

## MODELLING REVERSIBLE CIRCUITS BY IF-DECISION DIAGRAMS

Prihozhy A.A.

*Belarusian National Technical University, Minsk, Belarus*

*prihozhy@yahoo.com*

*Binary decision diagram.* Binary decision diagram (BDD) is a well-known and widely used graph model (data structure) of Boolean functions [1]. In work [2] Randal Bryant proposed a reduced ordered binary decision diagram (ROBDD) that emphasizes aspects of variables ordering and diagram size reduction. It lies in the basis of digital system modelling, synthesis and verification tools. It is canonical for a particular function and variable ordered. ROBDDs constitute a basis of developing efficient digital system modelling, synthesis, optimization and verification tools. Figure 1a shows a basic fragment of BDD constructed on the Shannon expansion of Boolean function  $f(x)$  of vector argument  $x = (x_1, \dots, x_n)$ :

$$f = x_i \wedge f_{x_i=1} \vee \neg x_i \wedge f_{x_i=0} \quad (1)$$

where  $\neg$ ,  $\wedge$  and  $\vee$  are Boolean inversion, conjunction and disjunction;  $f_{x_i=0}$  and  $f_{x_i=1}$  are residual functions or negative and positive cofactors respectively. The fragment consists of a vertex labeled by variable  $x_i$  that has two outgoing edges *low* and *high* directed to sub-diagrams, which represent the negative and positive cofactors.

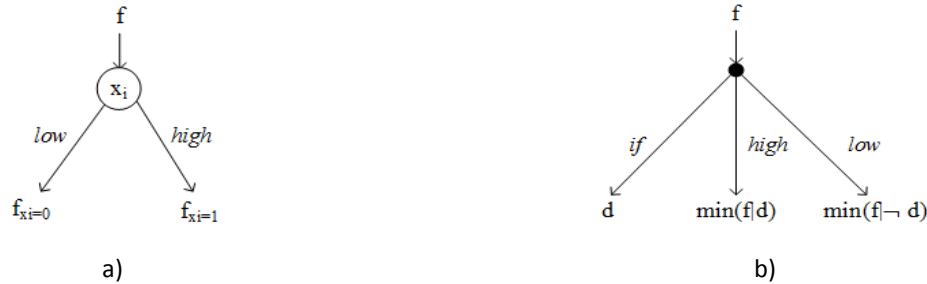


Figure 1 – Basic fragment of a) binary decision diagram BDD and b) if-decision diagram IFD

*If-decision diagram.* Works [3, 4] originally propose the concept of if-decision diagram derived from the theory of incompletely specified Boolean functions [5-7]. Let  $B = \{0, 1\}$  and  $M = \{0, 1, dc\}$  where 0 and 1 are Boolean values and  $dc$  is a don't care value. An incompletely specified Boolean function  $\varphi(x)$  of vector Boolean variable  $x = (x_1, \dots, x_n)$  is a mapping  $\varphi: B^n \rightarrow M$ . In  $\varphi$ , value  $dc \in M$  can be arbitrarily replaced by 0 or 1. Function  $\varphi(x)$  can be represented by three sets: on-set  $ON^\varphi$  where  $\varphi(x) = 1$ , off-set  $OFF^\varphi$  where  $\varphi(x) = 0$ , and don't care set  $DC^\varphi$  where  $\varphi(x) = dc$ . Three Boolean characteristic functions describe the sets:  $\varphi^{on}(x)$ ,  $\varphi^{off}(x)$  and  $\varphi^{dc}(x)$ . We call function  $f(x) = \varphi^{on}(x)$  a value function, and call function  $d(x) = \neg\varphi^{dc}(x)$  a domain function. Pair  $\varphi(x) = (f(x) | d(x))$  describes the incompletely specified function. In the pair, one may replace  $f(x)$  by other function  $v(x)$  of slice (2) without changing  $\varphi(x)$ .

$$(f \wedge d)^{on} \subseteq v^{on} \subseteq (f \vee d)^{on} \quad (2)$$

Since the functions of slice (1) can produce digital circuits of various time and area, we introduce an operation  $v(x) = \min(f(x) | d(x))$  to select a best function of the slice [3, 5]. The theorem as follows generalizes the Shannon expansion. Let  $\min(f | d)$  and  $\min(f | \neg d)$  be residual functions (cofactors) of function  $f$  on function  $d$ .

*Theorem.* Expansion (3) holds for arbitrarily Boolean functions  $f(x)$  and  $d(x)$ .

$$f = d \wedge \min(f | d) \vee \neg d \wedge \min(f | \neg d) \quad (3)$$

Expansion (3) is capable of efficiently solving many optimization problems of digital system design. The if-decision diagram (IFD) [3] represents expansion (3) by a node (Figure 1b) of directed acyclic graph. Its three descendants are if-node  $d$ , high-node  $g = \min(f | d)$  and low-node  $h = \min(f | \neg d)$ , which form a node notation  $v = ifd(d, g, h)$ . A labeled terminal node is Boolean constant 0, constant 1, variable  $x_i$  or its negation  $\neg x_i$ . If node  $v$  uses complement of  $d, g$  and  $h$ , then we write  $v = ifd(\neg d, \neg g, \neg h)$ . In this case, complement edges connect  $v$  with  $d, g$  and  $h$ . We represent normal edges by solid lines, and represent complement edges by dash lines.

IFD is a promising generalization of BDD. It has three outgoing edges instead of two in BDD; therefore, its capabilities for parallelization are larger. In our work, we use IFD for the modelling, optimization and parallelization [4, 8] of digital circuits at logic level, as well as for the synthesis of reversible circuits for quantum implementation.

*Reversible and quantum circuits.* Synthesis of reversible and quantum logic is an intensely studied topic [9-20]. A logic function is reversible if it maps each input assignment to a unique output assignment. Such a function must have the same number of input and output variables. Fan-out and feedback are not allowed in the reversible logic. A circuit realizing a reversible function is constructed of lines and reversible gates. The reversible gate has the form of  $G(T, C)$ , where  $T \subset X$  is a target line,  $C \subset X$  is a set of control lines ( $C \cap T = \emptyset$ ), and  $X$  is a set of variables. The gate operation is applied to the target lines if all control lines meet true conditions.

Several reversible gate libraries are available. The NCT library [10] includes such fundamental reversible gates as NOT gate, CNOT (Feynman) gate with one control line, and C2NOT (Toffoli) gate with two control lines. Figure 2a depicts the fundamental gates of the NCT library and the functions implemented by the gates. Syntactically the gate (target line) is represented by symbol  $\oplus$ , and its control lines are represented by symbol  $\bullet$ . The Toffoli gate implements a Boolean function that maps the three inputs to the three outputs:  $TG(L, C_0, C_1) = (L \oplus (C_0 \wedge C_1), C_0, C_1)$  where  $\oplus$  is Boolean exclusive or;  $L$  is the gate input at target line;  $C_0, C_1$  are conditions at two control lines. The gate may have one control line, or may have no control lines. Therefore, the reversible circuit describes the behavior as a superposition of three Boolean functions,  $\neg, \wedge$  and  $\oplus$ . If the Toffoli gate has one control line, then  $TG(L, C_0) = (L \oplus C_0, C_0)$ . Boolean inversion (not) of  $L$  is realized by a gate without control lines:  $TG(L) = (\neg L)$ .

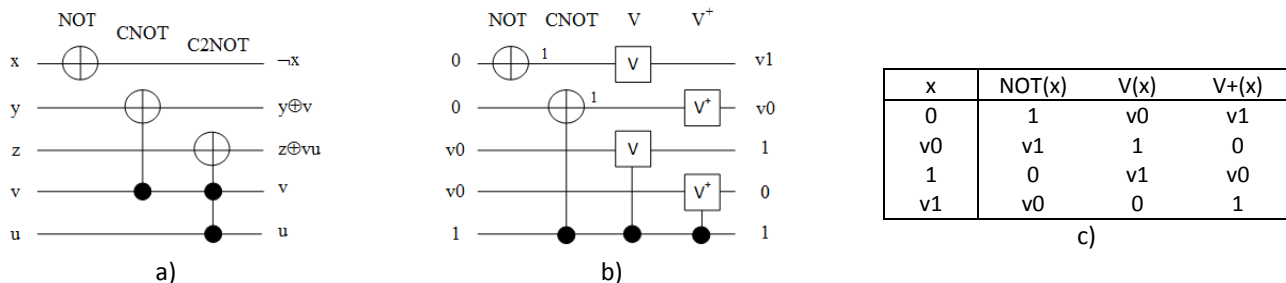


Figure 2 – Reversible gates of a) NCT library, quantum gates of b) NCV- $|v1\rangle$  library and c) operation of quantum gates

Quantum circuits carry out computations by manipulating quantum states of qubits. The qubit represents the state as  $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$  where  $\alpha$  and  $\beta$  are complex numbers such that  $|\alpha|^2 + |\beta|^2 = 1$ . Quantum logic gates are necessarily reversible in nature. A quantum circuit is a cascade of quantum

gates. Several quantum gate libraries are available in the literature. The NCV library [10] is the most commonly used for generating quantum circuits. Work [15] introduced the extended NCV- $|v_1\rangle$  quantum library. The library includes gates whose control lines may be sensitive to non-Boolean values. It uses qudits instead of qubits and considers a 4-level (0,  $v_0$ , 1 and  $v_1$ ) quantum system. Figure 2b depicts the quantum gates of the NCV- $|v_1\rangle$  library: NOT gate, controlled CNOT gate (both are similar to the corresponding reversible gates), not controlled and controlled V gate, and not controlled and controlled  $V^+$  gate. Figure 2c describes operation of the gates.

Works [14, 16, 20] propose a technique of synthesizing a reversible circuit from a function given as BDD. The technique substitutes all nodes of the BDD with cascades of reversible gates. The size of reversible circuit directly depends on the BDD size; therefore, the circuit size can grow exponentially over the BDD inputs count.

Since BDD is a special case of IFD, IFDs provide more universal and compact representation of Boolean functions against BDDs. Having an IFD, the traversal of its nodes and the substitution of each node by a cascade of reversible gates produce an appropriate reversible circuit. The preferable cascade of gates depends on the successors of the node. Moreover, choosing the appropriate cascade depends on the fan-out of successor nodes.

Table 1 provides the cascades of reversible gates for all possible scenarios of IFD node. The first row of the table describes a node that represents function  $f = ifd(d, g, h)$ . When both nodes  $g$  and  $h$  have the fan-out of 1, we may overwrite the inputs of corresponding circuit lines and realize node  $f$  by cascade a) or cascade b). The first cascade consists of one controlled CNOT gate and one Toffoli gate, while the second cascade consists of NOT, CNOT and Toffoli gates. When only node  $h$  has the fan-out of 1, we realize node  $f$  by cascade c) with one ancillary line, two CNOT gates and one Toffoli gate. When only  $g$  has the fan-out of 1, we realize node  $f$  by cascade d) of one ancillary line, two CNOT and one Toffoli gates. When nodes  $g$  and  $h$  have the fan-out larger than 1, the overwriting of all cascade inputs is forbidden, therefore we introduce an ancillary line 0 and realize node  $f$  by the most expensive cascade e) consisting of one CNOT and two Toffoli gates.

The second row of Table 1 describes a node that models function  $f = xor(d, g)$ . In the node view, a dash line represents complementation. When node  $g$  or node  $d$  has the fan-out of 1, we may realize node  $f$  by cascade a) or cascade b) respectively of only one controlled CNOT gate. When both  $d$  and  $g$  have the fan-out larger than 1, we introduce ancillary line 0 and realize  $xor$  by the cascade c) consisting of two CNOT gates.

The third row describes a node that represents function  $f = or(d, h)$ . When both nodes  $g$  and  $d$  have the fan-out of 1, we may realize node  $f$  by cascade a) of two NOT and one Toffoli gates. When the fan-out of both nodes  $g$  and  $d$  exceeds 1, we can realize node  $f$  by cascade b) or cascade c). Cascade b) includes four NOT gates and one Toffoli gate, while cascade c) includes two controlled CNOT gates and one Toffoli gate. Cascades a), b) and c) use one ancillary line.

The fourth row describes a node that models function  $f = or(-d, g)$ . When node  $g$  has the fan-out of 1, we may realize node  $f$  by cascade a) of one NOT gate and one Toffoli gate. When the fan-out of both nodes  $g$  and  $d$  is equal or larger than 1, we can realize node  $f$  by cascade b) of two NOT and one Toffoli gates. When the fan-out of both nodes  $g$  and  $d$  exceeds 1, we can use cascade c) of one NOT, one CNOT and one Toffoli gates. All cascades a), b) and c) use an ancillary line.

The fifth row describes a node that represents function  $f = and(d, g)$ . At any fan-out of nodes  $d$  and  $g$ , node  $f$  can be realized by cascade a) of one ancillary line and one Toffoli gate.

The sixth row describes a node that models function  $f = and(-d, h)$ . When node  $d$  has the fan-out of 1, we may realize node  $f$  by cascade a) of one NOT gate and one Toffoli gate. When the fan-out of both nodes  $d$  and  $g$  is equal or larger than 1, we can realize node  $f$  by cascade b) of two NOT and one Toffoli gates. Cascades a) and b) use an ancillary line.

Similar cascades realize IFD nodes with complement edges. Again, we construct cascades for all possible scenarios of an IFD node.

Table 1. Mapping IFD nodes to cascades of reversible gates

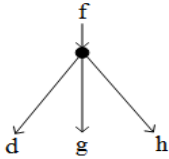
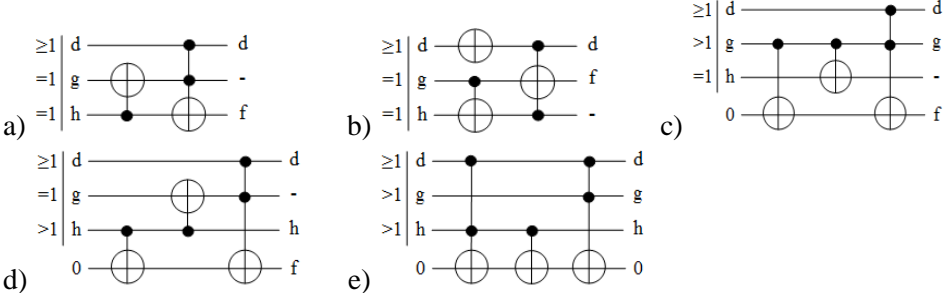
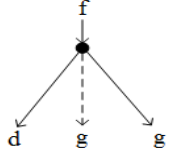
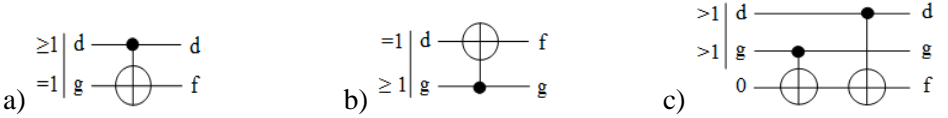
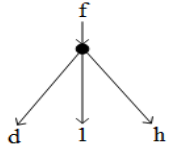
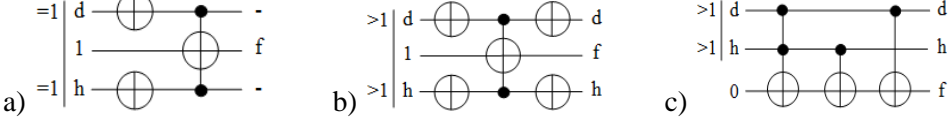
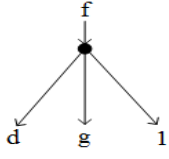
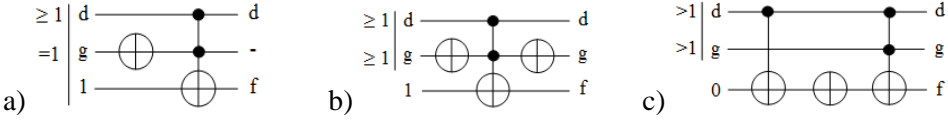
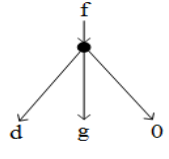
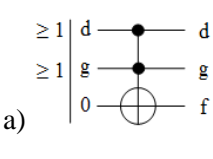
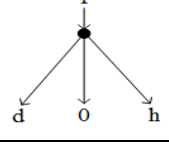
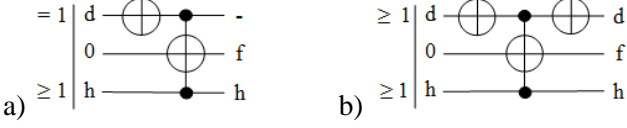
| N | IFD node  | Competitive cascades of reversible gates   |
|---|---|--|
| 1 |    |    |
| 2 |    |    |
| 3 |    |    |
| 4 |   |  |
| 5 |  |   |
| 6 |  |  |

Figure 3 illustrates the technique of transforming a reduced ordered BDD (representing an example Boolean function, Figure 3a) to a functionally equivalent IFD, and further mapping the IFD to a reversible circuit realization. The BDD consists of six nonterminal nodes labeled by variables  $x_0, x_1, x_2$  and  $x_3$ , and two terminal nodes labeled by 0 and 1. The direct transition from ROBDD to IFD yields the if-decision diagram that consists of five nonterminal nodes. Additionally, we have developed an optimization technique that produces the IFD depicted in Figure 3b. The IFD consists of three nonterminal nodes, two nodes less. Figure 3c depicts the reversible circuit obtained during the traversal of the optimized IFD and mapping its nodes to appropriate cascades of reversible gates. The vertical dash lines indicate the gates that realize each of three nonterminal nodes of the IFD. The circuit consists of five lines, two NOT gates, two CNOT gates, and two Toffoli gates. If we have generated a circuit directly from the ROBDD, the circuit size would be larger than those one shown in Figure 3c.

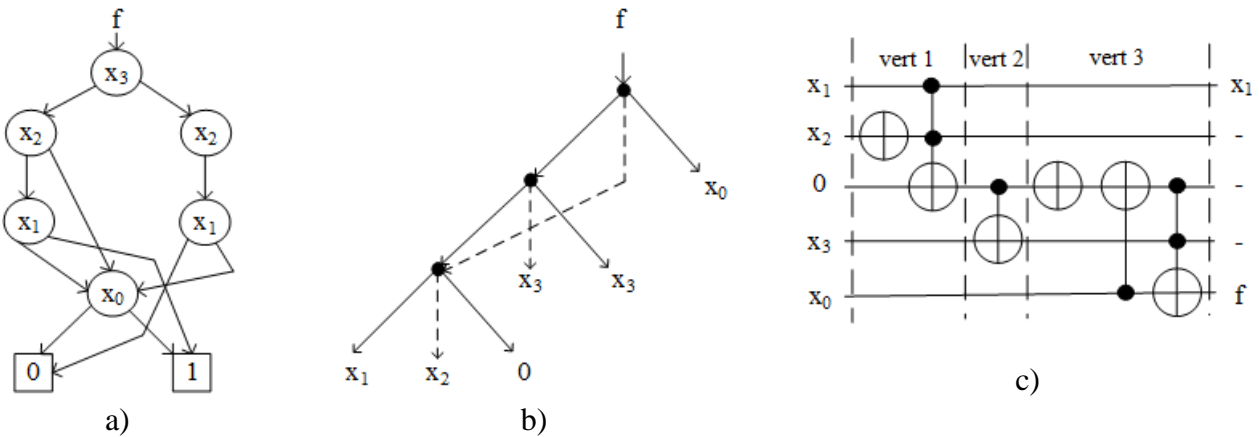


Figure 3 – An example of transforming a) ROBDD to b) IFD and mapping IFD to c) reversible circuit

*Conclusion.* Recently reversible and quantum computation has become an intensively studied topic. All operations in quantum circuits are reversible in nature. The literature describes several approaches for synthesis of reversible circuits. The binary decision diagram based synthesis has shown the improvement of reversible circuits parameters. In past years, we have done research that extends the binary decision diagrams to if-decision diagrams, which increase the modelling power of logic functions. In this paper, we have developed an approach for modelling the reversible circuits by if-decision diagrams. We have introduced rules, which put cascades of reversible gates in accordance to if-diagram nodes of various types. We have given an example of transforming a binary decision diagram to an if-decision diagram, and an example of further mapping the if-decision diagram to a small size reversible circuit.

### References

1. Lee, C.Y. Representation of Switching Circuits by Binary-Decision Programs / C.Y. Lee // Bell Systems Technical Journal, 1959, Vol. 38, No 4, pp. 985-999.
2. Bryant, R. Graph-based algorithms for Boolean function manipulation / R. Bryant, // IEEE Trans. on Comp. **35** (1986), pp. 677–691.
3. Prihozhy, A.A. If-Diagrams: Theory and Application / A.A. Prihozhy // Proc. 7<sup>th</sup> Int. Workshop PATMOS'97. – UCL, Belgium, 1997. – P. 369 – 378.
4. Prihozhy, A.A. Parallel Computing with If-Decision-Diagrams / A.A. Prihozhy, P.U. Brancevich // Proc. Int. Conference PARELEC'98. – Poland, Technical University of Bialystok. – 1998. – P. 179–184.
5. Прихожий А.А. Частично определенные логические системы и алгоритмы / А.А. Прихожий / Минск, БНТУ. – 2013. – 343 с.
6. Прихожий А.А. Обобщение разложения Шеннона для частично определенных функций: теория и применение / А.А. Прихожий / Системный анализ и прикладная информатика. – 2013, № 1-2. – С. 6-11.
7. Прихожий, А.А. Новые разложения булевых функций по операции исключающее или в системах логического проектирования / А.А. Прихожий / Системный анализ и прикладная информатика. – 2014, № 1-3. – С. 9-16.
8. Prihozhy, A.A. Analysis, transformation and optimization for high performance parallel computing / A.A. Prihozhy // Minsk, BNTU. – 2019. – 229 p.
9. Bennett, C.H. Logical reversibility of computation / C.H. Bennett // IBM J. Res. Dev 17 (1973), pp. 525–532.

10. Toffoli, T. Reversible computing, / T. Toffoli // in: W. de Bakker and J. van Leeuwen, editors Automata, Languages and Programming, Springer, 1980, p. 632.
11. Fredkin, E.F. Conservative logic / E. F. Fredkin, T. Toffoli // International Journal of Theoretical Physics 21 (1982), pp. 219–253.
12. Barenco, A. Elementary gates for quantum computation / A. Barenco, C. H. Bennett, R. Cleve, D. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, H. Weinfurter // The American Physical Society, vol. 52, 1995, p. 3457-3467.
13. Nielsen, M. Quantum Computation and Quantum Information / M. Nielsen, I. Chuang // Cambridge Univ. Press, 2000.
14. Wille R. Effect of BDD Optimization on Synthesis of Reversible and Quantum Logic / R. Wille, R. Drechsler // Electronic Notes in Theoretical Computer Science V. 253, 2010, pp. 57–70.
15. Sasanian, Z. Realizing Reversible Circuits Using a New Class of Quantum Gates / Z. Sasanian, R. Wille, D. M. Miller // DAC 2012, June 3-7, San Francisco, 2012, p. 36-41.
16. Drechsler, R. Reversible Circuits: Recent Accomplishments and Future Challenges for an Emerging Technology / R. Drechsler, R. Wille // In: Rahaman H., Chattopadhyay S., Chattopadhyay S. (eds) Progress in VLSI Design and Test. Lecture Notes in Computer Science. – Springer, 2012, vol 7373, p. 383-392.
17. Wille, R. From reversible logic to quantum circuits: Logic design for an emerging technology / R. Wille, A. Chattopadhyay, R. Drechsler // In Proceedings of the 2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS), Samos, Greece, 17–21 July 2016; p. 268–274.
18. Lukac, M. Minimization of quantum circuits using quantum operator forms / M. Lukac, M. Kameyama, M. Perkowski, P. Kerntopf // arXiv preprint arXiv:1701.01999, 2017.
19. Handique, M. An Extended approach for Mapping Reversible Circuits to Quantum Circuits using NCV- $|v1\rangle$  library / M. Handique, A. Sonka // Elsevier, Procedia Computer Science 125 (2018) 832–839.
20. Zulehner, A. Accuracy and Compactness in Decision Diagrams for Quantum Computation / A. Zulehner, P. Niemann, R. Drechsler, R. Wille. // Design, Automation and Test in Europe Conference – DATE, 2019, p.280-283.