Experiment-8

Aim: To design a 4 bit Synchronous up/down counter.
- 4 bit Asynchronous up/down counter
- Asynchronous decade counter using behavioral style of modelling

Software Used: ModelSim

Theory: A special type of sequential circuit used to count the pulse is known as counter or a collection of flip flops where the clock signal is applied is known as counters. The two types of counters are:
- Synchronous Counters
- Asynchronous Counters

Asynchronous counters are those whose output is free from clock signal. Because the flipflops in asynchronous counters are supplied with different clock signal. There may be delay in producing output.

* **4 bit Asynchronous Counter:** It is shown in the figure. It is capable of counting numbers from 0 to 15. The clocks input of all flipflop are cascaded & the D input of each FF is connected to a state output of the FlipFlop. let Input be 0000. When rising edge of clock pulse is defined to FF0 the output Q_0 will change to logic 1 and the next next clock pulse will change Q_0 output to logic 0. This means the output state of clock pulse toggles becomes cycle. Similarly for FF1, Q_1 high and in next pulse Q_0 becomes high again. For both Q_0 & Q_1 are high if we apply forth clock pulse it will make Q_0 & Q_1 low & output 2, will be shown.

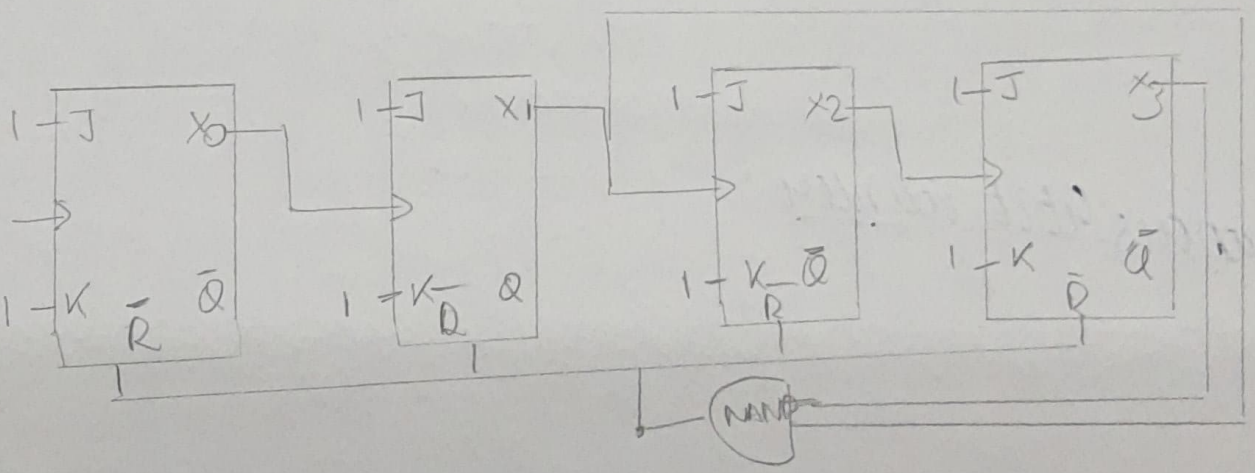
* Asynchronous Counter :- A 4 bit Asynchronous counter is shown. It will count numbers from 15 to 0 down to 0. The clock input of all flipflops are cascaded & D input of each flipflop is connected to 1. That means the flip flop will toggle at each activities edge of clock signal. The clock input of all flipflops is connected to 1. That means the flipflop will toggle at each active edge of clock signal. The clock input is connected to first flipflop. The other flip flop in counter recieve the clock signal input from Q output of previous flip flop rather than 'Q' output.

* Synchronous Counter : In this clock input across all the flip flops use the source & create the same clock signal at the same time.

* Synchronous 4 bit up counter : It start to count from 0 (0000 in binary) & increment or count upwards to 15 (1111 in binary) & then start new counting cycle by getting reset. There is no propogation delay in synchronous counter first because all flipflops is in parallel clock source & the clock triggers all counter at the same time. The external clock is directly provided to all JK FF at same time. The first FFA which is least significant bit is connected to logic 1. Due to this connection HIGH logic across the logic 1 signal, change the stage of first FF on every clock pulse. Next stage, the second flipflop FFB, is connected across the output of first FF. For third & fourth two seperate AND gate provide the necessary logic across them.

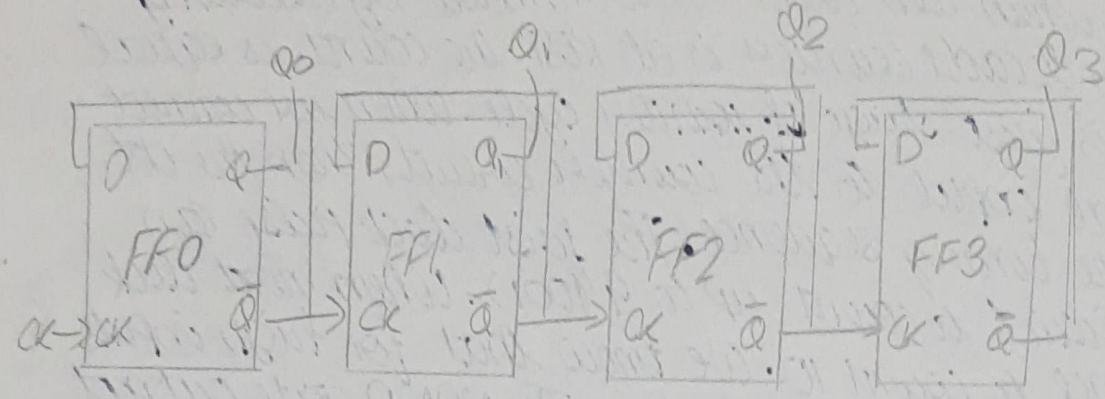
* Synchronous Down counter :- Slight changes in AND gate and using inverted output from JK flipflop can create a synchronous down counter. A 4 bit synchronous down counter start to count from 15 & decrement or count downward to 0 after that it will start a new counting cycle by getting reset.

* Decade Counter :- Same as asynchronous counter. A decade counter which can count 0 to 9 made by cascading flip-flop. When decade counter is at REST. The count is equal to 0000. This is first stage of counter cycle. When we connect a clock signal input to the counter circuit, then the ckt will count the binary sequence. The first clock pulse can make the circuit to count up to 9 (1001). When next clock pulse advance to count 10. We know that for high inputs the NAND gate output will be 1010. The NAND gate output is connected to clear input, so it reset all the FF in decade counter. This means the pulse after count 9 will again start count from 0.

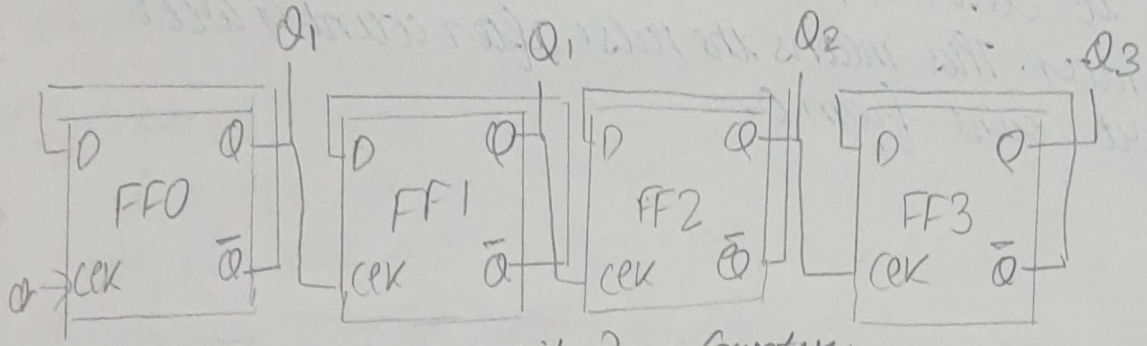


Decade Counter.

Asynchronous 4bit Counter

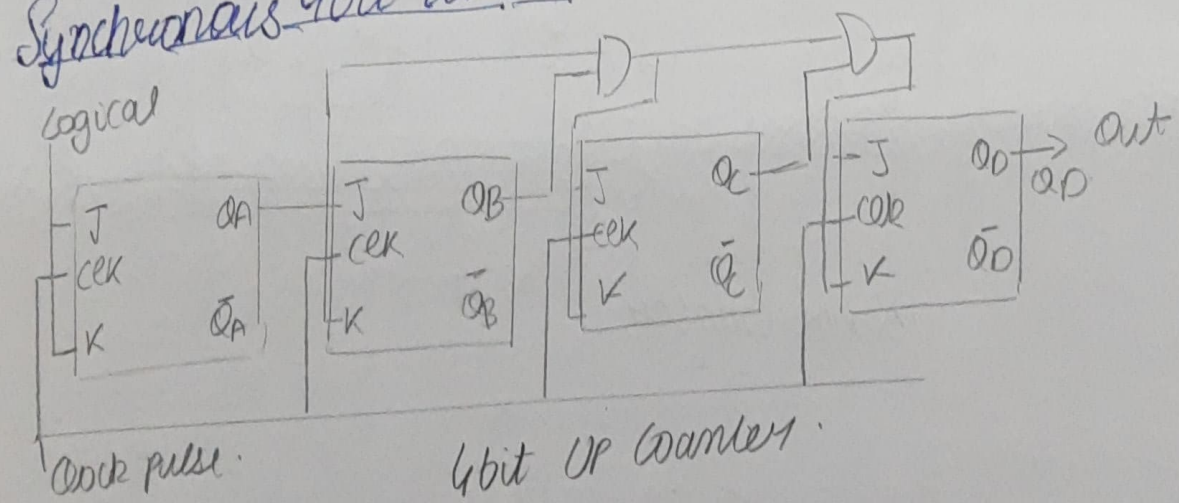


4bit UP Counter

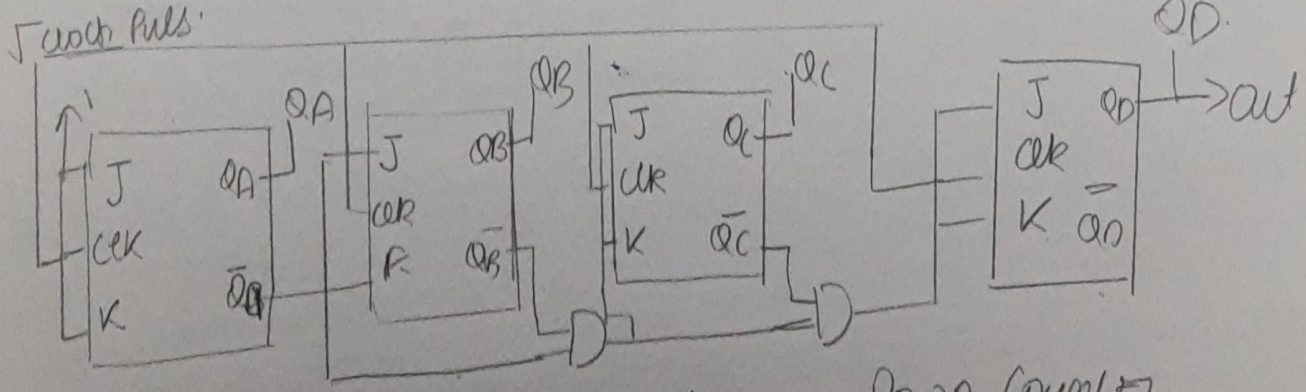


4bit Down Counter

Synchronous 4bit Counter:



4bit UP Counter.



4 Bit Synchronous Down Counter

Experiment-8

Code:

-- 4 bit synchronous up counter

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

entity syncupcounter is
   Port ( clk,reset : in  STD_LOGIC;
          count : inout  STD_LOGIC_VECTOR (3 downto 0));
end syncupcounter;

architecture bin of syncupcounter is

begin
process (clk,reset)
begin

if (reset = '1')then
count <= "0000";
elsif(rising_edge(clk))then
count <= count+1;

end if;
end process;
end bin;
```

4 bit synchronous down counter

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

entity syncdowncounter is
   Port ( clk,reset : in  STD_LOGIC;
          count : inout  STD_LOGIC_VECTOR (3 downto 0));
end syncdowncounter;

architecture bin of syncdowncounter is
```

```
begin
process (clk,reset)
begin

if (reset = '1')then
count <= "1111";
elsif(rising_edge(clk))then
count <= count-1;

end if;
end process;
end bin;
```

4 Bit Asynchronous up counter.

```
library ieee;
use ieee.std_logic_1164.all;
entity asyn_up is
  port( clk : in bit;
        q: inout bit_vector(3 downto 0));
end asyn_up;
 architecture asyn_ar of asyn_up is
 begin
  process(clk)
  begin
    if (clk'event and clk='0') then
        q(0)<= not q(0);
    end if;
end process;
process(q(0))
begin
    if (q(0)'event and q(0)='0') then
         q(1)<= not q(1);
    end if;
end process;
process(q(1))
begin
   if (q(1)'event and q(1)='0') then
         q(2)<= not q(2);
end if;
end process;
process(q(2))
begin
    if (q(2)'event and q(2)='0') then
         q(3)<= not q(3);
```

```
end if;
 end process;
end asyn_ar;
```

4 Bit Asynchronous down counter.

```
library ieee;
use ieee.std_logic_1164.all;
entity asyn_down_counter is
  port( clk : in bit;
        q: inout bit_vector(3 downto 0));
end asyn_down_counter;
 architecture asyn_ar of asyn_down_counter is
 begin
  process(clk)
  begin
    if (clk'event and clk='0') then
        q(0)<= not q(0);
    end if;
end process;
process(q(0))
begin
    if (q(0)'event and q(0)='1') then
         q(1)<= not q(1);
    end if;
end process;
process(q(1))
begin
   if (q(1)'event and q(1)='1') then
         q(2)<= not q(2);
end if;
end process;
process(q(2))
begin
    if (q(2)'event and q(2)='1') then
         q(3)<= not q(3);
end if;
 end process;
end asyn_ar;
```

DECADE COUNTER

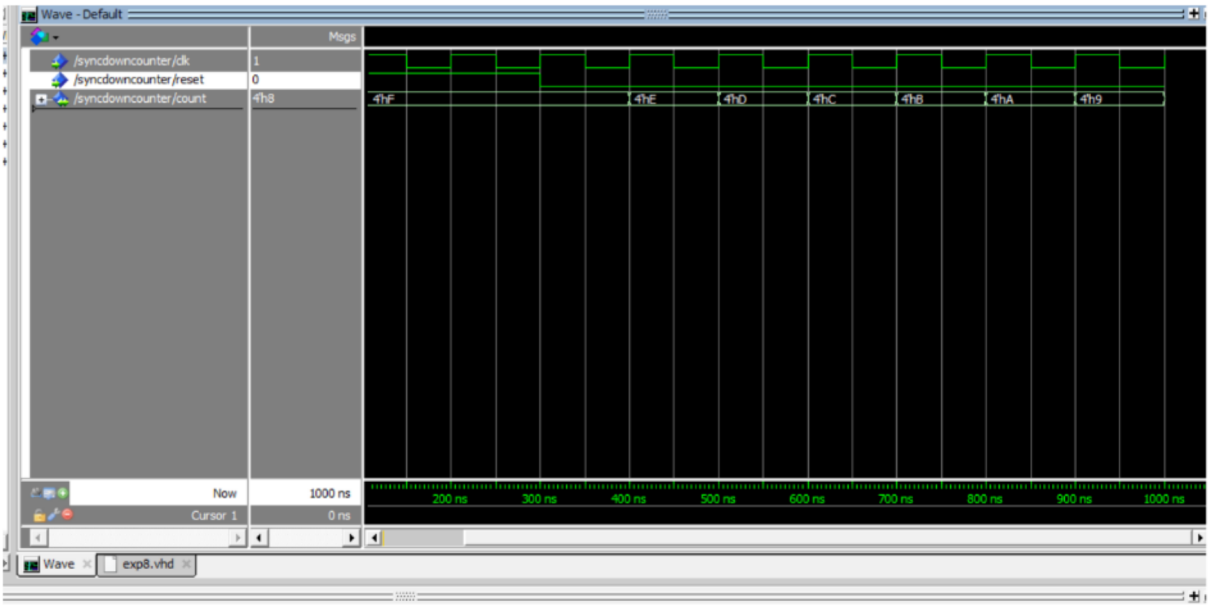
```
library ieee;
use ieee.std_logic_1164.all;
```

```
use ieee.std_logic_unsigned.all;
entity decade is
  port (clk,reset : in std_logic;
       output : out std_logic_vector(3 downto 0));
end decade;
architecture arch_decade of decade is
begin
   process(clk,reset)
      variable temp: std_logic_vector(3 downto 0);
   begin
      if reset='1' then
        temp:="0000";
        output<="0000";
     elsif clk'event and clk='1' then
         if temp<"1001" then
            temp:=temp+1;
         else
            temp:="0000";
         end if;
         output<=temp;
     end if;
end process;
end arch_decade;
```

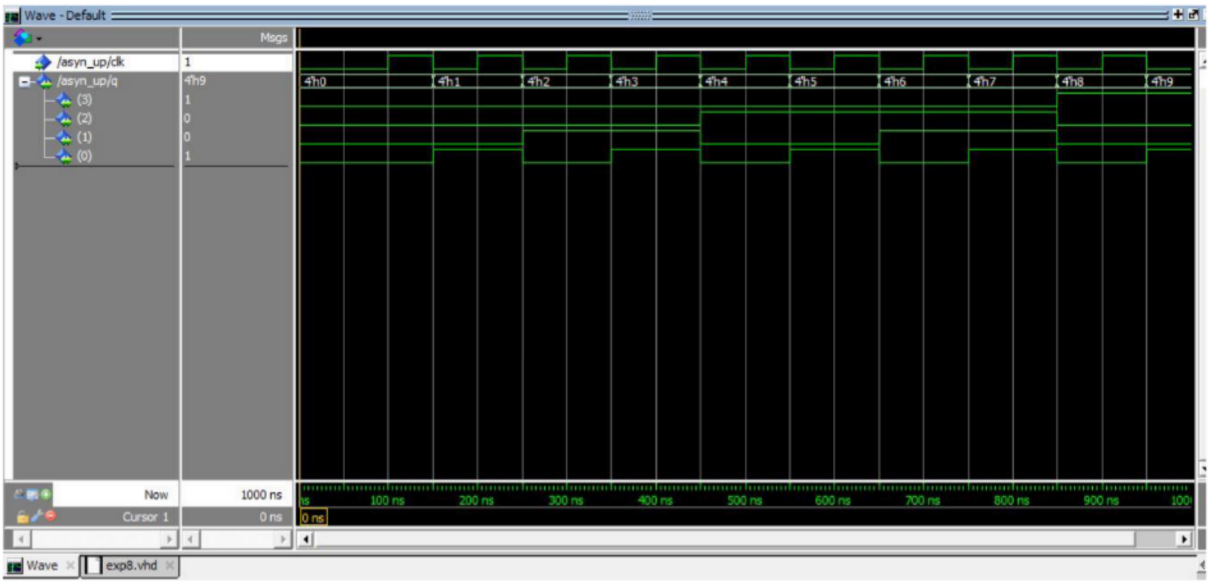
OUTPUT

Synch up counter

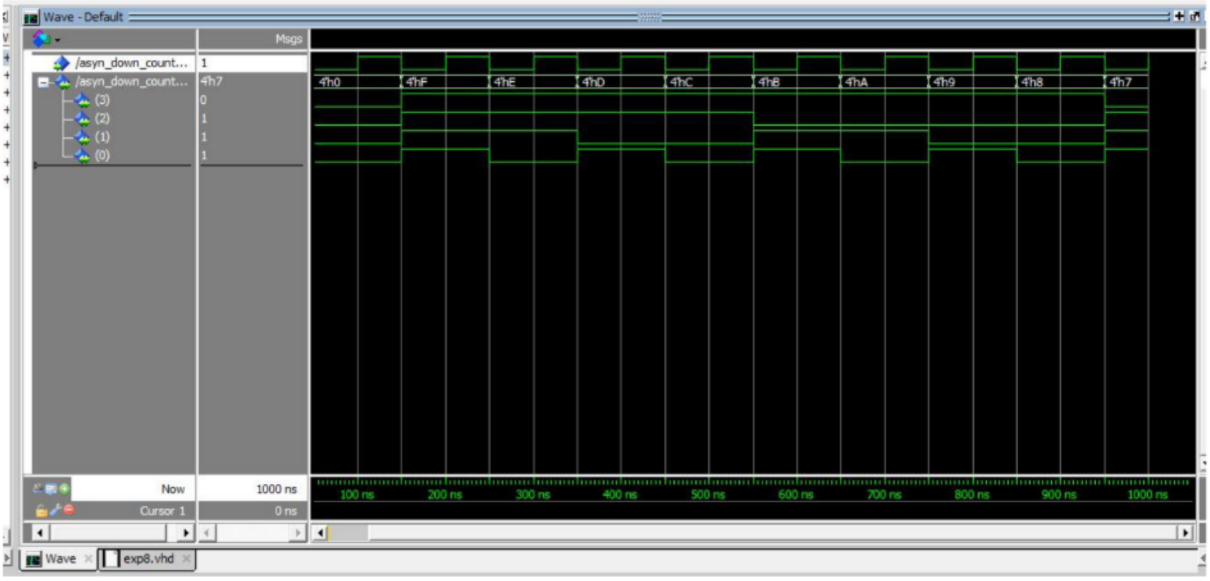
Synch down counter



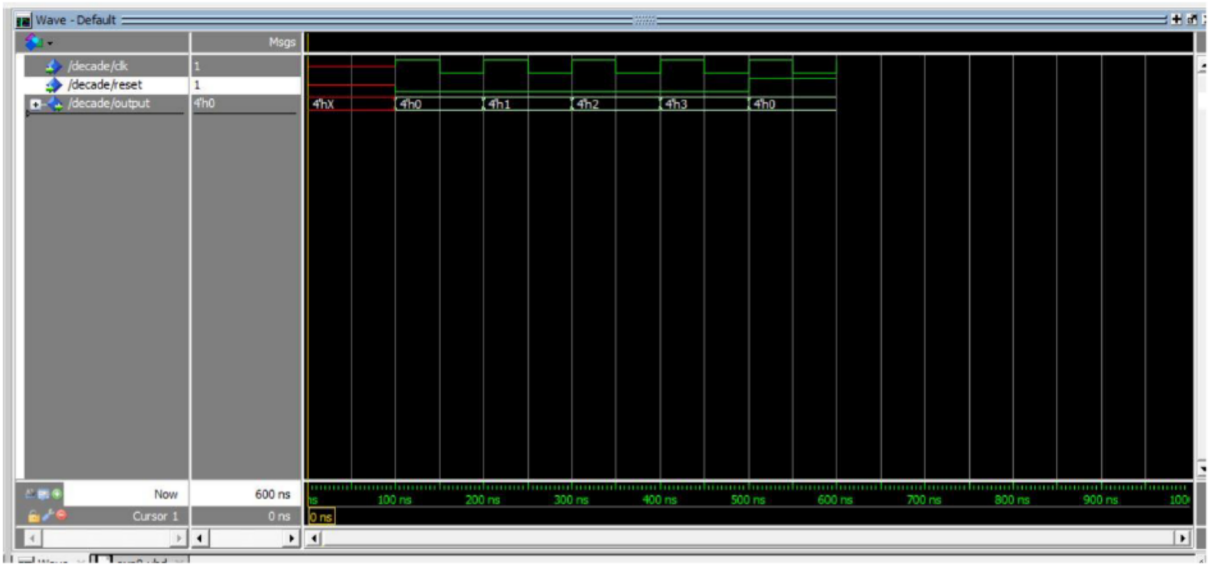
Asynch up counter



Asynch down counter



Decade counter



Result:Hence we have studied and redesigned all the counters in vhdl