CROSS-TALK NOISE IMMUNE VLSI DESIGN USING REGULAR LAYOUT FABRICS

by

Sunil P. Khatri

Memorandum No. UCB/ERL M99/70

16 December 1999

CROSS-TALK NOISE IMMUNE VLSI DESIGN USING REGULAR LAYOUT FABRICS

by

Sunil P. Khatri

Memorandum No. UCB/ERL M99/70

16 December 1999

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Cross-talk Noise Immune VLSI Design using Regular Layout Fabrics

by

Sunil P. Khatri

B. Tech. (Indian Institute of Technology, Kanpur, India) 1987
M. S. E. (University of Texas at Austin) 1989

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering-Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Robert K. Brayton, Chair
Professor Alberto Sangiovanni-Vincentelli
Professor Dorit Hochbaum

Fall 1999

The dissertation of Sunil P. Khatri is approved:

R. K. Brayton 12/15/99

Chair Date

12/11/99

Date

David Hodges 12/15/99

Date

University of California at Berkeley

1999

Cross-talk Noise Immune VLSI Design using Regular Layout Fabrics

Sunil P. Khatri

Abstract

Cross-talk Noise Immune VLSI Design using Regular Layout Fabrics

by

Sunil P. Khatri

Doctor of Philosophy in Engineering-Electrical Engineering and Computer Sciences

University of California at Berkeley

Professor Robert K. Brayton, Chair

In modern integrated circuit (IC) fabrication processes, the cross-coupling capacitance between adjacent wires on the same metal layer is a large fraction of the total capacitance of these wires. With shrinking process feature sizes, this fraction increases further. Since these fabrication processes have their minimum feature size well below 1 μm, they are referred to as deep sub-micron (DSM) processes.

As a result of this increase in cross-coupling capacitance, cross-talk between neighboring wires is becoming an increasingly important problem in VLSI design, and is expected to continue to be a problem for future fabrication processes. Two distinct problems occur due to cross-talk. First, cross-talk can cause a significant delay variation in a *victim* wire depending on the electrical state of neighboring wires. The exact delay of a victim wire depends significantly on whether its neighboring wire(s) (*aggressors*) perform a transition as well. If the aggressors perform a transition, then the victim delay depends strongly on whether the aggressors perform a like or an unlike transition. Secondly, cross-talk can cause the logic value of a wire to be incorrectly interpreted depending on the state of neighboring aggressor wires, resulting in a loss of signal integrity. If a victim wire is held at a steady logic value, and aggressors perform a rapid transition, this change in voltage capacitively couples to the victim wire, causing a glitch in its waveform. If the magnitude of this glitch is large enough, the logic value of the victim wire may be incorrectly sampled.

In this work, we propose two VLSI layout methodologies which address the cross-talk problem in DSM integrated circuit design. In our methodology, the cross-talk problem is solved by design, by imposing a fixed pattern of wires on the IC die, on all metal layers. In particular, this repeating pattern, referred to as the Dense Wiring Fabric (DWF) pattern is $\cdots$ *VSGSVSGS* $\cdots$, where V represents a *VDD* wire, G represents a *GND* wire, and S represents a signal wire. This sequence of wires recurs on the metal layers of the IC.

With this choice of layout fabric, the cross-coupling capacitance between signal wires drops by between one and two orders of magnitude, thereby all but eliminating the delay variation and signal integrity problems due to cross-talk. For example, in a 0.1μm process, the delay variation due to cross-talk drops from 2.01:1 to 1.03:1. Also, signal inductances drop by 50% as well, since the current return path for any signal wire is always adjacent to it. The uniformity of inductive and capacitive parasitics which results from the regularity of the DWF is a feature that CAD tools can exploit. Our layout "fabric" scheme eliminates the conventional notion of power and ground routing on the integrated circuit die. Instead, power and ground are essentially "pre-routed" all over the die. By suitably introducing vias whenever VDD (or GND) wires intersect on adjacent metal layers, a power and ground distribution network of low and uniform resistance is created. The DWF also results in tighter tolerances on the inter-layer dielectric thicknesses due to the fact that metal is maximally gridded all over the IC die. Finally, the DWF enables us to easily generate a low-skew global clocking network due to the tightly controlled and uniform parasitics. In our scheme, the characterization of interconnect parasitics (capacitance, inductance and resistance) becomes extremely simple due to the repeating nature of the DWF pattern. This characterization needs to be done only once for a design, resulting in significant time and computational savings.

In this thesis, we introduce two fabrics that utilize the DWF pattern. The first, *Fabric1*, uses the DWF pattern chip-wide, including within the layout cells in the design. This fabric is incorporated into a traditional VLSI design flow, and over a series of examples, exhibits an area overhead of about 60% when two metal layers are used for routing. If more layers are used, then this overhead drops to 17%.

Fabric3 attempts to reduce this area overhead. In this fabric, the logic network is implemented as a network of medium-sized Programmable Logic Arrays (PLAs). We choose an implementation of PLAs which is naturally cross-talk immune, and also extremely dense. The routing area between PLAs utilizes the DWF pattern, while a specialized layout fabric is utilized within the PLAs. We show that the delay of a single PLA is about 50%, and the area about 50% compared to that of a standard-cell based layout. We introduce synthesis algorithms to cluster a logic netlist into a network of PLAs in the Fabric3 style. Each of these PLAs has a bounded width and height. The number of inputs and outputs of each PLA are flexible as long as the resulting PLA width remains within the specified bound. We also perform folding on the resulting PLAs to achieve better logic density. For a series of examples, the area penalty for a network of PLA implementation in the Fabric3 scheme is shown to be a mere 2.4% compared to the standard-cell implementation style, while the circuit delay improves by around 15%. These results remain substantially unchanged regardless of whether 2, 3, 4, 5 or 6 metal layers are utilized for routing the design.

In summary, we show how the uniform and predictably low parasitics in our fabric methodology give rise to a reliable and predictable design style. We have implemented our scheme and compared it with the standard cell design styles. The crosstalk immunity, high speed, low area overhead, quick design turnaround time, and high predictability of our methodology indicate that it is a strong candidate as the preferred design methodology in the DSM era.

R. K. Brayton

Professor Robert K. Brayton
Dissertation Committee Chair

To Papaji and Mummy, and my siblings

Contents

List of Figures

List of Tables

Acknowledgements

My Ph.D. has been a very enjoyable and fruitful experience in great part because of several wonderful people who helped make it happen with their support and their encouragement.

Ever since I came to the United States, I have a tradition of sitting down with myself every Thanksgiving and thinking about all the things in my life that I am grateful about. My family and the strength I derive from it has always been on top of this list. I owe a great deal to the stability and support provided by my family.

My father instilled in me the value of learning and scholarship. He always told us that if there was anything we needed to purchase that would help in our education, that we should never hesitate to ask him for it. I remember with great fondness the many long hours we spent doing high school math together. His patience, calmness and tolerance in all worldly matters will be a source of constant inspiration and even wonder for me. It was hard for us to lose him in 1996, but in spite of her great loss, my mother has continued to be a symbol of strength, tenacity and positivity for all of us. Her simplicity and quiet dignity has fascinated me and helped me cope during hard times. I and my siblings owe a great deal to the great selflessness that my parents cheerfully practiced for our sakes.

My siblings have always been points of light in my life, and their caring and encouragement in good times and bad are most gratefully acknowledged. My brother-in-law Prabodh and my sister Daksha, with their continuous encouragement for me to finish writing soon have helped me finish this thesis in a timely fashion. My brother Arun and sister-in-law Tripat have always been there for me to cheer me on during difficult times. My brother Harshad and sister-in-law Shailja have always encouraged me with their insightful comments about career and the world. Without such a wonderful and supportive family, I don't think I would have had the courage and support to take on the task of doing a Ph.D., and I tip my hat to them.

Berkeley to me has been a wonderful place to spend the last few years, and a big reason for that has been the stellar quality of education that I feel I have received here. My advisors Professors Robert Brayton and Alberto Sangiovanni-Vincentelli

have been incredible influences on my life. Bob's gentleness, perseverance and meticulous scholarship has been something I admire greatly. Alberto's persistence, get-to-it-iveness, and ability to size up a situation in a short time was always something I marvelled at. I feel especially blessed to have had such stellar influences in my Ph.D. at Berkeley. The enthusiasm, insightfulness and straightforwardness of my qualifying committee chairman Professor Richard Newton has been something I valued greatly. Richard's energy and ability to balance so many duties fascinate me endlessly. Professor Kurt Kuetzer joined our group towards the later part of my studies at Berkeley, and I always appreciated his insights into the CAD problems we face today. Professor Shmuel Oren from the IEOR department, with his gentle graciousness, agreed to be on my qualifying exam committee, and I acknowledge that gesture gratefully. I would like to expressly thank Professor Dorit Hochbaum of the IEOR department for graciously agreeing to be on my thesis committee at short notice. Her help in this matter has helped me submit this thesis on time.

By sheer luck, my cubicle at Berkeley has always been shared by at least one post-doctoral researcher, and this has led me to come in contact with some of the most interesting people I have met at Berkeley. Eugene Goldberg was one such friend I made in this way, and it is always a pleasure to chat with Eugene, and obtain gems of wisdom from even casual conversation with him. Similarly, I got acquainted with Andreas Kuehlmann during his sabbatical in Berkeley, and it was great fun to hang out together, and discuss the future of Y2K, Microsoft and other burning issues of the day. Earlier such associations with Nagisa Ishihura and Atsushi Takahara were very enjoyable.

My years at Motorola in Austin were very enjoyable, and my friendship with my then-boss Gianfranco Gerosa has been very enriching. It was always been a pleasure for me to work with, and discuss shop with Gian. His unassuming and lets-get-it-done nature have been a source of great inspiration for me.

One of the biggest influences in my taking the Ph.D. route was Professor M. Ray Mercer. Professor Mercer and I have had innumerable long discussions about almost any topic on the face of the planet, including research and the meaning of life. Our discussions have always made me feel mentally invigorated, and I humbly acknowledge

his constant support for me before and during my Ph.D. studies.

Austin allowed me to form some wonderful friendships which I cherish greatly. Among the earliest friendships I formed was with Kapil Sabharwal, and his honesty and dedication in all matters has been something I have been very impressed by. I met Anup Tirumala while I was working at Motorola, and I learned a lot from his creativity and breadth of knowledge. Steven and Brenda Berger have been a couple who I always enjoyed meeting. Their simplicity and genuineness has been an inspiration for me. Similarly, my friendship with Kelly Baker has been very enjoyable, and his level-headedness and humility have been characteristics I admire greatly.

The students that I have been fortunate to interact with during my stay in Berkeley have all taught me a great deal. My early years at Berkeley were made extremely enjoyable by the company of Amit Narayan, Adrian Isles, and Ricky Ho, who had all joined the CAD group along with me. I have always been impressed with the calm wisdom and attention to detail of Wilsin Gosti, a friend whose company I have enjoyed very much. I enjoyed the company and advice of Sriram Krishan immensely, during my years at Berkeley. Amit Mehrotra's enthusiasm and high energy was always enjoyable and invigorating. Mukul Prasad's calm dignity and helpfulness is highly appreciated. It was enjoyable working with Subarna Sinha, whose common interest in SPFDs made for some interesting conversations. Philip Chong and William Jiang's thoroughness have made my work with them very interesting and enjoyable. It was always interesting to be around Sriram Rajamani, even though he decided to move over to the dark side and work for Microsoft. I acknowledge Edoardo Charbon's helpfulness with obtaining licenses for the CADENCE software which I used in my experiments. Rajeev Murgai's effervescence made my early years at UCB enjoyable. Among the earlier generation of CADgroup friends whose company I enjoyed very much were Paul Stefan, Henry Sheng, Jagesh Sanghvi, Tom Shiple and Gitanjali Swamy. I still fondly remember the Diwali parties that Jagesh held. I enjoyed very much my interactions and almost-weekly lunches with Sekhar Narayanaswamy, a person whose groundedness has always impressed me.

The administrative staff at Berkeley has some wonderful people, who not only are extremely competent at their jobs, but also are extremely easy to get along with.

Peggye Browne has been a wonderful and cheerful friend, and has gone out of her way to take care of my administrative requests even if it was not her assignment. Flora Oviedo, Kia Cooper, Diane Chang and Alberta Jackson have all made life at Berkeley easier by their professionalism and cheer. Tito Gatchalian and Jeffry Wilkinson were always happy to help with ERL reports, and always had a smile to brighten up my day.

I would also like to thank the sponsors of my research at UCB, which include the Semiconductor Research Corporation (SRC) and the Gigascale Research (GSRC/Marco) center at Berkeley, and the California MICRO program.

Chapter 1

Introduction

1.1 Cross-talk in DSM IC Design

With the rapid development of VLSI fabrication technologies, we have reached an era where the minimum feature sizes of the leading processes is well below 1 μm. Such processes are called Deep Sub-Micron (DSM) processes. With shrinking feature sizes, many new problems arise. Certain electrical problems like cross-talk, electro-migration, self-heat and statistical processing variations are becoming increasingly important. Until recently, IC designers were able to cleanly partition the design task into a logical and a physical one, with no interaction between the two sub-tasks. The increasing importance of the above electrical effects requires that designers consider the interaction between logical and physical design at the same time. This makes the design task more complex and time-consuming.

The *cross-talk* problem is perhaps the most important effect which jeopardizes the ability of designers to abstract the logical and physical aspects of design. Cross-talk typically occurs between adjacent wires on the same metal layer, when the cross-coupling capacitance between these wires is large enough for them to affect each other's electrical characteristics. As the minimum feature size of VLSI fabrication processes reaches the 0.1μm range, process engineers are forced to increase the height of wires in relationship to their width, in order to keep their sheet resistivity from increasing quadratically. This in turn increases the cross-coupling capacitance between

a wire and its neighbors as a fraction of its total capacitance, resulting in cross-talk problems.

Two distinct problems occur due to cross-talk. First, adjacent wires can affect each other's signal delay due to their large cross-coupling capacitance. The exact delay of a (*victim*) wire depends significantly on whether its neighboring wire(s) (*aggressors*) perform a transition as well. If the aggressors perform a transition, then the victim delay depends strongly on whether the aggressors perform a like or an unlike transition. Secondly, signal integrity of the victim wire is affected adversely. If a victim wire is held at a steady logic value, and aggressors perform a rapid transition, this change in voltage capacitively couples to the victim wire, causing a glitch in its waveform. If the magnitude of this glitch is large enough, the logic value of the victim wire may be incorrectly sampled. With the decreasing minimum feature size of VLSI fabrication processes, these problems are becoming increasingly common [Gro98].

1.2 Thesis Overview

In this work, we propose two VLSI layout methodologies which address the cross-talk problem in DSM integrated circuit design. In our methodology, the cross-talk problem is solved by design, by imposing a fixed pattern of wires on the IC die, on all metal layers. In particular, this repeating pattern, referred to as the Dense Wiring Fabric (DWF) pattern is $\cdots$ *VSGSVSGS* $\cdots$ where V represents a VDD wire, G represents a GND wire, and S represents a signal wire. This sequence of wires recurs on all the metal layers of the IC.

With this choice of layout fabric, the cross-coupling capacitance between signal wires drops by between one and two orders of magnitude, thereby all but eliminating the delay variation and signal integrity problems due to cross-talk. For example, in a 0.1μm process, the delay variation due to cross-talk drops from 2.01:1 to 1.03:1. Also, signal inductances drop by 50% as well, since the current return path for any signal wire is always adjacent to it. The uniformity of inductive and capacitive parasitics which results from the regularity of the DWF is a feature that CAD tools can exploit. Our layout "fabric" scheme eliminates the conventional notion of power and ground

routing on the integrated circuit die. Instead, power and ground are essentially "pre-routed" all over the die. By suitably introducing vias whenever VDD (or GND) wires intersect on adjacent metal layers, a power and ground distribution network of low and uniform resistance is created. The DWF also results in tighter tolerances on the inter-layer dielectric thicknesses because metal is maximally gridded all over the IC die. Finally, the DWF enables us to easily generate a low-skew global clocking network due to the tightly controlled and uniform parasitics. In our scheme, the characterization of interconnect parasitics (capacitance, inductance and resistance) becomes extremely simple due to the repeating nature of the DWF pattern. This characterization needs to be done only once for a design, resulting in significant time and computational savings.

In this thesis, we introduce two design methodologies that utilize the DWF pattern. The first, *Fabric1*, uses the DWF pattern chip-wide, including within the layout cells in the design. This fabric is incorporated into a traditional standard-cell based VLSI design flow, and over a series of examples, exhibits an area overhead of about 60% when two metal layers are utilized to route the design. If more metal layers are used, this overhead drops to approximately 17%. If three or more layers are used for routing, and if the DWF is not utilized in the routing area, then the area overhead of Fabric1 is about 2%. This shows that if cross-talk was not a problem in modern designs, the Fabric1 scheme would compete favorably with the existing standard cell based methodology.

Fabric3 attempts to reduce this area overhead. In this fabric, the logic network is implemented as a network of medium-sized Programmable Logic Arrays (PLAs). We choose an implementation of PLAs which is naturally cross-talk immune, and also extremely dense. The routing area between PLAs utilizes the DWF pattern, while a specialized layout fabric is utilized within the PLAs. We show that the delay of a *single* PLA is about 50%, and the area about 50% compared to that of a standard-cell based layout. We introduce synthesis algorithms to cluster a logic netlist into a network of PLAs in the Fabric3 style. Each of these PLAs has a bounded width and height. The number of inputs and outputs of each PLA are flexible as long as the resulting PLA width remains within the specified bound. We also perform folding on

the resulting PLAs to achieve better logic density. For a series of examples, the area penalty for a network of PLA implementation in the Fabric3 scheme is shown to be a mere 3% compared to the standard-cell implementation style. These timing and area comparison results remain essentially unchanged, regardless of whether 2, 3, 4, 5 or 6 metal layers are used to perform the routing.

On the other hand, if the network of PLAs is routed at minimum pitch on all metal layers (i.e. the DWF is not used in the routing region), then we obtain an overall timing improvement of about 15% and an overall area improvement of 20% compared to standard cells. Once again, these improvements remain essentially the same, regardless of whether 2, 3, 4, 5 or 6 metal layers are used to perform the routing. This comparison suggests that if cross-talk was not an issue (and therefore the DWF was not required), the Fabric3 scheme would still be a good choice over the standard cell methodology.

Since wires are routed at 2× minimum pitch in these methodologies, it is desirable to reduce the number of wires in a design. We introduce schemes for wires removal in the Fabric3 setting, and show they are effective in reducing the wiring, and hence the circuit area of a design implemented using the Fabric3 methodology. We obtain area reductions of about 11% using these techniques. This area reduction increases for the larger examples in our benchmark suite.

In summary, we show how the uniform and predictably low parasitics in our fabric methodology give rise to a reliable and predictable design style. We have implemented our scheme and compared it with the standard cell design styles. The crosstalk immunity, high speed, low area overhead, quick design turnaround time, and high predictability of our methodology indicate that it is a strong candidate as the preferred design methodology in the DSM era.

1.3 Thesis Outline

This dissertation is organized in the following manner. Chapter 2 lays the groundwork for this thesis. In this chapter, we introduce the VLSI processing technology predictions that our subsequent experiments are based on. Based on these predictions,

parasitics are extracted for on-chip wires on an IC. These parasitics are used to perform SPICE [Nag95] simulations to validate cross-talk effects like delay variation and signal integrity loss. Having shown that cross-talk is a significant problem for future VLSI design, we review the existing approaches to handle cross-talk. Among existing approaches, layout-based approaches and CAD-based approaches are discussed.

Chapter 3 introduces the basic idea of VLSI layout fabrics. The advantages and disadvantages of the fabric approach are discussed.

Chapters 4 and 5 introduce two design methodologies which both use the fabric idea of Chapter 3. Chapter 4 introduces a design methodology which is based on a standard-cell based design style. An area overhead of about 60% is exhibited in this scheme when two metal layers were used to route the design. If more layers are utilized, this penalty drops to about 17%.

Chapter 5 introduces a methodology which reduces this overhead significantly by utilizing a network of Programmable Logic Arrays (PLAs) to implement the logic circuit. Electrical characterization results for these PLAs are discussed. Algorithms to cluster a logic network into a network of PLAs are introduced. Experiments show that the area overhead using this methodology is 3%, with a delay improvement of 15%. These timing and area comparison results remain essentially unchanged, regardless of whether 2, 3, 4, 5 or 6 metal layers are used to perform the routing.

In Chapter 6, we outline techniques for performing wire removal in a network of PLAs. We show how these techniques are very effective in reducing the wiring, and hence the total circuit area of designs implemented using a network of PLAs. We obtain average area reductions of about 11%. This reduction is larger for the larger examples in our test suite.

Finally, in Chapter 7 we conclude the thesis, and discuss avenues for further work in this area.

Chapter 2

Validating Deep Sub-Micron Effects

2.1 Chapter Overview

In this chapter, we motivate our approach by a brief analysis on the trends of resistance and capacitance of on-chip interconnect with decreasing feature sizes of VLSI ICs, which is described in Section 2.2). We show analytically that the capacitance of a conductor to its neighboring conductors is becoming an increasing fraction of its total capacitance, thus giving rise to a situation where cross-talk problems become increasingly important. In Section 2.3, we detail our interconnect geometry predictions. Based on these predictions, we experimentally validate the above capacitance trends in Section 2.4. Finally, in Sections 2.5, we experimentally validate the delay variation (Section 2.5.1) and signal integrity (Section 2.5.2) problems by means of SPICE [Nag95] simulations.

2.2 Trends in DSM VLSI Interconnect

The resistance of the conductor in Figure 2.1 is given by: $R = \frac{\rho \cdot L}{A} = \frac{\rho \cdot L}{W \cdot T}$ where ρ is the resistivity of the wire material. It is assumed that the current flows along the "L" dimension. As the minimum feature size of a VLSI process decreases, the

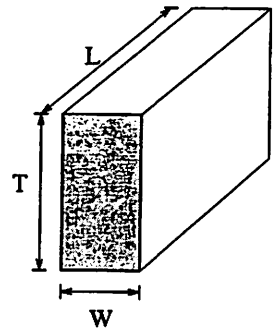


Figure 2.1: Resistance of a Conductor

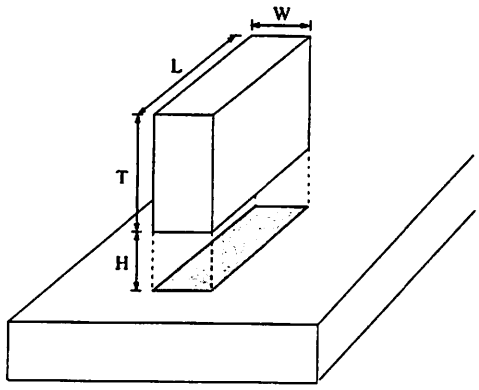


Figure 2.2: Plate Capacitance

resistance of a wire increases quadratically. This is because both T and W scale with the minimum feature size in general. Since a quadratic increase in resistance is unacceptable, the recent trend is to increase T in relation to W.

The capacitance of a conductor consists of two parts.

- The first is called *plate capacitance*, and models the capacitance of a wire to the conductor at the lower layer.

  In Figure 2.2, when $W \gg H$, the parallel plate model applies, so that $C = k \cdot \epsilon_0 \cdot \frac{W \cdot L}{H}$. Here k is the dielectric constant of the encasing material, and ϵ_0 is the permittivity of free space. When, however, $W \leq H$, the fringing model applies and $C = \alpha \; log(W)$.

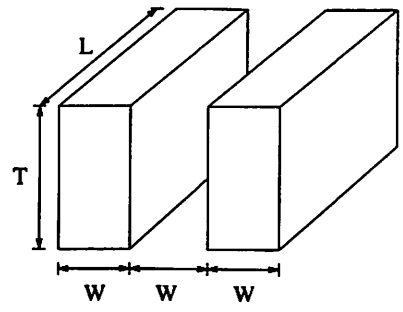


Figure 2.3: Edge Capacitance

For DSM processes, the fringing model applies. At 0.6 μm, W/H $\sim$ 2, so the parallel plate model was applicable, but at 0.35 μm and below, W/H $\sim$ 1, so the fringing model applies.

- The second is called *edge capacitance*. It models the capacitance of a wire to neighboring conductors on the same layer.

  In Figure 2.3, when $T \gg W$, the parallel plate model applies, and $C = k \cdot \epsilon_0 \cdot \frac{T \cdot L}{W}$. For $T \leq W$, the fringing model applies, and $C = \alpha\ log(T)$

  For DSM processes, the parallel plate model applies. At 0.6 μm, T/W $\sim$ 1, so the fringing model was applicable. But for 0.25 μm and below, T/W $\sim$ 3, so the parallel plate model applies.

From the above, we note that the edge capacitance is becoming the dominant capacitance for a wire, and that the plate capacitance contributes less to its total capacitance. In Section 2.4.1 we perform 3-dimensional parasitic extraction to verify this behavior.

From the above analysis, we also realize that increasing T/W decreases the resistance of a wire, but results in larger capacitances. It has been shown that increasing T/W beyond 2 does not cause a significant improvement in delay. Also, fabricating wires with T/W much more than 2 is difficult. Hence $T/W = 2$ is a practical choice of aspect ratio of the wires. In the sequel, we assume this value of *aspect ratio* for all metal layers.

2.3 Predicting VLSI Process Technology Trends

Our starting point for this research was to obtain estimates of interconnect geometries for future VLSI fabrication processes. Using the VLSI interconnect predictions from the NTRS [ntr97], as well as those from Sematech [Fis97], we came up with our "strawman" interconnect geometry parameters. We passed these parameters to several contacts in the VLSI design industry for their comments, and refined the estimates based on this feedback. The result of this exercise is detailed in Table 2.1. This table lists various process parameters for several processing generations. Here, V_{DD} refers to the power supply voltage, L_{eff} is the effective channel length of a transistor and t_{ox} is the gate oxide thickness of a transistor. For each conductor, H is the height, W its width, S refers to the minimum allowable spacing and t_{ins} is the thickness of the dielectric between metal layers. "Sheet ρ" refers to the sheet resistivity of the material, in ohms per square. Throughout this thesis, we assume that copper wires are used for every metal layer on the chip. The use of copper wires significantly alleviates electromigration, which was a significant problem for aluminum based processing technologies.

It is assumed that vias in our strawman technology are stackable. This means that a via between metal layers i and $i+1$ can be placed directly below a via between layers $i+1$ and $i+2$. This was not possible in prior process generations.

In the design methodologies we introduce in this thesis (Chapters 4 and 5), we perform all experiments using a 0.1 μm technology with copper interconnect, and a low-K dielectric. This technology is based on the parameters described in Table 2.1.

2.4 Extracting On-chip Layout Parasitics

Having established a "strawman" process technology for upcoming technologies, we now experimentally study the trends in interconnect parasitics as the feature size of modern VLSI fabrication processes decreases. Of the processes we considered in Table 2.1, the first two processes are used in aggressive circuit designs today, while the remaining processes are still a few years from being used. In our experiments

Process (μ)		0.25	0.18	0.13	0.10	0.07	0.05
V_{DD} (V)		2	1.8	1.5	1.2	0.9	0.6
L_{eff} (nm)		160	100	70	50	35	25
t_{ox} (Å)		60	45	35	30	20	12
Levels		6	6	7	8	9	9
Poly	H (μ)	0.2	0.15	0.13	0.1	0.07	0.07
	W (μ)	0.25	0.18	0.13	0.1	0.07	0.05
	S (μ)	0.25	0.18	0.13	0.1	0.07	0.05
	sheet ρ (Ω/□)	4	5.3	6.2	8	11.4	11.4
M1–2	H (μ)	0.5	0.46	0.34	0.26	0.2	0.14
	W (μ)	0.30	0.23	0.17	0.13	0.1	0.07
	S (μ)	0.30	0.23	0.17	0.13	0.1	0.07
	sheet ρ (Ω/□)	0.044	0.048	0.065	0.085	0.11	0.16
	t_{ins} (nm)	650	500	360	320	270	210
M3–4	H (μ)	2.0	2.0	1.2	1.0	0.6	0.6
	W (μ)	1.0	1.0	0.6	0.5	0.3	0.3
	S (μ)	1.0	1.0	0.6	0.5	0.3	0.3
	sheet ρ (Ω/□)	0.011	0.011	0.018	0.0224	0.036	0.036
	t_{ins} (nm)	900	900	900	900	900	900
M5–6	H (μ)	2.5	2.5	2.0	2.0	1.5	1.5
	W (μ)	2.0	2.0	1.0	1.0	0.75	0.75
	S (μ)	2.0	2.0	1.0	1.0	0.75	0.75
	sheet ρ (Ω/□)	0.009	0.009	0.011	0.011	0.015	0.015
	t_{ins} (nm)	1400	1400	900	900	900	900
M7–8	H (μ)	–	–	2.5	2.5	2.4	2.4
	W (μ)	–	–	2.0	2.0	1.2	1.2
	S (μ)	–	–	2.0	2.0	1.2	1.2
	sheet ρ (Ω/□)	–	–	0.009	0.009	0.0094	0.0094
	t_{ins} (nm)	–	–	1400	1400	900	900
M9 1	H (μ)	–	–	–	–	2.5	2.5
	W (μ)	–	–	–	–	2.0	2.0
	S (μ)	–	–	–	–	2.0	2.0
	sheet ρ (Ω/□)	–	–	–	–	0.009	0.009
	t_{ins} (nm)	–	–	–	–	1400	1400
Via (M1–M2)	size (μ)	0.5	0.36	0.26	0.2	0.14	0.1
	R (Ω)	0.46	0.69	0.95	1.43	2.16	3.27
κ		3.3	2.7	2.3	2	1.8	1.5

Table 2.1: "Strawman" process parameters

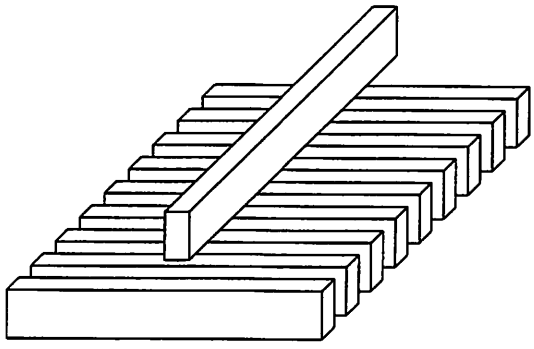


Figure 2.4: Single wire over a mesh

to project cross-talk and signal integrity trends into the future, we consider three processes from Table 2.1. These are the 0.25μm, 0.1μm and the 0.05μm process.

We utilize parasitic extractors to determine the parasitics for these three process technologies, and to validate the trends described in Section 2.2.

2.4.1 Capacitance

Interconnect parasitic characterization was experimentally performed using a 3-dimensional parasitic extractor called *Space3D* [spa]. The input to *Space3D* is a 3-dimensional circuit layout, and the output is the value of the parasitics between different features of that layout. *Space3D* uses a boundary element method to compute interconnect capacitances.

Figures 2.4 and 2.5 graphically describe the interconnect configurations for which we performed *Space3D* runs. Figure 2.4 represents a single wire of minimum width on metal layer i, which runs over a series of wires on metal layer $i-1$, which are separated from each other by the minimum spacing rule. This represents the extreme case where a wire is routed without any neighboring wires on the same metal layer. Since there are no capacitances to neighboring wires, such a routing would typically be done for speed-critical signals.

The other extreme case is shown in Figure 2.5. Here a series of wires on metal layer i of minimum width are routed at minimum spacing, and run over a series of

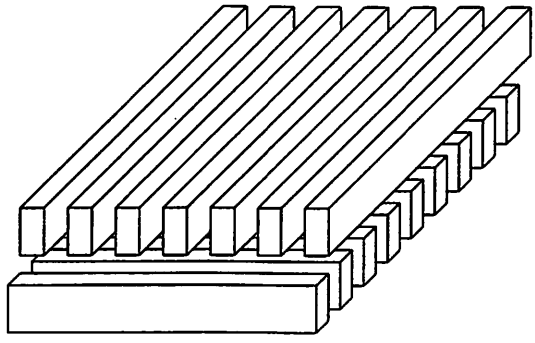


Figure 2.5: Mesh of wires

wires on metal layer $i-1$, which are also routed at minimum spacing. In this configuration, a wire would have higher capacitances to neighboring wires, and simultaneous transitions on neighboring wires could alter the delay of the wire of interest.

Table 2.2 reports the results of the *Space3D* runs on the configuration in Figure 2.4. Table 2.3 shows the results of the *Space3D* runs for the configuration in Figure 2.5. All *Space3D* runs were performed for all three processes described in the previous section, and for all metal layers.

In these tables, $C_{i,0}$ refers to the capacitance of a conductor on metal layer i, to ground. $C_{i,i}$ refers to the capacitance of a conductor on metal layer i, to its immediate neighbor. $C'_{i,i}$ is the the capacitance of a conductor on metal layer i, to its neighbor's neighbor. $C_{i,i+1}$ is the capacitance of a conductor to wires of the higher layer above it. All capacitances are reported per micron of length of the conductor of interest, and in the units of 10^{-18} F.

We observe that for Table 2.2, the capacitance to ground is larger in general than for Table 2.3. However, it is also true that for configuration of Figure 2.5, the total capacitance of a wire is larger than for the configuration of Figure 2.4. Hence, a signal in the configuration of Figure 2.5 is more susceptible to signal integrity and delay variation problems due to the switching of its neighbors. This is because for a wire in the configuration of Figure 2.5, a large fraction of the wire capacitance is to its neighboring wires.

In Table 2.3, we notice the trend of increasing $C_{i,i}$ with decreasing feature size.

Process	$C_{1,0}$	$C_{2,0}$	$C_{3,0}$	$C_{4,0}$	$C_{5,0}$	$C_{6,0}$	$C_{7,0}$	$C_{8,0}$	$C_{9,0}$
0.25μ	55.47	54.02	91.80	80.61	88.21	90.13	–	–	–
0.10μ	53.99	58.85	102.27	67.01	95.28	84.69	92.89	89.75	–
0.05μ	64.32	52.48	93.39	58.52	84.28	76.71	96.08	85.44	96.47

Process	$C_{1,2}$	$C_{2,3}$	$C_{3,4}$	$C_{4,5}$	$C_{5,6}$	$C_{6,7}$	$C_{7,8}$	$C_{8,9}$
0.25μ	40.46	22.5	50.70	59.82	63.65	–	–	–
0.10μ	34.52	30.54	43.67	62.04	65.03	67.48	70.08	–
0.05μ	35.69	48.87	37.40	52.82	58.33	69.35	67.11	57.24

Table 2.2: Capacitance for Single Wire of Figure 2.4 (10^{-18}F per μm)

Process	0.25μ				0.10μ				0.05μ			
Layer	$C_{i,0}$	$C_{i,i}$	$C'_{i,i}$	$C_{i,i+1}$	$C_{i,0}$	$C_{i,i}$	$C'_{i,i}$	$C_{i,i+1}$	$C_{i,0}$	$C_{i,i}$	$C'_{i,i}$	$C_{i,i+1}$
1	10.96	40.76	1.24	9.65	9.97	49.35	1.59	14.8	15.75	46.82	1.43	23.48
2	0.86	27.79	0.73	6.82	0.92	48.05	1.17	5.33	1.64	47.58	1.85	3.00
3	1.77	40.92	ϵ	31.81	2.58	45.84	1.46	30.14	5.38	48.28	1.11	12.45
4	0.96	41.83	ϵ	22.67	1.10	44.08	0.91	18.32	1.2	46.66	1.3	11.63
5	2.82	23.36	ϵ	41.87	2.31	39.84	ϵ	34.8	3.90	39.15	0.43	29.66
6	10.27	33.49	ϵ	–	1.07	40.59	0.55	22.04	1.51	38.33	0.15	39.2
7	–	–	–	–	3.00	23.51	0.25	38.65	3	37.61	0.355	40.27
8	–	–	–	–	10.63	30.80	2.66	–	1.57	38.95	0.14	27.15
9	–	–	–	–	–	–	–	–	13.43	31.50	1.96	–

Table 2.3: Capacitance for Intersecting Meshes of Figure 2.5 (10^{-18}F per μm)

Additionally, $C_{i,0}$ decreases in some cases with decreasing feature size. This capacitance is highly dependent on the value of t_{ins} for Metal1 in Table 2.1, and since this number decreases rapidly from 0.25μm to 0.05μm, the capacitance $C_{i,0}$ actually increases in many cases with decreasing feature sizes. However, in spite of this, the fraction of the cross-coupling capacitance of a wire to its total capacitance increases with decreasing feature size.

In some simulations, *Space3D* did not return the values of $C'_{i,i}$, (this is indicated in Table 2.3 as an "ϵ" symbol). This is probably a consequence of our setting the window-size parameter of *Space3D* to a small value. We made this choice in order to obtain faster runs. However, whenever the value of $C'_{i,i}$ was returned, it was at least an order of magnitude less than $C_{i,i}$.

In this section, we experimentally verified the analytic trend described in Section 2.2. We showed that the capacitance of a wire to its adjacent wires increases as a fraction of its total capacitance. Because of this, a signal's delay and integrity depend heavily on the switching activity of its neighboring wires.

2.5 Validating Cross-talk Effects

2.5.1 Delay Variation

Using the parasitics determined in Table 2.3, we constructed a SPICE model of an inverter driving a Metal2 wire. We modeled the two immediate neighbors of this wire similarly. At the end of each wire is a load corresponding to twice the gate capacitance of the driving inverter. The delay of the central victim wire varies depending on whether its aggressor neighbors undergo a like or unlike transition, or no transition at all. This delay variation is reported in Tables 2.4 and 2.5, for the three processes we consider, and for varying lengths of the wire.

Table 2.4 reports delays for a driving inverter of size 3× the size of a minimum inverter, while the driving inverter in Table 2.5 is 6× minimum. In both tables, all delays are measured in picoseconds, and are measured from the time the input voltage reaches *VDD/2* to the time the far end voltage reaches *VDD/2*. The L parameter denotes the length of each of the three wires in μm. t_f^+ represents the delay when both neighbors switch in the opposite direction compared to the center wire which switches from high to low. t_r^+ represents the same condition, except that the center wire switches from low to high. t_f^0 represent the delay when both neighbors are quiescent, while the center wire performs a high to low transition. t_f^- represent the delay when both neighbors switch in the same direction as the center wire which performs a low to high transition. t_r^- represent the delay when both neighbors switch in the same direction as the center wire which performs a low to high transition. The *max/min* column denotes the maximum delay variation of the center wire.

From tables 2.4 and 2.5 we note that for each of the process under consideration, the delay variation due to crosstalk (the *max/min* column) increases with the length

Process	L	t_f^+	t_r^+	t_f^0	t_f^-	t_r^-	*max/min*
0.25μm	25μm	96.44	93.85	89.99	78.91	79.92	1.222
0.1μm	25μm	52.35	53.18	43.90	38.71	38.56	1.379
0.05μm	25μm	23.34	29.02	19.75	19.68	16.78	1.729
0.25μm	50μm	108.36	105.49	95.45	80.39	85.78	1.348
0.1μm	50μm	63.74	65.08	51.69	39.86	39.64	1.642
0.05μm	50μm	30.72	39.27	23.66	20.58	17.73	2.215
0.25μm	75μm	126.15	120.77	102.04	82.06	85.18	1.537
0.1μm	75μm	78.72	82.43	57.39	41.23	40.93	2.014
0.05μm	75μm	39.06	50.40	27.43	21.77	18.74	2.689

Table 2.4: Delay variation for a Metal2 wire driven by a 3× minimum inverter

Process	L	t_f^+	t_r^+	t_f^0	t_f^-	t_r^-	*max/min*
0.25μm	25μm	85.76	82.89	81.78	76.79	79.16	1.117
0.1μm	25μm	44.18	45.20	39.97	37.61	37.39	1.209
0.05μm	25μm	19.28	23.57	18.02	19.59	16.56	1.423
0.25μm	50μm	91.62	88.83	85.06	77.15	79.75	1.188
0.1μm	50μm	51.83	53.16	43.78	38.13	37.88	1.403
0.05μm	50μm	24.19	29.80	20.27	20.17	17.30	1.723
0.25μm	75μm	97.65	97.35	88.44	79.45	80.39	1.229
0.1μm	75μm	58.17	59.71	47.80	39.45	38.51	1.551
0.05μm	75μm	28.66	35.04	22.81	20.83	18.06	1.940

Table 2.5: Delay variation for a Metal2 wire driven by a 6× minimum inverter

of the wires, as we would expect. Also, for each given length of wire, this variation is larger for processes with smaller feature sizes. This is because the capacitance of a wire to its neighboring wires becomes a larger fraction of its total capacitance with decreasing feature size.

We compare the delay variation of all three processes for the same length of wire, even though wire lengths scale with reducing feature sizes. This is appropriate because inter-module wires do not scale in DSM designs. Even though intra-module wires scale with decreasing feature sizes, the longer inter-module wires do not.

The delay variation for Table 2.5 is smaller than that of Table 2.4, since the driving inverters are larger, and hence the driving resistance smaller. This reduces the effect of the switching of neighboring wires. Nevertheless, the delay variation increases with increasing wire length, and with decreasing feature size, as observed in 2.4.

2.5.2 Signal Integrity

The increase in contribution of the edge capacitance to the total capacitance of a signal is a problem because adjacent wires in the same layer can in general be switching as well. These transitions on the adjacent wires can also cause signal integrity problems for the wire of interest.

To illustrate this, we conducted another experiment. Again using the parasitics determined in Table 2.3, we constructed a SPICE model of an inverter driving a Metal2 wire. We modeled the two immediate neighbors of this wire similarly. At the end of each wire is a load corresponding to twice the gate capacitance of the driving inverter. The central victim wire is held at a quiescent high value, while its aggressor neighbors undergo a high to low transition. The rapid change in voltage of the neighboring aggressors capacitively couples into the victim wire, giving rise to a glitch in its voltage waveform. If the magnitude of this glitch is large enough, the value of the victim wire can be erroneously sampled, resulting in incorrect circuit operation.

Table 2.6 reports the results of this experiment for a driving inverter of size 3× the size of a minimum inverter. The driving inverter in Table 2.7 is 6× minimum. In

Process	L	V_{glitch}	$\%VDD$
0.25um	100um	0.306	15.31
0.1um	100um	0.316	26.31
0.05um	100um	0.173	28.76
0.25um	200um	0.478	23.88
0.1um	200um	0.406	33.85
0.05um	200um	0.229	38.18
0.25um	300um	0.566	28.28
0.1um	300um	0.494	41.18
0.05um	300um	0.260	43.42

Table 2.6: Signal integrity for 3× minimum inverter

Process	L	V_{glitch}	$\%VDD$
0.25um	100um	0.193	9.66
0.1um	100um	0.210	17.48
0.05um	100um	0.116	19.39
0.25um	200um	0.312	15.62
0.1um	200um	0.317	26.42
0.05um	200um	0.178	29.59
0.25um	300um	0.409	20.46
0.1um	300um	0.383	31.88
0.05um	300um	0.231	38.43

Table 2.7: Signal integrity for 6× minimum inverter

both tables, the L parameter has the same meaning as in tables 2.4 and 2.5. The V_{glitch} column reports the magnitude of the voltage glitch on the victim wire, while the $\%VDD$ column reports the ratio V_{glitch}/VDD. This denotes the severity of the glitch. A glitch which is greater than $0.4\times VDD$ can cause incorrect sampling of the victim wire's logic value.

From Tables 2.6 and 2.7 we note that for each of the process under consideration, the signal integrity problem is aggravated with increasing wire lengths. Also, for a given length of wire, the magnitude of the glitch as a fraction of process voltage increases with smaller feature sizes. Again, this is because the capacitance of a wire to its neighboring wires becomes a larger fraction of its total capacitance with decreasing

feature size.

For all three processes under consideration, we compare the signal integrity characteristics for the same length of wire. This is appropriate because inter-module wires do not scale in DSM designs, even though intra-module wires scale with decreasing feature sizes.

For a given process and wire length, the signal integrity problem for Table 2.7 is less serious than that for Table 2.6, since the driving inverters are larger, and hence the driving resistance smaller. This reduces the susceptibility of the center wire to the transitions of the neighboring aggressor wires. However, the severity of the signal integrity problem increases with increasing wire length, and with decreasing feature size, as we noticed in 2.6.

2.6 Review of Existing Techniques

2.6.1 Overview

The problems faced in designing and manufacturing ICs using DSM fabrication processes are not entirely new. Over the past few years, these problems have slowly become significant enough that academia and industry have begun to take notice. As a result, techniques that address the cross-talk problem in DSM design are relatively recent.

In this section we review some of the existing techniques which address cross-talk in DSM VLSI design.

2.6.2 Layout-based Techniques

2.6.2.1 Ad-hoc Approaches

Cross-talk is typically handled in an ad-hoc manner in industry [Gro98]. The methodology utilized is to first create the logic and layout for a module. After the layout is generated, a separate engineer is assigned to the task of checking for cross-talk problems. This engineer analyzes the layout for signal pairs that have been

routed at minimum spacing, for more than a "critical" distance. This critical distance depends on the process geometry and the signal drive strengths. Now, the cross-talk specialist analyzes the circuit at the logic level, to determine if these signal pairs have logical and timing characteristics which could result in signal integrity or delay variation problems. If one signal is driven by a small driver, for instance, it could be a candidate for signal integrity problems. For such a signal, the critical distance would be small. If it is determined that both signals have similar timing characteristics, then delay variation problems can be exhibited.

Once the cross-talk specialist has extracted the problem signal pairs from the original set of signal pairs, this information is fed back to a layout designer, whose task it is to space the signals apart so that cross-talk problems are avoided.

The entire process depends on manual expertise, and is therefore slow and prone to errors. Also it requires the services of several additional designers, which adds to the design cost and design turn-around time.

2.6.2.2 Custom Layout

If the chip layout is performed manually, then cross-talk can be addressed at the time of layout design. Some design houses still depend on custom layout. This is especially practical for ICs with a large area under regular structures like datapaths and memory.

In this case, the timing and logical relationship between pairs of signals are known before-hand, and layout is performed to space apart signals that are likely to exhibit delay variation or signal integrity problems.

This approach too depends on manual dexterity, and is therefore prone to errors like the previous one.

2.6.2.3 The DEC Alpha Solution

In the past, some techniques to ensure characterizability and reliability of designs have been proposed and implemented. For example, in the DEC Alpha chip [G+97], metal layers 3 and 6 (in a 6 layer metal process) were exclusively dedicated to power

and ground routing. In this scheme, a signal on any of the remaining layers has a constant voltage plane either above or below it, thus ensuring that a larger fraction of its total capacitance is to a node of constant voltage. In the absence of such a scheme, signal wires on higher metal layers would have very small capacitances to a node of constant voltage, on account of their large distance from the substrate. Also, these wires would have relatively large capacitances to neighboring signal conductors. In such a situation, if the neighboring conductor would undergo a signal transition, it could couple a large noise voltage into the signal of interest, possibly resulting in a loss of signal integrity. It could also result in delay variations when the neighboring signal conductors undergo signal transitions while the wire of interest is also undergoing a transition.

The above solution worked well for the DEC Alpha microprocessor. However, with decreasing feature sizes, we find that the capacitance of a wire to its neighboring wire is an increasing fraction of the total capacitance of the wire. This was shown analytically and empirically in section 2.4.1. As a consequence, a solution such as the DEC Alpha's solution is not practical for future processes.

2.6.3 Design CAD based Techniques

2.6.3.1 Post-layout Methods

One class of solutions handles cross-talk after layout is generated, and is exemplified by the work of [RIXK94]. In [RIXK94], the layout is analyzed for signals that are liable to exhibit cross-talk. These pairs of signals are then subjected to a logical analysis to determine if a *cross-talk fault* can be justified and propagated to the primary outputs. This logical analysis filters out these pairs based on functionality. The remaining signals are flagged as potential cross-talk candidates, and a separate layout modification step is required to space them apart.

Such post-layout approaches are not expected to scale as cross-talk becomes an increasing problem with shrinking feature sizes of VLSI fabrication processes. This is because the number of pairs of signals that are potential cross-talk candidates increases significantly with diminishing feature sizes. Therefore, the computational

complexity of performing the required logical analysis will be extremely high.

2.6.3.2 Constraint-driven Methods

The second and more appealing class of solutions to the cross-talk problem was introduced in [KSV96]. In this work, the authors first analyze the logic netlist for pairs of signals that are likely to have a significant cross-talk interaction, such that this interaction is visible at the primary outputs. This step requires the computation of Compatible Output Don't-Cares [Sav92] and also requires image computation for sequential designs, both of which are computationally expensive tasks. Now the cross-talk candidates are passed as constraints to a cross-talk aware channel router [KSV94], which spaces them sufficiently apart in order to avoid cross-talk problems.

This work is appealing in that it handles cross-talk by design. However, as cross-talk becomes a significant problem in DSM IC design, it is anticipated that the number of such constraints will grow to an unacceptable level, making this approach impractical.

2.6.4 PLA based Techniques

Programmable Logic Arrays (PLAs) [GLL80, McC86] have recently experienced a renewed interest as a logic implementation style for high-performance designs. The IBM Gigahertz processor [PAB+98] utilized two-level PLAs to implement control logic. The stated reasons for this choice were high speed and the ability to quickly implement and modify the design. The predictable speed and area of the resulting circuit were also cited as reasons behind the choice.

As we observed in Chapter 2, one of the major detrimental effects of the cross-talk problem in DSM design is the unpredictability of signal delays. The use of PLAs in [PAB+98] alleviates this problem greatly.

2.6.5 Our Approach

Having discussed existing techniques to tackle the cross-talk problem in DSM IC design, we briefly outline our approach as well. Our approach is unique in that it eliminates cross-talk up-front and by design. It solves the cross-talk problem by all but eliminating it. This is achieved by utilizing a specialized arrangement of wires in the layout of the circuit. By ensuring that no two signal wires are ever routed adjacent to each other, we reduce cross-talk by between one and two orders of magnitude. Signal wires are always separated by *VDD* or *GND* wires. Whenever *VDD* (or *GND*) wires intersect on adjacent metal layers, we introduce metal vias. In this way, a high quality power and ground distribution network is also created. There are several other advantages to our approach as well, which will be discussed in Chapters 3, 4 and 5.

With our approach, cross-talk problems are essentially eliminated. In addition, inductive and capacitive parasitics are highly predictable and regular, giving rise to highly predictable designs with a low design turn-around time.

Other regular layout structures have been used in the past, for reasons of ease of programmability, and shorter times to market. FPGAs [Tri94, DGK94] and PLAs are an example. However these structures do not address DSM IC design problems. Our layout structures are designed with cross-talk in mind, and are unique in that sense.

2.7 Chapter Summary

In this chapter, we provided the motivation for our research by an analysis on the trends of resistance and capacitance of on-chip interconnect with decreasing feature sizes of VLSI ICs. We showed, both analytically and empirically, that the capacitance of a conductor to its neighboring conductors is becoming an increasing fraction of its total capacitance. Using our interconnect geometry predictions, we experimentally showed that the delay variation and signal integrity problems are getting progressively more serious with decreasing process feature sizes. In this way, we showed that cross-

talk problems are becoming increasingly important in DSM VLSI design.

Additionally, we reviewed some existing techniques utilized to address cross-talk in DSM designs. Among these techniques are layout-based approaches, which tend to be instance-specific and manpower-intensive. The CAD-based approaches include post-layout techniques, which we don't expect to scale well as process feature sizes decrease. The constraint-driven CAD-based approaches are more appealing, but are computationally expensive. PLAs have been used as a technique to control the timing uncertainty and high design turn-around time of existing methodologies.

Chapter 3

VLSI Layout Fabrics

3.1 Chapter Overview

In a typical VLSI IC, the preferred routing direction for any metal layer is usually perpendicular to the directions for layers above and below it. Other than this guideline, layout is performed without a strict prior arrangement of wires. This can easily give rise to situations where two or more wires are routed together for long distances on the same metal layer, resulting in cross-talk problems.

In order to curb the detrimental effects of cross-talk, our approach involves imposing a pre-defined layout "fabric" on the entire IC die. All wiring within the IC is required to conform to this fabric pattern. This repeating fabric pattern ensures that the cross-coupling capacitance between any two signal wires on the same metal layer is extremely small, and a very small fraction of the total capacitance of any one wire.

In this chapter, we outline the particular layout fabric proposed in this thesis. The characteristics of this fabric are outlined in Section 3.2. In Section 3.3 we describe the advantages of our scheme, while in Section 3.4 we describe its disadvantages.

3.2 Our Dense Wiring Fabric (DWF)

It is our opinion that DSM VLSI design using the familiar layout paradigms will not be feasible due to the problems described in Chapter 2. The additional analysis

required to design a reliable circuit using existing layout paradigms will prove to be prohibitive in cost and design turn-around time.

To eliminate the uncertainty in the effective capacitance of a wire and the resulting delay variation and signal integrity problems, we introduce a new layout methodology. The primary goal of our methodology is to ensure that each signal wire has the same immediate "electrical neighborhood". One advantage of this is that the wire has an effectively constant capacitance and inductance. The other advantage of this is that 3-dimensional parasitic characterization needs to be done only once for the *entire* circuit.

This goal is achieved by imposing a fixed pattern of wires on the IC die, on all metal layers. In particular, this repeating pattern, henceforth referred to as *Dense Wiring Fabric (DWF)* is $\cdots$ *VSGSVSGS* $\cdots$, where V represents a *VDD* wire, G represents a *GND* wire, and S represents a signal wire. The entire chip is maximally gridded with wires at minimum pitch. Wider wires can be implemented by discretely widening wires, in steps of $2 \cdot P$, where P is the wiring pitch. This is achieved by widening a wire so that it occupies the region originally designated for its neighboring *VDD* or *GND* wire and the other signal wire adjacent to this *VDD* or *GND* wire. We experimentally determined that this causes a negligible change in the power and ground resistance characteristics of the DWF. In the DWF, signal wires alternate with power and ground wires in the layout, on *all* metal layers. This fabric pattern ensures that adjacent signal wires are always capacitively shielded from each other. Metal wires on any layer run perpendicular to those on layers above and below it. The DWF arrangement is shown in Figure 3.1.

In addition, whenever *VDD* (or *GND*) wires on layer i intersect with *VDD* (or *GND*) wires on layer $i+1$, we introduce a metal via, and thus generate a power (and ground) distribution network for the IC. As we will see later, this network results in a high-quality power and ground distribution.

3.2.1 Parasitics in the DWF

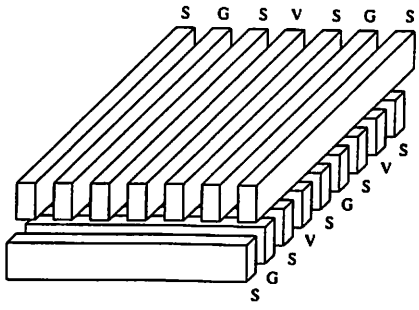


Figure 3.1: Arrangement of conductors in the DWF Fabric

3.2.1.1 Capacitance

In the DWF, every signal wire has a power wire as one neighbor, and a ground wire as the other neighbor. This presence of constant-valued nodes on either side of a signal wire ensures that *each* wire on a particular metal layer of the IC has the same parasitic capacitances per unit length. Also, the majority of the total capacitance of the wire is to the ground or power wires. The capacitance of any signal wire to the nearest *signal* wire is between one and two orders of magnitude smaller, hence the delay of a wire is effectively independent of the transitions of any of the other signal wires. This is true regardless of the whether other signals in the signal's neighborhood undergo any transitions. The delay variation of a wire due to transitions of a group of wires wires on higher or lower metal layers is discussed in Section 3.3.1. In these experiments, the worst case delay variation is shown to be 6% when a group of 128 wires on the next higher metal layer undergo transitions at the same time as a victim wire.

In order to quantify the interconnect parasitics of the DWF, we performed 3-dimensional parasitic extraction using *Space3D*, for a mesh-like configuration of wires shown in Figure 3.1. These extractions were based on the 0.25μm, 0.10μm and 0.05μm technologies described in Table 2.1. In the process technology we used, the width of the via plug between Metal1 and Metal2 is 1.33 times the minimum Metal1 width. In order to allow the place and route tools to place vias at any location along a wire, we fixed the Metal1 pitch to be twice the width of a Metal1 to Metal2 via, or 2.66 times

Process	0.25μ				0.10μ				0.05μ			
Layer	$C_{i,0}$	$C_{i,i}$	$C'_{i,i}$	$C_{i,i+1}$	$C_{i,0}$	$C_{i,i}$	$C'_{i,i}$	$C_{i,i+1}$	$C_{i,0}$	$C_{i,i}$	$C'_{i,i}$	$C_{i,i+1}$
1	17.82	17.38	ϵ	9.65	13.31	23.87	0.88	17.11	22.67	26.01	ϵ	39.51
2	1.46	17.26	0.69	16.03	1.35	26.14	1.03	9.83	2.05	27.91	1.71	3.84
3	8.91	21.18	0.25	48.93	2.05	32.61	2.83	23.57	5.08	29.73	1.23	4.0
4	1.67	19.68	0.34	31.76	1.24	26.76	0.40	22.94	1.03	29.20	1.03	14.35
5	0.88	11.05	ϵ	49.47	1.44	21.70	0.27	42.73	3.35	20.84	0.29	34.89
6	17.24	17.85	0.71	–	0.94	23.18	0.32	27.71	2.64	20.65	1.08	32.69
7	–	–	–	–	3.44	11.39	0.33	48.42	2.62	19.26	0.22	45.07
8	–	–	–	–	10.82	19.69	2.31	–	1.55	23.35	0.128	29.86
9	–	–	–	–	–	–	–	–	13.37	18.96	1.74	–

Table 3.1: 3-Dimensional Parasitics for DWF scheme (in 10^{-18}F per μ)

the minimum Metal1 width. The results of the 3-dimensional parasitic extraction are detailed in Table 3.1.

In Table 3.1, $C_{i,0}$ refers to the capacitance of a conductor on metal layer i, to ground. $C_{i,i}$ refers to the capacitance of a conductor on metal layer i, to its immediate neighbor. $C'_{i,i}$ is the capacitance of a conductor on metal layer i, to its neighbor's neighbor. $C_{i,i+1}$ is the capacitance of a conductor to wires of the higher layer above it. All capacitances are reported per micron of length of the conductor of interest, and in the units of 10^{-18} F.

In Table 3.1, we notice the general trend of an increase in $C_{i,i}$ with decreasing feature size. Other observations made for Table 2.3 generally hold here as well.

In some simulations, *Space3D* did not return the values of $C'_{i,i}$ (this is indicated in Table 3.1 as an "ϵ" symbol). This is a consequence of our setting the window-size parameter of *Space3D* to a small value. We made this choice in order to obtain faster runs. However, whenever the value of $C'_{i,i}$ was returned, it was at least an order of magnitude less than $C_{i,i}$.

We observe that the capacitance of a signal to its nearest *signal* wire (i.e. it's neighbor's neighbor) is between one and two orders of magnitude smaller than the capacitance to its immediate neighbor. Since the neighbors of a signal wire are *VDD* and *GND* wires, the majority of the capacitance of a signal wire is to a stable node. Therefore, the ratio of the cross-coupling capacitance of a wire to its total capacitance is very low. Hence cross-talk problems are significantly reduced in the DWF.

If wires in the design were spaced apart such that the pitch between wires was twice the minimum pitch (but there were no *VDD/GND* conductor between any adjacent signal wires), the cross-coupling capacitances were about 3× less than the traditional layout style. This indicates that the presence of intervening *VDD/GND* wires in our DWF fabric has the dual benefit of significantly reducing cross-coupling capacitances between signal wires, and also providing a power and ground distribution network for the IC.

It may appear that the total capacitance of a given length of wire in our scheme would be much larger than that of a wire in the arrangement of Figure 2.4. From Table 2.2, we determine that the total capacitance of a Metal2 wire is 58.85×10^{-18} F/μm (for a 0.1 μm process). Using our layout methodology, the total capacitance of a Metal2 wire is $(26.14 \times 2 + 9.83) \times 10^{-18} = 62.11 \times 10^{-18}$ F/μm. This is an increase of only 5.5%. Similarly close values are obtained for other metal layers. This shows that the total capacitance of a wire in our scheme is comparable to that of a wire routed for maximum speed.

3.2.1.2 Inductance

As DSM circuits move to gigahertz speeds, on-chip signal inductances start affecting wire delays [Ger99]. This is more severe in top layer metal wires.

We use [Lia80, TS86] to determine the inductance of a Metal8 conductor routed in our scheme versus a single Metal8 conductor routed over substrate.

When waveforms make transitions over time intervals which are smaller than the time of flight of a wire trace, then the transmission-line character of the wire becomes apparent. We now proceed to show that longer wires in our designs would exhibit transmission line effects.

The velocity of propagation u is

$$u = \frac{c}{\sqrt{\kappa}} \tag{3.1}$$

Here c is the speed of light in free space, and κ is the dielectric constant of the enclosing material.

In the 0.1μm process technology described in Table 2.1, $\kappa = 2.0$. This gives $u = 2.121 \times 10^8$ m/s. For a time of flight of 50 picoseconds, the length of the wire is about 1 centimeter. This means that if a wire of length 10 millimeters is driven by a signal with a risetime of less than 50 picoseconds, it would exhibit transmission-line effects. In the 0.1μm process, the signal rise times are less than 50 picoseconds (based on the tables in Section 3.3.1) and hence longer global wires in this process would exhibit transmission line effects.

For a transmission line, the velocity of propagation u is

$$u = \frac{1}{\sqrt{LC}} \tag{3.2}$$

where L and C are the inductance and capacitance of the line per unit length.

Now, using the value of C for a Metal8 wire from Table 3.1, we get a uniform inductance of $2.153 \times 10^{-4} nH/\mu$m of wire in the DWF scheme. Using *Space3D*, the capacitance of the single Metal8 wire over substrate was determined to be 51.9×10^{-18} F/μm. Hence this configuration has an inductance of $4.283 \times 10^{-4} nH/\mu$m of wire.

These numbers were experimentally verified using ASITIC [asi]. ASITIC is a CAD tool that aids the analysis and modeling of passive metal structures residing on a lossy conductive substrate. These structures includes inductors, transformers, capacitors, transmission lines, interconnect, and substrate coupling analysis.

The inductance value for a wire in a traditional layout style is very hard to determine in general. This number is highly dependent on the exact local layout topology, and is in general unpredictable using the existing layout paradigms of today. This is because different wires have current return paths different distances apart. The current return path is also dependent on the input vector. This gives rise to input vector dependent inductances, which is very undesirable.

However, in our scheme the current return path for each wire is adjacent to it. This ensures that the inductance of the wire is very low. Also, since the layout topologies are fixed, wires in our scheme have a predictable and uniform inductance value.

3.3 Advantages

There are several advantages to the DWF fabric, which we elaborate in this section.

3.3.1 Delay variation

In the DWF scheme, the capacitance of any signal wire is entirely predictable, and can be obtained simply by multiplying the corresponding entries in Table 3.1 by the wire length. This makes characterization of wire delays extremely easy. Since the capacitance of a signal wire to the nearest signal wire on the same metal layer is negligible, two adjacent signal wires on the same metal layer do not affect each others signal integrity or delay.

To verify this, we ran an experiment to determine the delay variation of a signal wire when its neighboring signal wires undergo simultaneous transitions.

Using the parasitics determined in Table 3.1, we constructed a SPICE model of an inverter driving a Metal2 wire. We modeled the two immediate neighbors of this wire similarly. At the end of each wire was a load corresponding to twice the gate capacitance of the driving inverter. The delay of the central victim wire varies depending on whether its aggressor neighbors undergo a like or unlike transition, or no transition at all. This delay variation is reported in Tables 3.2 and 3.3, for the three processes we consider, and for varying lengths of the wire. This experiment is very similar to that described in Section 2.5

Table 3.2 reports delays for a driving inverter of size 3× the size of a minimum inverter, while the driving inverter in Table 3.3 is 6× minimum. In both tables, all delays are measured in picoseconds, and are measured from the time the input voltage reaches *VDD/2* to the time the far end voltage reaches *VDD/2*. The L parameter denotes the length of each of the three wires in μm. t_f^+ represents the delay when both neighbors switch in the opposite direction compared to the center wire which switches from high to low. t_r^+ represents the same condition, except that the center wire switches from low to high. t_f^0 represent the delay when both neighbors are

Process	L	t_f^+	t_r^+	t_f^0	t_f^-	t_r^-	*max/min*
0.25μm	25μm	89.49	86.46	88.95	83.17	88.73	1.07
0.1μm	25μm	41.55	41.79	41.44	41.56	41.33	1.01
0.05μm	25μm	18.84	22.14	18.74	21.76	18.62	1.18
0.25μm	50μm	95.25	92.97	94.80	91.24	94.49	1.04
0.1μm	50μm	47.27	47.94	46.93	47.21	46.57	1.02
0.05μm	50μm	21.37	25.86	21.11	25.16	20.89	1.23
0.25μm	75μm	100.39	98.50	99.99	99.53	99.59	1.01
0.1μm	75μm	52.47	53.24	52.06	52.38	51.63	1.03
0.05μm	75μm	24.86	29.23	24.48	28.29	24.05	1.21

Table 3.2: Delay variation for a Metal2 wire driven by a 3× minimum inverter in the DWF

Process	L	t_f^+	t_r^+	t_f^0	t_f^-	t_r^-	*max/min*
0.25μm	25μm	82.26	80.82	81.32	80.81	81.23	1.01
0.1μm	25μm	38.99	39.30	38.98	39.17	38.92	1.01
0.05μm	25μm	17.55	20.61	17.51	20.49	17.45	1.18
0.25μm	50μm	85.18	82.51	83.96	84.00	83.81	1.03
0.1μm	50μm	41.55	42.66	41.06	42.34	40.88	1.04
0.05μm	50μm	19.35	22.89	19.24	22.51	19.13	1.19
0.25μm	75μm	87.85	85.31	86.58	88.05	86.34	1.03
0.1μm	75μm	44.67	45.69	44.10	45.22	43.86	1.04
0.05μm	75μm	20.93	25.23	20.77	24.67	20.61	1.22

Table 3.3: Delay variation for a Metal2 wire driven by a 6× minimum inverter in the DWF

quiescent, while the center wire performs a high to low transition. t_f^- represent the delay when both neighbors switch in the same direction as the center wire which performs a low to high transition. t_r^- represent the delay when both neighbors switch in the same direction as the center wire which performs a low to high transition. The *max/min* column denotes the maximum delay variation of the center wire.

From Tables 3.2 and 3.3 we observe that the delay variation due to crosstalk (the *max/min* column) is extremely low, since the cross-coupling capacitance between adjacent signal wires is significantly reduced compared to the traditional method. When we compare Table 3.2 with Table 2.4, we notice that the DWF concept results

in a great reduction in delay variation, for each combination of process and wire length. For example, in the 0.1μm process, for wires of length 75μm, the delay variation reduces from 2.014:1 to a mere 1.03:1 when the DWF concept is used.

Similar improvements are obtained when we compare Table 3.3 with Table 2.5.

This shows that the use of the DWF results in a significant improvement in delay variation. This improvement is obtained because the capacitance of any signal wire is entirely predictable, since the immediate neighbors of any signal wire are always one power and one ground wire. The capacitance of a signal wire to the nearest signal wires is negligible. We determined that the capacitance of a wire to its immediate neighbor is at least 10 to 15 times larger than its capacitance to its neighbor's neighbor. Hence the effect, on any signal, of its neighboring signal wires undergoing transitions is negligible.

Cross-talk can also occur when a series of wires on the metal layer immediately above or below a victim wire simultaneously transition at the time the victim wire undergoes its transition. The most common way in which this condition can occur on an IC is when all the wires of a bus undergo a transition (usually these transitions are simultaneous) while the victim wire undergoes its own transition. We simulated this condition for a victim wire on Metal2, which is driven by a 3$\times$ minimum inverter. It has a fixed load which corresponds to an inverter twice its size, and a length equal to the width of the bus above it. The wiring structure assumes the use of the DWF. Each Metal3 wire on the bus is driven by a 6$\times$ minimum inverter. The results of this experiment are reported in Table 3.4. The second column of this table lists the width of the bus. Columns 3 through 8 have the same meaning as in Tables 3.2 and 3.3.

Table 3.4 shows that cross-talk from adjacent layers is not a significant problem in the DWF. The cross-talk problem is greater for a 0.25μm process compared to a 0.1μm process. This is because the capacitance of a Metal2 wire to Metal3 wires above it is a smaller fraction of the total capacitance of the Metal2 wire in the 0.1μm process, as indicated in Table 3.1.

Even though the capacitance of a Metal2 wire to Metal3 wires above it is a smaller fraction of the total capacitance of the Metal2 wire in the 0.05μm process than in the 0.1μm process, cross-talk in the 0.05μm process is greater than in the 0.1μm

Process	#bits	t_f^+	t_r^+	t_f^0	t_f^-	t_r^-	max/min
0.25μm	32	83.90	83.90	82.86	81.14	81.79	1.03
0.1μm	32	38.10	38.45	37.89	38.09	37.72	1.02
0.05μm	32	26.80	27.99	26.76	27.91	26.71	1.05
0.25μm	64	85.98	86.52	84.03	81.46	82.07	1.06
0.1μm	64	38.44	38.80	38.08	38.12	37.75	1.03
0.05μm	64	26.88	28.08	26.80	27.91	26.72	1.05
0.25μm	128	89.51	90.57	86.47	82.50	82.89	1.10
0.1μm	128	39.06	39.46	38.45	38.23	37.85	1.04
0.05μm	128	27.04	28.24	26.89	27.93	26.74	1.06

Table 3.4: Delay variation for a Metal2 wire in the DWF, with aggressor bus on a higher metal layer

process. This is attributed to the fact that the transistors are significantly faster in the 0.05μm process. This results in a higher rate of change of the aggressor signal's voltage, resulting in a more aggravated cross-talk effect. In fact, the transistors in the 0.1μm process are faster than those in the 0.25μm process. As a result, the cross-talk improvement in the 0.1μm process over the 0.25μm process is not as high as the reduction in the fraction of the capacitance of a Metal2 wire to Metal3 wires above it compared to the total wire capacitance.

3.3.2 Signal Integrity

Signal integrity problems are also significantly improved by using the DWF, since signal integrity is a strong function of the capacitance of a signal to its neighbors. Since the cross-coupling capacitance to adjacent signal wires drops significantly, signal integrity improves dramatically in our scheme.

To illustrate this, we conducted an experiment similar to the one described in Section 2.5.2. Using the parasitics determined in Table 3.1, we constructed a SPICE model of an inverter driving a Metal2 wire. We modeled the two immediate neighbors of this wire similarly. At the end of each wire is a load corresponding to twice the gate capacitance of the driving inverter. The central victim wire is held at a quiescent high value, while its aggressor neighbors undergo a high to low transition. The rapid

Process	L	V_{glitch}	$\%VDD$
0.25μm	100μm	0.006	0.31
0.1μm	100μm	0.006	0.48
0.05μm	100μm	0.011	1.96
0.25μm	200μm	0.007	0.37
0.1μm	200μm	0.007	0.61
0.05μm	200μm	0.013	2.22
0.25μm	300μm	0.009	0.44
0.1μm	300μm	0.007	0.62
0.05μm	300μm	0.015	2.44

Table 3.5: Signal integrity for 3× minimum inverter in DWF

Process	L	V_{glitch}	$\%VDD$
0.25μm	100μm	0.004	0.20
0.1μm	100μm	0.004	0.33
0.05μm	100μm	0.010	1.83
0.25μm	200μm	0.006	0.31
0.1μm	200μm	0.006	0.49
0.05μm	200μm	0.012	2.14
0.25μm	300μm	0.007	0.35
0.1μm	300μm	0.007	0.57
0.05μm	300μm	0.009	1.47

Table 3.6: Signal integrity for 6× minimum inverter in DWF

change in voltage of the neighboring aggressors capacitively couples into the victim wire, giving rise to a glitch in its voltage waveform. If the magnitude of this glitch is large enough, the value of the victim wire can be erroneously sampled, resulting in incorrect circuit operation.

Table 3.5 reports the results of this experiment for a driving inverter of size 3× the size of a minimum inverter. The driving inverter in Table 3.6 is 6× minimum. The V_{glitch} column reports the magnitude of the voltage glitch on the victim wire, while the $\%VDD$ column reports the ratio V_{glitch}/VDD. This denotes the severity of the glitch. As mentioned earlier, a glitch with a magnitude greater than 40% of VDD can cause incorrect sampling of the victim wire's logic value.

Process(μ)	$R_{DWF}(\Omega)$	$R_{REG}(\Omega)$
0.25	0.17 - 0.24	5.5
0.10	0.39 - 0.54	10.63
0.05	0.68 - 0.86	20.1

Table 3.7: Power and Ground resistance

From Tables 3.5 and 3.6 we note that for each of the processes under consideration, the signal integrity problem is aggravated with increasing wire lengths. For a given length of wire, the magnitude of the glitch as a fraction of process voltage increases with smaller feature sizes. However, when we compare Table 3.5 with Table 2.6, we note that the magnitude of the glitch is significantly reduced in the DWF scheme, for the same process and wire length. For example, for a 0.1μm wire of length 200μm, the glitch magnitude drops from 33.85% of *VDD* to 0.61% of *VDD*. For the traditional scheme of Table 2.6, the maximum glitch magnitude is 43.42%, while the maximum glitch magnitude for the DWF scheme is 2.44%. Similar results are observed when we compare Table 3.6 with Table 2.7.

For a given process and wire length, the signal integrity problem for Table 3.6 is less serious than that for Table 3.5, since the driving inverters are larger, reducing the susceptibility of the center wire to the transitions of the neighboring aggressor wires.

3.3.3 Power Supply Resistance

Routing of power and ground on the entire chip is automatic in our scheme. At every point where a power (or ground) wire on metal layer i overlaps with a power (or ground) conductor on metal layer $i + 1$, a via is introduced. Given the large number of such intersections, the power and ground resistance at every point is held low, and almost constant. We extracted the resistive mesh corresponding to the power and ground networks, and ran this through SPICE to determine the effective power and ground resistance. Table 3.7 reports these results. We probed the power network resistance on Metal1 at different points in the power network, and report the maximum and minimum values of the power and ground resistance over the entire

mesh under the heading R_{DWF}. These probes were placed at various points on the die. For any metal layer, probes were placed on Metal1 at the vertices and center points of the smallest square enclosing a group of four power or ground vias on that metal layer. The absolute value of this resistance was extremely low, and its variation was within 50%. On the other hand, it is not easy to estimate the power and ground resistance of the existing routing methodologies, since they are extremely ad-hoc. For a comparison, we assume a standard cell methodology, with rows of length 1000 times minimum Metal1 width, and a width of 8 times minimum Metal1 width. The rows are powered from one end. The power or ground resistance at the end of such a row is listed under the heading R_{REG} , and is about 20 times larger than that obtained using our routing methodology.

This gridding of power and ground gives the layout the appearance of a "fabric".

3.3.4 Uniform and Predictable Inductance

As DSM circuits move to gigahertz speeds, on-chip signal inductances start affecting wire delays [Ger99]. This is more severe in top layer metal wires, and for wires carrying signals with fast slew-rates (like clock signals).

As we described in Section 3.2.1.2, the use of the DWF results in a significant decrease in the inductance of on-chip wires. This is because each signal in the DWF has a current return path which is adjacent to it. Since the layout topology is fixed in the DWF, wires in our scheme have a predictable and uniform inductance value.

However, the inductance of wires in the traditional layout style is highly dependent on the local layout topology, and hence difficult to predict. This is because different wires have current return paths a different distance apart. Additionally, the current return path is also dependent on the input vector.

3.3.5 Tighter control of t_{ins}

It has been empirically observed that for large chips with varying local densities of metalization, t_{ins} varies locally as well, since the changing metalization density makes it difficult to obtain a constant t_{ins} during the chemical-mechanical polish

(CMP) [Fur97] phase of processing. This in turn causes local changes in capacitance which is undesirable.

Our scheme results in a constant density of wires in any region of the chip, and on every metal layer. This has the added advantage that it results in a much tighter control of t_{ins}, which in turn results in more predictable capacitances across the die.

3.3.6 One-time Parasitic Extraction

In Chapter 2 we showed that the capacitance of a wire to its adjacent wires increases as a fraction of its total capacitance, with decreasing process feature size. Because of this, a signal's delay and integrity depend heavily on the switching activity of its neighboring wires. In the traditional design style, there could be a large number of unique layout configurations used in the IC. In order to design a reliable IC, each unique interconnect configuration would need separate parasitic characterization. This would involve running a 3-dimensional parasitic extractor (similar to *Space3D*), which is highly compute-intensive. Additionally, the large amount of data from these extractions would be difficult to manage.

In the DWF scheme, however, there is only one wiring configuration that needs to be characterized. This increases the ability to design reliable ICs with a quick design turn-around time.

3.3.7 Applicability of CAD Techniques

Another advantage of our scheme is that it is easy for CAD tools to make use of the regularity and predictability of parasitics to their advantage. For instance, it was shown [OB98a] that the regular layout structures give rise to predictable delays and a notion of "critical length" of a wire segment. A "critical length" is the optimum spacing between buffers to minimize the delay of a wire. Wires of critical length on any layer have the same "critical delay". More efficient CAD algorithms can be devised taking this theory into account.

Further, by modifying the minimum width and spacing of each layer, the delays of different metal layers can be tuned to the designer's specifications.

3.3.8 Routing Flexibility

This special layout style does not require a major modification to the existing routing tools. For our routing experiments, the routing was achieved by using a routing pitch which was twice the normal value. Power and ground routing can be achieved trivially after the signal wires have been routed.

3.3.9 Applicability to Special Circuits

Specialized circuits like memories can be handled easily within our layout scheme. Given the regular nature of memory structures, it is not difficult to see that our scheme lends itself naturally to such structures. We created the layout for an SRAM and a DRAM cell in our layout scheme, and found that this could be done without an area penalty. This is because of the rich availability of *VDD* and *GND* in the DWF. Similarly, we expect that other regular structures like data-paths would also map cleanly into our layout scheme.

Local breaks of the fabric structure on lower metal layers does not appreciably increase the power or ground resistance of the DWF, since the major contribution to the low power and ground resistance is made by the higher metal layers. This was experimentally verified. This flexibility allows us to compactly implement regular structures.

3.3.10 Global Clocking

The common method of distributing clock signals on ICs is referred to as an *H-tree*. Such an H-tree structure is shaped like the letter "H", and is driven from the center of the "H". The clock signals at the endpoints of the vertical limbs of the "H" have the same arrival time since the total wire length from the center of the "H" to the endpoints is identical. This is true if the parasitic capacitances and inductances of each of the traces is identical, which is difficult to ensure in practice.

Global clocking of a chip using our layout scheme would be easily achieved. We would reserve a series of grid wires for the clock signal. These grid wires would form

an *H-tree* structure. Since metal layers i and j run in perpendicular directions, it is easy to construct such a H-tree structure. The uniform capacitive and inductive parasitics of each of limbs of the "H" would ensure equal skew at each endpoint of the clock H-tree.

3.4 Disadvantages

The DWF scheme has some disadvantages, which are listed below.

3.4.1 Chip Area

The main disadvantage of the DWF technique is that it could result in a larger chip-wide area utilization. This would seem intuitive, since signal wiring is performed at twice-minimum pitch. So wiring intensive circuits are likely to utilize more area.

In Chapters 4 and 5, we introduce two design methodologies and compare the chip area utilization for DWF-based designs and standard cell-based designs.

For the first methodology, which we call Fabric1, we obtain an area penalty of about 60% if two metal layers are utilized for routing. This penalty drops to about 17% if more metal layers are utilized.

For the second methodology, which utilizes a network of PLAs to implement the circuit, the area penalty is shown to be about 3% regardless of the number of metal layers utilized to complete the routing.

3.4.2 Total Capacitance

The DWF technique results in an increase in the total capacitance of a wire, as described in Section 3.2.1.1. However, this increase is a mere 5.5%. In spite of this increase, designs using the DWF exhibit significantly lower maximum signal delays compared to the traditional design style (compare Table 3.2 with Table 2.4, and Table 3.3 with Table 2.5). This is because of the low inter-signal capacitance of the DWF compared to the traditional design style, which results in a very low delay *variation* due to cross-talk.

3.5 Chapter Summary

In this chapter, we introduced the DWF fabric concept. We reported results of capacitance and inductance extraction experiments on the DWF structure. We showed that cross-coupling capacitance between adjacent signal wires drops significantly in the DWF compared to the traditional layout methodology. The inductance of wires drops as well. We described the advantages of the DWF scheme, which are briefly outlined below:

- First, the capacitance of any signal wire is entirely predictable, since the immediate neighbors of any signal wire are always one power and one ground wire. The capacitance of a signal wire to the nearest signal wires is negligible. (at least 10 to 15 times smaller than its capacitance to its neighbor). Hence the signal integrity and delay variation problems are all but eliminated.

- Secondly, the routing of power and ground to the entire chip is automatically achieved in the DWF. At every point where a power (or ground) wire on metal layer i overlaps with a power (or ground) wire on metal layer $i + i$, a via is introduced. Given the large number of such intersections, the power and ground resistance at every point is held very low, and almost constant.

- Thirdly, each signal has a current return path which is adjacent to it, hence its inductance is very low. In the existing layout paradigms of today, the inductance of signals can vary greatly, since different wires have current return paths a different distance apart, depending on the exact layout of the circuit in the neighborhood of the signal.

Chapter 4

Fabric1 - Fabric Cell Based Design

4.1 Chapter Overview

In the Fabric1 design methodology, library cells correspond to the standard cells of a standard-cell methodology. We call our library cells *fabric cells*. The routing area between instances of the fabric cells utilizes the DWF fabric. The layout of a fabric cell utilizes the DWF fabric internally. As a result, the DWF fabric is utilized all over the IC layout.

In terms of logic synthesis, technology mapping and placement/routing, the Fabric1 methodology follows the same steps as the standard cell methodology. Thus, in an IC design, the use of a Fabric1 methodology in place of the traditional standard-cell based methodology requires minimal infrastructural changes.

We compare the Fabric1 methodology with the standard-cell methodology by employing two separate design flows. In both design flows, logic synthesis and technology mapping are performed using SIS [SSL+92]. The first design flow (Section 4.2) uses public domain placement and routing tools. The routing tool utilizes the channel routing methodology [YK82, RF82, BP83, SVSR84]. Two metal layers are used for routing. In the second design flow, described in Section 4.3, commercial placement and routing tools are used. The routing tool employs an area routing methodology and allows the use of up to six metal layers.

4.2 Design Flow 1

In this section, we compare the area utilization of the Fabric1 design methodology with that of a standard-cell based design methodology. The detailed routing of the design is performed using a channel routing methodology, using two metal layers for routing.

4.2.1 Design Methodology

In the Fabric1 methodology, the DWF fabric is used all over the IC layout, including within the layout cells of the design. The cells in the layout library in this methodology are functionally equivalent to the cells of the standard cell library that we use for comparison. We refer to our layout cells as *fabric cells.*

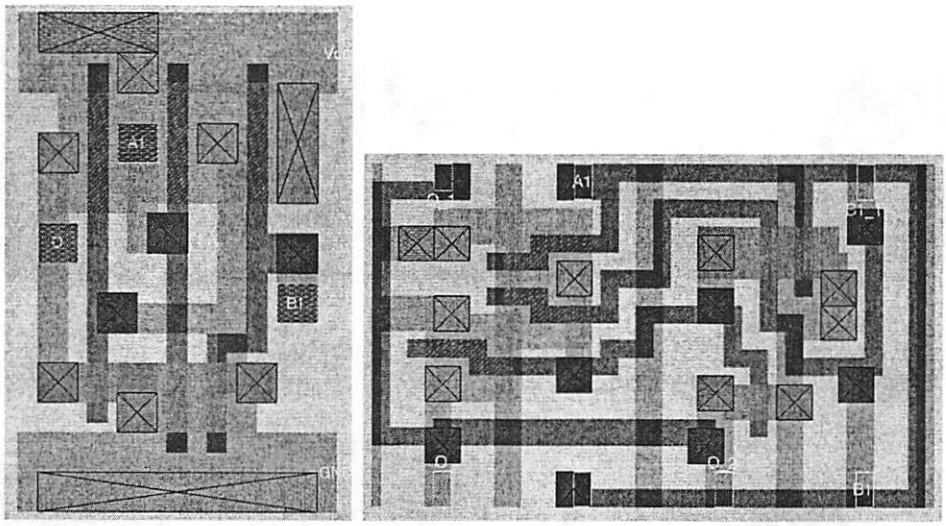
In terms of logic synthesis, technology mapping and placement/routing, the Fabric1 methodology is identical to the standard cell methodology.

We first created a group of 11 static CMOS cells in our Fabric1 methodology, using the MAGIC [HMO84] layout editor. These fabric cells are shown in Figure 4.2. The corresponding transistor level netlists for these cells are shown in Appendix A. These cells followed the DWF conventions described in Chapter 3, and did not use Metal2 at all. The transistor level design of these cells matched that of a control set of 11 standard cells that were part of an existing standard cell library [Bur94] that we had access to. This standard cell library used Metal1 for internal wiring and Metal2 for cell pins. However, our fabric cell library used Metal1 to contact the cell pins, and left *Metal2 free for over-the-cell routes.*

The layout of a sample standard cell and its corresponding fabric cell is shown in Figure 4.1. Note the vertical-running *VDD* and *GND* wires in the layout of the fabric cell. Alternate vertical tracks are reserved for signal wiring.

Once we had a standard cell library and a corresponding fabric cell library, we proceeded to perform placement and routing tests to compare the area utilization of the Fabric1 based design with that of a standard-cell based design.

Our design flow consisted of choosing a *blif* version of a benchmark circuit, per-



(a) Standard Cell (b) Fabric Cell

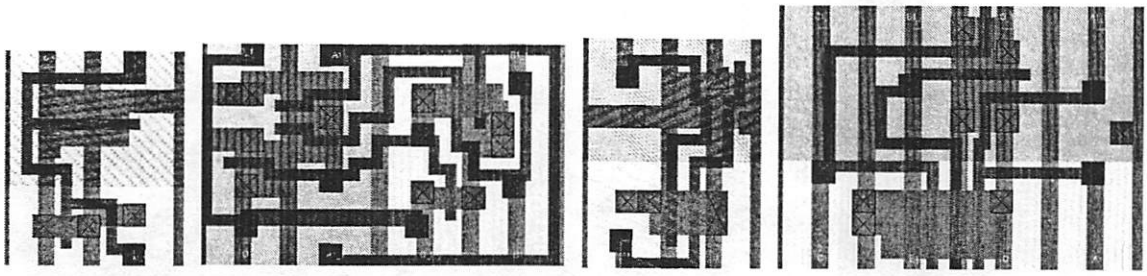
Figure 4.1: Sample standard cell and its corresponding fabric cell

forming logic optimizations on it using *SIS* [SSL+92], and then mapping the circuit using our libraries. After this we use the OCT [Cas91] toolset to place and route this mapped design (for both methods). We use the *wolfe* tool within *OCT* to do the placement and routing. *Wolfe* in turn calls *TimberWolfSC-4.2* [SSV85] to do the placement and global routing, and YACR [RSSV84] to do the detailed routing. For the Fabric1 methodology, we use a wiring pitch which is 2× that used for the standard-cell based methodology. Note that the fabric cell concept does not require the use a channel-based place and route technique. Once the routing was complete, we compared the total area of both methods.

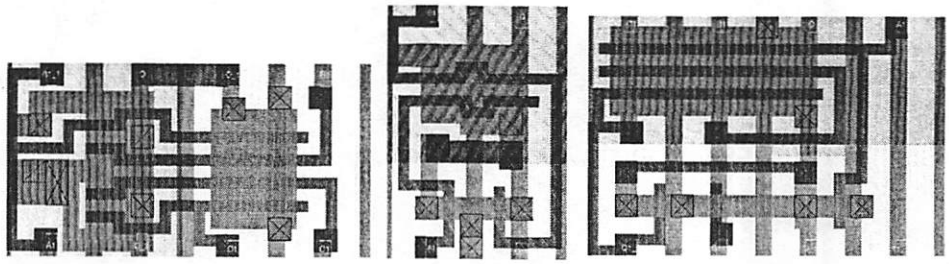
4.2.2 Experimental Results

The routing for the Fabric1 methodology is performed with a wiring pitch that is 2× the metal pitch. In the standard-cell based flow, routing is performed with a wiring pitch equal to the metal pitch.

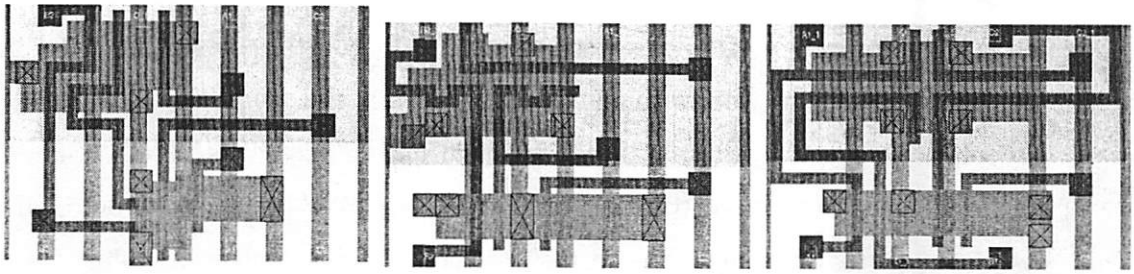
We performed a variety of experiments to compare the area utilization characteris-



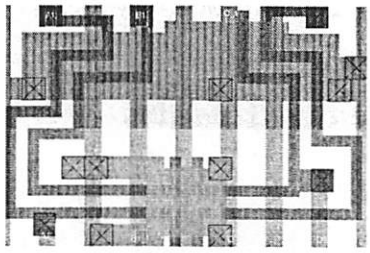
(a) invf101 (b) andf201 (c) nanf201 (d) nanf301



(e) nanf401 (f) norf201 (g) norf301



(h) blf00101 (i) blf00001 (j) aoif2201



(k) oaif2201

Figure 4.2: Fabric Cells

Circuit	Standard Cell		Fabric1		Ratio
	Area	Rows	Area	Rows	
C432	1971.67	8	3380.00	5	1.71
C499	4901.00	10	8431.22	9	1.72
C880	4337.67	10	9126.00	6	2.10
C1355	6346.89	13	8637.78	10	1.36
C1908	6835.11	13	9144.78	8	1.34
C2670	20974.78	13	20035.89	6	0.96
C3540	18101.78	19	33424.44	8	1.85
alu2	4093.56	9	6966.56	8	1.70
apex6	13613.89	13	24035.56	7	1.77
count	1220.56	6	2046.78	6	1.68
decod	375.56	4	431.89	4	1.15
pcle	507.00	4	769.89	5	1.52
rot	13651.44	13	24955.67	7	1.83
pair	36147.22	20	60032.56	11	1.66
AVERAGE					1.60

Table 4.1: Layout Area using 2 Routing Layers (Fabric cells routed using DWF)

tics of our layout scheme with the standard-cell based layout methodology. Intuitively, it would appear that since signals in the Fabric1 methodology can only use every other routing track, there would be a large area penalty. Since the routing pitch for Fabric1 is twice that of the standard-cell based method, it would seem that our designs would utilize 4 times the area of designs routed using the standard-cell methodology. This is because we could expect both the vertical and horizontal dimensions of the design to double (since the wiring pitch is twice the minimum metal pitch).

For all the examples, and in both the Fabric1 and standard cell methodologies, we performed numerous runs and report the best result in Table 4.1. The examples we used were some of the combinational circuits from the MCNC91 benchmark suite, and some additional examples as well. In Table 4.1, the layout area and the corresponding number of cell rows are reported for both styles of layout. The area is reported in square microns. The "Ratio" column represents the ratio of the size of the resulting design using Fabric1 to that using standard cells.

We observe that the average area penalty of using the Fabric1 methodology is 60%. This is much less than the expected 4× area penalty. Also, in case of the C2670

example, the area of the Fabric1 design is smaller than that of the standard-cell based design.

One of the techniques suggested [SK98] to minimize the delay variations due to crosstalk is to up-size transistors in the design. In [SK98] it was found that $W/L = 23$ for transistors was optimal in this sense. Such an increase in device sizes would certainly lead to a large increase in layout area. Our increase of 60% would be comparable, if not smaller than that of the approach of [SK98]. This is justified because the W/L ratio for our standard cells is very low. If it were to be increased to the extent described in [SK98], a significant area penalty would result. The device sizes for our standard cells are shown in A.

Table 4.2 reports the routing area utilization of the fabric cell methodology and the standard cell methodology. The only difference is that the fabric cells are routed without using the DWF. In other words, the routing pitch is the same as the metal pitch. If cross-talk was not a problem and the DWF was not required to be utilized, then this table would be useful in comparing the fabric cell methodology with the standard cell methodology. Table 4.2 is organized exactly like Table 4.1.

Over our benchmark examples, we observe that the average area penalty of the fabric cell methodology over the standard cell methodology in Table 4.2 is about 32%. Also, the variation in the area penalty of individual examples is reduced significantly. This is probably due to the fact that some examples have a large number of wires, resulting in larger area penalties when the DWF is utilized.

The reason for an area penalty of 32% is that our fabric cells do not utilize Metal2 at all in their layout. Therefore the channel router approaches our cells in Metal1 and *does not use the Metal2 area* available over the fabric cells. Therefore the routing resources are only partially utilized. In the case of the standard cells (which utilize Metal2 in their layout), the router approaches the cell in Metal2, and there is therefore a full utilization of the available routing resources. This area overhead would reduce with the use of a better router.

The area overhead for our scheme can be lowered further. Among the reasons for the large overhead are:

Circuit	Standard Cell		Fabric1		Ratio
	Area	Rows	Area	Rows	
C432	1971.67	8	2366.00	6	1.20
C499	4901.00	10	6534.67	9	1.33
C880	4337.67	10	5858.67	5	1.35
C1355	6346.89	13	8656.56	11	1.36
C1908	6835.11	13	9764.44	12	1.43
C2670	20974.78	13	27359.22	9	1.30
C3540	18101.78	19	22063.89	13	1.22
alu2	4093.56	9	5933.78	9	1.45
apex6	13613.89	13	18270.78	8	1.34
count	1220.56	6	1614.89	6	1.32
decod	375.56	4	525.78	5	1.40
pcle	507.00	4	600.89	3	1.19
rot	13651.44	13	18646.33	8	1.37
pair	36147.22	20	45836.56	11	1.27
AVERAGE					1.32

Table 4.2: Layout Area using 2 Routing Layers (Fabric cells routed at min. Pitch)

- First, the natural router for the Fabric1 methodology is an area router. We believe that an area router will reduce the area of Fabric1 based designs. This is because our fabric cells are designed to have a width which is a multiple of 4× the pitch for Metal1 (so that two cells, when placed side by side, will obey the gridding restrictions imposed by the DWF scheme. As a result, fabric cells have *variable heights*. When such cells are placed in a row, the channel router assumes the height of the row to be that of the tallest cell in the row. This results in a significant waste of routable area.

- Secondly, even though our fabric cells do not utilize Metal2, the channel router approaches our cells in Metal1 and *does not use the Metal2 area* available over the fabric cells. In the case of the standard cells, the router approaches the cell in Metal2 and there is therefore a full utilization of the available routing resources. However, in the case of the fabric cell based layout, the Metal2 area above the cells is not utilized at all. This problem would not occur if we had an area router.

- A better placement tool, which allows rotation of a cell along the horizontal axis, could improve our results.

- Using a richer fabric cell library will result in a better technology mapping result, and hence a more efficient design. This is true for both design styles.

In the next section, we overcome the first three restrictions by utilizing a commercially available area router in our flow.

4.3 Design Flow 2

In the design flow of Section 4.2, the detailed routing of the design was performed using a channel routing methodology which used two metal layers for routing. In this section, we remove this restriction, by performing the routing using a commercially available router which performs area routing, and allows the use of up to six metal layers for routing.

4.3.1 Design Methodology

In the design flow of this section, the fabric cells that we utilized are identical to those in Section 4.2. Also, logic synthesis and technology mapping is performed just as in Section 4.2.1.

Our design flow consists of choosing a *blif* version of a benchmark circuit, performing some simple logic optimizations on it using SIS [SSL+92], and then mapping the circuit using our libraries. Placement and routing were performed using the SEDSM-5.1 [Cad99] toolset from CADENCE. Placement was performed using the QPLACE tool within SEDSM-5.1. Routing was performed using the WARP area router, which can use up to 6 metal layers for routing. We compare the total area of the Fabric1 and standard-cell based design styles. In our experiments, we use between 3 and 6 metal layers to route the designs. The standard cells we used for this experiment were optimized to work with the CADENCE tools, and were provided with the SEDSM-5.1 package. For the pins of these cell, there were several possible contact locations

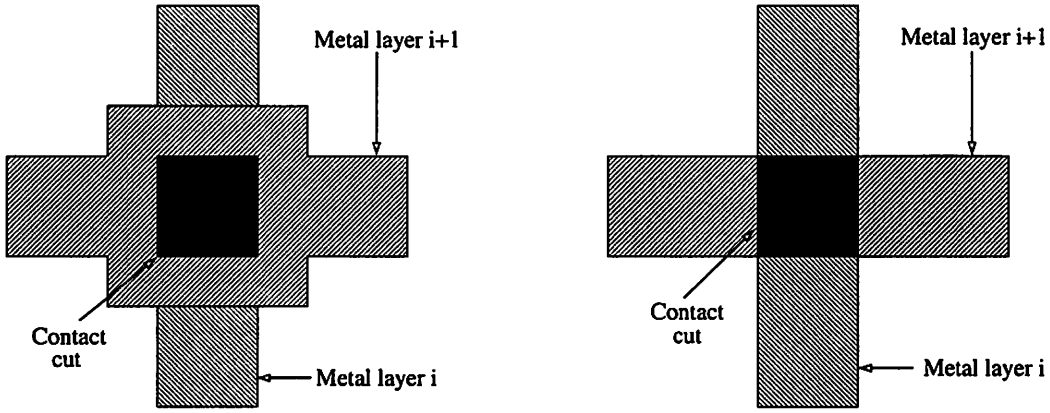


Figure 4.3: Vias with and without Borders

available, to enable easier routing. Also, rows of standard cells could be flipped and abutted so that their power and ground buses were shared, resulting in reduced circuit area.

We also assume that all vias in these processes are *borderless*. Figure 4.3 illustrates a borderless via in comparison to a via with borders. Previous process generations required vias to have borders.

4.3.2 Experimental Results

Tables 4.3 and 4.4 report the results of these experiments. In these tables, the total circuit area (in square microns) is reported for designs implemented in the standard cell methodology and the Fabric1 methodology. In these tables, routing is performed utilizing 3, 4, 5 and 6 metal layers.

In Table 4.3, routing of the Fabric1 design is performed with a wiring pitch that is 2× the metal pitch. In the standard-cell based flow, routing is performed with a wiring pitch equal to the metal pitch. In Table 4.4, routing for both methodologies is performed with a wiring pitch equal to the metal pitch.

From Table 4.3, we note that the overall area penalty of the Fabric1 methodology is about 17%, and has significantly dropped from the 60% penalty obtained when

	3 Routing Layers			4 Routing Layers			5 Routing Layers			6 Routing Layers		
Example	Std Cell	Fabric1	Ratio	Std Cell	Fabric1	Ratio	Std Cell	Fabric1	Ratio	Std Cell	Fabric1	Ratio
C432	1832.90	1840.41	1.004	1832.90	1976.69	1.078	1832.90	1840.41	1.004	1832.90	1840.41	1.004
C499	3501.54	3797.03	1.084	3501.54	3797.03	1.084	3501.54	3797.03	1.084	3501.54	3797.03	1.084
C880	2769.83	3242.16	1.171	2769.83	2896.58	1.046	2769.83	2896.58	1.046	2769.83	2896.58	1.046
C1355	4911.40	6206.29	1.264	4911.40	6206.29	1.264	4911.40	6206.29	1.264	4911.40	6206.29	1.264
C1908	5301.05	6206.29	1.171	4988.15	6206.29	1.244	4911.40	6206.29	1.264	4835.26	6206.29	1.284
C2670	6253.68	7496.10	1.199	6253.68	7228.39	1.156	6253.68	7228.39	1.156	6253.68	7228.39	1.156
C3540	12930.39	16164.59	1.250	11897.60	15770.34	1.326	11252.23	14996.45	1.333	11252.23	14996.45	1.333
alu2	4283.14	5262.05	1.229	4176.72	6206.29	1.486	4247.52	4819.14	1.135	3899.51	4395.69	1.127
apex6	5541.98	6707.61	1.210	5541.98	6707.61	1.210	5541.98	6454.51	1.165	5541.98	6454.51	1.165
count	1237.42	1460.77	1.180	1237.42	1124.93	0.909	1237.42	1232.01	0.996	1237.42	1232.01	0.996
decod	475.90	511.67	1.075	475.90	511.67	1.075	475.90	511.67	1.075	475.90	511.67	1.075
pcle	475.90	584.66	1.229	475.90	584.66	1.229	475.90	584.66	1.229	475.90	584.66	1.229
rot	4911.40	6454.51	1.314	4988.15	5724.43	1.148	4911.40	5724.43	1.166	4911.40	5724.43	1.166
pair	14593.07	17790.23	1.219	12806.65	15770.34	1.231	12317.66	16563.69	1.345	12439.02	15380.95	1.237
			1.186			1.178			1.161			1.155

Table 4.3: Layout Area using 3, 4, 5 and 6 Routing Layers (Fabric cells routed using DWF)

two metal layers were used. This is because the WARP router routes over cells, and therefore the available Metal2 above our fabric cells was utilized[1]. The two layer router of Section 4.2.2 did not utilize this area. Also, the fabric cells have a variable height, and the router of Section 4.2.2 assumed the height of a fabric cell row to be the maximum height among the fabric cells of that row. The router did not route over the row thus created, resulting in wasted area. However, the WARP router does not have this restriction, resulting in more compact routes.

The results of Table 4.4 indicate that when the DWF is not routed in the routing region between fabric cells, we obtain a layout area improvement of about 15%. The area penalty obtained in this table over the standard cell based design is insignificant.

4.4 Chapter Summary

In this chapter, we described the Fabric1 design methodology, and compared it with the standard-cell based methodology. In the Fabric1 methodology, library cells (*fabric cells*) correspond to the standard cells in the standard-cell methodology. The routing area between instances of the library cells utilizes the DWF fabric. The DWF fabric is utilized within the layout of the fabric cells as well.

[1]Our fabric cells do not utilize Metal2 in the layout

	3 Routing Layers			4 Routing Layers			5 Routing Layers			6 Routing Layers		
Example	Std Cell	Fabric1	Ratio	Std Cell	Fabric1	Ratio	Std Cell	Fabric1	Ratio	Std Cell	Fabric1	Ratio
C432	1832.90	1807.11	0.986	1832.90	1807.11	0.986	1832.90	2082.11	1.136	1832.90	1807.11	0.986
C499	3501.54	3560.51	1.017	3501.54	3560.51	1.017	3501.54	3749.11	1.071	3501.54	3749.11	1.071
C880	2769.83	2938.72	1.061	2769.83	2854.76	1.031	2769.83	2854.76	1.031	2769.83	2854.76	1.031
C1355	4911.40	5205.62	1.060	4911.40	4928.04	1.003	4911.40	4928.04	1.003	4911.40	4928.04	1.003
C1908	5301.05	4928.04	0.930	4988.15	4928.04	0.988	4911.40	4928.04	1.003	4835.26	4928.04	1.019
C2670	6253.68	7162.24	1.145	6253.68	7162.24	1.145	6253.68	7162.24	1.145	6253.68	7162.24	1.145
C3540	12930.39	10361.20	0.801	11897.60	10281.96	0.864	11252.23	10203.03	0.907	11252.23	10281.96	0.914
alu2	4283.14	3197.90	0.747	4176.72	3197.90	0.766	4247.52	3197.90	0.753	3899.51	3197.90	0.820
apex6	5541.98	5665.57	1.022	5541.98	5665.57	1.022	5541.98	5783.60	1.044	5541.98	5665.57	1.022
count	1237.42	1098.92	0.888	1237.42	1098.92	0.888	1237.42	1098.92	0.888	1237.42	1098.92	0.888
decod	475.90	547.56	1.151	475.90	547.56	1.151	475.90	547.56	1.151	475.90	547.56	1.151
pcle	475.90	547.56	1.151	475.90	547.56	1.151	475.90	547.56	1.151	475.90	547.56	1.151
rot	4911.40	5433.16	1.106	4988.15	5548.77	1.112	4911.40	5490.81	1.118	4911.40	5665.57	1.154
pair	14593.07	12791.61	0.877	12806.65	12879.99	1.006	12317.66	13057.63	1.060	12439.02	13416.59	1.079
			0.996			1.009			1.033			1.031

Table 4.4: Layout Area using 3, 4, 5 and 6 Routing Layers (Fabric cells routed at min. Pitch)

We introduced two design flows to perform this comparison. In terms of logic synthesis, technology mapping and placement/routing, the Fabric1 methodology uses the same steps as the standard cell methodology in both these design flows.

In the first design flow, public domain placement and routing tools were used. The routing tool utilizes the channel routing methodology, and two metal layers to perform the routing. We show that for our benchmark examples, the Fabric1 methodology utilizes on average 60% more area than the standard-cell based methodology.

In the second design flow, we use commercial placement and routing tools, employing an area routing methodology which allows the use of up to six metal layers for routing. With this flow, an the area penalty of the Fabric1 methodology is about 17%. This is due to a better utilization of the routing region.

In the first design flow, if we were not constrained to use the DWF, then the area penalty for our fabric cell methodology is 32%. This is because the router does not utilize the Metal2 routing area completely. However, in the second design flow, if we were not constrained to use the DWF in the routing region, then the Fabric1 methodology has an area penalty of about 2%.

Chapter 5

Fabric3 - Network of PLA based Design

5.1 Chapter Overview

This chapter introduces a new design methodology to address the cross-talk problem in DSM VLSI design. This new methodology retains the best features of the Fabric1 scheme of Chapter 4, with an extremely low area penalty.

In this approach, we implement the logic netlist in the form of a network of medium sized PLAs. Each PLA is a multi-output structure, laid out in a crosstalk-immune manner. Our layout implementation of PLAs is extremely dense as well.

We utilize two regular layout fabrics in this methodology. The first fabric results from our use of specialized PLAs to implement the design. We show that a PLA implemented in this fabric style is not only cross-talk immune, but also about 2× smaller and faster than a traditional standard-cell based implementation of the same logic. The second fabric is the DWF, and it is utilized in the routing region between individual PLAs. By utilizing these two fabrics, we ensure the cross-talk immunity of the overall design. The other benefits of the DWF, like a high quality power and ground distribution, and uniform and predictably low parasitics are retained in this design style. Our scheme results in a reduction in cross-talk between signal wires of between one and two orders of magnitude, compared to the standard-cell based

approach.

Our synthesis flow involves clustering the design into a multi-level network of PLAs, each of which has a bounded width and height. The number of inputs and outputs of each PLA are flexible as long as the resulting PLA width is bounded. We also perform folding of the resulting PLAs to achieve better logic density.

We have implemented this scheme using two design flows. Our methodology results in circuits that are extremely fast and dense, with an overall timing improvement of about 15% and an overall area penalty of 2.4% compared to standard cells. These timing and area comparisons remain essentially the same, regardless of whether 2, 3, 4, 5 or 6 metal layers are used to perform the routing.

On the other hand, if the network of PLAs is routed at minimum pitch (i.e. the DWF is not used in the routing region), then we obtain an overall timing improvement of about 15% and an overall area improvement of 20% compared to standard cells. Once again, these improvements remain essentially the same, regardless of whether 2, 3, 4, 5 or 6 metal layers are used to perform the routing. This comparison would be relevant if cross-talk was not an issue and therefore the DWF was not required.

With a network of PLAs, there is a more direct relationship between the cost function being optimized during synthesis, and the actual PLA implementation, since there is no intervening technology mapping step. As a result, multi-level logic synthesis is tightly coupled with logic implementation in our design flow.

The remainder of this chapter is organized as follows. Section 5.2 discusses PLAs, and the characteristics of the style of PLA we utilize in this research. Section 5.3 introduces the network of PLAs as an implementation style for DSM VLSI design. Section 5.4 describes the synthesis algorithms we use, which include a PLA folding algorithm (Section 5.4.2) and an algorithm to cluster a logic circuit into a network of PLAs (Section 5.4.3). Sections 5.5 and 5.6 describe two experimental design flows we utilize to verify the utility of the Fabric3 concept. Finally, Section 5.8 summarizes this chapter.

5.2 Programmable Logic Arrays

The Programmable Logic Array (PLA) [GLL80, McC86] is a logic implementation style that has existed for many years. PLAs implement the logic function in a 2-level form. PLAs can implement functions with several inputs and outputs in one flat structure. In a PLA, individual cubes of the logic function are implemented, in what is called an AND plane. Outputs are connected to their relevant cubes in the OR plane. PLAs are simple to implement, but for functions with a large number of cubes, they can be extremely slow.

PLAs have recently experienced a renewed interest as a logic implementation style for high-performance designs. The IBM Gigahertz processor [PAB+98] utilized PLAs to implement control logic. The reasons for this choice were high speed and the ability to quickly implement and modify the logic. We note that the IBM design did not utilize a *network* of PLAs as we are proposing; rather, single PLAs were used.

5.2.1 Introduction

Consider a PLA consisting of n input variables $x_1, x_2, \cdots, x_n$, and m output variables $y_1, y_2, \cdots, y_m$. Let k be the number of rows in the PLA. A *literal* l_i is defined as an input variable or its complement.

Suppose we want to implement a function f represented as a sum of cubes $f = c_1 + c_2 + \cdots + c_k$, where each cube $c_i = l_i^1 \cdot l_i^2 \cdots l_i^{r_i}$. We consider PLAs which are of the *NOR-NOR* form. This means that we actually implement f as

$$\overline{f} = \overline{\sum_{i=1}^{k}(c_i)} = \overline{\sum_{i=1}^{k}(\overline{c_i})} = \overline{\sum_{i=1}^{k}(\overline{l_i^1 + l_i^2 + \cdots + l_i^{r_i}})} \tag{5.1}$$

The PLA output $\overline{f}$ is a logical NOR of a series of expressions, each corresponding to the NOR of the complement of the literals present in the cubes of f. In the PLA, each such expression is implemented by *word lines*, in what is called the *AND plane.* Assume that these word lines run horizontally. Literals of the PLA are implemented by vertical-running *bit-lines.* For each input variable, there are two bit-lines, one for

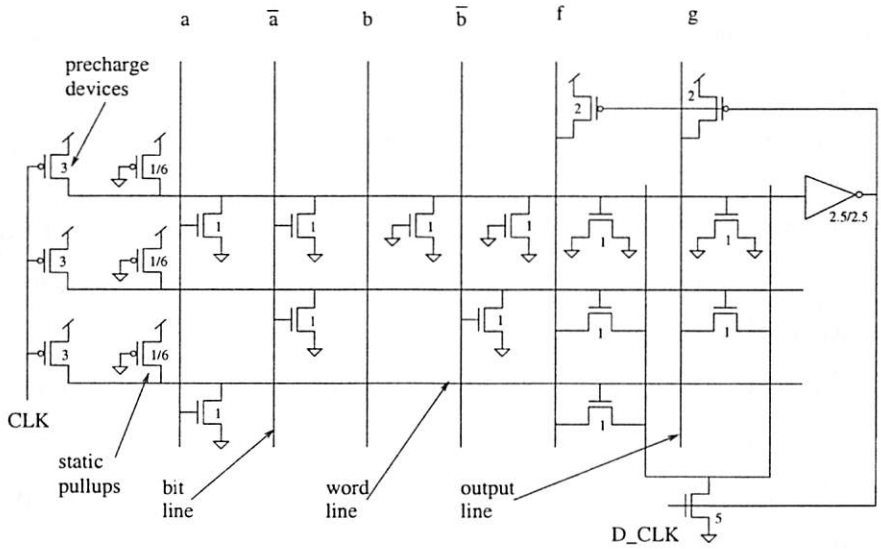


Figure 5.1: Schematic view of the PLA core

each of its literals. The outputs of the PLA are implemented by *output lines*, which also run vertically. This portion of the PLA is called the *OR plane*.

5.2.2 PLAs in DSM VLSI Design

We use a pre-charged NOR-NOR style of PLAs in our design. The schematic view of the PLA core is shown in Figure 5.1. Several observations can be made from this figure:

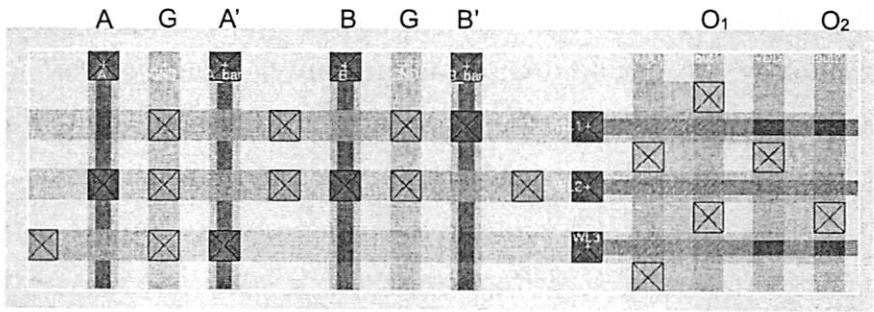


Figure 5.2: Layout of the PLA core

- In a pre-charged NOR-NOR PLA, each word-line of the PLA switches from high to low at the end of any computation, if it switches at all. As a result, there is no delay deterioration effect due to crosstalk with neighboring word-lines.

- However, there is a possibility that two "aggressor" word-lines on either side of a "victim" word-line may switch low during the evaluate phase of the clock, while the victim attempts to stay pre-charged. In this situation, it is possible that the switching of the aggressor word-lines will cause the victim word-line to pull low. We simulated this for the relevant sizes of PLAs, and determined that a long channel (i.e. "weak") static pull-up device suffices to avoid this situation. Hence word lines may be safely routed at minimum pitch.

- In the vertical direction, we shield an input and its complement by a GND wire, which is required by the devices in the AND plane anyway. This can be seen in Figure 5.2.

- One maximally loaded word-line is designed to switch low in the evaluate phase of every clock. It effectively generates a delayed clock, D_CLK, which delays the evaluation of the other word-lines until they have switched to their final values.

- Each bit-line is pre-charged low in the pre-charge phase. The corresponding devices are not shown in in Figure 5.1.

By using a pre-charged NOR-NOR PLA as the layout building block in our methodology, we incur no extra area penalty, either in the horizontal or vertical direction. This is because the horizontal wordlines are routed at minimum pitch. Also, in the vertical direction, both an input and its complement are required to be routed along with a *GND* wire and a separate track to form the wordline contact. By placing the required *GND* wire between the signal wire and its complement, we ensure that the PLA structure is crosstalk immune as well. Figure 5.2 shows the layout of the PLA core. The horizontal word lines are implemented in METAL2. The width of the PLA core is $4 \cdot n + 2 \cdot m$ tracks, since the each input requires 4 vertical tracks,

and each output requires 2. In Figure 5.2, the bitlines labeled A / A' and B / B' represent inputs to the PLA and their complements. The output lines labeled O_1 and O_2 represent two outputs. *GND* tracks are labeled G in this figure. Device sizes in Figure 5.2 correspond to those shown in Figure 5.1.

We implement the input and output drivers outside the footprint of the PLA[1] (i.e. in the routing channel). This gives rise to a much lower area overhead for our PLAs, and also allows us significant flexibility in sizing the drivers. The only effect it has on the routing region between PLAs is the introduction of one via per driver. We were able to complete the layout of all control signals with an additional cost of only 4 horizontal tracks. 4 extra bit-lines are required for the implementation of the pre-charge transistors per PLA. Figure 5.3 shows the relative orientation of pre-charge devices, muxes and drivers in our layout of each PLA. In all the simulations we report in this chapter, these overheads are accounted for. Also, in the electrical simulation of the PLA characteristics, the transistor sizes utilized are as shown in Figure 5.1.

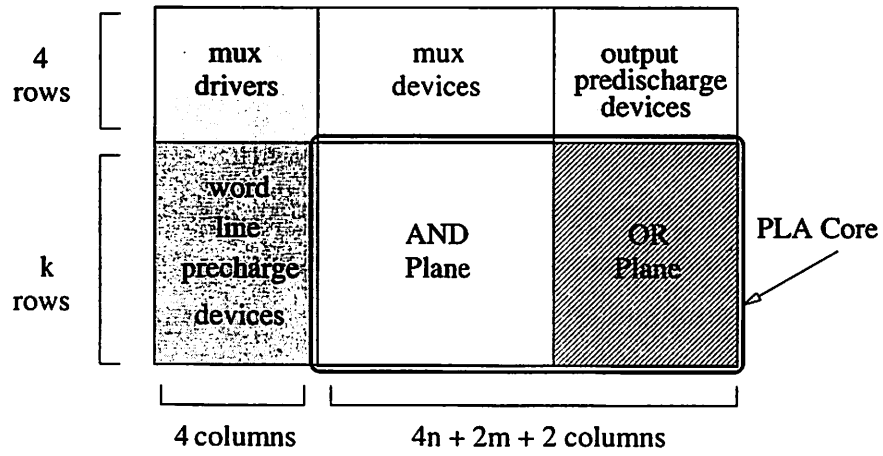


Figure 5.3: Layout Floorplan of PLA

Due to the regularity of the PLA structure, a simple delay formula can be used to estimate the worst-case delay of a PLA. As we will see, this formula is utilized in the synthesis step.

[1]These devices are not shown in Figure 5.1 and Figure 5.2.

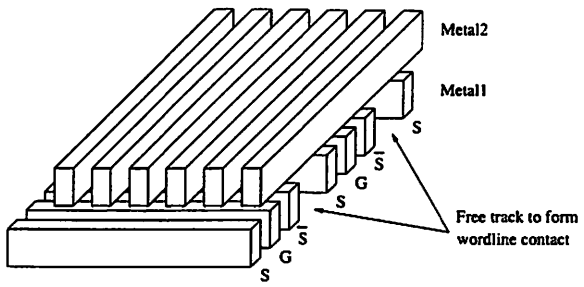


Figure 5.4: Arrangement of conductors in the PLA core

5.2.3 Electrical Characterization

Figure 5.4 shows the pattern of wires occurring within the core of our PLAs. Note that in the PLA core, *VDD* is not utilized. Also, any signal and its complement are always separated by a *GND* wire which is required in the PLA core anyway.

Capacitive parasitics for the wires within our PLA core were extracted using the 3-dimensional parasitic extractor *Space3D* [spa]. The input to *Space3D* is a 3-dimensional circuit layout corresponding to Figure 5.4. Layout dimensions correspond to the 0.1μm strawman process of Table 2.1. The output is the value of the capacitive parasitics between different features of the layout.

The results of these extractions are shown in Table 5.1. In this table $C^1_{i,i}$ refers to the capacitance between two metal conductors on the same level i, which are separated by minimum spacing. $C^2_{i,i}$ refers to the capacitance between two such conductors separated by twice the minimum spacing. $C_{i,0}$ is the capacitance of a conductor to the substrate, and $C_{i,i+1}$ is the capacitance of a conductor on level i to other conductors on level $i+1$.

Layer	$C^1_{i,i}$	$C^2_{i,i}$	$C_{i,0}$	$C_{i,i+1}$
1	47.17	14.57	13.72	15.78
2	48.37	-	0.77	5.96

Table 5.1: 3-Dimensional Parasitics for Figure 5.4 (10^{-18}F per μ)

To compare a *single* PLA implemented in our layout style against the standard-

cell layout style, we took four examples and implemented them in both styles. Delay and power results were obtained utilizing SPICE [Nag95]. The area comparison was done using actual layout for both styles.

For the standard-cell style of layout, we performed technology-independent optimizations in SIS [SSL+92], after which we mapped the circuit using our library of 11 standard cells. We use the *wolfe* tool within *OCT* [Cas91] to do the placement and routing. *Wolfe* in turn calls *TimberWolfSC-4.2* [SSV85] to do the placement and global routing, and YACR [RSSV84] to do the detailed routing.

For the PLA layout style, we flatten the examples, and then generate the *MAGIC* [HMO84] layout for the resulting PLA using a *perl* [WS92] script. To compute the delay, we simulate a maximally loaded word-line, which is discharged by a single pull-down device. The output line is pulled down by a single output pull-down device. The loads on the line are n diffusion loads in the AND plane, and m gate loads in the OR plane. Parasitics from Table 5.1 were utilized to model the interconnect within a PLA.

				PLA implementation			Standard Cell			Ratios		
Example	n	m	k	D	A	P	D	A	P	D	A	P
cmb	16	4	15	160.3	53.3k	5.32	300	159.8k	6.15	0.534	0.334	0.864
cu	14	11	19	189.1	69.5k	4.84	420	186.5k	4.24	0.450	0.373	1.140
x2	10	7	17	164.8	45.3k	4.23	290	136.8k	1.82	0.568	0.331	2.324
z4ml	7	4	59	200.5	95.2k	10.28	575	118.3k	3.17	0.349	0.805	3.243

Table 5.2: Comparison of Standard-Cell and PLA implementation styles

The results of this comparison are listed in Table 5.2. For each layout style, D refers to the delay in picoseconds, A refers to the layout area of the resulting implementation in square grids, and P refers to the power consumption. For the Standard cell layout style, D and P values are the maximum values obtained after simulating about 20 input vectors. Also, we don't account for wire capacitances in the Standard cell implementation, which would only increase its delay and power. In the case of the PLA layout style, however, the D and P are worst-case values. Despite this, the PLA layout style shows impressive improvements over the Standard cell layout

style in area and delay. The PLA layout utilizes between 0.33 and 0.81 times the area of the Standard cell layout. The average area utilization of the PLAs is 0.46 times that of the Standard cell layout style, which is an impressive reduction. The delay value for the PLA is on average 0.48 times that of the Standard-cell implementation.

The power consumption of the PLA is usually larger than that of the Standard cell implementation, mainly because the bit-line capacitances are charged and discharged on every cycle. PLAs with large values of k exhibit a much higher power consumption. For this reason, we use a value of $k \leq 25$ while clustering a circuit into a network of PLAs. This power consumption can also be curbed by gating the clocks of PLAs which are not utilized during a given computation. Also, power and delay can be traded off in our PLA design style. Finally alternative fabrics utilizing unate PLAs can be used to reduce the power consumption.

To estimate the effect of crosstalk between literals of neighboring variables in the PLA, we simulated a PLA with $k = 40$. Let l_i be a literal of variable x_i, and l_{i+1} be a literal of variable x_{i+1}. Assume l_i and l_{i+1} are separated by a blank track. In this situation, there is a 1:1.0156 delay variation for l_i, depending on whether l_{i+1} switches in the opposite or the same direction. This delay variation is small enough to be disregarded.

5.2.4 Discussion

The reason why PLAs result in very favorable area and delay characteristics compared to a standard cell layout are the following:

- First, PLAs implement their logic function in 2-level form, which results in superior delay characteristics as long as k is bounded. On the other hand, in a standard cell implementation, considerable delay is incurred in traversing the different levels (i.e. gates) of the design.

- In DSM processes, it is often stated that an increasing fraction of a signal's delay is attributable to wiring. In the PLA implementation scheme, local wiring is collapsed into a compact 2-level core, which is naturally crosstalk-immune.

Hence wiring delays are reduced.

- Devices in the PLA core are minimum-sized, giving rise to extremely compact layouts. Such is not the case for Standard cell layouts.
- In our PLA core, NMOS devices are used exclusively. As a result, devices can be placed extremely close together. However, in a Standard cell layout, both PMOS and NMOS devices are present in each cell, and the PMOS-to-NMOS diffusion spacing requirement results in a loss of layout density.

The fact that the IBM Gigahertz processor[PAB+98] utilizes two-level PLAs to implement control logic is further evidence that PLAs are an effective logic implementation style for high-performance designs.

5.3 Networks of Programmable Logic Arrays

Having discussed the characteristics of a single PLA, we now discuss how a *network* of PLAs is constructed.

Since the PLAs in our design are pre-charged, we need to ensure that the inputs to any PLA settle before its evaluation begins. A network of PLAs is *correct* iff each PLA in this network satisfies this constraint.

Definition 5.1 *The* PLA dependency graph *G(V, E) of a network of PLAs is a directed graph such that*
$V = \{v_1, v_2, \cdots, v_r\}$, *where each vertex* v_i *corresponds to a unique PLA in the network.* $(v_i, v_j) \in E$ iff *an output of PLA* p_i *is an input to PLA* p_j.

It is easy to see that if the PLA dependency graph has a cycle, then the corresponding network of PLAs is not correct. For a correct network, we need to ensure that the PLA dependency graph is acyclic, and also that the evaluation of PLA p begins only after the evaluation of the slowest PLA q, such that $(q, p) \in E$. This suggests a *self-timed* [Rab96] design style. For this reason, each PLA q generates a *completion* signal, which gates the evaluation clock of the appropriate PLA p. Given the regularity of

the PLA structure, the worst case delay of each PLA is easily known, and corresponds to the delay of a single word-line pulldown device discharging a (maximally loaded) word line and a single output device discharging the output line. This completion signal is generated with an overhead of one additional word line and one additional output line in each PLA. Additional timing margin is obtained by downsizing the output driver of the completion signal.

When PLAs are placed, local breaks occur in the power and ground gridding structure of Metal1 and Metal2. This condition was simulated, and determined to cause a negligible change in the power and ground resistance, because the contribution of upper metal layers to the resistance of the power and ground network far outweighs that of the lower metal layers.

5.4 Synthesis Algorithms for the Network of PLAs Methodology

5.4.1 Overview

In this section, we introduce the synthesis algorithms which are used in the network of PLAs design flow. In Section 5.4.2, we discuss our folding algorithm. This algorithm folds the inputs of individual PLAs in order to achieve a higher logic density. This folding procedure helps reduce the final number of PLAs in the network by between 20 and 50%. Section 5.4.3 discusses our clustering algorithm, which is used to cluster a multi-level netlist into a multi-level network of PLAs.

5.4.2 Folding Algorithm

We only fold the inputs of our PLAs. This is because even after a pair of inputs are folded, we can access each input from both the top and bottom directions of the PLA, since each input utilizes two signal tracks in our PLAs. This enables the router to access any input pin from both the top and bottom sides of the PLA, resulting in a smaller circuit area.

If pairs of outputs were folded as well, the router would not be able to access any one output on both the top and bottom sides of the PLA. This would result in a larger circuit area utilization. Hence output folding is not performed.

Algorithm 1: Folding the inputs of a PLA

```
M = construct_matrix(P)
G = initialize_fold_graph(P)
lbl:C = find_fold_candidates(M)
while (c = find_best_candidate(C)) != NIL do
  apply_fold(c, P)
  update_fold_graph(G, c)
  goto lbl
end while
```

Algorithm 1 describes our folding strategy. We first construct a 0-1 matrix M corresponding to the AND plane of the PLA P. "1" or "0" entries of the cubes are mapped to "1" entries in this matrix while "-" entries are mapped to "0" entries in the matrix. We then initialize our fold graph G. This is a graph with vertices corresponding to rows of the PLA. There is an edge between vertices p and q in G if row p must be above row q in the PLA as a result of a fold. Next we find the set of pairs of inputs whose columns in the matrix have a null intersection. This set forms the set of potential fold candidates, C.

The *find_best_candidate* routine returns the best candidate among the list of potential fold candidates, such that the fold implied by this candidate does not result in a cyclic fold graph G^2. The ranking of candidates is performed using the following figure of merit η.

$$\eta = \alpha \cdot (\#cubes - size(i_1) - size(i_2)) + \beta \cdot (|size(i_1) - size(i_2)|) \qquad (5.2)$$

Here *size(j)* is the number of non-zero matrix column entries corresponding to the input j. Also, $\#cubes$ is the total number of cubes in PLA P, while α and β are constants. The first term of η favors folds that add fewer edges to the fold graph. The second term favors folds whose inputs have a large imbalance in their number of non-zero column entries. This allows subsequent folds to proceed without hindrance.

[2] A cyclic fold graph means that some vertex p_1 (i.e. row p_1) must be above another vertex p_2 (i.e. row p_2), while p_2 must be above p_1 as well. This condition is not simultaneously satisfiable.

If *find_best_candidate* returns a candidate, then we apply the fold, update the fold graph to reflect this fold[3], and again attempt to find more folds. The process ends when no more legal folds can be found.

5.4.2.1 Results

We tested our algorithm on several 2-level circuits, with varying values of α and β. The results are shown in Table 5.3.

Circuit			Number of Folds			
Name	#inputs	#outputs	$\alpha = 1, \beta = 0$	$\alpha = 0, \beta = 1$	$\alpha = 1, \beta = 1$	$\alpha = 1, \beta = 4$
cm42a	4	10	2	2	2	2
cm85a	11	3	1	2	2	2
cm162a	14	5	4	6	5	6
cm163a	16	5	4	7	7	7
cmb	16	4	4	4	4	4
cu	14	11	3	3	2	3
x2	10	7	2	4	3	4
z4ml	7	4	0	0	0	0
TOTAL			20	28	25	28

Table 5.3: Folding Algorithm Results

The results of Table 5.5 suggest that a large β value is helpful in finding more folds. While performing clustering, we use this algorithm with $\alpha = 1$ and $\beta = 4$.

5.4.3 Clustering Algorithm

Problem Definition: Given an arbitrary logic circuit C, cluster C into a network N of PLAs, subject to :

- the network N is correct.
- each PLA has a height no larger than a specified maximum, H.
- each PLA has a width no larger than a specified maximum, W.

Algorithm 2 outlines our clustering strategy. We begin by performing some technology independent optimizations on C . Next, we decompose C into a network C^* of

[3]If columns c_1 and c_2 are folded, then the fold graph is updated by inserting edges between all vertices p_1 and p_2, such that $M[p_1, c_1] = M[p_2, c_2] = 1$.

nodes with at most p inputs. Now C^* is sorted in depth-first manner. The resulting array of nodes is sorted in *topological*[4] order, and placed into an array L.

Now we greedily construct the logic in each PLA, by successively grouping nodes from L such that the resulting PLA implementation of the grouped nodes N^* does not violate the constraints of PLA width and height. This check is performed in the *check_PLA* routine, which first flattens N^* into a two-level form, P. It then calls *espresso* [BHMSV84] on the result to minimize the number of cubes in P. Next, *check_PLA* calls the PLA folding routine of Section 5.4.2 which attempts to fold the inputs of P so as to implement a more complex PLA in the same area. Finally *check_PLA* ensures that the final PLA, after folding and simplification using *espresso*, satisfies the maximum width and height constraints respectively[5]. If so, we attempt to include another node into N^*, otherwise we append the last PLA satisfying the height and width constraints to the result.

Folding the PLAs resulted in a decrease of between 20% and 50% in the total number of PLAs required for a network.

The *get_next_element* routine returns nodes in the fanout of the nodes in N^* (in an attempt to reduce the wiring between PLAs), provided that the inclusion of such a node into N^* would not result in a cyclic PLA dependency graph. If such nodes are not available, the first un-mapped node from L is returned.

After clustering is performed, we invoke a procedure called *last_gasp*. This is a final effort in reducing the wiring between PLAs. This procedure attempts to move individual nodes in L to a different PLA than their currently assigned PLA. If a wiring gain is realized by such a move, the move is made. If no more nodes can be gainfully moved, or if a specified number of iterations have been made through L, the procedure returns. We note the following about this procedure:

- It is possible that a node n in L is the only node in some PLA p, and if n can

[4]Primary inputs are assigned a level 0, and other nodes are assigned a level which is one larger than the maximum level of all their fanins

[5]Each input in our PLA requires 4 vertical tracks, and each output requires 2 vertical tracks. Hence the width constraint is satisfied for a PLA with n inputs, m outputs and p cubes if $4 \cdot n + 2 \cdot m \leq W$, where W is the maximum allowable PLA width. The height constraint is satisfied if $p \leq H$, where H is the maximum allowable PLA height.

be gainfully moved to another PLA, then PLA p can be removed. We came across a few instances where a PLA was removed in this manner.

- *last_gasp* returns when no node can be gainfully moved. At this point, it is still possible that more than one node can simultaneously be moved to realize a gain in wiring. However, *last_gasp* does not check this.

Algorithm 2: Clustering a Circuit into a Network of PLAs

```
C = simplify_network(C)
C* = decompose_network(C, p)
L = dfs_and_levelize_nodes(C*)
N* = 0
RESULT = 0
while get_next_element(L) != NIL do
  N* = N* ∪ get_next_element(L)
  P = make_PLA(N*)
  if check_PLA(P, W, H) then
    continue
  else
    Q = remove_last_element(N*)
    RESULT = RESULT ∪ N*
    N* = Q
  end if
end while
last_gasp(RESULT)
```

Note that this algorithm does not attempt to ensure that the maximum delay between any PI-PO pair is bounded. As a result, it sometimes returns a network with delays larger than the corresponding standard cell implementation. However, on average, the delay of the network it returns is much better than a standard cell implementation (see Sections 5.5.2 and 5.6.2).

We implemented our algorithm in SIS, and performed extensive benchmarking of the PLA network clustering code. We found that a good choice of parameters was $p = 5$, W = 50 to 70, and H = 15 to 25. Increasing H beyond 30 did not usually result in a reduction in the total number of PLAs generated.

We verified functional correctness of the resulting network of PLAs, at the end of the clustering step.

5.4.3.1 Results

In this section, we outline the effectiveness of the *last_gasp* strategy. We list the number of wires in a network of PLAs generated by Algorithm 2, both with and without the *last_gasp* procedure. These results are listed in Table 5.4. The column labeled "Wires before" lists the number of wires in the network of PLAs if *last_gasp* is not used. The column labeled "Wires after" reports the number of wires if *last_gasp* is used. The percentage wiring improvement due to *last_gasp* is reported in the column labeled "%Improvement". The column labeled "PLAs removed" reports the number of PLAs removed by *last_gasp*. In all the examples in Table 5.4, the maximum allowable PLA height was 25, and the maximum PLA width was 60.

Example	Wires before	Wires after	PLAs removed	%Improvement
decod	21	21	0	0
pcle	39	39	0	0
count	74	74	0	0
C432	177	172	0	2.82
C499	256	223	0	12.89
alu2	177	120	2	32.20
rot	662	514	0	22.36
apex6	584	488	0	16.44
pair	1592	1506	0	5.40
C880	375	304	0	18.93
C1355	380	362	0	4.74
C1908	543	441	0	18.78
C2670	799	698	1	12.64
C3540	1465	1366	0	6.76
AVERAGE				9.62

Table 5.4: Wiring improvement with *last_gasp*

From the results of Table 5.4, we note that there is a significant reduction in wiring as a result of using the *last_gasp* procedure. On average, wiring is reduced by 9.62%. In examples *alu2* and *C2670*, *last_gasp* was able to remove at least one PLA from the network.

5.4.4 Other Clustering Algorithms

We experimented with another algorithm for clustering, in which we first perform a min-cut bi-partitioning using hMETIS [KAKS97, KK98]. If either of the two halves satisfies the maximum width and height constraints, we stop, otherwise we recurse on this procedure. We found that this resulted in a large increase in the number of resulting PLAs, though it usually reduced the number of wires.

We experimented with a variation of the min-cut bi-partitioning idea, in which we recursively performed bi-partitioning, but called Algorithm 2 when the halves of the cut got below a certain size. Although this resulted in a reduction in the number of PLAs compared to the regular bi-partitioning scheme, it did not reduce the number of PLAs compared to Algorithm 2. Also it did not reduce the number of wires significantly. In many examples, the number of wires remained unchanged.

In both the bi-partitioning schemes mentioned above, the min-cut does not guarantee that the PLA network will be correct. It is possible that a path from PI to PO is cut twice by hMETIS. In this case, the PLA network would not be correct. We contacted the authors of hMETIS and were informed that there was no way to avoid this situation. The work-around that we used was to perform a series of cuts, and choose one only if it satisfied the PLA correctness constraint.

5.5 Design Flow 1

In this section, we compare the area and timing characteristics of the Fabric3 design methodology with that of a standard-cell based design methodology. In both cases, the detailed routing of the design is performed using a channel routing methodology, using two metal layers for routing.

5.5.1 Design Methodology

The logic netlist is first clustered into a network of PLAs using Algorithm 2. The synthesized network of PLAs is placed using a simulated annealing-based FPGA placement tool called *VPR* [BR97]. Since the PLAs in our design have approximately

the same size, the problem of placing PLAs is similar to the FPGA placement problem. Hence *VPR* is a good choice.

The placed result is routed using *wolfe*. Global routing is performed on the design using TimberWolfSC-4.0 [SSV85]. Finally detailed routing is performed using YACR [RSSV84], a 2-layer channel router.

For the standard cell based methodology, the design flow is identical to the one described in Section 4.2.1.

5.5.2 Experimental Results

The area and delay characteristics of our network of PLAs based design methodology are shown in Table 5.5. The routing of the network of PLAs is performed with a wiring pitch that is 2× the pitch for Metal1 and Metal2. In the standard-cell based flow, routing is performed with a wiring pitch equal to the Metal1 or Metal2 pitch. Table 5.5 describes the results for a series of benchmarks, comparing the area of a standard cell implementation (column 2), and the Fabric3 approach (column 3). All areas are in units of square microns. Column 4 reports the ratio of column 3 to column 2.

Also reported is the total delay (in picoseconds) of the standard cell implementation (column 5), and that of the Fabric3 approach (column 6). The ratio of these two delays is shown in column 7.

Delays for the standard cell implementation were obtained by running the *exact timing analysis* [MSBSV93] technique on the mapped netlist. For the Fabric3 approach, we computed the worst case delays of each PLA in the network using Spice [Nag95], as described in Section 5.2.3. Then we found the worst case delay path from any primary input to any primary output in the PLA network by traversing the network of PLAs in DFS order.

In the results reported in Table 5.5, clustering into a network of PLAs was done with W = 50 to 70 in steps of 10, and H = 15 to 25 in steps of 5. The best area among these runs is reported in Table 5.5.

We note that over all these examples, the area overhead of the Fabric3 method

was a mere 2.4%. This is in spite of the fact that the DWF is used in the routing area between PLAs, but not used in the standard cell case. Also, the delay of the Fabric3 approach is approximately 15% better than that of a standard cell implementation. Some examples result in much higher delays for the network of PLAs style. This is attributed to the fact that our clustering routine does not attempt to control the delay of the network, but rather attempts to reduce wiring between PLAs.

	Area			Timing		
Example	Std Cell	Ntk of PLA	Ratio	Std Cell	Ntk of PLA	Ratio
C432	1971.67	2628.89	1.338	2326.4	2237.1	0.962
C499	4901.00	3868.22	0.789	1861.5	1575.8	0.847
C880	4337.67	5295.33	1.221	2007.5	1587.3	0.791
C1355	6346.89	7248.22	1.141	2688.4	1890.7	0.703
C1908	6835.11	10271.44	1.502	2203.9	3235.6	1.468
C2670	20974.78	15604.33	0.744	2388.3	2244.6	0.940
C3540	18101.78	39677.44	2.191	3324.3	4385.5	1.319
alu2	4093.56	2929.33	0.717	2528.3	1758.2	0.695
apex6	13613.89	8656.56	0.636	1332.4	1011.2	0.759
count	1220.56	694.78	0.564	2029.7	568.0	0.280
decod	375.56	225.33	0.567	330.4	184.3	0.558
pcle	507.00	469.44	0.925	1285.7	334.5	0.260
rot	13651.44	11454.44	0.838	2256.1	2110.8	0.936
pair	36147.22	41761.78	1.166	1951.9	2660.2	1.363
AVERAGE			1.024			0.848

Table 5.5: Layout Area and Timing (PLAs routed using DWF)

Table 5.6 reports the area and timing results for the condition in which the routing of the network of PLAs is performed with a wiring pitch equal to the Metal1 and Metal2 pitch. The organization of the table is identical to that of Table 5.5. In case cross-talk was not a consideration and wiring did not need to be performed in the DWF, then this would be the area and timing comparison of the Fabric3 methodology and the standard cell based methodology.

In Table 5.6, clustering into a network of PLAs was done with W = 50 to 70 in steps of 10, and H = 15 to 25 in steps of 5, and the best area among these runs is reported.

	Area			Timing		
Example	Std Cell	Ntk of PLA	Ratio	Std Cell	Ntk of PLA	Ratio
C432	1971.67	1765.11	0.895	2326.4	1791.4	0.770
C499	4901.00	2873.00	0.586	1861.5	1520.1	0.817
C880	4337.67	4037.22	0.931	2007.5	1761.4	0.877
C1355	6346.89	6853.89	1.080	2688.4	1919.1	0.714
C1908	6835.11	7548.67	1.104	2203.9	2785.3	1.264
C2670	20974.78	11923.89	0.568	2388.3	2361.7	0.989
C3540	18101.78	30326.11	1.675	3324.3	4529.8	1.363
alu2	4093.56	2234.56	0.546	2528.3	1235.3	0.489
apex6	13613.89	5933.78	0.436	1332.4	1236.9	0.928
count	1220.56	563.33	0.462	2029.7	568.0	0.280
decod	375.56	169.00	0.450	330.4	184.3	0.558
pcle	507.00	300.44	0.593	1285.7	319.0	0.248
rot	13651.44	8506.33	0.623	2256.1	2093.2	0.928
pair	36147.22	24918.11	0.689	1951.9	2900.2	1.486
AVERAGE			0.760			0.837

Table 5.6: Layout Area and Timing (PLAs routed at min. Pitch)

In Table 5.6 we notice that the average area utilization of the network of PLAs is significantly reduced from the standard-cell based implementation. This is as expected, since the network of PLAs methodology implements logic in a very dense fashion, as detailed in Section 5.2.3. Also, the timing of the network of PLA implementation is on average about 16% better than that of the standard cell implementation. This improvement is very similar to that reported Table 5.5.

5.6 Design Flow 2

In the design flow of Section 5.5, the detailed routing of the design was performed using a channel routing methodology which used two metal layers for routing. In this section, this restriction is removed. Routing is achieved using a commercially available router which performs area routing, and allows the use of up to six metal layers for routing.

5.6.1 Design Methodology

This design flow consists of choosing a *blif* version of a benchmark circuit, and clustering it into a network of PLAs using Algorithm 2. The resulting network of PLAs is now placed and routed using the SEDSM-5.1 [Cad99] toolset from CADENCE. Placement was performed using the QPLACE tool within SEDSM-5.1. Routing was performed using the WARP area router, which can use up to 6 metal layers for routing. We compare the total area of the Fabric3 and standard-cell based design styles. In our experiments, we use between 3 and 6 metal layers to route the designs. We also compare the timing of the Fabric3 based design with the standard-cell based design, using the technique described in Section 5.5.2.

The standard cells used for this experiment were optimized to work with the CADENCE tools, and were provided with the SEDSM-5.1 package. Each pin of these cells had several possible contact locations available, to enable easier routing. Also, rows of standard cells could be flipped and abutted so that their power and ground buses were shared, resulting in reduced circuit area.

For this flow, we assume that all vias in these processes are *borderless*. Figure 4.3 illustrates a borderless via in comparison to a via with borders.

5.6.2 Experimental Results

	3 Routing Layers			4 Routing Layers			5 Routing Layers			6 Routing Layers		
Example	Std Cell	Ntk of PLA	Ratio	Std Cell	Ntk of PLA	Ratio	Std Cell	Ntk of PLA	Ratio	Std Cell	Ntk of PLA	Ratio
C432	1832.90	1603.20	0.875	1832.90	1603.20	0.875	1832.90	1603.20	0.875	1832.90	1603.20	0.875
C499	3501.54	2440.36	0.697	3501.54	2046.66	0.585	3501.54	2046.66	0.585	3501.54	2046.66	0.585
C880	2769.83	3576.04	1.291	2769.83	3576.04	1.291	2769.83	3331.60	1.203	2769.83	3212.62	1.160
C1355	4911.40	5843.07	1.190	4911.40	4640.33	0.945	4911.40	4640.33	0.945	4911.40	4640.33	0.945
C1908	5301.05	7184.26	1.355	4988.15	6165.39	1.236	4911.40	6329.79	1.289	4835.26	5843.07	1.208
C2670	6253.68	9658.96	1.545	6253.68	8471.36	1.355	6253.68	8092.80	1.294	6253.68	8092.80	1.294
C3540	12930.39	27172.23	2.101	11897.60	22272.58	1.872	11252.23	20449.00	1.817	11252.23	20747.52	1.844
alu2	4283.14	2141.84	0.500	4176.72	2141.84	0.513	4247.52	2141.84	0.504	3899.51	2141.84	0.549
apex6	5541.98	4783.11	0.863	5541.98	4783.11	0.863	5541.98	4783.11	0.863	5541.98	4783.11	0.863
count	1237.42	597.31	0.483	1237.42	597.31	0.483	1237.42	597.31	0.483	1237.42	597.31	0.483
decod	475.90	227.41	0.478	475.90	227.41	0.478	475.90	227.41	0.478	475.90	227.41	0.478
pcle	475.90	499.97	1.051	475.90	499.97	1.051	475.90	499.97	1.051	475.90	499.97	1.051
rot	4911.40	5685.16	1.158	4988.15	5529.41	1.109	4911.40	5375.82	1.095	4911.40	5375.82	1.095
pair	14593.07	22584.08	1.548	12806.65	20449.00	1.597	12317.66	21351.05	1.733	12439.02	20449.00	1.644
			1.081			1.018			1.015			1.005

Table 5.7: Layout Area using 3, 4, 5 and 6 Routing Layers (PLAs routed using DWF)

	3 Routing Layers			4 Routing Layers			5 Routing Layers			6 Routing Layers		
Example	Std Cell	Ntk of PLA	Ratio	Std Cell	Ntk of PLA	Ratio	Std Cell	Ntk of PLA	Ratio	Std Cell	Ntk of PLA	Ratio
C432	1832.90	1460.77	0.797	1832.90	1460.77	0.797	1832.90	1460.77	0.797	1832.90	1460.77	0.797
C499	3501.54	1930.72	0.551	3501.54	1930.72	0.551	3501.54	1930.72	0.551	3501.54	1930.72	0.551
C880	2769.83	2785.73	1.006	2769.83	2785.73	1.006	2769.83	2785.73	1.006	2769.83	2785.73	1.006
C1355	4911.40	3733.21	0.760	4911.40	3733.21	0.760	4911.40	3733.21	0.760	4911.40	3733.21	0.760
C1908	5301.05	4604.98	0.869	4988.15	4124.21	0.827	4911.40	4327.01	0.881	4835.26	4604.98	0.952
C2670	6253.68	6206.29	0.992	6253.68	6206.29	0.992	6253.68	5803.39	0.928	6253.68	5803.39	0.928
C3540	12930.39	16164.58	1.250	11897.60	15901.21	1.337	11252.23	16430.11	1.460	11252.23	15901.21	1.413
alu2	4283.14	1666.27	0.389	4176.72	1460.77	0.350	4247.52	1666.27	0.392	3899.51	1666.27	0.427
apex6	5541.98	3861.38	0.697	5541.98	4327.01	0.781	5541.98	3861.38	0.697	5541.98	3861.38	0.697
count	1237.42	584.67	0.472	1237.42	584.67	0.472	1237.42	584.67	0.472	1237.42	584.67	0.472
decod	475.90	219.63	0.462	475.90	219.63	0.462	475.90	219.63	0.462	475.90	219.63	0.462
pcle	475.90	360.24	0.757	475.90	360.24	0.757	475.90	360.24	0.757	475.90	360.24	0.757
rot	4911.40	4124.21	0.840	4988.15	3733.21	0.748	4911.40	3733.21	0.760	4911.40	3733.21	0.760
pair	14593.07	17103.41	1.172	12806.65	16430.11	1.283	12317.66	15901.21	1.291	12439.02	17103.41	1.375
			0.787			0.794			0.801			0.811

Table 5.8: Layout Area using 3, 4, 5 and 6 Routing Layers (PLAs routed at min. Pitch)

Tables 5.7 and 5.8 report the results of these experiments. In these tables, the total circuit area (in square microns) is reported for designs implemented in the standard cell methodology and the network of PLAs methodology. In these tables, routing is performed utilizing 3, 4, 5 and 6 metal layers.

In Table 5.7, routing of the network of PLAs is performed with a wiring pitch that is 2× the metal pitch. In the standard-cell based flow, routing is performed with a wiring pitch equal to the metal pitch. In Table 5.8, routing for both methodologies is performed with a wiring pitch equal to the metal pitch.

In the results reported in Tables 5.7 and 5.8, clustering into a network of PLAs was done with W = 50 and H = 15. Had we chosen from among a larger set of values of W and H (as in Section 5.5.2), we would have obtained yet better results.

From Table 5.7, we note that the overall area penalty of the network of PLAs methodology is extremely small, and essentially matches the results obtained in Section 5.5.2. The average area penalty drops slightly when more routing layers are made available to complete the routing of the design. With four or more metal layers available to perform the routing, the overhead of the network of PLA methodology is insignificant.

The area improvements in Table 5.8 are very similar to those obtained in Table 5.6. This suggests that even if cross-talk was not a problem in modern designs, the Fabric3 approach has marked advantages over the standard-cell based approach.

Example	Std Cell	Ntk of PLA	Ratio
C432	2326.4	1791.4	0.770
C499	1861.5	1520.1	0.817
C880	2007.5	2089.7	1.041
C1355	2688.4	1825.4	0.679
C1908	2203.9	3397.1	1.541
C2670	2388.3	2367.0	0.991
C3540	3324.3	4529.8	1.363
alu2	2528.3	1674.3	0.662
apex6	1332.4	1380.3	1.036
count	2029.7	568.0	0.280
decod	330.4	184.3	0.558
pcle	1285.7	282.9	0.220
rot	2256.1	1737.0	0.770
pair	1951.9	2900.2	1.486
AVERAGE			0.872

Table 5.9: Timing Characteristics of Circuits derived in Flow2

Table 5.9 reports the timing comparison for circuits implemented in the network of PLAs style, compared to the standard cell based methodology. Over the 14 benchmark examples, a timing improvement of about 13% is obtained by using a network of PLAs design methodology. These results are substantially similar to those obtained in Section 5.5.2.

5.7 Discussion

In this section, we analyze the results obtained, and explore alternatives to using the DWF. We also address inter-macro wiring in the DWF.

5.7.1 Cross-talk Problems and our Benchmark Examples

In our delay variation experiments of Sections 2.5.1 and 3.3.1, we simulated wires up to 75μm in length. From the routing results for the standard cell and Fabric3 design styles in Section 5.6.2, we note that the semi-perimeter of the resulting designs

Process	Driver	L	t_f^+	t_r^+	t_f^0	t_f^-	t_r^-	*max/min*
0.1μm	3×	240μm	175.55	187.59	94.65	55.31	54.35	3.45
0.1μm	6×	240μm	104.90	109.63	65.99	46.42	44.94	2.44

Table 5.10: Delay variation for a 240μm Metal2 wire

Process	Driver	L	t_f^+	t_r^+	t_f^0	t_f^-	t_r^-	*max/min*
0.1μm	3×	240μm	80.95	83.55	79.47	80.43	77.97	1.07
0.1μm	6×	240μm	60.36	61.52	59.20	59.94	59.35	1.04

Table 5.11: Delay variation for a 240μm Metal2 wire in the DWF

can be as high as 240μm.

To get an idea of the delay variation due to cross-talk for this length of wire, we performed further delay variation experiments. The experimental setup was exactly as described in Sections 2.5.1 and 3.3.1, except that the wire lengths were 240μm. Tables 5.10 and 5.11 report these results. These tables are organized in the same manner as the delay variation tables of Sections 2.5.1 and 3.3.1. The only difference is that column 2 in the new tables reports the driver size utilized in the experiment.

From Tables 5.10 and 5.11 we observe a very large delay variation due to cross-talk (up to 3.45:1) in the traditional layout style. The delay variation for the DWF is at most 1.07:1.

Of the 14 benchmark examples we used, 11 have a semi-perimeter in excess of 75μm. For these examples, the delay variation due to cross-talk using the traditional layout style would be unacceptable. For the largest example, the delay variation is as high as 3.45:1.

Signal integrity for these examples using the traditional layout style would also be a problem, based on the results of Tables 2.6 and 2.7, but not for the DWF (see Tables 3.5 and 3.6).

Process	Driver	L	t_f^+	t_r^+	t_f^0	t_f^-	t_r^-	*max/min*
0.1μm	15×	240μm	66.30	68.66	52.75	41.45	41.54	1.66
0.1μm	30×	240μm	55.94	55.82	48.40	41.44	43.03	1.35
0.1μm	45×	240μm	56.81	54.53	50.92	44.11	47.33	1.29
0.1μm	60×	240μm	59.03	56.41	54.79	48.25	50.57	1.22

Table 5.12: Delay variation for a 240 μm Metal2 wire with larger drivers

5.7.2 Exploring an Alternative to the DWF

In Sections 5.5 and 5.6, we showed that the area overhead of using the DWF in the Fabric3 scheme was about 25%. We could consider not using the DWF, and using up the area gain to size up drivers in the design such that delay variation due to cross-talk is diminished[6]. The question is whether we can reduce the delay variation sufficiently by sizing up the drivers.

To determine the usefulness of this idea, we ran an experiment identical to that of Table 5.10, but with much larger driver sizes. These results are reported in Table 5.12.

Table 5.12 indicates that even with a driver which is 60× minimum, the delay variation is still as high as 1.22:1. A driver of this size would incur a significant area overhead. Therefore the above alternative to the DWF does not seem to be practical. This idea might be feasible if the delay variation due to cross-talk were smaller.

5.7.3 Inter-Macro Wiring in the DWF

In this chapter, we showed that the Fabric3 methodology is a practical design style for the implementation of a logic function (also referred to as a *macro*). However it is possible that the area overhead for inter-macro wiring may be high for the DWF. Further, the global wiring needed for inter-macro wiring may be a scarce resource in DSM technologies. Therefore, performing routing at twice minimum pitch may not be acceptable.

Obviously, with a non-DWF inter-macro wiring scheme, cross-talk and signal in-

[6]From our delay variation experiments in Section 2.5.1, we observe that using a larger driver results in a lower delay variation.

tegrity would be significant problems. On the other hand, the 50% wiring overhead of the DWF may be too high for inter-macro wires[7].

One way to reduce this overhead is to implement a different fabric in the inter-macro wiring region. A possibility is the $\cdots$ *VSSGSSVSSGSS* $\cdots$ fabric. The wiring overhead for this fabric is 33%, but in practice it would be lower since the non-DWF layout scheme would require the expenditure of some area for routing *VDD* and *GND*. In this fabric, a pair of signal wires are routed alongside each other. Such a pair of wires would be required to have temporal characteristics that do not make them cross-talk candidates. An algorithm to find such wires at the intra-macro level was described in [KSV96]. The fact that each wire has at most one aggressor would increase the number of cross-talk immune wire pairs in the design.

5.8 Chapter Summary

In this chapter, we have presented a network of PLA based design methodology for use in DSM IC design. In this methodology, we cluster the original circuit into a network of PLAs of bounded width and height, which we place and route within the DWF fabric described in Chapter 3.

For a series of examples, the area penalty of the PLA implementation style is shown to be a mere 2.4% compared to the standard-cell approach. This is in spite of the fact that the DWF is used in the routing area between PLAs. The DWF is not used in the standard cell approach. For the same examples, the timing of our approach was on average 15% better than the standard cell approach. These timing and area comparison results remain essentially unchanged, regardless of whether 2, 3, 4, 5 or 6 metal layers are used to perform the routing.

On the other hand, if the network of PLAs is routed at minimum pitch on all metal layers (i.e. the DWF is not used in the routing region), then we obtain an overall timing improvement of about 15% and an overall area improvement of 20%

[7]In practice, however, this overhead would be less than 50% since the non-DWF layout scheme would expend some area in performing power and ground routing, which is already accounted for in the DWF.

compared to standard cells. Once again, these improvements remain essentially the same, regardless of whether 2, 3, 4, 5 or 6 metal layers are used to perform the routing. This comparison would be relevant if cross-talk was not an issue and therefore the DWF was not required.

The *advantages* of our method are as follows:

- High speed. Each PLA is shown to be on average 2.1× faster than its corresponding standard-cell based circuit implementation. Also, the network of PLAs is about 15% faster than the standard cell implementation of the same netlist.

- Low area overhead. Over a series of examples, we show that our scheme has an area overhead of 2.4%. This is in spite of the fact that the DWF is used in the routing area between PLAs, but not used in the standard cell case. Each individual PLA is shown to be 2.17× smaller than its corresponding standard-cell based circuit implementation.

- Power and ground routing is done implicitly, and not in a separate step in the design methodology. Power and ground resistances are very low and vary much less compared to the power and ground distribution used in the standard cell methodology. Even though the PLAs locally break the power and ground gridding structure on Metal1 and Metal2, we determined that this causes a negligible change in the power and ground resistance. This is because the contribution of upper metal layers to the resistance of the power and ground network far outweighs that of the lower metal layers.

- The regular arrangement of metal conductors in our scheme results in low and highly predictable inductive and capacitive parasitics, resulting in highly predictable designs.

- With this methodology, the cross-coupling capacitance between signal wires drops by one to two orders of magnitude, thereby all but eliminating the delay variation and signal integrity problems due to cross-talk. The delay variation of

a signal wire due to switching activity on its neighboring signal wires is 1.03:1, compared to a 2.01:1 variation using conventional layout techniques.

- Smaller and uniform inductances for all wires on the chip, compared to larger and unpredictable values using the existing layout styles.
- The uniformity of inductive and capacitive parasitics which results from the regularity of the DWF is a feature that CAD tools can exploit [OB98b].
- The DWF also results in tighter tolerances on the inter-layer dielectric thicknesses due to the fact that metal is maximally gridded all over the IC die, which in turn results in a tighter control on inter-layer wiring capacitances.
- Finally, the DWF enables us to easily generate a low-skew global clocking network due to the low and uniform parasitics.
- Rapid design turn-around time due to highly regular structures and regular parasitics.

The *disadvantage* of our method is an increase in power consumption. We will see techniques to curb this increase in Chapter 7.

With a network of PLAs, there is a direct relationship between the cost function being optimized during synthesis, and the PLA implementation, since there is no intervening technology mapping step. This helps ensure that benefits of synthesis optimizations are not lost in the implementation step. We will see evidence of this in Chapter 6, where we will show that wire removal at the technology independent level is not effective for a standard cell based design, because the technology mapping step negates the benefits of wire removal. However, wire removal is shown to be very effective for a network of PLAs.

Chapter 6

Wire Removal in a Network of PLAs

6.1 Chapter Overview

So far, we have demonstrated that a circuit implementation based on a network of approximately equal-sized PLAs yields a fast, compact, and cross-talk resistant design. The use of minimum-sized transistors in the PLA core results in a fast and dense layout, while a structured arrangement of wires guarantees an effective shielding among signals. The speed and area of each PLA in this design style was reported to be about 50% less than the corresponding standard-cell based implementation.

In order to reduce the area utilized by such a network, the removal of wires between individual PLAs is effective. This increases the freedom to place the PLAs, reduces wiring area and eliminates potential wire congestion in the routing area. In this chapter, we focus on Sets of Pairs of Functions to be Distinguished (SPFDs) as a candidate technique for wire removal. We describe the wire removal experiments that we performed, using both binary Sets of Pairs of Functions to be Distinguished (SPFDs) (Section 6.2) as well as multi-valued SPFDS (Section 6.3).

Wire removal is a technique where the total number of wires between individual circuit nodes is reduced, either by removing wires, or replacing them with other existing wires.

We perform two separate sets of experiments to test the effectiveness of SPFD-based wire removal. In our first experiment, we show that the benefit of binary SPFD-based wire removal is insignificant when the circuit is mapped using standard cells. For this experiment, we use the binary SPFD code of [SB98]. The authors of [SB98] report a significant average reduction in the number of wires for technology-independent wire removal. However, we show that when technology mapping is performed on the resulting circuits, the benefits of wire removal are erased. Binary SPFD based wire removal *after* technology mapping is not effective since the logic in a single gate is small and hence SPFDs have little effect in removing wires. On the other hand, we demonstrate that binary SPFD based wire removal is very effective in the context of a network of PLAs. In this experiment we do not utilize multi-valued SPFDs. We apply the wire removal algorithm of [SB98] to a network of PLAs, and demonstrate an approximate 20% reduction in the number of wires, which directly translates into a reduction of the layout area. This is because a separate technology mapping step is not required when the circuit is implemented as a network of PLAs.

In addition, we generalize the notion of SPFDs to multi-valued networks. We observe that (multi-output) PLAs can be modeled as multi-valued functions. Hence a network of PLAs can be modeled as a multi-level network of multi-valued nodes. We extend the binary wire removal technique described in [SB98] to the multi-valued case, and use this idea to perform wire removal for a network of PLAs. This flavor of wire removal is performed after the clustering of a circuit into a network of PLAs. We also observe that since each multi-valued node is more complex than the binary nodes encountered in [SB98], additional flexibility is obtained in optimizing them, as evidenced by our results. Although the full flexibility of multi-valued wire removal has not been exploited in our work, we still get good reductions in layout area.

In our second experiment, we describe three separate wire removal experiments. Wire removal is invoked either before clustering the original netlist into a network of PLAs, or after clustering, or both before and after clustering. For wire removal before clustering, binary SPFD-based wire removal is used. Binary SPFD based wire removal is performed in the manner described in [SB98]. For wire removal after clustering, multi-valued SPFD-based wire removal is used since the multi-output PLAs can be

viewed as multi-valued single output nodes. We demonstrate that these techniques are effective. The most effective approach is to perform wire removal both before and after clustering. Using these techniques, we obtain a reduction in placed and routed circuit area of about 11% compared to a PLA network without wire removal. This reduction is significantly higher (about 20%) for the larger circuits we used in our experiments. In this set of experiments, we perform both binary SPFD based wire removal as well as multi-valued SPFD based wire removal.

In Section 6.2, we define binary valued SPFDs, and describe their use for wire removal. Section 6.3 introduces multi-valued SPFDs, along with a description of their use for wire removal. Section 6.4 reports our experimental results for the two sets of wire removal experiments we conducted.

6.2 Binary SPFDs

SPFDs were introduced in [YSN96] in the context of FPGA optimization. In [Bra97] this technique was refined and adapted to multi-level networks, while its application to technology-independent logic optimization was described in [SB98].

6.2.1 Definitions

Definition 6.1 *A function f is said to* **distinguish** *a pair of functions g_1 and g_2 if either one of the following two conditions is satisfied:*

$$g_1 \leq f \leq \overline{g}_2 \tag{6.1}$$

$$g_2 \leq f \leq \overline{g}_1 \tag{6.2}$$

Note that this definition is symmetrical between g_1 and g_2. We can think of 6.1 and 6.2 as specifying two incompletely specified functions, with g_1 as the onset and g_2 as the offset in 6.1 or vice-versa for 6.2.

Definition 6.2 *An* **SPFD**

$$\{(g_{1a}, g_{1b}), \ldots, (g_{na}, g_{nb})\}$$

is a set of pairs of functions to be distinguished.

We can think of an SPFD as an undirected graph with vertices

$$g_{1a}, g_{1b}, g_{2a}, g_{2b}, \cdots, g_{na}, g_{nb}. \tag{6.3}$$

This graph has edges

$$(g_{1a}, g_{1b}), (g_{2a}, g_{2b}), \cdots, (g_{na}, g_{nb}). \tag{6.4}$$

An edge (g_{ia}, g_{ib}) means that minterm g_{ia} must be assigned a different functional value from minterm g_{ib}.

Definition 6.3 *A function f* **satisfies** *an SPFD, if f distinguishes each pair of the set, i.e.*

$$[((g_{1a} \leq f \leq \overline{g}_{1b}) + (g_{1b} \leq f \leq \overline{g}_{1a})] \wedge \ldots \wedge$$
$$[(g_{na} \leq f \leq \overline{g}_{nb}) + (g_{nb} \leq f \leq \overline{g}_{na})]$$

An SPFD can be conveniently used to express the flexibility in implementing a node in a network. The only condition required is that the function implemented at the node satisfies its node SPFD. Note that vertices of a node's SPFD correspond to the on-set, off-set or don't-care minterms of the node function. There are no edges incident on don't-care minterms. There are edges between each on-set minterm and each off-set minterm.

When the SPFD consists of a single pair, it represents two incompletely specified functions (ISF) where one is the complement of the other. If each of the $\{(g_{1a}, g_{1b}), (g_{2a}, g_{2b}), \cdots, (g_{na}, g_{nb})\}$ are pairwise disjoint, then the SPFD represents 2^n ISFs[1].

Classically, in computing the flexibility at a node in a Boolean network, the don't cares that are computed, represent a single ISF.

[1]Note that an SPFD cannot represent a single function, it always represents at least a pair. Thus it cannot represent the function 1.

6.2.2 Wire Removal/Replacement Using Binary valued SPFDs

The information content of a wire (which is effectively the set of pairs of minterms it can distinguish) in a network can be effectively represented by an SPFD. This allows SPFDs to help remove certain "difficult" wires in the network or to replace them by other wires. The technique of wire removal/replacement using SPFDs works as follows.

Consider a multi-level network, with some nodes η_i, η_j and η_k. Given a wire (η_i, η_j), its SPFD represents the pairs of minterms that have to be distinguished by it. Thus, in a sense, the SPFD of (η_i, η_j) encodes the information content required of that wire. If the wire (η_i, η_j) need not uniquely distinguish any minterms required of node η_j (i.e. it has no unique information content required), we can remove it as an input to η_j. We can also try to replace it by another wire as long as the second wire has all the information required of the original. So, a wire (η_s, η_j) can replace the wire (η_k, η_j) if all the minterms required to be distinguished by the wire (η_k, η_j) are also distinguished by (η_s, η_j). In other words, the objective is to replace wire (η_k, η_j) from node η_k to η_j with a wire (η_s, η_j) from node η_s to η_j, such that the original SPFD at η_j is covered by the union of the SPFDs of its inputs, and some gain is realized by this change. In the sequel, we shall refer to this technique as *wire_replace*. In [SB98], it was shown that there can be a substantial reduction in the number of wires (at the technology-independent level) in the network using the *wire_replace* algorithm. Note that *wire_replace* also removes wires whose SPFDs are empty.

For a detailed exposition on SPFDs and how they are computed and used for wire replacement, see [SB98].

6.3 MV-SPFDs

We give a graph-theoretic definition of MV-SPFDs which is a generalization of the definition of binary SPFDs of the previous section.

6.3.1 Definitions

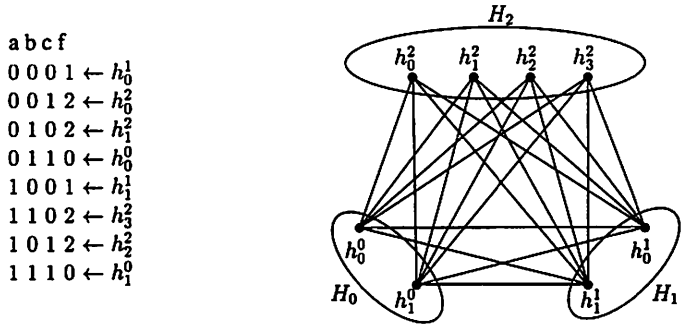


Figure 6.1: A Multi-valued SPFD.

Definition 6.4 *An* **MV-SPFD** $\mathcal{F}(y)$ *on a domain* Y *is an undirected graph* (V, E) *where each* $v \in V$ *corresponds to a unique minterm* $v = (y_1, y_2, \cdots, y_k) \in Y$. *An edge* $(e = (v_1, v_2)) \in E$ *means that the minterms corresponding to the two vertices* v_1 *and* v_2 *must have different functional values.*

Figure 6.1 shows a multi-valued node H with k values, and its corresponding MV-SPFD. This MV-SPFD can be described as a set with k tuples $\{H_0, H_1, \cdots, H_{k-1}\}$. Each tuple H_i consists of several minterms $\{h_1^i, h_2^i, \cdots, h_{n_i}^i\}$. Each minterm in H_i must be distinguished from (i.e. have different functional values than) minterms in each of the remaining $k-1$ tuples. Each H_i is also referred to as a *component.*

Definition 6.5 *A function* $F(y)$ **implements** $\mathcal{F} = (V, E)$ *if* $F(y)$ *is a valid coloring of* $\mathcal{F}$, *i.e.*

$$F(y^1) \neq F(y^2), \forall (y^1, y^2) \in E$$

For a function F to implement $\mathcal{F}$, F assigns a different value to minterms h_p^i and h_q^j, for $i \neq j$. Thus the *chromatic number* of an MV-SPFD is the minimum number of values required to implement the MV-SPFD using a multi-valued function. Each different coloring of this graph represents a different incompletely specified multi-valued function (ISF). This is one source of flexibility of MV-SPFDs.

Note that this definition of an MV-SPFD is a Multi-valued generalization of the definition of [SB98].

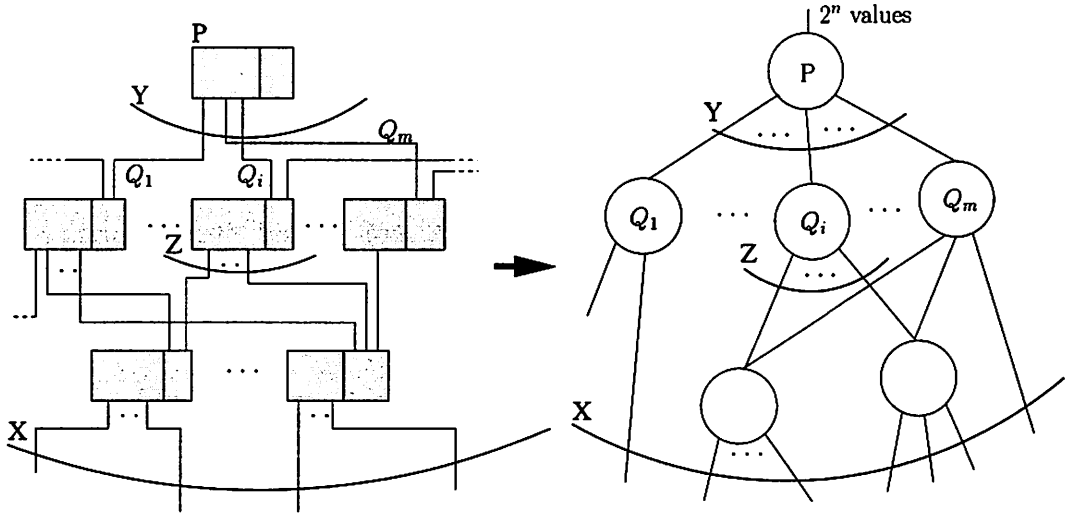
6.3.2 Wire Removal/Replacement Using MV-SPFDs

In a network of PLAs, each individual PLA is a multi-input, multi-output structure. Suppose a given PLA has k outputs. In that case, it can be modeled as a single output node with 2^k values. A multi-valued SPFD can be computed for each node and can be used to remove wires in its fanin. A network of PLAs can be modeled as a multi-level network of multi-valued nodes. The binary SPFD techniques for computing and distributing SPFDs using BDDs [Bry86] can be generalized to MV-SPFD techniques using MDDs [SKMB90]. The details of the computation are discussed below.

Consider a node η_j in a multi-level, multi-valued logic network. We know that the MV-SPFD of η_j represents the set of multi-valued minterms (henceforth equivalently referred to as minterms) that should be distinguished by η_j in order that it provides enough information to its fanouts. To achieve this, it is necessary and sufficient that each pair of minterms in the MV-SPFD of η_j be distinguished by at least one of its fanin wires. Thus, the union of the MV-SPFDs of its fanin wires should cover the MV-SPFD of η_j. Alternately stated, just as in the binary case, the MV-SPFD of a node/wire gives the information content required of the node/wire. So, all the information contained in a node has to be provided by its fanins.

We define the **minimum MV-SPFD** of a wire (η_i, η_j) to be the set of pairs of minterms of η_j that must be distinguished exclusively by this wire (i.e. for each pair, no other input computes different values for the pair). In order to ensure that all the pairs of minterms in the MV-SPFD of η_j are distinguished, the wire (η_i, η_j) must distinguish at least these pairs of minterms.

Given the MV-SPFD of the node η_j, we compute the minimum MV-SPFD of each fanin wire. If the minimum MV-SPFD of a fanin wire is not empty, then we cannot remove this wire since it uniquely distinguishes some pair of minterms in the MV-SPFD of the node η_j. On the other hand, if the MV-SPFD of a fanin wire is empty, it is a candidate for removal. However, we cannot simultaneously remove some or all fanin wires whose minimum MV-SPFDs are empty. This is because there could be two fanin wires (η_i, η_j) and (η_k, η_j) with empty minimum MV-SPFDs, such that



a) A network of PLAs

b) Its corresponding multi-valued network

Figure 6.2: Multi-valued SPFD based wire removal.

both wires distinguish the pair of minterms (m_1, m_2) in the MV-SPFD of η_j, and no other fanin wire distinguishes this pair of minterms. In such a situation, at least one of these wires must be retained. If both wires are removed, (m_1, m_2) will not be included in the new MV-SPFD of η_j, and hence the resulting network will not be functionally correct.

The procedure for removing wires in a PLA network is explained below. Consider the PLA P, which has m inputs and n outputs. Figure 6.2-a shows a sample network of PLAs, in which P resides. Each rectangle in this figure represents a PLA, with its AND (input) plane on the left, and the OR (output) plane on the right. The PLA P can be considered equivalently as a multi-valued node with 2^n values, and m multi-valued inputs, as shown in Figure 6.2-b.

The MV-SPFD of P, denoted as $\mathcal{P}(Y)$, is computed from its original multi-valued function (MVF)[2]. This MV-SPFD must distinguish every minterm in every component of its MVF from every minterm in every other component of its MVF. After computing $\mathcal{P}(Y)$, (here Y is the space of the fanins of P) we re-assign the task of

[2]A multi-valued function (formally, $\mathcal{F} : P_1 \times P_2 \times \ldots \times P_n \mapsto P_m$) of n variables $X_1, X_2, \ldots, X_n$ can take on P_m integer values $\{0, \ldots, |P_m| - 1\}$. The multi-valued variable X_i can take on integer values $P_i = \{0, \ldots, |P_i| - 1\}$.

distinguishing edges of $\mathcal{P}(Y)$ to the fanins of P, using the following procedure.

- Fanins of P that have non-empty minimum MV-SPFDs, denoted as Y', are first identified.
- All the edges of $\mathcal{P}(Y)$ that are distinguished by these fanins are assigned to these fanins and are removed from $\mathcal{P}(Y)$.
- A weighted covering problem is set up between the remaining fanins of P, $Y \backslash Y'$, and the edges of the modified $\mathcal{P}(Y)$. The fanins are weighted according to the following heuristic : the smaller the number of fanouts of a particular fanin, the greater its weight. This means that a fanin with a single fanout has the largest weight and so has the least likelihood of being included in the solution. Hence the corresponding wire is most likely to be removed. Let the solution of this weighted covering problem be Y''.

The new fanin space of P is the union of Y' and Y'' and will be subsequently referred to as $\hat{Y}$. Now, P is modified. First the image of $\mathcal{P}(Y)$ is computed on the primary input space X. This image is projected back to the $\hat{Y}$ space, to get $\hat{\mathcal{P}}(\hat{Y})$, the new MV-SPFD of P in terms of its new fanins. We use a coloring algorithm to obtain a new ISF at P. The connected components of the MV-SPFD are obtained and each component is colored appropriately to obtain a new ISF. A logarithmic encoding is used. Next we run Espresso-MV [RSV85] to get the new minimized function of P.

We proceed in a topological order from the inputs to the outputs in the network and perform wire removal on each node in the network.

In the sequel, we refer to this algorithm as *mv_wire_replace*.

The above method for performing wire removal is effective for the following reasons:

- It was observed [SB98] that whenever the nodes were simple, binary SPFD based optimizations resulted in little improvement. The multi-valued nodes corresponding to the PLAs are complex, since they usually constitute several inputs and outputs. As a result, the flexibility offered by MV-SPFDs can be exploited to the fullest.

- In an SPFD-based computation, a logic node and its fanins are optimized simultaneously. When the fanins of a node F are modified, the logic function of F needs to be changed as well. In [SB98], to avoid the propagation of changes throughout the transitive fanout of F, the CODCs (compatible output don't-cares) [Sav92] of the immediate fanouts of F are used as the SPFDs of these nodes, in order to *block* the changes of F from propagating to its fanouts. For the application of MV-SPFDs in PLA-based wire removal, the changes to any node (i.e. a PLA) F do not need to be blocked by the don't cares of nodes in the fanout of F. This is because fanout PLAs can be easily re-implemented if F changes, as long as the total number of product terms in the fanout PLAs are bounded. This is expected to result in significantly more flexibility while optimizing a given PLA.

6.3.3 Controlling change

As mentioned in the previous section, any valid coloring of $\mathcal{P}(Y)$ can be used to obtain an incompletely specified MV function for P. But, if a node is changed, then its changes must be propagated throughout the transitive fanout of P. Although this can be done, in practice it can prove to be computationally expensive. So we block the changes in the new function by its MV-CODCs [JB99] (a generalization of CODCs [Sav92] for the multi-valued case). Thus, at any point in the algorithm, the *region of change* consists of a single node, and possibly its immediate fanins.

6.3.4 Multi-valued SPFDs vs binary SPFDs

Although MV-SPFDs are a generalization of binary SPFDs, there are some interesting points that they bring up.

- In [SB98], the authors discuss how binary SPFDs could run into the problem of non-bipartition i.e. there could arise situations where an SPFD can no longer be colored by two colors. This situation arises because during the SPFD computation, one cannot restrict the chromatic number of an SPFD without losing

optimization flexibility. Since some binary SPFD algorithms exploit their bipartite nature, this could lead to inelegant solutions. MV-SPFDs are a simple and elegant way to handle this problem, since we do not need to restrict the chromatic number of an MV-SPFD.

- The coloring of MV-SPFDs gives rise to interesting possibilities. A binary SPFD with n connected components can be colored in 2^n ways. An MV-SPFD with n components can be colored in $k_1! * \cdots * k_n!$, where k_i is the chromatic number of the ith component. This flexibility can be exploited in many ways. In a network of PLAs, for instance, re-encoding a node could change the wiring connections between a node and its fanouts. So, if we expand the *region of change* to include a node *and* its fanouts, we can use some encoding algorithm to suitably modify the wiring between a node and its fanouts. This is a difficult problem and is currently being investigated.

6.4 Experimental Results

In Section 6.4.1, we perform experiments with binary SPFD based wire removal. We demonstrate the utility of this technique in a network of PLAs, and show that it is not useful in a traditional standard-cell based implementation style.

In Section 6.4.2, we perform binary and multi-valued SPFD based wire removal experiments in a network of PLAs, and demonstrate the effectiveness of these techniques.

In both sections, routing is performed using the DWF in the routing region between PLAs. The placement and routing flow utilizes two metal layers, and is described in Section 5.5.

6.4.1 Experiment 1

In our experiments to validate the usefulness of SPFD-based wire removal for a network of PLAs, we utilize the *wire_replace* code which was described in [SB98]. This

Circuit	Standard Cell			Network of PLAs		
	before WR	after WR	Ratio	before WR	after WR	Ratio
C432	1996.08	1956.64	0.980	2655.18	2285.26	0.861
C499	4840.91	4914.14	1.015	4131.11	3599.70	0.871
C880	4619.33	4869.08	1.054	7152.46	5400.49	0.755
rot	14254.21	13925.60	0.977	14612.87	8361.74	0.572
alu2	4104.82	4144.26	1.010	3141.52	1626.16	0.518
AVG			1.007			0.715

Table 6.1: Wire Removal without Prior Optimization

form of wire removal corresponds to the *WRB* wire removal experiment described in Section 6.4.2. The computation is done at the level of binary-valued SPFDs.

Table 6.1 reports the results of *wire_replace* on unoptimized circuits. In our tables, the final layout area of the circuit is measured in units of square microns. All reported numbers include the area for the actual logic as well as routing.

Columns 2 and 3 report the results for a standard-cell based implementation, with and without *wire_replace*, respectively. Column 4 reports the ratio of the standard cell area after wire removal, to the area before wire removal. Columns 5 and 6 report the results for a PLA based implementation, with and without *wire_replace*, respectively. Column 7 reports the ratio of the PLA based area after wire removal, to the area before wire removal. Table 6.2 is organized in the same fashion, except that each circuit was first subjected to *script.rugged*, a technology-independent optimization script in SIS [SSL+92].

In essence, we note that for the standard-cell based methodology, *wire_replace* does not impact the overall layout area. This is because the benefits attained through wire removal at the technology independent level are negated by the technology mapping step. However, in the case of the network of PLAs, *wire_replace* results in a significant reduction in circuit area (23.5% for the non-optimized case, and 16.9% in the optimized case). This is due to the absence of a technology mapping step after wire removal. As a result, the benefits of wire removal are directly translated into a reduction in circuit area.

Circuit	Standard Cell			Network of PLAs		
	before WR	after WR	Ratio	before WR	after WR	Ratio
C432	1701.27	1874.02	1.102	2634.52	2439.23	0.926
C499	4664.40	4547.98	0.975	4756.41	3785.60	0.796
C880	4722.61	4611.82	0.977	3909.53	3248.56	0.831
rot	12504.12	12042.19	0.963	10372.84	7980.56	0.769
alu2	3323.67	3030.73	0.912	1757.60	1774.50	1.010
C1355	4964.84	5011.79	1.009	3995.91	3635.38	0.910
C1908	5385.47	5528.18	1.026	4553.61	3836.30	0.842
AVG			0.995			0.869

Table 6.2: Wire Removal after *script.rugged*

6.4.2 Experiment 2

To validate the usefulness wire removal for a network of PLAs, we utilize the two SPFD-based wire removal techniques.

- For wire removal before clustering a circuit into a network of PLAs, we use the *wire_replace* code detailed in [SB98] and in Section 6.2.2. This computation is done at the level of binary-valued SPFDs, since the logic nodes are binary valued before clustering into PLAs.

- After clustering into a network of PLAs, each PLA can be viewed as a multi-valued node, as described in Section 6.3.2. At this point, multi-valued SPFD-based wire removal is invoked, using *mv_wire_replace* as described in Section 6.3.2.

The clustering and wire removal code was written in SIS [SSL+92]. Placement of the network of PLAs was done using VPR [BR97], an FPGA-based placement and routing tool. Since all PLAs in the network of PLAs have roughly the same size, VPR is a good choice for placement. However, routing is not done using VPR since it assumes an FPGA connection topology. Therefore, routing of the network of PLAs was performed using *wolfe* [SSV85], using the same design flow as described in Section 5.5.1.

The initial *blif* netlist for the benchmark circuit is first clustered into nodes with up to 5 inputs. This new netlist is the starting point for all wire removal experiments.

We now perform one of 4 wire removal experiments:

- For *no wire removal*, (NOWR) we cluster the netlist into a network of PLAs. This network is now placed and routed as described above.
- For *wire removal after clustering*, (WRA) we follow the clustering step by a wire removal step, using multi-valued SPFD-based wire removal. The result of this step is then placed and routed.
- For *wire removal before clustering*, (WRB) we perform binary-valued SPFD-based wire removal on the netlist, and then cluster the resulting netlist into a network of PLAs. This network is then placed and routed.
- For *wire removal before and after clustering*, (WRBA) we perform binary-valued SPFD-based wire removal on the netlist, and then cluster the resulting netlist into a network of PLAs. This is followed by multi-valued SPFD-based wire removal. The resulting netlist is placed and routed as described above.

We constrain the clustering step by imposing a maximum width and maximum height constraint on the PLAs. In this section we report the results of experiments with two such combinations which utilize a PLA height constraint of 15 and 20, and a PLA width constraint of 40. The total number of outputs of each PLA is constrained to be no larger than 5.

Table 6.3 reports the results of wire removal on some benchmark circuits. All examples in this table use a PLA height constraint of 15, and a PLA width constraint of 40. Table 6.4 reports the results of wire removal where all examples use a PLA height constraint of 20 and a PLA width constraint of 40. Each PLA has 5 or less outputs in both cases. In both tables, the final layout area of the circuit is measured in units of square microns. All reported numbers include the area for the actual PLA logic plus the routing area. For each table, the first column reports the circuit name. The second column reports the resulting layout area using no wire removal (NOWR), while the third column reports layout area using MV-SPFD based wire removal after clustering the circuit into a network of PLAs (WRA). The fourth column reports

Circuit	NOWR	WRA	Improve %	WRB	WRBA	Improve %	NOWR-WRA%	NOWR-WRB%	NOWR-WRBA%	BEST%
vda	20862.11	17331.89	16.96	19040.67	16693.44	12.24	16.96	8.78	19.95	19.95
frg2	12111.67	10703.33	11.63	11191.56	10233.89	8.61	11.63	7.49	15.45	15.45
C1908	10590.67	10534.33	0.52	8600.22	8130.78	5.44	0.52	18.81	23.23	23.23
apex6	8356.11	7586.22	9.25	8281.00	7980.56	3.54	9.25	0.98	4.48	9.25
x3.blif	8299.78	8149.56	1.88	8619.00	8281.00	3.89	1.88	-3.72	0.32	1.88
toolarge	8093.22	8074.44	0.22	8262.22	8187.11	0.98	0.22	-2.14	-1.14	0.22
xl	3849.44	3511.44	8.39	3398.78	3492.67	-2.98	8.39	11.75	9.12	11.75
x4	3830.67	3999.67	-4.25	3642.89	3642.89	0.00	-4.25	4.99	4.99	4.99
alu2	3042.00	2760.33	9.26	3098.33	3060.78	1.05	9.26	-1.40	-0.34	9.26
C432	2572.56	2535.00	1.95	2309.67	2328.44	-0.71	1.95	10.53	9.89	10.53
term1	2347.22	1971.67	15.96	1802.67	1652.44	8.62	15.96	23.26	29.88	29.88
apex7	1859.00	1596.11	13.92	1783.89	1727.56	3.03	13.92	3.96	6.87	13.92
ttt2	995.22	845.00	15.36	976.44	957.67	2.05	15.36	1.94	3.95	15.36
count	676.00	600.89	13.23	694.78	600.89	14.16	13.23	-1.08	13.23	13.23
pcle	507.00	488.22	2.78	507.00	488.22	2.78	2.78	0.00	2.78	2.78
decod	338.00	338.00	0.00	338.00	338.00	0.00	0.00	0.00	0.00	0.00
AVERAGE			7.32			3.92	7.32	5.26	8.92	11.35

Table 6.3: Wire Removal Experiments - max width 40, max height 15

Circuit	NOWR	WRA	Improve %	WRB	WRBA	Improve %	NOWR-WRA%	NOWR-WRB%	NOWR-WRBA%	BEST%
vda	23359.56	19284.78	17.41	21237.67	17857.67	15.94	17.41	9.04	23.53	23.53
frg2	10177.56	9050.89	11.15	10327.78	9520.33	7.83	11.15	-1.54	6.42	11.15
C1908	12543.56	10947.44	12.67	9726.89	8431.22	13.29	12.68	22.48	32.78	32.78
apex6	9126.00	8806.78	3.37	8957.00	8374.89	6.64	3.38	1.70	8.23	8.23
x3.blif	9370.11	8431.22	10.12	8750.44	8919.44	-1.94	10.12	6.76	4.95	10.12
toolarge	8844.33	8788.00	0.69	8769.22	8788.00	-0.18	0.69	0.93	0.75	0.93
xl	3962.11	3943.33	0.60	3887.00	3943.33	-1.64	0.60	1.95	0.34	1.95
x4	3868.22	3811.89	1.62	4056.00	4187.44	-3.32	1.62	-4.71	-8.19	1.62
alu2	3211.00	2985.67	6.73	3436.33	2891.78	15.50	6.73	-7.06	9.53	9.53
C432	2929.33	2760.33	5.81	2497.44	0.00	-	5.81	14.89	-	14.89
term1	2441.11	2121.89	13.36	1765.11	1614.89	8.61	13.36	27.74	33.97	33.97
apex7	1934.11	1821.44	5.75	2028.00	1934.11	4.71	5.75	-4.54	0.39	5.75
ttt2	1126.67	976.44	12.73	1126.67	1070.33	3.68	12.73	0.72	4.37	12.73
count	901.33	769.89	15.96	901.33	788.67	13.86	15.97	0.00	13.86	15.97
pcle	582.11	563.33	2.44	582.11	582.11	0.00	2.44	0.00	0.00	2.44
decod	375.56	375.56	0.00	375.56	375.56	0.00	0.00	0.00	0.00	0.00
AVERAGE			7.53			5.19	7.53	4.27	8.18	11.60

Table 6.4: Wire Removal Experiments - max width 40, max height 20

the improvement in layout area by performing WRA (compared to the NOWR case). The fifth column contains layout area results when binary-valued SPFD based wire removal is performed before clustering into a network of PLAs (WRB). The sixth column reports layout area when SPFD based wire removal is performed both before and after clustering into a network of PLAs (WRBA). The seventh column reports the area improvement of the sixth column over the fifth. The eighth, ninth and tenth columns represent the percentage area improvements of WRA, WRB and WRBA over the NOWR case, respectively. Finally, the eleventh column represents the best area improvement from the preceding three columns.

We observe that the best area reduction using any flavor of wire removal is above 11% for both tables. Also note that the best area reduction is in excess of 19% for the three largest examples. This suggests that SPFD-based wire removal is very effective for larger circuits.

Comparing the wire removal techniques in isolation, we observe that WRBA provides the best average improvement in area (8.92% and 8.18% for Table 6.3 and Table 6.4 respectively). In both these tables, WRBA improves on WRB by an average of 3.92% and 5.19% respectively. The least effective of the three wire removal flows is WRB.

Furthermore, the results reported in Section 6.4.1 indicated that wire removal applied to traditional standard-cell based designs results in no area improvement, since wire removal obtained by such techniques is negated by the technology mapping step required in such a design style. This suggests that using a network-of-PLAs design methodology has additional advantages over the standard-cell based design methodology. The reason for this is that in the network-of-PLAs design style, there is a more direct relationship between the cost function being optimized during synthesis, and the actual implementation of the logic. This is because there is no technology-mapping step required in this design style.

Among the three wire removal experiments conducted, the most effective are WRBA and WRA. These two experiments together contributed to a majority of the best case results (column 11). In Table 6.3, in the cases in which WRB contributed the best result, either WRA or WRBA had improvements very close to this. For the C432 example in Table 6.4, WRB contributed the best result, and the improvement provided by WRA trailed it significantly. However, WRAB was not able to complete on this example, so we are not sure if WRAB could have matched this result if the example had completed.

We performed another study where all four experiments used a series of 9 values of maximum PLA height and width. The maximum height varied from 15 to 25 in steps of 5, and the maximum width varied from 40 to 60 in steps of 10. The maximum number of outputs was restricted to 5. We used the best area from each of these 9 cases for each example, and compared the results just as in the tables above. The results

obtained were substantially similar to those reported in Tables 6.3 and 6.4. This is primarily due to the fact that the two combinations of maximum width and height used in Tables 6.3 and 6.4 accounted for the best results for most examples. In this study, the average best case area improvement due to any flavor of wire removal was 11.82%. WRBA once again was the most effective wire removal style, with an average improvement of 9.22%. WRA and WRB had an average improvement of 7.58% and 5.82% respectively. The detailed results of this experiment are not included, since they substantially track the results reported in this section.

6.5 Chapter Summary

In this section we have demonstrated that SPFD based wire removal is a powerful technique for reducing the wiring, and therefore the overall layout area, of a circuit implemented as a network of PLAs. In the first experiment, we show that the binary-valued wire removal algorithm of [SB98] provides a significant reduction in wiring for a network of PLAs, while the same algorithm delivers no improvement for a standard-cell based implementation. This is true regardless of whether logic optimization is performed on the netlist or not.

Our second experiment with wire removal in a network of PLAs also results in significant area savings. The findings of this experiment are summarized below.

- Wire removal results in a best case layout area reduction on average of about 11%.

- This reduction increases to 19% or higher for larger examples, further suggesting the effectiveness of the technique.

- By choosing the best result among WRA and WRBA, we obtain an improvement which is almost as good as the best case improvement over all 3 wire removal styles. These two styles of wire removal account for the best case improvement in a majority of the examples.

- Also, since each of the MV nodes are complex, the MV-SPFD based algorithm has a larger flexibility in re-implementing an MV node.

Chapter 7

Conclusions and Future Directions

In this chapter we summarize our contributions and point out to some future directions in which this research can be expected to grow.

7.1 Conclusions

In this thesis, we focussed on the cross-talk problem in DSM VLSI design. We showed that with decreasing feature sizes of modern VLSI fabrication processes, the cross-talk problem is becoming increasingly important. As a result, signal integrity and delay variation of wires are becoming difficult problems in modern ICs. We demonstrated that these problems are expected to get worse in future processes.

We proposed two VLSI design methodologies to address the cross-talk problem. In both these methodologies, cross-talk is solved a priori by designing a *routing fabric*. This is done by imposing a fixed pattern of wires on the IC die, on all metal layers. We call this pattern the DWF pattern, and it consists of a repeating sequence of wires, $\cdots$ *VSGSVSGS* $\cdots$. Here V represents a *VDD* wire, G represents a *GND* wire, and S represents a signal wire. This sequence of wires recurs on all the metal layers of the IC.

As a result of this choice, the cross-coupling capacitance between signal wires drops by between one and two orders of magnitude, thereby all but eliminating the delay variation and signal integrity problems due to cross-talk. We showed that for

a 0.1μm process, the delay variation due to cross-talk drops from 2.01:1 to 1.03:1 if the DWF is used (for wires of length 75μm).

Other salient advantages of the DWF fabric are

- Signal inductances drop by 50% as well, since the current return path for any signal wire is always adjacent to it. This is particularly significant because problems due to on-chip inductances are becoming evident in modern designs.
- Our scheme eliminates the conventional notion of power and ground routing on the integrated circuit die. Power and ground are essentially "pre-routed" all over the die by suitably introducing vias whenever *VDD* (or *GND*) wires intersect on adjacent metal layers. These power and ground distribution networks have a low and uniform resistance, significantly reducing on-chip voltage drops due to resistive effects.
- The DWF also results in tighter tolerances on the inter-layer dielectric thicknesses due to the fact that metal is maximally gridded all over the IC die.
- The uniformity of inductive and capacitive parasitics which results from the regularity of the DWF is a feature that CAD tools can exploit.
- Finally, the DWF enables us to easily generate a low-skew global clocking network due to the tightly controlled and uniform parasitics.
- In our scheme, the characterization of interconnect parasitics (capacitance, inductance and resistance) becomes extremely simple due to the repeating nature of the DWF pattern. This characterization needs to be done only once for a design, resulting in significant time and computational savings.

We introduced two fabrics that utilize the DWF pattern. *Fabric1* uses the DWF pattern chip-wide, including within the layout cells in the design. This fabric uses a standard-cell based VLSI design flow, and over a series of examples, exhibits an area overhead of about 60% when two metal layers are utilized to perform the routing. However, if more metal layers are available for routing, then this penalty drops to about 17%, due to a better utilization of the routing resources.

The second fabric, *Fabric3*, attempts to reduce this area overhead. In this fabric, logic is implemented as a network of medium-sized Programmable Logic Arrays (PLAs). We choose an implementation of PLAs which is naturally cross-talk immune, and also extremely dense. The routing area between PLAs utilizes the DWF pattern, while a specialized layout fabric is utilized within the PLAs. We show that a single PLA is extremely dense and fast compared to a standard-cell based implementation of the same logic. We introduced synthesis algorithms to cluster a logic netlist into a network of PLAs in the Fabric3 style. Each of these PLAs has a bounded width and height, and is folded to achieve better logic density. For a series of examples, the area penalty for a network of PLA implementation in the Fabric3 scheme is shown to be a mere 2.4% compared to the standard-cell implementation style. These timing and area comparison results remain essentially unchanged regardless of whether 2, 3, 4, 5 or 6 metal layers are used to perform the routing.

With a network of PLAs, there is a direct relationship between the cost function being optimized during synthesis, and the PLA implementation, since there is no intervening technology mapping step. This helps ensure that benefits of synthesis optimizations are not lost in the implementation step. This is demonstrated by the SPFD-based wire removal algorithms we introduced. These algorithms reduce the wiring between PLAs in the network. We showed that these techniques do not work for standard-cell based implementations, since the wire removal achieved at the technology-independent level is lost after technology mapping.

In summary, we have shown how the uniform and predictably low parasitics in our fabric methodologies give rise to a reliable and predictable design style. We have implemented our schemes and compared them with the standard cell design style. The crosstalk immunity, high speed, low area overhead, quick design turnaround time, and high predictability of our methodologies indicate that they are strong candidates as the preferred design methodologies in the DSM era.

7.2 Future Work

There are several avenues for future work in this area.

7.2.1 Alternate Fabrics

In this thesis, we introduced the DWF, which is a simple fabric. Alternative fabrics can be pursued as a means to reduce the power, delay and area characteristics of the network of PLAs. One such fabric that can be used has all PLA inputs available in the positive form. For example, if one of the cubes of the function f is $a \cdot \overline{b} \cdot c$, then we introduce the complement of b as a new input to f. Additionally, in each PLA, we generate each output as well as its complement. So if an output g of a PLA p_1 is an input to PLA p_2, both g and its complement are routed to p_2 (if they are both utilized in the PLA p_2 after the transformation described above). Since both g and its complement are pre-charged, they can be routed side by side without incurring cross-talk problems, using a $\cdots$ *VSSGSSVSSGSS* $\cdots$ fabric. This eliminates the need to route completion signals between PLAs, since a transition on either g or its complement automatically indicates completion. This makes the design style simpler, faster and more robust. In such a fabric, if the complement of an output g is not required to be routed, then other signals which do not exhibit cross-talk with g can be routed alongside g. Hence such a PLA implementation could result in a lower area penalty, and better timing characteristics for the network as well. Additionally, since the bit-lines in such a PLA implementation do not need to be pre-discharged, the overall power consumption could be lower.

Another alternative is to relax the DWF restriction on lower metal layers, while ensuring that routes on these layers are short. In this way, cross-talk problems are avoided on these metal layers, even though the DWF is not employed. This would result in a higher logic and routing density on these layers, resulting in better circuit area characteristics.

7.2.2 Modifications to the PLA Design

When we construct the PLAs in our methodology, we perform folding to increase the logic density of each PLA. In addition to this, other techniques can be employed to increase logic density of the PLAs. It was shown in [Sch80] that encoding of PLA inputs was an effective technique to reduce the number of terms in PLAs, especially

for arithmetic circuits. This can be investigated as well. Additionally, gates can be placed on the outputs of the PLA. This was also shown to be an effective technique for increasing PLA density [Wei79].

A NAND-NAND PLA implementation also requires some attention. It is possible that such a PLA could have better timing or area characteristics than the NOR-NOR style of PLA that we chose in our experiments. Also, PLAs that require the use of only one metal layer can be investigated. Such PLAs are likely to have worse timing characteristics, since polysilicon would need to be used either in the word lines or the bit lines. But since we constrain the PLA width and height in our methodology, it is likely that this penalty is not significant. The additional routing flexibility allowed by this style of PLA could result in smaller circuits.

7.2.3 Reducing Power Consumption

Exploring the power-delay tradeoff is one way to reduce the power consumption of our PLAs. In our methodology, it is possible to reduce driver strengths, and accordingly reduce power at the cost of additional delay.

Another method for reducing power is the alternate fabric described in Section 7.2.1.

Power can be reduced by turning off the clocks to PLAs that are not involved in a certain cycle of the computation.

Finally, restricting the height of the PLAs in our methodology is a simple way of curbing the power consumption, as we saw in Section 5.2.3.

7.2.4 Alternate Circuit Design Styles

There are several other circuit design styles that could be used instead of PLAs. For example, SLAs [PW79, SCH82] could be used to implement sequential circuits in our design style. Also, pass-transistor based circuit implementations [BNNSV97] could possibly fit into a fabric based methodology. A gate-matrix implementation of a circuit [HFK87, CCH87] also needs to be investigated in this context.

7.2.5 Wire Removal

The wire removal techniques we introduced in this thesis did not target specific problematic wires in the network of PLAs. Rather, we attempted to remove wires whenever possible. In the future wire removal could be performed after placement as well. After placement, we can identify wires that are *critical* in the sense that if these wires are removed, the layout area would decrease. A wire removal algorithm which targets such wires should further improve the results obtained.

Also, in our current implementation, the height of the PLAs is allowed to grow after we perform multi-valued SPFD-based wire removal. Were this not allowed, further area savings would probably be obtained.

As mentioned in Section 6.3.4, we also plan to investigate ideas to further exploit the flexibility of MV-SPFD based wire removal.

7.2.6 Alternative Clustering Strategies

Our current strategy for clustering a circuit into a network of PLAs attempts to reduce the wiring between PLAs. This is well motivated, since wires are expensive to implement in the DWF. However, our clustering algorithm does not attempt to reduce circuit delay. Efficient methods to cluster a logic netlist into a network of PLAs would be a natural extension of this work. In an ideal clustering strategy, the delay of the network of PLAs, as well as the wiring between PLAs is minimized.

7.2.7 "Pre-fabricated" Fabric based Circuits

The network of PLAs style of design can be extended to implement logic that has been "pre-wired". This is reminiscent of the *wire-planning* techniques introduced in [OB98a] and [GNBSV98]. Essentially, wires are addressed first in such a methodology, and after that, logic is assigned to nodes such that these pre-assigned wires are used.

In the context of PLA networks, we can pre-assign wiring between PLAs, and then determine the logic of the PLAs. The task is to decompose the original netlist

into a network of PLAs, such that these PLAs reside in their pre-assigned locations, and utilize the pre-assigned wires. This technique is useful because a majority of the delay in DSM circuits lies in the wires.

Bibliography

[asi] Analysis of Silicon Inductors and Transformers for ICs. `http://kabuki.eecs.berkeley.edu/~niknejad/asitic.html`. 3.2.1.2

[BHMSV84] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer Academic Publishers, 1984. 5

[BNNSV97] P Buch, A Narayan, A Newton, and A Sangiovanni-Vincentelli. Logic synthesis for large pass transistor circuits. In *Proceedings of the International Conference on Computer-Aided Design*, pages 663–70, Nov 1997. 7.2.4

[BP83] M Burstein and R Pelavin. Hierarchical channel router. In *Proceedings of the 20th Design Automation Conference*, pages 591–97, 1983. 4.1

[BR97] V Betz and J Rose. VPR: A new packing, placement and routing tool for FPGA research. In *Proceedings of the International Workshop on Field Programmable Logic and Applications*, 1997. 5.5.1, 6.4.2

[Bra97] R Brayton. Understanding SPFDs: A new method for specifying flexibility. In *Workshop Notes, International Workshop on Logic Synthesis*, Tahoe City, CA, May 1997. 6.2

[Bry86] R. Bryant. Graph-based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35:677–691, August 1986. 6.3.2

[Bur94] T Burd. *CMOS Standard Cell 2_3lp Library Documentation.* U C Berkeley, Mar 1994. 4.2.1

[Cad99] Cadence Design Systems, Inc., 555 River Oaks Parkway, San Jose, CA 95134, USA. *Envisia Silicon Ensemble Place-and-route Reference*, Nov 1999. 4.3.1, 5.6.1

[Cas91] Andrea Casotto, editor. *Octtools-5.1 Manuals*, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, September 1991. University of California at Berkeley. 4.2.1, 5.2.3

[CCH87] Y Chang, S Chang, and L Hsu. Automated layout generation using gate matrix approach. In *Proceedings of the 24th Design Automation Conference*, pages 552–58, 1987. 7.2.4

[DGK94] S Devadas, A Ghosh, and K Kuetzer. *Logic Synthesis.* McGraw-Hill Inc., 1994. 2.6.5

[Fis97] Phil D Fisher. Clock Cycle Estimation for Future Microprocessor Generations. Technical report, SEMATECH, 1997. 2.3

[Fur97] M Fury. The early days of CMP. *Solid State Technology*, 40(5):81–2, May 1997. 3.3.5

[G+97] B A Gieseke et al. A 600MHz Superscalar RISC Microprocessor with Out-of-Order Execution. In *Digest of Technical Papers, International Solid State Circuits Conference*, 1997. 2.6.2.3

[Ger99] Gianfranco Gerosa. Member, Intel texas development center. Personal communication, 1999. 3.2.1.2, 3.3.4

[GLL80] R Golden, P Latus, and P Lowy. Design automation and the programmable logic array macro. *IBM Journal of Research and Development*, 24(1):23–31, Jan 1980. 2.6.4, 5.2

[GNBSV98] W Gosti, A Narayan, R Brayton, and A Sangiovanni-Vincentelli. Wire-planning in logic synthesis. In *Proceedings of the International Conference on Computer-Aided Design*, pages 26–33, Nov 1998. 7.2.7

[Gro98] Joel Grodstein. Member, DEC Alpha microprocessor design team. Personal communication, 1998. 1.1, 2.6.2.1

[HFK87] D Hwang, W Fuchs, and S Kang. An efficient approach to gate matrix layout. *IEEE Transactions on Computer-Aided Design*, CAD-6(5):802–9, Sep 1987. 7.2.4

[HMO84] G T Hamachi, R N Mayo, and J K Ousterhout. Magic: A VLSI Layout system. In *21st Design Automation Conference Proceedings*, 1984. 4.2.1, 5.2.3

[JB99] W Jiang and R Brayton. Don't cares and multi-valued logic minimization. *Internal Report, CAD Group, UC Berkeley*, May 1999. 6.3.3

[KAKS97] G Karypis, R Aggarwal, V Kumar, and S Shekhar. Multilevel hypergraph partitioning: Applications in the VLSI domain. In *Proceedings of the 34th Design Automation Conference*, pages 526–9, 1997. 5.4.4

[KK98] G Karypis and V Kumar. *hMETIS : A Hypergraph partitioning package.* University of Minnesota, Dept of Computer Science and Engineering, Nov 1998. Version 1.5.3. 5.4.4

[KSV94] D Kirkpatrick and A Sangiovanni-Vincentelli. Techniques for cross-talk avoidance in the physical design of high-performance digital systems. In *Proceedings of the International Conference on Computer-Aided Design*, pages 616–19, Nov 1994. 2.6.3.2

[KSV96] D Kirkpatrick and A Sangiovanni-Vincentelli. Digital Sensitivity: Predicting signal interaction using functional analysis. In *Proceedings of the International Conference on Computer-Aided Design*, pages 536–41, Nov 1996. 2.6.3.2, 8

[Lia80] Samuel Y Liao. *Microwave Devices and Circuits.* Prentice-Hall, 1980. 3.2.1.2

[McC86] E McCluskey. *Logic design principles : with emphasis on testable semi-custom circuits.* Prentice-Hall, 1986. 2.6.4, 5.2

[MSBSV93] P McGeer, A Saldanha, R Brayton, and A Sangiovanni-Vincentelli. *Logic Synthesis and Optimization*, chapter Delay Models and Exact Timing Analysis, pages 167–189. Kluwer Academic Publishers, 1993. 5.5.2

[Nag95] L Nagel. Spice: A computer program to simulate computer circuits. In *University of California, Berkeley UCB/ERL Memo M520*, May 1995. 1.3, 2.1, 5.2.3, 5.5.2

[ntr97] The National Tecnology Roadmap for Semiconductors. `http://notes.sematech.org/97melec.htm`, 1997. 2.3

[OB98a] R Otten and R Brayton. Planning for performance. In *Proceedings of the Design Automation Conference*, pages 122–27, San Francisco, June 1998. 3.3.7, 7.2.7

[OB98b] R Otten and R Brayton. Planning for performance. In *Proceedings of the Design Automation Conference*, pages 122–127, Jun 1998. 5.8

[PAB+98] S Posluszny, N Aoki, D Boerstler, J Burns, S Dhong, U Ghoshal, P Hofstee, D LaPotin, K Lee, D Meltzer, H Ngo, K Nowka, J Silberman, O Takahashi, and I Vo. Design methodology for a 1.0 ghz microprocessor. In *Proceedings of the International Conference on Computer Design (ICCD)*, pages 17–23, Oct 1998. 2.6.4, 2.6.4, 5.2, 5.2.4

[PW79] S Patil and T Welch. A programmable logic approach for VLSI. *IEEE Transactions on Computers*, c-28(9):594–601, Sep 1979. 7.2.4

[Rab96] J Rabaey. *Digital Integrated Circuits: A Design Perspective.* Prentice Hall Electronics and VLSI Series. Prentice Hall, 1996. 5.3

[RF82] R Rivest and C Fiduccia. A "greedy" channel router. In *Proceedings of the 19th Design Automation Conference*, pages 418–24, 1982. 4.1

[RIXK94] A Rubio, N Itazaki, X Xu, and K Kinoshita. An approack to the analysis and detection of crosstalk faulits in digital VLSI circuits. *IEEE Transactions of Computer-Aided design of integrated circuits and systems*, 13(3):387–95, March 1994. 2.6.3.1, 2.6.3.1

[RSSV84] J Reed, M Santomauro, and A Sangiovanni-Vincentelli. A new gridless channel router: Yet another channel router the second (YACR-II). In *Digest of Technical Papers International Conference on Computer-Aided Design*, 1984. 4.2.1, 5.2.3, 5.5.1

[RSV85] R Rudell and A Sangiovanni-Vincentelli. Espresso-mv: Algorithms for multiple-valued logic minimization. In *Proceedings of the IEEE 1985 Custom Integrated Circuits Conference*, pages 230–4, May 1985. 10

[Sav92] Hamid Savoj. *Don't Cares in Multi-Level Network Optimization.* PhD thesis, University of California Berkeley, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, May 1992. 2.6.3.2, 10, 6.3.3

[SB98] S Sinha and R Brayton. Implementation and use of SPFDs in optimizing boolean networks. In *Proceedings of the International Conference on Computer-Aided Design*, pages 103–10, Nov 1998. 6.1, 6.1, 6.1, 6.1, 6.1, 6.1, 6.2, 6.2.2, 6.2.2, 6.3.1, 10, 10, 6.3.4, 6.4.1, 6.4.2, 6.5

[Sch80] M S Schmookler. Design of large ALUs using multiple PLA macros. *IBM Journal of Research and Development*, 24(1):2–14, Jan 1980. 7.2.2

[SCH82] K Smith, T Carter, and C Hunt. Structured logic design of integrated circuits using the storage/logic array (SLA). *IEEE Transactions on Electron Devices*, ED-29(4):765–76, Apr 1982. 7.2.4

[SK98] D Sylvester and K Keutzer. Getting to the bottom of deep submicron. In *Proceedings of the International Conference on Computer-Aided Design*, 1998. To Appear. 4.2.2, 4.2.2, 4.2.2, 4.2.2

[SKMB90] A. Srinivasan, T. Kam, S. Malik, and R. K. Brayton. Algorithms for Discrete Function Manipulation. In *Proc. of the Intl. Conf. on Computer-Aided Design*, pages 92–95, November 1990. 6.3.2

[spa] Physical Design Modelling and Verification Project (SPACE Project). `http://cas.et.tudelft.nl/research/space/html`. 2.4.1, 5.2.3

[SSL+92] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. SIS: A System for Sequential Circuit Synthesis. Technical Report UCB/ERL M92/41, Electronics Research Laboratory, Univ. of California, Berkeley, CA 94720, May 1992. 4.1, 4.2.1, 4.3.1, 5.2.3, 6.4.1, 6.4.2

[SSV85] Carl Sechen and A Sangiovanni-Vincentelli. The TimberWolf Placement and Routing Package. *IEEE Journal of Solid-State Circuits*, 1985. 4.2.1, 5.2.3, 5.5.1, 6.4.2

[SVSR84] A Sangiovanni-Vincentelli, M Santomauro, and J Reed. A new gridless channel router: Yet Another Channel Router the second (YACR-II). In *Proceedings of the International Conference on Computer-Aided Design*, pages 72–5, Nov 1984. 4.1

[Tri94] S Trimberger. *Field-programmable gate array technology.* Kluwer Academic Publishers, 1994. 2.6.5

[TS86] H Taub and D Schilling. *Digital Integrated electronics.* McGraw-Hill, Inc., 1986. 3.2.1.2

[Wei79] A Weinberger. High-speed programmable logic array adders. *IBM Journal of Research and Development*, 23(2):163–78, Mar 1979. 7.2.2

[WS92] L Wall and R Schwartz. *Programming perl.* O'Reilly and Associates, Inc., 1992. 5.2.3

[YK82] T Yoshimura and E Kuh. Efficient algorithms for channel routing. In *IEEE Transactions on CAD of ICs and Systems*, volume CAD-1, pages 23–35, 1982. 4.1

[YSN96] S Yamashita, H Sawada, and A Nagoya. A new method to express functional permissibilities for LUT based FPGAs and its applications. In *Proceedings of the International Conference on Computer-Aided Design*, pages 254–61, Nov 1996. 6.2

Appendix A

Standard Cells

In this section, we describe the standard cells that were used in our experiments. These were used in the experiments with standard-cell based and Fabric1 based design flows. Figure A.1 shows the details of these cells. All transistors have a channel length of 0.1μm. Individual transistor widths are labelled "W", and are in multiples of 0.1μm.

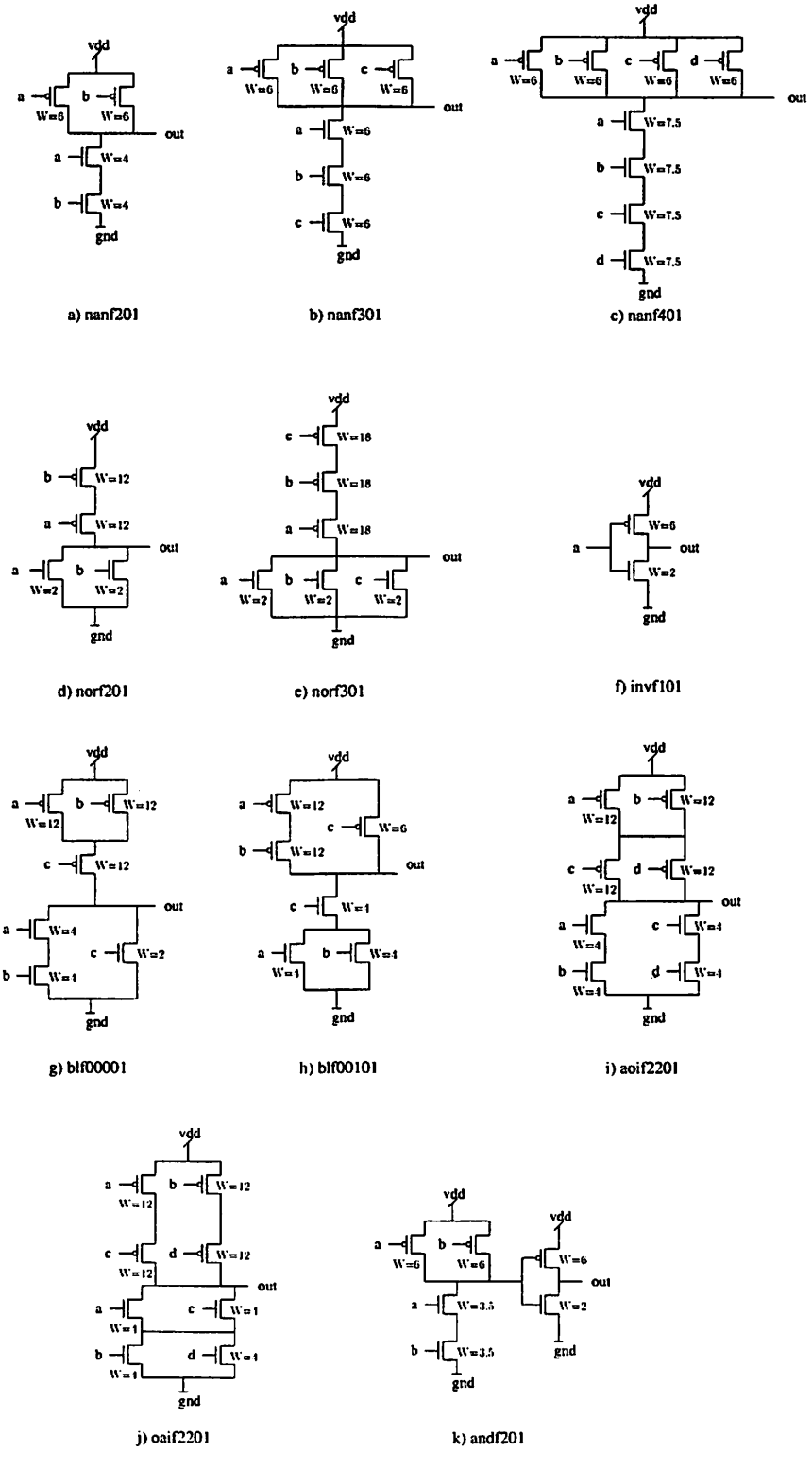


Figure A.1: Standard Cell Library