FPGA-Based Circuit Model Emulation of Quantum Algorithms

Mahdi Aminian, Mehdi Saeedi, Morteza Saheb Zamani, Mehdi Sedighi
Quantum Design Automation Lab
Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran
{mahdi_aminian, msaeedi, szamani, msedighi}@aut.ac.ir

Abstract

It can be shown that if quantum algorithms run on quantum computers, their processing speeds improve exponentially compared to their classical counterparts. However, due to the lack of quantum computers circuit model of quantum algorithms are currently simulated using classical computers to verify their functionalities. On the other hand, software simulation cannot use the intrinsic parallelism of quantum algorithm efficiently. To address the problem, in this paper hardware emulation of quantum algorithms are discussed. To emulate quantum algorithms using FPGAs, a new representation for quantum bits is emulated that improves the emulation of quantum circuits considerably. This representation could be used in both distinct and entangle qubit states.

1. Introduction

It has been shown that quantum computing could improve the rate of advance in processing power at least for several applications [1], [2]. In other words, there are several problems that cannot be executed on a classical Turing machine as efficiently as a quantum computer. As a result, quantum computing has received significant attentions recently.

On the other hand, there are many challenges that need to be solved to build a scalable quantum computer [2], [3]. Due to the lack of an existing quantum computer, simulating quantum algorithms on a classical computer is widely used to verify the functionalities of quantum algorithms [4], [5]. However, software simulation cannot profit the intrinsic parallelism of quantum algorithms completely. On the other hand, using the parallel nature of hardware architectures may be more suitable to efficiently emulate quantum algorithms.

In this paper, an efficient method is proposed to emulate quantum algorithms using a classical FPGA. The rest of the paper is organized as follows. In Section 2, basic concepts are presented. Previous work is reviewed in Section 3. The proposed method and the experimental results are described in Section 4 and Section 5, respectively and finally, Section 6 concludes the paper.

2. Quantum computing

2.1. Basic concepts

While a classical bit is represented by an electrical voltage value or a wire current value, a quantum bit, *qubit* in short, can take any linear combination of the two basic states $|0\rangle$ and $|1\rangle$ called a superposition state [1]. In other words, the state of a qubit $|\psi\rangle$, can be described by (1) where C_1 and C_2 are complex numbers and $\|C_1\|^2+\|C_2\|^2=1$. On the other hand, the state of a qubit can also be shown by a 2-dimentionl vector $[C_1\ C_2]^T$ denoted as (2) [3].

$$|\psi\rangle = C_1|0\rangle + C_2|1\rangle \quad (1)$$

$$|\psi\rangle = C_1\begin{bmatrix}1\\0\end{bmatrix} + C_2\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}C_1\\C_2\end{bmatrix} \quad (2)$$

Unlike the classical bits, it is not possible to find the exact value of an unknown qubit using a measurement operator [1]. In other words, while the state of a qubit may be any linear combination of two possible basic states $|0\rangle$ and $|1\rangle$, upon measurement its state collapses to $|0\rangle$ or $|1\rangle$ with the probability of $\|C_1\|^2$ and $\|C_2\|^2$, respectively [17].

Based on the above notations, a quantum system of size N can be constructed using the tensor product [1]. The property of working on multiple input states simultaneously leads to a significant parallelism in quantum algorithms and it is one of the major quantum computation advantages.

Furthermore, the state of a qubit may completely depend on the state of another qubit. This amazing property, known as *entanglement*, is one of the other

978-0-7695-3170-0/08 $25.00 © 2008 IEEE
DOI 10.1109/ISVLSI.2008.43

benefits of quantum computation. While two qubits are entangled, its state cannot be expressed as tensor product of two distinct qubits [1].

2.2. Quantum gates

A quantum system which performs a specific operation on a selected set of qubits in a fixed period of time is called a *quantum gate*. It can be shown that the behavior of a quantum gate is always linear [6]. Therefore, a quantum gate can be represented by a unitary matrix, called QMatrix [12]. Previously, various quantum gates with different functionalities have been proposed [1], [2], [6], among them, the following gates are used in this paper and defined as:

2.2.1. X (Figure 1-a). X is a 1-qubit gate that performs the NOT operation on the input qubit. In other words, the state $|\psi\rangle$ of (1) is changed to $|\psi_1\rangle$ illustrated as (3). The QMatrix of X, i.e. U_X, is also shown in (3).

$$|\psi_1\rangle = C_2|0\rangle + C_1|1\rangle, \quad U_X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{3}$$

2.2.2. Y (Figure 1-b). As another 1-qubit gate, consider the gate Y which changes the state $|\psi\rangle$ to $|\psi_2\rangle$ shown in (4) with the QMatrix U_Y.

$$|\psi_2\rangle = -iC_2|0\rangle + iC_1|1\rangle, \quad U_Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{4}$$

2.2.3. Z (Figure 1-c). To invert the phase of the input qubit $|\psi\rangle$, the Z gate with QMatrix U_Z defined as (5) could be applied.

$$|\psi_3\rangle = C_1|0\rangle - C_2|1\rangle, \quad U_Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{5}$$

2.2.4. Controlled-NOT (CNOT, Figure 1-d). This is a 2-input gate with two control and target qubits, where the target is complemented if the control input is equal to 1. This gate causes the entanglement of its two inputs. As a result, the output of this gate cannot be shown as two distinct qubits using the tensor product. The QMatrix of this gate is shown in (6).

$$|\psi_{Control}\rangle = C_1|0\rangle + C_2|1\rangle, \quad |\psi_{Target}\rangle = C_1'|0\rangle + C_2'|1\rangle$$
$$|\psi_4\rangle = C_1C_1'|00\rangle + C_1C_2'|01\rangle + C_2C_1'|11\rangle + C_2C_2'|10\rangle \tag{6}$$
$$U_{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2.2.5. Hadamard (H, Figure 1-e). This 1-input gate is used to produce a superposition state. The QMatrix of this gate and the result state $|\psi_5\rangle$ after applying H are shown in (7).

$$|\psi_5\rangle = \frac{1}{\sqrt{2}}((C_1 + C_2)|0\rangle + (C_1 - C_2)|1\rangle),$$
$$U_H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{7}$$

2.2.6. Phase-Shift (PS, Figure 1-f). The *PS* gate is a 1-input gate which changes the phase of vector $|1\rangle$ of the input state. In other words, the result of applying this gate on the input qubit $|\psi\rangle$ is $|\psi_6\rangle$ as shown in (8).

$$|\psi_6\rangle = C_1|0\rangle + e^{i\varphi}C_2|1\rangle, \qquad U_{PS} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix} \tag{8}$$

2.2.7. Rotate (R, Figure 1-g). The R gate is a special case of *PS* gate with a specific phase. The result of applying it on the input qubit $|\psi\rangle$ and its QMatrix are shown in (9).

$$|\psi_7\rangle = C_1|0\rangle + e^{2\pi i/2^j}C_2|1\rangle,$$
$$U_{R_j} = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^j} \end{bmatrix} \tag{9}$$

2.2.8. Controlled-Rotate (CR, Figure 1-h). This is a 2-input gate with two control and target qubits, where the target is rotated if the control input is equal to 1. This gate also causes the entanglement of its two inputs. The QMatrix of this gate is shown in (10).

$$|\psi_8\rangle = C_1C_1'|00\rangle + C_1C_2'|01\rangle + e^{2\pi i/2^j}C_2C_1'|10\rangle + e^{2\pi i/2^j}C_2C_2'|11\rangle, \tag{10}$$
$$U_{CR_j} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/2^j} \end{bmatrix}$$

Since each unitary $2^n \times 2^n$ matrix can describe a valid quantum gate of size n, it can be said that the number of quantum gates are infinite [6]. However, only a finite set of gates is needed to implement quantum algorithms where these gates comprise a universal set of quantum gates [1]. The set of CNOT, H, and PS gates that are used in this paper is universal set of quantum gates [1].

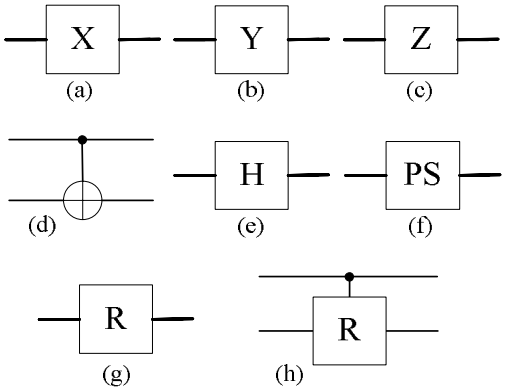

Figure 1. Symbols of quantum gates, (a) X, (b) Y, (c) Z, (d) Controlled-NOT, (e) Hadamard, (f) Phase-Shift, (g) Rotate, (h) Controlled-Rotate

3. Previous Work

When qubits are entangled, the size of required QMatrix is increased. As an example, for a system with three entangled qubits, the size of a basic 2×2 *H* QMatrix is changed to $2^3 \times 2^3$. Therefore, to implement it on a classical computer, too many resources will be needed. As a result, software simulation cannot be done efficiently due to exponential increase in the size of matrices [7].

The authors of [7] used the circuit model of a quantum algorithm to evaluate the functionality of a circuit using FPGA. In this method, the complex numbers C_1 and C_2 in (1) are represented as fixed point numbers and the gates *X*, *Z*, *CNOT*, *PS* and *H* are implemented based on multiply and add operations and several coefficient swapping. At each state, the gate outputs are saved using intermediate registers, while, it consumes too many registers to save.

In [8] DSP techniques are used to implement quantum gates. In other words, gates that can be implemented by a coefficient swapping operation are emulated based on the Network of Butterflies model [8]. However, only a small set of gates including classical reversible gates can be implemented by this method.

In [9], [10], a hardware emulation technique was proposed to implement a limited group of quantum algorithms. In other words, emulation of the algorithms containing an *H* in the first stage and a *CNOT* in the second stage is the main topic of this paper.

4. Proposed method

The goal of this paper is to relax gate type limitation of previous emulation methods. By adding several new bits to stored coefficients, a lot of quantum algorithms (for example see [11]-[15]) that only consist of *X*, *Y*, *Z*, and *CNOT* gates are simulated quickly with significant decrease in the required resources.

Based on the required operations, the used library of this paper is divided into two separated parts, where the first one includes the gates *H*, *PS*, *R*, and *CR*, and the second one contains the gates *X*, *Y*, *Z*, and *CNOT*. For the sake of simplicity, these two sets are named *HRC* and *XYZC*, respectively.

To represent complex coefficients of each qubit, fixed-point numbers are represented as Figure 2. Since each qubit has two complex coefficients, it can be represented by four fixed-point numbers. In addition, the number of mantissa bits in Figure 2 can be increased to attain more emulation accuracy.

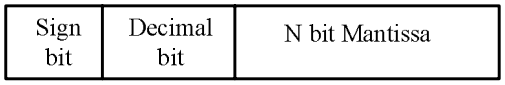

Figure 2. Representation of fixed-point numbers

4.1. The HRC group

Based on (7), *H* needs four multiplications and four additions on fixed-point numbers. Similarly, the implementation of *PS* gate needs four multiplications and two additions. As *R* can be seen as a special case of *PS* gate, the same number of resources should be assigned.

The two inputs *CR* gate rotating the target qubit if the control line is equal to 1 produces an entanglement state. When the entanglement occurs, more resources are needed for simulating. For example, applying a 1-input *H* gate on three entangled qubits needs sixteen multipliers and sixteen adders while applying this gate on one qubit needs four multipliers and four adders.

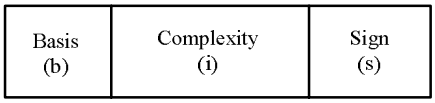

Figure 3. Added bits to qubit coefficient for the XYZC gate family

4.2. The XYZC group

The gates of the *XYZC* group are implemented using three extra bits added to each qubit coefficient as shown in Figure 3. This method causes that needed registers to save intermediate coefficients become zero. Therefore, these gates are efficiently manipulated as shown in Figure 4.

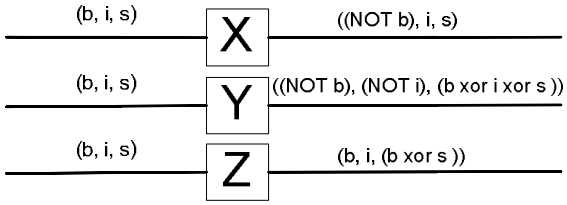

Figure 4. Implementation of X, Y, Z gates in the bit mode

However, in the case of *CNOT* gate, entangled qubits are produced and additional bits should also be considered as shown in Figure 5 where n is the number of qubits. In the case of entanglement, the number of additional bits will be too small. *CNOT* gate is implemented by XORing the control and target bits for this case.

Basis ($b_n \ldots b_1$)	Complexity (i)	Sign (s)

Figure 5. Added bits to entangled qubits coefficients

This strategy is used when a given circuit is built from only *XYZC* group gates. When both XYZC and HRC gates are used, the applied policy is different and gate operations are implemented by a coefficient swapping operator using intermediate registers as stated below:

1) The gate *X* swaps the complex coefficients C_1 and C_2. Therefore, it needs four registers.
2) The gate *Z* is implemented by multiplying -1 to the complex coefficient C_2.
3) For the gate *Y*, in addition to swapping coefficients, multiplying the complex number i (or –i) is also required. It is implemented by swapping the real and imaginary parts of the coefficients and inverting the sign bit if needed.
4) The outputs of a *CNOT* gate are produced in an entangled format as shown in Figure 6. To emulate it, the tensor product of distinct qubits should be used to create several new entangled qubits where four complex number multiplications are needed. Furthermore, applying the *CNOT* gate on two entangled qubits needs swapping the coefficients C_2C_1' and C_2C_2'.

$$|\psi_1\rangle = a_1|0\rangle + a_2|1\rangle$$

$$|\psi_2\rangle = b_1|0\rangle + b_2|1\rangle$$

$$|\psi_1\psi_2\rangle = a_1b_1|00\rangle + a_1b_2|01\rangle + a_2b_1|10\rangle + a_2b_2|11\rangle$$

Figure 6. Producing two entangled qubits using the tensor product

As shown in the experimental results section, the implementation of *XYZC* gates needs a few gates and registers for both distinct and entangled qubits (Table 1). When a quantum circuit includes only *X*, *Y*, *Z*, and *CNOT* gates, it is better to use entangled qubits from the beginning, as the cost of distinct to entangled qubit conversion is too high. In the next section, the experimental results are shown.

6. Experimental results

The proposed emulation method was implemented using VHDL and evaluated for various library gates in both distinct and entangled qubits.

Table 1 depicts logic cell usage for the implementation of each gate with two different mantissa sizes. In addition, the proposed method was compared with another hardware emulation method proposed in [7]. ALTERA STRATIX EP1S80B956C6 [19] device was chosen for all of the experiments. For CNOT gate, a two entangled qubits circuit was attempted. Furthermore, the number of intermediate registers was computed for the method of [7]. In this table, the percentage of LC usage improvement is denoted as *Imp*.

Table 1. LC usage on STRATIX EP1S80B956C6 device

Gate	Mantissa = 8 bits			Mantissa = 16 bits		
	Proposed method	[7]	Imp %	Proposed method	[7]	Imp %
H	398	704	43	808	1284	37
PS	200	386	48	405	708	43
X	2	40	95	2	72	97
Y	6	-	-	6	-	-
Z	2	40	95	2	72	97
CNOT	4	120	96	4	375	99

As shown in Table 1, the improvements for some circuits (e.g. Z gate), are much more than others (e.g. H gate), this is because the emulation method for circuits with XYZC gates are more efficient.

The Quantum Fourier Transform (QFT) is an important quantum algorithm with great applications in

phase estimation, order-finding and factoring algorithms [1], [7], [16]. An N-qubit QFT circuit is shown in Figure 7. This circuit is constructed using *H* and *CR* gates where the *CR* gate produces an entangled state.

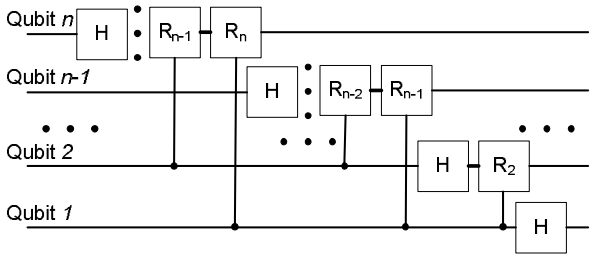

Figure 7. Circuit model of an N-qubit QFT algorithm

The synthesis results for a 3-input QFT algorithm is shown in Table 2. In addition, the run time of both the proposed method and a software simulator, called *Libquantum* [18], is depicted in Table 3.

Table 2. Synthesis result of QFT algorithm

Mantissa	LC Usage	Clock Frequency (MHz)	
		Proposed method	[7]
8	3905	137.9	-
16	8197	131.3	82.1

Table 3. The comparison of software simulation and the proposed hardware emulation method for the 3-input QFT algorithm

Mantissa (bits)	Run Time (Seconds)	
	Libquantum [17]	Proposed Method
16	40×10^{-6}	46×10^{-9}

To further analyze the proposed emulation method, several other quantum circuits, i.e. quantum full adder [2] and some of the available benchmarks [20] were synthesized. The characteristics of the circuits are illustrated in Table 4 and the synthesis results are depicted in Table 5.

Table 4. Specification of benchmark circuits

Circuit	# of Qubits	# of Gates	# of NOT	# of CNOT	# of C^2NOT	# of C^3NOT
Full Adder	5	6	0	3	3	0
3_17tc	3	6	1	3	2	0
graycode6	6	5	0	5	0	0
Ham3tc	3	5	0	4	1	0
Hwb4-11-23	4	11	0	8	3	0
Mod5adder-15	6	15	6	0	5	4
rd32	4	4	0	2	2	0
xor5d1	5	4	0	4	0	0
Mod5d1	5	8	0	4	4	0
Rd53d2	8	12	0	4	8	0

Table 5. LC usage for synthesis of benchmarks [20]

Circuit	M	Proposed Method			[7]		
		LC	E (ns)	T (min)	LC	E (ns)	T (min)
Full Adder	8	128	6.43	1	3840	15	4.5
	16	128	6.43	1.5	6912	15	12.5
3_17tc	8	24	4.48	0.5	960	15	1
	16	24	4.48	0.5	1728	15	1.25
graycode6	8	320	4.45	4	6400	12.5	10
	16	320	4.45	9.25	11520	12.5	25
ham3tc	8	24	4.48	0.25	800	12.5	0.75
	16	24	4.48	0.25	1440	12.5	1
hwb4-11-23	8	64	4.51	0.5	3520	27.5	1.5
	16	64	4.51	0.75	6336	27.5	3.5
mod5adder-15	8	576	11.1	10.5	19200	37.5	30.5
	16	576	11.1	27	34560	37.5	70.5
rd32	8	48	4.48	0.5	1280	10	1
	16	48	4.48	0.5	2304	10	1.25
xor5d1	8	128	5.51	1	2560	10	3.5
	16	128	5.51	1.75	4608	10	8
mod5d1	8	224	10.4	1.5	5120	20	4.5
	16	224	10.4	3.75	9216	20	19
rd53d2	8	3072	7.5	182	U	-	>240
	16	3072	7.5	210	U	-	>240

Note that the synthesis results of [7] were not reported in the paper. To make it possible to compare the results of the proposed method with [7], we implemented it and its results are compared to those of our method in Table 5. The following notations were used for this table: M as mantissa, E as required emulation time in nanosecond, *T* as total synthesis time in minutes and U to mean that it is impossible to fit a circuit in the selected device. As shown in Table 5, it can be verified that the required LCs as well as the emulation time for the proposed method are fewer than the method of [7], significantly.

7. Conclusion

In this paper, an efficient quantum circuit emulation technique is proposed. As this method uses the parallelism of quantum algorithms, it offers more efficiency than software simulation. Compared with the available hardware emulation methods, the proposed technique uses fewer logic cells for the implementation of various quantum algorithms. In other words, by using a novel representation schema and simulating the behaviors of various quantum gates using simple logical gates, a great improvement is obtained. Due to the efficiency of the proposed method, the simulation of large size quantum algorithms is also possible.

10. References

[1] M. A. Nielsen, I.L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Wiley Publishing, 2000.

[2] D. C. Marinescu, G. M. Marinescu, *Approaching Quantum Computing*, Prentice Hall, 2005.

[3] S. Imre, F. Balazs, *Quantum Computing and communications: An Engineering Approach*, John Wiley Publishing, 2005.

[4] I. G. Karafyllidis, "Quantum Computer Simulator Based on the Circuit Model of Quantum Computation", *IEEE Transactions on Circuits and Systems*, Volume 52, 2005, PP. 1590 – 1596.

[5] G. F. Viamontes, I. L. Markov, J. P. Hayes, "Improving Gate-Level Simulation of Quantum Circuits", *Quantum Information Processing*, Volume 2(5), 2003, PP. 347-380.

[6] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Schor, T. Sleator, J. Smolin, H. Weinfurter, "Elementary Gates for Quantum Computation", *Physical Rev*, No. 52, 1995, pp. 3457-3467.

[7] A. U. Khalid, Z. Zilic, K. Radecka, "FPGA Emulation of Quantum Circuits", *International Conference on Computer Design*, 2004, PP. 310- 315.

[8] G. Negovetic, M. Perkowski, M. Lukac, A. Buller, "Evolving Quantum Circuits and an FPGA-Based Quantum Computing Emulator", *International Workshop on Boolean Problems*, 2002.

[9] M. Fujishima, K. Saito, M. Onouchi, H. Hoh, "High-Speed Processor for Quantum-Computing Emulation and its Applications", *International Symposium on Circuits and Systems*, Volume 4, 2003, PP. IV-884 - IV-887.

[10] M. Fujishima, "FPGA-Based High-Speed Emulator of Quantum Computing", *International Conference on Field-Programmable Technology*, 2003, PP. 21 - 26.

[11] K. Iwama, Y. Kambayashi, S. Yamashita, "Transformation Rules for Designing CNOT-Based Quantum Circuits", *Design Automation Conference*, 2002, PP. 419-424.

[12] M. Saeedi, M. Sedighi, M. Saheb Zamani, "A New Methodology for Quantum Circuit Synthesis: CNOT-Based Circuits as an Example", *International Workshop on Logic Synthesis*, 2007.

[13] A. Younes, J. Miller, "Automated method for building CNOT based quantum circuits for Boolean functions", *Los Alamos physics preprint archive*, quant-ph/0304099, 2003.

[14] A. Younes, J. Miller, "Representation of Boolean Quantum Circuits as Reed-Muller Expansions", *Los Alamos Physics preprint archive*, quant-ph/0305134, 2003.

[15] M. Saeedi, M. Sedighi, M. Saheb Zamani, "A Novel Synthesis Algorithm for Reversible Circuits", *International Conference on Computer-Aided Design*, 2007, PP. 65-68.

[16] Z. Zilic, K. Radecka, "The Role of Super-fast Transforms in Speeding up Quantum Computations", *International Symposium on Multiple-Valued Logic*, 2002, PP. 129 - 135.

[17] Wikipedia, "Quantum Computer", July 2007.

[18] "LibQuantum", Online Quantum Library Documentation, Available: http://www.enyo.de/libquantum/

[19] http://www.altera.com/

[20] D. Maslov, "Reversible Logic Synthesis Benchmarks", Available: http://webhome.cs.uvic.ca/~dmaslov/