Genetic Programming and Evolvable Machines, 5, 11–29, 2004
© 2004 Kluwer Academic Publishers. Manufactured in The Netherlands.

Evolutionary Algorithms and Their Use in the Design of Sequential Logic Circuits

B. ALI b.ali@napier.ac.uk
A. E. A. ALMAINI a.almaini@napier.ac.uk
School of Engineering, Napier University, 10 Colinton Road, Edinburgh EH10 5DT, UK

T. KALGANOVA tatiana.kalganova@brunel.ac.uk
Electrical and Computer Engineering Department, Brunel University, Uxbridge, Middlesex UB8 3PH, UK

Submitted March 26, 2002; Revised April 12, 2003

Abstract. In this paper an approach based on an evolutionary algorithm to design synchronous sequential logic circuits with minimum number of logic gates is suggested. The proposed method consists of four main stages. The first stage is concerned with the use of genetic algorithms (GA) for the state assignment problem to compute optimal binary codes for each symbolic state and construct the state transition table of finite state machine (FSM). The second stage defines the subcircuits required to achieve the desired functionality. The third stage evaluates the subcircuits using extrinsic Evolvable Hardware (EHW). During the fourth stage, the final circuit is assembled. The obtained results compare favourably against those produced by manual methods and other methods based on heuristic techniques.

Keywords: sequential circuits, state assignment, genetic algorithm, evolvable hardware

1. Introduction

Design is the process of translating an idea into a product that can be manufactured. In the design of electronic circuits, this implies a product formed from electronic components, software and electromechanics. Effective design allows this translation to be done quickly, cheaply and accurately to produce a product that is fit for the purpose. The top–down automatic design method of electronic circuits is generally a complex task requiring knowledge of a large collection of domain specific rules. The process of implementing a digital electronic circuit in hardware has typically involved the following stages: (1) transforming the original logical specification into a form suitable for the target technology; (2) minimising and optimising the representation with respect to user defined constraints; (3) carrying out technology mapping onto the target device. The final stage typically involves placing and routing of component gates, which comprise the complete design [1, 2].

It should be emphasised that during all these stages great care has to be taken to maintain the logical functionality of the original circuit specification. Hence, there is a demand for effective tools that perform some of the design tasks leaving the designer to concentrate on issues of performance optimisation. The complexity of the electronic design search space has encouraged the use of Evolutionary Electronic Design (EED) procedures. EED has shown a high degree of flexibility in dealing with complex and computationally hard problems [3, 4, 5, 6]. The automated synthesis of digital logic to satisfy the function specification is a well-researched area [2, 4, 6]. A circuit synthesised using function specification is a relatively straightforward process. However, optimising either the size or the performance of such circuit is a considerably more difficult problem. There is very little research, which actually evolves the functionality of sequential logic circuits [3, 7]. Hardware evolution is performed through a succession of changes, reconfigurations of elementary cell functions and cell inter-connectivity, and selection of the fit configurations until the target functionality is reached. A particular goal of this research is to come up with a design for a sequential circuit that can evolve under the control of a selection algorithm.

EHW approach has begun to show that it is possible to evolve such circuits in a radically different way [3, 4, 6] as described in Table 1. The sequential circuits are divided into purely combinational blocks and registers. Large sequential circuits are typically modelled by smaller interacting finite state machines [1]. An FSM is defined as a mathematical model of a system with discrete inputs, discrete outputs and a finite number of internal configurations or states. The states of a system completely summarise the information concerning past inputs to the system that is needed to determine its behaviour on subsequent inputs.

The aim of this work is to look at the problem of automated synthesis of synchronous sequential circuits using a new approach based on evolutionary algorithms. The evolutionary algorithms are used to design synchronous sequential

Table 1. Recent EHW approach used to evolve sequential logic circuits

Author	Year	Type of sequential circuit	Evolving platform	Target application	Type of EHW
T Higuchi [14]	1993	State transition graph		Digital logic circuit	Extrinsic
H Hemmi [7]	1994	Digital sequential adder	AdAM system	Serial Adder	Extrinsic
A Thomson [11]	1995	Dynamic state machine (DSM)	Simulator "Mr. Chip robot"	Robotics	Intrinsic
C. Manovit [10] and P. Chongstitvatana [3]	1998 1999	Synchronous sequential logic circuits partial input/output sequence [10] and finite-state machine synthesis from multiple partial input/output sequences [3]	PLD, GAL	Frequency detector, Odd Parity Detector, Module-5 counter, Serial Adder	Intrinsic
C. Aporntewan et al. [27]	2000	Learning finite state machine synthesis from partial input/ output sequences	PLD, FPGA	Serial Adder, 0101 Detector, Module-4 counter, Reversible 8-counter	Intrinsic

logic circuits with minimum number of logic gates. GA is meant to mimic Darwinian evolution [9]. A population of candidates is maintained, and goes through a series of generations. For each new generation, some of the existing candidates survive, while others are created by a type of reproduction and mutation from a set of parents. EHW combine knowledge of both GA and logic design to evolve circuits.

This new approach can be expressed as a black box of the problem. From this point of view, one regards the problem of implementing the circuits as being equivalent to designing a black box with inputs and outputs. This black box should be such that on presentation of the original input signals the desired outputs are delivered. The essential new feature of this technique is that the details inside the box are encoded into chromosomes. The chromosome representing a circuit is subject to the usual processes of evolutionary algorithms.

The advantage of EHW over the traditional circuit design approach is its capacity for dynamic and autonomous adaptation. EHW can reconfigure its structure dynamically (on-line) and autonomously, according to changes in the task requirement or the environment in which the EHW is embedded [11]. Considerable progress has been recently made in the evolution of digital combinational logic circuits [8, 9]. The evolution of sequential logic circuits is considerably less mature [3, 10, 11]. The complexity of circuit connections and encoding chromosomes to evolve the sequential logic circuit may be one of the reasons that not much work has been done in this area.

The paper will consider the circuit design technique that is based on the use of evolutionary algorithm during different stages of the circuit design procedure. In the proposed approach both GA for state assignment problem and EHW are combined together, to produce optimal logic circuits. First GA is used to find optimal state assignment. Second, the extrinsic EHW is used to find the functional design of combinational parts of the sequential logic circuits. The remaining sections are organised as follows: Section 2 give summary of the previous work in this area. Sections 3 consider the basic idea of the proposed approach. Section 4 shows an application using GA to find optimal state assignment to identify state transition table for the circuit. In Section 5, the application of EHW is implemented where different simple sequential logic circuits are evolved. The experimental results are given in Section 6. Finally, Section 7 provides the conclusions and future work.

2. Related work

EHW has only recently been applied to the synthesis of sequential logic circuits. Though a number of authors made useful contributions [3, 4, 7, 11, 14]; the subject area is still in the early stage of development.

Table 1 summarises the recent work concerning the evolution of sequential logic circuits. Analysing the table, it can be noticed that EHW has been used mainly to synthesise relatively small sequential circuits. The hardware evolution uses primitive logic gates, which are not powerful enough for industrial applications. Development of large circuits is still an important challenge to EHW researchers. Research in EHW can be subdivided in two main categories: intrinsic evolution and extrinsic evolution.

Therefore, the circuits generated by EHW are evaluated either extrinsically (software simulation) or intrinsically. Intrinsic evolution implies that the circuit is downloaded into the reconfigurable hardware devices and then evaluated.

3. Basic idea of the proposed approach

The design of a synchronous sequential circuit starts from a set of specifications and culminates in a logic diagram or a list of Boolean functions from which a logic diagram can be obtained. In contrast to a combinational logic, which is fully specified by a truth table, a sequential circuit requires a state table for its specification. The first step in the design of sequential circuits is to obtain a state table or an equivalence representation, such as a state diagram.

In this section, a new approach is introduced to evolve sequential logic circuits. The complexity of logic circuits can be defined as a function of the number of gates in the circuit. The central idea of this approach is to represent the circuits in such a way that the genetic operations can be carried out. The architecture of the genetic synthesis of sequential logic circuits is shown in Figure 1. Stage 1 represents the target circuit specification using symbolic state transition table. The state minimization, if required, can be done using existing tools [5]. In the next stage, the genetic algorithm uses this state transition table (STT) to generate optimal state assignment to assign binary code for each state. Therefore, the STT of the sequential circuit is formatted as two-level logic PLA file. GA is used to generate the state assignment aiming to reduce the circuit area. The objective function of GA leads to simpler equations and therefore smaller area. Finally, the processing of genetic algorithm for state assignment and EHW to design the desired circuit are combined together to produce optimum logic circuit. This combined process leads to clear interface among components. Extrinsic EHW proposed in [26] is used to generate the combinational part of sequential logic circuit. The extrinsic EHW uses software models to evaluate the fitness function of the resulting circuit. The genetic synthesis

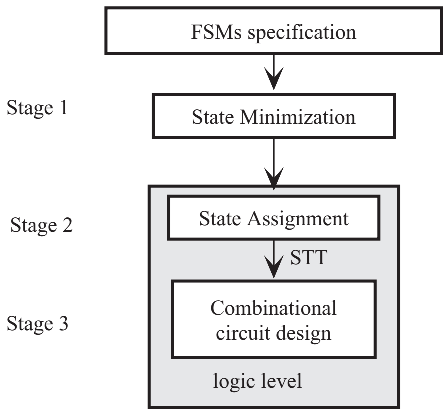


Figure 1. Processing of the proposed approach.

creates circuits at the gate-level by using function set of logic device such as AND, OR, NOT and D flip-flops.

4. Generation of state transition table

First stage to evolve a sequential circuit is to find optimum state assignment, which minimises the number of components. The only restriction for a valid minimal length state assignment is that each state has to be assigned a unique binary value. The total number of possible unique assignment for FSM is given by $A(n, b) = (2^b - 1)!/(b!(2^b - n)!)$ [1]. Where n is number of states and b is the smallest integer that is equal to or greater than $\log_2 n$. The number of distinct assignment is large enough to discourage any attempt at obtaining the solution. Table 2 presents a variety of available approaches for the state assignment problem. In addition, there are a number of algorithms implemented with SIS "A System for Sequential Circuit Synthesis" [5] such as NOVA [15], which targets a two-level implementation and JEDI [16], which targets a multilevel implementation. The input to the system for synthesis of an FSM is typically a KISS (Keep Internal State Simple) format file.

During recent years, several researchers have been applying genetic algorithm technique to solve the state assignment problem [17, 18].

There is no completely satisfactory manual technique for finding the optimum state assignment. The problem of finding the optimum state assignment is NP-hard [1]. The GA finds good optimal solutions short of complete enumeration and evaluation of all possible assignments.

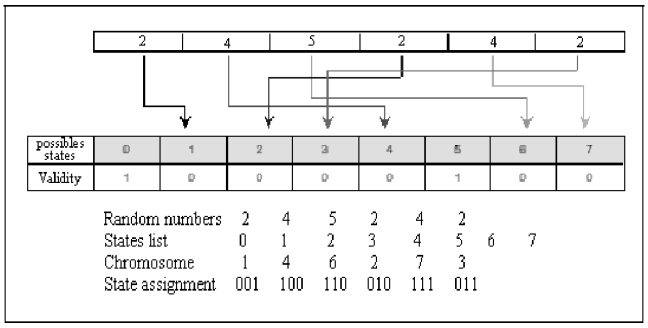
4.1. Chromosome representation for GA state assignment problem

A finite state machine can be described by STT or state transition graph (STG). When implementing finite state machines, they are commonly represented as two-dimensional tables with a number of rows equal to the number of states and a number of columns equal to the size of the input alphabet. The intersection between a state and an input alphabet contains the output symbol and the next-state transition. The chromosome represents the FSMs as a list of states. The length of the chromosome is equal to the number of the states used for the sequential machine. The initial population is generated randomly. Each chromosome represents a solution to the problem. The duplicated chromosomes are discarded. In order to encode the actual information, the FSM is represented as a list of n states; the ith element of the list is a number in the range from 1 to $(2^b - i + 1)$. Consider the example shown in Figure 2 where the genotype of the chromosome has been generated randomly. The genotype of a problem is represented by array of integers. The figure shows how a six state sequential machine is encoded. According to Figure 2, the method can be summarised as follows:

(a) Random function is used to generate six integers (2,4,5,2,4,2).
(b) The ordinal integers represents the states as list (0,1,2,3,4,5,6,7); the list starts with zero and contains all possible assignments for states of an FSM using minimum

Table 2. Summary of available approaches to find optimal state assignment

Approach	Theory	Merits	Comments
Partition Theory Hartmans and Stearns 1966 [19].	Algebraic techniques are used to decompose the state machine.	The state assignment is based on closed partitions resulting in reduced dependency between the state variables.	Not all machines have close partitions.
Column evaluation approach. Dolota and McCuskey 1964 [20].	The column of state table are scored with respect to various criteria to influence quality of assignment	This approach gives very good realisation, the result even better than approach [19].	It is intractable for machine of more than 12 states.
Enumerative approach of Story 1972 [12].	All possible partitions are evaluated as candidates for assignment separately by calculating the complexities of the corresponding Boolean function.	All partitions are optimally selected and the subset of best partition equally matching is selected for assignment.	The implementation of approach is slow.
Approach of Moroz 1970 [21].	The graph embedding created directly from the state graph whose edges corresponding to oriented transition between states.	The approach can be treated as approximate solution to quadratic assignment.	It doesn't solve the quadratic assignment but simply edge embedding problem.
Quadratic assignment approach. DeMicheli 1984 [22].	This approach is based on embedding some graphs created from FSM table to hypercube graphs.	This approach permits for realisation of machine up to 100 states.	The approach still can't be applied for machine with more than 100 states, and it doesn't take in to count output states for assignment.
Devadas "Mustang 1988" [23].	Commercial ECAD tools look for optimal state assignment.	Available	For multilevel implementation.
Approach of Villa "NOVA 1990" [15].	Commercial ECAD tools look for optimal state assignment.	Available	For two level implementation only.
"Genetic algorithm approach" Almaini and Amaral, 1995 [17, 18].	Using GA to generate optimal state assignment.	All possible assignment for a given problem maybe considered as search space of potential solution.	GA approach requires a set of operators for its implementation. The selection of these operators is crucial for the efficiency of the algorithm.
Proposed approach 2003	Using GA to generate optimal state assignment and conventional part of the circuit.	Combination of both GA and EHW for sequential logic circuit.	



possibles states	0	1	2	3	4	5	6	7
Validity	1	0	0	0	0	1	0	0

Random numbers	2	4	5	2	4	2		
States list	0	1	2	3	4	5	6	7
Chromosome	1	4	6	2	7	3		
State assignment	001	100	110	010	111	011		

Figure 2. Chromosome representation of state assignment.

length code, i.e. $b = [\log_2 n]$ bits to encode the set of n states. The content of the state list represents state assignments. The algorithm works through the status-validity table initially set to one. The numbers are counted from left to right.

The procedure is interpreted as follows: (1) The first random number is 2, take the second number from the possible state list 1 as first code for the initial state and set validity (0), so it will not be used for future selections; (2) The next random number is 4, take the fourth number from the possible state list and remove it from the list by setting validity (0); (3) The next random number is 5, take the fifth number from the possible state list and set validity (0).

The procedure continues in the same way for the remaining numbers in the list. It can be seen from the figure that the random number 2,4,5,2,4,2 would map the states 0,1,2,3,4,5,6,7 to the assignment 1,4,6,2,7,3 respectively to assign a unique code to each state. This method is applied to generate randomly the initial generation and is similar to the ordinal list described in [24].

4.2. *Fitness function*

The goal of the GA is to extract the optimum state assignment for the target state machine, which requires the least number of logic gates. Therefore the number of 2-inputs AND/OR logic gates are used to define the fitness function.

4.3. *Genetic operators*

Fitness value is assigned to each individual in the population. Roulette wheel selection is used to select the chromosomes from the previous population. Then, the recombination operations are applied. Two-point crossover operation has been applied to produce child chromosome as illustrated in Figure 3.

The mutation operation chosen is based on the interchange of two genes (states) in each chromosome. However, when creating a new population by crossover & mutation, the best chromosome can be lost. In order to avoid this situation, elitism is utilised to process the best chromosomes in the new population. Elitism rapidly improves the performance of the GA, by preventing the loss of the best-found

Before Crossover	After Crossover
Chromosome 1= 2 1 | **3 5** | 6	Offspring1= 2 1 | 7 3 | 6
Chromosome 2= 4 2 | 7 3 | 1	Offspring2= 4 2 | **3 5** | 1

Figure 3. The two-point crossover operation. Two offspring are produced from a pair of parents. The "|" symbol indicates the randomly chosen crossover point.

solution. Several parameters control the way GA optimises the state assignment of the FSM, allowing the users to vary their values. The parameters are:

- The population size of the genetic algorithm;
- The number of generations of the GA around the main loop;
- The initial numbers of runs of the GA to perform the optimisation;
- The probabilities of crossover rate (P_c) and mutation rate (P_m).

Selecting the GA parameter values for optimal state assignments will be discussed in Section 6. The mutation rate was variable and increased with each generation if there had been no improvement in the gate count of the best chromosome.

5. Extrinsic Evolvable Hardware approach for combinational logic design

Extrinsic EHW initially proposed in [26] have been used to design the combinational part of circuit. The EHW approach is a recently developed technique used to synthesize combinational and sequential circuits [11, 25, 26]. The automated design of digital systems uses both software simulation and programmable hardware techniques. Therefore, EHW approach is based on the idea of combining reconfigurable hardware devices with GA to perform reconfiguration autonomously.

The state transition table (STT) has been chosen to describe the behaviour of the synchronous sequential logic circuit. The structure of sequential logic circuits shown in Figure 4 comprises a set of two sections of combinational logic circuit and D flip-flops (DFFs). The circuit is generated using a given set of available logic gates. The combinational parts of sequential logic circuits are generated using extrinsic EHW approach. The desired functionality for the combinational parts of the logic circuit is described using STT. The search space is defined by a number of different factors: (1) type of building blocks presented to the framework; (2) the number of logic elements used to generate the circuit; (3) the application for which the circuit is being evolved.

In the design process, it has been long accepted that the best way to solve a problem is to decompose the problem into several sub-problem. The structure of a sequential logic circuit in the proposed approach contains 3 subcircuits as shown in Figure 4. Each subcircuit is evolved separately. Once the subcircuits have been designed, the sequential circuit is assembled. In this case, the 2-combinational logic circuits A and B have to be synthesised. Full details of extrinsic EHW approach to design combinational logic circuits can be found in [25, 26] but the approach is summarised here for convenience. Each combinational circuit is represented as a rectangular array of logic gates. Each logic gate in this array is uncommitted and can be removed from the network if it is proved redundant. The genotype is

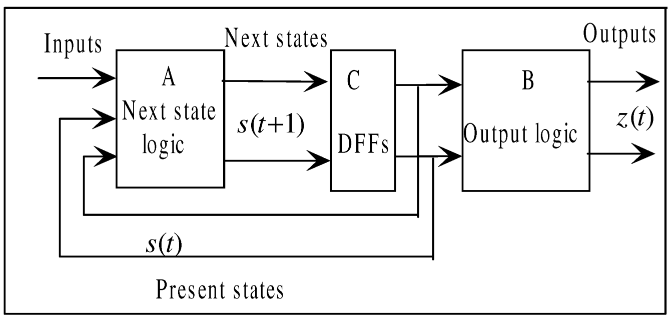


Figure 4. Description on the circuit parts.

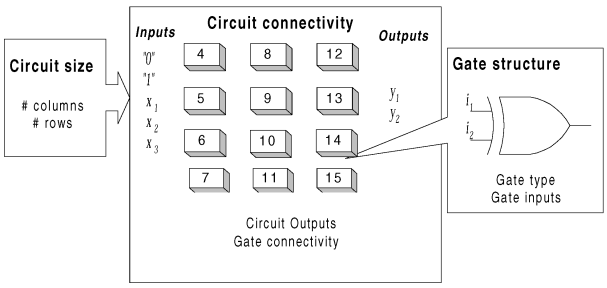


Figure 5. Schematics of the chromosome structure used in EHW approach with circuit layout 3×4 [26].

characterised by the following parameters: the number of columns, the number of rows and the level back parameter and connectivity list of rectangular array of logic gates [26] (see Figure 5). The first two parameters are merely the dimension of the rectangular array and the third one is a parameter, which controls the internal connectivity of logic circuit. The maximum cell connectivity can be achieved if the number of rows is one and the level back parameter is equal to the number of columns. At the same time, if the number of rows is one and the level back parameter is one then each cell must be connected to its immediate neighbour to the left. Further, cells within any particular column cannot be connected to each other and each logic gate has two-inputs and one output.

The chromosome defines the connection in the network between the primary inputs and primary outputs. Figure 5 shows the representation of the chromosome connection between the 3-primary inputs and 2-primary outputs. The network is designed using 2-input, one-output logic gates. The chromosome layout is 3×4 ($n_{\text{columns}} \times n_{\text{rows}}$) geometry of uncommitted logic cells and netlist numbering. Each logic cell is represented by a triple of integers $\langle c^1\ c^2\ c^3 \rangle$, where c^1 defines functional gene and c^2, c^3 define the gate inputs.

5.1. *Fitness function*

Dynamic fitness function ($F_1 + F_2$) is used to evaluate the circuit [26]. F_1 uses Hamming distance to measure the functionality of the circuit between a given set of inputs and outputs. F_2 defines the number of primitive logic cells that are used in the circuit. F_2 is activated once F_1 reaches 100% functionality.

The first fitness function compares the corresponding output of subcircuit A and subcircuit B (Figure 4) with given next state and output columns of STT. The percentage of correct next state bits corresponding to the j-th output, Fy_j is calculated as follows: $Fy_j = (\sum_{j=1}^{p} |y_j - d_j|/p) * 100$; where $|y_j - d_j|$ is the absolute difference between the actual next state output and the desired output d_j, y_j is the vector of the j-th circuit outputs and p is the number of input combinations in the given logic function. The circuit completely implements the output y_j reaches 100% functionality. Once the solution has been evolved the circuit optimisation criteria is activated. The second fitness function F_2 minimises the number of logic gates by rewarding those circuits with the least number of active logic gates. The procedure described above applies for both subcircuits A and B.

5.2. *Motivation examples*

Let us consider the design process of the sequential circuit based on the synthesis of the symbolic transition table given in Figure 6. The FSM has five states, one input and one output. In Figure 6 step 1 shows the symbolic state table of FSM and the state assignment generated by GA was assigned to each state. In step 2, the encoding is used to obtain the standard two-level PLA format.

In step 3, the STT of the circuit is divided into input combinational logic subcircuit A and output combinational logic subcircuit B. Once the decomposition is completed, the fully functional circuits can be generated using EHW.

The initial data of evolutionary algorithms parameters is given in Table 4 and circuit evolved is shown in Figure 7. The total number of logic gates in the assembled circuit is 10 (6 AND, 3 OR, 1 NOT). The most efficient evolved subcircuit consists of 3 logic gates in subcircuit A, 7 gates in subcircuit B and 3 D flip-flops. The initial parameters of GA to generate optimal state assignment (SA) to construct the STT and extrinsic EHW to design the desired logic circuits are given in

i/p	Ps	Ns	o/p
0	S0	S1	0
0	S1	S4	1
0	S2	S4	1
0	S3	S4	0
0	S4	S0	0
1	S0	S2	0
1	S1	S3	1
1	S2	S3	0
1	S3	S4	1
1	S4	S0	0

i/p inputs
o/p outputs
Ps Present state
Ns Next state

State 0 = 0 0 0
State 1 = 0 0 1
State 2 = 1 0 1
State 3 = 1 1 0
State 4 = 0 1 0

Step 1

STT of the circuit

```
.i   4
.o   4
.p   10

0000        001 0
0001        010 1
0101        010 1
0110        010 0
0010        000 0
1000        101 0
1001        110 1
1101        110 0
1110        010 1
1010        000 0
.e
```

Step 2

STT of subcircuit A

```
.i   4
.o   3
.p   10

0000        001
0001        010
0101        010
0110        010
0010        000
1000        101
1001        110
1101        110
1110        010
1010        000
.e
```

STT of subcircuit B

```
.i   4
.o   1
.p   10

0000        0
0001        1
0101        1
0110        0
0010        0
1000        0
1001        1
1101        0
1110        1
1010        0
.e
```

Step 3

Figure 6. The transformation process of STT into PLA file format. Where .i inputs = input + present state bits, .o defined the number of outputs calculated, outputs = next state + output bits, .p is the number of product terms, .e is end of file.

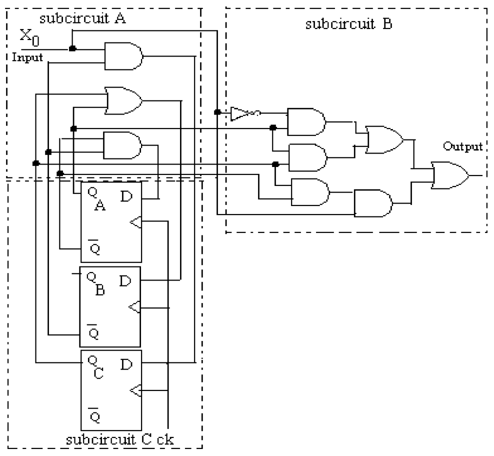


Figure 7. The circuit structure implemented according to state table given in Figure 6.

Table 3. Functional set of logic gates used in EHW

Gene	Function gene	Gene	Function gene	Gene	Function gene	Gene	Function gene
0	"0"	4	!a NOT (a)	8	!ab AND (!a, b)	12	!a|b OR (!a, b)
1	"1"	5	!b NOT (b)	9	!a!b AND (!a, !b)	13	!a|!b OR (!a, !b)
2	"a" wire	6	ab AND (a, b)	10	a|b OR (a, b)	14	a^b XOR (a, b)
3	"b" wire	7	a!b AND (a, !b)	11	a| !b OR (a, !b)	15	!a^!b XOR (!a, !b)

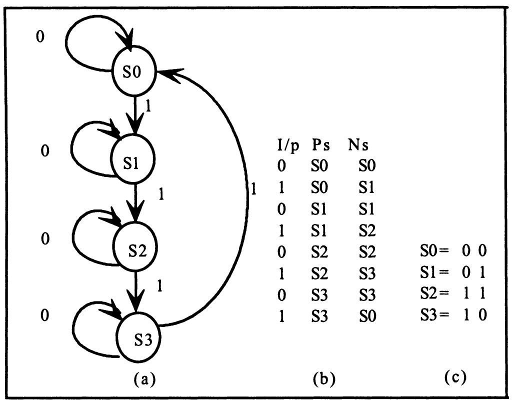
Table 4. Initial parameters used to evolve the circuit

Parameters	SA	EHW
Population size	20	10
Number of generations	100	50000
Number of GA runs	10	100
Crossover rate	0.25	0.6
Crossover type	Two-point	Uniform
Mutation rate	0.015	0.05
Circuit layout	—	3×4

Table 4. The functional set of logic gates contains all gates encoded from 0 to 15 (see Table 3).

6. Experimental results

In this section, the circuit structure synthesised using the proposed approach is considered and compared to manual designs. We present the problem of designing a digital circuit in terms of a set of examples. Where in this experimental result, the GA parameters in Table 4 are used for the state assignment problem. A number of



I/p	Ps	Ns
0	S0	S0
1	S0	S1
0	S1	S1
1	S1	S2
0	S2	S2
1	S2	S3
0	S3	S3
1	S3	S0

S0=	0 0
S1=	0 1
S2=	1 1
S3=	1 0

Figure 8. Module-4 counter (a)-state transition graph, (b) State table and (c) State assignment generated by GA.

experiments have been carried out in order to determine suitable value for the EHW parameters.

6.1. *Example 1: Module-4 counter*

The simplest synchronous digital system is the binary counter because it has no input and no combinational output block. Module-4 counter has four internal states as shown in Figure 8. The optimal state assignment has been identified using GA as in Figure 8(c). The evolutionary algorithm parameters to design the combinational parts of the GA are: the population size is 5, the maximum number of generation is 50000, the total of runs is 100, the crossover rate is 0.6, the mutation rate is 0.05, layout described by 1×10 in proposed approach (a) and by 4×4 in proposed approach (b) and the resulting equations are given in Table 5.

In this example, small size of population and large numbers of generations are used. The circuit shown in Figure 9 has also been tested using two layout sets with the same state assignment. Choosing too small circuit layout runs the risk that no 100% functional solution could be found because it is physically impossible to build the circuit of required functionality with few logic gates. Choosing too large a circuit layout gives the evolutionary algorithm too many possibilities to work with. This is has been proved for combinational logic circuit in [26].

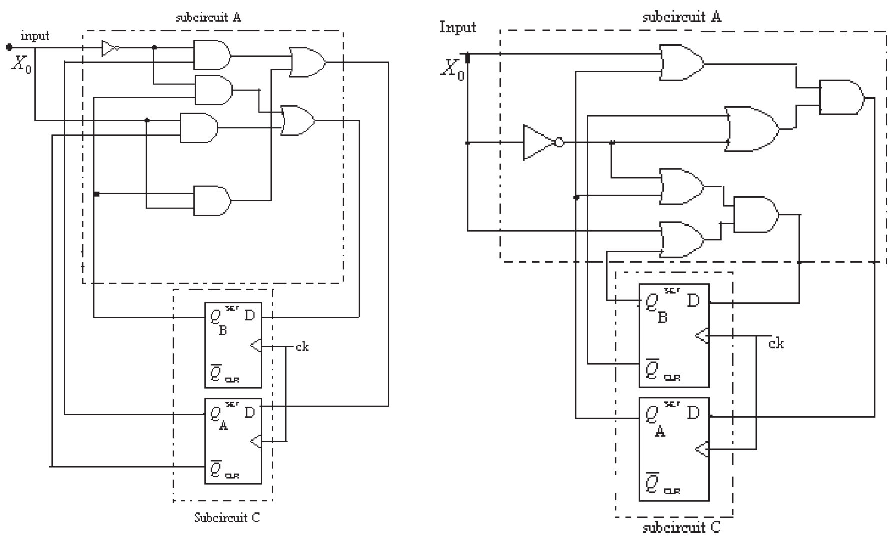
Figure 9 show how the number of the columns and rows influences the evolution of circuits. The search space of evolutionary algorithm increases with the number of rows and columns. It can be seen that in Figure 9(b) the evolved optimal circuit solution with circuit layout 4×4 required 2 AND, 4 OR and 1 NOT logic gates.

6.2. *Example 2: sequence detector*

The sequence detector circuit has one-input, one-output and 6-internal states. The behaviour of circuit can be described as shown in Figure 10. When the input

Table 5. Solution obtained for Module-4 counter produced using proposed method and manual design

Proposed approach (a)	Proposed approach (b)	Manual method
$D_A = \bar{X}_0 A + X_0 B$	$D_A = \bar{X}_0 A + X_0 \bar{B}$	$\mathrm{D_A} = \bar{\mathrm{X}}_0\mathrm{A} + \mathrm{X}_0\bar{\mathrm{B}}$
$D_B = X_0 \bar{A} + \bar{X} B$	$D_B = X_0 A + \bar{X}_0 B$	$\mathrm{D_B} = \mathrm{X}_0\mathrm{A}\bar{\mathrm{B}} + \bar{\mathrm{X}}_0\mathrm{B} + \mathrm{X}_0\mathrm{A}$
Subcircuit A = 7 gates	Subcircuit A = 7 gates	*Subcircuit A = 9 gates*
Subcircuit C = 2-D flip-flops	*Subcircuit C = 2 D flip-flops*	*Subcircuit C = 2 D flip-flops*



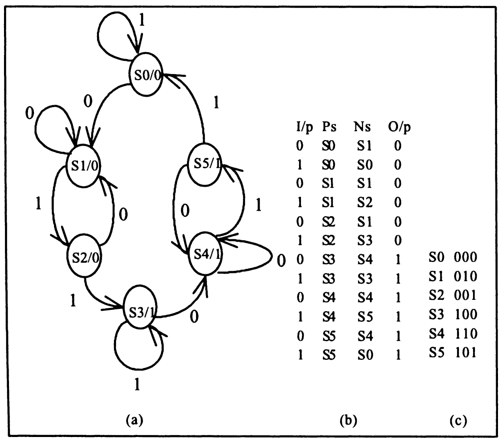
(a) Layout 1x10 (b) Layout 4x4

Figure 9. The evolved optimal circuit solution of the model-4 counter.

sequence 011 occurs, the outputs become 1 and remains 1 until the sequence 011 occur again. In this case, the output returns to 0. The output then remains 0 until the sequence 011 occurs a third time, etc. The comparison of result produced by the proposed approach and manual design based on GA state assignment are shown in Table 6.

Once the EHW generate the same output as the target machine, the sequential network of subcircuit A can be structured using 5 AND, 2 OR, 1 NOT and subcircuit C implemented by 3 D flip-flops as shown in Figure 11. The manual design uses a random state assignment and Karnaugh maps to minimize the equations. The parameters of the evolutionary algorithm for this example are as follows: the cell mutation rate is 0.05, the population size is 20, and the maximum numbers of generations is 50000. A circuit layout structure of 4 rows and 5 columns is used with maximum level back parameter equal to 5. The parameter help to obtain a reasonable probability of achieving 100% correct solution.

Analysing the evolved circuit structure in Figure 11 it can be seen that the resulting circuit has a more efficient structure than the circuit obtained by manual method.



I/p	Ps	Ns	O/p
0	S0	S1	0
1	S0	S0	0
0	S1	S1	0
1	S1	S2	0
0	S2	S1	0
1	S2	S3	0
0	S3	S4	1
1	S3	S3	1
0	S4	S4	1
1	S4	S5	1
0	S5	S4	1
1	S5	S0	1

S0	000
S1	010
S2	001
S3	100
S4	110
S5	101

Figure 10. A sequence detector described as (a) state transition graph, (b) state table, (c) state assignment.

Table 6. Solutions obtained for sequential detector produced by proposed approach and manual method

Proposed approach	Manual method
$D_A = XB$	$\mathrm{D_A = A\bar{C} + \mathit{A}\bar{X} + BC\bar{X}}$
$D_B = \bar{X}$	$\mathrm{D_B = BX + \bar{A}CX}$
$D_C = XA\bar{C} + \bar{X}C + \bar{A}C$	$\mathrm{D_C = BX + \bar{A}\bar{C}\bar{X} + \bar{A}\bar{B}\bar{X} + A\bar{C}X}$
$Z = C$	$\mathrm{Z = A + BC}$
Subcircuit A = 8,	Subcircuit A = 17,
Subcircuit B = 1	Subcircuit B = 2
Subcircuit C = 3 D flip-flops	Subcircuit C = 3 D flip-flops

6.3. Example 3: 1010 detector

The 1010 detector circuit has one input, one output and 4 internal states, as shown in Figure 12. The results produced by the proposed method are compared with results produced in [1] as shown in Table 7.

The solution reported in [1] uses almost 3 times more gates than the circuit produced by the proposed approach. It is interesting to note that the outputs are implemented in the same way for both circuits. The circuit was actually evolved by separating each subcircuit and evolving each subcircuit separately to obtain the target circuit. The probability of the cell mutation rate is 0.05, the population size is 20 and the number of generations is 50000 with 1×10 circuit layout. The resulting circuit is given in Figure 13.

6.4. MCNC benchmarks

In this section the experiment results have been determine with the application to a set of machines chosen from MCNS benchmark set [28]. Table 8 shows the state assignment generated by GA.

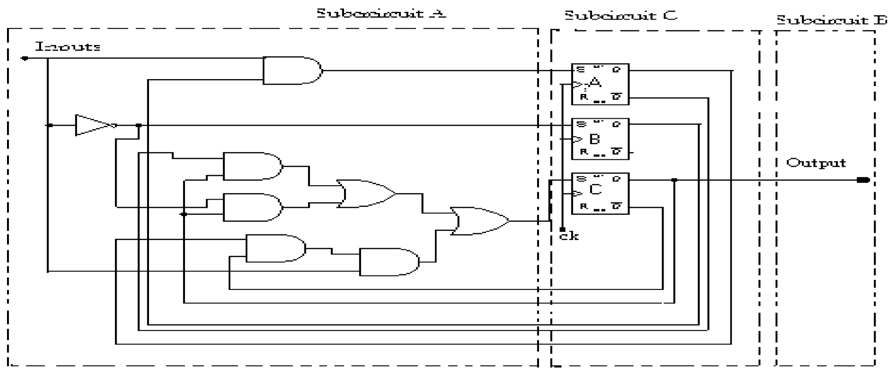
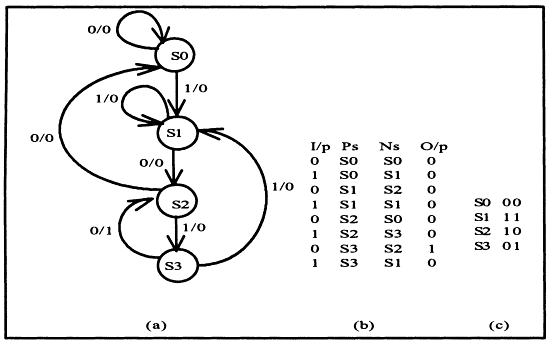


Figure 11. Evolved optimal circuit solution of the sequence detector with circuit layout 4×5.



I/p	Ps	Ns	O/p
0	S0	S0	0
1	S0	S1	0
0	S1	S2	0
1	S1	S1	0
0	S2	S0	0
1	S2	S3	0
0	S3	S2	1
1	S3	S1	0

S0	00
S1	11
S2	10
S3	01

Figure 12. 1010 Detector (a) state transition graph, (b) state transition table, (c) state assignment.

Table 7. Solution obtained for 1010 detector produced using proposed method and manual design

Proposal approach	Almaini, 1994 [1]
$D_A = X\bar{B} + A$	$D_A = \bar{X}\bar{A}B + \bar{X}A\bar{B} + XAB$
$D_B = X$	$D_B = \bar{A}B + A\bar{B} + X\bar{B}$
$Z = \bar{X}\bar{A}B$	$Z = \bar{X}A\bar{B}$
Subcircuit A = 2	Subcircuit A = 12
Subcircuit B = 3	Subcircuit B = 2
Subcircuit C = 2 D flip-flops	Subcircuit C = 2 D flip-flops

As usual EHW begins from randomly connected and randomly chosen logic gates and gradually evolves the target functionality. Neutrality of evolutionary algorithms does not guarantee that 100% functionality circuit of the resulting connections will be achieved in all cases so the results reported here are the average from 100 runs.

Table 9 present the experimental results obtained for general FSMs from MCNC benchmark set. The table shows the numbers of gates used to evolve each subcircuit after 100 runs. The particular set of logic gates used is fixed in advance, but whether

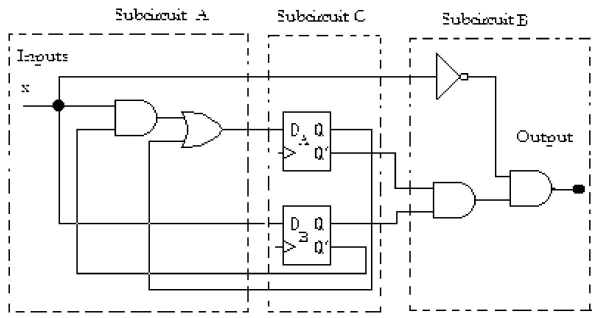


Figure 13. Evolved optimal circuit solution for 1010 detector using 4 × 5-circuit layout.

Table 8. State assignments generated by GA

FSM	#State	State assignment
bbara	10	2,3,5,4,7,8,9,0,1,11
bbtas	6	6,1,5,4,3,0
dk15	4	0,2,1,3
dk16	27	12,8,1,27,13,28,14,29,0,16,26,9,2,4,3,10,11,17,24,5,18,7,21,25,6,20,19
dk27	7	6,1,5,7,4,3,0
dk512	14	4,3,14,9,12,7,2,1,0,10,13,8,5,6
lion9	9	1,0,4,6,7,53,1,11
shiftreg	8	6,2,4,0,7,3,5,1
tav	4	4,0,2,1

or not any particular gate is used, or how many time a gate is used, is entirely free. The advantage of this approach is that it allows us to synthesis the benchmarks circuit using any set of logic gates. Consequently, it permits the synthesis of compact and unusual circuit structures. The quality of evolved circuits is defined by the number of logic gates in the circuit. It can be seen from Table 9 that large FSM benchmarks (dk16) is difficult to evolve with one valid solution after 100 runs. These benchmark sets results are compared against SIS [5] for sequential logic synthesis and optimisation. The inputs to SIS are given in state table kiss format and the library is given in genlib format. The output is a netlist of gates for the target technology. The table shows the measure of computing time t_{EHW} run on a 450 MHZ PC 128 MBRAM for the benchmark that appears in the literature. The result shows that the benchmark with small numbers of states the EHW required significantly less CPU run time. Furthermore, It may be conclude that the results found by this approach are at least as good as manual methods, but in some case better than those derived by available methods.

7. Conclusions

The paper presents a new synthesis approach which provide for both GA to find optimal state assignment and EHW to design the combitional part of sequential logic

Table 9. Experimental results of extrinsic EHW approach

	Specification				Estimation of the best obtained solution					SIS [5]	CPU Time (S)
Benchmark. Kiss	#in	#out	#stat	Functional set	Subcircuit A	Subcircuit B	Subcircuit C	Total	#100 cases	Total	t_{EHW}
Bbara	4	2	10	0–5,6,10,15	32	28	3	60	7	79	660
Bbtas	2	2	6	2–7,10,11,15	15	4	3	19	24	28	480
dk15	3	5	4	0,1,6,7,10,11–15	20	33	2	53	11	66	300
dk16	2	3	27	0,6,8,10–15	265	40	5	305	1	285	3180
dk27	1	2	7	0–6,10,15	11	5	3	16	28	20	420
dk512	1	3	14	2–6,9–14,15	25	22	4	47	31	58	484
Lion9	2	1	9	0–6,10,15	29	21	4	50	7	35	410
Shiftreg	1	1	8	0–13,15	13	5	3	18	21	12	100
Tav	4	4	4	0–6,10,15	3	23	2	26	9	29	360

#100 cases is the number of fully functional solutions obtained after 100 runs of GA circuit.The EHW approach described is based on the exterinsic evolution at gate-level, in the sense that each gene of a chromsome corresponds to a primitive logic gate. The problem of how to evolve a sequential logic circuit that performs a desired function (specified by state table); given a set of available logic gates has been discussed. The experimental results obtained are compared with results produced by the manual design method and other automted design tools. The minimum numbers of AND, OR, NOT logic gates in the combinational block of the circuit is the criteria set by the user to choose the optimal solution. The state assignment of state machine is often critical and a small change in the codes assigned to the state can lead to very wide difference in the number of logic gates and in the topological structure of that logic. The work shows how an evolutionary algorithm could be used to produce a novel and efficient design for digital logic circuit. As it has been verified through the presented EHW approach implementations, the proposed approach can be successfully applied to the design of simple sequential logic circuits. We believe that the GA-based approach has a great potential to provide a practical tool for assisting designers of logic circuits.

A common feature of most published work in this field is that the evolved circuit size is small and hardware evolution is based on logic gates. Future work on EHW will concentrate on using function-level EHW to evolve larger finite-state machine.

References

1. A. E. A. Almaini, Electronic Logic Systems, Prentice-Hall, 3rd Ed. 1994, UK.
2. P. K. Lala, Practical Digital Design and Testing, Prentice Hall, 1996.
3. P. Chongstitvatana and C. Aporntewan, “Improving correctness of finite-state machine synthesis from multiple partial input/output sequences,” in Proceedings of the 1st NASA/DoD Workshop of Evolvable Hardware, 1999, pp. 262–266.
4. S. Louis, “Genetic algorithm as computational tool for design,” PhD Dissertation, Department of Computer Science, Indiana University, 1993.
5. E. M. Sentovich, et al. “SIS. A system for sequential circuit synthesis,” Tech. Rep. UCB/ERL M92/41 Electronics Research Lab. University of California: Berkeley, CA 94720, May 1992.
6. A. Thompson, “An evolved circuit, intrinsic in silicon, entwined with physics,” in Proceedings of the 1st International Conference on Evolvable Systems: From Biology to Hardware (ICES96), Lecture Notes in Computer Science, T. Higuchi, et al. (eds.) Springer-Verlag, vol. 1259, 1997, pp. 390–405.
7. H. Hemmi, J. Mizoguchi, and K. Shimohara, “Development and evolution of hardware behaviours,” Brooks and Maes (eds.), Artificial Life IV, pp. 371–376, 1994.
8. T. Kalganova, “Bidirectional incremental evolution in evolvable hardware,” in Proceeding of the Second NASA/DoD Workshop on Evolvable Hardware J. Lohn, A. Stoica, D. Keymeulen, and S. Colombano (eds.), Palo Alto, California, USA. Published by IEEE Computer Society, 2000, pp. 64–74.
9. J. Koza, F. Bennett, D. Andre, and M. Keane, Genetic Programming III: Darwinian Invention and Problem Solving, Morgan Kaufman, 1999.
10. C. Manovit, C. Aporntewan, and P. Chongstitvatana, “Synthesis of synchronous sequential logic circuits from partial input/output sequence,” in Proceedings of the Second International Conference on Evolvable Systems (ICES’98) M. Sipper, D. Mange, and Andrés Pérez-Uribe (eds.), Springer-Verlag: Heidelberg, vol. 1478, 1998, pp. 98–105.
11. A. Thompson, “Evolving electronic robot controllers that exploit hardware resources,” in Proceeding 3rd European Conference on Artificial Life (ECAL 95) A Moreno, J. J. Merelo, and P. Chacon (eds.), Springer-Verlag, vol. 929, 1995, pp. 640–656.

12. J. Story, H. Harrison, and E. Reinhard, “Optimum state assignment for synchronous sequential circuit,” IEEE Trans. On Comp, vol. C-21, no. 12, pp. 1365–1373, 1972.
13. D. B. Fogel, Evolutionary Computation, IEEE Press, pp. 75–84, 1995.
14. T. Higuchi, T. Niwa, T. Tanaka, H. Iba, and T. Furuya, “A parallel architecture for genetic based evolvable hardware,” in Proceeding of International Joint Conference on Artificial Intelligence (IJCAI’93), H. Kitano, C. Suttner, and V. Kumar (eds.), Workshop on Parallel Processing for Artificial Intelligence, 1993, pp. 46–52.
15. T. Villa and A. Sangiovanni-Vincentelli, “NOVA: state assignment of finite state machines for optimal two level logic implementation,” IEEE Trans., C-9, pp. 905–924, 1990.
16. B. Lin and A. R. Newton, “Synthesis of multi-level logic from Symbolic High-level Description Language,” in Proceeding IFIP Conference on VLSI, J. M. Luc (ed.), Munich: W. Germany, 1989, pp. 187–196.
17. A. E. A. Almaini, J. Miller, P. Thomson, and S. Billina, “State assignment of state machine using genetic algorithm,” in IEE Proc. Comp. and Digital Techniques, no. 4, July 1995 vol. 142, 1995, pp. 279–286.
18. J. Amaral, K. Tumer, and J. Ghosh, “Design genetic algorithm for the state assignment problem,” IEEE Trans, vol. no. SMC-25, 4, pp. 689–694, 1995.
19. J. Hartmanis and E. Stearns, Algebraic Structure Theory of Sequential Machines, Prentice Hall, 1996.
20. T. Dolotta and E. McCluskey, “The Coding of Internal States of Sequential Machines,” IEEE Transaction on Electron. Comput, vol. EC-13, pp. 549–562, October 1964.
21. D. Z. Moroz, “An algorithm for encoding the states of an automaton,” Avtomatikai Vychislitelnaya Tekhnika, vol. 4, no. 4, pp. 21–24, 1970.
22. G. De Micheli, “Optimal Encoding of Control Logic,” Int. Conf. on Circ. and Comp. Des. Rye NY, September 1984.
23. S. Devadas, A. Newton, and V. Sangiovanni, “MUSTANG: State assignment of finite state machines targeting multilevel logic implementations,” IEEE Trans. Computer-Aided Design, vol. CAD-7, No.12, pp. 1290–1300, December 1988.
24. Z. Michalewicz, “Genetic Algorithms + Data Structure = Evolution Program,” Springer-Verlag, 1992.
25. T. Kalganvoa, J. Miller, and T. Fogarty, “Evolution of the digital circuit with variable layouts,” in Proceeding of the Genetic and Evolutionary Computation Conference (GECCO’99), W. Benzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith (eds.), Orlando, Florida, USA, Published by Morgan Kaufmann Publishers: San Francisco, California 1, vol. 2, 1999, p. 1235.
26. T. Kalganvoa and J. Miller, “Circuit layout evolution: an evolvable hardware approach,” Colloquium on “Evolutionary hardware systems,” IEE Colloquium Digest: London, UK, 1999.
27. C. Aporntewan and P. Chongstitvatana, “An on-line evolvable hardware for learning finite-state machine,” in Proceeding of Int. Conf. on Intelligent Technologies, Bangkok, December 13–15 V. Kreinovich and J. Daengdej (eds.), 2000, pp. 13–15.
28. S. Yang, Logic synthesis and optimisation benchmark user guide version 3.0, MCNC, 1991.