Ethan Nagelvoort
821234668

470L Lab 6 Digital Clock Lab

Description

This lab required me to create a digital clock where you can manually set the hour and minutes. I also added other features to improve the digital clock. Some of these features are an alarm, ability to see seconds and am/pm, and a pause ability. The modules needed to complete this lab are as follows; a top module, 3 different clock divider modules, an alarm module, an anode controller module, a segment controller module, and a clock controller module. It is important to note that the most important module is the clock controller module because it mostly controls the input signals that are sent to most of the modules. This digital clock utilizes many features on the Basys3, such as buttons, switches, and the seven segment display.

The clock controller module is used to control how fast the clock changes, allow the user to modify the clock, and many other functions. It instantiates a clock divider module that allows the clock to increase by seconds. At the beginning of the always block, I have a series of if statements that control how the user sets the time. For the first 6 switches, the user can manipulate these to look like a binary number that will represent either seconds or minutes. Then depending if btnC or btnL is pressed, the minutes or seconds will change. The next 4 switches control what the user wants to set the hour. If btnU is pressed, then the hour will change to what those 4 switches represent in binary. And finally the first switch on the board represents am or pm. If the switch is up and the user presses btnR, pm is activated. If the switch is down, am is activated. Switch 14 also acts as a pause switch and so if up, the clock will pause. This means that the incrementation of the clock part of the module is within a big if statement that prevents any of the incrementation code to run if this switch is up. In this incrementation part of the code, I increment a register cs which represent seconds, so every pos edge of CLK that comes from the clock divider mentioned before, cs increments. Then if cs equals 59, reg cm increments and cs goes back to 0. Cm represents minutes. Then once Cm is 59 and cs is 59, ch increments. Ch represents hours. Once ch is 12, cm is 59, and cs is 59, ch is reverted back to 1. Ch is 12, there is another if statement that changes the am/pm status with the use of a flag called AP. After this, the incrementation part of the code ends and the code is now outside the pause if statement. Now another if statement occurs that alarm_on is one and that cm equals the binary representation of the first 6 switches and ch represent the binary representation of the next 4 switches, an output called alarmFlag turns to 1, else 0. Alarm_on is switch 13 and controls whether the user wants the alarm to be active or not. Then after, cs, cm, and ch are each divided by 10 and modded by 10. Dividing by 10 will give each of these regs the left digit of their value and modding by 10 will give the right digit of their value. Each digit is then outputted from this module and will be used in anode/7-segment representation. The overall minute, overall hour, and ampm status are also outputted.

In the alarm module, the alarmFlag signal found in the clock_controller module is used to determine if this module should be used. This module also uses a clk that is derived from an

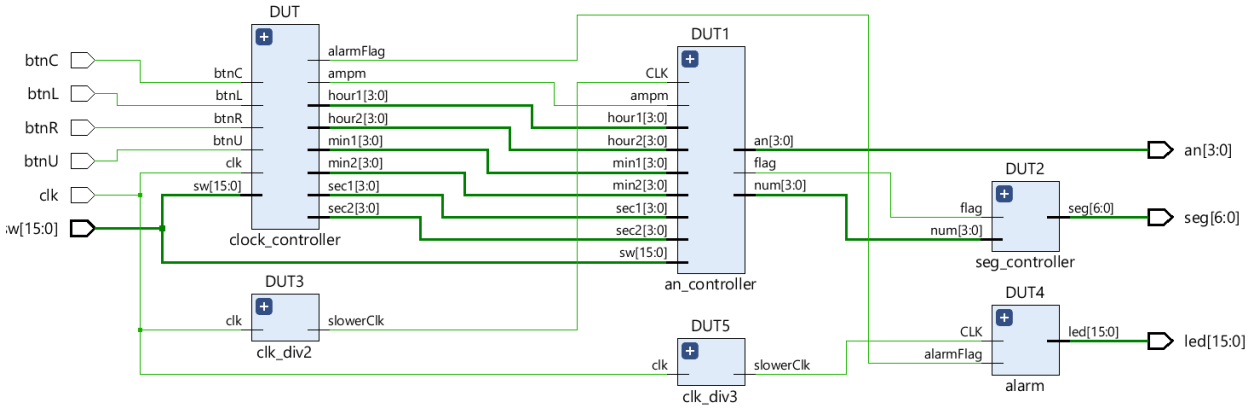
instantiation from a clock divider that is half a second fast. If alarmFlag is 1, then a 16 bit reg called blink is negated so at every half second all the bits in blink will change from 0 to 1 to 0 to etc. Blink is then assigned to all the leds on the basys3 so that all the leds blink at half a second fast when the clock on the seven seg becomes what is set to the alarm.

In the an_controller module, I activate the anodes to operate in a way where all anodes are visible and display different numbers. It uses another clock divider module, where the clock frequency is so low that it allows the module to cycle through all the anodes at a very fast rate on every positive clock edge. This rate is so fast that the human eye cannot notice it turning on and off and so will only perceive it to be on. I utilize switch 15 to switch between display hours and minutes to display seconds and AM/PM. If the switch is down, then there is a case statement that acts as an FSM where it cycles through all the states at once. In each state, I set an to equal the particular anode I want to access at that time. I then have an output reg called num in each state that will equal the digit that will be represented on that particular anode. For example, if the case is in a state where an equals the anode that is nearest to the buttons, then num will equal min1, the first minute digit value. These digit values are signals brought over as inputs from the clock_controller module. Now if switch 15 was up, a new case statement would be used that operates similarly to the other one. In this one, the states that operate the first two anodes will have num equal the seconds digits. The state that operates the anode after that will have num equal ampm, an input signal taken from clock_controller that represents the position of am/pm. It will also have an output reg called flag equal 1. The state that controls the anode after that will have num equal 4'b0010 and will have flag equal 1 as well. Flag equals 0 everywhere else.

In the seg_controller module, the seven segment display is manipulated using input signals that are outputted from the an_constoller module. This module takes the num and flag output from the an_controller module. The module begins with a always block based on * and if flag is 0, then there is a case statement based on num that will find that particular number and represent it on the seven segments like how it is done in the other labs. Then if flag is 1, then a new case statement appears based on num. However, this time the case states will either represent AM/PM or blank. If num is 0 here, then that means the segment will showcase a A for AM. If num is 1, then the segment will showcase a P for PM. Then if num is 3, then the segment will be blank. This part had to be done in a separate case statement utilizing a flag to access it because the other case statement was full for the input values 0-9.

Now it is important to note that a top module was used to connect all the modules together. The clock dividers used for an_controller and alarm appear here but the clock divider used in clock_controller is instantiated in the module itself. All the clock dividers operate as a normal clock divider would but the threshold value is changed in each clock divider to obtain the required frequency value needed for the clock that is connected to the module that uses that particular clock divider.

Block Diagram



User Guide

_____Once started, the clock will start at 12:00. If you want, you can pause the clock by flipping switch 14. You can switch from the hour and minute display to the seconds and am/pm display by flipping switch 15. You can set an alarm by flipping switch 13. Alarm is set at what the end half of the switches represent. Switches 0-5 represent minutes and switches 6-9 represent hours. These values are represented in binary. Once time on the seven segment display reaches what is represented on those switches, then all the leds will start the flash. They will stop flashing once switch 13 is flipped down. You can alter the clock by pressing btnR to change the am/pm setting. If switch 0 is up and btnR is pressed, then setting will switch to PM. If switch 0 is down then it will switch to AM. If btnC is pressed, then minutes will change to whatever is represented in binary in switches 0-5, as long as it is less than 60. If btnU is pressed, then the hour will change to whatever is represented in switches 6-9, as long as it is less than 13 and not 0. If btnL is pressed, then seconds will change to whatever is represented in binary in switches 0-5, as long as it is less than 60.

Source Code

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/16/2021 02:01:32 PM
// Design Name:
// Module Name: clock_controller
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
```

```
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module clock_controller(input clk, btnC, btnU, btnL, btnR, input [15:0] sw, output reg [5:0]
overall_min, output reg alarmFlag, output reg [3:0] overall_hour, output reg [3:0] hour1, hour2,
min1, min2, sec1, sec2, output reg ampm);
wire CLK;
reg flag;
reg alarm_on = 1'b0;
reg [3:0] H1, H2, M1, M2, S1, S2 = 4'b0000;
reg [5:0] cm = 6'b000000; //min and sec counter
reg [5:0] cs = 6'b000000;
reg [3:0] ch = 4'b1100; // hour counter, start at 12
reg [5:0] min_secSW = 6'b000000;
reg ampmSW, pauseSW = 1'b0;
reg [3:0] hourSW = 4'b0000;
reg AP = 1'b1;
clk_div DUT(.clk(clk), .slowerClk(CLK));
always@(posedge CLK) begin
alarm_on = sw[13];
min_secSW = sw [5:0];//first 6 switches used to manipulate minutes or seconds
hourSW = sw [9:6];//next 4 switches used to manipulate hours
ampmSW = sw[0]; // first switch controls weather in PM or AM
pauseSW = sw[14];//used to pause clock
if(btnR == 1)
AP <= ampmSW;
if(btnC == 1 && min_secSW <60)
cm <= min_secSW;
if(btnU == 1 && hourSW < 13 && hourSW != 0)
ch <= hourSW;
if(btnL == 1 && min_secSW <60)
cs <= min_secSW;
if(pauseSW == 0) begin //if 1 clock is paused
```

```
cs <= cs + 1;
if(cs == 59) begin
cm <= cm + 1;
cs <= 0;
end
if(cm == 59 && cs == 59)begin
ch <= ch+1;
cm <= 0;
end
if(ch == 12 && cm == 0 && cs == 0) begin
if(AP == 0)
AP <= 1;
if(AP == 1)
AP <= 0;
end
if(ch == 12 && cm==59 && cs == 59)begin
ch <= 1;
end
end
if(alarm_on == 1) begin
if(min_secSW == cm && hourSW == ch)
alarmFlag <= 1;
end
else
alarmFlag <= 0;
S1 <= cs%10;
S2 <= cs/10;
M1 <= cm%10;
M2 <= cm/10;
H1 <= ch%10;
H2 <= ch/10;
min1 <= M1;
min2 <= M2;
hour1 <= H1;
hour2 <= H2;
sec1 <= S1;
sec2 <= S2;
ampm <= AP;
overall_min <= cm;
overall_hour <= ch;
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/16/2021 01:12:59 PM
// Design Name:
// Module Name: 7-seg
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module an_controller(input [15:0] sw, input CLK, ampm, input [3:0] min1, min2, hour1, hour2,
sec1, sec2, output reg [3:0] an, output reg [3:0] num, output reg flag);
reg[3:0] an_pos = 0;
wire CLK;
reg display_con = 1'b0;
always @(posedge CLK)
begin
display_con = sw[15];
if(display_con == 0) begin
flag <= 0;
   case (an_pos)
      4'b0000: begin
         an <= 4'b1110;
         num <= min1;
         an_pos <= 1;
         end
      4'b0001: begin
         an <= 4'b1101;
```

```
            num <= min2;
            an_pos <= 2;
            end
        4'b0010: begin
            an <= 4'b1011;
            num<= hour1;
            an_pos <= 3;
            end
        4'b0011: begin
            an <= 4'b0111;
            num <= hour2;
            an_pos <= 0;
            end
            endcase
end
else
begin
case (an_pos)
        4'b0000: begin
            an <= 4'b1110;
            num <= 4'b0010;//represents _
            flag <= 1;
            an_pos <= 1;
            end
        4'b0001: begin
            an <= 4'b1101;
            num <= ampm;//represents A or P
            flag <= 1;
            an_pos <= 2;
            end
        4'b0010: begin
            an <= 4'b1011;
            num<= sec1;
            flag <= 0;
            an_pos <= 3;
            end
        4'b0011: begin
            an <= 4'b0111;
            num <= sec2;
            flag <= 0;
```

```
        an_pos <= 0;
        end
        endcase
end
end

endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/16/2021 01:46:46 PM
// Design Name:
// Module Name: second_hour_seg
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module seg_controller(input [3:0] num, input flag, output reg[6:0] seg);

always @(*)
begin

if(flag == 0) begin
   case(num)
      4'b0000: seg = 7'b1000000; // 0000
      4'b0001: seg = 7'b1111001; // 0001
      4'b0010: seg = 7'b0100100; // 0010
```

```
        4'b0011: seg = 7'b0110000; // 0011
        4'b0100: seg = 7'b0011001; // 0100
        4'b0101: seg = 7'b0010010; // 0101
        4'b0110: seg = 7'b0000010; // 0110
        4'b0111: seg = 7'b1111000; // 0111
        4'b1000: seg = 7'b0000000; // 1000
        4'b1001: seg = 7'b0010000; // 1001
        default: seg = 7'b1000000; // 0000
    endcase
end
else begin
case(num)
        4'b0000: seg = 7'b0001000; // A
        4'b0001: seg = 7'b0001100; // P
        4'b0010: seg = 7'b1111111; // _
        default: seg = 7'b1000000; // 0000
    endcase
end
end
endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/16/2021 11:38:53 PM
// Design Name:
// Module Name: alarm
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
```

```
//////////////////////////////////////////////////////////////////////////////////


module alarm(input CLK, input alarmFlag, output [15:0] led);
reg [15:0] blink = 1'b0000000000000000;
always@(posedge CLK) begin
if(alarmFlag) begin
blink <= ~blink;
end
else
blink<= 0;
end
assign led = blink;
endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/16/2021 02:00:37 PM
// Design Name:
// Module Name: cllk_div
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module clk_div (input clk, output reg slowerClk);
reg [31:0] counter;
always@(posedge clk) begin
```

```
if(counter == 1666666) begin//100000000 is threshold, it is close to 1 sec 6250000
slowerClk <= 1;//signal new clk signal
counter <= 0;
end
else begin
slowerClk <= 0;
counter <= counter+1;
end
end

endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/16/2021 01:44:49 PM
// Design Name:
// Module Name: first_hour_seg
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module clk_div2 (input clk, output reg slowerClk);
reg [31:0] counter;
always@(posedge clk) begin
if(counter == 20000) begin//100000000 is threshold, it is close to 1 sec 6250000
slowerClk <= 1;//signal new clk signal
counter <= 0;
```

```
end
else begin
slowerClk <= 0;
counter <= counter+1;
end
end

endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/17/2021 12:35:37 AM
// Design Name:
// Module Name: clk_div3
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module clk_div3 (input clk, output reg slowerClk);
reg [31:0] counter;
always@(posedge clk) begin
if(counter == 50000000) begin//100000000 is threshold, it is close to 1 sec 6250000
slowerClk <= 1;//signal new clk signal
counter <= 0;
end
else begin
slowerClk <= 0;
counter <= counter+1;
```

```
end
end

endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/16/2021 03:14:20 PM
// Design Name:
// Module Name: top
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module top(input clk, btnC, btnU, btnL, btnR, input [15:0] sw, output [3:0] an, output [6:0] seg,
output[15:0] led);
wire [3:0] hour1, hour2, min1, min2, sec1, sec2, num, overall_hour;
wire [5:0] overall_min;
wire CLK, ampm, flag, CLK2, alarmFlag;
clk_div2 DUT3(.clk(clk), .slowerClk(CLK));
clk_div3 DUT5(.clk(clk), .slowerClk(CLK2));
clock_controller DUT(.clk(clk), .alarmFlag(alarmFlag),  .btnR(btnR), .ampm(ampm),
.hour1(hour1), .btnL(btnL), .sec1(sec1), .sec2(sec2), .hour2(hour2), .min1(min1), .min2(min2),
.btnC(btnC), .btnU(btnU), .sw(sw));
an_controller DUT1 (.flag(flag),.CLK(CLK), .sw(sw), .min1(min1), .ampm(ampm), .sec1(sec1),
.sec2(sec2), .min2(min2), .hour1(hour1), .hour2(hour2), .num(num), .an(an));
seg_controller DUT2 (.seg(seg), .num(num), .flag(flag));
```

```
alarm DUT4 (.CLK(CLK2), .alarmFlag(alarmFlag), .led(led));
endmodule
```