FPGA LAB REPORT 7

Hassan Ahmed, Hamza Shabbir & Neha Chaudhary

JULY 28, 2021

Contents

Implementation of Digital Watch on LCD

Objectives:

- Designing of block diagram using LCD driver core and counter instances with connections.
- Implementation of behavioral abstractions of digital watch with LCD
- Generate synthesis model and test the digital watch on LCD.

Background:

ModelSim is a multi-language HDL simulation environment by Mentor Graphics, for simulation of hardware description languages such as VHDL, Verilog and SystemC, and includes a built-in C debugger. ModelSim can be used independently, or in conjunction with Intel Quartus Prime, Xilinx ISE or Xilinx Vivado.

The Spartan-3E family builds on the success of the earlier Spartan-3 family by increasing the amount of logic per I/O, significantly reducing the cost per logic cell. New features improve system performance and reduce the cost of configuration. These Spartan-3E FPGA enhancements, combined with advanced 90 nm process technology, deliver more functionality and bandwidth per dollar than was previously possible, setting new standards in the programmable logic industry.

Because of their exceptionally low cost, Spartan-3E FPGAs are ideally suited to a wide range of consumer electronics applications, including broadband access, home networking, display/projection, and digital television equipment.

In the previous lab seconds part was only implemented because of limited LEDs on the spartan-3E board. In this lab we will implement the whole digital watch and display it on the LCD. The top level module will contain the drivers for the LCD and will also call the digital watch module. The output of the digital watch module will be displayed on specific positions on LCD defined by us.

Procedure:

1. First we designed the block diagram using LCD driver core and counter instances and mentioned the intermediate connections.
2. After that we implemented the behavioral abstraction of digital watch with LCD.
3. At the end we generated the synthesis model and uploaded the bit file on the board to test the digital watch.

Block Diagram:

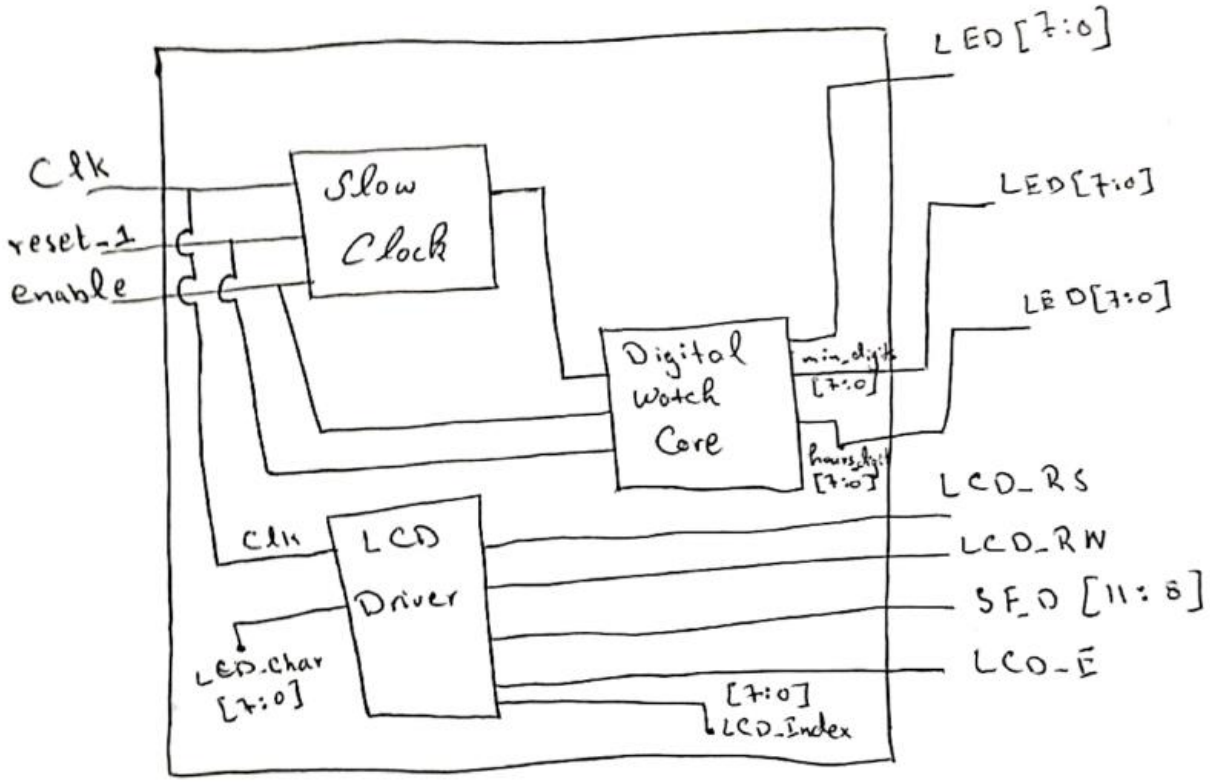

Figure 1: Digital Watch LCD Core

Verilog Code:

Top Level:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    14:48:32 06/29/2021
// Design Name:
// Module Name:    LCDTopModule
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
```

```
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
`timescale 1ns/1ps
module digital_watch_lcd_core (// Port List
CLK,
reset_1 ,
enable,
LED,
LCD_RS ,
LCD_RW ,
LCD_E,
SF_D,
SF_CE0
);
//---- Port Declarations ----
input CLK;
input reset_1 ;
input  enable;
output [7:0] LED;
output LCD_RS ;
output LCD_RW ;
output LCD_E;
output [11:8] SF_D;
output SF_CE0 ;
//-- Intermediate signals declaration
reg [7:0] LcdChar ;
wire [7:0] LcdIndex ;
wire [3:0] seconds_least, seconds_most,min_least,min_most,hours_least,hours_most ;
assign SF_CE0 = 1'b1;
assign LED = {min_least,min_most} ;
always @(LcdIndex or  seconds_least or seconds_most or min_least or min_most or
hours_least or hours_most)
begin
case (LcdIndex )
8'h00 : LcdChar = 8'h54; //-- char T (First line 1st char)
8'h01 : LcdChar = 8'h69; //-- char i
8'h02 : LcdChar = 8'h6d; //-- char m
8'h03 : LcdChar = 8'h65; //-- char e
8'h04 : LcdChar = 8'h3d; //-- char =
```

```
8'h05 : LcdChar = 8'h20; //-- char ''
8'h06 : LcdChar = {4'b0011 , hours_most [3:0]}; //-- Hours
8'h07 : LcdChar = {4'b0011 , hours_least [3:0]};
8'h08 : LcdChar = 8'h3a; //-- char :
8'h09 : LcdChar = {4'b0011 , min_most [3:0]}; //-- Min
8'h0a : LcdChar = {4'b0011 , min_least [3:0]};
8'h0b : LcdChar = 8'h3a; //-- char :
8'h0c : LcdChar = {4'b0011 , seconds_most [3:0]}; //-- Sec
8'h0d : LcdChar = {4'b0011 , seconds_least [3:0]};
8'h40 : LcdChar = 8'h50; //-- char P (second line 1st char)
8'h41 : LcdChar = 8'h41; //-- char A
8'h42 : LcdChar = 8'h4B; //-- char K
8'h43 : LcdChar = 8'h49; //-- char I
8'h44 : LcdChar = 8'h53; //-- char S
8'h45 : LcdChar = 8'h54; //-- char T
8'h46 : LcdChar = 8'h41; //-- char A
8'h47 : LcdChar = 8'h4E; //-- char N
default : LcdChar = 8'b00000000 ;
endcase
end
//-- Instantiation of the LCD Display driver
lcd_driver i_lcd_driver (// Port Connections
. Clk ( CLK ),
. rs ( LCD_RS ),
. rw ( LCD_RW ),
. enable ( LCD_E ),
. lcd_data ( SF_D ),
. index ( LcdIndex ),
. char ( LcdChar )
);
//-- Instantiation of the Digital Watch Core
digital_watch i_digital_watch_core (// Port Connections
. clk ( CLK),
. reset_1 ( reset_1 ),
. enable ( enable),
. seconds_least ( seconds_least ),
.seconds_most(seconds_most),
. min_least ( min_least ),
.min_most(min_most),
. hours_most ( hours_most),
.hours_least(hours_least)
);
endmodule
```

Digital Watch Module:

```
module counter (clk, reset, enable, count_val, over_flow);
input clk, reset, enable;
parameter Bits=4;
parameter Max=9;
output[Bits-1:0] count_val;
output over_flow;
reg[Bits-1:0] count_val;
reg over_flow;

always@(posedge clk)
if(reset) begin
count_val=0;
over_flow=0;
end
else if(enable)
if(count_val==Max) begin
count_val=0;
over_flow=1;
end
else begin

count_val=count_val+1;
over_flow=0;
end
endmodule

module digital_watch(clk, reset_1, enable, seconds_most,
seconds_least,min_least,min_most,hours_least,hours_most);
input clk, reset_1, enable;
output[3:0] seconds_least, seconds_most, min_least, min_most, hours_least, hours_most;
wire w1,w2,w3,w4,w5,w6, reset;
reg reset_2;
assign reset=reset_1 || reset_2;
always@(posedge clk) begin
if(hours_least==4'd4 && hours_most==4'd2)
reset_2=1'b1;
else
reset_2=1'b0;
end
counter #(32,50000000) C0(.clk(clk), .reset(reset), .enable(enable), .over_flow(w1));
```

```
counter #(4,10) C1(.clk(clk), .reset(reset), .enable(w1), .count_val(seconds_least[3:0]),
.over_flow(w2));
counter #(4,6) C2(.clk(clk), .reset(reset), .enable(w2), .count_val(seconds_most[3:0]),
.over_flow(w3));
counter #(4,10) C3(.clk(clk), .reset(reset), .enable(w3), .count_val(min_least[3:0]),
.over_flow(w4));
counter #(4,6) C4(.clk(clk), .reset(reset), .enable(w4), .count_val(min_most[3:0]),
.over_flow(w5));
counter #(4,10) C5(.clk(clk), .reset(reset), .enable(w5), .count_val(hours_least[3:0]),
.over_flow(w6));
counter #(4,6) C6(.clk(clk), .reset(reset), .enable(w6), .count_val(hours_most[3:0]));
endmodule
```

User Constraint File:

```
NET "clk" LOC = "C9" | IOSTANDARD = LVCMOS33 ;
NET "clk" PERIOD = 20.0ns HIGH 50%;

NET "LCD_E" LOC = "M18" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LCD_RS" LOC = "L18" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LCD_RW" LOC = "L17" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;


NET "SF_D<8>" LOC = "R15" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "SF_D<9>" LOC = "R16" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "SF_D<10>" LOC = "P17" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "SF_D<11>" LOC = "M15"        | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;


NET "enable" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP ;
NET "reset_1" LOC = "L14" | IOSTANDARD = LVTTL | PULLUP ;


NET "LED<0>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<1>" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<2>" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<3>" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<4>" LOC = "C11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<5>" LOC = "D11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<6>" LOC = "E9"    | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<7>" LOC = "F9"    | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
```

Discussion and Conclusion:

The 50 MHz clock from the Spartan 3E Board goes to the counter in the digital watch module. The counter devides the clock & gives the output as 1 Hz clock which derives the least significant bit of seconds in the digital watch module whose overflow drives the most significant digit of the seconds & similarly minutes & hours are drived. The outputs of the digital watch module is given to the LCD inputs to be displayed. LCD_char tells what to write & LCD_Index tells where to write on the LCD. LCD_RS provides the register selection. LCD_E is the read/write enable pulse & LCD_RW is the read/write control. All the characters to be written are stored in a display Data RAM before displayed on the 16x2 LCD.