



## Field-Programmable Gate Array (FPGA) Primer

**Day 4:**

## Verilog and Seven-Segment Displays

Sponsored by



## Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.



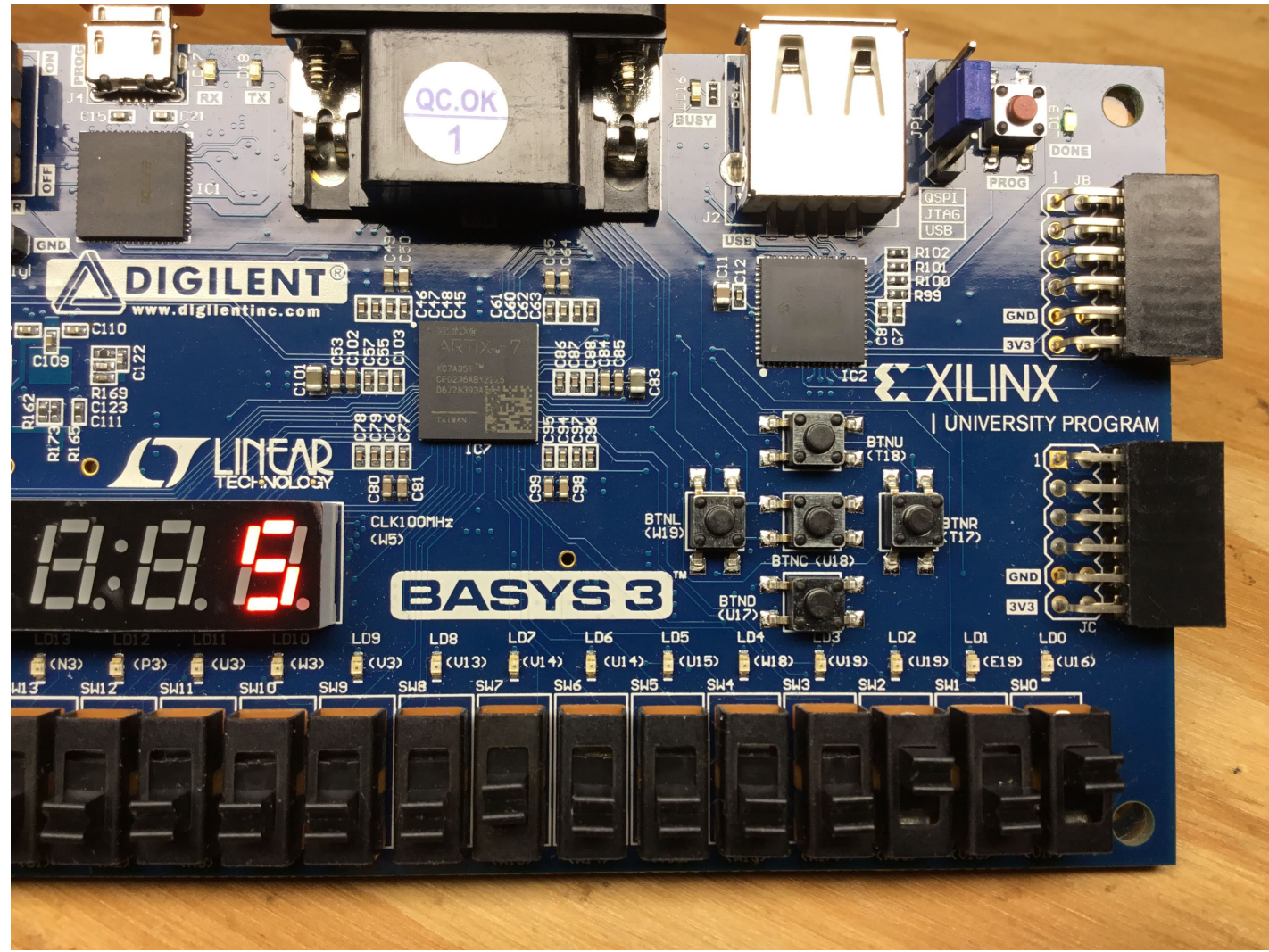
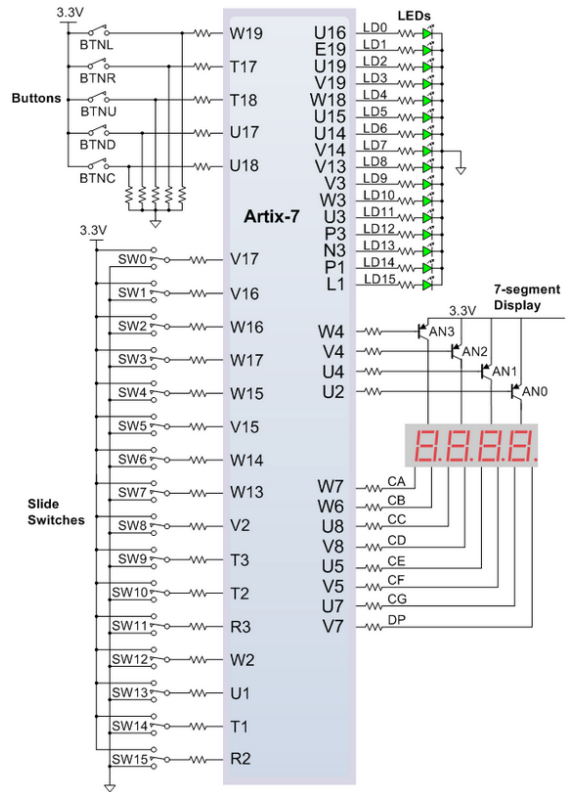
## Fred Eady

Visit 'Lecturer Profile' in your console for more details.



# AGENDA

- **Baysys 3 Seven-Segment Driver**
  - **refresh\_generator**
  - **digit\_driver**
  - **segment\_driver**

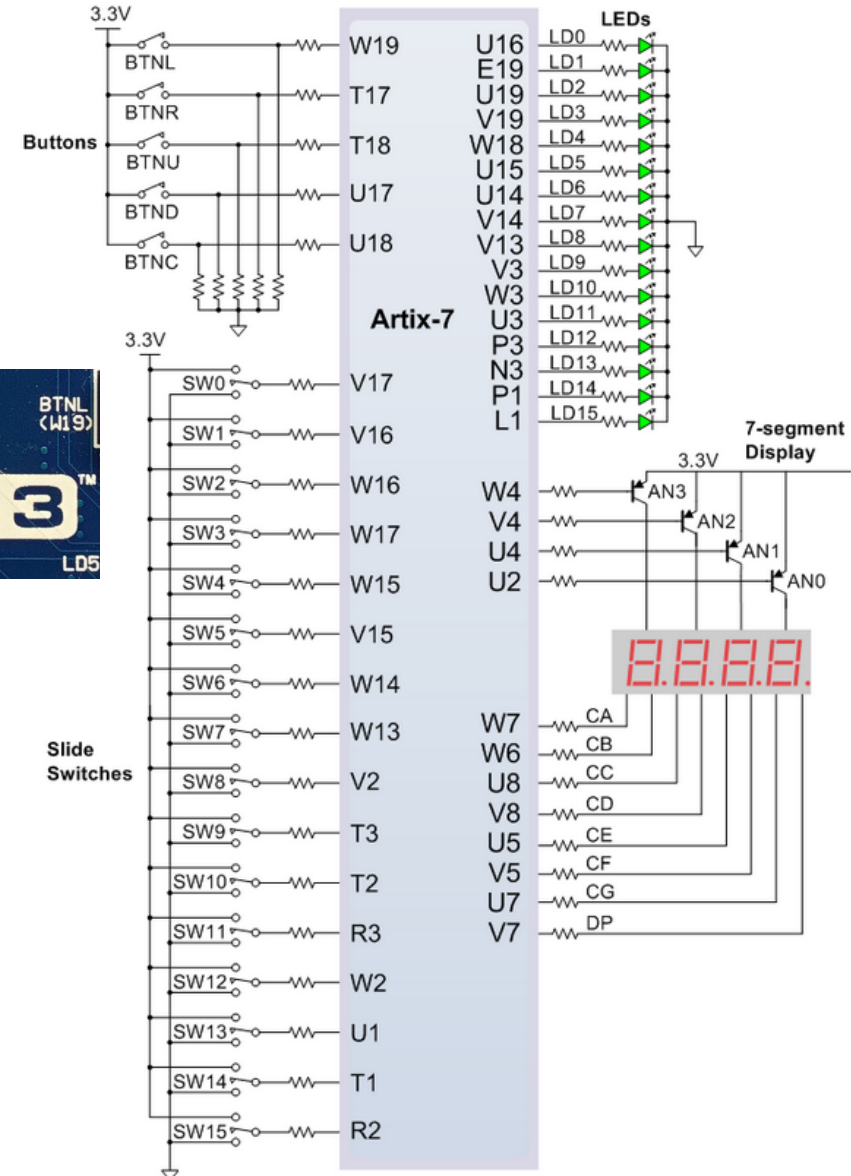
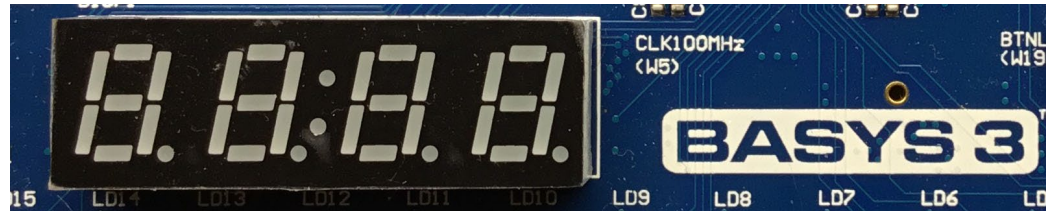




# seven\_seg\_driver\_top.v

```

23 module seven_seg_driver_top(
24     input clk,           //100MHz clock
25     input [15:0] sw,    //slide switches
26     output [6:0] seg,   //LED segments
27     output [3:0] an     //LED anodes
28 );
29
30 wire [1:0] w_refresh_out;
31 wire [3:0] w_anode_select;
32 wire [6:0] w_segment_select;
33 wire [3:0] w_switches_out;
34
35 refresh_generator ref_gen(
36     .clk(clk),
37     .reset(btnC),
38     .refresh_out(w_refresh_out)
39 );
40
41 digit_driver digit_drv(
42     .refresh_in(w_refresh_out),
43     .switches_in(sw),
44     .anode_select(w_anode_select),
45     .switches_out(w_switches_out)
46 );
47
48 segment_driver segment_drv(
49     .hex(w_switches_out),
50     .segment_out(w_segment_select)
51 );
52
53 assign an = w_anode_select;
54 assign seg = w_segment_select;
55 endmodule
  
```



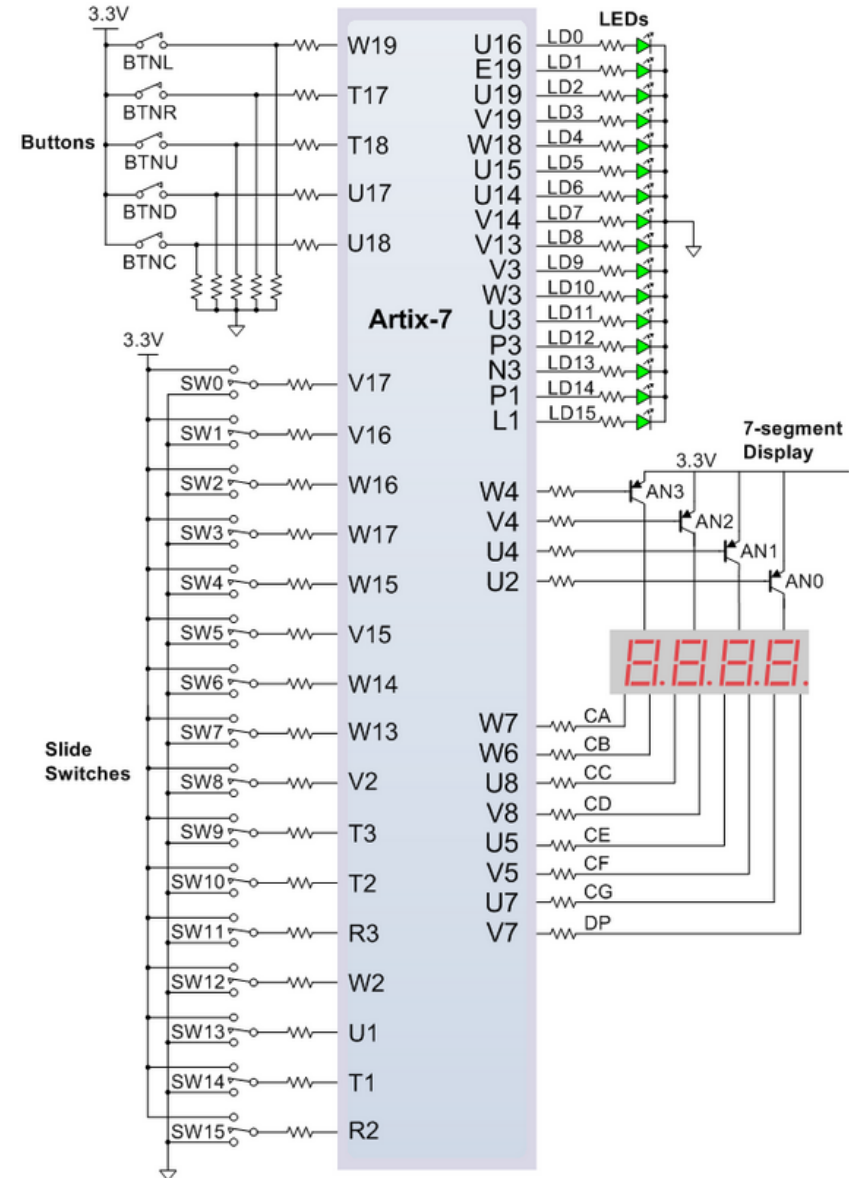
# Digit Refresh Generator

```

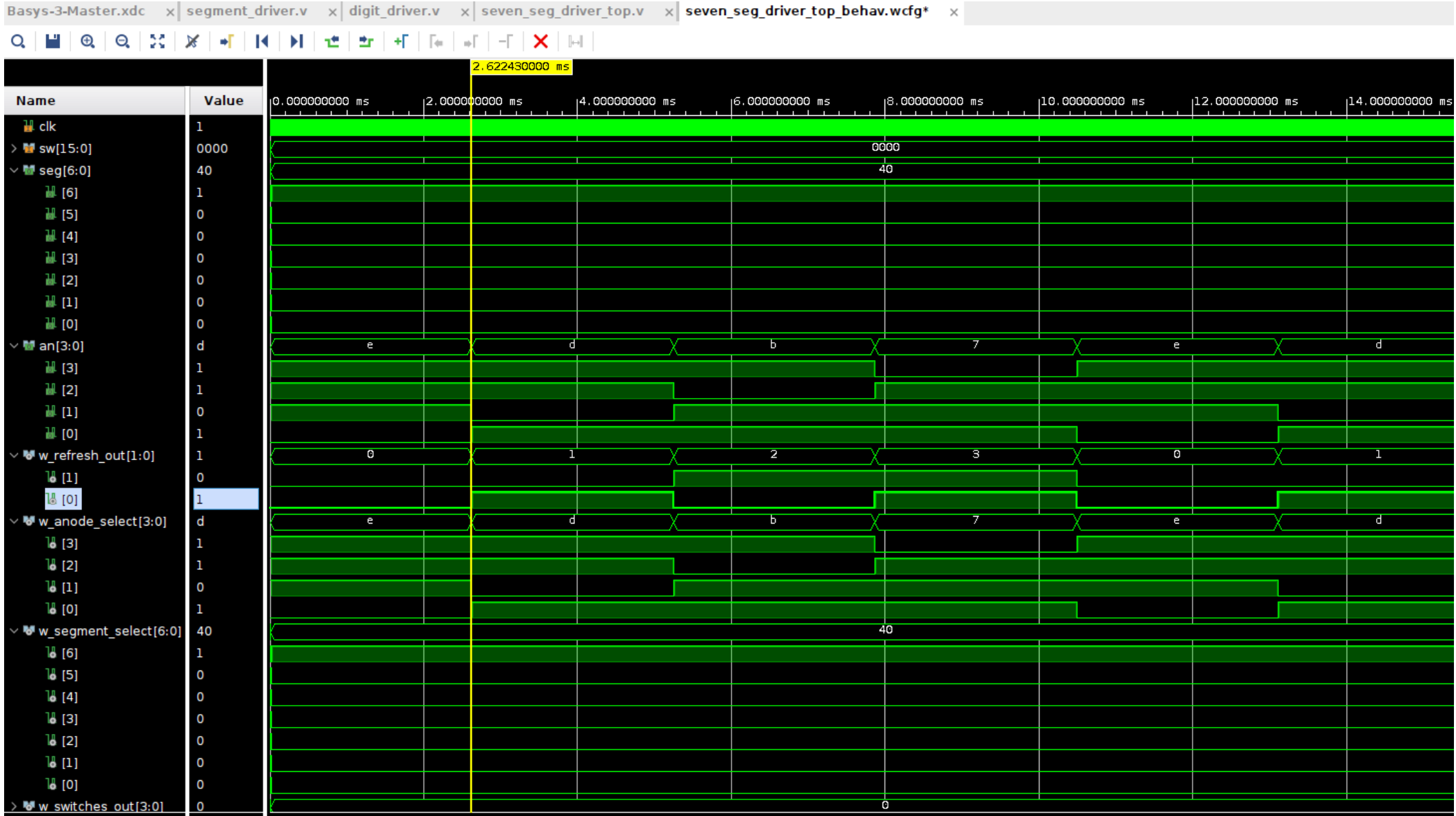
23 module refresh_generator(
24     input clk,
25     input reset,
26     output [1:0] refresh_out
27 );
28
29 // bits [17:0] overflow every 2.62ms
30 // bits [19:18] form a 4-bit counter that increments every 2.62ms
31 reg [19:0] refresh_counter = 0;
32 always @ (posedge clk)
33 begin
34     if(reset == 1)
35         refresh_counter <= 0;
36     else
37         refresh_counter <= refresh_counter + 1;
38 end
39 assign refresh_out[1:0] = refresh_counter[19:18];
40
41 endmodule
    
```

$2^{18} = 262,144$

Overflow time in ms =  $262,144 / 100,000,000,000 = 2.62144\text{ms}$

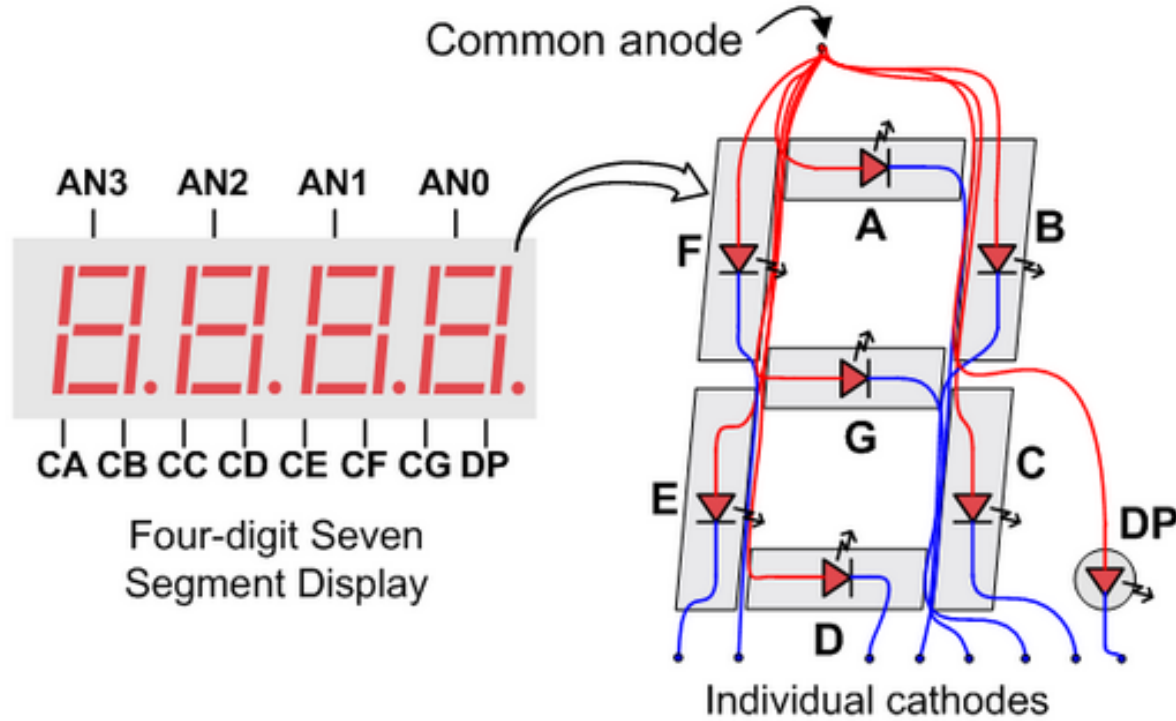


## 2.62ms Per Digit Refresh Period





# Seven-Segment Display Pin Assignments



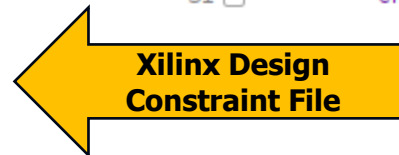
```

50 set_property -dict { PACKAGE_PIN W7 IOSTANDARD LVCMOS33 } [get_ports {seg[0]}]
51 set_property -dict { PACKAGE_PIN W6 IOSTANDARD LVCMOS33 } [get_ports {seg[1]}]
52 set_property -dict { PACKAGE_PIN U8 IOSTANDARD LVCMOS33 } [get_ports {seg[2]}]
53 set_property -dict { PACKAGE_PIN V8 IOSTANDARD LVCMOS33 } [get_ports {seg[3]}]
54 set_property -dict { PACKAGE_PIN U5 IOSTANDARD LVCMOS33 } [get_ports {seg[4]}]
55 set_property -dict { PACKAGE_PIN V5 IOSTANDARD LVCMOS33 } [get_ports {seg[5]}]
56 set_property -dict { PACKAGE_PIN U7 IOSTANDARD LVCMOS33 } [get_ports {seg[6]}]
57
58 #set_property -dict { PACKAGE_PIN V7 IOSTANDARD LVCMOS33 } [get_ports dp]
59
60 set_property -dict { PACKAGE_PIN U2 IOSTANDARD LVCMOS33 } [get_ports {an[0]}]
61 set_property -dict { PACKAGE_PIN U4 IOSTANDARD LVCMOS33 } [get_ports {an[1]}]
62 set_property -dict { PACKAGE_PIN V4 IOSTANDARD LVCMOS33 } [get_ports {an[2]}]
63 set_property -dict { PACKAGE_PIN W4 IOSTANDARD LVCMOS33 } [get_ports {an[3]}]
    
```

A  
B  
C  
D  
E  
F  
G

```

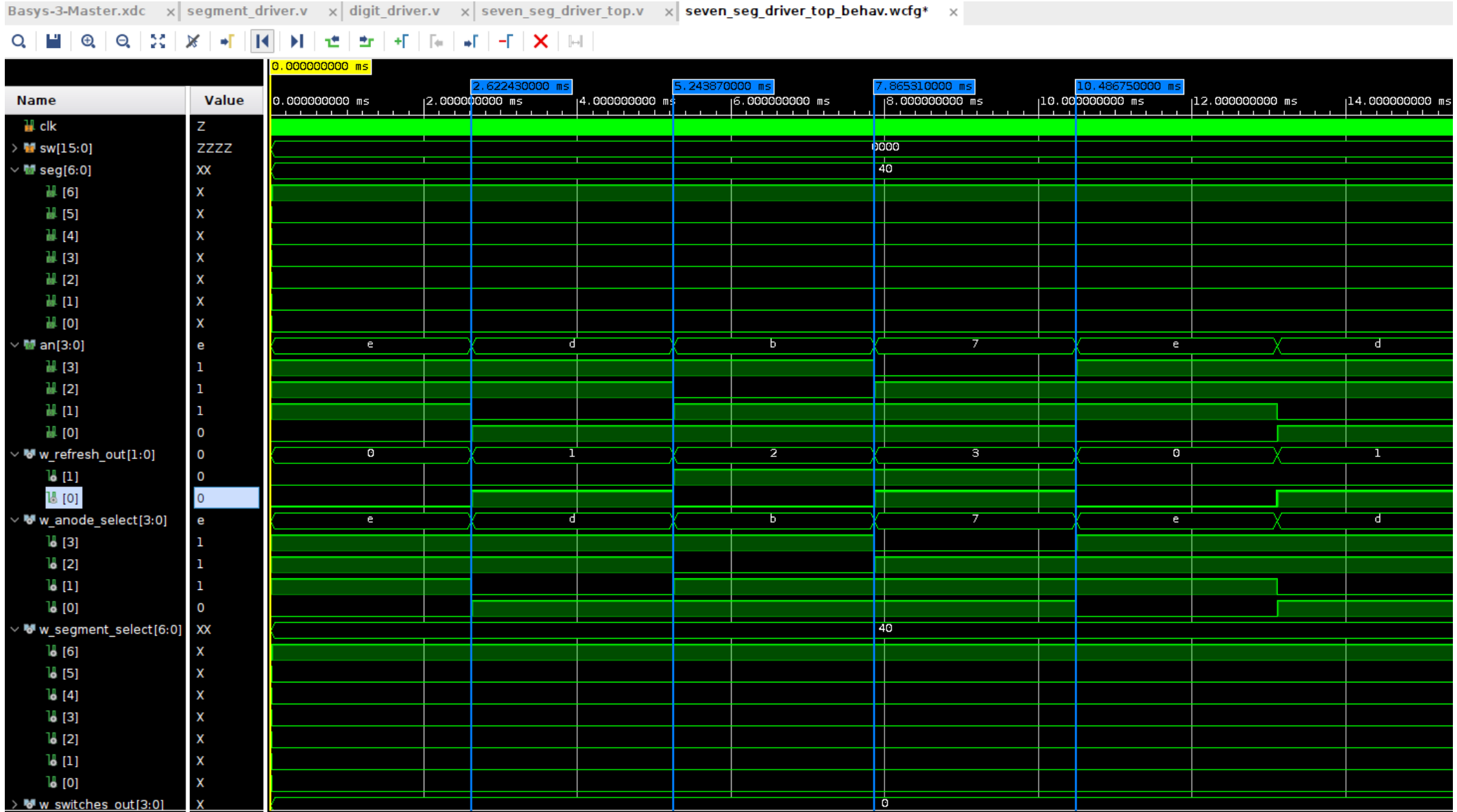
23 // seven-segment digit display driver
24
25 module sseg_display (
26     input [3:0] hex_in,
27     output reg [6:0] seg_out
28 );
29
30 // activate on any input change
31 always @*
32 begin
33     case(hex_in)
34         4'h0: seg_out <= 7'b1000000; // digit 0
35         4'h1: seg_out <= 7'b1111001; // digit 1
36         4'h2: seg_out <= 7'b0100100; // digit 2
37         4'h3: seg_out <= 7'b0110000; // digit 3
38         4'h4: seg_out <= 7'b0011001; // digit 4
39         4'h5: seg_out <= 7'b0010010; // digit 5
40         4'h6: seg_out <= 7'b0000010; // digit 6
41         4'h7: seg_out <= 7'b1111000; // digit 7
42         4'h8: seg_out <= 7'b0000000; // digit 8
43         4'h9: seg_out <= 7'b0010000; // digit 9
44         4'ha: seg_out <= 7'b0001000; // digit A
45         4'hb: seg_out <= 7'b0000011; // digit B
46         4'hc: seg_out <= 7'b1000110; // digit C
47         4'hd: seg_out <= 7'b0100001; // digit D
48         4'he: seg_out <= 7'b0000110; // digit E
49         4'hf: seg_out <= 7'b0001110; // digit F
50         default: seg_out <= 7'b1111111; // digit OFF
51     endcase
    
```







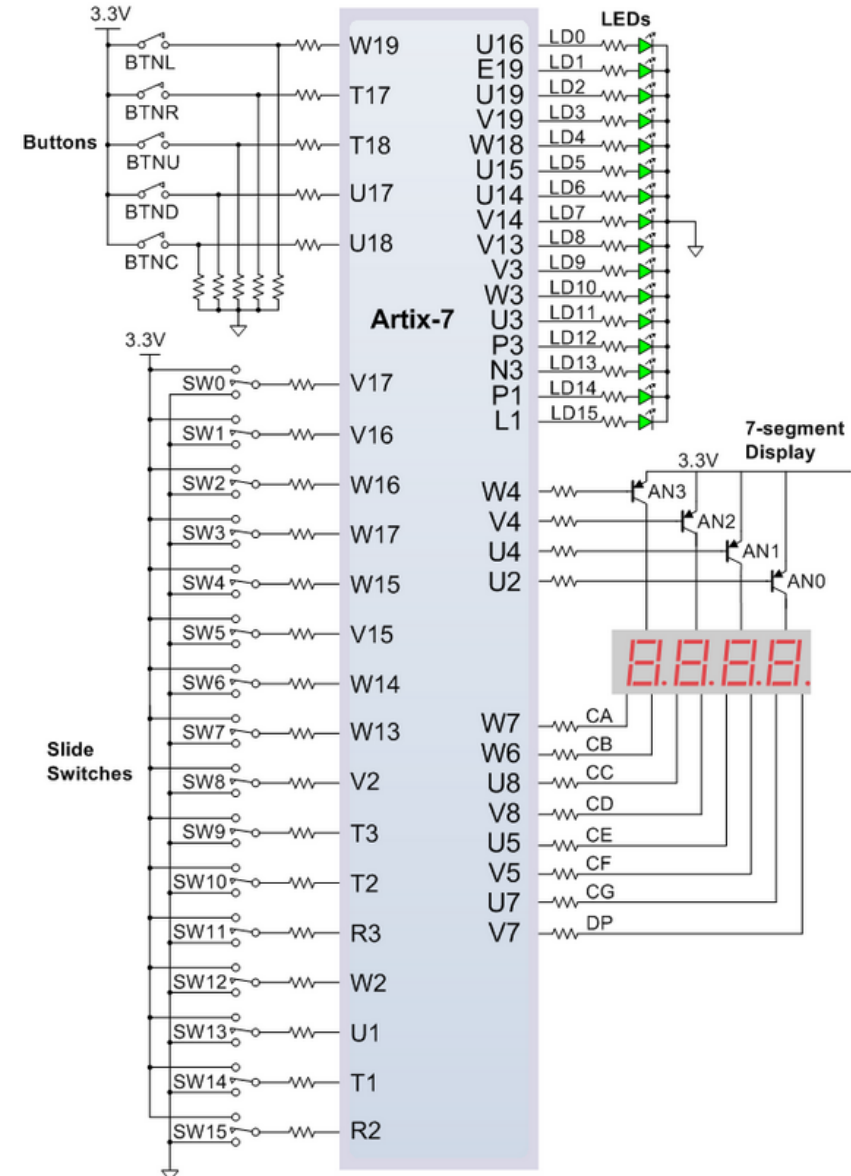
# Digit Refresh Generator – 4-Digit Scan Time



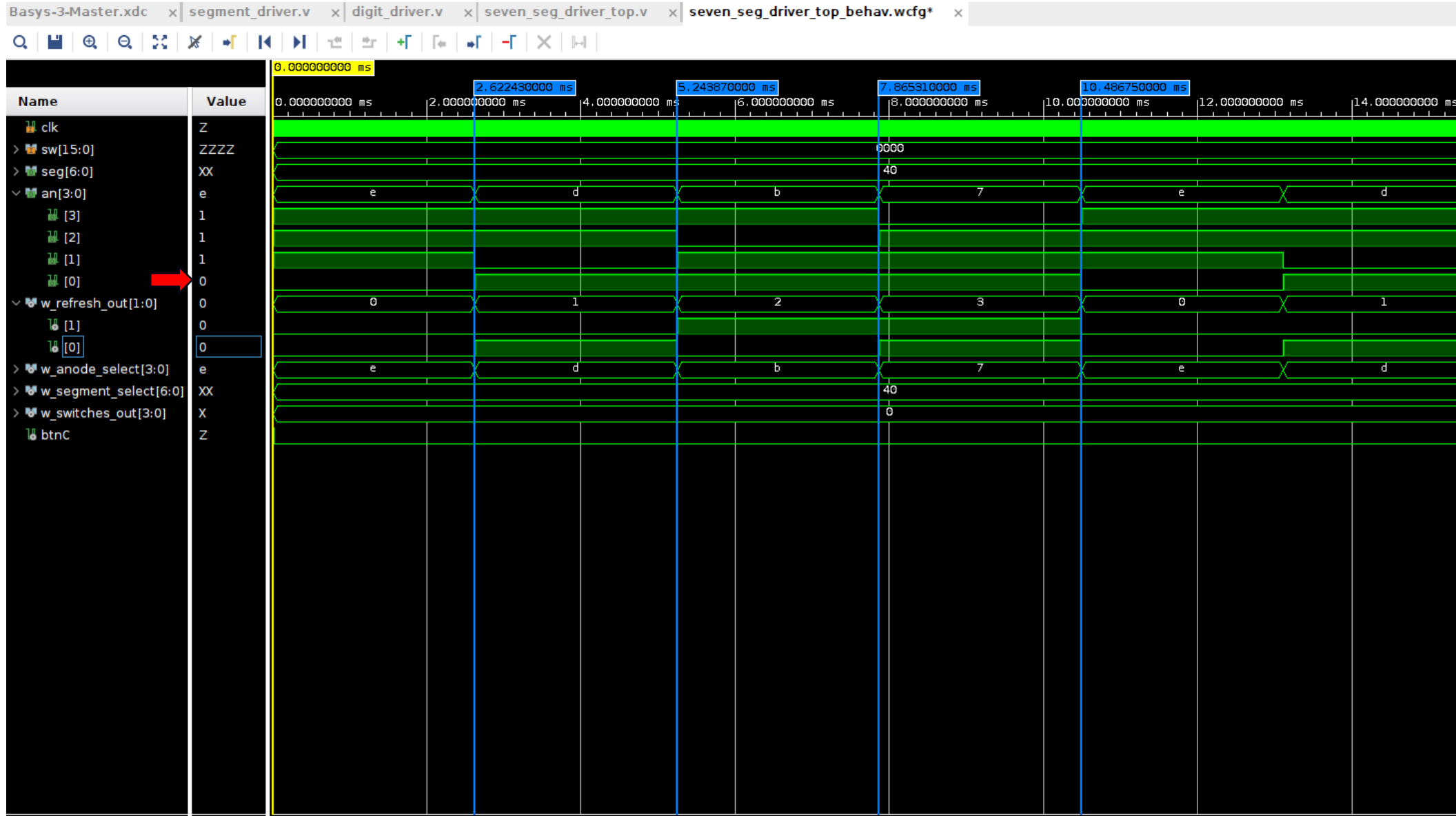
# Digit Driver

```

23 | module digit_driver(
24 |     input [1:0] refresh_in,
25 |     input [15:0] switches_in,
26 |     output reg [3:0] anode_select,
27 |     output reg [3:0] switches_out
28 | );
29 |
30 | always @*
31 | begin
32 |     case(refresh_in)
33 |     2'b00:
34 |         begin
35 |             anode_select = 4'b1110;
36 |             switches_out = switches_in & 16'h000F;
37 |         end
38 |     2'b01:
39 |         begin
40 |             anode_select = 4'b1101;
41 |             switches_out = (switches_in & 16'h00F0) >> 4;
42 |         end
43 |     2'b10:
44 |         begin
45 |             anode_select = 4'b1011;
46 |             switches_out = (switches_in & 16'h0F00) >> 8;
47 |         end
48 |     2'b11:
49 |         begin
50 |             anode_select = 4'b0111;
51 |             switches_out = (switches_in & 16'hF000) >> 12;
52 |         end
53 |     endcase
54 | end
55 | endmodule
    
```

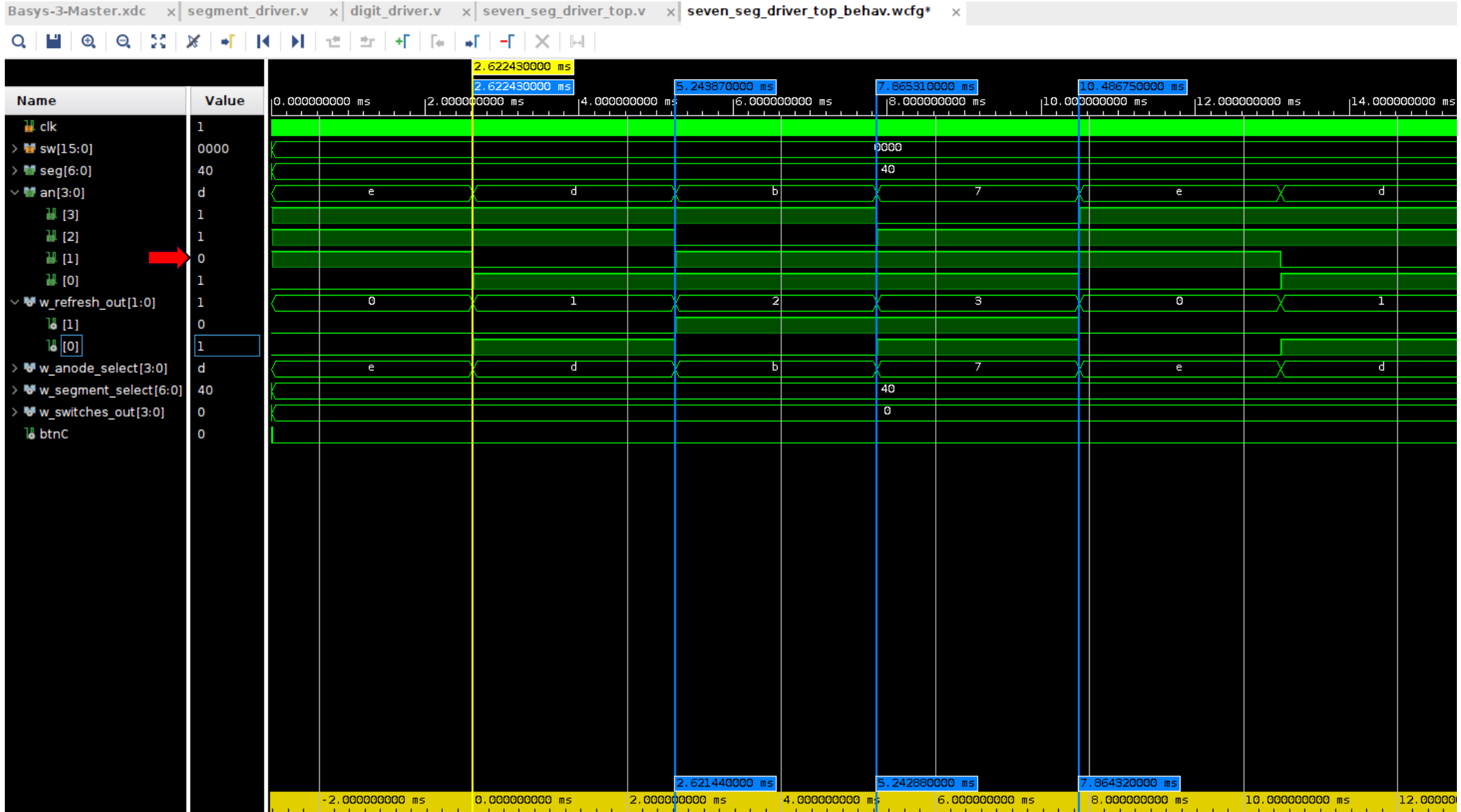


# Digit Driver – Anode 0

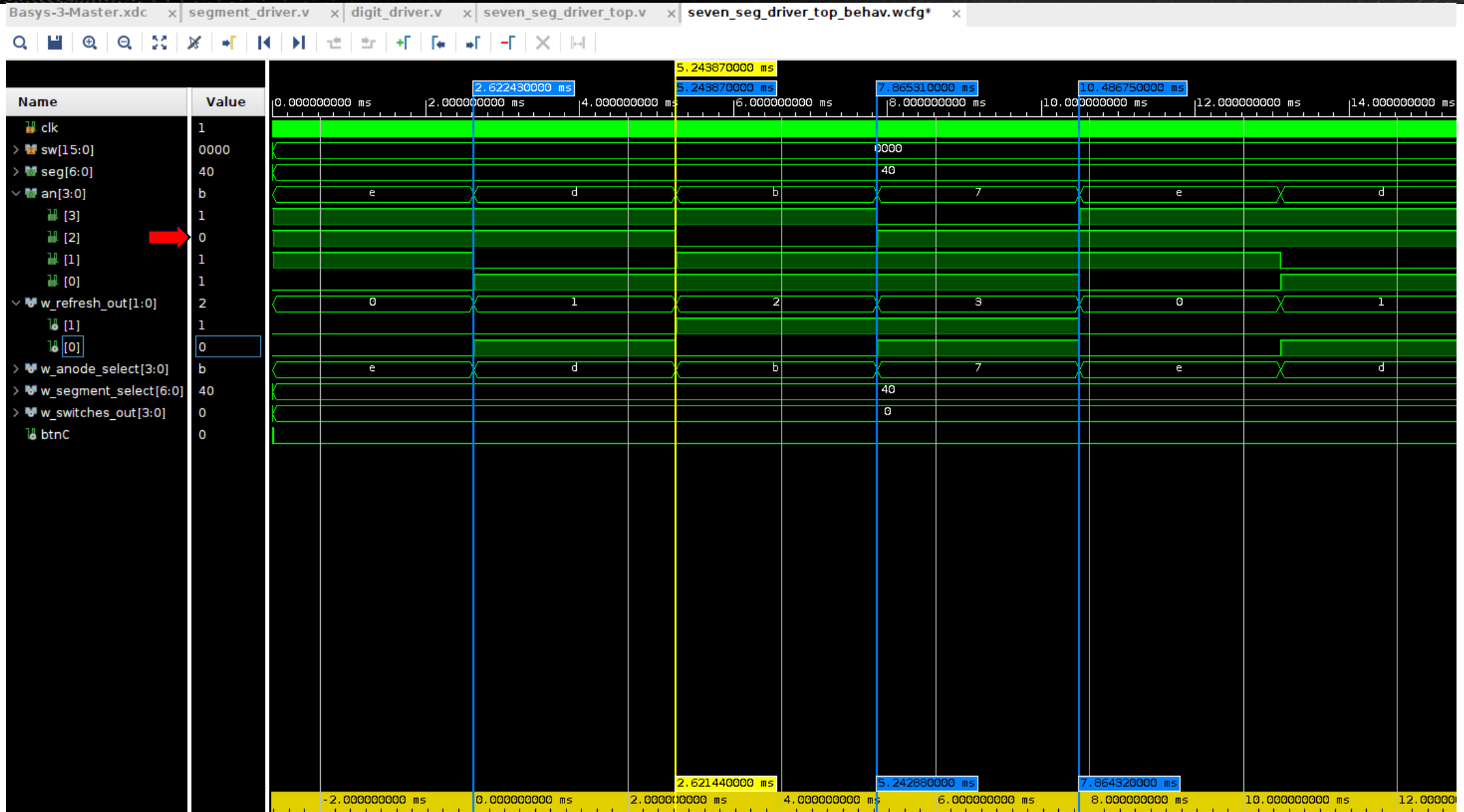




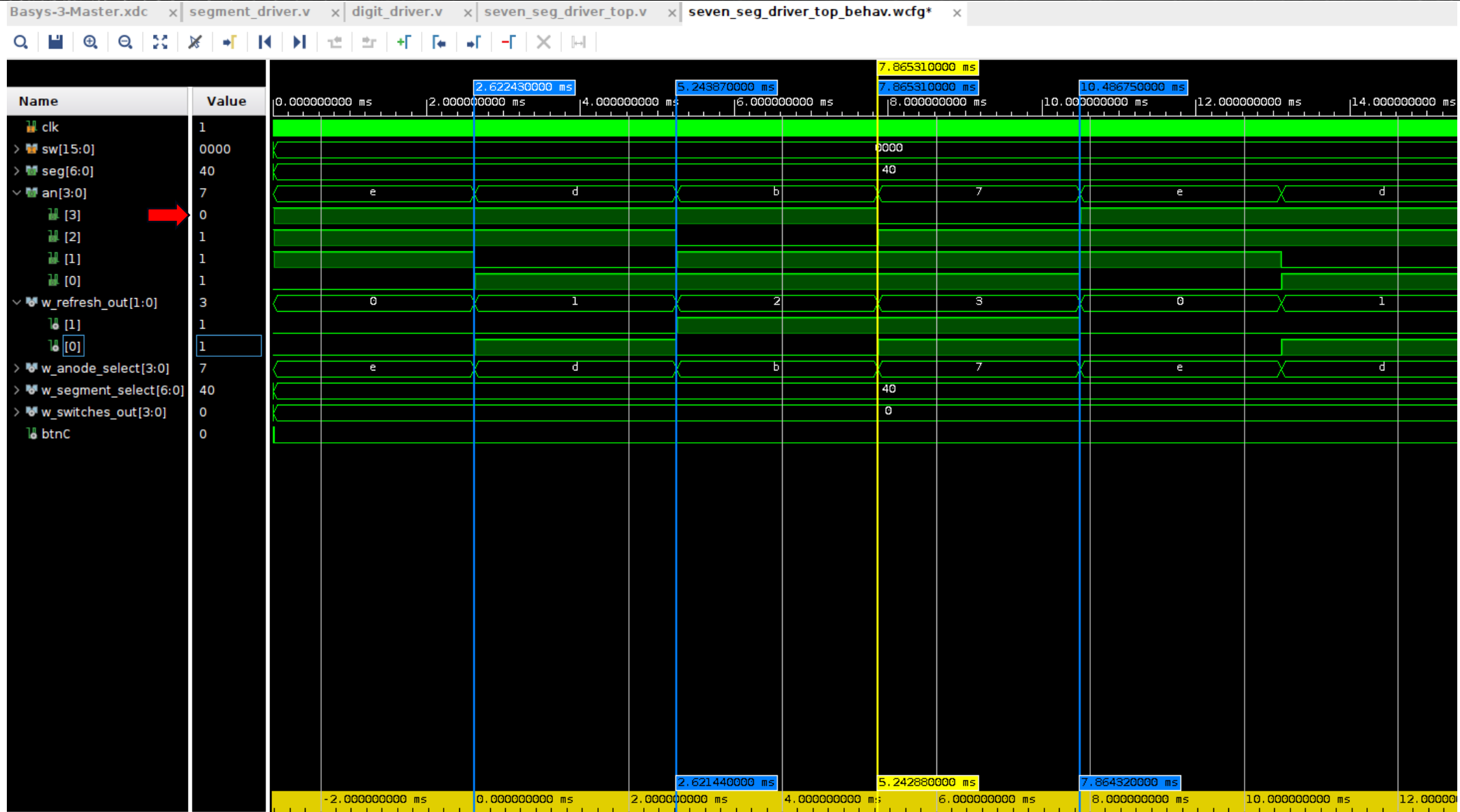
# Digit Driver – Anode 1



## Digit Driver – Anode 2



# Digit Driver – Anode 3

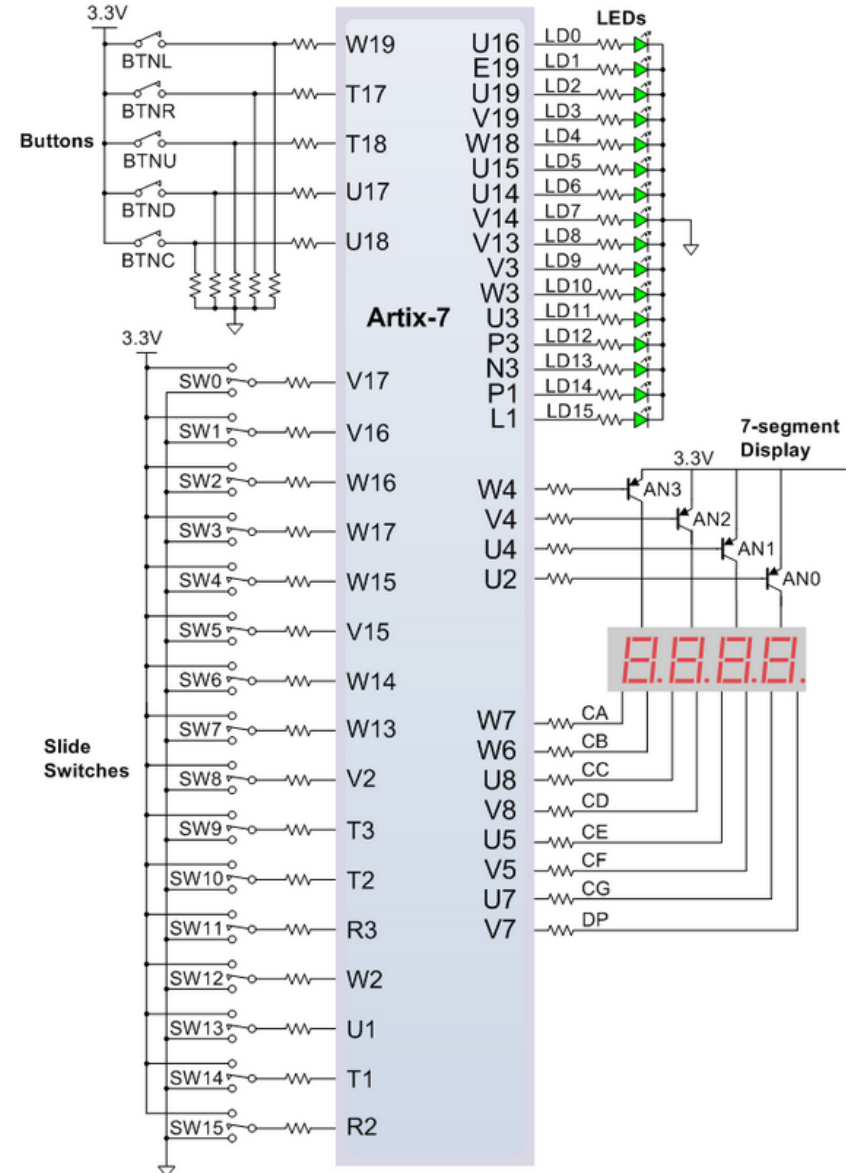




# Segment Driver

```

23 module segment_driver(
24     input [3:0] hex,
25     output reg [6:0] segment_out
26 );
27
28     // activate on any input change
29     always @*
30     begin
31         case(hex)
32             4'h0: segment_out <= 7'b1000000; // digit 0
33             4'h1: segment_out <= 7'b1111001; // digit 1
34             4'h2: segment_out <= 7'b0100100; // digit 2
35             4'h3: segment_out <= 7'b0110000; // digit 3
36             4'h4: segment_out <= 7'b0011001; // digit 4
37             4'h5: segment_out <= 7'b0010010; // digit 5
38             4'h6: segment_out <= 7'b0000010; // digit 6
39             4'h7: segment_out <= 7'b1111000; // digit 7
40             4'h8: segment_out <= 7'b0000000; // digit 8
41             4'h9: segment_out <= 7'b0010000; // digit 9
42             4'ha: segment_out <= 7'b0001000; // digit A
43             4'hb: segment_out <= 7'b0000011; // digit B
44             4'hc: segment_out <= 7'b1000110; // digit C
45             4'hd: segment_out <= 7'b0100001; // digit D
46             4'he: segment_out <= 7'b0000110; // digit E
47             4'hf: segment_out <= 7'b0001110; // digit F
48             default: segment_out <= 7'b1111111; // all segments OFF
49         endcase
50     end
51 endmodule
    
```



## Segment Driver

### Force Clock: /seven\_seg\_driver\_top/clk

Enter parameters below to force the signal to a constant value. Assignments made from within HDL code or any previously applied constant or clock force will be overridden.

Signal name: /seven\_seg\_driver\_top/clk

Value radix: Hexadecimal

Leading edge value: 1

Trailing edge value: 0

Starting after time offset: 0ns

Cancel after time offset:

Duty cycle (%): 50

Period: 10ns

### Force Constant: /seven\_seg\_driver\_top/sw

Enter parameters below to force the signal to a constant value. Assignments made from within HDL code or any previously applied constant or clock force will be overridden.

Signal name: /seven\_seg\_driver\_top/sw

Value radix: Hexadecimal

Force value: 0x3210

Starting after time offset: 0ns

Cancel after time offset:

### Force Constant: /seven\_seg\_driver\_top/btnC

Enter parameters below to force the signal to a constant value. Assignments made from within HDL code or any previously applied constant or clock force will be overridden.

Signal name: /seven\_seg\_driver\_top/btnC

Value radix: Hexadecimal

Force value: 0

Starting after time offset: 0ns

Cancel after time offset:

# Segment Driver - 0

Bsys-3-Master.xdc x segment\_driver.v x digit\_driver.v x seven\_seg\_driver\_top.v x seven\_seg\_driver\_top\_behav.wcfg x

0.000000000 ms 2.622430000 ms 5.243870000 ms 7.865310000 ms 10.486750000 ms 13.108190000 ms

Name	Value
clk	Z
sw[15:0]	ZZZZ
seg[6:0]	XX
[6]	X
[5]	X
[4]	X
[3]	X
[2]	X
[1]	X
[0]	X
an[3:0]	e
w_refresh_out[1:0]	0
w_anode_select[3:0]	e
w_segment_select[6:0]	XX
w_switches_out[3:0]	X
btnC	Z

```

23 module segment_driver(
24     input [3:0] hex,
25     output reg [6:0] segment_out
26 );
27
28 // activate on any input change
29 always @*
30 begin
31     case(hex)
32         4'h0: segment_out <= 7'b1000000; // digit 0
33         4'h1: segment_out <= 7'b1111001; // digit 1
34         4'h2: segment_out <= 7'b0100100; // digit 2
35         4'h3: segment_out <= 7'b0110000; // digit 3
36         4'h4: segment_out <= 7'b0011001; // digit 4
37         4'h5: segment_out <= 7'b0010010; // digit 5
38         4'h6: segment_out <= 7'b0000010; // digit 6
39         4'h7: segment_out <= 7'b1111000; // digit 7
40         4'h8: segment_out <= 7'b0000000; // digit 8
41         4'h9: segment_out <= 7'b0010000; // digit 9
42         4'ha: segment_out <= 7'b0001000; // digit A
43         4'hb: segment_out <= 7'b0000011; // digit B
44         4'hc: segment_out <= 7'b1000110; // digit C
45         4'hd: segment_out <= 7'b0100001; // digit D
46         4'he: segment_out <= 7'b0000110; // digit E
47         4'hf: segment_out <= 7'b0001110; // digit F
48         default: segment_out <= 7'b1111111; // all segments OFF
49     endcase
50 end
51 endmodule
  
```

Timing diagram showing signal transitions for clk, sw, seg, an, w\_refresh\_out, w\_anode\_select, w\_segment\_select, w\_switches\_out, and btnC. The segment output (seg) is shown as a sequence of characters: e, 0, e, 40, 0.



# Segment Driver - 1

Basy3-3-Master.xdc x segment\_driver.v x digit\_driver.v x seven\_seg\_driver\_top.v x seven\_seg\_driver\_top\_behav.wcfg x

0.000000000 ms 2.000000000 ms 4.000000000 ms 6.000000000 ms 8.000000000 ms 10.000000000 ms 12.000000000 ms 14.000000000 ms

Name	Value
clk	1
sw[15:0]	3210
seg[6:0]	79
[6]	1
[5]	1
[4]	1
[3]	1
[2]	0
[1]	0
[0]	1
an[3:0]	d
w_refresh_out[1:0]	1
w_anode_select[3:0]	d
w_segment_select[6:0]	79
w_switches_out[3:0]	1
btnC	0

```

23 module segment_driver(
24     input [3:0] hex,
25     output reg [6:0] segment_out
26 );
27
28 // activate on any input change
29 always @*
30 begin
31     case(hex)
32         4'h0: segment_out <= 7'b1000000; // digit 0
33         4'h1: segment_out <= 7'b1111001; // digit 1
34         4'h2: segment_out <= 7'b0100100; // digit 2
35         4'h3: segment_out <= 7'b0110000; // digit 3
36         4'h4: segment_out <= 7'b0011001; // digit 4
37         4'h5: segment_out <= 7'b0010010; // digit 5
38         4'h6: segment_out <= 7'b0000010; // digit 6
39         4'h7: segment_out <= 7'b1111000; // digit 7
40         4'h8: segment_out <= 7'b0000000; // digit 8
41         4'h9: segment_out <= 7'b0010000; // digit 9
42         4'ha: segment_out <= 7'b0001000; // digit A
43         4'hb: segment_out <= 7'b0000011; // digit B
44         4'hc: segment_out <= 7'b1000110; // digit C
45         4'hd: segment_out <= 7'b0100001; // digit D
46         4'he: segment_out <= 7'b0000110; // digit E
47         4'hf: segment_out <= 7'b0001110; // digit F
48         default: segment_out <= 7'b1111111; // all segments OFF
49     endcase
50 end
51 endmodule

```

Timeline showing signal transitions for clk, sw[15:0], seg[6:0], an[3:0], w\_refresh\_out[1:0], w\_anode\_select[3:0], w\_segment\_select[6:0], w\_switches\_out[3:0], and btnC. The seg[6:0] signal is highlighted in green, showing a transition from 40 to 79 at 2.622430000 ms.

# Segment Driver - 2

Basys-3-Master.xdc x segment\_driver.v x digit\_driver.v x seven\_seg\_driver\_top.v x seven\_seg\_driver\_top\_behav.wcfg x

clk 1  
sw[15:0] 3210  
seg[6:0] 24  
[6] 0  
[5] 1  
[4] 0  
[3] 0  
[2] 1  
[1] 0  
[0] 0  
an[3:0] b  
w\_refresh\_out[1:0] 2  
w\_anode\_select[3:0] b  
w\_segment\_select[6:0] 24  
w\_switches\_out[3:0] 2  
btnC 0

```
module segment_driver(  
    input [3:0] hex,  
    output reg [6:0] segment_out  
);  
  
    // activate on any input change  
    always @*  
    begin  
        case(hex)  
            4'h0: segment_out <= 7'b1000000; // digit 0  
            4'h1: segment_out <= 7'b1111001; // digit 1  
            4'h2: segment_out <= 7'b0100100; // digit 2  
            4'h3: segment_out <= 7'b0110000; // digit 3  
            4'h4: segment_out <= 7'b0011001; // digit 4  
            4'h5: segment_out <= 7'b0010010; // digit 5  
            4'h6: segment_out <= 7'b0000010; // digit 6  
            4'h7: segment_out <= 7'b1111000; // digit 7  
            4'h8: segment_out <= 7'b0000000; // digit 8  
            4'h9: segment_out <= 7'b0010000; // digit 9  
            4'ha: segment_out <= 7'b0001000; // digit A  
            4'hb: segment_out <= 7'b0000011; // digit B  
            4'hc: segment_out <= 7'b1000110; // digit C  
            4'hd: segment_out <= 7'b0100001; // digit D  
            4'he: segment_out <= 7'b0000110; // digit E  
            4'hf: segment_out <= 7'b0001110; // digit F  
            default: segment_out <= 7'b1111111; // all segments OFF  
        endcase  
    end  
endmodule
```

# Segment Driver - 3

Project Files: Basy3-3-Master.xdc | segment\_driver.v | digit\_driver.v | seven\_seg\_driver\_top.v | seven\_seg\_driver\_top\_behav.wcfg

Time (ms)	sw[15:0]	seg[6:0]
0.000000000	3210	40
2.622430000	3210	79
5.243870000	3210	24
7.865310000	3210	30
10.486750000	3210	40
13.108190000	3210	79

```

23 module segment_driver(
24     input [3:0] hex,
25     output reg [6:0] segment_out
26 );
27
28 // activate on any input change
29 always @*
30 begin
31     case(hex)
32         4'h0: segment_out <= 7'b100000; // digit 0
33         4'h1: segment_out <= 7'b1111001; // digit 1
34         4'h2: segment_out <= 7'b0100100; // digit 2
35         4'h3: segment_out <= 7'b0110000; // digit 3
36         4'h4: segment_out <= 7'b0011001; // digit 4
37         4'h5: segment_out <= 7'b0010010; // digit 5
38         4'h6: segment_out <= 7'b0000010; // digit 6
39         4'h7: segment_out <= 7'b1111000; // digit 7
40         4'h8: segment_out <= 7'b0000000; // digit 8
41         4'h9: segment_out <= 7'b0010000; // digit 9
42         4'ha: segment_out <= 7'b0001000; // digit A
43         4'hb: segment_out <= 7'b0000011; // digit B
44         4'hc: segment_out <= 7'b1000110; // digit C
45         4'hd: segment_out <= 7'b0100001; // digit D
46         4'he: segment_out <= 7'b0000110; // digit E
47         4'hf: segment_out <= 7'b0001110; // digit F
48         default: segment_out <= 7'b1111111; // all segments OFF
49     endcase
50 end
51 endmodule
  
```

Timing diagram showing signal transitions for clk, sw[15:0], and seg[6:0]. The diagram shows a sequence of input changes to the hex input, resulting in corresponding changes to the segment output. A red arrow points to the 4'h3 case in the Verilog code.



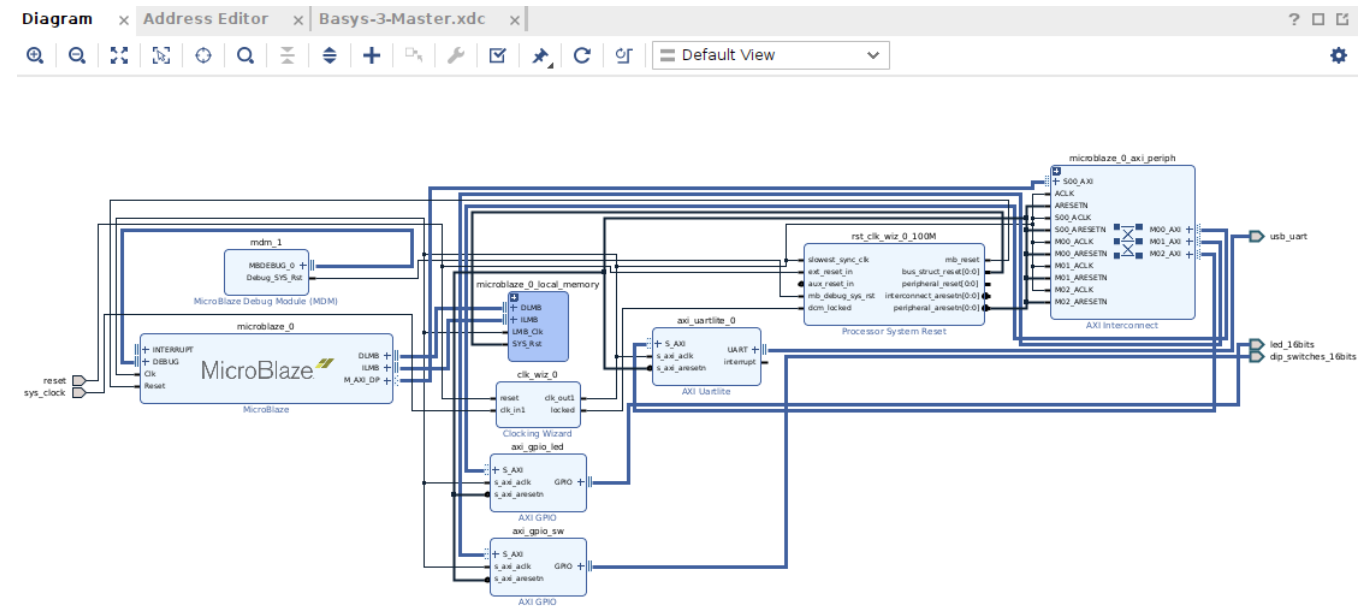


**MORE TO COME..**

# Thank you for attending!!!

Please consider the resources below:

- [xilinx.com](http://xilinx.com)
- [digilent.com](http://digilent.com)
- [Basys 3 Reference Manual](#)







**DesignNews**

Thank You

Sponsored by

