
Retiming for DSM with Area-Delay Trade-Offs and Delay Constraints

by Abdallah Tabbara

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences,
University of California at Berkeley, in partial satisfaction of the requirements for the
degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor A. Richard Newton
Research Advisor

(Date)

* * * * *

Professor Robert K. Brayton
Second Reader

(Date)

**RETIMING FOR DSM WITH AREA-
DELAY TRADE-OFFS AND DELAY
CONSTRAINTS**

by

Abdallah Tabbara

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of California at Berkeley

1999

ABSTRACT

The concept of improving the timing behavior of a circuit by relocating registers is called retiming and was first presented by Leiserson and Saxe. They showed that the problem of determining an equivalent minimum area (total number of registers) circuit is polynomial-time solvable. In this work, we show how this approach can be reapplied in the Deep Sub-Micron (DSM) domain when area-delay trade-offs and delay constraints are considered. The main result is that the concavity of the trade-off function allows for a casting of this DSM problem into a classical minimum area retiming problem. The solution paves the way for this retiming for DSM approach to be incorporated in a retiming and architectural floorplanning DSM design flow. Some examples are presented and, a new register-based interconnect strategy for DSM that implements the developed retiming technique on global wires is proposed.

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to Professor Robert K. Brayton for initially suggesting this area of research, and to my advisor, Professor A. Richard Newton for his continued support of this research effort. I would also like to thank all the CAD group students for providing a stimulating environment for research, especially the students currently involved in the NexSIS effort: Philip Chong, Mukul Prasad, Subarna Sinha, and William Jiang. In particular, I would like to give special thanks to my brother Bassam who has given great feedback on this and other research endeavors. Thanks also go to my parents, without whom I would not be where I am today.

TABLE OF CONTENTS

Abstract	iii
Acknowledgments	iv
Table of Contents	v
List of Figures	vii
List of Tables	viii
1 Background	1
1.1. Motivation	1
1.1.1. Challenges of DSM	1
1.1.2. Application Domain	4
1.2. New Design Methodology for DSM.....	5
1.2.1. The Domain	5
1.2.2. Design Flow Architecture.....	5
1.3. Retiming Problem Statement.....	7
1.4. Goals and Report Roadmap.....	9
2 Retiming	11
2.1. Leiserson-Saxe Retiming	11
2.1.1. Notation	11
2.1.2. Minimum Area Retiming.....	12
2.2. Modern Techniques	14
2.2.1. Shenoy-Rudell Approach	14
2.2.2. ASTRA and Minaret	15
2.3. The Minimum Cost Network Flow Problem.....	17
3 Minimum Area Retiming with Area-Delay Trade-Offs.....	19
3.1. Transformations.....	19
3.1.1. Granularity	23
3.1.2. Constraints	23
3.1.3. More General Delay Models.....	24
3.2. Algorithm	24
3.2.1. Phase I: Checking Satisfiability	24
3.2.2. Phase II: Minimum Area Retiming.....	25
4 Retiming Software Package	27

4.1. Retiming Package	27
4.2. NexSIS	28
4.2.1. Component Database (Cobase)	28
5 Examples.....	30
5.1. A Simple Retiming Example.....	30
5.2. A Simple SoC Example: The Alpha 21264	32
6 Interconnect Strategy	35
6.1. Pipelined IP Interconnect (PIPE).....	35
6.2. Circuit Implementation	36
6.2.1. Driver/Receiver	36
6.2.2. IP Interconnect Strategy Implementation	36
7 Future Work.....	41
7.1. Integration of Retiming with Architectural Floorplanning	41
7.2. Retiming-Driven Simultaneous Placement and Routing	42
7.3. Interconnect Strategy Evaluation.....	42
8 Conclusions.....	43
References	44

LIST OF FIGURES

Figure 1 DSM Design Flow	6
Figure 2: Retiming Problem statement	9
Figure 3: Transformation at the Graph Level.....	20
Figure 4: Transformation at the Vertex Level.....	22
Figure 5: Database View	29
Figure 6: S27 Retiming Example	31
Figure 7 The Alpha 21264 Floorplan.....	32
Figure 8 The Alpha 21264 Block Diagram.....	34
Figure 9: TSPC Latch (with and without split-output)	37
Figure 10: The basic four TSPC stages.....	37
Figure 11: D Flip-Flop (PN-SN-FL(P))	38
Figure 12: D Flip-Flop (SP-PN-SN).....	39
Figure 13: Interconnect Segment.....	40

LIST OF TABLES

Table 1 The Alpha 21264 Blocks	33
--------------------------------------	----

On the importance of good notation (and representation in general):

By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect increases the mental power of the race. Before the introduction of the Arabic notation, multiplication was difficult, and the division even of integers called into play the highest mathematical faculties. Probably nothing in the modern world would have more astonished a Greek mathematician than to learn that ... a large proportion of the population of Western Europe could perform the operation of division for the largest numbers. This fact would have seemed to him a sheer impossibility ... Our modern power of easy reckoning with decimal fractions is the almost miraculous result of the gradual discovery of a perfect notation.

Alfred North Whitehead

Chapter 1

Background

1.1. Motivation

1.1.1. Challenges of DSM

There has been a lot of debate about the challenges that the EDA industry will face in the DSM era. In this section, a few of the issues that are relevant to this work will be presented.

1.1.1.1. Managing Design Complexity

It is generally agreed, within the EDA community [1], that Deep Sub-Micron (DSM) semiconductor technology is forcing major discontinuities in traditional design methods. The complexity and scale of integration, and the need for greater design productivity, as well as the significant cost of design errors, promote a re-evaluation of design practice. The increasing raw capacity and level of integration is making possible the realization of a complete System-on-Chip (SoC) design. In the near future, a single chip will host a big number of different computational modules (microprocessor, controllers, DSP's, I/O unit, analog circuits, custom hardware, etc...) together with a large amount of (possibly distributed) memory blocks.

For these reasons, and to handle complexity issues associated with DSM, top-down block-oriented design methodologies have been advocated, as a means of reducing design complexity [5]. In such methodologies high-level design and budgeting constraints play an important role where high-

level information is used to externally constrain the design of sub-blocks in terms of the three most important design metrics: performance, area and power.

On the other hand, bottom-up approaches such as component-based design methodologies have also been suggested. The chip-level assembly of pre-characterized intellectual property (IP) modules with specified area-delay-power flexibilities is the goal. Proponents of these methods claim that the required productivity objectives can be met through replication and reuse. Managing concurrency is an essential issue that has to be addressed.

It is conceivable that the best strategy would be to combine both of the aforementioned approaches (i.e. combine pre-designed and characterized modules with different flexibilities along with new modules that serve a special purpose in a given design). We can therefore deduce that design methodologies using distributions (for example area-delay-power trade-offs of components) are becoming more important. System-level integration tools would have to solve complex optimization problems that involve meeting system level constraints provided by the environment while simultaneously balancing different trade-offs and exploring the possibly huge design space.

1.1.1.2. Global Interconnect

In such an IP integration framework, interconnect delays will play an increasingly important role in DSM. As System-on-Chip (SoC) designs and faster clock speeds become a reality, timing and interconnect delays will become more critical. Interconnect effects that impact performance, compromise signal integrity, and increase power dissipation are becoming more pronounced as technology moves deeper into sub-micron feature sizes, and designs are operated at higher frequencies [1][15]. According to NTRS, by 2006 designs will contain over 100M transistors in less than 0.1 μ m

technology with an increase in the number of I/O pins by a factor of 1.5 over present day technologies. More devices on chip and closer packed interconnects with thinner aspect ratios will mean more potential for simultaneous current surges, more coupling events that have the potential of causing false switching or delayed signals, and increased difficulty in determining circuit timing in the design flow [15]. Signal delay and degradation in the deep sub-micron era can have a crippling effect on the performance of designs. Signal preservation cannot be solved as an after thought in the design process. It must be dealt with at an early stage in the flow in order to minimize delay, and accommodate parasitic effects. Increased clock frequencies, larger chip dimensions, and smaller feature sizes (i.e. more device capacity) are making for variable inter-module interconnect lengths, as well as making interconnect delay at the global level dominant over that of devices [16]. While system timing constraints are being met by increased clock speeds, functional timing constraints (i.e. relative timing requirements between module inputs) are becoming harder to satisfy because of this interconnect delay, and the variable wire lengths.

Thus, when global wire delays approach or exceed the global clock period of the design, the delay on some global wires will become lower bounded by an integer number of clock cycles, based on a preselected system-level clock and an initial placement of the modules. The challenge is to use these delay flexibilities along with flexibilities in the implementation of the modules being integrated to improve the design (using the area, delay, and power metrics) while satisfying the pre-specified performance constraints (such as I/O timing constraints).

1.1.2. Application Domain

The target application of this work is the chip level assembly of IP blocks, each under 100k gates in size, either as hard, firm, or soft macros [9]. The design of such a system involves the placement and wireplanning for performance [11] of 200-2000 modules whose average size is 50k gates with a dynamic range of module sizes of 1-500k gates [16]. Modules can be of different types: hard (layout), firm (gate level), or soft (RTL). Such a network has a large number of nets: 40k-100k with 10-100 pins per modules.

This application domain puts limitations on which techniques can be used to address performance and timing issues. In order to manage complexity, components are abstracted where the designer may not be able to massage the function in order to remedy interconnect delay or parasitic problems, but has to work with a given set of implementations with different trade-offs (such as area vs. delay trade-offs).

Industry experts agree that IP integration is the ideal technology for rapid SoC design development in a cost-efficient and fast time-to-market manner. This technology has not been widely adopted yet (full-custom designs are still favored) mostly because many issues still need to be resolved in terms of interfacing these components together. One idea, now finding wide acceptance in the community, is to register-bound IP's, thus temporally decoupling the inside of the block from the outside. This allows IP blocks to be treated as "components" or "black-boxes" that are immune to glitches at the input, and do not generate any at their outputs. It also supports "plug-and-play" capabilities during design where system developers can substitute one black box IP by another, given that they have the same functionality.

1.2. New Design Methodology for DSM

Following is a brief overview of a new design methodology for DSM being developed as part of the NexSIS [9] project. The main thrust of this work i.e. the retiming for DSM approach is currently being incorporated into the NexSIS floorplanning and synthesis environment.

1.2.1. The Domain

The system-level integration framework being advocated supports one-level hierarchy of design. This is considered as the minimum number of levels needed to support reuse, midway between flat (very costly in terms of running time with low probability of convergence) and two-level or higher (may generate solutions that are too far away from the global optimum). The objective of this framework is the placement and routing of 200-2000 modules of average size 50k gates and a dynamic range of 1-500k gates. The types of IP modules that are envisioned are hard (layout), firm (gates + aspect ratio), and soft (RTL). The number of pins for each of those modules is between 10-100 pins, while the number of nets at the global level is between 40k-100k nets. It should be noted that this is a dramatically different problem than the one typically addressed in current design frameworks.

1.2.2. Design Flow Architecture

The aim of this new design flow (which is based on [11]) is to minimize the number of design iterations by supporting two critical guiding notions: planning at the early stages of the flow, and incremental successive refinement during the integration process. One of the main characteristics of this new design flow for DSM is the integration of retiming with architectural floorplanning in order to have a better handle on timing issues.

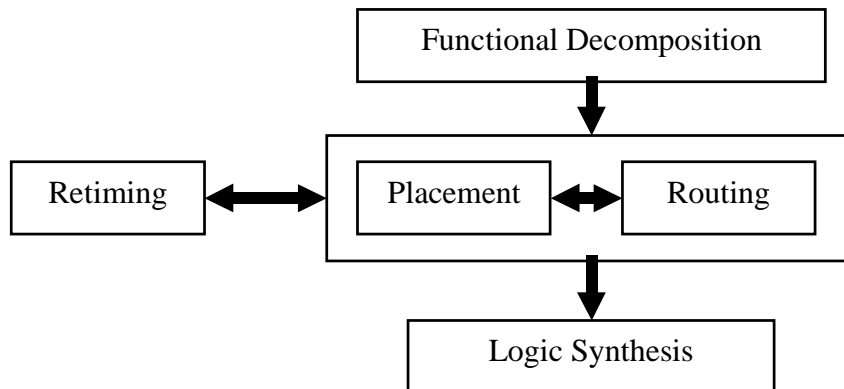


Figure 1 DSM Design Flow

The proposed flow has the following components:

- **Functional Decomposition:** This step provides an entry point for reused IP's, where RTL descriptions may already be well characterized, and area-delay trade-off are taken in as an important performance criterion. The result is a set of modules with some area-delay trade-off estimates
- **Retiming:** This step takes in lower bound timing constraints from placement, and creates upper bound constraints. This is a path-based approach that reduces the area of modules whenever possible. It can be made refinable and incremental, depending on the granularity of the representation.
- **Placement and Routing:** The initial placement and routing step can be a min-cut or any constructive approach. It has to be fast, and gives lower bounds on delays between modules. Subsequent iterations take in upper bounds from retiming as flexibility on placement. This step replaces modules resulting in better lower bound constraints. The main objective is to reduce total chip area, where delays are reduced indirectly.
- **Logic Synthesis:** The main assumption is that the problem can be solved in a predictable manner for a given size module. This step can be

performed in parallel for the different modules and provides better area-delay trade-off estimates for subsequent iterations.

- Iterations: There are two iterations within this flow:
 - Between placement/routing and retiming: This may iterate many times until no further improvements are possible. This step is very similar to initial min-cut partitioning followed by low temperature simulated annealing.
 - Between floorplanning/wireplanning and logic synthesis/layout: This involves only a few iterations, where for each iteration, information is retained through area-delay trade-offs to guarantee convergence.

Note that unlike current design flows, iterations are made incremental, with information from previous iterations being kept around to be used in subsequent iterations.

1.3. Retiming Problem Statement

Retiming is a technique for optimizing sequential circuits. It is a procedure that repositions the registers in a circuit leaving the combinational portion unchanged. The main objective of one form of retiming is to find a circuit with the minimum number of registers for a specified clock period. There are two common variants on this theme:

- a) minimizing the clock period without regard to the number of registers in the final circuit or,
- b) minimizing the number of registers in the final circuit with no constraints on the clock period.

Over a decade and a half have elapsed since Leiserson and Saxe first presented a graph-theoretic formulation to solve this problem for single-clock edge-triggered sequential circuits, where proposed algorithms have

polynomial time complexity. Since then research efforts have focused on incorporating retiming in a synthesis framework, addressing issues that arise due to retiming, and extending the domain of circuits for which retiming can be applied. For digital circuit design, the most useful problem is that of constrained minimum area retiming. One motivation for these algorithms is to examine the area-delay trade-off of the implementation. However, due to the high computational expense of certain forms of this optimization, its use has been limited.

A new "retiming problem" comes up in the non-iterative (i.e. constructive) flow [11] for deep sub-micron applications described above. The problem, which we call minimum area retiming with trade-offs and constraints (MARTC), can be formulated as follows:

Given: a graph $G(V,E)$ (i.e. system-level view), where each node v (i.e. component) has a specified area delay trade-off curve $a_v(d)$, which gives for each number of clock cycles d (i.e. the number of registers retimed into the node v), the area required to perform the computation associated with the node v (i.e. many possible implementations with different latencies). On each edge $e(u,v)$ (i.e. wire), there is an integer $k(e(u,v))$, giving the required number of clock cycles associated with that edge. This lower bound is provided by a current placement of the components using optimally buffered wires. The lower bound states that it is impossible to send a signal over this wire in less clock cycles. To satisfy the edge one needs to put at least $k(e(u,v))$ registers on it. On each edge $e(u,v)$ there is an initial number $w(e(u,v))$ of registers given. These represent the initial latencies allowed between u and v .

Problem: The optimization problem is to find a retiming of the graph G represented by the graph G_r such that:

Minimize:

$$A(G_r) = \sum_v a_v(d)$$

Subject to:

$$w_r(e(u,v)) \geq k(e(u,v))$$

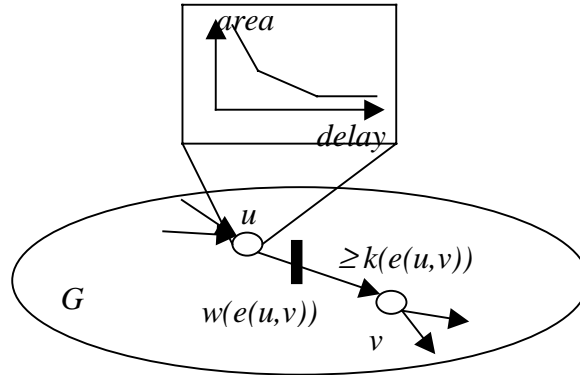


Figure 2: Retiming Problem statement

1.4. Goals and Report Roadmap

In this work, the aim is to incorporate retiming into the chip-level assembly of modules framework that was described in Section 1.2, where in this application domain, communication delays at the chip-level will become dominant. We present a solution to the retiming problem that was stated in Section 1.3 and that will become important in the next few technology generations as discussed earlier.

First, we describe in Chapter 2 previous minimum area retiming approaches, since this problem is very much related to the problem that we are trying to address, and provides the basic building blocks for our solution approach. In Chapter 3, we present a formulation of a solution and then discuss a possible algorithm. We then describe in Chapter 4 the implementation of our solution technique in a dedicated software package

that will be integrated into NexSIS, an IP integration framework and design environment [9]. We then present initial results in Chapter 5, and propose in Chapter 6 a new register-based IP interconnect scheme that implements the registers allocated on global wires.

Chapter 2

Retiming

2.1. Leiserson-Saxe Retiming

In this section, we describe the Leiserson-Saxe retiming approach. In this approach they model a circuit as a graph, and then give polynomial time algorithms for solving different objective functions of the retiming problem.

2.1.1. Notation

A sequential circuit can be represented by a directed graph $G(V,E)$, where each vertex v corresponds to a gate, and directed edge $e(u,v)$ represents a connection from the output of gate u to the input of gate v , passing through zero, or more registers. Each edge has a weight $w(e(u,v))$, which is the initial number of registers between the output of gate u and the input of gate v . Each vertex has a constant delay $d(v)$. A special vertex, the "host", is introduced in the graph, with edges from it to all primary inputs of the circuit, and edges from all primary outputs to the host.

A retiming is a labeling of the vertices $r : V \rightarrow Z$, where Z is the set of integers. The retiming label $r(v)$ for a vertex v represents the number of registers moved from its output towards its inputs. The weight of an edge $e(u,v)$ after retiming, denoted by $w_r(e(u,v))$ is given by

$$w_r(e(u,v)) = w(e(u,v)) + r(v) - r(u)$$

One may define the weight $w(p)$ of any path p originating at vertex u and terminating at vertex v (represented as $p : u \rightarrow v$), as the sum of the weights on the edges of p , and its delay $d(p)$ as the sum of the delays of the vertices on p . A path with $w(p) = 0$ corresponds to a purely combinational path with no registers on it. Therefore, the clock period can be calculated as

$$c = \underset{p, w(p)=0}{\text{MAX}} d(p)$$

Another important concept is that of the W and D matrices which are defined as follows:

$$W(u, v) = \underset{p:u \rightarrow v}{\text{MIN}} w(p)$$

$$D(u, v) = \underset{p, w(p)=W(u,v)}{\text{MAX}} d(p)$$

The matrices are defined for all pairs of vertices (u, v) such that there exists a path $p : u \rightarrow v$ that does not include the host vertex. $W(u, v)$ denotes the minimum latency, in clock cycles, for the data flowing from u to v and $D(u, v)$ gives the maximum delay from u to v for the minimum latency.

2.1.2. Minimum Area Retiming

Let the total number of registers of a circuit after retiming r be denoted by:

$$S(G_r) = \sum_e w_r(e)$$

The minimum area retiming problem can be stated as minimizing $S(G_r)$, subject to timing constraints. But, given the relationship between $w_r(e)$ and $w(e)$ one can rewrite $S(G_r)$ as:

$$\begin{aligned}
S(G_r) &= \sum_e w_r(e) \\
&= \sum_e (w(e) + r(v) - r(u)) \\
&= S(G) + \sum_v (|FI(v)| - |FO(v)|)r(v)
\end{aligned}$$

where $|FI(v)|$ and $|FO(v)|$ are the number of fanins and fanouts of node v . The minimum area retiming problem can then be formulated as the following linear program:

Minimize:

$$\sum_v (|FI(v)| - |FO(v)|)r(v)$$

Subject to:

$$\begin{aligned}
r(u) - r(v) &\leq w(e(u, v)) \\
r(u) - r(v) &\leq W(u, v) - 1, \forall u, v: D(u, v) > c
\end{aligned}$$

The case where not all register costs are the same can be modeled as:

$$\begin{aligned}
S(G_r) &= \sum_e cost(e)w_r(e) \\
&= \sum_e cost(e)(w(e) + r(v) - r(u)) \\
&= S(G) + \sum_v \left(\sum_{e: ? \rightarrow v} cost(e) - \sum_{e: v \rightarrow ?} cost(e) \right) r(v)
\end{aligned}$$

The significance of the objective function and the constraints is as follows (the reader is referred to [7] for details)

- The objective function represents the number of registers added to the retimed circuit in relation to the original circuit
- The first constraint ensures that the weight $w_r(e(u, v))$ for each edge (i.e. the number of registers between the output of gate u and the input of gate v) after retiming is non-negative.

- The second constraint ensures that after retiming, each path whose delay is larger than the clock period has at least one register on it.

This formulation assumes that all registers fanout to exactly one gate.

However, in physical circuits a register can fanout to several gates. Thus registers at the fanouts of a gate can be combined or shared. To accurately model the number of registers in a circuit one needs to take this sharing into account. This can be accomplished by using the model given by Leiserson and Saxe in [7], which introduces for every gate u with multiple fanouts a mirror vertex m_u . The objective function of the linear program can also be modified as described in [7].

It is also pointed out in [7] that the dual of this problem is an instance of a minimum cost network flow problem, hence the problem can be solved efficiently as a minimum cost flow problem.

2.2. Modern Techniques

Although the algorithms mentioned above are polynomial in the circuit size, naive implementations suffer the worst-case ($O(n^3)$ time and $O(n^2)$ space) for all circuits. Recently, algorithms for handling large VLSI circuits were introduced [14][4]. In the following subsections, we list several such algorithms.

2.2.1. Shenoy-Rudell Approach

Shenoy and Rudell [14] describe two major hurdles towards an efficient implementation of minimum area retiming:

- computing the W and D matrices
- solving the minimum cost circulation problem

Their method takes advantage of on-the-fly computations and eliminates the need to completely set up the problem (which can be huge in size) before starting to solve it.

They point out that the method proposed by Leiserson-Saxe to compute the W and D matrices, is an algorithm with $O(V^3)$ time and $O(V^2)$ space complexity even in the best case, because it is based on solving the all-pairs shortest-paths algorithm¹. The advantage in [14] is the space savings obtained by an algorithm that computes in $O(V)$ space and generates a smaller set of constraints.

One of the best known methods to solve the minimum cost circulation problem is a group of cost-scaling techniques. A disadvantage of these techniques is that the graph cannot change during the computations. Consequently, the period edges must be determined before starting the cost-scaling algorithm. Shenoy and Rudell's implementation is based on the generalized cost-scaling framework of Goldberg and Tarjan and has time complexity $O(V|E| \lg |V| \lg |V| C)$ where C is one more than the number of registers in the circuit.

2.2.2. ASTRA and Minaret

The ASTRA algorithm [4] proposed an alternative view of retiming using the equivalence between retiming and clock skew optimization. The Minaret algorithm [8] uses the linear programming formulation and incorporates the ASTRA approach to reduce the number of variables and constraints.

¹ To be fair, in [7] the authors state that, for sparse graphs one could use an $O(V|E| + V^2 \lg V)$ time algorithm which uses the Fibonacci heap data structure due to Fredman and Tarjan, and this could dramatically improve the running time although the space requirements would still be the same.

The introduction of clock skew at a register has an effect similar to moving it across module boundaries (gates). The effect of applying a positive clock skew on a register is equivalent to moving it from the inputs to the outputs. Similarly, application of a negative clock skew is equivalent to moving it from the output to the inputs. Hence both retiming and clock skew are equivalent and can be used for retiming optimization of sequential circuits. Since clock skew is a continuous optimization while retiming is discrete, the minimum clock period achievable by clock skew may not be obtained by retiming. This relationship between skew and retiming motivates the following two-phase solution to the minimum period retiming problem in the ASTRA approach [4].

- Phase A: The clock skew problem is solved to find the optimal values of the skew at each register, with the objective of minimizing the clock period, or to satisfy a given feasible clock period. This involves the (possibly repeated) application of the Bellman-Ford algorithm on a constraint graph.
- Phase B: The skew solution is translated to a retiming by relocating registers across gates in an attempt to set the values of all skews to be as close to zero as possible. The algorithm attempts to reduce the magnitude of all registers' skews by moving each positive skew register opposite to the direction of signal propagation and each negative skew register in the direction of signal propagation.

After Phase B, all skews are forced to zero. This can cause the clock period to increase from Phase A; however, it is shown that this increase will be no greater than the maximum gate delay. The minimum clock period using skews may not be achievable using retiming, since retiming allows cycle-borrowing only in discrete amounts (corresponding to gate delays), while skew is a continuous optimization.

As in the Leiserson-Saxe approach, Minaret [8] also formulates the minimum area retiming problem as a linear program. The ASTRA approach is utilized to obtain reliable bounds on the variables in this linear program. These bounds are then used to reduce the problem size by reducing both the number of variables and the number of constraints. By spending a small amount of additional CPU time on the ASTRA runs, this method leads to significant reductions in the total execution time of the minimum area retiming problem.

2.3. The Minimum Cost Network Flow Problem

The minimum area retiming problem can be reduced to the minimum cost network flow problem [7], by noting that the formulation presented earlier can be recast into a minimum cost flow problem. One can regard each edge $e(u,v)$ as a network flow arc having infinite capacity and cost $w(e(u,v))$ per unit of flow. The dual of the linear programming problem given asks that one assign to each edge a non-negative flow $f(e(u,v))$ such that:

Minimize:

$$\sum_{e(u,v)} w(e(u,v)) f(e(u,v))$$

Subject to:

$$\sum_{v \rightarrow ?} f(e) - \sum_{? \rightarrow v} f(e) = |FO(v)| - |FI(v)|$$

Thus the lags $r(v)$ in the minimum area retiming are the dual variables (potentials) for the optimal flow $f^*(e)$, which most minimum cost flow algorithms compute. The dominant cost in solving the minimum area retiming problem is solving the minimum cost flow problem, for which many algorithms exist. Using the algorithm due to Orlin [10], the problem

can be solved in $O(|E|^2 \lg|V| + |V||E| \lg^2 |V|)$ time. Under the assumption that the largest number of registers on any single edge in the circuit is at most polynomial in $|V|$, the algorithm of Goldberg and Tarjan can be used. Recent work, which is more relevant here, presents an algorithm for handling convex costs [12]. In this work, the authors present a method for extending the work of Orlin [10] to handle convex costs while retaining the strong polynomial properties of the problem.

Although the minimum cost flow problem is of interest in this problem, from a theoretical point of view, the implementation of these algorithms is not very practical in terms of performance [14]. Only the ideas of how to solve the convexity of the costs and the proofs that this can be done, as well as the polynomial complexity proofs, are relevant here. In practice better approaches need to be found to implement the convex costs approach.

Chapter 3

Minimum Area Retiming with Area-Delay Trade-Offs

In this chapter, the solution of MARTC, which is based on the minimum area retiming approach without clocking constraints, is presented. Algorithms for solving the resulting problem are then discussed.

3.1. Transformations

In order to cast the problem at hand as a minimum area retiming problem, we first state some assumptions:

- The area-delay trade-off curves are monotone decreasing and convex: the slope of the curve decreases less rapidly as the delay increases (i.e. the second derivative is positive). Without this assumption the problem could possibly become NP-hard, since the exploration of all possible combinations of trade-offs at the different nodes would be required (i.e. a combinatorial problem)
- Delays at the nodes are irrelevant. This is because we assume that the wire delays being considered are much larger than the delays at the nodes and the clock period. This seems to be a valid assumption in the DSM regime [11][16].
- The clock period is the time granularity in this problem. As clocking speeds increase, this fact will become more valid.

We now transform the problem at hand into a minimum area retiming problem. The nodes are split to form one edge, which can now contain registers. These will represent registers retimed into the node in order to

increase its delay but decrease its area. This problem matches closely the original Leiserson-Saxe formulation of minimum area retiming (i.e. without clock constraints), except that the function being optimized is area instead of the number of registers and is convex instead of being linear. Thus the mathematical programming problem is:

Minimize:

$$A(G_r) = \sum_e a_e(w_r(e(u,v)))$$

Subject to:

$$w_r(e(u,v)) = w(e(u,v)) + r(v) - r(u) \geq k(e(u,v))$$

$$\Rightarrow r(u) - r(v) \leq w(e(u,v)) - k(e(u,v))$$

where a_e represents the area-delay trade-off convex function for the node associated with that edge. The constraint represents the transformed non-negativity (lower bound) constraint on the number of registers on certain edges. Initially, registers can be positioned anywhere within the circuit.

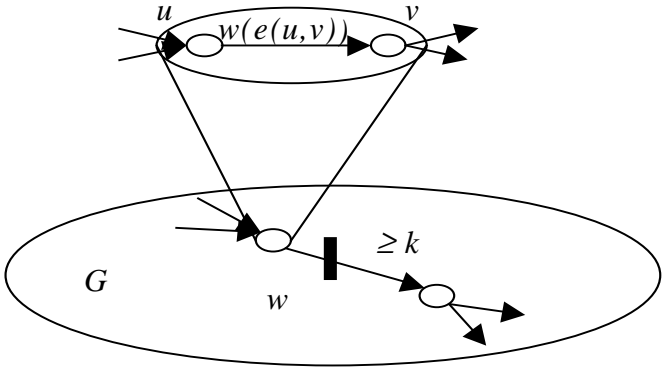


Figure 3: Transformation at the Graph Level

Using the approach of [12] for transforming piecewise-linear convex costs into additional edges and vertices of linear costs and keeping in mind the duality of the two problems, one can transform this problem into an exact

minimum area retiming problem. This is accomplished by splitting nodes in the original graph not by one edge but by several edges, one for each line segment of the trade-off curve. The equation for the total area can then be rewritten as:

$$\begin{aligned}
A(G_r) &= \sum_e a_e(w_r(e(u,v))) \\
&= \sum_e a_e(w(e(u,v))) + \sum_e \sum_l \text{slope}(l)(r(v_l) - r(u_l)) \\
&= A(G) + \sum_e \sum_l \text{slope}(l)(r(v_l) - r(u_l)) \\
&= A(G) + \sum_e \sum_v (\text{slope}(v_p \rightarrow v) - \text{slope}(v \rightarrow v_s))r(v)
\end{aligned}$$

where v_p and v_s are the predecessor and successor of v respectively. Thus each linear piece is converted to a corresponding arc with cost equal to its slope, and weight constrained by its projected length on the delay axis.

$$\begin{aligned}
\text{cost}(e(u,v)) &= \text{slope}(l) \\
0 &\leq w(e(u,v)) \leq \text{width}(l)
\end{aligned}$$

In general, there are two possible types of constraints:

$$\begin{aligned}
r(u) - r(v) &\leq w(e(u,v)) - w_l(e(u,v)) \\
r(v) - r(u) &\leq w_u(e(u,v)) - w(e(u,v))
\end{aligned}$$

where w_l and w_u denote the upper and lower bound on the weights respectively. For created edges $w_l = 0$ while for original edge $w_u = \infty$.

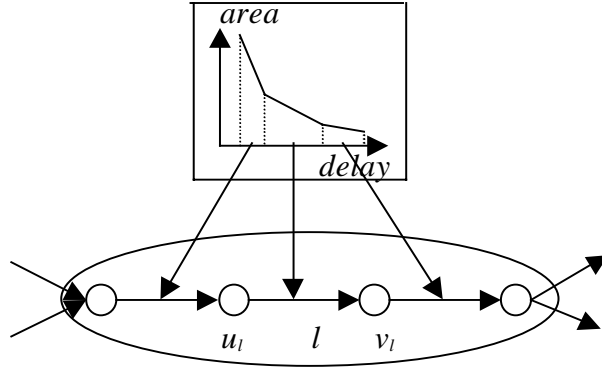


Figure 4: Transformation at the Vertex Level

Since it is guaranteed that registers on segments with lower cost (more negative slope) will be preferred, the resulting optimal retiming will always be correct in terms of filling edges with bigger area reductions first. This is due to the concavity of the area-delay trade-off curves.

Lemma 1: Given two consecutive segments, l_1 and l_2 ($slope(l_1) < slope(l_2)$), in the split node then it is always the case that l_1 is completely filled ($w_r(l_1) = width(l_1)$) whenever l_2 gets filled ($w_r(l_2) > 0$), in the minimum retiming solution.

Proof: Assume that a minimum solution such that the lemma is false ($w_r(l_1) < width(l_1)$ whenever $w_r(l_2) > 0$). We know that the coefficient of the retiming function at internal nodes is negative ($slope(l_1) - slope(l_2) < 0$) because the trade-off delay curve is convex. Since node retimings have negative coefficients for the internal nodes of the split node one can move registers from l_2 to l_1 across a node v ($r(v)$ is increased). This will reduce the cost of the solution even further ($A(G_r)$ is decreased) because it corresponds to a positive retiming of the node v in between. This can be done until the segment l_1 is filled completely ($w_r(l_1) = width(l_1)$) and no further

improvements are possible, proving that the given solution is not a minimum.

Theorem 1: The vertex level transformation is correct in the sense that a solution to the linear program is a minimum area solution for the original problem.

Proof: The lemma shows that no further minimum area retiming is possible at the vertex level. Thus, no special care has to be taken in solving the linear program since the resulting solution is guaranteed to be a minimum solution of the original problem.

In the following, we turn the reader's attention to the following important points about this description method.

3.1.1. Granularity

As with classical retiming approaches, the view here is that the graph represents a network of functional elements and globally clocked registers. The model is not concerned with the level of complexity of the functional elements, so the granularity of the problem can be controlled by the description. This fact can be used to control the complexity and level of optimization required in solving this problem.

3.1.2. Constraints

The constraints presented at each edge are independent and are set at the functional decomposition phase of the IP integration flow. For added edges, they represent the different possible ranges of implementations for a given module, while for original edges, they represent the lower bound constraints determined by placement of the modules. Note that it is possible to describe modules whose implementation has a delay which is greater than one global

clock cycle, and this is done by having lower bound constraint on added edges used to represent area-delay trade-offs of modules.

3.1.3. More General Delay Models

As described in [7] it is possible to extend the methods described in this section to deal with functional elements in which the propagation delay through individual functional elements are non-uniform. That work, with minor modifications, generalizes the original circuit model to handle this situation, and shows how the retiming problem can be reapplied to this more general case.

3.2. Algorithm

The algorithm is a two-phase process. The first phase involves checking the satisfiability of constraints or deriving constraints if none are given. In the second phase, after constraints have been handled, the minimization process begins resulting in a minimum area retiming of the given circuit.

3.2.1. Phase I: Checking Satisfiability

Checking feasibility can be easily performed on the transformed graph as a constraint problem. Constraints are given by:

$$\begin{aligned} r(u) - r(v) &\leq w(e(u, v)) - w_l(e(u, v)) = r_u(u, v) \\ r(v) - r(u) &\leq w_u(e(u, v)) - w(e(u, v)) = -r_l(u, v) \end{aligned}$$

From these constraints we set up a weight matrix R where the entry $R(u, v)$ represents the upper bound $r_u(u, v)$ on the difference $r(u) - r(v)$, while $R(v, u)$ represents the lower bound $r_l(u, v)$. This constraint matrix R is a difference bound matrix (DBM) [12], where there is no need to specify the tightness of the constraints (i.e. a flag in each entry of the matrix) since all

are tight. Satisfiability of given constraints can be determined using a classical all-pairs-shortest-path computation on this DBM [12].

In order to derive constraints, we apply an all-pairs-shortest-path approach to convert the DBM into canonical form [12], which represents tight constraints on the retimings. We then derive upper and lower bounds on the number of registers using the equations:

$$\begin{aligned}w_l(e(u, v)) &= w(e(u, v)) - r_u(u, v) \\w_u(e(u, v)) &= w(e(u, v)) - r_l(u, v)\end{aligned}$$

3.2.2. Phase II: Minimum Area Retiming

Now that the problem has been cast as a minimum area retiming problem with no cycle time constraint, it is easy to see that the problem can be solved using a linear programming approach originally suggested by Leiserson and Saxe. Many approaches have been described above on how to do this in an efficient manner.

One of the first approaches [7] is a minimum cost flow dual formulation which can be used to derive the optimal flow and then the optimal potential which directly translates into retimings at the nodes.

Using the improved techniques [14] [8] discussed earlier, one can optimize the space requirement and reduce the number of constraints and variables required thus making the problem more manageable.

Yet another method which can be used, which in some cases may not be efficient, is a relaxation-based approach. In this approach, the information derived from the slacks computed in the first phase can be used to decide where to put the registers on the edges with the most negative cost. Then new slacks are derived for the subgraphs, until the minimum area solution is reached.

The algorithm has the following steps:

- consider the area-delay trade-off curve for each node in the original graph
- split the node accordingly by representing each segment, where an edge corresponding to a segment has
 - a cost equal to slope of the segment
 - a lower and upper bound derived from the length of the segment on the time axis.
- solve the minimum area retiming of the resulting problem
- translate the solution into a retiming of the original graph

As described earlier, the construction will result in the correct retiming at the nodes and this can be attributed to the special form of the node splitting and the convex piece-wise linear area-delay trade-off curves.

Chapter 4

Retiming Software Package

Initially, our implementation approach was to reuse already available capabilities in the retime package in SIS [13]. In the following, we first give a description of what were the modifications needed to make the package work for the new problem. We then describe how a new implementation of this software package will be integrated into NexSIS, a new IP integration SoC framework currently being developed.

4.1. Retiming Package

The retime package contains an implementation of the two types of retiming using the Leiserson-Saxe approach described earlier. We concentrate here on describing the minimum area capabilities. Modifications done on the retime package to handle the new problem are as follows:

- building of the retiming graph was changed
- splitting of nodes to handle piece-wise linear area-delay trade-off segments was added
- data about weights and area-delay trade-off curve is externally specified and read in.
- no register sharing is considered
- W and D matrix are not relevant in this formulation and so are not computed
- clocking constraints are not added to the constraints matrix because they are not relevant

- Phase I: checking/deriving external timing constraints
- Phase II: the resulting linear program is solved using the Simplex approach

4.2. NexSIS

This is a very brief description of some of the initial work that has been done so far to create the kernel of a new IP integration SoC framework.

4.2.1. Component Database (Cobase)

The first part of the implementation has been focused on designing a new data structure able to hold design useful information about an SoC design. The characteristics that were deemed necessary for such a database are:

- hierarchical descriptions
- representation at different abstraction levels
- a focal point for the integration of different design tools into flows

The main parts of the description, which were modeled after previous design approaches namely OCT are:

- **Component:** This the basic unit of description in the database. A component can have descriptions at many different abstraction level by different tools. To basic components in this description are:
 - **Module:** will represent the different IP blocks being used, e.g.
 - **Net:** will represent wiring information. This connectivity information can be point to point or bus structures.
- **View:** represents a given abstraction level description of a component. At this point in time only the floorplanning view is of interest
 - **FloorplanView:** provides a very high level description of an SoC

- Model: a tool representation at a different abstraction level. Two special types of models available at all abstraction levels are:
 - ContentsModel: provides instantiation information
 - InterfaceModel: provides connectivity information

In Figure 5 is a view of the database representation of the Alpha 21264 microprocessor.

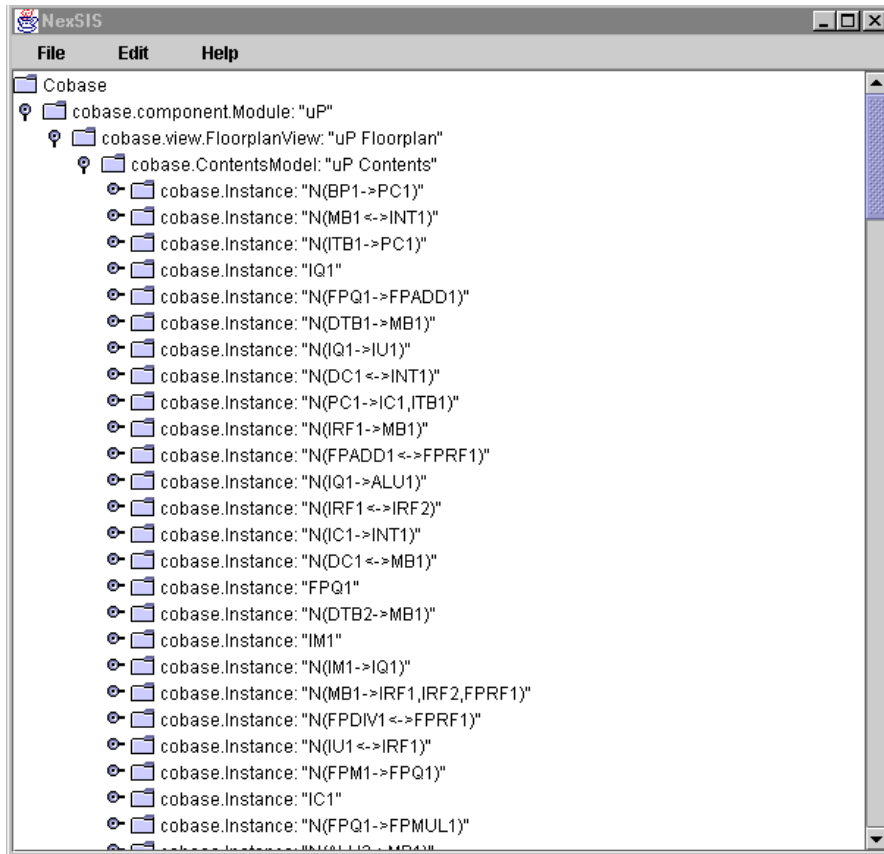


Figure 5: Database View

Chapter 5

Examples

There is currently work being done on creating benchmarks for the SoC integration domain. In this chapter, two examples that are currently being used to evaluate the different parts of the flow are presented.

5.1. A Simple Retiming Example

An example circuit, presented here is S27 derived from the ISCAS89 benchmark, which was used in [8]. The example was chosen to be small in order to show some of the aspects of the algorithm being used.

The retime graph has 17 edges and 8 nodes (the one first built by SIS from the original circuit). For convenience, the area-delay trade-off curve was the same for all nodes, but this doesn't affect the performance or correctness of the algorithm. Only the maximum number of segments of these curves affects the complexity of the algorithm since the number of constraints required to handle the splitting of nodes is $|E| + 2k|V|$ where k is the maximum number of segments.

The number of registers was not changed from the original circuit specification. The example shows an interesting case of retiming, and what could possibly occur during this process:

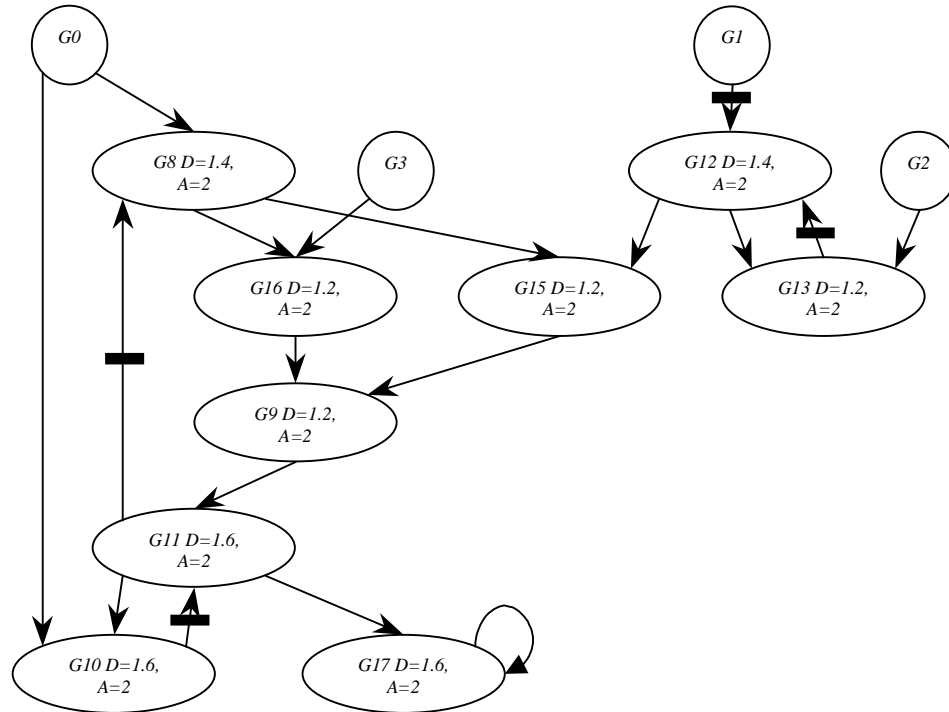


Figure 6: S27 Retiming Example

The results of retiming of this graph are:

- The register between $G8$ and $G11$ could not be moved because of the restrictions of correct retiming, even though a possible decrease in area would result.
- The register before $G12$ was moved into $G12$ to minimize the area of that node. It may seem that a combinational cycle between $G12$ and $G13$ has been introduced, but this is not so since there is a delay within $G12$.
- The register in $G12$ was not moved into $G13$ and $G15$ even though it decreases area because of correct retiming constraints.
- The register after $G10$ was moved back into it. It was not possible to move this register forward into $G11$ because this results again in incorrect retiming.

Thus, the resulting circuit has minimum area within the constraints of the trade-off function and correct retiming.

5.2. A Simple SoC Example: The Alpha 21264

In order to get a better feel for SoC design we collected information about the Alpha 21264 microprocessor. This simple example has served as an initial driver for designing the kernel for NexSIS (see Figure 5). In Figure 7 we show the floorplan of this complex microprocessor.

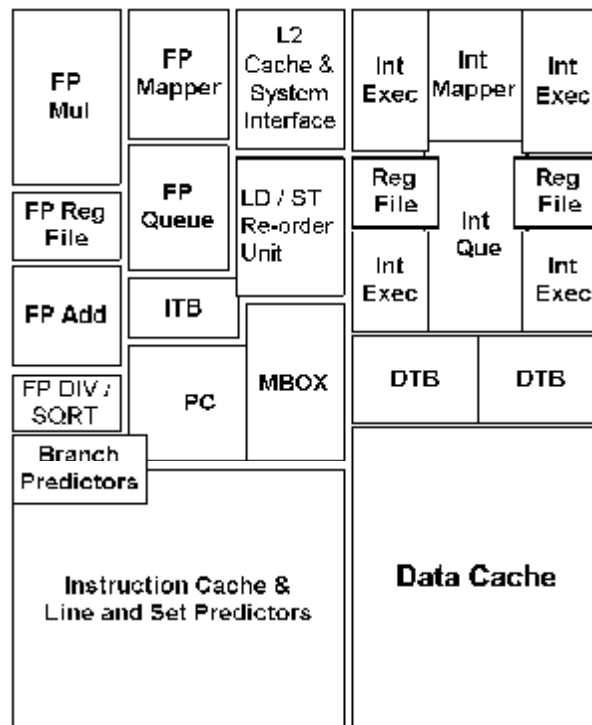


Figure 7 The Alpha 21264 Floorplan

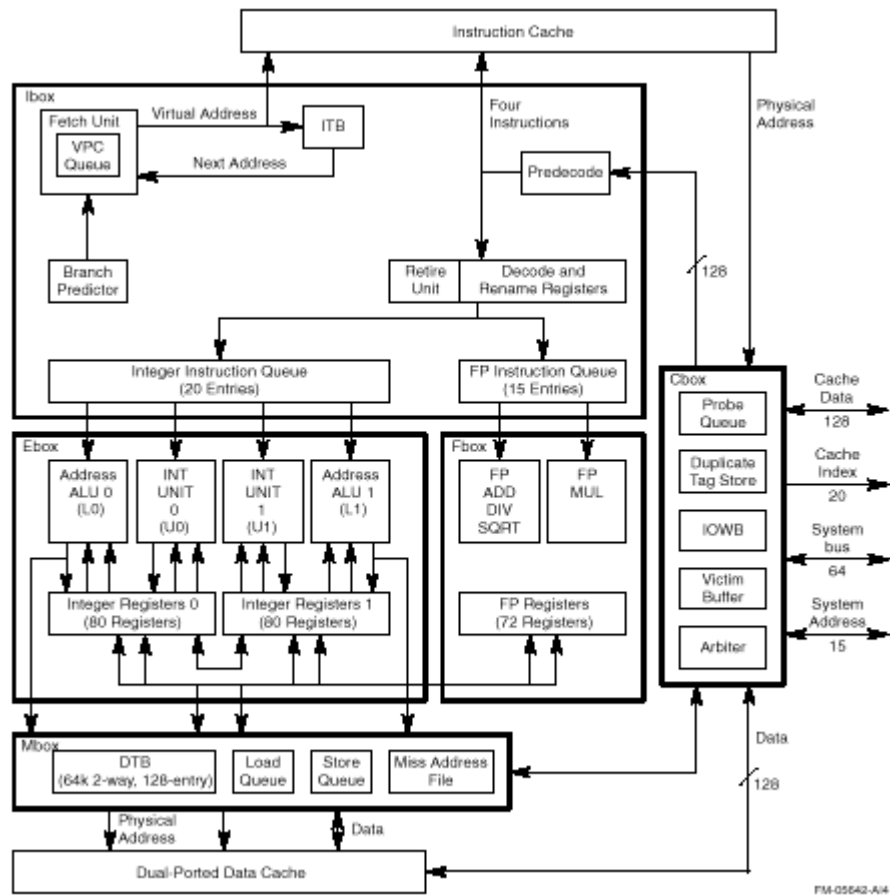
Analysis of the information available in this floorplan and using the fact that it is to scale, we come up with the data about the design presented in

Table 1. Note that the granularity of the design is the same as the one described earlier when discussing the application domain of SoC design.

<i>Unit</i>	<i>#</i>	<i>Aspect Ratio</i>	<i># Transistors</i>
<i>Instruction cache</i>	<i>1</i>	<i>0.73</i>	<i>2.9M</i>
<i>ITB</i>	<i>1</i>	<i>0.56</i>	<i>284k</i>
<i>PC</i>	<i>1</i>	<i>0.91</i>	<i>488k</i>
<i>Branch Predictor</i>	<i>1</i>	<i>0.53</i>	<i>337k</i>
<i>Data cache</i>	<i>1</i>	<i>0.82</i>	<i>2.8M</i>
<i>DTB</i>	<i>2</i>	<i>0.74</i>	<i>419k</i>
<i>MBox</i>	<i>1</i>	<i>0.61</i>	<i>586k</i>
<i>LD/ST Reorder Unit</i>	<i>1</i>	<i>0.78</i>	<i>612k</i>
<i>L2 Cache/System IO</i>	<i>1</i>	<i>0.79</i>	<i>596k</i>
<i>Integer Exec</i>	<i>2</i>	<i>0.75</i>	<i>290k</i>
	<i>2</i>	<i>0.54</i>	<i>404k</i>
<i>Integer Queue</i>	<i>1</i>	<i>0.5</i>	<i>617k</i>
<i>Integer Reg File</i>	<i>2</i>	<i>0.91</i>	<i>217k</i>
<i>Integer Mapper</i>	<i>1</i>	<i>0.71</i>	<i>432k</i>
<i>FP div/sort</i>	<i>1</i>	<i>0.57</i>	<i>252K</i>
<i>FP add</i>	<i>1</i>	<i>0.97</i>	<i>429k</i>
<i>FP Queue</i>	<i>1</i>	<i>0.81</i>	<i>515k</i>
<i>FP Reg File</i>	<i>1</i>	<i>0.67</i>	<i>296k</i>
<i>FP Mapper</i>	<i>1</i>	<i>0.81</i>	<i>515k</i>
<i>FP mul</i>	<i>1</i>	<i>0.61</i>	<i>725k</i>
<i>uP</i>	<i>24</i>	<i>0.81</i>	<i>15.2M</i>

Table 1 The Alpha 21264 Blocks

The block diagram presented in Figure 8 generates a network of modules description (Figure 5) of this SoC. More data needs to be collected about this and other examples to make them more realistic benchmarks for SoC intergration.



FM-05542-A14

Figure 8 The Alpha 21264 Block Diagram

Chapter 6

Interconnect Strategy

Retiming [17] and placement techniques [11] of modules will be able to satisfy delay constraints as long as interconnect delay is within bounds and can be properly characterized and predicted. To this end and to remedy the global wire delay problem, this chapter focuses on suggesting interconnect solutions, when the registers retimed onto wires can not be absorbed by reassigning wires to slower metal layers. We have devised a registered interconnect strategy for the IP integration component-based design SoC domain. This strategy is presented here.

6.1. Pipelined IP Interconnect (PIPE)

In DSM, global wires contribute significantly to the delay of a circuit and therefore need to be “retimed” in order to satisfy the functional timing constraint requirements. The idea here is to insert registers (i.e. pipelining) within the (register bounded) global interconnect wires in order to reduce “perceived” delays thus permitting modules to meet constraints on the relative timing of inputs. We propose to use registers that have the following characteristics:

- high performance,
- minimum area impact because of the large number of module pins, and
- low clock loading (to minimize clock distribution problems),
- small delay, and
- low power consumption

6.2. Circuit Implementation

The focus of this section is the implementation aspects of the pipelined interconnect strategy, and the exploration of the different implementation choices.

6.2.1. Driver/Receiver

As stated earlier, the requirement here is for registers to be present at the boundaries of the IP blocks. This is a desirable requirement as explained before (i.e. straightforward synchronous integration), and can be easily satisfied by design. Therefore, in order to minimize crosstalk effects and glitching, static (or pseudo-static) high-speed (edge-triggered) registers should be used at the IP boundaries. In addition, the driver should be able to support the required fanout. These are the guiding metrics for the driver/receiver design choices. We leave the choice of registers at the IP boundaries to the designer and assume standard CMOS line drivers. We focus next on the circuit implementation of the registers needed to support our proposed pipelined interconnect strategy.

6.2.2. IP Interconnect Strategy Implementation

6.2.2.1. True Single Phase Clock (TSPC) Latches

TSPC latches are commonly used in high performance digital systems due to their simplicity and fast operation [6]. The advantages of this choice are: the single clock phase which avoids clock overlap problems, and the low clock loading (1 NMOS gate in the case of split-output TSPC). Figure 9 shows the TSPC latch, and the split-output TSPC latch.

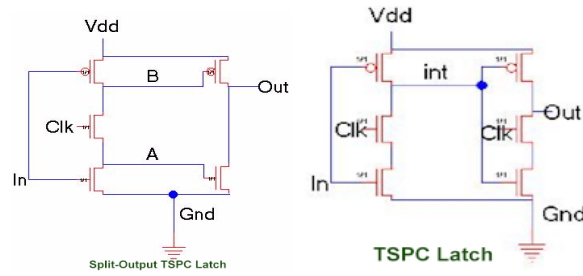


Figure 9: TSPC Latch (with and without split-output)

The split-output latch however not only has lower performance because of the threshold drop on the clocked NMOS: the presence of two interconnect wires internally increases the susceptibility to crosstalk effects between the lines marked “A” and “B” in Figure 9. Because of this, the split-output solutions will not be considered in the sequel even though they have half the clock loading of the regular TSPC latch.

We can generalize the above TSPC latch and recognize that there are four basic stages in TSPC latch or register design as shown in Figure 10 [19]

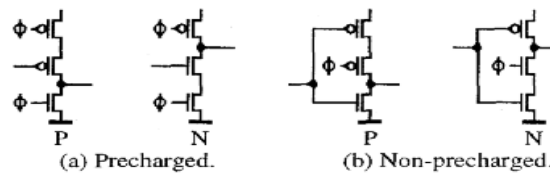


Figure 10: The basic four TSPC stages

A p-latch consists of two p-stages while an n-latch consists of two n-stages. A precharged latch is formed by a precharged stage followed by a non-precharged stage. A non-precharge latch is formed by two non-precharge stages. Registers are formed by the combination of these latches.

6.2.2.2. C^2 MOS-like Full Latch

The PN-SN-Full Latch (P)-INV register in Figure 11 has a C^2 MOS-like stage at the end to utilize the available precharged-node signal. This register is data-dependent during its evaluation phase but works perfectly for both one and zero inputs [20].

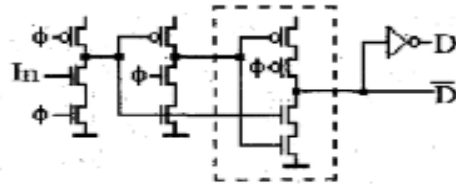


Figure 11: D Flip-Flop (PN-SN-FL(P))

6.2.2.3. Pipelines

Using the TSPC basic stages shown earlier we have devised several interconnect pipeline schemes. We present these configurations here, and leave the evaluation of these schemes for future work.

For each single stage block we can have the combinations listed below. The notation below stands for the following:

- SN = Static N
- PN= Precharged N
- SP = Static P
- PP = Precharged P
- “-” = Delimiter between half stages
- Full Latch = C^2 MOS NORA stage

We have identified four basic schemes for implementing a positive edge register, and these are:

- 1) SP-PN-SN (This is a D Flip Flop (DFF) as in Figure 12)

- 2) PP-SP-Full Latch (N)
- 3) SP-SP-SN-SN
- 4) PP-SP-PN-SN

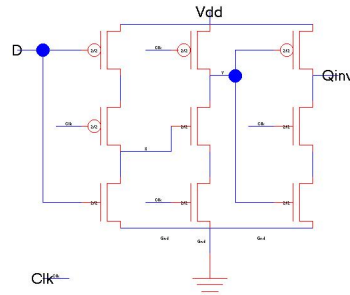


Figure 12: D Flip-Flop (SP-PN-SN)

These four schemes can each be implemented as:

- lumped: optimally buffered wire segments surrounded by registers (see Figure 13), or
- distributed: registers split up into stages which replace the buffers.

Each interconnect segment consists of four wire segments of length l_{crit} (at m4 in 0.35 μ m technology l_{crit} is $\sim 600\lambda = 600 \cdot 0.35\mu\text{m} = 210\mu\text{m} = 0.21\text{mm}$, where delay on a wire of l_{crit} length is $\sim 250\text{ps}$). Given the prediction for DSM designs, we picked a global clock of 500 MHz (intra-module clocks are projected to be one order of magnitude faster or $\sim 5\text{GHz}$) which amounts to $\sim 4 l_{crit}$'s (about $\sim 0.84\text{mm}$). Given a $\sim 16\text{mm}$ chip dimension (at 0.35 μ m technology), we see that ~ 20 clock cycles to cross the chip is conceivable, as pointed out by Dally [3]. Note that the 4 l_{crit} and 20 global clock cycle estimate also holds for smaller scale technologies as pointed out earlier. Interconnect was modeled as optimally buffered HSPICE lossy transmission lines.

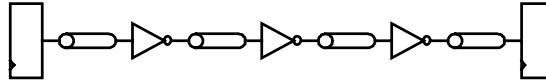


Figure 13: Interconnect Segment

The schemes described above can be evaluated with or without coupling to account for crosstalk. Initial results [18] show that these possible solutions provide a wide range of implementations that can potentially be used in a trade-off optimization setting, just as was done in the case of modules.

Chapter 7

Future Work

There is still a lot of work to be done to make the whole flow described in Section 1.2 a reality. Here we outline a few of the tasks that are currently being focused on.

7.1. Integration of Retiming with Architectural Floorplanning

The conventional VLSI design flow consists of an integration of various steps and tools in order to synthesize a circuit. Typically, the flow introduces a separation between the logic synthesis step and the physical design step. For designs with aggressive performance goals, this division entails hundreds of iterations between synthesis and physical design before converging to the desired implementation and achieving closure on design constraints, especially timing. In this work we plan to address this issue and lay the groundwork for a new design flow that exploits (a) the recently proposed idea of planning for performance [11] at the early stages of the flow in order to minimize design iteration, (b) recent work in the area of retiming [17], and (c) results of a recent study of delays in DSM [16]. The retiming work promises to permit area and delay trade-off decisions at the early stages of floor planning. In this work, we will introduce retiming into the architectural floorplanning stage, and show how it has the potential to address and provide solutions to timing closure issues in deep sub-micron [16].

7.2. Retiming-Driven Simultaneous Placement and Routing

The rapid scaling of IC technology leads to smaller devices, closer interconnects and higher degrees of on chip integration, which requires a stronger interaction between the high level synthesis and the low level physical design. Based on the idea of planning for performance [11], a new DSM design methodology has been proposed in Section 1.2, where retiming explores the area delay tradeoffs at a floorplanning level, combining synthesis and physical design and minimizing the number of iterations needed. Our proposed future research work in retiming-driven simultaneous placement and routing is to: (a) integrate module placement and routing within the same data structure, (b) search a feasible place/route solution that satisfies the constraints prescribed by retiming, and (c) incrementally change (or update) the solution within a reasonable number of iterations. Optimization algorithms and schemes to be explored need to be scalable as well as able to handle the size of problem envisioned in this application domain.

7.3. Interconnect Strategy Evaluation

In Chapter 6 we have proposed a new interconnect scheme for the IP integration framework. We outlined several domain requirements on the suitable interconnect solution, and then proceeded to present a pipelined interconnect strategy using TSPC registers. Our initial studies [18] show that this scheme has the potential to perform well with respect to area, delay, and power. In future work, we will evaluate the different trade-offs of the proposed TSPC-based pipelined schemes. This will be done by building an adequate test bench in order to evaluate using layout, modeling and simulation to measure performance, area, and power requirements.

Chapter 8

Conclusions

In this work we showed how the minimum area retiming approach can be reapplied to a problem arising in DSM, where area-delay trade-offs and delay constraints are considered. We presented a framework to cast that problem into a classical minimum area retiming problem with no cycle constraints. Results show the correctness of this approach, but in cases where the area-delay trade-off has many segments, the number of constraints may have to be reduced using available methods. Our initial implementation in SIS has not addressed efficiency issues, just feasibility. However, we should be able to apply many of the techniques in the literature used to make retiming efficient. This will be left to the implementation currently being developed, to be incorporated in the NexSIS framework.

REFERENCES

- [1] "National Technology Roadmap for Semiconductors", *Semiconductor Industry Association*, 4300 Stevens Creek Blvd., Suite 271, San Jose, CA 95129.
- [2] Alur R., "Timed Automata", *NATO-ASI Summer School on Verification of Digital and Hybrid Systems*, 1998.
- [3] Dally, W., Poulton, J. "Digital Systems Engineering", Cambridge University Press, 1998.
- [4] Deokar R.B., and Sapatnekar S.S., "A Fresh Look at Retiming via Clock Skew Optimization", *DAC* pp. 310-315, 1995.
- [5] Eory F. "Systems to Silicon Design: Methodology for Deep Sub-micron ASICs", *SuperCon*, 1997.
- [6] Ji-Ren Y., Karlsson I., Svensson C. "A True Single-Phase-Clock Dynamic CMOS Circuit Technique", *IEEE JSSC*, 1987.
- [7] Leiserson C.E., and Saxe J.B., "Retiming Synchronous Circuitry", *Algorithmica*, vol. 6, pp. 5-35, 1991.
- [8] Maheshwari N., and Sapatnekar S.S., "An Improved Algorithm for Minimum-Area Retiming", *DAC*, 1997.
- [9] NexSIS CAD Group, University of California at Berkeley, Principal Investigators: Robert K. Brayton, A. Richard Newton, and Kurt Keutzer.
- [10] Orlin J.B., "A Faster Strongly Polynomial Minimum Cost Flow Algorithm", *Operations Research*, vol.41, no.2, pp. 338-50, 1993.
- [11] Otten R.H.J.M., and Brayton R.K. "Planning for Performance", *DAC*, 1998.
- [12] Pinto Y., Shamir R., "Efficient Algorithms for Minimum-Cost Flow Problems with Piecewise-Linear Convex Costs", *Algorithmica*, vol.11, pp. 256-277, 1994.
- [13] Sentovich E., "Sequential Circuit Synthesis at the Gate Level", *Ph.D. Thesis, UC Berkeley*, Chap. 5, 1993.
- [14] Shenoy N. and Rudell R., "Efficient Implementation of Retiming", *ICCAD*, pp. 226-233, 1994.

- [15] SRC Physical Design Task Force: Ray Abrishami, LSI Logic; Eli Chiprout, IBM Corporation, Mike Jassowski, Intel Corporation; Bernd Koenemann, Logic Vision Corporation; ChiCheng Liaw, Lucent Technologies; Sury Maturi, National Semiconductor; Ching-Hao Shaw, Texas Instruments Incorporated, "Draft Report on Physical Design", May 1997.
- [16] Sylvester D., Keutzer K., "Getting to the Bottom of Deep Submicron", *ICCAD*, 1998.
- [17] Tabbara A., Brayton R.K., Newton A.R., "Retiming for DSM with Area-Delay Trade-Offs and Delay Constraints", *DAC*, 1999.
- [18] Tabbara A., Tabbara B., "Inter-Module Interconnect Strategy for System-on-Chip Applications", *EE241 Project Report*, 1999.
- [19] Yuan J., Svensson C., "Fast and Robust CMOS Double Pipeline Using New TSPC Multiplexer and Demultiplexer", International Conference on ASIC, 1996.
- [20] Yuan J., Svensson C., "New Single-Clock CMOS Latches and Flipflops with Improved Speed and Power Savings", IEEE JSSC, 1997.