FPGA Emulation

Hsi-Pin Ma

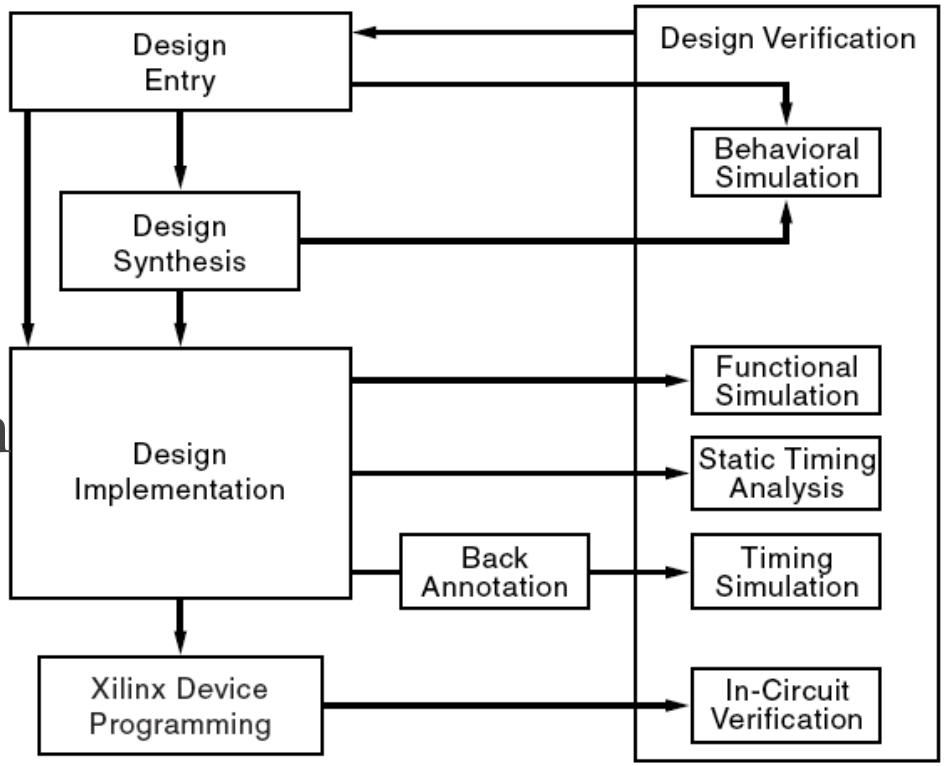
http://lms.nthu.edu.tw/course/43639
Department of Electrical Engineering
National Tsing Hua University

Notes from Lab1

- Always 'SAVE' before next step
- Select the right file for the next step
  - For simulation
  - For implementation
- Do not use specific names for files and directories
  - Leading with number,
  - Names with space,
  - Names in Chinese

Design Flow

- General design flow
  - Design construction
  - Behavioral simulation
  - Design implementation
  - Timing simulation
- HDL-based design Flow



Important Notes

- **Draw schematic first** and then construct Verilog codes.
- Verilog RTL coding philosophy is not the same as C programming
  - Every Verilog RTL construct has its own logic mapping (for synthesis)
  - You should have the logics (draw schematic) first and then the RTL codes
  - You have to write **synthesizable** RTL codes

Digilent Basys 3 Demo Board

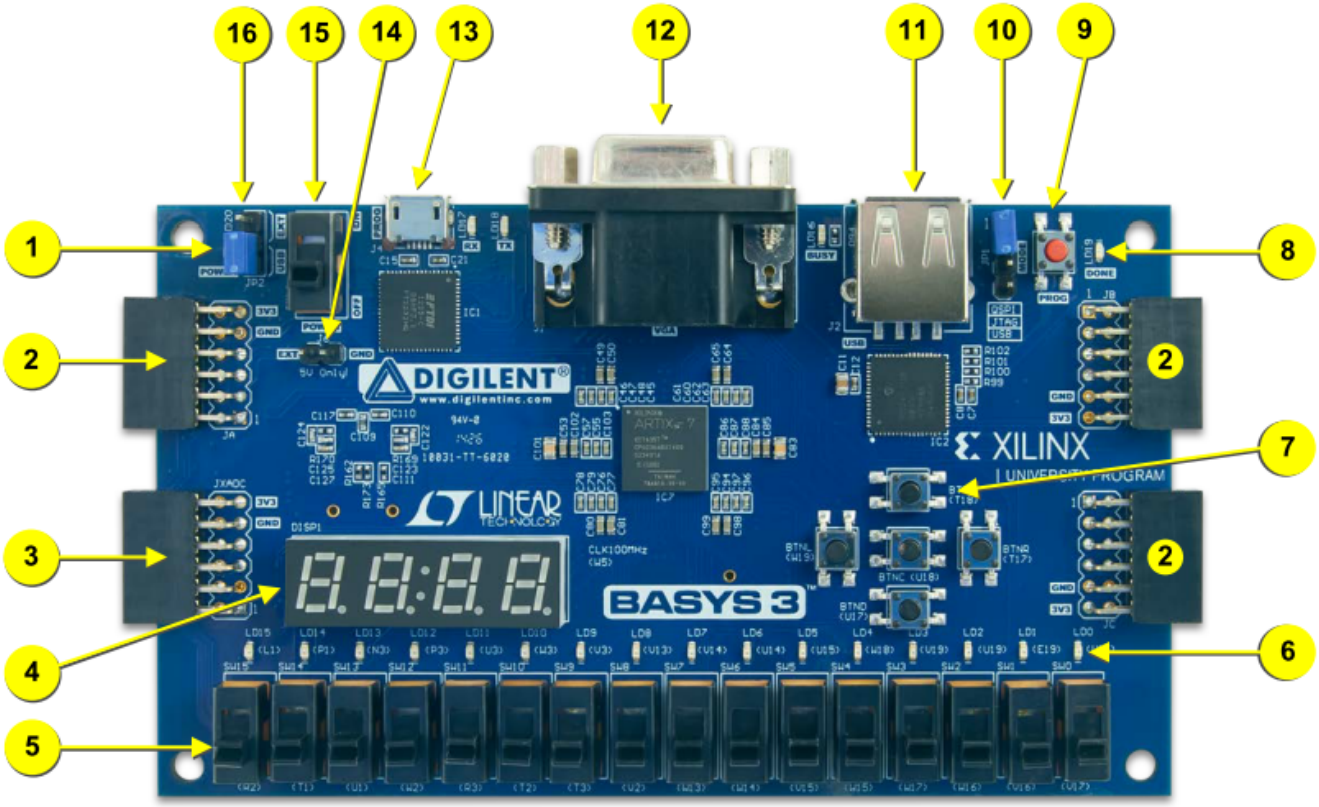


Figure 1. Basys3 FPGA board with callouts.

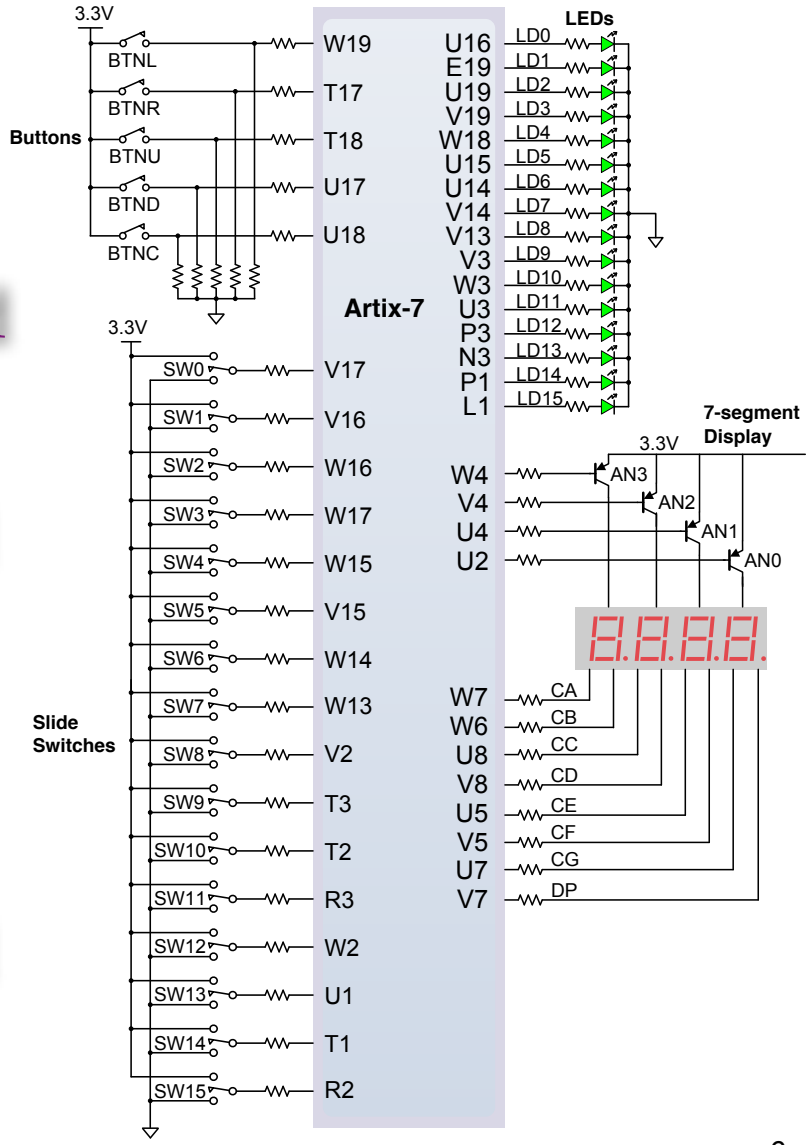
Callout	Component Description
1	Power good LED
2	Pmod connector(s)
3	Analog signal Pmod connector (XADC)
4	Four digit 7-segment display
5	Slide switches (16)
6	LEDs (16)
7	Pushbuttons (5)
8	FPGA programming done LED
9	FPGA configuration reset button
10	Programming mode jumper
11	USB host connector
12	VGA connector
13	Shared UART/ JTAG USB port
14	External power connector
15	Power Switch
16	Power Select Jumper

https://reference.digilentinc.com/reference/programmable-logic/basys-3/start?redirect=1

Check Basys 3 board reference manual for details

Input/Output Connections

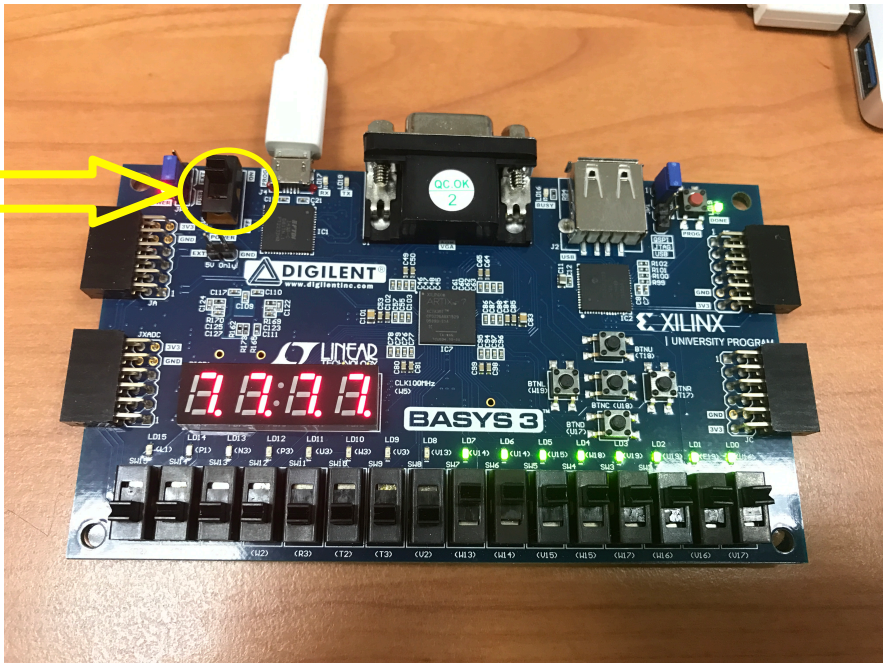
- LEDs are pre-wired **HIGH** active control
- 7-Seg Display are pre-wired **LOW** active control
- Push buttons are pre-wired **LOW** when they are at rest
- DIP switches generate **HIGH** when tuned up and generate **LOW** when tuned down



Test Your FPGA Board (1/3)

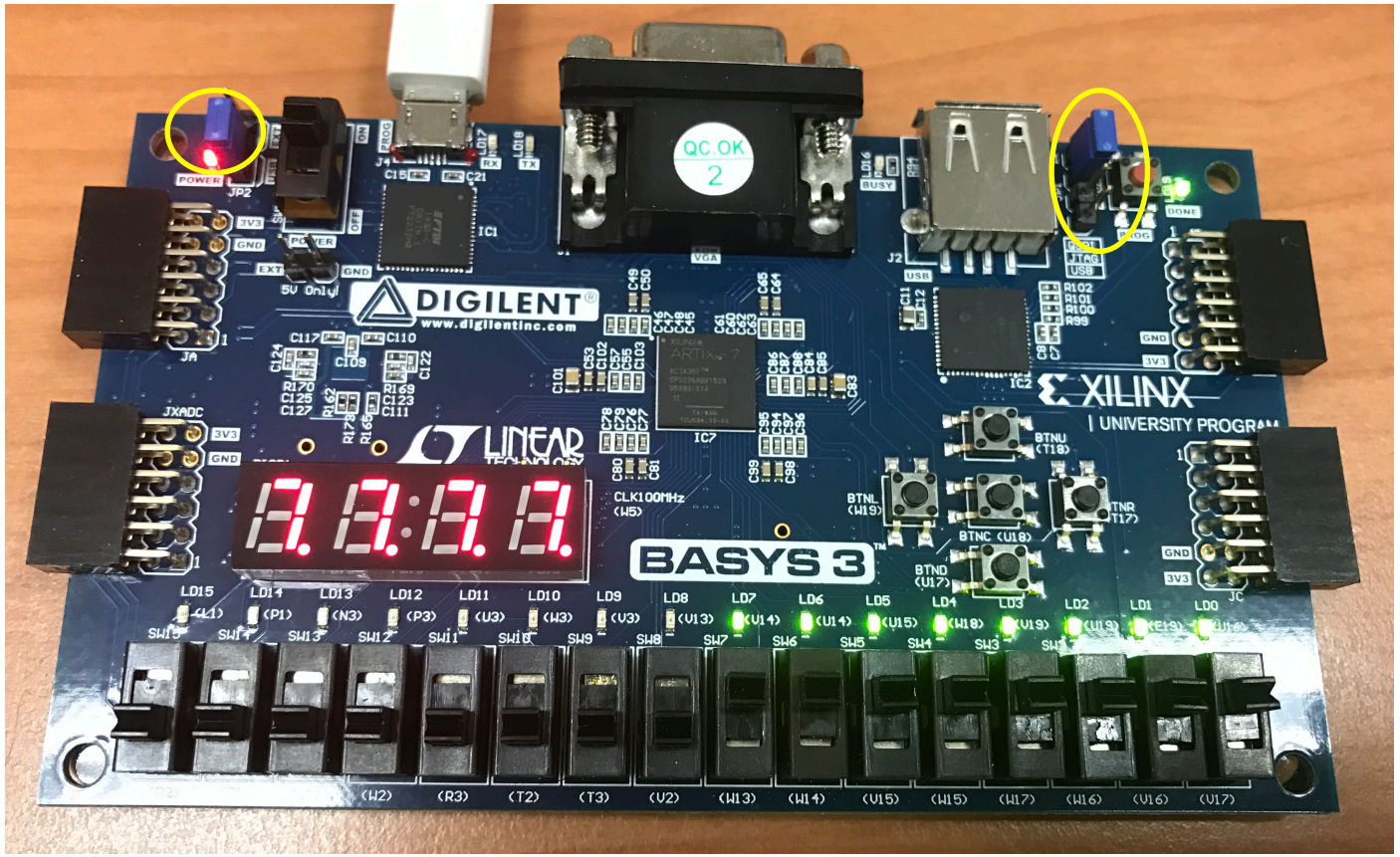
- Connect the demo board to PC and also the power supply

- 所有的線接好再開電源
- 關閉電源後再拔所有的線
- 勿用導體接觸針腳
- 插座別插反了
- 避免長時間開機



Test Your FPGA Board (2/3)

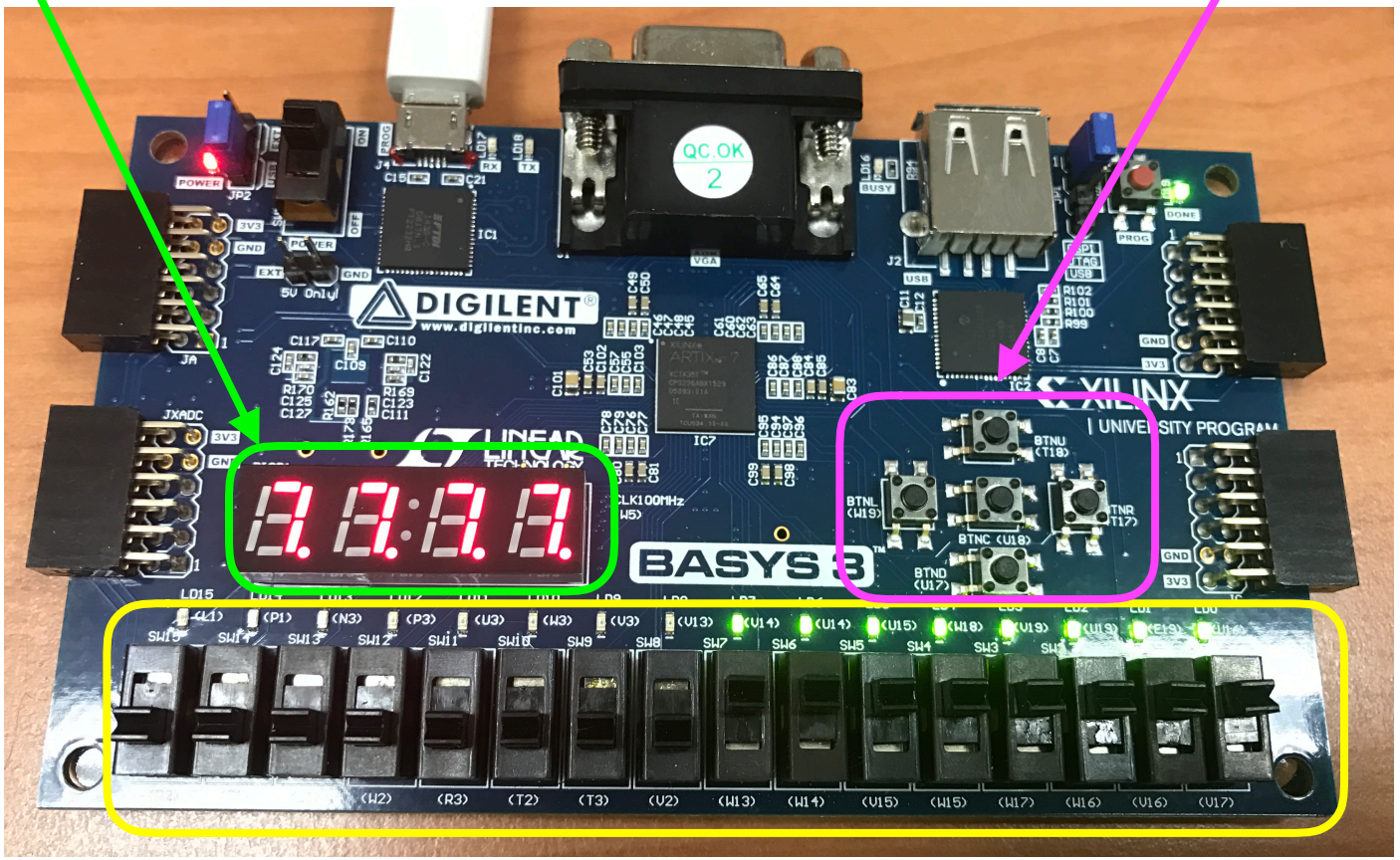
- Don't move or remove the jumpers



Test Your FPGA Board (3/3)

BCD counts on 7-Seg Displays continuously

Push buttons control the 7-Seg display



DIP Switch control the LEDs

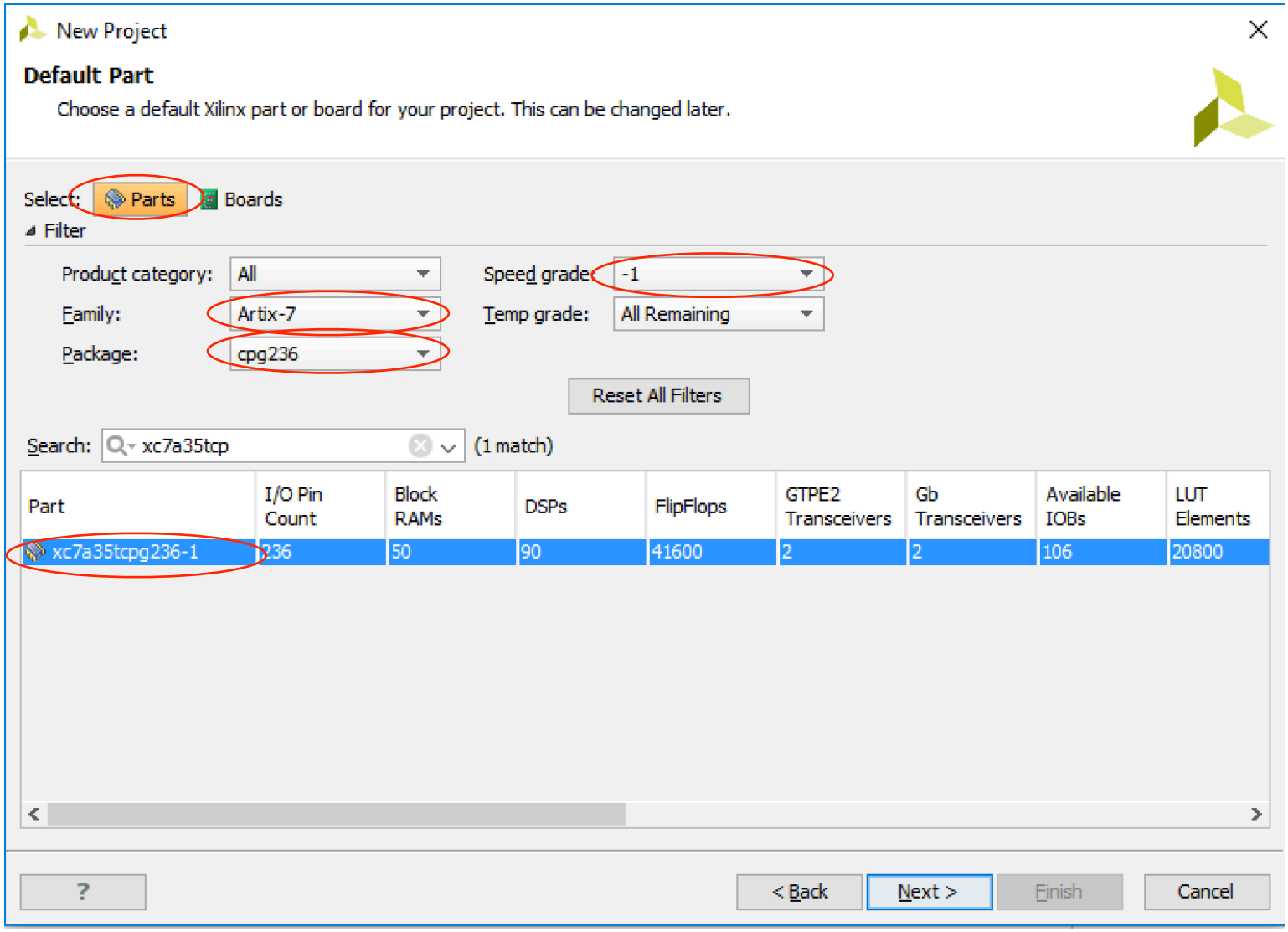
FPGA Emulation Using Xilinx Vivado

Design Flow

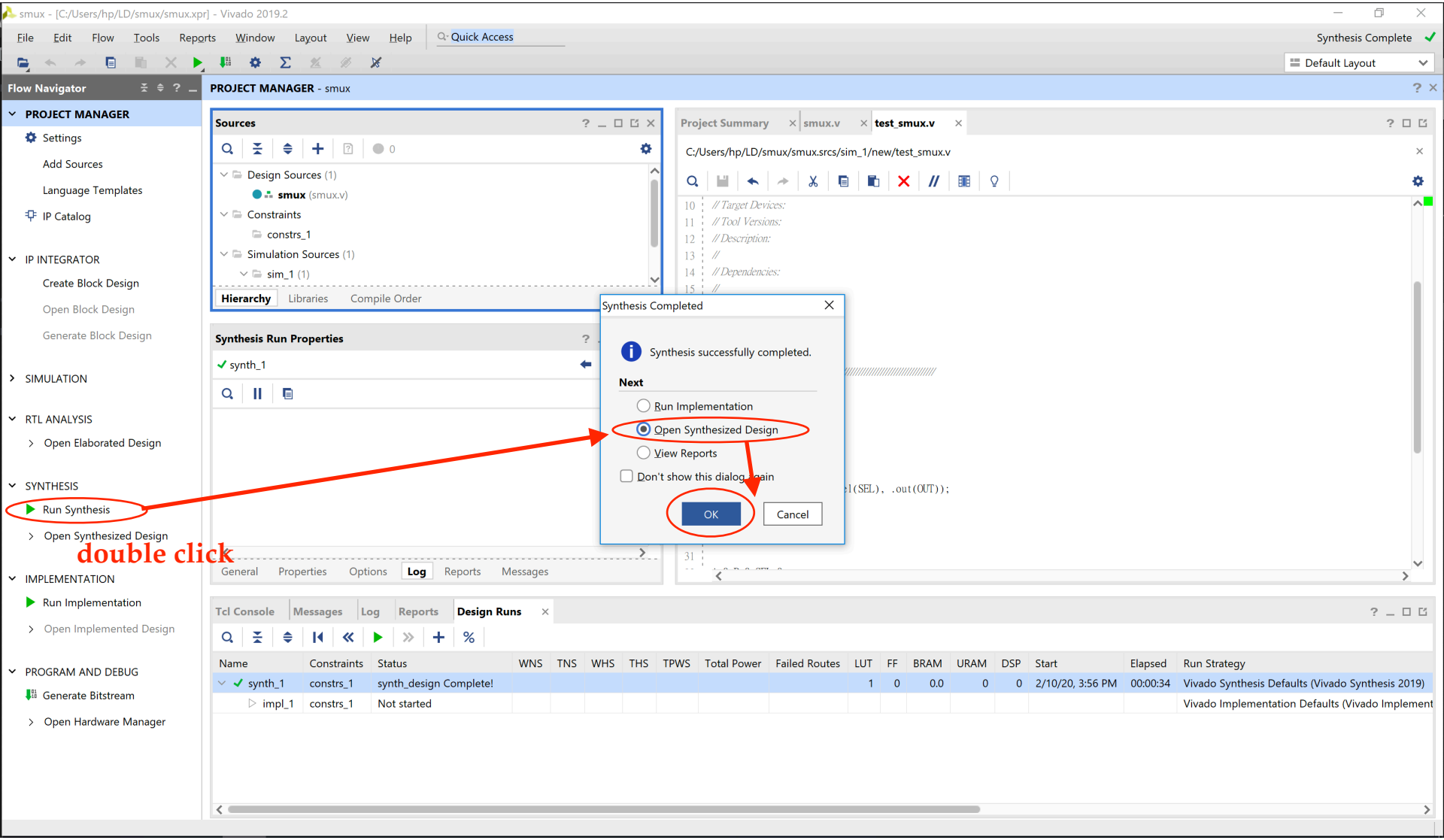
- Design Source Preparation
  - Design modules (.v)
  - Design constraints (I/O pin assignments) (.xdc)
- Design Simulation
  - testbench (.v)
- Design Synthesis and Implementation

Create New Project (2/3)

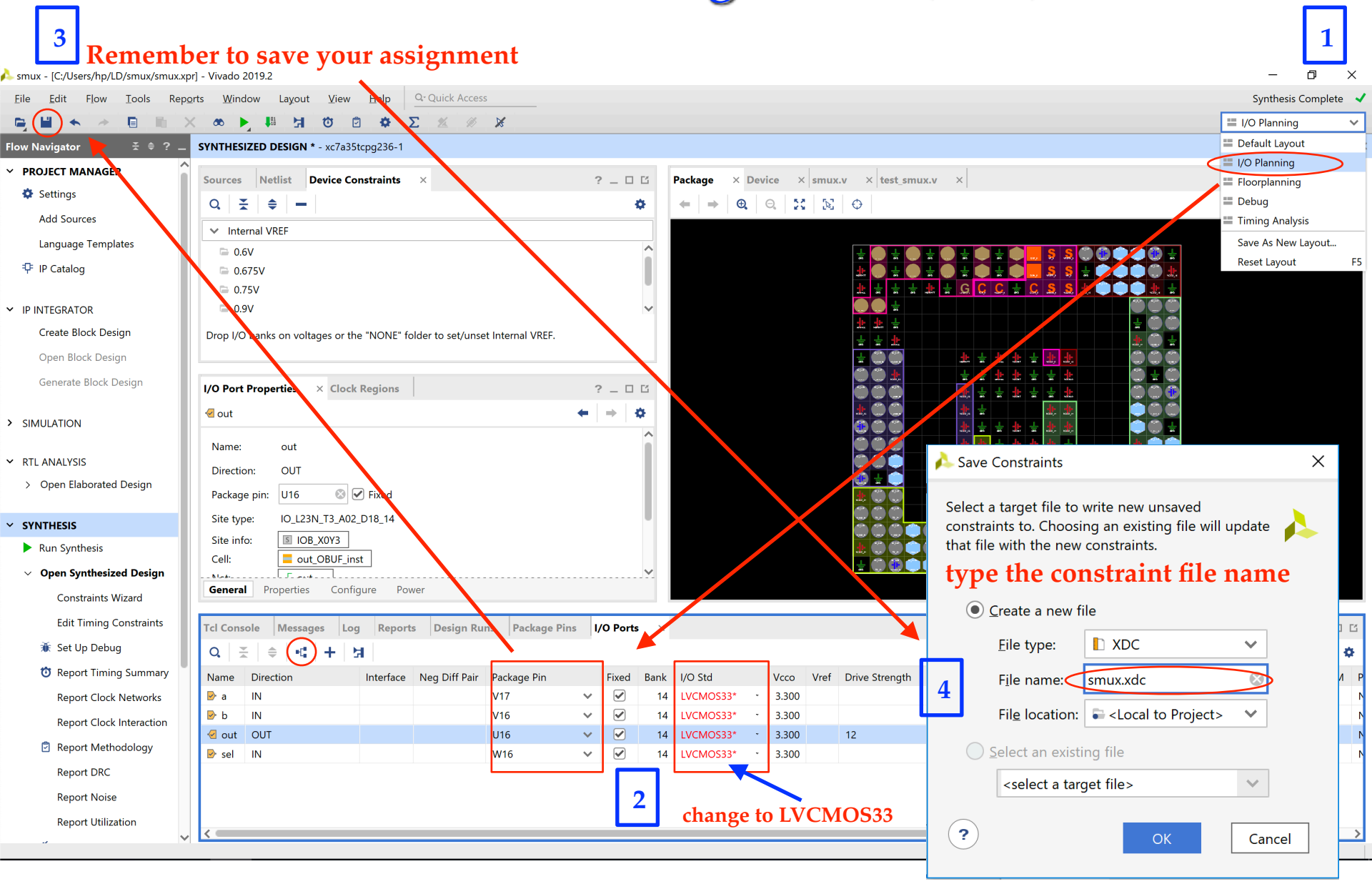
Continue from previous unit slides



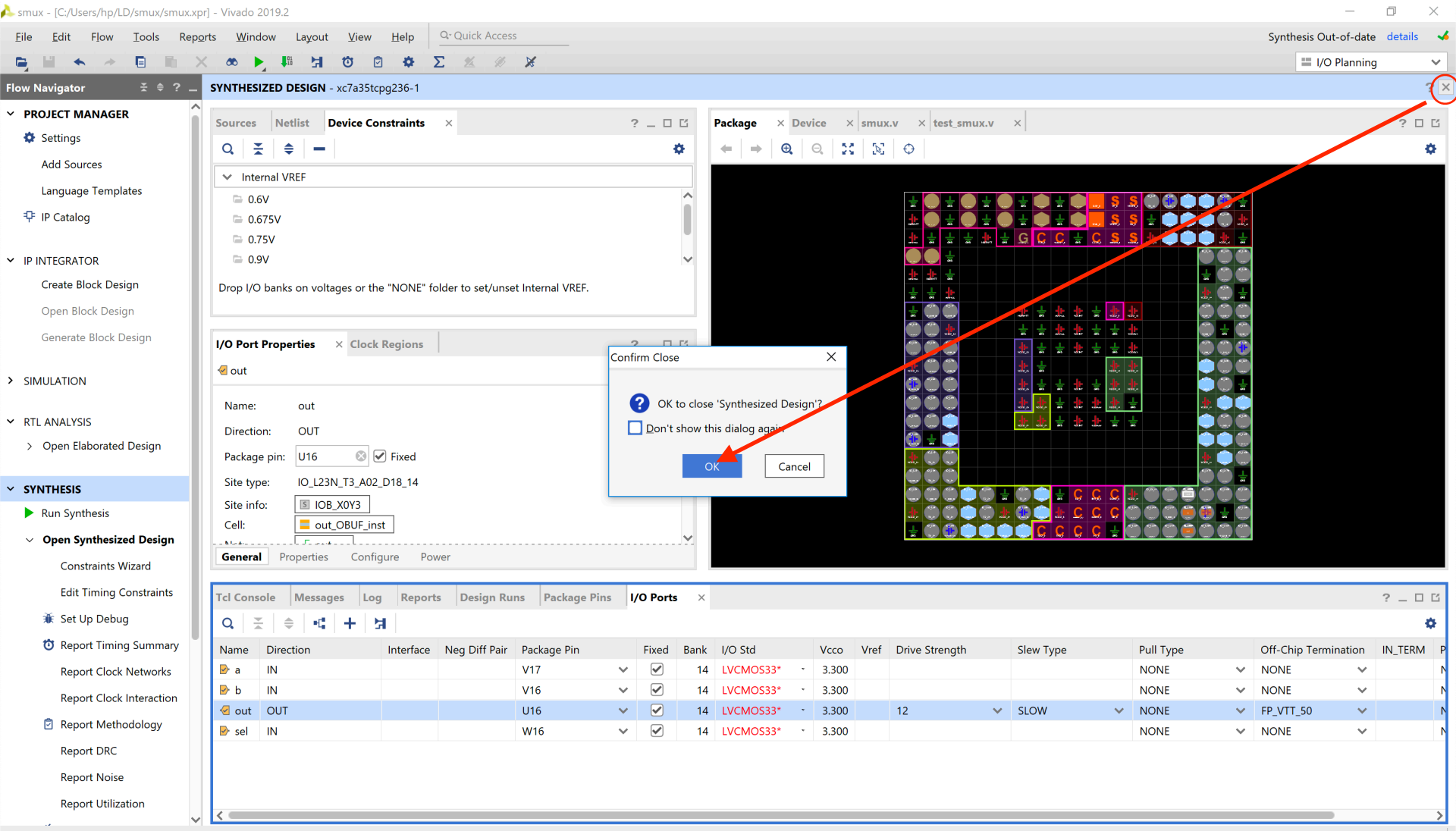
I/O Pins Assignment (1/4)



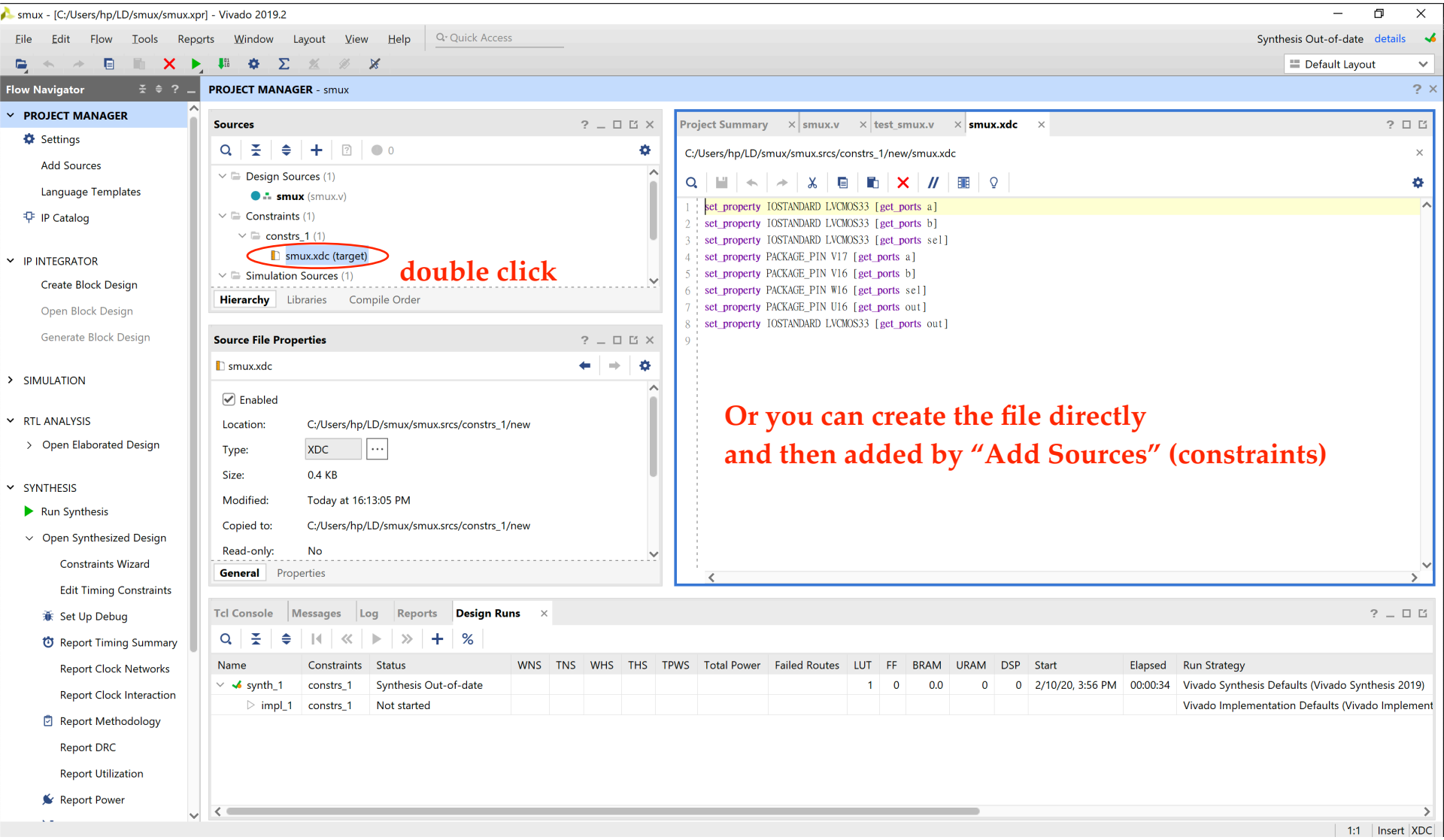
I/O Pins Assignment (2/4)



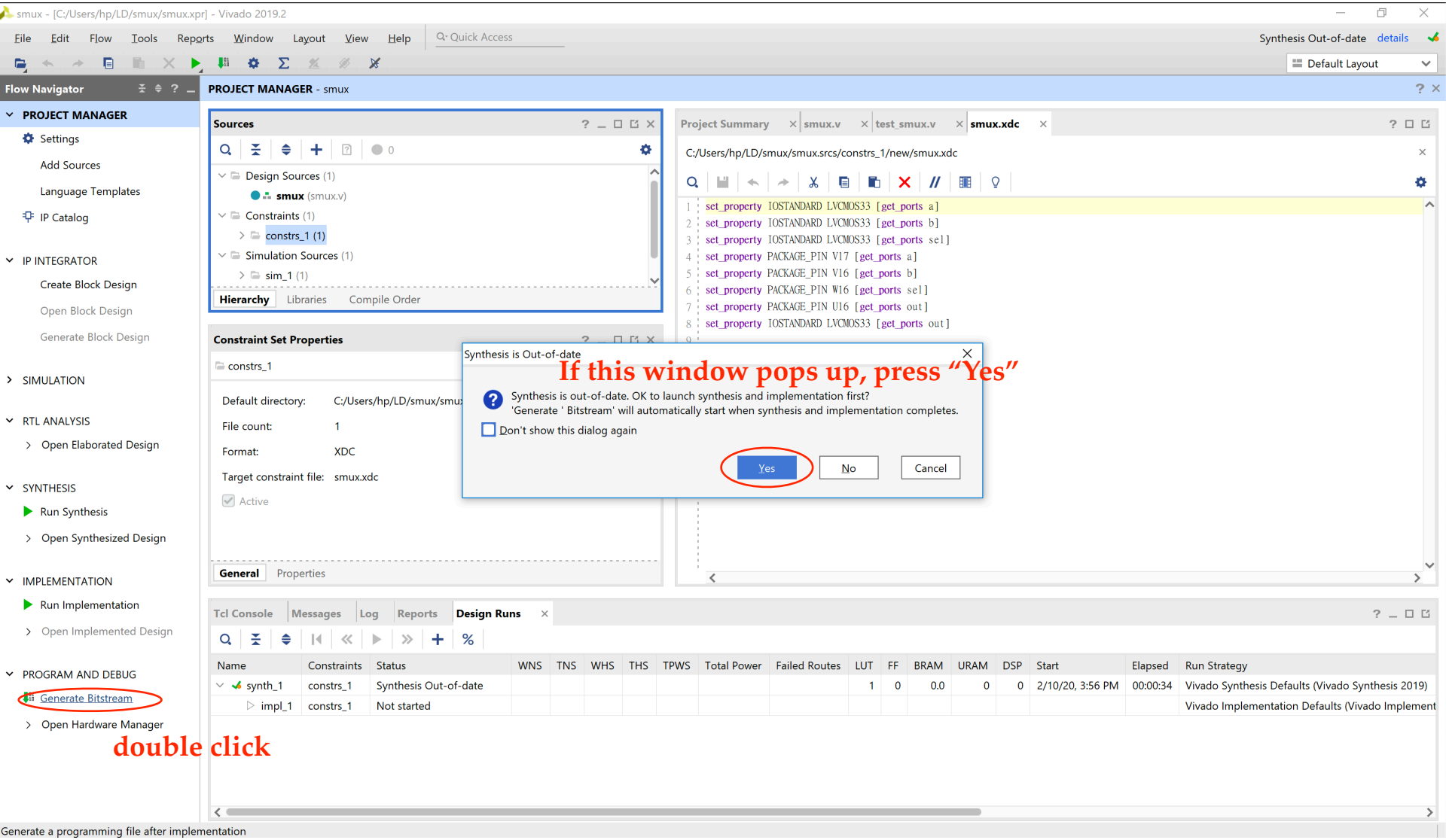
I/O Pins Assignment (3/4)



I/O Pins Assignment (4/4)

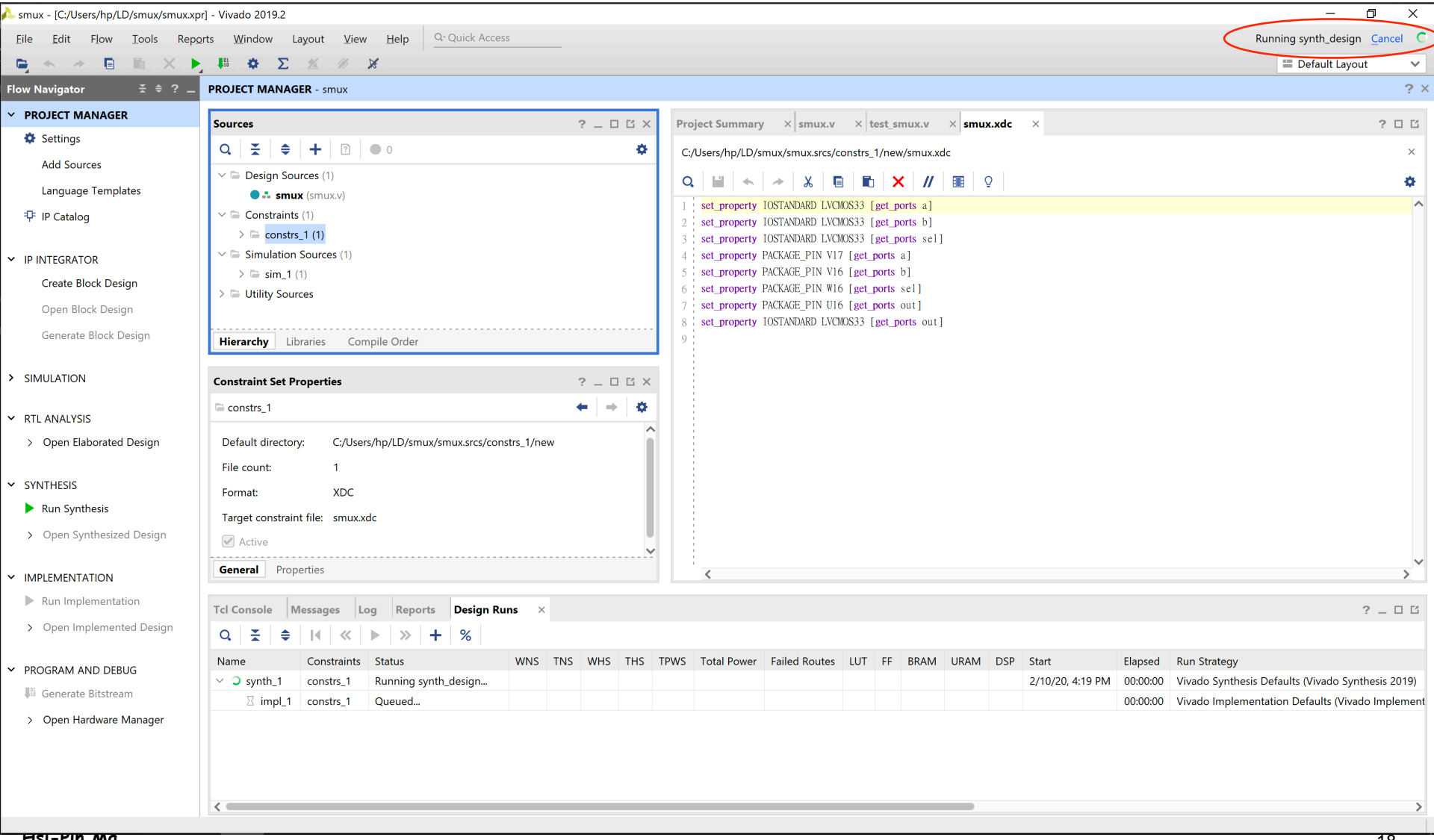


Synthesis and Implementation (1/7)

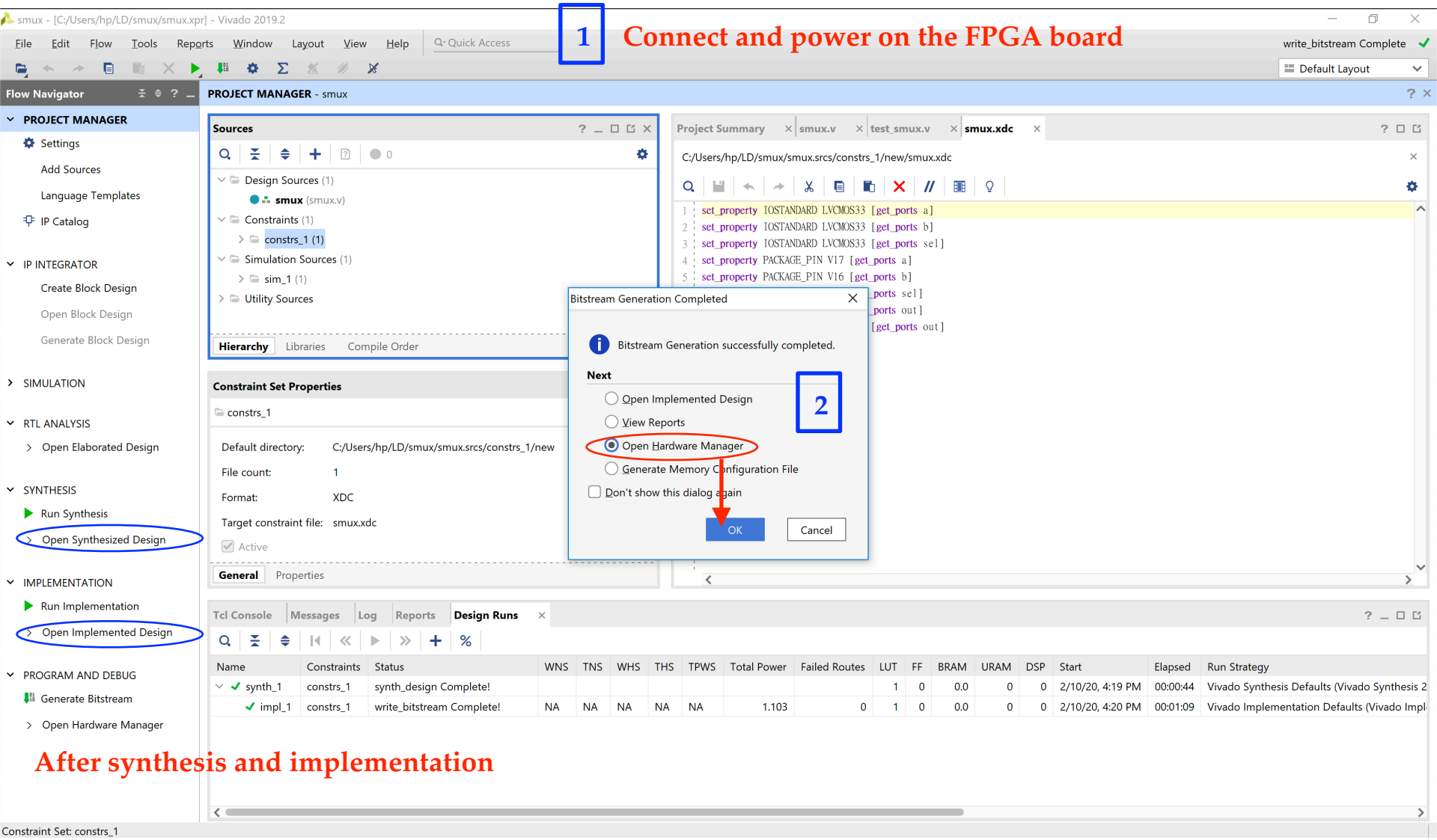


Synthesis and Implementation (2/7)

implementation progress information

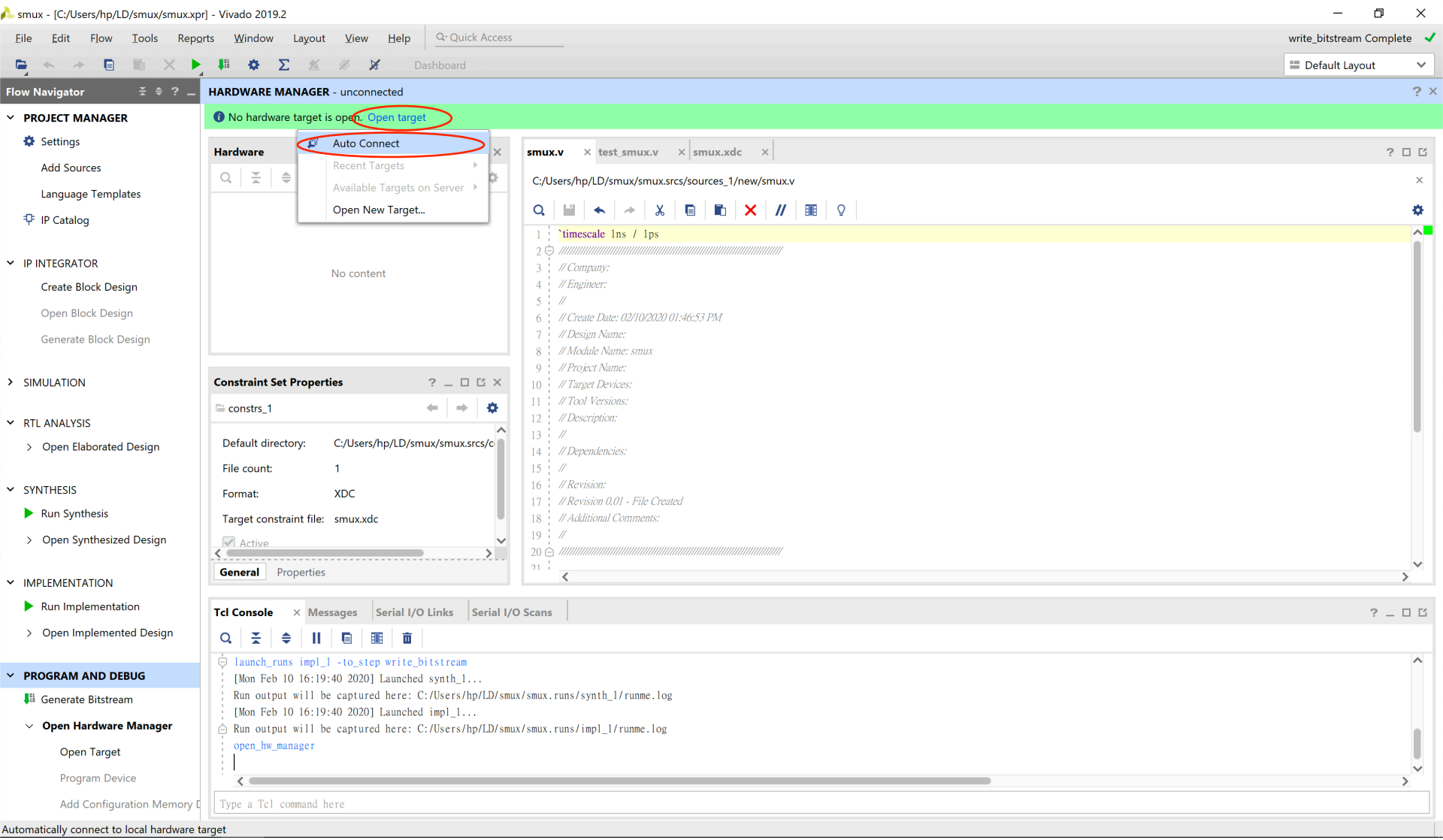


Synthesis and Implementation (3/7)

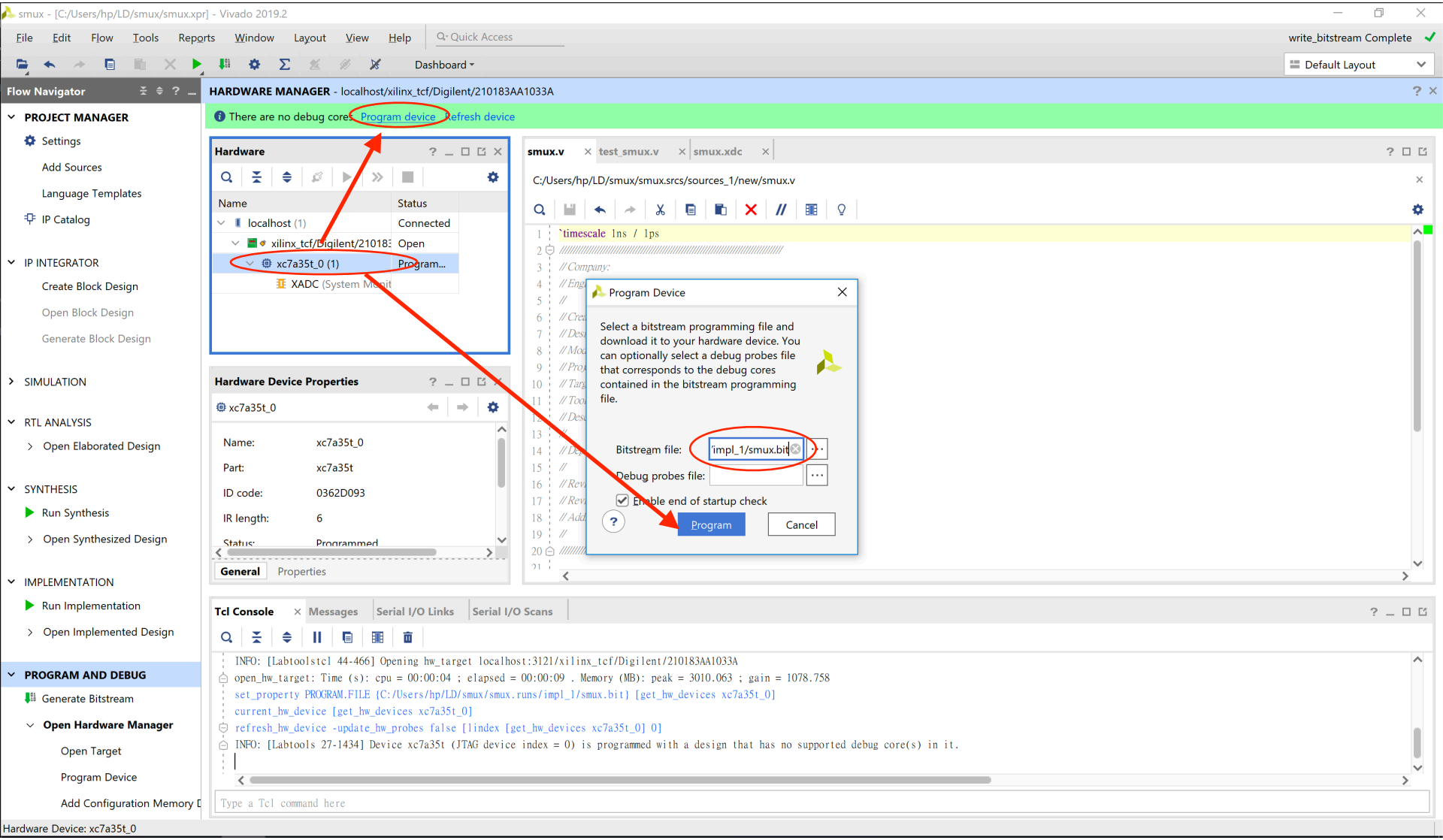


After synthesis and implementation

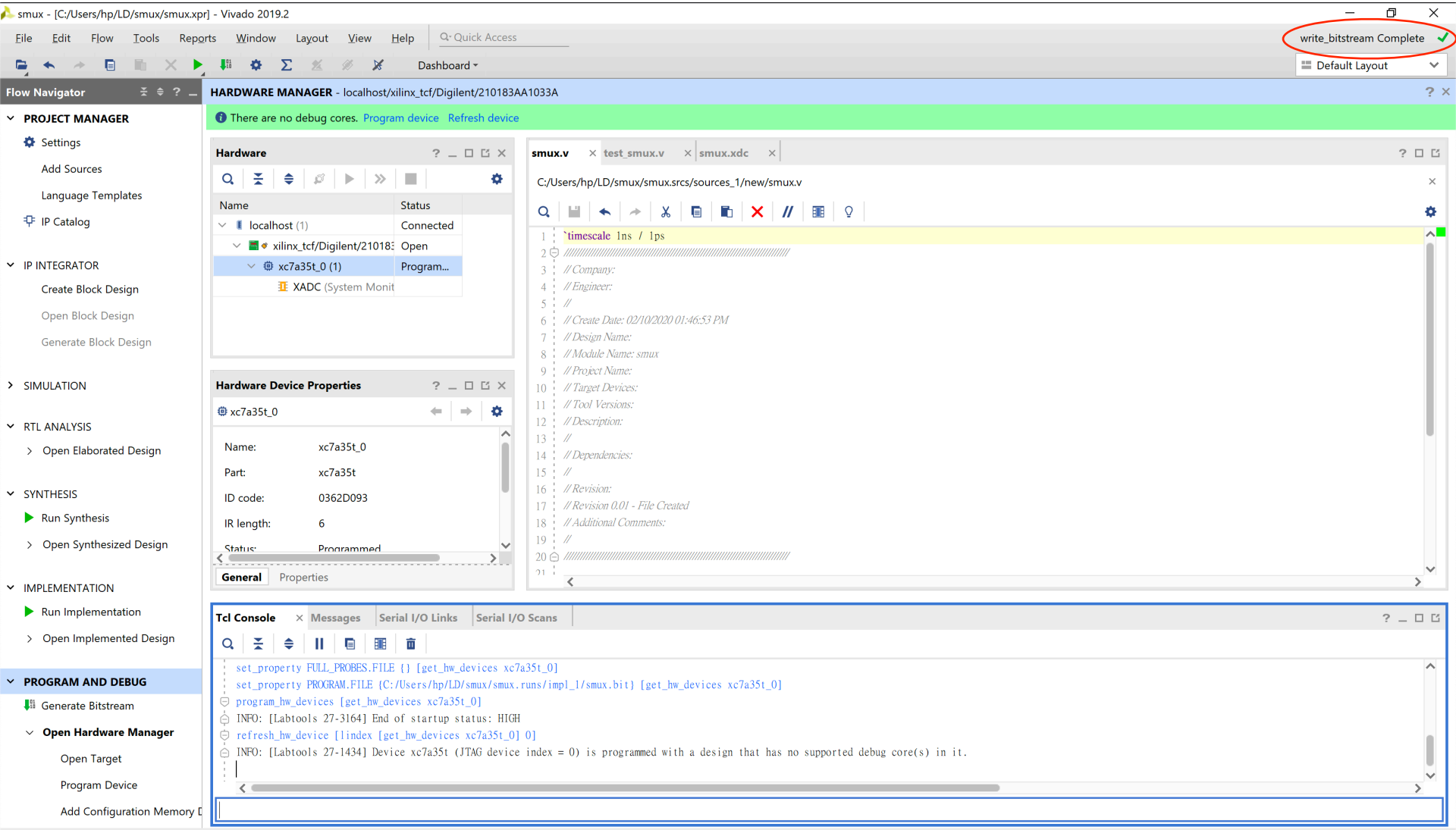
Synthesis and Implementation (4/7)



Synthesis and Implementation (5/7)

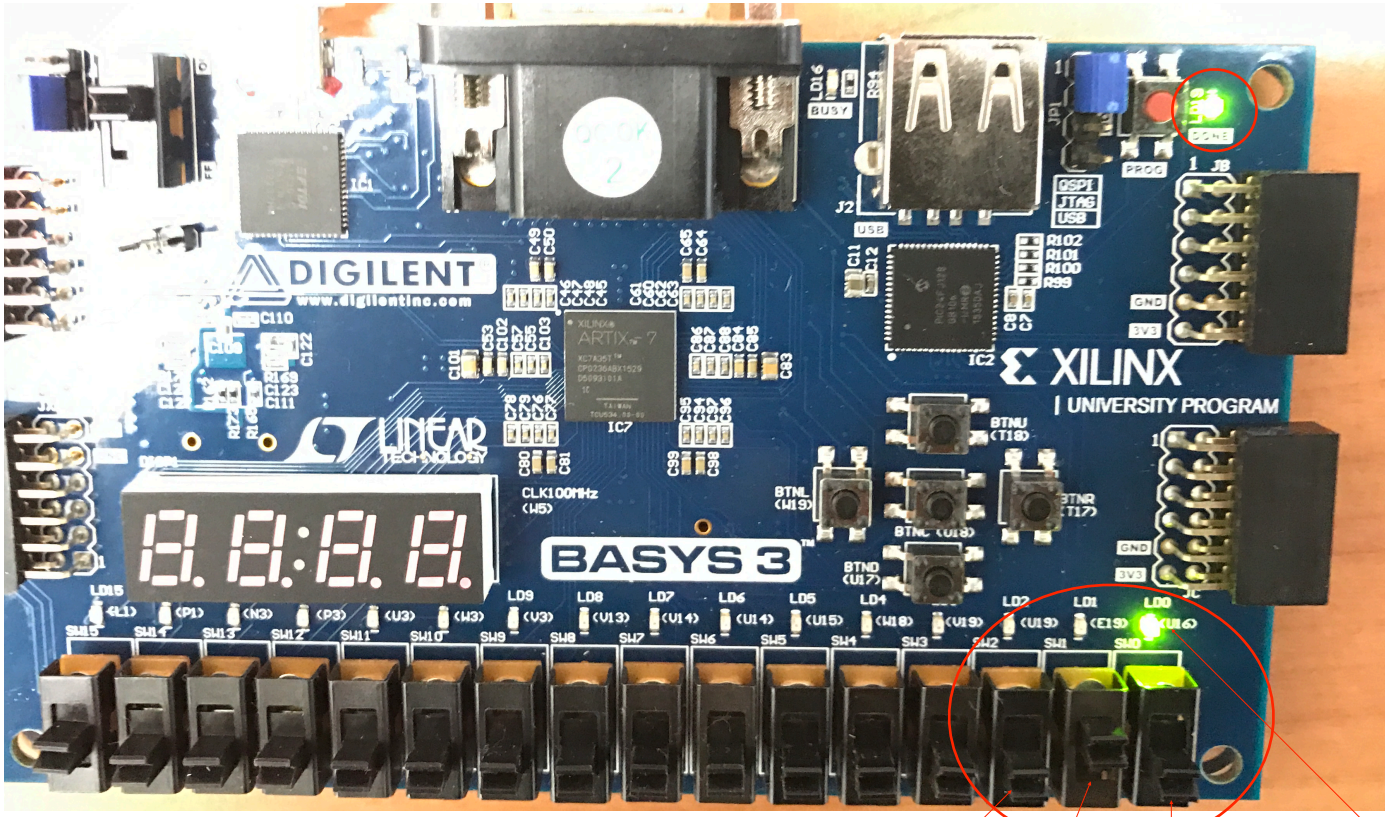


Synthesis and Implementation (6/7)



Synthesis and Implementation (7/7)

Program Finished



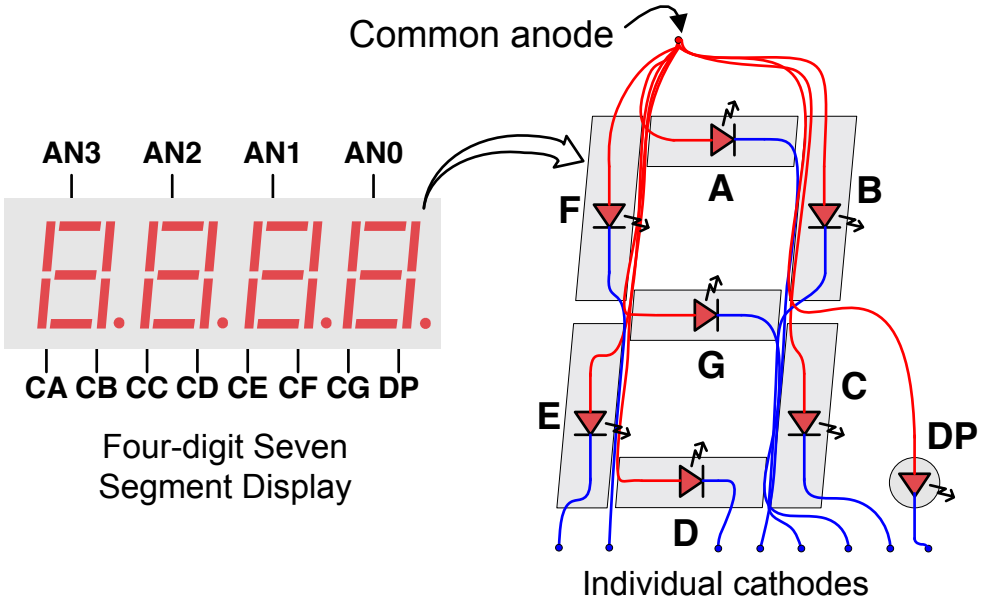
sel=0, b=1, a=0 ==> out=1

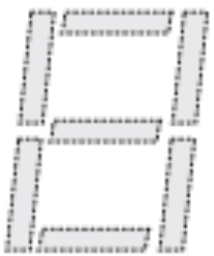
Some Notes

- Sometimes, the database of the design will be corrupted, and any changes will not take effect or your board behaves weird.
  - Open a new project with fresh source files.
- Look into the 'Errors' or 'Warnings' windows to debug your design.
- If you finish your lab at dorm and want to bring it to the lab for demo
  - DO NOT copy the entire directory to the lab and use the same directory for demo
    - Just copying the .v and .xdc files to the lab is sufficient.
    - Use 'New Project' in the lab and open the existing source files to re-implement your design for demo

7-Segment Display (1/2)

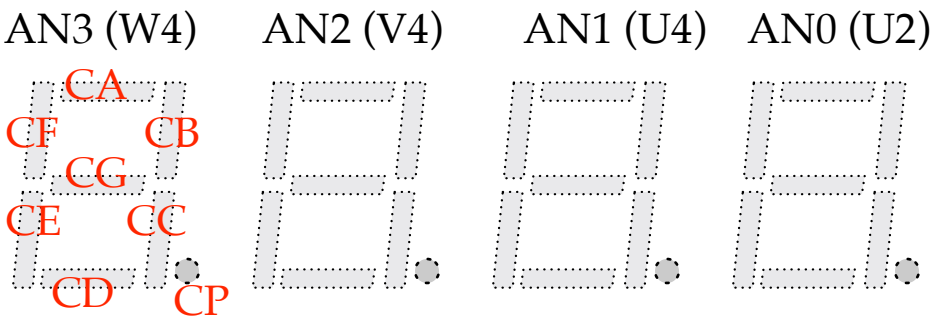
- The anodes of segments forming each digit on all four displays are connected to the same FPGA pin (AN3, AN2, AN1, AN0) (common anode)





7-Segment Display (2/2)

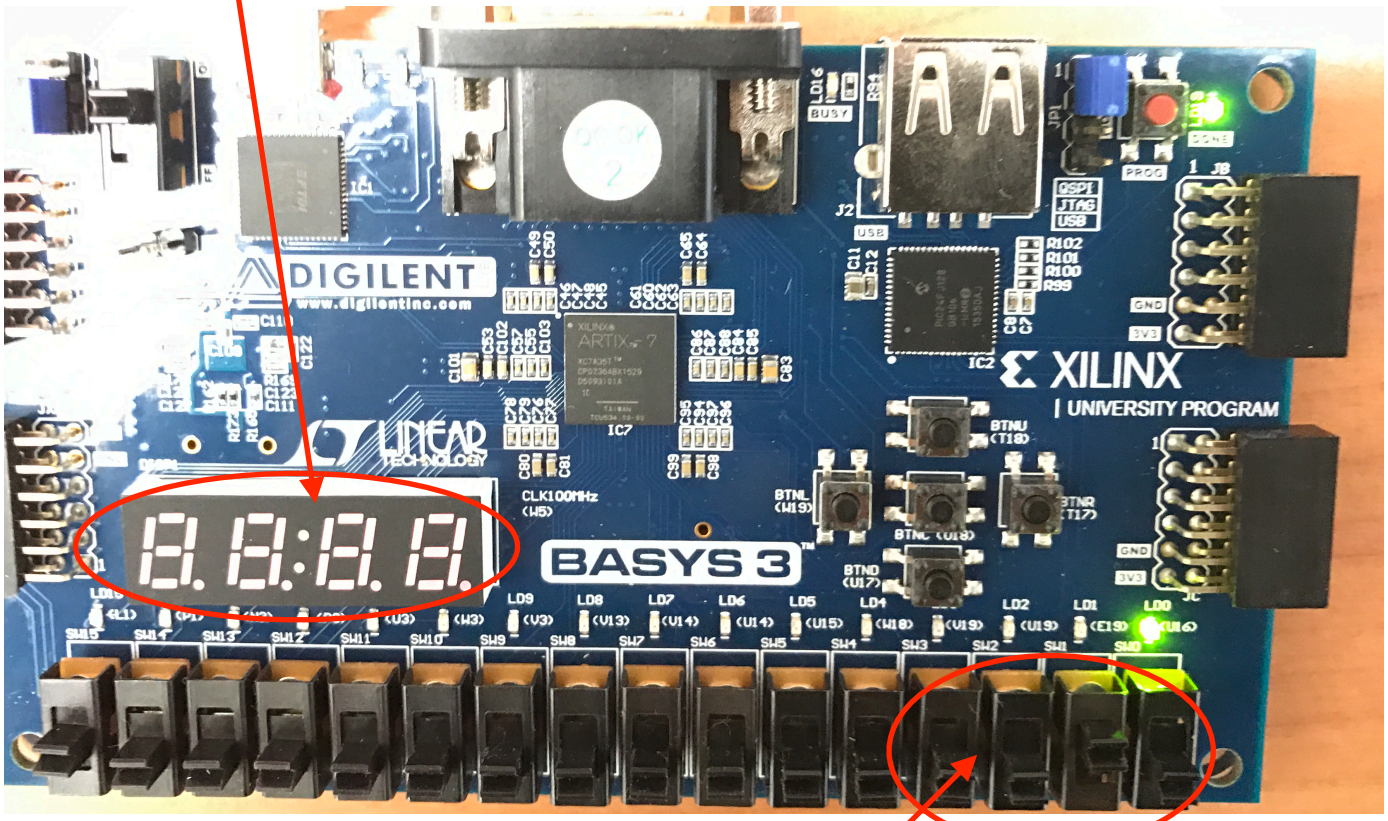
- 8 pins to control each 7-segment display
  - Including the point
- 4 pins (W4,V4,U4,U2) to choose which 7-seg to display
- Device is low activated



Symbol	CA	CB	CC	CD	CE	CF	CG	CP
FPGA Pin	W7	W6	U8	V8	U5	V5	U7	V7

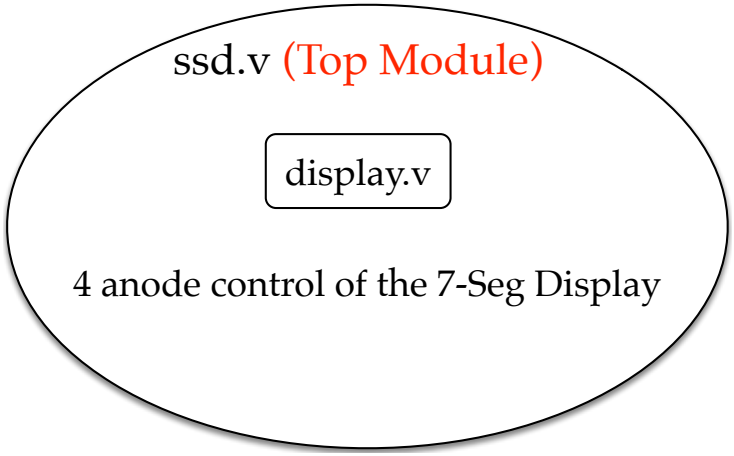
Example

All 4 digits of 7-Seg Display show same number (BCD number output)

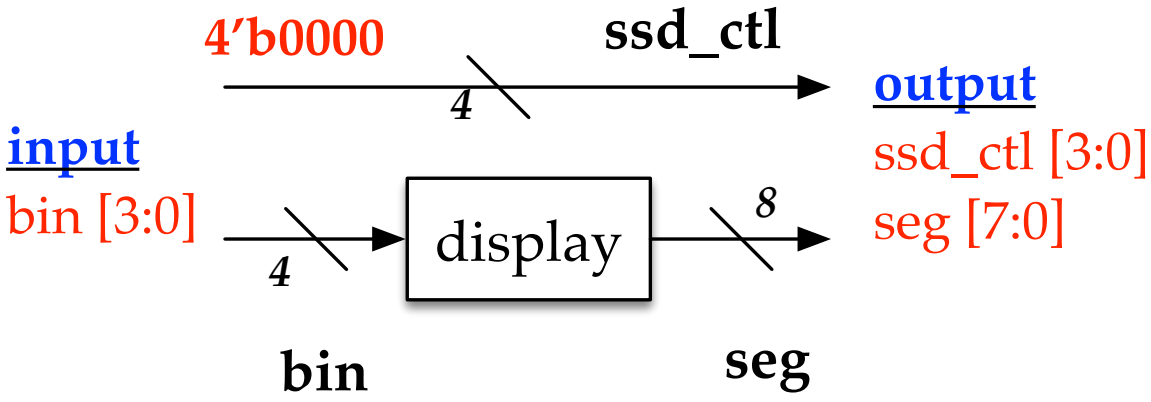


Block Diagram for 7-Seg Display

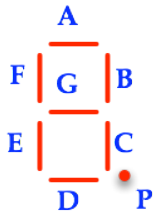
System Hierarchy



Block Diagram



A BCD to Seven-Segment Display Decoder




```
// define segment codes
`define SS_0 8'b00000011
`define SS_1 8'b10011111
`define SS_2 8'b00100101
`define SS_3 8'b00001101
`define SS_4 8'b10011001
`define SS_5 8'b01001001
`define SS_6 8'b01000001
`define SS_7 8'b00011111
`define SS_8 8'b00000001
`define SS_9 8'b00001001
```

```
module display(segs, bin);
output [7:0] segs;
input [3:0] bin;
reg [7:0] segs;

always @*
 case (bin)
   4'd0: segs = `SS_0;
   4'd1: segs = `SS_1;
   4'd2: segs = `SS_2;
   4'd3: segs = `SS_3;
   4'd4: segs = `SS_4;
   4'd5: segs = `SS_5;
   4'd6: segs = `SS_6;
   4'd7: segs = `SS_7;
   4'd8: segs = `SS_8;
   4'd9: segs = `SS_9;
   default: segs = 8'b0000000;
 endcase
endmodule
```

Top Module (ssd.v)

```
module ssd(seg, bin, ssd_ctl);
output [3:0] ssd_ctl;
output [7:0] seg;
input [3:0] bin;

display U0(.segs(seg),.bin(bin));

assign ssd_ctl = 4'b0000;

endmodule
```

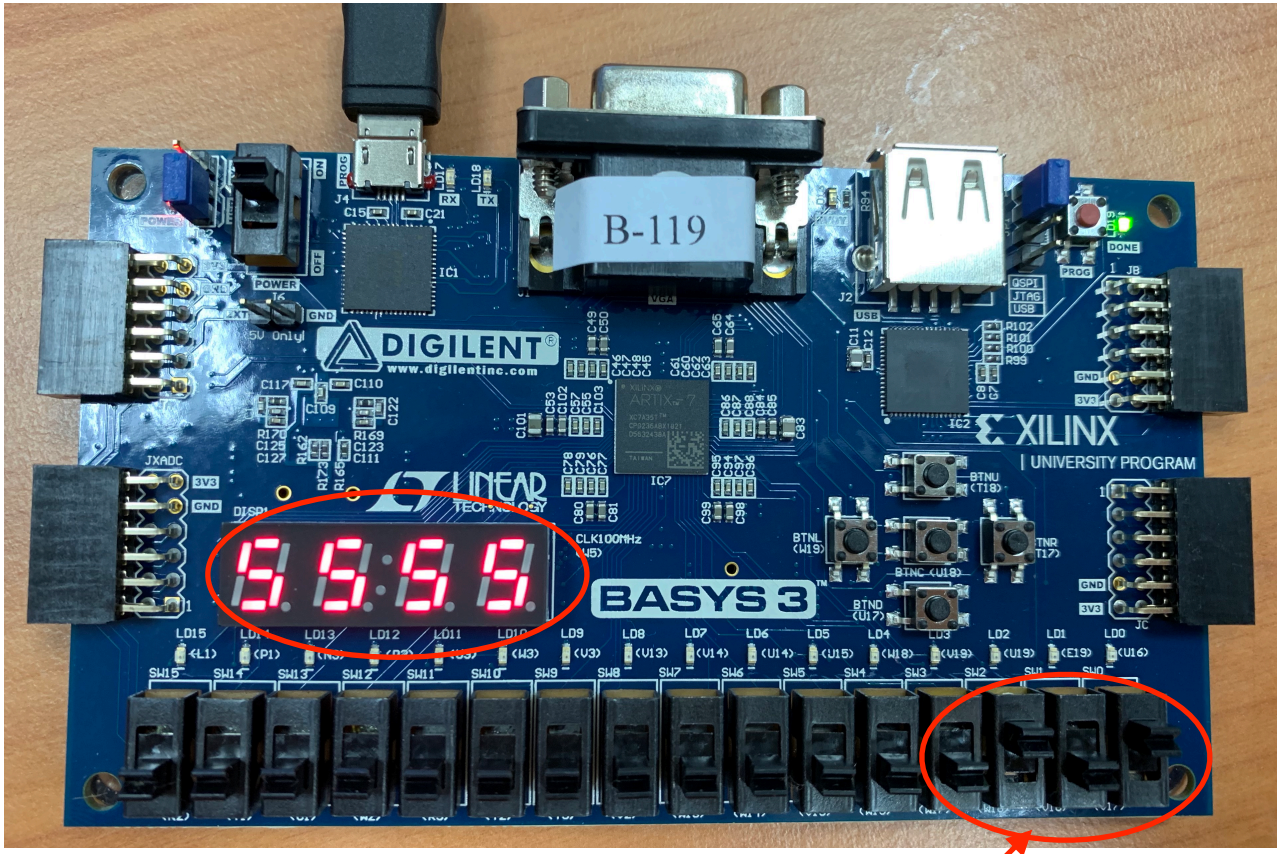
ssd.xdc (1/2)

```
# Four anode control signals
set_property PACKAGE_PIN W4 [get_ports {ssd_ctl[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {ssd_ctl[3]}]
set_property PACKAGE_PIN V4 [get_ports {ssd_ctl[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {ssd_ctl[2]}]
set_property PACKAGE_PIN U4 [get_ports {ssd_ctl[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {ssd_ctl[1]}]
set_property PACKAGE_PIN U2 [get_ports {ssd_ctl[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {ssd_ctl[0]}]

#8 common segment controls
set_property PACKAGE_PIN W7 [get_ports {seg[7]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[7]}]
set_property PACKAGE_PIN W6 [get_ports {seg[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
set_property PACKAGE_PIN U8 [get_ports {seg[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property PACKAGE_PIN V8 [get_ports {seg[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
```

ssd.xdc (2/2)

```
set_property PACKAGE_PIN U5 [get_ports {seg[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property PACKAGE_PIN V5 [get_ports {seg[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property PACKAGE_PIN U7 [get_ports {seg[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
set_property PACKAGE_PIN V7 [get_ports {seg[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]

#4-bit binary input
set_property PACKAGE_PIN W17 [get_ports {bin[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {bin[3]}]
set_property PACKAGE_PIN W16 [get_ports {bin[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {bin[2]}]
set_property PACKAGE_PIN V16 [get_ports {bin[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {bin[1]}]
set_property PACKAGE_PIN V17 [get_ports {bin[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {bin[0]}]
```

Example



binary input: 0101