Technology Mapping for Area Optimized Quasi Delay Insensitive Circuits

Bertrand Folco, Vivian Brégier, Laurent Fesquet, Marc Renaudin
TIMA Laboratory, 46 av. Félix Viallet
38031 Grenoble – FRANCE
{Bertrand.Folco, Vivian.Bregier, Laurent.Fesquet, Marc.Renaudin}@imag.fr

Abstract

Quasi delay insensitive circuits are functionally independent of delays in gates and wires (except for some particular wires). Such asynchronous circuits offer high robustness but do not perform well to automatically synthesize and optimize. This paper presents a new methodology to model and synthesize data path QDI circuits. The model used to represent circuits is based on Multi-valued Decision Diagrams and allows obtaining QDI circuits with two-input gates. Optimization is achieved by applying a technology mapping algorithm with a library of asynchronous standard cells called TAL. This work is a part of the back-end of our synthesis flow from high level language. Throughout the paper, a digit-slice radix 4 ALU is used as an example to illustrate the methodology and show the results.

1. Introduction

Asynchronous circuits do not have a global signal to synchronize them. Synchronization between blocks is locally done. Those circuits show very interesting properties such as low power consumption, noise emission, security, robustness, reusability, etc [1].

Today, to adopt the asynchronous technology the industry needs powerful asynchronous tools similar to synchronous ones.

This work is part of the TAST [2, 3] (Tima Asynchronous Synthesis Tool) project, aimed at developing and prototyping such tools. The synthesized circuits in TAST are quasi-delay insensitive (or QDI [4]). QDI circuits are functionally correct independently of delays in gates and wires, apart from the assumption that some forks are isochronic. This kind of asynchronous circuit is particularly robust. But robustness has a cost; these circuits usually have more transistors than the others, especially when standard cells are targeted. Many efforts are directed towards circuit optimization and transistor reduction; one of the main difficulties is to preserve the property of quasi-delay insensitivity [5-9].

2. Contributions

This paper presents a complete standard cells based design flow we have developed as illustrated in Figure 1. Our method uses Multi-valued Decision Diagrams as a model of the circuit that can be optimized while preserving the QDI property. Firstly, the model is generated from a CHP description. Secondly, the model is optimized. A two-input gates circuit is synthesized from the model. Thirdly, a technology mapping algorithm produces the final circuit, using gates from a library of standard asynchronous cells called TAL (TIMA Asynchronous Library).

This design flow includes a general technology mapping algorithm dedicated to QDI circuits. It enables to target any standard cells library, including or not asynchronous cells. The main objective of this work is to reduce the area of the asynchronous circuits. In fact, this is one of the main challenge for the asynchronous circuits to be adopted. Accordingly, the last part of the paper compares results obtained for our asynchronous circuits to its synchronous equivalent.

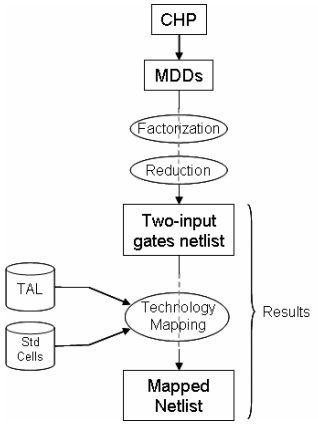

Figure 1 : Asynchronous Design Flow

3. Asynchronous Circuits

3.1. Communication channels and handshake protocol

In asynchronous circuits, a local mechanism is used to perform the synchronization called handshake

protocol. It relies on two signals: request and acknowledgment. When a block needs to transmit data to another, it sends a request signal along with the data, and holds them until it receives the acknowledgment. The request and acknowledgment signals may not be reset before the next communication, making two possible handshake protocols, well-known as two-phase and four-phase protocols. Asynchronous circuits considered in TAST implement the latter. Request, acknowledgment and data are linked together; therefore we consider them as a single entity called communication channel.

3.2. Quasi Delay Insensitivity

A circuit is said QDI (Quasi Delay Insensitive) when its correct operation does not depend on the delays of gates or wires, except for certain wires that form isochronic forks [10]. If a circuit is QDI, a transition on its input must cause a transition on its output. It is said that the transition on the output acknowledges the transition on the input. Mutual exclusion plays a very important role to prove this causality relationship [11].

3.3. Delay Insensitive Code

In QDI circuits, a mechanism must guarantee that when a channel emits a request, its data are available. To achieve this, the request is encoded with the data using a 1-of-n code: n rails are used to implement n possible values, numbered 0 to n-1. When all the rails are '0', there is no data and the request is '0'. The channel is said invalid. When one of the rails is '1', its number is the value of the data, and the request is '1'. The channel is said valid. Other codes, when several rails are '1', are out of the code, and therefore forbidden. The code is said Delay Insensitive since it guarantees that the request signal is always synchronized with the data.

3.4. The Muller gate

Asynchronous circuits need a gate that synchronizes several signals. This gate is called Muller gate (or C-element): when all inputs are equal, the output takes their value; when inputs are different, the output holds its value. Its symbol is a circle.

3.5. An example

Throughout this article, we illustrate our method with the example presented in Figure 2. This example is a digit-slice radix 4 ALU: it computes the function Op between its operands A and B, using the carry Cin and Cout when needed (addition and subtraction). Radix 4 was chosen to demonstrate that the method is not limited to dual rail. The ALU can compute seven different operations (add, sub, and, or, xor, neg, not); therefore Op is encoded with a 1-of-7 code. The CHP code is given in Figure 3.

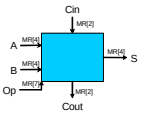

Figure 2: A digit-slice radix 4 ALU.

```
process alu_digit_slice
port( op: in di MR[7], a: in di MR[4],
      b: in di MR[4], cin: in di MR[2],
      s: out di MR[4],cout: out di MR[2];)
begin
variable op: MR[7],a: MR[4],b: MR[4],c: MR[2];
*[
Op?op;
@[
op = '0' => A?a, B?b;                    --add
 @[   a+b<3 => Cout!0, [Cin?c; S!a+b+c];          --K
      a+b=3 => Cin?c; [Cout!c, S!(c=0?3:0)];      --P
      a+b>3 => Cout!1, [Cin?c; S!(a+b+c-4)];      --G
 op = '1' => A?a, B?b;                    --sub
 @[   b-a<3 => Cout!0, [Cin?c; S!b-a+c];          --K
      b-a=3 => Cin?c; [Cout!c,S!(c=0?3:0)];       --P
      b-a>3 => Cout!1, [Cin?c; S!(b-a+c-4)];      --G
 op = '2' => A?a, B?b; S!a and b;  --and
 op = '3' => A?a, B?b; S!a or b;   --or
 op = '4' => A?a, B?b; S!a xor b;  --xor
 op = '5' => A?a; S!(not a+1);     --neg
 op = '6' => A?a; S!(not a);       --not
]]
end
```

Figure 3: CHP code of the example

4. Circuit modeling using MDDs

The first step of our method is to model the circuit with Multi-valued Decision Diagrams (MDDs). It is presented in this section.

A MDD [12] is a generalized BDD (Binary Decision Diagram, [13]) structure. This structure is very interesting for QDI circuits synthesis because it exhibits the notion of mutual exclusion, which plays a valuable role in quasi delay insensitivity.

4.1. Presentation of the Multi-valued Decision Diagrams

A MDD is a rooted directed acyclic graph. Each non-terminal vertex is labeled by a multi-valued variable and has one out-going arc for each possible value of the variable. Each terminal vertex is labeled by a value. Figure 4 presents an example of MDD.

Each path of the MDD from its root to a terminal vertex maps to an input vector (a state of the input variables). The value of the terminal vertex specifies the value that the MDD has to take under this input vector.

The above definition of MDDs does not specify what the label of a vertex can be. Obviously, it can be input ports of the circuit: the logical function that specifies the outputs depends on the inputs.

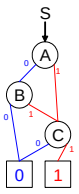

Figure 4: A simple example of MDD

We also want to be able to use internal variables in the circuit. To achieve this goal, we consider an internal variable as a MDD. Therefore, the label of a vertex can also be another MDD, which specifies an internal variable.

4.2. Direct and acknowledgment MDDs

A communication channel holds not only data, but also request and acknowledgment signals. The request signal is computed with the data, thanks to the 1-of-n DI code.

However the acknowledgment signal of the input channels needs to be computed separately. Moreover, not all input channels are read at each computation level; the circuit must not acknowledge an input channel that has not been read.

For each output channel, our model contains a MDD that specifies the logic function computed and is called a direct MDD. For each input channel, it contains one MDD, called an acknowledgment MDD. Acknowledgment signals are considered as 1-of-n DI code with n=1: an acknowledgment MDD has only one terminal, and specifies the conditions under which the channel must be acknowledged. Figure 5 illustrates the MDDs of the example 3.4.

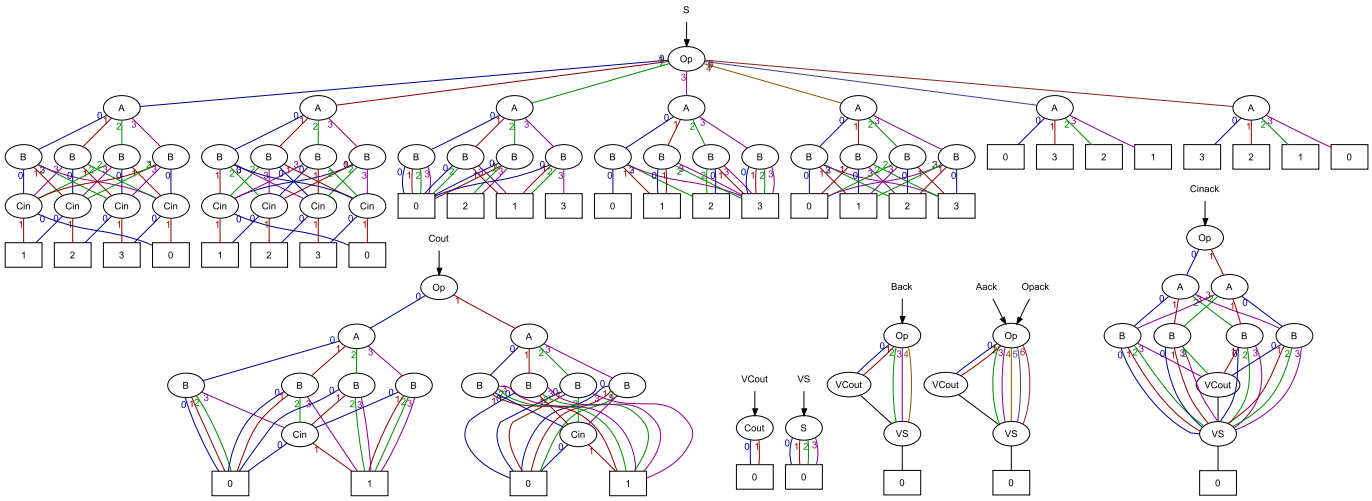

Figure 5: MDDs modeling the circuit specified in 2.4.

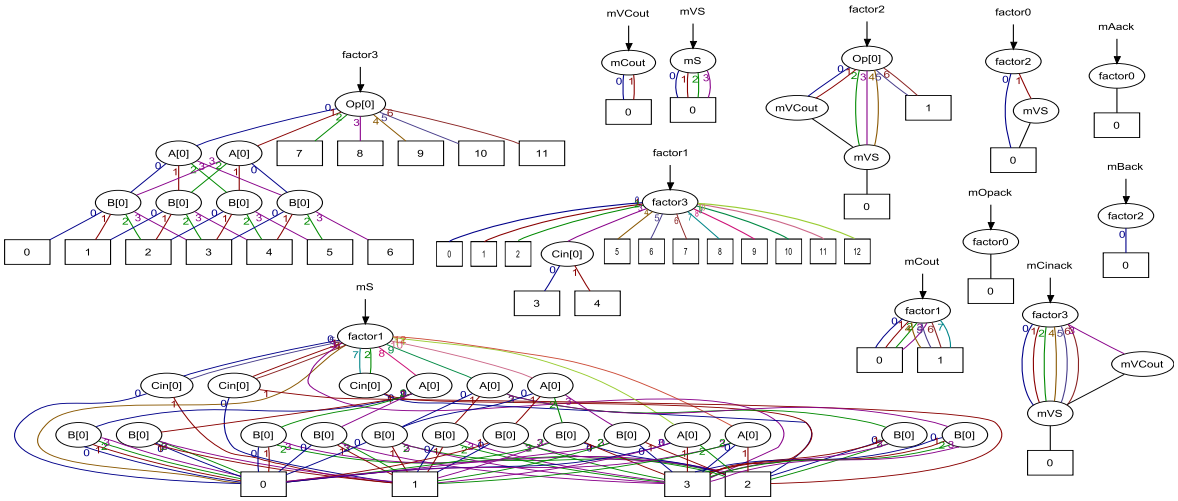

Figure 6: Result of the factorization over Figure 5

5. Basic gates synthesis from the MDDs

There are several steps to synthesize a circuit using basic two-input gates. First, a factorization is done between the different MDDs to share the common parts. Then, a reduction is applied to decrease the number of vertices in each MDD. Finally, each node of each MDD is synthesized using two-input gates.

5.1. Factorization

The factorization algorithm extracts the common part of a set of MDDs as an internal MDD, as illustrated in Figure 7.

To preserve the QDI property, the factorization algorithm must ensure that it extracts at least one node in each path of the MDD: otherwise, the extracted

MDD could become valid but be ignored in the calculation of the circuit's outputs, remaining unacknowledged and therefore violating the QDI property. To ensure this, the algorithm only extracts common parts that include the root vertex. Since we try all possible ordering of the variables, this restriction does not limit the efficiency of the algorithm. Figure 6 shows the result of this algorithm when applied to the MDDs of Figure 5.

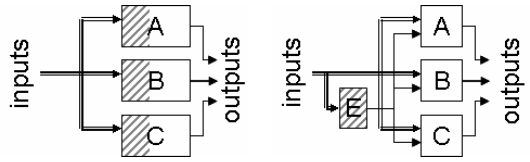

Figure 7: Before and after the factorization of a set of MDDs. E is the common part extracted from A, B and C.

5.2. Reduction

This step is similar to the reduction of BDDs: it merges the identical vertices of the MDD, which decreases their number and thus the size of the circuit. Note that this is different from factorization: the reduction acts on the structure of one MDD, whereas the factorization acts on the logical functions represented by a set of MDDs, independently of their structure.

5.3. Synthesis using basic two-input gates

To synthesize the circuit modeled by composed MDDs, each MDD is synthesized as a block of the circuit.

The algorithm is specified by the following rules:

- Each arc in a MDD corresponds to a rail in the circuit.
- Multiple arcs directed to the same vertex are grouped by an OR gate.
- A non-terminal vertex is implemented as set of two-input Muller gates that synchronize each rail of its variable with the in-going arc. The Muller gates outputs are the out-going arcs of the vertex.
- A terminal vertex with value i represents rail number i of the MDD.

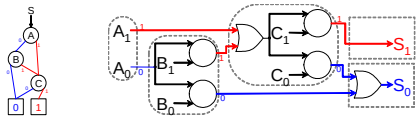

Figure 8: Example of basic two-input gates synthesis of a MDD.

Figure 9 presents the synthesized circuit from the MDDs of Figure 5.

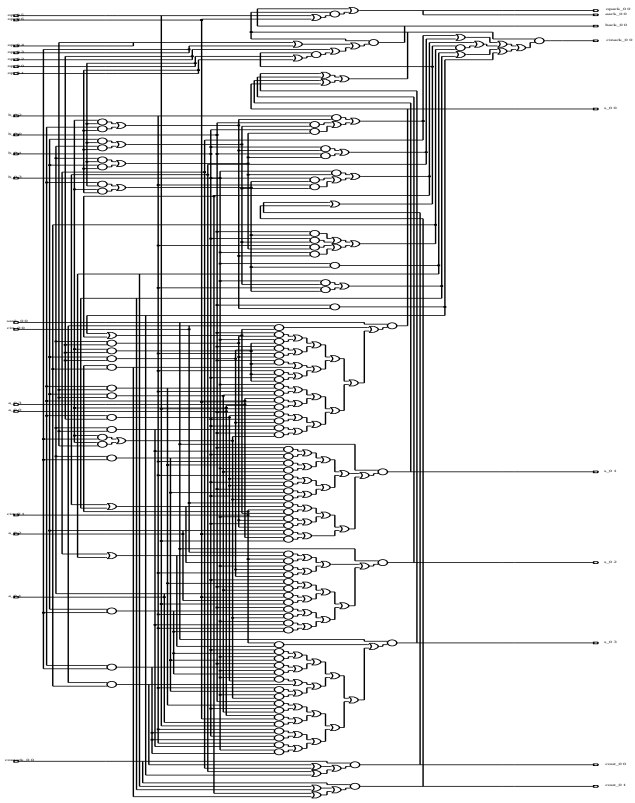

Figure 9: Basic two-input gates circuit synthesized from the MDDs of Figure 5.

6. Technology mapping

We first present a library of asynchronous standard cells we have developed and called TAL. Then, we give different results obtained by using this library in the design of the digit-slice radix 4 ALU, instead of the ST standard library. Finally we compare our asynchronous circuit to a synchronous equivalent circuit.

6.1. TAL library

The TAL library has been developed to design asynchronous circuits with the aim to reduce their area, consumption and increase their speed [14]. This library contains about 160 cells (representing 42 functionalities), and has been designed with the 130nm technology of STMicroelectronics. The main functionalities of the library are useful asynchronous functions as Muller gate, Half-Buffers, Mutex and complex gates as Muller-Or, Muller-And, …

To clarify what gains should be attributed to a dedicated asynchronous library, we can view in Table 1 the comparison, between basic cells of the TAL library and their standard cells equivalent, in terms of number of transistors and area. For example, the Muller gate presented in 3.4 is build with 9 transistors in the TAL library (for a Muller gate with 2 inputs). With standard cells we have to use an optimized AO222 gate with a loop as described in Figure 10, made of 14 transistors, to find the functionality of a Muller gate.

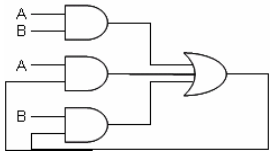

Figure 10 : Muller Gate in standard cells

Function	TAL Lib Nb of transistors/ Area (µm²)	Std cells Nb of transistors/ Area (µm²)	Gain (area)
Muller 2	9 tr. / 14,12	14 tr. / 20,17	30 %
Muller 4	13 tr. / 18,15	42 tr. / 60,51	70 %
Half-Buffer	28 tr. / 40,34	44 tr. / 62,53	35 %

Table 1 : Differences between TAL and Std cells implementations of basic functions.

The average gain in term of area for all the TAL library compared to the standard ST library is around 35%.

6.2. Technology Mapping algorithms

The main difficulty before mapping a library on asynchronous circuits is to decompose them and ensure to keep their property of quasi delay insensitivity.

For example, it's difficult to decompose a Muller gate with 3 inputs in 2 Muller gates with 2 inputs without introducing a hazard. This decomposition is automatic for an OR gate. This is described in Figure 11.

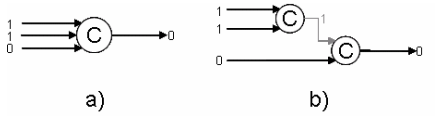

Figure 11 : Naïve Muller decomposition introduces hazard

In case a), the three inputs of the Muller gate are different and the output keeps its value 0. After the decomposition (b), the first Muller gate output switches while the output of the second one doesn't change. Thus the output of the first Muller gate is not acknowledged causing a possible glitch in the circuit with the next set of inputs.

The synthesis method presented in 0 ensures that the circuits obtained are QDI and formed of two-input gates. Thus the decomposition phase is done and the technology mapping consists in merging gates to obtain an optimized circuit following a selected criteria (area, speed, …). Merging gates do preserve delay insensitivity.

We decide to implement known synchronous algorithms of technology mapping [15-17] and adapt them to asynchronous circuits. Some algorithms of technology mapping exist for asynchronous circuits [18-20], but the aim of these algorithms is mainly to decompose circuits without hazards, and as we have seen before, the decomposition is solved.

Moreover, technology mapping has been an important domain of research in the synchronous world and the resulting algorithms are very powerful. Thus we extend the method presented in [16] because the technology mapping algorithm presented in this paper has really great performances. Thereby we represent the input library cells as tree of OR, AND and MULLER gate and we keep the structural relationship between the library cells using lookup table. These trees are then mapped on the netlist representing the circuit with the same algorithms as for synchronous circuits.

6.3. Results

In the following section, we intend to evaluate in terms of area the gain due to the TAL library and the gain due to the technology mapping algorithms.

The circuit netlist of Figure 9 comprises 95 OR gates and 107 MULLER gates. The Table 2 compares the number of transistors and the area of the circuit, before place and route, using the TAL library or the ST standard library.

Table 2 : Circuits with TAL or ST standard cells

	TAL library	Standard ST cells
Nb of transistors	1533	2068
Area (µm²) (before placement and routage)	2469	3116,36

We can conclude out of this figure that without any optimization of the netlist, if we only use TAL cells instead of the standard cells to build Muller gates, the number of transistors decreases by 35% and the area of the circuit decreases by 21%.

Now we want to evaluate the gain brought by the technology mapping algorithms on the netlist of the digit-slice radix 4 ALU. We can view results of algorithms in the Table 3. During the mapping phase, only complex gates of the TAL library are used as Muller-Or22, Muller-Or21. OR2 gates are also merged in OR3 and OR4 gates.

Table 3 : Results of technology mapping algorithms

	Native TAL netlist	Optimized TAL netlist
Nb of transistors	1533	1034
Area (µm²) (before placement and routage)	2469	1401,95

We can notice a decrease of 32% of the number of transistors, and a decrease of 43% of the area of the circuit compared to the same circuit netlist using the TAL library without technology mapping algorithm applied. We thus note a decrease of around 50% of the

number of transistors and area compared to the initial netlist using the ST standard cells library.

Another interesting point is to compare these circuit characteristics with an equivalent synchronous digit-slice radix 4 ALU. The asynchronous circuits remain bigger than their synchronous equivalent because of the delay insensitive code and the local controls of the circuit. However our goal is to reduce this difference as much as possible by applying aggressive technology mapping algorithms on the circuit and by using cells library specially designed for asynchronous circuit.

We describe the digit-slice radix 4 ALU using the VHDL language. As we want to compare our version to a synchronous circuit, we add a clock in the description. In fact, the outputs are memorized in the asynchronous circuit with the Muller gate. In the synchronous version, we have to add registers on each output, to achieve this memorization.
To synthesize this circuit, we used Design Analyser from Synopsys and the ST standard cells library. Table 4 shows the results.

Table 4 : Comparison with the equivalent synchronous circuit

	Optimized TAL netlist	**Synchronous netlist**
Nb of transistors	1034	386
Area (μm^2) (before placement and routage)	1401,95	476, 06

We can conclude that the synchronous circuit is less than 2,9 times smaller, and contains 2.7 times less transistors than the asynchronous one.

7. Conclusion

This paper presents a general method to model and synthesize asynchronous optimized QDI circuits. The method allows synthesizing circuits using multi-rail logic and maps them on to single output standard cells. Direct and reverse (acknowledge) paths are automatically and jointly synthesized. A first netlist of the circuit, containing only two-input gates is generated. Technology mapping is then applied targeting a dedicated asynchronous library to optimize the circuit area. Others criteria of optimization could be selected as well but the paper focuses on area which is one of the must important challenge.

The method based on Multi-valued Decision Diagrams, is illustrated on a digit-slice radix 4 ALU. We present different versions of the same circuit to evaluate the gain introduced by the asynchronous library and by the technology mapping algorithm. The last results show that our circuit is still 2.9 times larger than the synchronous one.

Future work will be focused on improving the methodology by working in two directions: logic synthesis and complex cells specification.

8. References

1. Renaudin, M., *Asynchronous circuits and systems: a promising design alternative.* Microelectronic Engineering, 2000. **54**(1-2): p. 133 - 149.
2. Dinh Duc, A.V., L. Fesquet, and M. Renaudin. *Synthesis of QDI Asynchronous Circuits from DTL-style Petri Net*. in *11th IEEE/ACM International Workshop on Logic & Synthesis*. 2002. New Orleans, Louisiana.
3. Dinh Duc, A.V., et al. *TAST CAD Tools*. in *ACiD-WG workshop*. 2002. Munich, Germany.
4. Martin, A.J., *The Limitations to Delay-Insensitivity in Asynchronous Circuits*, in *Advanced Research in VLSI*, W.J. Dally, Editor. 1990, MIT Press. p. 263-278.
5. Manohar, R., T.K. Lee, and A.J. Martin. *Projection: A Synthesis Technique for Concurrent Systems.* in *The 5th IEEE International Symposium on Asynchronous Circuits and Systems*. 1999.
6. Toms, W.B. *QDI Implementation of Boolean Graphs.* in *14th UK Asynchronous Forum*. 2003.
7. Burns, S.M., *General Condition for the Decomposition of State Holding Elements*, in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*. 1996, IEEE Computer Society Press.
8. Lemberski, I. and M.B. Josephs. *Optimal Two-Level Delay-Insensitive Implementation of Logic Functions*. in *PATMOS*. 2002. Spain.
9. Nielsen, C.D. *Evaluation of Function Blocks for Asynchronous Design*. in *eurodac*. 1994: icsp.
10. Martin, A.J., *The Limitations to Delay-Insensitivity in Asynchronous Circuits*, in *Advanced Research in VLSI*, W.J. Dally, Editor. 1990, MIT Press. p. 263--278.
11. Bregier, V., et al. *Modeling and Synthesis of multi-rail multi-protocol QDI circuits.* in *International Workshop on Logic Synthesis*. 2004.
12. Kam, T., et al., *Multi-valued decision diagrams: Theory and applications.* International Journal on Multiple-Valued Logic, 1998. **4**(1-2): p. 9-24.
13. Dreschler, R. and B. Becker, *Binary Decision Diagrams, Theory and Implementation.* Kluwer Academic Publishers ed. 1998: Kluwer Academic Publishers.
14. Maurine, P., et al. *Static Implementation of QDI Asynchronous Primitives*. in *PATMOS: 13th International Workshop on Power and Timing Modeling, Optimization and Simulation*. 2003.
15. Keutzer, K. *DAGON: technology binding and local optimization by DAG matching*. in *Proceedings of the 24th ACM/IEEE conference on Design automation*. 1987. Miami Beach, Florida, United States.
16. Zhao, M. and S.S. Sapatnekar. *A new structural pattern matching algorithm for technology mapping*. in *The 38th Conference on Design Automation*. 2001. Las Vegas, Nevada, United States.
17. Matsunaga, Y. *On Accelerating Pattern Matching for Technology Mapping*. in *International Conference on Computer Aided Design*. 1998. San Jose, California, United States.
18. Cortadella, J., et al. *Decomposition and technology mapping of speed-independent circuits using Boolean relations*. in *Proc. International Conf. Computer-Aided Design (ICCAD)*. 1997.
19. Myers, C.J., P.A. Beerel, and T.H.-Y. Meng, *Technology Mapping of Timed Circuits*, in *Asynchronous Design Methodologies*. 1995, Elsevier Science Publishers. p. 138-147.
20. Siegel, P.S.K., Automatic Technology Mapping for Asynchronous Designs. 1995, Stanford University.