Ex.No.: 1
Date:

Design and Simulation of Adders using Verilog HDL

AIM:

To design and simulate adders using Verilog HDL.

APPARATUS REQUIRED:

1. Xilinx Vivado 2017.4
2. PC with windows OS.

THEORY:

HALF ADDER:

A *half-adder* is a combinational circuit that can be used to add two binary bits. It has two inputs that represent the two bits to be added and two outputs, with one producing the SUM output and the other producing the CARRY.

FULL ADDER:

A *full adder* is a combinational circuit that forms the arithmetic sum of three input bits. Two of the input variables, represent the significant bits to be added. The third input represents the carry from previous lower significant position.

RIPPLE CARRY ADDER:

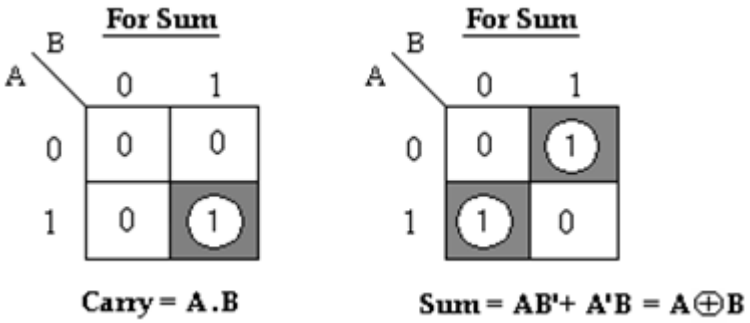
The 4-bit binary adder using full adder circuits is capable of adding two 4-bit numbers resulting in a 4-bit sum and a carry output. The bits are added with full adders, starting from the least significant position, to form the sum it and carry bit. The input carry C_0 in the least significant position must be 0. The carry output of the lower order stage is connected to the carry input of the next higher order stage.

Half Adder:

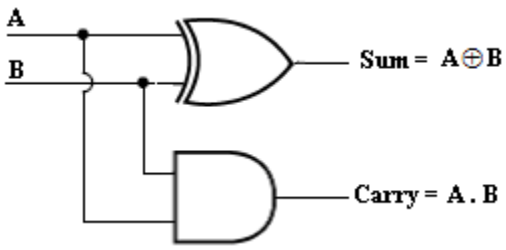
Truth table

Inputs		Outputs	
A	**B**	**Carry (C)**	**Sum (S)**
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

K-map Simplification:



Logic Diagram



Program:

Output:

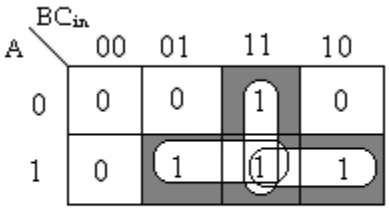
Full Adder:

Truth table

Inputs			Outputs	
A	**B**	**C_{in}**	**Sum (S)**	**Carry (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

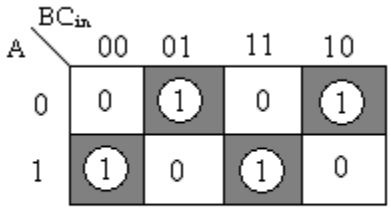
K-map Simplification:

For Carry



Carry, C_{out} = $AB + AC_{in} + BC_{in}$

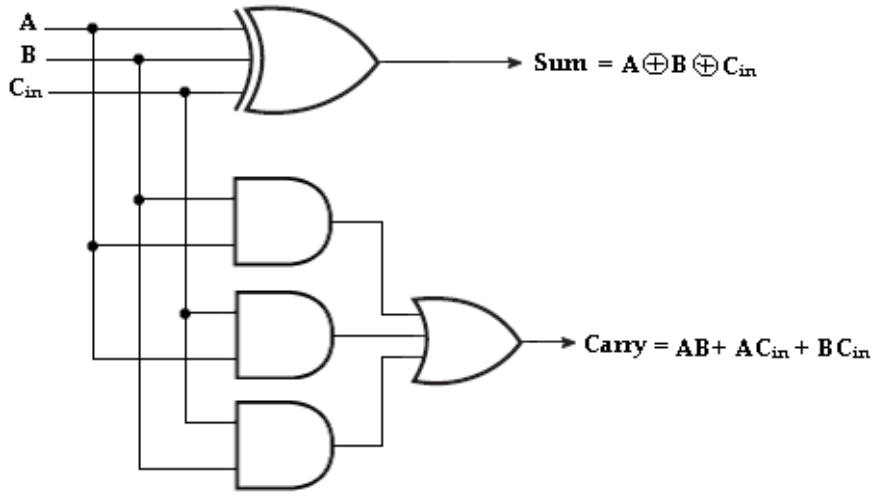
For Sum



Sum, S = $A'B'C_{in} + A'BC'_{in} + AB'C'_{in} + ABC_{in}$

= $A \oplus B \oplus C_{in}$

Logic Diagram

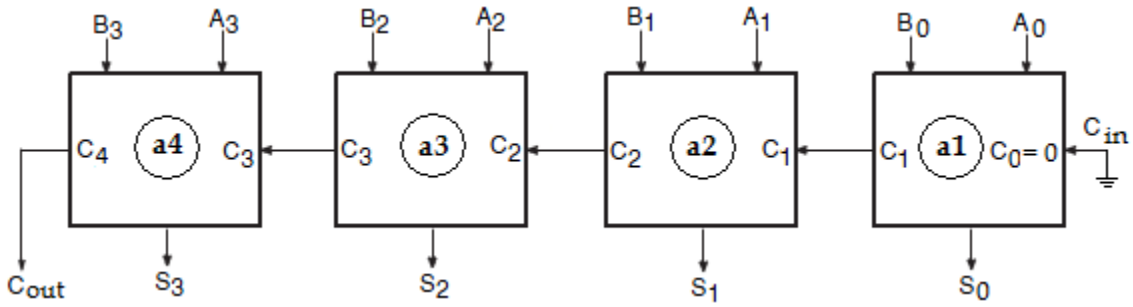


Program:

Output:

4-bit Binary Adder or Ripple Carry Adder:

Block Diagram



Program:

Output:

reparation	**/20**
Observation & Result	**/25**
Viva	**/10**
Record/ Regularity& Involvement	**/20**
Total	**/75**
Staff Sign	

RESULT:

Thus the adders are designed and simulated using Verilog HDL.

Ex.No.: 2
Date:

Design and Simulation of Flip-Flops using Verilog HDL

AIM:

To design and simulate Flip-flops using Verilog HDL.

APPARATUS REQUIRED:

1. Xilinx Vivado 2017.4
2. PC with windows OS.

THEORY:

D FLIP-FLOP:

The D flip-flop tracks the input, making transitions with match those of the input D. The D stands for "data"; this flip-flop stores the value that is on the data line. It can be thought of as a basic memory cell. A D flip-flop can be made from a set/reset flip-flop by tying the set to the reset through an inverter.

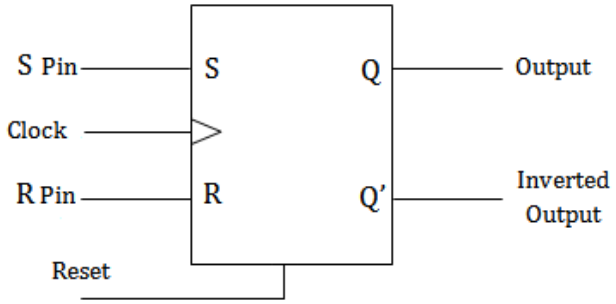
T FLIP-FLOP:

The T or "toggle"flip-flop changes its output on each clock edge, giving an output which is half the frequency of the signal to the T input. t is useful for constructing binary counters, frequency dividers, and general binary addition devices. It can be made from a J-K flip-flop by tying both of its inputs high.

JK FLIP-FLOP:

The J-K flip-flop is the most versatile of the basic flip-flops. It has the input-following character of the clocked D flip-flop but has two inputs, traditionally labelled J and K. If J and K are different then the output Q takes the value of J at the next clock edge. f J and K are both low then no change occurs. If J and K are both high at the clock edge then the output will toggle from one state to the other. It can perform the functions of the set/reset flip-flop and has the advantage that there are no ambiguous states. It can also act as a T flip-flop to accomplish toggling action if J and K are tied together. This toggle application finds extensive use in binary counters.

SR Flip-flop:



Symbol: SR Flip-flop

Inputs		Output	State
R	**S**	**Q_{n+1}**	
0	0	Q_n	No Change
0	1	0	Reset
1	0	1	Set
1	1	X	Indeterminate

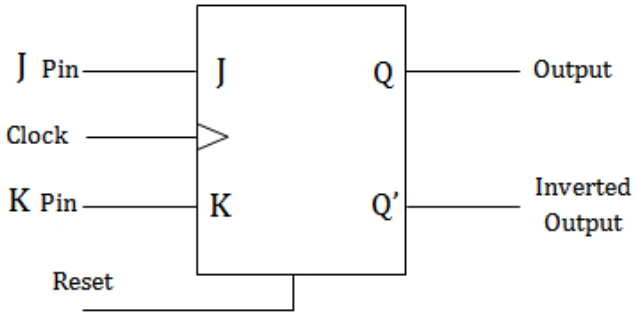
SR FLIP-FLOP:

The set/reset type flip-flop is triggered to a high state at Q by the "set" signal and holds that value until reset to low by a signal at the Reset input. This can be implemented as a NAND gate latch or a NOR gate latch and as a clocked version. One disadvantage of the S/R flip-flop is that the input S=R=0 gives ambiguous results and must be avoided. The J-K flip-flop gets around that problem.

Program:

Output:

JK Flip-flop:



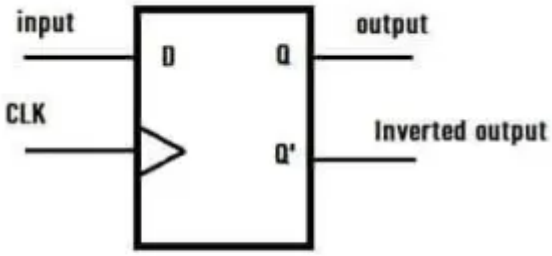
Symbol: JK Flip-flop

Inputs		Output	State
J	**K**	**Q_{n+1}**	
0	0	Q_n	No Change
0	1	0	Reset
1	0	1	Set
1	1	Q_n'	Toggle

Program:

Output:

D Flip-flop:



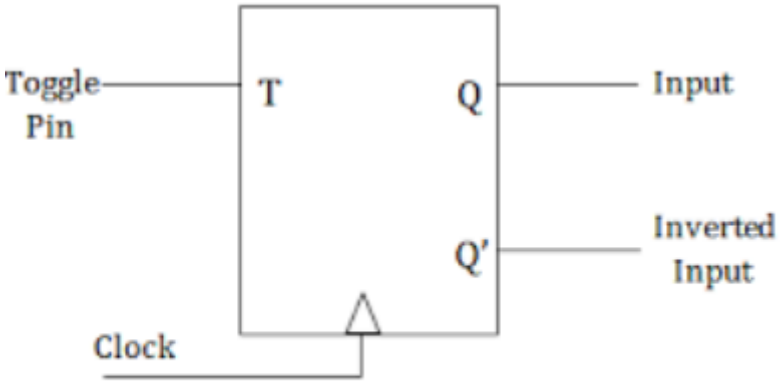
D flip flop Symbol

Inputs		Output
D	**CLK**	**Q_{n+1}**
0	0	0
0	1	0
1	0	0
1	1	1

Program:

Output:

T Flip-flop:



Symbol: T Flip-flop

Inputs	Output
T	**Q_{n+1}**
0	**Q_n**
1	**$\sim Q_n$**

Program:

Output:

Preparation	**/20**
Observation & Result	**/25**
Viva	**/10**
Record/ Regularity& Involvement	**/20**
Total	**/75**
Staff Sign	

RESULT:

Thus JK,SR,D and T Flip-flops are designed and simulated using Verilog HDL.

Ex.No.: 3
Date: **Design and Implementation of 4 bit Asynchronous Counter using FPGA**

AIM:

To design and implement a 4-bit Asynchronous counter using FPGA board.

APPARATUS REQUIRED:

1. Xilinx Vivado 2017.4
2. PC with windows OS.
3. FPGA Board - Artix 7

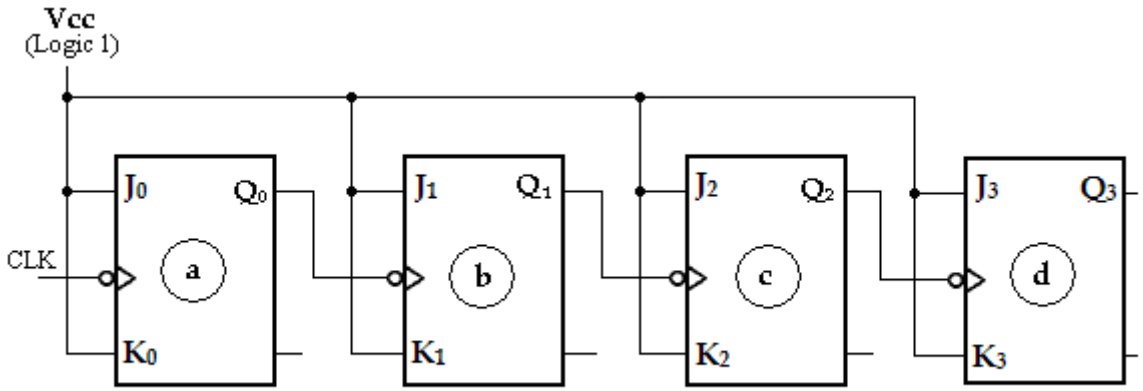
THEORY:

ASYNCHRONOUS COUNTER:

A ripple counter is an asynchronous counter where only the first flip-flop is clocked by an external clock. All subsequent flip-flops are clocked by the output of the preceding flip-flop. Asynchronous counters are also called ripple-counters because of the way the clock pulse ripples it way through the flip-flops.

A counter may count up or count down or count up and down depending on the input control. The count sequence usually repeats itself. When counting up, the count sequence goes from 0000, 0001, 0010, …1110, 1111 , 0000, 0001, ... etc. When counting down the count sequence goes in the opposite manner: 1111, 1110, ... 0010, 0001, 0000, 1111, 1110, ... etc. Asynchronous counters are slower than synchronous counters because of the delay in the transmission of the pulses from flip-flop to flip-flop.

4-bit ASynchronous Binary Counter:



Program:

Output:

Preparation	**/20**
Observation & Result	**/25**
Viva	**/10**
Record/ Regularity& Involvement	**/20**
Total	**/75**
Staff Sign	

RESULT:

Thus a 4-bit Asynchronous counter is designed and implemented using FPGA board.

Ex.No.: 4
Date: **Design and Implementation of 4 bit Synchronous Counter using FPGA**

AIM:

To design and implement a 4-bit synchronous counter using FPGA board.

APPARATUS REQUIRED:

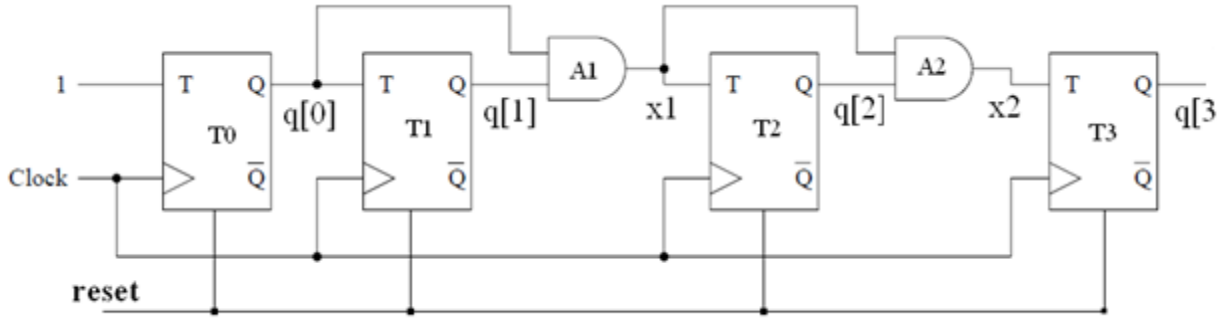
1. Xilinx Vivado 2017.4
2. PC with windows OS.
3. FPGA Board - Artix 7

THEORY:

SYNCHRONOUS COUNTER:

In synchronous counters, the clock inputs of all the flip-flops are connected together and are triggered by the input pulses. Thus, all the flip-flops change state simultaneously (in parallel). The circuit below is a 4-bit synchronous counter. The J and K inputs of FF0 are connected to HIGH. FF1 has its J and K inputs connected to the output of FF0, and the J and K inputs of FF2 are connected to the output of an AND gate that is fed by the outputs of FF0 and FF1. A simple way of implementing the logic for each bit of an ascending counter (which is what is depicted in the image to the right) is for each bit to toggle when all of the less significant bits are at a logic high state. For example, bit 1 toggles when bit 0 is logic high; bit 2 toggles when both bit 1 and bit 0 are logic high; bit 3 toggles when bit 2, bit 1 and bit 0 are all high; and so on.

4-bit Synchronous Binary Counter:



Program:

Output:

Preparation	**/20**
Observation & Result	**/25**
Viva	**/10**
Record/ Regularity& Involvement	**/20**
Total	**/75**
Staff Sign	

RESULT:

Thus a 4-bit synchronous counter is designed and implemented using FPGA board.

Ex.No.: 5
Date:

Design and Simulate a CMOS Inverter using Cadence EDA

AIM:

To simulate and Analyse the functionality of CMOS Inverter using Cadence EDA.

APPARATUS REQUIRED:

1. Cadence EDA
2. PC with Linux Os

THEORY:

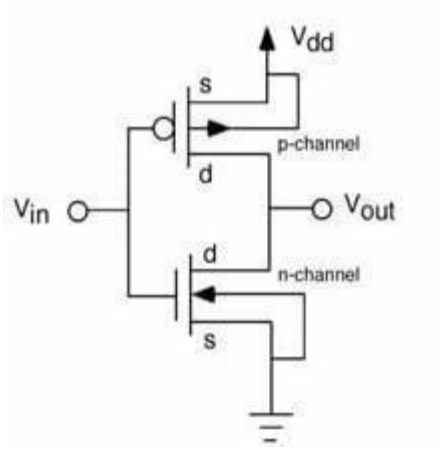
CMOS INVERTER

A CMOS inverter contains a PMOS and a NMOS transistor connected at the drain and gate terminals, a supply voltage VDD at the PMOS source terminal, and a ground connected at the NMOS source terminal, were V_{IN} is connected to the gate terminals and V_{OUT} is connected to the drain terminals. The CMOS design does not contain any resistors, which makes it more power efficient that a regular resistor-MOSFET inverter. As the voltage at the input of the CMOS device varies between 0 and 5 volts, the state of the NMOS and PMOS varies accordingly. When V_{IN} is low, the NMOS is "off", while the PMOS stays "on": instantly charging V_{OUT} to logic high. When Vin is high, the NMOS is "on and the PMOS is "on: draining the voltage at V_{OUT} to logic low.

The switch logic of the inverter is shown in the below table.

A	PULL DOWN NETWORK	PULL UP NETWORK
0	OFF	ON
1	ON	OFF

CMOS Inverter:



Vin	Vout
0	1
1	0

Design Specification:

S.No.	COMPONENT	PARAMETER	COMPONENT NAME	DESIGN VALUE
1	MOSFET (nmos 180nm) (pmos 180nm)	WIDTH (W)	NM0	2 u
			NM1	2 u
2	DC SOURCE	DC VOLTAGE	Vdc	1.8 V
			Vin	1.8 V

OUTPUT:

Preparation	**/20**
Observation & Result	**/25**
Viva	**/10**
Record/ Regularity& Involvement	**/20**
Total	**/75**
Staff Sign	

RESULT:

Thus the functionality of CMOS Inverter is simulated and analysed using Cadence EDA.

Ex.No.: 6
Date: **Design and Simulate a two input CMOS NAND gate using Cadence EDA**

AIM:

To simulate and Analyse the functionality of CMOS NAND gate using Cadence EDA.

APPARATUS REQUIRED:

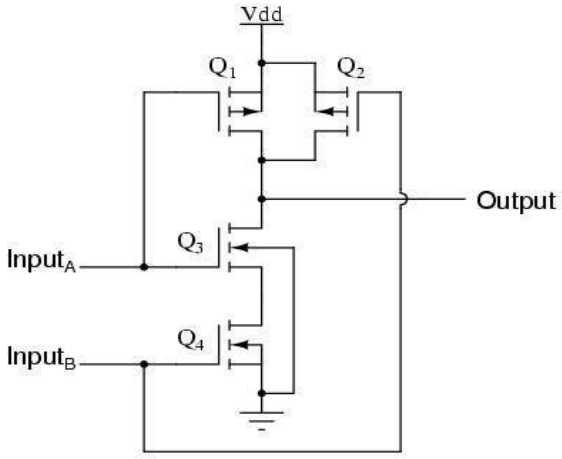
1. Cadence EDA
2. PC with Linux Os

THEORY:

CMOS NAND

It consists of two series NMOS transistors between output and Ground and two parallel PMOS transistors between output and VDD. The transistors Q_1 and Q_3 resemble the series-connected complementary pair from the inverter circuit. Both are controlled by the same input signal (input A), the upper transistor turning off and the lower transistor turning on when the input is "high" (1), and vice versa. Similarly Q_2 and Q_4 are similarly controlled by the same input signal (input B), and how they will also exhibit the same on/off behavior for the same input logic levels. The upper transistors of both pairs (Q_1 and Q_2) have their source and drain terminals paralleled, while the lower transistors (Q_3 and Q_4) are series-connected. This cause the output to go "high" (1) if *either* top transistor saturates, and to go "low" (0) only if *both* lower transistors saturate.

CMOS NAND:



A	B	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

Design Specification:

S.No.	COMPONENT	PARAMETER	COMPONENT NAME	DESIGN VALUE
1	MOSFET (nmos 180nm) (pmos 180nm)	WIDTH (W)	NM0	2 u
			NM1	2 u
3	DC SOURCE	DC VOLTAGE	Vdc	1.8 V
			Input A Input B	1.8 V 1.8 V

OUTPUT:

Preparation	**/20**
Observation & Result	**/25**
Viva	**/10**
Record/ Regularity& Involvement	**/20**
Total	**/75**
Staff Sign	

RESULT:

Thus the functionality of CMOS NAND gate is simulated and analysed using Cadence EDA.

Ex.No.: 7
Date: **Design and Simulate a two input CMOS NOR gate using Cadence EDA**

AIM:

To simulate and Analyse the functionality of CMOS NOR gate using Cadence EDA.

APPARATUS REQUIRED:

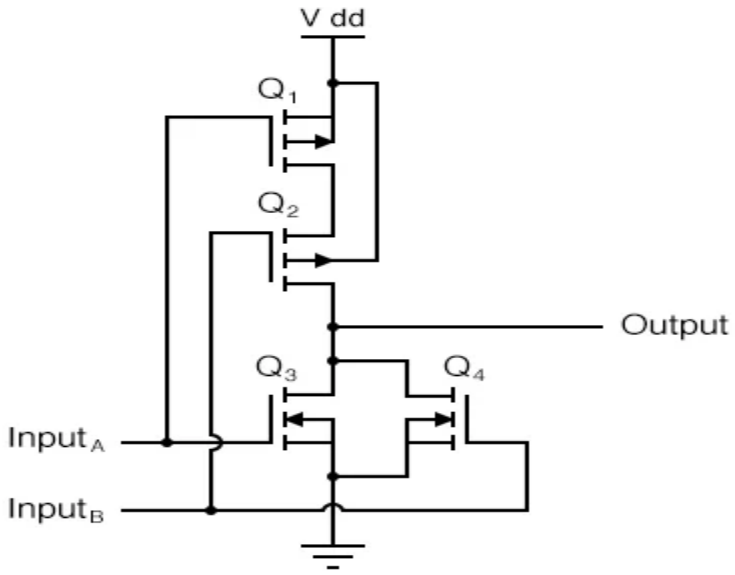
1. Cadence EDA
2. PC with Linux Os

THEORY:

CMOS NOR

A CMOS NOR gate circuit uses four MOSFETs just like the NAND gate, except that its transistors are differently arranged. Instead of two paralleled *sourcing* (upper) transistors connected to V_{dd} and two series-connected *sinking* (lower) transistors connected to ground, the NOR gate uses two series-connected sourcing transistors and two parallel-connected sinking transistors. As with the NAND gate, transistors Q_1 and Q_3 work as a complementary pair, as do transistors Q_2 and Q_4. Each pair is controlled by a single input signal. If *either* input A *or* input B are "high" (1), at least one of the lower transistors (Q_3 or Q_4) will be saturated, thus making the output "low" (0). Only in the event of *both* inputs being "low" (0) will both lower transistors be in cutoff mode and both upper transistors be saturated, the conditions necessary for the output to go "high" (1). This behavior, defines the NOR logic function.

CMOS NOR Gate:



A	B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

Design Specification:

S.No.	COMPONENT	PARAMETER	COMPONENT NAME	DESIGN VALUE
1	MOSFET (nmos 180nm) (pmos 180nm)	WIDTH (W)	NM0	2 u
			NM1	2 u
3	DC SOURCE	DC VOLTAGE	Vdc	1.8 V
			Input A	1.8 V
			Input B	1.8 V

OUTPUT:

Preparation	**/20**
Observation & Result	**/25**
Viva	**/10**
Record/ Regularity& Involvement	**/20**
Total	**/75**
Staff Sign	

RESULT:

Thus the functionality of CMOS NOR gate is simulated and analysed using Cadence EDA.

Ex.No.: 8
Date:

Design an CMOS Inverter Layout using Cadence EDA

AIM:

To design and Analyse the functionality of CMOS Inverter layout using Cadence EDA.

APPARATUS REQUIRED:

1. Cadence EDA
2. PC with Linux Os

THEORY:

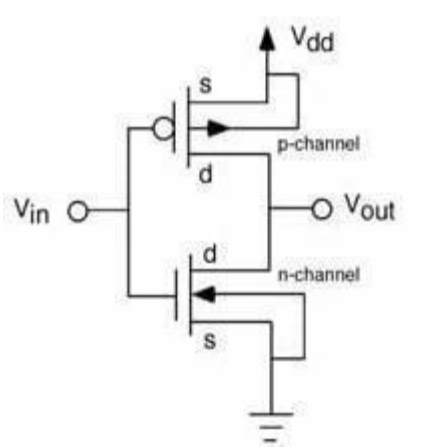
CMOS INVERTER

A CMOS inverter contains a PMOS and a NMOS transistor connected at the drain and gate terminals, a supply voltage VDD at the PMOS source terminal, and a ground connected at the NMOS source terminal, were V_{IN} is connected to the gate terminals and V_{OUT} is connected to the drain terminals. The CMOS design does not contain any resistors, which makes it more power efficient that a regular resistor-MOSFET inverter. As the voltage at the input of the CMOS device varies between 0 and 5 volts, the state of the NMOS and PMOS varies accordingly. When V_{IN} is low, the NMOS is "off", while the PMOS stays "on": instantly charging V_{OUT} to logic high. When Vin is high, the NMOS is "on and the PMOS is "on: draining the voltage at V_{OUT} to logic low.

The switch logic of the inverter is shown in the below table.

A	PULL DOWN NETWORK	PULL UP NETWORK
0	OFF	ON
1	ON	OFF

CMOS Inverter:



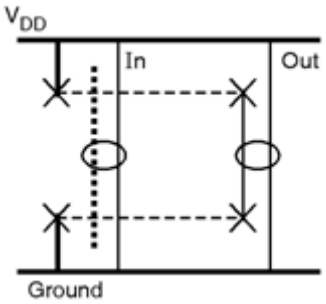
Vin	Vout
0	1
1	0

Design Specification:

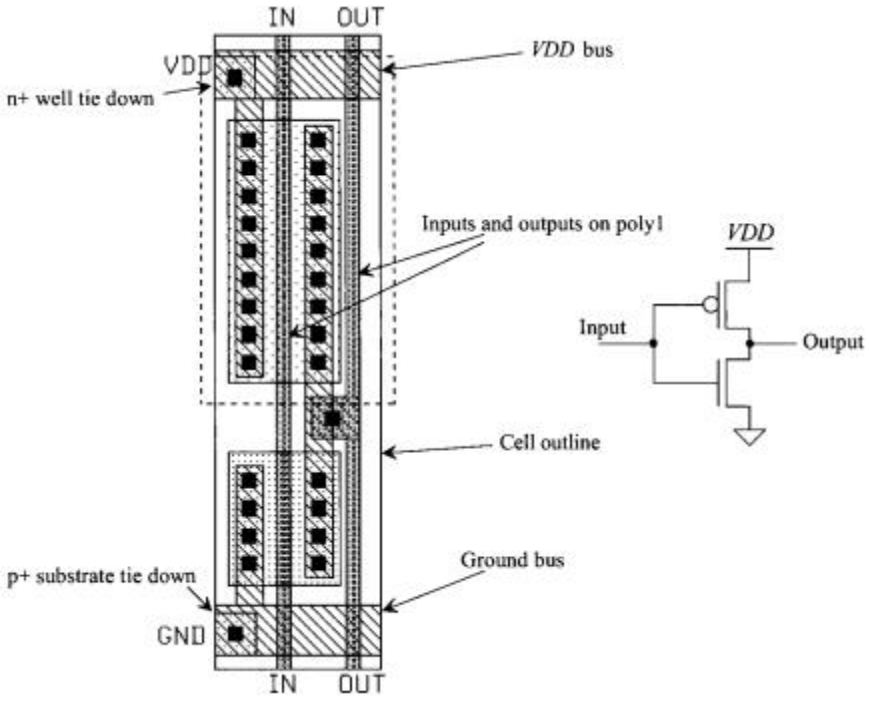
S.No.	COMPONENT	PARAMETER	COMPONENT NAME	DESIGN VALUE
1	MOSFET (nmos 180nm) (pmos 180nm)	WIDTH (W)	NM0	2 u
			NM1	2 u
2	DC SOURCE	DC VOLTAGE	Vdc	1.8 V
			Vin	1.8 V

LAYOUT DESIGN

The inital phase of layout design can be simplified significantly by the use of stick diagrams - or so-called symbolic layouts. Here, the detailed layout design rules are simply neglected and the main features (active areas, poly silicon lines, metal lines) are represented by constant width rectangles or simple sticks. The purpose of the stick diagram is to provide the designer a good understanding of the topological constraints, and to quickly test several possibilities for the optimum layout without actually drawing a complete mask diagram.



Stick Diagram



Layout of CMOS Inverter

OUTPUT:

Preparation	**/20**
Observation & Result	**/25**
Viva	**/10**
Record/ Regularity& Involvement	**/20**
Total	**/75**
Staff Sign	

RESULT:

Thus the layout of CMOS Inverter is designed and analysed using Cadence EDA.

EX.No.: 9
DATE: **Design the layout of CMOS NAND and NOR gate using Cadence EDA**

AIM:

To design and Analyse the functionality of CMOS NAND and NOR layout using Cadence EDA.

APPARATUS REQUIRED:

1. Cadence EDA
2. PC with Linux Os

THEORY:

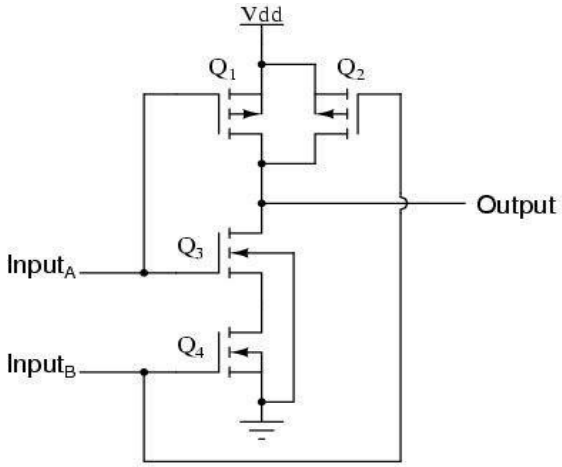
CMOS NAND

It consists of two series NMOS transistors between output and Ground and two parallel PMOS transistors between output and VDD. The transistors Q_1 and Q_3 resemble the series-connected complementary pair from the inverter circuit. Both are controlled by the same input signal (input A), the upper transistor turning off and the lower transistor turning on when the input is "high" (1), and vice versa. Similarly Q_2 and Q_4 are similarly controlled by the same input signal (input B), and how they will also exhibit the same on/off behavior for the same input logic levels. The upper transistors of both pairs (Q_1 and Q_2) have their source and drain terminals paralleled, while the lower transistors (Q_3 and Q_4) are series-connected. This cause the output to go "high" (1) if *either* top transistor saturates, and to go "low" (0) only if *both* lower transistors saturate.

CMOS NOR

A CMOS NOR gate circuit uses four MOSFETs just like the NAND gate, except that its transistors are differently arranged. Instead of two paralleled *sourcing* (upper) transistors connected to V_{dd} and two series-connected *sinking* (lower) transistors connected to ground, the NOR gate uses two series-connected sourcing transistors and two parallel-connected sinking

CMOS NAND:



A	B	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

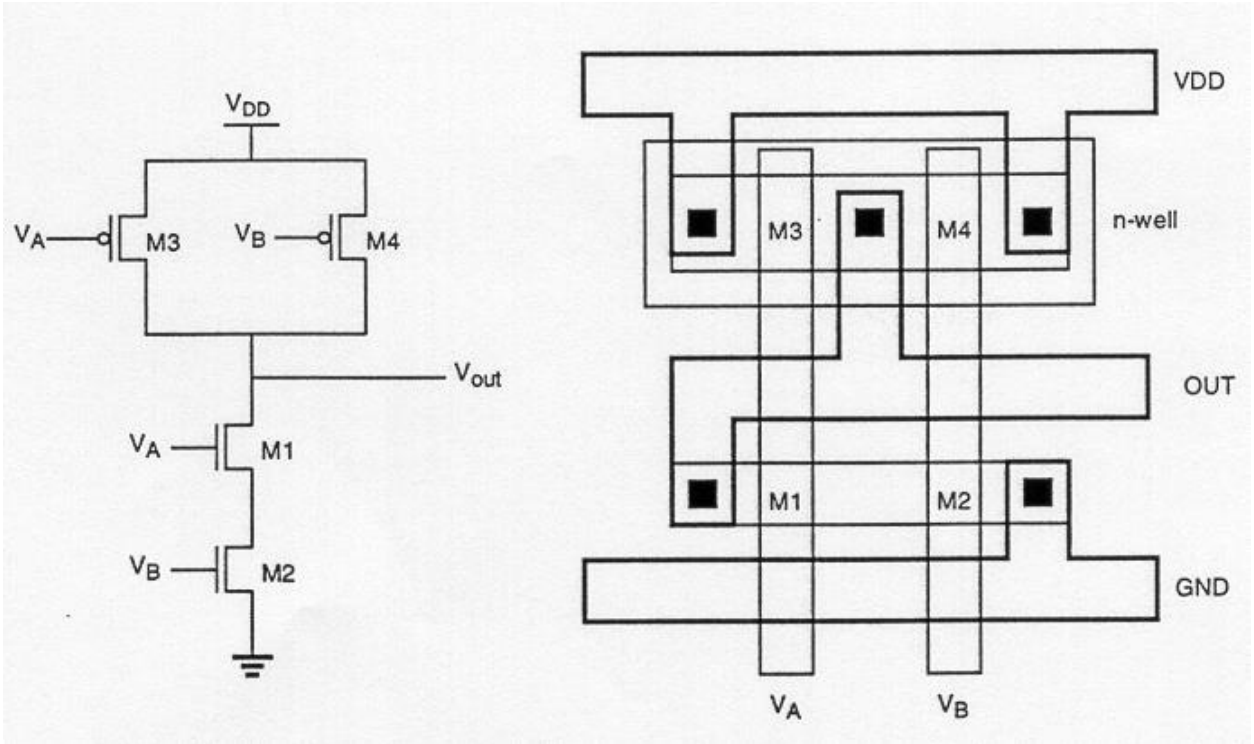
Design Specification:

S.No.	COMPONENT	PARAMETER	COMPONENT NAME	DESIGN VALUE
1	MOSFET (nmos 180nm) (pmos 180nm)	WIDTH (W)	NM0	2 u
			NM1	2 u
3	DC SOURCE	DC VOLTAGE	Vdc	1.8 V
			Input A Input B	1.8 V 1.8 V

transistors. As with the NAND gate, transistors Q_1 and Q_3 work as a complementary pair, as do transistors Q_2 and Q_4. Each pair is controlled by a single input signal. If *either* input A *or* input B are "high" (1), at least one of the lower transistors (Q_3 or Q_4) will be saturated, thus making the output "low" (0). Only in the event of *both* inputs being "low" (0) will both lower transistors be in cutoff mode and both upper transistors be saturated, the conditions necessary for the output to go "high" (1). This behavior, defines the NOR logic function.

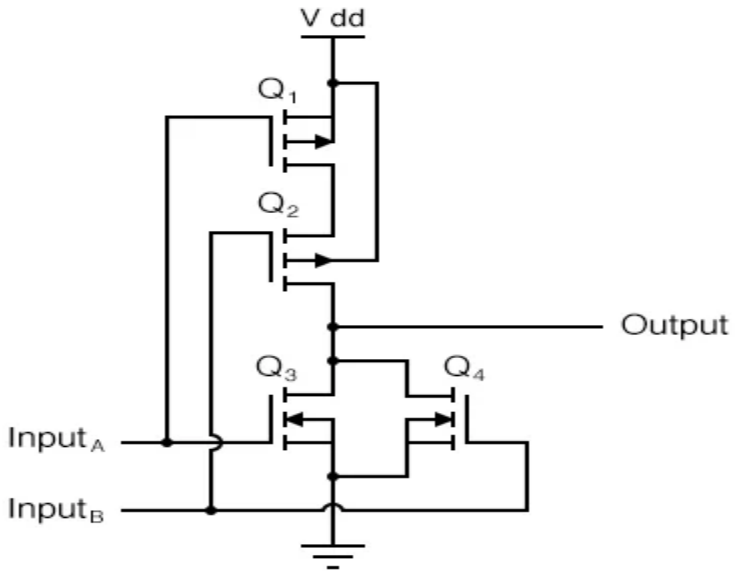
LAYOUT DESIGN

The inital phase of layout design can be simplified significantly by the use of stick diagrams - or so-called symbolic layouts. Here, the detailed layout design rules are simply neglected and the main features (active areas, poly silicon lines, metal lines) are represented by constant width rectangles or simple sticks. The purpose of the stick diagram is to provide the designer a good understanding of the topological constraints, and to quickly test several possibilities for the optimum layout without actually drawing a complete mask diagram.



CMOS NAND

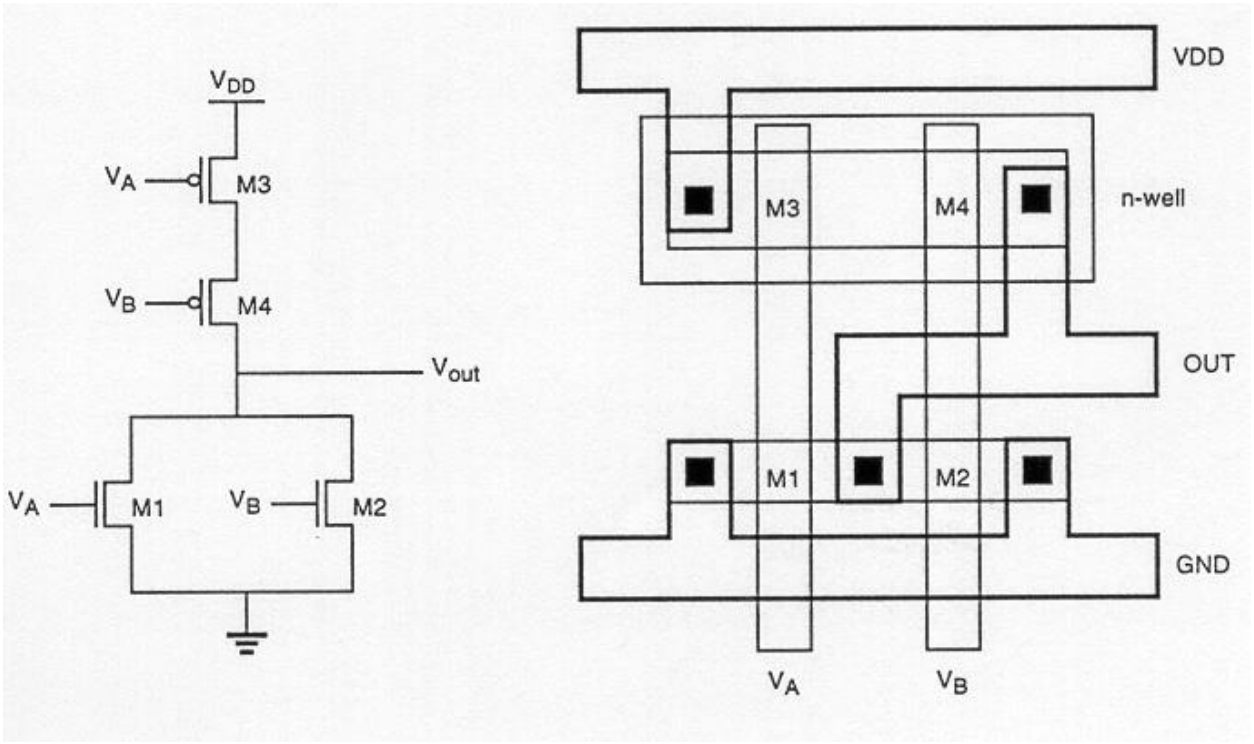
CMOS NOR Gate:



A	B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

Design Specification:

S.No.	COMPONENT	PARAMETER	COMPONENT NAME	DESIGN VALUE
1	MOSFET (nmos 180nm) (pmos 180nm)	WIDTH (W)	NM0	2 u
			NM1	2 u
3	DC SOURCE	DC VOLTAGE	Vdc	1.8 V
			Input A	1.8 V
			Input B	1.8 V



CMOS NOR

OUTPUT:

Preparation	**/20**
Observation & Result	**/25**
Viva	**/10**
Record/ Regularity& Involvement	**/20**
Total	**/75**
Staff Sign	

RESULT:

Thus the layout of CMOS NAND and NOR gates are designed and analysed using Cadence EDA.

Ex.No.: 10 **Design & Verification of 4-bit Ripple Carry Adder & D FF using System Verilog**
Date:

AIM:

To design and simulate 4 bit Ripple Carry Adder and D - FF using System Verilog.

APPARATUS REQUIRED:

1. Xilinx Vivado 2017.4
2. PC with windows OS.

THEORY:

System Verilog

SystemVerilog is an extension of Verilog with many such verification features that allow engineers to verify the design using complex testbench structures and random stimuli in simulation. A hardware design mostly consists of several Verilog (.v) files with one *top* module, in which all other sub-modules are instantiated to achieve the desired behavior and functionality. An environment called *testbench* is required for the verification of a given verilog design and is usually written in SystemVerilog these days. The idea is to drive the design with different stimuli to observe its outputs and compare it with expected values to see if the design is behaving the way it should.

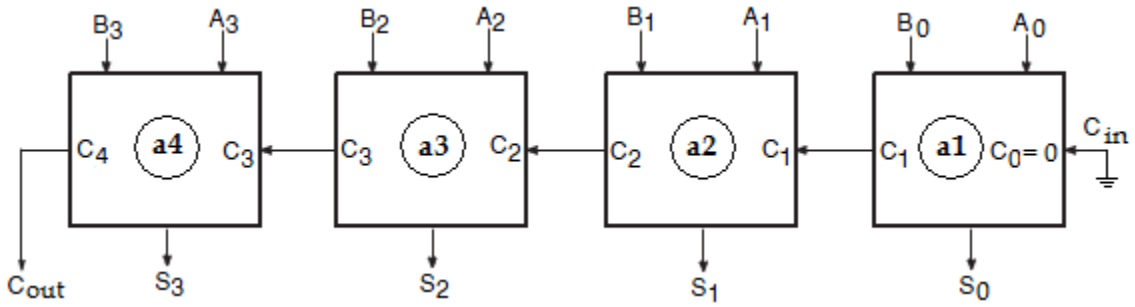
In order to do this, the top level design module is instantiated within the testbench environment, and design input/output ports are connected with the appropriate testbench component signals. The inputs to the design are driven with certain values for which we know how the design should operate. The outputs are analyzed and compared with the expected values to see if the design behavior is correct.

RIPPLE CARRY ADDER:

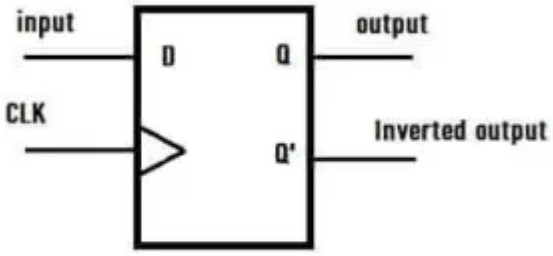
The 4-bit binary adder using full adder circuits is capable of adding two 4-bit numbers resulting in a 4-bit sum and a carry output. The bits are added with full adders, starting from the least significant position, to form the sum it and carry bit. The input carry

4-bit Binary Adder or Ripple Carry Adder:

Block Diagram



D Flip-flop:



D flip flop Symbol

Inputs		Output
D	**CLK**	**Q_{n+1}**
0	0	0
0	1	0
1	0	0
1	1	1

C_0 in the least significant position must be 0. The carry output of the lower order stage is connected to the carry input of the next higher order stage.

D FLIP-FLOP:

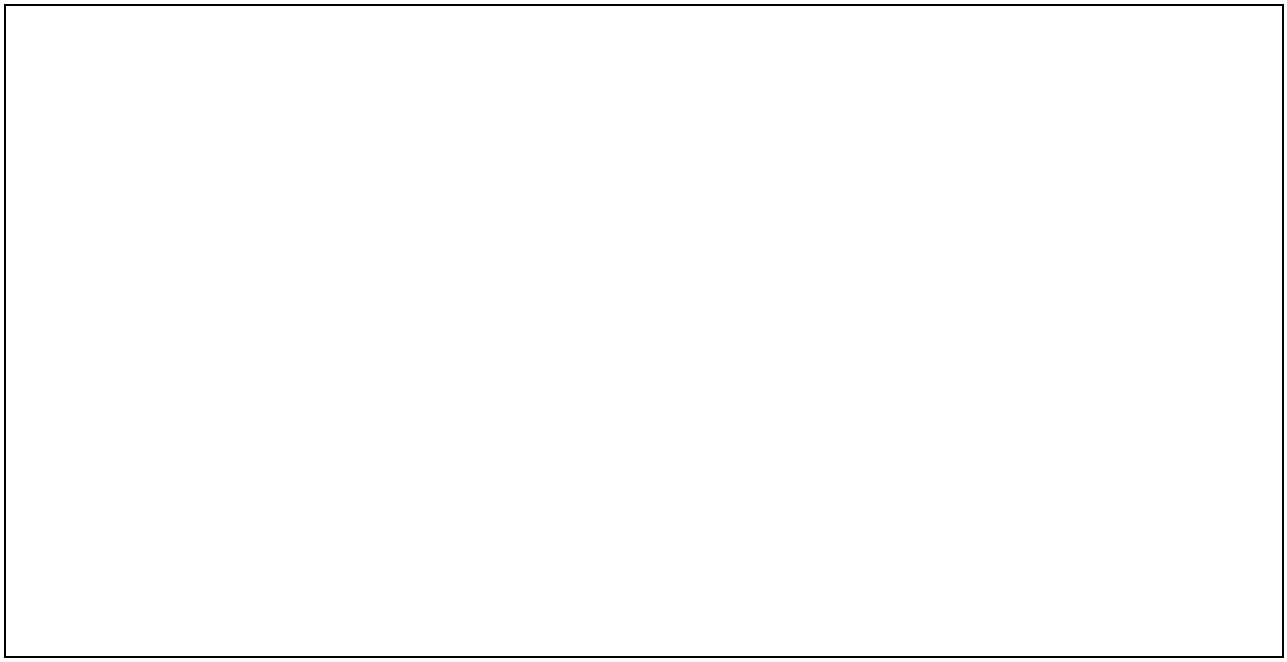
The D flip-flop tracks the input, making transitions with match those of the input D. The D stands for "data"; this flip-flop stores the value that is on the data line. It can be thought of as a basic memory cell. A D flip-flop can be made from a set/reset flip-flop by tying the set to the reset through an inverter.

Program: (D FF)

Output:

Program: (RCA)

Output:



Preparation	**/20**
Observation & Result	**/25**
Viva	**/10**
Record/ Regularity& Involvement	**/20**
Total	**/75**
Staff Sign	

RESULT:

Thus a 4 bit Ripple Carry Adder and D FF is designed and simulated using system Verilog.