Faster Static Timing Analysis via Bus Compression

by

David Van Campenhout and Trevor Mudge

CSE-TR-285-96

THE UNIVERSITY OF MICHIGAN

Computer Science and Engineering Division
Department of Electrical Engineering and Computer Science
Ann Arbor, Michigan 48109-2122
USA

Faster Static Timing Analysis via Bus Compression

David Van Campenhout and Trevor Mudge

Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan 48109-2122

January 15, 1996.

Abstract

Static timing analysis is used extensively in the design of high-performance processors. In this paper we present a method which transforms the circuit graph used by timing analysis algorithms to a smaller one, and hence results in faster timing analysis. This transformation, called bus compression, may lead to a more conservative analysis. However, experiments on several large designs yielded the same results with or without bus compression.

1 Introduction

The design of high performance computers is subject to stringent timing requirements. Static timing verification and optimization are used extensively to meet these requirements. In a typical design environment, a gate-level netlist is back-annotated with delay values derived directly from the layout parasitics. Static timing verification and optimization is performed on this netlist. The sheer size of today's systems results in very large netlists, and hence large execution times, especially when performing timing optimization. In this paper we investigate the effect of exploiting "design intent" to significantly reduce the size of the network to be processed by the timing algorithms. First we review a popular model for static timing verification and optimization.

2 SMO Model

A comprehensive model for analyzing the temporal behavior of synchronous systems was proposed by Sakallah, Mudge and Okulotun [1] (SMO). The SMO model assumes a multi-phase clocking system. All phases have a common clock period. During that clock period, each phase makes exactly one rising and one falling transition. The circuit is modeled by a network of synchronizers (either level-sensitive latches, or edge triggered flip-flops). The synchronizers are connected by edges corresponding to the combinational logic. The edges are labeled with the minimum and maximum delay through the logic. The synchronizers are characterized by their setup and hold times, the skew of their clock signal and its phase. For clock schedule verification, the variables in the SMO model are the minimum and maximum arrival time of signals at the input of the synchronizers, and the minimum and maximum departure time of signals at the output of the synchronizers. For clock schedule optimization, the clock period and the event times of the rising and falling transition of each phase are variable as well. The relationship between these variables combined with the setup and hold constraints of the latches constitute a set of constraints that can be relaxed to linear constraints.

For timing verification, it is checked whether a given clock schedule is valid, i.e., all setup and hold constraints are met. This problem is solved by relaxation. It has been shown [3] that convergence is obtained after at most $|L|$ iterations if the problem is feasible, where $|L|$ is the number of level-sensitive latches. Each iteration has a complexity of $O(|e|)$, where $|e|$ is the number of edges in the graph. This results in a time complexity of $O(|L||e|)$ for the complete algorithm. However, in practice convergence is obtained after just a few iterations resulting in a time complexity of $\Theta(|e|)$.

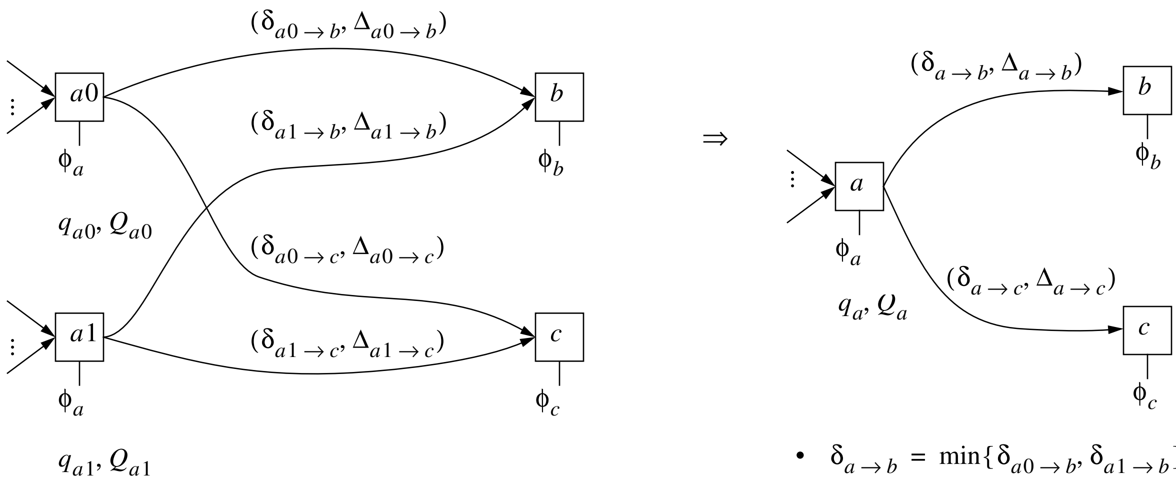
Another timing problem consists of finding the optimal clock schedule, i.e., the clock schedule with the smallest clock period that still satisfies all constraints. The optimization problem was originally tackled by linear programming [1]. However, Szymanski [4] proposed a more efficient approach. First the minimum cycle time set by the loops is computed. Subsequently all relevant constraints with respect to that cycle time are generated. The resulting linear program is much smaller in size than that of [1]. Again the time complexity of the algorithm is $O(|L||e|)$. It is clear that the number of level-sensitive latches critically affects the execution time of the algorithms.

3 Bus Compression

From the previous discussion it is clear that the time required to perform clock schedule verification and optimization is critically dependent on the number of level-sensitive latches and the number of edges in the circuit graph.

Many classes of circuits can be partitioned into a datapath and a controller. In the controller, most signals are just one bit wide. On the other hand, the signals the datapath operates on are bus signals (vectors). Some functional units in the datapath treat each of the bits in the buses identically. Multiplexers, logical units, gates are examples. In other units the relationship between the bits is more complex, e.g., in adders. The important observation is that these bus signals are single semantic entities. A bus signal is only valid if each of its bits is valid. From the perspective of timing this implies that the only events of importance are those that correspond to the latest and the earliest bit. Consequently we need only to retain one node in the timing graph for each bus synchronizer. We call the transformation from the original graph to the new graph, bus compression. The transformation is illustrated in Figure 1. The synchronizers $a0$ and $a1$ are replaced by a single synchronizer a. The resulting min-max combinational delays (δ, Δ), and the min-max clock skew (q, Q) are shown. It is very easy to show that the time to perform bus compression is $\Theta(|e|)$, where $|e|$ is the number of edges in the original graph.

Bus compression is conservative in the sense that it will never under-estimate the clock period. It can result in false negatives. For instance, if in Figure 1, $Q_{a0} > Q_{a1}$ and $\Delta_{a0 \to b} < \Delta_{a1 \to b}$, then $Q_a = Q_{a0}$ and $\Delta_{a \to b} = \Delta_{a1 \to b}$. This can lead to a conservative analysis. If the clock skew between the different bits of the same bus is negligible and the bus signal is truly a single semantic entity, then it is unlikely that buscompression will lead to false negative results.



- $\delta_{a \to b} = \min\{\delta_{a0 \to b}, \delta_{a1 \to b}\}$
- $\Delta_{a \to b} = \max\{\Delta_{a0 \to b}, \Delta_{a1 \to b}\}$
- $\delta_{a \to c} = \min\{\delta_{a0 \to c}, \delta_{a1 \to c}\}$
- $\Delta_{a \to c} = \max\{\Delta_{a0 \to c}, \Delta_{a1 \to c}\}$
- $q_a = \min\{q_{a0}, q_{a1}\}$
- $Q_a = \max\{Q_{a0}, Q_{a1}\}$

Figure 1: Original graph (left); after bus compression (right)

4 Experimental Results

The effectiveness of the approach was validated with some large designs. The first example is the core of a 32-bit ARM microprocessor [5]. The design contains 10,725 gates, and 1,007 latches. A second example is the floating-point unit of the Aurora3 processor [6]. This is a IEEE compliant double-precision floating-point processor. The netlist contains 86,800 gates and 8,518 latches. Some modifications to the original designs were made. For example, the register files were replaced by a single register addressed by the decoder logic. For each of these designs bus compression was performed using the high level description (the only information needed is which 1-bit synchronizers constitute a bus synchronizer). The design with bus compression applied is differentiated from the original design by the suffix BC in Table 1. The number of latches, $|L|$, and the number of edges, $|e|$, is indicated for each of the designs. Statistics for both designs are listed in Table 1. The time to perform clock schedule optimization and clock schedule verification is shown for both the bit-level design and the bus-level design. The execution times reported do not include the time to read the circuit description. For both designs, the algorithms yielded exactly the same results for the bit-level design as for the bus-level design. As expected the verification and optimization algorithms run significantly faster when the design is preprocessed with bus compression.

Table 1: Experimental Results

Design	$\|L\|$	$\|e\|$	verification [sec]	optimization [sec]
arm	1,007	22,206	0.50	514.70
arm-BC	116	510	0.01	1.46
a3fpu	8,518	189,503	11.13	4,197.87
a3fpu-BC	948	6,688	0.41	94.54

5 Conclusions

Bus compression, which transforms a timing graph to one in which synchronizers corresponding to the same bus signal are treated as single entities, was used as a preprocessing step in timing analysis. Although this technique can lead to a more conservative analysis, experiments showed that the results obtained did not differ from those obtained from analyzing the original design. Preprocessing the design with the bus compression transformation leads to significant speedups in the timing analysis, especially in the case of timing optimization.

Bus compression is just another instance of the strategy to exploit high level information when processing the lower-level description. This high-level information is generally information about the design intent. We believe that this strategy can be very beneficial in many related areas of timing analysis.

References

[1] K. Sakallah, T. Mudge, O. Olukotun, "checkTc and minTc: Timing Verification and Optimal Clocking of Synchronous Digital Circuits," in *ICCAD-90 Digest of Technical Papers*, pp. 552-555, November 1990.

[2] T. Burks, K. Sakallah and T. Mudge, "Critical Paths in Circuits with Level-Sensitive Latches," *IEEE Trans. VLSI Systems*, Vol. 3, No. 2, pp. 273-291, June 1995.

[3] T. Szymanski, "Verifying Clock Schedules," in *ICCAD-90 Digest of Technical Papers, 1990*, pp. 399-404, 1990.

[4] T. Szymanski, "Computing Optimal Clock Schedules," in *Proc. of the 29th Design Automation Conference,* pp. 399-404, 1992.

[5] D. Van Campenhout, "A CMOS ARM Processor," Internal Report, University of Michigan, 1995.

[6] T. Huff, "Architectural and Circuit Issues for a High Clock Rate Floating-Point Processor," Ph.D. Dissertation, University of Michigan, 1995.