Proceedings of the 6th WSEAS Int. Conf. on EVOLUTIONARY COMPUTING, Lisbon, Portugal, June 16-18, 2005 (pp302-307)

Design Of High Performance Synchronous Digital Circuits For Evolvable Hardware (EHW) Application Using Genetic Algorithm

J.BHARATHI, P.SAKTHIVEL
Department of Electronics and Communication Engineering
Anna University
College of Engineering, Guindy, Chennai 600 025
INDIA

Abstract: This paper presents both the genetic algorithm and the hardware evaluation environment. Results reveal that the genetic algorithm is able to exploit the flexibility provided by novel chromosome architecture, and utilize a combination of primitive gates and macro components from a component library, in order to produce circuits, which operate well within timing restrictions. Evolvable Hardware (EHW) is a new scheme inspired by natural evolution, for designing hardware systems. 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. The paper introduces this new approach and outlines how it can be applied for hardware design of the serial bit pattern recognizer. A distinct feature of GA is to directly evolve and evaluate circuits in a HDL within novel environment termed Virtual Chip (VC). The VC evolves circuit structures within a HDL, detailed simulation of each circuit is possible with any technology specific component library. This feature allows accurate analysis of timing and area. The GA is able to exploit the flexibility provided by novel chromosome architecture, and utilizes a combination of primitive gates and macro components from a component library in order to produce circuits, which operates well within timing restrictions.

Key-Words: Chromosome, EHW, GA, HDL, RTL, VC, Verilog

1 Introduction

Advances this decade in silicon technology have enabled design engineers to examine new methods of generating circuit designs. One such method has become known as EHW, which considers the automated design of digital systems using both software simulation and programmable hardware technologies. Although a range of evolutionary algorithms have been applied to EHW applications including Genetic Programming, Evolutionary Strategies, and Evolutionary Programming, the most dominant approach involves the use of GA. Automated circuit design attempts to re-define the methodology by which electrical circuits are developed. Traditional techniques utilize a top down or compartmentalized design methodology by which complex systems are broken down into small sub-systems and assigned to a number of design groups. EHW, inversely approaches the design problem as a whole system generating a 'black box' of the completed circuit. This technique is referred to as a bottom up design methodology. The only information an EHW framework has is that presented to it by the design engineer.

Circuits generated via evolvable hardware are evaluated by one of two methods: extrinsic evaluation (software simulation), and direct intrinsic evaluation by which a circuit is transferred directly into silicon and then evaluated. Intrinsic evaluation has become feasible due to recent advances this decade in programmable hardware technology such as PLDs and FPGAs (Programmable Logic Devices and Field Programmable Gate Arrays). The successful generation of digital circuits using both extrinsic and intrinsic evaluation has demonstrated the

potential of EHW for automated circuit design. It has also highlighted a number of inherent problems, particularly associated with intrinsic evaluation, and raised a number of questions as to how current EHW techniques can be further improved.

Evolvable hardware for automated digital design favors software based, or *extrinsic* evaluation, due to the simplicity of its implementation and the ease in which evolved circuits can be examined once a solution is found. Using this approach only the final solution is downloaded onto a reconfigurable device. The majority of frameworks which employ extrinsic evaluation use a technology independent net-list to model a digital circuit undergoing evolution, although representations which more closely model the characteristics of a particular hardware platform have also been presented.

Section 2 deals with the EHW and GA. Section 3 deals with the serial bit pattern recognizer. Section 4 deals with simulation of the pattern recognizer. Section 5 deals with conclusion.

2 EHW and GA

2.1 EHW

Earlier we have seen a limit in the size of hardware devices. However, we may very well soon see a limit in designability. That is, designers are not able to apply all the transistors in the largest integrated circuits becoming available. To overcome this problem, new and more automatic design schemes would have to be invented. One such method is Evolvable HardWare.

Instead of manually designing a circuit, only input/output-relations are specified. The circuit is automatically designed using an adaptive algorithm, which is illustrated in Fig. 1.

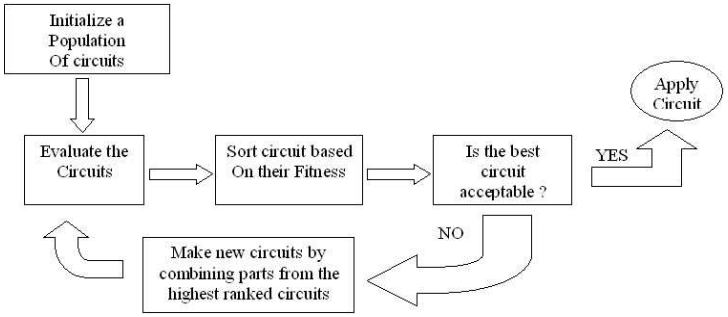

Fig. 1. The algorithm for evolving circuits.

In this algorithm, a set (population) of circuits - i.e. circuit representations, are first randomly generated. The behavior of each circuit is evaluated and the best circuits are combined to generate new and hopefully better circuits. The evaluation is according to the behavior initially specified by the user. After a number of iterations, the fittest circuit is to behave according to the initial specification.

2.2 GA

The most commonly used evolutionary algorithm is genetic algorithm (GA). The algorithm - which follows the steps described above, contains important operators like crossover and mutation for making new circuits.

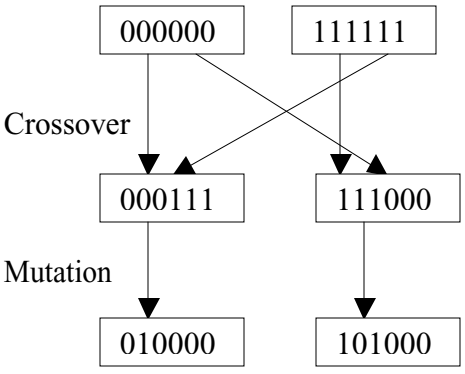

Fig. 2. The genetic algorithm operators.

Each individual in the population is often named chromosome or genotype and is represented by an array of bits. Each bit in the array is often called a gene. Thus, each chromosome contains a

representation of a circuit with a set of components and their interconnections. In crossover, the parameters of the pairwise selected circuits are exchanged to generate - for each couple, two new offspring - preferably fitter than the parents. Further, the best circuit may as well be directly copied into the next generation (called elitism). Mutations may also occur and involves inverting a few genes in the chromosome. This makes the chromosomes slightly different from what could be obtained by only combining parent chromosomes.

When the number of offspring circuits equals the number of circuits in the parent population, the new offspring population is ready to become the new parent population. The original parent population is deleted. Thus, one loop in Fig. 1 is named one generation. Randomness is introduced in the selection of parents to be mated. Not only the fittest circuits are selected. However, the probability of a circuit being selected for breeding decreases with decreasing fitness score.

In addition to the evolutionary algorithm (GA), a circuit specification would have to be available.

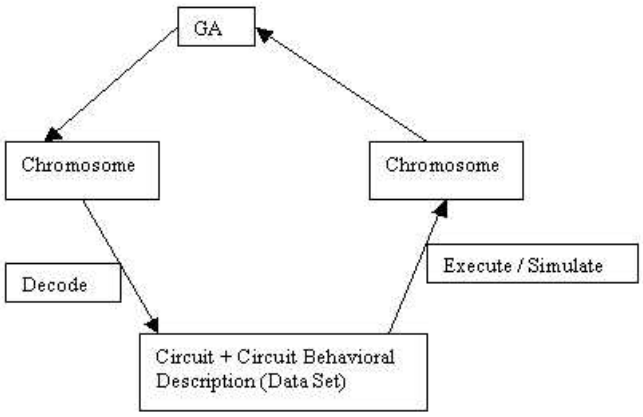

Fig. 3. The cycle of evolving a circuit.

This is often a set of training vectors (input/output mappings) assembled into a data set. The operation of GA together with the data set are given in Fig. 4. The most computational demanding part of GA is usually the evaluation of each circuit - typically named fitness value computation. This involves inputing data to each circuit and computing the error given by the deviation from the specified correct output.

2.3 Encoding a Circuit Within a Chromosome

Genetic algorithms for evolvable hardware are used to develop chromosomes, which then encode the functional description of a given circuit. As with many applications, which utilize genetic algorithms, the resulting circuit is termed a phenotype, as it comprises numerous smaller logic cells or genotypes. The terminologies used are designed to reflect the conceptual similarity between genetic algorithms, natural evolution, and genetics.

The genetic algorithm presented here uses a permutation-based encoding of fixed-length. As such only a specified number of logic elements are presented to the framework. From this, the desired circuit functionality must be generated. Using a fixed-length encoding is standard practice and is one of the main restrictions within which a genetic algorithm operates.

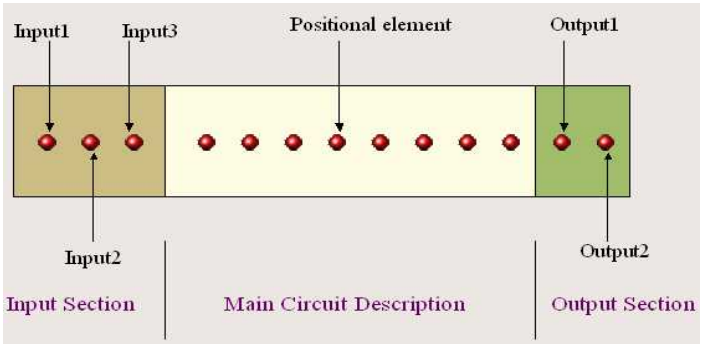

Fig. 4. Chromosome structure defining sections for specific circuit description.

Specific sections of each chromosome are reserved for describing the inputs and outputs required for the desired circuit. Logic elements are referenced by position within the chromosome. Figure 4 displays the relative location of each encoded

section. Circuit inputs are encoded in the first section of chromosome.

The encoding ensures that the number of inputs and outputs described by a chromosome remains consistent after operations such as crossover.

2.4 Connecting Cell Within the Chromosome

Each genotype (logic element) in a circuit is allocated a specific position within the corresponding chromosome. The type of logic cell at any given position is initially determined randomly, however cells can be allocated different positions after initialization through manipulation by genetic operators.

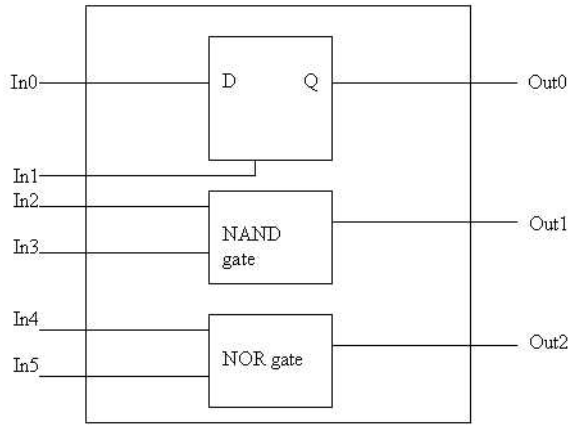

Fig. 4. Encoding describing a macro element and its connectivity, a character in a Chromosome.

Each possible circuit solution for a given task lies within a search space. The search space is defined by the number of different building blocks presented to the framework, the number of logic elements used to generate the circuit, and the application for which the circuit is being evolved. Evolutionary algorithms are employed within EHW as they provide a non-heuristic investigation of what is potentially a very large search space. Successful solutions are often made more difficult to find as the output response must be exact, for instance as part of a sequence of operations such as memory mapping. This differs from the other types of circuit, which instead approximate a specified analogue transfer function.

2.6 Parameters and Constraints

Several constraints are imposed during initialization of the genetic algorithm. Some are designed to eliminate contentious circuit configurations, while others are a result of the evolutionary algorithm employed. Initial global parameters are entered by the designer and are as follows:

- Number of inputs and outputs required for the desired circuit
- Definition of input and output vectors upon which evaluation takes place, and which describe circuit functionality
- Number of logic elements within a chromosome used to create the circuit
- The maximum number of possible fan-outs per cell output (user defined)
- Population size, defining the number of circuit solutions concurrently evolving within the search space

So as to optimize cell connectivity within a fixed-length circuit encoding, each output pin on a logic element is randomly allocated a fan-out ranging between one, and a user defined maximum.

A summary of parameters applied to the genetic algorithm is as follows:

- Number of characters in a chromosome is six
- Population size is fifty
- Number of generations is ten
- Fitness of the individual in a population is calculated by the following:

x = Decimal value of the binary encoding of the individual (1)

Fitness value, $f(x) = x^2$ (2)

Fitness value of each individual is compared with the expected fitness value to evolve the corresponding circuit.

2.6 The Virtual Chip Environment

Verilog is one of two dominant languages for describing digital electronic systems. It is a technology independent environment and describes the structure of a digital system by describing subsystems (logic elements) and how they are interconnected. In addition, circuit descriptions can then be accurately simulated without the need for hardware prototyping. After successful testing a circuit can then be synthesized to provide a technology specific net-list, ready for transfer onto silicon. Almost all technology vendors provide models for logic elements within component libraries.

The Virtual Chip has been designed to provide an automated digital design procedure. Within this framework a novel genetic algorithm is used to evolve digital circuits. Its simulated environment evolves the structure of a circuit directly within the Verilog language. This is performed within a specially designed testbench. It is this testbench which instantiates and interconnects all logic elements within each chromosome, used to describe a specific circuit solution. Evaluation is performed by instantiating and simulating all circuits described within a population of chromosomes, as if they were being implemented within a single reconfigurable chip.

The Virtual chip is a fusion of C code and Verilog. The genetic algorithm itself is executed in C and generates the Verilog required to instantiate each chromosome encoded circuit. After a circuit has been successfully evolved it is then passed through a CAD tool for optimization. Figure 5 displays the execution flow and coding format of the VC EHW framework.

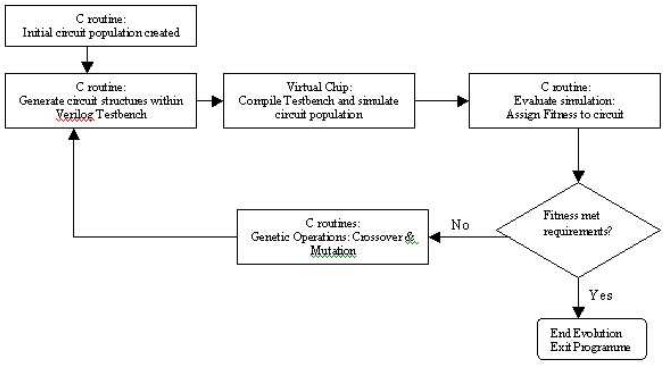

Fig. 5**:** Execution flow and coding format of the genetic algorithm and VC evaluation environment.

Due to the implicit parallelisation of the VC environment, the entire population is compiled, and simulated as one entity. This differs from most standard approaches, which evaluate each individual solution sequentially. As a result, within the VC environment, an entire population of fifty individuals can be simulated and evaluated at a time.

3 Serial Bit Pattern Recognizer

To design a pattern recognizer, which detects a serial bit pattern of six consecutive ones. The pattern recognizer detects the pattern “111111” and its output goes to “1” for one clock duration and it goes to “0” in the next clock. If the pattern is not detected the output is “0”.

Input vectors:
Serial input - Input serial bit pattern
Clock - Clock input
Reset - Input to reset the pattern recognizer

Output vector:
Output – Pattern recognizer output

4 Simulation of the Evolved Serial Bit Pattern Recognizer

Figure 6 shows the Active HDL simulation of the expected output. Figure 7 shows the output of the circuit evolved in the VC.

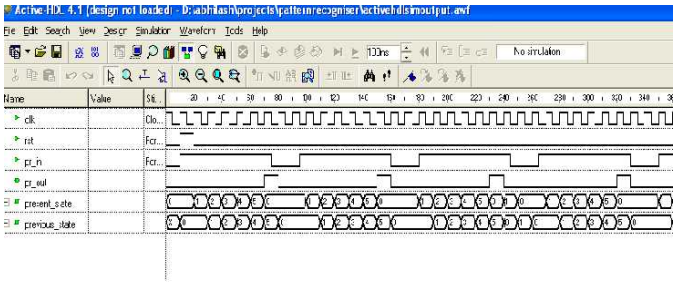

Fig. 6**:** Active HDL simulation of the expected output of the required circuit.

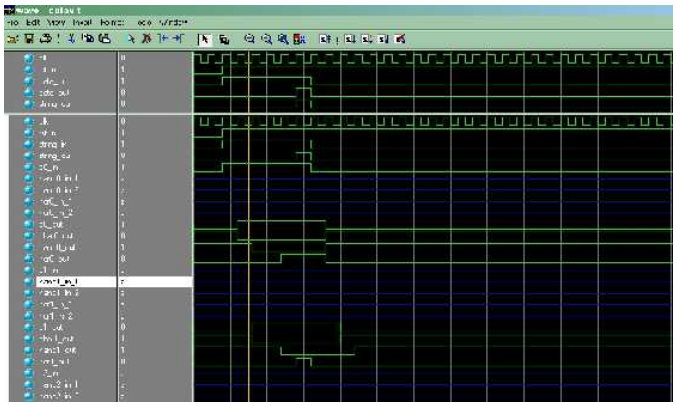

Fig. 7**:** ModelSim simulation of the circuit evolved in the VC.

5 Conclusion

A novel genetic algorithm for the automated design of serial bit pattern recognizer has been presented. The algorithm also describes a flexible chromosome encoding designed to fasciculate complex circuit structures with a minimal number of logic elements. The environment in which circuits are evaluated has also been presented. The architecture was autonomously generated using genetic algorithm and compared with equivalent architectures developed using conventional high-level design methodologies. The use of a genetic algorithm leads to a significant saving in design time.

References:

[l] GOLDBERG, D. E., Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, 1989

[2] ARSLAN, T., HORROCKS, D. H., AND OZDEMIR, E., 'Structural cell-based VLSI circuit design using a genetic algorithm', in IEEE International Symposium on Circuits and Systems, pp. 308-3 11, Atlanta, USA, 1996

[3] THOMPSON, A., LAYZELL, P., ANDZEBULUM, R.S., 'Explorations in design space: Unconventional electronics design through artificial evolution', IEEE Transactions on Evolutionary Computation, 3(3), 267-196, September 1999

[4] LEVI, D. AND GUCCIONE, S . A., 'Geneticfpga: Evolving stable circuits on mainstream fpgas', in A. Stoica, D. Keymeulen, and J. L. (Eds.), editors, In Proceedings of the First NASMDOD Workshop on Evolvable Hardware, pp. 12-17. IEEE Computer Society Press, Los Alamitos, July 1999

[5] BEN. I HOUNSELL AND TUGHRUL ARSLAN 'A Novel Genetic Algorithms for the Automated Design of Performance Driven Digital Circuits' Department of Electronics and Electrical Engineering, The University of Edinburgh, King's Buildings, Mayfield Rd, Edinburgh EH9 3JL