CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Power Estimation for DSP Components for Fiber-Optic Communication Systems

Master's thesis in Embedded Electronic System Design

Lorenzo Chelini

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2017

Master's thesis 2017

LORENZO CHELINI

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology | University of Gothenburg
Gothenburg, Sweden 2017

Power Estimation for DSP Components for Fiber-Optic Communication Systems
LORENZO CHELINI

Supervisor: Per Larsson-Edefors, Department of Computer Science and Engineering
Examiner: Lena Peterson, Department of Computer Science and Engineering

Master's Thesis 2017
Department of Computer Science and Engineering
Chalmers University of Technology | University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2017

Power Estimation for DSP Components for Fiber-Optic Communication Systems
LORENZO CHELINI
Department of Computer Science and Engineering
Chalmers University of Technology | University of Gothenburg

Abstract

In this thesis, we propose and examine a power estimation method for Finite Impulse Response (FIR) filters used in short-reach fiber-optic communication systems, where limitations on size and power consumption start to become a critical design metric. We first implement a library of optimized FIR filter netlists that can satisfy the high performance required for optical communication. Second, we analyze the FIR filter implementations and we propose a new activity-based macromodeling strategy. More precisely, the model makes use of the switching activity of the logic inputs to estimate the power consumption at architectural level. Following the taxonomy available in the literature, our estimator tool can be classified as a cycle-accurate macromodel and it consists of two phases: characterization and power estimation. The characterization is a fully automated procedure that takes as input a netlist and generates power values using gate-level simulations. Those values are classified accordingly to the Hamming distance of each input pair and tabulate in pre-defined structures. Characterization needs to be done only once for each filter architecture at a reasonably strict timing constraint. The power estimation takes as inputs the pre-defined tables, input trace, and produces power values in a cycle-by-cycle manner. The average power is computed summing up the per-cycle power values and divide the sum by the duration of the input trace expressed in clock cycles. A key strength of our proposed solution is that it provides results within seconds since we do not need to perform additional simulations during power estimation. For the operand word-lengths tested the error was below 15 percent for most of the circuits. The only exception was for the 8-bit input and 6-bit coefficient and the 8-bit input and 10-bit coefficient where the error was above 15 percent. The results are promising and demonstrate the practicability and validity of this approach.

Acknowledgements

I would like to thank my supervisor Per Larsson-Edefors. He has always been very present and he supported me with his valuable advice. A special thank to Lars Lundberg for providing me the MATLAB script to test my model in real case scenarios. Thanks to Christoffer Fougstedt for valuable discussions and the parallel filter provided.

Lorenzo Chelini, Gothenburg, June 2017

Contents

List of Figures

List of Tables

Acronyms

ASIC	Application specific integrated circuit
ALU	arithmetic logic unit
CMOS	Complementary metal oxide semiconductor
CPU	Control processing unit
DSP	Digital signal processor
EDA	Electronic design automation
FIR	Finite impulse response
FPGA	Field-Programmable Gate Array
FPU	Floating-point unit
IC	Integrated circuit
LUT	Look-up table
MAC	Multiplier–accumulator
NMOS	N-type metal-oxide-semiconductor
PMOS	P-type metal-oxide-semiconductor
SoC	System-on-a-Chip
RTL	Register transfer level
SAIF	Switching activity interchange format
SDF	Standard delay format
STA	Static timing analysis
TCF	Toggle count format
VCD	Value change dump
VLSI	Very-large-scale integration

1

Introduction

Today more than ever, power consumption is one of the major metrics in the VLSI design process, equaling area and performance [1]. In the deep-submicron era, the power density of modern microprocessors has risen above 100 W/cm^2 [2] leading to elevated junctions temperatures that reduce reliability and performance of the devices. Fans and heat sinks can be used to get rid of all this heat but they are expensive and prone to malfunctions [3]. Therefore, nowadays having power estimation in the early stage of the design is a critical issue, not only for the allocation of the power budget but also for reducing the time to market; it is unfeasible to synthesize different design architectures down to the gate level in a reasonable time and then choose the one that meets our design constraints in terms of power, area and performance. As a consequence power estimation tools are necessary today in order to give the designers the possibility to widely explore the architectural design space and to re-target architectural choices early on. This thesis will propose and examine a power estimation method for Finite Impulse Response (FIR) filters used in fiber-optic communication receivers. Since limitations on size and power consumption start to become a critical design metric for receivers that work in short-reach systems [4], [5].

1.1 Context and Motivation

Although a broad body of research has been carried out on high-level power estimation, to the best of our knowledge, it is not possible to extract accurate power estimation directly from mathematical modeling tools like MATLAB. Cardarilli et al. [6] developed an algorithm for power evaluation of digital filters implemented on FPGA taking into account the current drawn by each filter architecture. Their solution provides power evaluation considering random-generated input vectors. However, when dealing with applications for optical communications, properties of the input signal can be exploited to obtain more accurate power estimations. In addition, methodologies that based their characterization on completely random vectors give erroneous estimations when real traces are considered [7].

1.2 Goals and Challenges

Nowadays, with increasing system complexity, obtaining accurate power estimations at micro-architectural level is becoming more and more complex. Power consumption depends not only on different parameters such as switching activity and input correlations, but it also widely depends on cell sizing. Thus it is more difficult to estimate power consumption before the design has been refined down to gate level by the synthesis tool. Moreover, due to technology scaling, delay and capacitance introduced by wiring are becoming one of the main mechanisms of power consumption in ASICs [8]. Another aspect that should be taken into account while dealing with power estimation is that the average power is strongly related to the input patterns, such as average probability and transition density probability. Moreover, the relation between these two metrics (power and input patterns) has been proven to be nonlinear [9]. Hence it is difficult, in most of the cases, to find a mathematical model, leading us to think that a method based on tables could be a practical solution [9].

Table-based models have been extensively studied and proposed in recent years, however when dealing with these methods, it is essential to understand which metrics capture the features that are mainly responsible for the power consumption within the design [10].

The aim of this master thesis is to develop a power estimation methodology targeting:

- Run-time efficiency in the evaluation, otherwise we could just rely on low-level tools for power characterization.
- Relative accuracy, meaning that the algorithm should provide accurate estimations even if the environment of the circuit is not completely defined. Notice that with the word “accuracy” we address the relative accuracy that in high level power estimation tools is considered to be more important than the absolute accuracy, the main goal being the comparison among different design choices [11]. We are targeting an accuracy of 25 percent from the power value obtained at gate level.
- Automation, meaning that the estimation flow should be automated, with no user intervention.

1.3 Assumptions and Limitations

We rely on the fact that the user explicitly indicates which filter is assumed, in terms of e.g. taps and resolution, at the algorithmic level. We restrict ourselves to a set of filter architectures; specifically we will focus on a transposed implementation of an FIR filter.

1.4 Ethics

The power values, gathered during the different synthesis, will be obtained in full compliance with the aim of research such as clarity, validity and avoidance of errors. The methodology used to collect these data will be clearly explained.

1.5 Thesis Organization

The report is organized as follows:

Chapter 2 Background: in chapter two we introduce the reader to different methods for power estimation at RT level. We mainly focus on top-down and bottom-up approaches.

Chapter 3 Power Dissipation in CMOS: in chapter three we detail all the mechanisms of power consumption in CMOS technology.

Chapter 4 Filter Design: in chapter four we describe how the filters have been implemented in VHDL.

Chapter 5 RT Level Power Estimation: in chapter five the core of our work is presented. We first briefly discuss the recommended flow for power estimation suggested by the EDA vendor. Then, we propose our approach to power estimation.

Chapter 6 Results and discussion: in chapter six we present our results as well as highlight the pros and cons of our approach.

Chapter 7 Conclusion: Finally, in chapter seven we summarize the achievements of our work and we give recommendations for future work.

2

Background

The purpose of this chapter is to introduce the readers to the concept of power estimation and to present different methods that have recently been proposed in the literature. First, a brief overview of the complexity of present day digital designs will be presented. Subsequently, different flows for power estimation at different architectural levels will be reviewed, comparing accuracy and computational complexity. Finally, the concept of register transfer (RT) level power estimation will be introduced in addition to different categories of macromodel techniques.

2.1 Complexity of Electronic Devices

The continuing decrease in feature size for very large integrated circuits (VLSI) has resulted in an unprecedented level of integration. While the end user can benefit from such a high level of integration, having more functionality embedded on the same die, the systems designers have to face a growing design complexity. In order to cope with that different techniques have been introduced: system partitioning and design reuse.

2.2 Design Partitioning and Design Reuse

Design partitioning relies on the idea that a problem can be always decomposed into smaller sub-parts until these sub-parts become manageable in term of size and complexity [12]. Once these sub-systems have been identified, they are recursively solved and their solutions combined to solve the original problem. When applied to VLSI, this means that the design is broken down in a hierarchical way into smaller sub-units that can be independently designed. For example, a complex system like a CPU, can be seen as the interaction between two separate components: the datapath and the control unit. The datapath can be further divided into ALU, FPU and so on. Again the ALU, can be further split into different sub-modules like adder, multiplier, divider and registers. This is depicted in Fig. 2.1.

In other words, design partitioning not only allows designers to speed up the design process, but it also helps them to cope with the design complexity, spreading it across different sub-designs. Another method to reduce design complexity leverages

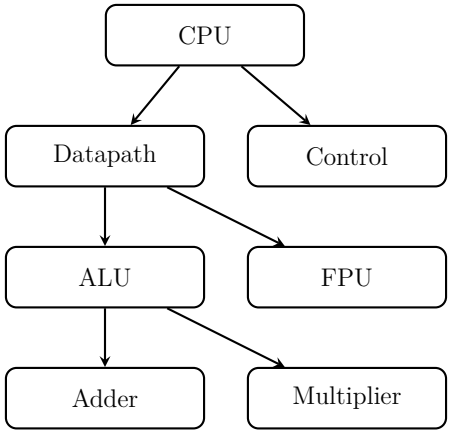

Figure 2.1: Possible decomposition of a CPU design.

on the reuse of pre-designed circuits. The idea behind the use of silicon IP (or cores) is to provide the system on a chip (SoC) designers with a suite of pre-verified blocks that can be integrated onto the chip to realize complex functions [13].

Further steps to increase the design granularity and hence decreasing the overall complexity led to the introduction of the platform-based design (PBD). Carloni and coauthors [14] defined platform-based design as a "meet-in-the-middle process where design specifications meet abstractions of potential design implementations". Conceptually, platform-based design aims to standardize buses or component architectures and reduce the custom interfaces design to contain development risks and gain productivity [15].

2.3 Design Abstraction

The term abstraction refers to the degree of details we are concerned with at a given stage during the design process, specifically, we can identify four main abstraction levels, namely: algorithm, register transfer, gate and transistor [16]. At algorithm level, the digital design is described using high-level programming languages like C, System-C or MATLAB. Thus its functionalities are described as a flow of instructions that is executed in sequence to perform a specific task [16]. At register transfer level, the design is modeled as a flow of signals between physical registers. Finally, at gate level and transistor level the design is specified as a set of geometrical entities, logic gates and transistors respectively.

At each level of abstractions, we can describe the circuit according to different views, in more detail: the behavioral or functional view specifies the circuit's functionality without considering its implementation. Behavioral descriptions are used when we want to visualize abstractly the different interactions among the elements of the circuit. The architectural view describes a circuit as the interconnection of components (e.g., registers, adders, multipliers, ALU). We are looking at the connectivity that exists between the building blocks rather than focusing on the overall functionality. Notice that, given a behavioral description we can always come up with different

structural views for the same network. Finally, the physical view deals with the geometry of a circuit and the physical objects (e.g. transistors).

The intersection between a level of abstraction and a view is the so called circuit model [12]. The three design domains and the different levels of abstraction are depicted in the well know Y-chart, shown in Fig. 2.2.

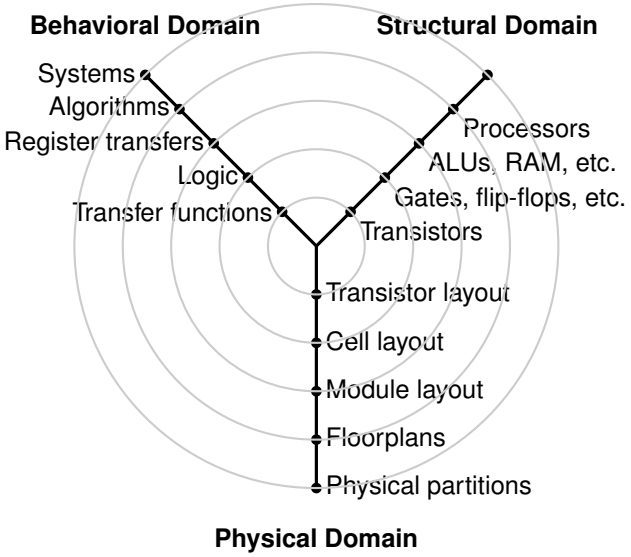

Figure 2.2: Gajski-Kuhn Y-chart [17].

2.4 Power Estimation in the Design Flow

Depending on the level of abstraction and availability of physical information, different estimation methodologies can be used, with corresponding variations in terms of speed and accuracy. At higher level of abstractions such as algorithm level, only spread-sheets are available to obtain coarse estimations. At this earliest stage, the most power-hungry components within the design can be identified. At register transfer level, more details about the design are specified, since it is described as a set of primitives (or macros) such as adders, multipliers, dividers and registers [16]. Still, at this stage, it is not possible to provide accurate power predictions, unless relying on specific primitives architectures. At gate level, not only the macro's implementations are known but also physical information about cells is provided by the IC vendors and EDA tools offer methodologies for wiring estimation. Hence at this stage, the accuracy can be greatly improved. However, even if it is possible to increase the accuracy, due to the short time to market, it is not feasible for today's system designers to widely explore the design space. At even lower level, circuit simulators, such as SPICE, can be used to estimate power; however, these simulators are too slow and the simulation is only feasible for fairly small circuits [18]. Fig. 2.3 shows the different levels of abstraction. It should be clear that higher levels of abstraction offer higher power saving opportunities but a lower accuracy in the power estimation.

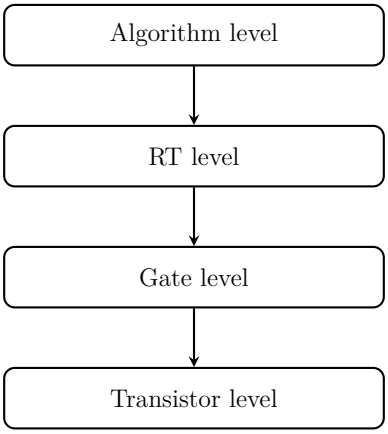

Figure 2.3: Different abstraction levels.

2.5 RT level Power Estimation

RT level power estimation techniques have been extensively studied in the last decade [19]. The main reason can be identified in the need to provide accurate power evaluations (with accuracy comparable to gate level) while maintaining the computational efficiency of the RT level simulations, notably faster than the ones performed at lower levels. In other words, most of these techniques evaluate, based on some metrics available at RT level, the power consumption of a given circuit's architecture and upon this information creates a model. Eventually, the model will be used, later on, to provide fast and accurate power estimation at RT level while evaluating different designs alternatives [10].

RT level power estimation techniques can be broadly classified in two categories: top-down or bottom-up approaches.

2.5.1 Top-down Approach

Top-down methods attempt to estimate the power consumption when the circuit is described only by Boolean equations, without having any information on its gate-level implementation [9]. These methods are useful when an implementation is not yet available to provide a rough measurement for power consumption. However, their usage is quite limited due to lack of implementation details for the target circuit that leads to a low accuracy in the estimation process [20]. Top-down approaches can be further classified as complexity based and information based.

Information-based Macromodeling

Information-based macromodels are based on the concept of entropy. If x is a random Boolean variable having probability P of being high, then the entropy is defined as following:

$$\mathrm{H}(x) = \mathrm{P}\log_2\frac{1}{\mathrm{P}} + (1-\mathrm{P})\log_2\frac{1}{(1-\mathrm{P})}. \tag{2.1}$$

Intuitively, the entropy can be seen as the amount of information carried by a signal. Najm and Nemani [21], proposed a technique to estimate the average dynamic power consumption in a combinational circuit, given only the Boolean representation. In the paper, they stated that the average power consumed by a circuit can be approximated with:

$$P_{avg} = \frac{1}{2}\sum_{i=1}^{N} C_i D(X_i) \tag{2.2}$$

where the unknowns are the node capacitance C_i and the transition density of node X_i defined as $D(X_i)$. By doing some approximations and simplifications, due to the fact that the internal representation of the circuit is not known, 2.2 can be approximated as:

$$P_{avg} \approx D\sum_{i=1}^{N} C_i \tag{2.3}$$

where $\sum_{i=1}^{N} C_i$ is an estimation of the total circuit capacitance and D is the average node transition density. The former can be evaluated directly from the complexity of the Boolean function [22]. The latter can be estimated using the following formula [21], under the assumption that the average transition density is proportional to the average entropy

$$h_{avg} \approx \frac{2}{3(n+m)}(h_{in} + 2h_{out}). \tag{2.4}$$

h_{in} and h_{out} are the entropy values computed on the input and on the output respectively; while n and m are the numbers of input and output bits respectively. Recently Dhaou et al. [23] proposed a new technique for low-power characterization still based on entropy as information measurement. However in contrast to previous techniques, they relied also on gate level simulations to achieve better accuracy. The authors claim to obtain an average error less than 16 percent compared to SPICE simulations.

Complexity-based Macromodeling

Complexity-based macromodels bind the design power consumption with some metrics that approximate the circuit complexity. Example of such metrics are: the number of cubes or the cube's complexity within the Boolean function, that describes the circuit functionality. Nemani and Najm [24] address the problem of estimating

the average area and, hence, the power consumption for single-output Boolean function, considering the average cube complexity. They stated that the average power consumption is proportional to

$$P_{avg} \approx D_{avg} A C_{avg} \tag{2.5}$$

where D_{avg} is the average node switching activity, A is the area estimation and C_{avg} the average node capacitance considering a specific gate library. The work has lately been extended to include multiple-output Boolean functions [25]. Other methods that fall within this category are based on the concept of equivalent gates. Following this definition the power dissipated by a given block can be roughly approximated as the number of equivalent gates multiplied by the power consumption of a single instance, representing the reference gate.

2.5.2 Bottom-up Approach

The idea behind the bottom-up approach is to construct a model based on existing implementations. Power values for a given circuit are extracted using extensive gate-level (or transistor-level) simulations. Subsequently, these values are used to build up the macromodel that can later be used for high-level power estimations. A macromodel can thus be defined as a function that binds the power consumed (the dependent variable) to a set of metrics (the independent variables). More formally:

$$Pow = f(X_1, X_2, ..., X_n). \tag{2.6}$$

It is worth to mention already at this point that one of most challenging aspects, while constructing a macromodel, is the choice of the metrics X_i [26].

Bottom-up techniques consist of two main stages: *characterization* and *estimation*. Characterization is the process by which the macromodel is trained to improve the accuracy. The training process could be based on gate-level or transistor-level simulations repeated for a given number of input vectors. Since the characterization process is done only once for each of the components within the library, the computational complexity and the execution time is allowed to be high. The tricky part during this process is to find out which kind of metrics should be extracted from each simulation and used for modeling the power consumption. Once the metrics have been decided and the library components characterized, power estimation can be performed. The power estimation algorithm reads the input trace, which the users should provide, and performs power evaluations. In the estimation a trade-off should be made between computational complexity and accuracy. To reduce the former, it is desirable to have a compact model; however, having few metrics reduces the latter.

Table-based Macromodeling

Power estimation in table-based macromodels consists of two main steps: table construction (during characterization) and table look-up (during estimation). Table construction can be further divided into two parts: metrics extraction and table filling. The former tries to find out some input characteristics (or metrics) that are mainly responsible for the power consumption within the design. The latter uses gate-level or transistor-level simulations to map the power values with a given set of input characteristics. An example of table construction is depicted in Fig. 2.4.

Once the table has been filled a look-up procedure can be used for accessing an entry within the table. An example of table look-up with a given set of input metrics is depicted in Fig. 2.5.

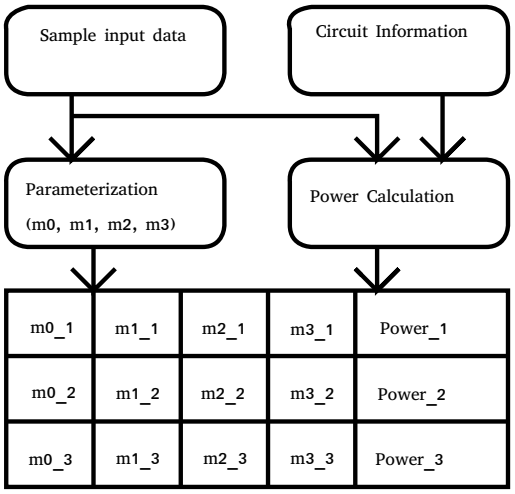

Figure 2.4: Table construction. Four metrics m_0, m_1, m_2 and m_3 are extracted form the input data. For each set of those selected metrics a power value is assigned averaging the power values obtained from gate-level simulations.

Table-based power macromodels have been around for quite long time; one of the first method introduced, was the three-dimensional table [9]. The three-dimensional-table takes into account the input switching activity in order to estimate the power consumption in a combinational logic circuit. Specifically, let P_{in} be defined as the average fraction of clock cycles when the signal is high [19]

$$P_{in} = \frac{1}{N_{in}} \sum_{i=1}^{N_{in}} \mathrm{P}(in_i(t) = 1).^{1} \tag{2.7}$$

Let D_{in} be the average fraction of clock cycles where the signal commutes from high to low or from low to high [19]

[1]Notice that in_i in the above formula, is the pin with index i on the input port while N_{in} is the primary input word-length

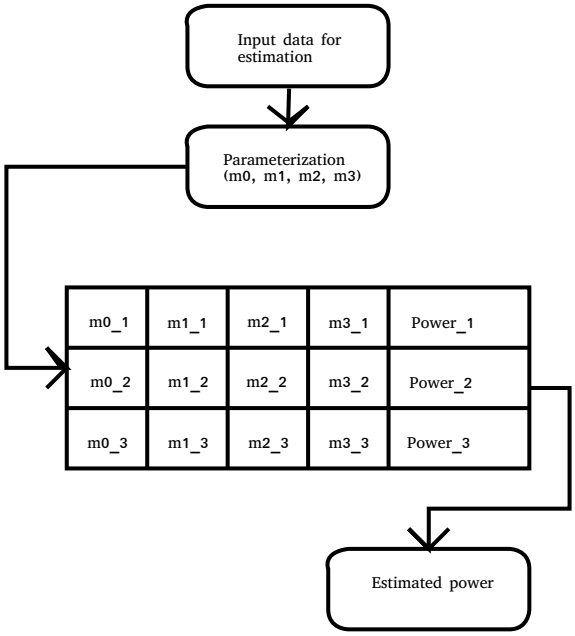

Figure 2.5: Table look-up. The metrics selected during characterization m_0, m_1, m_2 and m_3 are extracted from the input data and used to index the table.

$$D_{in} = \frac{1}{N_{in}} \sum_{i=1}^{N_{in}} \mathrm{P}(in_i(t) \neq in_i(t^+).^2 \tag{2.8}$$

Finally, let D_{out} be defined as the average fraction of the clock cycles where the output signal commutes from high to low or from low to high [19]

$$D_{out} = \frac{1}{N_{out}} \sum_{i=1}^{N_{out}} \mathrm{P}(out_i(t) \neq out_i(t^+).^3 \tag{2.9}$$

Then, the model postulates the following:

$$P_{avg} = f(P_{in}, D_{in}, D_{out}). \tag{2.10}$$

The idea behind this method is to partition P_{in} and D_{in} into a set of probability intervals; for each of those intervals it generates a large number of input vectors with the same average P_{in} and D_{in} probabilities. The authors refer to these assignments on the primary input as P and D vectors. For each given pair of P and D vectors the power is computed using Monte Carlo simulations and D_{out} obtained considering the switching activity on the output. The set P_{in}, D_{in} and D_{out} will identify a particular location in the three-dimensional table and the content of this location will be populated averaging the power values associated with the set. Once the table

[2]see footnote one

[3]Notice that out_i in the formula, is the pin with index i on the output port while N_{out} is the output word-length.

has been built the power estimation can be obtained using a simple table look up procedure.

The aforementioned method has been the subject of different subsequent papers. Barocci et al. [10], improved the method adding an adaptive algorithm for the generation of the P and D vectors in order to cope with statistical fluctuations that may have led to erroneous storage in the table. In addition they provided a two-step interpolation procedure that was only mentioned, as possible solution, in the original paper.

Concurrently to the work of Barocci et al., Gupta and Najm, introduced a fourth dimension, extending the three-dimensional table model, to account for spatial and temporal correlation among the input vectors. This fourth dimension is called SC_{in} and is defined as the probability for both input to be high at the same moment:

$$SC_{ij} = \mathrm{P}(\mathbf{x}_i \wedge \mathbf{x}_j). \tag{2.11}$$

This additional variable has been proven to be effective for capturing the behavior of circuits like adder; where spatial and temporal correlation should be taken into account to record the carry rippling [3].

Equation-based Macromodeling

Table-based macromodels are generally quite accurate, but they require a long characterization time and a huge memory resources for storing the entire table. The research for faster methods to solve the problem of estimating the power consumption led to the equation-base macromodels. As the name suggest those methods approximate the average power consumption for a given circuit instance using a general polynomial. For efficiency reasons the lowest order polynomial that gives an acceptable result is used. One of the first approaches proposed by Gupta and Najm [1] approximates the average power using a linear function, as follows:

$$P_{avg} = c_0 + c_1 P_{in} + c_2 D_{in} + c_3 SC_{in} + c_4 D_{out} \tag{2.12}$$

where the regression coefficients c_i are computed generating different test patterns for different values of P_{in}, D_{in} and SC_{in} trying to cover a wide range of input probabilities. Subsequently, for each trace, Monte Carlo simulations are used to extract the output transition probability D_{out} and to compute the average power consumption. The accuracy was evaluated using the ISCAS-85 benchmark. Unfortunately, large errors were obtained for most of the circuits. For this reason, in the same paper, the authors proposed a quadratic and even a cubic function to improve the accuracy. The quadratic equation is reported below:

$$P_{avg} = c_0 + c_1 P_{in} + c_2 D_{in} + c_3 SC_{in} + c_4 D_{out} + c_5 P_{in} D_{in} + c_6 P_{in} SC_{in} c_7 P_{in} D_{out} + c_8 D_{in} SC_{in} + c_9 D_{in} D_{out} + c_{10} SC_{in} D_{out} + c_{11} P_{in}^2 + c_{12} D_{in}^2 + c_{13} SC_{in}^2 + c_1 4 D_{out}^2. \tag{2.13}$$

The regression coefficients are computed in the same way as before both for the quadratic and cubic form. The accuracy is greatly improved especially in the cubic while in the quadratic still an average error above 15 percent is achieved in the c499 and c1355 circuits.

Cycle-accurate Macromodeling

Cycle-accurate power macromodeling can be utilized to obtain accurate average power estimations while performing functional RT level simulations. Incidentally, the same approach can be applied to compute instantaneous and peak current absorbed by a circuit that is known as "transient power analysis". Transient power analysis are useful to investigate IR-drop problems that lead to reduced performance as a consequence of a reduced supply voltage [27]. Since our model is based on this approach a more in-depth description of the idea behind these methodologies will be presented.

Consider a combinational or sequential circuit with n inputs $\mathbf{x} = [x_1, x_2, .., x_n]$ and m output $\mathbf{y} = [y_1, y_2, .., y_m]$. Suppose a single transition on the primary input from $\mathbf{x}_i$ to $\mathbf{x}_f$ occurs in a time interval t, an amount of power (hereafter referenced as $e(\mathbf{x}_i, \mathbf{x}_f)$) will be consumed by the circuit. The goal is to find a model of $e(\mathbf{x}_i, \mathbf{x}_f)$ considering only boundary information, such as Hamming distance. Said in other words, the macromodeling task is to find a function that associates any input transitions $(\mathbf{x}_i, \mathbf{x}_f)$ with the power consumption $e(\mathbf{x}_i, \mathbf{x}_f)$. The straightforward way to do so is to associate to any transitions the respective value obtained from gate- or transistor-level simulations. Unfortunately for a module with n-bit input, this requires to store 4^n values, hence it is only possible for small circuits.

One of the first method introduced was the clustering algorithm [28]. Starting from the observation that a fully characterized energy transition matrix is not feasible to store due the exponential growth of the table with the number of inputs; the clustering algorithm attempts to compress this matrix collapsing closely bit patterns, thereby reducing the table's size. In particular, a circuit with n-bit input is simulated using m random test vectors with $m \ll 2^n$, then clustering is performed on the simulated vectors. If the coverage is sufficient the algorithm stops, otherwise additional vectors are fed in input to the circuit. The flow is depicted in Fig. 2.6.

In the paper the author presents three different clustering algorithms along with a mathematical proof for the coverage sufficiency based on probabilistic analysis and confidence intervals. The first algorithm introduced (the exact algorithm) is similar to the Quine-McCluskey method used for logic minimization; neighbor patterns are merged together to form a cluster and the algorithm iterates until no further merging

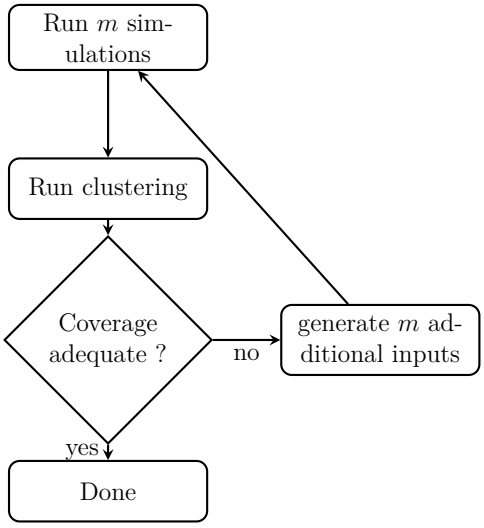

Figure 2.6: Flow chart for the clustering algorithm.

operations are possible. Nevertheless, the algorithm complexity grows exponentially with the number of inputs, hence it is feasible only for fairly small circuits.

The other algorithms proposed (Cluster 2 and Cluster 3) run in polynomial time and are based on heuristics such as the "expand" used in Espresso logic minimizer. Although promising for their low computational complexity, the clustering algorithms are based on an erroneous assumption. They assume that neighbor input patterns have close power values, but for many circuit this is not always the case. Consider, for instance the carry propagation within a ripple-carry adder. Here, vectors that differ even for one bit may generate different power values due to the propagation of the carry along the chain of full adders [29].

Benini et al. [29] proposed to represent the power consumption at each clock cycle for a given circuit as a linear function of the activity on the primary input and on the primary output. More precisely, considering a circuit with n input and m output, they stated the model as follows:

$$P(j) = c_0 + c_1 a_1 + .. + c_n a_n + c_{n+1} a_{n+1} + .. + c_{n+m} a_{n+m} \tag{2.14}$$

where the index j represents the current iteration, $\mathbf{c} = [c_0, c_1, .., c_{n+m}]$ is the coefficients vector and $\mathbf{a} = [a_0, a_1, ..a_{n+m}]$ is the variables vector. Each variable is defined as the bitwise XOR operation between two consecutive input patterns. The least mean square algorithm is used to iteratively update the coefficients trying to minimize the least square error produced by the model, specifically:

$$\mathbf{c}_{j+1} = \mathbf{c}_j + 2\mu\epsilon_j \mathbf{a}_j \tag{2.15}$$

where j is the current iteration, $\mathbf{c}_j$ the coefficients vector, μ is a constant that controls the convergence of the algorithm, $\mathbf{a}_j$ the bitwise XOR between the current and the previous input vectors and ϵ_k is the difference between the estimated power and the actual one.

3

Power Dissipation in CMOS

In this chapter we will briefly go through the main mechanisms of power consumption in CMOS technology. This is done to introduce the reader with basic concepts that will be used later on in chapter 5.

3.1 Mechanisms of Power Dissipation in CMOS

The power consumption of CMOS technology consists of four main components: switching power, short-circuit power, subthreshold leakage power and gate-leakage power. Of the four, the first two constitute the *dynamic* power, while the last two form the *static* power. Thus the total power dissipated by a CMOS implementation can be written as the sum of two terms:

$$P_{total} = P_{static} + P_{dynamic}. \tag{3.1}$$

Until very recently, *dynamic* power, which arises from the charging and discharging of the internal capacitance, has been the subject for almost all the power estimation techniques. This is because it has been the main mechanism of power consumption, even if Moore's law has helped to limit it. Nevertheless, for today's technology this is no longer true. *Static* power due to off-state current that leaks through the transistors starts to dominate the power consumption. Andrew Grove, chairman and CEO of Intel, in 2002, during the International Electron Devices Meeting stated that off-state current will become one of the limiting factor in future integration of modern microprocessors [30]. In the following sections we will briefly describe and will reason about the four main components that constitute the *dynamic* and *static* power. Glitching power will also be described in a separate section.

3.1.1 Switching Power

Switching power is the power dissipated by the circuit due to charging and discharging of the internal nodes capacitance. An expression for the switching power can be easily derived considering a simple CMOS inverter driving a load with a capacitance that equals C. Specifically, suppose the circuit is in a steady state with a logic one as input and a logic zero as output. Initially the capacitor is discharged

to zero. If an instantaneous high-to-low transition occurs on the inverter input, the load capacitance will be charged by the PMOS transistor and its voltage will rise from 0 to V_{DD}. The equivalent circuit is depicted in Fig. 3.1

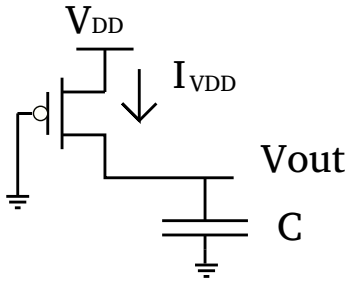

Figure 3.1: Equivalent circuit during a low-to-high transition.

Subsequently, if an instantaneous low-to-high transition occurs the NMOS transistor will discharge the capacitor and a logic zero will appear again on the inverter output. A precise expression for the energy consumption can be obtained. Let us first consider a one-to-zero transition on the input; the total energy drawn by the supply voltage, V_{DD}, can be computed integrating the instantaneous power over the period, leading to:

$$E_{VDD} = \int_0^\infty i_{VDD}(t) V_{DD}\, dt \; = V_{DD} \int_0^\infty C_L \frac{dv_{out}}{dt}\, dt \; = C_L V_{DD} \int_0^{V_{DD}} dv_{out} = C_L V_{DD}^2. \tag{3.2}$$

The energy stored in the capacitor is simply:

$$E_C = \frac{C_L V_{DD}^2}{2}. \tag{3.3}$$

Notice from 3.2 and 3.3 that part of the energy drawn by the supply voltage to charge the capacitor is dissipated on the PMOS device. During a zero-to-one transition the energy dissipated on the NMOS transistor is equal to the energy stored in the capacitor, hence equals to:

$$E_C = \frac{C_L V_{DD}^2}{2}. \tag{3.4}$$

In summary during an entire cycle, consisting of one high-to-low and one low-to-high transition the overall energy consumption is:

$$E_C = C_L V_{DD}^2. \tag{3.5}$$

The power being the energy per unit time, can be written as:

$$P_{cp} = C_L V_{DD}^2 f \tag{3.6}$$

where f is the switching frequency. The formula just derived can be extended to all nodes within a circuit yielding the following, with α the activity factor, f_{clk} the clock frequency, C the capacitance of the circuit, and V_{DD} the voltage

$$P_{sw} = \alpha f_{clk} C V_{DD}^2. \tag{3.7}$$

Notice that α has been introduced in order to account for the fact that not all nodes switch with the same frequency of the clock signal. It is also important to point out that the above formula has been obtained considering complete and non partial transitions. In other words the voltage swing, at each node, is considered to be equal to V_{DD}. Another important aspect that should be noticed is that all the terms, except for α, are known at gate level. Thus, not only, at this level of abstraction great accuracy can be achieved but, the power estimation is also pattern dependent due to the strong relationship between the input activity and α.

3.1.2 Short-circuit Power

Short-circuit power is the power consumed while both the p-network and the n-network are conducting simultaneously during switching. Consider, for example, a rising transition for a simple inverter; there will be a current flow in a timing frame, where the input voltage on the gate is higher than the threshold voltage for the NMOS but lower to completely switch-off the PMOS. As a consequence, within this timing window exists a path from V_{DD} to ground for the current to flow.

Let's now derive a simple formula for the short-circuit current considering an entire cycle of charging and discharging of the load capacitance. We reasonably assume to shape the current spikes with triangle waveforms. In addition, we consider to have symmetric rising and falling time for the input signal. We can write the energy lost during a cycle as follow:

$$E_{sc} = V_{DD} \frac{I_{peak} t_{sc}}{2} + V_{DD} \frac{I_{peak} t_{sc}}{2} \tag{3.8}$$

where t_{sc} represents the time interval where both the PMOS and NMOS are conductive simultaneously, while I_{peak} is the maximum value measured for the short-circuit current. Fig. 3.4 clarify what just said.

I_{peak} heavily depends on the input and output slopes. While t_{sc} is strongly dependent on the signal slope and the driving strength, as well as, the load driven by the network. This last concept can be explained considering, without loss of generality, a simple CMOS inverter. Assume that the circuit is driving a huge load capacitance and a low-to-high transition happens on the input. During the input transient the voltage drops across the drain and source of the PMOS device is approximately zero, thus incurring in a small short-circuit current. On the contrary, if we consider the reverse situation, where the driving capacitance is really small. The voltage drops, during the transient, across the drain and the source of the PMOS device remains equal to V_{DD} most of the time, thus incurring in a high short-circuit current.

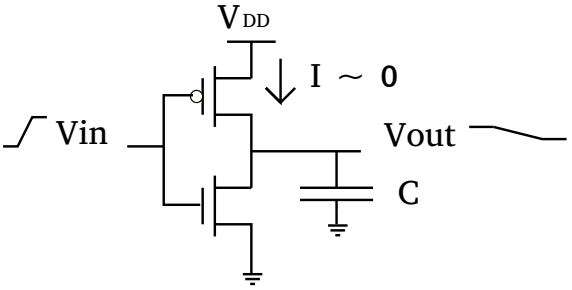

Figure 3.2: Large load capacitance.

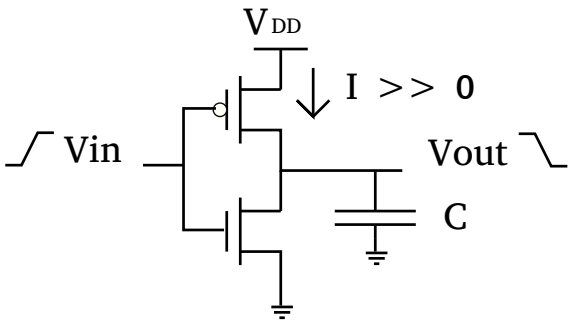

Figure 3.3: Small load capacitance.

Fig. 3.3 and Fig. 3.2 show the two different situations. The right picture illustrates the inverter driving a huge load capacitance; the left picture illustrates the opposite situation.

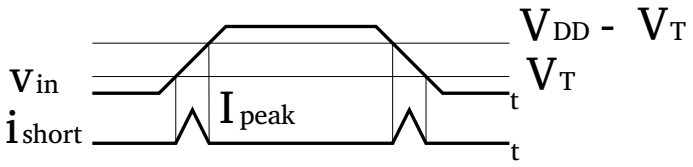

Figure 3.4: Short-circuit current for one low-to-high and one high-to-low transitions.

From the energy we can derive the average power, obtaining:

$$P_{sc} = t_{sc} V_{DD} I_{peak} f. \tag{3.9}$$

From the above discussion it is clear that short-circuit current can be greatly reduced using careful design and proper transistor sizing. Nevertheless, today P_{sc} plays a marginal role in the overall power consumption since V_{DDS} are low and rise/fall times are short [31].

3.1.3 Subthreshold Leakage Power

Subthreshold leakage power is due to a weak current that flows between the drain and the source when the device is supposed to be in the off state. Historically, this current has been considered to be very small, early models for MOS transistors implemented in SPICE even consider this current equals to zero. In today state-of-the-art technology however, with down-scaling in the devices and a lowered supply voltage, it starts to account for more than half of the total power dissipation [3]. Hence, it could not be ignored any longer. Chandrakasan et al. in their paper [32] proposed the following formula for modeling the subthreshold current

$$I_{sub} = K_1 W e^{\frac{-V_{th}}{nV_\theta}} (1 - e^{\frac{V}{V_\theta}}) \tag{3.10}$$

where K_1 and n are constants experimentally determined; while W and V_θ are the channel width and the thermal voltage respectively. The above formula suggests that a possible way to reduce I_{sub} is to increase the threshold voltage. Notice that even a small increment in the threshold voltage results in huge decrease in the current due to their exponential dependency. Unfortunately increasing the threshold voltage keeping the supply voltage constant results in a performance loss. In addition, in today process technology the trend is to decrease the supply voltage, to quadratically decrease the dynamic power; hence the threshold voltage is forced to reduce. Fig. 3.5 compares the subthreshold power consumption for different technologies.

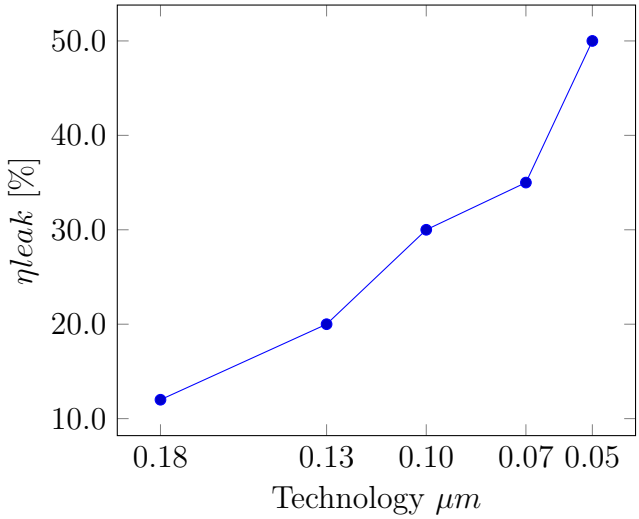

Figure 3.5: Increase in the subthreshold current for different technologies [3].

The Stacking effect helps to mitigate this phenomenon.

3.1.4 Gate-Leakage Power

Gate-leakage power is the direct consequence of the aggressive CMOS technology scaling that requires a decreased oxide thickness. Thinner oxide gives rise to high-electrical fields, that results in tunneling effects through the gate or the oxide bands. The following simplified formula from Chandrakasan et al. [32], clearly shows the exponential dependency between the gate-leakage current and the oxide thickness

$$I_{ox} = K_2 W (\frac{V}{T_{ox}})^2 e^{\frac{-\alpha T_{ox}}{V}} \tag{3.11}$$

where K_2 and α are constants experimentally determined. Accordingly to the upper mentioned formula a naive approach that one may think to reduce the leakage is to increase T_{ox} due to their exponential dependency. However, the thickness oxide should linearly decrease with the technology scaling to avoid short-channel effects. Today, the problem has partly been solved by using high-K gate oxide.

3.2 Glitch Power

When the input of a node changes, it is likely that the node will make several transitions before settling into a stable value. These spurious transitions are referred to as glitches and can, for some circuits, contribute to a significant portion of the overall power consumption. Glitches occur due to circuit topology and unbalance delay path within the circuit. Furthermore, once a glitch has been generated it can be propagated to the downstream logic, causing additional switching activity, and ultimately an increase in the power consumption. In the FIR-circuits that we will consider later on this phenomenon is mitigated by having registers at regular intervals (every tap). Still, it is important to consider glitching activity while designing a macromodel. Ignoring this aspect may lead to a loss in the absolute accuracy as well as in the relative. Fig. 3.6 depicts the generation of a glitch at the output of a NAND gate, due to the delay introduced by the inverter.

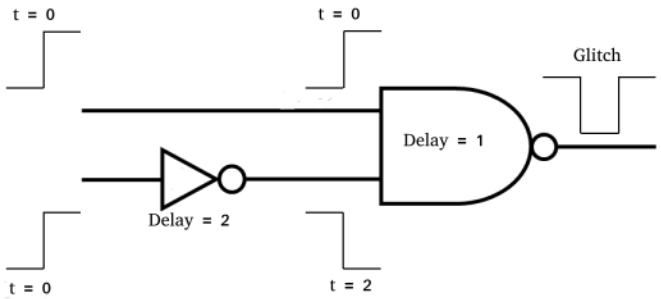

Figure 3.6: Example of a circuit showing a glitch generation.

4

Filter Design

In this chapter we will briefly describe how the filters for which we want to provide power estimates have been implemented in VHDL.

4.1 FIR Filter

In this section we take a more in-depth look at a special case of finite-impulse-response (FIR) filter. In addition, we introduce the basic terminologies used in digital filtering. A digital filter is considered to be an FIR filter if its impulse response $h[n]$ is finite in time and given by

$$h[n] = \begin{cases} 0, & \text{if } n < 0 \\ b_n, & 0 \leq n \leq M \\ 0, & n > M \end{cases}$$

where M is the filter order and b_n are the filter coefficients. A filter of order M has clearly length $N = M+1$. The output of an FIR filter of order M is given by the convolution between the input $x[n]$ and the filter response, precisely

$$y[n] = \sum_{i=0}^{M} b_i * x[n] \tag{4.1}$$

where "*" is the convolution operator. In the z-domain 4.1 can also be expressed as

$$Y(z) = H(z)X(z) \tag{4.2}$$

where $H(z)$ is the transfer function, defined by

$$H(z) = \sum_{i=0}^{M} h[i]z^{-i}. \tag{4.3}$$

The hardware interpretation of 4.1 leads to the following signal flow graph or system diagram, known as direct form.

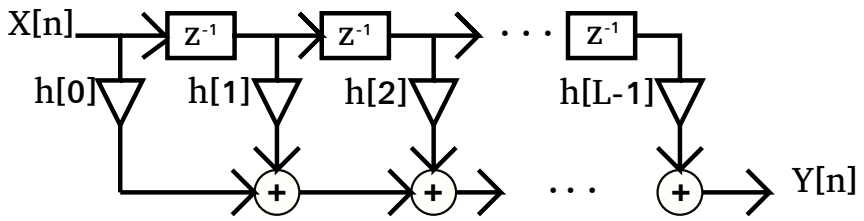

Figure 4.1: Schematic representation for a direct implementation of an FIR filter. This architecture is also known as tapped delay line.

However, a variation of the direct form, known as transposed architecture, is preferred. This is especially true for long filters, since the direct form would require extra pipeline registers to reduce the delay between the adders and achieve the same performance of the transposed architecture. Further gain in performance can be achieved in the transposed form if the coefficients are all power of two or values close to the power of two and fixed values for coefficients are considered. In this case multiplication operations can be transformed in simpler shifting and adding operations.

The transposed form can be obtained reversing or transposing the flow graph of Fig. 4.1 and make necessary modifications, in more detail:

- Exchange the input with the output
- Inverting the direction for each signal
- Substituting each adder with a node

This can be derived as a consequence of Mason's gain formula for signal flow graph or Tellegen's theorem [33] and leads to the circuit depicted in Fig 4.2

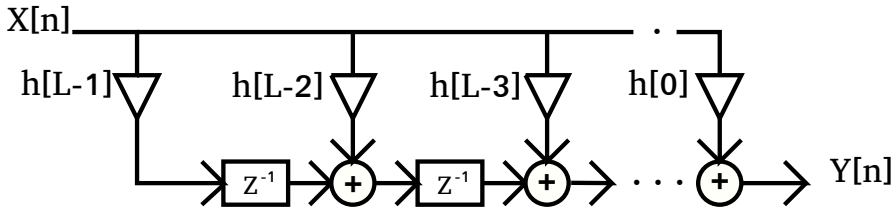

Figure 4.2: Schematic representation for a transposed form implementation of an FIR filter.

It is clear from Fig 4.1 and Fig 4.2 that an FIR filter simply consists of a collection of delay elements, multipliers and adders. In addition, we can also notice a regularity in the FIR filter structure given by the repetition of a basic building block (or atomic component). This basic building block is shown in Fig. 4.3 and we will refer hereafter to this as MAC.

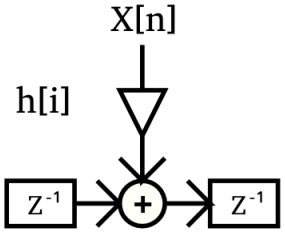

Figure 4.3: Basic building block of an FIR filter.

4.2 Complex FIR Filter

Let be x and c two complex numbers, i.e, $x = x_r + jx_i$ and $c = c_r + jc_i$. Multiplication between x and c can be performed as four real multiplications as can be seen in 4.4

$$(x_r + jx_i) \cdot (c_r + jc_i) = x_r \cdot c_r - (x_i \cdot c_i) + j(x_r \cdot c_i + x_i \cdot c_r) \tag{4.4}$$

this operation requires one addition, one subtraction and four multiplications. Hence, if we consider a 20-tap filter, this would require 80 multipliers, 20 adders and 20 subtractors. Notice that additional adders are required to sum up the results generated by each tap. Saving in hardware logic and power consumption can be achieved if a modify version of equation 4.4 is considered [34]. This form in known as canonical implementation and is reported below

$$(x_r + jx_i) \cdot (c_r + jc_i) = (c_r - c_i) \cdot x_r + (x_r - x_i) \cdot c_i + j[(c_r + c_i) \cdot x_i + (x_r - x_i) \cdot c_i]. \tag{4.5}$$

In this case the number of multipliers is reduced to three units, while the number of adders increased by one unit. Assuming that a multiplier is wider and more power hungry then an adder, on-chip area and power consumption are reduced. The entire 20-tap filter, now requires 60 multiplications, 60 additions and 40 subtractions. If we substitute 4.5 in 4.1 we obtain the entire filter expression

$$\begin{aligned} y &= \sum_{n=0}^{M} x_{r,(t-n)}(c_{r,n} - c_{i,n}) + \sum_{n=0}^{M} (x_{r,(t-n)} - x_{i,(t-n)})c_{i,n} + \\ &= j[\sum_{n=0}^{M} x_{i,(t-n)}(c_{r,n} + c_{i,n}) + \sum_{n=0}^{M} (x_{r,(t-n)} - x_{i,(t-n)})c_{i,n}] \end{aligned} \tag{4.6}$$

Notice that each term in 4.6 represents a real FIR filter. Therefore a complex structure can be simply implemented using three real FIR filter, being the second and the fourth summation identical. Fig. 4.4 shows the overall structure.

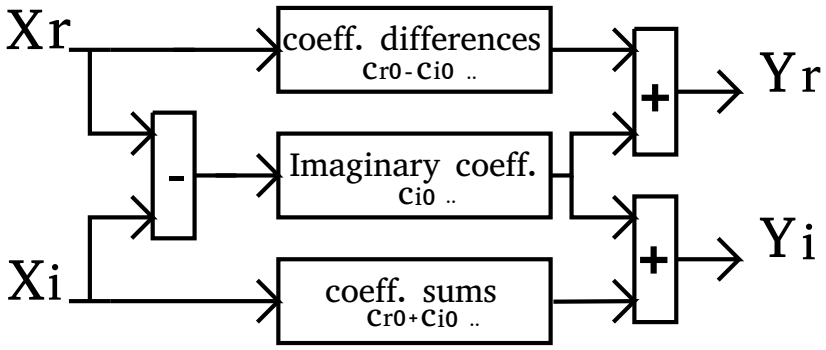

Figure 4.4: Complex FIR filter built with three real FIR filters.

5

RT Level Power Estimation

In this chapter we will describe our flow after a brief overview on tool used for our work in addition to the recommended flow for power estimation suggested by the EDA vendors. With our macromodel the system designer will be able to obtain power estimation directly at functional level, in MATLAB, where no circuit information is available. Our method can be classified as a cycle-accurate macromodeling and exploits the high correlation between power consumption and input activities.

5.1 Recommended Estimation Flow in RC-LP Engine

The tool used in our work is Encounter RC-LP engine [35]. Depending on the level of details that are available at the current level of abstraction different flows can be implemented. The simplest estimation flow we can think of at gate-level consists of a logic netlist and a technology library containing pre-characterized gates. If these are the two only items of information available, the tool will automatically assign default switching probabilities on the input pins, and let those probabilities propagate in the downstream logic.

RC-LP engine models the probability at each node in terms of signal probability and toggle rate. The former is the probability of a signal being in the high logic value, while the latter is the toggle count per unit time. For example, a net with a signal probability that equals to 0.5 and a toggle rate of 1 will be in the high state fifty percent of the time and will make one transition from one to zero or from zero to one per unit time. By default RC-LP engine uses a signal probability that is equal to 0.5 and a default toggle rate of 0.02 unit per nanoseconds [35].

A significant improvement in the accuracy can be achieved if real toggle rates are provided as the internal engine may use probabilities that do not reflect realistic values. In this respect, RC-LP engine allows the user to specify the actual probability at each node up to four different ways. Only two of them, that are interesting for our work will be explained, for the others we refer the readers to the manual. The first of them is to manually assert the switching activity, using the following commands:

- $set\_attributelp\_asserted\_probabilityfloat/designs/design/*/nets/net$
- $set\_attributelp\_asserted\_toggle\_ratefloat/designs/design/*/nets/net$

Notice that if a node does not have any user-asserted probability but it is in the fanin cone of a node that contains an asserted value, the tool will automatically propagate the switching activity. If it is not possible to find a fanin cone that contains the node with asserted probability the tool will use the default values. The second way, highly recommended by the vendor, is to provide the switching activity in the form of a toggle count format (TCF), a switching activity interchange format (SAIF) or a value change dump (VCD). Since we are extracting the power consumption at each clock cycle, to the best of our knowledge, the VCD file is the only solution. Hence, the VCD file will be used during the characterization phase.

In section 3.2 we also mention the importance in recording spurious activities at the output of a node to correctly estimate the power consumption. Glitches can be taken into account, using realistic gate delays during the dumping of the VCD file. One way to accomplish this is to use the static timing analysis (STA) function within the RC-LP engine to generate the standard delay format (SDF). The entire flow recommended by the vendor is depicted in Fig. A.1.

5.2 Mechanisms of Power Consumption in RC-LP Engine

Finally, it is important to briefly discuss how RC-LP engine models power dissipation. The tool classifies the power dissipated by each instance as the sum of two major components: leakage power and internal power. Leakage power is caused by an unwanted current that flows between the drain and the source when the transistor is supposed to be in the off state. As stated in previous chapter, it depends on several factors such as threshold voltage, sizing and so on. It used to be modeled within the library provided by the foundry as a constant. However, as the technology scaling advances, the importance that leakage plays in the overall power consumption increases. Hence, most libraries specify different power profiles for the same instance depending on the operating conditions.

Internal power is the power consumed within the boundaries of an instance, it consists of short-circuit and switching power. An instance's internal power is modeled as a function of the input slew rate and the output load capacitance using LUT tables. Extrapolation methods take place if the exact index is not stored in the tables.

Net power accounts for the switching power dissipated by the charging and discharging of the capacitance of each pin driven by the instance. Fig. 5.1 shows all the power mechanisms considered by RC-LP engine.

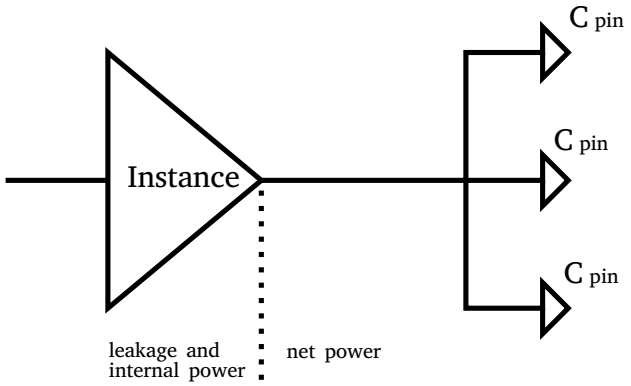

Figure 5.1: Mechanisms of power dissipation within an instance.

5.3 Proposed Macromodel

We start from the trivial observation that in CMOS logic inputs have to toggle in order to dissipate dynamic power. In general higher activity on the input pins results in higher power consumption. Hence, the basic idea in our model is to capture this behavior using the Hamming distance. The Hamming distance between two consecutive vectors can be defined as follows: let u and v be two inputs of length m; the Hamming distance is the number of bits that toggle between two consecutive words u and v, more precisely it can be defined as:

$$H = \sum_{i=1}^{m} v_i \; XOR \; u_i. \tag{5.1}$$

In general, if we consider vectors of length m, we can identify up to m+1 different classes of Hamming distance based on the definition given above. To every class we can associate two different parameters c_i and α_i. The former represents the average power consumption for class i, while the latter is an activity factor that assumes value '1' if class i occurs in the current cycle, otherwise '0'. Therefore, in each clock period j the power consumption can be estimated using the following equation:

$$Q[j] = [c_1 c_2 .. c_{m+1}][\alpha_1 \alpha_2 .. \alpha_{m+1}]^T \tag{5.2}$$

where $\mathbf{c}$ is the coefficients vector and $\boldsymbol{\alpha}$ is the activity vector. The methodology is fairly simple but its resolution, accuracy and robustness can be increased considering additional bit-level information [36]. For example, we can think of enhanced the model observing the number of stable ones or stable zeros among two consecutive input vectors. For instance, if we apply the first criterion, so we consider the number of bits that remain stuck to one between successive clock cycles, the class with Hamming distance one can be additional sub-divided in m sub-classes:

$$H_{1_1}, H_{1_2}, \mathrm{H}_{1_3}, ..., H_{1_m}. \tag{5.3}$$

The first index refers to the Hamming distance while the second refers to the number

of stable ones. This splitting criteria can be applied, in turn, for all the remaining classes.

We are now ready to present the two main steps that constitute our macromodel: *characterization* and *power estimation*.

5.3.1 Characterization

The first step in any RT level power macromodel is the characterization. As previously introduced in section 2.5.2, characterization is the process by which the macromodel is trained. Generally, a characterization flow consists of two main steps: macromodel tuning and metrics extraction. The former entails simulating, at gate level, the circuit under observation with a set of input stimuli in order to obtain the power values. The latter is the process to extract from the input stimuli a set of metrics and binds those metrics with the power values.

One of the key features during the macromodel tuning that is usually neglected in the analysis of performance for a macromodel technique is the stopping criteria. In section 2.5.2 we stated that the characterization is allowed to be quite expensive in term of computational complexity. However, if on one hand, it is acceptable to have long characterization time to achieve high accuracy, on the other hand, there are some situations in which we can trade-off precision in the estimation with speed-up in the macromodel tuning. In this work we present two stopping criteria. One is based on the full simulations of an entire trace, the other is based on a sequential stopping rule.

Proposed Stopping Criteria

The first method is based on the full simulation of an entire trace, we tune the macromodel by generating a trace of n uniformly distributed test vectors, where n is specified by the user. Then for each pair of input the instantaneous power value is collected and classified in the proper Hamming class. The hope is that generating a huge number of test vectors will allow us to cover most of the transitions that may happen within the circuit. However, when this method achieves the maximum precision for the average over the entire trace, it becomes far too slow. It becomes impractical to use on large circuit or when different runs are needed to estimate the average power. Moreover, when dealing with random test vectors it is likely that the sample average approximates the population average after few iterations [37]. As a consequence of this observation, we introduce the second method based on confidence intervals and a sequential stopping rule.

Before presenting the algorithm it is crucially important to give some mathematical definitions as well as introducing some theorems that are behind statistical inference. We start by defining the typical way of approximating the mean values for m independent and identically distributed random variables X_1, X_2 .. X_m as

$$X_M = \sum_{i=1}^{M} \frac{X_i}{M}. \tag{5.4}$$

The idea is to chose the sample size large enough to satisfy the following inequality:

$$\mathrm{P}(\mid X_M - \mu \mid \leq \epsilon) \geq (1 - \alpha) \tag{5.5}$$

where μ is the population mean, ϵ is half-width of the confidence interval (also known as tolerance error) and α is the uncertainty that can be decided in advance. It is clear that $\mid \mu - X_M \mid$ decreases as the number of samples increases. In the extreme case where the sample size equals the population size, X_M converges to μ. However, it is also clear that as the sample size increases also the cost of computing X_M increases. Hence, we would like to approximate μ using a limited number of samples. When information about the population distribution is known, a theoretical upper bound for the population mean can be computed using Hoeffding inequalities. Unfortunately, in the general case, such information is not available and the only way to estimate μ is to use sequential stopping rules. Usually, a sequential stopping rule consists of the following steps:

- Generate up to M samples, where M should be decided in advance.
- Infer population properties (variance or mean) from sample moments.
- Estimate the error probability on the basis of the sample moments. If the error is larger than a given threshold then generate additional samples, else stop.

The pseudo-code for our stopping rule is reported below. Notice that $t_{\alpha,k-1}$ is the $(1+\alpha)/2$ quantile of the t-distribution with $k-1$ degrees of freedom.

Algorithm 1: stopping rule

Result: X_M

```
generate a batch of samples k = M_0;
choose a value for the confidence coefficient α;
choose a value for the half-width of the confidence interval;
compute the sample mean X_M;
compute the sample variance as: S^2 = 1/(k-1) Σ_{i=1}^{k} (X_i - X_M)^2;
while (t_{α,k-1} sqrt(S^2/k)) ≤ ε do
    generate additional samples M;
    k = k + M;
    compute the sample variance;
end
```

We should highlight that the basic assumption in the previous algorithm is that the distribution of $\frac{(X_M-\mu)}{\sqrt{\frac{S^2}{k}}}$ is the Student's t distribution with $k-1$ degrees of freedom.

This happens to be true if the population is normally distributed. Therefore, we checked the normal distribution hypothesis plotting the Q-Q plot for each Hamming class. Below we report the results obtained for the class with Hamming distance equals to three, considering 70 samples.

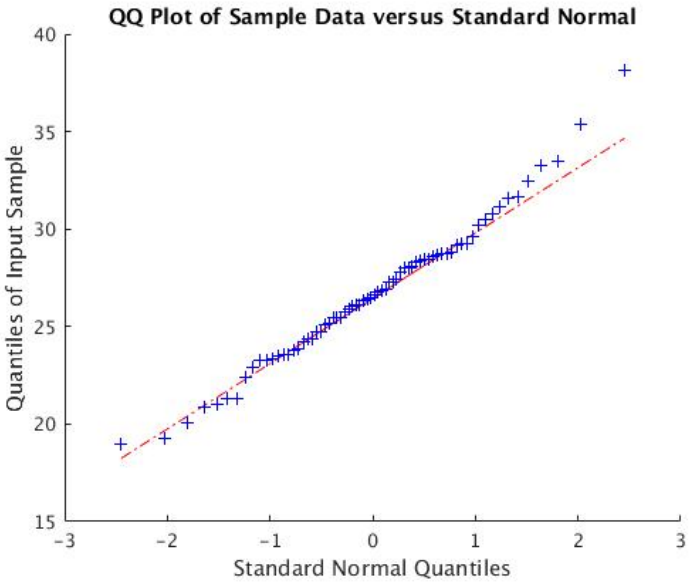

Figure 5.2: Q-Q plot for Hamming class three.

Figure 5.2 shows that our data are approximately normally distributed, as a consequence, the previous assumption hold. Extensively tests have been done also for the other Hamming classes leading to similar results.

Characterization: Overview of Methodology

In the following a brief overview of our characterization flow is presented. This is only a general overview, hence not all details are here reported. Notice also that some of the steps below might be skipped to speed-up the characterization time, or iterate for the entire trace if the user selects the flow based on the complete simulation.

1. Read parameters provided by the user
2. Create working directory
3. Generate coefficients values
4. Generate input trace
5. Read RTL component
6. Map design to cells
7. Read each pair of input vectors
8. Compute Hamming distance
9. Extract power if it is not converged (see section 5.3.1)
10. Repeat from 7 until no more vectors are available
11. Write on a file the coefficients for each Hamming class
12. Clean working directory

5.3.2 Power Estimation

Our estimation methodology relies on the characterization of an atomic component. For atomic component we intend any combinational logic in between registers. The reason why we restrict ourselves to such structure is that our characterization process registers the power dissipated as a function of the Hamming distance between two vectors. Hence, at most one transition on the primary inputs may happen per clock cycle.

Once the atomic components have been modelled and the macromodels stored in a library, the estimation process can take place. The average power consumption for a circuit is estimated evaluating the macromodel for each transition on the primary input of an atomic component. This returns an instantaneous power value. The instantaneous power values of each atomic component are summed up to obtain the power consumption for the current clock cycle. In turn, all the power estimates at each clock cycle are summed together over the entire simulation to obtain the total value. Finally, the average value is simply computed dividing the total value by the duration of the simulation expressed as numbers of clock cycles.

6

Results and Discussion

In this chapter, we will consider various parameters that influence the power consumption in the 65nm technology and we will evaluate how those parameters affect our estimation. In particular, we will focus on clock frequency, coefficient word-length, coefficient update frequency and roll-off. All the results have been obtained using the flow based on the stopping criterion, presented in section 5.3.1.

6.1 Atomic Structure

In order to provide power estimation for a complex circuit, atomic components should be identified and characterized. In the case of a transposed architecture this choice is straightforward and matches with the MAC structure previously introduced. However, one question still needs to be addressed: is it possible to estimate the power registering only the activity on the primary input of the atomic structure? To answer this question, we set up the following experiment. We considered a MAC structure with 8-bit input and 6-bit coefficient synthesized at 833.33 MHz. Subsequently, we performed a simulation feeding the circuit with uniformly distributed pseudo-random integers for coefficient and input and we updated the coefficient every 1000 clock cycles. It is reasonable to assume that the coefficients are updated less frequently than the input in DSP designs. The results are shown in Fig. 6.1.

Fig. 6.1 shows a clear relation between the average power consumption and the Hamming distance on the primary input pins. Notice that even if we register a Hamming distance that is equal to zero we still have power dissipation within the basic component. This is the consequence of having sequential logic in the MAC structure that introduces temporal correlation between the input vectors. Upon first consideration, it seems that registering the activity on the only primary input is not enough to model the power consumption for the MAC structure and information on the state registers is required. Though, if we run the simulation for a long period of time the states of the registers become independent of the initial one. This leads to a huge simplification of the problem when the main goal is to provide the average power [1]. From Fig. 6.1 it is also evident that if the coefficients are updated less frequently than the input pins modeling the power as a function of the Hamming distance on the primary input is enough to provide good power estimations.

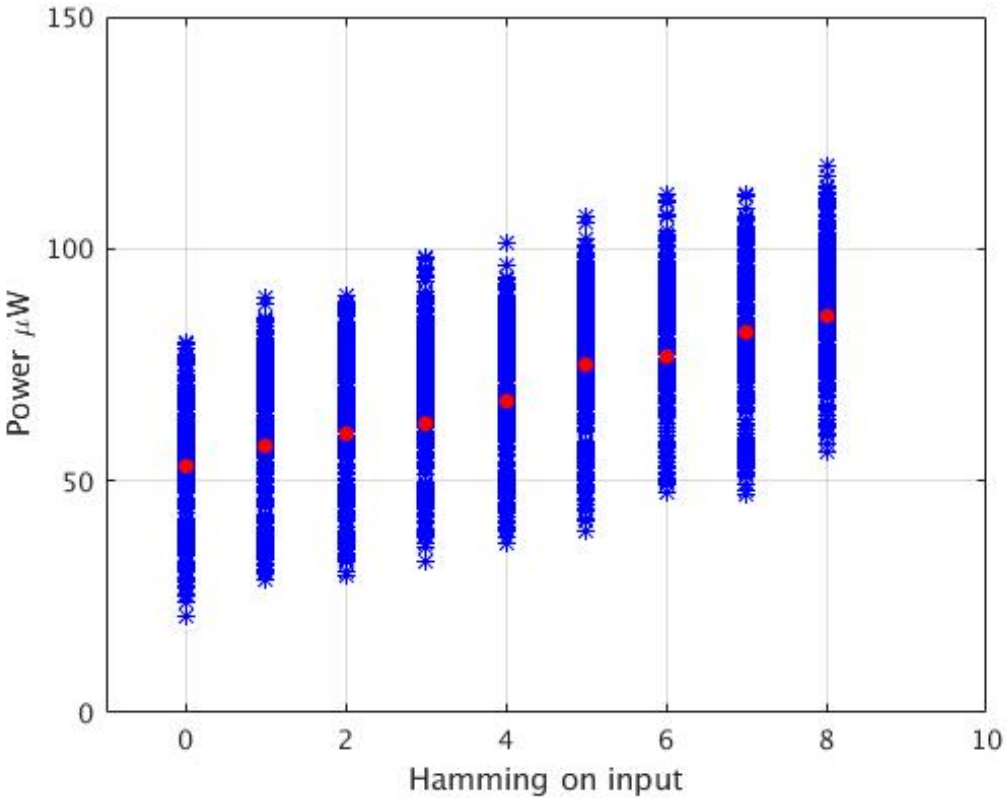

Figure 6.1: Average and instantaneous power values associated with the Hamming distance computed on the input pins. For each Hamming class the blue "*" represents the instantaneous power values, while the orange "*" the average value.

6.2 Results on FIR filter

In the next sections we will mainly focus on a 19-tap FIR filter since for this filter we have available a script in MATLAB to obtain a real trace. In addition, 19 taps is a reasonable length for a DSP design and it is enough to check the validity of our approach. We will observe how the model behaves for different input and coefficient word-lengths, as well as, clock and coefficient update frequencies. We will also carry out studies on the input trace to justify why we provide a characterization based on random-generated input signals. Unfortunately, due to lack of time only the results for the 19-tap transposed FIR filter are presented in the next sections and we have to postpone to future works the ones on the 19-tap complex design.

6.2.1 Statistics of Input Signal

In this section we will analyze the input trace and we will reason about our choice to characterize our basic building block with random-generated input signals. Signals in DSP components may be encoded using different representations such as one's complement, magnitude plus sign bit and two's complement. Nonetheless two's

complement representation is the most commonly used. Hence, we will limit our observations to this particular encoding. We considered a 19-tap transposed FIR filter and we simulated it using a random and a real trace. The latter is generated from a square-root-raised-cosine filter implemented in MATLAB. The results obtained dumping the toggle count format files are tabulated below.

Table 6.1: The table shows the results obtained from the toggle count format file for an 8-bit input 6-bit coefficients FIR transposed filter. The left columns show the results for a random trace. The right columns for a real trace generated from a square-root-raised-cosine filter implemented in MATLAB.

Output TCF file				
Pin name	Prob. to be high (random trace)	Toggles (random trace)	Prob. to be high (real trace)	Toggles (real trace)
clock	0.50	$1.66 \cdot 10^9$	0.50	$1.66 \cdot 10^9$
reset	0.00060	$1.67 \cdot 10^5$	0.00060	$1.67 \cdot 10^5$
Xin[0]	0.51	$4.08 \cdot 10^8$	0.50	$4.13 \cdot 10^8$
Xin[1]	0.49	$4.15 \cdot 10^8$	0.49	$4.10 \cdot 10^8$
Xin[2]	0.50	$4.19 \cdot 10^8$	0.50	$4.22 \cdot 10^8$
Xin[3]	0.49	$4.21 \cdot 10^8$	0.51	$4.08 \cdot 10^8$
Xin[4]	0.48	$4.26 \cdot 10^8$	0.50	$4.06 \cdot 10^8$
Xin[5]	0.49	$4.07 \cdot 10^8$	0.49	$4.34 \cdot 10^8$
Xin[6]	0.49	$4.13 \cdot 10^8$	0.51	$3.68 \cdot 10^8$
Xin[7]	0.49	$4.23 \cdot 10^8$	0.49	$2.17 \cdot 10^8$

As shown in Table 6.1 the differences for the signal probability are negligible between the two traces. Instead, we can identify some dissimilarities for the toggling activity in the most significant bits. In particular, we can notice that the real trace can be divided into two regions. The first goes from the LSB to the MSB-2 and behaves as a random signal. The second goes from the MSB-1 to the MSB and has a lower toggle activity. The two regions are shown in Fig. 6.2.

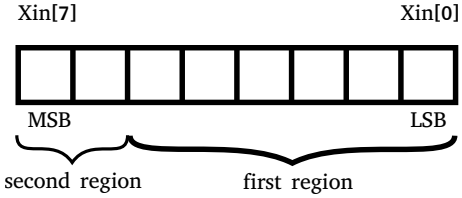

Figure 6.2: Observed regions for the real signal.

We observed the same behavior considering different input and coefficient word-length for our 19-tap transposed architecture.

Despite these differences we decided to perform characterization using random-generated input signals mainly for two reasons:

- simplicity. The choice of using random-generated input vectors simplify considerably the characterization phase.

- loss in accuracy is negligible.

To give an explanation to the last bullet point we set up the following experiment. We considered an 8-bit input and 6-bit coefficients transposed FIR filter and we performed two different simulations using the same set of coefficients. In the first simulation we used real inputs obtaining a power value of 1476.03 μW while in the second simulation we used random inputs obtaining a power value equal to 1581.74 μW[1]. The error made is only 6 percent.

While it is possible to provide a characterization based on random input vectors with a negligible loss in accuracy, it is not possible if we consider the coefficients. Performing a characterization based on random values for the coefficients will result in having large errors during the power estimation phase. As a consequence, care is required from the system designer to provide typical or expected coefficients for the target system. This can be explained if we consider Table 6.2.

Table 6.2: Power values for an 6-bit input and 5-bit coefficients FIR filter. The first column shows the reference value. The second and the third columns show power values obtained considering a characterization based on typical and random coefficients.

Reference value	Typical coeff.	Random coeff.
916.27μW	965.10μW	1067.59μW

The first column of Table 6.2 shows the reference power value for a 6-bit input and 5-bit coefficients FIR filter synthesized at 833.33 MHz. This power value[2] has been obtained from RC-LP engine considering a real trace generated as in previous case. The second and the third columns refer to power values obtained for the same trace with our macromodel considering two different training situations. In the second column typical (or expected) coefficients have been used for the characterization while in the third column random coefficients. As can be seen, a characterization with random coefficients leads to a larger error.

6.2.2 Clock Frequency

In this section we evaluate how variations in clock frequency affect the power consumption in our transposed implementations, as well as, how our model reacts to those variations. In particular, we should distinguish between two different cases. In the first case the circuit is synthesized at the maximum operating frequency and simulated at lower frequencies, while in the second case synthesis and simulation frequency coincide. We decided to narrow down our observations only to the first case, due to the linear dependency between power consumption and simulation frequency. We considered a confidence level of 95 percent and an error tolerance of +/- 5 around the mean. The size of the trace used for the characterization consists of 30000 random generated test vectors. The power estimation trace consists of 5000

[1]Both power values refer to a 65nm CMOS technology.

[2]See footnote 1.

test vectors generated from a square-root-raised-cosine filter implemented in MATLAB. The coefficients are the same for both characterization and power estimation, and are considered to be fixed over the entire simulations. It is worth explaining at this point what we mean with the words "fixed coefficients". This does not indicate that we synthesize the design with fixed values for the coefficients. Instead, it means that we are synthesizing using full multipliers but we keep as input to those multipliers the same set of coefficients for the entire simulation. The zero-delay model is used in our simulation, hence, glitches are here not taken into account. The results have been obtained running our model on netlists synthesized at 833.33 MHz. The outcomes for different combinations of input word-length and coefficient word-length are shown in Figs. 6.3, 6.4 and 6.5.

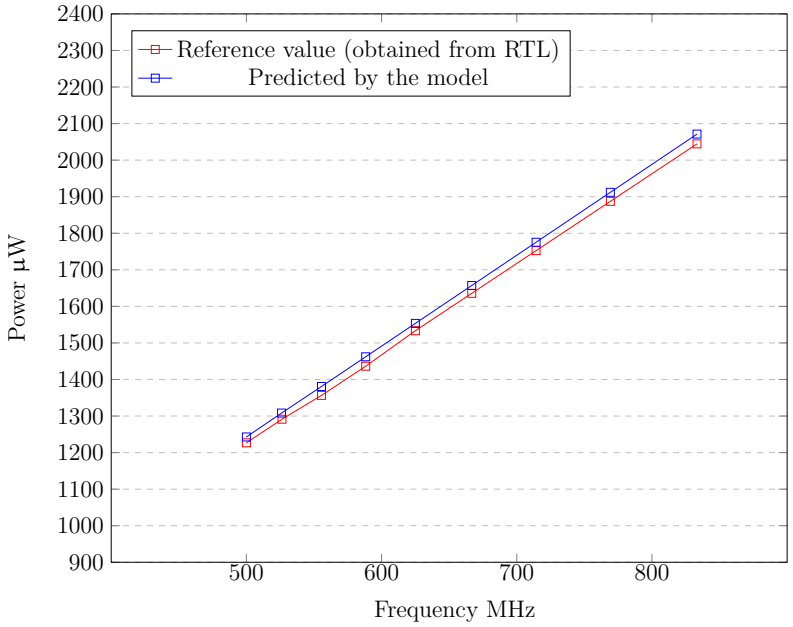

Figure 6.3: Circuit synthesized at 833.33 MHz. The simulations are performed in a range from 833.33 MHz to 500 MHz. In this case the zero-delay model is considered for the simulations. The input is 10 bit while the coefficients are 8 bit.

Figs. 6.3 and 6.4 show that our model follows the trend of our golden reference, depicted as a red line. We remind our readers that our golden reference is the power values provided by the RC-LP engine, which we assume to be correct. It is equally clear that in both cases we do not have a perfect overlap, even if we use the same set of coefficients for both characterization and power estimation. On the other hand, Fig. 6.5 shows a large underestimation in the power estimates. The root cause of this error can be identified in our characterization methodology. Specifically, we are performing an off-line characterization, hence, it is likely that the synthesis tool will map arithmetic operations with different logic for the MAC structure and the entire filter. This leads to considering two different netlists for the characterization and the power estimation phase. Therefore, on one hand an off-line characterization allows us to have a more general model since we are not binding it to one given netlist. On the other hand, there is an accuracy penalty during the power estimation. However the error made is acceptable in all the cases and below the requested targeting

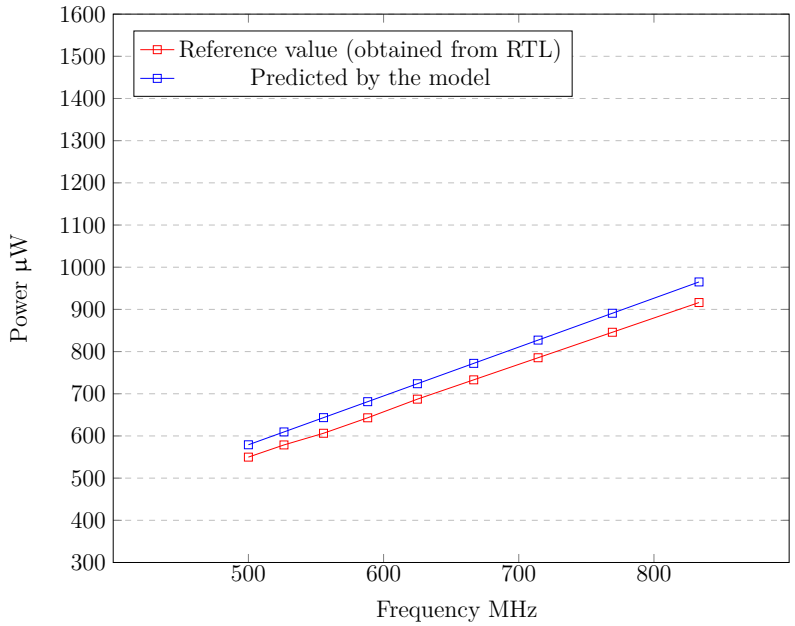

Figure 6.4: Circuit synthesized at 833.33 MHz. The simulations are performed in a range from 833.33 MHz to 500 MHz. In this case the zero-delay model is considered for the simulations. The input is 6 bit while the coefficients are 5 bit.

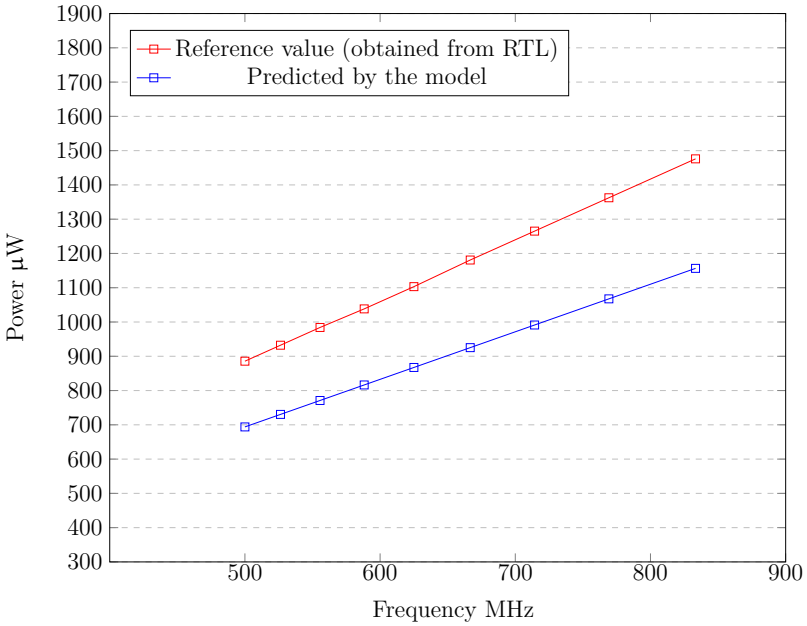

Figure 6.5: Circuit synthesized at 833.33 MHz. The simulations are performed in a range from 833.33 MHz to 500 MHz. In this case the zero-delay model is considered for the simulations. The input is 8 bit while the coefficients are 6 bit.

accuracy of 25 percent. In the first case it is lower than 3 percent, reaching a value up to 22 percent in the third case.

In order to compensate for variations in the clock frequency the following equation has been integrated in our model

$$P_{sw} = \frac{f}{f_0} P_{sw0} \tag{6.1}$$

where f_0 is the maximum operating frequency and P_{sw0} is the estimated power value at f_0.

6.2.3 Coefficient Word-length

Another aspect considered in this work is how power changes with respect to the coefficient word-length in a transposed architecture. We also evaluate how our model reacts to those changes. Again, we considered a confidence level of 95 percent and an error tolerance of +/- 5 around the mean for the characterization phase. Characterization and estimation trace have been generated as in section 6.2.2.

We can notice, from Fig. 6.6, that the power scales approximately linearly with the coefficient word-length. Therefore, we decided to run the characterization only for the two boundary netlists. Eventually, linear interpolation is used to extract power values for intermediate word-lengths. Consider, for example, a transposed architecture with a 6-bit input; its coefficients will be in a range from 5 up to 8 bit. Characterization is performed only for the sets (6,5) and (6,8) where the first number refers to the number of bits for the input, while the second refers to the number of bits for the coefficients. Subsequently, if the user requires another configuration (i.e, (6,7) or (6,6)) linear interpolation is used to extract power values. The results are shown in Figs. 6.7, 6.8 and 6.9. Once more Fig. 6.7 and Fig. 6.8 show that our power estimations are not far from the reference value. However, this is not the case if we consider Fig. 6.9, where the error ranges from an underestimation of 20 percent to an overestimation of 20 percent.

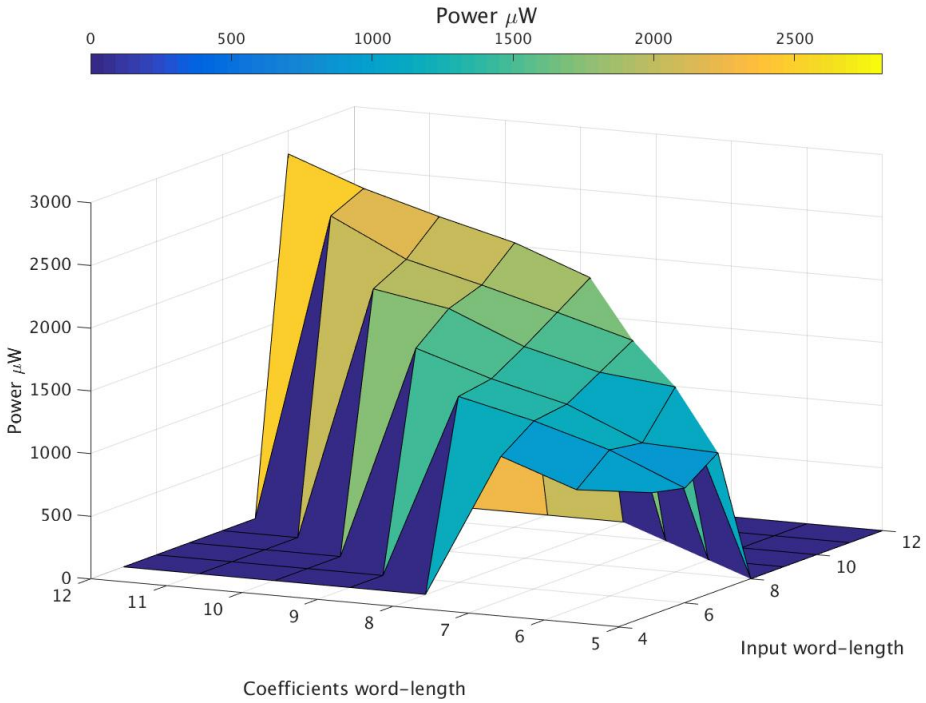

Figure 6.6: Power surface considering different input-coefficient combinations in a 19-tap transposed FIR filter. The input range from 5 to 10 bits while the coefficients from 5 to 12.

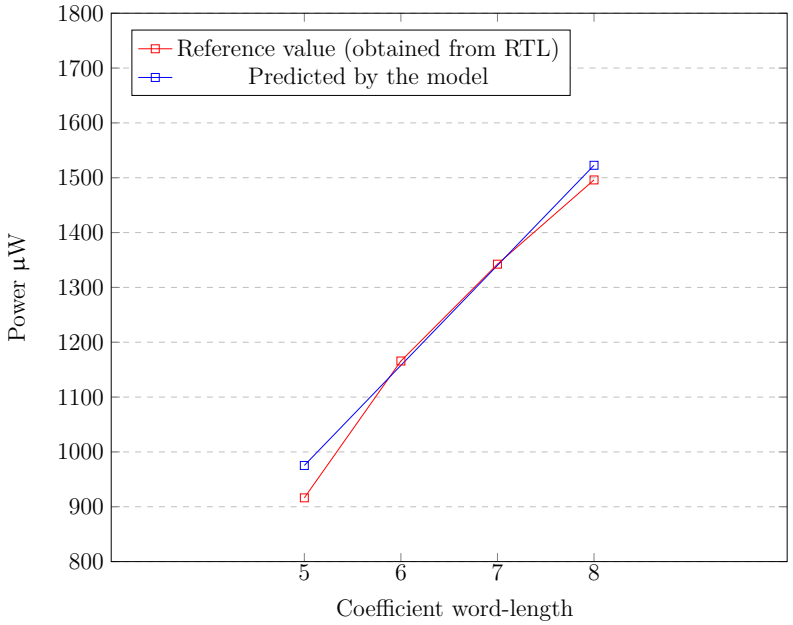

Figure 6.7: Netlist synthesized and simulated at 833.33 MHz. We consider a 6-bit input, while a range from 5 to 8 bits for the coefficients.

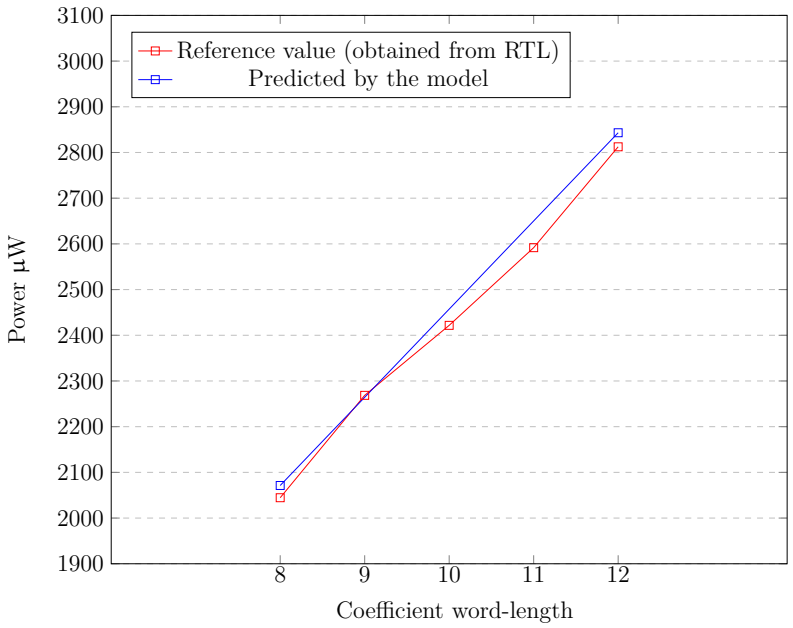

Figure 6.8: Netlist synthesized and simulated at 833.33 MHz. We consider a 10-bit input, while a range from 8 to 12 bits for the coefficients.

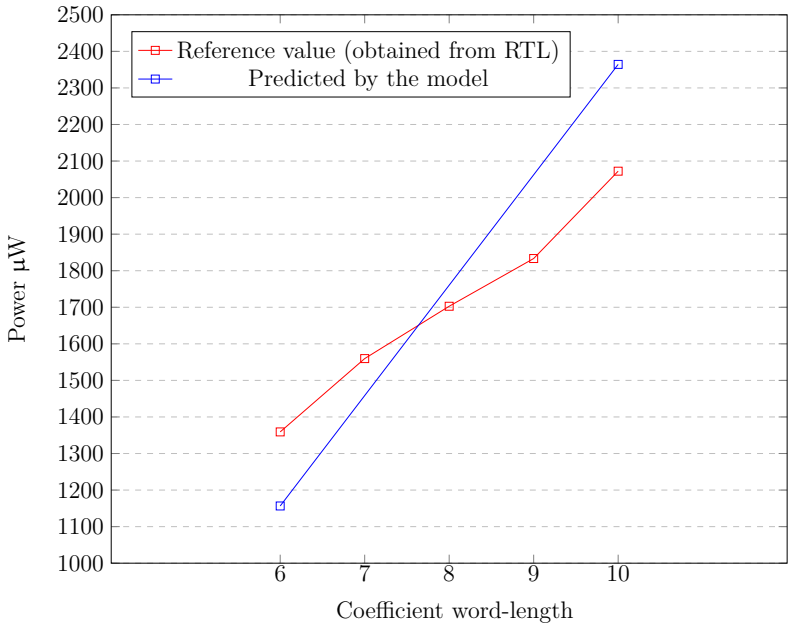

Figure 6.9: Netlist synthesized and simulated at 833.33 MHz. We consider a 8-bit input, while a range from 6 to 10 bits for the coefficients.

All the previous results are based on netlists synthesized at 833.33 MHz; however, the system designers may require higher throughput. If this is the case, there are two possible scenarios: use parallel architectures or increase the operating frequency. Focusing on the second option we decided to carry out some experiments to see if our assumption of linear dependency still holds at higher frequency. We considered a clock rate of 1 GHz; that is in a practical scenario, the maximum operating frequency for a DSP design. Hence, we first synthesized our FIR filters at 1 GHz and on top of those netlists we ran our characterization flow.

The results are reported in Fig. 6.10 and Fig. 6.11. It can be seen that at high frequency the assumption on linear dependency between power consumption and coefficient word-length does not hold anymore. This result is reasonable since the synthesis tool uses more complex heuristics to meet the stricter timing constraint; as a consequence, there is more variability in the power values. From the model point of view, the error made is still acceptable and below the precision requested, but in this case linear interpolation may not be a good solution to extract power values for intermediate word-lengths. In this case we propose a LUT-based approach. Characterization is performed for each input-coefficient combination and the obtained power values tabulate into a pre-defined structure. During power estimation the proper power values will be fetched from the table.

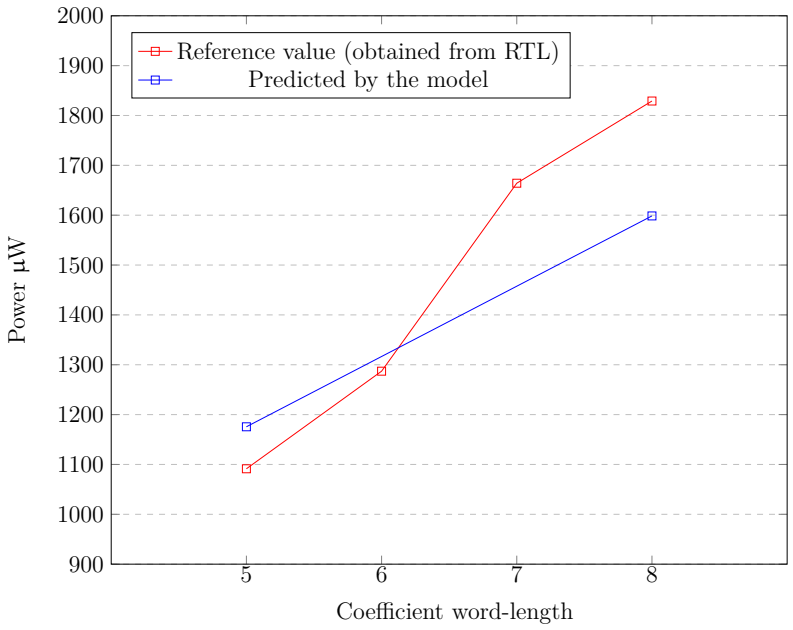

Figure 6.10: Netlist synthesized and simulated at 1 GHz. We consider a 6-bit input, while a range from 5 to 8 bits for the coefficients.

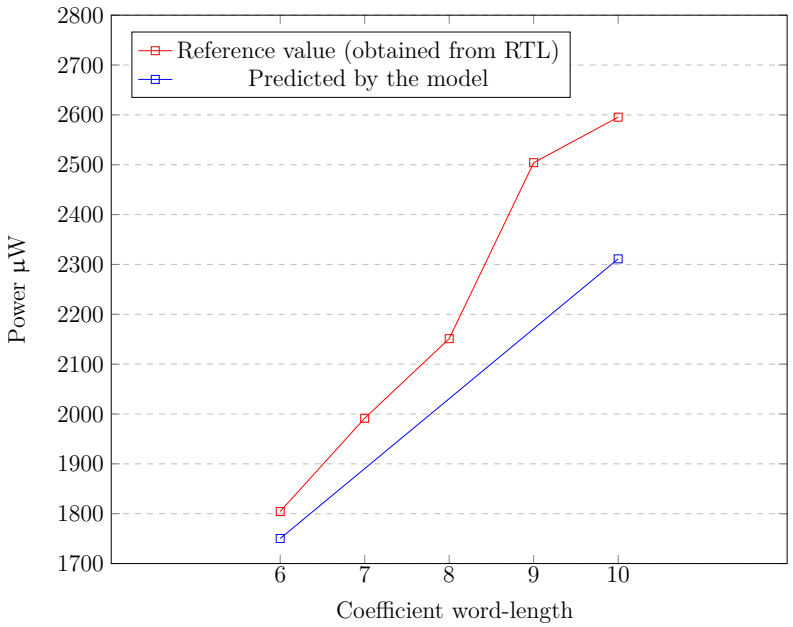

Figure 6.11: Netlist synthesized and simulated at 1 GHz. We consider a 8-bit input, while a range from 6 to 10 bits for the coefficients.

6.2.4 Coefficient Update

Until now we have considered only fixed coefficients for the entire simulation trace. Nevertheless, it is likely that the user may want to update them at run-time during simulations. We set up this experiment considering a simple but still common situation. We generated a set of coefficients using the square-root-raised-cosine filter implemented in MATLAB. Eventually, we scaled those coefficients using a factor between 0 and 1 to model the behavior that could happen if the filter was part of a multi-input multi-output filter (MIMO-filter) for tracking polarization rotation. Then for each set of input-coefficient word-length we ran multiple simulations changing the update frequency. The results are reported in Figs. 6.12, 6.13 and 6.14.

It is evident that the increase in power consumption is negligible in all cases if we consider infrequent updates. Nevertheless the power behavior changes as it approaches the maximum update of one clock cycle. We can observe that close to that value the power consumption increases faster for wider coefficient word-length.

Since in filters for fiber-optic communication systems the coefficients are not updated that often, we decided to rely on a characterization considering fixed coefficients. One more time, it is worth mentioning that with the words “fixed coefficients” we intend that we keep the same set of coefficients as input to our circuit synthesized using full multipliers, instead of synthesizing our design with fixed values for the coefficients.

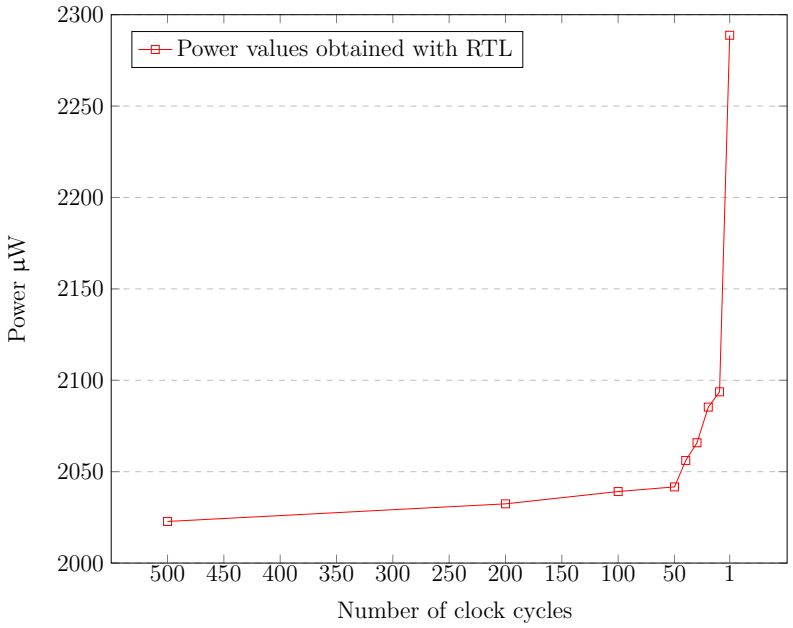

Figure 6.12: Power values considering different coefficients update frequencies for a 10-bit input and 8-bit coefficients transposed FIR filter.

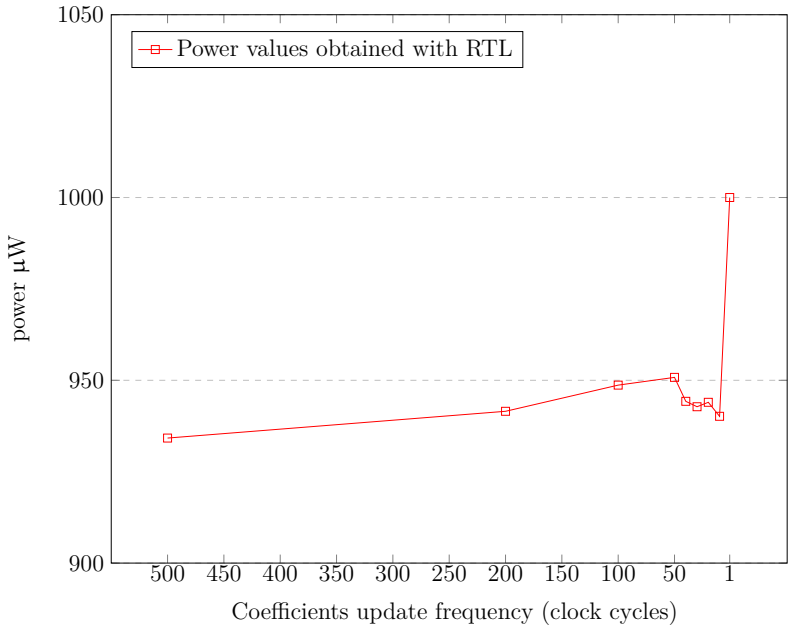

Figure 6.13: Power values considering different coefficients update frequencies for a 6-bit input and 5-bit coefficients transposed FIR filter.

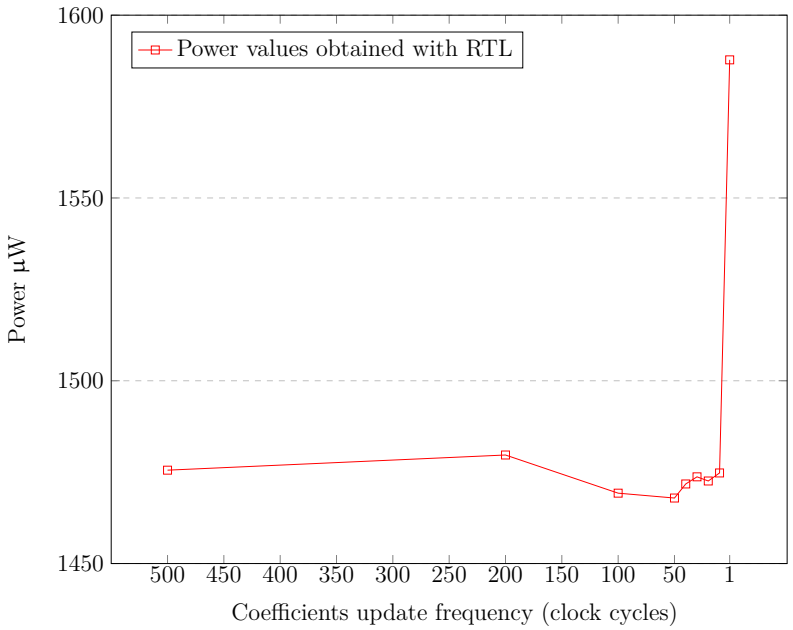

Figure 6.14: Power values considering different coefficients update frequencies for a 8-bit input and 6-bit coefficients transposed FIR filter.

6.2.5 Roll-off

When designing a filter, there are frequencies outside the desired pass-band range that are not completely rejected but only attenuated. The attenuation of these unwanted frequencies is known as roll-off. We performed different simulations considering different word-lengths for input and coefficients to understand how roll-off affects power consumption. Here we report the case only for an 19-tap transposed FIR filter with 8-bit input, considering a coefficient word-length of 6 and 10 bits respectively. The range for the roll-off spans from 0 to 1, where 0 corresponds to to a brick-wall filter while 1 to a pure raised cosine. For our simulation we considered a reasonable interval from 0.1 to 0.5. As shown in Fig. 6.15 relaxing the constraint on the roll-off usually tends to give lower power values. Interestingly, we can also notice that smaller word-length for the coefficients give more spread power values.

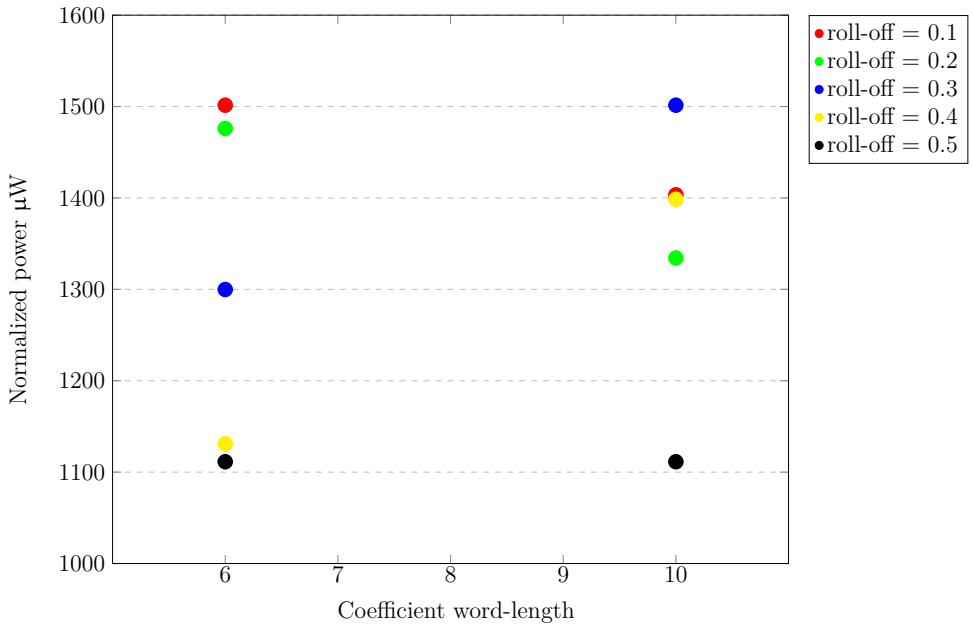

Figure 6.15: Power values for different roll-off values for a FIR filter with 8-bit input and 6-bit coefficients.

7

Conclusion

In this chapter we outline the achievements of our work and we provide some directions for future studies.

7.1 Achievement

The main reason behind this work was to provide an estimation methodology, to be used at system level, for DSP components in fiber-optic communication systems. We developed a new solution that attempts to provide power estimation for an entire filter starting from the power consumption of a basic (or atomic) component. In the case of a transposed architecture the choice of this basic component is straightforward and matches with the MAC structure. Our estimator tool can be classified as a cycle-accurate macromodel and consists of two main steps: characterization and power estimation.

Characterization models the power consumption of the basic component considering the transitions on the primary inputs pins. It is a fully automated procedure and has to be performed only once at a reasonably strict timing constraint. For this phase we devised a stopping criterion based on confidence intervals and a sequential stopping rule. The stopping criterion allows us to achieve a good trade-off between accuracy and run-time efficiency. Routines have been directly integrated in RTL Compiler using the Tcl language.

Power estimation uses an input trace, which the user should provide, and the output produced by the characterization to deliver power estimates for an entire filter. Routines have been implemented both in C++ and MATLAB.

In the model clock frequency, number of taps and operand word-lengths can be parameterized by the user. Nevertheless, at this stage the power estimation phase assumes:

- A maximum operating frequency for the filters that equals 833.33 MHz.
- A number of tap that equals 19.
- An input word-length in a range from 5 up to 10 bits.
- A coefficient word-length in a range from 5 up to 12 bits.

7.2 Future Work

The results are promising but further studies are require to obtain a complete model. In the following are reported some recommendations that the author believes may improve the model.

1. The current characterization does not take into account glitches. In section 3.2 we said that it is important to consider these spurious transitions and that ignoring this aspect may lead to loss in the relative accuracy. Nevertheless, no studies have been carried out to understand and evaluate the effect of glitches in a transposed architecture. Future works should evaluate and quantify the loss in accuracy due to glitching activity.
2. In section 6.2.3 we observed that with strict timing constrains the assumption on linear dependency between coefficient word-lengths and power does not hold anymore. In this case we proposed a LUT-based approach. However this solution has only been mentioned and not yet implemented.
3. In section 6.2.1 we reasoned about our choice to perform a characterization based on random-generated input signals. Nevertheless, we also observed that the real signal does not behave as a complete random one, but has a lower switching activity on the upper bits. As a consequence, the accuracy of the overall methodology can be improved by changing the model for the input signal during characterization.
4. The current power estimation works for a 19-tap FIR filter only. However, we should mention that the topology of the basic (or atomic) structure remains the same within continuous range of taps. Hence, it should be possible with a single characterization to cover for more tap configurations. For example, the characterization of a 19-tap FIR filter, in principle, will be enough to provide power estimation also for an FIR filter with number of taps that ranges between 17 up to 32. Unfortunately, no studies have been performed to prove this concept.
5. The power estimation phase should be extended to take into account parallel and complex FIR filters.

Bibliography

[1] S. Gupta and F. N. Najm, "Analytical model for high level power modeling of combinational and sequential circuits," in *Proceedings IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, Mar 1999, pp. 164–172.

[2] S. Heo, K. Barr, and K. Asanovic, "Reducing power density through activity migration," in *Low Power Electronics and Design, 2003. ISLPED '03. Proceedings of the 2003 International Symposium on*, Aug 2003, pp. 217–222.

[3] D. Eckerbert, "Power estimation and multi-phase clock generation for the deep submicron era," Ph.D. dissertation, Chalmers University of Technology, 2003.

[4] L. Lundberg, C. Fougstedt, P. Larsson-Edefors, P. A. Andrekson, and M. Karlsson, "Power consumption of a minimal-dsp coherent link with a polarization multiplexed pilot-tone," in *ECOC 2016; 42nd European Conference on Optical Communication*, Sept 2016, pp. 1–3.

[5] C. Fougstedt, P. Johannisson, L. Svensson, and P. Larsson-Edefors, "Dynamic equalizer power dissipation optimization," in *2016 Optical Fiber Communications Conference and Exhibition (OFC)*, March 2016, pp. 1–3.

[6] G. C. Cardarilli, A. D. Re, A. Nannarelli, and M. Re, "Power characterization of digital filters implemented on fpga," in *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No.02CH37353)*, vol. 5, 2002, pp. V–801–V–804 vol.5.

[7] Y. A. Durrani, "Accurate power estimation technique for dsp architectures," in *2009 IEEE International Symposium on Industrial Electronics*, July 2009, pp. 1123–1128.

[8] M. J. Flynn, P. Hung, and K. W. Rudd, "Deep submicron microprocessor design issues," *IEEE Micro*, vol. 19, no. 4, pp. 11–22, Jul 1999.

[9] S. Gupta and F. N. Najm, "Power macromodeling for high level power estimation," in *Proceedings of the 34th Design Automation Conference*, June 1997, pp. 365–370.

[10] M. Barocci, L. Benini, A. Bogliolo, B. Ricco, and G. D. Micheli, "Lookup table power macro-models for behavioral library components," in *Proceedings IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, Mar 1999, pp. 173–181.

[11] M. Pedram, *Power Simulation and Estimation in VLSI Circuits*, ser. Electrical

Engineering Handbook. CRC Press, Dec 1999, ch. 18, 0. [Online]. Available: http://dx.doi.org/10.1201/9781420049671.ch18

[12] D. Das and K. (e-book collection), *VLSI Design*, 2nd ed. Oxford University Press India, 2016.

[13] S. J. E. Wilton and R. Saleh, "Programmable logic ip cores in soc design: opportunities and challenges," in *Proceedings of the IEEE 2001 Custom Integrated Circuits Conference (Cat. No.01CH37169)*, 2001, pp. 63–66.

[14] M. Sgroi, A. Sangiovanni-Vincentelli, F. De Bernardinis, C. Pinello, and L. Carloni, *Platform-Based Design for Embedded Systems*, ser. Industrial Information Technology. CRC Press, Aug 2005, ch. 22, pp. 22–1–22–26, 0. [Online]. Available: http://dx.doi.org/10.1201/9781420038163.ch22

[15] H. Chang, L. Cooke, M. Hunt, G. Martin, A. J. McNelly, and L. Todd, *Surviving the SOC Revolution: A Guide to Platform-based Design.* Norwell, MA, USA: Kluwer Academic Publishers, 1999.

[16] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*, 1st ed. McGraw-Hill Higher Education, 1994.

[17] I. Griffin, "Gajski-kuhn y-chart," Online, 12 2009, available athttp://texample.net/tikz/examples/gajski-kuhn-y-chart/.

[18] V. Tiwari, J. Monteiro, and R. Patel, *Power Analysis and Optimization from Circuit to Register-Transfer Levels*, ser. Industrial Information Technology. CRC Press, Mar 2006, ch. 3, pp. 3–1–3–16, 2. [Online]. Available: http://dx.doi.org/10.1201/9781420007954.ch3

[19] F. N. Najm, "A survey of power estimation techniques in vlsi circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 446–455, Dec 1994.

[20] H. Kawauchi, M. Tsuzuki, I. Taniguchi, and M. Fukui, "An accurate rtl power estimation considering power library unevenness," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, May 2010, pp. 2618–2621.

[21] F. N. Najm, "Towards a high-level power estimation capability," in *Proceedings of the 1995 International Symposium on Low Power Design*, ser. ISLPED '95. New York, NY, USA: ACM, 1995, pp. 87–92. [Online]. Available: http://doi.acm.org/10.1145/224081.224097

[22] K.-T. Cheng and V. D. Agrawal, "An entropy measure for the complexity of multi-output boolean functions," in *Proceedings of the 27th ACM/IEEE Design Automation Conference*, ser. DAC '90. New York, NY, USA: ACM, 1990, pp. 302–305. [Online]. Available: http://doi.acm.org/10.1145/123186.123282

[23] I. B. Dhaou, N. Money, and H. Tenhunen, "Fast low-power characterization of arithmetic units in DSM CMOS," in *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No.01CH37196)*, vol. 5, 2001, pp. 531–534 vol. 5.

[24] M. Nemani and F. N. Najm, "High-level power estimation and the area com-

plexity of boolean functions," in *Proceedings of 1996 International Symposium on Low Power Electronics and Design*, Aug 1996, pp. 329–334.

[25] ——, "High-level area and power estimation for vlsi circuits," in *1997 Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*, Nov 1997, pp. 114–119.

[26] Y. A. Durrani and T. Riesgo, "Power estimation for intellectual property-based digital systems at the architectural level," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 26, no. 3, pp. 287–295, Sep. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.jksuci.2014.03.005

[27] S. Gupta and F. N. Najm, "Energy and peak-current per-cycle estimation at rtl," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 4, pp. 525–537, Aug 2003.

[28] H. Mehta, R. M. Owens, and M. J. Irwin, "Energy characterization based on clustering," in *33rd Design Automation Conference Proceedings, 1996*, Jun 1996, pp. 702–707.

[29] A. Bogliolo, L. Benini, and G. De Micheli, "Regression-based rtl power modeling," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 5, no. 3, pp. 337–372, Jul. 2000. [Online]. Available: http://doi.acm.org/10.1145/348019.348081

[30] N. S. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan, "Leakage current: Moore's law meets static power," *Computer*, vol. 36, no. 12, pp. 68–75, Dec. 2003. [Online]. Available: http://dx.doi.org/10.1109/MC.2003.1250885

[31] H. J.M. Veendrick, *Very Large Scale Integration (VLSI) and ASICs*. Cham: Springer International Publishing, 2017, pp. 321–380. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-47597-4_7

[32] A. P. Chandrakasan, W. J. Bowhill, and F. Fox, *Design of High-Performance Microprocessor Circuits*, 1st ed. Wiley-IEEE Press, 2000.

[33] J. Smith, *Introduction to Digital Filters: With Audio Applications*, ser. Music signal processing series. W3K, 2008. [Online]. Available: https://books.google.se/books?id=pC1iCQUAsHEC

[34] V. Mauer, "Designing filters for high performance," Online, 12 2015, availabe athttps://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/wp/wp-01260-stratix10-designing-filters-for-high-performance.pdf.

[35] *Cadence® Encounter® RTL Compiler, v. 14.11.000*, Cadence Design Systems, Inc., 2013.

[36] G. Jochens, L. Kruse, E. Schmidt, and W. Nebel, "A new parameterizable power macro-model for datapath components," in *Design, Automation and Test in Europe Conference and Exhibition, 1999. Proceedings (Cat. No. PR00078)*, March 1999, pp. 29–36.

[37] X. Liu and M. C. Papaefthymiou, "A markov chain sequence generator for power macromodeling," in *IEEE/ACM International Conference on Computer*

Aided Design, 2002. ICCAD 2002., Nov 2002, pp. 404–411.

A

Appendix 1

A.1 Suggested flow in RC-LP engine

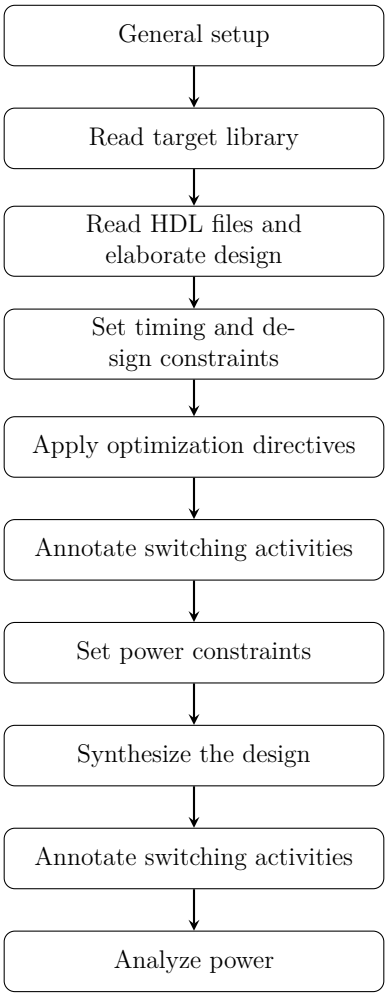

Figure A.1: Suggested flow for low power features