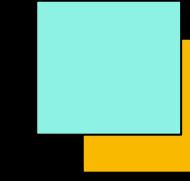
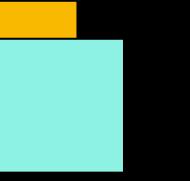


Vivado FPGA Guide

9/23/2021

 NTHU Logic Design Laboratory (Fall. 2021) 

By Prof. Chun-Yi Lee

Agenda

Overview

Preparation

Generate Bit Stream

Design Verification on FPGA

Agenda

Overview

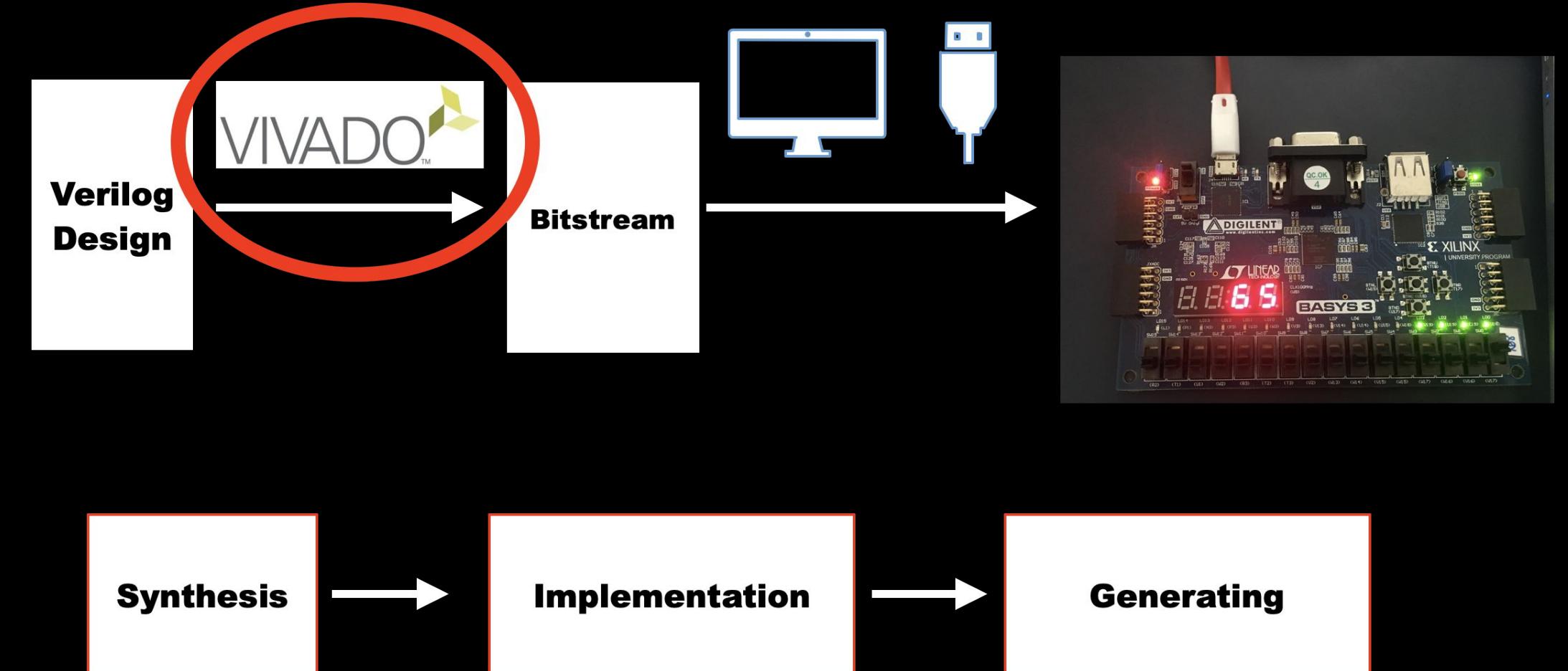
Preparation

Generate Bit Stream

Design Verification on FPGA

Overview (1/1)

- To program our verilog design into the FPGA, it first needs to be turn into Bitstream
- Done with the help from *Vivado*



Agenda

Overview

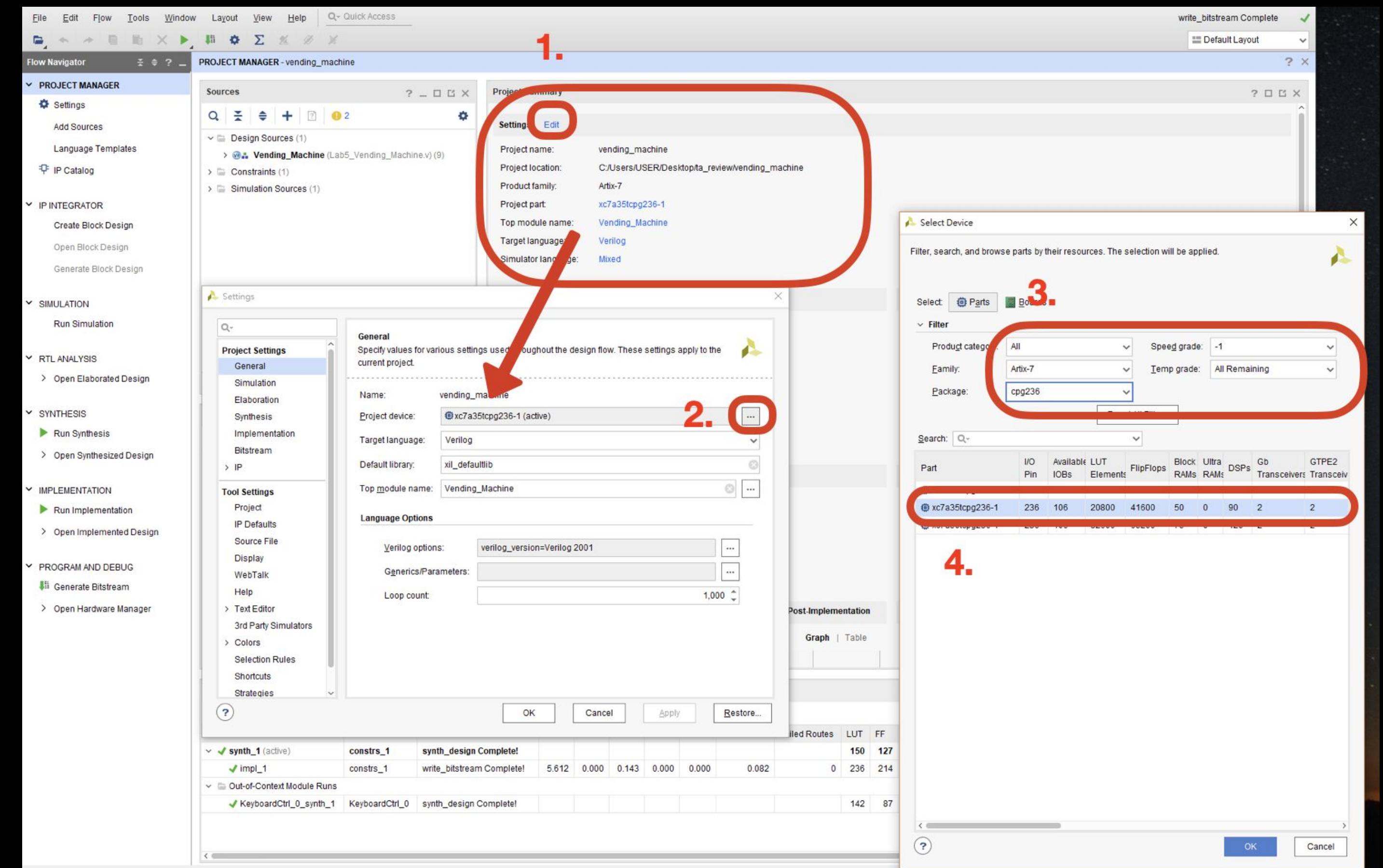
Preparation

Generate Bit Stream

Design Verification on FPGA

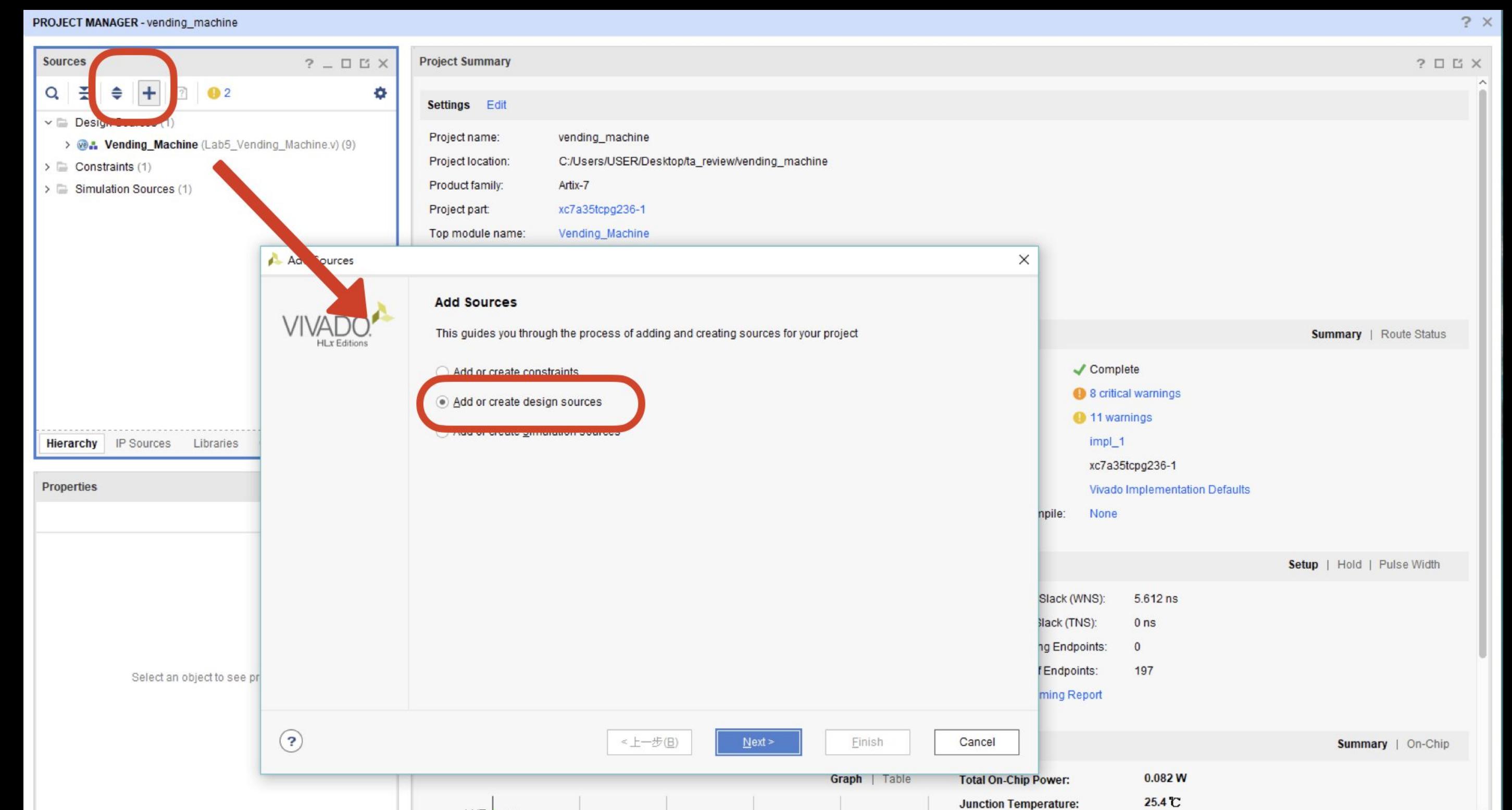
Preparation (1/5)

- First, we need to make sure that the correct board is selected



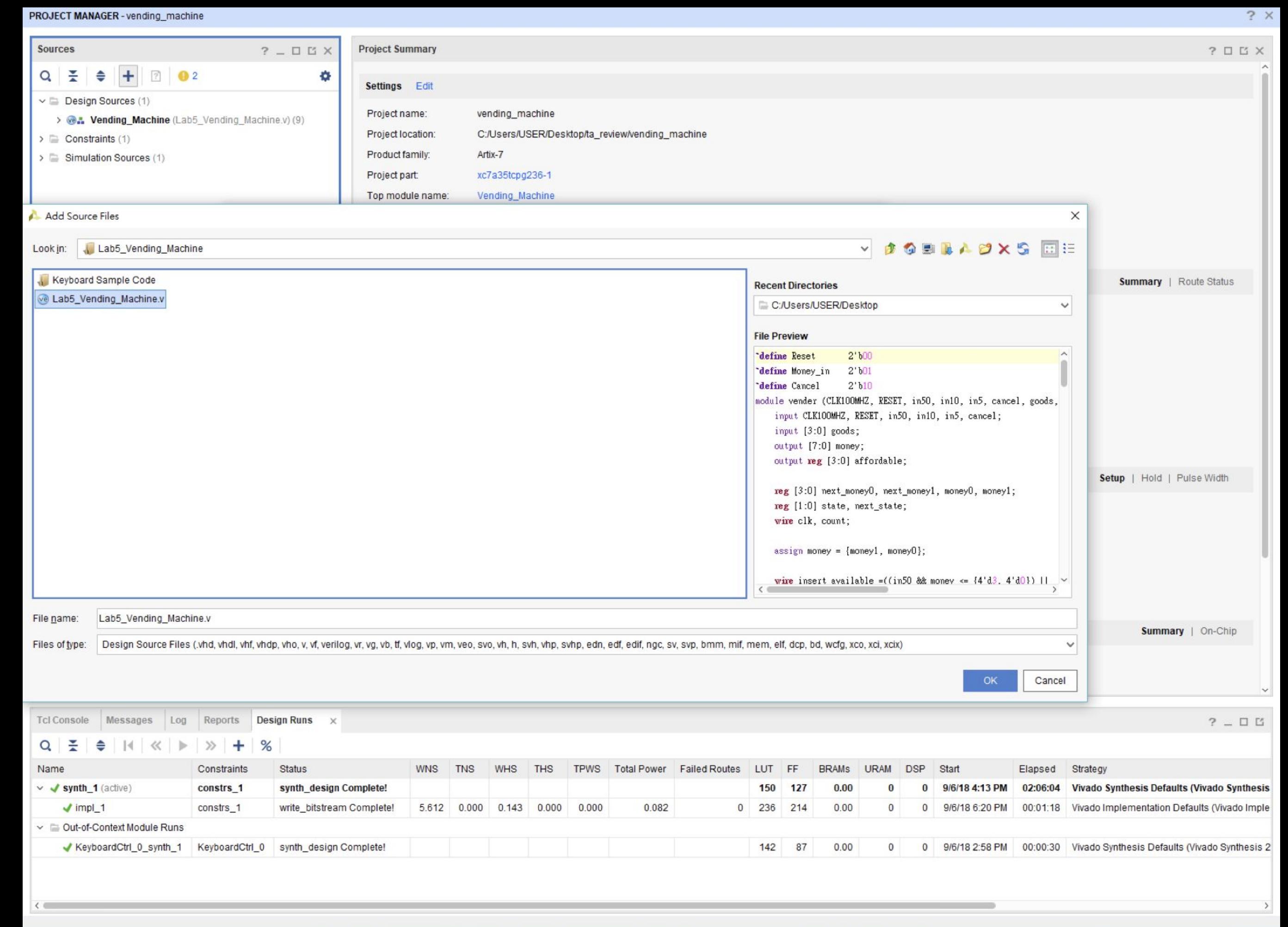
Preparation (2/5)

- Next, we need to add the design source file into the project (ignore this step if it is already added)



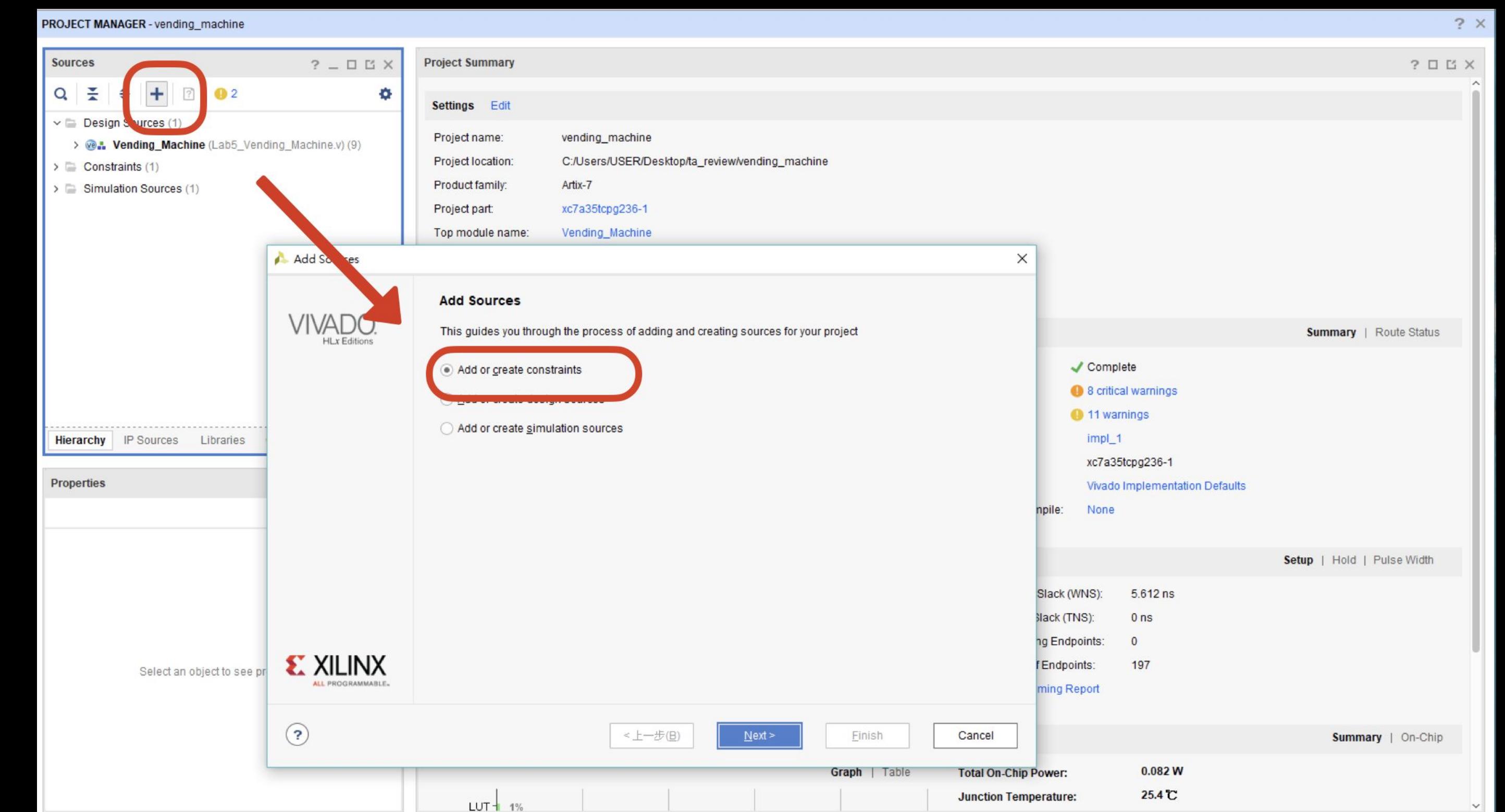
Preparation (3/5)

- Select the design source files you want to add to this project (ignore this step if they are already included in the project.)



Preparation (4/5)

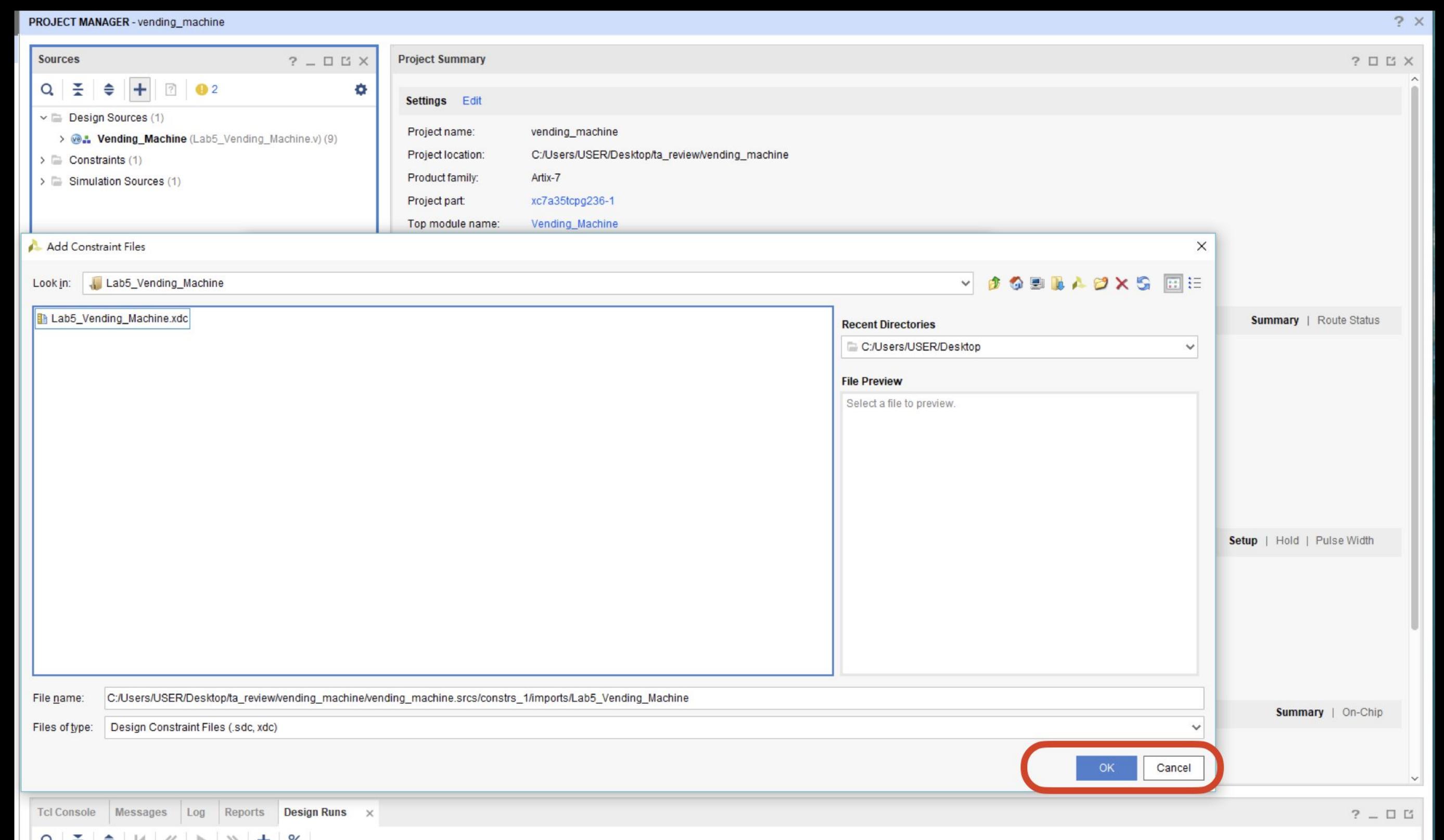
- Finally, we need to add the design constraint (.xdc file) to the project (ignore these steps if it is already included)



Preparation (5/5)

- Select the design

constraint you want to add
(ignore this step if it is
already included)



Agenda

Overview

Preparation

Generate Bit Stream

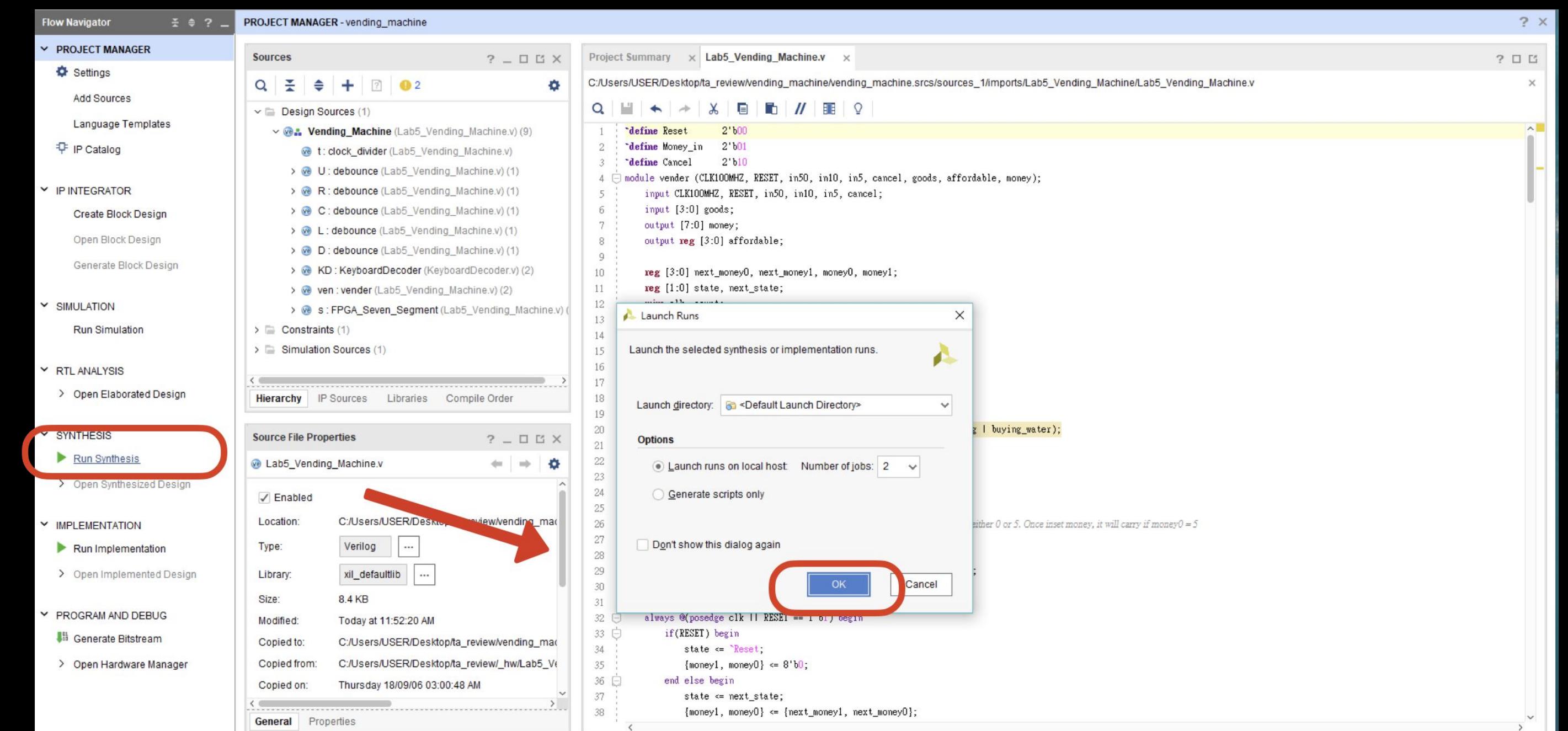
Design Verification on FPGA

Generate Bit Stream (1/7)

- Synthesize our RTL design

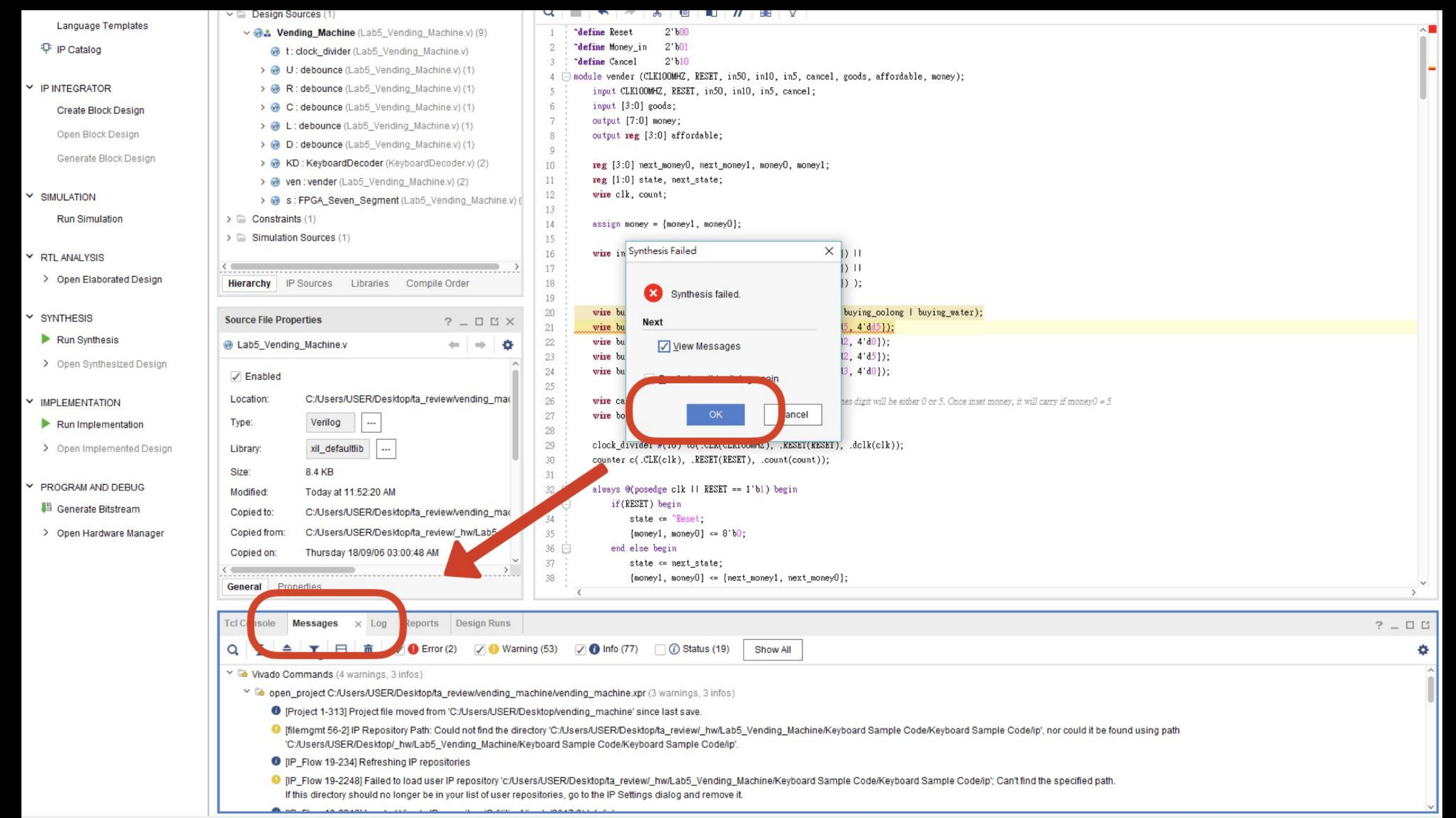
by clicking *Run Synthesis*

and then hit *OK*.



Generate Bit Stream (2/7)

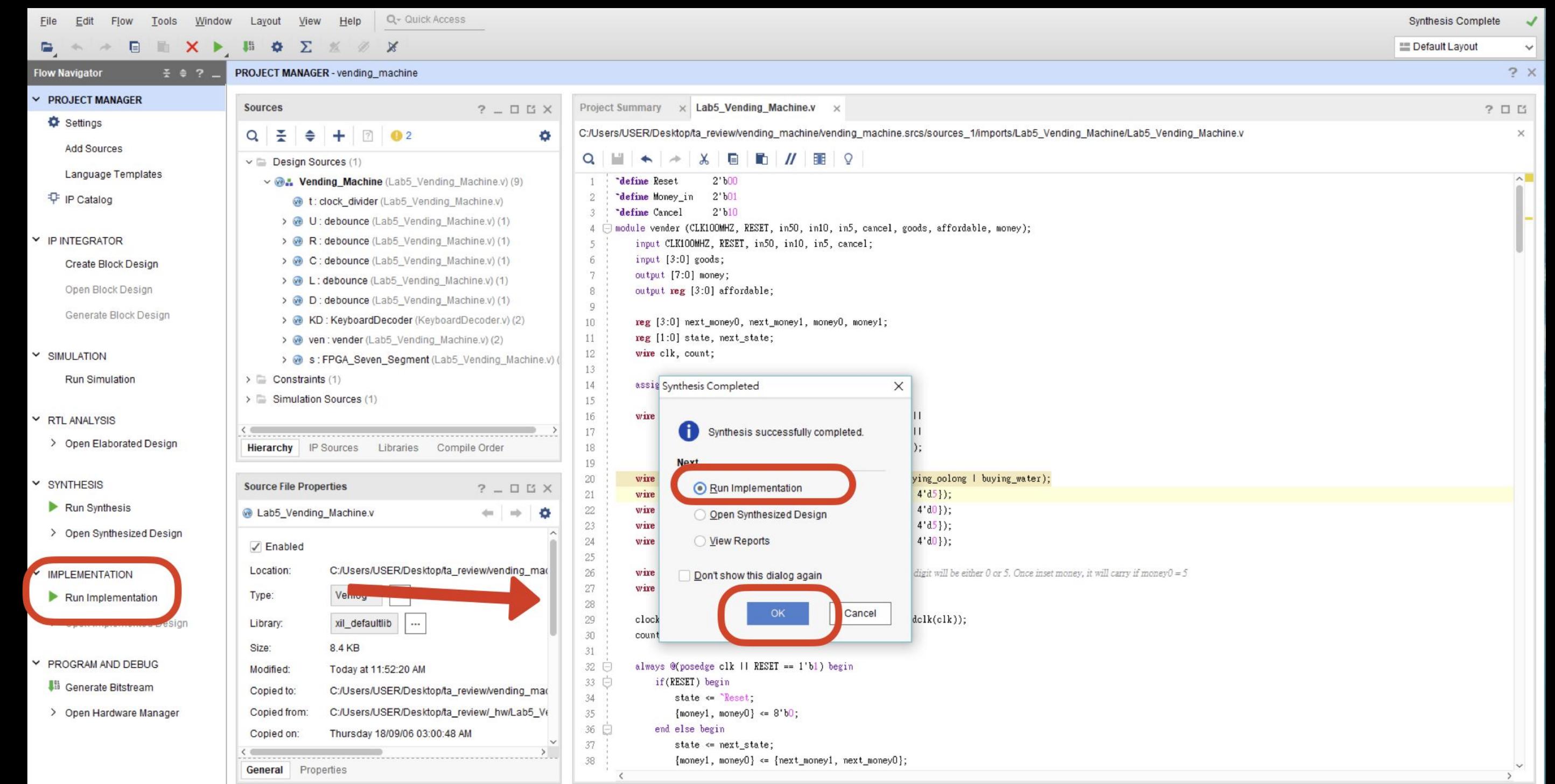
- If there are any error during the synthesis process, go to the Messages window below to check the error(s) and correct them.



Generate Bit Stream (3/7)

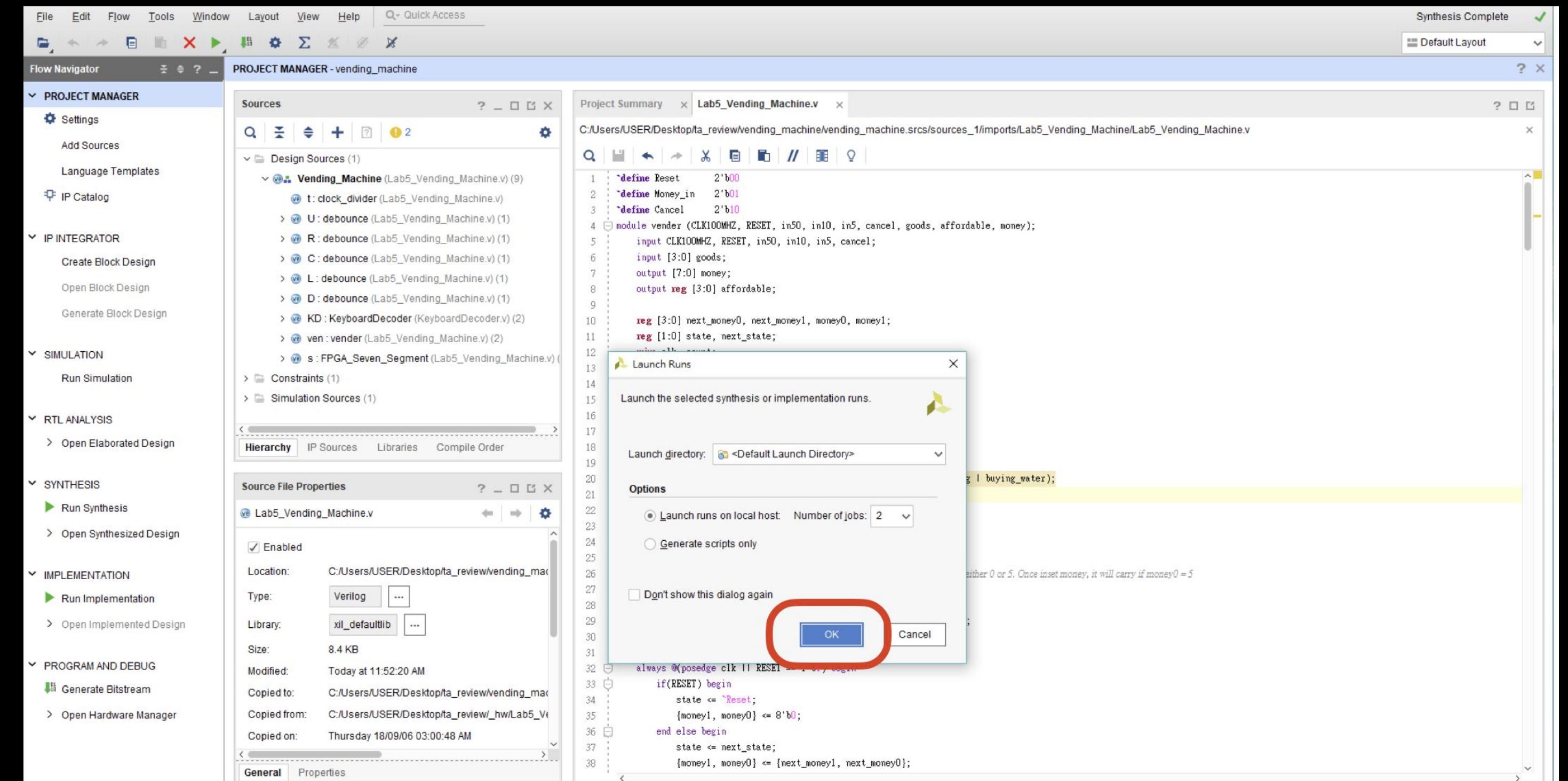
- Next, we need to run implementation to turn the netlist into a complete implementation.

- Hit ***Run Implementation*** and hit ***OK***.



Generate Bit Stream (4/7)

- Click **OK** once more



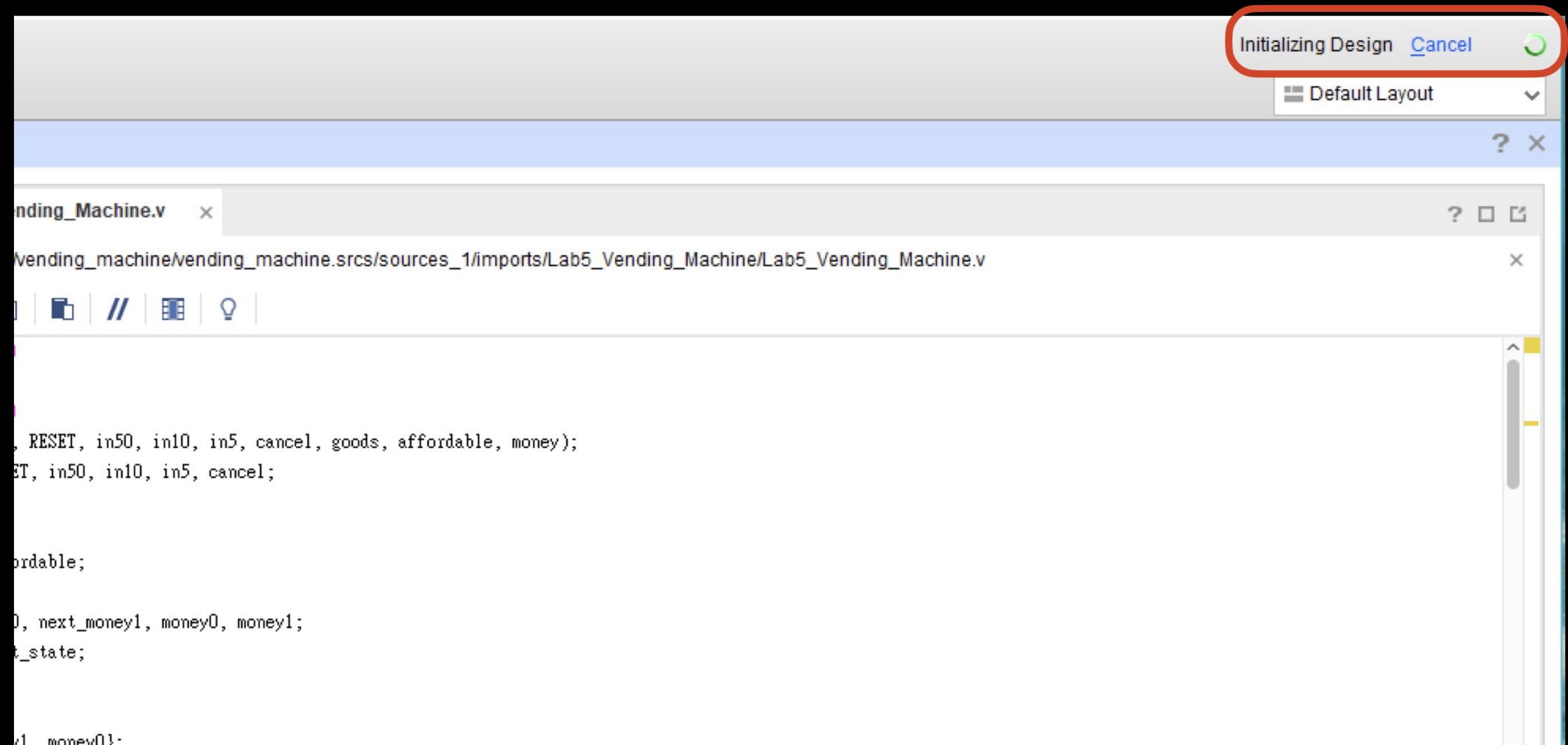
Generate Bit Stream (5/7)

- # • The Run Implementation

process will run in

background, this may take

some time.



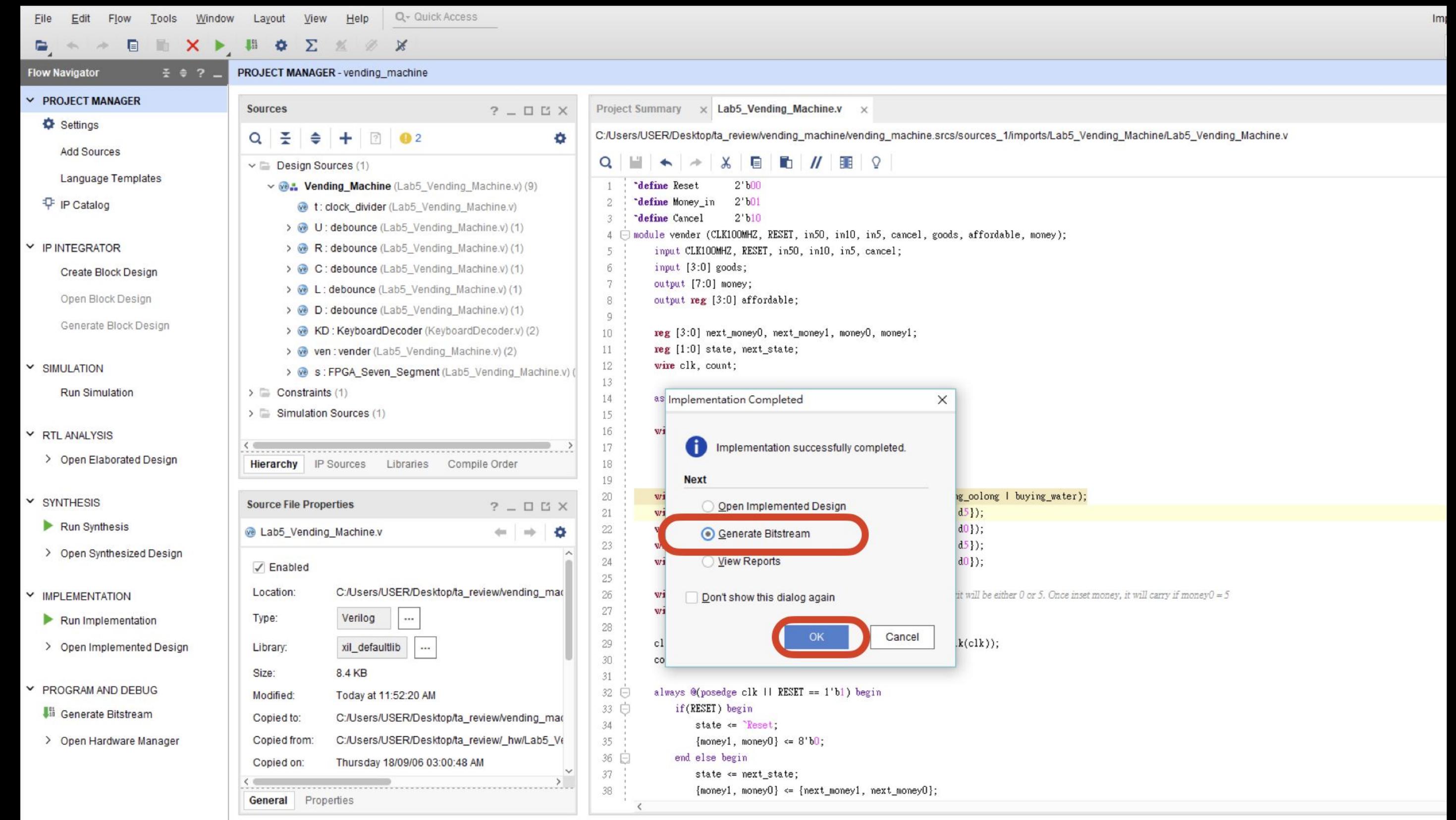
Generate Bit Stream (6/7)

- When Run Implementation

is completed, select

Generate Bitstream and

hit **OK**.



Generate Bit Stream (7/7)

- Or you can just click

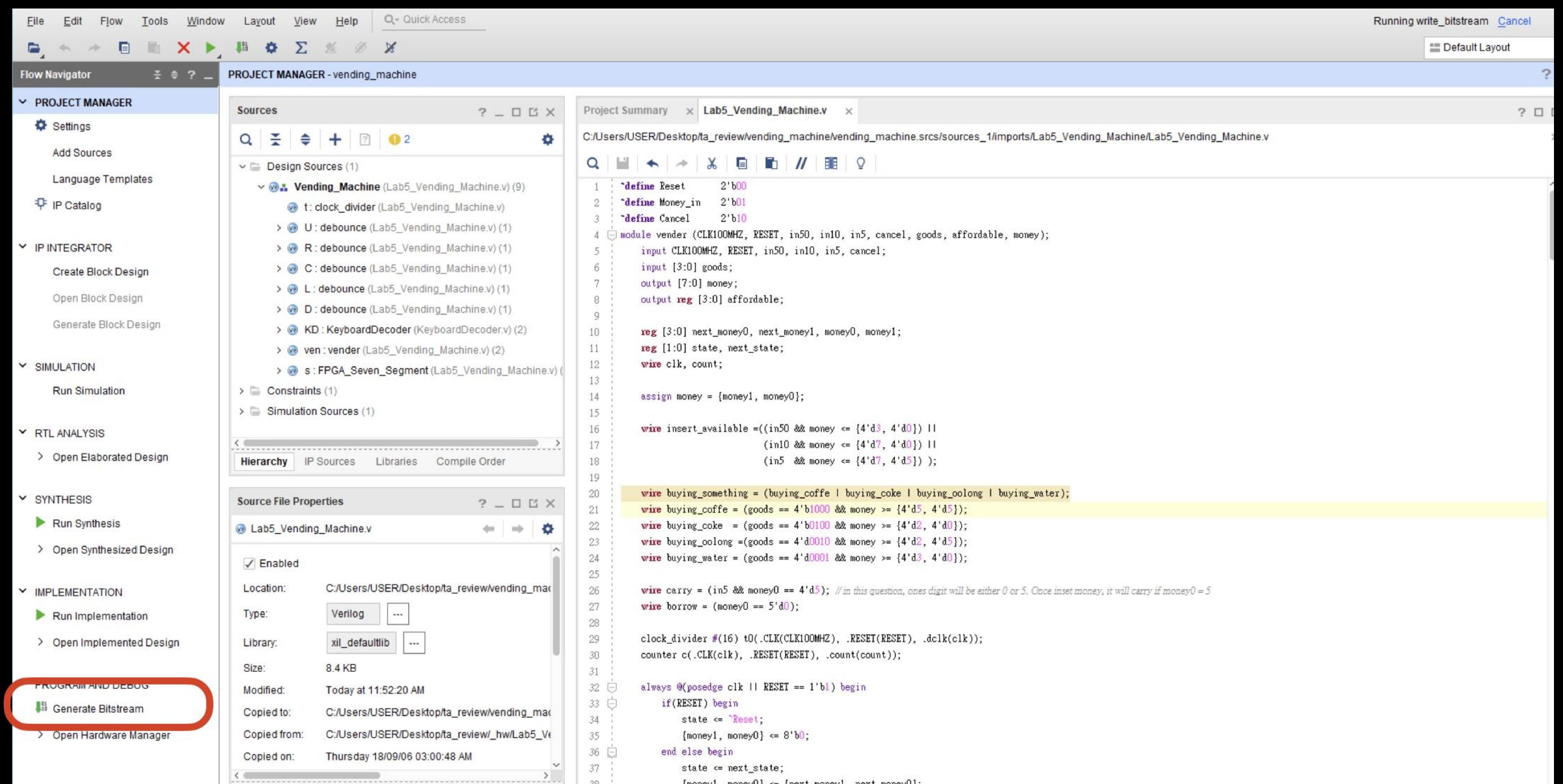
Generate Bitstream in the

very beginning, Vivado will

automatically run Synthesis

and Implementation for

you.



The screenshot shows the Vivado IDE interface. On the left, the Project Manager pane displays a tree structure with sections like PROJECT MANAGER, IP INTEGRATOR, SIMULATION, RTL ANALYSIS, SYNTHESIS, IMPLEMENTATION, and PROGRAM AND DEBUG. In the PROGRAM AND DEBUG section, the 'Generate Bitstream' button is highlighted with a red oval. The main workspace shows the source code for 'Lab5_Vending_Machine.v'. The code defines a module 'vender' with various inputs (clock, reset, money_in, cancel, goods, affordable, money) and outputs (goods, money). It includes logic for inserting money and calculating the next state. The code is annotated with comments explaining its functionality.

```
define Reset 2'b00
define Money_in 2'b01
define Cancel 2'b10
module vender (CLK100MHZ, RESET, in50, in10, in5, cancel, goods, affordable, money);
    input CLK100MHZ, RESET, in50, in10, in5, cancel;
    input [3:0] goods;
    output [7:0] money;
    output reg [3:0] affordable;
    reg [3:0] next_money0, next_money1, money0, money1;
    reg [1:0] state, next_state;
    wire clk, count;
    assign money = {money1, money0};
    wire insert_available =((in50 && money <= {4'd3, 4'd0}) ||
                           (in10 && money <= {4'd7, 4'd0}) ||
                           (in5 && money <= {4'd7, 4'd5}));
    wire buying_something = (buying_coffee | buying_coke | buying_pelong | buying_water);
    wire buying_coffee = (goods == 4'b1000 && money >= {4'd5, 4'd5});
    wire buying_coke = (goods == 4'b0100 && money >= {4'd2, 4'd0});
    wire buying_pelong = (goods == 4'd0010 && money >= {4'd2, 4'd5});
    wire buying_water = (goods == 4'd0001 && money >= {4'd3, 4'd0});
    wire carry = (in5 && money0 == 4'd5); // in this question, ones digit will be either 0 or 5. Once inset money, it will carry if money0 = 5
    wire borrow = (money0 == 5'd0);
    clock_divider #(16) t0(.CLK(CLK100MHZ), .RESET(RESET), .dclk(clk));
    counter c(.CLK(clk), .RESET(RESET), .count(count));
    always @(posedge clk || RESET == 1'b1) begin
        if(RESET) begin
            state <= `Reset;
            {money1, money0} <= 8'b0;
        end else begin
            state <= next_state;
            {money1, money0} <= {next_money1, next_money0};
        end
    end
endmodule
```

Agenda

Overview

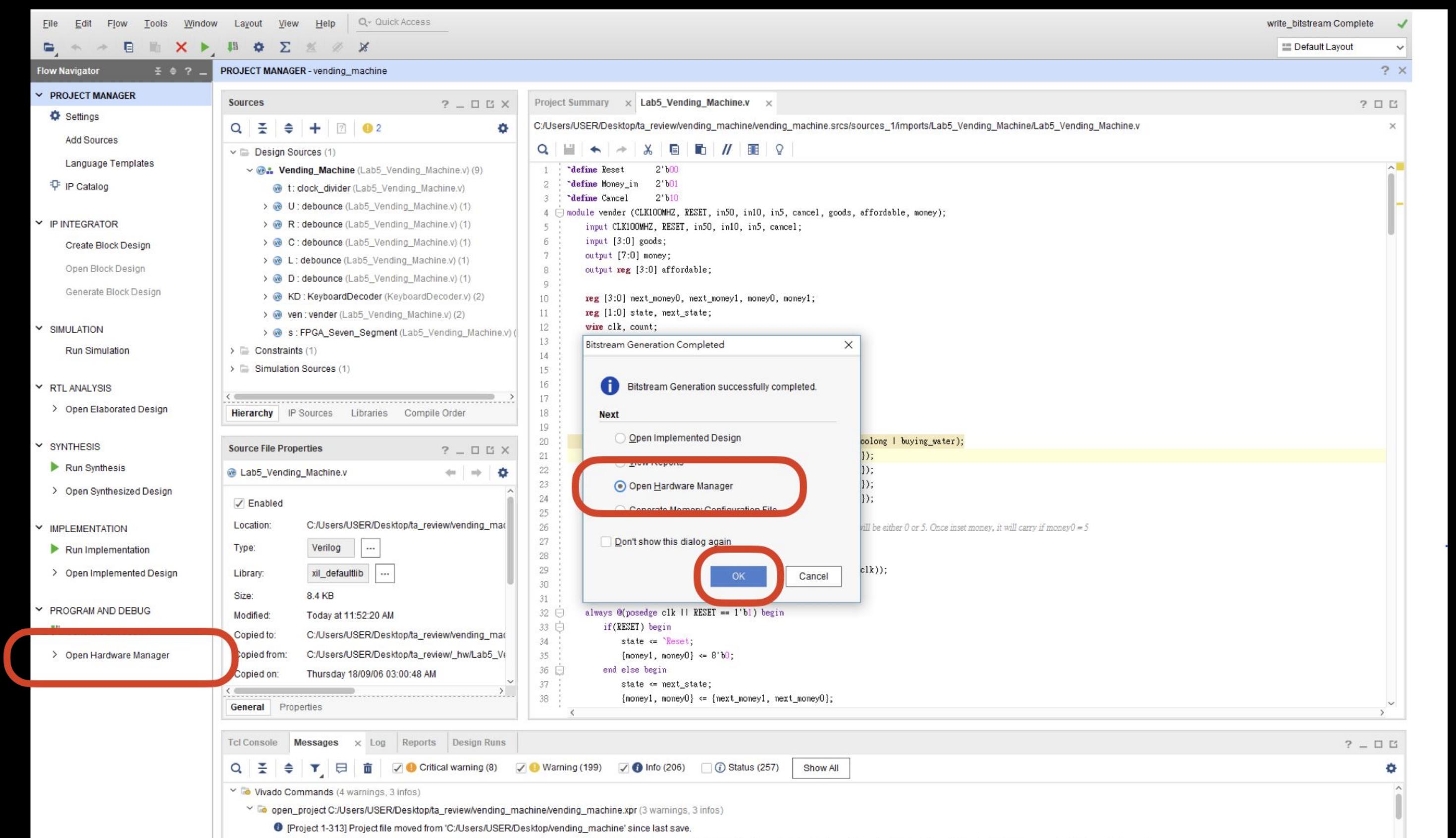
Preparation

Generate Bit Stream

Design Verification on FPGA

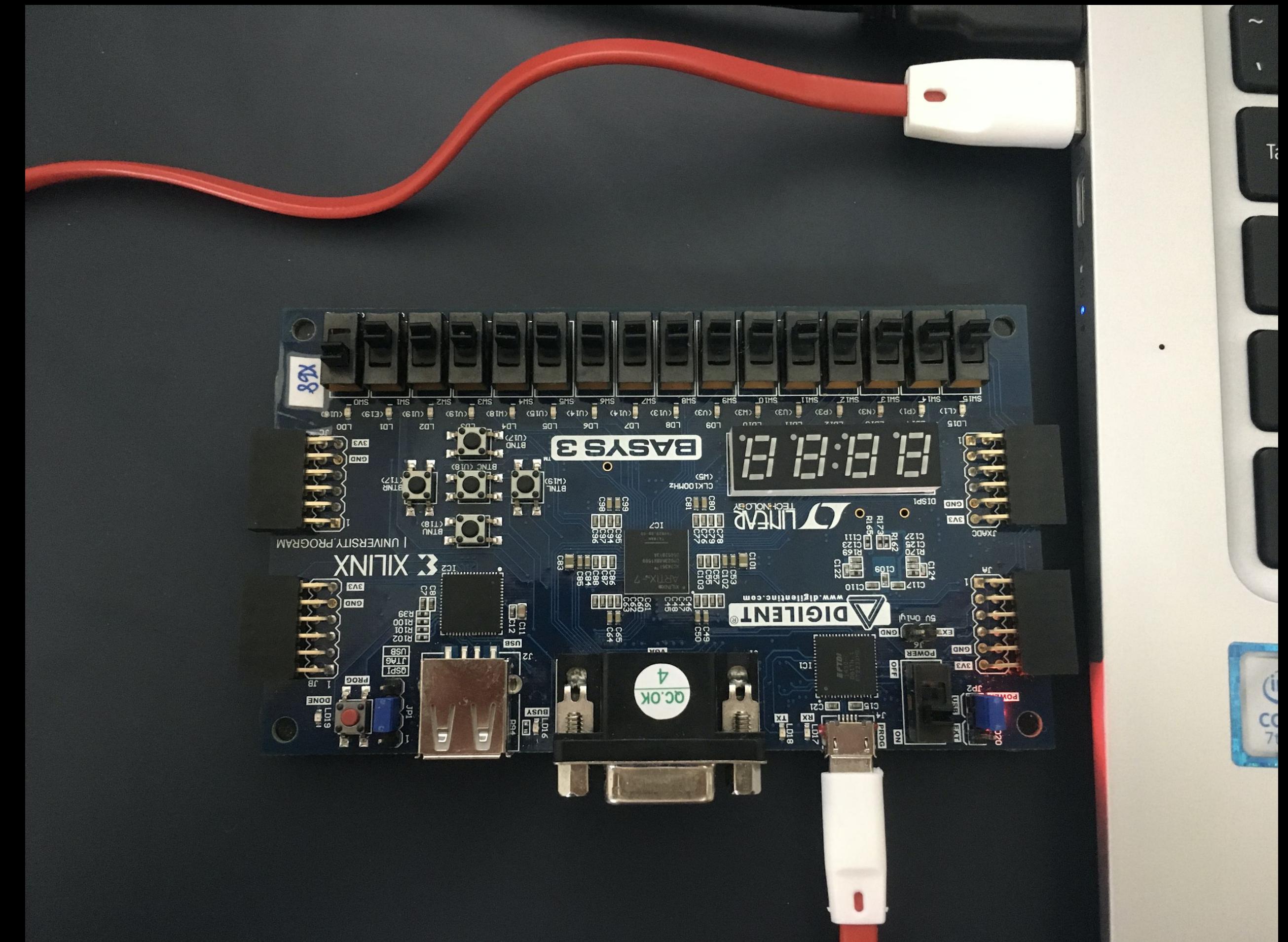
Design Verification on FPGA (1/11)

- After the bitstream generation, select *Open Hardware Manager* and hit **OK**.



Design Verification on FPGA (2/11)

- Connect the FPGA board
to your computer



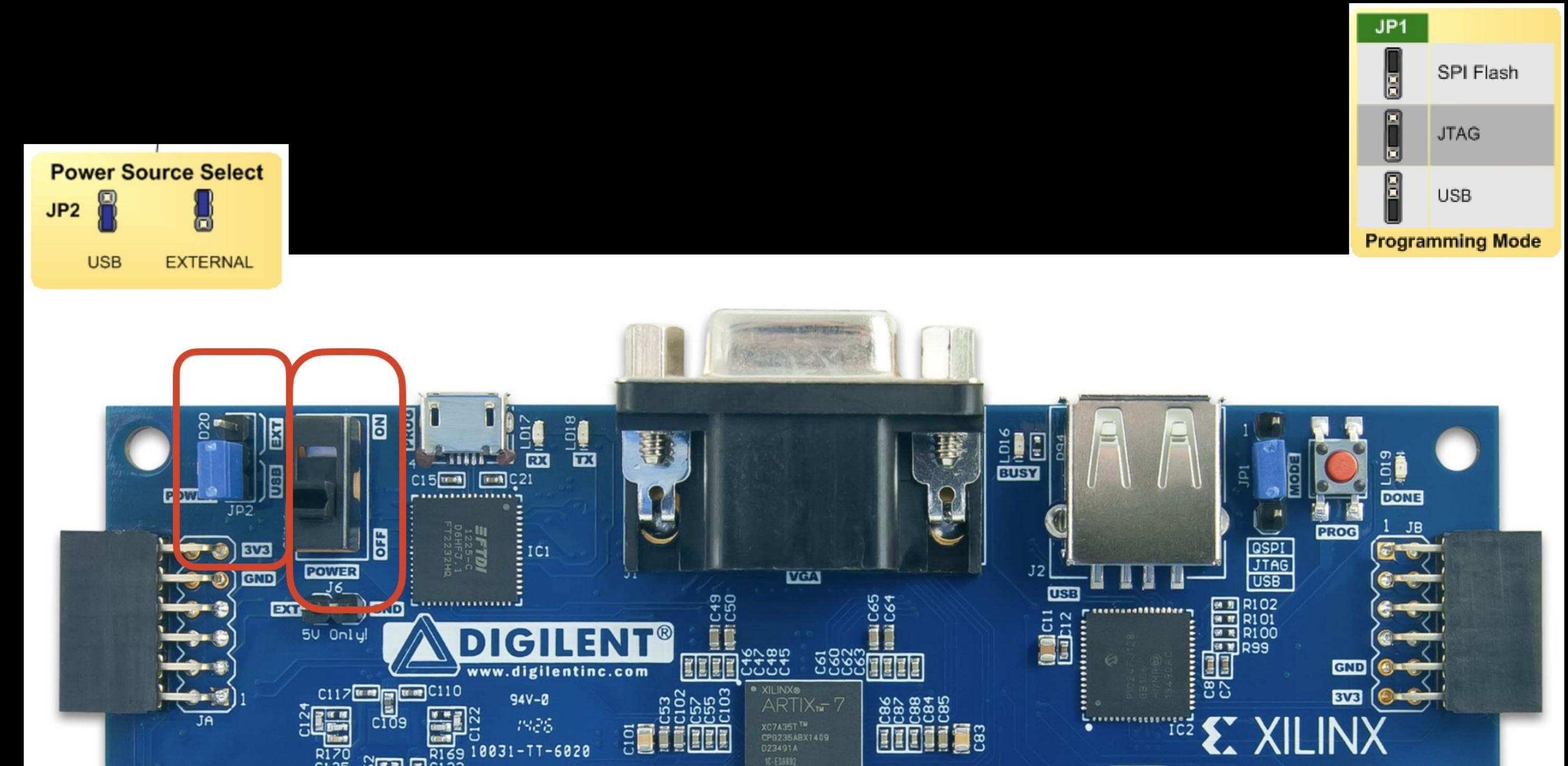
Design Verification on FPGA (3/11)

- Make sure JP1 has JTAG

mode selected and JP2

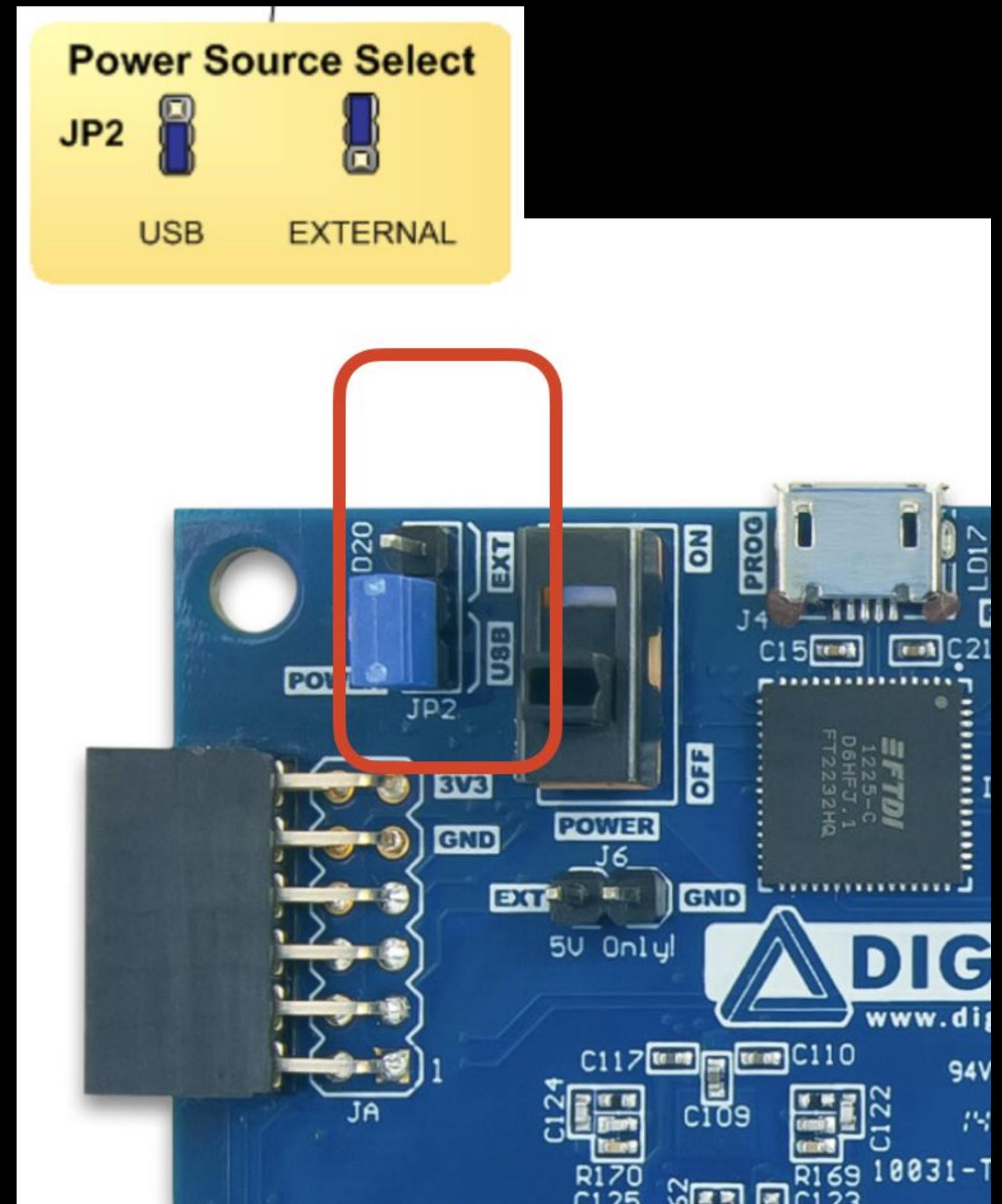
has USB mode selected.

- Switch the power switch to **ON**.



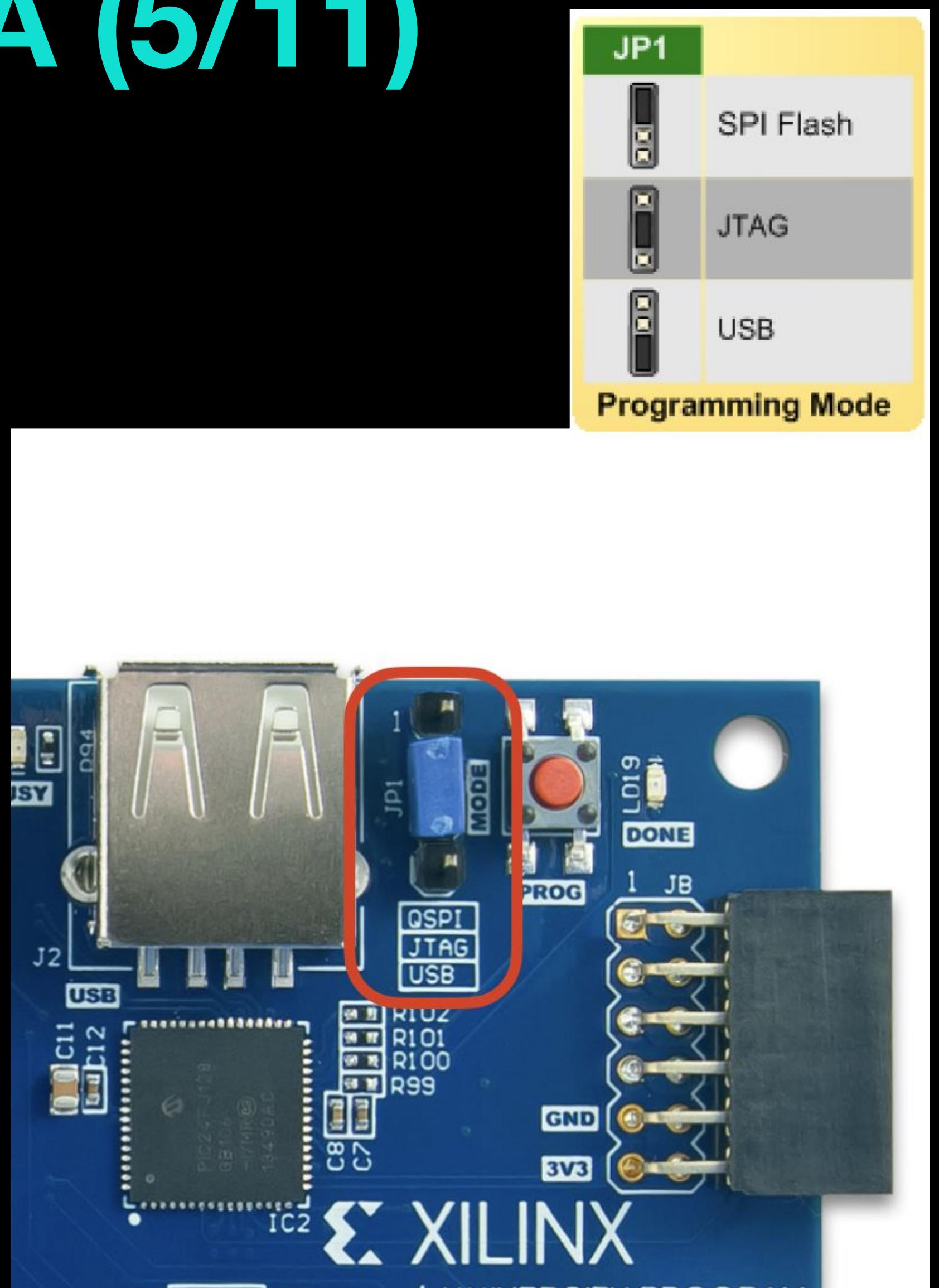
Design Verification on FPGA (4/11)

- JP2: power source selection
 - USB: the FPGA board is powered by the microUSB port (recommended)
 - EXTERNAL: the FPGA board is powered by 5V power supply (e.g. power bank)



Design Verification on FPGA (5/11)

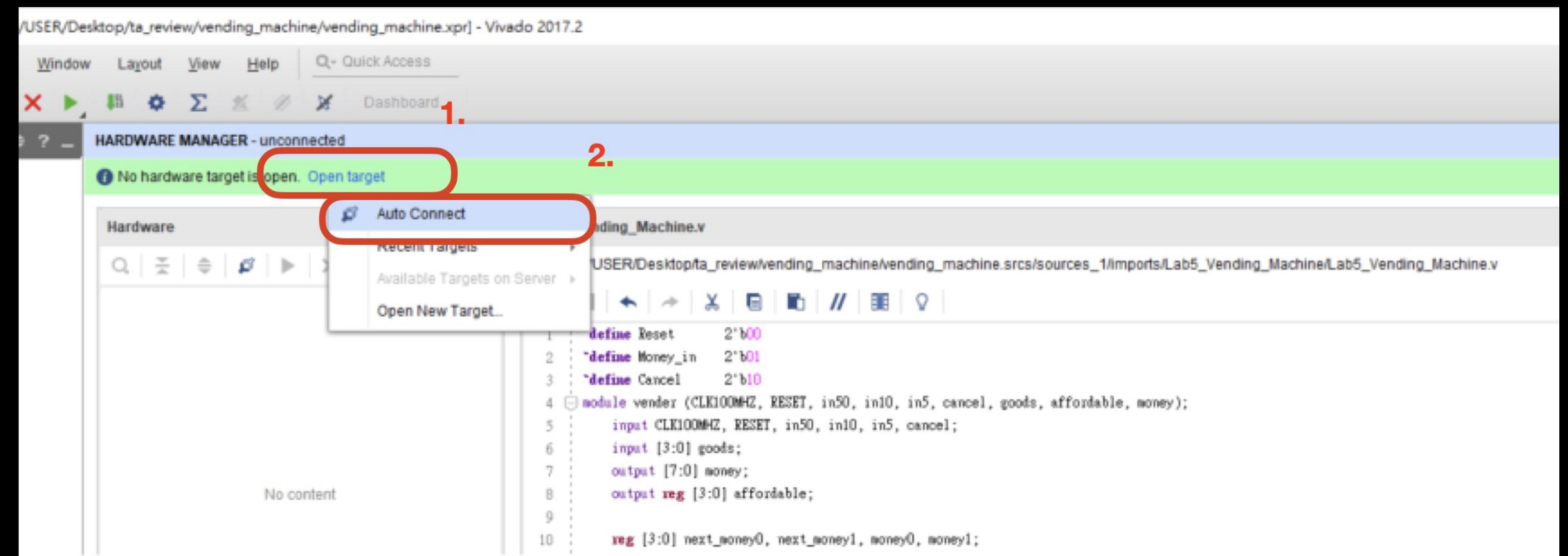
- JP1: Programming mode selection
- Receive Bitstream from:
 - Flash (on board, non-volatile)
 - JTAG (on board, volatile, recommend)
 - USB Drive



Design Verification on FPGA (6/11)

- Connect to our FPGA board in Vivado by hitting

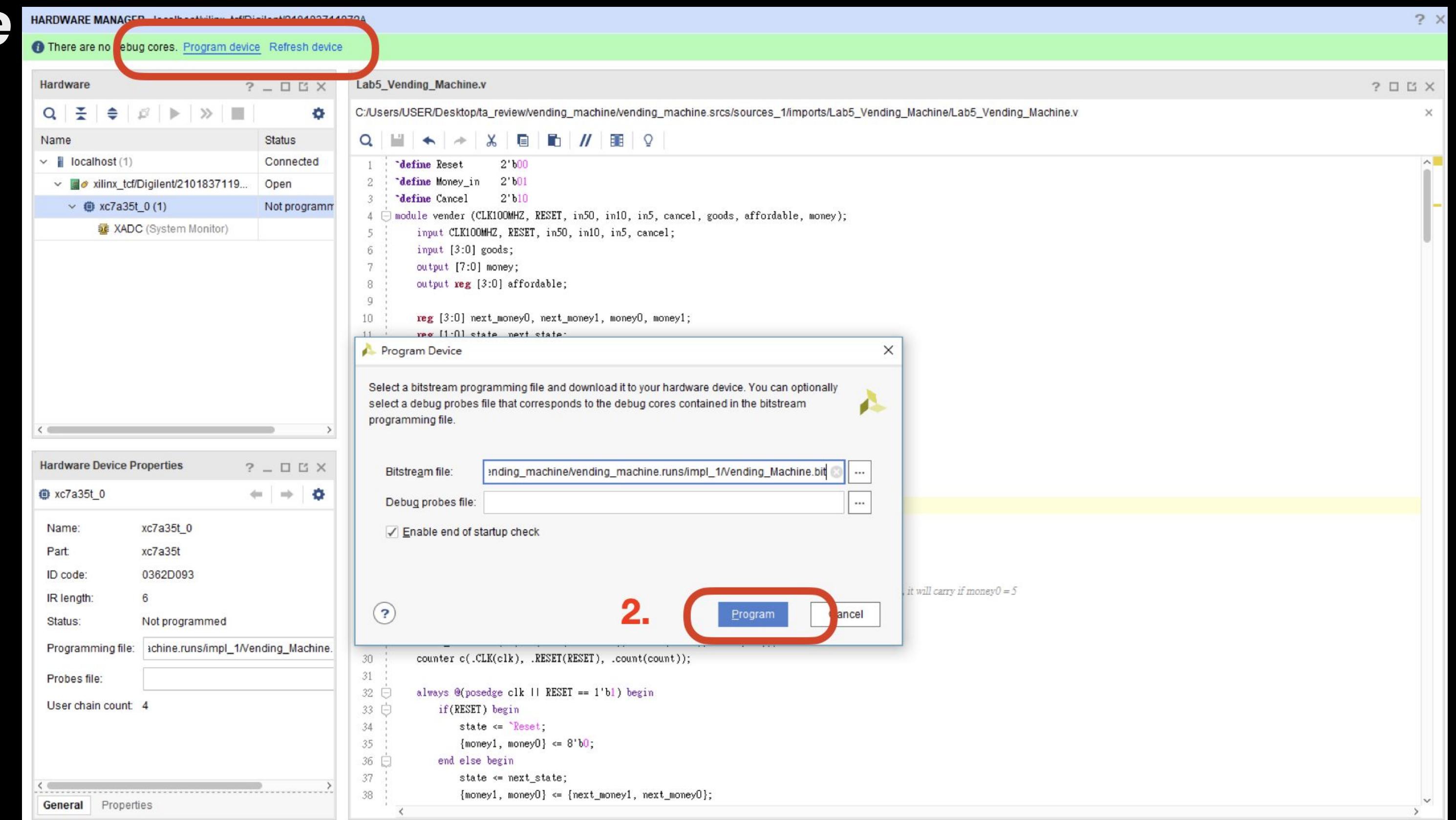
Open target and then
Auto Connect.



Design Verification on FPGA (7/11)

- Finally, hit **Program device**

and hit **Program**.

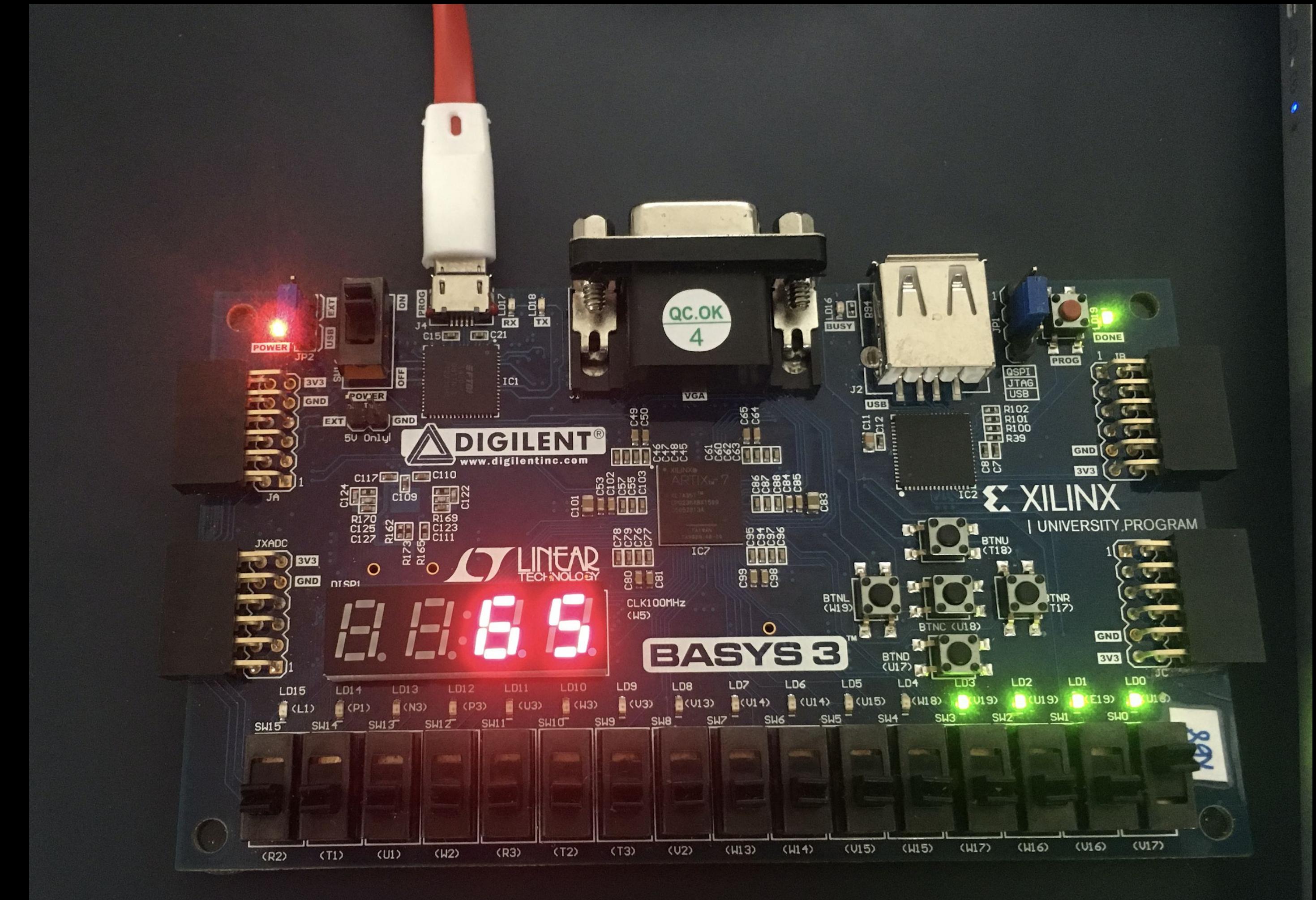


Design Verification on FPGA (8/11)

- Done, your verilog design

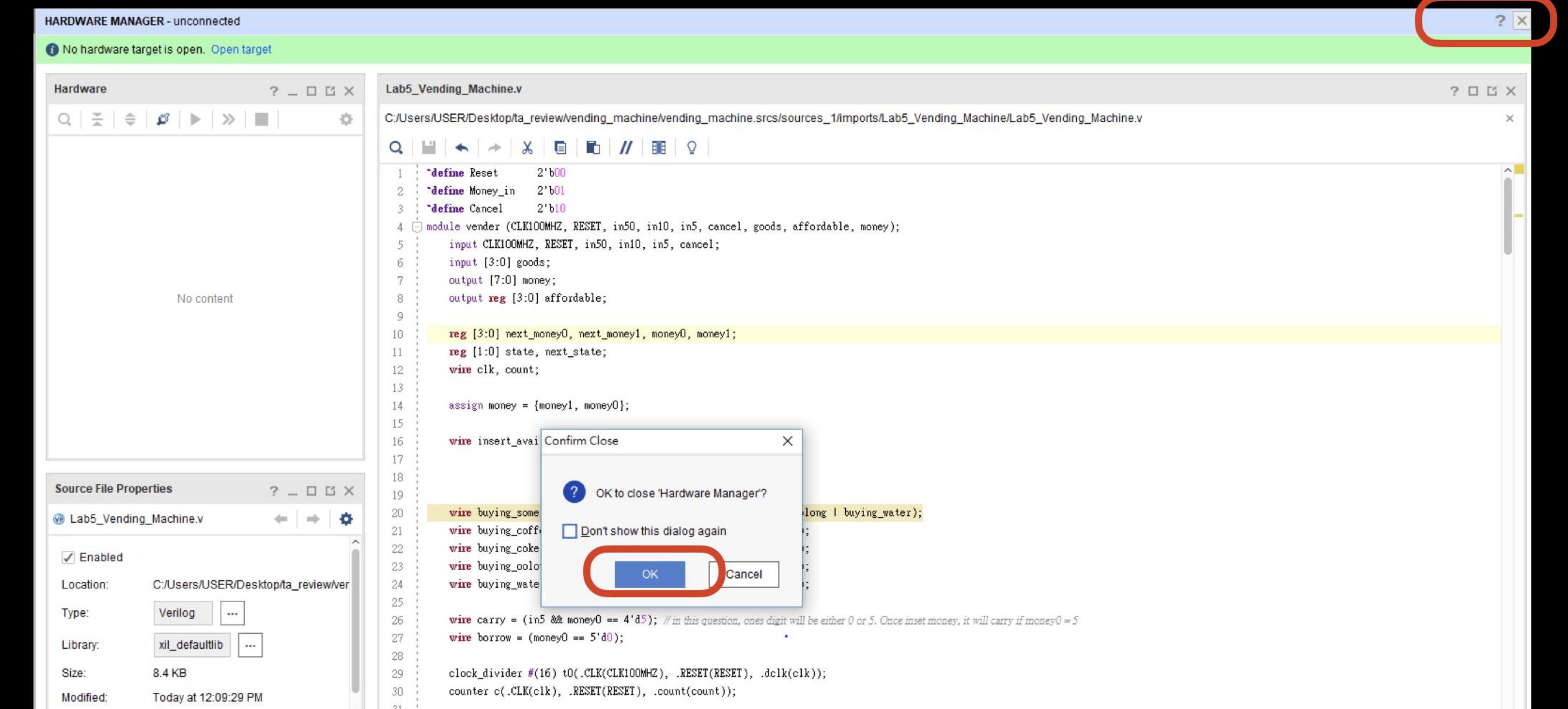
is now programmed into
your FPGA board.

- Verify that your design is
correct.



Design Verification on FPGA (9/11)

- To go back to the source panel, close the Hardware Manager.



Design Verification on FPGA (10/11)

- Done

The screenshot shows the Xilinx Vivado IDE interface. On the left, the Flow Navigator pane is visible with sections for PROJECT MANAGER, IP INTEGRATOR, SIMULATION, RTL ANALYSIS, SYNTHESIS, and IMPLEMENTATION. The PROJECT MANAGER section is expanded, showing options like Settings, Add Sources, Language Templates, and IP Catalog. A red circle highlights the Sources tab in the PROJECT MANAGER pane, which displays a list of Design Sources (1) including Vending_Machine (Lab5_Vending_Machine.v), t: clock_divider, U: debounce, R: debounce, C: debounce, L: debounce, D: debounce, KD: KeyboardDecoder, ven: vender, s: FPGA_Seven_Segment, Constraints (1), and Simulation Sources (1). Below this is a Properties panel for Lab5_Vending_Machine.v with checkboxes for Enabled and Location set to C:/Users/USER/Desktop/ta_review/vending_machine. To the right of the Sources tab is a code editor window titled Lab5_Vending_Machine.v, showing Verilog code for a vending machine design. The code defines modules for vendor, keyboard decoder, and seven-segment display, along with logic for inserting money and buying items.

```
define Reset 2'b00
define Money_in 2'b01
define Cancel 2'b10
module vendor (CLK100MHZ, RESET, in50, in10, in5, cancel, goods, affordable, money);
    input CLK100MHZ, RESET, in50, in10, in5, cancel;
    input [3:0] goods;
    output [7:0] money;
    output reg [3:0] affordable;
    reg [3:0] next_money0, next_money1, money0, money1;
    reg [1:0] state, next_state;
    wire clk, count;
    assign money = {money1, money0};
    wire insert_available =((in50 && money <= {4'd3, 4'd0}) ||
                           (in10 && money <= {4'd7, 4'd0}) ||
                           (in5 && money <= {4'd7, 4'd5}) );
    wire buying_something = (buying_coffe | buying_coke | buying_oolong | buying_water);
    wire buying_coffe = (goods == 4'b1000 && money >= {4'd5, 4'd5});
    wire buying_coke = (goods == 4'b0100 && money >= {4'd2, 4'd0});
    wire buying_oolong =(goods == 4'd0010 && money >= {4'd2, 4'd5});
    wire buying_water = (goods == 4'd0001 && money >= {4'd3, 4'd0});
    wire carry = (in5 && money0 == 4'd5);
```

Design Verification on FPGA (11/11)

- For additional information (e.g., configuration in to FPGA's flash

ROM), please refer to:

- <https://reference.digilentinc.com/basys3/refmanual>
- https://www.youtube.com/watch?v=6_GxksIqbcU

Q&A