The Implementation of Evolvable Hardware Closed Loop

CHU Jie, ZHAO Qiang, DING Guo-liang, YUAN Liang
Department of Computer Engineering,
Mechanical Engineering College, Shijiazhuang 050003, China
chujie2000@163.com

Abstract

To find an easy experimental way for evolvable hardware (EHW), an closed loop platform was discussed based on a Field Programmable Gate Array (FPGA) chip, a Microprocessing Unit (MPU) and the software tool-Quartus II. A binary digit to VHDL converter was proposed. It facilitated the using of the different FPGA chips as evolving carriers and avoided learning of the technological details inside the chips and analyzing of the different and complicated bitstream structures of chips. Tool command language (Tcl) scripting was used to create a Quartus II project and add VHDL files into the project automatically. Command-line operation and batch processing were used to achieve analysis, synthesis, and programming FPGAs automatically step by step. The closed loop was implemented, which is the key of online evolution study for EHW. The students can conduct EHW experiments and needn't to know the bitstream structures of FPGA with the closed loop platform.

1. Introduction

EHW refers to hardware that can change its architecture and behavior dynamically and autonomously by interacting with its environment. At present, EHW always uses genetic algorithm (GA) as its main adaptive mechanism. EHW is usually implemented on programmable logic device (PLD) such as FPGA [1]. By designing a fitness function to achieve a desired hardware function, the GA becomes a means of autonomous hardware reconfiguration. EHW regards the configuration bits of FPGA as the chromosomes of GA. Configuration bits "evolved" by the GA are repeatedly downloaded into the FPGA until the EHW performance is satisfactory in terms of fitness function value [2].

However, commercial FPGAs still remain difficulties when implementing evolvable hardware systems. Commercial FPGAs have little or no accessibility to internal nodes for probing. As well as impeding analysis, FPGAs offer little choice of circuit primitives, restricted interconnection architecture, and most are susceptible to self-destruction [3].

Without knowledge of the configuration bitstream's construction, it is difficult to use an evolutionary process to create and test low-level bitstreams based candidate solutions without risking device damage. Without resorting to well documented, simplistic, or out-of-date technology like the Xilinx® XC6200 [4], or proprietary bitstream manipulation tools, such as the also out-of-date JBits API [5], users must utilize a reconfigurable superplatform on top of an existent FPGA [6], [7], [8].

At the same time, automatic operation of the evolution software is another difficulty. There are too many factors involved in EHW, and the corresponding evolution time often lasts hours, days or even longer. How to avoid the human operating or intervening is another important problem for realization of EHW platform. So, a fully automatic procedure must be employed to finish the job.

2. Hardware design for EHW platform

An EHW platform is built, as shown in Figure.1. The EHW platform has an AT89C2051 as the MPU and a FPGA EP1K30TC144-3 as the evolution carrier. Host PC runs a GA program to evolve the configuration bitstreams of the FPGA and then download bitstreams into the FPGA with an interface from parallel port to JTAG port [9]. A Virtual Instrument application program is used to set the parameters of evolution process.

978-0-7695-3357-5/08 $25.00 © 2008 IEEE
DOI 10.1109/ICICTA.2008.349

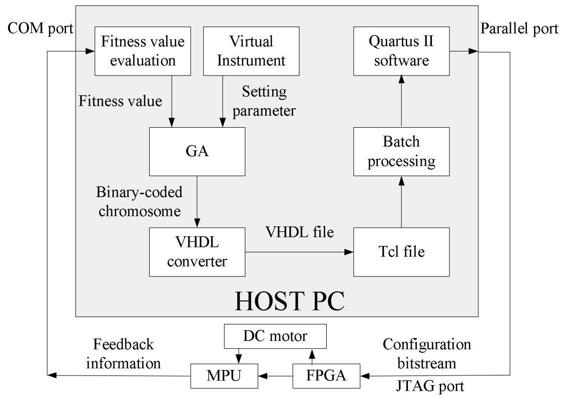

Figure 1. Online evolution framework

3. Software design for EHW platform

The FPGA EP1K30TC144-3 is one of the products of Altera corporation. It needs the assistant of Altera Quartus II design software which is a comprehensive environment for system-on-a-programmable-chip (SOPC) design. The Quartus II software includes solutions for all phases of FPGA design.

3.1. Command-line operation and Tcl Scripting

Each stage of the Quartus II software design flow corresponds to a command-line operation. Many of these command-line operations also support industry-standard Tcl scripting for custom functionality or processing beyond the Graphical User Interfaces (GUI) design flow. Tcl is a popular scripting language that is similar to many shell scripting and high-level programming languages. Developing and running Tcl scriptings to control the Altera Quartus II software allows designers to perform a wide range of functions, such as compiling a design or writing procedures to automate common tasks [10].

3.2. Binary digit-VHDL converter

It's important to avoid the analyzing of FPGA bitstreams, get rid of the constraint of the specified chips, and set up an EHW environment without dealing with a specific hardware. The key lies in transfering the chromosome figure based on binary code to the VHDL language procedure that is in a text form [11]. So an accurate, high-efficient binary digit-VHDL converter (D-V converter) is designed and implemented.

All kinds of digital circuits can be divided into three basic categories, the AND, OR and NOT gates. A circuit is just composed of gates and wires linked among them. So, the description of the chromosome in EHW can be changed into the description of the circuit according to certain rule. Based on the grammar that VHDL language followed, the main function of D-V converter is to analysis the chromosome codes produced by the individual from a GA, and to convert chromosome codes into VHDL files automatically. As shown in Figure.1, the VHDL files of the FPGA will be offered for the Tcl scripting file.

3.3. Tcl scripting codes

In the closed loop system of EHW, how to automate the process of closed loop is the key. The automation of the closed loop is implemented with command-line operations and Tcl scriptings. Tcl scriptings can be used to create a Quartus II project, define design constraints, make device assignments and add VHDL files into the project, as shown in following procedure.

```
# Check that the right project is open
if {[is_project_open]} {
        if {[string compare $quartus(project)
"file_name"]} {
                puts "Project file_name is not open"
                set make_assignments 0
        }
} else {
        # Only open if not already open
        if {[project_exists file_name]} {
                project_open -cmp file_name
file_name
        } else {
                project_new -cmp file_name
file_name
        }
        set need_to_close_project 1
}
# Make assignments
if {$make_assignments} {
        catch { set_global_assignment -name
ORIGINAL_QUARTUS_VERSION 5.0 } result
        catch { set_global_assignment -name
PROJECT_CREATION_TIME_DATE "11:48:15
FEBRUARY 21, 2008" } result
        catch { set_global_assignment -name
LAST_QUARTUS_VERSION 5.0 } result
        catch { set_global_assignment -name
VHDL_FILE file_name.vhd } result
 #Add a VHDL file to the project
        catch { set_global_assignment -name
COMPILER_SETTINGS file_name } result
        catch { set_global_assignment -name
SIMULATOR_SETTINGS file_name } result
        catch { set_global_assignment -name
SOFTWARE_SETTINGS file_name } result
        catch { set_global_assignment -name
FAMILY ACEX1K } result
```

```
        catch { set_global_assignment -name DEVICE "EP1K30TC144-3" } result
        catch { set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 1 } result
        catch { set_global_assignment -name USE_COMPILER_SETTINGS file_name } result
        catch { set_location_assignment PIN_117 -to m0 } result
        catch { set_location_assignment PIN_23 -to s0_final } result
        # Commit assignments
        export_assignments
        # Close project
        if {$need_to_close_project} {
                project_close
        }
}
```

3.4. Batch processing

After the Quartus II project is built with Tcl scriptings, other work for the project, such as analysis and synthesis, placing and routing, generating device programming images and programming FPGAs, can be achieved by the command-line operations. When the following command-line operations are written in a batch file, the work can be done automatically step by step.

- quartus_sh -t file_name.tcl

The Quartus II Shell acts as a simple Quartus II Tcl interpreter. The Shell can run a Tcl scripting-file_name.tcl.

- quartus_map file_name

This Quartus II command builds a single project database that integrates all the design files in a design entity or project hierarchy, performs logic synthesis to minimize the logic of the design, and performs technology mapping to implement the design logic using device resources such as logic elements.

- quartus_fit file_name

This Quartus II command performs place-and-route by fitting the logic of a design into a device. The fitting selects appropriate interconnection paths, pin assignments, and logic cell assignments.

- quartus_asm --import_settings_files file_name

This Quartus II command generates a device programming image in the form of Programmer Object Files(.pof).

- quartus_pgm -c byteblastermv file_name.cdf

This Quartus II command programs Altera devices with Programmer Object Files(.pof). The file_name.cdf is an ASCII text file (with the extension .cdf) that stores device name, device order, and optional programming file name information.

4. Experimental target choosing

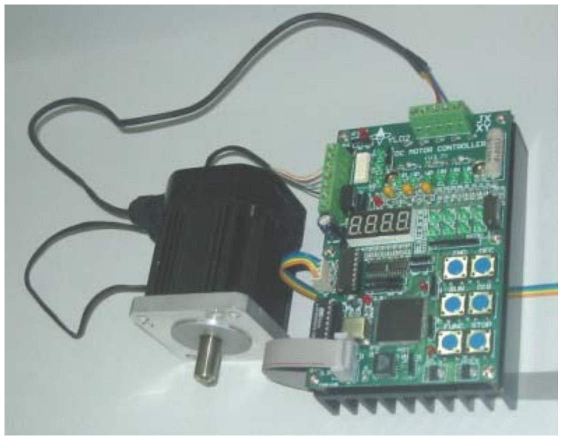
Figure 2. BLDC motor and control system based EHW

As shown in Figure.2, a three phase brushless DC (BLDC) motor is used as the controlled target in the EHW platform. There are three coils A, B and C in the fundamental three phase BLDC motor, as shown in Figure.3.

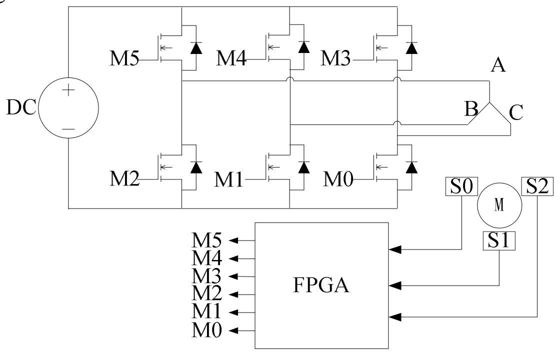

Figure 3. Control system schematic diagram

4.1. Controlling principle

Figure.3 indicates how the FPGA controller controls the BLDC motor. When BLDC motor works, its rotor-position sensors send out position signals S2-S0. Together with direction signal D, those signals produce six-state signals M5-M0 according to a fixed function, and then provided to six Insulated Gate Bipolar Transistors(IGBT) in an Intelligent Power Module (IPM) that drives the motor.

4.2. Constraint condition and object function

The output logic formulas that determined by the signals from position sensor decide the next position of rotor, and all these positions make up the whole circle. The object function is as the follows.

$$M5 = (\overline{S0} + S1 + \overline{D}) \bullet (S0 + \overline{S1} + D)$$
$$M4 = (\overline{S2} + S0 + \overline{D}) \bullet (S2 + \overline{S0} + D)$$
$$M3 = (\overline{S1} + S2 + \overline{D}) \bullet (S1 + \overline{S2} + D)$$
$$M2 = (\overline{S1} + S0 + \overline{D}) \bullet (S1 + \overline{S0} + D)$$
$$M1 = (\overline{S0} + S2 + \overline{D}) \bullet (S0 + \overline{S2} + D)$$
$$M0 = (\overline{S2} + S1 + \overline{D}) \bullet (S2 + \overline{S1} + D)$$

4.3. Fitness evaluation

For the control system, fitness function of GA equals to its object function, namely, only by implementing the object function correctly, can the motor rotate normally.

There are 6 binary codes in signal S2-S0 to form a set of input data, which determine 6 changing phases of the rotor used by evolving circuit. When evolved result in one phase is the same with that of a normal module, it means the result fits what the motor needed, so, GA can go on to deal with the next phase , until the process of all 6 phases are completed.

4.4. Operation settings

In this case, single crossover function is used. Namely, select one bit randomly through computer, change left and right parts of it, then create 2 new individuals, which inherit the characteristic of parent individuals.

The population of chromosome is 10, the length of chromosome string is 6, cross probability is 80%, and mutation probability is 0.001. Parameter choosing always causes a significant influence. But at present, it doesn't have a constant rule, only some experience can be summarized, so more researchers are needed greatly.

5. Conclusion

The EHW platform with closed loop is a convenient experimental environment. It keeps away from analyzing the bitstreams of FPGAs. It can also achieve the online evolution automatically. Then, a control circuit for a BLDC motor is implemented in evolutionary way.

6. Acknowledgements

The project is supported by the National Natural Science Foundation of China (No. 60471022), National Defense Science and Technology Key Laboratory Foundation (No.9140C8702020803) and Hi-Tech Research and Development Program of China (No. 2007AA01Z454).

7. References

[1] Xin Yao, Tetsuya Higuchi, "Promises and Challenges of Evolvable Hardware", *IEEE Transactions on Systems, Man, and Cybernetics*, *Part C 29(1)* , Publisher, Location, 1999, pp. 87-97.

[2] Tetsuya Higuchi, Yong Liu and Xin Yao, *Evolvable hardware*, Springer, Heidelberg, 2006.

[3] Paul Layzell, *Hardware Evolution: On the Nature of Artificially Evolved Electronic Circuits*, Submitted for the degree of D.Phil, University of Sussex, Brighton, UK, May, 2001.

[4] Xilinx, *XC6200 Field Programmable Gate Arrays - datasheet*, Xilinx Corporation, San Jose, California，1997.

[5] Xilinx, *JBits SDK web site*, Xilinx Corporation, San Jose, California，2007.

[6] Sekanina, L., "Towards Evolvable IP Cores for FPGAs", *Proceedings of the 3rd NASA/DoD Conference on Evolvable Hardware, EH-03*, IEEE Computer Society Press, Washington, DC, USA, 2003, pp. 145-154.

[7] Sekanina, L., "Virtual Reconfigurable Circuits for Real-world Applications of Evolvable Hardware", *ICES 2003.* Springer, Heidelberg, 2003, pp. 186-197.

[8] A.J. Greensted, A.M. Tyrrell, "Extrinsic Evolvable Hardware on the RISA Architecture", *ICES 2007*, Springer, Heidelberg, 2007, pp. 186-197.

[9] Yuan Liang, Ding Guoliang, Wu Wenshu, Lou Jianan and Zhao Qiang, "Building An EHW Environment Based on FPGA and QUARTUS", *Computer & Digital Engineering,* Publisher, Wuhan, 2006. 05, pp. 1-3.

[10] Altera, *Quartus II Scripting Reference Manual,* Altera Corporation, 2007.

[11] Yuan Liang, Huang Feiyun, Liu Wenbing and Liu Wenjie, "FPGA-Based Experimentation for TMR Structure and Evolutionary Approach of Self-Recovering", *Proceedings of the First International Conference on Maintenance Engineering*, Science Publishing Company, Beijing, 2006.10, pp. 172-179.