Design of Self-timed Multipliers: A Comparison[1]

Jens Sparsø, Christian D. Nielsen, Lars S. Nielsen, and Jørgen Staunstrup

Department of Computer Science, Building 344, Technical University of Denmark, DK-2800 Lyngby, Denmark. (E-mail: jsp@id.dth.dk)

Abstract
In recent years a number of methods for designing self-timed circuits have been proposed. As a first step towards a comparison of these, we have designed a vector-multiplier using three of the published approaches: a Caltech design [5, 7], a multi-ring design [10, 11] (both are delay-insensitive circuits using four-phase handshaking and dual-rail encoding of data), and a micropipeline design [12, 8] (using a two-phase bundled-data protocol). Furthermore, a synchronous design of the same multiplier has been made. All the designs have been completed down to the layout level, and chips for two of the self-timed designs have been fabricated and tested. Based on these design experiments we report both quantitative and qualitative comparisons.

Keyword Codes: B.4.3; B.7.1; B.2.1;
Keywords: Input/Output and Data Communications, Interconnections (subsystems); Integrated Circuits, Types and Design Styles; Arithmetic and Logic Structures, Design Styles;

1. INTRODUCTION

This paper presents a comparison of four different solutions to a design problem. Three of these are asynchronous designs and the fourth is a synchronous design, mainly given as a frame of reference. All the designs have been completed down to the layout level. Based on these design experiments we report a number of quantitative and qualitative comparisons.

A vector multiplier has been chosen as the design problem on which the comparisons are based. For many applications, e.g. signal processing, the multiplier is the bottleneck both with respect to area and computation time. Therefore, it is important to develop efficient multipliers, and whereas adders were among the first published self-timed designs, multipliers have received less attention in the literature.

[1]This has work has been supported by The Danish Technical Research Council and by DARPA. The CAD tools from Mentor Graphics Inc. have been provided by EUROCHIP. The second author is grateful to Alain Martin and his group for the possibility to gain insight in the Caltech design method during a visit to Caltech.

2. THE VECTOR MULTIPLIERS – ALGORITHM AND ARCHITECTURE

A common operation in signal processing and many other applications is to compute a sum of products, for example, the inner product of two vectors. Such a design is called a vector multiplier, and it is the design example used in this paper. The input is a stream of operand pairs (integers) along with a tag indicating the last pair of operands. Output from the circuit is the accumulated sum of products.

An iterative serial-parallel multiplication algorithm implementing the "paper and pencil approach" is used, see figure 1. In each iteration step the circuit performs a multiply, add and shift operation, corresponding to the processing of one row of bit products. The two operands are called the serial operand and the parallel operand respectively.

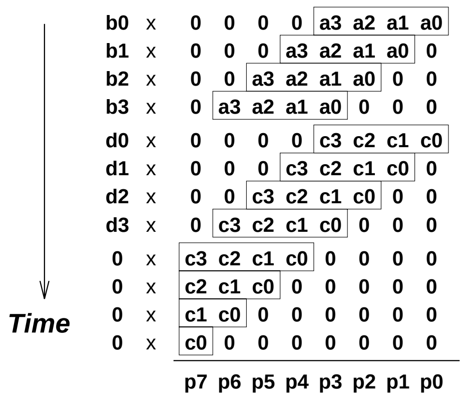

Figure 1. Example of a serial-parallel calculation of an inner product, $P = A \times B + C \times D$ where $P = p_7 p_6 p_5 p_4 p_3 p_2 p_1 p_0$ is an 8 bit unsigned integer, and where A, B, C and D are 4 bit unsigned integers.

The vector multiplier design is inspired by a systolic neural network architecture proposed in [4]. Briefly, this architecture performs repeated matrix-vector multiplications in a systolic fashion on a ring of vector multipliers. The matrix elements represent the connectivity weights in the network, and the majority of the matrix elements are either zero (representing no connection between neurons) or small numbers whose binary representation contains many leading zeros. The vector multiplier has been optimized to take advantage of the many leading zeros – it skips the corresponding multiplication steps.

To avoid ripple-carry propagation in each iteration step, the temporary result is represented in carry-save form. Conversion into binary representation is postponed until after the last two vector elements have been multiplied, and the conversion is then done by taking the circuit through a number of additional iteration steps as indicated in figure 1.

This algorithm represents a good compromise between area and speed, and its implementation involves some interesting and non-trivial circuit structures. The vector multiplier is therefore well suited for experiments with different design methods.

Implementation of the algorithm requires: (1) A multiply-accumulate block in which the result is gradually formed. (2) A shift register (with parallel load) for shifting the

Table 1
Characteristics of the four vector multiplier designs.

	Operands	Result	Technology	CAD-tools
Caltech	4 bit	8 bit	MOSIS 2 μm	Caltech and MAGIC
Micropipeline	4 bit	10 bit	EUROCHIP	Mentor Graphics:
Multi-ring	4 bit	10 bit	1.5 μm	GDT designer
Synchronous	4 bit	10 bit	CMOS	AutoCells

parallel operand, extended with zeros at both ends, one place to the left in each iteration. Depending on how the interface to the environment is specified some additional blocks may be needed: (3) A shift register with parallel load for the serial operand, and (4) a small finite state machine based control block.

The number of bits in the multiply-accumulate block and in the parallel operand shift register is the same as the number of bits in the result, and in all four designs we report, these two blocks have been implemented together using a bit-slice technique.

The Caltech design [7] implements the combined multiply-accumulate and shift block (MAS block), whereas the other designs implement all the above mentioned four blocks. The comparisons in section 5 is based on a bit-slice of the MAS block. Table 1 shows the main characteristics of the designs. All designs have been laid out and simulated. The Caltech design and the multi-ring design have been fabricated and successfully tested.

3. THE DESIGN METHODS

This section reviews the main characteristics of the four different design methods and introduces the overall structure of the corresponding vector multiplier designs. Section 4 describes some important circuit level issues that are relevant for a comparison of the methods. For an in-depth description of the designs we refer to [7, 8, 10, 11].

3.1. The Caltech design

In the Caltech design method a circuit is described as a set of communicating sequential processes [5]. The aim of the design process is to go from the sequential process description into an equivalent description of a parallel computation (with implicit sequencing). This is done through a series of transformation steps, e.g. re-shuffling, handshake expansion, state assignment, and transistor sizing. The resulting realization in VLSI is delay-insensitive with the exception of local isochronic forks.

Processes communicate via channels implemented with a four-phase handshake protocol using dual-rail encoding of data. Each process is decomposed into a control part and a data path, see figure 2. Data are received from other processes through registers, REG, which transform the data from dual-rail to boolean representation. The registers produce an acknowledge, ack, for the control part when the boolean variables are stable. To send a value on a channel, the control part activates the function block, f, through the signal go. The function block serves both to compute some function and to transform data from boolean to the dual-rail representation used for delay-insensitive communication. A function block implementing a full-adder is described in section 4.1.1.

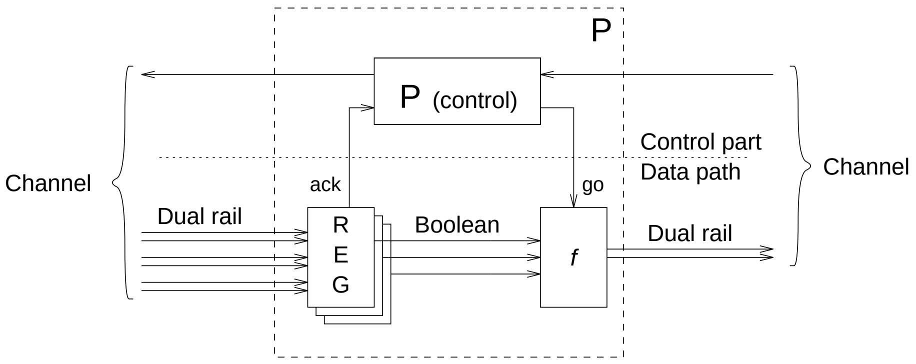

Figure 2. Decomposition of the process P into a control part and a data path.

Seen from the control part, any channel appears as two wires on which the handshake protocol is implemented. The inclusion of the data path in the channel will appear as an extra communication delay only.

Apart from ensuring correct realizations, much attention has been put into creating efficient circuits during the implementation steps. Analysis tools are provided to assist the designer in finding efficient solutions for each transformation [2]. At the circuit level, tools have been developed for cell generation, placement, and routing, using MAGIC [14].

3.2. The multi-ring design

The multi-ring design method uses a set of simple building blocks yielding designs which are delay-insensitive by construction [10, 11]. A multi-ring consist of interacting pipelines and rings. It is a data driven computing structure using a four-phase handshake protocol for all communication. There is no separation between the control part and data path of the circuit. The basic structure is a pipeline, see figure 3. In a pipeline with at least three latches it is possible to connect the input and output and form a ring that can perform iterative computations[3].

The building blocks used to implement the pipelines and rings are: latches and function blocks (combinational circuits). To compose pipelines and rings into multi-ring structures, join and fork elements are used to synchronize data streams, and switches are used to implement conditional data transfers. A switch is a function block that will either cross or pass two data signals – determined by a control signal. For the vector multiplier design

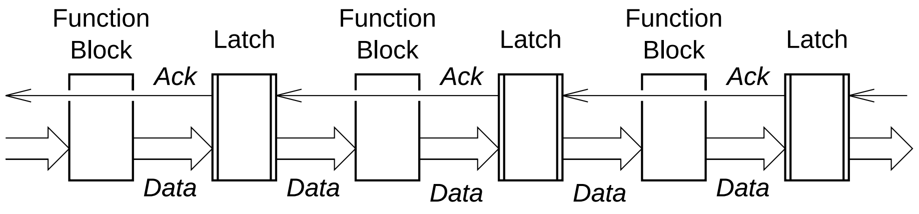

Figure 3. Delay-insensitive pipeline.

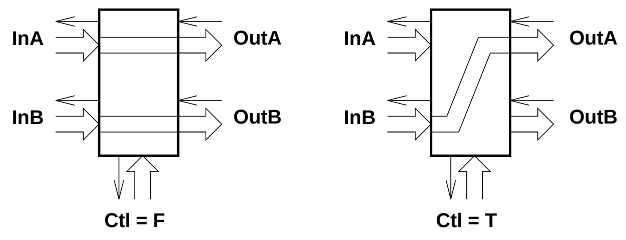

Figure 4. Asymmetric switch.

discussed in this paper, we need an asymmetric switch where either both data signals are passed through or only one of them is crossed over and the other waits, see figure 4.

The building blocks can be synthesized in different ways. We have used a particularly simple technique called DIMS (Delay-Insensitive Minterm Synthesis) that allows the components to be synthesized from C-elements, NOR-gates and inverters [10]. The DIMS technique is briefly described in section 4.1.2.

The physical realization of the design is a standard cell layout. Currently, we use the AutoCells tool that is part of the GDT design system from Mentor Graphics Inc. As C-elements are used extensively in the design, we have developed a set of C-element standard cells for the GDT design system.

3.3. The micropipeline design

The structure of the micropipeline design is similar to the multi-ring realization. However, the design is structured in a control part and a data path. The control part consists of C-elements, matched delays, and possibly some logic using event signaling. The data path is made of ordinary combinational circuitry and event driven registers of the type presented by Sutherland in [12, figure 12(b)]. This register is straightforward to implement using the standard cells available in the GDT design system.

When implementing a ring structure, there must be at least one empty storage element in order for the ring to be able to iterate. Because of the two-phase bundled-data protocol used in the micropipeline technique, each data value will only occupy one storage element – there are no empty data values. This allows a micropipeline ring to be implemented with only two storage elements.

The overall structure of the MAS block is shown in figure 5. Each of the two registers consists of a section for the accumulator (sum and carry) and a section for the parallel operand. Similarly, the switch consisting of a data- and a control part, can be sub-divided into a section controlling the flow of the parallel operand (shift or load a new operand) and a section controlling the accumulator (multiply and accumulate or output the result). The environment supplies the serial operand bit and the control signals for the switches, but this is not shown in figure 5. The detailed design of the switch is described in section 4.

3.4. The synchronous design

The synchronous design is straightforward. A two-phase non-overlapping clocking scheme is used, and the components used to implement a MAS bit-slice are: seven static latches, two multiplexors, an AND-gate, and a full-adder (figure 6).

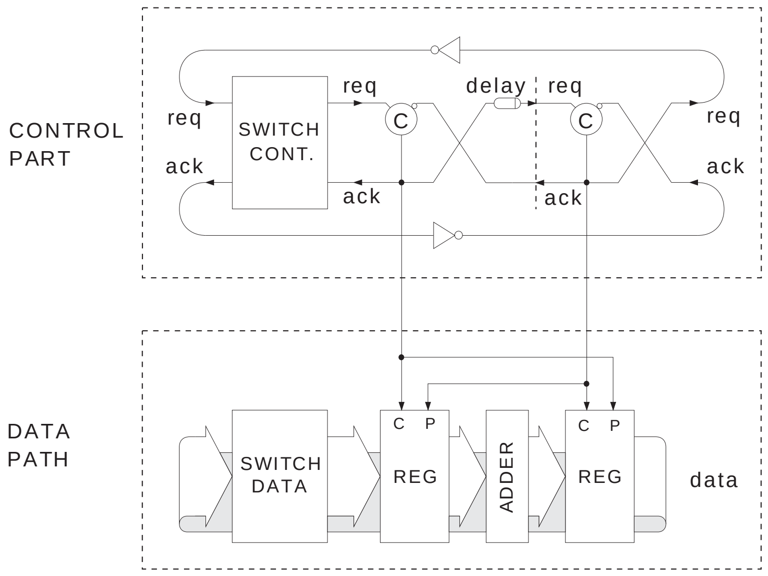

Figure 5. The overall structure of the micropipeline realization of the vector multiplier.

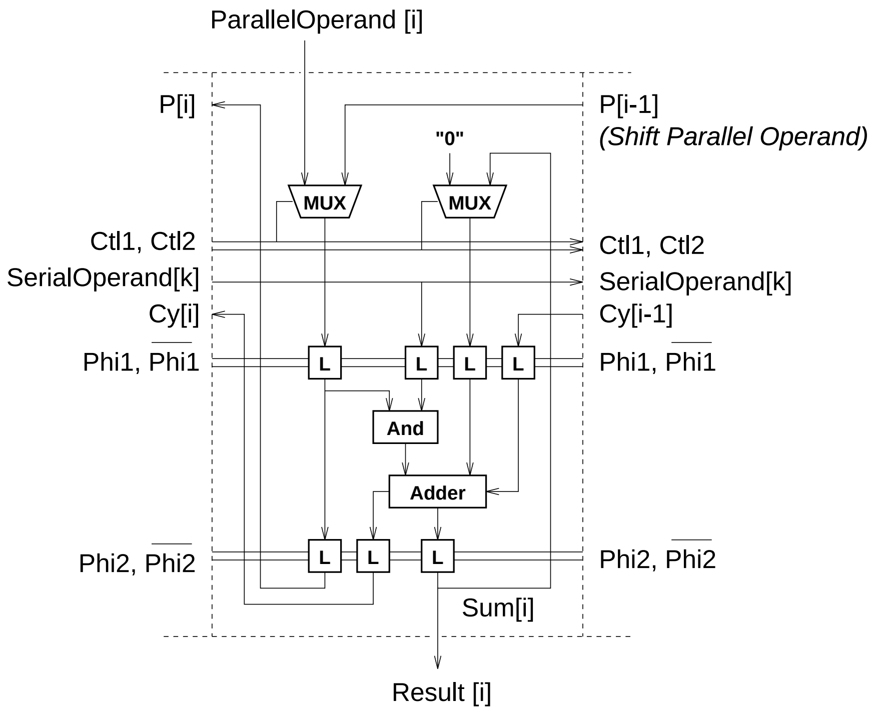

Figure 6. Synchronous realization of the ith MAS bit-slice.

4. CIRCUIT LEVEL ISSUES

In section 3, the four different design methods were described. From a macroscopic point of view, these methods yield designs which are structurally very different. For example, the micropipeline approach involves a separation of data and control, whereas these are merged in a multi-ring design. On the microscopic level there are also some significant differences between the resulting circuits. To illustrate these differences, section 4.1 presents the transistor level design of the full-adders used in the four designs, and section 4.2 presents some design details of micropipeline circuitry.

4.1. Adders

All four designs use a carry-save representation of intermediate results. A traditional full-adder is used to generate the pair: sum, cy, from the inputs: a,b, and c ($\oplus$ denotes exclusive or).

$$\begin{aligned} sum &= a \oplus b \oplus c \\ cy &= ab + cb + ac \end{aligned}$$

In this section, the circuit realizations of these functions are described in some detail. The two delay-insensitive designs use the dual-rail code where two wires, x.t and x.f, are used to represent a single bit x. The valid values true (T) and false (F) are represented by x.t high and x.f low, and by x.f high and x.t low respectively. The empty value (E) is represented by both wires low.

4.1.1. Caltech

The function blocks in the Caltech design are implemented using structures similar to dynamic CMOS logic. The logic functions for the dual-rail output are computed in an n-transistor network. The activation of the function block is controlled through the signal go. The control part must ensure that the inputs are stable while go is operated.

The behavior of the full-adder function blocks producing the four dual-rail signals: sum.t, sum.f, cy.t, and cy.f is described by the production rules:

$$\begin{aligned} &(a\,\overline{b}\,\overline{c} + \overline{a}\,b\,\overline{c} + \overline{a}\,\overline{b}\,c + a\,b\,c\,)\,go &&\rightarrow \overline{sum.t}\downarrow \\ &go + \overline{sum.f} &&\rightarrow \overline{sum.t}\uparrow \\ &(\overline{a}\,b\,c + a\,\overline{b}\,c + a\,b\,\overline{c} + \overline{a}\,\overline{b}\,\overline{c}\,)\,go &&\rightarrow \overline{sum.f}\downarrow \\ &go + \overline{sum.t} &&\rightarrow \overline{sum.f}\uparrow \end{aligned}$$

$$\begin{aligned} &(\,a\,b + (a+b)\,c\,)\,go &&\rightarrow \overline{cy.t}\downarrow \\ &go + \overline{cy.f} &&\rightarrow \overline{cy.t}\uparrow \\ &(\,\overline{a}\,\overline{b} + (\overline{a}+\overline{b})\,\overline{c}\,)\,go &&\rightarrow \overline{cy.f}\downarrow \\ &go + \overline{cy.t} &&\rightarrow \overline{cy.f}\uparrow \end{aligned}$$

The registers storing the input variables contain both the true and the inverted values. The pairwise cross-coupling between the pull-up parts enables the circuit to maintain the output value. The transistor level realization of these production rules is shown in figure 7. The transistor count (including inverters on the outputs) is 40.

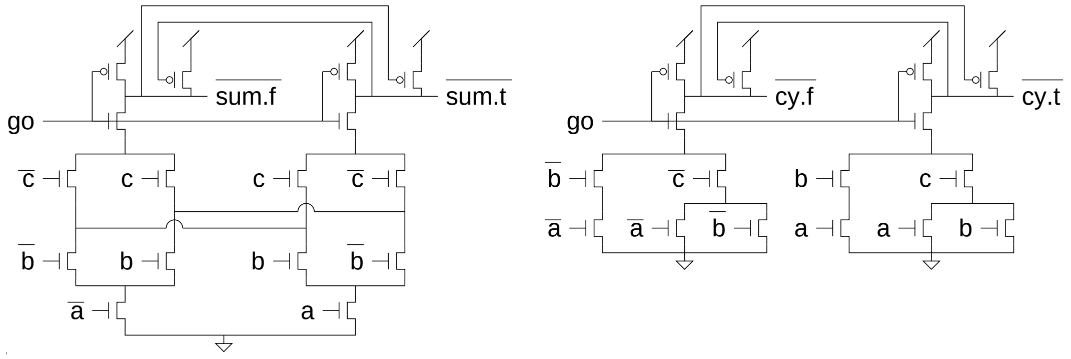

Figure 7. Transistor diagram for a full-adder in the Caltech realization

The detailed design of a full-adder is discussed in [6]. In that design the activation of the function block is not controlled by an explicit control signal, but by the arrival of the input signals themselves.

4.1.2. DIMS

The components used in the multi-ring design are synthesized using a technique called delay-insensitive minterm synthesis (DIMS). This technique resembles the traditional sum of products approach with a few important differences: (1) the minterms are formed using C-elements instead of AND-gates, and (2) reduction of the boolean equations by combining minterms into simpler terms is (in general) not allowed.

Together, these requirements assure that the function blocks do not produce any valid output signals until all input signals are valid, and that none of the output signals change back to the empty value until all inputs become empty. A similar technique has been used by others [9].

The DIMS technique does not in general allow reduction of boolean equations. If, however, multiple logic functions depend on the same input, they can share the C-elements and thus achieve a reasonably efficient circuit realization. The full-adder illustrates this: both the sum and the carry depend on the two input operands plus the incoming carry. As illustrated in figure 8, the adder can be implemented using 8 C-elements and 4 OR-gates.

The connections on the input side of the C-elements are the same no matter what function is computed (because the array of C-elements corresponds to all possible minterms), whereas the connections on the output side are specific for the particular function to be implemented. The columns in the truth table indicate which C-element outputs (i.e. which minterms) that are connected to the different OR-gates.

The C-element is realized as a standard cell, so the designer cannot change the internal details of a C-element. For completeness the transistor realization of a three input C-element is shown in figure 9. Using this C-element realization, the transistor count for the full-adder is 120.

a	b	c	sum.f	sum.t	cy.f	cy.t
E	E	E	0	0	0	0
Mixed E and T/F			No change			
F	F	F	1	0	1	0
F	F	T	0	1	1	0
F	T	F	0	1	1	0
F	T	T	1	0	0	1
T	F	F	0	1	1	0
T	F	T	1	0	0	1
T	T	F	1	0	0	1
T	T	T	0	1	0	1

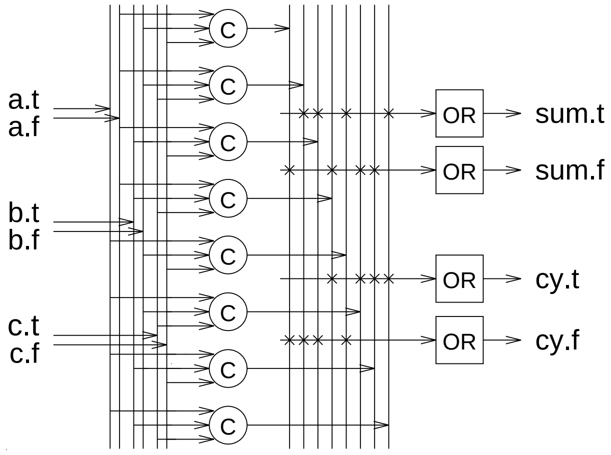

Figure 8. DIMS full-adder.

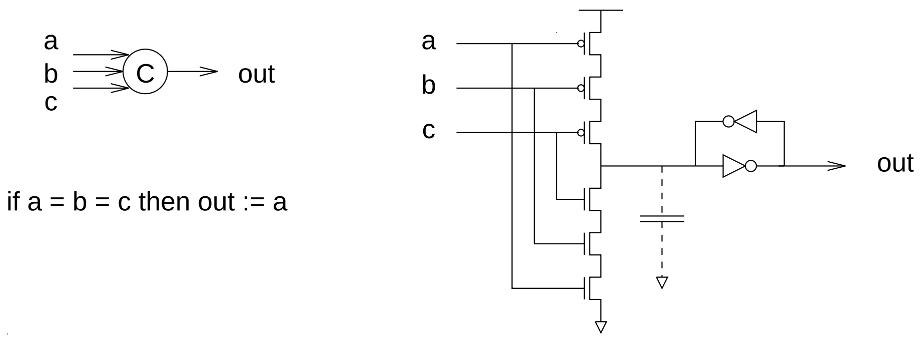

Figure 9. A three input C-element and its realization in CMOS.

4.1.3. Micropipeline and synchronous designs

The micropipeline design and the synchronous designs use the same full-adder. This adder is a direct implementation of the following equations using two "and-or-invert gates" (standard cells):

$$\begin{aligned} sum &= abc + a\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} \\ cy &= ab + cb + ac \end{aligned}$$

The transistor count of this adder including inverters for the input signals is 46. It is possible to design a synchronous full-adder using only 24 transistors [13, figure 8.4], however, it is not possible to utilize this design in the standard cell based design system used for the experiments reported here.

4.2. Circuits for two-phase signaling

The micropipeline design technique is based on a two-phase bundled-data protocol (with event signaling on the request and acknowledge wires). It is our experience that components for this protocol are more complex and difficult to design than similar components using a four-phase handshake protocol. As illustrated below, this is the case in both the data path and the control part.

Event controlled storage elements

The micropipeline design technique use an event controlled storage element whose function is equivalent to that of an ordinary static latch: it can hold a data value or it can be transparent. The storage element is controlled by two alternating event signals called capture and pass. In [12] Sutherland presents three different realizations. The first of these requires approximately 50 % more circuitry than an ordinary static latch, and the second design (which is slightly faster) requires more than 100 % extra circuitry. The third design uses ordinary static latches and a control circuit consisting of a merge and a toggle element. Because of the complexity of the control circuit, this realization is only relevant when the number of bits in the storage element is large (more than 10 bits), and it is also much slower than the two other designs. In summary, all three realizations of the event controlled storage element are 50–100 % larger than the corresponding static latch. The vector multiplier design uses the second design because of its straightforward implementation using the GDT standard cell library.

The switch

The micropipeline design makes use of a switch with a functionality similar to the one shown in figure 4. Figure 10 shows a block diagram of a realization for a two-phase bundled-data protocol, and the circuit details of the request and acknowledge control circuits are shown in figure 11. A key component in this circuitry is the select element [12], and we have used a realization presented in [1].

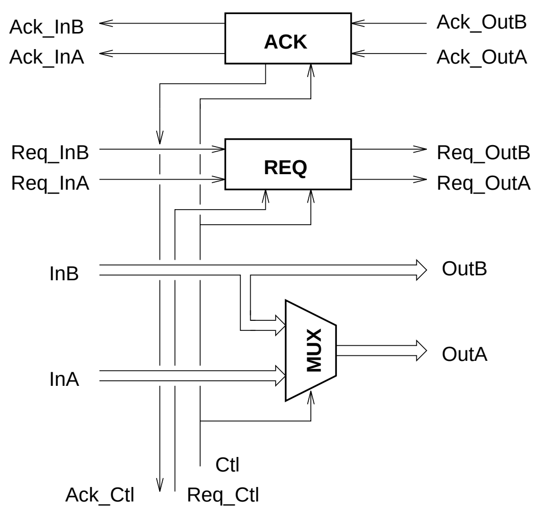

Figure 10. Block diagram of the micropipeline switch.

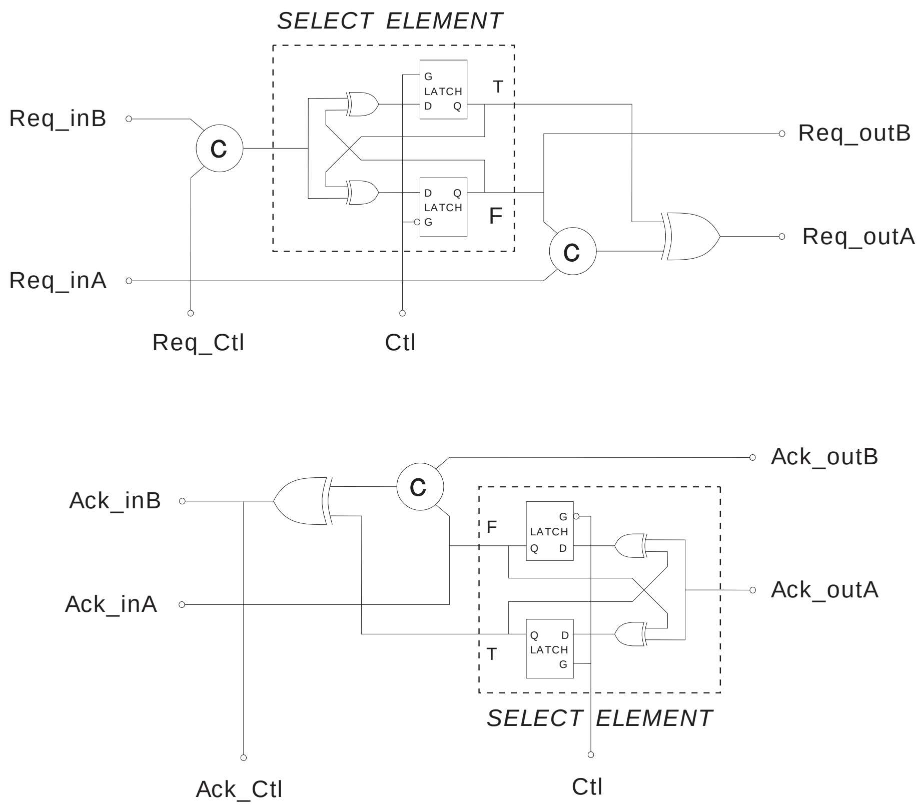

Figure 11. Realization of the request and acknowledge control circuits in the switch.

There is no relationship between the polarity of events on the request and acknowledge signals on the input and the output ports of the switch. Therefore, the signal transitions on the request and acknowledge wires on the output ports can not be derived from the transitions on the corresponding inputs using combinational circuits. Some internal memory is required to store the signal level on the outputs, and this memory is part of the select element.

5. DISCUSSION

The comparison of the four designs is divided into two parts: a quantitative and a qualitative comparison. The quantitative comparison presents our measurements of key parameters of the four designs. For each design we have tried to give an unbiased explanation for the reported measurements. Based on the measurements, we have found many possible optimizations which could change the parameters significantly, however, in this paper we have avoided the temptation of guessing the possible improvements suggested by the measurements. In the qualitative comparison, we try to report our experiences gained by making the four designs. These are of course somewhat subjective.

Table 2
Key figures for the different MAS bit-slice implementations.

Design	Transistor count	Layout area	Cycle time	Energy per cycle	Speed-Power product
		mm^2	ns	pJ / cycle	nJ
Caltech	347	0.37	27	–	–
Caltech (scaled down)	347	0.28	18	365	6.6
Micropipeline	395	0.26	24	189	4.5
Multi-ring (DIMS-circuits)	824	0.40	22	290	6.4
Synchronous	188	0.10	12	88	1.1

5.1. Quantitative comparison

The key parameters of the four designs are listed in table 2. As mentioned in section 2, the comparison of the designs is based on a MAS bit-slice, because all four designs use a bit-slice with exactly the same functionality. In this way, differences concerning word size and control logic are eliminated.

The Caltech design is implemented in a 2 μm CMOS technology using MOSIS design rules with dense contacts, and the first row of the table lists measurements on the fabricated chip. The other designs are implemented in a 1.5 μm CMOS technology via EUROCHIP. To compensate for this difference in fabrication technology, we have estimated the corresponding parameters for a Caltech layout scaled down to this process. These estimated figures are listed in the second row of the table.

Rows three, four, and five of the table list the parameters of the other MAS bit-slice realizations. The multi-ring design has been fabricated and yielded a working chip [10]. The micropipeline and the synchronous designs have not been fabricated, but the designs have been completed down to the layout level. For all four designs, the numbers for cycle time and power consumption are obtained from HSPICE simulations of the layouts using typical process parameters for the 1.5 μm EUROCHIP process and typical operating conditions ($V_{DD} = 5.0V$ and $T_A = 27^oC$).

The estimate of the down-scaled Caltech design has been done by extracting a HSPICE netlist from the original 2 μm layout using the electrical process parameters for the EUROCHIP process. This netlist consists of ideal transistors (with no diffusion capacitances), diodes modeling the parasitic capacitance of diffusion areas, and discrete capacitors modeling the parasitic capacitance of wires. In this netlist, all transistors have been scaled preserving the w/l ratio, and the capacitances have been scaled according to the design rules. This gives fairly accurate estimates of the speed and power. The area (of the down-scaled Caltech design) is a coarser estimate, because no actual layout exists.

Transistor count

It is commonly being claimed that the complexity of a delay-insensitive circuit (using the dual-rail code) is two times that of a corresponding synchronous circuit, and this is supported by our results. The Caltech design which is the smallest of the two delay-insensitive circuits contains approximately twice as many transistors as the synchronous circuit.

Maybe it is surprising that the micropipeline design is not the smallest of the three self-timed designs. This is because the two-phase signaling complicates the control part and the registers in the data path (as explained in section 4.2).

The high transistor count of the multi-ring design is due to the DIMS circuit technique rather than the multi-ring concept. Building blocks implemented directly as transistor networks (similar to the Caltech approach) could reduce the transistor count significantly as indicated by the adder designs described in section 4.1.

Cycle time

The Caltech and the multi-ring designs are delay-insensitive, and the circuits will operate at the maximal speed allowed by data and operating conditions. No safety margin or worst-case considerations are needed to guarantee safe operation. The micropipeline design relies on delay matching including some safety margin. In the synchronous circuit a similar safety margin is necessary when determining the cycle time. For a fair comparison with the self-timed designs this safety margin has to be taken into account, and the cycle time listed in the table includes a 50 % safety margin. The actual delay of the critical path at typical conditions is 8 ns.

Energy and speed-power product.

The Caltech design method leads to circuit realizations consisting of (complex) transistor networks, and the layout tools make extensive use of transistor sizing in order to improve performance [2] (the layout of the vector multiplier contains transistors with channel widths up to 100 μm). This sophistication is in contrast to the other designs that are implemented using a simple standard cell library with fixed-size transistors ($w_p = 12$ μm and $w_n = 4$ μm). This explains the higher speed and the higher power consumption of the Caltech design.

The speed-power product is a complexity measure that enables comparisons across such differences. It is interesting that the speed-power products of the Caltech and the multi-ring design are the same. This indicates that there are no fundamental differences between the speed that can be obtained using the two methods. The multi-ring design contains more than twice as many transistors, but the switching activity is the same.

A comparison with the synchronous and the micropipeline designs is more difficult, because their speed and speed-power products depend on a safety margin that is the subjective choice of the designer. However, the speed-power product of the synchronous design is less than one fifth of the delay-insensitive designs. This is a significant difference.

The power consumption of the micropipeline design is the power consumption in a bit-slice of the data-path plus one tenth of the control part.

5.2. Qualitative comparison

In addition to the quantitative figures and explanations presented above, it is also relevant to report some qualitative experiences and observations.

Micropipeline design

The event based control circuit in the switch (figure 10) made it rather difficult to design, and its complexity degrades the performance of the vector multiplier significantly. Except for the different storage elements the data path of the micropipeline and the synchronous designs are identical. The poor speed of the micropipeline design is due to the complexity of the control part.

In addition to this, some manual intervention (floor planning) was needed in the layout phase in order to preserve the ordering of signal events. This means that an automatic layout is less feasible than for the delay-insensitive designs.

Furthermore, our experience from this design experiment indicates that the two-phase bundled-data protocol has its potential in pipelined structures with a simple (unidirectional) data flow, whereas a four-phase bundled-data protocol may lead to smaller and faster circuits when the data flow is less regular and involves conditional sequencing.

Caltech design

The Caltech design method is by far the most refined/sophisticated of the self-timed methods we have worked with. This sophistication and the set of interactive CAD tools supporting it requires a skilled designer in order to do the sequence of transformations and optimizations (handshake re-shuffling, state assignment, guard strengthening, bubble re-shuffling, transistor sizing etc) yielding the efficient realization which is the objective of the Caltech design method.

Multi-ring design with DIMS circuits

This is a very simple and straightforward approach, and the layout can be produced using existing automatic standard cell place and route tools. However, this simplicity has its price: (1) the DIMS circuit technique is inefficient in terms of transistor count, and (2) the multi-ring concept is restrictive, and in some cases it may lead to inefficient realizations (although we did not experience it in this design experiment). On the other hand, the simple structural concept makes it fairly simple to verify delay-insensitivity.

6. CONCLUSION

In this paper we have presented four different designs of a vector multiplier: three different asynchronous and a synchronous circuit. The area, power consumption, cycle time and speed-power products of the four designs have been compared showing some interesting differences between the four designs. However, this comparison is based on a single design problem, and there are interesting aspects not covered by our comparison, for example scalability and testability. Hopefully, others will publish similar comparisons of other designs problems and measuring other quantities. We think a range of such comparisons are needed to stimulate research in self-timed design, and to provide some substance to the many claims made about the properties of self-timed circuits.

REFERENCES

1. Erik Brunvand. A cell set for self-timed design using Actel FPGAs. Technical report, VLSI Systems Research Group, Department of Computer Science, University of Utah, 1991. UUCS-91-031.
2. Steven M. Burns. *Performance Analysis and Optimization of Asynchronous Circuits.* PhD thesis, Computer Science Department, California Institute of Technology, 1991. Caltech-CS-TR-91-01.

3. Mark R. Greenstreet, Jørgen Staunstrup, and Ted E. Williams. Self-timed iteration. In Carlo H. Séquin, editor, *Proceedings of VLSI '87*, pages 269–282. IFIP, August 1987.
4. S.Y. Kung and J.N. Hwang. Parallel algorithms/architectures for neural networks. *Journal of VLSI Signal Processing*, 1:221–251, 1989.
5. Alain J. Martin. Compiling communicating processes into delay-insensitive VLSI circuits. *Distributed Computing*, 1(4):226–234, 1986.
6. Alain J. Martin. Asynchronous datapaths and the design of an asynchronous adder. *Formal Methods in Systems Design*, 1:117–137, 1992.
7. Christian D. Nielsen and Alain J. Martin. Design of a delay-insensitive multiply-accumulate unit. In *Proceedings from HICSS-26*, pages 379–388. IEEE Computer Society Press, 1993.
8. Lars Skovby Nielsen. Design and evaluation of micropipelined circuits. Master's thesis, Department of Computer Science, Technical University of Denmark, 1992. In Danish.
9. N.P. Singh. A design methodology for self-timed systems. Master's thesis, Laboratory for Computer Science, MIT, 1981. MIT/LCS/TR-258.
10. Jens Sparsø, Jørgen Staunstrup, and Michael Dantzer-Sørensen. Design of delay insensitive circuits using multi-ring structures. In Gerry Musgrave, editor, *Proceedings of EURO-DAC '92, European Design Automation Conference*, pages 15–20. IEEE Computer Society Press, September 1992.
11. Jens Sparsø and Jørgen Staunstrup. Design and performance analysis of delay insensitive multi-ring structures. In *Proceedings from HICSS-26*, pages 349–358. IEEE Computer Society Press, 1993.
12. Ivan E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, June 1989.
13. N. Weste and K. Esraghian. *Principles of CMOS VLSI Design – A Systems Perspective.* Addison-Wesley, Reading, 1985.
14. W. S. Scott, R. N. Mayo, G. Hamachi and J. K. Ousterhout, editors, *1986 VLSI Tools: Still More Works By the Original Artists,* Computer Science Division (EECS), University of California, Berkeley, Report No. UCB/CSD 86/272, December 1985.