Development of a system generator for Marsohod development boards

P Dmitriy¹, I Alexey¹, A Romanov¹,²,³

¹ National Research University Higher School of Economics, 20 Myasnitskaya Ulitsa, 101000, Moscow, Russia
² e-mail: a.romanov@hse.ru
³ Author to whom any correspondence should be addressed

Abstract. This article describes a technical solution of the system generator of configuration files of development boards “Marsohod 2”, “Marsohod 2bis”, “Marsohod 3”, “Marsohod 3bis” for Quartus Prime software. The solution includes a web interface for the system generator, generation of configuration files, introduction of additional modules into the generated project, such as a frequency divider, Uart8 (RS-232), a module for preventing of contact bounce, and several types of simplest MIPS processor cores. This technical solution improves the convenience and speed of FPGA development, as well as reduces its entry threshold which can be significant for starting developers.

Keywords: FPGA, System Generator, Quartus, MIPS, Marsohod, Marsohod 2, Marsohod 2bis, Marsohod 3, Marsohod 3bis, Uart, RS232.

Introduction

When working with development boards based on FPGA and system-on-chip design, it is necessary to set up the project within the framework of the tasks assigned. The automation of this task contributes not only to reduction in time for commencement of work on a project, but also to decrease in the number of potential problems and mistakes by eliminating the need to go into peculiarities of working and specifics of realization of development boards.

Many manufacturers create their own tools for making custom preconfigured projects for their boards. For example, Terasic Company with each of their development boards provides special programs (system generators) on its website. They make it possible to choose which modules should be integrated into the project. Each board requires its own generator, which must meet the requirements of software portability and be supported by different operating systems. In addition, existing system generators are limited by the existing periphery and functionality of development boards only; they cannot integrate additional modules or common add-ins into the project.

Many boards of less advanced manufacturers do not have system generators at all. This work is aimed at decreasing the entry threshold and reducing the time required to create projects using Marsohod boards by automation of initial configuration process, which will also make it easier to study FPGA design. The developed solution is suitable for executing projects which do not require a specific configuration and allows obtaining configuration files and preconfigured projects for development boards of Marsohod family.
1. Description of technologies used in the project

Currently, there are two main types of programmable logic devices (PLD): CPLD (Complex Programmable Logic Device) and FPGA (Field-Programmable Gate Array). The main difference between them is firmware saving after switching off the power supply of the board (such properties are possessed by devices like CPLD). Among Marsohod development board family, it is the first Marsohod that has CPLD chip. MAX II chip is installed on the first Marsohod, and Marsohod3, Marsohod3bis boards have MAX 10 chips [15]. Marsohod2 and Marsohod2bis are equipped with FPGA Cyclone III [13], and Cyclone IV [14] chips, respectively. All considered boards use PLDs from Intel FPGA company.

For each board from Terasic company of DE1-SoC type [12], based on Cyclone V, there is a specialized software – System Generator through which .qpf, .qsf, .v, .sdc files can be generated, and it is also possible to disable/enable some integrated peripherals on PLD and on the board for further operation in Quartus Prime project [3]. In addition, in Quartus Prime, there is a Template mechanism, so that a project for a specific board can be generated in Project Wizard.

The main differences of the developed technical solution for “Marsohod” boards are:
- the way of user interaction through web interface instead of a separate application;
- the possibility of introducing additional modules (in Verilog language) to the project;
- orientation to “Marsohod” development boards;

2. Description of development drift

As a result of analyzing of design principles of projects for PLD with following prototyping, as well as of analyzing of structure of the configuration files, the following main configuration files were identified:
- .qpf file containing the project description for Quartus Prime;
- .qsf file containing the information about the type of board, the used resources, and the assignments of the crystal pins;
- the top level file of the hierarchy (file in Verilog language) containing the description of the top level module in hardware description language [6];
- .sdc file containing information about the time delays of the board;

The main documents for working with “Marsohod” development boards are their technical specifications [2] which are a declarative description of the characteristics of the board, the available resources, and the ways to connect them. To present the board characteristics in a convenient form, and to further generate the configuration files out of them, the following set of formats was chosen: YAML [7] (used by default), JSON [5], and binary serialization Python (pickle). To support these formats, the developed software solution contains a special module that supports the conversion of documents of different formats. The choice of a format is due to the following considerations (in descending order of importance): convenience of reading and editing by a human; document size; prevalence. YAML is chosen as a compromise variant, because it has better readability, a satisfactory file size, and is common enough.

The developed software has a flexible modular structure that allows its parts being used for various variants of their implementation and consists of the following:
- The system core (“thumb”) is a module that encapsulates the generation of configuration files and performs related actions, such as writing the configurations into files on the disk, packaging configurations into an archive, loading templates for generation, event logging, etc;
- External interfaces – a set of modules that exploit the system core to provide a friendly interface to the end user. These modules include web service that allows generating configuration files;
- Configuration module of the high-level parameters for individual parts of the system representing a centralized repository of settings that define the system behavior without the necessity to change the entire logic of the program;
- Unit tests and functional tests for individual parts of the system;
• Documentation for the developed system;

The generator of configuration files is included into the system “core” and provides its main functionality. Each type of development boards is presented in the form of a separate class that provides loading corresponding document which describes the board through reflection method [8]. Each class of development boards is a child object of a generic class that implements the file generation in accordance with the loaded configuration, archiving and writing created files into the disk.

To generate the files, Jinja2 template library [4] is used which is an interface that allows performing substitutions with special internal language DSL (Domain-Specific Language) [9]. The use of this solution is due to the following reasons:

• The presence of a large number of different templates for generation. This makes it possible to avoid own tool development leading to more time consumption to perform such design stages as architecture development, implementation and debugging of the system, fine tuning, etc;

• The necessity to ensure the possibility to make patches and additions into the templates by the third party which provides extension of the proposed solution onto new development boards not limited to “Marsohod” boards only;

• The ability to highlight the common parts in the templates of the files. This Jinja2 feature allows introducing the additional information at the beginning and at the end of each generated file;

Basically, the generating process of the configuration files consists of the following sequence of actions:

• Determining the type of development board;

• Downloading relevant documents with board descriptions;

• Performing the substitutions in accordance with the requirements of the user;

• Saving the resulting configurations into the specified directory;

During the software behavior, performed actions and occurring errors logging are maintained. The log is kept in accordance with the required level of detail (debugging information, information about major events, information about errors, etc.) and includes information about the date of occurrence, level, place of event occurrence, and the additional message about the event. Event logging at the files generator level allows analyzing possible user problems occurred during the operation of the entire system.

To provide a cross-platform character of the developed solution, the variant of creating a web service was chosen which allows generating the required configurations for development boards. This solution has the following advantages:

• Availability on any device under the control of any OS including the mobile one;

• Centralized control of the entered changes which ensures that there is no need to make changes to application clients for each OS;

• Distribution control (providing additional content to a limited number of registered users depending on their access rights, for example, providing special setup options for students of certain institutions of higher education);

• Collecting of statistics of the application operation (the user registration mechanism allows receiving statistics of system usage including the addition of a feedback mechanism);

The necessity of Internet connection availability for functioning of the service is a limiting factor, but in modern realities, it is not so important.

The developed service provides the following functions:

• Generation of configuration files for development boards;

• Logging in by using an email address to gain an access to additional functionality;

• Viewing of articles associated with the system-on-chip development (implemented for authorized users);
• Downloading additional materials (for example, to perform laboratory work on Marsohod boards) for authorized users;

The service distribution is provided by message mailing to specified emails with a special link that contains a characters sequence (token) in JSON Web Tokens (JWS) format inside which the link expiration time and the user id in the database are encoded. The data is encrypted according to HS256 algorithm. This format is also used to generate the links for the materials distributed by the service which allows hiding the real structure of the files location and protecting the service from configuration files leakage and executing of arbitrary code. To manipulate the database, an object-relational model is used [10]. The developed software has a mobile version of the site, as well as Russian and English interface versions.

3. Additional modules for the system generator

In addition to the main modules, the system generator is capable to add various add-ons that allow additional modules to be integrated into the final generated projects (analogue of IP cores in Quartus Prime).

One of the analogues is Platform Designer (previously, QSYS). Platform Designer is a system for device designing at the level of system components; it allows combining IP cores while creating its own device logic based on the composition of these cores. Platform Designer also supports the ability to integrate own IP cores [17]. The second analog is MegaWizard Plugin [18]. This tool allows integrating Megafungions into the project. Megafungions are IP blocks parameterized and optimized for Intel FPGA (Altera) device architectures. Intel FPGA is a mega-function library including Library of Parameterized Module (LPM) functions and other parameterized functions that offer more efficient logic synthesis and implementation of digital devices [19].

The difference between the proposed implementation and the analogs, considered above, is the support for additional modules that are absent in MegaWizard Plugin and Platform Designer, such as the contact bounce module and UART module. Also, a distinctive feature is the support for inclusion of the simplest processor core in the project. Additional modules:

• **Debounce module** – the problem of contact bounce is relevant in the implementation of the processing systems of signals applied as a result of mechanical action (for example, by pressing the mechanical button). The essence of the problem includes in the fact that the contacts closure of mechanical element can catch several cycles of device operation, and, therefore, switching will occur several times. The debounce module provides the required delay before switching. The size of the delay can be adjusted by the parameters changing of CLOCK_RATE and MILLIS_DELAY. The first parameter indicates the clock frequency of the internal generator of the board and is set automatically while the second parameter, indicating the delay amount in milliseconds, is set by the user from the system generator interface (Fig. 1).

![Figure 1. Block diagram of the contact bounce eliminating module connection.](image)

• **Frequency divider module** – the limited number of frequency generators with a fixed frequency can lead to problems when working with peripherals, since the devices can operate at completely different frequencies. The implemented frequency divider module reduces the frequency of the board internal generator to the set one. CLOCK_FREQ parameter is
responsible for the incoming frequency (cycles per second); OUT_FREQ parameter is responsible for output frequency (cycles per second).

![Block diagram of the frequency divider module connection](image1)

**Figure 2.** Block diagram of the frequency divider module connection.

- **UART (RS-232) module** – Universal Asynchronous Receiver Transmitter (UART) is a serial universal asynchronous interface that is responsible for transfer data between digital devices [11]. This implementation makes it possible to separately control the operation of the receiver and transmitter (rxEn, txEn) using RS-232 protocol and provides for the installation of 2 parameters for flexible adjustment of the device operation. CLOCK_RATE parameter is responsible for the internal frequency of the development board; BAUD_RATE parameter is responsible for the frequency of the receiving / transfer bus.

![UART module Block diagram](image2)

**Figure 3.** UART module Block diagram.

- **schoolMIPS** – a small MIPS processor core is originally based on Sarah L MiraS processor from Sarah L. Harris[1], and its first version was developed for Young Russian Chip Architects Summer School [16]. The core has several versions. All implementations (simple, mmio, irq, pipelined, pipeline_irq, pipeline_ahb) were taken from the github repository and integrated into the current solution.

![Simple MIPS](image3)

**Figure 4.** Simple MIPS.
4. Testing
Testing of the developed system was carried out using Google Chrome 68 (x64) browser, on Marsohod3 and Marsohod2 platforms; for CAD Quartus Prime 17 – Marsohod3, and for Quartus II 9.0 – Marsohod2, on Windows 10 (x64) operating system.

Estimated calculation of consumed resources by the additional modules:
- Module for eliminating of contact bounce with parameters CLOCK_RATE = 100 MHz, MILLIS_DELAY = 100 Hz – 31 LUT, 23 REG.
- Module of the frequency divider with the parameters CLOCK_FREQ = 100 MHz, OUT_FREQ = 1 MHz – 12 LUT, 7 REG.
- Control module by seven-segment indicator based on a counter at 16 - 12 LUT, 4 REG.
- Module UART8 with parameters CLOCK_RATE = 100 MHz, BAUD_RATE = 9600 Hz, – 65 LUT, 38REG.

Estimated calculation of consumed resources for school MIPS processor:
- simple – 164 LUT, 65 REG.
- mmio – 159 LUT, 65 REG.
- irq – 171 LUT, 66 REG.
- pipeline – 1012 LUT, 403 REG, 2048 memory bits.
- pipeline_irq – 1004 LUT, 403 REG, 2048 memory bits.
- pipeline_abh – 1177 LUT, 403 REG, 2048 memory bits.

Testing of the server side:
- Server startup time ~ 5 s.
- Resources consumed: 100 MB.
- Response time ~ 1 ms.
- Simultaneous finding of users on the site – more than 500.

5. Conclusion
As a result of technical solution implementation, the following results were obtained:
- The configuration files generator was implemented;
- Additional functional modules were developed in Verilog language;
- The implementing possibility of the developed functional modules in the project were realized;
- The possibility of integration of a simple processor core schoolMIPS into the project was realized;
- web interface for user interaction was implemented;
- User authorization mechanism was created to provide selective access to the required sections of the proposed technical solution;
- Documentation for rapid deployment of a developed technical solution.

All the above made it possible to significantly improve the operation convenience with the development boards Marsohod 2, Marsohod 3, Marsohod 2bis, Marsohod 3bis and to increase the speed of projects development on these boards. The entry threshold for development on these boards was also reduced which may increase their attractiveness for training in FPGA development. The possibility to implement modules in the projects, which are not realized in such analogues as MegaWizard Plugin, or Platform Designer, allows quick solving of a number of additional tasks.

Acknowledgment
The publication was prepared within the framework of the Academic Fund Program at the National Research University Higher School of Economics (HSE) in 2018-2019 (grant № 18-01-0074) and by the Russian Academic Excellence Project “5-100”.
References

[1] Harris D and Harris S 2014 *Digital Design and Computer Architecture*

[2] Strogonov A and Bystritsky A 2013 "Designing systolic FIR filters in the FPGA basis using the ModelSim-Altera modeling system" (Components and technologies) chapter 9 pp 58-62

[3] Gupta P 2016 *Accelerating datacenter workloads* (26th International Conference on Field Programmable Logic and Applications)

[4] Ronacher and Armin 2008 *Jinja2 Documentation*

[5] Bradley J and Sakimura N 2015 *Json web token* (RFC 7519)

[6] Polyakov A 2018 "VHDL and VERILOG languages in the design of digital equipment" (Litres)

[7] Sinha V Doucet F Siska C Gupta R Liao S and Ghosh A 2000 "YAML: a tool for hardware design visualization and capture" (Proceedings of the 13th international symposium on System synthesis. IEEE Computer Society)

[8] Malenfant J Jacques M and Demers F 1996 "A tutorial on behavioral reflection and its implementation" (Proceedings of the Reflection) p 1-20

[9] Mernik M Heering J and Sloane A 2005 "When and how to develop domain-specific languages" (ACM computing surveys) p 316-344

[10] Schnicariol K Schnicariol M and Schnicariol M 2009 *Object-relational mapping* (Pro JPA 2. Apress) p 69-106

[11] Wei-qing, and Wei-Zhou 2009 *Design of USB-RS232 converter module based on FT2232H* [J] (Electronic Design Engineering) chapter 7

[12] Kashani-Akhavan Sahand and Beuchat R *SoC-FPGA Design Guide DE1-SoC Edition*

[13] Corporation Altera 2012 *Cyclone III device handbook*

[14] Corporation Altera 2016 *Cyclone IV, Device Handbook*

[15] Membrey P and Pang A 2017 *Lock and Load* (Beginning FPGA: Programming Metal. Apress, Berkeley) p 23-45

[16] Ryazanova A Romanova I 2018 *Analysis of options for the implementation of the interaction subsystem for a multiprocessor-on-chip system based on schoolmips*

[17] Edwards S 2015 *Embedded System Design Lab: Peripherals and Device Drivers* (CSSE)

[18] Bochem A Deschenes J Williams J and Kent B 2014 *An Overview of Embedded Systems and Prosthesis Technology* (Faculty of Computer Science University of New Brunswick Frederiction, NB, Canada)

[19] Jose S 2009 *Megafuction overview user guide*