PERFORMANCE IMPROVEMENT OF VLSI CIRCUITS WITH CLOCK SCHEDULING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KEREM KAPUCU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2009

Approval of the thesis:

PERFORMANCE IMPROVEMENT OF VLSI CIRCUITS WITH CLOCK SCHEDULING

submitted by **KEREM KAPUCU** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. Murat Aşkar _____
Supervisor, **Electrical and Electronics Engineering Department**

Examining Committee Members:

Prof. Dr. Hasan Güran
Electrical and Electronics Engineering, METU _____

Prof. Dr. Murat Aşkar
Electrical and Electronics Engineering, METU _____

Assist. Prof. Dr. Haluk Külah
Electrical and Electronics Engineering, METU _____

Assist. Prof. Dr. Şenan Ece Güran Schmidt
Electrical and Electronics Engineering, METU _____

Dr. Neslin İsmailoğlu
TÜBİTAK UZAY _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: KEREM KAPUCU

Signature :

ABSTRACT

PERFORMANCE IMPROVEMENT OF VLSI CIRCUITS WITH CLOCK SCHEDULING

Kapucu, Kerem
M.S., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. Murat Aşkar

December 2009, 87 pages

Clock scheduling is studied to improve the performance of synchronous sequential circuits. The performance improvement covers the optimization of the clock frequency and the peak power consumption, separately. For clock period minimization, cycle stealing method is utilized, in which the redundant cycle time of fast combinational logic is transferred to slower logic by proper clock skew adjustment of registers. The clock scheduling system determines the minimum clock period that a synchronous sequential circuit can operate without hazards. The timing of each register is adjusted for operation with the minimum clock period. The dependence of the propagation delays of combinational gates on load capacitance values are modeled in order to increase the accuracy of the clock period minimization algorithm. Simulation results show up to 45% speed-up for circuits that are scheduled by the system. For peak power minimization, the dependence of the switching currents of circuit elements on the load capacitance values are modeled. A new method, namely the Shaped Pulse Approximation Method (SPA), is proposed for the estimation of switching power dissipation of circuit elements for arbitrary capacitive loads. The switching current waves can accurately be estimated by using the SPA method with less than 10% normalized rms error. The clock scheduling algorithm of Takahashi for the reduction of the peak power consumption of synchronous sequential circuits is implemented using the SPA method. Up to 73% decrease in peak power

dissipation is observed in simulation results when proper clock scheduling scheme is applied to test circuits.

Keywords: vlsi, clock scheduling, peak power minimization, clock period minimization, vlsi power estimation

ÖZ

SAAT ZAMANLAMASI İLE VLSI TÜMDEVRELERDE BAŞARIMIN İYİLEŞTİRİLMESİ

Kapucu, Kerem
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Murat Aşkar

Aralık 2009, 87 sayfa

Bu çalışmada, senkron devrelerin saat hızları ve tepe güç tüketimlerinin, saat zamanlaması kullanılarak iyileştirilmesi amaçlanmıştır. Saat periyodunu küçültmek için, çevrim aşırma yöntemi kullanılmıştır. Bu yöntemde, yazmaçların saat zamanlaması ayarlanarak, devrenin hızlı birleşimsel bölümlerinden arta kalan zaman, daha yavaş birleşimsel bölümlere aktarılmaktadır. Saat zamanlaması sistemi, senkron devrelerin sorunsuzca çalışabileceği en düşük saat periyodunu belirler. Devrenin, en düşük saat periyodu ile çalışması için, her bir yazmacın zamanlaması ayarlanır. Saat periyodu küçültme yönteminin kesinliğini artırmak için, devre elemanlarının yayılma gecikmesinin sığal yük ile değişimi modellenmiştir. Benzetim sonuçlarına göre, saat zamanlaması sistem tarafından yapılan devrelerde %45'e varan hızlanma gözlenmiştir. Tepe güç tüketiminin azaltılması için, devre elemanlarının anahtarlama akımlarının sığal yük ile değişimi modellenmiştir. Devre elemanlarının herhangi bir sığal yük altındaki anahtarlama güç tüketimlerini tahmin etmeye yarayan, Biçimlendirilmiş Atım Kestirimi (BAK) isimli yeni bir yöntem önerilmiştir. BAK yöntemi ile devre elemanlarının anahtarlama akımları %10'un altında normalize rms hata ile saptanabilmektedir. Takahashi'nin senkron devrelerin tepe güç tüketimini düşürmeye yarayan saat zamanlaması yöntemi, BAK yöntemi kullanılarak oluşturulmuştur. Bu yöntem ile saat zamanlaması yapılan

devrelerin, tepe güç tüketimleri, %73'e varan oranlarda düşürülmüştür.

Anahtar Kelimeler: vlsi, saat zamanlaması, tepe güç azaltma, saat devri kisaltma, vlsi güç tahmini

To my family...

ACKNOWLEDGMENTS

The author wishes to express his deepest appreciation and gratitude for his supervisor Prof. Dr. Murat AŞKAR for his precious guidance, advice, criticism, encouragement and insight throughout the development of this work.

The author would also like to thank his examining committee member Dr. Neslin İsmailoğlu for her valuable suggestions and comments.

The author wishes to thank TÜBİTAK-UZAY for the facilities and environment provided to him throughout the research. The author's colleagues and friends at TÜBİTAK-UZAY are also appreciated for all the support and encouragement he received from them.

The author also wishes to thank TÜBİTAK for their support of his M.Sc. studies with their scholarship.

The author wishes to express his deepest gratitude to Pınar Ün for her invaluable friendship, patience and support that helped him through the hard times during the course of this work.

Last but not least, the author thanks especially to his family for their amazing love, constant support, great patience and extensive encouragement throughout his studies.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

The digital design is dominated by the synchronous approach, where the systems are composed of finite state machines and synchronously clocked registers. The systematic approach in designing the synchronous sequential circuits and the ease of verification have led to dramatic progress in the architectures of the systems and the productivity of the designers.

Owing to the advances in the semiconductor manufacturing technology, the number of elements in integrated circuits are increasing exponentially, following Moore's Law [1]. The scale, speed and power consumption of VLSI circuits has improved rapidly. However, shrinking the feature sizes in VLSI circuits, has dramatically decreased the interconnect thickness; which, in turn, increases resistance of the interconnects and the ratio of routing delay to propagation delay. This puts a limit to the performance improvements in fully synchronous circuits, where simultaneous switching of all the registers are assumed.

Power consumption and speed are among the major constraints in chip design. Power consumption is identified to be in top three overall challenges in chip design for the last five years [2]. For many consumer electronic applications, low average power dissipation is desirable. Moreover, mobile applications such as portable personal communication systems and entertainment devices usually require tight power dissipation constraints. The weight limitations of portable devices arising from the consumer tendency towards smaller and lighter devices put a limit on battery sizes, too. Thus, peak power consumption is as important as average power consumption. Also, the trend towards system-on-chip modules has increased the importance of low power consumption, because heat dissipation is a major problem at that level of integration.

In sequential VLSI circuits, clock signal is generated at a clock source and distributed to the

registers via a clock distribution network. The mismatches in resistance and capacitance of the interconnects due to process variations result in differences in interconnect delays on the clock distribution network. As a result, the clock signals do not arrive at all of the registers at the same time. With the decrease of feature sizes in VLSI circuits, these differences are not negligible anymore. Clock skew is the difference in the arrival times of clock edges to different registers in a circuit due to the differences in the interconnect delays on the clock distribution network.

In 1965, Cotten described a data race mechanism in which clock skew may cause a synchronous circuit to fail [3]. Synchronous circuit designers have made a constant effort to eliminate clock skew since then. Methods are proposed in order to achieve zero clock skew in clock distribution networks and satisfying results are obtained for skew minimization [4]-[9].

In synchronous circuits, all the registers are clocked simultaneously. When a register is clocked, its output signal starts to propagate through the combinational circuit to the input of the next register. In order to eliminate a hazardous operation, this signal must arrive at the input of the next register before the next clock edge. Hence, the minimum clock period is bounded below by the maximum propagation delay between two consecutive registers in the circuit. Taking clock skews into account, the minimum clock period of the circuit is chosen to be larger than the sum of the maximum signal propagation delay and the maximum clock skew to guarantee the correct function [10]. Thus, clock skew is usually perceived as an undesirable phenomenon deteriorating circuit performance and efforts are made towards its elimination.

Conservative design styles, such as those adopted for FPGAs, explicitly discourage "tampering with the clock" [11]. Another approach views clock skew as a "manageable resource than a liability" and increases circuit performance by careful adjustment of clock skews of registers with intentionally introduced delays [12]. In 1990, Fishburn suggested that clock skew can be approached as a means of circuit performance improvement by carefully adjusting the clock timings of each register [13]. Fishburn showed that, the clock period of a synchronous circuit can be minimized by clock skew optimization while maximizing the safety margins against clock hazards [13]. This is the introduction of the basic idea of semi-synchronous circuits, where each register is clocked periodically but not necessarily simultaneously. The conditions for a synchronous circuit to operate correctly with a clock period are given in terms of clock

timings of registers and maximum and minimum propagation delays between registers [13]. The process of adjusting the clock timings of registers of a synchronous circuit for circuit performance improvement is called *clock scheduling* [56].

The clock skew optimization problem is formulated by a linear program [13] and various algorithms are proposed to minimize the clock period of synchronous circuits [14]-[19]. Since the computation time to solve the linear program increases with larger circuits, methods to reduce the size of the linear program are also developed [15], [19].

Exploiting the special form of the timing constraints, graph algorithms are proposed for clock schedule optimization [20]. It is shown that, the clock skew optimization can efficiently be done using directed, weighted graphs to represent setup and hold time constraints [10], [12], [20]-[26]. The minimum feasible clock period is obtained by graph-theoretic approaches with binary search. Bellman-Ford algorithm is used to determine whether a clock period is feasible at each iteration of the binary search [12], [20]-[24]. Since this is a time consuming method, negative cycle detection strategies are utilized in order to fasten the algorithms for large circuits [25]. Graph-theoretic approaches without binary search are also proposed [10], [26]. Clock scheduling algorithms aiming to schedule a circuit with minimum cost to a given infeasible clock schedule are developed [25], [27].

The early works on clock scheduling aimed the minimization of clock period as the optimization goal [13]-[24]. However, clock scheduling has been applied for a number of other quality of circuit improvements, as well. Circuit reliability is improved using clock scheduling [10], [26]-[30]. Clock scheduling is applied to increase the tolerance of synchronous sequential circuits to clock jitter [28] and process variations [32], [33].

Peak current is a primary concern in the design of power distribution networks of VLSI circuits [34]. In order to account for the large current peaks observed in the synchronous digital circuits, the power and ground lines are over-dimensioned [34]. The maximum voltage drop and the probability of failure due to electromigration increases with large peak currents [35], [36]. Demicheli et al. proposed the minimization of peak power supply current by clock scheduling for the first time [34]. It is shown that the current peaks occurring due to the simultaneous switching of the registers and the first stages of combinational circuits can be minimized using clock scheduling without increasing the average power dissipation of the circuit. Also, a clustering method for easy realization of the clock scheduling is given [34].

The power dissipation of synchronous sequential circuits should accurately be estimated for peak power reduction with clock scheduling. Methods are proposed for power estimation of combinational and sequential circuits [37]-[42]. Current waveforms of digital circuits are estimated using probabilistic methods [38], and means of worst-case power estimation are developed [35], [39].The switching probabilities are estimated by calculating the transition densities [37] and taking the effects of glitching into account [41], [42]. In [34] the power consumption is estimated under the assumption that the switching timings of gates are fixed and independent of the clock scheduling. Genetic Algorithm is used to obtain a feasible clock schedule that reduces the peak power dissipation of a circuit. However, since the switching times of the gates in a circuit depend heavily on the clock schedule, the power estimation and the obtained clock schedule are not accurate. Moreover, as a characteristic property of the Genetic Algorithm, the computation time of this method is very long.

Extending the methods in [34], power supply noise suppression [43], [44] and reduction of leakage power is achieved [45]. In [43], the power consumption is estimated by assuming that the switching times of gates depend on the minimum delay from a register and the switching time of the register. However, the gates may switch due to a switching on a non-minimum delay path from a register, which decreases the accuracy of this method. Again the clock scheduling is obtained by a time consuming Genetic Algorithm based method.

Atsushi et al. proposed a fast and more accurate power estimation algorithm and a peak power reduction algorithm based on that power estimation method [46]. In [46], the power consumption of a circuit is modeled as the sum of register originated power consumptions. The switching probabilities of gates are assumed independent of the inputs and calculated iteratively. Clock scheduling for minimum peak power consumption is done by a fast two-stage algorithm [46].

The clock scheduling algorithms are developed under the assumption that the desired clock skews can be introduced to the clock tree. Methods have been proposed for clock routing [7], clock tree synthesis and physical realization [47]-[51] in order to implement the clock schedules efficiently. The minor increase in the power consumption of the clock tree due to the introduction of the clock skews can be eliminated with special clock synthesis algorithms [56].

The thesis study mainly consists of the application of clock scheduling to synchronous se-

quential circuits in order to improve two performance metrics of the circuits: clock period and peak power consumption. In complete-synchronous framework, all the registers are clocked simultaneously and this results in two problems. The first problem is that the minimum clock period of the circuit is determined by the longest combinational delay between two registers, whereas faster combinational paths wait idle for the next clock edge although their outputs are ready. The second problem arises from the simultaneous switching of all the registers in a circuit. This results in a high peak power for a short duration after the clock edge, whereas for the rest of the time the circuit dissipates relatively small static power. In this work, clock scheduling is utilized as a solution to both problems. The clock period of a synchronous sequential circuit can be minimized by cycle stealing. The redundant time of fast combinational paths are transferred to slower combinational paths by adjusting the skew of the registers. By this way, each combinational path between registers is assigned the time it needs to generate the outputs while increasing the overall speed of the circuit. The peak power of synchronous sequential circuits can be minimized by clock scheduling. The power dissipation wave can be suppressed and broadened without disturbing the proper operation of the circuit by carefully adjusting the switching times of the registers. In the chapters that follow, these issues are presented as follows:

The basics of clock scheduling is given in Chapter 2. The method for determining the minimum feasible clock period of a synchronous sequential circuit is explained. Clock scheduling algorithms for increasing the clock frequency of the circuit are discussed. The algorithms for scheduling a circuit with minimum cost to an infeasible clock schedule are also presented.

Chapter 3 covers the algorithm for estimating the peak power consumption of a synchronous sequential circuit. The newly proposed Shaped Pulse Approximation method for estimating the switching current waveform of circuit elements is explained. This chapter also covers how the clock scheduling is utilized to reduce the peak power consumption of a synchronous sequential circuit.

In Chapter 4 the simulations and tests made for the evaluation of the algorithms are discussed. The the delay and switching power characterization of gates and registers are explained in detail. The simulation results showing the performance improvement of the benchmark circuits are presented.

In Chapter 5 the thesis work is summarized, conclusions are drawn and suggestions are made

for future improvements of the proposed power estimation method.

CHAPTER 2

CLOCK SCHEDULING

Synchronous digital circuits can be examined in two frameworks [13, 46]: complete synchronous framework (c-frame) and general synchronous framework (g-frame). In c-frame, all registers are assumed to be simultaneously clocked; whereas, in g-frame registers are not necessarily clocked simultaneously, provided that the periodicity of the clock signals are still preserved. The circuits in which registers are clocked with more than one clock timing are called *semi-synchronous circuits* [27]. G-frame offers a degree of freedom to the circuit designer, which can be used to improve circuit performance metrics such as clock frequency, peak power consumption, and etc. by adjusting the clock arrival times of registers. The procedure of determining the clock arrival timings of registers is called *clock scheduling* [56].

In order to illustrate how the clock period of a synchronous circuit can be minimized by clock scheduling, consider the example circuit in Fig. 2.1. The circuit consists of three stages of registers, u, v and w, with combinational parts in between. Let the propagation delays of combinational parts C_1 and C_2 be 12 units and 4 units, respectively. The minimum clock period for the fully-synchronous clocking scheme is 12 units for this circuit, assuming zero clock skews among the registers. If there are clock skews, the minimum clock period is even required to be larger. However, if a skew of 4 units is applied to the clock line of register v, the circuit can operate with a minimum clock period of 8 units. The time that the output of register v stays idle after propagating to the input of register w is *stolen* and given to register u which has a longer propagation delay. Figures 2.2 and 2.3 show the signal waveforms for the example circuit without and with clock scheduling respectively. This method, proposed by Fishburn, is called *cycle stealing* or *cycle borrowing*.

This chapter starts with the basics of clock scheduling and general synchronous framework.

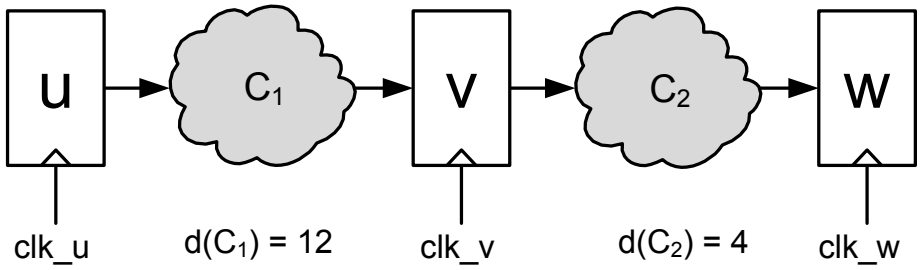

Figure 2.1: The example circuit for illustrating the clock period minimization by cycle stealing method. The circuit consists of three stages of registers and combinational parts in between.

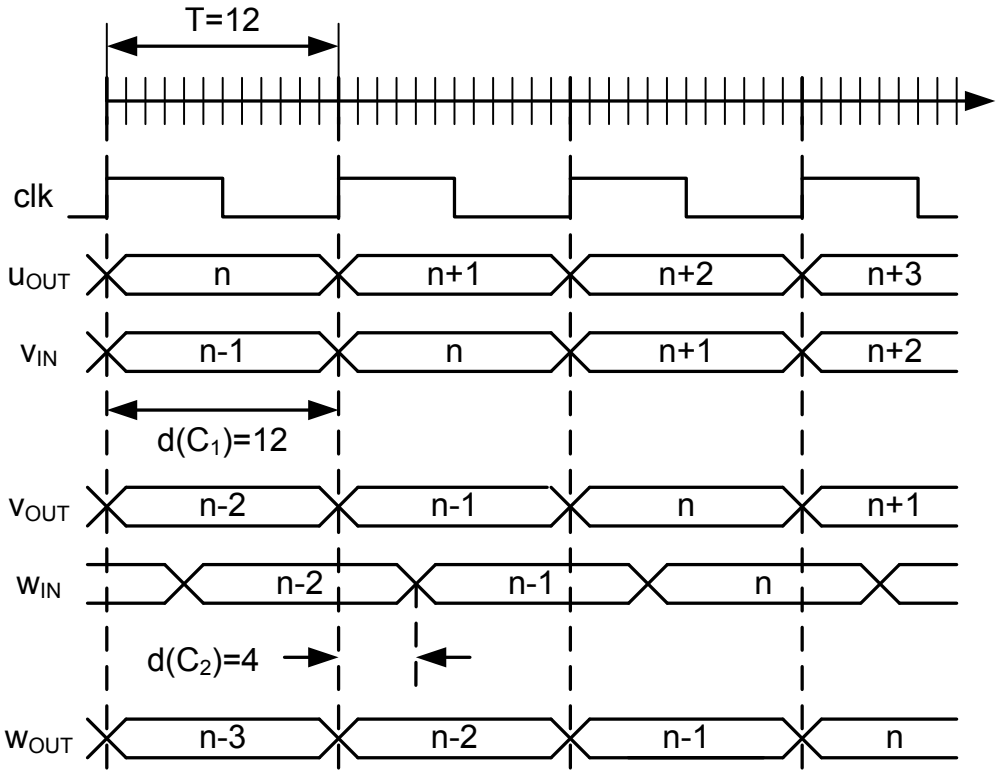

Figure 2.2: Input and output signal waveforms of the registers of the circuit in Fig 2.1 for fully synchronous operation. All the registers are clocked by the same clock with a period of 12 units.

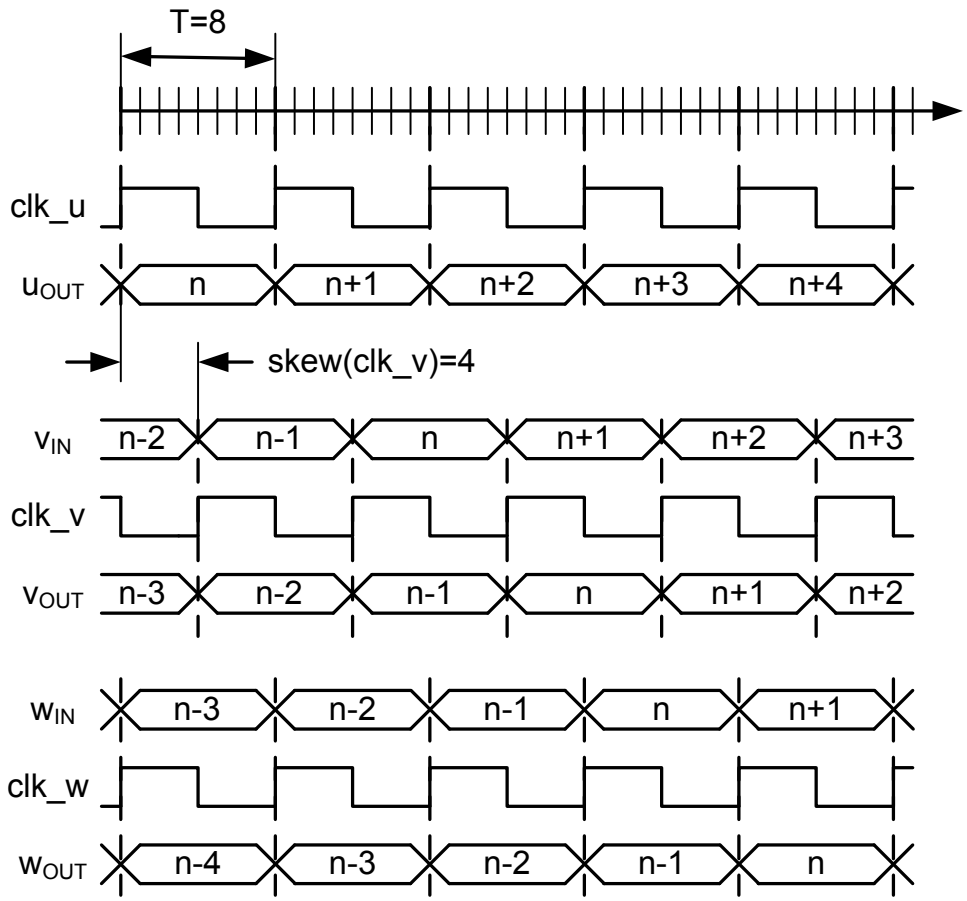

Figure 2.3: Input and output signal waveforms of the registers of the circuit in Fig 2.1 for semi synchronous operation. With the application of 4 units of delay to the clock signal of register *v*, the minimum clock period of the circuit is decreased from 12 units to 8 units. Circuit speed improvement with cycle stealing is illustrated.

The constraints for proper operation of a synchronous circuit and the graph representation of these constraints are discussed in the first section. The second section starts with a brief explanation of systems of difference equations, that are used for representing the constraints for a sequential circuit. The solution of such systems to be used in clock scheduling is also explained. Then, the theory behind the algorithm for finding the minimum clock period is discussed in detail. In the third section, the algorithm for finding the minimum clock period of a semi-synchronous circuit and the algorithm for clock scheduling to operate the circuit with the minimum clock period is explained. The fourth section covers the algorithms used for scheduling a circuit with minimum cost to a given infeasible target clock schedule. Finally, in the fifth section the *clock scheduling engine* that combines these algorithms into a complete

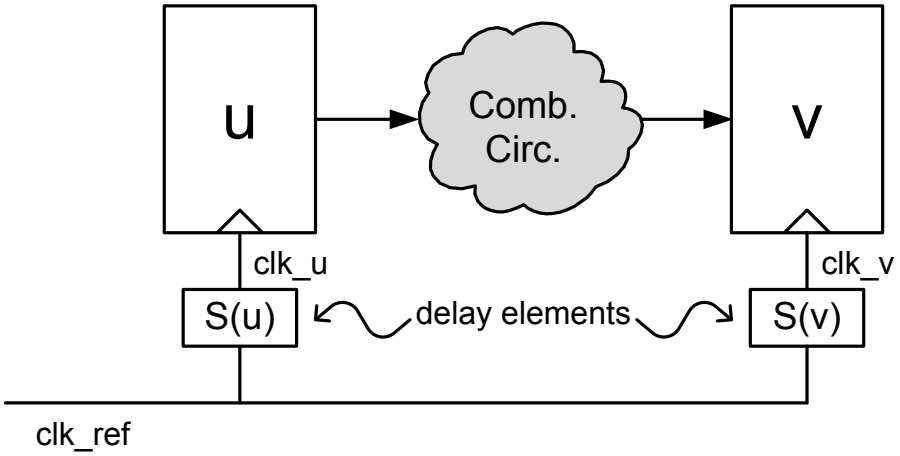

Figure 2.4: Two registers with data path taken as a part of a synchronous circuit

clock scheduling system is discussed.

2.1 General Synchronous Framework

In semi-synchronous circuits the registers are not necessarily simultaneously clocked. The time at which the clock edge arrives at a register v with respect to an arbitrarily chosen, maybe hypothetical, reference register is called the *clock timing* of register v, and denoted by $s(v)$. The clock timings of all registers in a semi-synchronous circuit is called the *clock schedule* of the circuit, $S(v)$. Note that the clock timing of a register is not required to be a unique value; rather, the clock timing can be determined as a range of values. The *clock timing range* of register v denoted by $r(v)$, is defined as the range of clock timings for that register [56] :

$$r(v) = [s_{min}(v), s_{max}(v)]. \tag{2.1}$$

Clock scheduling is the procedure of determining the clock timings of registers of a semi-synchronous circuit [56].

There are two data race mechanisms that may cause failure in synchronous systems [3] such as the one shown in Figure 2.4. The two registers, u and v, are clocked with different clock timings, $S(u)$ and $S(v)$, respectively, and the delay of the combinational path is d. If $s(u) + d < s(v)$, the output data of register u will be sampled twice by the same clock edge, which is called *double-clocking hazard*. Furthermore, if $s(u) + d > s(v) + T$, where T is the clock period, data will not be sampled at all, i.e. data will be lost, which is called *zero-*

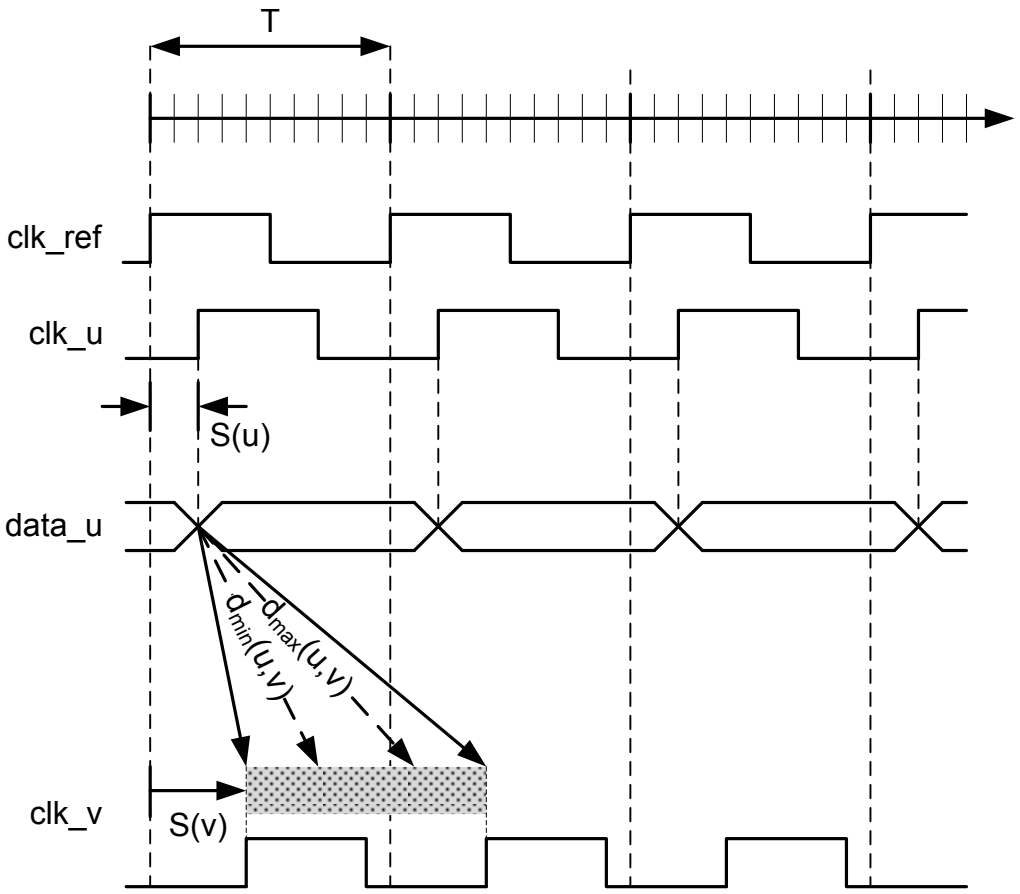

Figure 2.5: Timing diagram for setup and hold time constraints

clocking hazard. In other words, the output of register u should arrive at register v between $s(v)$ and $s(v) + T$ for proper operation without hazards. (See Figure 2.5)

For proper register operation the data should be stable at the register input during the time interval:

$$s(v) - setup(v) < t < s(v) + hold(v), \tag{2.2}$$

where $setup(v)$ is the setup time and $hold(v)$ is the hold time of register v. If the output of register u arrives at register v before $s(v)$, it will be a violation of the *setup-time constraint*, if it arrives later than $s(v) + T$, it will be a violation of the *hold-time constraint*.

In general, two types of constraints must be satisfied for each register pair with signal propagation in order to eliminate the clocking hazards and assure the correct operation of a semi-synchronous circuit [13, 25].

Hold (No Double-Clocking) Constraint:

$$s(v) - s(u) \leq d_{min}(u, v), \tag{2.3}$$

Setup (No Zero-Clocking) Constraint:

$$s(u) - s(v) \leq T - d_{max}(u, v), \tag{2.4}$$

where $d_{min}(u, v)$ is the minimum propagation delay from register u to v, and $d_{max}(u, v)$ is the maximum propagation delay from register u to v along the combinational path between the two registers. As more precisely defined in [25];

$$d_{min}(u, v) = d'_{min}(u, v) - hold(v) - margin_h(v), \tag{2.5}$$

where $d'_{min}(u, v)$ (≥ 0) is the minimum propagation delay from register u to register v, $hold(v)$ (≥ 0) is the hold time of register v, $margin_h(v)$ (≥ 0) is the predefined timing margin. Similarly;

$$d_{max}(u, v) = d'_{max}(u, v) + setup(v) + margin_s(v), \tag{2.6}$$

where $d'_{max}(u, v)$ (≥ 0) is the maximum propagation delay from register u to register v, $setup(v)$ (≥ 0) is the setup time of register v, $margin_s(v)$ (≥ 0) is the predefined timing margin. The timing margin is useful in practical aspects such as securing the feasibility in realizing the clock-timing, securing the reliability of the circuit and providing more room for optimization of circuit performance.

There may be timing constraints for the inputs and outputs of the circuit arising from the interaction of the circuit with other circuits, modules, etc. In order to handle these types of constraints, hypothetical registers outside the circuit with fixed clock timings are assumed to be connected to the inputs and the outputs of the circuit [34]. In the constraint inequalities, the constant input arrival time replaces $s(u)$ for input constraints, and the constant output required time replaces $s(v)$ for output constraints.

A clock schedule is said to be *feasible* for some clock period T if all the constraints are satisfied by that clock schedule for that clock period [10], i.e. there are no hazards or setup/hold time violations and the circuit operates properly.

In order to find a feasible clock schedule, all the constraints should be satisfied at the same time, which requires solving many equations at once. A *constraint graph* $G(V, E)$ is used to

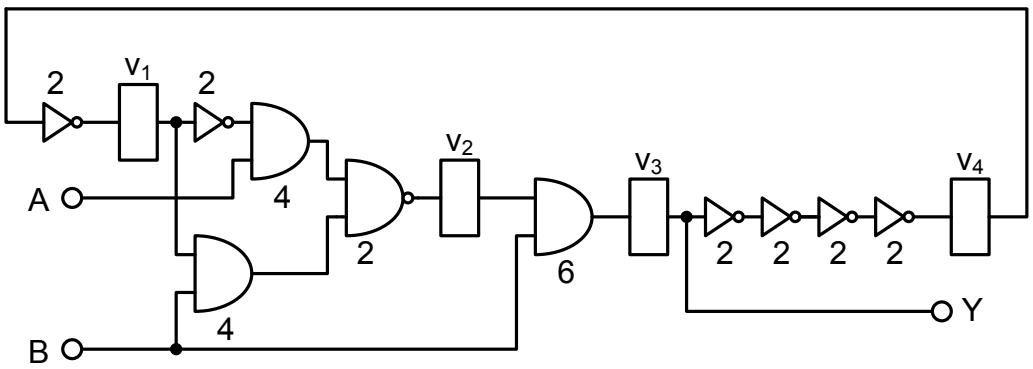

Figure 2.6: Example circuit [58]

represent these constraints [10], [25], [27], [46], [56]. A vertex $v \in V$ in the graph corresponds to a register in the circuit. If a group of registers in the circuit are constrained to be clocked simultaneously, they should be represented by a single vertex in the constraint graph [10]. A directed edge $(u, v) \in E$ in the graph corresponds to either a setup or a hold constraint, named as *setup edge* or *hold edge* respectively. The edge weight $\omega(u, v)$ is d_{min} for hold edges and $T - d_{max}$ for setup edges. Hence, in the constraint graph each directed edge represents the constraint:

$$s(v) - s(u) \leq \omega(u, v). \tag{2.7}$$

The constraint graph G for clock period T is denoted by G_T, similarly the edge weight $\omega(u, v)$ for clock period T is denoted by $\omega_T(u, v)$. The *slack* of an edge is defined as:

$$\Delta_T(u, v) = s(u) + \omega_T(u, v) - s(v). \tag{2.8}$$

An edge (u, v) is *legal* if $s(v) - s(u) \leq \omega_T(u, v)$, i.e., $\Delta_T(u, v) \geq 0$; and it is *illegal* otherwise. An edge with zero slack is called *critical edge*. A clock schedule is feasible for a clock period T if there is no illegal edge in the constraint graph [27], i.e. all constraints are satisfied. A feasible clock schedule is not necessarily unique, since each register may have a range of possible clock timings. A set of clock ranges is said to be *consistent* if a feasible clock schedule is obtained for any clock timing, $s(v)$, chosen within the clock timing range, $r(v)$, for each register $v \in V$ [56].

An example circuit with corresponding constraint graph is given in Fig. 2.6 and Fig. 2.7, respectively. Note that one vertex is assigned for the two inputs since the input timings are assumed to be equal, and another vertex is assigned for the output since the output required

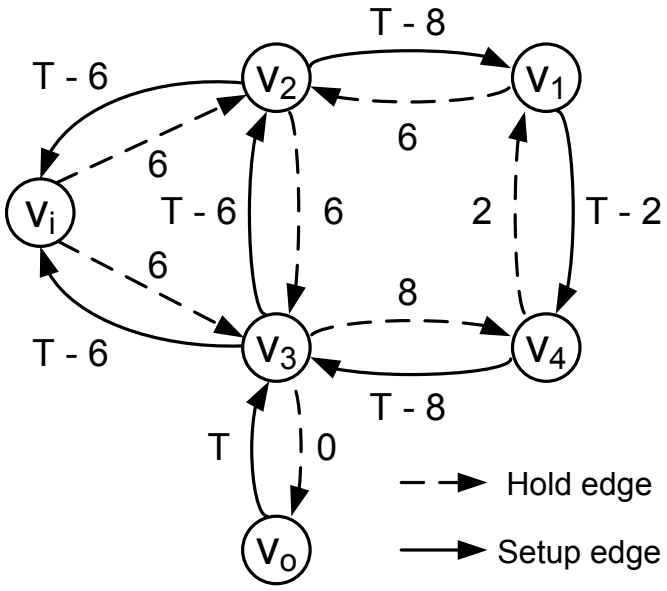

Figure 2.7: Constraint graph of the example circuit

time is assumed to be different than the input timings. Also note that, a single *host* register for representing the inputs and outputs is used if input and output timings are the same.

2.2 Systems of Difference Constraints

The constraints for a sequential circuit to operate without hazards for a given clock schedule are presented in Section 2.1. These constraints are in the form of *difference constraints*. Consider a set of m difference constraints, involving n unknowns, each being a simple linear inequality in the form:

$$x_j - x_i \leq b_k, \tag{2.9}$$

where, $1 \leq i, j \leq n$ and $1 \leq k \leq m$. This is a *system of difference constraints* represented as:

$$Ax \leq b, \tag{2.10}$$

where A is an $m \times n$ matrix, which contains a single "1" and a single "-1" in each row with all remaining entries being "0"; b is an m-vector and x is an n-vector, where x_i's are the unknowns. In order to find the minimum feasible clock period of the circuit, the objective is to find any *feasible solution*, x, that satisfies $Ax \leq b$ or to determine that no feasible solution exists. As the following lemma suggests a feasible clock schedule is not unique.

Lemma 2.2.1 *Let $x = (x_1, x_2, \ldots, x_n)$ be a solution to the system of difference constraints, $Ax \leq b$, and let d be any constant. Then $x + d = (x_1 + d, x_2 + d, \ldots, x_n + d)$ is also a solution to $Ax \leq b$ as well [59].*

Proof. For each x_i and x_j:

$$(x_j + d) - (x_i + d) = x_j - x_i. \tag{2.11}$$

Thus, if x satisfies $Ax \leq b$, so does $x + d$ [59]. □

It is practical to represent the systems of difference constraints with constraint graphs and interpret them graph-theoretically [59]. The system $Ax \leq b$ with n unknowns and m inequalities can be represented by a constraint graph $G(V, E)$ with n vertices and m directed edges. Each vertex v_i in the graph corresponds to an unknown x_i with $1 \leq i \leq n$, and each directed edge (x_i, x_j) with weight b_k in the graph represents one of m inequalities in the form $x_j - x_i \leq b_k$. More formally, given a system $Ax \leq b$ of difference constraints, corresponding constraint graph $G(V, E)$ is a weighted, directed graph with:

$$V = \{v_0, v_1, \ldots, v_n\}, \tag{2.12}$$

and

$$E = \{(v_i, v_j) : x_j - x_i \leq b_k \text{ is a constraint}\} \bigcup \{(v_0, v_1), (v_0, v_2), \ldots, (v_0, v_n)\}. \tag{2.13}$$

Note that, an additional vertex v_0 is added to guarantee that every other vertex is reachable from v_0. This vertex is named as *source* or *root*. In the constraint graph, the weight of directed edge (v_i, v_j) is $\omega(v_i, v_j) = b_k$ corresponding to the constraint inequality $x_j - x_i \leq b_k$. The weight of each edge, (v_0, v_i), leaving the source vertex is zero, that is:

$$\omega(v_0, v_i) = 0 \ \forall v_i \in V. \tag{2.14}$$

The *weight* of a path is defined as the sum of the weights of all edges in the path. The *distance* of a vertex from the source is defined as the weight of the shortest path among all paths between the source and the vertex [25], [59]. The distance of vertex v_i, from the source, v_0, is denoted by $\delta(v_0, v_i)$ or $\delta(v_i)$. A *cycle* is a set of edges, where the source vertex of the first edge and the target vertex of the last edge are the same, with no other mutual vertices among edges. A cycle is called a *critical cycle* if the sum of the weights of the edges in the cycle is zero [61].

Lemma 2.2.2 ***(Triangle Inequality)*** *Let $G(V, E)$ be a weighted, directed graph with weight function $w : E \to \mathfrak{R}$ and source vertex s. If G contains no negative-weight cycle, then the following inequality must be satisfied for all edges $(u, v) \in E$ [59]:*

$$\delta(s, v) \leq \delta(s, u) + \omega(u, v). \tag{2.15}$$

Proof. Let p be a shortest path from source s to vertex v. Then, the weight of p is not more than the weight of any other path from s to v. Specifically, the weight of path p, is not more than the weight of the particular path that consists of the shortest path from source s to vertex u and edge (u, v)[59]. □

The following theorem states that, utilizing shortest path algorithms on the constraint graph determines the existence of a solution to the constraint system and finds a feasible solution, if exists.

Theorem 2.2.3 *Let $G(V, E)$ be the corresponding constraint graph of a system of difference constraints, $Ax \leq b$. If G contains no negative-weight cycles, then a feasible solution for the system is given by [59]:*

$$x = (\ \delta(v_0, v_1),\ \delta(v_0, v_2),\ \delta(v_0, v_3), \ldots,\ \delta(v_0, v_n)\ ). \tag{2.16}$$

Moreover, if G contains a negative-weight cycle, then there is no feasible solution for the system [59].

Proof. Consider any edge $(v_i, v_j) \in E$ of the constraint graph G. By the triangle inequality,

$$\delta(v_0, v_j) \leq \delta(v_0, v_i) + \omega(v_i, v_j), \tag{2.17}$$

or equivalently,

$$\delta(v_0, v_j) - \delta(v_0, v_i) \leq \omega(v_i, v_j). \tag{2.18}$$

Thus, letting $x_i = \delta(v_0, v_i)$ and $x_j = \delta(v_0, v_j)$ satisfies the difference constraint that corresponds to edge (v_i, v_i): $x_j - x_i \leq \omega(v_i, v_j)$. Hence, if the constraint graph contains no negative-weight cycles, then (2.16) gives a feasible solution [59].

Suppose that the constraint graph G has a negative-weight cycle. Without loss of generality, let the negative-weight cycle be $C = \langle v_1, v_2, \ldots, v_k, v_1 \rangle$. Note that, source vertex v_0 can not

be on the negative cycle, since it has no entering edges. The edges in cycle C represents the following difference constraints:

$$x_2 - x_1 \leq \omega(v_1, v_2), \tag{2.19}$$

$$x_3 - x_2 \leq \omega(v_2, v_3), \tag{2.20}$$

$$x_4 - x_3 \leq \omega(v_3, v_4), \tag{2.21}$$

$$\vdots$$

$$x_k - x_{k-1} \leq \omega(v_{k-1}, v_k), \tag{2.22}$$

$$x_1 - x_k \leq \omega(v_k, v_1). \tag{2.23}$$

Suppose that, there is a solution for x that satisfies each of the k inequalities. Then, this solution must also satisfy the sum of these k inequalities:

$$x_2 - x_1 + x_3 - x_2 + \cdots + x_k - x_{k-1} + x_1 - x_k \leq \omega(v_1, v_2) + \omega(v_2, v_3) + \cdots + \omega(v_{k-1}, v_k) + \omega(v_k, v_1), \tag{2.24}$$

which reduces to:

$$0 \leq \omega(C), \tag{2.25}$$

because on the left-hand side of the inequality each unknown is added once and subtracted once. On the right-hand side of the inequality, the weights of all the edges in the cycle C are summed up to be the weight of the cycle C. But since C is a negative-weight cycle,

$$\omega(C) < 0, \tag{2.26}$$

resulting in a contradiction:

$$0 \leq \omega(C) < 0. \tag{2.27}$$

Thus, there is no feasible solution for a constraint system, if there is a negative-weight cycle in the corresponding constraint graph. □

As shown by Theorem 2.2.3, shortest path algorithms such as Bellman-Ford Algorithm can be used to solve systems of difference constraints. Note that Dijkstra's Algorithm is useless due to the existence of negative-weight edges in the constraint graph. The added source vertex v_0 guarantees that, any negative-weight cycle in the constraint graph is reachable from source vertex v_0, since there are out-edges from v_0 to all the other vertices in the constraint graph. Bellman-Ford Algorithm converges, i.e., the distances of all vertices are finite, if there is no negative-weight cycle in the constraint graph. If Bellman-Ford Algorithm returns TRUE, the

distance of each vertex to the source gives a feasible solution to the system of difference constraints. If Bellman-Ford Algorithm returns FALSE, it means a feasible solution to the system of difference constrains does not exist. The constraint graph $G(V, E)$ for a system of m difference constraints with n unknowns, has $n + 1$ vertices and $n + m$ edges. Hence, Bellman-Ford Algorithm solves the system in $O((n + 1)(n + m)) = O(n^2 + nm)$ time [59].

Lemma 2.2.4 *A clock period T is feasible, if constraint graph G_T for that clock period contains no negative cycle [61].*

Proof. Follows directly from Theorem 2.2.3. □

Lemma 2.2.5 *If a clock period T is not feasible,then the clock period $T' \leq T$ is not feasible [61].*

Proof. If clock period T is not feasible, then there is a negative-weight cycle in the constraint graph G_T. Let C be the negative-weight cycle, consisting of hold-edges C_h and setup-edges C_s, then:

$$\omega(C) \;=\; \sum_{e_h \in C_h} d_{min}(u, v) \;+\; \sum_{e_s \in C_s} T - d_{max}(u, v) \;\leq\; 0, \tag{2.28}$$

where $e_h = (u, v)$ is a hold edge and $e_s = (u, v)$ is a setup edge of the negative-weight cycle.

For $T' \leq T$:

$$\sum_{e_s \in C_s} T' - d_{max}(u, v) \;\leq\; \sum_{e_s \in C_s} T - d_{max}(u, v), \tag{2.29}$$

Hence;

$$\sum_{e_h \in C_h} d_{min}(u, v) \;+\; \sum_{e_s \in C_s} T' - d_{max}(u, v) \;\leq\; \sum_{e_h \in C_h} d_{min}(u, v) \;+\; \sum_{e_s \in C_s} T - d_{max}(u, v) \;\leq\; 0. \tag{2.30}$$

Thus, C will be a negative-weight cycle for clock period T', thus T' is not feasible. □

Lemma 2.2.6 *If a clock period T is feasible, then the clock period $T' \geq T$ is also feasible [61].*

Proof. Assume that T' is not feasible. Then, by Lemma 2.2.5, $T \leq T'$ is not feasible, which is a contradiction. Hence, T' is feasible [60]. □

Lemma 2.2.7 ***(Upper Bound on Clock Period)*** *An upper bound, U_T, on the clock period is given by [25], [60]:*

$$T \leq U_T = \max_{(u,v)\in E_S} \left\{ d_{max}(u,v) \right\} < \infty. \tag{2.31}$$

Proof. From the setup edge constraint given in 2.4, for zero clock skew condition, i.e., all registers are scheduled at 0:

$$T \leq d_{max}(u,v), \tag{2.32}$$

which is a feasible clock period for even the zero clock skew case and an upper bound for the minimum feasible clock period [25], [60]. □

Lemma 2.2.8 ***(Lower Bound on Clock Period)*** *A lower bound, L_T, on the clock period is given by [25], [60]:*

$$T \geq L_T = \max \left\{ \max_{(u,u)\in E_s} \{ d_{max}(u,u) \} , \max_{(u,v)\in E} \left\{ d_{max}(u,v) - d_{min}(u,v) \right\} \right\} \geq 0, \tag{2.33}$$

where $d_{max}(u,u) = 0$ if there is no signal path from a register to the same register, and E_S is the set of all setup edges.

Proof. The propagation delay from a register to the same register is obviously a lower bound on the clock period [25], [60].

For the second term, consider the setup and hold edge constraints given in (2.3) and (2.4) respectively, which are satisfied for each register pair with signal propagation:

$$s(v) - s(u) \leq d_{min}(u,v), \tag{2.34}$$

$$s(u) - s(v) \leq T - d_{max}(u,v). \tag{2.35}$$

Summing both sides of the inequalities gives:

$$0 \leq T - d_{max}(u,v) + d_{min}(u,v), \tag{2.36}$$

Hence;

$$d_{max}(u,v) - d_{min}(u,v) \leq T, \tag{2.37}$$

for all (u,v), for a feasible clock period T, which also holds for minimum feasible clock period T_{min}:

$$d_{max}(u,v) - d_{min}(u,v) \leq T_{min}, \tag{2.38}$$

for all edges (u, v). Hence, a lower bound for minimum feasible clock period:

$$\max_{(u,v)\in E} \left\{ d_{max}(u, v) - d_{min}(u, v) \right\} \leq T_{min}. \tag{2.39}$$

Consequently, the lower bound is the maximum of the two terms. □

Theorem 2.2.9 *A finite solution to the clock period minimization problem exists [60].*

Proof. Following directly from Lemmas' 2.2.7 and 2.2.8, the minimum feasible clock period is bounded by:

$$0 \leq L_T \leq T \leq U_T < \infty. \tag{2.40}$$

□

For the sake of completeness, the following lemmas from [61] are given without proof.

Lemma 2.2.10 *The constraint graph contains at least one critical cycle when the clock period is the minimum feasible clock period [61].*

Lemma 2.2.11 *For the constraint graph with a feasible clock schedule; each directed edge in the critical cycle is a critical edge [61].*

2.3 Minimum Feasible Clock Period

Theorem 2.2.9 states that the minimum clock period of a given sequential circuit is finite and bounded. Moreover, Theorem 2.2.3 states that shortest path algorithms such as Bellman-Ford Algorithm can determine if a clock period is feasible or not. Thus a binary search can be made to find the minimum feasible clock period [12], [25], [60]. The algorithm for finding the minimum feasible clock period similar to those in [12], [12], [60], is given in Figure 2.8. A binary search is made for the minimum feasible clock period in the interval bounded by the upper and lower bounds given in Lemmas 2.2.7, and 2.2.8, where at each step the interval is halved until the minimum feasible clock period is determined with an error less than ϵ. The upper and lower bounds are checked before the binary search is initiated. If the lower bound is feasible, then it is returned as the minimum feasible clock period. If the upper bound is not feasible, the algorithm returns *infinity*, meaning that there is no feasible clock period.

Procedure MinClock($G(V, E)$)
Input: constraint graph $G(V, E)$
Output: minimum feasible clock period T

1: $L_{self} := \max_{(u,u)\in E} d_{max}(u, u)$
2: $L_{diff} := \max_{(u,v)\in E}\{d_{max}(u, u) - d_{min}(u, v)\}$
3: $L := \max\{L_{self}, L_{diff}\}$
4: $T := \max_{(u,v)\in E} d_{max}(u, v)$
5: **if IsNoNegCycle(** G**,** L **) = "Yes" then**
6: **return** L
7: **end if**
8: **if IsNoNegCycle(** G**,** T **) = "No" then**
9: **return** ∞
10: **end if**
11: **while** $T - L \geq \epsilon$ **do**
12: $M := (T + L)/2$
13: **if IsNoNegCycle(** G**,** T **) = "Yes" then**
14: $T := M$
15: **else**
16: $L := M$
17: **end if**
18: **end while**
19: **return** T

Figure 2.8: Minimum feasible clock period algorithm.

At each step of the binary search, the feasibility of the clock period is checked using Bellman-Ford Algorithm. If there is a negative-weight cycle in the graph, Bellman-Ford Algorithm does not converge and the update procedure is repeated as many times as the number of vertices. Hence, a negative-weight cycle detection method is needed to determine the existence of a negative-weight cycle in the graph at early stages.

There are a number of negative-weight cycle detection strategies [57], one of which is the "walk-to-the-root" method. In "walk-to-the-root" method, a negative-weight cycle is detected by following parent pointers from a vertex. If source s is not reachable from a vertex, that means there is a negative-weight cycle in the graph. In [57], the checking is made before the labeling operation with respect to edge (u, v). If v is reached on the way from u to s, negative cycle is detected since the labeling with respect to (u, v) will create a negative cycle. In [25], a slightly modified walk-to-the-root method is proposed, where the checking is made after all the labeling operations to the neighbors of u are finished. If these labeling operations create a cycle, the path from u contains u. Also, labeling from u to r creates a negative cycle; which is

Procedure IsNoNegCycle($G(V, E)$, T)
Input: constraint graph $G(V, E)$, clock period T
Output: **Yes** or **No**

1: Construct G_T, add source vertex s, add edges $\{ (s, u) : \forall u \in V, \omega(s, u) = 0 \}$ // Init.
2: $Q_1 \leftarrow \emptyset$, $Q_2 \leftarrow \emptyset$ // Init.
3: **for all** $v \in V$ **do**
4: $\delta(v) := \infty$ // Initially all distances set to infinity
5: **end for**
6: $\delta(s) = 0$, push s to Q_1 // Init.
7: **while** Q_1 is not empty **do** // Update Procedure (steps 7-21)
8: $u \leftarrow pop(Q_1)$
9: **for all** v adjacent to u in G **do**
10: **if** $\delta(v) \geq \delta(u) + \omega_T(u, v)$ **then** // Labeling operation (steps 10-16)
11: $\delta(v) := \delta(u) + \omega_T(u, v)$
12: set parent pointer from v to u
13: **if** v is not in Q_2 **then**
14: push v to Q_2
15: **end if**
16: **end if**
17: **end for**
18: **if** reach u by following parent pointer from u **then** // Negative-weight cycle exists!
19: **return "No"**
20: **end if**
21: **end while**
22: **if** Q_2 is empty **then** // No negative-weight cycle
23: **return "Yes"**
24: **end if**
25: $Q_1 \leftarrow Q_2$, $Q_2 \leftarrow \emptyset$
26: **goto** step 7

Figure 2.9: Negative cycle detection algorithm: Bellman-Ford Algorithm with modified walk-to-the-root strategy.

also checked by the algorithm. Note that in [25], a *root* vertex r, is selected among the vertices instead of adding to the constraint graph a *source* node s, with only out-edges. Hence, the labeling from u to r needs to be checked as a part of walk-to-the-root method.

In this work, a slightly modified version of the approach in [25] is used. A source node s, with only out edges is added to the constraint graph in the beginning and the negative-weight cycle checking is made at a vertex after all labeling operations are made from that vertex. By this way, the checking of labeling from u to s is eliminated since the source vertex does not have any entering edges. The overhead of "walk-to-the-root" method for constraint graphs is less than 1% for graphs with no negative-weight cycles [25].

The accuracy of the algorithm depends on the determination of propagation delays of the elements in the circuit. Neglecting the dependence of the propagation delay on the load capacitance value or using coarsely tabulated values may cause errors in minimum clock period calculation. In this work, the variation of the propagation delay of each gate type with respect to the load capacitance value is modelled as discussed in Subsection 4.1.1. The propagation delay of each gate according to the actual capacitive load of the gate in the circuit is estimated by using the model and the constraint graph is constructed accordingly.

Let n be the number of vertices, m be the number of edges and k be the number of distinct edges in a shortest trail. The time complexity of the initialization (steps 1-6) is $O(n)$; the time complexity of the labeling operation (steps 10-16) is $O(1)$; the time complexity of walk-to-the-root (steps 18-20) is $O(k)$. Update procedure (steps 7-21) is repeated at most k times and in each update, labeling is made at most m times and walk-to-the-root is done at most n times [25]. Hence, the time complexity of the negative-weight cycle detection algorithm is:

$$O(n) \; + \; k \cdot (m \cdot O(1) \; + \; n \cdot O(k)) \; = \; O(km + kn^2). \tag{2.41}$$

The time complexity of the negative-weight cycle detection algorithm in this work is the same as that in [25], which is better than that in [57] with a minor difference.

The time complexity of finding upper and lower bounds for the binary search algorithm are $O(m)$. Hence, the time complexity of the minimum-feasible clock period algorithm is:

$$O\Big(\gamma(km + kn^2)\Big), \tag{2.42}$$

where γ is the number of repetitions [25]. Experimentally, $k < 100$ and $\gamma < 20$ are found for most circuits.

2.4 Minimum Cost Scheduling

The *cost* of a clock schedule is defined as the sum of differences between the clock schedule S and a target clock schedule O for all registers of a circuit [25], [27]. Minimum cost scheduling algorithm finds a feasible clock schedule S of a circuit for a given clock period T, with minimum cost from a given target clock schedule O. Obviously, if the target clock schedule is feasible, this is an easy task. Hence, throughout the discussion in this section, the target clock schedule is assumed to be infeasible.

Figure 2.10 shows the minimum cost scheduling algorithm. Firstly, an initial feasible clock schedule is found by using the Bellman-Ford algorithm as shown in Fig. 2.11. This algorithm simply chooses a *root* vertex and calculates the distance of every node with respect to the *root*. The distance of a register to the root $\delta(r, v)$, is assigned as the initial clock timing of each register, i.e., $s(v) = \delta(v)$. Then, the clock timing $s(v)$ of each register v is replaced with $s(v) - o(v)$, regarding the target clock timing of each register as zero. The initial clock schedule is iteratively improved, by using TryDec and TryInc given in Figs. 2.12 and 2.13, respectively. The iterations continue until no further improvement can be made on the initial clock schedule. After all the iterations are finished, the clock timing $s(v)$ of each register v is replaced by $s(v) + o(v)$.

FindCRD algorithm [27], finds a set of registers such that the overall cost can be reduced by decreasing the clock timing of each register in the set by the same amount. The optimal amount of decrease is calculated by TryDec. Similarly, FindCRI algorithm, finds a set of registers such that the overall cost can be reduced by increasing the clock timing of each register in the set by the same amount. The optimal amount of increase is calculated by TryInc.

2.4.1 Finding the Optimal Amount of Clock Timing Adjustment

Let $G(V, E)$ be a constraint graph of a circuit with a feasible clock schedule S and let $R \subseteq V$ be a set of registers of the circuit. Consider another clock schedule for the same circuit S' such that:

$$s'(v) = \begin{cases} s(v) - \alpha, & \text{if } v \in R \\ s(v), & \text{otherwise.} \end{cases} \tag{2.43}$$

If vertices u and v are both in R or are both in $V \backslash R$, the edge (u, v) is legal. The condition for the edge (u, v) to be legal in S' when $u \in R$ and $v \notin R$ is:

$$s(v) - \big(s(u) - \alpha\big) \leq \omega(u, v). \tag{2.44}$$

Hence;

$$\alpha \leq s(u) - s(v) + \omega(u, v) = \Delta(u, v). \tag{2.45}$$

Procedure MinCost($G_T(V, E_T)$, O)
Input: Constraint graph $G_T(V, E_T)$, target clock timing of all registers, $O = \{ o(v) : \forall v \in V \}$
Output: Clock schedule $S = \{ s(v) : \forall v \in V\}$

```
1: FindInitialSchedule( G_T(V, E_T) )  // Get an initial feasible clock-schedule
2: if s(v) = o(v) forall v ∈ V then
3:    return S
4: end if
5: for all v ∈ V do
6:    s(v) := s(v) − o(v)
7: end for
8: do
9:    do
10:      S := TryDec( G_T, S )
11:   while S is modified
12:   do
13:      S := TryInc( G_T, S )
14:   while S is modified
15: while S is modified
16: for all v ∈ V do
17:    s(v) := s(v) + o(v)
18: end for
19: return S
```

Figure 2.10: Minimum cost scheduling algorithm.

Similarly the condition for the edge (u, v) to be legal in S' when $u \notin R$ and $v \in R$ is:

$$(s(v) - \alpha) - s(u) \leq \omega(u, v). \tag{2.46}$$

Hence;

$$-\Delta(u, v) = s(u) - s(v) + \omega(u, v) \leq \alpha. \tag{2.47}$$

Thus, S' is feasible if and only if [25]:

$$L_R \leq \alpha \leq U_R, \tag{2.48}$$

where,

$$L_R = \max_{u \notin R, v \in R} \{ - \Delta(u, v)\}, \tag{2.49}$$

$$U_R = \max_{u \in R, v \notin R} \{\Delta(u, v)\}. \tag{2.50}$$

In order to minimize the cost of clock schedule S', clock timings of registers in R should be changed by such an amount that, at the end the number of registers with positive clock timing

Procedure FindInitialSchedule($G(V, E)$)
Input: Constraint graph $G(V, E)$
Output: Clock schedule $S = \{ s(v) : \forall v \in V\}$

1: Select a *root* node, r
2: Calculate the distance $\delta(r, v)$ of every node $v \in V$ to the root r using Bellman-Ford Algorithm
3: **for all** $v \in V$ **do**
4: $\quad s(v) = \delta(r, v)$
5: **end for**
6: **return** S

Figure 2.11: Algorithm that finds an initial feasible clock schedule.

is equal to the number of registers in with negative clock timing. If such a clock schedule is infeasible, then the difference between the number of registers with positive clock timing and the number of registers with negative clock timing should be minimized [27].

As defined in [27]; let β be zero, if the number of registers in R is even and $\left\lfloor \frac{n+1}{2} \right\rfloor$-th largest clock timing in R is non-negative in S and $\left\lceil \frac{n+1}{2} \right\rceil$-th largest clock-timing in R is non-positive in S. Otherwise, let β be the $\left\lfloor \frac{n+1}{2} \right\rfloor$-th largest clock timing of registers in R in S. The following lemma states the optimum amount of clock timing adjustment that minimizes the cost of the clock scheduling, α, in terms of β.

Lemma 2.4.1 *The optimal α, that minimizes the cost of S' is [27]:*

$$\alpha_{opt} = \begin{cases} \beta, & L_R \leq \beta \leq U_R \\ L_R, & \beta \leq L_R \\ U_R, & U_R \leq \beta \end{cases} \tag{2.51}$$

Obviously, if α_{opt} is zero, the cost of the clock scheduling can not be changed by the set of registers. For the set of registers to reduce the cost, α_{opt} must be positive or negative. α_{opt} is negative if and only if both L_R and β are negative, that is, there is no critical *in-edge* to R in S, and the number of registers in R with negative clock timing in S is larger than the number of registers in R with positive clock timing in S. Similarly, α_{opt} is positive if and only if both B_R and β are positive, that is, there is no critical *out-edge* from R in S, and the number of registers in R with positive clock timing in S is larger than the number of registers in R with

Procedure TryDec($G_T(V, E_T)$, S)
Input: Constraint graph $G_T(V, E_T)$, and current clock schedule $S = \{s(v) : \forall v \in V\}$
Output: **TRUE** or **FALSE**

1: $R \leftarrow \emptyset$
2: $\beta := 0$, $\alpha_{opt} := 0$
3: R :=**FindCRD(** $G_T(V, E_T)$ **)**
4: **if** R is empty **then**
5: **return FALSE**
6: **end if**
7: $U_R := \min_{u \in R, v \notin R}(\Delta(u, v))$
8: $L_R := \max_{u \notin R, v \in R}(-\Delta(u, v))$
9: **if** number of vertices in R is odd **then**
10: $\beta := \left\lfloor \frac{n+1}{2} \right\rfloor$-th largest clock-timing in R
11: **else if** $\left\lfloor \frac{n+1}{2} \right\rfloor$-th largest clock-timing in R is non-negative **and**
$\left\lceil \frac{n+1}{2} \right\rceil$-th largest clock-timing in R is non-positive **then**
12: $\beta := 0$
13: **end if**
14: **if** $L_R \leq \beta \leq U_R$ **then**
15: $\alpha_{opt} := \beta$
16: **else if** $\beta < L_R$ **then**
17: $\alpha_{opt} := L_R$
18: **else if** $U_R < \beta$ **then**
19: $\alpha_{opt} := U_R$
20: **end if**
21: **for all** $v \in R$ **do**
22: $s(v) := s(v) - \alpha_{opt}$
23: **end for**
24: **return TRUE**

Figure 2.12: TryDec algorithm

negative clock timing in S. The methods of finding such sets of registers will be discussed in the next subsection.

2.4.2 Finding Cost Reducible Register Sets

A set of registers R is called *cost reducible by decrease* (CRD), if the cost of the clock schedule can be reduced by decreasing the clock timing of each register in R by the same amount without violating the setup and hold time constraints [25]. Similarly, a set of registers R is called *cost reducible by increase* (CRI), if the cost of the clock schedule can be reduced by increasing the clock timing of each register in R by the same amount without violating the

Procedure TryInc($G_T(V, E_T)$, S)
Input: Constraint graph $G_T(V, E_T)$, and current clock schedule $S = \{s(v) : \forall v \in V\}$
Output: **TRUE** or **FALSE**

1: $R \leftarrow \emptyset$
2: $\beta := 0$, $\alpha_{opt} := 0$
3: R :=**FindCRI**($G_T(V, E_T)$)
4: **if** R is empty **then**
5: **return FALSE**
6: **end if**
7: $U_R := \min_{u \in R, v \notin R}(\Delta(u, v))$
8: $L_R := \max_{u \notin R, v \in R}(-\Delta(u, v))$
9: **if** number of vertices in R is odd **then**
10: $\beta := \left\lfloor \frac{n+1}{2} \right\rfloor$-th largest clock-timing in R
11: **else if** $\left\lfloor \frac{n+1}{2} \right\rfloor$-th largest clock-timing in R is non-negative **and**
$\left\lceil \frac{n+1}{2} \right\rceil$-th largest clock-timing in R is non-positive **then**
12: $\beta := 0$
13: **end if**
14: **if** $L_R \leq \beta \leq U_R$ **then**
15: $\alpha_{opt} := \beta$
16: **else if** $\beta < L_R$ **then**
17: $\alpha_{opt} := L_R$
18: **else if** $U_R < \beta$ **then**
19: $\alpha_{opt} := U_R$
20: **end if**
21: **for all** $v \in R$ **do**
22: $s(v) := s(v) - \alpha_{opt}$
23: **end for**
24: **return TRUE**

Figure 2.13: TryInc algorithm

setup and hold time constraints.

In order for a set of registers to be a cost reducible set, the optimum amount of adjustment, α_{opt}, of the registers must be non-zero. If α_{opt} is negative, the set is a CRI set; if α_{opt} is positive, the set is a CRD set.

As Lemma 2.4.1 suggests, given a constraint graph $G(V, E)$ with a clock schedule S; a set of registers R, is a CRD set if there is no critical out-edge from R in S and the number of registers in R with positive clock timing in S is larger than the number of registers in R with negative clock timing in S [25]. Considering the critical graph $G_C(V, E_C)$, consisting of only the critical edges of $G(V, E)$, and labeling the registers with $s(v) > 0$ as "+" and the registers with $s(v) \leq 0$

Procedure FindCRD($G_T(V, E_T)$, S)
Input: Constraint graph $G_T(V, E_T)$, and current clock schedule $S = \{s(v) : \forall v \in V\}$
Output: Set R of cost reducible by decrease vertices or empty set if no such vertex exist

1: $R \leftarrow \emptyset$
2: $R' \leftarrow \emptyset$
3: Construct critical graph $G_C(V, E_C)$ that consists of only critical edges
4: Construct $G'_H(V, E'_H)$, the transitive graph of $G_C(V, E_C)$ using Warshall's Algorithm
5: **for all** $v \in V$ **do**
6: **if** $s(v) > 0$ **then**
7: Label: $v \leftarrow$ "+"
8: **else if** $s(v) \leq 0$ **then**
9: Label: $v \leftarrow$ "-"
10: **end if**
11: **end for**
12: Construct $G_H(V, E_H)$ from $G'_H(V, E'_H)$ by deleting all edges in E_H except edges going from "+" to "-" vertices
13: Obtain a maximum matching of $G_H(V, E_H)$ by using Hopcroft-Karp Algorithm, and reverse the matched edges to obtain $G^*_H(V, E_H)$
14: **if** all "+" vertices matched **then**
15: **return** $R = \emptyset$
16: **end if**
17: Select an unmatched "+" vertex, v_S
18: $R' = \{v \in V : v$ is reachable from v_s in $G^*_H\}$
19: $R = R' \bigcup \{$vertex v reachable from R' in $G_C\}$
20: **return** R

Figure 2.14: Algorithm that finds cost reducible by decrease sets

as "–"; the definition can be re-stated as follows: Given a critical graph $G_C(V, E_C)$, a set of registers R, is a CRD set if there is no critical out-edge from R and the number of "+" registers in R is larger than the number of "–" registers in R [25].

The algorithm to find a CRD set is given in Figure 2.14. The algorithm takes a constraint graph $G_T(V, E_T)$ for a clock period T and the current clock schedule S of the circuit as input. The output of the algorithm is the set R of cost reducible by decrease vertices. If the algorithm returns an empty set, that means there is no set of CRD registers. The register set R, returned by the algorithm has more "+" registers than "–" registers and there are no edges (u, v) in G_C with $u \in R$ and $v \notin R$. The algorithm starts by constructing the critical graph $G_C(V, E_C)$ consisting of only critical edges. Then, the transitive graph $G'_H(V, E'_H)$ is constructed using Warshall's Algorithm for transitive closure [59]. The vertices with timing greater than zero ($s(v) > 0$) are labeled as "+" and the vertices with timing less than or equal to zero ($s(v) \leq 0$)

Warshall's Algorithm for transitive closure
Input: Directed graph $G(V, E)$
Output: Transitive closure matrix T, such that $T[u, v] = 1$ if there is a path from u to v; $T[u, v] = 0$ if there is no path from u to v

```
1:  for all u ∈ V do
2:    for all v ∈ V do
3:      if u = v or (u, v) ∈ E then
4:        T[u, v] ← 1
5:      else
6:        T[u, v] ← 0
7:      end if
8:    end for
9:  end for
10: for all w ∈ V do
11:   for all u ∈ V do
12:     for all v ∈ V do
13:       T[u, v] ← T[u, v] ∨ (T[u, w] ∧ T[w, v])
14:     end for
15:   end for
16: end for
17: return T
```

Figure 2.15: Algorithm that computes the transitive closure of a graph.

are labeled as "–". After that, the graph $G_H(V, E_H)$ is constructed from $G'_H(V, E'_H)$, consisting of only "+" to "–" edges. A maximum matching of $G_H(V, E_H)$ is obtained using Hopcroft-Karp Algorithm [62], [63], [64] and the matched edges are reversed to obtain $G^*_H(V, E_H)$. At this point, if all the "+" are matched, the algorithm returns an empty set, because no CRD set of vertices exist. Else, an unmatched "+" vertex, v_S, is selected and the set R' is constructed with all the vertices reachable from v_S in $G^*_H(V, E_H)$. Finally, the set R is constructed by adding to R' a vertex reachable from R' in $G_C(V, E_C)$.

The following theoretical discussion stated in [27] provides a theoretical basis for the Find-CRD algorithm and shows the correctness of the FindCRD algorithm.

Theorem 2.4.2 *There is no set of CRD vertices in the graph, if all "+" vertices are matched in step 14 of FindCRD algorithm [27].*

Proof. Assume that all "+" vertices are matched and there exists a CRD set of vertices $P \subseteq V$. That is, there is no edge (u, v) in G_C with $u \in P$ and $v \notin P$ and there are more "+" vertices than "–" vertices in P. Since all "+" vertices are matched, there exists a "–" vertex v for

every "+" vertex u in P and there is a path from u to v in G_C. Since P is a CRD set, there can not be any out-edges in the critical graph for vertices in P.Therefore, vertex v is in P, making the number of "+" vertices at least equal to the number "–" vertices in P. This results in a contradiction, because for CRD sets, the number of "+" vertices than is greater than the number of "–" vertices. □

Theorem 2.4.3 *There is no edge (u, v) in G_C, with $u \in R$ and $v \notin R$, i.e., there is no out-edge from R in G_C [27].*

Proof. Assume that an edge (u, v) in G_C, with $u \in R$ and $v \notin R$ exists. In step 19 of FindCRD algorithm, v is added to R, since v is reachable from u in G_C. This contradicts the assumption that v is not in R. □

Lemma 2.4.4 *No unmatched vertex except v_S exists in R' obtained in step 18 of FindCRD algorithm [27].*

Proof. Assume that there exists a path from v_S to an unmatched "–" vertex in $G^*_H(V, E_H)$. This path is an *augmenting path* of the matching. In step 13 of FindCRD algorithm, the *maximum matching* is obtained. Hence, this is a contradiction and there can not exist a path from v_S to an unmatched "–" vertex.

In $G^*_H(V, E_H)$, only the matched "+" vertices have in-edges, because the matched edges are reversed in step 13 of FindCRD algorithm. The remaining "+" vertices have only out-edges. Thus, no other unmatched "+" vertex is reachable from v_S in $G^*_H(V, E_H)$, i.e., no other unmatched "+" vertex is contained in R'. □

Lemma 2.4.5 *There are more "+" vertices in R' than there are "–" vertices [27].*

Proof. For every matched "+" vertex u in R', there is a corresponding matched "–" vertex v, because the only in-edge to u in $G^*_H(V, E_H)$ is (v, u), which is the reversed matched edge (u, v). Hence, the number of matched "+" and "–" vertices are equal. From Lemma 2.4.4, there is one unmatched "+" vertex in R'. Thus, the number of "+" vertices in R' is more than the number of "–" vertices in R'. □

Lemma 2.4.6 *Only "+" vertices are added to R' in step 19 of FindCRD algorithm [27].*

Proof. Assume that the "–" vertex $v \notin R'$ is added to R' in step 19 of FindCRD algorithm. That is, the "–" vertex v is reachable from a vertex in R' in G_C. For each "–" vertex w in R', there exists a path (u, w) in $G_H(V, E_H)$, such that u is a "+" vertex, since $G_H(V, E_H)$ consists of only "+" to "–" edges. That is every "–" vertex in R' is reachable from a "+" vertex in $G_C(V, E_C)$. Since v is reachable from a vertex in R' in $G_C(V, E_C)$, v is reachable from a "+" vertex in R' in $G_C(V, E_C)$. That is, there exists an edge (u, v) in $G_H(V, E_H)$ and v is in R'. This is a contradiction. □

Theorem 2.4.7 *There are more "+" vertices in R than there are "–" vertices [27].*

Proof. Directly from Lemmas 2.4.5 and 2.4.6, there are more "+" vertices in R' than there are "–" vertices and the vertices added to R' in step 19 of FindCRD algorithm are "+" vertices. Thus, the number of "+" vertices in R is more than the number of "–" vertices in R. □

Theorem 2.4.2 states that FindCRD algorithm detects the absence of cost reducible by decrease sets. That is, if FindCRD returns an empty set, it means that no CRD set exist in the constraint graph. Moreover, Theorems 2.4.3 and 2.4.7 show that a set R returned by the algorithm is a CRD a set, because no out-edge from R exists in G_C and the number of "+" vertices in R are larger than the number of "–" vertices in R. Hence, it is proven that FindCRD algorithm finds a cost reducible by decrease set, if exists [27].

As Lemma 2.4.1 suggests, given a constraint graph $G(V, E)$ with a clock schedule S; a set of registers R, is a CRI set if there is no critical in-edge to R in S and the number of registers in R with negative clock timing in S is larger than the number of registers in R with positive clock timing in S. Considering the critical graph $G_C(V, E_C)$ consisting of only the critical edges of $G(V, E)$, and labeling the registers with $s(v) > 0$ as "-" and the registers with $s(v) \geq 0$ as "+"; the definition can be re-stated as follows: Given a critical graph $G_C(V, E_C)$, a set of registers R, is a CRI set if there is no critical in-edge to R and the number of "–" registers in R is larger than the number of "+" registers in R.

The details of the algorithm to find a CRI set is not given in the literature, because it is very similar to FindCRD algorithm. Hence, the FindCRI algorithm is implemented based on the FindCRD algorithm description in [25] with the required modifications. The theoretical basis

of the algorithm is formed similar to that of the FindCRD and the correctness of the FindCRI algorithm is shown.

The algorithm to find a CRI set is given in Figure 2.16. The algorithm takes a constraint graph $G_T(V, E_T)$ for a clock period T and the current clock schedule S of the circuit as input. The output of the algorithm is the set R of cost reducible by increase vertices.If the algorithm returns an empty set, then there is no set of CRI registers. The register set R, returned by the algorithm has more "–" registers than "+" registers and there are no edges (u, v) in G_C with $u \notin R$ and $v \in R$. Firstly, the algorithm constructs the critical graph $G_C(V, E_C)$ consisting of only critical edges. Then, the transitive graph $G'_H(V, E'_H)$ is constructed using Warshall's Algorithm for transitive closure [59]. The vertices with timing less than zero ($s(v) < 0$) are labeled as "–" and the vertices with timing greater than or equal to zero ($s(v \geq 0)$) are labeled as "+". After that, the graph $G_H(V, E_H)$ is constructed from $G'_H(V, E'_H)$, by deleting all edges except those from "+" to "–" vertices. Hopcroft-Karp Algorithm is used to obtain a maximum matching of $G_H(V, E_H)$ [62], [63], [64] and the matched edges are reversed to obtain $G^*_H(V, E_H)$. At this point, the algorithm returns an empty set if all the "–" are matched, because no CRI set of vertices exist. Else, an unmatched "–" vertex, v_S, is selected and the set R' is constructed with all the vertices from which v_S is reachable from in $G^*_H(V, E_H)$. Finally, a vertex from which R' is reachable in $G_C(V, E_C)$ is added to R' and the resulting set R, is returned as output.

The following theorem is based on Theorem 2.4.2 and states that the FindCRI algorithm detects the absence of cost reducible by increase sets.

Theorem 2.4.8 *There is no set of CRI vertices in the graph, if all "-" vertices are matched in step 14 of FindCRI algorithm.*

Proof. Assume that all the "–" are matched and there exists a CRI set of vertices $P \subseteq V$. That is, there is no edge (u, v) in G_C with $u \notin P$ and $v \in P$ and there are more "–" vertices than "+" vertices in P. Since all "–" vertices are matched, there exists a "+" vertex u for every "–" vertex v in P and there is a path from u to v in G_C. Since P is a CRI set, there can not be any in-edges in the critical graph to vertices in P. Therefore hence vertex u is in P making the number of "+" vertices at least equal to the number of "–" vertices in P. This results in a contradiction, because for CRI sets, the number of "–" vertices is greater than the number of

Procedure FindCRI($G_T(V, E_T)$, S)
Input: Constraint graph $G_T(V, E_T)$, and current clock schedule $S = \{s(v) : \forall v \in V\}$
Output: Set R of cost reducible by increase vertices or empty set if no such vertex exist

1: $R \leftarrow \emptyset$
2: $R' \leftarrow \emptyset$
3: Construct critical graph $G_C(V, E_C)$ that consists of only critical edges
4: Construct $G'_H(V, E'_H)$, the transitive graph of $G_C(V, E_C)$ using Warshall's Algorithm
5: **for all** $v \in V$ **do**
6: **if** $s(v) < 0$ **then**
7: Label: $v \leftarrow$ "-"
8: **else if** $s(v) \geq 0$ **then**
9: Label: $v \leftarrow$ "+"
10: **end if**
11: **end for**
12: Construct $G_H(V, E_H)$ from $G'_H(V, E'_H)$ by deleting all edges in E_H except edges going from "+" to "-" vertices
13: Obtain a maximum matching of $G_H(V, E_H)$ by using Hopcroft-Karp Algorithm, and reverse the matched edges to obtain $G^*_H(V, E_H)$
14: **if** all "-" vertices matched **then**
15: **return** $R = \emptyset$
16: **end if**
17: Select an unmatched "-" vertex, v_S
18: $R' = \{v \in V : v_s$ is reachable from v in $G^*_H\}$
19: $R = R' \bigcup \{$a vertex v, from which R' is reachable in $G_C\}$
20: **return** R

Figure 2.16: Algorithm that finds cost reducible by increase sets

"+" vertices. □

The following theorem is similar to Theorem 2.4.3 and states that there is no critical in edge on the critical graph to the register set returned by the algorithm.

Theorem 2.4.9 *There is no edge (u, v) in G_C, with $u \notin R$ and $v \in R$, i.e., there is no in-edge to R in G_C.*

Proof. Assume that an edge (u, v) in G_C, with $u \notin R$ and $v \in R$ exists. In step 19 of FindCRI algorithm, u is added to R, since v is reachable from u in G_C. This contradicts the assumption that u is not in R. □

The Lemmas 2.4.10, 2.4.11, and 2.4.12 are based on Lemmas 2.4.4, 2.4.5, and 2.4.6, respectively.

Lemma 2.4.10 *No unmatched vertex except v_S exists in R' obtained in step 18 of FindCRI algorithm.*

Proof. Assume that there exists a path from an unmatched "+" vertex to v_S in $G^*_H(V, E_H)$. This path is an *augmenting path* of the matching. In step 13 of FindCRI algorithm, the *maximum matching* is obtained. Hence, this is a contradiction and there can not exist a path from an unmatched "+" vertex to v_S.

In $G^*_H(V, E_H)$, only the matched "–" vertices have out-edges, because the matched edges are reversed in step 13 of FindCRI algorithm. The remaining "–" vertices have only in-edges. Thus, v_S is reachable from no other unmatched "–" vertex in $G^*_H(V, E_H)$, i.e., no other unmatched "–" vertex is contained in R'. □

Lemma 2.4.11 *There are more "–" vertices in R' than there are "+" vertices.*

Proof. For every matched "–" vertex v in R', there is a corresponding matched "+" vertex u, because the only in-edge to u in $G^*_H(V, E_H)$ is (v, u), which is the reversed matched edge (u, v). Hence, the number of matched "–" and "+" vertices are equal. From Lemma 2.4.10, there is one unmatched "-" vertex in R'. Thus, the number of "–" vertices in R' is more than the number of "+" vertices in R'. □

Lemma 2.4.12 *Only "-" vertices are added to R' in step 19 of FindCRI algorithm.*

Proof. Assume that the "+" vertex $u \notin R'$ is added to R' in step 19 of FindCRI algorithm. That is, a vertex in R' is reachable from the "+" vertex u in G_C. For each "+" vertex w in R', there exists a path (w, v) in $G_H(V, E_H)$, such that v is a "–" vertex, since $G_H(V, E_H)$ consists of only "+" to "–" edges. That is, every "–" vertex in R' is reachable from a "+" vertex in $G_C(V, E_C)$. Since a vertex in R' is reachable from u in $G_C(V, E_C)$, a "–" vertex in R' is reachable from u in $G_C(V, E_C)$. That is, there exists an edge (u, v) in $G_H(V, E_H)$ and u is in R'. This is a contradiction. □

The following theorem based on Theorem 2.4.7 completes the proof of the correctness of FindCRI algorithm, by stating that the set returned by the FindCRI algorithm contains more '–" vertices than "+" vertices.

Theorem 2.4.13 *There are more "–" vertices in R than there are "+" vertices.*

Proof. Directly from Lemmas 2.4.11 and 2.4.12, there are more "–" vertices in R' than there are "+" vertices and the vertices added to R' in step 19 of FindCRI algorithm are "–" vertices. Thus, the number of "–" vertices in R is more than the number of "+" vertices in R. □

Theorem 2.4.8 states that FindCRI algorithm detects the absence of cost reducible by increase sets. That is, if FindCRI returns an empty set, it means that there is no CRI in the constraint graph. Moreover, Theorems 2.4.9 and 2.4.13 show that a set R returned by the algorithm is a CRI a set, because no in-edge to R exists in G_C and the number of "–" vertices in R are larger than the number of "+" vertices in R. Hence, it is proven that FindCRI algorithm finds a cost reducible by increase set, if exists.

The following theorem states that the minimum cost scheduling algorithm returns the minimum cost schedule to a given objective clock schedule.

Theorem 2.4.14 *MinCost algorithm outputs the minimum cost clock schedule [25], [27].*

Proof. Let s be the output clock schedule of MinCost algorithm. Assume that there exists another clock schedule s' with less cost than s, such that:

$$\sum_{v \in V} \left| s'(v) \right| < \sum_{v \in V} \left| s(v) \right|. \tag{2.52}$$

Let $\delta(v) = s(v) - s'(v)$, and let (u, v) be a critical edge in s. Since s is feasible and (u, v) is critical:

$$s(v) - s(u) = \omega(u, v). \tag{2.53}$$

Since s' is feasible:

$$s'(v) - s'(u) \leq \omega(u, v). \tag{2.54}$$

Hence,

$$s'(v) - s'(u) \leq s(v) - s(u), \tag{2.55}$$

and,

$$0 \leq (s(v) - s'(v)) - (s(u) - s'(u)). \tag{2.56}$$

Thus,

$$\delta(u) \leq \delta(v). \tag{2.57}$$

Let the set V be partitioned into three subsets: V_+, V_-, and V_0, such that:

$$V_+ = \{v \mid \delta(v) > 0\}, \tag{2.58}$$

$$V_- = \{v \mid \delta(v) < 0\}, \tag{2.59}$$

$$V_0 = \{v \mid \delta(v) = 0\}. \tag{2.60}$$

Further, let the set V_+ be partitioned into $V_1, V_2, \ldots, V_m$ as follows: for any vertex u and v in $V_i (1 \leq i \leq m)$, $\delta(u) = \delta(v) = \delta_i$ and for any vertex $u \in V_i$ and $v \in V_j$ $(1 \leq i < j \leq m)$, $\delta(u) > \delta(v)$.

Let,

$$f_i(x) = \sum_{v \in V_1 \cup V_2 \cup \ldots \cup V_i} \left| s(v) - x \right|. \tag{2.61}$$

Since clock schedule s is the output of MinCost algorithm, all the cost reducible sets are found and there is no cost reducible set for clock schedule s. Hence,

$$f_i(x) \geq f_i(x'), \quad \forall x > x' \geq 0. \tag{2.62}$$

Note the relation,

$$f_1(\delta_1) = \sum_{v \in V_1} \left| s(v) - \delta_1 \right| \tag{2.63}$$

$$f_2(\delta_2) = \sum_{v \in V_1} \left| s(v) - \delta_2 \right| + \sum_{v \in V_2} \left| s(v) - \delta_2 \right| \tag{2.64}$$

$$= f_1(\delta_2) + \sum_{v \in V_2} \left| s(v) - \delta_2 \right| \tag{2.65}$$

$$\vdots$$

$$f_m(\delta_m) = f_{m-1}(\delta_m) + \sum_{v \in V_m} \left| s(v) - \delta_m \right| \tag{2.66}$$

Hence,

$$\sum_{v \in V_k} \left| s(v) - \delta_k \right| = f_k(\delta_k) - f_{k-1}(\delta_k), \quad \forall k,\ 0 < k \leq m. \tag{2.67}$$

Procedure ClockSchedule(N, T, O)
Input: Circuit netlist N, target clock period T, target clock schedule O
Output: Clock period T, clock schedule $S = \{ s(v) : \forall v \in V\}$,
clock timing range $R = \{ (s_{min}(v), s_{max}(v)) : s_{min}(v) \leq s(v) \leq s_{max}(v), \forall v \in V\}$

1: Construct the circuit graph $G_{cir}(V_{cir}, E_{cir})$ from the netlist N
2: Determine the capacitive load of each gate and calculate the propagation delay of each gate using the model
3: Construct the constraint graph $G(V, E)$ from the the circuit graph $G_{cir}(V_{cir}, E_{cir})$ and the estimated propagation delay of gates
4: **if** constraints are infeasible **then**
5: **return Infeasible**
6: **end if**
7: **if** target clock period is not given **then**
8: T := **MinClock(** G **)**
9: **end if**
10: S := **MinCost(** G_T**,** O **)**
11: R := **MaxRange(** G_T**,** S **)**
12: **return** T, S and R

Figure 2.17: Clock scheduling system.

Note that, there is no critical out-edge from $\{V_1, V_2, \ldots, V_i\}$, and $f_i(x)$ is a convex function. Then,

$$\begin{aligned}
\sum_{v \in V_+} \left|s'(v)\right| &= \sum_{v \in V_+} \left|s(v) - \delta(v)\right| \\
&= \sum_{v \in V_1} \left|s(v) - \delta_1\right| + \sum_{v \in V_2} \left|s(v) - \delta_2\right| + \cdots + \sum_{v \in V_m} \left|s(v) - \delta_m\right| \\
&= f_1(\delta_1) + f_2(\delta_2) - f_1(\delta_2) + f_3(\delta_3) - f_2(\delta_3) + \cdots + f_m(\delta_m) - f_{m-1}(\delta_m) \\
&\geq f_m(\delta_0) = \sum_{v \in V_+} \left|s(v)\right|
\end{aligned} \tag{2.68}$$

Note that since $\delta_{k-1} > \delta_k$, $f_{k-1}(\delta_{k-1}) \geq f_{k-1}(\delta_k)$ for all $0 < k \leq m$.

Similarly,

$$\sum_{v \in V_-} \left|s'(v)\right| \geq \sum_{v \in V_-} \left|s(v)\right|. \tag{2.69}$$

Therefore,

$$\sum_{v \in V} \left|s'(v)\right| \geq \sum_{v \in V} \left|s(v)\right|, \tag{2.70}$$

which is a contradiction to the initial assumption. □

Procedure MaxRange($G_T(V, E_T)$, S)
Input: Constraint graph $G_T(V, E_T)$, feasible clock schedule S
Output: Consistent set of clock ranges of all registers:
$R = \{ (s_{min}(v), s_{max}(v)) : s_{min}(v) \leq s(v) \leq s_{max}(v), \forall v \in V\}$

1: **for all** $v \in V$ **do**
2: $r(v) := \left[s(v) - \frac{1}{2} \min_{(v,u)\in E_T} \Delta_T(v, u) , \quad s(v) + \frac{1}{2} \min_{(u,v)\in E_T} \Delta_T(u, v) \right]$
3: **end for**
4: **return** R

Figure 2.18: Algorithm that determines the clock schedule ranges of all registers.

2.5 Clock Scheduling System

The algorithms discussed in the previous sections of this chapter are combined into a *clock scheduling system* with a number of additional algorithms, such as; the algorithm to parse the circuit netlists and generate the graph representation of the circuit, the algorithm to estimate the propagation delays of the gates for the actual load capacitances in the circuit, and the algorithm to construct the constraint graph of the circuit.

The clock scheduling system takes a circuit in netlist format as an input and constructs the circuit graph $G_{cir}(V_{cir}, E_{cir})$, where each vertex represents a circuit element (an input, an output, a gate or a register) and each edge represents a connection between the elements. The setup and hold time information of the registers are supplied as an input to the system. The actual capacitive load of each gate in the circuit is determined using the tabulated input capacitance values of each gate type. Then the propagation delay of each gate is calculated using the model relating the propagation delay to the load capacitance. All the paths between every register pair with signal propagation is determined using a recursive function. Then, the minimum and maximum propagation delays are calculated for every register pair with signal propagation. This information is used to construct the constraint graph $G(V, E)$, of the circuit. Other inputs of the clock scheduling system are the target clock period T, and target clock schedule $O(v) \quad \forall v \in V$. If the target clock period is not supplied as an input, the minimum feasible clock period calculated using MinClock algorithm is used as the target clock period. Figure 2.17 shows the pseudo-code of the clock scheduling system. After the minimum cost scheduling is done, the system widens the clock timing range of every register by using the MaxRange algorithm [56]. The pseudo-code of MaxRange algorithm is given in Fig. 2.18.

The output of the clock scheduling system is a consistent set of clock timing ranges for all the registers.

CHAPTER 3

PEAK POWER MINIMIZATION

In the complete synchronous framework all the registers in a synchronous circuit are clocked at the same time. Hence, the switching of all the registers and the gates at the output cones of the registers take place in a short time period right after the clock edge. This results in a narrow pulse with a high peak value in the supply current of the circuit. The current pulse takes place in relatively small portion of the clock period, whereas for the rest of the clock period the supply current is at a lower value corresponding to the static power dissipation of the elements in the circuit. Hence, a power supply with a high current output capability is needed, although the peak value of the current is used for a relatively short time.

Peak power dissipation of a synchronous sequential circuit can be reduced in general synchronous framework, where synchronous clocking requirement of all the registers is released. The different clock timings for the registers mean that the switchings occur at different times in one clock period. Therefore, the supply current wave of the circuit is suppressed and broadened.

In the first section of this chapter, the estimation of the peak power consumption of a synchronous sequential circuit is discussed. The peak power consumption of the circuit is expressed as the sum of register originated power dissipations. The calculation of the switching probabilities is explained. A new method for estimating the switching power of circuit elements for different load capacitances is proposed. The first section ends with the explanation of the peak power estimation algorithm.

In the second section, the clock scheduling algorithm for peak power minimization is discussed. The algorithm is a two stage algorithm in which the result of an initial scheduling obtained from the first stage is greedily modified by the second stage until the objective of

peak power dissipation is minimized.

3.1 Estimation Of Peak Power Consumption

In this section, a fast power consumption estimation method for sequential circuits similar to that in [46] is discussed. The power consumption of a circuit is modeled as the sum the of power consumptions of gates and registers that switch. A *switching event* starts at a register when a clock edge arrives at the register and the input of the register is changed relative to the previous state. The time at which the clock edge arrives at a register is determined by the clock schedule, so as the emerging time of a switching event. The switching event propagates through the combinational circuit at the output of the register, if not blocked at a gate due to the other inputs of that gate. Thus, a combinational gate, to which a switching event reaches, switches with some probability [46]. The arrival time of a switching event at a combinational gate depends on the clock schedule, delays of previous gates and routing delays between the gates. The power consumption of a circuit can be expressed as the sum of the register originated power consumptions, due to the fact that each switching event emerges at a register [46].

Obviously, the actual power consumption of a circuit depends on the input vectors applied to the circuit, and the power wave of the circuit changes at each clock period. In order to save computation time in expense of accuracy, a probability based approach is used in peak power estimation as in [46]. The clock period is divided into relatively small time units and the peak power consumption of the circuit is calculated at each time unit in a discrete fashion.

After the switching event reaches an element in the circuit, the power consumption of that element continues for some time depending on the slew rate of the switching and the number of fan-outs of the element [46]. In the estimation, the simulated power wave of the elements are used. In [46], each gate is simulated by connecting n NAND gates to the output, where n is the number of fan-outs of the gate, and the resulting power wave is used in the peak power estimation. However, the power wave depends strongly on the load that is driven by the gate, i.e., the fan-out of the gate. In order to increase the accuracy, a new approach exploiting the dependence of the duration and peak value of the power wave on load capacitance is utilized in this work. In this method, the power consumption of a gate is obtained initially by simulating

the gate with a unit capacitive load. Then, the actual load of the gate in the circuit is calculated according to the number and type of the fan-outs of the gate in the circuit. Then, the power wave of the gate for the actual load capacitance in the circuit is estimated by reshaping the power wave for the unit load using the dependence of the shape of the power wave to the load capacitance.

The *register originated power consumption* is defined as the power consumption caused by a switching event that is emerged at a register. The register originated power consumption of a register includes the power consumption of the register and the sum of power consumptions of the gates at the output cone of the register that switch with a probability.

The actual register originated power consumption wave, depends on both the clock schedule and the input vectors applied to the circuit. The propagation of a switching event which is not blocked at a gate in a clock schedule may be blocked in another clock schedule because the register outputs may change according to the clock schedule. Moreover, a switching event which is not blocked at a gate for an input vector may be blocked for another input vector. However, in the estimation method of [46], the register originated power consumption is assumed to be independent of both the clock schedule and the input vectors, and estimated probabilistically. In this way, the computation time can substantially be reduced.

3.1.1 Calculation of Switching Probabilities

The switching probability calculation method proposed in [46], is used in the peak power estimation algorithm in this work. The *condition probability*, $c_a(v)$ of a gate v is defined as the probability of the output of gate v to be a [46]. Table 3.1 shows the condition probability of gates with respect to the number of inputs n. Note that, the condition probability of NOT gate (inverter) is not given in Table 3.1 since it only depends on the input of the NOT gate:

$$c_0(v) = c_1(v'), \tag{3.1}$$

$$c_1(v) = c_0(v'), \tag{3.2}$$

where v' is the fan-in gate of the NOT gate v.

The *switching probability*, $p(r, v, t)$, of an element v, is defined as the probability that the output of the element v, changes at time t, due to the switching event emerged at register r

Table 3.1: The condition probabilities of n input gates to be a

a	$c_a(NAND)$	$c_a(NOR)$	$c_a(AND)$	$c_a(OR)$
0	$\frac{1}{2^n}$	$\frac{2^n-1}{2^n}$	$\frac{2^n-1}{2^n}$	$\frac{1}{2^n}$
1	$\frac{2^n-1}{2^n}$	$\frac{1}{2^n}$	$\frac{1}{2^n}$	$\frac{2^n-1}{2^n}$

at time $t = 0$ [46]. Actual switching probability of an element depends on both the clock scheduling and the input vector supplied to the circuit.

In order for a gate output to switch, some inputs of the gate should switch while the other inputs do not prevent the switching. The switching of a NOT gate occurs when its input switches, since it has no other input to prevent the switching. For the NAND and AND gates, the switching of the output occurs, when a group of inputs switch in the same direction and the rest of the inputs are fixed at logical "1". For the NOR and OR gates, the switching of the output occurs when a group of inputs switch in the same direction and the rest of the inputs are fixed at logical "0". Note that, the case of inputs simultaneously switching in opposite directions that results in glitches at the output are not considered in this analysis.

Let $I(v)$ be a set of fan-in gates of gate v, X be a subset of fan-in gates of v and $d(v)$ be the delay of v. The switching probability of the NAND and AND gates is given by [**?**]:

$$\begin{aligned} p(r,v,t) = \sum_{X \subseteq I(v)} \Bigg\{ & \prod_{v' \in I(v) \setminus X} c_1(v') \cdot \Big(1 - p(r,v',t-d(v))\Big) \\ & \cdot \Big(\prod_{v' \in X} c_1(v') + \prod_{v' \in X} c_0(v') \Big) \cdot \prod_{v' \in X} \Big(1 - p(r,v',t-d(v))\Big) \Bigg\}. \end{aligned} \tag{3.3}$$

Similarly, the switching probability of the NOR and OR gates is given by:

$$\begin{aligned} p(r,v,t) = \sum_{X \subseteq I(v)} \Bigg\{ & \prod_{v' \in I(v) \setminus X} c_0(v') \cdot \Big(1 - p(r,v',t-d(v))\Big) \\ & \cdot \Big(\prod_{v' \in X} c_1(v') + \prod_{v' \in X} c_0(v') \Big) \cdot \prod_{v' \in X} \Big(1 - p(r,v',t-d(v))\Big) \Bigg\}. \end{aligned} \tag{3.4}$$

Finally, the switching probability of the NOT gate is given by:

$$p(r,v,t) = p(r,v',t-d(v)). \tag{3.5}$$

The output of a gate may switch more than once in a clock period, if some inputs switch more than once with others not blocking the switching. For the fan-in gate of a register, the gate

output just before the arrival time of the clock edge to the register is important, i.e., whether the output of the gate just before the arrival time of the next clock edge to the register is different from the output of the gate just before the arrival time of the previous clock period. In other words, if the fan-in gate of a register switches odd number of times within one clock period, the register switches. The clock edge is divided into n unit time intervals for the probability calculation. Let $p_1,\ p_2, \ldots,\ p_n$ be the switching probabilities of the fan-in gate of register r in one clock period. Then the switching probability of r is given by [46]:

$$\begin{aligned} p(r,r,0) &= p_1(1-p_2)(1-p_3)\cdots(1-p_n) \\ &+ (1-p_1)p_2(1-p_3)\cdots(1-p_n) \\ &\vdots \\ &+ (1-p_1)(1-p_3)\cdots(1-p_{n-1})p_n \\ &+ p_1p_2p_3(1-p_4)(1-p_5)\cdots(1-p_n) \\ &+ p_1p_2(1-p_3)p_4(1-p_5)\cdots(1-p_n) \\ &\vdots \\ &= \Big(1-(1-2p_1)(1-2p_2)\cdots(1-2p_n)\Big)\Big/2 \end{aligned} \tag{3.6}$$

The switching probability of a register depends on the switching probability of the fan-in gate of the register and the switching probability of a gate depends on the switching probabilities of registers [46]. Hence, the switching probability calculations are done iteratively until the probabilities converge. Note that, initially, the switching probability of all the registers and inputs are set to 1, so that maximum number of switchings occur and the peak power can be estimated more accurately.

3.1.2 Calculation of Peak Power Consumption

The probabilistic peak power estimation method proposed in [46] is used in this work. In this method, the peak power of the circuit is estimated probabilistically by assuming that the power consumption is independent from the input vectors applied to the circuit. The power consumption of a circuit can be stated as the sum of register originated power consumptions. Hence, the power consumption of a circuit, $W(t)$, is given by:

$$W(t) = \sum_{r \in R} W(r,t), \tag{3.7}$$

where $W(r,t)$ is the register originated power consumption of register $r \in R$ at time t due to the clock inputted to r at time $t = 0$. The power consumption of a circuit depends on the clock schedule S of the circuit. The register originated power consumption of each register is shifted in time according to the clock timing of the register. Let $W^S(r,t)$ be the register originated power consumption of register $r \in R$ at time t due to the clock inputted to r at time $t = s(r)$, where $s(r)$ is the clock timing of register r. $W^S(r,t)$ is given by:

$$W^S(r,t) = W(r, t - s(r)). \tag{3.8}$$

Thus, power consumption of a circuit for a clock schedule S, $W^S(t)$, is given by:

$$W^S(t) = \sum_{r \in R} W^S(r,t). \tag{3.9}$$

The switching of a register r due to a clock input, starts a series of switching events at the fan-out gates of the register. The switching gates trigger the switching of their fan-out gates. The propagation of this switching wave continues until a gate driving a register is switched, if not blocked by side inputs of the gates. The largest set of gates that can switch due to a switching event emerged at a register r is called *output cone* of the register r and is denoted by $O(r)$. Note that in finding the gates at the output cone, all the side inputs of the gates are assumed to be suitable for gate switching. In other words, the output cone of a register is the set of gates that are on any path from the output of a register to the input of another register.

The register originated power consumption of a register r, consists of the static power consumption of the register r and the dynamic power consumption of register r and the gates at the output cone of r. The static power consumption of the register, is the power consumed by the register at each clock arrival that is independent of the switching event, i.e., static power is consumed whether the register switches or not. The static power consumption of the register r, at time t for a clock inputted to the register at time $t = 0$ is denoted by $W_c(r,t)$. The dynamic power consumption of the register and the gates at the output cone of the register is the sum of the switching powers of the register and the gates. The dynamic power consumption of the register r, at time t for a clock inputted to the register at time $t = 0$ is denoted by $W_g(r,t)$. Thus, the register originated power consumption of register r is given by:

$$W(r,t) = W_c(r,t) + W_g(r,t). \tag{3.10}$$

The dynamic power consumption of the register r and the gates at the output cone $O(r)$ of

register r is estimated by:

$$W_g(r,t) = \sum_{v \in O(r)} \sum_{0 \le i < T} p(r,v,t-i) \cdot w_g(v,i), \tag{3.11}$$

where $w_g(v,i)$ is the switching power consumption of a gate at time t, where the switching of v occurs at $t = 0$; $p(r,v,t)$ is the switching probability of gate v, due to the switching of register r at time $t = 0$; and T is the clock period [46].

3.1.3 Switching Power Estimation

The switching power waves of gates and registers should be estimated fast and accurately for the register originated power calculation. There has been a number of approaches for the estimation of power consumption of gates and registers [34], [43], [44], [46]-[54]. The switching power wave of a circuit element depends strongly on the slew of the input signal and the output load. In [52], the gates are characterized by simulations for a range of fan-in and fan-out conditions. The characterization results are tabulated in look-up tables to be later used in power estimation of circuits. Since there is a trade-off between the size of the look-up table and the level of discretion, the probability of the actual fan-in and fan-out to be in the look-up table is very small. Hence, the accuracy of this approach is impaired.

In [34], [43] and [44], the switching power wave of gates and registers are assumed to be triangular. In [34], the sequential elements and combinational parts of the circuits are simulated separately using the event driven simulator described in [53] and [54], and the simulation results are used in peak power estimation. However, since the supply currents are approximated by triangular waves, the estimation is not accurate at all. Moreover, the number of simulations to be done for a circuit increases with the size of the circuit.

In [46], the power waves of the circuit elements are obtained empirically. An element with n fan-outs is simulated with n NAND gates as output load and the power wave data is used by the power estimation algorithm. This approach has two flaws. Firstly, the input capacitances of circuit elements vary from type to type; thus, assuming that all the fan-outs are NAND gates, leads to an error in output load calculation. Secondly, acquiring all the power waves empirically reduces the practicality of the method since all the elements need to be simulated for the actual number of fan-outs in the circuits. This results in a large number of simulations to be performed and large amount of data to be stored.

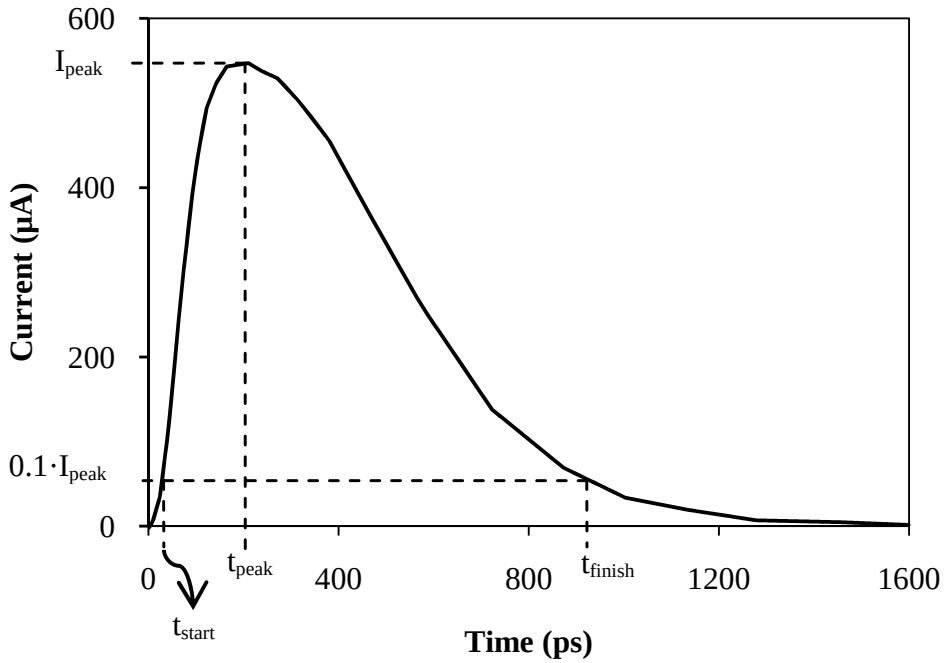

Figure 3.1: Switching current wave of a 2-input AND gate for output transition from low to high. The parameters for characterizing the current waveform are also shown.

In this study, a new method, namely the Shaped Pulse Approximation Method (SPA), is proposed for estimating the supply current waves of circuit elements. This method approaches the triangular approximation from a new direction. The triangular approximation is exploited to characterize the relation between the supply current waves for different loads, rather than estimating the current wave itself. The gates and registers are simulated with a range of capacitive loads and the variation of the triangular approximation parameters with respect to the load capacitance is modeled. The actual load capacitance of the circuit elements are extracted from the circuit graph using the tabulated input capacitance values for all types of gates and registers. Then, the current wave for an actual load capacitance is constructed from the current wave for a unit load using the modeled relation between the triangular parameters of the waves. The supply current waves for an arbitrary load can be approximated accurately by storing only one tabulated wave data for each element type.

Consider the switching current wave of a 2-input AND gate given in Fig. 3.1 for output transition from low to high. The current waveform can be characterized by four parameters. Let I_{peak} denote the peak value of the current and t_{peak} denote the time at which the current is at its peak value. Let t_{start} and t_{finish} be the times at which the current is at 10% of its peak value before and after the peak value is reached, respectively.

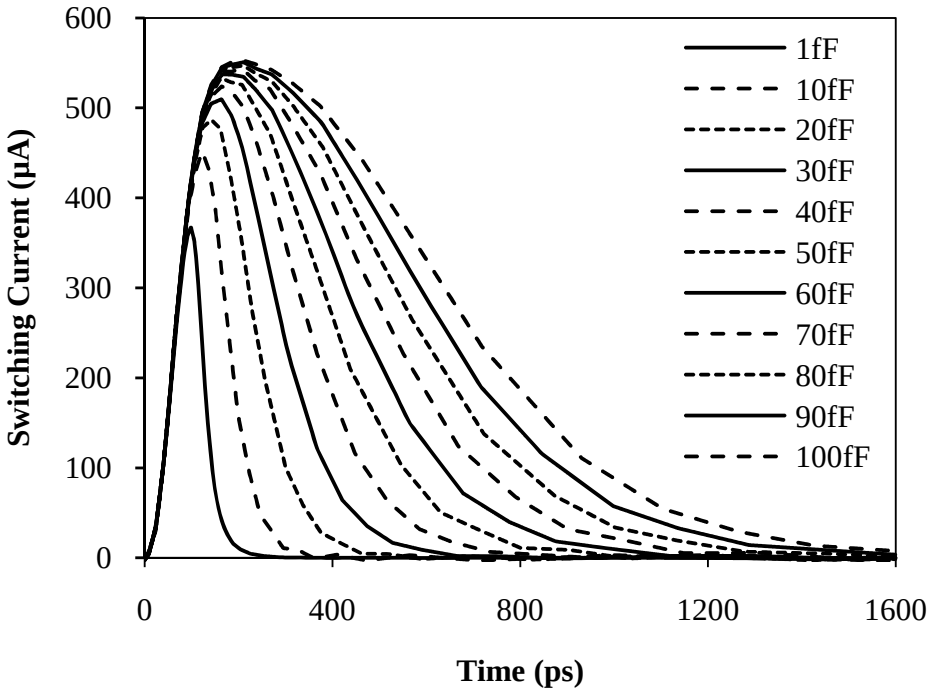

Figure 3.2: The switching current waves of a 2-input AND gate for output switching from low to high. The waves are plotted for various capacitive loads ranging from 1 fF to 100 fF.

Define three additional parameters, namely rise time (t_r), fall time (t_f) and pulse width (t_w) as follows:

$$t_r = t_{peak} - t_{start}, \tag{3.12}$$

$$t_f = t_{finish} - t_{peak}, \tag{3.13}$$

$$t_w = t_{finish} - t_{start}. \tag{3.14}$$

The switching current of a gate depends on the number and type of the fan-outs of the gate, i.e., the value of the capacitive load at the output of the gate. In order to illustrate this, consider the switching current waves of a 2-input AND gate for varying load capacitances as shown in Fig. 3.2. Note that, the portion of the switching current corresponding to the output transition from high to low do not change significantly with the value of the load capacitance. Hence, only the portion of the switching current corresponding to the output transition from low to high is considered as the switching current in this work.

The peak value of the switching current and the width of the current pulse increase as the value of the load capacitance increases. Fig. 3.3 shows the variation of the peak value of the

switching current of the 2-input AND gate with respect to load capacitance. The peak current can be fitted by a logarithmic curve but a polynomial curve results in less error. Fig. 3.4 shows the variation of the switching current pulse width of the 2-input AND gate with respect to the load capacitance. The pulse width is a linear function of the load capacitance. However, since the switching current pulse is not symmetric with respect to t_{peak}, it is necessary to treat rise and fall times separately. Otherwise, the shape of the pulse is distorted, which increases the total error in the estimation. Fig. 3.5 shows the variation of the rise and fall time of the 2-input AND gate with respect to the load capacitance. The rise time variation is best fitted by a polynomial function, whereas the fall time variation is best fitted by a linear function. These observations are the motivation behind the current estimation method discussed in this work: if the variations of the peak current, rise time and fall time with respect to the load capacitance can be determined, the switching current wave for an arbitrary load can be estimated by using the switching current wave for a unit load. The switching current wave for a unit load can be processed to estimate the switching current wave of any load capacitance. This is a three step process. In the first step, the magnitude of each point in the unit wave is multiplied by a magnitude correction factor. In the second step, the rising portion of the unit wave, i.e., the portion of the wave from t_{start} to t_{peak}, is stretched in time by a time correction factor. Finally, in the third step the falling portion of the unit wave, i.e., the portion of the wave from t_{peak} on is stretched in time by another time correction factor.

In order to formulate the process, let $w(v, t, C)$ be the switching power wave of element v for load capacitance C at time t. Then, $w(v, t, C)$ can be stated as:

$$w(v,t,C) = \begin{cases} w_1(v,t,C), & \text{if } t_{start} \leq t \leq t_{peak} \\ w_2(v,t,C), & \text{if } t > t_{peak}, \end{cases} \tag{3.15}$$

where $w_1(v, t, C)$ is the first part (rising part) of the switching power wave and $w_2(v, t, C)$ is the second part (falling part) of the switching power wave. The two parts of the switching power wave of a circuit element for an arbitrary load can be represented in terms of the switching power wave of the element with unit load capacitance $w(v, t, C_{unit})$ as:

$$w_1(v,t,C) = M(C) \cdot w_1(v, K_1(C) \cdot t, C_{unit}), \tag{3.16}$$

$$w_2(v,t,C) = M(C) \cdot w_2(v, K_2(C) \cdot t, C_{unit}), \tag{3.17}$$

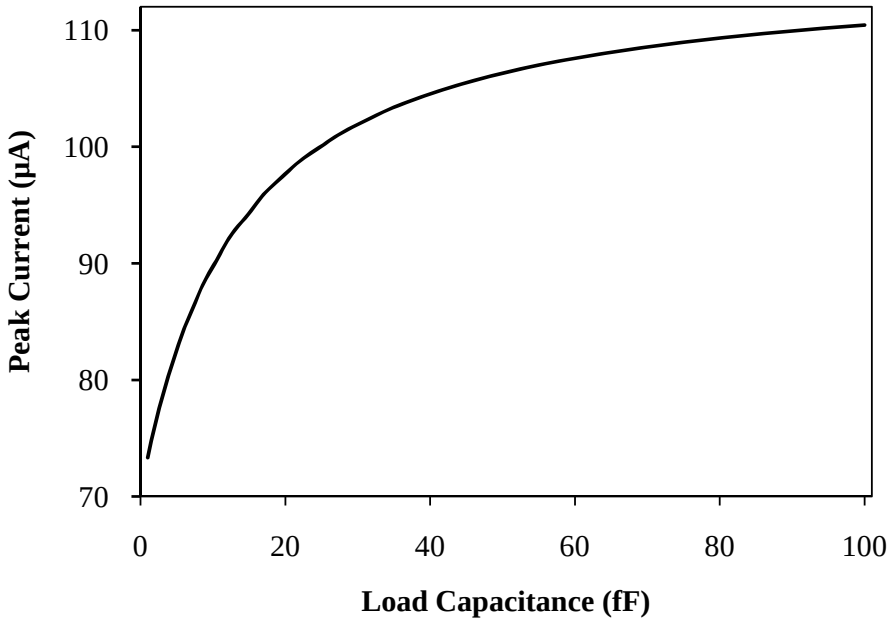

Figure 3.3: The variation of the peak value of the switching current of 2-input AND gate with respect to load capacitance. The curve can be fitted with a logarithmic function but a 6th order polynomial gives a better fitting with less error.

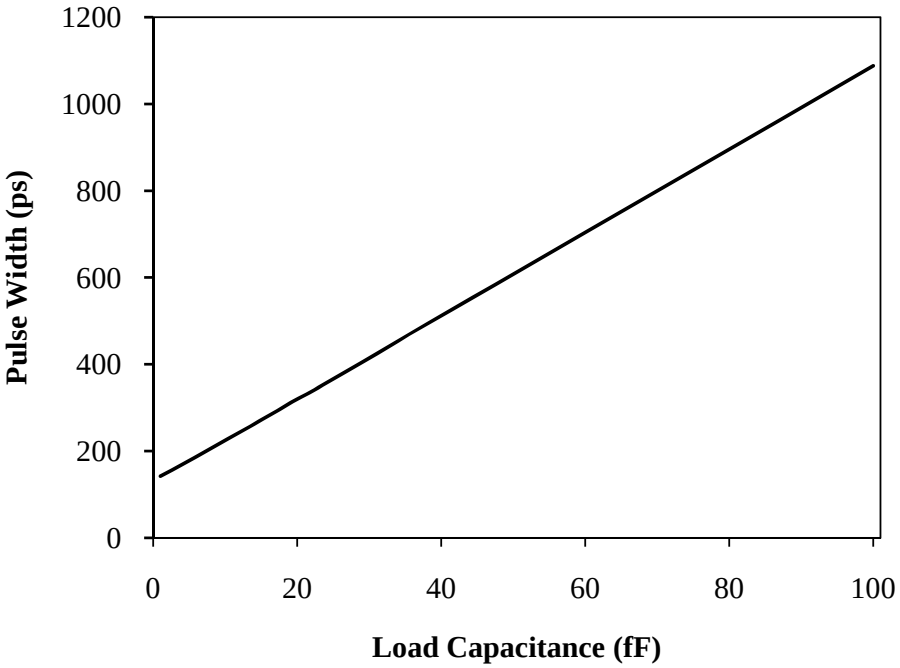

Figure 3.4: The variation of the switching current pulse width of the 2-input AND gate with respect to load capacitance. The pulse width is measured from t_{start} to t_{stop}.

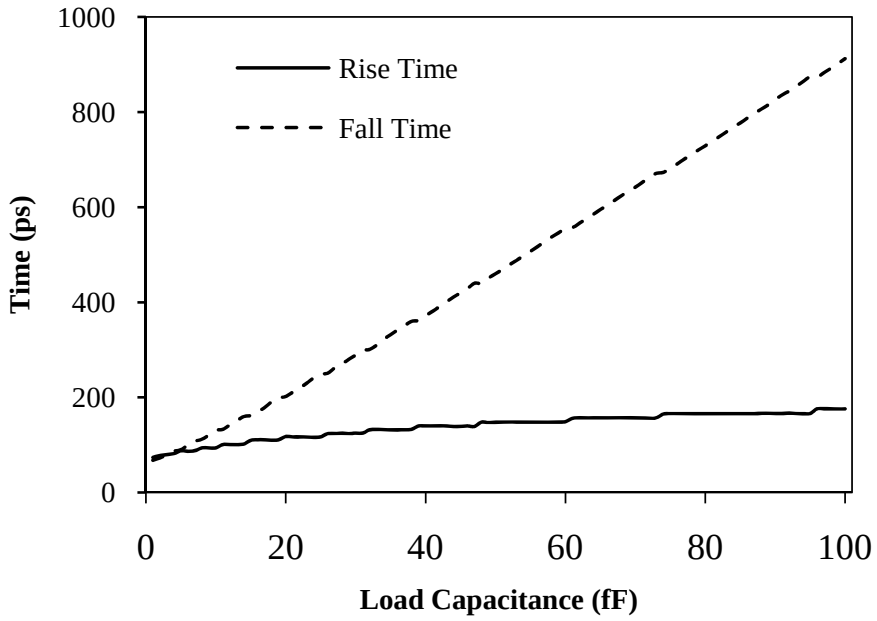

Figure 3.5: The variation of rise and fall times of the switching power wave of 2-input AND gate with respect to load capacitance. Rise time curve can be fitted with a 3rd order polynomial function. Fall time curve can be fitted with a linear function.

where $M(C)$ is the magnitude correction factor defined as:

$$M(C) = \frac{I_{peak}(C)}{I_{peak}(C_{unit})}, \tag{3.18}$$

and $K_1(C)$ and $K_2(C)$ are the time correction factors for the first and second parts of the switching power pulse respectively. $K_1(C)$ and $K_2(C)$ are defined as:

$$K_1(C) = \frac{t_r(C)}{t_r(C_{unit})}, \tag{3.19}$$

$$K_2(C) = \frac{t_f(C)}{t_f(C_{unit})}. \tag{3.20}$$

In order to evaluate the performance of the estimation method, the following normalized rms error function, E, is defined:

$$E = \frac{\sqrt{\sum_{0 \le t < T} (w_{est} - w_{org})^2}}{\sqrt{\sum_{0 \le t < T} (w_{org})^2}}, \tag{3.21}$$

where w_{est} is the estimated value of switching power wave at time t, and w_{org} is the value of the actual switching power wave at time t.

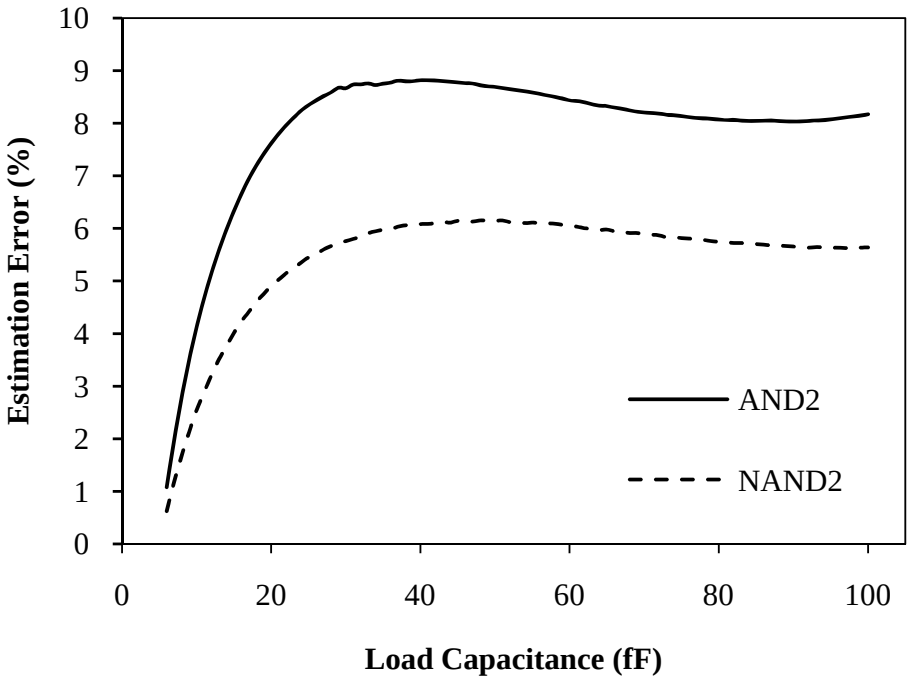

Figure 3.6: Estimation error for 2-input AND and 2-input NAND gates. The switching current wave for 5 fF capacitance is used as the unit wave, and the switching current waves for capacitive loads in 6 fF - 100 fF range are estimated using the method. The estimated waves are compared with simulation results and the estimation error is calculated using (3.21).

The estimation error for 2-input AND and 2-input NAND gates with respect to load capacitance are given in Fig. 3.6. Table 3.2 summarizes the maximum error for each type of gate. The rms error calculations are made for 1600 ps period. The switching current wave for 5 fF load is used as unit wave and the switching current waves for load capacitances in 6 fF - 100 fF range are estimated. The switching current waves of gates can be estimated with rms error less than 10% for 1600 ps period. The details of the simulations are given in Chapter 4. Figure 3.7 shows the switching current wave of a 2-input NAND gate for 100 fF load and the estimated switching current wave. The estimation is done by using the tabulated simulation data of the 2-input NAND gate for 5 fF load. The normalized rms error is 5.7% for this case.

3.1.4 Power Estimation Algorithm

The methods discussed in the previous subsections of this section are combined in a *power estimation algorithm* which estimates the register oriented power consumptions of the registers of a sequential circuit. The algorithm consists of two parts. In the first part, the switching probabilities of the gates and registers of the circuit are calculated. In the second part, the reg-

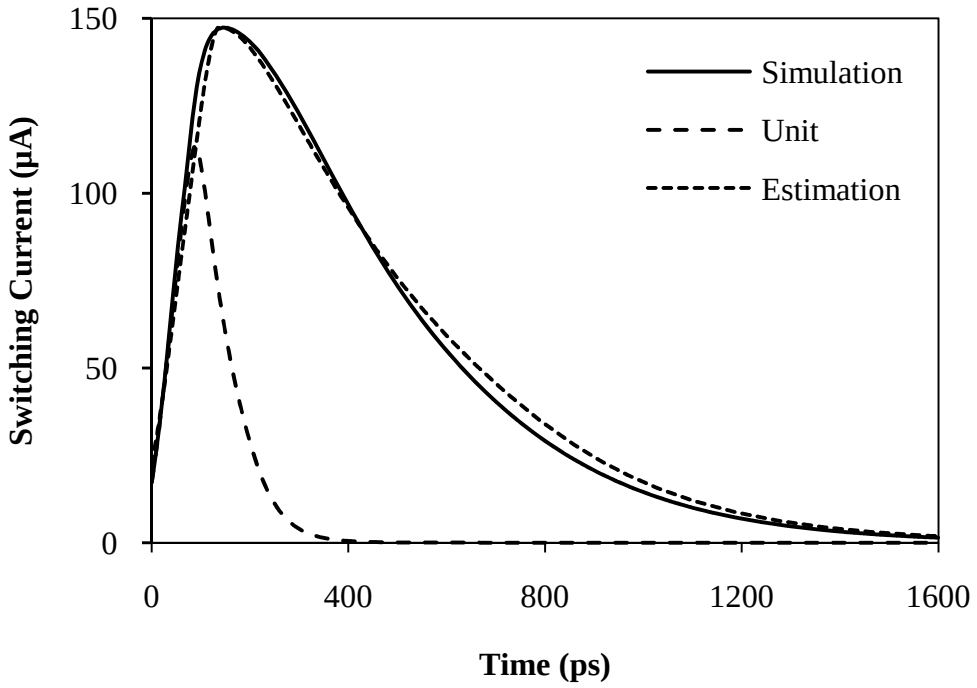

Figure 3.7: The switching current wave of 2-input NAND gate for 5 fF and 100 fF load capacitances shown with the estimated current wave for 100 fF load using the data for 5 fF load. The normalized rms error is 5.7%.

Table 3.2: The Normalized Rms Estimation Error For Gates

Gate Type	Maximum Estimation Error (%)
INV	8.09
AND2	8.82
AND3	7.44
AND4	6.54
NAND2	6.15
NAND3	8.55
NAND4	6.14
NOR2	9.12
NOR3	9.39
NOR4	8.56
OR2	9.18
OR3	9.85
OR4	9.91

ister originated power consumption is estimated for each register using the probability data from the first part. The inputs of the power estimation algorithm are the netlist of the circuit, the clock period, the propagation delays of gates and the switching power consumption data of the gates and registers. The output of the algorithm is the register originated power consumptions of the registers in the circuit. In addition to that, the algorithm computes the peak power consumption of the circuit for one clock period with zero clock timings for all registers, i.e., with fully synchronous clock. This data is used for comparison with the output of the peak power minimization algorithm. Figure 3.8 shows the power estimation algorithm.

The first part of the algorithm starts by parsing the netlist of the circuit, in order to construct a circuit graph $G_{cir}(V_{cir}, E_{cir})$. In the circuit graph, each vertex represents an element (an input, an output, a gate or a register) in the circuit and each edge represents a connection between the elements. The output cones are determined for each register and the set of fan-in gates are determined for each gate. Then, the iterative switching probability calculation starts by assuming that every input and every register of the circuit switches at time $t = 0$. At each iteration, the switching probabilities of the gates are calculated first, using the switching probability values of registers from the previous iteration. When the switching probabilities are calculated for all gates, the switching probabilities of the registers are updated for that iteration. The algorithm compares the results of the present iteration with the previous iteration and decides to stop if the differences are smaller than ϵ for every register, i.e., the probability calculation converges. ϵ is chosen as 0.001 for experiments. The calculation of the switching probability of a gate requires the determination of all the subsets of the set of fan-in gates for that gate. This is done efficiently using the lexicographic ordering [65] since the number of inputs for the gates are limited to four.

The second part of the algorithm estimates the capacitive load of each element according to the number and type of the fan-outs gates of each element. This information is used in estimating the switching power wave of each gate utilizing the SPA method discussed in subsection 3.1.3. Then, the register originated power consumption of each register is calculated using the switching probability and the power wave of each gate. The peak power consumption of the circuit for fully synchronous clock is calculated. Finally, the algorithm terminates by writing the peak power data for fully synchronous clock to a file.

Procedure EstimatePower(N, T, D_p, $w_g(r,t)$)
Input: Circuit netlist N, clock period T, the propagation delay of gates, $D_p(v)\ \ \forall v \in V_g$, the switching power consumption of gates and registers $w_g(v,t)\ \ \forall v \in V$
Output: The register originated power consumptions $W(r,t)\ \ \forall r \in V_r$

1: Construct the circuit graph $G_{cir}(V_{cir}, E_{cir})$ from the netlist N
2: Determine output cones $O(r)$, of each register $r \in V_r$
3: Determine the set of fan-in gates $I(v)$, of each gate $v \in V_g$
4: **repeat**
5: Calculate $p(r,v,t)$ for all gates using (3.3), (3.4) and (3.5)
6: Calculate $p(r,r,0)$ for all registers using (3.6)
7: **until** the switching probabilities of all the registers converge
8: Estimate the capacitive load and switching power of all the elements
9: Calculate the register originated power consumption of the registers using (3.11)
10: Output the peak power consumption of the circuit without clock scheduling to a file
11: **return** $W(r,t)\ \ \forall r \in V_r$

Figure 3.8: The algorithm to estimate the register originated power consumptions of the registers of a circuit.

3.2 Clock Scheduling For Peak Power Consumption Reduction

The clock scheduling algorithm used for peak power minimization in this work is a two-stage algorithm based on those proposed in [44] and [46]. This method is selected due to its speed and ease of implementation among several other approaches such as genetic algorithm [34], [43] and linear programming [45]. The SPA method is used as the switching power estimation method in calculation of the register originated power consumptions that are supplied as inputs to the peak power minimization algorithm.

The first stage of the algorithm shown in Fig. 3.9 starts with an initial feasible clock schedule obtained from the ClockSchedule Algorithm shown in Fig. 2.17. Then, starting with the register with the largest peak register originated power consumption, all the registers are scheduled one by one with the aim of minimizing the total peak power consumption. Everytime a register is selected for scheduling, its timing range is maximized without changing the timing of other registers and without disturbing the proper operation of the circuit. In this way a better minimization can be obtained while having a feasible output clock schedule. The algorithm stops when all the registers are placed within their timing ranges with minimum total peak power.

The first stage of the clock scheduling depends on the initial clock schedule and the order

Procedure MinPeak1(R, T, $D(r_1, r_2)$, $W(r, t)$)
Input: Set of registers R, clock period T, the delay between registers, $D(r_1, r_2)$ $\forall r_1, r_2 \in R$, the register originated power consumptions $W(r, t)$ $\forall r \in R$
Output: Clock schedule S minimizing the peak power consumption

1: Find an initial feasible clock schedule using **ClockSchedule** Algorithm
2: Set the timing of each register to the middle of its clock timing range determined by the **ClockSchedule** algorithm
3: $R_u := R$, $W^S(t) := 0$ $(0 \leq t < T)$
4: **while** $R_u \neq \emptyset$ **do**
5: Choose the register r_t in R_u with the largest peak register originated power consumption: $W_{peak}(r_t) = \max_{r \in R_u} W_{peak}(r)$
6: Maximize the clock timing range of r_t without changing the clock timings of other registers
7: Set the clock timing of r_t within its clock timing range, so that $W^S(t) + W^S(r_t, t)$ is minimized
8: $W^S(t) := W^S(t) + W^S(r_t, t)$, $R_u := R_u \backslash r_t$
9: **end while**
10: **return** S

Figure 3.9: First stage of the two stage algorithm that minimizes the peak power consumption of a circuit with clock scheduling.

Procedure MinPeak2(R, T, $D(r_1, r_2)$, $W(r, t)$, S, $W^S(t)$)
Input: Set of registers R, clock period T, the delay between registers, $D(r_1, r_2)$ $\forall r_1, r_2 \in R$, the register originated power consumptions, $W(r, t)$ $\forall r \in R$, the clock schedule output, S, of MinPeak1 Algorithm, the power consumption of the circuit, $W^S(r, t)$ $\forall r \in R$ $(0 \leq t < T)$
Output: Clock schedule S, minimizing the peak power consumption

1: $R_u := R$
2: **while** $R_u \neq \emptyset$ **do**
3: Choose the time t_{max} $(0 \leq t_{max} < T)$ when the power consumption of the circuit is maximum, that is, $W^S(t_{max}) = \max_{0 \leq t < T} W^S(t)$
4: Choose the register $r_t \in R_u$ with largest register originated power consumption at time t_{max}, that is, $W^S(r_t, t_{max})$ is maximum for r_t
5: Maximize the clock timing range of r_t without changing the clock timings of other registers
6: Set the clock timing of r_t within its clock timing range, so that $W^S(t)$ is minimized
7: **if** the clock timing of r_t is changed **then**
8: $R_u := R$
9: **else**
10: $R_u := R_u \backslash r_t$
11: **end if**
12: **end while**
13: **return** S

Figure 3.10: Second stage of the two stage algorithm that greedily minimizes the peak power consumption of a circuit with clock scheduling.

of the register processed [46], so the output of the first stage of the algorithm is greedily modified in the second stage of the algorithm. The second stage of the algorithm shown in Fig. 3.10 starts by finding the time t_{max} when the power consumption of the circuit is at maximum. Then, the register with the largest register originated power consumption at t_{max} is selected and its timing range is maximized without changing the timings of other registers and without disturbing the proper operation of the circuit. The clock timing of this register is set in its timing range so that the peak power consumption of the circuit is minimized. Since the algorithm is greedy, the register is removed from the set of registers to be processed, only if the timing of the selected register can not be changed any more to reduce the objective. The algorithm outputs the final clock schedule when there is no register left to be processed.

CHAPTER 4

SIMULATIONS AND TESTS

The clock scheduling algorithms discussed in the previous chapters require the delay and the switching power information of the gates and the registers. The propagation delays of the gates need to be known for the clock period minimization algorithm. Since the performance of the clock period minimization algorithm depends heavily on the accuracy of the gate delays, the dependence of the gate delay with respect to the load capacitance is modeled. Also, the setup time and hold time of the D flip-flop, which is used as the register in the circuits, is measured. The peak power reduction algorithm uses the switching power data of the gates and the D flip-flop. The dependence of the switching current waves of gates and the D flip-flop on the load capacitance is modeled by simulations.

In order to verify the correct operation and evaluate the performance of the clock scheduling algorithms, selected test circuits from the ISCAS'89 benchmark suite are used. The outputs of the clock period minimization algorithm and the peak power minimization algorithm are tested with simulations of the benchmark circuits.

This chapter begins with the details of the simulations for the delay modeling and power characterization of the gates. The second section covers the setup time and hold time measurements and the switching power characterization of the D flip-flop. The performance evaluation of the clock scheduling algorithms is covered in the third section. The implementation details are given along with the information on the test circuits. The simulation results are presented and the performance of the algorithms are discussed.

4.1 Gate Characterization

The test circuits that are used for the evaluation of the clock scheduling algorithms consists of inverters (NOTs), 2,3 and 4-input ANDs, NANDs, ORs, NORs and D Flip-Flops. The gates in Alcatel-Mietec MTC45000 0.35μm standard cell library are used for implementing the test circuits. The delay and power characterization of the gates are done in order to be used in the clock scheduling algorithms. All the simulations are done with SpectreS^{TM}Analog Simulator of Cadence^{TM}Design Environment.

4.1.1 Gate Propagation Delay Modeling

Propagation delay is defined as the time it takes for the output of a gate to reach 50% of its peak value after the input reaches 50% of its peak value [66]. The propagation delays of the gates are measured by simulations. Figure 4.1 shows the delay simulation circuit of a 4-input AND gate as an example. The circuit schematics for other types of gates are omitted due to the similarity. The square wave through five inverters is given to the gate in order to create a more realistic input with low slew rate. A capacitive load is used at the output of the gate. The delay simulation is done in two phases. In the first phase, simulations for every possible combination of switching inputs are done with a fixed load capacitance in order to determine which combination of switching inputs creates the longest propagation delay. Then, this switching scheme is simulated with capacitive loads ranging from 1 fF to 250 fF in 1 fF steps, in order to determine the relation of the propagation delay with the load capacitance value. The inputs that do not switch for a switching input combination are fixed to logic “1” for AND and NAND gates; to logic “0” for OR and NOR gates. Table 4.1 shows the result of the first phase of delay simulation for 4-input AND gate. Since the longest propagation delay is observed for the case when only the input D is switching, in the second phase of the simulation only input D is switched while other inputs are kept at logic “1”. The average of propagation delays for inputs switching from logic “0” to logic “1”, and from logic “0” to logic “1” are used as the final propagation delay values of the gates. The measurements are done with 1 ps precision, which is more than enough since the time precision of the clock scheduling algorithm is selected as 10 ps. Figure 4.2 shows the variation of propagation delay of 4-input AND gate with respect to the load capacitance value ranging from 1 fF to 250 fF in 1 fF steps. The dependence of the propagation delay of each gate type on the load capacitance

Table 4.1: The propagation delays of the 4-input AND gate for 15 different combinations of switching inputs, as the result of the first phase of delay simulation. The propagation delay values for the input signal switching in two directions are measured and the average of the two values are also shown. The inputs that do not switch are fixed to logic "1" for AND gate. The unit of the values is picoseconds and the measurements are done with 1 ps precision.

	Input Signal		
Switching Input(s)	Falling	Rising	Average
A	172	145	159
B	229	179	204
C	277	204	241
D	315	219	267
AB	132	181	157
AC	130	174	152
AD	129	170	150
BC	162	209	186
BD	159	203	181
CD	183	231	207
ABC	123	211	167
ABD	122	297	210
ACD	121	203	162
BCD	142	235	189
ABCD	121	240	181

value is modeled by fitting linear curves to the measurement results using a spreadsheet. The fitted equations are used by the clock scheduling system in creating the constraint graph of a circuit.

4.1.2 Gate Switching Current Modeling

The switching current of the gates are characterized by simulating the gates with capacitive loads ranging from 1 fF to 100 fF in 1 fF steps and recording the supply current of the gates. The supply currents for every possible input combination that results in a switching at the output of the gate are simulated. The average of the simulation results for all the input combi-

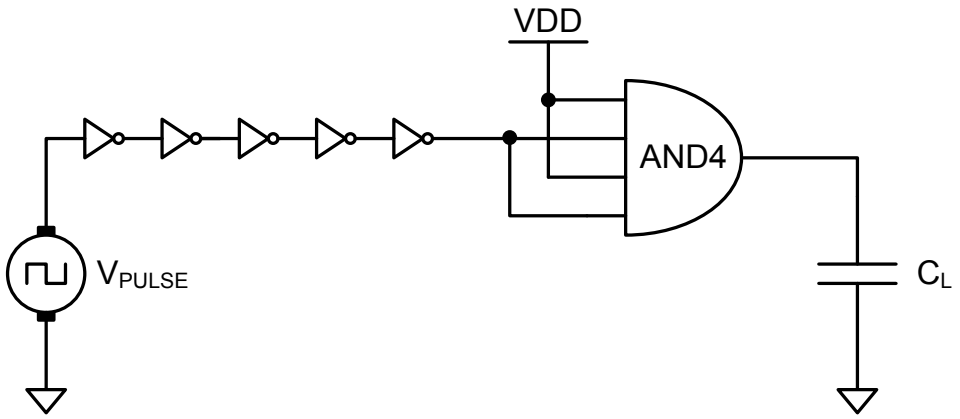

Figure 4.1: The circuit used in the propagation delay measurement of the 4-input AND gate. The case when the inputs A and C are fixed, and the inputs B and D are switched is shown.

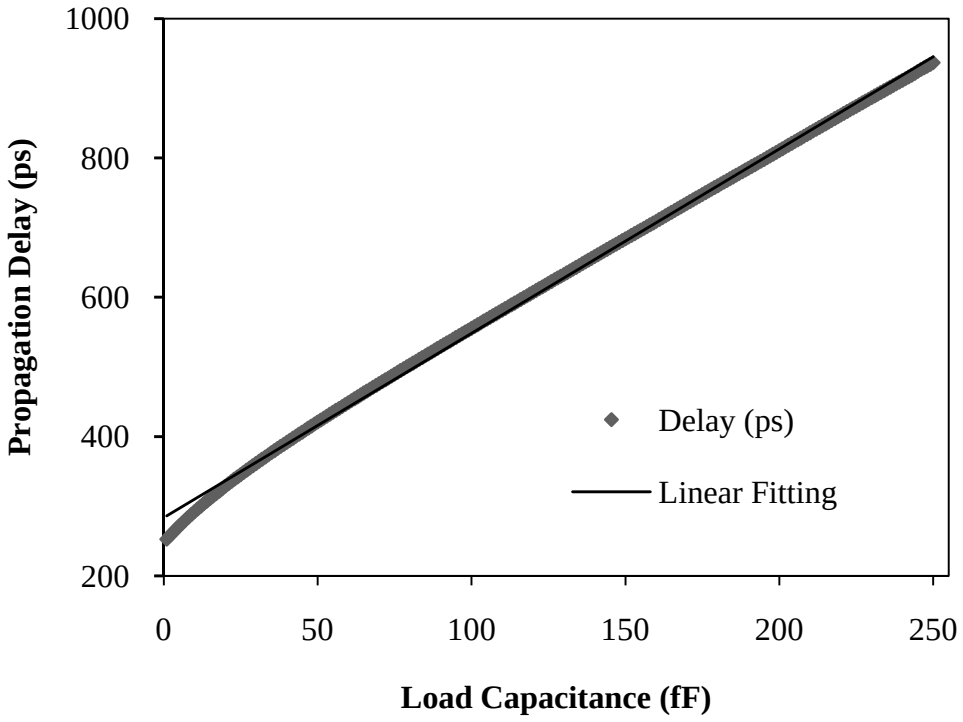

Figure 4.2: Variation of the propagation delay of the 4-input AND gate with respect to the load capacitance value. The fitted linear curve is also shown.

nations are used in the power estimation algorithm of the clock scheduling system. Figure 4.3 shows the switching power simulation circuit of the 2-input AND gate. The $2^n - 1$ different input combinations that causes switching at the output of an n-input gate, are considered by simulating $2^n - 1$ gates at once. The total power dissipation of the $2^n - 1$ gates are divided by $2^n - 1$ to find the average switching power dissipation.

The square wave from the ideal voltage source is passed through five inverters in order to have a more realistic input signal with low slew rate. The outputs are recorded in 1 ps steps and the data is used in determining the relation of the switching power current of a gate with respect to the value of the load capacitance. The current wave for the switching of the output from logic "0" to logic "1", i.e., charging the output capacitance, changes with load capacitance. However, the current wave for the switching of the output from logic "1" to logic "0", i.e., discharging the output capacitance, does not depend on the value of the load capacitance.

A computer program is written for analyzing the simulation data output and determining the four parameters used in characterization of the switching power wave as discussed in Subsection 3.1.3. The computer program extracts from the simulation output data, the peak value of the current (I_{peak}), and the time at which the current is at its peak value (t_{peak}), the times at which the current is at 10% of its peak value before and after the peak value is reached (t_{start} and t_{finish}) for each gate type and for each load capacitance value. Then, the computer program calculates the rise time (t_r) and fall time (t_f).

The curve fitting is done using a spreadsheet. The clock scheduling algorithm estimates the switching current wave for charging the actual load capacitance of the gate in the circuit using the relation as described in Subsection 3.1.3. The average of the calculated switching current for charging and the tabulated switching current for discharging the output capacitance is used for peak power estimation of the circuit.

4.2 Register Characterization

The test circuits are implemented with D flip-flops as registers. For simulations, the low power D flip-flop with positive edge triggered clock in Alcatel-Mietec MTC45000 0.35 μm standard cell library is used. Figure 4.5 shows the gate level schematics of the D flip-flop. The setup and hold times and the switching power of the D flip-flop are characterized with simulations.

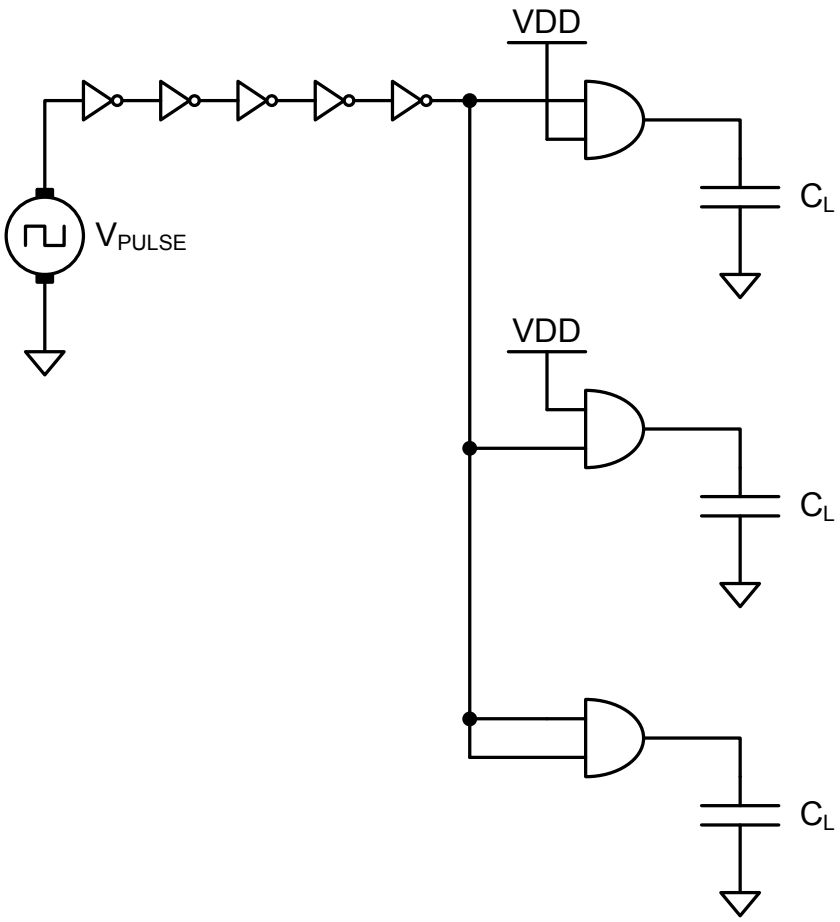

Figure 4.3: The circuit used in switching current measurement of the 2-input AND gate. All three switching input combinations that causes switching at the output are simulated at once. The total supply current of the three AND gates are divided by 3 to find the average switching power of the 2-input AND gate.

4.2.1 Register Setup and Hold Time Measurements

The time that the input of a register has to be stable before the clock edge is defined as the setup time of the register. Similarly, the time that the input of the register has to remain stable after the clock edge is defined as the hold time of the register [66]. Figure 4.4 shows the circuit used in setup time and hold time measurements of the D flip-flop.

For setup time measurement, two square wave signals with slightly different frequencies are applied to the data and clock input pins of the register. The clock signal is at 100 MHz whereas the input signal is at 100.2 MHz. As time passes, the input signal leads the clock wave more. The time between the clock and input waves when the output switches from logic

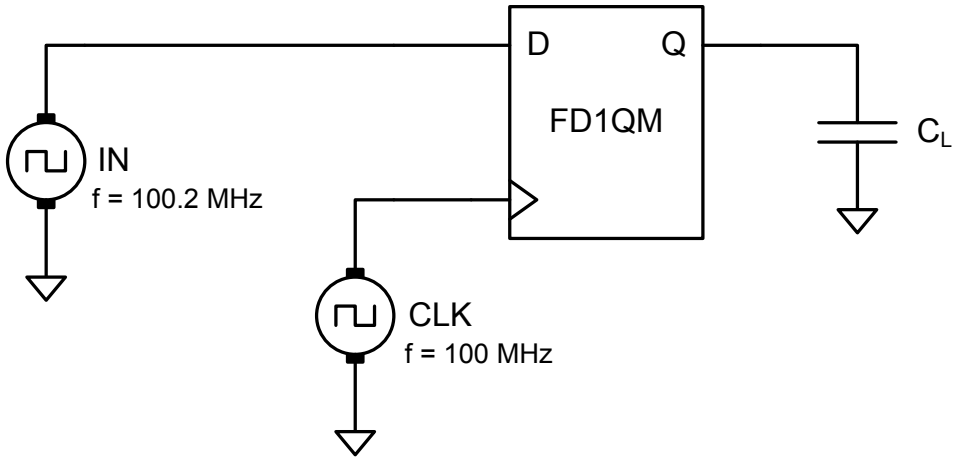

Figure 4.4: The circuit used in setup time and hold time measurements of the D flip-flop. Clock and input signals with slightly different frequencies are applied to the D flip-flop.

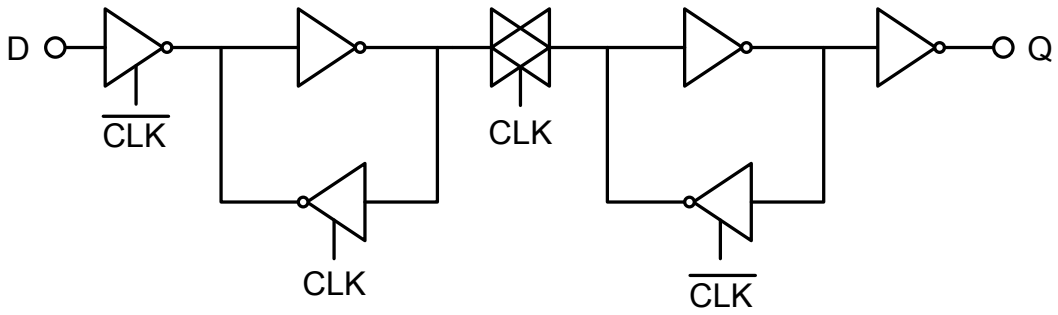

Figure 4.5: The gate level schematics of the D flip-flop that is used in the test circuits.

"0" to logic "1" for the first time is the setup time of the register. Figure 4.6 shows the input, clock and output signals from the simulation output when the setup time is measured. The high level input voltage (V_{IH}) is given as $0.8 \cdot V_{DD}$ for the process technology that is used for simulations, and the high level output voltage (V_{OH}) is given as $0.85 \cdot V_{DD} \sim 0.9 \cdot V_{DD}$ [67]. Hence, the difference between the times when the clock and input signals are at V_{IH}, at the clock edge when the output reaches V_{OH} value is measured. The setup time of the D flip-flop is measured to be 160 ps.

For hold time measurement, a method similar to setup time measurement method is used. The clock signal is a square wave at 100 MHz with 50% duty-cycle, whereas the input signal is at

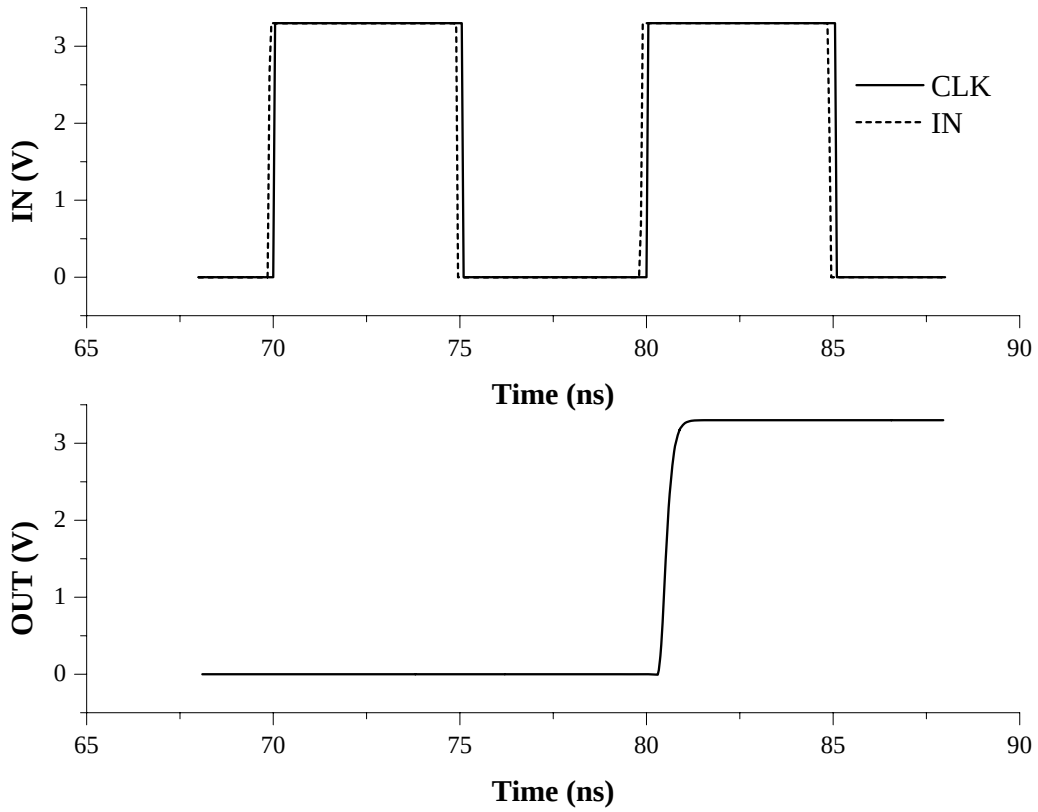

Figure 4.6: The input, clock and output signals from the simulation output when the setup time is measured.

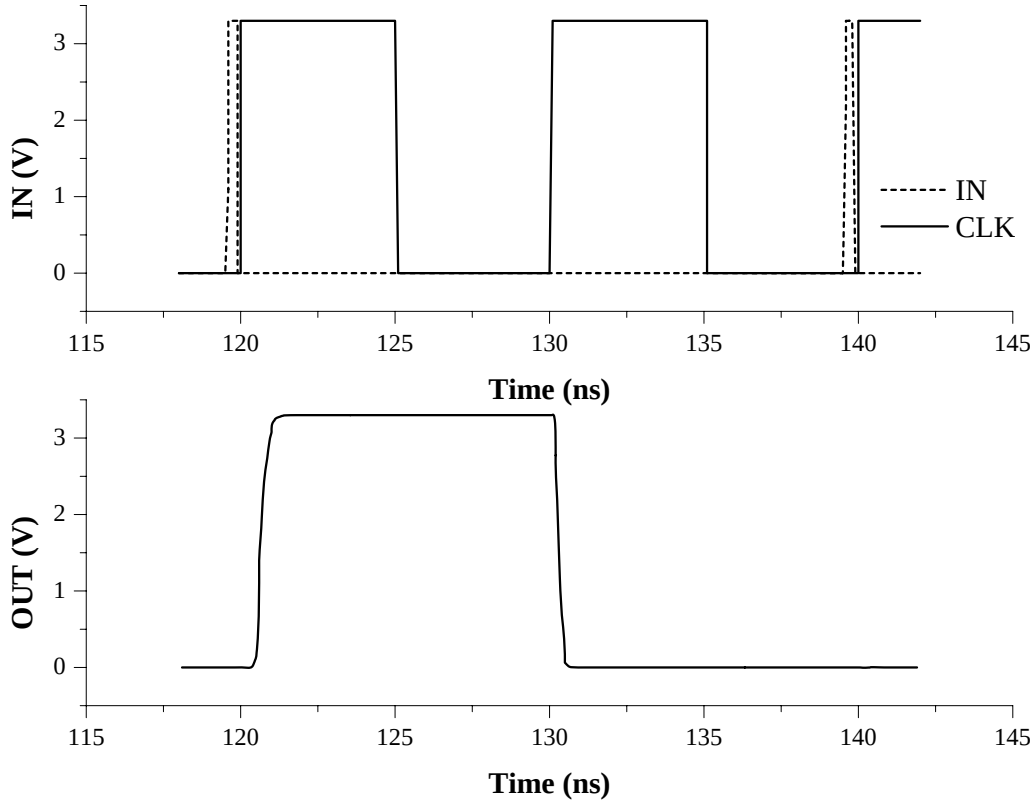

Figure 4.7: The input, clock and output signals from the simulation output when the hold time is measured.

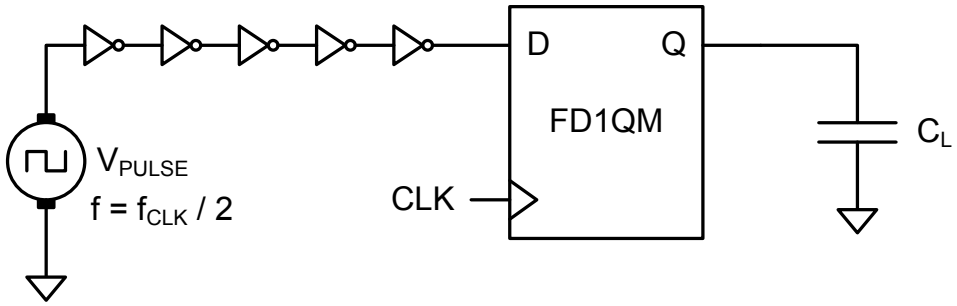

Figure 4.8: The circuit used in switching power measurement of the D flip-flop.

200.2 Mhz with 25% duty-cycle. The reason to use a double-frequency input signal with half the duty cycle is to reset the output of the register to logic "0" every other clock edge. As time passes, the input pulses lead the clock edge. At some clock edge, the output of the register can not reach V_{OH}, since the input falls from logic "1" to logic "0" before the hold-time. At this clock edge, the difference between the time when the clock is at V_{IH} and the time when the input is at V_{IL} is measured as the hold-time of the register. The hold time of the flip-flop is measured to be -187 ps. Figure 4.7 shows the input, clock and output signals from the simulation output when the hold time is measured.

4.2.2 Register Switching Power Measurements

The register is simulated using the circuit shown in Fig. 4.8. The input signal and the clock signal are both square wave signals. The frequency of the input signal is the half the frequency of the clock signal in order to observe the switching of the output in both directions. The clock signal is delayed by setup time with respect to the input signal for proper flip-flop operation. The average of the switching current waves for output switching in both directions are recorded. The variation of the shape of the switching current wave with respect to the load capacitance value is determined similar to gates.

4.3 Performance Evaluation of the Algorithms

4.3.1 Implementation of the Algorithms

The various clock scheduling algorithms and auxiliary algorithms that are used in the scope of this work are implemented in C++. Microsoft® Visual C++® Express Edition 2008 is used as the development environment. The graph data type of LEDA[1] Library is used for graph representation.

4.3.2 Test Circuits

Selected circuits from the ISCAS'89 benchmark suite are used for the evaluation of the algorithms. The benchmark suite is chosen because it offers a wide collection of synchronous sequential circuits and the prior utilization of the benchmark suite in similar works provides a comparison opportunity. The benchmark suite consists of synchronous sequential circuits that are composed of inverters (NOTs); 2,3 and 4-input ANDs, NANDs, ORs, NORs; and D flip-flops. The benchmark suite consists of a number of circuits ranging wide in size and function. Table 4.2 shows the information about the circuits in the benchmark. Although all the circuits are taken from actual industrial designs, only the function of a very limited number of circuits are known [68]. Reverse engineering efforts in order to determine the functions of the benchmark circuits are reported [69].

The circuits in the ISCAS'89 benchmark suite are supplied in a netlist format not supported by Cadence. Hence, a computer program is written to generate the gate level descriptions in Verilog Hardware Description Language (HDL). The gate level HDL descriptions are mapped to the gates in Alcatel-Mietec MTC45000 0.35μm standard cell library and the gate level schematics are imported to Cadence environment. The resulting schematics are simulated in SpectreS Analog Simulator.

The computer program takes the circuit graph as the input and outputs two gate level Verilog HDL descriptions of the circuit. The first output is for fully synchronous framework, with a single clock connected to all the flip-flops. The second output is for the semi synchronous

[1] In this work, the free edition of LEDA that contains no algorithms is used. LEDA is available from Algorithmic Solutions Software GmbH. See http://www.algorithmic-solutions.com.

Table 4.2: The characteristics of benchmark circuits.

Circuit Name	# of Inputs	# of Outputs	# of D Flip-Flops	# of Gates
s208	11	2	8	96
s298	3	6	14	119
s344	9	11	15	160
s382	3	6	21	158
s386	7	7	6	159
s400	3	6	21	162
s444	3	6	21	181
s510	19	7	6	211
s526	3	6	21	193
s641	35	24	19	379
s713	35	23	19	393
s820	18	19	5	289
s832	18	19	5	287
s838	35	2	32	390
s953	16	23	29	395
s1196	14	14	18	529
s1238	14	14	18	508
s1423	17	5	74	657
s5378	35	49	179	2779

framework, with separate clock inputs for each flip-flop. A sample circuit and the corresponding two gate level HDL descriptions are given in Appendix A for illustrative purposes.

4.3.3 Clock Period Minimization

The clock period minimization algorithm determines the minimum feasible clock period that a sequential circuit can operate without hazards and adjusts the clock timing of each register for operation with the minimum clock period. The precision of the algorithm is 1 ps. Table 4.3 shows the speed-up of the test circuits. In order to verify the operation of the algorithm the test circuits are simulated twice: once with fully synchronous clock and once with the clock schedule generated by the clock period minimization algorithm. The outputs of the circuits for the two simulations are recorded and compared with the each other. Comparison results show that, all the circuits operate properly with the minimum clock period. The circuits are simulated with random inputs because of the absence of information on the functions of all the circuits. However, the circuits whose functions are known are also simulated with realistic inputs. The details on the random input generation is explained in Appendix B. The clock period of all the test circuits are shortened by up to 45.37% except for one test circuit (s386).

4.3.4 Peak Power Minimization

The peak power minimization algorithm adjusts the clock timing of the registers in the circuit with the aim of minimizing the peak power consumption of the circuit. As discussed in Subsection 3.1.4, this is a two step process. The clock schedule outputs of the two steps are recorded and test circuits are simulated with both clock schedules. Test circuits are also simulated with fully synchronous clock for measuring the amount of peak power minimization. The test circuits are simulated with the minimum feasible clock period for fully synchronous operation in each case. In order to show that the peak power minimization does not disturb the proper operation of the circuit, the outputs for both cases are recorded and compared with the output of the simulation with fully synchronous clock. The comparisons show that the peak power minimization does not have a negative impact on the proper operation of the circuit, i.e., does not introduce timing hazards. The test circuits are simulated with random inputs. Moreover, the circuits whose functions are known are also simulated with realistic inputs. The unit time for the peak power estimation and the clock timing is selected as 10

Table 4.3: The speed-up of test circuits.

Circuit Name	Clock Period without Scheduling (ps)	# Clock Period with Scheduling (ps)	% Speed-Up
s208	1606	1119	30.32%
s298	1416	933	34.11%
s344	2212	1688	23.69%
s382	2112	1195	43.42%
s386	2227	2227	0.00%
s400	2266	1238	45.37%
s420	2979	2181	26.79%
s444	2188	1207	44.84%
s510	1679	1524	9.23%
s526	1553	1277	17.77%
s641	7160	6631	7.39%
s713	7523	6917	8.06%
s820	2643	2616	1.02%
s832	2686	2660	0.97%
s838	5287	4358	17.57%
s953	1951	1618	17.07%
s1196	3248	2581	20.54%
s1238	3356	2689	19.87%
s1423	9901	8415	15.01%
s5378	3431	2637	23.14%

ps, which is limited by the available computer memory. The simulation durations are limited by the available disk space and thus varies with the size of the circuits. Table 4.4 shows the duration of the simulations. A computer program is written in order to analyze the simulation outputs. The computer program determines the maximum power consumption among all the clock cycles for each time unit in the clock period.

Table 4.5 summarizes the results of the simulations. The algorithm decreases the peak power dissipation by up to 72.68%. Despite for several circuits, in which the second step of the algorithm resulted in larger peak power than the first step, the second step of the algorithm proved to be useful. Of course, the algorithm always estimates the peak power consumption after the second step to be lower than that after the first step. However, due to the nature of the probabilistic peak power estimation there are errors and this may result in faulty clock schedules for some circuits. Although in some cases the second step resulted in slightly larger peak power, the peak power consuption of the circuits are significantly decreased by the algorithm.

Comparing the peak power minimization results of test circuits s1238, s1423 and s5378 to those presented in [46] shows a similar result for s5378 (73.32%), whereas better peak power minimization is achieved in [46] for s1238 (63.51%) and s1423 (56.88%). However, the speed of the SPA algorithm is expected to outperform the method in [46] in terms of speed, because SPA eliminates the need for running a simulation for each gate in the circuit.

Table 4.4: The simulation durations of the test circuits.

Circuit	Sim Duration (# of clock periods)
s208	1000
s298	1000
s344	1000
s382	1000
s386	300
s400	300
s420	300
s444	300
s510	300
s526	300
s641	300
s713	300
s820	300
s832	300
s838	300
s953	300
s1196	300
s1238	300
s1423	200
s5378	16

Table 4.5: The peak power minimization of test circuits.

Circuit Name	Peak Supply Current without Scheduling (A)	Peak Supply Current with Scheduling Step 1 (A)	% Decrease	Peak Supply Current with Scheduling Step 2 (A)	% Decrease
s208	0.00678688	0.00435648	35.81%	0.00296034	56.38%
s298	0.0105936	0.00752192	29.00%	0.00806976	23.82%
s344	0.0178092	0.010167	42.91%	0.0100087	43.80%
s382	0.0178843	0.0083895	53.09%	0.00624071	65.11%
s386	0.0117345	0.00828416	29.40%	0.00846497	27.86%
s400	0.0159057	0.00686901	56.81%	0.00666514	58.10%
s420	0.0116429	0.00351301	69.83%	0.00352633	69.71%
s444	0.0168731	0.00668501	60.38%	0.00658724	60.96%
s510	0.0177533	0.00776628	56.25%	0.00689052	61.19%
s526	0.0135858	0.0082219	39.48%	0.0081779	39.81%
s641	0.0127367	0.00991737	22.14%	0.0099204	22.11%
s713	0.0120846	0.00865432	28.39%	0.00863658	28.53%
s820	0.0260013	0.0230006	11.54%	0.0215027	17.30%
s832	0.0262672	0.025251	3.87%	0.0193233	26.44%
s838	0.0147442	0.00550152	62.69%	0.00469743	68.14%
s953	0.0326145	0.0152455	53.26%	0.0152448	53.26%
s1196	0.0363944	0.0265775	26.97%	0.0265528	27.04%
s1238	0.0360493	0.028551	20.80%	0.0288205	20.05%
s1423	0.0259821	0.0200997	22.64%	0.0207246	20.24%
s5378	0.123983	0.033876	72.68%	0.0346984	72.01%

CHAPTER 5

CONCLUSION

Clock scheduling is applied to synchronous sequential circuits in order to improve two performance metrics of the circuits: clock period and peak power consumption. Graph theoretic approach of [25] is used for clock period minimization. In this approach, a binary search is made on the constraint graph, in which Bellman-Ford Algorithm is used to check if the clock schedule is feasible at each step. A negative weight cycle detection strategy, namely "modified walk-to-the-root", is utilized in order to determine the existence of the negative weight cycle at early steps of the Bellman-Ford Algorithm.

The accuracy of the clock scheduling algorithm for clock period minimization relies heavily on the determination of the propagation delays of gates used in the circuit. The propagation delay of a gate changes with the load capacitance value. Thus, the relation between the propagation delay and the load capacitance value should be well determined for the algorithm to produce accurate results. All the gate types used in the test circuits, namely: inverters, 2,3 and 4-input ANDs, NANDs, ORs and NORs, are simulated with varying load capacitances. Using the output data of these simulations, a model defining the propagation delay in terms of load capacitance is generated. This model is used in the algorithm to estimate the actual propagation delay of a gate in a circuit. Using only clock scheduling with no other modifications, up to 45.37% speed-up is observed in simulations of selected test circuits from ISCAS'89 benchmark suite. In order to verify that the proper operation of the circuits are not disturbed with the speed-up, the simulation outputs with and without clock scheduling are recorded and compared. It is shown that the test circuits operate properly at higher clock frequencies.

For peak power reduction, the two-stage clock scheduling algorithm of Takahashi given in [46] is implemented. The performance of this algorithm relies on the accuracy of the switch-

ing power estimation of gates and registers. In this work, a new method, namely the Shaped Pulse Approximation Method (SPA) is proposed for switching power estimation of gates and registers. The SPA method exploits the triangular approximation of supply current waves for determining the change in the shape of the current wave for different load values. The variations of the triangular approximation parameters of all the gate types used in the test circuits, namely: inverters, 2,3 and 4-input ANDs, NANDs, ORs and NORs, are modeled by simulations for a range of load capacitance values. The SPA method utilizes the model for the estimation of the supply currents of the gates for actual load capacitances by modifying the shape of a tabulated unit current wave, i.e., the magnitude of the unit current wave is adjusted with a magnitude correction factor and the wave is stretched in the time with two time correction factors. The SPA method outperforms the previous methods in the literature in terms of speed and memory storage requirement while offering high accuracy. The switching power dissipation of gates and registers for the actual load capacitance in the circuit can be estimated with less than 10% normalized rms error.

The peak power reduction algorithm is verified by simulating the selected test circuits from ISCAS'89 benchmark suite. The peak power reduction is done separately from the clock period minimization in order to have more room for optimization, since the clock timing ranges of the registers are wider for the nominal clock period. However, the algorithms can be applied simultaneously for minimizing the peak power and the clock period at the same time. Obviously, there will be a trade-off between the two metrics for this case. With the application of the clock scheduling for peak power reduction to the test circuits, up to 72.68% decrease in peak power consumption is observed in simulations. The simulation outputs with and without clock scheduling are recorded and compared in order to verify that the peak power minimization does not disturb the proper operation of the circuits. It is shown that the test circuits operate properly with lower peak power values. The electromagnetic radiation emission of a circuit is expected to be lower when scheduled for peak power reduction, because the supply current is not modulated by the clock period, i.e., the current peaks are distributed in time, and the peak values are lower [70].

The clock scheduling algorithms in this work are implemented with the assumption that the clock timings of all the registers can be adjusted. The methods for clock tree synthesis and clock delay adjustment are beyond the scope of this research. The unit time of clock scheduling can be adjusted in the clock scheduling system considering the precision of the delay

generation in the clock tree synthesis.

The power consumption of the clock distribution network of a circuit may increase with the introduction of the delays. However, there are methods in the literature for low-power clock tree synthesis such as [56], so that the overall peak power of the circuit can be minimized without extra power overhead of the clock distribution network. Hence, the power dissipation of the clock distribution network is not considered within the scope of this work.

Although major research objectives are accomplished by realizing a complete clock scheduling system and proposing a new power estimation method, there is still need for further research for improving the performance of the algorithms. Firstly, the memory usage of the peak power reduction algorithm could be optimized enabling it to handle larger circuits and smaller unit times. Secondly, the estimation error of the proposed switching power estimation algorithm could be further reduced by taking into consideration the effect of the input slew rate on the power dissipation of circuit elements. Finally, the clock scheduling system could be integrated into the available synthesis tools for more user friendly operation.

In conclusion, the major achievement of this research is the development of a new power estimation method for circuit elements to be used in the clock scheduling algorithm aiming at peak power reduction. With the help of the implemented clock scheduling system, the performance of synchronous sequential circuits can be improved in speed and peak power consumption. It is believed that the theoretical background and test results provided in this study would be helpful to the development of even better power estimation algorithms and clock scheduling systems.

REFERENCES

[1] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 114-117, 19 April 1965.

[2] *International Technology Roadmap for Semiconductors, 2008 Update*, SIA, Available: http://www.itrs.net/Links/2008ITRS/Home2008.htm, January 2008, Last Accessed in December 2009.

[3] L. W. Cotten, "Circuit implementation of high-speed pipeline systems," *AFIPS Joint Computer Conferences, Proc. of the November 30 – December 1, 1965, Fall Joint Computer Conference, part 1*, pp. 489-504, 1965.

[4] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh, "Clock routing for high performance IC's," *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 573-579, 1990.

[5] R. S. Tsay, "Exact zero skew," *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 336-339, 1991.

[6] M. Edahiro, "A clustering-based optimization algorithm in zero-skew routings," *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 612-616, 1993.

[7] R. S. Tsay, "An exact zero-skew clock routing algorithm," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 12, no. 2, February 1993.

[8] J. D. Cho and M. Sarrafzadeh, "A buffer distribution algorithm for high performance clock net optimization," *IEEE Transactions on VLSI Systems*, vol. 3, no. 1, March 1995.

[9] N. C. Chou and C. K. Cheng, "On general zero-skew clock net construction," *IEEE Transactions on VLSI Systems*, vol. 3, no. 1, March 1995.

[10] A. Takahashi and Y. Kajitani, "Performance and reliability driven clock scheduling of sequential logic circuits," *Proc. of Asia and South Pacific Design Automation Conference (ASPDAC)*, pp. 37-42, 1997.

[11] *Actel FPGA Databook and Design Guide*, Actel Corporation, Mountain View, CA, 1996.

[12] R. B. Deokar and S. S. Sapatnekar, "A graph-theoretic approach to clock skew optimization," *Proc. International Symposium on Circuits and Systems (ISCAS)*, pp. 407-410, 1994.

[13] J. P. Fishburn, "Clock skew optimization," *IEEE Transactions on Computers*, vol. 39, no. 7, pp. 945-951, 1990.

[14] K. Sakallah, T. N. Mudge, and O. A. Olukotun, "Check T_c and min T_c: Timing verification and optimal clocking of synchronous digital circuits," *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 552-555, 1990.

[15] T. G. Szymanski, "LEADOUT: A static timing analyzer for MOS circuits," *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 552-555, 1990.

[16] D. A. Joy and M. J. Ciesielski, "Placement for clock period minimization with multiple wave propagation," *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 640-643, 1991.

[17] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, no. 1, pp. 5-35, 1991.

[18] M. C. Papaefthymiou and K. H. Randall, "TIM: A timing package for two-phase level clocked circuitry," *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 497-502, 1993.

[19] N. Maheshwari and S. S. Sapatnekar, "Optimizing large multiphase level-clocked circuits," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 18, no. 9, pp. 1249-1264, September 1999.

[20] N. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Graph algorithms for clock schedule optimization," *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 132-136, 1992.

[21] T. G. Szymanski, "Computing optimal clock schedules," *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 399-404, 1992.

[22] T. M. Burks and K. A. Sakallah, "Min-max linear programming and the timing analysis of digital circuits," *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 152-155, 1993.

[23] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "Analysis and design of latch-controlled synchronous digital circuits," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 11, no. 3, March 1992.

[24] J. L. Neves and E. G. Friedman, "Optimal clock skew scheduling tolerant to process variations," *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 623-628, 1996.

[25] A. Takahashi, "Practical fast clock schedule design algorithms," *IEICE Transactions on Fundamentals of Electronics*, vol. E89-A, no. 4, pp. 1005-1011, 2006.

[26] C. Albrecht, B. Korte, J.Schietke, and J. Vygen, "Cycle time and slack optimization for VLSI-chips," *Proc. International Conference on Computer Aided Design (ICCAD)*, pp. 233-238, 1999.

[27] T. Yoda and A. Takahashi, "Clock schedule design for minimum realization cost," *IEICE Transactions on Fundamentals of Electronics*, vol. E83-A, no. 12, pp. 2552-2557, 2000.

[28] J. G. Xi and W. W. M. Dai, "Jitter-tolerant clock routing in two-phase synchronous systems," *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 316-320, 1996.

[29] I. Kourtev and E. Friedman, "Clock skew scheduling for improved reliability via quadratic programming," *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 239-243, 1999.

[30] R. Mader, E. Friedman, A. Litman, and I. Kourtev, "Large scale clock skew scheduling techniques for improved reliability of digital synchronous VLSI circuits," *Proc. International Symposium on Circuits and Systems (ISCAS)*, pp. 357-360, 2002.

[31] R. B. Deokar and S. S. Sapatnekar, "A fresh look at retiming via clock skew optimization," *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 310-315, 1995.

[32] X. Liu, M. C. Capaefthymiou, and E. G. Friedman, "Maximizing performance by retiming and clock skew scheduling," *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 231-236, 1999.

[33] X. Wei, Y. Cai, and X. Hong, "Clock skew scheduling under process variations," *Proc. Seventh International Symposium on Quality Electronic Design (ISQED)*, pp. 237-242, 2006.

[34] P. Vuillod, L. Benini, A. Bogliolo, and G. De Micheli, "Clock skew optimization for peak current reduction," *Journal of VLSI Signal Processing*, vol. 16, pp. 117-130, 1997.

[35] S. Chowdury and J. Barkatullah, "Estimation of maximum currents in MOS IC logic circuits," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 9, no. 6, pp. 642-654, 1990.

[36] F. M. D'Heurle, "Electromigration and failure in electronics: An introduction," *Proceedings of the IEEE*, vol. 59, pp. 1409-1418, October 1971.

[37] F. Najm, Transition density, "A stochastic measure of activity in digital circuits," *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 644-649, June 1991.

[38] R. Burch, F. Najm, P. Yang, and D. Hocevar, "Pattern-independent current estimation for reliability analysis of CMOS circuits," *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 294-299, June 1988.

[39] S. Devadas, K. Keutzer, and J. White, "Estimation of power dissipation in CMOS combinational circuits," *Proc. IEEE Custom Integrated Circuits Conference (IEEE-CICC)*, pp. 19.7.1-19.7.6, May 1990.

[40] F. Najm, "Estimating power dissipation in VLSI circuits," *IEEE Circuits and Devices Magazine*, vol. 10, no. 4, pp. 11-19, July 1994.

[41] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 253-259, June 1992.

[42] J. Monteiro, S. Devadas, and A. Ghosh, "Retiming sequential circuits for low power," *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 398-402, 1993.

[43] W. C. D. Lam, C. K. Koh, and C. W. A. Tsao, "Clock scheduling for power supply noise suppression using genetic algorithm with selective gene therapy," *Proc. Fourth International Symposium on Quality Electronic Design (ISQED)*, pp. 327-332, 2003.

[44] W. C. D. Lam, C. K. Koh, and C. W. A. Tsao, "Power supply noise suppression via clock scheduling," *Proc. Third International Symposium on Quality Electronic Design (ISQED)*, pp. 355-360, 2002.

[45] M. Ni and S. O. Memik, "Leakage power aware clock skew scheduling: Converting stolen time into leakage power reduction," *Proc. 45th Annual Design Automation Conference*, pp. 610-613, 2008.

[46] Y. Takahashi, Y. Kohira, and A. Takahashi, "A fast clock scheduling for peak power reduction in LSI," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E91-A, no. 12, pp. 3803-3811, 2008.

[47] A. Takahahsi, K. Inoue, and Y. Kajitani, "Clock-tree routing realizing a clock-schedule for semi-synchronous circuits," *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 260-265, 1997.

[48] K. Inoue, W. Takahashi, A. Takahahsi, and Y. Kajitani, "Schedule-clock-tree routing for semi-synchronous circuits," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E82-A, no. 11, pp. 2431-2439, November 1999.

[49] D. J. H. Huang, A. B. Kahng, and C. W. A. Tsao, "On the bounded-skew clock and steiner routing problems," *Proc. 32nd Annual Design Automation Conference*, pp. 508-513, 1995.

[50] A. B. Kahng, and C. W. A. Tsao, "More practical bounded-skew clock routing," *Proc. 34th Annual Design Automation Conference*, pp. 594-599, 1997.

[51] J. G. Xi, and W. M. Dai, "Useful-skew clock routing with gate sizing for low power design," *Proc. 33rd Annual Design Automation Conference*, pp. 383-388, 1996.

[52] B. J. George, G. Yeap, M. Wloka, S. C. Tyler, and D. Gossain, "Power analysis for semi-custom design," *Proceedings of the Custom Integrated Circuits Conference*, pp. 249-252, 1994.

[53] A. Bogliolo, L. Benini, G. De Micheli, and B. Ricco, "Gate-level current waveform simulation of CMOS integrated circuits," *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 109-112, 1996.

[54] A. Bogliolo, L. Benini, and B. Ricco, "Power estimation of cell-based CMOS circuits," *Proceedings of the 33rd Annual Design Automation Conference*, pp. 433-438, 1996.

[55] A. Chandrakasan, T. Sheng, and R. W. Brodersen, "Low power CMOS digital design," *IEEE Journal of Solid State Circuits*, vol.27, no. 10, pp. 473-484, April 1992.

[56] K. Kurokawa, T. Yasui, Y. Matsumura, M. Toyonaga, and A. Takahashi, "A high-speed and low-power clock tree synthesis by dynamic clock scheduling," *IEICE Transactions on Fundamentals of Electronics*, vol. E85-A, no. 12, pp. 2746-2755, 2002.

[57] B. Cherkassky and A. Goldberg, "Negative-cycle detection algorithms," *Math. Program.*, vol. 85, pp. 277-311, 1999.

[58] A. Kuehlmann, K. Ravindran, and E. Sentovich, Multi-domain clock skew scheduling, US Patent 7,296,246 B1, November 13, 2007.

[59] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed., Cambridge, MA, USA: MIT Press, 2001.

[60] S. S. Sapatnekar and R. B. Deokar, "Utilizing the retiming-skew equivalence in a practical algorithm for retiming large circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15 no. 10, pp. 1237-1248, October 1996.

[61] S. H. Huang and Y. T. Nieh, "Clock skew scheduling with race conditions considered," *ACM Transactions on Design Automation of Electronic Systems*, vol. 12, no. 4, September 2007.

[62] J. H. Hopcroft and R. M. Karp, "A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM Journal on Computing*, vol. 2, no. 4, pp. 225-231, 1973.

[63] Z. Galil, "Efficient algorithms for finding maximum matching in graphs," *Computing Surveys*, vol. 18, no. 1, pp. 225-231, March 1986.

[64] N. Blum, "A simplified realization of the Hopcroft-Karp approach to maximum matching in general graphs," University of Bonn, Bonn, Germany, Technical Report, 85232-CS, October 1999.

[65] J. Loughry, J. I. van Hemert, and L. Schoofs, "Efficiently enumerating the subsets of a set," preprint, Available: http://www.applied-math.org/subset.pdf , Last accessed in December 2009.

[66] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective*, 2nd ed., Addison-Wesley Publishing Company, 2001.

[67] *MTC-45000 CMOS 0.35 μ Standard Cell Library Product Description*, Alcatel Microelectronics, Belgium, 1998.

[68] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1924-1934, May 1989.

[69] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 Benchmarks: A case study in reverse engineering," *IEEE Design and Test of Computers*, pp. 72-80, 1999.

[70] C. H. van Berkel, M. B. Josephs, and S. M. Nowick, "Scanning the Technology: Applications of Asynchronous Circuits," *Proceedings of the IEEE*, vol. 87, pp. 223-233, February 1999.

Appendix A

HDL GENERATOR

Figure A.1 shows an example circuit. The gate level Verilog HDL description of this circuit generated by the computer program for fully synchronous framework is listed below:

```
module circ(CLK,G0,G1,G6);

input CLK,G0,G1;
output G6;

wire G10,G11,G12,G13,G14,G15,G2,G3,G4,G5;
wire G7,G8,G9;

FD1QM DFF_1(.Q(G13),.D(G12),.CP(CLK));
FD1QM DFF_2(.Q(G4),.D(G3),.CP(CLK));
FD1QM DFF_3(.Q(G6),.D(G5),.CP(CLK));
```

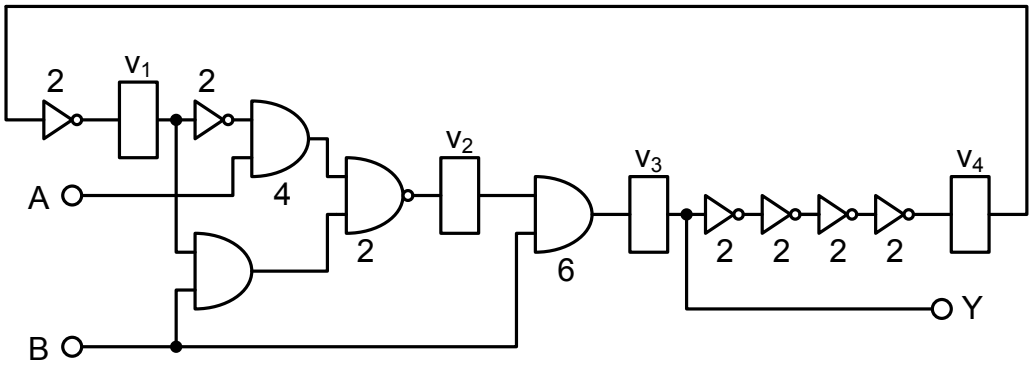

Figure A.1: Example circuit [58]

```
FD1QM DFF_4(.Q(G11),.D(G10),.CP(CLK));
IV IV_1(.Z(G12),.A(G11));
IV IV_2(.Z(G14),.A(G13));
IV IV_3(.Z(G7),.A(G6));
IV IV_4(.Z(G8),.A(G7));
IV IV_5(.Z(G9),.A(G8));
IV IV_6(.Z(G10),.A(G9));
AN2 AN_1(.Z(G2),.A(G1),.B(G13));
AN2 AN_2(.Z(G15),.A(G0),.B(G14));
AN2 AN_3(.Z(G5),.A(G1),.B(G4));
ND2 ND_1(.Z(G3),.A(G15),.B(G2));

endmodule
```

The Verilog HDL description of the same circuit for semi-synchronous framework is listed below:

```
module circ_SC(CLK_1,CLK_2,CLK_3,CLK_4,G0,G1,G6);

input CLK_1,CLK_2,CLK_3,CLK_4,G0,G1;
output G6;

wire G10,G11,G12,G13,G14,G15,G2,G3,G4,G5;
wire G7,G8,G9;

FD1QM DFF_1(.Q(G13),.D(G12),.CP(CLK_1));
FD1QM DFF_2(.Q(G4),.D(G3),.CP(CLK_2));
FD1QM DFF_3(.Q(G6),.D(G5),.CP(CLK_3));
FD1QM DFF_4(.Q(G11),.D(G10),.CP(CLK_4));
IV IV_1(.Z(G12),.A(G11));
IV IV_2(.Z(G14),.A(G13));
IV IV_3(.Z(G7),.A(G6));
IV IV_4(.Z(G8),.A(G7));
```

```
IV IV_5(.Z(G9),.A(G8));
IV IV_6(.Z(G10),.A(G9));
AN2 AN_1(.Z(G2),.A(G1),.B(G13));
AN2 AN_2(.Z(G15),.A(G0),.B(G14));
AN2 AN_3(.Z(G5),.A(G1),.B(G4));
ND2 ND_1(.Z(G3),.A(G15),.B(G2));

endmodule
```

Appendix B

RANDOM INPUT GENERATOR

The Random Input Generator, generates input vectors for the simulations of the test circuits. The program is implemented in C++, and uses the pseudo-random number sequence generator function `rand()` of the C programming language. The second of the clock is used as the seed of the pseudo-random sequence. The inputs of the program are the number of inputs of the circuit, the clock period and the simulation duration. One text file for each input is generated as the outputs of the program, which are in a format recognized by SpectreS Analog Simulator of Cadence. In order to prevent the metastability issues due to the simultaneous change in the clock signal and the input signals, the inputs change at the falling edges of the clock whereas the flip-flops in the circuits are rising edge triggered. Figure B.1 shows a timing diagram of example input signals with the clock signal. For each negative clock edge the program generates two random numbers and compares them. If the second one is larger, then the input is logic "0", else the input is logic "1".

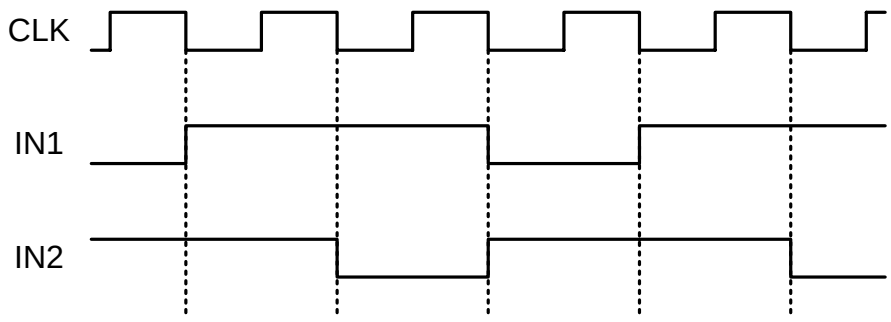

Figure B.1: Timing diagram showing the clock signal and two example input signals. The input signals change with the falling edges of the clock in order to eliminate metastability issues.