

Optimal Synthesis of Multiple Output Boolean Functions Using a Set of Quantum Gates by Symbolic Reachability Analysis

William N. N. Hung, Xiaoyu Song, Guowu Yang, Jin Yang, and Marek Perkowski

Abstract—This paper proposes an approach to optimally synthesize quantum circuits by symbolic reachability analysis, where the primary inputs and outputs are basis binary and the internal signals can be nonbinary in a multiple-valued domain. The authors present an optimal synthesis method to minimize quantum cost and some speedup methods with nonoptimal quantum cost. The methods here are applicable to small reversible functions. Unlike previous works that use permutative reversible gates, a lower level library that includes nonpermutative quantum gates is used here. The proposed approach obtains the minimum cost quantum circuits for Miller gate, half adder, and full adder, which are better than previous results. This cost is minimum for any circuit using the set of quantum gates in this paper, where the control qubit of 2-qubit gates is always basis binary. In addition, the minimum quantum cost in the same manner for Fredkin, Peres, and Toffoli gates is proven. The method can also find the best conversion from an irreversible function to a reversible circuit as a byproduct of the generality of its formulation, thus synthesizing in principle arbitrary multi-output Boolean functions with quantum gate library. This paper constitutes the first successful experience of applying formal methods and satisfiability to quantum logic synthesis.

Index Terms—Formal verification, logic synthesis, model checking, quantum computing, reversible logic, satisfiability.

I. INTRODUCTION

REVERSIBLE logic [1] plays an important role in the synthesis of quantum computing circuits [2], [3]. The synthesis of reversible logic circuits using elementary quantum gates [4], [5] is different from classical (nonreversible) logic synthesis. There are some works [6]–[9] on reversible logic synthesis using basic reversible gates (Toffoli, Fredkin [10], or Feynman gates). However, these reversible logic gates have different quantum implementation costs (e.g., the cost of Feynman is lower than Toffoli). Therefore, finding the smallest number of gates to synthesize a reversible circuit does not necessarily result in quantum implementation with the lowest cost (in terms of quantum gates).

In this paper, we focus on synthesizing reversible circuits using quantum primitives with the lowest total cost using a library of basic 2-qubit quantum gates, which will be described in Section III. Our synthesis method can also be modified to use other libraries of gates. We chose a library of basic 2-qubit quantum gates in this paper as they allow us to better evaluate the quantum implementation costs. The circuits we synthesized include common reversible gates that can next be used at higher levels of logic synthesis. Our approach can also be used as an equivalent of “technology mapping” for quantum circuits.

We reduce the quantum logic synthesis problem to multiple-valued logic synthesis; this reduction simplifies the search space and reduces the algorithm complexity. We formulate the above quantum logic synthesis task via symbolic reachability analysis [11], [12]. We used satisfiability-based model checking to solve the problem, but other decision methods or combinatorial optimization techniques can be similarly applied here. Our method not only guarantees to find a quantum implementation (for reversible circuits) but also guarantees the lowest quantum cost in the synthesized result (for the set of circuits where the control qubit of our 2-qubit gates is always basis binary). We also introduce an automated way of adding ancilla qubits and finding their appropriate constant values in the synthesis process. Thus, even irreversible circuits can be converted to reversible circuits that in turn are synthesized by our method. In contrast to previous works, which either use permutative reversible gates to design permutative circuits or universal quantum gates to design quantum circuits, we use a subset of quantum gates to design permutative circuits.

II. BACKGROUND

Given a function f , we say f is reversible if and only if there exists a function g such that $x = g(f(x))$ for all x in the domain of f . The corresponding function g (as described above) is usually referred to as f^{-1} . Given n Boolean inputs, any multiple-output Boolean function on such n Boolean inputs must have exactly n Boolean outputs so that it is reversible [2]. We use $n \times n$ to denote a reversible function with n Boolean inputs and n Boolean outputs. Given an $n \times n$ reversible function f , there are 2^n input rows and 2^n output rows in the truth table of f . The output rows must be a permutation of the input rows in the truth table of f .

In quantum computing [2], the fundamental information unit is a qubit. The state of a qubit is a superposition of 0 and 1

Manuscript received November 8, 2004; revised February 22, 2005 and June 8, 2005. This paper was recommended by Associate Editor J. H. Kukula.

W. N. N. Hung is with Synplicity Inc., Sunnyvale, CA 94086 USA (e-mail: william_hung@alumni.utexas.net).

X. Song, G. Yang, and M. Perkowski are with Portland State University, Portland, OR 97207 USA.

J. Yang is with Strategic CAD Labs, Intel Corporation, Hillsboro, OR 97124 USA.

Digital Object Identifier 10.1109/TCAD.2005.858352

85 states, also denoted as $|0\rangle$ and $|1\rangle$, respectively. The qubit state
86 q can be represented by

$$q = \alpha|0\rangle + \beta|1\rangle$$

87 where α and β are both complex numbers and $|\alpha|^2 + |\beta|^2 = 1$.
88 The classical state of binary 0 corresponds to the case where
89 $\alpha = 1$ and $\beta = 0$. Similarly, the classical state of binary 1 cor-
90 responds to $\alpha = 0$ and $\beta = 1$. We refer to them as basis binary
91 0 and basis binary 1, respectively. All other combinations of α
92 and β are not basis binary. The quantum state of a single qubit
93 is usually denoted by the vector

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

94 Given the state of each qubit, the overall quantum state is a
95 Kronecker product of the states of each qubit. Take two qubits
96 for example

$$\begin{pmatrix} u_0 \\ u_1 \end{pmatrix} \otimes \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} = \begin{pmatrix} u_0 \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} \\ u_1 \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} u_0 v_0 \\ u_0 v_1 \\ u_1 v_0 \\ u_1 v_1 \end{pmatrix}. \quad (1)$$

97 Notice that if the individual qubits are basis binary, then the
98 Kronecker product is simply an enumeration of all the possible
99 binary values (truth table) of its qubits. If we can use the
100 quantum state of multiple qubits to determine the individual
101 state of each qubit (such as the above case), we call it a
102 separable state. There are some cases where the quantum state
103 cannot be separated into individual states of each qubit, i.e.,
104 we cannot describe (mathematically) the state of each qubit but
105 we can describe the quantum state of all the qubits combined.
106 We call such states entangled states. This idea of entangled
107 state is called quantum entanglement, and it originated from
108 the Einstein–Podolsky–Rosen paradox [13].

109 The effect of quantum gates on a quantum state can be
110 described as vector operations, where the quantum gates are
111 represented by unitary matrices. A unitary matrix is a $n \times n$
112 complex matrix M with the property

$$M \times M^+ = M^+ \times M = I$$

113 where I is the identity matrix and M^+ is the conjugate trans-
114 pose (also known as the Hermitian adjoint) of M .

115 Given an n -qubit quantum gate G , we call G a permutative
116 quantum gate if and only if the outputs of G are all basis binary
117 when its inputs are all basis binary, i.e., G is a permutative
118 quantum gate if and only if G implements an $n \times n$ Boolean
119 reversible function (when its inputs are basis binary).

120 A generalized 2-qubit controlled U gate [5] is shown in
121 Fig. 1. Its unitary matrix is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{pmatrix}$$

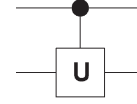


Fig. 1. Controlled- U gate.

where the four entries in the right bottom also form a (single
qubit) unitary matrix U by itself

$$U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}.$$

It has been shown [4], [5] that permutative quantum logic
circuits can be constructed using elementary quantum, XOR,
controlled- V , controlled- V^+ , or NOT gates, as shown in Fig. 2.
The NOT gate is also called an inverter. Its unitary matrix is

$$M_{\text{NOT}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Quantum XOR gates are also called Feynman gates or
controlled-NOT (CNOT) gates. The controlled- V gate's data
output is the same as its data input (B) when its control input
(A) value is 0 (FALSE). When its control value is 1 (TRUE),
the data output becomes V (input) [2]

$$V = \frac{1+i}{2} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}, \quad V^+ = \frac{1-i}{2} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}.$$

Similar rules apply to the controlled- V^+ gate, except that its
data output becomes V^+ (input), where V^+ is the Hermitian of
 V , i.e.,

$$\frac{1+i}{2} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix} \times \frac{1-i}{2} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

The quantum XOR (controlled-NOT), controlled- V , and
controlled- V^+ are all special cases of the generalized
controlled- U gate, where the matrix U corresponds to M_{NOT} ,
 V , and V^+ , respectively.

According to [2], the values V and V^+ are constructed
such that they are the square root of NOT (i.e., inverter gate):
 $V \times V = V^+ \times V^+ = M_{\text{NOT}}$. Hence, if the signal V (input)
is passed through another controlled- V gate with its control
value also equal to 1 (TRUE), the output of the second gate
becomes the NOT of the input.

The quantum XOR, controlled- V , and controlled- V^+ gates
are 2×2 gates. They are also called 2-qubit gates. Similarly, the
NOT gate (inverter) is a 1-qubit gate. For quantum implementa-
tion, the cost of 2-qubit gates far exceeds the cost of 1-qubit
gates. Hence, in a first approximation, the quantum cost of
1-qubit gates is usually ignored in the presence of 2-qubit
implementations [5], [14].

In this paper, we adopt the quantum gate cost evalu-
ation introduced in [4]. According to the method in [4],
each of the 2-qubit gates (quantum XOR, controlled- V ,
controlled- V^+) has a quantum implementation cost of 1.
In addition, when both quantum XOR and controlled- V (or
controlled- V^+) are operating on the same two qubits in a 160

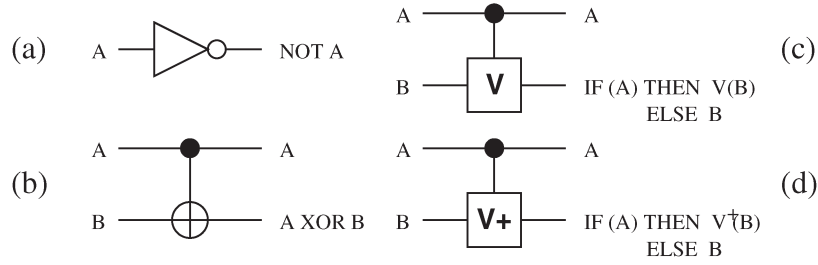


Fig. 2. Elementary quantum logic gates.

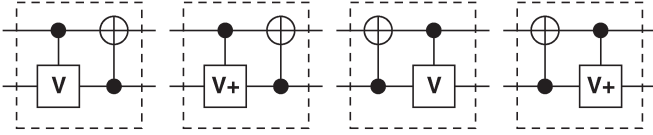


Fig. 3. Merged 2-qubit gates.

161 symmetric pattern (shown in Fig. 3), their total cost is consid-
 162 ered as 1 as well. A more accurate cost function can be created
 163 for a particular quantum technology such as nuclear magnetic
 164 resonance (NMR) [15], but for simplicity and comparison to
 165 previous work we will use here the cost function from [4].

166 Given a reversible function, the quantum logic synthesis task
 167 considered in this paper is to synthesize the function using
 168 the above elementary quantum logic gates with the minimum
 169 cost. Various heuristic methods have been applied to find low-
 170 cost quantum implementations (using the elementary gates)
 171 for the functionality of the Fredkin [4], Toffoli [16], and
 172 Peres [17] gates. Yet, nobody has been able to prove that they
 173 have the lowest quantum cost implementation (based on the
 174 cost evaluation criteria given above).

175 We can perform the above quantum logic synthesis task
 176 through reachability analysis. Symbolic reachability analysis is
 177 a well-known technique in formal verification [11]. Its basic
 178 idea is to find all the reachable states of a finite state machine
 179 (FSM). Using symbolic representation, we can check if an
 180 invariant (property) is true for all reachable states. This tech-
 181 nique is used in invariant checking [11], where the state space
 182 is traversed exhaustively against an invariant. Since the state
 183 space tends to be large for practical systems, recent symbolic
 184 reachability analysis techniques use various methods, such
 185 as binary decision diagram (BDD) [18], [19] or satisfiability
 186 (SAT), to avoid enumerating every system state while preserv-
 187 ing the completeness of the reachability analysis. We use state-
 188 of-the-art SAT-based bounded model checking [12] to check
 189 invariants. If the invariant is false, it can automatically generate
 190 a counter-example. We can find the shortest counter-example
 191 in this way by starting with a zero bound and gradually incre-
 192 menting the bound. If the invariant is true and given enough
 193 time, this method can also check that the bound is sufficiently
 194 large and establish the proof. SAT-based model checking has
 195 been successfully deployed in the industry [20]–[22].

196 III. SYMBOLIC FORMULATION

197 We consider each “quantum wire” of the quantum circuit as
 198 a superposition of $|1\rangle$ and $|0\rangle$, denoted as 1 and 0, respectively.
 199 We are interested in synthesizing quantum circuits with basis

binary inputs (1 and 0). The values of these signals are modified 200
 after passing through elementary gates (Fig. 2). There are six 201
 possible output values when we apply binary (1 and 0) inputs 202
 to one of those elementary gates: 0, 1, V_0 , V_1 , V_0^+ , V_1^+ , where 203
 V_0 represents $V(\text{input})$ when the input is 0, and similarly for 204
 V_1 , V_0^+ , V_1^+ , i.e., 205

$$V_0 = \frac{1+i}{2} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1+i}{2} \begin{pmatrix} 1 \\ -i \end{pmatrix}$$

$$V_1 = \frac{1+i}{2} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1+i}{2} \begin{pmatrix} -i \\ 1 \end{pmatrix}$$

$$V_0^+ = \frac{1-i}{2} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1-i}{2} \begin{pmatrix} 1 \\ i \end{pmatrix}$$

$$V_1^+ = \frac{1-i}{2} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1-i}{2} \begin{pmatrix} i \\ 1 \end{pmatrix}.$$

These six possible values are used as input values to gates 206
 in subsequent stages. We want to synthesize our circuit 207
 such that the “control” input of controlled-NOT (quantum 208
 XOR), controlled- V , or controlled- V^+ is always basis binary 209
 (0s and 1s), i.e., their input values cannot be V_0 or V_1 , etc. 210

We impose the above restriction because a nonbinary value 211
 at the control input of the controlled-NOT, controlled- V , or 212
 controlled- V^+ gate can generate an entangled quantum state. 213
 For example, if we have V_0 at both control and data inputs 214
 of the controlled- V gate, the unitary matrix multiplied by the 215
 Kronecker product (of the inputs) becomes 216

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} \\ 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix} \times \begin{pmatrix} 0.5i \\ 0.5 \\ 0.5 \\ -0.5i \end{pmatrix} = \begin{pmatrix} 0.5i \\ 0.5 \\ 0 \\ 0.5 - 0.5i \end{pmatrix}.$$

The vector result cannot be separated into two individual qubit 217
 states using (1). The u_1u_0 entry from (1) is 0, which requires 218
 $u_1 = u_0 = 0$. It contradicts with the other entries of the vector. 219
 This is an entangled quantum state. Similar scenarios exist for 220
 controlled- V^+ . The controlled-NOT also has similar examples 221
 [23]. For the rest of this paper, we focus on synthesizing quan- 222
 tum circuits using our set of quantum gates (NOT, controlled- 223
 NOT, controlled- V , and controlled- V^+), where the control 224
 input of the 2-qubit gates is always basis binary. However, the 225
 same approach can be used to synthesize circuits using other 226
 libraries of quantum gates as long as it can be reduced to a 227
 multiple-valued logic problem. 228

229 Based on the unitary matrices in Section II, we can see that
 230 if the input of the NOT gate is not basis binary, namely $V_0, V_1,$
 231 $V_0^+,$ or $V_1^+,$ its corresponding output is $V_1, V_0, V_1^+,$ or $V_0^+,$
 232 respectively. Given a basis binary 1 on the control input of the
 233 controlled-NOT gate, the data input and the data output exhibit
 234 the same property (above) as the NOT gate. Also, as shown in
 235 Section II, given the six possible values (0, 1, V_0, V_1, V_0^+ or
 236 V_1^+) at the data input of the controlled- V or controlled- $V^+,$
 237 their corresponding data output has the same set of six possible
 238 values. Hence, the input/output of every quantum gate in the
 239 circuit can be represented using the above six values.

240 If we look at the complex matrix representation of $V_0, V_1,$
 241 $V_0^+,$ and $V_1^+,$ we can deduce that $V_0 = V_1^+$

$$V_0 = \frac{1+i}{2} \begin{pmatrix} 1 \\ -i \end{pmatrix} = \begin{pmatrix} 0.5 + 0.5i \\ 0.5 - 0.5i \end{pmatrix}$$

$$V_1^+ = \frac{1-i}{2} \begin{pmatrix} i \\ 1 \end{pmatrix} = \begin{pmatrix} 0.5 + 0.5i \\ 0.5 - 0.5i \end{pmatrix}$$

242 and $V_1 = V_0^+$

$$V_1 = \frac{1+i}{2} \begin{pmatrix} -i \\ 1 \end{pmatrix} = \begin{pmatrix} 0.5 - 0.5i \\ 0.5 + 0.5i \end{pmatrix}$$

$$V_0^+ = \frac{1-i}{2} \begin{pmatrix} 1 \\ i \end{pmatrix} = \begin{pmatrix} 0.5 - 0.5i \\ 0.5 + 0.5i \end{pmatrix}.$$

243 Thus, it suffices to represent signals in the circuit using four
 244 values: 0, 1, $V_0, V_1.$ In this way, we reduce the problem of
 245 quantum circuit synthesis (which would normally use unitary
 246 matrices and Hilbert space to represent signals) to a simpler
 247 synthesis problem in mixed binary/quaternary algebra. This
 248 is a general approach to efficiently synthesize a subclass of
 249 quantum circuits. It can be applied to gates other than the
 250 2-qubit gates introduced above.

251 *Theorem 1:* For any deterministic quantum circuit (with n
 252 qubits, $n > 0$) that produces basis binary outputs for basis
 253 binary inputs, its unitary matrix is canonical, i.e., there is only
 254 one unitary matrix that represents the function of this circuit.
 255 This is a permutation matrix.

256 *Proof:* We prove the theorem in four steps. Step 1): There
 257 are 2^{2^n} distinct $n \times n$ binary reversible logic functions. Step 2):
 258 When all n qubits are basis binary, their Kronecker product has
 259 one entry equal to 1 while all the other entries are equal to 0.
 260 Step 3): Each row or column of the unitary matrix should have
 261 only one entry equal to 1 while all the other entries are equal
 262 to 0. Step 4): The unitary matrix must be unique under the above
 263 circumstances.

264 Step 1) The function of this quantum circuit is a binary
 265 reversible logic function. The output entries in the
 266 truth table are permutations of the input entries
 267 for this function. The truth table has 2^n rows, i.e.,
 268 2^n distinct binary input entries (and corresponding
 269 output entries). Since the output entries are permu-
 270 tations of the 2^n input entries, there are 2^{2^n} ways to
 271 permute them. Hence, there are 2^{2^n} distinct $n \times n$
 272 binary reversible logic functions.

Step 2) The Kronecker product of n qubits is

273

$$\begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} \alpha_n \\ \beta_n \end{pmatrix} = \begin{pmatrix} \alpha_1 \alpha_2, \dots, \alpha_{n-1} \alpha_n \\ \alpha_1 \alpha_2, \dots, \alpha_{n-1} \beta_n \\ \vdots \\ \beta_1 \beta_2, \dots, \beta_{n-1} \beta_n \end{pmatrix}.$$

For each qubit, $\alpha\beta$ have only two choices (10 or 01)
 274 to be basis binary. There are 2^n distinct ways for all
 275 n qubits to be basis binary. Under this circumstance,
 276 the above Kronecker product is an enumeration of
 277 the truth table patterns for α and β of each qubit.
 278 Hence, there is one entry in the Kronecker product
 279 equal to 1 while all the other entries are equal to 0.

Step 3) Let U be a unitary matrix of the n -qubit circuit.
 281 There are 2^n rows and 2^n columns in $U.$ Let P and
 282 Q be the Kronecker product of the input and output
 283 for this circuit, respectively. We have

$$U \times P = Q. \quad (2)$$

According to Step 2), the vector P has one entry
 285 equal to 1 and all the other entries are 0. Similarly,
 286 the vector Q has one entry equal to 1 and all the
 287 other entries are 0. We use u_{ij} to denote the value of
 288 matrix U in the i th row and j th column, and p_i and
 289 q_i to denote the value of vector P and Q in the i th
 290 row, respectively.

Given $0 \leq i \leq 2^n,$ suppose all the entries in the i th
 292 row of U are 0, then q_i will be 0 for all possible
 293 values of P due to (2). This is a contradiction
 294 because Q has 2^n rows and 2^n distinct values, so
 295 q_i must be 1 for one of those cases. Hence, any row
 296 of U cannot be all zeros.

Furthermore, suppose there are more than one en-
 298 try that is nonzero (say columns u_{ij} and u_{ik} are
 299 nonzero), then we can have two distinct patterns
 300 of $P,$ one with $p_j = 1$ and the other with $p_k = 1,$
 301 both being able to produce a nonzero $q_i.$ Again, this
 302 is a contradiction because we can only have one
 303 possibility for q_i to be nonzero. Hence, every row
 304 of U must have exactly one nonzero entry. In order
 305 to produce a corresponding 1 in the vector $Q,$ the
 306 nonzero entry in U must be 1.

Lastly, suppose we have $u_{ij} = 1$ and $u_{kj} = 1,$ both
 308 in the j th column. We can pick a valuation of P with
 309 $p_j = 1.$ The corresponding vector Q will have $q_i =$
 310 1 and $q_k = 1.$ This is again a contradiction since
 311 only one row of vector Q can be nonzero. Thus,
 312 every column of U must have exactly one nonzero
 313 entry (which must be 1).

Step 4) There are 2^{2^n} possibilities for U to satisfy the prop-
 315 erty in Step 3), which is exactly the number of
 316 distinct permutations. Hence, to each permutation
 317 corresponds a unique unitary matrix $U.$ This com-
 318 pletes the Proof of Theorem 1. ■

The importance of the above theorem is that once we have
 320 specified the basis binary input/output behavior of the quantum
 321

322 circuit, there is only one unitary matrix that can satisfy the
 323 specification (because it is canonical). Hence, the functional
 324 behavior of the synthesized quantum circuit, under nonbinary
 325 (complex number) input/outputs, would be deterministic, even
 326 though they were not in the original specification. This idea
 327 is especially important for the synthesis of binary reversible
 328 functions (Toffoli, Fredkin, etc.) using quantum gates. It suf-
 329 fices to specify the basis binary input/output behavior of the
 330 reversible function, and the synthesized quantum circuit would
 331 have identical behavior as those of classical quantum circuits
 332 for all quantum values.

333 Suppose we intend to synthesize an $n \times n$ reversible function
 334 R specified by its truth table with n input columns, n output
 335 columns, and 2^n rows corresponding to n output patterns using
 336 the 2-qubit quantum gates [Fig. 2(b)–(d)] described above. The
 337 synthesized result should be a cascade of L stages. Each stage
 338 consists of one of the above quantum gates. Since the function
 339 applies to n qubits and the quantum gates at each stage are
 340 1-qubit or 2-qubit gates, the synthesized result should indicate
 341 to which qubits the gates are connected. For each stage i , we
 342 use g_i to represent the gate selection variable [Fig. 2(b)–(d)],
 343 and we use A_i and B_i to indicate the two qubits that the
 344 gate is connected to, i.e., $A_i, B_i \in \{1, \dots, n\}$. As a naming
 345 convention, we refer to the qubit indicated by A_i [the upper
 346 qubit in Fig. 2(b)–(d)] as the control qubit, and we refer to the
 347 qubit indicated by B_i [the lower qubit in Fig. 2(b)–(d)] as the
 348 data qubit. Since the two qubits must be different, we have

$$A_i \neq B_i. \quad (3)$$

349 We denote the inputs of stage i as \vec{U}_i , where $\vec{U}_i =$
 350 $u_{1i}u_{2i}, \dots, u_{ni}$. Each qubit (u_{qi} , $q = 1, \dots, n$) of the stage i
 351 can have four possible values (0, 1, V_0 , V_1). The output of stage
 352 i is denoted by \vec{U}_{i+1} , i.e.,

$$u_{q(i+1)} = \begin{cases} u_{A_i i} \oplus_Q u_{q i}, & (q = B_i) \wedge (g = \text{Fig. 1(b)}) \\ V(u_{q i}), & (q = B_i) \wedge (g = \text{Fig. 1(c)}) \wedge u_{A_i i} \\ V^+(u_{q i}), & (q = B_i) \wedge (g = \text{Fig. 1(d)}) \wedge u_{A_i i} \\ u_{q i}, & \text{otherwise.} \end{cases}$$

353 Note that we use \oplus_Q to denote the quantum XOR operation.
 354 Due to our restriction on the control input, the values V_0 and
 355 V_1 cannot be applied to the control input of controlled-NOT,
 356 controlled- V , or controlled- V^+ gates. We create a Boolean
 357 signal E_i to represent whether the gate has been erroneously
 358 configured (misconfigured) with the V_0 or V_1 values in the
 359 current (i th) synthesis stage or any previous synthesis stages. At
 360 the initial stage, there is no misconfiguration, and we initialize
 361 by setting

$$E_0 = 0.$$

362 As we move to subsequent stages, the E_{i+1} value (in stage
 363 $i + 1$) is 1 if either of the following two cases is true.

- 364 1) E_i (in the previous stage) is already 1.
- 365 2) The value of the control qubit $u_{A_i i}$ is not binary (where
 366 A_i is the control qubit).

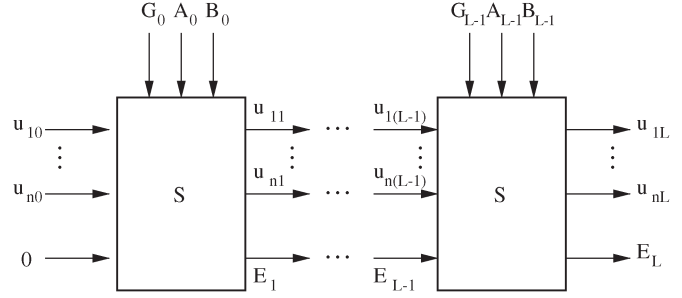


Fig. 4. L-2Syn problem.

Thus

$$E_{i+1} = E_i \vee (u_{A_i i} \notin \{0, 1\}).$$

So far, g_i had only three possible values [Fig. 2(b)–(d)]. To 368
 369 better reflect the quantum implementation cost, let us use a 369
 370 different gate selection variable G_i with seven possible values. 370
 371 G_i has all the three possible values of g_i , with four additional 371
 372 values to reflect the quantum XOR gate merged with controlled-
 373 V and controlled- V^+ gates (Fig. 3). We define the synthesis 373
 374 function S as

$$(\vec{U}_{i+1}, E_{i+1}) = S(G_i, A_i, B_i, \vec{U}_i, E_i). \quad (4)$$

Definition 1 (L-2Syn): The quantum logic synthesis problem 375
 376 for the reversible function R using 2-qubit gates as a cascade 376
 377 of L stages is to find a set of satisfying values to G_i, A_i, B_i 377
 378 (where $A_i \neq B_i$ and $i = 0, 1, \dots, L - 1$) such that $E_0 = E_L =$ 378
 379 0 and $\vec{U}_L = R(\vec{U}_0)$ for all possible Boolean input values of 379
 380 \vec{U}_0 . Mathematically speaking, a solution to the L-2Syn problem 380
 381 exists if and only if

$$\exists G_0 \exists A_0 \exists B_0, \dots, \exists G_{L-1} \exists A_{L-1} \exists B_{L-1} \cdot \left(\forall \vec{U}_0 \in \{0, 1\}^n \right. \\ \left. \cdot (E_0 = E_L = 0) \wedge (\vec{U}_L = R(\vec{U}_0)) \right) \wedge \left(\bigwedge_{i=0}^{L-1} A_i \neq B_i \right) \quad (5)$$

where $G_0 A_0 B_0, G_1 A_1 B_1, \dots, G_{L-1} A_{L-1} B_{L-1}$ form a solu- 382
 383 tion to the L-2Syn problem.

Fig. 4 illustrates the L-2Syn problem. Notice that we are 384
 385 performing $n \times n$ reversible logic synthesis here. E_0 is not 385
 386 an input constant to the reversible logic circuit because all the 386
 387 reversible gates use only qubits $1, \dots, n$. The E_i ($i = 0, \dots, n$) 387
 388 Boolean values are used to keep track of prohibited logic values, 388
 389 they are not a part of the reversible circuit.

Definition 2 (min-2Syn): The minimum length quantum 390
 391 logic synthesis problem for the reversible function R using 391
 392 2-qubit gates (quantum XOR, controlled- V , controlled- V^+ , or 392
 393 their merged versions) is to solve L-2Syn with the smallest 393
 394 possible number L .

Theorem 2: For any reversible function R that does not 395
 396 require inverters in its quantum implementation, finding its 396
 397 quantum logic implementation with the minimum cost is equiv- 397
 398 alent to solving the min-2Syn for R .

Proof: The min-2Syn solution consists of the smallest 399
 400 possible L stages where each stage has a quantum cost of 1. 400
 401 Thus, the minimum quantum cost is L . ■

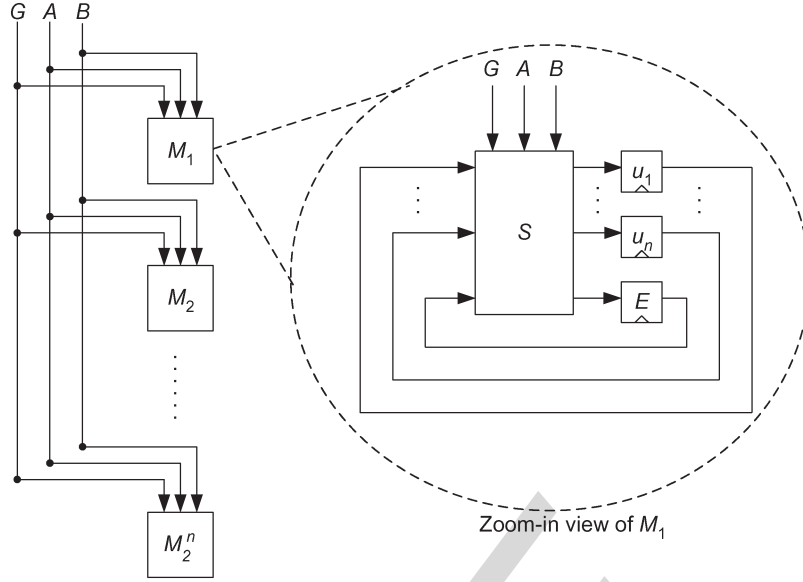


Fig. 5. FSM for reachability analysis.

402 So far, we have not considered inverters (1-qubit gates).
 403 Since the 1-qubit gate cost is negligible compared to 2-qubit
 404 gate costs, we can model our synthesis problem without worry-
 405 ing about the cost of inverters. This can be done by injecting
 406 inverters for each qubit at the inputs, outputs, and between
 407 stages. We can modify the equations mentioned in this section
 408 to arrive at a theorem similar to Theorem 2 for the minimum
 409 quantum logic implementation cost using inverters (1-qubit) or
 410 other 2-qubit gates.

411 IV. REACHABILITY ANALYSIS

412 Let us first formulate a solution for synthesizing reversible
 413 functions that do not require inverters. Later in this section, we
 414 will extend our formulation for any reversible function with or
 415 without inverters.

416 A. Invariant Checking

417 We have shown in Theorem 2 that finding the quantum
 418 implementation with the minimum cost of a reversible function
 419 (that does not require inverters) is equivalent to solving the
 420 min-2Syn problem.

421 We construct an FSM shown in Fig. 5, use a bounded model
 422 checker [12] to temporally unroll the FSM up to a specific
 423 bound, and invoke an SAT solver to find a counter-example.
 424 Our machine in Fig. 5 is in a way similar to Fig. 4, but there
 425 are some differences. Instead of cascading L instances of the
 426 S functional block in Fig. 4, we have 2^n parallel instances
 427 of FSMs (M_1, \dots, M_{2^n}) in Fig. 5, as many as the number
 428 of rows in the truth table. Each FSM contains a functional
 429 block S . Three primary inputs (G, A, B) are fed to every FSM.
 430 Each machine has its own set of registers (or memory states)
 431 containing \vec{U} (in terms of u_1, \dots, u_n) and E .

432 The FSM will be initialized at time $t = 0$, and then proceeds
 433 to new states at $t = 1, 2, \dots$. For convenience, we use $\vec{\mu}(M_h, t)$
 434 to denote the value of the register vector u_1, \dots, u_n of machine
 435 M_h at time t , where $h = 1, \dots, 2^n$. Similarly, we use $\varepsilon(M_h, t)$

to denote the value of the register E of machine M_h at time t .
 In addition, we use G_t, A_t, B_t to denote the input values at time
 t . As a constraint (environmental assumption), we require

$$\forall t \geq 0 \cdot (A_t \neq B_t). \quad (6)$$

From Fig. 5, we can see that the next state is computed
 from the current state and inputs through the combinational
 functional block S , i.e.,

$$\begin{aligned} &(\vec{\mu}(M_h, t+1), \varepsilon(M_h, t+1)) \\ &= S(G_t, A_t, B_t, \vec{\mu}(M_h, t), \varepsilon(M_h, t)). \end{aligned} \quad (7)$$

We initialize the E register of every machine to 0 (FALSE):
 $\varepsilon(M_h, 0) = 0$ for $h = 1, \dots, 2^n$. We also initialize the \vec{U} regis-
 ters of every machine to their corresponding patterns in a truth
 table, i.e.,

$$\begin{aligned} M_1 &: \vec{\mu}(M_1, 0) = 0 \dots 00 \\ M_2 &: \vec{\mu}(M_2, 0) = 0 \dots 01 \\ &\vdots \\ M_{2^n} &: \vec{\mu}(M_{2^n}, 0) = 1 \dots 11. \end{aligned} \quad (8)$$

Given the reversible function R that we want to synthesize,
 we want to check the nonsynthesizeability invariant

$$inv(t) = \neg \bigwedge_{h=1}^{2^n} (\vec{\mu}(M_h, t) = R(\vec{\mu}(M_h, 0))) \wedge (\varepsilon(M_h, t) = 0)$$

where $inv(t)$ is checked for all time $t \geq 0$.

Theorem 3: The function R is synthesizable using 2-qubit
 gates if and only if there exists a counter-example (input
 sequence G_t, A_t, B_t for $t = 0, \dots, L$) that satisfies (6)–(8)
 and violates the invariant $inv(t)$ at time $t = L$, where L is the

453 corresponding quantum cost using any of those seven 2-qubit
454 gates presented in Section III.

455 *Proof:* Given a counter-example of length L , this counter-
456 example will consist of assignments to the inputs $G_t, A_t,$
457 B_t for $t = 0, \dots, L$. The counter-example satisfies the initial
458 condition (8), which means that all Boolean patterns (from the
459 truth table) for $\vec{U}_{t=0}$ have been explored. The initial condition
460 essentially states that

$$\forall \vec{U}_{t=0} \in \{0, 1\}^n \cdot E_{t=0} = 0. \quad (9)$$

461 For any $t > 0$, the machine states $\vec{\mu}(M_h, t)$ and $\varepsilon(M_h, t)$ are
462 computed from their initial states $\vec{\mu}(M_h, 0)$ and $\varepsilon(M_h, 0)$ and
463 the inputs $G_{t'}, A_{t'}, B_{t'}$ for $t' = 0, \dots, t - 1$. Since our initial
464 condition explored all possible patterns of $\vec{U}_{t=0}$, any formula of
465 the form

$$\forall h \in \{1, \dots, 2^n\} \cdot (\varepsilon(M_h, 0) = 0) \wedge f(\vec{\mu}(M_h, t), \varepsilon(M_h, t))$$

466 can be rewritten as $\forall \vec{U}_{t=0} \in \{0, 1\}^n \cdot (E_0 = 0) \wedge f(\vec{U}_t, E_t)$.
467 We can conjunct the violated invariant $inv(t)$ with the initial
468 condition (9) and rewrite them as

$$\exists G_0 \exists A_0 \exists B_0, \dots, \exists G_L \exists A_L \exists B_L \cdot \forall \vec{U}_{t=0} \in \{0, 1\}^n \\ \cdot (E_{t=0} = 0) \wedge (E_{t=L} = 0) \wedge (\vec{U}_{t=L} = R(\vec{U}_{t=0})). \quad (10)$$

469 The existence of a counter-example is equivalent to the
470 conjunction of formulae (6), (9), and (10). We can rewrite (6)
471 as $\bigwedge_{t=0}^L A_t \neq B_t$. We can also push the conjunction inside the
472 quantification operators to obtain

$$\exists G_0 \exists A_0 \exists B_0, \dots, \exists G_L \exists A_L \exists B_L \\ \cdot \left(\bigwedge_{t=0}^L A_t \neq B_t \right) \wedge \forall \vec{U}_{t=0} \in \{0, 1\}^n \\ \cdot (E_{t=0} = E_{t=L} = 0) \wedge (\vec{U}_{t=L} = R(\vec{U}_{t=0})). \quad (11)$$

473 Equation (11) characterizes the Boolean condition for the ex-
474 istence of a counter-example. The difference between (11) and
475 (5) is that the existential quantification of inputs G_t, A_t, B_t
476 and the constraint on input assumption $A_t \neq B_t$ ranges from 0
477 to L in formula (11) but only ranges from 0 to $L - 1$ in (5). Now
478 observe that the registers (E and \vec{U}) in our FSM (Fig. 5) depend
479 only on the input values of the previous time cycle. Therefore,
480 the input values G_L, A_L, B_L do not affect the existence of
481 our counter-example at all. Hence, the existence of a counter-
482 example is equivalent to the existence of a solution to the
483 L-2Syn problem. ■

484 We have shown that synthesizing the quantum logic is equiv-
485 alent to finding a counter-example to the invariant checking
486 problem. Using bounded model checking, we can find the
487 existence of a counter-example within the length of the bound.
488 By starting with a small bound and gradually increasing the

bound, we can find the shortest counter-example, essentially the
489 minimum cost quantum implementation of the function R . 490

As mentioned in Section III, we can easily modify the above
491 invariant checking formulation to find the minimum quantum
492 implementation cost with inverters or other types of 2-qubit
493 gates. 494

The invariant checking formulation is useful for synthesizing
495 the quantum logic with the minimum cost as outlined above. 496
In case the function R is not synthesizable, as being not
497 reversible, the model checker will prove the invariant has no
498 counter-example (Theorem 3). However, we can easily add
499 ancilla qubits (input constants) to transform nonreversible func-
500 tions to reversible functions, thus making it synthesizable. 501
The next section describes an automatic approach for this
502 transformation. 503

B. Synthesizing With Input Constants 504

Our formulations so far concentrated on synthesizing a
505 function without additional input constants (ancilla qubits). 506
However, some functions (e.g., irreversible functions) cannot
507 be synthesized without input constants. For these functions, it
508 makes sense to synthesize them with the minimum number of
509 input constants. 510

We can add k input constants to the original $n \times n$ circuit,
511 making it an $(n + k) \times (n + k)$ circuit, run it through our
512 model checker, and see if we can get a counter-example or
513 a proof. If we get a proof, we can increment k until we
514 eventually get a counter-example (which should happen for
515 finite k according to [10]). A systematic way of doing this is
516 to start with $k = 1$ and gradually increment k until we reach a
517 counter-example. 518

The invariant checking formulation with k input constants is
519 slightly different from Section IV-A. For every input constant
520 bit, we do not know if it should be a constant 0 or a constant 1.
521 In order to get a counter-example (i.e., synthesize the circuit),
522 we want to find out these constant values. 523

Let us look back at our machines in Fig. 5. From the
524 figure, we have 2^n machines (M_1, \dots, M_{2^n}), each with n
525 registers. We modify this figure so that we have $n + k$ registers
526 (u_1, \dots, u_{n+k}) in each machine and each S (and δ if applica-
527 ble) functional block will handle $n + k$ instead of n registers,
528 as well as the E register. 529

For notational clarity, we still use $\vec{\mu}(M_i, t)$ to denote the
530 value of the register vector (u_1, \dots, u_n) for machine M_i (where
531 $i = 1, \dots, 2^n$) at time t . We use $\nu_j(M_i, t)$ to denote the value
532 of each register u_j (where $j = 1, \dots, n + k$) of machine M_i
533 (where $i = 1, \dots, 2^n$) at time t . Thus, the newly introduced reg-
534 ister values can be referred to as $\nu_{n+1}(M_i, t), \dots, \nu_{n+k}(M_i, t)$. 535

Let us also introduce a new state ζ in addition to all the 2^n
536 machines (M_1, \dots, M_{2^n}). This register is initialized to 0 and
537 then set to 1 thereafter, i.e., 538

$$\zeta = \begin{cases} 0, & t = 0 \\ 1, & t > 0. \end{cases} \quad (12)$$

For those additional k registers, we want to limit their initial
539 state to the set $\{0, 1\}$. In addition, we want to restrict the initial
540

541 state of the j th register ($j = n + 1, \dots, n + k$) at each machine
542 to be the same

$$\bigwedge_{j=n+1}^{n+k} \bigwedge_{i=1}^{2^n} \nu_j(M_i, 0) \in \{0, 1\} \quad (13)$$

$$\bigwedge_{j=n+1}^{n+k} \zeta_t \vee (\nu_j(M_1, t) = \dots = \nu_j(M_{2^n}, t)). \quad (14)$$

543 Equation (13) is possible because the symbolic model checking
544 formulations [11] allow the initial state to be a set of values.
545 The constraint (14) is used as an assumption that restricts the
546 state space. Notice that we still have 2^n FSMs (M_1, \dots, M_{2^n})
547 overall because the number of rows in the truth table for input
548 patterns is still the same as in the case without the additional
549 input constants.

550 By increasing the combinational function blocks S (and δ
551 if applicable), we are essentially synthesizing for $(n + k) \times$
552 $(n + k)$ reversible logic. Our initial state specification allows us
553 to consider the constant 0 and constant 1 cases. The generated
554 counter-example will contain specific values for the initial state
555 of each bit, thus finding out the constant input values.

556 C. Example

557 Consider a classical computation unit, half adder, which
558 takes two input bits ($n = 2$) and outputs a sum and a carry.
559 There are two input patterns in its truth table ($ab = 01, 10$)
560 that produce the same output pattern. Therefore, one needs to
561 add a single input constant in order to separate 01 and 10 and
562 to create a 3×3 reversible function R . We construct the 2^2
563 machines and the invariant according to Sections IV-A and
564 IV-B. A model checker can return a counter example that
565 contains values of initial states and a sequence of input values.
566 Most of the initial state values are already specified in (8),
567 except for the state values of the input constant in (13). In this
568 case, the model checker tells us that the input constant is 0, i.e.,

$$\nu_3(M_1, 0) = \nu_3(M_2, 0) = \dots = \nu_3(M_{2^2}, 0) = 0.$$

569 Hence, the initial state values of the FSM are

$$M_1 : \vec{\mu}(M_1, 0) = 000$$

$$M_2 : \vec{\mu}(M_2, 0) = 001$$

$$M_3 : \vec{\mu}(M_3, 0) = 010$$

$$M_4 : \vec{\mu}(M_4, 0) = 011.$$

570 The sequence of input values in the counter-example is

$$G_0 = V^+, \quad A_0 = 2, \quad B_0 = 3$$

$$G_1 = \text{XOR}, \quad A_1 = 1, \quad B_1 = 2$$

$$G_2 = V, \quad A_2 = 2, \quad B_2 = 3$$

$$G_3 = V_XOR, \quad A_3 = 1, \quad B_3 = 3.$$

Notice that the above input sequence satisfies the constraint (6).
If we substitute these inputs back into the FSM, we will arrive at

$$M_1 : \vec{\mu}(M_1, 4) = 000$$

$$M_2 : \vec{\mu}(M_2, 4) = 110$$

$$M_3 : \vec{\mu}(M_3, 4) = 010$$

$$M_4 : \vec{\mu}(M_4, 4) = 001.$$

We compare the final state with the initial state. The initial
state of M_1, \dots, M_4 corresponds to the input patterns in a truth
table. The final state of M_1, \dots, M_4 corresponds to the output
patterns in a truth table. Notice that the least significant bit
(rightmost column) in the final state satisfies the carry function,
and the middle bit (middle column) satisfies the summation
function. The bottom bit in Fig. 10 is a garbage function $a \wedge -b$.
Let us assume that in the circuit the top qubit corresponds to the
least significant bit of our truth table, and similarly the bottom
qubit corresponds to the most significant bit of our truth table.
The gate types (quantum XOR, controlled- V , etc.) are already
given by G_0, \dots, G_3 of the counter-example. The connection
of these gates to qubits are given by $A_0, B_0, \dots, A_3, B_3$. These
values directly translate to the circuit in Fig. 10.

V. COMPLEXITY AND TIME

Industrial experience [20], [22] suggests that the complexity
of model checking is sensitive to the number of state retaining
elements in the FSM. For our FSM in Fig. 5, there are $n \times$
 2^n registers, where n is the number of qubits. Each register
has four possible values (0, 1, V , V^+). If we use Boolean
states to encode these registers, we have $2n \times 2^n$ Boolean state
elements. However, the number of qubits n tends to be small
due to physical limitations. So far, the largest number [24] of
qubits is 7, which is 1792 Boolean state elements. This is still
manageable in the scope of industrial strength bounded model
checkers [20], [22]. Nevertheless, we would like to speed up
our synthesis process.

We introduce two speed up methods in this section. The first
method breaks the synthesis process into two or more smaller
synthesis stages. The second method constrains the location of
certain gates (such as the controlled- V or controlled- V^+ gates),
which reduces the search space of the algorithm.

A. Synthesis in Multiple Stages

We devised a strategy to speed up the synthesis process at the
expense of a higher circuit cost. Given an $n \times n$ reversible gate
to synthesize, there are 2^n cases to be enumerated. Assume,
however, that we pick one of the inputs, say the first input, and
consider only cases where it is 0. Then we have 2^{n-1} cases. To
perform reachability analysis, we construct the same FSM as
shown in Fig. 5, but check it with a different invariant $inv'(t)$

$$\neg \left[\bigwedge_{h=1}^{2^{n-1}} (\vec{\mu}(M_h, t) = R(\vec{\mu}(M_h, 0))) \wedge \bigwedge_{h=1}^{2^n} (\varepsilon(M_h, t) = 0) \right].$$

613 The main difference between $inv'(t)$ and $inv(t)$ is that the new
 614 invariant $inv'(t)$ checks that R is accomplished for only half of
 615 all the possible input patterns, which accounted for those cases
 616 where the first input is 0. It is easier to find a counter-example
 617 for this new invariant because only half of the cases have to
 618 be accomplished. We take a snapshot of all register states at
 619 the end of this counter-example and use it as the initial state of
 620 the FSM. We then run model checker again with our original
 621 invariant $inv(t)$. This time, since we started from a state fairly
 622 close to R , it is easier to generate a counter-example. According
 623 to Theorem 4, this method guarantees to generate the counter-

624 example if the function that we want to synthesize is reversible.
 625 *Theorem 4:* Suppose we want to synthesize a reversible
 626 function R , and suppose we have already synthesized another
 627 reversible function Q , then there exists a reversible function P
 628 such that R is equivalent to the cascade of Q and P , i.e., $R =$
 629 $Q \circ P$, where R , Q , and P are all $n \times n$ reversible functions.

630 *Proof:* Since Q is reversible, we have function Q^{-1} such
 631 that $Q \circ Q^{-1} = I$, where I is the identity function (outputs are
 632 equal to inputs). Hence, there exists $P = Q^{-1} \circ R$ such that
 633 $Q \circ P = Q \circ Q^{-1} \circ R = I \circ R = R$. ■

634 B. Constraining Search Space

635 The runtime complexity of model checking is due to its
 636 exhaustive nature. We can introduce more constraints to reduce
 637 the search space. For instance, we can limit the location of the
 638 data input for the controlled- V and controlled- V^+ gates to a
 639 subset of the qubits (such as the first qubit). This example will
 640 mean that (5) in Definition 1 will be changed to

$$\begin{aligned}
 & \exists G_0 \exists A_0 \exists B_0, \dots, \exists G_{L-1} \exists A_{L-1} \exists B_{L-1} \\
 & \cdot \left(\forall \vec{U}_0 \in \{0, 1\}^n \cdot (E_0 = E_L = 0) \wedge \left(\vec{U}_L = R(\vec{U}_0) \right) \right) \\
 & \wedge \left(\bigwedge_{i=0}^{L-1} ((G_i = \text{Fig. 1(c)}) \vee (G_i = \text{Fig. 1(d)})) \Rightarrow B_i \right) \\
 & \wedge \left(\bigwedge_{i=0}^{L-1} A_i \neq B_i \right). \tag{15}
 \end{aligned}$$

641 Once formula (5) is changed, all subsequent logic reasoning can
 642 be adjusted for the constraint as well.

643 Formula (15) is just an example to limit the location of the
 644 V input to the first qubit. Similar constraints can be constructed
 645 to limit the location of the control input for the controlled- V
 646 and/or controlled- V^+ gates, or to limit the control or data inputs
 647 of the Feynman gates, etc.

648

VI. EXPERIMENTS

649 We constructed our invariant checking formulations de-
 650 scribed in Section IV using NuSMV with BerkMin [25]. Our
 651 method was applied to synthesize some common quantum cir-
 652 cuits. All experiments are conducted on a 850-MHz Pentium III
 653 processor running on Linux.

654 The quantum costs of several circuits are summarized in
 655 Table I. The ‘‘Prior’’ and ‘‘Our’’ columns indicate the best pub-

TABLE I
 QUANTUM COST OF COMMON CIRCUITS

| Circuit | Prior | Our | Constraint | Circuit | Time (sec) |
|----------------|-------|-----|------------|---------|------------|
| Miller | 7 | 6 | No | Fig. 6 | 318.29 |
| Fredkin | 5 | 5 | No | Fig. 7 | 78.02 |
| Peres | 4 | 4 | No | Fig. 8 | 35.18 |
| Toffoli | 5 | 5 | No | Fig. 9 | 122.52 |
| Half-adder | 6 | 4 | No | Fig. 10 | 6.77 |
| Half-adder2 | N/A | 4 | No | Fig. 11 | 26.25 |
| q4-example | N/A | 5 | Yes | Fig. 12 | 34.78 |
| Peres-double | 8 | 6 | Yes | Fig. 14 | 171.27 |
| Toffoli-double | 10 | 7 | Yes | Fig. 15 | 853.78 |

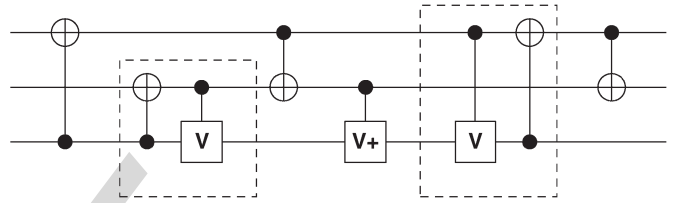


Fig. 6. Miller gate with optimum quantum cost = 6.

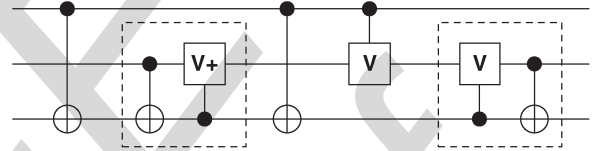


Fig. 7. Alternative Fredkin gate implementation with quantum cost = 5.

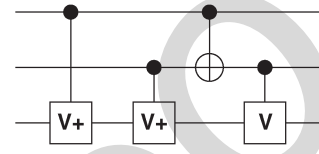


Fig. 8. Peres gate implementation with optimum quantum cost = 4.

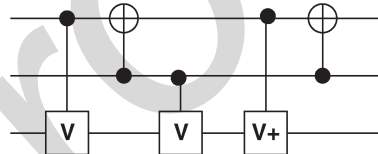


Fig. 9. Toffoli gate implementation with optimum quantum cost = 5.

lished quantum cost in previous literature and our synthesized
 quantum cost, respectively. For Miller gate [26], our synthesis
 result has a quantum implementation cost of 6, shown in Fig. 6.
 It is better than any previously published result (cost of 7)
 [26], [27].

For the Fredkin [10], Peres [17], and Toffoli [5], [16] gates,
 our synthesized results (Fig. 7–9) have the same quantum costs
 as reported in prior literature [4], [27]. But nobody was able to
 show that the cost was minimum until now. Notice also that our
 synthesized Fredkin circuit (Fig. 7) is different from the circuit
 in [4], but they are functionally equivalent (due to the canonical
 unitary matrix as described in Theorem 1).

We synthesized a classical half adder using input constants
 discussed in Section IV-B. In the past, people have been syn-
 thesizing the 2-bit adder using a Toffoli gate and a quantum
 XOR gate [23], [28]. Since the Toffoli gate has a minimum
 cost of 5 and the quantum XOR gate cost 1, the total quantum

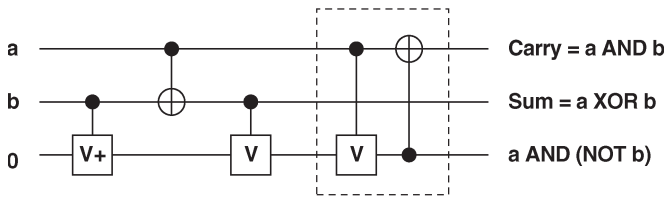


Fig. 10. Half adder with quantum cost = 4.

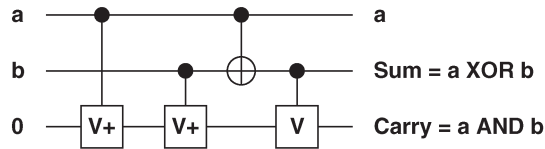


Fig. 11. Alternative half adder with quantum cost = 4.

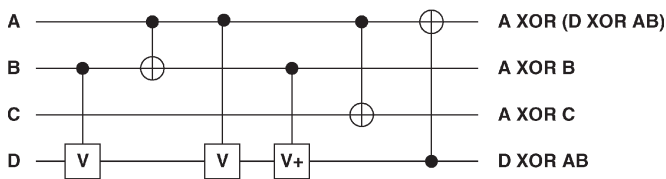


Fig. 12. q4 example.

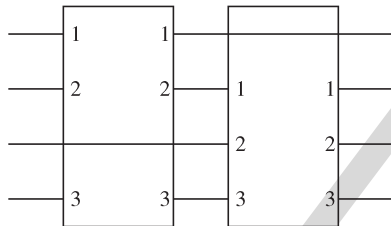


Fig. 13. Peres-double and Toffoli-double specification.

673 implementation cost would be 6 using that method. Our method
674 proved that the minimum quantum cost is actually 4, as shown
675 in Fig. 10. In fact, if we do not restrict the output of the adder to
676 be the top two qubits, we can put one of the desired outputs on
677 the ancilla qubit. Such an implementation is actually the Peres
678 circuit with the last qubit input set to zero, shown in Fig. 11.

679 We also synthesized several 4-qubit functions using the
680 method in Section V-B by restricting the data input/output
681 of the controlled- V or controlled- V^+ gates to be the fourth
682 qubit. The “q4-example” is a simple 4-qubit function shown in
683 Fig. 12. The “Peres-double” and “Toffoli-double” functions are
684 specified by cascading two 3-qubit Peres and Toffoli functions,
685 respectively, in a 4-qubit manner shown in Fig. 13, where the
686 numbers 1, 2, and 3 indicates the input/output correspondence
687 to the first, second, and third qubit of the original Peres or
688 Toffoli functions. Since the smallest quantum cost of Peres and
689 Toffoli gates are known to be 4 and 5, respectively, the quantum
690 cost of having two Peres and Toffoli in a cascading manner
691 would be 8 and 10, respectively. However, our synthesis result
692 indicate that their quantum cost can be heuristically decreased
693 to 6 (Fig. 14) and 7 (Fig. 15), respectively.

694 We synthesized the full adder using four different strategies
695 shown in Table II. Recent papers [6], [7] used two Toffoli
696 gates and two Feynman gates to implement a quantum cost

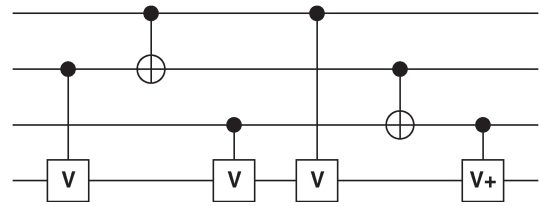


Fig. 14. Peres-double quantum cost = 6.

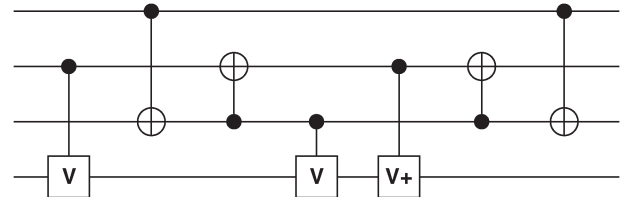


Fig. 15. Toffoli-double quantum cost = 7.

TABLE II
SYNTHESIS OF FULL ADDER

| Method | Garbage input to output | Circuit | Cost | Time |
|------------|-------------------------|---------|------|-------------|
| Optimal | Yes | Fig. 16 | 6 | 7 hours |
| 2-stage | Yes | Fig. 17 | 9 | 140.83 sec |
| Constraint | Yes | Fig. 16 | 6 | 1104.97 sec |
| Constraint | No | Fig. 18 | 6 | 176.09 sec |

of 12. We proved that the minimum quantum cost for a full
697 adder is 6, as shown in Fig. 16. To shorten the CPU runtime
698 for synthesizing the full adder, we used a two-stage strategy
699 mentioned in Section V-A and obtained an implementation with
700 quantum cost of 9, shown in Fig. 17. The CPU runtime is
701 significantly reduced (from 7 h to 140.83 s). Notice that the
702 cost of this implementation can be reduced to 8 if we choose
703 to omit the “propagate” logic (the last quantum XOR gate). We
704 also applied the synthesis method in Section V-B by restricting
705 the data input of the controlled- V or controlled- V^+ gates to the
706 location of the “sum” qubit. The runtime is reduced from 7 h
707 to 1104.97 s, and the quantum cost is the same as the original
708 optimal method. All the top three experiments in Table II use
709 a specification such that the useful output (sum and carry-out)
710 does not use the same qubit as the garbage input (ancilla qubit).
711 We remove this requirement in the last experiment of Table II
712 and used the input/output specification in [6] and [7]. The result
713 is shown in Fig. 18 and the synthesis took 176.09 CPU seconds.
714

VII. CONCLUSION

715

In this paper, we applied invariant checking, a formal veri-
716 fication technique, to the synthesis of quantum logic circuits.
717 We reduced problems in quantum logic synthesis to those of
718 multiple-valued logic synthesis, thus simplifying the search
719 space and algorithm complexity. To solve the synthesis prob-
720 lem, we created an optimal synthesis method, a multistage syn-
721 thesis method, and several constraint-related speed-up methods.
722 Our optimal method and the multistage method are guaranteed
723 to synthesize the circuit. We created minimum cost quantum
724 circuits for Miller gate, half adder, and full adder, which are
725 better than previous results. This cost is minimum for any
726 circuit using our set of quantum gates, where the control qubit
727

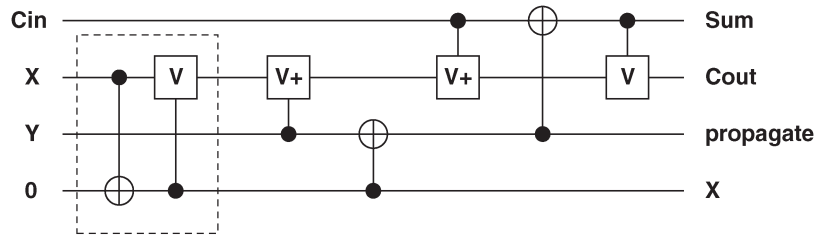


Fig. 16. Full adder with quantum cost = 6.

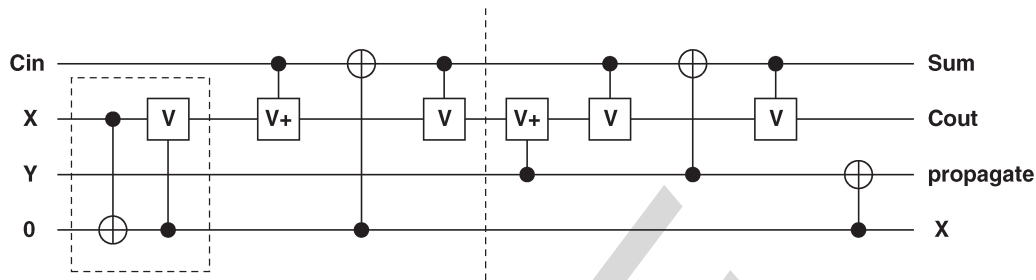


Fig. 17. Full adder with quantum cost = 9.

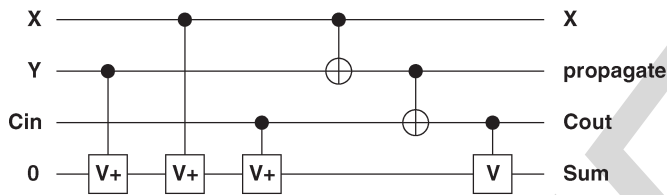


Fig. 18. Full adder with different output arrangement: quantum cost = 6.

728 of 2-qubit gates is always basis binary. We also proved the min-
 729 imum quantum cost in the same manner for Fredkin, Peres, and
 730 Toffoli gates. In addition, we found quantum implementations
 731 with lower cost (than previous known results) for (cascaded)
 732 double Peres gates and (cascaded) double Toffoli gates. As
 733 shown in Section VI, our method can also automatically convert
 734 a nonreversible (irreversible) function to the simplest equivalent
 735 reversible function (Fig. 16) by adding and initializing the
 736 minimum number of ancilla wires. This step is missing from
 737 most reversible circuit synthesis algorithms, and the problem
 738 of minimal conversion was never discussed in the literature.
 739 We have demonstrated our method on small circuits. It can
 740 be a starting point to create such methods for larger Boolean
 741 functions. Our work is the first successful application of formal
 742 methods and satisfiability in quantum logic synthesis.

REFERENCES

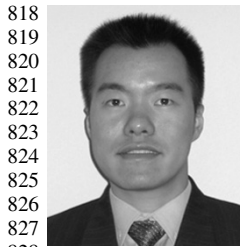
743
 744 [1] A. De Vos, "Reversible computing," *Prog. Quantum Electron.*, vol. 23,
 745 no. 1, pp. 1–49, Jan. 1999.
 746 [2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum*
 747 *Information*. Cambridge, U.K.: Cambridge Univ. Press, Dec. 2000.
 748 [3] K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation rules for
 749 designing CNOT-based quantum circuits," in *Proc. Design Automation*
 750 *Conf.*, New Orleans, LA, 2002, pp. 419–424.
 751 [4] J. A. Smolin and D. P. DiVincenzo, "Five two-bit quantum gates are
 752 sufficient to implement the quantum Fredkin gate," *Phys. Rev. A, Gen.*
 753 *Phys.*, vol. 53, no. 4, pp. 2855–2856, Apr. 1996.
 754 [5] A. Barenco *et al.*, "Elementary gates for quantum computation," *Phys.*
 755 *Rev. A, Gen. Phys.*, vol. 52, no. 5, pp. 3457–3467, Nov. 1995.
 756 [6] A. Khlopotine, M. Perkowski, and P. Kerntopf, "Reversible logic synthesis
 757 by iterative compositions," in *Proc. Int. Workshop Logic Synthesis*, New
 758 Orleans, LA, 2002, pp. 261–266.

[7] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based 759
 algorithm for reversible logic synthesis," in *Proc. Design Automation* 760
Conf., Anaheim, CA, 2003, pp. 318–323. 761
 [8] V. V. Shende *et al.*, "Synthesis of reversible logic circuits," *IEEE Trans.* 762
Comput.-Aided Des. Integr. Circuits Syst., vol. 22, no. 6, pp. 710–722, 763
 Jun. 2003. 764
 [9] X. Song *et al.*, (2005). Algebraic characteristics of reversible 765
 gates *Theory of Computing Systems* [Online]. Available: [http://www.](http://www.springerlink.com/link.asp?id=vh2mkff02xwm2gdb) 766
[springerlink.com/link.asp?id=vh2mkff02xwm2gdb](http://www.springerlink.com/link.asp?id=vh2mkff02xwm2gdb), Article In Press 767
 [10] E. Fredkin and T. Toffoli, "Conservative logic," *Int. J. Theor. Phys.*, 768
 vol. 21, no. 3/4, pp. 219–253, 1982. 769
 [11] K. L. McMillan, *Symbolic Model Checking*. Norwell, MA: Kluwer, 770
 1993. 771
 [12] A. Biere *et al.*, "Symbolic model checking using SAT procedures instead 772
 of BDDs," in *Proc. Design Automation Conf.*, New Orleans, LA, 1999, 773
 pp. 317–320. 774
 [13] A. Einstein, B. Podolsky, and N. Rosen, "Can quantum-mechanical de- 775
 scription of physical reality be considered complete?" *Phys. Rev. A, Gen.* 776
Phys., vol. 47, no. 10, pp. 777–780, Mar. 1935. 777
 [14] D. Deutsch, "Quantum computational networks," *Proc. R. Soc. Lond. A,* 778
Math. Phys. Sci., vol. 425, pp. 73–90, 1989. 779
 [15] J. A. Jones and M. Mosca, "Implementation of a quantum algorithm on a 780
 nuclear magnetic resonance quantum computer," *J. Chem. Phys.*, vol. 109, 781
 no. 5, pp. 1648–1653, Aug. 1998. 782
 [16] T. Sleator and H. Weinfurter, "Realizable universal quantum logic gates," 783
Phys. Rev. Lett., vol. 74, no. 20, pp. 4087–4090, May 1995. 784
 [17] A. Peres, "Reversible logic and quantum computers," *Phys. Rev. A, Gen.* 785
Phys., vol. 32, no. 6, pp. 3266–3276, Dec. 1985. 786
 [18] R. E. Bryant, "Graph-based algorithms for Boolean function manipula- 787
 tion," *IEEE Trans. Comput.*, vol. C-35, no. 8, pp. 677–691, Aug. 1986. 788
 [19] W. N. N. Hung *et al.*, "BDD minimization by scatter search," *IEEE Trans.* 789
Comput.-Aided Des. Integr. Circuits Syst., vol. 21, no. 8, pp. 974–979, 790
 Aug. 2002. 791
 [20] F. Copt *et al.*, "Benefits of bounded model checking at an industrial 792
 setting," in *Proc. Computer-Aided Verification*, Paris, France, 2001, 793
 pp. 436–453. 794
 [21] W. N. N. Hung *et al.*, "Segmented channel routability via satisfiability," 795
ACM Trans. Des. Automat. Electron. Syst., vol. 9, no. 4, pp. 517–528, 796
 Oct. 2004. 797
 [22] W. N. N. Hung and N. Narasimhan, "Reference model based RTL 798
 verification: An integrated approach," in *Proc. IEEE Int. High Level* 799
Design Validation Test Workshop, Sonoma Valley, CA, Nov. 2004, 800
 pp. 9–13. 801
 [23] A. Ekert, P. Hayden, and H. Inamori, "Basic concepts in quantum compu- 802
 tation," in *Coherent Atomic Matter Waves—Ondes de matiere coherentes.* 803
 NATO Advanced Study Inst., Aug. 1999, pp. 659–699. 804
 [24] L. M. K. Vandersypen *et al.*, "Experimental realization of Shor's quantum 805
 factoring algorithm using nuclear magnetic resonance," *Nature*, vol. 414, 806
 no. 6866, pp. 883–887, Dec. 2001. 807

AQ1

AQ2

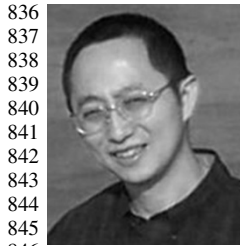
- 808 [25] E. Goldberg and Y. Novikov, "BerkMin: A fast and robust SAT solver,"
809 in *Proc. Design Automation Test Eur. (DATE)*, Paris, France, 2002,
810 pp. 142–149.
- 811 [26] D. M. Miller, "Spectral and two-place decomposition techniques in re-
812 versible logic," in *Proc. IEEE Midwest Symp. Circuits Systems*, Aug.
813 2002, pp. II-493–II-496.
- 814 [27] G. Yang, W. N. N. Hung, X. Song, and M. Perkowski, "Majority-based
815 reversible logic gates," *Theor. Comput. Sci.*, vol. 334, no. 1–3, pp. 259–
816 274, Apr. 2005.
- AQ4 817 [28] J. Gruska, *Quantum Computing*. McGraw-Hill, Apr. 1999.



William N. N. Hung received the B.S. and M.S. degrees in electrical and computer engineering from the University of Texas, Austin, in 1994 and 1997, respectively, and the Ph.D. degree in electrical and computer engineering from Portland State University, Portland, OR, in 2002.

From 1997 to 2004, he was a Senior Engineer at Intel Corporation, Hillsboro, OR, primarily focused on formal property verification of CPU designs. Since September 2004, he has been a Senior Staff Engineer at Synplicity Inc., Sunnyvale, CA. His

829 research interests include logic synthesis, physical design, formal methods,
830 satisfiability, combinatorial optimization, and quantum computing.
831 Dr. Hung served as a member in the Emergent Technologies Technical
832 Committee for the IEEE Computational Intelligence Society. He also served as
833 a member in the Program Committee of the IEEE/ACM Design Automation and
834 Test in Europe (DATE) and in the Program Committee of the IEEE International
835 Computer Software and Applications Conference (COMPSAC).



Xiaoyu Song received the Ph.D. degree from the University of Pisa, Pisa, Italy, in 1991.

From 1992 to 1999, he was on the faculty at the University of Montreal, Canada. In 1998, he was a Senior Technical Staff member at Cadence, San Jose, CA. Since 1999, he has been on the faculty at the Department of Electrical and Computer Engineering, Portland State University, Portland, OR. His research interests include synthesis, verification, and testing of high-performance digital system designs, low-power digital IC designs, timing analysis, and

847 formal methods.

848 Dr. Song served as an Associate Editor of the IEEE TRANSACTIONS ON
849 CIRCUITS AND SYSTEMS and the IEEE TRANSACTIONS ON VERY LARGE
850 SCALE INTEGRATION (VLSI) SYSTEMS.

Guowu Yang received the B.S. degree in mathematics from the University of Science and Technology, China, in 1989, and the Ph.D. degree in electrical and computer engineering from Portland State University, Portland, OR, in 2005.

851
852
853 He is currently a Post-Doctoral Researcher in the Department of Computer
854 Science, Portland State University. His research interests include quantum
855 computing, reversible logic, hardware formal verification, floor planning, and
856 routing. 857

Jin Yang received the B.S. and M.S. degrees in computer science from Peking University, China, and the Ph.D. degree in computer science from the University of Texas, Austin, in 1997.

858
859
860 He was a Faculty Member at Peking University for two years before he came
861 to the U.S. In 1995, he joined Intel, Hillsboro, OR, and is currently a Principal
862 Engineer at Intel Strategic CAD Labs. He holds five U.S. patents. His research
863 interests include in all aspects of formal methods, with a focus on developing
864 practical solutions for hardware specification and verification. 865



Marek Perkowski received the M.S. degree in electronics and the Ph.D. degree in automatic control from the Technical University of Warsaw, Warsaw, Poland.

866
867
868
869
870 He has been on the faculty of Warsaw Technical University, the University of Minnesota, and the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. He has been a Visiting Faculty Member at the Technical University of Eindhoven, The Netherlands, the University of Montpellier, France, and Kyushu Institute of Technology, Japan. He worked as Summer Professor at Intel, GTE, and Sharp, and was a Consultant to several companies including Cypress Semiconductor. He is currently a Professor at Portland State University, Portland, OR. His research interests include quantum computing, automated synthesis of quantum and reversible circuits, testing of quantum circuits, and quantum computational intelligence with intelligent robotics applications.

881
882
883 Dr. Perkowski is the Chair of the IEEE Computer Society Technical Committee on Multiple-Valued Logic. 884

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

AQ1 = Please provide issue number in Ref. [14].

AQ2 = Please provide publisher location in Ref. [23].

AQ3 = Please provide conference location in Ref. [26].

AQ4 = Please provide publisher location in Ref. [28].

AQ5 = Please specify the field of study.

AQ6 = Please specify the year when the degrees were earned.

AQ7 = Please specify the year when the degrees were earned.

Note: Please provide photo for authors G. Yang and J. Yang.

END OF ALL QUERIES

IEEE
Proof