WESTERN MICHIGAN UNIVERSITY
ECE 5510 APPLICATION SPECIFIC INTEGRATED CIRCUIT DESIGN
Homework Assignment #3

# Homework Assignment #3

**Name:** Erivelton Gualter dos Santos

**Win Numer:** 143101929

Dr. Janos L. Grantner

**WESTERN MICHIGAN UNIVERSITY**
**ECE 5510 APPLICATION SPECIFIC INTEGRATED CIRCUIT DESIGN**
**Homework Assignment #3**

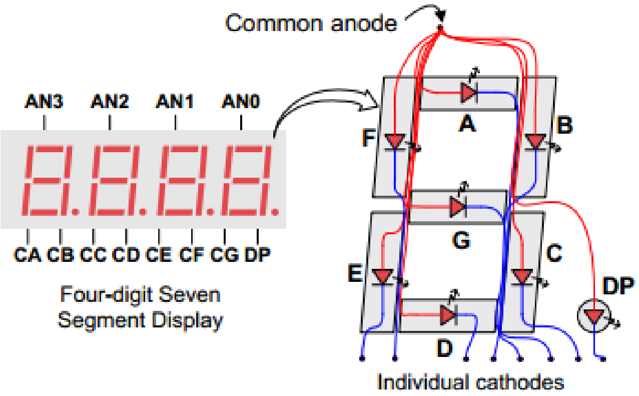
**Task 1:**

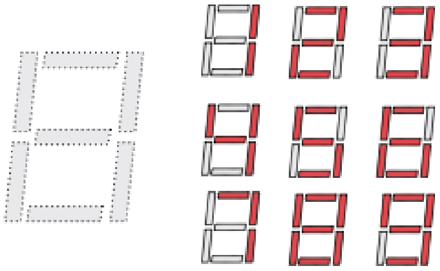
Develop a behavioral VHDL program for a **synchronous (clocked) 2-digit, modulo-32 binary-coded-decimal (BCD) up counter.** The counter should **count from 0 up to 31** and then continue from 0 again. The counter should have an **active high CLR** signal to reset its contents to 0 at any time. The counter's state should be displayed on the **7-segment display module** of the Nexys 3 Board. Both the CLK and the CLR signals should be mapped to pushbuttons, respectively. You should pick a push button for the CLK signal that is connected to a **GCLK pin** of your FPGA. You should implement this counter using the FPGA, the 7-segment display and push-buttons on your Nexys 3 Board. Note: you should use the 100MHz clock available on the Nexys 3 Board for creating the required signal timings for the 7-Segment Display module. Introduce a **suitable delay** to make the CLK signal **bounce free, if needed**.

## Task

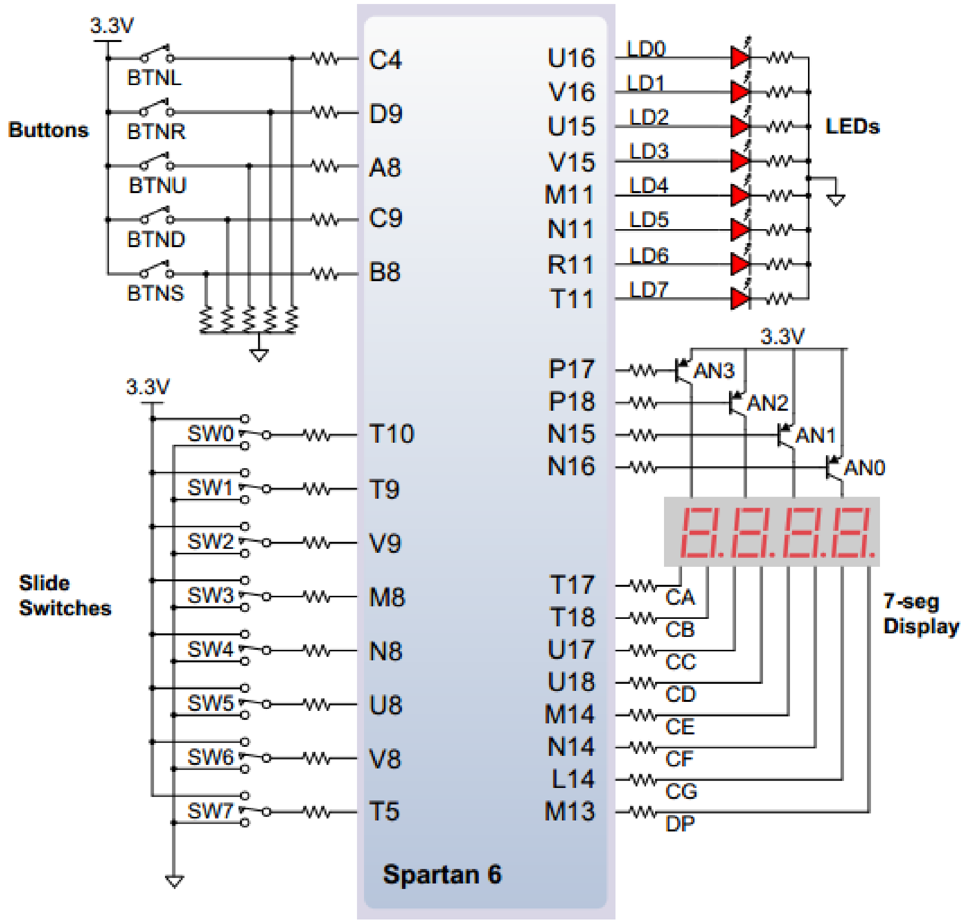
The task of “synchronous (clocked) 2-digit, modulo-32, binary-coded-decimal (BCD) up counter” is a counter 0-31 that shows in the display of 7-segment LEDS. The Nexys3 board contains four-digit; however, for this activity will be utilized only 2 display.

Check above the connection of seven-segment:





An un-illuminated seven-segment display, and nine illumination patterns corresponding to decimal digits



An un-illuminated seven-segment display, and 10 patterns corresponding to decimal digits

| **Numeric Number** | **CA CB CC CD CE CF CG DP** |
|---|---|
| 0 | 00000011 |
| 1 | 10011111 |
| 2 | 00100101 |
| 3 | 00001101 |
| 4 | 10011001 |
| 5 | 01001001 |
| 6 | 01000001 |
| 7 | 00011111 |
| 8 | 00000001 |
| 9 | 00001001 |

## Simulation: macro.do

```
restart

# For simulation I dived in two projects because there is a issue about the low frequency

force CLK 0 0, 1 10ns -r 20ns
force CLR 0

# Test the CLR
run 155ns
force CLR 1
run 155ns
force CLR 0

# Count the numbers
run 3000ns
```

## File .ucf

```
# PlanAhead Generated physical constraints

NET "AN[3]" LOC = P17;

# PlanAhead Generated IO constraints

NET "AN[3]" IOSTANDARD = LVCMOS33;

# PlanAhead Generated physical constraints

NET "AN[0]" LOC = N16;
NET "AN[1]" LOC = N15;
NET "AN[2]" LOC = P18;

# PlanAhead Generated IO constraints

NET "AN[2]" IOSTANDARD = LVCMOS33;
NET "AN[1]" IOSTANDARD = LVCMOS33;
NET "AN[0]" IOSTANDARD = LVCMOS33;
NET "C[7]" IOSTANDARD = LVCMOS33;
NET "C[6]" IOSTANDARD = LVCMOS33;
NET "C[5]" IOSTANDARD = LVCMOS33;
NET "C[4]" IOSTANDARD = LVCMOS33;
NET "C[3]" IOSTANDARD = LVCMOS33;
NET "C[2]" IOSTANDARD = LVCMOS33;
NET "C[1]" IOSTANDARD = LVCMOS33;
NET "C[0]" IOSTANDARD = LVCMOS33;

# PlanAhead Generated physical constraints

NET "C[7]" LOC = T17;
NET "C[6]" LOC = T18;
NET "C[5]" LOC = U17;
NET "C[4]" LOC = U18;
NET "C[3]" LOC = M14;
NET "C[2]" LOC = N14;
NET "C[1]" LOC = L14;
NET "C[0]" LOC = M13;
NET "CLR" LOC = C4;

# PlanAhead Generated IO constraints

NET "CLR" IOSTANDARD = LVCMOS33;
NET "CLK" IOSTANDARD = LVCMOS33;

# PlanAhead Generated physical constraints

NET "CLK" LOC = V10;
```

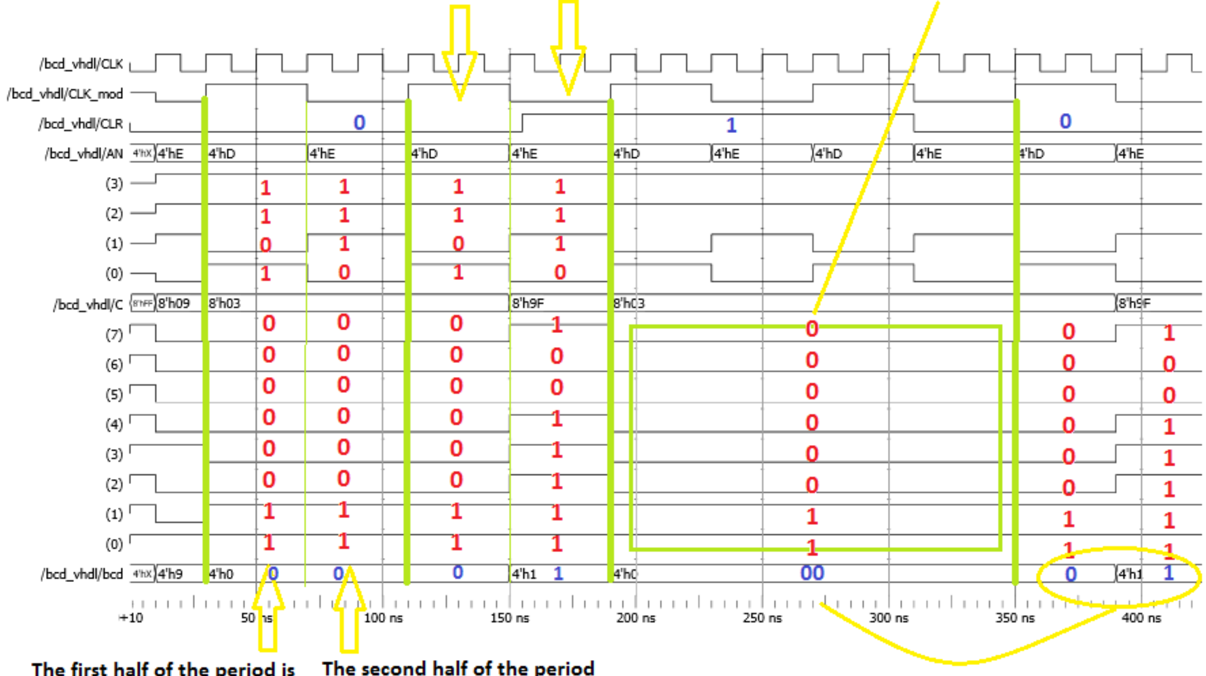
## Wave

The simulation has adapt for the simulation because the frequency is too low.

The next number is "01", Therefore, in the first half of the period is responsible to show the digit "0". The AN = 1101 and C = 00000011

The number is "01", Therefore, in the second half of the period is responsible to show the digit "1". The AN = 1110 and C = 10011111

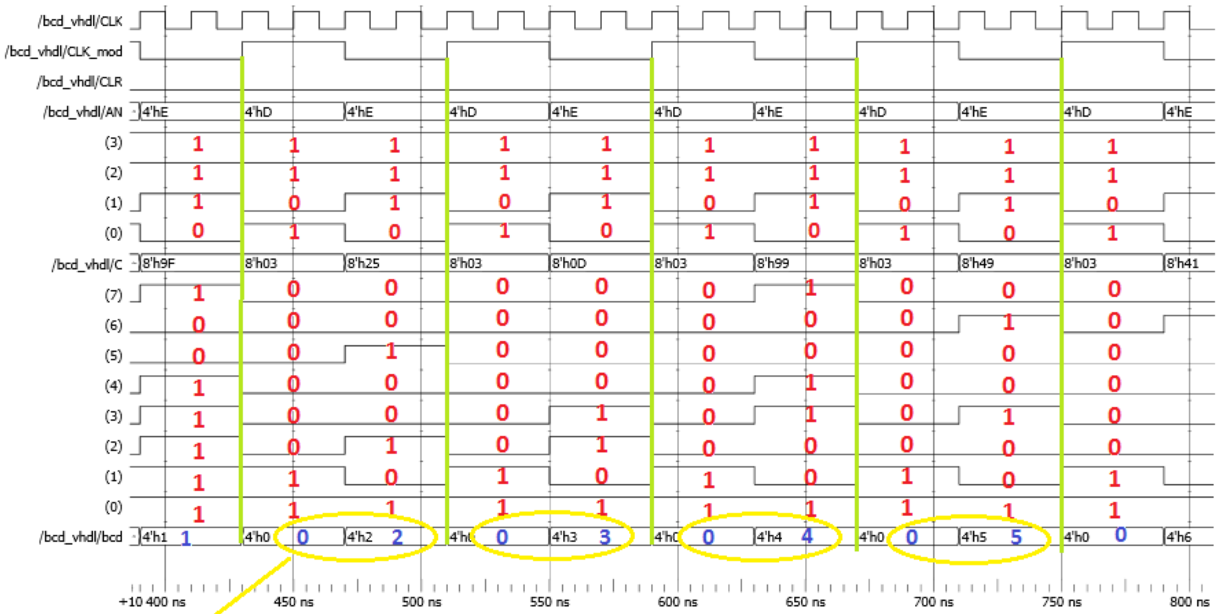
When the CLR = 1 , the display must show "00" until the CLR = 0 and the next clock signal



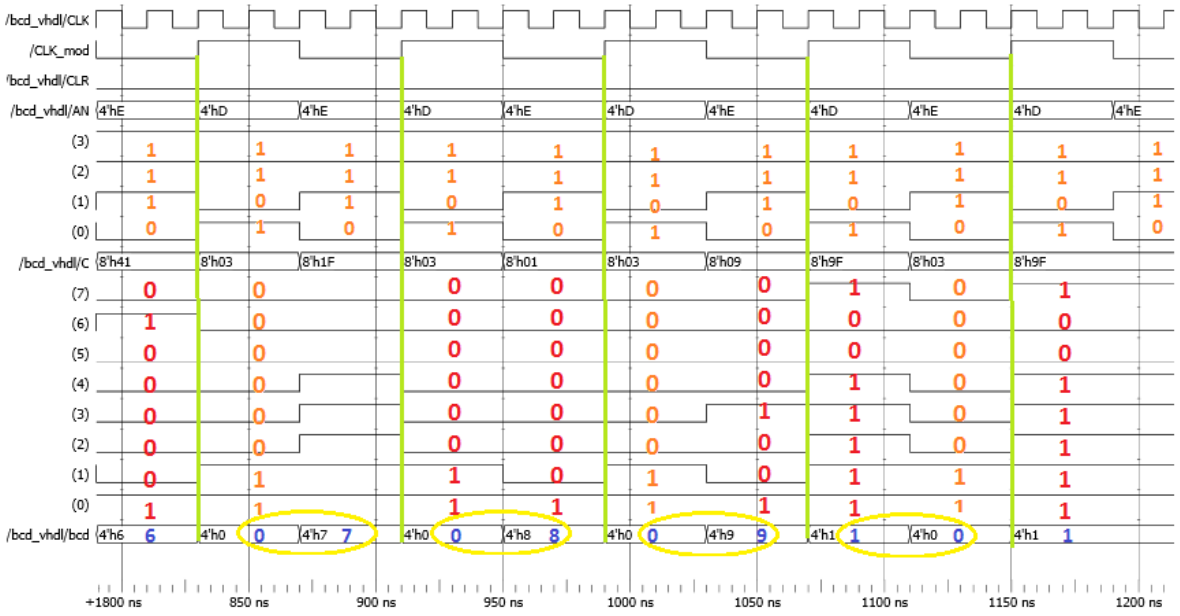
The first half of the period is responsible to show the first digit that can be 0, 1, 2, or 3. Therefore, AN = 1101 and the begin of the counter is "00". Then, C = 00000011

The second half of the period is responsible to show the second digit that can be 0 to 9. Therefore, AN = 1111 and the begin of the counter is "00". Then, C = 00000011

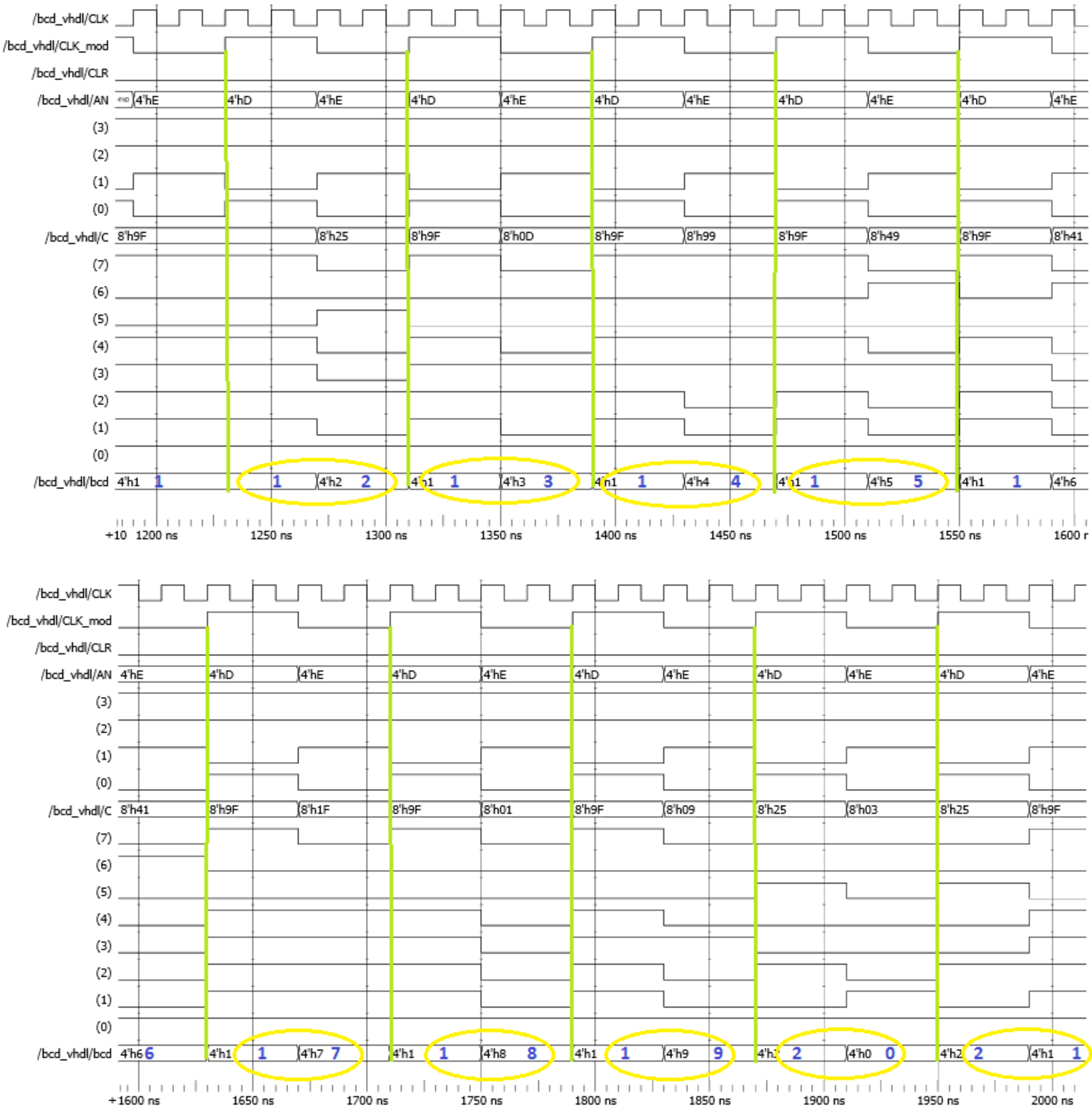
Next number

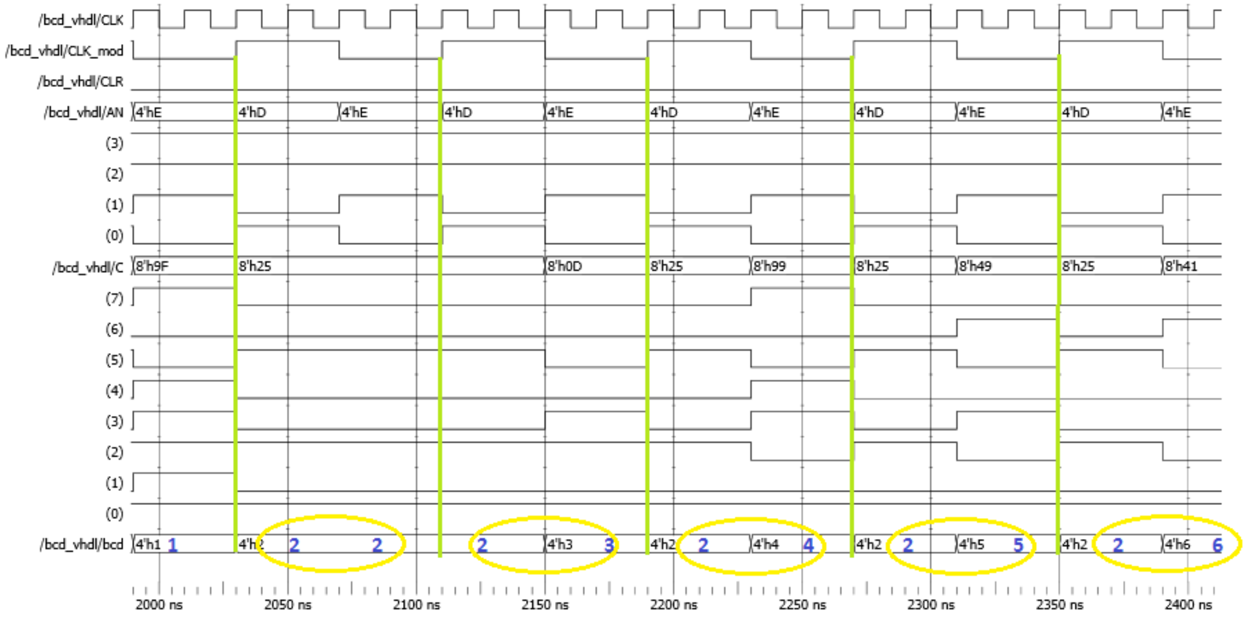


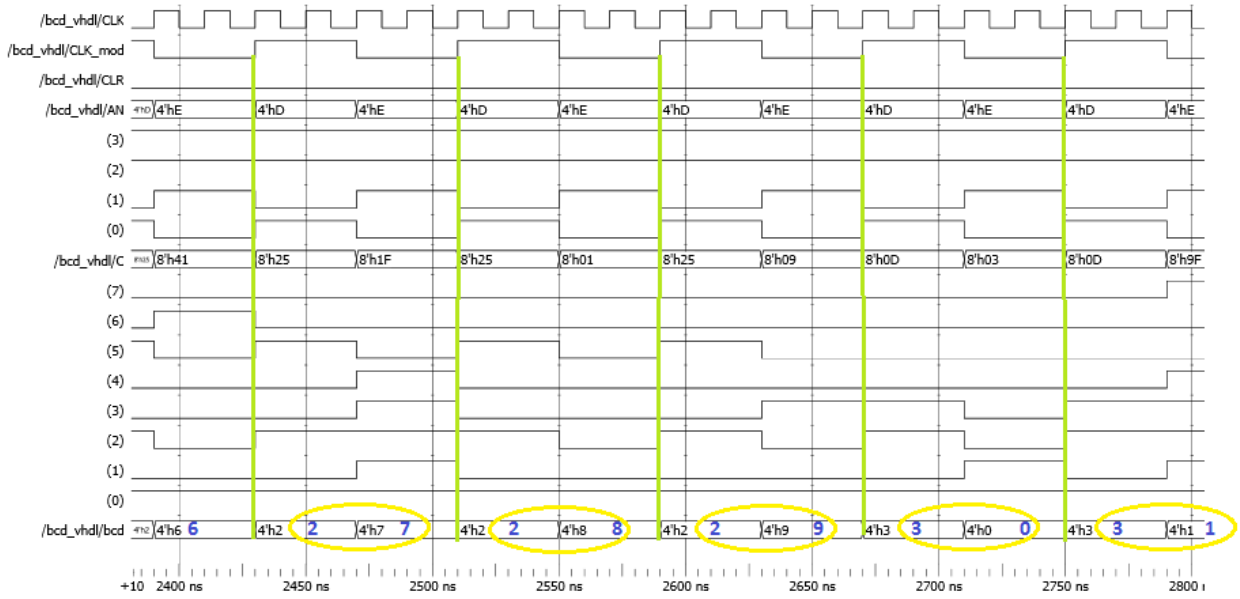
Counter working

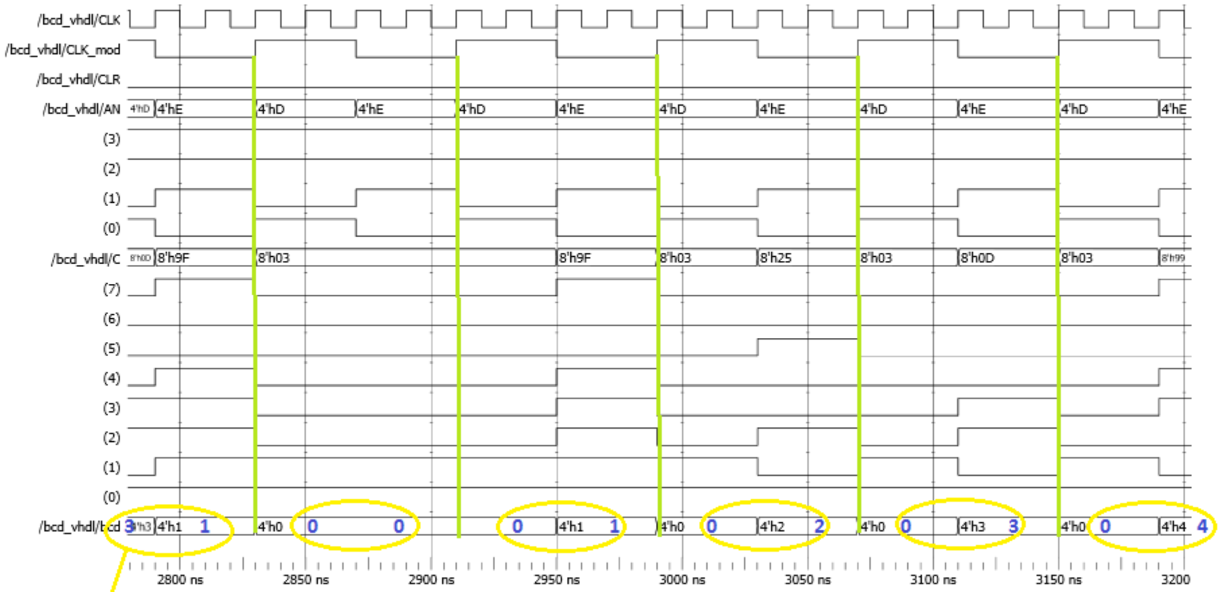


The counter keep going working normally

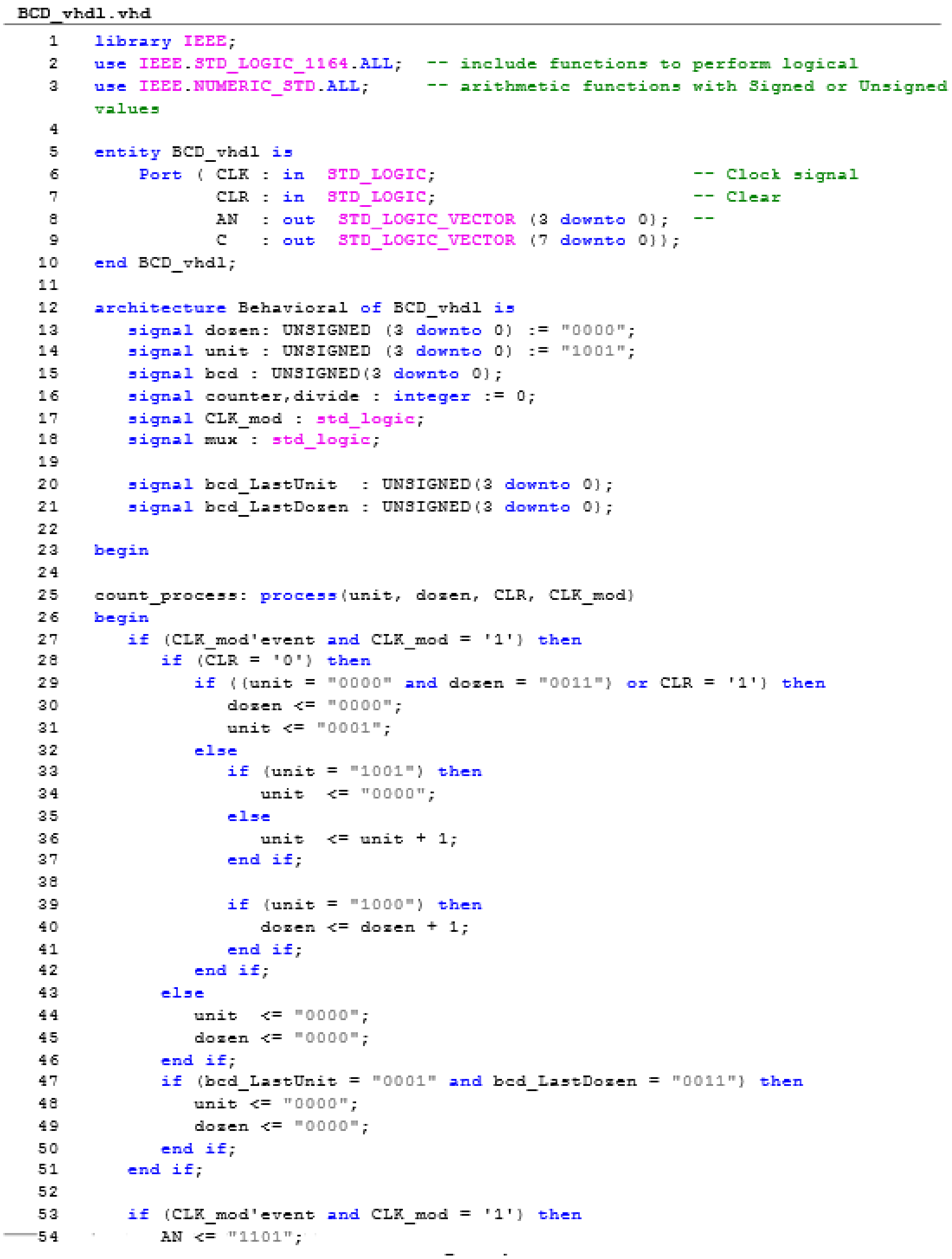



/bcd_vhdl/CLK
/bcd_vhdl/CLK_mod
/bcd_vhdl/CLR
/bcd_vhdl/AN
(3)
(2)
(1)
(0)
/bcd_vhdl/C
(7)
(6)
(5)
(4)
(3)
(2)
(1)
(0)
/bcd_vhdl/bcd
4'hE
4'hD
4'hE
4'hD
4'hE
4'hD
4'hE
4'hD
4'hE
4'hD
4'hE
8'h9F
8'h25
8'h0D
8'h25
8'h99
8'h25
8'h49
8'h25
8'h41
4'h1 1
4'h2 2 2
2 4'h3 3
4'h2 2 4'h4 4
4'h2 2 4'h5 5
4'h2 2 4'h6 6
2000 ns
2050 ns
2100 ns
2150 ns
2200 ns
2250 ns
2300 ns
2350 ns
2400 ns


/bcd_vhdl/CLK
/bcd_vhdl/CLK_mod
/bcd_vhdl/CLR
/bcd_vhdl/AN
(3)
(2)
(1)
(0)
/bcd_vhdl/C
(7)
(6)
(5)
(4)
(3)
(2)
(1)
(0)
/bcd_vhdl/bcd
4'hE
4'hD
4'hE
4'hD
4'hE
4'hD
4'hE
4'hD
4'hE
4'hD
4'hE
8'h41
8'h25
8'h1F
8'h25
8'h01
8'h25
8'h09
8'h0D
8'h03
8'h0D
8'h9F
4'h6 6
4'h2 2 4'h7 7
4'h2 2 4'h8 8
4'h2 2 4'h9 9
4'h3 3 4'h0 0
4'h3 3 4'h1 1
+10
2400 ns
2450 ns
2500 ns
2550 ns
2600 ns
2650 ns
2700 ns
2750 ns
2800 ns

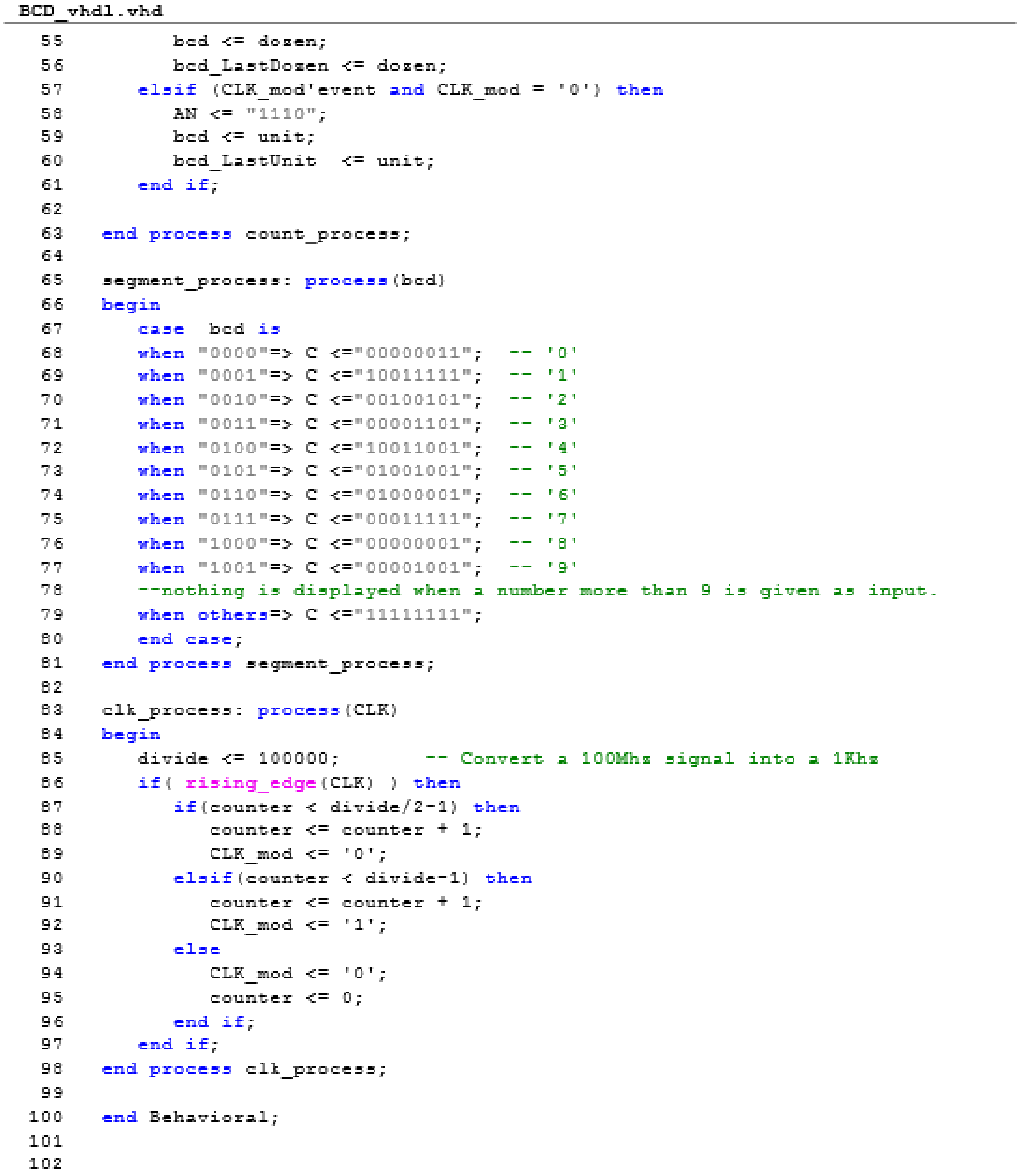


When the counter is "31".
The next digit is "00"



BCD_vhdl.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;  -- include functions to perform logical
use IEEE.NUMERIC_STD.ALL;     -- arithmetic functions with Signed or Unsigned values

entity BCD_vhdl is
    Port ( CLK : in  STD_LOGIC;                     -- Clock signal
           CLR : in  STD_LOGIC;                     -- Clear
           AN   : out  STD_LOGIC_VECTOR (3 downto 0);  --
           C    : out  STD_LOGIC_VECTOR (7 downto 0));
end BCD_vhdl;

architecture Behavioral of BCD_vhdl is
   signal dozen: UNSIGNED (3 downto 0) := "0000";
   signal unit : UNSIGNED (3 downto 0) := "1001";
   signal bcd : UNSIGNED(3 downto 0);
   signal counter,divide : integer := 0;
   signal CLK_mod : std_logic;
   signal mux : std_logic;

   signal bcd_LastUnit  : UNSIGNED(3 downto 0);
   signal bcd_LastDozen : UNSIGNED(3 downto 0);

begin

count_process: process(unit, dozen, CLR, CLK_mod)
begin
   if (CLK_mod'event and CLK_mod = '1') then
      if (CLR = '0') then
         if ((unit = "0000" and dozen = "0011") or CLR = '1') then
            dozen <= "0000";
            unit <= "0001";
         else
            if (unit = "1001") then
               unit  <= "0000";
            else
               unit  <= unit + 1;
            end if;

            if (unit = "1000") then
               dozen <= dozen + 1;
            end if;
         end if;
      else
         unit  <= "0000";
         dozen <= "0000";
      end if;
      if (bcd_LastUnit = "0001" and bcd_LastDozen = "0011") then
         unit <= "0000";
         dozen <= "0000";
      end if;
   end if;

   if (CLK_mod'event and CLK_mod = '1') then
      AN <= "1101";
```



BCD_vhdl.vhd

```
      bcd <= dozen;
      bcd_LastDozen <= dozen;
   elsif (CLK_mod'event and CLK_mod = '0') then
      AN <= "1110";
      bcd <= unit;
      bcd_LastUnit  <= unit;
   end if;

end process count_process;

segment_process: process(bcd)
begin
   case  bcd is
   when "0000"=> C <="00000011";  -- '0'
   when "0001"=> C <="10011111";  -- '1'
   when "0010"=> C <="00100101";  -- '2'
   when "0011"=> C <="00001101";  -- '3'
   when "0100"=> C <="10011001";  -- '4'
   when "0101"=> C <="01001001";  -- '5'
   when "0110"=> C <="01000001";  -- '6'
   when "0111"=> C <="00011111";  -- '7'
   when "1000"=> C <="00000001";  -- '8'
   when "1001"=> C <="00001001";  -- '9'
   --nothing is displayed when a number more than 9 is given as input.
   when others=> C <="11111111";
   end case;
end process segment_process;

clk_process: process(CLK)
begin
   divide <= 100000;       -- Convert a 100Mhz signal into a 1Khz
   if( rising_edge(CLK) ) then
      if(counter < divide/2-1) then
         counter <= counter + 1;
         CLK_mod <= '0';
      elsif(counter < divide-1) then
         counter <= counter + 1;
         CLK_mod <= '1';
      else
         CLK_mod <= '0';
         counter <= 0;
      end if;
   end if;
end process clk_process;

end Behavioral;
```