Comparison of Technology-Based and State-Based Representations for the Synthesis of Synchronous Sequential Logic from Partial Input/Output Sequence

Chaiyasit Manovit Chatchawit Aporntewan
Prabhas Chongstitvatana
Department of Computer Engineering, Faculty of Engineering,
Chulalongkorn University, Bangkok 10330, Thailand
Tel: (622)218-6982 Fax: (662)218-6955
prabhas@chula.ac.th

Abstract

This paper compares the computational efforts of synchronous sequential logic circuit synthesis from partial input/output sequence by Genetic Algorithm (GA) using different representations; technology-based and state-based. The state-based representation is technology independent. It can be further optimized and benefits from exising synthesis and optimization techniques which work with finite state machines. The resulting state transition diagram can be implemented in any Programmable Logic Device using vendors's CAD tools. The technology-based representation, in this experiment, is a registered PAL device. The result shows, however, that the state-based approach requires greater effort than that of technology-based approach when we permit the same number of maximum states in state machines. The reason is given in terms of the size of GA search space.

1 Introduction

1.1 Genetic Algorithm (GA)

GA [1] is a powerful search and optimization procedure. GA mimics the natural evolution mechanisms. Searching is performed by creating a population. An individual represents one possible solution. Naturally, the offsprings are produced employing standard genetic operators, namely reproduction, crossover and mutation. The selection scheme is employed to select the survivors to the next generation according to their fitness values defined by users. The GA process starts with random population and repeats until the terminated condition is met. (the optimal solution is found or reachs the maximum number of generation)

1.2 Programmable Logic Device (PLD)

PLD is one type of Programmable Integrated Circuits. The digital circuit that has a capability of restructuring itself. There are several numbers of PLDs such as PROM, EPROM, PAL, PLA, GAL and FPGA. Their programmable bits are called *architecture bits* [2]. The architecture bits specify the boolean functions of the logic cells and their interconnections.

1.3 Evolvable Hardware (EHW)

EHW [2, 3] is the hardware that can adapt itself to operate in the dynamic environment. The adaptation can be achieved by evolutionary techniques. The basic idea of EHW is to evolve the architecture bits by GA. The individual represents the architecture bits. Then, GA is performed. The implementations of EHW from different reserch groups [2, 4, 5, 6] are various. But the similarity of various EHW are reconfigurable hardware, evolution and adaptation. The recent classification of EHW is defined in [7].

1.4 Previous Work

This paper is the extension of our previous work [8], which attempts to synthesize sequential circuit of which the behavior description is in the form of partial input/output sequence. There are two related works, [9] and [10]. Fogel evolved the state machine that can predict the next output symbols from one given input/output symbol sequence using his famous technique, Evolutionary Programming (EP) [9]. However, his technique is not efficient enough. Fogel also showed the sequence of primeness (0110101000101...) can not be represented by a finite state machine. Higuchi evolved the architec-

ture bits of GAL from a given input/output sequence using GA. In Higuchi's work, the input/output sequence is generated by feeding random input sequence to a given state diagram and gets corresponding output sequence. The existent optimal solution can be reached in reasonable time. Certainly, some optimal solutions are wrong. They perform correctly on a given input/output sequence but they are not equivalent to the given state diagram.

In [8] [1] , our work extended Higuchi's. Various circuits and input/output sequence length were experimented. The search for a suitable circuit is performed in the technology dependent manner. The GAL structure, which is a registered PAL device, is used as the target technology. We synthesize only the state transition circuit because the part of circuit that produces output is synthesized later by a conventional method. We are able to calulate the appropriate length of an input/output sequence by making an analogy of a traversing through all paths in a state transition diagram to rolling a dice until all faces appeared (this problem is called waiting times in sampling. The appropriate length, denoted by upperbound length, $\mathcal{L}$, can be computed by this formula:

$$\mathcal{L} = E(\mathcal{S}) \times E(\mathcal{I}); E(n) = n(\frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{n}) \quad (1)$$

where

$\mathcal{S}$ is the maximum number of states in the destination hardware

$\mathcal{I}$ is the number of input patterns

This work studies an alternative technology dependent synthesis. We propose a technology independent approach which synthesize a state diagram instead of architecture bits. This approach is advantageous because many analysis and optimization techniques work with state diagrams. For example, the resulting diagram can be further optimized by using CAD tools. It is also possible to implement the final circuit using another technology such as FPGA.

2 The Experiment

We want to compare the results of technology-based synthesis from our previous work [8] with the results of state-based synthesis in terms of the computational effort. The computational effort is defined by Koza [11] and modified in [8] as the number of individuals needed to be produced to yield a correct solution with a satisfactorily high probability, which is 0.99 in this experiment.

The implementation of this work is almost indentical to the previous work except that desired solution is in different representation. In the previous work, an individual represents a wiring status on the GAL structure. On the contrary, in this work, an individual represents a state transition table which is mapped into a bit string by concatenation of all rows. The details of genetic process are the same as in the previous work, which can be summarized as:

1. Select the best 10 individuals from the preceding population by the rank-space method (i.e., fitness evaluation takes both quality rank and diversity rank in account)

2. All possible amoung the individuals pairs in 1 are employed to produce new offsprings by uniform crossover.

3. Each individual in 1 is duplicated with a 5 bits mutated (flipped) and added to the current population.

4. The above steps are repeated until a solution is found or the last permitted generation, 50000, is reached.

For a comparison purpose, both the GAL structure and the state transition diagram are chosen to represent FSMs with the same maximum number of states, which is 16 in this experiment. We consider only circuits in Mealy's model. Selected circuits having less than 16 states are listed below:

1. Frequency Divider :- give an alternate output with the period of 8

2. Odd Parity Detector :- give an output "1" when the number of "1" in inputs is odd

3. Modulo-5 Detector :- give an output "1" when the current input is the $5n^{th}$ "1"; $n \in I^+$

4. Serial Adder :- give the sum of 2 inputs, inputs are feeded from LSB to MSB

The experiment is done using 5 different input/output sequences of the upperbound length for each circuit and running 50 times for one sequence, totally 250 runs for each circuit. The computational effort is computed from all running results.

[1] As this work refers frequently to [8] which contains detailed description of the problem, we made it to be available at http://www.cp.eng.chula.ac.th/faculty/pjw/papers/manovit.pdf

3 The Result and Conclusion

Although the state transition diagram representation is advantageous as mentioned in the introduction, Table 1 shows that the representation in the GAL structure is preferable when the computational effort is concerned. The reason why GAL representation requires less effort can be explained by the calculation of the size of its search space. The search space size for the GAL structure, denoted by SS_{GAL}, and that for the state transition diagram, SS_{STD}, are calculated from:

$$SS_{GAL} = 3^{16\times(s+i)} \tag{2}$$

$$SS_{STD} = 2^{S\times I\times s} \tag{3}$$

where

s is the number of bits that represents a state symbol

S is the maximum number of states, which is 2^s

i is the number of bits that represents an input symbol

I is the number of input patterns, which is 2^i

16 is the number of product terms in the GAL structure

$S \times I \times s$ is the number of bits in a state transition table

Table 1 shows that the search spaces of the GAL representation are smaller than those of the state transition diagram representation except for the frequency divider. Therefore, it requires less effort to explore individuals in the search space. In the GAL representation, we also calculate the ratio of search space size to effort which could help determining the efficiency of the search using each representation. The Space/Effort ratios are comparable for both representations. It is notable that in the synthesis of serial adder using the state transition diagram representation, the ratio is very high compared to that of the GAL representation . It is possible the state-based approach performs effectively in the large search space.

References

[1] D. E. Goldberg. *Genetic Algorithm in search, optimization and machine learning.* Addison-Wesley, 1989.

[2] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, and T. Furuya. Evolving hardware with genetic learning: A first step towards building a darwin machine. In *Proc. of Int. Conf. on Simulation of Adaptive Behavior (SAB'92)*, 1992.

[3] D. Mange and M. Tomassini. *Bio-inspired Computing Machines, towards novel computational architectures.* Presses Polytechniques et Universitaires Romandes, 1998.

[4] J. R. Koza, F. H. BennettIII, D. Andre, M. A. Keane, and F. Dunlap. Automated synthesis of analog electronical circuits by means of genetic programming. In *IEEE Transactions on Evolutionary Computation*, volume 1 No. 2, pages 109–128, 1997.

[5] A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In *Proc. of Int. Conf. on Evolvable Systems (ICES'96)*, 1996.

[6] M. Sipper, M. Goeke, D. Mange, A. Stauffer, E. Sanchez, and M. Tomassini. The firefly machine: Online evolware. In *Proc. of Int. Conf. on Evolutionary Computation (ICEC'97)*, pages 181–186, 1997.

[7] M. Sipper, E. Shanchez, D. Mange, M. Tomassini, A. Perez-Uribe, and A. Stauffer. A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. In *IEEE Transactions on Evolutionary Computation*, volume 1 No. 2, pages 83–97, 1997.

[8] C. Manovit, C. Aporntewan, and P. Chongstitvatana. Synthesis of synchrounous sequential logic circuits from partial input/output sequence. In *Proc. of Int. Conf. on Evolvable Systems (ICES'98)*, pages 98–105, 1998.

[9] D. B. Fogel. *Evolutionary Computation*, pages 75–84. IEEE Press, 1995.

[10] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, and T. Furuya. A parallel architecture for genetic based evolvable hardware. In *Proc. of Int. Joint Conf. on Artificial Intelligence (IJCAI'93), Workshop on Parallel Processing for Artificial Intelligence*, pages 46–52, 1993.

[11] J. R. Koza. *Genetic Programming II: Automatic discovery of reusable programs*, pages 99–106. MIT Press, Cambridge, MA, 1994.

Circuit	Effort		Search Space Size		Log(Space/Effort)	
	GAL	STD	GAL	STD	GAL	STD
Frequency divider 8	440	220	$3^{16\times4}$	$2^{4\times16}$	27.9	16.9
Odd parity detector	1,760	4,070	$3^{16\times(4+1)}$	$2^{4\times16\times2}$	34.9	34.9
Modulo-5 detector	7,018,000	44,701,140	$3^{16\times(4+1)}$	$2^{4\times16\times2}$	31.3	30.9
Serial adder	26,730	643,610	$3^{16\times(4+2)}$	$2^{4\times16\times4}$	41.4	71.3

Table 1: Comparison of Effort, Search Space Size and Space/Effort Ratio