Interconnect Synthesis for Reconfigurable Multi-FPGA Architectures *

Vinoo Srinivasan, Shankar Radhakrishnan, Ranga Vemuri, and Jeff Walrath
E-mail: {vsriniva, sradhakr, ranga, jwalrath}@ececs.uc.edu
DDEL, Department of ECECS, University of Cincinnati, OH 45221.

Abstract. *Most reconfigurable multi-FPGA architectures have a programmable interconnection network that can be reconfigured to implement different interconnection patterns between the FPGAs and memory devices on the board. Partitioning tools for such architectures must produce the necessary pin-assignments and interconnect configuration stream that correctly implement the partitioned design. We call this process Interconnect Synthesis for reconfigurable architectures.*

The primary contribution of this paper is a interconnection synthesis technique that is independent of the reconfigurable interconnection architecture. We have developed an automatic interconnect synthesis tool that is based on Boolean satisfiability. The target interconnection architecture is modeled in an architecture specification language. The desired interconnections among the FPGAs are specified as in the form of a netlist. The tool automatically generates the pin-assignments and the required values for the configuration-inputs in the architecture specification. We modeled several reconfigurable architectures and performed interconnect synthesis for varying number of desired nets. We provide experimental results that demonstrate the effectiveness of the interconnection synthesis technique.

1 Introduction

Modern multi-FPGA (Field Programmable Gate Array) reconfigurable boards host several FPGAs and memory devices that communicate through both dedicated and programmable interconnections [1–3]. The presence of programmable interconnections on the boards highly increases the utility of the board. However, the interconnection architectures largely vary from one board to another. This diversity can be handled by existing partitioning and synthesis tools, only with a considerable effort. More importantly, it is usually not straight forward to migrate between reconfigurable boards. The effort required in modifying existing CAD tools to handle a new RC (Reconfigurable Computer) interconnect architecture, is often comparable to that of developing a new CAD algorithm specific to that interconnect architecture.

Partitioning tools for RC architectures, in addition to partitioning computational tasks and logical memory segments, must also generate an interconnect configuration that establishes an appropriate connectivity across partitions. Typically, in programmable interconnection architectures, the pins of an FPGA cannot be treated as identical. Pin assignment plays an important role during

* This work is supported in part by the US Air Force, Research Laboratory, WPAFB, contract #F33615-97-C-1043.

partitioning for RC architectures. The functionality of the programmable interconnection network is a direct consequence of pin assignment. We define the *interconnect synthesis* problem for RC architectures to be the unified problem of generating a pin assignment and synthesizing the appropriate configuration stream for the programmable interconnection network, such that the interconnection requirement is met.

In this paper we present an interconnection synthesis technique that is independent of the target architecture. Interconnection synthesis is modeled as a Boolean satisfiability problem. The Boolean function to be satisfied is represented as a BDD (Binary Decision Diagram) [4, 5] and operations are performed on the BDD to generate the pin assignment and the configuration stream that satisfy it. Boolean satisfiability approaches have been used earlier for the conventional bi-level channel routing for ASICs [6]. More recently, Boolean satisfiability is used to model FPGA routing [7]. The authors present a methodology to automatically generate the necessary Boolean equation that determines the routability of a given region. Although we use Boolean satisfiability techniques, we differ from the above works because modeling pin assignment and configuration stream generation for interconnection architectures is different from FPGA or ASIC routing. Secondly, we have developed an automated environment for performing interconnect synthesis. In addition, we have formulated a novel interconnect architecture specification technique which helps straight-forward retargeting of the interconnect synthesis tool to various RC architectures.

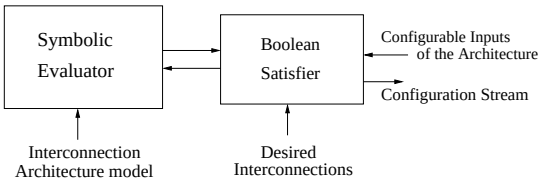

Fig. 1. Interconnect Synthesis Environment

A programmable interconnect network in a RC typically consists of a small number of such building block modules. Many instances of these building blocks are connected together, in a regular fashion, to form a reconfigurable interconnect fabric among the FPGAs and memories. In order to specify an RC interconnect architecture, we need two specification constructs: First, we should be able to specify the configuration behavior of the building blocks. Second, we should be able to specify a structural composition of instances of these building blocks. Given such specification, we need an elaboration mechanism that effectively extracts the symbolic Boolean equations that together represent the set of all allowable connections in the interconnect fabric. These equations can then be turned into the BDD representation and, in conjunction with the representation of the desired nets, be used to determine a viable configuration for the interconnect fabric if one exists.

Figure 1 shows the components that constitute the interconnect synthesis environment. We have used an RC architecture modeling language called PDL+ [8, 9] to specify interconnect architectures. ARC is an elaboration environment for PDL+ [10]. We use the symbolic evaluation capability in the ARC evalua-

tor to generate the Boolean equations representing the allowable connections in the interconnect fabric. The Boolean satisfier tool, checks for the existence of a pin assignment and a corresponding configuration stream that makes the interconnection network achieve the desired interconnections. Although a detailed discussion of the semantics of PDL+ and ARC are beyond the scope of this paper, we provide enough information to make this presentation self contained. Remainder of this paper is organized as follows. The next section presents the previous work. Section 3 elaborates the interconnection network specification methodology. Section 4 details the working of the Boolean satisfier and its integration with the symbolic evaluator. Section 5, presents a technique for Boolean formulation of the pin assignment problem. Results of interconnect synthesis are presented in Section 6. Conclusions are drawn in the final section.

2 Previous Work

The problem of pin-assignment and inter-FPGA routing, in the presence of interconnection networks, has been investigated before. Hauck and Borriello [11] present a force-directed pin-assignment technique for multi-FPGA systems with fixed routing structure. The pin-assignment and placement of logic in the individual FPGAs are performed simultaneously to achieve optimal routing results. Mak and Wong [12] present a optimal board-level routing algorithm for emulation systems with multi-FPGAs. The interconnection architecture considered is a partial crossbar. The authors present an optimal $O(n^2)$ algorithm for routing two-terminal nets. However, they do not deal with multi-terminal nets, but prove it NP-complete.

Selvidge et al. [13] present TIERS, a topology independent pipelined routing algorithm for multi-FPGA systems. The desired interconnections and target interconnection topology are both represented as graphs and the tool establishes the necessary routing. The interesting feature of TIERS is that it time-multiplexes the interconnection resources thus increasing its utility. However the limitation is that the interconnection topology has only direct two-terminal nets between FPGAs. Topologies for programmable interconnections and multi-way connections are not considered.

Kim and Shin [14] studied the performance and efficiency of various interconnection architectures shuch as, direct interconnections, K-partite networks and partial cross bars. They present a performance driven partitioning algorithm where performance is based on the number of *hops* (FPGA boundary crossing) of a net. The interconnection architecture is fixed before partitioning and cannot be programmed according to the cut-set requirements. Multi-terminal connections are broken into two-terminal nets.

Khalid and Rose [15] present a new interconnection architecture called the HCGP (Hybrid Complete-Graph Partial-Crossbar). They show that HCGP is better than partial crossbar architectures. They present a routing algorithm that is customized for HCGP. The router minimizes the number of *hops* and keeps it less than three.

The difference between our work and those discussed above is that the interconnection synthesis technique presented in this paper works for any generic

interconnection architecture. Any type of programmable architecture can be specified. The interconnection topology not fixed prior to partitioning. The necessary configuration information is produced as the result of interconnection synthesis. Both two-terminal and multi-terminal nets are allowed. The current model permits only two hops for routing each net.

3 Network Specification and Symbolic Evaluation

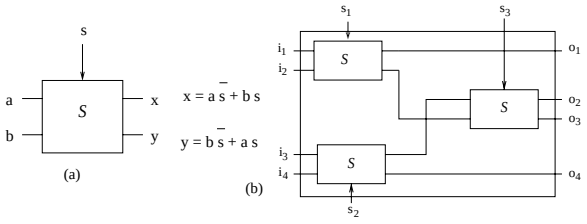

Fig. 2. Interconnection Network built using Switch-box Component

The interconnection network is a circuit that is composed of basic switch-box components of one or more types. These switch-box components determine the input-output mapping, based on certain *control* inputs. Figure 2(a) shows a simple switch-box component, $\mathcal{S}$. When the control input (s) is zero, $\mathcal{S}$ transmits a to x and b to y and when the s is one, it transmits b to x and a to y. Figure 2(b) shows the interconnection network that instantiates three $\mathcal{S}$ switch-boxes.

```
module switch_box -- defining the model switch box module
  ports
    a, b, s, x, y : ioport;    rules -- rules for outputs x and y
    x'val = (a'val and not s'val) or (b'val and s'val);
    y'val = (b'val and not s'val) or (a'val and s'val);
end module;

module interconnection_network = board -- interconnect network instance
  ports -- ports of the interconnection network
    ioports = {i1, i2, i3, i4, s1, s2, s3, o1, o2, o3, o4};
  modules
    switch_boxes = {S1, S2, S3}; -- has three switch box instances
end module;      ⋮
```

Fig. 3. Partial PDL+ Model of the Interconnection Network

Figure 3 shows a portion of the PDL+ model for the interconnection network. The design is composed of *modules* that are connected through *ports*. Ports have a single attribute called *val* of type Boolean. The module *switch_box* has *rules* to evaluate the attribute values of the output ports depending on the values of the inputs. The ARC system can perform symbolic evaluation of the various attributes in the specification model. The Boolean satisfier tool (Figure 1) communicates with the ARC runtime system through an API (Application Procedural Interface). Selective attribute values can be supplied by ARC system upon request from the Boolean satisfier.

4 Boolean Satisfiability Tool

The role of the Boolean satisfier is to generate values for the switches (configuration stream) that makes the interconnection network implement the desired interconnections. The desired interconnections are given to the Boolean satisfier in terms of a set of equivalence equations. Multi-terminal nets can also be specified using equivalence equations. The symbolic evaluator provides the boolean model of the interconnection architecture.

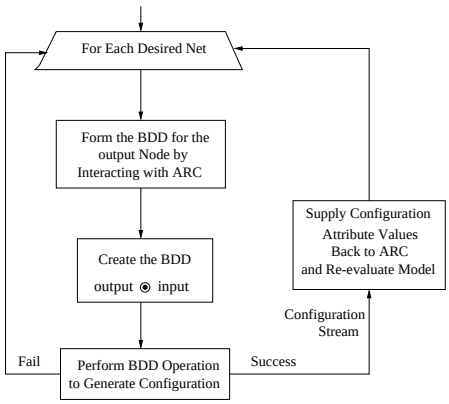

Fig. 4. Flow Diagram of the Boolean Satisfier Tool

The flow of the the Boolean satisfier tool is shown in Figure 4. For each desired net, the Boolean equation corresponding to the *val* attribute of the output port (or multiple output ports in the case of multi-terminal net) is extracted through the ARC-API. The Boolean equation is stored as a BDD. Binary Decision Diagrams (BDD) are efficient, graph-based canonical representations of Boolean functions [5, 4]. We use the BDD library developed by the model checking group at the Carnegie Mellon University [16]. In order to reduce the memory utilization and BDD generation time, the library has modes that perform dynamic reordering during BDD operations. Similarly the equation for the *val* attribute input port is extracted. BDD operations are performed to generate the values for all control inputs that makes the equivalence of the input and the output TRUE.

As shown in Figure 4, when a feasible set of switch values (configuration stream) exists, these switch values are fed back into the ARC system and the board model is reevaluated with the new switch values. For the next desired net, the new reevaluated model is used. This process continues until all desired nets have been tried. In the next section, we present a methodology to model the pin assignment as a Boolean satisfiability problem.

5 Modeling Pin Assignment as Boolean Satisfiability

Due to the presence of the interconnection network, pins in the FPGA cannot be viewed identically. For example, the Wildforce board has four FPGAs (F1 .. F4). There are 36 direct connections between FPGAs, F1 to F2, F2 to F3, and F3 to F4. There are 9 nibbles (36 bits) going from each FPGA to the interconnection network. Only *corresponding* nibbles in each FPGA can be connected, i.e., nibble number i cannot be connected to nibble j of another FPGA if $i \neq j$. Thus assignment of nodes in the design to pins plays a crucial role in interconnect synthesis. The problem of mapping design pins to physical pins in the FPGA is called the *pin assignment problem*.

Given a design partitioned across multiple FPGAs, the desired connections is only available in terms of the design pins rather than the physical FPGA pins. For this reason, pin assignment must be captured into our Boolean formulation. Since a physical pin on the FPGA can be connected to any design pin, we introduce a virtual interconnection network between the design pins and FPGA pins.

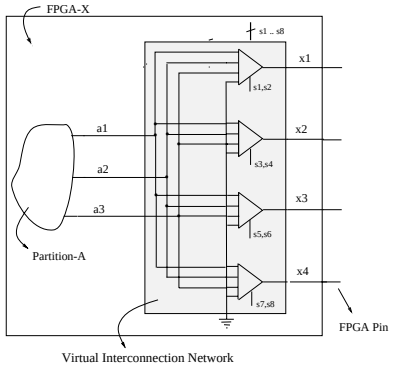

Fig. 5. Introduction of a Virtual Interconnection Network

Figure 5 shows the virtual interconnection network introduced for FPGA-X. This virtual interconnection network consists of four multiplexors. Each multiplexor can be independently controlled by two switches. Different values of switches implement different pin assignments. If the FPGA has N pins, then there are N multiplexors introduced to the design and correspondingly $O(N.logN)$ switches introduced. The design of the virtual interconnection network makes all the input lines to the virtual network symmetric. Without any loss in connectability, design pins can be mapped to any input of the virtual interconnection network. The network in Figure 5 is called "*virtual*" because this network is not synthesized as part of the design. Its presence is only to formulate the pin assignment into our Boolean satisfiability framework.

For multi-FPGA boards, pin assignment can be formulated as a Boolean satisfiability by introducing virtual interconnection networks for each FPGA. Thus the PDL+, for the interconnection architecture specification will include several virtual interconnection networks in addition to the existing interconnection network model. Now, the desired connections can be expressed in terms of design pins rather than FPGA pins. If a solution exists, the values of the switches that the Boolean satisfier tool produces will determine both pin assignment and interconnect network configuration.

6 Results

We have modeled the interconnection architecture of three boards – Wildforce [1] Gigaops, [2], and BorgII [3] and one commercial interconnection chip IQX160 [17]. We performed interconnect synthesis for various random desired nets on all four models. The Wildforce architecture has a complex interconnection network. Hence, for Wildforce, we present interconnect synthesis results with and without pin assignment. For Gigaops, BorgII, and IQX160, pin assignment exists at the time of interconnect synthesis. All experiments were conducted on an Sun Ultra Sparc 2 workstation, running at 296 MHz, with 128 MB of RAM.

The BorgII Architecture: BorgII [3] has two XC4000 FPGAs to perform logic, two for the purpose of interconnection. There are 38 programmable pins between one of the interconnect-FPGAs to both logic-FPGAs and 28 programmable pins between the other router-FPGA to the logic FPGAs. These are the only pins

through which the FPGAs communicate. The PDL+ model for BorgII has 3 modules and 132 ports.

Table 6 presents the interconnect synthesis results for the BorgII architecture. Column two gives the number of desired nets. *T_{bddgen} gives the total time spent in* BDD *generation.* This includes the time taken for symbolic evaluation and generation of BDD that represents the Boolean equation corresponding to the desired net. T_{congen} is the time spent in performing the BDD operation that generates the required configuration. T_{congen} is a very small percentage of the total time. T_{avg} is the average run-time per desired net. Table 6 shows that the total execution time (T_{bddgen} + T_{congen}) increases with the number of desired nets. Total execution time is less than 10 seconds for the largest test case, having 66 desired nets. T_{avg} is 0.55 seconds for run #1 but gradually reduces to 0.14 for run #9. The reason for this is that the symbolic evaluation time reduces as the number of nets increase. This is because, for each possible net, the configuration values are fed back into the symbolic evaluator. The reevaluated model with the updated configuration values is smaller than the original model. Hence, the BDD generation times are likely to reduce as the number of calls to the symbolic evaluator increases.

Run #	# nets	T_{bddgen} (sec)	T_{congen} (μsec)	T_{avg} (sec)
1	2	1.105	3.86	0.55
2	4	2.255	7.72	0.56
3	8	2.697	15.44	0.34
4	16	3.632	30.88	0.23
5	24	4.085	46.32	0.17
6	33	4.935	98.26	0.15
7	33	4.868	86.68	0.15
8	66	8.868	126.63	0.13
9	66	9.268	130.48	0.14

Fig.6. Results of BorgII

Run #	# nets	T_{bddgen} (sec)	T_{congen} (μsec)	T_{avg} (sec)
1	2	0.509	42.19	0.25
2	2	0.569	52.36	0.28
3	4	0.770	84.38	0.19
4	4	0.844	67.22	0.21
5	8	1.361	67.12	0.17
6	8	1.446	74.85	0.18
7	16	2.749	82.56	0.17
8	16	2.622	90.29	0.16
9	32	3.751	138.20	0.12
10	32	3.778	145.93	0.12

Fig.7. Results of GigaOps

The GigaOps Architecture: The interconnect model for the GigaOps [2] architecture has eight 16-input 16-output switches that either transmit or cut-off the input from the output. The eight switches are modeled as two rings that establish interconnections between four MODs (Gigaops' multi-FPGA card). The PDL+ model for Gigaops has 9 modules and 168 ports. Table 6 presents the interconnection synthesis results for the Gigaops board. For case 1 (two desired nets), the total execution time is in the order of half a second and for thirty two desired nets the total execution time is about 4 seconds. T_{congen} is a very small percentage of the total time. The average execution time per net reduces as the number of desired net increases. This is similar to the observation in the case of BorgII results. T_{avg} ranges between a quarter of a second to about a tenth.

The IQX160 Interconnection Chip: IQX160 is a 160 pin FPID (Field Programmable Interconnect Device) chip that implements a cross bar type of interconnection network. Any of its 160 pins can be connected to any other pin or a set of pins. The PDL+ model for IQX160 has 1 module and 160 ports. Table 6 presents the interconnection synthesis results. For the IQX chip any connection is feasible. Results show that for most cases the average run-time per net is only a fraction of second. For the cases 9 and 10, T_{avg} is less than a tenth of a second.

The Wildforce Board Without Pin Assignment: We modeled the interconnection network to be composed of 9 switch-boxes. Each switch box establishes a connectivity among wires in the corresponding nibbles of the four FPGAs. The wild force model we use is a comprehensive model that has all the dedicated interconnections, in addition to the programmable interconnection network. The PDL+ model for the board has 64 modules and 1386 ports.

Run #	# nets	T_{bddgen} (sec)	T_{congen} (μsec)	T_{avg} (sec)
1	2	1.123	15.02	0.56
2	4	1.539	40.76	0.38
3	8	2.336	44.10	0.29
4	16	5.496	79.87	0.34
5	20	5.473	85.39	0.27
6	24	5.803	115.70	0.24
7	32	5.911	130.05	0.18
8	48	5.859	179.06	0.12
9	64	5.877	188.20	0.09
10	80	5.949	323.41	0.07

Fig. 8. Results for IQX160

Run #	# nets	T_{bddgen} (sec)	T_{congen} (msec)	T_{avg} (sec)
1	4	0.894	208.91	0.22
2	8	1.519	438.92	0.19
3	12	1.624	519.03	0.14
4	16	2.536	761.03	0.16
5	20	3.462	952.72	0.17
6	24	4.113	1107.34	0.17
7	28	4.579	1207.70	0.16
8	32	5.134	1392.17	0.16
9	36	5.196	1428.18	0.14

Fig. 9. WF without Pin Assignment

Run #	# nets	T_{bddgen} (sec)	T_{congen} (msec)	T_{avg} (sec)
1	4	18.351	72.10	4.59
2	8	28.798	93.53	3.59
3	12	60.698	146.65	5.06
4	16	71.727	167.27	4.48
5	20	111.586	203.24	5.58
6	24	123.947	205.91	5.16
7	28	140.536	200.31	5.02
8	32	152.691	217.84	4.77
9	36	155.283	268.18	4.31

Fig. 10. WF with Pin Assignment

Table 6 presents the interconnection synthesis results of the Wildforce architecture. The interconnection synthesis time for the largest test case with 36 desired nets is about 5 seconds. The average synthesis time per desired net, varies between 0.22 seconds to 0.14 seconds. Unlike other boards there is not a sharp reduction in T_{avg} with the increase in number of desired nets. This is because, the Wildforce interconnection network has mutually disconnected components.

The Wildforce Board With Pin Assignment: For this experiment, a simple model of the Wildforce board is considered. We ignore direct connections between FPGAs for this experiment. Each FPGA has 9 lines connected to the interconnection network. Line i of any FPGA can only be connected to line i of any other FPGA. The PDL+ model for Wildforce architecture with modifications for pin assignment has 14 modules and 324 ports. Table 6 presents the results of interconnect synthesis with pin assignment for the Wildforce architecture. The test case 9 that has 36 desired nets, the run-time is about 155 seconds. The average time for interconnect synthesis per net is about 5 seconds. The synthesis time can be reduced with better Boolean formulations for the pin assignment.

Memory Usage: The ARC runtime system utilizes close to 12 MB of memory for all of the above experiments. Peak memory used by the Boolean satisfier for BorgII, Gigaops, IQX, and Wildforce (without pin assignment) experiments was approximately 70 KB. For Wildforce with pin assignment the memory used by the Boolean satisfier was 245 KB.

7 Conclusions

This paper presented a novel technique for interconnection synthesis for reconfigurable multi-FPGA architectures based on Boolean satisfiability. The technique works for any generic interconnection architecture. We have developed a fully automatic interconnection synthesis tool that has a symbolic evaluator and a BDD based Boolean satisfier. Currently, the interconnect synthesis tool is used as part of SPARCS [18, 19], an integrated partitioning and synthesis system for

adaptive reconfigurable computing systems. SPARCS provides automatic tools that perform temporal partitioning, spatial partitioning and high-level synthesis targeted toward dynamically reconfigurable multi-FPGA systems.

References

1. "Wildforce". *"WILDFORCE Reference Manual, Document #11849-0000"*. Annapolis Micro Systems, Inc.
2. "GigaOps". http://www.gigaops.com.
3. Pak K. Chan. "A Field-Programmable Prototyping Board: XC4000 User's Guide". Technical report, University of California, Santa Cruz, 1994.
4. R. E. Bryant. "Graph Based Algorithms for Boolean Function Manipulation". In *IEEE Trans. on Computers, C-35(8)*, pages 677–691, August 1986.
5. R. E. Bryant. "Binary Decision Diagrams and Beyond: Enabling Technologies for Formal Verification". In *Proc. ACM/IEEE ICCAD*, Nov. 1995.
6. S. Devadas. "Optimal layout via Boolean Satisfiability". In *Proc. ACM/IEEE ICCAD*, pages 294–297, 1989.
7. R. G. Wood, R. Rutenbar. "FPGA Routing and Routability Estimation via Boolean Satisfiability". In *IEEE Trans. on VLSI, vol. 6 No. 2*, pages 222–231, 1998.
8. Ranga Vemuri and Jeff Walrath. "Abstract models of reconfigurable computer architectures". In *SPIE 98*, Nov 1998.
9. Jeff Walrath and Ranga Vemuri. "A Performance Modeling and Analysis Environment for Reconfigurable Computers". In *Proc. ACM/IEEE ICCAD*, April 1998.
10. J. Walrath and R. Vemuri. "A Performance Modeling and Analysis Environment for Reconfigurable Computers". In *Proceedings of Parallel and Distributed Processing*, pages 19–24. Springer, March 1998.
11. Scott Hauck and Gaetano Borriello. "Pin Assignment for Multi-FPGA Systems". In *Proc. of FPGAs for Custom Computing Machines*, pages 11–13, April 1994.
12. Wai-Kei Mak and D.F. Wong. "On Optimal Board-Level Routing for FPGA based Logic Emulation". In *Proc. 32nd ACM/IEEE Design Automation Conference*, pages 552–556, 1995.
13. C. Selvidge, A. Agarwal, M. Dahl, J. Babb. "TIERS: Topology Independent Pipelined Routing and Scheduling for Virtual Wire Compilation". In *Proc. Int. Symp. on FPGAs*, pages 25–31, Feb. 1995.
14. Chunghee Kim and Hyunchul Shin. "A Performance-Driven Logic Emulation System: FPGA Network Design and Performance-Driven Partitioning". In *IEEE Trans. on CAD, vol. 15 No. 5*, pages 560–568, May 1996.
15. Mohammad Khalid and Jonathan Rose. "A Hybrid Complete-Graph Partial-Crossbar Routing Architecture for Multi-FPGA Systems". In *Proc. Int. Symp. on FPGAs*, pages 45–54, Feb. 1998.
16. http://www.cs.cmu.edu/ modelcheck/bdd.html.
17. *"IQX160 Starter Kit User's Guide"*. I-Cube, Inc.
18. I. Ouaiss, et.al. "An Integrated Partitioning and Synthesis System for Dynamically Reconfigurable Multi-FPGA Architectures". In *Proceedings of Parallel and Distributed Processing*, pages 31–36. Springer, March 1998.
19. S. Govindarajan, et.al. "An Effective Design Approach for Dynamically Reconfigurable Architectures". In *Int. Symposium on Field-Programmable Custom Computing Machines*, pages 312–313, April 1998.