Telescopic Units: A New Paradigm for Performance Optimization of VLSI Designs

Luca Benini [#] Giovanni De Micheli [#] Enrico Macii [‡] Massimo Poncino [‡]

[#] Stanford University
Computer Systems Laboratory
Stanford, CA 94305

[‡] Politecnico di Torino
Dip. di Automatica e Informatica
Torino, ITALY 10129

Abstract

In this work, we propose a technique for the automatic generation of variable-latency units that enables us to push the performance limit beyond the levels achievable with traditional synthesis approaches. The transformation can be used in conjunction with traditional design techniques, such as pipelining, to improve the overall performance of speed-critical systems. Experimental results, obtained on a large set of benchmarks, are very promising, but more work needs to be done to improve the robustness and flexibility of this optimization technique.

1 Introduction

The ever increasing clock frequency of high-performance systems pushes IC designers and synthesis tools to substantial efforts in reducing the critical path of combinational logic blocks that constrain the cycle time. Critical path optimization is often an expensive operation with a significant cost in area and power.

In this work we propose an innovative way to increase the average throughput with a small reduction in average latency. Two are the key intuitions behind our approach: First, a slow, fixed-latency unit can be transformed into a fast variable-latency unit which delivers a higher average throughput with low average latency; second, the transformation of the unit can be performed in a fully automatic way and estimates of the improvement in performance are available to the designer.

We call *telescopic unit* the final product of our automatic transformation. The name stems from its characteristic behavior: When needed, the telescopic unit requires additional cycles for terminating its computation. Seen as a black box, a telescopic unit produces two outputs: The original functional output and a handshaking *hold signal* which is activated when the functional unit cannot terminate its computation in the required cycle time. The overhead of realizing a telescopic unit consists of the circuitry needed for the generation of the hold signal. Additional circuitry may also be required in the external control logic that needs to observe the hold signal and behave accordingly. Intuitively, telescopic units represent the extension of the self-timed design paradigm to the world of synchronous circuits.

The synthesis of telescopic units entails several theoretical problems that need to be better understood. The main purpose of this paper is to propose these problems to the synthesis community. First, we summarize the essential features of telescopic units, as presented in [3]. Second, we provide a brief outline of the heuristics we developed in [3, 4] to synthesize telescopic units and we provide some experimental data on their effectiveness. Finally, we focus on the limitations of the current approach and on open issues that we are currently addressing.

Since this is a summary of work in progress, we do not have any final answer to several important questions. Nevertheless, we believe that this technique holds some promises, both for pure throughput optimization and for area optimization under area constraints.

2 ADD-Based Timing Analysis

The problem of calculating the timing response of a combinational logic block can be formulated as follows: Given a combinational block, find the set of input vectors for which the length of the critical path, under a specified mode of operation and a gate delay model, is maximum; the length of the critical path gives the overall block delay.

Given a gate g of the network and an input vector $x \in X$, where X is the set of all the *care* input vectors of the block, the arrival time at its output line, $AT(g,x)$, is evaluated in terms of the arrival times of its inputs, and the delays of its fanin connections, $d(c_j,x)$. Let c_j be the connection to pin j of gate g.

If all fanins of g have non-controlling values:

$$AT(g,x) = \max_j \{AT(c_j,x) + d(c_j,x)\}$$

If at least one fanin c_j of g has a controlling value for input $x \in X$, where X is the set of all possible care input vectors:

$$AT(g,x) = \min_j \{AT(c_j,x) + d(c_j,x) \mid c_j = \textit{controlling}\}$$

Finally, if $x \notin X$

$$AT(g,x) = -\infty$$

Differently from what happens with traditional delay analyzers, the use of the ADD-based timing analysis tool has made it possible to compute and store the length of the critical path for *each* input vector.

3 Telescopic Unit Architecture

Suppose that the objective is to increase the average throughput of a combinational unit, shown in Figure 1-a, for which the arrival time ADD $AT(g_{O_i},x)$ of each output O_i is available. Obviously, this can be done by shortening the cycle time of the unit from its original value, T, to $T^* < T$. One possible way of achieving this goal is through the addition to the combinational unit of an output signal, f_h (called the *hold* output), which takes the value 1 anytime an input vector requires more than T^* time units to propagate to the outputs of the block (see Figure 1-b).

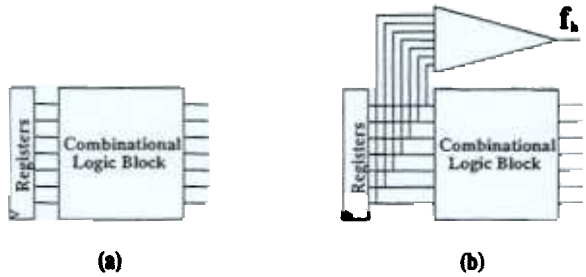

Figure A Combinational Unit (a) and a Telescopic Unit (b).

We call *telescopic unit* the modified unit, since it may require additional cycles for terminating the computation it has been designed for, depending on the specific patterns appearing at the primary input pins of the unit. In particular, the computation completes in T^* time units for patterns such that $f_h = 0$, and it completes in $2T^*$ time units for patterns such that $f_h = 1$.
Clearly, the lower the probability of the hold signal to take on the value 1, the larger the overall throughput improvement. In fact, the average throughput, P^*, of the telescopic unit is given by the following formula:

$$P^* = \frac{Prob(f_h)}{2T^*} + \frac{1 - Prob(f_h)}{T^*} \quad (1)$$

where $Prob(f_h)$ is the probability of the *hold* signal to be one. Since the average throughput of the original pipeline stage is:

$$P = \frac{1}{T} \quad (2)$$

the use of the telescopic unit is advantageous only for some values of T^* and $Prob(f_h)$, i.e., when $P^* > P$.
If we substitute Equations 1 and 2 in the inequality, we obtain the following condition for throughput improvement:

$$Prob(f_h) < \frac{T - T^*}{T} \quad (3)$$

Inequality 3 is valid only for $T^* \geq T/2$. Even though, in principle, the expression for P^* can be modified so as to account for values of $T^* < T/2$, it should be considered that in this case the circuitry needed to support the telescopic unit would become more complex, since the combinational logic may need, for some input patterns, more than two cycles to complete its computation. Here, we assume that T^* is always $T/2 \leq T^* \leq T$.

4 Automatic Synthesis of the Hold Logic

Given the arrival time ADD of output O_i, $AT(g_{O_i}, x)$, the BDD for the function $f_h^{O_i}$ which assumes the value 1 for all the input vectors for which the arrival time of O_i is greater than the desired cycle time T^* is given by:

$$f_h^{O_i}(x) = THRESHOLD(AT(g_{O_i}, x), T^*) \quad (4)$$

THRESHOLD is the ADD operator that takes two arguments: f, a generic ADD, and *val*, a threshold value, and sets to 0 all the leaves of f whose value is smaller than *val* and to 1 all the leaves of f whose value is greater than or equal to *val*. The resulting ADD, f_{val}, is thus restricted to have only 0 or 1 as terminal values; therefore, it is a BDD.
We need the input conditions for which *at least* one output O_i has an arrival time greater than T^*; f_h is then given by:

$$f_h(x) = \sum_{i=1}^{m} THRESHOLD(AT(g_{O_i}, x), T^*) \quad (5)$$

where m is the total number of block outputs. The ON-set of f_h that produces the best theoretical throughput improvement contains all and only those input values that propagate to the outputs of the unit with a delay longer than T^*. However, we need to guarantee that the hold logic itself has a delay shorter than T^*, and this may not be always possible. Thus, the target is to determine an *enlarged hold function*, $f_h^e \geq f_h$, such that the average performance of the unit only marginally degrades, but the implementation of f_h^e meets the timing constraint, T^*, and has a limited area.

We have devised two heuristics for determining and synthesizing f_h^e; they both start from the BDD representation of f_h. The first one generates the hold logic following an iterative paradigm. First, the BDD of f_h is mapped onto a multiplexor network; then, such network is optimized through traditional logic synthesis techniques; finally, a check is made to find out if the timing constraint $T(f_h) < T^*$ is met. If this is not the case, the ON-set of f_h is enlarged, to obtain f_h^e, by properly removing some BDD nodes, and the process is repeated. The second heuristics produces a sum-of-products (SOP) description of f_h^e directly from the BDD of the initial f_h.
The first heuristics runs extremely fast, but it has the drawback that the logic optimization step (namely, delay minimization under area constraints) is not as effective as it could be because of the sub-optimal network used as starting point of the optimization process. The second synthesis procedure, on the other hand, generates the representation of f_h^e in a form that can better exploit the existing logic optimization algorithms; however, execution times are much longer, since the BDD to SOP translation requires explicit cube enumeration. More details concerning the BDD-based and the SOP-based heuristics can be found in [3] and [4], respectively.

5 Results

We have implemented the synthesis procedures for the automatic generation of telescopic units, described in Section 4, as an extension of SIS [5] using CUDD [6] as the underlying BDD/ADD package. Experiments have been run on a DEC-Station 5000/240 with 64 MB of memory.
We present data concerning the use of telescopic units as a throughput optimization technique. Additional results, demonstrating the applicability of telescopic units for area optimization under throughput constraints, can be found in [3].
We have considered all the Mcnc'91 [7] combinational multi-level benchmarks with more than 100 gates (that is, a total of 53 examples). The circuits have been first optimized for speed using a version of the `script.delay` SIS script in which the `full_simplify -l` command has been dropped, and then mapped for speed with load constraints using the `map -n1 -AFG` command onto a cell library containing inverters, buffers, and two-input NAND and NOR gates. The unit gate delay model has been adopted for the ADD-based timing analysis.
We have run the BDD-based synthesis heuristics on the delay-optimized circuits trying to obtain maximum-throughput telescopic units. To accomplish this task we have specified several decreasing values for T^*, and we have synthesized the hold logic until we have found a value for which a further cycle time reduction caused a decrease in throughput.
For 39 examples the use of telescopic units has been beneficial throughput-wise. On the other hand, in 4 cases (circuits i3, i4, i6, and i7) the throughput did not increase. Finally, in 10 cases the ADD-based timing analysis did not complete.
Table 1 reports the data for the 39 examples on which throughput optimization has succeeded. Columns *Circuit*, *In*, *Out*, *Gt*, T and P give the name, the number of inputs, outputs, and gates, the true delay and the throughput of the original circuit. Column $Prob(f_h^e)$ shows the probability of f_h^e, column Gt^e gives the total number of gates of the telescopic unit, column T^* reports the reduced cycle time, column P^* indicates the improved throughput, and column $T(f_h^e)$ tells the arrival time of the hold signal. Columns ΔP and ΔGt give the throughput improvement and the area overhead of the telescopic unit. Finally, column *Time* reports the CPU time, in seconds, required to perform the timing analysis and to generate f_h^e for the given T^*.

In order to compare the effectiveness of the two heuristics, out of the 39 circuits optimized with the fast BDD-based procedure, we have chosen the ones for which either the throughput improvement was smaller than 10%, or the area penalty was larger than 10%. A total of 16 examples has thus been selected; the SOP-based procedure has been run on the reference versions of such examples for heavy-duty optimization of f_h^e.
Table 2 reports, for each circuit, the results obtained with the BDD-based and the SOP-based heuristics.
Our primary interest was the evaluation of the impact of the SOP-based procedure on the area of the telescopic units. However, in order to make the comparison of the two heuristics as fair as possible, we have not allowed any throughput degradation with respect to the units obtained through the BDD-based procedure. In addition, we have decided to keep the values of the reduced cycle time, T^*, fixed, that is, the ones that were used for the BDD-based synthesis; this is for the purpose of better identifying the effects of the synthesis heuristics on the implementation of the hold logic.
The results of the comparison are in favor of the SOP-based approach by an amount which goes beyond our expectations. In fact, not only the average area overhead has decreased from 13.4% to 10.8%, but a further average throughput increase from 17.0% to 18.8% has been achieved as a by-product. In a few cases, the worst-case delay of the hold logic has also decreased. As expected, the SOP-based heuristics is slower than the BDD-based one. Even though in most of the cases the difference in running time is negligible, there are examples where the SOP-based synthesis has required several minutes to complete.

6 Open Issues

In spite of the encouraging results, several open issues still need to be addressed before telescopic units can be considered as a robust and flexible design option. We are considering three directions of improvement.

Conservative Timing Analysis. ADD-based exact timing analysis cannot handle many large circuits. If the size of the ADD representing the circuit delays is such that it cannot fit in memory, we do not have the information needed for generating the hold logic. More work needs to be done for developing robust algorithms for the computation of the delay information. Obviously, since even the problem of finding the true critical path of a network is NP-complete, the accuracy of the delay computation must be relaxed. Several approaches have been proposed for the efficient computation of conservative delay estimates. The integration of such algorithms within the procedures for the synthesis of telescopic units is an interesting problem.

Improved Algorithms for f_h^e Generation. Although the results achieved by the heuristics for synthesis of the hold logic are quite good, there is margin for improvement. Observe that our techniques are based on increasing the size of the ON-set of the original f_h. Although our ON-set extension procedures are directed to reducing the estimated cost of the final implementation, our estimates have limited accuracy, because they do not directly take into account the impact of multi-level synthesis on f_h^e. An alternative approach is to generate f_h^e not by increasing the ON-set, but by creating and expanding a DC-set. In this way, the multi-level synthesis process could use the additional degrees of freedom only if needed to achieve a better multi-level implementation. This cannot be done if the ON-set is expanded, because we are specifying a different function, and not just degrees of freedom for its implementation.

Interaction with the Environment. Replacing fixed-latency units with variable-latency ones complicates the control flow. If telescopic units are instantiated in the data-path, the controller's complexity increases, because some handshaking is required to be able to control and conditionally stop the flow of data into the system. We need to guarantee that the increase in complexity of the controller does not off-set the benefits of using the telescopic units.
Once these problems are solved it would be possible to design complex systems where some or all functional units are telescopic. At this stage, another open issue is the estimation of the average throughput of a system with multiple variable-latency units. Even if much work needs to be done, we believe that telescopic units represent the basic building block for a new design paradigm, where it will be possible to achieve extremely high average throughput at the expense of a marginal increase in area.

7 Conclusions

We have presented a technique for the automatic generation of variable-latency units that allows us to push the performance limit beyond the levels achievable with traditional synthesis approaches. Thanks to symbolic exact delay computation, we identify the input conditions for which the propagation through the original logic takes longer than the cycle time. We then generate a combinational logic block which communicates to the environment when the correct result is available at the unit register boundaries. Experimental results, collected on a large set of standard benchmarks, have shown that our technique is valuable as performance-enhancement tool; in addition, telescopic units can serve as throughput-constrained area optimization devices (see [3] for more details). We have also discussed several open issues and directions for improvement that could increase the robustness and generality of our optimization paradigm, and make it viable as a practical design alternative in real-life systems.

Acknowledgments

We wish to thank Iris Bahar for helping us with the ADD-based timing analysis code, and Fabio Somenzi for useful suggestions on the use of the CUDD package.

References

[1] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, F. Somenzi, "Algebraic Decision Diagrams and their Applications," ICCAD-93, pp.188-191, Santa Clara, CA, Nov. 1993.

[2] R. I. Bahar, H. Cho, G. D. Hachtel, E. Macii, F. Somenzi, "Timing Analysis of Combinational Circuits using ADDs," EDTC-94, pp. 625-629, Paris, France, Feb. 1994.

[3] L. Benini, E. Macii, M. Poncino, "Telescopic Units: Increasing the Average Throughput of Pipelined Designs by Adaptive Latency Control," DAC-34, Anaheim, CA, Jun. 1997, To Appear.

[4] L. Benini, G. De Micheli, E. Macii, M. Poncino, "A Cube-Based Heuristics for the Automatic Synthesis of Variable-Latency Units," EuroDAC-97, Dusseldorf, Germany, Nov. 1997, Submitted for Publication.

[5] E. M. Sentovich, K. J. Singh, C. W. Moon, H. Savoij, R. K. Brayton, A. Sangiovanni-Vincentelli, "Sequential Circuits Design Using Synthesis and Optimization," ICCD-92, pp. 328-333, Cambridge, MA, Oct. 1992.

[6] F. Somenzi, CUDD: University of Colorado Decision Diagram Package, Release 2.1.2, Technical Report, Dept. of ECE, University of Colorado, Boulder, CO, Apr. 1997.

[7] S. Yang, Logic Synthesis and Optimization Benchmarks User Guide Version 3.0, Technical Report, MCNC: Microelectronics Center of North Carolina, Research Triangle Park, NC. Jan. 1991.

pcler8													
mux													
cordic													
frg1													
sct													
unreg													
b9													
											6.9%	10.6	
											8.6%	4.8	
x4													
alu2													
i9													
rot													
t481													
frg2													
dalu	75	16	1316	23	0.04	0.373	1394	16	0.05	6	16.9%	5.9%	59.6
vda	17	39	1416	12	0.08	0.026	1447	11	0.08	10	7.6%	2.2%	10.7
alu4	14	8	1457	39	0.02	0.359	1520	20	0.04	17	59.9%	4.3%	53.3
i8	133	81	1485	16	0.06	0.068	1519	13	0.07	11	18.8%	2.2%	32.1
pair	173	137	1956	28	0.03	0.008	2027	24	0.04	18	16.1%	3.6%	68.7
k2	45	45	2393	18	0.05	0.081	2571	16	0.05	15	7.9%	7.4%	60.1
des	256	245	5084	27	0.03	0.015	5119	24	0.04	7	11.6%	0.6%	73.2

Table 1: Throughput Optimization Using the BDD-Based Synthesis Heuristics.

Circuit	In	Out	Gt	T	P	Heur.	Prob(f_h^e)	Gt*	T^*	P^*	$T(f_h^e)$	ΔP	ΔGt	Time
mux	21	1	106	14	0.0714	BDD	0.05070	145	12	0.0812	9	13.7%	36.8%	1.8
						SOP	0.05070	143	12	0.0812	9	13.7%	34.9%	5.6
cordic	23	2	126	15	0.0667	BDD	0.05270	153	11	0.0885	10	32.8%	21.4%	1.3
						SOP	0.02636	148	11	0.0897	10	34.5%	17.5%	7.1
f51m	8	8	152	11	0.0909	BDD	0.10900	165	10	0.0946	7	4.0%	8.5%	1.1
						SOP	0.10900	165	10	0.0946	7	4.0%	8.5%	1.2
comp	32	3	174	21	0.0476	BDD	0.00366	220	19	0.0525	9	10.3%	26.4%	3.8
						SOP	0.00005	216	19	0.0526	9	10.5%	24.1%	820.5
cht	47	36	209	6	0.1666	BDD	0.25000	211	5	0.1750	2	5.0%	0.9%	0.8
						SOP	0.25000	211	5	0.1750	2	5.0%	0.9%	1.0
my_adder	33	17	225	34	0.0294	BDD	0.13600	286	18	0.0518	9	76.0%	27.1%	10.1
						SOP	0.03515	261	18	0.0545	7	85.6%	16.0%	672.3
term1	34	10	242	17	0.0588	BDD	0.10232	272	11	0.0862	10	46.5%	12.3%	5.3
						SOP	0.10232	272	11	0.0862	10	46.5%	12.3%	173.7
9symml	9	1	252	14	0.0714	BDD	0.03700	303	13	0.0755	9	5.7%	20.2%	5.7
						SOP	0.03610	290	13	0.0755	9	5.7%	15.1%	5.5
C432	36	7	404	27	0.0370	BDD	0.00052	435	26	0.0385	10	3.8%	7.7%	69.3
						SOP	0.00052	414	26	0.0385	9	3.8%	2.5%	58.8
too_large	38	3	417	21	0.0476	BDD	0.01921	462	17	0.0582	7	22.3%	10.7%	36.7
						SOP	0.01921	462	17	0.0582	7	22.3%	10.7%	751.0
x4	94	71	498	12	0.0833	BDD	0.16682	539	10	0.0916	9	9.9%	8.2%	8.8
						SOP	0.02434	510	10	0.0987	8	18.5%	2.4%	796.3
rot	135	107	840	23	0.0434	BDD	0.20955	937	19	0.0471	18	8.5%	11.5%	582.1
						SOP	0.15125	916	19	0.0486	16	12.0%	9.1%	1226.0
apex6	135	99	889	17	0.0588	BDD	0.00020	905	16	0.0625	10	6.2%	1.8%	6.8
						SOP	0.00020	905	16	0.0625	10	6.2%	1.8%	13.1
t481	16	1	1043	22	0.0454	BDD	0.16462	1151	18	0.0509	17	12.1%	10.3%	45.9
						SOP	0.10426	1137	18	0.0526	16	15.8%	9.0%	481.0
vda	17	39	1416	12	0.0833	BDD	0.02680	1447	11	0.0897	10	7.6%	2.2%	10.7
						SOP	0.01916	1430	11	0.0901	9	8.1%	1.0%	16.5
k2	45	45	2393	18	0.0555	BDD	0.08120	2571	16	0.0599	15	7.9%	7.4%	60.1
						SOP	0.08120	2558	16	0.0599	15	7.9%	6.8%	138.4