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

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

Department of Electrical Engineering and Computer Sciences,
University of California, Berkeley, CA 94720

{atabbara,brayton,rnewton}@ic.eecs.berkeley.edu

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 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 whose solution is polynomial time solvable.

1 Motivation

Retiming is a technique for optimizing sequential circuits. It 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 focussed 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.

It is generally agreed, within the EDA community [11], that Deep Sub-Micron (DSM) semiconductor technology is forcing major discontinuities in traditional design methods. The complexity and scale of integration, as well as the significant cost of design errors promotes a re-evaluation of design practice and an increase in "co-design" early in the design process. For this reason, and to handle complexity issues associated with DSM, block-oriented design methodologies have been advocated as a means of reducing complexity of design [3]. In this methodology 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 three metrics: timing, area and power. Design using distributions (for example area-delay-power trade-offs of components) is also becoming more important. Wire delays will also play an important role in DSM. As system on a chip (SOC) designs and faster clock speeds become a reality, timing and wire delays will become more critical; on some global wires, the delay will become lower bounded by an integer number of clock cycles.

In the following we present a solution of a specific formulation of a problem that will become relevant in the next few technology generations. In Section 2 we state the problem and then describe previous minimum area retiming approaches, since this problem is very related to the problem we are trying to solve. In Section 3 we describe a formulation of a solution and then discuss an algorithm. In Section 4 we describe the implementation and present initial results in Section 5.

2 Background

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.

2.1 Problem Statement

A new "retiming problem" comes up in a non-iterative flow [7] for deep sub-micron applications. 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: $$w_r(e(u,v)) \geq k(e(u,v))$$

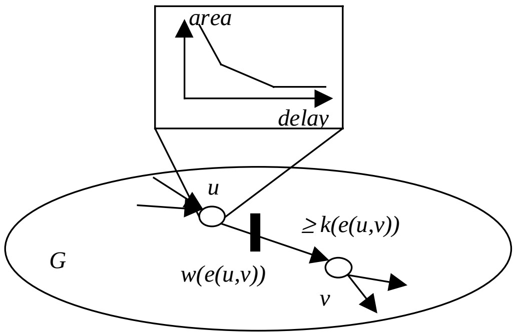

Figure 1. Problem statement

2.2 Leiserson-Saxe Retiming Algorithm

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.

2.2.1 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 $r : V \to Z$, 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

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

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

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

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.

2.2.2 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 $S(G_r)$

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

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:

$$r(u) - r(v) \leq w(e(u,v))$$
$$r(u) - r(v) \leq W(u,v) - 1, \forall u,v : D(u,v) > c$$

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

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

The significance of the objective function and the constraints is as follows (the reader is referred to [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 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 [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 in [4].

It is also pointed out in [4] 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.

2.3 Improved 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 [10] [2]. In the following subsections, we list several such algorithms.

2.3.1 Shenoy-Rudell Approach

Shenoy and Rudell [10] 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 algorithm[1]. The advantage in [10] 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 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 $O(|V||E|\ lg\ |V|\ lg\ |V|\ C)$ where C is one more than the number of registers in the circuit.

2.3.2 ASTRA and Minaret

The ASTRA algorithm [2] proposed an alternative view of retiming using the equivalence between retiming and clock skew optimization. The Minaret algorithm [5] 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 [2].

- 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 [5] 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.

2.4 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

[1] To be fair, in [4] 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.

programming problem given asks that one assign to each edge a non-negative flow *f(e(u,v))* such that:

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

$$\text{Subject to:} \quad \sum_{v\to?} f(e) - \sum_{?->v} f(e) = |FO(v)| - |FI(v)|$$

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 [6], 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. More recent work, which is more relevant here, presents an algorithm for handling convex costs [8]. In this work the authors present a method for extending the work of Orlin [6] to handle convex costs while still 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 [10]. 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

The solution of MARTC is based on the minimum area retiming approach without clocking constraints. Algorithms for solving the resulting problem are then discussed.

3.1 Transformation

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 [7].
- 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 represents 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:

$$\text{Minimize:} \quad A(G_r) = \sum_e a_e(w_r(e(u,v)))$$

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.

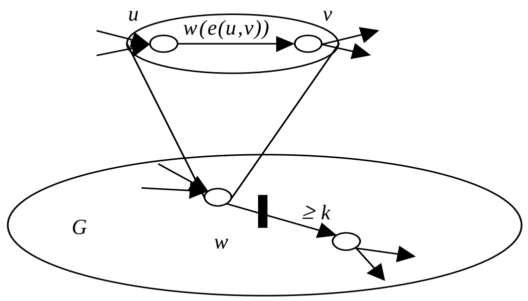

Figure 2. Transformation at the graph level

Using the approach of [8] 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:

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

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.

$$cost(e(u,v)) = slope(l)$$
$$0 \leq w(e(u,v)) \leq width(l)$$

In general, there are two possible types of constraints:

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

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.

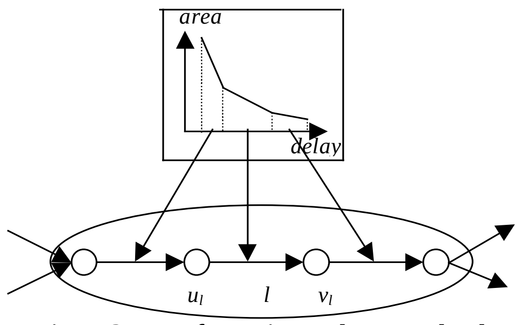

Figure 3. Transformation at the vertex level

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 concave. 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.

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.

3.2.1 Phase I

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

$$r(u) - r(v) \le w(e(u,v)) - w_l(e(u,v) = r_u(u,v)$$
$$r(v) - r(u) \le w_u(e(u,v)) - w(e(u,v)) = -r_l(u,v)$$

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) [1], 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 [1].

In order to derive constraints, we apply an all-pairs-shortest-path approach to convert the DBM into canonical form [1], which represents tight constraints on the retimings. We then derive upper and lower bounds on the number of registers using the equations:

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

3.2.2 Phase II

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 the of first approaches [4] 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 [10] [5] 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 and 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 concave piece-wise linear area-delay trade-off curves.

4 Implementation

Our implementation approach is to reuse already available capabilities in the retime package in SIS [9]. In the following we give, first a description of what is available in the retime package, and second what were the modifications needed to make the package work for the new problem.

4.1 The Retime 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

5 Retiming Example

An example circuit, presented here is s27 derived from the ISCAS89 benchmark, which was used in [5]. 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:

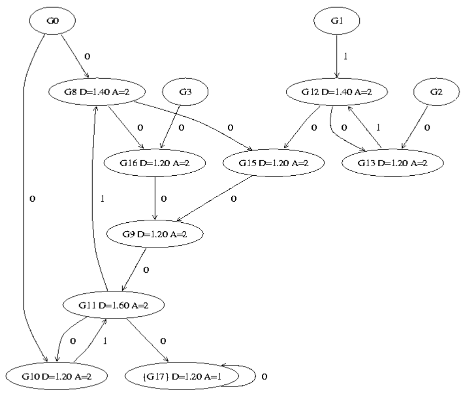

Figure 4. s27 retiming example

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.

6 Conclusions

In this paper 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.

Acknowledgements

We're grateful for 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.

References

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