On the Detection and Elimination of Superfluous Level-Sensitive Latches

Glenn Jennings
Department of Computer Engineering, Lund University
P.O. Box 118, S-221 00 Lund, Sweden
Tel. (+46)-46-10 97 63, Fax (+46)-46-10 47 14
Internet: glenn@dit.lth.se

Abstract

*We describe an automated technique for defining, identifying, and removing superfluous level-sensitive latches in large circuits which may contain any number of clocks. The technique presented here is based on a delay and function independent design methodology for level-sensitive latched circuits having completely general clocking schemes (***n** *clocks,* **m** *phases). More precisely, the technique identifies superfluous clock transitions at each latch, so that superfluous clock waveforms may be simplified. If after transformation the clock waveform becomes identically enabled, then the latch itself may be removed and replaced by a dead short. Analysis complexity and experience with an implementation are discussed.*

1 Introduction

In this paper we address a practical synthesis problem, that is, when assembling a large circuit from smaller structural[1] assemblies (for example, from a library) it is possible that some of the transparent latches in the resulting aggregation serve no useful purpose. The result is generally that these latches will slow the circuit down unless they are removed. For example, if we have a structural module with a latch at its output, and join it to a second module which has its own latch at its input, then we can eliminate one of those latches if they are driven by the same clock. However, if they are driven by different clocks, it is more difficult to detect (or for that matter, even to *define*) whether there is a redundancy. Finally, when we generalize the problem to redundancy of latches *anywhere* in a large circuit (that is, between *any pair* of latches), and allow the number of clocks *and* the number of phases to increase without bound, then there is no longer any hope of detecting such redundancy by visual inspection of the circuit.

The reason that we as well as others [NMN82] [GD85] [UT86] [CM88] [CB+90] [DY90] [Tjä91] are interested in constructions built with the level-sensitive latch is not only because of the simplicity of this clocking element, but also because of the benevolent phenomenon of *cycle borrowing* [DR89] [SMO90] [Ous85] [Szy86] [UT86] [Jou87] [CM88] which accompanies designs using it. When attempting to exploit cycle borrowing, it is especially true that superfluous latches can severely interfere with the maximum clock rate, as demonstrated in [Bre91].

We shall present a solution to this superfluous latch problem for circuits based on the formalism of [JJ91] such that we do *not* need to examine each pair of latches individually. Our solution requires an initialization pass proportional to the size of the circuit times the number of epochs (phases), followed by a small cost (worst case proportional to the number of epochs) per verified latch.

2 The Clocking/Circuit Syntax

In [JJ91] we have described a formalism for the specification of clocked circuits designed from combinational components and level-sensitive latches. The formalism allows the circuit to be clocked by **n** distinct clocks, where each full clock cycle is divided into **m** epochs (phases), as shown in Figure 1. Every clock may make *at most* one transition per epoch. Unlike [DR89] there is no discussion about "independent" events and unlike [SMO90] there is no limit on the

[1] By *structural* we mean that the circuit is described as components and their interconnections, for example, an RTL description, and *not* that it has been committed to layout. Thus we still have the opportunity for optimizing aggregations of such RTL modules.

0-8186-2610-0/92 $3.00 © 1992 IEEE

number of transitions a given clock may make during a complete clock cycle. The clocking scheme is encoded as an n-row m-column matrix called S which contains only ones and zeroes indicating the state of each clock at the end of each epoch. An example of schedule S and its derivative schedules D and T are shown in Figure 1. The formalism guarantees that even though the exact delays of the components are not yet known, the specification can be given in terms of this discrete clocking syntax together with a circuit syntax which is a graph formalism parameterized by the discrete clocking syntax. If complied with, the formalism guarantees the absence of timing ambiguities, oscillation and race free operation, and that there always exists *some* values for the individual lengths of the m epochs such that the circuit performs its function. Those constraints are all one-sided, that is, all slower schedules work as well.

The circuit G is formulated as a directed multigraph consisting of combinational components C and level-sensitive latches L_i treated as primitives and having clocks $1 \leq i \leq n$. We do not consider the *gated latch* L_i^* in this paper. From G and from the discrete scheduling matrix S we derive the graphs ${}^{\psi k}H(S)$ for $0 \leq k < m$. Intuitively, these m graphs represent the configuration of the circuit G when the clocks have settled at the end of each epoch, so that latches in G which are disabled are "broken" into a source and sink

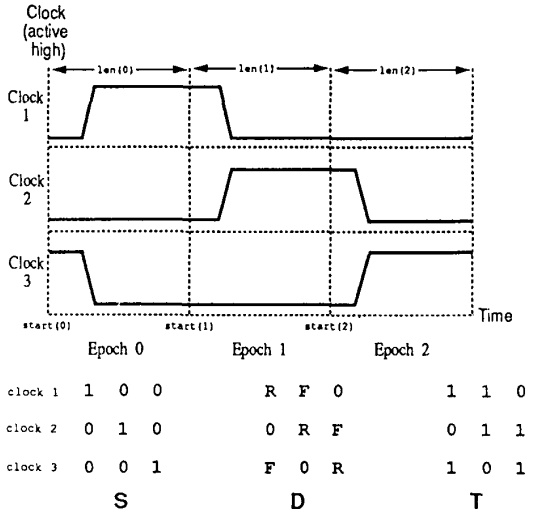

Figure 1: Example of n clocks and m epochs (phases), $m = n = 3$, and the resulting discrete clocking schedule S. Schedules D and T are derived directly from S.

vertex in the corresponding $H(S)$. In practice these graphs ${}^{\psi k}H(S)$ are trivial to construct automatically from G. The formalism requires that each ${}^{\psi k}H(S)$ be acyclic. Another feature of the formalism relevant to this paper is that within any given ${}^{\psi k}H(S)$, there will be no latch which is disabling (closing, $D = F$) which has as a predecessor any other latch which is enabling (opening, $D = R$).

Finally we associate delay parameters with the components C and latches L_i, accomodate skew and other timing uncertainties by the parameters *tau* as seen in Figure 1, and at last define the duration of each epoch with the parameters $len(\psi k)$. The cycle borrowing analysis which we use to compute minimum cycle times in this paper is based on a non-iterative linear programming formulation reported in [Jen92c] and which is also founded on the formalism of [JJ91]. Other investigations making use of the circuit formalism together with this cycle borrowing analysis may be found in [Jen92a] [Jen92b] [Jen92e].

The reason we stress this formulation of the circuit in terms of the matrix S is that the procedure we describe below not only transforms the circuit G (by removing superfluous latches) but it does so by transforming the clock waveforms, that is, the rows of S, as seen at each latch. This means that the matrix S is subject to change during the procedure. For example, if we are able to simplify the clock at some latch L_j then we have in effect synthesized a new row for the matrix S (assuming our new clock does not match some other already-existing row in S), and therefore the dimension n of S will change. Also as discussed in [JJ91], such changes to S often result in our being able to reduce the number of epochs, so that the dimension m of S will also change. An example of this will be given below. When we have removed the last latch driven by some row of S, then that row in S can be deleted. In short, formulating the circuit in terms of [JJ91] allows us to perform transformations on the temporal sequencing of the circuit as well as on the structure itself, and it is this facility which has allowed us to address the superfluous latch problem.

3 The Basic Idea

Detection of superfluous latches is based on an important result from the formalism of [JJ91], which is that the value at a latch input in ${}^{\psi k}H(S)$ will change only if it has a predecessor latch in that ${}^{\psi k}H(S)$ which is being enabled ($D = R$). Therefore if we find some latch Q which is disabled and then later

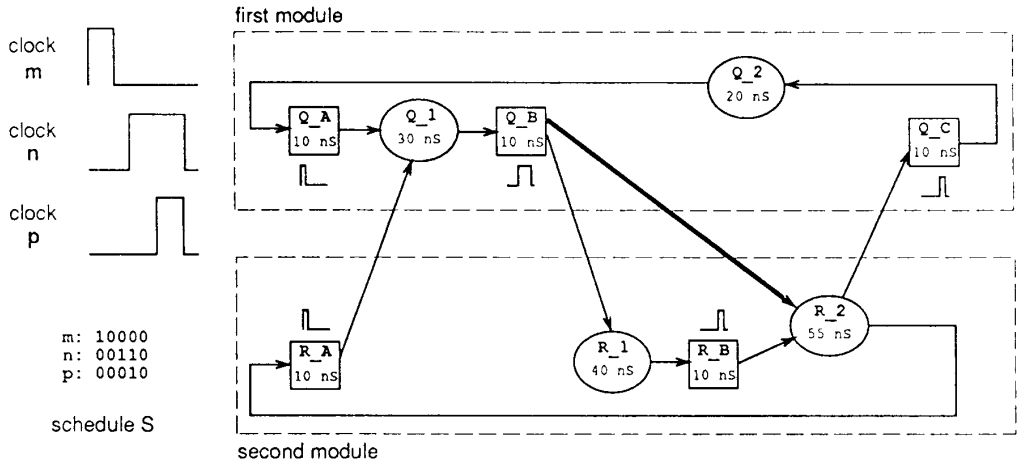

Figure 2: Two interconnected structural modules. The system is driven by three clocks, over a cycle having five epochs, as shown by schedule S. Latches are indicated by the small squares, and combinational units by ovals.

enabled, *but* during that disabled time *none* of its predecessor latches ever undergoes a disabled-to-enabled transition, then that off-then-on transition at latch Q serves no useful purpose, since the value trapped by Q when the disabling falling clock edge arrived at Q will be exactly the same as the value at Q's input during the entire time that Q is disabled by that falling transition. Therefore that entire off-then-on transition in the clock which drives Q can be removed, which simplifies the clock waveform driving Q. If the simplification has gone so far that the clock now remains "on" during every epoch (that is, for the entire clock cycle), then Q is always enabled (since we are not considering gated latches), and therefore Q itself can be removed from the circuit and replaced by a direct connection from input to output. Of course, a dead short is electrically much faster than the latch it replaces.

```
m:   10000
n:   00110
p:   00010

q_a(m)   10.0   → q_1
q_1      30.0   → q_b
q_b(n)   10.0   → r_1, r_2
q_c(p)   10.0   → q_2
q_2      20.0   → q_a
r_a(m)   10.0   → q_1
r_1      40.0   → r_b
r_b(p)   10.0   → r_2
r_2      55.0   → q_c, r_a
```

Figure 3: Text input to our analyzer for the circuit shown in Figure 2.

The algorithm is understood intuitively as follows: we examine each ${}^{\psi k}H(S)$ in turn, seeking latches for which $D = R$, that is, which are being *enabled* in epoch ψk. For each such latch Q we now loop in reverse order and examine that latch in ${}^{\psi(k-1)}H(S)$, ${}^{\psi(k-2)}H(S)$, and so on. This test will terminate on that ${}^{\psi(k-j)}H(S)$ at which latch Q's clock makes its *falling* transition ($D = F$) which is the last $H(S)$ we need to examine.[2] For each such $H(S)$, examined in the order indicated, we identify our latch Q and then perform a depth-first search with pruning *backwards* towards the *sources* of the directed acyclic graph $H(S)$. If we can identify a predecessor latch to Q which is opening ($D = R$) then the off-then-on transition at latch Q is *not* superfluous and we can terminate the test. Otherwise we must examine all predecessors in that $H(S)$, and when that $H(S)$ is exhausted, then we must examine the preceding $H(S)$ until either a rising predecessor is found (the transition, and hence Q, is useful) or else all $H(S)$ in the sequence have been exhausted (the transition is superfluous).

In practice we do not perform this reverse depth-first search for each $H(S)$ for each latch to be examined. Instead for each ${}^{\psi k}H(S)$ we perform a single initializing forward *topological sort* [Man89] pass in which

[2] Strictly speaking we do not need to examine this graph since as we have indicated, the formalism forbids latch Q from having a rising predecessor in this case.

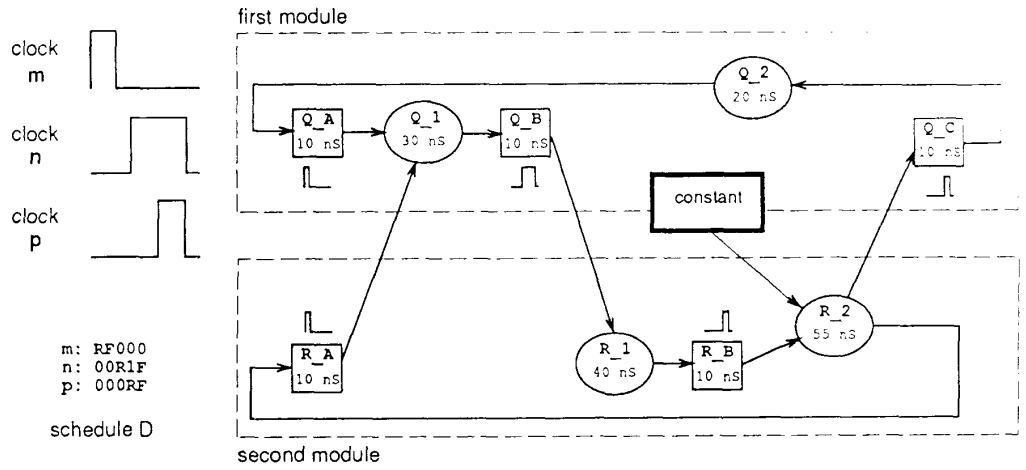

Figure 4: The same two modules from Figure 2, with the same clocking schedule, except that one of the inputs to combinational component "R_2" has been assigned to a constant. This causes latch redundancy to occur in the circuit. Here we show clock schedule D which is directly and trivially derived from schedule S in Figure 2.

the property "has rising predecessor" (*true* or *false*) is computed during forward propagation, and is recorded with each latch for that ψk. Starting from the sources of ${}^{\psi k}H(S)$ the value *false* is propagated forward until a latch is visited whc e driving clock ${}^{\psi k}D = R$, that is, whose clock is undergoing an off-to-on transition during ψk. From this node forward we now propagate the value *true*. When we visit nodes of the graph ${}^{\psi k}H(S)$ we also propagate out the value *true* if any of the incoming values to that node were *true*. At the same time, as we visit nodes which are latches, we also record the logical "or" of the incoming values with each latch for this ψk. Note that this means that a latch which is itself opening (and hence giving rise to the value *true* being propagated outward) may itself have the value *false* associated with it, that is, *false* is arriving at all of its inputs. This pass is linear in the size of ${}^{\psi k}H(S)$. The result, after performing all m of these passes (one for each epoch) is a Boolean array (of length m, one entry for each epoch) associated with each latch. Then, when it is time to examine each individual latch Q we simply need to search backwards in this array which states "has rising predecessor" for each epoch, in order to perform the test for every epoch during which the latch is disabled. This test is linear in m but with a very small coefficient. The cost for testing all latches in the circuit G (after the Boolean arrays have been built) is this small cost, incurred for each rising clock transition at each latch, summed over all latches in the circuit.

4 A Simple Example

Now consider the circuit in Figure 2. Here we suppose that we have two structural modules and three clocks available, and that our synthesis system has selected those two modules and cross-coupled them as shown, furthermore assigning clocks to the latches as shown in Figure 2. The textual equivalent to the circuit in Figure 2, and which is actually input into our present analysis tool, is shown in Figure 3. A cycle borrowing analysis gives the minimum cycle time for this circuit at 185 nS (without cycle borrowing this would be 200 nS).

This circuit gives a good example of what we do *not* do. Our analysis indicates that there are no superfluous latches in this circuit. This can be seen by studying the inputs to the combinational component "R_2." The upper input (bold arrow coming directly from latch "Q_B") will change its value at the beginning of epoch number two (the third column of the S matrix in Figure 2) because of the clocking of latch "Q_B." However the lower input, coming from latch "R_B," cannot change prior to epoch three (the fourth column of the S matrix in Figure 2). We do not attempt to second-guess the designer's intention in staggering the inputs to component "R_2" in time in this manner (that problem is further investigated in [Jen92d]). For that reason, all latches perform a necessary sequencing function, and none can be eliminated.

But suppose instead that the upper input to component "R_2" was a constant value, as we now see in Figure 4. This is a seemingly insignificant change, and could occur at any point during synthesis where a variable value can be fixed as a constant. However, this change influences the circuit sequencing. There is no longer any time-staggering of input values to "R_2" to preserve, since the upper input never changes. The result is that the output of "R_2" will have the same timing as latch "R_B," and this is in fact the same as the timing of latch "Q_C." Our analysis detects that latch "Q_C" is superfluous, and we automatically remove it from the circuit, in effect connecting "R_2" directly to "Q_2." The cycle borrowing analysis now reports 175 nS attainable minimal cycle time.

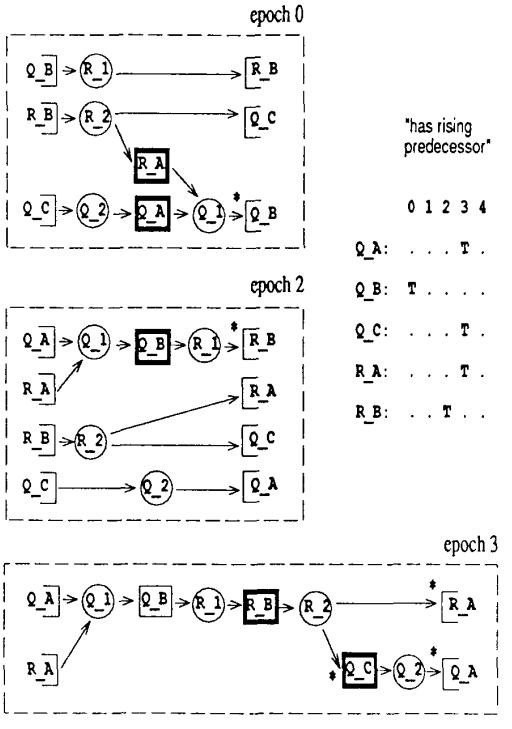

Figure 5: The graphs ${}^{\psi k}H(S)$ for $k = 0, 2, 3$, demonstrating the computations of the initialization pass. Topological sort over each graph eventually visits latches which are opening (clock rising), shown with heavy outline. The result is that all latches "downstream" from these will obtain the property "has rising predecessor" (marked with asterisk) and this is recorded in a Boolean array for each latch, indexed by epoch.

Mechanically, the analysis detects this redundancy as follows. First we must perform the initialization pass in which we build up the Boolean arrays we call "has rising predecessor" for each latch, see Figure 5 where we show three of the five graphs ${}^{\psi k}H(S)$ for $k = 0, 2, 3$. The two graphs for $k = 1, 4$ contain no latches with rising clock edges, as can be directly seen from the schedule D shown in Figure 4, so we can immediately enter *false* in entries 1 and 4 of all the Boolean arrays for all latches, see Figure 5. For the remaining three ${}^{\psi k}H(S)$ we perform the forward topological sort as discussed earlier. For the case of ${}^{\psi 0}H(S)$ in Figure 5 we see that latches "R_A" and "Q_A" are opening during this epoch (again from schedule D, Figure 4) which means that all latches "downstream" from them will inherit the property *true*. There is only one such latch, namely, latch "Q_B." Therefore we enter *true* in entry 0 of the array for latch "Q_B." Similarly we complete the analysis of all the ${}^{\psi k}H(S)$ and obtain the result shown in Figure 5.

Now we perform the latch-by-latch examination. For this we take the Boolean arrays and the schedule D, and begin to examine each separate array in turn. For the first latch, "Q_A," we know it is driven by clock "m," so we examine the first row (row "m") of schedule D. Now, for every occurrence of a rising transition, that is, for each "R" that we find in that row, we would perform the following test, but since there is only one such occurrence we demonstrate the technique using this example. We find an "R" in column 0, that is, epoch 0, so we now examine the Boolean array for "Q_A" starting at column $(0 - 1) \, mod \, 5$, that is, epoch 4. The array shows a *false* entry, so that so far this transition is possibly redundant. So we continue to the preceding epoch, epoch 3. Schedule D tells us that we may not yet terminate, since we have not yet come to the "F" entry which matches the "R" entry we started with. However, for this epoch 3 we now find the Boolean array entry of *true*, which means that this latch, as driven by the current clock "m," is *not* redundant.

However we obtain a different result when mechanically examining latch "Q_C." It is driven by clock "p," so we use the third row of schedule D (row "p") which has an "R" in epoch 3, so that we begin our examination in epoch 2. Here our reverse-order examination of the entries of the array consistently yields *false* until the "F" entry in schedule D for epoch 4 tells us that we are finished. Therefore that entire transition can be removed, so we synthesize a new clock for latch "Q_C" in schedule S replacing epochs $2, 1, 0, 4$

```
m:  10000
n:  00110
p:  00010

q_a(m)   10.0  → q_1
q_1      30.0  → q_b
q_b(p)   10.0  → r_1
q_c(p)   10.0  → q_2
q_2      20.0  → q_a
const     0.0  → r_2
r_a(m)   10.0  → q_1
r_1      40.0  → r_b
r_b(n)   10.0  → r_2
r_2      55.0  → q_c, r_a
```

Figure 6: Circuit in Figure 4, except that we have exchanged the clocks to latches "Q_B" and "R_B." The analysis will detect two superfluous latches.

(the epochs involved in the transition we just examined) with "clock-on" (1). But this new clock is now identically high, that is, this latch is always enabled, so we can now eliminate latch "Q_C" altogether.

5 Iterative Application

Consider now the circuit of Figure 4 but this time we exchange the clocks which drive the two latches "Q_B" and "R_B." A textual rendering of the resulting circuit is shown in Figure 6. If we now apply our procedure once, the clock of "R_B" will render latch "Q_C" as being useful, but latch "R_B" itself is redundant. Therefore "R_B" will be removed but "Q_C" will be retained. But if we then apply our procedure *again* on the new, resulting circuit, "Q_C" will be found redundant with respect to latch "Q_B" and so "Q_C" will itself be removed. For this reason we execute the procedure iteratively until no further optimizations are found, although in practice for conservative clock waveforms, such as two-phase non-overlapping clocking, such pathological cases do not occur.

For the example of Figure 6, we succeed in eliminating two of the original five latches and the cycle borrowing analysis on the resulting optimized circuit reports 165 nS attainable minimal cycle time, down from the 185 nS for the original circuit. The textual rendering of the original circuit (Figure 6) can be compared to that of the resulting circuit (Figure 7), as produced by our implementation. Here we can see that not only the structure of the circuit has changed (two latches have been removed) but the clocking schedule S has also been simplified, as we had discussed in Section 2.

```
m:  1000
p:  0010

q_a(m)   10.0  → q_1
q_1      30.0  → q_b
q_b(p)   10.0  → r_1
q_2      20.0  → q_a
const     0.0  → r_2
r_a(m)   10.0  → q_1
r_1      40.0  → r_2
r_2      55.0  → q_2, r_a
```

Figure 7: Circuit in Figure 6 after applying our procedure twice. Latches "Q_C" and "R_B" have been eliminated from the circuit. In addition, the clock schedule S has been simplified from five to four epochs, and from three to two clocks, that is, to simple two-phase non-overlapping clocking.

One problem with the procedure is that latches are removed without respect to path delay imbalances which can result. For example, in the above scenario, it may have been better to delete latch "Q_B" rather than latch "Q_C" to avoid the long combinational chain that results. However, this task of *retiming for level-sensitive latches* is itself a separate research problem, and we do not attempt to solve it here. Nevertheless we expect that the transformation performed by this procedure, of removing all clearly redundant latches *before* attempting to retime, will simplify the formulation of, and solution to, the retiming problem.

6 Conclusion

We have presented a technique for identifying superfluous level-sensitive latches in circuits constructed from such latches and from combinational components. The method is applicable to simple as well as complex clocking disciplines for a completely general n clocks and m phases. The performed transformations fully preserve the original sequencing of the circuit. For complex clocking schedules, the procedure may even synthesize a new clocking schedule, having no superfluous transitions. The time complexity of the analysis has been discussed, and has been shown to have a setup cost linear in the size of the circuit times the number of epochs in the timing schedule. To this setup cost there is a very small cost for each verified latch. For complex clocking schemes the procedure may have to be iteratively applied until no further optimizations are obtained.

Acknowledgements

This work was supported by the Swedish National Board for Technical Development (STU) under contract number 87-02427.

References

[Bre91] Björn Breidegard. Design of Efficient Two-phase Clocked Systems. Technical report, Dept. of Comp. Eng., Lund Univ., 1991.

[CB+90] Arthur F. Champernowne, Louis B. Bushard, et al. Latch-to-Latch Timing Rules. *IEEE Transactions on Computers*, 39(6):798–808, June 1990.

[CM88] E. Cheng and S. Mazor. The Genesil Silicon Compiler. In D. Gajski, editor, *Silicon Compilation*, chapter 9, pages 361–405. Addison-Wesley, 1988.

[DR89] Michel E. Dagenais and Nicholas C. Rumin. On the Calculation of Optimal Clocking Parameters in Synchronous Circuits with Level-Sensitive Latches. *IEEE Transactions on Computer-Aided Design*, 8(3):268–278, March 1989.

[DY90] Giovanni De Micheli and Roger Yip. Logic Transformations for Synchronous Logic Synthesis. In L. W. Hoevel and V. Milutinovic, editors, *Proceedings, 23rd Annual Hawaii Intl. Conference on System Sciences (HICSS-23), Vol I: Architecture Track*, pages 407–416, 1990.

[GD85] Lance A. Glasser and Daniel W. Dobberpuhl. *The Design and Analysis of VLSI Circuits*, chapter 6: Clocks and Communication, pages 331–374. Addison-Wesley, 1985.

[Jen92a] G. Jennings. On Cycle Borrowing Analyses for Interconnected Chips Driven by Clocks having Different but Commensurable Speeds. *(submitted to) International Conference on Application-Specific Array Processors (ASAP 92)*, August 4-7 1992. Berkeley, California.

[Jen92b] G. Jennings. On Cycle Borrowing Analyses for Pad Delays Shared over Multiple Cycles by Gated Transparent Latch Encompassing. *(submitted to) IEEE Multi-Chip Module Conference, MCMC-92*, March 17-20 1992. Santa Cruz, California.

[Jen92c] G. Jennings. On the Linear Programming Formulation of Cycle Borrowing Analyses for Level-Sensitive Latched Circuit Design. *(submitted to) 29th Design Automation Conference*, June 8-12 1992.

[Jen92d] G. Jennings. On the Simplification of Clock Waveforms for Level-Sensitive Latched Circuits having n Clocks and m Phases. Technical report, Dept. of Computer Eng., Lund University, Lund, Sweden, January 1992. (in preparation, to be submitted to) European Design Automation Conference, EURO-DAC '92, Hamburg Germany, September 1992.

[Jen92e] G. Jennings. Toward Exploiting Cycle Borrowing in Level-Sensitive Latched Circuit Design. *(submitted to) IEEE 1992 Custom Integrated Circuits Conference (CICC '92)*, 1992.

[JJ91] G: Jennings and E. Jennings. A Formal Methodology for Level-Sensitive Latched Circuit Design with n Clocks and m Phases. Technical report, Lund University, Dept. of Comp. Eng., Lund, Sweden, November 1991.

[Jou87] Norman P. Jouppi. Timing Analysis and Performance Improvement of MOS VLSI Designs. *IEEE Trans. Computer-Aided Design*, CAD-6(4):650–665, July 1987.

[Man89] Udi Manber. *Introduction to Algorithms, A Creative Approach*. Addison-Wesley, 1989. Section 7.4, Topological Sorting.

[NMN82] D. Noice, R. Mathews, and J. Newkirk. A Clocking Discipline for Two-Phase Digital Systems. In *Proc. IEEE Intl. Conference on Circuits and Computers (ICCC '82)*, pages 108–111, 1982.

[Ous85] John K. Ousterhout. A Switch-Level Timing Verifier for Digital MOS VLSI. *IEEE Transactions on Computer Aided Design*, CAD-4(3):336–349, July 1985.

[SMO90] Karem A. Sakallah, Trevor N. Mudge, and Oyekunle A. Olukotun. Analysis and Design of Latch-Controlled Synchronous Digital Circuits. In *Proceedings of the 27th Design Automation Conference*, pages 111–117, June 1990.

[Szy86] T. G. Szymanski. LEADOUT: A Static Timing Analyzer for MOS Circuits. In *Proc. IEEE Intl. Conference on Computer-Aided Design ICCAD-86*, pages 130–133, October 1986.

[Tjä91] Robert Tjärnström. Clock Independent Timing Verification of Level-Sensitive Latches. In *Proceedings of the 1991 European Conference on Design Automation (EDAC 1991)*, pages 271–275, February 1991.

[UT86] Stephen H. Unger and Chung-Jen Tan. Clocking Schemes for High-Speed Digital Systems. *IEEE Transactions on Computers*, C-35(10):880–895, October 1986.