CEC Continuing Education Center

DesignNews

Field-Programmable Gate Array (FPGA) Primer

Day 2:
Verilog 101

Sponsored by

Digi-Key ELECTRONICS

informa markets

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

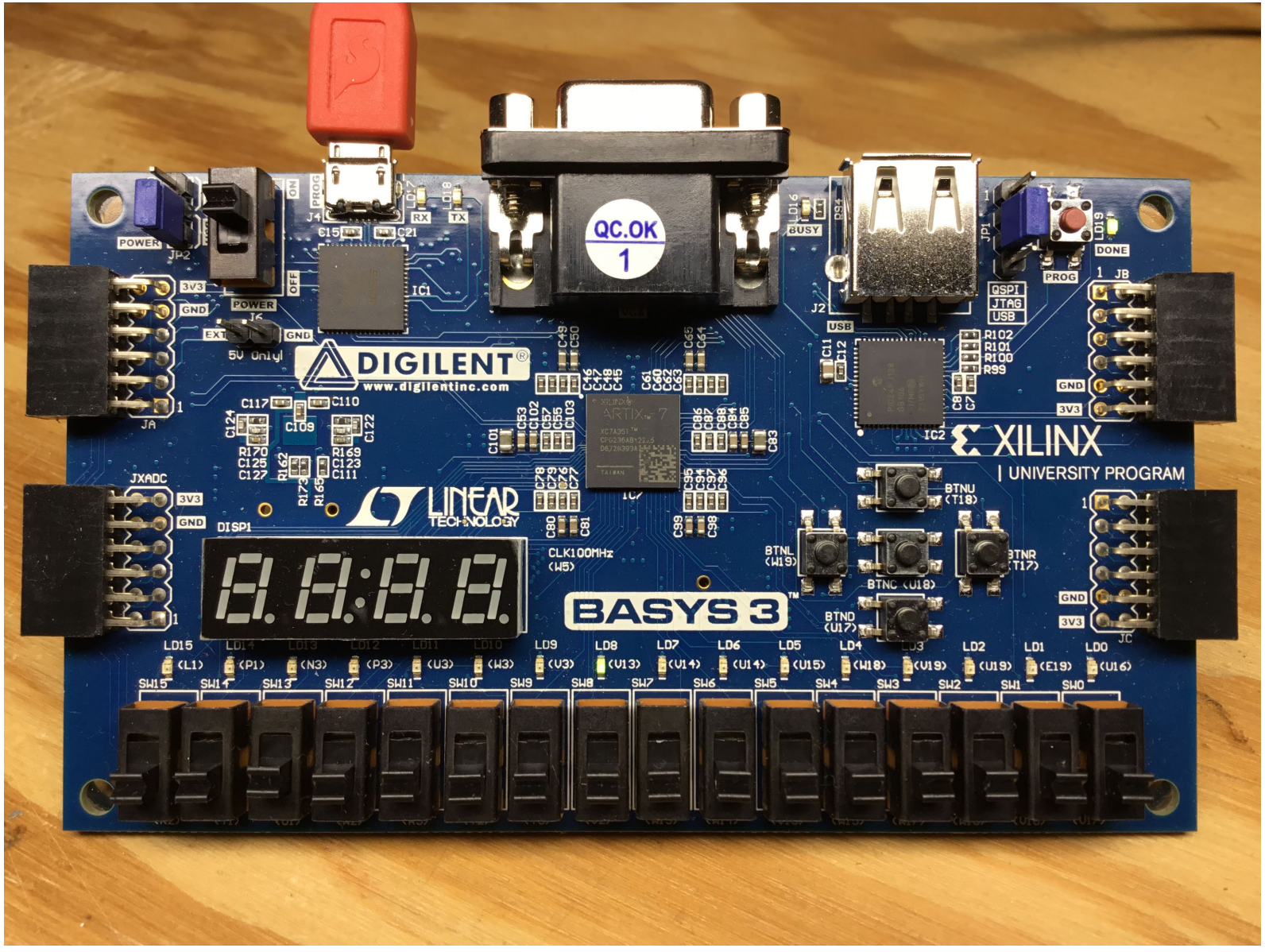
Visit ‘Lecturer Profile’ in your console for more details.

CEC Continuing Education Center

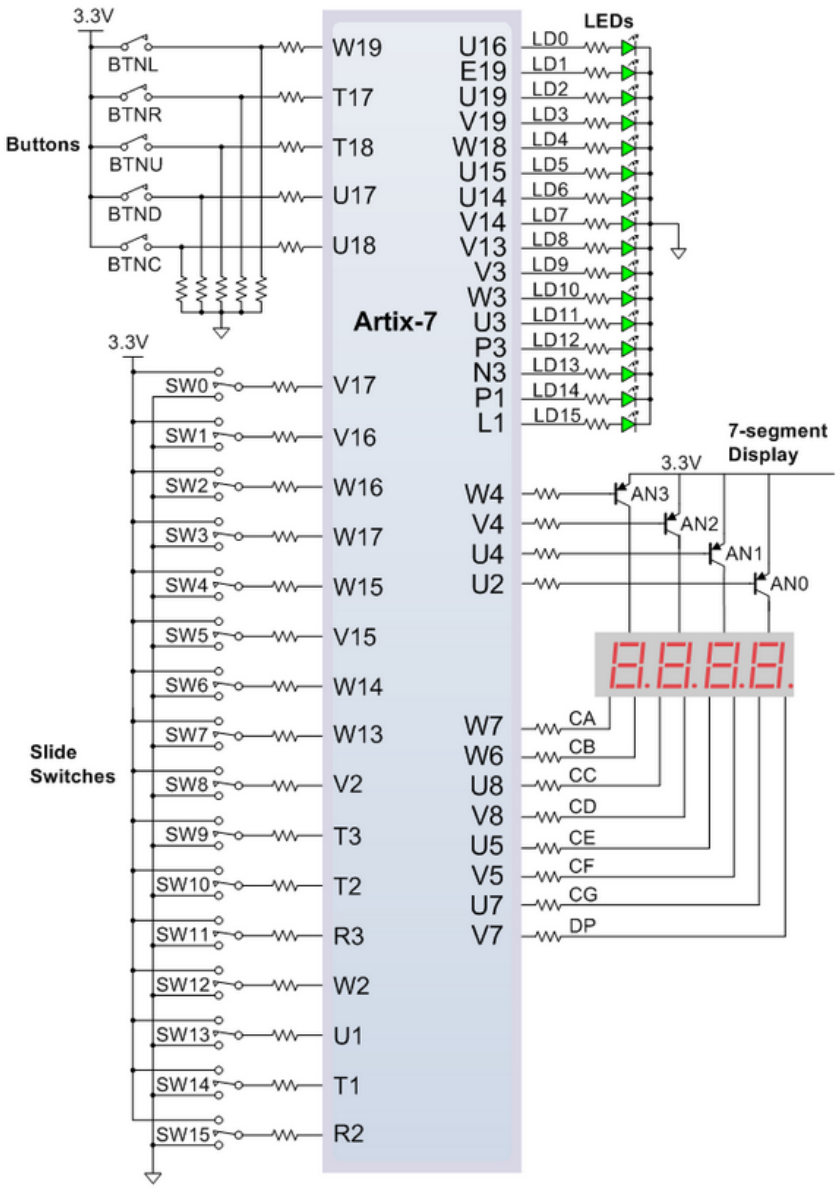
Sponsored By Digi-Key Corporation

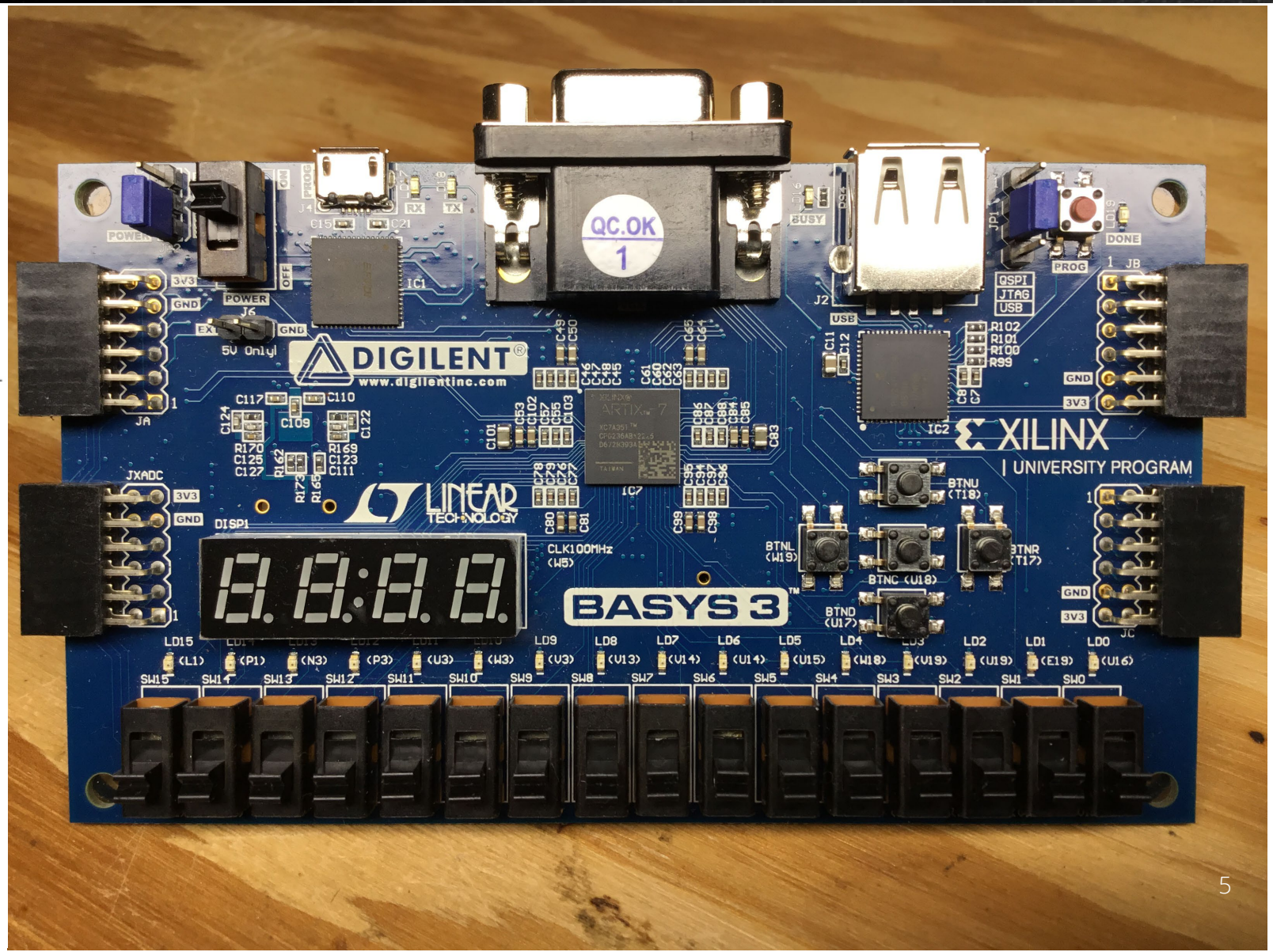
AGENDA

- The Hook Up
- Vivado Projects
  - *gates*
  - *sw_to_led*
  - *seven_seg_A*
  - *debounce_A*

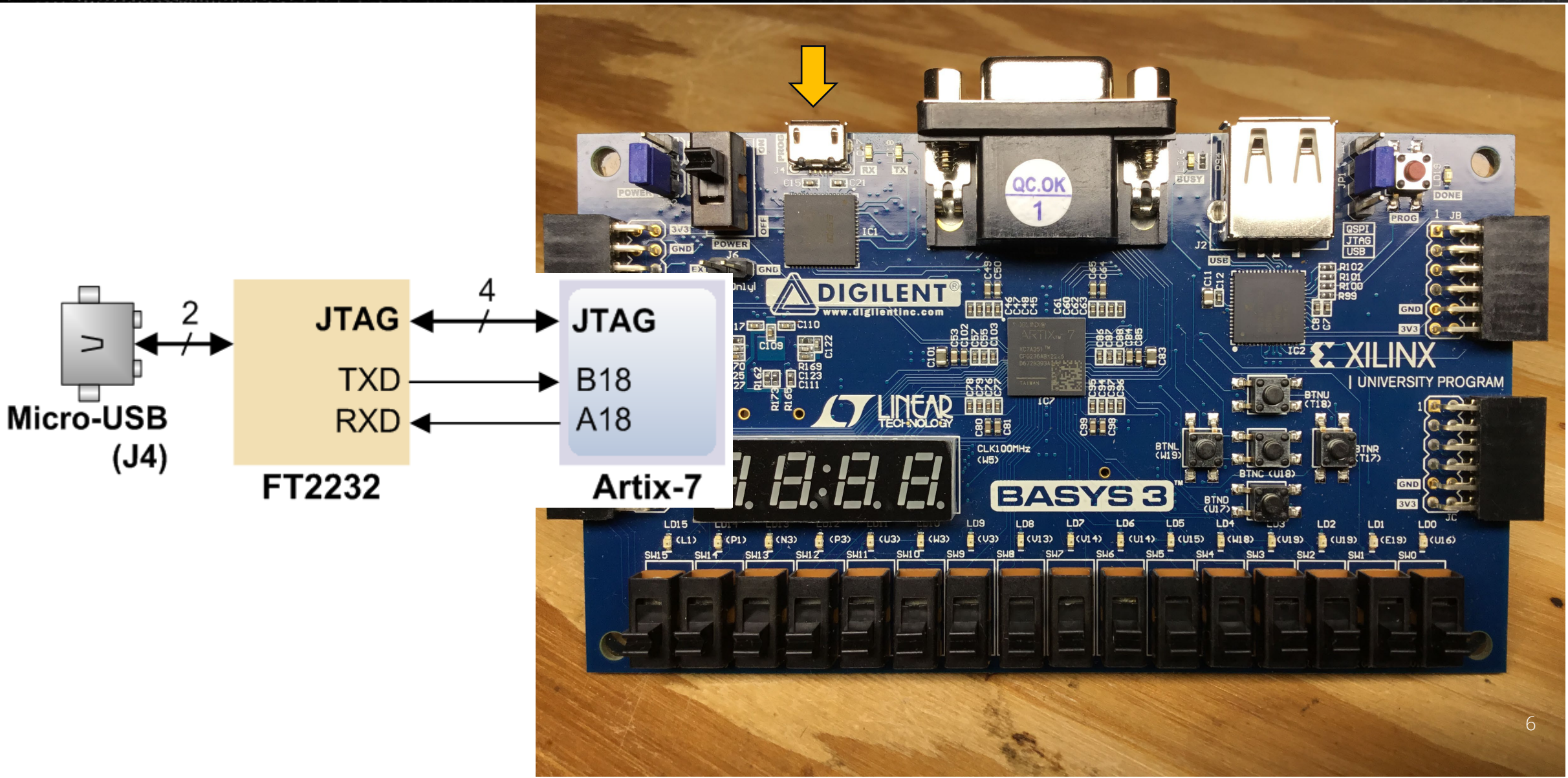


Buttons, Switches and LEDs





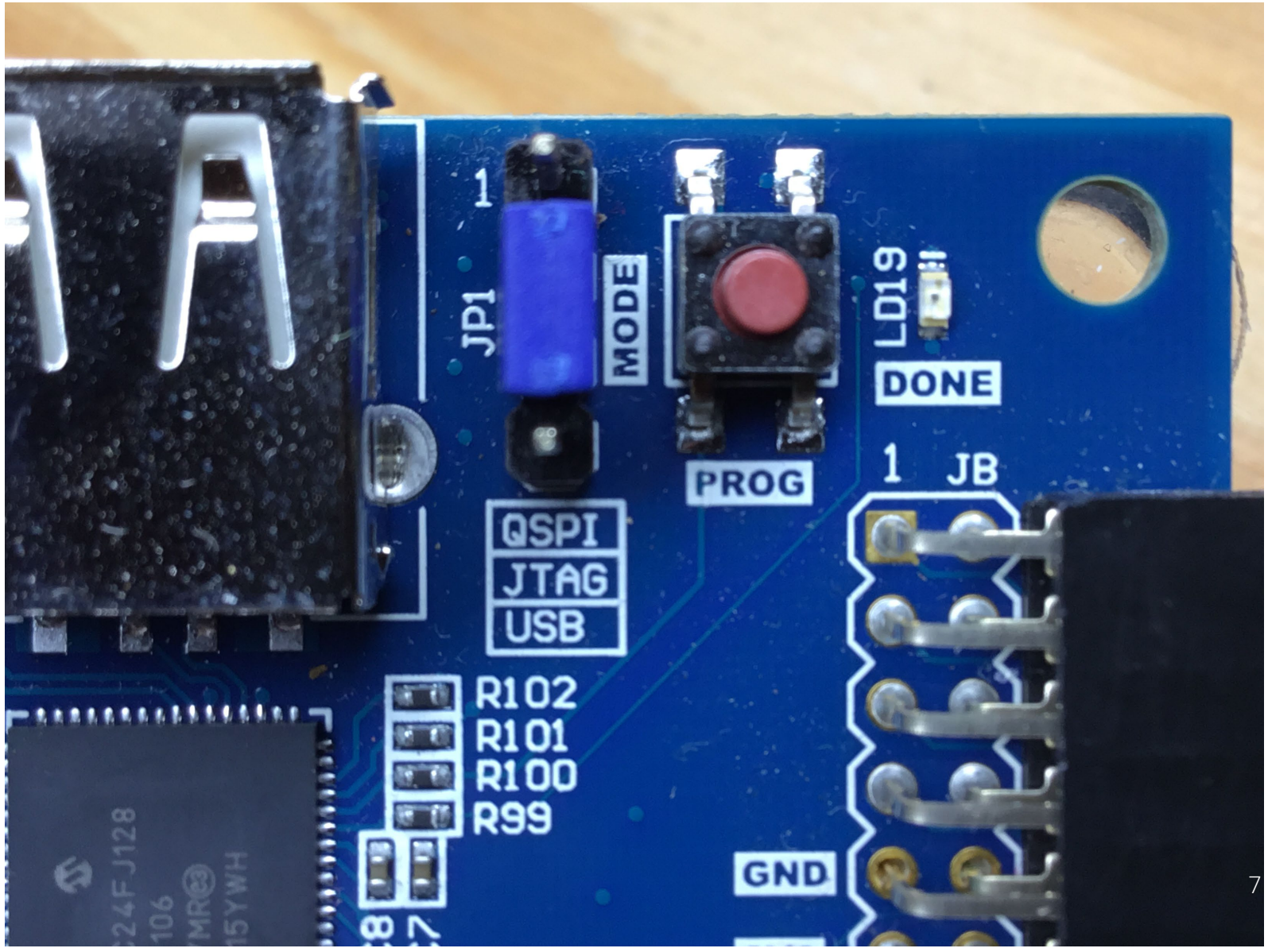
JTAG (Joint Test Action Group)



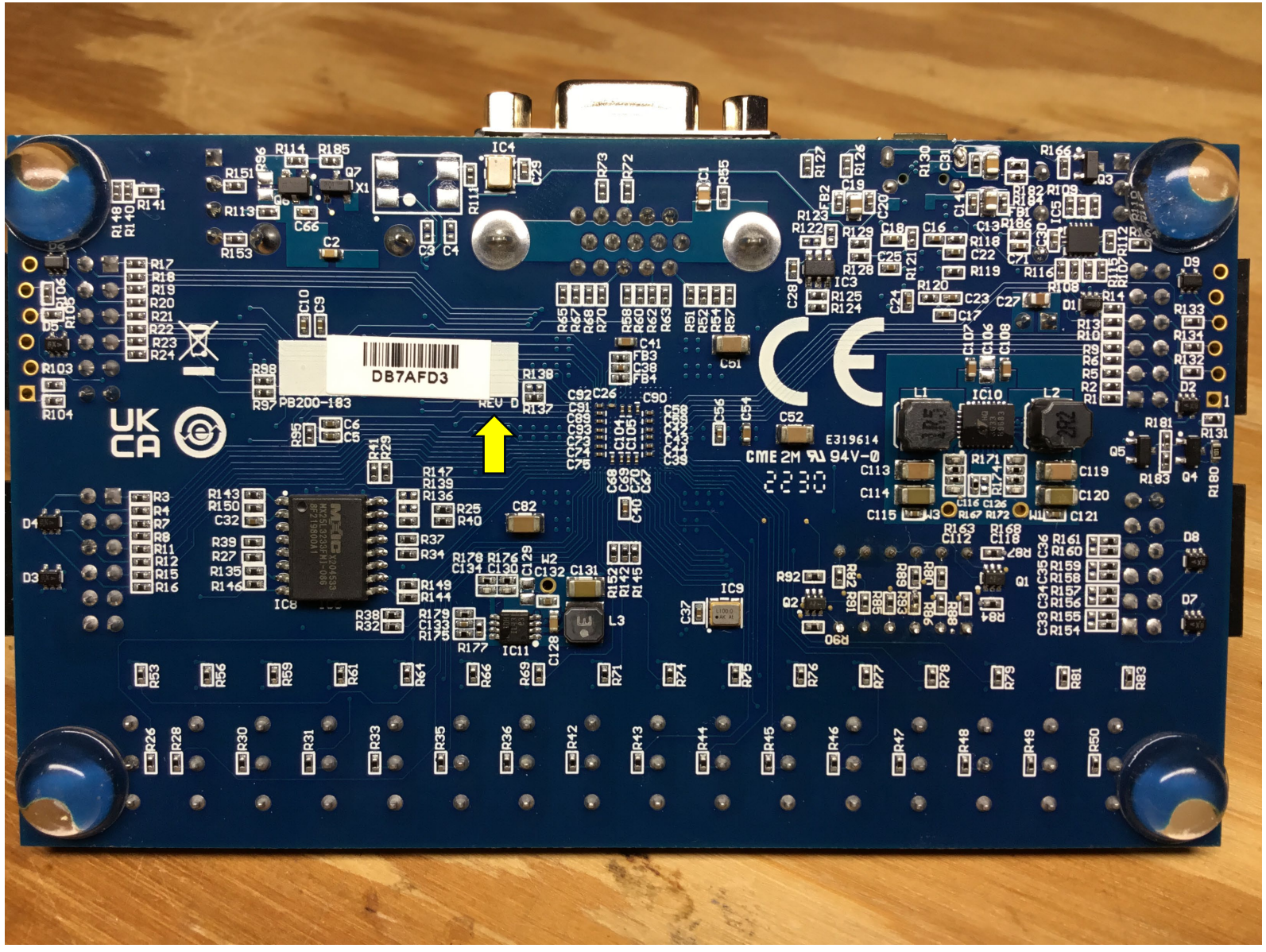
Programming Mode Jumper

JP1	
	SPI Flash
	JTAG
	USB

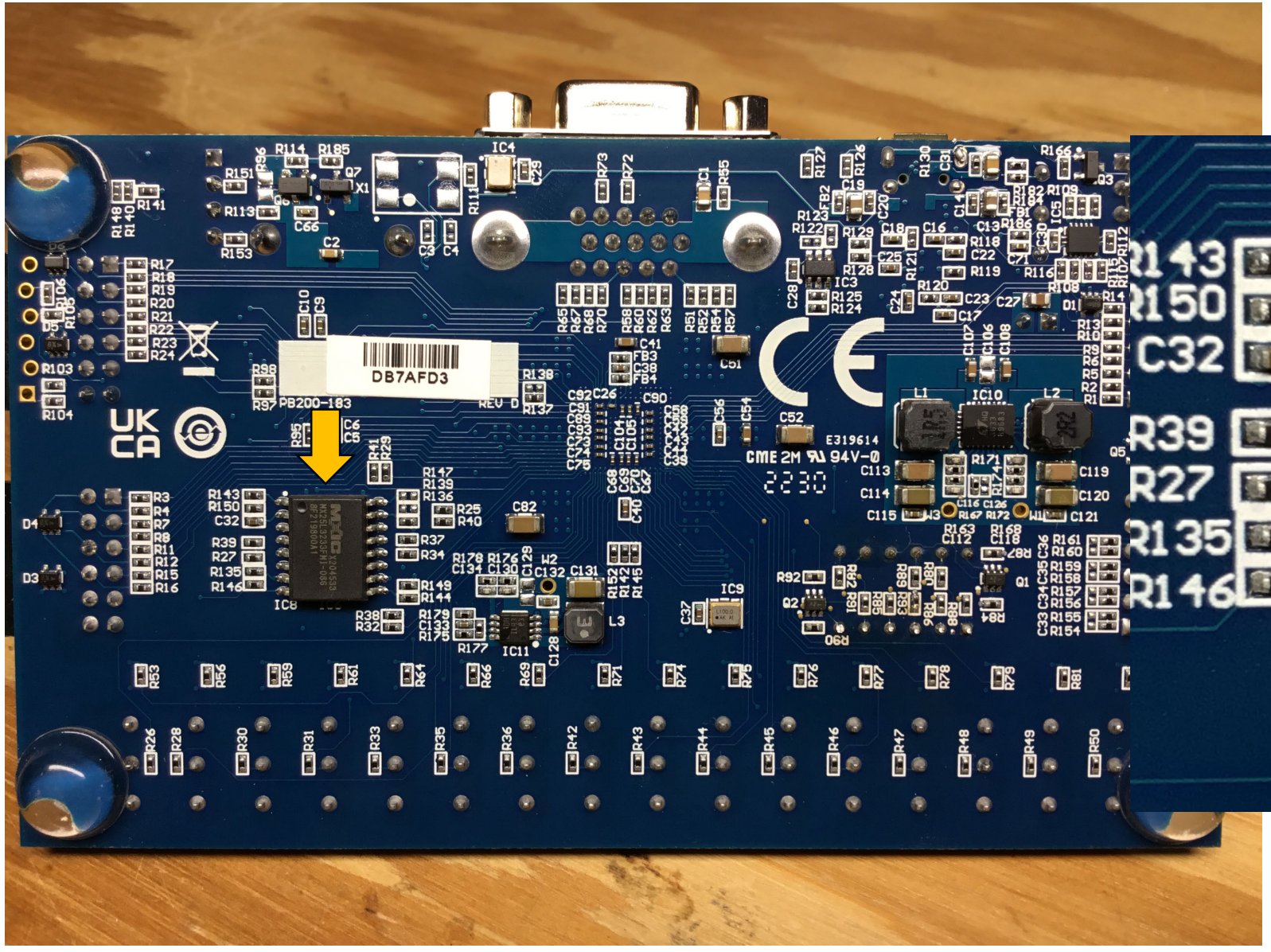
Programming Mode

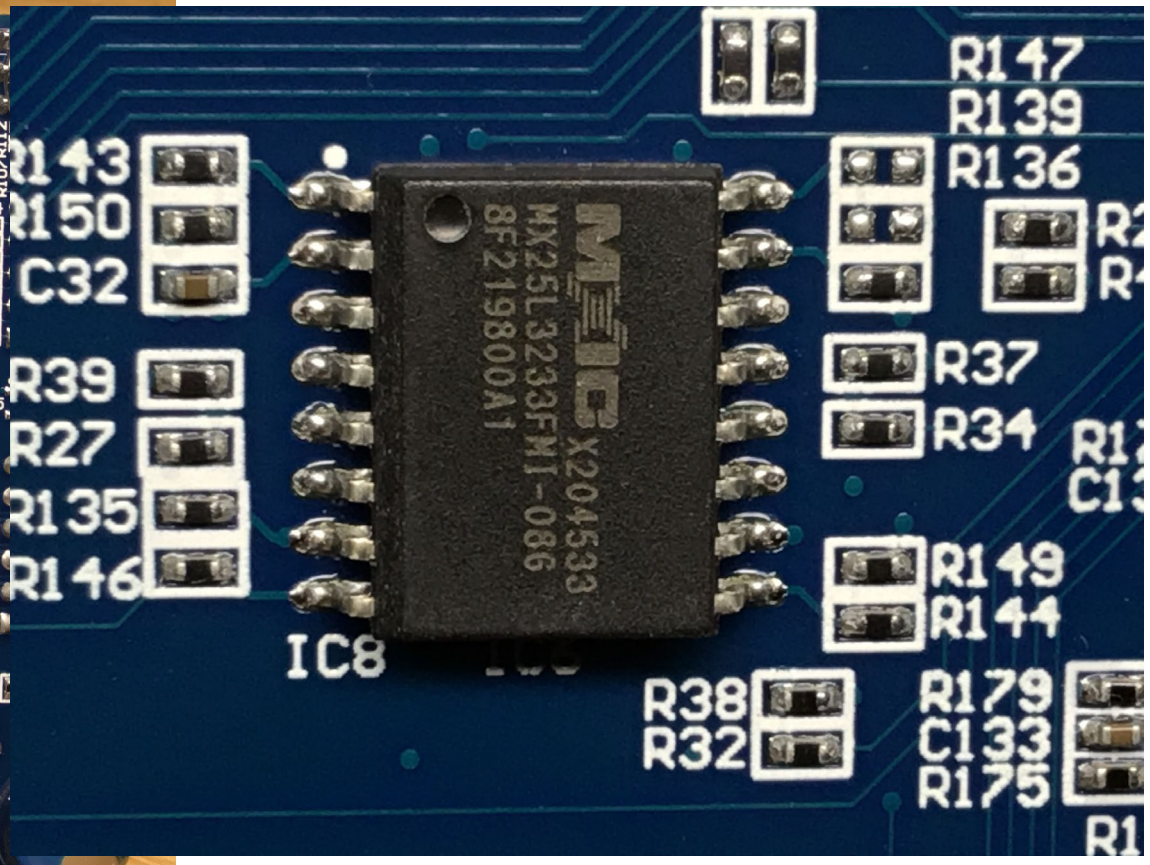


The Other Side

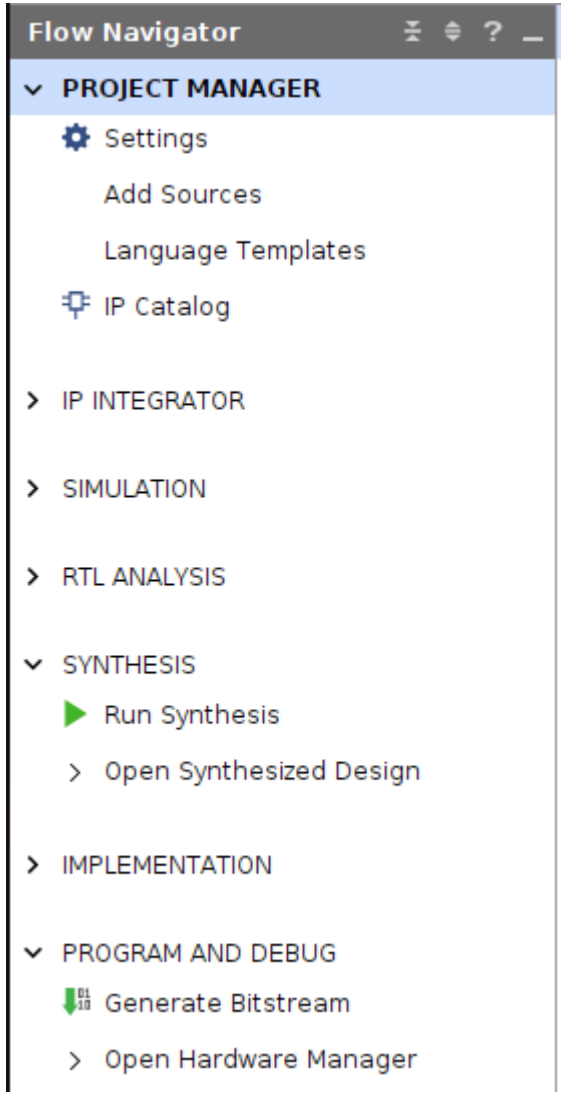


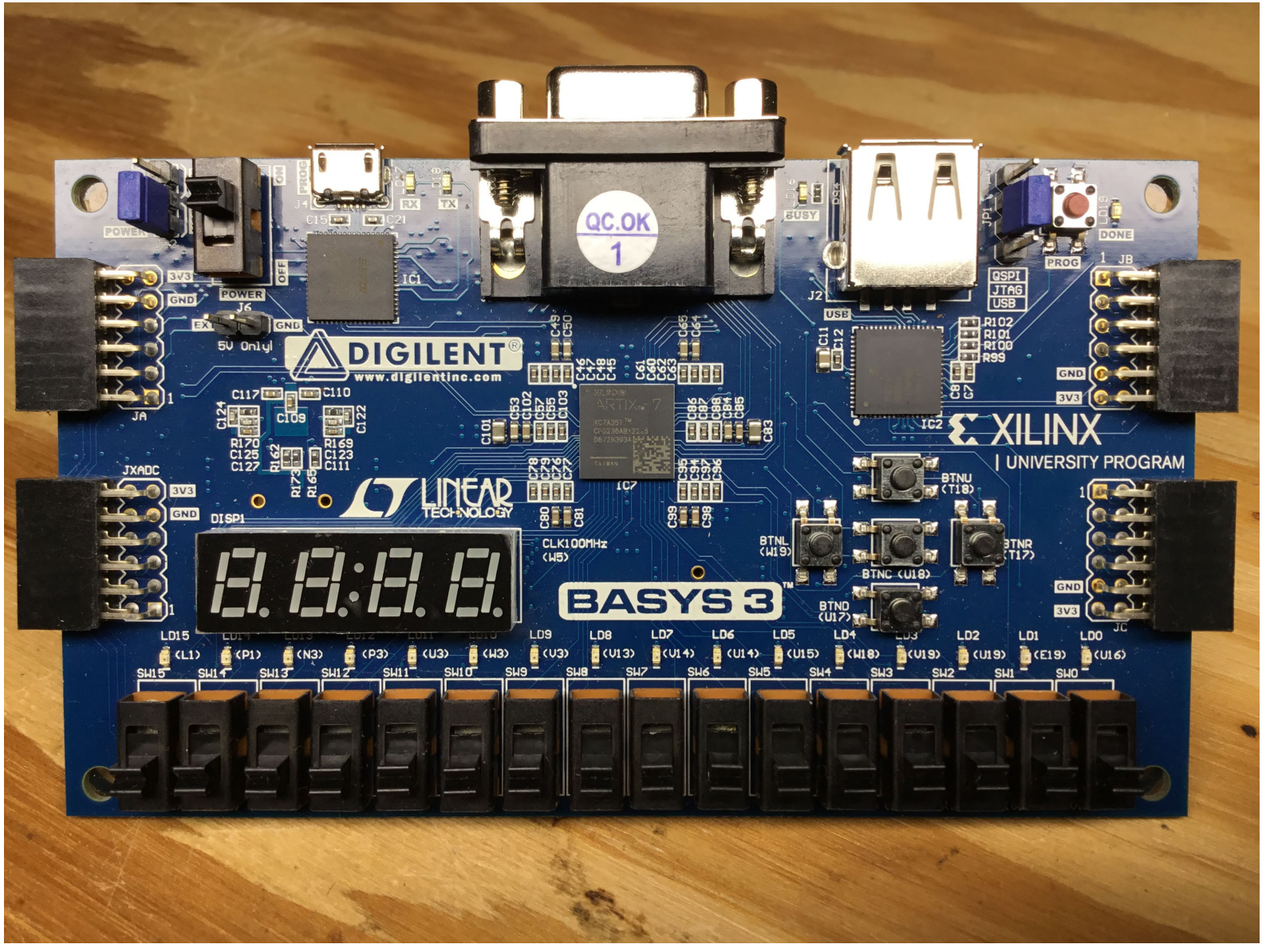
32Mbit Non-Volatile Serial Flash





Flow


Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
SIMULATION
RTL ANALYSIS
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager


DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
QC.OK
1

Gate Modules

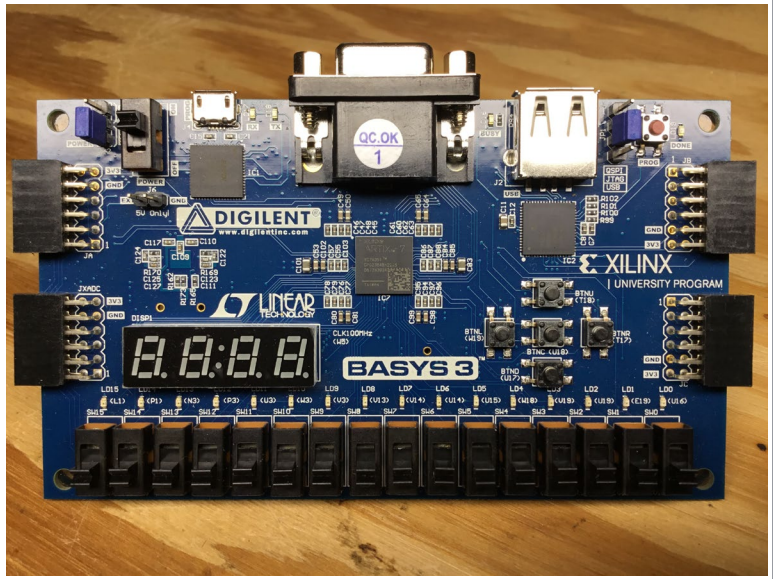
Top Module

```
22  module gates_top
23  (
24     input [15:0] sw,
25     output [15:0] led,
26     output [3:0] an
27  );
```

GATE Instantiations

```
30  andGate gateAND
31  (
32       .a(sw[0]),
33       .b(sw[1]),
34       .y(led[0])
35   );
36  nandGate gateNAND
37  (
38       .a(sw[4]),
39       .b(sw[5]),
40       .y(led[2])
41   );
42  orGate gateOR
43  (
44       .a(sw[2]),
45       .b(sw[3]),
46       .y(led[1])
47   );
```

```
48  norGate gateNOR
49  (
50       .a(sw[6]),
51       .b(sw[7]),
52       .y(led[3])
53   );
54  exorGate gateEXOR
55  (
56       .a(sw[8]),
57       .b(sw[9]),
58       .y(led[4])
59   );
60  exnorGate gateEXNOR
61  (
62       .a(sw[10]),
63       .b(sw[11]),
64       .y(led[5])
65   );
```



GATE Modules

```
22  module andGate(
23      input a,
24      input b,
25      output y
26      );
27      //assign y = a & b;
28      and(y,a,b);
29  endmodule
30
31  module nandGate(
32      input a,
33      input b,
34      output y
35      );
36      assign y = ~(a & b);
37      //nand(y,a,b);
38  endmodule
39
40  module orGate(
41      input a,
42      input b,
43      output y
44      );
45      //assign y = a | b;
46      or(y,a,b);
47  endmodule
```

```
49  module norGate(
50      input a,
51      input b,
52      output y
53      );
54      assign y = ~(a | b);
55      //nor(y,a,b);
56  endmodule
57
58  module exorGate(
59      input a,
60      input b,
61      output y
62      );
63      //assign y = a ^ b;
64      xor(y,a,b);
65  endmodule
66
67  module exnorGate(
68      input a,
69      input b,
70      output y
71      );
72      assign y = ~(a ^ b);
73      //xnor(y,a,b);
74  endmodule
```

AND Gate - Basys-3-Master.xdc

```
22  module gates_top
23  (
24     input [15:0] sw,
25     output [15:0] led,
26     output [3:0] an
27  );
28
29  //anode segments are active low
30  //turn off 7-segment anodes
31  assign an = 4'b1111;
32
33  andGate gateAND
34  (
35      .a(sw[0]),
36      .b(sw[1]),
37      .y(led[0])
38   );
```

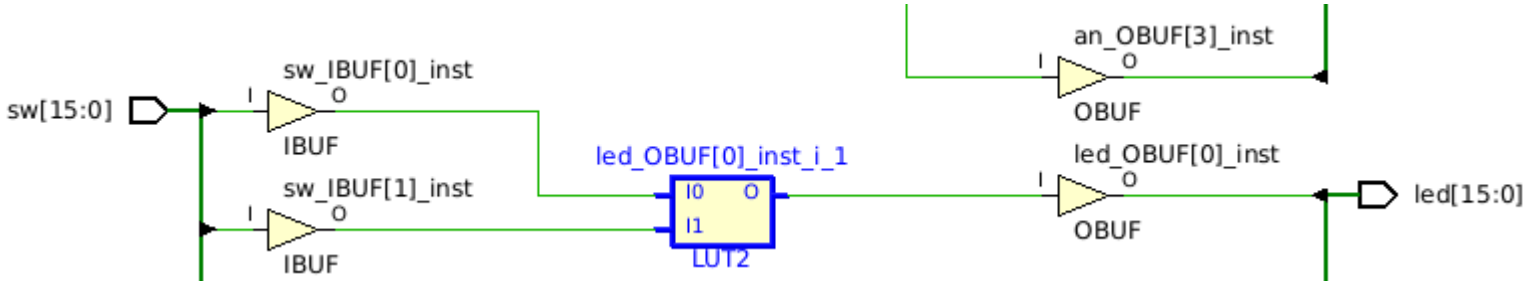
```
22  module andGate(
23      input a,
24      input b,
25      output y
26      );
27      //assign y = a & b;
28      and(y,a,b);
29  endmodule
```

```
11  ## Switches
12  set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {sw[0]}]
13  set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {sw[1]}]
14  set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports {sw[2]}]
15  set_property -dict { PACKAGE_PIN W17   IOSTANDARD LVCMOS33 } [get_ports {sw[3]}]
16  set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33 } [get_ports {sw[4]}]
17  set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports {sw[5]}]
18  set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33 } [get_ports {sw[6]}]
19  set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports {sw[7]}]
20  set_property -dict { PACKAGE_PIN V2    IOSTANDARD LVCMOS33 } [get_ports {sw[8]}]
21  set_property -dict { PACKAGE_PIN T3    IOSTANDARD LVCMOS33 } [get_ports {sw[9]}]
22  set_property -dict { PACKAGE_PIN T2    IOSTANDARD LVCMOS33 } [get_ports {sw[10]}]
23  set_property -dict { PACKAGE_PIN R3    IOSTANDARD LVCMOS33 } [get_ports {sw[11]}]
24  set_property -dict { PACKAGE_PIN W2    IOSTANDARD LVCMOS33 } [get_ports {sw[12]}]
25  set_property -dict { PACKAGE_PIN U1    IOSTANDARD LVCMOS33 } [get_ports {sw[13]}]
26  set_property -dict { PACKAGE_PIN T1    IOSTANDARD LVCMOS33 } [get_ports {sw[14]}]
27  set_property -dict { PACKAGE_PIN R2    IOSTANDARD LVCMOS33 } [get_ports {sw[15]}]
28
29
30  ## LEDs
31  set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {led[0]}]
32  set_property -dict { PACKAGE_PIN E19   IOSTANDARD LVCMOS33 } [get_ports {led[1]}]
33  set_property -dict { PACKAGE_PIN U19   IOSTANDARD LVCMOS33 } [get_ports {led[2]}]
34  set_property -dict { PACKAGE_PIN V19   IOSTANDARD LVCMOS33 } [get_ports {led[3]}]
35  set_property -dict { PACKAGE_PIN W18   IOSTANDARD LVCMOS33 } [get_ports {led[4]}]
36  set_property -dict { PACKAGE_PIN U15   IOSTANDARD LVCMOS33 } [get_ports {led[5]}]
37  set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports {led[6]}]
38  set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports {led[7]}]
39  set_property -dict { PACKAGE_PIN V13   IOSTANDARD LVCMOS33 } [get_ports {led[8]}]
40  set_property -dict { PACKAGE_PIN V3    IOSTANDARD LVCMOS33 } [get_ports {led[9]}]
41  set_property -dict { PACKAGE_PIN W3    IOSTANDARD LVCMOS33 } [get_ports {led[10]}]
42  set_property -dict { PACKAGE_PIN U3    IOSTANDARD LVCMOS33 } [get_ports {led[11]}]
43  set_property -dict { PACKAGE_PIN P3    IOSTANDARD LVCMOS33 } [get_ports {led[12]}]
44  set_property -dict { PACKAGE_PIN N3    IOSTANDARD LVCMOS33 } [get_ports {led[13]}]
45  set_property -dict { PACKAGE_PIN P1    IOSTANDARD LVCMOS33 } [get_ports {led[14]}]
46  set_property -dict { PACKAGE_PIN L1    IOSTANDARD LVCMOS33 } [get_ports {led[15]}]
```

AND Gate – Generated Schematic

```
22  module gates_top
23  (
24     input [15:0] sw,
25     output [15:0] led,
26     output [3:0] an
27  );
28
29  //anode segments are active low
30  //turn off 7-segment anodes
31  assign an = 4'b1111;
32
33  andGate gateAND
34  (
35      .a(sw[0]),
36      .b(sw[1]),
37      .y(led[0])
38   );
```

```
22  module andGate(
23      input a,
24      input b,
25      output y
26      );
27      //assign y = a & b;
28      and(y,a,b);
29  endmodule
```



Cell Properties

led_OBUF[0]_inst_i_1

I1	I0	O=I0 & I1
0	0	0
0	1	0
1	0	0
1	1	1

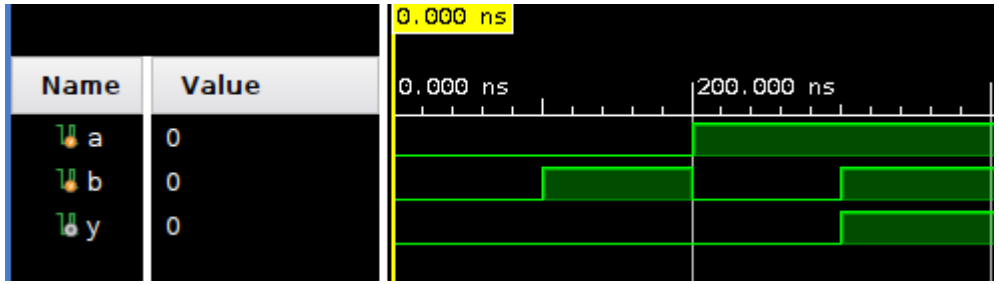
Edit LUT Equation...

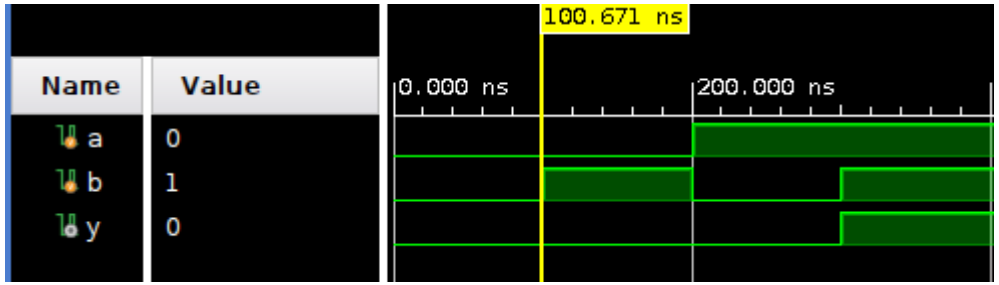
Power Nets Cell Pins **Truth Table**

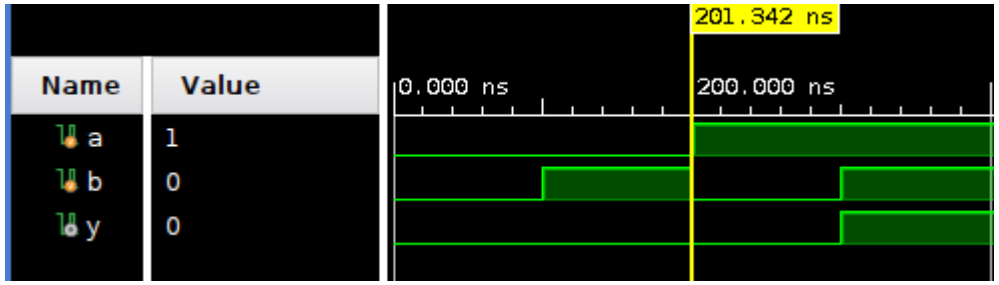
AND Gate – Simulation View

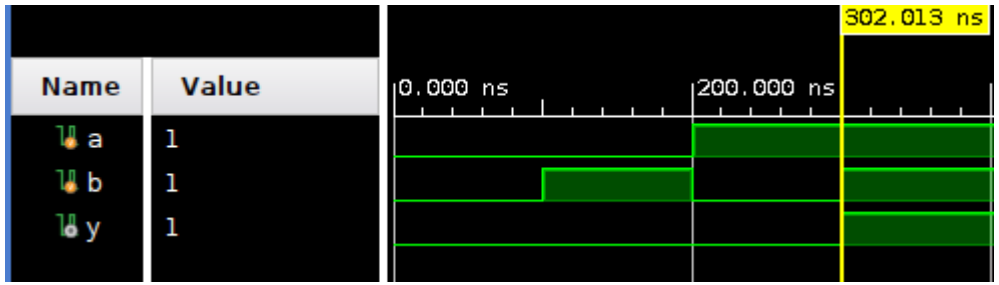
```
22  module gates_top
23  (
24     input [15:0] sw,
25     output [15:0] led,
26     output [3:0] an
27  );
28
29  //anode segments are active low
30  //turn off 7-segment anodes
31  assign an = 4'b1111;
32
33  andGate gateAND
34  (
35      .a(sw[0]),
36      .b(sw[1]),
37      .y(led[0])
38   );
```

```
22  module andGate(
23      input a,
24      input b,
25      output y
26      );
27      //assign y = a & b;
28      and(y,a,b);
29  endmodule
```





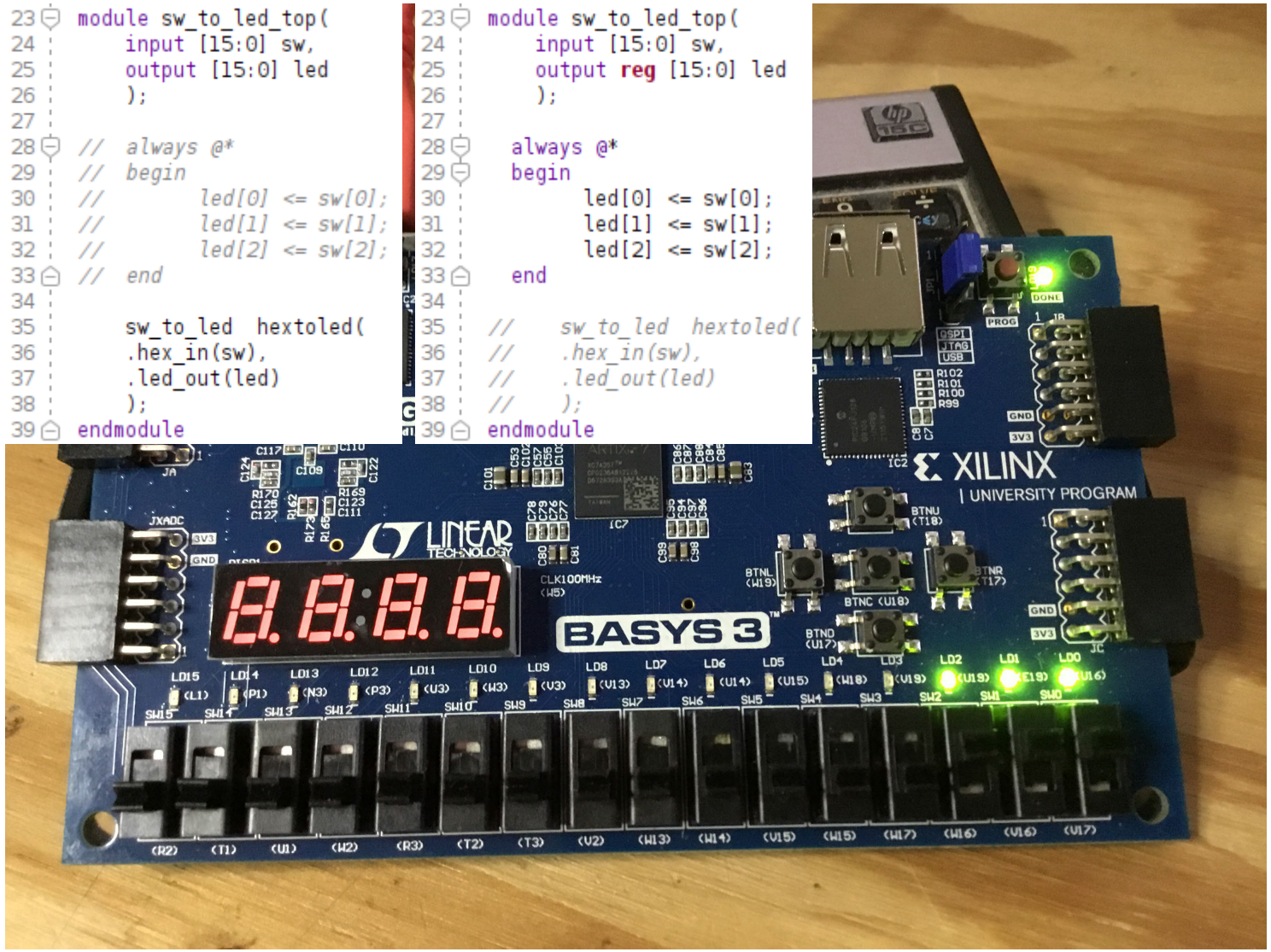




Switch x ON = LED x ON

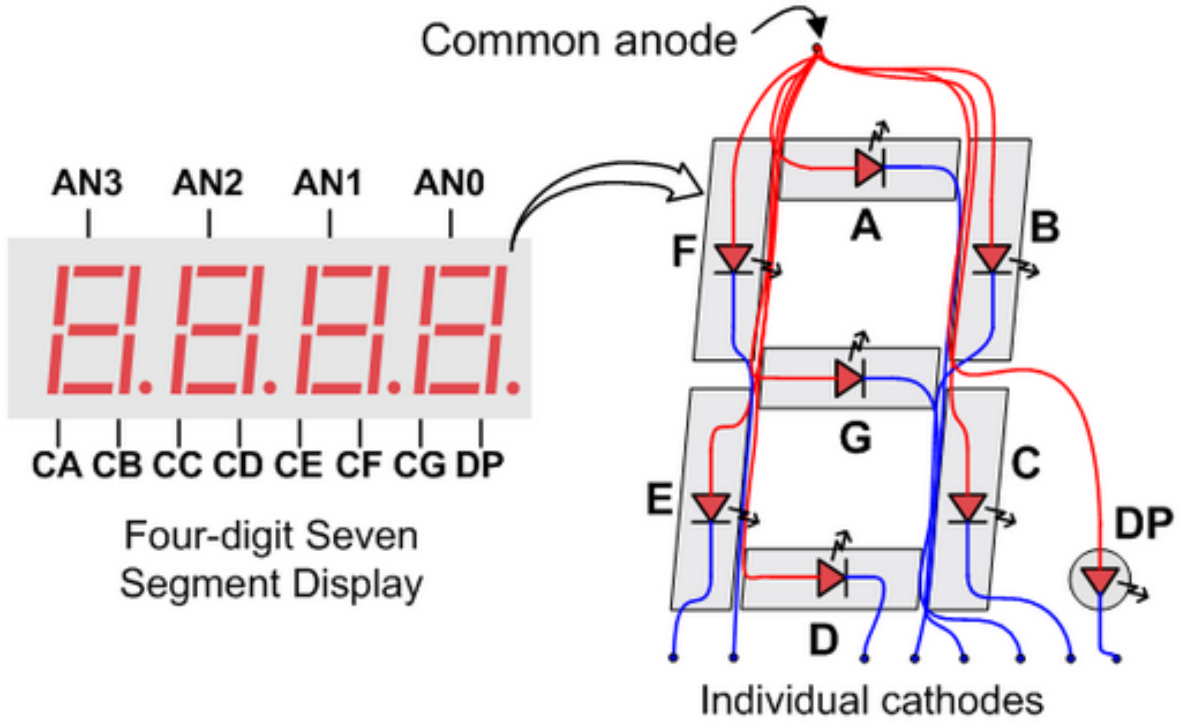
```
23  module sw_to_led_top(
24      input [15:0] sw,
25      output [15:0] led
26      );
27
28  //  always @*
29  //  begin
30  //          led[0] <= sw[0];
31  //          led[1] <= sw[1];
32  //          led[2] <= sw[2];
33  //  end
34
35      sw_to_led  hextoled(
36      .hex_in(sw),
37      .led_out(led)
38      );
39  endmodule
```

```
23  module sw_to_led_top(
24      input [15:0] sw,
25      output reg [15:0] led
26      );
27
28    always @*
29    begin
30          led[0] <= sw[0];
31          led[1] <= sw[1];
32          led[2] <= sw[2];
33    end
34
35  //    sw_to_led  hextoled(
36  //    .hex_in(sw),
37  //    .led_out(led)
38  //    );
39  endmodule
```




```
23  module sw_to_led(
24      input [15:0] hex_in,
25      output reg [15:0] led_out
26      );
27
28      // activate on any input change
29      always @*
30      begin
31          case(hex_in)
32              16'h0:    led_out <= 16'b0000000000000000;     // LED OFF
33              16'h1:    led_out <= 16'b0000000000000001;     // LED 0
34              16'h2:    led_out <= 16'b0000000000000010;     // LED 1
35              16'h4:    led_out <= 16'b0000000000000100;     // LED 2
36              16'h8:    led_out <= 16'b0000000000001000;     // LED 3
37              16'h10:   led_out <= 16'b0000000000010000;     // LED 4
38              16'h20:   led_out <= 16'b0000000000100000;     // LED 5
39              16'h40:   led_out <= 16'b0000000001000000;     // LED 6
40              16'h80:   led_out <= 16'b0000000010000000;     // LED 7
41              16'h100:  led_out <= 16'b0000000100000000;     // LED 8
42              16'h200:  led_out <= 16'b0000001000000000;     // LED 9
43              16'h400:  led_out <= 16'b0000010000000000;     // LED 10
44              16'h800:  led_out <= 16'b0000100000000000;     // LED 11
45              16'h1000: led_out <= 16'b0001000000000000;     // LED 12
46              16'h2000: led_out <= 16'b0010000000000000;     // LED 13
47              16'h4000: led_out <= 16'b0100000000000000;     // LED 14
48              16'h8000: led_out <= 16'b1000000000000000;     // LED 15
49              default:  led_out <= 16'b0000000000000000;     // LED OFF
50          endcase
51      end
52  endmodule
```

Seven-Segment Display Pin Assignments




```
50  set_property -dict { PACKAGE_PIN W7   IOSTANDARD LVCMOS33 } [get_ports {seg[0]}]   A
51  set_property -dict { PACKAGE_PIN W6   IOSTANDARD LVCMOS33 } [get_ports {seg[1]}]   B
52  set_property -dict { PACKAGE_PIN U8   IOSTANDARD LVCMOS33 } [get_ports {seg[2]}]   C
53  set_property -dict { PACKAGE_PIN V8   IOSTANDARD LVCMOS33 } [get_ports {seg[3]}]   D
54  set_property -dict { PACKAGE_PIN U5   IOSTANDARD LVCMOS33 } [get_ports {seg[4]}]   E
55  set_property -dict { PACKAGE_PIN V5   IOSTANDARD LVCMOS33 } [get_ports {seg[5]}]   F
56  set_property -dict { PACKAGE_PIN U7   IOSTANDARD LVCMOS33 } [get_ports {seg[6]}]   G
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]}]
```

```
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)                    G     A
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
```

Hex-to-Seven-Segment

```
22  module sseg_top(
23       input [3:0] sw,
24      output [6:0] seg,
25      output [3:0] an
26      );
27
28      // an is active low, turn on an[0], others off
29      assign an = 4'b1110;
30
31      // connect our input to the display
32      sseg_display display(
33      .hex_in(sw),
34      .seg_out(seg)
35      );
36
37  endmodule
```

Top Module

Hex-to-Segment

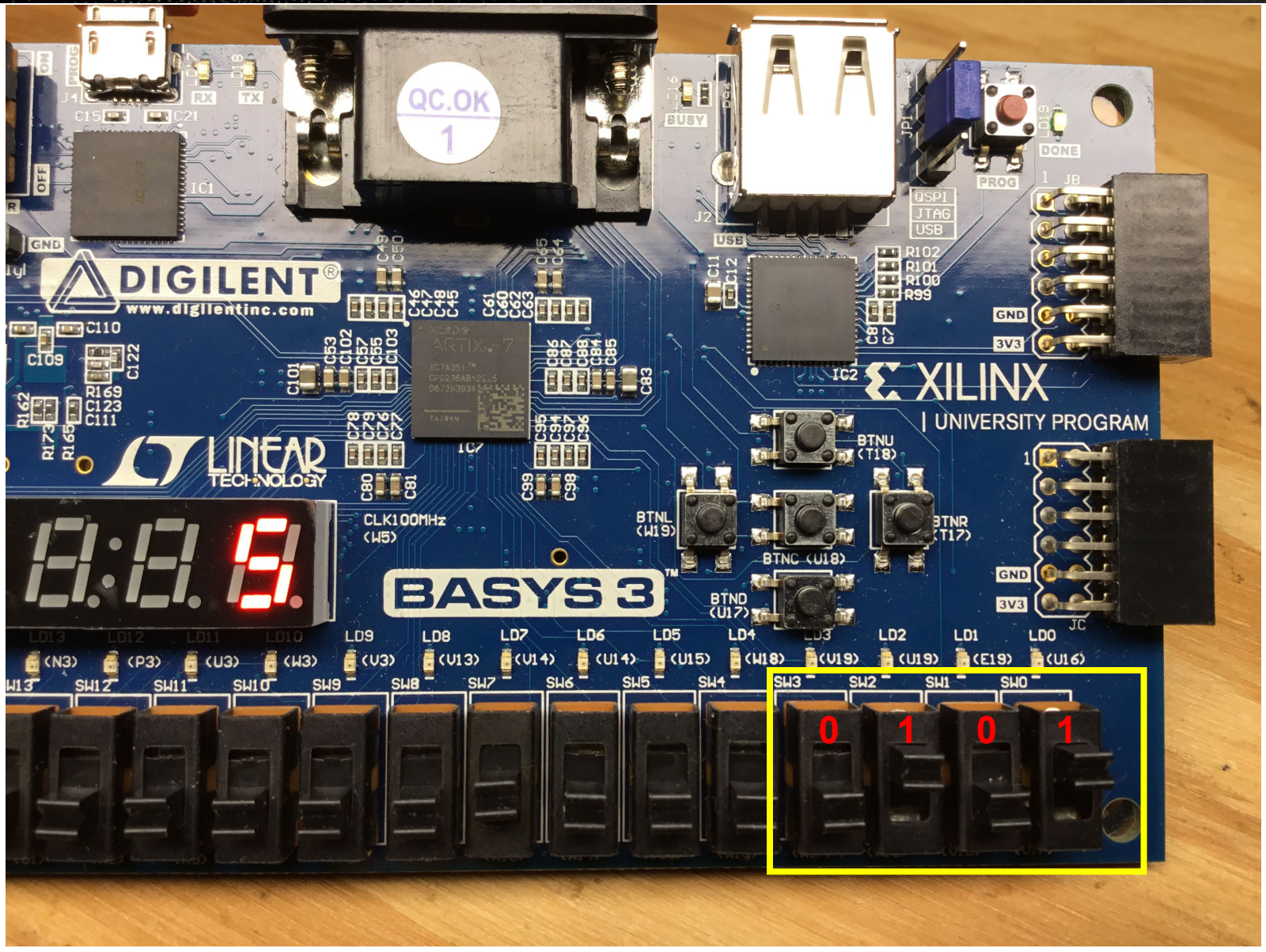
```
23  // seven-segment digit display driver
24
25  module sseg_display (
26      input [3:0] hex_in,
27      output reg [6:0] seg_out
28      );
29
30      // activate on any input change
31      always @*
32      begin
33          case(hex_in)
34              4'h0: seg_out <= 7'b1000000;    // digit 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
```

```
12  set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {sw[0]}]
13  set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {sw[1]}]
14  set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports {sw[2]}]
15  set_property -dict { PACKAGE_PIN W17   IOSTANDARD LVCMOS33 } [get_ports {sw[3]}]

49  ##7 Segment Display
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]}]
```

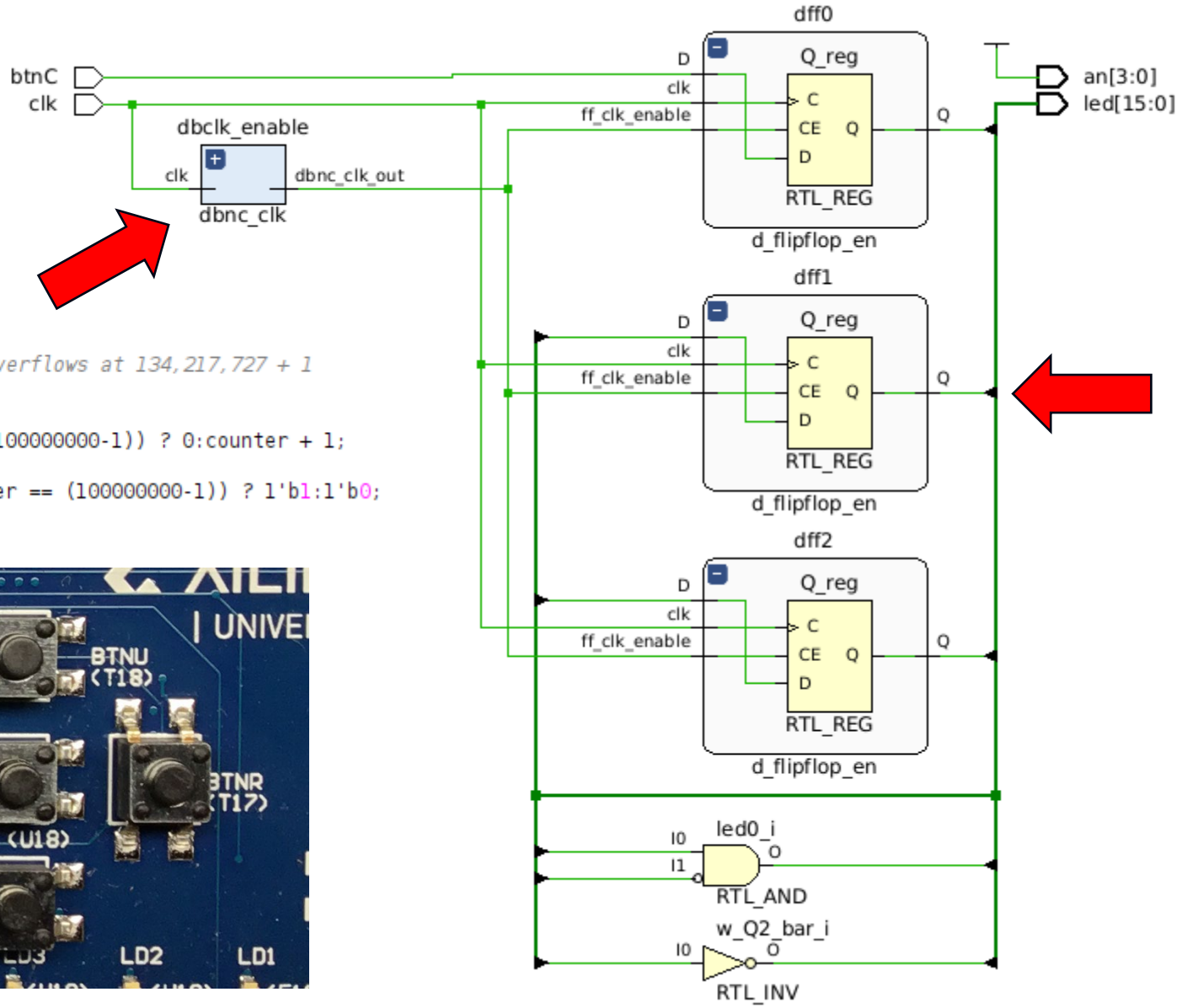
Xilinx Design Constraint File

Hex-to-Seven-Segment




```
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
52      end
53  endmodule
```

Debouncing Pushbuttons

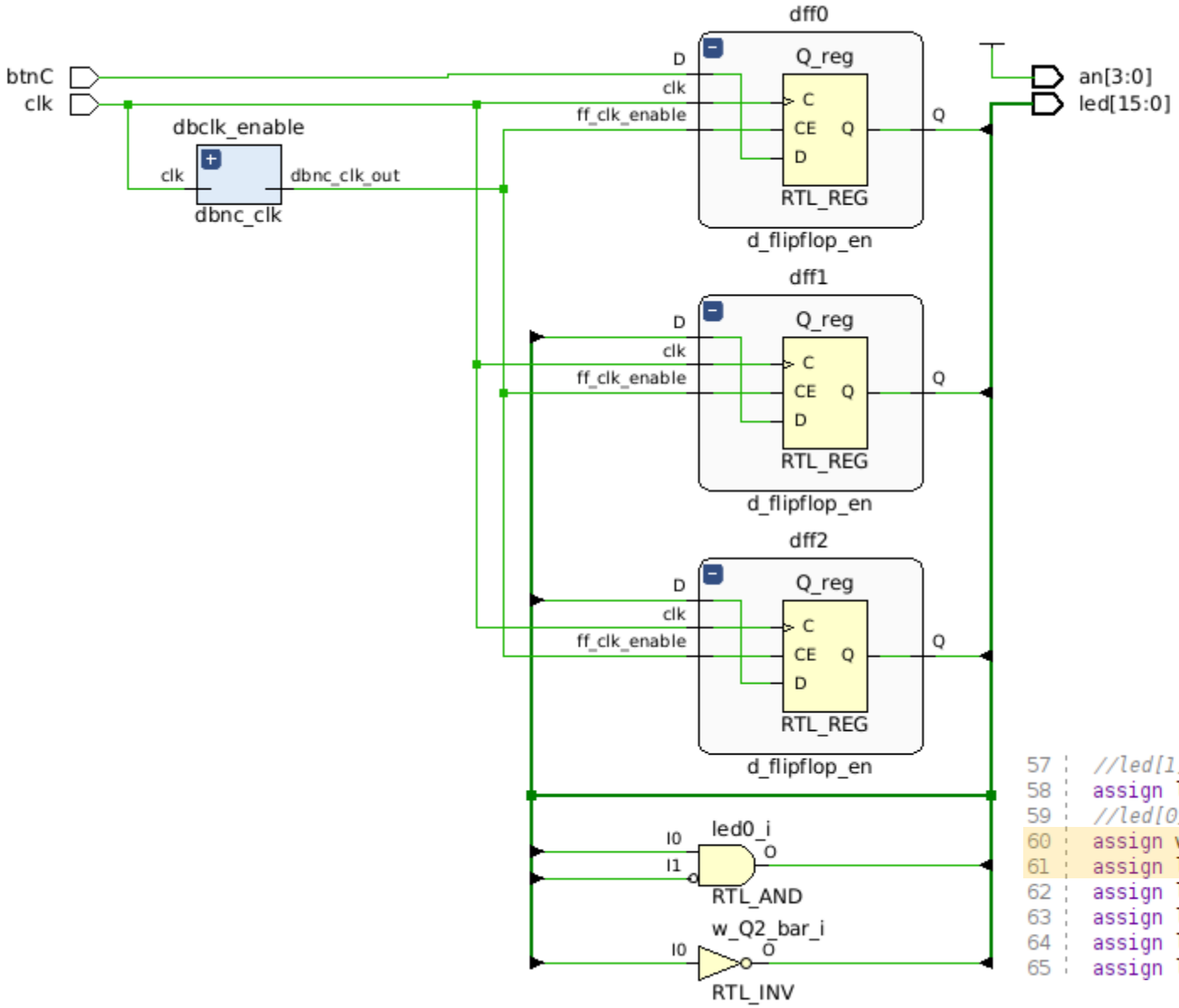



```
23  module dbnc_clk(
24      input clk,
25      output dbnc_clk_out
26  );
27      reg [26:0] counter  = 0; //overflows at 134,217,727 + 1
28      always @ (posedge clk)
29      begin
30          counter <= (counter >= (100000000-1)) ? 0:counter + 1;
31      end
32      assign dbnc_clk_out = (counter == (100000000-1)) ? 1'b1:1'b0;
33  endmodule
```

```
23  module d_flipflop_en(
24      input clk,
25      input ff_clk_enable,
26      input D,
27      output reg Q=0
28  );
29  always @ (posedge clk)
30  begin
31      if(ff_clk_enable == 1)
32          Q <= D;
33   end
34  endmodule
```

Debouncing Pushbuttons

```
23  module debounce_top(
24     input clk,
25     input btnC,
26     output [15:0]led,
27     output [3:0]an
28  );
29  //turn off 7-segment anodes
30  assign an = 4'b1111;
31
32  wire w_clk_enable;
33  wire w_Q0,w_Q1,w_Q2,w_Q2_bar;
34
35  dbnc_clk dbclk_enable(
36   .clk(clk),
37   .dbnc_clk_out(w_clk_enable)
38  );
39  d_flipflop_en dff0(
40   .clk(clk),
41   .ff_clk_enable(w_clk_enable),
42   .D(btnC),
43   .Q(w_Q0)
44   );
45  d_flipflop_en dff1(
46   .clk(clk),
47   .ff_clk_enable(w_clk_enable),
48   .D(w_Q0),
49   .Q(w_Q1)
50   );
51  d_flipflop_en dff2(
52   .clk(clk),
53   .ff_clk_enable(w_clk_enable),
54   .D(w_Q1),
55   .Q(w_Q2)
56   );
```



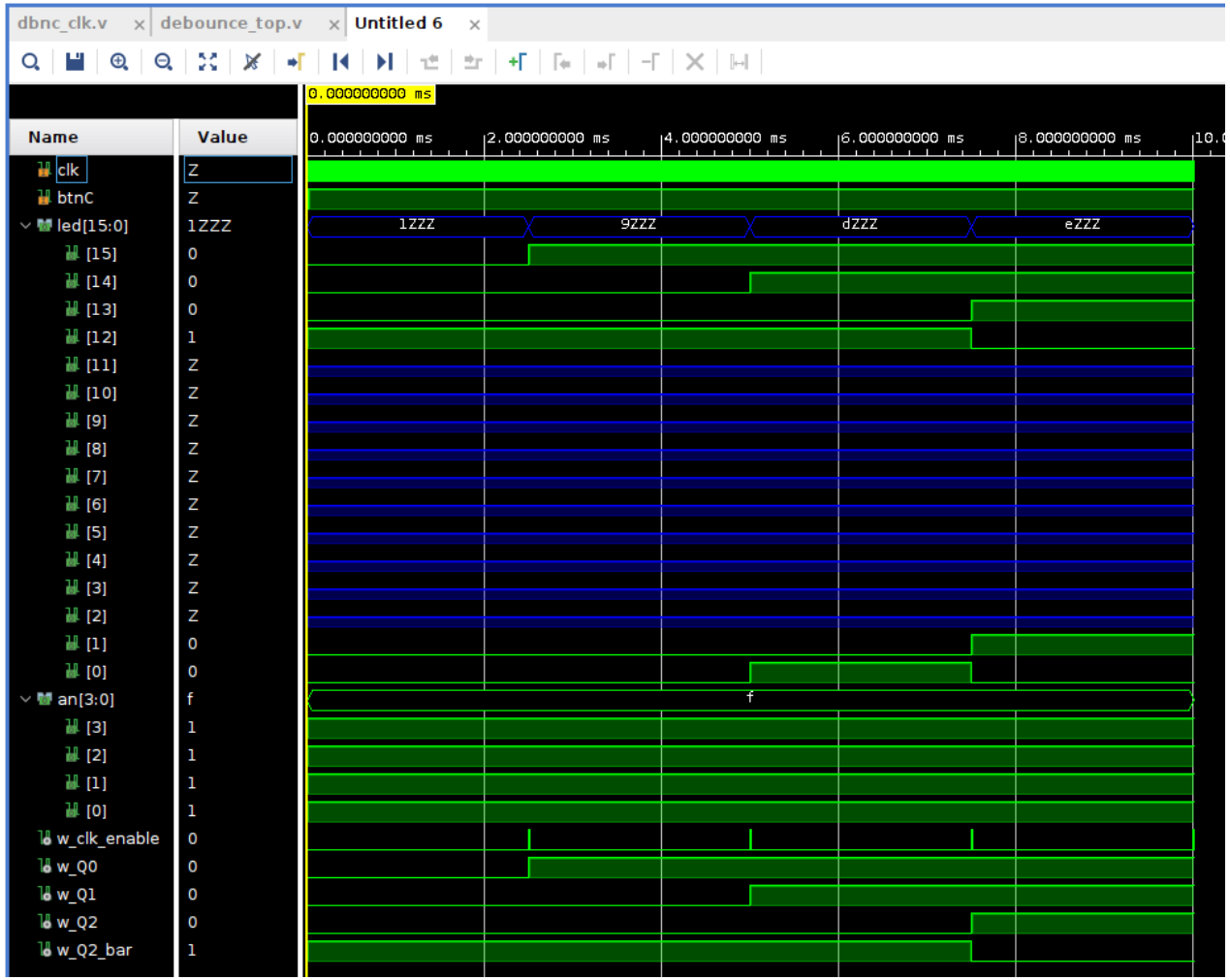

```
57  //led[1] on as long as btC is depressed
58  assign led[1] = w_Q2;
59  //led[0] on for 1 second
60  assign w_Q2_bar = ~w_Q2;
61  assign led[0] = w_Q1 & w_Q2_bar;
62  assign led[15] = w_Q0;
63  assign led[14] = w_Q1;
64  assign led[13] = w_Q2;
65  assign led[12] = w_Q2_bar;
```

Thank you for attending!!!

Please consider the resources below:

- xilinx.com
- digilent.com
- Basys 3 Reference Manual

MORE TO COME..



CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

Digi-Key ELECTRONICS

informa markets