Design and Implementation of 4 Bit Carry Skip Adder Using Nmos and Pmos Transmission Gate

Ashutosh Pandey, Harshit Singh, Vivek Kumar Chaubey and Utkarsh Jaiswal

February 5, 2020

CHAPTER-1

INTRODUCTION

In the field of electronics, a digital circuit that performs addition of numbers is called an adder or summer. In various kinds of processors like computers, adders have many applications in the arithmetic logic units, as well as in other parts, where these are used to compute table indices, addresses and similar operations. Mostly, the common adders operate on binary numbers, but they can also be constructed for many other numerical representations, such as excess-3 or binary coded decimal (BCD). It is insignificant to customize the adder into an adder-subtractor unit in situations where negative numbers are represented by one's or two's complement. The usage of power efficient VLSI circuits is required to satiate the perennial need for mobile electronic devices. The calculations in these devices ought to be performed using area efficient and low power circuits working at higher speed. The most elementary arithmetic operation is addition; and the most basic arithmetic component of the processor is the adder. Depending upon the delay, area and power consumption requirements; certain adder implementations such as ripple carry, carry-skip, carry select and carry look ahead are available. When large bit numbers are used, the ripple carry adder (RCA) is not very efficient. With the bit length, there is a linear increase in delay. For n-bit adder, it implements area of O(n) and a delay of O(n). A delay of O(log n) is present in the carry look ahead adder and it uses O(nlogn) area. On the contrary, the carry select adders and the carry skip adders have a delay of O(vn) and use an area of O(n). The carry skip adders dissipate lower power than the others because of their short wire lengths and low transistor counts .

1.1) CARRY SKIP ADDER

The carry skip adder is a compromise between a ripple carry adder and a CLA adder as shown in Fig.3. In carry skip adder, the data to be added is divided into blocks of variable size. Ripple carry produces the sum bit and the carry within each block. The main principle of carry Skip Adder is carry computation i.e. skipping carry over groups of consecutive adder stages as shown in Figs.5 to 8. This thus helps to reduce the delay . The Boolean expressions for sum and carry are: Carry Propagate: $Pi = Ai \oplus Bi$; Sum: $Si = Pi \oplus Ci$; Carry Out: $Ci+1= Ai\ Bi + Pi\ Ci$ Fig.2. Full adder. It must be noticed that if $Ai = Bi$ then $Pi = 0$, which makes carry out, $Ci+1$, depend only on Ai and Bi $\Diamond$ $Ci+1= Ai\ Bi$ • if $Ai = Bi = 0$, then $Ci+1 = 0$ • if $Ai = Bi = 1$, then $Ci+1 = 1$ Else if $Ai \neq Bi$, then $Pi = 1$ $\Diamond$ $Ci+1= Ci$ •

Hence, the new value Ci+1 does not need to be computed if each Ai ≠ Bi in a group or block. In this case, the carry-in (Ci) of the block can be directly propagated to the next block. For some „i" in the group, • If Ai = Bi = 1, then carry will be generated and propagated up to the output of that block. • If Ai = Bi = 0, then carry will not be propagated by that bit location. The fundamental idea of a carry-skip adder is to check if in each group all Ai ≠ Bi and whenever this happens, it enables the carry-in (Ci) to skip the block or group as shown in Fig.4. Generally, a block-skip delay and the propagation delay of a carry to the next bit position can be different .

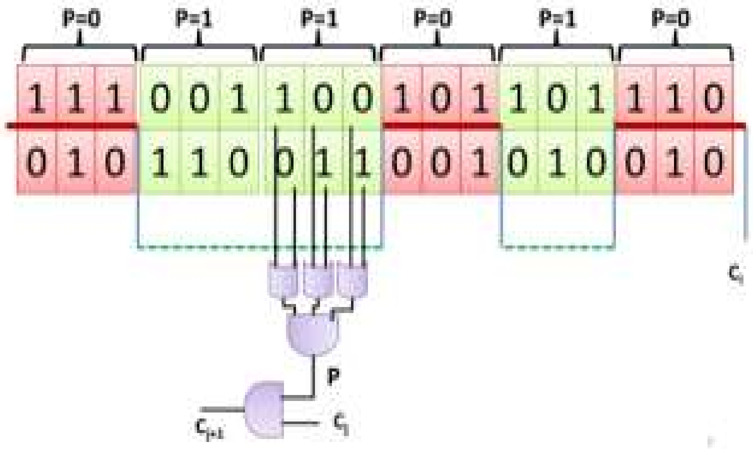


Fig 1.1 Carry Skip Example

CHAPTER-2

LITERATURE SURVEY/RELATED WORK

Our need is to obtain adders having performance on the higher side, better cost-effectiveness and minimum power consumption. The three most widely accepted standards to measure the quality of a circuit and to compare various styles of circuits are delay, area and power dissipation. We need to implement adders having minimized power dissipation and maximized speed due to the fact that most digital circuitry is made up of simple and/or complex gates. To optimize different design parameters, various adder implementations have been developed. Mostly, the adder implementations tend to trade off the power and performance. A regular parallel adder layout, also known as the Brent-Kung adder '82, was one of the earliest adder implementations of this kind . This is a variation of the basic carry look-ahead adder. To reduce design and implementation costs, they emphasized the need for regularity in VLSI circuits. An areatime optimal adder design using three types of adder cells was proposed by Wei-Thompson , they were; white cells, black cells and driver cells. The white and black cells are pretty similar to those used in Brent-Kung adder. To limit the number of bits in the final stage, the n-bit adder was divided into ascending and descending halves. The algorithm terminates in an unbalanced binary tree having a delay consuming an area O(nlogn). The design of a one-level carryskip adder with an approach very much similar to that of Wei-Thompson was presented by Kantabutra '93 . In contrast to the approach of Wei-Thompson, this design terminates in a symmetrical binary tree of adders[1]. Towards the middle of the adder, the fan-in to the carry-skip logic increases linearly. In a two-level carry-skip adder, the adder stage as a whole is divided into a number of sections, each having a number of RCA blocks of linearly increasing length. It is presented in . The delay is reduced by these adders at the cost of an increase in area and a less regular layout. A technique of choice in most Processor design is Conventional Static CMOS. Alternatively, for Low Power applications , Static Pass Transistor circuit has been suggested. When clocked carefully, the dynamic circuit can be used in Low International Journal of VLSI System Design and Communication Systems Volume.03, IssueNo.07, September-2015, Pages: 1116-1121 Power, High Speed Systems also. For its ability to reduce power dissipation, reversible logic has received great attention in the recent years. For their construction, reversible logic circuits are required by Quantum arithmetic components. It was demonstrated by R. Landauer in 1960 that high technology circuits and systems built using irreversible hardware result in energy dissipation because of information loss. In accordance with Landauer's principle, kTln2 joules of energy is dissipated due to the loss of one bit of information where „T" is set as the circuit temperature and 'k' is the Boltzmann's constant (approx. 1.38 x 10-23 Joules per Kelvin) . Later in 1973, Bennett presented that a circuit must be constructed from reversible hardware in order to avoid kTln2 joules of energy dissipation . When compared to other design styles, it has been observed that Reversible logic using Fredkin gate Full Adder (FFA) exhibit better Low-Power and Speed characteristics .The other one is carry skip adder (CSKA) in which the carry is skipped using a multiplexer . When every inputs are in propagation condition, the mux selects the carry for next state as previous state without waiting for addition of last bit. A much more developed architecture of CSKA was developed in that have very less delay when compared to conventional CSKA. The model is CICSKA (Concatenation Incrementation CSKA) which combines the concatenation and incrementation schemes to conventional CSKA and carry is skipped by using AOI and OAI logic. This structure focus on increasing speed of operation and decreasing the area. The use of AND OR invert logic as carry skip will also help to decrease area as it has less number of transistors when compared to multiplexer.

2.1) HALF ADDER

The half adder functions by adding two single binary digits A and B and giving two outputs, sum (S) and carry (C) as shown in Fig.1. In multi-digit addition, if overflow is generated from one bit to another, then this overflow is carry. The value of the sum in denary is 2C + S. The simplest halfadder design is implemented using an XOR gate for S and an AND gate for C. Full adder can be obtained by joining two half adders and by combining their carry outputs by OR gate.

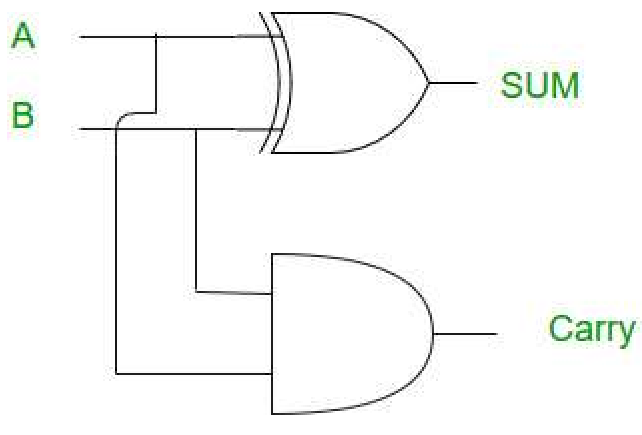
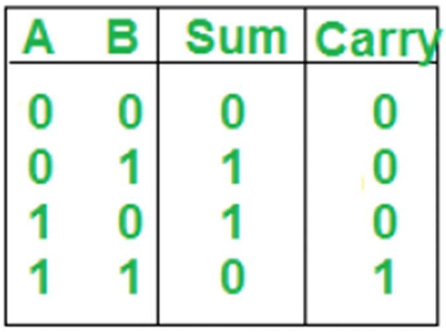


Fig. 2.1 Half Adder

Truth table of Half adder

1-bit adder can be easily implemented with the help of the XOR Gate for the output 'SUM' and an AND Gate for the 'Carry'. When we need to add, two 8-bit bytes together, we can be done with the help of a full-adder logic. The half-adder is useful when you want to add one binary digit quantities. A way to develop a two-binary digit adders would be to make a truth table and reduce it. When you want to make a three binary digit adder, do it again. When you decide to make a four digit adder, do it again. The circuits would be fast, but development time is slow as shown in fig.



A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

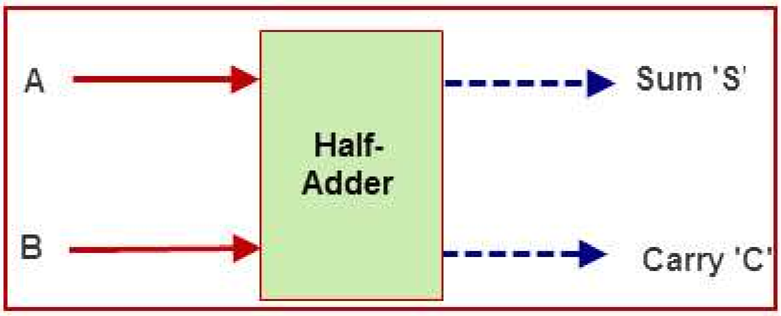


Fig 2.11 Truth table and Block Diagram Of Half Adder

The ALU (arithmetic logic circuitry) of a computer uses half adder to compute the binary addition operation on two bits. Half adder is used to make full adder as a full adder requires 3 inputs, the third input being an input carry i.e. we will be able to cascade the carry bit from one adder to the other.

2.2) FULL ADDER

A full adder has three inputs and two outputs and accounts for values carried in as well as out as shown in Fig.2. The three inputs of a one-bit full adder are often written as A, B, and Cin; A and B are the operands which are to be added, and Cin is a carry bit which is carried in from the previous less significant bit addition. The full adder is generally a basic component in a cascade of adders, for adding 8, 16, 32, etc. bit binary numbers. A two-bit output is produced by the circuit, output carry and sum usually represented by Cout and S. A full adder is a logical circuit that performs an addition operation on three one-bit binary numbers. The full adder produces a sum of the three inputs and carry value. It can be combined with other full adders (see below) or work on its own.

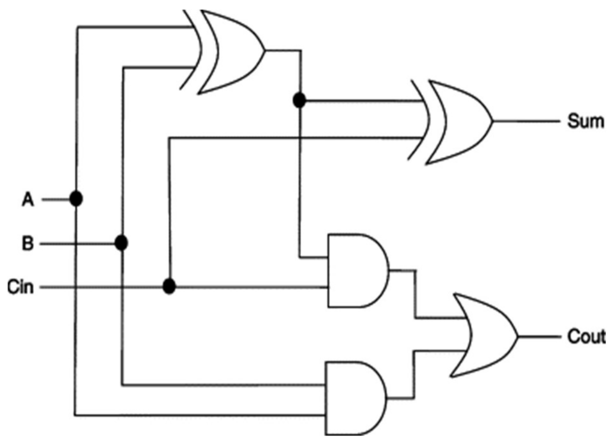


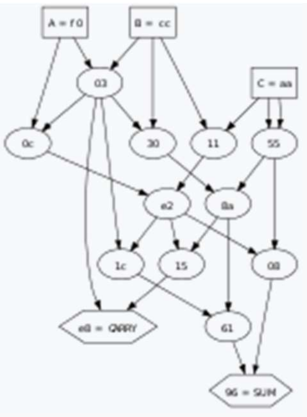
Fig. 2.2 Full Adder

A **full adder** adds binary numbers and accounts for values carried in as well as out. A one-bit full-adder adds three one-bit numbers, often written as A, B, and C_{in}; A and B are the operands, and C_{in} is a bit carried in from the previous less-significant stage.[2] The full adder is usually a component in a cascade of adders, which add 8, 16, 32, etc. bit binary numbers. The circuit produces a two-bit output. Output carry and sum typically represented by the signals C_{out} and S, where the sum equals $2C_{out} + S$.

A full adder can be implemented in many different ways such as with a custom transistor-level circuit or composed of other gates. One example implementation is with $S = A \oplus B \oplus C_{in}$ and $C_{out} = (A \cdot B) + (C_{in} \cdot (A \oplus B))$.

In this implementation, the final OR gate before the carry-out output may be replaced by an XOR gate without altering the resulting logic. Using only two types of gates is convenient if the circuit is being implemented using simple integrated circuit chips which contain only one gate type per chip.

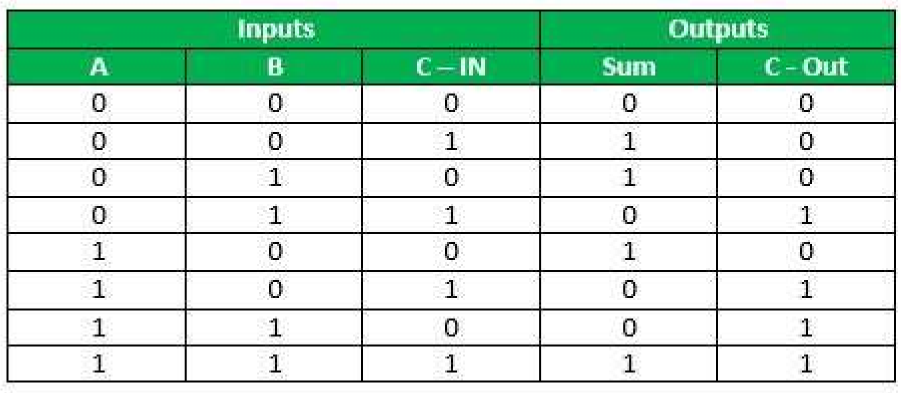
A full adder can also be constructed from two half adders by connecting *A* and *B* to the input of one half adder, then taking its sum-output *S* as one of the inputs to the second half adder and C_{in} as its other input, and finally the carry outputs from the two half-adders are connected to an OR gate. The sum-output from the second half adder is the final sum output (*S*) of the full adder and the output from the OR gate is the final carry output (C_{out}). The critical path of a full adder runs through both XOR gates and ends at the sum bit *s*. Assumed that an XOR gate takes 1 delays to complete, the delay imposed by the critical path of a full adder is equal to



NOR Full adder

Fig 2.21 Full Adder Using NOR Gate

The critical path of a carry runs through one XOR gate in adder and through 2 gates (AND and OR) in carry-block and therefore, if AND or OR gates take 1 delay to complete the one cycle.



Inputs			Outputs	
A	B	C – IN	Sum	C - Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Fig 2.22 Truth Table Of Full Adder

From the above truth-table, the full adder logic can be implemented. We can see that the output S is an EXOR between the input A and the half-adder SUM output with B and CIN inputs. We must also note that the COUT will only be true if any of the two inputs out of the three are HIGH.

Thus, we can implement a full adder circuit with the help of two half adder circuits. The first will half adder will be used to add A and B to produce a partial Sum. The second half adder logic can be used to add CIN to the Sum produced by the first half adder to get the final S output. If any of the half adder logic produces a carry, there will be an output carry. Thus, COUT will be an OR function of the half-adder Carry outputs. Take a look at the implementation of the full adder circuit shown below.

Implementation of Full Adder using Half Adders

2 Half Adders and a OR gate is required to implement a Full Adder.

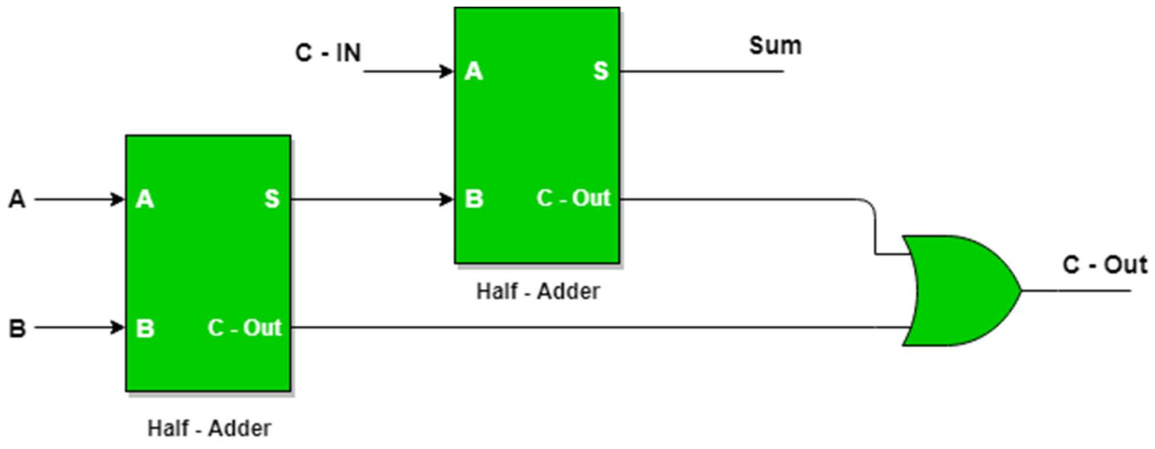


Fig 2.23 Full adder using Half Adder

With this logic circuit, two bits can be added together, taking a carry from the next lower order of magnitude, and sending a carry to the next higher order of magnitude.

2.3) Conventional Carry Skip Adder

In conventional Carry skip adder [2] addition is performed in multiple stages. Each stage consists of a RCA block, a multiplexer and a carry prediction unit. RCA is used to find the sum of each stage. It takes the input bits of two numbers A and B, and generates the sum. The designed model uses a 4-bit RCA block. When all bits of corresponding stage is in propagation condition (Eq.1) the carry prediction unit will generate one as output.
The operation of AOI and OAI stages can be explained by two cases, first case is that if carry is propagated from previous state and second is when carry is not propagated. Consider the case in which carry is propagated from previous stage and all input bits are in carry propagation condition, then all intermediate bits will be 1, hence the output of

AND gate will be 1. This one will be propagated to AOI logic. The first gate of AOI compound gate is AND gate. This AND gate will generate 1 only if there is a carry from previous stage is 1. The second gate of AOI compound gate is NOR. So the input to this NOR will be one so it will generate output 0, which is the inverted carry from previous stage. This is again inverted by a NOT gate and given to incrementation block. So the time of propagation of carry in this case (eq.4) will be the sum of time taken by and gate TAND and time taken by skip logics which can be TOAI or TAOI logic.

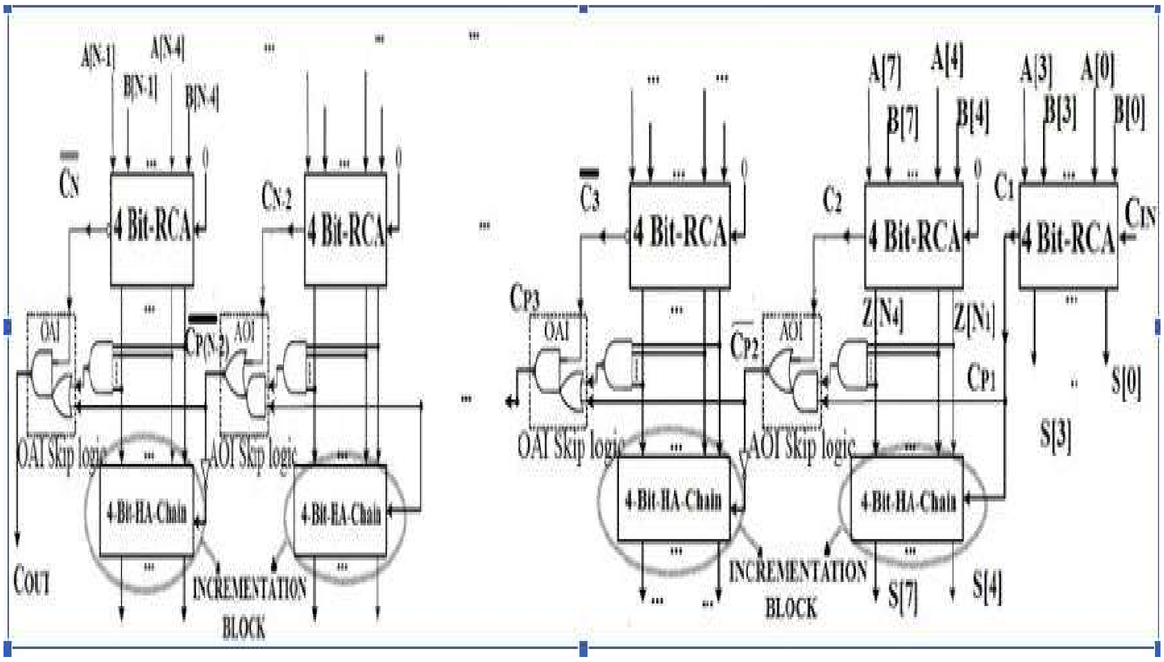


Fig 2.3 Block diagram of Conventional Carry Skip Adder

A 2 to 1 multiplexer is used as the carry skip logic. The output of carry prediction unit is fed to the mux. If the output of carry prediction unit is high the mux selects the previous carry Cp without waiting for the carry generated from the RCA block. When two numbers A and B are added the worst case delay happens when both numbers are in carry propagation condition. In this case the carry will propagate from one stage to the input of next stage without waiting for the carry generation by RCA. The maximum time taken for carry generation (Eq.2) is the propagation time of multiplexer and the time for carry prediction. If any of the input to RCA is not in propagation condition the multiplexer will select the first input that is the carry from the RCA block. In this case the next stage has to wait until the carry is generated by the RCA block of previous stage. The maximum time taken for carry generation of if the input bits are not in propagation condition (Eq.3) is the sum of time for carry propagation through RCA block and multiplexer. The propagation time through RCA TRCA depends on from where the carry is generated. If the carry is generated in the first input bit, then it have to propagate through four full adders that is eight XOR gates which makes the time for addition very high. The disadvantage of this adder is that it can skip carry only when all bits are in propagation condition. It is not sure that always every input bits are in carry propagation condition. If all input bits are in propagation condition and carry from previous stage is zero the output of AND gate in AOI will be zero and hence it have to wait to receive the third input which is from RCA. Same condition exists if the input bits are not in propagation condition. Similar case occurs for OAI block also. The time required for carry propagation if all the intermediate results are not one (Eq.5) will be the sum of time taken for carry propagation through the RCA chain TRCA and the time taken for carry to propagate through the skip logic.

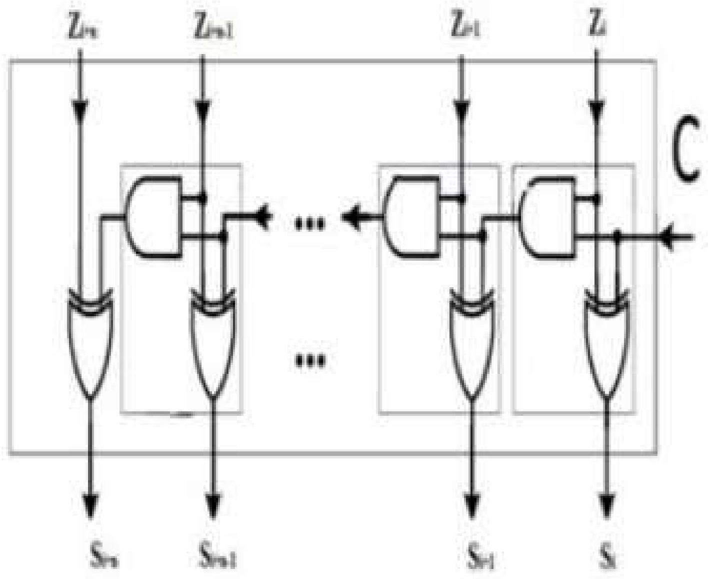


Fig 2.31 Incremental Block Of CSA

Advantage of this model is that for finding carry output of next stage the carry from incrementation block is not required. So the delay for generation of final result does not depends on the delay for carry propagation from one block to other. Also all the intermediate results can be calculated simultaneously by RCA without waiting for the carry from previous block. The carry is calculated based on intermediate results and carry from previous block. The disadvantage of this model is that if all inputs are not in propagation condition then all the intermediate results will not be one, in such cases the skip logic have to wait for the carry from RCA block.

CHAPTER 3

DESIGN SPECIFICATIONS

3.1) 4bit carry skip adder

A carry-skip adder (also known as a carry-bypass adder) is an adder implementation that improves on the delay of ripple carry adder with little effort compared to other adders. The improvement of the worst-case delay is achieved by using several carry-skip adders to form a block-carry-skip adder. The Carry skip adder is a skip the logic in propagation of carry and its implementation is to speed up the additional operation and to adding the propagation of carry bit around portion of entire adder. In this paper the effect of change in architecture of carry skip adder in terms of power dissipation, area, delay, is analysed. The schematic diagram and additional truth table for carry skip adder. The observed result indicates that the power dissipation, area, delay and other parameters vary with change in transistor technology. And this paper analysis the behaviour of carry skip adder in pass transistor logic and conventional CMOS logic architecture using cadence tool technology. And supply voltage 180nm and 1.8v are considered from experimental result. The result shows that carry skip adder is implemented in pass transistor logic architecture perform better then comparison to CMOS logic configuration, reference, mainly in terms of area, delay, power dissipation, and number of transistor counts.

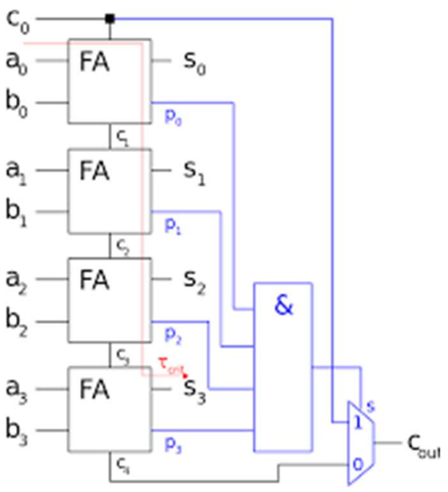


Fig. 3.1 4 Bit Carry Skip Adder

3.2) CMOS logic

Complementary metal oxide semiconductor (CMOS), also known as complementary-symmetry metal–oxide–semiconductor (COS-MOS), is a type of MOSFET (metal–oxide–semiconductor field-effect transistor) fabrication process that uses complementary and symmetrical pairs of p-type and n-type MOSFETs for logic functions.[1] CMOS technology is used for constructing integrated circuit (IC) chips, including microprocessors, microcontrollers, memory chips (including CMOS BIOS), and other digital logic circuits. CMOS technology is also used for analog circuits such as image sensors (CMOS sensors), data converters, RF circuits (RF CMOS), and highly integrated transceivers for many types of communication.

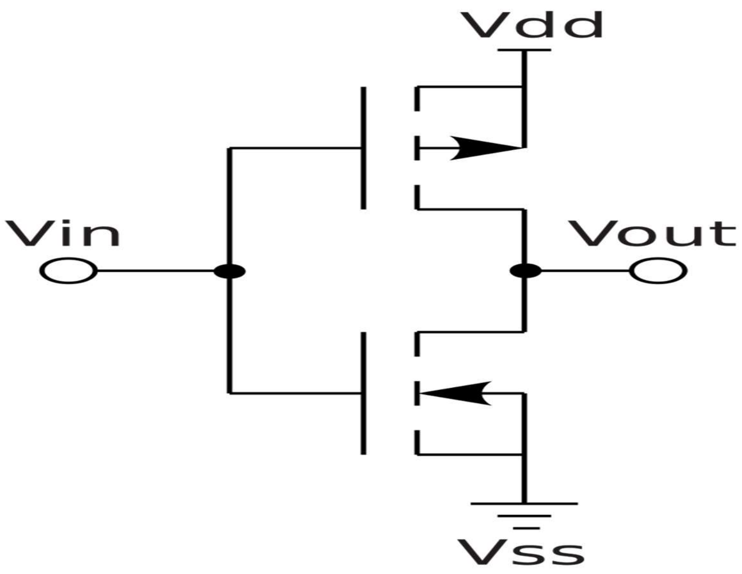


Fig.3.2 CMOS Circuit []

CMOS" refers to both a particular style of digital circuitry design and the family of processes used to implement that circuitry on integrated circuits (chips). CMOS circuitry dissipates less power than logic families with resistive loads. Since this advantage has increased and grown more important, CMOS processes and variants have come to dominate, thus the vast majority of modern integrated circuit manufacturing is on CMOS processes. CMOS logic consumes over 7 times less power than NMOS logic, and about 100,000 times less power than bipolar transistor-transistor logic (TTL).

CMOS circuits use a combination of p-type and n-type metal–oxide–semiconductor field-effect transistor (MOSFETs) to implement logic gates and other digital circuits. Although CMOS logic can be implemented with discrete devices for demonstrations, commercial

CMOS products are integrated circuits composed of up to billions of transistors of both types, on a rectangular piece of silicon of between 10 and 400 mm^2.

CMOS always uses all enhancement-mode MOSFETs (in other words, a zero gate-to-source voltage turns the transistor off).

3.3) Inversion

CMOS circuits are constructed in such a way that all P-type metal–oxide–semiconductor (PMOS) transistors must have either an input from the voltage source or from another PMOS transistor. Similarly, all NMOS transistors must have either an input from ground or from another NMOS transistor. The composition of a PMOS transistor creates low resistance between its source and drain contacts when a low gate voltage is applied and high resistance when a high gate voltage is applied. On the other hand, the composition of an NMOS transistor creates high resistance between source and drain when a low gate voltage is applied and low resistance when a high gate voltage is applied. CMOS accomplishes current reduction by complementing every n MOSFET with a p MOSFET and connecting both gates and both drains together. A high voltage on the gates will cause the n MOSFET to conduct and the p MOSFET not to conduct, while a low voltage on the gates causes the reverse. This arrangement greatly reduces power consumption and heat generation. However, during the switching time, both MOSFETs conduct briefly as the gate voltage goes from one state to another. This induces a brief spike in power consumption and becomes a serious issue at high frequencies.

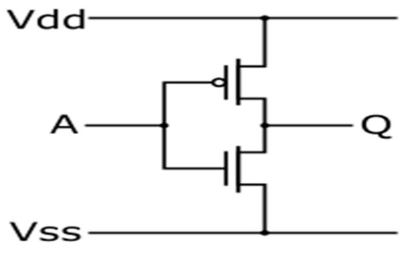


Fig. 3.3 Inversion

3.4) PMOS logic

P-type metal-oxide-semiconductor logic uses p-channel (+) metal-oxide-semiconductor field effect transistors (MOSFETs) to implement logic gates and other digital circuits. PMOS transistors operate by creating an inversion layer in an n-type transistor body. This inversion layer, called the p-channel, can conduct holes between p-type "source" and "drain" terminals.

The p-channel is created by applying voltage to the third terminal, called the gate. Like other MOSFETs, PMOS transistors have four modes of operation: cut-off (or subthreshold), triode, saturation (sometimes called active), and velocity saturation.

While PMOS logic is easy to design and manufacture (a MOSFET can be made to operate as a resistor, so the whole circuit can be made with PMOS FETs), it has several shortcomings as well. The worst problem is that there is a direct current (DC) through a PMOS logic gate when the PUN is active, that is, whenever the output is high, which leads to static power dissipation even when the circuit sits idle.

Also, PMOS circuits are slow to transition from high to low. When transitioning from low to high, the transistors provide low resistance, and the capacitive charge at the output accumulates very quickly (similar to charging a capacitor through a very low resistance). But the resistance between the output and the negative supply rail is much greater, so the high-to-low transition takes longer (similar to discharge of a capacitor through a high resistance). Using a resistor of lower value will speed up the process but also increases static power dissipation.

Additionally, the asymmetric input logic levels make PMOS circuits susceptible to noise.[1]

Most PMOS integrated circuits require a power supply of 17-24 volt DC.[2] The Intel 4004 PMOS microprocessor, however, uses PMOS logic with polysilicon rather than metal gates allowing a smaller voltage differential. For compatibility with TTL signals, the 4004 uses positive supply voltage V_{SS}=+5V and negative supply voltage V_{DD} = -10V.[3]

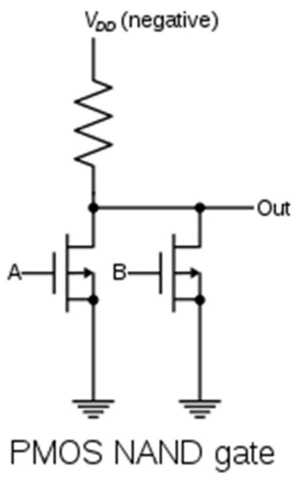
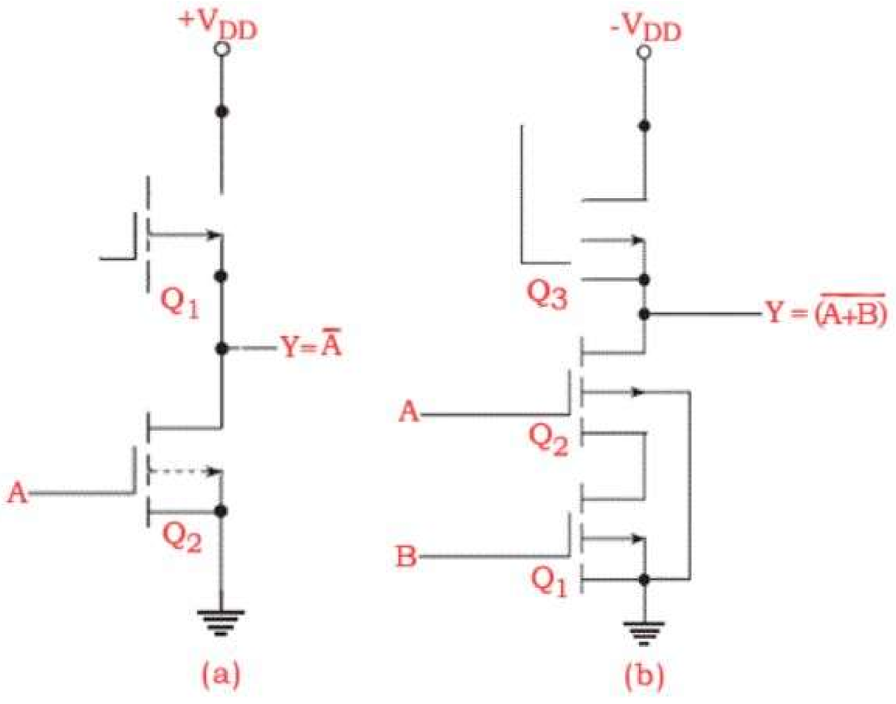


Fig. 3.4 PMOS NAND Gate

The **PMOS logic** family uses P-channel MOSFETS. Figure (a) shows an inverter circuit using PMOS logic (not to be confused with a power inverter). MOSFET Q_1 acts as an active load for the MOSFET switch Q_2. For the circuit shown,
GND and $-V_{DD}$ respectively represent a logic '1' and a logic '0' for a positive logic system. When the input is grounded (i.e. logic '1'), Q_2 remains in cut-off and $-V_{DD}$ appears at the output through the conducting Q_1. When the input is at $-V_{DD}$ or near $-V_{DD}$, Q_2 conducts and the output goes to near-zero potential (i.e. logic '1'). Figure (b) shows a PMOS logic based two-input NOR gate. In the logic arrangement of Fig.(b), the output goes to logic '1' state (i.e. ground potential) only when both Q_1 and Q_2 are conducting.
This is possible only when both the inputs are in logic '0' state. For all other possible input combinations, the output is in logic '0' state, because, with either Q_1 or Q_2 nonconducting, the output is nearly $-V_{DD}$ through the conducting Q_3. The circuit of Fig.(b) thus behaves like a two-input NOR gate in positive logic. It may be mentioned here that the MOSFET being used as load [Q_1 in Fig. (a) and Q_3 in Fig. (b)] is designed so as to have an ON-resistance that is much greater than the total ON-resistance of the MOSFETs being used as switches [Q_2 in Fig. (a) and Q_1 and Q_2 in Fig(b).]



(a) PMOS logic inverter and (b) PMOS logic two-input NOR

Fig 3.41 PMOS INVERTER

3.5) NMOS logic

N-type metal-oxide-semiconductor logic uses n-type (-) MOSFETs (metal-oxide-semiconductor field-effect transistors) to implement logic gates and other digital circuits. These n MOS transistors operate by creating an inversion layer in a p-type transistor body. This inversion layer, called the n-channel, can conduct electrons between n-type "source" and "drain" terminals. The n-channel is created by applying voltage to the third terminal, called the gate. Like other MOSFETs, n MOS transistors have four modes of operation: cut-off (or subthreshold), triode, saturation (sometimes called active), and velocity saturation.

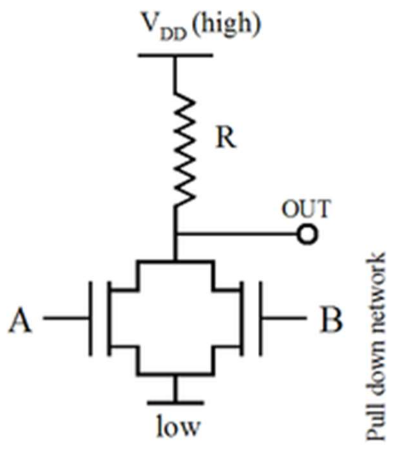


Fig. 3.6 Pull Down Network

MOS stands for metal-oxide-semiconductor, reflecting the way MOS-transistors were originally constructed, predominantly before the 1970s, with gates of metal, typically aluminium. Since around 1970, however, most MOS circuits have used self-aligned gates made of polycrystalline silicon. These silicon gates are still used in most types of MOSFET based integrated circuits, although metal gates (Al or Cu) started to reappear in the early 2000s for certain types of high speed circuits, such as high performance microprocessors.

The MOSFETs are n-type enhancement mode transistors, arranged in a so-called "pull-down network" (PDN) between the logic gate output and negative supply voltage (typically the ground). A pull up (i.e. a "load" that can be thought of as a resistor, see below) is placed between the positive supply voltage and each logic gate output. Any logic gate, including the logical inverter, can then be implemented by designing a network of parallel and/or series circuits, such that if the desired output for a certain combination of boolean input values is zero (or false), the PDN will be active, meaning that at least one transistor is allowing a current path between the negative supply and the output. This causes a voltage drop over the load, and thus a low voltage at the output, representing the *zero*.

More complex logic functions such as those involving AND and OR gates require manipulating the paths between gates to represent the logic. When a path consists of two transistors in series, both transistors must have low resistance to the corresponding supply voltage, modelling an AND. When a path consists of two transistors in parallel, either one or both of the transistors must have low resistance to connect the supply voltage to the output, modelling an OR.

Shown on the right is a circuit diagram of a NAND gate in CMOS logic. If both of the A and B inputs are high, then both the NMOS transistors (bottom half of the diagram) will conduct, neither of the PMOS transistors (top half) will conduct, and a conductive path will be established between the output and V_{ss} (ground), bringing the output low. If both of the A and B inputs are low, then neither of the NMOS transistors will conduct, while both of the PMOS transistors will conduct, establishing a conductive path between the output and V_{dd} (voltage source), bringing the output high. If either of the A or B inputs is low, one of the NMOS transistors will not conduct, one of the PMOS transistors will, and a conductive path will be established between the output and V_{dd} (voltage source), bringing the output high. As the only configuration of the two inputs that results in a low output is when both are high, this circuit implements a NAND (NOT AND) logic gate.

An advantage of CMOS over NMOS logic is that both low-to-high and high-to-low output transitions are fast since the (PMOS) pull-up transistors have low resistance when switched on, unlike the load resistors in NMOS logic. In addition, the output signal swings the full voltage between the low and high rails. This strong, more nearly symmetric response also makes CMOS more resistant to noise.

See Logical effort for a method of calculating delay in a CMOS circuit.

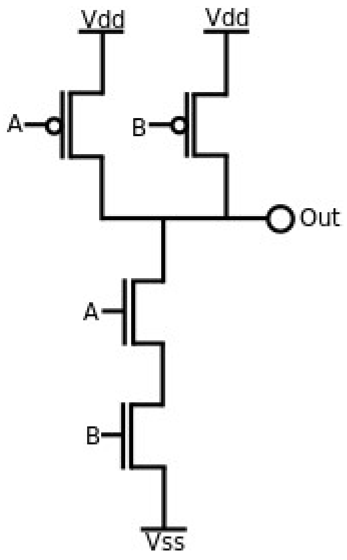
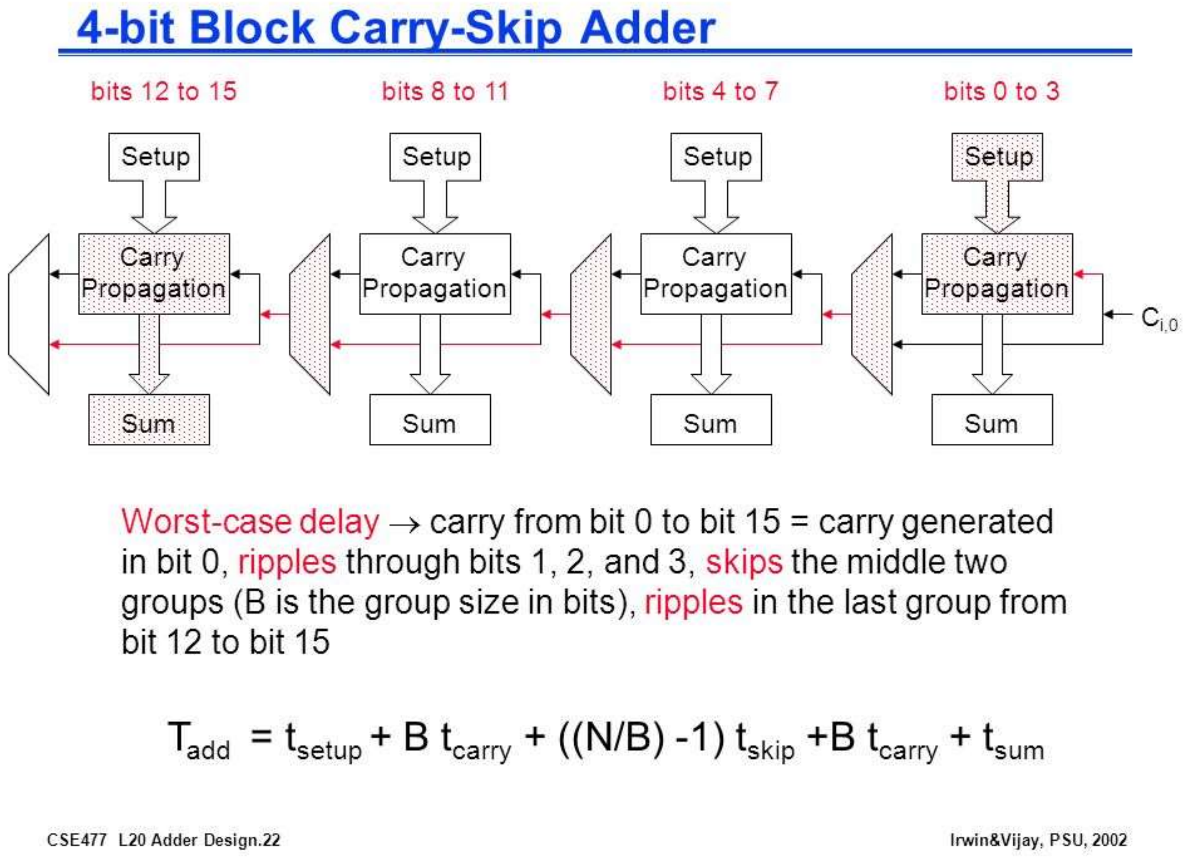


Fig. 3.7 NAND gate in CMOS logic

3.6) Block diagram of 4 bit carry skip adder

The digital circuit design is a most commonly used logic configuration but it has its own merits and demerits . Firstly the large numbers of transistors are required even in the implementation of small circuits. The bellow diagram shown that fig2 1bit full adder CMOS architecture its clear from the diagram that 28 transistors are required to implement the 1bit full adder and along with the AND logic gate also the OR logic gate to complete implementation of 4bit carry skip adder. Six transistors for AND gate total three AND gates, and for the OR gate six transistors are required. It can also understood from the circuit that large number of interconnects are used in this approach to connect the transistors. And in conventional process power dissipation delay and the transistor count are reduced and disadvantage is large number of interconnection are required and the cost is high where compare to pass transistor logic.


bits 12 to 15
bits 8 to 11
bits 4 to 7
bits 0 to 3
Setup
Carry Propagation
Sum
$C_{i,0}$

4-bit Block Carry-Skip Adder

Worst-case delay → carry from bit 0 to bit 15 = carry generated in bit 0, ripples through bits 1, 2, and 3, skips the middle two groups (B is the group size in bits), ripples in the last group from bit 12 to bit 15

$$T_{add} = t_{setup} + B\, t_{carry} + ((N/B) - 1)\, t_{skip} + B\, t_{carry} + t_{sum}$$

CSE477 L20 Adder Design.22

Irwin&Vijay, PSU, 2002

Fig. 3.8 Block Diagram of 4 Bit Carry Skip adder

3.7) Circuit diagram of 4 bit carry skip adder

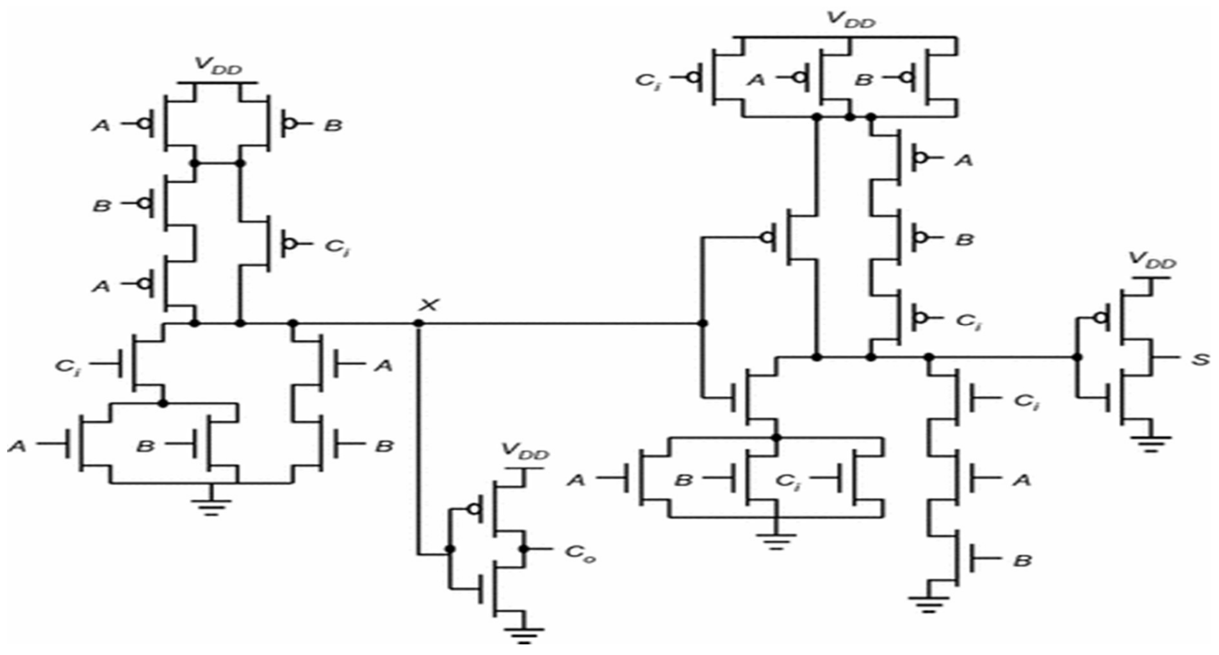


Fig.3.9 1 bit full adder

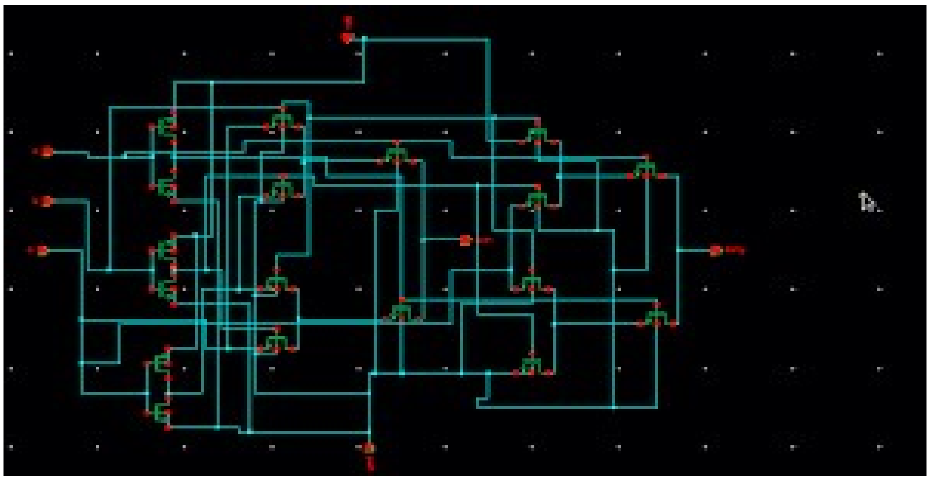


Fig 3.10 Conventional full adder 28transistors

3.8) Software used

H-spice is a device level circuit simulator. H-spice takes a spice file as input and produces output describing the requested simulation of the circuit. It can also produce output files to be used by the AWAVES post processor.

For beginners, chapters under tutorials provide a step by step guide to using H-spice. For a quick reference of mastering H-Spice simulations of EE476, a quick reference is a quick solution. Advance users may go on to the users manuals for specific commands and advance features proviced by H-Spice. If you have used PSpice before, you might want to go to chapters on transient analysis, DC analysis, and AC analysis.

CHAPTER 4
SIMULATION ENVIRONMENT

4.1) SIMULATION SETUP SOFTWARE

H-spice is a device level circuit simulator. H-spice takes a spice file as input and produces output describing the requested simulation of the circuit. It can also produce output files to be used by the AWAVES post processor.

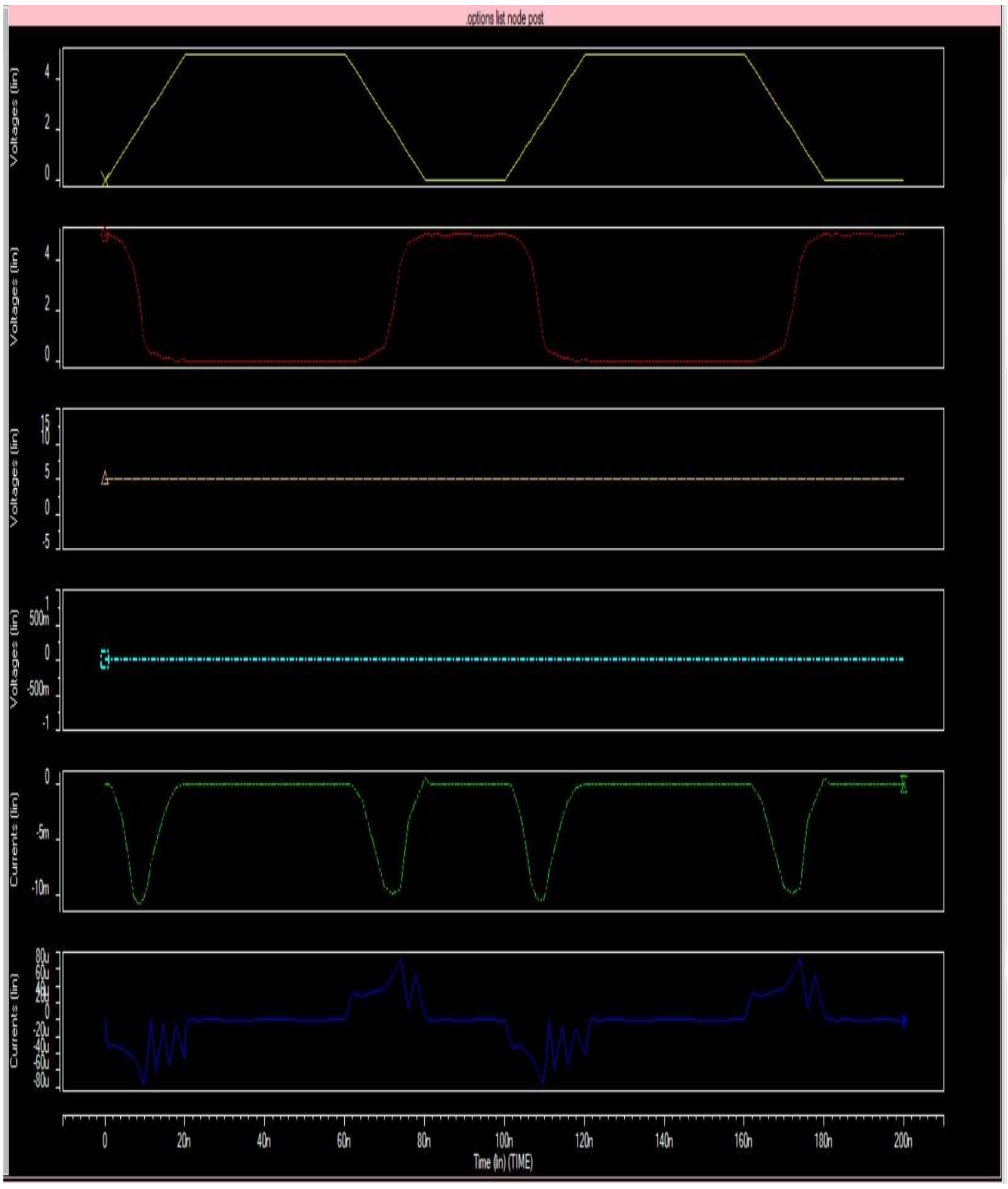
For beginners, chapters under tutorials provide a step by step guide to using H-spice. For a quick reference of mastering H-Spice simulations of EE476, a quick reference is a quick solution. Advance users may go on to the user's manuals for specific commands and advance features provided by H-Spice. If you have used PSpice before, you might want to go to chapters on transient analysis, DC analysis, and AC analysis.

4.2) NOR GATE USING HSPICE SOFTWARE

CODE FOR NOR GATE

```
.options list node post
.Trans 200p 200n
M1  OUT  IN  VDD  VDD  PCH  W=200u  L=1u  M2
OUT IN GND GND NCH W=200u L=1u
Cload  OUT  GND  0.75p  VDD
VDD GND 5
VIN IN GND PULSE 0 5 200p 20n 20n 40n 100n
.PRINT V(OUT) V(IN)
.model  PCH PMOS LEVEL=2
.model  NCH NMOS LEVEL=2
.END
```

4.3) SIMULATION RESULT


.options list node post
Voltages (lin)
Currents (lin)
Time (lin) (TIME)
20n
40n
60n
80n
100n
120n
140n
160n
180n
200n