Design of A Pipeline for A Fixed-Point Multiplication using Single Electron Tunneling Technology

Anup Kumar Biswas
Assistant Professor, Department of Computer Science and Engineering
Kalyani Govt. Engineering College, Kalyani,
Nadia-741235, West Bengal, India,

***Abstract*:- For high operating speed, low power consumption, and high integration density-based equipment(s) are financially indispensable in engineering, science and technology in the present era. Single Electron tunneling Device (SED) is one such equipment by which all Boolean logic gates can be implemented. SEDs and Linear Tunnel Gates (TLG) are capable of controlling the transport of only an electron through the tunneling junction. It is a single electron that is sufficient to store information in the SED in the atmosphere of 0K. Power consumed in the single electron tunneling circuits is very low in comparison with (CMOS) circuits. The processing speed of TLG based device will be nearly close to electronic speed. The single-electron transistor (SET) and LTG attracts the researchers, scientists or technologists to design and implement large scale circuits for the sake of the consumption of ultra-low power and its small size. All the tunneling events for the case of a LTG-based circuit happen when only a single electron tunnels through the tunnel junction under the proper applied bias voltage and multiple input voltages. For implementing a fixed-point Multiplication circuit, LTG would be a best candidate to fulfil the requirements of it. Ultra-low noise is generated during tunneling through the LTG based circuit. Different logic gates, a D-Flip-flop, a full adder are implemented, and based on them, a "Pipeline for a fixed-point Multiplication" is presented at last.**

Key words: Electron-tunneling, Coulomb-blockade, pipeline, Multiplication, linear threshold gate

1. COULOMB BLOCKADE AND SINGLE ELECTRON TRANSISTOR

A tunnel junction in Fig.1 (a) comprises a thin insulating barrier between the two conducting electrodes It is having a capacitance C and a resistance R_t. The electrodes of the tunnel junction may be a superconducting or semiconducting. If they are of superconducting, electrons with one elementary charge (1.602×10^{-19} C) carry the current.

In the classical electrodynamics, current can't flow through the insulating barrier. But in the case of quantum mechanics, there must be a positive probability (i.e., more than zero) for an electron residing one side of the barrier to reach the other side of it when the proper bias or input voltages are supplied. If bias voltage is applied properly, there will be a current flow. Neglecting other effects, in accordance with first-order–approximation-tunneling, current will be following in proportion to the applied bias voltage. In the case of electrical terms, a tunnel junction shown in Fig 1(a) behaves as a resistor having a constant value depending on its barrier thickness. If two conductors are connected with an insulating layer between them, there will have not only a resistance but also a capacitance in the junction. In this context, the insulator acts as dielectric and two conducting plates with dielectric forms a capacitor in the tunnel junction. For the discrete nature of electric charge, current following through a tunnel junction is a series of events in which merely one electron will be able to pass or tunnel through the tunnel junction. If the single electron tunnels the junction, the tunnel capacitance is charged with an elementary charge q_e (1.602×10^{-19} C) building up a voltage V=$\frac{e}{C}$, where C=junction capacitance. When the capacitance of the tunnel junction is ultra-small, the voltage building up in the tunnel junction may be adequate to prevent another electron to pass through. The electrical current is suppressed as the bias voltage is lower than the voltage created in the tunnel junction and the resistance of the device will no longer remain constant. The increment of the differential resistance of the tunnel junction around zero bias is called the Coulomb blockade [3, 10, 13, 17].

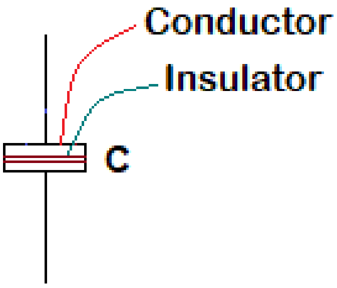


Fig.1 (a) Tunnel Junction

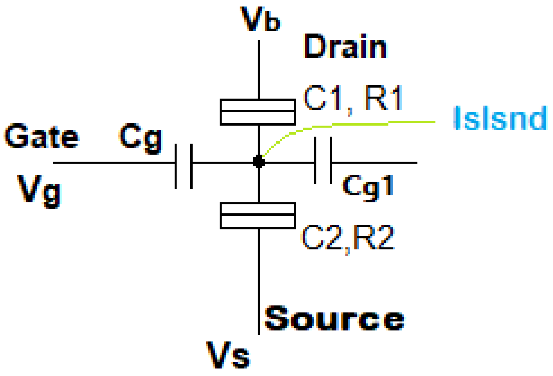


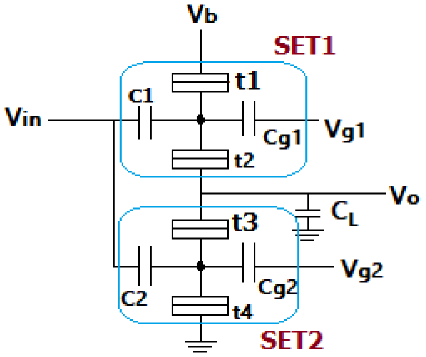
Fig.1 (b) Single Electron Transistor (SET)

The principle of single-electron technology [2, 3, 4, 9, 16] is developed on the basis of the single electron tunneling and Coulomb blockade. Single electron tunneling circuits seems to be a promising candidate for future VLSI circuits for its ultra-low power consumption, ultra-small size, reducing node capability and rich functionality.

A (SET) [4, 5] made up of two tunnel junctions shown in Fig. 1(b) having their capacitances and resistances C_1, C_2 and R_1, R_2 respectively, shares only a common inland with a low capacitance**.** The electric potential of the island can be tuned in by a third electrode, called gate, and the gate is capacitively coupled (gate capacitance C_g) to the island. An extra capacitance C_{g1} may intentionally be connected to the island for the purpose of manipulating the gate voltage (input). The drain, source and gate voltages are marked by V_b, V_s and V_g respectively. For proper operations of SET both of the resistances R_1, and R_2, are to be greater than R_q= h/e ^{2} ≈ 25.8 KΩ and charging energy $\mathrm{E_C}$=$\mathrm{e}^2/2\mathrm{C}$ [where C = C_1 + C_2+C_g + C_{g1}] has to be greater than thermal fluctuations kT, i.e, $E_C = \frac{e^2}{2C} > kT$, the product of the Boltzmann constant, k , and the temperature, T in kelvin. The Boltzmann constant value is 1.380649 × 10^{-23} joule / kelvin , or 1.380649 × 10^{-16} erg / kelvin.

2. INVERTER

The inverter [3, 7, 8] depicted in Fig. 2(a) is made up of two SETs connected in series. The same input voltage is directly coupled to the islands of the SET1 and SET2 through two capacitors C_1 and C_2 respectively. The island of each SET has a size smaller than 10 nm of gold and their capacitances must be less than 10^{-17} F. The output terminal V_0 is connected to the common channel of the two SETs and to the ground via a capacitor C_L to put down charging effects.



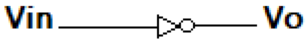


Fig.2 (a) an Inverter

Fig.2 (b) Symbol of an Inverter

For the inverter, the parameters values chosen are: V_{g1}=0, V_{g2}=0.1× $\frac{q_e}{C}$, $C_L = 9C$, $t_4 = \frac{1}{10}C$, $t_3 = \frac{1}{2}C$, $t_2 = \frac{1}{2}C$, $t_1 = \frac{1}{10}C$, $C_1 = \frac{1}{2}C$, $C_2 = \frac{1}{2}C$, $C_{g1} = \frac{17}{4}C$ and $C_{g2} = \frac{17}{4}C$, R1 =R2=100KΩ. For simulation purpose, the value of C is taken 1aF.
The operation of the inverter will be like this: - the output V_0 value will be high when the input voltage Vin is low and V_0 value will be low when the input voltage is high. For achieving this target, we set the voltages $for\ V_{g1} = 0\ and\ V_{g2}$ =16mV along with the tuning gate voltages, at present, Vin both for SET1 and SET2. Now if V_{in} is low, the SET1 is in conduction mode and the SET2 is in Coulomb blockade. This effectively results the output to voltage V_b and causes the output voltage to be high, Coulomb blockade interferes the steady flow of current because whenever the high voltage (logic 1) is applied to the input, it causes to shift the induced charge on each of the islands of the two SETs by a fraction of an electron charge and keeps the SET1 in Coulomb blockade and the SET2 in conducting mode. As a result, the output shifts from high to low.

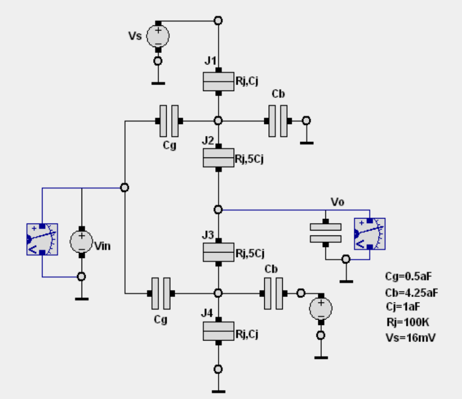


Fig. 2(c) Simulation set of Inverter

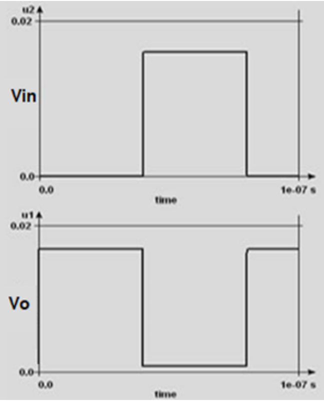


Fig.2 (d) Simulation result of Inverter

In this work we assume the Boolean logic inputs corresponding to the voltages like: logic “0” =0 Volts and logic “1”=0.1 × $\frac{q_e}{C}$

We assume, for simulation and other purposes, C=1aF then Logic “1”= 0.1× $\frac{1.602\times10^{-19}}{1\times10^{-18}}$=0.1 × 1.602 × 10^{-2}=16.02 × 10^{-3} =16.02 ≅ 16 mV.

3. MULTIPLE INPUT THRESHOLD LOGIC GATE

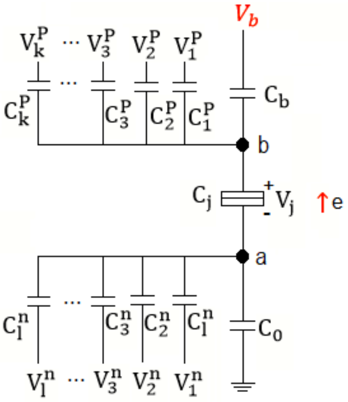


Fig. 3 Multiple input threshold logic gate

The multiple input threshold logic gate [5, 7, 8, 14] consists of a tunnel junction, two multiple inputs connected at points ‘a’ and ‘b’. Each input voltage V_k^P, for the upper side is connected to point “b” through the capacitor $C_k^P$and each input voltage V_l^n, for the lower side is connected to point “a” through the capacitor C_l^n. Bias voltage V_b is also connected to point “b” through a capacitor C_b. Point “a” is grounded through a capacitor C_0 . The capacitor of tunnel junction is C_j. This multiple input threshold logic gate can also be defined as a Junction-Capacitor (C-J) circuit**.** Using the Junction-Capacitor circuit we will be able to implement the linear threshold gate (LTG) being presented by the signum function of g(x) expressed by equations (3) and (4).

$$f(x) = \mathrm{sgn}\{g(x)\} = \begin{cases} 0, & if\ g(x) < 0 \\ 1, & if\ g(x) \geq 0 \end{cases} \quad \cdots\cdots (3)$$

$$g(x)= \sum_{k=1}^{n}(w_k \times x_k) - \emptyset \quad \cdots\cdots (4)$$

where x_k are being the n Boolean inputs and w_k are being the corresponding *n* integer weights. The linear threshold gate compares the weighted sum of the inputs $\sum_{k=1}^{n}(w_k \times x_k)$ with the threshold value $\emptyset$. If the value of the weighted sum is greater than the threshold or critical voltage value then the logic output of the LTG will be “1”, otherwise it will be “0”.

The tunnel junction capacitance C_j and the output capacitance C_0 are considered the two basic circuit elements in a LTG. The input signal vector elements $\{V_1^P, V_2^P, V_3^P, \dots, V_k^P\}$ are weighted by their respective vector element capacitances $\{C_1^P, C_2^P, C_3^P, \dots, C_k^P\}$ and added to the voltage, V_j, across the tunnel junction. On the other hand, the input signal vector elements $\{V_1^n, V_2^n, V_3^n, \dots, V_l^n\}$ are weighted by their respective vector element capacitances $\{C_1^n, C_2^n, C_3^n, \dots, C_l^n\}$ are being subtracted from the voltage, V_j, across the junction.

The critical voltage V_c *is* required to enable tunneling action, and it acts as the intrinsic threshold of the tunnel junction circuit. The bias voltage V_b connected to tunnel junction through the capacitance, C_b, is really being used to adjust the gate threshold to the desired value $\emptyset$. When a tunneling phenomenon happens though the tunnel junction an electron passes through the junction from a to b [Fig. 3].

We are using the following notations for the rest our discussion.

$$C_\Sigma^P = C_b + \sum_{k=1}^{g} C_k^P \quad \cdots\cdots (5)$$

$$C_\Sigma^n = C_0 + \sum_{l=1}^{h} C_l^n \quad \cdots\cdots (6)$$

$$C_T = C_\Sigma^P C_j + C_\Sigma^P C_\Sigma^n + C_j C_\Sigma^n \cdots\cdots (7)$$

When we are thinking of all voltage sources in Fig. 3 to be 0 (or connected to ground), the circuit can be thought of three capacitors connected in series, namely C_{Σ}^{P}, C_{Σ}^{n} and C_j. Here, C_T is represented by the sum of all 2-term products of these three capacitances. It is the time to find the expression of critical voltage V_c of the tunnel junction. We suppose the capacitance of the tunnel junction being C_j and the remainder of the circuit having the equivalent capacitance is C_e, as observed from the tunnel junction's perspective, we can measure the critical voltage[6,7] for the tunnel junction as

$$V_c = 0.5\frac{e}{C_j + C_e} \quad \cdots\cdots (8)$$

$$V_c = 0.5\frac{e}{C_j + (C_{\Sigma}^{P} || C_{\Sigma}^{n})} = 0.5\frac{e}{C_j + \frac{(C_{\Sigma}^{P}) * (C_{\Sigma}^{n})}{(C_{\Sigma}^{P} + C_{\Sigma}^{n})}}$$

$$= 0.5\frac{e(C_{\Sigma}^{P} + C_{\Sigma}^{n})}{C_j * (C_{\Sigma}^{P} + C_{\Sigma}^{n}) + (C_{\Sigma}^{P}) * (C_{\Sigma}^{n})}$$

$$= 0.5\frac{e(C_{\Sigma}^{P} + C_{\Sigma}^{n})}{C_T} \quad \cdots\cdots (9)$$

When we define the voltage across the junction as V_j , a tunnel event will happen through this tunnel junction if and only if the following condition is satisfied.

$$|V_j| \geq V_c \cdots\cdots (10)$$

If the junction voltage is less than the critical voltage i.e. $|V_j| < V_c$ there being no tunneling events through the circuit's tunnel junction. As a result, the tunneling circuit remains in a $stable\ state$.

Theoretically, the thresholds are being integer numbers. And the threshold logic equations for two inputs AND, OR, NAND and NOR gates can be written [6] as follows.

$$AND(X, Y) = sgn\{X + Y - 2\} \cdots\cdots (11)$$
$$OR(X, Y) = sgn\{X + Y - 1\} \cdots\cdots (12)$$
$$NAND(X, Y) = sgn\{-X - Y + 2\} \cdots\cdots (13)$$
$$NOR(X, Y) = sgn\{-X - Y + 1\} \cdots\cdots (14)$$

If the threshold becomes $\emptyset = i$ (i being an integer value), this implies that the gates perform their functions correctly for any value in the interval $[i - 1 < \emptyset \leq i\ ]$. For the purpose of maximizing robustness for variations in parameter values, the threshold value $\emptyset = i$ is replaced by the average $\emptyset = i - 0.5$. As a result, the threshold logic equations for the two-input AND, OR, NAND and NOR gates can be expressed as

$$AND(X, Y) = sgn\{X + Y - 1.5\} \cdots\cdots (15)$$
$$OR(X, Y) = sgn\{X + Y - 0.5\} \cdots\cdots (16)$$
$$NAND(X, Y) = sgn\{-X - Y + 1.5\} \cdots\cdots (17)$$
$$NOR(X, Y) = sgn\{-X - Y + 0.5\} \cdots\cdots (18)$$

The threshold gate-based implementations of the Boolean Logic gates have the same basic circuit topology for all we have to draw in the subsequent sections. The threshold gates consist of a bias capacitance C_b, a tunnel junction having capacitance C_j , and an output capacitance C_0 . The AND and OR gates contain two input capacitors. For both the AND and OR gates, the two input capacitors are $C_1^P = C_2^P = 0.5C$ for positively weighted inputs. On the other hand, for the NAND and NOR gates, two capacitors hold the values as $C_1^n = C_2^n = 0.5C$ for negatively weighted inputs.

Every threshold gates is augmented with an inverter/buffer. The function of the inverter/buffer is to invert the input of a threshold gate. The logic function done by the buffered threshold gate is being the inverse of "which is done by the threshold gate itself". For instance, a buffered AND gate implements the $NAND$ function. For the rest of our discussion, when we refer to a logic function such as AND, we imply the logic function performed by the entire gate (i.e., threshold gate plus an output buffer).

The parameters used for the implementations and simulations of different gates like AND, OR, NAND and NOR gates and other combinational or sequential circuits are given in Table-1 [6,7].

Table-1

Threshold gate	C_b	C_0	C_1^P	C_2^p	C_1^n	C_2^n	C_j	$C_{g1} = C_{g2}$	$C_2 = C_3$	$C_1 = C_4$	$C_{b1} = C_{b2}$	C_L
2-inout AND	10.6C	8C	-	-	0.5C	0.5C	0.1C	0.5C	0.1C	0.5C	4.25C	9C
2-inout OR	11.7C	8C	-	-	0.5C	0.5C	0.1C	0.5C	0.1C	0.5C	4.25C	9C
2-inout NAND	13.2C	9C	0.5C	05C	-	-	0.1C	0.5C	0.1C	0.5C	4.25C	9C
2-inout NOR	117C	9C	0.5C	05C	-	-	0.1C	0.5C	0.1C	0.5C	4.25C	9C
3-inout OR	11.8C	7.8C	0.4C	-	0.5C		0.1C	0.5C	0.1C	0.5C	4.25C	9C
$logic\ 0 = 0\ V,\ logic\ 1 = 16m\ V, R_j$=$10^5\ \Omega$, C_j=0.1aF, other capacitance values are in terms of C, where $C = 1\ aF$												

4. AND GATE

For implementing the AND gate we will use the parameters $C_1^n = C_2^n$=0.5aF, $\boldsymbol{C_{b1}} = \boldsymbol{C_{b2}} = 4.25aF$, $C_{g1} = C_{g2} = 0.5aF$, $C_L = 9aF, C_0 = 8aF$, R_j=10^5 Ω in Fig. 4(a) and accordingly after running the simulator the output we get is given in Fig. 4(b).

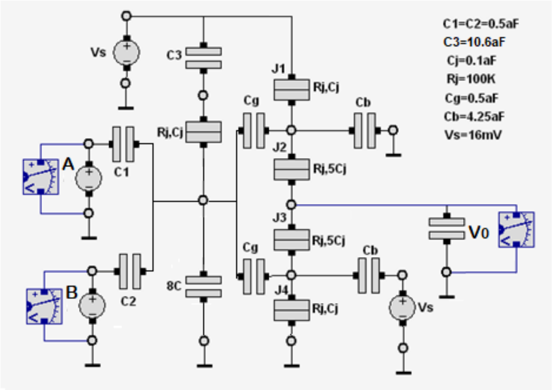


Fig.4 (a) AND Gate

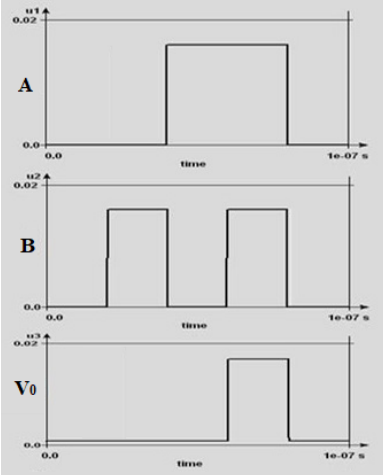
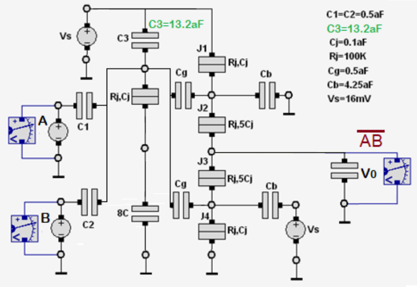


Fig.4 (b) Simulation result of AND gate

5. NAND GATE

For implementing the NAND gate, we will use the parameters $C_1^P = C_2^P$=0.5aF, $\boldsymbol{C_{b1}} = \boldsymbol{C_{b2}} = 4.25aF$, $C_{g1} = C_{g2} = 0.5aF$, $C_b = C_3 = 13.2aF, C_L = 9aF, C_0 = 8aF$, R_j=10^5 Ω in Fig. 5(a) and accordingly after running the simulator the output we get is given in Fig.5(b).



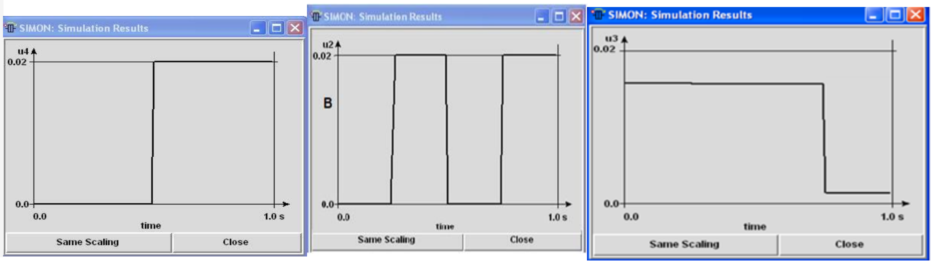
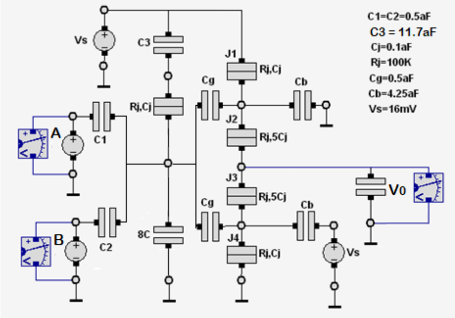


Fig.5 (a) NAND Gate

Fig.5 (b) Simulation result of NAND gate

6. OR GATE

For implementing the OR gate we will use the parameters $C_1^n = C_2^n$=0.5aF,$C_3 = 11.7aF$, $\boldsymbol{C_{b1}} = \boldsymbol{C_{b2}} = 4.25aF$, $C_{g1} = C_{g2} = 0.5aF$, $C_L = 9aF, C_0 = 8aF$, R_j=10^5 Ω, $V_s = 16mV$ and accordingly after running the simulator the output we get is given in Fig. 6(b).



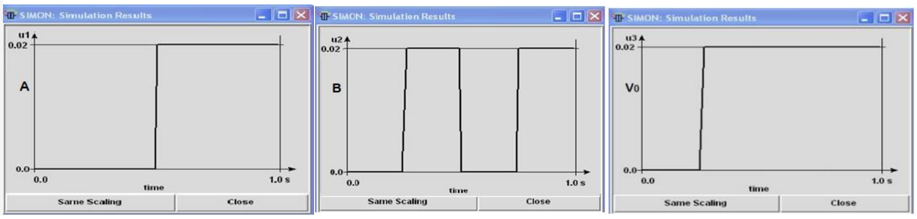
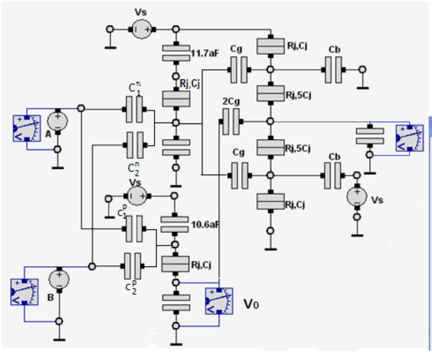


Fig. 6(a) OR gate

Fig. 6(b) simulation result of OR gate

7. EXCLUSIVE OR (XOR) GATE

For implementing the XOR gate we will use the parameters $C_1^n = C_2^n$=$C_1^P = C_2^P = C_g = 0.5$aF, $C_3 = 11.7aF$, $C_{b1} = C_{b2} = C_b = 4.25aF$, $C_{g1} = C_{g2} = 0.5aF$, $C_L = 9aF$, $C_0 = 8aF$, $C_j = 0.1$aF, R_j= 10^5 Ω, $V_s = 16mV$ and accordingly after running the simulator the output we get is given in Fig. 7(b).



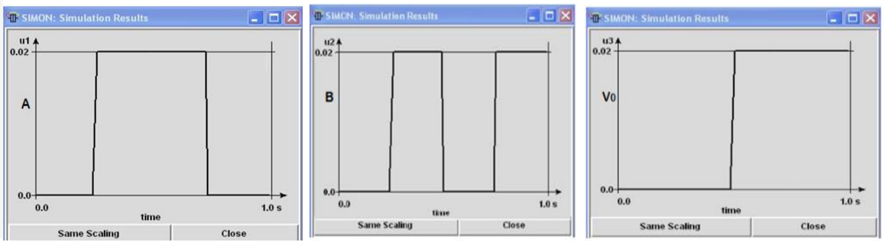
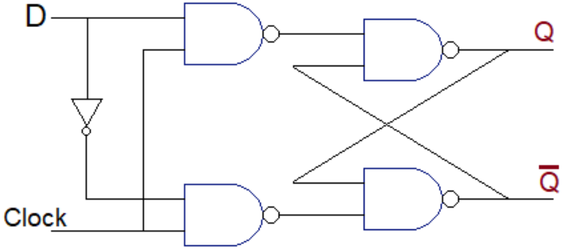


Fig. 7(a) XOR gate Fig. 7(b) simulation result of XOR gate

8. DESIGN OF D-FLIP-FLOP USING LOGIC GATES

The flip-flop is a sequential circuit used for storing data from two stable states. A classical D Flip-flop is consisting of four NAND gates and one inverter shown in Fig. 8(a). There are two inputs- D and Clock, and two outputs- Q and Q̄. The input output relationship is shown in the truth table in Fig. 8(b).



Truth table		
Clock	D	Q
0	0 or 1	No Change
↑	0	0
↑	1	1

Fig. 8(a) Diagram of D-flip flop Fig. 8(b) Truth table.

On the basis of the pattern of the D Flip-flop drawn in Fig. 8(a), we can easily implement a D Flip-flop with the help of threshold logic gate based NAND and a buffer/ inverter shown in Fig. 2(a) or 2(c).

The same parameters used in the case of a NAND and an inverter will be used in the TLG-based D Flip-flop. The output is provided with the positive edge trigger of the clock.

The input and output signals of the D-Flip-flop shown in the Fig. 9(b) through 9(e) are the D-Flip-flop input signal, Clock signals, output of the Q and output of Q̄ respectively.

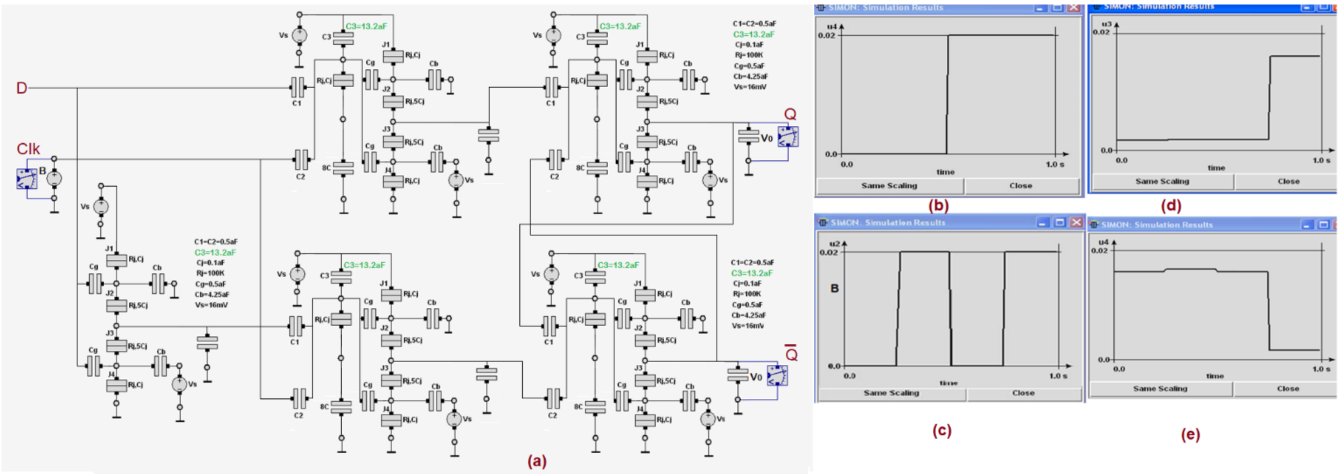


Fig. 9 (a) D Flip-flop using one buffer and four NAND gate based on TLG (b) D input (c) clock signal (d) output of Q and (e) output of Q̄

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 04, April-2021

9. FULL ADDER

The logical expressions of a full adder [7] for two inputs X_i and Y_i with carry C_i are: (i) sum S_i and (ii) Carry C_{i+1} are given in equations (19) and (20) respectively.

$$S_i = A_i \oplus B_i \oplus C_i \quad \cdots\cdots (19)$$

$$C_{i+1} = A_i\, B_i \vee B_i C_i \vee C_i A_i \quad \cdots\cdots (20)$$

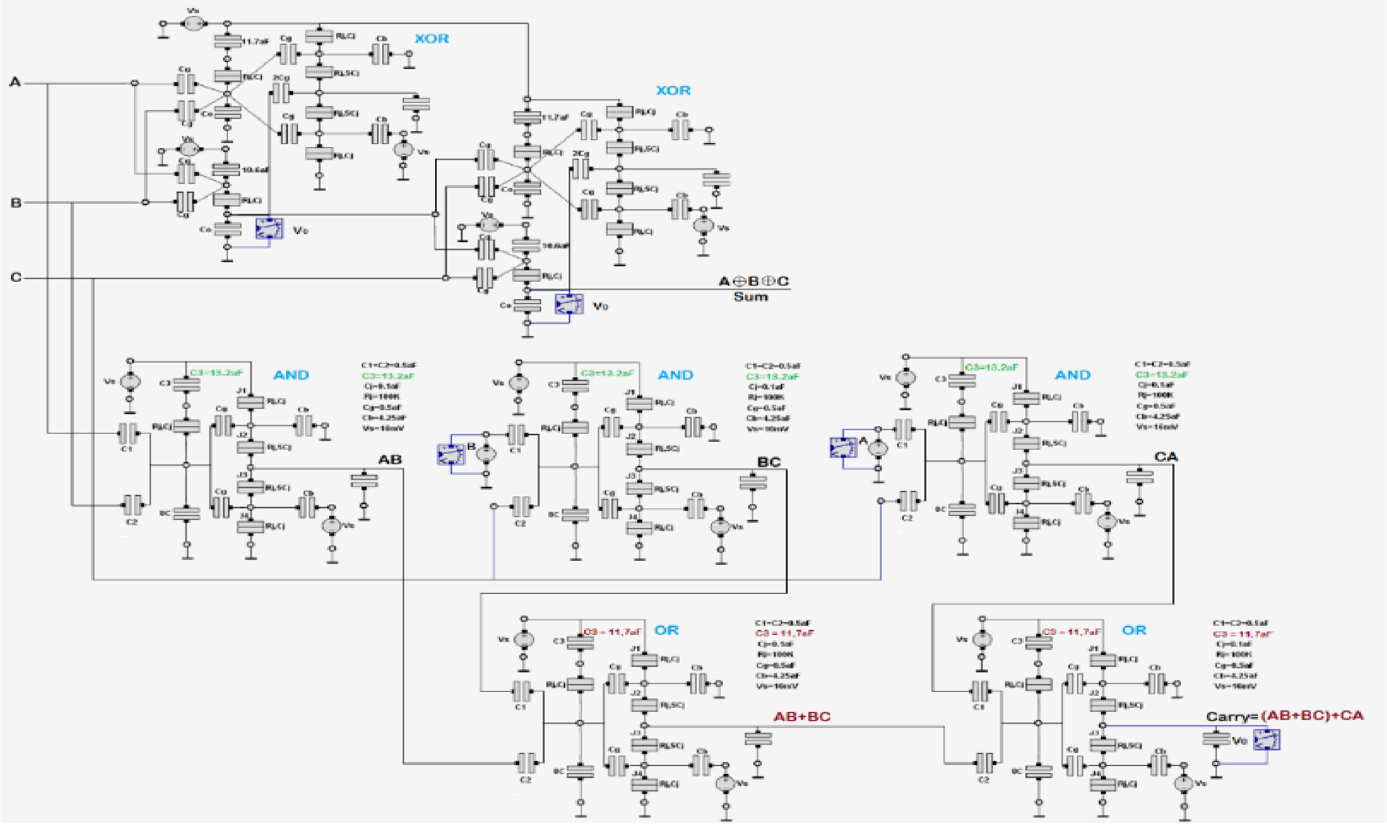


Fig.10 (a) A single bit Full Adder

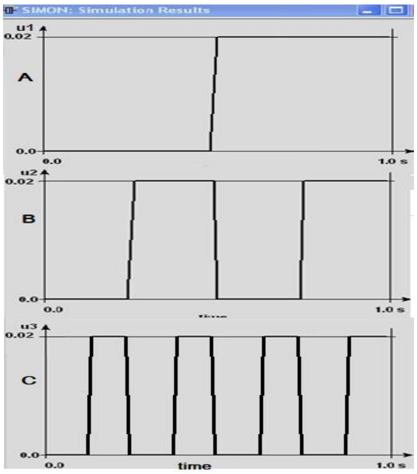


Fig. 10(b) three inputs

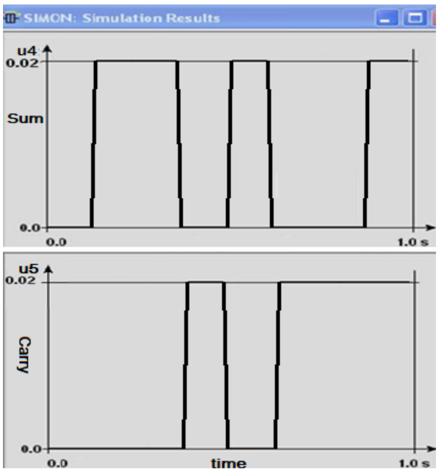


Fig. 10(c) sum & Carry output

According to the equations (19) and (20), we have implemented a full adder circuit in Fig. 10(a). To implement the full adder circuit, it requires two XOR gates, three AND gates and two OR gates which have already been described. The simulation results are shown in Fig. 10(b) and 10(c). Solution space of the full adder having two Boolean expressions of $S_i = (A_i \oplus B_i \oplus C_i) = 1$ and carry out $C_{i+1} = (A_i\, B_i \vee B_i C_i \vee C_i A_i\,)$ =1 is represented by the Fig. 10(d).

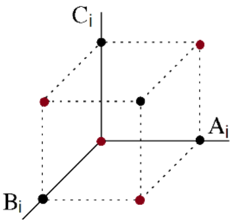


Fig. 10(d) Solution space of $S_i = (A_i \oplus B_i \oplus C_i) = 1$ Black color, and Solution space of $C_{i+1} = (A_i\, B_i \vee B_iC_i \vee C_iA_i\,)$ =1 Red color

10. CARRY PROPAGATION ADDER

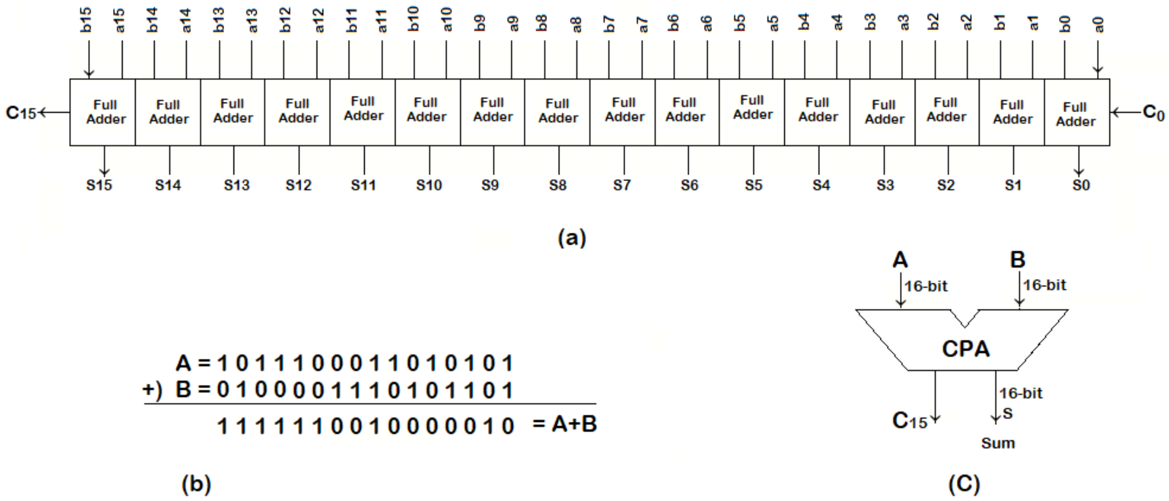

b15 a15 b14 a14 b13 a13 b12 a12 b11 a11 b10 a10 b9 a9 b8 a8 b7 a7 b6 a6 b5 a5 b4 a4 b3 a3 b2 a2 b1 a1 b0 a0
Full Adder
C15
C0
S15 S14 S13 S12 S11 S10 S9 S8 S7 S6 S5 S4 S3 S2 S1 S0
(a)
A = 1 0 1 1 1 0 0 0 1 1 0 1 0 1 0 1
+) B = 0 1 0 0 0 0 1 1 1 0 1 0 1 1 0 1
1 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 = A+B
(b)
A
B
16-bit
16-bit
CPA
C15
16-bit
S
Sum
(C)

Fig.11 (a) A sixteen-bit Carry-propagate Adder (CPA), (b) Sum of A and B, (c) Symbol of sixteen-bit CPA

For the purpose of high-speed operation of addition, the use of Carry-propagation adder (CPA) [19] is needed. The CPA adds two binary numbers of 16-bits and produces an arithmetic sum depicted in Fig. 11(b). The carries generated in this case in successive digits are propagated from LSB end toward MSB end. The CPA can be created either by using ripple carry propagation or carry look-ahead technique. Here we have adopted the first case, i.e., ripple carry propagation. A figure of a CPA, an example of its calculation and the symbol of CPA are given in Fig.11 (a), (b) and (c) respectively.

11. CARRY-SAVE ADDER

A separate high-speed addition technique will be required for adding three numbers at a time and we define this technique as Carry-save adder (CSA) [19]. In CSA the carries will not be allowed to propagate from low to high end but instead they will be saved in a carry vector representing a number. In a general way, an n-bit CSA is specified like this: We assume that A, B and C are three n-bit numbers expressed as A= $(a_{n-1}, a_{n-2}, \dots a_1, a_{0\,,})$, B= $(b_{n-1}, b_{n-2}, \dots b_1, b_{0\,,})$ and C = $(c_{n-1}, c_{n-2}, \dots c_1, c_{0\,,})$.

The CSA operations are performed bitwise simultaneously on a column of digits to generate two n-bit numbers as (i) sum vector S_v= $(0, S_{n-1}, S_{n-2}, \dots S_1, S_0\,)$ and carry vector C= $(c_n, c_{n-1}, \dots c_1, c_0,$ 0), Noted that, the leading bit for the S_v, of course, is a 0 and the tail bit for the carry vector C is a 0. Input-output relationship will be expressed by the equations given in (19) and (20), for i = 0, 1, 2, ···, n-1, where ⊕ indicates exclusive-OR and ⋁ indicates logical OR operation.

Now, after the addition of three binary numbers A, B and C we obtain the arithmetic sum S_v=A⊕B⊕C. And S_b is obtained by bit-wise addition of two binary numbers S_v and carry C_i shown in Fig. 12(c) i.e., S_b= S_v+C_i. The operation of CSA is presented in Fig. 12(d) and its symbol is given in the Fig.12(e) .

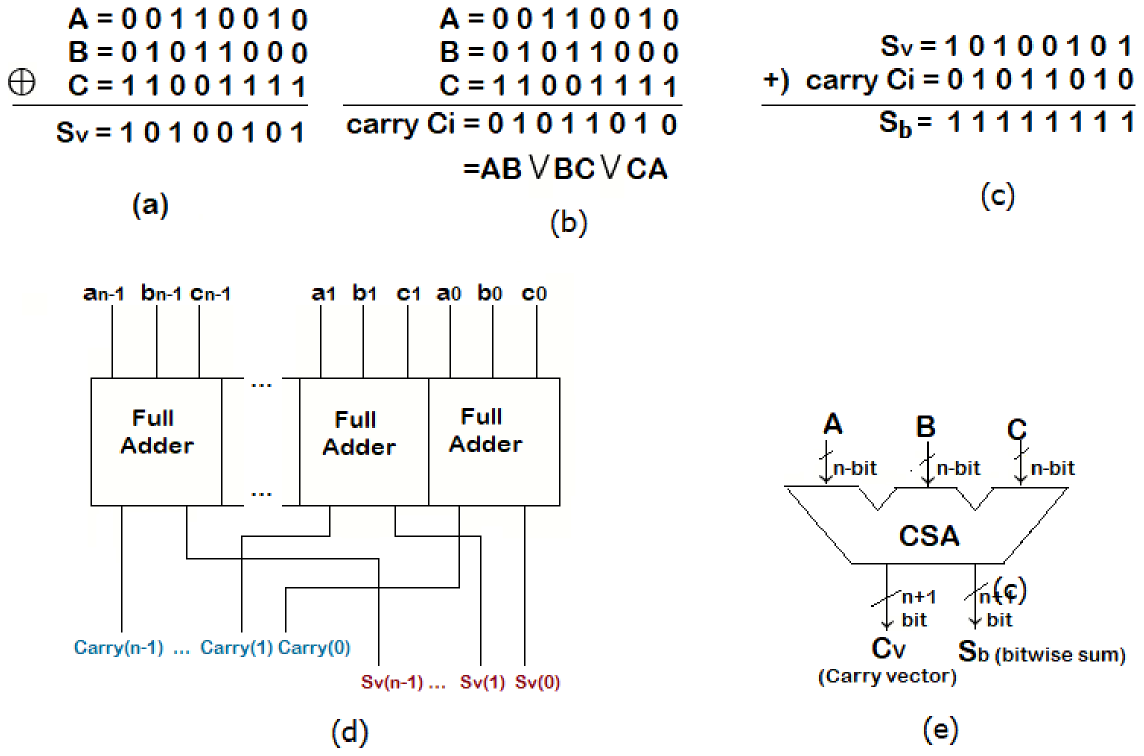


Fig.12 (a) Sum S_v, (b) carry C_i (c) bitwise sum S_b, (d) n-bit CSA and (e) symbol of CSA

12. DESIGN OF A PIPELINE FOR A FIXED-POINT MULTIPLICATION

The PCA and CSA discussed above will be used for implementing the Pipeline stages for a fixed-point Multiplication unit [19]. For the design purpose we consider an example of multiplication of two eight-bit integer numbers A and B and their product P=A×B, where P is a 16-bit number. The fixed-point multiplication can be treated as the sum of eight different partial products depicted in the Fig.13, where P = A×B = $P_7 + P_6 + P_5 + P_4 + P_3 + P_2 + P_1 + P_0$, in this case $\times$ *and* $+$ *are* arithmetic multiplication and addition operations.

								1	**0**	**0**	**1**	**0**	**0**	**1**	**1**	=	A
							×)	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	=	B
								1	0	0	1	0	0	1	1	=	P_0
							0	0	0	0	0	0	0	0	0	=	P_1
						1	0	0	1	0	0	1	1	0	0	=	P_2
					0	0	0	0	0	0	0	0	0	0	0	=	P_3
				1	0	0	1	0	0	1	1	0	0	0	0	=	P_4
			1	0	0	1	0	0	1	1	0	0	0	0	0	=	P_5
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	=	P_6
+)	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	=	P_7
0	1	1	0	0	1	1	1	1	1	1	0	1	1	1	1	=	P

Fig. 13 Product P and partial products P_0, P_1, P_2 , P_3 , P_4, P_5, P_6 and P_7

The partial product P_i , $i =$ 0, 1, 2, ... ,7 is obtained after multiplying the multiplicand A by the i-th bit of the multiplier B. Then we shift the result i bits to the left for i = 0, 1, 2, ⋯ ,7. Thus, we obtain P_i which is (8 + i) bits wide in length with i trailing zeros. The summation of all the partial products starting from P_0 to P_7 is completed with a *Wallace tree* of some Carry-save adders (CSAs) and one CPA which is the final stage.

Taking the generation vector of eight bits for the cases A and B, we assume A= $(a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0)$, B = $(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$ and P = (P_7, P_6, P_5, P_4 ,P_3, P_2, P_1, P_0). With the help of the AND gate and D-Flip-flop, eight numbers of circuit shown from Fig.14 (a) to Fig.14 (h) have been implemented to generate partial products P_7, P_6, P_5,P_4 , P_3, P_2, P_1 and P_0 with respect to $(a_7, a_6, a_5, a_4, a_3, a_2, a_1$ *and* $a_0)$ and one bit from $(b_7, b_6, b_5, b_4, b_3, b_2, b_1$ *and* $b_0)$ each time.

With the help of circuits drawn like CPA, CSA, partial product-circuits for P_0, P_1, P_2 , P_3 , P_4, P_5, P_6 and P_7 and AND gates we have presented a pipeline bearing four stages and five switching circuits which are driven with the rising edge of clock pulse.

In stage-1, all the required partial products P_0, P_1, P_2 , P_3 , P_4, P_5, P_6 and P_7 ranging from 8 bits to 15 bits simultaneously are generated by dint of the diagrams drawn in Fig.14(a) through Fig.14(h). The second stage (stage-2) is made up of two levels of four CSAs merges eight numbers ranging from 8 bits to 15 bits into four numbers ranging from 13 bits to 15 bits. The third stage (stage-3) comprising of two levels of two CSAs merges four numbers ranging from 13 bits to 15 bits into two numbers of 16 bits each. The fourth and final stage (stage-4) consisting of a CPA adds up the two 16 bits numbers to create the final value of 16 bits, the product of A and B, P=A×B.

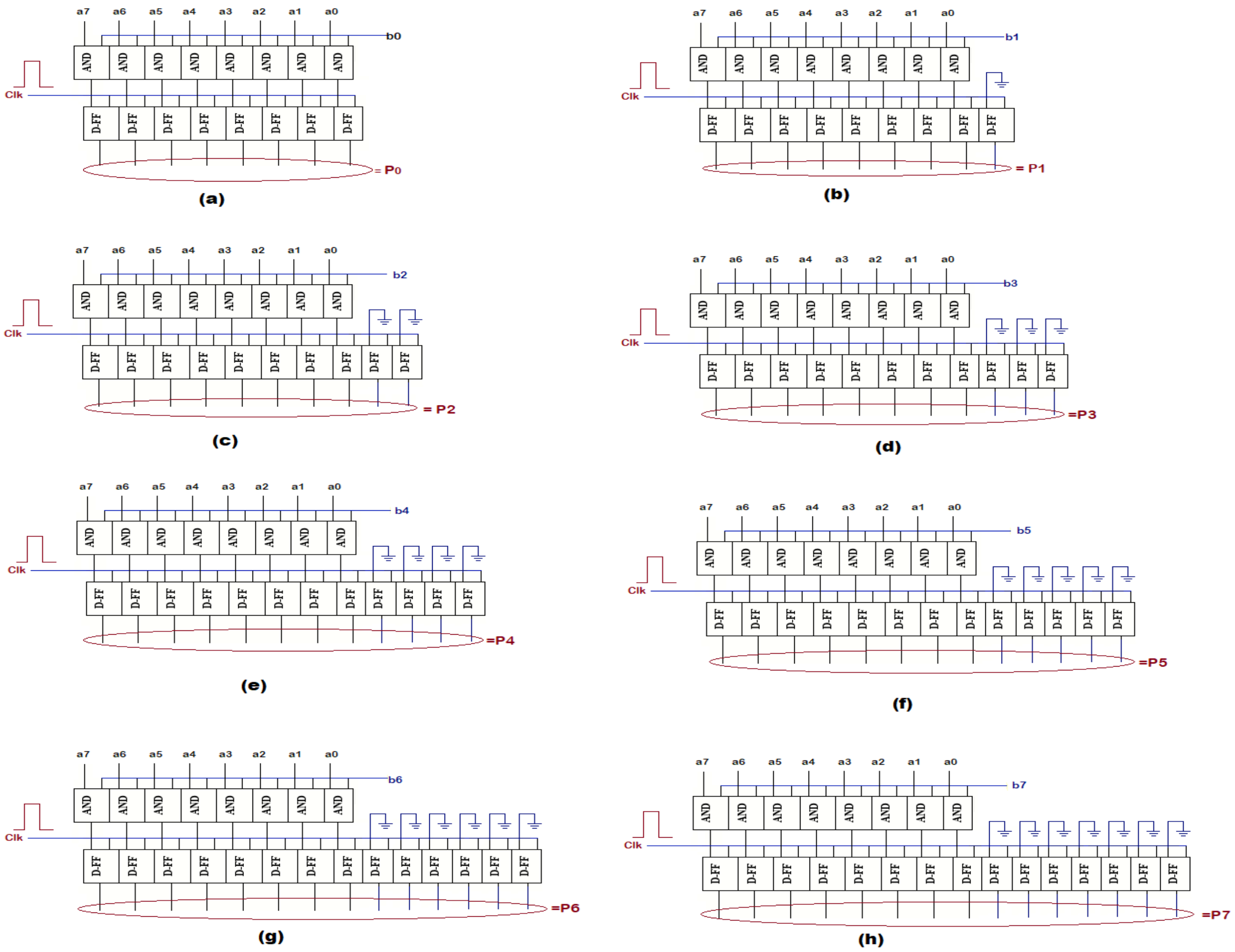


Fig.14 Partial product P_0, P_1, P_2 , P_3 , P_4, P_5, P_6 and P_7 circuits

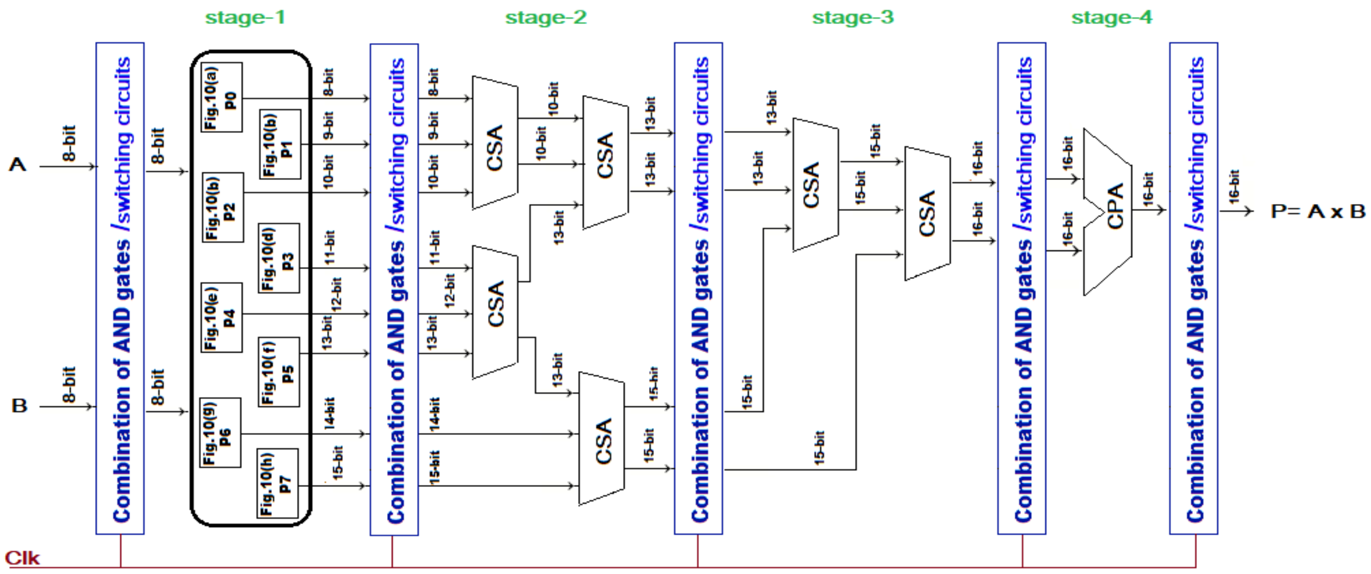


Fig.15 Pipeline for a fixed-point Multiplication

13. DISCUSSION

To find out the delay of a logic gate we require critical voltage V_c which is given in equations (8) and (9), and tunnel junction capacitance C_j . However, assuming at $T = 0K$, the switching delay of a logic gate can be calculated using the approach [6, 7].

$$\text{Delay}=\frac{-eR_t|\ln(P_{error})|}{|V_j|-V_c} \quad \text{..(21)}$$

where V_j is the junction voltage and V_c is the critical (threshold) voltage

The slowest switching happens when the critical voltage V_c becomes lesser than the tunnel junction voltage V_j, $V_c < |V_j|$, but very near to it. This will happen, for example when only V_{in1} is logic 1, resulting V_j =11.8mV for 2-input NOR, the critical voltage of the tunnel junction voltage V_c is 11.58mV. Given that the probability of error change P_{error} =10^{-12}, $R_t = 100K\Omega$. We get a gate delay = 0.07281|ln (P_{error})|ns = 1.675 ns. In the same way we can determine the circuit delays which have been written in Table-2. When charge tunnels through the tunnel junction, the amount of total energy changes before and after the tunneling. So the difference in the amount of energy in the tunnel circuit before and after the tunneling event is determined by $\Delta E = E_{before\ tunnel} - E_{after\ tunnel}$ =e($|V_j| - V_c$) and it is the amount of switching energy consumed in a tunnel event in the tunneling circuit.

We have depicted the switching delay as a function of the switching error probability in Fig. 16(a) and the switching delay as a function of the unit capacitance C is shown in Fig. 16(b).
At first, we have found out the area/element numbers, switching delay, and switching energy consumption for the individual linear threshold gates (using the same methodology as adopted for the Boolean gates). Next, we have calculated the same parameters for more bigger and complex circuits and that are presented in Table-2.

Gate	Area	Delay	Switching Energy
inverter	09 elements	0.022|ln(P_{error}) | ns	10.4 meV
2-input NOR	14 elements	0.072|ln(P_{error})| ns	10.7 meV
2-input OR	14 elements	0.062|ln(P_{error})| ns	10.8 meV
2-input NAND	14 elements	0.080|ln(P_{error}) |ns	10.7 meV
2-input AND	14 elements	0.062|ln(P_{error}) |ns	10.8 meV
2-input XOR	20 elements	0.102|ln(P_{error})| ns	21.2 meV
D latch	65 elements	0.342|ln(P_{error})| ns	53.2meV
Full Adder	40 for sum 70 for carry	0.204|ln(P_{error})| ns 0.186|ln(P_{error})| ns	42.4 meV 32.3 meV
CSA	n×40 for sum n× 70 for carry	0.204|ln(P_{error})| ns 0.186|ln(P_{error})| ns	0.204n meV 0.186n meV
CPA	240 for sum 280 for carry	3.264|ln(P_{error}) | ns 2.976|ln(P_{error}) | ns	678.4 meV 516.8 meV
P_7	1087 elements	0.404|ln(P_{error}) | ns	884.4 meV

Table-2

Time delay for the stage-1 is 0.404|ln(P_{error})| ns, for stage-2 is 0.408|ln(P_{error})| ns, for stage-3 is 0.408|ln(P_{error})| ns, for stage-4 is 3.264|ln(P_{error}) | ns and for the switching circuit if we consider it as an AND gate then its delay time is 0.062|ln(P_{error}) |ns.

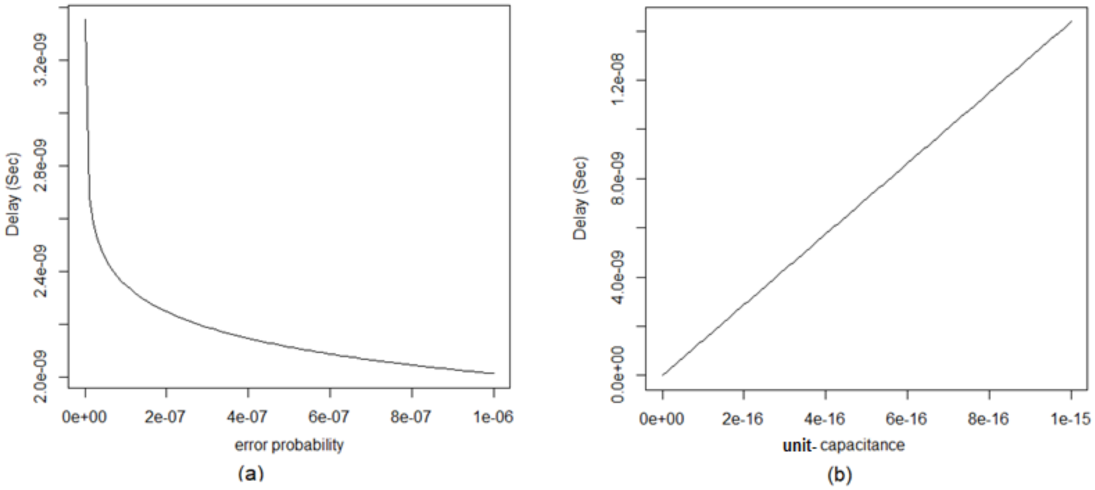


Fig. 16(a) Delay Vs. Error Probability (b) Delay Vs. capacitance

So the time delay to produce first output of product P=A×B for the case of a pipeline is equal to τ_{Total} .

$$\tau_{Total} = \mathbf{0.404|ln(}\boldsymbol{P_{error}}\mathbf{)}\ | + \mathbf{0.408|ln(}\ \boldsymbol{P_{error}})| + \mathbf{0.408|ln(}\ \boldsymbol{P_{error}})| + \mathbf{3.264|ln(}\boldsymbol{P_{error}}\mathbf{)}\ | + 5\times\mathbf{0.062|ln(}\boldsymbol{P_{error}}\mathbf{)}\ |$$

$$=\mathbf{4.794\ |ln(}\ \boldsymbol{P_{error}}\mathbf{)|\ (}ns)$$

If we take the value of P_{error} , for example, 10^{-10} then the time required to produce output of product

$$\tau_{Total} = 11.62ns \quad\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (21)$$

For the context of the pipeline, we are to consider worst case of delay for a stage of four stages in the present situation and this delay is calculated by equation (22)

$$\tau_{max} = \max\{\mathbf{0.404|ln(}\boldsymbol{P_{error}}\mathbf{)}\ |, \mathbf{0.408|ln(}\ \boldsymbol{P_{error}}\mathbf{)|}, \mathbf{0.408|ln(}\ \boldsymbol{P_{error}}\mathbf{)|}, \mathbf{3.264|ln(}\boldsymbol{P_{error}}\mathbf{)}\ |\} + \mathbf{0.062|ln(}\boldsymbol{P_{error}}\mathbf{)}\ |$$

$$=3.326\ |\ln(P_{error})\ |ns \quad\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (22)$$

Therefore the output rate or frequency of the pipeline having stages four is

$$f = \frac{1}{\tau_{max}} = \frac{1}{3.326\ |\ln(P_{error})\ |ns} \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (23)$$

$$=\frac{1}{2.3025850929940ns}$$

If we take the value of P_{error} , for example, 10^{-10} then the output rate will be

$$f=0.434294\times 10^{9}=434.294MHz \cdots\cdots\cdots\cdots\cdots\cdots\cdots (24)$$

From the equation (21) the first output of 16-bit product will be produced after 11.62ns and thereafter the output rate from equation (24) is 434.294MHz.

14. COMPARISON OF SET AND TLG

For a CMOS/TTL logic gate the time delay/processing delay for a gate like NAND, NOR, XOR is 12ns [20], on the contrary the time required for tunneling through a single electron transistor (SET) is approximately 4ns [4] [5]. The XOR gate using conventional logic circuits needs 16 transistors, whereas this function can be implemented with just one SET [1, 2, 4, 5, 10] i.e. number of nodes are reduced to 1 instead of 16. If the error probability is assumed to be 10^{-15} then the delay for the inverter will be 2.3025ns and similarly the other delays for the other gates can be calculated as shown in Table-3. And it is clear that the TLG based circuit must be faster than the SET based circuit when $P_{error}= 10^{-15}$.

Table-3

Circuit name	SET Circuit delay(ns)	TLG based circuit delays (ns)	TLG based circuit delays (ns) when $P_{error}= 10^{-15}$
Inverter	4×2=8	0.022|ln(P_{error}) |	2.3 ns
NAND gate	4×4=16	0.080|ln(P_{error}) |	2.76 ns
OR gate	4×1=4	0.062|ln(P_{error})|	2.14 ns
XOR gate	4×1=4	0.102|ln(P_{error})|	3.5 ns
D-Flip-flop	16× 2=32	0.342|ln(P_{error})|	11.81 ns

15. CONCLUSION

In this paper, first how an electron tunnels through a single electron transistor and an inverter is discussed. A generic Linear Threshold logic gate implementation from which we have derived a family of logic gates like AND, NAND, OR, NOR, XOR which are elaborately discussed. All the gates along with a full adder and a D Flip-flop are implemented and are verified by means of simulation using SIMON. The number of elements for logic gates, their delays, power consumed for them are also given in a tabular form. By dint of all these LTG gates and an inverter, a pipeline of complex circuit called "Pipeline for a fixed-point Multiplication" having four stages, has been presented. It is an 8 by 8 bit multiplication pipeline producing 16-bit output. Among the four stages, stage-4 takes more time to execute with respect to other three stages. In single electron tunneling technology the logic gates are at least 3 times faster than CMOS based logic gates. The atmosphere temperature should be close to 0K in real operation.

REFERENCES

[1] N . Kuwamura , K. Taniguch i, and C. 1-lamaguchi, "Simulation of single-electron logic circuits," IEICE Trans, vol. J77-C-II, no.5, pp.22 1-228. May 1994.

[2] Souvik Sarkar1, Anup Kumar Biswas2, Ankush Ghosh1, Subir Kumar Sarkar1 "Single electron based binary multipliers with overflow detection", International Journal of Engineering, Science and Technology Vol. 1, No. 1, 2009, pp. 61-73

[3] A. K. Biswas and S. K. Sarkar: "An arithmetic logic unit of a computer based on single electron transport system": Semiconductor Physics, Quantum Electronics & Opt-Electronics. 2003. Vol 6. No.1, pp 91-96

[4] A.K. Biswas and S. K. Sarkar: "Error Detection and Debugging on Information in Communication System Using Single Electron Circuit Based Binary Decision Diagram." Semiconductor Physics Quantum electronics and opt electronics, Vol. 6, pp.1-8, 2003

[5] Alexander N. Korotkov, "Single-electron logic and memory devices" INT. ELECTRONICS, 1999, Vol. 86, No. 5, 511- 547

[6] Casper Lageweg, Student Member, IEEE, Sorin Cot¸ofan˘a, Senior Member, IEEE, and Stamatis Vassiliadis, Fellow, IEEE "Single Electron Encoded Latches and Flip-Flops" IEEE TRANSACTIONS ON NANOTECHNOLOGY, VOL. 3, NO. 2, JUNE 2004

[7] C. Lageweg, S. Cot¸fan˘a, and S. Vassiliadis, "A linear threshold gate implementation in single electron technology," in IEEE Computer Society VLSI Workshop, Apr. 2001, pp. 93– A. Korotkov, "Single-electron logic and memory devices," Int. J. Electron., vol. 86, no. 5, pp. 511–547, 1999.

[8] K. Likharev, "Single-electron devices and their applications," Proc. IEEE, vol. 87, pp. 606–632, Apr. 1999.

[9] A. Korotkov, R. Chen, and K. Likharev, "Possible performance of capacitively coupled single-electron transistors in digital circuits," J. Appl. Phys., vol. 78, pp. 2520–2530, Aug. 1995.

[10] J. R. Tucker, "Complementary digital logic based on the "Coulomb blockade"," J. Appl. Phys., vol. 72, no. 9, pp. 4399–4413, Nov. 1992.

[11] A. Korotkov and K. Likharev, "Single-electron-parametron-based logic devices," J. Appl. Phys., vol. 84, no. 11, pp. 6114–6126, Dec. 1998.

[12] C. Wasshuber, H. Kosina, and S. Selberherr, "SIMON—A simulator for single-electron tunnel devices and circuits," IEEE Trans. Computer- Aided Design, vol. 16, pp. 937–944, Sept. 1997.

[13] A. B. Zorin, S. V. Lotkhov, H. Zangerle, and J. Niemeyer, "Coulomb blockade and cotunneling in single electron circuits with on-chip resistors:Toward the implementation of the R pump," J. Appl. Phys., vol. 88, no. 5, pp. 2665–2670, Sept. 2000.

[14] S. Muroga, Threshold Logic and Its Applications. New York: Wiley, 1971.

[15] T. Oya, T. Asai, T. Asai, T. Fukui, and Y. Amemiya, "A majority-logic device using and irreversible single-electron box," IEEE Trans. Nanotechnol.,vol. 2, pp. 15–22, Mar. 2003.

[16] L. Guo, E. Leobandung, and S. Chou, "A silicon single-electron transistor memory operating at room temperature," Science, vol. 275, pp.649–651, 1997.

[17] Z. Durrani, A. Irvine, and H. Ahmed, "Coulomb blockade memory using integrated single-electron transistor/metal–oxide–semiconductor transistor gain cells," IEEE Trans. Electron Devices, vol. 47, pp. 2334–2339, Dec. 2000.

[18] J. Millman and C. C. Halkias; Integrated Electronics- Analog and Digital Circuits and Systems_ McGraw Hill Education; 2 edition

[19] Kai Hwang and Naresh Jetwani, " Advanced Computer Architecture parallelism, scalability, programability" chapter 6, 2nd edition McGraw Hill

[20] Millman's Electronic Devices & Ciruits 4th Edition (English, Paperback, Millman Jacob)

BIOGRAPHY

Dr. Anup Kumar Biswas is an Assistant Professor in Commuter Science and Engineering in Kalyani Govt. Engineering College. He is awarded his PhD[Engg.] from Jadavpur University. He has engaged in teaching and research activities since the last 16 years. His Specialization field is Single Electron Device. Biswas has published several papers in various national, international conferences and journals.