Journal of Universal Computer Science, vol. 14, no. 15 (2008), 2532-2548
submitted: 22/2/08, accepted: 30/6/08, appeared: 1/8/08 © J.UCS

# Quantum-Inspired Evolutionary State Assignment for Synchronous Finite State Machines

**Marcos Paulo Mello Araujo**
(Department of Electronics Engineering and Telecomunications
Engineering Faculty, State University of Rio de Janeiro
Rio de Janeiro, Brazil
marcos@eng.uerj.br)

**Nadia Nedjah**
(Department of Electronics Engineering and Telecomunications
Engineering Faculty, State University of Rio de Janeiro
Rio de Janeiro, Brazil
nadia@eng.uerj.br)

**Luiza de Macedo Mourelle**
(Department of Systems Engineering and Computation
Engineering Faculty, State University of Rio de Janeiro
Rio de Janeiro, Brazil
ldmm@eng.uerj.br)

**Abstract:** Synchronous finite state machines are very important for digital sequential designs. Among other important aspects, they represent a powerful way for synchronizing hardware components so that these components may cooperate adequately in the fulfillment of the main objective of the hardware design. In this paper, we propose an evolutionary methodology to solve one of the problems related to the design of finite state machines. We optimally solve the state assignment $NP$-complete problem using a quantum inspired evolutionary algorithm. This is motivated by the fact that with an optimal state assignment one can physically implement the state machine using a minimal hardware area and response time.

**Key Words:** Quantum computation, finite state machine, state assignment problem.
**Category:** B.1.2, B.2.2, I.2.2

## 1 Introduction

In general, implementation of sequential digital systems consist of a datapath and controller. The datapath is a set of registers, counters and multiplexers and the controller is an implementation of the finite state machine (FSM) that models the required behavior. Implementation of FSMs, as depicted in Fig. 1, have two main characteristics: there is at least one feedback path from the system output signal to the system input signals; and there is a memory capability that allows the system to determine current and future output signal values based on the previous input and output signal values [Rhyne 1973]. Since a machine state is nothing but a counting device, combinational control logic is necessary to activate the flip-flops in the desired sequence.

In Fig. 1, the feedback signals constitute the machine state, the control logic is a combinational circuit that computes the state machine output signals (also called *primary output signals*) from the state signals (also called *current state*) and the input signals (also called *primary input signals*). It also produces the signals of new machine state (also called *next state*). Traditionally, the design process of a state machine goes through five main steps:

1. Specification of the sequential system, which should determine the next states and outputs for every present state of the machine. This is done using state tables and state diagrams;
2. State reduction, which should reduce the number of present states using equivalence and output class grouping;
3. State Assignment, which should assign a distinct binary combination to every state. This may be done using Armstrong-Humphrey heuristics [Armstrong 1962] [Humphrey 1958];
4. Minimization of the control combinational logic using K-maps and transition maps for the used flip-flops;
5. Implementation of the state machine, using gates and flip-flops.

In this paper, we concentrate on the third step of the design process, i.e. the state assignment problem. We present a quantum inspired evolutionary algorithm designed for finding a state assignment of a given synchronous finite state machine, which attempts to minimize the cost related to the state transitions.

The remainder of this paper is organized into sixsections. In Section 2, we introduce the problems that face the designer of finite state machine, which are mainly the state assignment problem and the control logic. We show that a better assignment improves considerably the cost of the control logic. In Section 3, we give an overview on the principles of quantum computation. In Section 4 we introduce the quantum inspired evolutionary algorithm and their application to solve NP-problems. In Section 5, we design a quantum inspired evolutionary algorithm for evolving best state assignment for a given state machine specification. We describe the state assignment encoding, the variation operator used as well as the fitness function, which determines whether a state assignment is better than another and how much. In Section 6, we present results evolved through our quantum inspired evolutionary algorithm for some well-known benchmarks. Then we compare the obtained results with those obtained by the genetic algorithms described in [Amaral et al. 1995] and [Nedjah and Mourelle 2004] as well

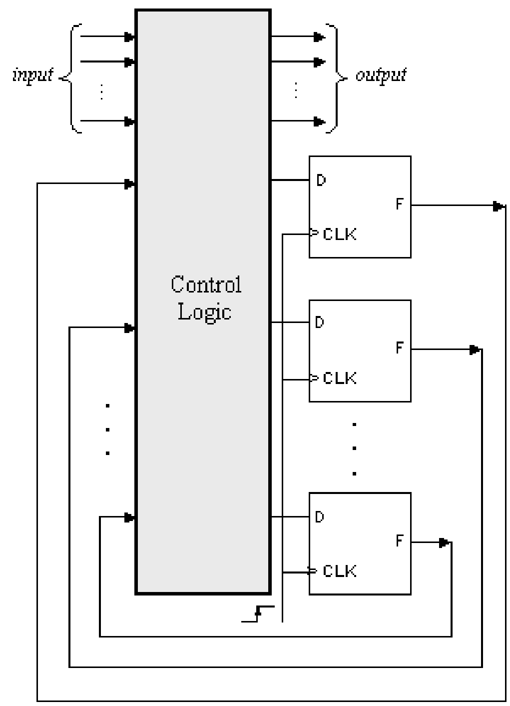



**Figure 1:** The structural description of a finite synchronous state machine

as to NOVA, which uses well established but non-evolutionary method [Villa and Sangiovanni-Vincentelli 1990]. In Section 7, we summarise the ideas presented throughout the paper and draw some conclusions.

## 2 State Assignment Problem in FSMs

Once the specification and the state reduction step have been completed, the following step consists of assigning a code to each state present in the machine. It is clear that if the machine has $N$ distinct states then one needs $N$ distinct binary combinations. So one needs $K$ flip-flops to store the machine current state, wherein $K$ is the smallest positive integer such that $2^K \geq N$. The state assignment problem consists of finding the best assignment of the flip-flop combinations to the machine states.

The control logic component in a state machine is responsible of generating the primary output signals as well as the signal that allow generating the next state. It does so using the primary input signals and the signals that constitute the current state. Traditionally, the combinational circuit of the control logic is

**Table 1:** Example of state transition function

| Present | Next State | | Output ($O$) | |
|---|---|---|---|---|
| State | $I = 0$ | $I = 1$ | $I = 0$ | $I = 1$ |
| $s_0$ | $s_0$ | $s_1$ | 0 | 0 |
| $s_1$ | $s_2$ | $s_1$ | 0 | 1 |
| $s_2$ | $s_0$ | $s_3$ | 1 | 0 |
| $s_3$ | $s_2$ | $s_1$ | 1 | 1 |

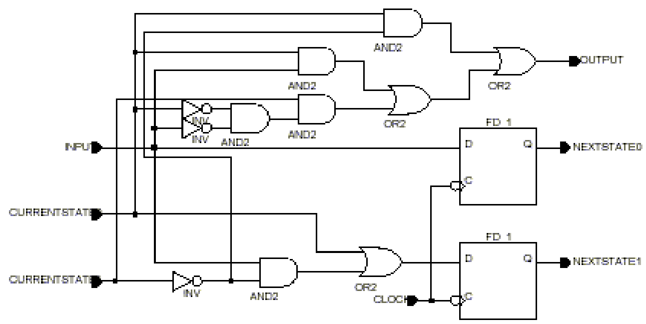



**Figure 2:** FSM schematics for different state assignment $A_0$

obtained using the transition maps of the flip-flops [Rhyne 1973]. The complexity of the control logic depend heavily on the outcome of the state assignment step. For instance, consider the state machine of one input signal ($I$), one output signal ($O$) and four states whose state transition function is given in tabular form in Table 1 and assume that we use D-flip-flops to store the machine current state. Then the state assignment $A_0 = \{s_0 \equiv 00, s_1 \equiv 11, s_2 \equiv 01, s_3 \equiv 10\}$ requires a control logic that consists of three NOT gates, five AND gates and three OR gates while the assignments $A_1 = \{s_0 \equiv 00, s_1 \equiv 10, s_2 \equiv 01, s_3 \equiv 11\}$ requires a control logic that consists of only two NOT gates, five AND gates and two OR gates. The schematics of the state machines that encode the state according to state assignments $A_0$ and $A_1$ are given in Fig. 2 and Fig. 2 respectively.

Given a state transition function, the requirements of area and time vary for different assignments of flip-flop combinations to allowed states. Therefore, so does the cost of the controller as a whole. Consequently, the designer or the Computer-Aided Design tool for hardware synthesis should always seek the assignment that minimizes the complexity and so the cost of the combinational logic required to control the state transitions. In the rest of the paper, we concentrate on the state assignment problem. We present a quantum inspired evolu-

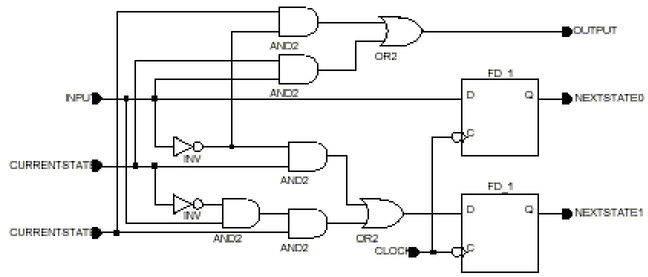



**Figure 3:** FSM schematics for different state assignment $A_1$

tionary algorithm designed for finding a state assignment of a given FSM, which attempts to minimize the cost related to the state transitions.

## 3 Principles of Quantum Computing

In quantum computing, the smallest unit of information stored in a two-state system is called a quantum bit or qubit [Hey 1999]. The 0 and 1 states of a classical bit, are replaced by the state vectors $|0\rangle$ and $|1\rangle$ of a qubit. This vectors are usually written using the *braket* notation, introduced by Paul Dirac. The state vectors of a qubit are represented as in (1)

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{e} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{1}$$

While the classical bit can be in only one of the two basic states that are mutually exclusive, the generic state of one qubit can be represented by the linear combination of the state vectors $|0\rangle$ and $|1\rangle$, as in (2)

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2}$$

wherein $\alpha$ and $\beta$ are complex numbers. The state vectors $|0\rangle$ and $|1\rangle$ form a canonical base and the vector $|\psi\rangle$ represents the superposition of this vectors, with $\alpha$ and $\beta$ amplitudes. The unit normalization of the state of the qubit ensures that (3) is true:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{3}$$

The phase of a qubit is defined by an angle $\zeta$ as in (4)

$$\zeta = \arctan(\beta/\alpha), \tag{4}$$

and the product $\alpha \cdot \beta$ is represented by the symbol $d$ and defined as in (5),

$$d = \alpha \cdot \beta, \tag{5}$$

where $d$ stands for the quadrant of qubit phase $\zeta$. If $d$ is positive, the phase $\zeta$ lies in the first or third quadrant; otherwise, the phase $\zeta$ lies in the second or fourth quadrant [Zhang et al. 2006].

The physical interpretation of the qubit is that it may be simultaneously in the states $|0\rangle$ and $|1\rangle$, which allows that an infinit amount of information could be stored in state $|\psi\rangle$. However, in the act of observing a quantum state, it collapses to a single state [Narayanan 1999]. The qubit collapses to state 0, with probability $|\alpha|^2$ or state 1, with probability $|\beta|^2$. A system with $m$ qubits contains information on $2^m$ states. The linear superposition of possible states can be represented as in (6)

$$|\psi\rangle = \sum_{k=1}^{2^m} C_k \left|S_k\right\rangle, \tag{6}$$

wherein $C_k$ specifies the probability amplitude of the corresponding states $S_k$ and subjects to the normalization condition of (7).

$$|C_1|^2 + |C_2|^2 + ... + |C_{2^m}|^2 = 1 \tag{7}$$

The state of a qubit can be changed by the operation of a quantum gate or Q-gate. The Q-gates apply a unitary operation $U$ on a qubit in the state $|\psi\rangle$ making it evolve to the state $U\,|\psi\rangle$, which maintains the probabilities interpretation defined in the (3). There are several Q-gates, such as the $NOT$ gate, *Controlled-*$NOT$ gate, *Hadamard* gate, *rotation* gate, etc.

## 4 Quantum Inspired Evolutionary Algorithm

As any evolutionary algorithms, this algorithm is based on a population of solutions which is maintained through many generations. It seeks the best fitted solution to the problem, evaluating the characteristics of those included in the current population. In the next sections, we describe the quantum inspired representation of the individual and the under laying computational process.

### 4.1 Quantum inspired representation

The evolutionary algorithms, like the genetic algorithms, for instance, can use several representation that have been used with success: binary, numeric and symbolic representation [Hinterding 1999]. The quantum inspired evolutionary algorithms use a new representation, that is a probabilistic representation based on the concept of qubits and a q-individual as a string of qubits. A q-individual can be defined as in (8) wherein $|\alpha_i|^2 + |\beta_i|^2 = 1$, for $i = 1, 2, 3, ..., m$.

$$p = \left[\begin{array}{c|c|c|c|c} \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_m \\ \beta_1 & \beta_2 & \beta_3 & \cdots & \beta_m \end{array}\right] \tag{8}$$

The advantage of the representation of the individuals using qubits instead of the classical representation is the capacity of representing the linear superposition of all possible states. For instance, an individual represented with three qubits ($m = 3$) can be represented as in (9):

$$p = \left[ \begin{array}{c|c|c} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \sqrt{\frac{2}{3}} & \frac{\sqrt{3}}{2} \end{array} \right], \tag{9}$$

or represented in the alternative way of (10),

$$\begin{aligned} p = &\frac{1}{2\sqrt{6}}\left|000\right\rangle + \frac{1}{2\sqrt{2}}\left|001\right\rangle + \frac{1}{2\sqrt{3}}\left|010\right\rangle + \frac{1}{2}\left|011\right\rangle \\ &\frac{1}{2\sqrt{6}}\left|100\right\rangle + \frac{1}{2\sqrt{2}}\left|101\right\rangle + \frac{1}{2\sqrt{3}}\left|110\right\rangle + \frac{1}{2}\left|111\right\rangle. \end{aligned} \tag{10}$$

The numbers in the (10) represent the amplitudes whose square-roots indicate the probabilities of representing the states $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$ and $|111\rangle$, which are $\frac{1}{24}$, $\frac{1}{8}$, $\frac{1}{24}$,$\frac{1}{12}$,$\frac{1}{24}$,$\frac{1}{8}$,$\frac{1}{24}$ and $\frac{1}{12}$, respectively.

The evolutionary algorithms with the quantum inspired representation of the individual should present a population diversity better than other representations, since they can represent the linear superposition of states [Akbarzadeh-T and Khorsand 2005] [Han and Kim 2002]. Only one q-individual, as the one indicated in (9) for instance, is enough to represent eight states. Using the classical representation, eight individuals would be necessary.

### 4.2 Algorithm description

The basic structure of the quantum inspired evolutionary algorithm presented in this paper is described by Algorithm 1.

The quantum inspired evolutionary algorithms maintain a population of q-individuals, $P(g) = \{\mathtt{p}_1^g, \mathtt{p}_2^g, ..., \mathtt{p}_n^g\}$ at generation $g$, where $n$ is the size of population, and $\mathtt{p}_j^g$ is a q-individual defined as in (11)

$$\mathtt{p}_j^g = \left[ \begin{array}{c|c|c|c|c} \alpha_{j_1}^g & \alpha_{j_2}^g & \alpha_{j_3}^g & \cdots & \alpha_{j_m}^g \\ & & & & \\ \beta_{j_1}^g & \beta_{j_2}^g & \beta_{j_3}^g & \cdots & \beta_{j_m}^g \end{array} \right], \tag{11}$$

where $m$ is the number of qubits, which defines the string length of the q-individual, and $j = 1, 2, ..., n$.

The initial population of $n$ individuals is generated setting $\alpha_i^0 = \beta_i^0 = 1/\sqrt{2}$ ($i = 1, 2, ..., m$) of all $\mathtt{p}_j^0 = \mathtt{p}_j^g\,|_{g=0}$ ($j = 1, 2, ..., n$). This allows each q-individual to be the superposition of all possible states with the same probability.

The binary solutions in $S_g$ are obtained by an observation process of the states of every q-individual in $P_g$. Let $S_g = \{\mathtt{s}_1^g, \mathtt{s}_2^g, ..., \mathtt{s}_n^g\}$ at generation $g$. Each

**Algorithm 1** Quantum Inspired Evolutionary Algorithm

$g := 0$;
**generate** initial population $P_0$ with $n$ individuals;
**observe** $P_0$ into $S_0$;
**evaluate** the fitness of every solution in $S_0$;
**store** $S_0$ into $B_0$;
**while** (**not** *termination condition*) **do**
  $g := g + 1$;
  **observe** $P_{g-1}$ into $S_g$;
  **evaluate** the fitness of every solution in $S_g$;
  **update** $P_g$ using a Q-gate;
  **apply** probability constraints;
  **store** best solutions among $B_{g-1}$ and $S_g$ into $B_g$;
  **store** the best solution in $B_g$ into $b$;
  **if** (*no improvement for many generation*) **then**
    **replace** all the solution of $B_g$ by $b$;
  **end if**
**end while**

solution, $\mathbf{s}_i^g$ for $(i = 1, 2, ..., n)$, is a binary string with the length $m$, that is, $\mathbf{s}_i^g = s_1 s_2 ... s_m$, where $s_j$ for $(j = 1, 2, ..., m)$ is either 0 or 1.

The observation process is implemented using random probability: for each pair of amplitudes $[\alpha_k, \beta_k]^T$ $(k = 1, 2, ..., n \times m)$ of every qubit in the population $P_g$, a random number $r$ in the range $[0, 1]$ is generated. If $r < |\beta_k|^2$, the observed qubit is 1; otherwise, it is 0.

The q-individuals in $P_g$ are updated using a Q-gate, which is detailed in later. We impose some probability constraints such that the variation operation performed by the Q-gate avoid the premature convergence of a qubits to either to 0 or 1. This is done by not allowing neither of $|\alpha|^2$ nor $|\beta|^2$ to reach 0 or 1. For this purpose, the probability $|\alpha|^2$ and $|\beta|^2$ are constrained to 0.02 as a minimum and 0.98 as a maximum. Such constraints allowed the algorithm to escape local minima.

After a given number of generation, if the best solution $b$ does not improved, all the solutions stored into $B_g$ are replaced by $b$. This step can induce a variation of the probabilities of the q-individuals. This operation is also performed in order to escape local minima and avoid the stagnant state.

## 5 Application to the State Assignment Problem

The identification of a good state assignment has been thoroughly studied over the years. In particular, Armstrong [Armstrong 1962] and Humphrey [Humphrey

| State | $S_0$ | | $S_1$ | | $S_2$ | | $S_3$ | |
|---|---|---|---|---|---|---|---|---|
| q-individual | $\alpha_1^0$<br>$\beta_1^0$ | $\alpha_2^0$<br>$\beta_2^0$ | $\alpha_1^1$<br>$\beta_1^1$ | $\alpha_2^1$<br>$\beta_2^1$ | $\alpha_1^2$<br>$\beta_1^2$ | $\alpha_2^2$<br>$\beta_2^2$ | $\alpha_1^3$<br>$\beta_1^3$ | $\alpha_2^3$<br>$\beta_2^3$ |
| Possible observation | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |

**Figure 4:** Example of state assignment encoding

1958] have pointed out that an assignment is good if it respects two rules, which consist of the following:

- two or more states that have the same next state should be given adjacent assignments;
- two or more states that are the next states of the same state should be given adjacent assignments. State adjacency means that the states appear next to each other in the mapped representation. In other terms, the combination assigned to the states should differ in only one position;
- the first rule should have precedence over the second.

Now we concentrate on the assignment encoding and the fitness function. Given two different state assignments, the fitness function allows us to decide which is fitter.

### 5.1 State Assignment Encoding

In this case, a q-individual represents a state assignment. Each q-individual consists of an array of $2 \times N \lceil(\log_2 N)\rceil$ entries, wherein each set of $2 \times \lceil \log_2 N \rceil$ entries is the qubit assigned to a single machine state. For instance, Fig. 4 represents a q-individual and a possible assignment for a machine with 4 states obtained after the observation of the qubits states.

Note that when an observation occurs, one code might be used to represent two or more distinct states. Such a state assignment is not possible. In order to discourage the selection of such assignment, we apply a penalty every time a code is used more than once within the considered assignment. This will be further discussed in the next section.

### 5.2 State Assignment Fitness Evaluation

This step of the quantum inspired evolutionary algorithm evaluates the fitness of each binary solutions obtained from the observation of the states of the q-individuals. The fitness evaluation of state assignments is performed with respect to two rules of Armstrong [Armstrong 1962] and Humphrey [Humphrey 1958]:

| Present | Next State | | Output ($O$) | |
|---|---|---|---|---|
| State | $I = 0$ | $I = 1$ | $I = 0$ | $I = 1$ |
| $s_0$ | $s_0$ | $s_1$ | 0 | 0 |
| $s_1$ | $s_2$ | $s_1$ | 0 | 1 |
| $s_2$ | $s_0$ | $s_3$ | 1 | 0 |
| $s_3$ | $s_2$ | $s_1$ | 1 | 1 |

**Table 2:** Exemplo of a state transition function

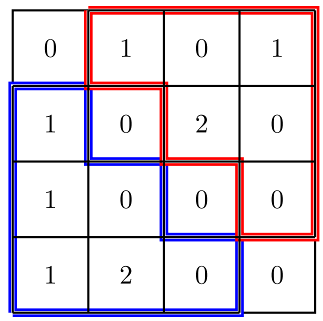



**Figure 5:** The adjacency matrix for the state transition function of 2

- how much a given state assignment adheres to the first rule, i.e. how many states in the assignment, which have the same next state, have no adjacent state codes;
- how much a given state in the assignment adheres to the second rule, i.e. how many states in the assignment, which are the next states of the same state, have no adjacent state codes.

In order to efficiently compute the fitness of a given state assignment, we use an $N \times N$ *adjacency matrix*, wherein $N$ is the number of the machine states. The triangular bottom part of the matrix holds the expected adjacency of the states with respect to the first rule while the triangular top part of it holds the expected adjacency of the states with respect to the second rule. The matrix entries are calculated as described in (12), wherein $AM$ stands for the adjacency matrix, functions $next(\sigma)$ and $prev(\sigma)$ yield the set of states that are next and previous to state $\sigma$ respectively. For instance, a state machine and its respective $4 \times 4$ adjacency matrix are shown in Table 2 and Fig. 5 respectively.

$$AM_{i,j} = \begin{cases} \#\left(next(q_i) \cap (next(q_j)\right) & \text{if } i > j \\ \#\left(prev(q_i) \cap (prev(q_j)\right) & \text{if } i < j \\ 0 & \text{if } i = j \end{cases} \quad (12)$$

Using the adjacency matrix $AM$, the fitness function applies a penalty of 2 or 1, every time the first or second rule are broken, respectively. In (13) shows the details of the fitness function applied to a state assignment $\sigma$, wherein function $na(q, p)$ returns 0 if the codes representing states $q$ and $p$ are adjacent and 1 otherwise. Note that state assignments that encode two distinct states using the same codes are penalized by adding the constant $\psi$ to the fitness function.

$$f(\sigma) = \sum_{i \neq j \ \& \ \sigma_i = \sigma_j} \psi + \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} \left(AM_{i,j} + 2 \times AM_{j,i}\right) \times na(\sigma_i, \sigma_j) \quad (13)$$

For instance, considering the state machine described in Fig. 5, the state assignment $\{s_0 \equiv 00,\ s_1 \equiv 10,\ s_2 \equiv 01,\ s_3 \equiv 11\}$ has a fitness of 5 as the codes of states $s_0$ and $s_3$ are not adjacent but $AM_{0,3} = 1$ and $AM_{3,0} = 1$ and the codes of states $s_1$ and $s_2$ are not adjacent but $AM_{1,2} = 2$ while the assignments $\{s_0 \equiv 00,\ s_1 \equiv 11,\ s_2 \equiv 01,\ s_3 \equiv 10\}$ has a fitness of 3 as the codes of states $s_0$ and $s_1$ are not adjacent but $AM_{0,1} = 1$ and $AM_{1,0} = 1$.

The objective of the quantum inspired evolutionary algorithm is to find the assignment that minimize the fitness function as described in (12). Assignments with fitness 0 satisfy all the adjacency constraints. Note that such an assignment does not always exist.

### 5.3 Q-gate for State Assignment

To drive the individuals toward better solutions, a Q-gate is used as a variation operator of the quantum inspired evolutionary algorithm presented at this paper. After an update operation, the qubit must always satisfy the normalization condition $|\alpha'|^2 + |\beta'|^2 = 1$, where $\alpha'$ and $\beta'$ are the amplitudes of the updated qubit.

Initially, each q-individual represents all possible states with the same probability. As the probability of every qubit approaches either 1 or 0 by the Q-qate, the q-individual converges to a single state and the diversity property disappears gradually. By this mechanism, the quantum inspired evolutionary algorithm can treat the balance between exploration and exploitation [Han and Kim 2002]. The

Q-gate used is inspired by a quantum rotation gate. This is defined in (14).

$$\begin{bmatrix} \alpha' \\ \beta' \end{bmatrix} = \begin{bmatrix} cos(\Delta\theta) & -sin(\Delta\theta) \\ sin(\Delta\theta) & cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \tag{14}$$

where $\Delta\theta$ is the rotation angle of each qubit toward states 0 or 1 depending on the amplitude signs.

The value of the angle $\Delta\theta$ can be selected from the Table 3, where $f(\mathtt{s}_i^g)$ and $f(\mathtt{b}_i^g)$ are the fitness values of $\mathtt{s}_i^g$ and $\mathtt{b}_i^g$, and $s_j$ and $b_j$ are the $j$th bits of the observed solutions $\mathtt{s}_i^g$ and the best solutions $\mathtt{b}_i^g$, respectively.

The rotation gate allows changing the amplitudes of the considered qubit, as follows:

1. If $s_j$ and $b_j$ are 0 and 1, respectively, and if $f(\mathtt{s}_i^g) \geq f(\mathtt{b}_i^g)$ is false then:
   - if the qubit is located in the first or third quadrant as defined in (5), $\theta_3$, the value of $\Delta\theta$ is set to a positive value to increase the probability of the state $|1\rangle$;
   - if the qubit is located in the second or fourth quadrant, $-\theta_3$ should be used to increase the probability of the state $|1\rangle$.
2. If $s_j$ and $b_j$ are 1 and 0, respectively, and if $f(\mathtt{s}_i^g) \geq f(\mathtt{b}_i^g)$ is false:
   - if the qubit is located in the first or third quadrant, $\theta_5$ is set to a negative value to increase the probability of the state $|0\rangle$;
   - if the qubit is located in the second or fourth quadrant, $-\theta_5$ should be used to increase the probability of the state $|0\rangle$.

When it is ambiguous to select a positive or negative number for the angle parameter, we set its value to zero as recommended in [Han and Kim 2002]. The magnitude of $\Delta\theta$ has an effect on the speed of convergence. If it is too big, the search grid of the algorithm would be large and the solutions may diverge or converge prematurely to a local optimum. If it is too small, the search grid of the algorithm would be small and the algorithm may fall in stagnant state. Hence, the magnitude of $\Delta\theta$ is defined as a variable, which values depend on the aplication problem. In the state assignment problem, we used $\theta_1 = \theta_2 = \theta_4 = \theta_6 = \theta_7 = \theta_8 = 0$, $\theta_3 = 0.05\pi$, and $\theta_5 = -0.05\pi$.

## 6 Performance Results

In this section, we compare the assignment evolved by the quantum inspired evolutionary algorithm presented in this paper to those yield by the genetic

**Table 3:** Look-up table of $\Delta\theta$

| $s_j$ | $b_j$ | $f(\mathtt{s}_i^g) \geq f(\mathtt{b}_i^g)$ | $\Delta\theta$ |
|---|---|---|---|
| 0 | 0 | false | $\theta_1$ |
| 0 | 0 | true | $\theta_2$ |
| 0 | 1 | false | $\theta_3$ |
| 0 | 1 | true | $\theta_4$ |
| 1 | 0 | false | $\theta_5$ |
| 1 | 0 | true | $\theta_6$ |
| 1 | 1 | false | $\theta_7$ |
| 1 | 1 | true | $\theta_8$ |

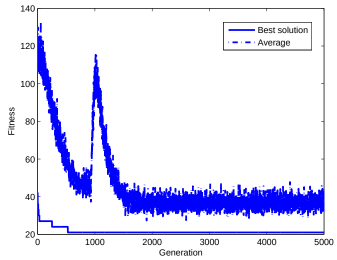
**Table 4:** Fitness of best assignments yield by the compared systems

| State machine | *#AdjRes* | QUANTUM | $GA_1$ | $GA_2$ | NOVA1 | NOVA2 |
|---|---|---|---|---|---|---|
| **Shiftreg** | 24 | 0 | 0 | 0 | 8 | 0 |
| **Lion9** | 69 | 21 | 21 | 27 | 25 | 30 |
| **Train11** | 57 | 17 | 18 | 19 | 23 | 28 |
| **Bbara** | 225 | 125 | 127 | 130 | 135 | 149 |
| **Dk14** | 139 | 68 | 68 | 75 | 72 | 76 |
| **Bbsse** | 328 | 216 | 223 | 225 | 230 | 239 |
| **Donfile** | 432 | 246 | 247 | 276 | 343 | 310 |

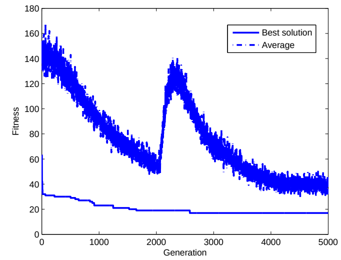
algorithms [Amaral et al. 1995] and [Nedjah and Mourelle 2004] and to those obtained using the non-evolutionary assignment system called NOVA [Villa and Sangiovanni-Vincentelli 1990]. The examples are well-known benchmarks for testing synchronous finite state machines [University 2008].

Table 4 gives the fitness of the best state assignment produced by the quantum inspired evolutionary algorithm, the genetic algorithms from [Amaral et al. 1995] ($GA_2$) and [Nedjah and Mourelle 2004] ($GA_1$) and the two versions of NOVA system [Villa and Sangiovanni-Vincentelli 1990]. The *#AdjRes* stands for the number of expected adjacency restrictions. Each adjacency according to rule 1 is counted twice and that with respect to rule 2 is counted just once. For instance, in the case of the $Shiftreg$ state machine, all 24 expected restrictions were fulfilled in the state assignment yielded by the compared systems. However, the state assignment obtained the first version of the NOVA system does not fulfil 8 of the expected adjacency restrictions of the state machine.

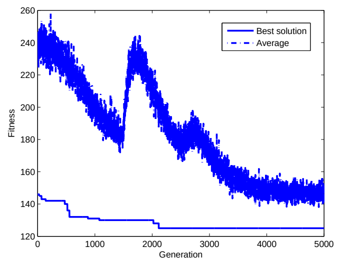
Table 5 shows the best state assignment generated by the compared systems. The size column shows the total number of states/transitions of the machine.



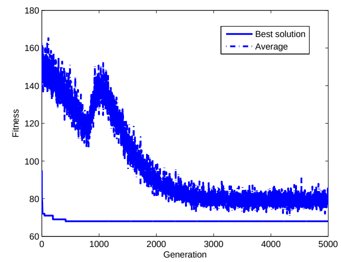

(a) Lion9



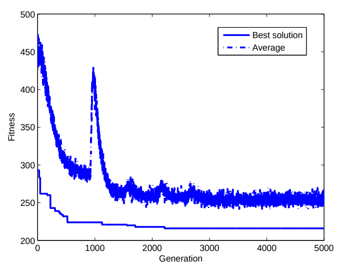

(b) Train11



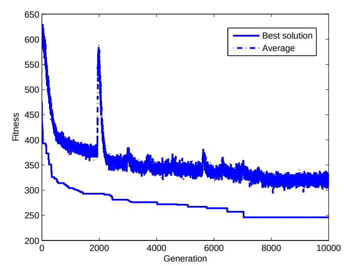

(c) Bbara




(d) Dk14




(e) Bbsse




(f) Donfile

Figure 6: Fitness of best and average assignments with a population size = 50

Table 6 shows the best state assignment generated by benchmark *donfile*. The total number of states, in this case, is 24 and that of transitions is 96.

For most of the benchmarks used, we show, in Fig. 6, the best fitness and the average fitness over the generations. The chart of Fig. 7 compares graphically the degree of fulfillment of the adjacency restrictions expected in the state machines used as benchmarks. The chart shows clearly that our quantum inspired evolutionary algorithm always evolves a better or equal state assignment.

Table 5: Best state assignment yield by the compared systems for the benchmarks

| FSM | System | State Assignment |
|---|---|---|
| **Shiftreg** | $GA_2$ | [0,2,5,7,4,6,1,3] |
| 8/16 | NOVA1 | [0,4,2,6,3,7,1,5] |
| | NOVA2 | [0,2,4,6,1,3,5,7] |
| | $GA_1$ | [5,7,4,6,1,3,0,2] |
| | QUANT | [4,0,2,6,5,1,3,7] |
| **Lion9** | $GA_2$ | [0,4,12,13,15,1,3,7,5] |
| 9/25 | NOVA1 | [2,0,4,6,7,5,3,1,11] |
| | NOVA2 | [0,4,12,14,6,11,15,13,7] |
| | $GA_1$ | [10,8,12,9,13,15,7,3,11] |
| | QUANT | [10,2,8,0,1,5,13,12,4] |
| **Train11** | $GA_2$ | [0,8,2,9,13,12,4,7,5,3,1] |
| 11/25 | NOVA1 | [0,8,2,9,1,10,4,6,5,3,7] |
| | NOVA2 | [0,13,11,5,4,7,6,10,14,15,12] |
| | $GA_1$ | [2,6,1,4,0,14,10,9,8,11,3] |
| | QUANT | [6,7,4,11,10,15,14,0,2,12,8] |
| **Bbarra** | $GA_2$ | [0,6,2,14,4,5,13,7,3,1] |
| 10/60 | NOVA1 | [4,0,2,3,1,13,12,7,6,5] |
| | NOVA2 | [9,0,2,13,3,8,15,5,4,1] |
| | $GA_1$ | [3,0,8,12,1,9,13,11,10,2] |
| | QUANT | [2,4,5,7,6,14,12,3,1,0] |
| **Dk14** | $GA_2$ | [0,4,2,1,5,7,3] |
| 7/56 | NOVA1 | [5,7,1,4,3,2,0] |
| | NOVA2 | [7,2,6,3,0,5,4] |
| | $GA_1$ | [3,7,1,0,5,6,2] |
| | QUANT | [0,4,1,3,6,7,5] |
| **Bbsse** | $GA_2$ | [0,4,10,5,12,13,11,14,15,8,9,2,6,7,3,1] |
| 16/56 | NOVA1 | [12,0,6,1,7,3,5,4,11,10,2,13,9,8,15,14] |
| | NOVA2 | [2,3,6,15,1,13,7,8,12,4,9,0,5,10,11,14] |
| | $GA_1$ | [15,14,9,12,1,4,3,7,6,10,2,11,13,0,5,8] |
| | QUANT | [9,1,0,5,2,4,6,14,10,15,7,3,11,13,8,12] |

## 7 Conclusions

In this paper we exploited a quantum evolutionary algorithm to solve the $NP$-complete problem of state encoding in the design process of finite state machines. We compared the state assignment evolved by our algorithm for machines of different sizes. Our algorithm always obtains better or equal assignments.

Table 6: Best state assignment yield by the compared systems for the benchmark *donfile*

| System | State Assignment |
|---|---|
| $GA_1$ | [0,12,9,1,6,7,2,14,11,17,20,23,8,15,10,16,21,19,4,5,22,18,13,3] |
| NOVA1 | [12,14,13,5,23,7,15,31,10,8,29,25,28,6,3,2,4,0,30,21,9,17,12,1] |
| NOVA2 | [6,30,11,28,25,19,0,26,1,2,14,10,31,24,27,15,12,8,29,23,13,9,7,3] |
| $GA_2$ | [2,18,17,1,29,21,6,22,7,0,4,20,19,3,23,16,9,8,13,5,12,28,25,24] |
| QUANT | [7,6, 23,31,26,27,15,14,13,5,10,4,22,30,12,8,11,9,18,19,2,0,3,1] |

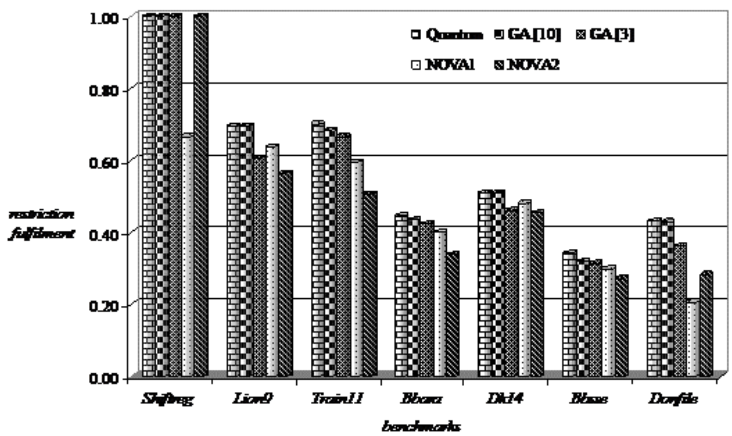



Figure 7: Graphical comparison of the degree of fulfillment of rule 1 and 2 reached by the systems

## Acknowledgment

We are grateful to the reviewers and the editor that contributed to the great improvement of the original version of this paper with their valuable comments and suggestions. We also are thankful to FAPERJ (Fundação de Amparo à Pesquisa do Estado do Rio de janeiro, `http:// www.faperj.br`) and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico, `http://www.cnpq.br`) for their continuous financial support.

## References

[Akbarzadeh-T and Khorsand 2005] Akbarzadeh-T, M.-R., Khorsand, A.-R. (2005). “Quantum Gate Optimization in a Meta-Level Genetic Quantum Algorithm". Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. volume 4. 3055–3062. IEEE Press, Piscataway, NJ, USA.

[Amaral et al. 1995] Amaral, J. N., Tumer, K., Glosh, J. (1995). "Designing Genetic Algorithms for the State Assignment Problem". IEEE Transactions on Systems, Man, and Cybernetics. 25, 4, 686–694.

[Armstrong 1962] Armstrong, D. B. (1962). "A Programmed Algorithm for Assigning Internal Codes to Sequential Machines". IRE Transactions on Electronic Computers. EC-11, 4, 466–472.

[Han and Kim 2002] Han, K.-H., Kim, J.-H. (2002). "Quantum-Inspired Evolutionary Algorithm for a Class of Combinatorial Optimization". IEEE Transactions on Evolutionary Computation. 6, 6, 580–593.

[Hey 1999] Hey, T. (1999). "Quantum computing". Computing Control Engineering Journal. 10, 3, 105–112.

[Hinterding 1999] Hinterding, R. (1999). "Representation, constraint satisfaction and the knapsack problem". Proceedings of the Congress on Evolutionary Computation. volume 2. 1286–1292. IEEE Press, Piscataway, NJ, USA.

[Humphrey 1958] Humphrey, W. S. (1958). Switching Circuits with Computer Applications. McGraw-Hill, New York, USA.

[Narayanan 1999] Narayanan, A. (1999). "Quantum computing for beginners". Proceedings of the Congress on Evolutionary Computation. volume 3. 2231–2238. IEEE Press, Piscataway, NJ, USA.

[Nedjah and Mourelle 2004] Nedjah, N., Mourelle, L. M. (2004). "Evolutionary synthesis of synchronous finite state machines". N. Nedjah, L. M. Mourelle, eds., Evolvable Machines. 103–128. Springer-Verlag, Berlin, Germany. 1 edition.

[Rhyne 1973] Rhyne, V. T. (1973). Fundamentals of digital systems design. Computer Applications in Electrical Engineering Series. Prentice-Hall.

[University 2008] University, N. C. S. (2008). "Collaborative benchmarking and experimental algorithmics lab".

[Villa and Sangiovanni-Vincentelli 1990] Villa, T., Sangiovanni-Vincentelli, A. (1990). "NOVA". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 9, 9, 905–924.

[Zhang et al. 2006] Zhang, G., et al. (2006). "Novel Quantum Genetic Algorithm and its Applications". Frontiers of Electrical and Electronic Engineering in China. 1, 1, 31–36.