MISSOURI S&T

Missouri University of Science and Technology

Scholars' Mine

Electrical and Computer Engineering Faculty Research & Creative Works

Electrical and Computer Engineering

01 Jan 2005

Evolving Combinational Logic Circuits Using a Hybrid Quantum Evolution and Particle Swarm Inspired Algorithm

Phillip W. Moore

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

Follow this and additional works at: https://scholarsmine.mst.edu/ele_comeng_facwork

🔴 Part of the Electrical and Computer Engineering Commons

Recommended Citation

P. W. Moore and G. K. Venayagamoorthy, "Evolving Combinational Logic Circuits Using a Hybrid Quantum Evolution and Particle Swarm Inspired Algorithm," *Proceedings of the 2005 NASA/DoD Conference on Evolvable Hardware (EH'05)*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2005.
The definitive version is available at https://doi.org/10.1109/EH.2005.28

Evolving Combinational Logic Circuits Using a Hybrid Quantum Evolution and Particle Swarm Inspired Algorithm

Phillip Moore, Ganesh K Venayagamoorthy, *Senior Member, IEEE*
Real-Time Power and Intelligent Systems Laboratory
University of Missouri-Rolla, MO 65409-0249, U.S.A.
gkumar@ieee.org

Abstract

In this paper, an algorithm inspired from quantum evolution and particle swarm to evolve combinational logic circuits is presented. This algorithm uses the framework of the local version of particle swarm optimization with quantum evolutionary algorithms, and integer encoding. A multi-objective fitness function is used to evolve the combinational logic circuits in order obtain feasible circuits with minimal number of gates in the design. A comparative study indicates the superior performance of the hybrid quantum evolution-particle swarm inspired algorithm over the particle swarm and other evolutionary algorithms (such as genetic algorithms) independently.

1. Introduction

There are many methods to design combinational logic circuits. Generally used methods are Karnaugh maps [1, 2, 3]. The problem with the human designs is that they become cumbersome and problematic when the number of inputs, number of outputs, and complexity of the function increases. The intricacy of the combinational circuit depends on the number of gates in the circuit. For real world applications, combinational circuit designs require for hardware realization circuits consuming less power and area, and fast.

The evolutionary design of electronic circuits refers to an autonomous process in which a highly efficient circuit may occur in a population of interacting instances of a logic function. Evolutionary hardware design has potential for technological advancement in the near future. Many papers have reported on the design of combinational circuits using genetic algorithms [4, 5] and particle swarm optimization [6, 7].

The particle swarm optimization (PSO) utilizes a population of particles on a hyper-dimensional search space to find feasible optimal solutions [8]. The authors investigated the use of particle swarm and differential evolution (DE) [9] independently, and as a hybrid algorithm (DEPSO) for evolving combinational circuits [10]. It was reported that the hybrid algorithm found the feasible solutions more frequently and faster than the PSO and DE independently.

Quantum-Inspired Evolutionary Algorithm (QEA) is a novel evolutionary algorithm proposed in [11]. It utilizes the concepts of a quantum bit, superposition of states and collapse of states. Like other evolutionary algorithms, QEA is also characterized by the representation of the individual, the evaluation function and the population dynamics. However, instead of binary, numeric or symbolic representation, QEA uses a Q-bit as a probabilistic representation, defined as the smallest unit of information. A Q-bit individual is defined by a string of Q-bits. The Q-bit individual has the advantage that it can represent a linear superposition of states (binary solutions) in search space probabilistically. Thus, the Q-bit representation has a better characteristic of population diversity than any other representation.

This paper presents a new algorithm inspired by hybrid quantum evolution (QE) and particle swarm (PSO) for the design of combinational logic circuits. The PSO framework is used with QEA in this new algorithm called the Quantum Evolution Particle Swarm Optimization (QEPSO) Algorithm.

The remaining sections of this paper are organized as follows: Section 2 gives a brief overview of the combination logic circuit representation structure. Section 3 describes the new algorithm QEPSO. Section 4 presents three case studies and the circuits evolved by the hybrid QEPSO algorithm. Finally, the conclusion and future work is given in section 5.

2. Evolving Combinational Logic Circuits

The basic process of hardware evolution is illustrated in Fig. 1. The "desired" circuit refers to the circuit that maps 100 % exactly the outputs for corresponding input combinations typically given by a truth table for digital circuits. The hardware evolution is carried out until the "desired" circuit is evolved and

then downloaded to a reconfigurable hardware platform. This sequential process is commonly referred to as extrinsic evolution.
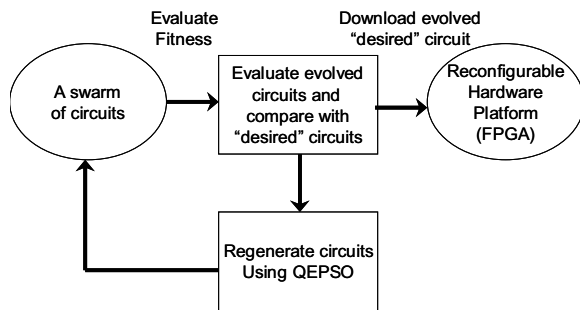

Figure 1. "Desired" circuit hardware evolution

Evolving digital circuits using an evolutionary approach uses individuals (particles or chromosomes) to represent each circuit. Each entity of the population represents possible potential solutions. The matrix shown in Fig. 2 is used to represent a circuit with *m* rows and *n* columns. The elements of the circuit are the logic gates, which are selected from a predefined library of 2-input 1-output gates. The library of the gates used in this study consists of NOT, WIRE, AND, OR, and XOR gates.
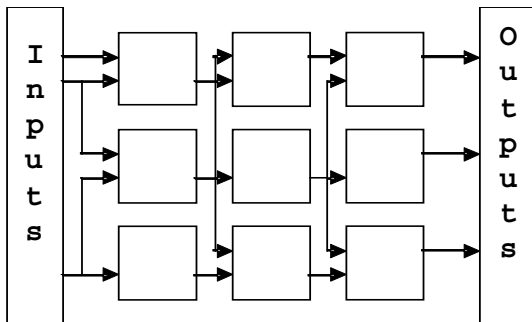

Figure 2. Circuit representation (m×n)

The inputs to the first column of the matrix come from the truth table of the function to be implemented, generally specified in terms of variables like *x* and *y*. For all other columns, the inputs come from the previous column outputs. The order of traversing the elements in the matrix is column wise, starting from the first column, going down through all the rows and then to the next column. For example, if there are 5 rows and 5 columns, then the gate matrix is of size 5 by 5.

3. Hybrid QEA & PSO Inspired Algorithm

In this paper, the quantum evolution and particle swarm inspired algorithm is used to evolve combinational circuits. The combinational logic circuit evolution is a multi-objective optimization. The first objective is finding a feasible circuit i.e. all outputs of the truth table must be matched. The second objective is to minimize the number of gates in the population of feasible circuits obtained (satisfying the first objective). The fitness evaluation of the particles in the hybrid algorithm is given in (1). Fitness1 function in (2) evaluates whether a particle has obtained a feasible solution or not. A zero value for Fitness1 means it is a feasible solution. Fitness2 function given in (3) is used to evaluate feasible solution in terms of number of gates. If a solution is not feasible, Fitness2 assigns a value equivalent to the size of the circuit structure (m×n), shown in Fig. 2.

$$Fitness = Fitness1 + Fitness2 \qquad (1)$$

$$Fitness1 = 1 - \frac{number\_correct\_outputs}{total\_number\_of\_outputs} \quad (2)$$

$$Fitness2 = \begin{cases} m \times n, & not\_feasible\_circuit \\ no\_gates, & feasible\_circuit \end{cases} \quad (3)$$

3.1 PSO Influence

The QEPSO algorithm is driven by the QEA and implemented with ideas taken from PSO. The neighborhood version of PSO is applied on the QEA. This is also referred to as the local version of PSO. The particle with the best fitness in the neighborhood is referred to as the local best or the L_{best} and each particle's best fitness found is stored in memory and is referred to as the P_{best}. The 3-integer approach to PSO is implemented with QEA to realize the QEPSO algorithm illustrated in the flowchart given in Fig. 3. When analyzing the 3-integer approach to PSO, the velocity of the PSO is used to update the position of a particle, as shown in (4), to 3 possible positions (the original position, the L_{best} position, or the P_{best} position). One of the 3 cases is chosen for the position update based on a random number. When this random number is less than or equal to a normalized velocity, the position of a current particle is updated with the L_{best} position. The second case emerges when a random number is less than or equal to one minus the normalized velocity and when the first case does not occur. The second case will update the current particle position with the P_{best} position. If both of these cases do not occur, then the third case will make sure that the current particle will keep its current position. This is how the normalized velocity updates the particles' positions for each dimension.

$$pos_{new} = \begin{cases} L_{best}, & vel_n < FF_1 \\ P_{best}, & (1-vel_n) < FF_2 \\ Pos_{old}, & vel_n > FF_1, or, vel_n > FF_2 \end{cases} \quad (4)$$

Where vel_n is the normalized velocity and FF is a flip-function.

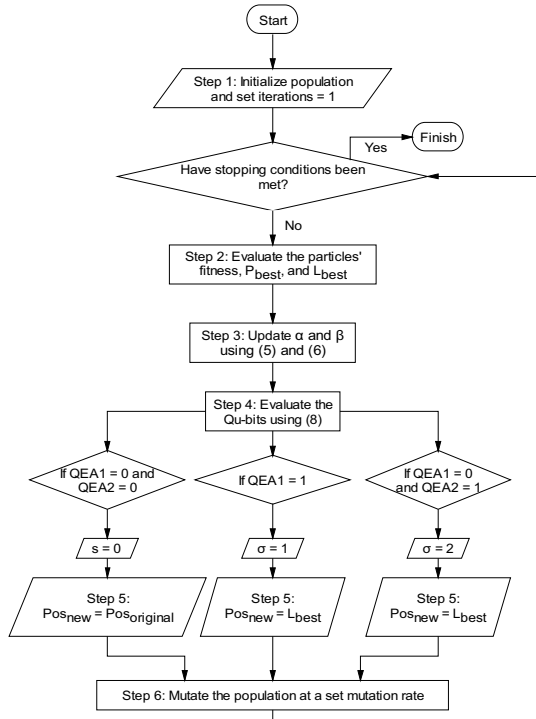Figure 3. QEPSO flowchart

3.2 Quantum Evolutionary Algorithm

The parameters for the QEA are θ and the population size. Different values of θ control the convergence speed of the algorithm. The population size is the number of circuits that are being evolved. The QEA updates the binary position numbers, $d\theta$, α, and β. There are 2 binary Q-bit numbers, binary position numbers, that represents possible solutions that the QEA is evolving to produce minimal gate circuit solutions. The $d\theta$ changes the direction of the rotational angle in order to direct the Q-bits to either a 0 or a 1. Depending on where the values of α and β lie in the Cartesian plain, the $d\theta$ will move their values closer to the α and β values of the best particle in the population. The $d\theta$ has the magnitude of θ and is also used to update the values of α and β given by (5) and (6) respectively. The α and β values are probabilities of

the each Q-bit state being either a 0 or a 1, respectively. The sum of α^2 and β^2 will always equal one (7). The QEA produces a 0 or a 1 to represent one of the possible positional changes by comparing a random number, 0 through 1, to the value of β^2 (8).

$$\alpha_{new} = [cos(d\theta) * \alpha] - [sin(d\theta) * \beta] \quad (5)$$

$$\beta_{new} = [sin(d\theta) * \alpha] - [cos(d\theta) * \beta] \quad (6)$$

$$\alpha^2 + \beta^2 = 1 \quad ` \quad (7)$$

$$Qu-bit = \begin{cases} 1 & rand[0,1] < \beta^2 \\ 0 & rand[0,1] \geq \beta^2 \end{cases} \quad (8)$$

3.3 QEPSO

Comparing a normalized velocity to a random number decreases the effectiveness of the velocity update equation for PSO. Instead of using the normalized velocity to produce 3 different cases to update a particle's position, the QEA algorithm has been chosen to be its replacement. The problem with using one QEA algorithm is that the results produced, either a 0 or a 1 for each particle's dimensions, only handles 2 separate cases effectively. In order to handle a third case for the QEA algorithm to evolve, a second QEA algorithm is introduced. The first case is taken care of when the first QEA produces a 1. This will update the current position of a particle to the L_{best} position. If a 0 is produced in the first QEA, a second QEA is called to handle the next two cases. When the second QEA produces a 0, the second case is implemented. The second case updates the current position of a particle to the P_{best} position. Finally, if the second QEA produces a 1 the last case is taken care of by keeping the current position of the particle. This is how the 2 QE algorithms relate to updating the positions of the particles for each dimension (9).

$$Pos_{new} = \begin{cases} L_{best}, & QEA1 == 1 \\ P_{best}, & (QEA1 == 0) \& (QEA2 == 1) \\ Pos_{old}, & (QEA1 == 0) \& (QEA2 == 0) \end{cases} \quad (9)$$

4. Case Studies and Results

Three case studies are presented to show the capability of the proposed hybrid algorithm for circuit evolution. A total of 50 particles are used with a

neighborhood size of 3. The maximum of iterations allowed is 4,000.

4.1 Case Study 1

The QEPSO algorithm evolves the circuit for the truth table given in Table 1. Table 2 shows the results obtained by the human designer, PSO and the QEPSO. The circuit representation structure (Fig. 2) is a 4×4. All the results presented in this paper are averaged over 20 trials. The QEPSO like the PSO finds the feasible solutions all the time on this smaller three input based circuit. The circuits obtained by the PSO and the QEPSO are shown in Fig. 4.
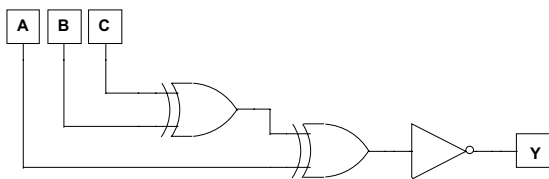

Figure 4. Evolved circuit by PSO and QEPSO for case study 1

Table 1. Truth table for case study 1

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Table 2. Result comparisons of the human designer, PSO, and QEPSO for case study 1

Case Study 1			
	Human Designer	PSO	QEPSO
Min Gates	3	3	3
Equation	$\overline{B \otimes (A \otimes C)}$	$\overline{A \otimes (B \otimes C)}$	$\overline{A \otimes (B \otimes C)}$
Types of Gates	2 XOR, 1 NOT	2 XOR, 1 NOT	2 XOR, 1 NOT
Percent of Best Solution	-	100%	100%
Feasible Circuits		100%	100%
Average Fitness & Gates		3	3
Number of Iterations		3.25	6.65

4.2 Case Study 2

The QEPSO algorithm evolves the circuit for the truth table given in Table 3. Table 4 shows the results obtained by MGA [4], PSO and the QEPSO. The circuit representation structure (Fig. 2) is a 5×5. The circuits obtained by the MGA, PSO and the QEPSO are shown in Figs. 5 and 6, where the PSO and the hybrid inspired QEPSO obtained the same circuits. Both PSO and QEPSO find feasible solutions all the time. The QEPSO finds the best feasible solution all the time unlike PSO or GA.

Table 3. Truth table for case study 2

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Table 4. Result comparison of the MGA, PSO, and the PSO and QEA hybrid for case study 2

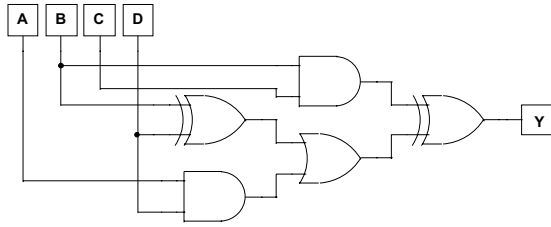| Case Study 2			
	MGA	PSO	QEPSO
Min Gates	5	5	5
Equation	$(C \bullet B) \otimes [(D \bullet A) + (B \otimes D)]$	$[(A \bullet D) + (A \otimes C)] \otimes (B \bullet C)$	$[(A \bullet D) + (A \otimes C)] \otimes (B \bullet C)$
Types of Gates	2 XOR, 2 AND, 1 OR	2 XOR, 1 OR, 2 AND	2 XOR, 1 OR, 2 AND
Percent of Best Solution	10%	60%	100%
Feasible Circuits	70%	100%	100%
Average Fitness & Gates		5.90	5
Num of Iterations		1032.40	1619.05

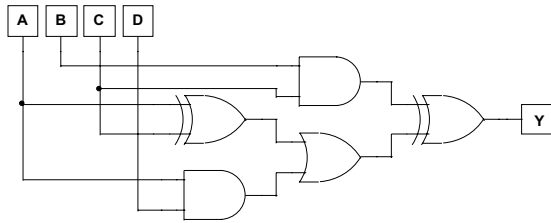Figure 5. Evolved circuit by MGA for case study 2

Figure 6. Evolved circuit by PSO and QEPSO for case study 2

4.2 Case Study 3

The QEPSO algorithm evolves the circuit for the truth table given in Table 5. Table 6 shows the results obtained by MGA [4], PSO and the QEPSO. The circuit representation structure (Fig. 2) is a 5×5 for this case study. The circuits obtained by the PSO and the QEPSO are identical and is shown in Fig. 7. Both PSO and QEPSO find feasible solutions all the time but the QEPSO finds them slightly faster. QEPSO unlike PSO finds the best feasible solution all time.

Table 6. Result comparison of the Human Designer, PSO, and QEPSO hybrid for case study 3

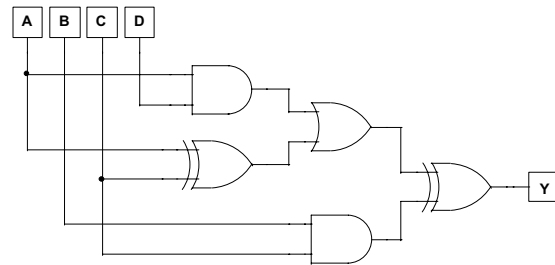| Case Study 3			
	Human Designer	PSO	QEPSO
Min. Gates	12	6	6
Equation	$[(A \bullet \overline{C}) + (A \bullet B \bullet \overline{D})] \bullet [(B + \overline{C} + \overline{D}) \bullet (A + B + \overline{C})]$	$[\overline{C} + (B \otimes D)] \otimes [(C \bullet D) + A]$	$[\overline{C} + (B \otimes D)] \otimes [(C \bullet D) + A]$
Types of Gates	5 AND, 4 OR, 3 NOT	2 XOR, 2 OR, 1 AND, 1 NOT	2 XOR, 2 OR, 1 AND, 1 NOT
Percent of Best Solution		70%	100%
Feasible Circuits		100%	100%
Average Fitness & Gates		6.55	6
Num of Iterations		1187.00	1122.55

Figure 7. Evolved circuit by PSO and QEPSO for case study 3

5. Conclusion

The new algorithm QEPSO inspired from quantum computing, particle swarm and evolutionary algorithms has been successfully applied for evolving combinational logic circuit design. The preliminary case studies and results show that the QEPSO is able to evolve feasible circuits with minimal gates all the time unlike the PSO or the MGA algorithms reported in literature. This is an important requirement for hardware evolution especially intrinsic evolution where small size and minimal power consumption of circuits may be preferred. Future work will involve robust testing of the proposed QEPSO algorithm on larger circuits and further refinement of the algorithm to speed up the evolution process.

Table 5. Truth table for case study 3

A	B	C	D	Z
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

6. Acknowledgment

7. References

[1]. M. Karnaugh, "The map method for synthesis of combinational logic circuits," *AIEE Trans.*, Vol. 72, no. 9, September 1953, pp. 593 – 599.

[2]. W.V. Quine, "A Way to Simplify Truth Functions," *American Mathematical Monthly*, 1955, Vol. 62, pp. 627–631.

[3]. E.J. McCluskey, "Minimization of Boolean Functions," *Bell Systems Technical Journal*, November 1956, Vol. 35, pp. 1417–1444.

[4]. C.A. Coello Coello, A.D. Christiansen, and A.A. Hernández, "Automated Design of Combinational Logic Circuits using Genetic Algorithms," *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms, ICANNGA'97*, Springer Verlag, 1997, pp. 335 – 338.

[5]. J.F. Miller, D. Job, and V. K. Vassiley, "Principles in the Evolutionary Design of Digital Circuits," *Genetic Programming and Evolvable Machines*, 2000, Vol. 1, No. 3, pp. 259 – 288.

[6]. C.A. Coello Coello, E.H. Luna, and A.H. Aguirre, "A Comparative Study of Encodings to Design Combinational Logic Circuits Using Particle Swarm Optimization," *2004 NASA/DoD Conference on Evolvable Hardware*, Seattle, Washington, USA, June 24 – 26, 2004, pp. 71 – 78.

[7]. V.G. Gudise, and G.K. Venayagamoorthy, "Evolving Digital Circuits Using Particle Swarm," *Proceedings of the INNS-IEEE International Joint Conference on Neural Networks*, Portland, OR, USA, July 20 – 24, 2003, pp. 468 – 472.

[8]. R.C. Eberhart, and J. Kennedy, "A New Optimizer Using Particles Swarm Theory," *Micro Machine and Human Science, MHS '95, Preceedings of the Sixth International Symposium*, October 4 – 6, 1995, pp. 39 – 43.

[9]. Vitaliy, Feoktistov, Stefan, and Janaqi. "Generalization of the Strategies in Differential Evolution," *IEEE 18th International Parallel and Distributed Processing Symposium*, Santa Fe, New Mexico, USA, April 26 – 30, 2004, pp. 165 – 170.

[10]. P.W. Moore, and G. K. Venayagamoorthy, "Evolving Digital Circuits Using Hybrid Particle Swarm Optimization and Differential Evolution," *Conference on Neuro-Computing and Evolving Intelligence*, Auckland, New Zealand, December 13 – 15, 2004, pp. 71 – 73.

[11]. Kuk-Hyun Han, and Jong-Hwan Kim, "Quantum-Inspired Evolutionary Algorithm for a Class of Combinatorial Optimization," *IEEE Transactions on Evolutionary Computation, Vol. 6, No. 6,* December 2002, pp. 580 – 593.

IEEE COMPUTER SOCIETY