ELSEVIER

INTEGRATION, the VLSI journal 29 (2000) 25–43

INTEGRATION
the VLSI journal

www.elsevier.com/locate/vlsi

Integration of retiming with architectural floorplanning

Abdallah Tabbara, Bassam Tabbara, Robert K. Brayton*, A. Richard Newton

Department of Electrical Engineering, University of California, P.O. Box 218, Cory Hall, Berkeley, CA 94720, USA

Received 1 October 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* 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 casting this problem into a classical minimum area retiming problem. The solution paves the way for retiming to be incorporated in the architectural floorplanning stage of a design flow tailored for deep sub-micron circuits. Some examples and a register-based interconnect strategy suitable to the developed retiming technique on global wires is presented.

1. Introduction

1.1. Managing design complexity

It is generally agreed within the design automation community 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 enabling the realization of a complete system-on-chip (SoC) design. In the near future, a single chip will host many different computational modules (such as controllers, microprocessor, DSP's, I/O units, analog circuitry, custom hardware), together with a large amount of (possibly distributed) memory blocks. For these reasons, and to handle other

* Corresponding author.

0167-9260/00/$ - see front matter
PII: S0167-9260(99)00021-8

complexity issues associated with DSM, top–down block-oriented design methodologies have been advocated, as a means of reducing design complexity [1]. 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, with chip-level assembly of pre-characterized intellectual property (IP) modules with specified area-delay-power flexibilities as 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: combine pre-designed and characterized modules with different flexibilities along with new modules that serve a special purpose in a given design. We can then deduce that design methodologies using trade-offs such as between area, delay and power 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.

In such an IP integration framework for DSM, interconnect delays will play an increasingly important role. As system-on-chip (SoC) designs and faster clock speeds become a reality, timing and interconnect delays will become more critical, while interconnect effects impact performance, compromise signal integrity, and increase power dissipation. By 2006, according to NTRS,[1] designs will contain over 100M transistors in a technology with feature sizes well below 0.1 μm technology and 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. Signal delay and degradation in the DSM era can have a crippling effect on the performance of designs. Signal preservation cannot be solved as an afterthought 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 variable inter-module interconnect lengths, as well as making interconnect delay at the global level dominant over that of devices [2]. 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 start to exceed the global clock period of the design, the delay on some global wires will become lower bounded

[1] Such forecasts can be found in the *National Technology Roadmap for Semiconductors*, from 1997 published by *Semiconductor Industry Association*. Another intersting source is the *Draft Report on Physical Design* from the same year, and composed by the *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.

by an integer number of clock cycles, based on a pre-selected 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).

1.2. Application domain

The target application of this work[2] is the chip level assembly of IP blocks, each fewer than 100 k gates in size, either as hard, firm, or soft macros. The design of such a system involves the placement and wireplanning for performance [3] of 200–2000 modules whose average size is 50 k gates with a dynamic range of module sizes of 1–500 k gates [2]. Modules can be of different types: hard (layout), firm (gate level), or soft (register transfer level). Such a network has a large number of nets: 40–100 k 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 versus delay). 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 IPs, and thus temporally decouple 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.

1.3. The target design flow

The system-level integration framework being advocated supports a 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- or higher-level (that may generate solutions that are too far away from the global optimum). The objective of this framework is the placement and routing of IP modules in order to integrate them into a system as described above. It should be noted that this is a dramatically different problem than the one typically addressed in current design frameworks.

The aim of this new design flow (which is based on [3]) is to minimize the number of design iterations by supporting two critical guiding notions: planning at the early stages of the flow, and

[2] This is part of the activities of the NexSIS CAD Group, University of California at Berkeley, with principal investigators: Robert K. Brayton and A. Richard Newton. (http://gigascale.org/nexsis).

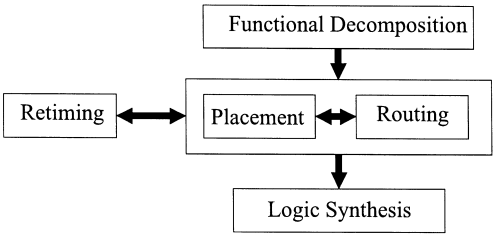


Fig. 1. Design flow.

incremental successive refinement during the integration process. One of the main characteristics of this design flow is the integration of retiming with architectural floorplanning in order to have a better handle on timing issues as required for DSM circuits (see Fig. 1).

The proposed flow has the following components:

Functional decomposition: This step provides an entry point for reused IPs, where register transfer level descriptions may already be well characterized, and area-delay trade-offs are taken in as important performance aids. 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 any constructive approach such as min-cut for example. It has to be fast, and should yield 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 about possible implementations for each module are retained through area-delay trade-offs to guarantee convergence.

Note that unlike current design flows, iterations are made incremental, with information from previous iterations kept around for use in subsequent iterations.

1.4. 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. There are two extreme forms:

1. minimizing the clock period without regard to the number of registers in the final circuit or,
2. minimizing the number of registers in the final circuit with no constraints on the clock period

and many more practical variants in between. A common variant on that theme is that of constrained minimum area retiming, that is retiming with the main objective of finding a circuit with the minimum number of registers for a specified 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. 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 [3] 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 (see Fig. 2):

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 implies 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

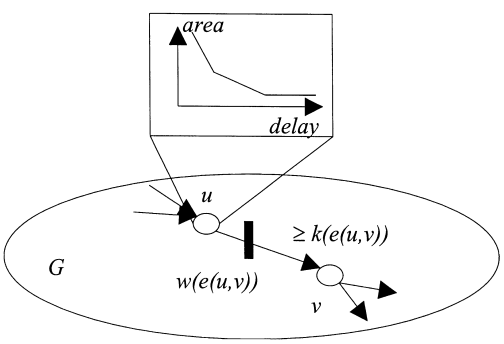


Fig. 2. Problem statement.

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:

$$\text{minimize} \quad A(G_r) = \sum_v a_v(d)$$

$$\text{subject to} \quad w_r(e(u,v)) \geqslant k(e(u,v))$$

2. Retiming

2.1. Leiserson–Saxe retiming algorithm

Retiming is a procedure that involves relocating registers across logic blocks to allow the circuit to be operated under a faster clock. Leiserson and Saxe first proposed the technique, where the algorithmic basis of retiming circuits with registers was described without specifically focusing on implementation aspects. Retiming to achieve the minimum clock period is termed minimum period retiming, while retiming to minimize the number of registers for a given target clock period is called minimum area retiming.

A sequential circuit is represented by a directed graph $G(V,E)$, where each vertex v corresponds to a gate, and a 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 $r : V \to Z$, 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 $p : u \to v$)), 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

$$c = \operatorname*{Max}_{\{p \mid w(p)=0\}} d(p).$$

Another important concept is that of the $\boldsymbol{W}$ and $\boldsymbol{D}$ matrices which are defined as follows:

$$W(u,v) = \operatorname*{Min}_{p\,:\,u \to v} w(p),$$

$$D(u,v) = \operatorname*{Max}_{\{p \mid w(p)=W(u,v)\}} \mathrm{d}(p).$$

The matrices are defined for all pairs of vertices (u, v) such that there exists a path $p: u \to v$ 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.

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

$$S(G_r) = \sum_e w_r(e) = \sum_e (w(e) + r(v) - r(u)) = S(G) + \sum_v (|FI(v)| - |FO(v)|)\, r(v),$$

where $|FI(v)|$ and $|FO(v)|$ are the number of fanins and fanouts of node v. The case where not all register costs are the same can be modeled as

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

The minimum area retiming problem can then be formulated as the following linear program:

$$\begin{aligned} \text{minimize} \quad & \sum_v (|FI(v)| - |FO((v)|)\, r(v) \\ \text{subject to} \quad & r(u) - r(v) \leqslant w(e(u, v)), \\ & r(u) - r(v) \leqslant W(u, v) - 1, \forall_{u,v}\colon D(u, v) > c. \end{aligned}$$

The meaning of the objective function and the constraints is as follows (the reader is referred to Leiserson and Saxe [4] 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 registers 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 [4], 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 there. It is also pointed out in 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. Improved techniques

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

Shenoy and Rudell [5] describe two major hurdles towards an efficient implementation of minimum area retiming:

- computing the $\boldsymbol{W}$ and $\boldsymbol{D}$ 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 $\boldsymbol{W}$ and $\boldsymbol{D}$ matrices, is an algorithm with $\mathcal{O}(|V|^3)$ time and $\mathcal{O}(|V|^2)$ space complexity even in the best case, because it is based on solving the all-pairs shortest-paths algorithm.[3]

The advantage in is the space savings obtained by an algorithm that computes in $\mathcal{O}(|V|)$ space and generates a smaller set of constraints. One of the best methods known 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 $\mathcal{O}(|V||E|lg|V|lg|V|C)$ where C is one more than the number of registers in the circuit.

With the introduction of the Astra algorithm [6] an alternative view of retiming using the equivalence between retiming and clock skew optimization was proposed. The Minaret algorithm [7] uses the linear programming formulation and incorporates the Astra approach to reduce the number of variables and constraints. 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.

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.

[3] To be fair, in [4] the authors state that, for sparse graphs one could use an $\mathcal{O}(|V||E| + |V|^2lg|V|)$ 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.

Phase B: The skew solution is translated into 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, but never by an amount larger than the greatest node delay in the network. 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 [7] 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 additional computation time on the ASTRA runs, this method leads to significant reductions in the total execution time of the minimum area retiming problem.

2.3. Minimum cost network flow

The minimum area retiming problem can be reduced to the minimum cost network flow problem [4], 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 can assign to each edge a non-negative flow $f(e(u, v))$ such that:

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

$$\text{subject to} \quad \sum_{v \to ?} f(e) - \sum_{? \to v} f(e) = |FO(v)| - |FI(v)|.$$

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 [8], the problem can be solved in $\mathcal{O}(|E|^2 lg|V| + |V||E|lg^2|V|)$ 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. More recent work, which is more relevant here, presents an algorithm for handling convex costs [9]. In this work the authors present a method for extending the work of Orlin [8] to handle convex costs while retaining the strong polynomial properties of the problem. Even though 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 [5]. 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.

3. Minimum area retiming with area-delay trade-offs and constraints

3.1. Transformation

Before casting the problem at hand into a minimum area retiming problem, we state some assumptions:

1. The area-delay trade-off curves are monotone decreasing and convex: the slope of the curve never gets steeper with increasing delay. Without this assumption the problem might be $\mathcal{NP}$-hard, since the exploration of all possible combinations of trade-offs at the different nodes seems to be required.
2. We assume that the wire delays considered are much larger than the delays at the nodes and the clock period. Delays at the nodes are neglected. This seems to be a valid assumption in the DSM regime [2,3].
3. 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 linear. Thus the mathematical programming problem is:

$$\begin{aligned} \text{minimize} \quad & A(G_r) = \sum_e a_e(w_r(e(u,v))) \\ \text{subject to} \quad & w_r(e(u,v)) = w(e(u,v)) + r(v) - r(u) \geqslant k(e(u,v)) \\ & \Rightarrow r(u) - r(v) \leqslant w(e(u,v)) - k(e(u,v)), \end{aligned}$$

where a_e represents the convex area-delay trade-off 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.

Using the approach of Pinto and Shamir [9] 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 (see Fig. 3). This is accomplished by splitting nodes in the original graph into two nodes connected by one edge, but into several nodes with one edge for each line segment of the trade-off curve. The equation for the total area can then be rewritten as

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

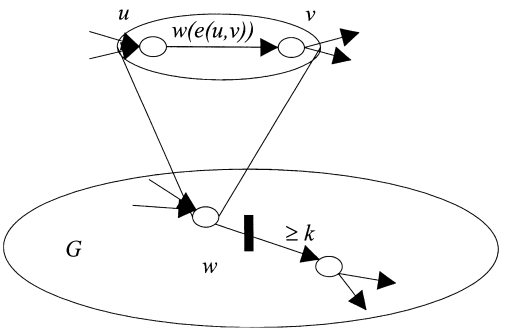


Fig. 3. Transformation at the graph level.

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 (*width*) on the delay axis:

$$cost(e(u,v)) = slope(l),$$

$$0 \leqslant w(e(u,v)) \leqslant width(l).$$

In general, there are two possible types of constraints:

$$r(u) - r(v) \leqslant w(e(u,v)) - w_l(e(u,v)),$$

$$r(v) - r(u) \leqslant w_u(e(u,v)) - w(e(u,v)),$$

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 $w_u = \infty$.

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, 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 in the minimum retiming solution it is always the case that l_1 is completely filled ($w_r(l_1) = width(l_1)$) whenever l_2 gets filled ($w_r(l_2) > 0$).*

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

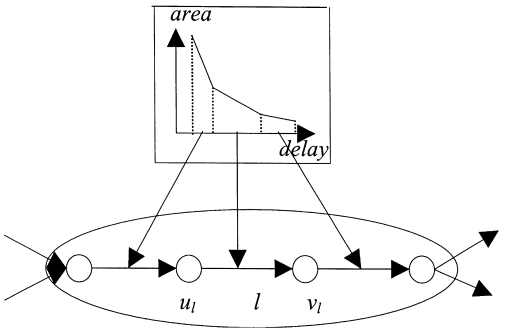


Fig. 4. Transformation at the vertex level.

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* (*see Fig. 4*) *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.

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 abstraction level of the description can control the granularity of the problem. This fact can be used to control the complexity and level of optimization required in solving this problem.

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 accomplished by having lower bound constraint on added edges used to represent area-delay trade-offs of modules.

More general delay models: As described in [4] 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 our original circuit model to handle this situation, and shows how the retiming problem can be reapplied to this more general case.

3.2. The algorithm

The algorithm is a two-phase process. The first phase involves checking 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.

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

$$r(u) - r(v) \leqslant w(e(u,v)) - w_l(e(u,v)) = r_u(u,v),$$

$$r(v) - r(u) \leqslant w_u(e(u,v)) - w(e(u,v)) = -r_l(u,v).$$

From these constraints we set up a weight matrix $\boldsymbol{R}$ 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 [10], 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 $\boldsymbol{R}$. In order to derive constraints, we apply an all-pairs-shortest-path approach to convert the $\boldsymbol{R}$ into canonical form, which represents tight constraints on the retimings. We then derive upper and lower bounds on the number of registers using the equations

$$w_l(e(u,v)) = w(e(u,v)) - r_u(u,v),$$

$$w_u(e(u,v)) = w(e(u,v)) - r_l(u,v).$$

Phase II: Now that the problem has been cast into a minimum area retiming problem without cycle-time constraints, it can be solved using the linear programming approach originally suggested by Leiserson and Saxe [4] who did this through a minimum cost flow dual formulation, which can be used to derive the optimal flow and then the optimal potential that directly translates into retiming at the nodes. Using the improved techniques [5,7] discussed earlier, one can optimize the space requirement and reduce the number of constraints and variables required, thus making the problem more manageable. A relaxation-based is also possible, although it might be inefficient in some cases. The information derived from the slacks computed in the first phase are then used to decide to put the registers on the edges with the most negative cost, after which new slacks are derived for the subgraphs, until the minimum area solution is reached.

The final algorithm has the following steps:

1. Consider the area-delay trade-off curve for each node in the original graph.
2. 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.
3. Solve the minimum area retiming of the resulting problem.

4. 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.

4. A simple example

We used already available capabilities in the retime package in SIS [11]. That package contains an implementation of the two types of retiming using the Leiserson–Saxe approach described earlier. The minimum area capabilities are relevant here. The necessary modifications are listed below.[4]

- 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,
- $\boldsymbol{W}$ and $\boldsymbol{D}$ 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.

The example was chosen to show some of the aspects of the algorithm. The retime graph has 17 edges and eight 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 does not 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 is an interesting case of retiming, and shows what can happen during this process:

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.

[4] A better implementation of this software package will be integrated into NexSIS, the new IP integration SoC framework currently being developed at the University of California at Berkeley. Currently work is going on in creating benchmarks for the SoC integration domain. In this section, the example to demonstrate the retiming part of the flow is derived from *s*27 from the ISCAS89 benchmarks and was used in [7] (see Fig. 5).

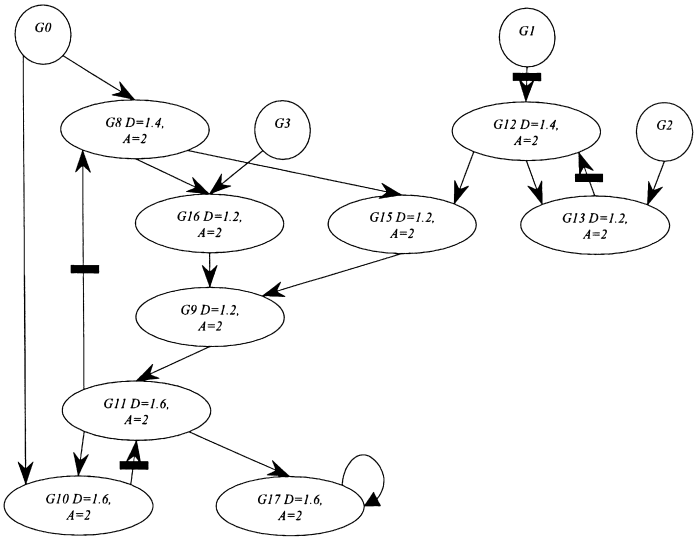


Fig. 5. s27 retiming example.

- 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. Interconnect strategy

Retiming [12] and placement techniques [3] 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 section focuses on suggesting interconnect solutions, when the registers retimed onto wires cannot be implemented by reassigning wires to slower metal layers. We have devised a registered interconnect strategy for the IP integration component-based design SoC domain [13]. This strategy is summarized here.

5.1. 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), and
- low power consumption.

5.2. Circuit implementation

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

5.2.1. 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 the proposed pipelined interconnect strategy.

5.2.2. IP interconnect strategy implementation

True single phase clock (TSPC) latches are commonly used in high-performance digital systems due to their simplicity and fast operation [14]. The advantages of this choice are: the single clock phase, which avoids clock overlap problems, and the low clock loading. One can generalize the above TSPC latch and recognize that there are four basic stages in TSPC latch or register design as shown in Fig. 6 [15].

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. Using the

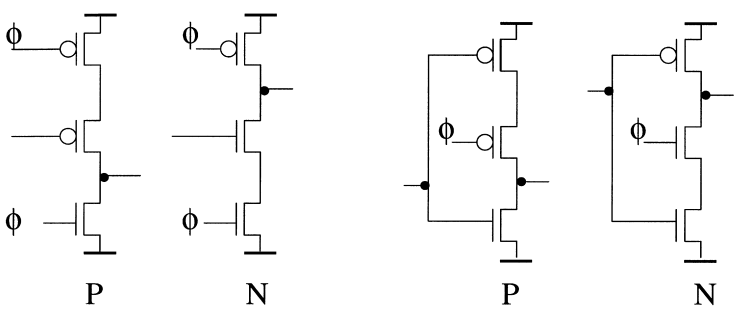


Fig. 6. The basic four TSPC stages.

TSPC basic stages shown earlier it is possible to devise several interconnect pipeline schemes [13]. These schemes can each be implemented as:

1. *lumped*: optimally buffered wire segments surrounded by registers, or
2. *distributed*: registers split up into the basic TSPC stages which replace the buffers.

The schemes can then be evaluated with or without coupling to account for crosstalk. Initial results [13] 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.

6. Conclusions and future developments

In this work we showed how the minimum area retiming approach could 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.

Acknowledgements

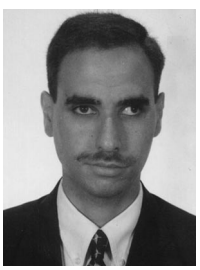
We are grateful for the financial support for this research provided by the SRC under contract 98-DC-324, and for the support of the California Micro program and participating industrial sponsors, Fujitsu, Motorola, Cadence, Synopsys, Intel, and Metamor Inc. This work was supported in part by the MARCO/DARPA Gigascale Silicon Research Center (http://www.gigascale.org). Their support is gratefully acknowledged.

References

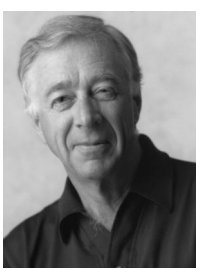
[1] F. Eory, Systems to Silicon Design: Methodology for Deep Sub-micron ASICs, SuperCon, 1997.
[2] D. Sylvester, K. Keutzer, Getting to the bottom of deep submicron, Proceedings of the International Conference on Computer Aided Design, 1998, pp. 203–211.
[3] R.H.J.M. Otten, R.K. Brayton, Planning for Performance, Proceedings of the Design Automation Conference, 1998, pp. 122–127.
[4] C.E. Leiserson, J.B. Saxe, Retiming synchronous circuitry, Algorithmica 6 (1991) 5–35.
[5] N. Shenoy, R. Rudell, Efficient implementation of retiming, Proceedings of the International Conference on Computer Aided Design, 1994, pp. 226–233.
[6] D.B. Deokar, S.S. Sapatnekar, A fresh look at retiming via clock skew optimization, Proceedings of the 32nd Design Automation Conference, 1995, pp. 310–315.

[7] N. Maheshwari, S.S. Sapatnekar, An improved algorithm for minimum-area retiming, Proceedings of the 34th Design Automation Conference, 1997, pp. 2–7.
[8] J.B. Orlin, A faster strongly polynomial minimum cost flow algorithm, Oper. Res. 41 (2) (1993) 338–350.
[9] Y. Pinto, R. Shamir, Efficient algorithms for minimum-cost flow problems with piecewise-linear convex costsD, Algorithmica 11 (1994) 256–277.
[10] R. Alur, Timed automata, NATO-ASI Summer School on Verification of Digital and Hybrid Systems, 1998.
[11] E. Sentovich, Sequential circuit synthesis at the gate level, Ph.D. Thesis, UC Berkeley, 1993 (Chapter 5).
[12] A. Tabbara, R.K. Brayton, A.R. Newton, Retiming for DSM with area-delay trade-offs and delay constraints, Proceedings of the Design Automation Conference, 1999, pp. 725–730.
[13] A. Tabbara, B. Tabbara, Inter-module interconnect strategy for system on chip applications, University of California at Berkeley Electronics Research Laboratory, Memorandum No. UCB/ERL M99/45, September 1, 1999.
[14] Y. Ji-Ren, I. Karlsson, C. Svensson, A true single-phase-clock dynamic CMOS circuit technique, IEEE J. Solid State Circuits J-SC-22 (5) (1987) 899–901.
[15] J. Yuan, C. Svensson, Fast and robust CMOS double pipeline using new TSPC multiplexer and demultiplexer, International Conference on ASIC, 1996.

Abdallah Tabbara is a Ph.D. Candidate in EECS at the University of California at Berkeley. He received his B.S. (magna cum laude) from UC Riverside and his M.S. in Electrical Engineering from Berkeley. He is a member of ACM, and the CAD for VLSI group at Berkeley. His research interests include synthesis, and physical design for DSM, as well as system-level design, especially in the context of IP integration for System-on-Chip designs.



Bassam Tabbara is a Ph.D. Candidate in EECS at the University of California at Berkeley. He received his B.S. in Electrical Engineering (summa cum laude) from UC Riverside in 1994, and his M.S. in Electrical Engineering from Berkeley in 1998. He is a fellow of the SRC, member of IEEE, and the CAD for VLSI group at Berkeley. His research interests include specification, validation, optimization, and synthesis of embedded systems, and IP assembly and integration for System on Chip applications.



Robert Brayton received the BSEE degree from Iowa State University in 1956 and the Ph.D. degree in mathematics from MIT in 1961. From 1961 to 1987 he was a member of the Mathematical Sciences Department of the IBM T.J. Watson Research Center. In 1987 he joined the EECS Department at Berkeley, where he is a Professor and director of the SRC Center of Excellence for Design Sciences. He has authored over 300 technical papers, and 7 books. Dr. Brayton holds the Edgar L. and Harold H. Buttner Endowed Chair in Electrical Engineering in the EECS Department at Berkeley. He is a member of the National Academy of Engineering, and a Fellow of the IEEE and the AAAS. He received the 1991 IEEE CAS Technical Achievement Award, and five best paper awards, including the 1971 IEEE Guillemman-Cauer award, and the 1987 ISCAS Darlington award. He was the editor of the Journal on Formal Methods in Systems Design from 1992–1996. Past contributions have been in analysis of nonlinear networks, electrical simulation and optimization of circuits and asynchronous synthesis. Current research involves combinational and sequential logic synthesis for area/performance/testability, formal design verification and synthesis for DSM designs.

A. Richard Newton received the B.Eng and M.Eng.Sci degrees from the University of Melbourne, in 1973 and 1975, respectively, and the Ph.D. degree from the University of California, Berkeley, in 1978. Since 1979, he has been involved with research and teaching with the University of California, Berkeley, in the area of computer-aided design for electronic systems. Dr. Newton had received a number of awards, including Best Paper awards at the 1988 European Solid State Circuits Conference, the 1987 and 1989 ACM/IEEE Design Automation Conferences, and the International Conference on Computer Design. He was selected in 1987 as the national recipient of the C. Holmes McDonald Outstanding Young Professor Award of the Eta-Kappa-Nu Engineering Honor Society. In 1989, he was corecipient of the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems Best paper Award. He was an Associate Editor of the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems from 1985 to 1988, and the Technical Program Chair of the 1988 and 1989 ACM/IEEE Design Automation Conferences. He was General Chair of the 1991 Conference. He was a Founding Member of the EDIF Technical and Steering Committees.