A PROJECT REPORT
ON

Hardware Description Language

Named as "Smart Watch"

Submitted to

DR. VIPIN KAMBLE

In Partial Fulfilment of the Requirement for the Award of

B TECH. IN
ELECTRONICS AND COMMUNICATION ENGINEERING

BY

ASHUTOSH SINGH BT18ECE010
MUDIT GUPTA BT18ECE013
ADITYA JAISWAL BT18ECE022

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, NAGPUR

BSNL-RTTC,Near TV Tower,Beside Balaji Temple,
SEMINARY HILLS,NAGPUR-440006.

2019-2020

Index

Abstract:

The Project idea was inspired from the concept used in our smartphones to measure small periods of times that is stopwatch, timer and alarm. A stopwatch starts measuring/counting time from the moment the start button is pressed and it goes on until pause button is pressed, to start counting time again we can reset it by pressing the stop/reset button. For timer, we set enter a specific amount of time after which we want an alert that the specified time has passed. Functioning of an alarm is similar to that of timer but instead of alerting once it repeats forever until stopped to alert after the specified time has passed.

Only the simulation part could be achieved due to unavailability of components and hence its shown in simulation how the concept could be implemented.

Introduction:

For achieving the given aim, we are using Xilinx Vivado for simulation purposes which can be implemented on FPGA for practical model. The HDL used is VHDL or VHSIC HDL.

There are 3 .vhd scripts each for stopwatch, timer and RIA (Repeated Interval Alerts) respectively which are integrated as components.

There will be an external 1KHz clock which will be used for all time measurement purposes. This clock is made using IC 555.

Components Used:

For Simulation:

Xilinx Vivado ISE Desing Suite 14.7

For Practical Implementations:

Spartan FPGA board

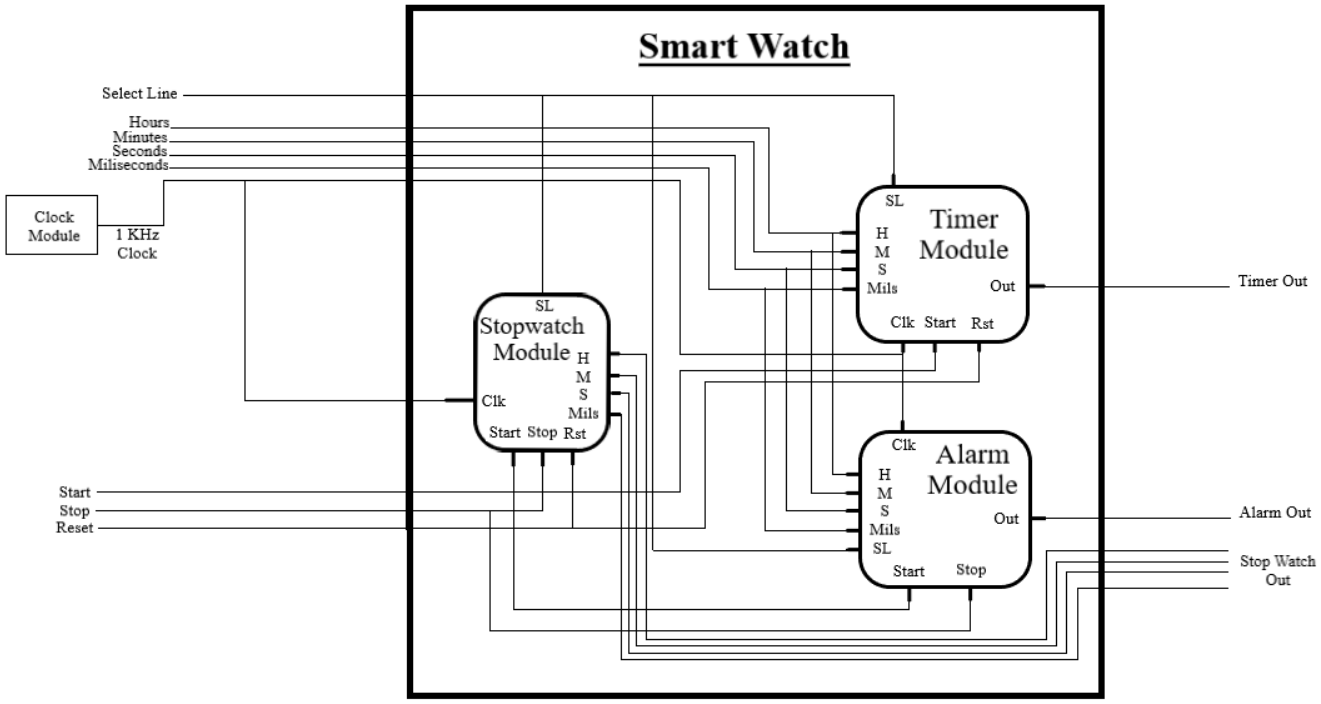
Numerical Number-pad

Push buttons

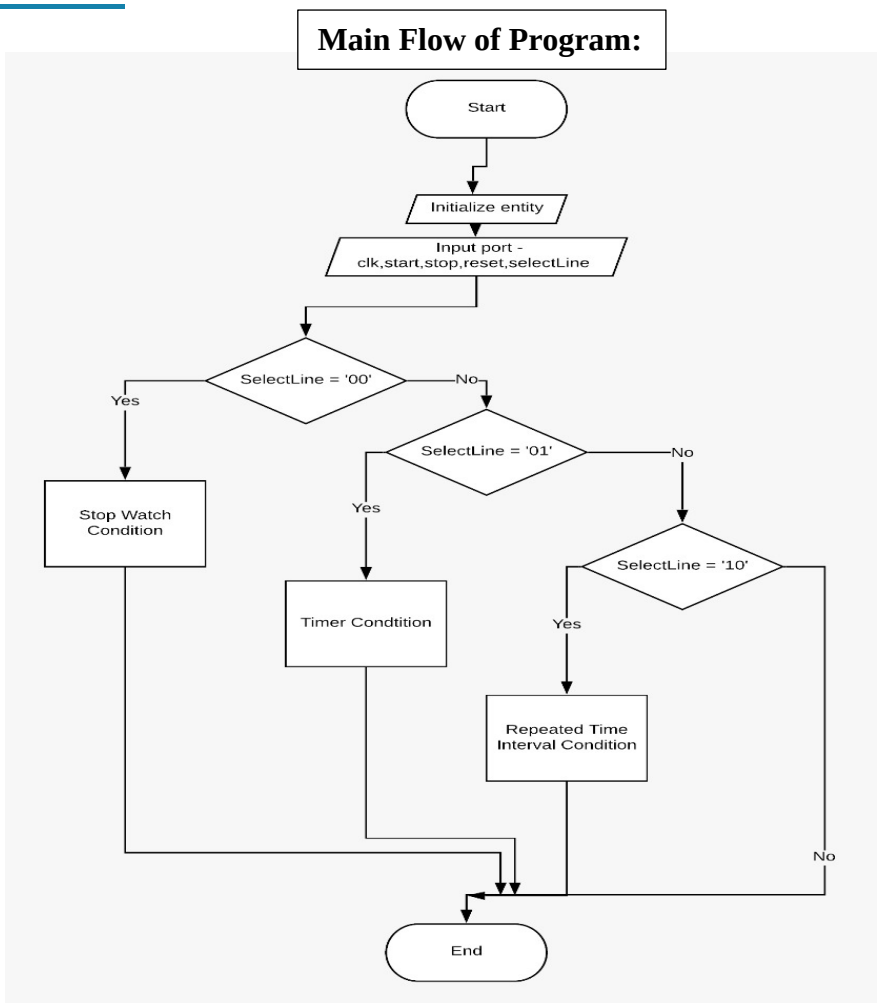
Wires

IC555 timer module

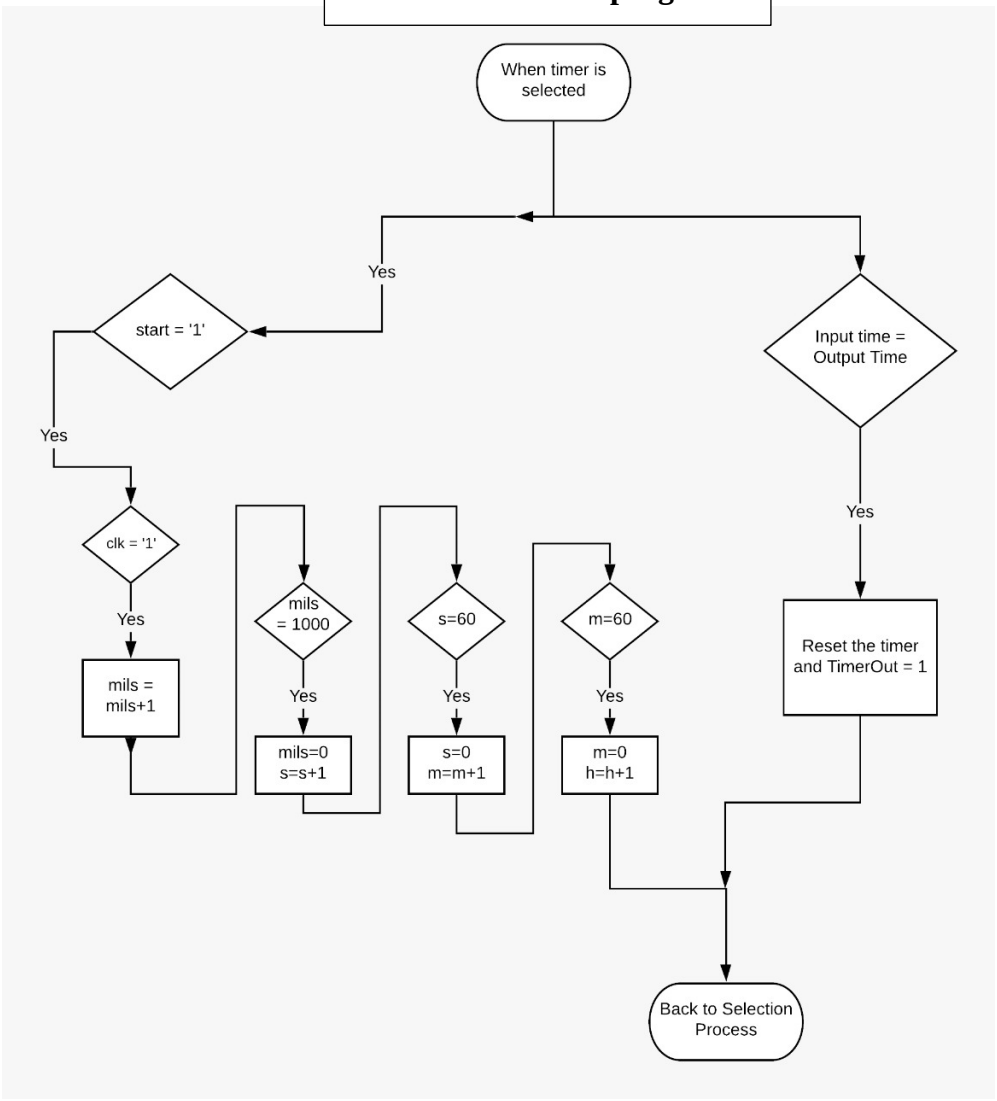
Block Diagram:



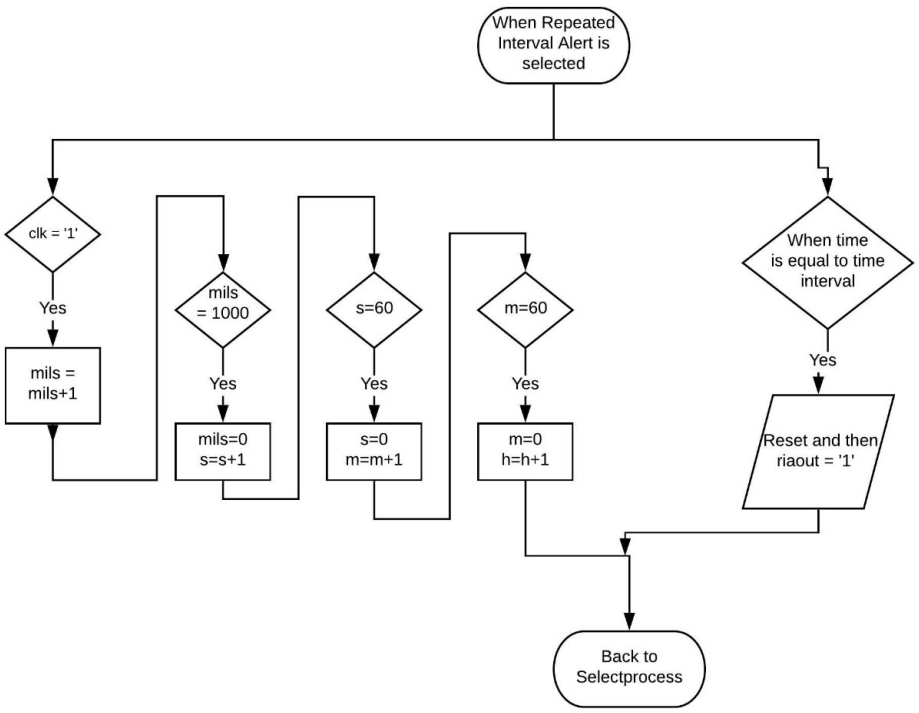
Flow Charts:



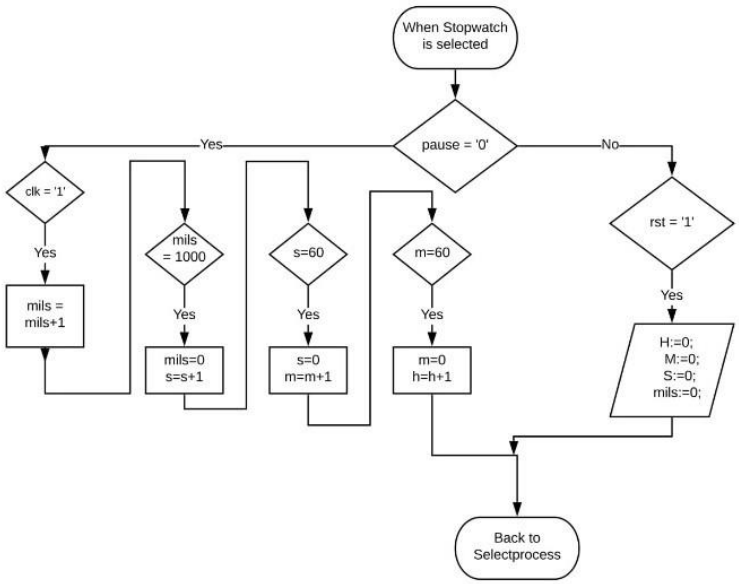
Flow for Timer subprogram:



Flow for Alarm subprogram:



Flow for Stopwatch subprogram:



Simulation Program:

```
-- Importing Libraries
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

--Entity Declaraction
entity Integration is
        port(clk,start,pause,rst: in bit;
        SelectLine:in bit_vector(1 downto 0);
        intime_H,intime_M,intime_S,intime_MIL,PulseDur:inout integer;
        Out_H,Out_M,Out_S,Out_Mils:out integer;
        TimerOut:out bit;
        RIAout:out bit);
end Integration;

--Archtiecture Declaration
architecture Behavioral of Integration is

shared variable H,M,S,mils: integer :=0;

begin
        intime_H<=0;
        intime_M<=0;
        intime_S<=5;
        intime_MIL<=500;
        process(clk)
                begin
                --Timer Selection
                if (SelectLine="01") then
                if(rst='1')then
                        H:=0;
                        M:=0;
```

```
                S:=0;
                mils:=0;
        end if;
        -- Starting Of timer
        if(start='1')then
                if (clk='1') then
                mils:=mils+1;
                end if;
                if (mils=1000) then
                        S:=S+1;
                        mils:=0;
                end if;
                if (S=60) then
                        M:=M+1;
                        S:=0;
                        mils:=0;
                end if;
                if (M=60) then
                        H:=H+1;
                        M:=0;
                        S:=0;
                        mils:=0;
                end if;
        end if;

        if (H=intime_H and M=intime_M and S=intime_S and
        mils=intime_MIL) then
                TimerOut<='1';
                TimerOut<='0' after 1000 ms;
                H:=0;
                M:=0;
                S:=0;
                mils:=0;
        end if;
end if;
-- Stopwatch Selection
if (SelectLine="00") then
        if (pause='0') then
                if (clk='1') then
                        mils:=mils+1;
                end if;
                if (mils=1000) then
                        S:=S+1;
                        mils:=0;
                end if;
                if (S=60) then
                        M:=M+1;
                        S:=0;
                        mils:=0;
                end if;
                if (M=60) then
                        H:=H+1;
                        M:=0;
                        S:=0;
                        mils:=0;
                end if;
        end if;
        if (rst='1') then
                H:=0;
                M:=0;
```

```
                        S:=0;
                        mils:=0;
                end if;
                Out_H<=H;
                Out_M<=M;
                Out_S<=S;
                Out_Mils<=mils;
        end if;
        -- Repeated Interval Alarm
        if (SelectLine="10") then

                if (clk='1') then
                        mils:=mils+1;
                end if;
                if (mils=1000) then
                        S:=S+1;
                        mils:=0;
                end if;
                if (S=60) then
                        M:=M+1;
                        S:=0;
                        mils:=0;
                end if;
                if (M=60) then
                        H:=H+1;
                        M:=0;
                        S:=0;
                        mils:=0;
                end if;
                if (H=intime_H and M=intime_M and S=intime_S and
mils=intime_MIL) then
                        RIAout <= '1';
                        RIAout <= '0' after 1000 ms;
                        H:=0;
                        M:=0;
                        S:=0;
                        mils:=0;
                end if;
        end if;
        end process;
        -- For select line selection
        process(SelectLine)
        begin
                H:=0;
                M:=0;
                S:=0;
                mils:=0;
        end process;

end Behavioral;
```

Program Explanation:

According to the block diagram, the select line decides which module will be active and will take the available inputs.

For stopwatch: On every rising edge of external 1KHz clock, the millisecond counter increases and for every 1000th millisecond, the second counter increments and millisecond counter resets, similarly for every 60th second minute counter increments resetting seconds and milliseconds, same goes for the hour counter. These increments take place only if input start signal is high. These increments in counters are also reflected in the output signals which are integers and hence give an output of current ongoing time. The counters and output resets to zero if input reset signal is toggled to high.

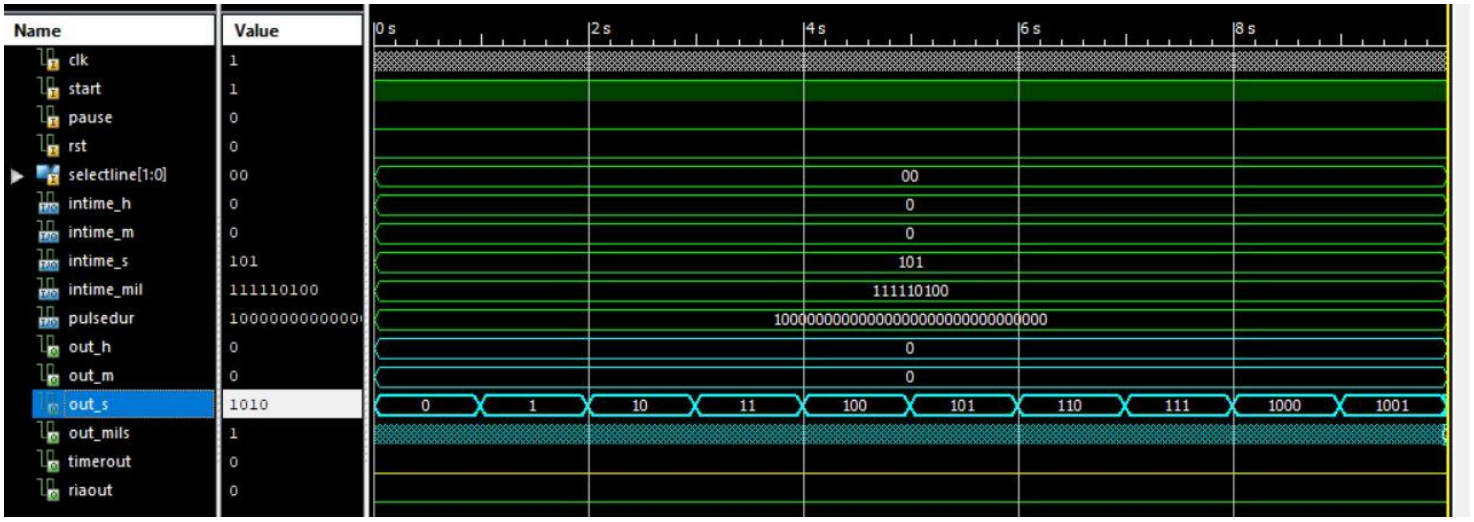
For Timer: On every rising edge of external 1KHz clock the hours, minutes, seconds, milliseconds counter increases just like in stopwatch if start is high, but it also checks whether the counters match the input values or the time when the timer should stop. If the condition of matching satisfies then the timer stops giving high on output signal for 1 second and then turns back to low. The counters and output resets to zero if input reset signal is toggled to high.

For Alarm: On every rising edge of external 1KHz clock the hours, minutes, seconds, milliseconds counter decreases until it reaches zero and then it gives high as an output for 1 seconds and then starts the counting again and goes on forever until stop is high.

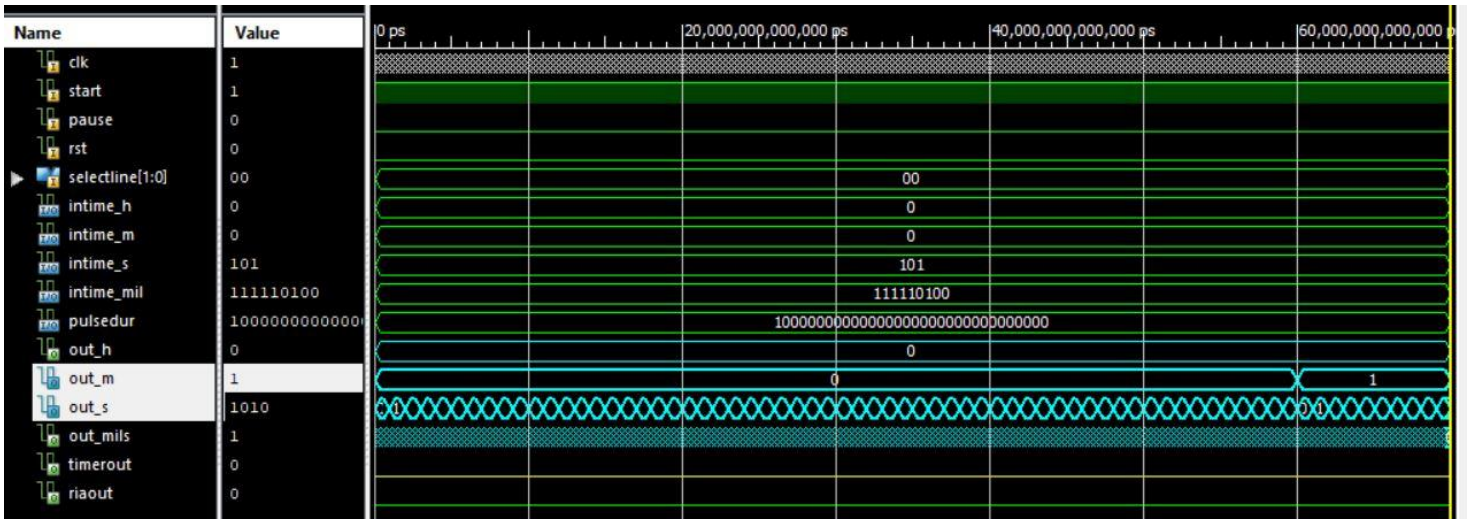
Results:

The results for stopwatch simulation are as follows:

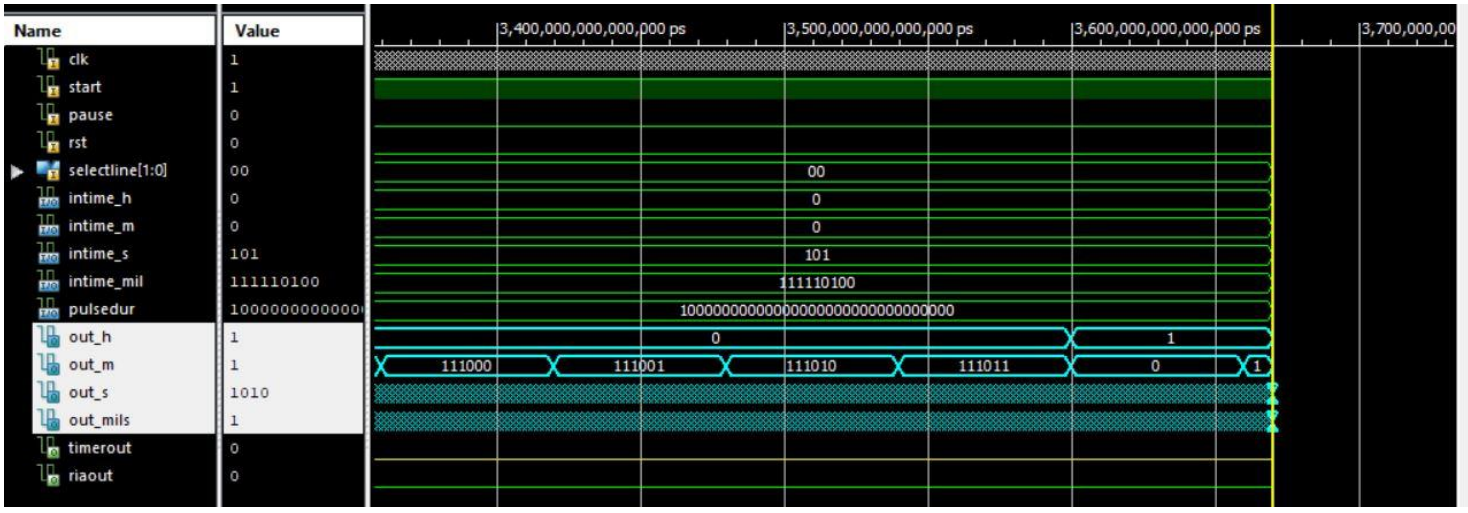
- When ran the simulation for 10 seconds:



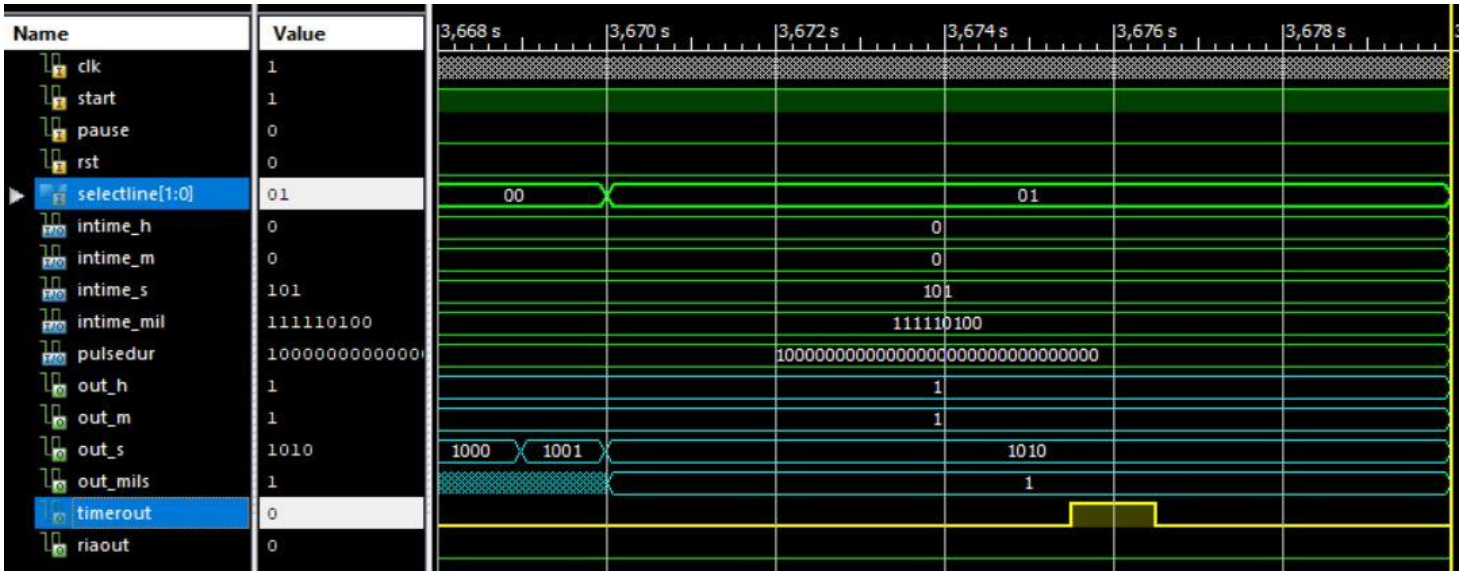
- When ran the simulation for more 1 minute:



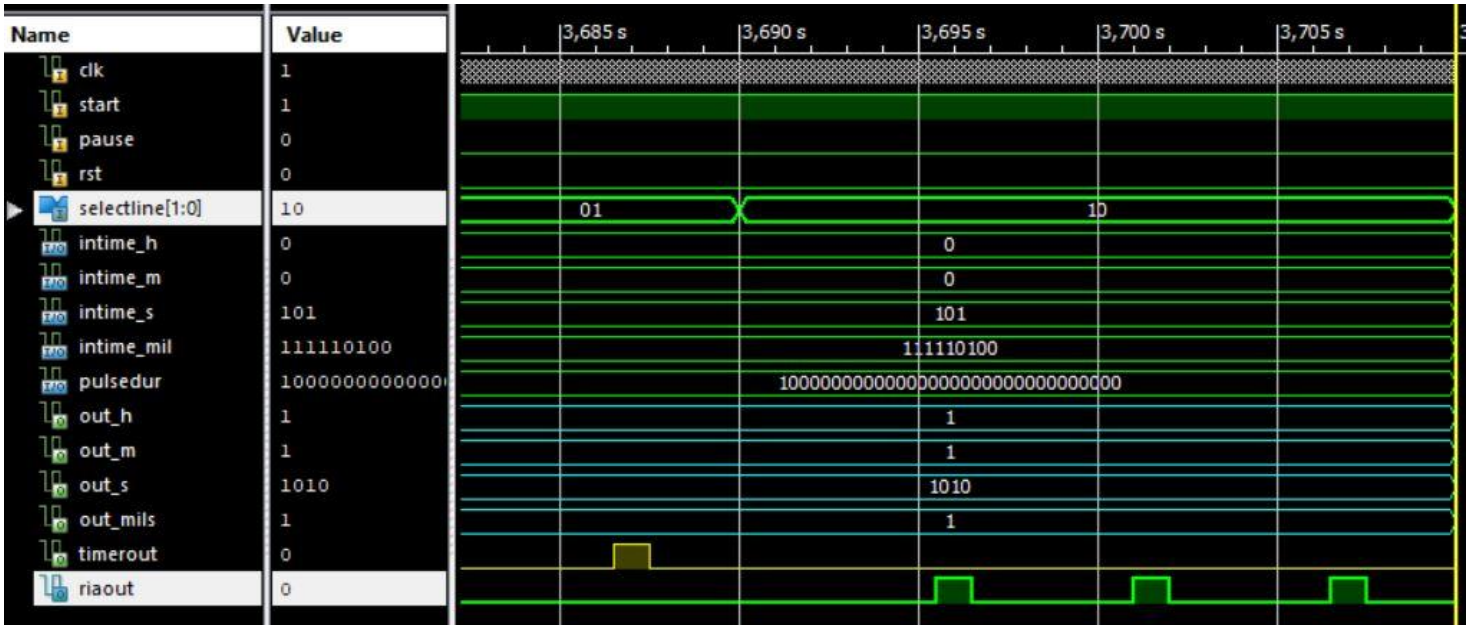
- When ran the simulation for 1 hour:



The results for Timer simulation is as follows:



The results for Alarm simulation are as follows:



SHORTCOMINGS AND FUTURE SCOPE:

Initially it was decided to use structural style of modelling but due to ineffective ways of handling condition of select line, the logical errors started to stack up.

The code for structural style modelling is as follow

```
1  ----------------------------------------------------------------------
2                              --SmartWatch--
3  --Functions:
4  --1. Timer.
5  --2. Stopwatch.
6  --3. Repeated Interval Alert.
7  ----------------------------------------------------------------------
8  library IEEE;
9  use IEEE.STD_LOGIC_1164.ALL;
10
11 entity SmartWatch is
12    port(intime_Hrs,intime_Min,intime_Sec,intime_MILs:inout integer;
13          Clk,Start,Stop,Reset:inout bit;
14          Select_Line:in bit_vector(1 downto 0);
15          Timer_Out, RIA_Out: out bit;
16          Stop_Out_H,Stop_Out_M,Stop_Out_S,Stop_Out_MIL:out integer);
17 end SmartWatch;
18
19 architecture Behavioral of SmartWatch is
20
21 component timer
22    port(rst,clk:in bit;
23       SelectLine:in bit_vector(1 downto 0);
24       start:inout bit;
25       intime_H,intime_M,intime_S,intime_MIL:inout integer;TimerOut:out bit);
26 end component;
27
28 component Stopwatchin
29       port( clk,start,pause,reset: in bit;
30          SelectLine:in bit_vector(1 downto 0);
31          Out_H,Out_M,Out_S,Out_Mils:out integer);
32 end component;
33
34 component RIA
```

```
35    port(clk:in bit;
36    SelectLine:in bit_vector(1 downto 0);
37    intime_H,intime_M,intime_S,intime_MIL:inout integer;
38    RIAout:out bit);
39 end component;
40
41 --signal intime_Hrs,intime_Min,intime_Sec,intime_MILs: integer;
42 --signal Clk,Start,Stop,Reset:bit;
43 --signal Timer_Out, RIA_Out: bit;
44 --signal Stop_Out_H,Stop_Out_M,Stop_Out_S,Stop_Out_MIL:integer);
45
46 begin
47 --    intime_Hrs<= intime_hrs when SelectLine="";
48 --    intime_Min<= intime_min when SelectLine="";
49    -- intime_Sec<= intime_sec when SelectLine="";
50       --intime_MILs<= intime_mILs when SelectLine="";
51 --    Clk<= Clck when SelectLine="";
52 --    Start<=Star_t when SelectLine="";
53 --    Stop<=Sto_p when SelectLine="";
54 --    Reset<=Rese_t when SelectLine="";
55 --    Timer_Out<=Timer_Ou_t when SelectLine="";
56 --    RIA_Out<=RIA_Ou_t when SelectLine="";
57 --    Stop_Out_H<=Stop_Ou_t_H when SelectLine="";
58 --    Stop_Out_M<=Stop_Ou_t_M when SelectLine="";
59 --    Stop_Out_S<=Stop_Ou_t_S when SelectLine="";
60 --    Stop_Out_MIL<=Stop_Ou_t_MIL when SelectLine="";
61
62       STP: Stopwatchin port map(
63       SelectLine=>Select_Line,
64       clk=>Clk,
65       start=>Start,
66       pause=>Stop,
67       reset=>Reset,
```

```
68          Out_H=>Stop_Out_H,
69          Out_M=>Stop_Out_M,
70          Out_S=>Stop_Out_S,
71          Out_Mils=>Stop_Out_MIL);
72
73          TMR: timer port map(
74          SelectLine=>Select_Line,
75          clk=>Clk,
76          start=>Start,
77          rst=>Reset,
78          intime_H=>intime_Hrs,
79          intime_M=>intime_Min,
80          intime_S=>intime_Sec,
81          intime_MIL=>intime_MILs,
82          TimerOut=>Timer_Out);
83
84          Alarm: RIA port map(
85          SelectLine=>Select_Line,
86          clk=>Clk,
87          intime_H=>intime_Hrs,
88          intime_M=>intime_Min,
89          intime_S=>intime_Sec,
90          intime_MIL=>intime_MILs,
91          RIAout=>RIA_Out);
92
93  end Behavioral;
```

However due to certain circumstances the errors couldn't be solved and we went for easier all in one approach where all the three modules are implemented using behavioural modelling in a single process.

This project has countless implementations, may it be this one alone or integrating this with some bigger circuits. So this project acts as a module which makes it easier for it to be used in any and every modular projects. Some examples are:

- Integrating it in some smart watch or DIY smartphone.
- Making an automated college bell which can also be programmed.
- Making any timed controlled circuit such as smart power sockets.

With some improvements and additions this project can also evolve in a modular RTC (Real Time Clock) which can be used for fetching live date and time which is essential for almost every VLSI based projects.

For reducing size GPS can also be incorporated for accurate date and time according to position on earth even with daylight savings.

However for all this the limitation is the size of an FPGA board, the smaller it is the better as it can be added in every possible circuit as size wont be a limitation there.

References:

1. stackoverflow.com
2. VHDL Primer by J.Bhasker
3. www.geeksforgeeks.com