PROJECT: BEE INVADERS

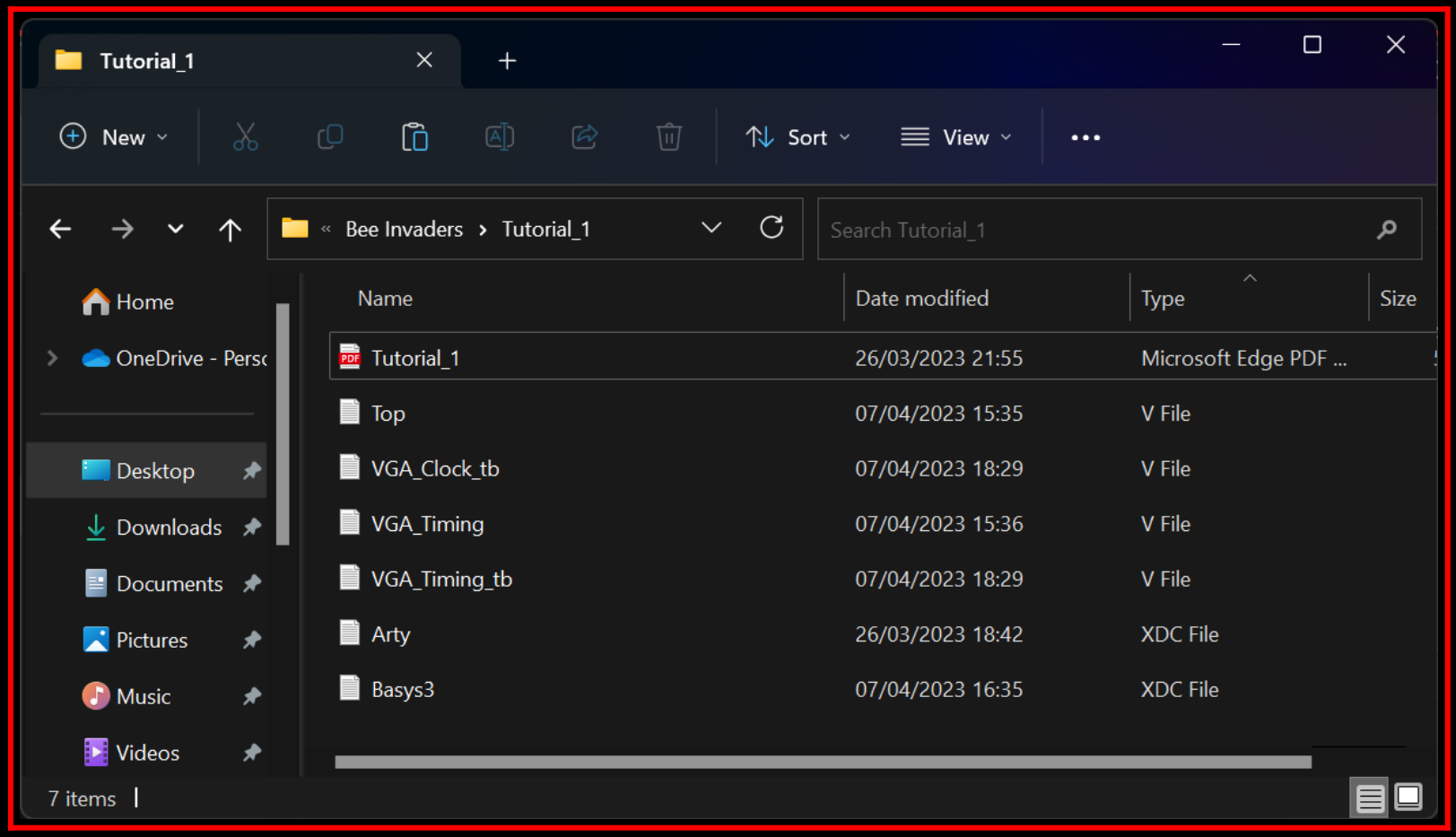
This Tutorial Is For The
Basys3 FPGA Board Or The Arty A7-35 FPGA Board With A VGA Pmod Connected
But Can Be Adapted To Other FPGA Boards
A Modern Version Of The Popular Arcade Game
Space Invaders

Tutorial 1 - Fill A VGA Screen With One Colour

CONTENTS

(A) INSTRUCTIONS ON HOW TO CREATE THE PROJECT IN VIVADO

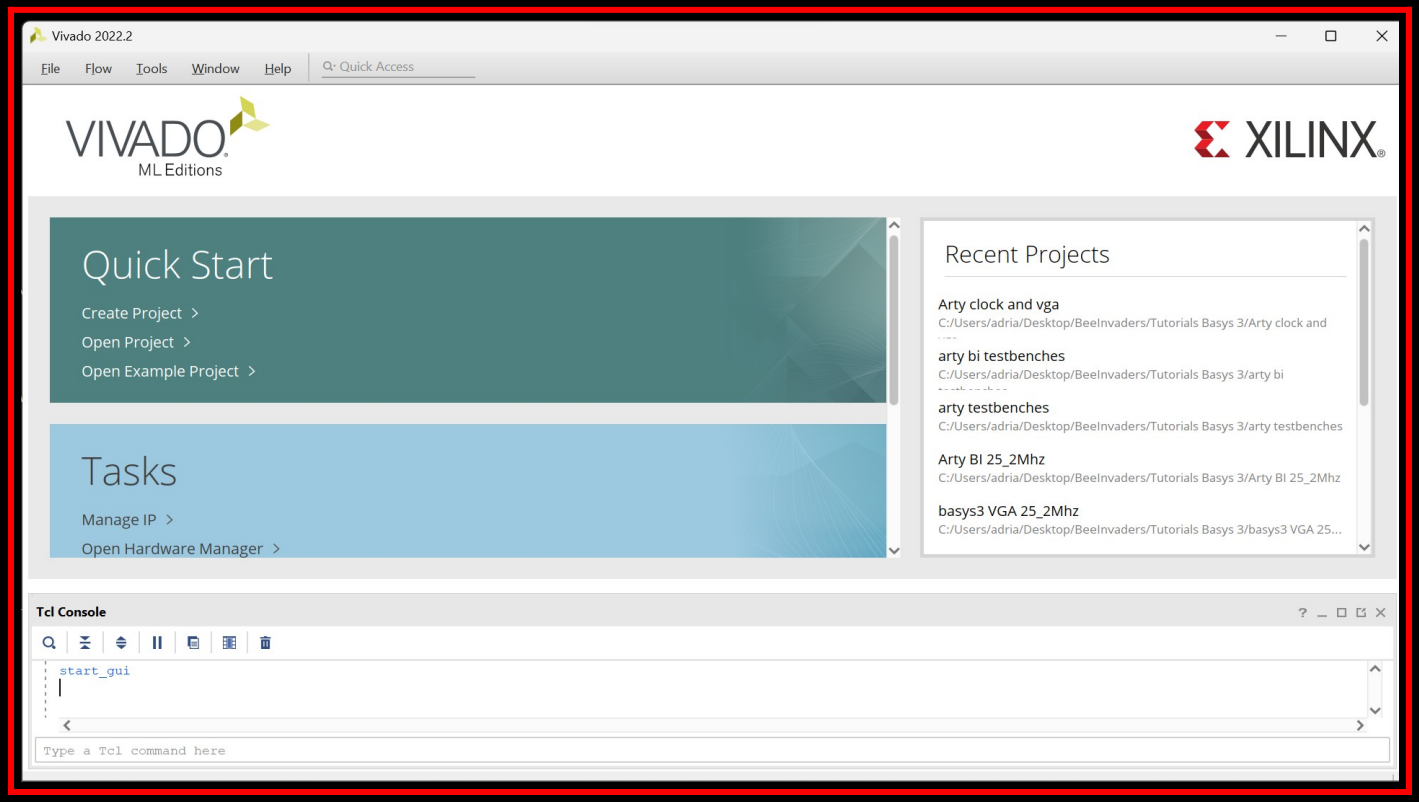
01 Create a folder on your Windows Desktop called "Bee Invaders". In this folder create a folder called "Tutorial_1". Copy the downloaded files to the "Tutorial_1" folder, as shown below



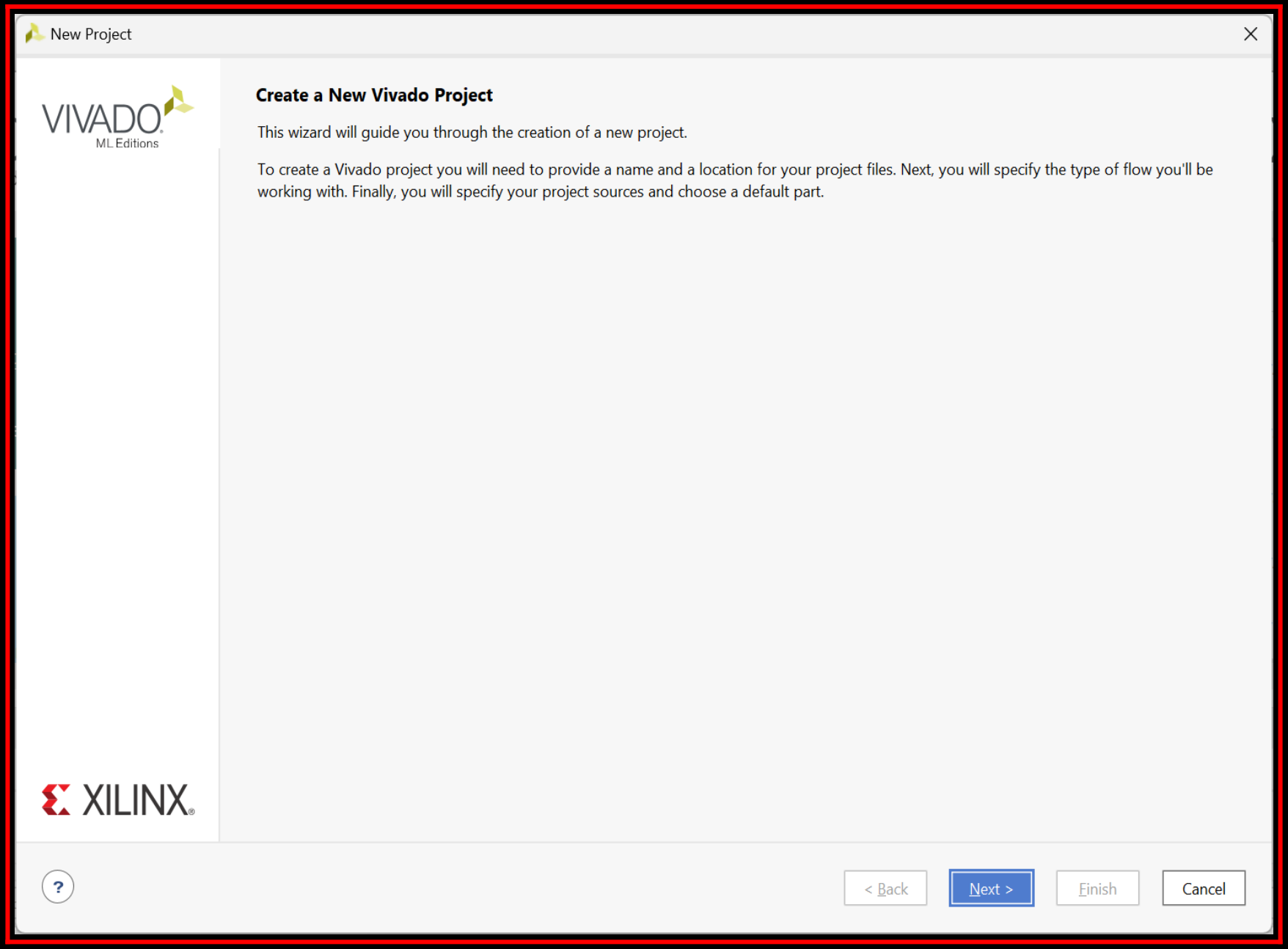
02

Download and install from the Xilinx website the free software Vivado ML Standard Edition 2022.2 Full Product Installation, or the most recent version. "Xilinx Unified Installer 2022.2: Windows Self Extracting Web Installer". Disable any Antivirus software during installation

Run Vivado and you will see a screen similar to this:

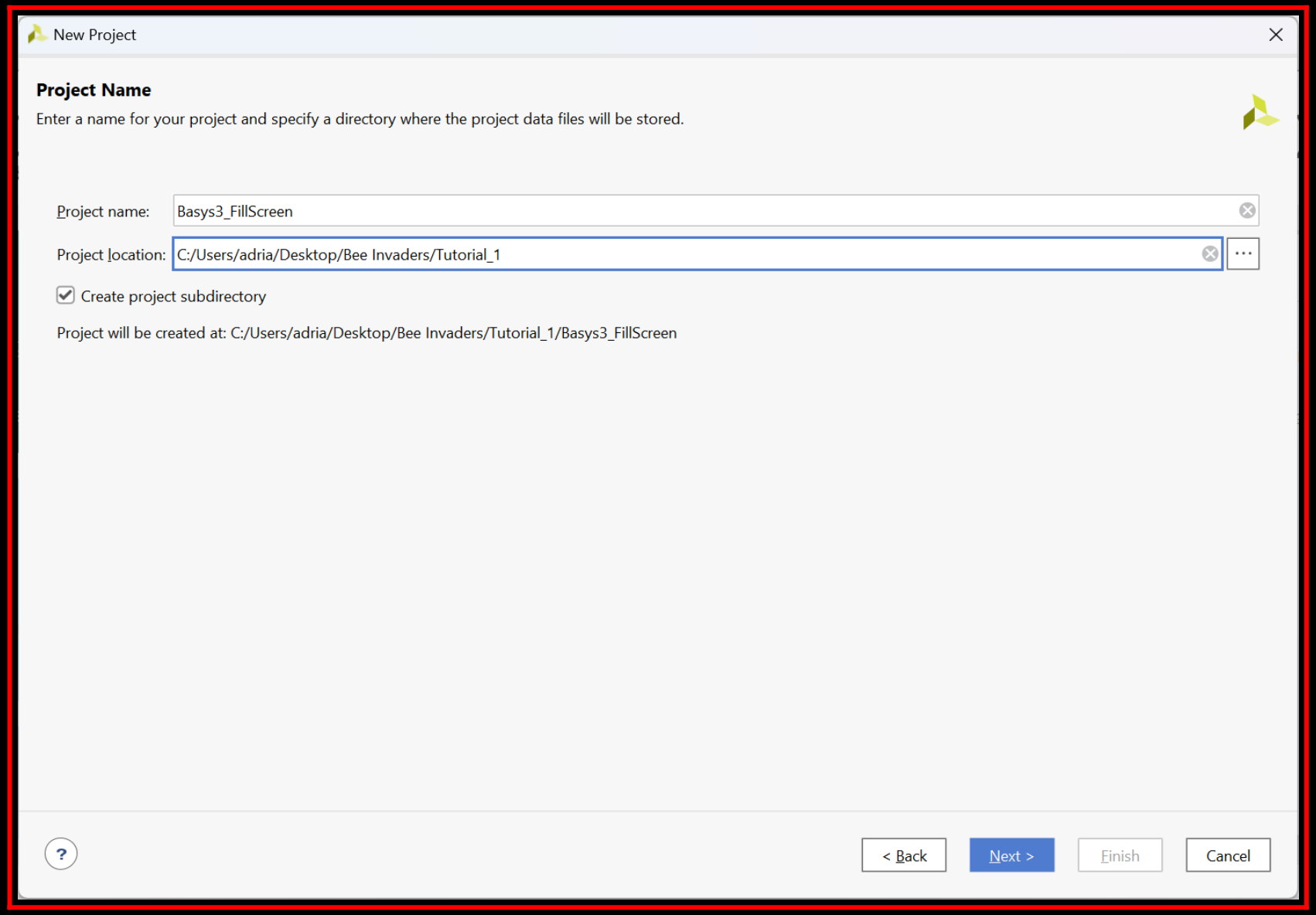


03 Click "Create Project" and when the below screen appears click "Next"

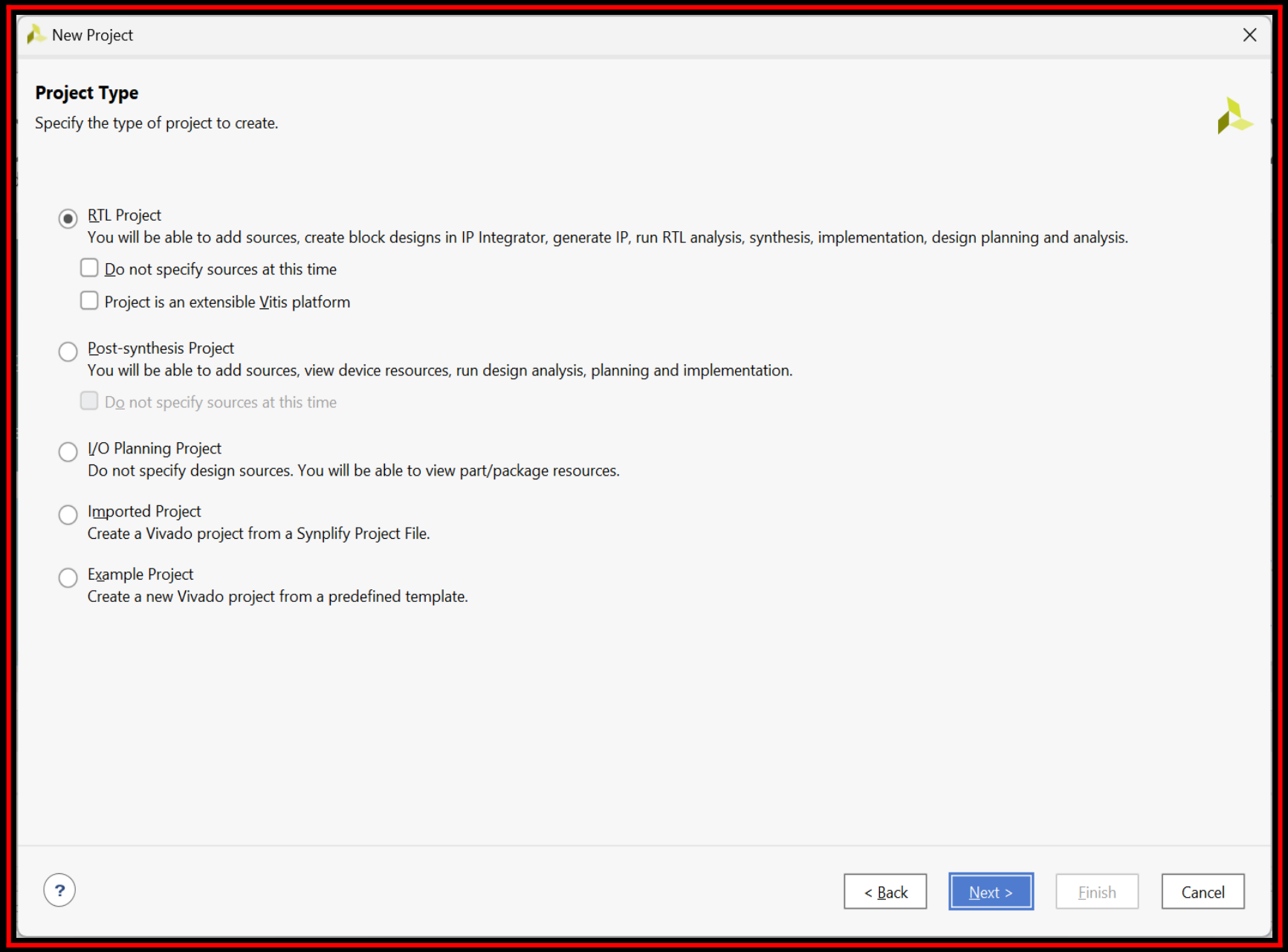


04

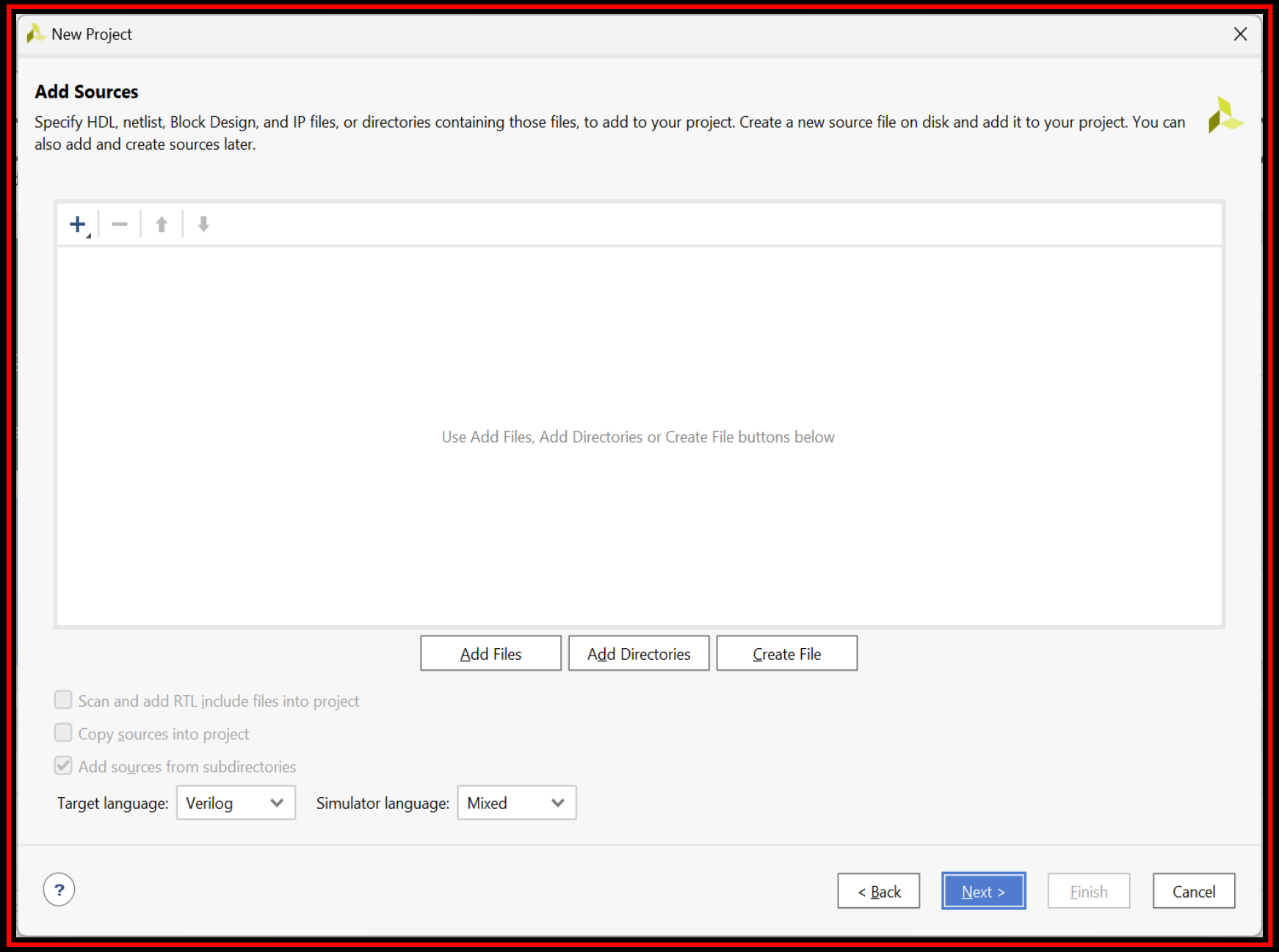
Enter "Basys3_FillScreen" (or "Arty_FillScreen" if you are using this board) in the "Project name" box and change the path for the "Project location" to the "Bee Invaders/Tutorial_1" folder you created in step 1. Tick the box entitled "Create project subdirectory" and click "Next"



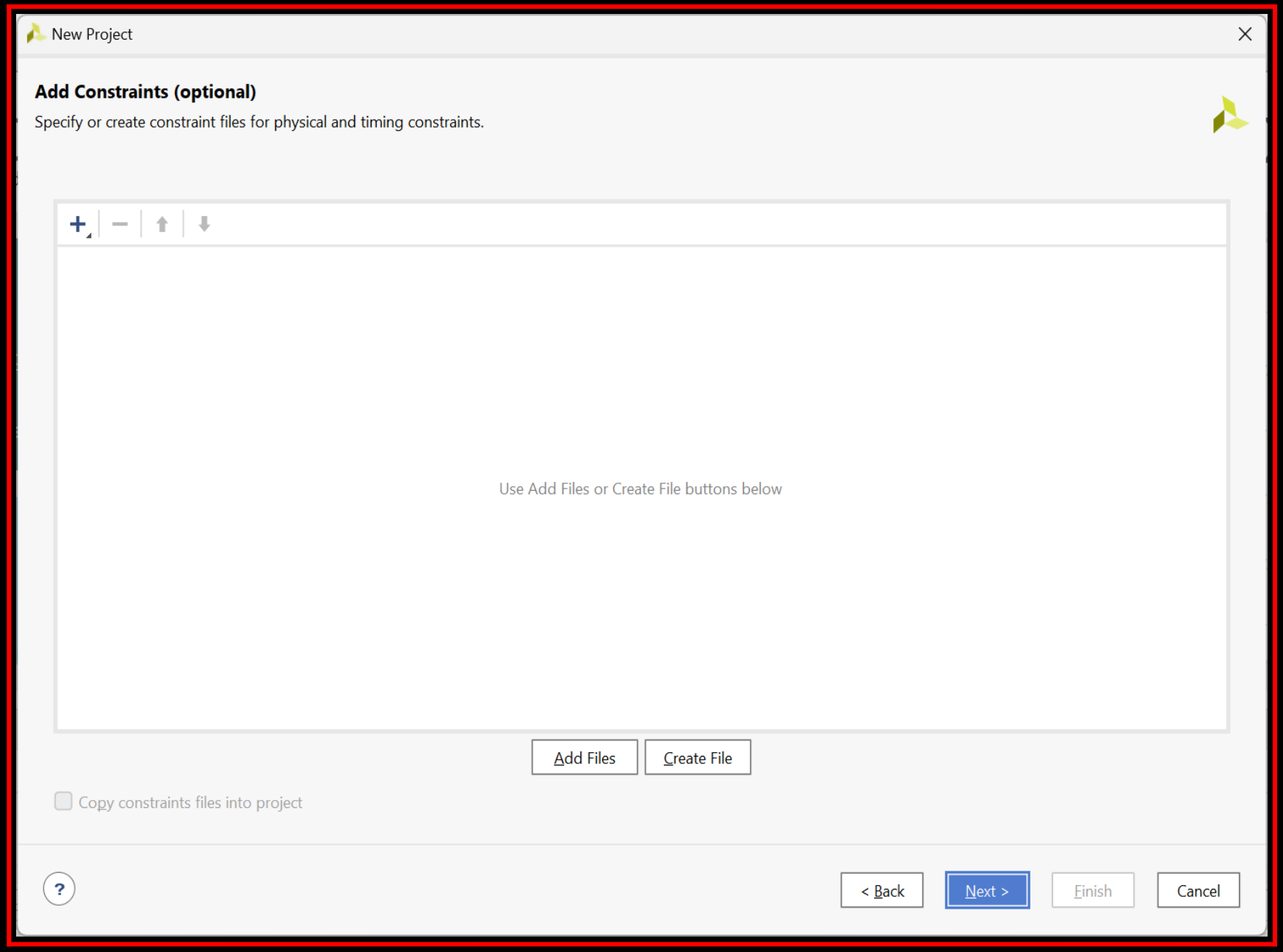
05 Select "RTL Project" and click "Next"



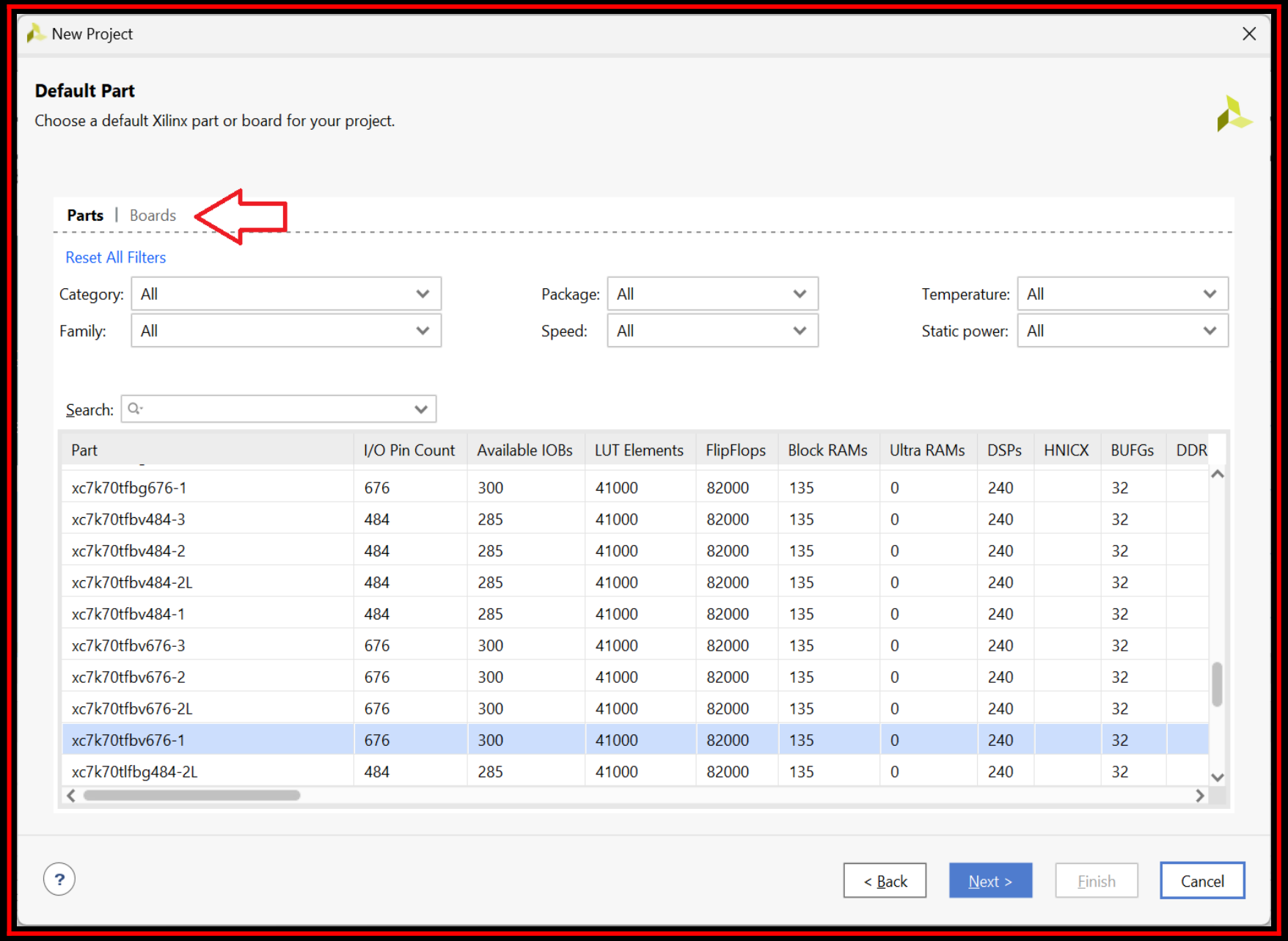
06 At the next screen make sure "Verilog" is selected for the "Target language" and click "Next"



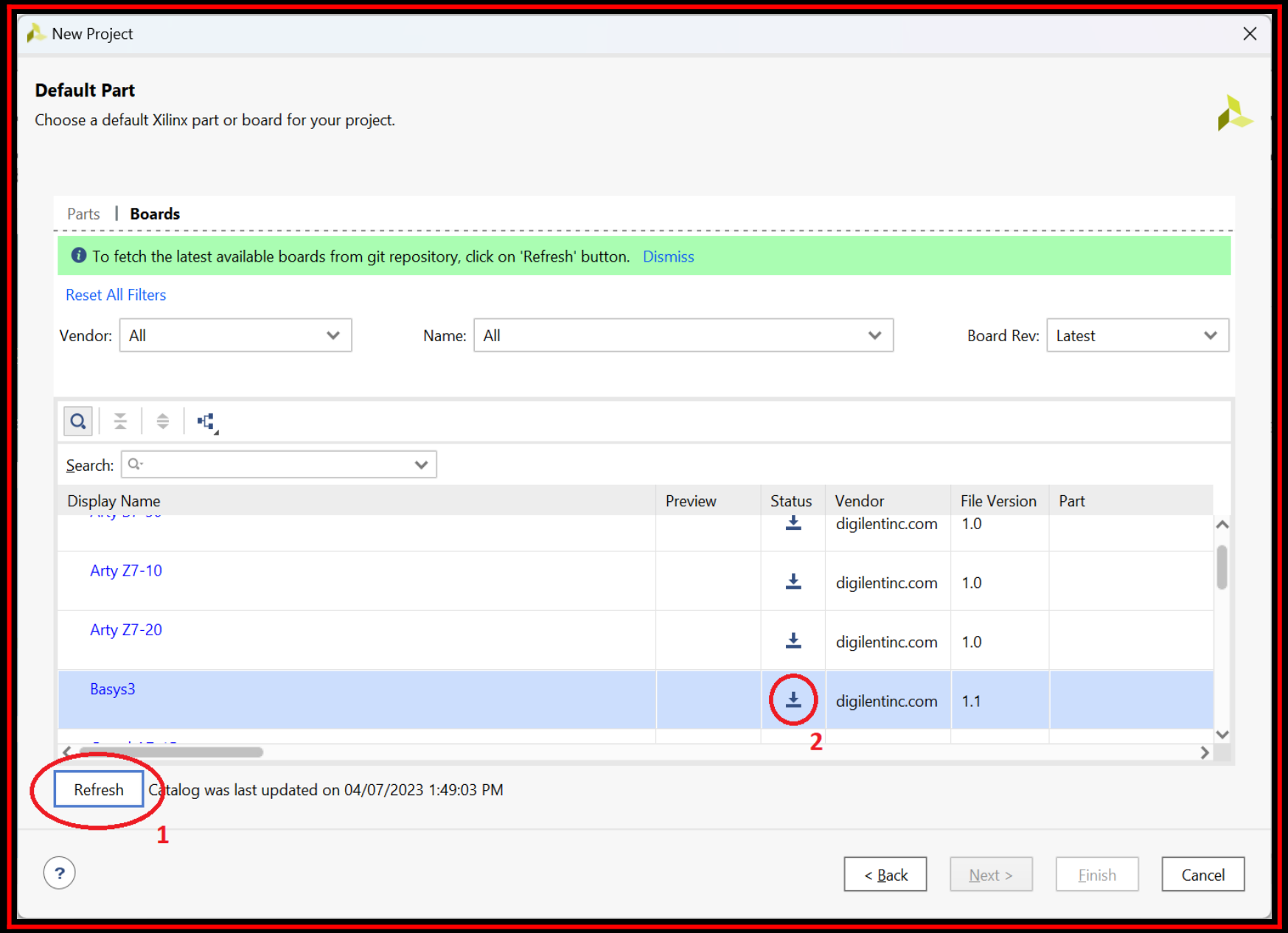
07 Click "Next" at the screen below



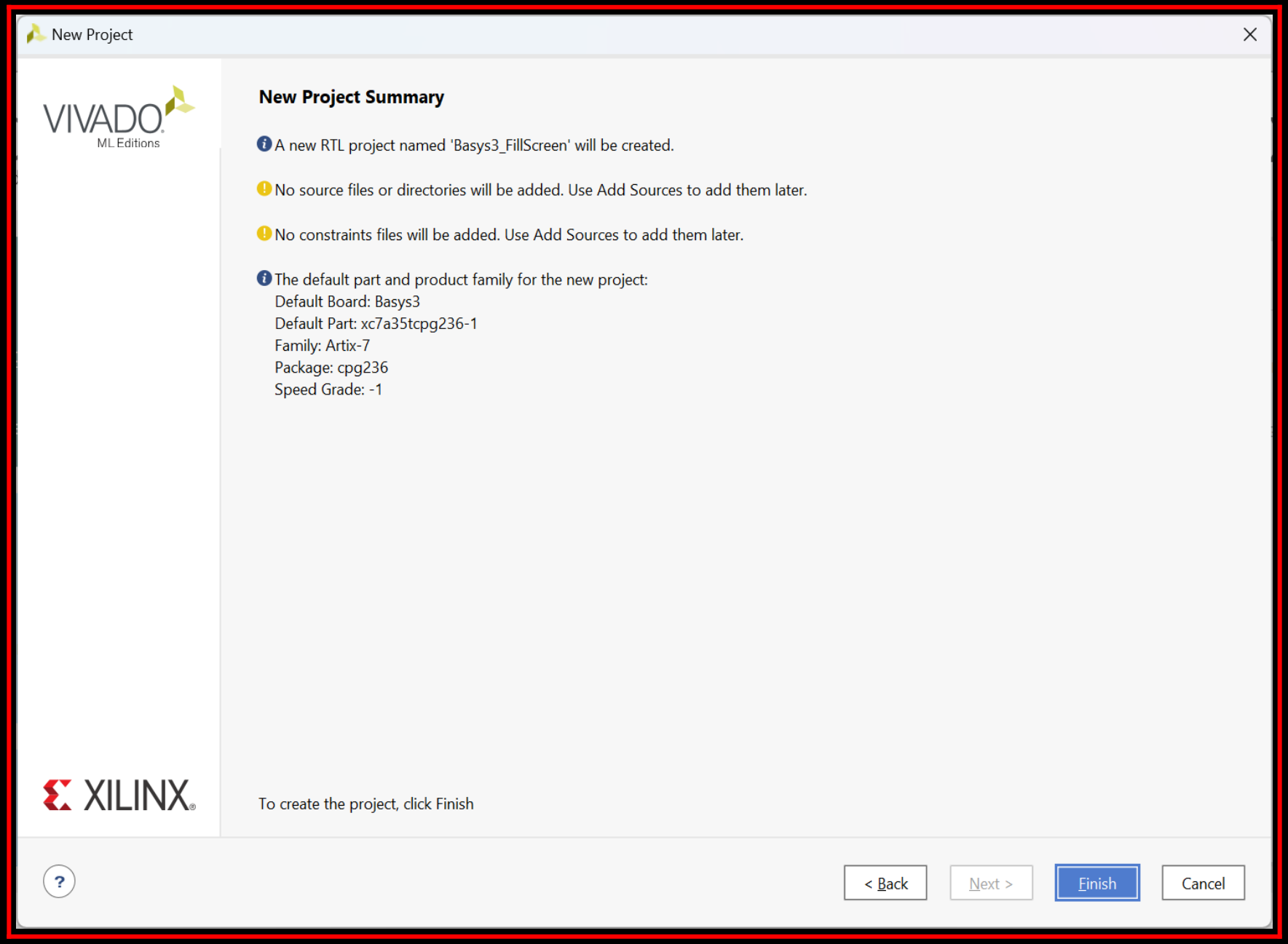
08 When the below screen appears click the "Boards" tab



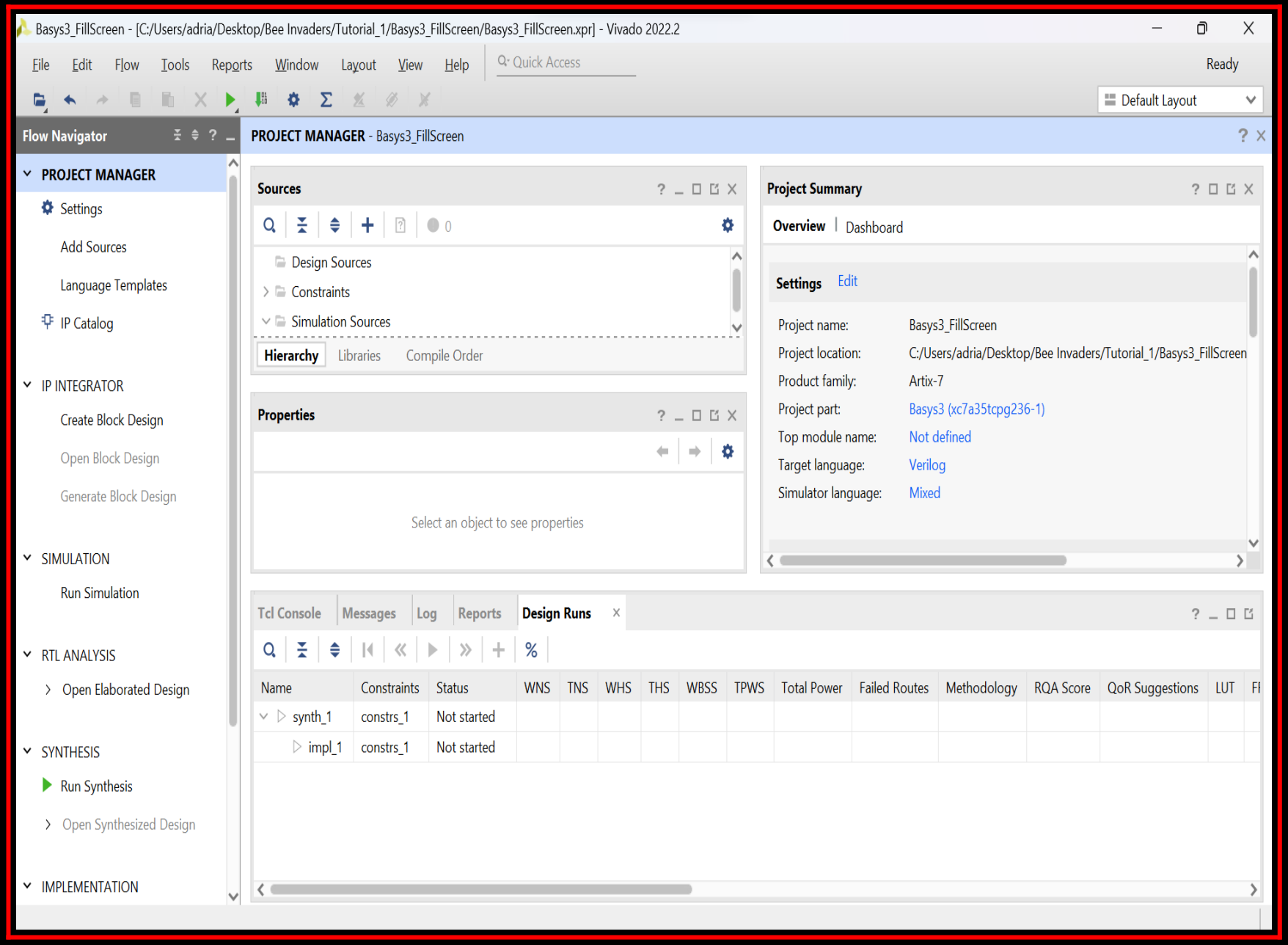
09 If the Basys3 board (or Arty A7-35 board) is not installed you will need click "Refresh" (1) and then install the board as shown below (2). Select the "Basys3" board (or "Arty A7-35") and click "Next"



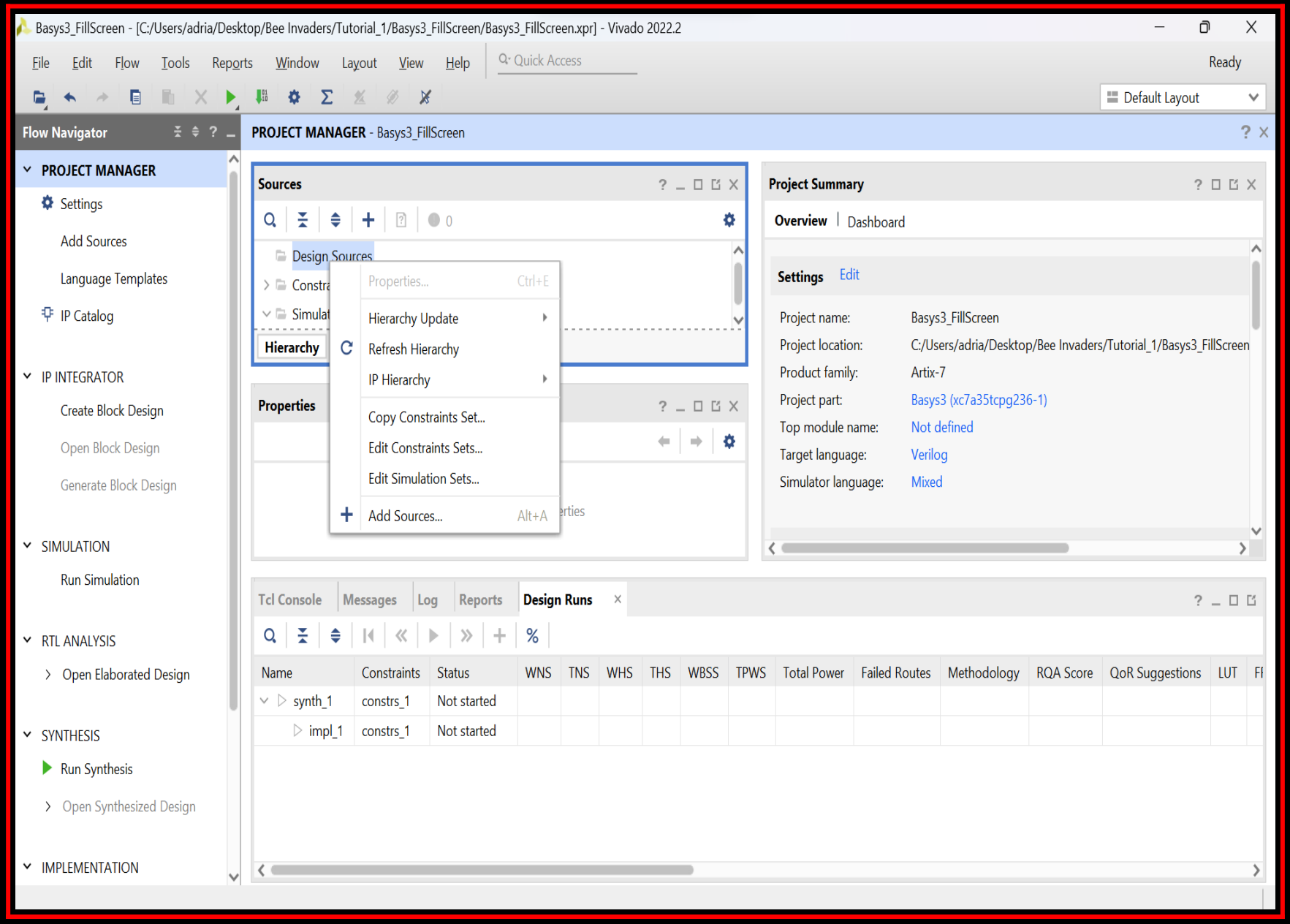
10 Click "Finish" at the screen below



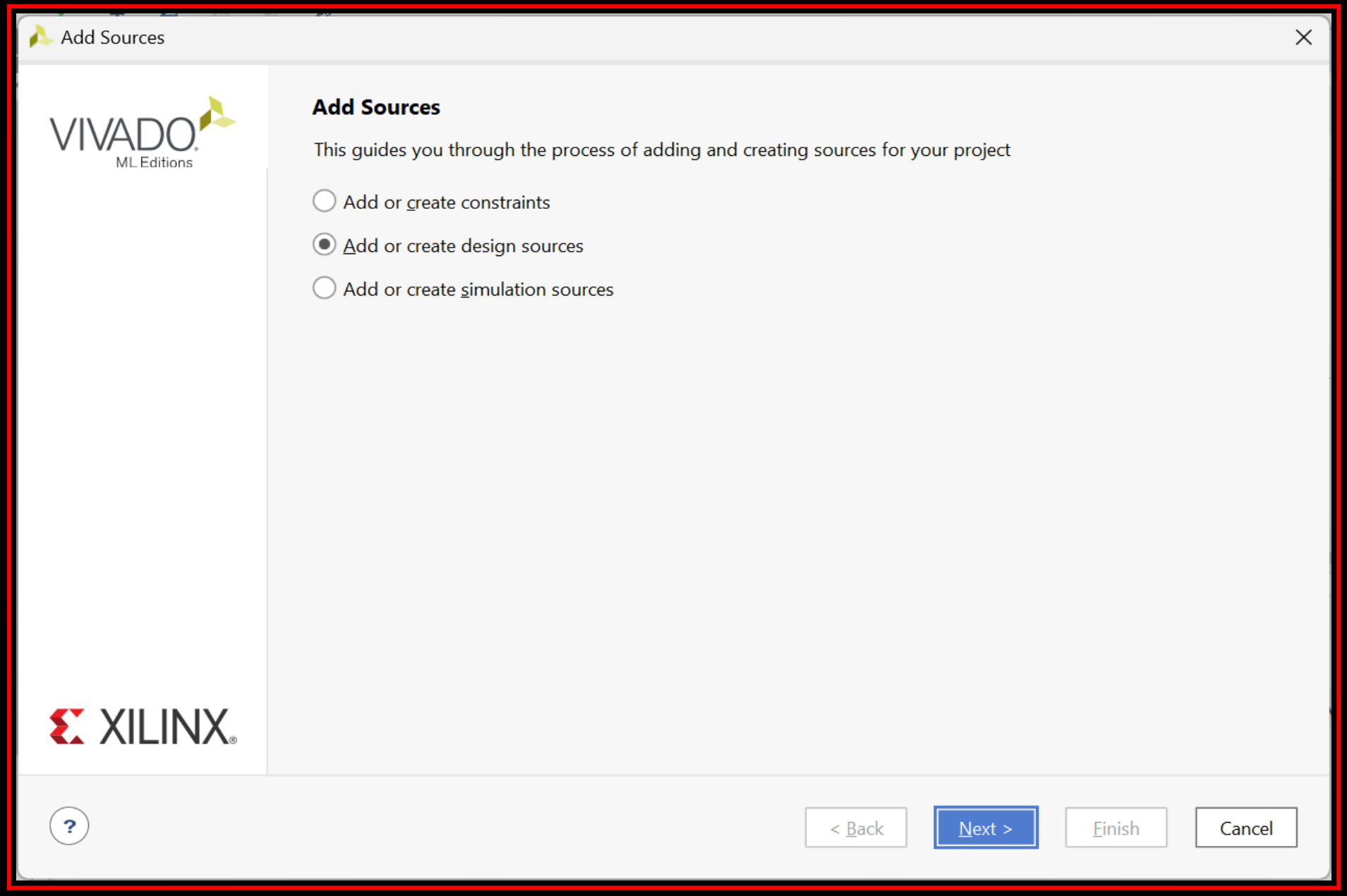
11 You should now see a screen as shown below



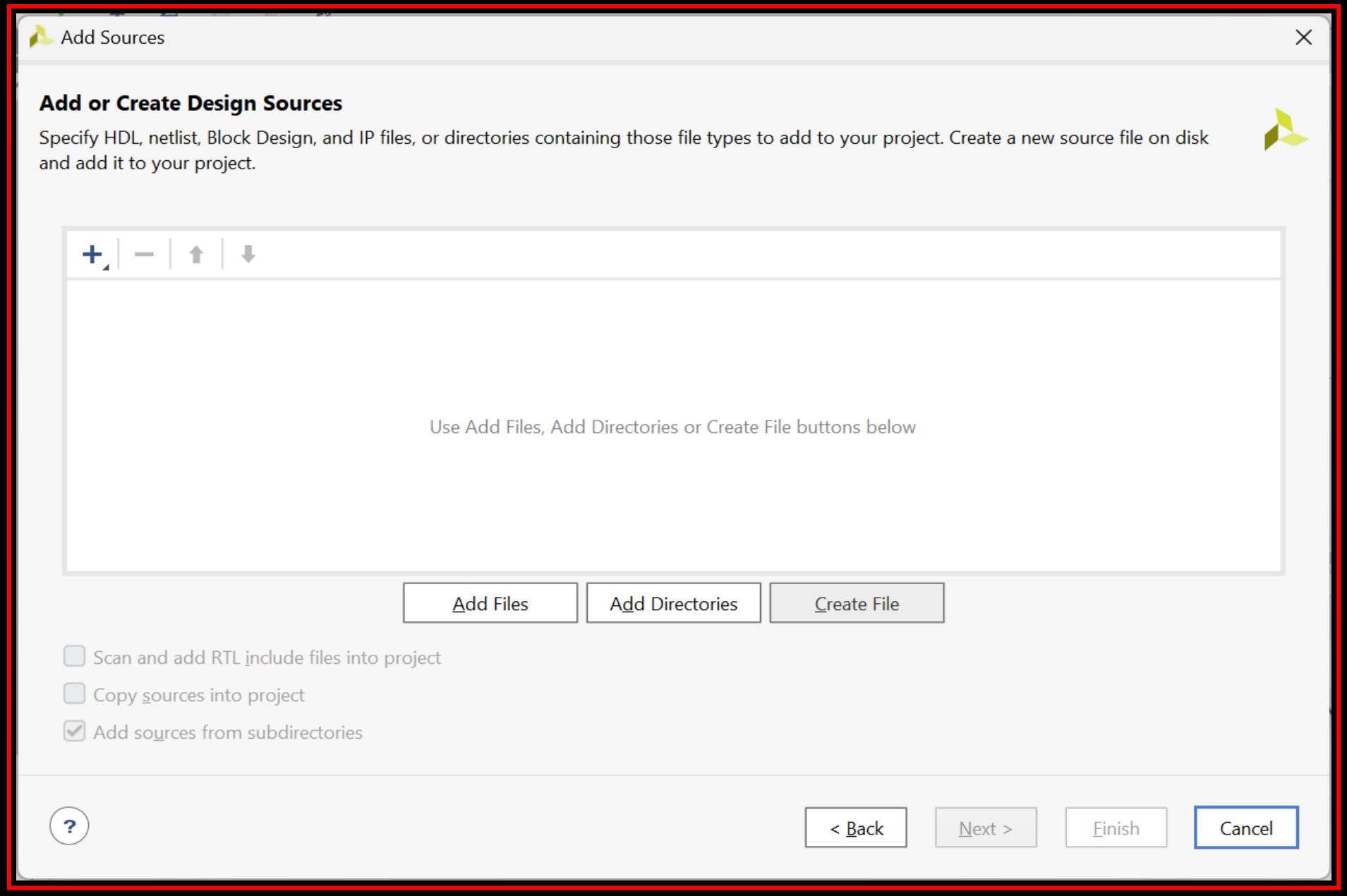
12 Right click on "Design Sources" and left click on "Add Sources" as shown below



13 Select "Add or create design sources" and click "Next"

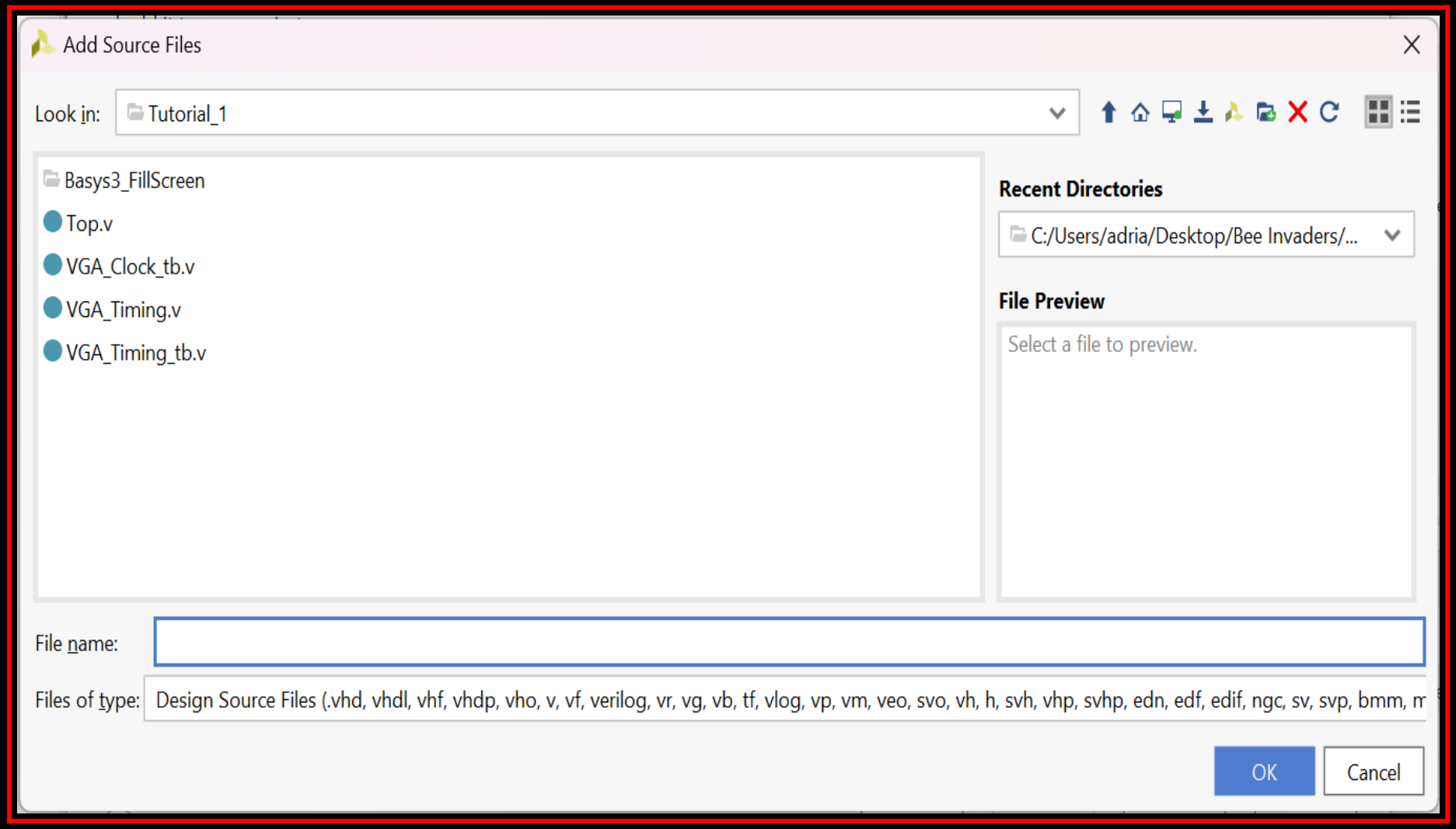


14 Click on "Add Files" button

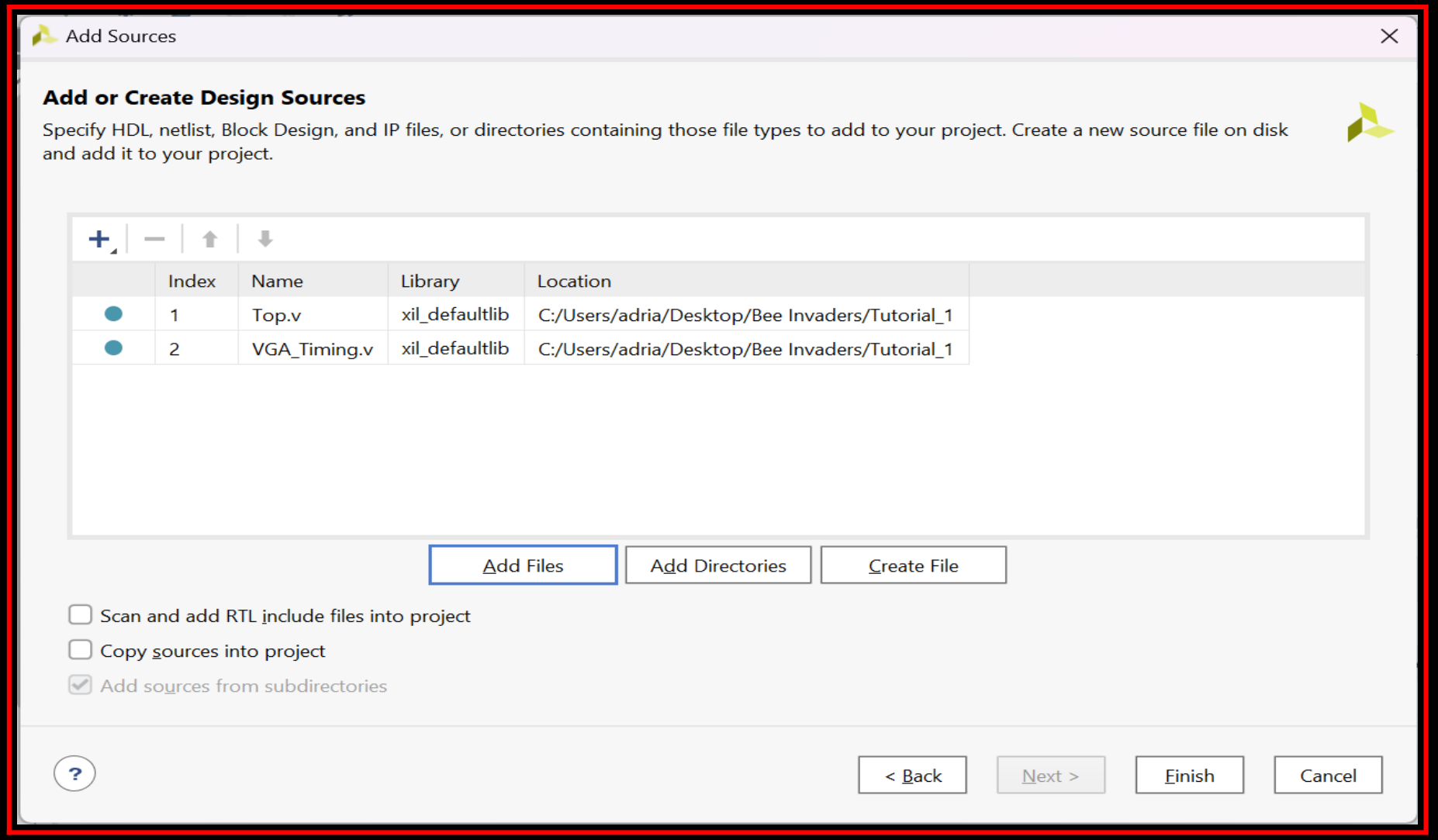


15 Make sure the path you are looking in is the "Bee Invaders/Tutorial_1", select the "Top.v" file and click "OK"

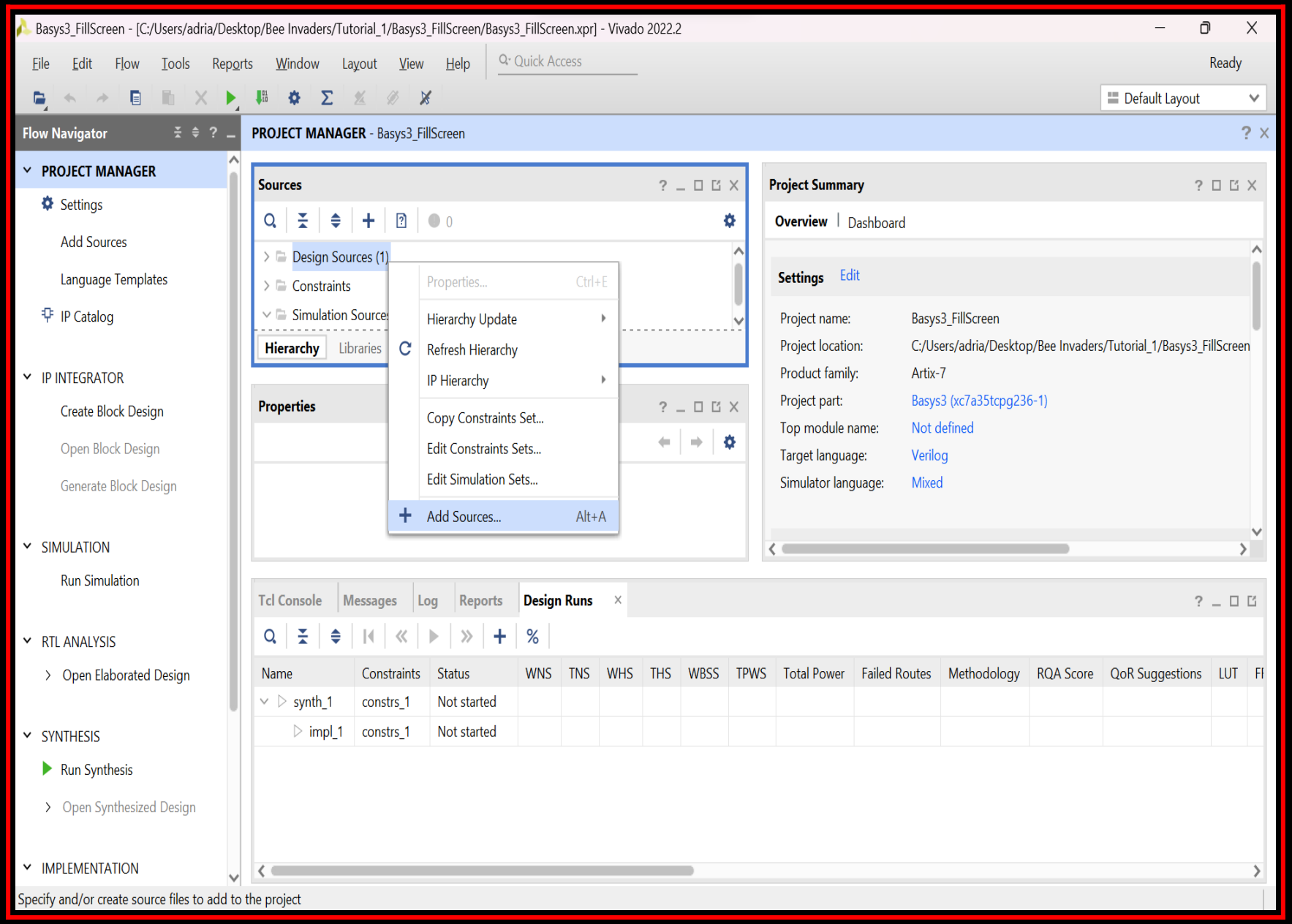
16 Click the "Add Files" button again and select the "VGA_Timing.v" file and click "OK"



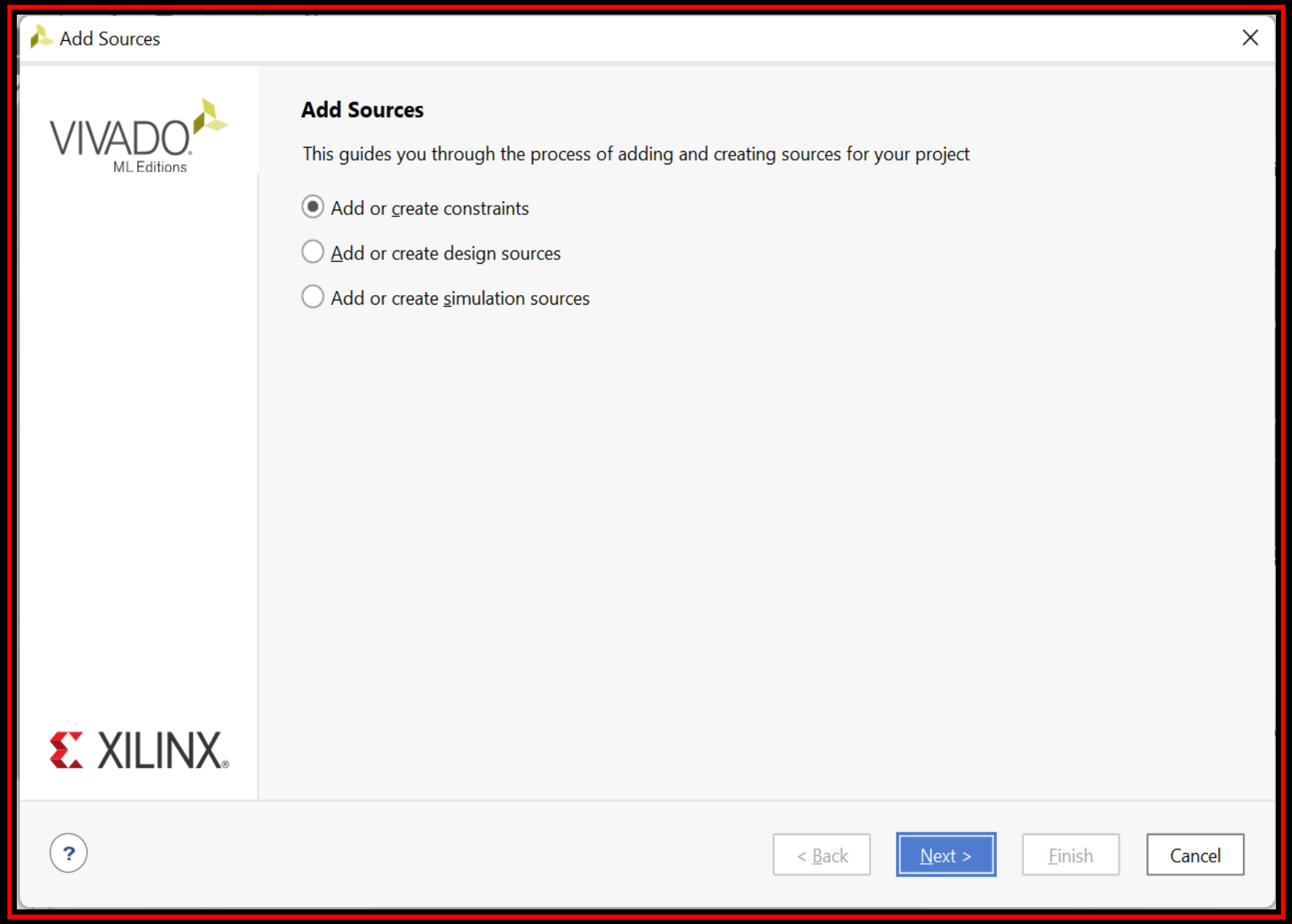
17 At the next screen select "Finish"



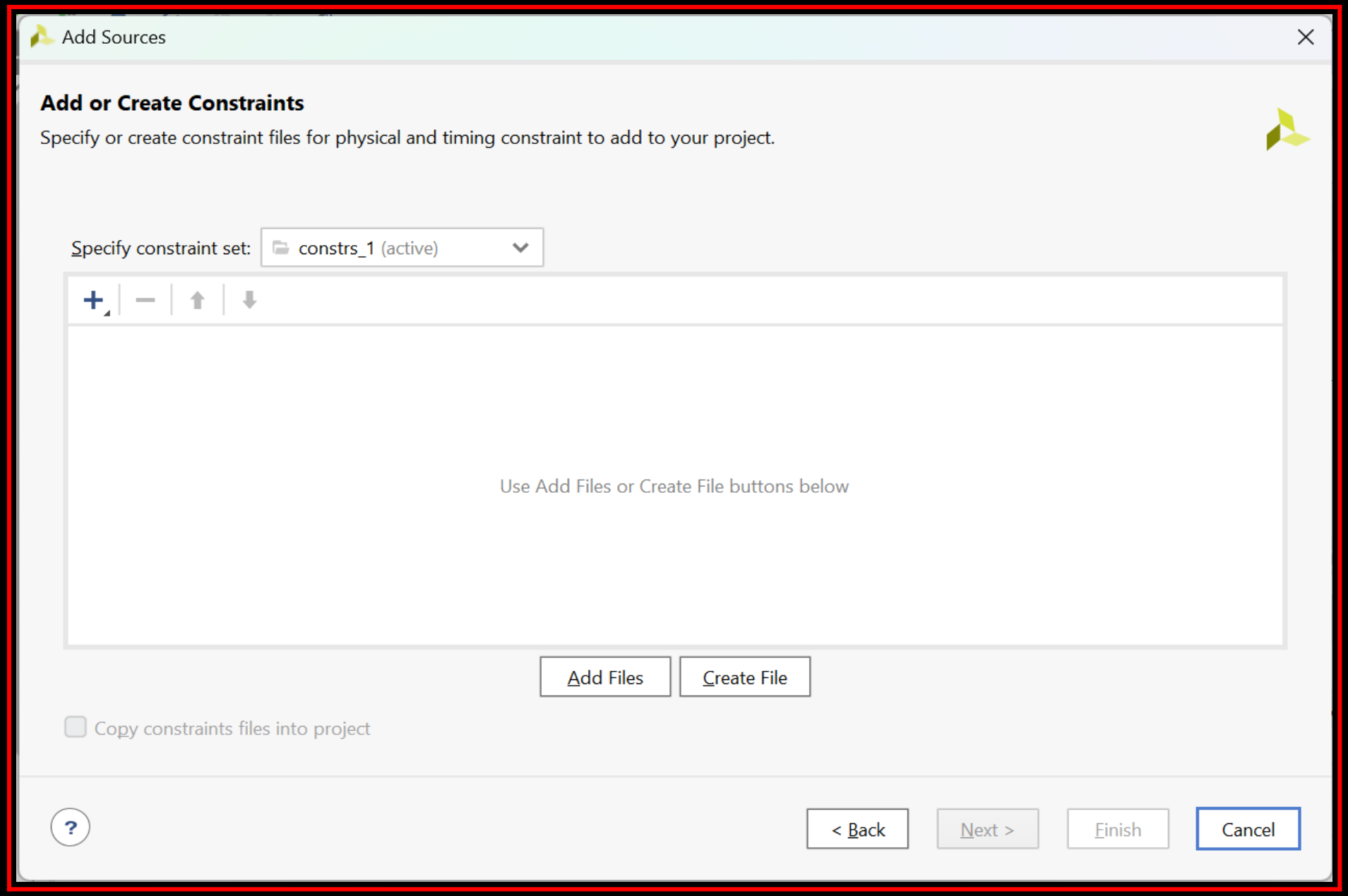
18 Right click on "Design Sources" and left click on "Add Sources" as shown below



19 Select "Add or create constraints" and click "Next"

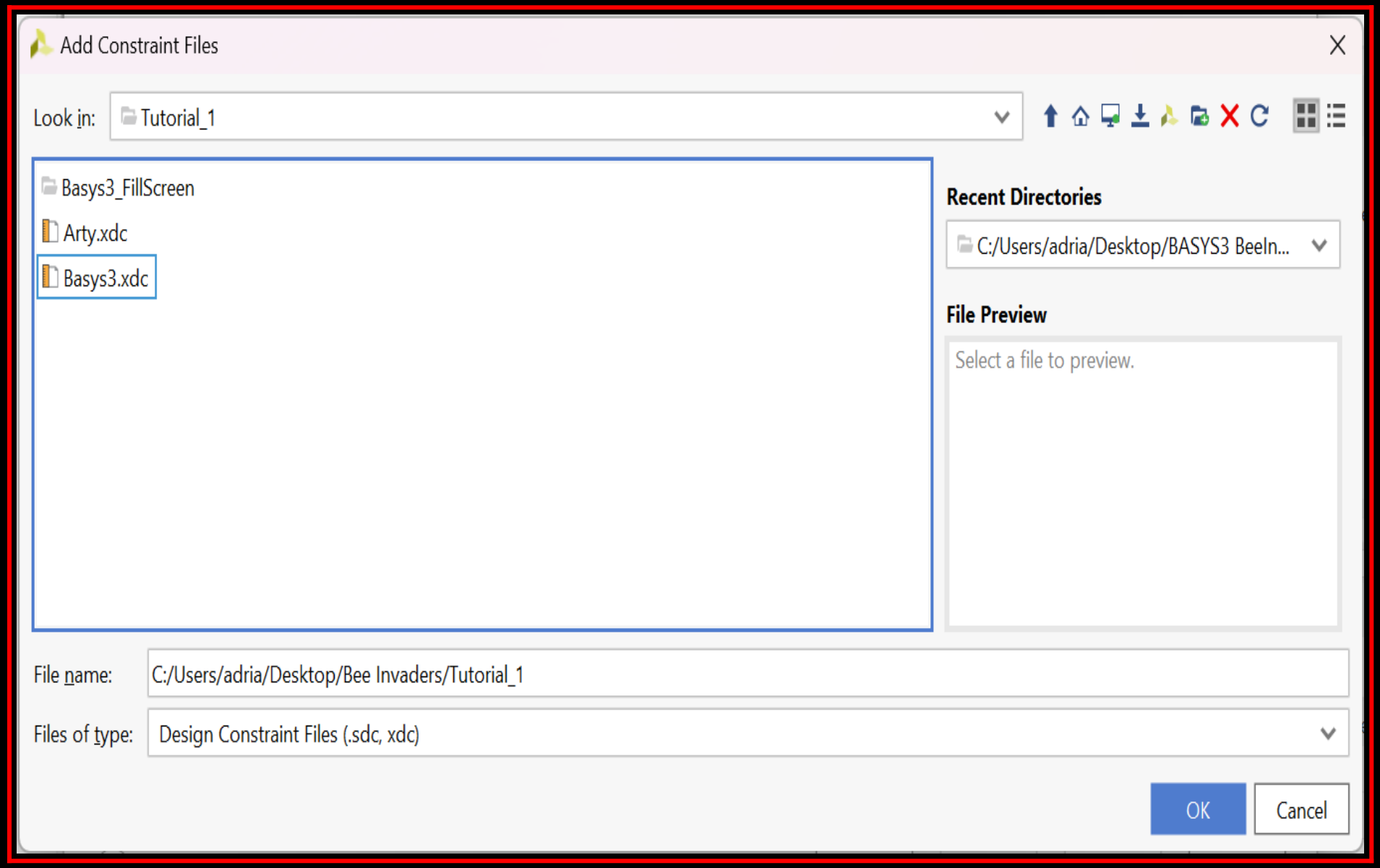


20 Click on "Add Files" button

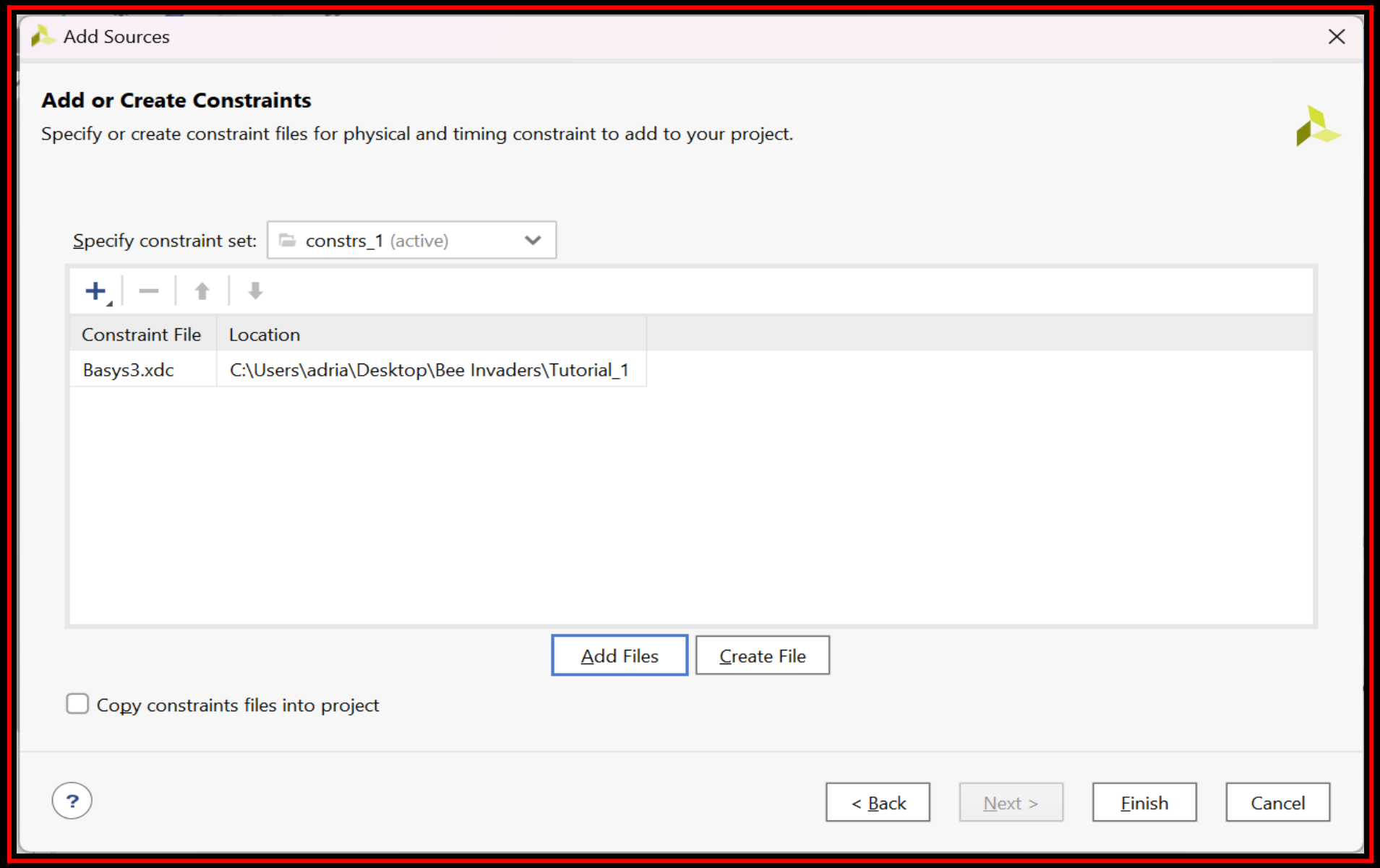


21 Select the "Basys3.xdc" (or "Arty.xdc" if you are using this board) file and click "OK"

Important Note: Make sure you select the correct file for your board



22 At the next screen select "Finish"



To change the colour scheme for the text editor in Vivado select:

Tools - Settings - (Expand) Text Editor

THE CODE

This is the code from the file "Top.v"

For this to work on the Arty A7-35 all you need to do is replace this line:

```
.reset(btn_rst_n),      // reset button is active high
```

With:

```
.reset(!btn_rst_n),      // reset button is active low
```

```
//------------------------------------------------
// Top.v module
// Digilent Basys 3
// Bee Invaders Tutorial_1
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//------------------------------------------------

`default_nettype none
`timescale 1ns / 1ps

module Top (
    input  wire clk_100m,       // 100 MHz clock
    input  wire btn_rst_n,      // reset button
    output wire vga_hsync,      // VGA horizontal sync
    output wire vga_vsync,      // VGA vertical sync
    output reg [3:0] vga_r,     // 4-bit VGA red
    output reg [3:0] vga_g,     // 4-bit VGA green
    output reg [3:0] vga_b      // 4-bit VGA blue
    );

    // generate pixel clock
    reg reset;                  // Reset Button
    wire clk_pix;               // 25.2Mhz Pixel clock
```

```
    wire clk_pix_locked;        // Pixel clock locked?

    VGA_Clock clock_pix_inst (
       .clk_100m(clk_100m),
       .reset(btn_rst_n),      // reset button is active high
       .clk_pix(clk_pix),
       .clk_pix_locked(clk_pix_locked)
    );

    // display sync signals and coordinates
    localparam CORDW = 10;      // screen coordinate width in bits
   reg rst_pix;
    wire [CORDW-1:0] sx, sy;
    wire hsync;
    wire vsync;
    wire de;
    VGA_Timing display_inst (
        .clk_pix(clk_pix),
        .rst_pix(!clk_pix_locked),  // wait for clock lock
        .sx(sx),
        .sy(sy),
        .hsync(hsync),
        .vsync(vsync),
        .de(de)
    );

    // VGA Output
    assign vga_hsync = hsync;
    assign vga_vsync = vsync;
    always @(posedge clk_pix) begin
        if (de)
            begin  // VGA colour blue
                vga_r <= 4'hD;
                vga_g <= 4'hA;
                vga_b <= 4'h5;
            end
        else
            begin  // VGA colour should be black in blanking interval
                vga_r <= 4'h0;
                vga_g <= 4'h0;
                vga_b <= 4'h0;
            end
        end
endmodule
```

This is the code from the file “VGA_Timing.v”

```
//---------------------------------------------
// VGA_Timing.v module
// Digilent Basys 3
// Bee Invaders Tutorial_1
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//---------------------------------------------

`default_nettype none
`timescale 1ns / 1ps
```

```
module VGA_Timing (
      input  wire clk_pix,   // pixel clock
      input  wire rst_pix,   // reset in pixel clock domain
      output reg [9:0] sx,   // horizontal screen position
      output reg [9:0] sy,   // vertical screen position
      output wire hsync,     // horizontal sync
      output wire vsync,     // vertical sync
      output wire de         // data enable (low in blanking interval)
      );

      // horizontal timings
      parameter HA_END = 639;           // end of active pixels
      parameter HS_STA = HA_END + 16;   // sync starts after front porch
      parameter HS_END = HS_STA + 96;   // sync ends
      parameter LINE   = 799;           // last pixel on line (after back porch)

      // vertical timings
      parameter VA_END = 479;           // end of active pixels
      parameter VS_STA = VA_END + 10;   // sync starts after front porch
      parameter VS_END = VS_STA + 2;    // sync ends
      parameter SCREEN = 524;           // last line on screen (after back porch)

      assign hsync = ~(sx >= HS_STA && sx < HS_END);  // invert: negative polarity
      assign vsync = ~(sy >= VS_STA && sy < VS_END);  // invert: negative polarity
      assign de = (sx <= HA_END && sy <= VA_END);

      // calculate horizontal and vertical screen position
      always @(posedge clk_pix) begin
         if (sx == LINE) begin  // last pixel on line?
               sx <= 0;
               sy <= (sy == SCREEN) ? 0 : sy + 1;  // last line on screen?
         end else begin
            sx <= sx + 1;
         end
         if (rst_pix) begin
               sx <= 0;
               sy <= 0;
         end
      end
   endmodule
```

This is the code from the file “Basys3.xdc”

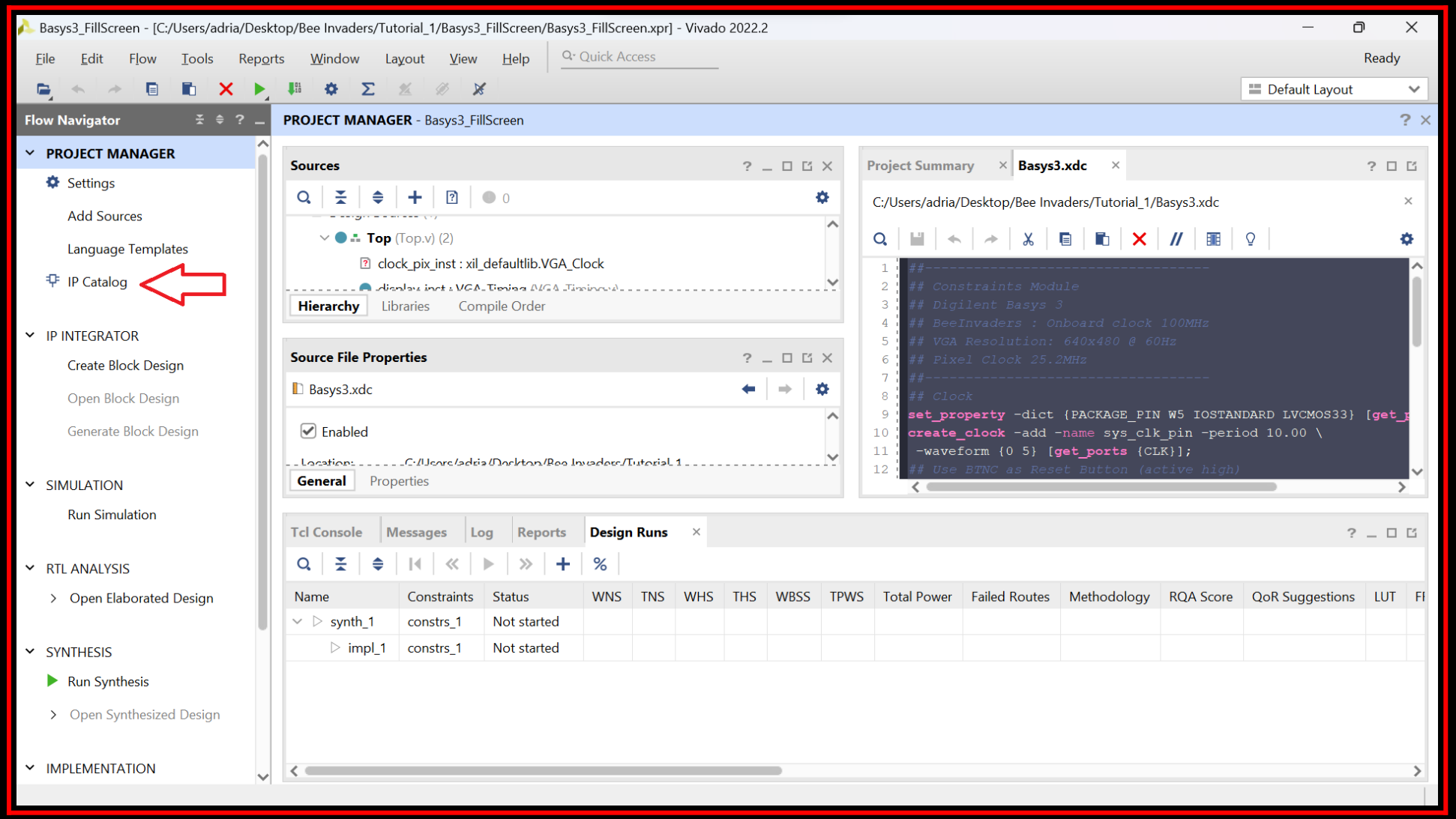
```
##-----------------------------------
## Constraints Module
## Digilent Basys 3
## BeeInvaders : Onboard clock 100MHz
## VGA Resolution: 640x480 @ 60Hz
## Pixel Clock 25.2MHz
##-----------------------------------
## Clock
set_property -dict {PACKAGE_PIN W5 IOSTANDARD LVCMOS33} [get_ports {clk_100m}];
create_clock -add -name sys_clk_pin -period 10.00 \
-waveform {0 5} [get_ports {clk_100m}];
## Use BTNC as Reset Button (active high)
set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports {btn_rst_n}];
## VGA Connector
set_property -dict {PACKAGE_PIN G19 IOSTANDARD LVCMOS33} [get_ports {vga_r[0]}];
set_property -dict {PACKAGE_PIN H19 IOSTANDARD LVCMOS33} [get_ports {vga_r[1]}];
```

```
set_property -dict {PACKAGE_PIN J19 IOSTANDARD LVCMOS33} [get_ports {vga_r[2]}];
set_property -dict {PACKAGE_PIN N19 IOSTANDARD LVCMOS33} [get_ports {vga_r[3]}];
set_property -dict {PACKAGE_PIN N18 IOSTANDARD LVCMOS33} [get_ports {vga_b[0]}];
set_property -dict {PACKAGE_PIN L18 IOSTANDARD LVCMOS33} [get_ports {vga_b[1]}];
set_property -dict {PACKAGE_PIN K18 IOSTANDARD LVCMOS33} [get_ports {vga_b[2]}];
set_property -dict {PACKAGE_PIN J18 IOSTANDARD LVCMOS33} [get_ports {vga_b[3]}];
set_property -dict {PACKAGE_PIN J17 IOSTANDARD LVCMOS33} [get_ports {vga_g[0]}];
set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports {vga_g[1]}];
set_property -dict {PACKAGE_PIN G17 IOSTANDARD LVCMOS33} [get_ports {vga_g[2]}];
set_property -dict {PACKAGE_PIN D17 IOSTANDARD LVCMOS33} [get_ports {vga_g[3]}];
set_property -dict {PACKAGE_PIN P19 IOSTANDARD LVCMOS33} [get_ports {vga_hsync}];
set_property -dict {PACKAGE_PIN R19 IOSTANDARD LVCMOS33} [get_ports {vga_vsync}];
## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
```

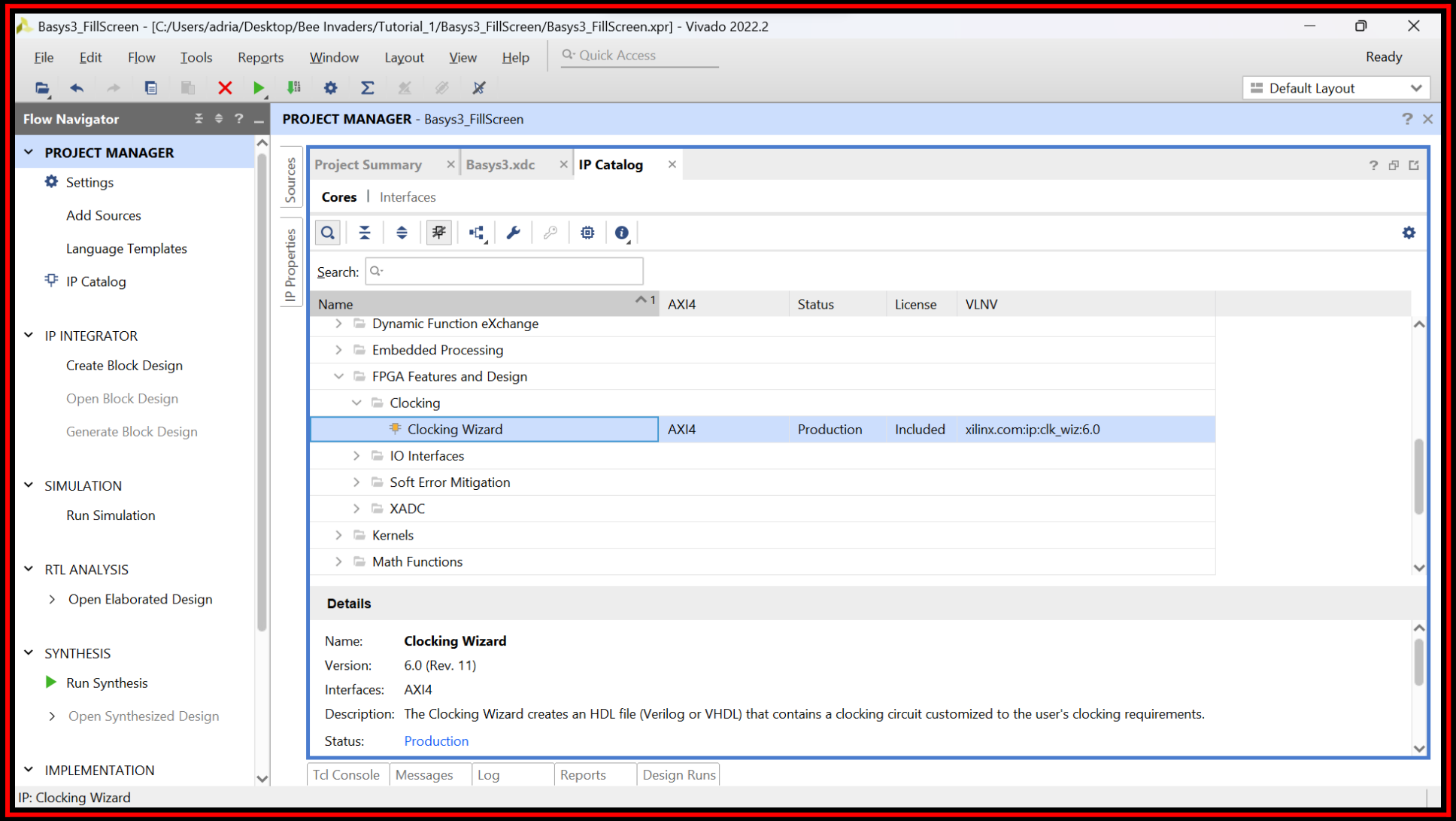
This is the code from the file “Arty.xdc”

```
##-----------------------------------------------
## Constraints File
## Digilent Arty A7-35
## Bee Invaders Tutorial_1
## Onboard clock 100MHz
## VGA Resolution: 640x480 @ 60Hz
## Pixel Clock 25.2MHz
//-----------------------------------------------

## FPGA Configuration I/O Options
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## Board Clock: 100 MHz
set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports {clk_100m}];
create_clock -name clk_100m -period 10.00 [get_ports {clk_100m}];

## Buttons
set_property -dict {PACKAGE_PIN C2 IOSTANDARD LVCMOS33} [get_ports {btn_rst_n}];

## VGA Pmod on Header JB/JC
set_property -dict {PACKAGE_PIN U14 IOSTANDARD LVCMOS33} [get_ports {vga_hsync}];
set_property -dict {PACKAGE_PIN V14 IOSTANDARD LVCMOS33} [get_ports {vga_vsync}];
set_property -dict {PACKAGE_PIN E15 IOSTANDARD LVCMOS33} [get_ports {vga_r[0]}];
set_property -dict {PACKAGE_PIN E16 IOSTANDARD LVCMOS33} [get_ports {vga_r[1]}];
set_property -dict {PACKAGE_PIN D15 IOSTANDARD LVCMOS33} [get_ports {vga_r[2]}];
set_property -dict {PACKAGE_PIN C15 IOSTANDARD LVCMOS33} [get_ports {vga_r[3]}];
set_property -dict {PACKAGE_PIN U12 IOSTANDARD LVCMOS33} [get_ports {vga_g[0]}];
set_property -dict {PACKAGE_PIN V12 IOSTANDARD LVCMOS33} [get_ports {vga_g[1]}];
set_property -dict {PACKAGE_PIN V10 IOSTANDARD LVCMOS33} [get_ports {vga_g[2]}];
set_property -dict {PACKAGE_PIN V11 IOSTANDARD LVCMOS33} [get_ports {vga_g[3]}];
set_property -dict {PACKAGE_PIN J17 IOSTANDARD LVCMOS33} [get_ports {vga_b[0]}];
set_property -dict {PACKAGE_PIN J18 IOSTANDARD LVCMOS33} [get_ports {vga_b[1]}];
set_property -dict {PACKAGE_PIN K15 IOSTANDARD LVCMOS33} [get_ports {vga_b[2]}];
set_property -dict {PACKAGE_PIN J15 IOSTANDARD LVCMOS33} [get_ports {vga_b[3]}];
```

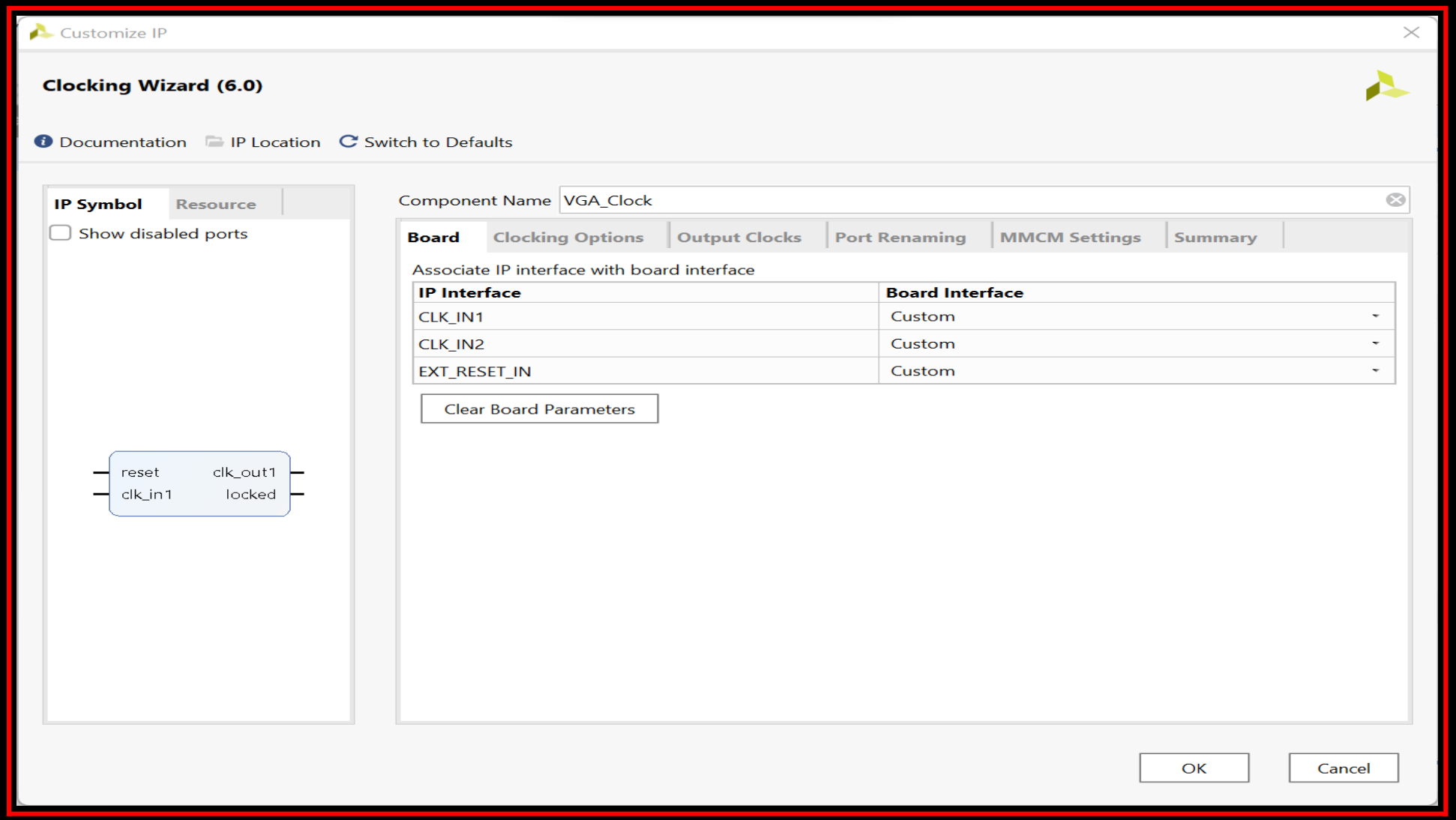
24 Next we need to create a 25.2Mhz clock by clicking on "IP Catolog"



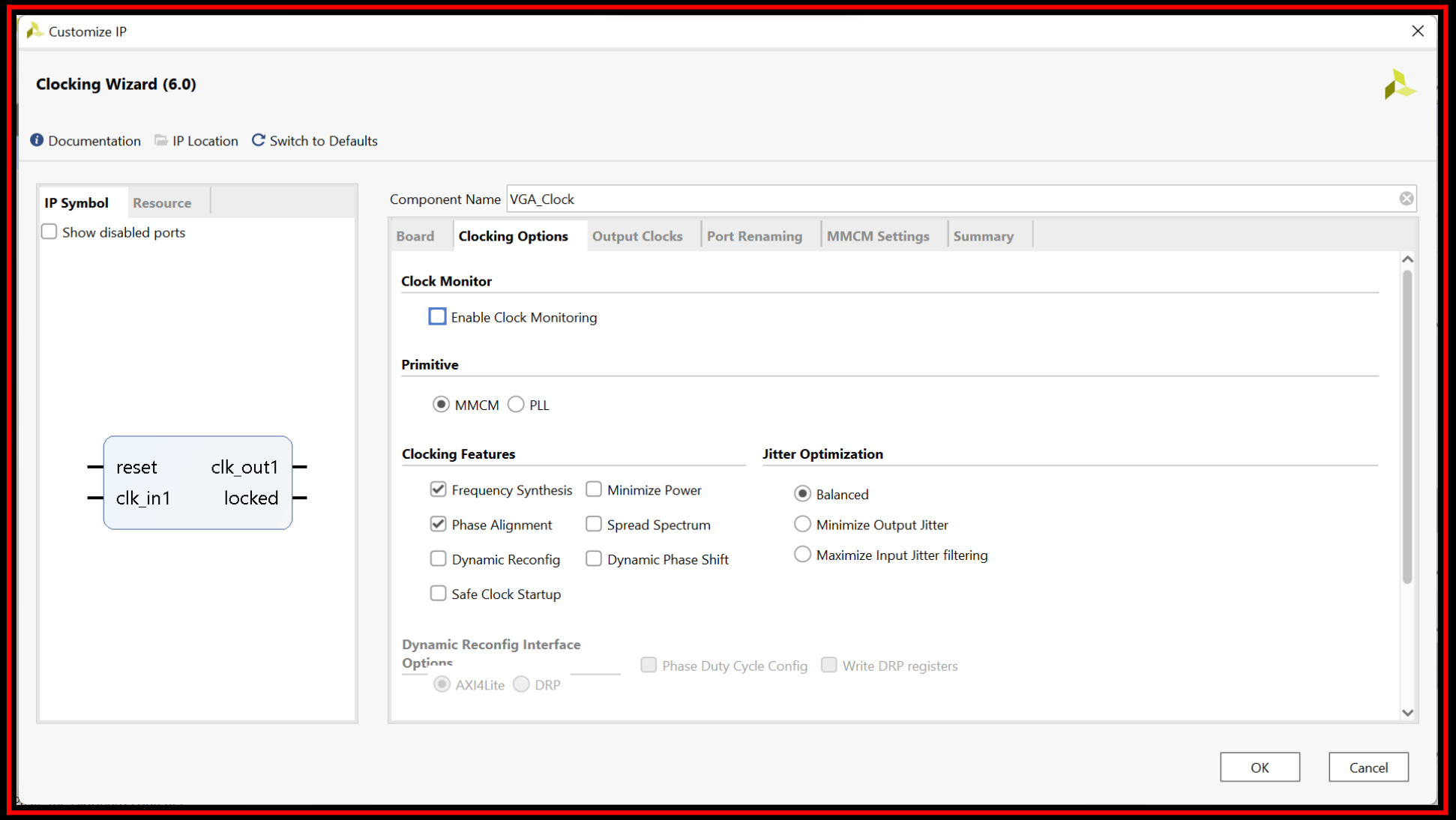
25 Navigate and double click on "Clocking Wizard"



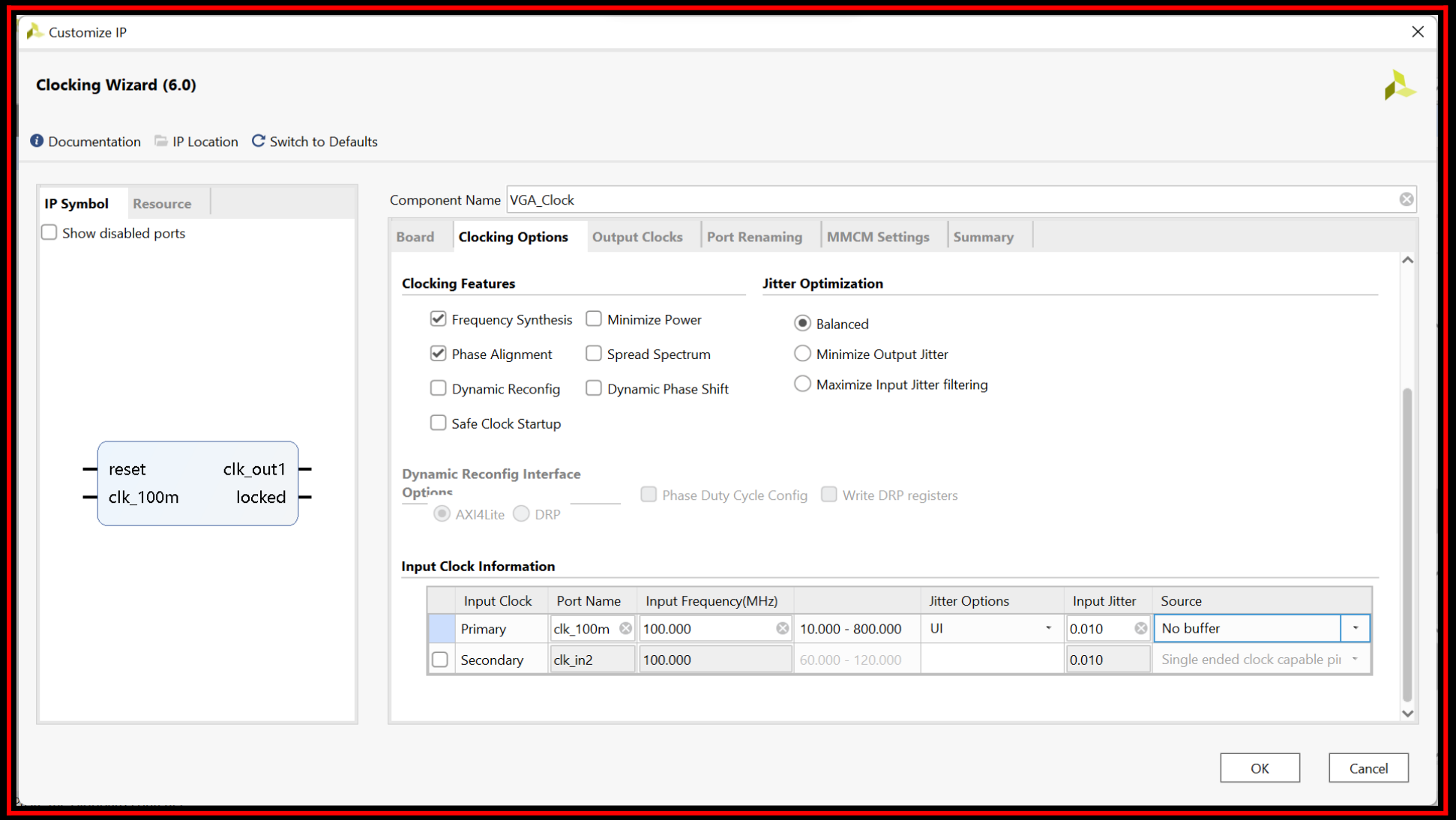
26 Untick "Show disabled ports", In the "Component Name" box enter "VGA_Clock", Then click on the tab "Clocking Options"



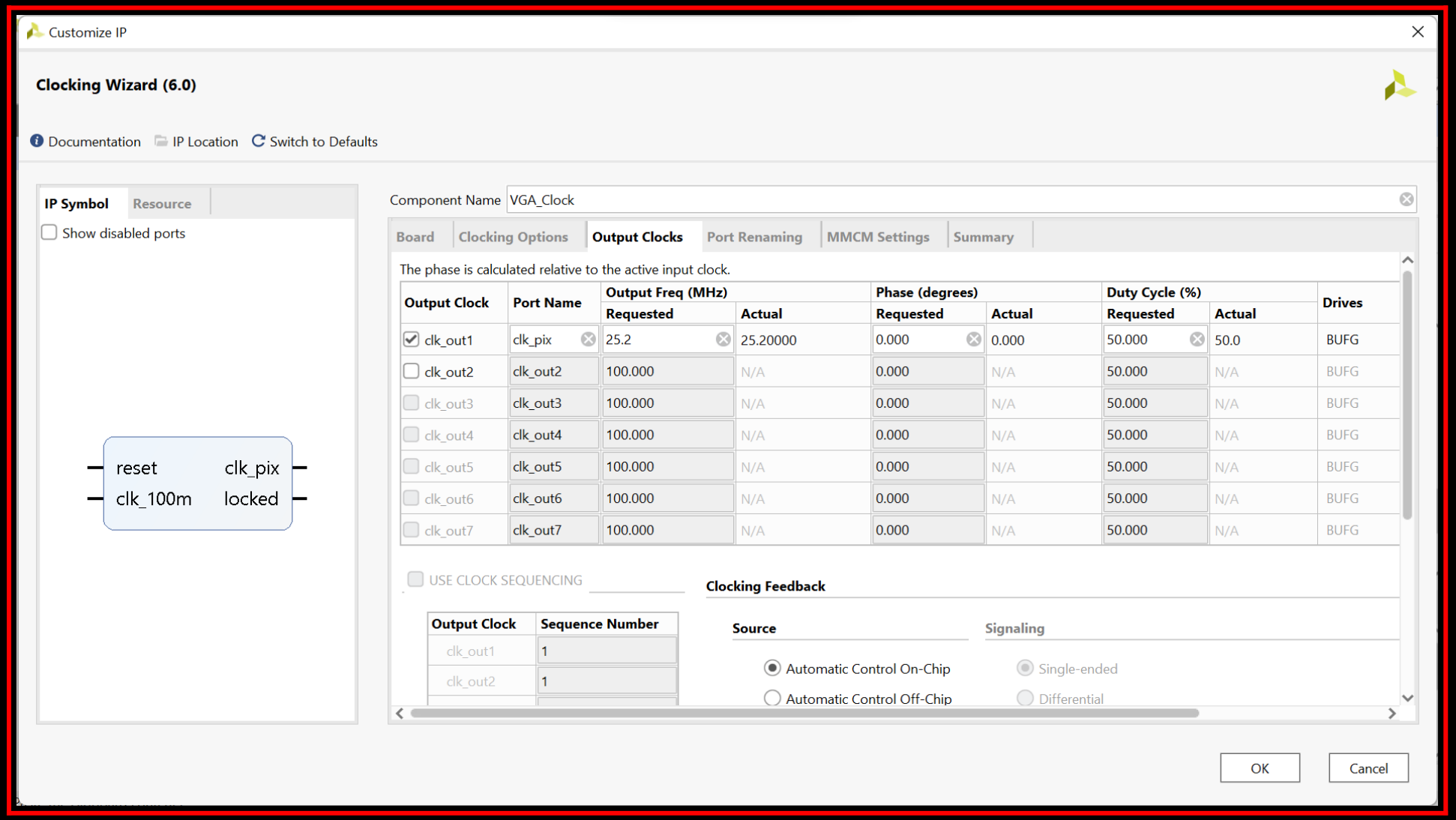
27 Select "MMCM"



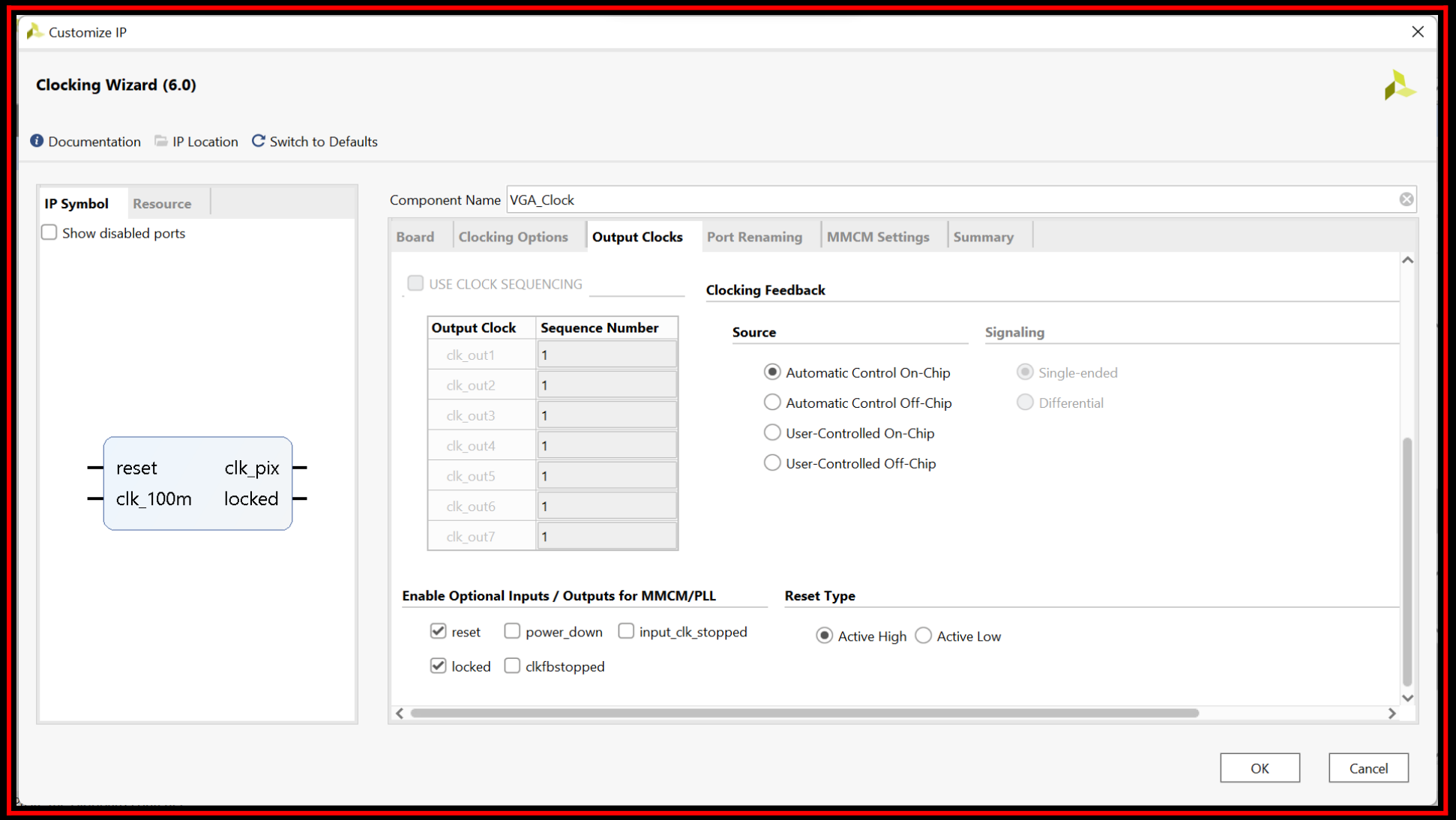
In the "Input Clock Information" change the "Port Name" to "clk_100m" next to "Primary", Change the "Source" to "No buffer" and then click the "Output Clocks" tab



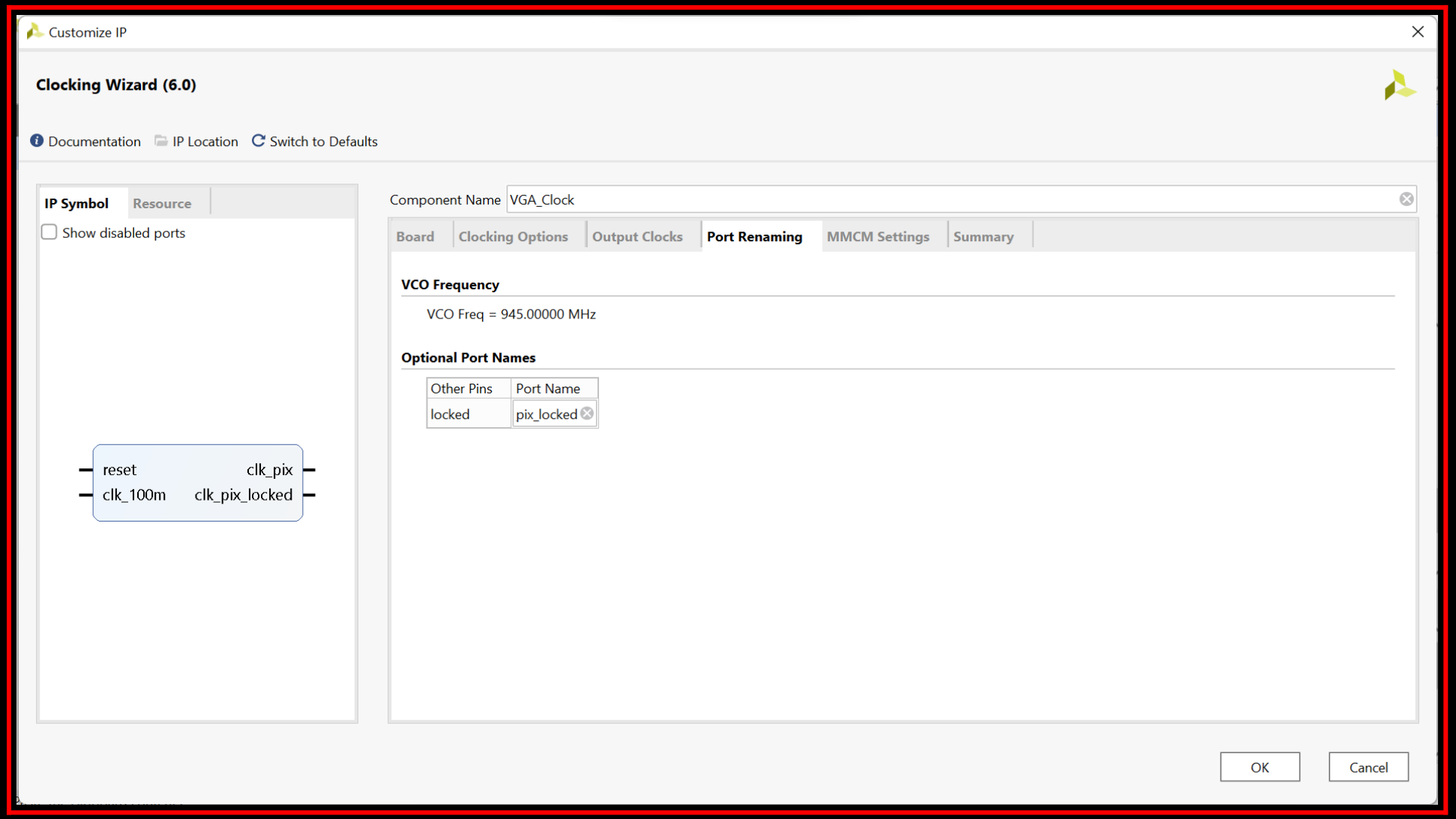
Change the "Port Name" to "clk_pix" next to "clk_out1", Change the "Output Freq (MHz) Requested" to "25.2"



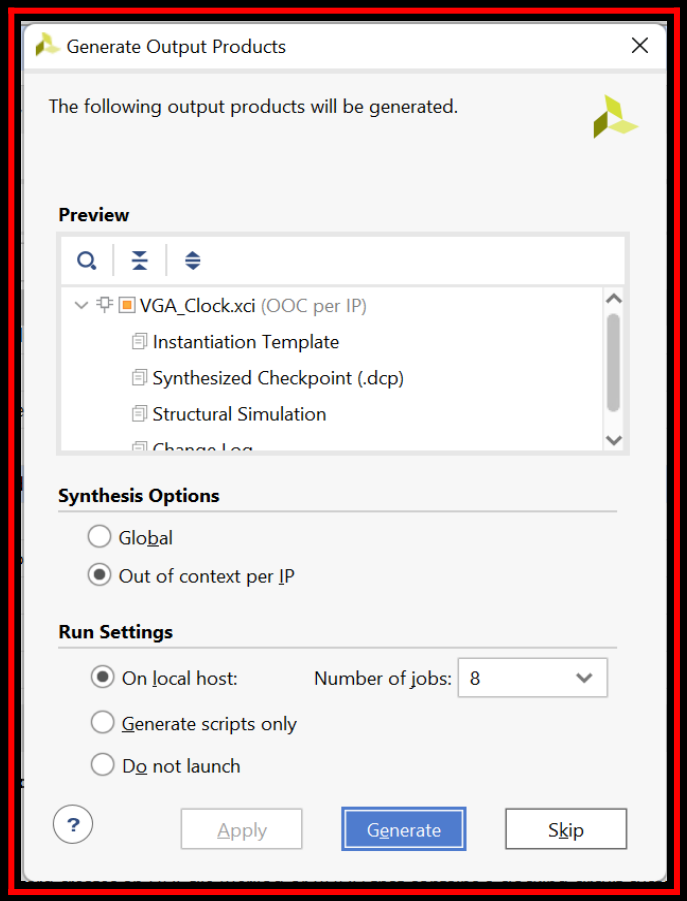
Ensure "reset" and "locked" are ticked and select "Active High" for the "Reset Type", then click on the "Port Renaming" tab



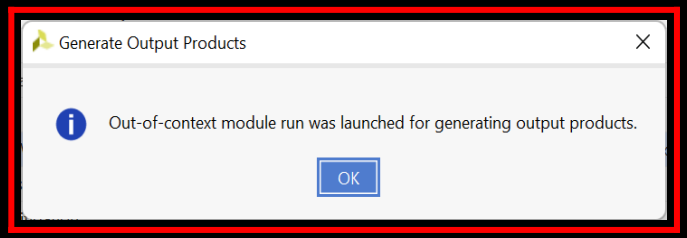
Change the "Port Name" to "clk_pix_locked" next to "locked" and then click "OK"



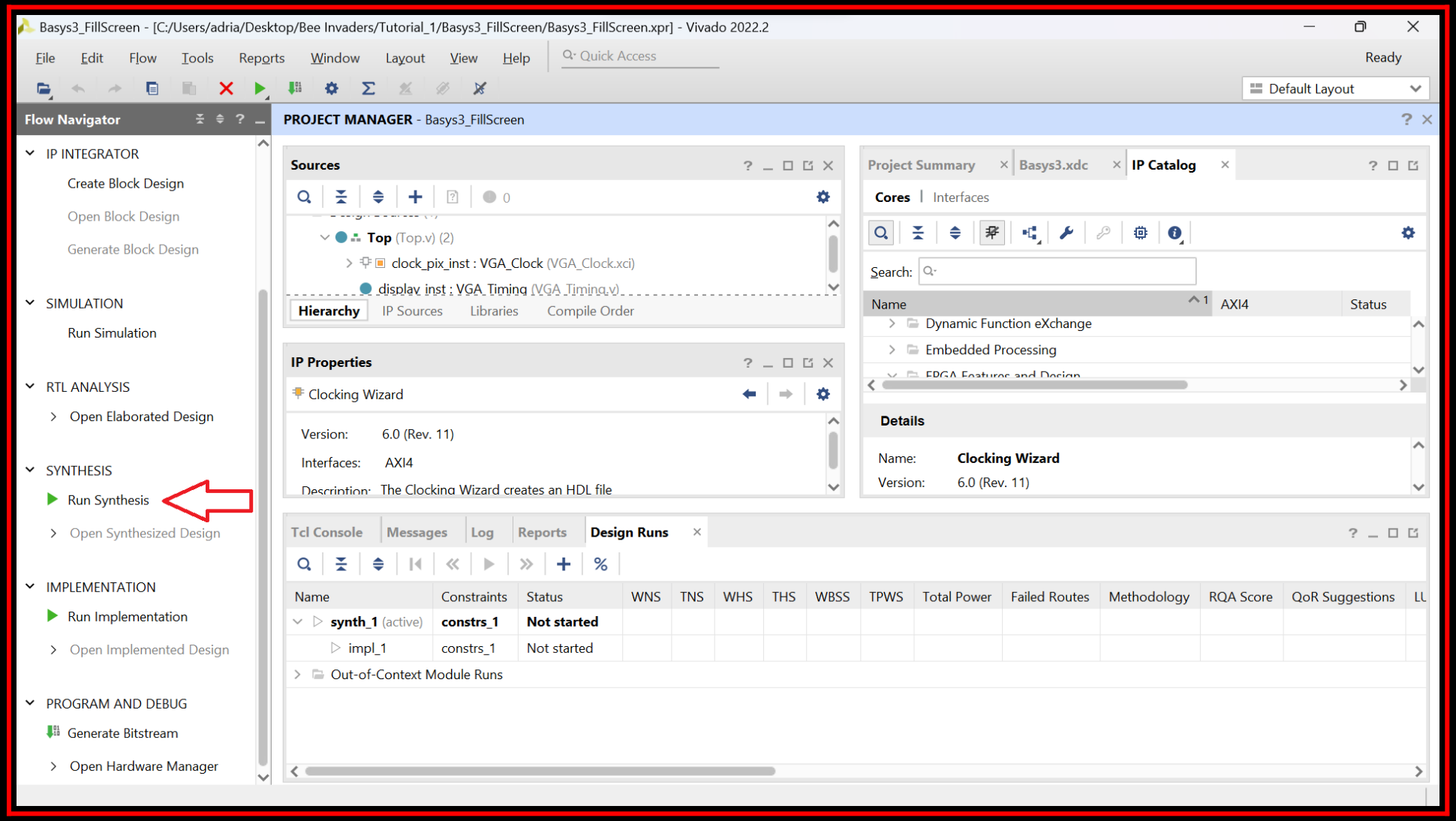
28 Click "Generate" at the next screen



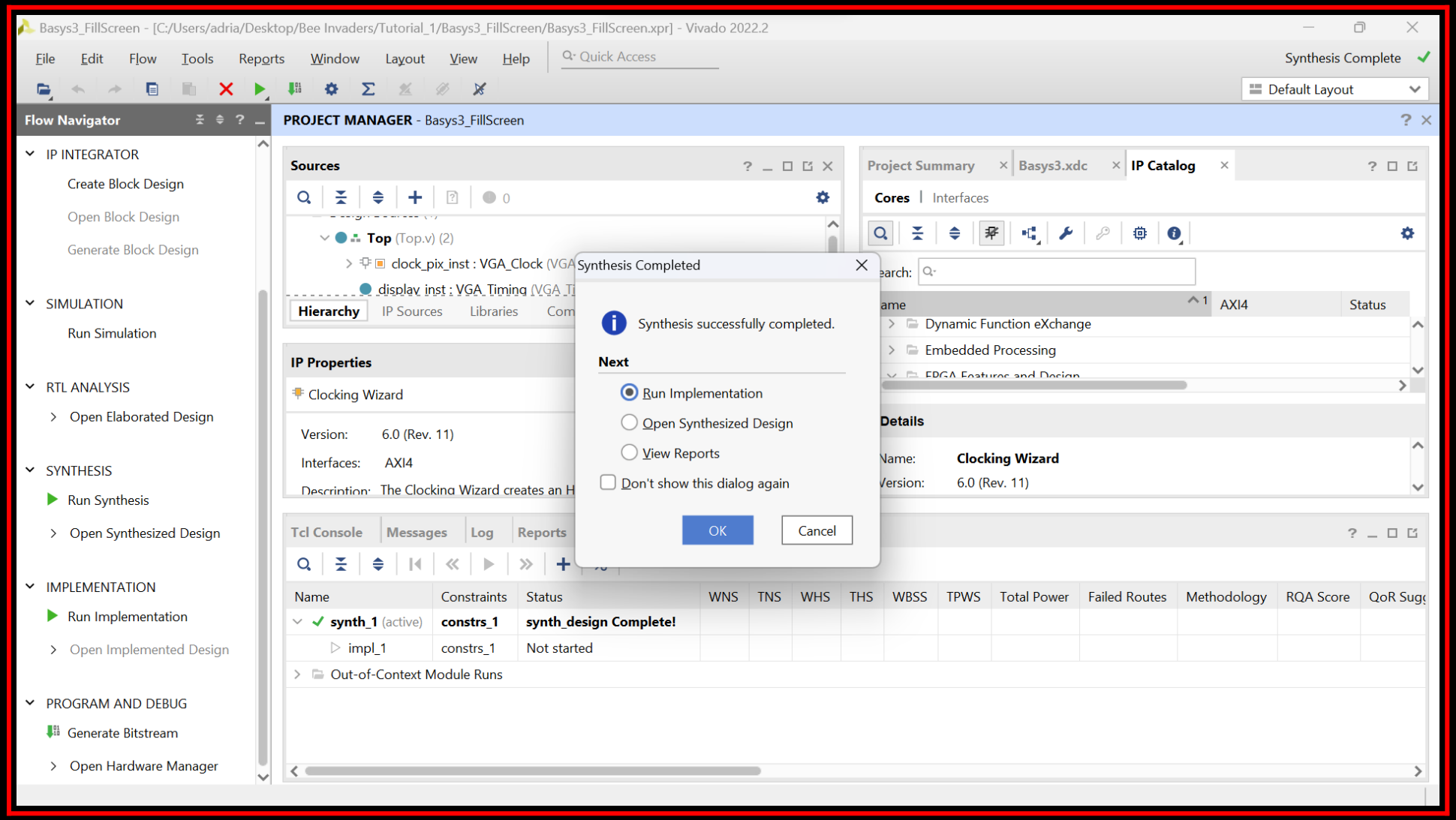
Click "OK" at the next screen



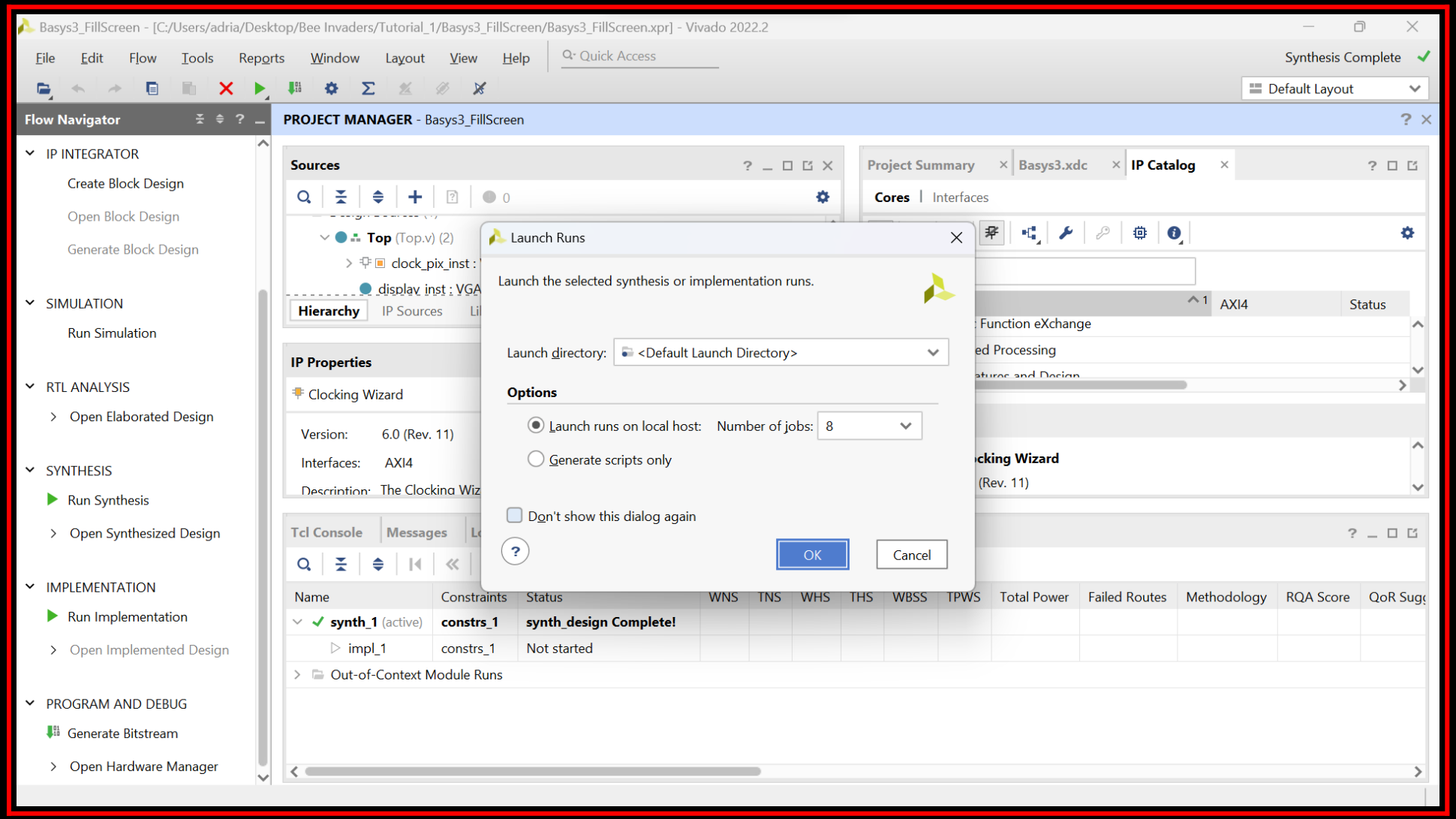
29 Click on "Run Synthesis" (if a window pops up asking if you would like to save the project click "Save") and when the "Launch Runs" window appears click "OK"



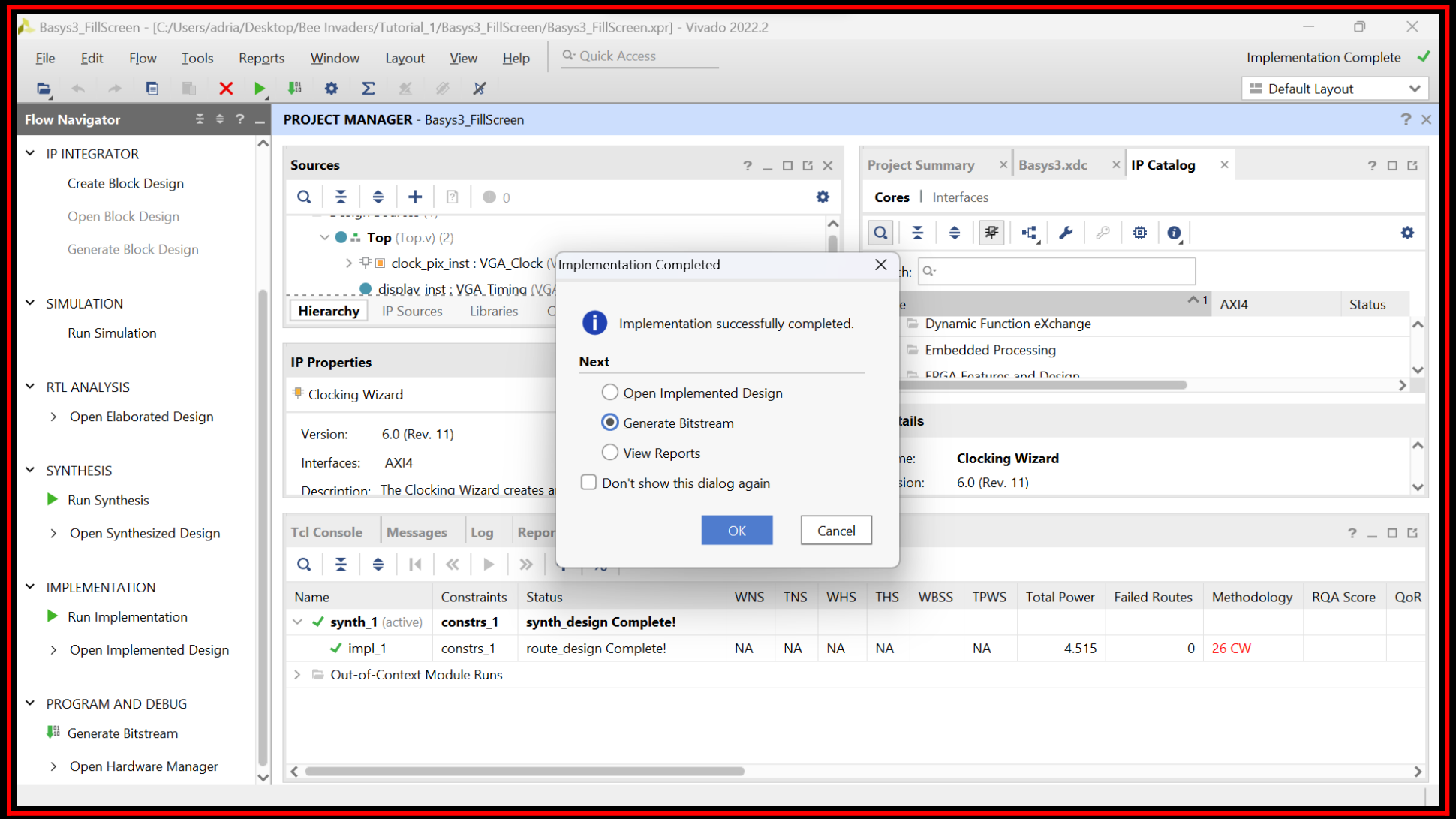
30 When the window "Synthesis Completed" appears ensure "Run implementation" is selected and click "OK"



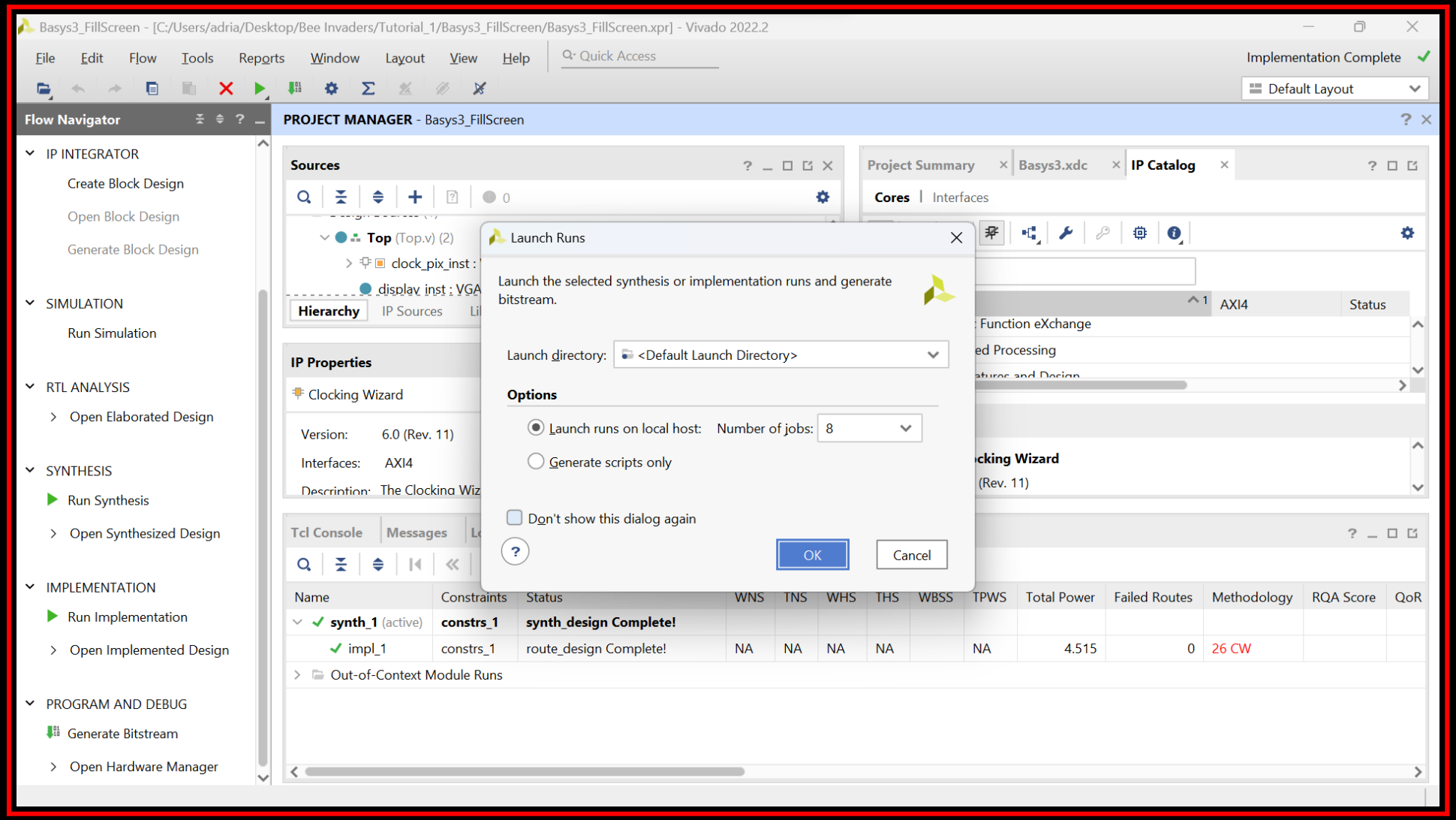
31 When the "Launch Runs" window appears click "OK"



32 When the "Implementation Completed" window appears select "Generate Bitstream" and click "OK"

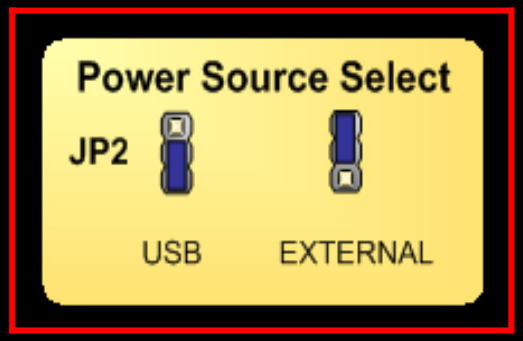


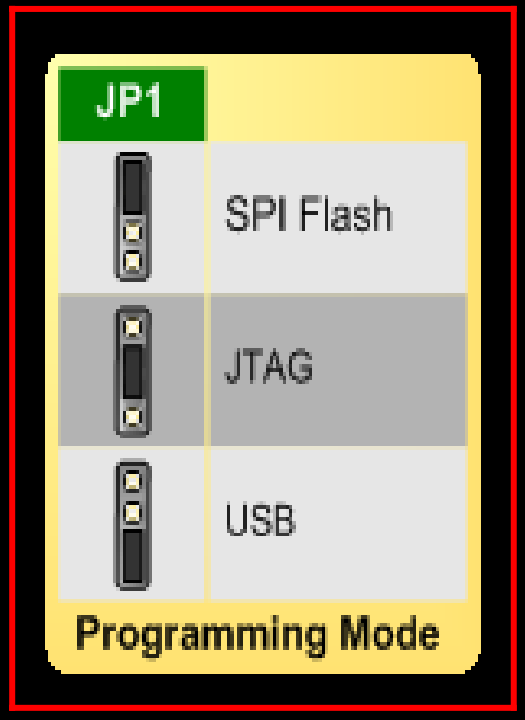
33 When the "Launch Runs" window appears click "OK"

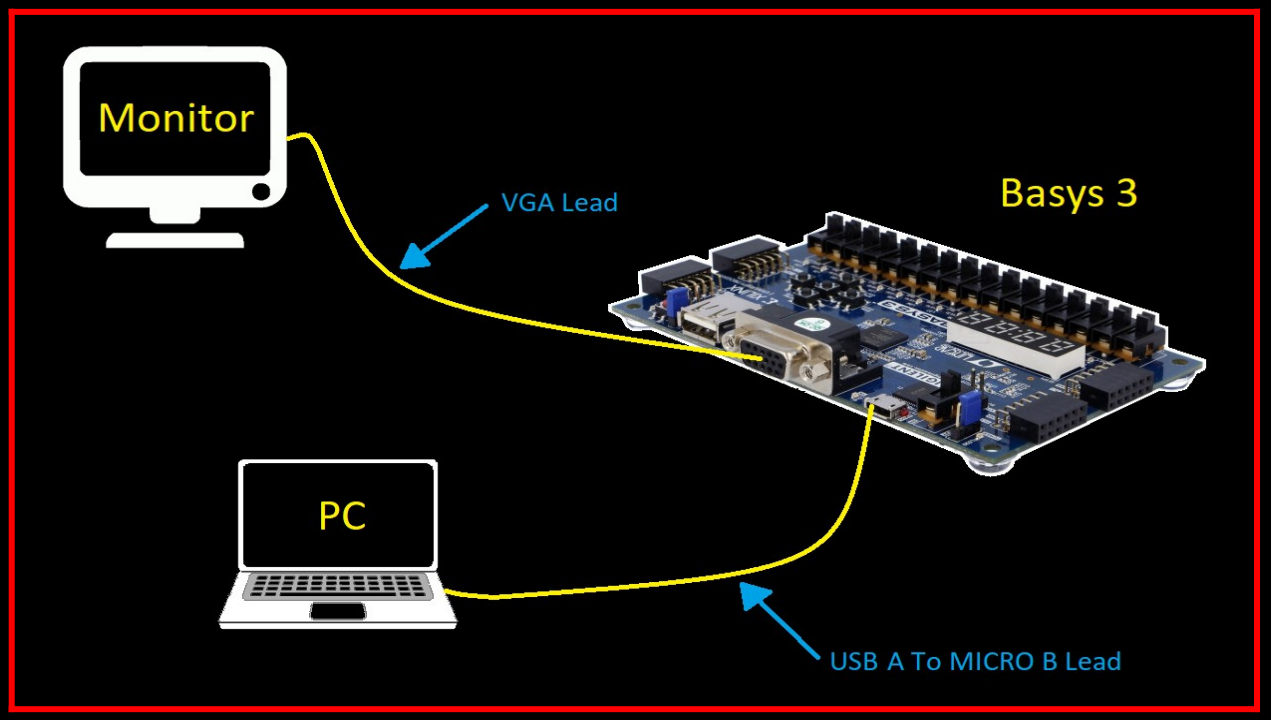


(B) HOW TO PROGRAM THE FPGA BOARD

34 When the "Bitstream Generation Completed" window appears you will need to make sure that the Basys3 board jumper JP2 is set to USB and JP1 is in the JTAG position, connect the Basys3 board to your computer and your VGA screen (as shown below) and switch the board on

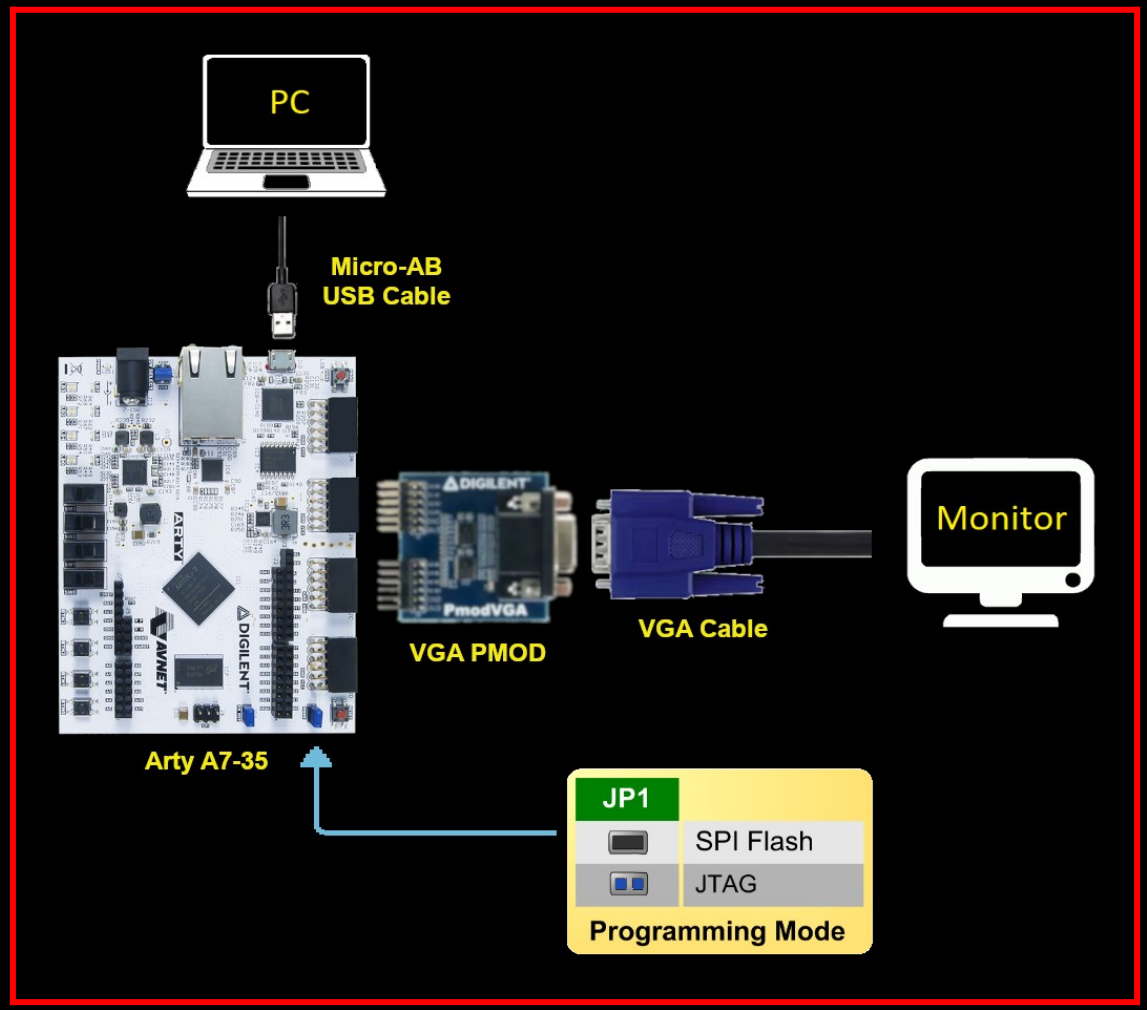




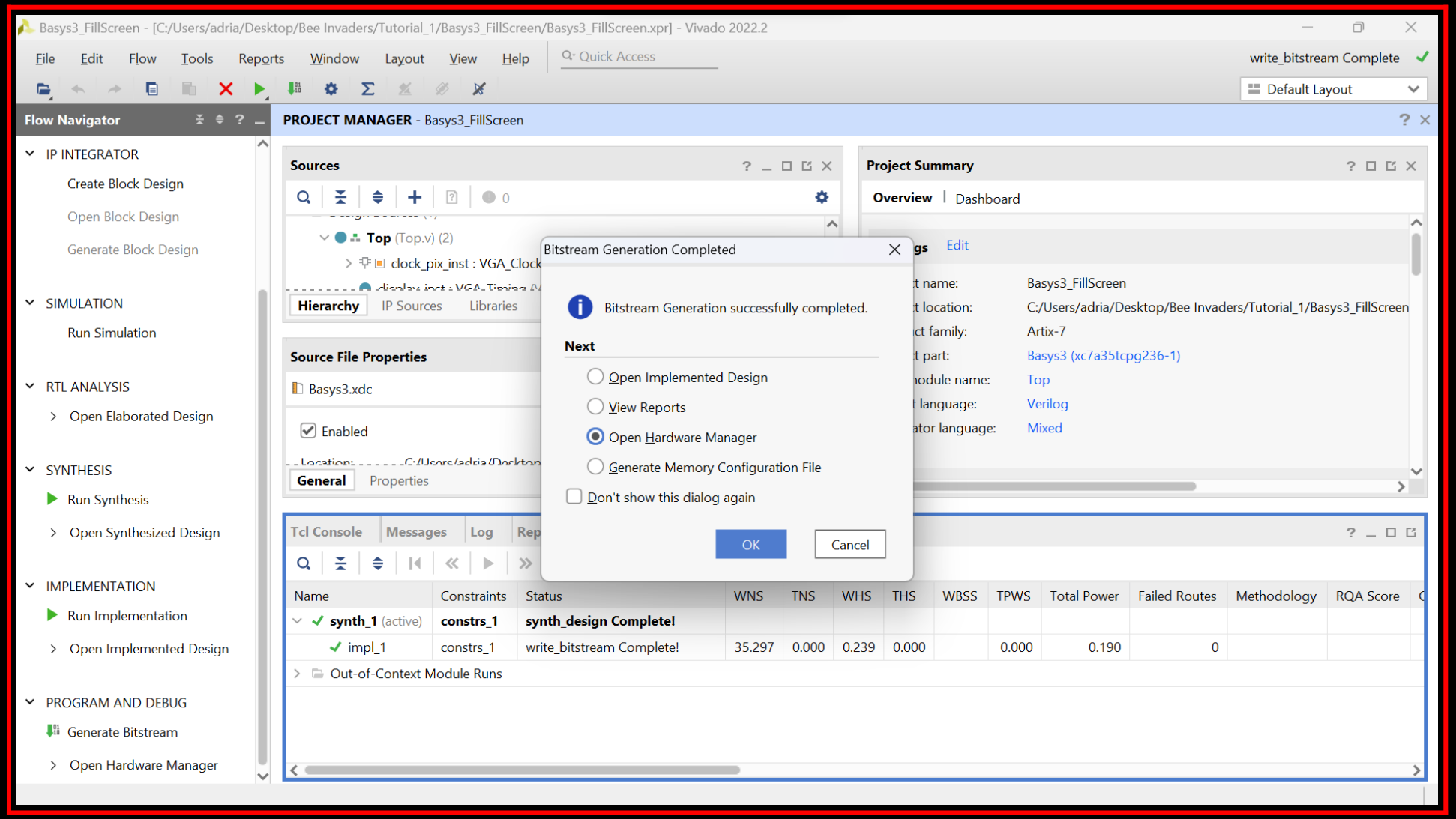


If you are using the Arty A7-35 board, when the "Bitstream Generation Completed" window appears you will need to make sure that the Arty A7-35 board jumper JP1 is set to JTAG and the VGA PMOD is connected to the Arty board

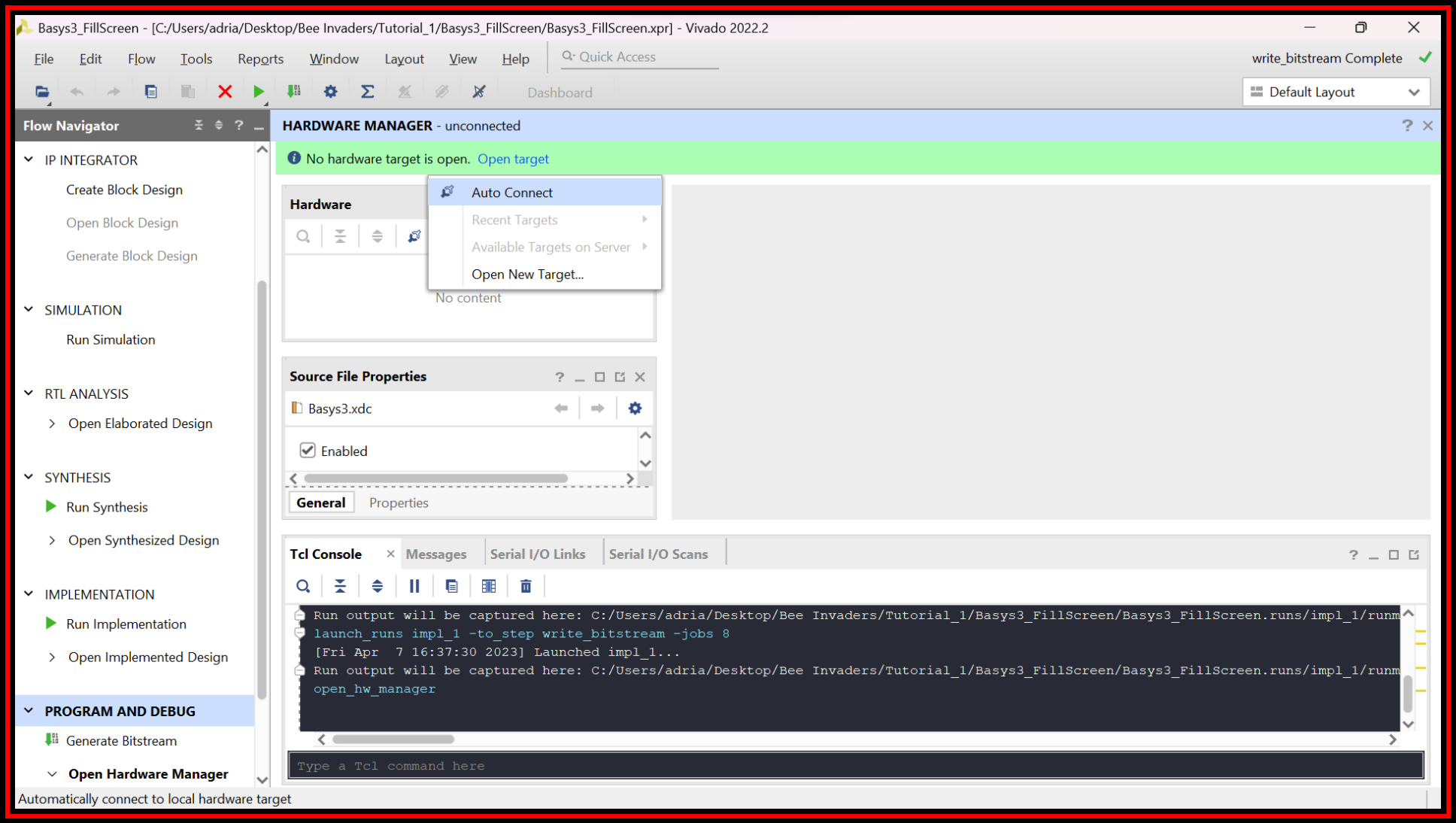
Next connect the Arty board to your computer and your VGA screen (as shown below)



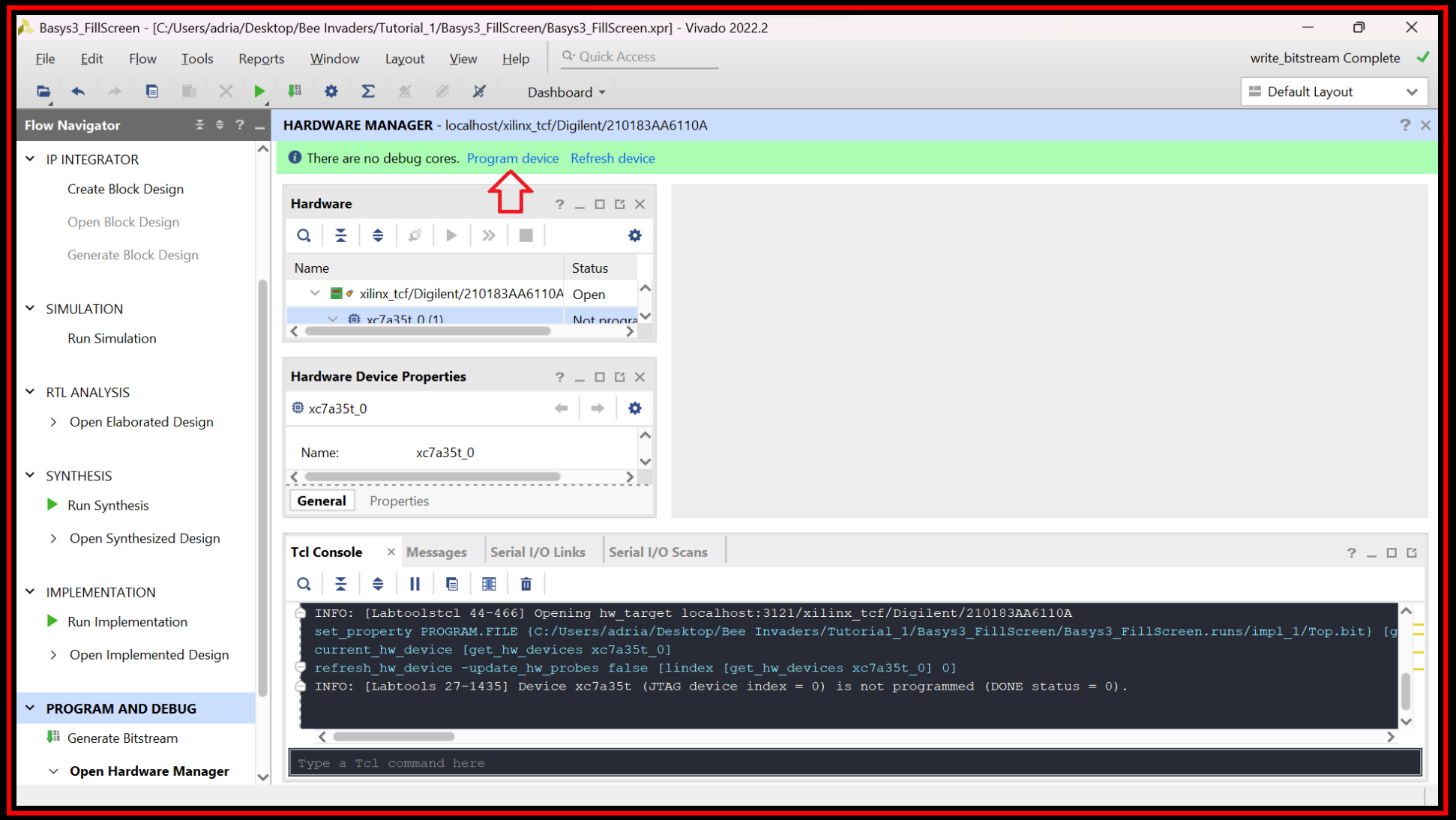
Now select "Open Hardware Manager" and click "OK"



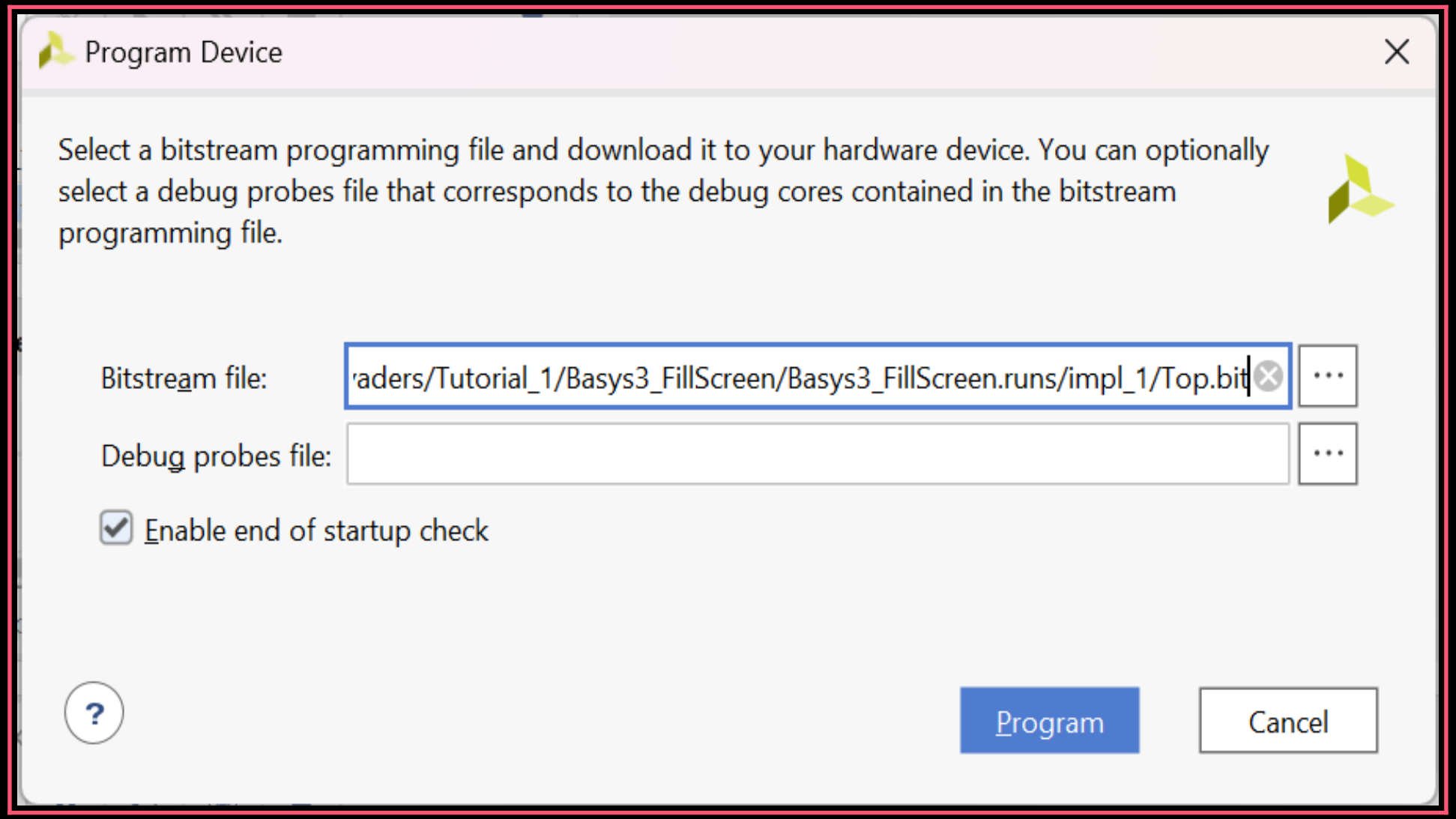
35 Next click "Open Target" and select "Auto Connect"



36 Now click "Program Device"



37 When the "Program Device" box appears make sure the "Bitsteam file" path is correct and then click "Program"



38 You should see an orange/brown coloured screen on your VGA monitor

(C) EXPLANATION OF HOW THE VGA SIGNAL WORKS

VGA is an analogue video standard using a 15-pin D-sub connector

It doesn't require high clock speeds or complex encoding,
so is an excellent place to begin when learning about FPGA graphics

VGA has five main signal pins:
one for each of red, green and blue signals and two for sync

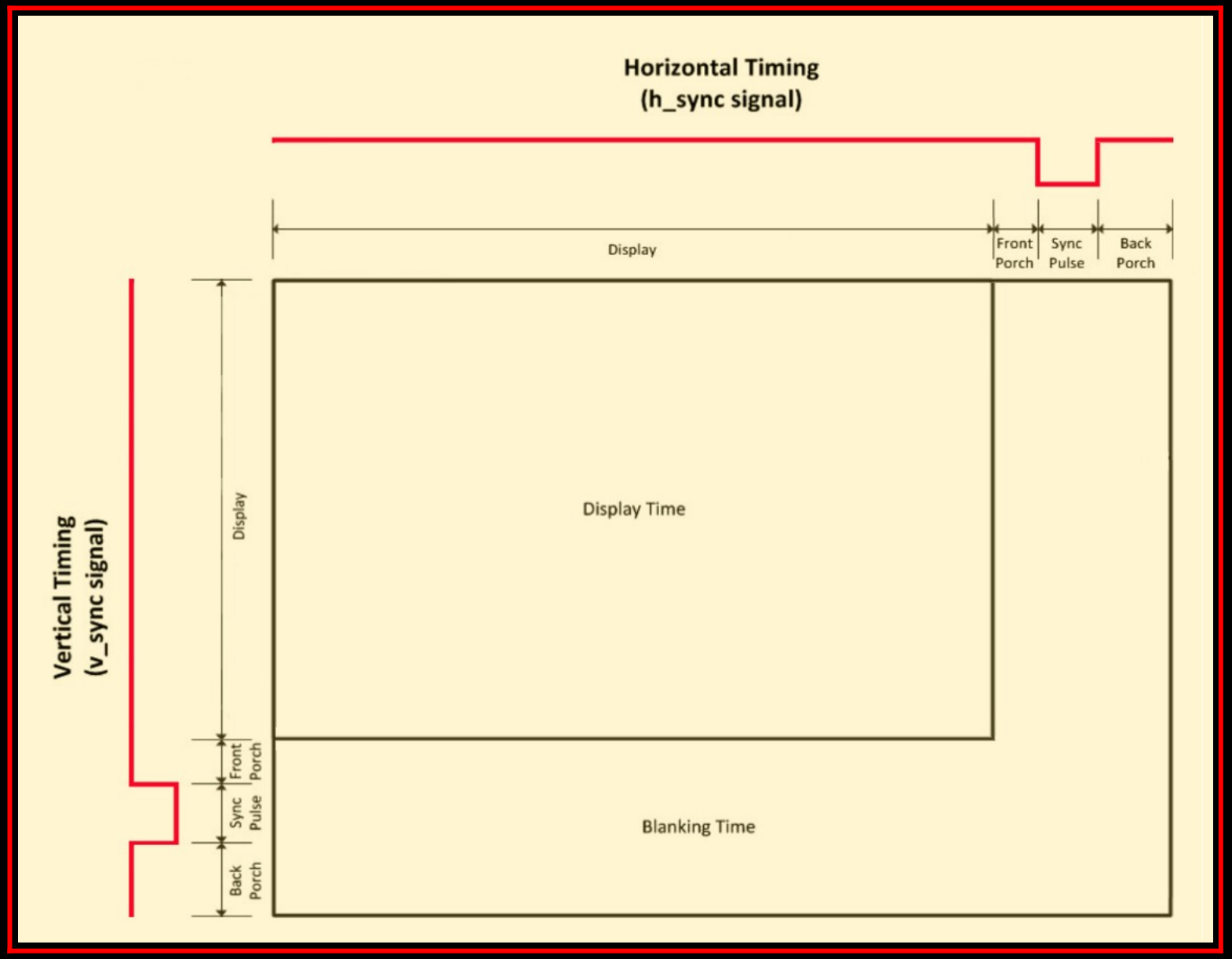
Horizontal sync separates a line and Vertical sync separates a screen, also known as a frame

VGA signals have two phases:
drawing pixels and the blanking interval.

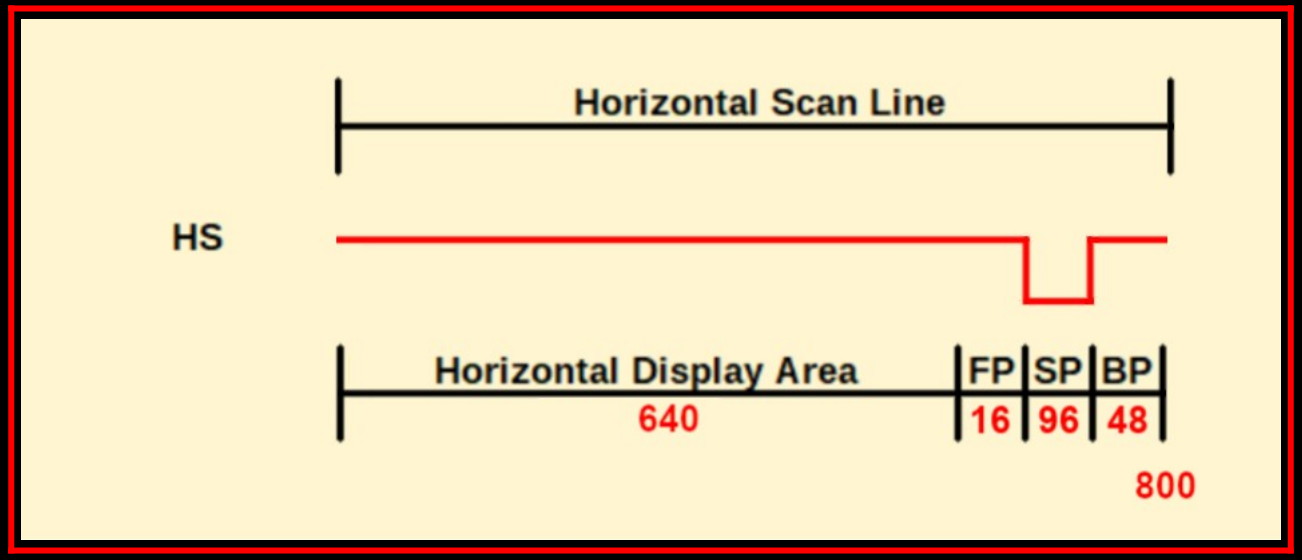
The sync signals occur within blanking intervals;
separated from pixel drawing by the front porch and back porch.

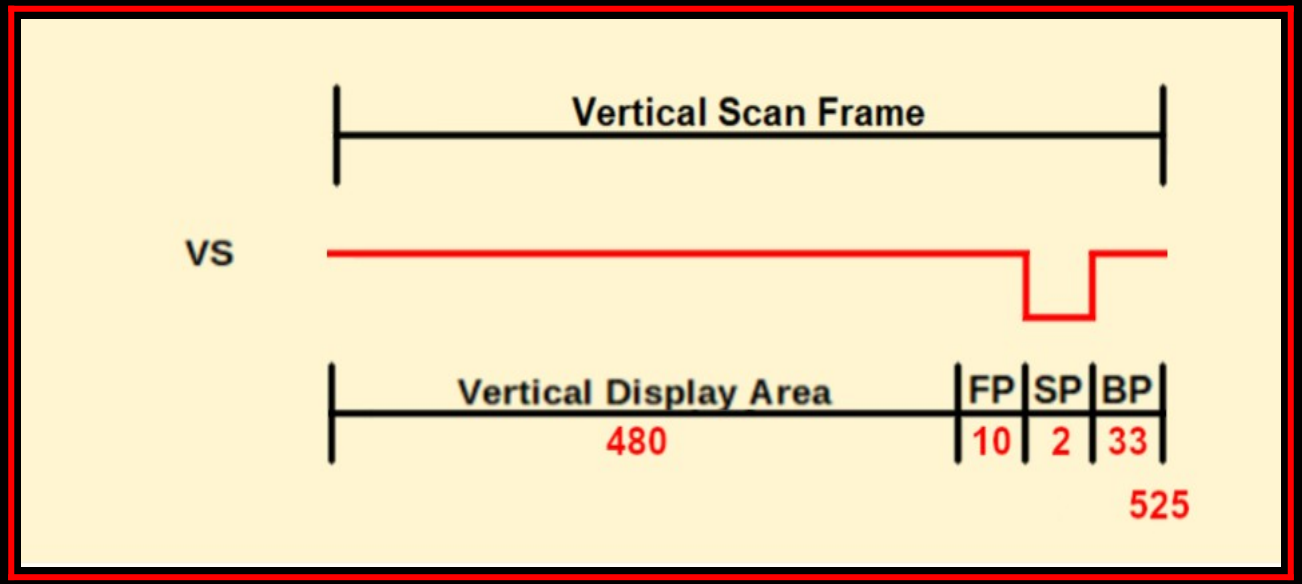
When VGA was developed monitors were based on cathode ray tubes (CRTs):
the blanking interval gave time for voltage levels to stabilise and for the electron gun
to return to the start of a line or screen

The below diagram shows the horizontal and vertical timings for a VGA signal



Recommended: Display Area, Front Porch (FP), Sync Pulse (SP) and Back Porch (BP) for the horizontal / vertical timings for a 640 x 480 @ 60 Hz VGA video mode





General timing:	Horizontal refresh rate:			31.5kHz or (1 / 31.746031746031 75µs)		
	Vertical refresh rate:			60Hz or (1 / 16.666666666666 67ms)		
	Pixel clock:			25.2MHz (800 x 525 x 60 = 25,200,000)		
Horizontal:	Scanline part	Pixels		1 / 25.2MHz Pixel Clock		Time (µs)
	Visible area	640				
	Front porch	16				
	Sync pulse	96				
	Back porch	48				

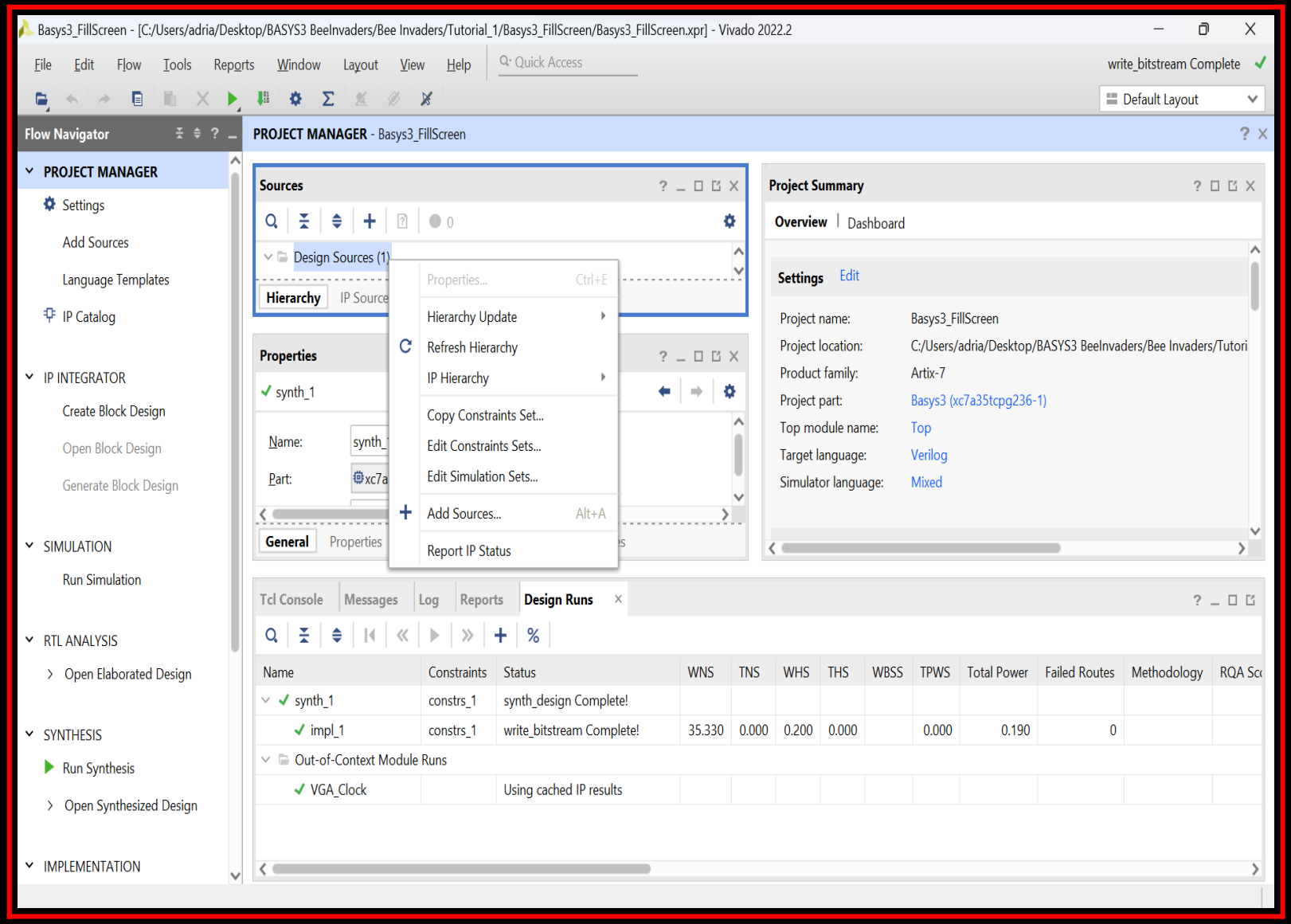
	Whole line	800	x	0.039682539682539 7µs	=	31.746031746031 75µs
Vertical:	Frame part	Lines		1 / 31.5kHz		Time (ms)
	Visible area	480				
	Front porch	10				
	Sync pulse	2				
	Back porch	33				

	Whole frame	525	x	0.031746031746031 7ms	=	16.666666666666 67ms

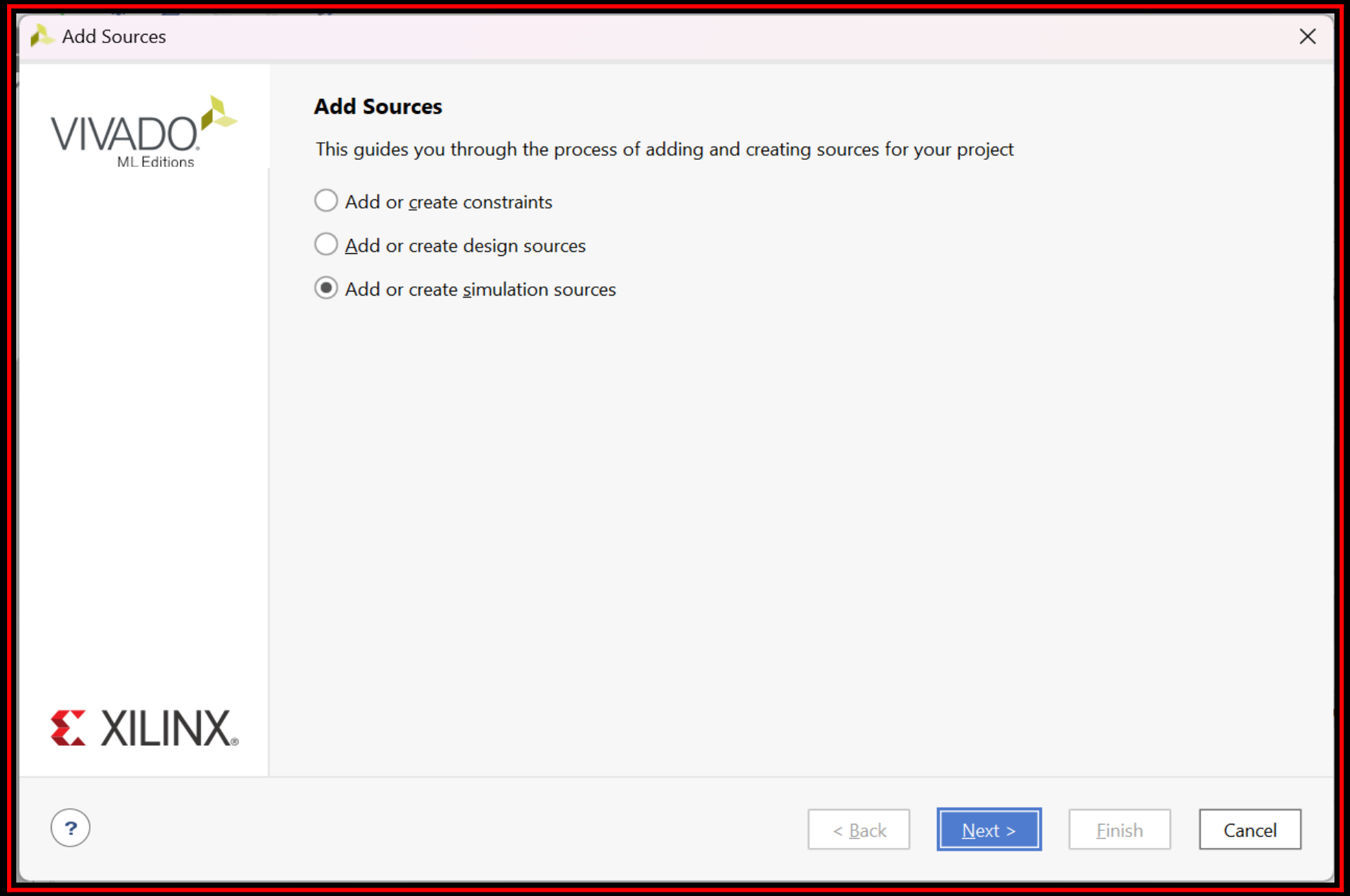
The precise Pixel Clock = 800 whole line pixels * 525 vertical lines * 60 Hz (Vertical refresh rate) = 25,200,000 cycles per second or 25.2MHz.

(D) TESTBENCHES

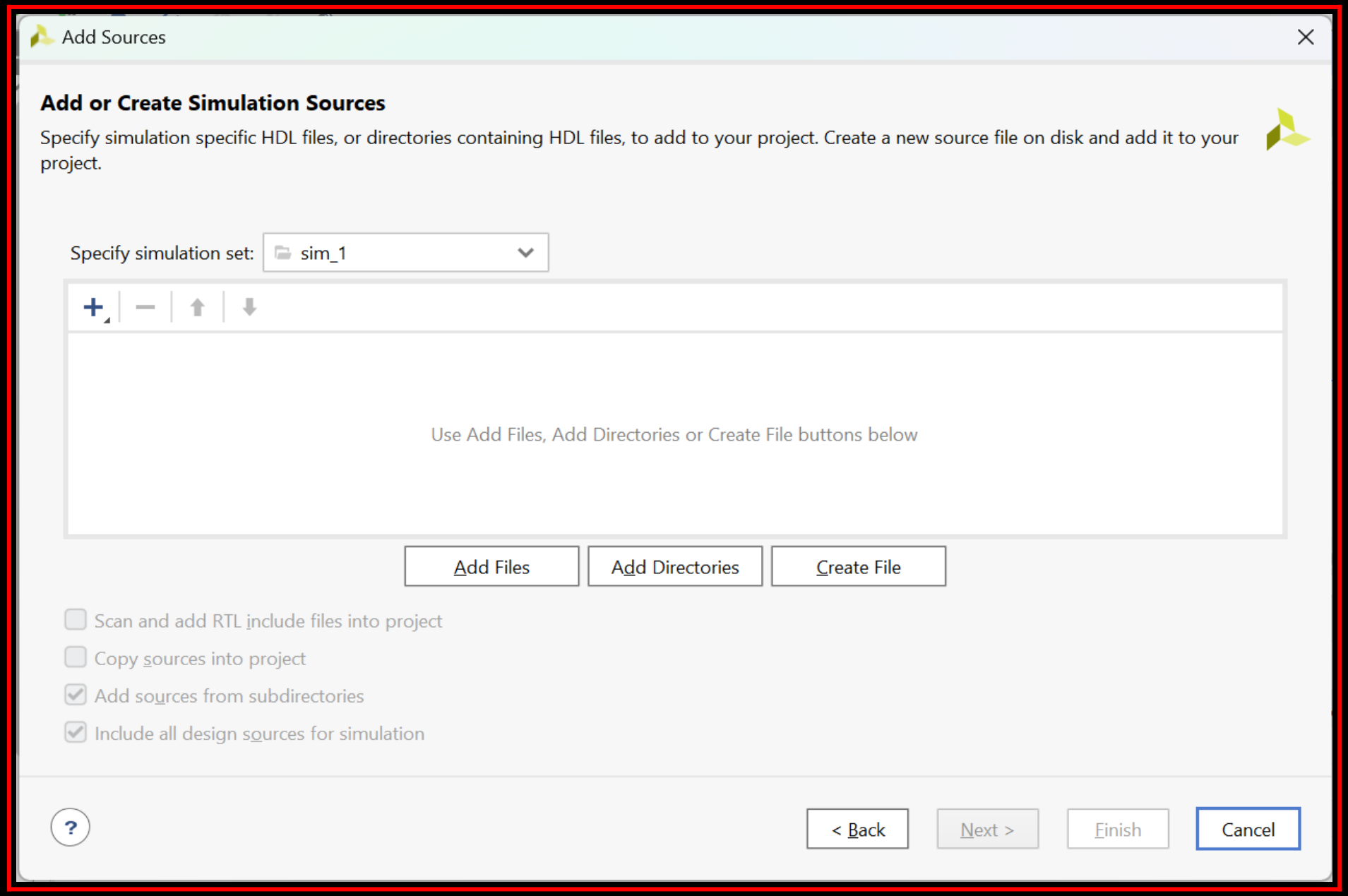
01 Right click on "Design Sources" and left click on "Add Sources" as shown below



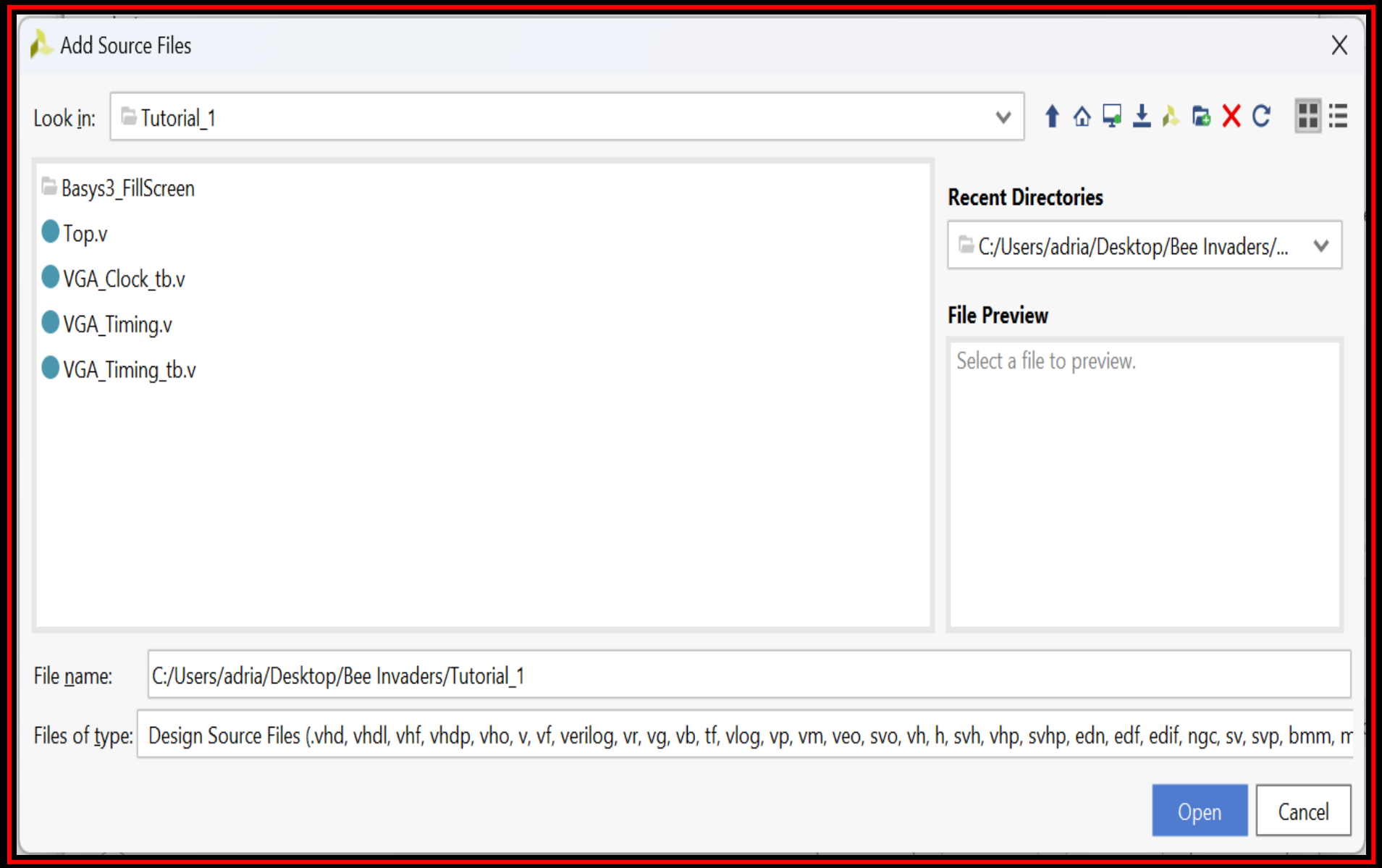
02 Select "Add or create simulation sources" and click "Next"



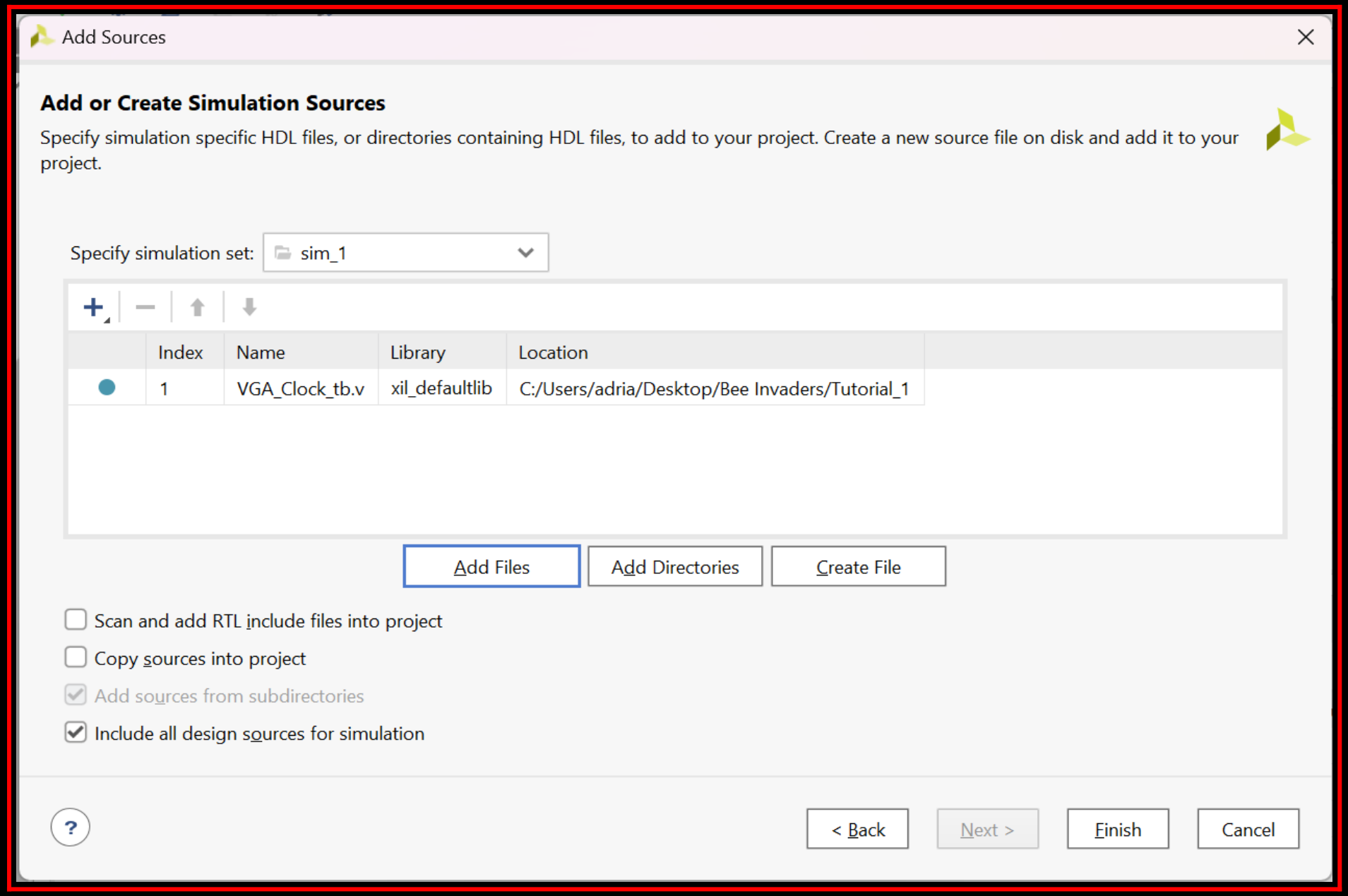
03 Click on "Add Files" button



04 Select the "VGA_Clock_tb.v" file, ensuring you are looking in the path "Bee Invaders/Tutorial_1" and click "OK"

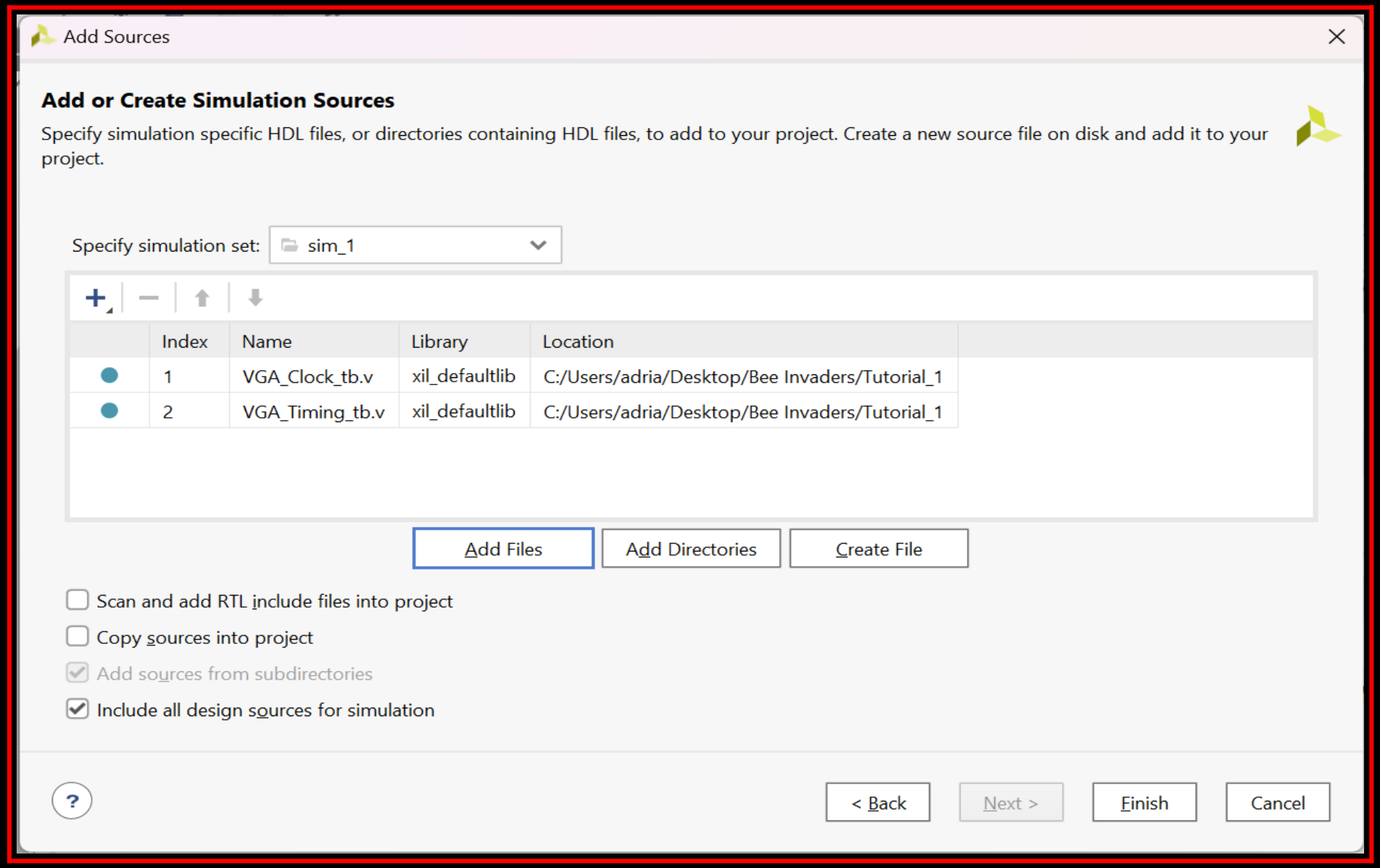


05 Click on "Add Files" button

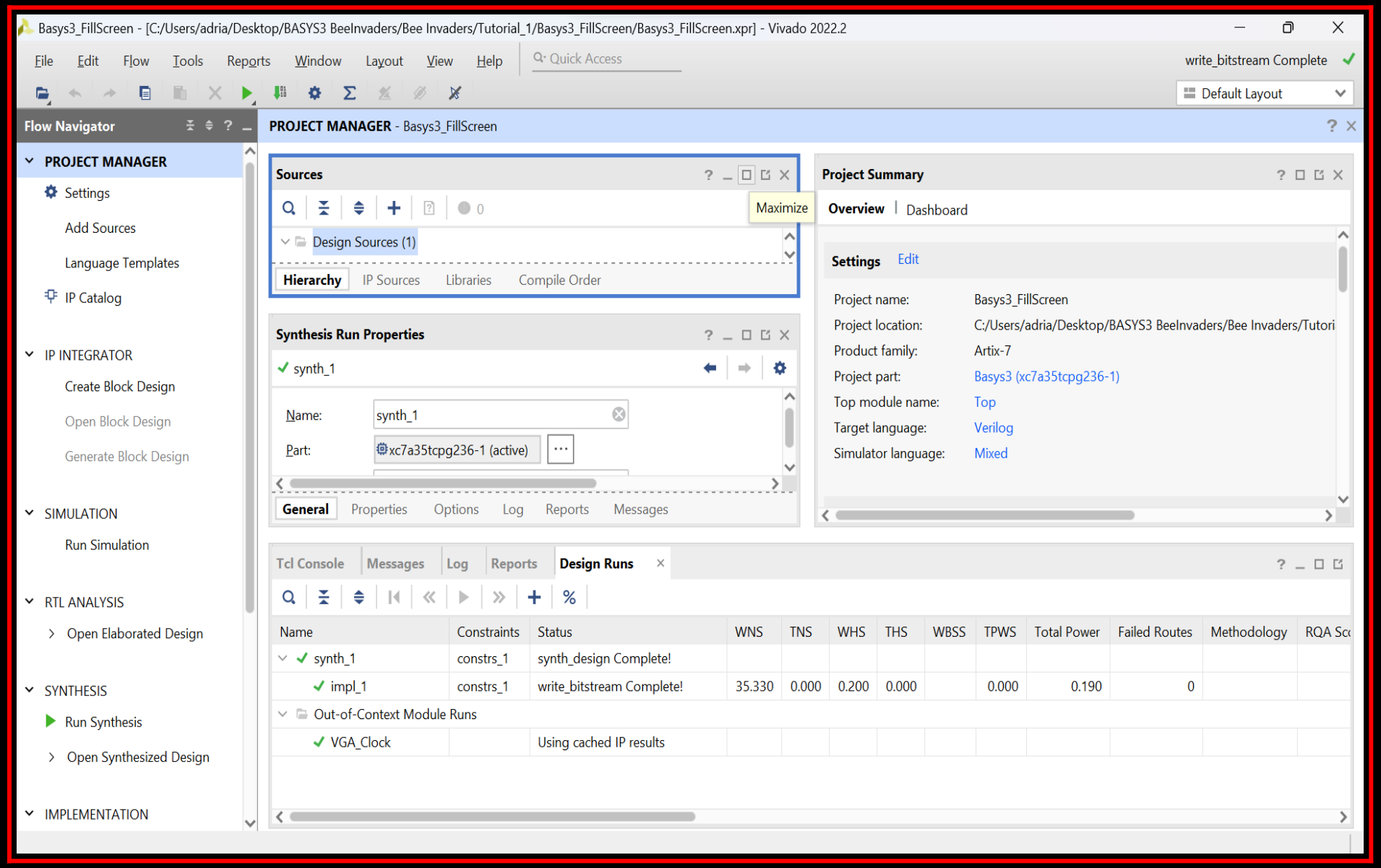


06 Select the "VGA_Timing_tb.v" file and click "OK"

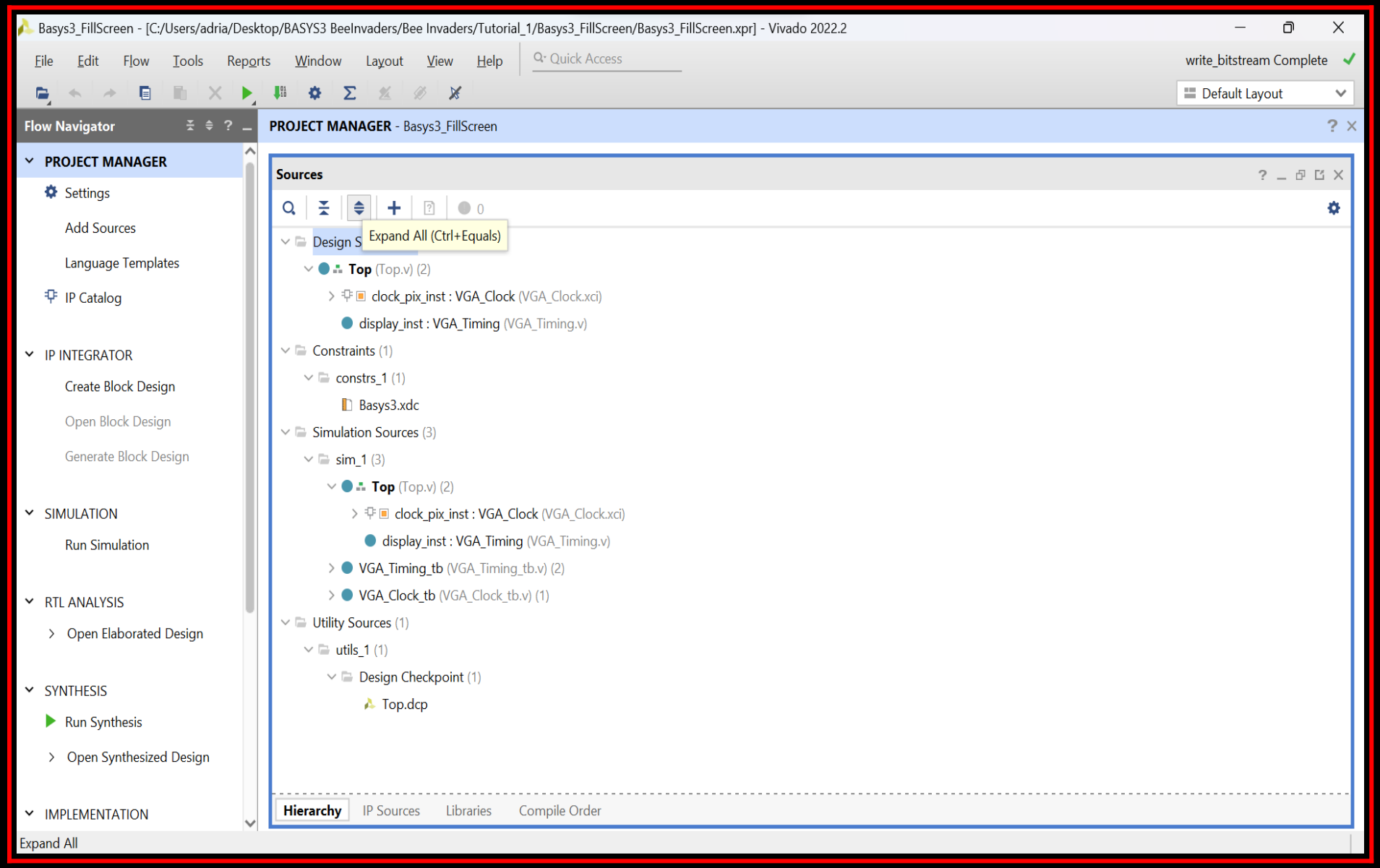
07 At the next screen select "Finish"



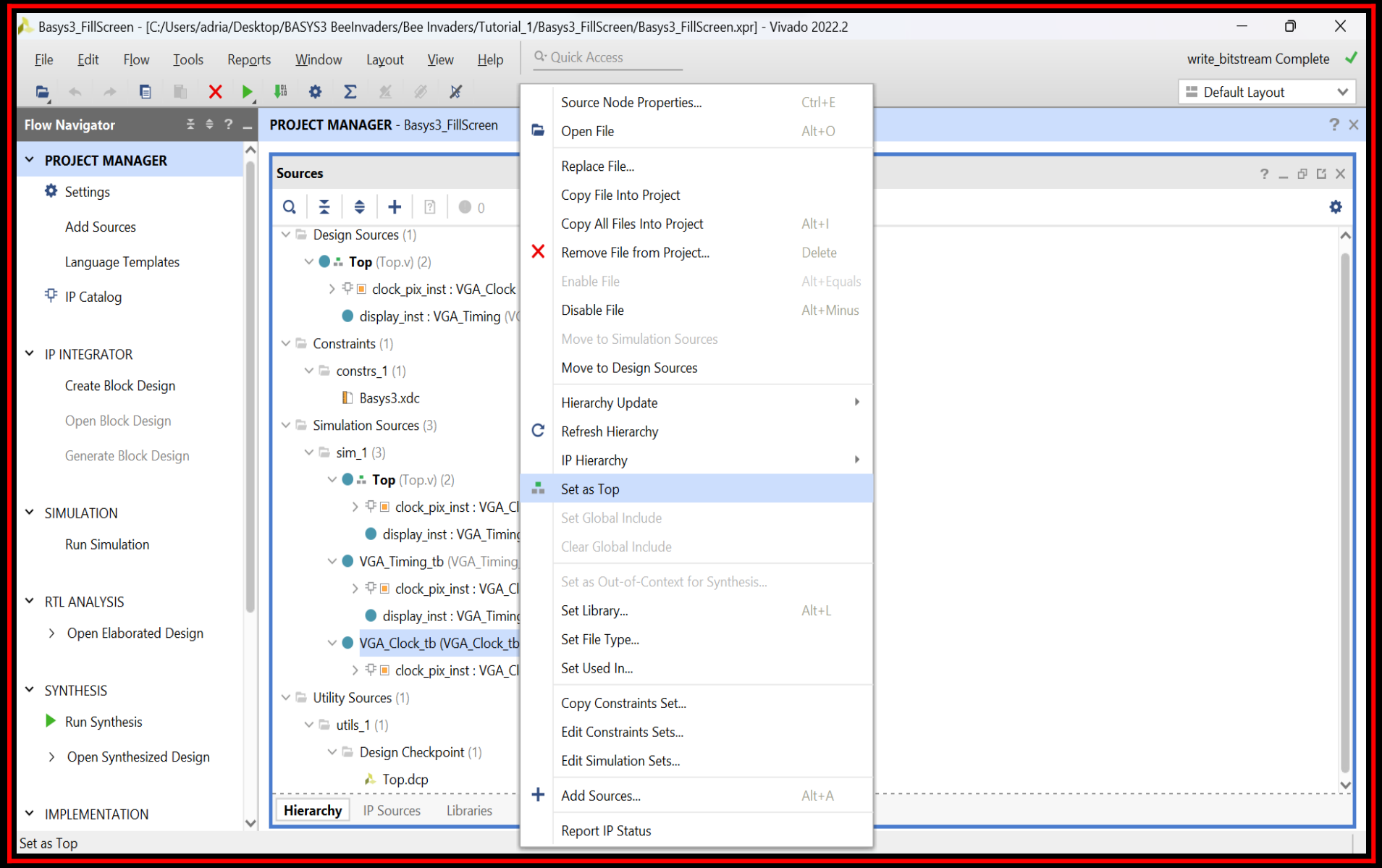
08 Click on the "Maximise" button in the "Sources" window



09 Click on the "Expand All" button



10 Right click on the file "VGA_Clock_tb.v" and select "Set as Top"



THE CODE

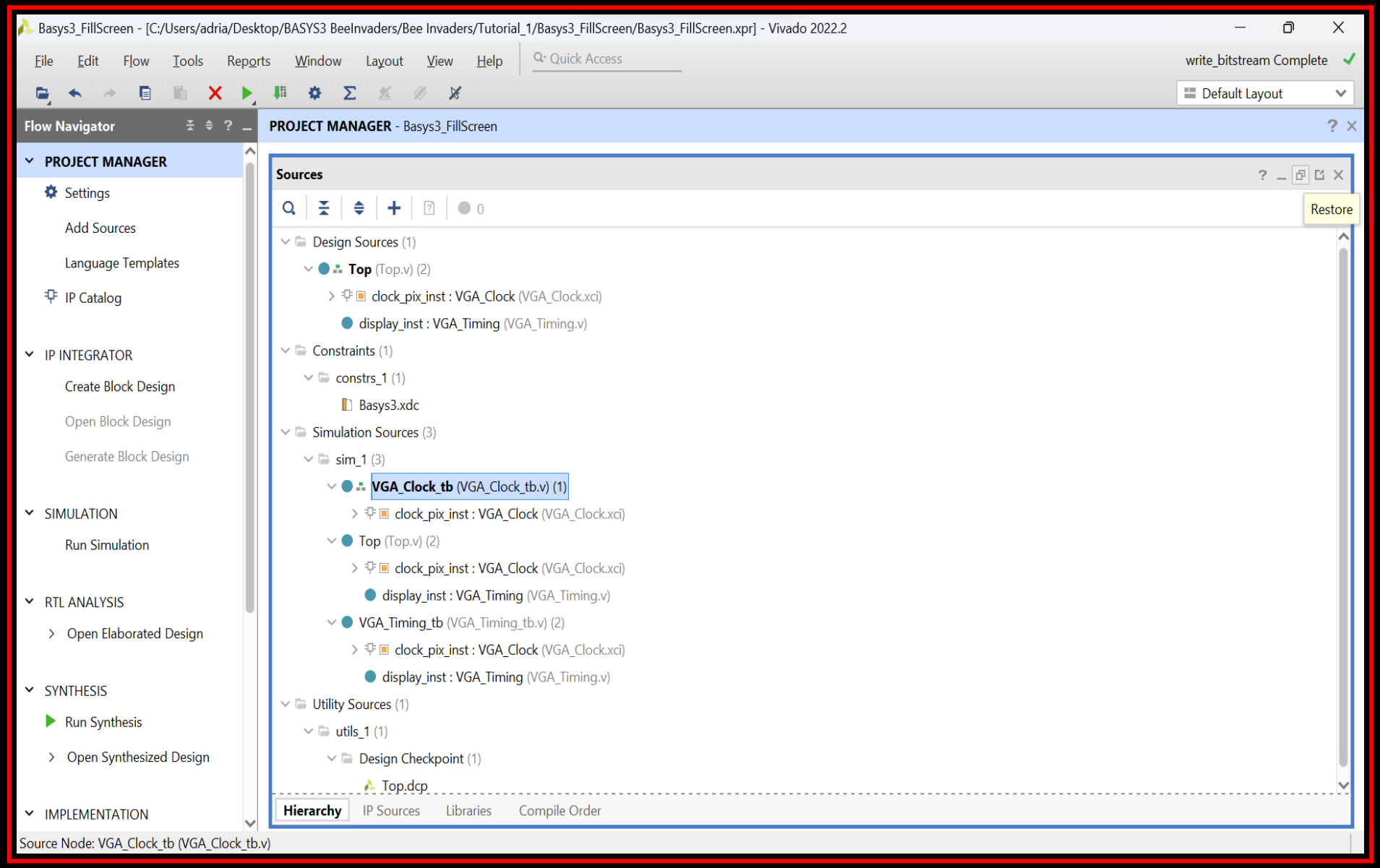
This is the code from the file "VGA_Clock_tb.v"

```
//-----------------------------------------------
// VGA_Clock_tb.v module
// Digilent Basys 3
// Bee Invaders Tutorial_1
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//-----------------------------------------------

`timescale 1ns / 1ps

module VGA_Clock_tb();

parameter CLK_PERIOD = 10;

   reg reset;
   reg clk_100m;

   // 640x480p60 clocks
   wire clk_pix;
   wire clk_pix_locked;

   VGA_Clock clock_pix_inst (
      .clk_100m(clk_100m),
      .reset(reset),
      .clk_pix(clk_pix),
      .clk_pix_locked(clk_pix_locked)
   );

   // generate clock
   always #(CLK_PERIOD / 2) clk_100m = ~clk_100m;


   initial
   begin
      clk_100m = 1;
      reset = 1;
      #100
      reset = 0;

      #19000
      $finish;
   end
endmodule
```

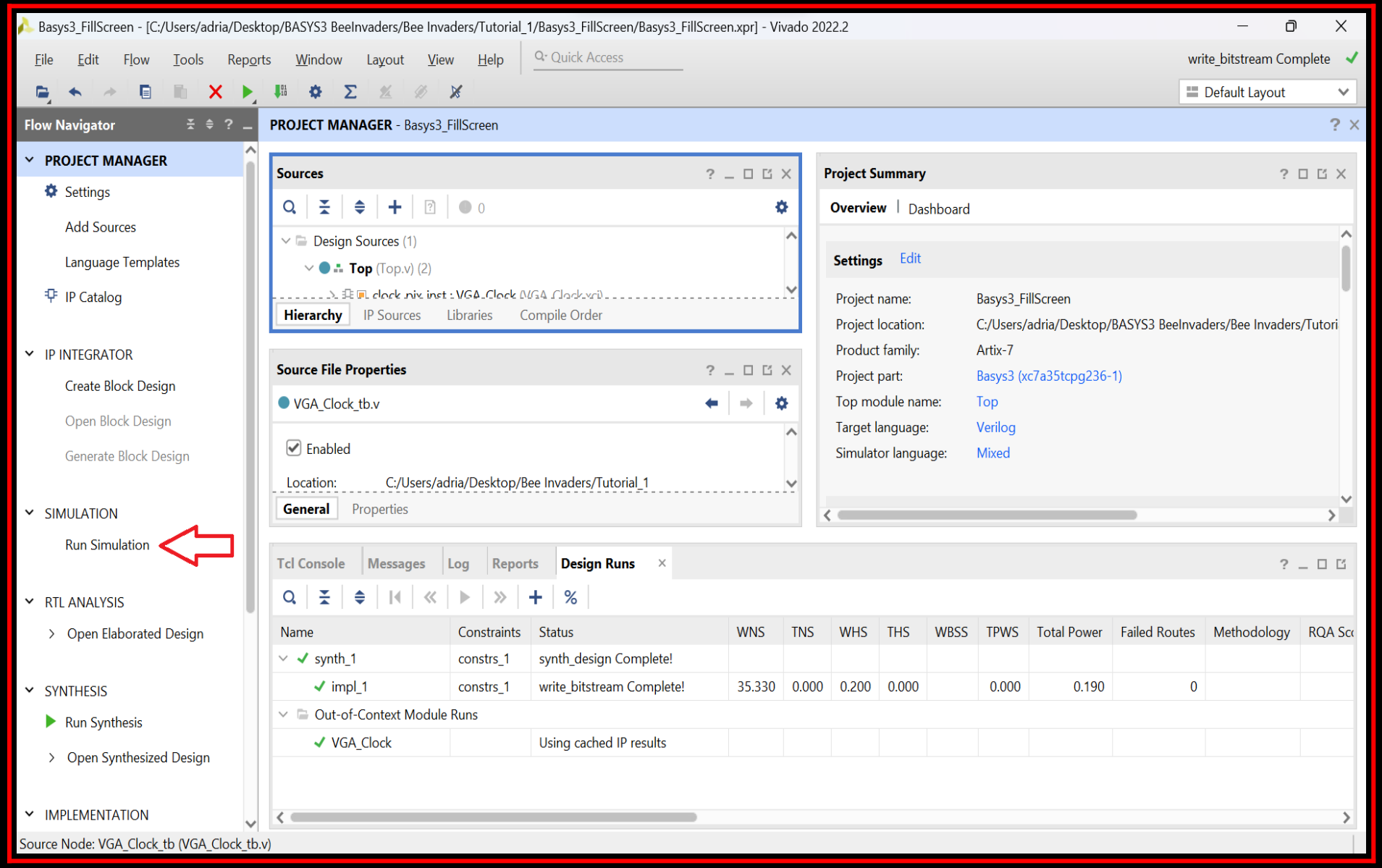
This is the code from the file "VGA_Timing_tb.v"

```
//-----------------------------------------------
// VGA_Timing_tb.v module
// Digilent Basys 3
// Bee Invaders Tutorial_1
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//-----------------------------------------------

`timescale 1ns / 1ps

module VGA_Timing_tb();

   parameter CLK_PERIOD = 10;  // 10 ns == 100 MHz
   parameter CORDW = 10;  // screen coordinate width in bits

   reg reset;
   reg clk_100m;
   wire clk_pix;
   wire clk_pix_locked;

   VGA_Clock clock_pix_inst (
      .clk_100m(clk_100m),
      .reset(reset),
      .clk_pix(clk_pix),
      .clk_pix_locked(clk_pix_locked)
   );

   reg rst_pix;
   wire [9:0] sx;
   wire [9:0] sy;
   wire hsync;
   wire vsync;
   wire de;

   VGA_Timing display_inst (
      .clk_pix(clk_pix),
      .rst_pix(!clk_pix_locked),
      .sx(sx),
      .sy(sy),
      .hsync(hsync),
      .vsync(vsync),
      .de(de)
   );

   // generate clock
   always #(CLK_PERIOD / 2) clk_100m = ~clk_100m;

   initial
   begin
      clk_100m = 1;
      reset = 1;
      #100
      reset = 0;

      #20_000_000;
      $finish;
   end
endmodule
```

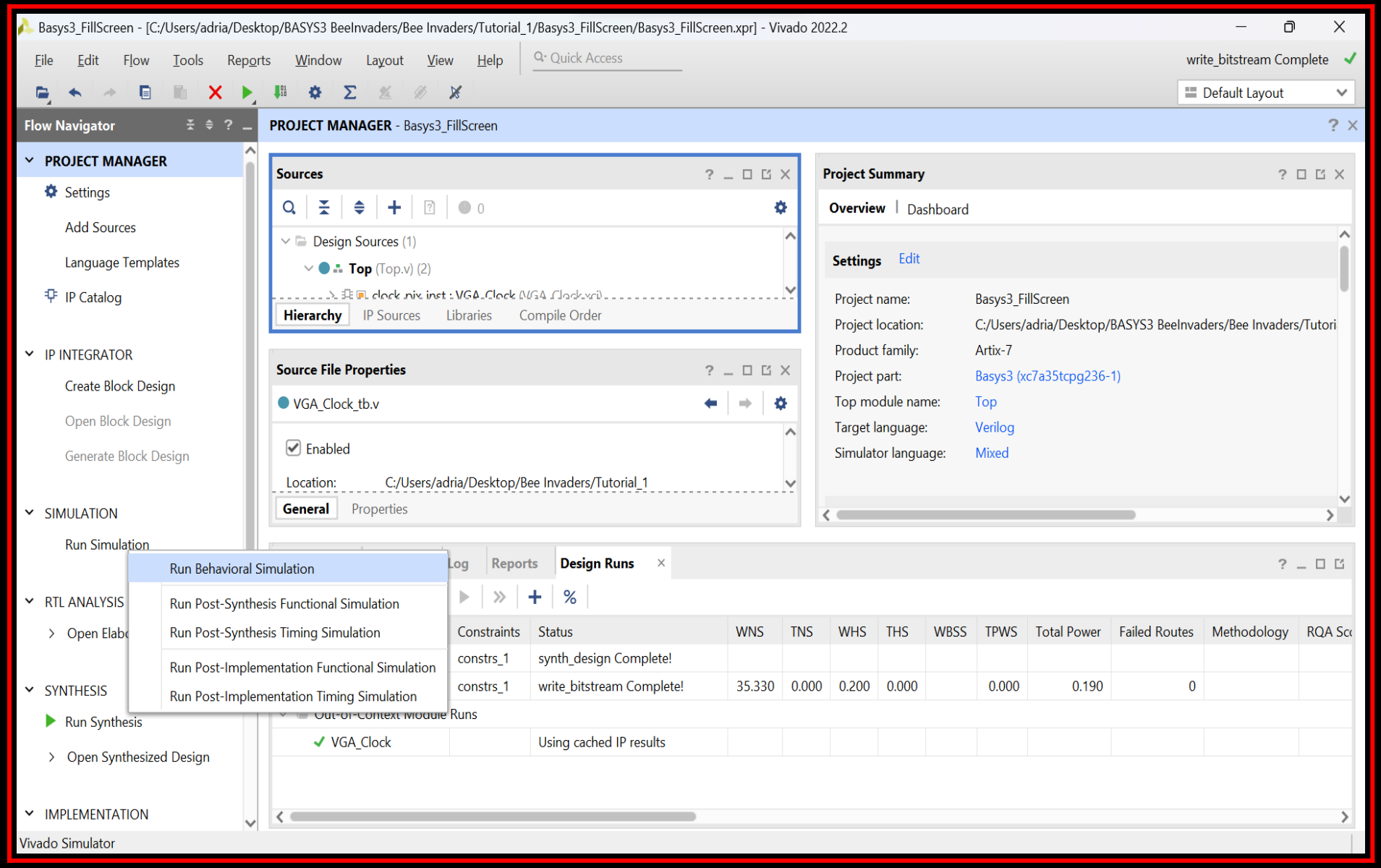
11 Click on the "Restore" button



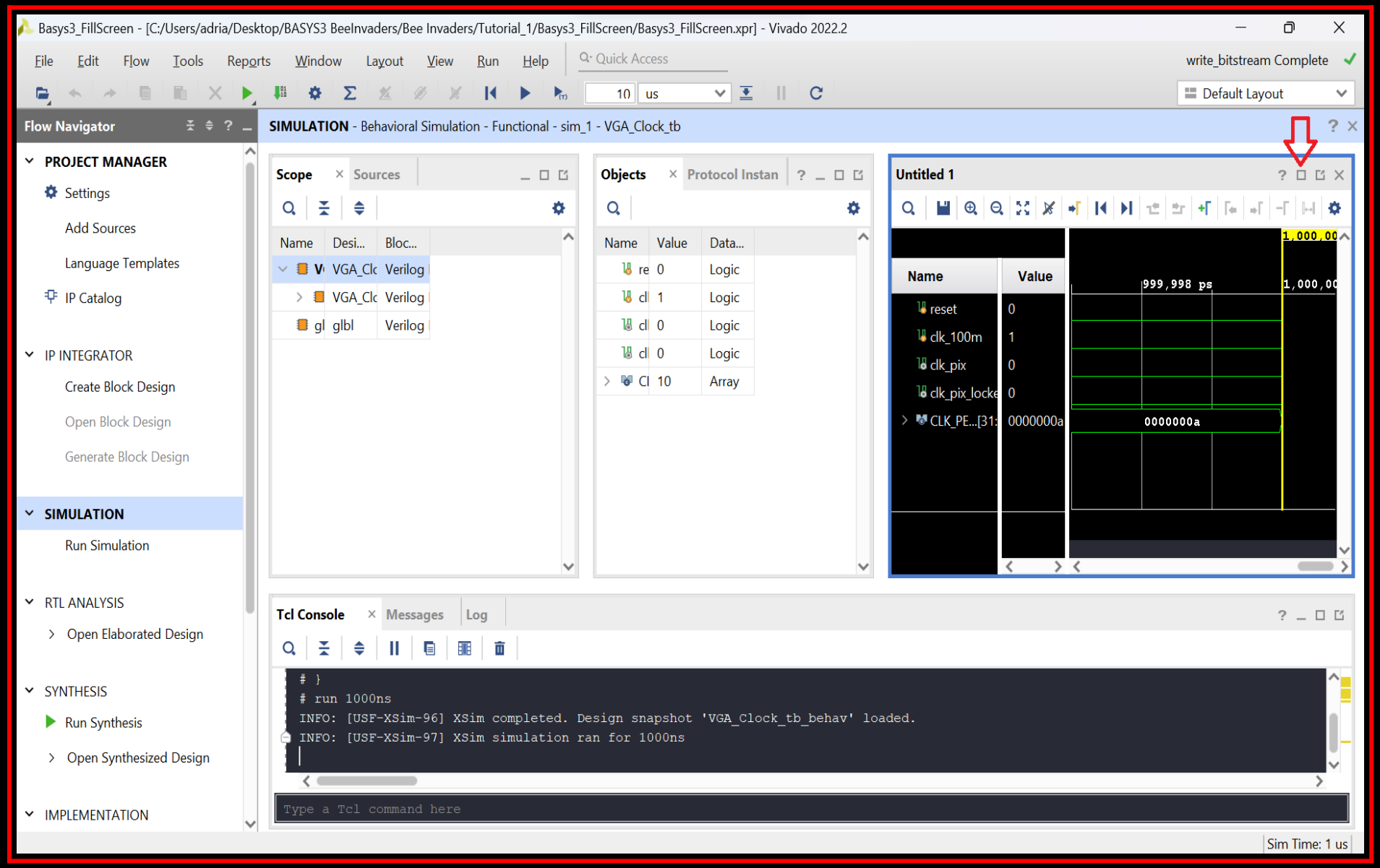
12 Click on "Run Simulation" in the "Flow Navigator" window



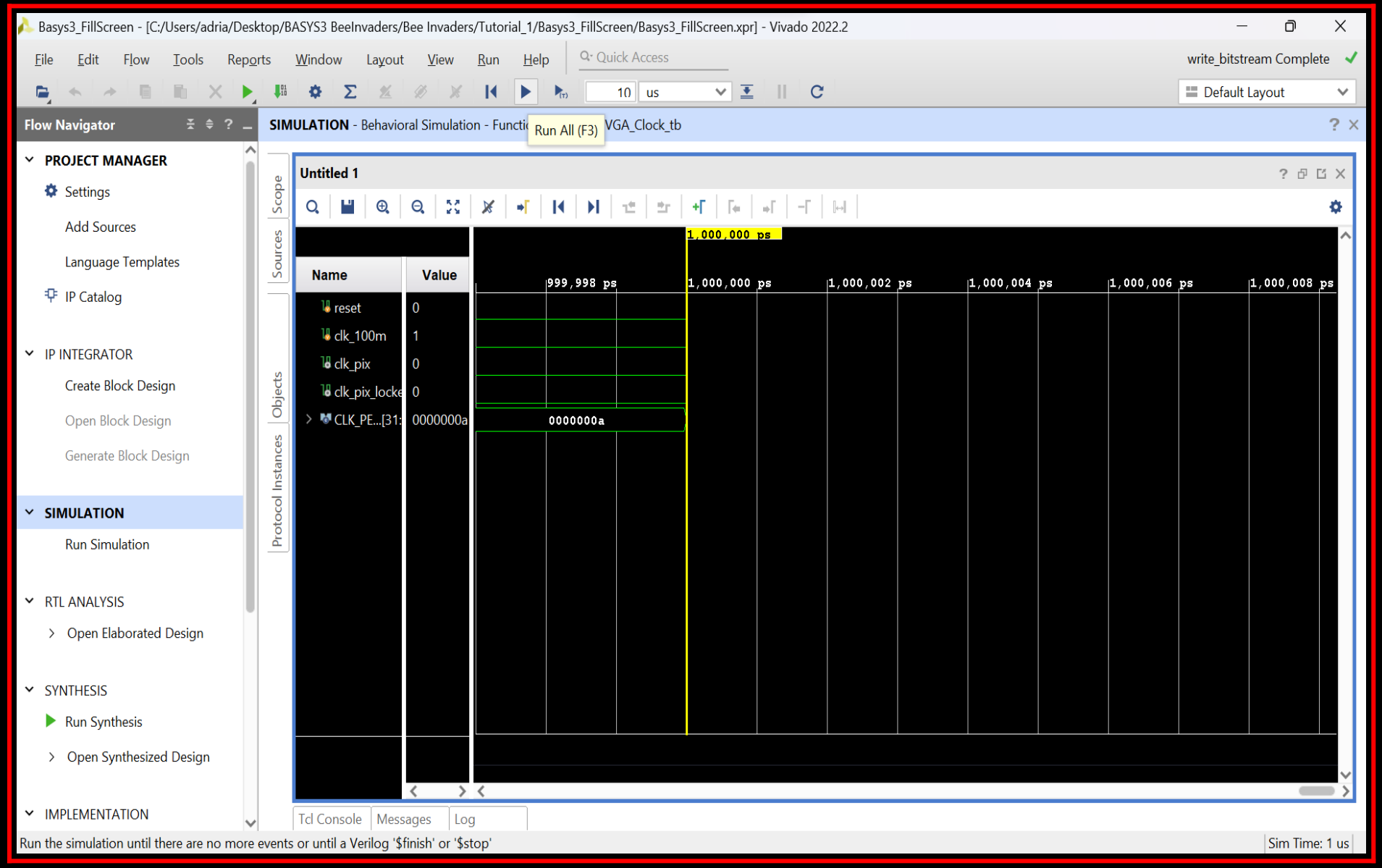
13 Click on "Run Behavioral Simulation". Note, some virus software require you to add an exception to the "Bee Invaders" folder



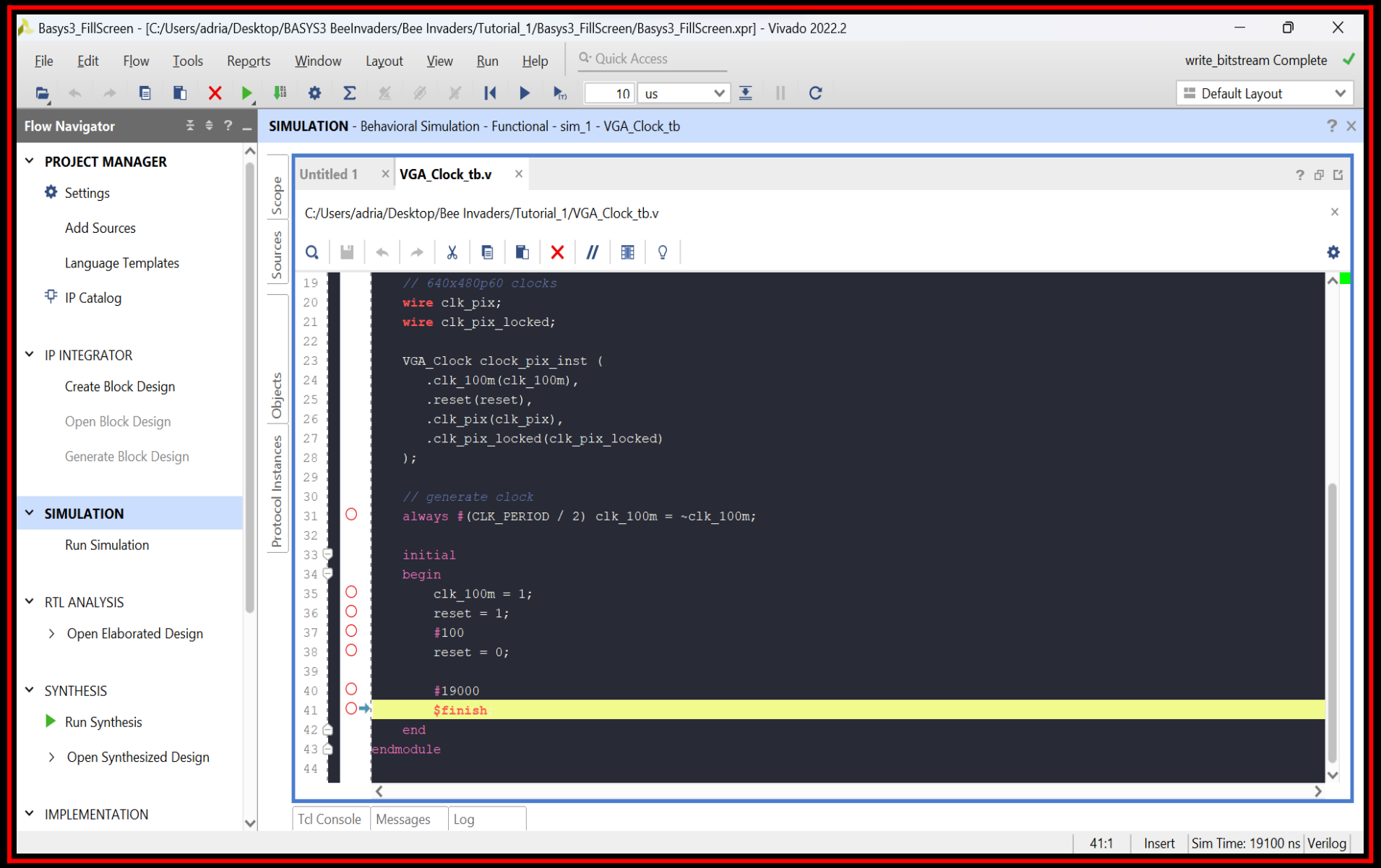
14 Click on "Maximise" button in the "Simulation" window



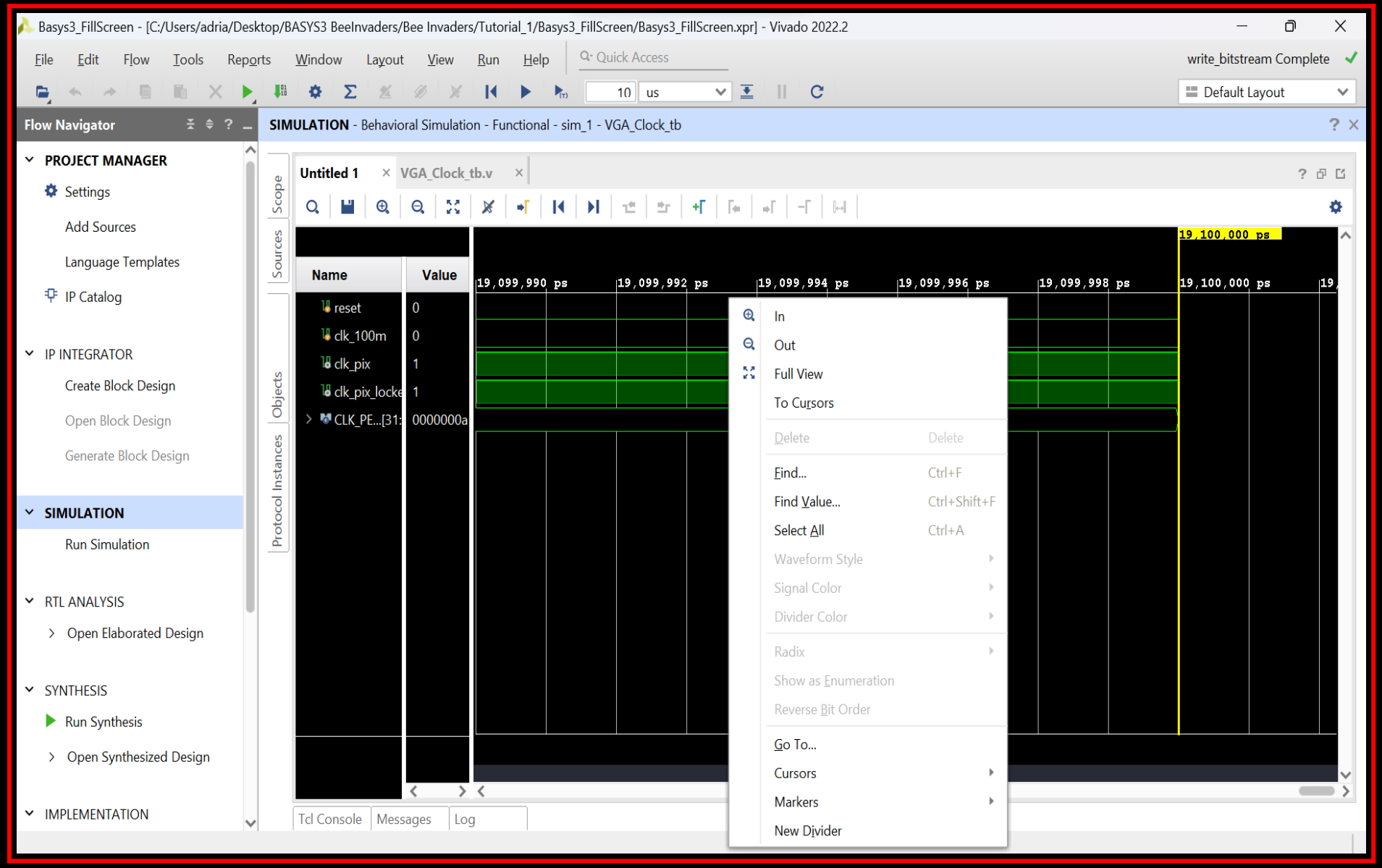
15 Click on the "Run All" button



16 You may have to click on the "Untitled 1" tab to get back to the Simulator window

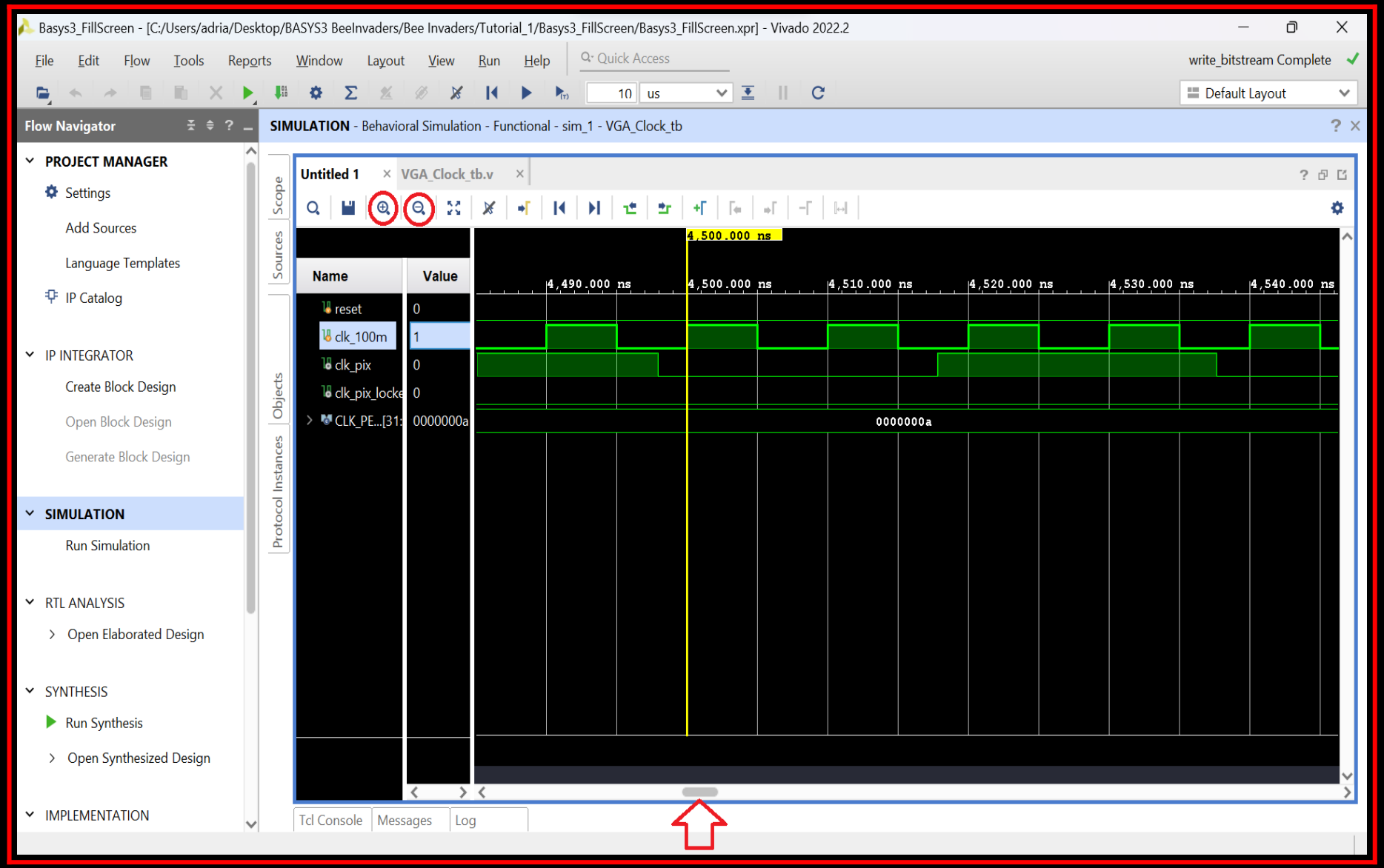


17 Right click in the Simulation window and select "Full View"

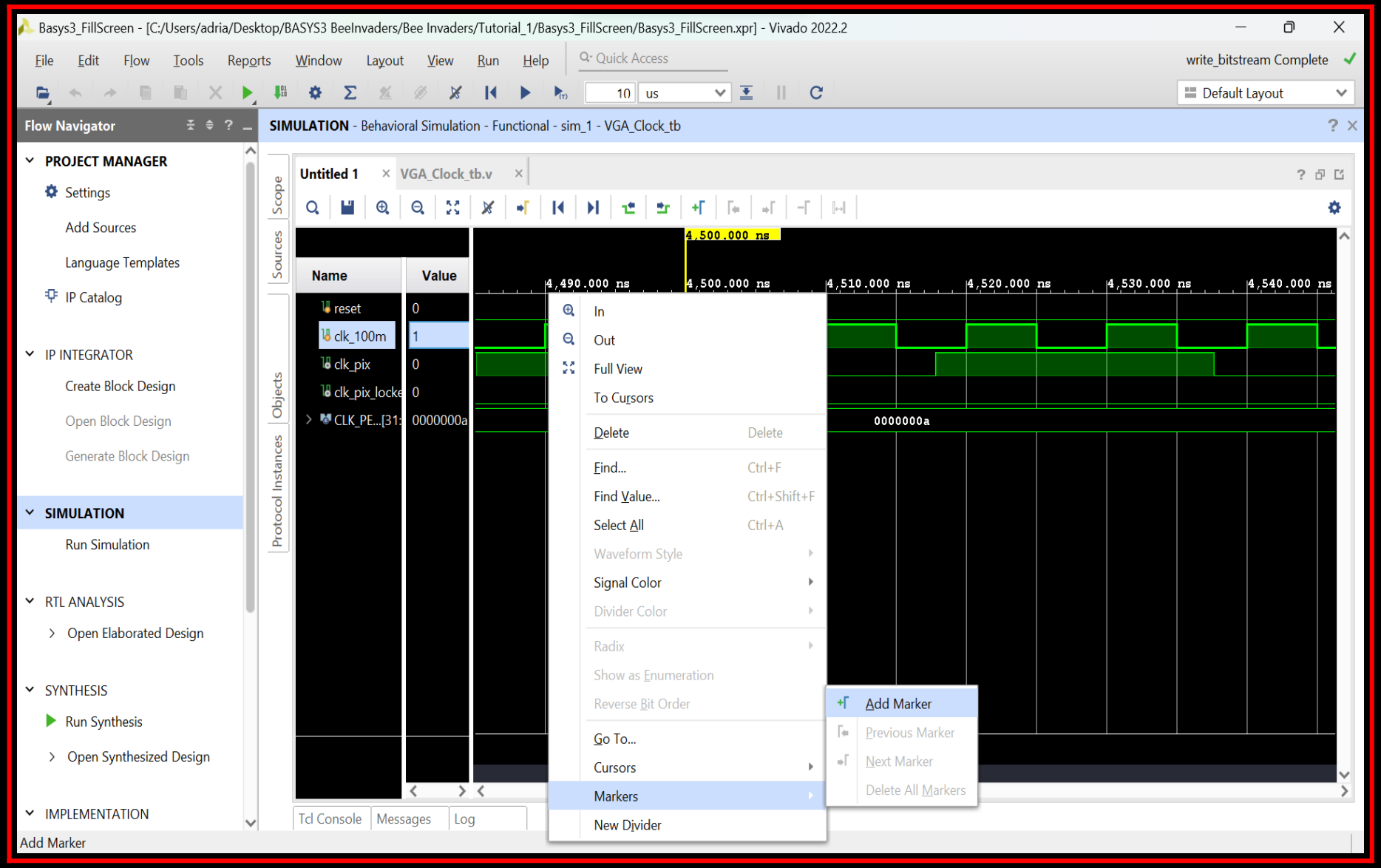


18

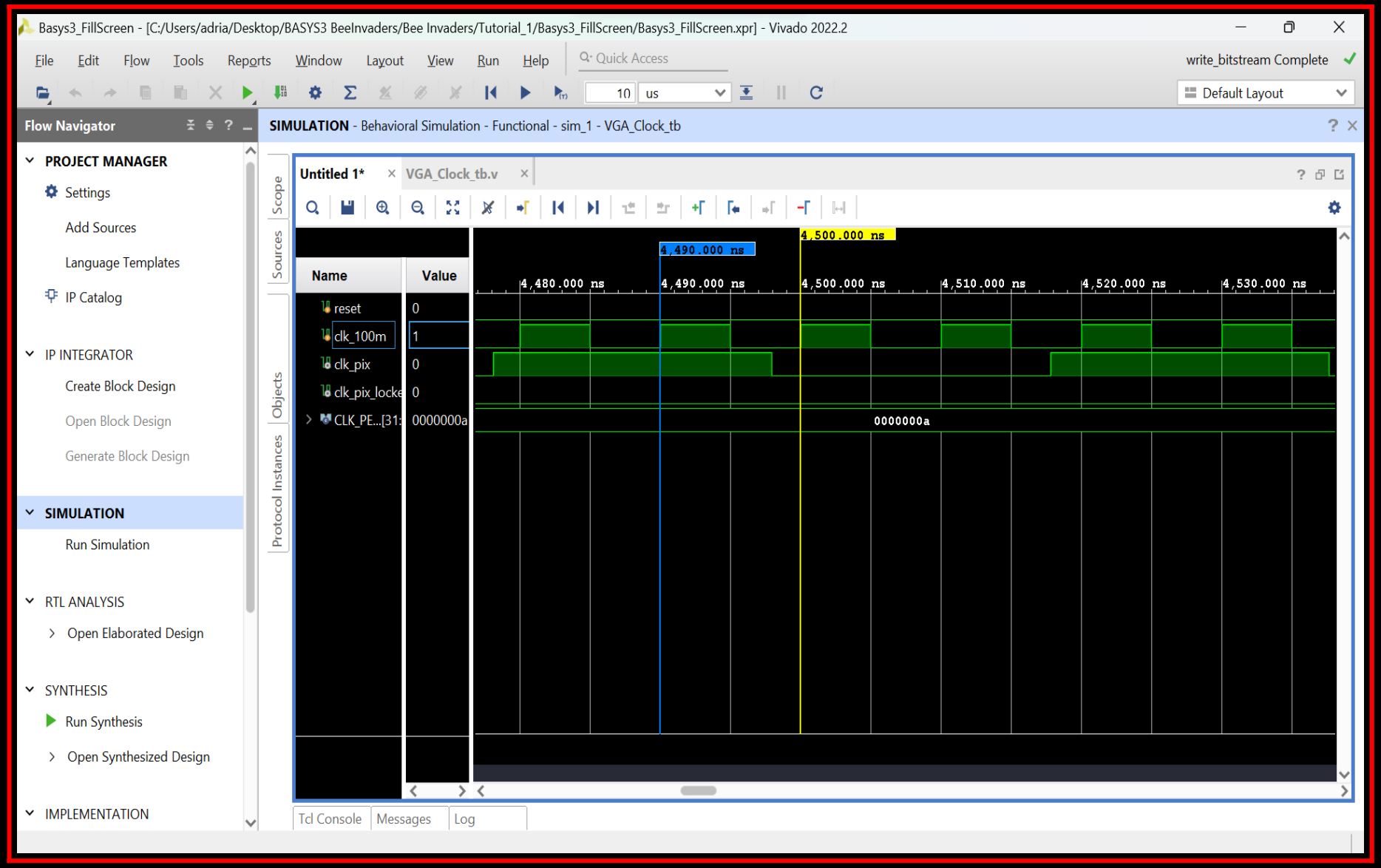
Use the "Zoom" buttons (highlighted with a RED circle) and the "Scroll Bar" (highlighted with a RED arrow) to position the "clk_100m" and "clk_pix" signals similar to the below. Left click where the "clk_100m" signal ends at "4,500.000 ns", this places a yellow marker at this point



19 Right click where the "clk_100m" signal starts, select "Markers" and select "Add Marker"

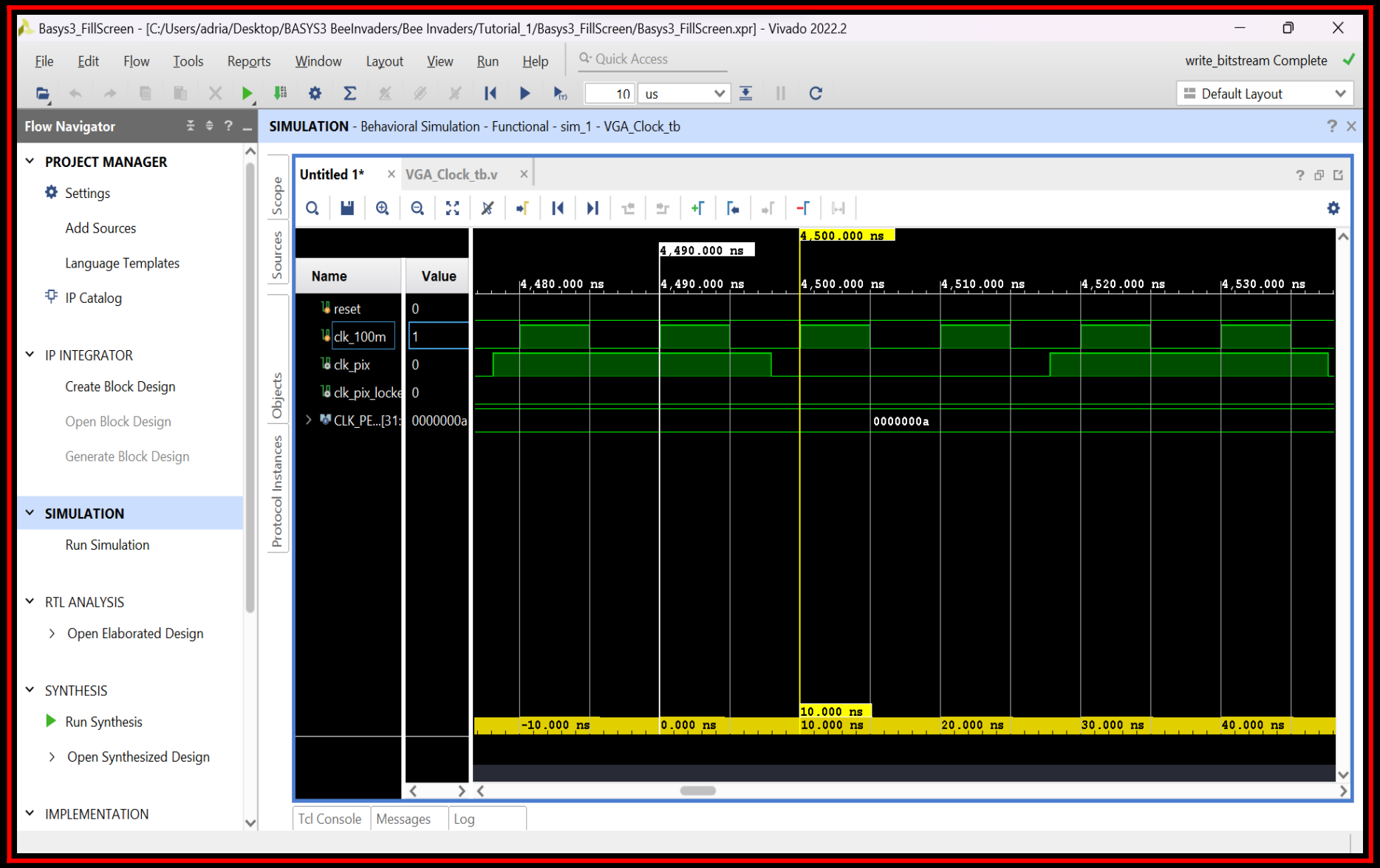


20 This places a Blue marker at the start of the "clk_100m" signal at "4,490.000 ns". You may find it easier to use the "Zoom" buttons to position the Blue marker (select the top of the Blue marker and drag it left or right accordingly)



21 Left click the top of the Blue marker and this displays the distance between both markers "10.000 ns"

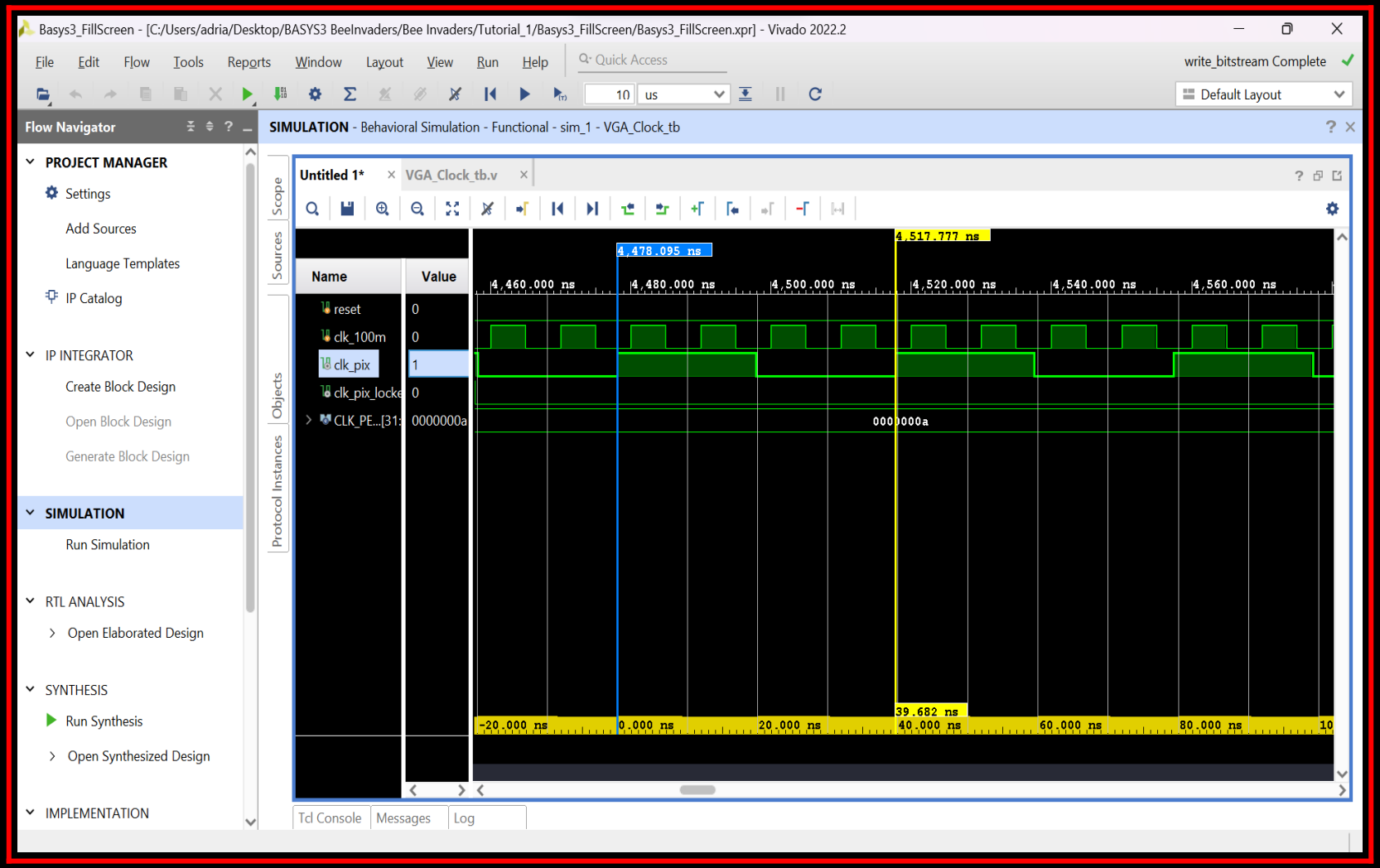
1/10ns = 0.1 = 100MHz (clock rate)



22

Do the same for the "clk_pix" (the 25.2MHz pixel clock) and this displays the distance between both markers "39.682 ns"

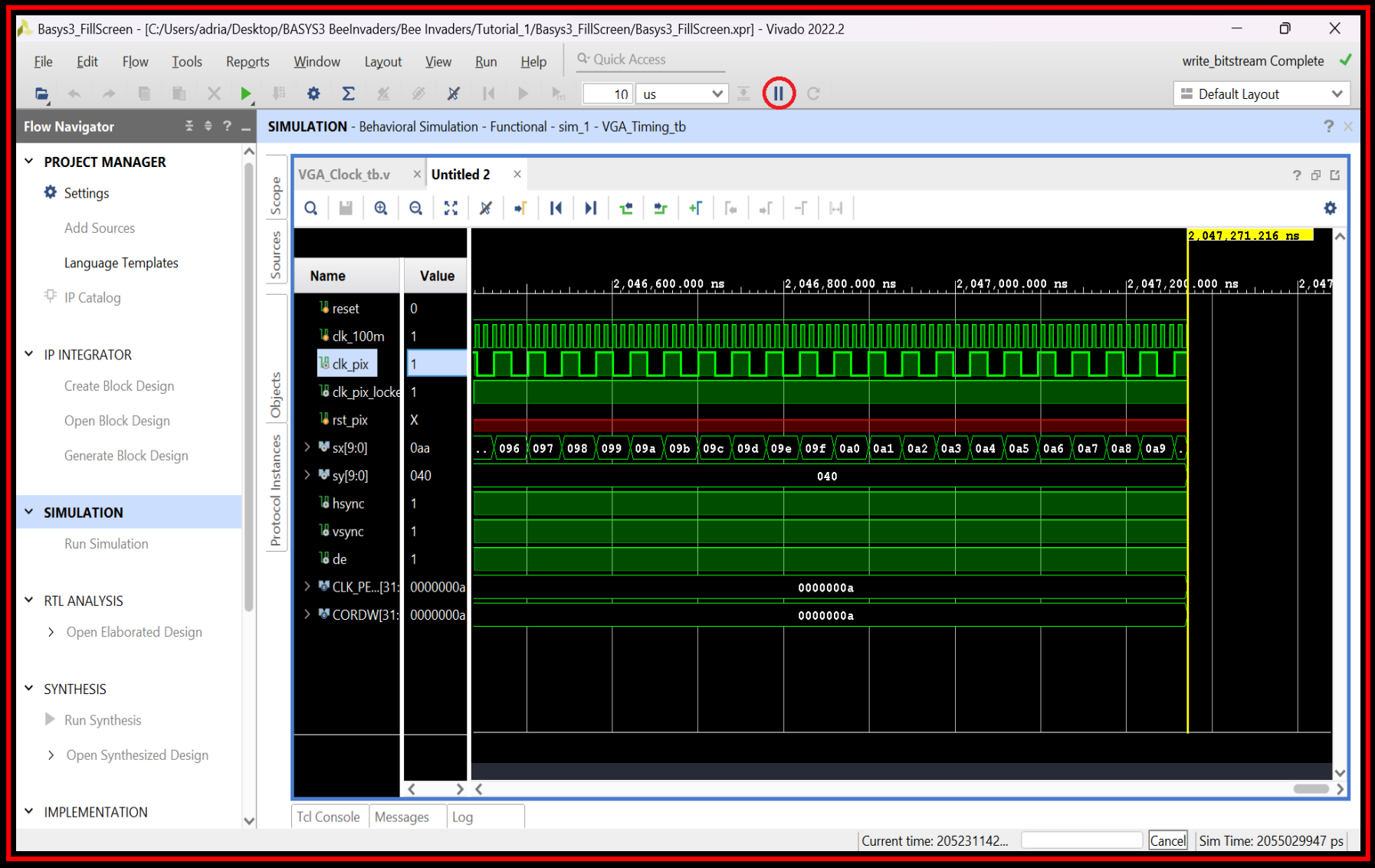
1/39.682ns = 0.0252003427246611 = 25.2003427246611MHz (Clock Pixel rate)
To be precise, it is actually 1/39.6825396825397 = 0.0252 = 25.2MHz (Clock Pixel rate) - See Page 53



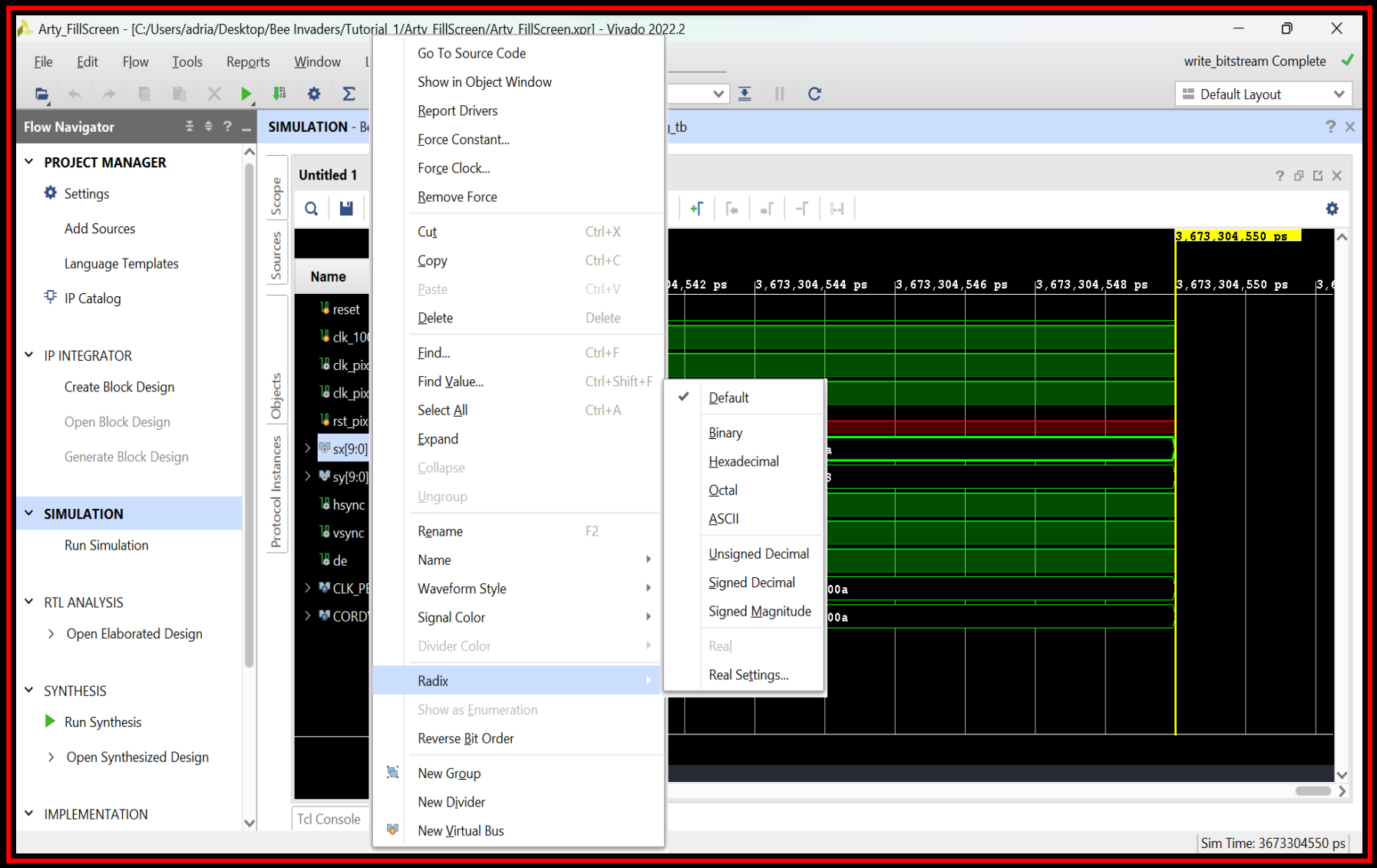
23

Close the "Simulation" window and repeat steps 08 to 15 but at step 10 right click on the file "VGA_Timing_tb.v" and select "Set as Top"

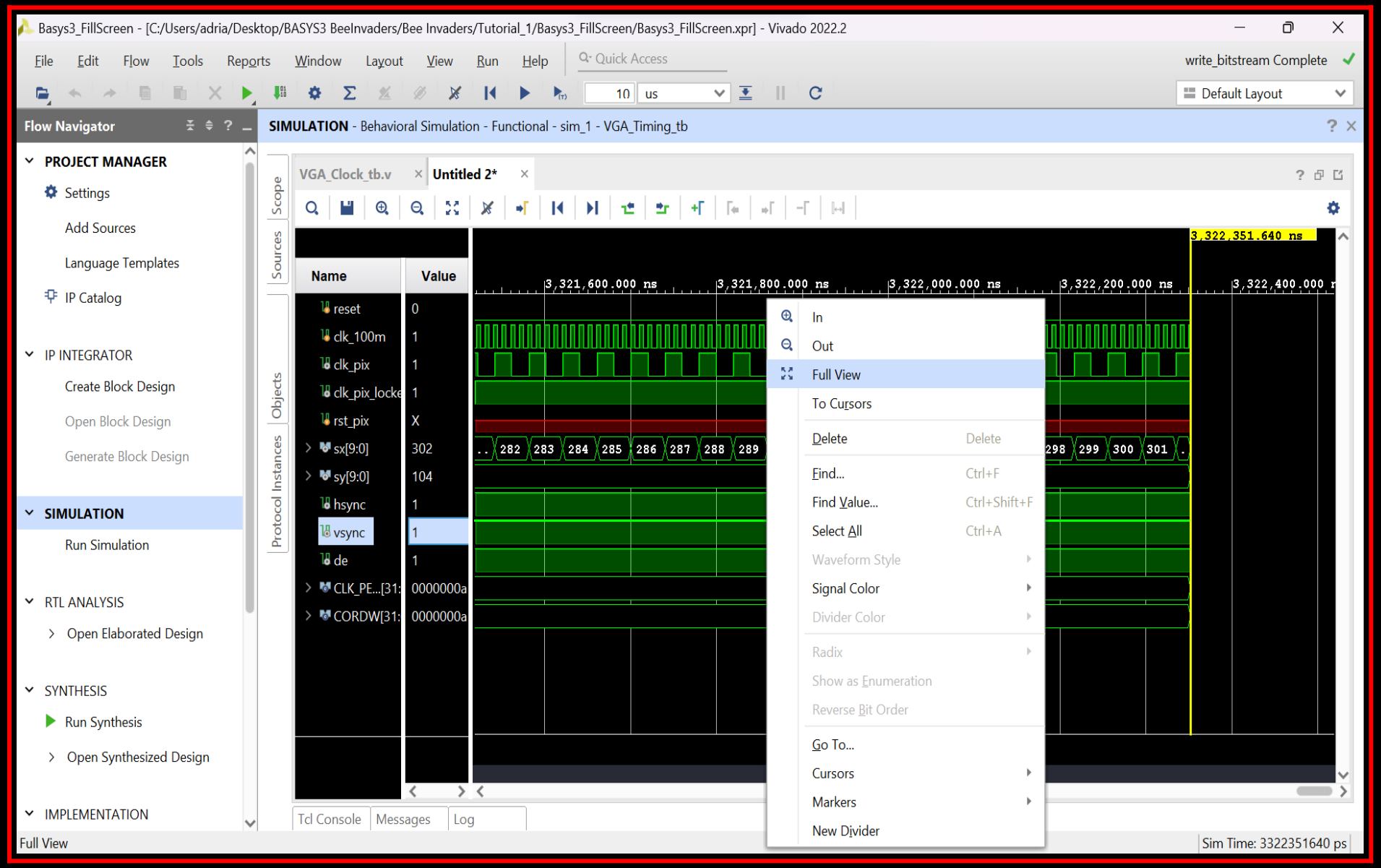
Let the simulator run for about 10 seconds and then click the Break button (highlighted below with a RED circle



24 Right click on "sx[9:0]", move your mouse over "Radix" and left click on "Unsigned Decimal". Do the same for "sy[9:0]"

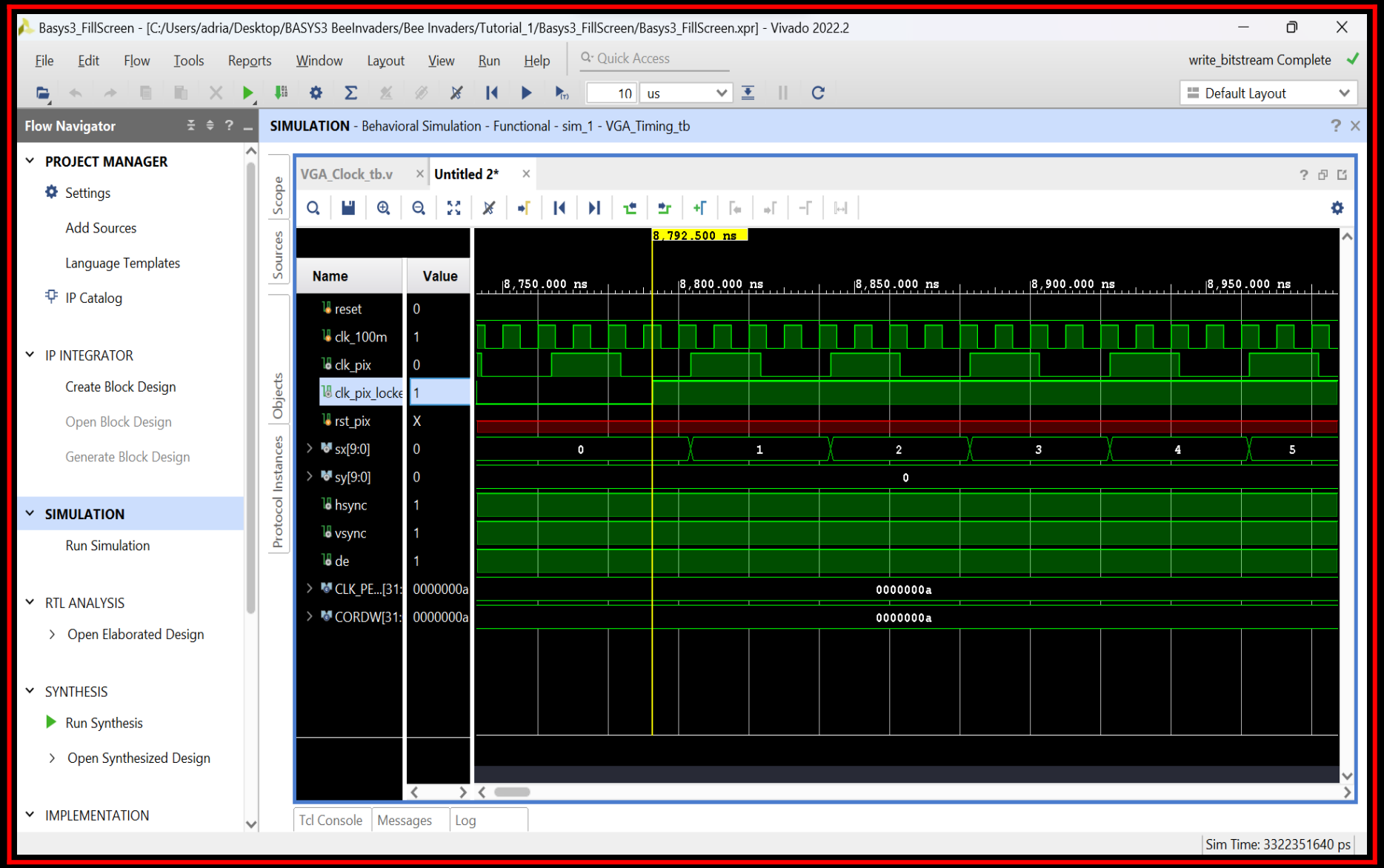


25 Right click in the Simulation window and select "Full View"



26

Use the "Zoom" buttons again and the "Scroll Bar" to position the "clk_pix_locked" signal similar to the below. Click where the "clk_pix_locked" changes to "1" at "8,792.500 ns". You will notice that after this point "sx[9:0]" increments at the start (positive edge) of each "clk_pix" cycle



If you look at the code in the file "VGA_Timing.v" you will see the lines:

```
// calculate horizontal and vertical screen position
always @(posedge clk_pix) begin
```

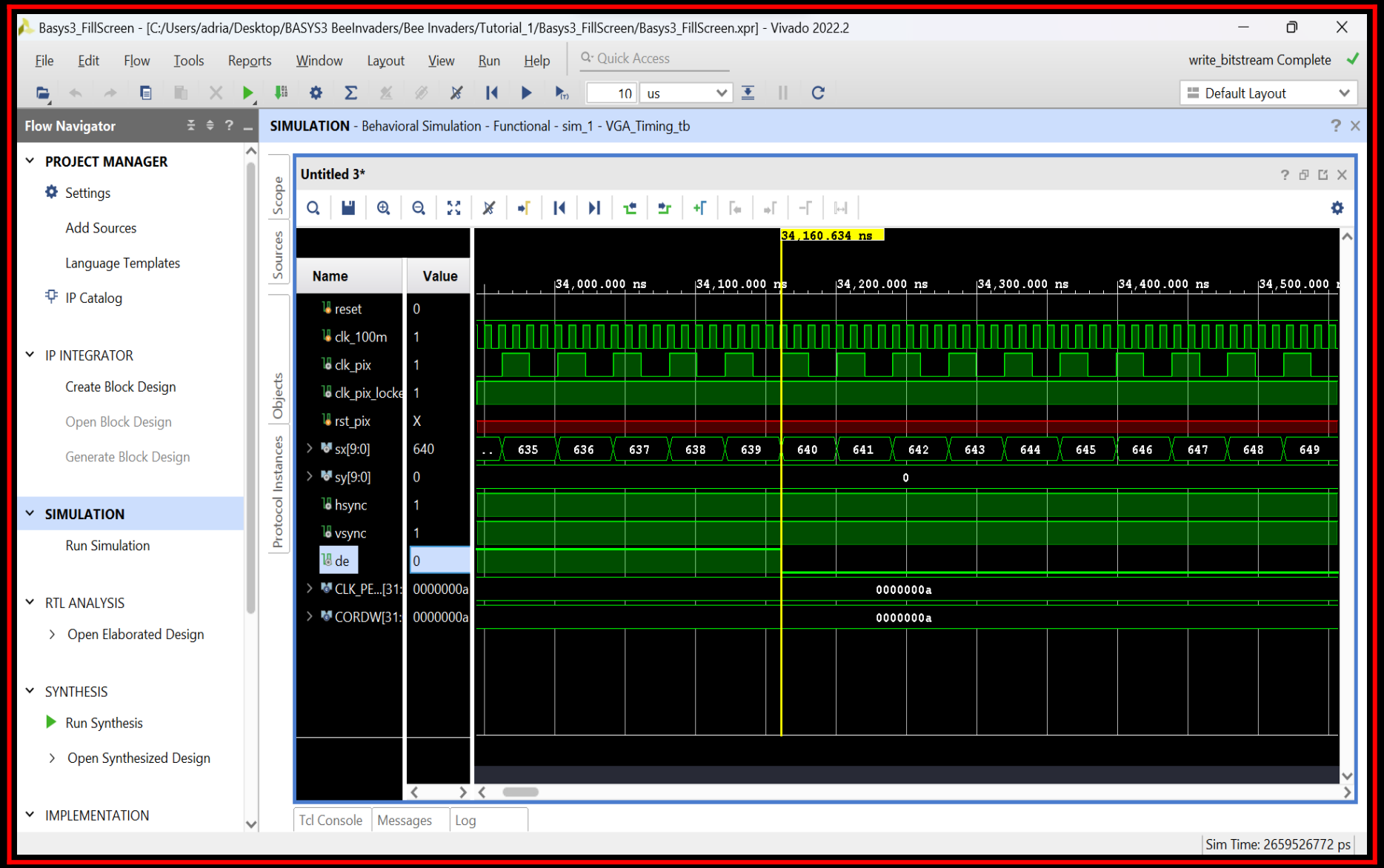
This is telling the code that follows to execute at the start (positive edge) of the "clk_pix" signal

```
if (sx == LINE) begin  // last pixel on line?
    sx <= 0;
    sy <= (sy == SCREEN) ? 0 : sy + 1;  // last line on screen?
end else begin
    sx <= sx + 1;
end
```

"sx" initially starts at "0" and "LINE" equals "799". "sx" is incremented until it equals "LINE". At this point "sx" is reset to "0" and "sy" is incremented until it equals "SCREEN" ("524")

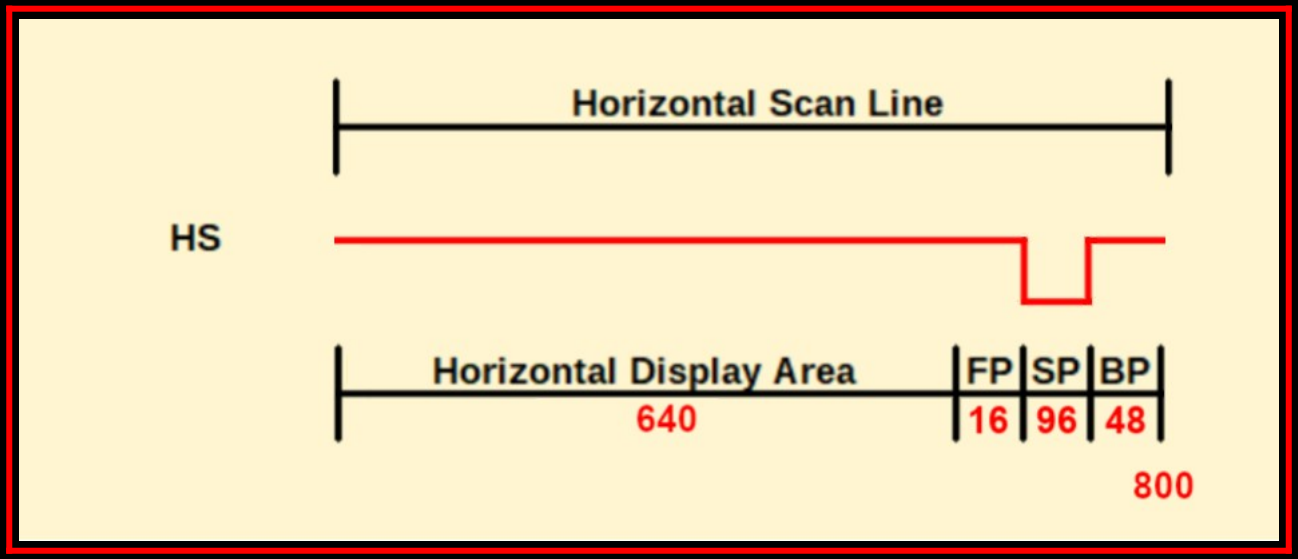
27

Use the "Zoom" buttons again and the "Scroll Bar" to position the "de" signal similar to the below. Click where the "de" changes to "0" at "34,160.634 ns". You will notice that "sx[9:0]" is greater than "639"



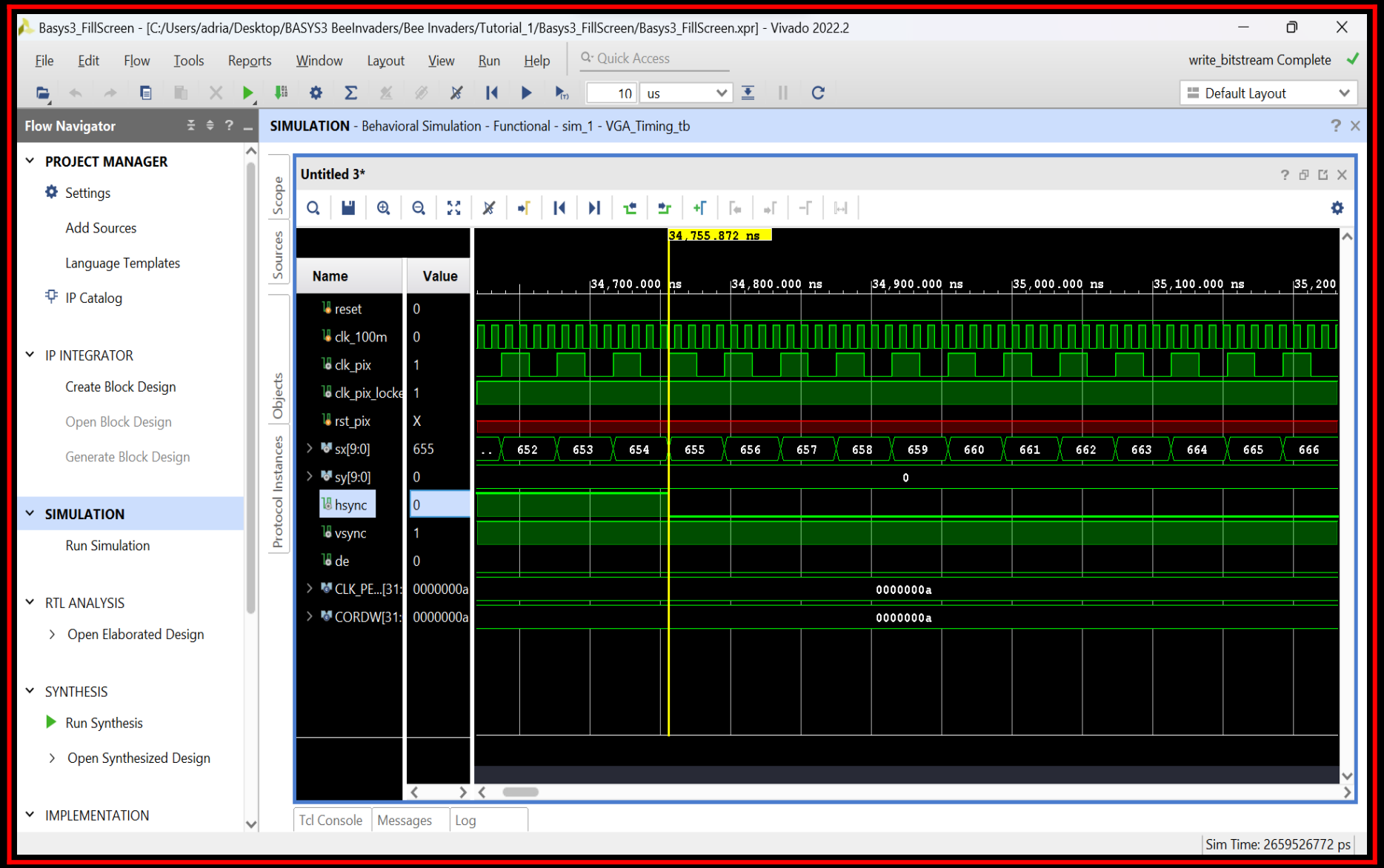
The following code sets "de" (data enable or visible pixels) to "1" when "sx" is less than or equal to "639" ("HA_END") and "sy" is less than or equal to "479" ("VA_END"), else "de" equals "0". When ""de" changes to "0" this signifies the end of the visible pixels and the start of the "Front Porch"

```
// horizontal timings
parameter HA_END = 639;          // end of active pixels
.
.
// vertical timings
parameter VA_END = 479;          // end of active pixels
.
.
assign de = (sx <= HA_END && sy <= VA_END);
```



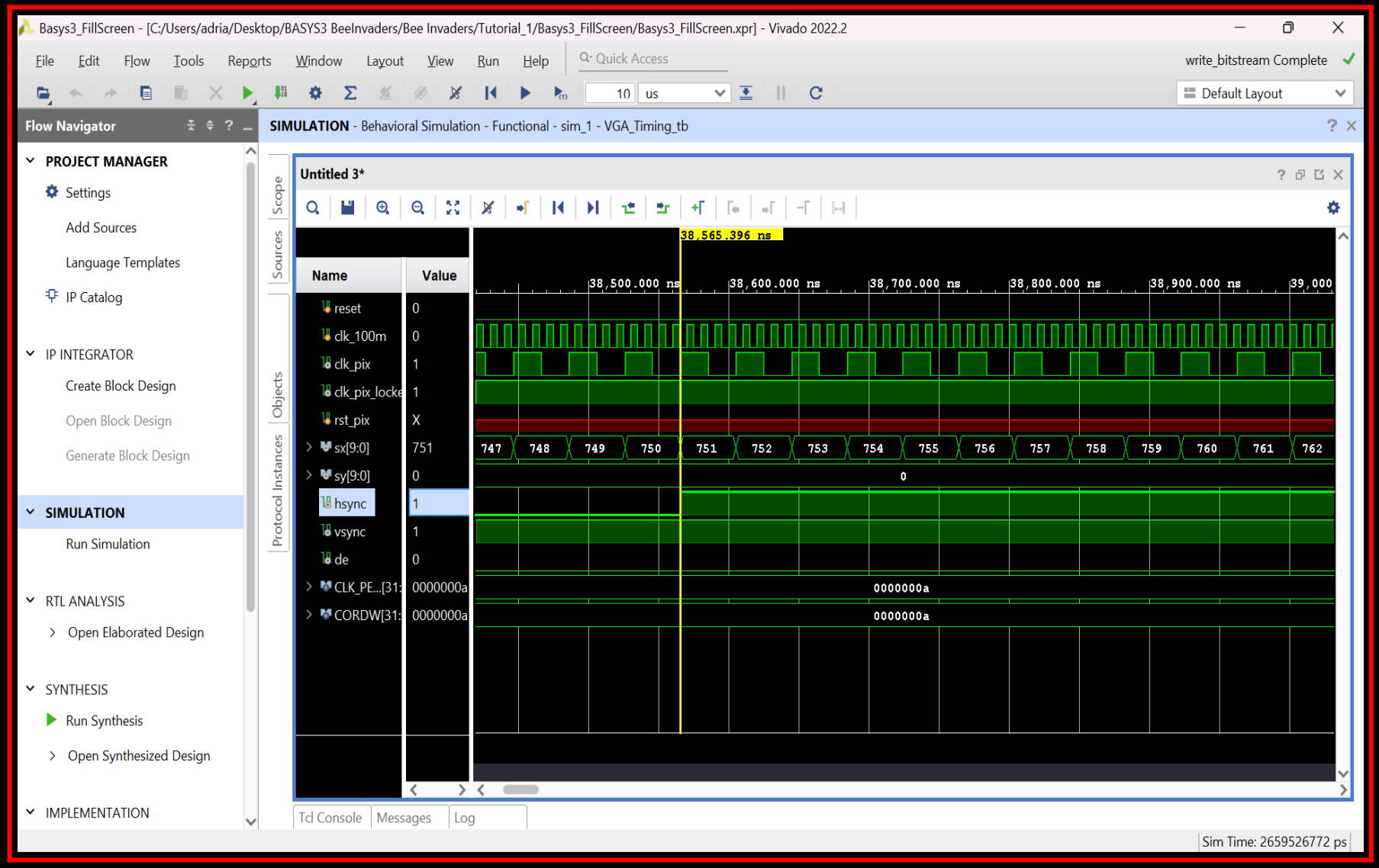
28

Use the "Zoom" buttons again and the "Scroll Bar" to position the "hsync" signal similar to the below. Click where the "hsync" changes to "0" at "34,755.872 ns". You will also notice that "sx[9:0]" equals "655"



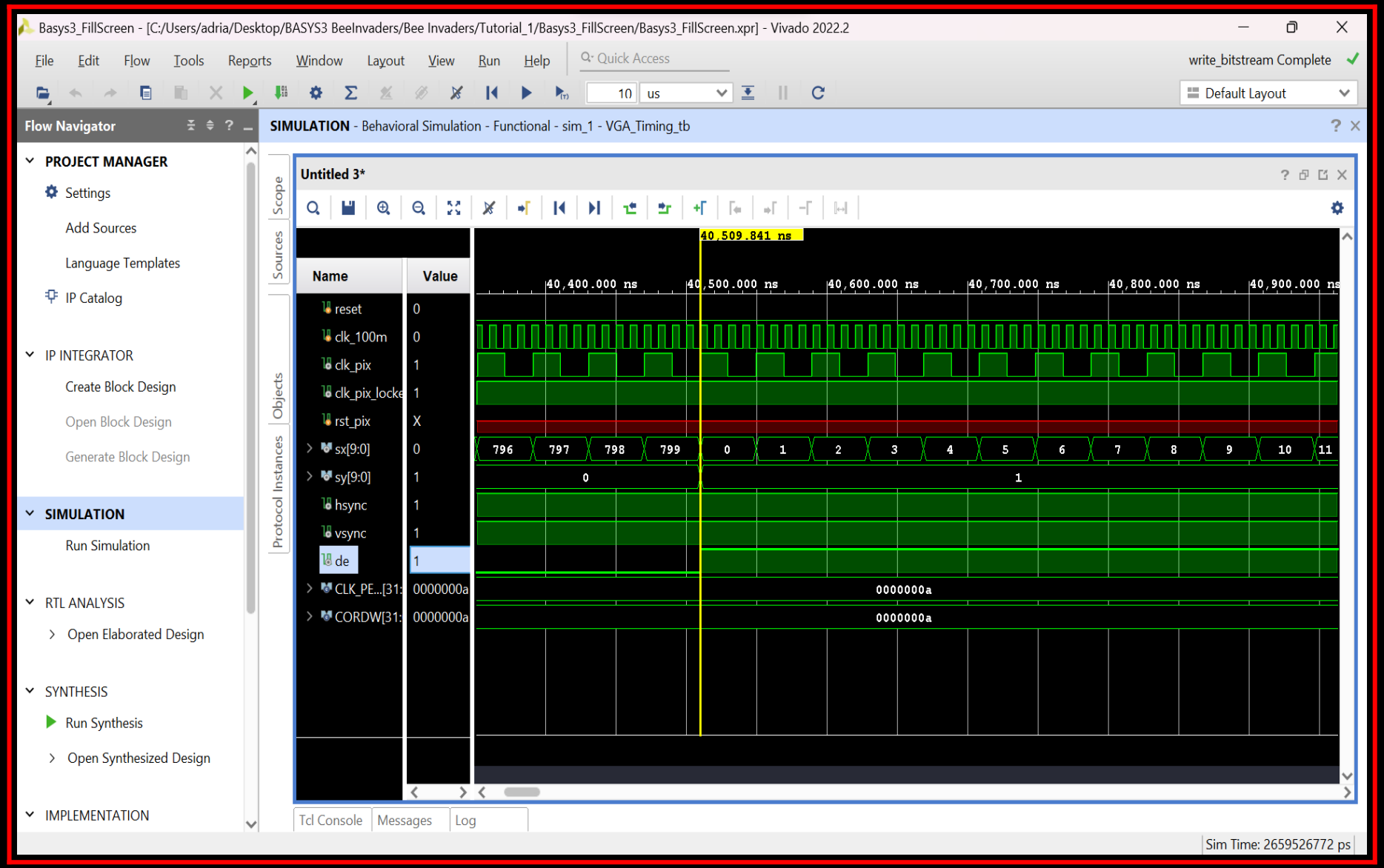
```
assign hsync = ~(sx >= HS_STA && sx < HS_END);  // invert: negative polarity
```

"Hsync" equals "1" unless it is greater than or equal to "655" ("HS_STA") and less than "751" ("HS_END"). "Hsync" equals "0" during the "Sync Pulse" period

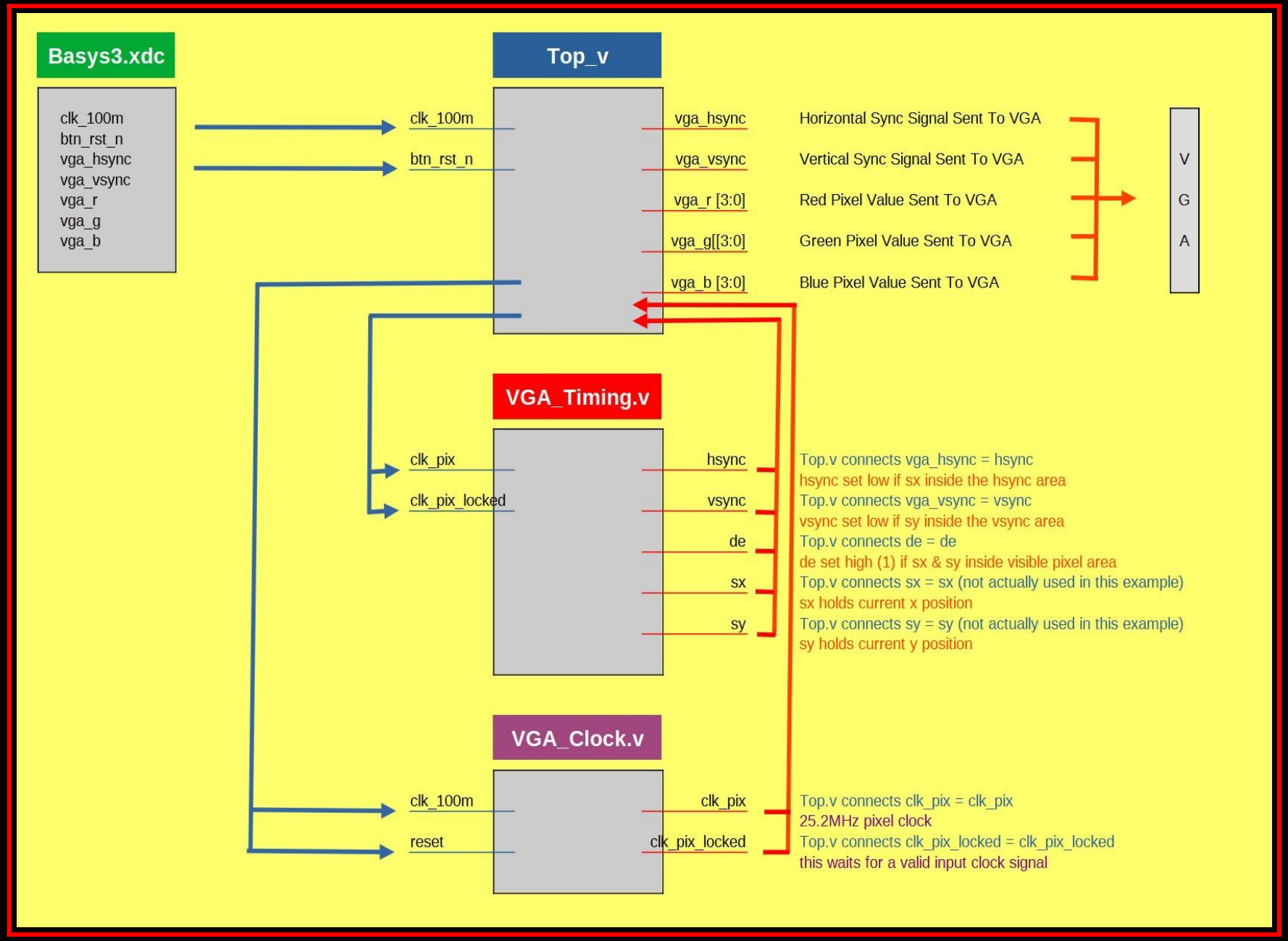


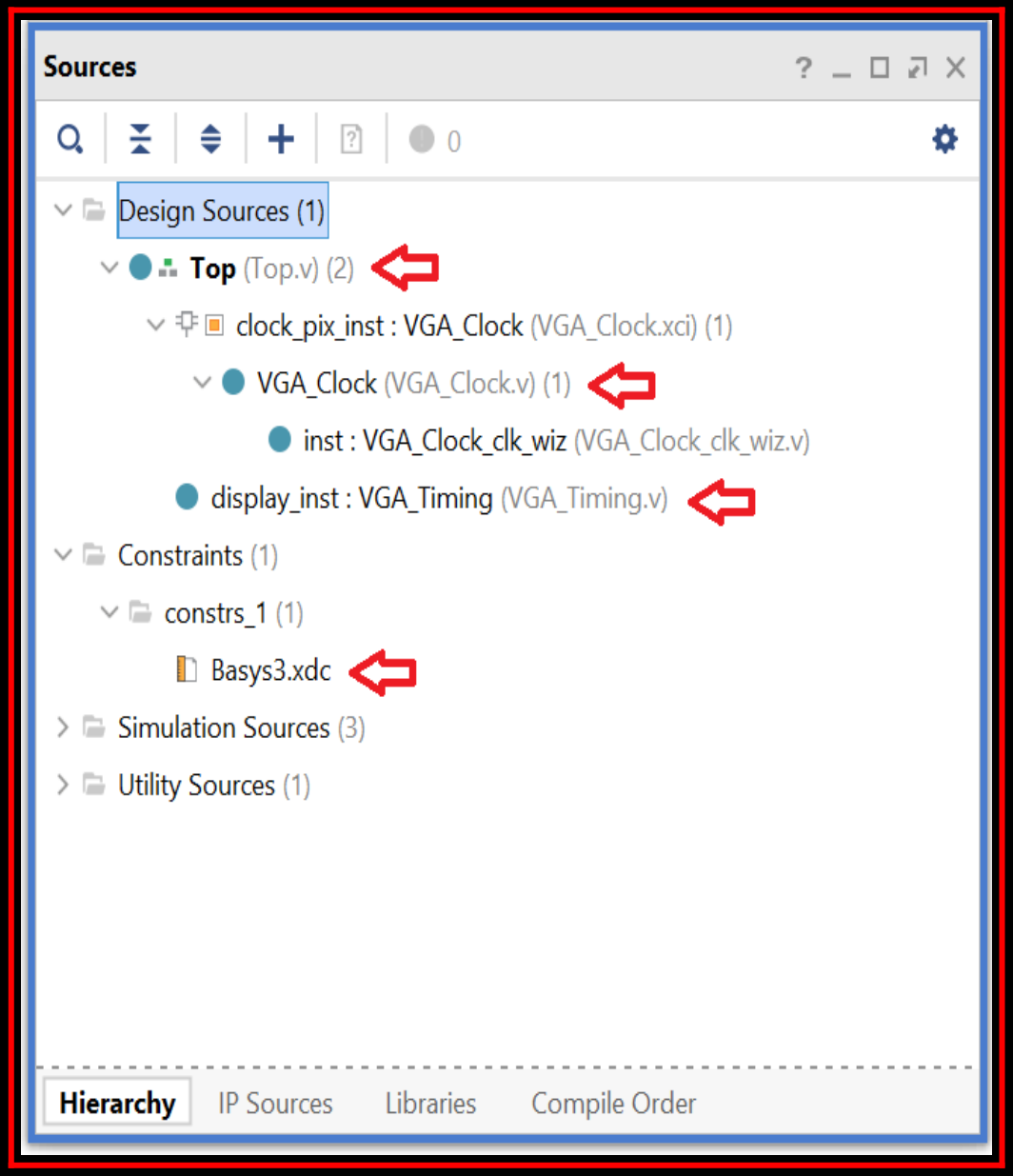
29

Use the "Zoom" buttons again and the "Scroll Bar" to position the "sx[9:0]" signal similar to the below. You will notice that when "sx[9:0]" is greater than "799" it is reset to "0". At the same time "sy[9:0]" is incremented to "1" and "DE" is set to "1"



(E) EXPLANATION OF THE VERILOG CODE USED




Sources
0
Design Sources (1)
Top (Top.v) (2)
clock_pix_inst : VGA_Clock (VGA_Clock.xci) (1)
VGA_Clock (VGA_Clock.v) (1)
inst : VGA_Clock_clk_wiz (VGA_Clock_clk_wiz.v)
display_inst : VGA_Timing (VGA_Timing.v)
Constraints (1)
constrs_1 (1)
Basys3.xdc
Simulation Sources (3)
Utility Sources (1)
Hierarchy
IP Sources
Libraries
Compile Order

01 Top.v module

```
//----------------------------------------------
// Top.v module
// Digilent Basys 3
// Bee Invaders Tutorial_1
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//----------------------------------------------

`default_nettype none
`timescale 1ns / 1ps

module Top (
    input  wire clk_100m,       // 100 MHz clock
    input  wire btn_rst_n,      // reset button
    output wire vga_hsync,      // VGA horizontal sync
    output wire vga_vsync,      // VGA vertical sync
    output reg [3:0] vga_r,     // 4-bit VGA red
    output reg [3:0] vga_g,     // 4-bit VGA green
    output reg [3:0] vga_b      // 4-bit VGA blue
    );
```

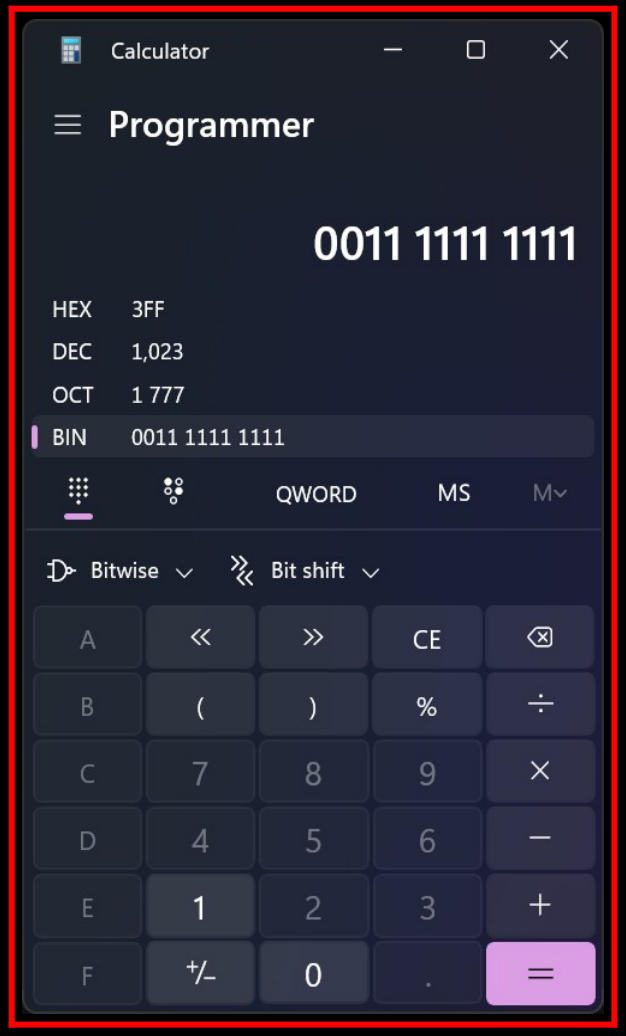
This defines the 100MHz clock (clk_100m wire input), the reset button (btn_rst_n wire input) and the 5 sections of the VGA on the Basys3 board (vga_hsync wire output, vga_vsync wire output, vga_r 4 bit RED reg output, vga_g 4 bit GREEN reg output and vga_b 4 bit BLUE reg output).

```
// generate pixel clock
reg reset;                      // Reset Button
wire clk_pix;                   // 25.2Mhz Pixel clock
wire clk_pix_locked;            // Pixel clock locked?

VGA_Clock clock_pix_inst (
  .clk_100m(clk_100m),
  .reset(btn_rst_n),            // reset button is active high
  .clk_pix(clk_pix),
  .clk_pix_locked(clk_pix_locked)
);
```

This sets reset as a Reg and clk_pix, clk_pix_locked as Wires. The "VGA_Clock.v" module is called: clk_100m and reset are sent to "VGA_Clock.V" and clk_pix, clk_pix_locked are returned to the "Top.v" module. clk_pix is the 25.2MHz pixel clock and clk_pix_locked is used to signify when clk_100m is stable

```
// display sync signals and coordinates
localparam CORDW = 10;      // screen coordinate width in bits
reg rst_pix;
wire [CORDW-1:0] sx, sy;
wire hsync;
wire vsync;
wire de;
VGA_Timing display_inst (
    .clk_pix(clk_pix),
    .rst_pix(!clk_pix_locked),  // wait for clock lock
    .sx(sx),
    .sy(sy),
    .hsync(hsync),
    .vsync(vsync),
    .de(de)
);
```

This part of the code links the "Top.v" module to the "VGA_Timing.v" module



Using the Windows Calculator you can see that 10 bits "1111111111" is 1023 in Decimal

CORDW = 10 is used to define the width in bits (value) of sx nd sy (the current x, y screen co-ordinates)

sx maximum value will be 800 and sy maximum value will be 525

If 9 bits (Decimal 511) was used this would not be enough width to store values in sx and sy

Note, sx and sy are not used it this tutorial

clk_pix and !clk_pix_locked (invert clk_pix_locked, 0 = 1 or 1 = 0) originated from "VGA_Clock.v" and are passed to "VGA_Timing.v"

"VGA_Timimg.v" return values sx, sy, hsync, vsync and de

```
// VGA Output
   assign vga_hsync = hsync;
   assign vga_vsync = vsync;
   always @(posedge clk_pix) begin
      if (de)
         begin  // VGA colour blue
            vga_r <= 4'hD;
            vga_g <= 4'hA;
            vga_b <= 4'h5;
         end
      else
         begin  // VGA colour should be black in blanking interval
            vga_r <= 4'h0;
            vga_g <= 4'h0;
            vga_b <= 4'h0;
         end
      end
endmodule
```

This section outputs the VGA signal to the VGA screen

It sets vga_hsync = hsync and vga_vsync = vsync

The line "always @(posedge clk_pix)" tells the following code to execute on the positive edge of the 25.2MHz pixel clock

If de (data enable) = 1 (when sx and sy are within the visible screen pixel area) 4 bit values for Red, Green and Blue are sent to the VGA screen, in this case an orange/brown colour

If de = 0 (blanking period) values for Red, Green and Blue must to set to 0 (black)

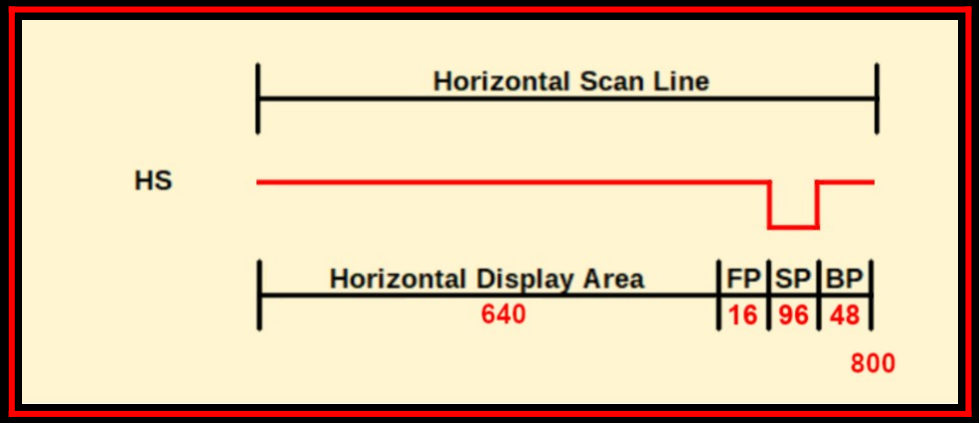
02 VGA_Timing.v module

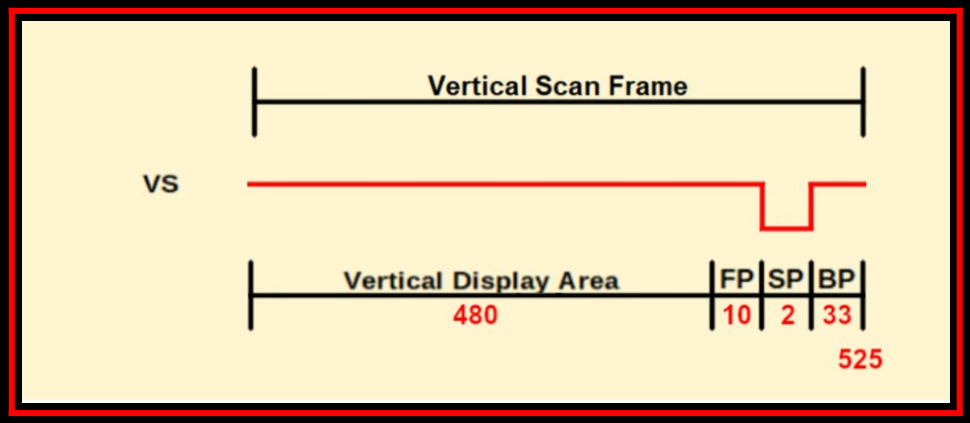
```
//----------------------------------------------
// VGA_Timing.v module
// Digilent Basys 3
// Bee Invaders Tutorial_1
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//----------------------------------------------

`default_nettype none
`timescale 1ns / 1ps


module VGA_Timing (
        input  wire clk_pix,    // pixel clock
        input  wire rst_pix,    // reset in pixel clock domain
        output reg [9:0] sx,    // horizontal screen position
        output reg [9:0] sy,    // vertical screen position
        output wire hsync,      // horizontal sync
        output wire vsync,      // vertical sync
        output wire de          // data enable (low in blanking interval)
        );
```

This consists of defining the 25.2MHz pixel clock (clk_pix) as an input wire, the !clk_pix_locked as an input wire called rst_pix, sx as a reg output, sy as a reg output, hsync as a wire output, vsync as a wire output and de as a wire output

```
// horizontal timings
parameter HA_END = 639;           // end of active pixels
parameter HS_STA = HA_END + 16;   // sync starts after front porch
parameter HS_END = HS_STA + 96;   // sync ends
parameter LINE   = 799;           // last pixel on line (after back porch)

// vertical timings
parameter VA_END = 479;           // end of active pixels
parameter VS_STA = VA_END + 10;   // sync starts after front porch
parameter VS_END = VS_STA + 2;    // sync ends
parameter SCREEN = 524;           // last line on screen (after back porch)
```

This defines the Start and End of the horizontal sync, vertical sync and active area





hsync and vsync are assigned to the horizontal and vertical sync regions. de is assigned to the active pixel area

```
    assign hsync = ~(sx >= HS_STA && sx < HS_END);      // invert: negative polarity
    assign vsync = ~(sy >= VS_STA && sy < VS_END);      // invert: negative polarity
    assign de = (sx <= HA_END && sy <= VA_END);

    // calculate horizontal and vertical screen position
    always @(posedge clk_pix) begin
        if (sx == LINE) begin                          // last pixel on line?
            sx <= 0;
            sy <= (sy == SCREEN) ? 0 : sy + 1;         // last line on screen?
        end else begin
            sx <= sx + 1;
        end
        if (rst_pix) begin
            sx <= 0;
            sy <= 0;
        end
    end
endmodule
```

If sx = last pixel on the line (799), sx = 0 and sy is incremented (unless sy = the last line on the screen (524), then sy = 0)

Else sx is incremented

If rst_pix (!clk_pix_locked) is true sx and sy = 0 (jump to the start of the frame)

03 Basys3.xdc constraints file

```
##-----------------------------------
## Constraints Module
## Digilent Basys 3
## BeeInvaders : Onboard clock 100MHz
## VGA Resolution: 640x480 @ 60Hz
## Pixel Clock 25.2MHz
##-----------------------------------
## Clock
set_property -dict {PACKAGE_PIN W5 IOSTANDARD LVCMOS33} [get_ports {clk_100m}];
create_clock -add -name sys_clk_pin -period 10.00 \
 -waveform {0 5} [get_ports {clk_100m}];
## Use BTNC as Reset Button (active high)
set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports {btn_rst_n}];
## VGA Connector
set_property -dict {PACKAGE_PIN G19 IOSTANDARD LVCMOS33} [get_ports {vga_r[0]}];
set_property -dict {PACKAGE_PIN H19 IOSTANDARD LVCMOS33} [get_ports {vga_r[1]}];
set_property -dict {PACKAGE_PIN J19 IOSTANDARD LVCMOS33} [get_ports {vga_r[2]}];
set_property -dict {PACKAGE_PIN N19 IOSTANDARD LVCMOS33} [get_ports {vga_r[3]}];
set_property -dict {PACKAGE_PIN N18 IOSTANDARD LVCMOS33} [get_ports {vga_b[0]}];
set_property -dict {PACKAGE_PIN L18 IOSTANDARD LVCMOS33} [get_ports {vga_b[1]}];
set_property -dict {PACKAGE_PIN K18 IOSTANDARD LVCMOS33} [get_ports {vga_b[2]}];
set_property -dict {PACKAGE_PIN J18 IOSTANDARD LVCMOS33} [get_ports {vga_b[3]}];
set_property -dict {PACKAGE_PIN J17 IOSTANDARD LVCMOS33} [get_ports {vga_g[0]}];
set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports {vga_g[1]}];
set_property -dict {PACKAGE_PIN G17 IOSTANDARD LVCMOS33} [get_ports {vga_g[2]}];
set_property -dict {PACKAGE_PIN D17 IOSTANDARD LVCMOS33} [get_ports {vga_g[3]}];
set_property -dict {PACKAGE_PIN P19 IOSTANDARD LVCMOS33} [get_ports {vga_hsync}];
set_property -dict {PACKAGE_PIN R19 IOSTANDARD LVCMOS33} [get_ports {vga_vsync}];
## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
```

The constraints file should set up all the Input/s and Output/s used in the code

Each type of input or output has a designated pin number and the name used in the code for each input or output is included in the constraints file

PIN	NAME	PIN NAME	DESCRIPTION
W5	clk_100m	IO_L12P_T1_MRCC_34	Onboard 100MHz clock
U18	btn_rst_n	IO_L18N_T2_A11_D27_14	Reset Button
P19	vga_hsync	IO_L10P_T1_D14_14	VGA horizontal sync
R19	vga_vsync	IO_L10N_T1_D15_14	VGA vertical sync
G19	vga_r[0]	IO_L4N_T0_D05_14	VGA Red Bit 0
H19	vga_r[1]	IO_L4P_T0_D04_14	VGA Red Bit 1
J19	vga_r[2]	IO_L6N_T0_D08_VREF_14	VGA Red Bit 2
N19	vga_r[3]	IO_L9N_T1_DQS_D13_14	VGA Red Bit 3
J17	vga_g[0]	IO_L7P_T1_D09_14	VGA Green Bit 0
H17	vga_g[1]	IO_L5P_T0_D06_14	VGA Green Bit 1
G17	vga_g[2]	IO_L5N_T0_D07_14	VGA Green Bit 2
D17	vga_g[3]	IO_0_14	VGA Green Bit 3
N18	vga_b[0]	IO_L9P_T1_DQS_14	VGA Blue Bit 0
L18	vga_b[1]	IO_L8P_T1_D11_14	VGA Blue Bit 1
K18	vga_b[2]	IO_L8N_T1_D12_14	VGA Blue Bit 2
J18	vga_b[3]	IO_L7N_T1_D10_14	VGA Blue Bit 3

The VGA section of the Basys3 board consists of 4 bit Red, 4 bit Green and 4 bit Blue (12 bit or 4096 colours) as shown below