Ho Chi Minh City University of Technology
FACULTY OF COMPUTER SCIENCE & ENGINEERING

LABOTARY REPORT

Logic Design with HDL

Assignment: SPORT STOPWATCH

Group 4

Nguyễn Đức Bảo Huy – 2152089

Hoàng Tiến Đức – 2152520

SPORT STOPWATCH

I. AIMS/PURPOSES

Practice and improve the ability to use Verilog HDL hardware to design digital logic circuit.

- Training skills in analyzing and designing digital circuits according to the hierarchical model: split functions, modules, … Enhance creativity in designing application products.
- Practice research skills, self-study, gain more knowledge of related fields: techniques in common digital circuits, fields of digital system applications, …

II. PROJECT INTRODUCTION – SPORT STOPWATCH

1. Introduction

Stopwatch is widely used in practice, sports competitions such as athletics, swimming or can be applied in counting time of the job or producing process.

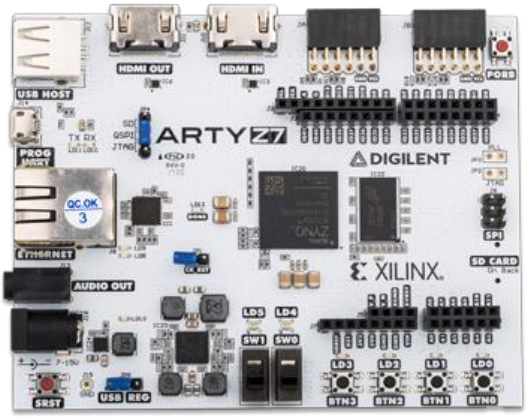
2. Requirement – Functions

Use your knowledge of Logic Design with HDL and related subjects to create, build a sport stopwatch on Arty Z7 FPGA development kit or equivalent.

- The system can count up to the maximum units of time is the maximum displayed of LED of units of minutes, the minimum unit of time is hundredth of second. Each unit of time is displayed on 2 leds 7-segment.
- The system is able to count down to the maximum units of time is the maximum displayed of LED of units of minutes, the minimum unit of time is hundredth of second. When down counter completes, the system has LED effect to notice.
- The system allows pause time, continue counting up/down from the timeline paused, and reset back to 00:00:00.
- The system can modify initial time, and re-set up time for counting.

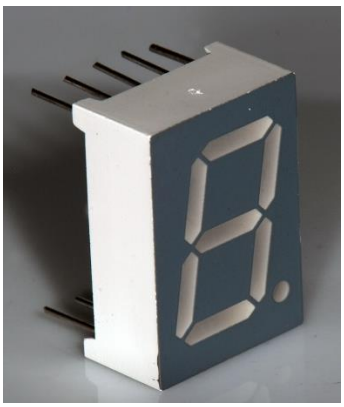
3. Apparatus

- Kit FPGA Arty Z7 within Vivado



- 2 Pro-ject Boards GL No.12

- 6 Led 7-segments



- Resistances
- Wires

4. User manual

- 2 switches 0 & 1 control up counter and down counter:
    - SW0 = 0, SW1 = 0 : Reset stopwatch to 00:00:00.
    - SW0 = 0, SW1 = 1 : Up counter.
    - SW0 = 1, SW1 = 1 : Stop temporarily / Pause.
    - SW0 = 1, SW1 = 0 : Down counter.
    - LD4 : When the system countdown to 00:00:00, LD4 will glow.
- 4 buttons 0 & 1 & 2 & 3 are set up to import initial time to count down. When a button is pressed 1 time, the value count up 1.
    - BTN 3 : adjust the tens of minute.
    - BTN 2 : adjust the units of minute.
    - BTN 1 : adjust the tens of second.
    - BTN 0 : adjust the units of second.

III. WORKING PRINCIPLE

1. Block diagram

- Stopwatch is able to count up, count down based on the following block diagrams:

a. Count up block diagram

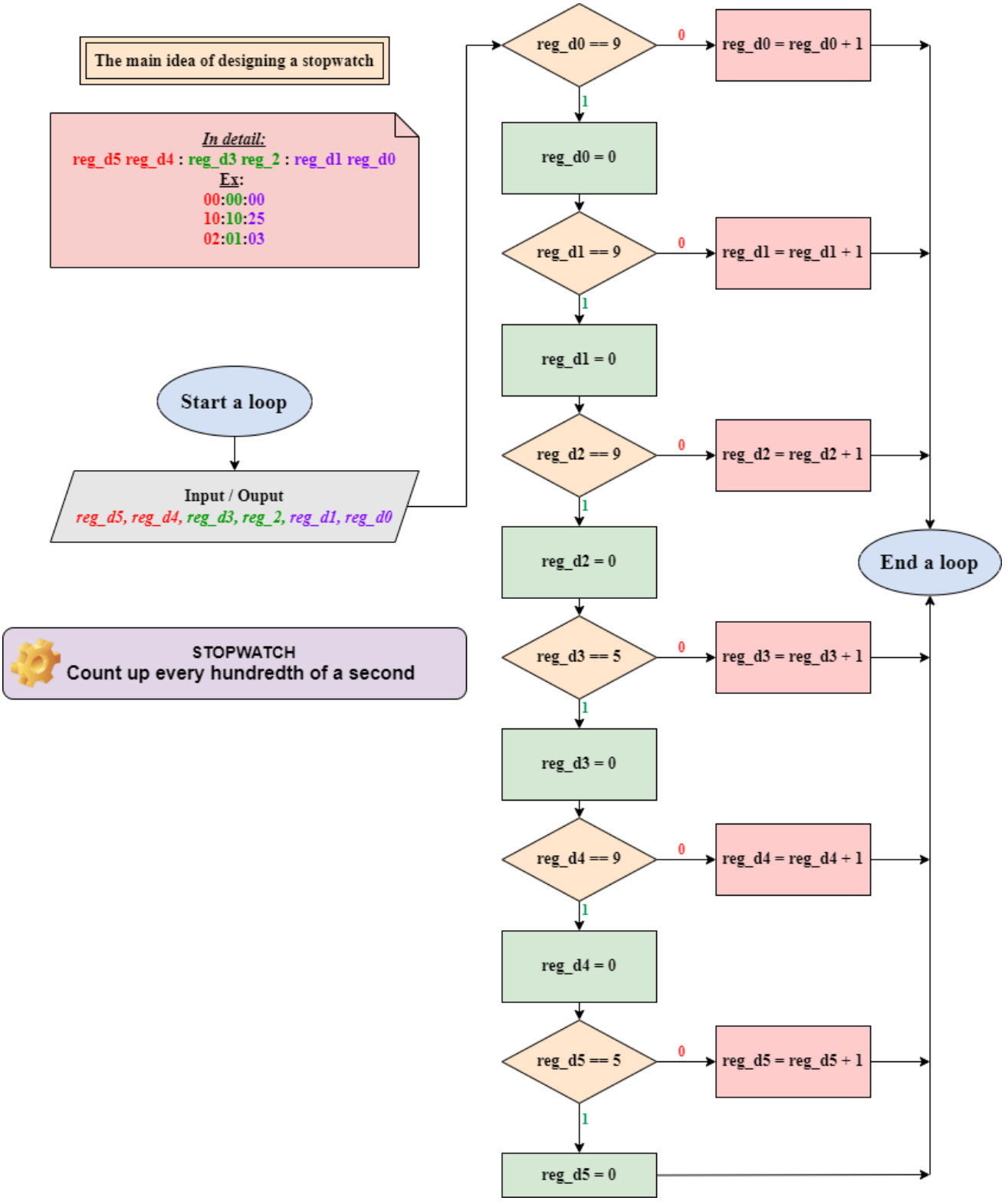


Figure 0.1. Count up diagram

b. Count down block diagram

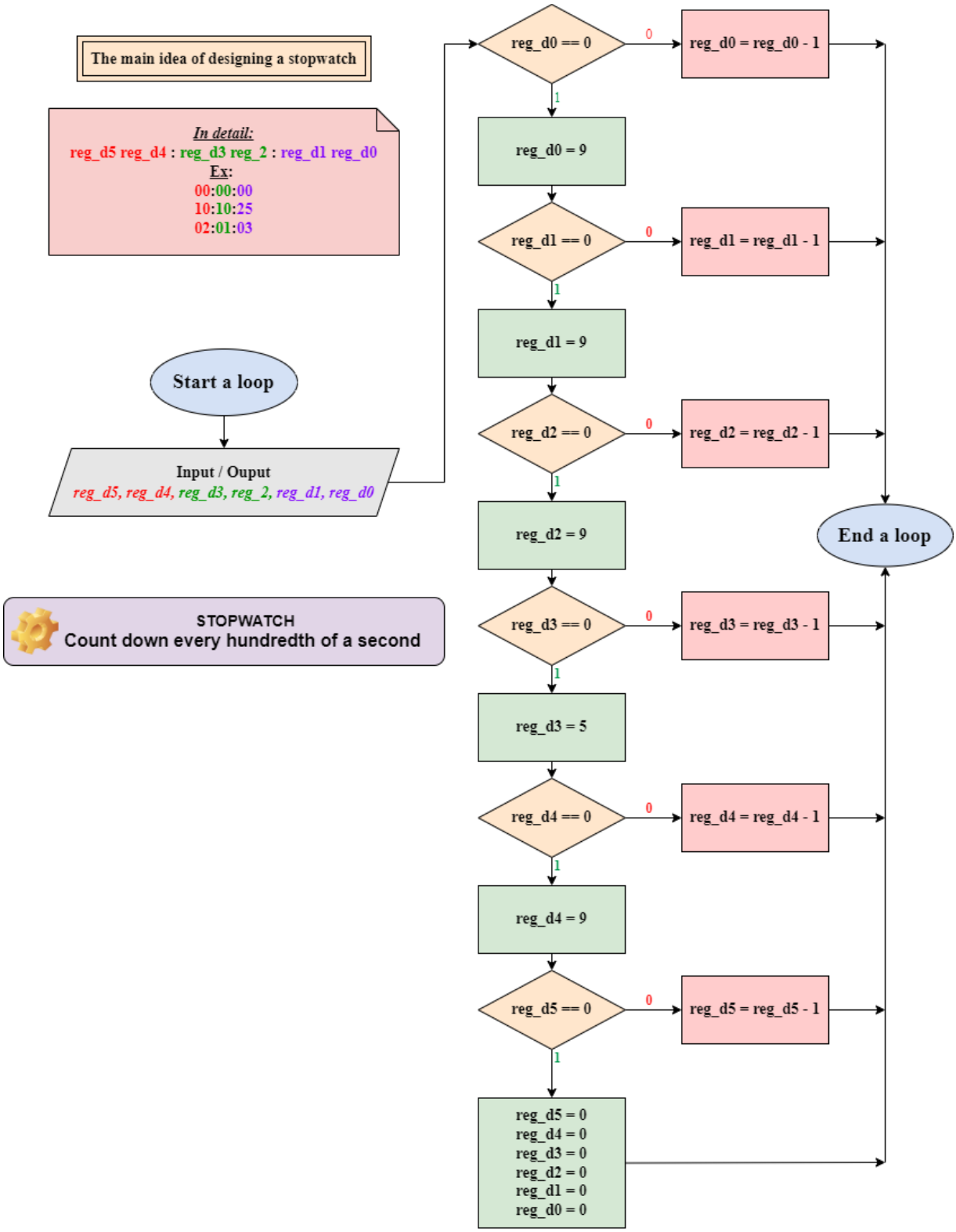


Figure 0.2. Count down diagram

2. Function blocks

- A module frequency divider that can set the cycle of the clock is 0.01 second.
- Create buttons/switches for 4 functions:
  - Function 1: Reset stopwatch to 00:00:00.
  - Function 2: Count up from 00:00:00 / previous time.
  - Function 3: Stop temporarily.
  - Function 4: Count down from previous time / specific time.
- Create buttons/switches to set the initial time:
  - The unit of second: 1 button (switch).
  - The tens of second: 1 button (switch).
  - The unit of minute: 1 button (switch).
  - The tens of minute: 1 button (switch).
- Actualize 2 block diagrams above to count time up and count down.

IV. REALIZATION

1. Design the module

- File: **frequencydivider.v** & **sportstopwatch.v**

a. A sub-module "frequencydivider" to set the clock of Flip Flop.

```
23  module frequencydivider(rst,clk,newclk);
24      input [1:0] rst;
25      input clk;
26      output reg newclk;
27      reg[19:0] count = 0;
28  always @(posedge clk) begin
29      count = count + 1;
30      if(rst == 2'b01)
31          count = 0;
32      else
33          begin
34              newclk = count[19];//19
35          end
36      end
37  endmodule
```

Figure 1. Module frequencydivider

b. Main module introduction

- Input:
  - **clk**: initial clock signal (125MHz).
  - **btn[3:0]**: a modifier which sets the initial second and minute before running a stopwatch.
  - **control**: a controller which sets different function/state of a stopwatch.

- Output:
  - **[6:0] a0,a1,a2,a3,a4,a5**: contain a number representing for minute, second and hundredth of second respectively in order to activate 7-segment LEDs.
  - **red, green**: set colors to led when the stopwatch counts down to 00:00:00.

c. Set up 2 switches SW0(**control[0]**), SW1(**control[1]**) to switch between 4 functions:
- **control** == 2'b00: Reset stopwatch to 00:00:00 / Set up a specific time.
- **control** == 2'b01: Up counter.
- **control** == 2'b11: Stop temporarily / Pause.
- **control** == 2'b10: Down counter.

```
else if(control == 2'b11)
begin
reg_d0 = reg_d0;
reg_d1 = reg_d1;
reg_d2 = reg_d2;
reg_d3 = reg_d3;
reg_d4 = reg_d4;
reg_d5 = reg_d5;
end
```

Figure 2. When **control** == 2'b11, stopwatch will stop and display the current time.

d. Set up functions for 4 buttons BTN0, BTN1, BTN2, BTN3 to set up a specific time:

```
if (control == 2'b00 )
   begin
   if(btn[0] == 1 && count < 1)
        begin
           check = 1;
           count = count+ 1;
           if(reg_d2 == 9)
               reg_d2 = 0;
           else
               reg_d2 = reg_d2 + 1;
        end
```

Figure 3.1. Count up 1 unit of second

```
    else if(btn[1] == 1 && count < 1)
      begin
          check = 1;
          count = count +1;
          if(reg_d3 == 5)
              reg_d3 = 0;
          else
              reg_d3 = reg_d3 + 1;
      end
```

Figure 3.2. Count up a 10-unit of second

```
else if(btn[2] == 1 && count < 1)
    begin
        check = 1;
        count = count + 1;
        if(reg_d4 == 9)
            reg_d4 = 0;
        else
            reg_d4 = reg_d4 + 1;
    end
```

Figure 3.3. Count up 1 unit of minute

```
else if(btn[3] == 1 && count < 1)
    begin
        check = 1;
        count =count+ 1;
        if(reg_d5 == 5)
            reg_d5 = 0;
        else
            reg_d5 = reg_d5 + 1;
    end
```

Figure 3.4. Count up a 10-unit of minute

```
else if(check == 1 && btn == 4'b0000)
begin
count = 0;
end
else if(check == 0)
begin
reg_d0 = 0;
reg_d1 = 0;
reg_d2 = 0;
reg_d3 = 0;
reg_d4 = 0;
reg_d5 = 0;
a0 = 7'b1000000;
a1 = 7'b1000000;
a2 = 7'b1000000;
a3 = 7'b1000000;
a4 = 7'b1000000;
a5 = 7'b1000000;

end
```

Figure 3.5. When none of buttons are pressed, time is set to 00:00:00

e. Set up a 7-segment LED to accept a 4-bit input and generate a 7-bit output.

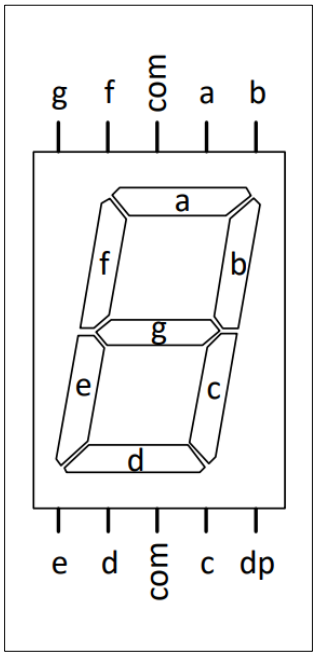


Figure 4. How to connect a 7-segment LED to ARTY Z7 Kit

```
case(reg_d2)
            4'd0 : a2 = 7'b1000000;
            4'd1 : a2 = 7'b1111001;
            4'd2 : a2 = 7'b0100100;
            4'd3 : a2 = 7'b0110000;
            4'd4 : a2 = 7'b0011001;
            4'd5 : a2 = 7'b0010010;
            4'd6 : a2 = 7'b0000010;
            4'd7 : a2 = 7'b1111000;
            4'd8 : a2 = 7'b0000000;
            4'd9 : a2 = 7'b0010000;
            default : a2 = 7'b0111111;
 endcase
```

Figure 5. Set up **"reg_d2"** 7-segment LED

f. Actualize 2 block diagrams above to count time up and count down.

```
begin
    if(reg_d0 == 0)
        begin
        reg_d0 = 9;
            if(reg_d1 == 0)
                begin
                reg_d1 = 9;
                    if(reg_d2 == 0)
                        begin
                        reg_d2 = 9;
                            if(reg_d3 == 0)
                                begin
                                reg_d3 = 5;
                                if(reg_d4 == 0)
                                    begin
                                    reg_d4 = 9;
                                        if(reg_d5 == 0)
                                            begin
                                            reg_d0 = 0;
                                            reg_d1 = 0;
                                            reg_d2 = 0;
                                            reg_d3 = 0;
                                            reg_d4 = 0;
                                            reg_d5 = 0;
                                            red = 1;
                                            green = 1;
                                            end
                                        else
                                            begin
                                            reg_d5 = reg_d5 - 1;
                                            end
                                        case(reg_d0)
                                                    4'd0 : a5 = 7'b1000000;
                                                    4'd1 : a5 = 7'b1111001;
                                                    4'd2 : a5 = 7'b0100100;
                                                    4'd3 : a5 = 7'b0110000;
                                                    4'd4 : a5 = 7'b0011001;
                                                    4'd5 : a5 = 7'b0010010;
                                                    4'd6 : a5 = 7'b0000010;
                                                    4'd7 : a5 = 7'b1111000;
                                                    4'd8 : a5 = 7'b0000000;
                                                    4'd9 : a5 = 7'b0010000;
                                                    default : a5 = 7'b0111111;
                                         endcase
                                    end
```

Figure 6.1. Down counter code

```
                    else
                        begin
                        reg_d4 = reg_d4 - 1;
                        end
                    case(reg_d4)
                                4'd0 : a4 = 7'b1000000;
                                4'd1 : a4 = 7'b1111001;
                                4'd2 : a4 = 7'b0100100;
                                4'd3 : a4 = 7'b0110000;
                                4'd4 : a4 = 7'b0011001;
                                4'd5 : a4 = 7'b0010010;
                                4'd6 : a4 = 7'b0000010;
                                4'd7 : a4 = 7'b1111000;
                                4'd8 : a4 = 7'b0000000;
                                4'd9 : a4 = 7'b0010000;
                                default : a4 = 7'b0111111;
                      endcase
                    end
                else
                    begin
                    reg_d3 = reg_d3 - 1;
                    end
                case(reg_d3)
                            4'd0 : a3 = 7'b1000000;
                            4'd1 : a3 = 7'b1111001;
                            4'd2 : a3 = 7'b0100100;
                            4'd3 : a3 = 7'b0110000;
                            4'd4 : a3 = 7'b0011001;
                            4'd5 : a3 = 7'b0010010;
                            4'd6 : a3 = 7'b0000010;
                            4'd7 : a3 = 7'b1111000;
                            4'd8 : a3 = 7'b0000000;
                            4'd9 : a3 = 7'b0010000;
                            default : a3 = 7'b0111111;
                endcase

            end
        else
            begin
            reg_d2 = reg_d2 - 1;
            end
        case(reg_d2)
                    4'd0 : a2 = 7'b1000000;
                    4'd1 : a2 = 7'b1111001;
                    4'd2 : a2 = 7'b0100100;
                    4'd3 : a2 = 7'b0110000;
                    4'd4 : a2 = 7'b0011001;
                    4'd5 : a2 = 7'b0010010;
                    4'd6 : a2 = 7'b0000010;
                    4'd7 : a2 = 7'b1111000;
                    4'd8 : a2 = 7'b0000000;
                    4'd9 : a2 = 7'b0010000;
                    default : a2 = 7'b0111111;
```

Figure 6.2. Down counter code

```
                    endcase

                end
            else
                begin
                reg_d1 = reg_d1 - 1;
                end
            case(reg_d1)
                        4'd0 : a1 = 7'b1000000;
                        4'd1 : a1 = 7'b1111001;
                        4'd2 : a1 = 7'b0100100;
                        4'd3 : a1 = 7'b0110000;
                        4'd4 : a1 = 7'b0011001;
                        4'd5 : a1 = 7'b0010010;
                        4'd6 : a1 = 7'b0000010;
                        4'd7 : a1 = 7'b1111000;
                        4'd8 : a1 = 7'b0000000;
                        4'd9 : a1 = 7'b0010000;
                        default : a1 = 7'b0111111;
             endcase
        end
    else
        begin
        reg_d0 = reg_d0 - 1;
        end
    case(reg_d0)
        4'd0 : a0 = 7'b1000000;
        4'd1 : a0 = 7'b1111001;
        4'd2 : a0 = 7'b0100100;
        4'd3 : a0 = 7'b0110000;
        4'd4 : a0 = 7'b0011001;
        4'd5 : a0 = 7'b0010010;
        4'd6 : a0 = 7'b0000010;
        4'd7 : a0 = 7'b1111000;
        4'd8 : a0 = 7'b0000000;
        4'd9 : a0 = 7'b0010000;
        default : a0 = 7'b0111111;
     endcase
     check = 0;
end
end
```

Figure 6.3. Down counter code

2. Test bench

- File: **assignment_tb.v**

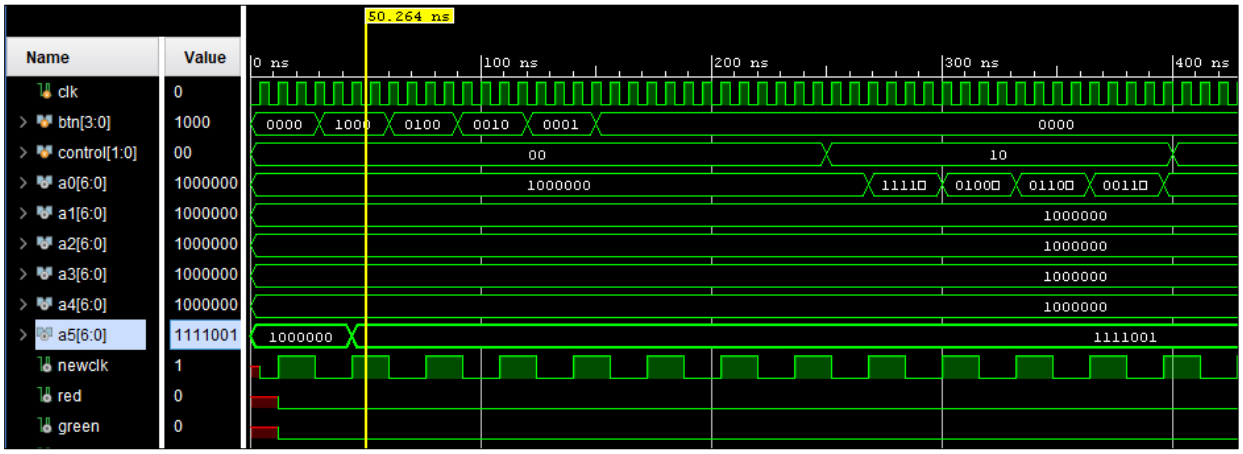


Figure 7.1. When BTN3(the tens of the minute) is pressed and control is 2b'00, a5 rises from 7b'1000000 to 7b'1111001

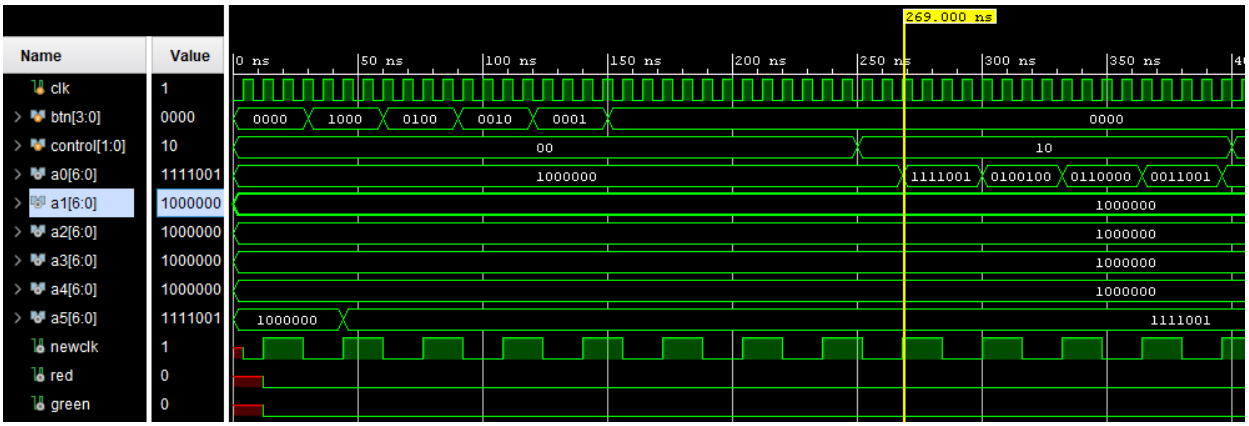


Figure 7.2. When BTN[3:0] = 4b'0000 and control[1:0] = 2b'10, stopwatch starts counting up (a0: 1000000 – 1111001 – 0100100 – 0100100 - …)

3. Constraints

```
## Clock Signal
set_property -dict { PACKAGE_PIN H16    IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L13P_T2_MRCC_35 Sch=SYSCLK
#create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { clk }];#set

## Switches
set_property -dict { PACKAGE_PIN M20    IOSTANDARD LVCMOS33 } [get_ports { control[0] }]; #IO_L7N_T1_AD2N_35 Sch=SW0
set_property -dict { PACKAGE_PIN M19    IOSTANDARD LVCMOS33 } [get_ports { control[1] }]; #IO_L7P_T1_AD2P_35 Sch=SW1

## RGB LEDs
#set_property -dict { PACKAGE_PIN L15    IOSTANDARD LVCMOS33 } [get_ports { led4_b }]; #IO_L22N_T3_AD7P_35 Sch=LED4_B
set_property -dict { PACKAGE_PIN G17    IOSTANDARD LVCMOS33 } [get_ports { green }]; #IO_L16P_T2_35 Sch=LED4_G
set_property -dict { PACKAGE_PIN N15    IOSTANDARD LVCMOS33 } [get_ports { red }]; #IO_L21P_T3_DQS_AD14P_35 Sch=LED4_R
#set_property -dict { PACKAGE_PIN G14    IOSTANDARD LVCMOS33 } [get_ports { led5_b }]; #IO_0_35 Sch=LED5_B
#set_property -dict { PACKAGE_PIN L14    IOSTANDARD LVCMOS33 } [get_ports { led5_g }]; #IO_L22P_T3_AD7P_35 Sch=LED5_G
#set_property -dict { PACKAGE_PIN M15    IOSTANDARD LVCMOS33 } [get_ports { led5_r }]; #IO_L23N_T3_35 Sch=LED5_R

## LEDs
set_property -dict { PACKAGE_PIN R14    IOSTANDARD LVCMOS33 } [get_ports { led }]; #IO_L6N_T0_VREF_34 Sch=LED0
#set_property -dict { PACKAGE_PIN P14    IOSTANDARD LVCMOS33 } [get_ports { led[1] }]; #IO_L6P_T0_34 Sch=LED1
#set_property -dict { PACKAGE_PIN N16    IOSTANDARD LVCMOS33 } [get_ports { led[2] }]; #IO_L21N_T3_DQS_AD14N_35 Sch=LED2
#set_property -dict { PACKAGE_PIN M14    IOSTANDARD LVCMOS33 } [get_ports { led[3] }]; #IO_L23P_T3_35 Sch=LED3

## Buttons
set_property -dict { PACKAGE_PIN D19    IOSTANDARD LVCMOS33 } [get_ports { btn[0] }]; #IO_L4P_T0_35 Sch=BTN0
set_property -dict { PACKAGE_PIN D20    IOSTANDARD LVCMOS33 } [get_ports { btn[1] }]; #IO_L4N_T0_35 Sch=BTN1
set_property -dict { PACKAGE_PIN L20    IOSTANDARD LVCMOS33 } [get_ports { btn[2] }]; #IO_L9N_T1_DQS_AD3N_35 Sch=BTN2
set_property -dict { PACKAGE_PIN L19    IOSTANDARD LVCMOS33 } [get_ports { btn[3] }]; #IO_L9P_T1_DQS_AD3P_35 Sch=BTN3
```

Figure 8.1. Constraints

```
## ChipKit Outer Digital Header
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { a3[0]  }]; #IO_L5P_T0_34            Sch=CK_IO0
set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { a3[1]  }]; #IO_L2N_T0_34            Sch=CK_IO1
set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { a3[2]  }]; #IO_L3P_T0_DQS_PUDC_B_34 Sch=CK_IO2
set_property -dict { PACKAGE_PIN V13   IOSTANDARD LVCMOS33 } [get_ports { a3[3]  }]; #IO_L3N_T0_DQS_34        Sch=CK_IO3
set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports { a3[4]  }]; #IO_L10P_T1_34           Sch=CK_IO4
set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 } [get_ports { a3[5]  }]; #IO_L5N_T0_34            Sch=CK_IO5
set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 } [get_ports { a3[6]  }]; #IO_L19P_T3_34           Sch=CK_IO6
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { a2[0] }]; #IO_L9N_T1_DQS_34        Sch=CK_IO7
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { a2[1]   }]; #IO_L21P_T3_DQS_34       Sch=CK_IO8
set_property -dict { PACKAGE_PIN V18   IOSTANDARD LVCMOS33 } [get_ports { a2[2]   }]; #IO_L21N_T3_DQS_34       Sch=CK_IO9
set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { a2[3]  }]; #IO_L9P_T1_DQS_34        Sch=CK_IO10
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { a2[4]  }]; #IO_L19N_T3_VREF_34      Sch=CK_IO11
set_property -dict { PACKAGE_PIN P18   IOSTANDARD LVCMOS33 } [get_ports { a2[5]  }]; #IO_L23N_T3_34           Sch=CK_IO12
set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { a2[6]  }]; #IO_L23P_T3_34           Sch=CK_IO13

## ChipKit Inner Digital Header
set_property -dict { PACKAGE_PIN U5    IOSTANDARD LVCMOS33 } [get_ports { a1[0]}]; #IO_L19N_T3_VREF_13  Sch=CK_IO26
set_property -dict { PACKAGE_PIN V5    IOSTANDARD LVCMOS33 } [get_ports { a1[1] }]; #IO_L6N_T0_VREF_13   Sch=CK_IO27
set_property -dict { PACKAGE_PIN V6    IOSTANDARD LVCMOS33 } [get_ports { a1[2] }]; #IO_L22P_T3_13       Sch=CK_IO28
set_property -dict { PACKAGE_PIN U7    IOSTANDARD LVCMOS33 } [get_ports { a1[3] }]; #IO_L11P_T1_SRCC_13  Sch=CK_IO29
set_property -dict { PACKAGE_PIN V7    IOSTANDARD LVCMOS33 } [get_ports { a1[4] }]; #IO_L11N_T1_SRCC_13  Sch=CK_IO30
set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS33 } [get_ports { a1[5] }]; #IO_L17N_T2_13       Sch=CK_IO31
set_property -dict { PACKAGE_PIN V8    IOSTANDARD LVCMOS33 } [get_ports { a1[6] }]; #IO_L15P_T2_DQS_13   Sch=CK_IO32
set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { a0[0] }]; #IO_L21N_T3_DQS_13   Sch=CK_IO33
set_property -dict { PACKAGE_PIN W10   IOSTANDARD LVCMOS33 } [get_ports { a0[1]  }]; #IO_L16P_T2_13       Sch=CK_IO34
set_property -dict { PACKAGE_PIN W6    IOSTANDARD LVCMOS33 } [get_ports { a0[2]  }]; #IO_L22N_T3_13       Sch=CK_IO35
set_property -dict { PACKAGE_PIN Y6    IOSTANDARD LVCMOS33 } [get_ports { a0[3]  }]; #IO_L13N_T2_MRCC_13  Sch=CK_IO36
set_property -dict { PACKAGE_PIN Y7    IOSTANDARD LVCMOS33 } [get_ports { a0[4]  }]; #IO_L13P_T2_MRCC_13  Sch=cCK_IO37
set_property -dict { PACKAGE_PIN W8    IOSTANDARD LVCMOS33 } [get_ports { a0[5]  }]; #IO_L15N_T2_DQS_13   Sch=CK_IO38
set_property -dict { PACKAGE_PIN Y8    IOSTANDARD LVCMOS33 } [get_ports { a0[6]  }]; #IO_L14N_T2_SRCC_13  Sch=CK_IO39
set_property -dict { PACKAGE_PIN W9    IOSTANDARD LVCMOS33 } [get_ports {a5[5]}]; #IO_L16N_T2_13       Sch=CK_IO40
set_property -dict { PACKAGE_PIN Y9    IOSTANDARD LVCMOS33 } [get_ports { a5[6]}]; #IO_L14P_T2_SRCC_13  Sch=CK_IO41
```

Figure 8.2. Constraints

```
set_property -dict { PACKAGE_PIN Y11   IOSTANDARD LVCMOS33 } [get_ports { a4[0] }]; #IO_L18N_T2_13      Sch=CK_A0
set_property -dict { PACKAGE_PIN Y12   IOSTANDARD LVCMOS33 } [get_ports { a4[1] }]; #IO_L20P_T3_13      Sch=CK_A1
set_property -dict { PACKAGE_PIN W11   IOSTANDARD LVCMOS33 } [get_ports { a4[2] }]; #IO_L18P_T2_13      Sch=CK_A2
set_property -dict { PACKAGE_PIN V11   IOSTANDARD LVCMOS33 } [get_ports { a4[3] }]; #IO_L21P_T3_DQS_13  Sch=CK_A3
set_property -dict { PACKAGE_PIN T5    IOSTANDARD LVCMOS33 } [get_ports { a4[4] }]; #IO_L19P_T3_13      Sch=CK_A4
set_property -dict { PACKAGE_PIN U10   IOSTANDARD LVCMOS33 } [get_ports { a4[5] }]; #IO_L12N_T1_MRCC_13 Sch=CK_A5

## ChipKit Inner Analog Header - as Differential Analog Inputs
## NOTE: These ports can be used as differential analog inputs with voltages from 0-1.0V (ChipKit analog pins A6-A11) or as digital I/O.
## WARNING: Do not use both sets of constraints at the same time!
## NOTE: The following constraints should be used with the XADC core when using these ports as analog inputs.
set_property -dict { PACKAGE_PIN F19   IOSTANDARD LVCMOS33 } [get_ports { a4[6] }]; #IO_L15P_T2_DQS_AD12P_35 Sch=AD12_P   ChipKit pin=A6
set_property -dict { PACKAGE_PIN F20   IOSTANDARD LVCMOS33 } [get_ports { a5[0] }]; #IO_L15N_T2_DQS_AD12N_35 Sch=AD12_N   ChipKit pin=A7
set_property -dict { PACKAGE_PIN C20   IOSTANDARD LVCMOS33 } [get_ports { a5[1]  }]; #IO_L1P_T0_AD0P_35       Sch=AD0_P    ChipKit pin=A8
set_property -dict { PACKAGE_PIN B20   IOSTANDARD LVCMOS33 } [get_ports { a5[2]  }]; #IO_L1N_T0_AD0N_35       Sch=AD0_N    ChipKit pin=A9
set_property -dict { PACKAGE_PIN B19   IOSTANDARD LVCMOS33 } [get_ports { a5[3]  }]; #IO_L2P_T0_AD8P_35       Sch=AD8_P    ChipKit pin=A10
set_property -dict { PACKAGE_PIN A20   IOSTANDARD LVCMOS33 } [get_ports { a5[4] }]; #IO_L2N_T0_AD8N_35       Sch=AD8_N    ChipKit pin=A11
```

Figure 8.3. Constraints

4. Schematic

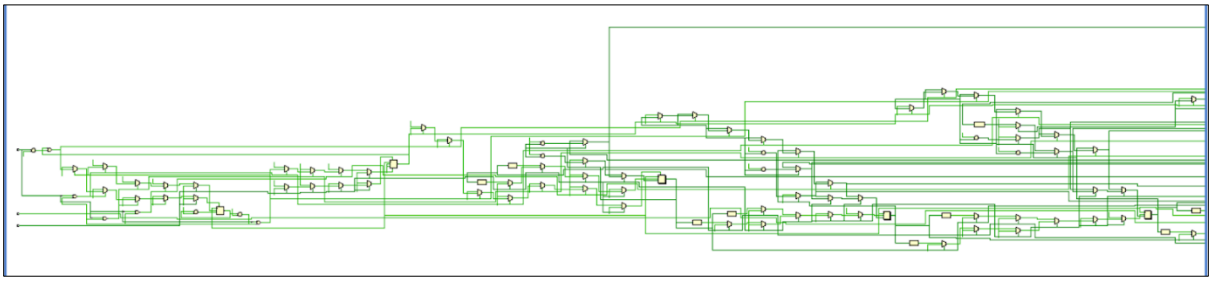


Figure 9.1. Schematic

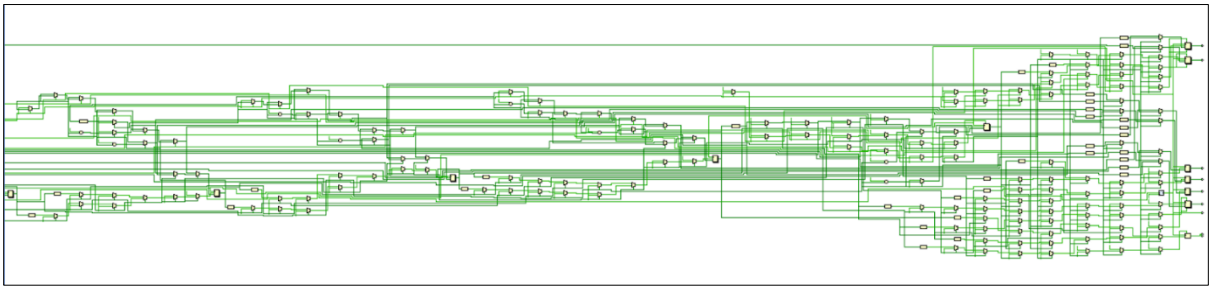


Figure 9.2. Schematic

V. CONCLUSION

1. Results

- Video implementation for each state:
  - Introduction of Stopwatch: https://bom.so/jKP8tF
  - Count up: https://bom.so/mxIoPA
  - Count down: https://bom.so/HLHzNk
  - Set up initial time for down counting: https://bom.so/bRqz8Y

2. Achievement

Complete designing a sport stopwatch that has 4 modes: count up, count down, stop, reset. Moreover, the device is able to set up a specific time to count down.

3. Drawbacks

Since we had used all buttons and switches on ARTY Z7 kit, we are not able to actualize saving a variety of timer results (we can save only 1 result via the pause button).

----------THE END----------