**University of Alexandria**

Faculty of Engineering

Communications & Electronics Department

# VHDL PROJECT:

# (STOP WATCH)

## NAMES AND IDS:

| | | |
|---|---|---|
| 1. | سعيد مبروك سعيد الشيخ -Leader- | 18010788 |
| 2. | ابانوب مرقص جادالسيد عبدالمسيح | 18010004 |
| 3. | سيمون جوزيف شحاته عبدالملاك | 18010835 |
| 4. | مارينا فؤاد عزيز | 18011310 |
| 5. | احمد محمد رشدي ابوالعينين | 18010212 |

## VHDL CODE:

```
--Please Don't Enter Seconds More Than 59 Because It Force The Program To
Finish Immediately.

library ieee;

use ieee.std_logic_1164.all;

entity LABVHDL is

port (clk,start_stop :in std_logic ;

     min_in , sec_in : in std_logic_vector (5 downto 0)  ;

                min_out , sec_out : out std_logic_vector (5 downto 0);

                finish: out std_logic);

end LABVHDL ;

architecture behavioural of LABVHDL is

signal c : integer range 0 to 3;

begin

                process (clk)

          variable min , sec : std_logic_vector ( 5 downto 0) ;

                begin

                          if (clk 'event and clk ='1' and start_stop ='1' ) then

                          if(c=0 ) then

                                        min:= min_in ;

                                        sec:=sec_in ;

                                        end if ;
```

```
                    if  ((sec > "111011") or (min>"111011")) then
                            min:= "000000" ;
                            sec:="000000" ;
                            finish <='1';
                    elsif (sec="000000") then
                        if (c<2) then
                            c<=c+1;
                            end if;
                        if (min="000000") then
                                finish <= '1';
                            else
                                if min(0)='1' then min(0):='0' ;
                                elsif min(1)='1' then min(0):='1';min(1):='0' ;
                                elsif min(2)='1' then
min(0):='1';min(1):='1';min(2):='0' ;
                                elsif min(3)='1' then
min(0):='1';min(1):='1';min(2):='1';min(3):='0' ;
                                elsif min(4)='1' then
min(0):='1';min(1):='1';min(2):='1';min(3):='1';min(4):='0' ;
                                elsif min(5)='1' then
min(0):='1';min(1):='1';min(2):='1';min(3):='1';min(4):='1';min(5):='0' ;
                                end if;
                                sec:="111011";
```

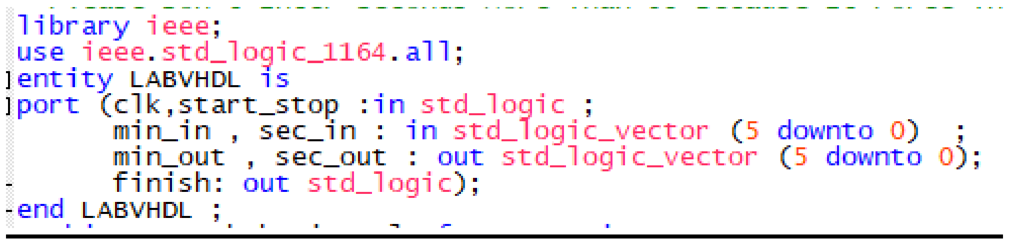
```
                                                  finish<='0';
                                                  end if;
                                  elsif (sec>"000000") then
                                        if (c<2) then
                                              c<=c+1;
                                              end if;
                                        if sec(0)='1' then sec(0):='0' ;
                                        elsif sec(1)='1' then sec(0):='1';sec(1):='0' ;
                                        elsif sec(2)='1' then sec(0):='1';sec(1):='1';sec(2):='0' ;
                                        elsif sec(3)='1' then
sec(0):='1';sec(1):='1';sec(2):='1';sec(3):='0' ;
                                        elsif sec(4)='1' then
sec(0):='1';sec(1):='1';sec(2):='1';sec(3):='1';sec(4):='0' ;
                                        elsif sec(5)='1' then
sec(0):='1';sec(1):='1';sec(2):='1';sec(3):='1';sec(4):='1';sec(5):='0' ;
                                        end if;
                                        finish<='0';
                                   end if ;
                            end if ;
                                  min_out<= min ;
                            sec_out<= sec;
                  end process;
end behavioural;
```

## STEP OF DESIGN:

First we create an entity

its inputs: clk , start_stop (std_logic) and min_in , sec_in (std_logic_vector 6 bits).

its outputs: min_out , sec_out (std_logic_vector 6 bits) and finish (std_logic).



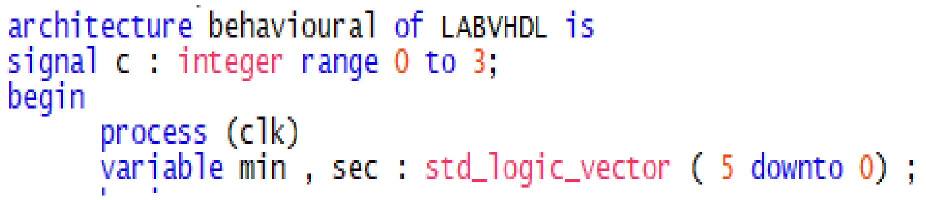
```
library ieee;
use ieee.std_logic_1164.all;
entity LABVHDL is
port (clk,start_stop :in std_logic ;
      min_in , sec_in : in std_logic_vector (5 downto 0)  ;
      min_out , sec_out : out std_logic_vector (5 downto 0);
      finish: out std_logic);
end LABVHDL ;
```

Then we create an architecture :

**First step** is to create a signal "c" and it works as a counter and it's an integer

**Second** we create a process depends on the clock(clk)

Before begin the process we create two variables called min , sec (std_logic_vector 6 bits)



```
architecture behavioural of LABVHDL is
signal c : integer range 0 to 3;
begin
      process (clk)
      variable min , sec : std_logic_vector ( 5 downto 0) ;
```

Before beginning to decrease the mins and the seconds that the user input as the stop watch works we check some conditions

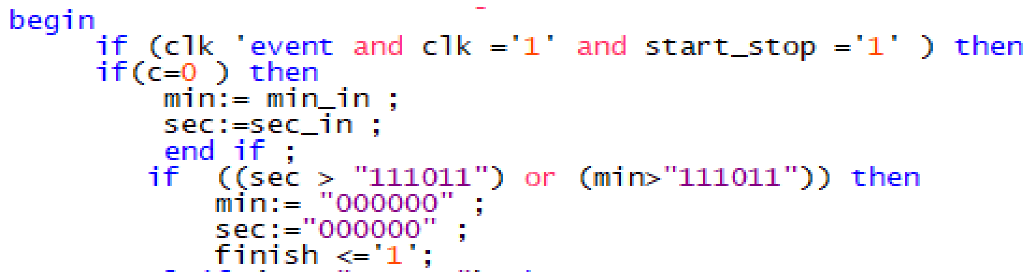
first thing we check if the clock is raising edge and start_stop equal to 1

second if the counter c equals 0.. we put the variables min and sec equal to min_in and sec_in that the user input (the role of the counter is only to dectect the first value to decrease…if c doesn't equal zero we will take the last value and decrease it)

third we check if  sec > "111011" (59 decimal) or min > "111011" (59 decimal)

If this condition is true we make min="000000"

sec="000000" and finish='1' because it's an invalid input for this code

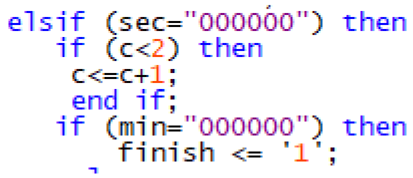


```
begin
     if (clk 'event and clk ='1' and start_stop ='1' ) then
     if(c=0 ) then
         min:= min_in ;
         sec:=sec_in ;
         end if ;
        if  ((sec > "111011") or (min>"111011")) then
            min:= "000000" ;
            sec:="000000" ;
            finish <='1';
```

if the inputs are valid (less than or equal 111011) we check if the seconds is equal "000000"

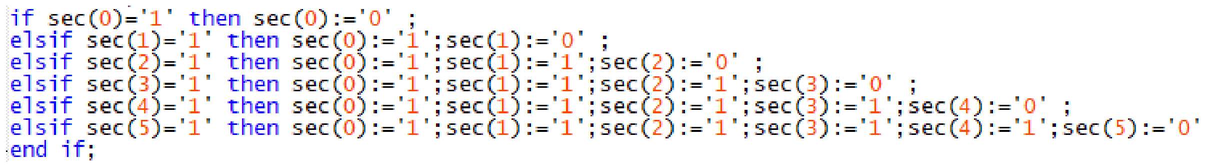
if sec= "000000"  so we increase the counter c by 1

and if min = "000000" too so finish='1'.



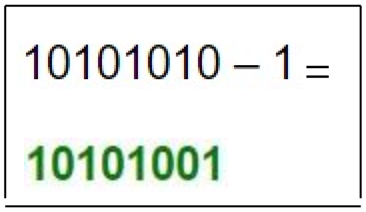
```
elsif (sec="000000") then
   if (c<2) then
    c<=c+1;
    end if;
   if (min="000000") then
       finish <= '1';
```

if not…we begin to decrease the mins as this block of codes makes:



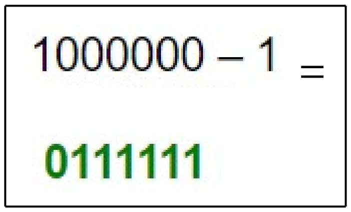
```
if sec(0)='1' then sec(0):='0' ;
elsif sec(1)='1' then sec(0):='1';sec(1):='0' ;
elsif sec(2)='1' then sec(0):='1';sec(1):='1';sec(2):='0' ;
elsif sec(3)='1' then sec(0):='1';sec(1):='1';sec(2):='1';sec(3):='0' ;
elsif sec(4)='1' then sec(0):='1';sec(1):='1';sec(2):='1';sec(3):='1';sec(4):='0' ;
elsif sec(5)='1' then sec(0):='1';sec(1):='1';sec(2):='1';sec(3):='1';sec(4):='1';sec(5):='0'
end if;
```

The idea is that when we subtract 1 from any number in binary we flip the bits from right to left until the first '1' then don't change the rest of the bits like this examples:



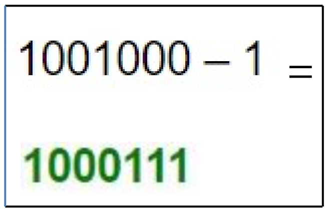
10101010 – 1 =

10101001



1000000 – 1 =

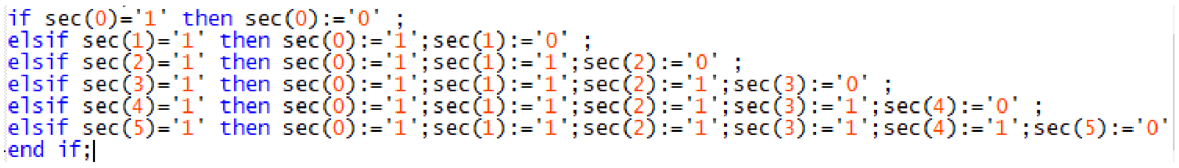
0111111



1001000 – 1 =

1000111

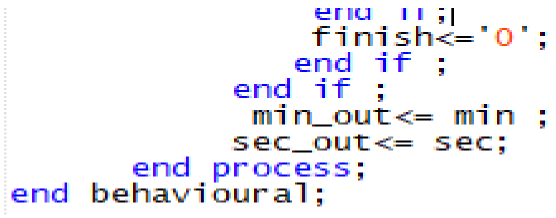
If the sec> "000000" then we must decrease it first by this block of codes (by the same idea above):



```
if sec(0)='1' then sec(0):='0' ;
elsif sec(1)='1' then sec(0):='1';sec(1):='0' ;
elsif sec(2)='1' then sec(0):='1';sec(1):='1';sec(2):='0' ;
elsif sec(3)='1' then sec(0):='1';sec(1):='1';sec(2):='1';sec(3):='0' ;
elsif sec(4)='1' then sec(0):='1';sec(1):='1';sec(2):='1';sec(3):='1';sec(4):='0' ;
elsif sec(5)='1' then sec(0):='1';sec(1):='1';sec(2):='1';sec(3):='1';sec(4):='1';sec(5):='0'
end if;
```

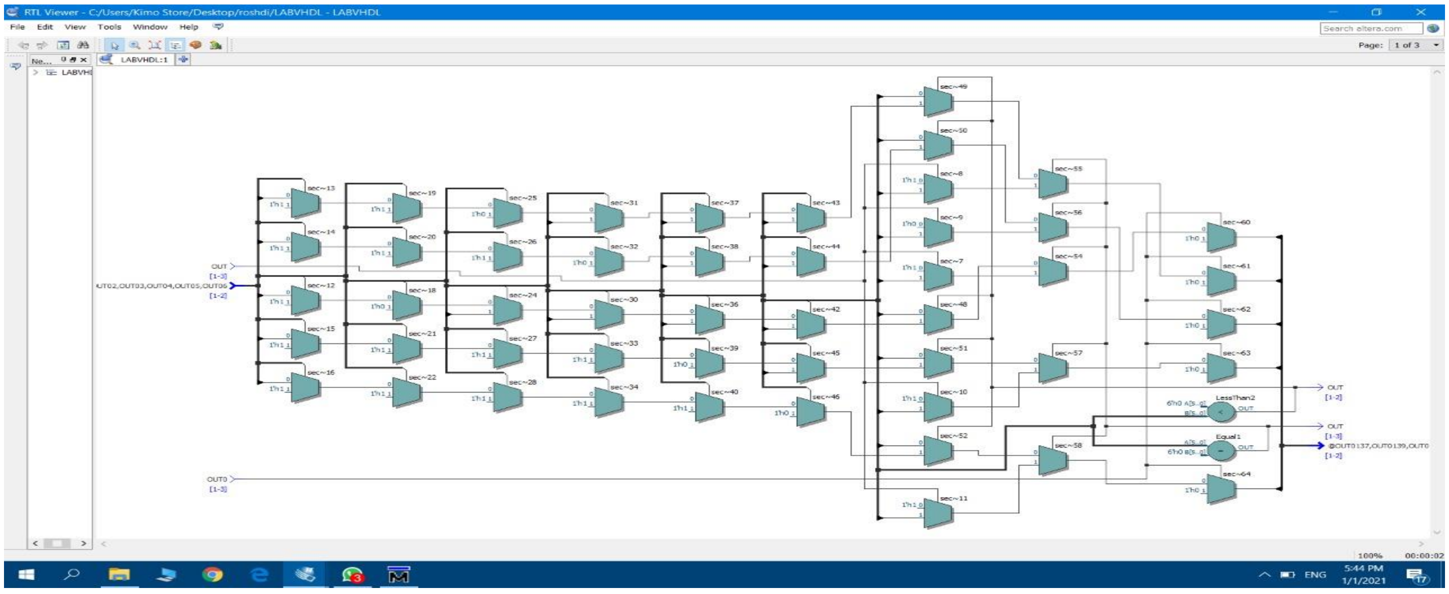
And so on until sec= "000000" we decrease mins by 1 and go to decrease sec until it be "000000" and the process still works until min and sec equals to "000000" so we done the work and finish='1' as we explain above

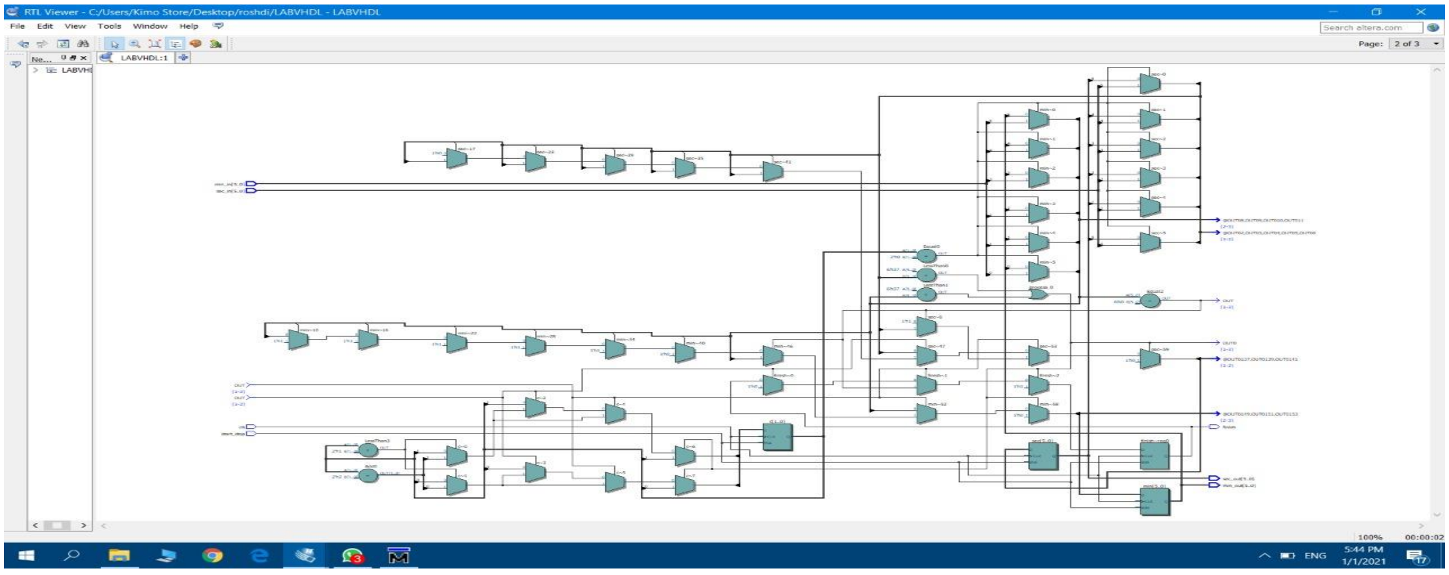
And in every stage we make min_out equals to min and sec_out equals to sec.

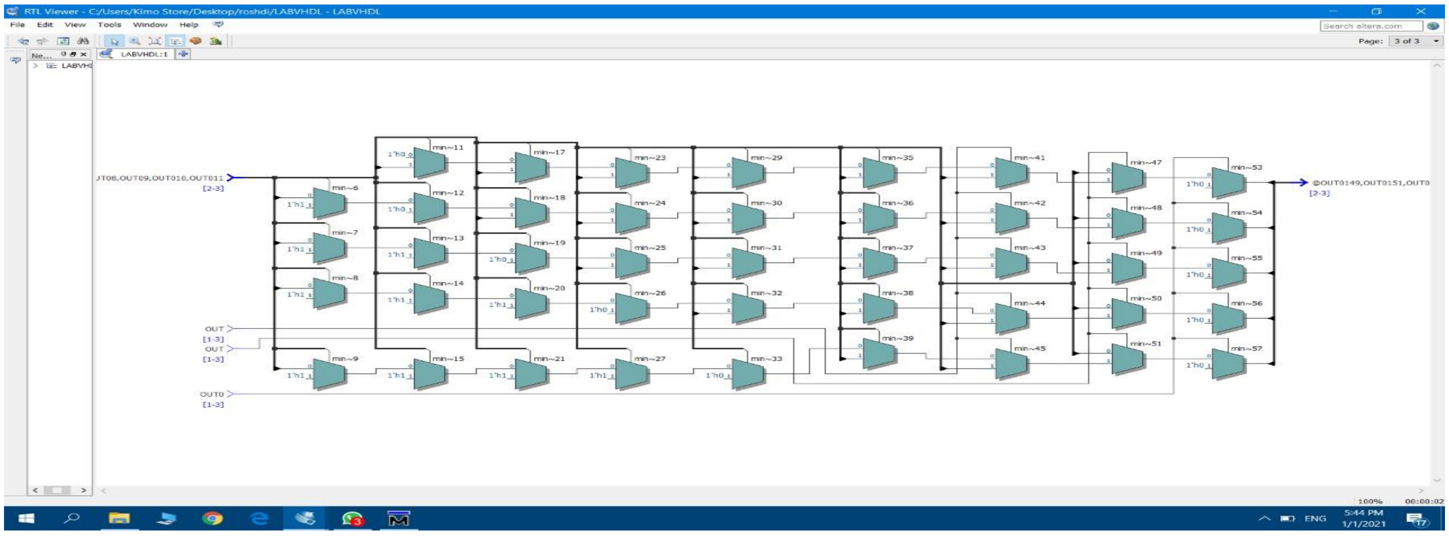


```
                    end if;
                    finish<='0';
                   end if ;
                end if ;
                 min_out<= min ;
                sec_out<= sec;
           end process;
end behavioural;
```

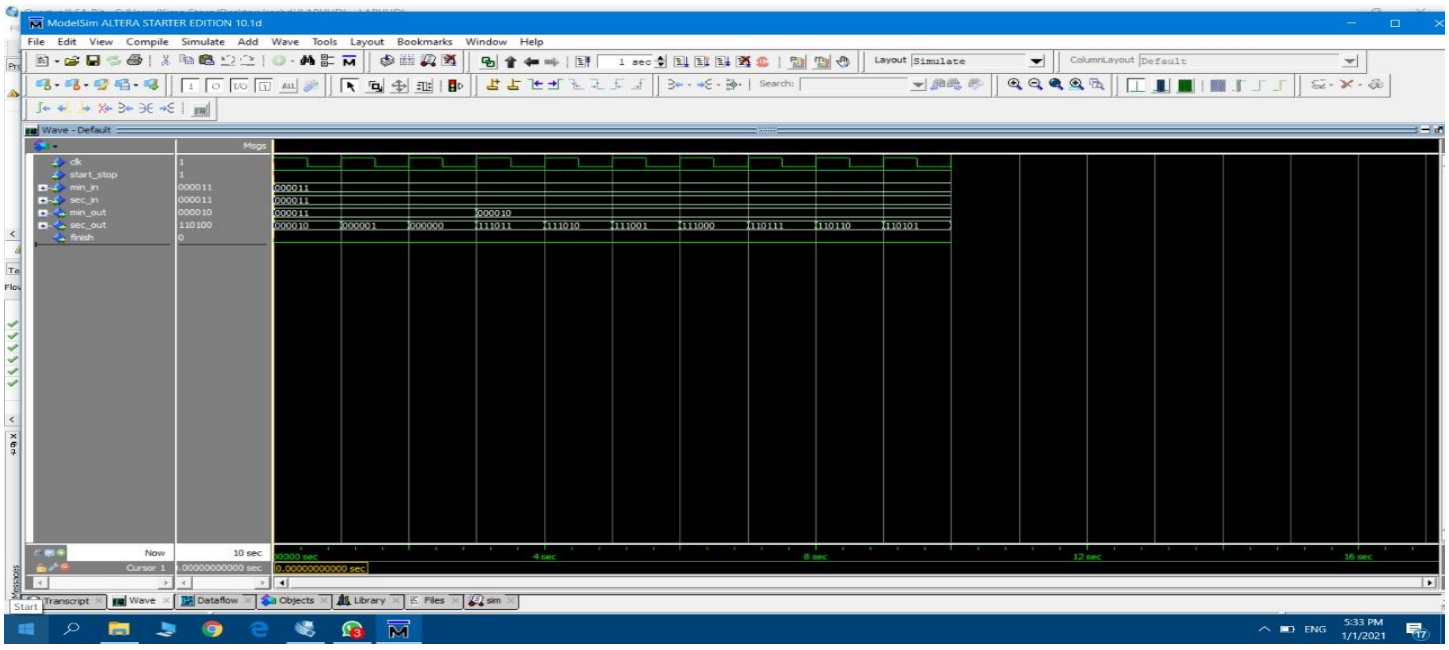
<u>The shematic:</u>




RTL Viewer - C:/Users/Kimo Store/Desktop/roshdi/LABVHDL - LABVHDL
File Edit View Tools Window Help
Page: 2 of 3
LABVHDL:1
100%
00:00:02
5:44 PM
1/1/2021

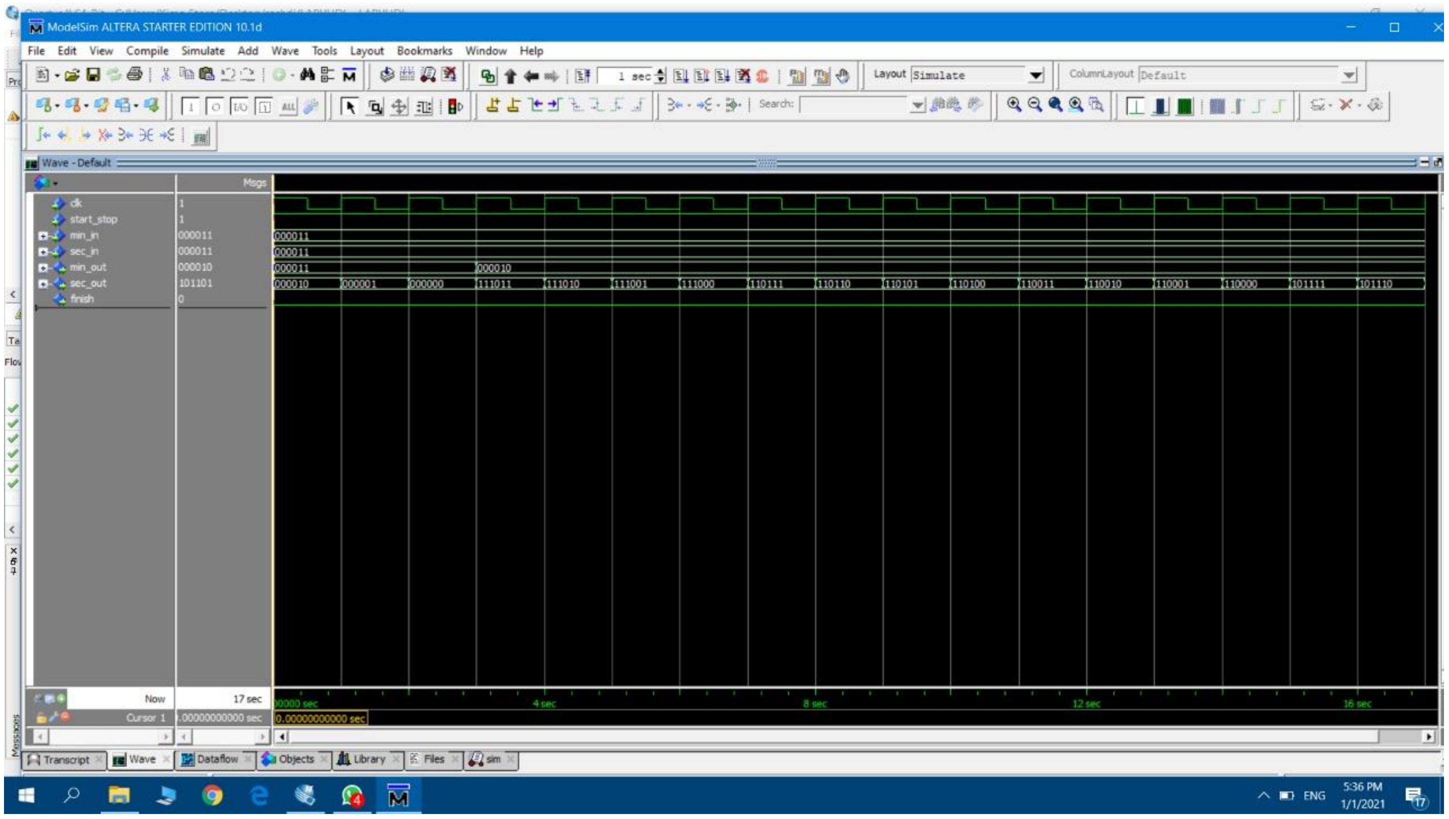

RTL Viewer - C:/Users/Kimo Store/Desktop/roshdi/LABVHDL - LABVHDL
File Edit View Tools Window Help
Page: 3 of 3
LABVHDL:1
OUT08,OUT09,OUT010,OUT011 [2-3]
min~6
min~11
min~53
@OUT0149,OUT0151,OUT0
OUT [1-3]
OUT0 [1-3]
100%
00:00:02
5:44 PM
1/1/2021

# MODELSIM SIMULATION:

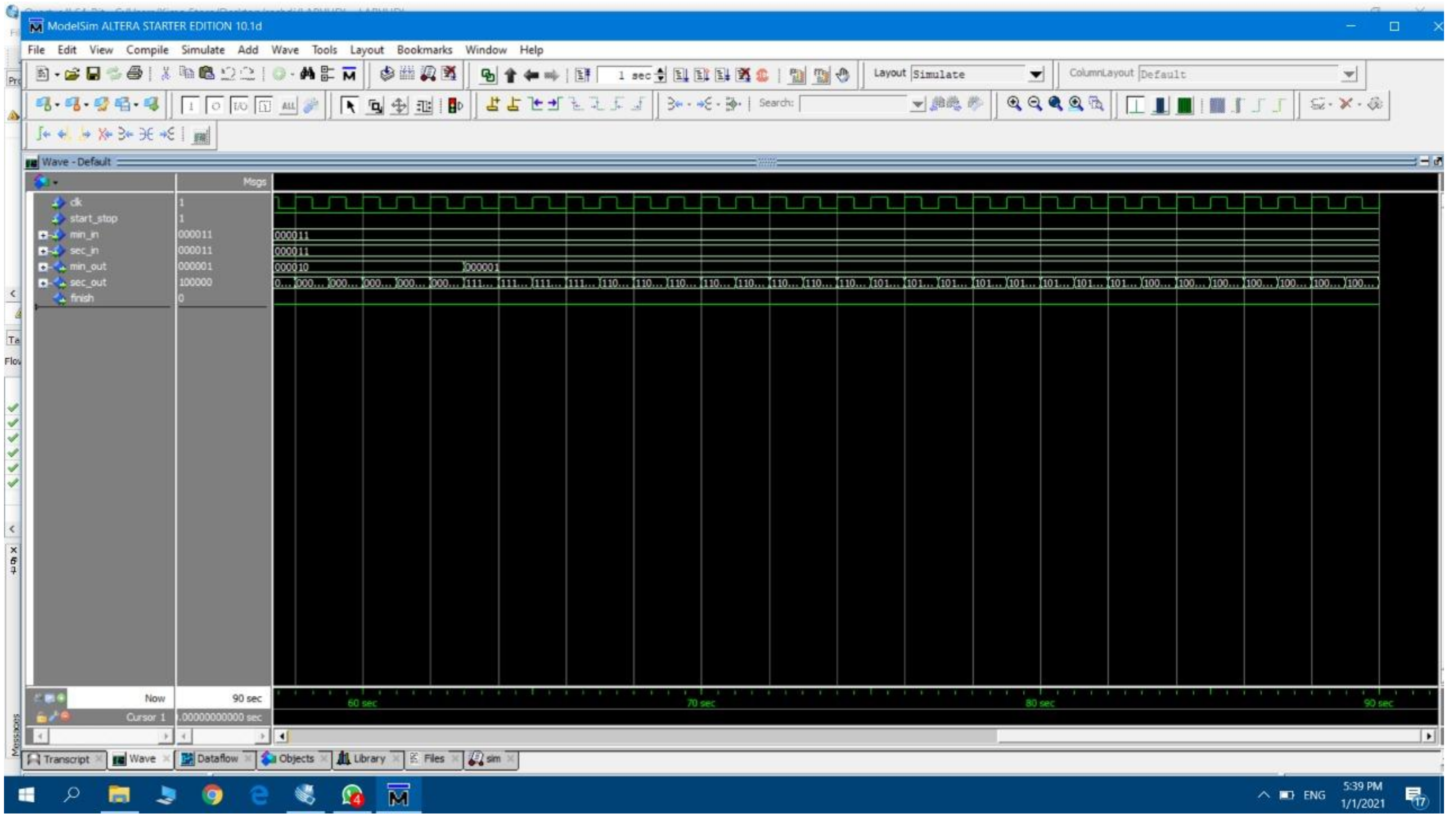


```
ModelSim ALTERA STARTER EDITION 10.1d
Wave - Default
                Msgs
clk             1
start_stop      1
min_in          000011   000011
sec_in          000011   000011
min_out         000010   000011                     000010
sec_out         110100   000010 000001 000000 111011 111010 111001 111000 110111 110110 110101
finish          0
Now             10 sec
Cursor 1        1.00000000000 sec   0.00000000000 sec
```

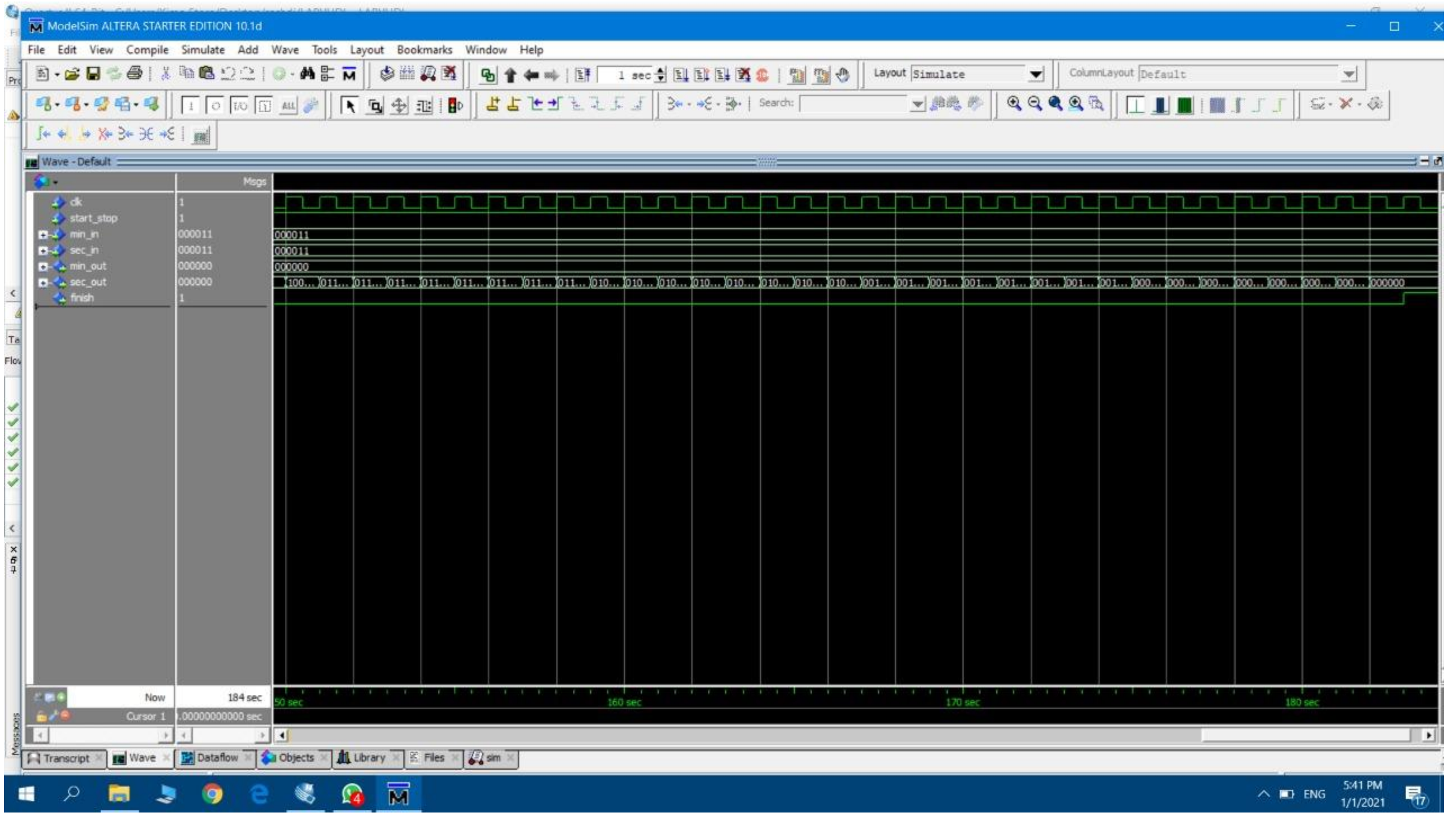
This screenshot explain how the seconds decrease by 1 in every step until reaching to zero then the mins decrease by 1 then the seconds begin to decrease as before and so on



```
ModelSim ALTERA STARTER EDITION 10.1d
Wave - Default
                Msgs
clk             1
start_stop      1
min_in          000011   000011
sec_in          000011   000011
min_out         000010   000011                     000010
sec_out         101101   000010 000001 000000 111011 111010 111001 111000 110111 110110 110101 110100 110011 110010 110001 110000 101111 101110
finish          0
Now             17 sec
Cursor 1        1.00000000000 sec   0.00000000000 sec
```

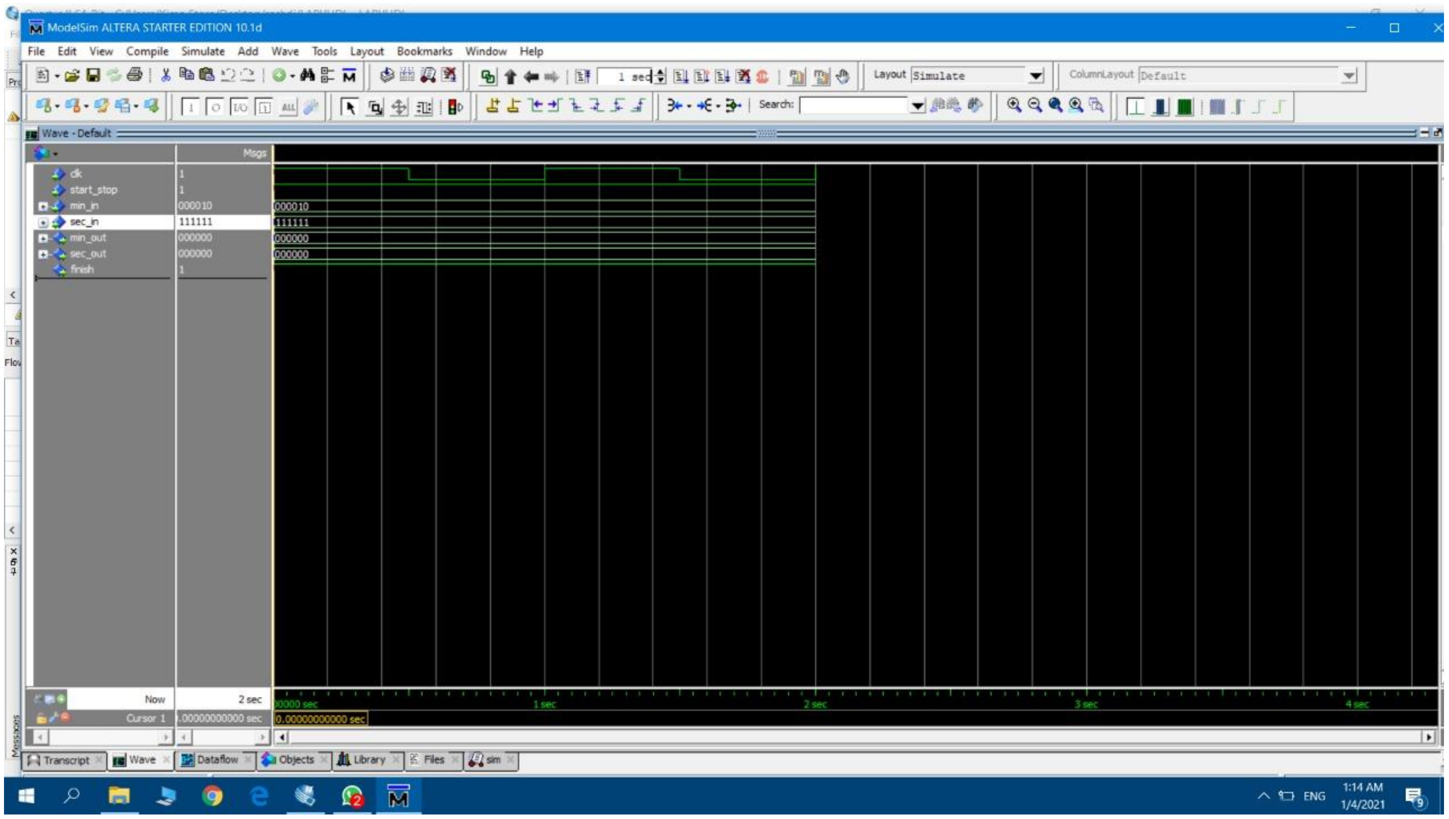


```
clk          1
start_stop   1
min_in       000011
sec_in       000011
min_out      000001
sec_out      100000
finish       0

Now          90 sec
Cursor 1     1.00000000000 sec
```



```
clk          1
start_stop   1
min_in       000011
sec_in       000011
min_out      000000
sec_out      000000
finish       1

Now          184 sec
Cursor 1     1.00000000000 sec
```

When the mins and seconds equal to ‘’000000’’… finish=1


ModelSim ALTERA STARTER EDITION 10.1d
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Wave - Default
Msgs
Now 2 sec
Cursor 1
Transcript Wave Dataflow Objects Library Files sim

| | Msgs |
|---|---|
| clk | 1 |
| start_stop | 1 |
| min_in | 000010 |
| sec_in | 111111 |
| min_out | 000000 |
| sec_out | 000000 |
| finish | 1 |

When we input mins or seconds > "111011" …outputs will be zeros and finish will be 1