EL 203: Embedded Hardware Design

Final Project : WristWatch using VHDL Module

Submitted to : Prof. Biswajit Mishra

Group 2:
(Question: 12)

Group Members:

Meet shah(201701143)

Yuvraj Raulji(201701144)

Dhruv Mendpara(201701146)

Avi Patel(201701147)

Jay Patel(201701148)

WristWatch using VHDL Module

Abstract - This is a multifunctional wristwatch project working in three modes: time, alarm and stopwatch. Report explains the logic behind the working of three wristwatch modules. Wristwatch works in a circular manner from time to alarm to stop-watch.

Introduction

In this, we will design a multifunction wristwatch that has time-keeping, alarm, and stopwatch functions. The wristwatch has three buttons (B1, B2, and B3) that are used to change the mode, set the time, set the alarm, start and stop the stopwatch, and so on. Pushing button B1 changes the mode from Time to Alarm to Stopwatch and back to Time. The functions of buttons B2 and B3 vary depending on the mode.The purpose of this project is to design the working wristwatch modelled on VHDL (Verilog Hardware Description Language) using Finite State Machine (FSM).

Block Diagram of Wristwatch Design

Figure shown below shows a block diagram for the design of a wristwatch. The input module divides the system clock down to a 100-Hz clock, CLK. It debounces the input buttons (PB1,PB2, and PB3) and synchronizes them with CLK. Each time PB1, PB2, or PB3 is pressed, the corresponding signal, B1, B2, or B3, will be 1 for exactly one clock time.The wristwatch module contains the main control for the wristwatch; the clock module, which implements the timekeeping and alarm functions; and the stopwatch module, which implements the stopwatch functions. The 100-Hz clock (CLK) synchronizes operation of the control unit and time registers.

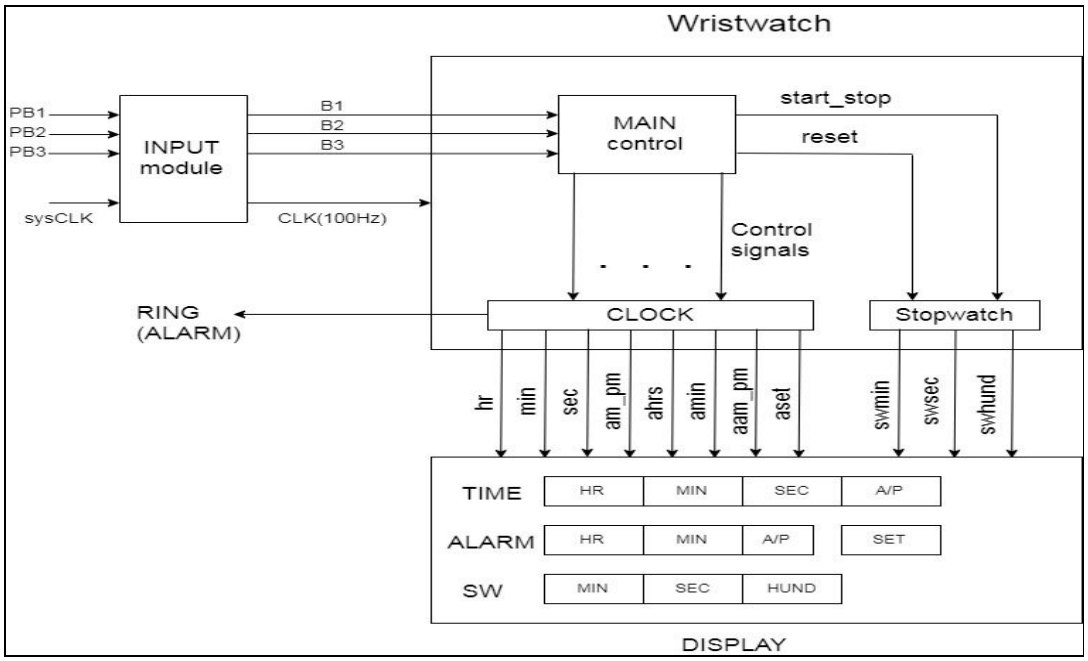

Fig 1:Block diagram

State Graph of WristWatch Module

Figure-2 shows the state graph for the controller. State machine generates the following control signals in response to pressing the buttons:

Inch :- it increments hours in the set_hours state
incm :- it increments minutes in the set_minutes state
alarm_off :- it turns off the alarm when it is ringing
incha :- it increments hours for the alarm
incma :- it increments minutes for the alarm
set_alarm :- it toggles the alarm set on and off
start_stop :- it starts or stops the stopwatch counter
reset :- it resets the stopwatch counter

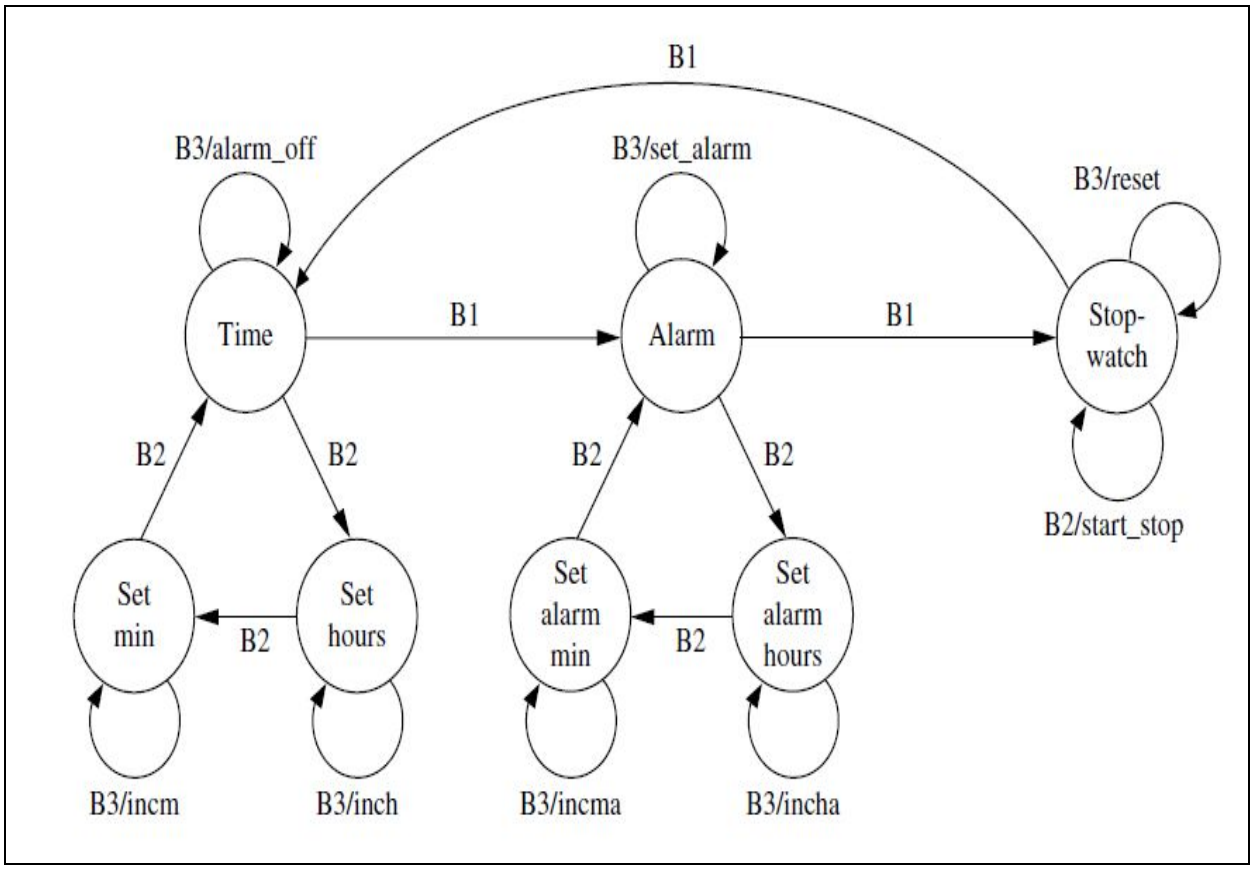

Fig 2: State Graph

As we can refer from FSM, there are total seven states in FSM. Time, Alarm and Stop-watch are three main states. At state Time if we press B1 then mode changes from Time to Alarm and again pressing B1 will change mode from Alarm to Stop-watch. In Time mode on pressing B3 we can set alarm off. In Alarm mode, on pressing B3 we can set alarm and in Stop_watch mode B3 will reset the stop-watch. There are two other loops of states for modes Time and Alarm. From Time mode, on pressing B2 state changes to Set hours of time and on pressing B2 again state changes to Set minutes of time and on pressing B2 again, state changes to default time state. Similar logic applies for alarm state. In each of the four states Set min, Set hours, Set alarm min and Set alarm hours, on pressing B3 corresponding values of minutes,hours,alarm minutes and alarm hours will be incremented.

State Machine Chart for WristWatch Module

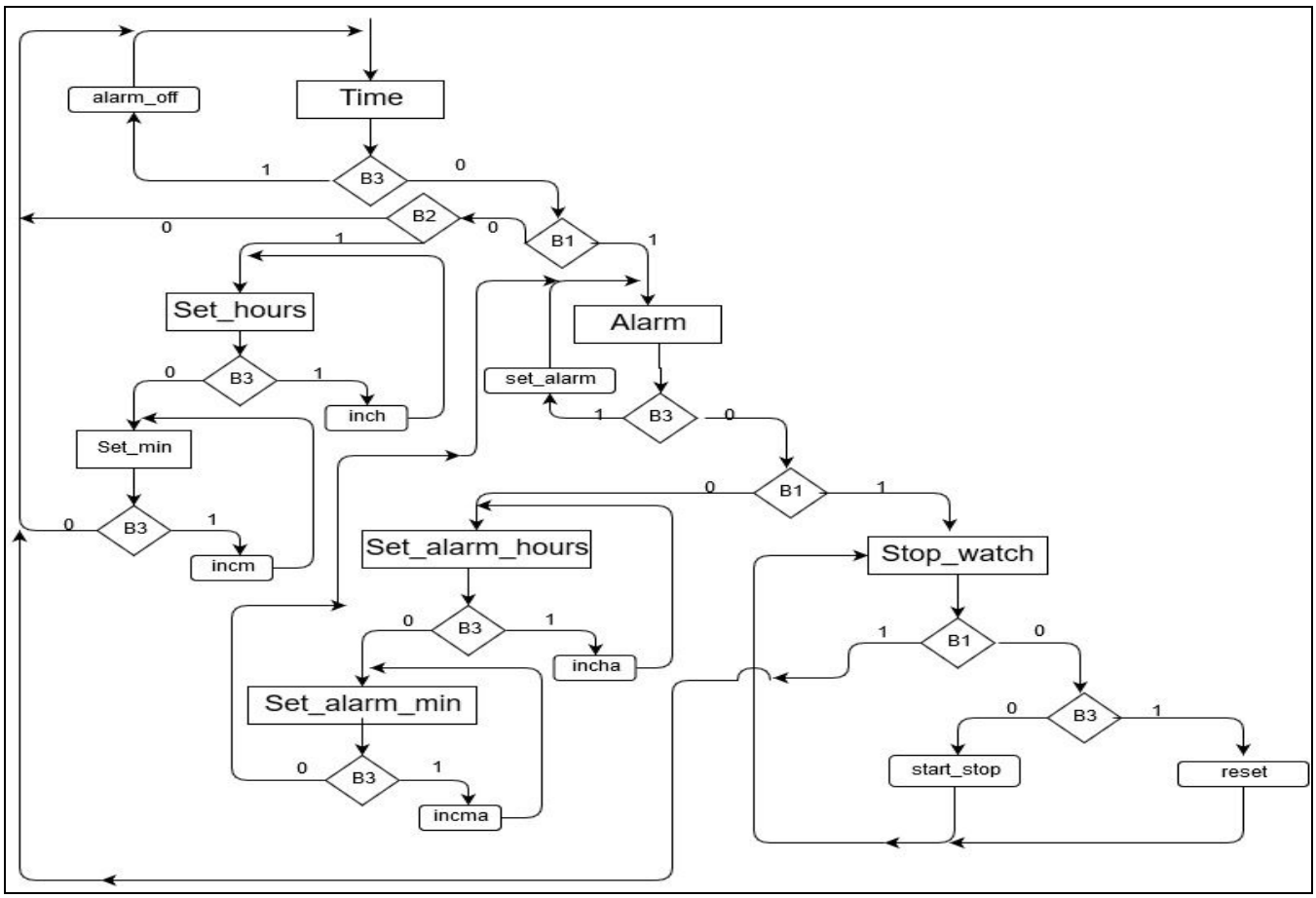

Fig 3: SM Chart

VHDL code Snippets

Following are some of the signal names used in the VHDL code:

am_pm :- it is A.M. or P.M. in time mode
alarm_set :- it indicates that alarm is set
time counters,if alarm is set
ahours :- hours in alarm mode minutes minutes in the time mode
swhundredths :- hundredths of a second during stopwatch mode

aam_pm :- it is A.M. or P.M. in alarm mode
ring :- it indicates that alarm setting matches
hours :- hours in time mode
aminutes :- minutes in the alarm mode
seconds :- seconds in the time mode
swseconds :- seconds in stopwatch mode
swminutes :- minutes in stopwatch mode

```
when time1 =>
   if B1 = '1' then nextstate <= alarm;
   elsif B2 = '1' then nextstate <= set_hours;
   else nextstate <= time1;
   end if;
   if B3 = '1' then alarm_off <= '1';
   end if;
   disp1_input <= minutes & seconds;
```

```
when set_hours =>
   if B3 = '1' then inch <= '1'; nextstate <= set_hours;
   else nextstate <= set_hours;
   end if;
   if B2 = '1' then nextstate <= set_min;
   end if;
   disp1_input <= minutes & seconds;
```

```
when set_min =>
   if B3 = '1' then incm <= '1'; nextstate <= set_min;
   else nextstate <= set_min;
   end if;
   if B2 = '1' then nextstate <= time1;
   end if;
   disp1_input <= minutes & seconds;
```

```
when alarm =>
   if B1 = '1' then nextstate <= stop_watch;
   elsif B2 = '1' then nextstate <= set_alarm_hrs;
   else nextstate <= alarm;
   end if;
   if B3 = '1' then set_alarm <= '1'; nextstate <= alarm;
   end if;
   disp1_input <= ahours & aminutes;
```

```
when set_alarm_hrs =>
   if B2 = '1' then nextstate <= set_alarm_min;
   else nextstate <= set_alarm_hrs;
   end if;
   if B3 = '1' then incha <= '1';
   end if;
   disp1_input <= ahours & aminutes;
```

```
when set_alarm_min =>
   if B2 = '1' then nextstate <= alarm;
   else nextstate <= set_alarm_min;
   end if;
   if B3 = '1' then incma <= '1';
   end if;
   disp1_input <= ahours & aminutes;
```

```
when stop_watch =>
   if B1 = '1' then nextstate <= time1;
   else nextstate <= stop_watch;
   end if;
   if B2 = '1' then start_stop <= '1';
   end if;
   if B3 = '1' then reset <= '1';
   end if;
   disp1_input <= swminutes & swseconds;
```

Problems faced during implementation and their solutions:

- The clock frequency used in code for software implementation of wristwatch was not matching with the default clock frequency of the hardware (FPGA). So we resolved this issue by slowing down the clock and matching the hardware clock frequency.
- Another small challenge faced during hardware implementation of wristwatch was mapping of seven segment LED display.
- Also there was an issue in switching between different states of wristwatch, the state changed very fast because of clock frequency mismatch. which was finally solved by using separate code for debouncing.

Work Distribution:

Meet shah(201701143): 20% (State Diagram, Report, Hardware Implementation)
Yuvraj Raulji(201701144): 20% (Block Diagram, Coding Part)
Dhruv Mendpara(201701146): 20% (Report, Coding Part)
Avi Patel(201701147): 20% (SM Chart, Report,Hardware Implementation)
Jay Patel(201701148): 20% (Coding Part, Necessary changes and improvisation in Code)

References:

1) Lizy Kurian John ,Charles H Roth Jr. “Digital Systems Design using VHDL”.
2) GitHub. Wrist Watch VHDL. url: https://github.com/BreadDan/wristwatch-in-VHDL.