MODULE-2 LOGIC DESIGN WITH MSI COMPONENTS AND PROGRAMMABLE LOGIC DEVICES

Binary Adders and Subtractors, Comparators, Decoders, Encoders, Multiplexers, Programmable Logic Devices (PLD's).

DESIGN OF COMBINATIONAL CIRCUITS

The design of combinational circuits starts from the verbal outline of the problem and ends in a logic circuit diagram. The procedure involves the following steps:

1. State the given problem completely and exactly
2. Interpret the problem, and determine the available input variables and required output variables.
3. Assign a letter symbol to each input and output variables.
4. Design the truth table, which defines the required relations between inputs and outputs.
5. Obtain the simplified Boolean expression for each output using k-maps.
6. Draw the logic circuit diagram to implement the Boolean expression.

ARITHMETIC CIRCUITS

One essential function of most computers and calculators is the performance of arithmetic operations. The logic gates designed so far can be used to perform arithmetic operations such as addition, subtraction, multiplication and division in electronic calculators and digital instruments. Since these circuits are electronic, they are very fast. Typically an addition operation takes less than 1 μs.

HALF – ADDER

A Logic circuit used for the addition of two one bit numbers is referred to as a half-adder. From the verbal explanation of a half adder, we find that this circuit needs two binary inputs and two binary outputs. The input variables designate the augend and addend bits; the output variables produce the sum and carry. We assign the symbols A and B to the two inputs and S (for sum) and C (for carry) to the outputs. The truth table for the half-adder is shown below.

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Here the C output is 1 only when both inputs are 1. The S output represents the least significant bit of the sum. The logic expression for the sum output can be obtained from the truth table. It can be written as a SOP expression by summing up the input combinations for which the sum is equal to 1.

In the truth table, the sum output is 1 for **A′B and AB′.** Therefore, the expression for the sum is

$$\mathbf{S = A'B + AB' = A \oplus B.}$$

Similarly, the logic expression for carry output can be written as a SOP expression by summing up the input combinations for which the carry is equal to 1. In the truth table, the carry is 1 for AB. Therefore **C = AB** This expression for C cannot be simplified. The sum output corresponds to a logic Ex-OR function while the carry output corresponds to an AND function. So the half-adder circuit can be implemented using Ex-OR and AND gate as shown below.

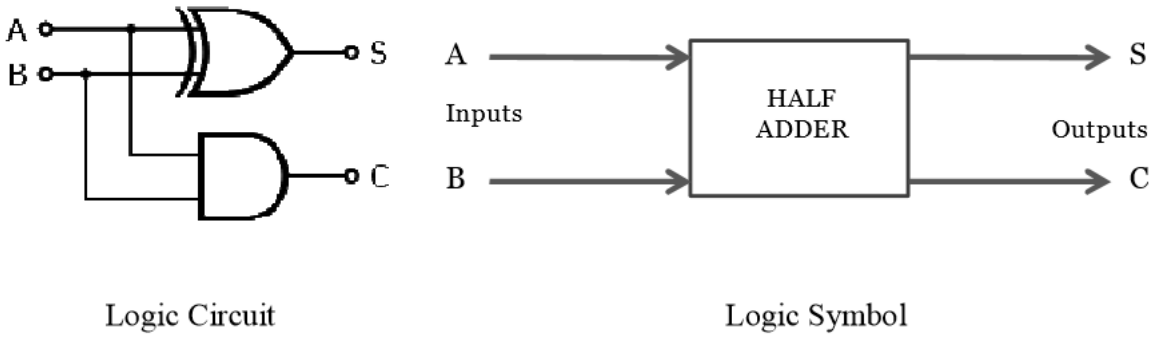


Fig 1: Half Adder Logic circuit

This circuit is called Half-Adder, because it cannot accept a CARRY-IN from previous additions. This is the reason that half – adder circuit can be used for binary addition of lower most bits only. For higher order columns we use a 3-input adder called full-adder

FULL – ADDER

A combinational logic circuit for adding three bits. As seen, a half-adder has only two inputs and there is no provision to add carry coming from the lower bit order when multi bit addition is performed. For this purpose we use a logic circuit that can add three bits, the third bit is the carry

from the lower column. This implies that we need a logic circuit with 3 inputs and 2 outputs. Such a circuit is called a full – adder. The truth table for the full-adder is as shown below.

Inputs			Outputs	
A	B	C_{in}	S	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

As shown there are 8 possible input combinations for the three inputs and for each case the S and Cout values are listed. From the truth table, the logic expression for S can be written by summing up the input combinations for which the sum output is 1 as:

S = A′B′Cin + A′BC′in + AB′C′in + ABCin

= A′(B′Cin + BC′in) + A(B′C′in + BCin)

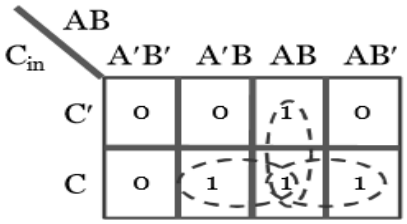
= A′(B⊕Cin)+ A(B⊕Cin)′ Let B⊕Cin = X

Now, S = A′X + AX′ = A ⊕ X Replacing X in the above expression we get

S = A ⊕ B ⊕ Cin

Similarly the logic expression for Cout can be written as

Cout = A′BCin + AB′Cin + ABC′in + ABCin



Cout = BCin + ACin + AB (using the map shown)

From the simplified expressions of S and C the full adder Circuit can be implemented using two 2-input XOR gates, Three 2 –input AND gates and one 3-input OR gate a shown below fig (a). The logic symbol is also shown as fig (b).

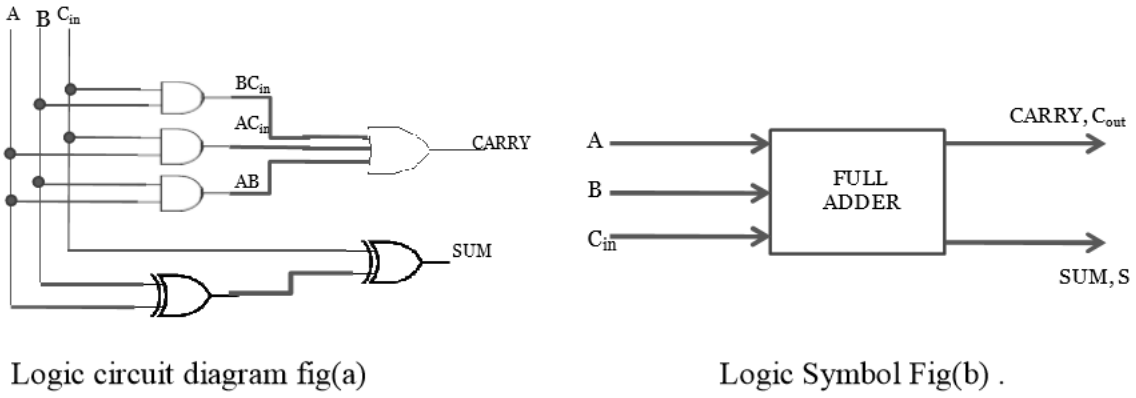


Fig 2: Full Adder Logic Circuit

The logic symbol has two inputs A and B plus a third input Cin called the Carry-in and two outputs SUM and the Carry called Carry out, Cout going to the next higher column.. A full adder can be made by using two half adders and an OR gate as shown below.

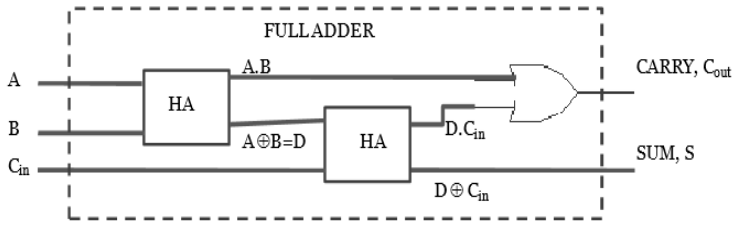
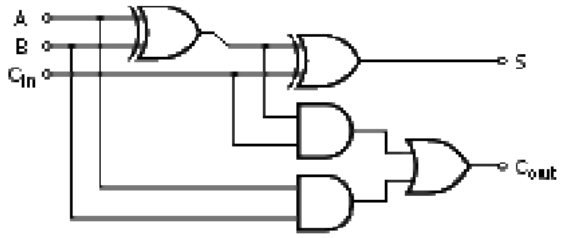


Fig 3: Full adder circuit using two half adders



HALF – SUBTRACTOR

A logic circuit that subtracts Y (subtrahend) from X(minuend), where X and Y are 1-bit numbers, is known as a half-subtractor. It has two inputs X (minuend) and Y (subtrahend) and two outputs D (difference) and B (borrow), as shown in the block diagram.

The operation of this logic circuit is based on the rules of binary subtraction given in the truth table reproduced on the basis of the subtraction process.

Inputs		Outputs	
X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

The difference output in the third column has the same logic pattern as when X is XORed with Y (same as in the case of sum). Hence an Ex-Or gate can be used to give difference of two bits. The borrow output in the 4th column can be obtained by using a NOT gate and AND gate, as shown in the circuit diagram below.

The logical equations for the difference D and borrow B are given as

$$\mathbf{D = X'Y + XY' = X \oplus Y.}$$

$$\mathbf{B = X'Y}$$

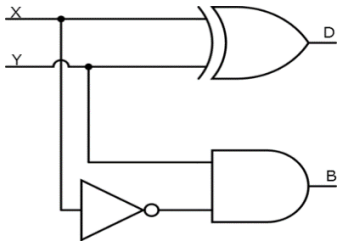
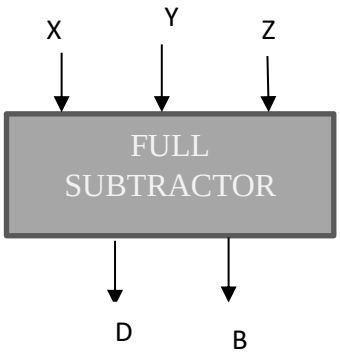


Fig 4: Half Subtractor Logic circuit

FULL – SUBTRACTOR

The full-subtractor is a combinational circuit which is used to perform subtraction of three single bits.



The truth table for the full-subtractor is as shown below.

INPUT			OUTPUT	
X	**Y**	**Z**	**D**	**B**
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

As shown there are 8 possible input combinations for the three inputs and for each case the D and B values are listed. From the truth table, the logic expression for D can be written by summing up the input combinations for which the Difference output is 1 as:

D = X'Y'Z + X'YZ' + XY'Z'+ XYZ

=X' (Y'Z + YZ') + X (Y'Z'+ YZ)

= X'(Y ⊕Z) + X(Y ⊕Z)'

D=X ⊕Y ⊕ Z

B = X'Y'Z + X'YZ' + X'YZ + XYZ

= Z (X'Y + XY') + X'Y (Z' + Z)

B = Z (X⊕ Y) + X'Y

The circuit diagram for full subtractor is constructed from half subtractor and the extension to it , as shown below.

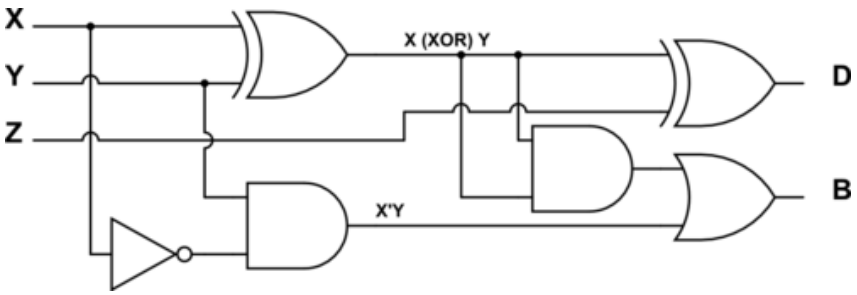


Fig 5: Full Subtractor Logic circuit

4-BIT PARALLEL ADDER

The 4-bit binary adder performs the **addition of two 4-bit numbers**. Let the 4-bit binary numbers, A=A3 A2 A1 A0 and B=B3 B2 B1 B0. We can implement 4-bit binary adder in one of the two following ways.

- Use one Half adder for doing the addition of two Least significant bits and three Full adders for doing the addition of three higher significant bits.
- Use four Full adders for uniformity. Since, initial carry C_0 is zero, the Full adder which is used for adding the least significant bits becomes Half adder.

For the time being, we considered second approach. The **block diagram** of 4-bit binary adder is shown in the following figure.

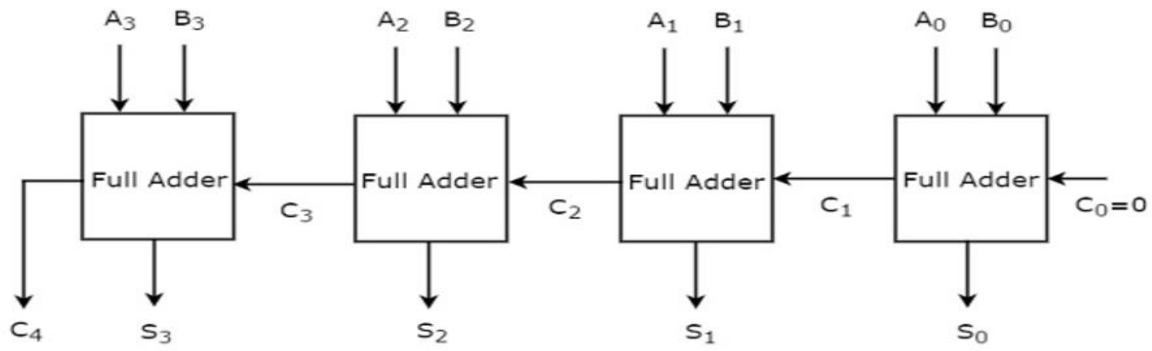


Fig 6: 4-Bit Binary Adder Circuit

Here, the 4 Full adders are cascaded. Each Full adder is getting the respective bits of two parallel inputs A & B. The carry output of one Full adder will be the carry input of subsequent higher order Full adder. This 4-bit binary adder produces the resultant sum having at most 5 bits. So, carry out of last stage Full adder will be the MSB.

In this way, we can implement any higher order binary adder just by cascading the required number of Full adders. This binary adder is also called as **ripple carry (binary) adder** because the carry propagates (ripples) from one stage to the next stage.

BINARY SUBTRACTOR

The circuit, which performs the subtraction of two binary numbers is known as **Binary subtractor**. We can implement Binary subtractor in following two methods.

- Cascade Full subtractors
- 2's complement method

In first method, we will get an n-bit binary subtractor by cascading 'n' Full subtractors. So, first you can implement Half subtractor and Full subtractor, similar to Half adder & Full adder. Then, you can implement an n-bit binary subtractor, by cascading 'n' Full subtractors. So, we will be having two separate circuits for binary addition and subtraction of two binary numbers.

In second method, we can use same binary adder for subtracting two binary numbers just by doing some modifications in the second input. So, internally binary addition operation takes place but, the output is resultant subtraction.

We know that the subtraction of two binary numbers A & B can be written as,

$A-B = A+(2's \text{ complement of } B)$

$A-B = A+(2's \text{ complement of } B)$

$\Rightarrow A-B = A + (1's \text{ complement of } B) + 1$

4-bit Binary Subtractor

The 4-bit binary subtractor produces the **subtraction of two 4-bit numbers**. Let the 4bit binary numbers, A = A3 A2 A1 A0 and B = B3 B2 B1 B0 . Internally, the operation of 4-bit Binary subtractor is similar to that of 4-bit Binary adder. If the normal bits of binary number A, complemented bits of binary number B and initial carry (borrow), C_{in} as one are applied to 4-bit Binary adder, then it becomes 4-bit Binary subtractor. The **block diagram** of 4-bit binary subtractor is shown in the following figure.

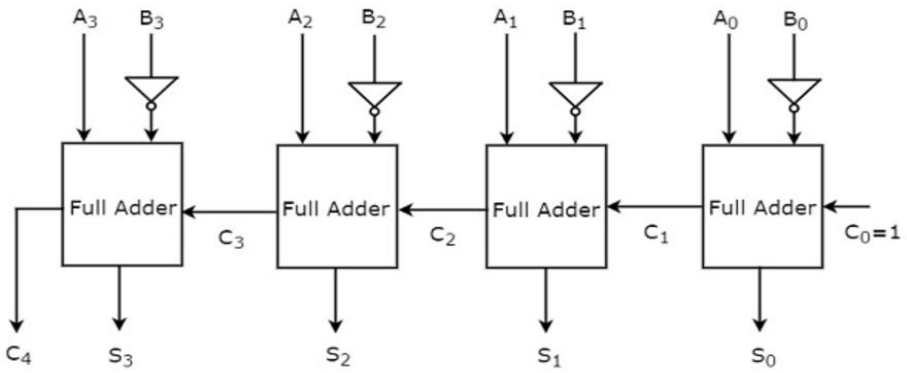


Fig 7: 4-Bit Binary Subtractor Circuit

This 4-bit binary subtractor produces an output, which is having at most 5 bits. If Binary number A is greater than Binary number B, then MSB of the output is zero and the remaining bits hold the magnitude of A-B. If Binary number A is less than Binary number B, then MSB of the output is one. So, take the 2's complement of output in order to get the magnitude of A-B.

In this way, we can implement any higher order binary subtractor just by cascading the required number of Full adders with necessary modifications.

4-bit Binary Adder / Subtractor

The 4-bit binary adder / subtractor produces either the addition or the subtraction of two 4-bit numbers based on the value of initial carry or borrow, C_0. Let the 4-bit binary numbers, A = A3 A2 A1 A0 and B = B3 B2 B1 B0. The operation of 4-bit Binary adder / subtractor is similar to that of 4-bit Binary adder and 4-bit Binary subtractor.

Apply the normal bits of binary numbers A and B & initial carry or borrow, C_0 from externally to a 4-bit binary adder. The **block diagram** of 4-bit binary adder / subtractor is shown in the following figure.

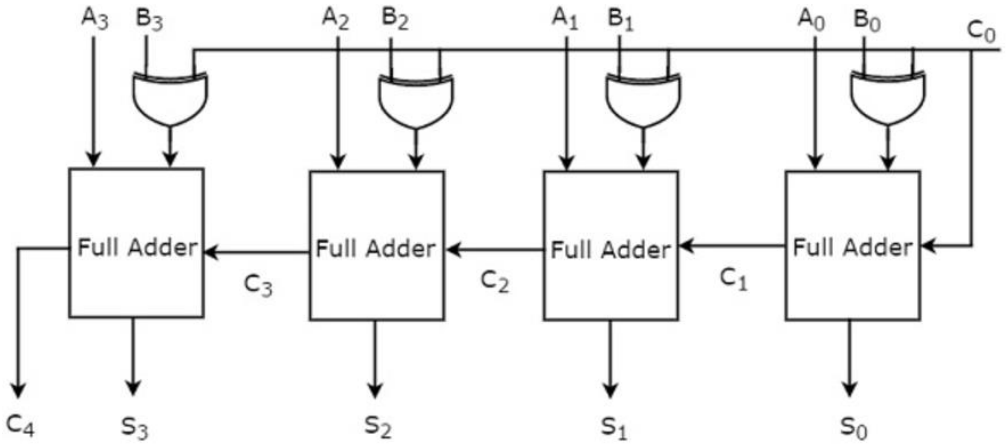


Fig 8: 4-Bit Binary Adder /Subtractor Circuit

If initial carry, C_0 is zero, then each full adder gets the normal bits of binary numbers A & B. So, the 4-bit binary adder / subtractor produces an output, which is the **addition of two binary numbers** A & B.

If initial borrow, C_0 is one, then each full adder gets the normal bits of binary number A & complemented bits of binary number B. So, the 4-bit binary adder / subtractor produces an output, which is the **subtraction of two binary numbers** A & B.

Therefore, with the help of additional Ex-OR gates, the same circuit can be used for both addition and subtraction of two binary numbers.

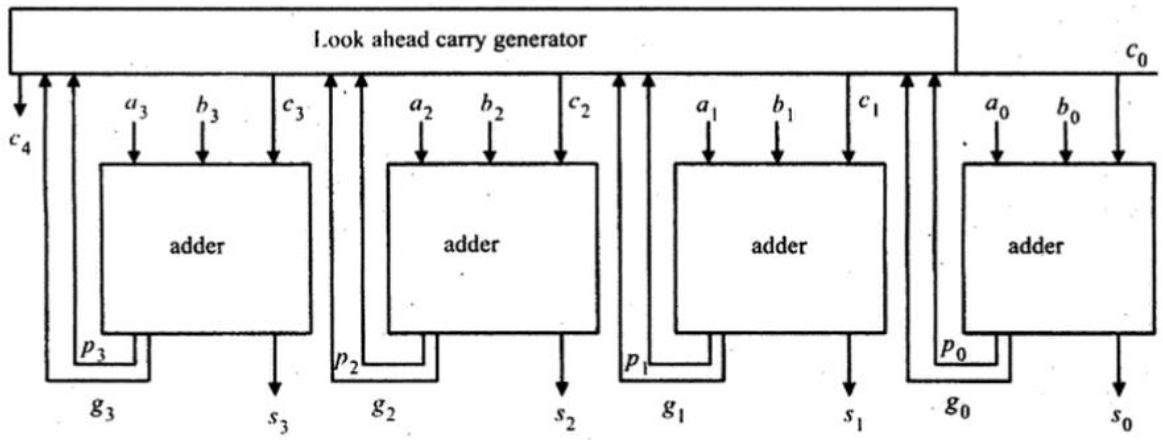


Fig 9: 4-Bit Carry Look Ahead Adder Block Diagram

Pi = Ai ⊕ Bi Carry propagate

Gi = AiBB i Carry generate

For i=0

$c_1 = g_0 + p_0c_0$

For i=1

$c_2 = g_1 + p_1c_1$

$= g_1 + p_1(g_0 + p_0c_0)$

$= g_1 + p_1g_0 + p_1p_0c_0$

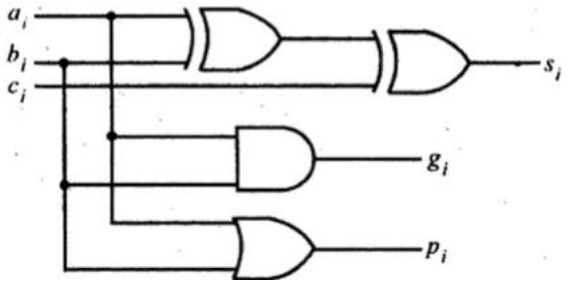
For i=2

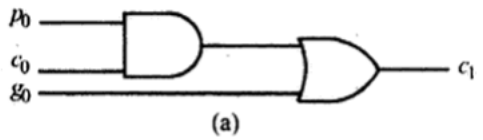
$c_3 = g_2 + p_2c_2$

$= g_2 + p_2g_1 + p_2p_1g_0 + p_2p_1p_0c_0$

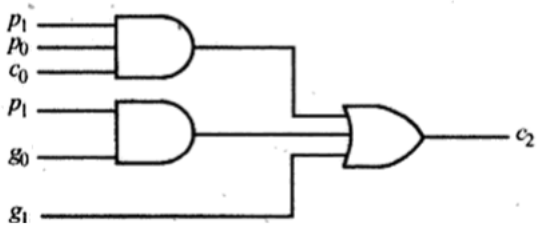
For i=3

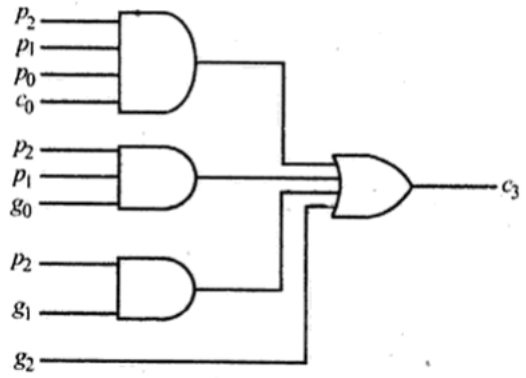
$c_4 = g_3 + p_3c_3 = g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1g_0 + p_3p_2p_1p_0c_0$





(a)





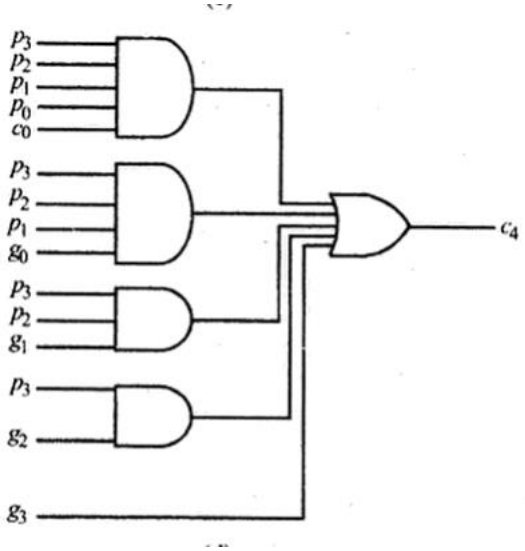


Fig 10: 4-Bit Carry Look Ahead Adder Logic circuit

Complex Programmable Logic Devices (CPLD's)

- As number of Boolean expression increases, designing a digital circuit using PLD's becomes difficult.
- To overcome this problem we can use complex programmable logic devices.
- Using CPLDs we can implement more than 20 Boolean expression in a digital circuit.

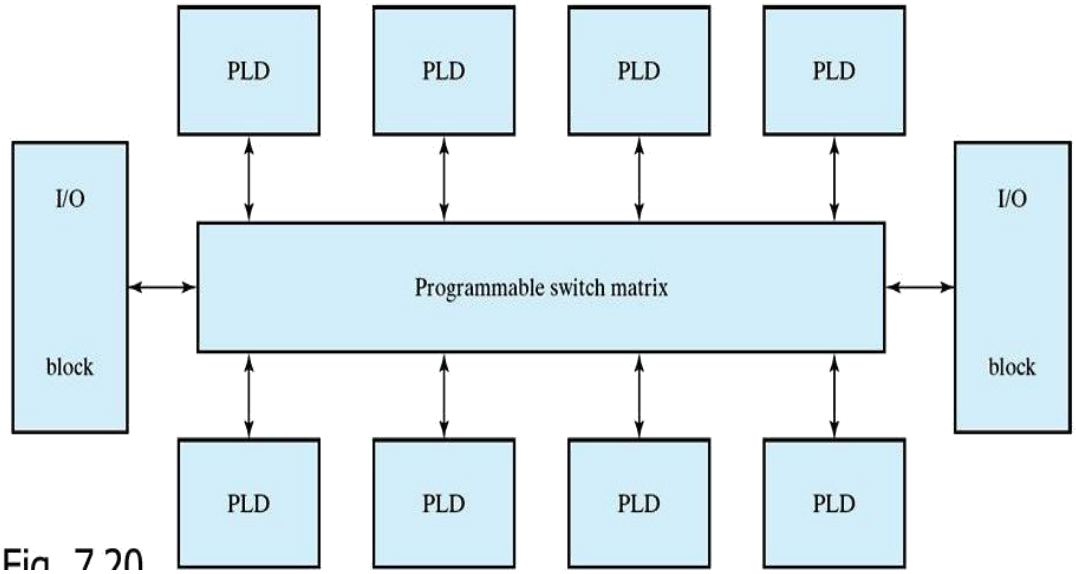


Fig 11: Block Diagram of CPLD

Features of CPLD

- High Performance
- Fast connection of Switch matrix
- Programmable switching mode

Applications of CPLD

- It is used in a digital circuit where Number of input and output are > 32.
- Television and Automation Industries.
- Implementation of large digital circuits.

Field Programmable Gate Arrays (FPGA)

- A **Field-Programmable Gate Array** is an integrated circuit silicon chip which has array of logic gates and this array can be programmed in the field i.e. the user can overwrite the existing configurations with its new defined configurations and can create their own digital circuit on field. The FPGAs can be considered as blank slate. FPGAs do nothing by itself whereas it is up to designers to create a configuration file often called a **bit file for the FPGA**. The FPGA will behave like the digital circuit once it is loaded with a bit file.
- Field Programmable Gate Arrays (**FPGAs**) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects.
- FPGAs are particularly useful for prototyping application-specific integrated circuits (ASICs) or processors.
- An FPGA can be reprogrammed until the processor design is final and bug-free.

FPGA Architecture

- An FPGA has a regular structure of logic cells or modules and interlinks which is under the developers and designers complete control. The FPGA is built with mainly three major blocks such as Configurable Logic Block (CLB), I/O Blocks or Pads and Switch Matrix/ Interconnection Wires. Each block will be discussed below in brief.
- CLB (Configurable Logic Block): These are the basic cells of FPGA. It consists of one 8-bit function generator, two 16-bit function generators, two registers (flip-flops or

latches), and reprogrammable routing controls (multiplexers). The CLBs are applied to implement other designed function and macros. Each CLBs have inputs on each side which makes them flexile for the mapping and partitioning of logic.

- I/O Pads or Blocks: The Input/Output pads are used for the outside peripherals to access the functions of FPGA and using the I/O pads it can also communicate with FPGA for different applications using different peripherals.
- Switch Matrix/ Interconnection Wires: Switch Matrix is used in FPGA to connect the long and short interconnection wires together in flexible combination. It also contains the transistors to turn on/off connections between different lines.
-

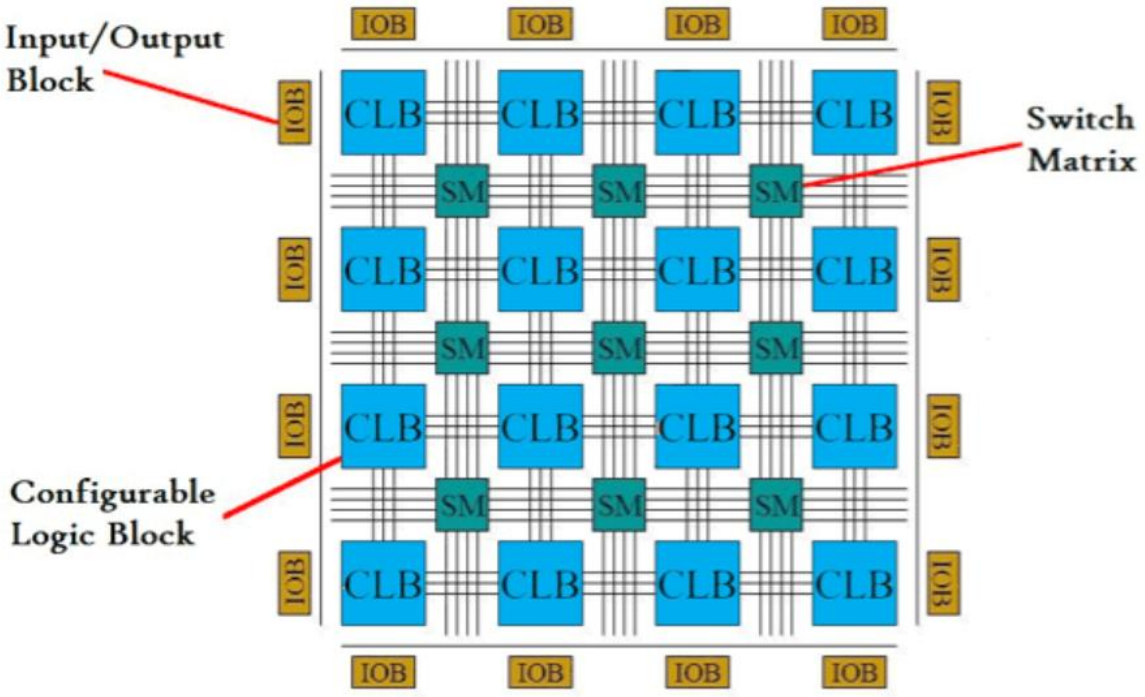


Fig 12: Block Diagram of FPGA

Configurable logic blocks:

Each Configurable logic block can generate a Logic function with many inputs.

Interconnection Switches:

They are used to interconnect various block with input/output blocks.

Application:

FPGA most commonly used in Digital systems such as smart phones, Computer systems etc.

CPLD vs FPGA comparison summary

	CPLD	FPGA
1.	Instant-on. CPLDs start working as soon as they are powered up	Since FPGA has to load configuration data from external ROM and setup the fabric before it can start functioning, there is a time delay between power ON and FPGA starts working. The time delay can be as large as several tens of milliseconds.
2.	Non-volatile. CPLDs remain programmed, and retain their circuit after powering down. FPGAs go blank as soon as powered-off.	FPGAs uses SRAM based configuration storage. The contents of the memory is lost as soon as power is disconnected.
3.	Deterministic Timing Analysis. Since CPLDs are comparatively simpler to FPGAs, and the number of interconnects are less, the timing analysis can be done much more easily.	Size and complexity of FPGA logic can be humongous compared to CPLDs. This opens up the possibility less deterministic signal routing and thus causing complicated timing scenarios. Thankfully implementation tools provided by FPGA vendors have mechanisms to assist achieving deterministic timing. But additional steps by the user is usually necessary to achieve this.
4.	Lower idle power consumption. Newer CPLDs such as CoolRunner-II use around 50 uA in idle conditions.	Relatively higher idle power consumption.
5.	Might be cheaper for implementing simpler circuits	FPGAs are much more capable compared to CPLDs but can be more expensive as well.
6.	More "secure" due to design storage within built in non-volatile memory.	FPGAs that use external memory can expose the IP externally. Many FPGA vendors offer mechanisms such as encryption to combat this. Design specific protection mechanisms also can be implemented.

7.	Very small amount of logic resources.	Massive amount logic and storage elements, with which incredibly complex circuits can be designed. FPGAs have thousands times more resources! This point alone makes FPGAs more popular than CPLDs.
8.	No on-die hard IPs available to offload processing from the logic fabric.	Variety of on-die dedicated hardware such as Block RAM, DSP blocks, PLL, DCMs, Memory Controllers, Multi-Gigabit Transceivers etc give immense flexibility. This is not even thinkable with CPLDs.
9.	Power down and reprogramming is always required in order to modify design functionality.	FPGAs can change their circuit even while running! (Since it is just a matter of updating LUTs with different content) This is called Partial Reconfiguration, and is very useful when FPGAs need to keep running a design and at the same time update the it with different design as per requirement. This feature is widely used in Accelerated Computing.