Design of delay insensitive circuits using multi-ring structures

Jens Sparsø Jørgen Staunstrup Michael Dantzer-Sørensen

Department of Computer Science, Building 344,
Technical University of Denmark, DK-2800 Lyngby, Denmark.

E-mail: jsp@id.dth.dk, jst@id.dth.dk

Abstract

This paper describes the design and VLSI implementation of a delay insensitive circuit that computes the inner product of two vectors. The circuit is based on an iterative serial-parallel multiplication algorithm. A test chip has been fabricated via EUROCHIP.

The circuit is the result of a design experiment that we have conducted as part of our ongoing work to construct CAD tools supporting our design approach.

The design is based on a data flow approach using pipelines and rings that are composed into larger multi-ring structures by joining and forking of signals. The implementation is based on a small set of building blocks (latches, combinational circuits and switches) that are composed of C-elements and simple gates.

By following this approach we have not found it difficult to design delay insensitive circuits with nontrivial functionality and reasonable performance.

1 Introduction

Design of delay insensitive circuits is an active area of research [1, 2, 3, 4]. Because design of delay insensitive circuits is different from design of synchronous systems, and because delay insensitive circuits are difficult to verify by simulation, most research is devoted to the development of formal design methods. A number of specific circuits designed by research groups have been published, but only a limited number of these have been fabricated and characterized in terms of silicon area (in mm^2), speed (in nano seconds) and power (in watts).

The work reported in this paper represents a nontrivial delay insensitive circuit design experiment that we have conducted in order to gain practical design experience and to stimulate our work on design methods.

The goal is to design a circuit that computes the inner product of two vectors. Inputs to the circuit are two streams of integers along with a tag indicating the last pair of operands, and output from the circuit is the accumulated sum of products. The circuit has been optimized for use in an environment where many of the operands are small numbers whose binary representations contain many leading zeros. The circuit will skip the corresponding multiplication steps. The intended application for this circuit is a delay insensitive implementation of a digital artificial neural network based on a matrix-vector multiplication architecture.

A test chip has been fabricated via EUROCHIP. This chip multiplies 4 bit vector elements and it produces a 10-bit result.

The paper is organized in two parts as follows: Part I, consisting of sections 2 and 3, describes the design technique, and Part II, consisting of sections 4 to 7, describes the design experiment.

Part I : Design principles

2 Delay insensitive multi-rings

Delay insensitive circuits are asynchronous and the sequencing of their computations is determined by the data flow rather than by clock signals or other global control signals. When inputs to a sub-circuit are ready, the computation can start and as soon as the result is computed, the next computation can be initiated. In this section we describe a class of circuits, called *delay insensitive multi-rings*, using such a data driven approach [5].

2.1 Data representation

In a delay insensitive circuit there is no clock signal to determine when a computation can start and when it is complete. Instead, it must be possible to detect the arrival of a new input from the data themselves. To be able to do this, a protocol is used where *empty* (or spacer) values are inserted between proper values. The empty value is denoted E and all values representing proper data ready for computation are called *valid* (denoted V). To distinguish between empty and valid values, data is encoded. It is essential that the encoding of valid values ensures that no partial result is interpreted as a valid value, otherwise a computation can start prematurely. There are many ways of encoding data for delay insensitive computations [7]; in section 3.1 we describe a particular and very commonly used encoding called *dual rail*. Alternating empty and valid values are used in a four cycle handshaking protocol.

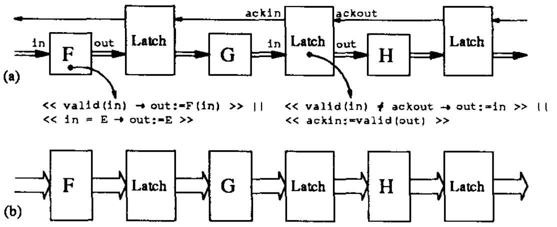
Consider a simple computation consisting of the composition of three functions F, G and H. When a valid input is given to F, it can start its computation, and as soon as the result from F is ready, G can begin, immediately followed by H. When all three functions

0-8186-2780-8/92 $3.00 © 1992 IEEE

have completed their computations, the valid values must be flushed out, before a new computation can be initiated. This is done by giving an empty value on the input of F. This empty value must now ripple through, and when the output of H has changed to empty, the next computation can start. Hence, each sub-circuit corresponding to F, G, or H must be capable of propagating empty values in addition to computing their respective functions. Such an element is called a (delay insensitive) *functional block.*

2.2 Pipelines

The composition of F, G and H described above can be realized as a pipeline, where latches are placed between the functional blocks. This is illustrated in figure 1(a) together with a behavioural description of a latch and a functional block (using the language *Synchronized Transitions* [4]).



```
<< valid(in) → out:=F(in) >> ||
<< in = E → out:=E >>

<< valid(in) ≠ ackout → out:=in >> ||
<< ackin:=valid(out) >>
```

Figure 1: Delay insensitive pipeline.

A delay insensitive latch holds back input data until the successor circuits are ready to receive them. The latch is controlled by acknowledge signals from succeeding latches. A latch may load and hold a valid value when its successor latch in the pipeline holds the empty value (indicated by ackout being false). Similarly a latch may load and hold an empty value when its successor latch in the pipeline holds a valid value (indicated by ackout being true).

We often simplify the schematics by representing data signals and their associated acknowledge signal (going backwards), by a single bus-symbol as shown in figure 1(b). This descriptional simplification is used throughout the rest of the paper. It must be kept in mind that the functional blocks in general just pass the acknowledge signals backwards.

2.3 Delay insensitive rings

In this section we describe a number of generalizations of the latched pipeline leading to a general characterization of the delay insensitive circuit structures used in this work.

In a latched pipeline with at least three latches, it is possible to connect the output of the last stage to the input of the first, forming a *delay insensitive ring*. Such a ring is capable of performing an iterative computation [8]. In [9] it is described how such a ring is used to perform floating point division. The data in the ring represents a partial remainder, and each stage computes one bit of the final result and forms a new partial remainder which is sent to the next stage etc. Consider a delay insensitive ring with three latches. These latches will always contain one of two patterns: either two valid and one empty element or two empty and one valid. A sequence of computations is shown in figure 2.

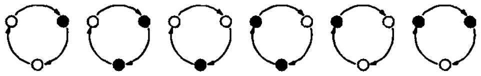


Figure 2: Sequence of computations in a delay insensitive ring.

It is quite simple to build rings with more than three latches and as many pipeline stages as one wishes. The choice is mainly governed by performance considerations. In a large ring with many latches and pipeline stages many computations can be overlapped yielding a high throughput.

Independent rings and pipelines can be connected in different ways using fork and/or join elements. In a simple join element both inputs are propagated when either they are both empty or they are both valid. There are many possible variations and combinations of join and fork elements. The switch described in section 3.2 is an example such an element: It has two data inputs and two data outputs and a control signal selects how inputs and outputs are connected.

Combining the various building blocks such as pipeline stages, latches, fork elements, join elements and switches, it is possible to construct a class of delay insensitive circuits consisting of interacting rings and pipelines called *multi-ring* structures.

3 Realization of building blocks

In this section we describe the realization of the building blocks (functional blocks, latches and switches) that are needed for constructing the delay insensitive multi-ring structures introduced in section 2.

3.1 Functional blocks

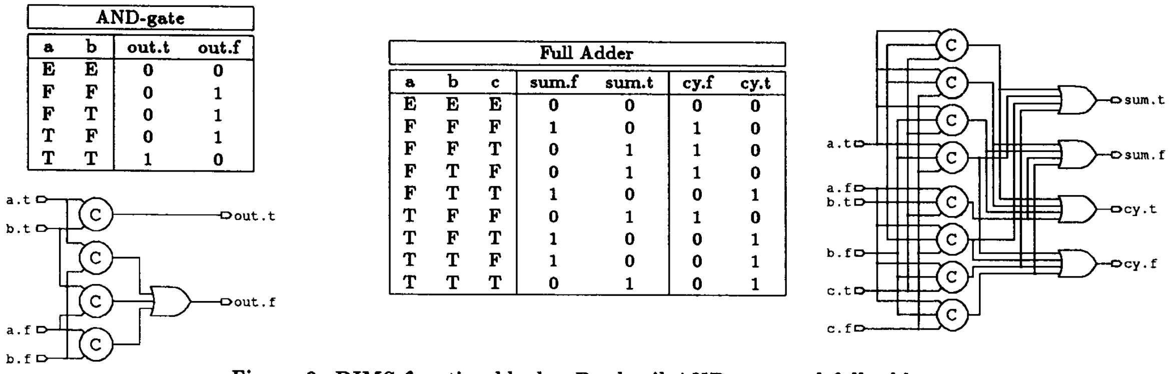
A functional block computes a specific (combinational) function, F, and it can propagate empty values as required by the delay insensitive protocol.

To synthesize a delay insensitive circuit for a functional block we use a technique called *delay insensitive min-term synthesis (DIMS)*. This technique resembles the traditional sum of products approach, but there are also a few important differences:

- the min-terms are formed using C-elements (instead of AND-gates),
- reduction of the boolean equations by combining min-terms into simpler terms is (in general) not allowed.

Together, these requirements assure that our combinational circuits 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 are empty. A similar technique has been used by others [6]

Data is represented in a dual rail code where two wires, x.t, x.f, are used to represent a single bit x. The value empty (E) is represented with the signals on both wires low, true (T) is represented by x.t high and x.f low, and false (F) by x.f high and x.t low.



AND-gate			
a	b	out.t	out.f
E	E	0	0
F	F	0	1
F	T	0	1
T	F	0	1
T	T	1	0

Full Adder						
a	b	c	sum.f	sum.t	cy.f	cy.t
E	E	E	0	0	0	0
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 3: DIMS function blocks: Dual rail AND-gate and full adder.

The DIMS technique can be illustrated with the circuit for a delay insensitive dual rail AND-gate, see figure 3. As can be seen from the truth table, the sum of product representation of **out.f** is:

$$\mathtt{out.f} = \mathtt{a.f}\cdot\mathtt{b.f} + \mathtt{a.f}\cdot\mathtt{b.t} + \mathtt{a.t}\cdot\mathtt{b.f}$$

The products are realized by C-elements as shown in figure 3.

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 effective circuit. As an example, we mention a full adder where both the sum and the carry depend on the two input operands and the incoming carry. The full adder can thus be built using 8 C-elements and 4 OR-gates as shown in figure 3.

The DIMS technique has been automated in a tool which can synthesize delay insensitive circuits from high-level descriptions [10].

3.2 Switches

Switches are used as data flow control elements. In the general case two data signals are either crossed or just passed through, determined by a control signal. For the design 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.

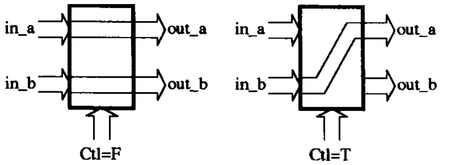


Figure 4: Asymmetric switch.

The switch consist of three sets of combinational circuitry: one for generating the data on the output ports, one for generating the acknowledge signals on the input ports and one for generating the acknowledge signal on the control port. It should be noted that the control input also follows the four cycle protocol alternating between empty and valid values.

The asymmetric switch makes it possible to have rings with different data rates, e.g. one which supplies one value every time the other supplies ten.

3.3 Latches

A latch for a single dual rail encoded bit is built from two C-elements an OR-gate and an inverter, see figure 5. The OR-gate generates the acknowledge signal, indicating whether the latch holds a valid or empty value. The corresponding acknowledge from the succeeding register, ackout, determines whether the register should hold its current value or load a new.

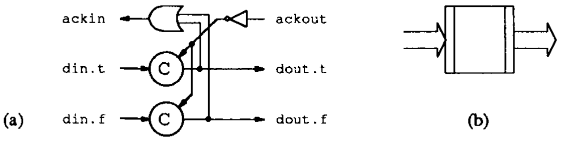


Figure 5: Latch, (a) implementation and (b) symbol.

Two degenerated forms of the latch are also needed, one for consuming values (and generating acknowledgements) and one for producing constant values (and consuming acknowledgements). These are quite simple variations of the fundamental latch shown in figure 5. They are for example used to "terminate" unused inputs and outputs at the boundary of bit slice structures.

3.4 Fork and Join elements

The fork and the join elements shown in figure 6 complete the set of building blocks. Forks are used when the same signal is input to more circuits and joins are used when signals from more sources are input to a single circuit.

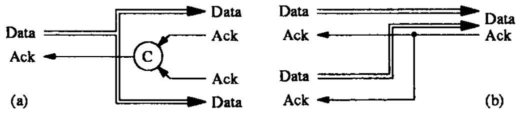


Figure 6: (a) Fork element. (b) Join element.

3.5 Initialization

The initialization of a delay insensitive circuit plays a major role. Because the circuit is data driven, it is important to insert valid data values in strategic places to make the circuit run. In a synchronous circuit it is very often the case that only a subset of the registers are reset. The remaining registers will then assume well defined values during the first clock cycles of normal operation. This is not a feasible scheme in delay insensitive circuits because all latches depend, not only on their input, but also on the output of the succeeding state holding element (via the acknowledge signal). This bidirectional flow of information (data forward and acknowledge backward) makes it necessary to explicitly initialize all C-elements including those used in combinational logic.

Part II : The design experiment

4 Algorithm

An iterative serial-parallel multiplication algorithm is used for our design experiment. Its implementation requires some interesting and nontrivial circuit structures, and it represents a good compromise between area and speed. As our main interest is design of delay insensitive circuits we restrict our considerations to multiplication of positive integers in order to simplify the design. It should however be noted that the circuit structure which we develop can be modified to support multiplication of signed numbers using for example the algorithm described in [11]. The algorithm is based on the paper and pencil approach shown in figure 7.

$y_0 \cdot 0$	$y_0 \cdot 0$	$y_0 \cdot 0$	$y_0 \cdot 0$	$y_0 x_3$	$y_0 x_2$	$y_0 x_1$	$y_0 x_0$
$y_1 \cdot 0$	$y_1 \cdot 0$	$y_1 \cdot 0$	$y_1 x_3$	$y_1 x_2$	$y_1 x_1$	$y_1 x_0$	$y_1 \cdot 0$
$y_2 \cdot 0$	$y_2 \cdot 0$	$y_2 x_3$	$y_2 x_2$	$y_2 x_1$	$y_2 x_0$	$y_2 \cdot 0$	$y_2 \cdot 0$
$y_3 \cdot 0$	$y_3 x_3$	$y_3 x_2$	$y_3 x_1$	$y_3 x_0$	$y_3 \cdot 0$	$y_3 \cdot 0$	$y_3 \cdot 0$
$w_0 \cdot 0$	$w_0 \cdot 0$	$w_0 \cdot 0$	$w_0 \cdot 0$	$w_0 z_3$	$w_0 z_2$	$w_0 z_1$	$w_0 z_0$
$w_1 \cdot 0$	$w_1 \cdot 0$	$w_1 \cdot 0$	$w_1 z_3$	$w_1 z_2$	$w_1 z_1$	$w_1 z_0$	$w_1 \cdot 0$
$w_2 \cdot 0$	$w_2 \cdot 0$	$w_2 z_3$	$w_2 z_2$	$w_2 z_1$	$w_2 z_0$	$w_2 \cdot 0$	$w_2 \cdot 0$
$w_3 \cdot 0$	$w_3 z_3$	$w_3 z_2$	$w_3 z_1$	$w_3 z_0$	$w_3 \cdot 0$	$w_3 \cdot 0$	$w_3 \cdot 0$
p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0

Figure 7: Example of inner product calculation, $P = X \times Y + Z \times W$ 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 X, Y, Z and W are 4 bit unsigned integers.

It is well known that in a fully combinational "paper and pencil multiplier", only one row of circuitry is active at a time. Therefore, almost the same multiplication speed can be obtained by implementing only one row of circuitry which is used in an iterative fashion. This serialization along the vertical axis requires (1) an accumulator in which the result is gradually formed, (2) a shift register that converts one of the operands into serial representation (called the *serial operand* or simply the *S-operand*), and (3) a shift register shifting the other operand (extended with zeros at both ends) one place to the left in each iteration. This operand is called the *parallel operand* or simply the *P-operand.* The carry signals produced in one row are passed to the the row below in order to avoid ripple carry propagation in the rows. In an iterative implementation this corresponds to using carry save representation for the (temporary) results stored in the accumulator.

Conversion from carry save representation into binary representation is postponed until after the last two integers (of the vectors) have been multiplied. The conversion is done by extending the last S-operand with leading zeroes and making this multiplication take as many steps as there are bits in the final result. To facilitate this the environment supplies along with the P-operand a tag (called Last) that indicates the last pair of operands.

In the intended neural network application the connectivity matrix is usually sparse, and the nonzero elements are often small numbers meaning that they will have a significant number of leading zeros. If these two characteristics are taken into account the computation time can be reduced significantly. We therefore assume the existence of some external zero-detection circuitry that along with the serial operand supplies a bit pattern indicating leading zeros. This is called the *zero-pattern* or simply the *Z-pattern.*

5 Design

5.1 Overall structure

The structure of the inner product circuit is shown in figure 8. It consists of three blocks: (1) a parallel to serial converter called PISO (Parallel In Serial Out), that converts the serial operand and the zero-pattern into bit serial data streams, S and Z, (2) an iterative multiply and accumulate ring called RING, and (3) a small control block called CONTROL, that delivers control signals for the switches in the circuit.

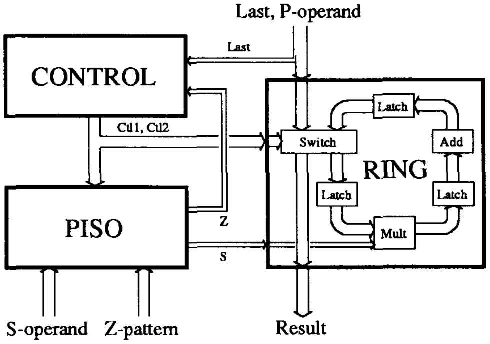


Figure 8: Structure of the inner product element.

The RING consists of three latches with combinational circuitry in between. Two variables are circulating in the RING: the temporary result in carry save representation and the parallel operand that is shifted one place to the left for each iteration cycle. The following subsections describe the three blocks in more detail.

5.2 The RING

The ring is built from identical bit slices, figure 9. In every iteration step each bit slice consumes the two bits to be multiplied (called Par and Ser), and the relevant sum and carry bits from the previous iteration step (Sum and Cy[i-1]). The serial operand bit, S, is broadcast (i.e. forked) to all bit slices and these copies are denoted Ser in the figure. Par and Ser are ANDed using a dual rail AND-gate (see section 3.1). This bit product is added to Cy[i-1] and Sum using a dual rail full adder (see section 3.1). The resulting carry is passed to the i+1th bit slice. The resulting sum bit stays inside the bit slice.

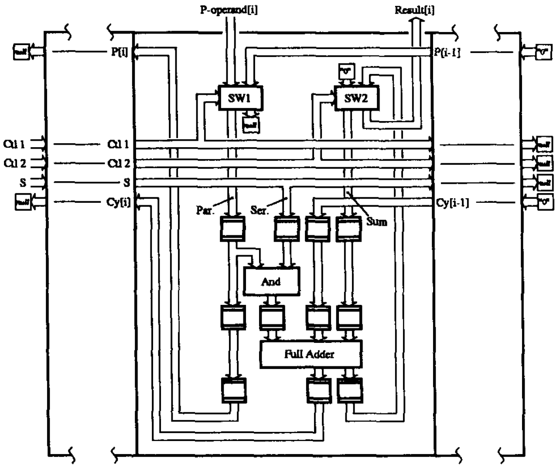


Figure 9: Circuit diagram of the ith bit slice

The switches SW1 and SW2 steer the data flow in the circuit and the control signals Ctl1 and Ctl2 for these switches are broadcast (i.e. forked) to all bit slices. In each iteration the ring consumes a set of control signals.

SW1 controls the Par input to the iteration. It can either be the P[i-1] bit from the i-1th bit slice (corresponding to a normal iteration step in which the parallel operand is shifted), or it can be an externally supplied value (corresponding to loading of a new P-operand). SW2 controls the Sum input to an iteration step. It can either be the sum bit from the previous iteration step (corresponding to another normal iteration step) or it can be "0" when the result of the computation is output.

5.3 The PISO

The PISO is a double shift register that loads the S-operand and the Z-pattern as described in section 4. Both are shifted out of the PISO with the least significant bit first. The operand bit, S, is input to the RING and Z-pattern bit, Z, is input to the CONTROL block. When the S-operand is shifted out to the right, zeros are shifted in from the left. These zeros enable the final conversion to binary representation to be implemented by taking the RING through a number of additional iterations during which the PISO delivers the necessary zeros. The PISO is build from latches and asymmetric switches.

5.4 The CONTROL block

A diagram of the CONTROL block is shown in figure 10. The control block generates the signals Ctl1 and Ctl2 which control the switches in the RING, the PISO and the CONTROL unit itself. Ctl1 and Ctl2 goes through one four phase cycle per iteration step, and the input signals In0, In1 and In2 to the combinational circuit that generate the control signals must also follow this protocol.

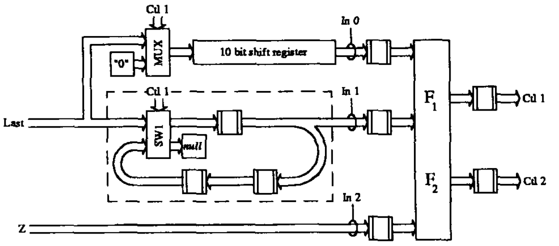


Figure 10: The control block

Input to the control unit is the Z-pattern in bit serial form (one four phase cycle per iteration step) and the last-operand tag, Last, associated with the parallel operand (going through a four phase cycle only when a pair of operands is loaded). From these two signals the signals In0, In1 and In2 are derived:

In0 is a delayed copy of the last-operand tag. It is used to terminate the inner product computation. The shift register is used as a counter and its length corresponds to the number of bit slices in the RING (i.e. the number of bits in the result).

In1 is a copy of the last-operand tag (one four phase cycle in each iteration step). If the circuit is in the process of multiplying the last pair of operands (In1 = T) it must continue to iterate until In0 becomes true. In any other case the circuit may stop iterating as soon as the Z-pattern indicates that the remaining bits in the serial operand are leading zeros (In2 = T).

In2 is simply the Z bit from the zero-pattern from the PISO.

It should be noted that the CONTROL unit is based on the same circuit elements that are used in the PISO and in the RING (switches, latches and combinational circuits) and that its structure is also based on pipelines and rings. In addition to the small ring from which the signal In1 is derived, the CONTROL unit contains some additional rings by virtue of the switch control signal Ctl1.

6 Performance analysis

The updating of storage elements (latches) in a delay insensitive circuit is controlled by local handshaking (rather than by global clock), and an analysis of the performance of a delay insensitive circuit involves not only the structure of the circuit but also the initialization of the circuit and the way in which the circuit is used by the environment.

An understanding of this dynamic behaviour is essential to the design of efficient delay insensitive circuits. This important topic has recently been addressed in [9, 12]. It is beyond the scope of this paper to go into details. The interrested reader is referred to [13] in which we give a detailed introduction to the topic and in which we analyze and optimize the performance of the vector multiplier design (using the techniques and results of [9]).

The result of the analysis is that the cycle time of the RING is 15.8 ns. This is the minimum possible time for an iteration step. Unfortunately broadcasting (i.e. forking) of the switch control signals constitute a bottleneck and the design performs an iteration step in 24.6 ns. These figures are computed from the delays in C-elements and gates. The effects of wiring capacitance has not been taken into account and the figure therefore conforms nicely with the result of a post layout simulation that showing a 30 ns. iteration step time [13].

7 Physical implementation

A test chip has been fabricated on EUROCHIP's October 1991 run at ES2 (European Silicon Structures Inc.) in a 1.5 micron CMOS technology. This chip multiplies vectors with 4-bit elements, and it has a 10-bit accumulator for the result. The test of the fabricated chips showed that they are fully functional.

The physical implementation is a standard cell layout. The AutoCells tool (part of the GDT design system from Mentor Graphics Inc.) has been used for the physical implementation. As C-elements are used extensively in the design, we have developed a C-element standard cell generator for the GDT design system. The chip contains 12.450 transistors, and the area of the core of the chip is 7.3 mm^2. The area including pad-cells is 18 mm^2.

8 Conclusion

We have described a data flow based approach to the design of delay insensitive circuits. The underlying structural concept is simple: pipelines and rings. These can be combined into more complex structures – called multi-rings – by joining and forking of signals. Such multi-rings can be implemented from a small set of building blocks, consisting of latches and a variety of combinational circuits: functional blocks, switches etc. A technique for the design of delay insensitive building blocks (including combinational circuits) called "DIMS" (Delay Insensitive Min-term Synthesis) has also been presented.

This design technique has been successfully applied to the design of a nontrivial delay insensitive circuit. This circuit computes the inner product of two vectors whose elements are unsigned integers, and it is based on an iterative serial-parallel multiplication algorithm. The design has been fabricated through EUROCHIP. The chip is a standard cell design containing 12.450 transistors, and it is fully functional.

The experiment described in the paper is part of our ongoing work to construct CAD tools supporting our design approach. This includes a synthesis tool which uses the DIMS technique and tools for formal verification.

Acknowledgement

We are grateful to Mark Greenstreet and Ted Williams who have both strongly influenced our design approach. This has work has been supported by The Danish Technical Research Council. The CAD tools from Mentor Graphics Inc. have been provided via EUROCHIP.

References

[1] A. Martin et al., "The Design of an Asynchronous Microprocessor," *Decennial Caltech Conference on VLSI*, Ed. C.L. Seitz, MIT Press 1989, pp. 351-357.

[2] I. E. Sutherland, "Micropipelines," *Communication of the ACM*, Vol. 32, no. 6, 1989, pp. 720-738.

[3] C.H. van Berkel, C. Niessen, M. Rem, and R. Sajs, "VLSI Programming and Silicon Compilation: A Novel Approach from Philips Research," *Proc. of ICCD 88*, IEEE Computer Science Press, 1988, pp. 150-166.

[4] J. Staunstrup and M. R. Greenstreet, "Designing delay insensitive circuits using 'Synchronized Transitions' ," In Claesen (ed.), *Formal VLSI Specification and Synthesis. VLSI Design Methods - vol. I*, North-Holland/Elsevier, 1990, pp. 209-226.

[5] P. C. Treleaven, D. R. Brownbridge, R. P. Hopkins, "Data-Driven and Demand-Driven Computer Architecture," *ACM Computing Surveys*, Vol. 14, No. 1, March1982, pp. 93-143.

[6] N. P. Singh, "A Design Methodology for Self-Timed Systems," M.Sc. thesis, MIT/LCS/TR-258, Laboratory for Computer Science, Massachucetts Institute of Technology, February 1981.

[7] T. Verhoeff, "Delay Insensitive codes – an overview," *Distributed Computing*, Vol. 3, no. 1, 1988. pp. 1-8.

[8] M. R. Greenstreet, T. E. Williams and J. Staunstrup, "Self-Timed Iteration," *Proceedings of VLSI-87*, North-Holland, Amsterdam, 1987.

[9] T. E. Williams, "Self-Timed Rings and their Application to Division", Ph.D. thesis, Department of Electrical Engineering and Computer Science, Stanford University, May 1991.

[10] H. Hulgaard and P. H. Christensen, "Automated Synthesis of Delay Insensitive Circuits," M.Sc. thesis (ID-E 511), Dept. of Computer Science, Tech. Univ. of Denmark, Lyngby, 1990.

[11] C. J. Majithia and R. Kita, "An Iterative Array for Multiplication of Signed Binary Numbers," *IEEE Trans. Comput.*, Vol. C-26, Feb. 1971, pp. 214-216.

[12] M. R. Greenstreet and K. Steiglitz, "Bubbles Can Make Self-Timed Pipelines Fast," *Journal of VLSI Signal Processing*, 2, 1990. pp. 139-148.

[13] J. Sparsø, J. Staunstrup, and M. Dantzer-Sørensen, "Design and performance analysis of delay insensitive multi-ring structures," Technical report, Department of Computer Science, Technical University of Denmark, May 1992.