UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MICROELETRÔNICA

GUSTAVO REIS WILKE

Analysis and Optimization of Mesh-based Clock Distribution Architectures

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Microelectronics

Ricardo Augusto da Luz Reis
Advisor

Rajeev Murgai
Coadvisor

Porto Alegre, September 2008

CIP – CATALOGING-IN-PUBLICATION

Wilke, Gustavo Reis

Analysis and Optimization of Mesh-based Clock Distribution Architectures / Gustavo Reis Wilke. – Porto Alegre: PGMICRO da UFRGS, 2008.

155 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica, Porto Alegre, BR–RS, 2008. Advisor: Ricardo Augusto da Luz Reis; Coadvisor: Rajeev Murgai.

1. Clock. 2. Clock mesh. 3. Skew. 4. High performance. 5. Microprocessor. 6. Variability. I. Reis, Ricardo Augusto da Luz. II. Murgai, Rajeev. III. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. José Carlos Ferraz Hennemann
Vice-Reitor: Prof. Pedro Cezar Dutra Fonseca
Pró-Reitora de Pós-Graduação: Prof^{a}. Valquíria Linck Bassani
Diretor do Instituto de Informática: Prof. FlÃąvio Rech Wagner
Coordenador do PGMICRO: Prof. Henri Ivanov Boudinov
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

You have to be, then you have to do, then you will have...in that order.
— Ricardo Benjamin Salinas Pliego

ACKNOWLEDGMENTS

Agradeço a minha companheira Roberta por todo o apoio me dado para concluir essa tese e por estar sempre e incondicionalmente ao meu lado. Agredeço aos meus pais por terem propiciado anos de dedicação exclusiva a um ensino de qualidade que me permitiu formar a base do meu conhecimento.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

ASIC	Application Specific Integrated Circuit
CDF	Cumulative Distribution Function
CVD	Clock Vernier Device
DFD	Digital Frequency Dividers
DLL	Delay Locked Loop
DME	Deferred-Merge Embedding
FF	Flip-Flop
FO	Fanout Of
GCG	Global Clock Grid
IA	Instruction set Architecture
IO	Input-Output
LCB	Local Clock Buffer
LCD	Local Clock Driver
MC	Monte Carlo
MLT	Mesh + Local Trees
MMM	Method of Mean and Medians
MOR	Model Order Reduction
PDF	Probabilty Density Function
PGCN	Pre-Global Clock Network
PLL	Phase Locked Loop
PVT	Process Voltage and Temperature
SLCB	Second Level Clock Buffers
SPD	Symmetric Positive Definite
SOI	Silicon Over Insulator
SWS	Sliding Window Scheme
VT	Voltage Threshold

TLM	Tree + Local Meshes
UC	Units of Capacitance

LIST OF SYMBOLS

f	Femto
μ	Micron/Mean
m	Milli
n	Nano
Ω	Ohms
p	Pico
$\sum$	Summation
σ	Standard deviation

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Process and environmental variations are a great challenge to clock network designers. Variations effect on the clock network delays can not be predicted, hence it can not be directly accounted in the design stage. Clock mesh-based structures (i.e. clock mesh, clock spines and crosslinks) are the most effective way to tolerate variation effects on delays. Clock meshes have been used for a long time in microprocessor designs and recently became supported by commercial tools in the ASIC design flow. Although clock meshes have been known for some time and its use in ASIC design is increasing, there is a lack of good analysis and optimization strategies for clock meshes. This thesis tackles both problems.

Chapter 1 presents a basic introduction to clock distribution and important definitions. A review of existent clock dsitribution design strategies is presented in chapter 2. A study about the clock distribution architecture used in several microprocessor and a comparison between mesh-based and pure tree clock distribution architectures is shown in chapter 3.2. A methodology for enabling and speeding up the simulation of large clock meshes is presented in chapter 4. The proposed analysis methodology was shown to enable the parallel evaluation of large clock meshes with an error smaller than 1%. Chapter 5 presents two optimization strategies, a new mesh buffer design and a mesh buffer sizing algorithm. The new mesh buffer design was proposed improving clock skew by 22% and clock power by 59%. The mesh buffer sizing algorithm can reduce clock skew by 33%, power consumption by 20% with at the cost of a 26% slew increase. At last conclusions are presented on chapter 6.

Keywords: Clock, Clock mesh, Skew, High performance, Microprocessor, Variability.

Analise e Otimização de Arquiteturas de Relógio do Tipo Malha

RESUMO

Variações ambientais e de processo representam um grande desafio a ser vencido pelas redes de distribuição de relógio. O efeito das variações nos atrasos da rede de distribuição de relógio não pode ser previsto com precisão e portanto não podem ser diretamente considerados no projeto das redes de distribuição de relógio. Estruturas baseadas em clock meshes (i.e. clock mesh, clock spines e crosslinks) são a maneira mais eficiente de proteger a rede de relógio do efeito das variações nos atrasos. Clock meshes tem sido utilizados por bastante tempo no projeto de microprocessadores e recentemente foram incluídos no fluxo de síntese de ASICs. Embora o uso de clock meshes esteja aumentando há uma grande necessidade por métodos de analise e otimização dos mesmos. Essa tese propõe soluções para ambos os problemas. Uma metodologia para permitir a simulação elétrica de clock meshes grandes é proposta. O método proposto permite que a simulação dos clock meshes seja paralelizada com um erro menor que 1%. Duas metodologias de otimização também são propostas nessa tese. A primeira consiste em um algoritmo para dimensionamento para os mesh buffers. Esse algoritmo permite que o clock skew e o consumo de potência sejam reduzidos ao custo de aumentar o clock slew. O segundo método de otimização proposto consiste em um novo projeto para os mesh buffers. O novo mesh buffer é capaz de reduzir o clock skew em 22% e o consumo de potencia em 59%.

Palavras-chave: Distribuição de Relógio, Desempenho, Variabilidade.

1 INTRODUCTION

The clock is the most important signal in any synchronous design. It controls the instant data is stored inside every sequential element. If clock timing is not extremely accurate, invalid data can be stored inside sequential elements. The clock period must be defined in such a way that data will always be ready and stable before clock edge arrives at the clock sinks.

Figure 1.1 shows the timing parameters that must be considered to safely determine clock frequency. Assume that $T_{CK}(n)'$ is clock arrival time at flip-flop A, at clock cycle n and $T_{CK}(n+1)''$ is clock arrival time at flip-flop B during clock cycle $(n+1)$. Data propagation time throught flip-flop A is represented by TP_{FFA}. Combinational logic delay is represented by T_C and flip-flop B setup time is represented by TS_{FFB}. Clock period, T_{CLOCK}, is defined by equation 1.1.

$$T_{CLOCK} \geq TP_{FFA} + T_C + TS_{FFB} + (T_{CK}(n)' - T_{CK}(n+1)'') \qquad (1.1)$$

Equation 1.1 represents a lower bound to the clock period. To assure the correct behavior of a synchronous design it is necessary to guarantee that equation 1.1 is going to be respected for any path connecting any two flip-flops in the design. Besides that, it is also required that all delays associated to the combinational and to the sequential logic of the design obey the robustness property (GUNTZEL, 2000) (i.e. all sequential and combinational delays have to be a safe upper bound for the actual delays).

As can be seen in equation 1.1 the clock period has to be larger than the sequential delays plus the combinational delays plus the difference between the clock arrival time at the flip-flops A and B for any clock cycle. Since clock arrival time can change from cycle

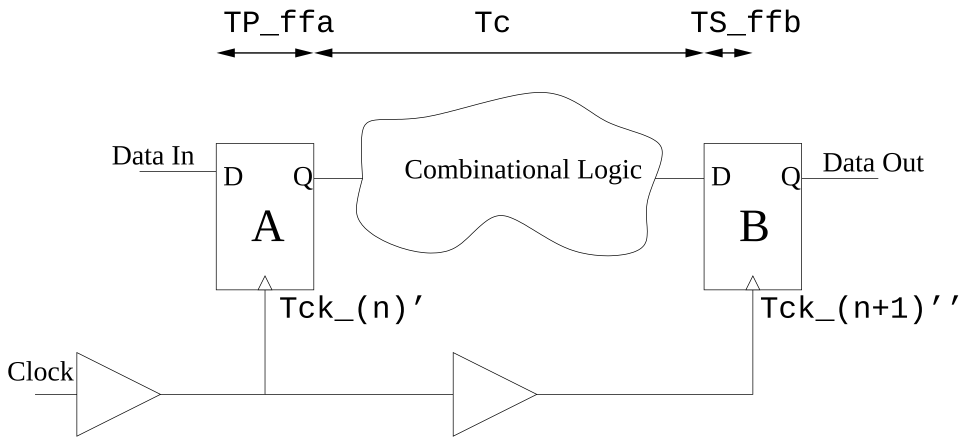


Figure 1.1: Clock period definition.

to cycle due to the effect of environmental variations an upper bound on the maximum difference between the two arrival times has to be considered when defining the clock period. Besides accounting for clock arrival time variations the clock period also has to consider the maximum difference between clock arrival times at any two flip-flops connected by a combinational path.

The maximum difference between all clock arrival times at sequential elements input is called clock skew. As discussed above, in order to assure that data will be ready to be stored when clock edge arrives at a sequential element, it is necessary to account for the clock skew in the clock period definition. Therefore it is important to design a clock network in which clock arrival times are almost the same for all sequential elements, i.e., clock skew is much smaller than clock period.

Clock skew affects not only the clock period definition but also the timing constraints related to fast paths in combinational logic. Fast paths can cause the circuit to fail whenever clock skew is larger than the path delay added to the propagation delay of the input flip-flop and to the hold delay of the output flip-flop. Considering the example illustrated in figure 1.1 the minimum delay allowed to any path connecting flip-flops A and B is defined by equation 1.2 in which TH_{FFB} represents the hold time for flip-flop B. This condition is also known as race condition (WESTE; ESHRAGHIAN, 1985). To assure the correct behavior of the design all race conditions must be satisfied.

$$Tp_C \geq TH_{FFB} - TP_{FFA} + (T_{CK}(n)' - T_{CK}(n+1)'') \tag{1.2}$$

Avoiding race condition is easy since it is necessary only to increase the delay of paths that violate this condition. RESTLE et al. (2001) discusses in more details how to address this problem.

1.1 Definitions

In order to make the comparisons and analysis presented in the next sections clear some important concepts related to the clock signal timing are defined in this section. Section 1.1.1 defines how to compute clock arrival times, delays and transition times. Section 1.1.2 defines the meaning of clock skew and section 1.1.3 defines what clock jitter is. Sections 1.1.4 and 1.1.5 discusses the differences between process variations and environmental variations and their effects on clock timing.

1.1.1 Clock Timing

In this work the clock arrival time at a given node n, $At(n)$, is given as the time when the voltage at n reaches $Vdd/2$ during a transition. Arrival times are measured with respect to the time the simulation starts. Arrival times can be measured during both, rise and fall transitions, in either case arrival times are measured at $Vdd/2$.

Given a circuit element E with i inputs and a single output, the delay of E with respect to the input j, $j \geq 0 \wedge j < i$, is given by the difference between the signal arrival time in the output of e minus the signal arrival time in the input j of e, $D(E,j) = At(E_{out}) - At(E_j)$. Therefore the delay of E, $D(E)$ is defined as $D(E) = max_j D(E,j) = At(E_{out}) - At(E_j)$. Delays can be associated with fall and rise transitions, a fall delay is associated with a falling transition in the output of E, while a rise delay is associated to a rise transition.

Another important timing characteristic of the clock signal is the time it takes during a transition, this is called clock slew. In a falling transition clock slew at a given node n,

$Sl_{FALL}(n)$, is defined as the time when the voltage at n reaches 10% of Vdd minus the time it reaches 90% of Vdd, i.e., $Sl_{FALL}(n) = T_{10\%} - T_{90\%} \quad | \quad V_n(T_{10\%}) = Vdd \times 0.1 \wedge V_n(T_{90\%}) = Vdd \times 0.9$, where $V_n(T)$ is voltage at the node n at the time T. Analogously, the rise slew can be defined as: $Sl_{RISE}(n) = T_{90\%} - T_{10\%} \quad | \quad V_n(T_{90\%}) = Vdd \times 0.9 \wedge V_n(T_{10\%}) = Vdd \times 0.1$. It should be noticed that the rise slew can only be computed during a rise transition while the fall slew can only be computed during a fall transition.

1.1.2 Clock Skew

Considering a set containing n instances $I = (i_0, i_1, ..., i_n)$ and a set with m clock sinks $S = (s_1, s_2, ..., s_m)$, a path, p, can be defined as $p = (E, F)$, where $E \subset I$ and $F = (s_i, s_j) \quad | \quad s_i, s_j \in S$. A circuit C is defined as $C = (p_1, p_2, ..., p_n)$. Let us define the clock propagation time from the clock source to a clock sink $s \in S$ as $D(s)$. Now consider a function $conn(C, s_i, s_j)$ that is defined as:

$$conn(C, s_i, s_j) = \begin{cases} 1 & \text{if } \exists p \Rightarrow (s_i, s_j) \in p \quad | \quad s_i, s_j \in S \\ 0 & \text{if } \neg\exists p \Rightarrow (s_i, s_j) \in p \quad | \quad s_i, s_j \in S \end{cases} \tag{1.3}$$

The clock skew, Sk, for a circuit C can be defined as:

$$Sk(C) = \{max_{s_i, s_j}(|D(s_i) - D(s_j)|) \quad | \quad conn(s_i, s_j) = 1 \vee conn(s_j, s_i) = 1\} \tag{1.4}$$

The definition presented by equation 1.4 means that the clock skew of a circuit is given by the maximum difference at the clock arrival time at any two flip-flops connected through some combinational path. Although the circuit topology is fixed the circuit delays may not be. Since clock skew may change according to process and environmental variations it is necessary to model its effect on clock delays. To account for this variation another concept is used, the concept of clock jitter.

1.1.3 Clock Jitter

Clock signal behavior has become extremely non-deterministic due to the effect of variability sources. The clock jitter concept quantifies the effect of variability on clock timing behavior. Consider that for a given circuit the clock arrival time observed at a clock sink A is described by the histogram *A Clock Arrival Time Histogram* of figure 1.2 while the arrival time observed for a clock sink B is described by histogram *B Clock Arrival Time Histogram*. Comparing both histograms it can be seen that the smallest and largest arrival times observed are the same for A and B but histogram A has a higher occurence of values closer to its mean while histogram B arrival times are further spread away from the mean. If the clock arrival time variation for clock sinks A and B were modeled only by the minimum and maximum values observed the information about how the arrival times are spread between the minimum and maximum values would be lost.

Assuming that the clock arrival time at a given clock sink s is modeled according to a given Gaussian probability density function (PDF) described by the mean and standard deviation pair (μ, σ) the jitter at s, $J(s)$, is defined as $J(s) = 3 \times \sigma$. By assuming that arrival times are Gaussian distributions and describing jitter as three times the standard deviation the information regarding how the arrival times are distributed is preserved and can be precisely retrieved.

Many authors study how to better model the effect of variability sources using different PDFs. The formal definition of jitter presented above is valid in the scope of this

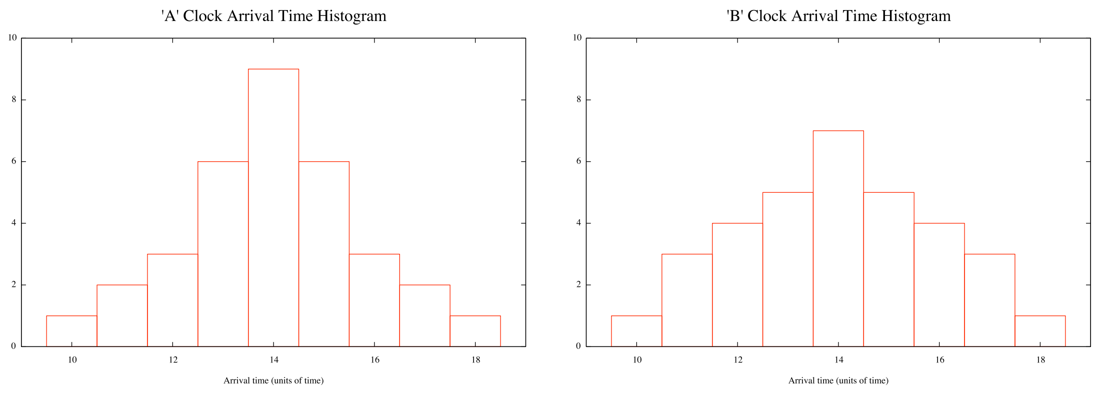


Figure 1.2: Clock arrival time histogram.

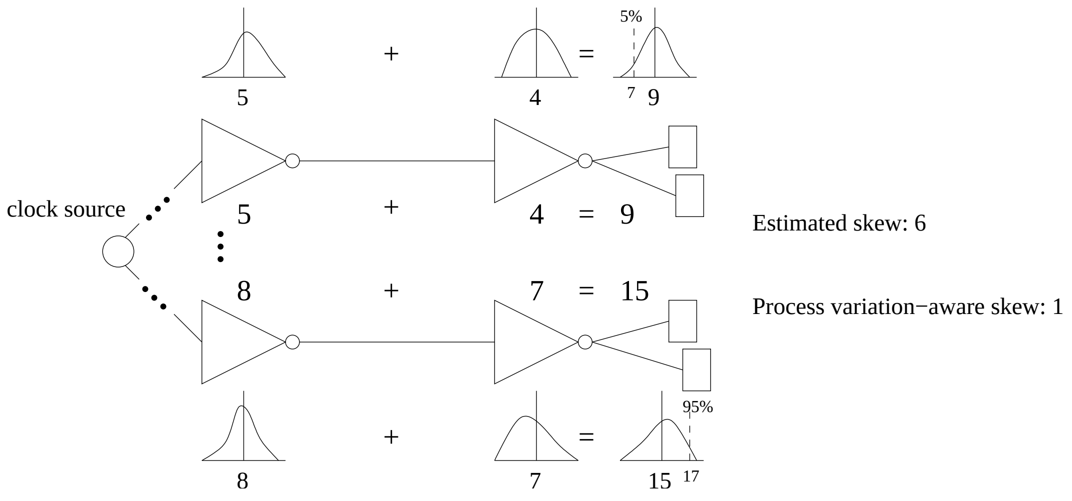


Figure 1.3: SSTA in clock skew computation.

work only. If arrival times are modeled in any other way than by Gaussian distributions the definition of jitter may change. In a broader sense, jitter can be defined as how, and how much the arrival time of a signal changes in respect to time.

1.1.4 Process Variability

Process variability describes any source of variability that affects the performance of a chip due to variations in the fabrication process. The effect of process variability in the performance of ICs has become a major concern for chip designers. If process variations are not taken into account in early design stages final production yield will be low.

Process variability has to be accounted in a statistical fashion. Process variability effects on different circuit parameters are obtained by measuring a set of samples. By knowing the probability distribution for each electrical parameter value, a chip can be designed accounting for the effect of process variability on its electrical behavior.

Figure 1.3 shows the different methodologies used to consider the effect of process variations in the circuit timing affect the final yield. It should be noticed that the values presented in the example above are very rough and its only purpose is to illustrate how process variations should be dealt. By performing a statistical analysis on the delay distribution of each gate for all the paths, it is possible to compute the delay distribution for each path in the clock network. The mean for the delay distribution of a path corresponds to the sum of the mean of the delay distribution for each gate. The mean value means that there is a 50% chance that the actual delay of a path will be smaller than the mean and 50% chance it will be greater than it. Considering the slowest clock path in the clock network, it means that half of fabricated circuits will present a delay smaller than the mean delay for this path. The same is true for the fastest path in the clock network. 75% of fabricated circuits will not properly work because one of these situations will happen:

- The delay of the slowest and the fastest path will be smaller than the mean delay.
- The delay of the slowest and the fastest path will be faster than the mean delay.
- The delay of the slowest path will be smaller than the mean delay and the delay of the fastest path will be faster than the mean delay.

This design strategy represents a large yield loss. One approach to maximize fabrication yield is to replace nominal gate delays by safe lower bounds and upper bounds to compute safe delay estimates. Let us assume that for the slowest instead of using delays of 5 and 4 for the first and second inverters respectively values of 3 and 2 are used. As it was mentioned before it is know that 50% of the times the first inverter will be faster than 5, therefore, if we assume that the delay of the first inverter is 3 maybe only 5% of the times the delay of this inverter will be smaller than 3. The same idea can be applied to the second inverter in the fastest path. If conservative estimates are added up the final path delay is 5 instead of 9 when nominal values are considered. The analogous procedure can be applied to the slowest path. The final delay estimates are safe for a great majority of chips but they are also very conservative and may demand more effort from the chip designer to meet the timing constraints.

To maximize fabrication yield and reduce the timing analysis pessimism the design should be analyzed statistically-wise. Instead of adding up conservative gate delays estimates each path delay should be computed by adding up the PDFs representing each gate delay. In the example presented in figure 1.3, if 7 is assumed to be the delay for the fastest path only 5% of the times this path will present a delay smaller than 7; while for the slowest path if 17 is used instead of 15 only 5% of the times this path will present a delay larger than 17. This means than only 9.75% of the times either the slowest or the fastest paths will present a delay that violates the estimate. Computing PDFs for circuit delays reduces the timing analysis pessimism because it accounts to the fact that if a gate was slowed down it can be compensated by other gates that can get faster due to process variations also. Another advantage is that statistical analysis allows a much better prediction of yield.

1.1.5 Environmental Variability

Environmental variations are variations in the electrical behavior of the circuit elements caused by temperature, crosstalk or voltage variations. This sort of variation can not be treated in the same way as process variations are. Even events with a very low probability may happen since they are a result of the environment variations. Changes in the operating environment can occur at any minute.

Variations in the supply voltage level are the most significant environmental variation effect affecting the clock network. Supply level variations are caused by the IR drop effect. When data switches, capacitances are charged by the VDD network and discharged through GND. In order to charge and discharge the capacitances current flows through the power supply lines. Since power supply lines are not zero resistance lines current flowing through their lines causes voltage to drop (for VDD) or rise (for GND). More details about IR drop effect on clock skew can be found at (SALEH et al., 2000).

It is necessary to be pessimistic when treating environmental variations. This is done by analyzing the circuit at different corners. For a set of environmental parameters E in which $E = (e_0, e_1, ..., e_n)$, each corner is a set of values V_k, where each value corresponds to one environmental parameter. For each set of values (i.e. corner) $V_0, V_1, ..., V_r$, the circuit has to be evaluated. Corner values have to be carefully chosen to guarantee that after the evaluation of all corners the worst case scenario was evaluated.

1.2 Motivation

Design of clock distribution architectures for synchronous digital circuits is an increasing complexity task. As technology advances, scaling allows designing faster combinational blocks. (MEHROTRA; BONING, 2001) shows that process and environmental variations are responsible for increasing clock skew in comparison to clock period. It is shown that clock skew due to variations divided by clock period almost doubles from a 180nm technology to a 50nm technology. Since delay of combinational blocks and delays related to the synchronous circuitry are decreasing, clock frequencies are becoming more dependent on the skew of the clock distribution architecture.

Besides process and environmental variations influence, other aspects contribute to the increasing importance of reducing the clock skew such as:

- *Area increase:* Chip area is rapidly increasing in relation to the transistor dimensions. As the chip area increase more resources are needed to route the clock to all sequential elements (e.g. more buffers need to be added, wire lengths increase). Therefore the clock distribution architecture becomes more sensitive to variations and harder to be tuned.

- *Design complexity:* Increase in the complexity of the designs demands precise engineered clock architectures. New designs often use more than one clock frequency. Macros used in the design usually represent an obstruction to clock lines. As the number of macros increases the complexity to route the clock lines with a small skew also increases.

- *New effects:* Clock frequencies are increasing to gigahertz scale. Some effects that could be safely overlooked for smaller frequencies must be taken into account now (e.g. inductance and transmission line theory (THOMSON; RESTLE; JAMES, 2006)).

Another important aspect in the design of clock architecture is power consumption. (GRONOWSKI et al., 1998) shows that 40% of the total circuit power can be spent on the clock distribution. In more recent microprocessor designs the total amount of power consumed by the clock network is about 25% (ALIMADADI et al., 2008). Power constraints are becoming tighter. Design strategies such as clock meshes or clock spines can distribute a low skew clock signal by the cost of a high power consumption. This sort of approach may not be suitable in a near future where clock power consumption has to be minimal. Solutions for reducing power dissipated by the clock architecture without affecting its performance must be further researched.

Although clock meshes are becoming more present in high performance designs we can observe a lack of optimization and analysis techniques for clock meshes. The current techniques for low power clock distribution designs target at tree-based distribution and can hardly be applied to clock meshes. Besides, clock mesh analysis is very costly due to its long electrical simulations times. In order to develop any optimization technique for clock meshes it is necessary first to develop an accurate and efficient analysis methodology. By reducing clock mesh power consumption and making it easier to be characterized we expect to widen the range of designs in which clock meshes can be applied.

1.3 Thesis Proposal

This thesis presents solutions to analyze and optimize clock meshes. In chapter 2 several design strategies used in clock networks are discussed. In chapter 3, the clock distribution architectures used in the latest microprocessor designs are presented. A comparison between clock meshes and clock trees is also provided. Both studies motivate the use of clock meshes as a way to design variability tolerant clock distributions.

In order to optimize clock meshes we must first analyze them. Chapter 4 offers a simple methodology to enable the analysis of large clock meshes through electrical simulation. Related works are also studied and compared to the proposed methodology.

Two independent optimization techniques are presented in chapter 5. One technique proposes a new design for clock mesh buffers reducing power consumption and improving clock skew by a large factor. The second technique proposes a clock mesh buffer sizing algorithm that improves power and clock skew with a minimum penalty on clock slew. Other clock mesh optimization techniques present in the literature are also presented. However, the optimization techniques proposed in this work are fundamentally different from all other mesh optimization techniques. We propose to optimize the clock mesh considering the timing of the clock network driving the clock mesh while the other methodologies optimize the clock mesh assuming that the clock signal arrives at the clock mesh times perfectly synchronized.

At last, in chapter 6 we present some concluding remarks and discuss about the future directions of this work. In summary, the main contributions of this work are:

- To summarize a clock distribution scheme for microprocessors. A large set of microprocessor clock distribution architectures was studied. The details for each clock distribution scheme were reported in chapter 3. Section 3.3 summarizes the most significant characteristics of each microprocessor clock distribution by describing a generic clock distribution for microprocessors.

- To compare skew mesh-based clock distribution architectures to a pure tree clock distribution. Section 3.2 compares the clock skew and power consumption among a pure mesh architecture, two hybrid architectures and a pure tree clock distribution. This study allows us to notice the effectiveness of clock meshes in reducing clock skew.

- To propose a simple and effective methodology to enable large meshes electrical simulation. Section 4.2 describes the proposed methodology to simulate large clock meshes. This was the first work to address this problem. Related works later proposed are described and discussed in section 4.3.

- To propose two new strategies for clock mesh optimization. Previous work has been done on clock mesh optimization, but current mesh optimization techniques optimize clock meshes assuming that a perfectly synchronized clock signal is applied to the clock mesh. The two mesh optimization strategies described in chapter 5 are the first ones to address the problem of clock mesh optimization considering the different clock arrival times at the mesh buffers.

The work related to the architecture evaluation study presented in section 3.1 and related to the proposed analysis methodology in chapter 4 were developed while the author was on an intership at Fujitsu Laboratories of America and were developed in coperation with other authors. The author of this thesis has worked, more specifically, on the evalution study of the TLM architecture reported in section 3.1.1.3 and on the study of the effect of the border used to increase the accuracy of the SWS methodology reported in section 4.2.3. The main contribution of this thesis relies on the optimization methodologies proposed in chapter 5.

2 CLOCK DESIGN STRATEGIES

Clock distribution has always been an issue for IC designs. Due to this fact, several strategies to address the problem of delivering a high performance clock signal respecting power constraints were developed. In this chapter some of these techniques are presented. The first section presents techniques that tackle at increasing robustness of clock distribution network to noise sources. Section 2.2 presents techniques to reduce clock power consumption. Section 2.3 presents different clock routing topologies. The last section, 2.4, presents architectural level techniques to improve the clock network performance.

2.1 Reliability

Clock is the most important signal in any synchronous design. Any glitch in the clock signal can cause many sequential elements to store corrupted data. Clock designers must guarantee that clock is glitch free. Technology scaling increases design sensitivity to noise source due to the increase in the coupling capacitances and decrease of supply voltage levels. This section address techniques to prevent noise sources to affect the correct behavior of the clock signal.

2.1.1 Shielding

Clock lines must be protected from crosstalk noise. Crosstalk can either speedup or delay the clock signal or even cause a glitch. When two aggressors, the clock wire and a neighbour wire, are switching to the same final value, both signals are going to be sped up. If they switch to opposite values, they will get delayed. When clock is steady, if coupling capacitance is strong enough, a crosstalk aggressor can cause a glitch in the clock wire (victim) as illustrated by figure 2.1.

The best way to protect clock signal from crosstalk aggressors is by shielding it. Shielding relies on adding wires connected to ground or Vdd to protected signal's neighbor tracks. Usually shield wires are added only in the same layer as the protected signal's layer.

Top and bottom layers are usually not shielded. Multi metal layer designs usually adopt a routing strategy in which every metal layer follows a preferred orientation, except for metal 1 layer. If metal 2 allows only vertical wires metal 3 will allow only horizontal wires, in such a way there will be no neighbor layers following the same orientation. Coupling capacitance between nets on different metal layers is minimal since they are not running in the same orientation. Coupling capacitance between metal 1 and metal 2 layers is also minimal. Although metal 1 allows wires to be added with any orientation those wires are very short since metal 1 layer is used only for internal cell connections.

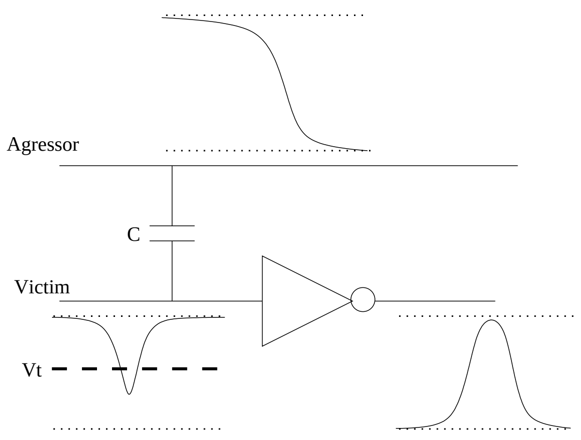


Figure 2.1: Glitch caused by crosstalk noise.

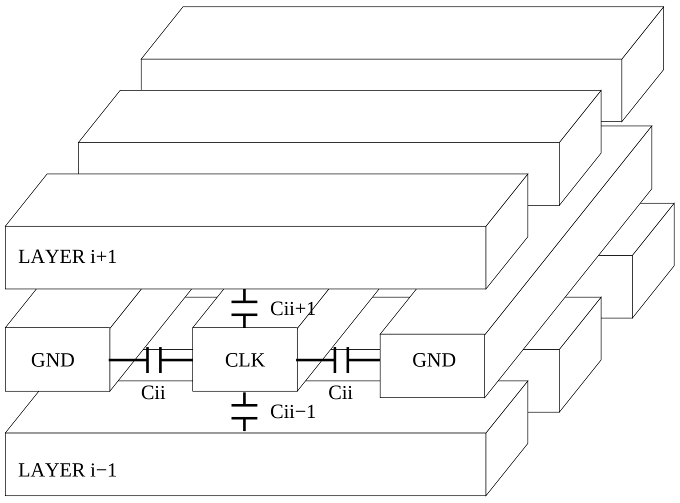


Figure 2.2: Routing management for different metal layers.

Figure 2.2 shows how the routing layers orientation management affects parasitic capacitances. The intersection between wires on different layers is minimal. Coupling capacitance between nets on different layers is not important for crosstalk effects. Unless all aggressors switch in the same direction at the same time they will not affect a victim on a different layer. Neighbor wires on the same layer can have a large coupling capacitance since they can be side by side for a long distance. For these reasons, the capacitance Cii illustrated in figure 2.2 is much larger than $Cii \pm 1$ and therefore shielding can be performed only within the same layer. If shielding were performed also on top and bottom layers, the routability on those layers would be severely affected.

Shielding is usually applied on the higher branches of clock networks. It can lead to a huge resource utilization penalty if applied to the whole clock network.

2.1.2 Differential Signaling

Differential signaling relies on sending a signal through the voltage difference in a pair of wires. This approach protects the signal against crosstalk and allows the signal to be transmitted using a reduced voltage swing. The differential pair is routed side by side. The differential signal needs to be converted back to a single ended signal before reaching the flip-flops. Usually only the higher branches of the clock network are protected by

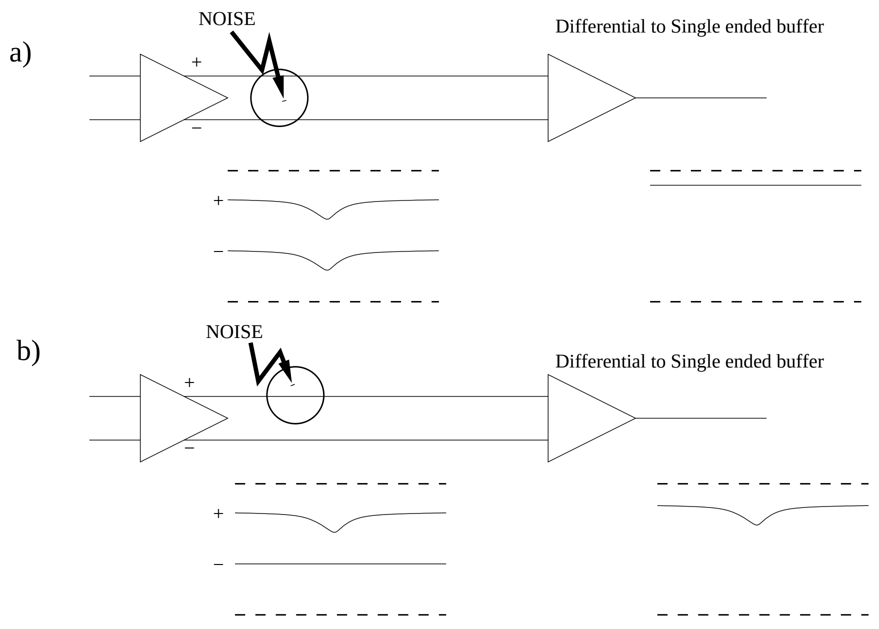

a)
NOISE
Differential to Single ended buffer
+
−
b)
NOISE
Differential to Single ended buffer
+
−

Figure 2.3: Differential signaling noise immunity.

this technique since each sink of the protected portion of the clock networks requires a differential to single ended converter. The closer to the flip-flops a differential signal is taken higher is the number of converters required, increasing the overhead associated to this technique.

By encoding the information in the voltage difference of a pair of wires any noise source affecting both wires of the differential pair would be filtered. Only when a single wire of the differential pair is affected noise can be observed. Figure 2.3 illustrates both situations.

Shielding is still desired since any aggressor in the same layer would affect mostly a single wire in the pair. The increased protection against noise allows the voltage swing to be reduced, reducing power consumption. This technique does not necessarly improves power since differential voltage repeater and differential to single ended converters consume more power than an inverter.

2.2 Low Power

Keeping clock power consumption within its budget is an increasing complexity task. The clock frequency increase linearly increases clock power consumption. At the same time, electronic market demand for low power products is pushing ASIC power consumption down. For many current designs, power constraints have become more important than timing constraints. This section presents two techniques used to reduce clock power consumption.

2.2.1 Clock Gating

Clock gating consists in freezing the clock signal for regions of the chip that are not being used. Regions where clock is frozen are said to be on sleep mode. When clock is not switching dynamic power consumption is reduced to zero since no transitions occur in these regions. Clock signal can be set either to zero or one in sleeping regions. All regions in sleep mode are unable to process any data. Sleeping regions are able to restore

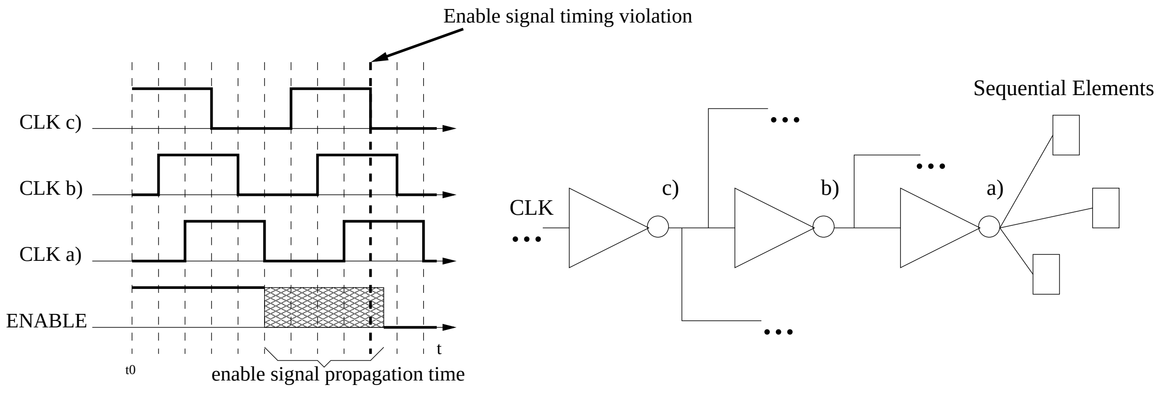

Enable signal timing violation
CLK c)
CLK b)
CLK a)
ENABLE
t0
t
enable signal propagation time
Sequential Elements
CLK
c)
b)
a)

Figure 2.4: Enable signal timing issues.

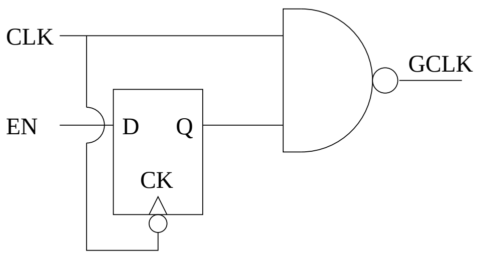


Figure 2.5: Clock gater design.

all information stored in sequential elements after exiting from sleep mode.

Since a large part of dynamic power consumption comes from the clock network itself, gating clock close to the clock root saves more power than gating it close to the clock sinks. It should be noticed that enable signal timing must be respected when deciding in which stage clock is going to be gated. The closer to the root clock signal is gated, shorter is the time for enable logic to be stable. Figure 2.4 demonstrates how moving clock gaters towards the clock root compromises enable signal timing. In this example clock gaters can not be added above stage c) since enable signal would only be captured in the next clock cycle from this point on.

Besides respecting timing constraint, clock gater cells must be glitch free. Enable signal glitches should not propagate to clock lines, since clock glitches cause the circuit to fail. A possible way to prevent enable signal glitches to propagate through the clock gater is by adding a negative level triggered latch as illustrated by figure 2.5. When clock is at level $'0'$ the gater output is set to level $'1'$. When clock is at level $'1'$ the gater output will be determined by the value stored in the latch.

2.2.2 Reduced Swing

One effective way to reduce clock network power consumption is by reducing capacitance charge/discharge power consumption. Equation 2.1 shows how capacitance charge/discharge power is computed

$$P = f \times C_L \times Vdd \times Vs \tag{2.1}$$

where f is the switching frequency, C_L is the load capacitance, Vdd is the supply voltage and Vs is the output swing of the buffers.

The most effective way to reduce power consumption according to equation 2.1 is by reducing Vdd, since Vs is a fraction of Vdd and most often $Vs = Vdd$. By reducing Vdd dynamic power consumption is reduced quadratically. Dynamic power consumption could be reduced in a linear fashion by reducing only Vs.

Changing supply voltage and voltage swings for all elements in a chip would heavily affect timing characteristic. A better approach is to change Vdd and Vs only for the clock distribution network. Since clock sinks are not going to be affected by the voltage reduction it is necessary to convert clock back to the standard voltage swing before sinks are reached. (PANGJUN; SAPATNEKAR, 2002) and (IGARASHI et al., 1997) assume that the best approach to minimize clock power is to design most of the clock network within the low power region, i.e., voltage swing is reduced at clock root and restored only before reaching clock sinks. This solution is optimal if power consumption at voltage converters is equivalent to a single inverter power consumption. Adding voltage swing converters in the last stage of the clock distribution maximizes area and power overhead introduced by voltage swing converters since the last level of the clock distribution requires more drivers than any other level of the clock network.

As discussed above, there are two distinct ways of reducing voltage swings in the clock network. It can be reduced either by reducing Vdd for all the elements in the clock network or only by reducing the voltage swing without changing Vdd. Although using different vdds, $Vddh$ and $Vddl$, for the clock network and for the rest of the chip can save more power, it adds design complexity since another power signal must be distributed over the chip and low Vdd clock cells can only be placed in the regions where $Vddl$ is available. It should also be noticed that reducing voltage swing of any signal makes it more sensitive to noise.

2.2.2.1 Multiple Supply Voltages

Using multiple supply voltages allow a low power consumption in low Vdd regions. Low Vdd regions power consumption is reduced quadratically with respect to the Vdd reduction. Assuming that a region that was initially connected to $Vddh$ is now connected to $Vddl$, where $Vddl = 0.9 \times Vddh$, the dynamic power reduction in this region should be in the order of 0.9^2 (i. e. 19% reduction from a 10% Vdd reduction).

The design of a $Vddh$ to $Vddl$ converter is straightforward, it consists of a regular inverter supplied by $Vddl$. $Vddl$ buffers are regular inverters in which VT is adjusted to the new supply voltage values. The design of the $Vddl$ to $Vddh$ converter is more complex. Its design is illustrated by figure 2.6. This approach was used in (IGARASHI et al., 1997).

2.2.2.2 Reduced Voltage Swing

Conversion from a full swing signal to a reduced swing signal is done by a reduced swing driver. In order to prevent huge delays introduced by interconnection RC, reduced swing buffers are required. Reduced swing buffers receive a reduced swing signal in its input and transmit a reduced swing signal in the output. Since clock sinks require a full swing signal, a reduced swing receiver is required to convert clock signal from a reduced swing back to a full swing.

Figure 2.7 presents the design of all the elements required by the reduced swing clock scheme. The reduced swing driver illustrated in the figure was presented in (HANAFI et al., 1992), the reduced swing receiver was presented in (ZHANG; RABAEY, 1998) and the reduced swing buffer was proposed in (PANGJUN; SAPATNEKAR, 2002).

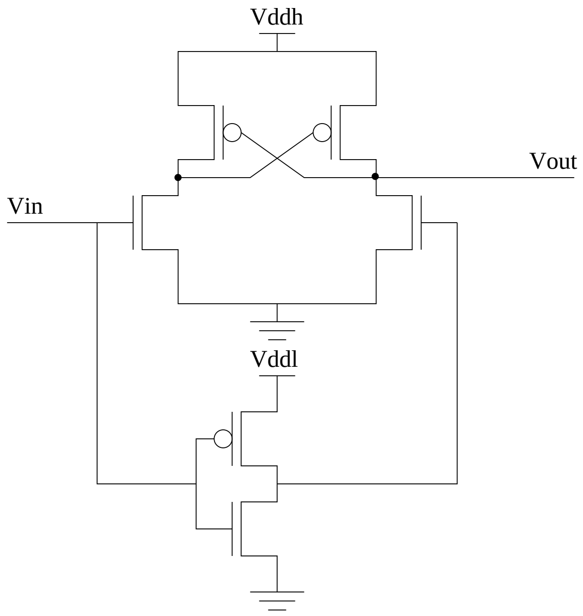


Figure 2.6: $Vddl$ to $Vddh$ converter.

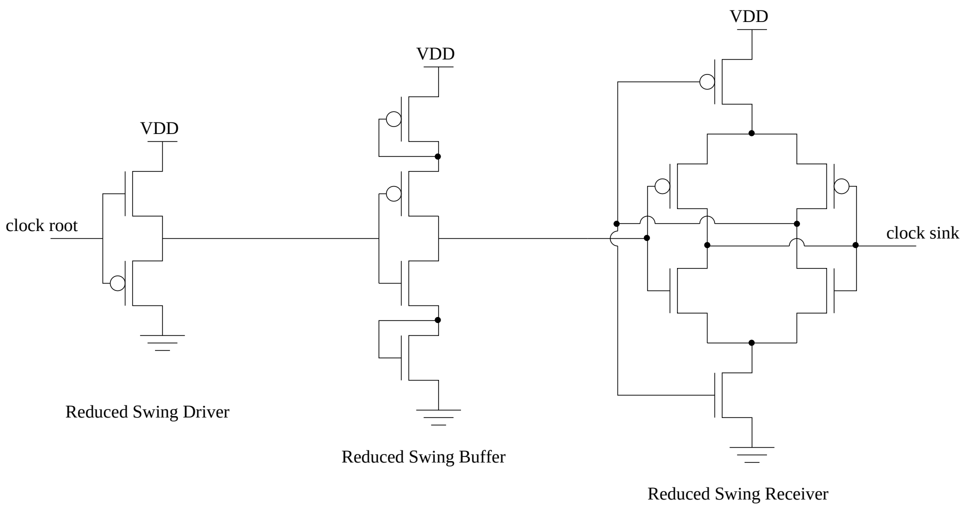


Figure 2.7: Reduced swing driver, buffer and receiver.

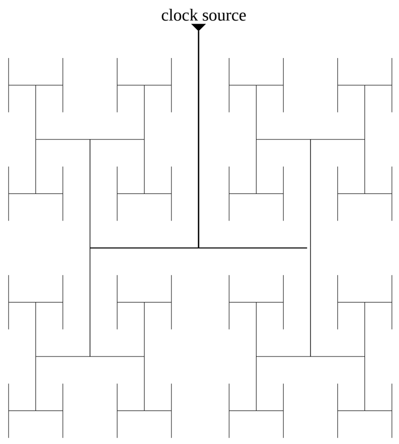


Figure 2.8: Htree example.

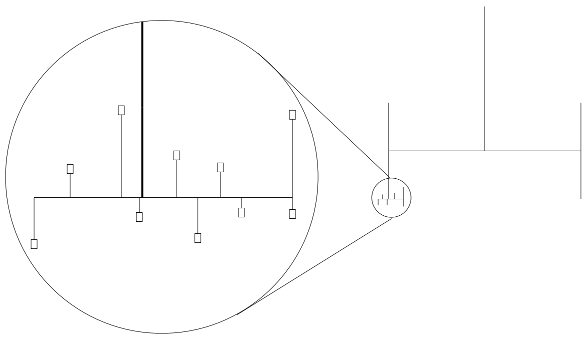


Figure 2.9: Fishbone routing connecting clock sinks to htree sink.

2.3 Routing Topologies

Clock skew, power consumption and tolerance to variations is extremely dependent on the clock routing. Clock routing has the complex task of equalizing the delays from the clock source to each clock sink. At the same time, the longer is the clock routing the higher the power consumption, clock skew and sensitivity to variations are going to be. Usually different routing strategies are used in different levels of the clock distribution. Each routing strategy presents advantages and disadvantages. The routing strategy has to be selected according to the constraints of each design. This chapter presents five of the most commonly used routing strategies and discusses the advantages and disadvantages of each one.

2.3.1 Htree

An htree is a symmetric tree in which wire length from any sink to the root is the same. Figure 2.8 is an illustration of an htree topology. This figure shows a topology in which the clock signal is driven from a central location to multiple clock sinks. Since the clock pin may not be located in the center of the chip it is necessary to route the clock from the clock pin to the center of the htree.

The total number of sinks in a htree is usually much less than the total number of clock sinks connected to it. Clock sinks are directly connected to the htree sinks using a fishbone structure, as shown by figure 2.9.

An htree necessarily presents a homogeneous sink distribution in the X and Y axis. Htree can be used to drive the clock signal directly to the flip-flops or to the inputs of a

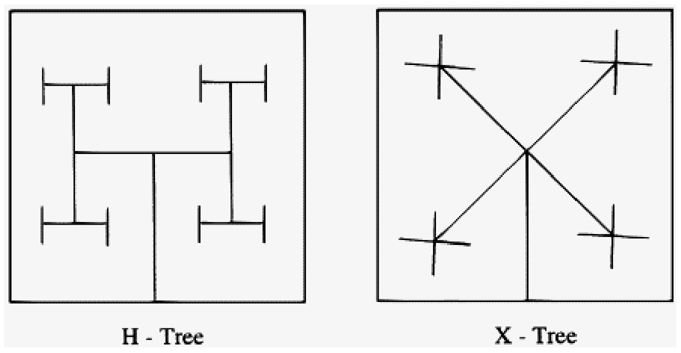


Figure 2.10: Htree vs xtree example (FRIEDMAN, 2001).

mesh. Although wire lengths are equalized by the htree structure, buffers must be carefully inserted and sized in order to keep skew small. Wire widths can also be changed either to compensate different loads driven by each branch or to satisfy electro-migration rules. In both cases the larger the load driven is larger the wire width should be.

Htree is highly vulnerable to process and environmental variations since variations may unbalance the delays on the different branches of the htree. Htrees are most often applied to ASICs due to its performance limitations. Still, some microprocessors claim to use a clocking scheme based on htrees without using clock meshes, such as, (ANDERSON; WELLS; BERTA, 2002) and (TAM; LIMAYE; DESAI, 2004).

2.3.2 Xtree

The xtree architecture is analogous to the htree architecture. Both, xtree and htree present the same wire length from the root to any sink, the difference between them is that the xtree uses 45 degree connections, as shown by figure 2.10. This architecture can be found in the Alpha 1.2GHz microprocessor (JAIN et al., 2001).

The main advantage offered by this architecture compared to the htree is the reduction of total wire length due to 45 degree connections. The wire length reduction comes from the fact that in a square shape with a side length equal to s, the diagonal length (45 degree line) is given by $s \times \sqrt{(2)}$ while the Manhattan distance between the opposite corners is given by $s \times 2$. By reducing the total wire length a smaller power consumption and smaller clock skew are expected to be achieved.

2.3.3 Clock Routing

Clock net requires a very special sort of routing to minimize clock skew. Instead of reducing wire lengths clock routing should try to match, as close as possible, latencies from the root to all sinks. A simple way to do that is by using patterns to equalize the wire length from the clock root to all sinks (e.g. htree and xtree).

Htrees are very easy to build but it presents two major drawbacks, the wire length overhead and the mismatch between htree sinks locations and clock sinks locations. An htree distributes the clock signal to a symmetrical array of buffers that may not match the actual clock sink locations. Extra routing must be added to connect clock sinks to htree sinks, which may increase clock skew.

This section presents two methods to route the clock network from the clock root to the clock sinks with close to zero skew and reduced wire length.

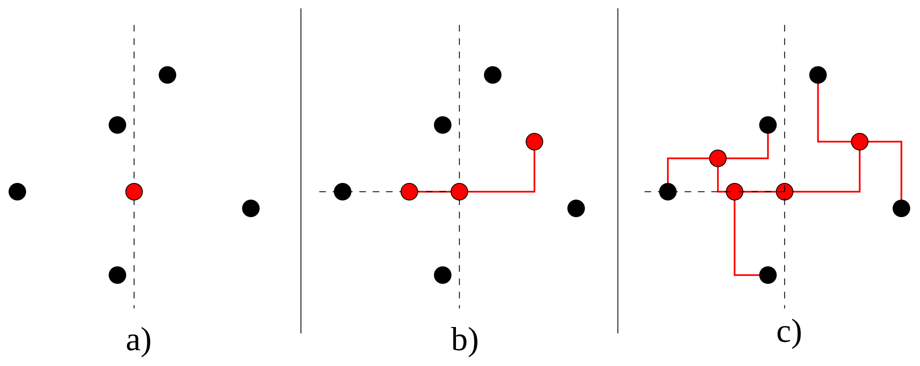


Figure 2.11: MMM algorithm example.

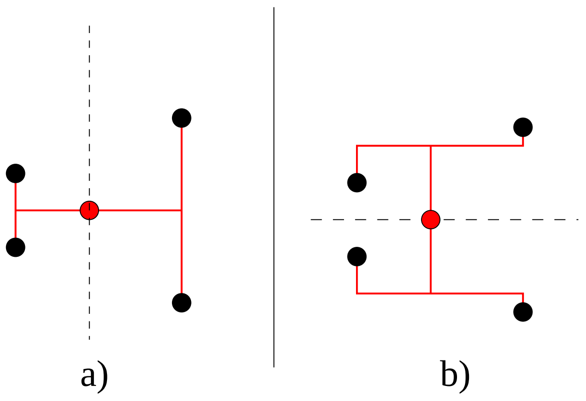


Figure 2.12: Clock tree with a a) vertical cut and b) horizontal cut.

2.3.3.1 Method of Mean and Medians (MMM)

The method of mean and medians (MMM) was firstly presented in (JACKSON; SRINIVASAN; KUH, 1990). It can greatly reduce clock skew in comparison to a minimum spanning tree routing and it is also better than an htree for asymmetric distributions of clock sinks.

The idea of this algorithm is conceptually simple. Given a distribution of clock sinks, the center of mass of this distribution is computed. The distribution is then divided into two parts by a line crossing at the center of mass either horizontally or vertically. The centers of mass for the two new sink distributions are computed and then connected to the center of mass of the former distribution. This algorithm is executed recursively until each sink distribution is composed by a single sink.

Figure 2.11 illustrates an example of how the algorithm works. In a) the distribution is divided vertically by a line crossing the center of mass. In b) the center of masses for the two new distributions are computed. The centers of mass of the new distributions are connected to the center of mass of the former distribution. The distribution on the left was divided horizontally. The final routing is shown in c).

Deciding whether a set of sinks is going to be divided vertically or horizontally is an important step in this algorithm. Figure 2.12 shows how performing a vertical or a horizontal cut can produce different clock routings. The author in (JACKSON; SRINIVASAN; KUH, 1990) proposes a one level look-ahead strategy to decide which cut should be performed. A horizontal cut followed by a vertical cut is performed, then a vertical cut followed by a horizontal cut is performed. The cut direction that produces the smallest clock skew is chosen.

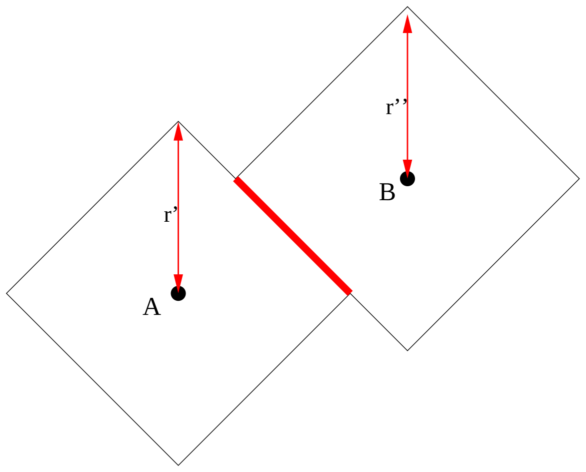


Figure 2.13: Construction of a merging segment.

This algorithm present a $O(n \log n)$ complexity, where n is the number of sinks in the clock distribution.

2.3.3.2 Deferred-Merge Embedding (DME)

The deferred-merge embedding (DME) algorithm is able to generate a zero skew clock tree with minimum wire length. It was proposed in (BOESE; KAHNG, 1992) and in the following years many improvements were proposed to this algorithm. This algorithm requires the clock network topology to be previously defined. It finds the optimal routing for the defined topology.

The DME algorithm is divided into two phases, a bottom up phase in which the location of the internal nodes in the clock network are replaced by lines which represent all possible locations, and a top-down phase in which the clock root is fixed and all the internal node locations are fixed thereafter.

Figure 2.13 shows how a merging segment is constructed when two sinks are merged. If wire lengths need to be matched the merging segment is computed by the intersection of the Manhattan circles with radius r' and r'', where r' equals to r'' which is equal to half of the Manhattan distance between nodes A and B. The same process can be applied when, instead of clock sinks, two segments are merged. In this case, the radius of each Manhattan circle is given by the *minimum* Manhattan distance between both segments.

After all the internal node positions were deferred and merged, the position of the clock root is embedded. When the position of a node is fixed the merging segments connected to that node are going to be restricted by this node location. Figure 2.14 illustrates how the set of possible positions to a node is restricted when a position is embedded for its parent. Segment C was built from the merging of segment A and B. When position of C is chosen to be the black dot, the possible positions for A and B are restricted.

The DME algorithm can be modified to, instead of equalizing wire lengths, equalize Elmore Delay values. This algorithm presents a linear complexity in terms of number of nodes in the clock network.

2.3.4 Clock Spine

A clock spine is a wire, usually wide, used to take the clock signal from a clock driver across the chip in one dimension. It can be used to deliver the clock to the root of one

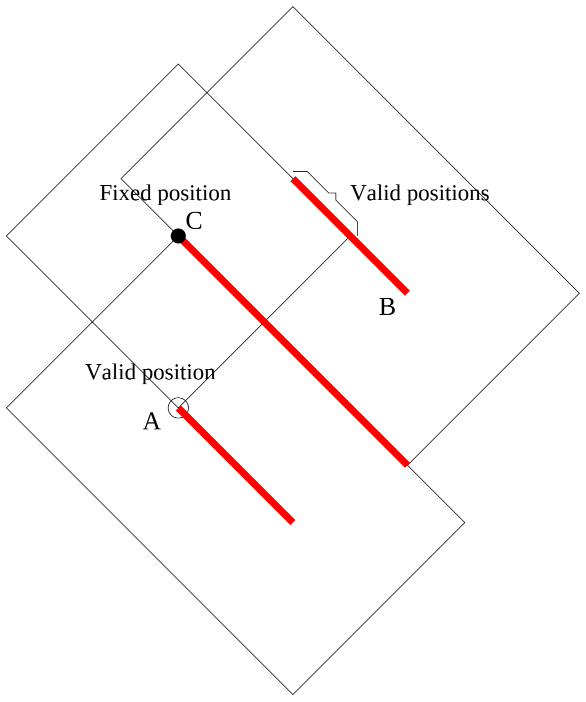


Figure 2.14: Position embedding.

or several local clock trees. Clock spines are a simplification of a clock mesh, it can be described as a one dimensional clock mesh. Processors such as Intel's Pentium III (SENTHINATHAN et al., 1999) and Pentium 4 (KURD et al., 2001) (KURD et al., 2001) use clock spines.

In the design of the clock distribution for the Pentium 4 microprocessor (KURD et al., 2001)(KURD et al., 2001) three clock spines are used. At each clock spine a different binary tree is connected and each binary tree drives a different clock domain.

Figure 2.15 illustrates the three clock spines used in the Pentium 4 design. The clock spines are represented by the white lines crossing the chip in a west-east fashion. Clock spines present a small skew due to the low resistance of its lines. By adding a low skew clock trunk the distance between any clock sink and the clock source is reduced. The total

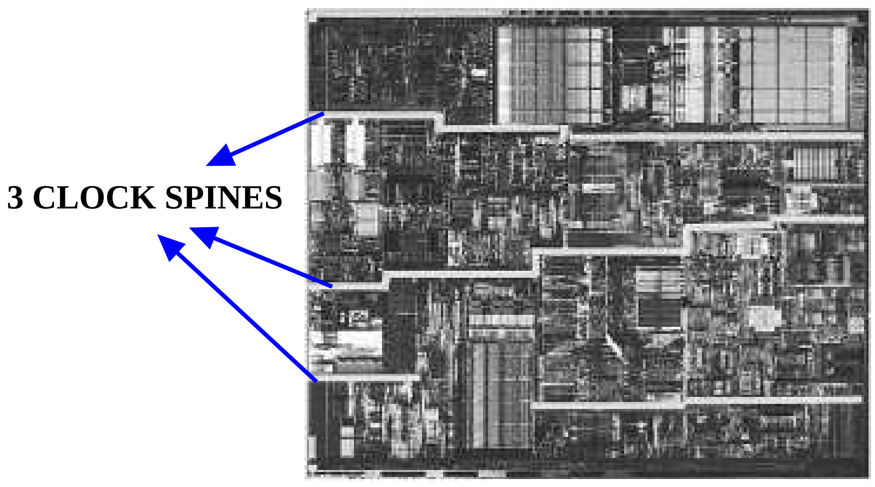


Figure 2.15: Pentium4 Clock Spines. (KURD et al., 2001)

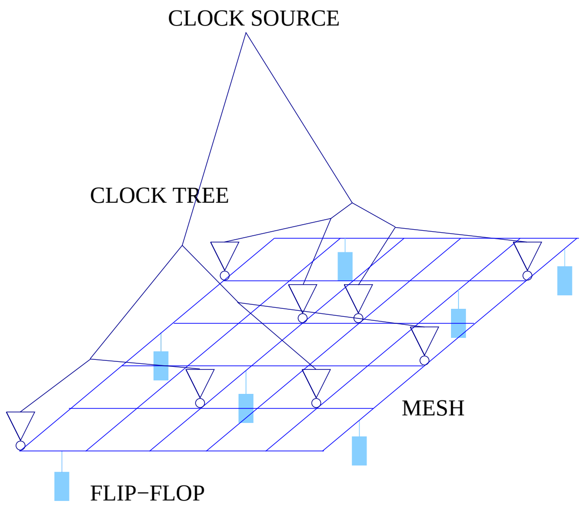


Figure 2.16: Mesh architecture example.

clock skew is also smaller.

Clock spines are not tree-like topologies since it adds cycles to the clock network. Power consumption may be in the same order as a clock mesh with the same number of drivers.

2.3.5 Clock Mesh

A mesh is a grid composed by wires to which the sequential elements are directly connected. Figure 2.16 illustrates a mesh being driven by a clock source and some elements connected to the mesh wires. Meshes are widely used in the design of the clock distribution for microprocessors (BAILEY; BENSCHNEIDER, 1998), (TAM; LIMAYE; DESAI, 2004), (KURD et al., 2001), (TAM et al., 2000). Reconvergent paths created by the mesh structure are able to smooth out the difference between the clock signal arrival times at the mesh inputs. Since reconvergent paths may produce short circuit currents between the mesh drivers, they are, along with the high capacitance associated with the mesh wire structure, responsible for the higher power consumption in comparison to tree-like clock networks power consumption.

Clock meshes are usually represented as a regular and homogeneously distributed set of vertical and horizontal wires. Figure 2.17 presents the clock mesh designed for a 600-MHz Alpha processor (BAILEY; BENSCHNEIDER, 1998) which shows that meshes are not always regular and homogeneous. The mesh wire density can be tuned to reduce the skew over the most critical regions in a chip.

Mesh buffers are inserted at the mesh grid nodes (i.e. the connection between a vertical and an horizontal line). Mesh performance and power consumption are highly related to the characteristics of mesh buffers. A large number of mesh buffers usually means a high performance and high power consumption. The most straightforward approach to mesh buffer insertion relies on inserting a mesh buffer on every mesh grid node. Mesh buffers can be sized according to any fanout rule, the only constraint for a good performance is to use the same sizing rule to all mesh buffers in a mesh, so that mesh buffer delays are equalized.

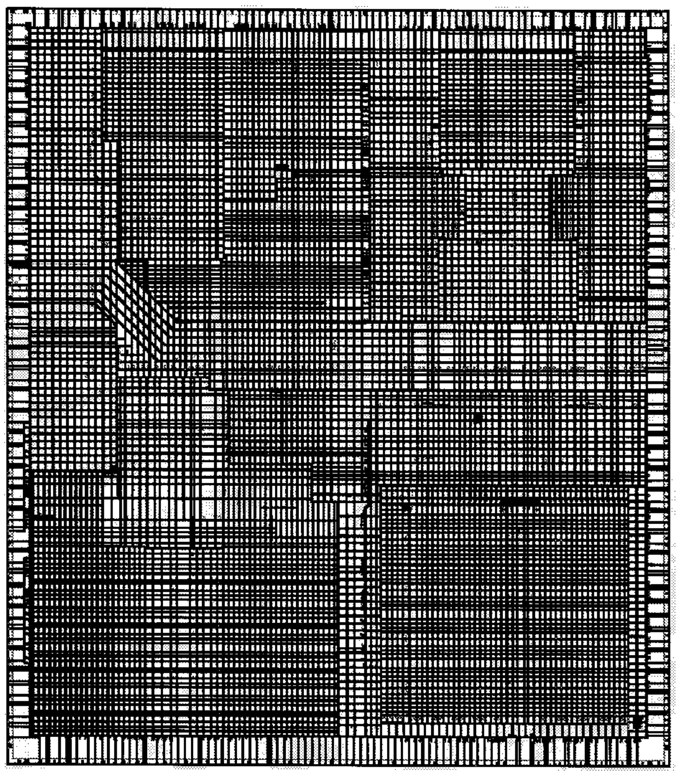


Figure 2.17: Mesh for 600-MHz Alpha Microprocessor. (BAILEY; BENSCHNEIDER, 1998)

2.4 Architectural Strategies

This section discusses strategies to plan the design of the clock network in a higher level. This is done by dividing the clock network into stages and domains. The idea of this methodology is to provide high performance only where it is required. A design to improve the performance between different clock domains is also presented in this section also.

2.4.1 Clock Domains

When a single clock signal is distributed, clock domain definition is related to the regions within which clock signal requires a higher synchronization. Hierarchy present in chip designs demands a very small skew within the same functional block, while constraints on the clock signal are usually more relaxed regarding the synchronization between two different functional blocks.

A low skew clock signal within a functional block is usually achieved using clock meshes. Synchronization between two distinct blocks is done by using a balanced clock tree and by applying some deskew methodology as presented in section 2.4.2.

Figure 2.18 illustrates an example of a design containing multiple clock domains. In this figure, the clock signal is driven through a tree-like (i.e. no loops) clock distribution architecture until different domains are reached. A deskew buffer compensates different arrival times at the sinks of the top level distribution. Clock is then driven from the deskew buffers to flip-flops through another tree-like structure. A clock mesh is added in the sinks of each domain to compensate for inter-domain skew.

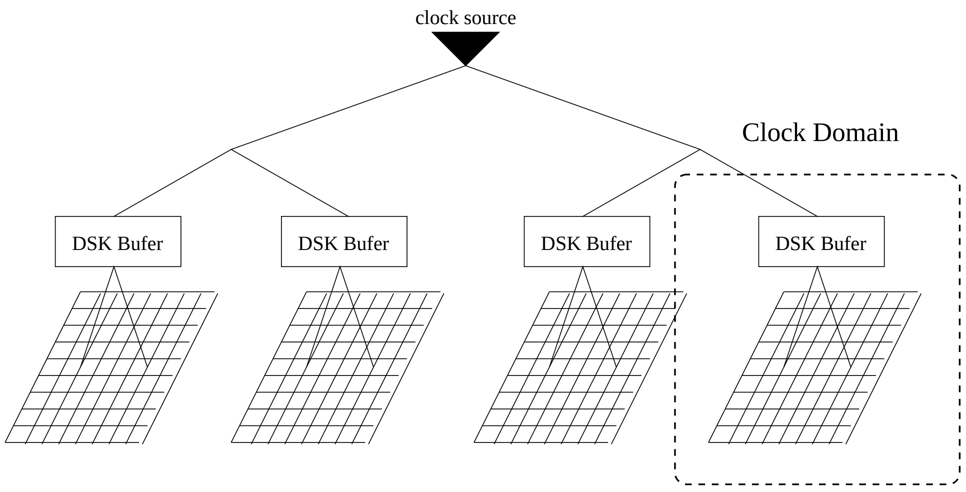


Figure 2.18: Clock Domain Definition.

2.4.2 Deskew

Reduced clock skew values are often achieved by using balanced clock trees, applying load matching techniques using dummy devices or by increasing or reducing the length and width of clock lines. None of these techniques is able to compensate skew caused by process variations since it is not possible to predict the actual effect of process variations on the electrical characteristics of the circuit. To account for the effect of process variations during a local path tuning would require a post-fabrication analysis of process variations effects over the clock distribution.

Deskewing design methodologies tackle at post-fabrication tuning of the clock distribution. Deskewing process must be automatic or semi-automatic otherwise it would become impractical. Existent techniques can be divided in active techniques or fuse-based techniques. The first group refers to approaches that are constantly calibrating the delay of the clock structure while the former refers to approaches where a single calibration is performed after the circuit fabricated.

Deskewing methodologies are widely used in microprocessor designs (TAM et al., 2000), (KURD et al., 2001) and (TAM; LIMAYE; DESAI, 2004). The deskew process is performed using a variable delay buffer, which may be calibrated according to the process variations influence on the chip design. In (GEANNOPOULOS; DAI, 1998) a variable delay buffer is proposed. Figure 2.19 illustrates the proposed buffer. The clock signal is delayed by two inverters on whose outputs a variable load is connected. The load connected to the output of each inverter is controlled by transmission gates connected to a PMOS and a NMOS transistors. According to the values stored in the Delay Control Register a different set of capacitances will be connected to the output of each inverter. The loads should be equally distributed between the first and the second inverters in order to equalize the duty cycle and fall/rise delays. In (GEANNOPOULOS; DAI, 1998) ten stages of load are used in the output of each inverter. Loads are controlled by a 20 bit register, in which the logic value ’1’ represents that the load is connected to the output of the inverter.

Deskew is usually performed between different clock domains. Within a single domain, clock signal is deskewed by a clock mesh. Deskew buffers are the only alternative available today in the literature to smooth out process variations effect on the skew between two different clock domains. The number of deskew buffer is proportional to the

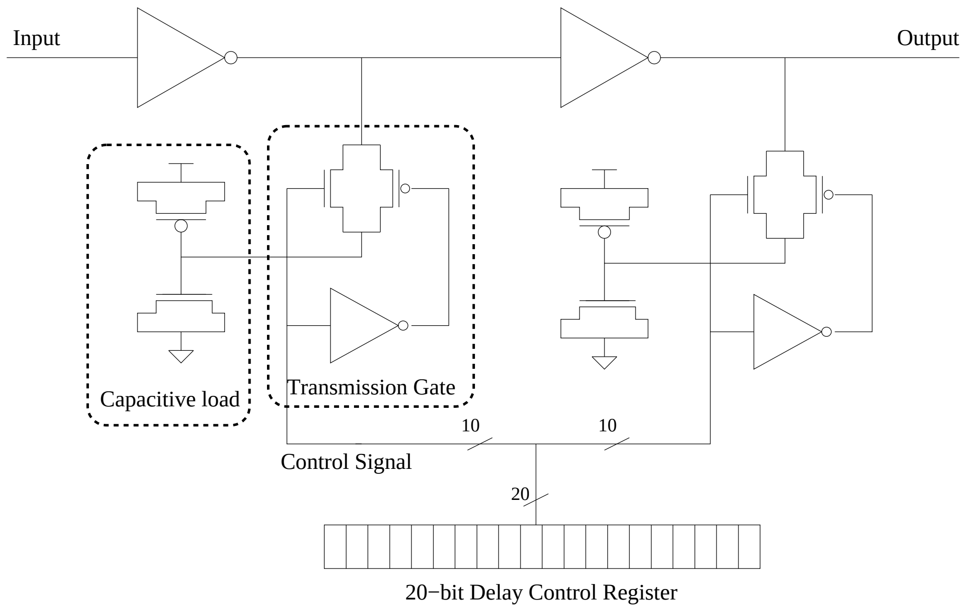


Figure 2.19: Variable delay clock buffer.

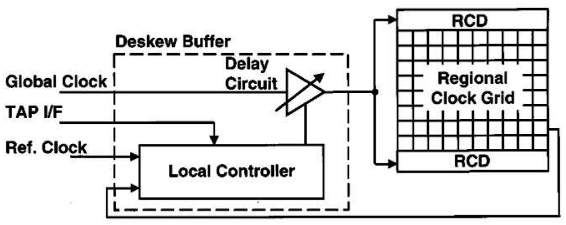


Figure 2.20: Active deskew scheme. (TAM et al., 2000)

number of clock domains.

2.4.2.1 Active deskew

Figure 2.20 illustrates the active deskew scheme. The clock signal on the mesh lines is compared to a reference clock. The phase difference between both is computed by the local controller and a new control signal is generated and passed to the variable delay buffer.

The phase detection is done according to the circuit represented in figure 2.21. The phase difference between both clock signals is detected by the phase detector block. During the enable signal generated by a counter block, the phase difference is forwarded to a digital low-pass filter. The low-pass filter removes any phase comparison noise. In the circuit presented in figure 2.21 the variable delay buffer is updated at every 16 clock cycles.

2.4.2.2 Fuse-based deskew

In a fuse-based approach, tuning of variable delay structures is performed only once. The 20-bit delay controller is configured by fuses. The benefits of the fuse-based deskew methodology in comparison to an active approach rely on the simplicity of implementation. By configuring a single time the delays at variable delays buffers, it is not necessary to include in the circuit the phase detection and correction circuitry.

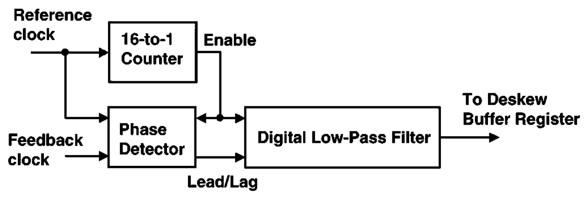


Figure 2.21: Adjustable delay block controller. (TAM et al., 2000)

The fuse-based methodology has been presented in (TAM; LIMAYE; DESAI, 2004) and it was used in the design of the Itanium 2® microprocessor.

3 CLOCK ARCHITECTURES REVIEW

This chapter presents a study on the impact of using different clock distribution architectures and optimization techniques on the final clock distribution performance and power consumption. Section 3.1 presents a study comparing a mesh-based clock distribution scheme to a tree-based clock distribution. Section 3.2 presents a bibliographic study about the clock architecture of several microprocessors. The design strategies used to achieve the high performance required without degrading power consumption are discussed. On section 3.3 a general clock distribution scheme for microprocessors derived from the bibliographic study is presented.

3.1 Clock Distribution Architectures: A Comparative Study

Chapter 2 has presented different strategies for the design of clock networks and discussed its advantages and disadvantages. This section presents a detailed comparison based on electrical simulation experiments between different clock architectures. The focus of this comparison is to study the design trade-offs between tree-based and mesh-based clock distribution architectures. This work was previously published in (YEH et al., 2006). This work was developed in comperation with other authors, the contribution of the author of this thesis in this study was in the evaluation of the Tree + Local Meshes architecture.

3.1.1 Target Architectures

We have investigated four different clock distribution architectures, a single mesh architecture, a pure tree architecture and two hybrid approaches mixing tree and meshes. A brief description of each architecture is given below.

3.1.1.1 Mesh

A single mesh architecture is an architecture that has a global clock tree driving a clock mesh to which sequential elements are directly connected. This architecture is explained in section 2.3.5. In this study the clock meshes were characterized by their size, $m \times n$, where m is the number of rows and n is the number of columns.

3.1.1.2 Tree

A pure tree clock distribution can use an htree, and xtree or a specific routing algorithm to distribute the clock from a source to the clock sinks. In this study an htree routing, as described in section 2.3.1, is assumed.

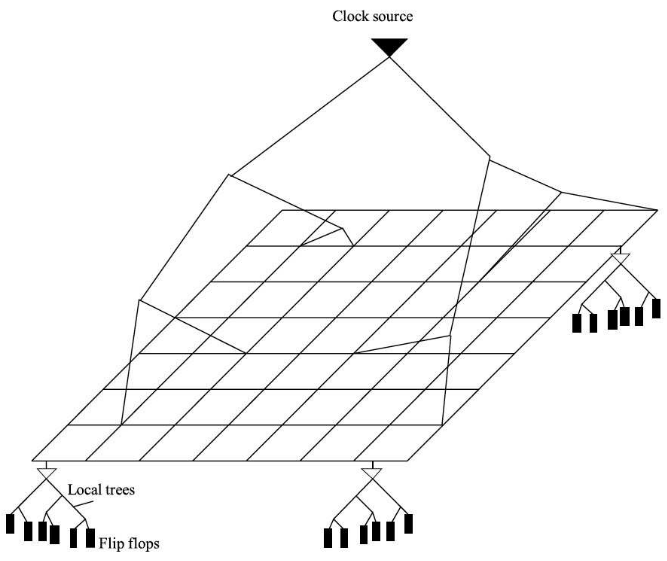


Figure 3.1: MLT architecture example. (YEH et al., 2006)

3.1.1.3 Hybrid

Two hybrid configurations were evaluated.

1. *Mesh + Local Trees (MLT)*: A single clock mesh driven by a global tree is used to drive the clock signal to the different regions of the chip. Connected to the clock mesh local clock trees are used to drive the clock signal from the mesh to the clock sinks. A simpler version of this architecture was studied in (SU; SAPATNEKAR, 2001). This architecture is illustrated in figure 3.1.

2. *Tree + Local Meshes (TLM)*: In the TLM architecture the clock sinks are divided into different domains. A single clock tree is adopted for the global distribution. Each clock sink domain is driven by a different clock mesh to which the clock sinks are directly connected. Figure 3.2 represents this architecture. More details about this architecture can be found in (WILKE; MURGAI, 2007).

Although more hybrid architectures could be evaluated we believe that focusing our study in those two architecture is enough to understand the design trade-offs related to the clock distribution choices.

3.1.2 Target Chip Specification

During this evaluation study we have used three benchmark circuits, $D1$, $D2$ and $D3$, to perform our experiments. $D1$ and $D2$ are dummy designs while $D3$ is an actual industrial design. Table 3.1 summarizes the characteristics of each benchmark circuit. All three circuits were designed using Fujitsu's $11\mu m$ technology. The nominal supply voltage used was $1.2V$. The experiments were simulated in the nominal temperature of 55^oC.

For our experiments we have extracted the actual location of each flip-flop in the design. The clock network model was generated assuming that there were no placement or routing obstructions. A single clock domain was assumed also. We have modeled the clock network wires using metal 6 and metal 7 for the global clock tree and for the clock

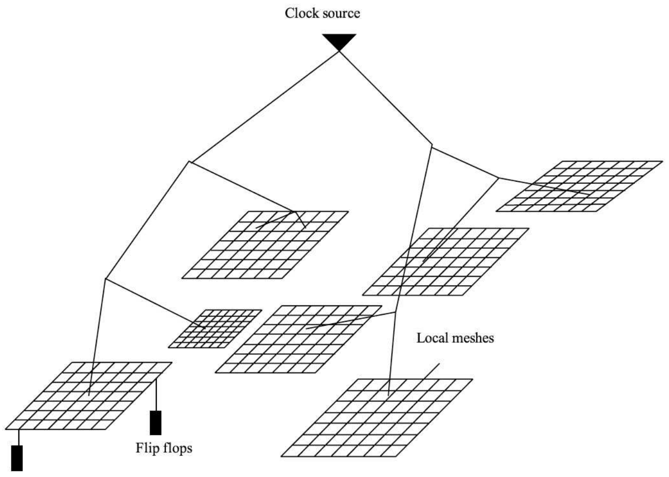


Figure 3.2: TLM architecture example. (YEH et al., 2006)

Table 3.1: Test chip statistics.

Circuit	#gates	#FFs	area (mm^2)	FF-spanned area (mm^2)
$D1$	536.5K	16.75K	5×10	0.8×6.67
$D2$	1016.6K	39.16K	5×10	2.23×9.62
$D3$	7659.6K	287.39K	16×16	12.03×14.63

mesh and using metal 1 to metal 4 to model the local connections. The clock source was assumed to be in the center of the chip.

We have imposed a maximum slew constraint of 15% of the clock period, in this case a clock frequency of $1GHz$ was selected, therefore the maximum slew allowed is $150ps$. An electromigration constraint was imposed limiting the maximum current flowing through a wire with a given width. This constraint was derived from the technology specifications. The target skew for our clock network is $0ps$.

3.1.3 Experimental Set-Up

Each of the target architectures was evaluated through electrical simulation. The electrical model for wires was derived from a sample layout; capacitances values were extracted using Calibre xrc; resistances were calculated from technology specifications, and inductance values were estimated using Raphael.

It was assumed that the clock wires have parallel two-sided shielding. It was also assumed that all the tracks crossing the clock wire in the above and below metal layers were occupied. This assumption can be fulfilled by inserting fill-in metal in empty tracks. Accurate inductance computation is enabled by the ground shield running next to the clock wire.

For each architecture we have developed software for designing the clock distribution network using the technology information (e.g., capacitance, resistance and inductance values per unit lenght) and clock design rules. The software accepts certain parameters from the user. For instance, for the mesh architecture, in addition to the chip dimensions, flip-flop locations and technology information, the designer supplies mesh size, technol-

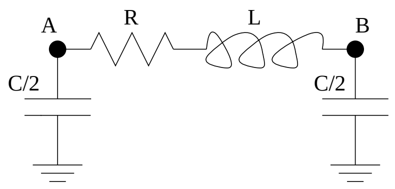


Figure 3.3: Single-π model for interconnect.

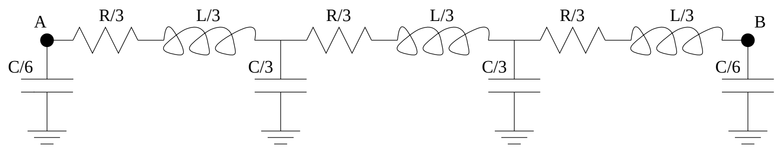


Figure 3.4: 3-π model for interconnect.

ogy rules (e.g., value of l for interconnect model, as described below) and design rules (e.g., mesh buffer sizing rule). We performed experiments with several values of these parameters and determined the best values before comparing with other architectures. For the given technology, we also derived rules for optimum buffer sizing and spacing to minimize latency and power. These rules are used in synthesizing a clock network that has close to optimum latency and power.

In general, the intent in the synthesis tool was not to generate absolutely the best clock network with minimum latency, skew and power by optimizing the topology, wire widths and buffer sizes and locations, since this can be a huge undertaking. Instead, generating close to the best network sufficed, since common features shared by different architectures (such as the global tree) are synthesized using the same algorithm, which is sufficient for our comparative study.

We also developed analysis software that generates SPICE netlists for the clock network, runs circuit simulators HSPICE and HSIM (Synopsys) on these netlists for timing analysis, and reports latency and skew values for the FFs. To generate the SPICE netlists, we used accurate models of buffers and interconnect in the clock network. For interconnect with length less than l = 100μm, we use a single-π RLC model (figure 3.3. Otherwise, we use a 3-π model, as shown in Figure 3.4. Such a rule was shown to have less than 0.5% delay error as compared to a golden 4-π or 5-π model (WILKE, 2004).

We evaluate architectures using the following metrics.

1. Clock latency: Latency is the time taken by the clock to arrive at a FF from the root. We would like to minimize latency, since it has a direct impact on timing uncertainty and jitter.

2. Maximum skew: The difference between the maximum and minimum latency over all the FFs. Minimizing the maximum skew is important, since in a fixed clock cycle, it limits the maximum delay in a path.

3. Maximum timing uncertainty: The clock timing uncertainty is defined as the deviation of the clock edge timing at FFs from the expected or nominal value due to parameter variations. As described in Section 3.1.2, our analysis incorporates the following sources of variations: Process (P) variations, supply voltage (V) variations, temperature (T) gradients, and crosstalk noise (X).

4. Power consumption: We use ${CV_{dd}}^2 f$ to compute the power dissipated in the clock network, where C is the capacitance of the clock network, Vdd is the power supply, and f is the clock frequency. This computation ignores the short circuit power dissipation in the clock mesh. The short circuit power in the mesh should be negligible, otherwise mesh short circuit power should be considered. Power dissipated in the clock network is also used as an indicator of area resources used in the clock network, i.e., device and wire areas.

3.1.3.1 Mesh

A htree was used to drive the clock mesh in our experiments. The mesh and htree buffers were sized using the fanout 4 rule (FO4) (SUTHERLAND; SPROULL, 1991), i.e., to drive a capacitive load C, a buffer with input capacitance $C/4$ is used. This rule was found to yield close to optimum delay/mm and power for a stage (using the optimization feature of HSPICE). The optimum distance between buffers/repeaters in the htree was also determined using HSPICE optimization feature. Mesh buffers are assumed to be inserted at every mesh node, i.e., mesh buffers are inserted in the intersection between vertical and horizontal lines.

Cock meshes were build on the smallest rectangular area which contains all FFs. Details of the mesh areas are shown in table 3.1, column FF-spanned area. It can be seen that for $D1$, this area is only about 11% of the entire design area, whereas for $D2$ and $D3$, this ratio is 43% and 69% respectively.

3.1.3.2 Tree

The tree topology chosen to be evaluated is composed by an htree followed by a fishbone structure to which the flip-flops connect directly, as described in section 2.3.1. As happens with the mesh architecture, the htree also spans only the smallest rectangle in the chip that contains all the flip-flops.

3.1.3.3 Mesh + Local Trees

The MLT architecture was derived from the single mesh architecture. A global htree is used to drive the clock signal to a clock mesh to which unbuffered clock trees are connected. The local tree clock routing was performed using the MMM algorithm presented on section 2.3.3.1.

3.1.3.4 Tree + Local Meshes

The TLM methodology relies on assembling individual clock meshes for each of the different clock domains in such a way that each clock mesh can be powered off according to the sleep signal logic of each domain. Since circuits $D1$ and $D2$ are small, TLM methodology was applied only for circuit $D3$. Clock domains were artificially created since no blocks in $D3$ presented sleep mode functionality. $D3$ was partitioned into seven different clock domains. Information about the flip-flop density and area of each partition can be found in table 3.2.

A htree was used to drive all the clock meshes. The htree can not be perfectly aligned to the different clock meshes, therefore htree sinks are not aligned to the mesh grid nodes. Clock sinks are directly connected to the closest mesh wire in each partition.

Table 3.2: TLM partition information.

Partition	#FFs	Area (mm^2)	#FFs/mm^2
1	51.5K	22.63	2281.88
2	51.7K	23.87	2165.06
3	21.0K	33.64	623.34
4	28.4K	35.38	802.2
5	30.1K	17.98	1674.92
6	51.6K	26.28	1964.84
7	53.0K	27.72	1910.86
total	287.4K	256.00	1122.66

3.1.4 Analysis

Each one of the tested configurations was evaluated through electrical simulation. The Sliding Window Scheme (SWS) decribed in section 4.2 was used to enable the accurate electrical simulation of large meshes. The methodology relies on splitting the simulation of a large mesh into several smaller simulation tasks by sweeping an accurate region window inside which circuit elements are accurately modeled. Elements outside the accurate region are lumped, reducing drastically the total number of elements in the mesh model.

Variations effect was evaluated by estimating the clock jitter. If the clock network is a tree, uncertainty analysis can be carried out using gate-level statistical static timing analysis as shown by (BERKELAAR, 1997), (VISWESWARIAH et al., 2004) or (AGARWAL; BLAAUW; ZOLOTOV, 2003). However, such an approach is not directly applicable for a mesh-based clock network due to metal loops (cycles) present in the mesh. One solution is that if the mesh model fits in the memory, we can run Monte Carlo (MC) simulations (HITCHCOCK, 1988) assuming some distribution for parameter variations and obtain a delay distribution at each FF, from which timing uncertainties at FFs could be derived. This is possible only for small design and mesh instances. A study on the effects of using the SWS to perform MC simulation is presented in (REDDY; WILKE; MURGAI, 2006). To compare uncertainties in tree and mesh architectures, we use MC simulation on small design and mesh instances.

We model various sources of uncertainty. Supply noise is modeled by supplying independent power supplies to each clock buffer, and allowing them to vary randomly according to a noise model. Temperature variation of transistors is modeled by specifying an underlying temperature for the entire chip and then applying random local temperature variations on each clock buffer and interconnect. Transistors process variations are modeled using only channel length and threshold voltage variations. Other variations, such as oxide thickness and dopant concentration have the overall effect of varying the threshold voltage and hence are indirectly included in our model. Interconnect variations are modeled through resistance and capacitance variations. The 3σ variations we used in our study for various parameters are shown in table 3.3.

3.1.5 Results

Table 3.4 report the capacitance distribution among each stage of the mesh architecture. Some interesting observations can be made on this table.

Table 3.3: 3σ variations for different parameters.

Parameters	3σ variations
NMOS/PMOS channel length	0.008μm
NMOS/PMOS threshold voltage	20mV
Interconnect resistance	20%
Interconnect capacitance	5%
Temperature	20C
Vdd	10%
Crosstalk switching probability	0.5

Table 3.4: Capacitance distribution (%) for mesh architecture.

chip	mesh size	wires				buffers			FFs
		mesh	local	htree	total	mesh	htree	total	
$D1$	16×16	17.9	19.5	17.0	54.4	14.5	10.5	25.0	20.6
	64×64	32.6	2.3	30.8	65.7	11.1	13.9	25.0	9.3
	128×128	35.6	0.6	33.7	69.9	10.4	14.7	25.0	5.1
$D2$	64×64	28.8	7.9	26.1	62.8	12.2	12.8	25.0	12.2
	128×128	34.3	2.4	31.1	67.7	11.0	14.0	25.0	7.3
$D3$	64×64	11.7	38.1	9.1	58.9	16.5	8.5	25.0	16.1
	128×128	22.2	20.3	17.2	59.8	14.4	10.8	25.0	15.2

1. From 60% to 70% of the total clock capacitance is in the wires.

2. Buffer capacitance is always 25% of the total clock capacitance. This is a consequence of the FO4 rule applied to size htree and mesh buffers. A buffer with input capacitance $C/4$ is used. To drive that, a buffer of input capacitance of $C/16$ is used (assuming, for simplification, no wiring capacitance on the tree), and so on. The total capacitance in the clock network is $(C + C/4 + C/16 + ...) = 4C/3$, whereas the buffer capacitance is $(C/4 + C/16 + ...) = C/3$. Thus, the buffer capacitance is 25% of the total clock capacitance.

3. The mesh wiring capacitance for smaller designs is a large fraction (30-35%) of the total clock capacitance. However, for the largest design $D3$, it is 10-20%, thus making mesh more attractive for larger designs.

4. When mesh granularity increases the mesh wires get closer to the flip-flops, reducing capacitance due to local wiring.

5. Local wiring capacitance is much larger in $D3$ than in $D1$ and $D2$. This is due to the much larger number of flip-flops and the larger area of design $D3$.

Table 3.5 provides a comparison between the mesh architecture and the tree architecture regarding timing and power characteristics. An operating frequency of 400MHz was assumed to compute dissipated power. We make the following observations.

Table 3.5: Mesh architecture vs. tree architecture.

chip	mesh/ htree size	max skew (ps)		max latency (ps)		max slew (ps)		capacitance (pF)		power (W)	
		mesh	tree	mesh	tree	mesh	tree	mesh	tree	mesh	tree
$D1$	16×16	3.3	29.1	480.6	499.1	89.5	89.5	255.5	231.0	0.15	0.13
	64×64	0.1	1.1	636.9	638.4	89.6	89.6	563.2	396.3	0.32	0.23
	128×128	0.1	1.3	717.5	718.9	89.6	89.6	1030.2	603.7	0.59	0.35
$D2$	64×64	0.5	4.2	674.2	678.2	105.7	105.7	1008.1	721.4	0.58	0.41
	128×128	0.2	1.4	754.3	756.1	105.1	105.1	1694.5	1048.4	0.97	0.60
$D3$	64×64	4.8	29.3	975.1	995.0	110.9	108.6	5570.1	5084.0	3.2	2.9
	128×128	0.9	5.5	1050.8	1055.0	110.2	75.2	5885.2	4892.2	3.4	2.8

1. In all the three designs, a very low skew value can be achieved with mesh. Increasing the mesh granularity from 64×64 to 128×128 reduces the skew, for instance, from $5ps$ to below $1ps$ for $D3$. For the tree architecture, $D3$ presents a maximum skew of 30ps using a htree with 64×64 sinks and 5.5ps using a htree with 128×128 sinks. Although these numbers are small for the tree architecture, they are worse than the corresponding skew numbers for the mesh architecture.

2. The skew observed for an htree was from 5× to 10× larger than the skew observed when a clock mesh was added to the htree sinks. Since the global clock distribution for a clock mesh is also done by an htree we can conclude that *the clock mesh is able to reduce the clock skew by a factor of 5× to 10×, i.e., the skew applied in the input of the mesh buffers is 5× to 10× larger than the skew observed at the clock sinks.*

3. Maximum FF latency values for tree and mesh are almost identical. Latency increases with increase in mesh size and #htree sinks due to higher total clock capacitance. Note that the maximum skew is a very small fraction of the maximum latency. This implies that our clock network design methodology is effective for low skew for both mesh and tree architectures.

4. As expected, capacitance and power values for the tree are smaller than the mesh architecture. For $D1$ and $D2$, the tree values are about 30-40% smaller. For $D3$, the difference is only 10-20%.

5. Maximum slew values are similar for tree and mesh architectures and within the maximum limit of $150ps$.

The MLT architecture was compared to a mesh architecture using benchmark circuit $D3$. Maximum clock skew, slew, capacitance and power are reported in table 3.6. A 64×64 mesh size was chosen in this comparison. It can be seen that the best skew achieved by the MLT architecture is much worse than the skew achieved by the mesh architecture where clock sinks are directly connected to the mesh wires. It turns out that this is because in some local regions there are hundreds of FFs. MMM algorithm does not insert buffers on local trees. Although mesh buffers are sized according to their loads, it

Table 3.6: Comparing Mesh and MLT architectures.

	Max Skew (ps)	Max Slew (ps)	Capacitance (pF)	Power (W)
Mesh	4.77	110.9	5570	3.2
MLT	58.51	191.0	6076	3.5

Table 3.7: TLM architecture evaluation.

Partition	Mesh Size	Max Skew (ps)	Max Latency (ps)	Max Slew (ps)
1	32×32	2.41	1073.3	71.85
2	32×32	2.55	1073.6	72.08
3	32×32	3.39	1074.3	72.24
4	32×32	4.87	1075.7	73.72
5	32×32	2.09	1073.3	71.39
6	32×32	3.36	1074.3	72.54
7	32×32	3.77	1074.9	72.78
all	87×87	4.87	1075.7	73.72

may be difficult to have good slew rates when such a high load is to be driven. The maximum slew of 191ps (which violates our maximum slew constraint of 150ps) also confirms this. The results for MLT can be improved if buffer insertion capability is introduced in the local trees.

The TLM architecture was also evaluated using benchmark circuit $D3$. Maximum skew, slew and latencies measured are reported in table 3.7. A global htree with 128×128 sinks, with each of the seven meshes having size 32×32. The skew within each partition is small, the maximum skew being 4.87ps for the partition 4. It turns out that the maximum skew for the entire circuit is also 4.87ps, since the FFs with maximum and minimum latencies are both in partition 4. To compare with a single mesh architecture, in row all, we report the effective mesh size, the maximum skew and the maximum slew values for the entire chip. The effective mesh size is computed from the combined mesh square count of all the seven meshes. The maximum skew is slightly worse than when there is only one mesh. The latency values for TLM are similar to pure mesh (table 3.5).

3.1.5.1 Uncertainty Analysis

An experiment was made to evaluate the mesh capability of reducing input jitter caused by variations. The variations shown in table 3.3 were applied to each mesh wire. Two mesh sizes were used: 8×8 and 16×16 for a chip size of 500μm×500μm for both meshes. 1000 flip-flops were randonly distributed over the chip area and connected to the closest mesh node. Clock signal arrival time at the input of each mesh buffer was modeled as a Gaussian distribution where the mean arrival time is 0ps and the $3 \times \sigma$ jitter values varies from 3ps to 150ps. 1000 Montecarlo simulations were performed in each case. The worst output jitter is reported in table 3.8 for both mesh sizes. We can see that the 8×8 can reduce jitter by a factor of 7 to 8 times. The 16×16 mesh can reduce jitter by a factor of 2 vis-a-vis to a 8×8 mesh, i.e., the total reduction ranges from 14 to 16 times.

Table 3.8: Reduction of uncertainty by mesh.

Input uncertainty (ps)	Max. output uncertainty (ps)	
	8×8 mesh	16×16 mesh
3	0.39	0.19
15	1.89	0.95
30	3.78	1.89
150	19.8	9.9

Another experiment was performed simulating a Global htree + Mesh architecture. A $5mm\times5mm$ chip with 1000 FFs randonly placed was used. The clock architecture was generated and variations reported in table 3.3 were applied to the htree and mesh wires and buffers. Jitter was measured at the mesh buffers and at the FFs. The maximum jitter measured at the mesh buffer was $43.08ps$ while the maximum jitter observed at the FFs was $35.52ps$, a 18% reduction. In this case the jitter reduction observed was much less than the 7 to 8 times factor reported in table 3.8. This is due to the following reasons:

1. When the htree was included in the simulations the correlations between the arrival times that share many htree paths were considered. We can not assume that arrival times at the mesh buffers are uncorelated.

2. Mean arrival time clock skew at the mesh buffers is not zero. Different loading unbalance htree delays generating skew.

Those two facts reduce the ability of the clock in compensating early and late arrival times at the mesh buffers. When correlations are considered neighbouring buffers may have similar arrival times, therefore they can not compensate each other. Mean arrival time clock skew is also bad for jitter reduction since it increases the difference between the arrival times at the mesh buffers.

3.1.5.2 Closing Remarks

This study has shown that clock meshes are extremelly important in the design of low skew clock distribution architectures. Besides of reducing nominal skew clock meshes are effective in mitigating clock jitter. Reducing the granularity of a clock mesh increases its skew reduction factor.

We have also evaluated variations of mesh-based architectures such as the MLT and the TLM architecture. We have shown that the TLM architecture can enable clock gating capabilities with a small skew penalty. The MLT architecture was shown to be inefficient unless buffers are inserted in the local clock tree.

Considering the increasing importance of designing variations-aware clock distributions this study has shown that clock meshes are an important tool to achieve this goal.

3.2 Microprocessor Clock Distribution Bibliographic Study

3.2.1 Pentium 4 (2000)

The Pentium 4 processor (HINTON et al., 2001) is an IA-32 processor containing 42 million transistors in a $217mm^2$ area. The first version of the clock distribution for

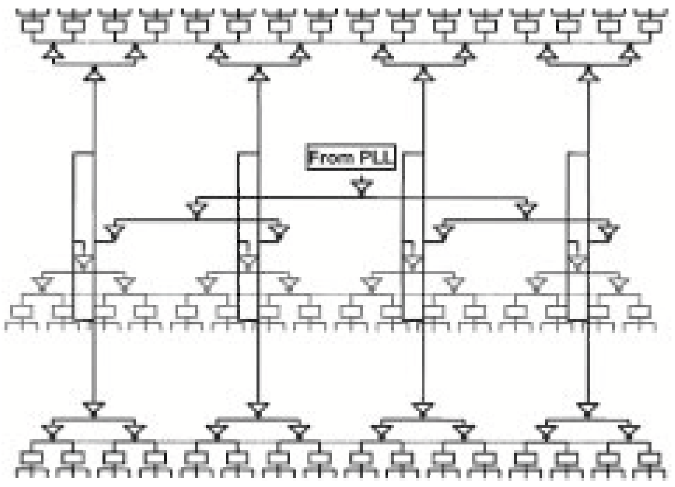


Figure 3.5: Clock tree driving Pentium4 spines. (KURD et al., 2001)

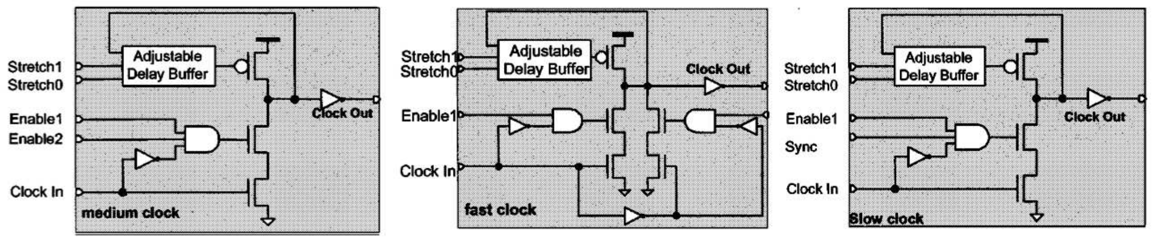


Figure 3.6: Pentium4 Local Clock Drivers (LCDs). (KURD et al., 2001)

this processor operated at frequencies higher than 2GHz (KURD et al., 2001). Another version, with a sub-10ps was presented in (BINDAL et al., 2003).

The initial clock distribution for the Pentium 4 (KURD et al., 2001) processor uses three clock spines to distribute clock at global level as shown in figure 2.15. Each clock spine is driven by a generic tree and each clock spine drives a set of binary trees. Figure 3.5 shows this architecture. Jitter reduction is obtained by filtering the clock buffers power supply and shielding clock wires. An active deskew mechanism was implemented to reduce the skew between different domains. Clock signal is deskewed at each binary tree root by an adjustable delay domain buffer. This design presents 47 independent clock domains therefore 47 adjustable delay buffers are used.

Local clock drivers (LCD) are located on the leaves of the domain binary trees. LCDs are responsible to multiply and divide the base clock frequency to achieve 1GHz and 4GHz respectively. LCDs are also responsible to gate the clock signal and adjust the duty cycle. The circuitry used in LCDs is illustrated in figure 3.6. Clock gating capability is implemented by an AND gate connected to the NMOS transistor in the pull down logic. LCDs work different from a regular inverter, the pull-down path connects the output to ground at each rising edge of the clock signal for a short period of time. The adjustable delay buffer is responsible to connect the output to vdd, therefore it controls the cycle. The driving strength is given by an output buffer. Clock frequency division by 2 is done using clock gating logic and a global synchronization signal. Frequency multiplication by 2 is achieved by using two pull-down paths, one enabled at the rising edge and the other enabled at falling edge.

In (BINDAL et al., 2003) the clock distribution design for Pentium 4 microprocessor

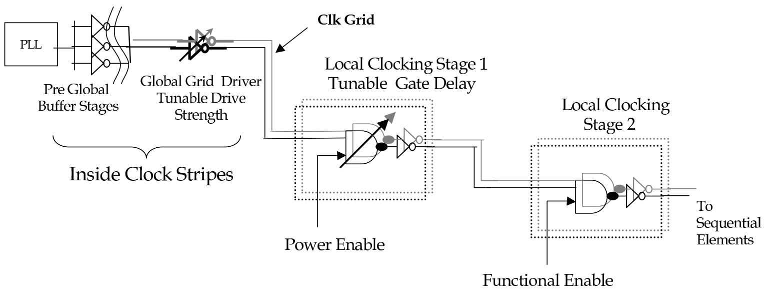


Figure 3.7: Pentium 4 clock distribution scheme. (BINDAL et al., 2003)

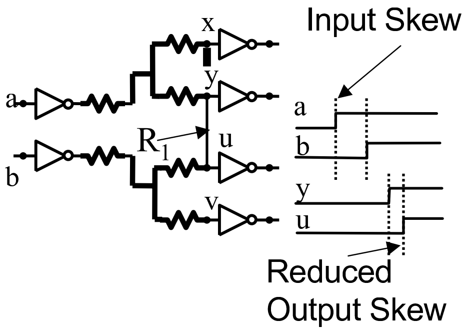


Figure 3.8: Skew reduction methodology. (BINDAL et al., 2003)

was improved to achieve sub-10ps skew. Figure 3.7 illustrates the sub-10ps clock distribution scheme proposed for the Pentium 4 microprocessor. The clock distribution is divided into three steps, the Pre-Global Clock Network (PGCN), Global Clock Grid (GCG) and local clocking.

PGCN takes the clock signal from the output of the PLL to the input of the GCG. The clock signal passes through 27 inversion stages until it reaches one of the GCG drivers. Skew over 27 inversion stages is reduced by shorting inverters outputs as shown in figure 3.8. This methodology is able to average the variability effect, reducing the skew between adjacent paths. This idea was generalized later in (RAJARAM; HU; MAHAPATRA, 2004). Reducing clock skew in the PGCN is important to reduce power consumption due to short-circuit (crowbar) current at the GCG stage.

In total there are 1474 GCG drivers. GCG drivers are placed inside 8 stripes equally spaced over the chip area, as shown in figure 3.9. All grid drivers have a tunable drive strength. The input capacitance is the same for every GCG driver, therefore PGCN design can be decoupled from the GCG and the local distribution.

The local clock distribution presents two stages. The first stage presents delay programmability and clock gating capabilities. The second stage presents functional clock gating capability.

3.2.2 Itanium 1st Generation (2000)

The 1st generation of the Itanium processor was the first IA-64 processor. It was build on a 180nm technology and operated on a frequency of 800MHz. The clock architecture of this microprocessor was reported in (TAM et al., 2000). An overall view of the clock distribution for the first Itanium processor can be seen in figure 3.10.

Global clock distribution was implemented using a lateral shielded htree. Htree takes

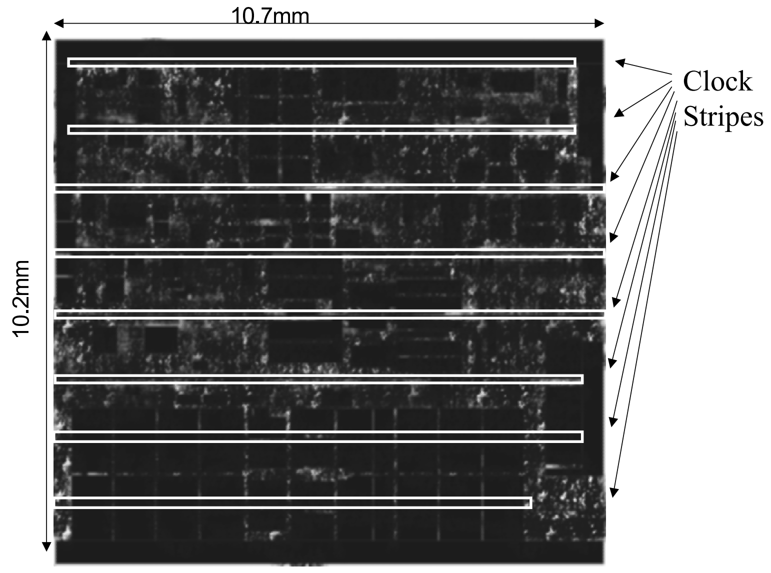


Figure 3.9: GCG drivers stripes. (BINDAL et al., 2003)

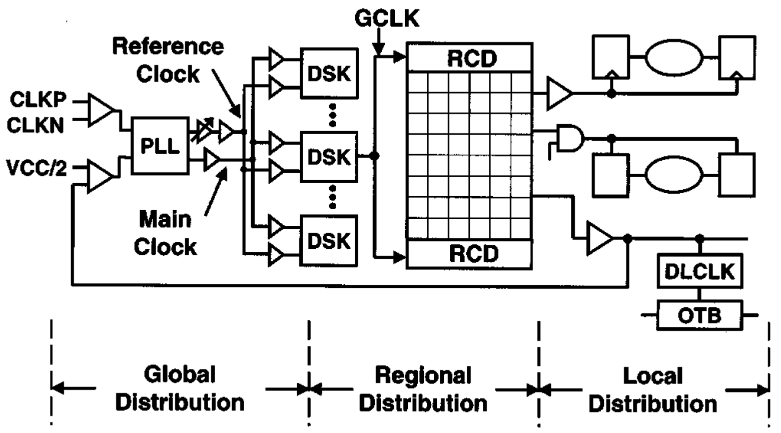


Figure 3.10: First generation Itanium clock distribution. (TAM et al., 2000)

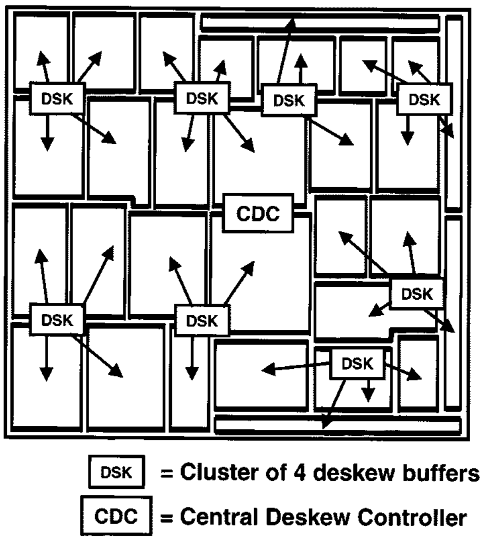

DSK
CDC
DSK = Cluster of 4 deskew buffers
CDC = Central Deskew Controller

Figure 3.11: Deskew buffer positions. (TAM et al., 2000)

the clock signal from the output of the PLL to eight different deskew buffers clusters. Each deskew buffer cluster contains four deskew buffers. Two deskew buffer clusters at the right bottom corner of the chip are misplaced from their original positions, as shown in figure 3.11, therefore wire length had to be matched to preserve htree symmetry. The deskew buffer is a variable delay inverter implemented as presented in (GEANNOPOULOS; DAI, 1998). The delay inserted by the deskew buffer is updated every 16 clock cycles.

The deskew buffer separates global and regional clock distributions. Each deskew buffer drives a regional clock driver that is connected to a clock mesh (grid). There is a total of 30 different clock regions, therefore 2, out of the 32 deskew buffers, are unused. Regional clock distribution is composed by the deskew buffer, the regional clock driver and the regional clock grid.

Local clock distribution is composed by the local routing connecting the regional clock grid to clock buffers and clock buffers to flip-flops. Each local clock buffer was designed to drive a specific load range. Additional local clock buffers can be inserted to intentionally skew the clock.

3.2.3 1.2GHz Alpha Microprocessor (2001)

The clock distribution of 1.2GHz Alpha microprocessor (JAIN et al., 2001) was released in 2001 by Compaq Computer Corporation. This microprocessor was built in a 180nm technology with 7 metal layers. It has 152 million transistors distributed over a 396.68mm^2 area. Supply voltage is 1.5V. Total power consumption is 125W.

The clock distribution for 1.2GHz Alpha microprocessor (XANTHOPOULOS et al., 2001) can be divided into four distinct clock domains, as shown in figure 3.12. The clock distribution for the microprocessor core is done by GCLK clock domain. NCLK distributes clock over memory and network subsystems. The other two clock domains distribute the clock signal to the cache L2 banks (L2RCLK and L2LCLK). No deskew methodology is applied between different clock domains.

The core of the clock distribution, GCLK domain, was process-migrated from Alpha 600MHz clock design (BAILEY; BENSCHNEIDER, 1998). Alpha 600MHz uses htrees and xtrees to distribute the clock signal to 16 GCLK drivers. Each GCLK driver is con-

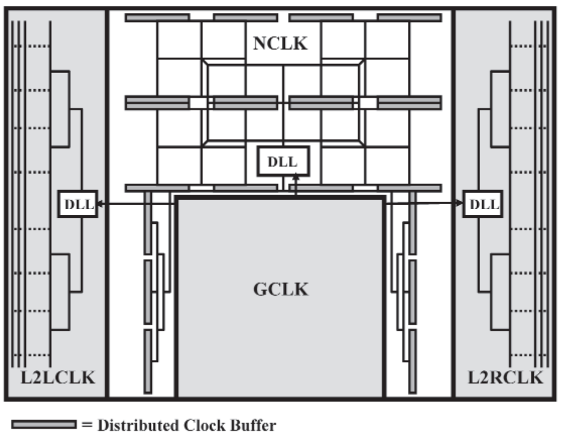


Figure 3.12: Clock domains for Alpha 1.2GHz microprocessor. (XANTHOPOULOS et al., 2001)

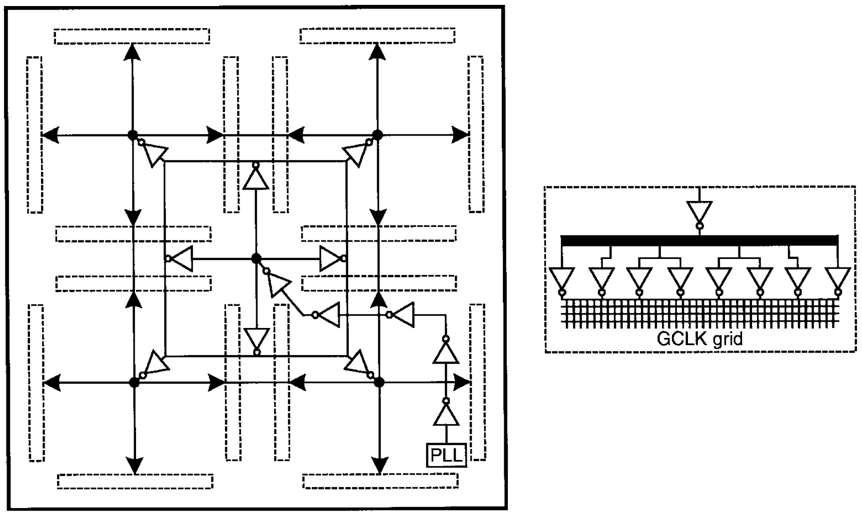


Figure 3.13: Clock distribution for Alpha 600MHz microprocessor. (BAILEY; BENSCHNEIDER, 1998)

nected to an array of buffers driving a clock mesh. Figure 3.13 shows the GCLK clock distribution scheme for Alpha 600MHz microprocessor.

NCLK domain uses an rectangular xtree to distribute the clock signal on the north portion, while two htrees were used to distribute the clock signal along the sides. NCLK domain can be further divided into another 11 sub-domains, as illustrated in figure 3.14. Each subdomain is driven by a subdomain buffer and contains a metal grid twice denser than NCLK global grid. Skew within a subdomain is about 50% smaller than the total skew in the NCLK domain.

L2RCLK and L2LCLK use sparse horizontal clock spines to distribute the clock along each memory bank. Partial htrees connect clock spines to the DLL clock output.

3.2.4 Power4 (2002)

Power4 microprocessor was released in 2002 by IBM (WARNOCK et al., 2002)(RESTLE et al., 2002). It operates in a 1.3GHz frequency. It was fabricated in a 180nm technology and contains 174 million transistors. The clock network drives 15200 clock sinks.

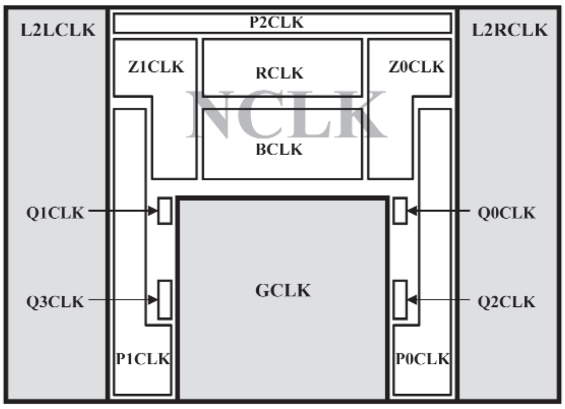


Figure 3.14: NCLK subdomains for Alpha 1.2GHz microprocessor. (XANTHOPOULOS et al., 2001)

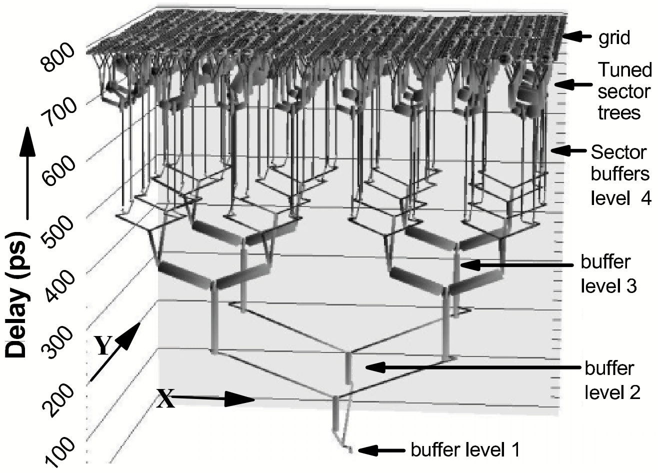


Figure 3.15: Power4 clock distribution. (RESTLE et al., 2002)

Clock distributions done by a global 8×8 buffered htree that drives a 64 array of sector buffers. Each sector buffer drives a minimum delay sector tree connected to a 32×32 clock mesh. Figure 3.15 shows a 3D visualization for the entire POWER4 clock distribution. Sector trees are tuned to minimize clock skew. Crosslinks are added and wire widths are changed to minimize skew in sector trees. Figure 3.16 shows 4 sector trees driving 1/16 of the clock mesh. Non-symmetrical routing, different wire widths and crosslinks can be observed in this figure. In this design a single clock domain is used, therefore no deskew methodology was implemented.

3.2.5 Itanium 2nd Generation (2002)

The second generation of the Itanium processor was released in 2002. This version was renamed to Itanium 2. It was implemented using a 180nm technology and operates at a 1GHz frequency. Itanium 2 clock distribution was presented in (ANDERSON; WELLS; BERTA, 2002).

The highest level of clock distribution is composed by a primary buffer, repeaters and variable delay buffers at its sinks. The clock signal is transmitted using differential signaling. In a regional level clock distribution is converted from a differential to a single-

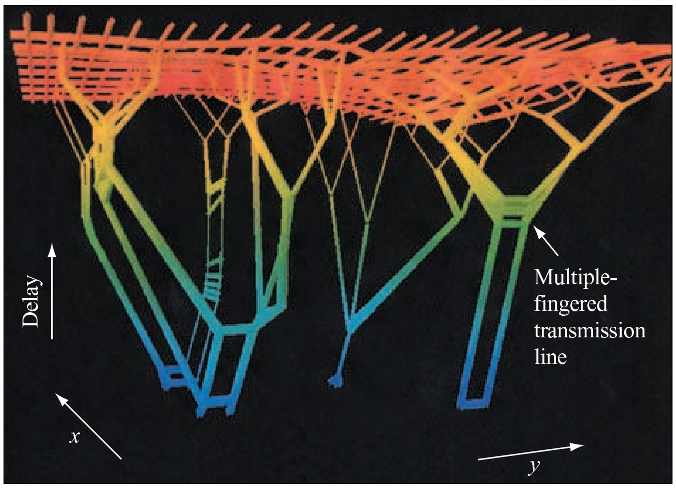


Figure 3.16: Power4 sector tree. (WARNOCK et al., 2002)

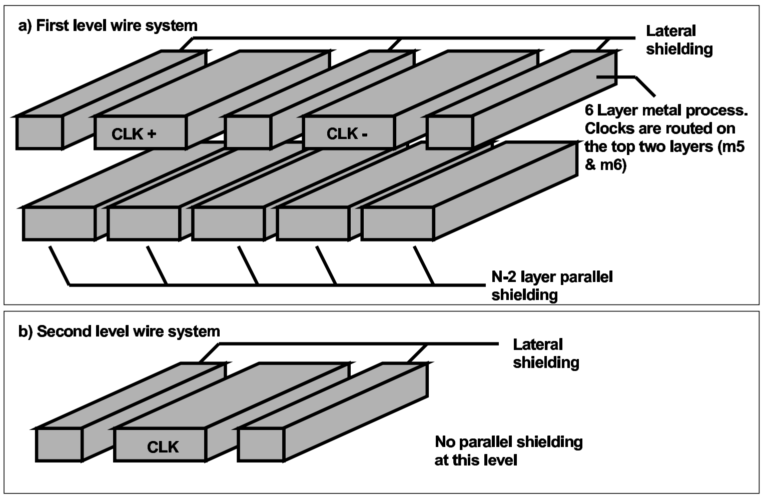

a) First level wire system
Lateral shielding
CLK +
CLK -
6 Layer metal process. Clocks are routed on the top two layers (m5 & m6)
N-2 layer parallel shielding
b) Second level wire system
Lateral shielding
CLK
No parallel shielding at this level

Figure 3.17: Clock lines shielding for Itanium 2nd generation. (ANDERSON; WELLS; BERTA, 2002)

ended signaling. Clock gaters are added to switch off clock for regions with no activity. Clock routing at both levels was performed using a htree topology. The clock wires were heavy shielded as shown in figure 3.17.

Although variable delay buffers were included in the clock distribution structure no deskew methodology was used. Variable delay buffer are used only for identifying critical paths at the silicon level. A global picture of the clock distribution scheme used in this microprocessor can be seen in figure 3.18.

3.2.6 Itanium 3rd Generation (2004)

The third generation Itanium processor was finished in 2004. It contains 410 million transistors and was fabricated in a 130nm technology. Clock operates at 1.5GHz. This generation is also commercially called Itanium 2. In comparison to the previous version, besides of 50% increase in clock frequency this version also presents a 100% increase in the cache L3 memory size.

Figure 3.19 presents the clock distribution scheme implemented for this microproces-

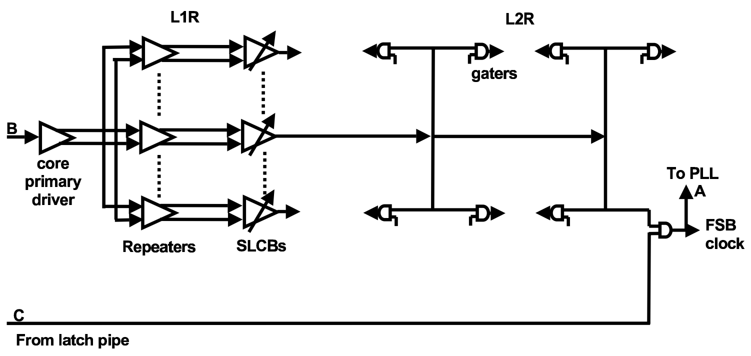


Figure 3.18: Clock distribution scheme for Itanium 2nd generation. (ANDERSON; WELLS; BERTA, 2002)

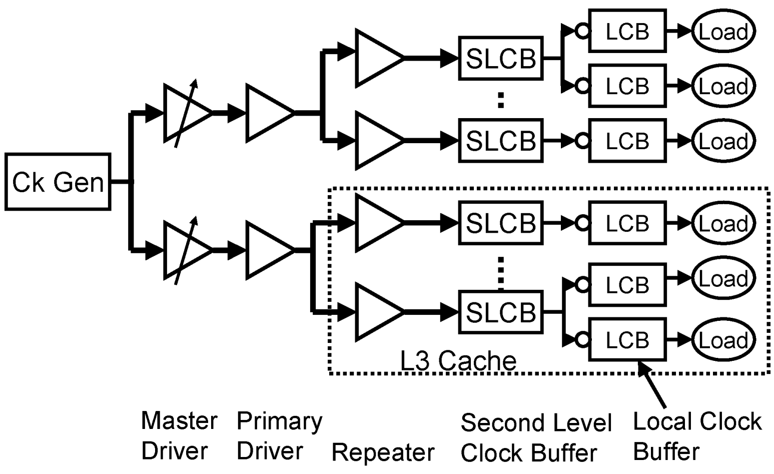


Figure 3.19: Clock distribution scheme for 3rd generation Itanium. (TAM; LIMAYE; DESAI, 2004)

sor. A differential signaling approach was implemented at the higher level of the clock distribution, where clock is distributed through an htree until each clock domain is reached. Regional clock distribution is done using a clock tree with tunable delay buffers. No clock meshes are used at global or regional level.

A fuse-based deskew methodology was implemented to reduce clock skew due to process variations. The methodology used in the deskew architecture design is the same presented in section 2.4.2.2. Variable delay buffers are calibrated only once after the chip is fabricated.

3.2.7 Power5 (2004)

Designed using IBM's 130nm technology Power5 microprocessor (KALLA; SINHAROY; TENDLER, 2004) was released in 2004. This processor operates above 1.5GHz with a supply voltage of 1.3V. This processor has two separate clock distribution schemes, one exclusively for memories and another one for all other synchronous elements.

The global clock distribution scheme relies on a non-symmetric htree. A global htree is used to drive the clock signal to a matrix of 40×32 sinks as illustrated in figure 3.20. The main clock distribution has a total of 91 buffers.

A clock mesh is used in the sinks of the global clock distribution. Sequential elements are connected to the clock mesh through clock gater cells, allowing an aggressive use of

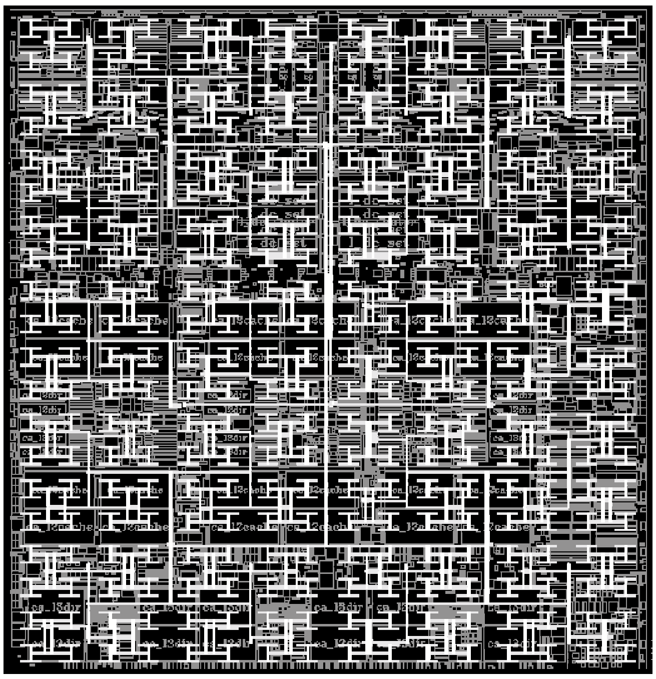


Figure 3.20: Power5 htree. (CLABES et al., 2004)

clock gating to conserve power.

3.2.8 Dual-Core SPARC V9 (2005)

The Dual-Core SPARC V9 (UltraSPARC IV+) microprocessor (HART et al., 2006) is the fourth generation of 64-bit SPARC architecture. It has 295 million transistors and operates at 1.8GHz. It was built on a 90nm dual-VT technology. It consumes 90W with a supply voltage of 1.1V operating at the nominal frequency. The total die area is 336mm^2.

The global clock network uses a true and completely symmetrical binary tree. By equalizing the number of levels and building a symmetrical tree the effect of power, voltage and temperature (PVT) variations on clock skew is minimized. The global clock tree has 16 stages and drives the clock signal from the clock root the a clock grid. Some nodes in the global clock network structure are shorted in order to make the clock network less sensitive to PVT variations.

Clock gaters (local clock headers) are used to drive the clock from the clock mesh to the local clock distribution. Local clock networks are shorted rows of local clock headers, in which only local clock headers with the same enable function can be shorted.

The total clock skew measured in this structure was 20ps.

3.2.9 First Cell Processor (2005)

The first cell processor is composed by a Power architecture processor combined with 8 other synergistic processors. It has a total of roughly 234 million transistors. It was built on a 90nm SOI technology with 8 metal layers. It operates at 4Ghz with a supply voltage of 1V (PHAM et al., 2006)(PHAM et al., 2005)(PHAM et al., 2006).

The chip contains three different global clock domains and each domain operates in a different frequency. The main clock domain is distributed through a clock mesh that covers 85% of the chip. The other two domains are also distributed using clock meshes interleaved with the main clock mesh. Both domains operate in a frequency fractioned

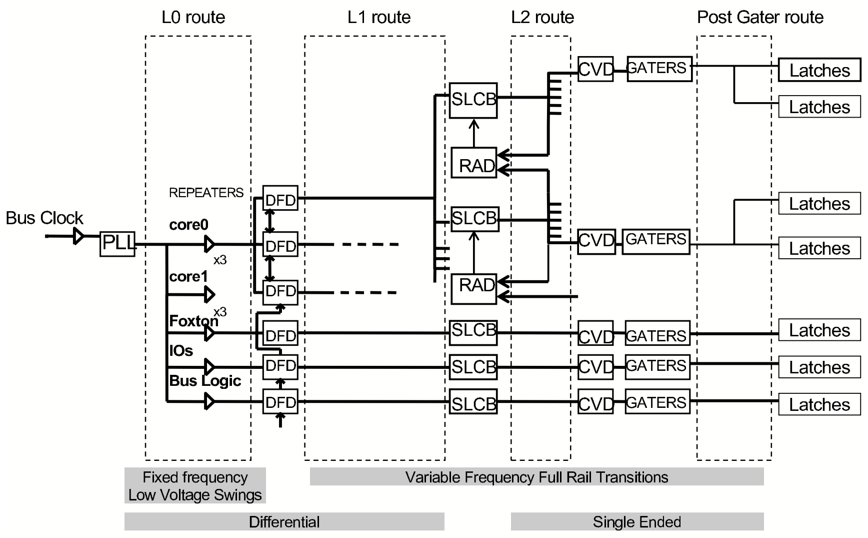


Figure 3.21: Clock distribution for Itanium Montecito microprocessor. (MAHONEY et al., 2005)

from the main clock.

The design of the main clock distribution is an extension of the clock distribution designed for Power4 processor, described in section 3.2.4. No information is given about the differences between Power4 global clock distribution and the global clock distribution used in this processor. The clock mesh located at the sink of the global clock distribution is built in the lowest impedance metal layers. 850 tunable buffers are used to drive the clock meshes. Tunable buffers allow a better control of clock arrival times, reducing clock skew, even for regions with very different load densities.

Clock signal is gated only at the flip-flops and latches. The clock gating circuitry is fully integrated to the flip-flop and latch designs.

3.2.10 Itanium Montecito (2005)

Itanium Montecito is a dual core Itanium microprocessor released in 2005 (MC-NAIRY; BHATIA, 2005). It operates at 1.8GHz and contains 1.72 billion transistors. The total die area is $595.55mm^2$. It was built using Intel's $90nm$ technology. The total dissipated power is $100W$.

The clock distribution for this processor is divided into four levels, from 0 to 3. Figure 3.21 gives a top view of all the levels. Level 0 drives the clock from the clock source to the different clock domains. Level 1 corresponds to the global clock distribution levels within each different domain. Level 2 corresponds to the semi global distribution, for most of the designs a clock mesh is used in this level but in this design a RLC matched htree is used. Local clock distribution corresponds to level 3.

Level 0 clock distribution drives the clock signal to the different ip cores inside the die. It uses a differential low voltage swing scheme to drive the clock signal to each one of the cores. The routing topology is a tapered htree. Clock wires are shielded in this level. Digital frequency dividers (DFDs) are at the sinks of this level.

DFDs are used to divide clock frequency by half. Level 1 distribution starts in DFDs outputs. This level uses a full swing differential distribution. A 90 degrees phase shifted copy of the clock signal is distributed along with it. Level 1 route follows the same route topology as level 0. Level 1 is terminated by the second level clock buffers (SLCBs) which are frequency multipliers and delay adjustable buffers. Clock is restored to its original frequency with a very low skew in the output of SLCBs. The deskew adjustment is done by a set of hierarchically arranged phase comparators located in the output of SLCBs. This is an active deskew mechanism.

Level 2 routing drives the clock signal from SLCBs to clock vernier devices (CVDs). It uses single-ended full swing signaling. This level also uses a htree routing topology. To minimize skew an in-house tool was used to tune wire widths to balance RLC delay in the htree. This sort of routing was also used in the second generation Itanium processors, described in section 3.2.5.

CVDs fulfill two roles, besides of driving the clock signal they can insert skew in the clock network that can be used to debug timing issues in the post fabricated chip. Clock gaters are inserted in the CVDs output. Level 3 routing is not designed along with the rest of the clock network. It is done specifically when each block is designed. The only constraint imposed to level 3 routing is that it can present a maximum delay of, at most, 12ps. Most of the clock skew in this microprocessor comes from this routing structure. This is the only routing level that may not use shields for all the wires. All the other routing levels are fully shielded.

3.2.11 Power6 (2007)

Power6 microprocessor was released in 2007 (FRIEDRICH et al., 2007)(THOMSON; RESTLE; JAMES, 2006). It uses IBM’s 65nm SOI technology with 10 levels of metal layers. Containing 700 million transistors distributed over a die area of $341mm^2$ it operates at frequencies higher than 5GHz. Power consumption can drop below 100W in power sensitive applications.

The global clock distribution is done by a 7 to 9 stages clock tree. Programmable delay buffers are inserted in this structure to reduce skew between different meshes. The global clock distribution is divided into two distinct clock domains. Within each domain "PVT bars" (i.e. clock spines) are used to short parallel stages reducing clock skew. Tunable sector htrees are driven by the global clock tree. Local Clock Buffers (LCBs) are located on the sinks of the sector htree. Local Clock Buffers are variable delay buffers that besides of calibrating the clock network delays they can also change duty cycle. Local Clock Buffers are connected to a virtual mesh. Since LCBs can change clock signal characteristic two LCBs can only be shorten when they always generate the same clock signal. For this reason a single clock mesh can not be used to connect all LCBs. By connecting only LCBs with the same clock signal characteristics, a virtual mesh is formed. A global view from Power6 clock distribution network and its latencies can be seen in figure 3.22.

(THOMSON; RESTLE; JAMES, 2006) presents another feature used during Power6 microprocessor clock network design. To reduce environmental variations effects and keep the duty cycle correct, clock lines are designed having transmission line theory in mind. By properly selecting clock lines width and lengths wave reflection effects are used to reduce environmental effects on propagation times and avoid undesired duty cycle changes. For this design, the wire length between buffer was set to 2.5mm in the clock network.

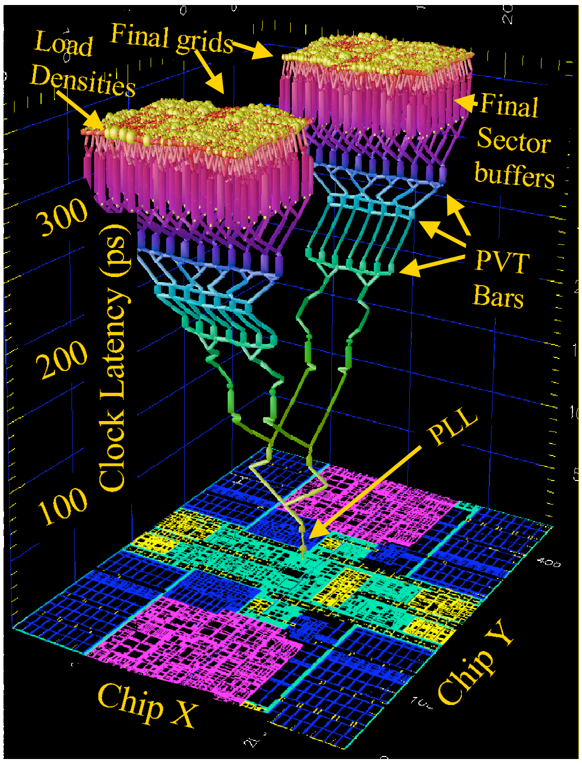


Figure 3.22: Power6 clock distribution. (FRIEDRICH et al., 2007)

3.3 A General Microprocessor Clock Distribution Architecture

This section has described the architecture of several different microprocessors. We can observe that each microprocessor utilizes different strategies to solve the problem of controlling the power consumption and designing a clock network robust to the effect of process and environmental variations. Although no two microprocessors use the same solution to distribute the clock signal we can observe some points in common among the characteristics of each clock distribution scheme:

- Global clock distribution is done by a global tree.
- Clock sinks are divided into different domains, domains can operate at different frequencies.
- Deskew circuitry is used to reduce the skew between different domains.
- A clock mesh is used within each domain to reduce the effect of variability.
- A local buffered clock tree drives the clock signal from the clock mesh to the clock sinks.

Figure 3.23 illustrates a general clock distribution scheme that combines the common characteristics observed in the different clock architectures described in this section. A global tree is used to drive the clock signal from the clock source to the clock meshes in the different clock domains. The global tree is shielded to reduce crosstalk effects. Other techniques can be applied in the global tree such as: differential signal to increase robustness to noise and reduced swing to reduce power consumption. Although shielding

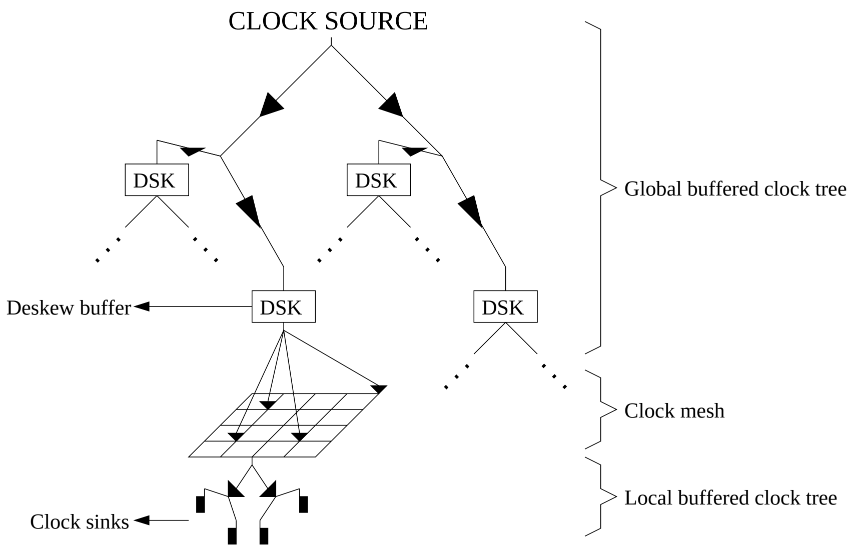


Figure 3.23: General clock distribution for microprocessors.

and differential signaling techniques may be applied to the global tree it may still be highly affected by environmental and process variations. The global clock tree covers large distances, requiring the repeaters to be inserted to guarantee the quality of the clock signal, therefore it is more affected by variations than any other part of the clock distribution network. To guarantee a low skew in spite of asymmetrical loading, global clock tree can be built in a completely symmetrical fashion by inserting dummy loads to equalize capacitance at branches.

Clock sinks may be divided into different clock domains, each clock domain is driven by a different clock mesh. This idea is the same as the TLM architecture presented in section 3.1.1.3. Deskewing circuitry may be inserted between the different clock domains to offset the effect of process variations. Clock meshes are responsible for filtering the effects of environmental variations in the global clock tree delivering a low skew and low jitter clock signal to the local clock trees.

Local clock trees deliver the clock signal from the clock mesh to the clock sinks, similarly to the MLT architecture described in section 3.1.1.3. However, differently from the MLT architecture the local clock tree found in the clock architecture of microprocessors is buffered. Local clock trees are not significantly affected by variations since they have very few buffer stages and a small wire length. The importance of having a buffered local clock tree is to decouple the total clock sink capacitance from the clock mesh and therefore reduce the short circuit currents in the clock mesh.

4 CLOCK MESH ANALYSIS

Clock mesh analysis is an extremely important task. Clock meshes are extremely hard to analyse due to the large number of elements required to model it and to loops introduced by the mesh structure. Besides that, clock arrival time estimation has to be extremely accurate. Small error margins at the arrival time estimation can reflect huge error on the clock skew estimation.

This section targets at the mesh analysis problem. We will first discuss some aspects of clock mesh modeling and next we propose a simple methodology to enable and speedup the simulation of large clock meshes. Related works are reviewed in the end of this section and after that conclusions are presented.

4.1 Modeling

Mesh wires are modeled using a π-model. A single-π model is shown in figure 3.3 while a 3π model is shown in figure 3.4. The number of πs used to model a wire depends on the wire length, l. Figure 4.1 shows the error obtained in the delay measurements performed on a wire model using different π-models. The larger the number of π used to model a wire the more accurate the model will be. In our experiments a 5π model was considered as the golden value.

Clock analysis needs to be extremely accurate. Therefore error margins above 1% are not acceptable. At the same time using a larger number of πs to model wires increases the electrical simulation time. Considering the tradeoff between accuracy and execution time performance we have adopted the following heuristic: any wire shorter than 300μ long is modeled using a single-π model, any wire longer than that is modeled using a 3π model.

Flip-flops don't need to be fully described since gate capacitance is the only thing seen by the clock network, therefore flip-flops are replaced by capacitors conencted to ground.

4.2 The Sliding Window Scheme

To enable the simulation of large clock meshes for the architecture evaluation study reported in section 3.1 a new simulation methodology was proposed. Long clock mesh simulation time is due to the very high number of circuit elements required to accurately model a clock mesh. Pre-characterization timing analysis techniques can not be applied to the clock mesh since gate models are not able to correctly model short-circuit current that are present among the buffers driving a clock mesh.

In this section we propose to use the divide-and-conquer heuristic to divide the mesh characterization task into smaller tasks that can be run in parallel in different computers

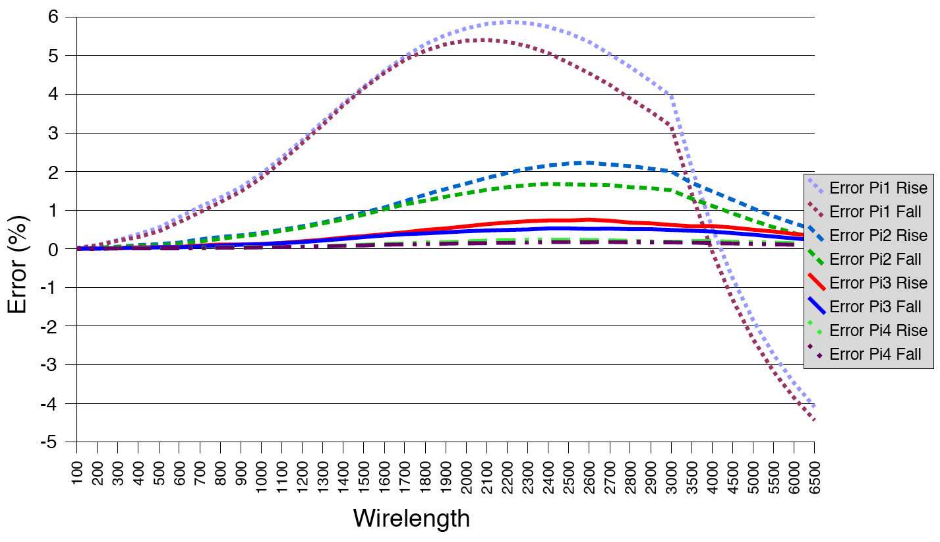
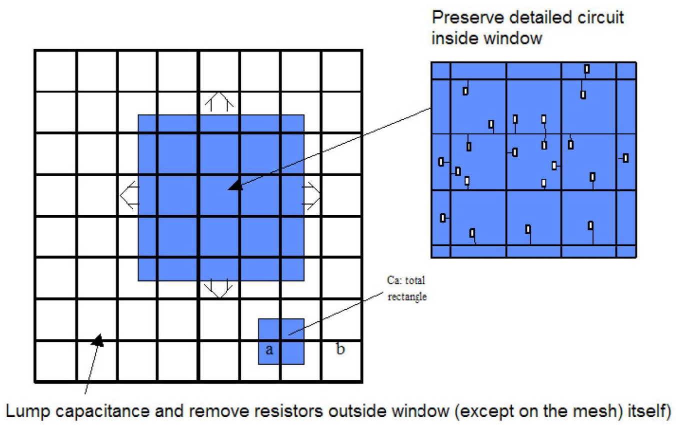


Figure 4.1: π-model accuracy comparison.



Preserve detailed circuit inside window
Ca: total rectangle
a
b
Lump capacitance and remove resistors outside window (except on the mesh) itself)

Figure 4.2: Sliding window scheme. (CHEN et al., 2005)

without significant loss of accuracy. This work was first presented in (CHEN et al., 2005). This work was developed in colaboration with other authors. The author of this thesis has participated in this work by studying how the method accuracy could be increased by discarding measures in the border of the accurate region.

We propose a new method called the sliding window scheme (SWS) for analyzing latency in clock distribution networks involving meshes. The sliding window scheme is based on the observation that for each signal source, i.e., the mesh buffer, the clock mesh can be deemed as a cascaded low-pass RC filter. For this RC filter, the attenuation of a ramp input signal is proportional to the exponential of the distance. Because of this exponential attenuation, if two nodes are geometrically far, they have very small electrical impact on each other. This phenomenon enables us to ignore some of the circuit details that are geometrically distant from the node we are interested in. In our proposed method, the mesh is modeled with two different resolutions: we use a detailed circuit model for the mesh elements geometrically close to the nodes we are measuring and simplified model for the mesh elements far from the nodes being measured. The simplification refers to the local FF connections.

The basic idea of SWS is very simple. Given a mesh $m \times n$, we define a rectangular window W of size $r \times s$ where $r < m$ and $s < n$. If we fix the lower left corner of W to a point on the mesh, W covers some fixed region of the mesh. The connection of a FF within W to the nearest mesh segment is modeled accurately by an appropriate π model. The FF clock input pin is modeled as a capacitance. If there are f FFs connected to a mesh segment, the mesh segment is divided into at most $(f + 1)$ sub-segments. Each sub-segment is modeled with an appropriate π model. FFs that lie outside W and their connections to the mesh are modeled approximately. The wire connecting such a FF to the mesh is replaced by an equivalent single capacitance. The wire resistance is ignored. Given a mesh node a outside W, the region covered by a is the unit rectangle shown in figure 4.2. Let C_a be the sum of the clock input pin capacitances of all the FFs in this region along with the capacitances of the wires connecting them to the mesh. Then, C_a is lumped as a single capacitance at a. The mesh segments outside W are still modeled with appropriate π models.

The Spice file corresponding to this model for the window location is generated and simulated. The clock latencies at all FFs in W are measured. Next, the window is slid either horizontally or vertically so as not to overlap with the previous locations. Once again, a Spice model is created and run. Simulating the entire mesh is thus broken down into multiple window-based simulations. In fact, $\lceil \frac{m-1}{r-1} \rceil \times \lceil \frac{n-1}{s-1} \rceil$ Spice simulations are needed to cover the entire mesh and thus all the FFs in the design.

As we will show, our scheme can complete on fine meshes and is accurate within 1% of the complete mesh simulation. Also, it is naturally suited to parallelization or grid-computing, since different Spice simulations are completely independent to each other.

The SWS scheme is a divide-and-conquer partitioning technique. Approximating the region outside the window reduces the number of nodes in the circuit model. Approximating each FF saves either 7 nodes if the wire is longer than 100μ or 3 nodes otherwise. In a typical design, where there are hundreds of thousands of FFs, the reduction on the size of the Spice model can be huge. Also, we can obtain CPU speed-up as well, as the following example illustrates:

Assume a 65$\times$65 mesh, and a design with 100K FFs where FFs are uniformly distributed over the chip. Let us also assume that all the wires and mesh segments are modeled with a single-π model. Let N_g be the number of nodes in the golden model, which is obtained when all the local FF wires that connect FFs to the mesh are modeled accurately. Each mesh segment is modeled with the single-π model and has 2 nodes (figure 3.3). The number of mesh segments is 64$\times$64 = 4096. Hence, the number of nodes on the mesh is about 8200. Each FF contributes with three nodes: one for the FF, one for the point where it is hooked to the mesh, and one internal node in the π model. Thus FFs contribute about 300K nodes. Then, N_g = 308K.

By using a window W measuring 17$\times$17, for a given location of W, let the number of nodes in the Spice model be N_W. As before, the mesh segments will contribute 8K nodes to the model. However, only about 1/16 of the total FFs lie within W. Then, only 7K FFs are modeled accurately. They contribute 21K nodes. The FFs outside W do not contribute any additional nodes, since they are lumped at the nearest mesh node. Then, N_W = 29K. Thus, we see a 10$\times$ reduction in the model size using SWS.

Let us assume that Spice run-time is $O(N^{1.5})$ to estimate the run-time of SWS. Since the number of nodes reduces by a factor of 10, each window simulation is about $10^{1.5}$ = 32 times faster than the golden model simulation. A total of 16 simulations are required to cover the entire mesh. Thus we can expect an overall speed-up of 2 for sequential execution on a single machine and a speed-up of 32 for parallel execution (assuming 16

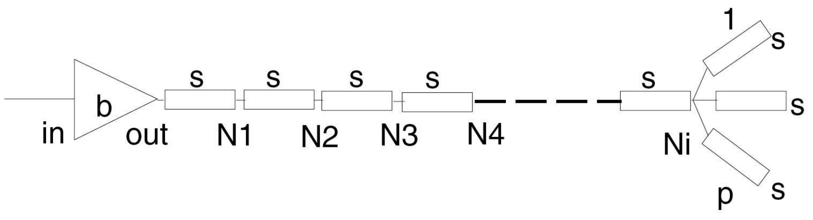


Figure 4.3: Model for justifying SWS. (CHEN et al., 2005)

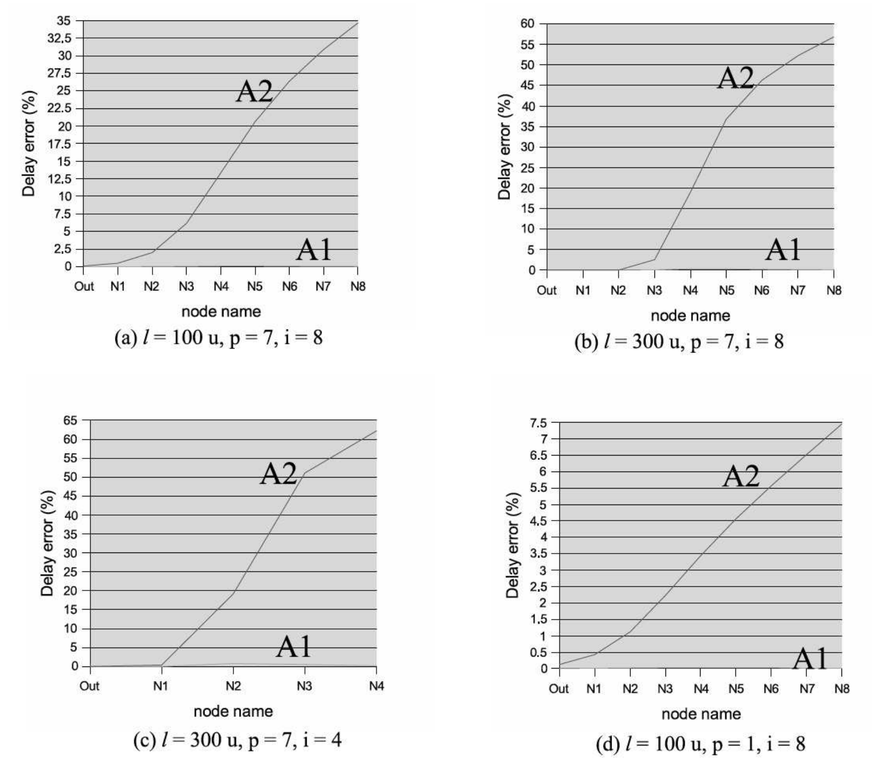


Figure 4.4: Experimental data justifying SWS. Approximation A_1 mimics SWS; A_2 does not include model of the circuit outside the region of interest. (CHEN et al., 2005)

machines are available).

4.2.1 SWS Justification

To justify the basic idea of SWS, we performed a series of simple experiments on 0.13μ technology in the circuit in figure 4.3. An appropriately-sized buffer b drives a series connection of i identical wire segments s (i is varied from 4 to 8), each of length. The final node N_i fans out to p branches, where each branch contains a single segment s. The value of p is varied from 1 to 7, and that of l from 100μ to 300μ. Nodes out and N_1 through N_i mimic the nodes inside the window of SWS, where we wish to make accurate delay measurements. The p branches mimic the local FF-mesh connections outside the window. During simulation, each s is replaced by a single-π model. This is the exact model. We measure the (rise) arrival times at out and various N_j nodes (the waveform in the input of b serves as the timing reference). These arrival times form the basis for determining the accuracy of the next two approximations, in which the p branches are represented by simpler models.

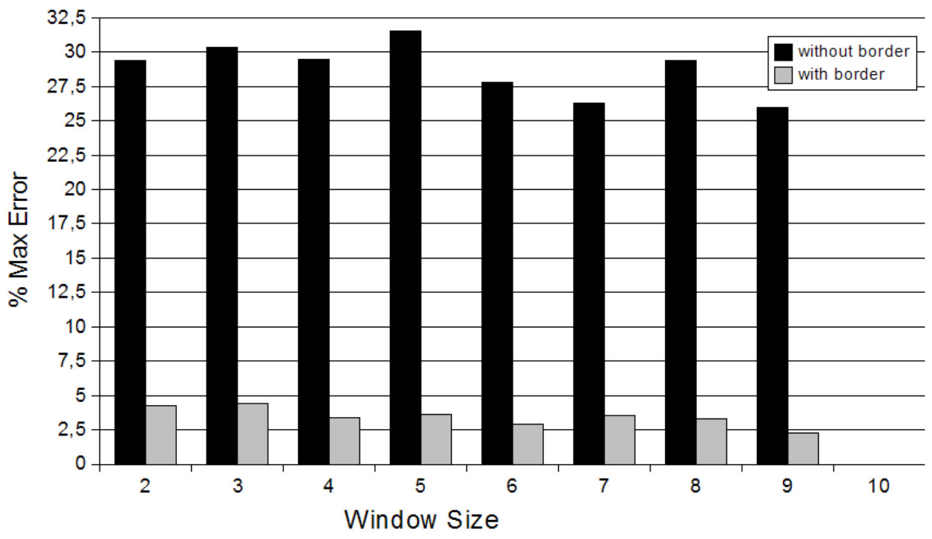


Figure 4.5: Maximum error without and with border for $10mm \times 10mm$ chip, 10×10 mesh, 10K FFs and a buffer on every other mesh node. (CHEN et al., 2005)

In the first approximation, we set the resistance of each of the p branch wires at N_i to zero. This makes all the branch wires purely capacitive and effectively lumps all branch capacitances at N_i. Let us call this approximation A_1. A_1 mimics the basic idea of SWS, where the local connections to FFs are replaced by appropriate capacitances at the nearest mesh nodes; the resistance components of local connections are set to zero. We measure the arrival times at various nodes N_j and compare them with the corresponding values from the exact model.

In the second approximation, we remove all the branches. This mimics the extreme choice that the region outside the window is completely eliminated and not included in the model. Let us call this approximation A_2.

We simulated the clock mesh exact model and the models obtained from A_1 & A_2 for different values of l, i, and p. Figure 4.4 shows the percentage delay error at various measurement nodes for A_1 and A_2. In all the experiments, delay errors for the approximation A_1 were found to be less than 0.5%. However, the errors for A_2 were as high as 65%. The error is higher if s is longer (b vs. a), i is smaller (c vs. b), or p is more (a vs. d). Also, the error increases sharply as the measured node shifts to the right and gets closer to the branch region that is being approximated. This confirms the basic premise of the window scheme: ignoring resistance of the local FF wires outside the window introduces insignificant delay error, but completely ignoring the region outside the window can result in large errors.

Note: It is not necessary to consider the length l of each segment s longer than 300μ. If $l = 300\mu$, the wire between b and N_i has at least 4 copies of s and hence is at least $1.2mm$ long. A repeater is inserted in the clock after $1.2mm$ to restore the signal rise and fall times. As for each of the branches to FFs, skew considerations dictate that these connections must not to be longer than 300μ.

4.2.2 SWS Accuracy

The circuit set-up in the last section included a single buffer driving a simple tree of wire segments, it did not take into account loops and the multi-driver nature of the

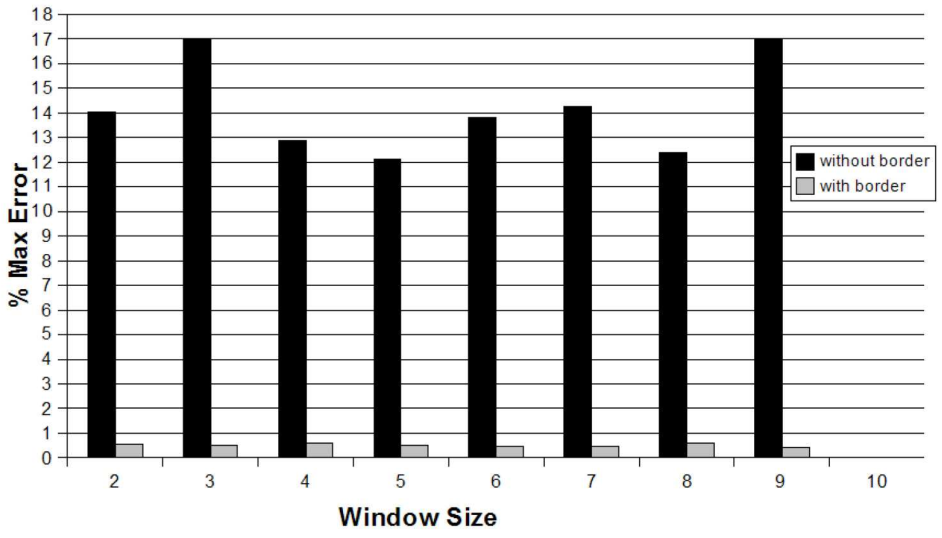


Figure 4.6: Maximum error without and with border for $10mm{\times}10mm$ chip, 10×10 mesh, 10K FFs and a buffer on every mesh node. (CHEN et al., 2005)

mesh. In this section, we model the actual mesh architecture to check the accuracy of SWS. The mesh size was set to 10×10, and the number of FFs to 10,000. The FFs are placed randomly with a uniform distribution on a $10mm{\times}10mm$ chip. Mesh buffers are assumed to be present on every other mesh node and are sized according to the load in the region around their respective mesh nodes. The flat simulation for the entire mesh could finish, yielding the golden clock latency values $L_g(i)$ for each FF i. For SWS, we vary the window size from 2×2 to 10×10. For each window size $j{\times}j$, let $L_{W_j}(i)$ be the latency for FF i in the SWS. We compute E_{max}, the maximum percentage error in latency over all FFs as follows:

$$E_{max} = max_i\{\frac{L_g(i) - L_{W_j}(i)}{L_g(i)}\}\times 100 \tag{4.1}$$

We plot the maximum error, E_{max}, for each window size j in figure 4.5 as bars labeled without border. The size $j = 10$ means that the whole mesh is simulated in its entirety by including accurate models for the connections to all FFs. In other words, this simulation yields the golden latencies. It can be seen that E_{max} ranges from 26% to 32%. Such large error values are unacceptable, given the stringent accuracy requirements for the clock signal.

4.2.3 Improving SWS Accuracy

Analysis of flip-flops with large latency errors revealed that such FFs were almost always close to the window boundary. SWS accuracy improves drastically when the latencies of FFs on the border are ignored. So we enhanced the basic SWS by expanding the original window W to W' by including a border around W, as shown in figure 4.7. The window W' was modeled accurately. However, the clock latencies were measured only for the FFs in the original window W. The latencies for FFs in $W - W'$ were not measured when they fall within the core of another appropriate window.

We reran the above experiment with the border-enhanced SWS. The border was chosen as one mesh segment along each one of the four window boundaries. As shown in figure 4.5, bars labeled with border, the maximum error with the border is less than 4.5%

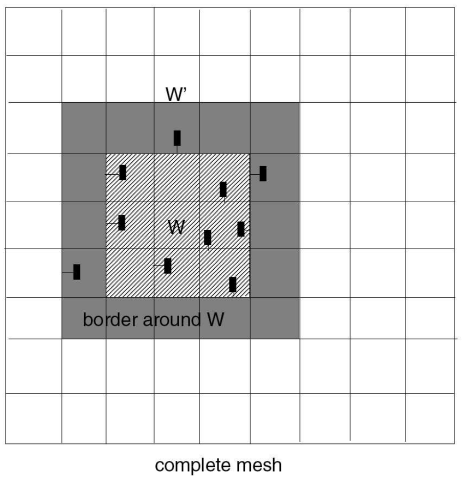


Figure 4.7: Window W and its border. (CHEN et al., 2005)

over all window sizes. Interestingly, *the maximum error is almost constant as the window size changes*. Then, other criteria can be used to select an optimal window size, as described in section 4.2.4.3.

As another set of results, in figure 4.6, we show the results for the same set-up as before, except the mesh buffers are now present on every mesh point. Without border, the maximum error ranges from 12% to 17% for different window sizes. With the border, it is always less than 1%.

Intuitively, there are large errors in the region outside W'. The border $W' - W$ provides a buffer zone between the outside region and W, through which the latency inaccuracy reduces.

4.2.4 Experimental Results

We have developed a clock mesh analysis tool which reads a chip specification (e.g., chip dimensions), FF locations, technology information, mesh buffer sizes and locations, and mesh parameters (such as mesh size, wire widths). It then uses Spice transient simulation to compute clock latencies for the FFs with respect to the clock source (which is connected to the inputs of all the mesh buffers). The computation is based on the proposed sliding window scheme. Currently, it requires window size as input. For each window location, the tool generates the Spice model for the mesh, local wires and flip-flops within and outside the window. Unix shell scripts were written to manage the sliding windows' generation, simulation, and extraction of clock latencies from the simulation output.

We designed our experiments to achieve three goals:

1. To show that our proposed scheme is accurate for measuring clock latencies. For this, we need to use mesh and circuit instances that do not exceed the capacity of Spice and can be completed in one single simulation.

2. To show examples where the (flat) simulation for the whole mesh could not finish, but SWS could. Also, to study the CPU time and memory trade-offs as a function of the window size.

3. To come up with a method for determining the best window size. The best window size is the one that generates a Spice model within the machine's memory capac-

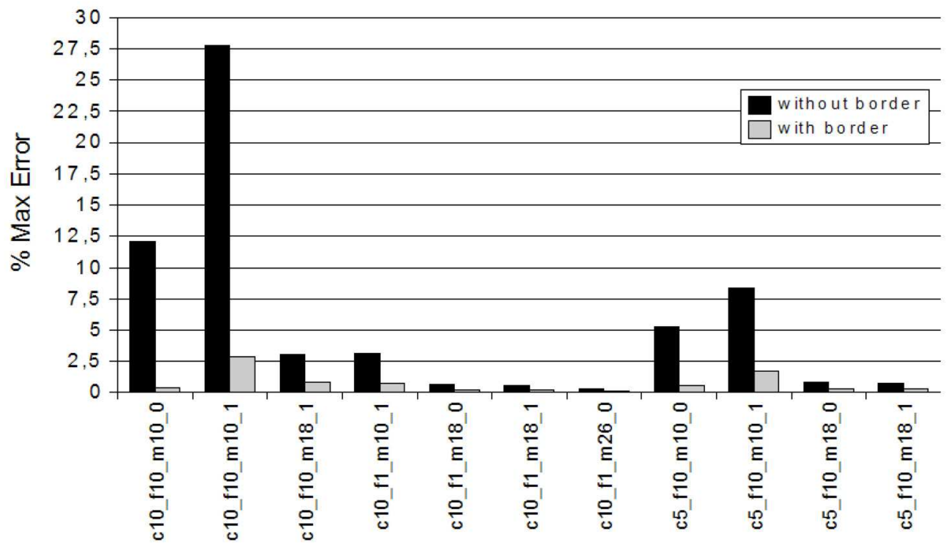


Figure 4.8: Accuracy of SWS for different experimental settings. (CHEN et al., 2005)

ity, minimizes the error, and either a) minimizes the overall simulation time, or b) minimizes the turn-around time for parallel simulation.

All the experiments were conducted in an industrial 0.13 technology.

A typical run of a circuit simulator (e.g., Hspice) session to simulate one Spice file is called a simulation. A win-experiment is defined as the collection of simulations resulting from sliding the window across the mesh to cover all the FFs, with a given chip size, mesh size, FF count, window size and mesh buffer locations. The set of *win-experiments* obtained when the window size is varied from 1 to the maximum possible value is called an experiment. We carried out numerous experiments with different values of chip size, mesh size, and FF count. Two chip sizes were used: $5mm{\times}5mm$ and $10mm{\times}10mm$. Three different mesh sizes were used: $10{\times}10$, $18{\times}18$, and $26{\times}26$. FF counts of 1K and 10K (1K = 1000) were used. FFs were placed randomly with a uniform distribution. Buffer steps used were 0 and 1; 0 means every mesh node has a mesh buffer and 1 means every other mesh node has a buffer. Due to lack of space, we only present results for a selected window size for each experiment. This window size was picked to minimize an average of percentage maximum error, percentage average error and CPU time. The following labeling scheme was used for an experiment. For instance, the experiment with the chip size of $10mm{\times}10mm$, FF count of 10K, mesh size of $10{\times}10$, and a buffer attached to each mesh node was labeled *c10_f10_m10_0*. The accuracy results for this experiment were shown in figure 4.6 for all window sizes.

4.2.4.1 Accuracy

Figures ??? and ??? have already shown SWS accuracy results for experiments *c10_f10_m10_1* and *c10_f10_m10_0* respectively over all window sizes. Figure 4.8 shows results for different experiments, but only for the selected window size (as described above) when running SWS without including the border. It can be seen that the maximum error in most of the cases is below 2.5%. However, for *c10_f10_m10_1*, the maximum error is 27.5%.

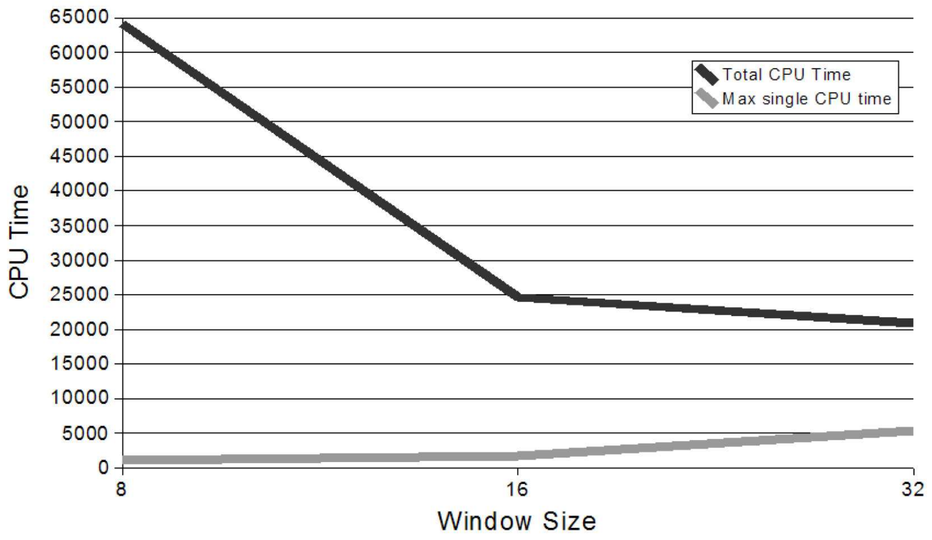


Figure 4.9: CPU time as a function of the window size. Total CPU time is relevant for sequential execution. Max single CPU time is the turn-around time, assuming maximum parallel processing. (CHEN et al., 2005)

We reran all the experiments using border-enhanced SWS, with a border of 1. For all the experiments except two, the maximum error is now below 1%. The other two experiments had maximum errors of 2.9% and 1.8%. Although not reported in the figures, for any given FF, the difference between the golden latency and the SWS-computed latency was less than $1ps$ almost always. The 2.9% error case was an exception, where a delay difference of $7ps$ was seen: the golden latency for a FF was $233ps$, but the SWS computed a latency of $226ps$. These results show that the border-enhanced sliding window scheme is extremely accurate for computing clock latencies.

4.2.4.2 Large (Mesh + Design) Instances

Another experiment was run using a 65×65 mesh with 100K FFs placed randomly with a uniform distribution on a 1.1GHz Sparc64-V with 4GB main memory. The goal of this experiment was to demonstrate that there are cases where the flat golden simulation cannot complete, but SWS can. For SWS, window sizes equal to 8×8, 16×16 and 32×32 were evaluated. We used Hspice from Synopsys for this experiment. It was found out that Hspice could not complete the flat simulation for the whole 65×65 mesh. Hspice used up more than 2GB of memory and aborted after 36 hours. Note that 2GB is the address space limit of the Hspice binary we used. In figure 4.9, we plot the total execution time for the win-experiment for each window size. Note that 64, 16, and 4 simulations are required for the window sizes 8×8, 16×16 and 32×32 respectively. We also show the maximum execution time for a single simulation (i.e., for a fixed location of the window). If different simulations for a given mesh could be done in parallel in different machines, the execution time would be determined by the maximum CPU time for simulating a single window location. The maximum memory required by the simulator is shown in figure 4.10. The amount of memory shown for the flat simulation (i.e., window size 64) is a lower bound, since the simulation did not finish.

We note from these plots that as the window size increases, the amount of memory needed for the simulation increases and so did the simulation time for a fixed window

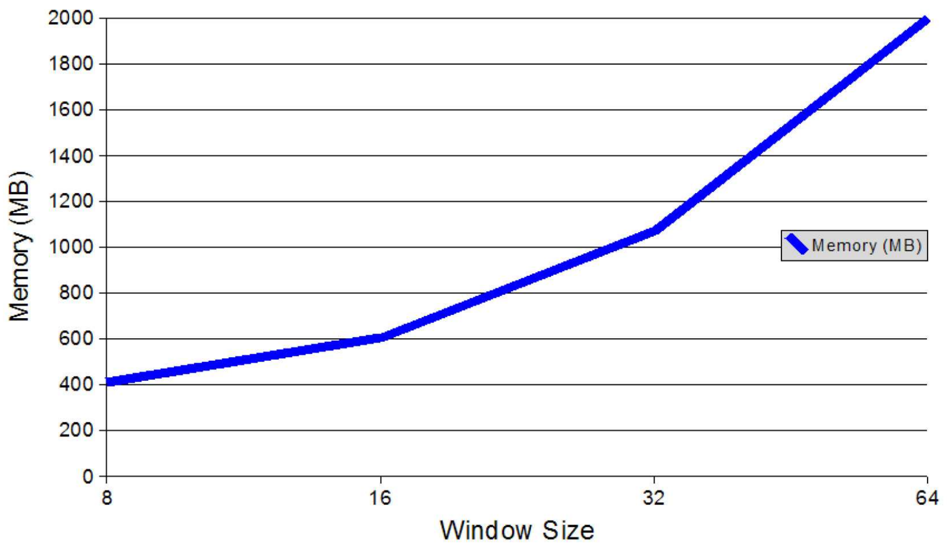


Figure 4.10: Memory usage as a function of window size. (CHEN et al., 2005)

Table 4.1: Runtime on a real design with about 300K FFs. Parallel execution assumes 4 processors.(CHEN et al., 2005)

Mesh size	Execution Time	
	Sequential	Parallel
65×65	6h48min	1h46min
129×129	20h22min	5h18min

location. This was expected, since a larger window covers more FFs whose connections to the mesh are modeled accurately in SWS. Hence the size of the Spice model and the execution time grew. However, the total time over all simulations tends to be large for small window sizes and goes down as the window size increases to half the mesh size. This happens because the number of simulations for the whole mesh also reduces with increasing window size.

We also tested the sliding window scheme in a real design, which had almost 300K FFs. The cell and FF placement had already been done using a commercial placement tool. We used two differentmesh sizes: 65×65 and 129×129. The execution time is reported in table 4.1 for two cases: sequential execution in one machine and parallel execution assuming 4 machines. The table shows that SWS can handle real designs overlaid with fine-grain clock meshes. Parallelization, even with only 4 machines, can make the turnaround time practical.

4.2.4.3 *Optimal Window Size*

The main advantage of SWS is to be able to accurately simulate large meshes and designs that cannot be completed with a flat simulation. So the first requirement on the optimal window size is that the resulting model fits in the main memory. Next, we would like to complete the simulations quickly. As for accuracy, we note from figures 4.5 and 4.6 (with border) that the SWS accuracy does not depend on the window size, and hence it is not a determinant of window size.

It is clear from section 4.2.4.2 that a smaller window simulates much faster than a bigger one. So for parallel simulation, it is better to pick small window sizes. A smaller window also yields smaller simulation model. On the other hand, larger windows tend to have smaller total simulation time. So they are preferable for sequential simulation, as long as the SWS model fits in the machine memory.

4.2.5 Conclusions

Analysing clock mesh of a large industrial design has been a difficult problem. In this study, we presented a new sliding window scheme to analyse the latency in clock meshes. We showed that Hspice could not finish on a 65$\times$65 mesh with 100K FFs. It needed more than 2GB of memory. Our technique could complete in less than 1.5 hours within 1GB memory. The border-enhanced SWS, when applied to smaller instances of mesh and FFs, almost always comes within 1% of the delay computed from the Spice simulation of the complete mesh. We also proposed strategies for selecting the optimal window size, which take into account the total machine memory and the degree of parallelization. We applied our scheme successfully on a large, real industrial design. Our technique extends the capability of Spice in handling large (RLC) clock meshes and designs. Finally, our scheme is naturally suited for parallelization and achieves a turn-around time of less than 2 hours on a design with almost 300K FFs and with a 65$\times$x65 mesh. Thus, our scheme effectively solves the problems associated with traditional clock mesh analysis. This should enable widespread use of clock mesh architectures in ASIC and processor designs.

Our technique can be further improved to reduce the memory requirement and run-time as follows.

1. We can reduce the complexity of the model by simplifying the region of the mesh outside the window. For instance, each mesh segment outside the window could be modeled always as a single-π, instead of the current threshold-length-triggered switch between single-π and 3-π models.

2. We need to explore the applicability of linear and non-linear model order reduction techniques in the context of the mesh. These techniques could potentially reduce the model size and help speed up the clock mesh analysis.

3. With technology scaling, the variations in voltage, temperature, and crosstalk noise lead to clock jitter. We plan to work on jitter measurement in clock meshes.

4.3 Related Works

The work presented in section 4.2 was the first to address the problem of clock mesh analysis. Later other works were published about this topic. This section presents other solutions to speedup clock mesh simulation and enable the simulation of large clock meshes.

4.3.1 Accelerating Clock Mesh Simulation Using Matrix-Level Macromodels and Dynamic Time Step Rounding

This work was presented in (YE et al., 2008). It presents a mathematical approach to speedup the transient simulation of clock meshes. Clock mesh simulation is expensive due to the large number of elements used to model the clock mesh and also due to the

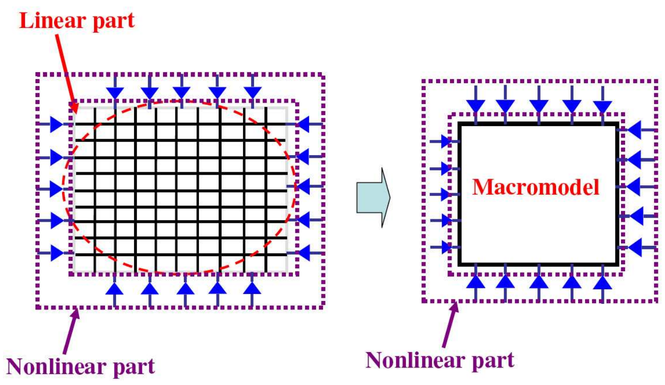


Figure 4.11: Macromodel for linear part. (YE et al., 2008)

non-linear equations that model the behavior of mesh buffers. This work proposes to speedup mesh simulation by computing a simplified model for the linear components of the clock mesh, therefore reducing the number of equations that need to be solved when the non-linear part is solved.

Figure 4.11 shows linear and non-linear part of the circuit and shows which part is replaced by the macromodel. The size of the macromodel is determined by the number of ports in the linear part of the clock mesh.

The macromodel can be efficiently computed because the linear modified nodal analysis equations form a symmetric positive definite (SPD) matrix. A SPD matrix can be efficiently solved by using Cholesky factorization (GOLUB; LOAN, 1989). This is not possible if linear and non-linear equations are solved together. Cholesky factorization solvers are, in practice, a few times faster than generic LU solvers. After computing the macromodel, the macromodel is computed it is combined with the nonlinear circuit equations and then solved using Newton-Raphson method. Since the macromodel is a reduced set of equations a large speedup is gained in comparison to applying Newton-Raphson method to the initial set of equations.

It should be noticed that the macromodel for the linear part of the mesh has to be recomputed for each different time step. Dynamic time steps algorithms are widely used to enhance simulation efficiency and accuracy. To prevent the macromodel to be recomputed for every time step change the macromodel is precomputed for a fixed set of time steps and then the dynamic time steps are approximated by the closest precomputed time step smaller than the desired time step. The time step range can be divided by geometrically spaced time steps, e.g., for a range of time steps $[h_{min}, h_{max}]$ the set of time steps is going to be $\{h_{min}, 2^1 \times h_{min}, 2^2 \times h_{min}, ..., 2^n \times h_{min}\}$, so that $2^n \times h_{min} \geq h_{max}$. By using geometrically spaced time steps the total number of time steps at which the macromodel needs to be computed is $1 + \lceil log_2(h_{max}/h_{min}) \rceil$.

Table 4.2 presents a runtime comparison between a SPICE simulation and the proposed technique using dynamic time step computation. It can be seen that the proposed technique can reach a $40\times$ speedup factor.

The proposed technique has shown to be effective in reducing mesh electrical simulation time. No approximation is made during the macromodel generation then no error is introduced in the simulation. In comparison to other techniques this technique has the disadvantage of not enabling parallel simulation. Besides it requires the user to implement its own simulation environment. It should be also noticed that the proposed methodology is not able to compute the waveforms at nodes inside the macromodel. A post-processing

Table 4.2: Runtime comparison between macromodel-based simulation and SPICE simulation. (YE et al., 2008)

Ckt	#nodes	#elements	#drivers	Spice Runtime	Macro. Runtime	Speedup
ckt1	400	1200	2	8.736s	1.008s	8.67
ckt2	1000	3000	6	63.885s	4.654s	13.73
ckt3	2500	7500	20	727.24s	34.18s	21.28
ckt4	6500	20K	30	1h18min	114.84s	40.66
ckt5	40K	120K	100	4h21min	778.16s	20.15
ckt6	70K	200K	150	7h30min	1149.91s	23.48

step is required to derive the waveforms observed at the clock sinks.

4.3.2 Analysis of Large Clock Meshes Via Harmonic-Weighted Model Order Reduction and Port Sliding

This paper presents a model order reduction technique based on harmonic weighting combined to a port sliding technique which increases the method scalability (YE et al., 2007). The harmonic-weighted model order reduction weights the importance of each frequency domain harmonic on specific waveform characteristics. The model order reduction is performed in such a way that selected waveform characteristics are preserved. The port sliding technique is used to compute the waveform in the mesh buffers output. The port sliding technique comes from the observation that computing a compact multi-port model for the whole mesh is a complex task when the number of ports is high.

The overall algorithm for the proposed analysis methodology presented in (YE et al., 2007) is shown in figure 4.12. The first step of the algorithm is to compute the weights for each harmonic for a general clock waveform in such a way that the delay estimation error is minimized. Then for each mesh buffer i the output waveform is estimated using approximation techniques for elements with small influence over the buffer under evaluation. The final step relies on analyse the whole mesh and compute the waveform at the clock sinks.

Three different approaches are proposed to simplify circuit elements with small influence on the buffers under evaluation, a driver merging strategy, an importance-weighted model reduction and a combination of both techniques. The driver merging technique relies on merging far away mesh buffers into a single effective mesh buffer with size equal to the sum of the sizes of the merged drives. The importance-weighted model reduction consists in using only the DC and the first order moment to model far away buffers. As mentioned earlier, both techniques can be used in combination.

The proposed techniques were shown to be effective in speeding up the characterization of clock meshes and enabling the characterization of large clock meshes. The error introduced by this technique is very small. Table 4.5 reports the runtimes and errors associated with this technique. We can see that the error with the combined simplifications associated to the port sliding method produces an error of at most $2.5ps$ for the tested circuits.

The proposed methodology doesn't require any simplification to be applied to the mesh buffer, unlike most of other works. This methodology is more scalable than other

```
Input: Full Model: $G$, $C$, $B$, $L$; $f_0$, Ctrl fac: $\kappa$, Red-mod. size: $S_R$
Output: Reduced order model: $\tilde{G}$, $\tilde{C}$, $\tilde{B}$, $\tilde{L}$
1  Compute $W_k^{\prime}s$ using harmonic sensitivities equations
2  V ← [ ]
3  for each input $i$ do
4      Compute the transfer function at dc: $V_i \leftarrow TF(0, i)$
5      for each harmonic $k$, $k = 1$, ..., $N_h$ do
6          Compute the transfer function: $TF(k, i)$
7          $V_i \leftarrow [V_i, Re\{TF(k, i)\}, Im\{TF(k, i)\}]$.
8      end
9      Normalize each column in $V_i$ and multiply each column using the
       corresponding weight $W_k$.
10     Perform singular value decomposition (SVD) on the weighted $V_i$ matrix:
       $V_{i,w} = P_i \sum_i Q_i^T$.
11     Keep the first $k$ dominant singular vectors in $P_i$: $V \leftarrow [V[p_{i,1}, ..., p_{i,k}]]$.
12 end
13 Perform SVD on $V = P \sum Q^T$.
14 Keep the first $S_R$ dominant singular vectors $X$ of $P$, $X = [p_i, ..., p_{S_R}]$ for model
   reduction: $\tilde{G} = X^T G X, \tilde{C} = X^T C X, \tilde{B} = X^T B, \tilde{L} = X^T L$
```

Figure 4.12: Harmonic-weighted Model Order Reduction (MOR).

Table 4.3: CPU time comparison of CSAV and Hspice (unit: second). (ZHANG et al., 2008)

ckt	Size	# Drivers	Spice	Comb. merge and MOR		
				runtime	ave. err	max err
mesh2	27K	53	2h2min	17min40s	0.4ps	2.2ps
mesh3	50K	100	3h	28min52s	0.8ps	2.5ps
mesh4	100K	100	6h30min	40min16s	0.6ps	1.9ps
mesh5	300K	200	NA	3h10min	-	-

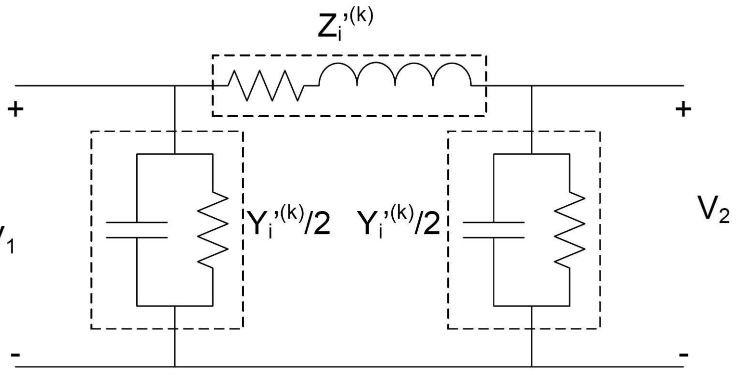


Figure 4.13: π-model used to model mesh wires. (WANG; KOH, 2007)

works since it computes the waveforms at the output of the mesh buffers independently. The waveform computation in the output of mesh buffers can be parallelized, improving the speedup in comparison to the Spice simulation.

4.3.3 A Frequency-domain Technique for Statistical Timing Analysis of Clock Meshes

This work presents a frequency domain technique to analyse clock mesh timing in the presence of variations (WANG; KOH, 2007). The authors claim that their work presents the following contributions:

- Formulates a second order timing model for the sink arrival time with respect to wire width and input arrival times.
- Computes the mean and covariance expressions for clock arrival time at the clock sinks considering mesh wire width and input signal arrival time variations.
- When compared to Monte Carlo simulations the proposed statistical timing analysis technique can reduce execution time by orders of magnitude with only 1% error in the mean estimation and 3% error in the standard deviation estimation.

The method models the input clock signal arrival times and the wire widths by Gaussian distributions. Mesh wires are modeled using the single-π model shown in figure 4.13. Mesh drivers are replaced by $1/T$-periodic voltage source with rise/fall time of τ. For convenience the Thevenin equivalent is replaced by the Norton's equivalent using that $I_N = V_{TH}/R_{TH}$, $R_{TH} = R_N$.

In the next step all input current sources are translated to the frequency domain by a Fourier transform, to keep the equations consistent to the equations presented in the paper the physical time is represented by t':

$$I(t') = \sum_{k=0,\pm1,\pm2,\pm3...} diag[e^{jkw_0(t'-s)}]\cdot I^k \tag{4.2}$$

In 4.2 $diag(v)$ denotes a diagonal matrix with the diagonal being the vector v and kw_0 is the k-th order harmonic angular frequency. For $k \neq 0$, $I_i^{(k)} = -2T \times C_i \times Vdd \times sin(kw_0\tau/2)/(\tau k^2\pi^2)j$ and $I_i^{(0)} = Gi \times Vdd/2$ (TANG; FRIEDMAN, 2000). For the k-th harmonic the voltages at the clock sinks can be computed by solving the complex linear equation:

$$G^{(k)}\cdot V^{(k)} = I^{(k)} \tag{4.3}$$

Table 4.4: Runtime comparison. Time unit: Seconds. (WANG; KOH, 2007)

Mesh size	Proposed	1K MC	Hspice 100 MC	SWS 100 MC
30 x 30	13	668	3499	3681
50 x 50	40	2003	12298	35417
100 x 100	219	10028	fail	$>$ 3 days
150 x 150	675	32844	fail	$>$ 5 days
200 x 200	1541	89063	fail	$>$ 5 days
300 x 300	6538	$>$ 5 days	fail	$>$ 5 days

In 4.3 G^k is the admittance matrix. The arrival times at the clock sinks can be obtained by letting $V_i(t') = Vdd/2$ and solving for t'. Tipically, only few harmonics are required to achieve sufficient accuracy.

A methodology to compute the admittance matrix as a function of wire widths and arrival times is presented in (WANG; KOH, 2007) but it will not be shown here. (WANG; KOH, 2007) also discusses how to compute the covariance matrix for the clock sinks.

Experimental results show that with more than 33 harmonics the error in the arrival time estimation is close to 0 for a deterministic arrival time computation. The statistical analysis was compared to the Monte Carlo, the proposed methodology has presented an error smaller than 1% in the mean estimation and smaller than 3% for the standard deviation estimation. The authors have shown a great speedup compared to Spice simulation runtime. Table 4.4 shows the runtime comparison among 4 different approaches. It should be noted that the Montecarlo simulation reported in table 4.4 consists of Monte Carlo runs using the frequency domain method proposed by this work, which is considered a golden measure since a single run of the frequency domain method was shown have an error smaller than 1%.

Although the proposed methodology has shown to be accurate in comparison to Spice simulation the error introduced by replacing the inverter by a current source in parallel with a resistor was not studied. Current source models are used as timing model for gates and buffers in combinational circuits, the clock mesh, on the other hand, presents several loops connecting the inverters output. When a skewed signal is applied to the clock mesh, short circuit currents will flow among the different inverters. The current source model may present a large error in the presence of short circuit currents.

Another important aspect overlooked by the author is the effect of increasing the number of elements in the mesh model. In the experiments performed in the paper the author computes arrival times at the intersection of the mesh wires, there are no flip-flops in the model. By not including the flip-flops, the total number of elements in the model is drastically reduced. If the number of elements increases the method proposed in this work could face memory problems.

4.3.4 Clock Skew Analysis via Vector Fitting in Frequency Domain

A frequency domain analysis methodology for fast simulation of clock networks is presented in (ZHANG et al., 2008), this method is called CSAV. The proposed technique is able to speedup the simulation not only for clock meshes but also for the clock tree. The proposed method relies on solving the frequency domain state equation in a few points and then applying a vector fitting technique to derive the rational approximate (GUSTAVSEN;

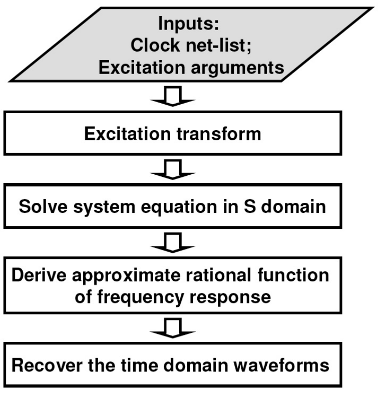

Inputs:
Clock net-list;
Excitation arguments
Excitation transform
Solve system equation in S domain
Derive approximate rational function of frequency response
Recover the time domain waveforms

Figure 4.14: Clock skew analysis via vector fitting flow. (ZHANG et al., 2008)

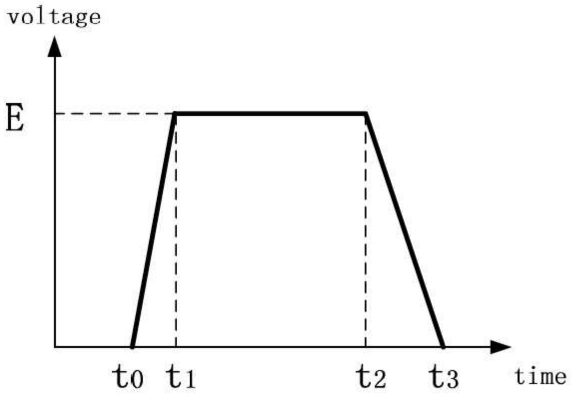


Figure 4.15: Ramp signal waveform. (ZHANG et al., 2008)

SEMLYEN, 1999). Vector fitting and waveform recovery are applied only to the critical sinks of the clock network.

Figure 4.14 shows the flow used to analyse the clock network. The first step relies on transforming input excitations from the time domain to the frequency domain. In order to do that, input stimuli are assumed to be ramps. The Laplace transform for ramp signals are known, the Laplace transform for the ramp signal shown in figure 4.15 is expressed by the equation 4.4. By assuming the input stimuli are ramps, error is introduced in the analysis.

$$Vin(s) = \frac{E}{(t_1 - t_0)s^2}e^{-st_0} - \frac{E}{(t_1 - t_0)s^2}e^{-st_1} - \frac{E}{(t_3 - t_2)s^2}e^{-st_2} + \frac{E}{(t_3 - t_2)s^2}e^{-st_3} \quad (4.4)$$

The next step relies on solving the system equation, shown in equation 4.5, in the frequency domain. Clock buffers are replaced by a linear current/voltage source in parallel/series to a resistor. Solving equation 4.5 is computationally costly due to large circuit size and the matrix inversion operation. In order to minimize the number of matrix operations the system equation will be solved only to selected frequency points.

$$x(s) = (sM + G)^{-1}PV_{in}(s) \quad (4.5)$$

After solving the system equation for the selected frequency points the frequency response is approximated for the intermediate points by vector fitting. Vector fitting is a

Table 4.5: CPU time comparison of CSAV and Hspice (unit: second). (ZHANG et al., 2008)

Test Case	Hspice CPU time	CSAV		Speed-up for
		T_p	$(T_v + T_w)$	$\frac{N_{out}}{N_{total}}$ = 5%
s1423	0.78	0.16	0.029	2.15
s5378	2.08	0.34	0.037	2.07
s15850	22.45	2.39	0.038	4.81
r4	277.15	18.49	0.041	10.55
r5	668.71	46.78	0.067	9.90
m80	338.14	7.55	0.053	13.80
m100	681.17	13.27	0.038	21.10
m120	1342.06	23.67	0.037	26.68
m150	2965.53	43.03	0.037	35.03
m200	9352.85	132.75	0.069	34.54

numerical method for rational approximation in the frequency domain. The vector fitting and the frequency point selection is performed in such a way that the maximum error allowed in the estimation is 1% when compared to Spice.

The inverse Laplace transform is applied after the vector fitting was performed. The output waveform is computed in the time domain using equation 4.6, in which N_a is the order of approximation for vector fitting.

$$v_{out}(t) = \sum_{i=1}^{N_a} c_i e^{a_1 t}, t > 0 \tag{4.6}$$

By assuming that a fixed time step will be used and that N_t time steps are required equation 4.6 can be rewritten to:

$$E_i = e^{a_i \Delta t} \tag{4.7}$$

$$v_{out}(N_t \Delta t) = \sum_{i=1}^{N_a} c_i E_i^{N_t} \tag{4.8}$$

Solving equations 4.7 and 4.8 is less computational expensive than solving equation 4.6 because, since the time step is fixed, the exponential in equation 4.7 needs to be computed only once. In order to speedup even further the analysis of clock networks the authors propose to apply the vector fitting step only to the set of the 5% most critical paths. Non critical paths have a larger slack and therefore can tolerate a larger error on the clock skew estimation.

The proposed methodology was able to achieve larger speedup when compared to hspice simulation time. Error was kept within 1% of the "golden" hspice values. Table 4.5 reports the speedup and runtime values for a set of benchmark circuits. Table 4.5 reports CSAV vs. Hspice runtimes for both, clock trees and clock meshes test cases. The Bounded Skew Clock Tree Routing algorithm proposed in (CONG et al., 1998) was used to create the clock trees. Clock meshes were created using a regular structure.

In the circuits s1234, s3578, s15850, r4 and r5 clock distribution is done by a clock tree only. Circuits m80, m100, m120, m150 and m200 are the same circuits used in the tree-only evaluation with the inclusion of a clock mesh. The CSAV execution time is reported individually for each step, T_p stands for the time to solve the frequency domain equation. $(T_v + T_w)$ is the time required by the vector fitting and waveform recovery, this time is linearly dependent on the number of paths being evaluated. In this experiment only the 5% most critical paths are being treated. CSAV runs up to 35$\times$ faster than the Spice simulation. The largest speedups are observed on circuits containing a clock mesh.

Although CSAV can greatly reduce clock network analysis time with great accuracy this method may not support very large clock meshes. Memory constraints may still prevent the evaluation of large clock meshes. Besides, the inaccuracy introduced by replacing mesh buffers by linear circuits was not studied. It is expected that replacing mesh buffers by linear elements adds large inaccuracy especially when short circuit currents are observed.

4.4 Conclusions

Altough the Sliding Window Scheme (4.2) was the first method proposed to address the problem of clock mesh simulation it still is one of the most advantageous methods. All methods presented in section 4.3 require the user to implement his own simulation environment while the SWS can be implemented with minimum effort using any electrical simulator. However, all the other four methods presented a greater speedup than the SWS.

It should also be noticed that SWS is more accurate than the methods presented in sections 4.3.3 and 4.3.4 since both methods require mesh buffers to be linearized. The SWS is also more scalable than all other methods except the one in section 4.3.2, since only the SWS method and the method in section 4.3.2 use a divide-and-conquer heuristic.

The works presented in sections 4.3.1 and 4.3.2 use model reduction techniques to speedup clock mesh simulation while the works presented in sections 4.3.3 and 4.3.4 use frequency domain techniques. Model order reduction techniques are prone to be more accurate than frequency domain techniques since the mesh buffer doesn't need to be linearized. On the other hand, frequency domain techniques seem to be able to achieve a greater speedup.

5 CLOCK MESH OPTIMIZATION STRATEGIES

Although clock meshes are very important in the design of low skew and variability tolerant clock distribution systems not much work has been made in optimizing it. Besides, most of the work done in the optimization of mesh based networks focus on optimizing a clock mesh without considering the characteristics of the driving tree.

This thesis presents a new approach for clock mesh optimization. Instead of optimizing it considering only the clock sink positions and mesh size distribution profile the optimization is done considering the clock arrival times at the mesh buffers. Two optimization techniques are proposed here focusing on minimizing short circuit currents between different mesh buffers when clock signal presents different arrival times at each mesh buffer.

In the next section we will discuss the short circuit currents between different mesh buffers observed when a skewed signal is applied to a clock mesh. A mesh buffer sizing methodology focused on the reduction of inter-buffer short circuit currents is presented in section 5.3. In section 5.4 a new mesh buffer design that prevents short circuit currents between different mesh buffers is presented. To the best of our knowledge, the works presented in sections 5.3 and 5.4 are the first ones to address the problem of reducing short circuit currents in mesh-based clock distribution architectures.

5.1 Related Works

Previous works have been done for clock mesh optimization. This sections describes two different methodologies to optimize clock meshes. Unlike the methods described in sections 5.3 and section 5.4 the methods described here optimize clock meshes without using any information about the clock network driving it.

The method in section 5.1.1 proposes a mesh buffer placement and sizing algorithm along with a mesh reduction technique which removes some of the mesh redundant edges. A simplified driver model is proposed to speedup clock mesh analysis. Section 5.1.2 presents an algorithm for computing a good initial size for the clock mesh and an improved mesh reduction algorithm based on a sensitivity analysis. The work presented in section 5.1.2 shows a significant improvement in comparison to the method described in section 5.1.1.

5.1.1 Combinatorial Algorithms for Fast Clock Mesh Optimization

This work was published in (VENKATARAMAN et al., 2006). It presents a set-cover based algorithm for mesh buffer placement and sizing along with a mesh reduction technique using survivable network theory. Also, a mesh buffer model able to speedup

Hspice simulation up to 65$\times$ is presented.

For a given buffer library each mesh buffer available in the library is characterized by a given driving strength. The mesh buffer placement/sizing relies on finding a set of mesh buffers and positions that cover the whole set of clock sinks and minimize the sum of mesh buffers input capacitances. Mesh buffers can be placed only on mesh grid nodes (i.e, the spot where a horizontal mesh wire crosses a vertical mesh wire).

The mesh buffer library is limited to a fixed number of mesh buffer sizes. Since mesh and local wiring are not affected by mesh buffer positions and sizes, the only effect of mesh buffer sizing and placement is to minimize the overlaps between the covering regions of adjacent buffers and to assure that there will be no clock sink left uncovered. The amount of overlaps between the cover regions of different buffers happens as a consequence of the limited number of mesh buffers. If unlimited sizes of mesh buffers were available, any set of buffer locations could generate a minimum cost solution if mesh buffers were properly sized.

This can be seen in the following example. *Assume a mesh buffer library L with n mesh buffers, $L = (l_1, l_2, ..., l_n)$. For a given set of m buffers positions $P = (p_1, p_2, ..., p_m)$ the cost of a solution using placement P is given by $cost = \sum_{i=1}^{m} (best\_size(L, i) - opt\_size(i))$, in which $best\_size(L, i)$ returns the best size in L to drive the region buffer i is responsible. If unlimited buffers sizes are available, then $best\_size(L, i) - opt\_size(i) = 0$ and therefore $cost = 0$. This means that for any buffer placement solution the minimum cost can be achieved.*

The second important contribution of this paper is an edge removal algorithm based on survivable network theory. The mesh reduction problem attempts to reduce the total number of mesh edges while keeping a minimum number of buffers connected to each clock sinks through minimum distance paths. Two parameters are used to control the mesh reduction step, l and k. The edges are removed from the clock mesh such that for any given clock sink s_i there are at least k mesh buffer locations connected to s_i through l disjoint paths. Only drivers whose distance from the clock sink s_i is smaller than L_{max} are counted as one of the k buffers. The mesh reduction problem is solved by using any Steiner Network Optimization algorithm. More details can be found in the paper (VENKATARAMAN et al., 2006).

Mesh reduction is a powerful tool to exchange skew and power consumption, however, its applicability is limited. Mesh reduction effectiveness is highly dependent on the number of clock sinks within a clock mesh square. If the clock mesh wiring is denser than the clock sink distribution many edges in the clock mesh are likely to be removed. If the clock sink distribution is denser than the clock mesh most likely no clock mesh edges are going to be removed.

A mesh buffer model was proposed to speedup spice simulation. Mesh buffer modelling is a very complex problem. Besides providing a model that can precisely compute slew times and delays, the mesh buffer model has to accurately model the interaction between the different mesh buffers. The author proposes to use the model presented in (LI; ACAR, 2005). The buffer model is illustrated in figure 5.1. The first stage is composed by a variable capacitance to model mesh buffer input capacitance. The second stage is a linear delay line which is included to model the intrinsic delay of the mesh buffer. The final stage is the driver stage. It is composed by a variable current source in parallel to a variable capacitance. The variable capacitance and the voltage dependent current source are both described by two dimensional look-up tables. Both, capacitance and current, are dependent on the final stage and second stage output voltages.

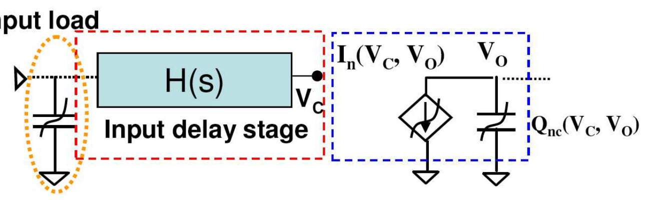


Figure 5.1: Proposed clock driver model. (VENKATARAMAN et al., 2006)

Table 5.1: Buffer model vs. HSPICE comparison. (VENKATARAMAN et al., 2006)

Circuit	# Sinks	Mesh Size	Runtime		Speedup	Max Err (%)	Avg Err (%)
			Hspice	Model			
s9234	135	9×9	10.8	0.23	47.13	4.34	1.49
s5378	165	10×10	3.9	0.41	9.54	2.83	1.36
s13209	500	30×30	145.6	4.59	31.73	7.15	2.01
s15850	566	30×30	84.8	4.86	17.45	4.54	1.34
s38584	1426	40×40	590.9	12.75	46.35	3.63	1.00
s35932	1728	40×40	934.4	15.08	61.96	5.92	1.52
Average	753.3	27×27	295.1	6.32	35.69	4.74	1.45

The proposed mesh buffer has presented a large speed-up with a high accuracy. Table 5.1 shows runtime and accuracy for the proposed mesh buffer model for a set of benchmark circuits. The maximum error observed is of at most 7.15%, while the average error is 1.45%. The average speed-up was about 35×. Since the maximum error can be relatively big the buffer model is not adequate for accurate analysis, however, it can be used to provide a fast timing estimation helping the user to experiment with different mesh reduction parameters.

(VENKATARAMAN et al., 2006) shows experimental data reporting that the set covering algorithm can meet slew constrains within 2.52% with few seconds runtime. As mentioned before the importance of the buffer placement and sizing algorithm decreases as the number of mesh buffer sizes available in the library increases.

The power improvement and slew penalty observed when the mesh reduction algorithm is applied is reported in table 5.2. By looking at the power reduction numbers it can be noticed that the mesh wire reduction algorithm allows the designer to trade between power consumption and clock skew. This paper contributes to allow designers to better tune clock mesh based architectures according to each design constraints. However mesh wire reduction method may only be effective for very dense clock meshes.

5.1.2 MeshWorks: An Efficient Framework for Planning, Synthesis and Optimization of Clock Mesh Networks

(RAJARAM; PAN, 2008) Meshworks presents mesh optimization strategies to further improve the clock mesh design in comparison to (VENKATARAMAN et al., 2006). Meshworks proposes solutions to address the problem of finding a good initial mesh and to achieve a smooth trade-off between power and skew. Experimental results demonstrate that a further improvement of 26% in buffer area, 19% in wire length and 18% in power can be achieved in comparison to (VENKATARAMAN et al., 2006).

Table 5.2: Results for mesh reduction. (VENKATARAMAN et al., 2006)

Circuit	Wire length (mm)			Power (mW)			Skew (ps)		
	Orig.	Red.	% Imp	Orig.	Red.	% Imp	Orig.	Red.	% Imp
s9234	30.4	27.2	10.5	7.13	6.7	6.1	110.97	124.1	-11.8
s5378	32.3	24.9	22.85	7.81	6.72	13.96	103.23	128.06	-24.0
s13207	153.5	109.5	28.62	30.3	23.8	21.47	86.94	95.81	-10.2
s15850	164.7	100.8	38.8	33.2	23.8	28.13	86.36	106.64	-23.5
s38584	371.9	262.5	29.41	78	60.9	21.99	122.72	130.7	-6.5
s35932	427.9	321.3	24.91	92.4	74.3	19.58	118.99	134.65	-13.2

The initial clock mesh planning algorithm address the problem of obtaining a clock mesh with minimum routing and buffer resources such that design constraints are satisfied. The basic algorithm used to find the initial mesh configuration is shown in figure 5.2.

```
   Input: $L_{max}$, $L_{min}$, $S_{max}$
   Output: $m$, $n$
1  Get $m$, $n$ such that total wire length from equation 5.1 is $\approx L_{min}$ ;
2  Using the current $m$, $n$ obtain $L_{tot}$ using equation 5.1;
3  if $L_{tot} \geq L_{max}$ then
4      Print "Relax design constraints";
5      Quit;
6  end
7  Obtain value of $Sk_{bound}$ from Equation 5.2;
8  if $Sk_{bound} \leq S_{max}$ then
9      Return $m$,$n$;
10 else
11     Increment values of $m$, $n$ by 1 and go to step 2;
12 end
```

Figure 5.2: The top-level algorithm of selecting the initial mesh size. (RAJARAM; PAN, 2008)

The m and n parameters define respectively the number of lines and columns of a clock mesh. The algorithm start by setting m and n to the value that leads to the smaller total capacitance by using equation 5.1. Then, total wire length is computed. If total wire length, L_{tot} is greater or equal to the maximum wire length constraint, L_{max}, design constraints should be relaxed and the procedure is aborted. Otherwise, a skew upper bound estimation is computed to the given mesh size, $m{\times}n$, using equation 5.2. If the skew upper bound is smaller or equal to the maximum skew constraint the algorithm successfully finishes, otherwise the number of columns and lines is incremented by 1 and the new mesh wire length is computed starting the process again.

$$L_{tot} = L_{mesh} + L_{stub} = m{\cdot}X_{bound} + n{\cdot}Y_{bound} + \sum_{i=1}^{N} L^{i}_{stub} \tag{5.1}$$

$$Sk_{bound} = [Max(D_p(CL_p^{max})) - Min(D_q(CL_{q-1}^{max}))] + Delay(D_{max}) + IntDel(L_{stub}^{max}, C_L^{max}) \tag{5.2}$$

Equations 5.1 and 5.2 were obtained from (RAJARAM; PAN, 2008). Equation 5.1 computes the total wire length, L_{tot}, for a given mesh size $m \times n$ for a design in which the clock sinks are spread over a region delimited by X_{bound}, Y_{bound}. Equation 5.2 presents an approach to estimate the worst case skew given the maximum possible difference between the delays of two mesh buffers, the maximum distance between a clock sink and the nearest mesh buffer and the maximum stub length, L_{stub}^{max}. The first part of equation 5.2, $[Max(D_p(CL_p^{max})) - Min(D_q(CL_{q-1}^{max}))]$, estimates the maximum skew caused by the different delays of mesh buffers. The second term, $Delay(D_{max})$ computes the skew caused by the distance from the clock sink to the clock driver, while the term $IntDel(L_{stub}^{max}, C_L^{max})$ adds the skew caused by the different delays presented by a short stub driving a small load versus a long stub driving a large load.

Besides proposing the algorithm to find a good initial mesh size Meshworks proposes a modified cost function to be applied to the set cover algorithm. (VENKATARAMAN et al., 2006) searches for any solution that minimizes the total sum of mesh buffer input capacitances while Meshworks proposes to prioritize the insertion of smaller mesh buffers instead of larger ones. By prioritizing the insertion of smaller buffers mesh buffers more buffers will be inserted in the clock mesh and therefore mesh buffers tend to be closer to clock sinks. Reducing the distance between mesh buffers and clock sinks reduces the attenuation due to the clock mesh wire resistance and capacitance. The modified cost function and a more detailed explanation about the effect of using distributed buffers instead of larger lumped ones can be found in (RAJARAM; PAN, 2008).

A network sensitivity based approach was proposed to reduce the number of redundant edges in the clock mesh. If the sensitivity of clock arrival time at the clock sinks with respect to the width of mesh edges are computed the mesh edges with smaller influence on arrival times can be removed. The most straightforward way to compute the sensitivity of clock arrival times with respect to the mesh edges width is by perturbing each parameter individually and evaluate its effect on delays. Meshworks propose a more computationally efficient method which is based on (LEEDS; UGRON, 1967). Also, Meshworks proposes to use Elmore delay to compute the clock arrival times sensitivity with respect to the mesh wire widths.

The required steps to perform the sensitivity analysis according to (RAJARAM; PAN, 2008):

1. Identify and lump clock sink clusters. This reduces the total number of elements in the mesh model speeding up the sensitivity analysis.

2. Replace all mesh buffers by a linear model.

3. Obtain Elmore delays sensitivities for every lumped sink cluster w.r.t. to every mesh edge. LU factorization is used to solve the linear system enabling the reuse of the factorization among every mesh edge.

4. Sort mesh edges in a sensitivity increasing order. Mesh edges are removed such that removed edges are at least N mesh nodes away from each other. This requirement prevents interactions between removed mesh segments, i. e., the only mesh segment far away from another removed mesh segment can be removed to make sure that the mesh segment sensitivity was not affected.

Table 5.3: Summary of optimization results for all test cases.

Method	% BA Red.	% WL Red.	% PWR Red.	μ_{skew} Avg.	σ_{skew} Avg.	F_{max} MHz	% F_{max} Red.
MO	0.00	22.94	21.11	18.09	3.72	971.58	1.53
MP&S	20.74	19.88	19.84	13.13	2.58	979.55	0.72
NSMO	14.25	31.63	28.89	21.35	2.60	971.65	1.52
MPSO	26.92	42.53	39.80	30.66	4.38	958.05	2.90

5. By removing mesh edges the total load driven by each mesh buffer is reduced. A post processing sizing step is performed to reduce the size of mesh buffers whose covering region overlaps with another mesh buffer.

To reduce inaccuracy of Elmore delays estimates, a buffer model containing 2 resistors and 2 capacitances and a signal source is proposed. Experiments comparing the accuracy of the proposed model with actual inverters show that the proposed buffer model leads to an error in the order of $3ps$ in the sink's delays during a spice simulation. The author claims that the proposed buffer model can better model the iteration between the different mesh buffers, however more experiments are required to sustain this claim. The proposed buffer accuracy was evaluated only when a perfectly synchronized signal is applied to the clock mesh. When clock is perfectly synchronized in the mesh buffer the interaction among the mesh buffers is minimal. The reported error is expected to increase if a skewed signal is applied to the clock mesh.

Using Elmore delay to compute clock sinks delays is also very inaccurate. Elmore delay is meant to single driver nets since it does not account for the short-circuit current among the different mesh buffers. Although the error using Elmore delay to estimate clock sink delays can be high it can still be a good heuristic to estimate the clock sinks arrival times sensitivities with respect to the mesh wires widths.

The summary of the improvement obtained by Meshworks is shown in table 5.3. The method identified as "MO" identifies the method of (VENKATARAMAN et al., 2006). The mesh planning strategy proposed in the Meshworks framework is represented by "MP&S". The network sensitivity method for mesh reduction technique is represented by method "NSMO". The total contribution of the mesh planning strategy used along the network sensitivity mesh reduction is identified by "MPSO". Improvement factor is computed with respect to the a manually designed clock mesh.

Meshworks has presented a 19% wire length reduction, 18% power reduction and 26% smaller buffer area when compared to (VENKATARAMAN et al., 2006)(MO). Experimental data suggests that the network sensitivity based mesh reduction is more efficient than the mesh reduction algorithm presented in (VENKATARAMAN et al., 2006). It can also be noticed that the mesh planning strategy is very useful to achieve a good initial trade-off between chip resources and skew.

The clock mesh planning strategy proposed in this paper is interesting and able to produce a good initial clock mesh solution but the clock skew in the input of the clock mesh should not be considered zero. If clock is skewed in the input of mesh buffers the skew observed at the clock sinks will be higher, therefore it should be accounted in equation 5.2. Considering the mesh input skew in equation 5.2 will greatly affect the mesh size choice.

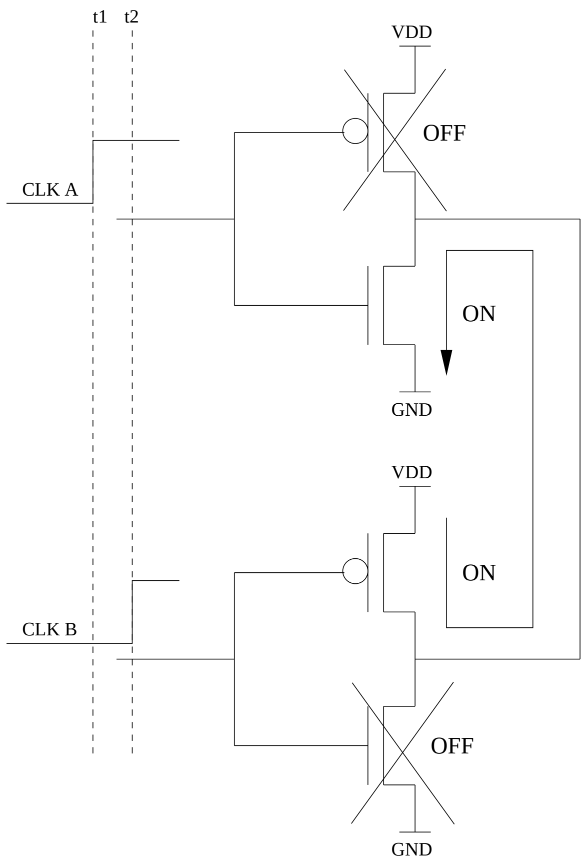


Figure 5.3: Short circuit example.

Besides of considering mesh input clock skew in the mesh planning step it should also be considered at the mesh edge sensitivity computation. The clock sinks delay sensitivities with respect to a clock mesh edge wire width are affected by the clock mesh arrival times at the mesh buffers.

Although Meshworks has made large improvement in comparison to the work of (VENKATARAMAN et al., 2006) it does not address the clock arrival time characteristics at the mesh buffers. In practice, the clock signal will present a large skew in the mesh buffer inputs, consequently the clock mesh behavior will be dominated by the short circuit currents among the mesh buffers. If clock skew at the mesh buffers is not considered the optimized clock meshes generated by (RAJARAM; PAN, 2008) and (VENKATARAMAN et al., 2006) will not perform well in practice.

5.2 Motivation

Both works described in section 5.1 ignore the effect of clock skew in the mesh buffer inputs. In practice, clock signal arrival times at mesh buffers varies a lot from buffer to buffer, i.e., a large skew is present in the clock mesh buffers inputs. When a skewed signal is applied to a clock mesh short circuit currents are generated.

A short circuit is generated every time a mesh buffer switches before or after the others. Consider the simple case of Figure 5.3 where the outputs of the two inverting buffers are shorted. If two different clock signals CLK_A and CLK_B are applied to two different buffers whose outputs are shorted, there will be a time when one buffer will connect its output to Vdd while the other will be connected to the GND. During this

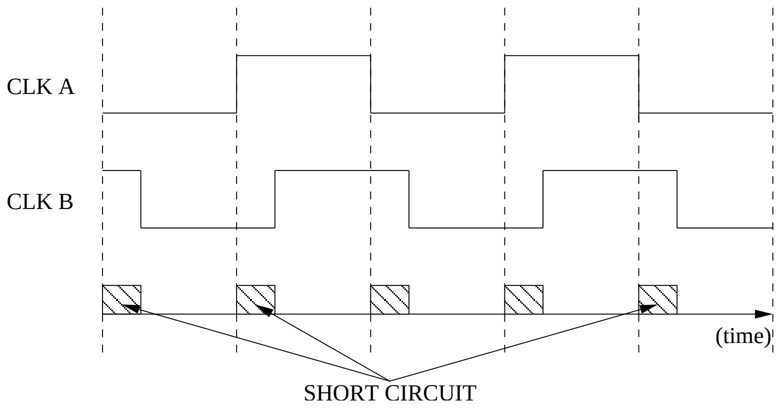


Figure 5.4: Short circuit due to skew.

period short circuit currents are created between the outputs of the buffers as shown in figure 5.3 and figure 5.4. In a clock mesh this situation happens in a larger scale among mesh buffers with different clock arrival times.

If a clock mesh is optimized without considering the effect of short circuit current the optimized clock mesh will not present the expected performance and power consumption in practice. For instance, both works presented in section 5.1 present algorithms to remove edges from a clock mesh with minimum increase in the clock skew. If inter-buffer short circuit currents are not accounted for during the edge removal the effect of removing mesh edges becomes unpredictable. For the same reason, input skew at the mesh buffers should be accounted for in the formulation presented in section 5.1.2 to compute a bound in the maximum output skew for a clock mesh.

Although short circuit currents are responsible for slowing down fast mesh buffers and speeding up slow buffers they should be minimized. In this section we show the effect of inter-buffer short circuit currents on clock power consumption, clock slew and clock skew. We show that if inter-buffer short circuit currents could be minimized without removing the redundancy present in a clock mesh clock mesh power consumption, slew and skew can be improved.

5.2.1 Power Consumption Due To Inter-Buffer Short Circuit Current

The most severe drawback of clock meshes is the high power consumption. Reconvergent paths present in the mesh structure not only use additional wiring resources and increase clock capacitance, but also enable a high short circuit power consumption.

Ideally, if the clock signal had no skew in mesh input buffers , i.e., clock edge arrives at all mesh buffers at the same time, the clock power consumption would only be due to charging and discharging of circuit capacitances. Under this assumption, equation 5.3 is a good approximation for the total power dissipated by the mesh, as presented in (KANG; LEBLEBICI, 1996). Assuming that f is the clock frequency, C is the total mesh and flip-flops capacitance, and Vdd is the supply voltage, the power consumption due to capacitance charge/discharge is given in equation 5.3.

$$P = f \times C \times Vdd^2 \tag{5.3}$$

Now, let us consider the case when the clock arrives at different times in the mesh buffer inputs, i.e., the input skew is non-zero. Then, we expect that short circuit power will become significant.

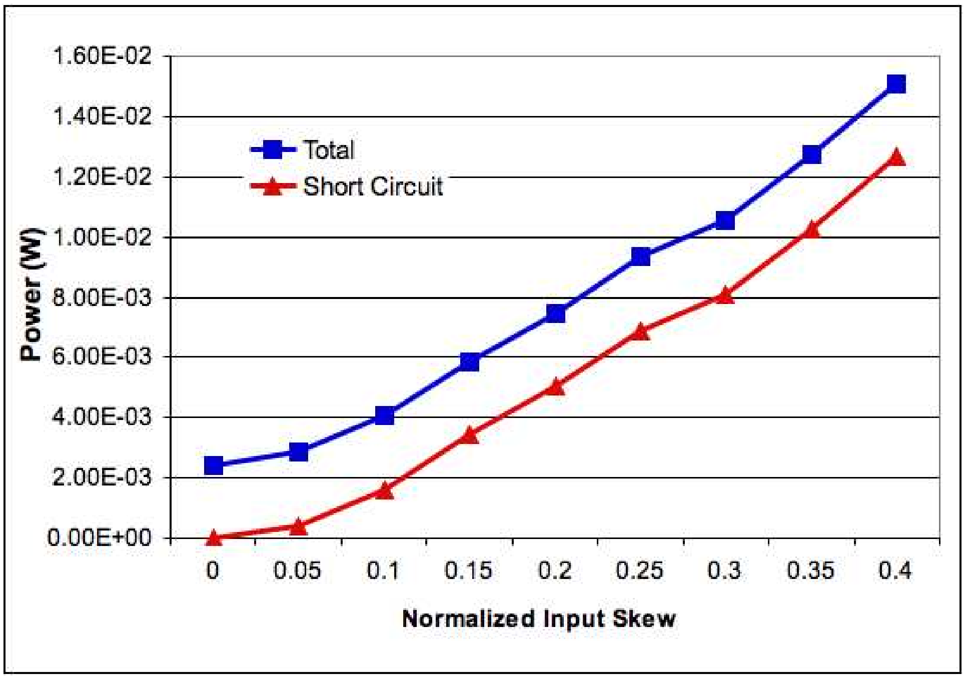


Figure 5.5: Total Power and Short circuit Power vs. Maximum Input Skew.

We performed the following experiment to determine the contribution of short circuit power to total mesh power as a function of mesh input skew. A set of 50 randomly generated circuits containing 100 flip-flops distributed over a 50.000 μm^2 area was simulated. A 4x4 mesh was placed over the circuit and flip-flops were connected directly to mesh wires. Each mesh node was driven by a single mesh buffer. The input of each mesh buffer was connected to a separate clock source. All the clock sources were skewed with randomly generated arrival times varying within k% of the clock period where k varies from 0% to 40% with 5% steps. The random generated arrival times model the combined effect of nominal skew and delay variations at the driver tree. The clock period was set to $1GHz$. Figure 5.5 shows average circuit power over the 50 benchmark circuits (the upper curve) and short circuit power (lower curve) as a function of the maximum input skew. It can be seen that a maximum input skew of only 15% doubles total mesh power consumption, i.e., short circuit power is 50% of total mesh power. For higher input skews, more than 80% of total clock mesh power is due to short circuit.

Thus, the ability of the mesh to smooth out undesired delay variations comes at the price of high short circuit power consumption. As input skew increases, short circuit power also increases, and so does total power. If the input skew is sufficiently large, the short circuit power accounts for most of the clock power. In other words, *any effort to reduce the total clock mesh power must attempt to minimize the short circuit power.*

5.2.2 Skew Due To Inter-Buffer Short Circuit Current

A multi-driver net is a net which is driven by more than one driver, e.g., a clock mesh. When a short circuit current is flowing between the drivers of a multi-driver net, the voltage value observed within the net ranges from 0 to Vdd. If short circuit between different buffers is temporary, e.g., when caused by a skewed signal, it will degrade output transition time and may increase skew observed in net sinks.

Figure 5.6 shows three equally sized buffers whose outputs are shorted. All buffers

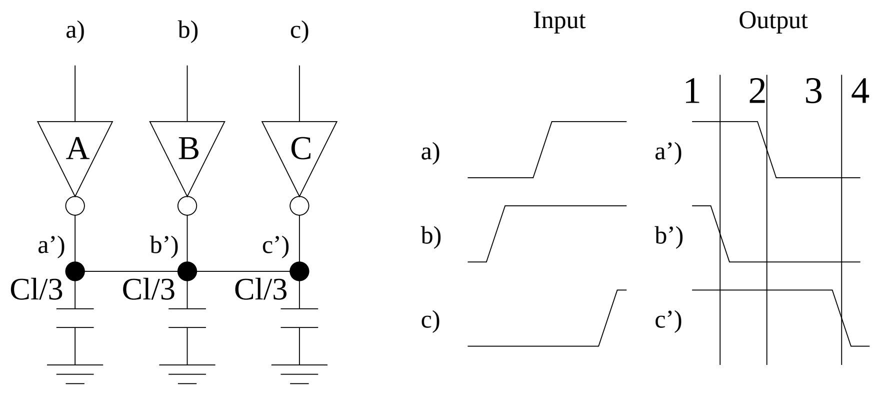


Figure 5.6: Improving slew by buffer sizing.

are driving the same signal, but the arrival time at each buffer input is different. As can be seen in the waveform representation, the signal arrives first on node b, than on node a and at last on node c. By looking at the output waveform for each inverter in figure 5.6, we can see 4 different behaviors in the output net:

1. At this moment all nodes are driving the same signal. There is no short circuit current flowing between the buffers and the output signal is steady.

2. Next, node b' switches. A short circuit current flows between node b' and the other nodes. Node b' is trying to set the output net to 0, while the other nodes are driving this net to 1. The output net voltage is still closer to 1 than to 0.

3. When node a' switches then two nodes are driving the output net to 0 while only node c' is driving it to 1. As a result the output net voltage begins to move closer to 0 than to 1.

4. Finally when node c' switches all nodes are driving the output net to 0 and it finishes switching.

Now let us assume that buffers B and C are smaller than buffer A. In this configuration the behavior in regions 2 and 3 is affected. In region 2, the short circuit current flowing between the buffers is going to be reduced due to the reduced driving strength of buffer B. The voltage during this stage is going to be held closer to 1 than before. During region 3 only node c' is driving 1 while the other two are driving 0. Since buffer C is undersized the value observed in the output net is going to be closer to 0 than in the former case. When node c' switches it will finish charging the output net faster since it will be almost fully charged. By analyzing the example in figure 5.6 we see that the different buffers in a multi-driver net compensate each other through short circuit currents. We will discuss now how we can improve transition time and output clock skew by better controlling short circuit currents.

5.2.2.1 Improving transition time

The circuit proposed on figure 5.6 was simulated on Spice to show that skewed signals generate large inter-buffer short circuit currents that degrade rise and fall transitions. To evaluate the effect of short circuit currents we are going to reduce the driving strength of buffers in which clock arrival time arrives earlier or later than the mean arrival time. By

Table 5.4: Reducing buffer sizes.

Buffer	Arrival time	Size	Reduced Size
A	$500ps$	10	10
B	$375ps$	10	7.5
C	$625ps$	10	7.5
Skew		Fall	Rise
Size		$243ps$	$259ps$
Resized		$239ps$	$246ps$

reducing the driving strength of buffers at which clock arrives before or after the mean arrival time the short circuit currents between different buffers that drive the same net is going to be reduced. Short circuit currents are a consequence of different arrival times at buffers whose outputs are shorted as shown by figure 5.3. If the driving strength of buffers causing the short circuit currents is reduced the short circuit current itself will be reduced. It should be noticed that reducing the driving strength of a buffer member of a multi-driver net reduces the overall driving strength driving the net and therefore is prone to increase the net transition time.

An experiment was made to evaluate the effect of reducing short circuit currents at the transition time of a multi-driver net. A load of $180fF$ was equally distributed among nodes a', b' and c'. Initially, all three buffers were sized equally, using the FO4 rule. The output transition time was measured. The experiment was redone using smaller sizes for buffers B and C. Experimental setup data and delay values measured can be seen in table 5.4.

Data reported in table 5.4 shows that, when one or more buffers outputs are shorted, output transition time can be improved by reducing the sizes of buffers with the largest and the smallest arrival times. This experiment assumes a zero resistance line shorting the output of the buffers. The reason why transition time was reduced when the sizes of buffers B and C were reduced was explained in figure 5.6 example. Buffer A drives a signal arriving with an arrival time which is the mean between the arrival times in the buffer B and C inputs. By reducing the sizes of buffers B and C the effect of buffer A is increased and the conflict (i. e., short circuit currents) between A and the other signals is reduced.

5.2.2.2 Improving skew

Let us assume that resistors are inserted in figure 5.6 circuit between nodes a' and b', b' and c', and c' and a'. When the buffer outputs are separated by resistors, different waveforms will be observed in the output of each inverter. In this case, not only the transition time is reduced when buffer with skewed inputs are undersized, but also output skew.

As resistance between the buffers increases, the effect of undersizing skewed buffers is lost. For instance, assuming that an infinite resistance is added among the buffers output all three buffer outputs are independent, i. e., early or late arrival times do not compensate each other. If skewed buffers are undersized transition time increases and so does skew between buffers.

Using the setup shown in table 5.4, an experiment was performed to evaluate the effect

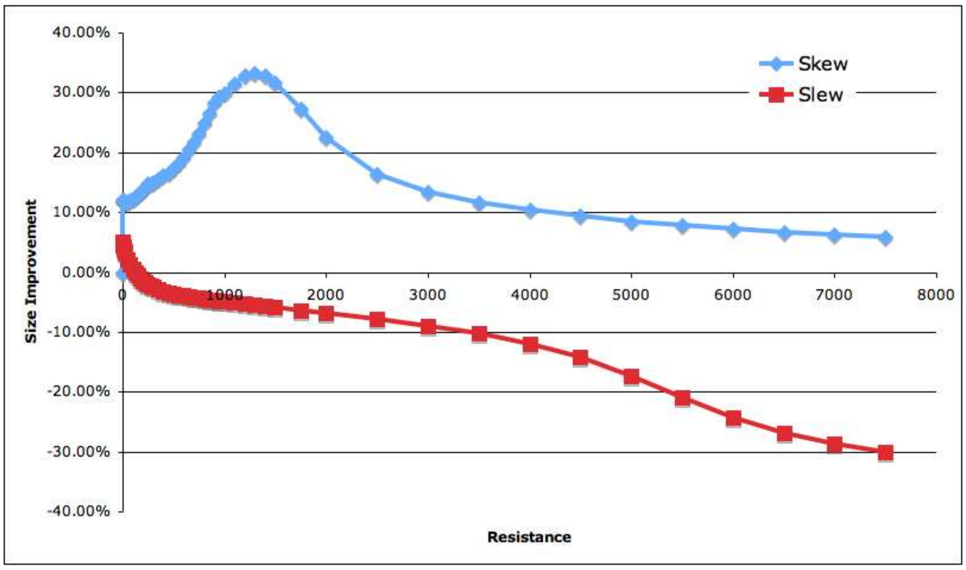


Figure 5.7: R effect on skew and slew reduction.

of inserting a resistance with value R among all buffers output. By varying the value of R the effectiveness of reducing the size of skewed buffers changed. Figure 5.7 shows the improvement factor for both, maximum skew and slew, for different values of resistance between the nodes when the sizes of buffers B and C are reduced.

Figure 5.7 shows that slew improvement monotonic decreases as the resistance between the nodes increases. This happens because, as the resistance increases, each buffer is able to charge only the capacitance directly connected in its output. When the size of skewed buffers is reduced then the time to charge its load capacitance increases. Skew reduction between the different outputs initially increases until 1300Ω. After this point skew reduction decreases as resistance increases, until the skew reduction is zero. No skew increase was observed. Skew reduction presents a non monotonic behavior because skew is very small for small resistance values, hence relative skew improvement is also small (e.g. when R is zero skew is zero therefore improvement is also zero). This experimental setup is a closer model to a mesh structure than figure 5.6 experimental setup since mesh wires have a resistive behavior.

The experiments reported in this section are an indication that reducing short circuit current between different drivers of a multi-driver net can improve output transition time and skew. It was also shown that the reduction in the short circuit current can be achieved by reducing the sizes of buffers in which the driving signal arrives either too early or too late. The sizing of mesh buffers to reduce clock skew and power will be discussed in section 5.3

5.3 Mesh Buffer Sizing

Clock mesh buffers are initially sized before the generation of the global clock network. At this moment, no information about the timing of the global clock network is available. The initial sizing step can only take into account design information from downstream levels. Both methods presented in section 5.1 propose solutions to find a better set of sizes for the mesh buffers considering only the downstream clock distribution. The approach proposed in this section relies on a post processing sizing step to tune the

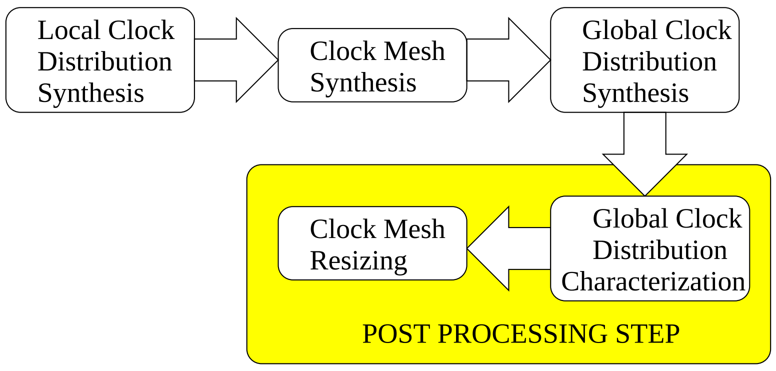

Local Clock Distribution Synthesis
Clock Mesh Synthesis
Global Clock Distribution Synthesis
Clock Mesh Resizing
Global Clock Distribution Characterization
POST PROCESSING STEP

Figure 5.8: Mesh buffer sizing flow.

mesh sizes according to the delays in the global clock distribution. Figure 5.8 shows a flow illustrating how the post processing sizing step is added in the clock synthesis flow.

Sizing mesh buffers according to the arrival times in mesh buffer inputs requires accurate timing estimation in the global clock network. If mesh buffers are sized it will affect global clock network timing. To prevent this to happen two actions have to be taken:

- *Mesh buffer sizes can only be reduced*. To increase the size of mesh buffers it would be required to increase the size of buffers in the global clock network. Global clock network can not be changed at this point.
- When mesh buffers sizes are reduced, delays on the global clock network change. To prevent this to happen *dummy capacitances must be added* so that the initial capacitance seen by the global tree is matched with the new mesh buffer size plus the added dummy capacitance.

Considering the two constraints stated above, two different algorithms were proposed to size clock mesh buffers taking into account the global clock network timing information. The first one considers only the mean arrival time estimated in the input of mesh buffers, while the other considers the probability density function (PDF) estimated in the input of each mesh buffer.

5.3.1 Mean Sizing

Given a set At of arrival times in the mesh buffer inputs, a set of mesh buffer sizes M and a maximum reduction factor MR, the set of new sizes for each mesh buffer, $Mnew$, is computed using the algorithm shown in figure 3. In the input of each mesh buffer a capacitance C_i is added to match the previous capacitance of M_i. The value of C_i is computed as $C_i = (M_i - Mnew_i) * unitary\_cap$.

This algorithm linearly decreases buffer sizes according to the difference between clock arrival time in each mesh buffer input and mean arrival time. Figure 5.10 illustrates how the multiplication factor $reduction$ is computed for each mesh buffer according to its input arrival time. From figure 5.10 it can be noticed that new mesh buffer sizes are always going to be smaller than initial ones.

5.3.2 Probabilistic Sizing

Variations affect performance of every designed chip. Chip behavior is not completely predictable, process and environmental variations may speedup or slow down gates and

```
   Input: $At$,$M$,$MR$
   Output: $Mnew$
1  $Atm = compute\_mean(At)$;
2  $Atdist = 0$;
3  foreach Mesh Buffer i do
4      $Atdist$ = max($Atdist$,$|At_i - Atm|$);
5  end
6  foreach Mesh Buffer i do
7      $reduction$ = $1 - MR * (|At_i - Atm|/Atdist)$;
8      $Mnew_i$ = $M_i * reduction$;
9  end
```

Figure 5.9: Mean sizing algorithm.

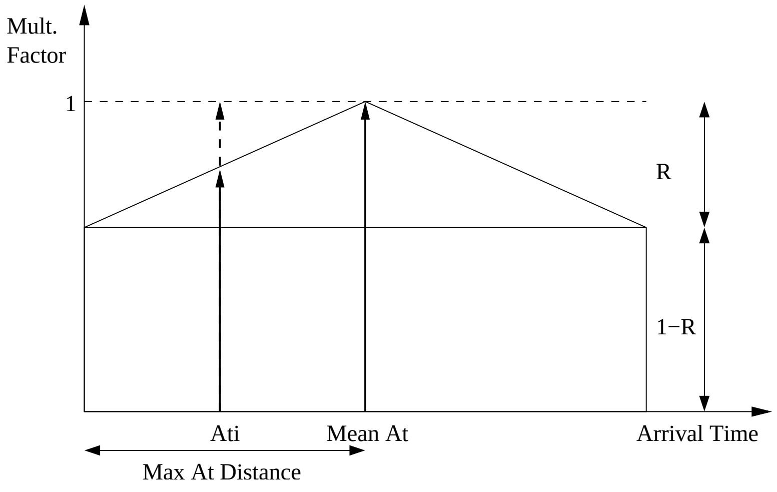


Figure 5.10: Mean sizing algorithm illustration.

wires. It is necessary to model chip delays as PDFs to account for variations effects. The probabilistic sizing algorithm proposed here uses Gaussian probability distributions to model clock arrival time in the mesh buffer inputs. Although only Gaussian distributions were used in this work, any sort of PDF can be used, as long as its cumulative density function (CDF) is known.

The probabilistic sizing method reads a set At of arrival times in the mesh buffer inputs, a set of mesh buffer sizes M, a maximum reduction factor MR and a sizing control value Th as inputs. It produces a new set of mesh buffer sizes, $Mnew$, as output. Dummy load is added to match the previous input capacitance value for each mesh buffer. The probabilistic sizing algorithm is described in the algorithm shown in figure A.11.

```
Input: $At$,$M$,$MR$,$Th$
Output: $Mnew$
1  $Atm = compute\_mean(At)$;
2  $Ats = compute\_mean\_sigma(At)$;
3  $CDF_{BASE} = CDF(Atm, Ats)$;
4  $A_{BASE} = CDF_{BASE}(Atm + Th) - CDF_{BASE}(Atm - Th)$;
5  foreach Mesh Buffer i do
6      $CDF_i = CDF(Atm_i, Ats_i)$;
7      $A_i = CDF_i(Atm + Th) - CDF_i(Atm - Th)$;
8      $reduction = 1 - MR * (A_i/A_{BASE})$;
9      if $reduction > 1$ then
10         $reduction = 1$;
11     end
12     $Mnew_i = M_i * reduction$;
13 end
```

Figure 5.11: Probabilistic sizing algorithm.

The probabilistic sizing algorithm begins by computing a base probabilistic distribution. This distribution is built using the mean arrival time and the mean standard deviation over all arrival time distribution in the mesh buffer inputs. The A_{BASE} area, comprised between $At_m - Th$ and $At_m + Th$, is computed. Then, for each arrival time distribution, the A_i area, comprised between $At_m - Th$ and $At_m + Th$, is computed. The reduction factor is given by $1 - MR * (A_i/A_{BASE})$. When arrival time distributions have different standard deviations, it can happen that $A_i > A_{BASE}$. In that case, reduction factor is bounded to 1. Figure 5.12 illustrates how this algorithm works.

5.3.3 Experimental Setup

The proposed sizing algorithms were applied on clock meshes designed for random generated circuits. The effect of each sizing algorithm in comparison to sizing all mesh buffers according to the FO 4 rule was evaluated by performing a set of electrical simulations using Hspice. Clock skew, clock slew (i.e. transition time) and power consumption were measured.

We have performed our experiments over 8 random generated circuits. The information for each circuit is given in table 5.5. Although mesh sizes, die sizes and flip-flop count are reduced, flip-flop density is in accordance with the industrial design used in the experiments in section 3.1.2. Experiments were performed using a 90nm technology. To

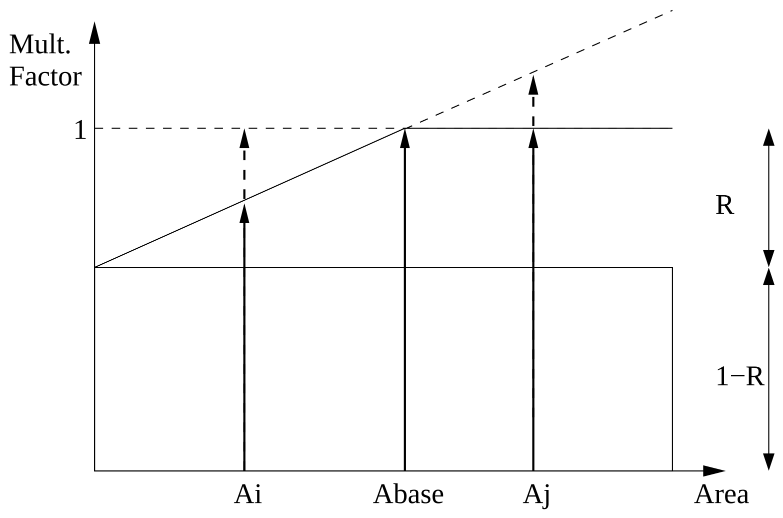


Figure 5.12: Probabilistic sizing algorithm illustration.

Table 5.5: Benchmark set.

Circuit	Mesh size	Die size (μm)	#FFs	FF Density
c1	4×4	500×500	250	1000/mm^2
c2	4×4	500×500	500	2000/mm^2
c3	4×4	500×500	750	3000/mm^2
c4	4×4	800×800	600	937.5/mm^2
c5	4×4	800×800	900	1406.25/mm^2
c6	4×4	800××800	1200	1875/mm^2
c7	8×8	800×00	900	1406.25/mm^2
c8	8×8	800×800	1200	1875/mm^2

reduce the amount of data reported and to increase the significance of our experiments, values for clock skew, clock slew and mesh power consumption were averaged among the benchmark circuits.

To have a better control over the clock skew applied to the clock mesh, the global clock network was replaced by artificially delayed voltage sources. Each voltage source produces a slope with a rise/fall time of 8% of the clock cycle. To generate a more realistic waveform each voltage source drives 2 cascade inverters sized according to the FO4 rule.

Clock signal arrival time at the mesh buffer was modeled as a Gaussian distribution, therefore it is described by a mean value and a standard deviation. Each arrival time mean represent the clock skew that is predicted at the design stage, while the sigma determines the amount of skew caused by process and environmental variations.

Each mean arrival time was randomly generated. For each mesh buffer the mean arrival time is generated by randomly generating a number between 0 and 1 and then multiplying it by the desired mean input skew value. The random numbers between 0 and 1 are generated a single time for every circuit, therefore, for different input skew configurations, the ratio between each buffer mean arrival time is equal to the ratio between the mean skew for each configuration.

The three different arrival time configurations described in table 5.6 were simulated.

Table 5.6: Arrival time characteristics.

Mean At Skew (%)	Sigma (%)
0.1	0.03
0.2	0.07
0.4	0.1

Skew values model three different situations, a low skew setup, a typical skew setup and a high skew setup. The skew values for each one of the setups in table 5.6 were chosen according to the author's experience. Mean arrival time skew and sigma values are given as a percentage of the clock cycle. A 1GHz operating frequency was assumed for all experiments.

We have manually set the best parameters for both sizing algorithms. The maximum reduction factor, MR for both algorithms was set to 0.5. The sizing control value for the probabilistic sizing algorithm, Th, was set to 2.5% of the clock period. For simplicity we have named the mean sizing algorithm presented in section 5.3.1 as $msize$, while the probabilistic sizing algorithm presented in section 5.3.2 was named $psize$.

5.3.4 Experimental Data

Table 5.7 reports the power, skew and slew improvements for all the benchmark circuits using the two proposed sizing algorithms. A mean input skew of 20% of the clock period was assumed. Clock mesh input arrival times are modeled using Gaussian distribution with a standard deviation equal to 7% of the clock period.

A significant clock skew improvement can be achieved using the proposed sizing algorithms at the cost of increasing clock slew. Power consumption in the clock mesh was also reduced significantly, 11.89%. By comparing the results for circuits $c1$, $c2$ and $c3$ to circuits $c4$, $c5$, $c6$, $c7$ and $c8$ it can be noticed that the sizing methodology is more efficient for circuit with a smaller area. It can also be noticed that larger improvements were obtained on circuits with a smaller number of flip-flops. Both facts can be explained by the behavior of short circuit currents. A larger area means higher wire lengths and wire length increase decreases short circuit currents. Whenever short circuit currents are decreased the efficiency of our method is reduced. For the same reason the efficiency of the proposed sizing algorithm is reduced when more flip-flops are added. Adding more flip-flops increases the time required to charge the capacitance at the neighbourhood of a mesh buffer and therefore delays the occurrence of short circuit currents.

It can also be noticed in the experiments reported in table 5.7 that the proposed sizing algorithms present a greater advantage for denser clock meshes. Short circuit currents are more significant in denser clock meshes since mesh buffers are closer.

A second experiment was made averaging the improvements over the 8 benchmark circuits. This experiment allows a better comparison between both sizing algorithms. We have also evaluated the effect of different input skew characteristics on the improvements of the sizing algorithms. The averaged data can be observed in figures 5.13, 5.14 and 5.15. Each color represents a different input skew setup and each set of bars represents a different algorithm used to size the mesh buffers in the post-processing step. The average reduction in the mesh buffer sizes is reported in figure 5.16. The average mesh buffer reduction for a circuit containing n mesh buffers is computed as per equation 5.4, in

Table 5.7: Sizing Improvement for a 20% mean input skew with 7% sigma.

Circuit	Sizing Method	Power (mW)		Skew (ps)		Slew (ps)	
		Total	Imp. (%)	Total	Imp. (%)	Total	Imp. (%)
c1	typ	3.09	-	49.51	-	210.33	-
	$msize$	2.74	11.25	23.29	52.95	247.80	-17.82
	$psize$	2.72	11.89	19.83	59.94	271.78	-29.22
c2	typ	4.66	-	75.83	-	212.48	-
	$msize$	4.23	9.24	45.06	31.62	253.80	-19.45
	$psize$	4.20	9.82	43.21	43.02	276.52	-30.14
c3	typ	6.21	-	85.33	-	215.36	-
	$msize$	5.73	7.70	60.91	28.62	257.42	-19.53
	$psize$	5.70	8.22	57.53	32.58	279.28	-29.68
c4	typ	7.62	-	115.77	-	239.55	-
	$msize$	7.16	6.03	92.48	20.12	283.88	-18.51
	$psize$	7.14	6.34	95.37	17.62	302.94	-26.46
c5	typ	10.06	-	139.13	-	255.10	-
	$msize$	9.59	4.62	107.06	23.05	300.68	-17.87
	$psize$	9.57	4.86	107.47	22.76	317.76	-24.56
c6	typ	12.68	-	158.81	-	279.07	-
	$msize$	12.23	3.55	130.96	17.53	325.38	-16.60
	$psize$	12.21	3.73	133.78	15.76	341.13	-22.24
c7	typ	16.20	-	130.75	-	261.49	-
	$msize$	15.00	7.39	94.36	27.83	292.76	-11.96
	$psize$	14.73	9.02	77.54	40.70	320.55	-29.36
c8	typ	18.91	-	152.42	-	274.92	-
	$msize$	17.70	6.37	114.34	24.99	311.08	-13.15
	$psize$	17.43	7.81	103.69	31.97	334.03	-21.50

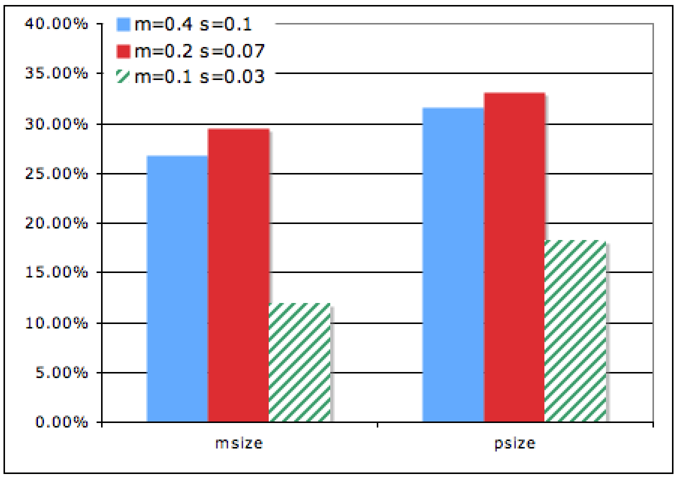


Figure 5.13: Average Skew improvement.

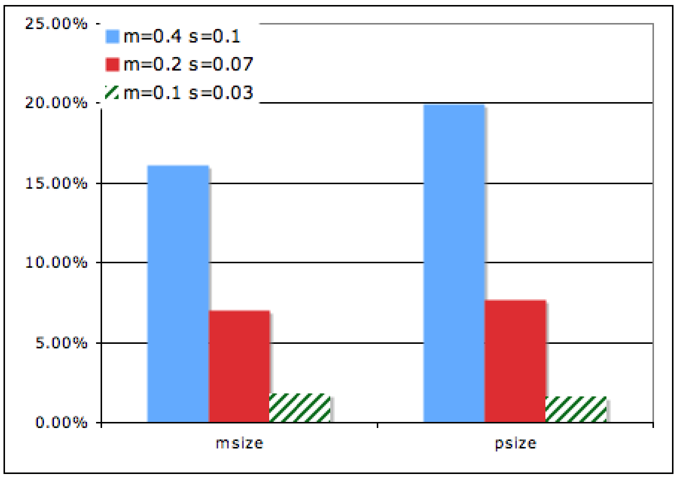


Figure 5.14: Average Power improvement.

which new_m_i and old_m_i are, respectively, the new and old buffer sizes for mesh buffer i.

$$1 - (\sum_{i=0}^{n} new\_m_i / \sum_{i=0}^{n} old\_m_i) \tag{5.4}$$

An average reduction of 33% on clock skew was achieved. Our method was able to reduce power consumption by 20%. On the other hand, undersizing mesh buffers has increased clock slew up to 26%.

The slew penalty observed in these experiments is a direct consequence of the limitation of allowing mesh buffers only to be reduced. The overall driving strength of the mesh buffer is reduced and therefore clock slew is increased. If the average mesh buffer size was kept constant, i.e. if mesh buffers could be increase to compensate for undersized mesh buffers, the clock slew observed at the clock sinks would decrease since short circuit currents would be reduced.

By comparing the two sizing algorithms proposed we can observe that the probabilistic algorithm, $psize$, performs a more aggressive sizing and, in general, achieves a smaller clock skew and a smaller power consumption at the cost of a higher clock slew. This is explained in figure 5.16 which shows that for the two higher input skew cases the total

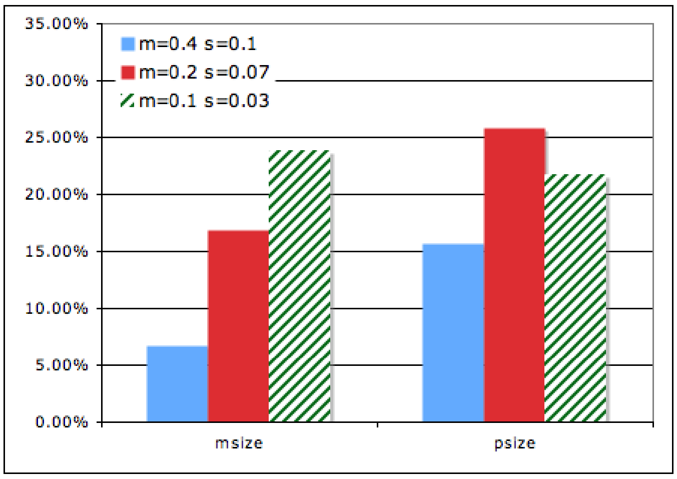


Figure 5.15: Average Slew penalty.

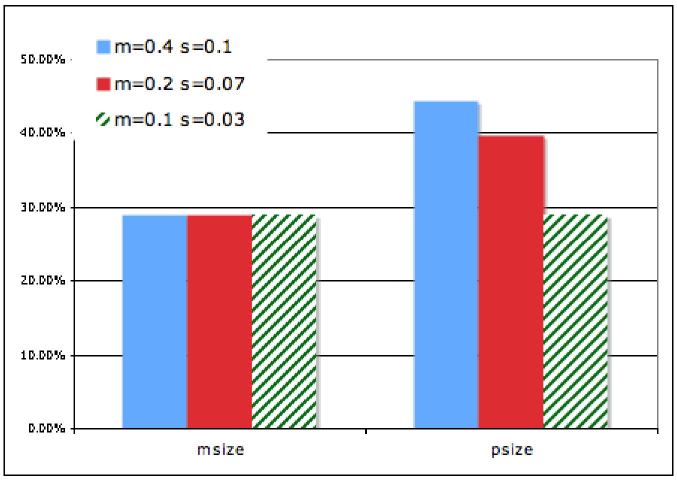


Figure 5.16: Average Undersize.

mesh buffer sizes reduction was higher to the $psize$ algorithm than to the $msize$.

Figure 5.16 also shows the behavior of the $psize$ algorithm with respect to the different input skew values. For the probabilistic sizing algorithm, the higher the input skew is the larger will be the reduction in the mesh buffer sizes. The mesh buffer size reduction factor is determined according to the difference between the base area, A_{BASE} and relative area for each buffer i, A_i, as shown in figure 5.12. The largest is the interval used to compute A_{BASE} and A_i the smaller is the overall buffer size reduction. When the arrival time standard deviation increases the sizing control value, Th, should be increased in the same proportion in order to keep the average mesh buffer size reduction constant. Since Th was kept fixed in spite of the input arrival time standard deviation, when an arrival time with a large standard deviation is applied the area between $\mu - Th$ and $\mu + Th$ will vary less with respect to μ than when the standard deviation is small.

It can also be noticed that the average mesh buffer sizes reduction for the $msize$ algorithm is constant in spite of the input skew. This is explained by the way that the arrival times were generated. The total mesh buffer size reduction is determined only by the mean arrival times at each mesh buffer. Since all arrival times were generated by scaling a set of random generated values between 0 and 1, the relative distance between the arrival times is the same and, therefore, the effect of mesh buffer sizing is the same.

The best input skew case to compare both algorithms is the mean input skew of 10% with a 3% sigma. By looking at figure 5.16 we can see that the total buffer size reduction was almost the same for both algorithm for this input skew. The average clock skew reduction for this input skew situation is higher to the $psize$ algorithm than to the $msize$ algorithm, 18.16% versus 11.90%. The average power reduction is almost the same for both algorithms, 1.77% for the $msize$ algorithm and 1.57% for the $psize$. The slew increase for the $msize$ algorithm is 23.86% while the $psize$ algorithm presents a slew increase of 21.73%.

5.3.5 Conclusions

Clock mesh power consumption and skew can be improved at the expense of clock slew degradation by reducing the mesh buffers sizes whose inputs are skewed. The proposed algorithms were shown to be efficient for improving clock skew and reducing clock mesh power consumption at the expense of degrading clock slew. Both algorithms were tested under a set of three different input arrival time configurations and were effective for every configuration.

By looking at the data reported in table 5.7 we have seen that both sizing algorithms are more effective for circuits with a small die area. This is in accordance with the arguments presented in section 5.2.2 since smaller die area means smaller wire resistance and hence greater improvement. We also noticed that the gain of both sizing algorithms grows for denser clock meshes with less clock sinks per mesh square. Those observations suggest that the sizing algorithm is not accounting for the low pass filtering characteristics of the clock mesh. In order to account for it the mesh sizing should be performed locally, considering only the interaction among the mesh buffers within a given distance threshold.

By analyzing the average improvement for both algorithms for different input skew characteristics we could see that the $psize$ algorithm presents a larger buffer size reduction than the $msize$ algorithm and therefore it also presents a large power and skew reduction with a high clock slew increase. The $msize$ algorithm presents a constant mesh buffer sizing reduction in spite of the clock skew characteristics while the $psize$ algorithm is

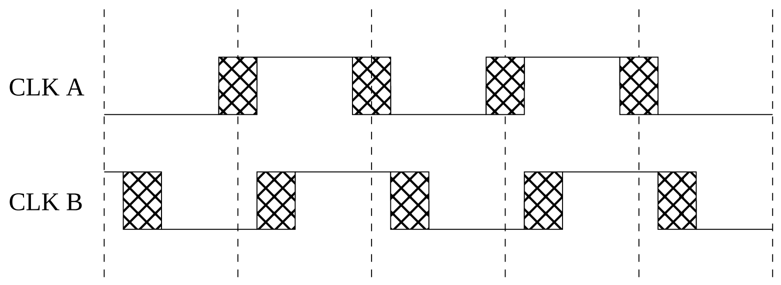


Figure 5.17: High Impedance Time.

affected by the standard deviation of the arrival times at the mesh buffers and the value of the Th parameter.

The $msize$ algorithm can be easily tuned since its behavior does not depend on the input skew characteristics while the $psize$ algorithm requires a more detailed analysis to chose the best parameters for the algorithm. On the other hand, when both algorithms were set with equivalent parameters the $psize$ has produced a more efficient sizing.

5.4 A New Mesh Buffer Design

Section 5.2 has demonstrated the importance of reducing short circuit currents between the different buffers in a clock mesh. This section presents a new design for mesh buffers that reduce short circuit currents between the different mesh buffers. It relies on implementing a break-before-switch inverter: before setting the output to ‘1‘ in a rise transition the output is held in a high impedance state for a fraction of the clock period, the same is done for a fall transition.

5.4.1 Fast Turning Off, Slow Turning On Heuristic

Short circuit currents between the different mesh buffers flow during the time that any two mesh buffers drive opposite values, this situation is shown in figure 5.3. This sort of short circuit current can be prevented by forcing mesh buffers to stay some time in a high impedance state before switching to the final value. Let us assume now that instead of switching from 0 to Vdd, the inverters stay for a while in a high impedance state. When at least one of the two inverters is in the high impedance state, there will be no short circuit between the inverters. This situation is illustrated in Figure 5.17, in opposition to the previous situation illustrated in figure 5.4.

A high impedance period can be achieved by delaying the input signal differently for PMOS and NMOS plans of an inverter. Assuming that for a simple inverter the input signal has two delays (one for each logic level) from the clock source to the PMOS transistor gate: $d0_P$ and $d1_P$, and two delays to the NMOS transistor gate: $d0_N$ and $d1_N$, the inverter can have two different high impedance periods, one when the output switches from 0 to 1, $Ztime_{rise}$ and another when it switches from 1 to 0, $Ztime_{fall}$. The high impedance times observed at the inverter output are determined by equations A.3 and A.4. It is required that the high impedance time be positive. Therefore, $d1_N$ must be greater than $d1_P$ and $d0_P$ must be greater than $d0_N$.

$$Ztime_{fall} = d1_N - d1_P \tag{5.5}$$

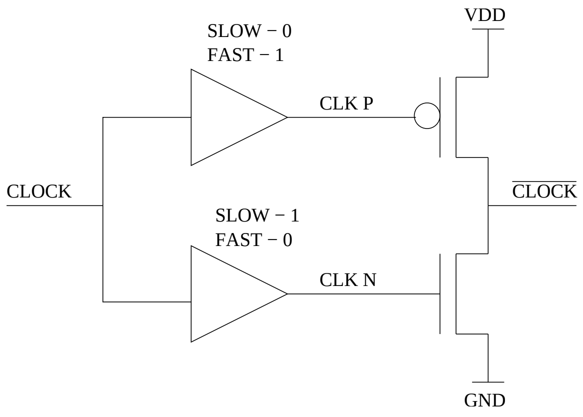


Figure 5.18: A high impedance inverting buffer.

$$Ztime_{rise} = d0_P - d0_N \tag{5.6}$$

By adding a high impedance time to mesh buffers, it is expected that the short-circuit time and consequently the total mesh power will be reduced significantly. Further, if the mesh buffers are properly sized and the high impedance time is carefully chosen, the clock skew and clock slew should also improve, as shown in section 5.2.2. This can be explained as follows. Let CLK A and CLK B be two clock signals arriving at two buffer inputs, as shown in figure 5.3. Assume that CLK A rises before CLK B. With the addition of the high impedance period, rise times for both signals, CLK A and CLK B, are delayed (figure 5.17). However, now the time to charge the capacitance may be much shorter, since when CLK B is in high impedance, the capacitances are being charged by the CLK A buffer.

5.4.2 Electrical Implementation

Based on the idea that adding a high impedance time reduces power consumption and improves output skew, we designed a new mesh buffer. This buffer implements the strategy of slowing down the signal that turns a transistor on and speeding up the signal that turns it off. To realize this, we added two delay elements: one between the input and the PMOS transistor gate, and the other between the input and the NMOS transistor gate. A delay element can be implemented in several different ways. We implemented it with a buffer. The buffer connected to the PMOS is designed such that it propagates the logic-level one fast and the logic-level zero slow, while the buffer connected to the NMOS has the opposite behavior. The resulting high impedance mesh inverter is shown in Figure 5.18.

Each buffer delay element is a two-inverter cascade; the resulting mesh buffer with 10 transistors is shown in Figure 5.19. Although the number of transistors has increased, we have a much finer control on the delays; large values of high-impedance time can be realized easily. In addition, to slow down some transistors, they are converted to higher threshold voltage versions, and their channel lengths are tuned.

To balance the delays of different mesh buffers, it is necessary to size them according to the load they drive. Due to mesh loops, it is difficult to accurately compute the load driven by each buffer. We approximate it by the load of the flip-flops and wires in the surrounding region. For simple, single-stage inverters, the fanout of 4 (F04) rule is used. We use an extension of the FO4 rule to determine transistor sizes. To determine the

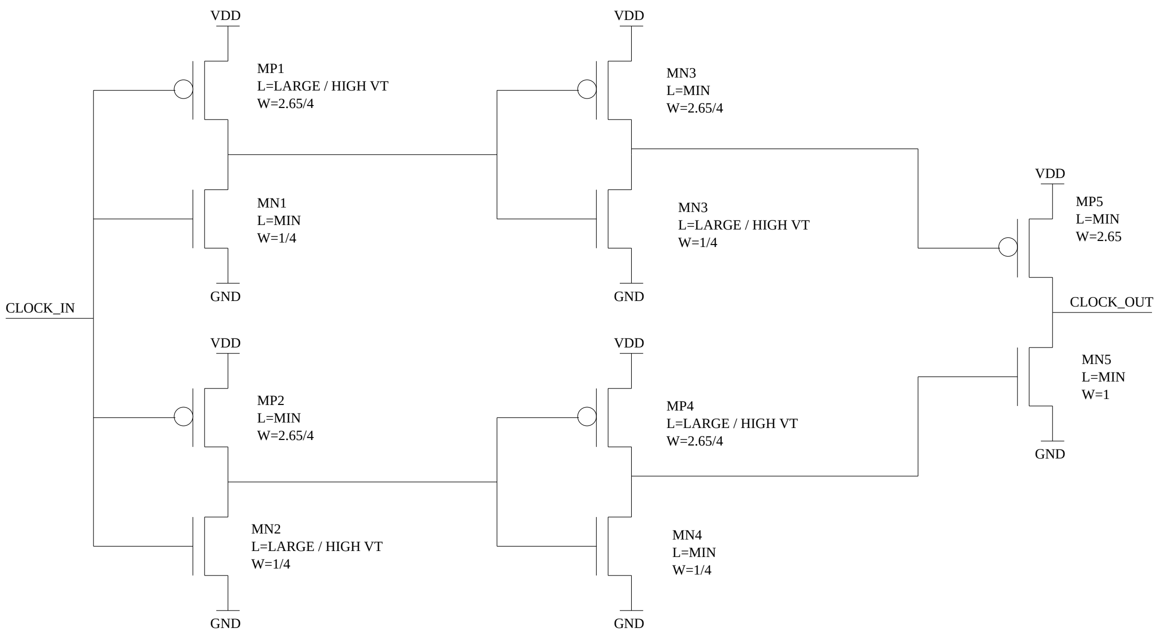


Figure 5.19: Electrical Scheme for Tri-State Buffer.

relative sizes of the NMOS & PMOS, their different channel carrier mobilities for P and N transistors are taken into account. The PMOS transistors are 2.65× wider than the NMOS transistors. The sizes for a unitary size mesh buffer are shown in Figure 5.19.

5.4.3 Applicability and Limitations

To understand the applicability and limitations of this work it is necessary to have clear in mind the differences between the concepts of mesh input skew and mesh output skew. Mesh input skew is the maximum difference among the clock signal arrival times at the input of the mesh buffers. After propagating through the clock mesh the clock signal reaches the sequential elements. Mesh output skew is the maximum clock signal arrival time difference observed at the clock sinks.

Clock meshes are used to provide an output skew much smaller than the mesh input skew. It should be observed that the mesh input skew depends only on the clock network used to drive the clock signal from the clock root to the mesh buffers, while the output skew is determined by clock mesh properties (e. g. mesh line density, buffer sizes, mesh wire widths...). Our methodology is limited by the *INPUT SKEW* characteristics and *NOT* by the output skew. As can be observed in section 5.4.4 our methodology presents greater benefits when applied to designs which present a large mesh *INPUT SKEW*.

Clock meshes are used in designs in which clock skew, due to both, variations and design mismatch, is required to be very small. In other words, clock meshes are used whenever its input skew is large and it needs to be reduced. From the experiments reported in section 3.1 we known that the clock mesh input skew can be roughly 10× larger than the clock mesh output skew (remember that this is determined by each clock mesh characteristics). Since in microprocessor designs, the final clock skew is close to 10% of the clock period, it is very reasonable so assume that the input skew of a clock mesh will be between 10% to 40% of the clock period.

It should also be noticed that as the clock frequencies increase the high impedance time required to prevent mesh short circuit currents gets shorter, hence becoming easier to be implemented. Therefore our methodology can easily be applied to high performance circuits since less delay is needed to create the high impedance time, as shown in equations A.3 and A.4.

Another important aspect that needs to be mentioned is that although the high impedance buffer requires more elements than a simple inverter it effect on clock jitter and clock latency is expected to be small. This happens because clock delay and jitter introduced by the new mesh buffer will be much smaller than the jitter and delay introduced by the clock tree.

5.4.4 Experimental Setup

To verify the effectiveness of this proposal, two set of experiments were performed. The first set (5.4.5) was designed to show that the idea presented in section 5.4.1 is able to reduce the short circuit power and improve skew and slew. The second set (5.4.6) demonstrates that the buffer design proposed in section 5.4.2 can implement the idea presented in section 5.4.1.

To make a fair comparison between different input skew values we have performed a set of fifty simulations for each configuration. In each simulation a different flip-flop distribution was randomly generated.

For both sets, we use a chip containing 100 flip-flops, with an area of 50.000 μm^2. A 4x4 mesh covers the chip area, with a mesh buffer at every mesh grid node. Since we have performed a lot of simulations, we chose a small design instance to reduce the overall simulation time. The flip-flop density and mesh granularity are in line with the industrial design used in the experiments in section 3.1.2.

As mentioned earlier, to model different path delays in the tree driving the mesh, we introduced (input) skews between the clock arrival times in the mesh buffer inputs (thereby abstracting out the tree). Given a maximum input skew value, arrival times in different mesh buffer inputs are generated by a random number generator, respecting the maximum skew.

For several of our experiments, we report normalized input skew values. Since skew is generated by random delay number, in practice the observed input skew can be smaller than the reported values. Reported input skew values represent an upper bound estimation for the input skew. Cases when input skew is significantly different from the value expected are smoothed out by the set of fifty experiments performed for each configuration.

5.4.5 Methodology Verification

To verify our proposed idea, it is sufficient to model the mesh inverting buffer in figure 5.18 with the two output transistors connected to two independent inputs: i.e., a PMOS, a NMOS, and two input signals CLK P & CLK N: CLK P connected to the PMOS gate and CLK N to the NMOS gate. Both inputs are identical, except that they are phase shifted. The phase shift models the difference in the delays of the clock signal through the two delay elements and determines the high impedance time for the buffer, as per equations A.3 and A.4.

In the experiments, we varied the high impedance time of each mesh buffer by inserting appropriate phase shift between the two inputs. For each value of the high impedance time (normalized according to the clock period) – 0.0, 0.2, 0.4 and 0.6 – we measured average power, skew and slew as a function of the normalized mesh input skew. It should be noticed that the high impedance time accounts for the time the fall and rise high impedance times. Figure 5.20 shows how the average total mesh power changes for different input skew values. Figure 5.21 reports how the input skew affects the mesh

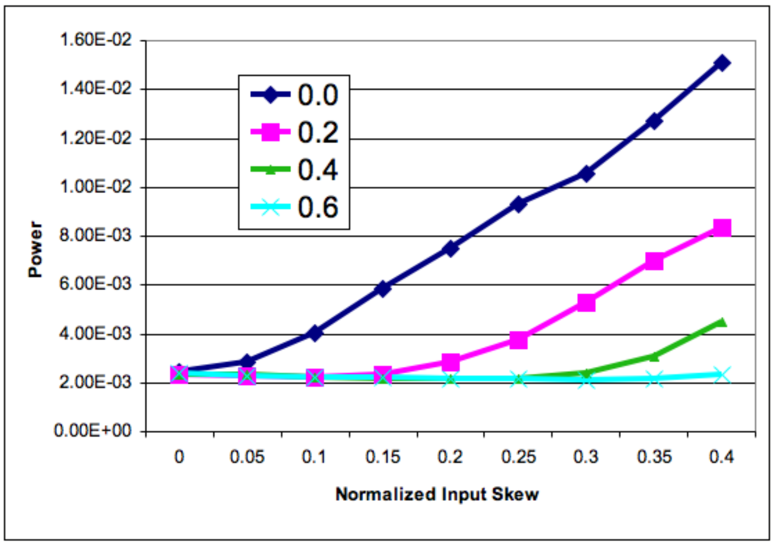


Figure 5.20: Power vs. Input Skew for delays clock.

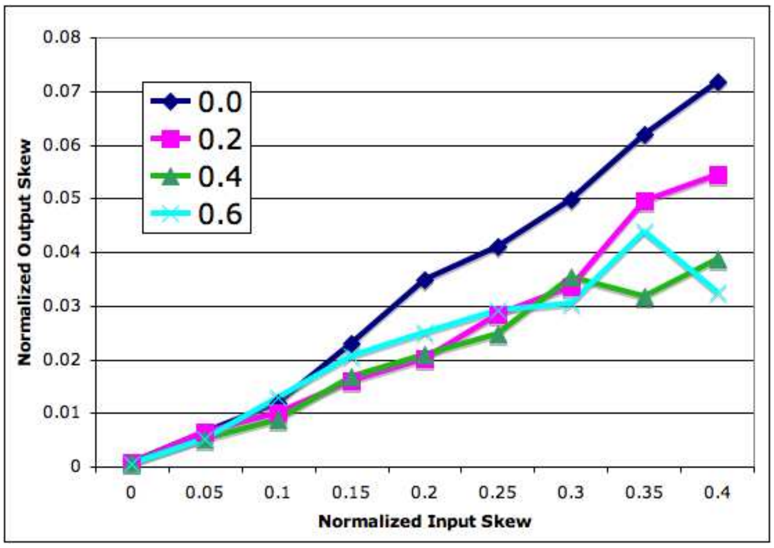


Figure 5.21: Output Skew vs. Input Skew for delays clock.

output skew. Finally, figure 5.22 demonstrates how output slew is affected by input skew. The legend in each chart represents the value of high impedance time in relation to the clock period. Thus each curve line in these charts shows the inverter behavior for a particular value of high impedance time. Note that the 0.0 curve represents the behavior of the traditional inverter.

It can be seen that by adding the high impedance time, total clock power, clock skew at the sequential elements and clock slew are all reduced. Figure 5.20 shows that the total power reduction was very significant. For an input skew that is 10% of the clock period, total power reduction was about 45%. For larger input skew values the maximum power reduction obtained was 84%. Figure 5.20 shows that output skew was also reduced in most cases. For an input skew of 10%, the output skew was reduced by 25%. The maximum reduction in output skew was achieved for an input skew of 40% and a high impedance time of 60%; in this case output skew reduction was 54%. The improvement in relation to output slew can only be noticed for input skew values larger than 15% of

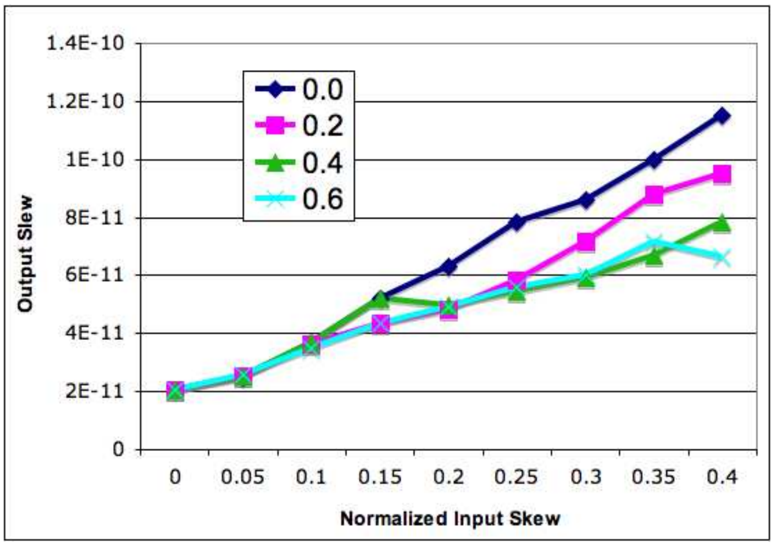


Figure 5.22: Output Slew vs. Input Skew for delays clock.

the clock period. Nevertheless for input skew values smaller or equal to 15% of the clock period the largest slew penalty was smaller than 2% and therefore can be ignored. A close relation between the slew and output skew improvement as a function of input skew can be noticed by comparing figures 5.22 and figure 5.21. This behavior is in accordance with the ideas presented in section 5.2.2.

It can be noticed that bigger improvements can be obtained when larger input skew values are applied to the clock mesh in combination to long high impedance time. This can easily be explained by the short circuit currents present in the clock mesh. When a signal with a large skew is applied to the clock mesh the power, output skew and slew penalties are higher since the larger input skew increases short circuit currents between the different mesh buffers. Long high impedance times can prevent short circuit currents for a longer time and therefore provide a higher improvement for large input skew values in comparison to short high impedance times. On the other hand increasing the high impedance time reduces the time available for mesh buffers to charge the clock mesh and clock sinks parasitic capacitances. The maximum high impedance time must guarantee that there is enough time for the clock mesh capacitances to be fully charged.

5.4.6 Buffer Verification

To verify whether the buffer proposed in section 5.4.2 can reduce power, skew and slew, we used the same set-up as in section 5.4.5, except that now we used the actual mesh buffer implementation shown in figure 5.19. This mesh buffer has an overhead of 8 transistors over a simple inverter. This overhead results in a power penalty that was not taken into account in the results presented in section 5.4.5.

Four different configurations of the inverter were evaluated, by scaling the channel lengths of the selected transistors by a factor of 1, 3, 5, and 7 over the minimum length. This scaling changes the high impedance time of the buffer. The legend of each chart denotes the increased transistor channel length for each configuration. The value “reg” is for the traditional two-transistor inverter.

Figure 5.23 demonstrates that the proposed mesh buffer was able to reduce the overall power consumption despite the transistor overhead. Figures 5.24 and 5.25 present the results for skew and slew. For an input skew of 10%, the maximum power reduction was

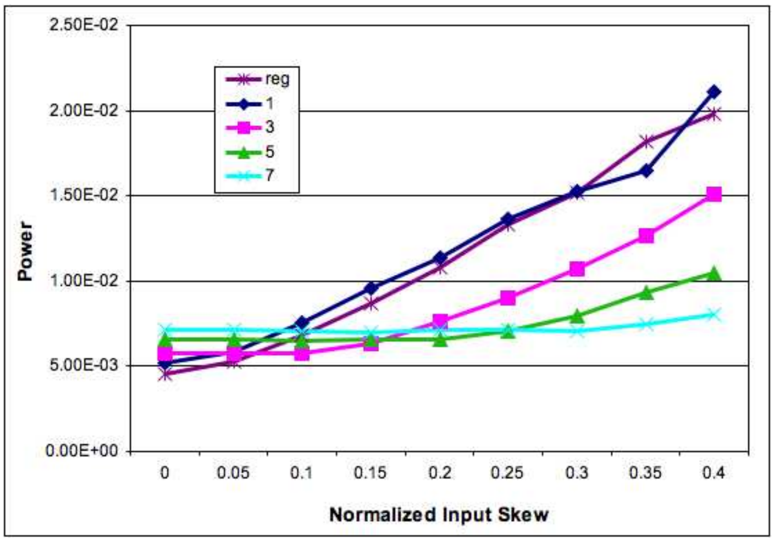


Figure 5.23: Power vs. Input Skew for proposed buffer.

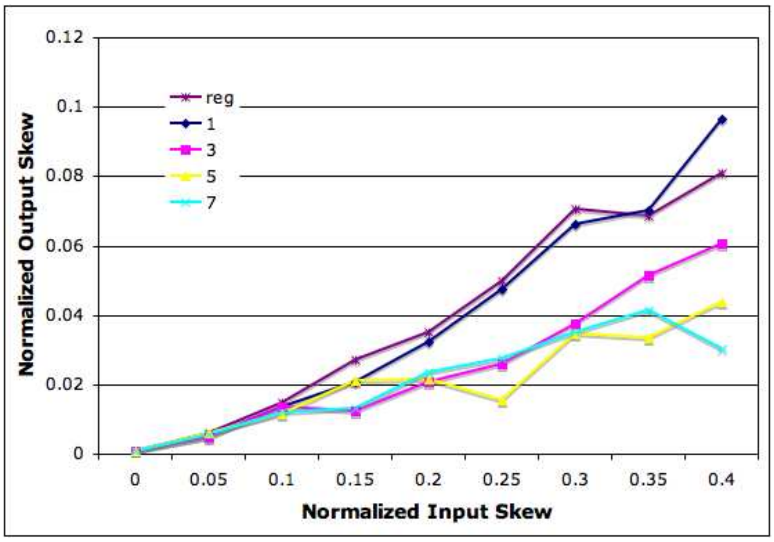


Figure 5.24: Output Skew vs. Input Skew for proposed buffer.

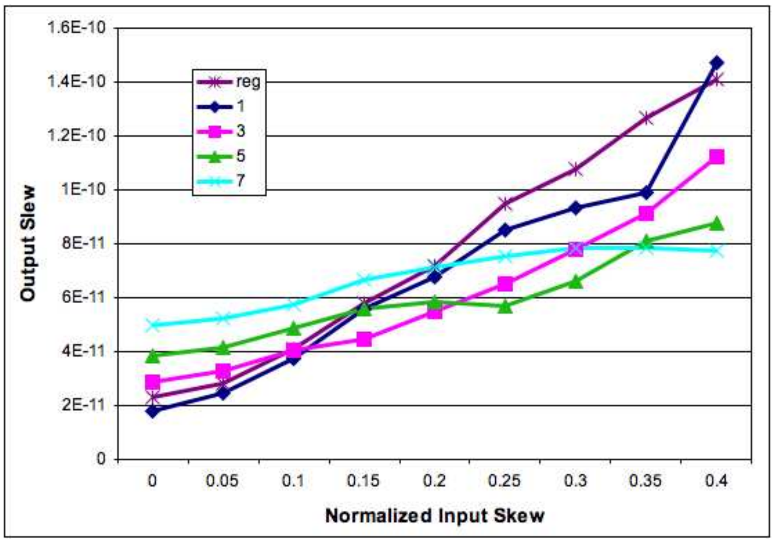


Figure 5.25: Output Slew vs. Input Skew for proposed buffer.

15%. However, for an input skew of 40%, power reduction of 59% was obtained. A small power overhead can be noticed for small input skew values. The power overhead was caused by the higher number of transistors present in the high impedance buffer design.

The slew reduction for an input skew of 10% was 1%. A significant slew penalty was observed for input skew values smaller than 10%. The increased channel length transistors present a much larger channel resistance which increases the transition times inside the mesh buffer which was responsible for producing a larger clock slew at the clock sinks. It should be noticed, however, that larger channel length mesh buffers present an output clock slew which is less dependant on the input skew applied to the clock mesh. For a 40% input skew value the slew improvement was about 43% when a mesh buffer with the maximum channel length is used. In practice the skew in the mesh buffer inputs is expected to be about 20% of the clock period.

Although clock slew has increased for small input skew values the same behavior was not observed for the skew measured at the clock sinks. The overhead associated to the new mesh buffer do not degrade clock skew for small input skew values. For large input skew values the proposed mesh buffer has decreased the output skew in more than 60%. In opposition to the results presented in section 5.4.5 it is not possible to see a relation between the slew and skew improvements as a function of the input skew. In the experiments reported in section 5.4.5 clock slew was a consequence of the large input skew. When the high impedance time was added the short circuit currents were reduced and therefore clock slew and skew at the clock sinks were also reduced. In the experiments reported in this section the clock slew was dominated by the increase in the channel lengths inside the proposed mesh buffer, and therefore it is not affected by the reduction in the short circuit currents as the output skew is.

5.4.7 Leakage Analysis

The most important requirement related to the clock signal is to assure that clock is glitch free. Considering the proposed heuristic to reduce short circuit currents within the clock mesh, one may think that a glitch can be generated during clock switching due to leakage at the clock sinks.

It should be noticed that gate leakage actually helps the capacitance charge/discharge process and therefore can not cause glitches. Leakage could only cause a glitch when it tries to discharge the clock mesh while it is being charged (or to charge it while it is being discharged). This situation only happens after the clock signal crosses 50% of Vdd during a transition. During the first half of the transition, leakage helps the transition, only after the clock signal has crossed $Vdd/2$ is that the gate leakage pushes the clock signal against the transition direction. If gate leakage is in the same order of magnitude as the current which is provoking the transition and, for any reason, the leakage current varies, it could create a glitch in the clock signal.

In a traditional clock mesh, the mesh buffer is constantly being driven by several mesh buffers. In that scenario, leakage current is negligible when compared to the inverter currents. Unlike standard inverters the proposed mesh buffer switches to a high impedance state before switching from 0 to 1 (and from 1 to 0). Assuming a situation when clock signal is almost completely synchronized in the input of mesh buffers but for a group of n mesh buffers in which clock arrives early. Assuming that the high impedance time imposed for each mesh buffer is $400ps$ (40% of the clock period for an $1GHz$ clock frequency) if clock arrives also $400ps$ earlier for group of n mesh buffers the clock mesh is going to be driven exclusively by the group of n mesh buffers during $400ps$. If the current provided by the group of n mesh buffers is in the same order as the leakage current and if during the $400ps$ the leakage current added to the current provided by the group of n mesh buffers is able to charge/discharge the mesh buffer past $vdd/2$ then variations on the leakage current could, occasionally, produce a clock glitch.

Consider a square clock mesh with m lines and m columns designed over a square area of x^2. Consider that k flip-flops are connected to the clock mesh and therefore the flip-flop density, d, is given by $d = \frac{k}{x^2}$. Assuming that the clock sinks are connected to the clock mesh by stubs the total capacitance represented by the clock mesh is given by the sum of the mesh wiring capacitance plus the sum of the stubs capacitance plus the sum of clock sink capacitances. The total capacitance can then be expressed as a function of the mesh size m, chip area x and flip-flop density d as shown in equation 5.7.

$$Cap_{total} = x \times 2m \times Wcap + x^3 \times \frac{1}{2m} \times Wcap \times d + x^2 \times d \times FFcap \tag{5.7}$$

In equation 5.7 the first term represents the capacitance of the mesh lines. The second term represents the total stub capacitance. It is assumed that the average stub length is $\frac{x}{2m}$. The final term represents the total clock sink capacitance. If we consider the flip-flop clock pin capacitance, $FFcap$, to be $1.5fF$ and the mesh wire capacitance, $Wcap$, equal to $0.15nF/m$ for a design with an area of $10mm \times 10mm$ with $100K$ flip-flops the total mesh capacitance is $1.51nF$. We know that for nanometric technologies such as $90nm$ or $65nm$ technologies the maximum gate leakage is about $100nA$ (BUTZEN, 2007). Considering that a typical flip-flop, as shown in figure 5.26, the clock pin drives 3 PMOS and 3 NMOS transistors, since when gate leakage is maximum for PMOS transistors it is negligibly for NMOS, in the worst case the gate leakage will be dominated by the three NMOS transistors. The total gate leakage current drained by the clock sinks can be at most $100\mu A$. Since the maximum high impedance time considered in this work is 40% of the clock period the mesh capacitance will be discharged during at most $400ps$ by gate leakage currents. It means that during a clock cycle $40fC$ will be discharged from the clock mesh, assuming that the supply voltage is $1V$ the total load in the clock mesh will be of $1.51nC$.

It can be seen that the leakage current is too small to affect, during the high impedance time, the charge stored in the clock mesh. Therefore under the situation described earlier

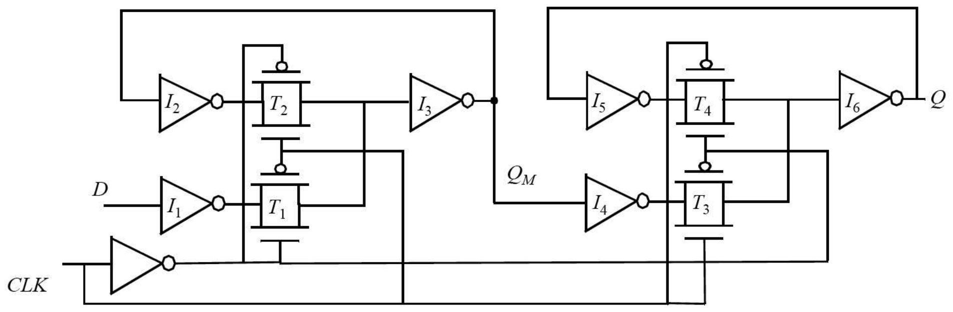


Figure 5.26: Master-slave positive edge-triggered register, using multiplexers. (RABAEY, 1996)

it is not reasonable to assume that the clock mesh can reach the voltage of $Vdd/2$ without a current much larger than the gate leakage current to be applied to the clock mesh.

5.4.8 Conclusions

This work addressed the problem of reducing the short circuit power in mesh-based clock architectures. Its contribution can be summarized as follows.

- We have shown that adding a high impedance time for the clock signal can significantly reduce short-circuit power and hence the total power consumption and improve skew and slew.
- A novel mesh buffer design was proposed to implement the high impedance time. It was shown that our design can reduce short-circuit power and total power up to 59% and skew by 22%, with minimal penalty on slew (depending on the input skew applied).

Although our buffers have more transistors than traditional buffers, they have a 50% smaller input capacitance (for a minimum channel length mesh buffer) due to multiple internal stages. This, in turn, will reduce the buffer sizes in the driving clock tree. The proposed mesh buffer can be further improved. Our sizing scheme is quite naïve. If all the internal stages were sized with the FO4 rule, the mesh buffer input capacitance could be further reduced.

Using increased channel lengths to insert delays in the proposed mesh buffer has produced a significant slew penalty. New techniques to introduce delays in the proposed mesh buffer have to be researched to reduce the slew penalty. Although the slew penalty is large in a zero skew case it is not easily degraded as the input skew increases, which presents an important advantage since in practice mesh input skew is expected to be about 20% of the clock period.

6 CONCLUSIONS

Chapter 3 has shown through a bibliographic review and an architectural study that mesh-based clock distribution architectures are the best solution to design low skew and variability tolerant clock networks. The bibliographic study has shown that clock meshes are used in most microprocessor designs while an experimental study has shown that clock meshes deliver a clock skew about 10$\times$ smaller than a pure tree architecture

The lack of solutions to analyze large clock meshes has motivated the chapter 4 of this thesis. Clock mesh analysis is a very complex task due to the high number of elements required to accurately model a clock mesh and to the accuracy required in the delay estimation. A SWS was proposed to enable the electrical simulation of large clock meshes and speed it up through parallel execution. Later other techniques were proposed but the SWS stands out for its simplicity and accuracy. The SWS is the only method that allows the mesh to be simulated in any electrical simulator and therefore can be used with minimal implementation effort by anyone. Unlike other methods the SWS is scalable and therefore can handle any mesh size. The SWS was shown to be 1% accurate while related works do not estimated the error imposed by model simplification required by their methods.

In chapter 5 a new strategy was proposed to optimize clock meshes. Differently from all the previous optimization techniques for clock meshes the techniques presented in this chapter propose to optimize the clock mesh by reducing the short circuit currents between the different clock mesh buffers when a skewed signal is applied to it. Two approaches were proposed, a new clock mesh buffer design and a sizing algorithm.

The new buffer design was shown to improve total mesh power and skew by switching to a high impedance state in between a rise/fall transition. The total mesh power improvement was 59% while the clock skew improved by 22% in relation to the traditional inverter design. The higher the clock skew observed in the inputs of a clock mesh the more advantageous is the new mesh buffer.

A post processing sizing algorithm for mesh buffers was presented in section 5.3. It also relies on the principle of reducing short circuit currents to improve clock mesh power and timing. Short circuit currents are reduced by reducing the sizing of mesh buffers that cause the higher short circuit currents. This technique has shown an average improvement of 20% in total clock mesh power consumption, 33% reduction in clock skew and a penalty of 26% in clock slew. Mesh buffer sizing can be applied to almost any design with minimal effort.

Altough mesh buffer sizing has proven to reduce clock skew and power, it presents two major drawbacks, the inclusion of dummy capacitances when buffers are undersized and effect of the implemented heuristics that is responsible for the slew penalty observed. The development of a formal mesh buffer sizing algorithm is a natural sequence of the work of section 5.3.

Another important thing to be continued is the evaluation of the new buffer design. The benefits of the mesh buffer proposed in section 5.4 should be evaluated considering its impact on the complete clock distribution scheme. The smaller the clock mesh input capacitance is and the higher its skew reduction is, fewer are the resources required in the global clock network to meet the power and timing constraints.

REFERENCES

AGARWAL, A.; BLAAUW, D.; ZOLOTOV, V. Statistical Clock Skew Analysis Considering Intra-Die Process Variations. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2003, San Jose, CA. **Proceedings...** New York: ACM, 2003. p.914–921.

ALIMADADI, M. et al. Energy Recovery from High-Frequency Clocks Using DC-DC Converters. In: IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI, VLSI, 2008, Montpellier, France. **Proceedings...** Los Alamitos: IEEE Computer Society, 2008. p.162–167.

ANDERSON, F. E.; WELLS, J. S.; BERTA, E. Z. The Core Clock System on the Next Generation Itanium Microprocessor. In: IEEE INTERNAIONAL SOLID STATE CIRCUITS CONFERENCE, ISSCC, 2002. **Proceedings...** [S.l.: s.n.], 2002. p.146–147.

BAILEY, D. W.; BENSCHNEIDER, B. J. Clocking Design and Analysis for a 600-MHz Alpha Microprocessor. **IEEE Journal of Solid-State Circuits**, [S.l.], v.33, n.11, p.1627–1633, Nov. 1998.

BERKELAAR, M. Statistical Delay Calculation, a Linear Time Method. In: TAU, 1997. **Proceedings...** [S.l.: s.n.], 1997. p.15–24.

BINDAL, N. et al. Scalable Sub-10ps Skew Global Clock Distribution for a 90nm Multi-GHz IA Microprocessor. In: IEEE INTERNAIONAL SOLID STATE CIRCUITS CONFERENCE, ISSCC, 2003. **Proceedings...** [S.l.: s.n.], 2003. p.346–349.

BOESE, K. D.; KAHNG, A. B. Zero-Skew Clock Routing Trees With Minimum Wirelength. In: ANNUAL IEEE INTERNATIONAL ASIC CONFERENCE AND EXHIBIT, 5., 1992. **Proceedings...** [S.l.: s.n.], 1992. p.17–21.

BUTZEN, P. F. **Leakage Current Modeling in Sub-micrometer CMOS Complex Gates**. 2007. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil.

CHEN, H. et al. A sliding window scheme for accurate clock mesh analysis. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.939–946.

CLABES, J. et al. Design and Implementation of the POWER5 Microprocessor. In: DESIGN AUTOMATION CONFERENCE, DAC, 41., 2004. **Proceedings...** [S.l.: s.n.], 2004. p.670–672.

CONG, J. et al. Bounded-skew clock and Steiner routing. **ACM Trans. Des. Autom. Electron. Syst.**, New York, NY, USA, v.3, n.3, p.341–388, 1998.

FRIEDMAN, E. G. Clock Distribution in Synchronous Digital Integrated Circuits. **Proceeding of IEEE**, [S.l.], v.89, n.5, p.665–692, May 2001.

FRIEDRICH, J. et al. Design of the Power6 Microprocessor. In: IEEE INTERNAIONAL SOLID STATE CIRCUITS CONFERENCE, ISSCC, 2007. **Proceedings...** [S.l.: s.n.], 2007. p.96–97.

GEANNOPOULOS, G.; DAI, X. An Adaptative Digital Deskewing Circuit for Clock Distribution Networks. In: IEEE INTERNAIONAL SOLID STATE CIRCUITS CONFERENCE, ISSCC, 1998. **Proceedings...** [S.l.: s.n.], 1998. p.400–401.

GOLUB, G. H.; LOAN, C. F. V. **Matrix Computations**. 2nd ed. Baltimore, MD: Johns Hopkins University Press, 1989.

GRONOWSKI, P. et al. High-performance microprocessor design. **IEEE Journal of Solid-State Circuits**, [S.l.], v.35, n.5, p.676–686, May 1998.

GUNTZEL, J. L. **Functional Timing Analysis of VLSI Circuits Containing Complex Gates**. 2000. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil.

GUSTAVSEN, B.; SEMLYEN, A. Rational approximation of frequency domain responses by vector fitting. **IEEE Transactions on Power Delivery**, [S.l.], v.14, n.3, p.1052–1061, July 1999.

HANAFI, H. I. et al. Design and characterization of a CMOS off-chip driver/receiver withreduced power-supply disturbance. **IEEE Journal of Solid-State Circuits**, [S.l.], v.27, n.5, p.783–791, May 1992.

HART, J. M. et al. Implementation of a fourth-generation 1.8-GHz dual-core SPARC V9 microprocessor. **IEEE Journal of Solid-State Circuits**, [S.l.], v.41, n.1, p.210–217, Jan. 2006.

HINTON, G. et al. A 0.18-μm CMOS IA-32 Processor With a 4-GHz Integer Execution Unit. **IEEE Journal of Solid-State Circuits**, [S.l.], v.36, n.11, p.1617–1627, November 2001.

HITCHCOCK, R. B. Timing verification and the timing analysis program. In: ANNUAL ACM IEEE DESIGN AUTOMATION CONFERENCE, 1988. **Papers on Twenty-five Years of Electronic Design Automation**. New York: ACM, 1988. p.446–456.

IGARASHI, M. et al. A low-power design method using multiple supply voltages. In: INTERNATIONAL SYMPOSIUM ON LOW POWER ELECTRONICS AND DESIGN, ISLPED, 1997, Monterey, California. **Proceedings...** New York: ACM Press, 1997. p.36–41.

JACKSON, M. A. B.; SRINIVASAN, A.; KUH, E. S. Clock routing for high-performance ICs. In: ACM/IEEE CONFERENCE ON DESIGN AUTOMATION, DAC, 27., 1990. **Proceedings...** New York: ACM Press, 1990. p.573–579.

JAIN, A. et al. A 1.2GHz Alpha Microprocessor with 44.8GB/s Chip Pin Bandwidth. In: IEEE INTERNAIONAL SOLID STATE CIRCUITS CONFERENCE, ISSCC, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.240–241.

KALLA, R.; SINHAROY, B.; TENDLER, J. M. IBM Power5 chip: a dual-core multi-threaded processor. **IEEE Micro**, [S.l.], v.24, n.2, p.40–47, Jan. 2004.

KANG, S.-M.; LEBLEBICI, Y. **CMOS Digital Integrated Circuits**: Analysis and Design. New York: McGraw-Hill, 1996.

KURD, N. A. et al. A Multigigahertz Clocking Scheme for the Pentium 4 Microprocessor. **IEEE Journal of Solid-State Circuits**, [S.l.], v.36, n.11, p.1647–1653, November 2001.

KURD, N. A. et al. Multi-GHz Clocking Scheme for Intel Pentium 4 Microprocessor. In: IEEE INTERNAIONAL SOLID STATE CIRCUITS CONFERENCE, ISSCC, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.404–405.

LEEDS, J.; UGRON, G. Simplified Multiple Parameter Sensitivity Calculation and Continuously Equivalent Networks. **IEEE Transactions on Circuits and Systems**, [S.l.], v.14, n.2, p.188–191, June 1967.

LI, P.; ACAR, E. A Waveform Independent Gate Model for Accurate Timing Analysis. In: INTERNATIONAL CONFERENCE ON COMPUTER DESIGN: VLSI IN COMPUTERS & PROCESSORS, ICCD, 2005, San Jose, CA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2005. p.363–365.

MAHONEY, P. et al. Clock distribution on a dual-core, multi-threaded Itanium-family processor. In: IEEE INTERNAIONAL SOLID STATE CIRCUITS CONFERENCE, ISSCC, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.6–10.

MCNAIRY, C.; BHATIA, R. Montecito: a dual-core, dual-thread Itanium processor. **IEEE Micro**, [S.l.], v.25, n.2, p.10–20, March-April 2005.

MEHROTRA, V.; BONING, D. Technology Scaling Impact of Variation on Clock Skew and Interconnect Delay. In: IEEE INTERNATIONAL INTERCONNECT TECHNOLOGY CONFERENCE, 2001, Burlington, CA. **Proceedings...** Piscataway: IEEE, 2001. p.122–124.

PANGJUN, J.; SAPATNEKAR, S. S. Low-Power Clock Distribution Using Multiple Voltages and Reduced Swings. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, [S.l.], v.10, n.3, p.309–318, June 2002.

PHAM, D. C. et al. Overview of the architecture, circuit design, and physical implementation of a first-generation cell processor. **IEEE Journal of Solid-State Circuits**, [S.l.], v.41, n.1, p.179–196, 2006.

PHAM, D. et al. The design and implementation of a first-generation CELL processor. In: SOLID-STATE CIRCUITS CONFERENCE, ISSCC, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.184–592.

PHAM, D. et al. Key features of the design methodology enabling a multi-core SoC implementation of a first-generation CELL processor. In: CONFERENCE ON ASIA SOUTH PACIFIC DESIGN AUTOMATION, ASP-DAC, 2006, Yokohama, Japan. **Proceedings...** New York: ACM Press, 2006. p.871–878.

RABAEY, J. M. **Digital integrated circuits**: a design perspective. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

RAJARAM, A.; HU, J.; MAHAPATRA, R. Reducing Clock Skew Variability via Cross Links. In: DESIGN AUTOMATION CONFERENCE, DAC, 41., 2004. **Proceedings...** [S.l.: s.n.], 2004. p.18–23.

RAJARAM, A.; PAN, D. Z. MeshWorks: an efficient framework for planning, synthesis and optimization of clock mesh networks. In: CONFERENCE ON ASIA AND SOUTH PACIFIC DESIGN AUTOMATION, ASP-DAC, 2008, Seoul, Korea. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 2008. p.250–257.

REDDY, S. M.; WILKE, G. R.; MURGAI, R. Analyzing timing uncertainty in mesh-based clock architectures. In: CONFERENCE ON DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 9., 2006, Munich, Germany. **Proceedings...** New York: ACM Sigda, 2006. p.1097–1102.

RESTLE, P. et al. A clock distribution network for microprocessors. **IEEE Journal of Solid-State Circuits**, [S.l.], v.36, n.5, p.792–799, May 2001.

RESTLE, P. et al. The clock distribution of the Power4 microprocessor. In: IEEE INTERNAIONAL SOLID STATE CIRCUITS CONFERENCE, ISSCC, 2002. **Proceedings...** [S.l.: s.n.], 2002. p.144–145.

SALEH, R. et al. Clock skew verification in the presence of IR-drop in the power distribution network. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.19, n.6, p.635–644, June 2000.

SENTHINATHAN, R. et al. A 650-MHz, IA-32 Microprocessor with Enhanced Data Streaming for Graphic and Video. **IEEE Journal of Solid-State Circuits**, [S.l.], v.34, n.11, p.1454–1465, November 1999.

SU, H.; SAPATNEKAR, S. S. Hybrid structured clock network construction. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2001, San Jose, California. **Proceedings...** Piscataway: IEEE Press, 2001. p.333–336.

SUTHERLAND, I. E.; SPROULL, R. F. Logical effort: designing for speed on the back of an envelope. In: UNIVERSITY OF CALIFORNIA/SANTA CRUZ CONFERENCE ON ADVANCED RESEARCH IN VLSI, 1991. **Proceedings...** Cambridge: MIT Press, 1991. p.1–16.

TAM, S. et al. Clock generation and distribution for the first IA-64 microprocessor. **IEEE Journal of Solid-State Circuits**, [S.l.], v.35, n.11, p.1545–1552, Nov. 2000.

TAM, S.; LIMAYE, R. D.; DESAI, U. N. Clock Generation and Distribution for the 130-nm Itanium 2 Processor with 6-MB On-Die L3 Cache. **IEEE Journal of Solid-State Circuits**, [S.l.], v.39, n.4, p.636–624, Apr. 2004.

TANG, K.; FRIEDMAN, E. Lumped versus distributed RC and RLC interconnect impedances. In: IEEE MIDWEST SYMPOSIUM ON CIRCUITS AND SYSTEMS, 43., 2000, Lansing, USA. **Proceedings...** Los Alamitos: IEEE, 2000. p.136–139.

THOMSON, M. G. R.; RESTLE, P. J.; JAMES, N. K. A 5GHz Duty-Cycle Correcting Clock Distribution Network for the POWER6 Microprocessor. In: IEEE INTERNAIONAL SOLID STATE CIRCUITS CONFERENCE, ISSCC, 2006. **Proceedings. . .** [S.l.: s.n.], 2006. p.1522–1529.

VENKATARAMAN, G. et al. Combinatorial algorithms for fast clock mesh optimization. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2006, San Jose, California. **Proceedings. . .** New York: ACM Press, 2006. p.563–567.

VISWESWARIAH, C. et al. First-order incremental block-based statistical timing analysis. In: CONFERENCE ON DESIGN AUTOMATION, DAC, 41., 2004, San Diego, CA. **Proceedings. . .** New York: ACM, 2004. p.331–336.

WANG, R.; KOH, C.-K. A frequency-domain technique for statistical timing analysis of clock meshes. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2007, San jose, CA. **Proceedings. . .** Piscataway: IEEE Press, 2007. p.334–339.

WARNOCK, J. D. et al. The circuit and physical design of the POWER4 microprocessor. **IBM Journal of Research and Development**, [S.l.], v.46, n.1, p.27–52, Jan. 2002.

WESTE, N. H. E.; ESHRAGHIAN, K. **Principles of CMOS VLSI design**: a systems perspective. Boston, MA, USA: Addison-Wesley Longman Publishing, 1985.

WILKE, G. **Evaluating Clock Distribution Architectures**. 2004. Trabalho de Diplomação (Curso de Engenharia de Computação) — , Instituto de Informática, UFRGS, Porto Alegre.

WILKE, G. R.; MURGAI, R. Design and Analysis of "Tree+Local Meshes" Clock Architecture. In: INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN, ISQED, 8., 2007, San Jose, CA. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 2007. p.165–170.

XANTHOPOULOS, T. et al. The Design and Analysis of the Clock Distribution Network for a 1.2GHz Alpha Microprocessor. In: IEEE INTERNAIONAL SOLID STATE CIRCUITS CONFERENCE, ISSCC, 2001. **Proceedings. . .** [S.l.: s.n.], 2001. p.402–403.

YE, X. et al. Analysis of large clock meshes via harmonic-weighted model order reduction and port sliding. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2007, San Jose, CA. **Proceedings. . .** Piscataway: IEEE Press, 2007. p.627–631.

YE, X. et al. Accelerating Clock Mesh Simulation Using Matrix-Level Macromodels and Dynamic Time Step Rounding. In: INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN, IEEE ISQED, 9., 2008, San Jose, CA. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 2008. p.627–632.

YEH, C. et al. Clock Distribution Architectures: A Comparative Study. In: INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN, ISQED, 7., 2006, San Jose, CA. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 2006. p.85–91.

ZHANG, H.; RABAEY, J. Low-swing interconnect interface circuits. In: INTERNATIONAL SYMPOSIUM ON LOW POWER ELECTRONICS AND DESIGN, ISLPED, 1998, Monterey, CA. **Proceedings...** New York: ACM Press, 1998. p.161–166.

ZHANG, L. et al. Clock Skew Analysis via Vector Fitting in Frequency Domain. In: INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN, ISQED, 9., 2008, San Jose, CA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2008. p.476–479.

APÊNDICE A ANALISE E OTIMIZAÇÃO DE ARQUITETURAS DE RELÓGIO DO TIPO MALHA

A.1 Introdução

O relógio é o mais importante sinal em qualquer circuito síncrono. O sinal de relógio controla o momento em que os dados são armazenados nos elementos seqüenciais. Se a temporização do sinal de relógio não for extremamente precisa os valores armazenados pelo elementos seqüenciais podem apresentar erros. Portanto, o período do sinal de relógio deve ser definido de forma que sempre haja tempo suficiente para que os dados sejam corretamente computados pela lógica combinacional.

O período de relógio é limitado pelo relação de atrasos demonstrada na equação A.1. A equação deve ser interpretada da seguinte forma. Dois flip-flops, A e B são separados por uma lógica combinacional de atraso T_C e o sinal de relógio chega ao flip-flop A no ciclo n com um atraso de $T_{CK}(n)'$ e no flip-flop B no ciclo $n+1$ com o atraso de $T_{CK}(n+1)''$. Sendo TS_{FFB} o tempo de setup para o flip-flop B, o menor valor que o sinal de relógio pode assumir para garantir o correto funcionamento do circuito considerando-se os atrasos do caminho em questão é T_{CLOCK}.

$$T_{CLOCK} \geq TP_{FFA} + T_C + TS_{FFB} + (T_{CK}(n)' - T_{CK}(n+1)'') \tag{A.1}$$

Circuitos mais rápidos exigem períodos de relógio menores. O escalamento da tecnologia faz com que os atrasos relacionados com os blocos combinacionais e com os elementos de memória sejam reduzidos, no entanto, a diferença entre os atrasos da rede de relógio para os diferentes elementos de memória não é reduzida na mesma proporção. O aumento do efeito da variabilidade de fabricação, a maior sensibilidade a variações ambientais e a maior complexidade de projeto tornam as redes de relógio menos síncronas.

Circuitos de alto desempenho exigem que técnicas especiais sejam utilizadas no projeto de rede de distribuição de relógio tornando-os menos suscetíveis às fontes de variabilidade e mais sincronizados. Para o projeto dessas redes normalmente são utilizados *clock meshes*. A estrutura dos *clock meshes* reduz variações nos atrasos da rede de relógio devido aos caminhos redundantes inerentes a estrutura dos mesmos.

Esse trabalho estuda formas de analisar e otimizar estruturas de relógio baseadas em *clock meshes*. Um método para viabilizar a simulação elétrica de *clock meshes* é proposto e comparado com outros métodos estado da arte. Duas estratégias de otimização também são propostas e comparadas com trabalhos relacionados.

A.1.1 Definições

- Atraso do relógio: O atraso da rede de relógio é estimado em relação a cada um dos terminais da mesma. O atraso da rede de relógio para um dado terminal é calculado como o tempo necessário para um pulso de relógio percorrer a rede de relógio desde a fonte até o terminal em questão.

- *Skew* do relógio: O *skew* da rede de relógio é calculado como a maior diferença entre o atraso da rede de relógio para quaisquer dois terminais conectados por um caminho puramente combinacional.

- *Slew* do relógio: O *slew* da rede de relógio é calculado como o maior tempo de subida ou descida do sinal de relógio em qualquer um dos terminais da rede.

- *Jitter* do relógio: O *jitter* do sinal de relógio representa a variação do atraso da rede de relógio para um dado terminal em função do tempo. Quando o atraso da rede de relógio é modelado como uma distribuição Gaussiana o jitter é calculado como $3\times\sigma$. O *jitter* total da rede de relógio é determinado pelo maior *jitter* observado para cada terminal.

A.2 Estratégias para Distribuição de Relógio

Diversas técnicas para distribuir o sinal de relógio com alta confiabilidade, baixo consumo de potência e baixo *skew* foram desenvolvidas. Essas técnicas foram dividas em quatro grupos: as que focam no aumento da confiabilidade da rede de relógio, as que reduzem o consumo de potência, estratégias de roteamento para a rede de relógio e as que são aplicadas no nível arquitetural.

A.2.1 Confiabilidade

São duas as técnicas mais utilizadas para aumentar a confiabilidade da rede de relógio: o isolamento elétrico e o assinalamento diferencial.

O isolamento elétrico consiste em ocupar as trilhas adjacentes à trilha de relógio com um fio conectado a terra. A figura A.1 demonstra como é feito o isolamento elétrico. Conectando as trilhas adjacentes à terra, a linha de relógio se torna menos vulnerável a ruído de *crosstalk*. É possível observar que essa estratégia não protege a rede de relógio da influência das trilhas das camadas superior e inferior, no entanto essa influência é reduzida devido a diferente orientação do roteamento adotada nessas camadas.

Outra maneira de proteger a rede de relógio de fontes de ruído é através do uso do assinalamento diferencial. Esta técnica consiste em codificar a informação na diferença de tensão de um par diferencial, desta forma, qualquer fonte de ruído que afete o par diferencial igualmente não é capaz de gerar erros. A figura A.2 demonstra o funcionamento desta técnica.

A.2.2 Baixo Consumo

Reduzir o consumo de potência de redes de distribuição de relógio tem se tornado um grande desafio para os projetistas. São duas também as técnicas utilizadas para reduzir o consumo de potência de redes de distribuição de relógio: *clock gating* e assinalamento reduzido.

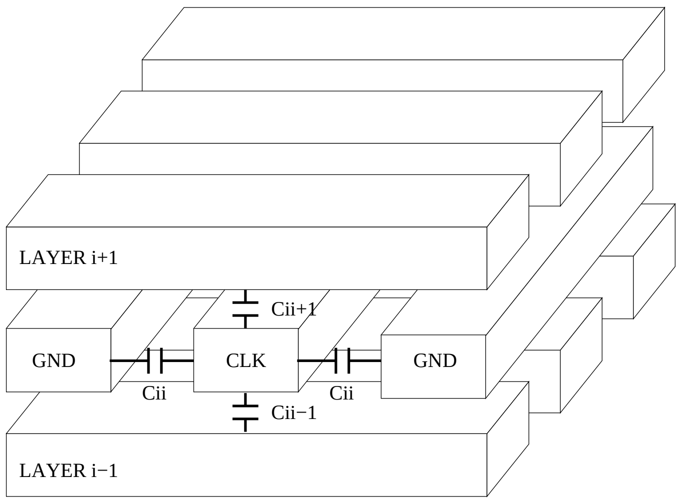


Figura A.1: Gerenciamento das camadas de roteamento.

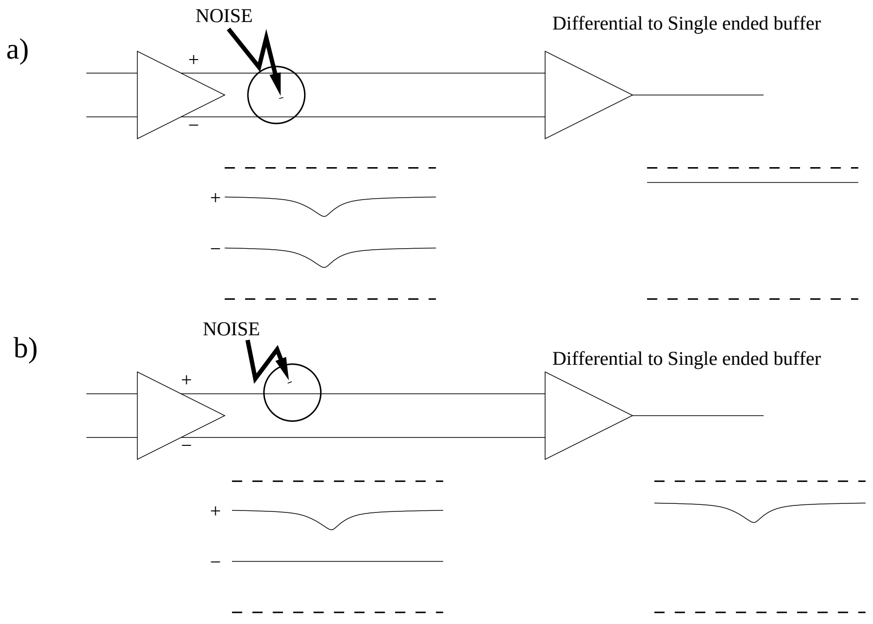

a)
NOISE
Differential to Single ended buffer
+
−
+
−
b)
NOISE
Differential to Single ended buffer
+
−
+
−

Figura A.2: Assinalamento diferencial.

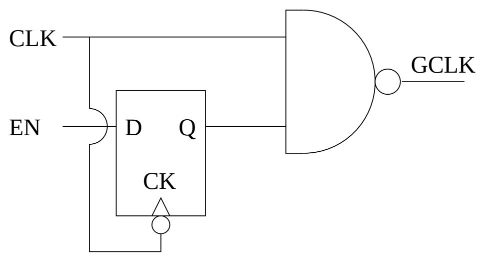


Figura A.3: Célula *clock gater*.

Clock gating é uma técnica que consiste em congelar o sinal de relógio para blocos que não est ao sendo utilizados. O congelamento do sinal de relógio reduz o consumo de potência dinâmico na árvore de relógio e nos terminais de relógio. Como o relógio é fixado em um dado valor lógico o consumo dos blocos combinacionais é igualmente reduzido uma vez que as estradas desses blocos são mantidas fixas enquanto não houver transição no sinal de relógio. A figura A.3 ilustra uma célula utilizada para congelar o sinal de relógio quando o sinal EN é colocado em 0.

A técnica do assinalamento reduzido consiste em reduzir o consumo de potência reduzindo o valor da tensão de alimentação ou o valor da excursão de sinal para a rede de relógio. Caso o assinalamento reduzido seja implementado através da redução da tensão de alimentação (Vdd), a redução no consumo de potência dinâmica será quadrática em relação a redução de Vdd, caso apenas a excursão do sinal de relógio seja reduzida a redução no consumo de potência será linear em relação a redução da excursão do sinal. A equação A.2 demonstra a relação entre o consumo de potência dinâmica, as tens de alimentação e as tensões de assinalamento. A tensão de assinalamento V_S é uma fração de Vdd e, portanto, quando a tensão Vdd é reduzida, a redução no consumo de potência é quadrática. Cabe salientar que esta técnica aumenta a sensibilidade da rede de relógio.

$$P = f \times C_L \times Vdd \times Vs \qquad \text{(A.2)}$$

A.2.3 Topologias de Roteamento

O sinal de relógio deve ser distribuído de uma fonte única para centena de milhares de terminais. Para que o sinal de relógio chegue aproximadamente ao mesmo tempo a todos os terminais é necessário que o roteamento da rede de relógio iguale o comprimento de fio entre a fonte de relógio e cada um dos terminais. Para tanto diferentes estratégias são utilizadas.

ASICs em geral utilizam redes de distribuição baseadas em árvores. Uma árvore de relógio pode ser uma estrutura regular como uma árvore-H ou então gerada a partir de algum algoritmo especifico. Árvores-H possuem a propriedade de terem o mesmo comprimento de fio da fonte até qualquer um dos terminais. A figura A.4 ilustra uma árvore-H.

Diversos algoritmos para a geração de árvores de relógio com baixo *skew* já foram desenvolvidos. O primeiro trabalho a tratar desse problema é o *Method of Mean and Medians (MMM)* (JACKSON; SRINIVASAN; KUH, 1990). Esse trabalho procura criar uma árvore com aproximadamente o mesmo comprimento de fio entre a fonte de relógio e os terminais. Ao contrário de uma árvore-H o o método *MMM* é capaz de rotear o sinal de relógio até os terminais independente de como os terminais se espalham na área do chip.

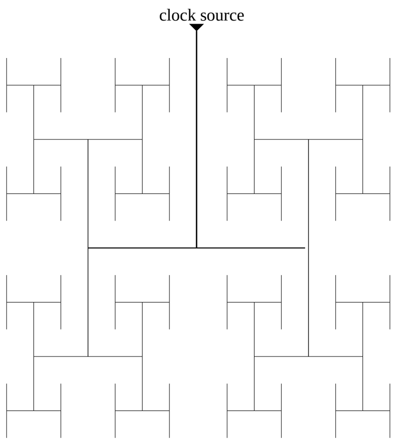


Figura A.4: Exemplo de árvore-H.

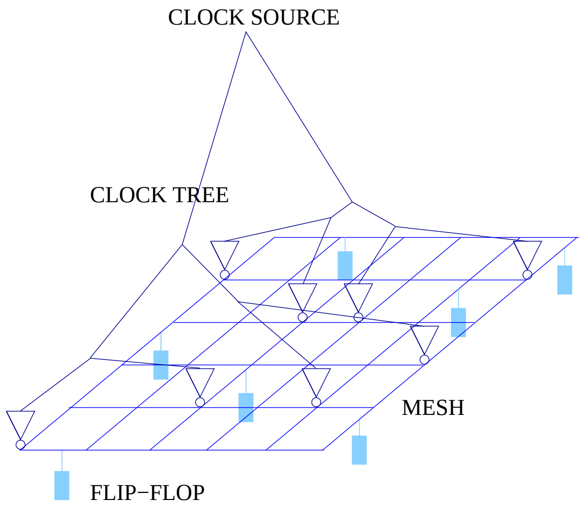


Figura A.5: Exemplo de um *clock mesh*.

O algoritmo *Deffered-Merge Embeding (DME)* (BOESE; KAHNG, 1992) foi proposto como uma evolução do *MMM*. O algoritmo *DME* é capaz de garantir que a árvore de relógio gerado possua comprimento de fio mínimo e *skew* igual a zero.

Circuitos que operam a freqüências maiores, como microprocessadores, necessitam de estruturas de relógio que sejam capazes de distribuir o sinal de relógio com um baixo *skew* independente de variabilidade ambiental e de processo. Nesse contexto, normalmente, são utilizados *clock meshes* ou outras estruturas baseadas em *clock meshes* como *clock spines* ou *crosslinks*. Os *clock meshes* consistem em uma malha de fios que é conectada nos terminais de uma árvore de relógio. A malha de fios cria caminhos redundantes entre os *buffers* do ultimo estágio da árvore de relógio de forma que os caminhos mais rápidos da árvore são compensados pelos caminhos mais lentos. A figura A.5 ilustra um *clock mesh* conectado a uma árvore de relógio. A usabilidade dos *clock meshes* é limitada pelo maior consumo de potência e maior utilização dos recursos de roteamento.

A.2.4 Arquiteturais

Um bom projeto da rede de distribuição de relógio no nível arquitetural também é importante para que a mesma apresente um baixo consumo de potência e um baixo *skew*. A rede de relógio deve ser dividida em diferentes domínios, dentro de um mesmo domínio

Tabela A.1: Características dos circuitos de teste

Circuito	#Portas	#FFs	Área (mm^2)	Área dos FFs área (mm^2)
$D1$	536,5K	16,75K	5×10	0,8×6,67
$D2$	1016,6K	39,16K	5×10	2,23×9,62
$D3$	7659,6K	287,39K	16×16	12,03×14,63

devem ficar terminais entre os quais existe uma intensa comunicação. Além disso os terminais pertencentes a um mesmo domínio devem pertencer a um mesmo bloco funcional de forma que, quando o bloco funcional relacionado a um domínio não esteja realizando nenhuma computação, o sinal de relógio para aquele domínio possa ser interrompido.

Dentro de um mesmo domínio a sincronização do sinal de relógio é maior do que entre diferentes domínios. Para aumentar a sincronização do relógio entre os diferentes domínios se utilizam *buffers* com atraso variável de forma que o atraso desses *buffers* seja calibrado para que o *skew* entre os diferentes domínios seja o menor possível. Esse método é chamado de *deskew* (TAM et al., 2000), (KURD et al., 2001) e (TAM; LIMAYE; DESAI, 2004).

A.3 Arquiteturas de Distribuição de Relógio

A.3.1 Uma comparação entre *clock meshes* e árvores de relógio

É sabido que *clock meshes* apresentam um menor *skew* de relógio do que arquiteturas do tipo árvore as custas de um maior consumo de potência. Para avaliar o quanto o *skew* apresentado por um *clock mesh* é menor do que o *skew* de uma árvore de relógio realizamos um conjunto de simulações elétricas.

Três circuitos de teste foram utilizados, os dados relacionados a cada circuito são apresentados na tabela A.1. Os circuito $D1$ e $D2$ foram gerados artificialmente enquanto $D3$ é um circuito industrial real. Todos os três circuitos foram modelados em uma tecnologia de $11\mu m$. A tensão de alimentação foi estabelecida em 1,2V e a temperatura em 55^oC.

A posição de cada terminal da rede de relógio foi extraída dos circuitos de teste. A rede de distribuição de relógio foi gerada utilizando a informação do posicionamento dos terminais da rede de relógio e assumindo que não existem obstruções de posicionamento para os *buffers* da rede de relógio e de roteamento para os fios. O modelo elétrico da rede de distribuição de relógio é gerado e utilizado para o cálculo do *skew* de relógio de cada distribuição.

A tabela A.2 mostra os valores do *skew*, *slew* e consumo de potência para duas redes de distribuição de relógio, uma que utiliza *clock meshes* e outra que utiliza apenas uma árvore-H. É possível observar que os valores de latência e *slew* do relógio são muito semelhantes para ambas arquiteturas. No entanto, o *skew* de relógio apresentado pelo *clock mesh* é aproximadamente 10× menor do que o *skew* apresentado pela arquitetura que utiliza apenas uma árvore-H. Entretanto o consumo de potência dinâmico, calculado como $C \times Vdd^2 \times f$, é maior para o *clock mesh* devido a maior utilização de recursos de roteamento que se refletem numa maior capacitância.

Esse estudo permite observar que a utilização de um *clock mesh* em uma rede de distribuição de relógio é capaz de reduzir o *skew* em 10× em troca de um pequeno aumento

Tabela A.2: *Clock mesh* vs. árvore de relógio

chip	mesh/ árvore-H size	max skew (ps)		max atraso (ps)		max slew (ps)		Capacitância (pF)		potência (W)	
		mesh	arv.	mesh	arv.	mesh	arv.	mesh	arv.	mesh	arv.
$D1$	16×16	3,3	29,1	480,6	499,1	89,5	89,5	255,5	231,0	0,15	0,13
	64×64	0,1	1,1	636,9	638,4	89,6	89,6	563,2	396,3	0,32	0,23
	128×128	0,1	1,3	717,5	718,9	89,6	89,6	1030,2	603,7	0,59	0,35
$D2$	64×64	0,5	4,2	674,2	678,2	105,7	105,7	1008,1	721,4	0,58	0,41
	128×128	0,2	1,4	754,3	756,1	105,1	105,1	1694,5	1048,4	0,97	0,60
$D3$	64×64	4,8	29,3	975,1	995,0	110,9	108,6	5570,1	5084,0	3,2	2,9
	128×128	0,9	5,5	1050,8	1055,0	110,2	75,2	5885,2	4892,2	3,4	2,8

no consumo de potência. Devido a essa característica os *clock meshes* são amplamente utilizados nos projetos de arquiteturas de relógio para circuitos que necessitam ter um *skew* de relógio muito baixo.

A.3.2 Distribuição de relógio para microprocessadores

As técnicas utilizadas nos diferentes microprocessadores comerciais variam muito de acordo com as necessidades de cada projeto. No entanto, certos aspectos se mantém constantes e, portanto, podem ser entendidos como tendências relacionadas ao projeto da rede de relógio. Esse pontos são:

- A distribuição de relógio global é feita por uma árvore de relógio global.
- Os terminais da rede de relógio são divididos em diferentes domínios de acordo com a funcionalidade do bloco ao qual cada pertence.
- Algum mecanismo de *deskew* é inserido para reduzir o *skew* entre os diferentes domínios de relógio.
- Um *clock mesh* é inserido em cada domínio para reduzir o efeito da variabilidade.
- Uma árvore de relógio local (com *buffers*) que levam o sinal de relógio do *clock mesh* até os terminais da rede.

A figura A.6 ilustra essa arquitetura de distribuição de relógio genérica. Uma árvore de relógio é usada para distribuir o sinal de relógio da fonte para cada um dos diferentes domínios. A distancia percorrida por essa árvore é grande e portanto o efeito de variações ambientas é significativo nessa árvore. A árvore global é normalmente isolada eletricamente para reduzir o efeito de *crosstalk* nos atrasos e na confiabilidade do sinal de relógio.

Cada diferente domínio de relógio representa um terminal da árvore global. Algum mecanismo de *deskew* pode estar presente entre os domínios. Dentro de cada domínio de relógio um *clock mesh* é utilizado para reduzir o *skew* do relógio.

Os terminais da rede se conectam ao *clock mesh* de seu respectivo domínio através de árvores de distribuição de relógio locais. As árvores locais possuem *buffers* e portanto reduzem a capacitância que é carregada pelo *clock mesh*. Como as árvores locais cobrem

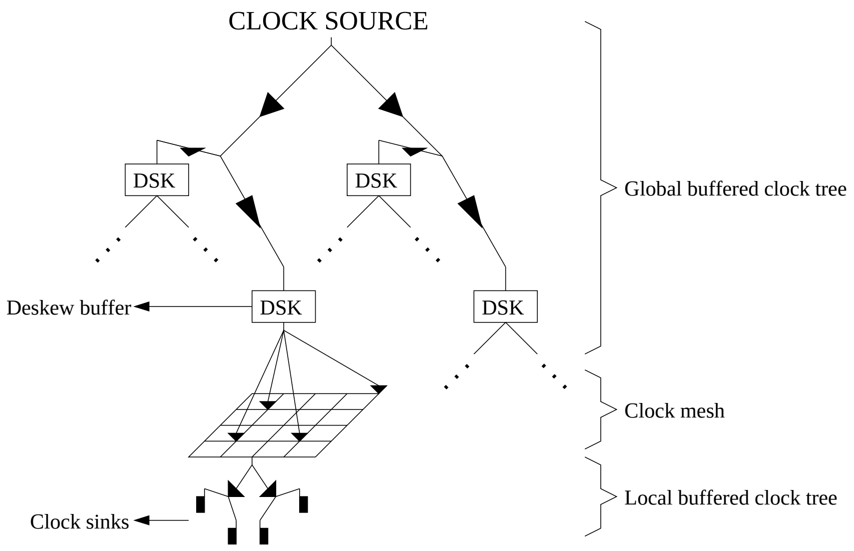


Figura A.6: Distribuição de relógio genérica para microprocessadores

distâncias reduzidas o efeito das fontes de variabilidade nelas é pequeno e não aumentam significativamente o *skew* da rede de relógio.

É importante salientar o papel fundamental do *clock mesh* no projeto das redes de distribuição de relógio para microprocessadores. O *clock mesh* é responsável por filtrar o efeito das fontes de variação ambiental, reduzindo o *skew* e o *jitter* do sistema.

A.4 Métodos para Analise de Malhas de Relógio

Clock meshes são redes difíceis de analisar. O grande numero de ciclos inerentes a estrutura do mesh somado as características não lineares dos inversores torna a analise desta estrutura extremamente complexa. Ao mesmo tempo a analise dos atrasos da rede de relógio deve ser precisa uma vez que o *skew* é medido como a diferença entre os atrasos da rede.

Esta seção apresenta o primeiro método proposto para tratar este problema (CHEN et al., 2005). A proposta consiste em simplificar o modelo elétrico do *clock mesh* dividindo a simulação completa do *clock mesh* em simulações menores. Outros métodos foram propostos após esse trabalho, sendo que esses atingem uma maior aceleração na caracterização do *clock mesh* as custas de um maior erro na estimativa (YE et al., 2008) (WANG; KOH, 2007) (ZHANG et al., 2008) (YE et al., 2007).

A.4.1 O método da janela deslizante

O método da janela deslizante (ou *Sliding Window Scheme (SWS)*) consiste na aplicação da heurística dividir-para-conquistar através de simplificações no modelo elétrico do *clock mesh*. Ao invés de realizar apenas uma única simulação elétrica utilizando um modelo elétrico preciso para todo o *clock mesh* a metodologia *SWS* propõe que sejam realizadas n simulações aonde em cada simulação o *clock mesh* é modelado com precisão apenas para uma pequena região, reduzindo o tempo de execução e os requisitos de memória da simulação elétrica.

Apenas dentro da região aonde o *clock mesh* é modelado com precisão os atrasos são medidos. Sucessivas simulações são feitas de forma que o atraso para cada flip-flop seja medido pelo menos uma única vez. A figura A.7 ilustra o esquema proposto. O método

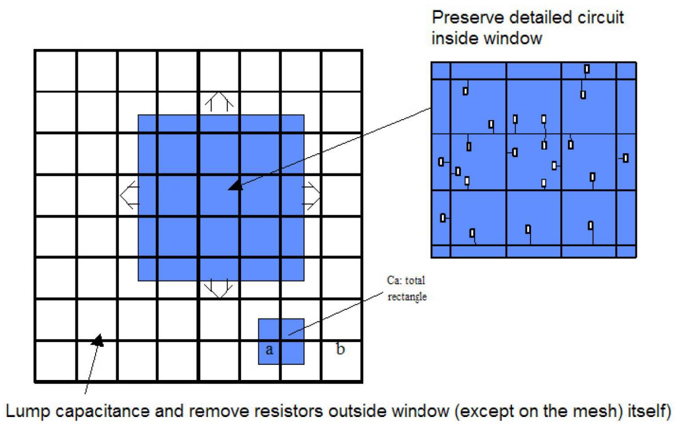

Preserve detailed circuit inside window
Ca: total rectangle
a
b
Lump capacitance and remove resistors outside window (except on the mesh) itself)

Figura A.7: *Sliding Window Scheme* (CHEN et al., 2005).

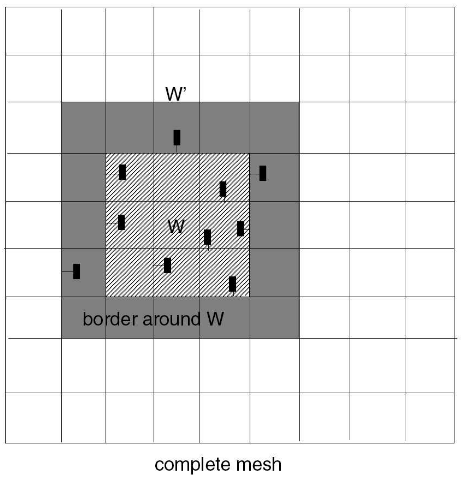


Figura A.8: Exemplo de uma janela com borda para aumentar precisão.

da janela deslizante se basead no efeito da resistência dos fios que faz com que apenas os *buffers* na redondeza de um determinado terminal da rede de relógio afetem o atraso do sinal de relógio até aquele terminal. Desta forma, apenas esses necessitam ser modelados com precisão para que o atraso seja estimado corretamente.

O método da janela deslizando oferece uma alta precisão nas medidas realizadas mais próximas ao centro da região que é modelada com precisão, no entanto os pontos próximos a fronteira entre a região precisa e a região simplificada podem apresentar um erro grande nas medidas. Para resolver este problema se propões a criação de uma borda de precisão ao redor da região aonde os atrasos estão sendo medidos. Dentro dessa borda os atrasos dos terminais da rede de relógio não são medidos, no entanto o *clock mesh* é modelado com precisão. A figura A.8 ilustra como a borda é incluída no modelo a ser simulado.

O uso da borda é fundamental para que o método apresente a precisão necessária para que o *skew* do relógio seja estimado corretamente. No entanto a inclusão da borda aumenta o numero de nodos presentes no modelo elétrico do mesh reduzido e portanto aumenta o tempo de execução da simulação elétrica. A figura A.9 mostra o erro máximo observado nas medidas para um conjunto de *benchmarks* utilizando-se a metodologia da janela deslizando com e sem a borda.

Cada uma das medidas de erro realizadas consiste na aplicação do método da janela

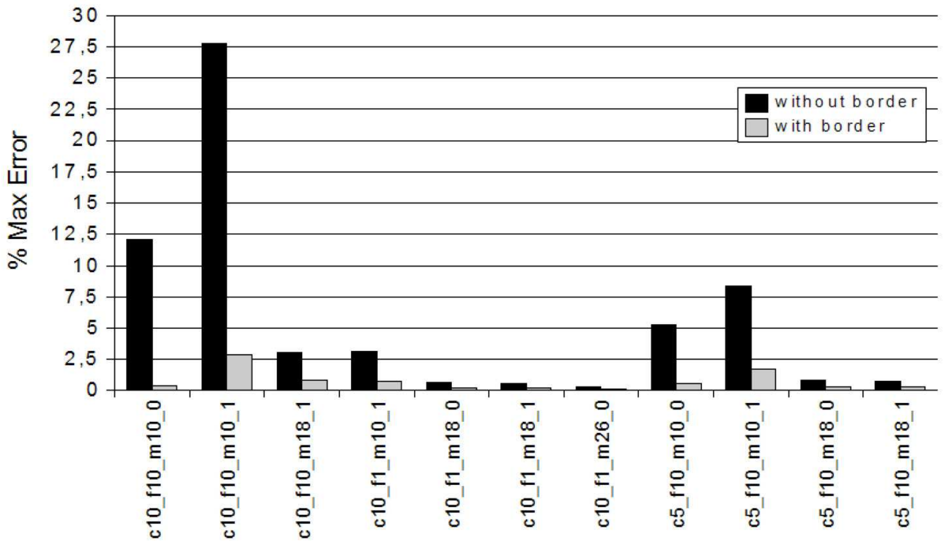


Figura A.9: Precisão do método da janela deslizante.

deslizante para um circuito sintético com características diferentes. Os nomes de cada circuito podem ser vistos no eixo X, sendo que o nome *c10_f1_m18_0* denota um circuito gerado numa área de $10mm \times 10mm$ contento 1000 terminais de relógio, projeto com um mesh com 18 linhas e 18 colunas aonde os *mesh buffers* são inseridos sempre que um fio horizontal do *mesh* cruza um fio vertical. A terminação *_1* indica que os *mesh buffers* são inseridos em um cruzamento sim e outro não. Pela analise do gráfico da figura A.9 podemos perceber que com a utilização da borda o erro máximo obtido na medida dos atrasos nos terminais da árvore de relógio é menor que 2.5%. Se a borda não for utilizada o erro na medida do atraso chega a 27.5%.

O tempo total de simulação em uma única maquina é maior quando o método da janela deslizante é utilizado do que quando uma simulação única é feita. No entanto esse método permite a paralelização das simulações e reduz o uso máximo de memória, permitindo que *clock meshes* maiores possam ser simulados.

A.5 Estratégias para Otimização de Malhas de Relógio

Recentemente o tópico da otimização de *clock meshes* passou a ser abordado po alguns autores como VENKATARAMAN et al. (2006)RAJARAM; PAN (2008). Ambos trabalhos propõe soluções para reduzir o *skew* de relógio considerando-se as seguintes suposições:

- O sinal de relógio chega ao mesmo tempo a todos os *mesh buffers*.
- O clock mesh apresenta uma alta densidade de linhas e colunas de maneira que existem regiões (definidas pelo quadrilátero formado por duas linhas e duas colunas adjacentes) dentro das quais não existem terminais.
- O conjunto de *mesh buffers* com diferentes tamanhos disponíveis é limitado.

Na prática essas suposições não correspondem a realidade da maioria dos circuitos. O *skew* observado na entrada do *clock mesh* não é zero, a densidade de linhas e colunas do *clock mesh* é menor do que a densidade de terminais e o conjunto de *mesh buffers* é muito grande ou ilimitado.

A suposição de que o *skew* de relógio aplicado no *clock mesh* é zero não é realista pois, caso não houvesse *skew* não haveria necessidade de se utilizar o clock mesh para reduzir o *skew*. Entretanto, é importante observar que o *skew* de relógio na entrada do *clock mesh* é conseqüência das características da árvore de relógio conectada ao *clock mesh* e portanto qualquer método que otimize o *clock mesh* com base no *skew* de entrada só pode ser aplicado depois que a árvore de relógio conectada ao *clock mesh* já foi projetada.

A.5.1 Dimensionamento dos *buffers* do *clock mesh*

A capacidade do *clock mesh* de reduzir o *skew* de relógio é resultado das correntes de curto-circuito entre os *mesh buffers* que transicionam antes e os que transicionam depois. A existência das correntes de curto-circuito é possível graças aos caminhos reconvergentes que conectam entre si a saída dos *mesh buffers*. Embora as correntes de curto-circuito entre os *mesh buffers* sejam necessárias para que os caminhos rápidos compensem os caminhos mais lentos elas também são responsáveis por um maior consumo de potência e por aumentar o *slew* de relógio.

Para reduzir as correntes de curto-circuito dentro de um *clock mesh* se propõe que o tamanho dos *buffers* que tem maior participação na corrente de curto-circuito sejam reduzidos. É sabido que quanto mais distante da média é o tempo de chegada do sinal de relógio na entrada de um *mesh buffer* maior é a corrente de curto circuito causada pelo *buffer*. Portanto, dois aspectos devem ser obedecidos durante o dimensionamento dos *mesh buffers*:

- O tamanho dos *mesh buffers* só pode ser reduzido. Caso o tamanho de algum mesh buffers fosse aumentado seria necessário redimensionar os *buffers* da árvore de relógio, o que tornaria os tempos de chegada do sinal de relógio nos *mesh buffers* inválidos.

- Quando o tamanho de um *mesh buffer* é reduzido é necessário incluir uma capacitância extra para que a carga percebida pela árvore de relógio não seja alterada.

Obedecendo ambas restrições descritas acima dois algoritmos de dimensionamento de *mesh buffers* para redução da corrente de curto-circuito foram propostos.

A.5.1.1 Dimensionamento baseado na média

O dimensionamento baseado na média utiliza o tempo de chegada médio do sinal de relógio em cada *mesh buffer* para calcular o quanto o tamanho de cada mesh buffer será reduzido. O tamanho da redução no tamanho de um dado *mesh buffer* é calculado proporcionalmente a diferença entre o tempo de chegada médio do sinal de relógio e o tempo de chegada do sinal de relógio na entrada do *mesh buffer* em questão.

O algoritmo utilizado é ilustrado pela figura 5. O primeiro passo consiste no cálculo do tempo de chegada médio do sinal de relógio nos *mesh buffers*, Atm. A seguir, a máxima diferença entre o tempo de chegada do sinal de relógio em um dado *mesh buffer* e Atm, $Atdist$, é calculada. Para cada *buffer* o fator de redução do tamanho é calculado.

A.5.1.2 Dimensionamento probabilístico

O algoritmo de dimensionamento probabilístico utiliza informações da função de distribuição de probabilidades (*Probability Density Function*) para calcular o fator de redu-

```
Input: $At$,$M$,$MR$
Output: $Mnew$
$Atm = compute\_mean(At)$;
$Atdist = 0$;
foreach Mesh Buffer i do
    $Atdist$ = max($Atdist$,$|At_i - Atm|$);
end
foreach Mesh Buffer i do
    $reduction$ = $1 - MR * (|At_i - Atm|/Atdist)$;
    $Mnew_i$ = $M_i * reduction$;
end
```

Figura A.10: Mean sizing algorithm

ção de tamanho de cada *mesh buffer*. O fator de redução é calculado proporcionalmente a probabilidade do tempo de chegada do sinal de relógio na entrada do *mesh buffer* em questão estar dentro de um intervalo pré definido. Nesse trabalho é assumido que os tempos de chegada do sinal de relógio na entrada dos *mesh buffers* é estocástico e se comporta como uma distribuição Gaussiana.

O primeiro passo do algoritmo é calcular o tempo de chegada médio considerando as médias das distribuições Gaussianas utilizadas para modelar os tempos de chegada nos *mesh buffers* (Atm). A seguir, o desvio padrão médio do conjunto de distribuições gaussianas é calculado. Com base nesses dois valores uma função de densidade acumulada (CDF_{BASE}) de base é construída. Utilizando-se essa função de probabilidade acumulada é calculada a probabilidade de ocorrência de valores dentro de um intervalo de controle descrito como $[Atm - Th, Atm + Th]$, A_{BASE}. Para cada *mesh buffer* i é calculada a probabilidade A_i de o tempo de chegada do sinal de relógio estar entre o intervalo de controle $[Atm_i - Th, Atm_i + th]$. O fator de redução do tamanho do *mesh buffer* é calculado proporcionalmente a diferença entre A_{BASE} e A_i. Caso A_i seja maior que A_{BASE} o tamanho do *mesh buffer* em questão não é alterado.

```
Input: $At$,$M$,$MR$,$Th$
Output: $Mnew$
$Atm = compute\_mean(At)$;
$Ats = compute\_mean\_sigma(At)$;
$CDF_{BASE} = CDF(Atm, Ats)$;
$A_{BASE} = CDF_{BASE}(Atm + Th) - CDF_{BASE}(Atm - Th)$;
foreach Mesh Buffer i do
    $CDF_i = CDF(Atm_i, Ats_i)$;
    $A_i = CDF_i(Atm + Th) - CDF_i(Atm - Th)$;
    $reduction$ = $1 - MR * (A_i/A_{BASE})$;
    if $reduction > 1$ then
        $reduction = 1$;
    end
    $Mnew_i$ = $M_i * reduction$;
end
```

Figura A.11: Probabilistic sizing algorithm

Tabela A.3: Circuitos de teste.

Circuito	Tam. do *Mesh*	Tam. do *chip* (μm)	#FFs	Densidade de FFs
c1	4×4	500×500	250	1000/mm^2
c2	4×4	500×500	500	2000/mm^2
c3	4×4	500×500	750	3000/mm^2
c4	4×4	800×800	600	937.5/mm^2
c5	4×4	800×800	900	1406.25/mm^2
c6	4×4	800×x800	1200	1875/mm^2
c7	8×8	800×00	900	1406.25/mm^2
c8	8×8	800×800	1200	1875/mm^2

A.5.1.3 Experimentos

Ambos algoritmos de dimensionamento foram aplicados a clock meshes projetados para circuitos gerados aleatoriamente. O efeito do dimensionamento produzido pelos algoritmos propostos foi comparado com o uso da regra *Fanout-of-4* (FO4). As informações relativas aos circuitos gerados aleatoriamente estão descritas na tabela A.3.

Para cada circuito três simulações elétricas foram realizadas, uma utilizando apenas o dimensionamento FO4, uma aonde os *mesh buffers* foram redimensionados utilizando o algoritmo baseado no atraso médio (*msize*) e outra utilizando o algoritmo de dimensionamento probabilístico (*psize*). A tabela A.4 compara os valores de consumo de potência, *skew* e *slew* observados na rede de relógio para cada algoritmo. Durante a simulação elétrica, o tempo de chegada do sinal de relógio em cada *mesh buffer* foi modelado como uma distribuição Gaussiana cujo o desvio padrão é equivalente a 7% do período de relógio. O tempo de chegada médio do sinal de relógio é diferente para cada mesh buffer, tendo sido gerado aleatoriamente e apresentando uma variação máxima equivalente a 20% do período de relógio.

Pode-se observar que ambos algoritmos propostos são capazes de reduzir o consumo de potência e o *skew* dos *clock meshes* testados em troca de degradar o *slew* do sinal de relógio. Analisando os dados apresentados na tabela A.4 pode-se fazer algumas outras constatações. Comparando os valores produzidos para os três primeiros circuitos em relação aos últimos 5 pode-se observar que o dimensionamento é mais efetivo para circuitos com uma área menor. Também é possível observar que o ganho obtido com o dimensionamento é maior para circuitos com um numero menor de flip-flops. Ambos comportamentos podem ser explicados pelo modo como as correntes de curto-circuito são afetadas pelas características do *clock mesh*. Sempre que as correntes de curto-circuito são reduzidas a eficiência do redimensionamento é reduzida. Quando a área do circuito é maior o comprimento dos fios entre os *mesh buffers* é maior e, portanto, as correntes de curto-circuito entre os mesmos são menores. Da mesma forma, quando o numero de flip-flops é maior o tempo necessário para carregar as capacitâncias é maior e conseqüentemente o tempo em que a corrente de curto-circuito ira circular no *clock mesh* é menor.

Um segundo experimento foi feito fazendo-se a média entre os resultados obtidos para cada um dos oito circuitos. Os dados obtidos podem ser vistos nas figuras A.12, A.13, A.14 e A.15. Cada barra de cor diferente representa uma configuração diferente dos atrasos do sinal de relógio na entrada do *clock mesh* e cada conjunto de barras mostra os dados para um diferente algoritmo. Três diferentes configuração de atraso para o sinal

Tabela A.4: Melhora obtida com algoritmos de dimensionamento.

Circuito	Método	Potência (mW)		Skew (ps)		Slew (ps)	
		Total	Ganho (%)	Total	Ganho (%)	Total	Ganho (%)
c1	tip	3.09	-	49.51	-	210.33	-
	$msize$	2.74	11.25	23.29	52.95	247.80	-17.82
	$psize$	2.72	11.89	19.83	59.94	271.78	-29.22
c2	tip	4.66	-	75.83	-	212.48	-
	$msize$	4.23	9.24	45.06	31.62	253.80	-19.45
	$psize$	4.20	9.82	43.21	43.02	276.52	-30.14
c3	tip	6.21	-	85.33	-	215.36	-
	$msize$	5.73	7.70	60.91	28.62	257.42	-19.53
	$psize$	5.70	8.22	57.53	32.58	279.28	-29.68
c4	tip	7.62	-	115.77	-	239.55	-
	$msize$	7.16	6.03	92.48	20.12	283.88	-18.51
	$psize$	7.14	6.34	95.37	17.62	302.94	-26.46
c5	tip	10.06	-	139.13	-	255.10	-
	$msize$	9.59	4.62	107.06	23.05	300.68	-17.87
	$psize$	9.57	4.86	107.47	22.76	317.76	-24.56
c6	tip	12.68	-	158.81	-	279.07	-
	$msize$	12.23	3.55	130.96	17.53	325.38	-16.60
	$psize$	12.21	3.73	133.78	15.76	341.13	-22.24
c7	tip	16.20	-	130.75	-	261.49	-
	$msize$	15.00	7.39	94.36	27.83	292.76	-11.96
	$psize$	14.73	9.02	77.54	40.70	320.55	-29.36
c8	tip	18.91	-	152.42	-	274.92	-
	$msize$	17.70	6.37	114.34	24.99	311.08	-13.15
	$psize$	17.43	7.81	103.69	31.97	334.03	-21.50

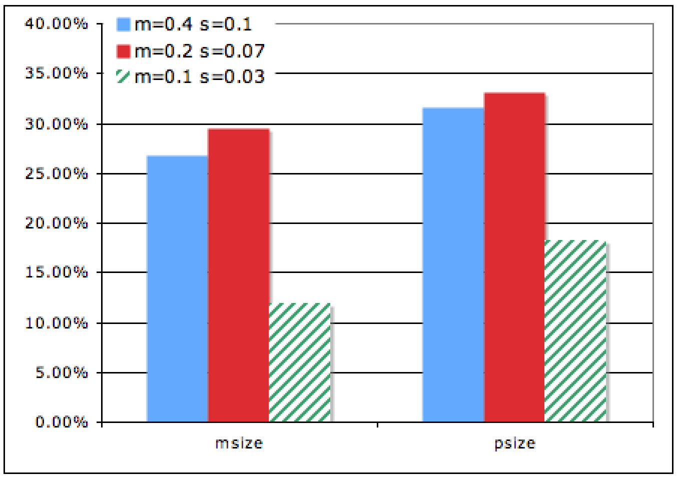


Figura A.12: Melhora média no *skew*.

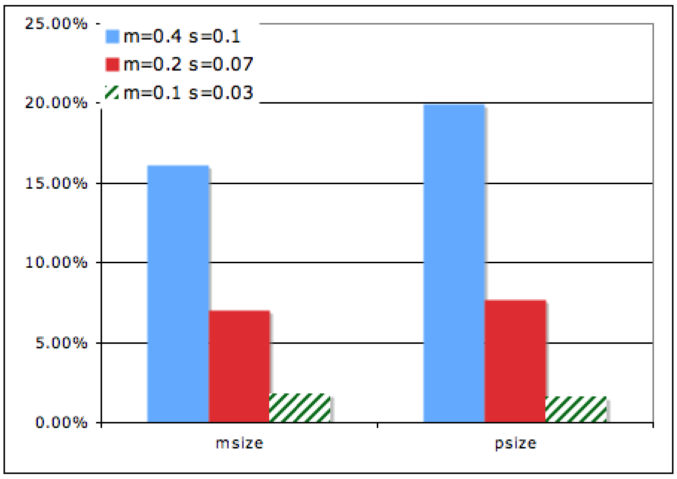


Figura A.13: Melhora média no consumo de potência.

de entrada foram utilizadas, um caso com baixo *skew*, um caso com *skew* típico e um caso com um *skew* alto. Os valores de *skew* que caracterizam cada um desses casos é mostrado na legenda dos gráficos normalizados pelo período de relógio, $1ns$. O *skew* de entrada aplicado no clock mesh possui duas componentes, uma determinística, representada pela diferença nas médias das distribuições Gaussianas utilizadas para representar o tempo de chegada do sinal de relógio em cada *mesh buffer* e uma componente estocástica, representada pelo desvio padrão da distribuição Gaussiana. Nas legendas o valor da componente determinística é representado pela letra m, enquanto a componente estocástica é representada pela letra s.

A figura A.12 mostra que reduções médias de 33% no *skew* do sinal de relógio e 20% no consumo de potência foram atingida às custas de um aumento de 26% no *slew* do sinal de relógio. O aumento do *slew* do sinal de relógio é uma conseqüência direta da limitação de permitir apenas que os *mesh buffers* sejam reduzidos. A capacidade de carga do conjunto de *mesh buffers* foi reduzida devido a diminuição nos tamanhos dos mesmos, aumentando o *slew* do sinal de relógio.

Comparando o efeito do algoritmo de dimensionamento no conjunto de circuitos de teste é possível observar que o algoritmo $psize$ propicia um algoritmo mais agressivo, gerando uma maior redução no tamanho dos buffers. Entretanto, quando a redução pro-

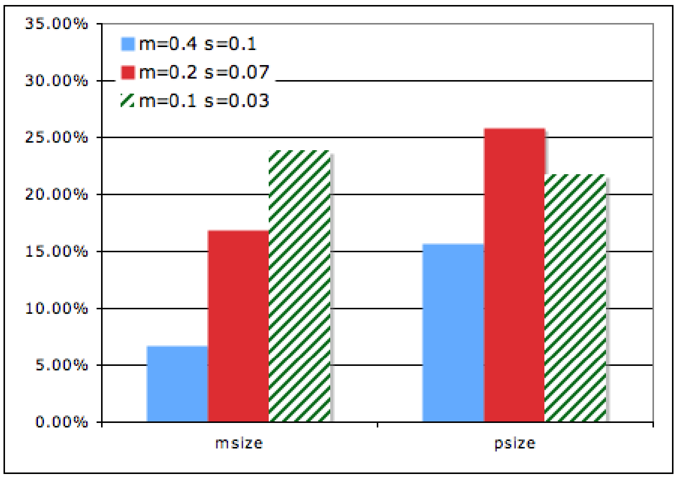


Figura A.14: Degradação média do *slew*.

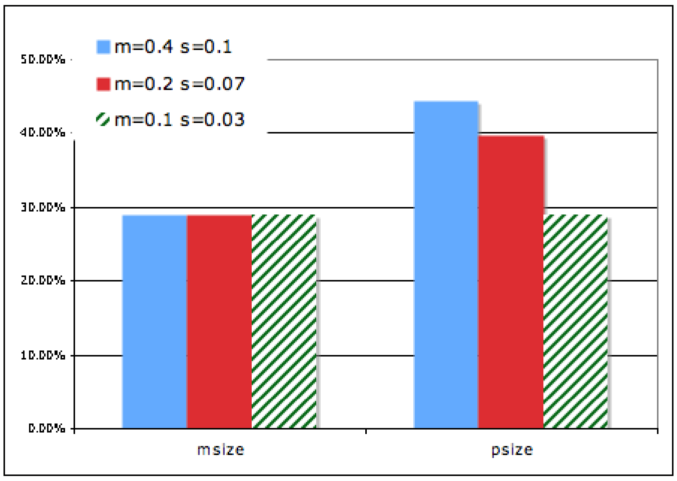


Figura A.15: Redução média no tamanho dos *mesh buffers*.

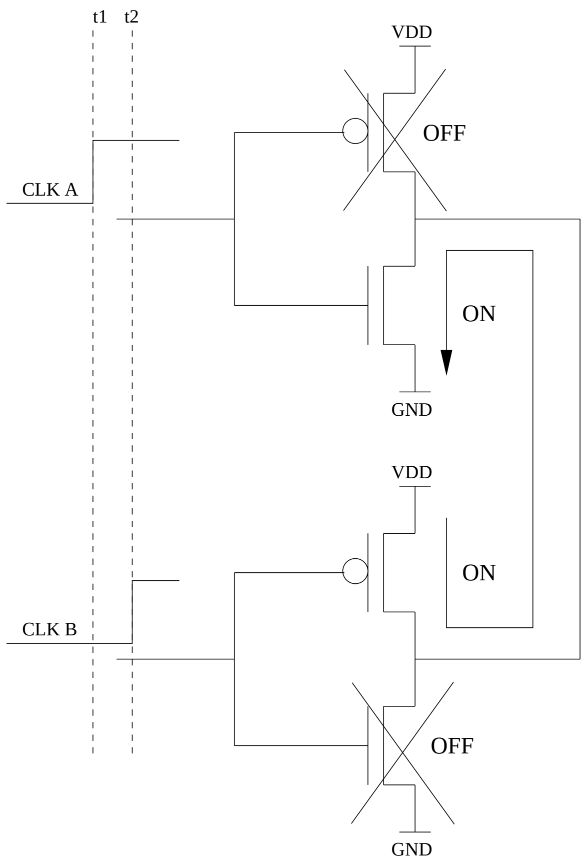


Figura A.16: Exemplo de curto-circuito entre *mesh buffers*.

piciada por ambos algoritmos foi semelhante observa-se que o dimensionamento gerado pelo algoritmo $psize$ é mais eficiente.

É possível concluir com base nos experimentos feitos que o dimensionamento com base nas características temporais do sinal aplicado ao *clock mesh* é capaz de reduzir consumo de potência e *skew* do sinal de relógio. O maior ganho obtido com o algoritmo $psize$ se deve a natureza estocástica do atraso do sinal de relógio. Melhores resultados podem ser obtidos se o tamanho dos *mesh buffers* forem aumentados, no entanto é necessário considerar o efeito dos mesmos nos atrasos da rede que leva o sinal de relógio ao *clock mesh*.

A.5.2 Um novo *mesh buffer*

Correntes de curto-circuito circulam entre os *mesh buffers* sempre que o sinal de relógio não chega exatamente ao mesmo tempo a algum *mesh buffer*, conforme ilustrado pela figura A.16. Esse tipo de corrente de curto circuito pode ser reduzido através da inserção de um tempo durante o qual o mesh buffer tem sua saída mantida em alta impedância.

No caso ilustrado pela figura A.16 quando no mínimo um dos *buffers* esta em alta impedância não existe corrente de curto-circuito fluindo entre eles. No caso de um *clock mesh* esse método é capaz de reduzir as correntes de curto-circuito ou até mesmo elimina-las completamente.

O tempo em que a saída do *mesh buffer* fica em alta impedância é gerado atrasando diferentemente o relógio até os *gates* dos transistores PMOS e NMOS. Considerando-se o *buffer* da figura A.17 o tempo de alta impedância de subida e descida são representados pelas equações A.3 e A.4. Os atrasos $d1_N$ e $d0_N$ representam respectivamente o atraso de

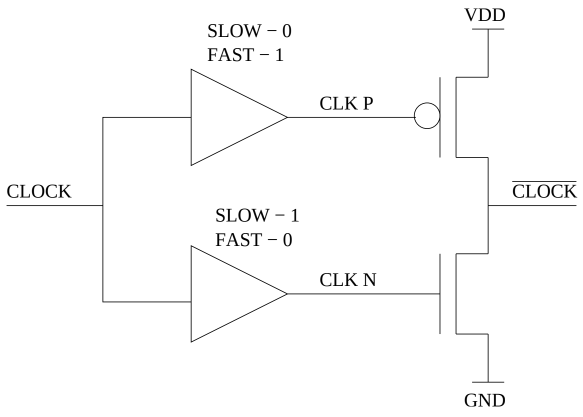


Figura A.17: Um *buffer* de alta impedância inversor.

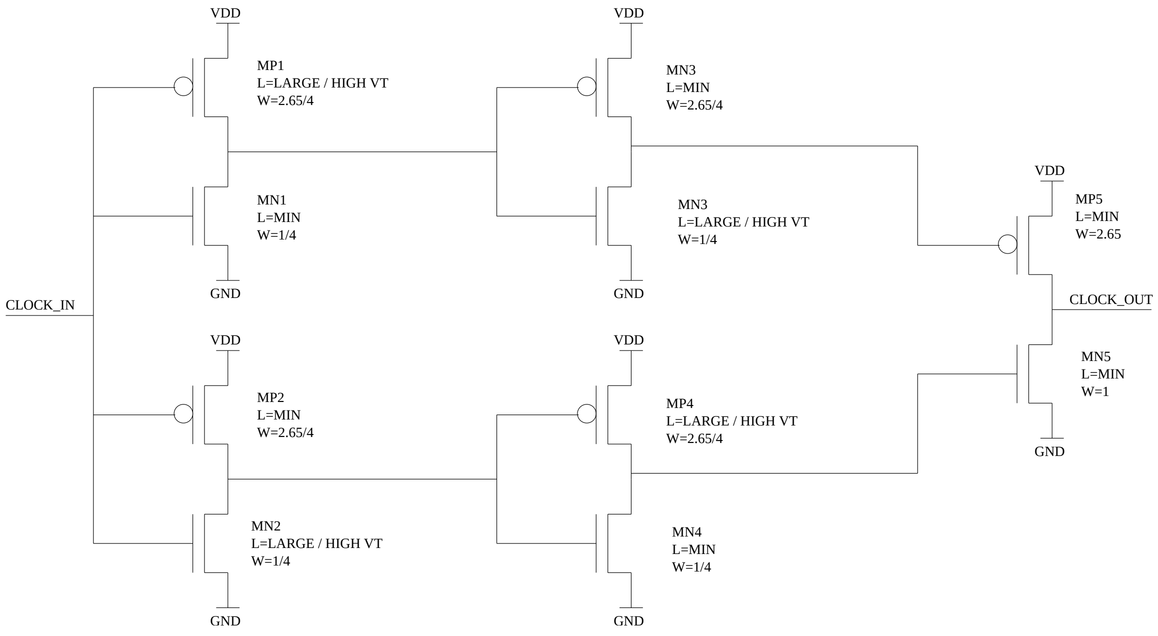


Figura A.18: Esquema elétrico para *buffer tri-state*.

subida e o atraso de descida do sinal de relógio do $CLOCK$ até CLK_N enquanto os atrasos $d1_P$ e $d0_P$ representam os atrasos do nodo $CLOCK$ ao nodo CLK_P.

$$Ztime_{fall} = d1_N - d1_P \tag{A.3}$$

$$Ztime_{rise} = d0_P - d0_N \tag{A.4}$$

Para implementar o tempo de alta impedância é proposto o circuito da figura A.18. Para gerar os atrasos $d0_N$, $d0_P$, $d1_N$ e $d1_P$ são utilizados dois inversores em cascata. Os tempos de subida e descida são aumentados quando necessário aumentando o comprimento do canal dos transistores e utilizando-se transistores com uma voltagem de *threshold* (vt) maior. A figura A.18 também demonstra a relação entre o tamanho dos transistores usados internamente dentro do *buffer tri-state*.

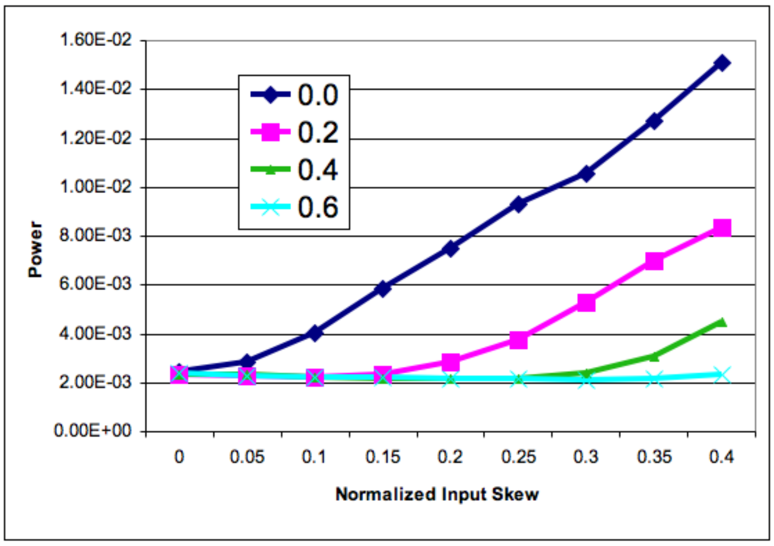


Figura A.19: Potência vs. *Skew* de entrada para inversor simples.

A.5.2.1 Experimentos

Foram realizados dois conjuntos de experimentos para avaliar a eficiência do método proposto para diminuir as correntes de curto circuito. Num primeiro momento o tempo de alta impedância foi gerado artificialmente aplicando-se dois sinais a um inversor simples, um sinal foi aplicado ao *gate* do transistor NMOS enquanto outro foi aplicado ao gate do transistor PMOS.

Nas figuras A.19, A.20 e A.22 são apresentados os ganhos obtidos com a utilização do método proposto para evitar correntes de curto-circuito. O tempo em que o sinal de relógio permanece em alta impedância foi gerado artificialmente. Cada linha dos gráficos das figuras A.19, A.20 e A.22 representa um tempo de alta impedância diferente aplicado ao *clock mesh*. O eixo X indica diferentes valores de *skew* de entrada, normalizados pelo período de relógio, aplicados ao *clock mesh* enquanto o eixo Y demonstra os valores de potência, *skew* e *slew* obtidos em cada caso. Para aumentar a significância dos experimentos, os valores apresentados foram obtidos através da média entre 50 experimentos diferentes sendo que em cada experimento um diferente *clock mesh* foi gerado. Os circuitos gerados automaticamente são formados por 100 flip-flops distribuídos em uma área de 50.000μ^2 e utilizam um *clock mesh* 4$\times$4 para distribuir o relógio para os flip-flops.

Nos resultados demonstrados acima a curva para o tempo de alta impedância igual a 0.0 representa o comportamento de um inversor típico. É possível perceber que o consumo de potência, o *skew* e o *slew* do sinal de relógio foram melhorados com a inserção do tempo de alta impedância. As figura A.19, A.20 e A.21 demonstram respectivamente as melhoras no consumo de potência, *skew* e *slew* do sinal de relógio.

A estratégia proposta é capaz de proporcionar uma grande melhora nas características temporais e no consumo de potência do *clock mesh*. O consumo de potência do *clock mesh* foi reduzido em 84%. O *skew* e *slew* do sinal de relógio foram reduzidos respectivamente em 60% e 45%. O fator de melhora obtido é dependente do *skew* aplicado na entrada do *clock mesh*. Se o sinal de relógio esta completamente sincronizado na entrada do *clock mesh* nenhum ganho é obtido com esta técnica, no entanto nenhuma desvantagem é observada também. Na prática o sinal de relógio apresenta um *skew* na entrada do *clock mesh* entre 10% a 20% do período do sinal de relógio e portanto o método proposto deve apresentar um ganho significativo.

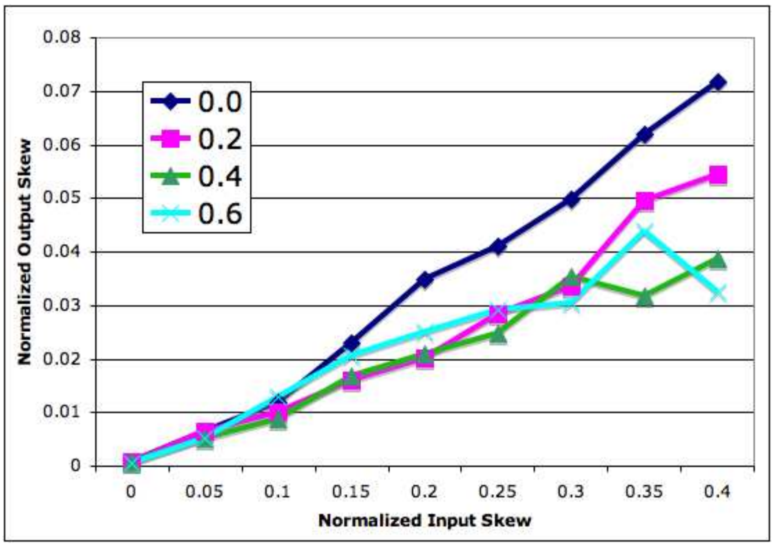


Figura A.20: *Skew* de saída vs. *Skew* de entrada para inversor simples.

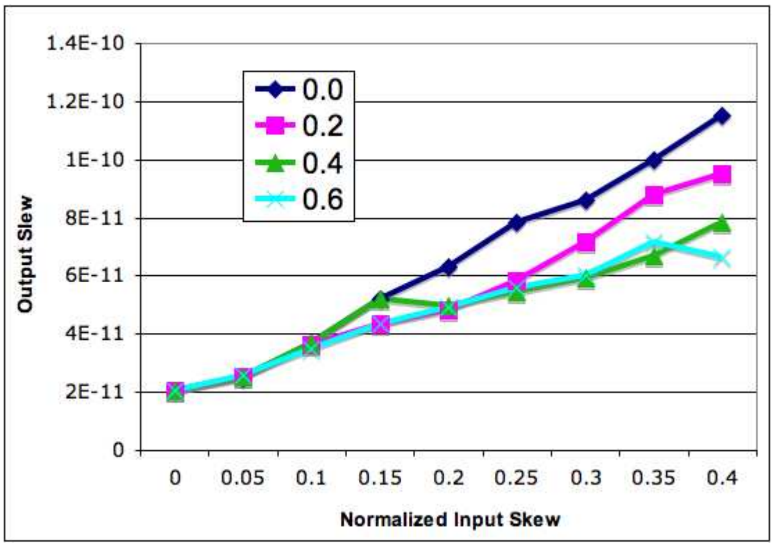


Figura A.21: *Slew* de saída vs. *Skew* de entrada para inversor simples.

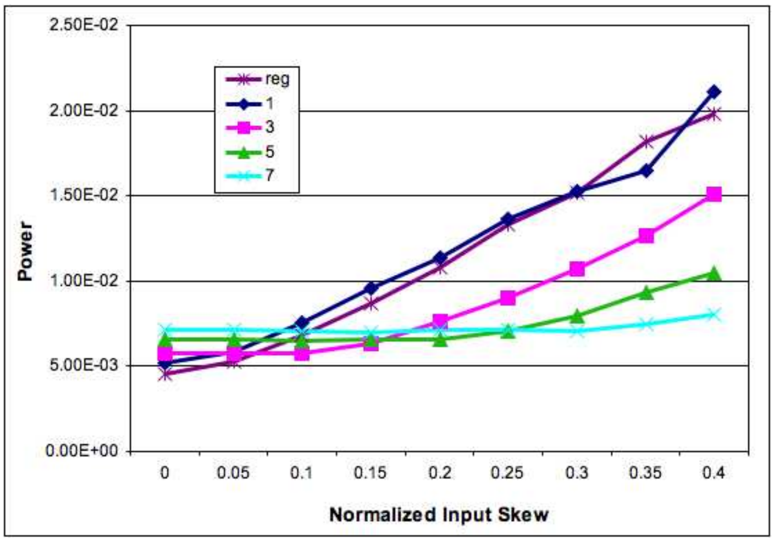


Figura A.22: Potência vs. *Skew* de entrada para inversor de alta impedância.

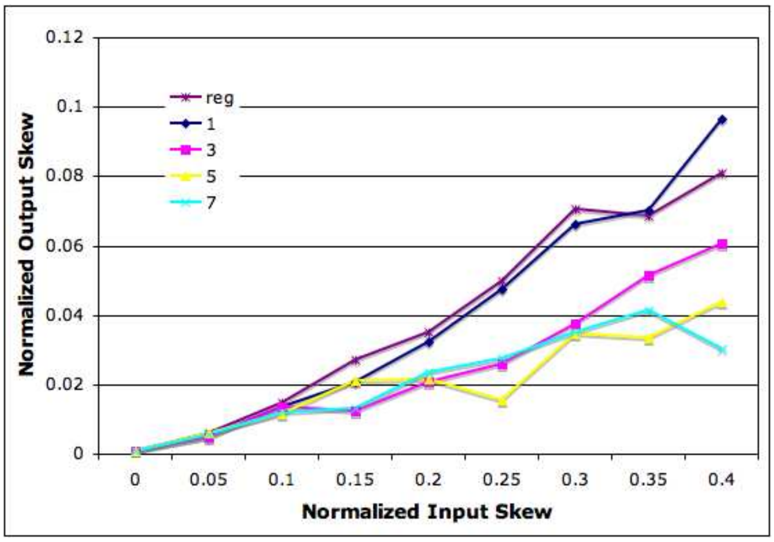


Figura A.23: *Skew* de saída vs. *Skew* de entrada para inversor de alta impedância.

No entanto, a implementação do esquema proposto não é trivial, como ilustrado na figura A.18. A inclusão de mais transistores e o aumento do comprimento de canal de alguns transistores gera uma penalidade no desempenho do *buffer* proposto em comparação com o inversor simples. Para avaliar a penalidade introduzida pelo projeto apresentado na figura A.18 os experimentos anteriores foram repetidos substituindo-se os inversores de duas entradas pelos *mesh buffers* propostos. Para gerar diferentes tempos de alta impedância o inversor proposto foi simulado utilizando diferentes comprimentos de canal nos transistores responsáveis por atrasar o relógio. Cada curva dos gráficos das figuras A.22, A.23 e A.24 representa uma configuração aonde o comprimento do canal dos transistores foi aumentado 1,3,5 ou 7 vezes. O comportamento do inversor típico é denotado pela curva reg.

Percebe-se que, embora tenha ocorrido o aumento no número de transistores e no comprimento do canal de alguns transistores, o *mesh buffer* proposto apresenta melhor desempenho do que um inversor tradicional quando o *skew* de entrada do *clock mesh* é maior que zero. O *overhead* associado ao *mesh buffer* proposto só pode ser percebido

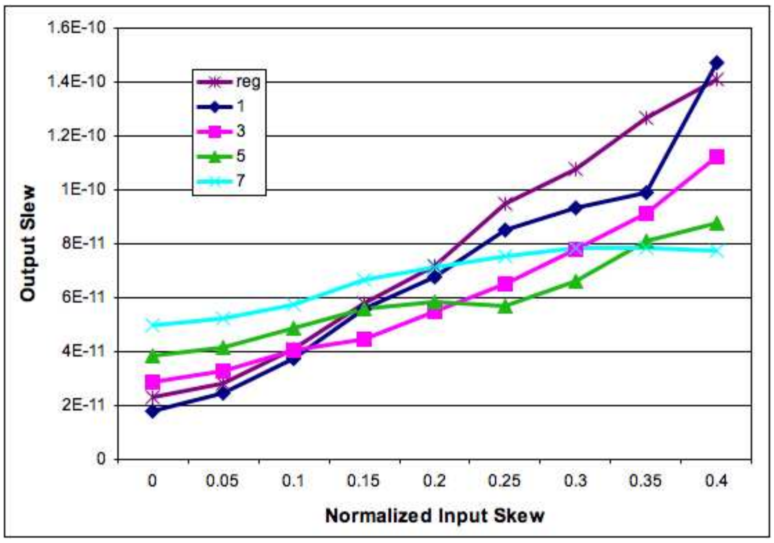


Figura A.24: *Slew* de saída vs. *Skew* de entrada para inversor de alta impedância.

quando o *skew* de entrada aplicado ao *clock mesh* é menor do que 10% do período de relógio.

Observando as figuras A.22, A.23 e A.24 verifica-se que o *mesh buffer* proposto é capaz de reduzir o consumo de potência em 59%, o *skew* do sinal de relógio em 60% e o *slew* em 43%. Cabe salientar que devido ao aumento do comprimento do canal de alguns transistores o *slew* apresentado pelo *mesh buffer* proposto é maior do que o *slew* do inversor tradicional, no entanto percebe-se que quando esse overhead reduzido a medida que o *skew* de entrada aumenta. Quando o comprimento de canal 7 vezes o comprimento de canal mínimo o *overhead* observado no *slew* do sinal de relógio é máximo, entretanto essa configuração também propicia os maiores ganhos quando o *skew* de entrada do *clock mesh* é máximo.

Os resultados obtidos mostram que o *mesh buffer* proposto é capaz de reduzir o consumo de potência, o *skew* e o *slew* do sinal de relógio. No entanto os ganhos obtidos dependem das características temporais do sinal de relógio na entrada do *clock mesh*. Maiores ganhos são observados para maiores valores de *skew* na entrada do *clock mesh*.

A.6 Conclusões

Essa tese demonstrou na seção A.3 que *clock meshes* são a melhor alternativa para lidar com os problemas advindos das fontes de variabilidade que afetam o desempenho dos circuitos integrados.

A falta de metodologias para a analise de *clock meshes* motivou a criação do método da janela deslizante apresentado na seção A.4.1. Foi demonstrado que o método da janela deslizante viabiliza a caracterização de *clock meshes* grandes com um erro menor que 1% em comparação com os valores obtidos durante uma simulação elétrica.

Com base na observação que a aplicação de um sinal de relógio com *skew* diferente de zero produz correntes de curto-circuito entre os diferentes *mesh buffers* dois métodos de otimização para clock meshes foram propostos. O primeiro método apresentado propõe dois algoritmos de dimensionamento para os *mesh buffers* que visam reduzir o tamanho dos *mesh buffers* responsáveis por causar mais curto-circuito. Em média os algoritmos propostos reduziram o consumo de potência em 20% e o *skew* em 33% com um aumento de 26% no *slew* do sinal de relógio. Embora ambos algoritmos de dimensionamento tenham obtido uma boa redução no *skew* do sinal de relógio e no consumo de potência em troca de um aumento no *slew* ambos algoritmos apresentam duas grandes desvantagens, a inserção de capacitâncias *dummy* quando o tamanho dos *mesh buffers* são reduzidos e a limitação de que o tamanho dos *mesh buffers* não pode ser aumentado. Essas limitações são responsáveis pelo aumento no *slew* do sinal de relógio percebido e, portanto, melhores resultados podem ser obtidos se essas limitações forem relaxadas.

O segundo método consiste em um projeto de um novo *mesh buffer* que através da inserção de um tempo de alta impedância evita que as correntes de curto-circuito circulem pelo *clock mesh*. A redução obtida respectivamente no consumo de potência e no *skew* do sinal de relógio foi de 59% e 22% para um caso típico. Neste caso não é observado nenhuma alteração significativa no *slew* do sinal de relógio. A melhora ou degradação do *slew* do sinal de relógio depende do valor do *skew* dos sinal de relógio presente na entrada do *clock mesh*. Para avaliar melhor a contribuição do *mesh buffer* proposto é necessário realizar experimentos simulando-se a estrutura de relógio completa (árvore de relógio mais *clock mesh*).

A eficiência dos métodos de otimização propostos é proporcional ao valor do *skew* do sinal aplicado na entrada do *clock mesh*, quanto maior é o *skew* presente nas entradas do *clock mesh* maior são as vantagens dos métodos propostos.