# Clock Counter Module

## 1.Specification:

Count the number of positive cycles between the Sync Signal range with respect to Input Reference Signal as 0 ➠ 1 ➠ 0 using three counters.(shown in Fig.1).

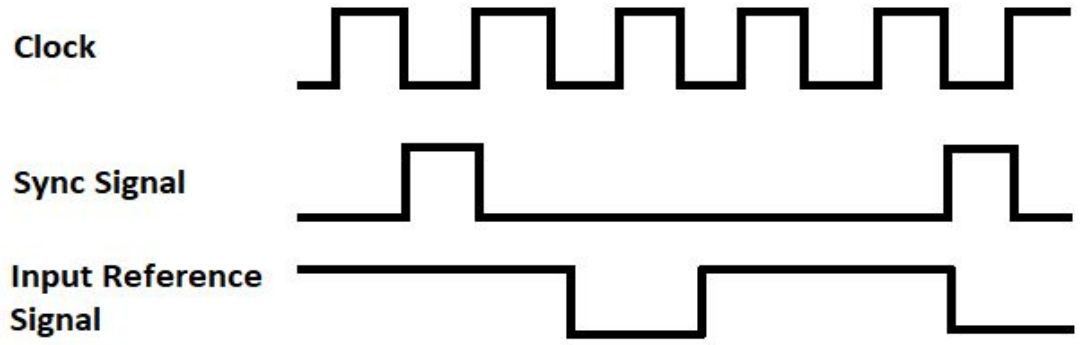


**Fig.1**.Question

## 2.Design Module:

From the given specifications, the block of Clock Counter is designed as

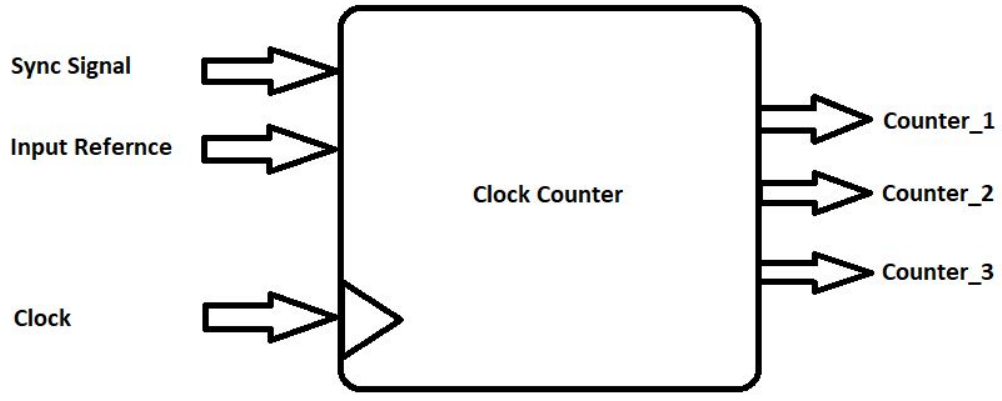


**Fig.2.1.** Block Diagram of Clock Counter

The above mentioned Fig.1 gives the detailed description of all the input and output ports in the Clock Counter Module.

## 2.1.Design Code-VHDL:

---

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
entity Module is
Port (
        clk: in std_logic;
        sync_signal: in std_logic;
        input_ref: in std_logic;
        count1_op: out std_logic_vector(3 downto 0);
        count2_op: out std_logic_vector(3 downto 0);
        count3_op: out std_logic_vector(3 downto 0)
        );
end Module;
```

---

**Fig.2.2**.Library and Entity blocks of Module

In the above mentioned Fig.2.1 and Fig.2.2,

Counter_1 ⇔ count1_op

Counter_2 ⇔ count2_op

Counter_3 ⇔ count3_op

with a maximum count of 15(resizable).

---

**architecture Behavioral of Module is**

**signal count1,count2,count3: std_logic_vector(3 downto 0):= "0000" ;**

**signal state: std_logic_vector(4 downto 0):="00000";**

**signal count_ref: std_logic;**

**constant initial: std_logic_vector(4 downto 0):="00001";**

**constant check1 : std_logic_vector(4 downto 0):="00010";**

**constant check2: std_logic_vector(4 downto 0):="00100";**

**constant check3: std_logic_vector(4 downto 0):="01000";**

**constant done: std_logic_vector(4 downto 0):="10000";**

---

**Fig.2.3**.Architecture block of Module

In the above mentioned Fig.2.3,

**Count1,Count2,Count3** ➠ To count the number of positive edge clocks with respect to input ref signal.

**State** ➠ Gives information about the current state at a particular instant of time.

**Count_ref** ➠ The previous value of Input ref signal to check whether there is any change in it

**Initial,Done** ➠ Initial and Final state of the Clock Counter

**Check1,Check2,Check3** ➠ States of all three counters

```
process(sync_signal,input_ref,clk)
begin
        if(sync_signal='1') then
          count1<=(others=>'0');
          count2<=(others=>'0');
          count3<=(others=>'0');
          state<=initial;
        else
          if(rising_edge(clk) and sync_signal='0') then
                case state is
                        when initial => if(input_ref='0') then
                                                count_ref<='0';
                                                count1<=count1+1;
                                                state<=check1;
                                        elsif(input_ref='1') then
                                                count_ref<='1';
                                                count2<=count2+1;
                                                state<=check2;
                                        else
                                                state<=initial;
                                        end if;
                        when check1 =>
                                        if(input_ref=count_ref) then
```

```
                count1<=count1+1;
                state<=check1;
            elsif(input_ref/=count_ref) then
                count_ref<= not count_ref;
                count2<=count2+1;
                state<=check2;
            else
                state<=done;
            end if;
        when check2 =>
            if(input_ref=count_ref) then
                count2<=count2+1;
                state<=check2;
            elsif(input_ref/=count_ref) then
                count_ref<= not count_ref;
                count3<=count3+1;
                state<=check3;
            else
                state<=done;
            end if;

        when check3 =>
            if(input_ref=count_ref) then
                count3<=count3+1;
```

```
                                        state<=check3;
                                    end if;
                when done =>    count1<=(others=>'0');
              count2<=(others=>'0');
              count3<=(others=>'0');
                        state<=initial;

                when others =>  state<= initial;
          end case;
 end if;
 end if;
end process;
```

---

**Fig.2.4.**Process block of Module

Process Block initiation with sensitivity list containing sync_signal,input_ref and clk.The If block with sync condition checks for sync signal with high value.If Sync signal is high,all counter values are initialised to 0 or else the counters works as usual.In the else block,the if condition checks for the rising edge of clock and low value of sync to start the counter.

Now the case statement is used to check the current state and execute the block of that particular state.

**Initial** ➠ This block checks the initial value of Input Reference Signal.Based on the result it either updates the values of count_ref,counter value and redirects the state to Check1 block(Counter 1) or updates the values of count_ref,counter value and redirects the state to Check2 block(Counter 2) or redirects the state to Initial block itself.

**Check1** ➠ This block checks the change in the previous and current value of Input reference signal.Based on the result it either updates the counter value or redirects the state to Check2 block(Counter 2) or redirects the state to Done block.

**Check2** ➠ This block checks the change in the previous and current value of Input reference signal.Based on the result it either updates the counter value or redirects the state to Check3 block(Counter 3) or redirects the state to Done block.

**Check3** ➠ This block checks the change in the previous and current value of Input reference signal.Based on the result it updates the counter value.

**Done** ➠ This block resets all the counter value to 0 and redirects the state to Initial block.The values are then copied to output ports as follows.

## 3.Testbench- SystemVerilog:

---

```
module tb_Module;

 reg clk;

 reg sync_signal;

 reg input_ref;

 wire [3:0] count1_op;

 wire [3:0] count2_op;

 wire [3:0] count3_op;

 Module   a (              .clk (clk),

                           .sync_signal (sync_signal),

                            .input_ref(input_ref),

                            .count1_op(count1_op),

                            .count2_op(count2_op),

                            .count3_op(count3_op)

                           );
```

```
always #5 clk = ~clk;
```

---

**Fig.3.1.1**.Testbench using SystemVerilog

In the above Fig.3.1.1,all the input and output ports are initialised as reg and wire datatypes and the instance of Module is initialised using explicit association.The Clock signal is made to toggle for every 5ns.

---

```
initial begin

  $monitor ("[%0tns] clk=%0b sync_signal=%0b input_ref=%b
count1_op=0x%0h count2_op=0x%0h count3_op=0x%0h", $time,
clk,sync_signal,input_ref,count1_op,count2_op,count3_op);

   clk <= 0;

   input_ref <= 0;

   sync_signal<=0;


   #10 sync_signal <= 1;

   input_ref<=1;

   #10 sync_signal <= 0;

   #40 input_ref<=0;

   #70 input_ref<=1;

   sync_signal<=1;

   #10 sync_signal<=0;

   #40 sync_signal <= 1;

   input_ref<=0;
```

```
 #10 sync_signal <= 0;
   #50 input_ref<=1;
   #40 input_ref<=0;
   #50 sync_signal<=1;
   #10 sync_signal<=0;
   #50 sync_signal <= 1;
   input_ref<=0;
   #10 sync_signal <= 0;
   #40 input_ref<=1;
   #50 sync_signal<=1;
   #10 sync_signal<=0;
   input_ref<=0;
   #40 input_ref<=1;
   #70 input_ref<=0;    #30 sync_signal<=1;
   #50  $finish;
end
```

---

**Fig.3.1.2**.Testbench using SystemVerilog

In the above Fig3.1.2, $monitor is used to check for any changes in the mentioned list.Now by providing different possible values for the input ports,we are monitoring the values of all three counters and the $finish is used to terminate the waveform after 50ns.

```
initial begin
  $dumpvars;
  $dumpfile("dump.vcd");
end
```

**Fig.3.2**.VCD file

In the above Fig3.2,.The $dumpvars and $dumpfile saves the changes that occur in the variables in the  dump.vcd file.

## 4.Waveform:

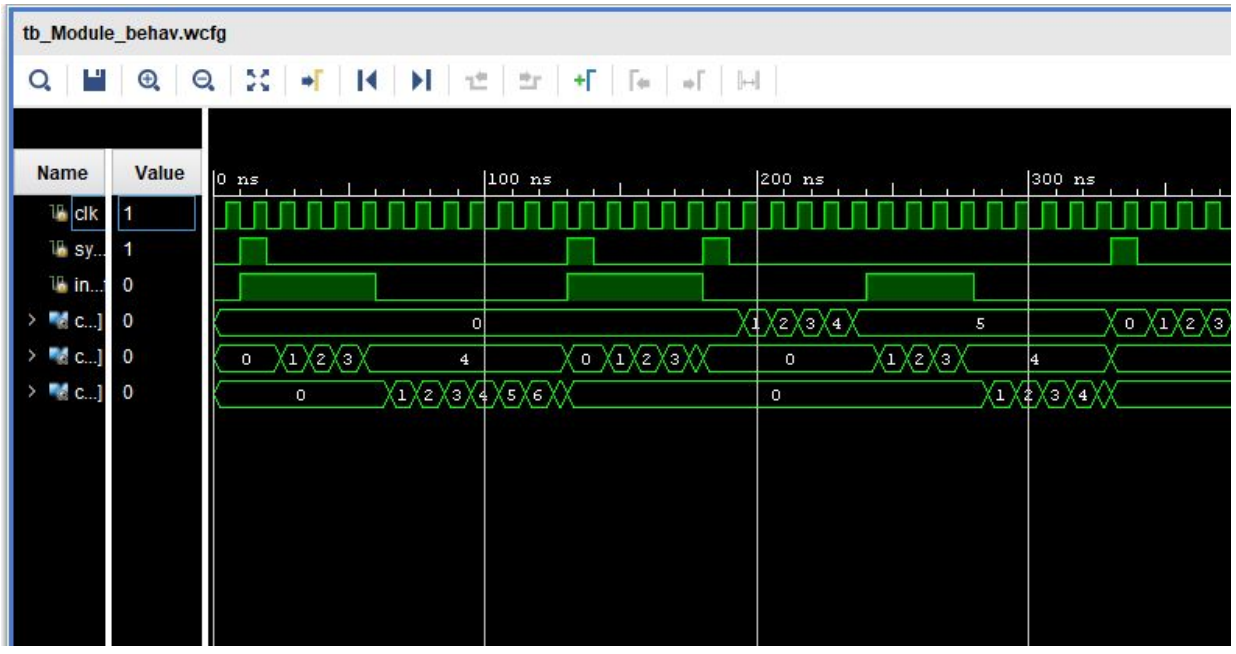


**Fig.4.1**.Waveform output

In the above Fig.4.1, for first sync signal range,the input reference signal changes as

1 ➠ 0  where the counter1 gives **0 condition** as expected and other counters works properly.In the next sync signal range, there is no input reference signal changes,so the counter2 alone counts.And then in the next range, the input reference signal changes as

0 ➠ 1 ➠ 0,so all the three counters works properly.

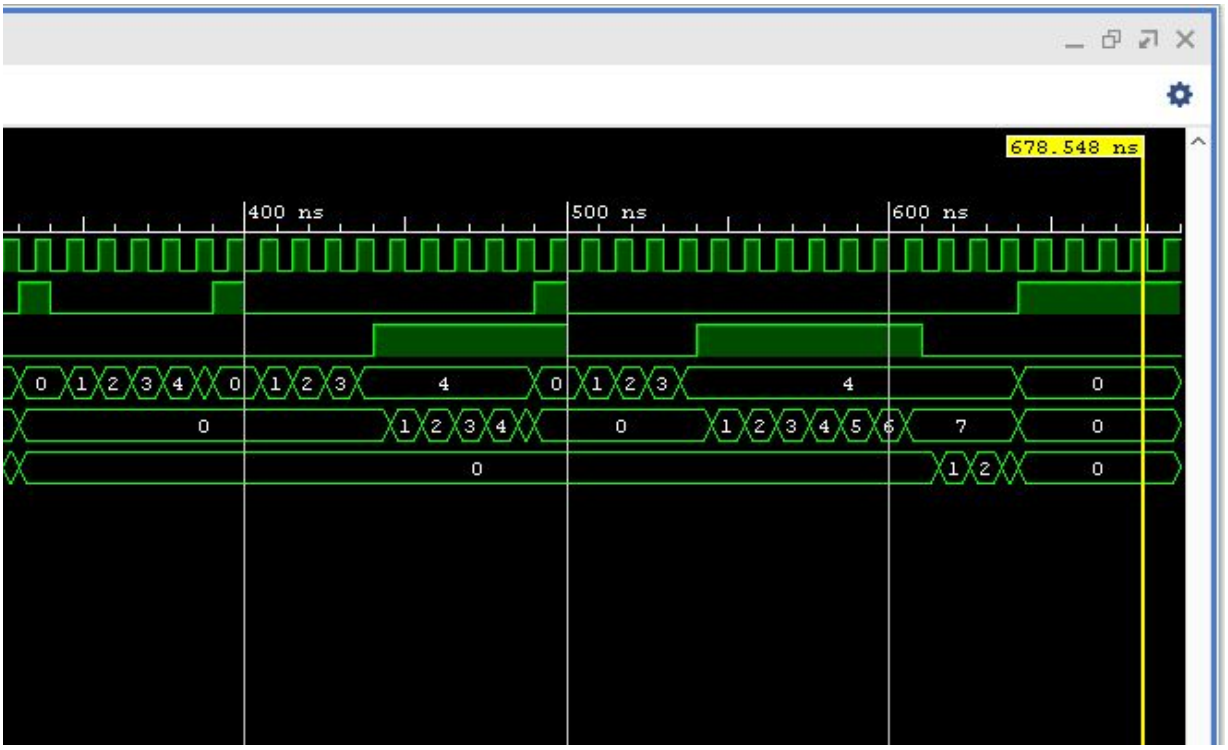


**Fig.4.2**.Waveform output

In the above Fig4.2, for first sync signal range,there is no input reference signal changes,so only the counter1 works.In the next sync signal range, the input reference signal changes as 0 ➠ 1 ,so the counter1 and counter2 alone works and not the counter3.And then in the next range, the input reference signal changes as

0 ➠ 1 ➠ 0,so all the three counters works properly.