.7	0.999	17.1	2.1	-0.6	308	41	7
	3	0.7	1.000	4.8	0.6	0.0	1597	138	15
	4	0.5	1.000	4.5	0.5	0.0	1821	382	20
	5	0.3	1.000	4.5	0.4	0.0	2157	1102	23
	6	0.2	1.000	1.9	0.1	0.0	3249	2965	34
	7	0.1	1.000	1.7	0.1	0.0	4230	6836	38
	8	0.1	1.000	2.3	0.1	0.0	5322	14795	46
4	1	2.6	0.997	15.7	2.6	-0.6	84	21	4
	2	1.7	0.999	16.5	2.1	-0.6	336	41	7
	3	0.7	1.000	5.7	0.6	0.0	1092	112	15
	4	0.5	1.000	5.2	0.5	0.0	2241	388	21
	5	0.3	1.000	5.2	0.4	0.0	2605	1186	28
	6	0.2	1.000	2.8	0.1	0.0	3642	3289	35
	7	0.1	1.000	2.9	0.1	0.0	3726	6259	35
	8	0.1	1.000	2.1	0.1	0.0	5014	13344	48
Nonlinear Regression, Relative Errors									
1	1	2.2	0.998	7.7	1.7	0.0	1372	22	3
	2	1.6	0.999	7.0	1.3	0.0	1989	58	7
	3	1.6	0.999	7.0	1.3	0.0	1933	115	7
	4	1.6	0.999	7.0	1.3	0.0	2017	195	7
	5	1.3	0.999	5.9	1.0	0.0	3642	709	11
	6	1.2	0.999	6.3	1.0	0.0	4034	1352	11
	7	1.4	0.999	7.1	1.1	0.0	3473	1776	8
	8	1.4	0.999	7.1	1.1	0.0	3445	2374	8
2	1	2.2	0.998	7.6	1.8	-0.1	1344	21	3
	2	1.4	0.999	7.0	1.2	0.0	2549	57	6
	3	1.6	0.999	7.2	1.3	0.0	3670	275	10
	4	1.3	0.999	6.8	1.0	0.0	4986	919	14
	5	1.6	0.999	9.5	1.2	0.0	2129	1000	6
3	1	2.2	0.998	7.3	1.7	-0.1	1372	22	3
	2	1.9	0.998	9.0	1.5	0.0	1344	44	4
	3	1.4	0.999	6.8	1.1	0.0	2325	102	7
	4	1.3	0.999	5.2	1.1	0.0	3445	332	10
	5	1.3	0.999	5.8	1.0	0.0	4342	781	12
	6	1.3	0.999	5.5	1.0	0.0	4566	1718	13
	7	1.5	0.999	6.5	1.1	0.0	4258	2874	14
4	1	2.2	0.998	6.6	1.7	0.0	1372	21	3
	2	1.7	0.999	6.8	1.4	0.0	2241	58	6
	3	1.5	0.999	6.6	1.1	0.0	3445	162	9
	4	1.5	0.999	6.0	1.2	0.0	2885	355	8
	5	1.5	0.999	6.9	1.2	0.0	1877	436	6
	6	1.5	0.999	6.9	1.2	0.0	1765	566	6

Table D.4: (continued)

C	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Linear Regression, Relative Errors									
1	1	3.0	0.996	7.0	2.1	0.0	56	22	4
	2	2.0	0.998	4.8	1.5	0.0	336	42	8
	3	0.6	1.000	4.9	0.6	0.0	952	118	15
	4	0.4	1.000	4.9	0.5	0.0	1344	304	20
	5	0.3	1.000	5.2	0.4	0.0	2157	976	22
	6	0.2	1.000	4.6	0.1	0.0	2409	2155	30
	7	0.2	1.000	4.7	0.1	0.0	3277	4468	34
	8	0.1	1.000	3.0	0.1	0.0	3922	9937	41
	9	0.1	1.000	3.0	0.1	0.0	4510	34374	43
	10	0.1	1.000	3.0	0.1	0.0	5098	51873	45
2	1	3.0	0.996	8.4	2.1	-0.1	56	28	4
	2	2.0	0.998	5.6	1.5	0.0	364	55	8
	3	0.6	1.000	5.1	0.6	0.0	924	149	15
	4	0.5	1.000	5.1	0.5	0.0	1933	403	21
	5	0.3	1.000	4.2	0.4	0.0	1737	1277	22
	6	0.2	1.000	2.6	0.1	0.0	2605	2658	30
	7	0.2	1.000	2.5	0.1	0.0	3193	5308	35
3	1	3.0	0.996	8.1	2.1	-0.1	56	21	4
	2	2.0	0.998	5.3	1.5	0.0	336	41	8
	3	0.7	1.000	4.9	0.6	0.0	728	103	12
	4	0.5	1.000	4.9	0.5	0.0	1401	238	18
	5	0.3	1.000	4.2	0.4	0.0	1597	770	22
	6	0.2	1.000	1.9	0.2	0.0	2913	2515	32
	7	0.2	1.000	1.7	0.1	0.0	3249	5323	35
	8	0.1	1.000	2.6	0.1	0.0	3978	11744	40
	9	0.1	1.000	2.5	0.1	0.0	4622	21281	41
	10	0.1	1.000	1.6	0.1	0.0	4426	34351	41
4	1	3.0	0.996	7.6	2.1	-0.1	28	21	4
	2	2.0	0.998	5.1	1.5	0.0	336	42	8
	3	0.7	1.000	5.4	0.6	0.0	924	117	15
	4	0.5	1.000	5.4	0.5	0.0	1372	284	20
	5	0.3	1.000	5.0	0.4	0.0	2577	1007	26
	6	0.2	1.000	2.8	0.1	0.0	2913	3270	32
	7	0.2	1.000	2.9	0.1	0.0	3726	6344	38
	8	0.1	1.000	2.2	0.1	0.0	4482	13210	42
	9	0.1	1.000	2.2	0.1	0.0	4034	21207	42
	10	0.1	1.000	2.2	0.1	0.0	4678	34041	48

D.5 Wallace Tree Multiplier Module

Table D.5: Complete cross validation data for wallace tree multiplier module.

CV	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Interpolation									
1	1	17.0	0.933	65.7	12.3	0.6	3621	-	-
Nonlinear Absolute									
1	1	14.5	0.951	114.6	12.5	-0.8	1235	0	3
	2	15.0	0.948	94.1	12.9	-1.5	2510	8	6
	3	14.6	0.951	64.5	11.6	-0.3	2263	13	4
	4	14.5	0.951	56.8	11.3	0.1	1893	23	4
	5	14.6	0.950	76.1	11.7	-0.7	2840	52	5
Linear Absolute									
1	1	14.6	0.950	71.2	12.2	2.6	62	1	3
	2	14.8	0.949	61.4	12.0	2.6	144	1	5
	3	15.1	0.947	82.7	12.8	3.4	123	2	5
	4	15.0	0.948	52.8	12.0	1.8	144	5	6
	5	15.1	0.947	53.3	11.4	0.9	761	18	10
	6	15.0	0.948	60.0	11.4	0.8	1049	48	11

Table D.5: (continued)

CV	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
	7	15.0	0.948	60.0	11.4	0.8	988	90	11
Nonlinear Relative									
1	1	15.6	0.944	50.0	11.2	-2.9	1296	1	3
	2	15.7	0.943	47.5	10.8	-3.5	2428	7	5
	3	15.7	0.943	116.6	10.9	-3.9	4342	49	8
	4	18.0	0.925	80.3	12.1	-3.9	3519	93	7
	5	18.0	0.925	80.3	12.1	-3.9	3663	159	7
Linear Relative									
1	1	15.5	0.944	48.6	11.2	-2.9	82	1	4

D.6 Sine Module

Table D.6: Complete cross validation data for sine module.

CV	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Interpolation									
1	1	25.9	0.878	36.6	8.7	-4.2	4291	-	-
Nonlinear Absolute									
1	1	32.5	0.806	43.0	14.7	-2.6	1982	0	3
	2	18.7	0.936	80.0	12.5	-2.8	3382	6	6
	3	12.1	0.973	44.4	9.9	-2.3	1927	16	7
	4	89.7	-0.476	202.7	19.0	13.2	3782	94	11
Linear Absolute									
1	1	46.5	0.603	56.4	16.4	-9.4	1982	0	3
	2	35.1	0.774	72.4	18.2	-1.5	3400	5	5
Nonlinear Relative									
1	1	37.7	0.740	135.2	23.8	-0.8	109	0	3
	2	23.2	0.901	157.7	16.3	1.5	91	0	3
Linear Relative									
1	1	15.8	0.451	57.2	26.2	-18.6	91	0	2

Table D.7: Second cross validation data set for sine module.

CV	M	RMS	r^2	XARE	MARE	ARE	t_{eval}	t_{gen}	Size
Interpolation									
2	1	19.5	0.927	74.5	10.4	2.7	4473	-	-
Nonlinear Absolute									
2	1	29.4	0.844	68.9	19.9	4.6	1455	0	3
	2	16.9	0.949	210.3	18.9	7.9	1018	4	4
	3	9.3	0.984	31.3	6.3	1.1	1527	24	7
Linear Absolute									
2	1	33.4	0.799	43.2	14.3	-1.5	2000	1	4
	2	28.3	0.856	55.7	12.7	-2.0	1873	5	5
	3	11.6	0.976	53.1	8.2	-0.1	1891	67	8
Nonlinear Relative									
2	1	35.5	0.773	169.0	30.7	11.6	55	1	3
	2	21.7	0.915	158.1	18.4	8.1	91	1	4
	3	9.7	0.983	85.7	8.5	3.1	364	5	7
	4	10.7	0.979	87.9	9.2	3.4	418	13	8
	5	10.3	0.981	45.8	6.9	1.9	836	58	12
	6	11.7	0.976	43.4	8.9	1.9	782	90	11
Linear Relative									
2	1	45.5	0.627	74.4	26.2	-5.5	55	0	2
	2	38.1	0.738	62.0	20.0	-3.2	145	1	6
	3	18.5	0.938	29.8	8.9	0.1	182	3	6

List of Figures

List of Tables

Bibliography

[1] Y. Agata, K. Motomochi, F. Yoshifumi, M. Shirahama, M. Kurumada, N. Kuroda, H. Sadakata, K. Hayashi, T. Yamada, K. Takahashi, and T. Fujita. An 8-ns random cycle embedded RAM macro with dual-port interleaved DRAM architecture (d^2ram). In *IEEE Journal of Solid-State Circuits*, volume 35, no. 11, pages 1668–1671, November 2000.

[2] K. Agawa, H. Hara, T. Takayanagi, and T. Kuroda. A bitline leakage compensation scheme for low-voltage SRAMs. In *IEEE Journal of Solid-State Circuits*, volume 36, no. 10, October 2001.

[3] A. Allara, M. Bombana, A. Stammermann, E. Schmidt, L. Kruse, and W. Nebel. VHDL behavioural power estimations for telecom devices. In *Forum for Desgin Languages (FDL)*, 2001.

[4] B. Amrutur and M. Horowitz. Speed and power scaling of SRAM's. In *IEEE Transactions on solid-state circuits*, volume 35 no. 5, pages 175–185, Febuary 2000.

[5] B. S. Amrutur and M. A. Horowitz. Fast low-power decoders for RAMs. In *IEEE Journal of Solid-State Circuits*, volume 36, no. 10, October 2001.

[6] M. Barocci, L. Benini, A. Bogliolo, B. Ricco, and G. De Micheli. Lookup table power macro-models for behavioral library component. In *IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, pages 173–181, Como, Italy, march 1999. IEEE Computer Society.

[7] D. Bates and D. Watts. *Nonlinear Regression Analysis and its applications*. John Wiley & Sons, 1988.

[8] C. Beightler and D. Phillips. *Applied Geometric Programming*. John Wiley & Sons Inc., 1976.

[9] N. Bellas, I. Hajj, and C. Polychronopoulos. Using dynamic cache management techniques to reduce energy in a high-performance processor. In *Proceedings 1999 international symposium on Low power electronics and design*, pages 64–69. ACM Press, 1999.

[10] L. Benini, A. Bogliolo, M. Favalli, and G. De Micheli. Regression models for behavioral power estimation. In *Int. Works. Power Timing Modelling Performance Integrated Circuits (PATMOS)*, pages 179–187, 1996.

[11] L. Benini, L. Macchiarulo, A. Macii, E. Macii, and M. Poncino. From architecture to layout: Partitioned memory synthesis for embedded systems-on-chip. In *Design Automation Conference (DAC)*, pages 784–789, Las Vegas, June 2001.

[12] L. Benini, A. Macii, and M. Poncino. Synthesis of application-specific memories for power optimization in embedded systems. In *Proceedings of the 2000 international symposium on Low power electronics and design*, pages 300–303, 2000.

[13] G. Bernacchia and M. C. Papaefthymiou. Analytical macromodeling for high-level power estimation. In *Proceeding of the 1999 international conference on Computer-aided design*, pages 280–283. IEEE Press, 1999.

[14] C. Bingham and G. W. Oehlert. *An Introduction to MacAnova*. Univ. of Minnesota, School of Statistics, version 4.12 edition, 2001. `http://www.stat.umn.edu/macanova/macanova.home.html`.

[15] A. Bogliolo, L. Benini, and G. De Micheli. Adaptive least mean square behavioral power modeling. In *ED&TC'97*, 1997.

[16] A. Bogliolo, R. Corgnati, E. Macii, and M. Poncino. Parameterized RTL power models for combinational soft macros. In *Proceedings of the IEEE-ACM International Conference on Computer Aided Design*, pages 284–287, 1999.

[17] A. Bogliolo, R. Corgnati, E. Macii, and M. Poncino. Parameterized RTL power models for soft macros. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, volume 9, no. 6, pages 880–887, December 2001.

[18] I. Bomze and W. Grossmann. *Optimierung - Theorie und Algorithmen*. BI Wissenschaftsverlag, 1993. in German.

[19] G. Box and N. Draper. *Empirical Model-Building and Response Surfaces*. John Wiley & Sons, 1987.

[20] P. Box and P. Tidwell. Transformations of the independent variables. In *Technometrics*, volume 4, no. 4, november 1962.

[21] E. Brockmeyer, L. Nachtergaele, F. V. M. Catthoor, J. Bormans, and H. J. De Man. Low power memory storage and transfer organization for the MPEG-4 full pel motion estimation on a multimedia processor. In *IEEE Transactions on Multimedia*, volume 1, no. 2, pages 202–216, June 1999.

[22] E. Brockmeyer, A. Vandecapelle, and F. Catthoor. Systematic cycle budget versus system power trade-off: a new perspective on system exploration of real-time data-dominated applications. In *Proceedings of the IEEE International Symposium on Low Power Eletronics and Design*, 2000.

[23] R. Camposano and W. Wolf, editors. *High-Level VLSI Synthesis*. Kluwer Academic Publishers, 1991.

[24] F. Catthoor, S. Wuytack, E. De Greef, F. Balasa, L. Nachtergaele, and A. Vandecappelle. *Custom Memory Organisation for Embedded Multimedia System Design*. Kluwer Academic Publishers, 1998.

[25] A. Chandna, C. D. Kibler, R. B. Brown, M. Roberts, and K. A. Sakallah. The aurora RAM compiler. In *Design Automation Conference*, 1995.

[26] A. P. Chandrakasan and R. W. Brodersen. *Low Power Digital CMOS Design*. Kluwer Academic Publishers, 1995.

[27] A. P. Chandrakasan and R. W. Brodersen. *Low Power CMOS Design*. IEEE Press, 1998.

[28] C.-R. Chang, J.-S. Wang, and C.-H. Yang. Low-power high-speed ROM modules for ASIC applications. In *IEEE Journal of Solid-State Circuits*, volume 36, no. 10, October 2001.

[29] Z. Chen, K. Roy, and E. K. P Chong. Estimation of power dissipation using a novel power macromodeling technique. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 19, no. 11, pages 1363–1369, 2000.

[30] M. Chinosi, R. Zafalon, and C. Guardino. Automatic characterization and modelling of power consumption in static RAMs. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 112–114, 1998.

[31] *User's Guide of COPL_GP - Computational Optimization Program Library: Geometric Programming*, May 2000.

[32] S. Coumeri and D Thomas. Memory modelling for system synthesis. In *IEEE Transactions on VLSI Systems*, volume 8, no. 3, june 2000.

[33] W. Daems, G. Gielen, and W. Sansen. Simulation-based automatic generation of signomial and posynomial performance models for analog integrated circuit sizing. In *International Conference on Computer Aided Design (ICCAD)*, 2001.

[34] DARPA, editor. *DARPA Neural Network Study*. AFCEA International Press, 1990.

[35] E. de Angel and Jr. E. E. Swartzlander. Survey of low power techniques for ROMs. In *Proceedings of the 1997 international symposium on Low power electronics and design*, pages 7–11. ACM Press, 1997.

[36] IEEE Standards Department. P1497 draft standard for standard delay format(SDF) for the electronic design process. `http://www.eda.org/sdf/`, 2001.

[37] C. Edwards, P. Clarke, and S. Ohr. Power problems threaten launch of 3G handsets. *EETimes, 23.1.2001*, 2001.

[38] M. Elrabaa, I. Abu-Khater, and M. Elmasry. *Advanced Low-Power Digital Circuit Techniques*. Kluwer Academic Publishers, 1997.

[39] R. Evans and P. Franzon. Energy consumption modelling and optimization for SRAM's. In *IEEE Transactions on solid-state circuits*, volume 30 no. 5, pages 571–579, may 1995.

[40] A. Ferre and J. Firueras. Leakage power bounds in cmos digital technologies. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 21, no. 6, pages 731–738, June 2002.

[41] M. Fowler and K. Scott. *UML Distilled. Applying the Standard Object Modeling Language*. Addison Wesley Longman, 1st corrected reprint edition, 1998.

[42] K. Ghose and M. B. Kamble. Reducing power in superscalar processor caches using subbanking, multiple line buffers and bit-line segmentation. In *Proceedings 1999 international symposium on Low power electronics and design*, pages 70–75. ACM Press, 1999.

[43] T. D. Givargis, F. Vahid, and J. Henkel. Fast cache and bus power estimation for parameterized system-on-a-chip design. In *Design, Automation and Test in Europe (DATE)*, pages 333–338, Paris, France, 2000.

[44] T. D. Givargis, F. Vahid, and J. Henkel. Evaluating power consumption of parameterized cache and bus architectures in system-on-a-chip designs. In *IEEE Transactions on Very Large Scale (VLSI) Systems*, volume 9, no. 4, pages 500–508, August 2001.

[45] W. Glichrist. *Statistical Modelling*. John Wiley & Sons, 1984.

[46] G. Grimmett and D. Strirzaker. *Probability and Random Processes*. Oxford Science Publications, 1992.

[47] P. Grun, N. Dutt, and A. Nicolau. Access pattern based local memory customization for low power embedded systems. In *Design, Automation and Test in Europe (DATE)*, pages 778–784, Munich, Germany, 2001.

[48] S. Gupta and F. Najm. Power macromodelling for high level power estimation. In *34th ACM/IEEE Design Automation Conference*, pages 365–370, june 1997.

[49] S. Gupta and F. Najm. Analytical models for RTL power estimation of combinational and sequential circuits. In *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, volume 19, no. 7, pages 808–814, july 2000.

[50] S. Gupta and F. Najm. Power modeling for high level power estimation. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, volume 8, no. 1, pages 18–29, February 2000.

[51] F. Hamzaoglu, Y. Te, A. Keshavarzi, K. Zhang, S. Narendra, S. Borkar, M. Stan, and V. De. Dual-VT SRAM cells with full-swing single-ended bit line sensing for high-performance on-chip cache in 0.13 μm technology generation. In *Proceedings of the 2000 international symposium on Low power electronics and design*, pages 15–19. ACM Press, 2000.

[52] T. Haraszti. *CMOS Memory Circuity*. Kluwer Academic Publishers, 2000.

[53] S. Hein and N. Wehn. Embedded DRAM applications and challenges. In *DAC Tutorial: Embedded Memories in System Design*. Siemens Semiconductor Group, 1999.

[54] M. Hershenson, S. S. Mohan, S. P. Boyd, and T. H. Lee. Optimization of inductor circuits via geometric programming. In *Design Automation Conference*, 1999.

[55] J. Hezavei, N. Vijaykrishnan, and M. J. Irwin. A comparative study of power efficient SRAM designs. In *Great Lake Symposium on VLSI 2000*, pages 117–122, 2000.

[56] P. Hicks, M. Walnock, and R. M. Owens. Analysis of power consumption in memory hierarchies. In *Proceedings of the 1997 international symposium on Low power electronics and design*, pages 239–242. ACM Press, 1997.

[57] J. Hjorth. *Computer Intensive Statistical Methods*. Chapman & Hall, London, England, 1994.

[58] C. X. Huang, B. Zhang, A.-C. Deng, and B. Swirski. The design and implementation of PowerMill. In *ISLPD'95 Symposium Proceedings*, 1995.

[59] S. i. Mianato. *Binary Decision Diagrams and Applications for VLSI CAD*. Kluwer Academic Publishers, 1996.

[60] *IEEE Journal of Solid-State Circuits*, volume 34, no. 5, May 1999.

[61] *IEEE Journal of Solid-State Circuits*, volume 34, no. 11, November 1999.

[62] H. Ikeda and H. Inukai. High-speed DRAM architecture development. In *IEEE Journal of Solid-State Circuits*, volume 34, no. 5, pages 685–692, May 199.

[63] K. Inoue, T. Ishihara, and K. Murakami. Way-predicting set-associative cache for high performance and low energy consumption. In *Proceedings 1999 international symposium on Low power electronics and design*, pages 273–275. ACM Press, 1999.

[64] K. Inoue, V. G. Moshnyaga, and K. Murakami. A history-based I-cache for low-energy multimedia applications. In *Proceedings of the International Symposium on on Low Power Electronics and Design (ISLPED)*, pages 148–153, August 2002.

[65] International Technology Roadmap for Semiconductors, SEMATECH, 3101 Industrial Terrace Suite 106, Austin TX 78758. *International Technology Roadmap for Semiconductors (2000 update)*, 2000.

[66] International Technology Roadmap for Semiconductors, SEMATECH, 3101 Industrial Terrace Suite 106, Austin TX 78758. *International Technology Roadmap for Semiconductors*, 2001.

[67] T. Ishihara and H. Yasuura. A power reduction technique with object code merging for application specific embedded memories. In *Design, Automation and Test in Europe (DATE)*, pages 617–623, Paris, France, 2000.

[68] K. Itoh. Low-voltage memories for power-aware systems. In *Proceedings of the International Symposium on Low Power Eletronics and Design*, August 2002.

[69] K. Itoh, K. Sasaki, and N. Yoshinobu. Trends in low-power RAM circuit technologies. In *Proceedings of the IEEE*, volume 83, no. 4, April 1995.

[70] G. Jochens, L. Kruse, E. Schmidt, and W. Nebel. A new parameterizable power macro-model for datapath components. In *DATE'99*, pages 29 – 36, Munich, Germany, 1999.

[71] G. Jochens, L. Kruse, E. Schmidt, A. Stammermann, and W. Nebel. Power macro-modelling for firm-macros. In *PATMOS'00*, Goettingen, Germany, 2000.

[72] M. Kamble and K. Ghose. Analytical energy dissipation models for low power caches. In *Proc. 1997 Int'l Symp. on Low Power Electronics and Design*, 1997.

[73] D. Karchmer and J. Rose. Definition and solution of the memory packing problem for field-programmable systems. In *International Conference on Computer-Aided Design (ICCAD)*, November 1994.

[74] D. Keitel-Schulz and N. Wehn. Issues in embedded dram development and applications. In *Proceedings on 11th international symposium on System synthesis*, pages 23–31. IEEE Computer Society, 1998.

[75] Keyes. The future of the transistor. *Scientific American*, March 1998.

[76] M. M. Khellah and M. I. Elmasry. A low-power high-performance current-mode multiport SRAM. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, volume 9, no. 5, October 2001.

[77] K.O. Kortanek, X. Xu, and Y. Ye. An infeasible interior-point algorithm for solving primal and dual geometric programs. In *Mathematical Programming 76*, pages 155–181, 1996.

[78] L. Kruse, E. Schmidt, G. Jochens, and W. Nebel. Low power binding heuristics. In *PATMOS'99*, pages 41–50, Kos, Greece, 1999.

[79] L. Kruse, E. Schmidt, G. Jochens, and W. Nebel. Lower and upper bounds on the switching activity in scheduled data flow graphs. In *International Symposium on Low Power Electronics and Design (ISLPED'99)*, pages 115–120, San Diego, California, 1999.

[80] L. Kruse, E. Schmidt, G. Jochens, A. Stammermann, and W. Nebel. Lower bound estimation for low-power high-level synthesis. In *13th International Symposium on System Synthesis (ISSS 2000), Invited Paper*, 2000.

[81] L. Kruse, E. Schmidt, G. Jochens, A. Stammermann, and W. Nebel. Lower bounds on the power consumption in scheduled data flow graphs with resource constraints. In *Design, Automation and Test in Europe (DATE)*, page p. 737, Paris, France, 2000.

[82] L. Kruse, E. Schmidt, G. Jochens, A. Stammermann, M. Schulte, E. Macii, and W. Nebel. Estimation of lower and upper bounds on the power consumption from scheduled data flow graphs. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, volume 9 no. 1, pages 3–14, feb 2001.

[83] J. Landman, P. Rabaey. Architectural power analysis: The dual bit type method. In *IEEE Transactions on VLSI Systems*, volume 3, no. 2, pages 163–187, june 1995.

[84] Y. Li and J. Henkel. A framework for estimating and minimizing energy dissipation of embedded hw/sw systems. In *Proceedings of the Design Automation Conference (DAC)*, pages 188–193, San Francisco, CA, USA, June 1998.

[85] D Liu and C. Svensson. Power consumption estimation in CMOS VLSI chips. In *IEEE Journal of Solid-State Circuits*, volume 29, no. 6, pages 663–670, june 1994.

[86] LSI Logic Corporation, 1551 McCarthy Boulevard, Milpitas, CA95035. *G11-p Cell-Based ASIC Products Databook*, second edition edition, July 1998.

[87] P Mandal and V. Visvanathan. CMOS op-amp sizing using a geometric programming formulation. In *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, volume 20, no. 1, pages 22–38, January 2001.

[88] S. Meftali, F. Gharsalli, F. Rousseau, and A. Jerraya. An optimal memory allocation for application-specific multiprocessor system-on-chip. In *International Symposium on System Synthesis*, pages 19–24, 2001.

[89] H. Mehta, R. M. Owens, and M. J. Irwin. Energy characterization based on clustering. In *Proceedings of the 33rd annual conference on Design automation conference*, pages 702–707. ACM Press, 1996.

[90] A. Miller. The convergence of Efroymson's stepwise regression algorithm. In *The American statistician, no. 50*, pages 180–181, 1996.

[91] Model Technology. *ModelSim SE User's Guide*, version 5.5e edition, 2001.

[92] D. Montgomery and E. Beck. *Introduction to linear regression analysis*. John Wiley & Sons Inc., 1982.

[93] A. K. Murugavel, N. Ranganathan, R. Chandramouli, and S. Chavali. Least-squares estimation of average power in digital CMOS circuits. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, volume 10, no. 1, pages 55–58, February 2002.

[94] B. Nadel. The green machine. *PC Magazine*, 1993.

[95] W. Nebel and J. Mermet, editors. *Low Power Design in Deep Submicron Electronics*. Kluwer Academic Publishers, 1997.

[96] K. Nose and T. Sakurai. Analysis and future trend of short-circuit power. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 19, no. 9, September 2000.

[97] K. Ogawa, M. Kohno, and F. Kitomura. Pastel: A parameterized memory characterization system. In *Design, Automation and Test in Europe (DATE)*, pages 15–20, Paris, 1998. IEEE Computer Society.

[98] P. Panda, N. Dutt, and A. Nicolau. *Memory Issues in Embedded Systems-On-Chip*. Kluwer Academic Publishers, 1999.

[99] P. R. Panda. Low-power memory mapping through reducing address bus activity. In *IEEE Transactions on Very Large Scale (VLSI) Systems*, volume 7, no. 3, pages 309–319, 1999.

[100] P. R. Panda. Memory bank customization and assignment in behavioral synthesis. In *International Conference on Computer-Aided Design (ICCAD)*, pages 477–481, 1999.

[101] P. R. Panda, F. Catthor, N.D. Dutt, K. Danckaert, E. Brockmeyer, C. Kullkarni, A. Vandercapelle, and P. G. Kjeldersberg. Data and memory optimization techniques for embedded systems. In *ACM Transactions on Design Automation od Electronic Systems*, volume 6, no. 2, pages 149–206, April 2001.

[102] A. Papoulis. *Probability and Random Variables and Stochastic Processes*. McGraw-Hill, 1991.

[103] P. Petrov and A. Orailoglu. Data cache energy minimizations through programmable tag size matching to the applications. In *Proceedings of the international symposium on Systems synthesis*, pages 113–117. ACM Press, 2001.

[104] P. Petrov and A. Orailoglu. Performance and power effectiveness in embedded processors – customizable partitioned caches. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 20, no. 11, pages 1309–1318, November 2001.

[105] M. Pilsl and B. Rohfleisch. Embedded DRAMs for hard disk drive controllers. In *DAC Tutorial: Embedded Memories in System Design*. Infineon Computer Peripherals Engeneering, 1999.

[106] P. Pirsch. *Architectures for Digital Signal Processing*. John Wiley & Sons, 1998.

[107] F. Poppen and W. Nebel. Comparison of a RT and behavioral level design entry regarding power. In *SNUG Europe, 2001*, 2001.

[108] F. Poppen and W. Nebel. Evaluation of a behavioral level low power design flow based on a design case. In *SNUG Boston, 2001*, 2001.

[109] A. Pratsch. Entwicklung einer Benutzungsoberfläche für ein Low-Power Framework in Tcl/Tk. Master's thesis, University of Oldenburg, 2000. in German.

[110] B. Prince. *Semiconductor Memories, a Handbook of Design, Manufacture and Application*. John Wiley & Sons, Inc., 1991.

[111] B. Prince. *High Performance Memories, New Architecture DRAMs and SRAMs evolution and function*. John Wiley & Sons, Inc., 1994.

[112] J. Rabaey, L. Guerra, and R. Mehra. Design guidance in the power dimension. In *Proc. of the ICASSP*, 1995.

[113] J. M. Rabaey. *SPICE User Manuals*. Univ. of California, Berkeley. `http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/`.

[114] A. Raghunathan, N. K. Jham, and S. Dey. *High-Level Power Analysis and Optimization*. Kluwer Academic Publishers, 1998.

[115] L. Ramachandran, D. Gajski, and V. Chaiyakul. An algorithm for array variable clustering. In *Proceedings of the IEEE European Conference on Design Automation (EURO-DAC'93)*, 1993.

[116] A. Robbins. GAWK: Effective AWK programming. `http://www.gnu.org/manual/gawk-3.1.0/html\protect\unhbox\voidb@x\kern.06em\vbox{\hrulewidth.3em}mono/gawk.html`, 2001.

[117] W. Roethig. Coherent functional, electrical and physical modeling of ip blocks using alf. Technical report, Invited Tutorial CICC'01 2001.

[118] W. Roethig and J. Daniels. A standard for an advanced library format (ALF) describing integrated circuit technology, cells and blocks. Technical Report IEEE 1603 Draft 4, IEEE Standards Department, April 2002. Unapproved Draft Standard.

[119] R Rovatti, M. Borgatti, and R. Guerrieri. A geometric approach to maximum-speed n-dimensional continuous linear interpolation in rectangular grids. In *IEEE Transactions on Computers*, volume 47, no. 8, pages 894–899, august 1998.

[120] K. Roy and S. C. Prasad. *Low-Power CMOS VLSI Circuit Design*. John Wiley & Sons, Inc., 2000.

[121] E. Schmidt. Analysis and modeling of memories on integrated circuits. Master's thesis, Carl-von-Ossietzky University Oldenburg, April 1998. Diplomarbeit at Philips Research Einhoven, Confidential.

[122] E. Schmidt and K. Kopperschmidt. *ORINOCO-BEACH User Guide*, version 2.0 edition, 2001.

[123] E. Schmidt, L. Kruse, G. Jochens, E. Huijbregts, W. Nieuweboer, and W. Seelen, E.and Nebel. Power consumption of on-chip ROMs: Analysis and modelling. In *Int. Works. Power Timing Modelling Performance Integrated Circuits (PATMOS)*, pages 171–180, Lyngby, Denmark, 1998.

[124] E. Schmidt, A. Schulz, L. Kruse, G. von Cölln, and W. Nebel. Automatic generation of complexity functions for high-level power analysis. In *Int. Works. Power Timing Modelling Performance Integrated Circuits (PATMOS)*, 2001.

[125] H. Schmidt and D. Thomas. Synthesis of application-specific memory designs. In *IEEE Transactions on Very Large Scale Integration*, volume 5, no. 1, pages 101–111, 1997.

[126] C. Schurgers, F. Catthor, and M. Engels. Energy efficient data transfer and storage organization for a MAP turbo decoder module. In *Proceedings of the International Symposium on on Low Power Electronics and Design (ISLPED)*, pages 76–81, 1999.

[127] C. Schurgers, F. Catthor, and M. Engels. Memory optimization of MAP turbo decoder algorithms. In *IEEE Transactions on Very Large Scale (VLSI) Systems*, volume 9, no. 2, pages 305–312, April 2001.

[128] W.-T. Shiue and C. Chakrabarti. Memory exploration for low power, embedded systems. In *Proceedings of the 36th ACM/IEEE conference on Design automation conference*, pages 140–145. ACM Press, 1999.

[129] W.-T. Shiue, S. Udayanarayanan, and C. Chakrabarti. Data memory design and exploration for low-power embedded systems. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 6(4):553–568, 2001.

[130] Philips ED&T/Analogue Simulation. *Pstar User Manuals*, third edition, 1995.

[131] G. Smith, D. Nadamuni, L. Balch, N. Wu, and J. Tully. *EDA 2001: Its' good To be an EDA Vendor - Market Trends*, October 2001. Gartner Report no. SWTA-WW-MT-0102.

[132] A. Stammermann, L. Kruse, W. Nebel, A. Pratsch, E. Schmidt, M. Schulte, and A. Schulz. System level optimization and design space exploration for low power. In *ISSS 2001*, 2001.

[133] L. Stok and J. A. G. Jess. Foreground memory management in datapath synthesis. In *International Journal on Circuit Theory and Applications*, volume 20, no. 3, pages 235–255, 1992.

[134] C.-L. Su and A. M. Despain. Cache design trade-offs for power and performance optimization: a case study. In *Proceedings 1995 international symposium on Low power electronics and design*, pages 63–68. ACM Press, 1995.

[135] Synopsys, Inc. *PowerMill Datasheet*. `http://www.synopsys.com/products/etg/powermill\protect\unhbox\voidb@x\kern.06em\vbox{\hrulewidth.3em}ds.html`.

[136] Synopsys, Inc., 700 East Middlefield Road, Mountain View, CA. *Design Compiler Reference Manual*, v2001.08 edition, August 2001.

[137] Synopsys, Inc., 700 East Middlefield Road, Mountain View, CA. *Design Compiler User Guide*, v2001.08 edition, August 2001.

[138] Synopsys, Inc., 700 East Middlefield Road, Mountain View, CA. *DesignWare Foundation Library Databook*, release2001.02 edition, February 2001.

[139] T. Takahashi, T. Sekiguchi, R. Takemura, S. Narui, H. Fujisawa, S. Miyatake, M. Morino, K. Arai, S. Yamada, S. Shukuri, M. Nakamura, Y. Tadaki, K. Kajigaya, K. Kimura, and K. Itoh. A multigigbit DRAM technology with $6f^2$ open-bitline cell, distrubuted overdriven sensing, and stacked-flash fuse. In *IEEE Journal of Solid-State Circuits*, volume 36, no. 11, November 2001.

[140] S. Tomishima, T. Tsuji, T. Kawaski, M. Ishikawa, T. Inokuchi, H. Kato, H. Tamizaki, W. Abe, A. Shibayama, Y. Fukushima, M. Niiro, M Maruta, T. Uchikoba, M. Senoh, S. Sakamoto, T. Ooishi, H. Kikukawa, H. Hidaka, and K Takahahi. A 1.0-V 230-MHz column access embedded DRAM for portable MPEG applications. In *IEEE Journal of Solid-State Circuits*, volume 36, no. 11, pages 1721–1727, November 2001.

[141] A. Turier, L. Ben Ammar, and A. Amara. An accurate power and timing modelling technique applied to a low-power ROM compiler. In *Int. Works. Power Timing Modelling Performance Integrated Circuits (PATMOS)*, pages 181–190, Lyngby, Denmark, 1998.

[142] A. Vandecapelle, M. Miranda, E. B. F. Catthhoor, and D. Verkest. Global multimedia system design exploration using accurate memory organization feedback. In *Proceedings of the 36th Conference on Design Automation*, 1999.

[143] G. von Cölln. *Modellierung und Simulation der Verlustleistung von integrierten Schaltungs-Makros.* PhD thesis, Univ. Oldenburg, Germany, 2001. in German.

[144] L. Wall, T. Christiansen, and J. Orwant. *Programming Perl.* O'Reilly, 2000.

[145] C.-C. Wang, C. F. Wu, R.-T. Hwang, and C.-H. Kao. Single-ended SRAM with high-test coverage and short test time. In *IEEE Journal of Solid-State Circuits*, volume 35, no. 1, January 2000.

[146] J.-S. Wang, W. Tseng, and H.-Y. Li. Low-power embedded SRAM with current-mode write technique. In *IEEE Journal of Solid-State Circuits*, volume 35, no. 1, January 2000.

[147] N. Wehn and S. Hein. Embedded DRAM architectural trade-offs. In *Proceedings of the Design, Automation and Test in Europe 1998 conference on Design, automation and test in Europe*, pages 704–708. Institution of Electrical Engineers, 1998.

[148] B. Welch. *Practical Programming in Tcl and Tk.* Prentice Hall,Inc, 1997.

[149] B. Wicht, S. Paul, and D. Schmitt-Landsiedel. Analysis and compensation of the bitline multiplexer in SRAM current sense amplifiers. In *IEEE Journal of Solid-State Circuits*, volume 36, no. 11, pages 1745–1755, November 2001.

[150] T. Williams and C. Kelley. *gnuplot - An Interactive Plotting Program*, version 3.7 edition, 1998. http://www.comnets.rwth-aachen.de/doc/gnu/gnuplot37/gnuplot.html.

[151] S. J. E. Wilton and N. P. Jouppi. CACTI: An enhanced cache access and cycle time model. In *IEEE Transactions on solid-state circuits*, volume 35, no. 5, pages 677–687, May 1996.

[152] D. Wismer and R. Chattergy. *Introduction to nonlinear optimization.* Elsevier North Holland Inc., 1979.

[153] A. Worm, H. Lamm, and N. Wehn. A high-speed map architecture with optimized memory size and power consumption. In *Proc. SiPS 2000*, pages 265–274, October 2000.

[154] A. Worm, H. Michel, and N. Wehn. Power minimization by optimizing data transfers in turbo-decoders, September 199.

[155] Q. Wu, Q. Qiu, M. Pedram, and C. Ding. Cycle-accurate macro-models for RT-level power analysis. In *IEEE Transaction on VLSI Systems*, volume 6, no. 4, december 1998.

[156] S. Wuytack, F. Catthoor, G. de Jong, and H. J. De Man. Minimizing the required memory bandwidth in VLSI system realizations. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, volume 7, no. 4, December 1999.

[157] S. Wuytack, F. Catthoor, F. Franssen, L. Nachtergaele, and H. De Man. Global communication and memory optimizing transformations for low power systems. In *IWLPD'94*, 1994.

[158] Y. Zhang, X. Hu, and D. Chen. Global register allocation for minimizing energy consumption. In *Proceedings of the International Symposium on on Low Power Electronics and Design (ISLPED)*, pages 100–102, 1999.

[159] Y. Zorian, editor. *IEEE Design&Test of Computers*, May-June 2001.

[160] V. Zyuban and P. Kogge. The energy complexity of register files. In *Proceedings 1998 international symposium on Low power electronics and design*, pages 305–310. ACM Press, 1998.

I herewith declare to have written this thesis only based on the sources listed and without the help of others. I have not submitted or prepared the submission of this or any other doctoral thesis at the Carl von Ossietkzy University Oldenburg or any other university.

Hiermit erkläre ich, diese Arbeit ohne fremde Hilfe und nur unter Verwendung der angegebenen Quellen verfasst zu haben. Ich habe bis dato weder an der Carl von Ossietzky Universität Oldenburg noch an einer anderen Universität die Eröffnung eines Promotionsverfahrens beantragt oder anderweitig eine Promotion vorbereitet.

Curriculum Vitae

1973	Born in Bremerhaven
1979 - 1982	Grundschule Horstedt Elemenatry school
1982 - 1985	Freie Rudolf-Steiner-Schule in Ottersberg
1985 - 1992	Ratsgymnasium in Rotenburg an der Wümme
07/1992 - 09/1993	Military training
10/1993 - 06/1998	Study of 'Allgemeine Informatik' General computer science at the Carl von Ossietzky Universität Oldenburg
11/1997- 30/1998	Diplomarbeit (Master thesis) at "Philips Natuurkunding Laboratorium", Netherlands
6/1998	Diplom in Informatik (MSc in Computer Science)
7/1998 - 11/01	Research assistant at Kuratorium OFFIS e.V.
12/01 - date	Manager System Analysis and Optimisation Group Kuratorium OFFIS e.V.