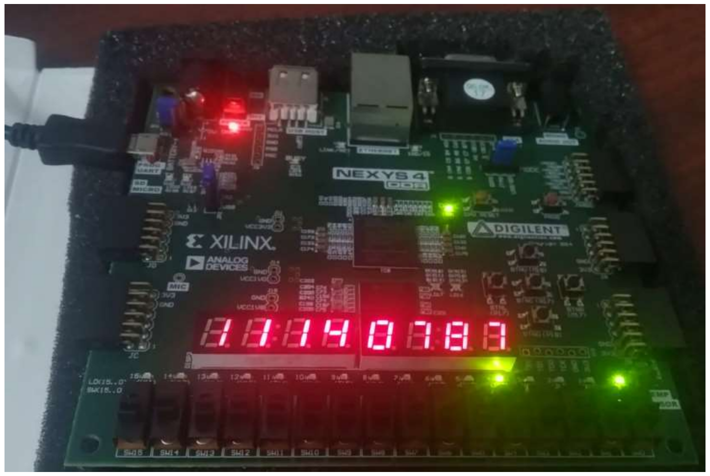
12 HOUR DIGITAL CLOCK ON FPGA BOARD



NARENDRA KUMAR NEHRA

CONTENTS

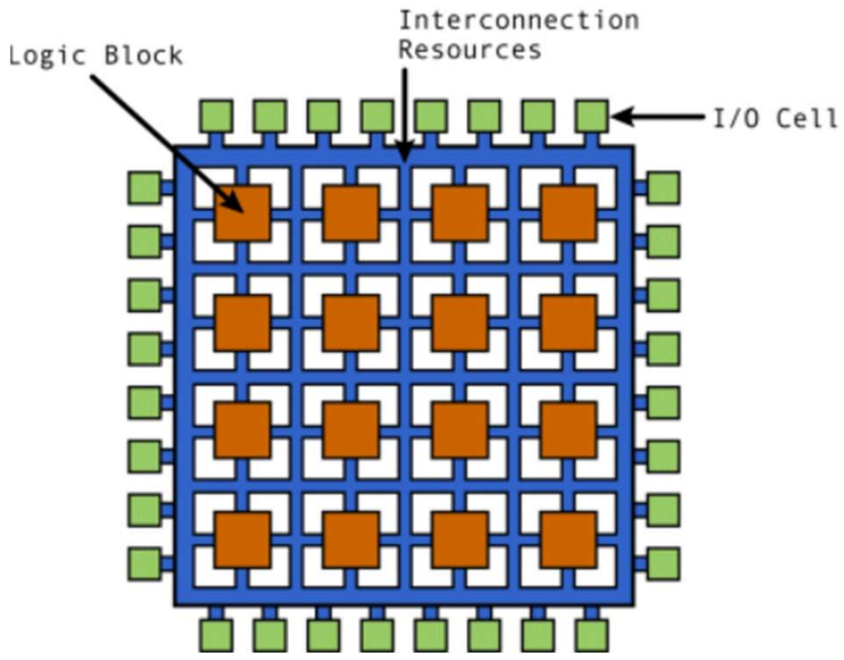
- Introduction
- What is FPGA?
- Programming FPGA
- Algorithm of 12 Hour Digital clock
- Ports and Peripherals of FPGA needed to implement 12 hour digital clock
- Modules in Verilog code
- Implementation on FPGA Board
- References

INTRODUCTION

- The main objective of the project is to display the time digitally using 7-segment display on Artix-7 FPGA Board.
- The digital clock designed is a 12-clock hour format. It displays the time in format of **Hours : Minutes : Seconds : Millisecond**.
- A LED is used to show AM/PM.

WHAT IS FPGA?

- Field Programmable Gate Array
- Consists of configurable logic blocks, interconnections and input-output blocks.
- The gate arrays are programmed for a specific function by the user instead of by the manufacturer of the device.
- Introduced by XILINX in 1985.
- First FPGA is XILINX XC2064.



- The main advantage of the FPGA is it’s ability to easily change its functionality after a product has been designed.
- FPGA require a smaller board space and can be more energy efficient than the equivalent discrete circuit.
- Used in aerospace, defence, medicine, image processing, digital signal processing etc.

PROGRAMMING FPGA

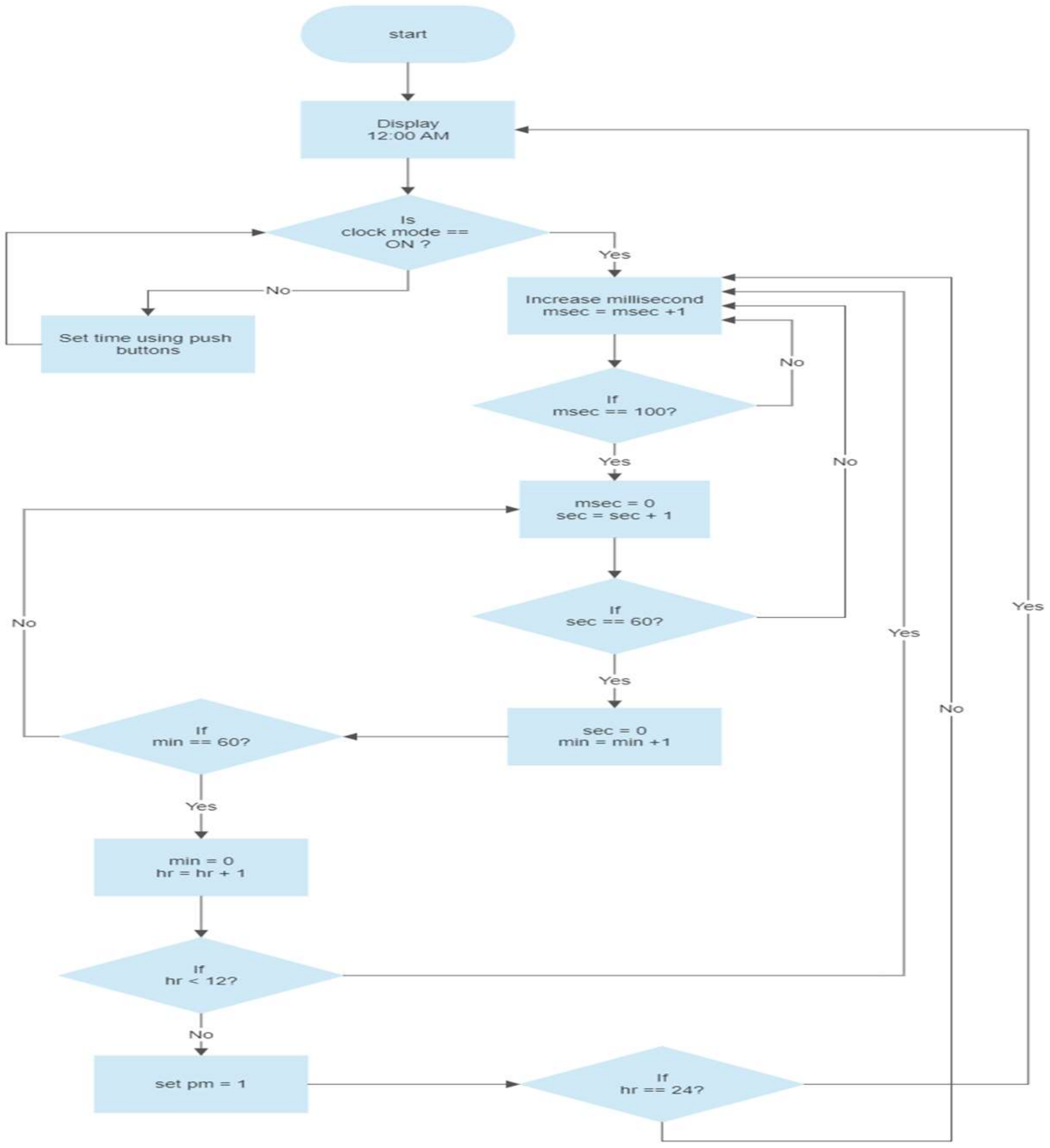
- For making 12 hour digital clock, we would be using the behaviour level modelling.
- In this, we are more concerned about the behaviour of the digital clock rather than it’s internal structure.

Steps included in implementing any code

- First selecting the hardware device and writing the code for required logic.
- **Simulation** is used for verification of the logic.
- **Register-transfer level (RTL)** models a circuit in terms of the data flow between hardware register, and the logical operations performed on those signals.
- **Input/Output planning** is done by selecting the ports.
- **Synthesis** turns HDL files into a transistor level description based on timing and I/O constraints.

- **Implementation** gives pseudo schematic of hardware for given logic.
- **Bitstream Generator** generates a bit file for the final outputs needed for programming the FPGA.
- **Hardware Manager** is used for programming the target device. First connect the target device, then, program it with the bit file generated.

Algorithm of 12 Hour Digital Clock

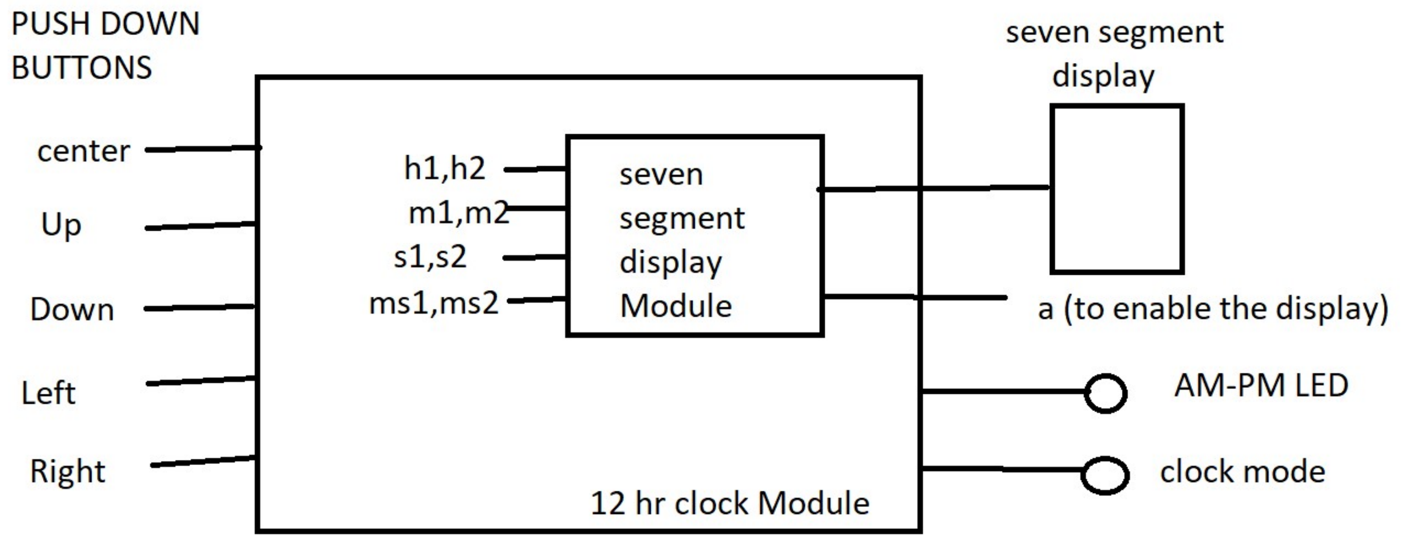


Ports and Peripherals

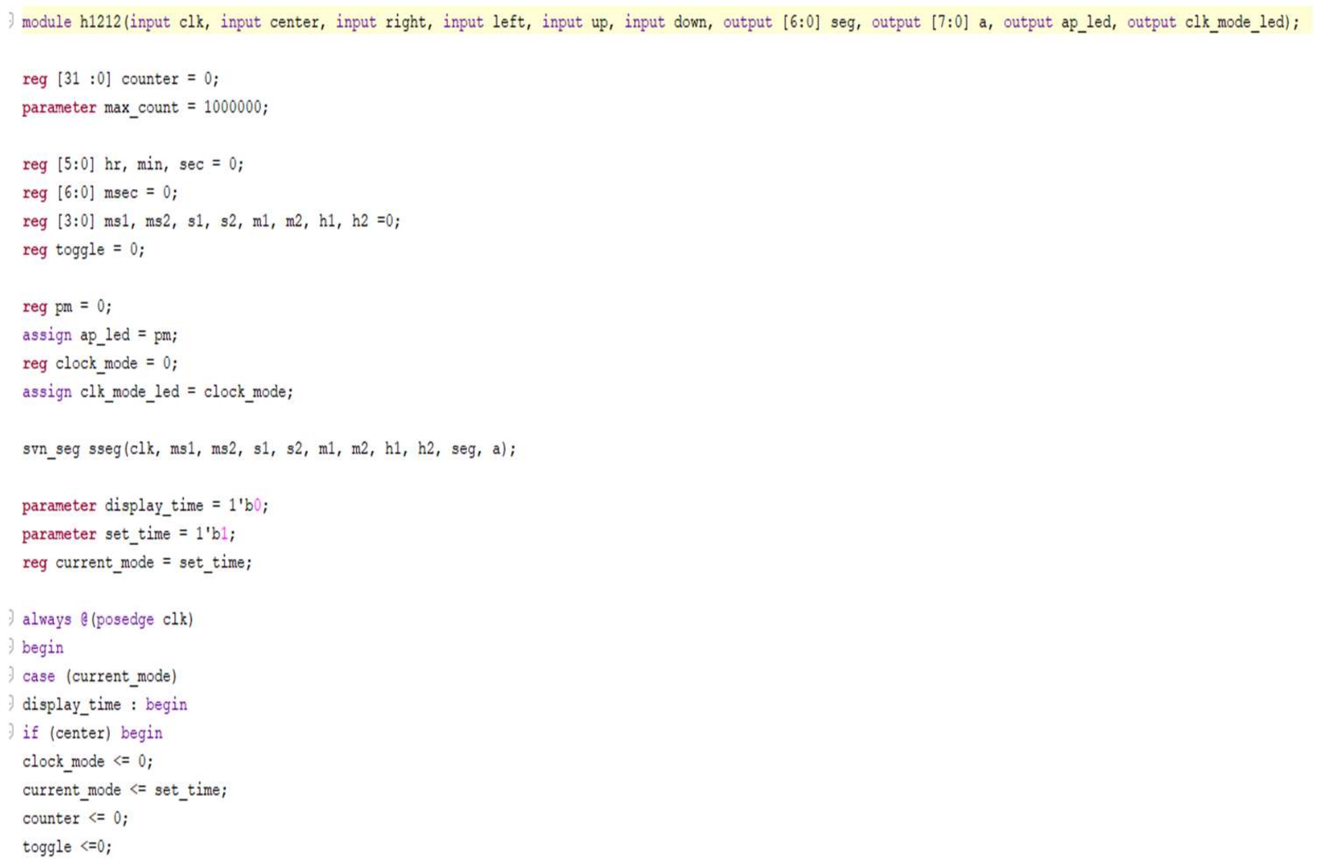
- Seven segment display
- 5 push buttons
- 1 LED to show whether it is clock mode or Set mode
- 1 LED to show AM/PM

NEXYS 4 DDR FPGA BY XILINX

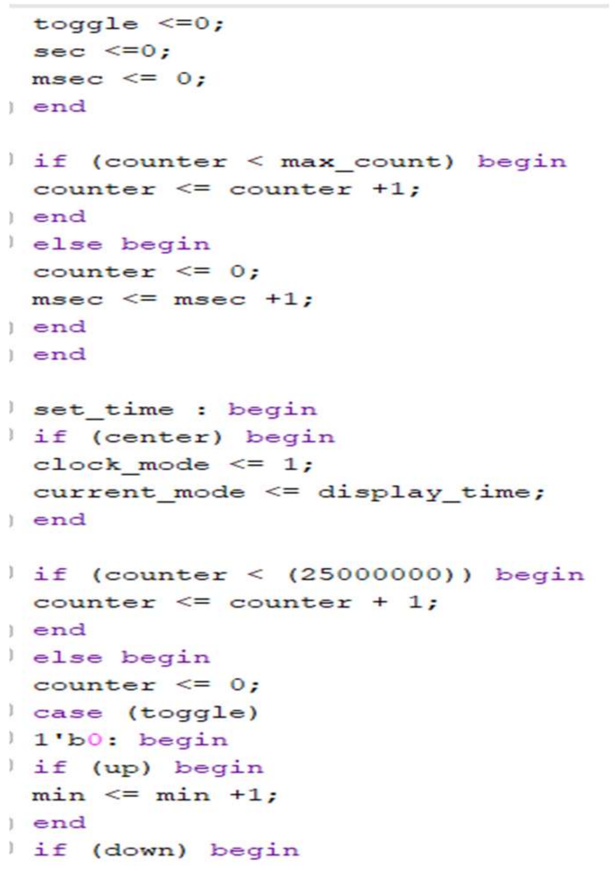
BLOCK DIAGRAM



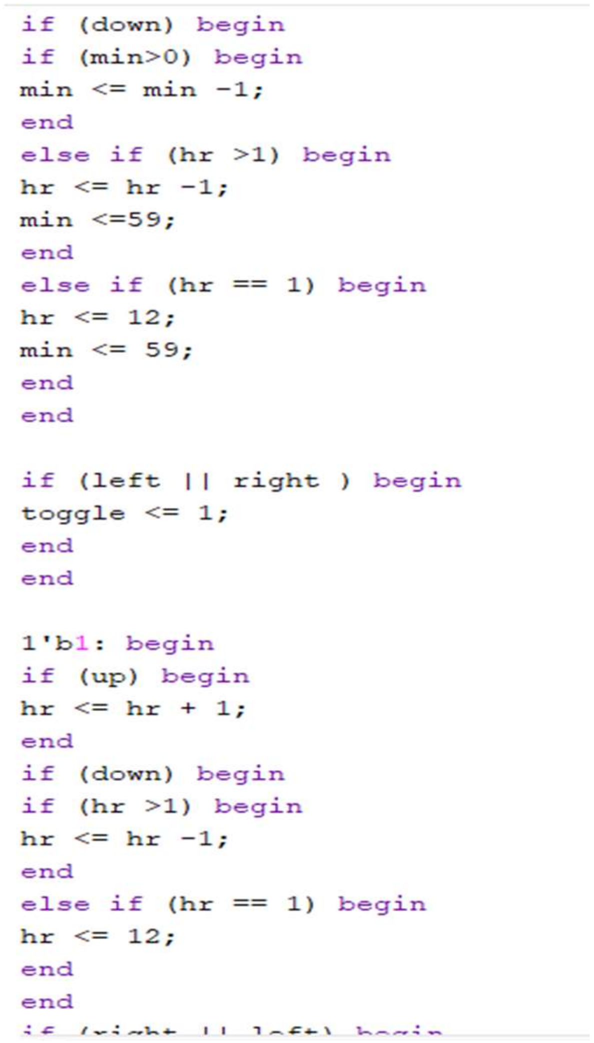
Verilog Code



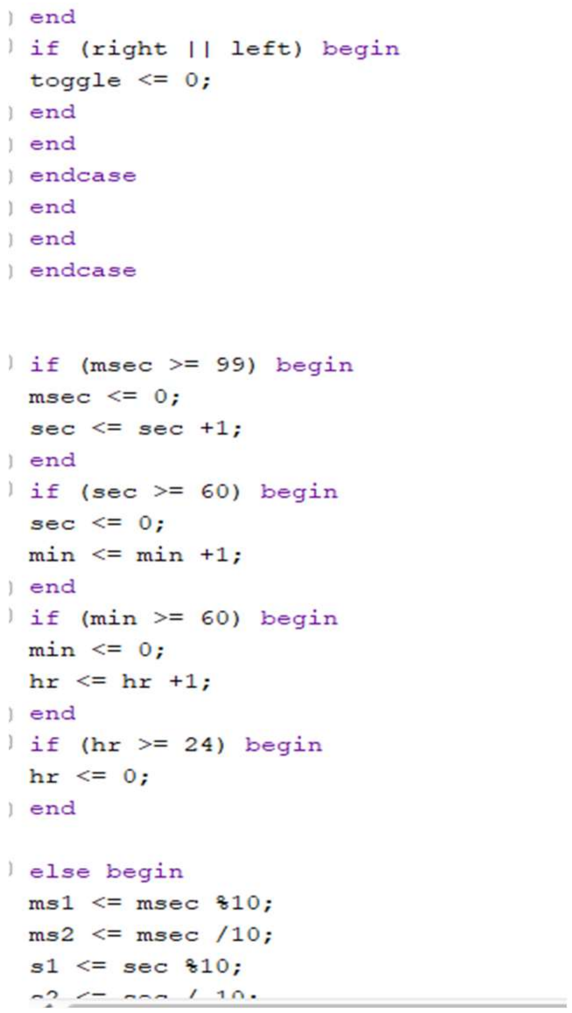
```
module h1212(input clk, input center, input right, input left, input up, input down, output [6:0] seg, output [7:0] a, output ap_led, output clk_mode_led);

reg [31 :0] counter = 0;
parameter max_count = 1000000;

reg [5:0] hr, min, sec = 0;
reg [6:0] msec = 0;
reg [3:0] ms1, ms2, s1, s2, m1, m2, h1, h2 =0;
reg toggle = 0;

reg pm = 0;
assign ap_led = pm;
reg clock_mode = 0;
assign clk_mode_led = clock_mode;

svn_seg sseg(clk, ms1, ms2, s1, s2, m1, m2, h1, h2, seg, a);

parameter display_time = 1'b0;
parameter set_time = 1'b1;
reg current_mode = set_time;

always @(posedge clk)
begin
case (current_mode)
display_time : begin
if (center) begin
clock_mode <= 0;
current_mode <= set_time;
counter <= 0;
toggle <=0;
```



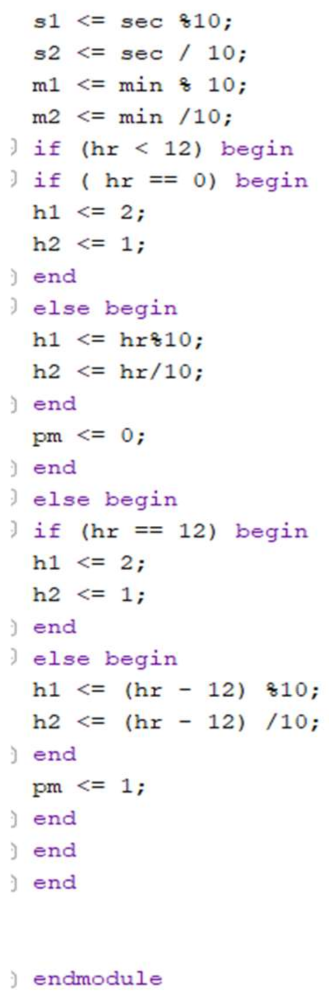
```
toggle <=0;
sec <=0;
msec <= 0;
end

if (counter < max_count) begin
counter <= counter +1;
end
else begin
counter <= 0;
msec <= msec +1;
end
end

set_time : begin
if (center) begin
clock_mode <= 1;
current_mode <= display_time;
end

if (counter < (25000000)) begin
counter <= counter + 1;
end
else begin
counter <= 0;
case (toggle)
1'b0: begin
if (up) begin
min <= min +1;
end
if (down) begin
```



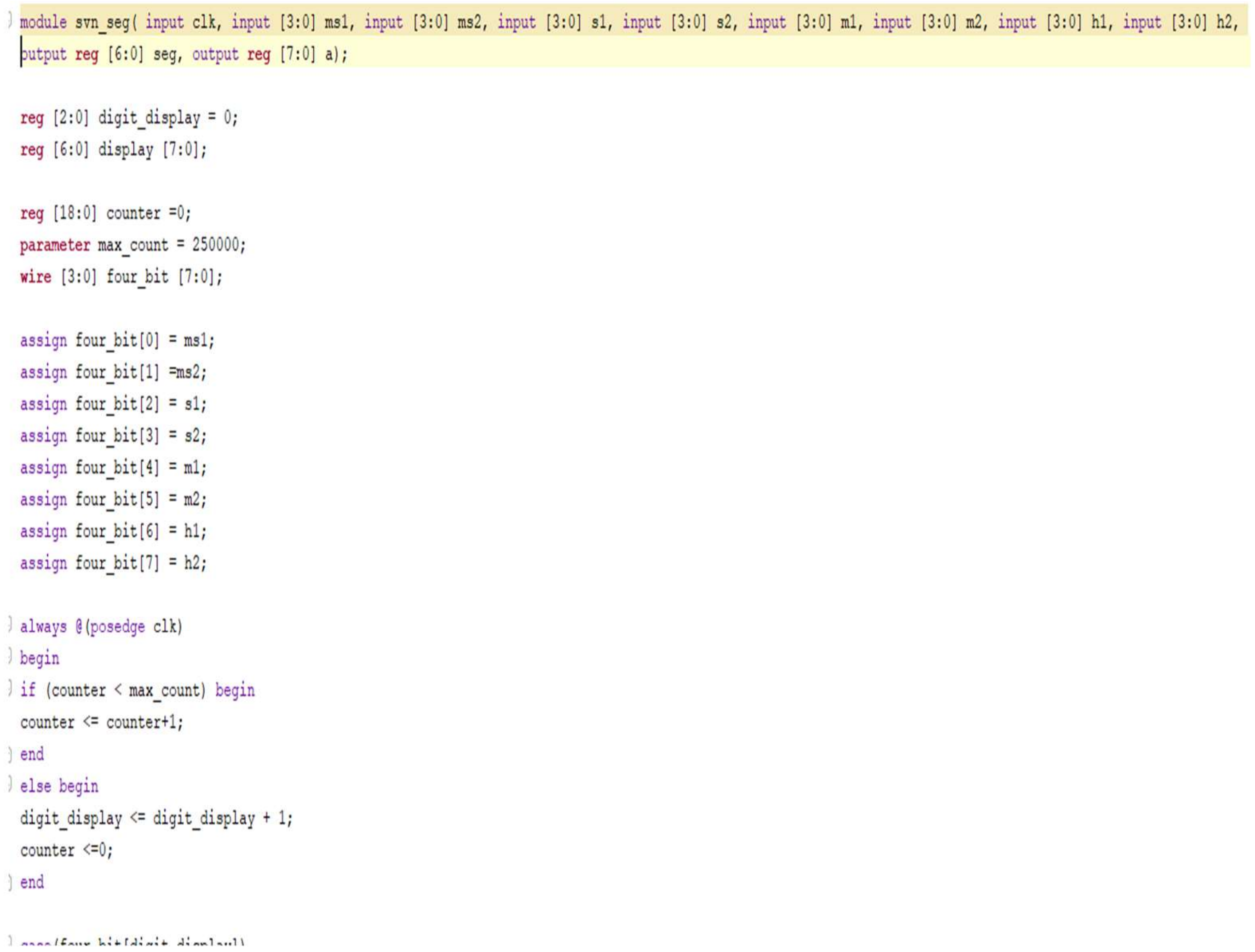
```
if (down) begin
if (min>0) begin
min <= min -1;
end
else if (hr >1) begin
hr <= hr -1;
min <=59;
end
else if (hr == 1) begin
hr <= 12;
min <= 59;
end
end

if (left || right ) begin
toggle <= 1;
end
end

1'b1: begin
if (up) begin
hr <= hr + 1;
end
if (down) begin
if (hr >1) begin
hr <= hr -1;
end
else if (hr == 1) begin
hr <= 12;
end
end
if (right || left) begin
```



```
end
if (right || left) begin
toggle <= 0;
end
end
endcase
end
end
endcase


if (msec >= 99) begin
msec <= 0;
sec <= sec +1;
end
if (sec >= 60) begin
sec <= 0;
min <= min +1;
end
if (min >= 60) begin
min <= 0;
hr <= hr +1;
end
if (hr >= 24) begin
hr <= 0;
end

else begin
ms1 <= msec %10;
ms2 <= msec /10;
s1 <= sec %10;
s2 <= sec / 10;
```



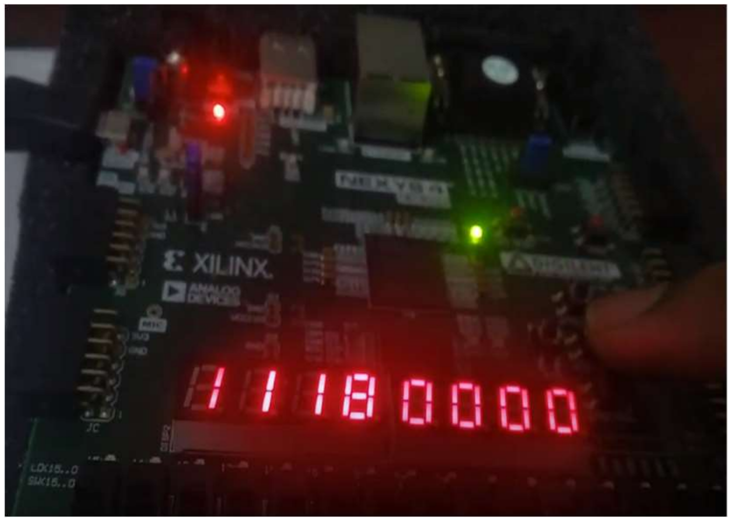
```
s1 <= sec %10;
s2 <= sec / 10;
m1 <= min % 10;
m2 <= min /10;
if (hr < 12) begin
if ( hr == 0) begin
h1 <= 2;
h2 <= 1;
end
else begin
h1 <= hr%10;
h2 <= hr/10;
end
pm <= 0;
end
else begin
if (hr == 12) begin
h1 <= 2;
h2 <= 1;
end
else begin
h1 <= (hr - 12) %10;
h2 <= (hr - 12) /10;
end
pm <= 1;
end
end
end


endmodule
```

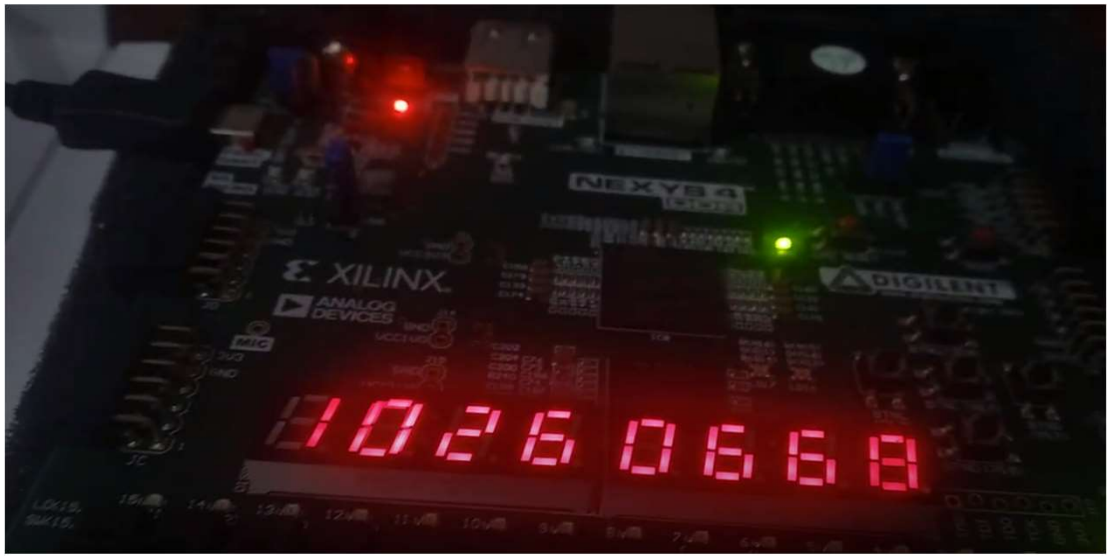


```
module svn_seg( input clk, input [3:0] ms1, input [3:0] ms2, input [3:0] s1, input [3:0] s2, input [3:0] m1, input [3:0] m2, input [3:0] h1, input [3:0] h2,
output reg [6:0] seg, output reg [7:0] a);

reg [2:0] digit_display = 0;
reg [6:0] display [7:0];

reg [18:0] counter =0;
parameter max_count = 250000;
wire [3:0] four_bit [7:0];

assign four_bit[0] = ms1;
assign four_bit[1] =ms2;
assign four_bit[2] = s1;
assign four_bit[3] = s2;
assign four_bit[4] = m1;
assign four_bit[5] = m2;
assign four_bit[6] = h1;
assign four_bit[7] = h2;

always @(posedge clk)
begin
if (counter < max_count) begin
counter <= counter+1;
end
else begin
digit_display <= digit_display + 1;
counter <=0;
end

case(four_bit[digit_display])
case(four_bit[digit_display])
4'b0000 : display[digit_display] <= 7'b1000000;
4'b0001 : display[digit_display] <= 7'b1111001;
4'b0010 : display[digit_display] <= 7'b0100100;
4'b0011 : display[digit_display] <= 7'b0110000;
4'b0100 : display[digit_display] <= 7'b0011001;
4'b0101 : display[digit_display] <= 7'b0010010;
4'b0110 : display[digit_display] <= 7'b0000010;
4'b0111 : display[digit_display] <= 7'b1111000;
4'b1000 : display[digit_display] <= 7'b0000000;
4'b1001 : display[digit_display] <= 7'b0011000;
default : display[digit_display] <= 7'b1111111;
endcase


case(digit_display)
0: begin
a <= 8'b11111110;
seg <= display[0];
end
1: begin
a <= 8'b11111101;
seg <= display[1];
end
2: begin
a <= 8'b11111011;
seg <= display[2];
end
3: begin
a <= 8'b11110111;
seg <= display[3];
```

```
3: begin
a <= 8'b11110111;
seg <= display[3];
end
4: begin
a <= 8'b11101111;
seg <= display[4];
end
5: begin
a <= 8'b11011111;
seg <= display[5];
end
6: begin
a <= 8'b10111111;
seg <= display[6];
end
7: begin
a <= 8'b0111111;
seg <= display[7];
end
endcase
end

endmodule
```

Results


XILINX
ANALOG DEVICES
11180000


NEXYS 4
XILINX
ANALOG DEVICES
DIGILENT
10260668

Thank You