LOGIC COMPLETION DETECTION IN PROGRAMMABLE LOGIC DEVICES

C. A. Sampaio, R. T. Vaz da Silva, A. I. Reis, R. P. Ribas
Instituto de Informática – UFRGS
CxP.15064, CEP 91501-970 Porto Alegre – RS (BR)
{csampaio, rsilva, andreis, rpribas}@inf.ufrgs.br

ABSTRACT

The advantages of asynchronous circuits, in particular self-timed implementations, are being seriously considered to reduce the limitations in synchronous design. Today Programmable Logic Devices – PLDs (FPGAs and CPLDs), on the other hand, are a very attractive option to ASIC implementation due mainly to the fast prototyping feature. Design techniques to make the Logic Completion Detection for self-timed circuits and the performance of PLDs when implementing these blocks are discussed on this paper. An 8-bits Ripple Carry Adder is used as case study, taking into account programmable devices from Altera Inc. and Xilinx Inc. and their respective CAD environments.

1. INTRODUCTION

The asynchronous architecture of circuits was proposed long time ago, in the 50´s. Nowadays it has gained again the focus of the academic and industrial communities, due to some advantages over the traditional synchronous design, which is becoming very limited in some aspects, like the clock distribution over the chip.

Some of these advantages are the low and constant power dissipation, once there is no global clock, each part of the circuit works only when requested, otherwise remain unharmed. It also reduces current spikes and the emission of electromagnetic noise. The asynchronous circuits compute on their best speed, because it does need to be projected to wait for the worst case latency to move to the next process [1].

On the other hand, the use of Programmable Logic Devices (PLDs) has greatly increased. It is justified by the fast prototyping of the ASIC (Application Specific Integrated Circuit), the approximate zero risk of project since the logic cells and components are easily reconfigured, and the technological advances which have increased the density and complexity of the circuits implemented in a single chip.

The article's purposes are to discuss some techniques to perform the logic completion detection in combinational blocks for self-timed circuits and to evaluate commercial PLD's performance when synthesizing these blocks. Components from Altera and Xilinx vendors have been considered to synthesize the case study, an 8-bits Ripple Carry Adder using the dual rail protocol.

2. LOGIC COMPLETION DETECTION

In the asynchronous circuit design, the absence of global clock signal demands the use of handshaking protocols to coordinate the execution of the signal processing blocks. In other words, these protocols are responsible for the correct functionality of the circuit, since the control signals communicate the end of a calculation of one block and the data availability to another block which could, in turn, start the next computation [1].

In this approach, the signal processing, executed by the combinational blocks (here called function blocks), has to wait for a request command before starting the computation and provide the logic completion detection (acknowledge signal). The communication protocol uses the request and acknowledge signals to activate the function blocks and then the data flows.

The Dual Rail Protocol was used in order to implement the Logic Completion Detection on the commercials PLDs, since any other methodology is not suitable in these devices. And dealing with this protocol two ways are possible in the PLDs, one through Delay Insensitive Minterm Synthesis (DIMS) [2] and the second Sum-of-Products and Product-of-Sums method (SOP/POS). These methods generate hazard-free blocks, but unfortunately both cases demand a considerable amount of area.

In DIMS approaches, we redesign the way how to elementary Gates are constructed. This new elementary gates are now constructed through a combination of "C-elements" an "OR" gates in a PLA like structure. DIMS are classified as strongly indicating implementations, which means to say that its outputs never become valid before all the inputs and its outputs never become empty before all of its inputs do.

The "C-elements" cells that compose the circuits generate all minterms needed to implement the function. The truth table that generates this kind of circuits is divided in three groups of rows that define the outputs when the inputs are: (1) the "empty" word (the circuit responds setting all outputs at low level logic); (2) the intermediate values (the circuit responds without changing the outputs); (3) an "valid" input (the circuit responds setting the correct value in the output).

Figure 1 exemplifies how can be implemented an AND logic function using dual rail and DIMS. Each logic value is represented by its true and false signals.

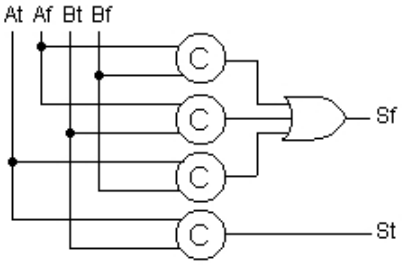

Figure 1 – AND gate using DIMS

The sum-of-products and product-of-sums (SOP/POS) method is other alternative dealing with dual-rail protocol, once its outputs are hazard-free. In this method the traditional truth table are used, as well all the well known methods of extracting Boolean equations from it, like the Karnaugh map. And merging SOP and POS blocks in the same circuit does not cause any risk. The truth table must have empty outputs, true and false values set to '0', for every combination of inputs where one of them is empty. And for the cases where any input has the true and false values set to '1', the output can be "don't care", because this never happen in the dual-rail protocol. This process generates function blocks called weakly indicating, because it starts to compute as soon as it has some valid inputs and it produces empty outputs just after receiving some empty input [2]. It deserves only a little remark; all outputs should not become valid before all inputs do and exactly the condition for the empty transition.

Facing the two methodologies exposed above it is possible to take some conclusions. The DIMS logic is easily extracted, once it works with a simplified truth table where the signals are represented with values 'true' or 'false' instead of their binary codification. In the other hand the circuits created with SOP/POS are potentially faster and smaller, due to all simplifications that can be made on them. Another difference is about the circuit behavior. Dims generates strongly indicating blocks, then some extra circuitry must be added to ensure that, even if an input will keep its value constant, it alternates between valid and empty states. While with SOP/POS there isn't this problem and a constant input can keep its valid state, since one of others inputs alternates their states, saving in that way some silicon area.

3. CASE STUDY AND RESULTS

Altera Max plus II v 10.1 and Xilinx WebPack ISE 5 CAD tools were used to synthesize the circuit exposed before on the commercial PLDs and to extract the results of these implementations. From Altera it was taken into account the Max9000 (CPLD) and the Flex10k (FPGA) families and from Xilinx the Virtex2 (FPGA) family.

Three implementations of an 8-bit Ripple Carry Adder were made, one synchronous and two asynchronous confronting the methodologies discussed here. Tables 1 expose the results obtained. For the asynchronous circuits the time delay (Td) measured was the mean value of the worst and best cases, due to their characteristics of working in the best speed for each input vector.

Table 1a. Results for Max9000

	# LCs *	Td (ns)
Synchronous	24	26,8
Dims	83	79,5
SOP/POS	72	66,5

* LCs – logic cells

Table 1b. Results for Flex10k

	# LCs	Td (ns)
Synchronous	16	26,8
Dims	101	60,8
SOP/POS	77	80,5

Table 1c. Results for Virtex2

	# LCs	Td (ns)
Synchronous	23	8,237
Dims	101	25,132
SOP/POS	168	33,857

4. CONCLUSIONS

This work tried to analyze methods to design function blocks for self-timed circuits which were capable of doing the logic completion detection in PLDs. Taking the results we can see that the SOP/POS, method has a better performance in most cases and also regarding to some advantages shown in the end of section 2, this method can be considered as a reasonable choice. Analyzing the results it becomes clear that an asynchronous circuit consumes a greater area on common PLDs, specially due to their lack of structures to help detecting the end of calculation, and consequently generates a greater delay. As a future work it can be studied a new PLD model which is dedicated for the implementation of asynchronous circuits since the common PLD used nowadays don't fit well in this kind of approach.

5. ACKNOWLEDGMENTS

The students C. Sampaio and R. Vaz da Silva receive their scholarships from FAPERGS and CNPq/Milenio respectively, and the professors A. I. Reis, R. P. Ribas have research support from CNPq.

6. REFERENCES

[1] Ivan E. Sutherland, *Micropipelines*. Communication of the ACM, June 1989

[2] Jens Sparso and Steve Furber. *Principles of asynchronous circuit design – A system Perspective*. Kluwer Academic Publishers, September 2001