Evolutionary design of combinational logic circuits using VRA processor

Jin Wang\textsuperscript{a)} and Chong Ho Lee

Department of Information & Communication Engineering, Inha University,
253 YongHyun-Dong, Nam-Gu, Incheon 402–751, Korea

a) wangjin@liips@yahoo.com.cn

Abstract: This paper presents a virtual reconfigurable architecture (VRA)-based evolvable hardware for automatic synthesis of combinational logic circuits. The VRA processor is implemented on a Xilinx FPGA and works through two-stage evolutions: (1) finding a functional circuit, and (2) minimizing the number of gates used. To optimize the algorithm performance in the evolutionary process, a self-adaptive mutation rate control scheme is introduced. The efficiency of the proposed methodology is tested with the evolution of a 3-bit multiplier. The obtained results demonstrate that our approach improves the evolutionary design of electronic circuits in terms of quality of the evolved circuit as well as the computational effort.

Keywords: evolvable hardware, self-adaptive mutation rate control, two-stage evolution, VRA

Classification: Integrated circuits

References

1 Introduction

As an alternative to conventional specification-based circuit design methods, evolvable hardware (EHW) has been introduced as a methodology for the design automation of combinational logic circuits [1, 2, 3, 4, 5]. In this paper, we present a virtual reconfigurable architecture (VRA)-based intrinsic evolvable system to synthesize combinational logic circuits at gate level. To reduce the production and running costs of the evolved circuits, we tackle the synthesis procedure by two-stage evolution which produces a fully functional circuit with minimized number of active gates in the layout.

Mutation probability settings have a major impact on the success and efficiency of hardware evolution [5, 6]. In evolutionary design of combinational logic circuits, most researchers [1, 2, 3] define the mutation rate parameters as constants for simplicity reason. However, to attain better system performance, the mutation rates are generally required to be optimized as the evolutionary algorithm (EA) progresses [4, 5, 6]. On the other hand, the process of parameter tuning to seek an optimized mutation rate is time-consuming and the tuned result is viable only for some very limited applications. In this work, we employ an efficient and hardware feasible self-adaptive mutation rate control (SAMRC) mechanism [6] to enhance the performance of the proposed VRA processor. This scheme allows EA to adjust its initial mutation parameters and to optimize its performance during running time.

The main purpose of this paper is to show the effectiveness and applicability of the VRA-based EHW. We perform the experiments on evolving 3-bit multipliers which is generally employed as a benchmark problem by the EHW community [1, 2, 3, 4]. Experimental results show that the VRA processor can automatically synthesize logic circuits. The quality of the evolved circuits and the computational effort are superior to other existing schemes.

2 The proposed VRA processor

An attractive feature of EHW is that it may produce optimized or innovative designs unlike traditional human designers. This process doesn’t rely on the designer’s knowledge and experiences. In the contemporary literature, both extrinsic EHW and intrinsic EHW have been employed to synthesize electronic circuits [1, 2, 3, 4, 5, 6]. As a feasible and efficient approach to intrinsic EHW, virtual reconfiguration technique has been introduced by several researchers [2, 6]. In this paper, a virtual reconfiguration-based intrinsic EHW, named VRA, is introduced for the evolutionary design of combinational logic circuits.

2.1 Phenotype and genotype description

The proposed VRA, which is described in HDL, is a second reconfiguration layer developed on top of common FPGAs. As illustrated in Fig. 1, the VRA consists of a function element (FE) array and an EA. The top function of the FE array is configured using the chromosomes generated by EA, which is implemented on the same FPGA chip.
Fig. 1. VRA and its genotype-phenotype mapping

The method of encoding a circuit into a genotype presented in this work is by a development and modification of traditional Cartesian Genetic Programming (CGP), a computational model that has been widely adopted for the evolutionary design of combinational logic circuits [2, 3, 5]. The phenotype of CGP is a connected logic cell network in a two-dimensional cell array. In VRA, a revised two-dimensional FE array that includes more connection restrictions than a standard CGP is employed. As shown in Fig. 1, a circuit is presented by a fixed-size array of cells, which are placed in a grid of 6 columns and 12 rows for evolving 3-bit multipliers. Every cell in this array represents a 2-input and 1-output FE and each FE executes one of eight logic functions predefined in a functional block (see Fig. 1). Each FE has a single bit function identification \((F1)\) port to denote its current status, wherein \(F1 = 1\) hints a FE is inactive (we consider the FE executes F1, or F3 function defined in Fig. 1 as an inactive FE). All connections of the FE are feed-forward. Compared to CGP, where each FE may obtain its inputs from the external inputs of the FE array and/or the FE output in its preceding layers, each FE in the proposed array is limited to connect to the FE outputs from its immediate preceding layer.

The genotype is presented as a fixed-length linear binary bit string in our approach. To cooperate with the reconfiguration of the FE array which is performed column by column, an intact chromosome consists of 6 configuration bits strings \((cbs)\) and each of them defines the input connections and the active functions of its corresponding FE column. Each \(cbs\) undergoes independent mutation operations. To achieve self-adaptive mutation rate, rather than fixed one, the mutation rate control parameters are also encoded into chromosome as 6 \(\times 2\) bits additional genes (gray blocks in Fig. 1). Each 2-bit additional gene selects the mutation probability \((P_m)\) of its corresponding \(cbs\) according to the match table shown in Fig. 1.
2.2 Evolutionary algorithm with SAMRC

The EA employed in our VRA processor is according to the 1+λ evolutionary strategy with only selection and mutation operators, where λ = 2. SAMRC is realized using the idea of adjusting the parameters of \( P_m \) by evolving them [6]. When the EA runs, an initial population of 2 individuals is randomly generated. Once the fitness of each individual is calculated, the best chromosome including 6 \( cbs \) and their corresponding 6 \( \times \) 2 bits additional genes is selected as parent chromosome. The population of next generation is generated by mutating all \( cbs \) in parent chromosome using their corresponding \( P_m \) which are defined according to the mutants of the 6 \( \times \) 2 bits additional genes in parent. The additional genes are mutated separately with a constant mutation rate 3.125%. The new population consists of the parent chromosome and its 2 mutants. Thus, the \( P_m \) parameters for each \( cbs \) also undergo an evolutionary process by mutation and selection, and may adjust themselves dynamically.

In this work, our goal is to produce a fully functional circuit with the minimum number of active logic gates. A multi-objective fitness function is employed in our approach and EA works in two stages: 1) design stage; 2) optimization stage. In design stage, only the valid circuit outputs are taken into account, and the EA works for finding a fully functional circuit. The following formula is used to compute the fitness of an individual:

\[
f_1 = \sum_{i=0}^{2^n-1} \sum_{j=0}^{m-1} (w_{ij} \oplus v_{ij})
\]  

Eq. (1)

In Eq. (1), \( m \) and \( n \) are the number of bits of outputs and inputs of a circuit, respectively, \( w_{ij} \) is the \( j \)th bit of the output for the \( i \)th input pattern, \( v_{ij} \) is the desired output bit defined by truth table. The \( f_1 \) increases by one for every matched output bit. In the 3-bit multiplier case, it includes \( 2^6 = 64 \) six-bit output vectors and the maximal \( f_1 \) is \( 64 \times 6 = 384 \).

Once a correct circuit appears, the EA operates in the optimization stage. In this stage, EA tries to reduce the number of active gates as many as possible in a fully functional circuit and a chromosome is evaluated using the following fitness function:

\[
f_2 = f_1 + f_G
\]  

Eq. (2)

\[
f_G = \sum_{k=1}^{l} F I_l
\]  

Eq. (3)

In Eq. (3), \( l \) is the total number of FEs in the FE array. \( f_G \) counts the number of inactive FEs in a circuit encoded by the chromosome. In optimization stage, \( f_2 \) is calculated based on \( f_1 + f_G \) only if the function of a circuit is correct. Otherwise, \( f_2 \) is equal to \( f_1 \).

3 Results and discussion

The VRA processor was designed using VHDL. The detail of the FPGA implementation of VRA can be found in [6]. After simulations with ModelSim,
the design was synthesized using Xilinx ISE 6.3 to Virtex xcv2000E FPGA, which is available on a Celoxica RC1000 PCI board. According to the synthesis reports, the maximum FPGA clock frequency attained was 96.108 MHz. However, the real running speed was limited to 33 MHz for easier synchronization with current version of PCI interface employed by the prototyping board.

With the addition of SAMRC, we define the length of additional genes \(= 6 \times 2\) bits. \(P_m\) is chosen among 0.2%, 0.4%, 0.8% or 1.6% (also shown in Fig. 1). For comparison, traditional EA with constant mutation probabilities are also tested with the VRA processor. For each case, we perform 100 EA runs individually. The EA terminal condition is defined as: the predefined number of generations \((2^{26})\) reaches. The execution time for one EA run in VRA processor is 260.3 seconds.

Table I. Comparison of results for evolving 3-bit multipliers with various schemes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>(P_m) (%)</th>
<th>Device Cost (slices)</th>
<th>Design stage</th>
<th>Optimization stage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>avg. gen.</td>
<td>avg. gates</td>
<td>suc. rate</td>
</tr>
<tr>
<td>SAMRC</td>
<td>0.2-1.6</td>
<td>4681</td>
<td>5321766</td>
<td>52.97</td>
</tr>
<tr>
<td>Constant mutation rate</td>
<td>0.2</td>
<td>4427</td>
<td>7588148</td>
<td>53.90</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>4428</td>
<td>8500657</td>
<td>52.98</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>4444</td>
<td>14661452</td>
<td>53.46</td>
</tr>
<tr>
<td></td>
<td>1.6</td>
<td>4653</td>
<td>46242803</td>
<td>52.53</td>
</tr>
<tr>
<td>Cheang [1]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Miller [3]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Zhao [4]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

![Fig. 2. An evolved 3-bit multiplier (26 gates, 6 propagation gate delays)](image-url)

The comparisons of the device cost, the average number of generations required (avg. gen.) and the percentage of the successful runs that find the correct circuits among 100 EA runs (success rate) in design stage, the average number of active logic gates (average gates) after design/optimization stage, and the number of active logic gates in the obtained best solutions (best gates) after the optimization stage are summarized in Table I for analysis and discussion. Note that all average results are calculated by taking account
of the success runs among 100 EA runs. The best evolved 3-bit multiplier is shown in Fig. 2.

In Table I, the “avg. gates” values shown under design stage and optimization stage columns present the average qualities of the evolved 3-bit multipliers in each of the two evolutionary stages. These values decrease significantly after the optimization stage when comparing with the values after the design stage. For instance, in the SAMRC case, the “avg. gates” decrease from 52.97 to 31.70. This demonstrates the efficiency of the employed multi-objective fitness function in the two-stage evolution.

From Table I, it may be observed that, in terms of the “suc. rate,” the “avg. gen.” at the design stage, and the “avg. gates” at the optimization stage for evolving 3-bit multipliers, the best constant $P_m$ is 0.8%, 0.2% and 0.8%, respectively. In theory a higher $P_m$ is required at the initial design stage to prevent EA from premature convergence, a lower $P_m$ is desired at the final design stage to improve the quality of elitist solutions and a higher $P_m$ is wanted in the optimization stage to reducing the number of active gates in a full functional circuit. By dynamically adjusting the $P_m$ parameter in the whole evolutionary process, the proposed SAMRC scheme presents very competitive results in the mentioned items above. When compared to various constant mutation rate settings, SAMRC model obtains the best results in terms of the “suc. rate” and the “avg gen.” at the design stage, and near best result in terms of “avg gates” at the optimization stage.

Device cost of the proposed SAMRC model is very close to its competitors. The SAMRC uses only 4,681 slices for evolving 3-bit multipliers. It is about 24% of the 19,200 slices available in the Virtex xcv2000E FPGA. In order to compare the qualities of the 3-bit multipliers evolved by the SAMRC running on VRA processor, the best results of CGP [3], GPP+MLP [1], and multi-objective evolutionary design scheme [4] are also listed in Table I. Our best results are in fact as good as the most efficient 3-bit multipliers evolved by other researchers. Some nonstandard logic gates such as one-input-inverted AND, one-input-inverted OR are employed in Miller’s and Cheang’s circuit layouts (e.g. Miller’s result includes 3 nonstandard gates, and Cheang’s result includes 6 nonstandard gates). To make a fair comparison, these nonstandard gates should be considered as two separate gates. Even though the gate count in our result is higher than in Zhao’s, the length of propagation gate delays is significantly lower than his circuit which includes 9 gate delays. The best 3-bit multiplier evolved by us is also 13.3% more efficient in gates usage than the best one designed by human experts, which includes 30 logic gates [3].

4 Conclusion

A FPGA implemented intrinsic EHW, named VRA, is presented and modified in this article for synthesis of a combinational logic circuit. To improve the performance of EA search, the SAMRC method is executed on our VRA processor. The efficiency and feasibility of the proposed scheme is demonstrated by the evolution of a 3-bit multiplier. Simulations show noticeable
improvements in terms of average number of generations, success rates, and active logic gate counts in comparison with the traditional constant mutation rate-based approaches. The best evolved 3-bit multiplier we got is more efficient than other existing 3-bit multipliers considering the values of active gates and propagation delays.

Acknowledgment

This work was supported by INHA University Research Grant.