Performance Optimization of Self-Timed Circuits

Mark A. Franklin and Prithvi Prabhu[1]

Computer and Communication Research Center
Washington University
St. Louis, Missouri, 63130-4899
U.S.A.

Abstract

In this paper, we present methods for improving the performance of self-timed computation blocks. The Hybrid Completion method permits the design of a spectrum of completion circuits ranging from those based on pure bounded delays to those based on full complementary circuit development. This is achieved by using a subset of the outputs of the computation block to generate the overall completion signal. Thus, the extra circuitry for the completion signals of the other outputs is eliminated. The computation block's delay might also be reduced since fewer signals are required to generate the overall completion signal. The approach seeks to incorporate the area efficiency of the bounded delay approach and the operand based delay sensitivity of the full complementary approach.

1 Introduction

In recent years, asynchronous design methodologies have attracted increased attention. The reasons for this stem from the increasing power dissipation and clock distribution problems associated with the design of large clocked microprocessors. In contrast, asynchronous circuits do not require clock routing, have no associated clock skew, and have the potential for lower power consumption. In addition, asynchronous performance is determined (in part) by average versus worst case time delays, and asynchronous design methodologies have the potential of leading to a more modular design style.

However, there are drawbacks associated with asynchronous design. These are due to its increased design complexity and chip area requirements, and handshaking overhead. Work is progressing at automating the synthesis of self-timed circuits [8] and at reducing handshaking overheads due to synchronization [12]. In this paper we present a new approach to developing completion signals which indicate that a computation has finished.

We assume in this paper that we are dealing with bounded delays on wires and circuits, and that a bundled data protocol is employed[2]. Figure 1 shows an asynchronous pipeline where two types of blocks are present: computation blocks and interconnection blocks [8]. Computation blocks perform processor operations and interconnection blocks control transfers of data between the computation blocks. Note that in addition to calculating the required data outputs, computation blocks also have to generate completion signals to indicate that valid data outputs are present and to signal the interconnection blocks to initiate a data transfer operation.

The completion signal may be derived as a function of each of the computation block's outputs as shown in Figure 2. A simple approach to implementing the completion circuit is to first determine the maximum delay associated with each output and create a delay element which is equal to the overall maximum. By placing this delay in parallel with the computation block and initiating the delay with the Request signal, the completion signal can be generated at the delay output. This is referred to as the *bounded delay* approach [11] and is shown in Figure 3.

While this approach is area efficient, the delay associated with the computation is fixed at its maximum, and therefore there is no ability to take advantage of variations in completion times which may arise due to operand variations. Another approach, the *complementary circuit* approach, develops the complementary circuits associated with the outputs. Completion is signalled when either the output or its complement has been computed. A overall completion signal for

[1] Currently with Intel Corp.

[2] We are dealing with self-timed but not speed-independent circuits and assume that the propagation times of the control signals are greater than or equal to that of the bundled data signals.

0-8186-8409-7/98 $10.00

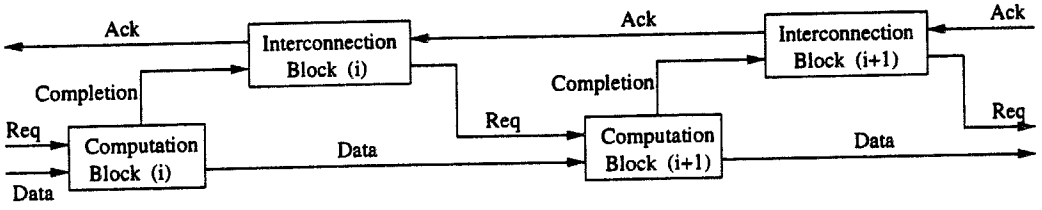

Figure 1: An asynchronous pipeline with Computation and Interconnection blocks

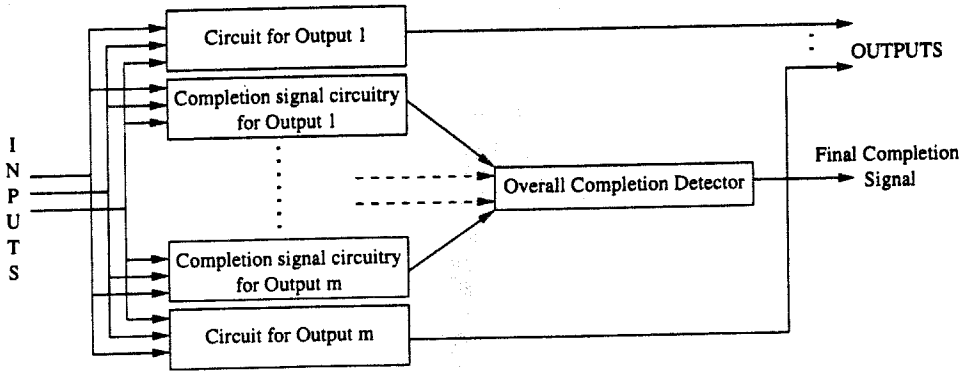

Figure 2: A Computation block

the computation block can be obtained by checking if all the individual completion signals have been asserted (using multi-input or a tree of C elements). Generally, this will require more area than the pure bounded delay approach with the area increasing directly with the number of outputs [12]. However, the completion signal now reflects average rather than worst case computation delays.

In this paper, we present the *Hybrid Completion (HC)* method which implements a spectrum of completion circuits whose performance ranges from the fixed delay, low area case of the "pure bounded delay" to the average delay, large area case of the "pure complementary circuit" approach. The HC method uses the completion circuits associated with a subset of the outputs. This may decrease overall area requirements since the extra circuitry needed to produce completion signals for the remaining outputs is not required. Basically we reduce the number of "completion signal circuitry" boxes present in Figure 2. Also, since fewer completion signals are developed, the circuitry necessary to produce the overall block completion signal is reduced (i.e., this reduces the complexity of the "overall completion detector" box in Figure 2).

There are numerous implementation options with the hybrid approach. Thus, part of this method consists of enumerating and selecting the best hybrid circuit to employ. In this paper we show for several simple examples that the hybrid approach yields circuits which are more area efficient than circuits developed using the pure complemetary approach while having comparable delays. The approach also yields circuits having lower delays than those produced by the pure bounded delay approach, although requiring somewhat more area.

1.1 Related Work

Extensive work has been done on formalizing and designing the various types of asynchronous circuits [8, 12]. Other distinctive methods of completion detection have also been proposed, including the use of an internal clock within a computation module [3], and the use of current sensing techniques for completion detection [6, 4] etc.

In this paper, circuit delay information available is used to analyze the design of completion signal generators. Using delay information in optimization of asynchronous circuits is not new [9, 7, 5]. In [7], the design optimization is performed using techniques which required the addition of delay elements to avoid circuit hazards. In [9] an *Event Rule* (ER) system [1] is used to specify asynchronous circuits and the timing information available with the ER specification is used to optimize the design of asynchronous control circuits. It is shown in [1] that specifications that

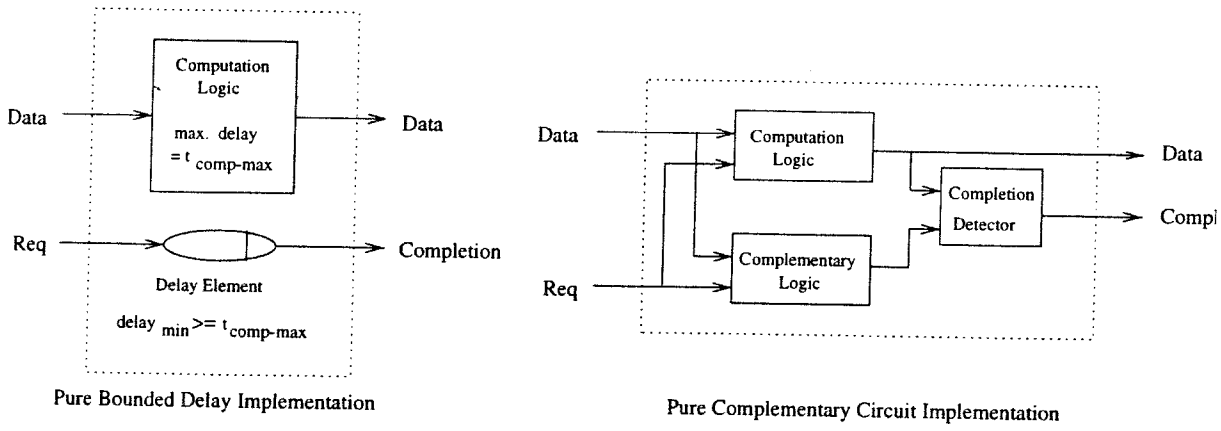

Figure 3: Pure Completion Signal Methods

are not *data dependent* or *disjunctive* can be transformed into ER systems. Due to these reasons, ER specifications (such as Signal Transition graphs [2]) can only be used to synthesize asynchronous control circuits (Finite state machine controllers, Interconnection blocks etc.). The circuits implemented using the ER specification include the state holding elements.

In the pipeline model of design, the computation circuitry is disassociated from the state holding elements. Request and Completion signals are required for this implementation with the state holding function being handled by the interconnection block. The computation blocks considered in this work are implemented using purely combinatorial elements. Several circuits (*e.g.*, adders, multipliers etc.) are implemented as computation blocks which are a part of a pipeline and not as control circuits. Such circuits can be analyzed by the method outlined in this paper. The optimizations made using the ER approach are not compared with those presented in this paper since the basic design methodologies used by the two approaches are different. This research concentrates on the computation blocks (which are a part of a computational pipeline) and specifically on how the outputs and their delays affect the design of the completion signal generators.

Related work done on optimizing asynchronous circuit performance of systems following the pipelined model has been done in the AMULET ALU [5] project. With this system, the ALU implementation signalled the end of computation in a manner that depended on the instruction being executed. If an add instruction was encountered, the completion was signalled when a carry signal had been transmitted to all the bits in the word. If a bitwise instruction was encountered, then a fixed delay element was used to produce the completion signal. Thus, this design approach was similar in spirit to the hybrid design methodology in that neither the fully complementary circuit nor the pure bounded delay implementations were used. However, this design approach was not extended further to other circuits, nor have we found a formal description of this method. This work seeks to extend and formalize this hybrid design approach for general computation block circuits and also to consider tradeoffs between the different modes of generating completion signals.

The hybrid method is discussed in Section 2. Section 3 presents some experimental results and the final section gives our conclusions.

2 Hybrid Completion Approach

A completion signal should be generated by a computation block once all of its outputs are valid. However, instead of using all the outputs to produce the overall completion signal, a subset of the outputs can be used, and, along with a fixed delay in series, a valid overall block completion signal can be produced. The use of a subset of outputs can improve performance in a number of ways:

- The overall area requirements are reduced since, instead of all outputs, only a subset of outputs require completion circuits.
- With fewer inputs to the overall completion detector the area and delay required for generating the overall completion signal is reduced [8, 12].
- The removal of the completion generators for some of the outputs can also result in a reduction of power dissipation.

We begin by introducing the nomenclature used in the development (see Table 1). A computation block

Table 1: Notations used in this Paper

I	Input node set
O	Output node set
j	Output node number
i	Input combination
$D_{i,j}$	Output j delay, Input combination i
D_i	Maximum delay, Input combination i
D	Average delay of the computation block
T_{comp}	Overall Completion generation time

consists of a set of inputs I (with n elements), and a set of outputs O (with m elements). There are a total of 2^n input combinations and, for each of these, each output will have a particular delay. D_{ij} denotes the delay associated with output j for the input operand combination i. Using this set of delays, the performance of different hybrid designs can be obtained.

In the method presented in this paper, for simplicity, we assume that a separate circuit is present for each output. D denotes the average delay of the computation block and is determined by the sum of the computation circuit delay and the overall completion detection delay. Delay for the overall completion detection circuitry (T_{comp}) will depend, in part, on the number of computation block outputs used to produce the overall completion signal.

2.1 Analysis of Outputs

Consider the set of outputs, O. The minimum average delay might be achieved if all of the outputs in O are used to generate the completion signal for the computation block (*i.e.*, full complementary circuit implementation). The completion circuitry for a subset of these outputs, O^*, can be used in series with a fixed delay element to produce the overall completion signal (Figure 4).

Consider the set of all 2^m possible subsets of O. For a particular subset, O^*, a partial completion signal may be generated by using the complementary circuit approach for this subset. If a fixed delay of appropriate size is now added to this partial completion signal, an overall completion signal can be developed. Correctness is guaranteed by ensuring that the partial completion signal derived from the subset O^* when added to the fixed delay goes high only after all other outputs have been computed. Thus, the value of the fixed delay depends on the elements of the subset selected.

The pure bounded delay situation corresponds to the NULL subset of O. In this case the overall completion signal is generated by use of a single fixed delay for the entire computation block. Define D_i as the maximum delay for input combination i. For a pure bounded delay implementation, the delay is given by:

$$D_{bd} = \max_i (D_i) + T_{comp} \tag{1}$$

In this case, T_{comp}, the overall completion detection time is 0 and $D_{bd} = \max(D_i)$.

At the other extreme consider the case where the subset consists of all elements in O. In this situation all the complementary circuits of the elements of O are used to develop the partial completion signal (*i.e.*, the pure complementary circuit implementation) and the value of the fixed delay in series is 0. Equation 2 gives the average delay for this situation (assume that all input combinations are equally likely).

$$T_{pcc} = \frac{1}{2^n} \sum_{i=0}^{(2^n-1)} D_i + T_{comp} \tag{2}$$

While one would expect the average delays of the hybrid implementations to be between the pure bounded delay and pure complementary cases, this is not always the case. The overall average delay of some of the subsets can be lower than T_{pcc}, depending on their respective overall completion detection times

Consider a subset of O, O^* (with m^* elements). Let D^*_{ij} be the delay for input operand combination i for the jth output in O^* ($j = 1, \ldots, m^*$). Denote D^*_i as the maximum delay for the input combination i over all the outputs in O^*. That is:

$$D^*_i = \max_j (D^*_{ij}) \tag{3}$$

If the outputs in O^* with their complements are being used to develop a partial completion signal then an additional delay must be added to ensure that the overall completion always occurs after that of the outputs in $O - O^*$. That is, there are some input combinations where the output which has the maximum delay may not be an element of O^*. The value of the additional delay (T) is given by:

$$T = \max_i (D_i - D^*_i) \tag{4}$$

where i varies from $0, 1, \ldots, 2^n - 1$. Thus, the average delay, D^*, when the outputs of subset O^* are used and all input combinations are equally likely is given by:

$$D^* = T + \frac{1}{2^n} \sum_{i=0}^{(2^n-1)} D^*_i + T_{comp} \tag{5}$$

This average delay can be found for each of the subsets of O and the one with the best performance would be chosen for implementation.

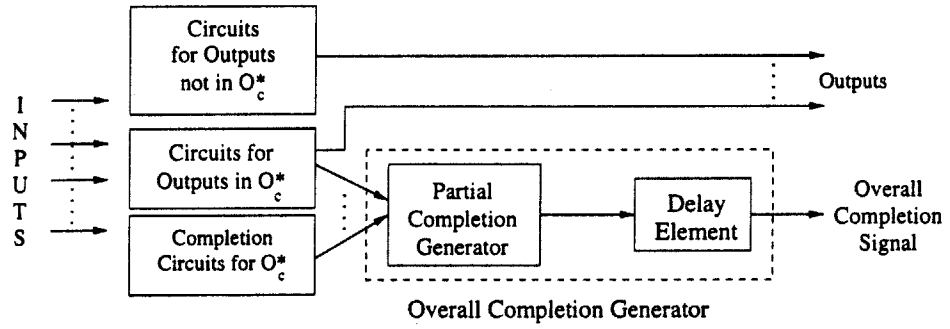

Figure 4: Computation Block using the Hybrid Completion Method

3 Experimental Results

In this section specific examples are considered and for each example the HC method is used obtain the final design. Adders, a 4-bit comparator and two state machines are used as examples. The computation blocks analyzed here are assumed to be in a single stage pipeline (see Figure 1). In all cases, the overall boolean functions of the outputs were determined and used in designing the circuits. For simplicity, all transistors were taken to be of equal size and there was no buffering for input signals driving high capacitive loads. The designs were thus not optimized for delays. Similar analysis can be performed for designs optimized for delays, though the results might differ since the delays will be different.

The circuits were implemented using DCVSL logic [8] and circuit layout was done using the MAGIC layout tool. The SPICE parameters were extracted from the circuit layouts and the timing simulations were performed by feeding these parameters to the CAzM timing simulation tool (using a 1.2 μm FET model). The delay for each output was taken to be the time for the completion signal of the output to go high after *Request* goes high. The overall completion signal was produced by using a tree of 2-input C elements. The delay of the computation block was taken to be the time for this overall completion signal to go high after *Request* goes high. The unoptimized (*i.e.*, full complementary) computation block used all the output completion signals to produce the overall completion signal. It was assumed that all input combinations were equally likely.

While only the results for the 4 bit adder are presented here, details for the other circuits can be found in [10]. The choice of the subset of outputs, O is considered using the HC method. From Table 2 it can be seen that subset $Sum2$ has the best performance in terms of average delay. Note that multiple output subsets generally have higher delays than single output subsets. This is due in part to the added delay of the overall completion signal generator, which adversely affects the performance of those subsets with more than one element for this case.

Table 2: Performance of some of the subsets of the 4-bit adder

Subset	Additional delay (ns)	Area (λ^2)	Av. delay in ns (D)
NULL	8.37	**95648**	8.47
Sum2	1.72	112800	**7.67**
Sum3	1.78	118800	8.52
Carry	4.63	124310	9.92
Sum1,2	1.72	115995	10.37
Sum2,3	0.86	123927	9.16
Sum1,2,3	0.17	129580	9.73
Unoptimized	0.00	148617	11.01

Table 3 gives the performance gains achieved by using the HC method for the different computation blocks. In all cases the delay and area values were obtained from simulations and the circuit layouts. However, an exhaustive analysis of all the design choices and the extraction of delays in each case is not necessary. A heuristic is presented in [10] which reduces the number of subsets to be considered from 2^m to m^2 without a significant sacrifice in performance. Methods of estimating the area and delay of various design choices are also presented in [10]. If the estimation methods are used, then the best design choice can be identified without the layout and simulation of the unoptimized block being required. The computational requirements for implementing the methods and identifying the design with a good performance can thus be significantly reduced.

Table 3: Performance improvements for subsets with least average delay

Adder	Computation Block	Area (λ^2)	Delay (ns)
2-bit	Unoptimized	67017	8.66
Adder	HC result	54978	4.59
3-bit	Unoptimized	109120	9.17
Adder	HC result	81048	6.71
4-bit	Unoptimized	148617	11.01
Adder	HC result	112800	7.67
5-bit	Unoptimized	209588	11.99
Adder	HC result	153816	7.87
4-bit	Unoptimized	107341	8.86
Comparator	HC result	76713	7.56
DMA	Unoptimized	102201	10.45
controller	HC result	58786	5.76
Toy CPU	Unoptimized	790485	18.31
controller	HC result	546744	14.38

4 Conclusion

In this paper we have presented a method for improving the performance of asynchronous computation blocks (which are a part of a pipelined system) by focusing on the problem of completion detection. The Hybrid Completion method helps in analyzing completion generators which use both the complementary circuit and the bounded delay approach. Using this method, considerable performance gains with respect to average delay and area are possible. Detailed knowledge of the circuit delays is required and the methods are computationally intensive. However, using delay and area estimates, the analysis time required to arrive at a near optimal design can be significantly reduced. Since this is a one time operation done at design time, the investment in time and effort may be worthwhile.

We have illustrated how this method helps in improving the performance of 2, 3, 4 and 5-bit adders, a 4-bit comparator and 2 finite state machines. Performance gains in average delay ranging from 14.7% to 47.0% were achieved with the average improvement over the 7 computation blocks being 28.2%. Future work includes improving the estimation methods so that they can be applied to more circuits. The goal of finding good and often optimal design choices for completion signal generation in asynchronous computation blocks can be achieved by using the methods presented in this paper.

References

[1] Steve Burns. *Performance Analysis and Optimaztion of Asynchronous Circuits.* PhD thesis, California Institute of Technology, 1991.

[2] Tam-Anh Chu. *Synthesis of Self-Timed VLSI Circuits from Graph theoretic Specifications.* PhD thesis, Massachusetts Institute of Technology, 1987.

[3] A.L. Davis. The architecture and system method of DDM-1: A recursively-structured data driven machine. In *Proc. fifth Annual Symposium on Computer Architecture*, 1978.

[4] Mark E. Dean, David L. Dill, and Mark Horowitz. Self-timed logic using current-sensing completion detection (CSCD). In *Proc. ICCD*, pages 187–191, October 1991.

[5] J.D. Garside. A CMOS VLSI Implementation of an Asynchronous ALU. In *Proc. IFIP Conf. on Asynchronous Design Methodologies*, Manchester, England, March 1993.

[6] E. Grass and S. Jones. Asynchronous circuits based on multiple localised current-sensing completion detection. In *Asynchronous Design Methodologies*, pages 170–177, May 1995.

[7] Luciano Lavagno, Kurt Keutzer, and Alberto Sangiovanni-Vincentelli. Algorithms for synthesis of hazard-free asynchronous circuits. In *Proc. ACM/IEEE Design Automation Conference*, pages 302–308. IEEE Computer Society Press, 1991.

[8] T.H. Meng. *Synchronization Design for Digital Systems.* Kluwer Academic Publishers, Norwell, MA, 1991.

[9] Chris Myers and Teresa H.-Y. Meng. Synthesis of timed asynchronous circuits. In *Proc. International Conference on Computer Design*, pages 279–282. IEEE Computer Society Press, October 1992.

[10] Prithvi Prabhu. Performance optimization of self-timed circuits. Master's thesis, EE, Washington University, St. Louis, July 1996.

[11] I.E. Sutherland. Micropipelines. *Commun. ACM*, pages 720–738, June 1989.

[12] B.K.V. Sarma Wuu Tzyh-Yung. Design of fast and area efficient Multi-input Muller C-elements. *IEEE Trans. VLSI*, pages 215–219, June 1993.