

## Task -1

### Verilog :

/\*

Name-Shadeeb Hossain

ID: sh7492

This is a Verilog file that generates a counter Up .

It resets on the first digit of your N Number

The following were provided:

clk-1 bit: Design clock

rst-1 bit: Asynchronous active low reset

cnt-ent-1 bit: Enable for the counter

If 1 counter will be counting up or down.

0 it should be stopped at the most recent output.

cnt-6 bits: Counter output as a 6 bit unsigned vector

\*/

/\*

ChatGPT Prompt:

Hi, Can you please design a verilog file that generates a counter UP ,

resets to the first digit of your N number.

The following should be used clk- design clock, rst-1bit- Asynchronous active low reset.

cnt\_ent-2 bit to enable for the counter, If 1 counter will be counting up or down. 0 it should be stopped at its most recent output.

cnt-6 bits: Counter output as a 6 bit unsigned vector

\*/

module up\_counter (

```

input wire clk,    // Clock input
input wire rst,    // Asynchronous active low reset
input wire [1:0] cnt_ent, // 2-bit enable for the counter (01: up, 10: down, 00: stop)
output wire [5:0] cnt // 6-bit counter output
);

```

```

reg [5:0] count; // 6-bit counter register

```

```

always @(posedge clk or negedge rst) begin
    if (!rst) begin
        count <= 6'b000000; // Reset the counter to 0
    end else begin
        case (cnt_ent)
            2'b01: count <= count + 1; // Count up
            2'b10: count <= count - 1; // Count down
            default: count <= count; // Hold the current value
        endcase
    end
end

```

```

assign cnt = count;

```

```

endmodule

```

```

/*

```

In this Verilog module:

We have a 6-bit counter register called count to store the counter value.

The counter is sensitive to both the positive edge of the clock (clk) and the negative edge of the reset signal (rst).

When the reset signal is low (rst is active low), the counter is reset to 0.

Depending on the value of cnt\_ent (2-bit enable for the counter), the counter can either count up (01), count down (10), or stop (00).

The counter output is provided as a 6-bit unsigned vector through the cnt output wire.

You can instantiate this module in your larger Verilog design and connect the clock, reset, cnt\_ent, and cnt signals as needed.

\*/

## VHDL File

--Name-Shadeeb Hossain

--ID: sh7492

--This is a VHDL code that generates a counter Down .

--It resets on the last digit of your N Number

--The following were provided:

--clk-1 bit: Design clock

--rst-1 bit: Asynchronous active low reset

--cnt-ent-1 bit: Enable for the counter

--If 1 counter will be counting up or down.

--0 it should be stopped at the most recent output.

--cnt-6 bits: Counte output as a 6 bit unsigned vector

--ChatGPT Prompt:

--Write a VHDL code that performs count down, resets to the last digit of your N number.

--The following should be used clk- design clock, rst-1bit- Asynchronous active low reset.

--cnt\_ent-2 bit to enable for the counter, If 1 counter will be counting up or down.

--0 it should be stopped at its most recent output.

--cnt-6 bits: Counter output as a 6 bit unsigned vector

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity down\_counter is

```
Port (  
    clk    : in STD_LOGIC;      -- Clock input  
    rst    : in STD_LOGIC;      -- Asynchronous active low reset  
    cnt_ent : in STD_LOGIC_VECTOR(1 downto 0); -- 2-bit enable for the counter (01: down, 10: up, 00:  
stop)  
    cnt    : out STD_LOGIC_VECTOR(5 downto 0) -- 6-bit counter output  
);  
end down_counter;
```

architecture Behavioral of down\_counter is

```
    signal count : STD_LOGIC_VECTOR(5 downto 0); -- 6-bit counter signal  
  
begin  
    process (clk, rst)  
    begin  
        if rst = '0' then  
            count <= "000000"; -- Reset the counter to 0  
        elsif rising_edge(clk) then  
            case cnt_ent is  
                when "01" =>  
                    if count = "000000" then  
                        count <= "111111"; -- Reset to the last digit of N  
                    else  
                        count <= count - 1; -- Count down  
                    end if;  
                when "10" =>  
                    count <= count + 1; -- Count up  
            end case;  
        end process;
```

```
        when others =>
            null; -- Hold the current value
        end case;
    end if;
```

```
cnt <= count;
```

```
end Behavioral;
```

--We have an entity called down\_counter with the specified ports, including clk, rst, cnt\_ent, and cnt.

--The counter is implemented using a process sensitive to the rising edge of the clock (clk) and the asynchronous active low reset signal (rst).

--Depending on the value of cnt\_ent (2-bit enable for the counter), the counter can count down (01), count up (10), or stop (00).

--When counting down, if the counter reaches 0, it resets to "111111," which corresponds to the last digit of N.

--The counter output is provided as a 6-bit unsigned vector through the cnt signal.

--You can instantiate this module in your larger VHDL design and connect the clock, reset, cnt\_ent, and cnt signals as needed.

## Task-2

(a)

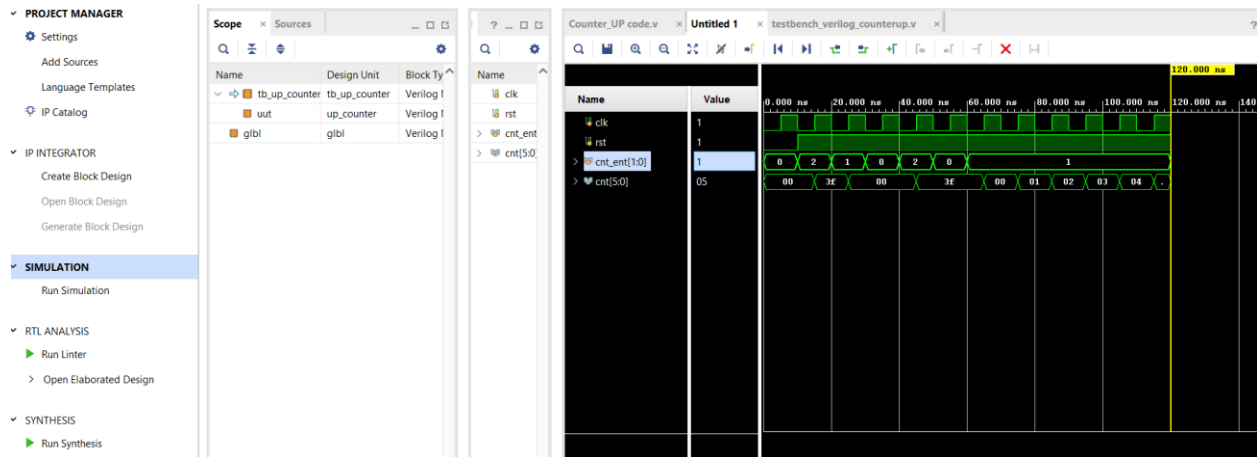


Fig. 2(a) : Functional Simulation model for Up counter (Verilog)

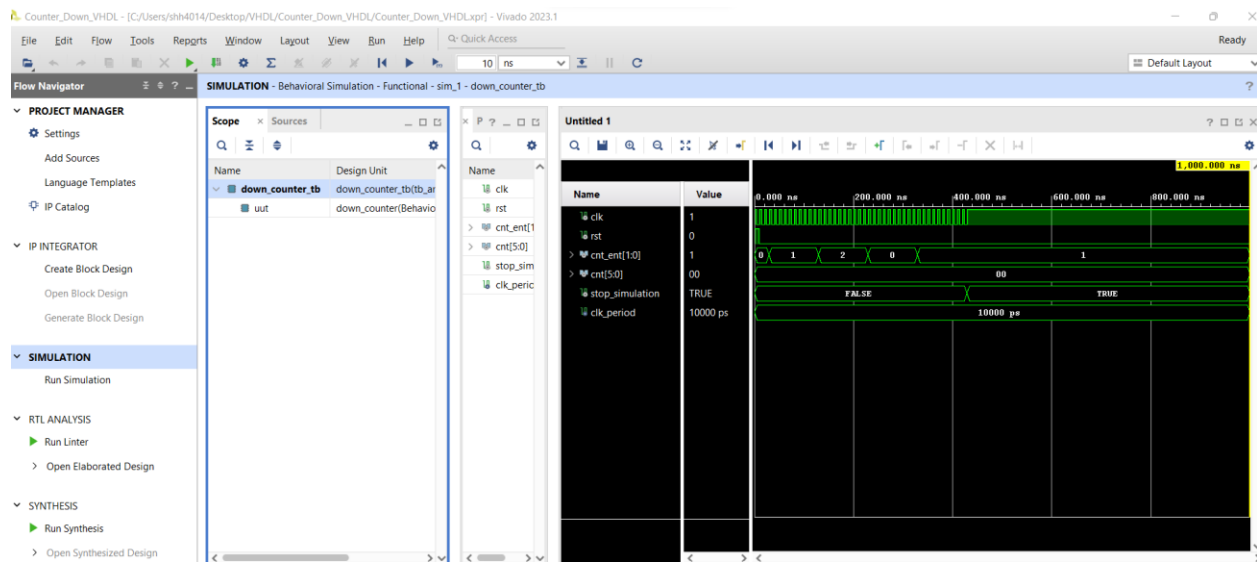
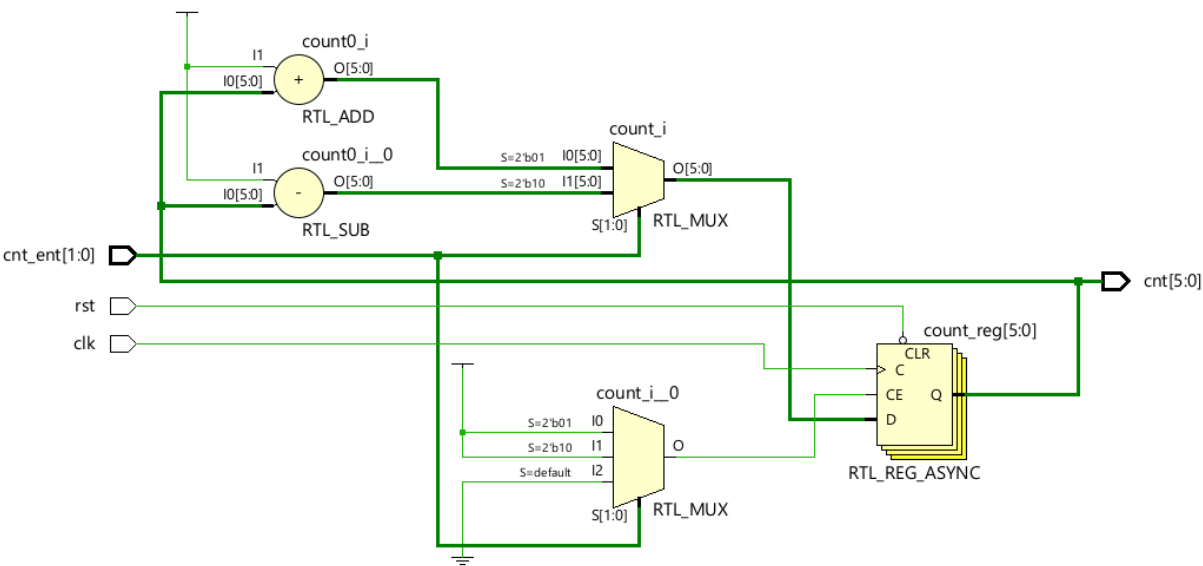


Fig.2 (b) : Functional Simulation model for Down counter (VHDL)

### Task-3



**Fig. 3 (a) : Schematic for elaborated design for Up counter (Verilog)**

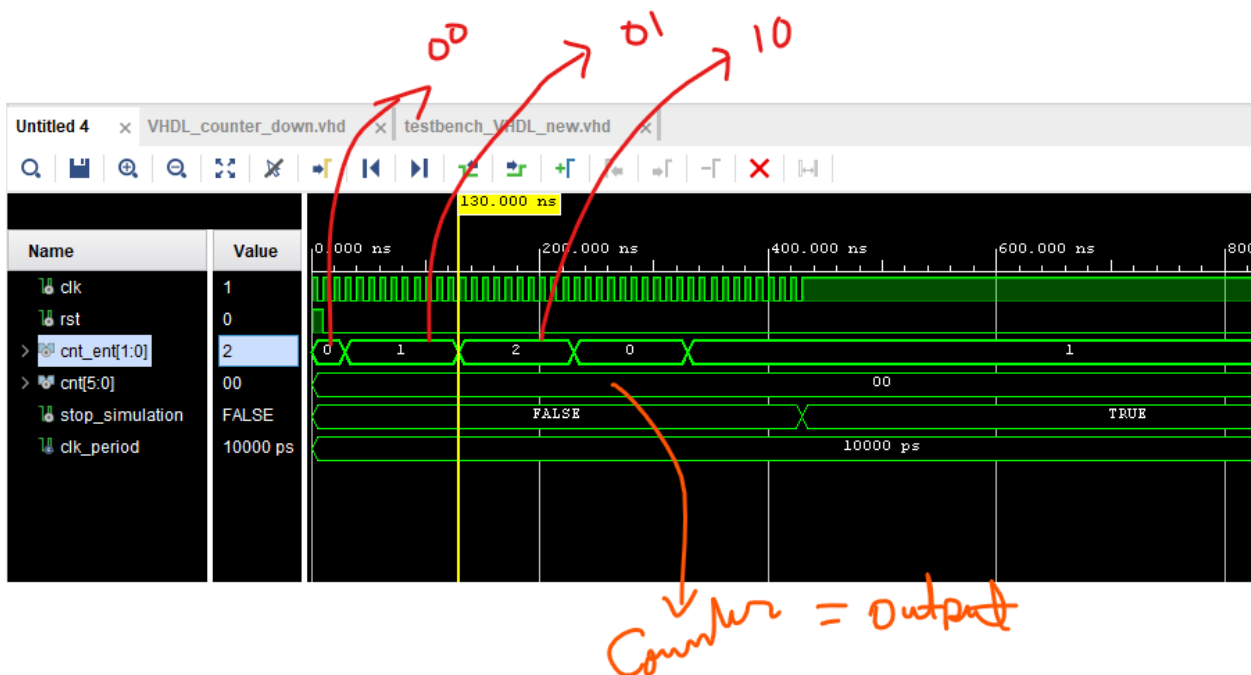




VHDL Counter

**Waveform :** At 12.15 ns, at the rising edge of the clk , it checks the status of rst pin .

Since the rst pin is set to '0', it checks the status of count ent. There are three states for count ent: 00 (0), 01 (1) and 10 (2). At 00- it is supposed to stop the count. At 01 it is suppose to count down and at 10 (2) it is suppose to count up



According to the VHDL code:

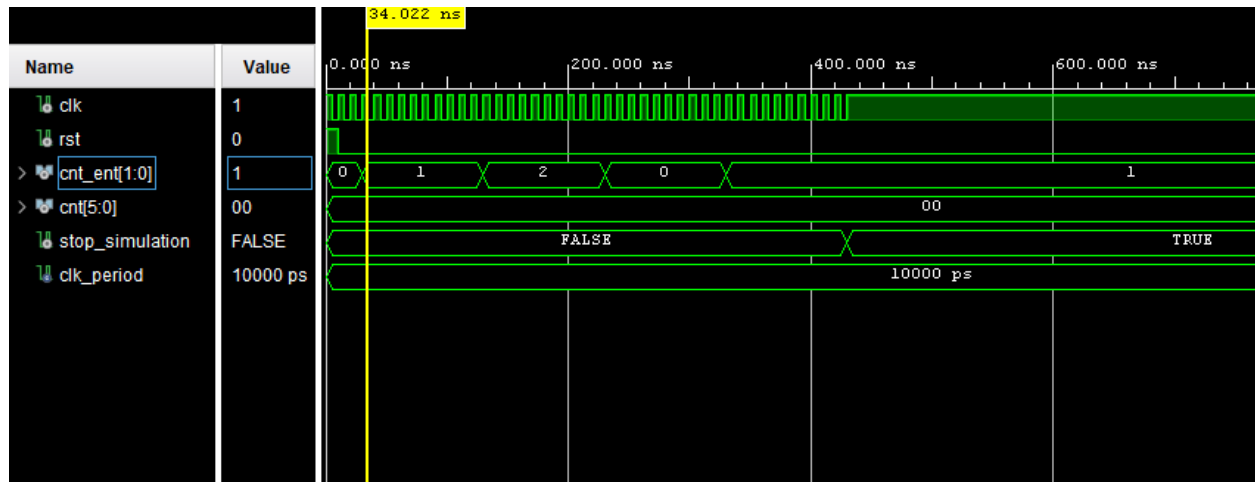
```
if rst = '0' then
    count <= "000000"; -- Reset the counter to 0
```

Hence at rising edge of clock at 12.15 ps since the rst is set to '0'. Therefore, the count output is set to '00000'.

```

elsif rising_edge(clk) then
    case cnt_ent is
        when "01" =>
            if count = "000000" then
                count <= "111111"; -- Reset to the last digit of N
            else
                count <= count - 1; -- Count down
            end if;
    end case;
end if;

```



**However, at 34.022 ns**

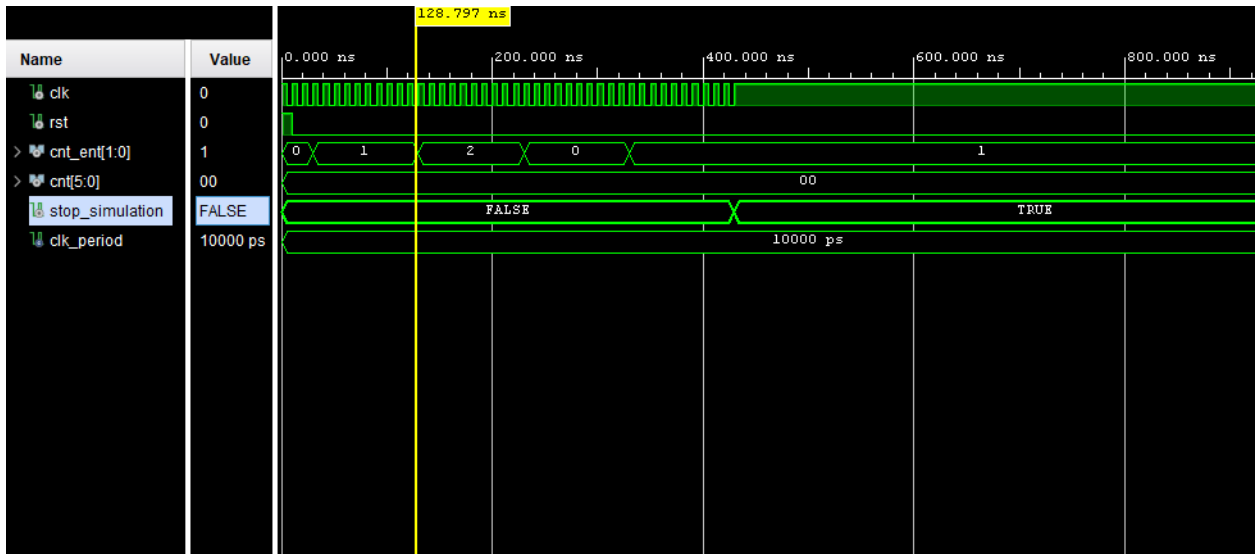
**At the rising edge of the clk, since rst remains at '0' therefore cnt is '00000'.**

However **if** it should check that **rst is '1'** and if **cnt\_ent is 01**. Then according to the code, the cnt should have been 11111. However, it remains at '00000'.

```

;    when "10" =>
        count <= count + 1; -- Count up
    when others =>
        null; -- Hold the current value
    end case;
end if;

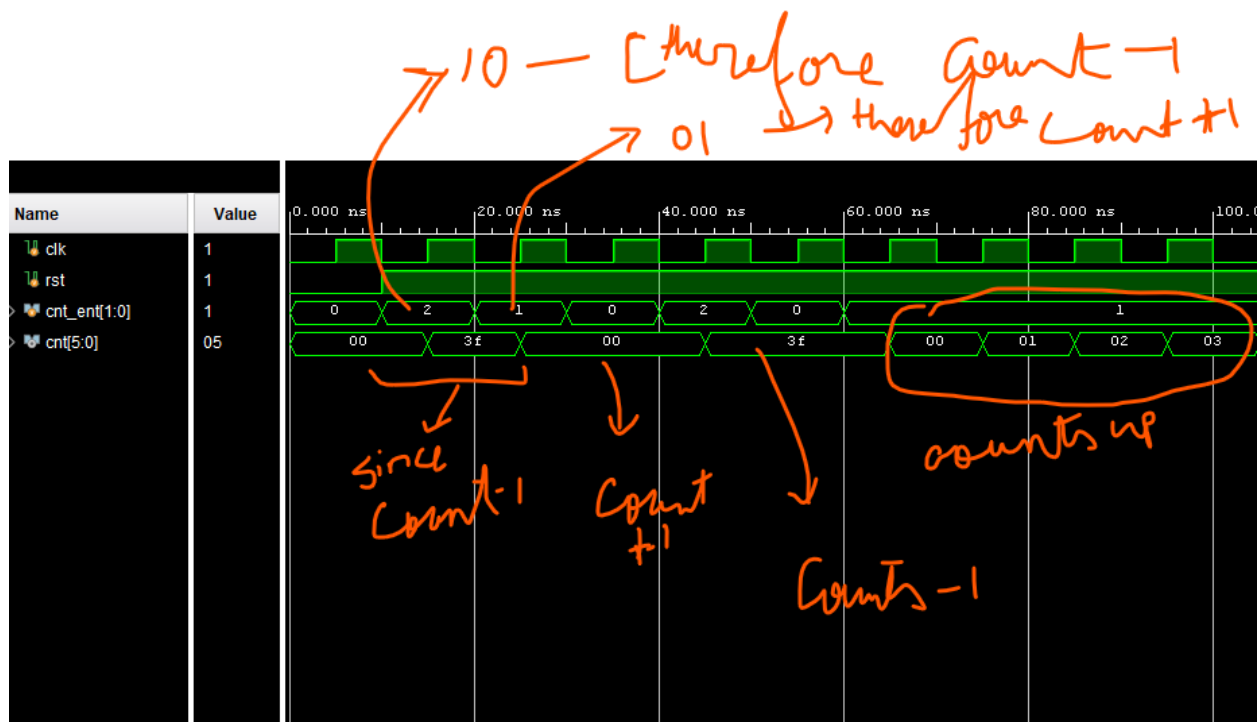
```



Similarly, at 128.797 ns, even if rst is '0' and since cnt\_ent is 10 (2).

According to the code, the count (output) should have been '00001'. However, it still remains at '00000'. This seems like an error on the testbench and ChatGPT and needs to be configured accordingly.

## Verilog



At 10 ns(at rising clock edge), when the cnt\_ent becomes 2 ( as in binary =10), the counter is suppose to subtract by count-1.

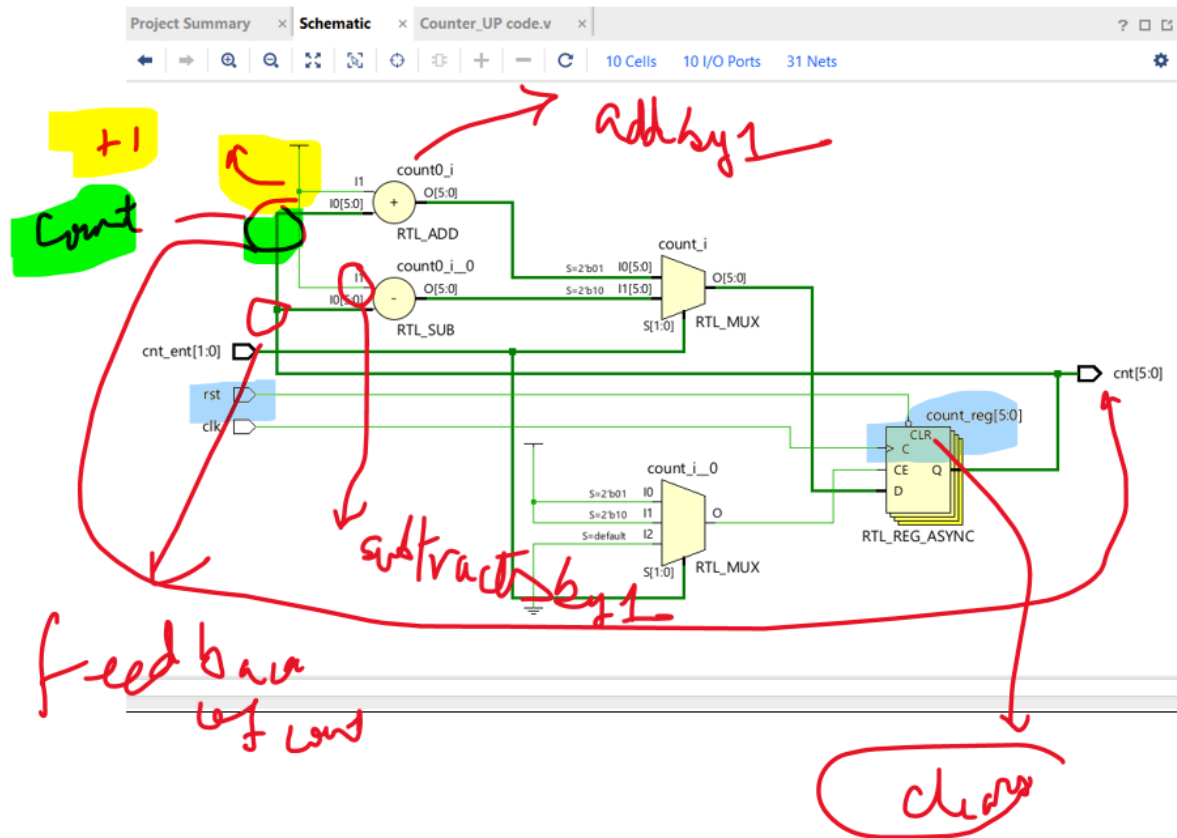
However since the cnt is already at '00000'. Therefore it subtracts from 00000 and therefore in **hexadecimal the output is 3f ( equivalent to decimal 63).**

**Similarly at 25ns**(at rising clock edge), since the cnt\_ent is 1 ( as in binary 01), the counter is suppose to add count+1. Hence , it returns to '00000'.

Similarly , at 45 ns(at rising clock edge), when the cnt\_ent becomes 2 ( as in binary =10), the counter is suppose to subtract by count-1.

**However, from 65ns,** the cnt\_ent remains at 01 and hence it just counts up. Therefore '00001'(equivalent to 1), 00010(equivalent to 2) and it keeps on adding +1 as long as cnt\_ent is at 01.

## Verilog- Block Diagram



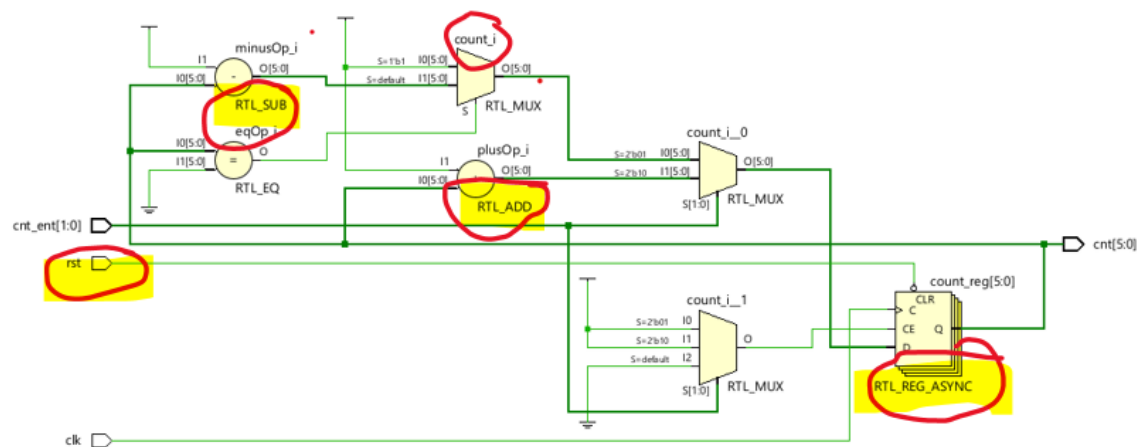
RTL\_add is the Register Transfer level that performs : **count = count + 1.**

Similarly, RTL\_SUB is the Register Transfer level that performs: **count = count - 1.**

**count ent** determines the status of the output **of count i.** If the select pin (S) is 01, it would take the output from RTL\_add. However, if the select pin (S=10), then the output will be taken from RTL\_sub.

**RST** is the reset pin and it can clear all the data using the **RTL REG ASYNC.**

## VHDL-Block Diagram



The block diagram for the VHDL and Verilog is almost the same.

In the VHDL, there is an additional RTL equal option which was not available in the Verilog file. The RTL EQ takes one of the input from output(cnt) and the other is grounded and performs arithmetic addition and returns the previous count output.

RTL\_add is the Register Transfer level that performs :  $count = count + 1$ .

Similarly, RTL\_SUB is the Register Transfer level that performs:  $count = count - 1$ .

count ent determines the status of the output of count i. If the select pin (S) is 01, it would take the output from RTL\_add. However, if the select pin (S=10), then the output will be taken from RTL\_sub.



**RST** is the reset pin and it can clear all the data using the  
**RTL REG ASYNC.**