

A High Performance Synthesizable Unsymmetrical Reconfigurable Fabric For Heterogeneous Finite State Machines

Zhenyu Liu¹, Tughrul Arslan^{1,2}, Sami Khawam¹, Iain.Lindsay

¹ School of Engineering and Electronic, The University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh EH9 3JL, UK.

² Institute for System Level Integration, Livingston, EH54 7EG, UK
{zhenyu.liu, Tughrul.Arslan, S.Khawam, Iain.Lindsay} @ee.ed.ac.uk

Abstract - The use of synthesizable reconfigurable cores in system on chip (SoC) designs is increasingly becoming a trend. Such domain-special cores are being used for their flexibility, powerful function and low power consumption. A reconfigurable Finite State Machine (FSM) is constantly required for the purpose of control in any reconfigurable SoC. This paper presents a novel unbalanced unsymmetrical reconfigurable architecture for generic FSM; Compared with commercial FPGA devices, the new architecture results in area reduction of 43% and power consumption decrease of 82%.

I Introduction

Today, computational requirements have risen faster than the improvement in silicon technology. Custom reconfigurable SoC technology aims to satisfy the simultaneous demand for flexibility and efficiency. The Reconfigurable-System-on-Chip concept is rapidly becoming a reality and is mainly derived from the developed reconfigurable technologies which have been quickly evolving in recent years. In contrast to generic SoC, the reconfigurable array core in a Reconfigurable SoC performs the critical computation to meet the overall application's constrains. A typical architecture for Reconfigurable System-on-Chip is shown in Fig. 1. The architecture achieves higher performance and its reconfigurability compared to platform FPGAs permits adaptation of the hardware for different computations.

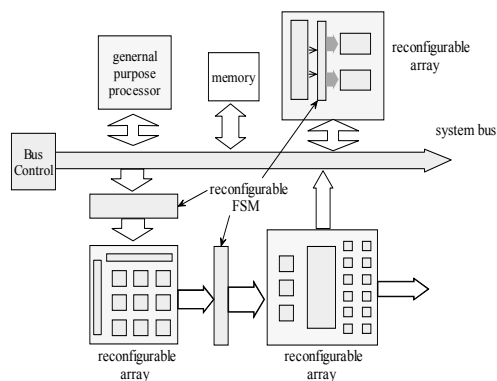


Fig. 1. The architecture of reconfigurable system-on-chip

The control circuit is one of the fundamental parts in any

system. For example, every reconfigurable datapath costumed for given application would require an associated reconfigurable FSM to sequence its operation. Reconfigurable FSMs are found at various locations in the reconfigurable SoC as shown in Fig. 1. FSM sub-modules can be located between system bus and reconfigurable arrays, between the different arrays and inside the arrays themselves. Without the special control units (such as FSM), the system microprocessor must deal with some additional control tasks for the array. This maybe at a point when the processor is extremely busy with the current computation. Therefore a reconfigurable FSM can be utilized to free up the workload from the processor.

For many practical applications, digital systems are heterogeneous, so their specifications may change continuously. The main way to meet the requirement of flexibility is to adopt a FPGA style device. Based on the fine grain framework and look up table (LUT), FPGA devices can implement any given logic or arithmetic operation but at the cost of tens times the area and power consumption of an ASIC. The FPGA device is also a special-define device with the fixed number of gates and area. Even if only a small special function is needed, the whole chip must be used in the system which may contain more gates than really needed. New hardware architecture must be developed to cope with this challenge.

In this paper, we present a novel reconfigurable FSM architecture which can implement any logic operation with less area occupied and power consumption. Our new architecture takes the advantages of both fine grain framework and coarse grain framework which makes the new architecture exhibit the characteristics of extra flexibility and highly efficient usage of area and power. The reconfigurable FSM architecture provides the ability to make post-fabrication changes after it has been embedded into a large IC or microprocessor. The characteristics of the architecture in that it can be pre-synthesized, pre-verified and post-changed could shorten the period of system development, reduce the risk of development and improve the flexibility of the system.

The rest of the paper is organized as follows. In section 2 we review related work in the literature. In section 3, we provide a brief describe about the basic definitions of FSM and the decomposition of FSM. Our architecture is described in details in section 4 which includes the impact of FPGA, the overview of reconfigurable FSM architecture, the

functional sections in the architecture, the logic and sequential block, the construction of PTB, interconnection network and data path. Finally, the experimental results and summary are presented.

II. Related Work

Over the years, significant research work has been carried out on FSM algorithms and their implementation. A great part of these works focus on the algorithm which include the decomposition of FSM, optimization of FSM and self-reconfigurable FSM.

Lees et.al. [1] research presents a method of runtime configuration scheduling for the implementation of hierarchical FSM with synchronous data flow model. The method is applied to a reconfigurable design of MPEG4 natural video decoder. In Yevtushenko et.al. [2] paper, an algorithm is given to resolve the problem of reducing component machines of the synchronous FSM composition. The problem can be formulated as the problem of FSM equation solved. The algorithm is more effective to solve not only a single equation but also a system of equations. A novel concept is introduced and implemented in paper [3]. The concept of self-reconfigurable finite state machines as a formal model used to describe state-machines implemented in hardware and it has the ability to be reconfigured during operation.

III. Background

A. Definition and Category of FSM

A Finite State Machine is a 6-tuple (I, O, S, R, T, S_0) :

- I is a finite set of input of the FSM, which either are symbolic or are represented as a value of binary vector of its signals.

- O is a finite set of output of the FSM, which either are symbolic or are represented as a value of binary vector of its signals.

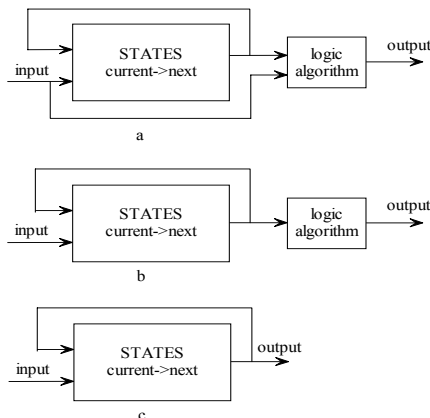


Fig. 2. Three kinds of FSM

- S is a finite set of internal states of the FSM. It consists

- of two parts: S_c and S_n . Two symbols present current states and next states respectively.

- R (i, s) is a relation from the (input, current states) pairs to the next states (i.e., $R \subseteq I \times S_c \times S_n$)

- T (i, s) is a relation from the (input, current states) pairs to the output (i.e., $T \subseteq I \times S_c \times O$).

- S_0 is an initial (or reset) state.

From the output function point of view, there are three cases found in the practical implementation, shown in Fig. 2. If the outputs of FSM are always associated with the inputs, the FSM is a Mealy machine, shown in Fig. 2a. If the outputs are influenced directly by the internal states of FSM, inputs affect the outputs only through the states, the FSM is a Moore machine, see Fig. 2b. For the third case, the internal states of FSM outputs straightforward as the result of FSM.

B. Decomposition of FSM

The decomposition of finite state machines is adopted to improve the hardware efficiency in the complex cases. The initial work date back to 1960 by Hartmanis [4] and has been further developed by several authors. The structure of the resulting finite state machine is shown in Fig. 3. The inputs of each sub-machine are not only the primary inputs and its own last state, but also the state variables from the sub-machine after decomposition.

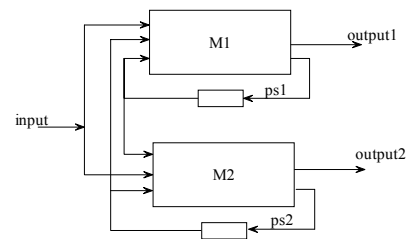


Fig. 3. Generic decomposition

To illustrate the procedure of FSM's decomposition, consider the state transition graph of a simple FSM shown in Fig. 4. It is assumed that a desired decomposition of the corresponding FSM is given that state A and state B belong to sub-machine T2 and all the other states belong to sub-machine T1. So the original FSM is divided into two smaller sub-FSMs which are depicted with two state transition graphs (STGs).

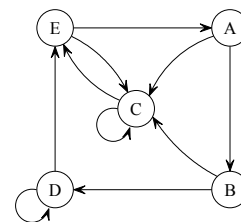


Fig. 4. State transition graph of FSM

As the one result of the decomposition, a RESET state is

added in both sub-FSMs labeled with alphabet S. The transitions between the states except the state S in T1 and T2 are same with the original FSM without transformation. As shown in Fig. 5, the number of inputs to each sub-FSM is also changed for the reason of the additional RESET state. In each sub-FSM, the extra inputs from another sub-FSM are required for the purpose of communication between two sub-FSMs, so do for the extra outputs.

Following this way, we can decompose the whole original FSM into several smaller ones which only have very limited states with a closed circle between different states. However it must be emphasized that too many sub-circles will make the implementation more complex. So we usually adopt 2 sub-circles, 3 sub-circles at the most, in the reality case.

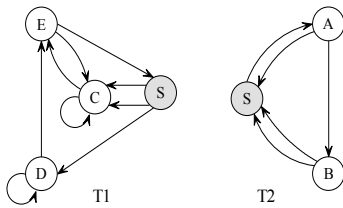


Fig. 5. State transition graph after decomposition

IV. Architecture of Reconfigurable FSM

A. Impact of FPGA Architecture

The generic FPGA's architecture is shown in Fig. 6.

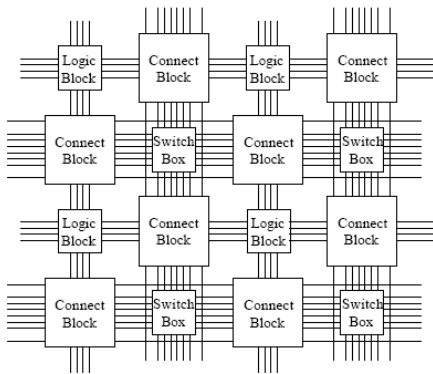


Fig. 6. The architecture of conventional FPGA

The construction of the whole reconfigurable architecture consists of functional module (Logic Block) and routing source (routing channels) which is made up of switch boxes and connect blocks. Both parts in the architecture are mapped in a balanced and symmetrical way. Functional modules are equably arranged in the routing network; each module is surrounded by the same amount of routing source.

The devices implement expected routing by a relatively smooth sea of routing resources which make the communication between the rows and columns of logic blocks. The input signals and output signals reach the routing line through the connect blocks. The long global

routing wire and short local routing wire across the connect blocks are also connected with all the input/output signals. [5]

Obviously, the FPGA device's reconfigurable ability is benefited from their powerful interconnection network. But there is also an area penalty to be paid for their flexibility. We give an example using 4-bit full directions switch box in which all the ports in each sides are bidirection and the signal can enter the box through discretionary port and reach any port for output. The area of such switch box is $1433.025\mu\text{m}^2$ based on the UMC $0.18\mu\text{m}$ CMOS technology library whereas it is only $16.262\mu\text{m}^2$ for a AND or OR gate targeted on the same library. The data above predicate that a 4-bits full directions switch box equivalent to 88 AND or OR gates, this amount of gates are enough to implement a small size FSM.

B. Reconfigurable FSM Architecture overview

Unlike the architecture of an FPGA device whose logic blocks and routing network are arranged in a balanced and symmetrical method, our reconfigurable FSM architecture map their computing modules and routing sources in an unbalanced and unsymmetrical mode. More details are given in the following sub-sections.

The architecture is a hierarchical system with different functional modules in the corresponding levels. These are the sequential section and the logic section, sequential blocks and logic blocks, adder/subtractor and product term block (PTB). Each functional module is made up of the sub-modules. The architecture of these functional modules and their interconnection network will be given in detail in the following sub-sections.

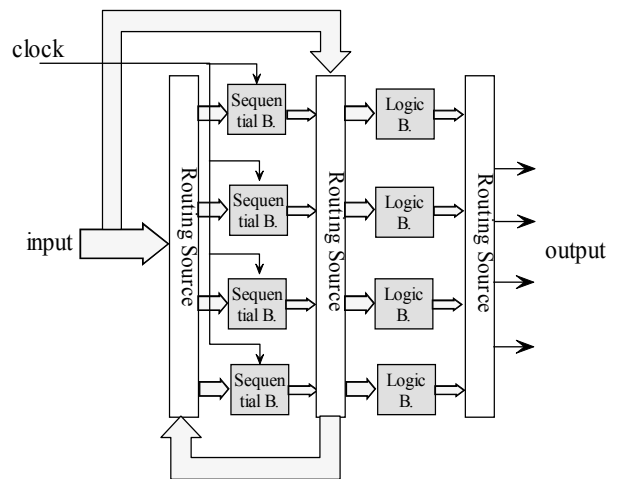


Fig. 7 The architecture of system

C. Functional Sections in the Architecture

FSM are usually used to describe the behavior of digital circuits that transform sequences over one (input) alphabet

into sequences over another (output) alphabet. So we divided an FSM into two functional sections: logic function section and sequential function section. The architecture of the system is shown in Fig. 7. Sequential section's works depend on the clock of FSM and it has the different current state and next state in different time. Logic section hasn't any relationship with the clock and is purely combinational. It is decided only by the Boolean equation given. According to the two functional sections of FSM, there are two corresponding sections in the architecture of the whole system: sequential section and logic section, shown in Fig. 7. The sequential block (labeled with Sequential B.) and logic block (Logic B.) are the functional units in their respective sections.

The input signals which are up to 8 in the design can reach both sections through the routing source. The outputs of sequential block can not only arrive at the input ports of logic block but also feedback to the input ports of the sequential block. The number of logic blocks with 8 max in the current design depends on the number of FSM's outputs. The number of sequential blocks is up to 8 in our design which can implement a generic FSM with the maximum states for 256 (2^8). By the benefit of adder/subtractor inside the sequential block, the architecture can also implement maximum 65536 (2^{16}) states FSM which state transition graph is a close circle.

D. Architecture of Logic Block

The logic block which consists of product term-based block (PTB) is the functional module in the logic section. The architecture of logic block is shown in Fig. 8. The function of logic block is to realize some combinational Boolean functions. PTB acts as the basic computing unit to realize some basic Boolean computing.

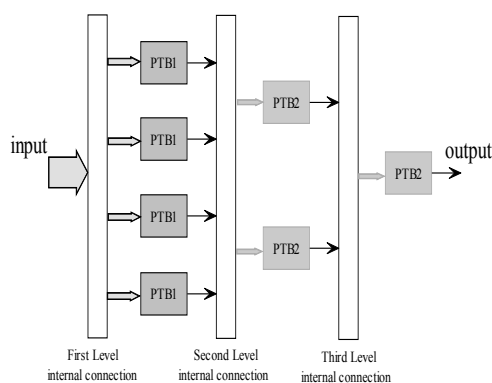


Fig. 8. The architecture of logic module

The number of product term-based block in logic block determines the efficiency of the block. A single product term-based will lead to the complexity of mapping and routing. A big number of product term-based block in the logic block will reduce the burden of interconnect switch matrix, but it will also increase the size of the total area. A

balanced is needed to resolve this question.

The PTBs in the construction are put in several levels and a triangular architecture is adopted in arranging PTBs which the number of PTBs in each level is half of its previous level. The outputs of one level can only reach the inputs of next level's PTBs.

The first reason why we call the architecture an unbalanced one is to select different type PTB labeled with PTB1 and PTB2 for the diverse levels. We refer to the size of a product term-based block using the tuple (i, p, o) where i is the number of input, o is the number of output, and p is the number of product terms. From the experimental data in [6], we select (8,4,2) as PTB1's parameter and (2,1,1) as PTB2's parameter. The unbalanced arranging of PTB is corresponding to the mapping of interconnection which will be given more details in sub-section G. If only the PTB1 is selected for the different level, lower PTB efficiency in the second and third level will lead the bigger whole system area. At the same way, PTB2 is as the unique type as the PTBs, will result lower switch box efficiency in the first level and get the same outcome as previous case.

E. Architecture of Sequential Block

Like the logic block, the sequential block is the functional module in the sequential section. Sequential block can implement the combination of Boolean function and transition between two different single bit states which is synchronous with the input clock.

In most cases, the number of states in an FSM is in the range of tens. For the use of special fixed functional module in our architecture, the decomposition of finite state machines is adopted. The large FSM is decomposed into several small FSMs to realize an efficient architecture.

As described in last section, a complex FSM with numbers of states can be divided into several small simple FSMs each a closed circle in STG. Therefore the sequential block in sequential section employs a reconfigurable computing module which can implement the function of adder or the function of subtractor. In order to realize bidirection chain in FSM, the configurable bit is designed to control the computing module. However, there is a limit in many cases with the use of adder/subtractor to implement the transition between the internal states in FSM. With an enlarging number of sub-circles of FSM, additional states and signals will make system hard to implement and inefficient. A logic module following with a D Flip-Flop is also employed in sequential section which can realize the transition. The purpose of using adder/subtractor is to improve the efficiency of hardware. The logic module mentioned in the previous sub-section can realize the same function as adder/subtractor with about six times larger in area.

F. Construction of PTB

We select product term-based style as the way to construct

logic module. A product term-based block consists of two sub-modules: the AND sub-module and the OR sub-module. The AND sub-module can generate a product term; The output of sub-module is used for the OR sub-module. The OR sub-module is used to "sum" all the results of AND sub-module and finally to create the desired Boolean function. It is easy for the logic computing unit to implement the AND sub-module and OR sub-module which construct of product term-based block with the corresponding reconfigurable bits. This architecture is natural if the relationship between the inputs and outputs is concluded by the truth table.

We use a two-input reconfigurable logic computing unit which can realize some basic Boolean equation to implement AND sub-module or OR sub-module. Being the base computing unit, this base computing unit is able to implement almost all possible 2-input logic functions. All the realizable logic functions with two inputs A and B are listed in Table 1.

Logic Functions with Input A and B	
1	0
A	\overline{A}
B	\overline{B}
A+B	$\overline{A+B}$
$\overline{A+B}$	$A+B$
A•B	$\overline{A•B}$
$\overline{A•B}$	$A•B$

Table 1. Logic function of basic unit

G. Interconnection Network and Data Path

The purpose of interconnection is to provide a network for data exchanging which can make it possible to form a combination of computing module for the given logic function. Good construction of the interconnection network is the key to solve the problem of area consumption. In some architecture like FPGAs, where the logic blocks are embedded in the routing network made up of connect boxes and switch boxes, the area allocated to routing source will be over 80% of the whole system.

The reason why we call our architecture unbalanced is not only for selecting different type PTB for the diverse levels but also arranging the interconnection (switch box and fixed line) in an asymmetry way.

Based on carefully analysis of the routing source area, our unbalanced architecture uses an amount of fixed connection to replace the flexible switch box in order to reduce the area. There are three internal connection levels shown in Fig. 8. In the first level internal connection, the switch box based on multiplexer technology is adopted to feed the inputs of the logic computing module. In this level, the internal connection keeps the flexibility to make sure that input signal can reach responding logic computing module. With the guarantee of the required function successfully

implemented, the redundancy connection path is reduced to save area. In the second and third level interconnection, the most part are made up of fixed connection which can greatly reduce area compared to using switch boxes.

V. Experimental Results

In order to evaluate the efficiency of our architecture, we have implemented several commonly used test cases from LGSynth93. [9] These benchmarks are adopted by the many researcher [7][8][6] who work on implementing FSMs. The FSMs in the benchmark are all Mealy machines except for Cnt32 and Cnt64 which are Moore type.

We select six test cases which are all Mealy type to be implemented with our architecture, all with different complexity, number of inputs, internal states and outputs as shown in Table 2. We describe the characteristic of FSM with the tuple (I, O, S, P) where I is the number of input, O is the number of output, S is the internal states of FSM, and P is the number of product which indicates the relationship's complexity among inputs, outputs, current states and next states of FSM.

Name	I	O	S	P
lion	2	1	4	11
dk27	1	2	7	14
dk512	1	3	15	30
s27	4	1	6	34
tav	4	4	4	49
bbara	4	2	10	60

Table 2. Test cases and their characterizations

We have implemented the test cases both with our reconfigurable architecture mapped to an ASIC technology and FPGA device to compare the area occupation and power consumption. The experimental results of area occupation with the unit of μm^2 are listed in Table 3. The synthesis tool used is Ambit BuildGates V4.0-s002 from Cadence Design Systems, Inc. The architecture is targeted on UMC 0.18 μm three-metal CMOS technology library. The synthesis and mapping tools for FPGA device is ISE V6.2i of Xilinx, Inc. The target FPGA device is Virtex-E [10] which obtains a typical look-up table and symmetric routing channel based on the switch box and connect box. The area estimation of the Virtex-E is based on the estimation of 3303 μm^2 per slice and its surrounding routing. [11]

It can be seen from the table that our reconfigurable architecture achieves between 53.3% to 34.6% improvement compared with the Virtex-E. This result proves our strategy of reducing routing network leading to significant saving in area.

We have also evaluated power consumption for both implement forms as shown in Table 4. It is well known that the difference in switching activity will greatly affect the power consumption result. The same testbench and data are

Name	FPGA		Our Re. Architecture	Improvement
	Slices	Area		
lion	4	13212	7496.59	43.2%
dk27	5	16515	10809.85	34.6%
dk512	13	42939	22534.51	47.6%
s27	10	33030	15420.13	53.3%
tav	5	16515	10232.68	37.9%
bbara	15	49545	28640.94	42.2%

Table 3. Experimental results for area

used for both cases. The power consumption values for our reconfigurable architecture are obtained after post-routing simulation by the *Cadence Silicon Ensemble and Synopsys PrimePower*. In the case of the Virtex-E FPGA, the power consumption data is obtained from XPower V6.2i which is specially designed for evaluating the power consumption of FPGA devices. The power consumption of FPGA device consists of two parts: internal power consumption which includes clocks power, signals power, inputs power and logic power and outputs power. The FPGA power consumption listed in the table only contain the internal power.

Name	FPGA Virtex-E (mW)	Our Re. Architecture (mW)	Improvement
lion	0.87	0.176	79.8%
dk27	1.94	0.227	88.3%
dk512	1.29	0.31	76.0%
s27	1.23	0.234	81.0%
tav	1.17	0.249	78.7%
bbara	2.25	0.297	86.8%

Table 4. Experimental results for power consumption

From the Table 4, the power consumption of our architecture achieves between 76.0% to 88.3% improvement compared with the FPGA device. It can be concluded that our architecture achieve better performance in power consumption than area.

VI. Summary

A novel reconfigurable low-power architecture for control circuit was introduced in this paper. The architecture doesn't adopt the popular internal connection network in which the functional modules are embedded. Taking the advantage of the unbalanced construction, the architecture can implement generic FSMs with low area occupation and power consumption when compared with commercial reconfigurable FPGA devices.

In this paper, six test cases from the widely adopted FSM benchmark set were implemented using both our

architecture and a FPGA device. It was demonstrated that our design can obtain approximately 82% reduction in power consumption and a decrease of 43% in area occupation when implementing the same circuit on a commercial FPGA. These figures show that the proposed reconfigurable architecture provides an efficient hardware platform for implementation generic FSMs in various power sensitive designs. The flexibility of the architecture makes it convenient being embedding in various systems such as mobile devices, portable players, etc.

References

- [1] Sunghyun Lee, Sungjoo Yoo, Kiyoun Choi, "Reconfigurable SoC design with hierarchical FSM and synchronous dataflow model," *Hardware/Software Codesign, CODES 2002*. pp. 199-204, 2002.
- [2] Yevtushenko. N, Zharikova, S, Vetrova. M, "Multi component digital circuit optimization by solving FSM equations," *Digital System Design, 2003*. pp. 62-68, 2003.
- [3] Koster, M., Teich, J., "(Self-)reconfigurable finite state machines: theory and implementation," *DATE 2002*. pp. 559-566, 2002.
- [4] J. Hartmanis, "Symbolic analysis of a decomposition of information processing," *Information Control*, Vol. 3, June, pp. 154-178, 1960
- [5] Katherine Compton, Scott Hauck, "Reconfigurable Computing: A Survey of Systems and Software," *ACM Computing Surveys*, Vol. 34, No. 2, June, 2002.
- [6] Yan, A., Wilton, S.J.E., "Product-term based synthesizable embedded programmable logic cores," *Field-Programmable Technology, 2003*. pp. 162-169, 2003.
- [7] Kuusilinna, K., Lahtinen, V., Hamalainen, T., Saarinen, J., "Finite state machine encoding for VHDL synthesis," *Computers and Digital Techniques, IEE Proceedings*, Vol. 148, Issue 1, pp. 23-30, Jan, 2001.
- [8] Yevtushenko, N., Zharikova, S., Vetrova, M., "Multi component digital circuit optimization by solving FSM equations," *Digital System Design, 2003*. pp. 62-68, 2003.
- [9] Benchmarks:
<http://www.cbl.ncsu.edu/benchmarks/LGSynth93>
- [10] Xilinx, *Programmable Logic Data Book*, Xilinx Inc., 2001
- [11] S. Khawam, T. Arslan, F. Westall, "Synthesizable reconfigurable array targeting distributed arithmetic for system-on-chip applications," *Parallel and Distributed Processing Symposium, 2004* pp. 150, 2004