

### **LIST OF EXPERIMENT**

<b>SN.</b>	<b>EXPERIMENT DESCRIPTION</b>
01.	Write a VHDL a program and design basic 2-bit input combinational gates
02.	Write a VHDL program to design 2:1 MUX.
03.	Write a VHDL program to design 8:3 priority encoder by behavioral description.
04.	Write a VHDL program to design full adder using structural modeling.
05.	Write a VHDL program to design D flip-flop using behavioral description.
06.	Write a VHDL program to design JK flip-flop using behavioral description.
07.	To Design 4-bit counter by using behavioral description.
08.	To design 4-bit Even Parity Generator Using Structural Description

# Experiment-1

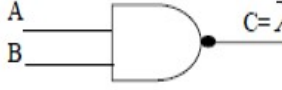
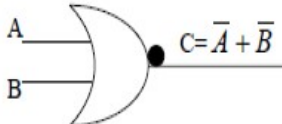

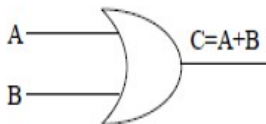

## 01. AIM:

Write a VHDL program and design basic 2-bit input combinational gates.

02. **FACILITIES/ ENVIRONMENT:** Computer system, Xilinx 13.4

03. **OBJECTIVE :** To know basic of combinational design and familiar with VHDL programming.

## 04. Truth Table:

S.NO	GATE	SYMBOL	IN
			A
1.	NAND IC 7400		0
			0
			1
			1
2.	NOR IC 7402		0
			0
			1
			1
3.	AND IC 7408		0
			0
			1
			1
4.	OR IC 7432		0
			0
			1
			1
5.	NOT		1

## 05. PROCEDURE :

- Open Project Navigator, Open New Project.
- Add proper project constraints with suitable package and language (VHDL).
- Write the VHDL code for given circuit.
- Check the syntax and synthesize the code.
- Observe the Timing and fitter report.

- f. Select behavioral mode and again add project constraints for Test bench Waveforms.
- g. Apply suitable stimulated inputs to input variables.
- h. Run the test bench waveforms and observe the waveforms.

## 06. VHDL CODE :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity and_or_top is
    Port ( INA1 : in  STD_LOGIC;  -- AND gate input
          INA2 : in  STD_LOGIC;  -- AND gate input
          OA  : out STD_LOGIC;  -- AND gate output
    end and_or_top;
```

```
architecture Behavioral of and_or_top is
begin
    OA <= INA1 and INA2;  -- 2 input AND gate
end Behavioral;
```

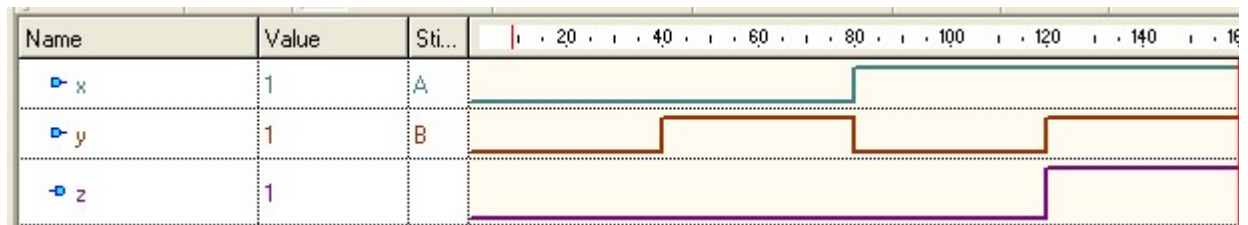
---

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

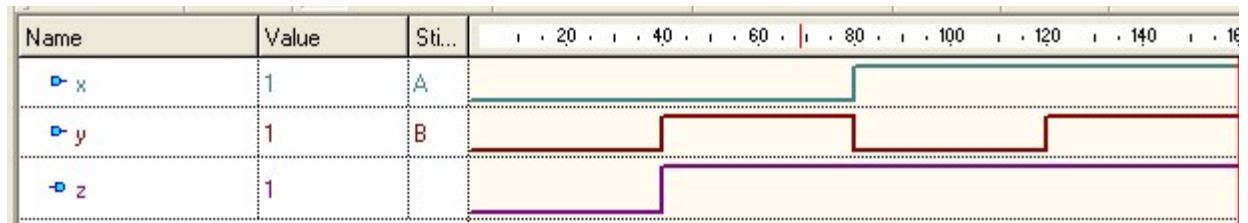
entity and_or_top is
    Port ( INA1 : in  STD_LOGIC;  -- AND gate input
          INA2 : in  STD_LOGIC;  -- AND gate input
          OA  : out STD_LOGIC;  -- AND gate output
    end and_or_top;
```

```
architecture Behavioral of and_or_top is
begin
    OA <= INA1 or INA2;  -- 2 input AND gate
end Behavioral;
```

## 07. RESULT :



**Simulated waveform of AND Gate**



**Simulated waveform of OR Gate**

08. **CONCLUSION:** Hence the truth table is verified and design basic 1-bit input combinational gates performed successfully.

## Experiment-2

### AIM:

To design 2:1 MUX with active low enable by using data flow description.

### FACILITIES/ ENVIORNMENT:

Computer system  
Xilinx-ISE9.1

### OBJECTIVE :

To know about data flow description of VHDL programming and understanding the working of 2:1 mux.

### THEORY:

#### DATA FLOW DESCRIPTION:

Data flow modeling style shows how the data flow from input to output. Data flow modeling style works on concurrent execution. The signal assignment statement `<=`, implies an assignment of a value to a signal. A signal assignment statement is executed only when any signal used in the expression on the RHS has an event on it i.e; the value for the signal changes. In data flow approach, circuits are described by indicating how the inputs and outputs of built in primitive components (eg. An AND gate) are connected together.

#### MULTIPLEXER:

Multiplexer is a device that selects one of the several analog or digital input signals and forwards the selected input into a single line. It has multiple input lines and single output line. A multiplexer of  $2^n$  inputs has  $n$  select lines, which are used to select which input line to send

to output. Multiplexer is also called as data selector, switching device. There can be any combination of multiplexer such as 2:1, 4:1 etc.

### 2:1 Mux

2:1 line multiplexer is a combinational circuit that uses one control switch to connect one of the two input data line to a single output. Only one of the input data lines can be aligned to the output of multiplexer at any given time. It is used to split any large Boolean function into two smaller parts.

## 09. TRUTH- TABLE :

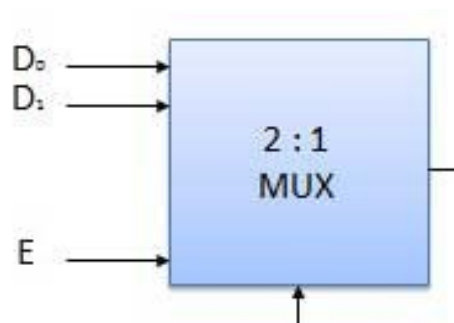
For Active low:

E	S	Y
0	0	D <sub>0</sub>
0	1	D <sub>1</sub>
1	0	X
1	1	X

## 10. PROCEDURE :

- Open File, Select New Project.
- Specify Project Name then Next.
- Click on New Source, Select VHDL Module.
- Specify File Name then Next.
- Specify input and output variables then Next then Finish.
- Type Code.
- Select “Synthesis XST” for compiling.
- Select “View RTL Schematic” for block diagram.
- Select “Behavioral Simulation” from drop-down list.
- Create New Source, click on “Test Bench Waveform”.
- Specify waveform name, then next and then finish.
- Select Combinational clock and GSR, then Finish.
- Select “Process” then simulate Behavioral Model.

## 11. DIAGRAM :



## 08. VHDL CODE :

```
library ieee;
use ieee.std_logic_164.all;
entity mux is
port(D0, D1 : inout std_logic;
      s, e : in std_logic;
      Y : out std_logic);
end mux;
Architecture mux_2 of mux is
begin
    D0 <= (not e) and (not s);
    D1 <= (not e) and s;
    Y <= D0 or D1;
end mux_2;
```

## 09. CONCLUSION :

Hence the truth table is verified and practical of 2:1 mux is performed successfully.

## Experiment-3

## 12. AIM:

### Design of combinational circuits using VHDL 8 to 3 priority encoder.

**13. FACILITIES/ ENVIORNMENT:** Computer system, Xilinx 13.4

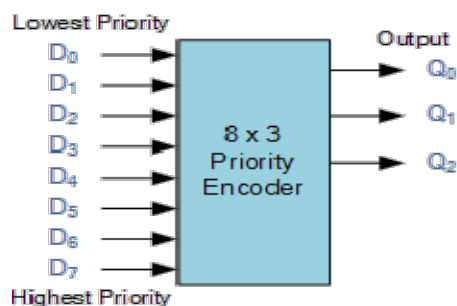
**14. OBJECTIVE :** To know how the about priority design works and processes syntax will used for designing such systems.

## 15. THEORY:

**Encoder:** A logic circuit that produces coded binary outputs from un-coded inputs.

**Priority encoder:** Whenever two or more inputs are applied at a time, internal hardware will check this condition and if the priority is set such that higher numbered input should be taken into account and remaining are considered as don't care then output code will be appear will be “higher numbered input”.

## 16. Truth Table



Inputs									Outputs		
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>		Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
0	0	0	0	0	0	0	1		0	0	0
0	0	0	0	0	0	1	x		0	0	1
0	0	0	0	0	1	x	x		0	1	0
0	0	0	0	1	x	x	x		0	1	1
0	0	0	1	x	x	x	x		1	0	0
0	0	1	x	x	x	x	x		1	0	1
0	1	x	x	x	x	x	x		1	1	0
1	x	x	x	x	x	x	x		1	1	1

## 17. PROCEDURE :

- a. Open Project Navigator, Open New Project.
- b. Add proper project constraints with suitable package and language (VHDL).
- c. Write the VHDL code for given circuit.
- d. Check the syntax and synthesize the code.
- e. Observe the Timing and fitter report.
- f. Select behavioral mode and again add project constraints for Test bench Waveforms.
- g. Apply suitable stimulated inputs to input variables.
- h. Run the test bench waveforms and observe the waveforms.

## 08. VHDL CODE :

### Program for 8:3 Encoder :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity encod is
Port ( a : in  STD_LOGIC_VECTOR (7 downto 0);
      y : out STD_LOGIC_VECTOR (2 downto 0));
end encod;

architecture Behavioral of encod is

begin
  process(a)
  begin
    if(a(7)='1')then
      y<="000";
    elsif(a(6)='1')then
      y<="001";
    elsif(a(5)='1')then
      y<="010";

    elsif(a(4)='1')then
      y<="011";
    elsif(a(3)='1')then
      y<="100";
```

```

    elsif(a(2)='1')then
        y<="101";
    elsif(a(1)='1')then
        y<="110";
    else
        y<= "XXX";
    end if;
end process;

end Behavioral;

```

## 10. CONCLUSION :

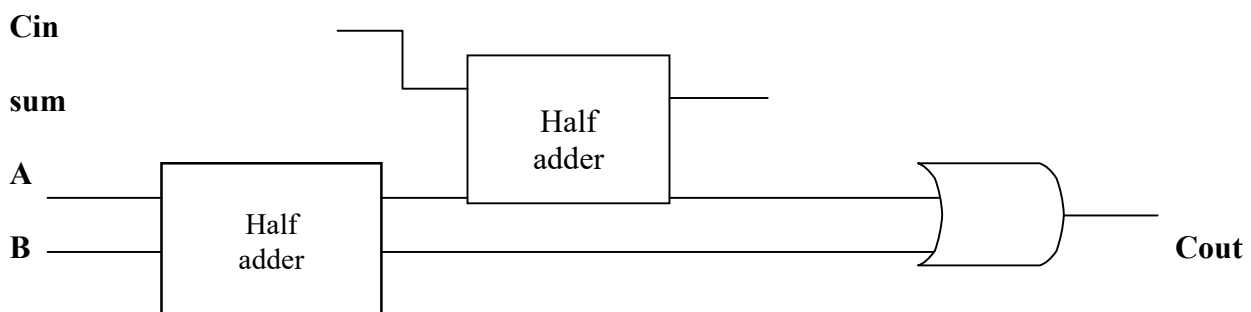
Hence the truth table is verified and Design of combinational circuits using VHDL 8 to 3 priority encoder performed successfully.

# Experiment-4

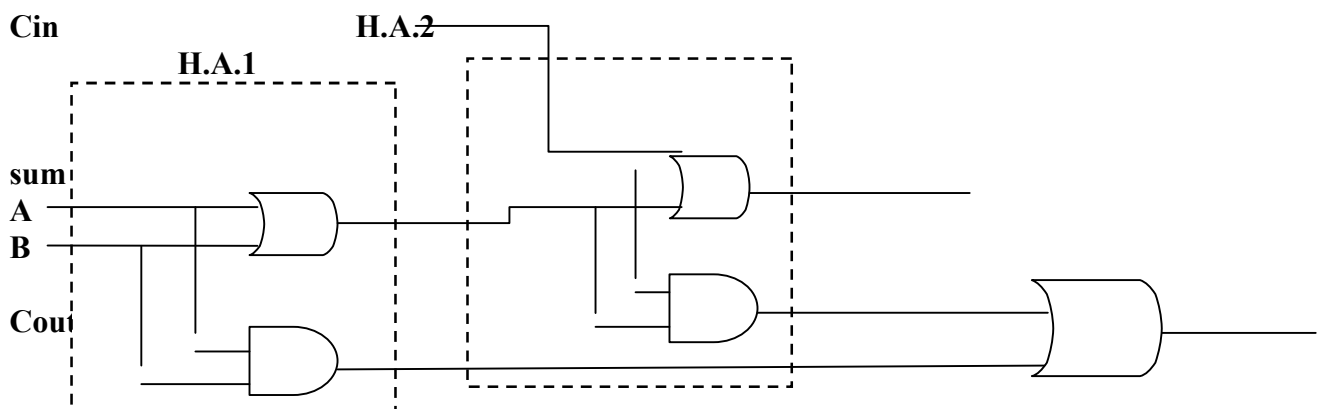
**01. AIM::** To design full adder using structural description.

**02.FACILITIES/ ENVIORNMENT:** Hardware:Computer System  
Software: Xilinx-ISE9.1

**03.DIAGRAM:**



**Block diagram**





## Full Adder by using two half adders

### 04. THEORY:

#### Structural Description-

It describes how the component are connected behavioral models are assume to exist in local working directory or in a library structural facility the use of hierarchy an abstraction in modelling complex system. Structural models can be integrated into model that also used process and CSA statement.

#### Adder-

Adder is a digital circuit that performs addition of number.

#### Full Adder-

A full adder adds binary number accounts for values carried in as well as carried out.

A 1 bit full adder adds their 1 bit number after written as A, B &  $C_{in}$ . A & B are operands and  $c_{in}$  is a bit carried in the form of previous bus significant stage the full adder is usually a component in a cascade of adders, which adds 8, 16, 32, etc bit binary number. The circuit produces a two bit output. Output carry and sum typically represented by the signals  $c_{out}$  and s, where  $sum = 2 * c_{out}$ .

### 05. TRUTH-TABLE :

A	B	Cin	sum	Cout
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

### 06. VHDL CODE :

#### Code for half adder.

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity ha is

    port (p,q : in std_logic;

          sum1,carry1 :out std_logic);

end ha;

architecture hlf of ha is

begin

    sum1 <= p xor q;
```

```
carry1 <= p and q;
```

```
end hlf;
```

### **Code for OR gate.**

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity or1 is
```

```
port (p1,q1 : in std_logic;
```

```
      carry2:out std_logic);
```

```
end or1;
```

```
architecture dataflow of or1 is
```

```
begin
```

```
carry2<=p1 or q1;
```

```
end dataflow;
```

### **Code for full adder adder using structural description**

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity fulladder is
```

```
    Port ( a : in  STD_LOGIC;
```

```
          b : in  STD_LOGIC;
```

```
          cin : in  STD_LOGIC;
```

```
          sum : out STD_LOGIC;
```

```
          carry : out STD_LOGIC);
```

```
end fulladder;
```

```
architecture Behavioral of fulladder is
```

```
component ha is
```

```
port (p,q : in std_logic;
```

```
      sum1,carry1 :out std_logic);
```

```

end component;

component or1 is

port (p1,q1 : in std_logic;

      carry2:out std_logic);

end component;

signal s0,s1,s2 : std_logic;

begin

h1 :ha port map(p=> a, q =>b, sum1 =>s0, carry1 =>s1);

h2 :ha port map(p => s0, q=>cin, sum1 =>sum, carry1 =>s2);

h3 :or1 port map(p1=> s1, q1 =>s2,carry2 => carry);

end Behavioral;

```

#### **PROCEDURE :**

- a. Open Project Navigator, Open New Project.
- b. Add proper project constraints with suitable package and language (VHDL).
- c. Write the VHDL code for given circuit.
- d. Check the syntax and synthesize the code.
- e. Observe the Timing and fitter report.
- f. Select behavioral mode and again add project constraints for Test bench Waveforms.
- g. Apply suitable stimulated inputs to input variables.
- h. Run the test bench waveforms and observe the waveforms.

**07. CONCLUSION :** The program for full adder is successfully executed and the truth table of full adder is verified.

## **Experiment-5**

**01. AIM:** To design D-Flip-Flop Using Behavioral Description

**02. FACILITIES/ ENVIORNMENT:**

**Hardware:** Computer System  
**Software:** Xilinx-ISE

**03. OBJECTIVE:**

To know about Behavioral description of VHDL programming and understanding the working of D- flip-flop.

#### 04. THEORY:

An edge triggered flip-flop changes states either at the positive edge (rising edge) or at the negative edge (falling edge) of the clock pulse on the control input. The three basics types of flip-flops are-

- 1)SR flip-flop
- 2)JK flip-flop
- 3)D- flip flop

Operation:

As long as the clock input is low, changes at the D input make no difference to the outputs. The truth table in Fig. 5.3.1 shows this as a 'don't care' state (X). The basic D Type flip-flop shown in Fig. 5.3.1 is called a level triggered D Type flip-flop because whether the D input is active or not depends on the logic level of the clock input.

Provided that the CK input is high (at logic 1), then whichever logic state is at D will appear at output Q and (unlike the SR flip-flops) Q is always the inverse of Q).

In Fig. 5.3.1, if D = 1, then S must be 1 and R must be 0, therefore Q is SET to 1.

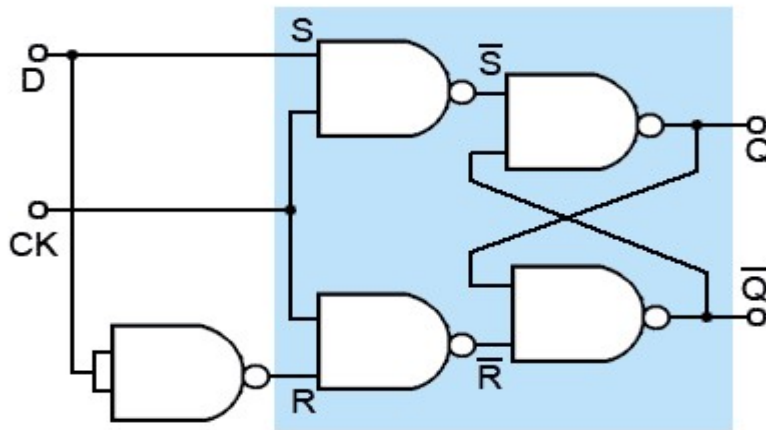
Alternatively,

If D = 0 then R must be 1 and S must be 0, causing Q to be reset to 0.

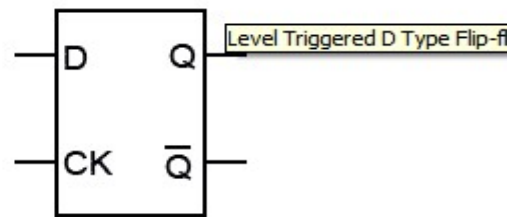
#### 05. TRUTH- TABLE:

INPUT		OUTPUT	
clk	D	Q	Qbar
0	x	x	X
1	0	0	1
1	1	1	0

#### 06.CIRCUIT DIAGRAM



Inputs		Outputs	
CK	D	Q	$\bar{Q}$
0	X	No change	
1	0	0	1
1	1	1	0



## 07. VHDL CODE:

```
Library IEEE;
USE IEEE.Std_logic_1164.all;
```

```
entity RisingEdge_DFlipFlop is
  port(
    Q : out std_logic;
    Clk : in std_logic;
    D : in std_logic
  );
```

```
end RisingEdge_DFlipFlop;
architecture Behavioral of RisingEdge_DFlipFlop is
begin
  process(Clk)
  begin
    if(rising_edge(Clk)) then
      Q <= D;
    end if;
  end process;
end Behavioral;
```

## PROCEDURE :

- a. Open Project Navigator, Open New Project.
- b. Add proper project constraints with suitable package and language (VHDL).
- c. Write the VHDL code for given circuit.
- d. Check the syntax and synthesize the code.
- e. Observe the Timing and fitter report.
- f. Select behavioral mode and again add project constraints for Test bench Waveforms.
- g. Apply suitable stimulated inputs to input variables.
- h. Run the test bench waveforms and observe the waveforms.

## 08. CONCLUSION:

The VHDL program for D-Flip-Flop is successfully executed and the truth table D-Flip-Flop is verified.

# Experiment-6

**01. AIM:** To design positive edge trigger JK flip-flop using behavioral description.

**02. FACILITIES/ ENVIORNMENT:** Hardware: Computer System  
Software: Xilinx-ISE

## 03. OBJECTIVE :

To know about Behavioral description of VHDL programming and understanding the working of JK flip-flop.

**04. THEORY:** An edge triggered flip-flop changes states either at the positive edge (rising edge) or at the negative edge (falling edge) of the clock pulse on the control input. The three basics types of flip-flops are-

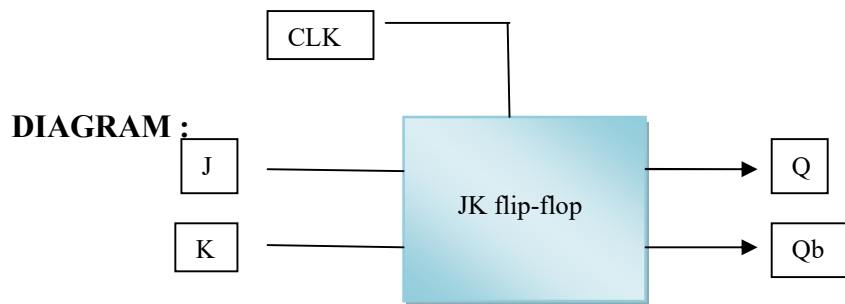
- 1)SR flip-flop
- 2)JK flip-flop
- 3)D- flip flop

The JK flip-flop are Called synchronous inputs because data on these inputs are transferred to the flip-flops output only on the triggering edge of the clock pulse. On the other hand ,the direct set (SET)and clear (CLEAR) inputs are called asynchronous inputs. As they are inputs that affect the state of the flip-flop independent of the clock. For the synchronous operation to work properly, these asynchronous inputs must both be kept at LOW.

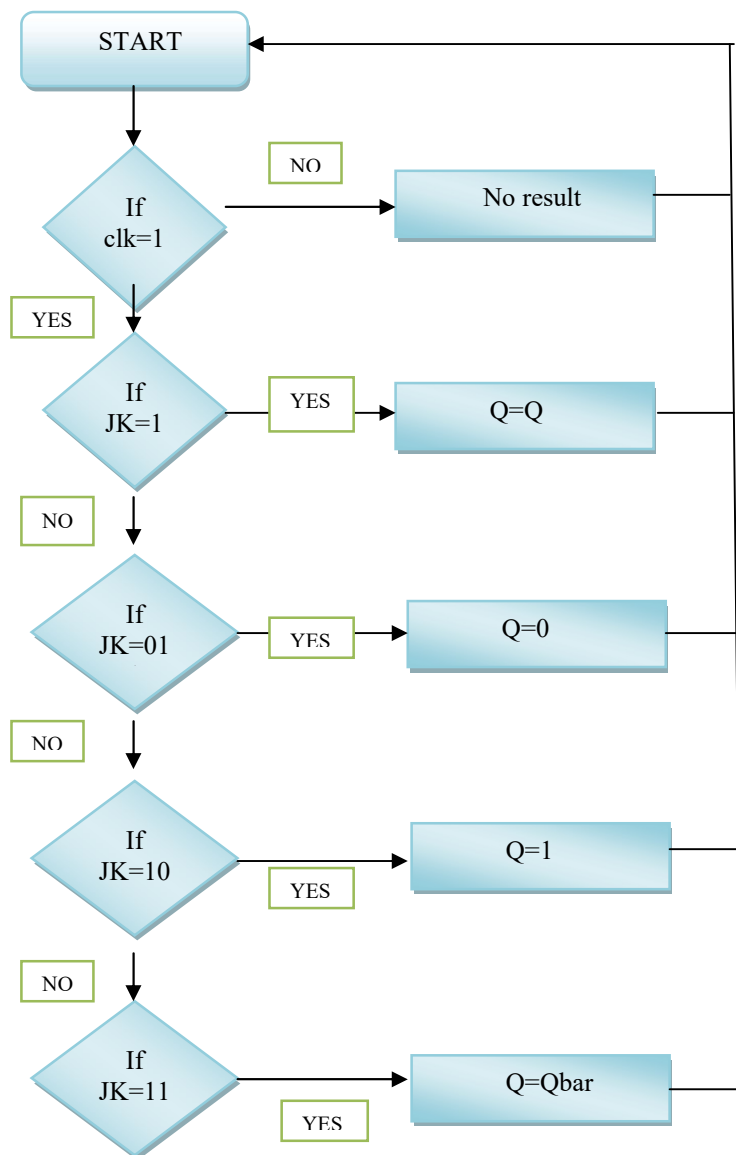
## 05. TRUTH- TABLE :

INPUTS			OUTPUTS	
CLOCK	J	K	Q	Qb
1	0	1	0	1

1	1	0	1	0
1	0	0	Q	qb
1	1	1	qb	q
0	x	x	x	x



**07. FLOW CHART :**



## 08 VHDL CODE :

```
Library ieee;

Use ieee.std_logic_1164.all;

Entity jkff is

    Port(clk: in std_logic;

        Jk: in std_logic_vector(1 down to zero);

        Q: out std_logic;

        Qb: out std_logic);

End jkff

Architecture flipflop of jkff is

Begin

    Process (clk)

        Variable temp1: std_logic;

        Variable temp2: std_logic;

    Begin

        If rising edge (clk)then

            Case jk is

                When "00" =>temp1:= temp1;

                When "01" =>temp1:= 0;

                When "10" =>temp1:= 1;

                When "11" =>temp1:= (not temp1);

                When others => null;

            End case;

            Q<= temp1;

            temp2:=(not temp1);

            Qb<= temp2;

        End if;
```



End flipflop;

**PROCEDURE :**

- a. Open Project Navigator, Open New Project.
- b. Add proper project constraints with suitable package and language (VHDL).
- c. Write the VHDL code for given circuit.
- d. Check the syntax and synthesize the code.
- e. Observe the Timing and fitter report.
- f. Select behavioral mode and again add project constraints for Test bench Waveforms.
- g. Apply suitable stimulated inputs to input variables.
- h. Run the test bench waveforms and observe the waveforms.

**09. CONCLUSION :** The program for positive edge trigger JK flip-flop is successfully executed and the truth table of JK flip-flop is verified.

## **Experiment-7**

**AIM:**To Design 4-bit counter by using behavioral description

**FACILITIES/ ENVIORNMENT:**

Computer system  
Xilinx-ISE9.1

• **OBJECTIVE :**

To know about behavioral description of VHDL programming and understanding the working of 4-bit counter

• **THEORY:**

A digital counter is a set of flip flop whose states change in response to pulse applied to the input of the counter. The flip flops are interconnected such that their combined state at any time is the binary equivalent of the total number of pulse that have occurred up to that time counter is used for.

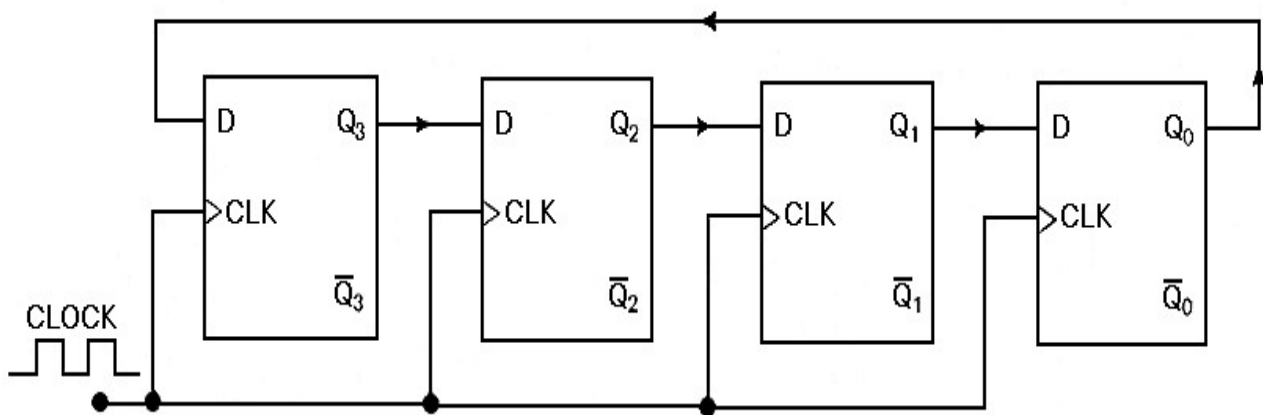
**Applications**

1. Counting pulse
2. Frequency divider
3. To perform the timing function as in digital watches

- **PROCEDURE :**

- Open File, Select New Project.
- Specify Project Name then Next.
- Click on New Source, Select VHDL Module.
- Specify File Name then Next.
- Specify input and output variables then Next then Finish.
- Type Code.
- Select “Synthesis XST” for compiling.
- Select “View RTL Schematic” for block diagram.
- Select “Behavioral Simulation” from drop-down list.
- Create New Source, click on “Test Bench Waveform”.
- Specify waveform name, then next and then finish.
- Select Combinational clock and GSR, then Finish.
- Select “Process” then simulate Behavioral Model.

- **DIAGRAM :**



**08. VHDL CODE :**

-----

*Library IEEE;*

*use IEEE.STD\_LOGIC\_1164.ALL;*

*use IEEE.STD\_LOGIC\_ARITH.ALL;*

*use IEEE.STD\_LOGIC\_UNSIGNED.ALL;*

*---- Uncomment the following library declaration if instantiating*

*---- any Xilinx primitives in this code.*

*--library UNISIM;*

*--use UNISIM.VComponents.all;*

*entity Count is*

*Port ( Clk : in STD\_LOGIC;*

*Clr : in STD\_LOGIC;*

*Q : out STD\_LOGIC\_VECTOR (3 downto 0));*

*end Count;*

*architecture Behavioral of Count is*

*begin*

*process(clk)*

*variable temp : std\_logic\_vector(3 downto 0);*

*begin*

*if(clk='1') then*

*if(clr = '0') then*

*case temp is*

*when "0000" => temp := "0001";*

*when "0001" => temp := "0010";*

*when "0010" => temp := "0011";*

*when "0011" => temp := "0100";*

*when "0100" => temp := "0101";*

*when "0101" => temp := "0110";*

*when "0110" => temp := "0111";*

*when "0111" => temp := "1000";*

*when "1000" => temp := "1001";*

```

when "1001" => temp := "1010";

when "1010" => temp := "1011";

when "1011" => temp := "1100";

when "1100" => temp := "1101";

when "1101" => temp := "1110";

when "1110" => temp := "1111";

when others => temp := "0000";

end case;

end if;

end if;

Q <= temp;

end process;

end Behavioral;

```

## **01. PROCEDURE :**

- a. Open Project Navigator, Open New Project.
- b. Add proper project constraints with suitable package and language (VHDL).
- c. Write the VHDL code for given circuit.
- d. Check the syntax and synthesize the code.
- e. Observe the Timing and fitter report.
- f. Select behavioral mode and again add project constraints for Test bench Waveforms.
- g. Apply suitable stimulated inputs to input variables.
- h. Run the test bench waveforms and observe the waveforms.

## **11. CONCLUSION :**

Thus in this way we performed the 4-bit counter successfully.

# Experiment-8

- **AIM:**

To design 4-bit Even Parity Generator Using Structural Description

- **FACILITIES/ ENVIRONMENT:**

Computer system  
Xilinx-ISE9.1

- **OBJECTIVE :**

To know about Structural Description of VHDL programming and understanding the working of 4-bit Even Parity Generator.

- **THEORY:**

**STRUCTURAL DESCRIPTION:**

It is this top-level entity that has a **structural style description**. In **VHDL**, a component is actually a placeholder for a design entity. A **structural** design that uses components simply specifies the interconnection of the components..

**Parity Generator:**

A **Parity Generator** is a Combinational Logic Circuit that Generates the **Parity** bit in the Transmitter. A **Parity** bit is used for the Purpose of Detecting Errors during Transmissions of binary Information. • It is an Extra bit Included with a binary Message to Make the Number of 1's either Odd or Even

**Even Parity Generator:**

The sum of the data bits and parity bits can be even or odd . In even parity, the added parity bit will make the total number of 1s an even amount whereas in odd parity the added parity bit will make the total number of 1s odd amount.

- **Truth Table**

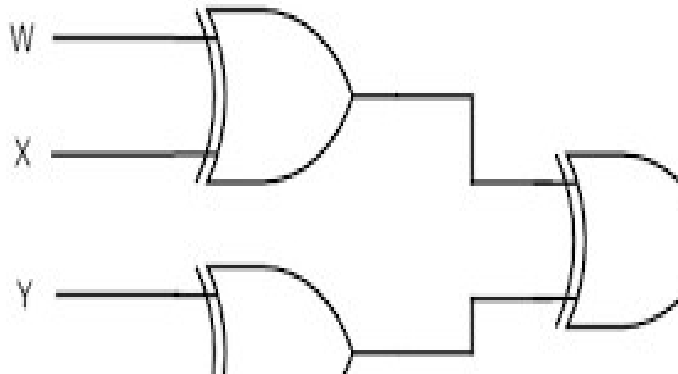
**Table 1**

$D_3$	$D_2$	$D_1$	$D_0$	Even-parity P	Odd-parity P
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	0

- **PROCEDURE :**

- Open File, Select New Project.
- Specify Project Name then Next.
- Click on New Source, Select VHDL Module.
- Specify File Name then Next.
- Specify input and output variables then Next then Finish.
- Type Code.
- Select “Synthesis XST” for compiling.
- Select “View RTL Schematic” for block diagram.
- Select “Behavioral Simulation” from drop-down list.
- Create New Source, click on “Test Bench Waveform”.
- Specify waveform name, then next and then finish.
- Select Combinational clock and GSR, then Finish.
- Select “Process” then simulate Behavioral Model.

- **CIRCUIT DIAGRAM :**



## 08. VHDL CODE :

### Program of 2 input xor gate

```
library IEEE;
use IEEE.std_logic_1164.all;
entity kanhe_xor is
port ( a,b : in std_logic;
      y: out std_logic);
end kanhe_xor;
architecture xor_kanhe of kanhe_xor is
begin
  Y <= a xor b ;
end xor_kanhe;end mux_2;
```

```

library IEEE;
use IEEE.std_logic_1164.all;
entity epp is
port ( w,x,y,p : in std_logic;
      e: out std_logic);
architecture ep of epp is

component kanhe_xor is
port ( a,b : in std_logic;

      y: out std_logic);
end component;

signal s1, s2 : std_logic;

begin

xor1 : kanhe_xor port map (a=>w , b=>x, y =>s1);

xor2 : kanhe_xor port map (a=>y , b=>p, y =>s2);

xor3 : kanhe_xor port map (a=>s1 , b=>s2, y =>e);

```

## 12. CONCLUSION :

Hence the truth table is verified and practical design 4-bit Even Parity Generator Using Structural Description performed successfully.