Performance-Occupation Trade-off Examination in Custom Processor Design

Danko Ivošević, Nikolina Frid
Faculty of Electrical Engineering and Computing,
Department of Electronics, Microeletronics, Computer and Intelligent Systems
Zagreb, Croatia
danko.ivosevic@fer.hr, nikolina.frid@fer.hr

Abstract - Trade-off between execution time and resource occupation arise in all kinds of digital system designs. Here we present such relation for FPGA-based custom processor design. Usually, the optimal tradeoff is directed by device sizing on all scales of the design, but for FPGA device, as predefined hardware platform, it is more focused on comparison of existing platforms organizations. The customization of processor architecture as a point of design performance improvement is usually focused on selection of parameter set that governs the most the design characteristics.

In this paper, the focus is on processor architecture datapath with predefined design template and relationship of its structure to final FPGA implementation it maps to. For purpose of evaluation of multiple design at the same time, the appropriate software flow is applied to construct the design space based on constraining of datapath functional units operation types. The data are collected throughout the whole design flow starting from input control and data flow characteristics of the application and ending with FPGA implementation data. The analysis of the design flow showed dependence of final implementation on datapath structure and its components complexities.

I. INTRODUCTION

As Moore's law is predicting for over a 50 years now, the miniaturization of electronic devices is constant, [1]. During this period great improvement in performance, i.e. speed, is achieved also, but lately the system designers community turned to finding other possibilities for future design advances. Multiprocessor System-on-Chip (MPSoC) became a paradigm in embedded system world for distributed cooperation of existing hardware cores since increase in single core speeds appeared to be more demanding task to accomplish in short period, [2]. During advances in submicron scales the phenomena as parasitic capacitances and inductances had always been an open issue to describe and neutralize during very large scale chip integration, [3, 4]. The tradeoff between system size and performance was always actual challenge to handle during system design, independent of design methodology and level of abstraction the designer had been looking. It arises in everyday embedded system design that is trying to satisfy design functionality in all kinds of environment usually with exact sizing and performance goals. It is valid not only for isolated hardware components the system is composed off, but also for their communication and interface to human what includes software and communication protocols embodied within the system. From the point of system-level design, that is motivated by controlling the whole design process from a high level perspective and targeting specific functionalities under strict design constraints, describing the expected tradeoff means even a harder task of predicting system behavior in earlier design phases.

Our previous work coped with processor based custom design. Especially our development corresponds to hardware synthesis or high level synthesis (HLS) as a one of system level design topics, [5]. High level synthesis usually assumes specification in programming language that is not of specialized type, but familiar to more potential users. C code is usual choice because of its acceptances in computer engineering community and easiness in presentation of application control and data flow, [6, 7].

Previous work in high level synthesis field, as in custom processor design we assumed, shows this task as very acceptable when trying to minimize overall design time. Also, the fact of coping with many questions that direct the final solution appeared to be very challenging, especially in early development phases. Every such design process was, besides the specification itself, governed with some kind of directives and constraints that describe target platform and execution environment features. The common goals are design minimization and performance maximization. As these goals are in contraposition for practically every digital design, the design space exploration takes place as natural assumption and consequence of research that governs the design process.

The foundation of custom processor assumed in this paper is the *No-Instruction-Set Computer* (NISC) concept, where datapath of processor architecture is fully customizable according to needs and wishes of the corresponding toolset user, [8, 9]. Since application specification assumed C coded algorithm our and some previous work effort was on automation of datapath design out of arbitrary C code written by user, [10-12]. During corresponding research the appropriate design

This work was supported by research grant No. 036-0362980-1929 from the Ministry of Science, Education and Sports of the Republic of Croatia.

flow was developed with several transformations of input C code until final FPGA implementation is achieved. The idea behind such development was in automation of the design that shortens design cycle time and eases the overall designer's work as are the light motives of Electronic Design Automation (EDA) approach. It assumes all kinds of software support for digital system design starting from high level specification to low level phenomena description. Development of software that supports embedded system design became dominant in designer's effort during last decade, [13].

One of the usual steps towards design size optimization that was recognized in HLS, and also in our approach, was in reducing the interconnect complexity by reduction of global interconnect seen through reduction of multiplexer inputs, [14].

FPGA device is the target hardware platform for many HLS tools and for our approach. It proved itself as appropriate platform for diverse design domains and development flows, [15]. The exploration of FPGA area-delay tradeoffs is however a specific task as many families of such devices are available at the market today. In fully custom hardware design the device sizing impact is the key characteristic that affects the signal delay because of electrical effects, while FPGA is predefined device to work with. Therefore, the comparison and choice of the appropriate device is the specific task when having different architecture organization, transistor and LUT sizing, [16].

In this paper the exploration of the design space does not have an approach of devices or components sizing variations, but variations from the point of processor datapath structure when mapped to the predefined hardware platform. On the side of processor customization, predicting the best design tradeoff primarily has a meaning of selecting the set of most meaningful design parameters [17], and here we vary the functional unit types as a base of datapath design.

In Section II the background of design space exploration is given, and in Section III the motivational example for examining occupation-performance tradeoff is presented. Section IV describes the principle of forming and analyzing the solution space and the results are illustrated in Section V. Section VI gives final conclusions.

II. BACKGROUND

The foundation for design space exploration is in processor datapath design algorithm, [12]. It is situated as the central point of the design flow where C code specification is conducted to FPGA implementation.

Basically, the datapath is designed with integration of application control flow blocks into unique solution. Before that, every block of the control flow is processes by its data dependencies to form appropriate datapath that enters the final integration into unique datapath. The Control and Data Flow Graph (CDFG) is used as the intermediate representation that exposes those two aspects of every application, [18]. The datapaths that belong to control flow blocks consist of functional units performing operations of every block and registers encapsulated within register files holding variables for those operations. Therefore, the integration of blocks contributions relies on their functional units. The functional units are added to the final datapath along with register files they interconnect. During this process, multiplexers are instanced at functional units and register files input ports if more than one connection enters any of those ports.

TABLE I. COMPARISON OF DESIGNS

Characteristic		Cnt1	CntMax	NoCnt
Datapath	# Adders	1	1	14
	# Subtractors	1	1	1
	# Shifters	1	1	5
	# ANDs	1	2	4
	# ORs	1	2	7
	# XORs	1	1	3
	# NOTs	1	1	1
	# Comparators	1	1	6
	# Assigns	1	1	10
	# RFs	26	27	47
	RFs by register count	10-6-6-2-1-1	11-6-7-1-1-1	44-0-0-1-1-1
	# Muxes	35	41	61
	Mux widths	6-10-13-5-1	8-18-10-3-2	23-23-14-1-0
	# Interconnect lines	235	248	354
Implementation	DM depth	202	198	198
	CW width	217	224	242
	# Exec. cycles	3850	3770	3770
	RAM36_EXPs	11	11	13
	Slices	753	856	1097
	Slice Registers	310	343	1149
	Slice LUTs	2797	3072	3910
	Min. period	15.2 ns	14.0 ns	10.5 ns
	Max. frequency	65.9 MHz	71.7 MHz	95.0 MHz

During the final datapath integration the designer can interfere with setting the top numbers of functional units per operation types allowed in the final datapath. When new functional unit is fetched from the processed block datapath it is firstly tested if its operation type count in the final datapath exceeds the limit set by the designer. If it does not exceed this limit a new functional unit is instanced into the final datapath, otherwise not. Thus the diverse datapaths can be produced for same application what naturally reflects in different end implementations.

III. MOTIVATIONAL EXAMPLES

Comparison of diverse resulting designs is presented in Table I as motivational design example. The application beneath is SHA-1 encryption algorithm, [19, 20]. The C code of this algorithm is not very long but has quite

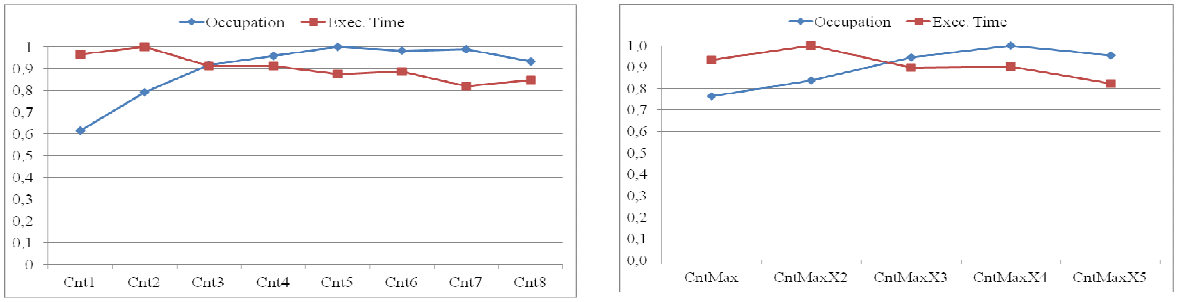
complex control flow consisting of 31 blocks. Moreover, it consists of nine different operation types: addition (ADD), subtraction (SUB), shifting (SHIFT), logical operations AND, OR, XOR and NOT, comparing (COMP) and assignment of values (ASSIGN). We examine its datapath design with some "milestones" defined by different operation type constraining styles:

- *Cnt1* as the design with limit of only one instance of every operation type.
- *CntMax* as the design with limits set to values calculated after analysis of all basic block schedules. Those values represent minimum numbers of all operation types needed to allow the full execution parallelism within blocks. For SHA-1 those values are two for logical operations AND and OR, and one for all others.
- *NoCnt* as the design without imposed limits on any operation type.

The designs in Table I are evaluated according to datapath and FPGA implementation features. The first nine rows of correspond to functional units instances of all operation types. For first two constraining cases they clearly correspond to set limits on allowed operation types, while for third they actually show the summed numbers of all blocks functional units. For last column there are no reuse of any functional unit or register, i.e. there is much redundancy in the datapath. There are much more functional units and register files and, consequently, multiplexers and interconnect lines. The first row below that describing register files count (*RFs*) shows distribution of registers across register files. Thus, there are 10 single registers (or "register file with single register"), six register files with two and three registers (each), two register files with four registers and one register file with five and six registers (each) for *Cnt1*, and even 44 single registers for *NoCnt* case. The row *Mux widths* shows the numbers of multiplexers per widths in format *Mux2-Mux4-Mux8-Mux16-Mux32*. As expected, the datapaths with less components use multiplexers with more inputs (in average) than those with more component redundancy. On the side of implementation the slices occupations of the first two are 20-30% smaller than the last one, but they are 30-45% slower than this last one.

Elaboration that is started by *Table I* data is further extended by establishing a set of constraints with clear tendency of increasing datapath sizes. Fig. 1 shows outcomes of two such constraining approaches. First, in Fig 1a is extension of *Cnt1* constraining style where *Cnt2* to *Cnt8* denote limiting the operation types to two to eight instances of every type. The second approach in Fig. 2 extends the *CntMax* constraining style where *CntMaxX2* to *CntMaxX5* denote limiting the operation types to double, triple, quadruple and quintuple numbers of every type compared to *CntMax* numbers. Graphs in Fig. 1 display existence of tendencies of rising occupations and performance looking from left to right.

Therefore, to investigate if such tendencies will be, and in what extent, recognized as common relation for our processor based design we implemented the software support that will in short time produce a range of diverse



(a) (b)

Figure 1. Correlation between occupation area and execution time under simple constraining variations

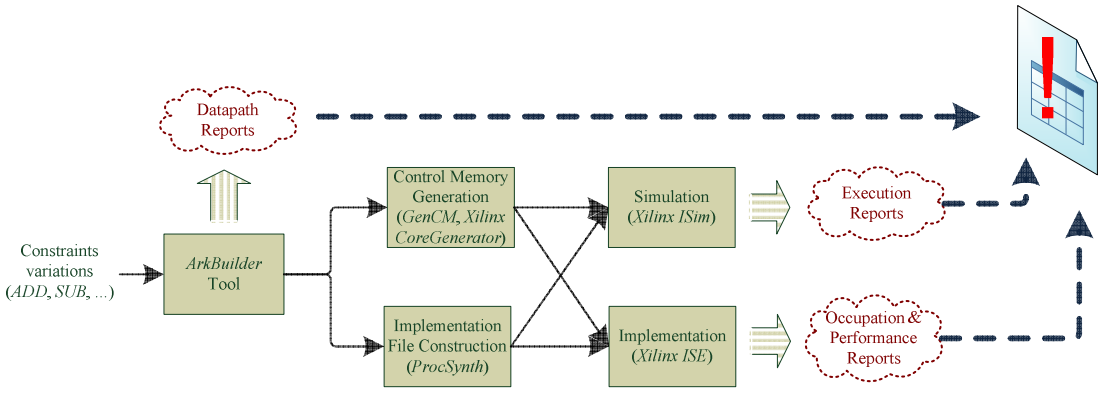


Figure 2. Design Space Exploration Flow

solutions for analysis.

IV. Design Space Exploration Principle

The design flow presented in [12] is supported with several scripts that automate generation of diverse final implementations, Fig. 1. Variations of allowed operation types numbers are generated for running datapath generation tool *ArkBuilder*. This is followed by control memory generation with *GenCM* and *Xilinx* CoreGenerator tools and Verilog implementation file construction with *ProcSynth* tool. Generated control memory cores and implementation file are then used to synthesize and simulate the final FPGA implementation. This flow is covered with several report types that serve as sources for deeper design analysis. Those are reports on:

- Datapath.
- Execution
- Occupation & Performance

Datapath reports include data on components and interconnect that form the datapath. Components data include data on component types: functional units, registers, register files, multiplexers, data memory, control unit. Functional units can be of 11 different types what is determined by operation type they perform. Register files can consist of any number of registers and registers can store any number of variables. Multiplexers can be of different widths (2, 4, 8, 16, etc.), while interconnect lines connect different components.

Execution report is generated during design simulation and has information on execution cycle count.

Occupation and performance reports assume files generated from *Xilinx ISE* implementation. Those include place and route (*.par* file), timing (*.twr* file) and power (*.pwr* file) data.

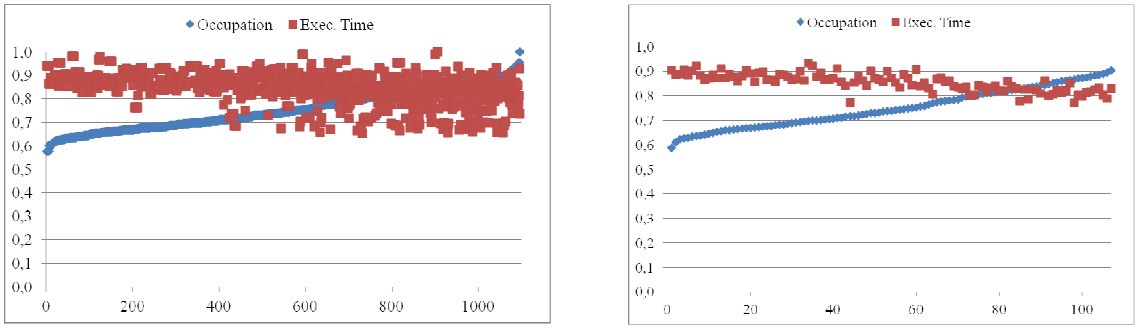
Therefore, all further analysis is based on those three report types.

V. Results

We generate three groups of cases according to different constraining levels:

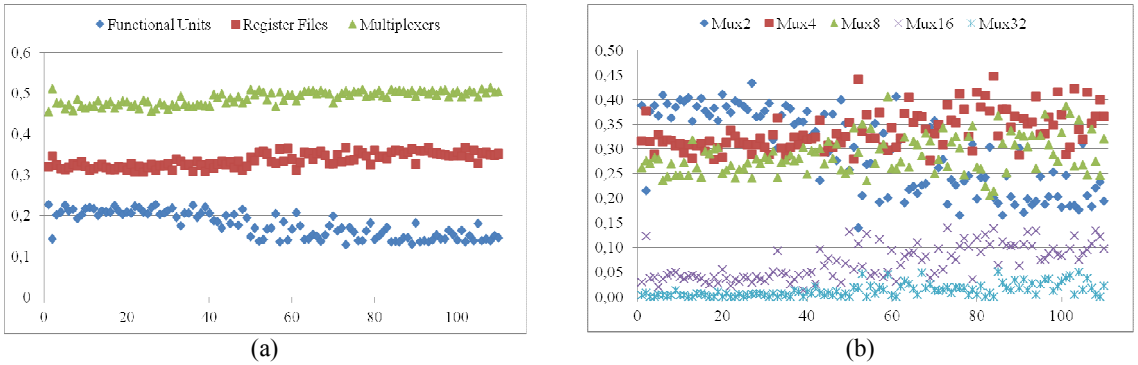
- *ordered12* resulting from variations of every operation type constraint between one and two. As there are nine different operation types, this set consists of 512 cases.
- *random14* as randomly generated 64 cases of constraints with minimum of one and maximum of four allowed instances of all operation types.
- *random17* as randomly generated 520 cases of constraints with minimum of one and maximum of seven allowed instances of all operation types.

Therefore, the complete solution set consists of 1096 datapaths and implementations, and Fig. 3a shows the relationship between occupation (in FPGA slices) and execution time for the complete set. The data are sorted by increasing occupation and show tendency of performance improvement with greater occupations, however with significant variations. This tendency is more emphasized



(a) (b)

Figure 3. Tradeoff between occupation area and execution time: (a) without grouping, and (b) when grouped by 10 constrained cases



(a) (b)

Figure 4. Dependence of execution time on: (a) component types shares (b) multiplexer widths shares

when solutions are grouped by 10 as in Fig. 3b. The same relationship between occupation and execution time when looked for constituents of the data set, i.e. separated *ordered12*, *random14* and *random17* groups, appears to differs only in steep of rising occupation points. It is steeper for cases with more relaxed constraints, i.e. *random14* and *random17* (not presented because of limited space).

Fig. 4 illustrates the solution space data grouped by 10 solutions where solutions are sorted by rising execution time. Starting points assume lower and ending points assume greater execution times, i.e. left sides of the graphs represent solutions with better performance (not explicitly drawn because of the illustrations clarity). Fig. 4a shows dependence of the execution time on component types, functional units, register files and multiplexers, shares in total component numbers. It can be noted that more functional units allow better performance as it is the main datapath design constituent allowing higher level of parallelism. However, it is not the only point of influence on execution time that is noticed. Greater number of functional units, even when having some redundancy, is better for performance when analyzing timing relationships between designs. The designs with more functional unit instances have more other component types also, i.e. register files and multiplexers. Their total count is greater, but their structures are simpler, i.e. register files have (in average) less registers and multiplexers use fewer inputs. Fig. 4b illustrates the execution times dependency on number of multiplexers per width showing their shares in total multiplexers count. The most significant in this graph is the drop of *Mux2* share that is happening along with performance drop. Reciprocally the other multiplexer widths types shares are greater when moving to the right of the graph what confirms the conclusion that components simplicity significantly influence the performance.

During system design it is usual to employ the concept of resource sharing not only by reuse of registers, but also the by use of functional units that can accomplish different operation types. Using such combined functional units reduces their overall number and numbers of other components that serve them by providing and storing data. Consequently it reflects in area occupation reduction. Therefore, the parallel solution space for *ordered12* solution set is formed by allowing combinations of different operation types within the same functional unit. Fig. 5. illustrates the relationships of area occupations (a) and execution times (b) between the cases with different functional unit sharing policies. With respect of area occupation it is noticed that in 92,5% of cases occupation is smaller when functional units sharing is allowed, but in 100% of such cases the execution speed is worse. This also leads to conclusion that complexity of datapath components is an important parameter for optimality of solution space as more complex units, i.e. combined functional units, have negative impact on timing and, consequently, design execution time.

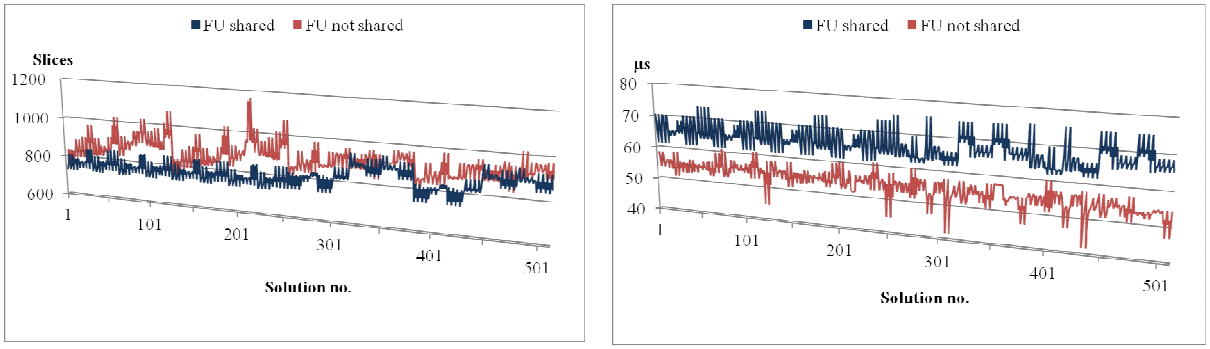
The simulation and implementation were accomplished in Xilinx ISE 14.3 Design Suite for Virtex XC5VSX50T device, [21, 22].

VI. CONCLUSIONS

The tradeoff between digital system occupied area and performance is open issue in every design process. Here we examined the design space evaluation for custom processor design that is based on automated C code algorithmic specification. This task is highly dependable on the quality of target hardware platform description and its correspondence with early design phases. The knowledge of implementation optimization principles is the to final solution occupation and performance estimation. Quality design space exploration and precise prediction of area-performance behavior depends on many parameters. Here we used control over functional units types to form the design space.

As the part of the overall custom processor design flow, we used proprietary IP tools for achieving FPGA device implementation so we could direct the design focus between area and performance only partially. However, we established the structure for design space exploration of custom processor design based development by constraining the functional unit component instantiation as a base datapath design component. Design space data collecting and reporting time was primarily dependable on particular tools run times and did not introduce noticeable time overhead.

We noticed that examined tradeoff is only partially impacted by execution cycles counts, but more on complexity of components.



(a) (b)

Figure 5. Comparisons of: (a) occupation area, and (b) execution time data with dependence on functional units sharing for *ordered12* set

Future work will be on exploiting established solution space to find best performance solution within strict occupation area constraint. Also, another point of future research interest is datapath interconnect reduction based on reduction of multiplexers widths to move the occupation-performance tradeoff focus closer to optimality within the solution space.

REFERENCES

[1] C. A. Mack, "Fifty Years of Moore's Law," IEEE Transactions on Semiconductor Manufacturing, vol.24, no.2, pp.202,207, May 2011

[2] W. Wolf, A. A. Jerraya, G. Martin, "Multiprocessor System-on-Chip (MPSoC) Technology," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems , vol.27, no.10, pp.1701-1713, Oct. 2008

[3] N. D. Arora,, K. V. Raol,, R. Schumann, L. M. Richardson, , "Modeling and extraction of interconnect capacitances for multilayer VLSI circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, , vol.15, no.1, pp.58-67, Jan 1996

[4] K. L. Shepard, Zhong Tian, "Return-limited inductances: a practical approach to on-chip inductance extraction," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.19, no.4, pp.425-436, Apr 2000

[5] P. Coussy, D. D. Gajski, M. Meredith, A. Takach, "An Introduction to High-Level Synthesis," in *IEEE Design & Test of Computers*, vol. 26, no. 4, pp. 8-17, July-August 2009

[6] TIOBE Software, "TIOBE Programming Community Index", http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[7] SourceForge, "The Transparent Language Popularity Index", http://lang-index.sourceforge.net

[8] M. Reshadi, B. Gorjiara, D. D. Gajski, "NISC Technology and Preliminary Results," Technical Report, University of California, Center for Embedded Computer Systems, Irvine, August 2005

[9] B. Gorjiara, D. D. Gajski, "Custom Processor Design Using NISC: A Case-Study on DCT algorithm," in *Workshop on Embedded Systems for Real-Time Multimedia*, pp. 55-60, 2005

[10] J. Trajković, S. Abdi, G. Nicolescu, D. D. Gajski, "Automated Generation of Custom Processor Core from C Code", Journal of Electrical and Computer Engineering, Vol. 2012, 26 p., March 2012.

[11] D. Ivošević, V. Sruk, "Automated Modeling of Custom Processors for DCT Algorithm", Proceedings of 34th International Convention MIPRO, Opatija, Croatia, pp. 762-767., 2011

[12] D. Ivošević, V. Sruk, "Unified Flow of Custom Processor Design and FPGA Implementation", Proceedings of International Conference on Computer as a Tool EUROCON, Zagreb, Croatia, pp 1721-1727, 2013

[13] C. Ebert, C. Jones, "Embedded Software: Facts, Figures, and Future", IEEE Computer, Vol. 42, No. 4, pp. 42-52., April 2009

[14] Taemin Kim; Xun Liu, "A Functional Unit and Register Binding Algorithm for Interconnect Reduction," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.29, no.4, pp.641-646, April 2010

[15] J. J. Rodríguez-Andina, M. J. Mouré, M. D., Valdés, "Features, Design Tools, and Application Domains of FPGAs", IEEE Transactions on Industrial Electronics, Vol. 54, No. 4., pp. 1810-1823, July-August 2007

[16] I. Kuon, J. Rose, "Exploring Area and Delay Tradeoffs in FPGAs With Architecture and Automated Transistor Design," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.19, no.1, pp.71-84, Jan. 2011

[17] M. Zuluaga, E. Bonilla, N. Topham, "Predicting best design trade-offs: A case study in processor customization," Design, Automation & Test in Europe Conference & Exhibition (DATE), pp.1030-1035, March 2012

[18] A. Orailoglu, D. D. Gajski, "Flow graph representation," in Proceedings of the 23rd ACM/IEEE Design Automation Conference, pp. 503 - 509, 1986

[19] Secure Hash Standards (SHS). [Online]. Available: http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

[20] C to Verilog. [Online]. Available: http://c-to-verilog.com/howtos.html

[21] ISE Design Suite - Xilinx, http://www.xilinx.com/products/design-tools/ise-design-suite

[22] Virtex-5 FPGA Data Sheet: DC and Switching Characteristics, http://www.xilinx.com/support/documentation/data_sheets/ds202.pdf