An Evolutionary Strategy Based State Assignment for Area-minimization Finite State Machines

Yanyun Tao
School of urban rail transportation,
Soochow University
Suzhou, 215137, China
taoyanyun@suda.edu.cn

Lijun Zhang
School of urban rail transportation,
Soochow University
Suzhou, 215137, China

*Yuzhen Zhang
The first affiliated hospital of
Soochow University
Suzhou, 215006, China

***Abstract*—Most published results show that area reduction of the finite-state machines(FSMs) is achieved by optimizing the state assignment. In order to solve this state assignment problem(SAP), an evolutionary strategy based state assignment, called ESSA, is proposed in this study. Two cost functions(i.e. fitness functions) are defined for two-level and multilevel area minimization, respectively. A new selection strategy and a new mutation are proposed in ESSA, which are specifically designed based on the search space of SAP. The selection strategy employs the niche mechanism based on the crowding to select diversified individuals, and mutation uses 'replacement', '2-exchange' and 'shifting' operators, which is controlled by the hamming distance constraint, to generate offspring from the parental individuals. Experimental results show ESSA achieves a significant area reduction to the previous publications in terms of number of cubes and literals in most benchmarks.**

I. INTRODUCTION

As the rapid increase in density and size shrinking of chips, area of circuit implementation has been a critical attraction in Very Large Scale Integration(VLSI) design. VLSI systems are mainly composed by combining combinational logic modules and sequential modules. Sequential circuits play a role in linking and controlling combinational logic parts. Finite-state machine (FSM) is usually used for modeling sequential circuits. Design of a FSM involves encoding and assigning the states, synthesizing the combinatorial part, and guaranteeing self-starting of the circuit when the redundant states occur. State assignment is a mapping from the set of states of an FSM to the set of binary codes. In the field of FSMs, the efforts have been dedicated to find a state assignment that leads to area or power minimization [1-4,8-15]. In this study, we concentrate on the area optimization for two-level and multilevel circuits.

Using state assignments to optimize area is a complex problem. Especially, for a FSM with many states, codes assigned to the states frequently occur in transitions that may lead to a large area distance. The reason is that, a bit difference between two assignments to the same states may largely change the next states logic blocks and output logic blocks.

Currently, some commercial tools such as QuartusII and Design Compile, and open-source tools such as "ABC"[5] and "SIS"[6], are very popular in VLSI design. However, these tools are unable to obtain good solutions sometimes. Some state assignment techniques such as nova[10] and jedi [11] have been integrated in "SIS". However, the state assignments generated by nova and jedi are far from the optimal solutions in terms of area minimization of FSM. Evolutionary algorithms (EAs) have become more and more popular in solving complex optimization problems [7] such as Travelling Salesman problem(TSP), data mining and regression problem. Many researchers have used EA for addressing the state assignment problem. For optimizing area and power consumption simultaneously, multi-objective evolutionary algorithm (MOEA) was employed[18-19]. However, MOEA usually spent a longer evolution time on achieving same good results as single-objective evolutionary optimization. In terms of area reduction, an evolutionary approach for single-objective optimization is preferred. The published methods mostly used Genetic Algorithm(GA) to look for low area or power state assignments. Besides GA, simulated annealing (SA)[16] and particle swarm optimization(PSO)[17] were also applied in state assignment optimization. However, it is hard for EA to approach the optimal state assignments of a large FSM within an acceptable time.

Despite many works on solving the state assignment problem, the current state assignment methods are mainly based on the general evolutionary algorithms such as GA and PSO, which is not proposed for the specific applications. The general rules of genetic operators may fail to solve the state assignment problem. The aim of this work is to look for the lowest area state assignment for two-level or multilevel circuits, where "lowest area" indicates the smallest number of cubes or literals. For accomplishing the task, we provide a new evolutionary strategy(ES) based state assignment method, called ESSA, of which the selection and mutation are specifically designed based on the analysis of the state assignment problem and individual's distribution. The proposed method in this paper will be tested comprehensively on the benchmarks.

II. RELATED WORK

An arbitrary choice of state assignment is easy, and will always work. However, it may require more logic gates or power to accomplish a FSM in some cases. In the field of state assignment, some researchers concern on the power dissipation reduction of FSMs by optimizing state assignment. Cho et al. proposed a m-block partition method to improve the testabilities and power consumption simultaneously[1]. Xia et al. proposed a GA-based method for low power state assignment[2]. It aims to use state assignment to minimize the power dissipation. Xia et al. also proposed a low power FSM partition method [3]. This method uses GA to optimize partition of FSMs and makes the existing monolithic FSM state assignment be applicable to partitioned FSMs. By partition of FSMs, the power dissipation can be reduced up to 80%. At 2004, Chattopadhyay proposed a FSM state assignment method for low power consumption [4].

Unlike the power-orient approaches, a few researches concentrate on state assignment for area minimization. The determitic methods such as mustang[8], muse[9], nova[10]and jedi[11] are typical area-orient methods proposed for area minimization. Almaini et al. used GA to search state assignments for optimizing synchronous FSM [12]. Its fitness is formed by the literal count of ESPRESSO minimization. However, the method of Almaini has not been tested on certain large circuits. Amaral used a set of heuristic rules to find excellent state assignments that result in small-area circuit implementation [13]. It is stated that the small hamming distance between the states, which are associated with each other more frequently is beneficial; however, sometimes it cannot result in area reduction of circuits. Chattopadhyay firstly proposed Flip-Flop and output polarity selection into representation of solution[14]. The Flip-Flop and output polarity selection both expands the search space and solution space. However, the numbers of cubes obtained by this method for benchmarks are far away from the optimal solutions. Ali et al. aims to optimize state assignment for the evolutionary design of circuits[15]. However, their state assignments may not be efficient in the conventional design of circuits because the evolutionary circuit design brings redundancy. In addition, Aly used SA as a finder of optimal assignments [16], but he just gave a case study for the method. Different from the methods to the two-level logic optimization, Aiman et al. [17] proposed a state assignment approach, called binary-PSO (BPSO), and a GA state assignment approach[18], for multilevel circuits. The experimental results illustrate that the method is efficient in multilevel area minimization.

In addition, some researchers have focused on multi-objective evolutionary optimization for FSMs. Xia et al. introduced two cost functions and used multi-objective genetic algorithm (MOGA) for area and power optimization [19]. They tried to obtain low power dissipation assignments without the large area penalty. Jassani proposed a MOGA based method for state assignment [20]. It aims to find a state assignment that makes the product of hamming distance between states and state transition probability minimized, so that the area and power can be both reduced. Jassani obtained a set of assignments with a long evolution time penalty. For example, the benchmark "planet" costs more than 25 hours. No matter which multi-objective optimization algorithm is used, it is hard to obtain competitive state assignments in terms of any single-objective.

III. State Assignment Problem

State assignment problem, denoted by SAP, is formulated in this section. For a FSM, state assignment encodes the states into binary codes. Using state assignment to reduce area is an efficient way for FSMs area minimization. However, a FSM with n states(k bits/state) has $(2^k!)/(2^k-n)!$ state assignments. When n increases, k will also increase and the search space will increase largely. It requires large computational time to exhaust all assignments in a large search space to look for the one will minimize the area. Therefore, finding a state assignment that leads to area minimization is a NP problem. SAP is actually a combinatorial optimization problem and the formulation of SAP is as (1).

$$Area(x^*)=\mathbf{min}\ Area(x_i) \quad (1)$$
$$x_i\in\Omega,\ \ \Omega=\{x_1,x_2,\ldots,x_n\}$$

Where $Area(x_i)$ is the area of logic implementation using state assignment x_i. Ω is the solution space of state assignments.

Let us give an example to specify SAP. A 1-input-5-state FSM is given in Table.I. For this FSM, there are $(2^3!)/(2^3-5)!=6720$ feasible state assignments, two of them, denoted by Assign #1 and Assign #2, are present in Table.I. By implementing two-level logic minimization, Assign #1 obtains 5 cubes and 15 literals while Assign #2 obtains 4 cubes and 12 literals. In terms of the number of cubes and literals, Assign #2 is superior to Assign #1. This example illustrates that the state assignment has a very significant impact on the area minimization of a synthesized FSM.

TABLE I. LOGIC IMPLEMENTATION OF A FSM USING DIFFERENT STATE ASSIGNMENTS

Present State	**Next State**		**Assign #1**	**Assign #2**
	X=0	**X=1**		
S_0	S_1	S_2	000	100
S_1	S_4	S_3	100	110
S_2	S_4	S_3	110	010
S_3	S_4	S_4	011	001
S_4	S_0	S_0	010	000
Logic implementation(two-level) using the state assignment			$D_2=\overline{Q}_2\overline{Q}_1\overline{Q}_0$ $D_1=\overline{Q}_2Q_1Q_0+Q_2\overline{Q}_0$ $+X\overline{Q}_2Q_1\overline{Q}_0$ $D_0=XQ_2\overline{Q}_0$	$D_2=\overline{X}\overline{Q}_1\overline{Q}_0+\overline{Q}_2\overline{Q}_1\overline{Q}_0$ $D_1=Q_2\overline{Q}_1\overline{Q}_0$ $D_0=XQ_1\overline{Q}_0$
The number of cubes and literals			5, 15	4, 12

IV. Proposed Method

In this study, we propose a new ES based state assignment method, called ESSA, to solve SAP. The chromosome, selection, mutation, and fitness function of ESSA are described in details in the following subsections.

A. Chromosome

Chromosome is the representation of solutions to SAP. To SAP, the chromosome contains three parts, state assignment code, function polarity code, and strategy-parameter. State assignment code indicates the code assigned to the states of FSM, and function polarity code (1/0) indicates the function/the complement of output or next state logic blocks. The function polarity code and the state assignment code together form the object variables, which is the essential component of the chromosome. Strategy-parameter in this study is the mutation step size for individual variation, which is coevolving together with the object variables. In the object variables, the state assignment codes would be changed by multiple mutation operators, of which the mutation step size is determined by the strategy-parameter. The function polarity codes and the strategy-parameter are mutated randomly in the evolution.

Given a n-state-m-output FSM. The state assignment code for the FSM can be defined as an integer string $\boldsymbol{x}=(x_1, x_2..., x_n)$, where x_i is selected from the set $\{0,1\dots,2^k\}$, $x_i \in N$ and $k=\log_2 n$. In a chromosome, the codes cannot be duplicated. Function polarity code is composed by output polarity code and next-state polarity code, which is defined by a binary string $\boldsymbol{p}=(p_1, \dots, p_m \mid p_{m+1}, \dots, p_{m+k})$, where $p_1,\dots, p_m$ denotes the output polarity code and $p_{m+1},\dots, p_{m+k}$ denotes the next-state polarity code. $p_i=1$ indicates the function and $p_i=0$ indicates the complement. Strategy-parameter is defined as σ, which controls the mutation step size of state codes. The full size chromosome of state assignment is defined as $(\boldsymbol{x}, \boldsymbol{p}, \sigma)$.

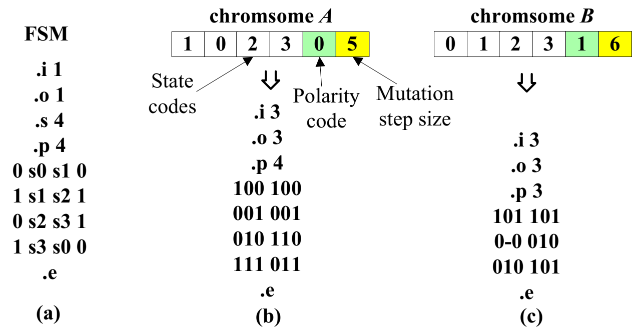

Fig.1 (a) state transition of FSM (b) Chromosome A and its logic implementation (c) Chromosome B and its logic implementation

Fig.1 gives two chromosomes encoding state assignments of a 1-input-4-state-1-output FSM. Fig.1(a) presents the state transition of the FSM. The state codes of A and B are "1,0,2,3" and "0,1,2,3",which are selected from the set {0,1,2,3}. The polarity codes in A and B are 0 and 1. The mutation step size (strategy parameter) for A and B are 5 and 6(this is only made use in mutation). Fig.1(b) shows the logic implementation of A using ESPRESSO obtains 4 cubes and 12 literals, while the implementation of B shown in Fig.1(c) has 3 cubes and 8 literals.

B. Selection

Selection exploits the fitness information in order to guide the search into promising search space regions. In ES, "plus selection", denoted by $(\mu+\lambda)$, and "comma selection", denoted by (μ,λ), are two typical versions of selection strategies. Both selection variants have their specific applications. Let us analyze the search space of SAP. For the semantic of state assignment, the codes assigned to states must be no duplicated in a chromosome (e.g. Fig.1). This leads to a combinatorial search space, which is discrete and extremely versatile. According to the literature [21], the (μ,λ) selection is recommended for unbounded search spaces, while the $(\mu+\lambda)$ selection should be used in discrete finite size search spaces, e.g., in combinatorial optimization problems. Therefore, $(\mu+\lambda)$ selection is more adaptable than (μ,λ) selection to SAP.

$(\mu+\lambda)$ selection is able to accelerate the convergence to solution and suitable for SAP. However, it preserves better intermediate solutions and further steps into the premature stagnation. In contrast to $(\mu+\lambda)$ strategy, (μ,λ) selection doesn't take the parents into account, and makes immediate use of new information gathered by the next offspring. It finally climbs to the top of mount even the current good solutions maybe replaced with the worse ones. However, as the literature[21] stated, (μ,λ) selection is not adaptive to combinatorial optimization problem. Therefore, both selections have disadvantages in solving SAP.

In this study, a new selection strategy is proposed for SAP. The strategy inherits $(\mu+\lambda)$ strategy and derives the advantage from (μ,λ) strategy. In the strategy, niche mechanism is employed based on the crowding, which is defined as an evaluation to select diversified good fitness individuals from the selection pool. Hence, the information derived from the new generated individuals can be utilized. Let A and B are two individuals; let $d(A,B)$ be the hamming distance between A and B. Let $D(A)$ be the crowding of A. $D(A)$ is calculated as (2).

$$D(A)=\sum sh[d(A,B)] \tag{2}$$

Where $d(A,B)$ indicates the hamming distance between A and B, which is calculated by $d(A,B)=\sum A[i]\oplus B[i]$, where $A[i]$ and $B[i]$ indicates the i-th state code bit of A and B. $sh[d]$ denotes a share function, which is defined as (3).

$$sh[d]=\begin{cases}0, & d>\phi_{\max}\\ 1-d/\phi_{\max}, & d<\phi_{\max}\end{cases} \tag{3}$$

Where $\varphi_{\max}$ denotes the radius of niche. A small $D(A)$ implies that A is rather distant from the other individuals $\varphi_{\max}$ in terms of the hamming distance, and a large value of $D(A)$ implies that A is close to the other individuals within $\varphi_{\max}$. The fitness together with the crowding determines the survival of individuals.

The pseudo code of the proposed selection strategy is as follow.

```
Selection()
{
 /*μ parents and λ offspring has been combined as a selection pool. The fitness, denoted by chr.F, and the crowding, denoted by chr.D of each individual chr in the selection pool are calculated.*/
    Select the best-fitness individual into the parental set ßp;
    Rank the (μ+λ-1) individuals based on (chr.F × chr.D) in ascending order;
    Select the (μ-1) top individuals into ßp;
}
```

According to the pseudo code, μ parents and λ offspring are together sorted based on the fitness and the crowding. The best individual is always selected as a parent. The remaining

(μ-1) parents are derived from the individuals with good fitness and small crowding. The new strategy is able to accelerate the convergence of ES. Since the selected individuals(no-duplicate) have relatively small crowding. The diversity of parental population is secured and the premature, from which (μ+λ) strategy suffers, can be overcome.

C. Mutation

Mutation is an important variation operator in ES, which explores the search space from parental population. The design of mutation operators is problem-dependent. According to the literature [21], a successful mutation usually meets three requirements, *reachability*, *unbiasedness*, and *scalability*. In this study, three operators, "replacement", "2-exchange" and "shifting", are used for mutation. In real-valued search spaces the variation strength on a parental individual is usually regarded as the mutation step size. However, the neighborhood size will be limited in the combinatorial search space if variation strength is used; the scalability requirement of the mutation steps cannot be guaranteed. In this study, we use the hamming distance between an individual and a parental individual, not the variation strength, as the mutation step size in the search space of SAP. The hamming distance is controlled by the strategy parameter σ.

The pseudo code of mutation is denoted as follow.

```
Mutation()
{
  For each individual chr to be mutated;
    step ← 0;
    While step < chr.σ
      s∈{statecode};
      chr.x'← replacement(chr.x, s);              /* replacement */
      or chr.x' ← 2-exchange(chr.x, chr.y);       /* 2-exchange */
      or chr.x' ← shifting(chr.x[i, j], p);       /* shifting*/
      step←step+d(chr.x, chr.x');                 /* hamming distance*/
    End while
    chr.f:=chr.f|{ 0|1, 0|1, …,0|1};
    chr.σ ←chr.σ + rand(-r, r);
  End For;
}
```

In the pseudo code, {*statecode*} denotes the feasible state codes have been used. Three operators, "replacement", "2-exchange", and "shifting", are randomly chosen to change the state assignment codes in an individual. p in 'shifting($chr.x[i, j], p$)' indicates the number of steps, which the parental state codes $x[i, j]$ shifts. With respect to the other components of chromosome, the polarity code $\boldsymbol{f}$ is randomly changed, and the parameter σ varies in the range of r.

Fig.2 presents the mutation operator applied on an individual. Each numbered cell indicates a state code. The operators change the state codes by randomly cutting the connections between the cells, exchanging, and replacing the cells in a specific way. As shown in Fig.2, the mutation applied on state assignment codes will continue until the mutation step size σ is reached. In the mutation process, the parameter σ is randomly changed from 5 to 8.

It is noted that the mutation operators do not introduce any bias on the selected parental individuals. In the mutation, a small hamming distance(variation step) on individuals may leads to better solutions in the same local domain, while a large hamming distance may find solutions on the other domains(e.g. the global optima). In theory, this mutation strategy guarantees that any other individual can be reached within a finite number of mutation steps. Therefore, the *reachability* requirement can be meet. It can be seen that the mutation step size based on hamming distance is tunable, which adapts to the properties of the fitness landscape of SAP. Hence, the proposed mutation is also able to meet the *scalability* requirement.

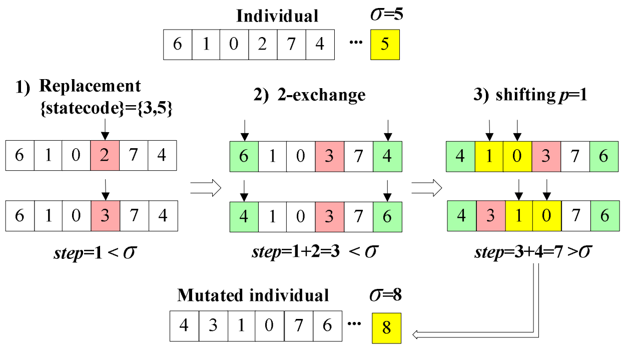

Fig.2 The effect of the mutation operator for SAP

D. Cost function

Cost function (i.e. fitness function) is used to evaluate the individuals in the evolution. In the two-level area minimization, the objective is to reduce the number of Sum of Product (SOP) terms (i.e. cubes), while in multilevel area minimization the objective is to reduce the number of literals in the expressions of the synthesized circuit. Therefore, we define two cost functions based on the number of cubes and literals for area optimization. ESPRESSO and SIS are two conventional tools for circuit synthesis. For two-level circuits, the number of cubes is obtained by run ESPRESSO implementing the minimization with term sharing. For multi-level circuits, Espresso-Single Output with fast extraction (Espresso-SO+fx) available in SIS is used to obtain the number of literals of logic expression.

The cost functions are defined as follow.

a) Let *#cube* be the number of cubes the resulting logic expression. The cost function for two-level logic implementation can be denoted by (4).

$$F=\#cube \tag{4}$$

b) Let *#lit* be the number of literals of the resulting logic expression. The cost function for multi-level logic implementation can be denoted by (5).

$$F=\#lit \tag{5}$$

The optimization orients to two-level and multi-level area minimization runs twice – one time to find minimum number of cubes for two-level logic implementation and another time to find minimum number of literals for multi-level logic implementation. Therefore, (4) and (5) are not combined into a fitness function, but are used as fitness function separately.

E. Process of ESSA

- Read STG description file (i.e. kiss2 file) of a FSM;

- Create (μ+λ) initial individuals, g=0;
- Create Berkeley standard PLA files for the initial individuals;
- Run ESPRESSO or SIS to process the PLA files for two-level or multilevel area minimization;
- Calculate the fitness value F of the individuals based on (4) or (5);
- Calculate the crowding D of the individuals based on (2);
- **Repeat**
  - Select μ out of (μ+λ) individuals by the selection strategy as a parental population $S(g)$;
  - Mutate the μ parents to generate λ offspring;
  - Create PLA files for the λ offspring;
  - Run ESPRESSO or SIS to process the PLA files;
  - Calculate the fitness value F of the λ offspring based on (4) or (5);
  - Combine the μ parental individuals and λ offspring as a selection pool;
  - Calculate the crowding D of the (μ+λ) individuals based on (2);
  - g= g +1;
- If an individual with the desired fitness is evolved, or other required demand, such as the maximum generation, is achieved; the evolutionary process terminates; else return to chromosome selection.
- Output the best-evolved individual and its parameters(the number of cubes and literals , evolutionary time);

V. EXPERIMENTS

We use twenty-two FSMs from MCNC to test the performance of ESSA in two-level and multilevel logic minimization of FSM. The FSMs have been commonly used for state assignment optimization. In this section, we compare the ESSA with the other state assignment methods in terms of the number of cubes and literals to demonstrate the superiority of ESSA. Table.II gives the information of experimental FSMs, including PI(the number of primary inputs), St (the number of states) and PO(the number of primary outputs).

TABLE II. EXPERIMENTAL FSMS

Circuit	PI	St	PO	Circuit	PI	St	PO
bbara	4	10	2	**dk512**	1	14	3
dk16	2	27	2	**bbsse**	7	16	7
dk14	3	7	5	**cse**	7	16	7
opus	5	10	6	**ex4**	6	14	9
donfile	2	24	1	**ex2**	2	19	2
bbtas	2	6	2	**ex3**	2	10	2
tav	4	4	4	**ex1**	9	20	19
tbk	6	32	3	**keyb**	7	19	2
s1	8	20	6	**styr**	9	30	10
s832	18	25	19	**planet**	7	48	19
s1494	8	48	19	**sand**	11	32	9

The codes of ESSA implemented in Java run on the PC with CPU 2.13GHz 2GB RAM. The ESSA parameters are shown in Table.III. *#MaxGen* indicates the maximum generation for evolution termination. *#MaxGen* is set to a large value 5000 for the experimental FSMs. The parental population size(μ) is set to 20, and the offspring population size (λ) is 100. *#Run* indicates number of evolutionary runs. In this paper, it is set to 20.

TABLE III. ESSA PARAMETERS FOR STATE ASSIGNMENT EVOLUTION

ESSA –parameters	Setting
#MaxGen	5000
μ, λ	20, 100
#Run	20

A. Comparison of ESSA and the other state assignment methods

In this section, we compare ESSA with nova(1990), the method of Almaini(1993), the methods of Xia(2002, 2003), the method of Chattopadhyay(2005),and the method of Jassani(2011) in terms of two level logic minimization. In terms of multilevel logic minimization, we compare ESSA with nova, GA(2006) and BPSO(2013). In the following tables, '***impr***' denotes the total improvement on all the cases achieved by ESSA over the comparable methods, which is calculated by (6).

$$impr = \frac{\sum_I ESSA - \sum_I alt}{\sum_I alt} \qquad (6)$$

Where *alt* indicates the value obtained by the comparable methods. *I* denotes the set of experimental FSMs. In Table.IV, *impr* is calculated based on $\#C_B$

Table.IV gives the comparison between ESSA and the reference state assignment methods in terms of two-level area minimization. $\#C_A$ and *#lit* denotes the number of cubes and literals obtained by ESSA without using polarity selection in chromosome, $\#C_B$ indicates the number of cubes obtained by using polarity selection. *#C* stands for the number of cubes. The best values in tables achieved by the alternative methods are marked bold.

In terms of $\#C_A$, ESSA outperforms the other methods in most cases. Although the improvement obtained by ESSA is not very large in some FSMs, the small improvement is achieved with the premise that the comparable state assignments have been already approximated to the optimal solution. Even a slight improvement on state assignment indicates a large progress on the two-level logic minimization of FSM.

ESSA presents an equal good result to the other methods on "ex4". On only two FSMs (14.2%), ESSA cannot obtain better solutions than the other methods. On "donfile", $\#C_A$ obtained by ESSA is better than the other methods except the one of Jassani. On "ex1", ESSA performs a little worse than nova, but much better than the method of Jassani and the one of Xia. When using function polarity, ESSA obtains an improvement on the area reduction(i.e. $\#C_B$< $\#C_A$) on five FSMs(35.7%). According to the result of two-level logic implementation, ESSA obtains better solutions, at least parallel ones, compared to the other methods on twelve FSMs(85.8%).

In addition to two-level logic implementation, ESSA is also compared with two state assignment methods, GA and BPSO, for multi-level logic implementation. Table.VI gives result of multi-level logic minimization implemented by ESSA, GA and BPSO. In Table.V, “**ave**” indicates the average number of literals obtained by the method for several runs. “**best**” indicates the smallest number of literals, which is achieved by the method in several runs.

TABLE IV. COMPARISON BETWEEN ESSA-STATE ASSIGNMENT AND OTHER ASSIGNMENTS IN TERMS OF THE NUMBER OF CUBES AND LIT

FSMs	**nova[10] (1990)**	**Almaini[12] (1993)**	**Xia[19] (2002)**	**Xia[2] (2003)**	**Chat.[14] (2005)**	**Jassani[20] (2011)**		**ESSA**		
	#C/#lit	***#C/#lit***	***#C***	***#C***	***#C***	***#C***	***t_{evo}***	***$\#C_A/\#lit$***	***$\#C_B/\#lit$***	***t_{evo}***
dk512	23/95	23/90	---	---	---	---	---	**17/62**	**17/55**	0:13
bbara	24/134	--	22	22	23	22	0:8	**21/105**	**21/105**	0:12
bbsse	29/--	--	27	---	26	---	---	26/139	**23/138**	2:13
cse	45/--	--	43	43	45	43	3:09	**42/279**	42/281	2:36
donfile	28/--	36/158	36	36	31	**22**	6:04	28/111	25/107	10:51
opus	16/--	--	19	15	**12**	15	0:40	15/86	15/89	0:56
dk16	72/--	69/380	--	--	68	57	6:03	56/292	**55/268**	4:49
keyb	48/--	--	48	46	46	46	3:32	45/403	**45/392**	3:37
s1	63/--	--	66	66	68	43	6:03	56/334	**41/211**	8:11
ex1	**37/--**	--	52	52	47	48	6:07	45/287	45/293	4:11
ex4	19/--	--	16	14	15	13	6:01	**13/62**	13/65	2:09
styr	94/--	--	88	88	78	78	6:05	78/546	**76/522**	5:51
planet	86/--	--	86	86	84	81	25:23	78/500	**78/495**	8:11
sand	89/--	--	89	---	96	---	---	86/587	**85/589**	5:40
Total	**673/229**	**128/628**	**592**	**468**	**629**	**468**	--	**606/3793**	**581/3610**	--
%impr	**13.7/30.1**	**24.2/31.5**	**14.02**	**14.32**	**10.33**	**2.56**	**--**	--	**--**	--

TABLE V. COMPARISON BETWEEN ESSA AND OTHER APPROACHES IN TERMS OF MULTI-LOGIC SYNTHESIS

FSMs	**nova**	**GA[18](2006)**			**BPSO[17](2013)**			**ESSA**		
		best	**ave**	**t_{evo}(s)**	**best**	**ave**	**t_{evo}(s)**	**best**	**ave**	**t_{evo}(s)**
dk14	111	102	103.1	1543.2	98	99.8	784	**72**	76.7	660
tav	365	24	24	893.8	24	24	--	**23**	24.1	718
bbara	57	**49**	49.4	1204.3	**49**	52.9	793	52	54.3	2063
bbsse	140	101	100.5	1350.7	102	107.1	823	**93**	103.1	2769
cse	214	179	184.2	1743.7	184	191.0	987	**174**	182.5	2399
keyb	201	**142**	152.4	2096.1	143	163.8	1480	145	156.3	8220
s1	340	**132**	216.2	2270.9	173	231.7	1592	158	233.6	3298
s1494	715	570	591.1	3885.1	588	602.0	3073	**541**	581.2	13140
s832	274	230	256.7	2254.9	216	245.4	1751	**214**	233.1	6211
ex2	127	**64**	90.1	1373.7	66	99.5	861	70	71.6	1808
ex3	71	54	54.7	1149.1	**51**	54.0	796	56	98.8	1562
tbk	365	410	462.1	8060.8	261	368.2	6671	**252**	366.2	3342
styr	502	405	422.5	3506.2	412	437.5	2326	**387**	417.4	9371
planet	591	466	505.4	3775.0	494	526.1	2288	**454**	495.0	2658
sand	558	498	519.8	3002.6	488	510.1	2374	**486**	505.8	7480
Total	**4631**	**3426**	**3732.2**	**38110.1**	**3349**	**3713.1**	**26599**	**3177**	**3599.7**	**65699**
%impr	**31.5**	**7.26**	**3.55**	**--**	**5.13**	**3.05**	**--**	**--**	**--**	**--**

TABLE VI. COMPARISON BETWEEN STANDARD $(M+\Lambda)$ **ES** , (M,Λ) **ES** AND ESSA IN MULTI-LOGIC IMPLEMENTATION

FSMs	$(\mu+\lambda)$ **ES**			(μ,λ) **ES**			**ESSA**		
	best	**ave**	**t_{evo}(s)**	**best**	**ave**	**t_{evo}(s)**	**best**	**ave**	**t_{evo}(s)**
dk14	**72**	78.2	578	105	111.8	912	**72**	76.7	660
tav	25	27.4	478	25	29.5	602	**23**	24.1	718
bbara	54	56.5	2133	67	69.5	3988	**52**	54.3	2063
bbsse	104	106.6	2289	124	135.7	3567	**93**	103.1	2769
cse	182	185.2	1982	199	212.2	4343	**174**	182.4	2399
keyb	156	159.8	6684	189	192.2	12012	**145**	151.3	8220
s1	167	199.1	4141	187	273.3	6289	**158**	214.9	3298
s1494	548	578.1	13223	591	602.1	14521	**541**	561.3	13140
s832	225	246.3	2877	256	271.2	7021	**214**	233.2	6211
ex2	84	91.4	1292	102	111.3	2531	**70**	72.6	1808
ex3	**55**	78.2	1455	64	102.6	1933	56	68.8	1562
tbk	273	351.2	3660	281	371.3	4196	**252**	327.8	3342
styr	412	422.2	5412	456	488.7	11309	**387**	417.4	9371
planet	457	492.9	3789	505	522.0	7789	**454**	495.0	2658
sand	491	521.1	8720	507	525.2	12129	**486**	505.8	7480
Total	**3305**	**3594.2**	**58713**	**3658**	**4018.6**	**93142**	**3177**	**3488.7**	**65699**
%impr	**3.87**	**2.94**	**-11.89**	**13.14**	**13.20**	**29.46**	**--**	**--**	**--**

It is noted that ESSA outperforms the other methods in terms of "**ave**". This illustrates that ESSA is more stable in finding a good solution in these cases. In terms of "**best**", ESSA also achieves the best or equally good results in most cases. Compared to GA, ESSA obtains better solutions in ten out of fifteen FSMs(66.67%), and performs better than BPSO in eleven FSMs(78.5%). In only five FSMs (33.3%), ESSA cannot perform better or as well as GA or BPSO do in terms of "**best**". This is because finding a near optimal or the optimal state assignment for some simple FSMs is relatively easy, so that the room of improvement for these FSMs is quite small. Compared to the classical state assignment method 'nova', ESSA achieves better solutions in all the cases.

In Table.IV and .V, t_{evo} indicates the evolutionary time of finding the best solution. Time format in Table.IV is unified to hh:mm, e.g. 1:55. This means 1 hour and 55 minutes. According to t_{evo} in Table.IV, ESSA spends large evolutionary time on finding solutions in most cases. ESSA needs more computation time on "donfile", "bbara" and "s1" than the method of Jassani. However, on "planet", "styr" and "ex1", ESSA costs relatively small evolutionary time to find good solutions. For example, ESSA obtains a better solution on "planet" than the method of Jassani by using only 8 hours and 11 min. Generally, the computational effort of minimizing FSMs is still in the range of hours despite fast computer. The value of t_{evo} in Table.V represents the evolutionary time in seconds. According to t_{evo} in Table.V, ESSA spends a larger evolutionary time on finding a good solution than BPSO and GA in most cases. However, the situation is precisely the opposite in some cases. On "planet", ESSA obtains a better solution than GA within only 2658s, which is much smaller than GA needs. On "tbk", it costs ESSA only 3342s to find a solution, while it costs BPSO and GA 6671s and 8060.8s to accomplish the task.

For illustrating the ability of ESSA in solving SAP, ESSA is compared to two standard ES methods, ($\mu+\lambda$) ES and (μ,λ) ES. Table.VI gives the results obtained by three ES methods. According to t_{evo}, the standard ($\mu+\lambda$) ES is the fastest finder of a solution (maybe a local optima), and the (μ,λ) ES is the worst one among the three methods. This illustrates that ESSA is poorer than ($\mu+\lambda$) ES in terms of the convergence, but much better than (μ,λ) ES. In terms of multilevel area minimization, ESSA is comprehensively superior to (μ, λ) ES in all the cases, and obtains better results than ($\mu+\lambda$) ES in most cases. According to ***impr***, ESSA has a total improvement over ($\mu+\lambda$) ES and (μ,λ) ES. This is because ESSA is able to jump out of local optima, which is usually a trap of ($\mu+\lambda$) ES. Meanwhile, ESSA is able to converge to multi local domains (one of these domains maybe the global optima), which (μ, λ) ES cannot reach.

Although many researchers claimed that (μ,λ) ES is more efficient in solving optimization problem than ($\mu+\lambda$) ES, this is not always true for some specific applications. Within an acceptable time, (μ,λ) ES usually cannot achieve a good convergence in solving SAP. This is because the search space of SAP is so versatile in some cases that (μ,λ) ES likely vacillates between local domains and cannot continually search for a better solution in a domain. This is also assistance with the point of view that ($\mu+\lambda$) selection is more adaptive to discrete finite size search spaces, e.g., in combinatorial optimization problems than (μ,λ) ES[22].

B. results and analysis

In Table.IV, .V and .VI, the total improvement on area minimization for all the cases achieved by ESSA has been present. According to the value of ***impr***, ESSA is totally superior to the other methods except BPSO. In addition to the total improvement, the improvement on area reduction for single case is also an important evaluation of the methods. In this section, the improvement for each case achieved by ESSA over the other methods is discussed and analyzed.

It is noted that ESSA can achieve a large area reduction in terms of two-level logic implementation compared to nova and two methods of Xia. Compared to the methods of Chattopadhyay and Jassani, ESSA also achieves a preferable area reduction. On "s1", ESSA achieves 22.7% reduction to the methods of Xia, 39.7% area reduction to the method of Chattopadhyay and 4.7% to the method of Jassani. Sometimes, different state assignment methods can obtain equal number of cubes. In this case, the number of literals is another important evaluation of state assignment method.

In multi-level logic minimization, ESSA is efficient in area reduction in terms of "**best**" compared to nova, jedi, BPSO and GA, where BPSO and GA are the relatively new state assignment methods. ESSA achieves a 38.53% reduction over GA on "tbk" and 29.4% over GA on "dk14". Compared to BPSO, ESSA achieves 8.09% on "planet" and 26.5% on "dk14". Although ESSA cannot perform well in all the cases, ESSA still achieved better results than GA and BPSO in more than half experimental FSMs. Compared to two conventional methods nova and jedi, ESSA has a very large improvement on multi-level logic minimization in all the cases. On "s1", the number of literals reduced by ESSA is up to 53.52%, and on "tav", the reduction is up to 93.69%.

Generally, ESSA outperforms the other state assignment methods in terms of area reduction for most experimental FSMs. Especially compared with the method of Chattopadhyay in two-level logic minimization, which also employs the function polarity code into chromosome, ESSA is comprehensively over this method. It is also noted that ESSA gave some poor solutions compared to GA and BPSO in some cases. In future work, it is necessary to make further improvements on ESSA.

VI. Conclusion

In the field of state assignment research, many researchers that their state assignments performed well in reducing area of two-level and multilevel logic implementation of FSMs. However, the optimal state assignment is usually hard to be achieved, and the room for improving the state assignment method is still large. This work aims to find a state assignment to reduce area of two-level and multi-level FSMs. Therefore, we propose an evolutionary assignment method called ESSA. The ESSA is extensively tested on many benchmarks, and is compared to the other state assignment methods in these cases. The area reduction(i.e. cubes and literals reduction) obtained by ESSA are satisfied, and the state assignments for a part of benchmarks have been the best ones by far.

ACKNOWLEDGMENT

This work is supported by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (Grant No. 13KJB520023), National Natural Science Found of China(Grant No.61502327) and Science Project of Suzhou (SYG201407).

REFERENCES

[1] Cho, S. Park, "A new synthesis technique of sequential circuits for low power and testing", *Current Applied Physics*. Vol.04.pp:83-86. 2004

[2] Y. Xia, A.E.A. Almaini, and Xunwei Wu. "Power Optimization Of Finite State Machines based on Genetic Algorithm", *Journal of Electronics*, 2003, 20, (3), pp.194- 201.

[3] Y. Xia, X. Ye, and L. Wang. "A Uniform Framework of Low Power FSM Partition Approach", *IEEE International Conference on communication , circuits and systems*, China , 2006, pp. 2642-2646.

[4] S. Chattopadhyay. and P. Reddy, "Finite State Machine State Assignment targeting low power comsumption". *IEE Proc.-Comput. Digit. Tech*. 2004, 151, 61-70

[5] ABC:A System for Sequential Synthesis and Verification. Berkeley Logic Synthesis and Verification Group, http://www.eecs.berkeley.edu/~alanmi/abc/.

[6] Berkeley, Electronics Research Laboratory, SIS: A System for Sequential Circuit Synthesis, Release 1992.05, http://www.eecs.berkeley.edu/Pubs/TechRpts/1992/ERL-92-41.pdf, 1992

[7] Ashlock, D.,Evolutionary Computation for Modeling and Optimization, Springer,(2006) ISBN 0-387-22196-4

[8] Devadas;Ma H T;Newton A R;Vincentelli Sangiovanni MUSTANG:State assignment of finite state machines for optimal multi-level logic implementations 1987

[9] Du X;Hactel G;Lin B;Newton A R MUSE:A multilevel symbolic encoding algorithm for state assignment, 1991(01)

[10] T. Villa, Alberto Sangiovanni-Vincentelli, "NOVA: state assignment of finite state machines for optimal two-level logic implementation"*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. Vol.9(9).pp: 905-924. 1990.

[11] Lin B;Newton A R Synthesis of multi-level logic from symbolic high-level description languages 1989

[12] A.E.A Almaini, J.F. Miller, and P. Thomson. "State assignment of finite state machines using a genetic algorithm". *IEE Proceedings Computers and Digital Techniques,* - Vol.142(4).pp: 279-286. 1995.

[13] J.N Amaral, K. Tumer, and J. Ghosh, "Designing Genetic Algorithms for the State Assignment Problem". *IEEE Transaction on system, man and cybernetics*, Vol.25(4).pp: 100-108. 1995

[14] S. Chattopadhyay, "Area Conscious State Assignment with Flip Flop and Output Polarity Selection for Finite State Machine Synthesis Genetic Algorithm Approach", *The Computer Journal*, 2005, 48, (4),pp.443-450.

[15] B. Ali, A. E. A. Almaini, and T. Kalganova, "Evolutionary algorithms and their use in the design of sequential logic circuits", *Genetic Programming and Evolvable Machines*, Vol.05 pp:11-29. 2004.

[16] W. M. Aly, "Solving the State Assignment Problem Using Stochastic Search Aided with Simulated Annealing**",** *American J. of Engineering and Applied Sciences* ,2009, 2, (4): pp. 710-714.

[17] H. El-Maleha Aiman, T. Sheikhb Ahmad, and M. Sait Sadiq, "Binary particle swarm optimization (BPSO) based state assignment for area minimization of sequential circuits". *Applied Soft Computing* .13 (2013) 4832–4840.

[18] A. El-Maleh, S.M. Sait, F. Nawaz Khan, Finite state machine state assignment for area and power minimization, in: IEEE International Symposium on Circuits and Systems, ISCAS 2006, 21–24 May, 2006.

[19] Y. Xia, A.E.A. Almaini. "Genetic algorithm based state assignment for power and area optimization". IEE Proc.-Comput. Digit. Tech., Vol. 149(4). pp:128-133. 2002

[20] B.A. AL Jassani, N. Urquhart, and A.E.A. Almaini, "State assignment for Sequential Circuits using Multi-Objective Genetic Algorithm". *IET Computers & Digital Techniques*. Vol.5(4).pp: 296-305. 2011.

[21] H.G. Beyer, H.P. Schwefel. Evolution Strategies: A Comprehensive Introduction. Journal Natural Computing, 1(1):3–52, 2002

[22] H.G. Beyer, Some aspects of the 'evolution strategy' for solving tsp-like optimization problems. In: Männer R and Manderick B (eds) Parallel Problem Solving from Nature, 2, pp. 361–370. Elsevier, Amsterdam .1992