IEICE Electronics Express, Vol.5, No.17, 638–643

Synthesis of reversible circuits using a moving forward strategy

Mehdi Saeedi[a)], **Mehdi Sedighi, and Morteza Saheb Zamani**

Quantum Design Automation Lab

Computer Engineering Department, Amirkabir University of Technology,

Hafez St., Tehran 15914, Iran

a) *msaeedi@aut.ac.ir*

Abstract: Reversible circuits have applications in various research areas including signal processing, cryptography and quantum computation. In this paper, a non-search based moving forward synthesis algorithm (MOSAIC) for Boolean reversible circuits is proposed to convert an arbitrary well-formed matrix into an identity matrix using a set of reversible gates. In contrast with the widely used search-based methods, MOSAIC is guaranteed to produce a result and can lead to a solution in much fewer algorithmic steps. To evaluate the proposed algorithms, different circuits and benchmarks were used that show the efficiency of the proposed algorithm to lead a result.

Keywords: quantum computing, reversible circuits, synthesis

Classification: Integrated circuits

References

[1] International Technology Roadmap for Semiconductors, 2005.

[2] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*, Cambridge University Press, 2000.

[3] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," *Proc. DAC*, pp. 318–323, 2003.

[4] M. Saeedi, M. Saheb Zamani, and M. Sedighi, "Algebraic Characterization of CNOT-Based Quantum Circuits with its Applications on Logic Synthesis," *Proc. DSD*, pp. 339–346, 2007.

[5] P. Gupta, A. Agrawal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits," *TCAD*, vol. 25, no. 11, pp. 2317–2330, 2006.

[6] M. Saeedi, M. Saheb Zamani, and M. Sedighi, "On the behavior of substitution-based reversible circuit synthesis algorithms: investigation and improvement," *Proc. ISVLSI*, pp. 428–436, 2007.

[7] D. Große et al., "Exact synthesis of elementary quantum gate circuits for reversible functions with don't cares," *Proc. ISMVL*, 2008.

[8] [Online] http://webhome.cs.uvic.ca/~dmaslov

[9] [Online] http://ceit.aut.ac.ir/QDA/benchmarks

DOI: 10.1587/elex.5.638
Received May 27, 2008
Accepted August 04, 2008
Published September 10, 2008

ELectronics EXpress

IEICE Electronics Express, Vol.5, No.17, 638–643

1 Introduction

The limitations of current CMOS technology for increasing the processing power in the near future led researchers to work on new computational models [1]. Among various proposed models, quantum computing has the potential to increase the rate of advances in computing power drastically [2]. However, a large quantum circuit requires a systematic synthesis algorithm [3, 4, 5, 6, 7].

Boolean reversible circuits can be viewed as a special case of quantum circuits as quantum evolution is reversible in nature [1]. While the intrinsic parallelism of quantum algorithms is not available without purely quantum gates, popular universal quantum gate libraries often contain a set of Boolean reversible gates [2]. As a result, working on automatic synthesis methods for reversible circuits has received significant attentions recently [3, 4, 5, 6, 7].

In this paper, the characterizations of matrix representation of Boolean reversible circuits are used to propose a moving forward strategy to synthesize circuits fast.

2 Preliminaries

A quantum bit (or qubit) is typically derived from the state of a two-level quantum system. An n-qubit quantum gate is a device that performs a specific unitary operation on selected qubits in a specific period of time. An n-qubit quantum gate is associated with a unitary $2^n \times 2^n$ matrix, QMatrix, describing its functionality.

An n-input, n-output CNOT$^n(x_1, x_2, \cdots, x_n)$ gate passes the first n-1 lines unchanged and flips the n^{th} line if the control lines are all one. For $n = 1$, $n = 2$, and $n = 3$ the gates are called NOT, CNOT and C^{2}NOT (Toffoli), respectively. These gates comprise an important class of quantum gates that mainly used in Boolean reversible circuits [3, 4, 5, 6]. By combining some primitive gates, any related quantum gate and circuit can be constructed [2].

Several algorithms have recently been proposed to synthesize a circuit. Some authors used transformation-based algorithms, e.g. [3], which apply local transformations to optimize the results of other algorithms. Several authors, e.g. [7], used SAT-based formulations for the synthesis problem where the application of these approaches is limited to circuits with a few gates due to high complexities of resulted SAT clauses. Some authors proposed search-based methods to synthesize a given specification, e.g. [4, 5, 6]. These algorithms need a time-consuming procedure for the examination of all possible gates to lead to a result. In addition, the proposed algorithm did not guarantee to reach a valid result.

3 Synthesis Algorithm

A QMatrix of an n-qubit quantum circuit is well-formed if its elements are either zero or one and each column or row has exactly one element with a value of 1 [4]. It has been shown that the set of well-formed matrices is closed under tensor product and matrix multiplication [4]. As a result, the

DOI: 10.1587/elex.5.638
Received May 27, 2008
Accepted August 04, 2008
Published September 10, 2008

QMatrix of a C^2NOT gate as well as the QMatrix of a circuit containing only C^2NOT gates are well-formed [4]. Since the C^2NOT gate is universal for Boolean reversible logic [2], the QMatrix of a Boolean reversible circuit is also well-formed. Hereafter, a general k-qubit gate is represented as C^kNOT and n is used as the number of available qubits. The QMatrix M is also denoted as $M(x_1, x_2, \cdots)$ where x_i is the row number of an element with the value of 1 in the i^{th} column.

Definition 1: The application of a k-qubit gate on a circuit is called L_k-QTranslation. As the set of well-formed QMatrix is closed under matrix multiplication, the result of using an L_k-QTranslation is also well-formed.

Definition 2: The i^{th} and the j^{th} rows of a QMatrix form a quantum pair ($QPair_{i,j}$) if the numbers i and j differ in only one bit position.

Definition 3: The 2^k rows of a QMatrix whose row numbers have the same value on their n-k bit locations form a single group called C^kQPair.

Lemma 1: (a) The application of an L_k-QTranslation on M leads to 2^{n-k-1} row exchanges. (b) Equally, exchanging the locations of 2^k QPairs of the same C^kQPair is equivalent to applying an L_{n-k-1}-QTranslation.

Lemma 2: Consider a C^kQPair of a QMatrix having 2^k rows where the n-bit row numbers have the same value on their n-k bits and two QPair rows differ from each other only in one bit position. Exchanging the locations of each $QPair_{i,j}$ ($QPair_{i,j} \in C^k$QPair) has the same effect as applying an L_{n-k+1}-QTranslation.

The proposed MOSAIC algorithm is shown in Fig. 1.

Variable *flag* is used to verify the requirement of further steps. A set of 2^n flags are used to mark the rows visited in previous steps. Initially, MOSAIC set $b = 1$ and reset all rows to be unvisited rows. Then, the algorithm selects a column c and set r to be the c row number which has a value of 1. If the

```
Algorithm MOSAIC
1.  b=1;
2.  REPEAT
3.     reset all rows of M to be unvisited;
4.     flag=TRUE;
5.     FOR each column index c of M, c∈(1..2^n)
6.        set r to be the c row number which has a value of 1;
7.        IF the r^th row is not marked as visited
8.           IF the b^th bits of r and c are not equal
9.              flag=FALSE;
10.             find the number p ∈ QPair_r,p which differs with r in its b^th bit;
11.             set q to be the column number of row p which has a value of 1;
12.             IF !(q = p AND p < r)
13.                exchange the locations of the p^th and r^th rows;
14.                mark the p^th and r^th rows as visited;
15.             END
16.          END
17.       END
18.    END
19.    b = (b + 1) mod n;
20.    Identify each C^kQPair group whose QPairs are exchanged;
21.    Extract an L_k translation for each C^kQPair based on Lemma 1 and Lemma 2;
22. UNTIL flag=TRUE
```

Fig. 1. MOSAIC synthesis algorithm

DOI: 10.1587/elex.5.638
Received May 27, 2008
Accepted August 04, 2008
Published September 10, 2008

b^{th} bits in the binary expansions of c and r are identical, the algorithm does nothing to mean that r and c have the same value in their b^{th} bit locations. Otherwise, MOSAIC tries to correct the b^{th} bit of r by exchanging the r^{th} row with another row p where $p \in \text{QPair}_{r,p}$ and differs from r in its b^{th} bit. This process is continued to consider all 2^n columns sequentially. Then b is incremented to evaluate the next bit. An L_k-translation is also extracted considering the exchanged rows. These steps are repeated until all rows are placed at their right locations.

Example 1: Consider a 3-qubit circuit with QMatrix $M(7,0,1,2,3,4,5,6)$. Fig. 2 shows the application of the proposed algorithm. In this figure, p, q and the exchanged QPair rows enclosed in {} are shown. The resulted C^kQPairs for each step are also demonstrated. The synthesized circuit of this example is {NOT(c), CNOT(c',b), C^2NOT(b',c',a)}.

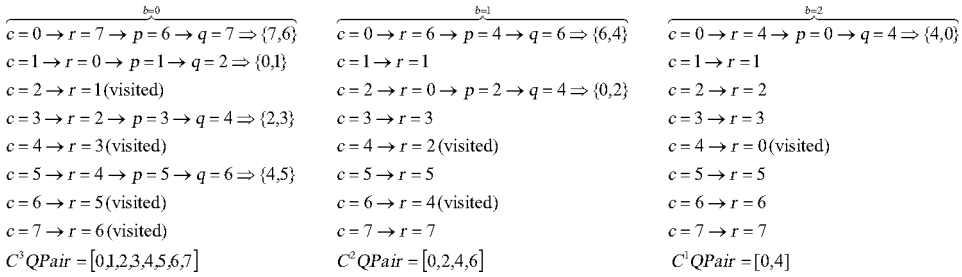

Fig. 2. The results of applying MOSAIC on Example 1

Theorem 1: MOSAIC converges to a valid circuit after several steps.
Proof: Assume that after a number of steps, several rows represented as a set Σ, are placed at their right positions and the algorithm is working on the k^{th} bit (i.e. $b = k$) of the c^{th} column and sets r to the column c row number with the value of 1. Consider the case where r differs from c in its k^{th} bit ($r \notin \Sigma$). Accordingly, the algorithm finds a row number p that differs from r only in its k^{th} bit.

If $p \in \Sigma$ and $p < r$, the algorithm does nothing to avoid instability in row locations. However, as the r^{th} row is placed at a wrong position, there must be another row, i.e. the t^{th} row, which should be exchanged with the r^{th} row during the next steps. Consider the other cases ($p \notin \Sigma$ or ($p \in \Sigma$ and $p > r$)) where the algorithm exchanges the location of the p^{th} row with that of the r^{th} row. Then, the k^{th} bit of the row r is correct and the algorithm moves forward. As each QTranslation does not change the previous results, the algorithm will gradually place all rows at right positions.

To compare MOSAIC with search-based methods in relation to the time complexity, assume a possible implementation of a QMatrix needs at most h gates.

Theorem 2: A search-based synthesis method needs $O(n \times 2^n)^h$ steps.
Proof: For a quantum circuit of size n, there are C_n^1 possible NOT gates and C_n^2 possible C^2NOT gates in which one of its two inputs can be the target output. On the other hand, as each of the C^2NOT inputs could be used as

DOI: 10.1587/elex.5.638
Received May 27, 2008
Accepted August 04, 2008
Published September 10, 2008

the target qubit, the total number of $2 \times C_n^2$ gates can be obtained.

In contrast, for a $(k+1)$-qubit gate, $k \in (2, 3, \cdots, n\text{-}1)$, there are C_{n-1}^k possible gates when the target can be any i^{th} $(i \in [1, n])$ qubit. Considering all possible qubits as the target variable leads to the total number of $n \times C_{n-1}^k$ $(k+1)$-qubit gates. Therefore, the total number of gates is $C_n^1 + 2 \times C_n^2 + \mathrm{n} \times (\Sigma_{i \in (2 \cdots n-1)} C_{n-1}^i) = \mathrm{n} \times 2^{n-1}$. As at most h steps are required, search-based methods need $\mathrm{O}(\mathrm{n} \times 2^n)^h$ node searches.

Theorem 3: MOSAIC needs at most $\mathrm{O}(h \times 2^n)$ steps to reach a result.
Proof: It can be verified that except the lines 2 and 5 of the algorithm, the other lines take only O(1) time. The time complexities of line 5 and line 2 are $\mathrm{O}(2^n)$ and $\mathrm{O}(h)$, respectively. As a result, MOSAIC needs $\mathrm{O}(h \times 2^n)$ steps.

4 Experimental Results

All of the experiments were done on an Intel Pentium IV 3 GHz computer with 1 GB memory. To evaluate the algorithm, we used several examples and benchmarks taken from literature [3, 4, 5, 6, 8, 9]. Furthermore, we compared the results of our algorithm with several recent papers including [3, 5, 6].

Table I shows the synthesized results of different algorithms for some examples. For search-based algorithms the number of searched nodes and

Table I. Comparison of average results for some examples

	No. of Steps vs. No. of Searched Nodes				No. of Gates			
QMatrix	MOSAIC	[3]	[6]	[5]	MOSAIC	[3]	[6]	[5]
1	40	32	15	11	5	4	4	4
2	16	24	300	761	3	3	3	3
3	32	24	10	7	3	3	3	3
4	32	56	786	156	5	7	5	5
5	96	224	8256	9515	7	14	7	7
6	16	24	4	4	3	3	4	3
7	48	32	5	5	4	4	4	4
8	32	64	139	230	4	4	4	4
9	112	288	16033	146244	10	18	12	11
10	256	256	-	-	22	16	-	-
11	400	192	-	-	40	12	-	-
12	32	80	66	-	6	10	7	-
13	40	96	77	-	10	12	6	-
14	40	56	4387	-	8	7	7	-
15	512	1408	>	>	29	44	>	>
16	1472	1664	>	>	82	52	>	>
17	480	1344	>	>	30	42	>	>
18	32	80	352	-	6	10	7	-
19	208	288	678	-	6	10	7	-
20	240	208	9712	-	20	13	14	-
21	144	368	74521	-	20	23	17	-
22	192	352	85191	-	21	22	16	-

DOI: 10.1587/elex.5.638
Received May 27, 2008
Accepted August 04, 2008
Published September 10, 2008

Table II. Comparison of results for available benchmarks

Circuit	Benchmark [8]	No. of Gates MOSAIC	[5]	Circuit	Benchmark [8]	No. of Gates MOSAIC	[5]
1	2of5	10	20	17	hwb7	404	-
2	rd32	4	4	18	decode24	10	11
3	rd53	16	13	19	shift10	22	27
4	rd73	15	-	20	5one013	22	19
5	3-17	6	6	21	5one245	27	20
6	4-49	14	13	22	6one135	2	5
7	alu	29	18	23	6one0246	2	6
8	cycle10-2	24	-	24	majority3	5	4
9	xor5	2	4	25	majority5	24	16
10	4mod5	2	5	26	graycode6	7	5
11	5mod5	8	11	27	graycode10	13	9
12	ham3	4	5	28	mod5adder	26	19
13	ham7	48	24	29	mod32adder	20	15
14	hwb4	14	15	30	mod15adder	25	10
15	hwb5	42	-	31	mod64adder	31	26
16	hwb6	124	-	32	6symd2	19	-

for MOSAIC and the algorithm of [3] the number of steps were reported to have the same O(1) time complexity for all primitive operations. Number of resulted gates for each algorithm was also shown and a time limit of 60 seconds was used for the experiments. The "–" and ">" symbols are used when the circuit cannot be synthesized or required more steps, respectively. To further evaluate the cost of generated circuits, reversible benchmarks are also used and numbers of generated gates using both the method of [5] and MOSAIC are reported in Table II.

As shown in these tables, MOSAIC not only has the ability to produce a result for all of the attempted specifications but also it can reach a result in much fewer steps. It can be seen that MOSAIC can also reach a circuit with comparable cost. As the quality of search-based methods highly depends on the application order of each possible gate at each step, if the number of qubits increases or the resulted circuits need too many gates, search-based methods may not lead to a result due to memory and/or time limits.

5 Conclusions

In this paper, a non-search based synthesis algorithm was proposed which requires a few steps to synthesize a given specification. To evaluate the algorithm, we used some examples and benchmarks taken from the literature and compared the results with those generated by several recent methods. It was shown that the presented algorithm could lead to valid results for all of the circuits much faster on average with comparable gate counts.

DOI: 10.1587/elex.5.638
Received May 27, 2008
Accepted August 04, 2008
Published September 10, 2008