TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.6, June 2014, pp. 4457 ~ 4462
DOI: 10.11591/telkomnika.v12i6.5482

A Power Effective Algorithm for State Encoding

Anping He, Hao Wu, X. Song[1], Jinzhao Wu*[2]
Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis,
Guangxi University for Nationalities, 530006, China
*Corresponding author, e-mail: song20100826@163.com[1], hidrwu@sohu.com[2]

Abstract

Reducing the area and power dissipation of FSM circuit is of significant importance for EDA technology. Many methods are adopted to achieve an effective and fast transformation of FSMs to binary codes, including Genetic algorithm (GA) and others. In this paper, we propose a GA based state assignment of a FSM circuit to gain the minimization of power consumption and area. We modify the traditional mutation to be an ordered operation, which is also a substitution of the crossover that guarantees every new individual owns better fitness than the old one. We test the proposed algorithm with benchmarks, as well as do the comparison with the published; our method saves both power and area dissipation in reasonable computation time.

Keywords: *state assignment, low area, low power, genetic algorithm*

1. Introduction

With the increasing scale of system-on-chip (SoC), the area and power dissipation become the critical concerns in VLSI design, especially for portable computing and personal communication applications. Sequential circuits, playing a major role in VLSI, is characterized by the outputs depending on both the inputs and the past state, e.g., a feedback at the input of the combinational logic. The Finite state machine (FSM) is of a most common way of system level description for sequential logic.

In EDA technologies, the automatic synthesis of FSM to circuit plays a very important role. The encoding procedure of the synthesis called state assignment that maps FSM states to binary codes is essential for the whole synthesis, since it will not only affect circuit area but also power dissipation with different switching activities finally. The problem of finding the state assignment for minimization of power consumption and area belongs to NP hard.

The genetic algorithm (GA) is regarded as an excellent intelligent search algorithm, and also an effective method to achieve fast convergence for some NP-hard problems. Many investigations with GA have been done for state assignments, such as [1-5]. Almaini et al. demonstrated that the GA method produced significantly simpler solutions in [2]. In [1], multi objective GA has been used to optimize both area and power. Chattopadhyay et al. in [3] optimized power only, Xia et al. in [4] optimized both area and power, Chattopadhyay et al. [5] optimized area only. There are other effective methods based on symbolic minimization [6-8].

Other heuristic algorithms have been proposed: Shiue in [9] showed a new comprehensive method consisting of an efficient state minimization and state assignment technique. Goren and Ferguson [10] presented a heuristic for state reduction of incompletely specified FSMs.

In this article, we proposed an enhanced GA based state assignment algorithm. Comparing with the original one, the improvements include: the number of population, removing the crossover operation and improving the mutation operation. Moreover, with this proposed algorithm, each generation has only one individual, which enables the population evolving via mutation instead of crossover. More importantly, the enhanced mutation operation ensures the new individual owns better fitness than the old one. Comparing with others, our algorithm saves more power and area dissipation in a reasonable computation time.

Our paper is structured as follows: in section 2 we introduce the state assignment and the cost function; in section 3, we show our GA algorithm in detail and we show the experiment and comparison of our algorithm in section 4; we concluded in section 5.

Received December 23, 2013; Revised February 17, 2014; Accepted March 3, 2014

2. State Assignment and Cost Functions

In this paper, *state* is a vector, $S(s_1, s_2, \cdots, s_l)$, of a stable FSM/sequential-logic output. The sequential circuit is usually modeled as Mealy FSM (with assumption of outputs relating both input and current state).

Definition 1. A *FSM* is a quintuple $(I, O, S, \lambda, \varphi, S_0)$, where I is the sets of inputs, O is the sets of outputs , S is a set of states, S_0 is initial states, λ is the state-transition function: $\lambda: S \times I \to S$, φ is the output function: $\varphi: S \times I \to O$.

EDA tools try to synthase the FSM to real circuits; it is a fundamental problem of how to encoding the states with binary codes. Different encoding distinguishes the switching activates from one binary code to another, which would finally affect circuit area and power dissipation. On the other hand, the amount of sort of encoding would be huge, e.g., let *n* be the total number states of *S*, it need $s = \lceil \log_2 n \rceil$ ($\lceil\ \rceil$ for upper bound) state variables to encoding the states, then according to [11], the total number of state assignments will be $\frac{(2^s - 1)!}{(2^s - n)! s!}$.

For a concrete state assignment, we can use ESPRESSO [12] to generate the minimized circuit. The number of the generated circuits varies with their encoding methods, so it would be very useful to find a state encoding corresponding to less gates that be with less area and power cosumption consequently.

We evaluate the state encodings by a cost function. With the preliminaries in [1], the cost function of a transition could be computed by the production of the *Hamming distance* and *total transition probability* [13], and the whole cost of a state graph would be the sum of all possible transitions:

$$C = \sum_{s_i, s_j \in S} tp_{s_i s_j} \cdot HD(enc(s_i), enc(s_j)) \tag{1}$$

3. A GA Based Power Effective Encoding

GA is a heuristic optimization algorithm imitating the process of natural evolution, the solution of optimization is seen as individual, which expressed by a variable sequence, called chromosomes. Chromosome is generally expressed as an alphabetic string or numeric one, and then to gain the string is called encoding. While GA processing, it generates a certain number of individuals generally and randomly. In every generation, each individual get its fitness by a specific fitness function. The next generation and composition can be calculated with selecting and breeding operations in terms of current fitness. The mutation exists anywhere that can generate new "child" individuals always by exchanging the position of two genes. Figure 1 shows the pseudo code for GA.

After long term study of the state and coding pattern, we find GA based state encoding algorithm would enhanced more if do some modifications, including the number of population, removing the operation of crossover, modified the way of mutation and the way calculating the fitness. The main idea of our algorithm is that, every generation has one individual only, and in each generation, the optimal individual is generated by mutating the one individual only, then we would get the global optimal individual by comparing all optimal ones. The detailed explanation is shown below.

Initially, we talk about why every generation only has one individual in this algorithm. There is considerable amount of population in traditional GA, and a lot of new individual product by crossover, in this process, the search region for the assignments is enlarged, meanwhile, a sizable majority individual of new generation don’t have better fitness than the individuals of the old generation, besides, this process Consumes a lot of CPU time. So in our algorithm every generation only has one individual, the mutation takes the place of crossover, and after every mutation the new individual must have better fitness than the old. Specific method is as follows.

```
Procedure GA
{
Create an initial population of random genes
        Evaluate all chromosomes
        Repeat
        {
                Select chromosomes with the best fitness to reproduce
                Apply crossover operator
                Apply mutation operator
                Evaluate the new child
                If(child fitness != any existing fitness)
                        Apply termination operator
        } until termination condition
} end GA
```

Figure 1. Proposed Structure for Dynamic Overmodulation in DTC-CSF Based

Our GA based algorithm encodes all FSM states to a individual. Let s be the amount of the states, $b = \lceil \log_2 s \rceil$ bits binary code for each state. In each generation, we initialize an individual randomly and then find a local optimal assignment. The mutation is the primary operation in this algorithm, which includes several swaps of exchanging the position of two genes, after each swap the fitness of the individual would be better or we knock off the swap. In detail, the mutation consists of two loops, the first gene i loops from 1 to n (n is the amount of genes), while the second gene j loops from i to n, if exchanging not exists, exchange their position and then calculate the fitness; if the fitness is better than the previous, then the exchange occurs and then continue the loop; if the fitness is not better than the previous, exchange the two genes back and then continue the loops. In the comparison procedure, the individual with less product terms owning the better fitness, however, if equal, the one with less switching activities would be better. This method reduces a lot of CPU time.

In each generation, we could get a local optimal assignment by some steps of mutation. At the last mutation, if there is no swap occurs, we consider the current individual is the optimal assignment; the generation ends and a new generation would be initialized continually.

The pseudo code of proposed algorithm is as follow (Figure 2, Figure 3 and Figure 4):

We explain the pseudo in detail: In the main function in Figure 2, the loop in the main function generates the local optimal state assignment for each generation; the main function outputs the global optimal assignment by comparing all the state assignments finally. The *get_the_local_optima* function in Figure 3 generates the optimal assignment for one generation, the loop there guarantee that exchanging any of two genes of the individual not result in a better fitness, which calls *Mutations* function in Figure 4. The Mutation function is the main procedure for the whole algorithm, the Mutation function swap two genes by nested loops, which ensures that any two genes can be swapped except for the one has been exchanged.

```
Procedure proposed algorithm //main function
{
        Input (benchmark file, number of generation)
        Loop until generation=0 //generate all the local optimal state assignment
                {
                        Initialize individual (individual)
                        Get the local optima (individual)
                        Output local optima
                        Number of generation - 1
                }
                Output the global individual
}
```

Figure 2. Main Function

```
Get the local optima(individual) //generate the local optimal state assignment
{
        Loop until there is no exchange happen in this loop
        {
                Initial the exchange tag (individual)
                Mutation (individual)
        }
}
```

Figure 3. Get the Local Optima

```
Mutation(individual)//main operator to find the local optimal state assignment
{
        Loop for (i=0 to i=number of genes)//first state i
        {
                Loop for (j=i to j=member of genes)//second state j
                {
                        Exchange i and j gene//exchange position of the i and j
                        Calculate fitness
                        If fitness better mark the tag of the genes//have been exchanged
                        Else exchange back
                }
        }
}
```

Figure 4. Mutation

4. Experimental Result

In this section, we show the experimental results of the proposed algorithm. We implement our algorithm by C++ and Matlab with a 2.9GHz AMD CPU and 1.75GB RAM.

In our algorithm, each generation produces a local optimal solution. Table 1 shows local optimal produced by every generation (the 20 generation in the front), product terms, switching activity and time comsuming, the circuit designers could make a targeted selection form so many local optimal assignments.

Table 1. Experimental Result

generation	1	2	3	4	5	6	7	8	9	10
PT	27	27	23	23	23	24	23	23	23	24
Esw	0.239	0.314	0.267	0.257	0.286	0.311	0.291	0.341	0.285	0.492
Time/sec	3	4	7	9	13	15	17	22	24	26
generation	11	12	13	14	15	16	17	18	19	20
PT	26	25	25	23	26	27	23	23	21	22
Esw	0.334	0.352	0.272	0.329	0.318	0.370	0.288	0.281	0.377	0.326
Time/sec	27	29	31	33	35	37	41	43	45	47

Table 2 shows the comparison of the experimental results of our algorithm and one from MOGA [1] by time requirement, low power consumption and area requirement. From this table, the area requirement is slightly better than MOGA in average, low power is substantially equal to the MOGA, the time for seeking the best results is very less than MOGA.

Table 2. Experimental Result

Benchmarks	In/out/no.of states	This algorithm			MOGA [1]		
		PT	C	Time	PT	C	Time (sec)
bbtas	2/2/6	9	0.502	4sec	9	0.613	1min
					10	0.56	
					11	0.44	
bbara	4/2/10	21	0.377	45sec	22	0.49	8min
					27	0.39	
keyb	7/2/19	44	0.643	7min	46	0.98	3h
					47	0.75	
					55	0.54	
Cse	7/7/16	41	0.442	28min	43	0.39	2h
					49	0.32	
					54	0.30	
S1	8/6/20	44	1.726	1h4min	43	1.37	6h
					53	1.19	
					60	1.04	
S1a	8/6/20	31	1.233	11min	29	1.21	5h19min
					30	1.174	

Table 3 shows the comparison of our algorithm and the algorithms introduced in [14], [15, 16]. On average, our algorithm has fewer product terms and less switching activities in terms of the results from [16]. It also outperforms other methods in both area saving and less power consumption.

Table 3. Experimental Result

Algorithm Benchmarks		This algorithm	[14]		[15]		[16]	
			Result	Improved (%)	Result	Improved (%)	Result	Improved (%)
bbtas	PT	9	-	-	9	0	8	-13
	C	0.502	-	-	-	-	0.815	34
bbara	PT	21	22	5	23	9	24	13
	C	0.337	0.317	-6	-	-	0.459	27
keyb	PT	44	46	4	46	4	48	8
	C	0.643	0.674	5	-	-	1.469	56
Cse	PT	41	43	5	45	9	46	11
	C	0.442	0.355	-26	-	-	0.602	27
S1	PT	44	66	33	68	32	80	45
	C	1.726	1.48	-17	-	-	1.698	-2
S1a	PT	31	-	-	66	53	80	61
	C	1.233	-	-	-	-	-	-

5. Conclusion

In this paper, we presented a FSM state encoding procedure for reducing the power and area consumption of circuits. Our algorithm bases on GA, the enhancements include: removement of the operation of crossover and modified the operation of mutation. With the comparison of the published algorithms, ours shows its strong effects. In short, we regard our algorithm more suitable for area/power optimized realization of FSMs.

Our future work is focused on two directions: firstly, we improve the efficiency of the mutation operation since the most swaps in mutation may be unnecessary; secondly, we should improve the model of power consummation more accurate.

Acknowledgements

This work is partly supported by Bagui scholarship project, the Natural Science Foundation of Guangxi under Grant No. 2011GXNSFA018154 and 2012GXNSFGA060003, the Science and Technology Foundation of Guangxi under Grant No. 10169-1, and Guangxi Scientific Research Project No.201012MS274 , Funded Projects of Innovation Plan for Guangxi Graduate Education No.gxun-chx2013t18 and Guangxi University for Nationalities Project, No. 2012QD017

References

[1] Al Jassani BA, N Urquhart, AEA Almaini. State assignment for sequential circuits using multi-objective genetic algorithm. *IET computers & digital techniques.* 2011; 5.4: 296-305.

[2] Almaini AEA, Miller JF, Thomson P, Billina S. *State assignment of finite state machines using a genetic algorithm.* IEE Proc. Comput. Digit, Tech., 1995; 142(4): 279-286.

[3] Chattopadhyay S, PN Reddy. *Finite state machine state assignment targeting low power consumption." Computers and Digital Techniques.* IEE Proceedings. 2004; 151(1).

[4] Xia Y, Almaini AEA. Genetic algorithm based state assignment for power and area optimisation. *IEE P. Comput. Dig. T.*, 2002; 149(4): 128–133.

[5] Chattopadhyay S, Kumar A, Tewari N. *Flipflop (D/T) and polarity selection for finite state machine synthesis with area overhead constraint genetic algorithm approach.* Proc. International Conference on Recent Trends and New Directions of Research in Cybernatics and Systems Theory, Guwahati, India. 2004.

[6] S Devadas, HK Ma, R Newton, A Sangiovanni Vincentelli. State Assignment of Finite State Machines Targeting Multilevel Logic Implementations. *IEEE Transactions on Computer-Aided Design*. 1988: 1290-1300.

[7] T Kam, T Villa, R Brayton, A Sangiovanni Vincentelli. Synthesis of Finite State Machines: Functional Optimization. Kluwer Academic Publishers, Boston/London/Dordrecht. 1998.

[8] T Villa, T Kam, R Brayton, A Sangiovanni Vincentelli. Synthesis of Finite State Machines: Logic Optimization," Kluwer Academic Publishers, Boston/London/Dordrecht. 1998.

[9] Shiue WT. Novel state minimization and state assignment in finite state machine design for low power portable devices. *Integr. VLSI J.,* 2005; 38: 549-570.

[10] Goren S, Ferguson FJ. On state reduction of incompletely specified finite state machines. *Computer Electr. Eng.,* 2007; 33(1): 58-69.

[11] Dolotta TA, EJ McCluskey. The coding of internal states of sequential circuits. Electronic Computers. *IEEE Transactions on.* 1964; 5: 549-562.

[12] G De Micheli Synthesis and Optimization of Digital Circuits. New York: McGraw Hill. 1994.

[13] Benini L, Micheli De G. *State assignment for low power dissipation*. IEEE Custom. Integr. Circuits Conf., 1994; 30(3): 136-139.

[14] Xia, Yinshui, AEA Almaini, Xunwei Wu. Power optimization of finite state machine based on genetic algorithm. *Journal of Electronics (China)*. 2003; 20.3: 194-201.

[15] Chattopadhyay S. Area conscious state assignment with flip flop and output polarity selection for finite state machine synthesis genetic algorithm approach. *Comput. J.,* 2005; 48(4): 443-450.

[16] Hong SK, Park IC, Hwang SH, Kyung CM. State assignment in finite state machines for minimal switching power consumption. *IEE Electron. Lett.,* 1994; 30(8): 627–629.