Using Standard ASIC Back-End for QDI Asynchronous Circuits: Dealing with Isochronic Fork Constraint

Mehrdad Najibi Kamran Saleh Hossein Pedram

{najibi, k.saleh, pedram}@ce.aut.ac.ir

Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic)

424 Hafez Ave, Tehran 15785, Iran

ABSTRACT

Asynchronous circuits already have shown their benefits. The main drawback is the lack of powerful CAD and layout generation tools limiting the widespread use of the asynchronous methodology. QDI asynchronous circuits are known as a powerful category of asynchronous circuits targeting performance and power driven design. In this paper we addressed standard cell implementation of the template based QDI circuits utilizing standard layout generation tools. This is achieved by analyzing and removing outer cell isochronic fork constraint which is the main timing constraints limiting the standard layout generation. The isochronic fork free final netlist has 10-20% area overhead in average which is the cost of facilitating the use of standard CAD tools.

Categories and Subject Descriptors

B.6.1 [**Logic Design**]: Design Styles–*Asynchronous circuits*

General Terms

Design

Keywords

Asynchronous Circuits, Quasi-Delay Insensitive, Standard-Cell Layout.

1. INTRODUCTION

Reduction in the size and the growth in number of the transistors in contemporary circuits signify the problem of global synchronization. One solution is to eliminate the global clock signal and take advantage of asynchronous design methods[1][2].

Persia [3] is an asynchronous synthesis tool intended to provide an automatic synthesis in the field of QDI [4] (Quasi Delay Insensitive) asynchronous circuits. In this paper the techniques used in template synthesizer of the Persia (TSYN) to eliminate global timing constrained and simplify layout generation will be introduced.

While traditionally asynchronous pipeline templates [5] are implemented as full-custom transistor netlists using non-standard layout tools, standard cell approach is gaining popularity [6]. Lines introduced transistor level PCFB/PCHB templates [5] which are later used in the synthesis flow of the Persia. The outlines of the standard cell implementation of these templates will be presented in this paper while the main focus is on the isochronic fork timing constraint in standard cell implementation. Each pipeline template in Persia is composed of a number previously laid out standard cells that can be connected to each other with nearly no special timing constraints. The cells are designed to encapsulate all isochronic forks inside to simplify the task of layout generation. TSYN can optionally produce a completely isochronic fork-free netlist at the cost of 10-20% increase in the layout size as will be discussed in the paper. For the purpose of benchmarking, the synthesis results for a Read Solomon decoder and a simple 8–bit RISC processor will be demonstrated in the results section.

The outline of the paper is as follows: section 2 briefly introduces QDI templates. Traditional and standard cell implementation for PCFB QDI templates are discussed in subsection 2-1 and 2-2. In section 3 the nature of the isochronic forks constraint is discussed. Section 4 is dedicated to the isochronic forks in the standard cell implementation of the QDI templates. Techniques to eliminate inter-cell isochronic forks are also presented in this section. Results are presented in section 5, and finally section 6 concludes the paper.

2. QDI TEMPLATE BASED CIRCUITS

As one of the main benefits of asynchronous design is the relaxation of timing constraints, the correct functionality of QDI asynchronous circuit is still requires some weakened timing constraints.

Most of the asynchronous methodologies including QDI assume unbounded finite delays for both functional elements and wires which can be considered as a great timing relaxation. In QDI methodology for a special class of branched wires known as *isochronic forks* while the unbounded finite delay assumption is still valid for both the root and all the branches, correct functionality requires that the delay of the branches are nearly equal. More precisely the difference between the delays of the branches must be lower than the delay of a functional unit. It has been shown that practical asynchronous circuits can not be made without isochronic forks [4].

In this section we briefly review the QDI PCFB templates and their traditional synthesis method and then we provide a method to synthesize a PCFB template to standard-cell library.

2.1 Traditional PCFB Implementation

A PCFB template is an asynchronous buffer circuit that in each cycle of its operation reads some inputs, performs a particular calculation, and then writes the results to one or more of its output ports. All I/O read or write operations are done using dual-rail four-phase handshaking protocol.

Figure 1 shows the internal implementation of the simple buffer described using the traditional custom transistor netlist. The following sub-circuits can be enumerated for the circuit:

1. Output generation circuit
2. Input validity check circuit
3. Output validity check circuit
4. A sub circuit that generates the acknowledge of inputs
5. A sub circuit that generates *en* signal

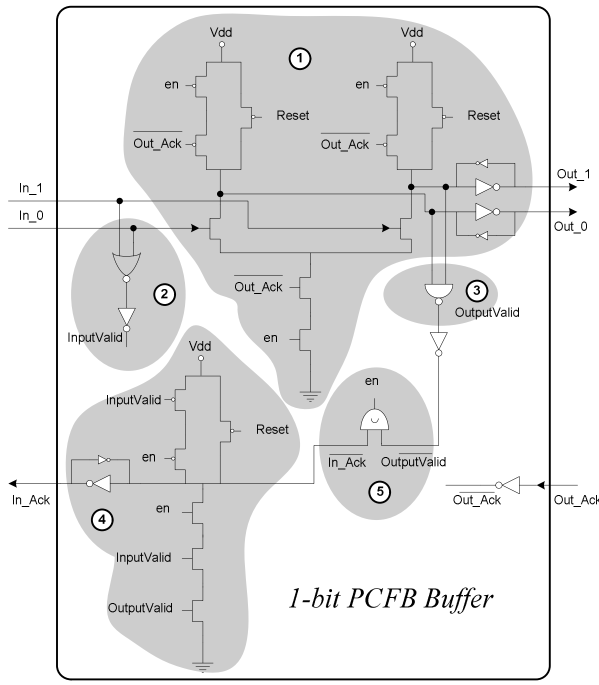

Figure 1: Internal structure of a simple on-bit PCFB buffer

2.2 Standard cell PCFB Implementation

In general, besides some parts of I/O validity check circuitry that can be directly implemented using standard gates, most parts of a PCFB template can not be built using standard cell.

I/O validity check: Input and output validity check is the only part of a PCFB that can be implemented using standard gated and C-Elements [7] in a straight forward manner without any modifications.

Input Ack Cell: The circuit that generates the acknowledge signal of input ports, always has a unique form of Figure 2. So we can create a special cell for this purpose in the standard library.

en Signal Cell: In PCFBs with unconditional input ports, *en* signal can be created by combining acknowledge signals of the inputs and the validity signals of the outputs using a C-Element. As shown in Figure 3, we can create this signal with standard cells.

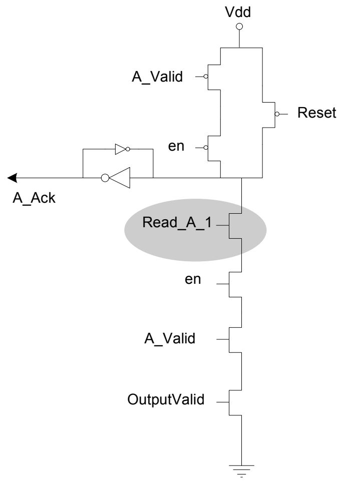

Figure 2: Input Ack Circuit

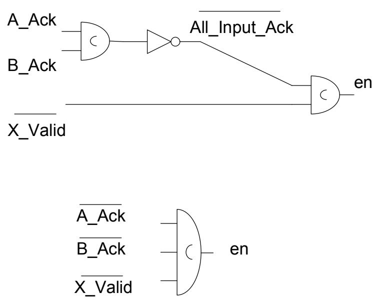

Figure 3: Generating *en* with standard cells

Generating outputs: Traditional PCFB templates have transistor-level nonstandard net-lists and each function must be implemented using a specific transistor network. In contrast to the traditional way, we can implement F as a combination of special standard library operators using a simple technology mapper which targets dual rail standard library primitives. Figure 4 shows the implementation of network of the simple function F = xy + z using special dual rail standard cells.

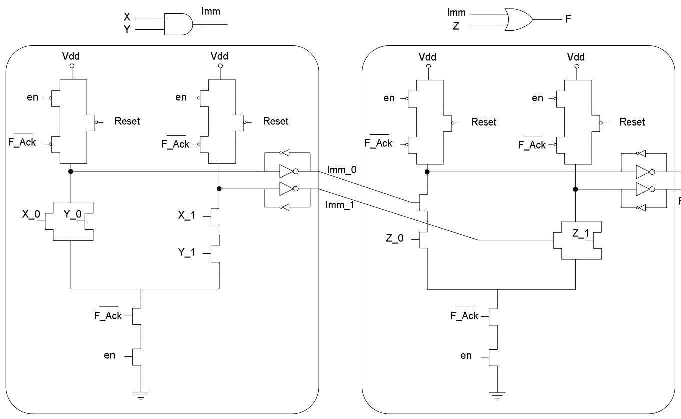

Figure 4: Implementation of F = xy + z with standard cells

3. THE ISOCHRONIC FORK CONSTRAINT

For correct functionality, isochronic fork property must be asserted in the final implementation of the circuit. Violating this property can lead to introduction of logical hazards, premature firing of signals, and generation of unwanted tokens.

Hazards must be avoided in asynchronous design since the values of the signals are continuously monitored by its fan-out gates. Theoretically hazard avoidance is done by means of the concept of indication. A transition on signal *A* indicates a transition on signal *B* if the former transition can not be occurred until the latter transition completes. There will be no hazards in a system if every signal is indicated by at least one other signal in the design [7].

Indication is a property of the elementary gates; Depending on the logical structure of a gate, it may or may not indicate each of its inputs. In general in each CMOS gate, every input signal that goes to the pull up network is indicated for falling transitions as far as there is no bypassing path available in the pull up. The same is true about the signals connected to pull down for rising edge transition. Every signal that is connected to both pull down and pull up is indicated by the output of the CMOS gate for both rising and falling transition.

In a QDI circuit any forks that has non-indicated branches must be implemented as an isochronic fork. So the major candidate branched wires for isochronic forks are those that their output branches go to the inputs of the gates that can not acknowledge both rising and falling transitions on their inputs.

As an example of hazard introduction in the circuits due to the violation of an isochronic consider Figure 5.

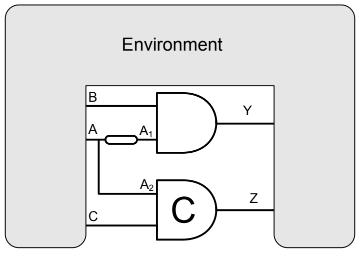

Figure 5: Hazard due to violation of iso. fork

First of all, the fork is isochronic because the AND gate doesn't indicate its input for falling edge transition. Consider the situation that the branch connected to the AND gate *(A_1)* has a larger delay value; A falling edge on this branch is not indicated so while C-elements indicated that its corresponding signal *(A_2)* is low, it is still possible that A_1 has a high value. Considering that at the start of each circuit cycle the value of *A* is assumed to be zero a premature transition on *Y* can be generated as soon as B goes high. This pre-matured firing can lead to a static hazard if A_1 goes low close enough to rising transition on B.

3.1 Dealing with Isochronic Forks in The Layout Generation

The simplest way to handled Isochronic forks is using manual layout for coarse grained logic cells (at PCFB level for example) and only allow completely delay insensitive interconnects inside the cells. This approach is not applicable for large practical designs and can not be used in a fully automated asynchronous synthesis tool knowing that the logic portion of the PCFB/PCHB must be implemented arbitrary.

In another way, Isochronic forks can be handled in physical design by letting placement and routing to be aware of this new timing constraint. While some efforts are made on isochronic fork-aware placement and routing, there is no reliable layout tool available yet.

Persia synthesis tool proposes a new solution for the problem. This method is based on handling isochronic forks inside the manually laid out standard cells. Inter-cell connection between these cells while is not completely DI has no isochronic forks. Since Persia doesn't impose any timing constraints on physical design, it can work with every standard back-end layout tool. The major technique that assists Persia in achieving this goal is the innovative selection of the standard cells and some modifications in the template synthesizer.

4. ISOCHRONIC FORKS IN STANDARD CELL BASED PCFB TEMPLATES

In this section we analyze the nature of the branched wires in general circuits synthesized by Persia using the set of standard cells introduced in previous sections.

Isochronic forks can be distinguished by considering internal structure of the cells. Any branch that is not indicated by the cell connected to makes the whole branched wire isochronic.

According to our analysis, branched wires can be classified to the following categories:

- ***Input Signals:*** Each rail of the input signals of a PCFB is branched and connected to both *inputValid* and function cells.
- ***InputAck*:** internal *inputAck* signal is forked to *inputAck* port of the corresponding input and the circuit producing the *allInpAck* signal which is required to produce the *en* signal.
- ***OutputValid*:** output valid goes to all *inputAck* generator cells and the *en* cell.
- ***en signal*:** *en* signal goes to each *inputAck* Generator cell, every function cell, and the cells that compute conditions in conditional input/output PCFBs.
- ***OutputAck***: *outputAck* branches to all function cells.

Hereby, we will analyze each of these classes to indicate that which inter-cell isochronic forks are exists in the final implementations. We also present a number of solutions that are used to eliminate global isochronic forks.

4.1 Forks on Input Signals

The Branch on the input signals is the most important isochronic fork on PCFB template. Either the template is implemented in custom transistor level or using our proposed standard cells, we must care about the implementation of this type of forks. Each of the rails of every dual rail input is connected to both input validity check and the function circuitry as shown if Figure 6.

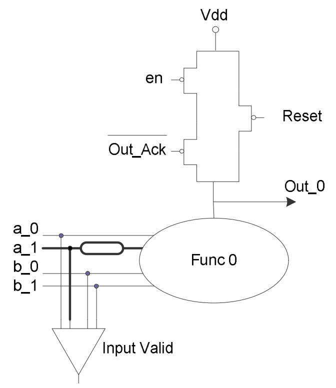

Figure 6. Forks on Input Signals

While input validity circuitry completely indicates both rising and falling transition on each rail of the inputs, function cells can not perform complete indication on the inputs. First of all, since input signals only go to the pull down network, they can not be indicated for falling edge transitions.

Secondly, depending on the general structure of transistors used in the pull down of each function cell, parallel paths may exist in the pull down of the gates which can harm the indication of the rising edge of the input signals. For example in an OR gate there exist two parallel paths in the pull down, therefore activation of either path can lead to the validation of the output. This property is known as weak indication that may generate new isochronic forks in an implementation.

The problem with this type of forks is that the non-indicated inputs can remains valid and produce unwanted tokens in the next execution cycle of the PCFB. To eliminate weak indication, it is required that no gate can produce output until all of its inputs get valid. This is done by adding all required minterms to the pull down network such that the activation of each path in the pull down ensures the validation of all the inputs. For example consider the pull down network of a 3-input OR gate shown in Figure 7 (a). By adding all the min-terms to the pull down and performing the possible transistor sharing, the network of Figure 7 (b) which fully indicates the validation of all inputs can be obtained.

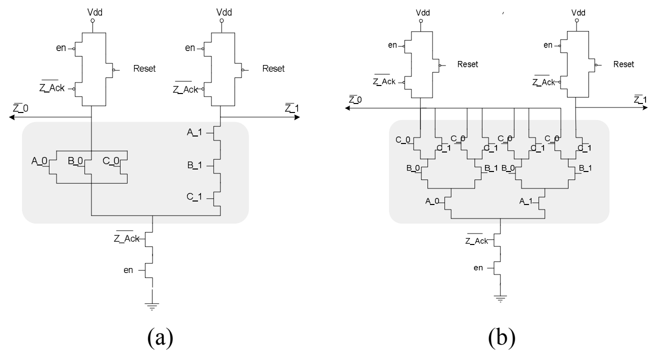

Figure 7. Full validity indication (a) ordinary gate (b) full validity indicating gate

Table 1 shows the amount of area overhead resulted from applying above solution to standard dual railed logic gates.

Table 1. Strong Indication Transistor Overheads

	Weak Indication	Strong Indication	Overhead
Or2	21	23	8.7%
And2	21	23	8.7%
Nand2	21	23	8.7%
Nand3	23	31	25.8%
Nand4	25	47	46.8%

By accepting this amount of area overhead, the input forks can be implemented without any special timing considerations.

4.2 Forks on the OutputAck Signal

In standard cell implementation of the PCFB templates, general functional expressions must be realized as stated before. In our synthesis flow this is done in a technology mapping section. The output of the technology mapping is a netlist of dual rail library gates that are cascaded to form the general function. As it can be seen in Figure 4 each of these gates must receive *outputAck* signal which is responsible of neutralizing the intermediate outputs. It can be shown that the branched *outputAck* must be implemented as isochronic fork.

Consider Figure 8 which shows a function composed of two cascaded gates.

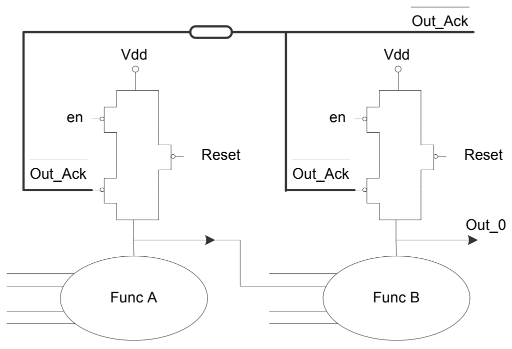

Figure 8. Forks on the *OutputAck* Signal

If Function A receives the *outputAck* signal with considerable delay comparing to Function B, it can be expected that its intermediate output remains valid even after neutralization of Function B which produces the final output. Since the output validity circuitry only indicates neutralization of the output of Function B, and no signal indicates the neutralization of intermediate outputs, it is always possible to have unwanted data tokens on intermediate signals which can disturb the functionality of the circuit in next execution cycle.

To solve this problem, the neutralization of the intermediate signals must be indicated too. This can be done by adding weaker validity check circuitry on the intermediate signals which only indicates neutralization. While adding weak validity checks have the disadvantage of adding more area it can completely removes the global isochronic assumption on the *outputAck* signal.

4.3 Fork on the InputAck Signal

The other branched wire exists in the final standard cell implementation is on *InputAck* which is connected to both the corresponding outgoing *inputAck* port and the *allInpAck* circuit.

The *allInpAck* circuit is a circuit composed of OR gates followed by a C-element gate as shown in Figure 9.

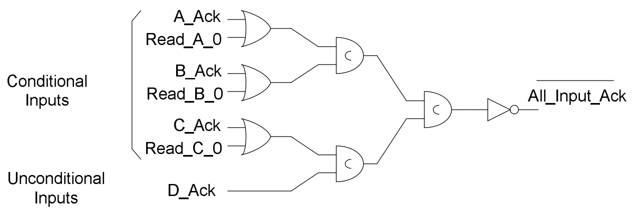

Figure 9. All*InputAck* circuit

While the branch to the *inputAck* port is indicated due to the delay insensitive assumption on handshake ports, it seems that *allInpAck* circuit can not indicate the rising edge of the *inputAck* signals. The OR gates seems not to be able to indicate the rising edge of its input but this is not the case here, since by the nature of the logic it is guaranteed that the two input signals of the OR gate is completely exclusive; a high value on the read condition signal (*Read_A_0*) informs that no acknowledge signal (*A_Ack*) is ever generated and so the OR gate in this special case can indicate both falling and rising edge transitions on both of its exclusive inputs. By this discussion it is fairly clear that no special care must be considered in implementing *inputAck* branched signal as it is not an isochronic fork.

4.4 Fork on the outputValid Signal

The output valid signal goes to all *inputAck* cells and to the *en* generation circuit which is a simple C-element gate. Since *inputAck* cell does not indicates *outputValid* for falling transition (Figure 2), output valid must be implemented as an inter-cell isochronic fork which is not acceptable in our design methodology.

Consider the situation where one of the *inputAck* cells observes the *outputValid* signal after the time *en* generation circuit sees. On receiving the falling edge of the *outputValid*, the enable will goes high and the next execution cycle of the PCFB initiated. In this situation a premature acknowledge may be generated for the corresponding input while for that *inputAck* cell both *en* and *outputValid* are high (refer to Figure 2). To solve this problem it is only required to ensure that all the *inputAck* cells observe the output valid sooner than the *en* generation circuitry. This is down by a simple layout trick in implementing *inputAck* cell. If we ensure that *outputValid* passes through a chain of *inputAck* cells before going to the C-element of the *en* generation circuitry by changing the interconnection of these cells as shown in Figure 10, it can be guaranteed that no premature input acknowledge can be generated. So by a simple layout technique by paying nearly no overheads, we can get ride of the isochronic fork on *outputValid* Signal.

4.5 Fork on the en signal

No special consideration must be made for *en* signal as all the function cells and the *InputAck* cells have it in their both pull up and pull down networks.

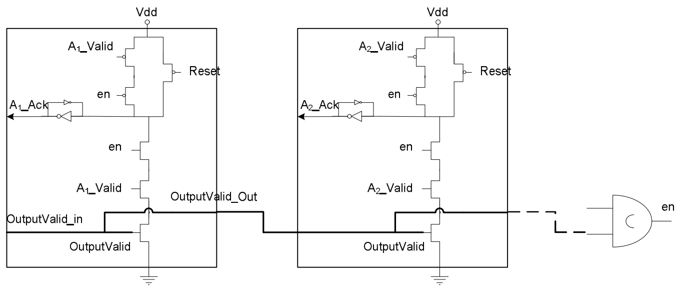

Figure 10. Chaining inputAck cells

5. RESULTS AND ANALYSIS

The proposed techniques to eliminate inter-cell isochronic forks are implemented in Persia asynchronous synthesis tool. The results for a set of benchmark circuits from simple buffers to a Reed Solomon Decoder and a simple 8-bit RISC is depicted in Table 2 and also shown graphically in Figure 11.

An area overhead of 10%- 20% is the cost for simpler layout implementation with less timing constraints. By elimination of the inter-cell isochronic forks the final netlist of standard cells can be placed and routed using standard layout generation tools which is an important benefit; recalling that non of the commercially available layout tools consider a timing constraint similar to isochronic requirement.

Table 2. Benchmarking results

	Transistor Count		Area Overhead
Circuits	Global Iso Fork	No Global Iso fork	
Buffer1	70	84	16.67%
Buffer4	229	263	12.93%
Buffer8	449	503	10.74%
Buffer32	1709	1911	10.57%
Mux2_1	280	320	12.50%
Demux1_2	208	238	12.61%
GMULT	1502	1928	22.10%
Incrementer	2561	2891	11.41%
RS Decoder	86490	106730	18.96%
8-bit RISC	426733	517130	17.48%

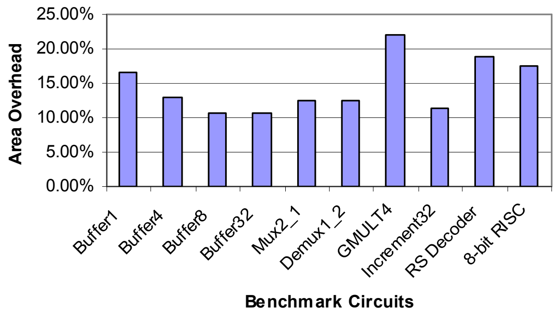

Figure 11. Area Overheads

6. CONCLUSION

A novel standard cell implementation technique for asynchronous QDI circuits based on pre-designed templates (PCHB/PCFB) was introduced. Using standard cells is the basic step toward utilizing standard layout tools however as have been shown for QDI circuit it is not sufficient in the presence of global isochronic fork timing constraints.

The nature of isochronic fork in asynchronous circuits and the ways violating this constraint can harm system functionality were discussed extensively. The major isochronic forks in proposed standard cell implementation for QDI PCFB/PCHB templates are classified.

Global inter-cell isochronic forks can be eliminated by accepting some area overheads. By means of encapsulating isochronic forks inside the standard cells no timing constraints are imposed on inter-cell interconnect which is the key to use standard cell layout generation tools for asynchronous circuits.

REFERENCES

[1] Alain J. Martin, Mika Nyström, Karl Papadantonakis, Paul I. Pénzes, Piyush Prakash, Catherine G. Wong, Jonathan Chang, Kevin S. Ko, Benjamin Lee, Elaine Ou, James Pugh, Eino-Ville Talvala, James T. Tong, Ahmet Tura: *The Lutonium: A Sub-Nanojoule Asynchronous 8051 Microcontroller*. ASYNC 2003

[2] S. B. Furber, D. A. Edwards and J. D. Garside *AMULET3: a 100 MIPS Asynchronous Embedded Processor*. ICCD'00 17-20th September 2000

[3] http://www.async.ir

[4] A. J. Martin, *"Compiling communicating processes into delay-insensitive VLSI circuits,"* Distributed Computing, vol. 1, no. 4, pp. 226–234, 1986.

[5] Andrew M. Lines. *Pipelined asynchronous circuits*. Master's thesis, California Institute of Technology, Computer Science Department, 1995. CS-TR-95-21.

[6] Marcos Ferretti, Recep O. Ozdag, Peter A. Beerel: *High Performance Asynchronous ASIC Back-End Design Flow Using Single-Track Full-Buffer Standard Cells*. ASYNC 2004: 95-105

[7] J. Sparso, S. Furber, "*Principles of Asynchronous Circuit Design – A System Perspective*", Kluwer Academic Publishers, 2002.