FPGA-based Control System Reconfiguration Using Open Source Software

Alexey Romanov, Mikhail Romanov, Alexander Kharchenko
Institute of Cybernetics
Moscow Technological University
Moscow, Russia
romanov@mirea.ru

***Abstract*— The paper proposes a new approach for FPGA reconfiguration based on usage of the open-source synthesis and place-and-route tools. The well-known partial reconfiguration technologies assume pre-compilation of all required FPGA configurations by the designer. In autonomous intelligent control systems it can become quite a strong limitation, due to control law couldn't be changed on-board without using external FPGA development tool. The paper proposes automatic Verilog code generation and synthesis directly on embedded controller, followed by full FPGA reconfiguration. The experimental results shows, that the resource usage of the FPGA configurations built by open-source tools are comparable to the ones built by well-known commercial toolchain)**

Keywords—— FPGA; Verilog; control systems; reconfiguration; Yosys; arachne-pnr; IceStorm

I. INTRODUCTION

Field-programmable gate arrays (FPGA) are widely used in the process of creating modern control systems. They significantly shorten the market launch period of new devices by enabling the reconfiguration of hardware/software both on the stage of the device's development and installation and throughout its lifecycle. This simplified the development of new devices and reduced the costs for full-scale tests of new microchip samples. Application of FPGA is particularly relevant in military and aerospace equipment [1, 2] that is typically characterized with wide range of various application-specific integrated circuits (ASIC) that are manufactured one-off or in small batches.

Using FPGA to create control systems has several advantages:

Cost reduction due to the usage of typical FPGA manufactured in medium and large volumes instead of small-batch specialized ASIC.

Smaller size and increased reliability due to the replacement of several separate microchips with one FPGA.

Possibility to update hardware remotely (if reprogrammable FPGA are used).

Most modern systems find only limited application for the FPGA capacity for reconfiguration. Most usually this is the default microchip configuration that is never changed further on. The most technically advanced systems also have the option of updating FPGA hardware by downloading the new configuration file that is sent by the system manufacturer in the process of planned update or in terms of debugging previously detected errors.

However, the FPGA potential for reconfiguration is significantly higher. Thus, one promising task is the modification of control system structure in order to adapt it to the environmental changes, type and range of control devices and feedback sensors. In the simplest case this task can be solved by simultaneously putting on the FPGA all hardware units necessary for all possible system configurations. This approach can only be executed in a laboratory environment since it requires expensive FPGA with high logical capacity. Other issues include the fact that a high number of units that are not used in every given moment leads to increased time delay and consequential reduction of maximum possible FPGA productivity and higher microchip energy consumption. An alternative solution would be to have several pre-prepared FPGA bitstreams with subsequent full FPGA reconfiguration if the control system operation is changed. The main drawback of this approach consists in the fact that linear increase in the number of parameters during reconfiguration results in the exponential growth of the amount of needed stored configuration files. For example, if the single axis motor drive control system with single feedback supports 4 possible types of actuating motors, 5 different sensor types and 3 different controller types, it would require storing up to 60 different FPGA bitstreams to ensure its operation in all possible modes. In order to reduce the number of needed bitstreams leading FPGA manufacturers suggested the partial reconfiguration technology [3, 4] that implies changing only the modified module instead of the whole FPGA. Thus, in the above described example the change of the sensor type would require updating only the sensor processing unit. This approach has the undisputable benefits of storing smaller amount of data that is needed to ensure system operation in all possible configurations and the opportunity to update separate FPGA sections without stopping the others. However, it also has several drawbacks:

Ensuring reconfiguration in the process of creating FPGA hardware would require observing several conditions [5], which requires re-engineering previously created FPGA modules to support reconfiguration;

Whole sections of FPGA will be reconfigured [5], which will prevent maximum utilization of FPGA resources when

 978-1-5090-4865-6/17/$31.00 ©2017 IEEE

interchangeable modules have different sizes (e.g. if a PID controller is interchangeable with a significantly more resource-intensive smart controller the FPGA should have pre-reserved resources to implement the latter; these resources will be wasted in a configuration with a PID controller).

There is another possibility to reconfigure an FPGA-based system in the course of operation. This is the synthesis of the needed FPGA bitstream directly on the embedded computer. The most promising application of this method would be (if the strategic and tactical control levels are implemented) to use Linux-operated embedded computers and the FPGA-based execution level. In this case the embedded computer of higher control level can be used to synthesize the most optimal FPGA bitstreams directly from the source code on HDL languages (hardware description languages). The main drawbacks of this approach are higher load of the embedded computer and longer synthesis time of the embedded computers compared to the specialized high-end workstations. However, configuring the control system hardware for a short range of tasks and specific set of control devices makes it possible to utilize FPGA with significantly lower logic capacity, which also simplifies synthesis and place and route tasks. Therefore, we believe that the capacity of modern embedded computers should be sufficient to perform these tasks.

At the same time this method has significant advantages:

- The final FPGA bitstream is optimized on the level of the whole microchip, not on the level of separate modules.
- No special requirements have to be observed. The final bitstream can implement all possibilities supported by the HDL and specific FPGA.
- There is no need to store separate non-editable bitstreams of modules or the whole FPGA. Hardware modules are stored as HDL descriptions that can be easily analyzed by the developer or specialized software, and can be modified if errors are found.

Until recently the main limitation for the FPGA synthesis on the embedded computer was the absence of software technically and economically suitable for this task. Corresponding software packages [6-8] released by the FPGA manufacturers or third-party developers (e.g. Synopsys) have extremely high prices, usually several thousand dollars per workstation. Installing such software on each manufactured control system is usually uneconomic. Another limiting factor are the minimal requirements of software packages for FPGA bitstream synthesis for the workstation hardware. Thus, a popular software package Xilinx ISE 14.7 requires over 15 Gb of hard drive free space and more than 3Gb of RAM [9]. This makes it almost impossible to run it on most embedded computers. At the same time, the functionality of this product significantly exceeds the needs for synthesizing control system elements, however Xilinx company doesn't provide any alternatives with lower hardware requirements. This situation is similar for other FPGA manufacturers.

II. Open-Source Software for FPGA Hardware Synthesis

The problem described above could be solved by using open-source software. This would enable creating its modification with minimally necessary functionality adapted to the embedded computer hardware. However the tasks of FPGA synthesis and place and route are so complicated that not even all microchip manufacturers can afford the development of own solution. For example, Russian manufacturer VZPP-S JSC suggests using Quartus package by the Altera company for its FPGA bitstream, and Chinese company Gowin Semiconductor uses Synplify Pro by Synopsys for synthesis. For a long time the only open-source solution was the VTR package [10], however it was initially oriented only on research tasks in the area of creating new FPGA architectures and provided no opportunity for synthesis and place and route for any of the really manufactured microchips. One significant step forward in this direction was the paper [11] that described the process of receiving the configuration file from the HDL description using VTR for Xilinx FPGA of the Virtex family. Results received in [11] demonstrate that VTR turned out to be a significantly less efficient instrument compared to the Xilinx ISE package. Bitstreams created with it required в 1.2-1.4 times more FPGA resources, the synthesis itself required up to twice as more memory, and its duration was up to 9 times longer compared to ISE. Besides, the approach suggested in [11] is not suitable for the task of reconfiguring FPGA-based control system described in this paper since its implementation requires complete installation of Xilinx proprietary packages that are used for place and route and creating the FPGA configuration file.

At the end of 2015 information emerged that in terms of IceStorm project [12] configuration file format for FPGA Lattice ICE40 was completely reconstructed. This made it possible to create a toolkit for implementing the whole cycle of operations needed to configure serial-produced FPGA based on HDL description with Verilog language using only open-source software. According to authors [12], the open toolchain they suggested can operate on systems fitted with only 512 MB of RAM and requires less than 1 GB of hard drive free space. This enables using the toolchain directly on the embedded computers, which opens up new opportunities for reconfiguring FPGA directly in the process of control system operation.

The goal of this paper is reviewing the possibility and evaluating the efficiency of using the tools described in [12] for the task of reconfiguring FPGA-based control systems.

III. Evaluation Criteria

The set of open-source components for the synthesis of FPGA hardware suggested by the authors [12] includes three components: HDL synthesizer Yosys [13], place and route tool Arachne-pnr [14] and a set of utilities for creating and downloading IceStorm configuration files [12]. This is the first and as of today only open-source toolchain that makes it possible to complete all stages of designing FPGA software from synthesizing the HDL code using Verilog language to downloading the bitstream into the FPGA. The reference used to evaluate the efficiency of open-source toolchain operation

for this paper will be the integrated FPGA design system ISE from the Xilinx company, which is the leader on the FPGA market as of 2016 [15]. Since both toolchains are designed for microchips of different manufacturers, it is important to choose specific FPGA models to evaluate the synthesis productivity and efficiency. The architecture and logical capacity of these microchips should be as close as possible. For the purposes of this research microchips Lattice ICE40HX8K-CT225 and Xilinx XC3S400-4FT256 were chosen. The technical characteristics of these microchips are provided in Table I.

TABLE I. CHARACTERISTICS OF MICROCHIPS USED IN THE RESEARCH

Characteristic	ICE40HX8K-CT225	XC3S400-4FT256
Logical element	LUT4+Flip-Flop	LUT4+Flip-Flop
Logical capacity (LUT4)	7680	8064
Block memory	128 Kb	288 Kb
Number of programmable inputs/outputs	206	173
Number of programmable differential inputs	26	76
Technological process of microchip manufacture	40 nm	90 nm

These microchips use similar basic logical elements based on 4 input lookup tables, and also practically identical logical capacity. More technologically advanced process was used to manufacture the Lattice microchip, which potentially gives it less delays compared to Xilinx chip, however this advantage is compensated with the more complex inner structure of the XC3S400-4FT256 microchip. From the application engineer's point of view both microchips have close technical characteristics and are meant to solve the same tasks, so using them to compare the efficiency of synthesis tools can be considered correct.

Following criteria were chosen to evaluate the efficiency of open source toolchain in this paper:

- Overall creation time of FPGA bitstream from the description in the Verilog language, including the time for the stages of synthesis, mapping, place and route, static timing analysis and bitstream generation;
- Number of used LUT4 in the resulting FPGA bitstream;
- Maximum operating frequency of the resulting FPGA bitstream based on the static timing analysis.

The evaluation by the above mentioned criteria was performed by conducting experimental synthesis and generation of FPGA bitstream based on 12 test HDL modules of varied type, size and task. Experiment automation was achieved using CoreBase toolbox [16] that was adapted for the Octave environment specifically for this task. The list of the test modules is provided in Table II.

TABLE II. THE LIST OF THE TEST MODULES USED TO EVALUATE PERFORMANCE OF THE OPEN-SOURCE FPGA TOOLCHAIN

#	Name	Description
1	cordic_core_v3	Sinus/cosinus generator, based on CORDIC alghoritm [17]
2	cordic_mult_core	Fixed-point multiplier, based on CORDIC alghoritm [18]
3	sd_foc	Permanent magnet synchronous motor field oriented control module, based on direct sigma-delta bitsream processing (DSDBP) [19]
4	sd_notch	Notch filter module, based DSDBP
5	sd_resolver	Resolver processing module, based DSDBP
6	sd_pid	PID controller module, based DSDBP
7	sd_modulator	First order sigma-delta modeulator
8	rndr_30	32-bit random number generator
9	j1_core	Forth processor core (without memory unit) [20]
10	rs232_rx	RS-232 receiver
11	sqrt_core	Fixed-point square root evaluation module
12	spi_master	SPI master module

IV. HARDWARE AND SOFTWARE USED DURING EVALUATION

The experimental study was performed on the basis of Dell Latitude E6530 personal computer with the characteristics described in Table III.

TABLE III. CHARACTERISTICS OF THE COMPUTER USED FOR THE EXPERIMENTAL STUDY

Processor	Intel Core I7-3720QM CPU, 2.6 GHz, 8 core
Architecture	x86
RAM	8 GB
Harddrive	SSD, 240 GB
Operating system	Ubuntu 16.04 LTS, 64 Bit
Octave version	4.0.1
Xilinx ISE version	14.7 WebPACK
Yosys version	0.5+396 (git sha1 e61c7f8, gcc 4.8.4-2ubuntu1~14.04 -fPIC -Os)
Arachne-pnr version	0.1+0+0 (git sha1 , g++ 5.4.0-6ubuntu1~16.04.2 -O2)

All experiments included the optimization of synthesis results by the occupied area.

V. COMPARISON IN TERMS OF SYNTHESIS EFFICIENCY

The results of the FPGA bitstream synthesis based on the test modules are provided in Table IV and also on Figures 1-3.

TABLE IV. THE RESULTS OF TEST MODULE SYNTHESIS

Test module #	ISE build time, s	Open-source toolchain build duration, s	Size (ISE), LUT4	Size(Open-source toolchain), LUT4	Maximal frequency (ISE), MHz	Maximal frequency (Open-source toolchain), MHz
1	90	29	203	282	86	77
2	79	24	113	179	138	117
3	134	74	827	1134	79	74
4	122	44	360	629	118	88
5	112	56	508	759	79	73
6	93	34	256	395	134	149
7	75	19	19	24	204	123
8	79	23	33	65	325	223
9	110	60	481	601	78	48
10	78	20	30	46	154	189
11	77	22	63	74	135	119
12	78	20	36	46	152	172

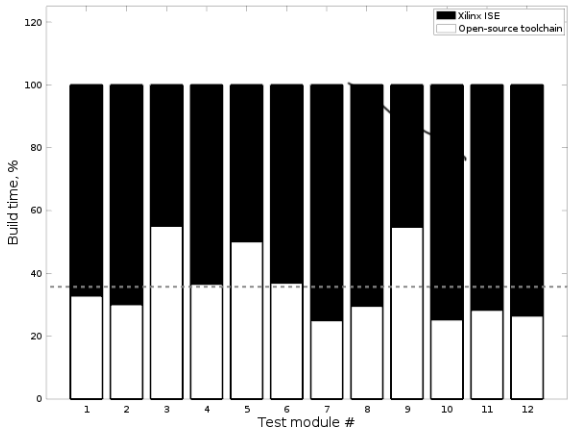

Fig. 1. Time needed to perform FPGA bitstream synthesis and generation. The mean level is shown by the dashed line.

The received results demonstrate that bitstreams created with open-source toolchains require on average 1.5 time more area compared to their analogues created with Xilinx ISE. In this respect the results of the reviewed open-source toolchain concur with the VTR results VTR described in [11]. According to the static-timing analysis the maximum operating speed of both toolchains is not significantly different, however the average results of open-source tools are 12% behind those of Xilinx ISE by this parameter. As for the overall time of PFGA bitstream synthesis, Yosys, Arachne-pnr and IceStorm are significantly better not only than VTR (basing on the results of [11]), but also than Xilinx ISE as they require on average 65% less time. The latter factor is extremely important for using these tools in the control systems on embedded computers.

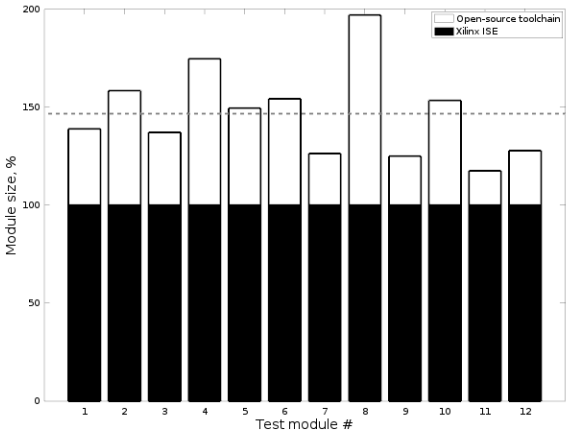

Fig. 2. Area (in LUT4) of synthesis results achieved with open-source toolchain compared to the area of synthesis results achieved with Xilinx ISE. The mean level is shown by the dashed line.

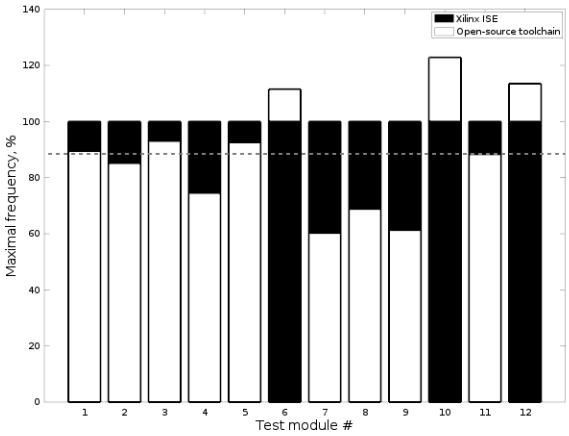

Fig. 3. Maximum operating speed of synthesis results achieved with open-source toolchain compared to the maximum speed of synthesis results achieved with Xilinx ISE. The mean level is shown by the dashed line.

VI. PERFORMANCE EVALUATION ON EMBEDDED COMPUTERS

The results of the above reviewed experiments demonstrate satisfactory efficiency and high productivity of the open-source tools for FPGA synthesis and bitstream generation suggested by the authors of [12]. Due to this fact it was decided to evaluate their productivity on two embedded controllers of different productivity and architecture.

First of such controllers was chosen to be the embedded computer of Kuka YouBot lab robot[21], its characteristics are provided in Table V.

In the experiment this embedded computer was used for the FPGA bitstream synthesis based on the test module #3 (sd_foc). The experiment demonstrated that the stages of synthesis, mapping, place and route and bitstream generation required 135 seconds, which is twice as long as the corresponding result achieved on the Core i7-based workstation. At the same time, the maximum memory volume

used in the bitstream generation process did not exceed 180 MB, and processor load totaled 100% for one of the two cores. The second core was not loaded, which leaves the opportunity to use it for running control programs parallel to solving the task of FPGA bitstream synthesis.

The second embedded computer to test the possibility of FPGA bitstream synthesis was the popular embedded system Raspberry Pi 3 Model B Rev 1.2. Its technical characteristics are provided in Table VI.

TABLE V. CHARACTERISTICS OF THE EMBEDDED COMPUTER USED FOR THE FIRST EXPERIMENT

Processor	Intel Atom D510 CPU, 1.66 GHz, 2 core
Architecture	x86
RAM	2 GB
Harddrive	SSD, 32 GB
Operating system	Ubuntu 12.04 LTS, 32 Bit
Yosys version	0.5+396 (git sha1 e61c7f8, gcc 4.7.3-2ubuntu1~12.04 -fPIC -Os)
Arachne-pnr version	0.1+0+0 (git sha1 , g++ 4.7.3-2ubuntu1~12.04 -O2)

TABLE VI. CHARACTERISTICS OF THE EMBEDDED COMPUTER USED FOR THE SECOND EXPERIMENT

Processor	ARMv7 Processor rev 4 (v7l), 1.2 GHz, 4 core
Architecture	armv7l
RAM	1 GB
Harddrive	SD card, 32 GB
Operating system	Arch Linux ARM 4.4.35, 32 Bit
Yosys version	0.7+38 (git sha1 a44cc7a, gcc 6.2.1 -march=armv7-a -mfloat-abi=hard -mfpu=vfpv3-d16 -O2 -fstack-protector -fPIC -Os
Arachne-pnr version	0.1+171+0 (git sha1 52e69ed , g++ 6.2.1 -O2)

During the experiment Raspberry Pi was also used to run the test module #3 №3 (sd_foc). It took 235 sec, which is almost twice as long compared to the time used by the Intel Atom processor in the previous experiment. Nevertheless, the results show that the open-source toolchain reviewed in the paper can be used for the FPGA bitstreams synthesis directly on the embedded computers in reasonable time.

VII. CONCLUSIONS

Basing on the experiment results it is possible to make the conclusion that the open-source toolchain suggested by the authors of [12] can efficiently solve the complete cycle of tasks connected with the FPGA bitsreams synthesis for real microchip samples. Indirect indicators demonstrate that the optimization of the resulting solutions in terms of occupied area is currently worse than the commercial analogues, but still is already acceptable for solving practical tasks.

At the same time the reviewed open-source toolchain has significantly lower hardware requirements and faster operation speed, which makes it possible to use it on embedded computers of adaptive control systems. The latter opens up the opportunity for online reconfiguration of the control system hardware, which is a relevant for:

- Adaptation of the interfaces of sensors and actuating devices;
- Changes in the structure of controller implemented on FPGA;
- Choice of optimal connection structure of hardware neural networks etc.

At the same time, compared to the solutions based on partial reconfiguration, the approach suggested in this paper is significantly more flexible and requires no preliminary preparation of any FPGA bitstreams.

The received results demonstrate the possibility of implementing the suggested approach and the promising potential of further studies of using open-source software for online reconfiguration of FPGA-based adaptive control systems.

REFERENCES

[1] Feduhin A., Muha A.A. PLIS-sistemy kak sredstvo povysheniya otkazoustojchivosti // MMC. 2010. №1 pp.198-204.

[2] Trusov V. A., Bannov V. YA., Grigor'ev A. V. PRAKTICHESKOE PRIMENENIE PLIS //Teoreticheskie i prikladnye aspekty sovremennoj nauki. – 2015. – p. 133.

[3] Dye D. Partial reconfiguration of Xilinx FPGAs using ISE design suite // White Paper. – 2012.

[4] Wiśniewski R., Wiśniewska M., Adamski M. Effective Partial Reconfiguration of Logic Controllers Implemented in FPGA Devices // Design of Reconfigurable Logic Controllers. – Springer International Publishing, 2016. – pp. 45-55.

[5] Partial Reconfiguraion User Guide (UG702) http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ug702.pdf

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

[7] Synplify Pro https://www.synopsys.com/tools/implementation/fpgaimplementation/pages/synplify-pro.aspx

[8] Quartus Prime and Quartus II Software Support https://www.altera.com/support/support-resources/design-software/quartus-ii/sof-quartus.html

[9] 1. Installation and Licensing Guide Xilinx Design Tools: Installation and Licensing Guide (UG798) www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/iil.pdf

[10] Rose J. et al. The VTR project: architecture and CAD for FPGAs from verilog to routing //Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays. – ACM, 2012. – pp. 77-86.

[11] Hung E., Eslami F., Wilton S. J. E. Escaping the academic sandbox: Realizing VPR circuits on Xilinx devices //Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annual International Symposium on. – IEEE, 2013. – pp. 45-52.

[12] Clifford Wolf and Mathias Lasser, Project IceStorm http://www.clifford.at/icestorm/

[13] Wolf C., Glaser J., Kepler J. Yosys-a free Verilog synthesis suite //Proceedings of 21st Austrian Workshop on Microelectronics, Linz, Austria. – 2013.

[14] Arachne-pnr Place and route tool for FPGAs https://github.com/cseed/arachne-pnr

[15] Sarma S., Sun S. L. THE GENESIS OF FABLESS BUSINESS MODEL: INSTITUTIONAL ENTREPRENEURS IN AN ADAPTIVE ECOSYSTEM //United States Association for Small Business and Entrepreneurship. Conference Proceedings. – United States Association for Small Business and Entrepreneurship, 2016. – C. R1.
[16] Romanov, A.; Bogdan, S., Open source tools for model-based FPGA design, in Control and Communications (SIBCON), 2015 International Siberian Conference on , vol., no., pp.1-6, 21-23 May 2015
[17] Volder J. E. The CORDIC trigonometric computing technique //IRE Transactions on Electronic Computers. – 1959. – №. 3. – pp. 330-334.
[18] Andraka R. A survey of CORDIC algorithms for FPGA based computers //Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays. – ACM, 1998. – = pp. 191-200.
[19] A.M. Romanov, "Analysis and synthesis of elements for mechatronic module systems control devices powered by FPLD, using sigma-delta modulation", Natural and Technical Sciences, "Sputnik+ company", 2013. - No 6, pp 348-361.
[20] Bowman J., Garage W. J1: a small Forth CPU Core for FPGAs //EuroForth 2010 Conference Proceedings. – 2010. – pp. 43-46.
[21] Lokhin V.M., Manko S.V., Romanov M.P., "The universal technologies testing ground for developing the intellectual and network-centric management of autonomous robot and multi-agent robotic systems", Herald of MSTU MIREA, № 3 (8) (September 2015, Issue 8) Vol 1, pp. 230-248, 2015