Study Report: FPGA implementation of low power and high-speed multiplier using Ripple Carry Adder

By

Student Name: **EeshwarVannemreddy**

Enroll no: 18STUHH010**043**

Under Guidance of

Dr. P Akhendra kumar

A Project report submitted for fulfillment of academic requirement for the Special Project-1 EC491

ICFAI Foundation for Higher Education
(Deemed-to-be-University under Section 3 of the UGC Act, 1956)

IcfaiTech
Faculty of Science & Technology

The ICFAI Foundation for Higher Education
Faculty of Science & Technology
(a Deemed University under Section 3 of UGC Act. 1956)
Donthanapally, Shankarapalli Road, Hyderabad – 501203
July – August, 2020

TABLE OF CONTENTS

ABSTRACT:

Need for Digital Signal Processing (DSP) systems which are embedded and portable has been increasing as a result of the speed growth of semiconductor technology. The multiplier is the most crucial part in almost every DSP application. So, the low power, high-speed multipliers is needed for high-speed DSP. Array multiplier is one of the fast multipliers because it has a regular structure and it can be designed very easily. Array multiplier is used for multiplication of unsigned numbers by using full adders and half adders. It depends on the previous computations of the partial sum to produce the final output. Hence, there it is delayed to produce the output. The modified hybrid full adder is a low power full adder, which has a good characteristic in terms of speed and power. Performance criteria for the logic styles are circuit speed, area and power dissipation for the major goal that requires ultimate attention is power consumption. Hence adders are designed in such a way that reduces the propagation delay which is also a cause for low power consumption. Here the new design adopts to produce low power and high-speed ripple carry adder achieved by Gate Diffusion Input (GDI) structure and Hybrid CMOS logic style. The hybrid logic style is employed in order to carry through a wide range of applications. The Gate-Diffusion-Input Multiplexer full adder (GDI-MUX) design withdraw the need for XOR and XNOR gates for designing the full adder cell. The performance parameters being area, delay, power and Power-Delay-Product (PDP) is analyzed using Software Process Improvement and Capability Determination (SPICE).

INTRODUCTION

Demand for Digital Signal Processing systems which are embedded and portable has been increasing as a result of the speed growth of semiconductor technology. Multiplication is a major operation in almost every Digital Signal Processing (DSP) applications. In processor, multiplier speed defines digital signal processor speed. Complex digital signal processing and microprocessor system performance can be enhanced by designing an effective multiplier. Multiplier implementation is a challenging task due to its requirements of the performance. With respect to area, power, delay and throughput, multipliers must be efficient for a wide bit width range. In ICs, for executing the array multiplier we need only small space. In digital integrated circuits, it is an effective method of multiplication, Shifted partial products accumulation and partial products evaluation are principles used in multiplication. Successive operation of addition is required for performing this operation.

Adder is also an important component needed for designing multiplier. It may be a Ripple Carry Adder (RCA), Carry Look Ahead (CLA), Carry Select, Carry Skip and Carry Save Adder (CSA). Various fast adder performances are analyzed by several research works. The addition is the most common operation in arithmetic applications. Subtraction, multiplication, division are some of the basic operations based on addition. Full adder is a simple circuit with three inputs and two outputs in its structure. To implement a full adder in switch level, different logic styles have been used and one among them will be static CMOS logic styles. It is classified into Classical design and hybrid CMOS logic style. As classical design uses only one design style the hybrid style utilizes two or more styles in it.

One example of the classical design approach will be the Complementary CMOS (C-CMOS) full adder. This is from the idea of regular CMOS structure with PMOS pull-up and NMOS pull-down transistors. The layout of this type of full adder is simple and efficient because of the complementary transistor pairs however due to employing number of large PMOS transistors in its structure, the input capacitance is large and also the existence of sized up

PMOS transistors has a direct impact on its area. The full swing is achieved when this structure is utilized in a more complex structure.

The Complementary Pass Transistor logic (CPL) full adder will be another classical circuit. It has a dual-rail structure with 32 transistors in its structure. It provides high speed, full swing output and also good driving capability because of the output static inverters and the fast differential stage of crosscoupled PMOS transistors. The main drawback of CPL is large power consumption due to the existence of a number of internal nodes and static inverters which are the primary source of the static power dissipation.

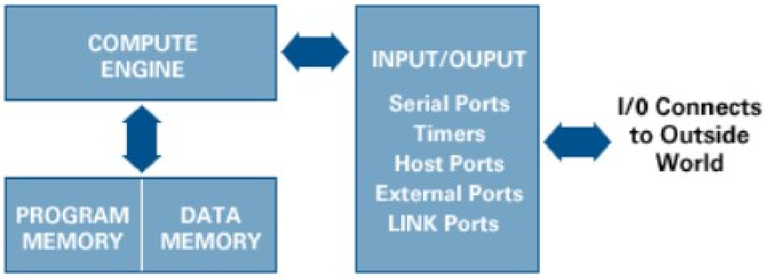
The key difference between the pass-transistor logic and the complementary CMOS logic styles is that the source side of the pass logic transistor network is connected to some input signals rather than to the supply. By this, the advantage is that one pass transistor network (PMOS or NMOS) is enough to perform the logic function.

DIGITAL SIGNAL PROCESSORS

Digital Signal Processors (DSP) take real-world signals in particular voice, audio, video, temperature, pressure, or position that have been digitized and then achieve the mathematically manipulated result. A DSP is designed to perform mathematical functions like "add", "subtract", "multiply" and "divide" very quickly.

Signals need to be processed so that the information that they contain can be displayed, analyzed, or converted to another type of signal that may be of use. In the real-world, analog products detect signals such as sound, light, temperature or pressure and manipulate them. Converters such as an Analog-to-Digital converter then take the real-world signal and turn it into the digital format of 1's and 0's. From here, the DSP takes over by capturing the digitized information and processing it. It then feeds the digitized information back for use in the real world. It does this in one of two ways, either digitally or in an analog format by going through a Digital-to-Analog converter. All of this occurs at very high speeds.

A DSP's information can be used by a computer to control such things as security, telephone, home theater systems, and video compression. Signals may be compressed so that they can be transmitted quickly and more efficiently from one place to another (e.g. teleconferencing can transmit speech and video via telephone lines). Signals may also be enhanced or manipulated to improve their quality or provide information that is not sensed by humans (e.g. echo cancellation for cell phones or computer-enhanced medical images). Although real-world signals can be processed in their analog form, processing signals digitally provides the advantages of high speed and accuracy.



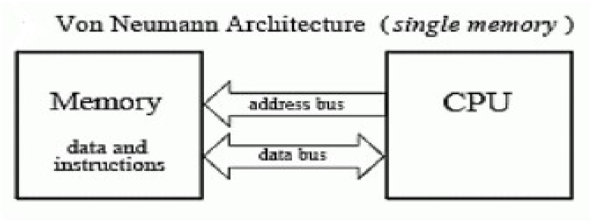
ARCHITECTURE OF DIGITAL SIGNAL PROCESSOR

One of the biggest obstruction in executing DSP algorithms is transferring information to and from memory. This includes data, such as samples from the input signal and the filter coefficients, as well as program instructions, the binary codes that go into the program sequencer. For example, suppose we need to multiply two numbers that reside somewhere in memory. To do this, we must fetch three binary values from memory, the numbers to be multiplied, plus the program instruction describing what to do. Because it's programmable, a DSP can be used in a wide variety of applications. You can create your own software or use software provided by ADI and its third parties to design a DSP solution for an application.

1. VON NEUMAN ARCHITECTURE

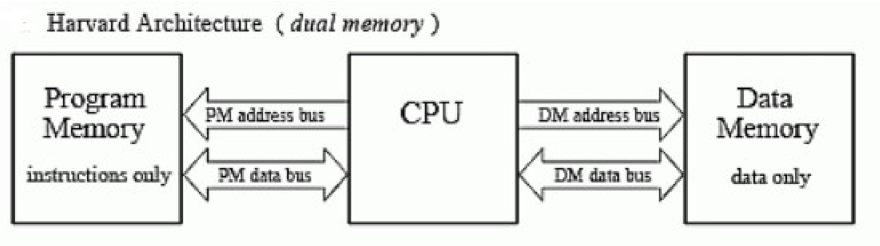
Von Neumann architecture contains a single memory and a single bus for transferring data into and out of the central processing unit (CPU).

Multiplying two numbers requires at least three clock cycles, one to transfer each of the three numbers over the bus from the memory to the CPU. We don't count the time to transfer the result back to memory, because we assume that it remains in the CPU for additional manipulation (such as the sum of products in an FIR filter). The Von Neumann design is quite satisfactory when you are content to execute all of the required tasks in serial. In fact, most computers today are of the Von Neumann design. We only need other architectures when very fast processing is required, and we are willing to pay the price of increased complexity.



2. HARVARD ARCHITECTURE

The Harvard architecture has two separate memory spaces dedicated to program code and to data, respectively, two corresponding address buses, and two data buses for accessing two memory spaces. The Harvard processor offers fetching and executions in parallel.



What are mulpliers?

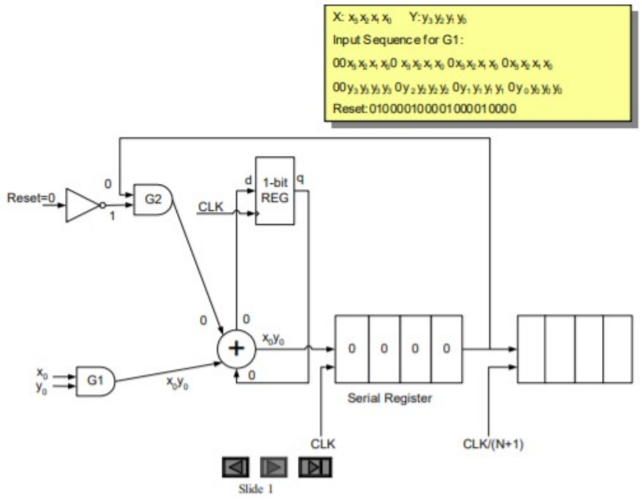
Multipliers play an important role in today’s digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets – high speed, low power consumption, the regularity of layout and hence less area or even a combination of them in one multiplier thus making them suitable for various high speed, low power and compact VLSI implementation.

The common multiplication method is “add and shift” algorithm. In parallel multipliers number of partial products to be added is the main parameter that determines the performance of the multiplier. To reduce the number of partial products to be added, the Modified Booth algorithm is one of the most popular algorithms. To achieve speed improvements Wallace Tree algorithm can be used to reduce the number of sequential adding stages. Further by combining both Modified Booth algorithm and Wallace Tree technique we can see the advantage of both algorithms in one multiplier. However, with increasing parallelism, the number of shifts between the partial products and intermediate sums to be added will increase which may result in reduced speed, increase in silicon area due to irregularity of structure and also increased power consumption due to increase in interconnect resulting from complex routing. On the other hand, “serial-parallel” multipliers compromise speed to achieve better performance for the area and power consumption. The selection of a parallel or serial multiplier actually depends on the nature of the application. In this lecture, we introduce the multiplication algorithms and architecture and compare them in terms of speed, area, power and combination of these metrics.

Types of multipliers

1. SERIAL MULTIPLIER

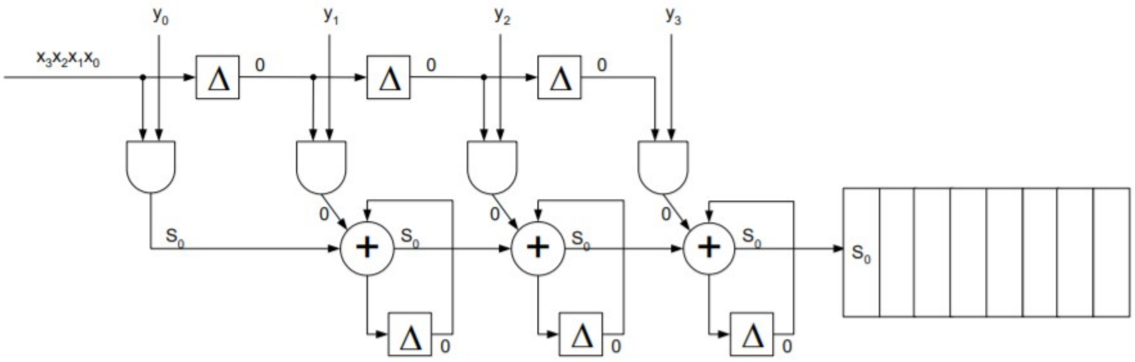
Where area and power are of utmost importance and delay can be tolerated the serial multiplier is used. This circuit uses one adder to add the m * n partial products. The circuit is shown in the fig. below for m=n=4. Multiplicand and Multiplier inputs have to be arranged in a special manner synchronized with circuit behaviour as shown on the figure. The inputs could be presented at different rates depending on the length of the multiplicand and the multiplier. Two clocks are used, one to clock the data and one for the reset. A first-order approximation of the delay is O (m,n). With this circuit arrangement the delay is given as D =[(m+1)n + 1] tfa.


X: $x_3 x_2 x_1 x_0$ Y: $y_3 y_2 y_1 y_0$
Input Sequence for G1:
$00x_3x_2x_1x_00\ x_3x_2x_1x_0\ 0x_3x_2x_1x_0\ 0x_3x_2x_1x_0$
$00y_3y_3y_3y_3\ 0y_2y_2y_2y_2\ 0y_1y_1y_1y_1\ 0y_0y_0y_0y_0$
Reset: 010000100001000010000
Reset=0
0
1
G2
CLK
d
1-bit REG
q
0
0
+
x_0y_0
0
x_0
y_0
G1
x_0y_0
0
0
0
0
Serial Register
CLK
CLK/(N+1)
Slide 1

As shown the individual PP is formed individually. The addition of the PPs are performed as the intermediate values of PPs addition are stored in the DFF, circulated and added together with the newly formed PP. This approach is not suitable for large values of M or N.

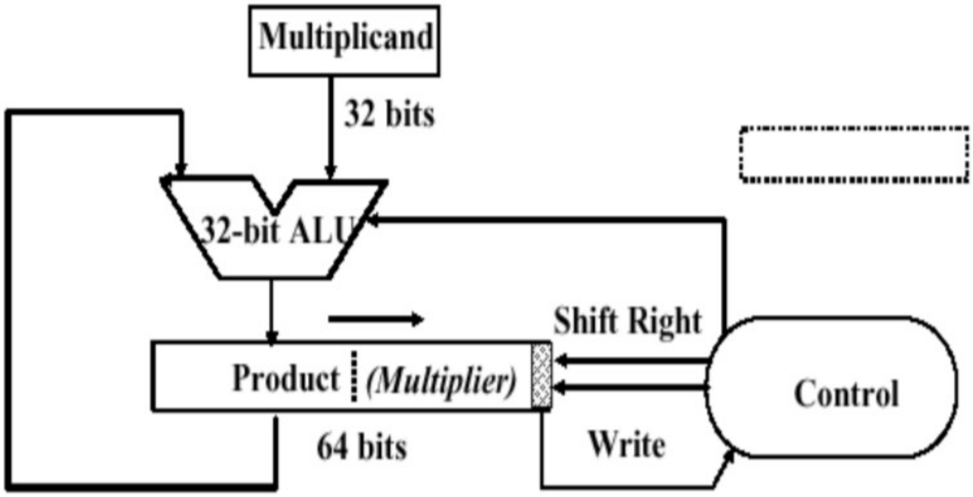
2. SERIAL/PARALLEL MULTIPLIER

The general architecture of the serial/parallel multiplier is shown in the figure below. One operand is fed to the circuit in parallel while the other is serial. N partial products are formed in each cycle. On successive cycles, each cycle does the addition of one column of the multiplication table of M*N PPs. The final results are stored in the output register after N+M cycles. While the area required is N-1 for M=N.



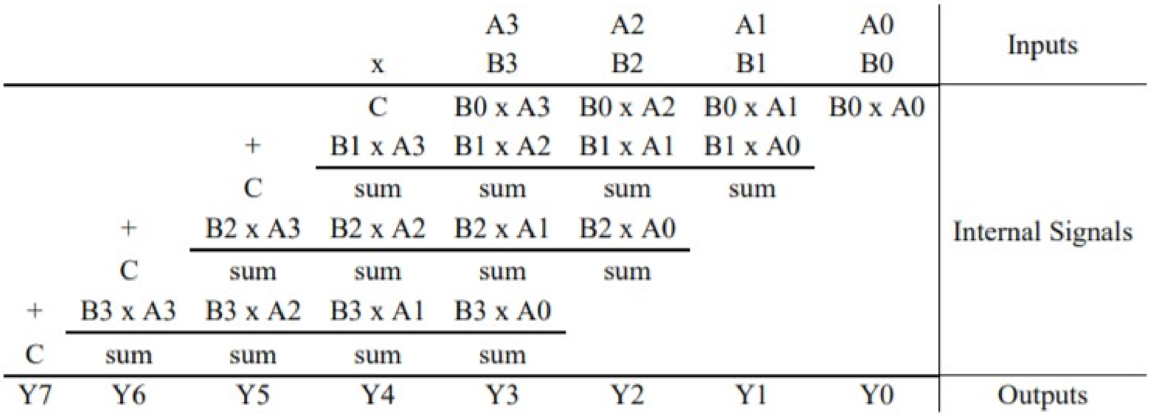
3. SHIFT AND ADD MULTIPLIER

The general architecture of the shift and add multiplier is shown in the figure below for a 32-bit multiplication. Depending on the value of multiplier LSB bit, a value of the multiplicand is added and accumulated. At each clock cycle, the multiplier is shifted one bit to the right and its value is tested. If it is a 0, then only a shift operation is performed. If the value is a 1, then the multiplicand is added to the accumulator and is shifted by one bit to the right. After all the multiplier bits have been tested the product is in the accumulator. The accumulator is 2N (M+N) in size and initially the N, LSBs contains the Multiplier. The delay is N cycles maximum. This circuit has several advantages in asynchronous circuits.



4. ARRAY MULTIPLIER

Array multiplier is well known due to its regular structure. A multiplier circuit is based on add and shift algorithm. Each partial product is generated by the multiplication of the multiplicand with one multiplier bit. The partial product is shifted according to their bit orders and then added. The addition can be performed with normal carry propagate adder. N-1 adders are required where N is the multiplier length.



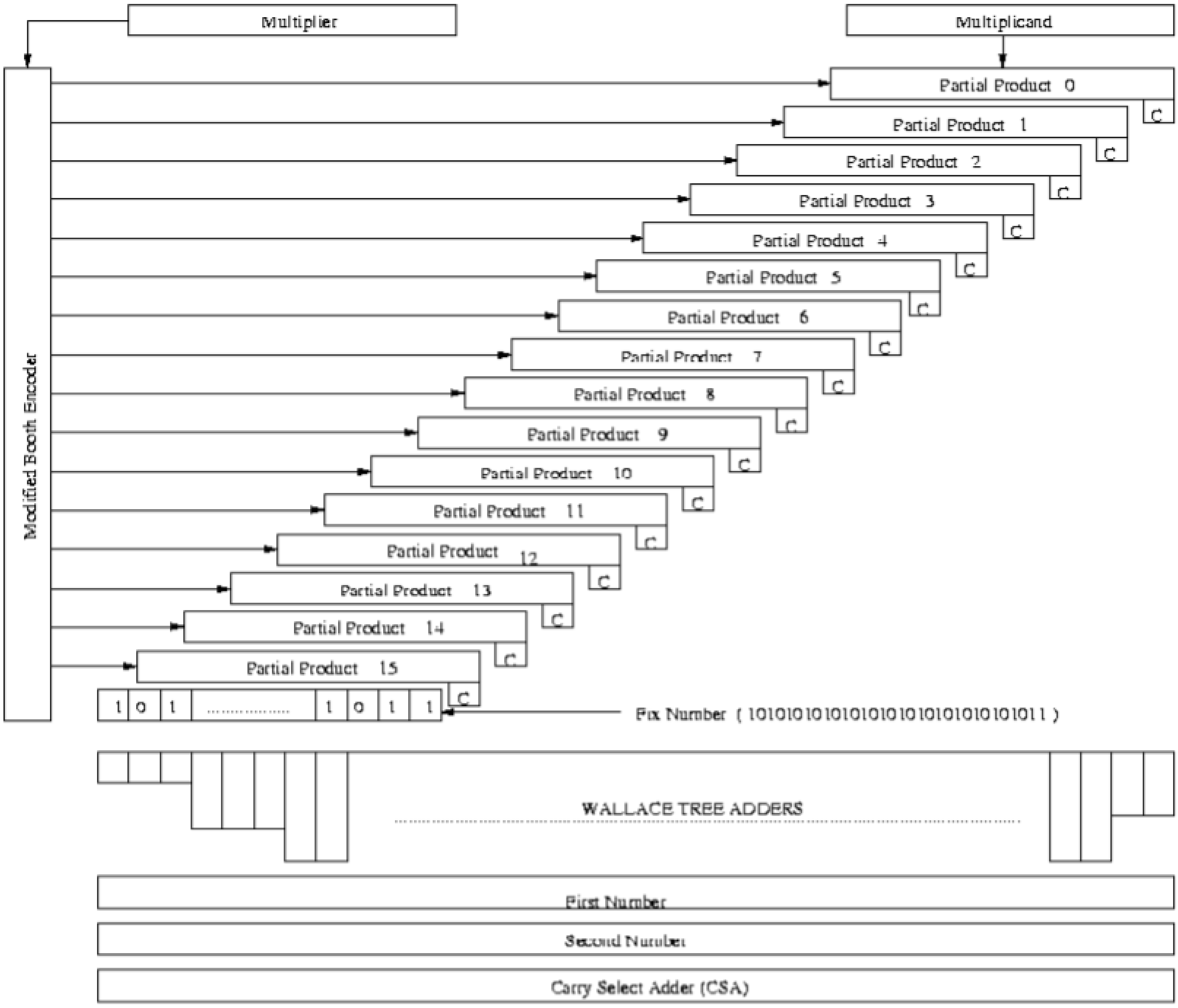
				A3	A2	A1	A0	Inputs
			x	B3	B2	B1	B0	
			C	B0 x A3	B0 x A2	B0 x A1	B0 x A0	Internal Signals
		+	B1 x A3	B1 x A2	B1 x A1	B1 x A0		
		C	sum	sum	sum	sum		
	+	B2 x A3	B2 x A2	B2 x A1	B2 x A0			
	C	sum	sum	sum	sum			
+	B3 x A3	B3 x A2	B3 x A1	B3 x A0				
C	sum	sum	sum	sum				
Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Outputs

5. BOOTH MULTIPLIER

It is a powerful algorithm for signed-number multiplication, which treats both positive and negative numbers uniformly. For the standard add-shift operation, each multiplier bit generates one multiple of the multiplicand to be added to the partial product. If the multiplier is very large, then a large number of multiplicands have to be added. In this case, the delay of the multiplier is determined mainly by the number of additions to be performed. If there is a way to reduce the number of additions, the performance will get better.

6. OPTIMIZED WALLACE TREE MULTIPLIER

In Wallace tree architecture, all the bits of all of the partial products in each column are added together by a set of counters in parallel without propagating any carries. Another set of counters then reduces this new matrix and so on, until a two-row matrix is generated. The most common counter used is the 3:2 counter which is a Full Adder. The final results are added using usually carry propagate adder. The advantage of the Wallace tree is speed because the addition of partial products is now O (logN). A block diagram of 4 bit Wallace Tree multiplier is shown in below. As seen from the block diagram partial products are added in the Wallace tree block. The result of these additions is the final product bits and sum and carry bits which are added in the final fast adder (CRA).



What are adders?

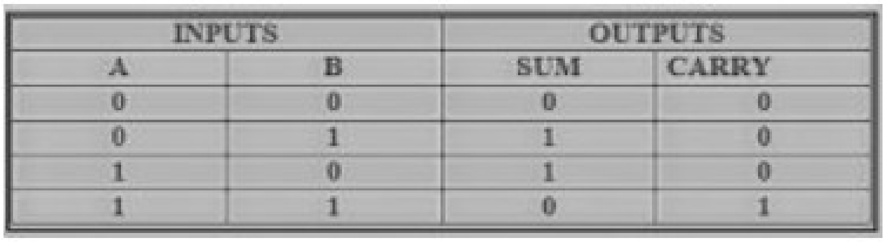
An adder is a digital logic circuit in electronics that is extensively used for the addition of numbers. In many computers and other types of processors, adders are even used to calculate addresses and related activities and calculate table indices in the ALU and even utilized in other parts of the processors. These can be built for many numerical representations like excess-3 or binary coded decimal. Adders are basically classified into two types: Half Adder and Full Adder.

What is Half Adder and Full Adder Circuit?

The half adder circuit has two inputs: A and B, which add two input digits and generates a carry and a sum. The full adder circuit has three inputs: A and C, which add three input numbers and generates a carry and sum. We know that the main and crucial purpose of adders is addition.

1. Half adder

The scenario of half adder, it adds two binary digits where the input bits are termed as augend and addend and the result will be two outputs one is the sum and the other is carry. To perform the sum operation, XOR is applied to both the inputs, and AND gate is applied to both inputs to produce carry.



INPUTS		OUTPUTS	
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Whereas in the full adder circuit, it adds 3 one-bit numbers, where two of the three bits can be referred to as operands and the other is termed as bit carried in. The produced output is 2-bit output and these can be referred to as output carry and sum. By using a half adder, you can design simple addition with the help of logic gates.

- Half Adder IC Number

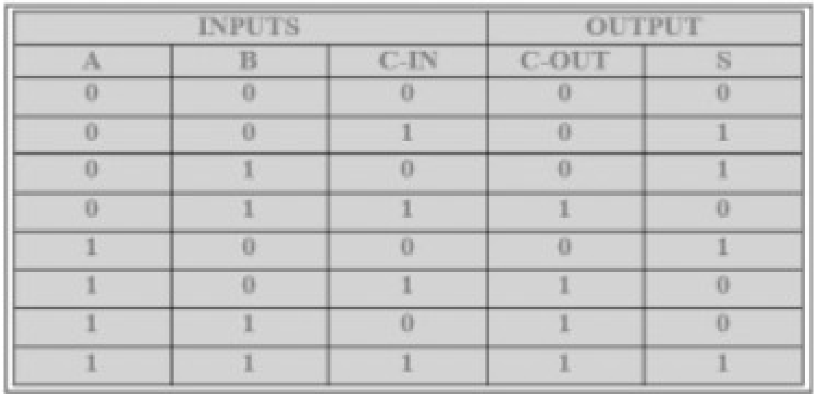
The implementation of half adder can be done through high-speed CMOS digital logic integrated circuits like 74HCxx series which includes the SN74HC08 (7408) & SN74HC86 (7486).

- Half Adder Limitations

The main reason to call these binary adders like Half Adders is, that there is no range to include the carry bit using an earlier bit. So, this is a main limitation of HAs once used like binary adder particularly in real-time situations which involve adding several bits. So this limitation can be overcome by using the full adders.

2. Full Adder

The difference between a half-adder and a full-adder is that the full-adder has three inputs and two outputs, whereas half adder has only two inputs and two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. When a full-adder logic is designed, you string eight of them together to create a byte-wide adder and cascade the carry bit from one adder to the next.



INPUTS			OUTPUT	
A	B	C-IN	C-OUT	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The output carry is designated as C-OUT and the normal output is represented as S which is ‘SUM’.

With the above full adder truth-table, the implementation of a full adder circuit can be understood easily. The SUM ‘S’ is produced in two steps:

- By XOR the provided inputs ‘A’ and ‘B’
- The result of A XOR B is then XOR with the C-IN

This generates SUM and C-OUT is true only when either two of three inputs are HIGH, then the C-OUT will be HIGH. So, we can implement a full adder circuit with the help of two half adder circuits. Initially, the half adder will be used to add A and B to produce a partial Sum and a second-half adder logic can be used to add C-IN 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. So, C-OUT will be an OR function of the half-adder Carry outputs.

The difference between the half adder and full adder is that half adder produces results and full adder uses half adder to produce some other result. Similarly, while the Full-Adder is of two Half-Adders, the Full-Adder is the actual block that we use to create the arithmetic circuits.

Carry Adders

In the concept of ripple carry adder circuits, the bits that are necessary for addition are immediately available. Whereas every adder section needs to hold its time for the arrival of carry from the previous adder block. Because of this, it takes more time to produce SUM and CARRY as each section in the circuit waits for the arrival of input.

For instance, to deliver output for the nth block, it needs to receive input from (n-1)th block. And this delay is correspondingly termed as propagation delay. To overcome the delay in ripple carries adder, a carry-lookahead adder was introduced. Here, by using complicated hardware, the propagation delay can be minimized.

Application of Half and Full Adders

- The binary bits addition can be done by half adder using ALU within the computer because it uses adder.
- Half adder combination can be used for designing a full adder circuit. Half adders are used in the calculators and to measure the addresses as well as tables
- These circuits are used to handle different applications within digital circuits. In the future, it plays a key role in digital electronics.
- A FA circuit is used as an element in many large circuits such as Ripple

Carry Adder. This adder adds the number of bits simultaneously.

- FAs are used in the Arithmetic Logic Unit (ALU). FAs are used in graphics-related applications like GPU (Graphics Processing Unit)
- These are used in the multiplication circuit to execute Carryout Multiplication.
- In a computer, to generate the memory address & to build the program counterpoint toward subsequent instruction, the Arithmetic Logic Unit is used by using Full Adders.

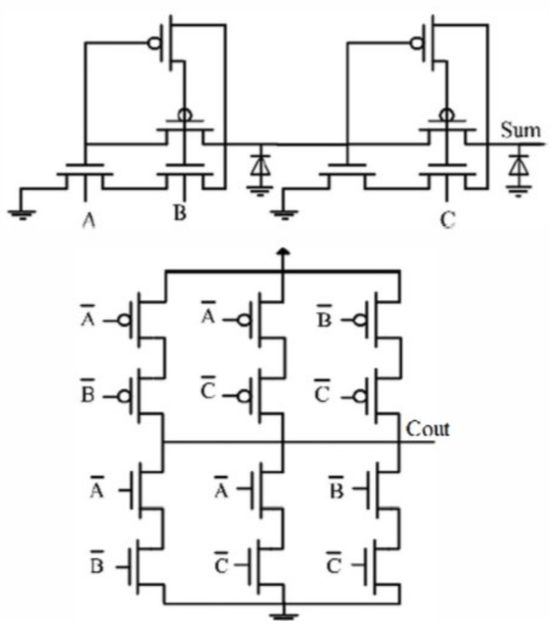
Notable mention: Conventional full adders

There are some standard implementations with various logic styles that have been used in the past to design full adder circuits. The logic style used in logic gates basically influences the circuit speed, area, power consumption, and the wiring complexity of a circuit. A full adder may be cascaded in order to build a multi-bit binary number. Full adders are very useful in constructing arithmetic circuits like ripple carry, carry look ahead, carry-save, carry skip adders. Also, it can be used in higher-order structures like multipliers.

- **Ultra low power full adder**

  This structure utilizes the combination of Pass Transistor Logic and static-CMOS Logic style. This adder utilizes special voltage restorer called ULPD (Ultra Low- Power Diode) which eliminates the speed problem of the conventional voltage restorer in order to create full swing voltage output. The ULPD which is shown in Figure has been created by the combination of an NMOS and a PMOS transistor. It has low leakage current in a reverse-biased situation.

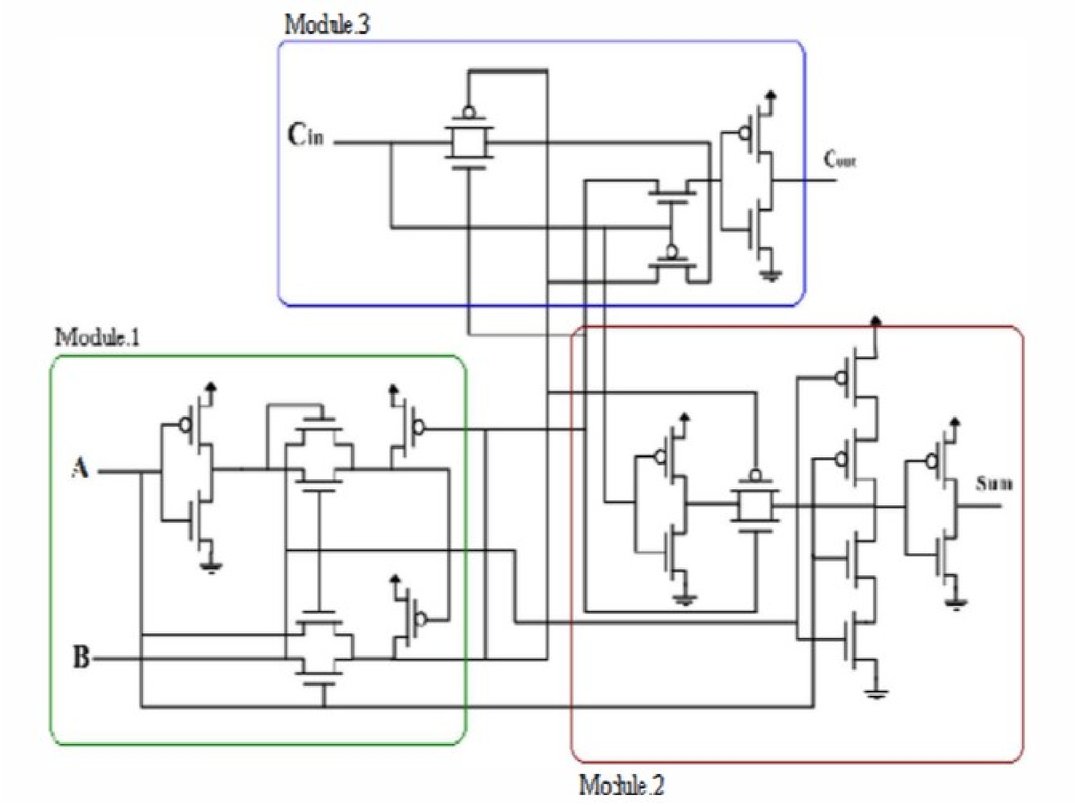
  While increasing the reverse bias voltage, the reverse current of ULPD is increased, but after reaching a maximum value it will decrease due to the gate-source voltage of both transistors becoming more negative.



It results in a negative resistance region that could be used in level restoration. ULPF A uses Low-Power XOR-XNOR gates which are implemented with Pass Transistor Logic style in order to produce the SUM output and by utilizing ULPD as the voltage level restorer. This circuit is robust against voltage scaling and also transistor sizing.

- **Conventional hybrid CMOS full adder**

The hybrid logic style utilizes different logic styles to create new full adders with its desired performance. The structure shown in Figure utilizes a novel XOR-XNOR design to produce intermediate signals. It uses three modules. Module 1 is based on Complementary Pass Transistor Logic and an inverter. Module 2 uses the Low-Power consuming Transmission gates. Module 3 uses four transistors XOR gate and an inverter.

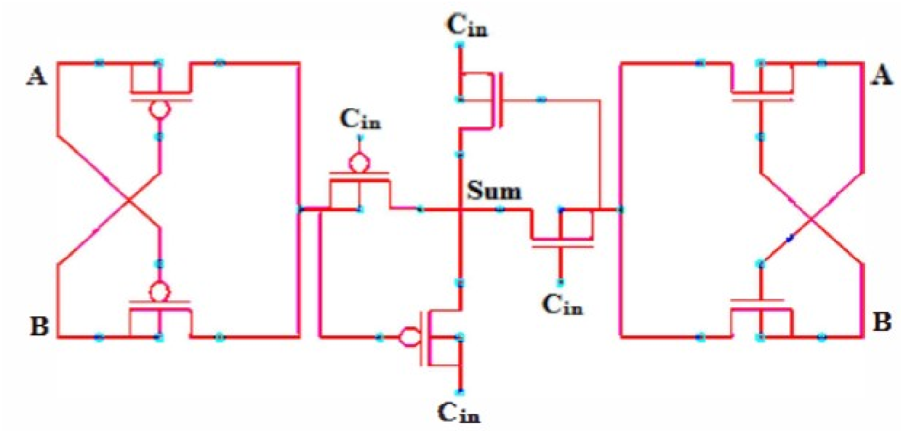


Main idea: Modified hybrid full adder

The modified hybrid full adder is a low power full adder, which has a good characteristic in terms of speed and power. This design is based on Semi XOR-XNOR gates but it has a lack of ability to generate all the possible outputs as in conventional XOR-XNOR gates.

A	B	Semi XOR	Semi XNOR
0	0	0	Z
0	1	1	0
1	0	1	0
1	1	Z	1

From the above full adder Truth Table, the Semi XOR gate has the ability to generate the first four states of the sum output, with the remaining states produced with the Semi XNOR gate. In order to design the SUM circuit, these Semi XOR-XNOR gates could be employed with Cin input shown in the figure below which is used as a selector. When Cin is equal to '0', the Semi XOR the gate is similar to the SUM, and when Cin is equal to '1', the Semi XNOR gate is similar to the remaining states of the SUM.

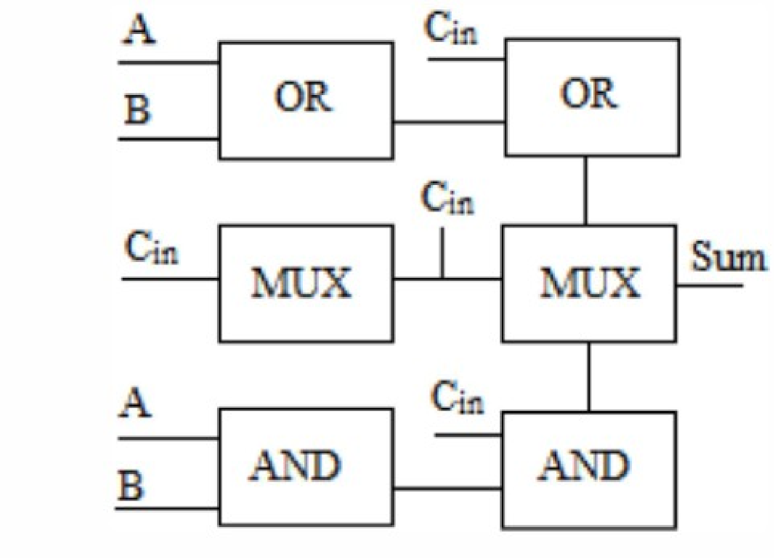


The high impedance states may cause an error in the circuit. One of the situations occurs when 'A' and 'B' inputs are equal to '1' and when Cin input is '0'. To have the correct value at the output node, one more transistor is added to compensate for the inability of the circuit. If the output of the Semi XNOR the gate is connected with the gate of an NMOS transistor and source or drain of this transistor is connected with SUM output and also its drain or source is connected to the Cin, the high impedance state that occurs in the output will be set to the correct logic.

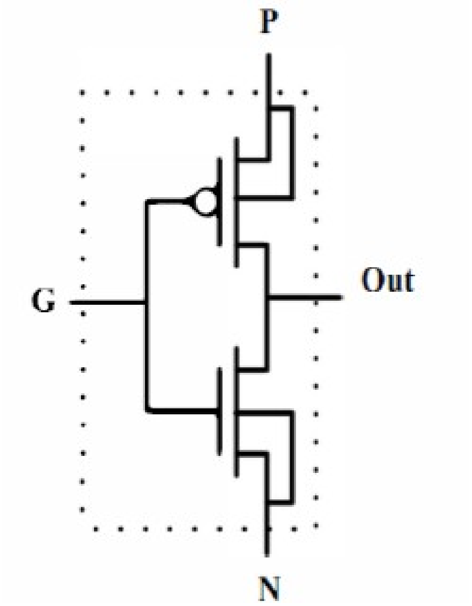
To remove the other high impedance situation, one more transistor needs to be added. This situation occurs when 'A' and 'B' is equal to '0 ' and Cin is '1'. If Semi XOR output is connected to the gate of one PMOS while the source or drain of this transistor is connected to the SUM output and its drain or source is connected to Cin, the other high impedance state of the circuit can be removed. The newly added PMOS transistor is only turned ON in two states when the Semi XOR gate output is equal to '0'. In these two states, the output of the sum generator is equal to Cin, and this will be resulting in a new design for the SUM part which is shown in Fig. 3. The Cout generator is constructed in a similar manner.

Designing GDI-MUX Full Adder

The approach for designing full adder eliminates the need for complicated XOR-XNOR gates. From the full adder truth table, a logic scheme shown in Figure below for sum and Cout may be designed.

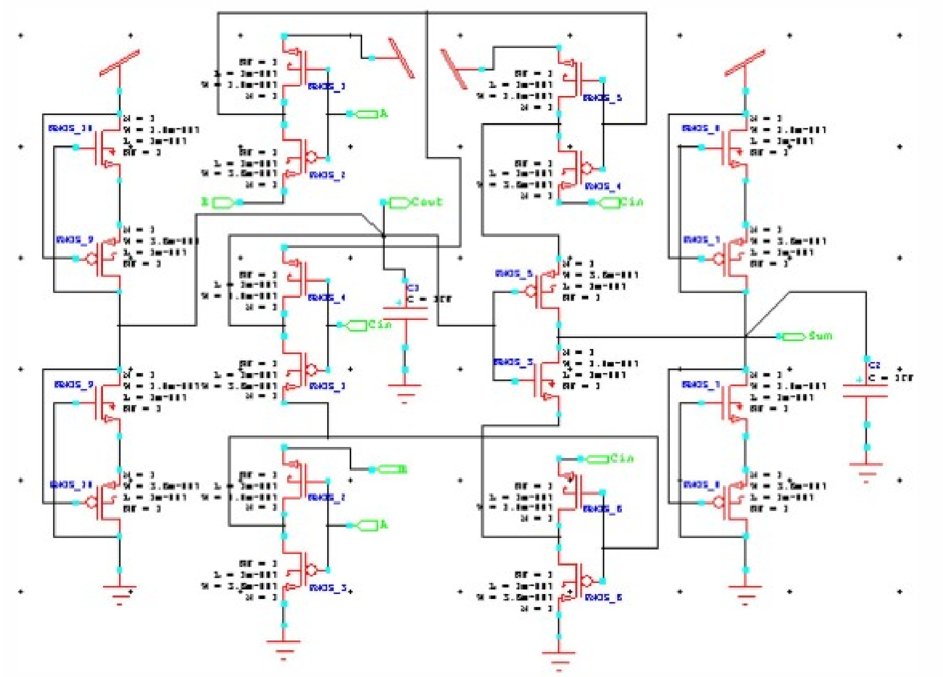


To implement the logic scheme GDI technique is used. The basic Gate-Diffusion Input structure is shown in Figure below. GDI structure is used in designing lower power circuits with the help of a reduced number of transistors.



To implement the logic scheme GDI technique is used. The basic Gate-Diffusion Input structure is shown in Fig. 6. GDI structure is used in designing lower power circuits with the help of reduced number of transistors.

In order to implement (A + B), N input is connected to supply voltage, P is connected to Band G is connected to A. Similarly (A . B) is designed by connecting G, N and P to A, Band GND respectively. To produce Cout. Cin is connected to G input of GDI as selector and N is connected to (A+B) and P is connected to (A.B) which is the implementation of multiplexer. Similarly (A + B + Cin) is implemented by connecting G input to (A + B), P to Cin and N to Vdd. (A . B . Cin) by connecting G input to (A . B), P to GND and N to Cin' Finally in order to produce the SUM, GDI is used as a multiplexer with its G input connected to COU! and finally its P and N inputs are linked to (A + B + Cin) and (A . B. Cin) which is shown below

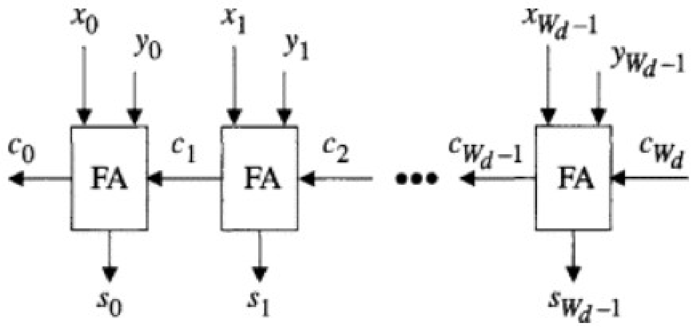


The full adder cells are simulated using SPICE with lOO MHz frequency and at 27 degree Celsius with the supply voltages varying from 0.6 to 1.2 Voltage for 90 nm technology. The threshold voltages of the PMOS and NMOS transistors are 0.27 V for 90 nm. Different loading conditions are also considered to evaluate the performance of the test circuits. Loading conditions are varying from 1 to 70 femto Farad (fF) for 90nm.

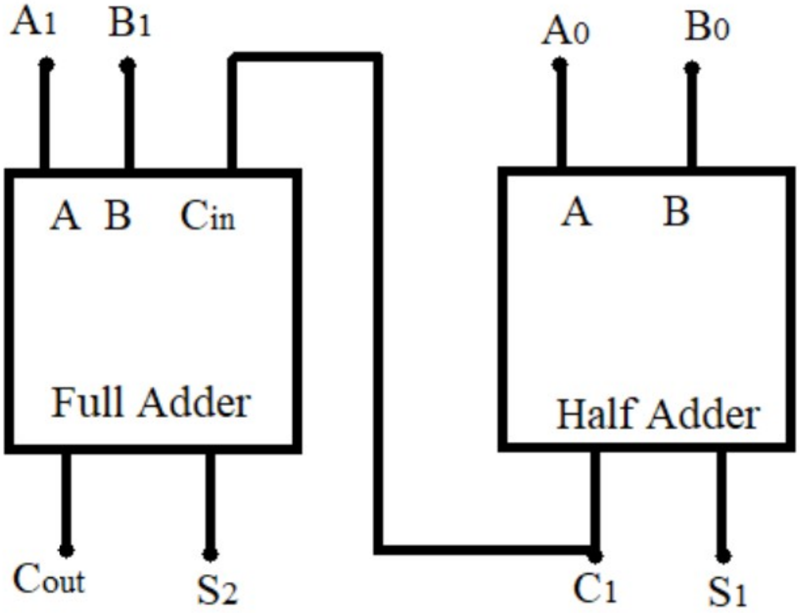
It is significant to note that, the modified hybrid full adder has the smallest delay. When comparing the performance parameters the GDI-MUX full adder has the best performance mainly in terms of power consumption

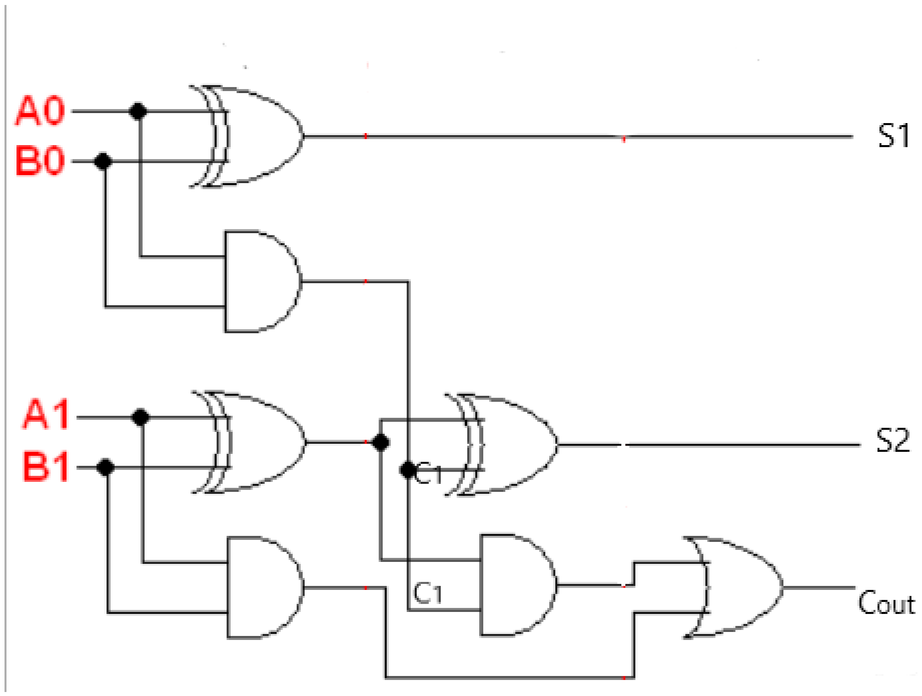
RIPPLE CARRY ADDER:

The ripple-carry adder (RCA) is the simplest form of adder. Two numbers using two's-complement representation can be added by using the circut shown. A *Wd*-bit RCA is built by connecting *Wd* full-adders so that the carry-out from each full-adder is the carry-in to the next stage. The sum and carry bits are generated sequentially, starting from the LSB. The carry-in bit into the rightmost full-adder, corresponding to the LSB, is set to zero, i. e., (*cWd* = 0). The speed of the RCA is determined by the carry propagation time which is of order *O*(*Wd*). Special circuit realization of the full-adders with fast carry generation are often employed to speed the operation.



2-BIT RIPPLE CARRY ADDER





This is a 2-bit ripple carry adder which has 2-bit inputs (A1, A0) and (B1, B0) with one signle input Cin which is the carry from the rightmost , it should have 2-bit output (S1, S0) and one single bit output Cout.

Implementation of Ripple Carry Adder

In a ripple carry adder (RCA) carry out of one adder is the carry-in of succeeding adders. New full adders such as modified hybrid and GDI-MUX are cascaded to realize the logic function of the ripple carry adder. The structure of the ripple carry adder is much simple and it allows for fast design time. The main advantage is that ripple carry adder is faster than the serial adder.

CONCLUSION

The modified hybrid and GDI-MUX full adder designs successfully operate at low voltages. There is a significant improvement in terms of power dissipation, delay and PowerDelay-Product (PDP) parameter. The modified hybrid and GDI-MUX full adder is implemented in two bit ripple carry adder. The performance parameters are then compared with conventional ripple adder. Here modified ripple carry adder achieves better power, delay and power-delay product.

REFERENCES:

https://www.sciencedirect.com/topics/computer-science/ripple-carry-adder

https://ieeexplore.ieee.org/document/6422217

https://www.researchgate.net/figure/Structure-of-reduced-MUX-using-GDI-logic_fig2_320490451

https://ieeexplore.ieee.org/document/4295280