Boise State University
ScholarWorks

Electrical and Computer Engineering Faculty Publications and Presentations

Department of Electrical and Computer Engineering

8-1-2009

On-Chip Intrinsic Evolution Methodology for Sequential Logic Circuit Design

Fan Xiong
Boise State University

Nader Rafla
Boise State University

©2008 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE. DOI: 10.1109/MWSCAS.2009.5236119

On-Chip Intrinsic Evolution Methodology for Sequential Logic Circuit Design

Fan Xiong
Electrical and Computer Engineering
Boise State University
Boise, ID 83725
fanxiong@u.boisestate.edu

Nader I. Rafla, Ph.D., P.E.
Electrical and Computer Engineering
Boise State University
Boise, ID 83725
nrafla@boisestate.edu

Abstract

This paper focuses on the application of Virtual Reconfigurable Circuit (VRC) design methodology and intrinsic evolution for the design of small sequential circuits and their implementation on a single programmable chip with an embedded hardcore processor. The evolutionary algorithm is developed in software that runs on the embedded processor. Fitness function is calculated using hardware architecture and is used to guide the evolution process. This new method is applied to the development of a 3-bit sequence detector and the evolved architecture is implemented on a Xilinx^{tm} Virtex-II pro device. Simulations were run on the evolved architecture and on the same circuit designed using conventional Hardware Descriptive Language (HDL). Both designs showed the same functional behavior. Synthesis results show that the new method can be used in successfully implementing small sequential circuits on a reconfigurable hardware environment.

1. Introduction

Evolvable Hardware (EHW) is a new bio-inspired technique that uses evolutionary algorithm (EA) to auto-configure and optimize circuits [1][2]. By exploring a large design search space, EHW may find solutions for a task, unsolvable, or more optimal than those found using traditional design methods. Instead of manually designing a circuit, only input/output-relations are specified. The circuit is automatically designed using an adaptive algorithm inspired from natural evolution.

EHW encodes circuit structure and parameters as chromosomes using one of several predefined encoding methods. EA uses these chromosomes to generate random bit-sequence for chip configuration. A fitness value is then calculated and used to judge and refine the sequence. Fitness is an indicator for the goodness of the solution expressed by a particular chromosome.

There exist two methods for calculating fitness in EHW; intrinsic and extrinsic. In intrinsic EHW, the fitness evaluation is done after downloading the configuration sequence to the chip. The circuit is then tested and its output is compared to predetermined expected results. Comparison results are used to guide the evolution process to either continue or stop. On the other hand, extrinsic EHW uses software program to evaluate the fitness, reiterate if necessary and finally download the resulted configuration bit-sequence to the chip, so it's also called off-line evolution.

There are four commonly used EAs in EHW: Genetic Algorithm (GA), Genetic Programming (GP), Evolutionary Programming (EP) and Evolutionary Strategy (ES). GA is the most widely used evolutionary algorithm [3].

As has been discussed in previous work [4], the majority of circuits that have been evolved are combinational logic circuits, and only a small portion of work has been done on evolving sequential logic circuits. Although some researchers made useful contributions to this area [4][5][6][7][8], the research of evolving sequential logic circuits is still in its early stages and did not use the combination proposed in this research.

In this paper, we propose a novel method to evolve sequential logic circuits using intrinsic evolution and virtual reconfigurable circuit technique. A simple GA is used along with a new structure of VRC to design flexible and easy reconfigurable architecture. This architecture is suitable for implementation on a Field Programmable Gate Array (FPGA) platform. Since the FPGA configuration bit-stream is considered proprietary information and it is challenging to deal with, VRC is best suited to use in such a situation. Although some researchers tried to use reverse

978-1-4244-4480-9/09/$25.00 ©2009 IEEE

Authorized licensed use limited to: Boise State University. Downloaded on October 12, 2009 at 11:32 from IEEE Xplore. Restrictions apply.

engineering to obtain Xilinx™ FPGA bit-stream file format, these efforts deemed to be inefficient [9]. Other VRC architectures were proposed by different research groups for developing reconfigurable architectures while trying to keep the architecture flexible and simple [10][11][12][13][14].

The idea adopted here is to use VRC as a second reconfiguration layer built on top of the FPGA. This VRC is generally constructed from an array of programmable elements (PEs). Each PE can easily be configured to connect to others elements and/or to circuit inputs or outputs.

Routing is accomplished via multiplexers while configuration memory of the VRC is typically implemented as a register array. Data loaded into the configuration memory is used to control the routing through multiplexer selection. Based on VRC, users can design any desired configuration according to the application on hand. Furthermore, the VRC is described in HDL which is independent of the target platform. Thus, makes the design portable to other different reconfigurable platforms.

The rest of the paper is organized as follows: Section 2 introduces a case study of designing a sequence detector using the proposed approach and section 3 describes the implementation environment. Experimental results are shown in Section 4. Section 5 concludes the paper with suggestions of future work

2. Case Study: A sequence Detector

Finite State Machines (FSM) are typical examples for sequential logic circuit design. Therefore, the chosen case study of FSM is a sequence detector that has one input, one output and three internal states. It is capable of detecting two overlapping bit occurrences of 101 and 100. Its Mealy state transition diagram is shown in Figure 1.

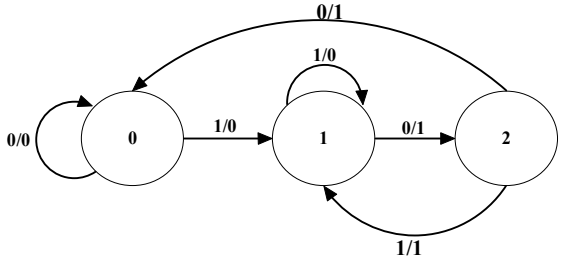

Figure 1. State transition diagram for sequence detector

2.1. Virtual reconfigurable architecture

The hardware implementation of the VRC is coded in VHDL. It has one input *Sin,* one output *Sout,* and 24 configuration bit inputs. This VRC architecture is viewed as a programmable element that implements our FSM via these configuration bits. Once a target configuration is obtained, *Sin* and *Sout* are used as the input and output of the evolved sequential circuit respectively. The VRC is a simple configuration that consists of three 8-to-1 multiplexers and two flip-flops as shown in Figure 2.

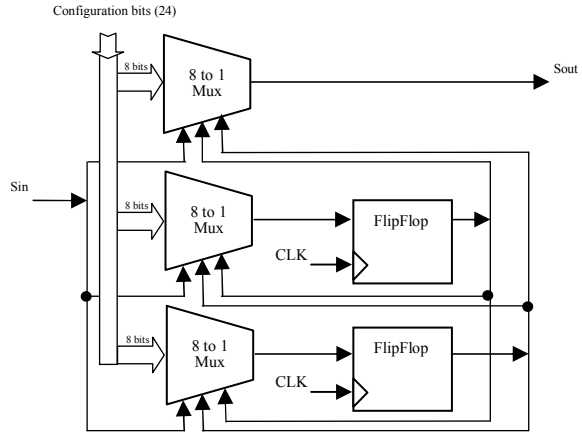

Figure 2. Virtual reconfigurable circuit architecture

2.2. Evolution process

The GA runs on the embedded PowerPC (PPC) processor to generate a configuration chromosome. This configuration chromosome is used to configure the VRC architecture that implements the function of the sequence detector. The PPC also has a random generator that generates test bit-sequences. These sequences are applied simultaneously to both the VRC and to a reference design for the FSM, coded in C. The corresponding output sequence of this FSM is stored in a Random Access Memory (RAM). Both output sequences, from the RAM and VRC, are compared for equality in the comparator, a simple XOR circuit, to generate a fitness value. In fact, the number of *ones* in this value indicates how good the match between the two sequences is. For example, for a bit sequence of *n* bits and a perfect match occurs, the fitness value would be *n*. If this fitness value is not perfect, it is used to guide the GA (through mutation and crossover) to generate a new chromosome for a new VRC configuration and the cycle repeats until a perfect match is found or after a certain number of iterations is reached. The evolved architecture becomes the final

Authorized licensed use limited to: Boise State University. Downloaded on October 12, 2009 at 11:32 from IEEE Xplore. Restrictions apply.

implementation of the FSM. This evolution process is depicted in Figure 3.

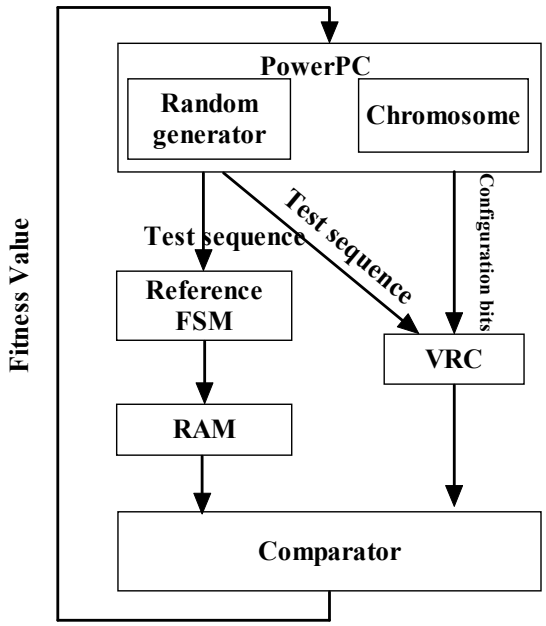

Figure 3. Sequence detector evolution process

3. Implementation environment

The evolvable system is implemented on a Xilinx™ XUPV2P Virtex-II Pro FPGA prototyping board [15], shown in Figure 4. This device contains 13696 slices, 428Kbit distributed RAM, 428Kbit multiplier blocks, 2448 block RAMs, 8 Digital Clock Managers (DCMs), 2 PowerPC RISC embedded processors and 8 multi-gigabit transceivers. The maximum processor speed is 300MHZ. The development board has a wealth of resources among which an RS232 port that facilitated the communication between the PC and the FPGA chip via an on-chip UART IP core module. Configuration and debugging of the FPGA is done through USB port.

The evolvable system is built using the Xilinx™ Embedded Development Kit (EDK) software (version 10.1) [16]. The implementation can be viewed as a software/hardware co-design. As described previously, the virtual reconfigurable circuit is written in VHDL and communicates with the PowerPC microprocessor using an On-chip Peripheral Bus (OPB) bus structure. The C code ran on the processor to generate the configuration chromosome used as a configuration bit-stream for the FPGA.

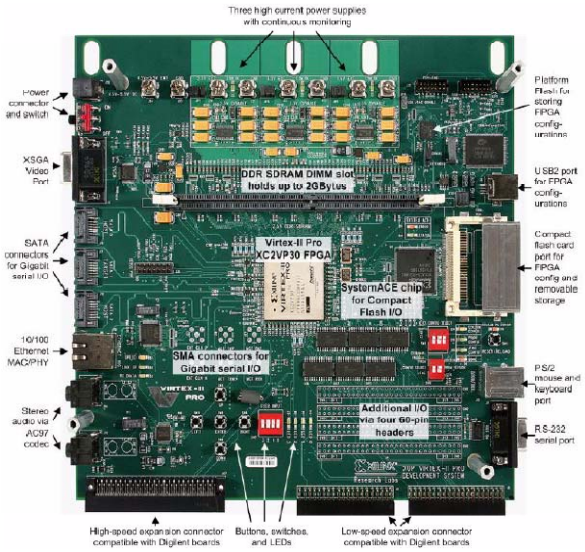

Figure 4. XUPV2P board

4. Experimental results

FPGA Device resource utilization were collected and analyzed to evaluate the effectiveness of the design. Table 1 shows the amount of resources used by the case study implementation.

Table I. Device utilization of EHW system

Resources	Used	Available	Percent
Slices	1138	13696	8.3%
Slice Flip Flops	1078	27392	3.9%
4 input LUTs	1560	27392	5.7%

A maximum of 8.3% of FPGA's total resources were used. This indicates that there is enough space for more complex and much larger designs.

The system functionality is verified using a clock frequency of 100MHz for the PPC and 50MHz for the rest of the system. The allowable maximum clock frequencies, as defined by FPGA specification, are 300MHz and 100MHz respectively. We can conclude that it is possible to accommodate applications requiring faster higher clocking frequencies.

The parameters used by the Genetic Algorithms, determined experimentally, to generate the FSM for this particular case study are:

- Population size is 256
- Maximum number of generation is 2000
- Crossover rate is 0.7
- Mutation rate is 0.04
- Number of runs is 10

Since a large crossover rate and small mutation rate were used, the genetic algorithm found a global optimization. Two test bit-sequences were used to test

Authorized licensed use limited to: Boise State University. Downloaded on October 12, 2009 at 11:32 from IEEE Xplore. Restrictions apply.

the proposed design: a 200 and 300 bit-sequences. Evolution results for the 10 runs of the two sequences are shown in Figures 5 and 6 respectively. A solution was found after a different number of generations for each run with a probability of 90% or greater. If a mach does not occur, 2000 generations are used to terminate the evolution process. Preliminary results of an improved implementation of the GA algorithm guarantees the ability of finding a solution for longer bit sequences each run.

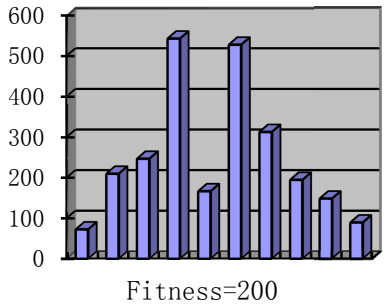

Figure 5. Number of generations vs run number for the 200 bit-sequence

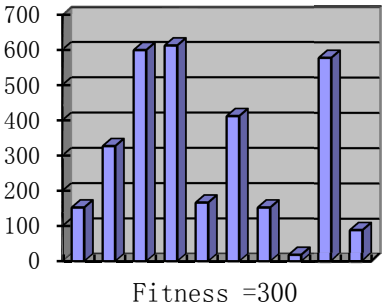

Figure 6. Number of generations vs run number for the 300 bit-sequence

5. Conclusion

This paper proposes a new innovative approach to evolve small sequential logic circuits using Intrinsic Evolution with Virtual Reconfigurable Circuit design technique. Experimental results of a case study, a sequence detector, show that a sequential circuit can be evolved 90% of the time when a genetic algorithm is used with intrinsic evolution. The evolved design behaves as if the same circuit is designed using conventional HDL methods. The case study also shows that the evolution time is reduced using intrinsic evolution compared to other currently available hardware evolution methods [17]. Future work includes applying this design technique to larger sequential logic circuits.

6. References

[1] H. de Garis, "Evolvable Hardware: Principles and Practice. Communications of the Association for Computer Machinery," *CACM Journal*. Aug. 1997.

[2] X. Yao and T. Higuchi, "Promises and Challenges of Evolvable Hardware," *IEEE Trans. Systems, Man and Cybernetics*, Part C, vol. 29, Feb.1999, pp. 87-97.

[3] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Pub. Co., Reading, Mass., 1989.

[4] B. Ali, A.E. Almaini and T. Kalganova, "Evolutionary Algorithms and Their Use in the Design of Sequential Logic Circuits," *Genetic Programming and Evolvable Machines*. Vol. 5, No. 1, Mar. 2004, pp. 11-29.

[5] N. Nedjah and L.D.M Mourelle, "Mealy Finite State Machines: An Evolutionary Approach," *CIC International*, vol. 2, No. 4, Aug. 2006, pp. 789-806.

[6] Shanthi, A. P. and Singaram, L. K.. "Evolution of Asynchronous Sequential Circuits," *NASA/DoD Conference on Evolvable Hardware*, 2005. pp. 93-96.

[7] P. Chongstitvatana and C. Aporntewan, "Improving Correctness of Finite-State Machine Synthesis from Multiple Partial Input/Output Sequences," *NASA/DoD Workshop of Evolvable Hardware*, 1999, pp. 262–266.

[8] A. Thompson, "Evolving Electronic Robot Controllers that Exploit Hardware Resources," Proceeding *3rd European Conference on Artificial Life (ECAL)*, vol. 929, 1995, pp. 640–656.

[9] Andres Upegui, *Dynamically Reconfigurable Bio-inspired Hardware*, Ph.D thesis, 1996.

[10] L. Sekanina, "Virtual Reconfigurable Circuits for Real-World Applications of Evolvable Hardware," *Evolvable Systems: From Biology to Hardware*, vol. 2606, Springer-Verlag, 2003, pp. 186–197.

[11] L. Sekanina, *Evolvable Components from Theory to Hardware Implementations*, Springer, Berlin, 2004.

[12] L. Durbeck and N. Macias, "Defect-tolerant, Fine-Grained Parallel Testing of a Cell Matrix," *Proceedings of SPIE ITCom* , 2002, pp. 71-85.

[13] Pauline C Haddow and Gunnar Tufte, "Bridging the Genotype-Phenotype Mapping for Digital FPGAs," *Proceedings of the third NASA/DOD Workshop on Evolvable Hardware*, 2001, pp. 109-115

[14] P.C. Haddow and G. Tufte, "An Evolvable Hardware FPGA for Adaptive Hardware," *Proceedings of the 2000 Congress on Evolutionary Computation*, 2000, pp.553-560.

[15] Xilinx™ Corp, *Xilinx University Program Virtex-II Pro Development System Hardware Reference Manual*, Apr., 2008

[16] Xilinx™ Inc., *EDK Concepts, Tools, and Techniques Reference Manual (V10.1*), Sept., 2008

[17] Garrison W. Greenwood and Andrew M. Tyrrell, *Introduction to Evolvable Hardware: A Practical Guide for Designing Self-Adaptive Systems*, Wiley-IEEE Press, 2006.

Authorized licensed use limited to: Boise State University. Downloaded on October 12, 2009 at 11:32 from IEEE Xplore. Restrictions apply.