International Journal of Computer Theory and Engineering, Vol. 2, No. 2 April, 2010
1793-8201

Use of Symmetric Functions Designed by QCA Gates for Next Generation IC

Pijush Kanti Bhattacharjee

Abstract—**Different logic gates like MV, NOT, AOI, NNI etc under QCA nanotechnology are introduced. NNI gate is highly effective regarding space and speed consideration. Any Boolean functions, confined in thirteen number standard Boolean functions, are synthesized by MV and NNI gates or simply NNI gates alone, eliminating inverter (NOT) gate. A new method for realizing symmetric functions in binary reversible logic is introduced. My procedure synthesizes for a more efficient realization both the unate and non unate symmetric functions with little "garbage". The method is used in a classical logic. I use only 2-inputs and 2-outputs AND-NAND and OR-NOR cells designed on QCA technology. It admits a recursive construction. Ultimately I achieve a general equation for the minimum number of gates required to an arbitrary number of input variables, causing synthesis of symmetric function. These symmetric functions consist of different Boolean functions with adder circuits functions, subsequently all other combinational circuits like subtractor, multiplier, divisor, multiplexer, encoder, comparator etc can be designed by the adder circuit only which ensures next generation IC design. It provides a significant reduction in hardware cost and switching delay compared to other existing techniques.**

Index Terms—**Majority Voter (MV) gate, And-Or-Inverter (AOI) gate, Nand-Nor-Inverter (NNI) gate, 2-Inputs and 2-Outputs AND-NAND (A-NA) gate, 2-Inputs and 2-Outputs OR-NOR (O-NO) gate, Symmetric Function.**

I. Introduction

Reversible Quantum Computers (QCs) will be created for improvement in speed, reducing cost and space etc [1]-[6], [11]. Reversible are the circuits or the gates that have the same umber of inputs and outputs and have one-to-one mappings between vectors of inputs and outputs; thus the vector of the input states can be uniquely reconstructed from the vector of the output states. A symmetric function [11]-[14] means a Boolean function invariant to the permutation of any of its input variables. Also every Boolean function can be made symmetric by repeating its input variables. The application of symmetric function is in enormous field of the processor (IC) design and in higher mathematics problem solving. Any sort of Boolean functions can be simplified by symmetric functions. In practical oriented problem like design of complex circuitry, complex control systems with different parameters or variables, solving more variables consisting of functions etc may be done by proper application of symmetric function.

Fredkin Gate (FG) is a fundamental concept in reversible and quantum computing, the base of "realization-related" papers. It has been introduced by Ed Fredkin and Tomasso Toffoli in 1982 [12], [13].

1) Every Boolean function can be build from (binary) Fredkin Gates (FGs), such that it has three inputs A, B and C and three outputs P, Q and R like

P = A
Q = if A then C else B
R = if A then B else C

2) Feyman Gates is "Controlled NOT" or "quantum XOR", such gates have two inputs A and B and two outputs P and Q, they are called linear likewise

P = A
Q = A EXOR B

Thus synthesis of symmetric Boolean functions is achieved by using a simple recursive arrangement of 2-inputs, 2-outputs AND-NAND (A-NA) and OR-NOR (O-NO) logic gates realizable with QCA (Quantum Dot Cellular Automata) technology. It eliminates the use of NOT or inverter gate which is more costly and spacious as well as introduced delay in realizing the symmetric function. By this approach, synthesize symmetric functions can be implemented for an arbitrary number of input variables. A general equation for estimating the number of gates (AND-NAND, OR-NOR etc) required for symmetric function realization is invented. Further, the proposed technique is applicable for any symmetric functions either unate or non unate functions results a logic design with less hardware cost which drastically reduces the "garbage" compared to the other existing techniques. In unate symmetric functions, all variables or literals are either non complemented form, called unate positive, or complemented form, called unate negative function.

e. g. $X_1X_2 + X_2X_3 + X_3X_1 \longrightarrow$ Unate positive function
$X_1' + X_2' + X_3' \longrightarrow$ Unate negative function

In non unate symmetric function, the variables are complemented or non complemented form i.e. imposing no restriction of the variable form,

e. g. $X_1'X_2'X_3 + X_1X_2'X_3' + X_1'X_2X_3'$

II. Quantum Dot Cellular Automata

QCA (Quantum-dot Cellular Automata) [1]-[10] devices encode and process binary information as charged arrays of

Pijush Kanti Bhattacharjee is an Assistant Professor in the Department of Electronics and Communication Engineering, Bengal Institute of Technology and Management, Santiniketan, P.O. Doranda, West Bengal, Pin-731236, India. He was an Ex Asssitant Director in the Department of Telecommunications (DoT), Government of India, India. He has possessed vast working experience in the field of Telecommunications including Mobile Communications, Image Processing, VLSI etc during last 29 years. He is a member of IACSIT, Singapore; CSTA, USA; IAEng, Hongkong. (phone: +91-33-25954148; email: pijushbhatta_6@hotmail.com)

charge coupled quantum dots. A quantum cell can be viewed as a set of four charge containers or dots positioned at the corners of a square, as shown in Fig. 1. It contains two extra mobile electrons. The electrons can quantum mechanically tunnel between dots but can not come out from the cell and are forced to settle at the corner positions due to coulomb interaction. Thus, there exists two equivalent energetically minimal arrangements for the electrons in a QCA cell (Fig. 1), i.e. the polarization P = +1 (representing logic 1) and P = –1 (representing logic 0).

In Fig. 1, a QCA cell and its binary logic are shown, the energetically position of the diagonal electrons identifies the binary logic 0 or 1. This phenomenon is useful in nano technology which affects high resolution fast electronic circuits. In this power consumption for changing the charge of electron is very much less compare to that of general charge carriers (hole-electron) electronic components.

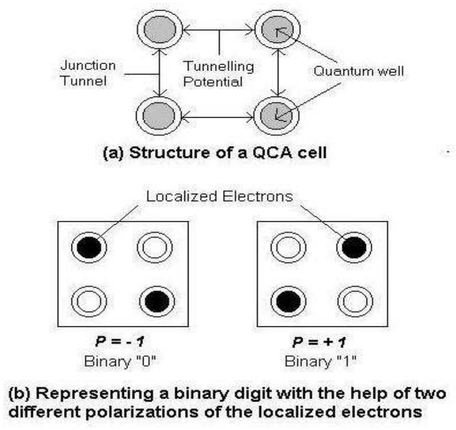

Junction Tunnel
Tunnelling Potential
Quantum well
(a) Structure of a QCA cell
Localized Electrons
P = -1
Binary "0"
P = +1
Binary "1"
(b) Representing a binary digit with the help of two different polarizations of the localized electrons

Fig. 1. A QCA cell and its binary logic

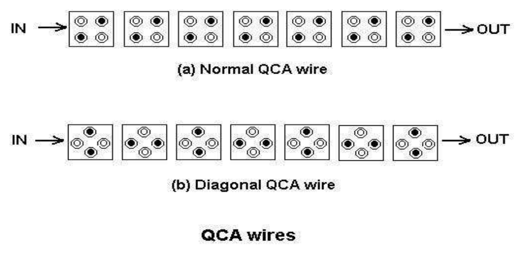


Fig. 2. Information propagating through QCA wires

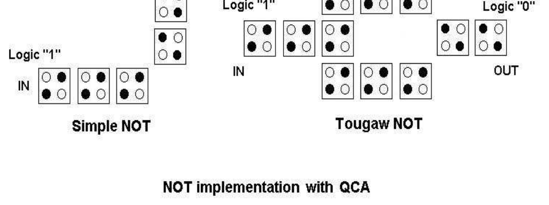
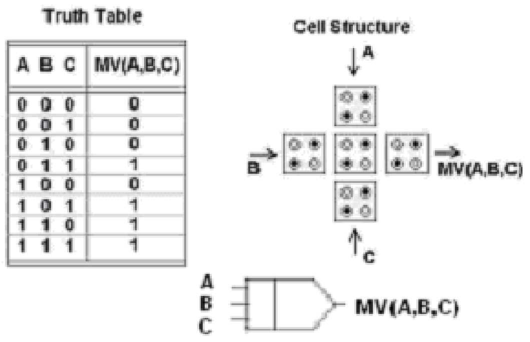


Fig. 3. Implementation of NOT with QCA Gate.

A QCA Cell with its binary logic creates a new direction in nano technology [1]-[9]. It requires minimum current or energy to change any state i.e. previous state. Thus, a minimum recurring cost is effective in this QCA gates which is highly applicable in super fast processors. Also power or heat dissipation, electro magnetic wave radiation etc are very much less in QCA based gates.



A	B	C	MV(A,B,C)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Fig. 4. QCA Majority Voter (MV) Gate

The basic QCA logic elements [1]-[8] include a QCA wires are shown in Fig. 2, QCA inverter or NOT gate in Fig. 3 and Majority Voter (MV) or Majority Gate (Maj) in Fig. 4. In diagonal (also called 45^0) wire, binary signal alternates polarization in successive cells. The QCA Majority Voter (MV) shown in Fig. 4 realizes MV(A, B, C) = Maj(A, B, C) = AB+BC+CA, outputs '1' if there are two or more 1s in an input pattern. The classical AND and OR gates can be realized with the majority gate by fixing an input as 0 and 1 respectively. The majority gate is not a universal gate. It can not realize the logical NOT operation. The functionally complete set is {MV, NOT}. Therefore, the designers have to use separate QCA cell arrangement for realization of the logical NOT. The 5-input (A, B, C, D and E) And-Or-Inverter (AOI) gate [7] with embedded AND, OR and INV functions has been proposed to provide the universal gate function. However the AOI suffers from the limitation of proper separation of input and output binary wires- that is, in fixing the distances d1, d2 and d3 of Fig. 5.

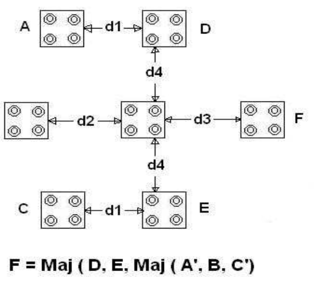


Fig. 5. QCA And-Or-Inverter (AOI) Gate.

And-Or-Inverter is not a universal gate. All Boolean functions can not be designed alone A-O-I gate. Another drawback of A-O-I gate is that it requires more space and the A-O-I gates are more complex nature comparing to that of MV or NOT gates.

Thus to implement MV with NOT functions, a new gate called Nand-Nor-Inverter (NNI) [1]-[10] is constructed, where NNI(A, B, C) = MV(A', B, C') = A'B+BC'+C'A'. It is shown in Fig. 7. The NNI gate is a universal gate and can be employed for realizing versatile logic functions. It proves to be as effective as the AOI (And-Or-Inverter) gate and requires lesser overhead, for setting the variables, than that of an AOI, while realizing the basic logic gates. NNI gate ensures very less space comparing to that of the other gates like MV, AOI and inverter (NOT) gate.

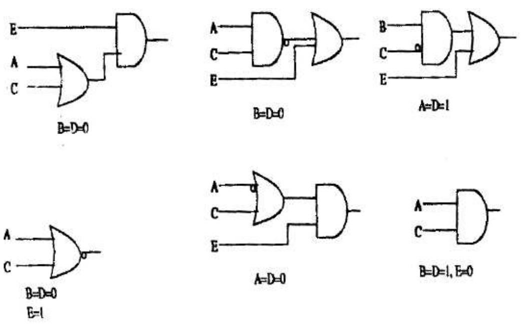
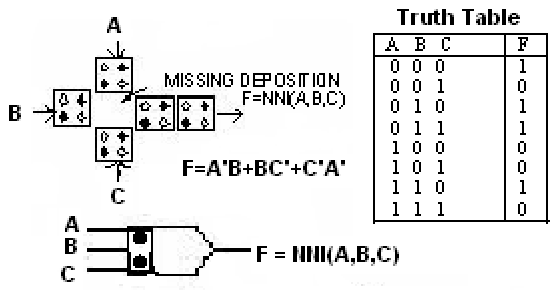


Fig. 6. Various logic functions realized with AOI Gate



Truth Table

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Fig. 7. QCA Nand-Nor-Inverter (NNI) Gate

It is shown in Table-I that any Boolean function is realized either by combination of MV and NNI gates or by NNI gates alone.

A. Boolean Functions Synthesis by QCA gates

All Boolean functions which are confined or simplified within thirteen number standard Boolean functions [7], [10]-[15] are realized in two level implementation by different QCA gates like F = X(f_i, f_i, f_i), where F is a Boolean function, X is a MV or NNI gate, f_i is MV(A, B, C) or NNI (A, B, C) or A or 0 or 1, in which A, B, C are in complemented or non complemented literals or variables. These standard Boolean functions are synthesized either by MV and NNI gates or simply NNI gates only, which is elaborately explained in Table-I. So, it eliminates the use of inverter or NOT gate which is provided either by QCA technology or conventional CMOS in VLSI. This inverter gate takes more chip area.

III. Synthesis of Symmetric Function

The symmetric functions are paying attention to the researchers in the field of VLSI and nanotechnology design, especially for logic synthesis. A number of synthesis technique for Boolean symmetric functions are reported [12]-[15]. I represent a simple cost effective synthesis with QCA gates for future generation digital design.

A. Symmetric Function

A switching function f(x_1,x_2,…………,x_n) is called totally symmetric with respect to the variables x_1, x_2, x_3,……….,x_n, [11]-[15], if it is invariant under any permutation of the variables.

Table I. Standard Boolean Functions Realized by Quantum Gates

Standard Functions with Serial Number	Realization of Functions using different Logic gates and finally synthesized without complemented (NOT) variable or gate.

1) F1 = AB'C	F1 = MV(MV(0, A, B'), C, 0)
	F1 = MV(NNI(1, A, B), C, 0)
	F1 = NNI(NNI(A, B, 0), C, 1)
	F1 = NNI(NNI(C, B, 0), NNI(B, A, C), NNI(B, A, 0))

2) F2 = AB	F2 = MV(A, 0, B)
	F2 = MV(MV(A', 0, 1), MV(A, B, 0), MV(A, 1, 0))
	F2 = MV(NNI(A, 0, 0), MV(A, B, 0), MV(A, 1, 0))
	F2 = NNI(NNI(A, 1, B), NNI(A, 0, 0), NNI(A, 0, 0))

3) F3 = A'BC + A'B'C'	F3 = MV(MV(A', 1, 0), MV(B', C', 0), MV(B, C, 0))
	F3 = MV(NNI(A, 1, 1), NNI(B, 0, C), MV(B, C, 0))
	F3 = NNI(NNI(0, A, 1), NNI(B, 0, C), NNI(B, 1, C))

4) F4 = A'BC + AB'C'	F4 = MV(MV(A, B, 1), MV(B', C', 0), MV(A', C, 0))
	F4 = MV(NNI(B, 0, C), MV(A, B, 1), NNI(A, C, 1))
	F4 = NNI(MV(B, C, 1), MV(A, B, 1), NNI(C, A, 1))
	F4 = NNI(NNI(A, 0, B), NNI(B, 0, C), NNI(C, A, 0))

5) F5 = A'B + BC'	F5 = MV(0, MV(1, A', C'), B)
	F5 = MV(0, NNI(A, 1, C), B)
	F5 = MV(MV(A', B, C'), MV(B, 1, 0), MV(A, B', 0))
	F5 = NNI(NNI(B, 0, 0), NNI(A, B, C), NNI(A, B, 0))

6) F6 = AB' + A'BC	F6 = MV(MV(A, B, C), MV(A, B', 0), MV(A', B', 1))
	F6 = MV(MV(A, B, C), NNI(B, A, 1), NNI(A, 1, B))
	F6 = NNI(NNI(A, B, 0), MV(A, B, C), MV(A, B, 0))
	F6 = NNI(NNI(A, 0, B), NNI(B, A, 1), NNI(C, A, 0))

7) F7 = A'BC + ABC' + A'B'C'	F7 = MV(MV(A', C, 0), MV(A', B, C'), MV(A, B', C'))
	F7 = MV(NNI(A, C, 1), NNI(A, B, C), NNI(B, A, C))
	F7 = MV(NNI(A, B, C), MV(A, B, C), NNI(B, 0, C))
	F7 = MV(MV(A', B, C'), MV(A, B', 1), MV(A', BC, 0))
	F7 = NNI(NNI(A, B, 1), NNI(A, B, C), NNI(BC, A, 0))

8) F8 = A	F8 = MV(A, 0, 1)
	F8 = NNI(0, A, 1)

9) F9 = AB + BC + CA	F9 = MV(A, B, C)
	F9 = MV(MV(A, B, 1), MV(C, 1, 0), MV(A, B, 0))
	F9 = NNI(NNI(A, 0, B), NNI(1, C, 0), NNI(A, 1, B))

10) F10 = A'B + B'C	F10 = MV(MV(A', B, 0), 1, MV(B', C, 0))
	F10 = MV(NNI(A, B, 1), 1, NNI(B, C, 1))
	F10 = NNI(NNI(B, A, 0), 1, NNI(C, B, 0))
	F10 = NNI(NNI(B, A, C), NNI(A, 1, B), NNI(C, B, 0))

11) F11 = A'B + BC + AB'C'	F11 = MV(MV(A, B, 1), MV(A', B, C), MV(B', C', 0))
	F11 = NNI(NNI(B, A, C), MV(A, B, 1), MV(B, C, 1))
	F11 = NNI(NNI(A, 0, B), NNI(B, 0, C), NNI(B, A, C))

12) F12 = AB + A' B'	F12 = MV(MV(A, B, 0), MV(A', B', 0), 1)
	F12 = MV(MV(A, B, 0), NNI(A, 0, B), 1)
	F12 = NNI(MV(A, B, 1), MV(A, B, 0), 0)
	F12 = NNI(NNI(B, A, 1), NNI(A, 0, B), NNI(A, 1, B))

13) F13 = ABC'+AB'C +A'BC+A'B' C'	F13 = MV(MV(A', B, C'), MV(A, B, C), MV(B', 0, 1))
	F13 = MV(MV(A', B, C'), MV(A, B, C), B')
	F13 = MV(NNI(A, B, C), MV(A, B, C), NNI(B, 0, 0))
	F13 = MV(MV(A', B, C'), MV(A, B', 1), MV(A'BC, AB'C, 1))
	F13 = MV(NNI(A', B', 0), NNI(A, B, C), MV(MV(A', BC, 0), 1, MV(B', AC, 0)))
	F13 = NNI(NNI(A, B, 1), NNI(A, B, C), NNI(NNI(A, BC, 1), 0, NNI(B, AC, 1)))

Total symmetry can be specified by a set of integers (called a numbers) A = $(a_i,\ldots,a_j,\ldots,a_k)$ where $A \subset (0,1,2,\ldots,n)$; all the vertices with weight $w \in A$ will appear as true minterms in the function.

A n-variable symmetric function is denoted as $S^n(a_i,\ldots,a_j,\ldots,a_k)$. A symmetric function is called consecutive, if the set A consists of only consecutive integers $(a_l, a_{l+1},\ldots,a_r)$. It is expressed as $S^n(a_l - a_r)$ where l<r.

For n variables, there can be $(2^{n+1} - 2)$ different symmetric functions (excluding constant functions 0 and 1). Each totally symmetric functions, $S^n(A) = S^n(A_1) + S^n(A_2) + \ldots + S^n(A_m)$, where m is minimum, $\forall$ i, j, $1 \leq i, j \leq m$, $A_i \cap A_j = \varnothing$ (null), and $i \neq j$. Thus, a symmetric function may be broken in lower order consecutive symmetric functions keeping all condition same.

IV. Methodology And Implementation

I discuss the proposed method of symmetric function synthesis and its implementation in this section [15]. I use two type of gates e.g. AND-NAND (A-NA) and OR-NOR (O-NO) gates. The output of an AND-NAND gate is AND and NAND functions of the inputs. Similarly, the output of an OR-NOR gate is OR and NOR functions of the inputs. The gate is having two inputs and two outputs. These two gates are the simplest gates of QCA cells under nano technology. These AND-NAND (A-NA), OR-NOR (O-NO) gates can be designed by QCA MV and NOT (Inverter) gates by fixing an input as 0 and 1 of MV gates respectively and output of MV gate is fed to NOT gates for NAND or NOR functions. I eliminate the use of NOT or inverter gate which takes more area, costlier, time and delay consuming.

In the synthesis of symmetric functions, I get all the combinations of symmetric functions for a particular set of input variables including unate positive and unate negative symmetric functions. This becomes a generalized solution for the symmetric functions realization for any number of input variables.

A. Method of Synthesis Symmetric Function

First of all, I take 2-inputs and 2-outputs AND-NAND, OR-NOR gates as shown in Fig. 8.

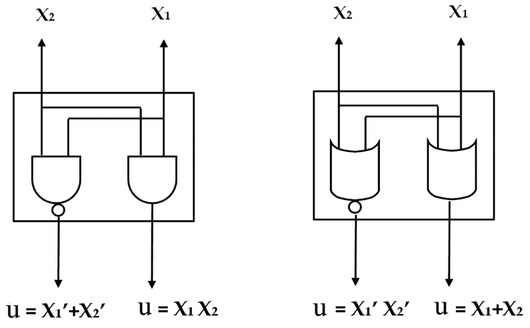


Fig. 8. 2-inputs and 2-outputs AND-NAND Gate (left), 2-inputs and 2-outputs OR-NOR Gate Circuits (right).

Here, AND-NAND gate and OR-NOR gate are two type of QCA gates which are designed on nano technology process. It requires less overhead area and at the same time ensures best efficiency in speed.

B. 2-Inputs Symmetric Function Synthesis

For 2-input variables, I take one AND(A)-NAND(NA) gate and two OR(O)-NOR(NO) gates to get all the symmetric functions composed by two variables. The number of symmetric functions consists of two variables as per $(2^{n+1} - 2)$ formula are 6 followings:

1) AB = u1 = $S^2(2)$
2) A'+B' = u2 = $S^2(0,1)$
3) A+B = u3 = $S^2(1,2)$
4) A'B' = u4 = $S^2(0)$
5) AB+A'B' = u5 = $S^2(0,2)$
6) AB'+A'B = u6 = $S^2(1)$

The realization of symmetric functions by AND(A)-NAND(NA) and OR(O)-NOR(NO) gates is shown in Fig. 9. These QCA based gates are taking minimum space area as well as minimum power consumed to formulate any type of symmetric outputs. Therefore, these symmetric functions realization integrated circuits may be manufactured at a minimum cost surprising all other type of invented ICs, especially when it is manufactured at large scale. This type of symmetric functions synthesis may be implemented in all other complex circuit solution also.

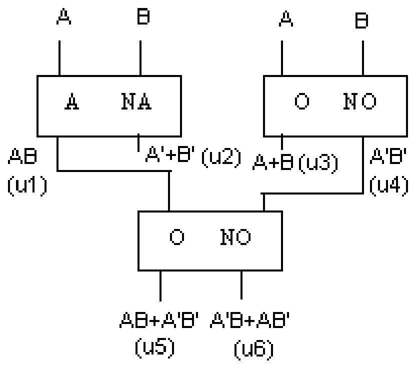


Fig. 9. Symmetric functions circuit diagram for 2-Input variables

For two input variables, I require total three gates, one AND-NAND gate and two OR-NOR gates and no output from any gate is garbage output i.e. not belonging to the symmetric functions. Therefore, this construction of 2-input symmetric functions is the simplest form and ensures fastest operation. Since this circuit is not having any extra output i.e. garbage output, it ultimately curtails the cost of this 2-input symmetric function IC design. Here half adder Boolean function for sum is u6 i.e. $S^2(1)$ and carry is u1 i.e. $S^2(2)$.

C. 3-Inputs Symmetric Function Synthesis

For 3-input variables, I require additional 8 gates to get all the symmetric functions (14 numbers). So, total 11 number of gates are necessary if starting from 2-input variables. The numbers of symmetric functions consisting of three input variables are followings.

1) ABC = u7 = S3(3)
2) A'+B'+C' = u8 = S3(0,1,2)
3) A+B+C = u9 = S3(1,2,3)
4) A'B'C' = u10 = S3(0)
5) ABC+A'B'C' = u11 = S3(0,3)
6) A'B+AB'+A'C+AC'+B'C+BC' = u12 = S3(1,2)

7) AB+BC+CA = u13 = S3(2,3)
8) A'B'+B'C'+C'A' = u14 = S3(0,1)
9) A'B'+B'C'+C'A'+ABC = u15 = S3(0,1,3)
10) A'BC+AB'C+ABC' = u16 = S3(2)
11) AB+BC+CA+A'B'C' = u17 = S3(0,2,3)
12) AB'C'+A'BC'+A'B'C = u18 = S3(1)
13) AB'C'+A'BC'+A'B'C+ABC = u19 = S3(1,3)
14) A'BC+ABC'+AB'C+A'B'C' = u20 = S3(0,2)

In Fig. 10, total 2 numbers of AND-NAND gates and 6 numbers of OR-NOR gates are required for construction of all the symmetric functions (14 numbers) by 3-input variables and hence total 8 gates are required. Two outputs from the gate are garbage outputs i.e. not a symmetric function. Out of 2-garbage outputs, one output is not associated to act as further input. So, one output is completely garbage output. The other garbage output is feeding as input to next stage OR-NOR gate. The efficiency of the circuit regarding power consumption and speed is the maximum.

Here Full adder Boolean functions for sum and carry are u19 i.e. $S^3(1,3)$ and u13 i.e. $S^3(2,3)$ respectively. Therefore the adder circuit outputs are derived from this 3-input symmetric function synthesis.

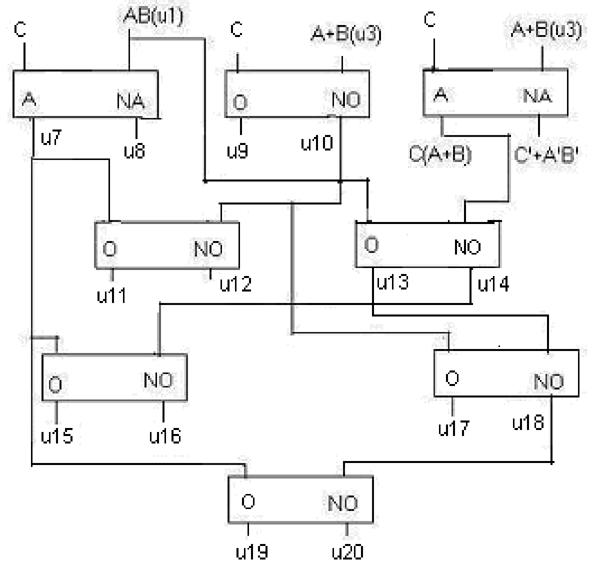


Fig. 10. Symmetric functions circuit diagram for 3-Input variables

D. 4-Inputs Symmetric Function Synthesis

Similarly, I realize the symmetric functions for 4-input variables. Total numbers of symmetric functions for 4 input variables (30) are shown below:

1) A+B+C+D = u21 = $S^4(1,2,3,4)$
2) A'B'C'D' = u22 = $S^4(0)$
3) ABCD = u23 = $S^4(4)$
4) A'+B'+C'+D' = u24 = $S^4(0,1,2,3)$
5) ABCD+A'B'C'D' = u25 = $S^4(0,4)$
6) A'B+AB'+B'C+BC'+A'C+AC'+A'D+AD'+B'D+BD'+C'D + CD' = u26 = $S^4(1,2,3)$
7) ABC+ABD+BCD+ACD = u27 = $S^4(3,4)$
8) A'B'+B'C'+C'A'+A'D'+B'D'+C'D' = u28 = $S^4(0,1,2)$
9) A'B'C'+A'B'D'+A'C'D'+B'C'D'= u29 = $S^4(0,1)$
10) AB+BC+CA+AD+BD+CD= u30 = $S^4(2,3,4)$
11) A'B'+B'C'+C'A'+A'D'+B'D'+C'D'+ABCD=u31= $S^4(0,1,2,4)$
12) A'BCD+AB'CD+ABC'D+ABCD' = u32 = $S^4(3)$
13) ABC'D'+A'BCD'+A'BC'D+AB'CD'+AB'C'D+A'B'CD= u33 = $S^4(2)$
14) ABC+ABD+ACD+BCD+A'B'C'+B'C'D'+A'C'D'+A'B'D' = u34 = $S^4(0,3,4,1)$
15) ABC+ABD+BCD+ACD+A'B'C'D' = u35 = $S^4(0,3,4)$
16) AB'C'+AC'D'+A'B'D+A'C'D+A'CD'+BC'D'+AB'D'+ A'B'C+B'C'D+B'CD'+A'BD'+A'BC'=u36 = $S^4(1,2)$
17) AB+BC+CA+AD+BD+CD+A'B'C'D'= u37 = $S^4(0,2,3,4)$
18) AB'C'D'+A'BC'D'+A'B'CD'+A'B'C'D = u38 = $S^4(1)$
19) A'B'C'+A'B'D'+A'C'D'+B'C'D'+ABCD =u39 =$S^4(0,1,4)$
20) A'BC+A'CD+A'BD+AB'C+B'CD+AB'D+ABC'+BC'D+ AC'D+ABD'+BCD'+ACD' = u40 = $S^4(2,3)$
21) A'BCD+AB'CD+ABC'D+ABCD'+A'B'C'D'=u41= $S^4(0,3)$
22) AB'C'+AC'D'+A'B'D+A'C'D+A'CD'+BC'D'+AB'D'+A'B 'C+B'C'D+B'CD'+A'BD'+A'BC'+ABCD=u42= $S^4(1,2,4)$
23) ABC'D'+A'BCD'+A'BC'D+AB'CD'+AB'C'D+ A'B'CD + A'B'C'D' = u43 = $S^4(0,2)$
24) ABC+ABD+BCD+ACD+AB'C'D'+A'BC'D'+A'B'CD'+ A'B'C'D = u44 = $S^4(1,3,4)$
25) A'BC+A'CD+A'BD+AB'C+B'CD+AB'D+ABC'+BC'D+ AC'D+ABD'+BCD'+ACD'+A'B'C'D' = u45 = $S^4(0,2,3)$
26) AB'C'D'+A'BC'D'+A'B'CD'+A'B'C'D+ABCD=u46= $S^4(1,4)$
27) ABC'D'+A'BCD'+A'BC'D+AB'CD'+AB'C'D+ A'B'CD + ABCD = u47 = $S^4(2,4)$
28) A'B'C'+A'B'D'+A'C'D'+B'C'D'+A'BCD+AB'CD+ABC'D +ABCD' = u48 = $S^4(0,1,3)$
29) ABC'D'+A'BCD'+A'BC'D+AB'CD'+AB'C'D+A'B'CD+ ABCD+A'B'C'D' = u49 = $S^4(0,2,4)$
30) A'BCD+AB'CD+ABC'D+ABCD'+A'B'C'D+A'B'CD'+A' BC'D'+AB'C'D' = u50 = $S^4(1,3)$

I require total 17 gates, out of which total AND-NAND gates are 3 numbers and OR-NOR gates are 14 numbers for 4 input variables. Fig. 11 shows the circuit diagram of the realized symmetric functions by taking 4-input variables and Table-II indicates the comparison of all the parameters of symmetric functions realization in a nutshell.

For adding 4-inputs binary variable, the sum is obtained as u50 or $S^4(1,3)$ and the carry is obtained as u27 or $S^4(3,4)$ in symmetric function synthesis. Generally sum of the adder circuit is appearing at output of the last gate.

V. General Equation of Symmetric Function Sysnthesis for Any Arbitrary Number of Input Variables

Total numbers of gates are required for two variables are 3, for three variables are 8, for four variables 17 etc. Thus a general equation for total number of gates requiring is framed. For n number of input variables, I have $(2^{n+1} – 2)$ numbers of symmetric functions, in this case total $(2^n + n – 3)$ number of gates are required. Similarly, for 5 input variables to realize all the 62 numbers symmetric functions, at least 34 numbers of total gates are required, accordingly AND-NAND, OR-NOR gates are distributed.

Therefore, maximum optimization is done to have all the symmetric functions for an arbitrary number of input variables (say n variables) by AND-NAND and OR-NOR gates on QCA based design principle. I achieve a general equation for requiring the total number of gates (like AND-NAND, OR-NOR gates) to n number of input variables as ($2^n + n - 3$) number of gates. This equation is explained and modified as below:

If X is the total number of gates requiring for n number of input variables comprising with ($2^{n+1} - 2$) number of symmetric functions,

$$X = (2^n + n - 3)$$

or,

$$X = [\frac{\text{Total No of n-variable Symmetric Function}}{2} + (\text{n-2})]$$

or,

$$X = [\frac{\text{Total No of n-variable Symmetric Function}}{2} + (\text{Number of garbage output})]$$

Here, (n – 2) is the garbage output. The garbage output is neither a symmetric function nor fed as further input to the gates. Thus, the garbage output can not be used further. Symmetric functions realized for 2 input variables, garbage output is 0; for 3 inputs, garbage output 1; for 4 inputs garbage outputs 2; for 5 inputs garbage outputs 3 etc, so garbage outputs are in AP series, which is clearly indicated in Table-II.

Again, $X = 2^n + n - 3 = (n - 1) + (2^n - 2)$

or, X = Number of AND-NAND gates + Number of OR-NOR gates

I require at least total (n − 1) number of AND-NAND gates and total ($2^n - 2$) number of OR-NOR gates to have all the symmetric functions realized by n-input variables. The detailed comparison of the parameters for symmetric function synthesis is shown in Table-II.

These symmetric functions yield adder circuit functions in addition to all other different kind of Boolean functions. All other combinational circuits like subtractor, multiplier, divisor, multiplexer, encoder, comparator etc can be designed by the adder circuit only. Thus it enables to design next generation IC with higher capacity and processing speed as well as less power and space consumption.

VI. Conclusions

Although symmetric functions are difficult to realize practically with the minimum number of gates, my proposed design with adoption of QCA technology enlightens a new direction in the field of symmetric function synthesis. Ultimately a general solution for minimum number of QCA gates (AND-NAND and OR-NOR) requiring is invented, identifying the "garbage" output. Thus QCA based design of symmetric functions are simulated by AND-NAND, OR-NOR gates and compared with the results of conventional CMOS based technique. These symmetric functions provide different kind of Bollean functions or combinational circuits with high resolution for next generation IC design. Moore's Law is satisfied in the coming age by implementation of symmetric function designed by these special QCA gates (AND-NAND, OR-NOR). It is also observed that the implementation of symmetric functions with QCA design is more advantages than that of CMOS design. In all aspects, this QCA based design symmetric functions are highly suitable for higher growth of IC processors which can afford as a precious element in the coming generation.

References

[1] C. S. Lent, P. D. Taugaw, W. Porod and G. H. Berstein, "Quantum Cellular Automata", *Nanotechnology*, vol. 4, no. 1, January 1993, pp 49-57.

[2] Z. Kohavi, Switching and Finite Automata Theory, Tata McGraw Hill Pub Company Ltd, New Delhi, 2nd Edition, 2007.

[3] I. Amlani, A. O. Orlov, G. Toth, C. S. Lent, G. H. Bernstein and G. L. Sinder, "Digital Logic Gate using Quantum Dot Cellular Automata", Science, vol. 284, no. 5412, April 1999, pp. 289-291.

[4] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information, 1st Edition, Cambridge University Press, New Delhi, 2006.

[5] A. O. Orlov, I. Amlani, G. H. Bernstein, C. S. Lent and G. L. Sinder,"Realization of a Functional Cell for Quantum Dot Cellular Automata", Science, vol. 277, no. 5328, August 1997, pp 928-930.

[6] C. S. Lent, P. D. Taugaw, "A Device Architecture for Computing with Quantum Dots", Proceedings IEEE, vol. 85, no. 4, April 1997, pp. 541-557

[7] M. Momenzadeh, M. B. Tahoori, J. Huang and F. Lombardi, "Characterization, Test and Logic Synthesis of AND-OR-INVERTER (AOI) Gate Design for QCA Implementation", IEEE Trans on Computer Aided Design of Integrated Circuits and Systems, vol. 24, no. 12, December, 2005, pp. 1881-1893.

[8] R. Zhang, K. Walus, W. Wang and G. A. Jullien, "A Method of Majority Logic Reduction for Quantum Cellular Automata", IEEE Trans on Nanotechnology, vol. 3, no. 4, Dec 2004, pp. 443-450.

[9] K. Walus, G. Schulhof, G. A. Jullien, R. Zhang, W. Wang, "Circuit Design Based on Majority Gates for Application with Quantum Dot Cellular Automata", IEEE Trans Signals, Systems and Computers, vol. 2, Nov 2004, pp. 1354-1357.

[10] M. Lieberman, S. Chellamma, B. Varughese, Y. Wang, C. S. Lent, G. H. Bernstein, G. L. Snider and F. Peiris, "Quantum Dot Cellular Automata at a Molecular Scale", Annals of the New York Academy of Sciences, vol. 960, 2002, pp. 225-239.

[11] M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowskajeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jozwiak, A. Coppola, B. Massey, " Regular Realization of Symmetric Functions Using Reversible Logic", Euromicro Symposium on Digital Systems Design, 2001, pp. 245-252,

[12] D. T. Lee, S. J. Hong, "An Algortithm for Transformation of an Arbitrary Switching Function to a Completely Symmetric Function", IEEE Trans on Computer, vol. 25, no. 11, November 1976, pp 1117-1123.

[13] E. Fredkin, T. Toffoli, "Conservative Logic", International Journal of Theoretical Physics, vol. 21, no. 3-4, April 1982, pp 219-253.

[14] D. L. Dietmeyer, "Generating Minimal Covers of Symmetric Function", IEEE TCAD, vol. 12, no. 5, , May 1993, pp. 710-713.

[15] P. K. Bhattacharjee, "Efficient Synthesis of Symmetric Functions", World Congress in Computer Science, Computer Engineering and Applied Computing (WORLDCOMP-2008) in International Conference on Computer Design (CDES'08), las Vegas, USA, July 2008, pp. 23-29.

Dr. Pijush Kanti Bhattacharjee is associated with the study of Engineering, Management, Law, Indo-Allopathy, Herbal, Homeopathic and Yogic medicines. He is having qualifications ME, MBA, MDCTech, AMIE, BSc, BA, LLB, BIASM, CMS, PET, EDT, FWT, DATHRY, BMus, KOVID, DH, ACE, FDCI etc. He worked in Department of Telecommunications (DoT), Govt. of India from June 1981 to Jan 2007 (26 years), lastly holding Assistant Director post at RTEC [ER], DoT, Kolkata, India. Thereafter, he worked at IMPS College of Engineering and Technology, Malda, WB, India as an

Assistant Professor in Electronics and Communication Engineering Department from Jan,2007 to Feb,2008 and Feb, 2008 to Dec, 2008 at Haldia Institute of Technology, Haldia, WB, India. In Dec, 2008 he joined at Bengal Institute of Technology and Management, Santiniketan, WB, India in the same post and department. He has written two books "Telecommunications India" & "Computer". He is a Member of IE, ISTE, IAPQR, India; CSTA, USA; IACSIT, Singapore and IAENG, Hongkong. His research interests are in Mobile Communications, Image Processing, Network Security, VLSI, Nanotechnology etc.

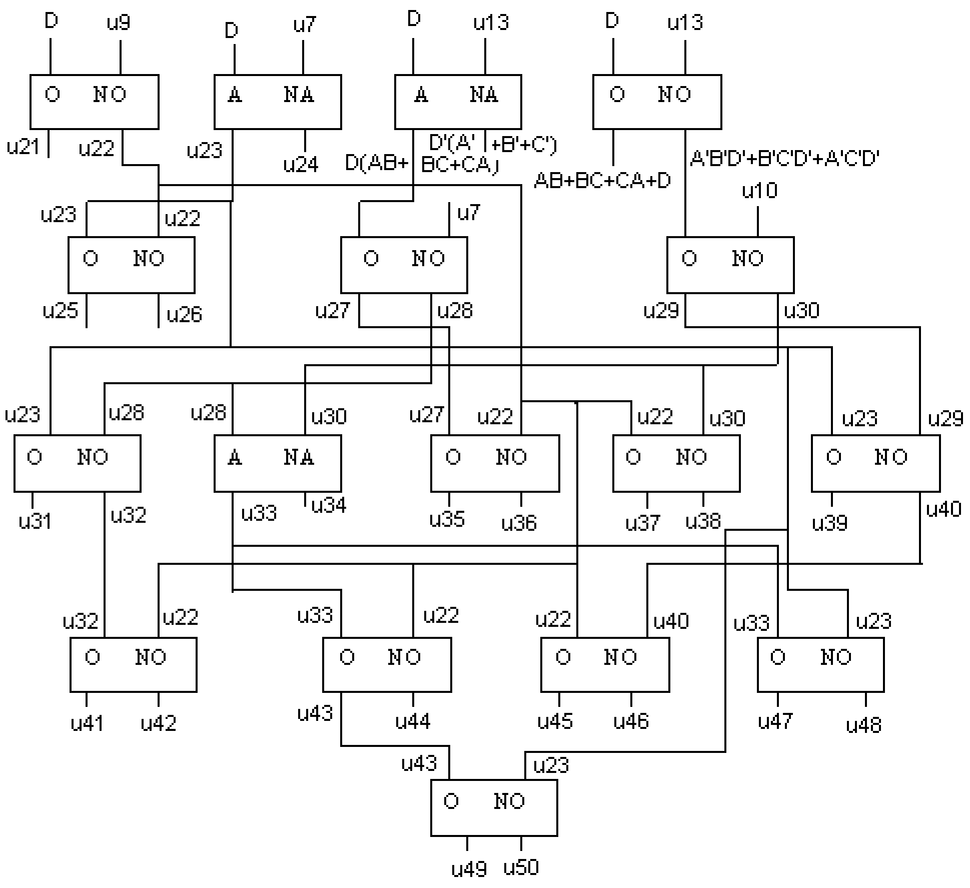


Fig. 11. Symmetric functions circuit diagram for 4-Input variables.

TABLE II. COMPARISON OF SYMMETRIC FUNCTIONS REALIZED BY ANY ARBITRARY NUMBER OF INPUT VARIABLES

Srl No	Number of Input Variables	Total Number of Symmetric Functions	Total Number of AND-NAND gates	Total Number of OR-NOR gates	Total Number of Gates	Total Number of Garbage Outputs	Total No of Garbage Output not act as further Input
i)	2	6	1	2	3	NIL	NIL
ii)	3	14	2	6	8	2	1
iii)	4	30	3	14	17	4	2
iv)	5	62	4	30	34	6	3
v)	n	$2^{n+1} - 2$	$n - 1$	$2^n - 2$	$2^n + n - 3$	$2n - 4$	$n - 2$