

2004 VHDL Assignment Solution

First Solution Principle of operation

Random Time

A random time is produced by having a counter running whenever power is applied to the circuit. The random time is the time between the circuit being powered up and the time the start button is pressed. This counter "Count" is a 14 bit counter to frequency divide the 2kHz clock and produce a 8 second period for the rollover of that clock. (i.e. the rollover from Count = 11111111111111 to 00000000000000).

Pressing the start button, sets a WaitRan variable to 1, denoting wait for the next rollover of the counter and clears the RunDisp variable to 0, denoting hold the display value. These two variables are flags controlling the counting and display operations. Pressing the start button also clears the most significant bit of the counter to 0, but does not effect the other bits of the counter, so that between 4 and 8 seconds are required before the counter rolls over.

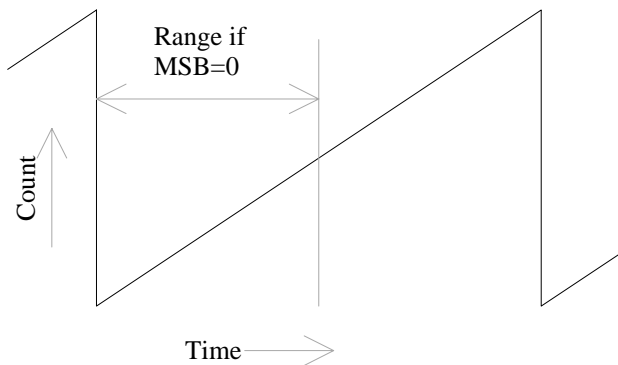


Figure 1. Counter Value with time.

Display

Once the counter rolls over, the RunDisp flag is set to 1 causing the display to start counting. The display counter consists of 4 decade counters, with a value 9 in a display digit, causing that digit to be set to 0 on the next input to that display bit counter. The output from the 4 decade counters are connected to the seven segment displays to display the value of these decade counters. The stop button clears the RunDisp variable to 0, thus holding the display value. If the stop button is not pressed, then the counter counts to 9999 and halts there.

The block diagram for the whole system is shown in Figure 2. The 2 KHz clock is firstly divided by 2, to give a 1kHz clock, which then drives the Least significant digit of the display. The reaction time is thus displayed in milliseconds.

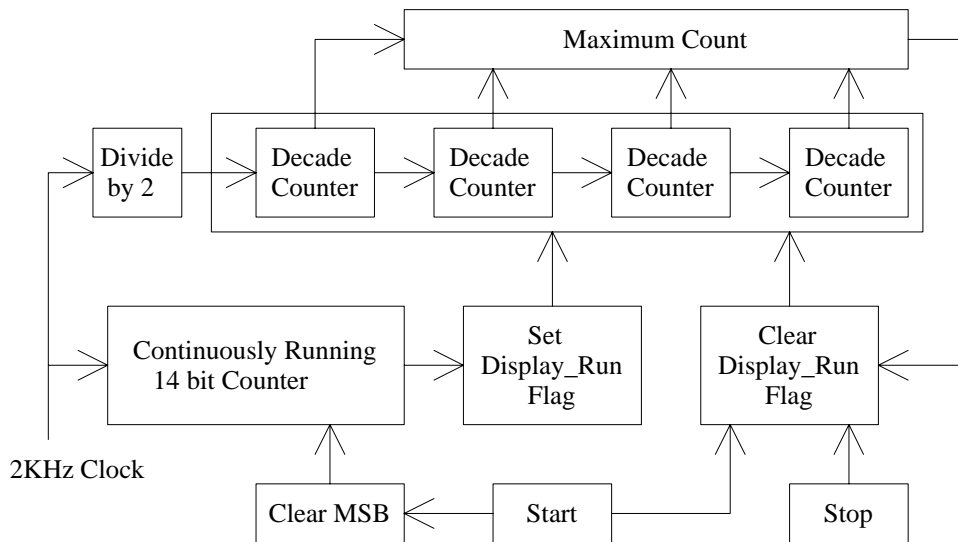


Figure 2. Block Diagram of Reaction Timer Operation

VHDL Code

The VHDL code corresponding to the above block diagram is as follows:

```
-- Reaction Timer for EE4306 Assignment
-- Solution Author C. J. Kikkert Aug 2004
-- Operation: Press Start (SW1) clears the display and stops it incrementing
--   it also clears the MSB of the 14 bit random time counter 'Count',
--   resulting in a 4 to 8 second time before the counter rolls over.
--   This counter runs as long as the circuit is powered up, giving a random time
--   between the powering up of the circuit and the start button being pressed.
-- The counter rolling over starts the display incrementing
-- Pressing the Stop button stops the display incrementing and holds the displayed value
-- If no stop button is pressed, the display stops at the largest value 9999,
-- corresponding to 9.999 seconds. The accuracy of the timer is determined by the
-- RC values of the oscillator and is within about 5%.
```

```
-- A module for driving the seven segment display on the Lattice Boards
```

```
library ieee;
```

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

```
entity SevenSeg is
```

```
  Port (Hexin: in std_logic_vector (3 downto 0);
```

```
        SevSegOut: out std_logic_vector (6 downto 0));
```

```
end;
```

```
architecture SevenSeg_arch of SevenSeg is
```

```
begin
```

```
  process(Hexin)
```

```
  begin
```

```
    Lab0: case Hexin is
```

```
      --need to have inverse of outout coded here
```

```
        when X"0" => SevSegOut <= "0000001";  --0
```

```
        when X"1" => SevSegOut <= "1001111";  --1
```

```
        when X"2" => SevSegOut <= "0010010";  --2
```

```
        when X"3" => SevSegOut <= "0000110";  --3
```

EE4306 VHDL Assignment Solution

```

when X"4" => SevSegOut <= "1001100";    --4
when X"5" => SevSegOut <= "0100100";    --5
when X"6" => SevSegOut <= "0100000";    --6
when X"7" => SevSegOut <= "0001111";    --7
when X"8" => SevSegOut <= "0000000";    --8
when X"9" => SevSegOut <= "0000100";    --9
when X"A" => SevSegOut <= "0001000";    --A
when X"B" => SevSegOut <= "1100000";    --b
when X"C" => SevSegOut <= "0110001";    --C
when X"D" => SevSegOut <= "1000010";    --d
when X"E" => SevSegOut <= "0110000";    --E
when others => SevSegOut <= "0111000";    --F
    end case Lab0;
end process;
end;
-- end of seven segment display driver

-- The main program for the Reaction Timer
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity ReactTime is
    Port (Clk : in std_logic ;    --2KHz to permit 0.001sec resolution

          SevSeg0: out std_logic_vector (6 downto 0); -- Seven segment display LSB
          SevSeg1: out std_logic_vector (6 downto 0); -- Seven segment display
          SevSeg2: out std_logic_vector (6 downto 0); -- Seven segment display
          SevSeg3: out std_logic_vector (6 downto 0); -- Seven segment display MSB
          Stop: in std_logic;      --stop button SW3
          Start: in std_logic);    --start button SW1

    attribute loc : string;
    attribute loc of Clk:signal is "P11" ;
    attribute loc of Start: signal is "P9";    --Start button SW1
    attribute loc of Stop: signal is "P31";    --Stop button SW3
    attribute loc of SevSeg3:signal is "P2 P3 P4 P5 P6 P7 P8" ; -- Pin allocation of Displays
    attribute loc of SevSeg2:signal is "P14 P15 P16 P17 P18 P19 P20" ;
    attribute loc of SevSeg1:signal is "P24 P25 P26 P27 P28 P29 P30" ;
    attribute loc of SevSeg0:signal is "P36 P37 P38 P39 P40 P41 P42" ;
end;

architecture ReactTime_arch of ReactTime is
    component SevenSeg
        port (Hexin: in std_logic_vector;      -- input for Display Module
              SevSegOut : out std_logic_vector); -- output from Display Module
    end component;

    signal CountB0: std_logic_vector (3 downto 0);    -- LS Digit Decade Display Counter.
    signal CountB1: std_logic_vector (3 downto 0);    -- Digit 2 Decade Display Counter.
    signal CountB2: std_logic_vector (3 downto 0);    -- Digit 3 Decade Display Counter.
    signal CountB3: std_logic_vector (3 downto 0);    -- Digit 4 Decade Display Counter.
    signal Count: std_logic_vector (13 downto 0);
        -- 14 bit counter, 8 second interval for 2.048kHz clock
    --signal Count: std_logic_vector (6 downto 0);    -- 8 bit short counter for testing
    signal RunDisp: std_logic; --Flag to increment Display counter
    signal WaitRan: std_logic; --wait for random start time

```

EE4306 VHDL Assignment Solution

begin

```
D0: SevenSeg port map (CountB0, SevSeg0);    -- Seven Segment Display LSDigit
D1: SevenSeg port map (CountB1, SevSeg1);    -- Seven Segment Display Digit 2
D2: SevenSeg port map (CountB2, SevSeg2);    -- Seven Segment Display Digit 3
D3: SevenSeg port map (CountB3, SevSeg3);    -- Seven Segment Display MSDigit
```

P_Clock2: process

begin -- counter running always with 8 second period

wait until rising_edge(Clk);

```
Count <= Count + "0000000000001"; -- always increment Counter
```

```
-- Count <= Count + "0000001";      -- use this for testing only
```

```
if ((Count = "000000000000") and (WaitRan = '1')) then
```

```
RunDisp <= '1';    --Random time arrived, Start the display
```

```
elsif (Start = '0') then -- start button pressed initiate process when button released.
```

```
WaitRan <= '1';    -- Wait for random starting time.
```

```
RunDisp <= '0';    -- do not increment the display
```

```
Count(13) <= '0';  -- clear MSB of counter to ensure >4 sec to rollover
```

```
-- Count(6) <= '0';  -- clear MSB of short test counter
```

```
CountB0 <= "0000";    --clear the display
```

```
CountB1 <= "0000";    --clear the display
```

```
CountB2 <= "0000";    --clear the display
```

```
CountB3 <= "0000";    --clear the display
```

```
elsif (Stop = '0') then    -- stop button pressed
```

```
RunDisp <= '0';    -- Stop the display incrementing
```

```
WaitRan <= '0';    -- Disable waiting for random time
```

```
elsif ((RunDisp = '1') and (Count(0) = '1')) then -- increment display until stop press
```

```
-- Count(0)=1 results in 1024Hz clock
```

```
if (CountB0 = X"9") and (CountB1 = X"9") and (CountB3 = X"9")
```

```
and (CountB2 = X"9") then
```

```
-- maximum count reached halt the display on 9999
```

```
RunDisp <= '0';    -- Stop Counter (same as Stop)
```

```
WaitRan <= '0';    -- Disable waiting for random time
```

```
elsif (CountB0 = X"9") then    -- Maximum for this digit increment next digit
```

```
CountB0 <= "0000";
```

```
if (CountB1 = X"9") then -- Maximum, increment next digit
```

```
CountB1 <= "0000";
```

```
if (CountB2 = X"9") then -- Maximum, increment next digit
```

```
CountB2 <= "0000";
```

```
CountB3 <= CountB3 + "001";
```

```
-- CountB3 is stopped at max value no rollover
```

```
else
```

```
CountB2 <= CountB2 + "0001";    -- increment digit
```

```
end if;    --B2 Count
```

```
else
```

```
CountB1 <= CountB1 + "0001"; -- increment digit
```

```
end if;    --B1 Count
```

```
else
```

```
CountB0 <= CountB0 + "0001";    -- increment digit
```

```
end if;    --B0 Count
```

```
end if;
```

```
--end mode selection
```

```
end process P_Clock2;
```

```
end ReactTime_arch;
```

EE4306 VHDL Assignment Solution

This design uses the MA4A5. For this design a smaller fit is obtained using the “Speed” optimisation constraint than with the Area optimisation constraint. That is not normal. In addition the above code will fit with the precision compiler, but not with the Synario compiler. Using the default Optimization Constraints and the Precision compiler for ispLever 5.0 gives:

Design_Summary

~~~~~

|                         |     |
|-------------------------|-----|
| Total Input Pins :      | 3   |
| Total Output Pins :     | 28  |
| Total Bidir I/O Pins :  | 0   |
| Total Flip-Flops :      | 32  |
| Total Product Terms :   | 249 |
| Total Reserved Pins :   | 0   |
| Total Reserved Blocks : | 0   |

### Device\_Resource\_Summary

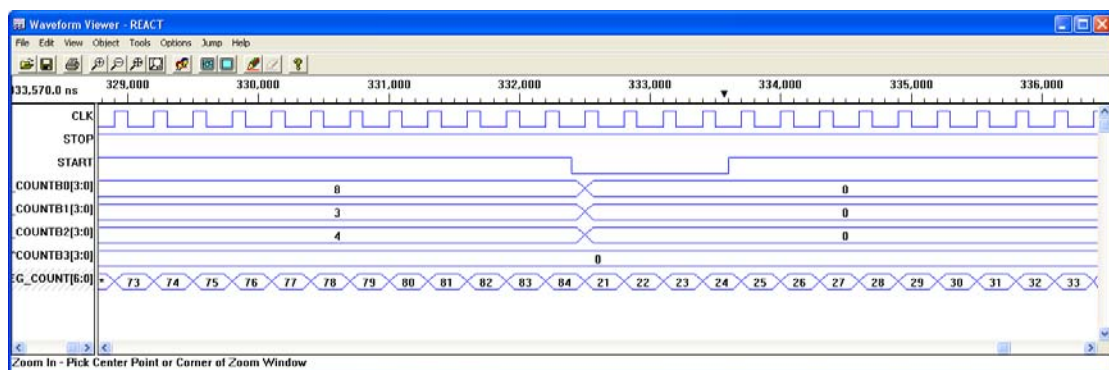
~~~~~

Total

	Available	Used	Available	Utilization
Dedicated Pins				
Input-Only Pins -->	..
Clock/Input Pins	2	1	1 -->	50%
I/O Pins	32	30	2 -->	93%
Logic Macrocells	64	64	0 -->	100%
Input Registers	32	0	32 -->	0%
Unusable Macrocells	..	0	..	
CSM Outputs/Total Block Inputs	132	120	12 -->	90%
Logical Product Terms	320	260	60 -->	81%
Product Term Clusters	64	64	0 -->	100%

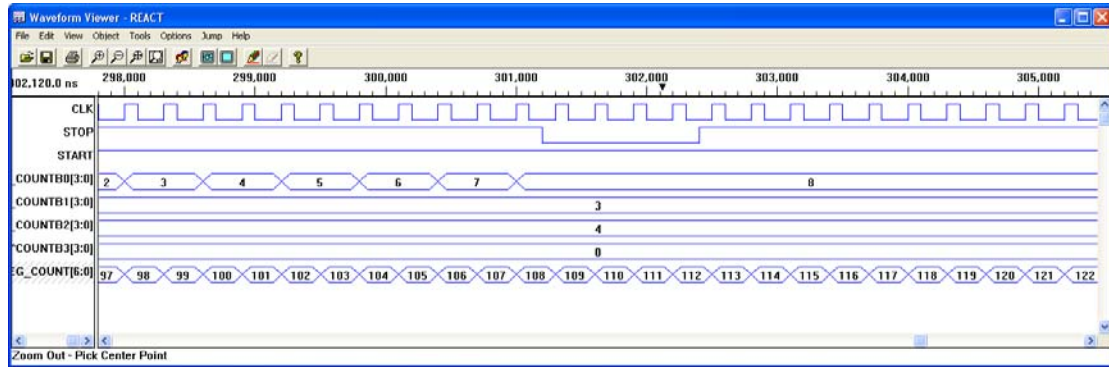
Different compiler versions will give slightly different fitting results.

For testing the design a random delay is used. The code for this is commented out to facilitate the change between the full realisation and the testing only realisation. The resulting waveforms are shown below:



Waveform showing how the most significant bit of the count(6:0) is cleared.

EE4306 VHDL Assignment Solution



Waveform showing how the count being halted and the reaction time being displayed

The VHDL code can be implemented in the CPLD and the reaction timer operates as required.

Second Solution Principle of operation

The above solution requires a 14 bit counter for producing the random time and a 16 bit counter for storing the displayed value. By only using one counter to drive both the display and produce the random time, a smaller solution is obtained. To use the one counter, the display needs to be turned OFF when the random time is counted and the display needs to be ON when the reaction time is being evaluated. This can be achieved by adding an extra input to the seven segment display, to have the Hex input displayed or have the display blanked or showing say - - - -, when the random delay is being evaluated. See the Sevenseg component in the VHDL code, to see how that is achieved.

To produce the random delay, the counters are incremented directly from the 2 kHz clock when the start button is being pushed. The randomness will result from the different length of time that the start button is pushed. To obtain the correct delay, the counters are cycled between 2000 and 5999, resulting in a 4 to 8 second delay before 0000 is reached.

The states are as follows:

The Start button is pushed: Increment each digit individually to produce the random number.

The MSB digit cycles between 2 and 5, MSB2 digit cycles between 0 and 9, LSB2 digit counts up in hex, LSB digit counts down in hex, to produce a random number.

Once the Start button is released, the counter counts as a normal decade counter.

Once 999 is reached the ShowDisp flag is set, turning on the display to show the count value.

This is the cue to the operator to press the stop button.

When the stop button is pressed or the display reaches 9999, the Halt flag is set which causes the display to stop incrementing

The resulting code is as follows:

```
-- Reaction Timer for EE4306 Assignment
-- Solution2 Author C. J. Kikkert Aug 2004

-- Operation: Press Start (SW1) blanks the display (shows ---- )
-- it also increments the display counters at a 2 kHz rate,
-- resulting in a 4 to 8 second time before the counter rolls over.
-- The counter rolling over starts the display incrementing
-- Pressing the Stop button stops the display incrementing and holds the displayed value
-- If no stop button is pressed, the display stops at the largest value 9999,
-- corresponding to 9.999 seconds. The accuracy of the timer is determined by the
```

EE4306 VHDL Assignment Solution

```
-- RC values of the oscillator and is within about 5%.

-- A module for driving the seven segment display on the Lattice Boards
library ieee;
use ieee.std_logic_1164.all;

entity SevenSeg is
    Port (ShowDisp: in std_logic;
          Hexin: in std_logic_vector (3 downto 0);
          SevSegOut: out std_logic_vector (6 downto 0));
end;

architecture SevenSeg_arch of SevenSeg is

begin
    process(ShowDisp, Hexin)
    begin
        if (ShowDisp = '1') then
            Lab0: case Hexin is
                -- need to have inverse of outout coded here
                -- A to F are not used here but best to use full display
                -- a 5 bit input is used to give one bit to turn the counter ON and OFF and 4 bits for the value.

                when X"0" => SevSegOut <= "0000001"; --0
                when X"1" => SevSegOut <= "1001111"; --1
                when X"2" => SevSegOut <= "0010010"; --2
                when X"3" => SevSegOut <= "0000110"; --3
                when X"4" => SevSegOut <= "1001100"; --4
                when X"5" => SevSegOut <= "0100100"; --5
                when X"6" => SevSegOut <= "0100000"; --6
                when X"7" => SevSegOut <= "0001111"; --7
                when X"8" => SevSegOut <= "0000000"; --8
                when X"9" => SevSegOut <= "0000100"; --9
                when X"A" => SevSegOut <= "0001000"; --A
                when X"B" => SevSegOut <= "1100000"; --b
                when X"C" => SevSegOut <= "0110001"; --C
                when X"D" => SevSegOut <= "1000010"; --d
                when X"E" => SevSegOut <= "0110000"; --E
                when others => SevSegOut <= "0111000";--F;
            end case Lab0;
            else -- Display OFF
                SevSegOut <= "1111110"; -- show ----
            end if;
        end process;
    end;
-- end of seven segment display driver

-- The main program for the Reaction Timer
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
--use ieee.std_logic_arith.all;

entity ReacTime is
    Port (Clk : in std_logic ; --2KHz to permit 0.001sec resolution
          SevSeg0: out std_logic_vector (6 downto 0); -- Seven segment display LSB
          SevSeg1: out std_logic_vector (6 downto 0); -- Seven segment display
          SevSeg2: out std_logic_vector (6 downto 0); -- Seven segment display
          SevSeg3: out std_logic_vector (6 downto 0); -- Seven segment display MSB
          Stop: in std_logic; --stop button SW3
          Start: in std_logic); --start button SW1

    attribute loc : string;
    attribute loc of Clk:signal is "P11" ;
    attribute loc of Start: signal is "P9"; --Start button SW1
    attribute loc of Stop: signal is "P31"; --Stop button SW3
    attribute loc of SevSeg3:signal is "P2 P3 P4 P5 P6 P7 P8" ; -- Pin allocation of Displays
    attribute loc of SevSeg2:signal is "P14 P15 P16 P17 P18 P19 P20" ;
    attribute loc of SevSeg1:signal is "P24 P25 P26 P27 P28 P29 P30" ;
    attribute loc of SevSeg0:signal is "P36 P37 P38 P39 P40 P41 P42" ;
end;
```

EE4306 VHDL Assignment Solution

```
architecture ReactTime_arch of ReactTime is
  component SevenSeg
    port (ShowDisp: in std_logic;           -- Display ON/OFF
          Hexin: in std_logic_vector;      -- input for Display Module
          SevSegOut : out std_logic_vector); -- output from Display Module
  end component;

  signal CountB0: std_logic_vector (3 downto 0); -- LS Digit Decade Display Counter.
  signal CountB1: std_logic_vector (3 downto 0); -- Digit 2 Decade Display Counter.
  signal CountB2: std_logic_vector (3 downto 0); -- Digit 3 Decade Display Counter.
  signal CountB3: std_logic_vector (3 downto 0); -- Digit 4 Decade Display Counter.
  signal Clk1: std_logic; -- 1 kHz clock
  signal ShowDisp: std_logic; --Flag to turn Display counter ON and OFF ShowDisp=1 display ON
  signal Halt: std_logic; --Run or Hold Display Halt = 1 stop counter
  signal Max : std_logic; --Max = 1 when display = 9999
begin

  D0: SevenSeg port map (ShowDisp,CountB0, SevSeg0); -- Seven Segment Display LSB
  D1: SevenSeg port map (ShowDisp,CountB1, SevSeg1); -- Seven Segment Display
  D2: SevenSeg port map (ShowDisp,CountB2, SevSeg2); -- Seven Segment DisplayMSB
  D3: SevenSeg port map (ShowDisp,CountB3, SevSeg3); -- Seven Segment Display LSB

  P_Clock2: process
  begin -- counter running always with 8 second period
  wait until rising_edge(Clk); -- Clk is a 2048Hz clock
  Clk1 <= not Clk1; --Toggle 2 kHz clock
  if ((Start = '0') or ((Stop = '0') and (ShowDisp = '0'))) then -- start button pressed randomise the display
    -- or Stop button pushed before the display in OFF
    Halt <= '0'; -- Run the counter.
    ShowDisp <= '0'; -- blank the display set msb of dispin
    if (CountB3 < X"5") then --5999 gives 4 seconds to 9999
    --
    if (CountB3 < X"9") then --test only
      CountB3 <= CountB3 + X"1"; --increment the display MSB at 2kHz rate
    else
      CountB3 <= X"2"; --2000 gives 8 seconds to 9999
    --
      CountB3 <= X"9"; --test only want short time
    end if;
    if (CountB2 < X"9") then -- MSB2 displays numbers 0 to 9
      CountB2 <= CountB2 + X"1"; --increment the display MSB at 2kHz rate
    else
      CountB2 <= X"0"; -- rollover MSB2
    --
      CountB2 <= X"8"; -- test only
    end if;
    CountB1 <= CountB1 + X"1"; -- increment the display 2LSB at 2kHz rate
    CountB0 <= CountB0 - X"1"; -- decrement the display LSB
    --Note CountB0 and CountB1 have different rollover rates making the display sufficiently random
    -- also depends on how long the start button is pressed.

  elsif ((Halt = '1') or (Clk1 = '0')) then -- Don't increment the display
    -- Count(0)=0 results in 1024Hz incrementing of the counters
    CountB3 <= CountB3; -- keep the same values
    CountB2 <= CountB2;
    CountB1 <= CountB1;
    CountB0 <= CountB0;

    -- all below only is done when Clk1 = 1
    elsif (Stop = '0') and (ShowDisp = '1') then -- stop button when display ON Stop incrementing
      Halt <= '1'; -- Halt the display
      CountB3 <= CountB3; -- keep the same values
      CountB2 <= CountB2;
      CountB1 <= CountB1;
      CountB0 <= CountB0;

    elsif ((max = '1') and (ShowDisp = '1')) then
      Halt <= '1'; --counter reached 9999, Halt the display Stop not pushed

    else -- run display counter until stop button pressed or max is reached
      if ((max = '1') and (Showdisp = '0')) then
        ShowDisp <= '1'; -- turn ON display
      end if;
      -- Clk1=1 results in 1024Hz clock
    end if;
  end process;
end;
```

EE4306 VHDL Assignment Solution

```

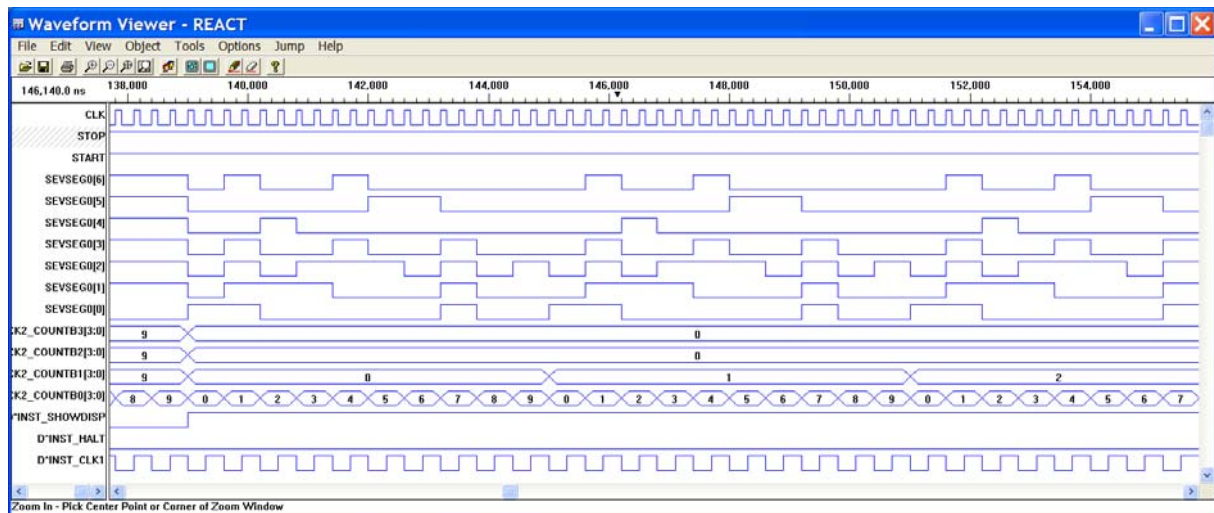
if (CountB0 = X"9") then          -- Maximum for this digit increment next digit
    CountB0 <= X"0";
    if (CountB1 = X"9") then      -- Maximum for this digit increment next digit
        CountB1 <= X"0";
        if (CountB2 = X"9") then  -- Maximum for this digit increment next digit
            CountB2 <= X"0";
            if (CountB3 = X"9") then -- Maximum for this digit increment next digit
                CountB3 <= X"0";   -- this rollover only occurs when start showing display ON
            else
                CountB3 <= CountB3 + X"1"; -- CountB3 is stopped at max value no rollover
            end if;
        else
            CountB2 <= CountB2 + X"1"; -- increment digit
        end if;
    else
        CountB1 <= CountB1 + X"1"; -- increment digit
    end if;
else
    CountB0 <= CountB0 + X"1";    -- increment digit
end if;
--B0 Count
--B1 Count
--B2 Count
--end mode selection
end process P_Clock2;

process (CountB0, CountB1, CountB2, CountB3)
begin
    max <= ( CountB0(3) and (not CountB0(2)) and (not CountB0(1)) and CountB0(0) and -- 9 = 1001
             CountB1(3) and (not CountB1(2)) and (not CountB1(1)) and CountB1(0) and -- 9 = 1001
             CountB2(3) and (not CountB2(2)) and (not CountB2(1)) and CountB2(0) and -- 9 = 1001
             CountB3(3) and (not CountB3(2)) and (not CountB3(1)) and CountB3(0)); -- max=1 if

    display 9999
end process;
end ReactTime_arch;

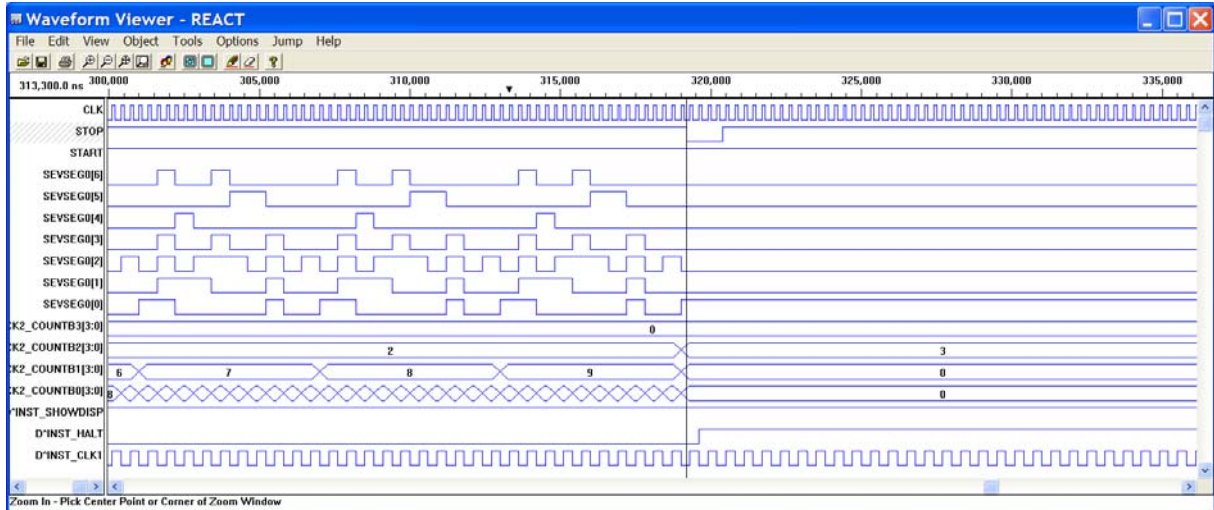
```

The resulting waveforms are as follows:

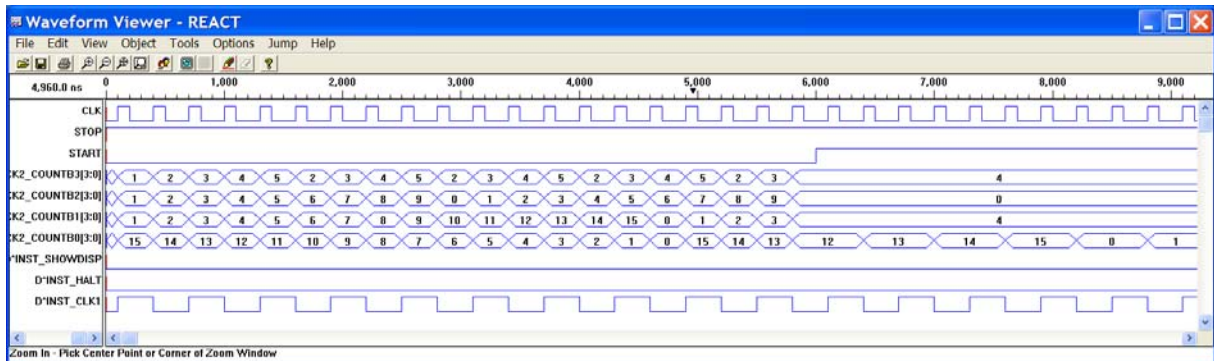


The time when the counter reaches 9999 and turns ON the display.

EE4306 VHDL Assignment Solution



The Stop button being pushed to stop the counter incrementing.



The rapid counting when the Start Button is pushed.

Device_Resource_Summary

~~~~~

|                                | Total Available | Used | Available | Utilization |
|--------------------------------|-----------------|------|-----------|-------------|
| Dedicated Pins                 |                 |      |           |             |
| Input-Only Pins                | ..              | ..   | ..        | --> ..      |
| Clock/Input Pins               | 2               | 1    | 1         | --> 50%     |
| I/O Pins                       | 32              | 30   | 2         | --> 93%     |
| Logic Macrocells               | 64              | 51   | 13        | --> 79%     |
| Input Registers                | 32              | 0    | 32        | --> 0%      |
| Unusable Macrocells            | ..              | 0    | ..        |             |
| CSM Outputs/Total Block Inputs | 132             | 80   | 52        | --> 60%     |
| Logical Product Terms          | 320             | 225  | 95        | --> 70%     |
| Product Term Clusters          | 64              | 58   | 6         | --> 90%     |

It can be seen that this realisation uses 13 less macrocells, when the fitting is optimised for speed. One more macrocell is used when the fitting is optimised for area.