9-26-2003

High Level Techniques for Leakage Power Estimation andOptimization in VLSI ASICs

Chandramouli Gopalakrishnan
University of South Florida

High Level Techniques for Leakage Power Estimation and Optimization in VLSI ASICs

by

Chandramouli Gopalakrishnan

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Srinivas Katkoori, Ph.D.
N. Ranganathan, Ph.D.
Murali Varanasi, Ph.D.
Wilfrido Moreno, Ph.D.
A.N.V. Rao, Ph.D.

Date of Approval:
September 26, 2003

Keywords: clique partitioning, power management, 0-1 knapsack formulation,
multi-cycling, vhdl model, tabu search

DEDICATION

To my dear grandfather,

late Shri T. R. Thiagarajan

ACKNOWLEDGEMENTS

I would like to thank Dr. Katkoori profusely for his continuous support and encouragement throughout the course of this research. The long hours of discussion and brainstorming helped a lot in guiding me through this dissertation. His easy availability was a major asset to me. I would like to thank Dr. Ranga, Dr. Varanasi, Dr. Moreno, and Dr. Rao for being on my committee. I would also like to acknowledge all the help provided by Daniel Prieto and his team with the Computer Science Tech Support whenever problems arose. I would like to thank all my friends in the VCAPP group and outside, who have been a constant source of encouragement. Last but not the least, I would like to thank my parents and family for their constant support and encouragement in all my endeavours.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

High Level Techniques for Leakage Power Estimation and Optimization in VLSI ASICs

Chandramouli Gopalakrishnan

ABSTRACT

As technology scales down and CMOS circuits are powered by lower supply voltages, standby leakage current becomes significant. A behavioral level framework for the synthesis of data-paths with low leakage power is presented. There has been minimal work done on the behavioral synthesis of low leakage datapaths. We present a fast architectural simulator for leakage (FASL) to estimate the leakage power dissipated by a system described hierarchically in VHDL. FASL uses a leakage power model embedded into VHDL leafcells. These leafcells are characterized for leakage accurately using HSPICE. We present results which show that FASL measures leakage power significantly faster than HSPICE, with less than a 5% loss in accuracy, compared to HSPICE. We present a comprehensive framework for synthesizing low leakage power data-paths using a parameterized Multi-threshold CMOS (MTCMOS) component library. The component library has been characterized for leakage power and delay as a function of sleep transistor width. We propose four techniques for minimization of leakage power during behavioral synthesis: (1) leakage power management using MTCMOS modules; (2) an allocation and binding algorithm for low leakage based on clique partitioning; (3) selective binding to MTCMOS technology, allowing the designer to have control over the area overhead; and (4) a performance recovery technique based on multi-cycling and introduction of slack, to alleviate the loss in performance attributed to the introduction of MTCMOS modules in the data-path. Finally, we propose two iterative search based techniques, based on Tabu search, to synthesize low leakage data-paths. The first technique searches for low leakage scheduling options. The second technique simul-

taneously searches for a low leakage schedule and binding. It is shown that the latter technique of unified search is more robust. The quality of results generated bytabu-based technique are superior to those generated by simulated annealing (SA) search technique.

CHAPTER 1

INTRODUCTION

The complexity of Very Large Scale Integrated (VLSI) circuits has increased progressively over the past decade. Significant advances in fabrication technology has led to packing more than a billion transistors on a chip. Design of chips with such a heavy packing density has increased the importance of design automation significantly. VLSI CAD is one of the most rapidly advancing fields in the present hardware arena.

Current fabrication techniques allow a feature size as small as 90 nanometers. Technologies with feature sizes less than a 100 nanometers has come to be known as Deep sub micron (DSM) technologies. As we descend into DSM technologies, researchers are faced with several new problems that have never been encountered before. Some of the important problems are leakage power, cross-talk, and signal-integrity.

1.1 Increasing concern about leakage power

Until the last decade, area and performance were the two main driving factors in the semiconductor industry. In more recent times, power dissipation has gained importance. The power dissipated by a MOS device is given by Eqn. 1.1. Until recent years, the static leakage power has been assumed to be negligible. This assumption is proving to be no longer true.

$$P_{total} = P_{dynamic} + P_{shortcircuit} + P_{leakage} \tag{1.1}$$

In current technologies, the dominant component of power is the dynamic power. The dynamic power has a quadratic relationship to the supply voltage, as given by Eqn. 1.2.

$$P_{dyn} = \sum_i \alpha.C_L.V_{dd}^2.f \tag{1.2}$$

where α is the switching activity at node i, C_L is the load capacitance at node i, V_{dd} is the supply voltage, and f is the clock frequency.

The simplest and most effective way to reduce the dynamic power in a circuit is to reduce the supply voltage. As we approach voltages close to 1V (closer to the threshold voltage of the transistors used), the performance of the circuit starts to degrade. It has been shown that the propagation delay through a transistor is:

$$T_d \propto \frac{C_L.V_{dd}}{K.(V_{dd} - V_T)^\alpha} \tag{1.3}$$

where C_L is the load capacitance, K is a factor that depends on the process and the gate size, V_T is the threshold voltage of the transistor, and α takes any value between one and two depending on channel length [2].

Reducing the threshold voltage of the transistor improves performance at low supply voltages. This results in increased leakage current given by:

$$I_{sub} = A.e^{\frac{q}{n'kT}.(V_G - V_S - V_T - \gamma' V_S + \eta V_{DS})}.(1 - e^{\frac{-qV_{DS}}{kT}}) \tag{1.4}$$

where $A = \mu_0.C_{ox}.\frac{W_{eff}}{L_{eff}}.(\frac{kT}{q})^2.e^{1.8}$, C_{ox} is the gate oxide capacitance per unit area. μ_0 is the zero bias mobility. n' is the sub-threshold swing coefficient of the transistor. V_T is the zero bias threshold voltage. γ' is the linearized body effect coefficient and η is the Drain Induced Barrier Lowering (DIBL) coefficient [3].

We notice the exponential relationship of the leakage current to the change in the threshold voltage. As we reduce feature sizes, the supply voltages would also be scaled down further. Fig. 1.1 shows the alarming projections for the increase in leakage power by Intel. The International Technology Roadmap for Semiconductors (ITRS) 2001 [4] describes the leakage power reduction as one of the key design challenges.

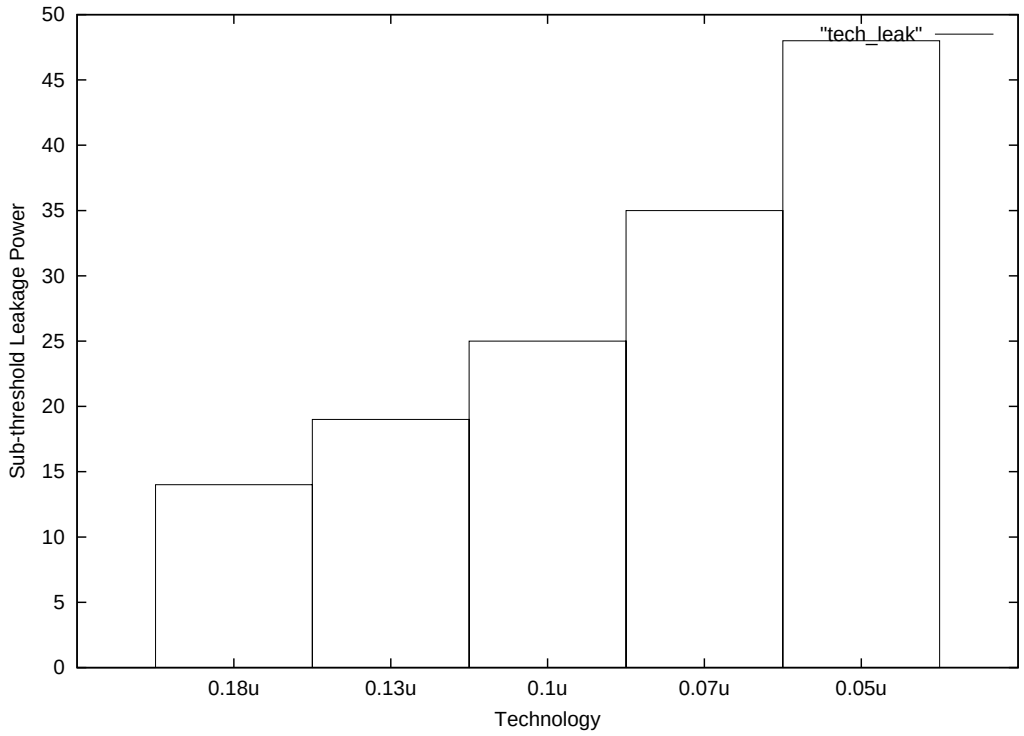


Figure 1.1. Alarming rate of increase in leakage power (Source: Borkar.S [1])

1.2 Behavioral synthesis

Behavioral Synthesis is the process of synthesizing an RT-Level design from a behavioral description. It is also commonly referred to as high level synthesis (HLS). The input to a high level synthesis system is usually a data-flow graph (DFG). An example DFG is shown in Fig. 1.2(a). This process of synthesizing RT-Level design consists of three inter-related tasks viz., scheduling, allocation, and binding. Scheduling is the process of assigning a time-stamp to each operation in the DFG (see Fig. 1.2(b)). The high level synthesis system then *allocates* the required number of resources to implement this schedule. The schedule shown in Fig. 1.2(b) requires two multipliers. Binding is the process of assigning each operation in the DFG to a specific instance. A possible binding for the scheduled DFG shown in Fig. 1.2 (b) is shown in Table 1.2. These three tasks may be performed sequentially in any order or may be performed simultaneously.

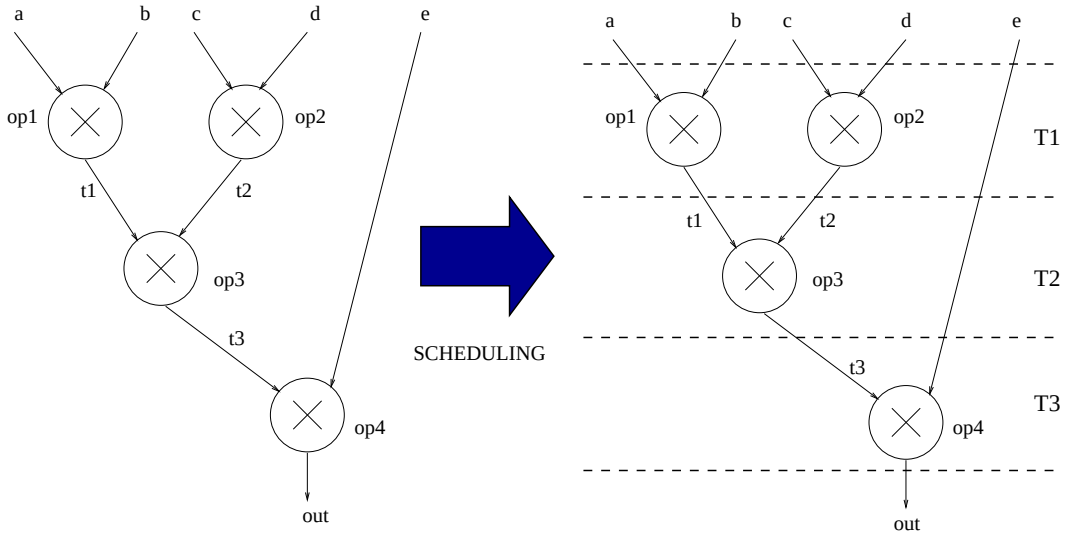


Figure 1.2. An example data-flow graph and a schedule

Table 1.1. A possible binding for the scheduled DFG shown in Fig. 1.2(b)

Operation	Multiplier Instance
op1	1
op2	2
op3	1
op4	1

1.3 Leakage optimization and estimation during behavioral synthesis

Decisions made during these three tasks however influence the three parameters of design - area, performance, and power. This influence has been the motivation factor for significant research done in the area of power optimization during behavioral synthesis [5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. There are broadly two objectives that these techniques attempt to minimize. Firstly, optimization is performed during the scheduling and allocation phases, to minimize the number of modules, resulting in smaller capacitance [7]. Secondly, optimization can be done during binding, to minimize the transition activity. Bit level statistics are utilized in these kind of methods to evaluate the transition activity.

In addition, minimizing the number of instances reduces the leakage power. In this dissertation, we also propose the use of a multi-threshold CMOS (MTCMOS) modules, that

have the ability to be switched ON and OFF. During high level synthesis, power management techniques can be implemented to switch OFF modules hence minimizing leakage power. An allocation and binding algorithm based on clique parititioning is proposed for low leakage power. A selective binding algorithm is proposed to bind a subset of modules to MTCMOS technology.

An equally important problem of estimating leakage power at higher levels of abstraction is also addressed in this dissertation. Estimating the leakage power during the early stages of design, can guide the designer towards low leakage solutions. As we ascend to higher levels of design abstraction, the time required for power estimation reduces significantly at the cost of decreased accuracy. We propose a simulation based leakage estimation approach, which speeds up simulation time by orders of magnitude, compared to HSPICE. The compromise in accuracy is under 5%, which is affordable.

1.4 Automated search for a low leakage data-path

The number of schedules and binding alternatives for a given data-flow graph is significantly large. In literature, several solution search strategies have been proposed for searching large solution space[15, 16]. Tabu search is a recently proposed search technique by Glover and Laguna [17] which efficiently searches large solution spaces. The tabu search technique deemphasizes randomness in the search. It is based on the human tendency to avoid paths (or solutions) that have been visited before. The solutions that have been visited are considered *tabu*. This prevents cycling within solutions. The tabu search has been applied to high level synthesis[18], to search for data-paths with minimum area. In the case of our search for low leakage data-paths, the cost function can be modelled as the average leakage power dissipated by the data-path. Two tabu-based approaches have been proposed in this dissertation. The first approach searches the large solution space of schedules. The second approach models the schedule-binding as a two dimensional placement problem. This space is then searched for a low leakage data-path. The RT-level leakage power estimator is used in both approaches to evaluate the cost function.

1.5 Contributions

The key contributions of this work are:

- A High Level Synthesis framework for synthesizing MTCMOS data-paths for low leakage power
- A resource allocation algorithm based on clique partitioning, which maximizes the idle times of allocated resources
- An MTCMOS component library characterized with respect to leakage power and delay, as a function of sleep transistor width
- A selective binding approach to do strategic mapping to MTCMOS for a given area overhead, maximizing leakage power savings
- A performance recovery technique based on multicycling and introduction of slack, to alleviate the performance loss due to introduction of MTCMOS modules
- A simulatable leakage power model in VHDL
- A fast and accurate alternative to HSPICE for measuring leakage power for designs, described in hierarchical VHDL
- An effective decision space exploration tool based on Tabu Search, for the synthesis of low leakage power data-paths

1.6 Organization of the dissertation

Chapter 2 provides some background on the sources of leakage power, and discusses related research that has been done in the area of leakage power estimation and minimization. The chapter also discusses the MTCMOS circuit design style in detail.

Chapter 3 outlines the framework for synthesizing low leakage data-paths. The chapter introduces the Automatic Design Instantiation (AUDI) high level synthesis system, which

has been used as the vehicle of research, for implementing the approaches proposed in this dissertation. The chapter also details the MTCMOS component library that has been developed. We also present an extensive area-delay tradeoff analysis of the modules in the library.

Chapter 4 proposes allocation and binding algorithms for synthesizing low leakage power data-paths. The chapter also proposes a power management technique which involves effective use of MTCMOS modules to reduce the leakage power dissipated in a data-path. A resource allocation algorithm based on clique partitioning is presented. Results are presented for three data-path intensive DSP benchmarks. Leakage power reduction of upto 28.52% in 180nm and upto 42.63% in 100nm technology nodes respectively is achieved with an area overhead of 17.9% and 18% in 180nm and 100nm respectively. The introduction of the MTCMOS modules leads to a 40% increase in execution time. We present a performance recovery technique, based on multicycling and slack introduction, to alleviate the performance loss due to introduction of MTCMOS modules. Upon performance recovery, the execution time increases only by 16.8%. A selective binding algorithm is proposed to overcome the area overhead imposed due to MTCMOS technology. Area-leakage tradeoff analysis has been carried out and leakage power reduction for various permissible area overheads is reported.

Chapter 5 presents a fast architectural leakage simulator for hierarchically described RT-Level designs. The tool provides an accurate estimate of the leakage power dissipated by a design. Exhaustive and accurate leakage characterization is done at the leaf-cell level. The leakage power dissipated by an RT-Level design is obtained by fast simulation using a VHDL simulator. The characterization has been done very accurately taking into account the transient characteristics between input vector changes. Estimated leakage energy has been compared to leakage energy measured using HSPICE for different vector sets. Results have been presented for three datapath-intensive designs and 20 MCNC random logic benchmarks. The average error percentage is 1.38% in the case of data-paths and 2.11%

in random logic. The estimation time compared to HSPICE is lowered by several orders of magnitude.

Chapter 6 presents two search-based approaches, based on the tabu search technique, to explore the solution space for low leakage power data-paths. The first approach explores different schedules using the Tabu search technique. The second approach simultaneously searches for schedules *and* mappings that dissipate low leakage power. We show that our approaches find low leakage data-paths which dissipate upto 40% lesser leakage power than data-paths synthesized using traditional heuristics. We also observe that the tabu search technique explores the search space more efficiently than a competing technique based on simulated annealing. We observe that the tabu search finds significantly better solutions at the cost of a moderate increase in computation time.

Chapter 7 summarizes the dissertation with a brief summary of the high level leakage power estimation and minimization algorithms that have been proposed. The chapter also provides directions for future work in this research area.

CHAPTER 2

LEAKAGE POWER ESTIMATION AND OPTIMIZATION: BACKGROUND AND RELATED WORK

Section 2.1 presents the various sources of leakage power in CMOS circuits. Section 2.2 discusses the recent research that has been done in leakage power optimization at various levels of design hierarchy. Section 2.3 presents recent work done on leakage power estimation. Section 2.5 outlines the techniques proposed in literature for RT-Level power estimation. Section 2.6 describes simulated annealing and tabu search, which are two commonly used design space search techniques. Section 2.7 summarizes the current state-of-the-art in leakage power estimation and optimization.

2.1 Sources of leakage power in CMOS

The static leakage power is dissipated due to five main reasons [19]. The sources are shown diagrammatically in Fig. 2.1.

- PN reverse bias junction leakage: The leakage current that is induced due to the reverse bias junctions formed near the edges of the depletion region. This component of leakage power is negligible.

- Sub-threshold leakage: The sub-threshold leakage current is the weak inversion current between the drain and source of the transistor, when the transistor is `OFF`, ie. when the gate voltage is less than the threshold voltage of the transistor. The sub-threshold leakage is exponentially related to the threshold voltage, and is given by:

$$I_{sub} = A.e^{\frac{q}{n'kT}.(V_G - V_S - V_T - \gamma' V_S + \eta V_{DS})}.(1 - e^{\frac{-qV_{DS}}{kT}}) \tag{2.1}$$

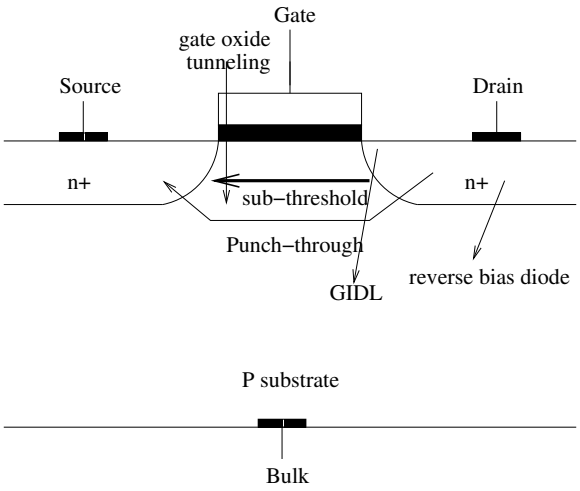


Figure 2.1. Sources of leakage power in MOS transistors

where $A = \mu_0.C_{ox}.\frac{W_{eff}}{L_{eff}}.\left(\frac{kT}{q}\right)^2.e^{1.8}$,$C_{ox}$ is the gate oxide capacitance per unit area. μ_0 is the zero bias mobility. n' is the sub-threshold swing coefficient of the transistor. V_T is the zero bias threshold voltage. γ' is the linearized body effect coefficient and η is the Drain Induced Barrier Lowering (DIBL) coefficient [3].

- Gate Induced Drain Leakage (GIDL): The GIDL leakage occurs due to the high electric field between the gate and the drain induces carriers into the substrate. This component of leakage current is very small in current technologies.

- Punch-through: This phenomenon occurs when the drain voltage is high, and the drain and source depletion regions make contact. During this phenomenon, the gate loses control of the channel control.

- Gate Oxide tunneling: This component arises due to the high electric field over the gate oxide. Gate oxide tunneling becomes significant only when the oxide thickness is less than 2-3nm.

The dominant component of the leakage power in low voltage low power circuits is the sub-threshold leakage [19]. In the remainder of the dissertation, the term leakage current refers to the sub-threshold leakage current, and leakage power refers to the power dissipated

due to this leakage current. Leakage power is the power dissipated when the transistor is OFF, and hence is also referred to as standby power.

2.2 Leakage power optimization

Several leakage power minimization techniques have been proposed in the recent work [3]. These techniques can be broadly classified into two categories - CAD-driven techniques and technology-driven techniques. CAD-driven techniques involve circuit styles and design methodologies to minimize leakage power dissipated. On the other hand, technology-driven techniques involve developing newer technologies on silicon such as Silion-On-Insulator (SOI) to minimize leakage power dissipation. This section outlines some of the CAD-driven techniques that have been proposed in recent years. The MTCMOS circuit design style has been discussed in detail in Section 2.2.3.

2.2.1 Input vector control

Halter and Najm [20] established the static dependency of input vectors to the leakage current, in a CMOS circuit. The authors target designs, where sub-circuits can be identified to be put into "standby mode." The authors identify vectors that would put these sub-circuits into a "low leakage standby state," when idle. The authors determine low leakage input vectors using a statistical sampling technique. The authors determine the number of random samples n, to determine a lower bound on leakage with a certain level of confidence α. To avoid long simulation times in HSPICE, the authors propose a characterized gate library. Leakage current dissipated by gates are then calculated using a table look-up, which matches the leakage current to the current input to the gate. The authors also propose low area-low power jamb latches to aid in forcing these sub-circuits to the low leakage standby state. These modified jamb latches have the capability to restore the state of the sub-circuit to the original state. The technique yields an average of 54% reduction in leakage power.

In a related work, Naidu and Jacobs propose an ILP based technique to determine the low leakage input vector. This technique provides an accurate estimate, but explodes and fails to provide a solution, as the circuit size increases.

Chen *et al.* [21] estimate standby leakage power dissipation in a CMOS circuit using a Genetic Algorithm (GA) which models the transistor stacks in the circuit. The algorithm is also used to determine the input vector that results in the lowest leakage power. During standby operation of the circuit, this *minimum* vector is forced onto the primary inputs to reduce leakage power. For sequential circuits, special memory logic is required to restore state variables.

2.2.2 Dual-Vt techniques

Wei *et al.* [22] propose a methodology in which the circuit is synthesized with the logic gates in the critical path made of low-V_T transistors, maintaining the performance at low supply voltages. The authors have developed an algorithm which selectively assigns the remainder of the gates with either high V_T or low V_T transistors, based on the effect on the overall leakage power of the circuit.

Wang and Vrudhula [23, 24], propose techniques to reduce standby leakage in delay-constrained environments. The authors describe three techniques that operate on gate level netlists. These algorithms assign one of two available threshold voltages (high V_t and low V_t), to each transistor, so that standby power is minimized without violating delay constraints specified by the user. The approach initially models the netlist as a directed acyclic graph (DAG), where each vertex represents a CMOS transistor network, that realizes a gate, and the edges in the graph determine the connections in the circuit. The transistors within each network are assigned to the same V_t. The effect of assigning V_t to these networks on the delay and the standby power are appended to the DAG, in the form of edge weights. The authors then present the three algorithms which are based on min-cut and max-cut algorithms to determine the V_t assignments. The technique yields upto a 67% reduction in standby power.

Sirichotiyakul *et al.* [25] present DUET, an accurate leakage estimation and optimization tool for dual-V_t circuits. The authors propose the concept of dominant leakage states, and the use of state probabilities to optimize standby power. The authors present a combination of graph reduction techniques and non-linear simulation that result in three to four orders of speed up compared to HSPICE simulations. Results are presented for medium sized random logic circuits. The tool achieves 1/3 to 1/6 reduction in standby current at the cost of only 20% reduction in performance.

2.2.3 MTCMOS design style

Leakage energy is dissipated when sub-threshold current flows through `OFF` transistor (transistor in cutoff region) stacks between supply voltage and ground rails. The MTCMOS circuit design style as proposed by Mutoh [26] minimizes the leakage power dissipated by a CMOS circuit by isolating the circuit from the supply voltage and the ground rails during idle periods. Two sleep transistors are introduced between the circuit and the supply voltage and the ground rails (see Fig. 2.2). The sleep transistors are controlled by complementary *sleep* and $\overline{sleep}$ signals.

When the sleep transistors are turned `OFF`, the circuit is isolated from the supply rails, breaking all the leakage current paths. During this time, the only leakage current flowing in the circuit is due to the sleep transistors which are in the `OFF` state. The main disadvantages of the MTCMOS circuit design style are the area overhead and performance loss due to the sleep transistors.

The feasibility of fabricating such a dual-threshold logic on a single chip was verified by Mutoh *et al.* [26] , by implementing a PLL-LSI chip in $0.5\mu m$ technology. The authors have more recently developed a 1V DSP chip for mobile phone applications [27].

Kao and Chandrakasan [2] address several issues in multi-threshold voltage circuits, including sizing of the sleep transistors for high performance [28]. They propose a hierarchical sizing scheme for the sleep transistors and a novel modification of MTCMOS called

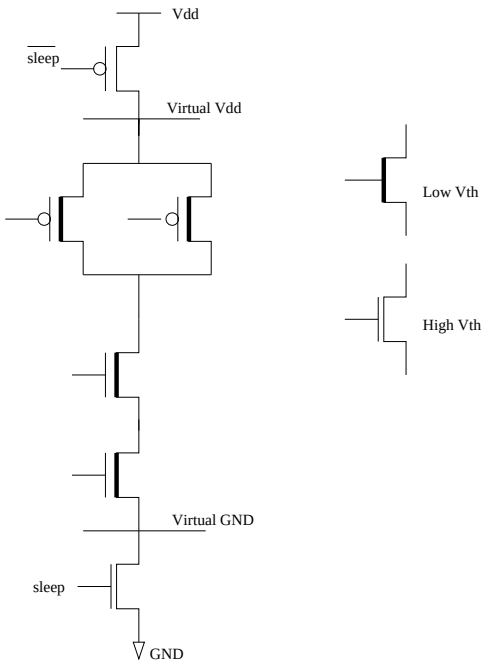


Figure 2.2. MTCMOS NAND gate

dual-V_T domino logic. Chandrakasan *et al.* [29] discuss the effectiveness of MTCMOS in event driven computation such as X-Terminals.

2.2.4 Stacking effect

In a recent work, Hanchate and Ranganathan [30] propose a method of leakage reduction by introduction of an additional transistor in a stack of transistors, such that the functionality is not altered. The authors connect the gate and the source of the added transistor, making the transistor invisible in terms of functionality. The introduction of the transistor however, increases the stacking effect, thereby reducing the leakage current flow considerably. The authors also present extensive transient analysis. This method introduces minimal area overhead.

2.2.5 Leakage minimization at higher levels of abstraction

Khouri *et al.* [31] present an algorithm to analyze and optimize leakage power dissipation during behavioral synthesis. The authors use pre-characterized device level module libraries, consisting of low V_T and high V_T modules. The data flow graph (DFG) is analyzed and modules that have significant idle time are identified. These modules are bound to high V_T resources. Modules in the critical path are assigned low V_T modules to achieve better performance.

Powell *et al.* [32, 33] introduce a circuit technique called Gated-Vdd for the design of cache memories. The authors address the minimization of the high amount of leakage in cache memories. In Gated-Vdd, a sleep transistor is introduced in the path between the supply voltage and the V_{dd} rails of the SRAM cells. Unused sections of the caches are powered off thus reducing the leakage power. In a more recent work, Agarwal *et al.* [34] propose a similar architecture for high-performance caches with data retention capabilities (DRG Caches), with significant reduction in leakage. Roy and his research group have proposed several novel techniques to reduce leakage in cache memories and CMOS circuits [22, 35, 36, 19].

A recent work addressed by Tadas and Chakrabarti [37], presents architectural approaches for minimizing leakage energy in caches. The authors propose techniques for reducing leakage energy with less than 1% performance overhead. Another recent work by Zhang *et al.* [38] exploits VLIW schedule slacks for leakage energy reduction. Anis *et al.* [39] present an efficient gate clustering technique to apply MTCMOS to combinational circuits. A comprehensive survey of leakage minimization techniques is provided by Roy [3] and Duarte *et al.* [40]. Extensive experimentation on MTCMOS technology is provided in [41].

2.3 Leakage power estimation

Until recently, the dynamic power dissipation had been assumed to approximate the total power in a design. However, with the advent of deep sub-micron technologies, leakage

power is becoming significant. The most accurate estimate of the leakage power dissipated can be obtained using simulation in HSPICE. This method can be applied to estimation of leakage power in cache memories and small circuits [42, 40]. This method however is extremely slow. To alleviate this, several models have been proposed for leakage estimation at gate-level and transistor levels of abstraction [20, 21].

Several leakage power optimization techniques suggested in literature [19, 21] use the stacking model of transistors to estimate the leakage power in a design. The stacking model involves calibrating a single transistor for leakage power and applying a constant multiplier to reduce the leakage when more than one transistor is connected in series. This linear scaling factor is not accurate. Other work [23, 43, 44] use the leakage current equation (Eqn. 2.1) for estimating the leakage power. Both the methods discussed would be accurate only after quiescent state has been reached. For large circuits, this may be upto several milliseconds [3]. For DSP designs similar to the examples presented in this dissertation, the clock periods are in the order of tens of nanoseconds. For such designs, ignoring the transient leakage dissipation would not provide accurate results. Results in favor of this argument are presented in Section 5.4.

Johnson *et al.* [45] propose an accurate model relating the topology of transistor stacks to the leakage power dissipated. The authors present a genetic algorithm based approach to determine the upper and lower bounds on the leakage power dissipated by a circuit. The leakage current model, based on transistor stacks has been verified with HSPICE. The estimation is accurate, and the estimation times vary from 4.5s to 327.9s for the MCNC and ISCAS benchmarks presented. The results presented in [45] give the maximum and minimum leakage bounds in the benchmark circuits.

Kumar and Ravikumar [44] propose a model to estimate the leakage power in large system level designs. The authors present a linear regression model relating the complexity of a module to the leakage power dissipated. The model parameters are obtained by performing extensive simulations on MCNC benchmarks. The authors have validated this

model by estimating the leakage power dissipated by a DSP core. The authors report error percentages in the range of 1.1% to 12.5% for the various components in the core.

Butts and Sohi [43] propose a static power model for architects, which is a high level estimation model. The authors propose a simplistic model in terms of the number of transistors (N), a design dependent parameter (k_{des}), and a technology dependent parameter (I_{leak}). The authors characterize the leakage energy dissipated in a technology and using regression methods, form an equation involving the above parameters. The paper discusses in detail the technology dependent and the design related parameters in the leakage current equation.

2.4 RT-Level power minimization

There has been considerable work done int he minimization of dynamic power during behavioral synthesis. For the sake of completeness, we mention a few salient ones here. Raghunathan and Jha [46] discuss methods of minimizing pwoer during the behavioral synthesis of data-dominated circuits. The authors also provide controller optimization techniques to minimize power. Potkonjak and Srivastava [47] present techniques for the behavioral synthesis of high throughput, low cost, and low power ASICs. Kumar *et al* [48], present a profile driven synthesis system for synthesizing low power VLSI circuits. Shiue and Chakrabarthy [49] propose an ILP based scheduling and allocation algorithm for low power. In a more recent work, Mohanty and Ranganathan [50], present a framework for transient power reduction during high level synthesis.

2.5 RT-Level power estimation

Typically, a VHDL structural description of a data-path contains instantiations of various `components` from a precompiled module library, connected using `signals`. The module library contains storage components such as registers, functional units (FUs) such as adders and multipliers, and interconnect components such as multiplexers and buses. The components are described behaviorally with `generic` parameters such as width. By terming the

components as described hierarchically, we mean that, there exist two `architectures` for each component - the 1-bit architecture and n-bit architecture. The 1-bit architecture is described either completely behaviorally or may contain one or more leaf-cells. The n-bit architecture is a scaled n-bit implementation, using instantiations of 1-bit modules using the `generate` statement.

Katkoori and Vemuri [51] present a dynamic power simulator for VHDL structural descriptions. Dynamic power dissipated by data-paths, which are generated using High Level Synthesis (HLS) tools can be estimated using this simulator. The VHDL structural description consists of modules from a hierarchically described VHDL module library. Characterization for switched capacitance is done exhaustively only on the small leaf-cells. This information is incorporated into the VHDL module library. The switched capacitance is accumulated during simulation of the VHDL design using a simulator. This results in a highly accurate and fast power estimator.

Ravi, Raghunathan, and Chakradhar [52] present an efficient RT-level power estimator for large industrial design that is conceptually similar to [51]. The authors however propose several enhancements to improve the efficiency of RT-level simulation for power. The authors propose a suite of acceleration techniques such as transformation of the RT-level description and partitioned sampling.

2.6 Search based techniques

Search based techniques such as simulated annealing were originally applied to a large class of industrial engineering problems, such as job shop scheduling, flexible manufacturing systems, etc. Kirkpatrick *et al.,* [53] was one of the first to propose a simulated annealing approach to solve CAD related problems. Newton and Devadas [54] model the simultaneous scheduling and binding problem in the form of a simulated annealing problem. A relatively new search space technique that is proving to be very efficient is the Tabu Search technique proposed by Glover and Laguna [17]. The tabu search de-emphasizes randomness in the search. The search is more organized and *human-like.* Amellal and Kaminska [18] describe

a method to synthesize digital systems using the tabu based search(TASS). Section 2.6.1 details the simulated annealing algorithm, and its application to the simultaneous scheduling and binding [54]. The tabu search and its application to data-flow graph scheduling (TASS) is described in Section 2.6.2.

2.6.1 Simulated annealing

Simulated annealing is a meta-heuristic that mimics the annealing process of metals. The metal is heated to a high temperature, which increases the energy in the atoms of the metal, enough to break from their bonds and become free to move. The metal is then cooled in a controlled manner, until the metal crystallizes and equilibrium is reached. During the cooling process, also called the *cooling schedule*, the free energy of the metal is minimized. As explained by Kirkpatrick *et al.* [53], a solution to a combinatorial problem is analogous to a state of the metal, and the cost of the solution is equivalent to the energy of the metal in that state. The simulated annealing procedure is sketched in Fig. 2.3. An improving move, which is a move resulting in a lower cost (energy) (Lines 17-18) is accepted. A non-improving move is accepted with a probability which is related to the current temperature (Line 20) and a random number generated (Line 21). Such random moves are attempted for a specifed number of iterations (Line 6). After each set of iterations, the temperature parameter is updated (Line 12). More non-improving moves are accepted at high temperatures. The number of non-improving moves accepted reduces as temperature decreases. Accepting non-improving moves is called *hill climbing*. These non-improving moves help in escaping out of local minima.

Newton and Devadas [54] map the scheduling and binding problem onto a two dimensional placement problem. Scheduling an operation in a DFG is equivalent to placing it in a row, which corresponds to a timestep. The binding problem reduces to placing the operation to one of the columns, which correspond to a possible instance. Fig. 2.4 shows an example of such a matrix. Moving an operation to a different location within a matrix,

```
1  Select starting solution x ⇐ x_start
2  Set best solution x_best ⇐ x_start
3  Set best cost best_cost ⇐ cost(x_best)
4  Set temperature = T_0 (Initial temperature)
5  while (cost is changing)
6     for (a certain number of iterations)
7        generate_new_state(j)
8        if (accept(cost(j),cost(x),T)) then
9           x = j
10       endif
11    endfor
12    T = update(T)
13 endwhile
14
15 procedure accept(cost(j),cost(i),T)
16    δcost = cost(j)-cost(i)
17    if (δcost < 0) then
18       return 1
19    else
20       Y = exp(-δcost/T)
21       R = random(0,1)
22       if (R < Y) then
23          return(1)
24       else
25          return(0)
26       endif
27    endif
28 end procedure
```

Figure 2.3. Simulated annealing procedure

shifts the current solution to a new solution. The authors propose three different classes of moves:

- Interchange: Interchange the positions of two operations in the matrix
- Displacement: Find a new position for an operation in the matrix
- Swap-inputs: If the operation is a symmetric operation (such as `ADD`, `MULT`), interchange two inputs

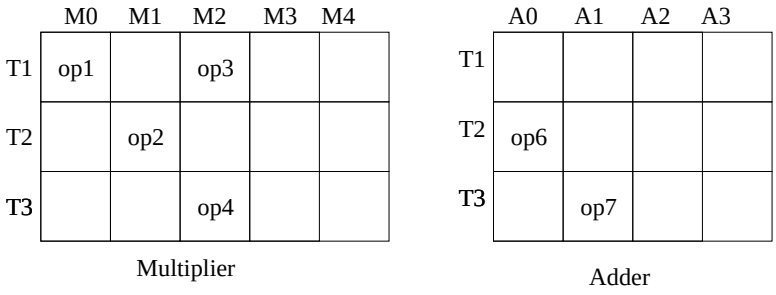


Figure 2.4. Placement of operations in a 2-dimensional space-time matrix

Interchange moves produce different bindings. Displacement moves change the schedule. The swap-input moves result in different sharing, and hence alter the number of interconnect units (such as multiplexers). The choice of the move type is dependent on the temperature, and the generation of two random numbers. At higher temperatures, the moves are more flexible. At lower temperatures, moves are made with adjoining neighbor cells only. The cost of each solution is a function of the total hardware area and the execution time. Hence, the search progresses towards low area and minimum latency solutions.

The authors provide a comprehensive framework for synthesizing data-paths using the simulated annealing framework, resulting in reduced area and minimum latency. The main disadvantage of the simulated annealing approach is the inability to recognize visited solutions. This can result in cycling, due to the random nature of move choice. This disadvantage is the main motivation behind the tabu search technique proposed by Glover and Laguna [17].

2.6.2 Tabu search

Tabu search is a meta-heuristic proposed by Glover and Laguna [17]. The key difference between tabu search and simulated annealing lies in choosing a move. In simulated annealing, the solution space is searched using randomized moves. The moves may be dependent on the temperature parameter, so that certain moves may be carried out during the start of the search and certain others towards the end of the search. Hill climbing moves, or

moves with negative cost gain, are accepted probabilistically. In the tabu search technique, the solution space is searched more strategically. The technique tries to *de-emphasize* randomness used in the search. The search is based on the human tendency to stay away from a path that has been trodden before. A single run of the tabu search algorithm is shown in Fig. 2.5.

```
1  Select starting solution x ⇐ x_start
2  Set best solution x_best ⇐ x_start
3  Set best cost best_cost ⇐ cost(x_best)
4  Initialize tabu lists and Aspiration level(AL)
5  for (fixed number of iterations)
6    Generate neighbor solutions for x ∈ N(x)
7    Find best x* in N(x) for which cost is minimum
8    if (move x ⇒ x* is not Tabu)
9       Accept move and update best solution
10      Update tabu list and aspiration level
11      Increment iteration number
12   else
13     if (cost(x*) < AL)
14        Accept move and update best solution
15        Update tabu list and aspiration level
16        Increment Iteration number
17     endif
18   endif
19   if (stopping criteria is reached)
20      break
21   endif
22 endfor
```

Figure 2.5. Tabu search algorithm

The algorithm starts with an initial solution (Line 1). In the case of scheduling problems, it can be any permissible schedule. The *cost* of this schedule is recorded as the current best cost (Line 2). At each iteration of the search, all the neighbor solutions are determined and their leakage cost metric is calculated (Line 6). The notation $x \Rightarrow x^*$ is used to represent a move which takes a solution x to its neighboring solution The best neighbor solution is then identified (Line 7), and checked if the move is *tabu*. A move is said to

be tabu if it takes the current solution to a solution that has been visited recently. This memory is maintained in a list data structure called the *tabu list*. The extent of memory, which is the length of the list, is called the *tabu tenure*. If the move is not tabu, then the move is made and the tabu list is updated (Line 9-11). This concept of short term memory is illustrated in Fig. 2.6. On making a move from solution $X1$ to a neighboring solution $X2$, the move $X2 \Rightarrow X1$ is considered tabu. This prevents the search from returning to $X1$, whose neighbor costs we have already evaluated once.

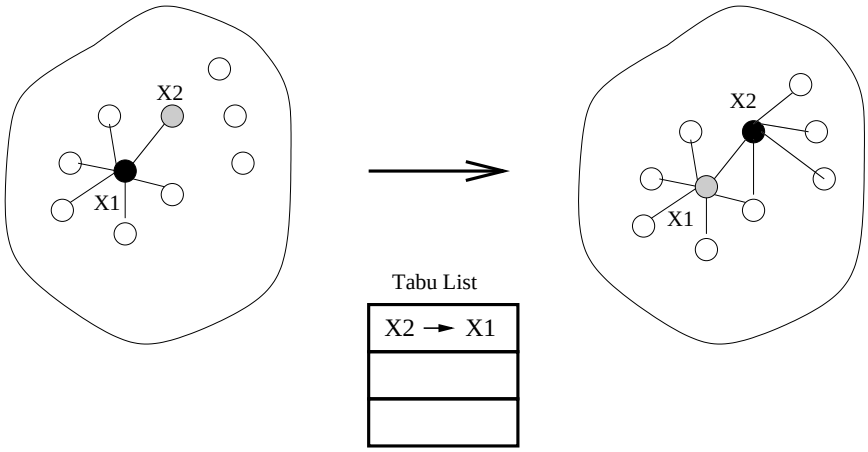


Figure 2.6. Short term memory in tabu moves

If the cost of the new solution is lower than the current minimum cost, then a new minimum is reached and the current minimum cost is updated. Note that this is not a necessary criterion for the move to be made. Hill climbing moves are also allowed. To avoid missing a good move even if it is tabu, the cost of the move is compared to an *aspiration level* (Line 13). If the cost is lesser than the AL, we accept the move. Otherwise, we reject the move. The simplest aspiration level is the current minimum cost. If the tabu move results in a cost lesser than the current minimum cost, we accept the move as an exception. Glover[17] defines the aspiration level (AL) as a condition used to over-ride the tabu status of a move. This helps to avoid the case when a good solution, reachable only by a tabu move, is missed.

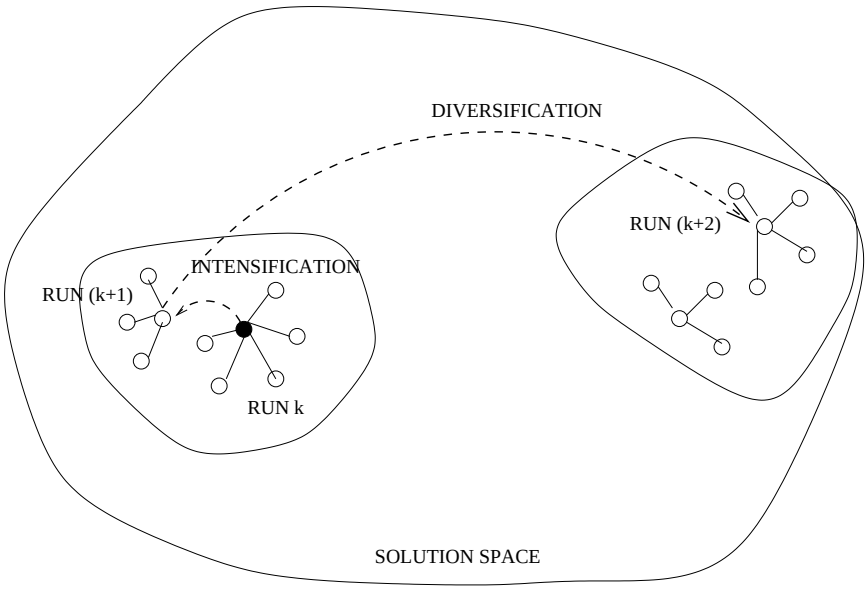


Figure 2.7. Intensification and diversification in tabu search

Algorithm in Fig. 2.5 describes one run of a tabu search that requires only short term memory (tabu lists). *Intensification and Diversification* strategies help to search more effectively. Intensification strategies are based on the premise that good solutions are localized. Diversification strategies help in moving out of local minima and exploring previously unexplored territories. These strategies are implemented using long term memory, which span across multiple runs. Fig. 2.7 illustrates solution space search using intensification and diversification strategies. The figure shows the alternate application of intensification strategies. If the current run is Run k, intensification is carried out in Run $(k+1)$ and diversification is carried out in Run $(k+2)$.

The tabu search synthesis system (TASS) works towards a schedule that optimizes area in the synthesized data-path. Moves consist of changing the timestep of an operation to a new timestep within the mobility of the operation. Schedules that are reached by making moves on the current schedule are called the neighboring solutions. For each of these schedules, TASS calculates a cost, which is the sum of penalty weights. Each weight is cost-based penalization of a worst assignment of operation nodes requiring the greatest

amount of a given hardware resource. The TASS system is a time constrained scheduling algorithm. Given a maximum number of timesteps, the nodes of a graph are moved, such that minimum number of resources are allocated.

To prevent cycling, TASS stores the recently made moves in tabu lists. The moves are stored in the form of (operation, move_from, move_to) triads. To avoid missing good solutions while not cycling, TASS uses the minimum cost as the aspiration level. If a tabu move results in a cost which is lower than the current best cost, then the aspiration level is reached, and the move is accepted. Long term memory is also exploited in TASS. Moves are recorded across runs. Moves that are made too frequently are penalized, so that search may move on to unexplored territory.

2.7 Summary

In this chapter, we briefly outlined the various sources of leakage power. We discussed various leakage power estimation algorithms that have been proposed in the near past in Section 2.3. We observe that most of these techniques addressed the leakage power estimation at gate level or lower levels of hierarchy. These techniques provide good accuracy, but are significantly time consuming. Section 2.2 outlined the recent techniques that have been proposed to minimize the leakage power dissipation in CMOS circuits. A majority of these techniques optimize leakage power either at the gate-level or at the transistor level. The MTCMOS technology style is described in detail in Section 2.2.3. Related work to this dissertation, in minimization of leakage power at higher levels of design abstraction, is presented in Section 2.2.5. There has been significant work that has been carried out in minimization of leakage power in cache memories. Section 2.5 gives an overview of related work in the area of RT-Level power estimation. The chapted concludes with related reseach work done in the area of design space exploration. We describe in detail, simulated annealing and a relatively newly proposed search technique called tabu search.

CHAPTER 3

FRAMEWORK FOR SYNTHESIS OF LOW LEAKAGE DATA-PATHS

The synthesis algorithms presented in this dissertation to synthesize low leakage power data-paths have been incorporated into AUDI, a behavioral synthesis system. The AUDI (**AU**tomatic **D**esign **I**nstantiation) system is capable of automatically synthesizing data-paths with MTCMOS components. Leakage power management techniques presented in Chapter 4 and the tabu search framework presented in Chapter 6 for the synthesis of low leakage power data-paths have been incorporated into this system. The system is also capable of generating synthesized data-paths with components embedded with the leakage power model described in Chapter 5. Section 3.1 describes the high level synthesis sytem. Section 3.3 describes the MTCMOS component library that has been developed and extensively characterized for leakage power and delay. Section 3.4 analyses the tradeoff between leakage power and propagation delay as a function of the sleep transistor width.

3.1 Automatic Design Instantiation System (AUDI)

The AUDI system is the central vehicle of research for this dissertation. The AUDI system is capable of synthesizing fully functional structural VHDL from behavioral data flow graph (DFG) representations. The system is currently capable of synthesizing data-path intensive designs. Several scheduling algorithms have been implemented in the system. They vary from simple algorithms such as the as-soon-as-possible (ASAP) and the as-late-as-possible (ALAP), to complex force directed algorithms such as the Force Directed Scheduling (FDS) proposed by Paulin and Knight [55] and the simultaneous scheduling, allocation, and mapping algorithm (SAM) proposed by Cloutier [56].

Allocation and mapping are done using a clique partitioning heuristic proposed by Tseng and Siewiorek [57]. The heuristic attempts to form a minimal set of maximal sized cliques. This results in maximum sharing between components allocated. Both functional unit (FU) and register mapping is done using the same heuristic. In the case of FU mapping, the input to the clique partitioning heuristic is the compatibility graph of the operations in the scheduled DFG. In the case of register mapping, lifetime analysis is done prior to the mapping. A compatibility graph of the edges in the DFG is formed. Two edges are compatible if they have non-overlapping lifetimes. Sharing of FUs and registers may be implemented using multiplexers or buses. Fig. 3.1 show the compatibility graph and the cliques formed for FUs in a scheduled DFG.

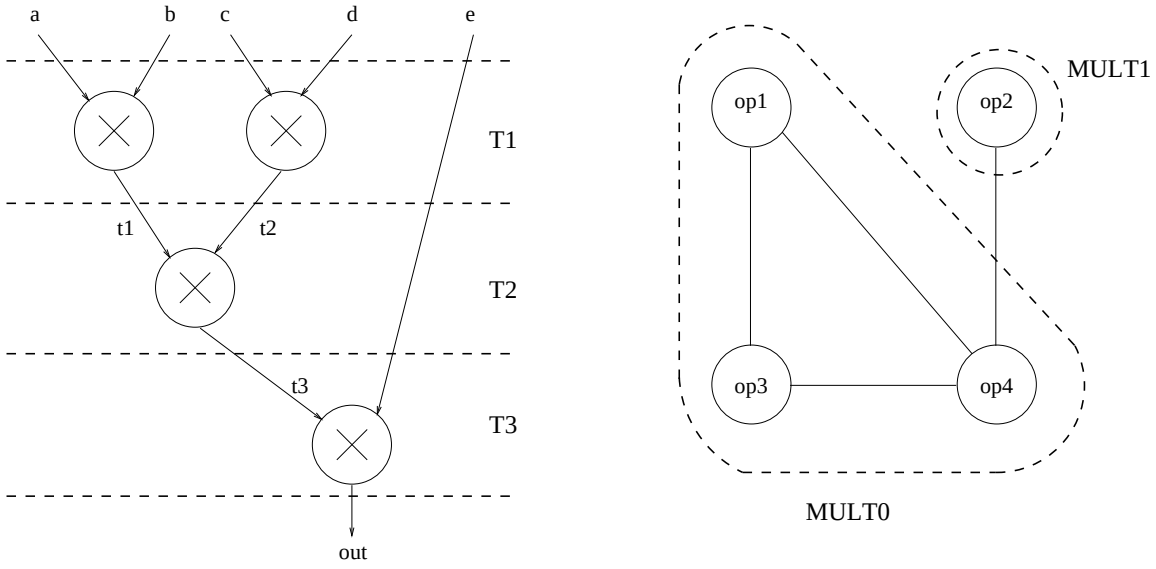


Figure 3.1. Mapping of functional units using clique partitioning

The datapath may be synthesized using either components from an available standard cell library or the MTCMOS component library that has been developed. The MTCMOS component library has been characterized in detail and presented in Section 3.3.

Fig. 3.2 shows the RT-level model synthesized by AUDI. The top level design instantiates a data-path and a controller. The data-path and the controller communicate using flags and control signals. The data-path and the controller are driven by the same clock

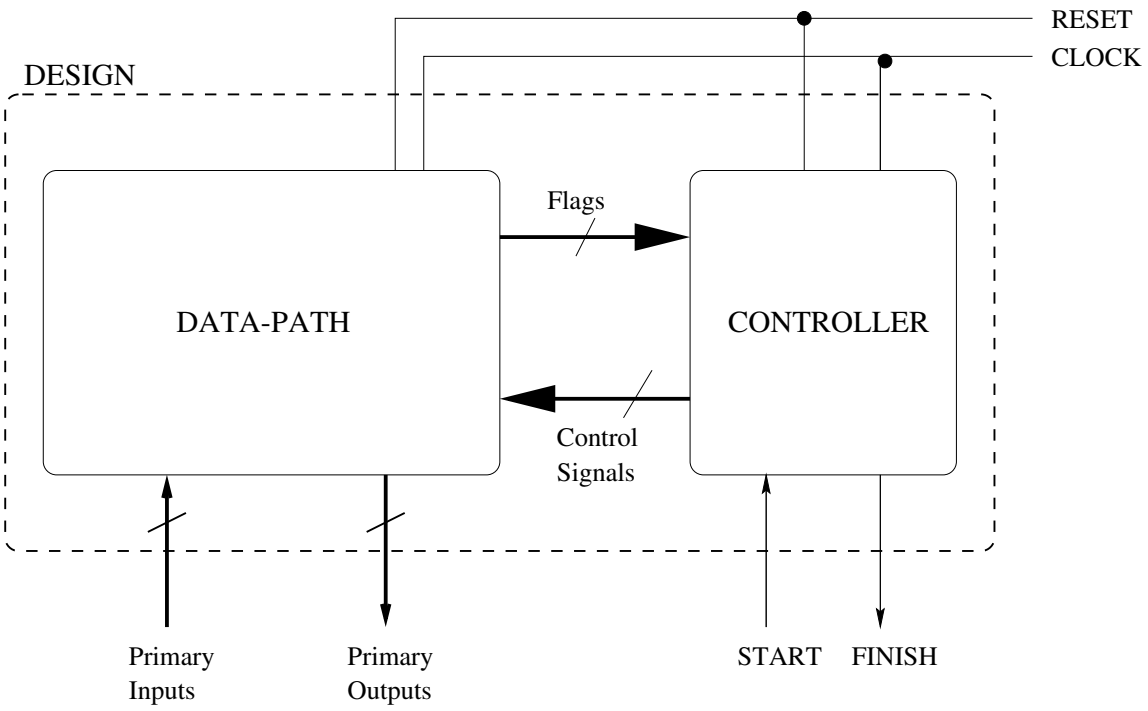


Figure 3.2. RT-Level design model

signal. The controller is implemented as a Finite State Moore Machine. Fig. 3.3 and 3.4 summarizes the state machine of the controller generated by AUDI. Each time step maps to a single clock cycle. The controller is synthesized such that functional units perform active computation during the first half of the clock cycle and control signals are generated in the latter half. State 0 is the IDLE state when the circuit waits for an input vector on its primary inputs. Registers need to be cleared using the `CLEAR` signal at the start of operation. When a new input vector arrives, the `Start` signal is required to be asserted. This takes the controller to the first state where the primary inputs are latched on to registers. Once the primary outputs are computed, they are latched on to registers and the `FINISH` signal is asserted. The circuit then goes back to idle state. The controller is responsible for generating the control signals which enable the registers (EN) for writing (WR) and the select signals ($SELECT$) for the interconnect units (multiplexers).

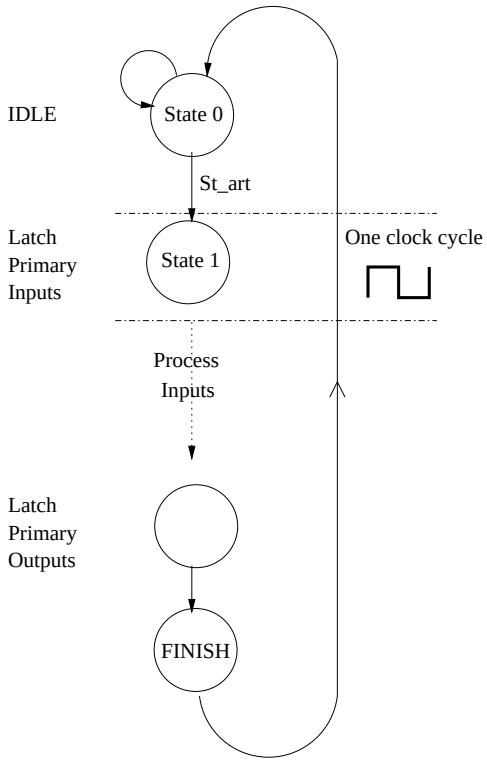
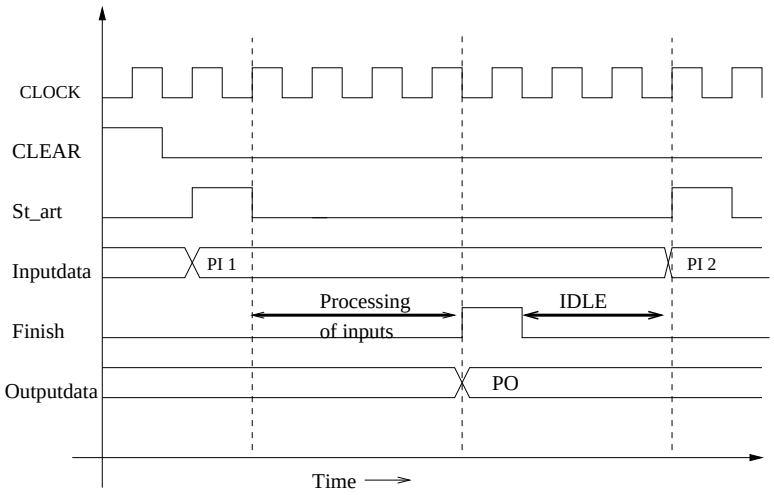


Figure 3.3. Controller state machine in AUDI



PI = Primary Input PO = Primary Output

Figure 3.4. Sequence of events in the synthesized design

Designs synthesized using the system have been verified at the RT-Level using the Cadence VHDL Simulator. Layouts were generated for the designs using the LAGER IV Silicon Compiler [58] and have been verified using HSPICE.

3.2 Proposed approach for synthesizing low leakage data-paths

Figure 3.5 describes the overall approach to synthesizing low leakage power data-paths. The input to the framework is a data-flow graph. Flows 1 and 2 represent the constructive approaches to synthesize low leakage power data-paths. Chapter 4 describes the clique partitioning based allocation algorithms proposed. Section 4.3 explains the power management techniques in detail. Section 4.2 outlines the modifications that have been made to the clique partitioning algorithm proposed in [57], for synthesizing low leakage power data-paths. The selective binding algorithm discussed in Section 4.4 binds components optimally to MTCMOS technology, depending on the area constraints provided by the designer. The selective binding problem has been mapped to the 0-1 Knapsack problem. To alleviate the performance loss due to MTCMOS components in the data-paths, we propose a performance recovery technique in Section 4.5, based on multi-cycling and slack introduction.

Flows 3 and 4 describe the tabu based search techniques that have been proposed to search for low leakage power data-paths. Flow 3 searches the large design space of possible schedules for a given data flow graph, for a low leakage power data-path (Section 6.2). Flow 4 performs simultaneous scheduling, allocation, and binding based on tabu search. The result is a low leakage schedule and binding (Section 6.3).

The leakage power of RT-Level data-paths can be measured in two different ways. The first and more time consuming method is to obtain the magic layout and spice netlist through the Lager IV Silicon compiler [58], and obtain leakage power using HSPICE. The second method is using the fast architectural simulator for leakage power (FASL) that has been developed to measure leakage power of RT-Level data-paths described in hierarchical VHDL. The mechanism of measuring leakage power through FASL is explained in detail in

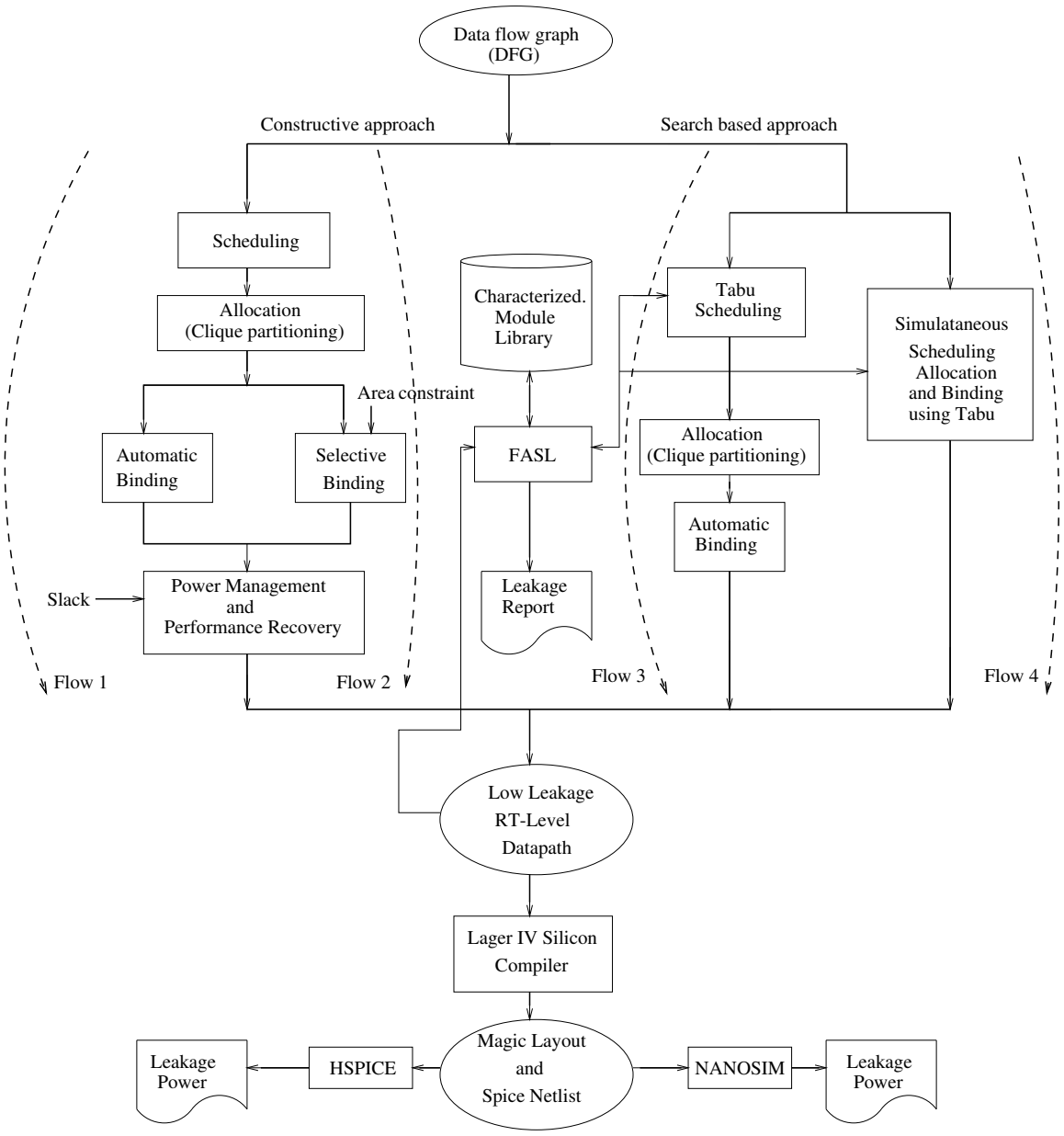


Figure 3.5. Proposed approach (extension to AUDI) for synthesizing low leakage data-paths

Chapter 5. Section 3.3 describes the MTCMOS component library that has been developed and characterized extensively for use in the synthesis of low leakage power data-paths.

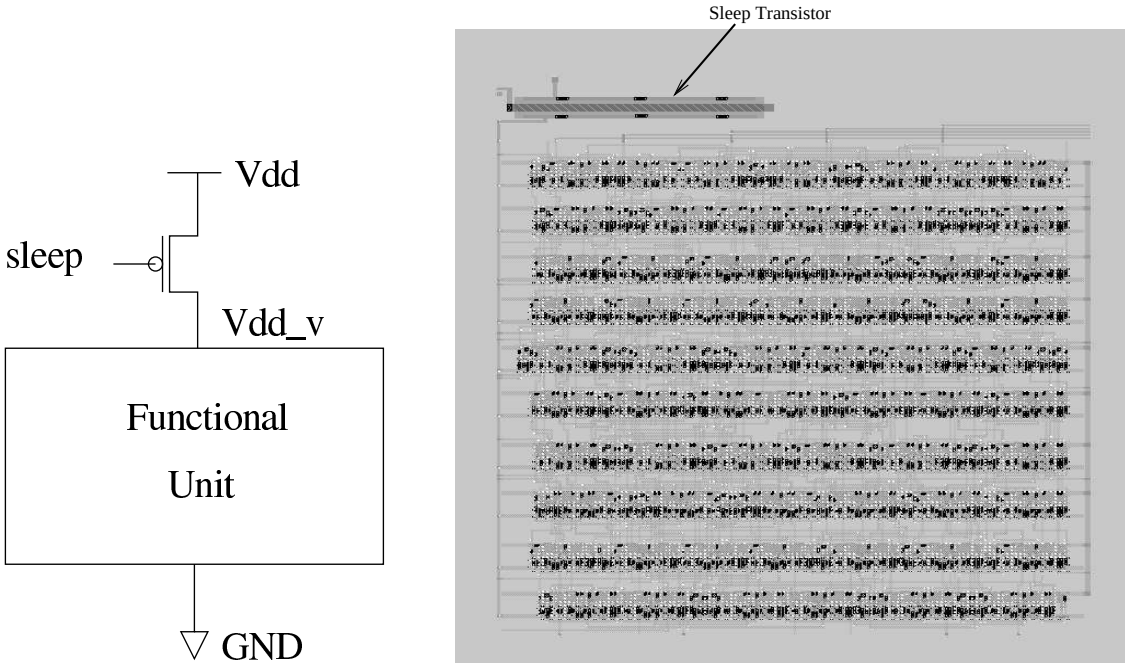
3.3 MTCMOS component library

The data-paths synthesized by AUDI comprise of components from a parameterized macro-cell library, developed at the University of Cincinnati [59]. The library consists of functional units (such as adders, subtractors, and multipliers), storage units (such as registers), and interconnect units (such as multiplexers and buses).

Motivated by Powell's work [32], we extend the concept of MTCMOS to gating the V_{dd} to components in the macro-cell library. Section 2.2.3 introduced the MTCMOS design style. A characterized library of MTCMOS components for use in the synthesis of low leakage power designs is presented. The MTCMOS components are characterized for leakage power and propagation delay as a function of the sleep transistor width. This aids the designer to choose the sleep transistor width for a design.

The MTCMOS component library includes registers, functional units (adders, subtractors, and multipliers), and interconnect units (multiplexers), which are capable of being gated `OFF` from supply voltage using a *sleep* control signal. The modules are available as parameterized descriptions that can be synthesized to layout using the LAGER IV Silicon compiler [58]. Fig. 3.6 and Fig. 3.7 show schematic and layout representations of a sleep multiplier and a sleep register respectively.

In the MTCMOS design style, it is sufficient to have a single sleep transistor - either a PMOS transistor gating the supply voltage from the circuit or an NMOS transistor gating the ground from the circuit [19]. A low V_T ($V_T \approx 0.3V$) PMOS transistor gating the supply voltage to the module has been used. The modules are also synthesized using low V_T transistors. The module is gated `OFF` by an active high sleep signal. Registers need to be active when they are being cleared. For this purpose, there is an additional parallel sleep transistor, in the case of MTCMOS registers (Fig. 3.7).



(a) Block level (b) MAGIC layout

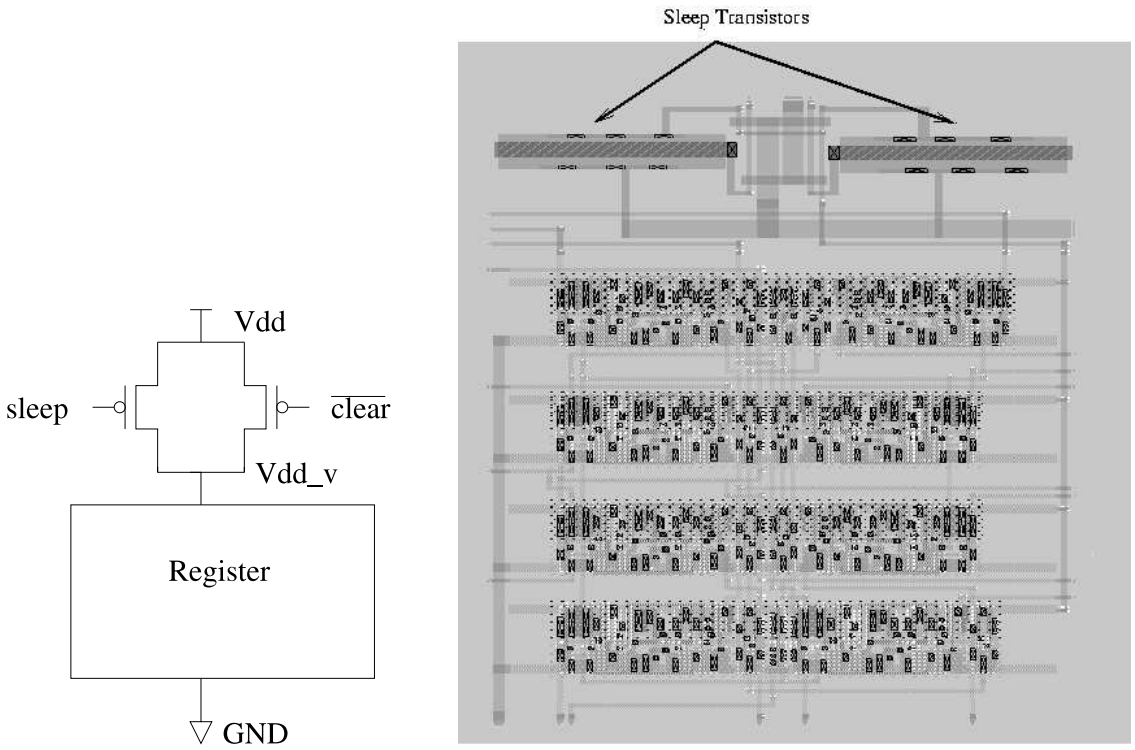
Figure 3.6. MTCMOS FU Modules - shown here is an MTCMOS multiplier

3.4 Leakage power and delay tradeoff analysis

The MTCMOS modules have different leakage power and performance characteristics with varying widths of the sleep transistor. The module library has been characterized for leakage power and delay as a function of the sleep transistor width. Characterization of leakage power is performed when the module is in sleep mode, with the sleep transistor switched off. In the sleep mode, the leakage power dissipated is entirely due to the sleep transistor. Power characterization is done using HSPICE at level 49 (BSIM3v3) using 0.18 μm technology parameters.

It is observed that, as the sleep transistor becomes wider, the leakage power dissipated by the sleep transistor increases. This is due to the dependency of the subthreshold leakage current on the (W/L) ratio of the transistor (See Eqn. 2.1).

The module characterization is a one-time procedure. The characterization has been automated using a unix SHELL script, and has been integrated into the design approach



(a) Block level (b) MAGIC layout

Figure 3.7. MTCMOS register

(Fig. 4.1). The spice netlist of the module (FU or storage element) is appended with sleep transistors of varying widths. A series of HSPICE simulations determine the leakage characteristics of the module with respect to the sleep transistor widths. The module is switched off during leakage measurements. The characterization for delay is done when the module is in active mode. This measures the delay overhead incurred because of the sleep transistor when the module is in normal operation. We observe that the delay overhead decreases as the sleep transistor width increases. This is due to the dependence in the propagation delay equation, where the delay is inversely proportional to the sleep transistor width. Delay measurements were made using the HSPICE at 0.18 μm technology node. A $(previous\_input, current\_input)$ pair that excites the critical path of the module is determined using IRSIM[60]. With this input pair, a series of HSPICE simulations determine the delay characteristics.

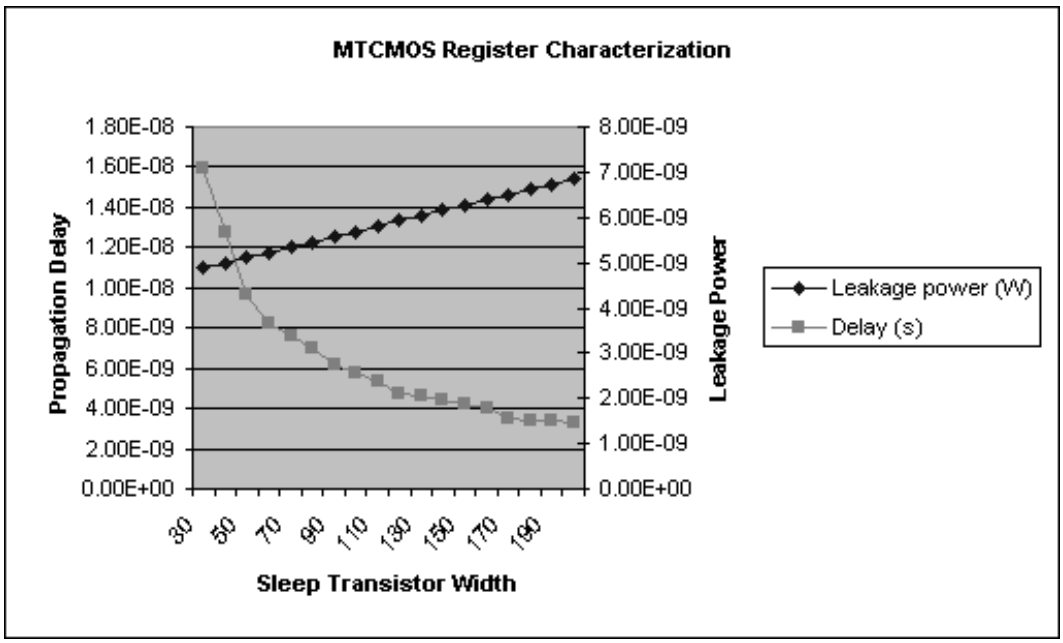


Figure 3.8. Power-delay characterization of 8-bit MTCMOS register

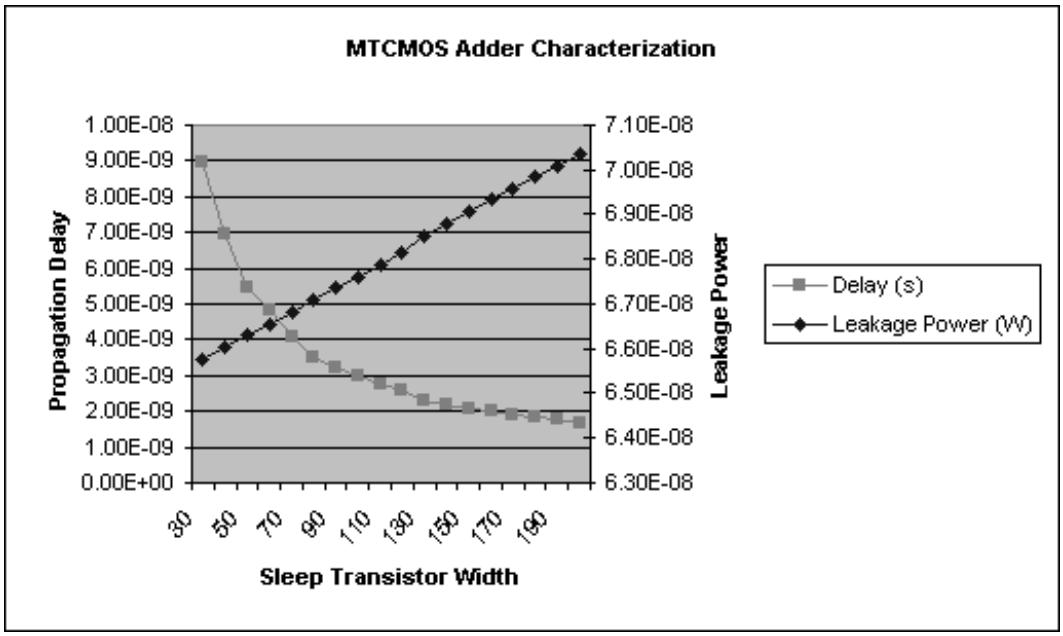


Figure 3.9. Power-delay characterization of 8-bit MTCMOS adder

This characterization is helpful in picking the optimal widths for MTCMOS modules depending on the application and permissible overhead available for the design. An *optimal*

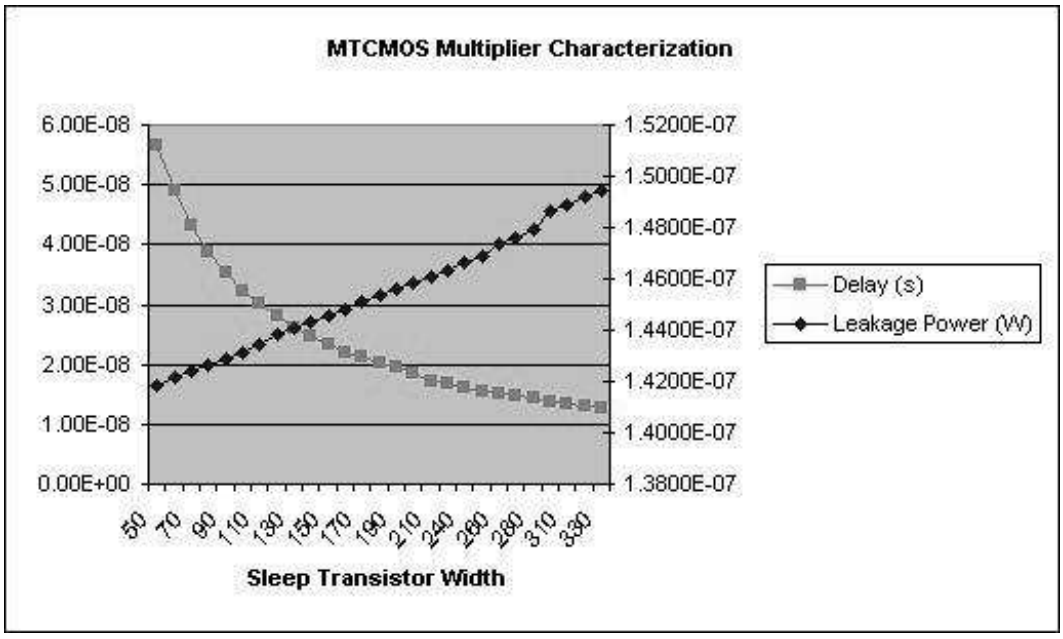


Figure 3.10. Power-delay characterization of 8-bit MTCMOS multiplier

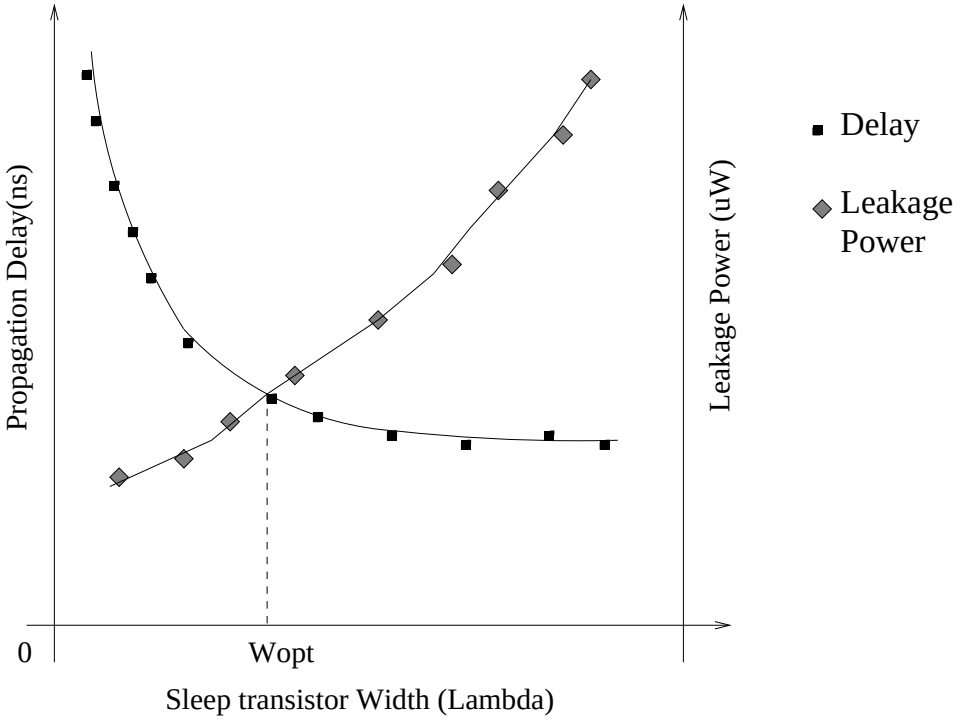


Figure 3.11. Characterization profile

Table 3.1. Optimal widths for the modules determined using characterization

Module	Bit Width	Optimal Width of Sleep Transistor (λ)
Adders	4	50
	8	60
	16	70
Registers	4	30
	8	40
	16	50
Multipliers	4	100
	8	200
	16	300

sleep transistor width is that which yields good performance with minimal leakage power overhead. Fig. 3.11 shows the typical leakage power and delay profiles of a module as a function of sleep transistor width. The optimal width is given by the intersection of the two curves. For example, consider the adder shown in Fig. 3.9. An MTCMOS adder with $w = 70\lambda$ would give reasonable performance with minimal leakage power dissipation. An adder with $w = 160\lambda$ would give significant improvement in performance with an increase in the leakage power dissipated. It should be noted that, the leakage power dissipated is still lesser than a regular adder (non-MTCMOS). An adder with $w = 40\lambda$ would be leakage efficient but, with reduced performance. If the designer can tolerate this degradation in performance, this adder would dissipate significantly lower leakage power. The optimal widths picked for the modules implemented in TSMC 0.18 μm technology are shown in Table.3.1. It should be noted that the width of the sleep transistor for 8-bit multiplier is greater than the optimal width. This is because of the higher speed desired.

3.5 Summary

In this chapter, the Automatic Design Instantiaton System (AUDI), a behavioral synthesis system capable of synthesizing data-path intensive designs, was presented. Section 3.2 outlined the various extensions that have been proposed to enable AUDI to synthesize low leakage power data-paths. Finally, the chapter concludes with the introduction of the

MTCMOS component library that has been developed. A comprehensive trade off analysis of leakage power and delay, with respect to the sleep transistor widths, has been discussed in the section.

CHAPTER 4

ALLOCATION AND BINDING APPROACH FOR LOW LEAKAGE POWER

The three main tasks of high level synthesis viz. scheduling, allocation, and mapping have influences on the leakage power dissipated by a design. Appropriate allocation and binding followed by power management using MTCMOS modules can significantly reduce leakage power dissipated. An allocation and binding algorithm is proposed, which attempts to maximize contiguous idle times of resources, thus minimizing leakage power. The proposed approach is shown in Fig. 4.1.

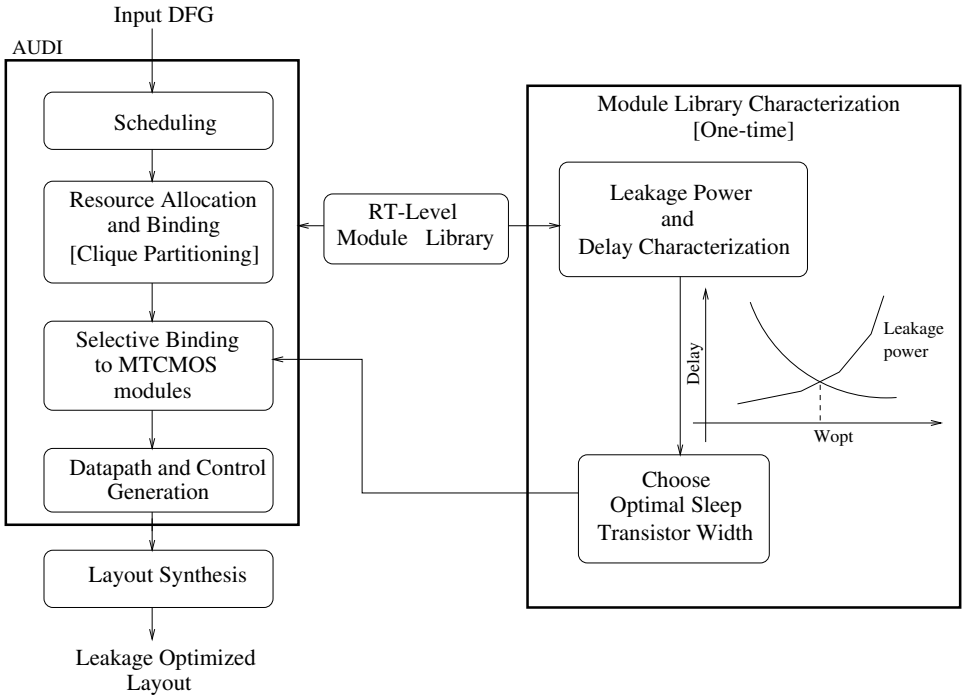


Figure 4.1. Allocation and binding approach for low leakage power

The modules are then characterized for leakage power and delay as described in Chapter 3. Using the trade-off plots (such as Fig. 3.11), optimal widths are chosen for the modules. A resource allocation and binding algorithm based on clique partitioning, to ensure maximal idle times is proposed. Modules with significant idle times are then selectively bound to MTCMOS modules.

4.1 Motivational example

As a motivational example, consider the scheduled data-flow graph of the IIR filter shown in Fig. 4.2. The schedule shown is the ASAP (as soon as possible) schedule. The DFG consists of nine operations - four additions and five multiplications.

The schedule takes six control steps to process a set of input vectors. Five multipliers and two adders are allocated. If the multipliers are implemented using MTCMOS modules, they can be turned off during control steps T2 to T5. There are three possible bindings for the adders as given in Fig. 4.3. Intuitively, the third binding seems to be the best, since it offers the longest contiguous idle time.

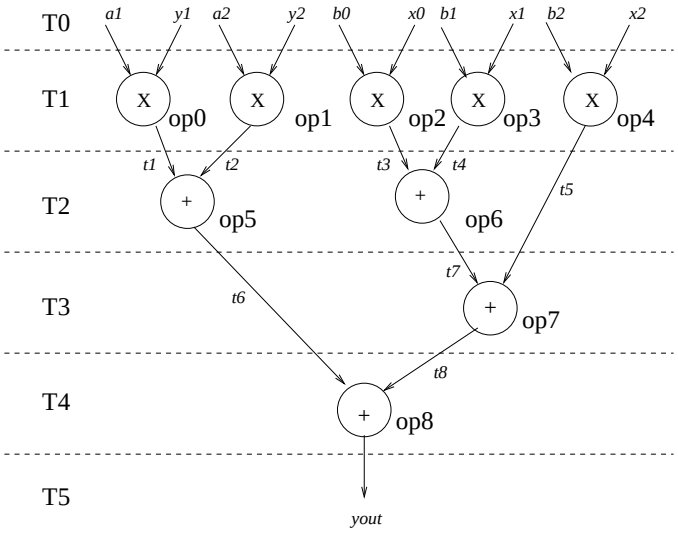
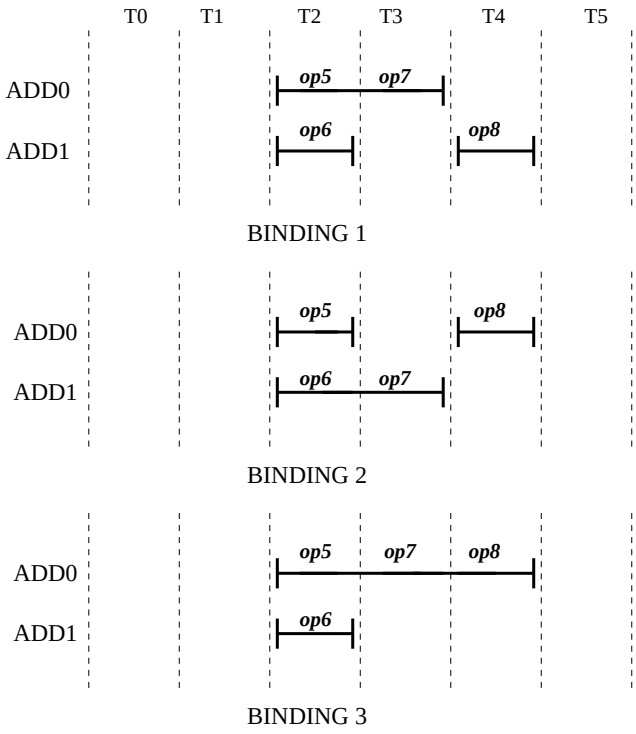


Figure 4.2. As-soon-as-possible schedule for IIR filter



	Binding 1/2		Binding 3	
	Activity	Leakage	Activity	Leakage
ADD0	ON--OFF--ON	0.5062 uW	ON--ON--ON	0.4052 uW
ADD1	ON--ON--OFF	0.4527 uW	ON--OFF--OFF	0.4526 uW
		0.9589		0.8578

Figure 4.3. Possible bindings for adders in IIR

Table in the Fig. 4.3 tabulates the leakage power measured for the different binding alternatives. We observe that turning off the multipliers results in a reduction of 14.83% in leakage power dissipated by the multipliers. We also observe that the third binding provides the best alternative and results in a reduction of 11.19% in the leakage power dissipated by the adders. Leakage energy measurements were made using HSPICE in 0.18 μm technology node. Table in Fig. 4.3 compares switching off modules over long continuous idle periods to shorter idle periods. In the table, `OFF` denotes an idle period and `ON` denotes an active period. A 10% difference in leakage power dissipation between

the two activity patterns is observed. Bindings 1 and 2 have similar activity patterns, but with interchanged operations. In the case of more complex and larger modules such as multipliers, this effect is more prominent. For example, the same kind of activity patterns as shown in the table in Fig. 4.3 for an 8 bit-multiplier results in a 70% reduction in leakage power dissipation.

4.2 Clique partitioning for low leakage power

The proposed resource allocation and binding algorithm is based on the clique partitioning heuristic proposed by Tseng and Siewiorek [57]. In the case of register allocation and binding, the input to the algorithm is an undirected graph whose vertices represent the variables in the dataflow graph (DFG). The clique partition algorithm attempts to form minimum number of cliques which represent the registers that would store the variables. In the case of FU allocation and binding, the input to the algorithm is an undirected graph whose vertices represent the operations in the DFG. The algorithm obtains the minimal number of FU instances needed to execute the operations. If the operation is bound to an MTCMOS module, larger number of idle clock periods will enable the module to be switched `OFF` for longer durations, thus minimizing leakage power. At the same time, frequent transitions between idle and active states would result in increase in the short circuit power and capacitive power. The increase in capacitive power is due to the charging and discharging of the input capacitances every time such a transition happens.

Let G = (N,E) be the graph to be partitioned into a minimum number of cliques. The nodes in the graph represent carriers (edges in the data-flow graph). The objective of the clique partitioning algorithm is to obtain a minimal set R of sets of carriers (cliques), each of which will share a resource. To minimize the leakage power dissipated by the resource, it is desirable to have a longer sleep time (idle time), preferably contiguous. This is to avoid the possible increase in short circuit power dissipation due to too many transitions between idle and active periods. During each iteration of the algorithm, a node is picked to be added to the current clique so that its addition to the clique will ensure a long idle

```
1 procedure clique_partition(N, E, W_sleep, W_trans)
2 while (N is not empty) do
3   x ← select_node();
4   if x is not connected to any other node
5      R = R ∪ C(x)
6   else
7      Form set Y with all neighbors of x
8      for each node i in Y,
9        determine set of nodes Y_i ∈ Y
10        that are incompatible with i.
11     endfor
12     Determine Y_1 from Y_i that excludes
13     the minimum number of nodes in Y
14     foreach y ∈ Y_1
15       Calculate new sleep time if y is added
16       Calculate new number of transitions if y is added
17       cost factor =
18             W_sleep * new_sleep_time
19           + W_trans * new_transitions
20     endfor
21     Y_2 ← nodes with max cost factor
22     Merge nodes x and y into a supernode z and give it
23     higher priority
24   endif
25 end procedure
```

Figure 4.4. Clique partitioning algorithm for leakage power optimized resource allocation

time and lesser number of transitions between idle and active periods (Lines 17-21). The user can specify the values for W_{sleep} and W_{trans} for sleep time and number of transitions. The algorithm is outlined in Fig. 4.4.

During each iteration, the clique partitioning algorithm picks a pair of nodes in the graph and merges them (Line 22). The merged node is given a higher priority for being picked in the next iteration (Line 23). The first node of the pair is selected using the *select_node()* procedure (Line 3) which works as follows: If any node has a priority of 1 then that node is selected. Otherwise, it selects a node with the highest *weight*. Weight of a node n is defined to be the degree of the node plus $|C(n)|$. In case of a tie, the node with the maximum *neighbor weight* is selected. Neighbor weight of a node n is defined to

be the sum of the weights of the nodes connected to n. If a tie still exists then one of these nodes is selected arbitrarily. The second node is picked as shown in the algorithm.

In summary, two new parameters have been added to the cost function to minimize leakage power dissipated :

- *new_sleep_time* is the number of control steps the resource would be idle if the node is added to the clique
- *new_transition* is the number of transitions that the resource would make between idle and active states, if the node is added to the clique

The clique partitioner allocates and binds operations (or edges) to FU instances (or registers) so that if bound to MTCMOS instances, they would minimize leakage power. However, it is observed that binding all the modules to MTCMOS instances is not always advantageous. It is observed that, it is not cost effective to map modules with less than two consecutive idle clock periods, for significant leakage power reduction.

4.3 Power management in the data-path

MTCMOS modules in the datapath have the capability of being turned `ON` or `OFF` using the *sleep* signal. There are two power management techniques that can be implemented in this context. Firstly, modules can be turned `OFF` during their idle times, during the active stage (between *Start* and *Finish*). Secondly, all MTCMOS modules may be turned `OFF` between the *Finish* state and the arrival of the next input.

MTCMOS Functional Units (FUs) such as adders and multipliers are turned `ON` only in timesteps that they need to compute valid data. The FUs are turned `OFF` during all other timesteps. This data may be obtained by traversing the dataflow graph (Fig. 4.5). The *sleep* signals are generated by the controller. Registers may be used to store one or more carriers with non-overlapping lifetimes. Registers may be turned `OFF` during intervals when they are not being used to store any valid data, i.e., outside of the lifetimes of all its carriers. For example, in Fig. 4.6, the register R0 is being shared by carriers i1, y1, and

```
procedure shutdown_regs()
 foreach timestep t do
   foreach register r  do
     foreach carrier c sharing register r do
       if (birth(c) < t < death(c)) then
          sleep(t) ← '0'
       else sleep(t) ← '1'
       endif
     endfor
   endfor
 endfor
end procedure

procedure shutdown_FU()
 foreach timestep t  do
   foreach FU  do
     if the FU performs any valid computation in t  then
       sleep(t) ← '0'
     else sleep(t) ← '1'
     endif
   endfor
 endfor
end procedure

procedure shut_down_mux()
 foreach timestep t  do
   foreach mux  do
     R ← resource (reg/FU) that the mux feeds
       if R performs any valid computation in t  then
         sleep(t) ← '0'
       else sleep(t) ← '1'
     endif
   endfor
 endfor
end procedure
```

Figure 4.5. Determining idle periods in datapath

y2. The lifetimes of these carriers are shown in Fig. 4.6. The register however is not being used to store any valid data in the idle period (t2-t3). The register can be switched OFF during this interval. The corresponding *sleep* signal to be generated is also shown. This

information can be obtained from the life time data of the carriers and the dataflow graph as shown in Fig. 4.5. A simple heuristic was applied for binding modules to the MTCMOS component library. Modules were bound to an MTCMOS component if and only if it had more than 2 contiguous idle control steps.

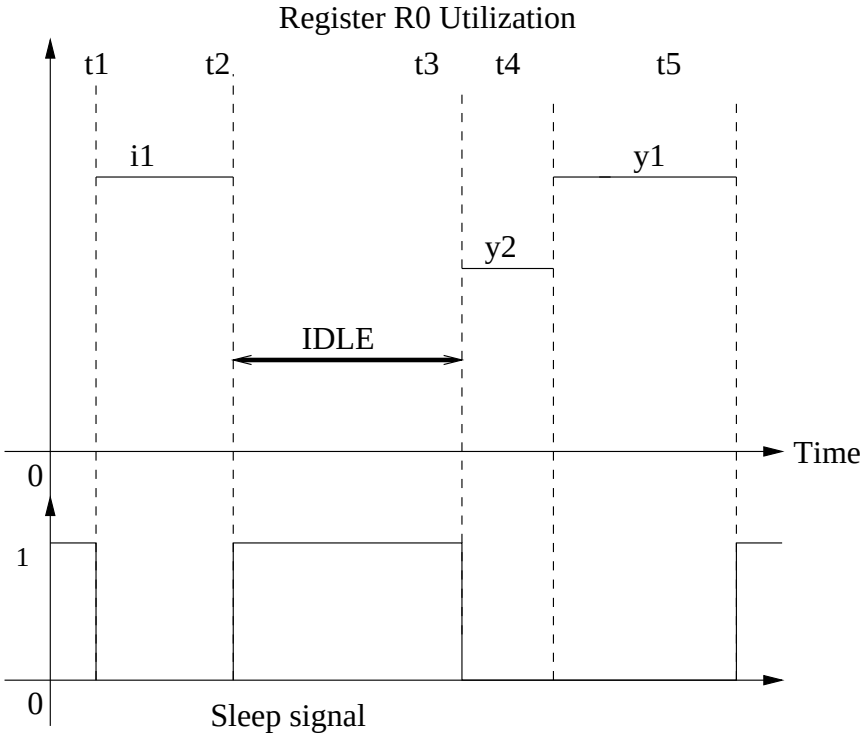


Figure 4.6. Register utilization and sleep signal generation

Switching `OFF` interconnect units is dependent on the Functional unit or Register that it feeds. For example, in Fig. 4.7, the multiplexer Mux0 needs to be `ON` during all the time intervals ADDER1 is `ON`. Similar is the case when a multiplexer feeds a register.

Chandrakasan *et. al.* analyses the effects of different data environments on the leakage power dissipated by a system. In the case of bursty applications, such as XTERMs, the authors present results that show significant idle times. The second power management technique is to shut down all the MTCMOS modules during idle times between processing two sets of input vectors.

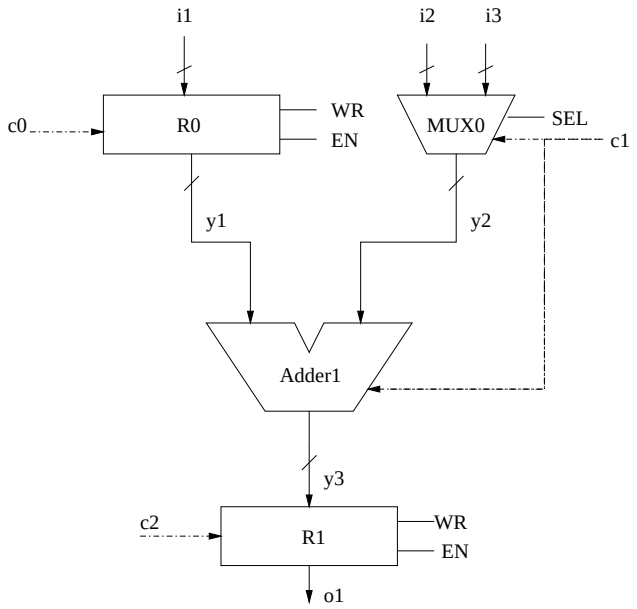


Figure 4.7. Assignment of sleep signals to modules

4.4 Selective binding of MTCMOS modules

The empirical binding rule suggested in Section 4.3 does not always yield the best binding. There are two reasons. Firstly, the leakage power profile of different module types is diverse. For instance, the leakage power dissipated by multipliers is almost an order of magnitude more than that dissipated by registers and adders. Secondly, the heuristic might result in a large number of MTCMOS modules, increasing the area overhead significantly, without significant reduction in leakage power. Hence, a more strategic binding algorithm is needed to selectively pick module instances for binding to MTCMOS technology. The binding must be in such a manner that, the subset of instances picked must result in maximum leakage power reduction. This problem is mapped to the familiar 0-1 Knapsack Problem. Mathematically, the 0-1 Knapsack problem is defined as:

Given a set Z of n items, with weights $w_1, w_2, \ldots, w_n$ and values $v_1, v_2, \ldots, v_n$, and a maximum capacity of knapsack W_{max}, pick a subset $Z^* \subset Z$ of items.

$$maximize\ the\ function\ (v_1.x_1 + v_2.x_2 + \ldots + v_n.x_n)$$

$$such\ that \quad w_1.x_1 + w_2.x_2 + \ldots + w_n.x_n \leq W_{max}$$

$$and \quad x_1, x_2, \ldots, x_n \in \{0, 1\}$$

The value of x_i determines whether the item i is picked or not. The set of items with $x_i = 1$ form the subset Z^*. The 0-1 Knapsack problem exhibits the optimal substructure property and can be solved using Dynamic Programming. The `minknap` algorithm proposed by Pissinger [61] has been used. It is based on dynamic programming where enumerative bounds are used to fathom unpromising states. It is proved that the algorithm enumerates the smallest possible core satisfying some given properties. In the case of selective binding of MTCMOS modules, the following mappings are proposed:

- **Z**: corresponds to our allocated module instances to implement the given DFG
- w_i: Each MTCMOS module type has an associated area overhead, due to the addition of the sleep transistor(s). This corresponds to the weight of the items in the knapsack.
- v_i: Each MTCMOS module type has an associated leakage power reduction over the original non-MTCMOS module.
- W_{max}: The designer has an area overhead constraint which corresponds to the maximum weight (capacity) of the knapsack.
- Z^*: The designer wishes to maximize his reduction in leakage power, which corresponds to the value of items picked in knapsack.

The area overhead of each module type (cost/weight) is calculated by the difference in area between the MTCMOS module and its non-MTCMOS equivalent. Area overheads for 180nm technology modules are shown in Table. 4.1. The leakage power reduction of each module is calculated by simulating the MTCMOS instance and non-MTCMOS instance over a long random input sequence (100 input vectors) and measuring the leakage power reduction due to MTCMOS technology. This is just a representative number signifying the magnitude of possible leakage power savings by that component type (Table. 4.1). This

value is the basis power number for each component type. To calculate the value of an item for the knapsack problem, this power number is multiplied by the number of contiguous idle control steps of the module instance. For example, a register `R10` with 4 idle control steps would have a value of $4 \times 0.1471983 = 0.5887932$, and that of `R0` with 1 idle control step would have a value of 0.1471983. When a choice needs to be made which of these registers to fit into the knapsack (or picked), `R10` is picked.

Table 4.1. Area-Leakage power values of 4-bit (180nm components)

Module type	Bit Width	Area Overhead(λ^2)	Base Leakage Power(uW)
Adder	4	72100	0.0952678
Register	4	91492	0.1471893
Multiplier	4	121568	0.5256012

Often, VLSI designers would have an upper limit on the area overhead affordable. This is provided as the maximum weight to the knapsack algorithm. Results are presented for various affordable area overheads. The 0-1 Knapsack algorithm provides us with the list of components to be mapped to corresponding MTCMOS instances.

A typical leakage power vs. permissble area overhead curve is shown in Fig. 4.8. It is observed that, if the designer can afford more area, more number of modules can be bound to MTCMOS instances. This would result in larger savings in leakage power. The modules to be mapped to MTCMOS instances are picked strategically, so that the leakage power reduction is maximized. Referring to Table 4.1, depending on the area permissible, the multipliers are almost always the first modules to be picked, because of the large leakage power savings they result in. However, beyond a certain number of MTCMOS modules, diminishing returns are observed. This can be attributed to the fact that, beyond a certain limit, the leakage power dissipated by the sleep transistors overcome the leakage power saved.

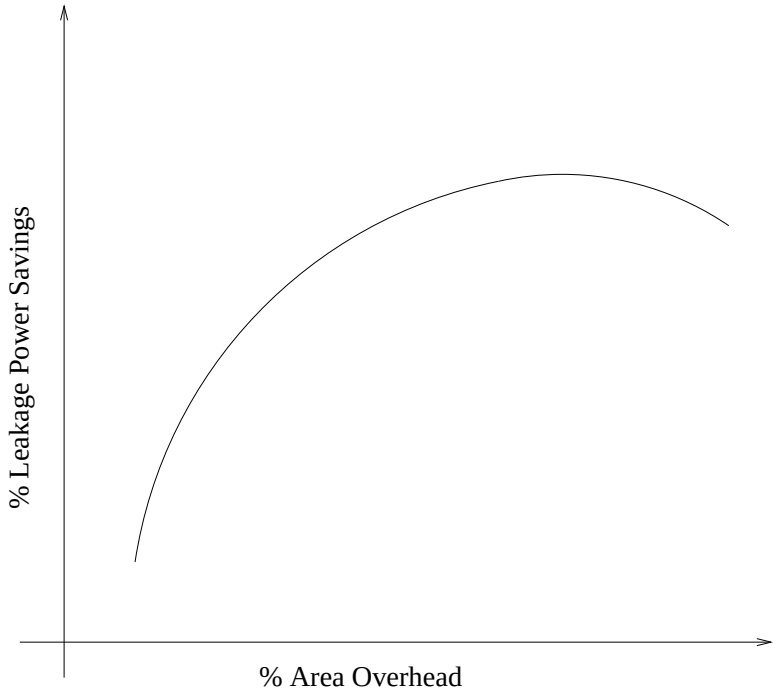


Figure 4.8. Typical leakage power savings vs permissble area overhead curve

4.5 Performance recovery by multicycling and slack introduction

As observed in Section 3.4, the introduction of the sleep transistor in an MTCMOS component increases the propagation delay of the component. Hence MTCMOS components are slower. When MTCMOS components are added to the data-path, the clock period of the design needs to be re-evaluated. AUDI estimates the clock period of a design using the model shown in Fig. 4.9.

The clock period is calculated as

$$T_{clock} = max(T_{FU}) + T_{reg} + T_{mux1} + T_{mux2}$$

where $max(T_{FU})$ is the largest of the critical path delays of the functional units used in the design. For example, if the design consists of 8-bit adders, subtractors, and multipliers, the value of $max(T_{FU})$ is the critical path delay of an 8-bit multiplier. Mux1 and Mux2 are multiplexers that arise due to register sharing and functional unit sharing in the design

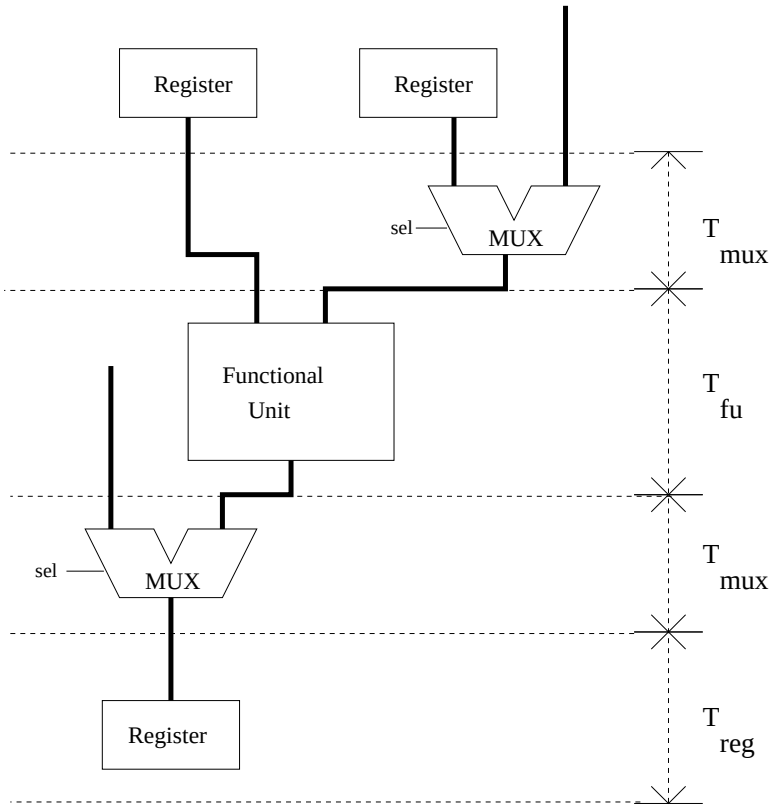


Figure 4.9. Determination of clock period

respectively. T_{mux1} and T_{mux2} are their corresponding propagation delays. The two multiplexer delays have been considered separately because they may have different number of inputs. T_{reg} is the register delay; the time a register to takes to latch on a value after the clock triggers it.

MTCMOS modules are inherently slower than regular modules. There are two contributors to the delay. Firstly, the sleep transistor itself is in the critical path of the module. Secondly, for larger modules, such as the multiplier, the *wake-up* times may be significant. The *wake-up* time arises due to recharge of all the nodal capacitances after an idle state. By choosing the sleep transistor widths as described in Section 3.4, the clock width increases by 40%.

The main motivation behind the performance recovery effort is the observation that the complex modules (such as multiplier) contribute mainly to the performance loss. A

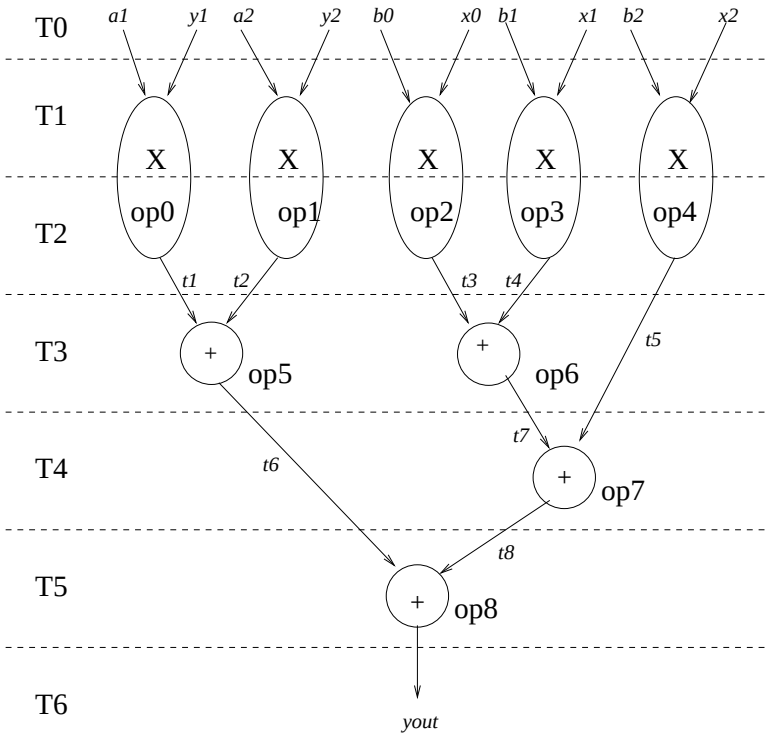


Figure 4.10. Performance recovery

method based on multicycling and slack introduction is proposed. The multicycling of the multipliers to two control steps enables us to use the same clock width as the original design. In the designs presented here, the multipliers are on the critical path and hence the reason for introducing slack. Application of the performance recovery technique to the motivational example in Section 4.1 is shown in Fig.4.10.

In this work, the metric used for measuring performance is the total execution time for a given number of vectors. It is observed that even after introducing an extra control step slack, this optimized schedule results in lesser execution time than the regular MTCMOS design. To illustrate this, let us again consider the motivational example that we presented in 4.1. The design takes 6 control steps for each input vector. The introduction of the MTCMOS modules widens the clock width from 50ns to 70ns. Thus, the MTCMOS design takes 4200ns for a set of 10 vectors. The multicycled MTCMOS design has a clock width of 50ns. However it takes 7 control steps per vector. Thus, the performance-

recovered design processes the 10 vectors in 3500ns. The performance penalty in this case is 14.28% as opposed to the 40% penalty in the regular MTCMOS design. This calculation is summarized in Table. 4.2.

Table 4.2. Performance penalty calculation for IIR

	Clock width(ns)	Control Steps	Exec. Time(ns)	Penalty
Original	50	6	3000	-
Regular MTCMOS	70	6	4200	40.0%
Multicycled MTCMOS	50	7	3500	16.6%

4.6 Experimental results

Results for three DSP designs, synthesized using the AUDI synthesis system are presented. The experimental flow is shown in Fig. 4.11. The RT-Level VHDL code obtained from the HLS system is synthesized to layout level using the LAGER IV Silicon Compiler [58]. Transistor netlist obtained from the MAGIC layout system is profiled using Synopsys Nanosim to obtain input traces at each of the data-path modules. Leakage power dissipated by the modules is measured using HSPICE and a post-processor that has been developed. Nanosecond-accurate power and voltage values are obtained for each transistor in the module from HSPICE. The post-processor determines the `ON/OFF` state of a transistor using the threshold voltage, and accordingly accumulates the leakage energy when the transistor is `OFF`. Results shown in this work, are obtained for the TSMC-0.18 μm technology parameters obtained from MOSIS[62] and 100nm technology parameters obtained from the Berkeley Predictive Technology Models (BPTM) [63].

Fig. 4.12 shows an example of a layout generated using the LAGER IV Silicon Compiler. The designs have been verified at RT-Level (using Cadence VHDL Simulator) and at layout level (using HSPICE). For each of the examples, ten random input vectors were generated using MATLAB. It should be noted here that, though the leakage power dissipated is statically input dependent, this technique capitalizes on the idle clock periods present in

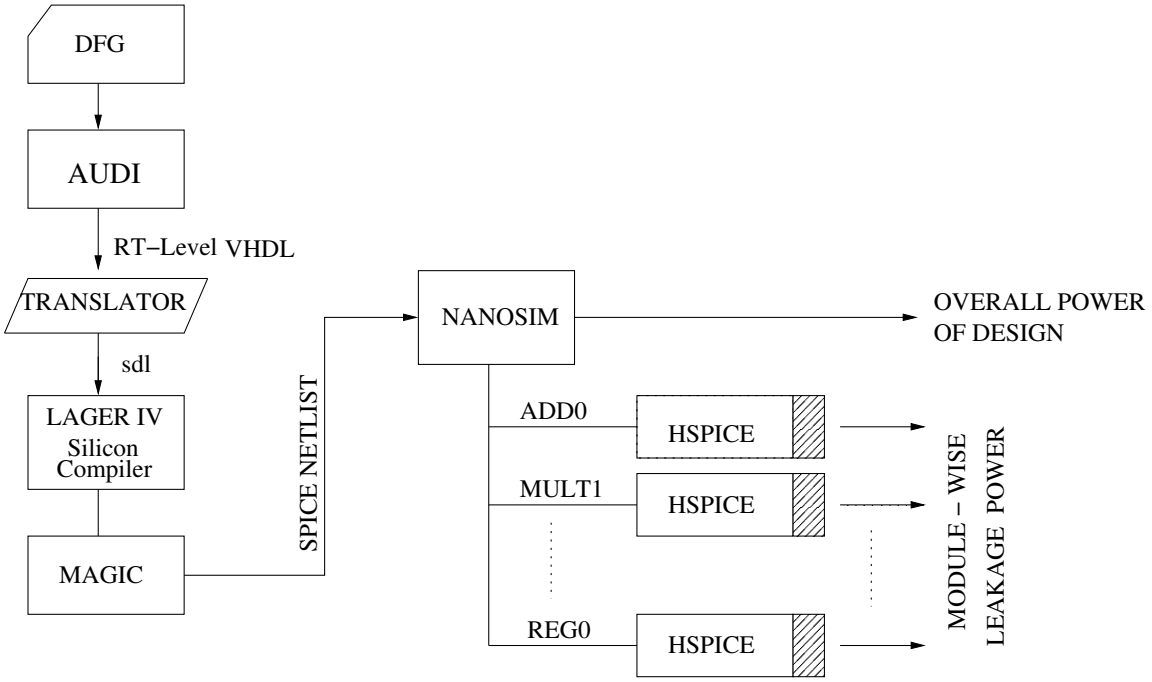


Figure 4.11. Experimental flow

the schedule. The small number of input vectors is due to the high computational time of HSPICE in calculating the leakage power. The main consideration of the work has been to minimize leakage power without an increase in the dynamic power. Significant work has been done [6, 7, 8, 9, 10, 11, 12, 13, 14] on dynamic power optimization during the process of behavioral synthesis.

4.6.1 IIR filter

The IIR Filter has ten primary inputs and one primary output. The DFG has nine operations - five multiply and four add operations. The ASAP schedule takes six control steps. The original design with no MTCMOS modules, was run at a clock frequency of 20 MHz. The leakage optimized design was run at a clock frequency of 14.28 MHz. This leads to a performance loss of 40%. The contribution of the leakage power to the overall power in the modules is shown in Figures. 4.13- 4.14. *In this discussion of experimental results, overall power refers to the total power dissipated by the design and is measured*

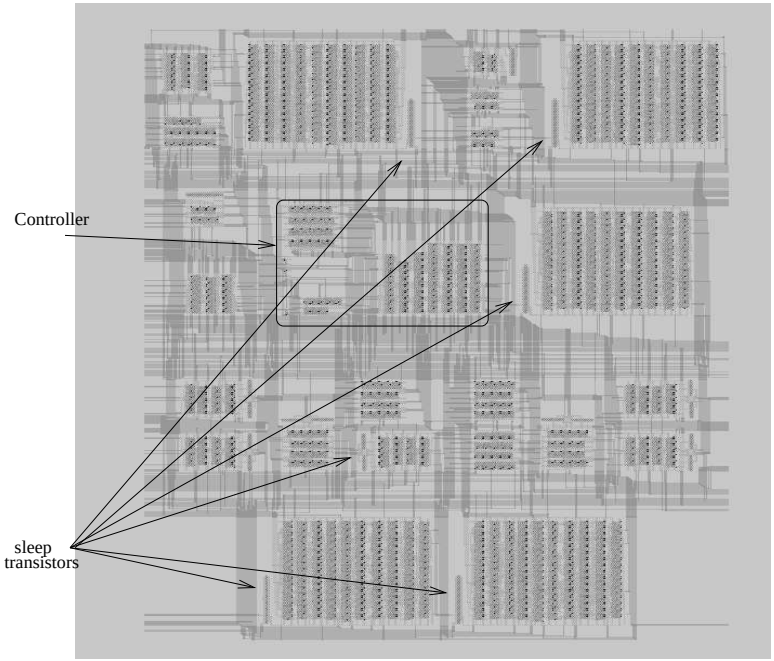


Figure 4.12. Layout of MTCMOS version of FIR filter

using the `measure avg power` *command of HSPICE.* It is observed that larger and more complex modules such as multipliers, multiplexers, and adders (in that order) dissipate more leakage power than smaller modules such as registers. It is seen that leakage power in modules such as multipliers, can contribute as much as 20-25% to the overall power. It is also observed from Fig. 4.14, that as one descends into deeper submicron technologies, the contribution of leakage power to the total power increases. The module-level leakage power savings in the datapath are shown in Tables 4.3 and 4.4.

A reduction in datapath leakage power of 18.22% and 12.94% in 180nm and 100nm technologies respectively is observed. The last three columns in the table give the number of MTCMOS modules in the data-path. It can be seen that the leakage power in the controller is very small compared to the total leakage power (datapath+controller). Hence, the increase in the controller leakage power is insignificant.

It is observed that the deterioration in performance is primarily due to the MTCMOS multiplier. On multicycling this multiplier over two clock cycles and introducing a one

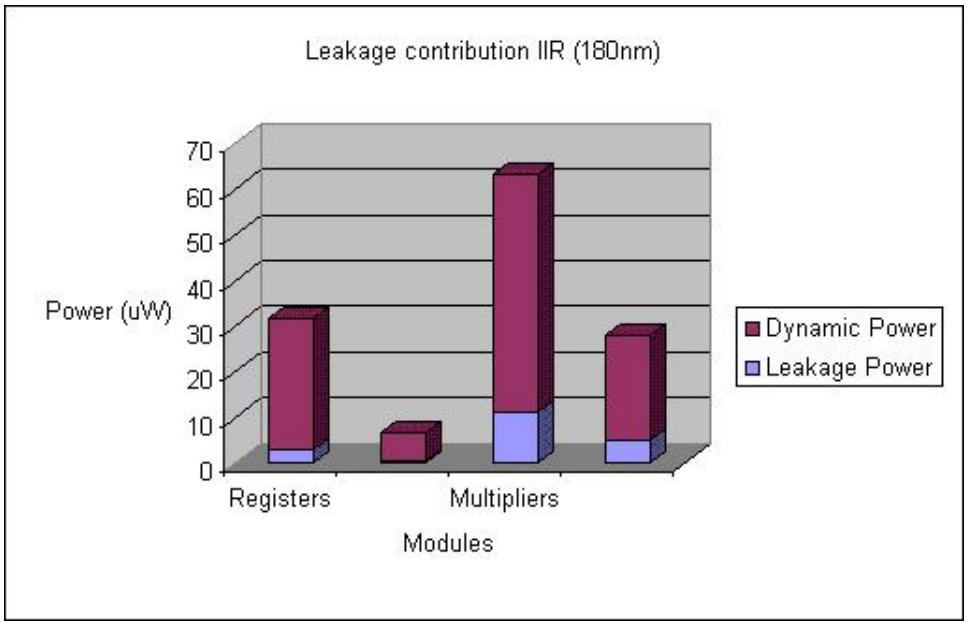


Figure 4.13. Leakage power contribution to overall power - IIR (180nm)

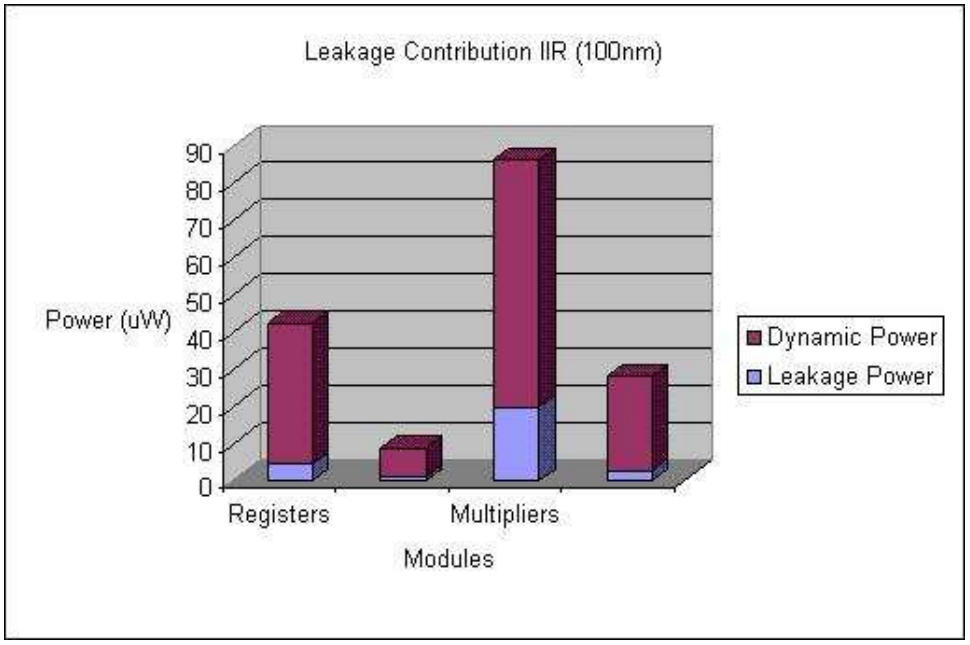


Figure 4.14. Leakage power contribution to overall power - IIR (100nm)

control step slack, the design could be made to run at 20 MHz clock. The penalty in execution time, in this case is 16.66%. A datapath leakage power reduction of 17.21% and

Table 4.3. Leakage power savings in IIR (180nm)

Module	Leakage energy(uW)		Savings	Module Distribution		
	Orig	MTCMOS	(%)	Instances	Non-MTCMOS	MTCMOS
Adders	0.5158	0.4577	11.19	2	1	1
Registers	2.8292	2.6894	4.94	10	3	7
Multipliers	11.3141	9.6362	14.83	5	0	5
Multiplexers	5.0866	3.3630	33.88	7	5	2
Data-path	19.7453	16.1463	18.22			
Controller	1.6645	2.9236	-75.6			

Table 4.4. Leakage power savings in IIR (100nm)

Module	Leakage energy(uW)		Savings	Module Distribution		
	Orig	MTCMOS	(%)	Instances	Non-MTCMOS	MTCMOS
Adders	0.8493	0.1923	77.35	2	1	1
Registers	4.5184	4.3061	4.69	10	3	7
Multipliers	19.6111	16.6775	14.96	5	0	5
Multiplexers	2.5919	2.8268	33.88	7	5	2
Data-path	27.5707	24.0027	12.94			
Controller	2.0401	3.8164	-87.06			

14.25% is observed in the 180nm and 100nm technologies respectively, in the performance recovered design.The module-level power savings in this new multicycled design is given in Tables 4.5 and 4.6.

Table 4.5. Leakage power savings in IIR with multicycling (180nm)

Module	Leakage energy(uW)		Savings	Module Distribution		
	Orig	MTCMOS	(%)	Instances	Non-MTCMOS	MTCMOS
Adders	0.5381	0.4475	16.83	2	1	1
Registers	2.9687	2.8639	3.53	10	3	7
Multipliers	11.3378	9.2798	18.15	5	0	5
Multiplexers	3.0836	2.2504	27.02	7	5	2
Data-path	17.9282	14.8416	17.21			
Controller	1.6548	1.6245	1.83			

Table 4.6. Leakage power savings in IIR with multicycling (100nm)

Module	Leakage energy(uW)		Savings	Module Distribution		
	Orig	MTCMOS	(%)	Instances	Non-MTCMOS	MTCMOS
Adders	0.8792	0.7984	9.19	2	1	1
Registers	4.5599	4.4203	3.06	10	3	7
Multipliers	19.4055	15.8842	18.14	5	0	5
Multiplexers	2.6665	2.4857	6.78	7	5	2
Data-path	27.5111	23.5886	14.25			
Controller	2.0937	1.7032	18.65			

The introduction of the MTCMOS modules leads to an area overhead of 23%. In this regard, it is apt to mention that current chip manufacturers are willing to tradeoff area for the sake of minimizing leakage power [64].

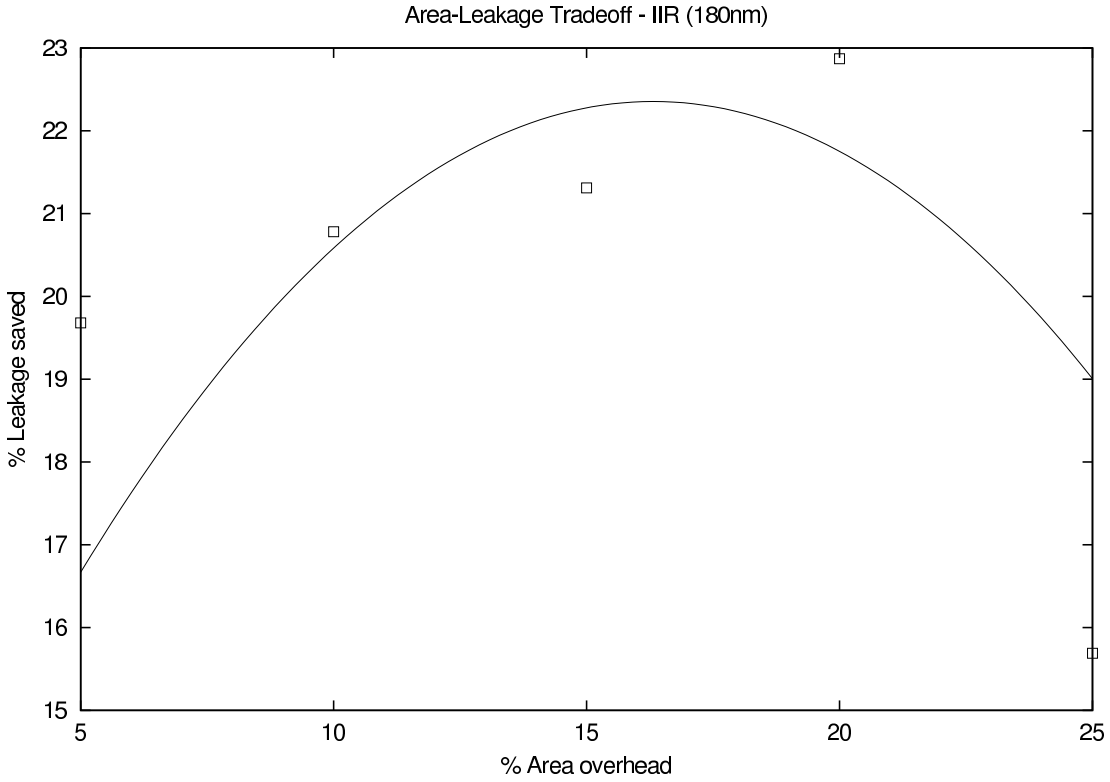


Figure 4.15. Area-Leakage power tradeoff analysis - IIR (180nm)

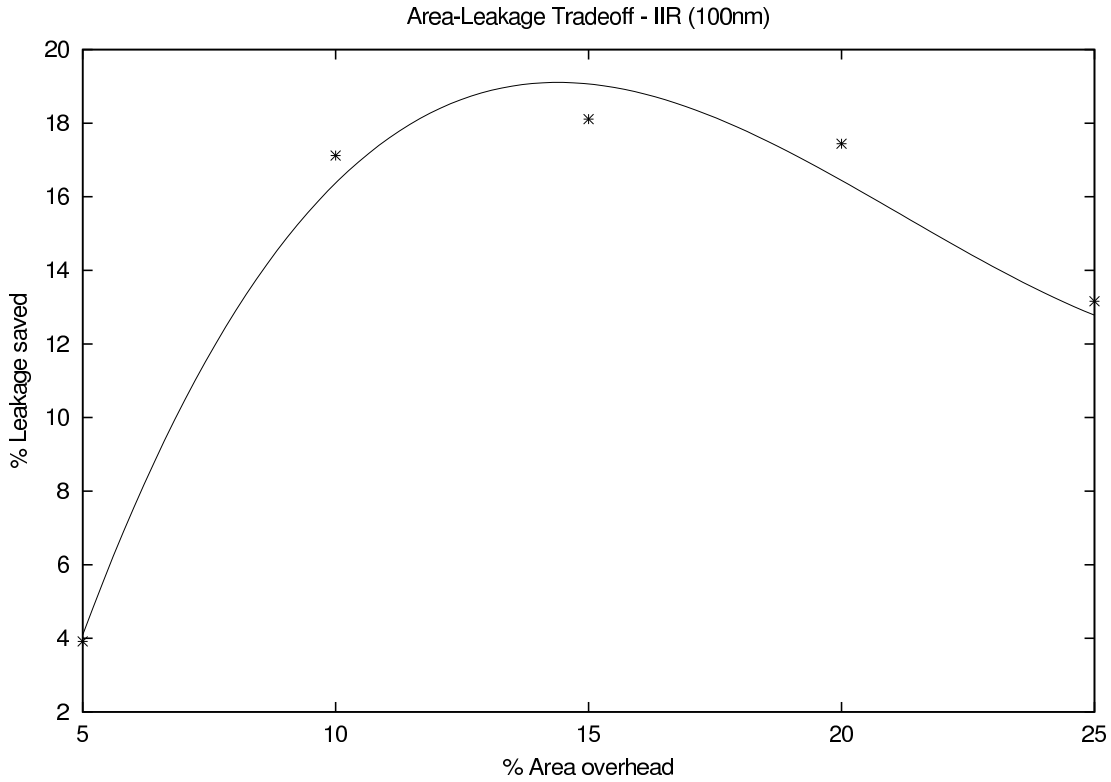


Figure 4.16. Area-Leakage power tradeoff analysis - IIR (100nm)

Figures 4.15-4.16 show the area-leakage power tradeoff analysis performed for the IIR filter. Leakage power savings for five different area penalties is presented. It is observed that the leakage power saved increases with more number of MTCMOS modules (increase in area). However, there is a threshold beyond which, the leakage power dissipated by the sleep transistors exceed the savings obtained. Table 4.7 lists the MTCMOS modules picked by the Knapsack algorithm for IIR.

4.6.2 FIR filter

The FIR filter has ten primary inputs and one primary output. The design consists of five multiplies and four adds. The ASAP schedule takes seven control steps and was run at 20 MHz and the leakage optimized design was run at 14.28 MHz clock. The performance penalty in this case is 40%. The contribution of the leakage power to the overall power in

Table 4.7. MTCMOS modules picked using the 0-1Knapsack for IIR (* - Multipliers ; r - Registers)

Area Overhead	MTCMOS modules	Total Modules	MTCMOS	Non-MTCMOS
5%	m4 (*)	17	1	16
10%	m0,m1,m4 (*)	17	3	14
15%	m0-m4 (*), r8 (r)	17	6	11
20%	m0-m4 (*), r5-r9 (r)	17	10	7
22%	m0-m4 (*), r3,r5-r9 (r)	17	11	6

the modules is shown in Figures 4.17-4.18. A datapath leakage power reduction of 20.59% and 14.57% is obtained in 180nm and 100nm technologies respectively. As in the case of IIR Filter, the controller leakage power contributes very little to the overall leakage power. Tables 4.8 and 4.9 shows the module-wise savings in leakage power and the MTCMOS module distribution. The area overhead due to the MTCMOS modules in this case is 22%.

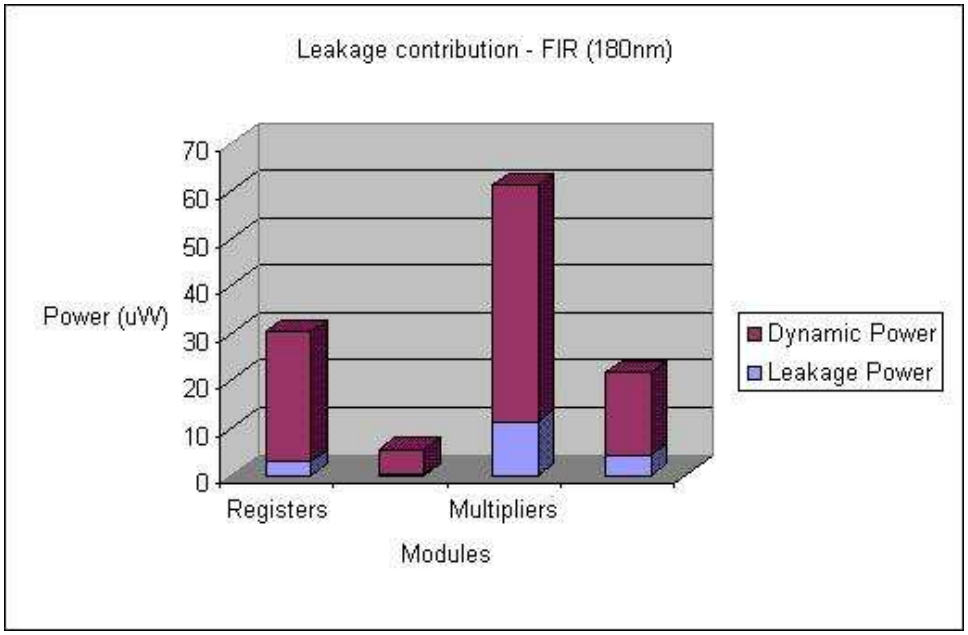


Figure 4.17. Leakage power contribution to overall power - FIR (180nm)

The FIR filter is re-synthesized for the multicycled design in a similar way as described for IIR. In this case, the introduction of one cycle slack results in a reduction of perfor-

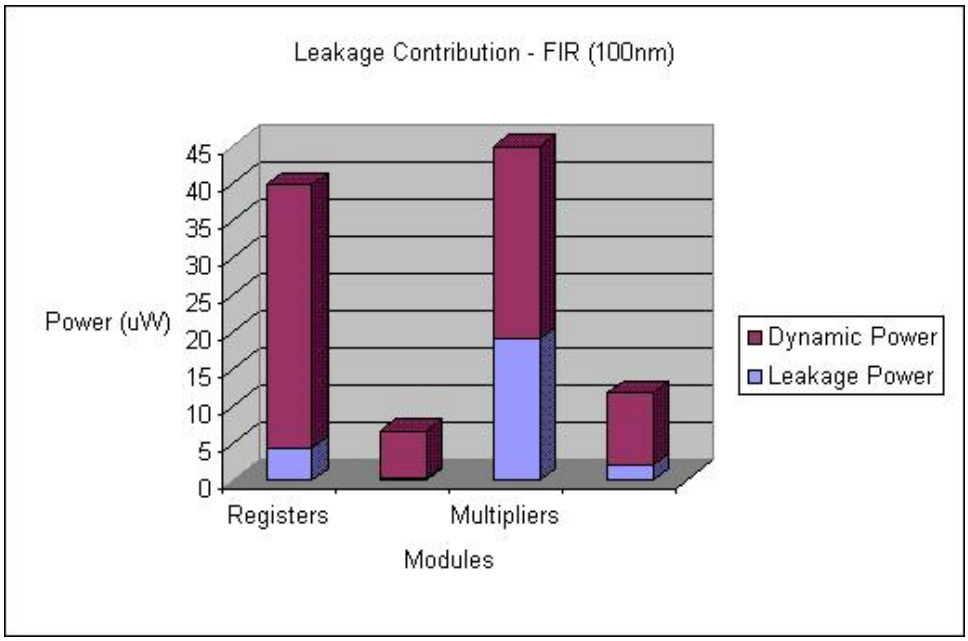


Figure 4.18. Leakage power contribution to overall power - FIR (100nm)

Table 4.8. Leakage power savings in FIR filter (180nm)

Module	Leakage energy(uW)		Savings	Module Distribution		
	Orig	MTCMOS	(%)	Instances	Non-MTCMOS	MTCMOS
Adders	0.2449	0.08586	64.94	1	1	0
Registers	2.8051	2.5561	8.87	10	3	7
Multipliers	11.2765	9.3894	16.73	5	0	5
Multiplexers	4.3132	2.7633	35.93	6	4	2
Data-path	18.6307	14.7946	20.59			
Controller	1.3347	1.9664	-47.32			

mance penalty from 40% to 14.28%. The leakage optimized design runs at the same clock frequency as the original design. The module-wise savings in leakage power is shown in Tables 4.10 and 4.11.

Figures 4.19-4.20 show the area-leakage power tradeoff analysis performed for the FIR filter. Leakage power was measured for 5 area penalties. Again, it is observed that the leakage power saved increases with increase in the number of MTCMOS modules. Table 4.12 lists the MTCMOS modules picked by the Knapsack algorithm for FIR.

Table 4.9. Leakage power savings in FIR filter (100nm)

Module	Leakage energy(uW)		Savings	Module Distribution		
	Orig	MTCMOS	(%)	Instances	Non-MTCMOS	MTCMOS
Adders	0.3964	0.0829	79.08	1	1	0
Registers	4.4320	3.9586	10.68	10	3	7
Multipliers	19.0154	15.9074	16.35	5	0	5
Multiplexers	2.1187	2.291	-8.13	6	4	2
Data-path	25.9625	22.178	14.57			
Controller	1.8205	2.9564	-62.39			

Table 4.10. Leakage power savings in FIR with multicycling (180nm)

Module	Leakage energy(uW)		Savings	Module Distribution		
	Orig	MTCMOS	(%)	Instances	Non-MTCMOS	MTCMOS
Adders	0.2589	0.0877	66.12	1	1	0
Registers	2.9095	2.8716	1.3	10	3	7
Multipliers	11.3224	9.7062	14.27	5	0	5
Multiplexers	2.5556	1.8997	26.05	6	4	2
Data-path	17.0464	14.8416	12.93			
Controller	1.2978	1.8809	-47.01			

Table 4.11. Leakage power savings in FIR with multicycling (100nm)

Module	Leakage energy(uW)		Savings	Module Distribution		
	Orig	MTCMOS	(%)	Instances	Non-MTCMOS	MTCMOS
Adders	0.4096	0.4046	1.22	1	1	0
Registers	4.4446	1.8468	58.43	10	3	7
Multipliers	19.6354	16.6694	15.10	5	0	5
Multiplexers	2.1269	1.9388	8.84	6	4	2
Data-path	26.6161	20.8596	21.62			
Controller	1.8474	2.9069	-57.35			

Leakage power savings for five different area penalties is presensted. It is observed that the leakage power saved increases with more number of MTCMOS modules (increase in area). However, there is a threshold beyond which, the leakage power dissipated by the sleep transistors exceed the savings obtained.

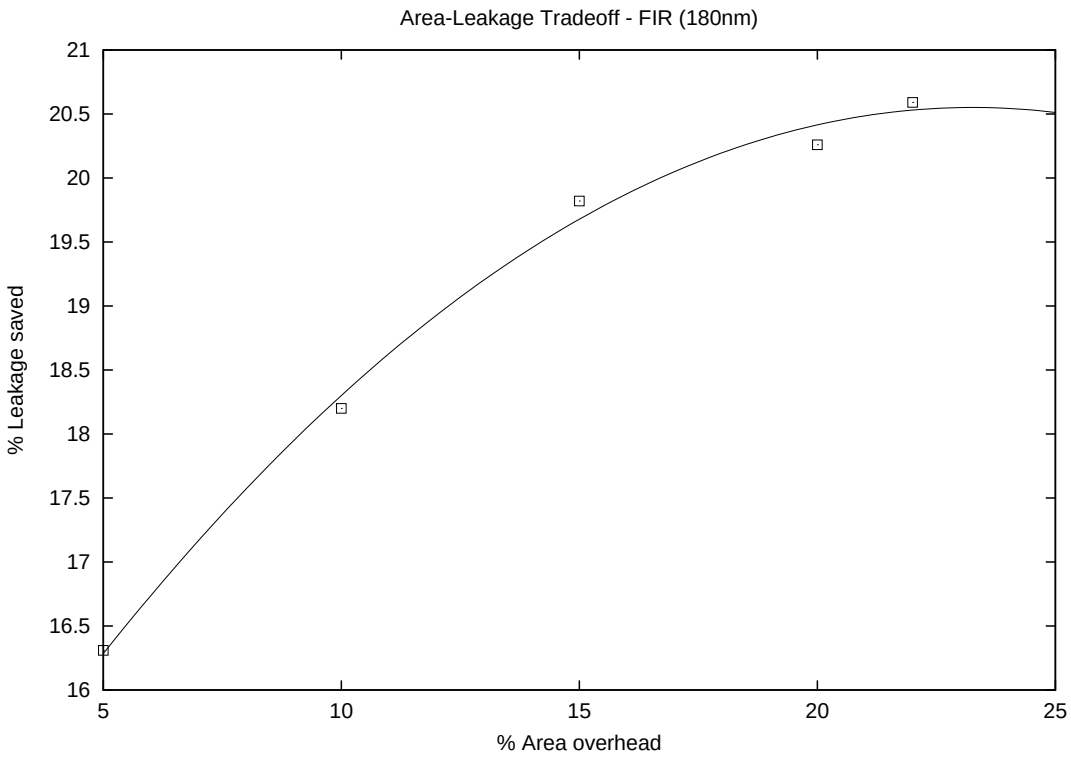


Figure 4.19. Area-Leakage power tradeoff analysis - FIR (180nm)

Table 4.12. MTCMOS modules picked using the 0-1Knapsack for FIR (* - Multipliers ; r - Registers)

Area Overhead	MTCMOS modules	Total Modules	MTCMOS	Non-MTCMOS
5%	m4 (*)	16	1	15
10%	m0,m1,m4 (*)	16	3	13
15%	m0-m4 (*), r8 (r)	16	6	10
20%	m0-m4 (*), r5-r9 (r)	16	10	6
22%	m0-m4 (*), r3,r5-r9 (r)	16	11	5

4.6.3 Elliptic Wave Filter (EWF)

The Elliptic wave filter consists of twenty six add operations and seven multiply operations. The ASAP schedule takes sixteen control steps. The distribution of MTCMOS modules to non-MTCMOS modules and the module level leakage power savings is shown in Tables 4.13 and 4.14. The original design is run at 20 MHz clock and the optimized

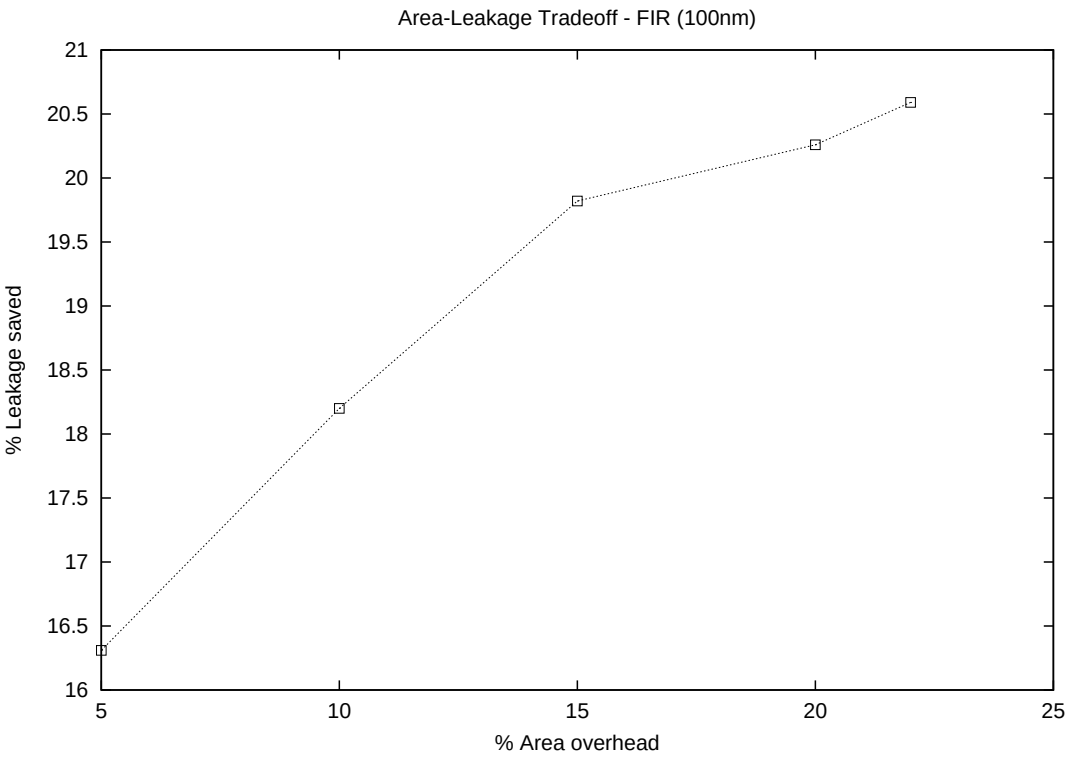


Figure 4.20. Area-Leakage power tradeoff analysis - FIR (100nm)

design was run at 14.28 MHz clock. Performance loss in this case is 40%. In this example, the allocation algorithm performed exceedingly well, reducing the number of multiplexers by 17.64%. This optimized sharing, led to a decrease in the area of the datapath modules of 7.12%. On closer analysis, it was found that the EWF DFG has a significant number of edges with non-overlapping lifetimes. The clique partitioner had produced tighter cliques, resulting in lesser number of multiplexers. This also has an effect in the controller leakage power.

The leakage contribution to the overall power of the modules is given in Figures 4.21-4.22. The controller power in this example is 38.33% of the total leakage power (datapath+control). This is due to the implementation of EWF using 4 bit modules. The 4 bit modules dissipate considerably lesser leakage than 8-bit modules. The ASAP schedule of the EWF has multiply operations in 4 control steps. The multiply operations in these

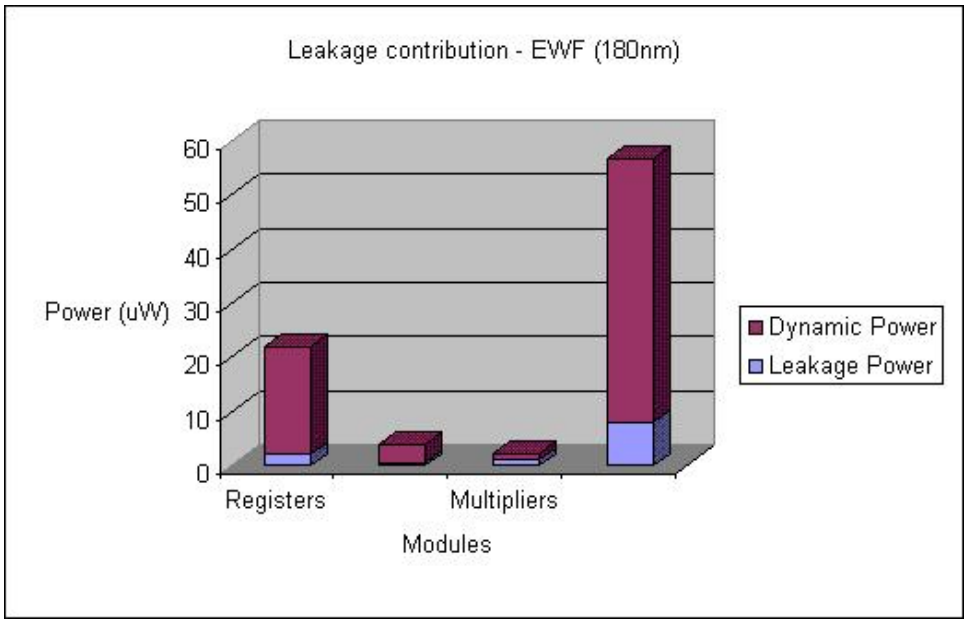


Figure 4.21. Leakage power contribution to overall power - EWF (180nm)

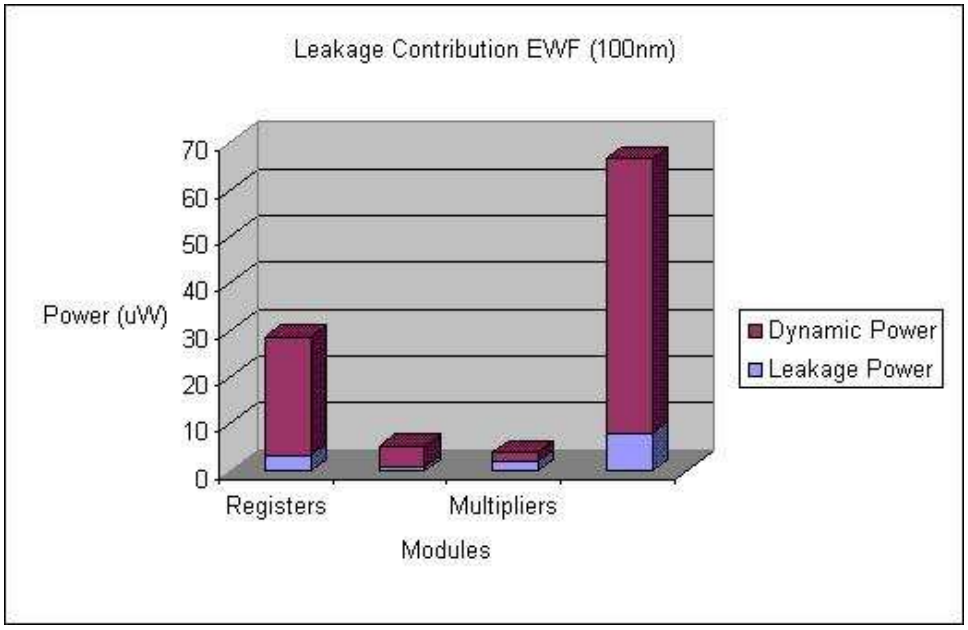


Figure 4.22. Leakage power contribution to overall power - EWF (100nm)

control steps are multicycled over 2 control steps and 4 cycles of slack are provided to the entire design. This reduces the performance penalty to 25%. The module level power

Table 4.13. Leakage power savings in EWF (180nm)

Module	Leakage energy(uW)		Savings	Module Distribution		
	Orig	MTCMOS	(%)	Instances	Non-MTCMOS	MTCMOS
Adders	0.4555	0.1677	63.18	4	2	2
Registers	1.9642	1.9571	0.36	14	11	3
Multipliers	1.0500	0.9999	4.77	2	0	2
Multiplexers	7.9567	5.0426	36.62	15	11	4
Data-path	11.4264	8.1673	28.52			
Controller	7.1030	5.9348	16.44			

Table 4.14. Leakage power savings in EWF (100nm)

Module	Leakage energy(uW)		Savings	Module Distribution		
	Orig	MTCMOS	(%)	Instances	Non-MTCMOS	MTCMOS
Adders	0.8002	0.2108	73.65	4	2	2
Registers	3.2387	1.9452	39.93	14	11	3
Multipliers	1.8793	0.9968	46.95	2	0	2
Multiplexers	8.0596	4.8657	39.63	15	11	4
Data-path	13.9778	8.0185	42.63			
Controller	6.0508	5.4761	9.44			

numbers are shown in Tables 4.15 and 4.16 . The reduction in control power is due to the more optimized sharing and lesser number of multiplexers.

Table 4.15. Leakage power savings in EWF with multicycling (180nm)

Module	Leakage energy(uW)		Savings	Module Distribution		
	Orig	MTCMOS	(%)	Instances	Non-MTCMOS	MTCMOS
Adders	0.4660	0.1603	65.60	4	2	2
Registers	2.0129	1.9459	3.32	14	11	3
Multipliers	1.4629	0.9826	32.83	2	0	2
Multiplexers	7.9984	6.1937	22.56	15	11	4
Data-path	11.9404	9.2826	22.26			
Controller	9.2936	2.3637	74.57			

Figures 4.23-4.24 shows the area-leakage power tradeoff analysis performed for the EWF filter. Leakage power was measured for four area penalties. In this case, the design was

Table 4.16. Leakage power savings in EWF with multicycling (100nm)

Module	Leakage energy(uW)		Savings	Module Distribution		
	Orig	MTCMOS	(%)	Instances	Non-MTCMOS	MTCMOS
Adders	0.8413	0.1437	52.07	4	2	2
Registers	3.2835	2.5162	23.36	14	11	3
Multipliers	2.6180	1.7176	34.39	2	0	2
Multiplexers	8.0669	5.1386	36.30	15	11	4
Data-path	11.9404	9.2826	38.06			
Controller	4.3665	4.1600	4.73			

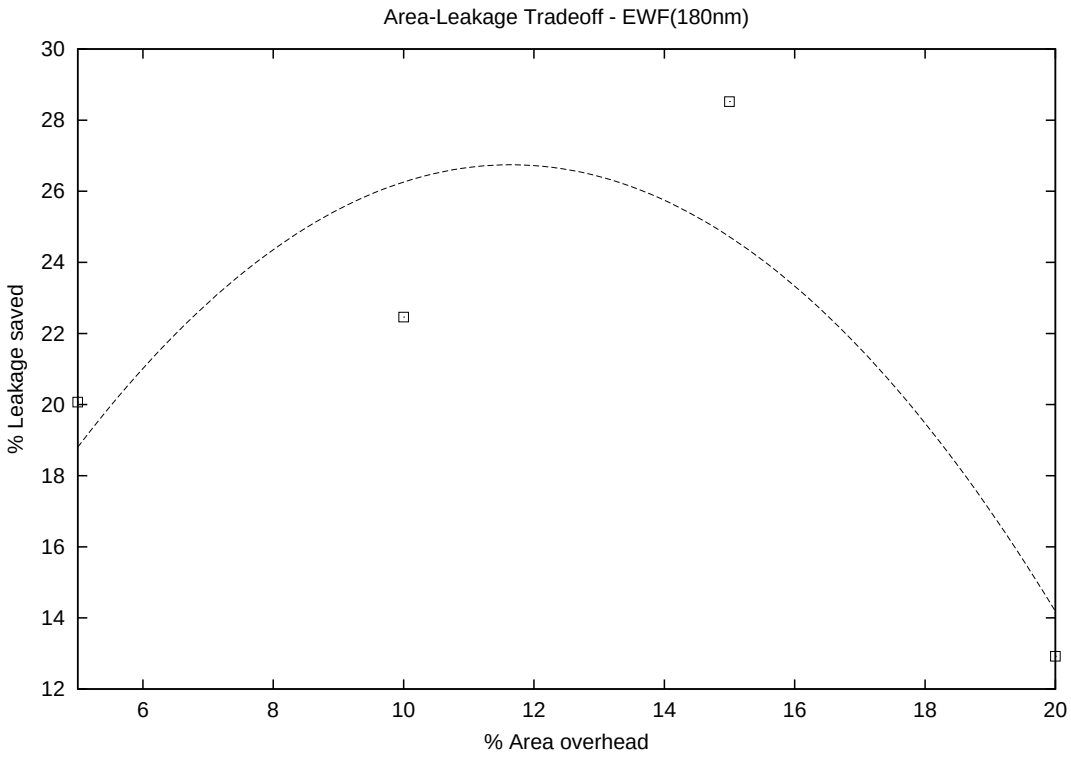


Figure 4.23. Area-Leakage power tradeoff analysis - EWF (180nm)

implemented using 4-bit modules. Table 4.18 lists the MTCMOS modules picked by the Knapsack algorithm for EWF.

The focus of the proposed approach was to synthesize low leakage power data-paths. However, in most simulation runs, a reduction in the dynamic power was observed. The reduction is primarily due to the shutting `OFF` of modules when they are idle. The dynamic

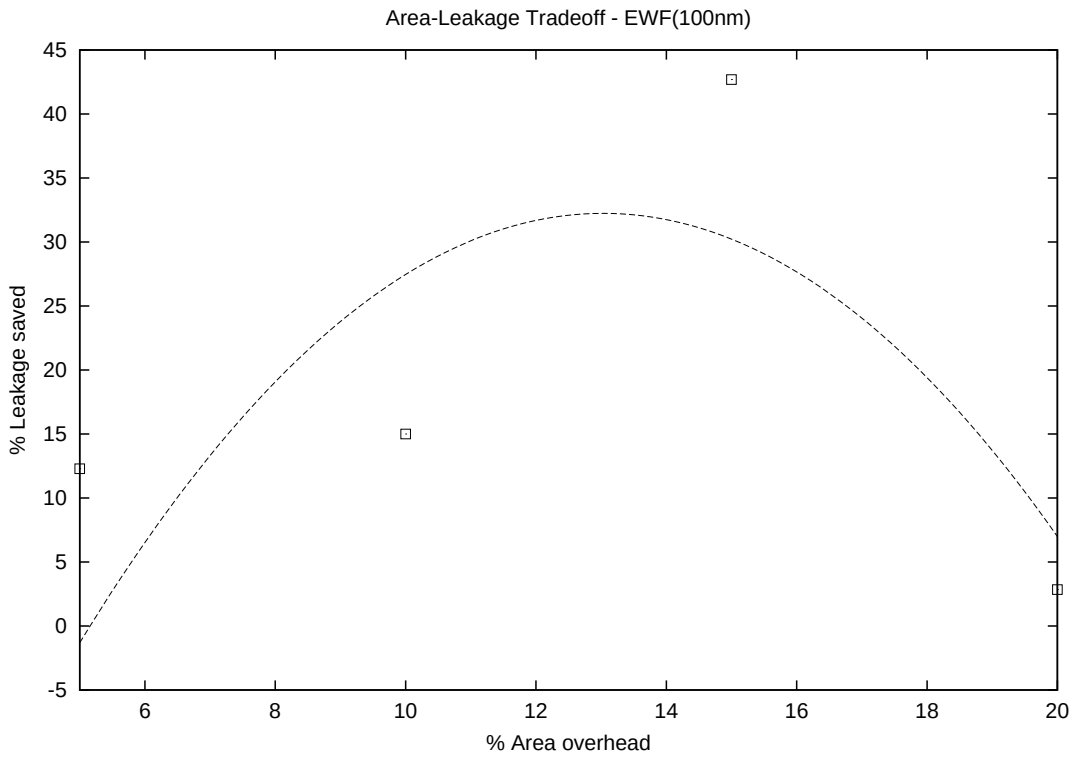


Figure 4.24. Area-Leakage power tradeoff analysis - EWF (100nm)

Table 4.17. MTCMOS modules picked using the 0-1 Knapsack for EWF (* - Multipliers ; + - Adders ; r - Registers)

Area Overhead	MTCMOS modules	Total Modules	MTCMOS	Non-MTCMOS
5%	m0,m1(*)	20	2	18
10%	m0,m1(*),a2,a3(+),r8,r13(r)	20	6	14
15%	m0,m1(*),a2,a3(+),r5,r8,r9,r13 (r)	20	8	12
20%	m0,m1(*),a1-a3(+),r1,r3,r5,r8-r10,r13(r)	20	12	8

Table 4.18. MTCMOS modules picked using the 0-1 Knapsack for EWF (* - Multipliers ; + - Adders ; r - Registers)

power of the designs (Datapath + controller) was measured using Synopsys Nanosim. The results for 180nm and 100nm design runs are reported in Tables 4.19 and 4.20. The incorporation of the power management logic does lead to increased controller complexity. This increased complexity is handled by merging identical control lines. This simple, yet

Table 4.19. Dynamic power (180nm)

Module	Regular MTCMOS			Multicycled MTCMOS		
	Orig(uW)	Ours(uW)	Savings(%)	Orig(uW)	Ours(uW)	Savings(%)
IIR	224.09	178.10	20.52	243.29	219.37	9.83
FIR	221.01	168.55	23.73	224.57	175.55	21.82
EWF	257.72	206.59	19.83	301.84	214.46	28.94

powerful controller optimization technique, reduces the controller power overhead considerably. This observation is supported strongly by the reduction in the total dynamic power of the designs.

Table 4.20. Dynamic power (100nm)

Module	Regular MTCMOS			Multicycled MTCMOS		
	Orig(uW)	Ours(uW)	Savings(%)	Orig(uW)	Ours(uW)	Savings(%)
IIR	188.89	168.84	10.61	244.90	243.32	0.64
FIR	221.88	193.50	12.79	227.13	197.37	13.10
EWF	261.93	234.89	10.32	313.72	222.26	29.15

A reduction in dynamic power in all the simulation runs is observed. However, in the case of multicycled designs, the dynamic power dissipated was observed to be more than the regular MTCMOS design. Consequently the reduction in dynamic power was smaller as compared to the regular designs. The Elliptic Wave filter is an anomaly, because of the rigid nature of its data-path. In this case, multi-cycling of four time-steps in which the multipliers are used, result in increased power reduction.

4.7 Summary

In this chapter, a novel leakage power management technique using MTCMOS technology was presented. A characterized MTCMOS module library was developed. Modules are shut down during time steps in which they they do not perform useful computation or store valid data. A resource allocation and binding algorithm based on clique parititioning was proposed which increases the idle times of the modules bound. Significant savings in leakage as compared to the normally used greedy algorithms for allocation and binding have

been observed. It is also observed that the mapping onto a library of MTCMOS modules also optimized overall power. To overcome the area overhead incurred due to the bulky sleep transistors, a selective binding approach was proposed. Multicycling of large modules was proposed to overcome the performance penalty incurred due to the introduction of MTCMOS modules.

CHAPTER 5

FAST ARCHITECTURAL SIMULATOR FOR LEAKAGE POWER (FASL)

Accurate estimation of leakage power using HSPICE is slow and tedious. Large designs such as data-paths consist of tens of thousands of transistors. Simulating such designs would prove unmanagable using HSPICE. Existing tools such as Synopsys Nanosim do not report true leakage power. Nanosim reports wasted power as the difference between total power dissipated and the dynamic power [65]. However this includes other components of power such as short circuit power. In this chapter, a Fast Architectural Simulator for Leakage power (FASL) is presented, which provides an accurate estimate of the leakage power dissipated by a design represented at the RT-Level of abstraction. Modules in the RT-Level description of the data-path are described hierarchically in VHDL.

Section 5.1 describes characterization of leafcells using which components in the module library are described. Section 5.2 explains the procedure used to accurately estimate leakage power dissipated by a design, using HSPICE. Section 5.3 introduces the estimation of leakage power of data-paths and random logic circuits using FASL. Section 5.4 presents experimental results.

5.1 Characterization of leaf cells

Halter and Najm [20] establish that the leakage power in a circuit is dependent on the values at the inputs to the circuit. On exhaustive simulation of the leaf cells, it is observed that, this is true only after a certain period of time. For accurate characterization, this time is divided into the transient period and the steady-state period. It is seen that the leakage power dissipated by the circuit during the transient period, is dependent on the previous value of inputs to the circuit. During this transient time interval, drain-source

voltages in the transistors get changed by the new input applied. During this settling time, the leakage power dissipated by the transistors (which is related exponentially to V_{ds}), is a function of the previous input and the current input to the leaf cell. The leakage power equation (Eqn. 2.1) is repeated here for convenience.

$$I_{sub} = A.e^{\frac{q}{n'kT}.(V_{gs}-V_T-\gamma' V_s+\eta V_{ds})}.(1-e^{\frac{-qV_{ds}}{kT}}) \tag{5.1}$$

where $A = \mu_0.C_{ox}.\frac{W_{eff}}{L_{eff}}.(\frac{kT}{q})^2.e^{1.8}$, C_{ox} is the gate oxide capacitance per unit area. μ_0 is the zero bias mobility. n' is the sub-threshold swing coefficient of the transistor. V_T is the zero bias threshold voltage, V_s is the source voltage, V_{gs}, the gate-source voltage, and V_{ds}, the drain-source voltage. γ' is the linearized body effect coefficient and η is the Drain Induced Barrier Lowering (DIBL) coefficient [3].

Fig. 5.1 shows typical leakage power profiles for an NMOS transistor. Similar profiles are observed for a PMOS transistor. The profiles represent the two different types of curves obtained during the exhaustive simulation of leaf-cells. The leakage current equation (Eqn. 2.1) is a product of two exponential terms, involving the gate-source voltage (V_{gs}) and the drain-source voltage (V_{ds}) respectively. The two plots represent the two situations:

- Gate voltage (V_g) = 0V. Drain Voltage (V_d) rises from 0V to V_{dd}. In this case, the $e^{V_{gs}}$ term dictates the rising exponential curve

- Drain-source voltage (V_{ds}) = V_{dd}. Gate Voltage (V_g) rises from V_{dd} to 0V. In this case, $(1-e^{\frac{-qV_{ds}}{kT}})$ term dictates the falling exponential curve

Characterization plots are obtained for each combination of previous and current inputs. For each combination, the threshold time that separates the transient stage and the steady-state stage is identified using the algorithm shown in Fig. 5.2. The leakage power dissipated during each nanosecond interval is obtained using HSPICE simulations. From this data, the mean leakage power (P_{avg}) over the time period of simulation T is calculated (Line 3). The cumulative leakage energy at any time instant t is obtained by summing up the instantaneous leakage power values until time t (Line 5). The mean squared error

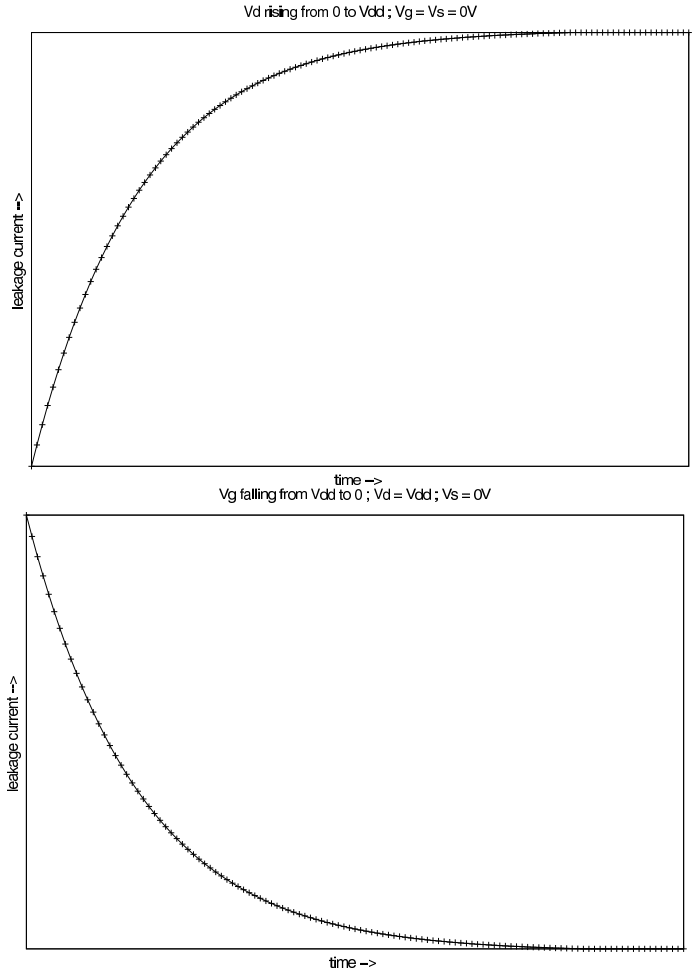


Figure 5.1. Typical leakage power profiles (NMOS transistor)

between the cumulative leakage power and the average power P_{avg} is calculated (Line 7). We determine the threshold time as the time when this mean squared error is consistently less than 5%, ie., the cumulative leakage power settles down close to P_{avg}(Line 11-13). The average leakage power during the two intervals on either side of the threshold time is measured using HSPICE and entered in the look-up table as shown in Fig.5.8. The leaf cells such as `NAND2` or the `full adder` are small circuits. Exhaustive characterization of leakage power using HSPICE for the leaf cells, can be done in reasonable time, as compared to the complete modules themselves. Table 5.1 give the complexity of the modules in terms of number of transistors and the runtimes taken for characterization using HSPICE. It is

```
1 procedure get_threshold_time(spice_netlist)
2   foreach input combination(prev_ip,curr_ip)
3     Calculate Avg leakage power for interval (0,T) = P_avg
4     foreach nanosecond interval [t,t+1] in t ∈ (0,T)
5       Calculate leakage power at time t = P^t = Σ_{i=0}^{t} P_i/t
7       Calculate mean squared error (MSE) = | P^t − P_avg |^2
9       /* The threshold has been reached when the P^t
10            settles down close to the P_avg */
11       if ((MSE(t) < 5%) and (MSE(t-1) < 5%))
12            print threshold (curr_ip) = t
13             exit
14       endif
15     endfor
16   endfor
17   repeat for all curr_ip exhaustively
18 return max(threshold(spice_netlist))
```

Figure 5.2. Pseudo-code to determine threshold time between transient state and steady State

observed that as the complexity of the module increases, the time taken for the HSPICE simulation run increases.

Table 5.1. Modules in the data-paths - complexity and HSPICE run times

Module	Size (transistors)	HSPICE simulation run time (s)	Leaf Cells Used
Register (8 bit)	238	15.17	D-flipflop, Inverter
Adder (8 bit)	224	10.92	Full Adder
Multiplier (8 bit)	1630	224.48	Full Adder, NAND
Multiplexer 2x1 (8 bit)	102	6.77	Multiplexer 2x1 (1 Bit)

Figures 5.3 - 5.6 shows a subset of the characterization plots for the leaf cell `faf001`, which is a Full adder, provided as part of the LAGER IV Silicon Compiler [58]. This is the basic building block of the `n-bit` Adder module. The threshold time that separates the transient and the steady state time is observed to be 27 ns. Similar characterizations were done on the basic gates and other modules in the data-path. HSPICE simulations

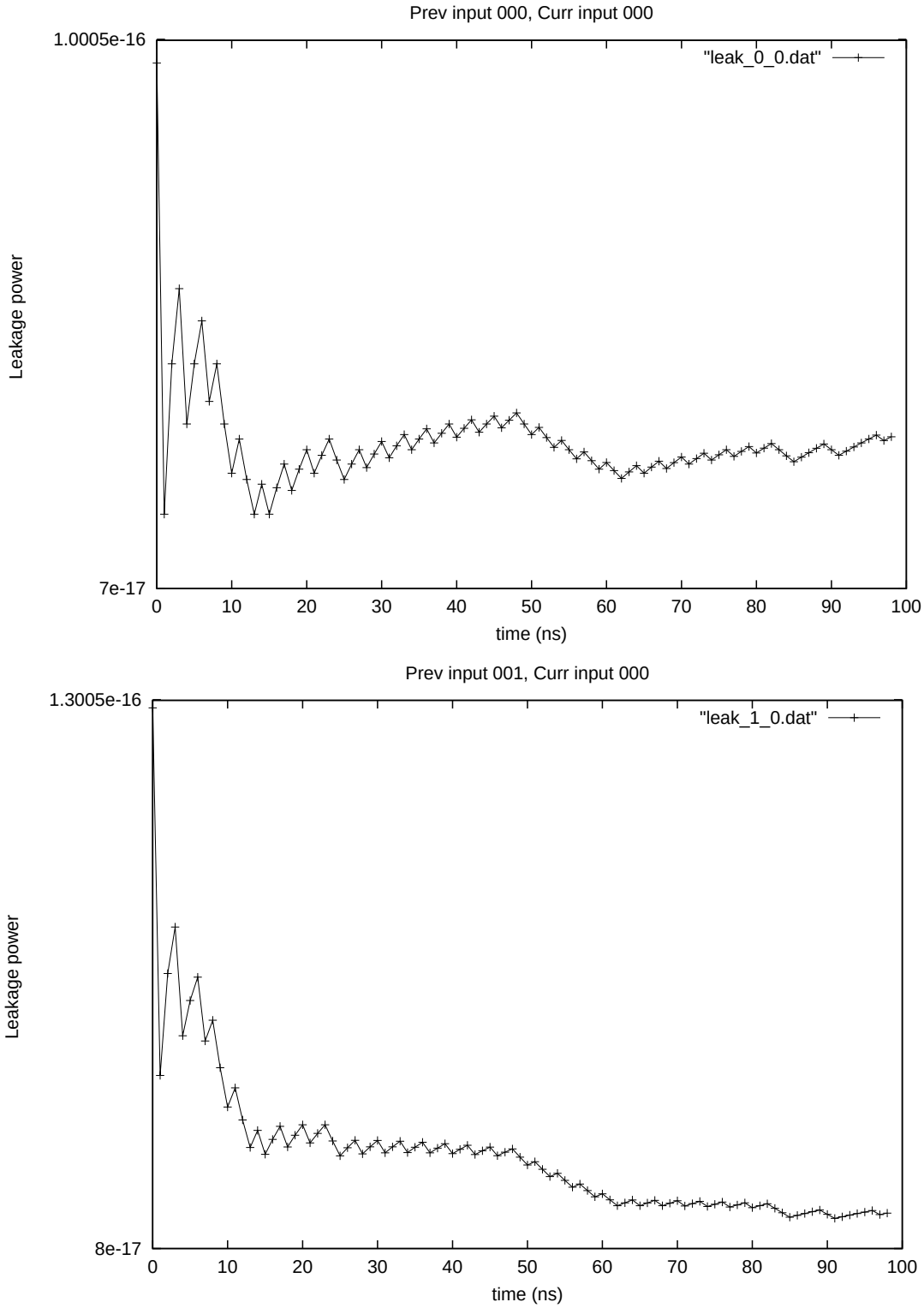


Figure 5.3. Instantaneous leakage power plots for a few input combinations for full adder for 100nm CMOS technology model

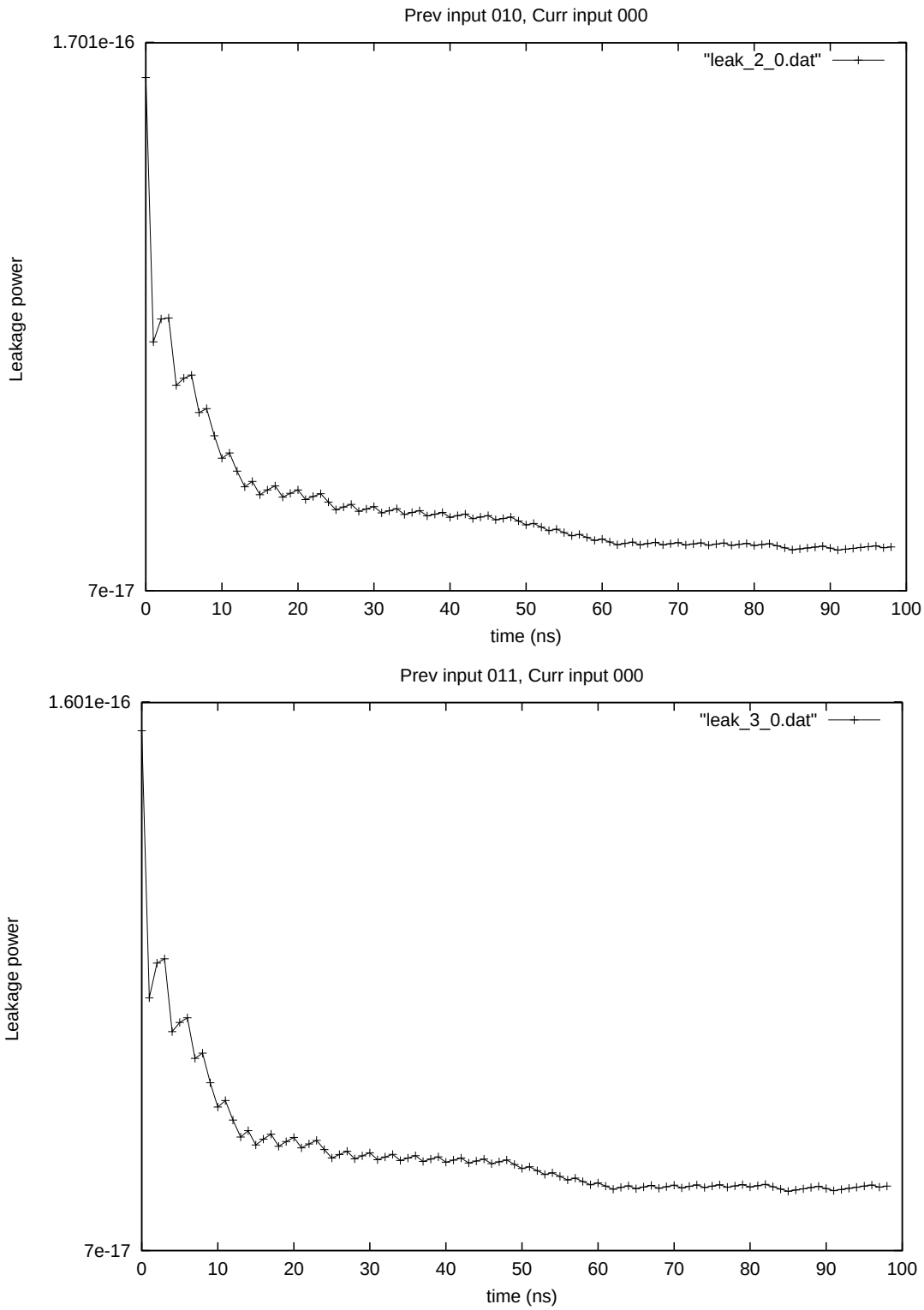


Figure 5.4. Instantaneous leakage power plots for a few input combinations for full adder for 100nm CMOS technology model

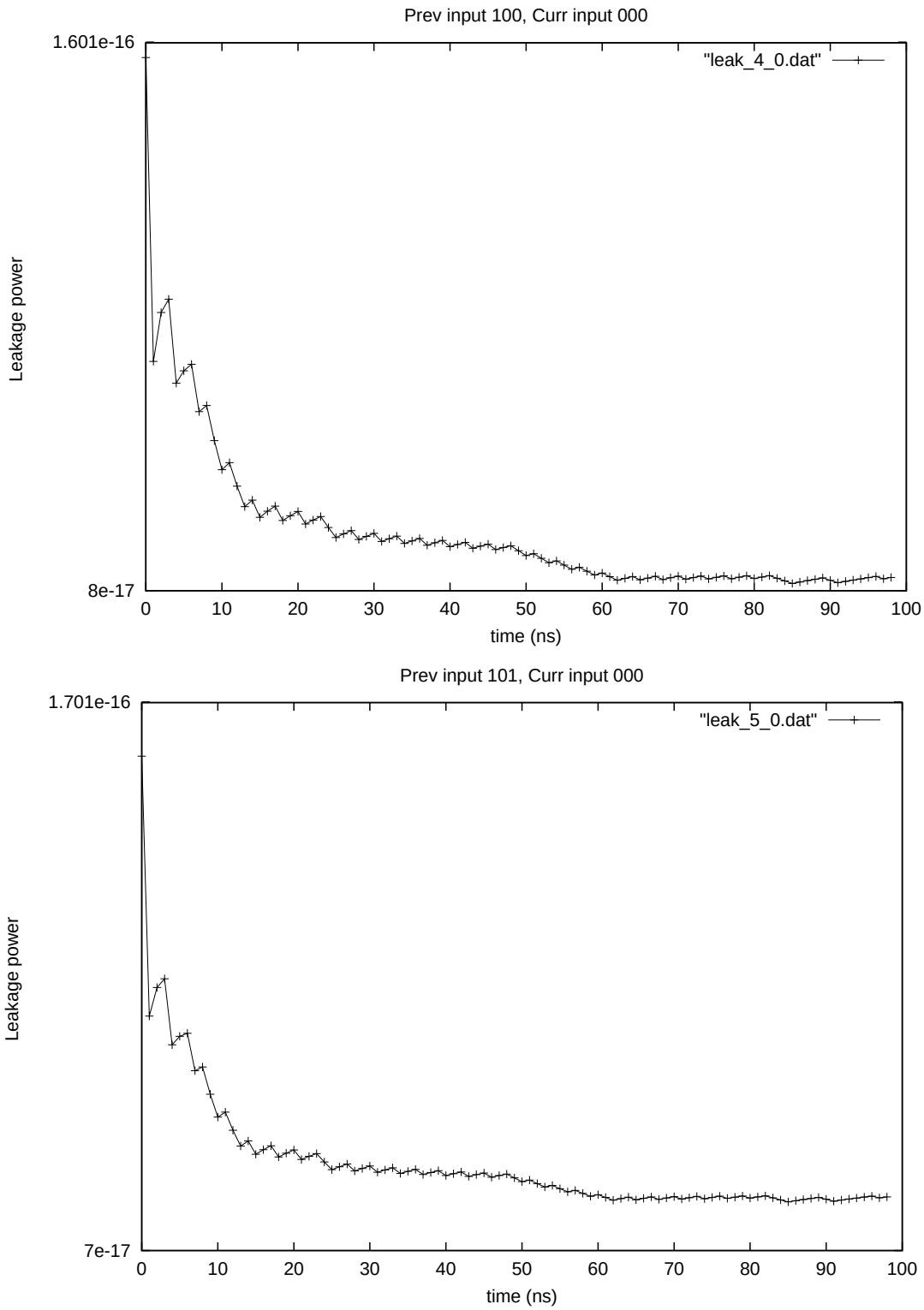


Figure 5.5. Instantaneous leakage power plots for a few input combinations for full adder for 100nm CMOS technology model

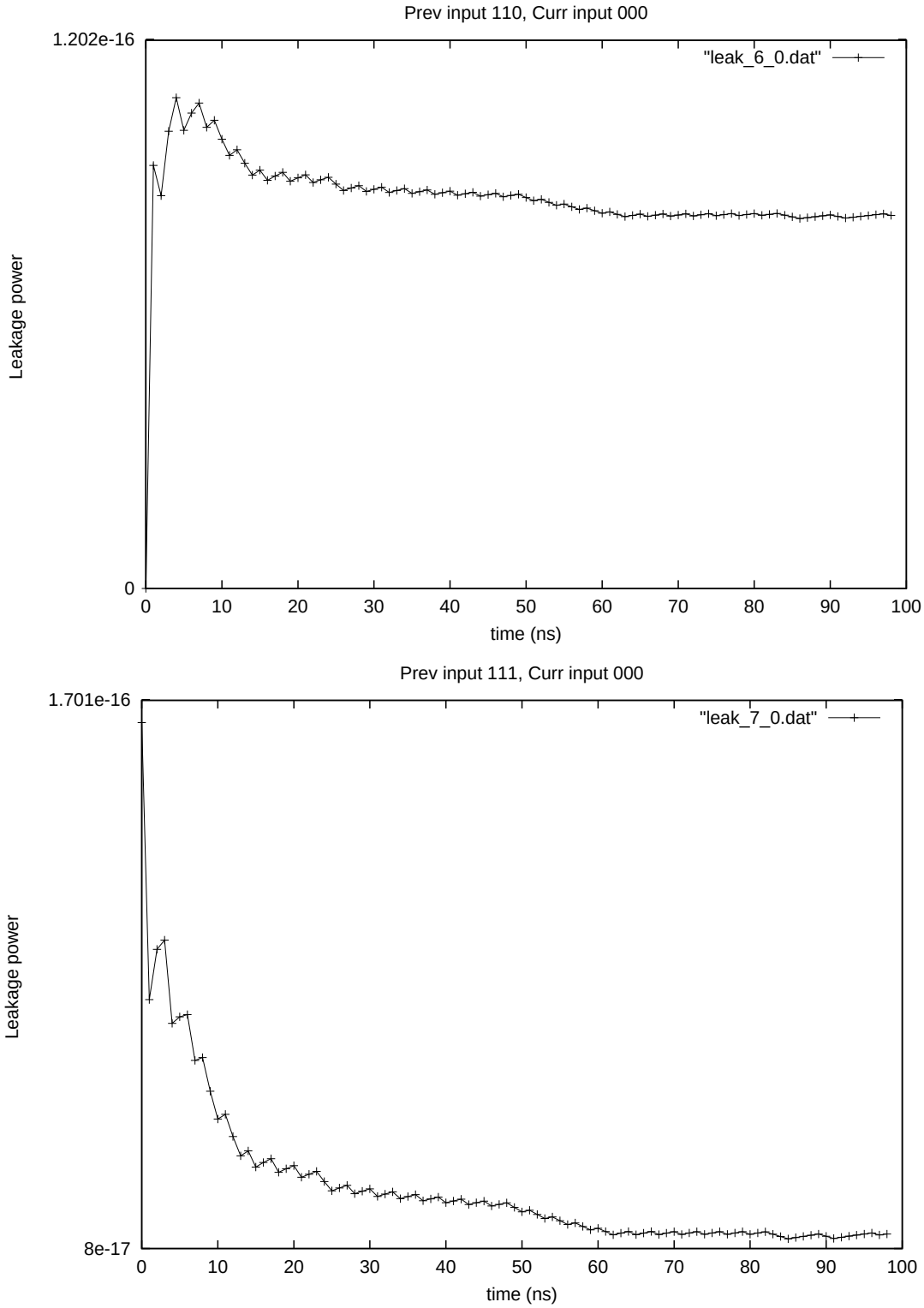


Figure 5.6. Instantaneous leakage power plots for a few input combinations for full adder for 100nm CMOS technology model

were done using a 100nm 1V technology model obtained from the Berkeley Predictive Technology Model group [63].

5.2 Leakage power measurement using HSPICE

For the sake of completeness, the methodology for measuring leakage power using HSPICE is described in this section. The input to HSPICE is modified by adding commands to print: (1) Voltage at each terminal of each transistor, and (2) Instantaneous power of each transistor for each nanosecond interval.

The procedure to calculate the leakage power is summarized in Fig. 5.7. During every nanosecond interval, the terminal acting as source changes due to the voltage differences between the terminals. This is determined as shown in lines (5-10) in the case of a PMOS transistor and lines (15-20) in the case of an NMOS transistor. The state of the transistor is then determined by comparing the V_{gs} and the threshold voltage of the technology (lines 7-15). If the transistor is determined to be in its `OFF` state, the instantaneous power dissipated by the transistor is accumulated. This procedure has been coded in UNIX SHELL script and run as a post-processor to HSPICE.

5.3 Estimation of leakage power in data-paths and random logic

The module library consists of basic logic gates, storage units such as registers, functional units (FUs) such as adders and multipliers, and interconnect units such as multiplexers. There exists two architectures for each of the modules - the 1-bit architecture and the n-bit architecture. Fig. 5.8 shows the VHDL description of Full Adder. The n-bit architecture is obtained using the `GENERATE` statements, depending on the parameters (usually the bit width) supplied to it. The 1-bit architecture itself consists of two parts -

- A behavioral process which implements the functionality of the leaf cell (lines 11-15)
- A process which accumulates the leakage energy dissipated by the leaf cell. This process is sensitive to the inputs to the leaf cell (lines 34-53)

```
1 Procedure get_leakage(netlist N, threshold voltage Vth, transient time T )
2 foreach nanosecond interval in 0 to T
3     foreach transistor m(t1, gate, t2) in N
4         if (m is PMOS)
5            if (Vt1 > Vt2)
6            then
7               source ← t1
8            else
9               source ← t2
10           endif
11           if ((Vgate - Vsource) > (-1 *  Vth))  /* Transistor is off */
12                 leakage energy ← leakage energy + [inst.power(m) * 1ns]
13           endif
14        elseif (m is NMOS)
15            if (Vt1 < Vt2)
16            then
17               source ← t1
18            else
19               source ← t2
20            endif
21            if ((Vgate - Vsource) < Vth)  /* Transistor is off */
22               leakage energy  ← leakage energy + [inst.power(m) * 1ns]
23            endif
24        endif
25     endfor
26 endfor
27 Leakage power  ← (leakage energy)/T
28 end procedure
```

Figure 5.7. Calculating leakage power from HSPICE raw data

The leaf cells are characterized as described in Section 5.1. The transient leakage power measured is entered into the `leakage1` constant array (Lines 21-23), and is indexed by the 2-tuple (previous input, current input). The steady state leakage power is entered into the `leakage2` constant array (lines 28-29). The threshold time separating the transient and the steady state periods was calculated using the algorithm described in the previous section. The threshold time in the case of the full adder was found to be 27 ns. This can be confirmed visually using the Figures 5.3-5.5. Thus, the leakage energy accumulation is split into two parts - lesser than and greater than 27 ns (Lines 36-44). The total time is accumulated in

```
01 library IEEE;
02 use ieee.std_logic_1164.ALL;
03 entity faf001 is
04 port(A1: in Bit; B1: in Bit; CIN: in Bit;
05        SUM: out Bit; CO: out Bit );
06 end faf001;
07
08 architecture behavior of faf001 is
09 <type declarations for constant arrays>
10 begin
11    add : process(A1,B1,CIN) -- implements full adder functionality
12    begin
13        SUM <=  (A1 xor B1 xor CIN) ;
14        CO  <=  ((A1 and B1) or ((A1 xor B1) and CIN));
15    end process add;
16
17    leak_print: process(A1,B1,CIN)
18    <variable declarations>
19    function leak_table1(X,Y: NATURAL) return real is
20    constant leakage1 : array_8x8 := ( -- transient leakage table
21    (   ..     ..                                        .. ),
22    (    :                                               ),
23    (   ..                                          ..  .. ));
24    begin
25         RETURN leakage1(X,Y);
26    end leak_table1;
27    function leak_table2(X: NATURAL) return real is
28    constant leakage2 : array_8x8 := (  -- steady state leakage table
29    (   ..     ..                                        .. ));
30    begin
31         RETURN leakage2(X);
32    end leak_table2;
33    begin                -- implements leakage accumulation
34         if ((A1'EVENT) or (B1'EVENT) or (CIN'EVENT)) then
35            if (now /= prev_time) then
36               if ((now - prev_time) > 27 ns) then
37                   leak := leak + time_to_real(27 ns) *
38                       leak_table1(bits_to_int(prev_bus),bits_to_int(pprev_bus)) ;
39                   leak := leak + time_to_real((now- prev_time - 27 ns)) *
40                       leak_table2(bits_to_int(prev_bus)) ;
41               else
42                   leak := leak + time_to_real(now - prev_time) *
43                       leak_table1(bits_to_int(pprev_bus),bits_to_int(prev_bus)) ;
44               end if;
45               total_time := total_time + (now-prev_time) ;
46            else
47               prev_bus := CIN&B1&A1;
48            end if;
49            prev_time := now;
50            pprev_bus := prev_bus;
51            prev_bus := CIN & B1 & A1;
52         end if;
53    end process leak_print;
54 end Behavior;
```

Figure 5.8. Leakage model of full adder in VHDL

the `total_time` variable, which is necessary for determining the leakage *power* at the end of the simulation. The functions *bits_to_int* and *time_to_real* are precompiled functions for type conversion.

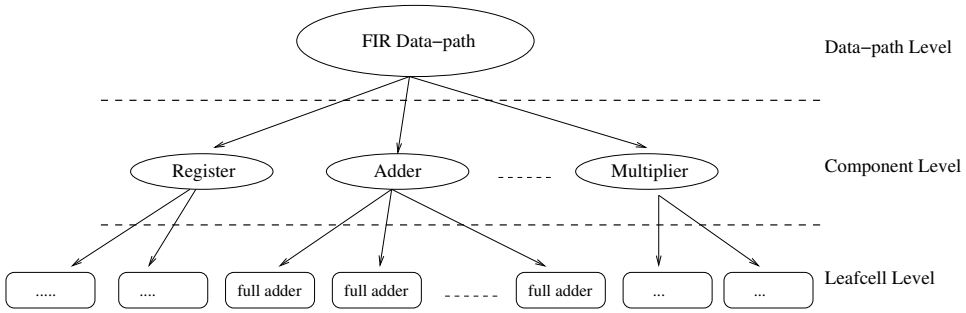


Figure 5.9. Hierarchy in bit-sliced designs

The n-bit architectures, are derived using these 1-bit architectures and glue logic such as `NAND` gates, using generate statements. Due to the bit-sliced design of these components, simulating the top level component VHDL file would result in simulating the leafcells, with appropriate inputs. This is taken care of by the VHDL simulator (NCLAUNCH in our case). Data-paths in turn consist of these instances connected appropriately (Fig. 5.9) using signals. Hence simulating the top level data-path design in VHDL would *percolate* the values on the primary inputs all the way down to the leafcells. Hence it is enough to incorporate our leakage power measuring logic only at the leafcell level. Random logic circuits are synthesized using basic logic gates that are pre-characterized for leakage power. VHDL descriptions of the circuits are required. Figures 5.10-5.12 show the absolute percentage error that is incurred when the components are simulated for different input vectors. It is observed that the error is approximately constant, certifying that our characterization of the module is accurate.

5.4 Experimental results

Results are presented for three DSP benchmarks synthesized with 100 nm 1V technology models [63]. The leafcells are accurately characterized exhaustively simulation using

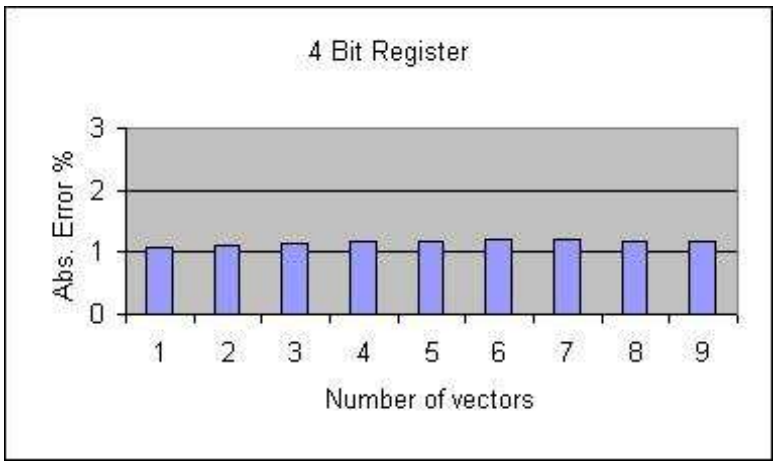


Figure 5.10. Absolute percentage error vs. number of vectors plot for register

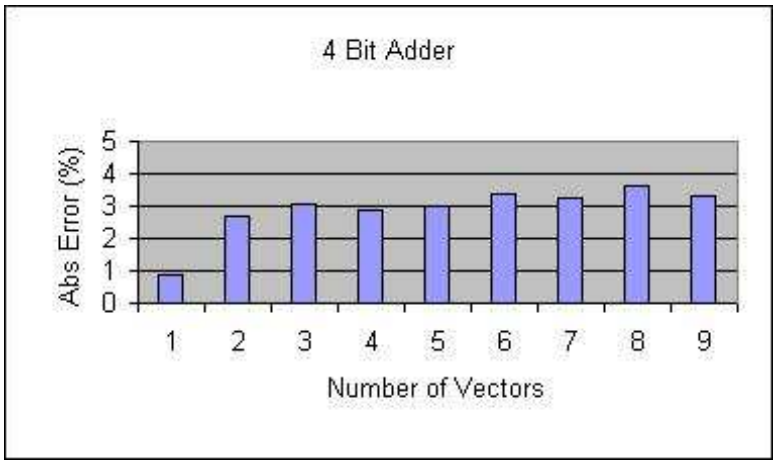


Figure 5.11. Absolute percentage error vs. number of vectors plot for adder

HSPICE. The leakage power is estimated for the DSP designs for 10 vectors. The number of vectors is limited by the long simulation times for comparison with HSPICE. The simulator is capable of estimating leakage power for random logic benchmarks as well. Results are presented for 20 MCNC benchmark circuits described in structural VHDL.

5.4.1 Data-path examples

The three DSP designs are generated using the AUDI synthesis system [66]. The system generates structural VHDL consisting of components described in Section 5.1. The

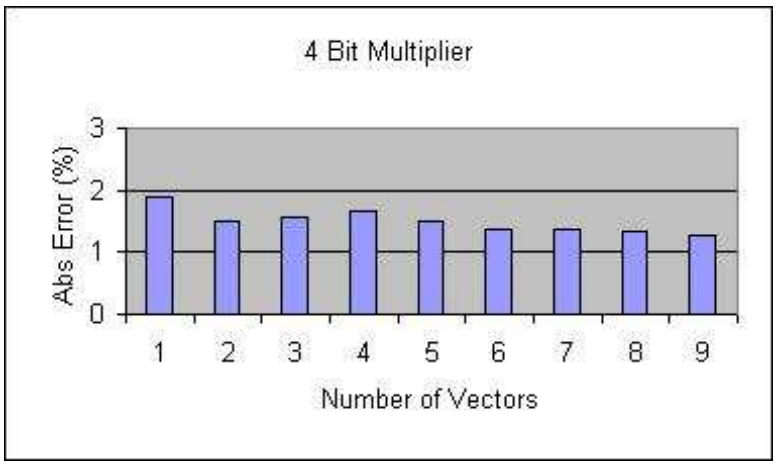


Figure 5.12. Absolute percentage error vs. number of vectors plot for multiplier

simulatable VHDL design is simulated using a VHDL simulator and the leakage power profile for the design is generated. The designs are synthesized using the as soon as possible (ASAP) scheduling algorithm.

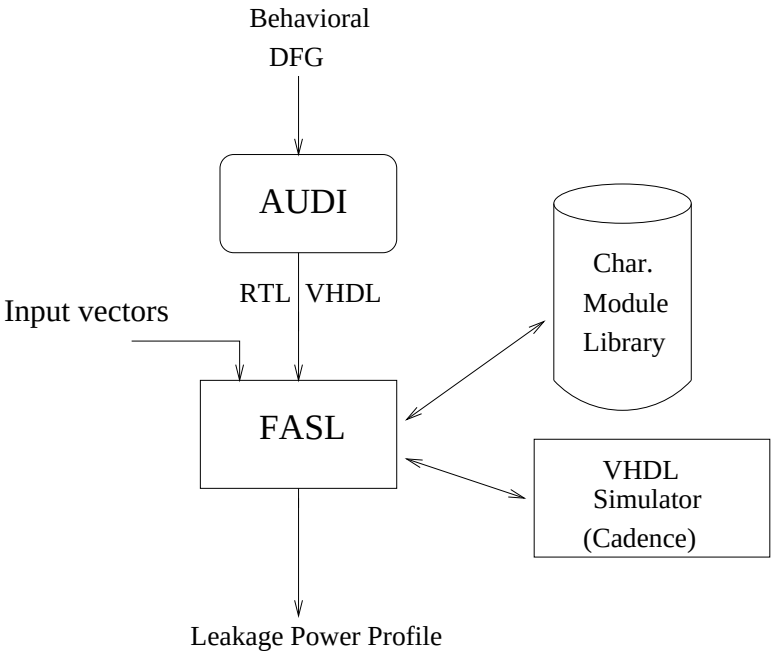


Figure 5.13. Leakage power simulation in FASL

The leakage power profile that is generated by FASL is compared to that generated by using HSPICE. The experimental flow for the latter is summarized in Fig. 5.14. The structural VHDL generated by AUDI is translated into the Structural Description Language (SDL) format. The LAGER IV Silicon Compiler synthesizes a MAGIC layout of the design. Synopsys Nanosim is used to profile the input data streams through the spice netlist and obtain input traces for all instances in the data-path. The leakage power for each of the instances is obtained using HSPICE and the post processor described in Fig. 5.7.

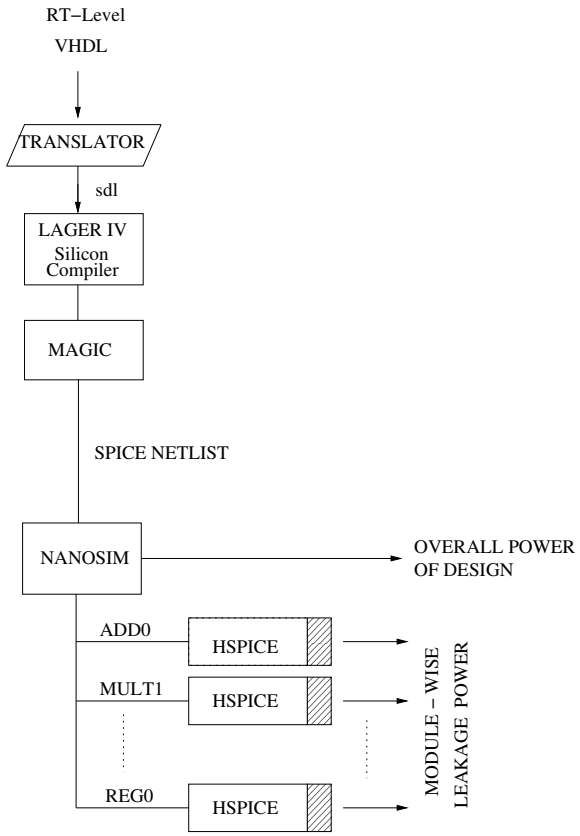


Figure 5.14. Experimental flow to obtain leakage power in data-paths

Table 5.2. Design data of benchmarks

Design	Width	PIs	POs	Operations
IIR	8	10	1	5 Mults, 4 Adds
FIR	8	10	1	5 Mults, 4 Adds
EWF	4	9	7	7 Mults, 26 Adds

Table 5.2 shows the design data for the three benchmarks chosen. Tables 5.3-5.5 present the leakage power profiles obtained using HSPICE and by FASL. Three different random vector sets are used for simulation, with varying lengths. It is observed that the maximum error percentage in the leakage power for the datapath is 3.567%. For most of the results, the error percentage is less than 1%. Table 5.9 show the simulation times. The HSPICE simulations are run on SUN SPARC Ultra 2 Machines with 256M RAM. The simulation times for the FASL, is the user time that the VHDL simulator reports. As seen in Table 5.9, a speedup of 4-5 orders of magnitude is acheived. Tables 5.6-5.8 present the effect of including the transient region in the characterization plots. It is observed that, there is a considerable difference in the leakage power values in the combinational modules such as adder and multipliers.

Table 5.3. IIR : Leakage power profiles for different vector sets

Module	1 Vector			5 Vectors			9 Vectors		
	HSP (uW)	FASL (uW)	% error	HSP (uW)	FASL (uW)	% error	HSP (uW)	FASL (uW)	% error
Adders	0.111	0.101	9.00	0.497	0.454	8.65	0.893	0.812	9.07
Registers	0.557	0.545	2.15	2.558	2.427	5.18	4.586	4.310	6.01
Multipliers	2.408	2.452	-1.82	11.026	11.421	-3.58	19.663	20.427	-3.88
Mux	0.315	0.277	12.06	1.518	1.272	16.2	2.716	2.266	16.56
Datapath	3.391	3.375	0.47	15.600	15.574	0.16	27.858	27.816	0.15
Sim.Time			400ns			1800ns			3200ns

Table 5.4. FIR : Leakage power profiles for different vector sets

Module	1 Vector			5 Vectors			9 Vectors		
	HSP (uW)	FASL (uW)	% error	HSP (uW)	FASL (uW)	% error	HSP (uW)	FASL (uW)	% error
Adders	0.413	0.404	2.17	0.412	0.410	0.48	0.411	0.411	0.0
Registers	4.332	4.372	-0.92	4.474	4.344	2.97	4.486	4.345	3.14
Multipliers	19.284	19.69	-2.1	19.607	20.342	-3.74	19.654	20.434	-3.96
Mux	2.095	1.861	11.16	2.161	1.899	12.12	2.190	1.893	13.56
Datapath	26.123	26.331	0.796	26.654	26.994	1.27	26.740	27.083	1.28
Sim.Time			450ns			2050ns			3650ns

Table 5.5. EWF : Leakage power profiles for different vector sets

Module	1 Vector			5 Vectors			9 Vectors		
	HSP (uW)	FASL (uW)	% error	HSP (uW)	FASL (uW)	% error	HSP (uW)	FASL (uW)	% error
Adders	0.825	0.833	-0.969	0.822	0.845	2.798	0.816	0.849	-4.04
Registers	3.220	3.035	5.74	3.312	2.926	11.65	3.287	2.912	11.40
Multipliers	1.889	1.928	-2.06	1.891	1.919	-1.53	1.885	1.909	-1.27
Mux	8.102	8.005	1.19	8.208	8.034	2.11	8.090	7.968	1.50
Datapath	14.037	13.801	1.68	14.232	13.724	3.56	14.079	13.638	3.132
Sim.time			900ns			4300ns			7700ns

Table 5.6. IIR : Effect of transient characterization on leakage power profile

Module	HSP	With Transient		Without Transient	
		FASL	Error	FASL	Error
	(uW)	(uW)	(%)	(uW)	(%)
Adders	0.893	0.812	9.07	0.779	12.76
Registers	4.586	4.310	6.01	4.549	0.81
Multipliers	19.663	20.427	-3.88	19.938	-1.39
Mux	2.716	2.266	16.56	2.699	0.63
Datapath	27.858	27.816	0.15	27.967	0.39

Table 5.7. FIR : Effect of transient characterization on leakage power profile

Module	HSP	With Transient		Without Transient	
		FASL	Error	FASL	Error
	(uW)	(uW)	(%)	(uW)	(%)
Adders	0.411	0.411	0.0	0.387	5.74
Registers	4.486	4.345	3.14	4.549	1.41
Multipliers	19.654	20.434	-3.96	19.963	1.57
Mux	2.190	1.893	13.56	2.258	3.10
Datapath	26.740	27.083	1.28	27.158	1.54

5.4.2 Random logic circuits

In the case of random logic circuits, SIS is used to synthesize the MCNC descriptions. To simplify the analysis, the circuit is mapped only to the universal gates (NAND and NOR). A translator is written to convert descriptions in bdnet format to structural VHDL.

Table 5.8. EWF : Effect of transient characterization on leakage power profile

Module	HSP	With Transient		Without Transient	
		FASL	Error	FASL	Error
	(uW)	(uW)	(%)	(uW)	(%)
Adders	0.816	0.849	-4.04	0.804	1.49
Registers	3.287	2.912	11.40	3.262	0.76
Multipliers	1.885	1.909	-1.27	1.943	2.98
Mux	8.090	7.968	1.50	9.661	9.40
Datapath	14.079	13.638	3.132	15.671	11.30

Table 5.9. Simulation run times

Module	1 Vector		5 Vectors		9 Vectors	
	HSP	FASL	HSP	FASL	HSP	FASL
IIR	12hr 25min	1.5s	10hr 42min	1.8s	2hr 47min	2.3s
FIR	15hr 39min	1.5s	12hr 36min	1.6s	2hr 34min	1.8s
EWF	7hr 39min	1.4s	3hr 11min	1.5s	36min	1.7s

Simulatable leakage models for NAND and NOR gates have been created as described in Section 5.1.

Results are presented for 20 medium sized MCNC benchmarks. The circuits are simulated with 100 random vectors. The accuracy of the FASL simulator is compared with HSPICE (HSP). The designs are synthesized using LAGER IV Silicon Compiler to layout level. The extracted spice netlists are simulated with 100nm technology models.

Table 5.10 present the leakage power estimates for the MCNC benchmarks and the simulation times. The simulation time is reduced by several orders of magnitude. It is also observed that the maximum error percentage is less than 5%, which is acceptable. Table 5.11 shows the effect of including the leakage power dissipated during the transient period in the characterization graphs. The benchmarks presented were randomly chosen from the MCNC suite of benchmarks. The long simulation times taken by HSPICE, which is used to validate the results, deterred us from picking larger examples.

Table 5.10. Random logic - MCNC benchmarks

Circuit	Simulation Time		Leakage Power (uW)			Size
	FASL	HSP	FASL	HSP	Error	(gates)
C17	2.0s	50.4s	0.5655	0.5635	0.35	6
C432	6.7s	30min40s	2.0197	2.0539	1.66	282
C499	4.9s	1hr37min	4.2603	4.2861	0.60	567
C1355	8.1s	1hr29min	5.1405	5.1388	0.03	552
C1908	8.7s	2hr38min	5.5304	5.5882	1.03	772
cm42a	2.5s	3min5s	0.2234	0.2260	1.15	33
cm82a	1.1s	2min1s	0.2287	0.2257	1.33	28
cm138a	2.7s	1min54s	0.1761	0.1708	3.10	29
cm152a	0.9s	1min34s	0.2127	0.2109	0.85	24
cm162a	0.7s	4min7s	0.4132	0.3948	4.66	54
x1	1.2s	47min	3.2364	3.1743	1.96	440
x2	2.6s	6min52s	0.5497	0.5427	1.29	73
alu2	4.2s	1hr20min	3.3427	3.3319	0.32	462
b9	3.9s	15min50s	1.1091	1.0698	3.67	147
c8	3.9s	12min8s	1.5054	1.4500	3.82	211
cordic	3.2s	13min45s	0.9643	0.9203	4.79	124
count	2.6s	20min10s	1.1961	1.1485	4.14	161
example2	4.9s	32min57s	2.5996	2.4884	4.46	351
frg1	5.6s	19min2s	1.0434	1.0242	1.87	143
pcle	1.5s	6min36s	0.5678	0.5607	1.26	71

5.5 Summary

A Fast and Accurate Simulator for Leakage Power (FASL) is presented, to estimate leakage power in RT-Level VHDL descriptions. Exhaustive simulation of leaf cells shows that the leakage power is temporally dependent on the previous input during the transient time. Exploiting the hierarchical design of the data-path during simulation, results in a speed-up of 4-5 orders of magnitude, and yet achieves accuracy, close to that of HSPICE.

Table 5.11. Random logic - effect of transient characterization

Module	HSP	With Transient		Without Transient	
		FASL	Error	FASL	Error
	(uW)	(uW)	(%)	(uW)	(%)
C17	0.5635	0.5655	0.35	0.0516	8.44
C432	2.0539	2.0197	1.66	1.9019	7.39
C499	4.2861	4.2603	0.60	4.1190	3.89
C1355	5.1388	5.1405	0.03	5.0034	2.63
C1908	5.5882	5.5304	1.03	5.2731	5.63
cm42a	0.2260	0.2234	1.15	0.2139	5.35
cm82a	0.2257	0.2287	1.33	0.2187	3.10
cm138a	0.1708	0.1761	3.10	0.1646	3.62
cm152a	0.2109	0.2127	0.85	0.2105	0.18
cm162a	0.3948	0.4132	4.66	0.4001	1.34
x1	3.1743	3.2364	1.96	3.0631	3.50
x2	0.5427	0.5497	1.29	0.5262	3.04
alu2	3.3319	3.3427	0.32	3.2214	3.31
b9	1.0698	1.1091	3.67	1.0652	0.43
c8	1.4500	1.5054	3.82	1.4333	1.15
cordic	0.9203	0.9643	4.79	0.9608	1.46
count	1.1485	1.1961	4.14	1.1481	0.03
example2	2.4884	2.5996	4.46	2.5179	1.18
frg1	1.0242	1.0434	1.87	0.9957	2.78
pcle	0.5607	0.5678	1.26	0.5587	0.35

CHAPTER 6

TABU SEARCH FOR LOW LEAKAGE POWER DATA-PATHS

In Chapter 4, we proposed a constructive approach for synthesizing low leakage power data-paths. Search based techniques start with an initial solution and search for better solutions by applying a series of moves. A very commonly used technique is simulated annealing, which has been discussed in detail in Section 2.6.1. A relatively new search based technique called the tabu search was also discussed in 2.6.2.

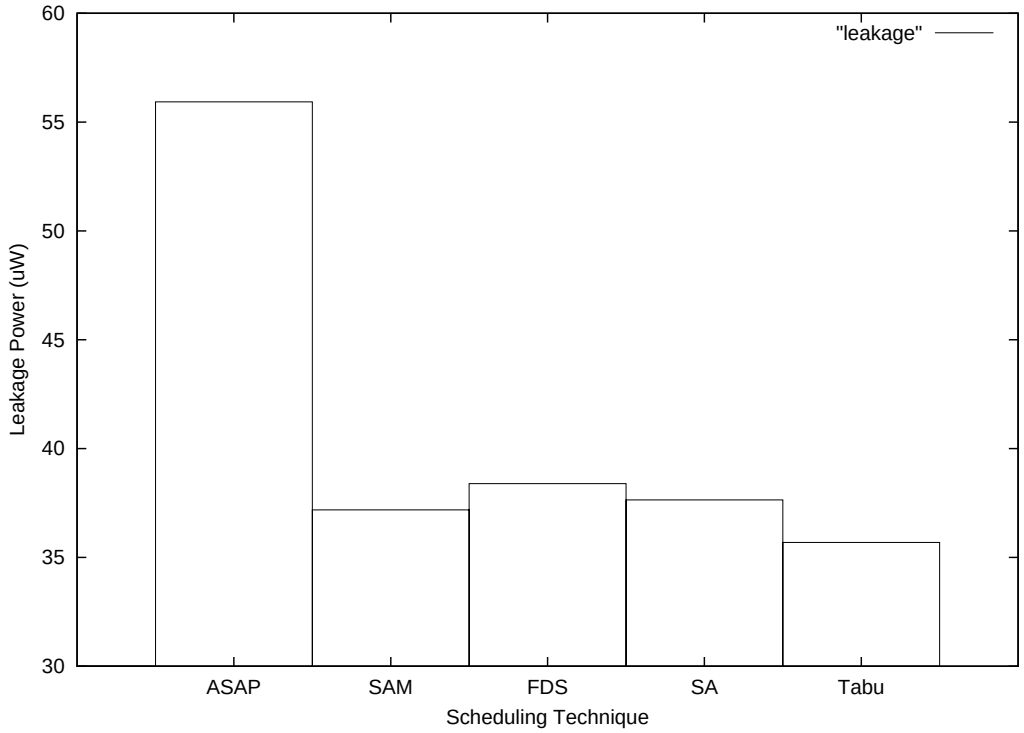


Figure 6.1. Leakage power profile for AR filter

6.1 Motivational example

The leakage power dissipated by datapaths synthesized using different scheduling algorithms for a design, exhibit large variations. Fig. 6.1 shows the leakage power dissipated by designs synthesized using various scheduling techniques for the AR Filter. We observe the significant variations in leakage power dissipated in the different datapaths synthesized. In Fig. 6.1, ASAP refers to as soon as possible scheduling, FDS refers to the Force Directed Scheduling[55], SAM refers to the Simultaneous Scheduling, Allocation and Mapping algorithm [56], SA refers to the Simulated Annealing, and Tabu refers to the search technique described in 6.2.

We present two tabu search approaches for finding low leakage power data-paths. Section 6.2 describes our first approach to search for low leakage power schedules. Section 6.3 describes our second appraoch to simultaneously schedule and bind for low leakage power. Section 6.4 presents the experimental results.

6.2 Approach I: Tabu search for low leakage schedule

In section 6.2.1, we describe the move set that is used to generate neighboring solutions. Neighbor solutions are visited by *moving* mobile operations within their mobility range. Section 6.2.2 describes the cost metric that has been developed to evaluate the leakage power of each solution. Section 6.2.3 describes the intensification and diversification strategies that have been implemented to guide the tabu search.

6.2.1 Move set and attributes

The move set is similar to the TASS [18] move set. Each operation in the DFG has an associated mobility, defined as the difference between the as-late-as-possible (ALAP) time-step and the as-soon-as-possible (ASAP) time-step. A mobile operation can be moved from the current time-step to any possible time-step within its mobility range. Tabu search requires memory to store solutions that have been visited recently. However, it is

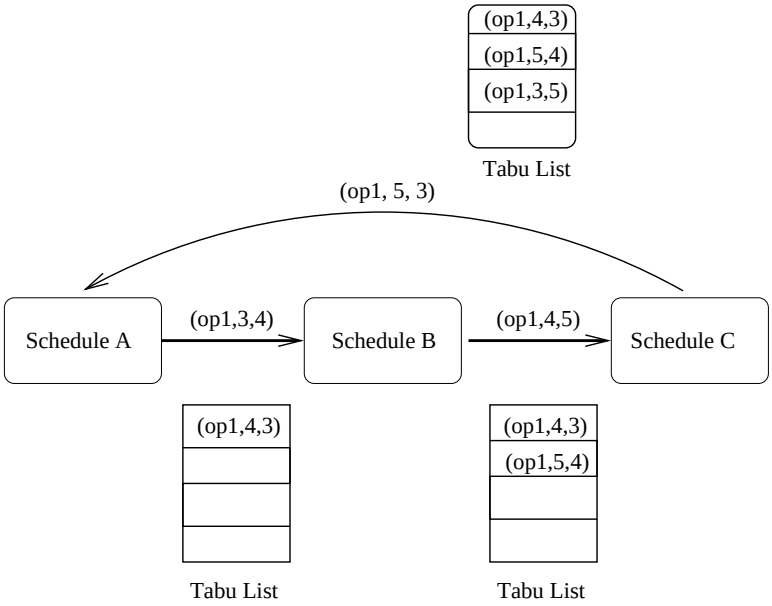


Figure 6.2. Cycling observed while using move attributes used in TASS

prohibitive to store whole solutions. Glover proposes the concept of move-attributes. A move attribute is a property of a move that describes the change that is made on the original solution to reach a possible new solution. The TASS procedure uses the following move attribute to describe a move:

$$(operation,\ move\_from\_cstep,\ move\_to\_cstep)$$

However, we observed that this might lead to cycling in some cases. Fig. 6.2 illustrates one such case. Consider an intermediate schedule A. Assuming, we find a non-tabu move (op1, 3, 4), which moves operation 1 from cstep 3 to cstep 4. At this juncture, the tabu list is updated with (op1, 4, 3), so that the move $4 \Rightarrow 3$ is made tabu. Similarly, on making a successive move (op1, 4, 5) on S2, we update tabu list by adding (op1, 5, 4). At this point, if the algorithm finds move (op1, 5, 3) to be lucrative, it examines the tabu list and finds that it is not tabu. However, we observe that we are cycling back to solution A that has been visited. To avoid this cycling, we modify the move attributes to include two

separate tabu lists $tabu\_from$ and $tabu\_to$. The checking for tabu now involves checking if the move_from_cstep is in the $tabu\_from$ list, **or** move_to_cstep is in the $tabu\_to$ list.

6.2.2 Leakage cost metric

At each iteration of the tabu search, the neighboring solutions for a schedule are examined. Each neighbor is essentially a schedule. For each of these schedules, a minimum number of modules are allocated and mapped using a clique partitioning algorithm [57]. Similarly, the edges interconnecting the operations are mapped to a minimum number of registers. With this information, we also obtain the number of multiplexers needed for sharing FU instances and registers. We formulate a leakage cost metric as a function of the number of these modules needed (Eqn. 1).

$$Leakage\_cost = k_{reg} * (\#registers) + k_{add} * (\#adders)$$
$$+ k_{mult} * (\#multipliers) + k_{mux} * (\#multiplexers) \quad (6.1)$$

where k_{reg} and k_{fu} are functions of the widths of the registers and FUs respectively. k_{mux} is a function of the width of the multiplexor inputs and the number of select lines.

The constants k_{reg}, k_{fu}, and k_{mux} are obtained by characterizing modules of different widths with long sequences. Fast and accurate characterization is done using the Fast Architectural Simulator for Leakage (FASL) [67], described in Chapter 5. The modules are characterized with respect to bit widths to facilitate solution space exploration of a wide variety of designs. Additionally, k_{mux} is dependent on the number of select lines. We observed that this cost function changes monotonically with the leakage power and is also reasonably accurate. Fig. 6.3 shows the cost metric plotted against the actual leakage for a set of 40 iterations, obtained during solution space search for EWF. Ideally, the points should all fall on a single line with unity slope.

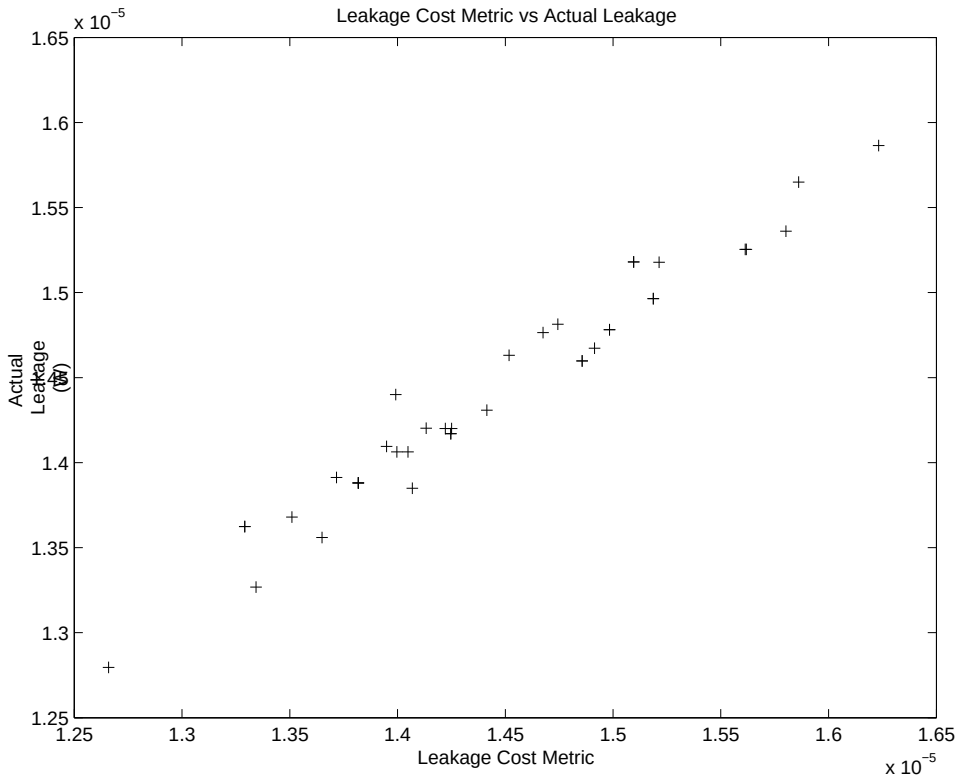


Figure 6.3. Accuracy of cost metric

6.2.3 Intensification and diversification

In this work, we use the *elite candidate* approach for intensification, where the run is restarted with the best solution as the initial solution [17]. During each run, we keep track of the number of times an operation is moved using a move table. In a diversification run, we pick the operation that has been least moved and make a random move on the operation. The schedule hence obtained is used as the initial solution in the next run. We implemented an oscillating intensification and diversification framework (first proposed by Glover). In this approach, intensification and diversification are applied in alternate runs.

6.3 Approach II: Low leakage power scheduling and mapping

Unifying the three stages of behavioral synthesis viz., scheduling, allocation, and mapping, produces more optimal results compared to the three stages performed sequentially [56].

The proposed approach performs this unified search for a low leakage power datapath using the tabu search technique. The data path is modeled as a space-time matrix [54] (See Fig. 6.4). There exists a row for each time step. There exists a column for each instance that needs to be bound. An initial loose upper bound is assumed to be the number of instances required if sharing was not implemented. Moves are made on the operations in these ma-

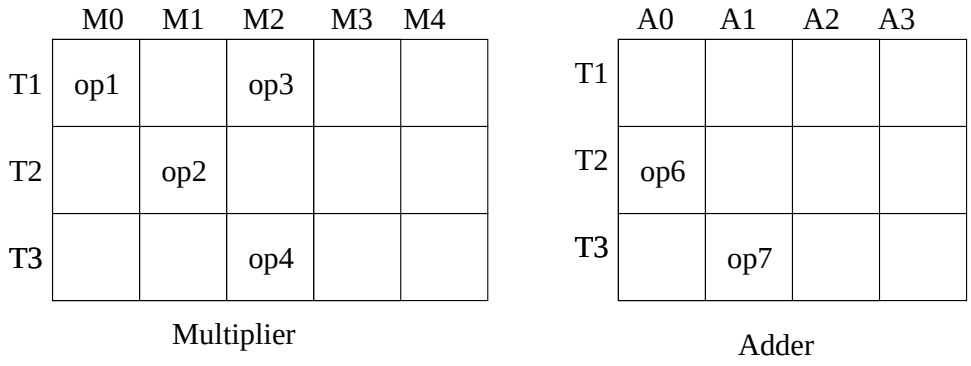


Figure 6.4. Space-time matrices

trices to move between solutions. We have used the same move set from [54]. The different moves are: (1) displacing an operation from one time step to another; (2) interchanging two operations within a time step; (3) interchanging the inputs of symmetric operations; (4) displacing an operation within a time step. The first move type work towards obtaining a compact schedule. The second and third move types improves sharing and tries to reduce the number of interconnect units required. The fourth move type attempts to reduce the number of instances required in that time step. The high temperature moves are not distinguished from the low temperature moves in the tabu framework. The move types are chosen at random. Tabu lists are maintained for each move type. This reduces cycling between solutions. The cost metric in this approach is a combination of leakage and the latency of the schedule. The leakage component is calculated as described in Section 6.2.2. The latency cost is the number of csteps in the schedule. This drives the search to search for low latency, low leakage power schedules. We introduce intensification and diversification runs similar to those described in the approach I. In addition to these runs, we propose a

compaction run. The algorithm performs a compaction run once in 3 runs. The objective of the compaction run is to reduce the number of instances required. The run comprises of iteratively running the move type (4) a fixed number of times. The application of intensification, diversification, and compaction strategies leads to the discovery of a large number of unexplored solutions with low area and leakage power.

6.4 Experimental results

Results are presented for four datapath intensive benchmarks viz., Elliptic Wave Filter (EWF), IIR Filter, Avenhaus Filter (Cascaded Form), and AR Filter. For the first approach, results are collected for four different slacks available for each example, to artificially create a large solution space. Increasing the slack for each operation increases the mobility of each operation, thereby increasing the solution space significantly. In the second approach, the search finds a schedule with minimum latency.

6.4.1 Approach I

The initial solution is an ASAP schedule. In each iteration, all neighbor solutions (schedules) are examined. Each schedule is then allocated and mapped to the minimum number of functional unit (FU) instances and register instances using a clique partitioning algorithm [57]. The minimum number of interconnect units is then estimated. The leakage cost is then estimated. In this manner, a neighborhood leakage table is created. The best move is then checked for tabu status. If the move is not tabu, or if it satisfies the aspiration criteria, the move is made and the tabu list data structures are updated. If the move is tabu and the AL is not reached, the next available non-tabu is move is searched for in the neighborhood leakage table. The process is repeated for a certain number of iterations or until a stopping criterion is reached. The approach is diagrammatically shown in Fig. 6.5. The tabu search has been coded in C, and has been integrated into a synthesis system being developed by the authors. Fig. 6.6 shows the progress of the tabu search for the EWF filter.

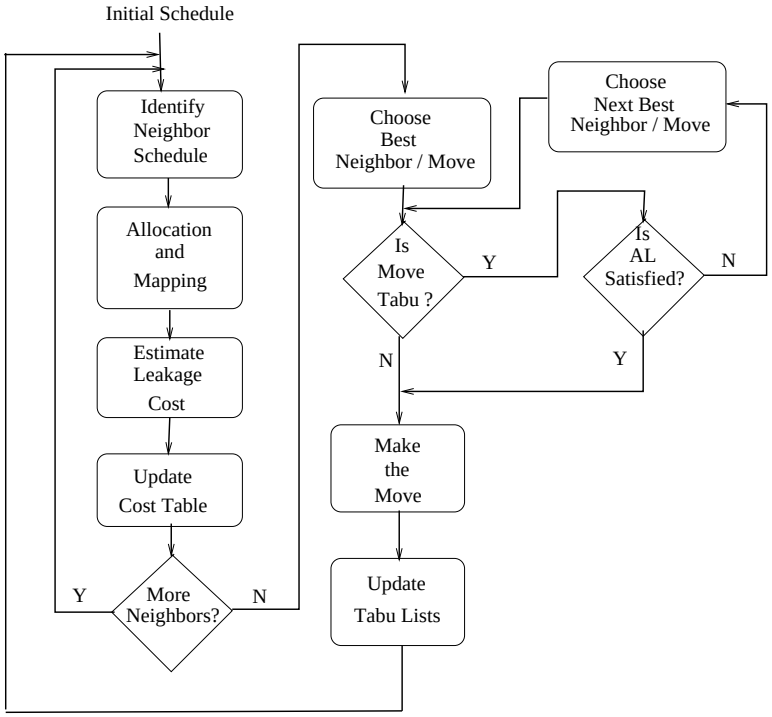


Figure 6.5. Experimental approach I

Tables 6.4.1-6.4.1 detail the leakage power for various slacks provided. Column 3 gives the leakage power dissipated by the design generated using FDS. Columns 4 and 5 list the leakage power dissipated by the design discovered by SA and improvement (in %) over FDS respectively. Columns 6 and 7 list the leakage power dissipated by the design discovered by Tabu and improvement (in %) over FDS. The leakage power was measured using the FASL simulator [67] for 1000 random vectors. The leafcells in the hierarchical description were initially characterized using HSPICE for 100nm technology [63]. The FASL tool measures leakage power from hierarchically described RT-Level VHDL designs, and has been shown to measure leakage power with accuracy of less than 5% error with respect to HSPICE. The tool is also significantly faster compared to HSPICE. We observe that in the majority of the designs, the tabu search consistently discovers datapaths which dissipate lower leakage power than SA and FDS. In general, we observe that larger designs, with more mobility of operations are more effectively searched using the tabu search technique compared to SA.

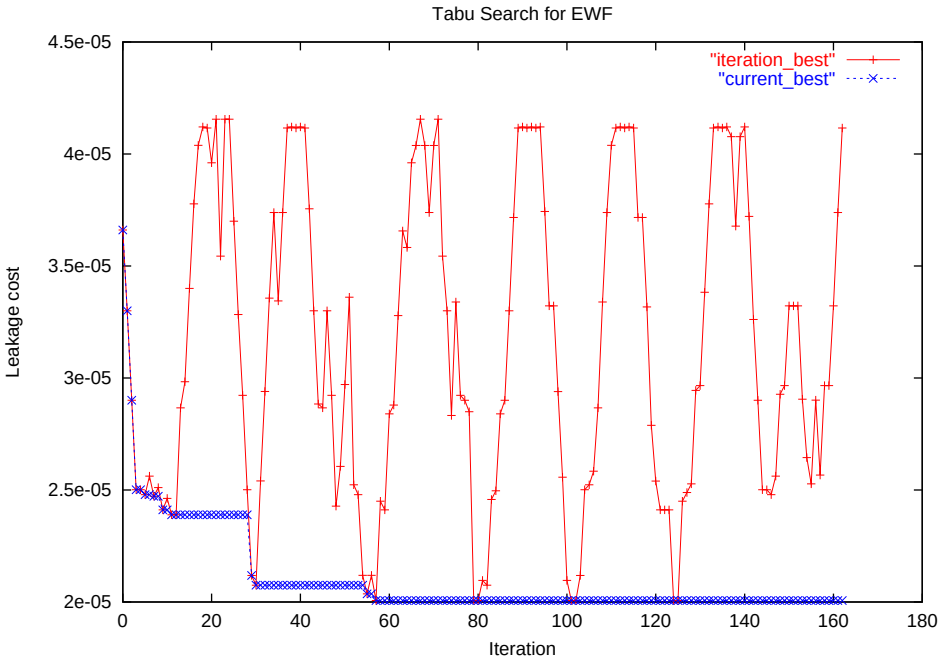


Figure 6.6. Tabu search for low leakage data-path (EWF)

Table 6.4.1 compare the run times for the EWF example. EWF is the most complicated example in the set of designs presented, because of the large number of inherently mobile operations, creating a large solution space. The times shown for SA are approximately the same because the SA algorithm timed out, because of cycling and failed to produce a better solution than tabu. We observe that for large solution spaces (slack = 4), tabu takes as much as 3 times more time than SA. This is due to the analysis of a large number of neighbor solutions before making a decision. We also notice that as the solution space becomes smaller, the gap between tabu and SA narrows and eventually, tabu becomes faster than SA. The remaining three examples show similar results.

Table 6.1. Leakage power values for different slack - EWF; number of operations = 33

Slack	FDS(uW)	SA(uW)	Tabu(uW)	$\frac{FDS-Tabu}{FDS}(\%)$	$\frac{SA-Tabu}{SA}(\%)$
0	31.5047	28.6895	26.4118	16.16	7.93
1	24.1219	27.4365	19.8357	17.77	27.70
2	20.2149	25.7863	19.2727	4.66	25.25
4	19.9109	26.9534	19.4182	2.47	27.95

Table 6.2. Leakage power values for different slack - avenhaus filter; number of operations=15

Slack	FDS(uW)	SA(uW)	Tabu(uW)	$\frac{FDS-Tabu}{FDS}$(%)	$\frac{SA-Tabu}{SA}$(%)
0	20.8236	20.8147	19.5688	6.07	5.98
1	21.2615	19.7899	18.9522	10.86	4.23
2	19.8522	19.8142	19.1756	3.41	3.22
4	20.1395	17.4757	14.4978	28.01	17.04

Table 6.3. Leakage power values for different slack - IIR filter; number of operations=9

Slack	FDS(uW)	SA(uW)	Tabu(uW)	$\frac{FDS-Tabu}{FDS}$(%)	$\frac{SA-Tabu}{SA}$(%)
0	20.5277	20.5276	21.3333	-3.92	-3.92
1	17.2163	17.2216	16.3088	5.27	5.30
2	17.2358	16.9462	15.2640	11.44	9.99
4	17.2669	16.9758	12.5631	27.24	25.99

Table 6.4. Leakage power values for different slack - AR filter; number of operations=28

Slack	FDS(uW)	SA(uW)	Tabu(uW)	$\frac{FDS-Tabu}{FDS}$(%)	$\frac{SA-Tabu}{SA}$(%)
0	38.3922	37.6472	35.6908	7.04	5.19
1	39.5671	36.6882	30.0559	24.04	18.07
2	30.7558	36.8426	29.9605	2.58	18.67
4	42.0932	37.9624	25.2536	42.00	33.47

Table 6.5. Run times for EWF for various slacks

Slack	Tabu	SA
4	783s	261s
2	392s	263s
1	138s	251s
0	24s	188s

This work has concentrated on searching for low leakage power datapaths. Once a low leakage power datapath has been identified, well established techniques for dynamic power reduction may be applied at RT-Level and lower levels of design abstraction. Most designs that tabu discovered were incidentally smaller in area than FDS and SA. Table 6.4.1 tabulates the area of the datapaths generated by FDS, SA, and tabu for zero slack.

The area has been calculated by summing up the bounding box areas of the individual modules that make up the datapath.

Table 6.6. Area of datapaths synthesized. All units in λ^2; Slack = 0

Design	FDS	SA	Tabu	$\frac{FDS-Tabu}{FDS}(\%)$	$\frac{SA-Tabu}{SA}(\%)$
EWF	10870448	11931152	9257944	14.83	22.40
AR	13697448	13249992	12927224	5.62	2.43
IIR	7518760	8784976	7788680	-3.58	11.34
Aven	7438960	8773128	7076656	4.87	19.33

6.4.2 Approach II

In this approach, the initial solution is first mapped to a space-time matrix. During each iteration, moves are made on this space-time matrix. The minimum number of registers are found using a clique partitioning algorithm, and the number of multiplexors are determined for the sharing required. The cost metric can then be determined as described in Section 4. The experimental approach is shown in Fig. 6.7. Results are shown for the same four DSP benchmarks. For fairness of comparison, we chose to compare against the simultaneous scheduling and mapping (SAM) approach, proposed by Cloutier and Thomas [56]. This is a force directed approach, which simultaneously schedules and maps operations in a data flow graph. We also compare against the simulated annealing approach proposed by Devadas and Newton [54].

Table 6.7. Approach II: Leakage power values for various data-paths

Design	SAM	SA	Tabu	Improv. over SAM(%)	Improv. over SA(%)
IIR	23.82	24.88	20.27	14.90	18.52
AR	37.18	55.43	36.27	2.44	34.56
EWF	34.06	33.04	28.25	17.05	14.49
Avenh	24.01	26.25	21.29	11.32	18.89

Table 6.4.2 tabulates the leakage power of the datapath synthesized by SAM and the solutions that were found by SA and Tabu. As in the case of Approach I, we observed a

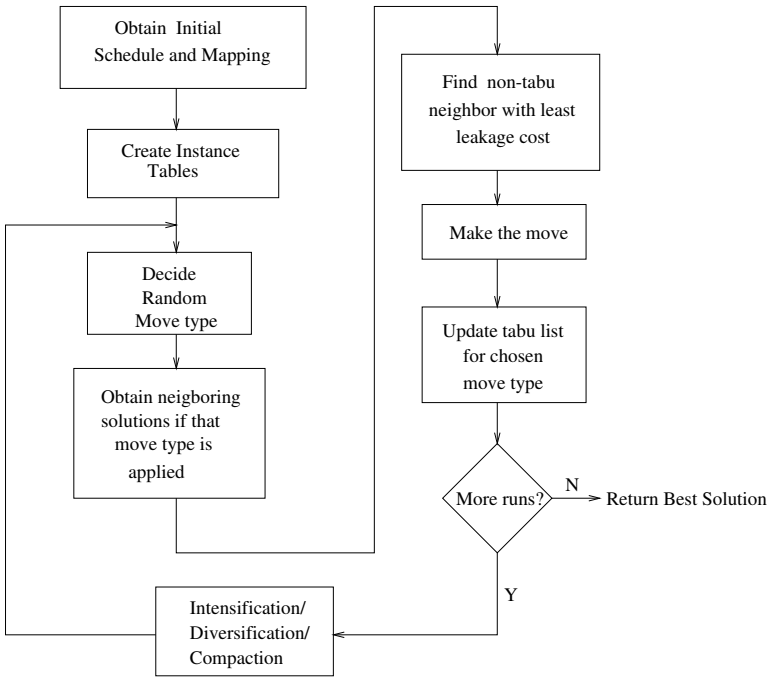


Figure 6.7. Experimental approach II

Table 6.8. Approach II: Area of datapaths synthesized. All units in λ^2

Design	SAM	SA	Tabu	Improv. over SAM(%)	Improv. over SA(%)
IIR	8784976	10274968	7369580	16.11	28.27
AR	13249992	20034888	12859326	2.94	35.81
EWF	11931152	11588416	9482226	20.52	18.17
Avenh	8773128	14641176	8325098	5.10	43.14

reduction in area in the solution obtained by the tabu search. In this approach, due to the compaction runs, we observe a significant reduction in area. Table 6.4.2 gives the area reduction.

6.5 Summary

In this chapter, two search based approaches based on tabu search were presented. The first approach searches for schedules which dissipate low leakage power. The second approach takes a unified approach to search simultaneously for low leakage schedules and

binding alternatives. The latter approach models the search as a two dimensional placement problem, placing each operation in a DFG onto a time-space matrix. It is shown that the unified approach finds better solutions than the first approach. Results are also shown comparing the tabu search to another commonly used search heuristic, simulated annealing. Due to reduced cycling, tabu finds better solutions faster than SA.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

The fraction of leakage power to the total power dissipated in CMOS circuits is increasing as we move into newer technologies. Several approaches to estimate and minimize leakage power at various levels of design hierarchy have been proposed in the near past.

With the growing importance of leakage power in design automation, a fast and accurate leakage estimation tool becomes critical. An estimate of leakage power during early stages of the design cycle, such as RT-Level, would aid designers significantly. We present an RT-Level leakage power simulator for designs described hierarchically in VHDL. An accurate leakage model has been proposed for leafcells. It was established that for designs with high performance (and hence small clock periods), the assumption that leakage power is statically dependent on the current inputs to the circuit, is not valid. The leakage power model proposed incorporates the transient characteristics as well, to enable accurate measurement of leakage power. We present results which clarify the importance of the transient characterization. Results were presented for three DSP data-paths and 20 MCNC random logic benchmarks. The average error percentage was 1.38% and 2.11% in the data-path examples and the random logic benchmarks respectively. The estimation time compared to HSPICE was lower by several orders of magnitude.

Several opportunities exist for minimizing leakage power at high levels of abstraction. This dissertation presents leakage power reduction during the process of behavioral synthesis - the process of transforming a behavioral description to an RT-Level description. There are several techniques that have been presented in literature to reduce leakage power dissipation at lower levels of design hierarchy. There has been minimal work done in minimizing leakage power during the early stages of design cycle. We present a comprehensive

framework for synthesizing low leakage data-paths using a multi-threshold CMOS (MTCMOS) component library. Components in this library are capable of being switched off. The components have been characterized for leakage power and delay with respect to width of the sleep transistor.

We propose a power management methodology for the data-paths that are synthesized with MTCMOS components. Allocation and binding algorithms are proposed to maximize the leakage power reduction. MTCMOS components impose an area and performance penalty. To overcome the area overhead imposed by MTCMOS components, we propose a selective binding methodology based on the 0-1 Knapsack problem. The approach synthesizes a data-path within area constraints specified by the user. A performance recovery technique was also described to alleviate the performance degradation due to the introduction of MTCMOS components in the data-path. The technique is a combination of multi-cycling and introduction of slack. Leakage power reduction of upto 28.52% and upto 42.63% was observed in 180nm and 100nm technologies.

Finally, we proposed two search based approaches for synthesizing low leakage data-paths using tabu search. The first approach explored the schedule space for a low leakage scheduling option. The second approach searched simultaneously for a low leakage schedule and binding. The latter unified approach was shown to discover better data-paths which dissipated lower leakage power. The approaches proposed identified data-paths which dissipate upto 40% lower leakage power than traditional heuristics. It was also shown that the tabu search approaches explores the search space more efficiently than a competing technique for design space exploration - simulated annealing. Tabu search found better solutions at a moderate increase in computation time.

Future work in the synthesis of low leakage power data-paths may be carried out in the following directions:

- Develop power management algorithms for control flow intensive designs
- Extend the allocation and binding algorithms for control flow intensive designs

- Create a VHDL leakage model which does not require characterization of leafcells, but would calculate leakage based on technology parameters provided by user
- Extend the second tabu search approach to selectively bind to MTCMOS components

REFERENCES

[1] S. Borkar. " Low power design challenges for the decade ". In *Proceedings of the Asia and South Pacific Design Automation Conference*, pages 293–296, 2001.

[2] J.T. Kao and A.P. Chandrakasan. "Dual-Threshold Voltage Techniques for Low-Power Digital Circuits". *IJSSC*, 35(7):1009–1018, July 2000.

[3] K. Roy. "Leakage Power Reduction in Low-Voltage CMOS Design". In *Proceedings of the IEEE International Conference on Circuits and Systems*, pages 167–173, 1998.

[4] –. *"The International Technology Roadmap for Semiconductors"*. SIA (Semiconductor Industry Association), 2001.

[5] M. Pedram. "Power Minimization in IC Design: Principles and Applications". *ACM Transactions on Design Automation of Electronic Systems*, Vol. 1, No. 1, pp:3–56, January 1996.

[6] A. Chandrakasan et. al.,. "Optimizing Power Using Transformations". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, pages 12–31, January 1995.

[7] A. Raghunathan and N. K. Jha . "SCALP : An Iterative-Improvement-Based Low-Power Data Path Synthesis System". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(11):1260–1277, November 1997.

[8] M. Johnson and K. Roy. "Optimal Selection of Supply Voltages and Level Conversions During Data-Path Scheduling Under Resource Constraints". In *Proceedings of International Conference on Computer Design*, pages 72–77, 1996.

[9] J. Monteiro, S. Devadas, P. Ashar, and A. Mauskar. "Scheduling Techniques to Enable Power Management". In *Proceedings of 33rd Design Automation Conference*, pages 349–352, June 1996.

[10] J-M. Chang and M. Pedram. "Low Power Register Allocation and Binding". In *Proceedings of the 32nd Design Automation Conference*, pages 29–35, 1995.

[11] S. Katkoori and R. Vemuri. "Scheduling for Low Power under Resource and Latency Constraints ". In *ISCAS*, pages 53–56, 2000.

[12] A.Raghunathan, N. K. Jha, S. Dey . *"High Level Power Analysis and Optimization "*. Kluwer Academic Publishers, 1998.

[13] S. P. Mohanty, N. Ranganathan, and S. K. Chappidi. Simultaneous peak and average power minimization during datapath scheduling for dsp processors. In *Proceedings of the 13th ACM Great Lakes Symposium on VLSI*, pages 215–220, 2003.

[14] V. Krishna, N. Ranganathan, N. Vijaykrishnan. "Energy efficient datapath synthesis using dynamic frequency clocking and multiple voltages ". In *Proceedings of the Twelfth International Conference on VLSI Design*, pages 440–445, 1999.

[15] S. M. Sait and H. Youssef. *"Iterative Computer Algorithms with Applications in Engineering"*. IEEE Computer Society Press, 1999.

[16] C. R. Reeves, editor. *"Modern Heuristic Techniques for Combinatorial Optimization Problems"*. McGraw Hill, Europe, 1995.

[17] F. Glover and M. Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England, 1993. Blackwell Scientific Publishing.

[18] S. Amellal and B. Kaminska. "Functional synthesis of digital systems with TASS". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13, 1994.

[19] L.Wei, K.Roy, and V.K. De. "Low Voltage Low Power CMOS Design Techniques for Deep Submicron ICs ". In *Proceedings of the Thirteenth International Conference on VLSI Design, 2000*, pages 24–29, 2000.

[20] J. Halter and F. Najm. "A Gate-level Leakage Power Reduction Method for Ultra-low-power CMOS Circuits". In *Proceedings of the CICC*, pages 475–478, 1997.

[21] Z. Chen, L. Wei, M. Johnson, and K. Roy. "Estimation of Standby Leakage Power in CMOS Circuits Considering Accurate Modeling of Transistor Stacks". In *Proceedings of the ISLPED*, pages 239–244, 1998.

[22] Liqiong Wei, Zhanping Chen, Kaushik Roy, Mark C.Johnson, Yibin Ye, and Vivek De. "Design and Optimization of Dual-Threshold Circuits for Low-voltage Low-Power Applications". *ITVLSI*, 7(1):16–24, March 1999.

[23] Qi Wang and S.B.K.Vrudhula. "Algorithms for Minimizing Standby Power in Deep Submicrometer, Dual Vt CMOS Circuits ". *ITCAD*, 21(3):306–318, March 2002.

[24] Qi Wang and S.B.K.Vrudhula. "An investigation of power delay trade-offs for dual Vt CMOS circuits". In *ICCAD*, pages 556–562, 1997.

[25] S. Sirichotiyakul, T. Edwards, C. Oh, R. Panda, D. Blaauw. "Duet: An Accurate Leakage Estimation and Optimization Tool for Dual-V_t Circuits ". *ITVLSI*, 10:79–90, April 2002.

[26] S. Mutoh. "1V Power Supply HighSpeed Digital Circuit Technology with Multithreshold Voltage CMOS". *IJSSC*, 30(8):847–853, August 1995.

[27] S. Mutoh, S. Shigematsu, and Y. Matsuya. "A 1 V multi-threshold voltage CMOS DSP with an efficient power management technique for mobile phone applications". In *Proceedings of the IEEE ISSC Conference*, pages 168–169, Feb 1996.

[28] J.Kao, A. Chandrakasan, and D. Antoniadis. "Transistor sizing issues and Tool for Multi-threshold CMOS technology ". In *Proceedings of the DAC*, pages 409–414, 1997.

[29] A. Chandrakasan, I. Yang, C. Vieri, and D. Antoniadis. "Design considerations and tools for low-voltage digital system design". In *Proceedings of the DAC*, pages 113–116, 1996.

[30] N. Hanchate and N. Ranganathan. "A New Technique for Leakage Reduction in CMOS Circuits Using Self-Controlled Transistor Stacks". In *Proceedings of the Seventeenth International Conference on VLSI Design*, 2004.

[31] K.S. Khouri and N.K. Jha. "Leakage Power Analysis and Reduction during Behavioral Synthesis". In *Proceedings of the ICCD*, pages 561–564, 2000.

[32] M. Powell, S. Yang, B. Falsafi, K. Roy, and T. Vijaykumar. "Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories". In *Proceedings of ISLPED*, pages 90–95, 2000.

[33] M. Powell, Se-Hyun Yang, B. Falsafi, K. Roy, and N. Vijaykumar. " Reducing leakage in a high-performance deep-submicron instruction cache ". *ITVLSI*, 9(1):77–89, February 2001.

[34] A.Agarwal, H.Li, and K.Roy. "DRG-Cache: A Data Retention Gated-Ground Cache for Low Power". In *Proceedings of the DAC*, pages 473–478, 2002.

[35] S. Yang, M. D. Powell, B. Falsafi, K. Roy, and T. N. Vijaykumar. " An integrated circuit/architecture approach to reducing leakage in deep-submicron high-performance I-caches ". In *Proceedings of The Seventh International Symposium on High-Performance Computer Architecture (HPCA)*, pages 147–157, 2001.

[36] M. C. Johnson, D. D. Somasekhar, Lih-Yih Chiou, and K. Roy. " Leakage control with efficient use of transistor stacks in single threshold CMOS ". *ITVLSI*, 10(1):1–5, February 2002.

[37] S.Tadas and C.Chakrabarti. "Architectural Approaches to Reduce Leakage Energy in Caches". In *Proceedings of ISCAS*, pages 481–484, 2002.

[38] W. Zhang, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, D. Duarte, and Y. Tsai. "Exploiting VLIW Schedule Slacks for Dynamic and Leakage Energy Reduction ". In *Proceedings of 34th International Symposium on Microarchitecture*, pages 102–113, 2001.

[39] M. Anis, S. Areibi, M. Mahmoud, and M. Elmasry. " Dynamic and leakage power reduction in MTCMOS circuits using an automated efficient gate clustering technique ". In *Proceedings of 39th DAC*, pages 480–485, June 2002.

[40] D.Duarte, Y.Tsai, N.Vijaykrishnan, and M.J.Irwin. "Evaluating Run-time techniques for Leakage Power Reduction". In *Proceedings of the 15th International Conference on VLSI Design*, pages 31–38, 2002.

[41] Y.Ye, S. Borkar, and V.De. " A new technique for standby leakage reduction in high-performance circuits ". In *Digest of Technical Papers, Symposium on VLSI circuits*, pages 40–41, 1998.

[42] L. Li, I. Kadayif, Y.-F Tsai, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and A. Sivasubramaniam. " Leakage Energy Management in Cache Hierarchies". In *Proceedings of International Conference on Parallel Architectures and Compilation Techniques*, pages 131–140, 2002.

[43] J. A. Butts and G. S. Sohi . "A Static Power Model for Architects ". In *Proceedings of the 33rd IEEE/ACM Annual Symposium on Microarchitecture (MICRO33)*, pages 191–2001, 2000.

[44] R. Kumar and C. P. Ravikumar . "Leakage Power Estimation for Deep Sub-micron Circuits in an ASIC Design Environment ". In *Proceedings of the Fifteenth International Conference on VLSI Design, 2002*, pages 45–50, 2002.

[45] D. Somasekhar M. C. Johnson and K. Roy. "Models and Algorithms for Bounds on Leakage in CMOS Circuits ". *ITCAD*, 18(6):714–725, June 1999.

[46] A. Raghunathan and N. K. Jha. "Behavioral Synthesis for Low Power". In *Proceedings of the ICCD*, pages 318–322, 1994.

[47] M. Potkonjak and M. B. Srivastava. "Behavioral synthesis of high performance, low cost, and low power application specific processors for linear computations".

[48] R. Vemuri S. Katkoori, N. Kumar. "Profile Driven Synthesis System". In *Proceedings of the ICCD*, pages 446–453, 1995.

[49] Wen-Tsong Shiue and C. Chakrabarti. "ILP-based Scheme for Low Power Scheduling and Resource Binding. In *Proceedings of ISCAS*, pages 279–282, 2000.

[50] S. P. Mohanty and N. Ranganathan. "A framework for energy and transient power reduction during behavioral synthesis". In *Proceedings of the 16th International Conference on VLSI Design*, pages 539–545, 2003.

[51] S. Katkoori and R. Vemuri. "A Power Simulator for VHDL Structural Descriptions ". In *Proceedings of the VIUF Fall Conference*, pages 4.17–4.25, 1995.

[52] S.Ravi, A.Raghunathan, and S. Chakradhar. "Efficient RTL Power Estimation for Large Designs ". In *Proceedings of the Sixteenth International Conference on VLSI Design*, 2003.

[53] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[54] S. Devadas and A.R. Newton. "Algorithms for Hardware Allocation in Datapath Synthesis". *ITVLSI*, 8:768–781, June 1989.

[55] P.G. Paulin and J.P. Knight. "Force-Directed Scheduling for the Behavioral Synthesis of ASICs". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 8(6):661–679, June 1989.

[56] R. J. Cloutier and D. E. Thomas. The combination of scheduling, allocation, and mapping in a single algorithm. In *Proceedings of the 27th DAC*, pages 71–76, 1990.

[57] C. Tseng and D. Siewiorek. "FACET: A Procedure for the Automated Synthesis of Digital Systems". In *Proceedings of 20th DAC*, pages 566–572, June 1983.

[58] University Of California, Berkeley. *LagerIV Release 4.0*, 1991.

[59] S. Katkoori. *Behavioral Profiling Based High Level Power Estimation Methodologies for VLSI ASIC and FPGA Synthesis*. PhD thesis, University of Cincinnati, 1996.

[60] A. Salz and M. Horowitz. "IRSIM: an incremental MOS switch-level simulator". In *Proc. Design Automation Conf.,*, pages 173–178, June 1989.

[61] D. Pisinger. A minimal algorithm for the 0-1 knapsack problem. *Journal on Operations Research*, 45:758–767, 1997.

[62] http://www.mosis.org.

[63] http://www-device.eecs.berkeley.edu/ ptm/.

[64] A.Cataldo. "Intel to employ sleep transistors to tame leakage current". *EE-Times*, 2002.

[65] Synopsys. *Synopsys Nanosim User Guide*, tld 2001.06 edition, June 2001.

[66] C. Gopalakrishnan and S. Katkoori. "Resource Allocation and Binding Approach for Low Leakage Power ". In *Proceedings of the Sixteenth International Conference on VLSI Design.*

[67] C. Gopalakrishnan and S. Katkoori. "An Architectural Leakage Power Simulator for VHDL Structural Datapaths ". In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pages 211–212, Feb 2003.

ABOUT THE AUTHOR

Chandramouli Gopalakrishnan received his Bachelors Degree in Computer Science from the University of Madras (India) in 1998. As part of his senior project, he had developed a reconfigurable DSP processor based on systolic elements. He received his Masters degree in Computer Science from the University of South Florida, at Tampa, in 2000. The thesis formulated a novel scheme of power optimization in digital circuits using input transformations.

While in the Ph.D. program at the University of South Florida, he developed allocation and binding algorithms for reducing leakage power. This work has been one of the first in dealing with leakage power at high levels of abstraction. Also, he had developed a fast architectural simulator for leakage power. He has published several conference papers related to this area. His research interests are behavioral synthesis and power optimization.