Automatic Synthesis of Composable Sequential Quantum Boolean Circuits

Li-Kai Chang and Fu-Chiung Cheng
Department of Computer Science and Engineering, Tatung University
Taipei, Taiwan, R.O.C.
{lkchang, cheng}@gamma.cse.ttu.edu.tw

Abstract

This paper presents a methodology to transfer self-timed circuit specifications into sequential quantum Boolean circuits (SQBCs) and composable SQBCs (CQBCs). State graphs (SGs) are used to describe the behaviors of self-timed circuits and then are translated into SQBCs based on Toffoli gates. The concept of IP (Intellectual Property) reuse is applied to the constructed SQBCs to produce reusable and composable quantum Boolean circuits (CQBCs). Therefore, these reusable CQBCs as basic modular components can be exploited to construct more complicated quantum Boolean circuits.

A set of self-timed components is successfully and automatically synthesized into CQBCs by our methodology. These CQBCs can be used as building blocks to compose control-path components of self-timed systems.

Keywords: Quantum Boolean circuits, Sequential circuits, Asynchronous circuits, State graph, Synthesis.

1. Introduction

Due to the discovery of Shor's prime factorization and Grover's fast database search algorithm [12, 13] quantum computing becomes one of the most rapidly expanding research fields. To perform quantum algorithms, required unitary operations should be expressed as a sequence of basic operations which can be implemented by a quantum computer. To implement a quantum computer, quantum Boolean circuits need to be constructed first [1].

The major differences between conventional circuits and quantum ones are their logic gates and wires [6]. Firstly, conventional circuits are based on AND, OR and NOT gates and quantum Boolean circuits are based on NOT, Controlled-Not and Controlled-Controlled-Not gates (i.e. Toffoli gates) [8]. Secondly, the wires in conventional circuits are used to connect components. This is very different in quantum Boolean circuits because wires represent time evolution.

Due to the above differences, a completely different methodology to synthesize quantum Boolean circuits must be investigated and proposed. Tsai and Kuo [1] propose a methodology to synthesize combinational quantum Boolean circuits based on transformation tables. Any general *m*-to-*n* bit combinational Boolean logic can be synthesized by using Toffoli gates. Iwama et al. [6] propose transformation rules for optimize Controlled-Not-based combinational quantum Boolean circuits and point out a design theory for a sequential quantum circuit is very interesting. Miller et al. [15] propose a transformation based algorithm for synthesizing the combinational circuits and reduction rules for optimizing the synthesized circuits. Younes and Miller [17] propose an automated method to build CNOT based quantum circuits for Boolean functions. To the best of our knowledge there are no related works on synthesizing sequential circuit behaviors into quantum Boolean circuits yet.

This paper presents a novel methodology to transfer self-timed circuit specifications into composable sequential quantum Boolean circuits. State graphs [4, 9] are used to describe the behaviors of self-timed circuits and then are translated into SQBCs based on Toffoli gates by our synthesis tool.

A set of self-timed components [7, 14] is successfully and automatically synthesized into CQBCs by our methodolgy. These CQBCs can be used as building blocks to compose control-path components of self-timed systems.

The rest of this paper is organized as follows: Section 2 introduces basic knowledge on quantum systems and self-timed systems. Section 3 and 4 present our methodologies to synthesize SQBCs and reusable CQBCs based on state graph specifications, respectively. The experimental results of SQBCs and CQBCs synthesis are given in section 5. Section 6 concludes this paper as a whole and provides some suggestions for future work.

2. Background

This section provides the background knowledge on quantum systems and self-timed systems. For quantum systems quantum bits, quantum gates and quantum circuits are briefly described and for self-timed system, state graphs are presented and used in this paper.

2.1. Fundamental of quantum systems

Circuit design and data representation in conventional computers and quantum ones are different in nature. In classical computers data are represented by bits and circuits are networks of logic gates while in quantum computers data are represented by quantum bits (Qubits) and quantum circuits are made of a sequence of unitary operations which are represented by quantum gates and quantum wires [2, 6, 8, 11].

2.1.1. Quantum gates

In classical gates, any function can be realized by NAND gates alone, which is thus known as a *universal* gate. In quantum Boolean circuits, any multiple qubit logic can be composed from *controlled-NOT* (CNOT) type logic gates.

In order to construct quantum Boolean circuits, any classical circuit can be replaced by an equivalent circuit of only reversible element, by making use of a *reversible* gate known as the *Toffoli gate* [8, 12, 13].

The Toffoli gate, shown in Fig. 1(a), has three qubits. The third qubit is the target qubit which is flipped when both control qubits (i.e. the first two qubits) are set to 1.

The action of the Toffoli gate can be summarized as $|a, b, c\rangle \rightarrow |a, b, c \oplus ab\rangle$. Furthermore, applying the Toffoli gate twice has the effect $|a, b, c\rangle \rightarrow |a, b, c \oplus ab\rangle \rightarrow |a, b, c\rangle$, and thus the Toffoli gate is reversible.

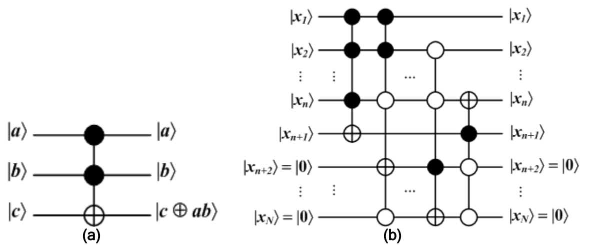


Figure 1: (a)Toffoli gate and (b)a quantum Boolean circuit

2.1.2. Quantum Boolean circuits

In quantum computing the behaviors of a quantum circuit are represented by a sequence of unitary operations applied to the qubits of the quantum circuit. The results can be read out by measuring the quantum states of the qubits. That is quantum circuits consist of a sequence of unitary operations represented by quantum gates and quantum wires.

Fig. 1(b) shows a QBC with N qubits, denoted by $|\chi_1\rangle|\chi_2\rangle\ldots|\chi_N\rangle$. The sequence of unitary operations are applied from left to right to corresponding qubits $|\chi_1\rangle$ to $|\chi_N\rangle$. $|\chi_1\rangle|\chi_2\rangle\ldots|\chi_N\rangle$ on the left-head-side is regarded as the input to the quantum circuit, and the states on the right-head-side side keep the final result. A quantum Boolean circuit can use any finite number of auxiliary qubits for storing intermediate states [6].

There are three different kinds of logic gates in Toffoli gates: one-controlled gates (denoted by closed circles), zero-controlled gates (denoted by open circles) and target gates which are similar to sum (mod 2). When all controlled gates in the same wire (i.e. quantum operation) are active, the target gate flips [8].

2.2. Self-timed systems

The operations of a quantum Boolean circuit are quite different from those of a classical synchronous circuit which are controlled by a global clock. In quantum Boolean circuits, a sequence of operations are applied to the qubits and are not controlled by a global clock. Furthermore, a quantum operation cannot be applied to a QBC unless the previous one is complete and the quantum system is stable. This behavior is similar to the fundamental mode of asynchronous circuits [9, 10, 16].

In the fundamental mode of asynchronous circuits, when the inputs of logic block are triggered, outputs are changed by the inputs and current states, and the next states of circuits are stored in the latches. The changes in inputs are forbidden until the system is stable.

Because the operations of a quantum Boolean circuit are similar to asynchronous circuit behaviors in the fundamental mode [9, 16], we exploit the specifications of asynchronous circuit design to construct sequential quantum Boolean circuits. Note that asynchronous circuits can also operate in input-output mode. For more information on this topic please refer to [9, 10].

2.2.1. State graph

State graphs (SGs) [3, 4] can be used to specify the behaviors of circuits. State graphs are directed binary coded graphs containing states (or nodes) and directed edges. An edge in SGs is labeled with input or output *signal transitions*. Each signal transition can be represented as x_i+ or x_i– for the rising (0→1) or falling (1→0) transition of signal x_i.

A node in SGs represents one state of the circuit. Each state $s \in S$ is labeled with *binary code* $\langle s(1), s(2), \ldots, s(n)\rangle$, and the value of $s(i)$ is 0 or 1. The state binary code is formed by an input binary code and an output binary code. Suppose the circuit has m-bits input and n-bits output. The input and output binary codes of node i are defined as follows:

- ib(i) $\in \{0, 1\}^m$ is the *input binary code function*,
- ob(i) $\in \{0, 1\}^n$ is the *output binary code function*.

The state binary code of node i, sbc(i), can be defined as ib(i) + ob(i) where the symbol '+' denotes the concatenation. And, the k-th state bit of node i is denoted as sbc(i, k).

For example, a modulo-3 element [7] has one input a and two outputs, Y (yes) and N (no). The specification of the module-3 in prefix-closure form is *Pref*(a?N!a?N!a?Y!)*. That is output Y is triggered when input a is triggered three times. Valid partial behaviors are: *a*, *a N*, *a N a*, *a N a N*, *a N a N a*, *a N a N a Y*, …. A possible implementation of the modulo-3 element using XOR and toggle elements is shown in Fig. 2(a).

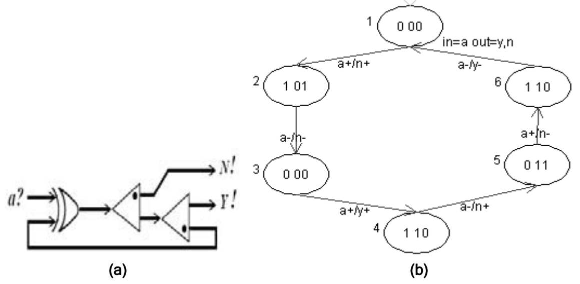


Figure 2: The (a) classical circuit and (b) SG of modulo-3

The SG of the modulo-3, shown in Fig. 2(b), has 6 states and 6 edges. The state binary code of node 1 is "000" since the ib(1) is '0' (i.e. a=0) and ob(1) is "00" (i.e. Y=0, N=0).

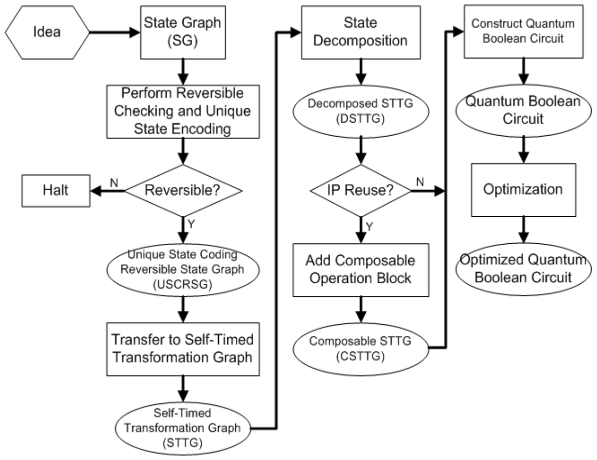
2.2.2. Unique state coding

An unambiguous state assignment is required for deriving logic. An SG has the *Unique State Coding* (USC) [3, 18] property if no two distinct states in the state graph have identical binary codes. A state graph is USC-conflict if any two states in the state graph have the same state binary code. To be synthesizable, a state graph specification of a circuit must satisfy the USC requirement.

3. Synthesis of sequential QBCs

For synthesizing SQBCs state graphs are used to specify the behaviors of circuits and are transferred to SQBCs automatically. The complete synthesis flow of our synthesis methodology, shown in Fig. 3, consists of five main steps.

First, SGs (i.e. classic circuit specifications) are transformed to USC reversible state graphs (USCRSGs) by using unique state encoding. Second, USCRSGs are transformed to self-timed transformation graphs (STTGs) which are quantum circuit specifications. Third, STTGs are transformed to decomposed STTGs (DSTTGs). Then, the DSTTGs can be optionally transformed into composable and thus reusable STTGs. Fourth, DSTTGs are synthesized into SQBCs based on Toffoli gates. Finally, the SQBCs are optimized based on reduction rules [1].



Idea
State Graph (SG)
Perform Reversible Checking and Unique State Encoding
Reversible?
N
Halt
Y
Unique State Coding Reversible State Graph (USCRSG)
Transfer to Self-Timed Transformation Graph
Self-Timed Transformation Graph (STTG)
State Decomposition
Decomposed STTG (DSTTG)
IP Reuse?
N
Y
Add Composable Operation Block
Composable STTG (CSTTG)
Construct Quantum Boolean Circuit
Quantum Boolean Circuit
Optimization
Optimized Quantum Boolean Circuit

Figure 3: Synthesis flow of SQBCs

3.1. Perform reversible checking and unique state encoding

To be synthesizable, USC property is required for both classic and quantum circuits. Quantum circuits must consist of only reversible gates; therefore, to be synthesizable for QBCs, a SG must have reversible property. Here we define reversible SG without proof as follows: a SG is irreversible if for any node j in the SG either the indegree(j)=1 or indegree(j)>1 and for all source nodes of j, i_1, i_2,…, i_k, ob(i_1)=ob(i_2)=…=ob(i_k). The USGSG shown in Fig. 2(b) is reversible since the indegree of all nodes in USCRSG is equal to one.

Figure 4 illustrates an irreversible SG which is the circuit specification of a call module [16]. This SG cannot be synthesized since node 10 has two source nodes, node 7 and node 8 and ob(node 7)$\neq$ob(node 8).

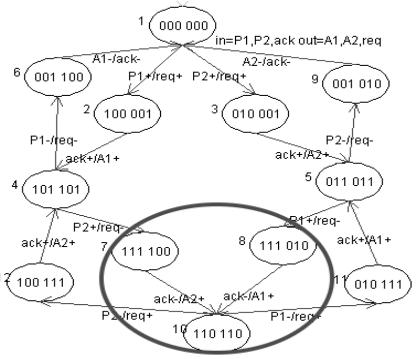


Figure 4: The irreversible SG of call module

Since an unambiguous state assignment is needed to construct both classic circuits and SQBCs, unique state encoding algorithm is applied first to avoid USC conflict. To satisfy the USC requirement, different *auxiliary state bits* are appended to the original state binary code to distinguish states in SGs. Two states (nodes) are called USC-conflict states if and only if their state binary codes are the same. If there are s states in the SG and the number of USC-conflict states for each state binary code is di ($0 \leq i < s$), then the number of auxiliary state bits needed is $k = \log\lceil max(d_i)\rceil$.

For example, the states 1, 3 and states 4, 6 in the RSG of modulo-3 of Fig. 5, have the same state binary codes. Thus one auxiliary state bit is appended to each node in the USCRSG. The auxiliary state bit appended to states 1 and 6 is '0' and to states 3 and 4 is '1'. The USCRSG for modulo-3 is shown in Fig. 5(a). Note that there are some other ways to form a USC. [3, 18]

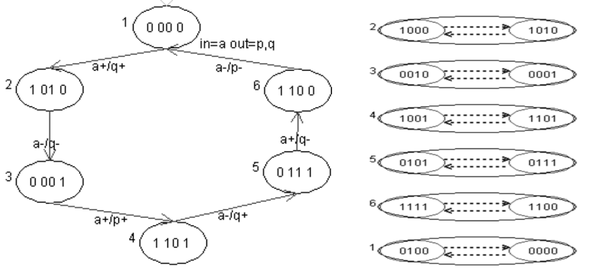


Figure 5: (a) The USCRSG and (b) STTG of modulo-3

3.2. Construct ST transformation graphs

For a USC reversible state graph (USCRSG) specifying a sequential circuit with m-bit input, n-bit output and e edges (i.e. e next-state functions), the corresponding self-timed transformation graph (STTG) is a hyper-graph which consists of e transformation sub-graphs (TSG) and each TSG consists of two nodes connecting to each other by two direct edges. If the nodes in a TSG have the same state binary code, such a sub-graph degenerates into a self-loop sub-graph. A USCRSG is transformed into a STTG which is the specification for constructing QBC.

Each edge in a STTG represents a transition from one state to another. For each edge e with source node i and target node j in a USCRSG, a corresponding transformation sub-graph (TSG) is formed for the target node j. The TSG consists of two new nodes *source* and *target* connecting to each other by two directed edges (i.e. *source* $\rightarrow$ *target*, *target* $\rightarrow$ *source*). These two direct edges are called *quantum links* and are marked in dashed lines.

The quantum states for the source node and target node in each TSG are reversible due to the quantum links while applying quantum operations.
The state binary codes of the *source* and *target* nodes are labeled with ib(j) + ob(i) + ab(i) and ib(j) + ob(j) + ab(j), respectively. The function ab(i) is the binary code function of auxiliary state bits which is similar to ib(i) and ob(i) in section 2.2.1.

For example, the USCRSG of modulo-3, shown in Fig. 5(a), can be transferred to the STTG, shown in Fig. 5(b). The STTG contains 6 TSGs. The dashed line connected two nodes in a TSG are the *quantum links*.

3.3. Perform state decomposition

To construct SQBCs based on Toffoli gates, the state binary codes of adjacent nodes in the TSGs must differ in only one bit [1]. This property can be retained by performing state decomposition.

If two nodes in a TSG have Hamming distances more than one, they have to be decomposed and some appropriate states are added between them so that any adjacent states differ only one bit. Furthermore, the added states must be never used in the STTG.

Taking the STTG of modulo-3, shown in Fig. 5(b), as an example, the states of the 3rd and 6th TSGs have to be decomposed. The possible state decompositions for the states, (0010, 0001), of the 3rd TSG are (0010, 0000, 0001) and (0010, 0011, 0001). The first transposition is illegal since the state 0000 is used in 1st TSG. In the same way, the possible state decompositions for the states, (1111, 1100), of the 6th TSG are (1111, 1101, 1100) and (1111, 1110, 1100). The first transposition is illegal since the state 1101 is used in the 4th TSG. Therefore, the state decomposition of (0010, 0001) and (1111, 1100) becomes (0010, 0011, 0001) and (1111, 1110, 1100), respectively. The complete STTG of modulo-3 is shown in Fig. 6.

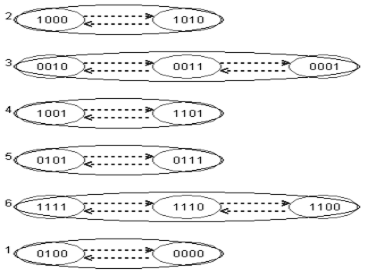


Figure 6: The decomposed STTG of modulo-3

3.4. Construct quantum Boolean circuits

Once a STTG is decomposed, the corresponding quantum circuit based on *Toffoli gates* can be constructed. Quantum wires (i.e. quantum operations) based on Toffoli gates can be generated by each state transposition of TSGs.

Taking the first TSG (labeled as 2) of the STTG, shown in Fig. 6, as an example, the states in the TSG are 1000 and 1010. S=1000$\wedge$1010=1000 so the 1st qubit uses the one-controlled-gate. Similarly, since R=$\neg$(1000$\vee$1010) =0101, the 2nd and 4th qubits use the zero-controlled-gate. Finally, I=1000$\oplus$1010=0010 so the 3rd qubit uses the target gate. Therefore a quantum operation can be formed by the above Toffoli gates in the QBC.

The SQBC of modulo-3 is constructed and shown in Fig.7, where the qubit *a* is the input, the qubits *Y* and *N* are the outputs and the qubit *s0* is the auxiliary qubit for assisting in the work of circuits.

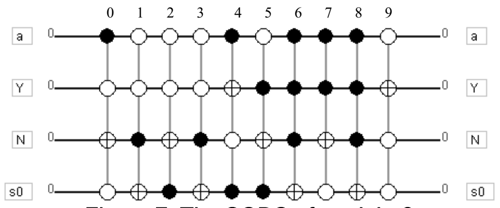


Figure 7: The SQBC of modulo-3

3.5. Optimize quantum Boolean circuits

The optimization of QBCs is to simplify and merge the Toffoli gates and wires in QBCs and thus reduce the complexity of circuits. Two reduction rules [1] are used to optimize QBCs:

(1) For any two quantum operations in a QBC, if they are identical then they can be removed from the QBC.

(2) For any two quantum operations in a QBC, if they are identical except one qubit in which one of the logic gates is one-controlled gate and the other is zero-controlled gate then the different logic gates can be removed and these two operations can be merged into one.

For example, the SQBC of the modulo-3, shown in Fig. 7, has 10 quantum operations (labeling from left to right with 0 to 9). Applying the above reduction rules, the 0th and 7th and the 2nd and 5th quantum operations can be merged. Thus the number of quantum operations and gates are optimized from 10 to 8 and 40 to 30, respectively. The optimized SQBC of the modulo-3 is shown in Fig. 8.

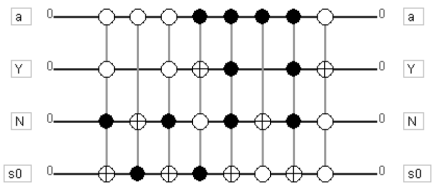


Figure 8: The optimized SQBC of modulo-3

4. Synthesis of Composable QBCs

In the classical domain, the large circuit can be constructed by several reusable IP components. Hence, if the synthesized SQBCs are reusable and composable like IPs, they can be exploited to construct a large and more complicated QBC rapidly.

Unfortunately, the SQBCs cannot be exploited immediately to construct QBCs as the circuit specifications will be violated by the reversible characteristic if SQBCs are composed together.

The composable problems and a new methodology for synthesizing composable QBCs (CQBCs) are proposed and described in the following sections.

4.1. Composibility problems of CQBCs

For classical circuits, an input change may cause some output and state changes. When a circuit is stable, the same input pattern reapplying again to the circuit can change neither output nor state signals.

This is not true in quantum circuits. Applying the same input pattern twice in quantum circuits may result in the different or wrong state as quantum operations are reversible and the states are always changed according to the matched patterns. This makes QBCs not reusable and large quantum Boolean circuits cannot be composed by basic QBCs like classical circuits.

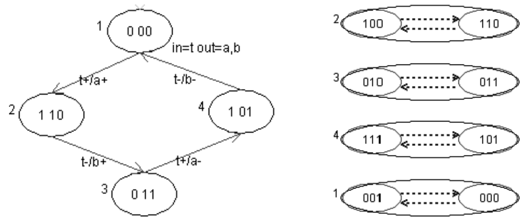


Figure 9: The (a) SG and (b) STTG of toggle

For example, a toggle [7, 10] has one input t and two outputs a and b. The circuit specification of a toggle is *Pref*(*t*?*a*!*t*?*b*!). The first transition on *t* triggers the output signal *a* and the second transition on *t* triggers the output signal *b*. Thus a toggle element can trigger two transition signals alternatively.

The state graph of the toggle is shown in Fig. 9(a). Initially the toggle is in the target node of state 1 with the state code “000.” If t is turned on then it goes to the target node of state 2 with the state code “110.” Now if t=1 is applied again then it goes to the source node of state 2 with state code “100.” The state sequence is 100, 110, 100, 110, … by applying the same input pattern more than once.

To make QBCs composable and reusable, only one extra auxiliary qubit added to QBCs is sufficient. The idea is to reset the auxiliary qubit before applying operations so that the quantum system will not go to the un-expected state.

In the Fig. 10(a), the state is changed between the source node (state code "100") and target node (state code "110") which is the expected state in this circuit specification. The expected state has to be hold while applying operations more than once.

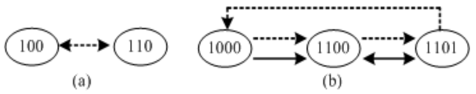


Figure 10: TSG (a) without and (b) with an auxiliary qubit

To solve the above problem, an extra auxiliary qubit has to be exploited. If an extra auxiliary qubit is added, an auxiliary state copied from the state of the target node (the auxiliary qubit is set to one) can be added into the TSG to keep the state staying in the expected one. In order to do so, the extra auxiliary qubit has to be reset before applying any operation.

The TSG with an extra auxiliary qubit is shown in Fig. 10(b). The dash line is the state transition without resetting the auxiliary qubit; the states are always changed between 1000 and 1101. The solid line is the modified state transition with resetting the auxiliary qubit; the state will be changed from 1000 to 1100 and then always from 1100 to 1101. As the pattern "1101" will be changed to 1100 by resetting the auxiliary qubit before applying this operation, the state transition "1101→1000" will become to "1100→1101". The new state sequence is: 1000, 1100, 1101, 1101, … and that follows the circuit behavior.

Therefore, the composable problem can be solved by adding only one extra auxiliary qubit.

4.2. Construction of CQBCs

To synthesize CQBCs, the composable STTGs are transferred from decomposed STTGs. An auxiliary qubit with initial value 0's is appended to each state binary code of all nodes in STTGs. An auxiliary node with sbc(the corresponding target node) is added for each TSG in a STTG and then set the auxiliary bit of the auxiliary node to 1. The new STTGs are called composable STTGs.

The composable STTGs contains two operation blocks: a core operation block representing the same circuit behavior of non-composable QBCs and a composable operation block preventing going to a wrong state if repeated operations are applied. The core operation block is made by the quantum operations constructed with the source and target nodes and the composable operation block is made by the quantum operations with the target and auxiliary nodes.

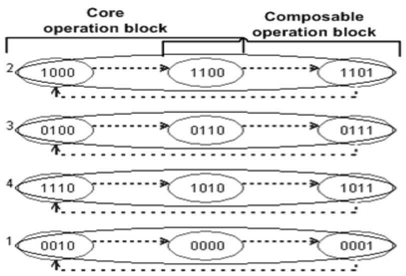


Figure 11: The composable STTG of toggle

The STTG of a toggle is transferred by the composable synthesis methodology proposed in the section 4.1 and the resulting composable STTG is shown in Fig. 11. The CSTTG consists of a core operation block STTG (STTG0) which is composed by the source and target nodes and a composable operation block STTG (STTG1) which is composed by the target and auxiliary nodes. A composable QBC is generated by cascading STTG1 with STTG0. The resulting CQBC of the toggle through optimizations is shown in Fig. 12 and the composable operation block and core operation block are from the 0th to 3rd and 4th to 5th quantum operations, respectively.

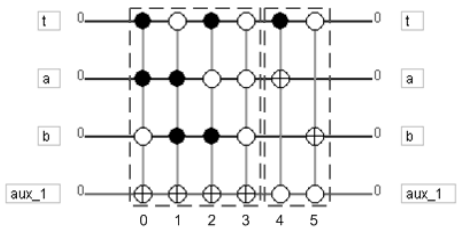


Figure 12: The CQBC of toggle

The composable operation block is used to prevent going to the wrong state when an input is applied to a QBC more than once. For the CQBC in Fig. 12, initially all input, output and auxiliary qubits are set to zero. Suppose the input t is turned on, the system will go to the state with state binary code "1101." If "1101" is applied again, the system will go to the state with state binary code "1000." Now if the input t is turned off, the system ends up with the state with state binary code "0001" which is a wrong state. Since the auxiliary qubit is to prevent the system going to the wrong state, the correct operation is to reset the auxiliary qubits when applying any input pattern to the system. Thus the correct input for the system is "1100" (by resetting the fourth qubit) and the system stays in the state with state binary code "1101."

4.3. Composition of composable QBCs

Once CQBCs are synthesized by the above methodology they can be exploited to construct a larger quantum Boolean circuit rapidly like classic circuits.

For example, the circuit implementation of classic mod-4 counter can be directly composed by two toggle elements, shown in Fig. 13(a). The quantum version of mod-4 counter can be constructed similarly, shown in Fig. 13(b).

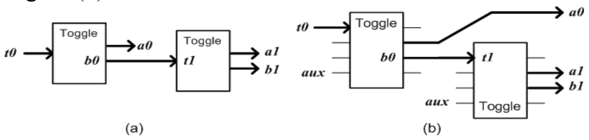


Figure 13: The (a) classical and (b) quantum circuit implementation of mod-4 counter

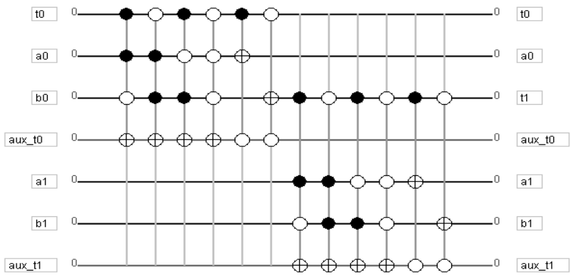


Figure 14: The QBC of mod-4 counter

Figure 14 shows the QBC of mod-4 counter. If the output (i.e. *b0*) of the first toggle element connects to the input (i.e. *t1*) of the second toggle element, they share the same qubit (i.e. *b0*/*t1*) in the QBC. The QBC of the mod-4 counter has 7 qubits, 12 quantum operations and 44 logic gates. Furthermore, mod-4 counter is composable since the toggle element is composable.

5. Experimental results

A set of self-timed components [14] is used to test QCAD and the circuit specifications of these self-timed components are shown in Table 1. These components can be used as building blocks to compose control-path components of self-timed systems.

The synthesis results are shown in Table 1. Columns 2 and 3 (i.e. #*N* and #*E*) are the numbers of nodes (states) and edges (transitions) of SGs, respectively. Column 4 shows the number of total qubits required for the QBC containing input, output and auxiliary state qubits. Column 5 and 6 show the number of quantum operations and logic gates for the synthesized QBC and the optimized QBC, respectively. The modulo-3 example is also shown in the row 6 of Table 1.

The results of Table 1 show that the optimization algorithm can significantly reduce the numbers of quantum operations and logic gates for the self-timed components. And Table 2 shows the results of composable QBC synthesis.

Table 1: The result of SQBC synthesis

Circuit	*#N*	*#E*	*Qubits*			*Quantum Operations*		*Logic gates*	
			I	*O*	*aux*	*org.*	*opt.*	*org.*	*opt.*
Fork	2	2	1	1	0	2	1	4	1
Merge	4	8	2	1	0	4	1	12	1
Join	6	12	2	1	1	2	2	8	8
Toggle	4	4	1	2	0	4	2	12	4
Modulo-3	6	6	1	2	1	10	8	40	30
If-Else	16	20	5	4	0	16	16	144	144
Call	12	16	*Non-synthesizable*						
2P-4P	6	6	2	2	1	10	10	50	50
4P-2P	6	6	2	2	1	10	10	50	50

Table 2: The result of CQBC synthesis

Circuit	*Qubits*			*Quantum Operations*		*Logic gates*	
	I	*O*	*aux*	*org.*	*opt.*	*org.*	*opt.*
Fork	1	1	1	4	3	12	8
Merge	2	1	1	8	5	32	18
Join	2	1	2	4	4	20	20
Toggle	1	2	1	8	6	32	22
Modulo-3	1	2	2	14	10	70	46
Modulo-4	1	2	2	12	12	44	44
If-Else	5	4	1	32	32	320	320
Call	*Non-synthesizable*						
2P-4P Convertor	2	2	2	32	192	32	192
4P-2P Convertor	2	2	2	32	192	32	192

Since CQBCs have an additional composable operation block, the number of quantum operations of CQBCs is much larger than the non reusable QBCs. This is because the optimization algorithm can not reduce any operations or logic gates in the composable operation block.

6. Conclusions and future works

This paper presents a novel methodology to transfer self-timed circuit specifications into sequential quantum Boolean circuits (SQBCs) and Composable SQBCs (CQBCs). State graphs (SGs) used for self-timed system design are exploited to describe the behaviors of circuits and then are automatically translated into SQBCs based on Toffoli gates.

The concept of IP reuse is also applied to the constructed SQBCs to produce reusable and composable quantum Boolean circuits (CQBCs). These reusable CQBCs as building blocks can be exploited to construct more complicated quantum Boolean circuits.

To the best of our knowledge there are no related works on synthesizing sequential circuit behaviors into quantum Boolean circuits.

A set of self-timed components is successfully synthesized into CQBCs by our methodology. These CQBCs can be used as building blocks to compose control-path components of self-timed systems.

Our future work will focus on synthesizing data-path components and translating Quantum algorithms into QBCs.

7. References

[1] Tsai, I.-M.; Kuo, S.-Y.; Quantum Boolean Circuit Construction and Layout under Locality Constraint, *IEEE-NANO 2001. Proceedings of the 2001 1st IEEE Conference on ,* 28-30 Oct. 2001 Pages:111 – 116

[2] Wei-Min Zhang, *Introduction to Quantum Information Processing*, Lecture notes in the 2003 Symposium on Digital Life and Internet Technologies

[3] Pastor, E.; Cortadella, J.; An efficient unique state coding algorithm for signal transition graphs, *Computer Design: VLSI in Computers and Processors, 1993. ICCD '93. Proceedings., 1993 IEEE International Conference on* 3-6 Oct. 1993 Page(s):174 - 177

[4] Al Davis and Steven M. Nowick. An introduction to asynchronous circuit design. *The Encyclopedia of Computer Science and Technology,* vol. 38, 1996

[5] David L. Dill, Trace theory for automatic hierarchy verification of speed-independent circuits. Pages 51-65, 1987

[6] Iwama, K.; Kambayashi, Y.; Yamashita, S.; Transformation Rules for Designing CNOT-based Quantum Circuits, *Design Automation Conference, 2002. Proceedings. 39th*, 10-14 June 2002 Pages:419 - 424

[7] Priyadarsan Patra, Donald Fussell, Building-blocks for Designing DI Circuits, *Technical report tr93-23*, Dept. of Computer Sciences, The Univ. of Texas at Austin, November 1993

[8] M. A. Nielsen and I. L. Chung. *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000

[9] Chris J. Myers, *Asynchronous Circuit Design*, Wiley & Sons, Inc., 2001

[10] Jens Sparso; Steve Furber, *Principle of Asynchronous Circuit Design*, Kluwer Academic Publishers, 2001

[11] Julian Brown. *Minds, Machines, and the Multiverse*. Simon & Schuster, 2000

[12] Arthur O. Pitternger, *An Introduction to Quantum Computing Algorithm*. Birkjauser, 1998

[13] Michael Brooks, *Quantum Computing and Communications*. Springer, 1998.

[14] J.C. Ebergen, *Translating programs into delay-insensitive circuits*. Stichting Mathematisch Centrum, Amsterdam, 1989.

[15] Miller, D.M.; Maslov, D.; Dueck, G.W.; A Transformation Based Algorithm for Reversible Logic Synthesis, *Design Automation Conference, 2003. Proceedings, 2-6 June 2003* Pages: 318 – 323

[16] S. H. Unger. *Asynchronous Sequential Switching Circuits*. Wiley-Interscience, New York, 1969.

[17] Ahmed Younes, Julian Miller, Automated Method for Building CNOT Based Quantum Circuits for Boolean Functions, *arXiv:quantum-ph/0304099 14 Apr 2003*

[18] Meng-Lin Yu; A new approach for checking the unique state coding property of signal transition graphs, Subrahmanyam, P.A.; *Design Automation, 1992. Proceedings. [3rd] European Conference on* 16-19 March 1992 Page(s):312 – 321