ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A Novel 128 bit adder using QCA

N. Archana[*1], M. Chandana[2], M. Sudha Lakshmi[3], M. Nandini[4], Mrs. K. Praveena[5]
[*1,2,3,4] Student, [5]Assistant Professor, Dept of EconE, SVEC, A.Rangampet, Tpt., India
archananagineni20@gmail.com

Abstract

As transistors decrease in size more and more of them can be accommodated in a single die, thus increasing chip computational capabilities. However, transistors cannot get much smaller than their current size. The quantum-dot cellular automata (QCA) approach represents one of the possible solutions in overcoming this physical limit. In this brief, we propose a new adder that outperforms all state-of-threat competitors and achieves the best area-delay tradeoff. The above advantages are obtained by using an overall area similar to the cheaper designs known in literature. In this we are reducing the delay which occurs in RCA and area which occurs in CLA. The 128-bit version reduced an area of 32.25µm2 and ADP of 542.

Keywords: Adders, quantum-dot cellular automata (QCA), Xilinx14.3.

Introduction

Nanotechnology draws much attention from the public now-a-days. Because the current silicon transistor technology faces challenging problems, such as high power consumption and difficulties in feature size reduction, alternative technologies are sought from researchers. Quantum-dot cellular automata (QCA) is one of the promising future solutions. Since it was first introduced in 1993 , experimental devices for semiconductor, molecular, and magnetic approaches have been developed . Quantum dot cellular automata, which is an array of coupled quantum dots to implement boolen logic functions. The advantage of QCA is high packing densities due to the small size of the dots, simplified interconnection and low area delay product.

Adders

In electronics, an adder or summer is a digital circuit that performs addition of numbers. In many computers and other kinds of processors, adders are used not only in the arithmetic logic unit(s), but also in other parts of the processor, where they are used to calculate addresses, table indices, and similar operations. Although adders can be constructed for many numerical representations, such as binary-coded, decimal or excess-3, the most common adders operate on binary numbers. In cases where two's complement or ones' complement is being used to represent negative numbers, it is trivial to modify an adder into an adder–subtractor. Other signed number representations require a more complex adder.

Adders are fundamental circuits for most digital systems and several adder designs in QCA have been proposed, and a performance comparison was improved. Better adder performance depends on minimizing the carry propagation delay and reducing the area.

Quantum dot Cell

In 1993, Lent et al. proposed a physical implementation of an automaton using quantum-dot cells. The automaton quickly gained popularity and it was first fabricated in 1997. Lent combined the discrete nature of both cellular automata and quantum mechanics, to create nano-scale devices capable of performing computation at very high switching speeds and consuming extremely small amounts of electrical power. Today, standard solid state QCA cell design considers the distance between quantum dots to be about 20 nm, and a distance between cells of about 60 nm. Quantum dot Cellular Automata are based on the simple interaction rules between cells placed on a grid. A QCA cell is constructed from four quantum dots arranged in a square pattern. These quantum dots are sites electrons can occupy by tunneling to them. Because of Coulombic repulsion, the two electrons will always reside in opposite corners. The locations of the electrons in the cell (also named polarizations *P)* determine two possible stable states that can be associated to the binary states 1 and 0. Although adjacent cells interact through electrostatic forces and tend to align their polarizations, QCA cells do not have intrinsic data flow directionality.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

Cell design

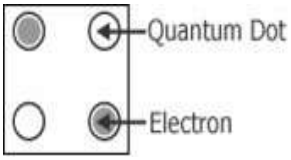

Fig 1: Simplified Diagram of QCA Cell

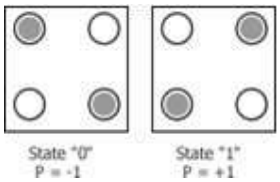

Fig 2: Four Dot Quantum Cell

The Figure 1 shows a simplified diagram of a quantum-dot cell. If the cell is charged with two electrons, each free electron to tunnel to any site in the cell, these electrons will try to occupy the furthest possible site with respect to each other due to mutual electrostatic repulsion. Therefore, two distinguishable cell states exist. Figure 2 shows the two possible minimum energy states of a quantum-dot cell. The state of a cell is called its polarization, denoted as P. Although arbitrarily chosen, using cell polarization P = -1 to represent logic "0" and P = +1 to represent logic "1" has become standard practice.

Clock zones

Several approaches have been suggested for computation with an array of QCA cells. One approach is based on transferring the array to an excited state from a ground sate by merely applying input data (without explicit clocking). The array is expected to settle to a new ground state. However, sometimes the transition may result in a metastable which is an intermediate state. To facilitate transfer to a new ground state, another approach based on clocking has been suggested.Clocking means application of an appropriate voltage to a cell which leads to adjustment of tunneling barriers between quantum dots for transfer of electrons between the dots. To achieve controllable data directions, the cells within a QCA design are partitioned into the clock zones that are progressively associated to four clock signals, each phase shifted by 90°. A QCA cell is clocked by using a four-phase clocking scheme as shown in figure. The four phases correspond to these clock zones are *switch*, *hold*, *release* and *relax*. In the switch phase, cells begin un polarized and with low potential barriers but the barriers are raised during this phase. In the hold phase, the barriers are held high while in the release phase, the barriers are lowered. In the last phase, namely relax, the barriers remain lowered and keep the cells in an un polarized state.

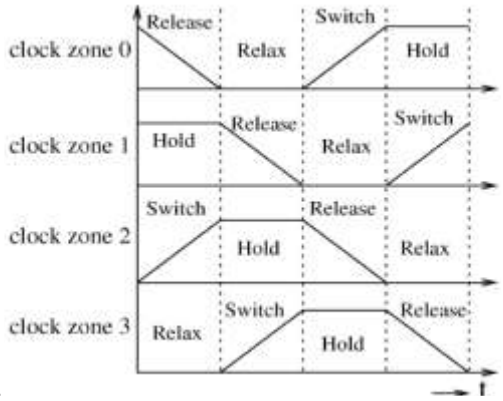

Fig 3: QCA clock zones

Logic gates

The logic elements of QCA are an inverter and majority gate. An inverter is designed by positioning cells diagonally from each other to achieve the inversion functionality. A majority gate consists of five QCA cells that realize the function of $M(a; b; c) = ab + bc + ac$. Two-input AND gate and OR gates can be designed by fixing one of the majority gate inputs to "0" and "1", respectively shown as follows.

AND = M(a,b,0)
OR = M(a,b,1)

If one input is set to 0, then the output is the AND of the other two inputs. If one input is set to 1, then the output is the OR of the other two inputs. With ANDs, ORs, and inverters, any logic function can be realized.

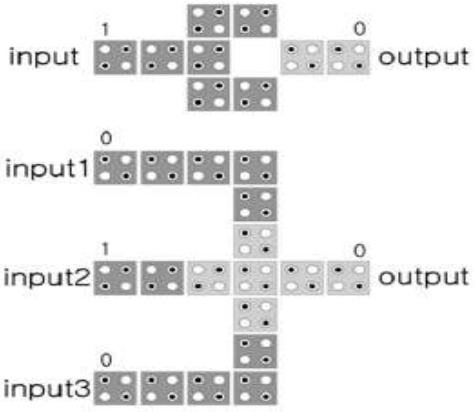

Fig 4: QCA inverter and Majority gate

Ripple Carry Adder

Half Adders can be used to add two one bit binary numbers. It is also possible to create a logical circuit using multiple full adders to add N-bit binary numbers.Each full adder inputs a **Cin**, which is the **Cout** of the previous adder. This kind of adder is

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

a **Ripple Carry Adder**, since each carry bit "ripples" to the next full adder. The first (and only the first) full adder may be replaced by a half adder.The block diagram of 4-bit Ripple Carry Adder is shown here below -

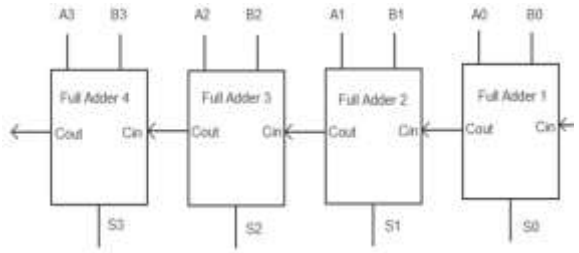

Fig 5: A conventional Ripple carry adder.

Excluding a *serial adder*, which consists of just a full adder and 1-bit storage, the simplest possible adder is the *carry-ripple.* The design of an *n*-bit ripple adder that takes two operands,
$A= A_{n-1}A_{n-2}A_{n-3...}A_0$, $B= B_{n-1}B_{n-2}B_{n-3...}B_0$ and produces a sum of $S= S_{n-1}S_{n-2}S_{n-3...}S_0$ is shown in Figure6. C_{i-1} is the carry into the adder and is usually 0 for unsigned arithmetic; Cn-1 is the carry out of the adder. The logical equations are
$S_i= (A_i \wedge B_i) \wedge C_{i-1}$
$C_i=A_iB_i+(A_i \wedge B_i) \wedge C_{i-1}$

If performance is measured in terms of logical date-delays, then the serial adder appears to be rather slow, because the full adders cannot always operate in parallel. In general, the full adder at stage *i* has to wait for a possible carry from stage i-1, which in turn has to wait for a possible carry from stage i-2, and so forth. The operational time is therefore O(*n*), in contrast with the O(log *n*) of the fastest adders.

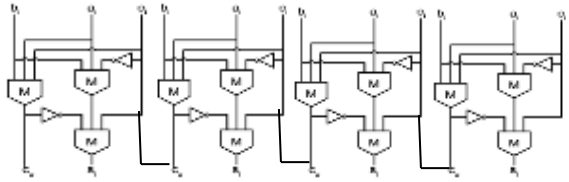

Fig 6: Ripple carry adder

Carry Look Ahead Adder

Carry-lookahead is arguably the most important technique in the design of fast adders, especially large ones. In straightforward addition, e.g. in a ripple adder, the operational time is limited by the (worst-case) time allowed for the propagation of carries and is proportional to the number of bits added. So faster adders can be obtained by devising a way to determine carries before they are required to form the sum bits. Carry-lookahead does just this, and, in certain cases the resulting adders have an operational time that is independent of the operands' word-length.

A carry, *Ci*, is produced at bit-stage *i* if either one is *generated* at that stage or if one is *propagated* from the preceding stage. So a carry is generated if both operand bits are 1, and an incoming carry is propagated if one of the operand bits is 1 and the other is 0. Let *Pi* and *Gi* denote the generation and propagation, respectively, of a carry at stage *i*, A_i and B_i denote the two operands bits at that stage, and C_{i-1} denote the carry into the stage. Then we have
$G_i=A_iB_i$
$P_i = A_i \wedge B_i$
$C_i = G_i + P_iC_{i-1}$

and the sum can be written as $S_i= P_i \wedge C_{i-1}$ which allows the use of shared logic to produce S_i and P_i.

$C_0 = G_0 + P_0C_{i-1}$
$C_1 = G_1 + P_1P_0C\text{-}1+ P_1G_0$
.
.
.
$C_i = G_i + P_{i-1}G_{i-1}+P_iP_{i-1}G_{i-2}+\ldots+P_iP_{i-1}P_{i-2}\ldots P_0C_{-1}$

where C_{i-1} is the carry into the adder. The equation for C_i states that there is a carry from stage i if there is a carry generated at stage i, or if there is a carry that is generated at stage i-1 and propagated through stage i or if , or if the initial carry-in, C_{i-1}, is propagated through stages 0,1,… i. The complete set, of equations show that, in theory at least, all the carries can be determined independently, in parallel, and in a time (three gate delays) that is independent of the number of bits to be added. The same is also therefore true for all the sum bits, which require only one additional gate delay.

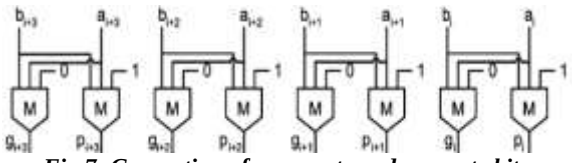

Fig 7: Generation of propagate and generate bits

$G_i = A_iB_i$
$P_i = A_i \wedge B_i$

Compared with a ripple adder, as well as some of the other adders, a pure carry-look ahead adder has high logic costs. Furthermore, high fan-in and fan-out requirements can be problematic: the fan-out required of the G_i and P_i signals grows rapidly with n, as does the fan-in required to form C_i. For sufficiently large values of *n*, the high fan- in and fan-out requirements will result in low performance, high cost, or designs that simply cannot be realized.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

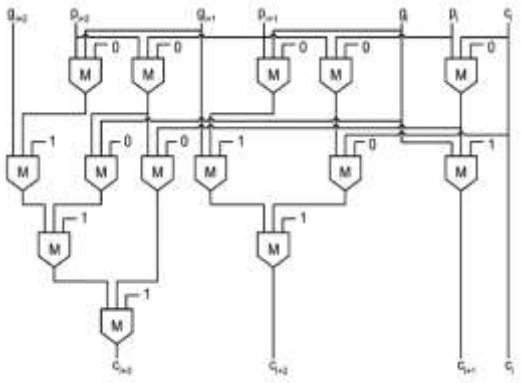

Fig 8: Carry block

This carry block is cascaded with the propagate and generate block. So, that carry is obtained with the following equation.

$C_i = G_i + P_iC_{i-1}$

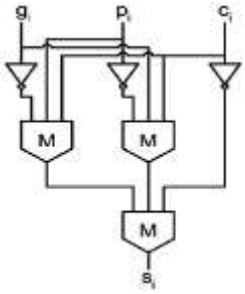

Fig 9: Sum block

This sum block is cascaded with the above carry block to obtain the sum. The following equation gives the sum bit

$S_i = P_i \wedge C_{i-1}$.

Novel bit adder

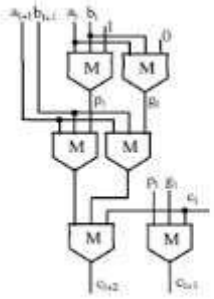

Fig 10: Basic novel 2 bit adder

The following shows the carry block which generate the carry bits.

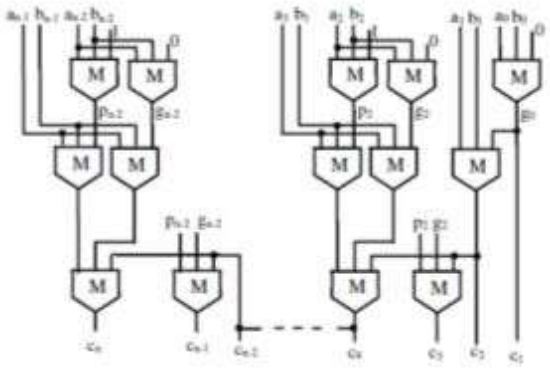

Fig11: Carry block

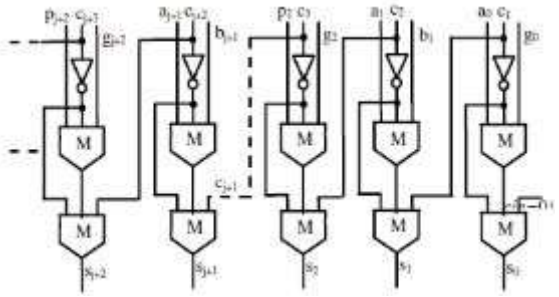

Fig 12: Sum block

The above carry block is cascaded with the sum block which generate the sum bits. The following are the equations for the carry bits and the sum bits.

$C_{i+2} = M(M(a_{i+1},b_{i+1},g_i)M(a_{i+1},b_{i+1},p_i)c_i)$

For sum block:

- For odd

  $S_{j+1}=M(\sim C_{j+3}M(a_{j+2},\sim C_{j+3},b_{j+2}),C_{j+2})$
- For even

  $S_{j+2}=M(\sim C_{j+3}M(P_{j+2},\sim C_{j+3},G_{j+2}),C_{j+2})$

Simulation results:

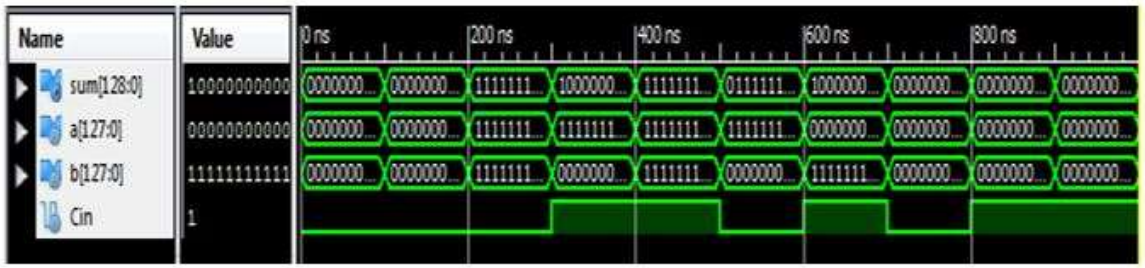

Fig 13: Simulation results obtained for the 128 novel bit adder

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

Implementation

In this paper code is written in Verilog for developing the software. The XINLINX 14.3 is used to edit, compile and debug this code. The following shows the simulation results for 128 novel bit adder

Comparision results

Critical path consistencies and post layout characteristics, such as cell count, overall size, delay, number of clock phases, and ADP, are shown in Table II for all the compared adders. The number of cascaded MGs within the worst case computational path directly impacts on the achieved speed performances as an MG always adds one more clock phase.

Adder	N	Cell count	Size(μm)^2	Delay	Clock phases	ADP
NEW	8	1008	1.08	2	8	2.26
	16	4032	4..3	3	12	7.98
	32	8064	8.06	5	20	33.25
	64	16128	16.12	9	36	168.48
	128	32256	32.25	17	68	548.25
RCA	8	712	0.74	2	11	2.03
	16	1602	1.99	16	19	9.45
	32	3901	6.46	128	35	56.52
	64	10926	20.92	1024	67	350.41
CLA	8	1785	1.46	0.5	9	3.29
	16	4114	3.67	6.75	15	13.76
	32	12540	11.84	9	27	79.92
	64	33302	35.63	3	49	436.47

Conclusion

A new adder designed in QCA was presented. It achieved speed performances higher than all the existing QCA adders, with an area requirement comparable with the cheap RCA and CFA demonstrated in [13] and [16]. The novel adder operated in the RCA fashion, but it could
propagate a carry signal through a number of cascaded MGs significantly lower than conventional RCA adders. In addition, because of the adopted basic logic and layout strategy, the number of clock cycles required for completing the elaboration was limited. A 128-bit binary adder designed as described in this brief exhibited a delay of only seventeen clock cycles, occupied an active area of 32.25 μm2, and achieved an ADP of only 548.25.

References

1. *Stefania Perri, Pasquale Corsonello, and Giuseppe Cocorullo, "**Area-Delay Efficient Binary Adders in QCA",** 2013 Ieee Transactions On Very Large Scale Integration (Vlsi) Systems.*
2. *C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernestein, "Quantum cellular automata," Nanotechnology, vol. 4, no. 1, pp. 49–57, 1993*
3. *W. Liu, L. Lu, M. O'Neill, and E. ESwartzlander, Jr., "Design rules for quantum-dot cellular automata," in Proc. IEEE Int. Symp. Circuits Syst., May 2011, pp. 2361–2364*
4. *H. Cho and E. E. Swartzlander, "Adder design and analyses for quantum-dot cellular automata," IEEE Trans. Nanotechnol., vol. 6, no. 3, pp. 374–383, May 2007.*
5. *H. Cho and E. E. Swartzlander, "Adder and multiplier design in quantum-dot cellular automata," IEEE Trans. Comput., vol. 58, no. 6, pp. 721–727, Jun. 2009.*

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

6. *V. Pudi and K. Sridharan, "Efficient design of a hybrid adder in quantumdot cellular automata," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 9, pp. 1535–1548, Sep. 2011.*
7. *S. Perri and P. Corsonello, "New methodology for the design of efficient binary addition in QCA," IEEE Trans. Nanotechnol., vol. 11, no. 6, pp. 1192–1200, Nov. 2012.*
8. *Amos omondi, Benjamin premkumar "Chapter 4 Addition", Residue number systems pp. 83 to 93.*