



Hardware Design Lab

Lab 1: Get Started with VIVADO and BASYS3

Instructor: Dr. Haiyu Mao

TA:

Zihao Pu

Ali Alsarraf

Stephen Johannesson

05 FEB 2026

Gratefully acknowledge:

Prof. Onur Mutlu (ETH)

Dr. Yair Linn (TRIUMF Canada)

Table of Contents

- ❑ Part 1: Introduction to FPGA and Hardware Design Procedure
- ❑ Part 2: The Lab Task – Using Vivado and build a 4-bit full adder
- ❑ Part 3: Coursework Introduction and Task 1 Assignment



Learning Outcomes

- ❑ How to make **trade-offs** between **performance** and **area/complexity** in your hardware implementation

- ❑ Hands-on experience on:
 - Hardware **Prototyping** on Field Programmable Gate Arrays (FPGAs)
 - **Debugging** Your Hardware Implementation
 - Hardware Description Language (**HDL**)
 - Hardware Design Flow
 - Computer-Aided Design (**CAD**) Tools

Part 1

Introduction to FPGA

Introduction to FPGA

Where do FPGAs fit in history?

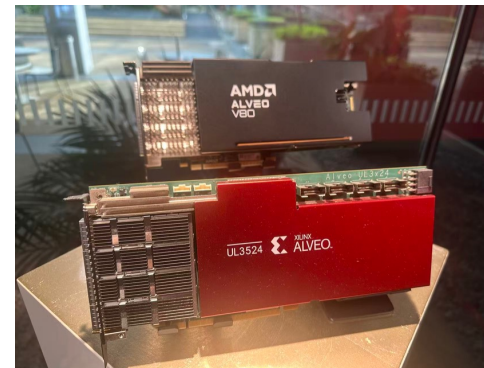
- ❑ <1960 individual transistors
- ❑ 1960s – 1970s:
 - SSI, MSI, LSI (10,000 transistors)
- ❑ 1980s:
 - Programmable Logic Devices (PLAs, PALs)
 - 16-bit, 32-bit processors (> 1,000,000 transistors)
- ❑ 1990s:
 - Full custom chips
 - Gate arrays (semi-custom chips)
 - **Field-Programmable Gate Arrays**
 - > 100,000,000 transistors
- ❑ 2026: > 100,000,000,000 transistors
 - NVIDIA RTX 5090: 92 billion
 - Apple M3 Ultra: 184 billion



First silicon transistor,
Patented 1954

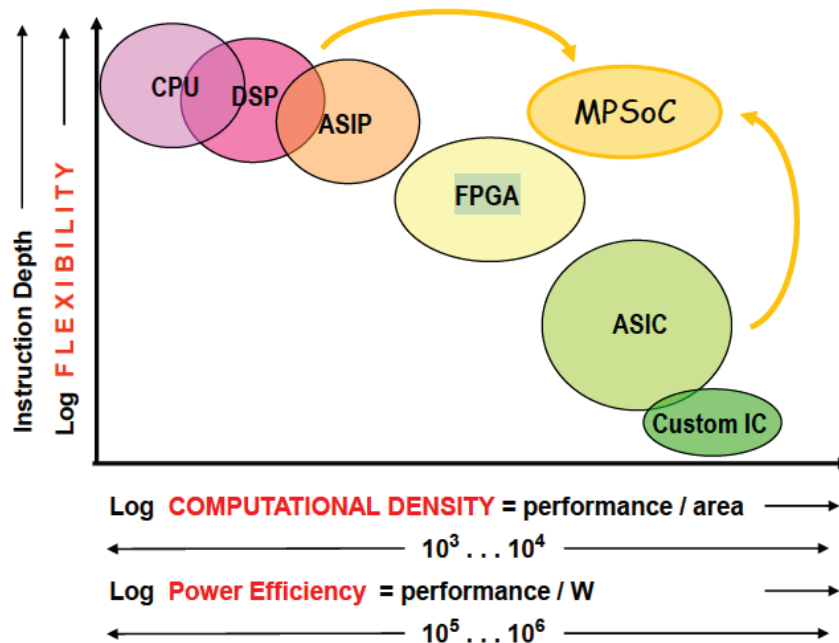


First FPGA,
Patented 1984



FPGA in 2024,
shoot by
Zihao@AMD

Introduction to FPGA



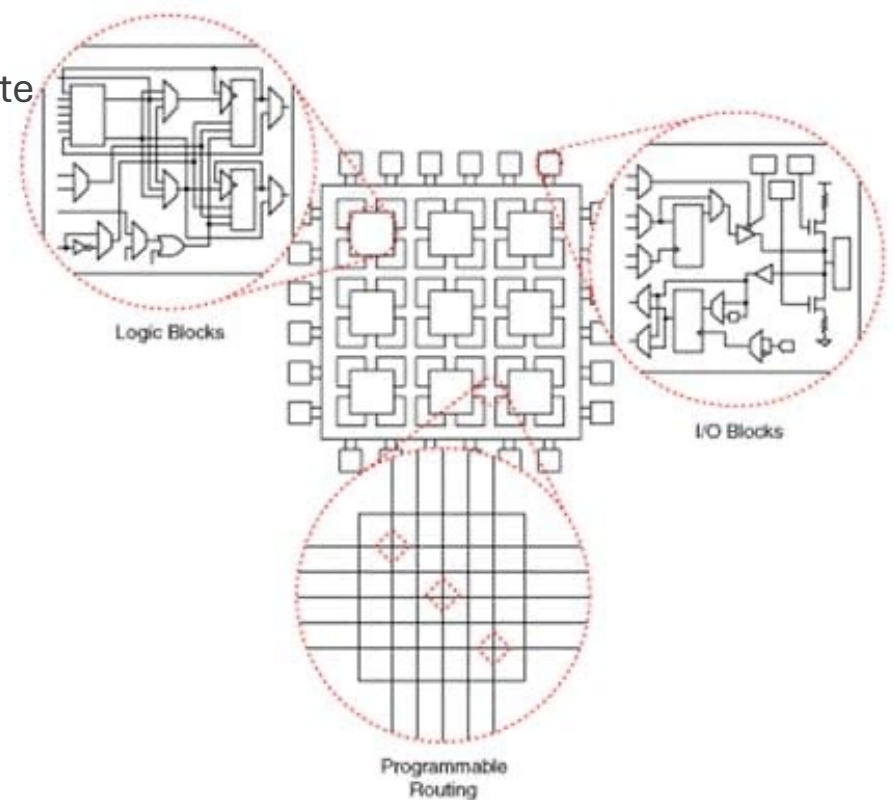
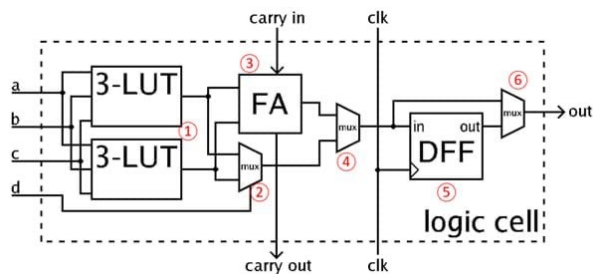
Flexibility vs.
Performance:
Power dissipation
dilemma

Sources:

- [1] Reconfigurable Architectures for General Purpose Computing, Andre DeHon, PhD Thesis, MIT, 1996
 [2] A. Cuomo, Semiconductor Challenges, DATE03 Keynote, March 03, <http://www.dateconference.com/conference/2003/keynotes/andrea/andrea.pdf>

Introduction to FPGA

- ❑ Field Programmable Gate Array (FPGA) is
- ❑ FPGA is a **software-reconfigurable** hardware substrate
 - Reconfigurable **functions**
 - Reconfigurable **interconnection** of functions
 - Reconfigurable **input/output (IO)**

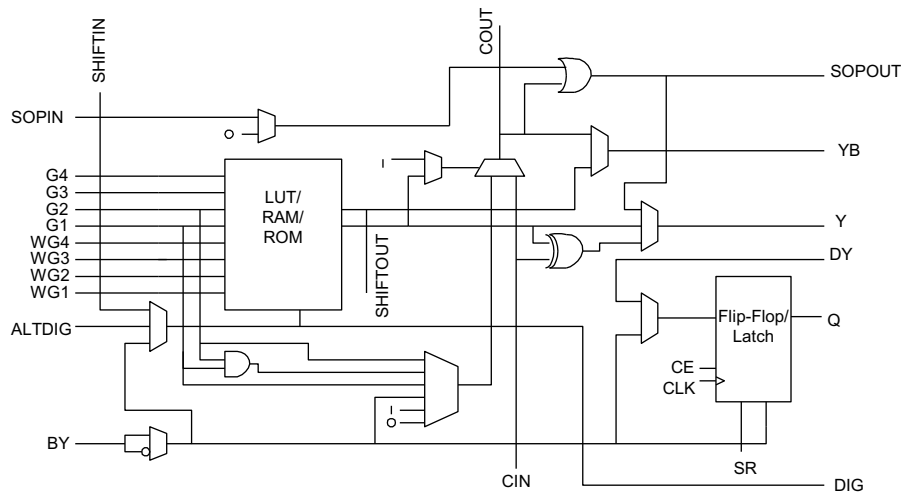


<https://www.arrow.com/en/research-and-events/articles/fpga-basics-architecture-applications-and-uses>

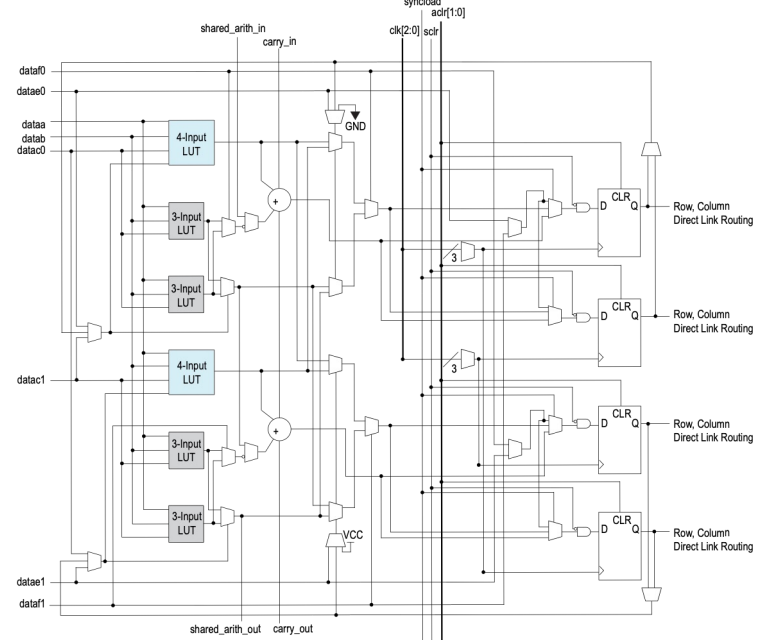
Introduction to FPGA

Different manufacturers have different logic cells

Xilinx Virtex II Logic Block

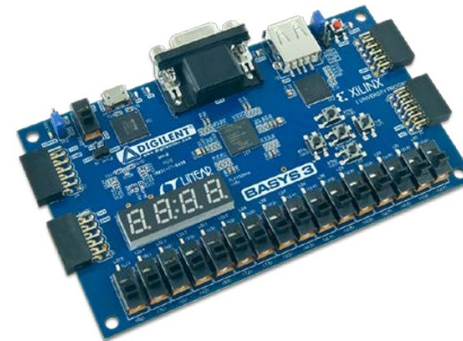


Altera Stratix V Logic Block:



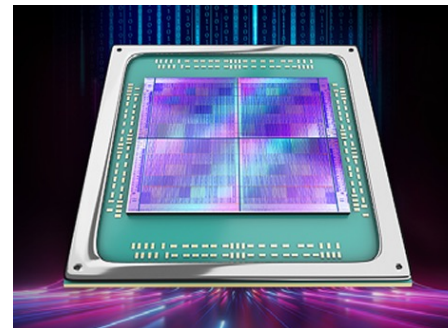
Introduction to FPGA

- ❑ How many logic blocks in a typical FPGA?
- ❑ We will use chips that have around 33k logic blocks.
- ❑ Some other typical chips has around 10k-100k logic cells



- 33,280 logic cells in 5200 slices (each slice contains four 6-input LUTs and 8 flip-flops)
- 1,800 Kbits of fast block RAM
- Five clock management tiles, each with a phase-locked loop (PLL)
- 90 DSP slices
- Internal clock speeds exceeding 450MHz
- On-chip analog-to-digital converter (XADC)

Part	Logic Cells
XC7VX330T	326,400
XC7VX415T	412,160
XC7VX485T	485,760
XC7VX550T	554,240
XC7VX690T	693,120
XC7VX980T	979,200
XC7VX1140T	1,139,200

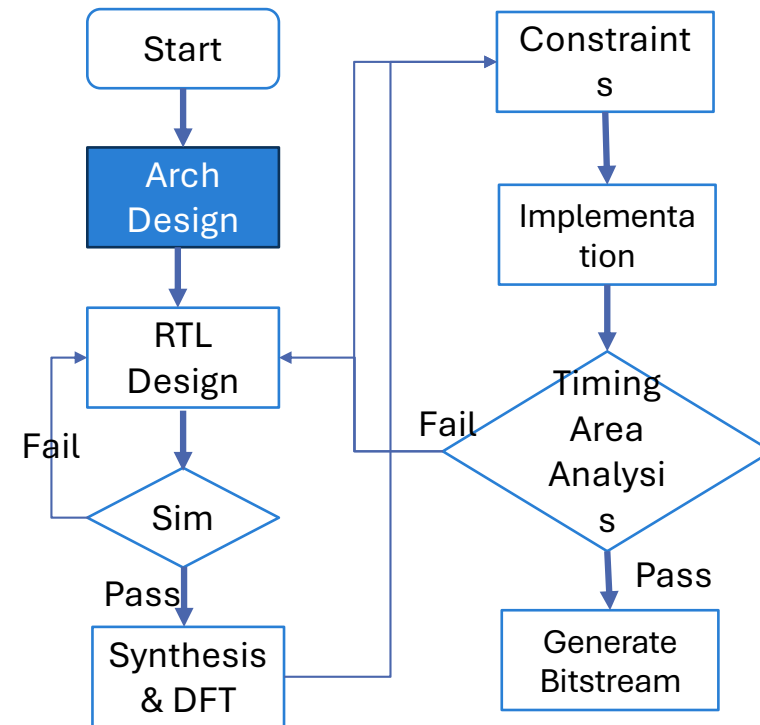
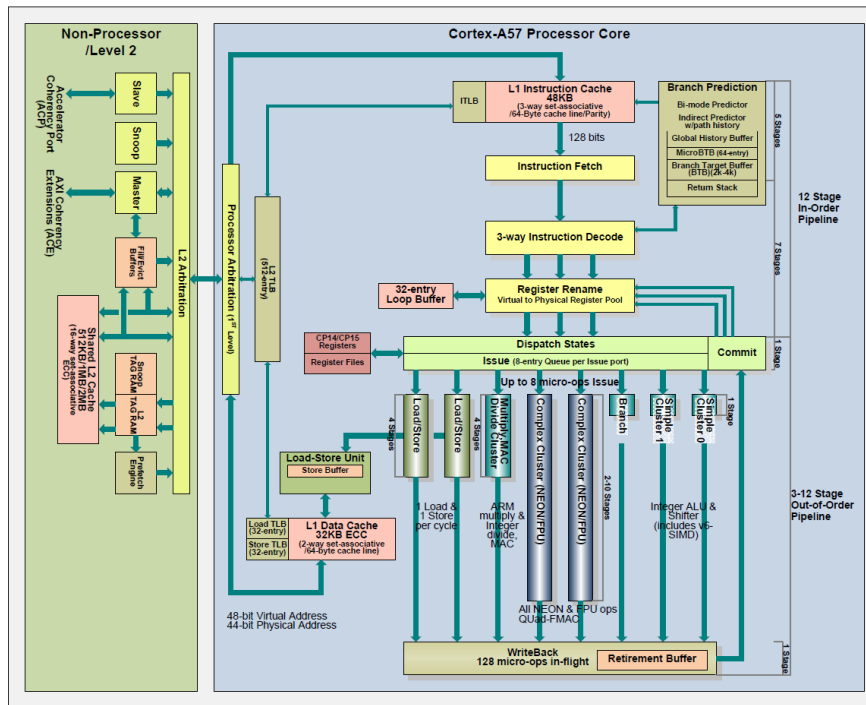


State of the Art: AMD
Versal VP1902
18.5M logic cells, world's
largest FPGA

Sources:
https://www.amd.com/content/dam/amd/en/documents/university/aup-boards/XUPBasys3/documentation/Basys3_rm_8_22_2014.pdf
<https://docs.amd.com/v/u/en-US/7-series-product-selection-guide>
<https://www.amd.com/en/products/adaptive-socs-and-fpgas/versal/premium-series/vp1902.html>

FPGA Design Procedure

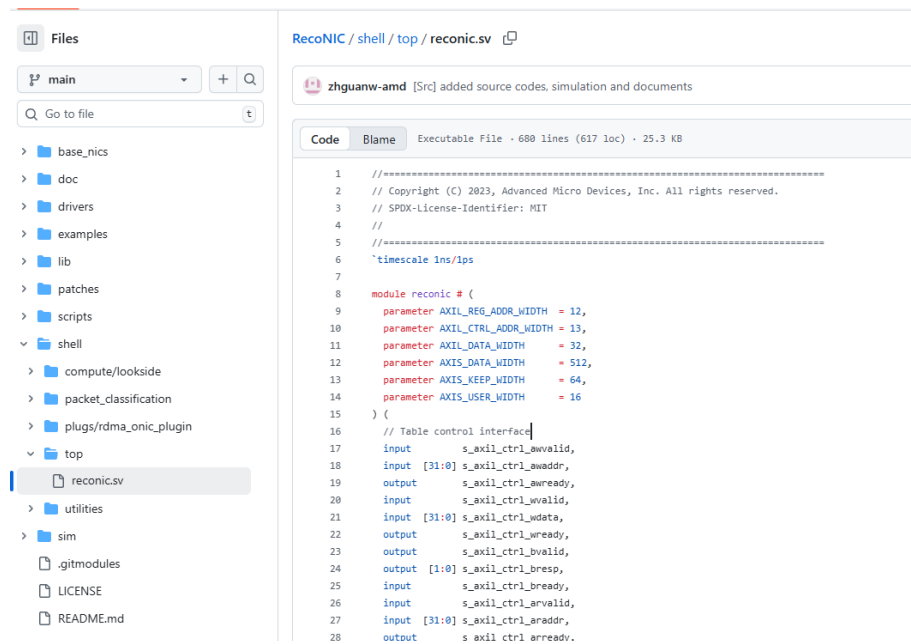
- Step 1. Architecture Design
 - Draw block diagrams, cook circuit specifications



FPGA Design Procedure

Step 2: RTL Design

- Write your RTL/HDL code

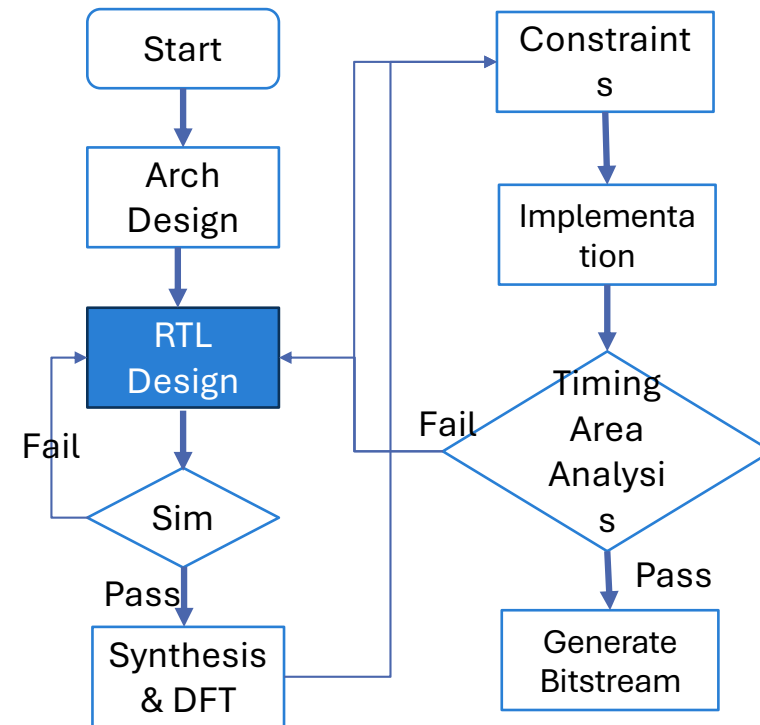


The screenshot shows a file explorer on the left with a tree view containing folders like 'base_nics', 'doc', 'drivers', 'examples', 'lib', 'patches', 'scripts', 'shell', 'compute/lookside', 'packet_classification', 'plugins/rdma_onic_plugin', 'top', 'utilities', 'sim', and files like '.gitmodules', 'LICENSE', and 'README.md'. The main editor window displays the code for 'reconic.sv' with the following content:

```
1 //-----  
2 // Copyright (C) 2023, Advanced Micro Devices, Inc. All rights reserved.  
3 // SPDX-License-Identifier: MIT  
4 //  
5 //-----  
6 `timescale 1ns/1ps  
7  
8 module reconic # (  
9     parameter AXIL_REG_ADDR_WIDTH = 12,  
10    parameter AXIL_CTRL_ADDR_WIDTH = 13,  
11    parameter AXIL_DATA_WIDTH = 32,  
12    parameter AXIS_DATA_WIDTH = 512,  
13    parameter AXIS_KEEP_WIDTH = 64,  
14    parameter AXIS_USER_WIDTH = 16  
15 ) (  
16     // Table control interface  
17     input s_axil_ctrl_0wvalid,  
18     input [31:0] s_axil_ctrl_0waddr,  
19     output s_axil_ctrl_0wready,  
20     input s_axil_ctrl_0wvalid,  
21     input [31:0] s_axil_ctrl_0wdata,  
22     output s_axil_ctrl_0wready,  
23     output s_axil_ctrl_0bvalid,  
24     output [1:0] s_axil_ctrl_0bresp,  
25     input s_axil_ctrl_0bready,  
26     input s_axil_ctrl_0arvalid,  
27     input [31:0] s_axil_ctrl_0araddr,  
28     output s_axil_ctrl_0arready.
```

Source:

<https://github.com/Xilinx/RecoNIC/blob/main/shell/top/reconic.sv#L20>



FPGA Design Procedure

- Step 3: Simulation and verification
 - Write a testbench for simulation
 - Run your testbench using a simulator, and verify the results

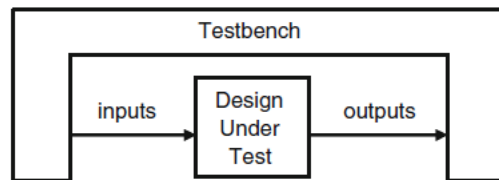
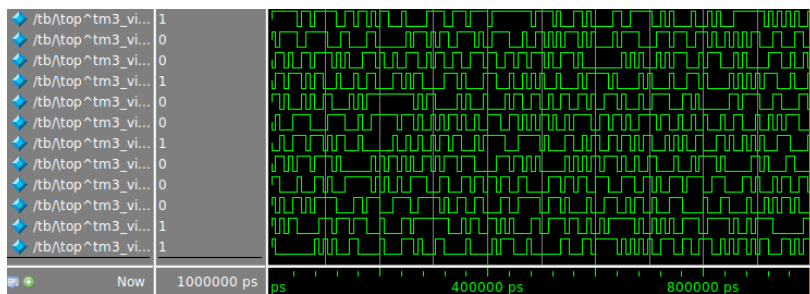
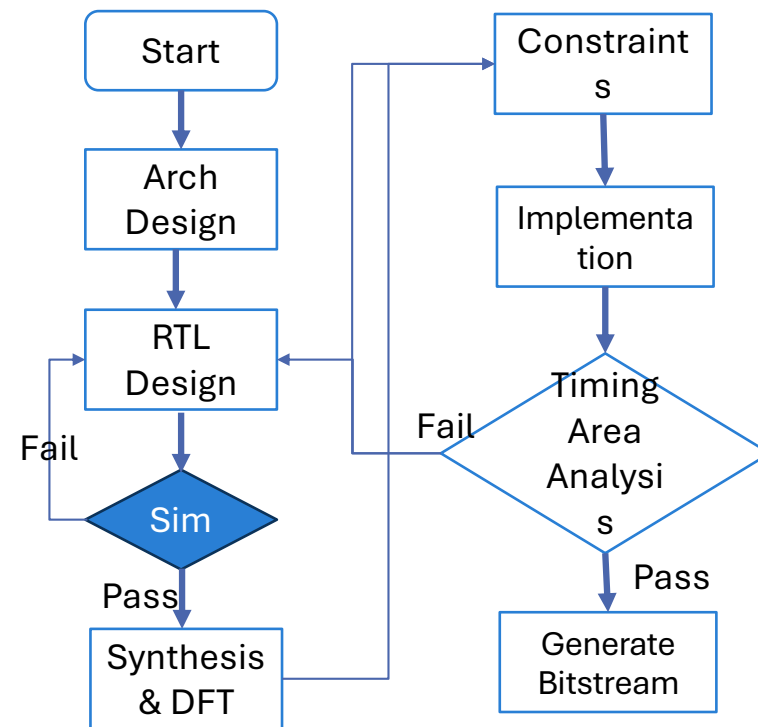


Fig. 1.7 The testbench — design environment



Source:

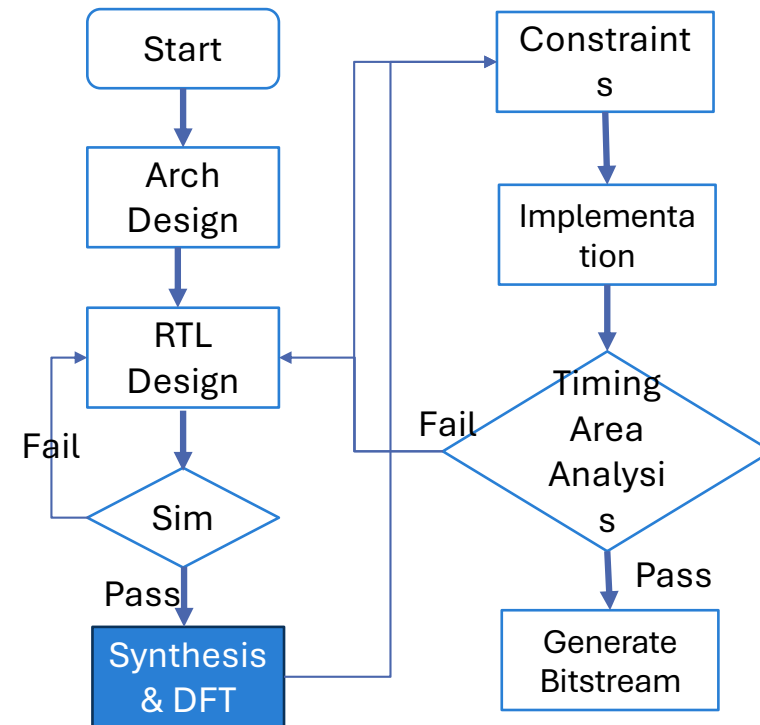
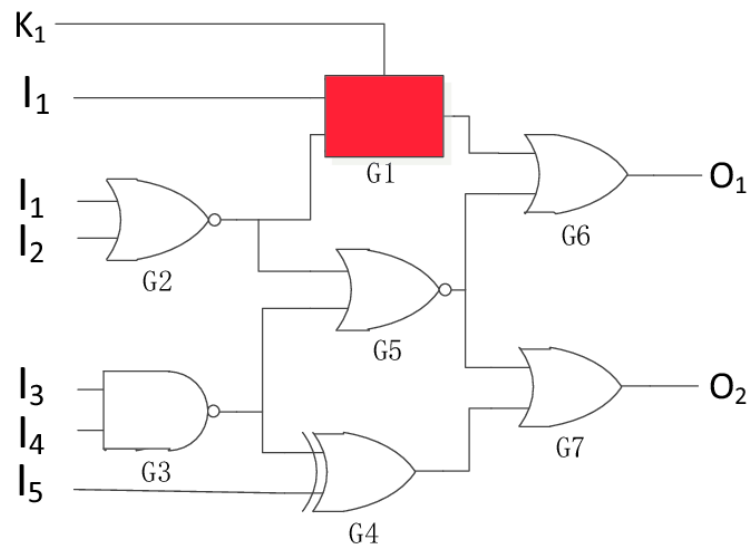
https://docs.verilogtorouting.org/en/latest/tutorials/timing_simulation/



FPGA Design Procedure

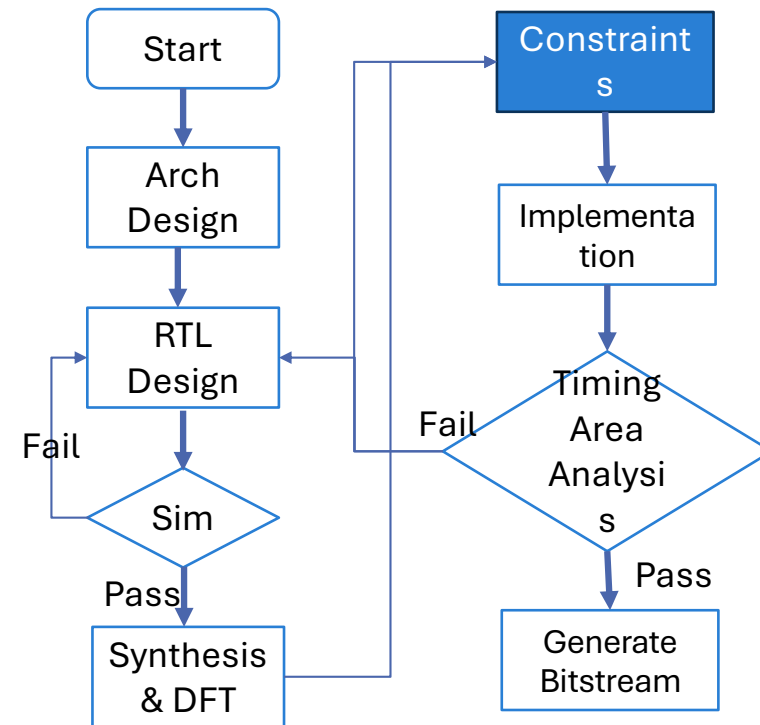
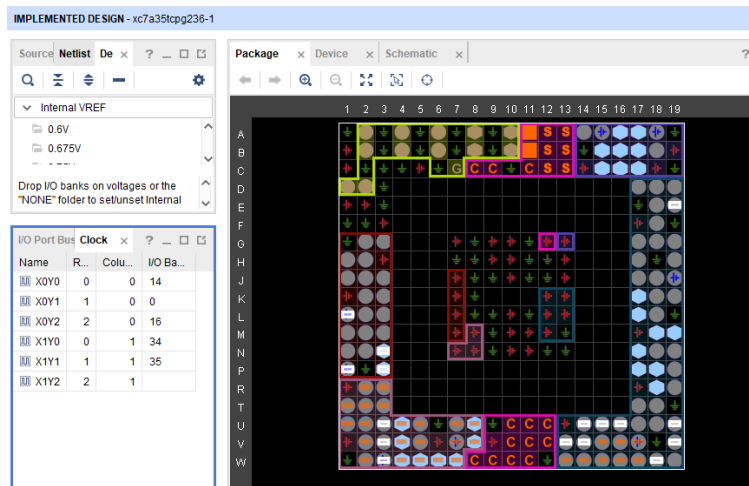
□ Step 4: Synthesis & DFT

- Synthesize your code into a gate netlist
- Add debug cells if needed



FPGA Design Procedure

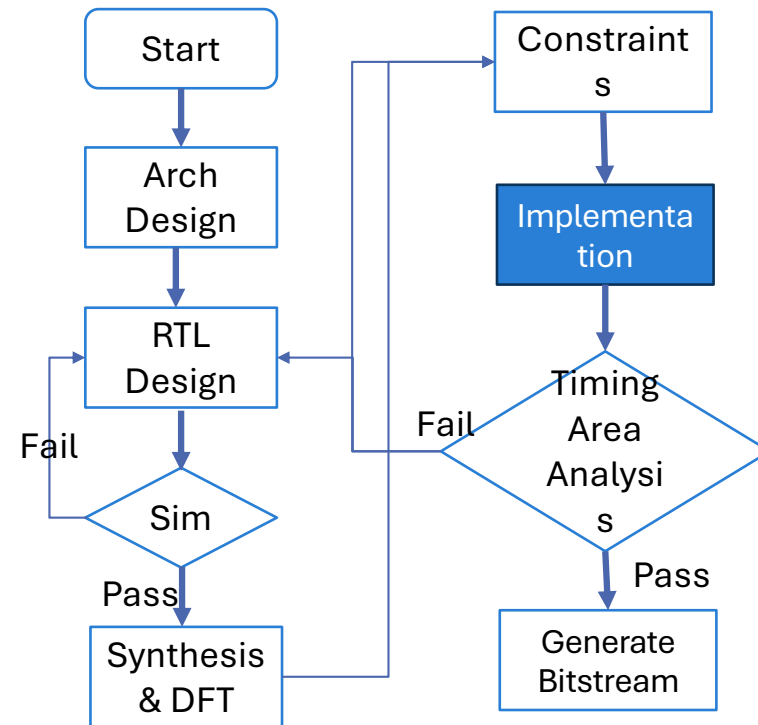
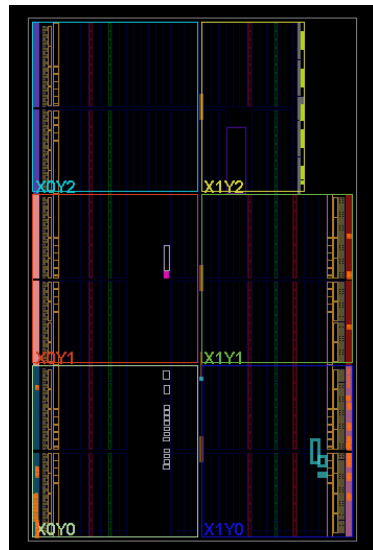
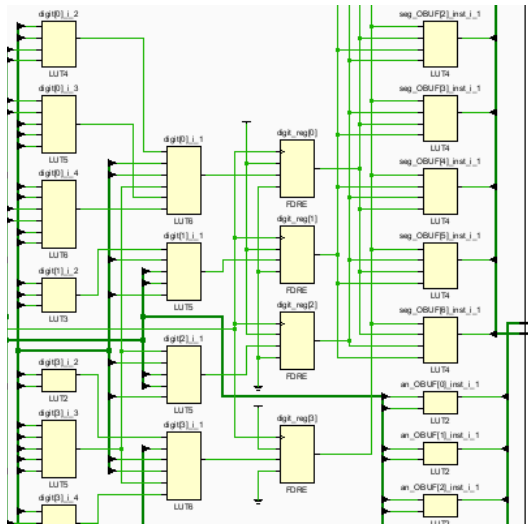
- Step 5: Constraints
 - Set up a constraint to bond the module IO to the pins
 - Constrain the timing requirements



FPGA Design Procedure

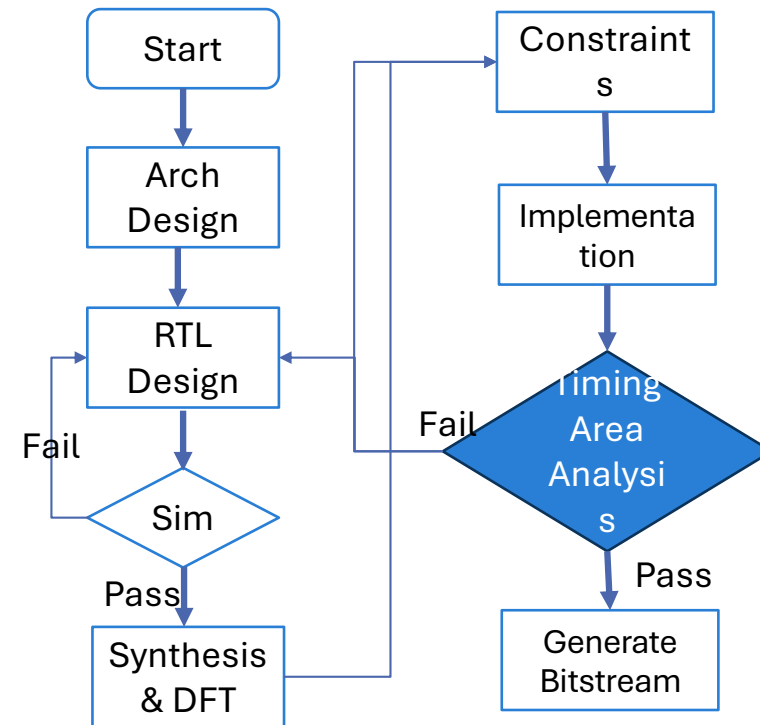
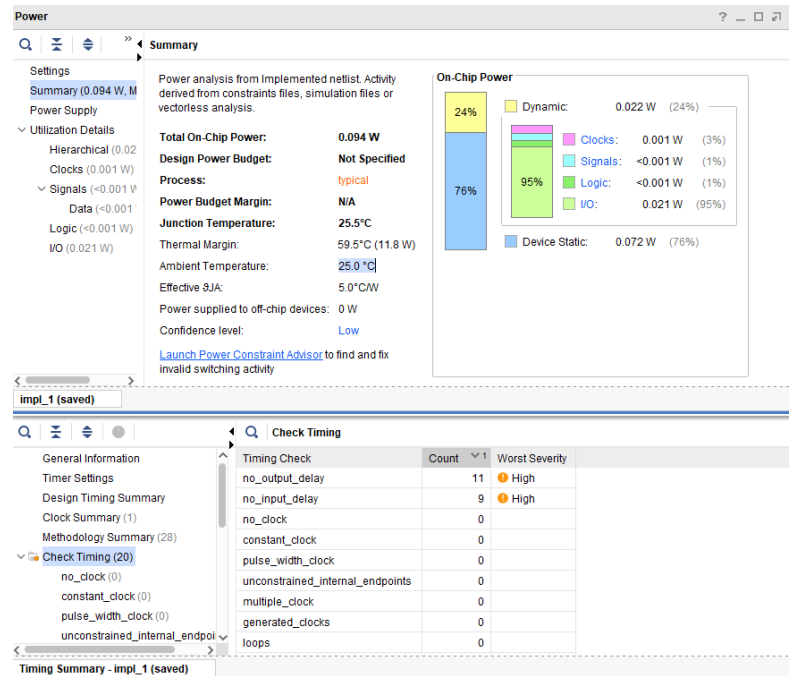
Step 6: Implementation

- Convert the constraints and synthesized gate netlist to an FPGA-compatible logic cell netlist
- Map the cells to a real device



FPGA Design Procedure

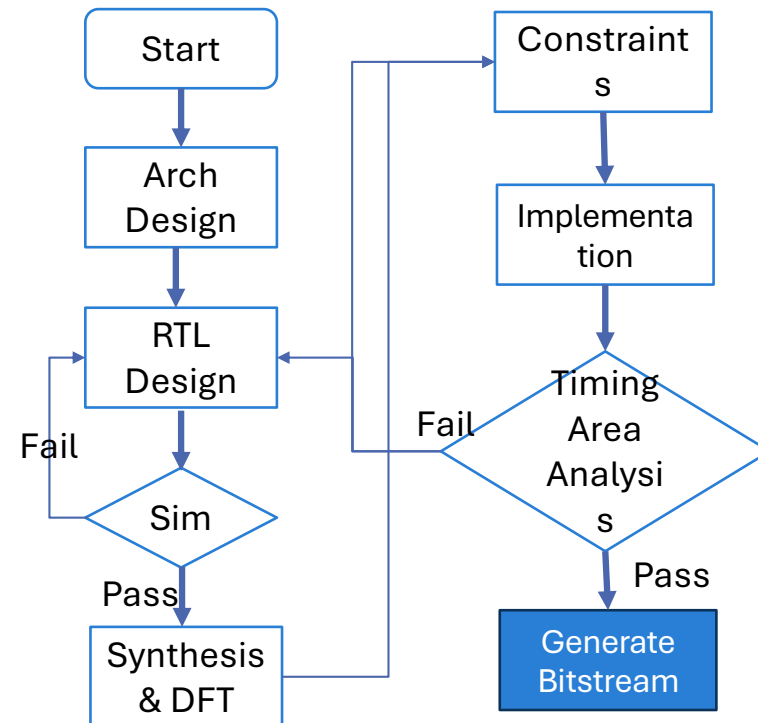
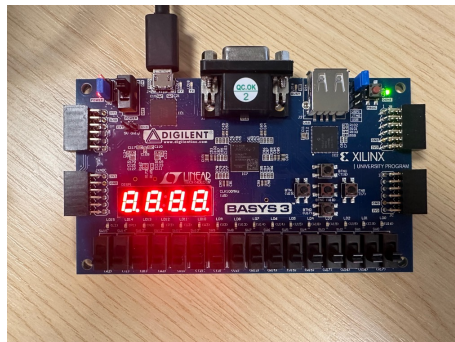
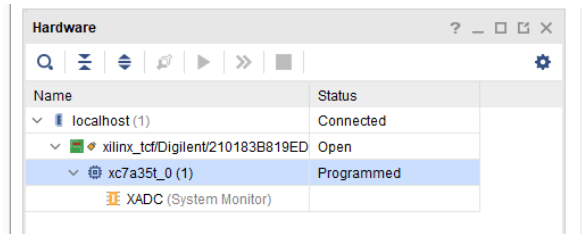
- Step 7: Timing and Area Analysis
 - Analyze if there is a timing/area problem.



FPGA Design Procedure

□ Step 8: Generate Bitstream

- Generate the Bitstream based on the implemented design.
- Program the hardware to make it run on a physical device



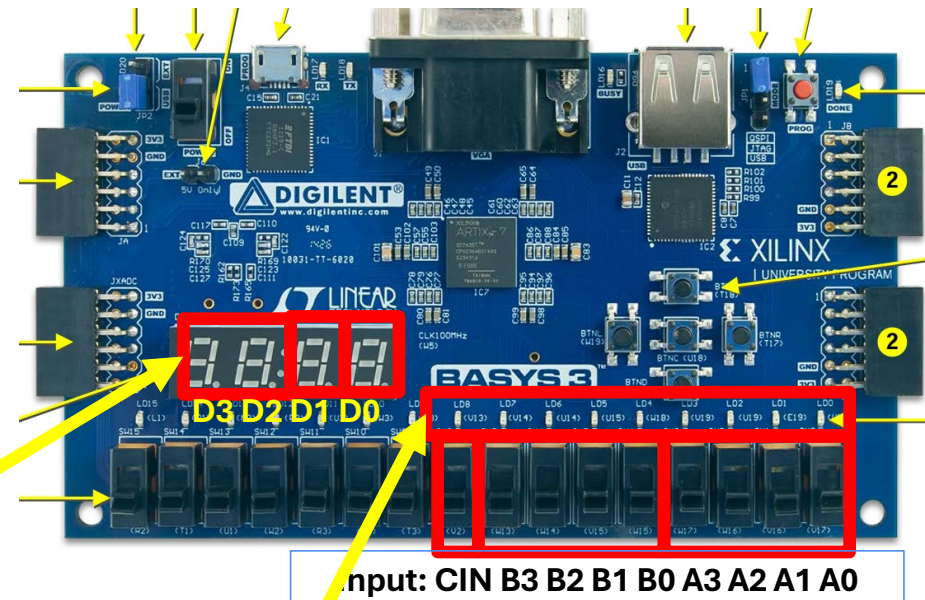
Part 2

LAB 1: IMPLEMENT 4-BIT FULL ADDER ON BASYS3



Lab 1: Implement 4-bit full adder on BASYS3

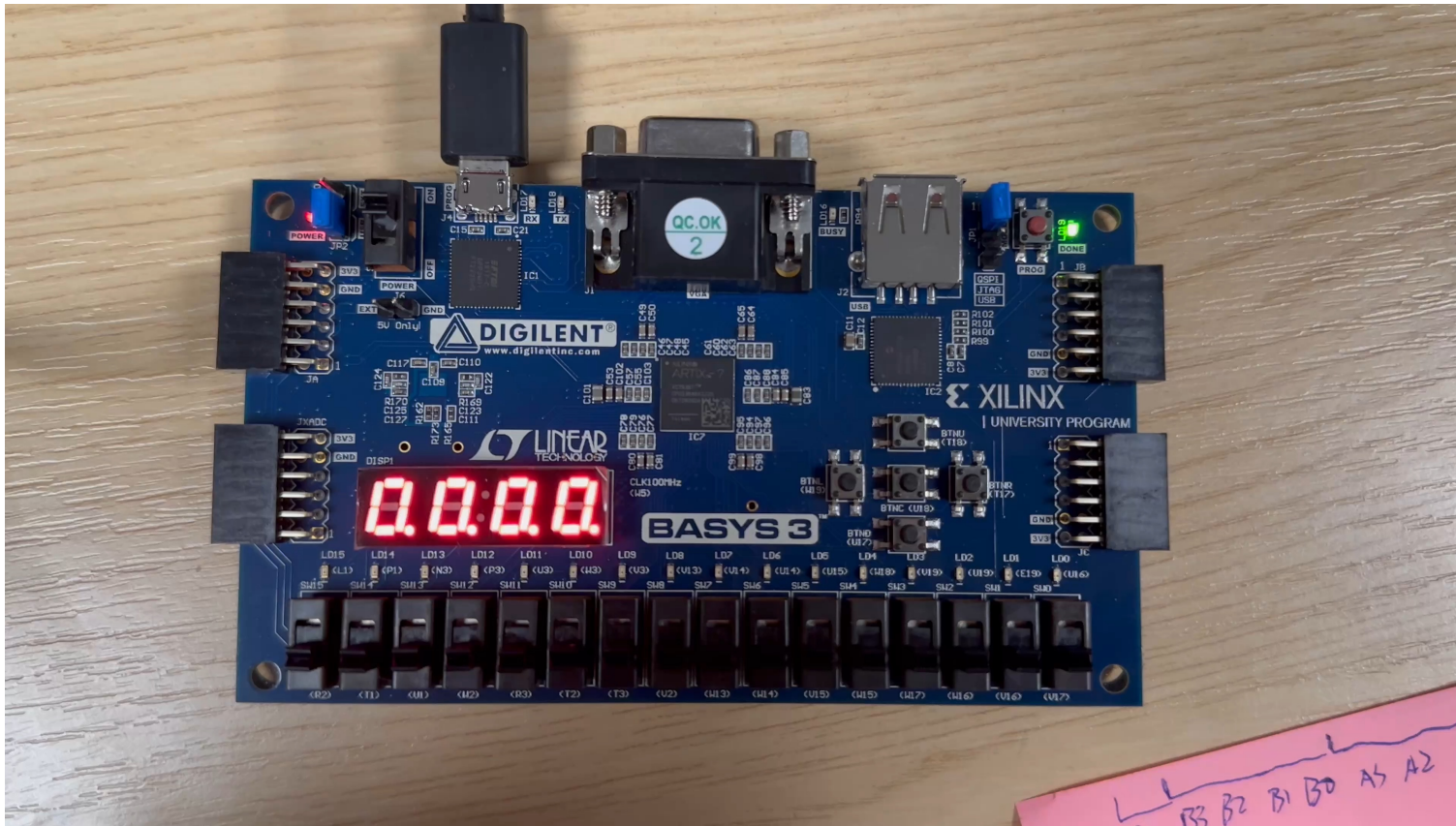
- In this lab, you will design, simulate, synthesize, implement, and program a ripple-carry 4-bit full adder on the BASYS3 FPGA board using the standard Vivado workflow.
- You'll write RTL for the adder, verify it with a simple testbench, apply pin and timing constraints to map inputs (board switches) and outputs (LEDs), including carry-out, generate the bitstream, and finally download it to the board to validate correct operation with on-board I/O.



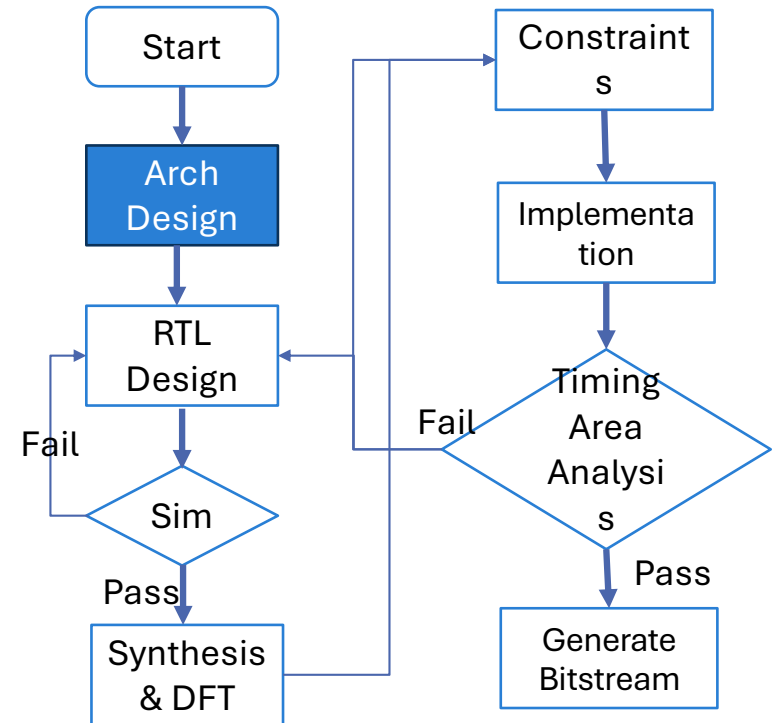
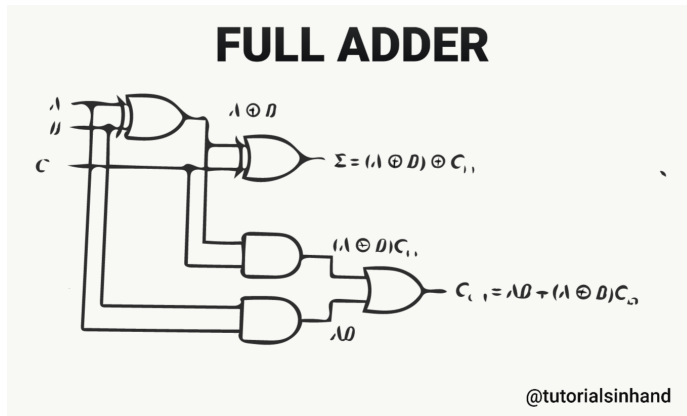
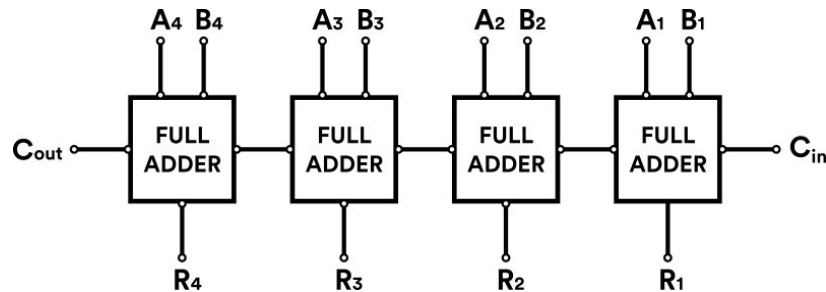
D3: Show the Carry-out
D2: Sum, $D2 = A+B$
D1: Show input B
D0: Show input A

LED: ON when switch is ON (UP)

Lab 1: Implement 4-bit full adder on BASYS3



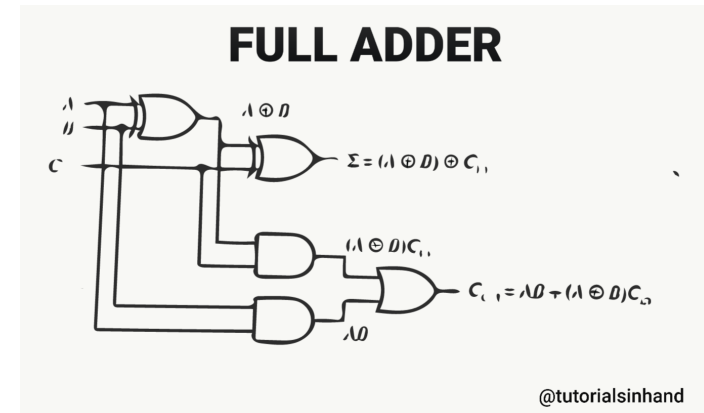
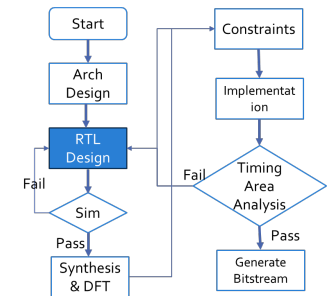
Background: 4-bit full adder



Background: 4-bit full adder

full_adder.v

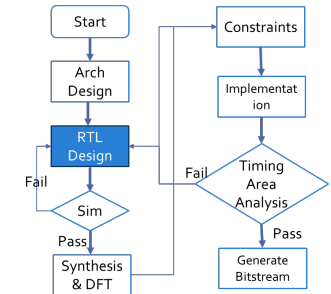
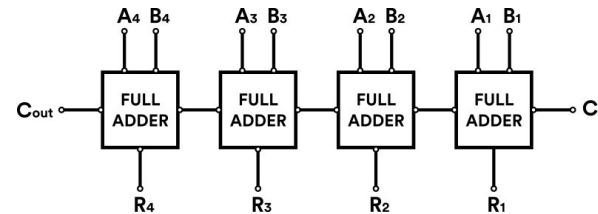
```
module full_adder (  
    input wire a,           // First input bit  
    input wire b,           // Second input bit  
    input wire cin,        // Carry-in bit  
    output wire sum,       // Sum output bit  
    output wire cout       // Carry-out bit  
);  
  
    // Internal signals for intermediate calculations  
    wire ab_xor;           // XOR of a and b  
    wire ab_and;           // AND of a and b  
    wire axor_cin_and;     // AND of (a XOR b) and cin  
  
    // Calculate sum as XOR of a, b, and cin  
    assign ab_xor = a ^ b;  
    assign sum = ab_xor ^ cin;  
  
    // Calculate carry-out as OR of (a AND b) and ((a XOR b) AND cin)  
    assign ab_and = a & b;  
    assign axor_cin_and = ab_xor & cin;  
    assign cout = ab_and | axor_cin_and;  
  
endmodule
```



Background: 4-bit full adder

full_adder_4bit.v

```
module full_adder_4bit(  
    input wire [3:0] a,           // First 4-bit input  
    input wire [3:0] b,           // Second 4-bit input  
    input wire cin,              // Carry-in bit  
    output wire [3:0] sum,        // 4-bit Sum output  
    output wire cout             // Carry-out bit  
);  
  
    // Internal carry signals between the full adders  
    wire c1, c2, c3;  
  
    // Instantiate four 1-bit full adders  
    full_adder fa0 (.a(a[0]),.b(b[0]),.cin(cin),.sum(sum[0]),.cout(c1));  
    full_adder fa1 (.a(a[1]),.b(b[1]),.cin(c1),.sum(sum[1]),.cout(c2));  
    full_adder fa2 (.a(a[2]),.b(b[2]),.cin(c2),.sum(sum[2]),.cout(c3));  
    full_adder fa3 (.a(a[3]),.b(b[3]),.cin(c3),.sum(sum[3]),.cout(cout));  
endmodule
```



Simulation: Testbench for 4-bit full adder

□ full_adder_4bit_tb.v

```
module full_adder_4bit_tb;

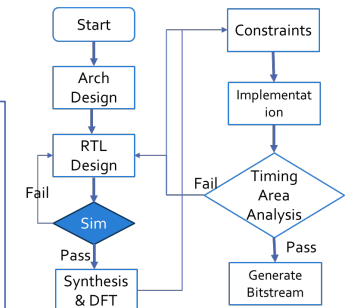
// Testbench signals
reg [3:0] a;
reg [3:0] b;
reg cin;
wire [3:0] sum;
wire cout;

wire [3:0] expected_sum;
wire expected_cout;

assign {expected_cout, expected_sum} = a + b + cin;

// Instantiate the 4-bit full adder module
full_adder_4bit dut
(.a(a),.b(b),.cin(cin),.sum(sum),.cout(cout));
```

```
// Test procedure
initial begin
// Test various combinations of inputs
a = 4'b0000; b = 4'b0000; cin = 0; #10;
a = 4'b0001; b = 4'b0010; cin = 0; #10;
a = 4'b0011; b = 4'b0101; cin = 1; #10;
a = 4'b1111; b = 4'b0001; cin = 0; #10;
a = 4'b1010; b = 4'b0101; cin = 1; #10;
a = 4'b1111; b = 4'b1111; cin = 1; #10;
// Finish simulation
$finish;
end
endmodule
```



I/O Mapping and TOP file

- ❑ The TOP file is the essential module for ALL FPGA PROJECTS.
- ❑ It's an RTL module that
 - Wrap up ALL submodules
 - Interfacing with FPGA IO

- ❑ Design Rules for TOP module:
 - Clearly define the module IO **BASED ON PHYSICAL DATASHEET!**
 - Only implement the modules and connect them using wires in the top module.
 - You can define constants and parameters in the top module.
 - You should not write any logic in the top module, though it's not wrong. It's just not a good coding style.

I/O Mapping and TOP file

- The TOP file for this lab:

```
module top(  
    input      clk,           // Clock input  
    input [15:0] sw,         // Input switches  
    output CA, CB, CC, CD, CE, CF, CG, DP, // 7-segment  
    display segments  
    output [3:0] AN,         // 7-segment display anodes  
    output [15:0] led        // LED outputs to show the  
    result  
);  
  
//Wires and signals  
wire [3:0] a;  
wire [3:0] b;  
wire      cin;  
wire [3:0] sum;  
wire      cout;  
  
// Assign inputs from switches  
assign a  = sw[3:0];  
assign b  = sw[7:4];  
assign cin = sw[8];  
  
// Display inputs on LEDs  
assign led = {7'b0, cin, b, a};  
//continue...  
  
// Instantiate seg7 display module  
seg7display u0 (  
    .x_l({3'b0, cout, sum, b, a}),  
    .clk(clk),  
    .reset(1'b0),  
    .a_to_g({CA, CB, CC, CD, CE, CF, CG}),  
    .an_l(AN),  
    .dp_l(DP)  
);  
  
// Instantiate full adder 4 bit module  
full_adder_4bit u1 (  
    .a(a),  
    .b(b),  
    .cin(cin),  
    .sum(sum),  
    .cout(cout)  
);  
  
endmodule
```

I/O Mapping and TOP file

□ The TOP file for this lab:

```

module top(
  input      clk,           // Clock input
  input [15:0] sw,         // Input switches
  output CA, CB, CC, CD, CE, CF, CG, DP, // 7-segment display segments
  output [3:0] AN,         // 7-segment display anodes
  output [15:0] led        // LED outputs to show the result
);

```

4 Oscillators/Clocks

The Basys3 board includes a single 100 MHz oscillator connected to **pin W5** (W5 is a MRCC input on bank 34). The input clock can drive MMCMs or PLLs to generate clocks of various frequencies and with known phase relationships that may be needed throughout a design. Some rules restrict which MMCMs and PLLs may be driven by the 100 MHz input clock. For a full description of these rules and of the capabilities of the Artix-7 clocking resources, refer to the “7 Series FPGAs Clocking Resources User Guide” available from Xilinx.

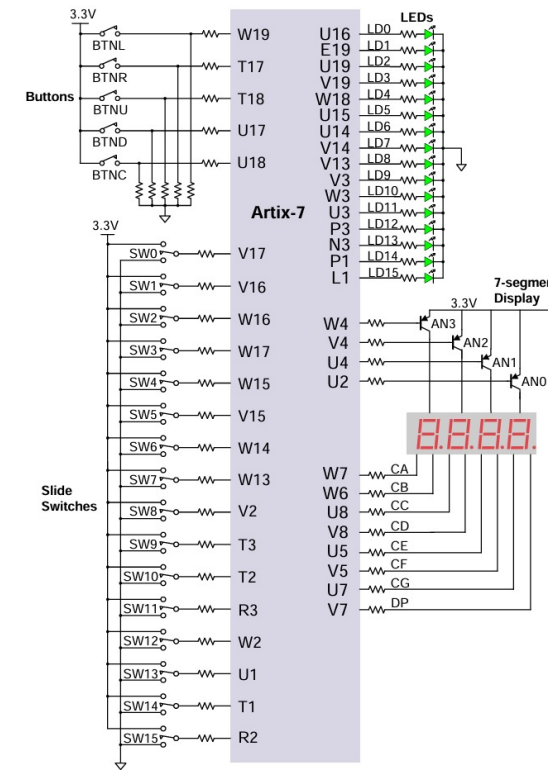
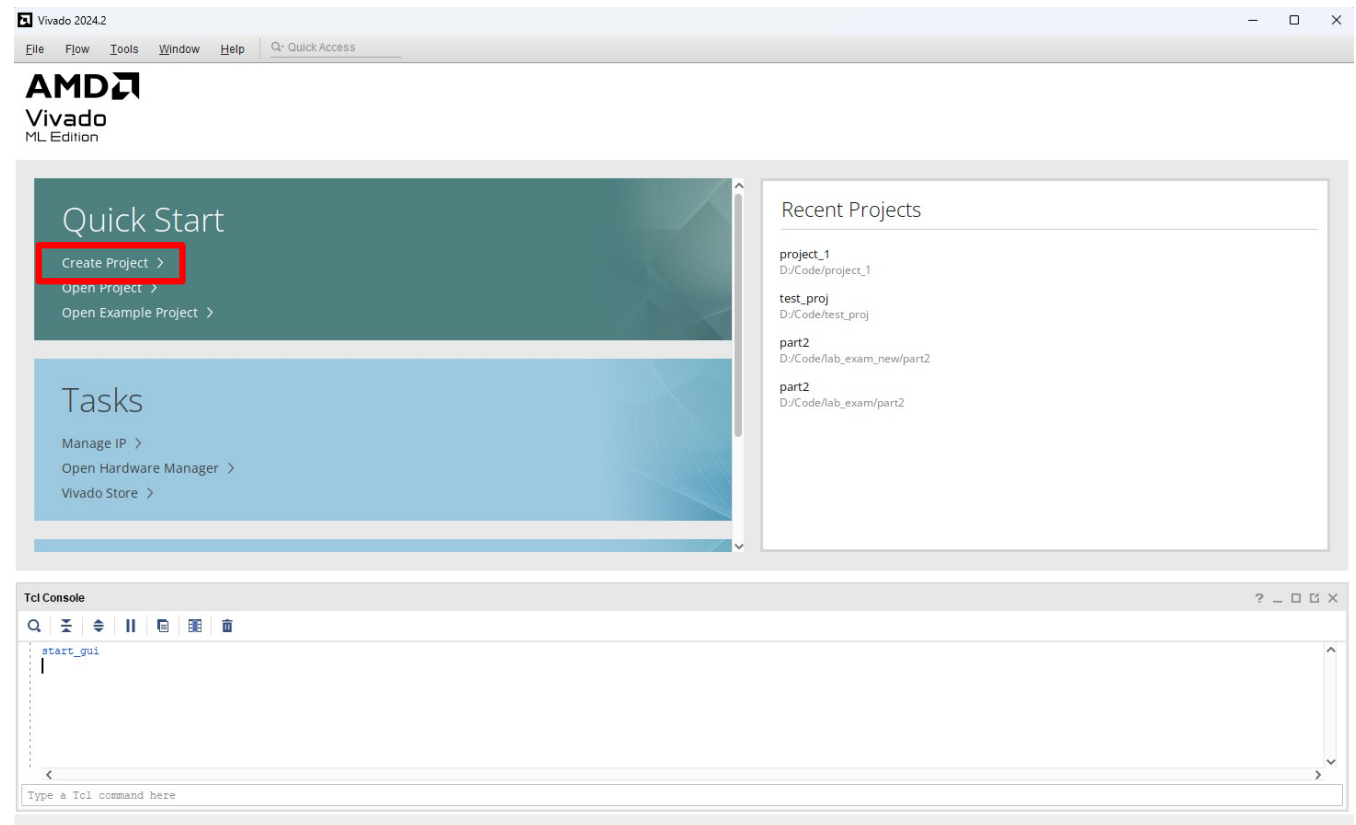


Figure 16. General purpose I/O devices on the Basys3.

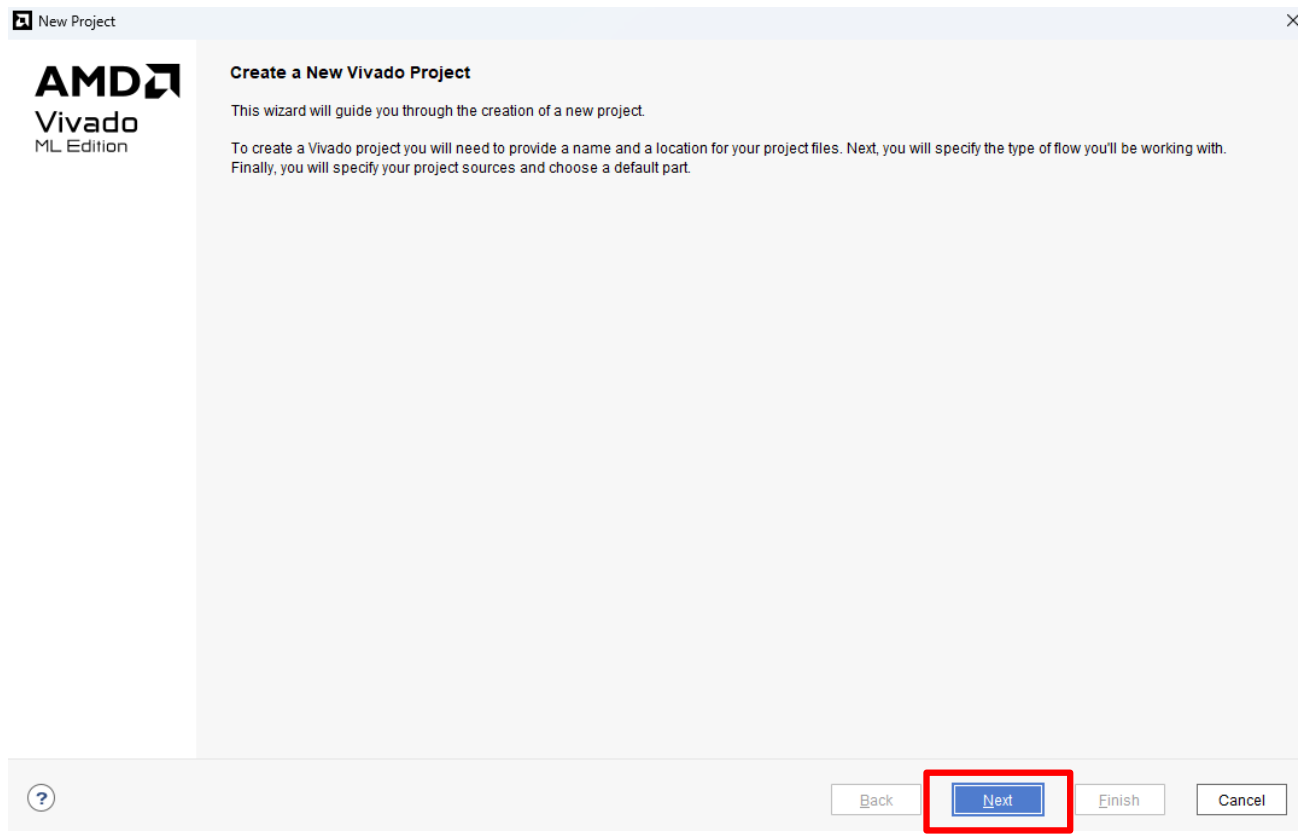
Simulation – Start with VIVADO

- Start Vivado, click “Create Project”



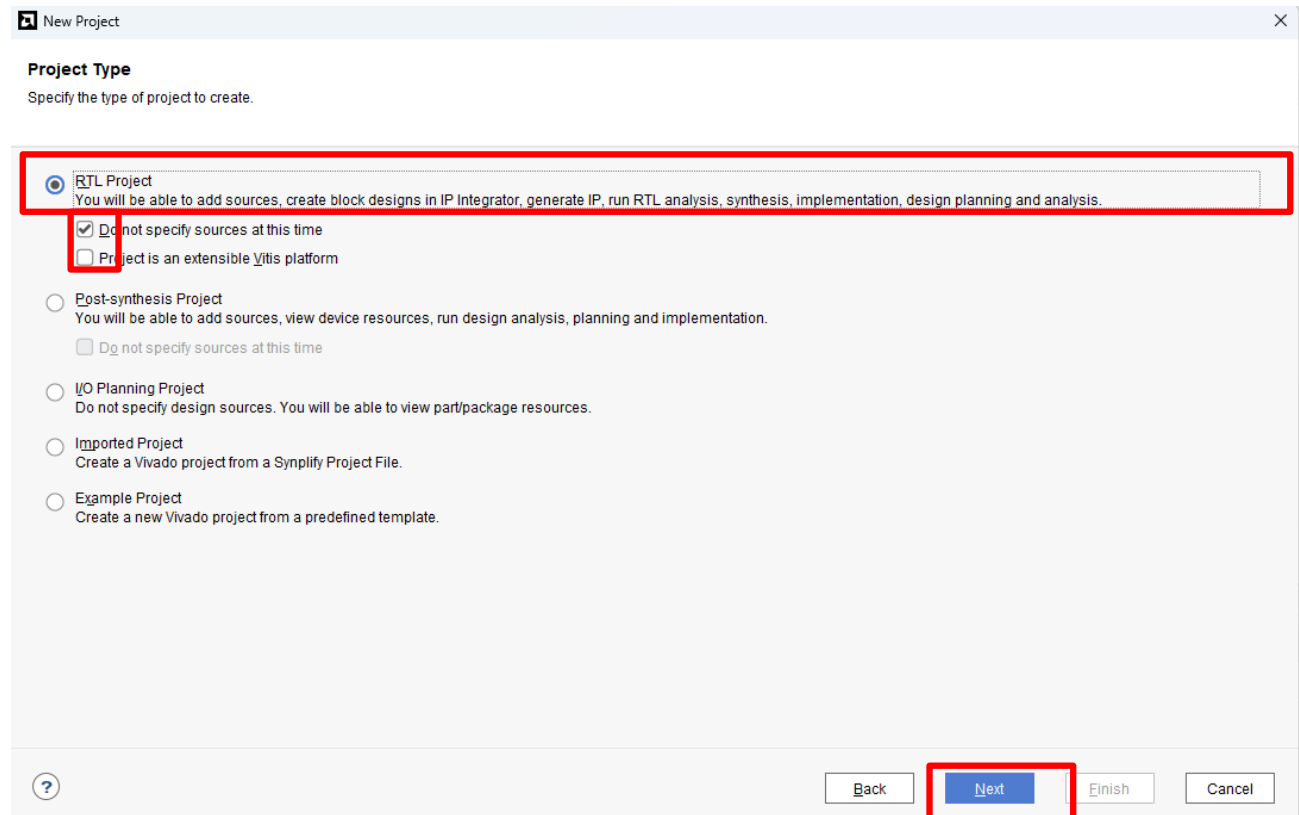
Simulation – Start with VIVADO

- ❑ In the New Project wizard, click “next”



Simulation – Start with VIVADO

- ❑ In the New Project wizard, choose RTL project. Check “Do not specify sources at this time”.



Simulation – Start with Vivado

- ❑ In the New Project wizard, choose **xc7a35tcpg236-1**. This is the FPGA device used on your BASYS3.
- ❑ Click next

New Project

Default Part
Choose a default AMD part or board for your project.

Parts | Boards

Reset All Filters

Category: All Package: All Temperature: All
Family: All Speed: All Static power: All

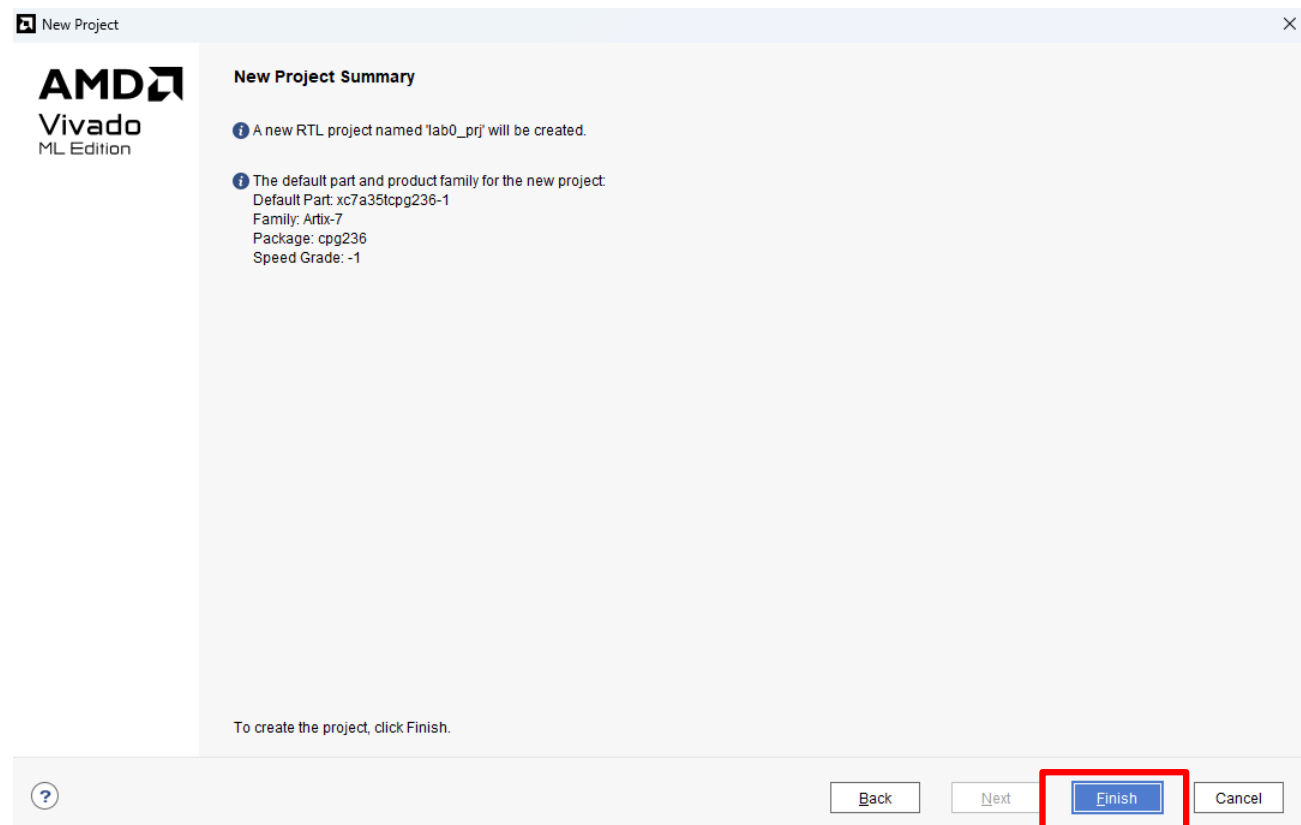
Search: Q-

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	BUFGs	Gb Transceivers	GTPE:
xc7a35tcpg236-3	236	106	20800	41600	50	0	90	32	2	2
xc7a35tcpg236-2	236	106	20800	41600	50	0	90	32	2	2
xc7a35tcpg236-2L	236	106	20800	41600	50	0	90	32	2	2
xc7a35tcpg236-1	236	106	20800	41600	50	0	90	32	2	2
xc7a35tcs g324-3	324	210	20800	41600	50	0	90	32	0	0
xc7a35tcs g324-2	324	210	20800	41600	50	0	90	32	0	0
xc7a35tcs g324-2L	324	210	20800	41600	50	0	90	32	0	0
xc7a35tcs g324-1	324	210	20800	41600	50	0	90	32	0	0
xc7a35tcs g325-3	325	150	20800	41600	50	0	90	32	4	4
xc7a35tcs g325-2	325	150	20800	41600	50	0	90	32	4	4
xc7a35tcs g325-2L	325	150	20800	41600	50	0	90	32	4	4

Back Next Finish Cancel

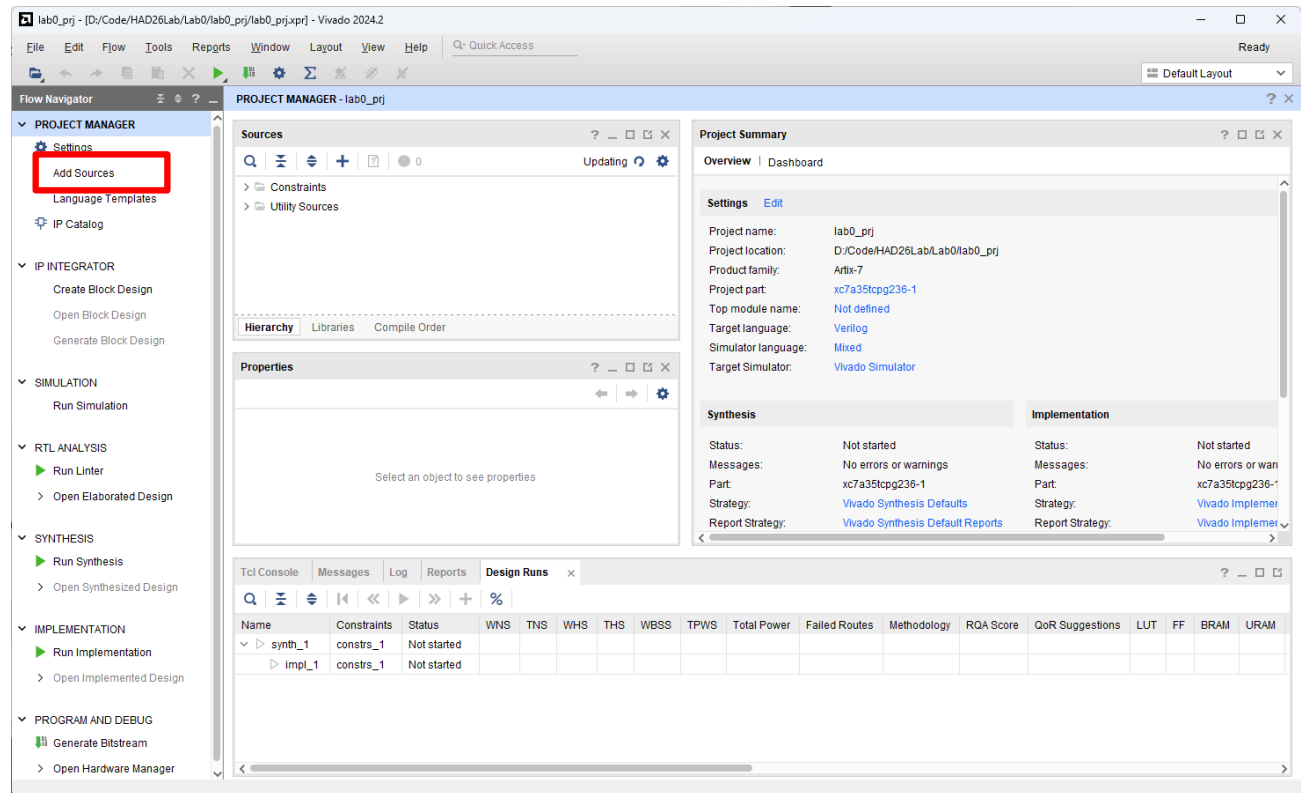
Simulation – Start with VIVADO

- ❑ In the New Project wizard, click “Finish”.



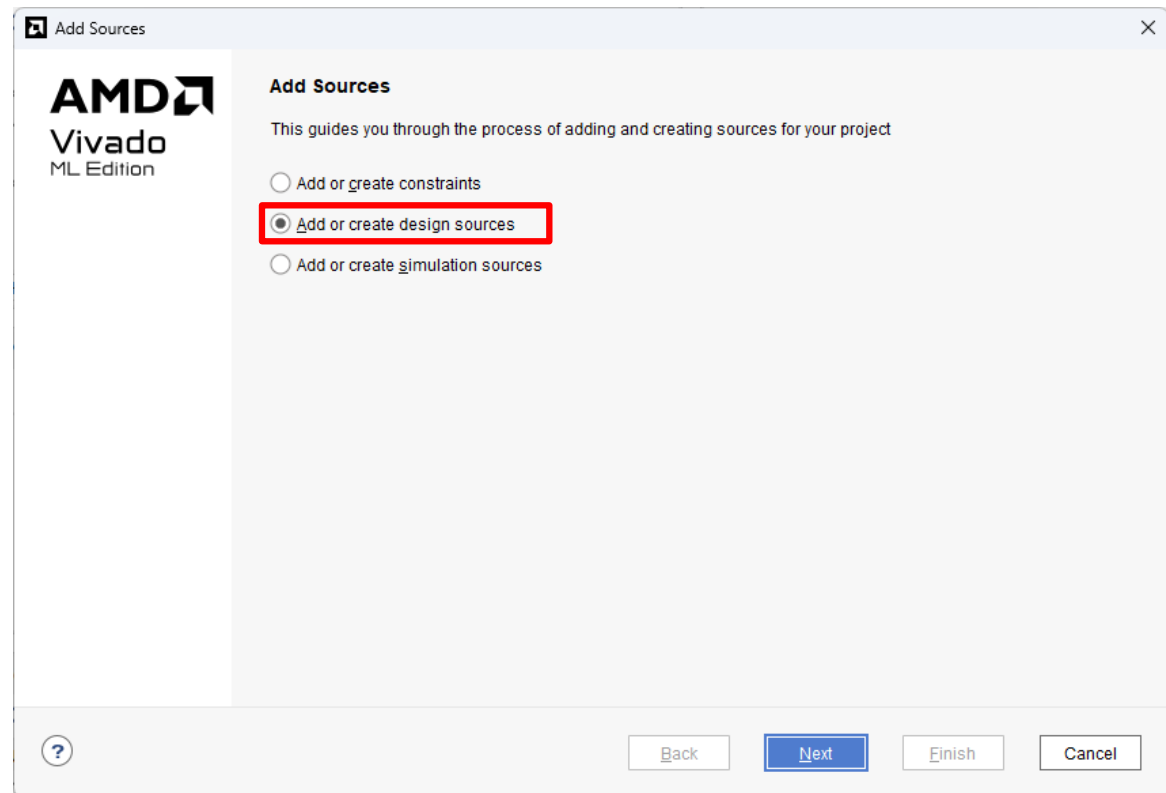
Simulation – Start with Vivado.

- ❑ This is the workspace for your Vivado.
- ❑ On the left if your flow navigator, basically is the same workflow as our FPGA design workflow.
- ❑ Click add sources. I have provided the codes required for this lab. Import using “Add Sources”



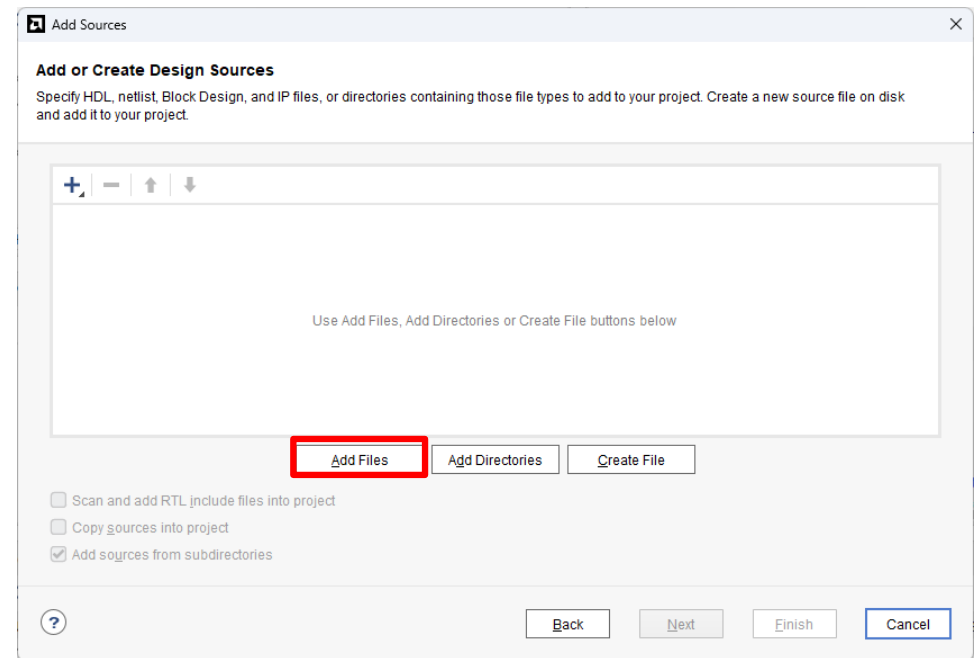
Simulation – Start with VIVADO

- ❑ In the Add Sources wizard, choose “Add or create design sources”, then click next



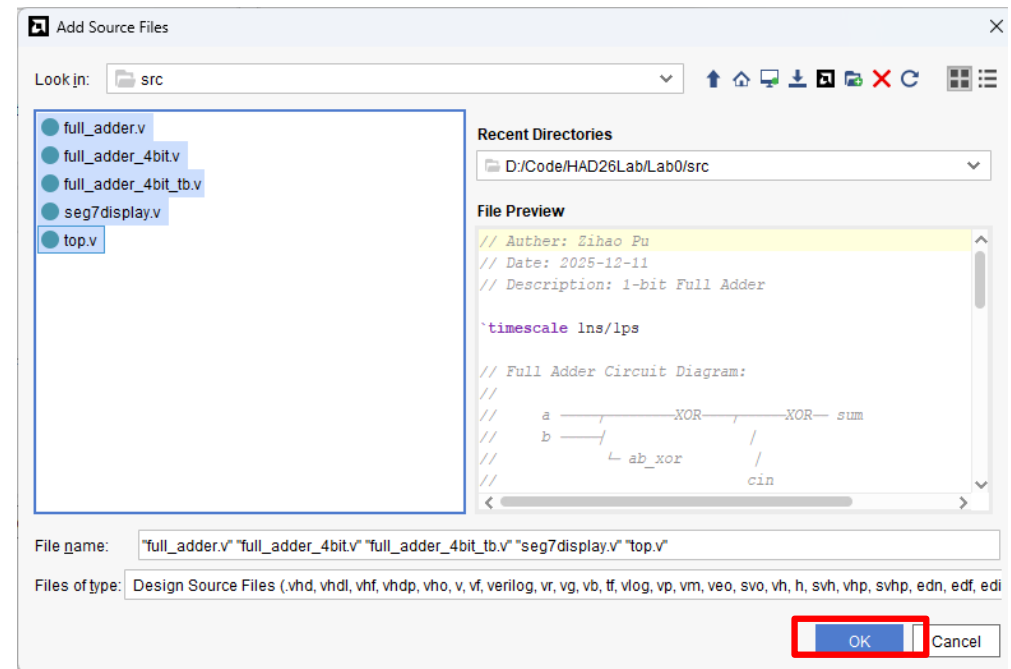
Simulation – Start with VIVADO

- ❑ In the Add Sources wizard, click “Add files” to import the source files.



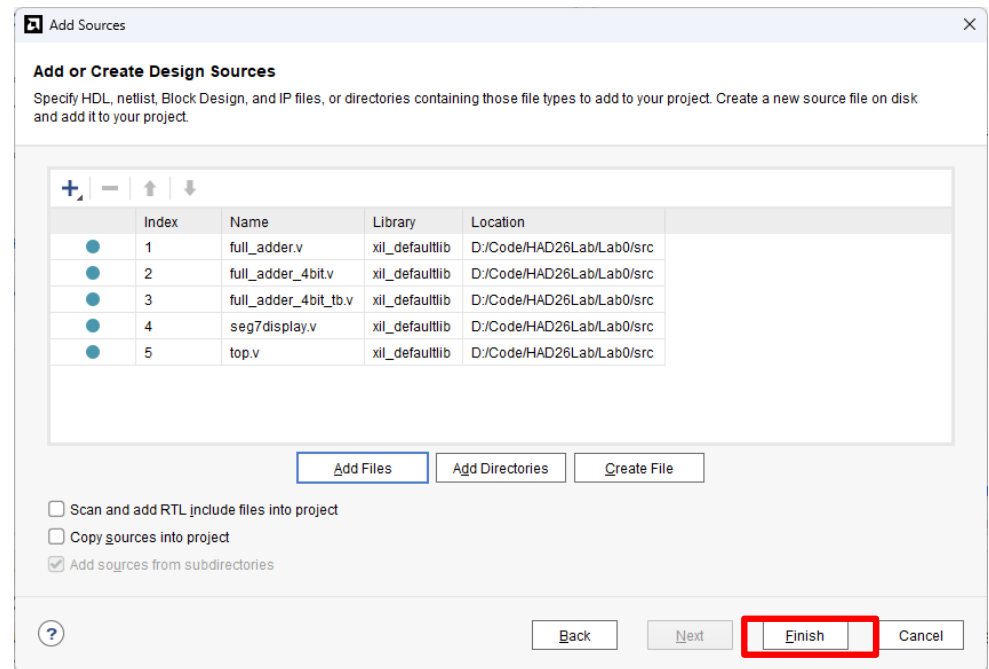
Simulation – Start with VIVADO

- ❑ In the Add Source File wizard, navigate to the source files, and choose all the provided source files.
- ❑ Then click OK



Simulation – Start with VIVADO

- ❑ In the Add Source wizard, you will see the imported source files.
- ❑ Then click Finish



Simulation – Starting with Vivado

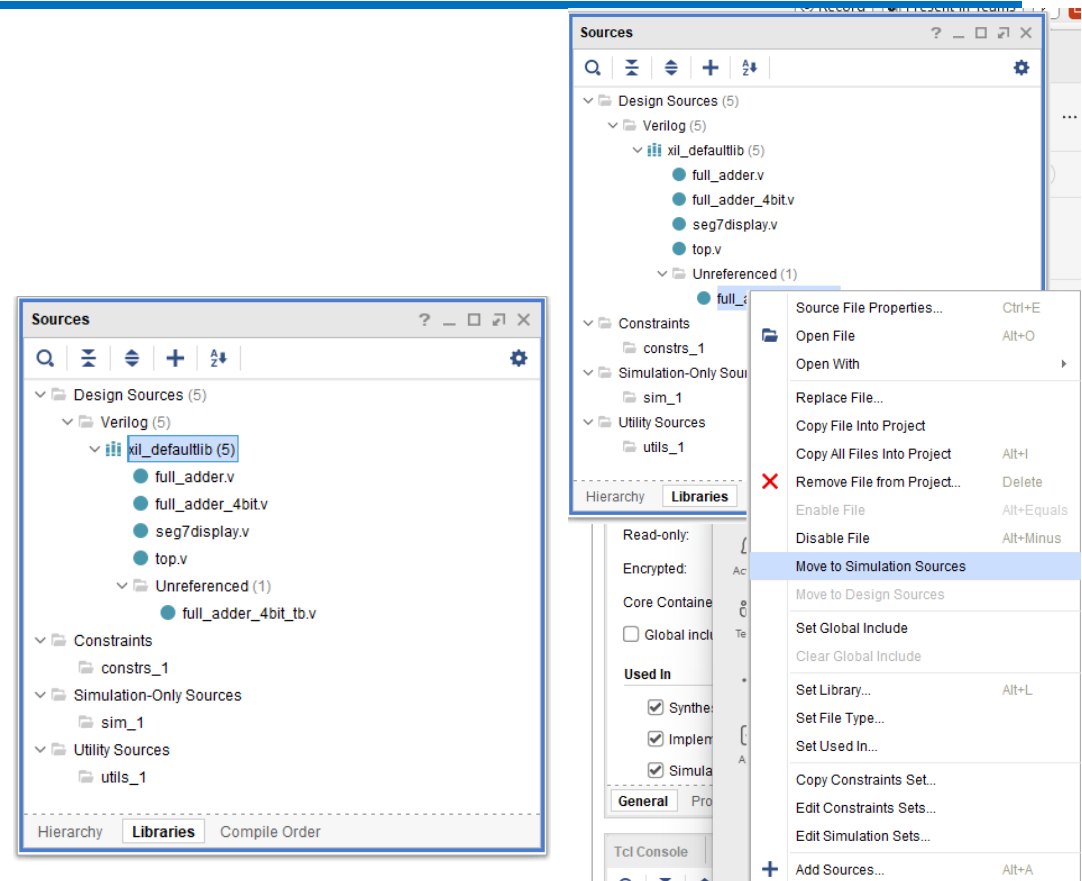
- ❑ In the Vivado workspace, you will see the imported modules in the “Source” window.
- ❑ Note: the imported sources are automatically ordered in the “hierarchy” tab.
- ❑ In order to see the files, click “Libraries” tab.

The screenshot displays the Vivado 2024.2 workspace for a project named 'lab0_proj'. The interface is divided into several panes:

- Flow Navigator:** Shows the project workflow, including Settings, Add Sources, Language Templates, IP Catalog, IP INTEGRATOR, SIMULATION, RTL ANALYSIS, SYNTHESIS, IMPLEMENTATION, and PROGRAM AND DEBUG.
- PROJECT MANAGER - lab0_proj:** The central pane showing the project structure. The 'Sources' tab is active, displaying a tree view of Design Sources (top, full_adder_4bit_tb), Constraints, Simulation Sources (sim_1, top, full_adder_4bit_tb), and Utility Sources. The 'full_adder_4bit_tb' source is selected.
- Hierarchy:** A tab at the bottom of the Project Manager pane, currently selected and highlighted with a red box. It shows the source file properties for 'full_adder_4bit_tb.v', including 'Enabled' and 'General' properties.
- Libraries:** A tab at the bottom of the Project Manager pane, currently unselected.
- Project Summary:** A pane on the right showing project details such as Project name, Project location, Product family, Project part, Top module name, Target language, Simulator language, and Target Simulator.
- Design Runs:** A table at the bottom showing the status of design runs. The table has columns for Name, Constraints, Status, WNS, TNS, WHS, THS, WBSS, TPWS, Total Power, Failed Routes, Methodology, RQA Score, QoR Suggestions, LUT, FF, and BRAM. The table shows two runs: 'synth_1' and 'impl_1', both with 'Not started' status.

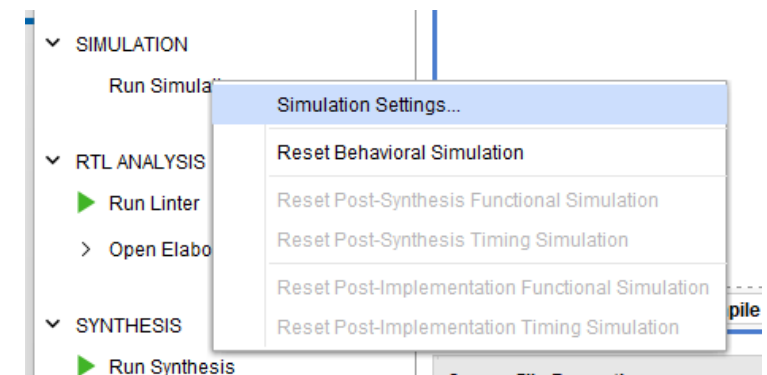
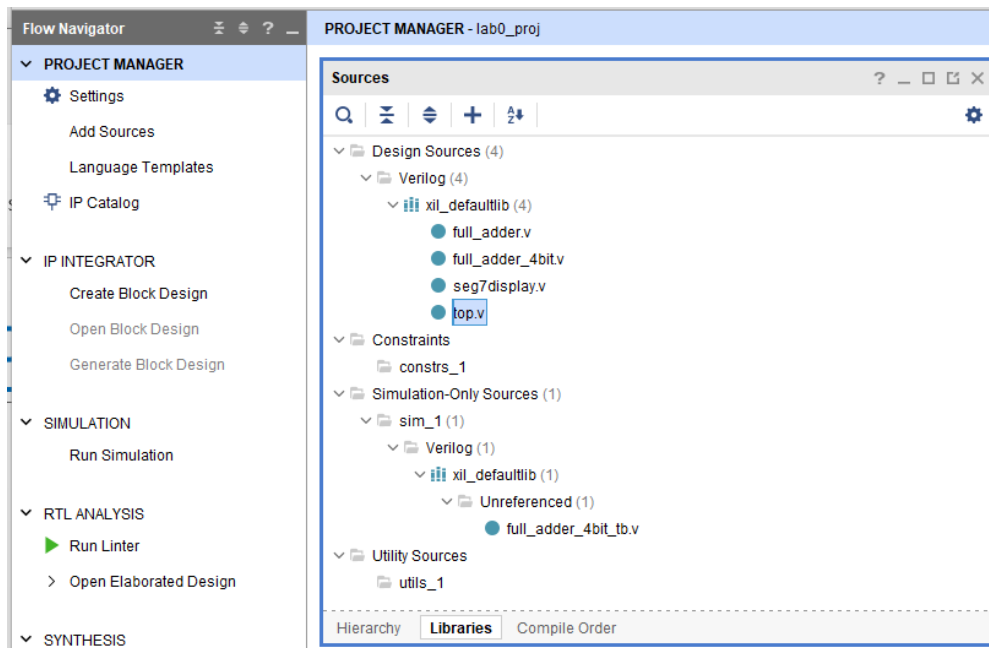
Simulation – Starting with Vivado Project

- ❑ In the Libraries tab, click “xil_defaultlib” to expand the library, then click the “Unreferenced” tab to expand the folder. You will see all the imported files.
- ❑ Right-click “full_adder_4bit_tb.v”, click “Move to Simulation Sources”.
- ❑ Testbenches are not synthesizable. Make sure they are not part of “Design Sources”.



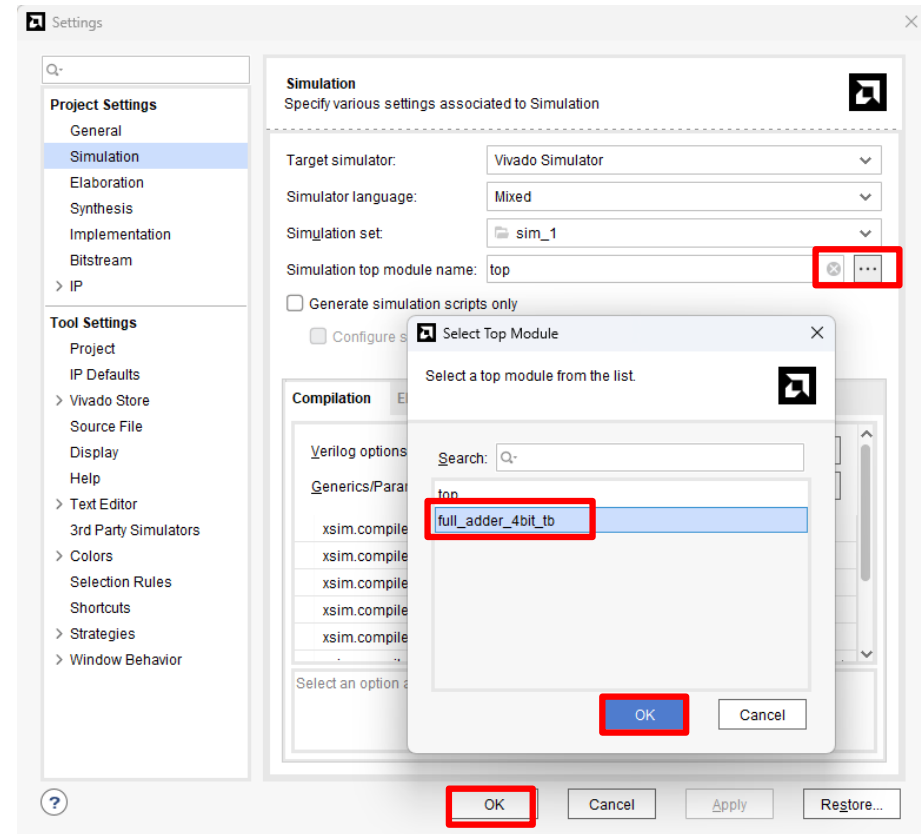
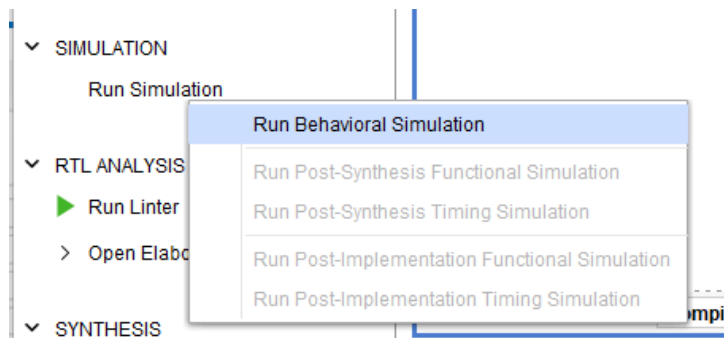
Simulation – Starting with Vivado Project

- ❑ In the end, you will see the library like this. Right-click “Run Simulation” to enter “Simulation Settings...”



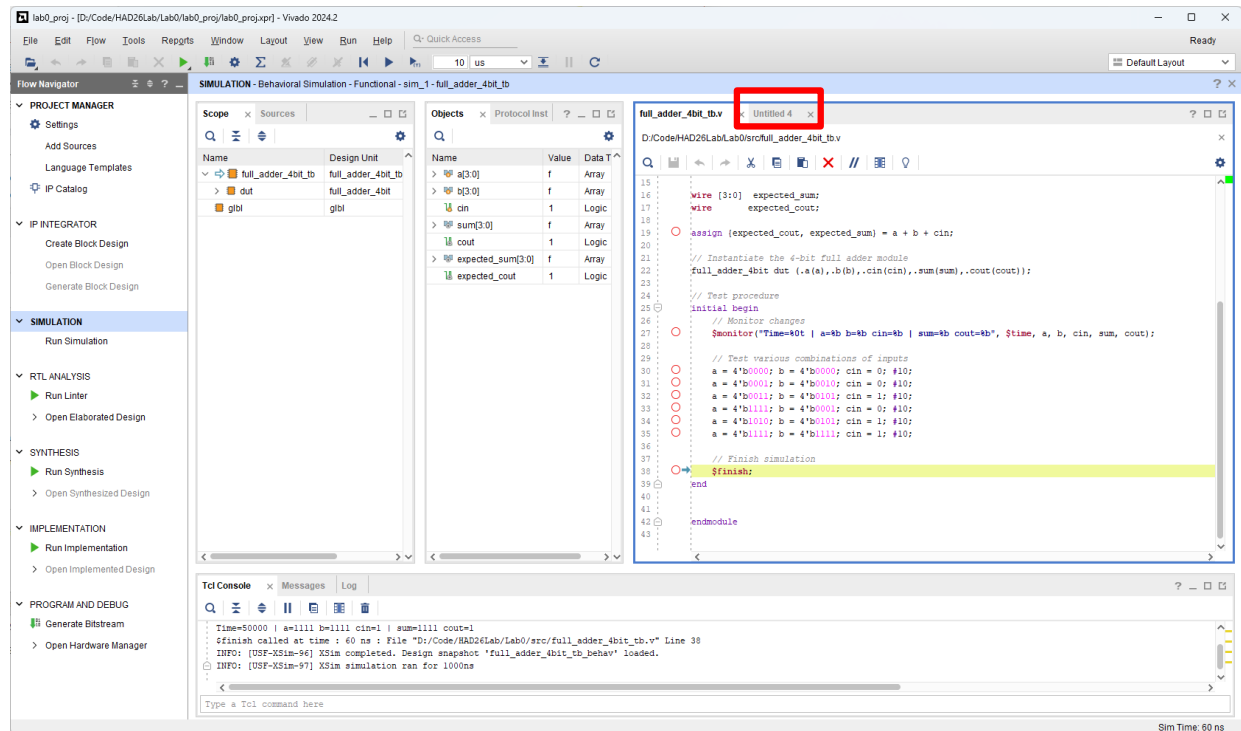
Simulation – Starting with Vivado Project

- ❑ In the “Setting” wizard, click “...” to change the simulation top module name, then choose “full_adder_4bit_tb”. Click OK then click OK.
- ❑ After settings, click “Run Simulation” to get started.



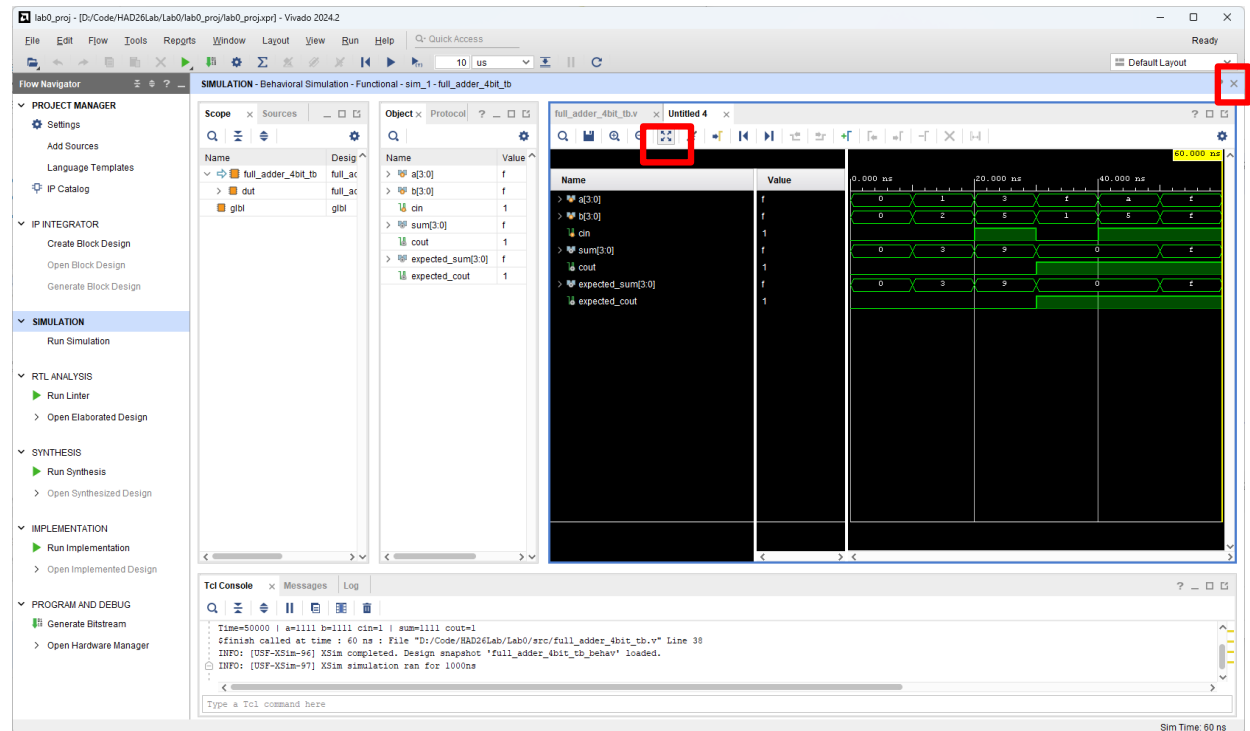
Simulation – Starting with Vivado Project

- ❑ Then you will enter the simulation window. Click “Untitled 1” to enter the waveform window.
- ❑ Note: The waveform window name changes as you run multiple times of simulations.



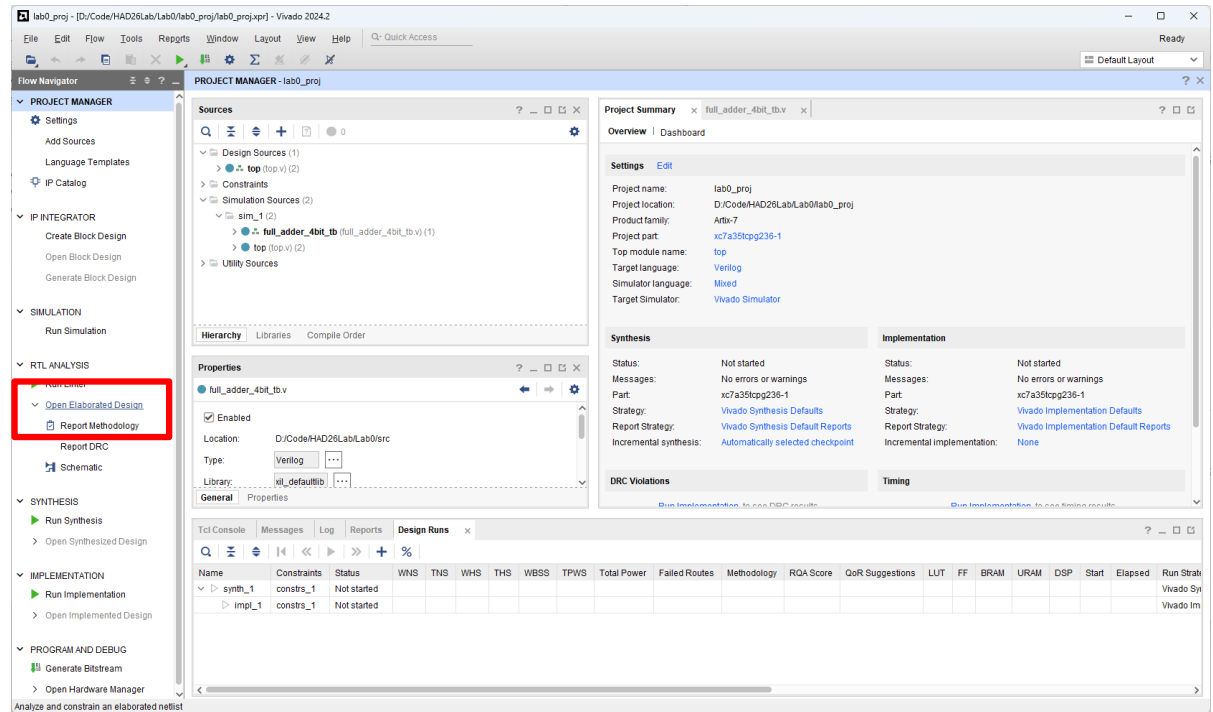
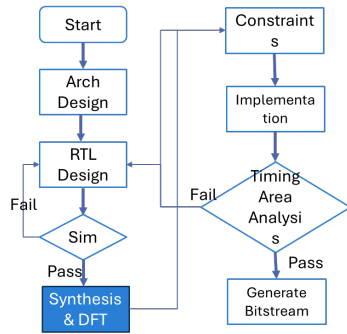
Simulation – Starting with Vivado Project

- ❑ Click “Zoom Fit” to show the complete waveform. You will see all the interesting signals.
- ❑ Once you confirm that your signal is OK, click “x” on the top-right corner of the “SIMULATOR” window to quick simulation.



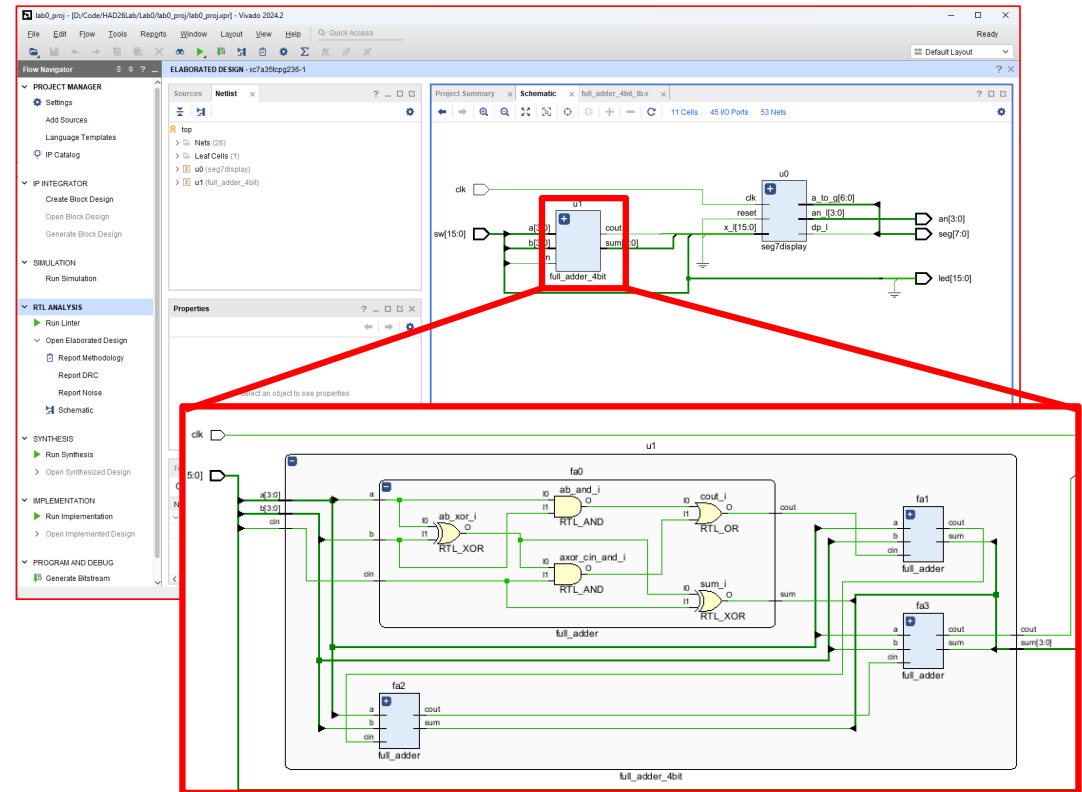
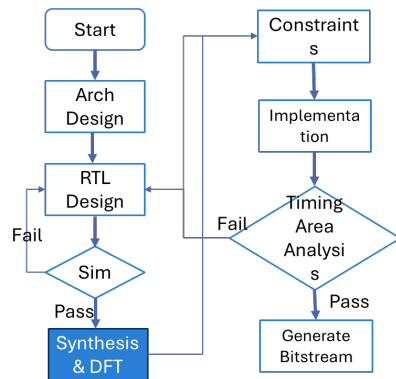
Starting with Vivado Project - Elaboration

- ❑ Click “Open Elaborated Design”, then click OK to start elaboration.
- ❑ This allows Vivado to analyze your code and make an elaborate block diagram.



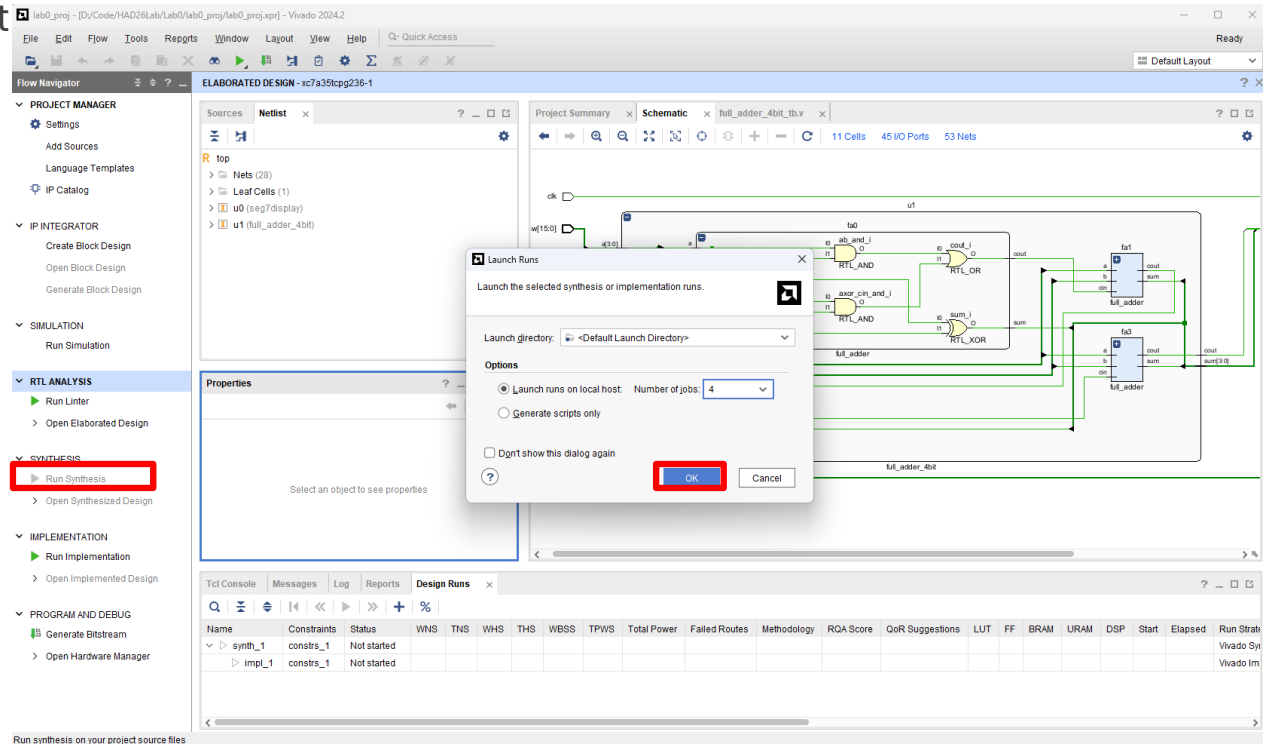
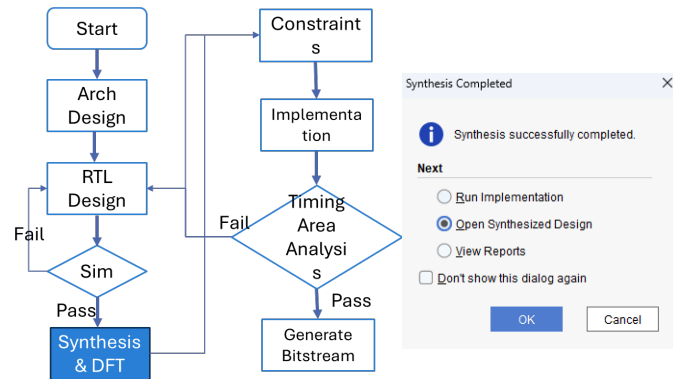
Starting with Vivado Project - Elaboration

- In the block diagram, click “+” can open the block and view the internal block diagram. Check if this is the logic you want.
- After you finished, click “x” on the top-right corner to close “Elaborated Design”



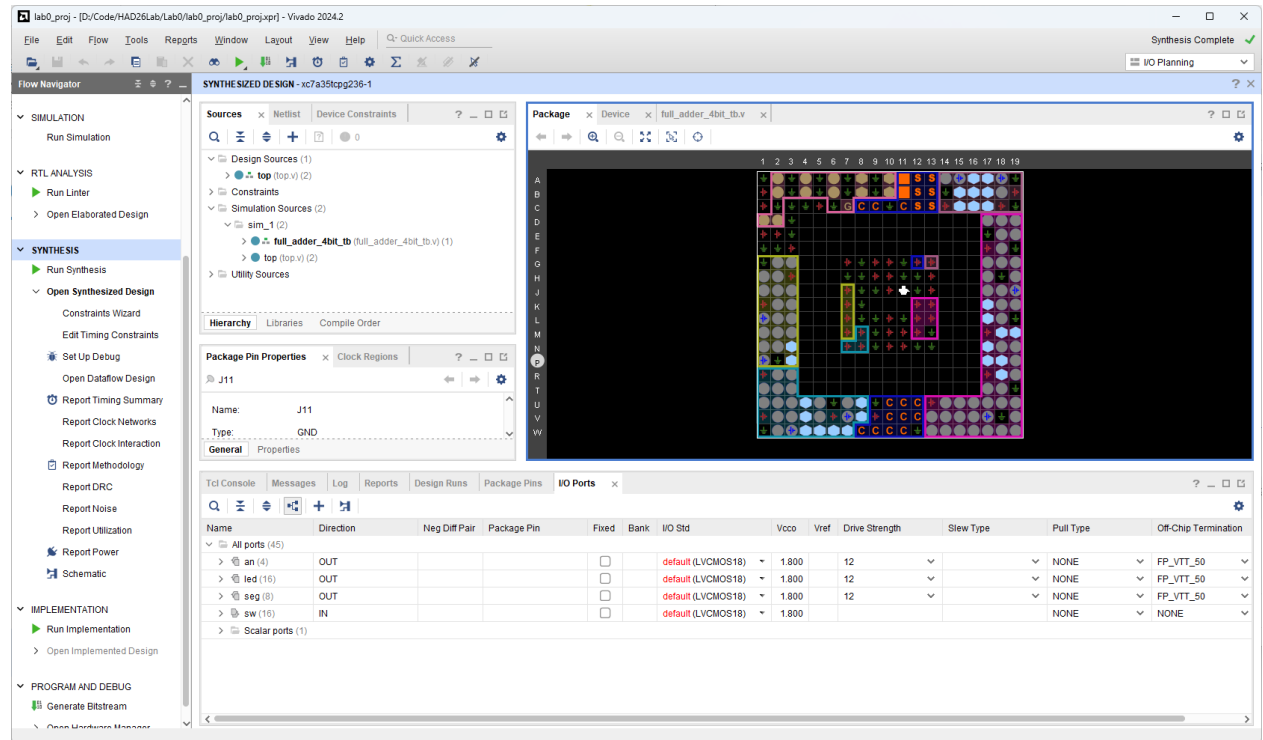
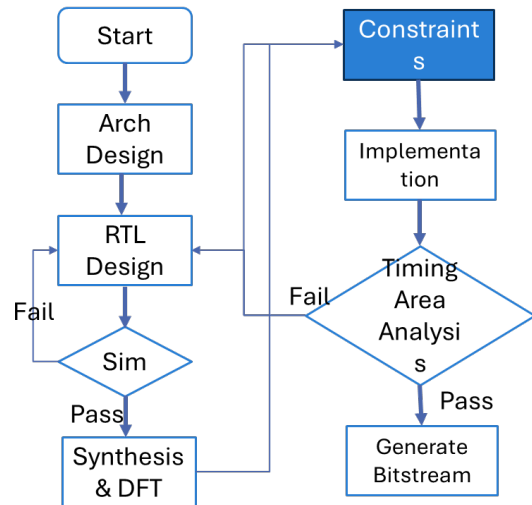
Starting with Vivado Project - Synthesis

- In the Flow Navigator, click “Run Synthesis”, then click OK to start synthesis. This process takes a while.
- When completed, choose “Open Synthesized Design”.



Starting with Vivado Project - Constraints

- The Synthesized Design view is on the right. For now, we are going to make two important constraints:
 - I/O mapping
 - Clock constraints



Starting with Vivado Project - Constraints

- ❑ The datasheet gives you how the **EXTERNAL** physical devices (e.g., LED) are connected to the FPGA. In the constraint file, you have to define how your **INTERNAL** module connects to the IO ports.
- ❑ Based on the BASYS3 Datasheet, we need to gather the following information:
 - 1. What are the associated PINs for each port? (check figure 16)
 - 2. What is the voltage standard for the PINs? (check Table 2)
 - 3.3V -> LVCMOS33 (Low Voltage CMOS 3.3V)

4 Oscillators/Clocks

The Basys3 board includes a single 100 MHz oscillator connected to **pin W5** (W5 is a MRCC input on bank 34). The input clock can drive MMCMs or PLLs to generate clocks of various frequencies and with known phase

Supply	Circuits	Device	Current (max/typical)
3.3V	FPGA I/O, USB ports, Clocks, Flash, PMODs	IC10: LTC3633	2A/0.1 to 1.5A
1.0V	FPGA Core	IC10: LTC3633	2A/ 0.2 to 1.3A
1.8V	FPGA Auxiliary and Ram	IC11: LTC3621	300mA/ 0.05 to 0.15A

Table 2. Basys3 power supplies.

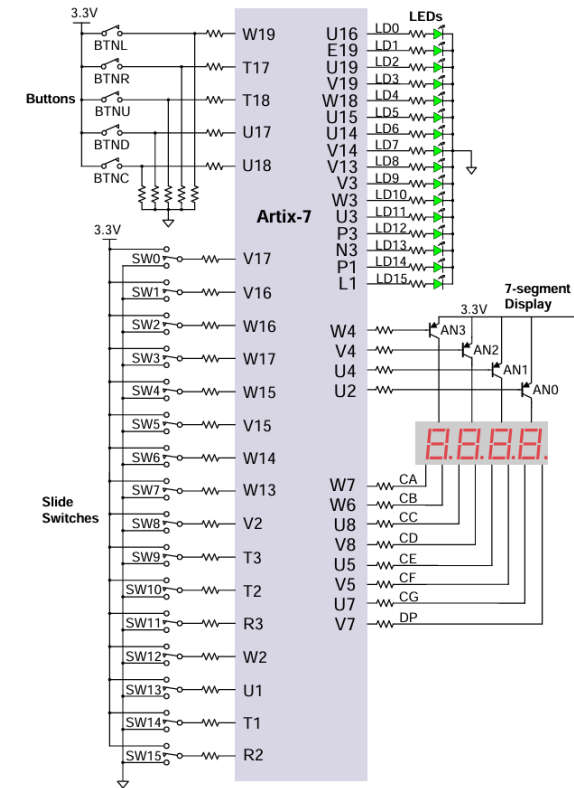


Figure 16. General purpose I/O devices on the Basys3.

Starting with Vivado Project - Constraints

- Based on the datasheet, fill the table in the “I/O Ports” window.

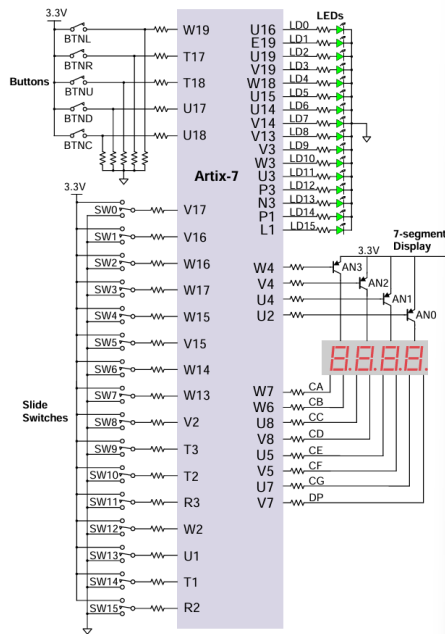
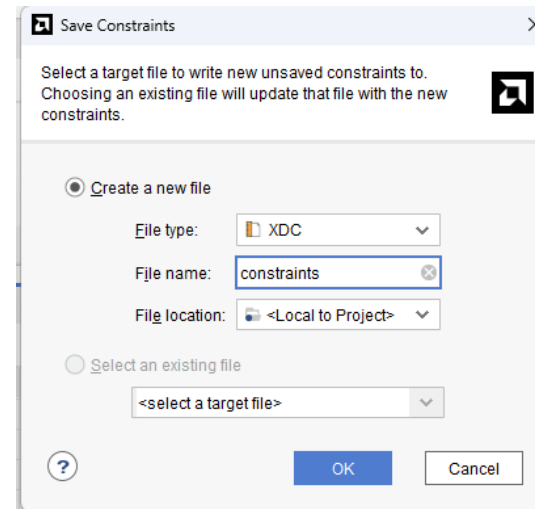
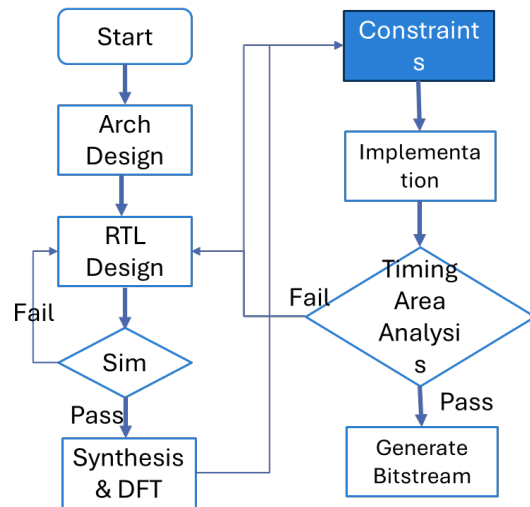


Figure 16. General purpose I/O devices on the Basys3.

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Stew Type	Pull Type	Off-Chip Terminator
AN[4]	OUT			<input checked="" type="checkbox"/>	34	LVCMOS33*	3.300	12			NONE	FP_VTT_50
AN[3]	OUT		W4	<input checked="" type="checkbox"/>	34	LVCMOS33*	3.300	12			NONE	FP_VTT_50
AN[2]	OUT		V4	<input checked="" type="checkbox"/>	34	LVCMOS33*	3.300	12			NONE	FP_VTT_50
AN[1]	OUT		U4	<input checked="" type="checkbox"/>	34	LVCMOS33*	3.300	12			NONE	FP_VTT_50
AN[0]	OUT		U2	<input checked="" type="checkbox"/>	34	LVCMOS33*	3.300	12			NONE	FP_VTT_50
led[15]	OUT			<input type="checkbox"/>		default (LVCMOS18)	1.800	12			NONE	FP_VTT_50
led[14]	OUT			<input type="checkbox"/>		default (LVCMOS18)	1.800	12			NONE	FP_VTT_50
led[13]	OUT			<input type="checkbox"/>		default (LVCMOS18)	1.800	12			NONE	FP_VTT_50
led[12]	OUT			<input type="checkbox"/>		default (LVCMOS18)	1.800	12			NONE	FP_VTT_50
led[11]	OUT			<input type="checkbox"/>		default (LVCMOS18)	1.800	12			NONE	FP_VTT_50
led[10]	OUT			<input type="checkbox"/>		default (LVCMOS18)	1.800	12			NONE	FP_VTT_50
led[9]	OUT			<input type="checkbox"/>		default (LVCMOS18)	1.800	12			NONE	FP_VTT_50
led[8]	OUT			<input type="checkbox"/>		default (LVCMOS18)	1.800	12			NONE	FP_VTT_50

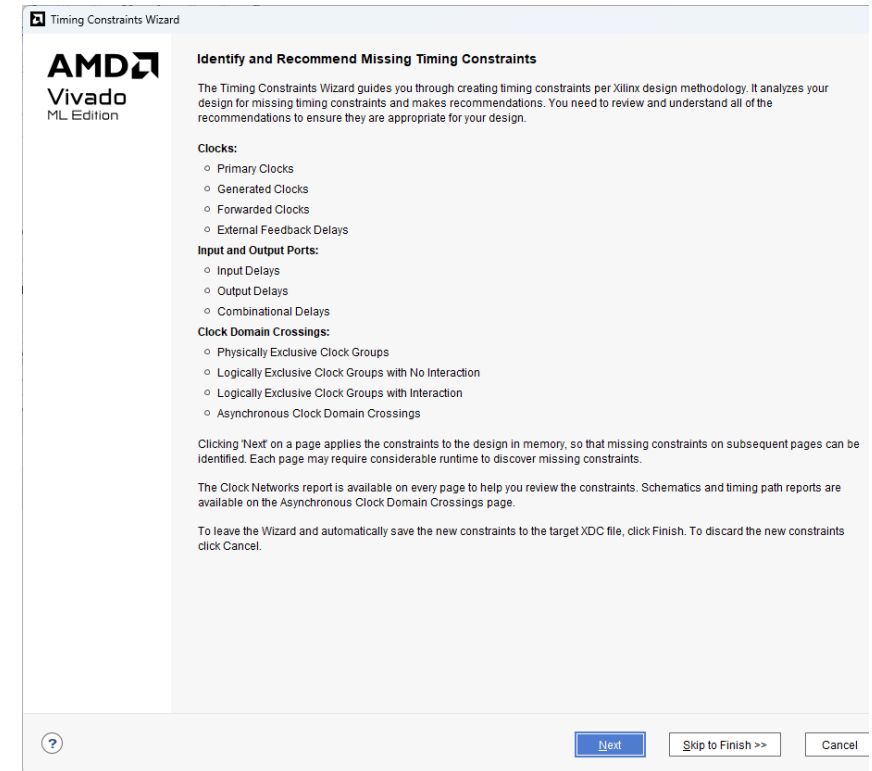
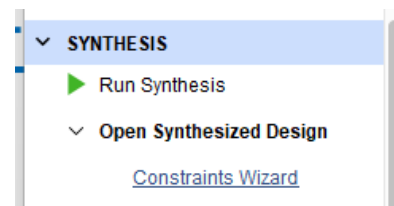
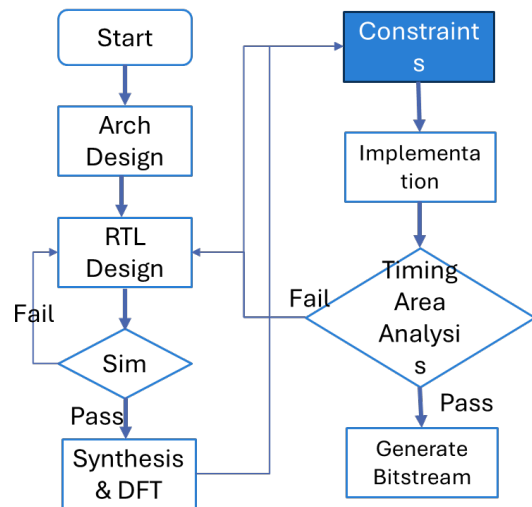
Starting with Vivado Project - Constraints

- ❑ After filling out the table, type “Ctrl+S” to save a constraint file



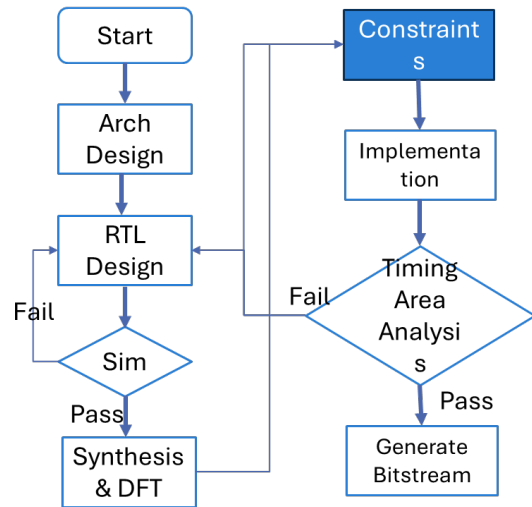
Starting with Vivado Project - Constraints

- ❑ Then you are going to create clock constraints.
- ❑ Click Constraints Wizard to create a clock. In this module, the 7-segment display module requires a 100MHz clock. Since we already have the clock from input IO W5, bonded to “clk” signal, we simply create a clock constraint to let Vivado know we have a clock.



Starting with Vivado Project - Constraints

- ❑ Vivado recognize “clk” signal as a clock. Type 100 in the Frequency cell to create a 100 MHz clock, then click next.



Timing Constraints Wizard

Primary Clocks
Primary clocks usually enter the design through input ports. Specify the period and optionally a name and waveform (rising and falling edge times) to describe the duty cycle if not 50%. [More info](#)

Recommended Constraints

<input checked="" type="checkbox"/>	Object	Name	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)	Jitter (ns)
<input checked="" type="checkbox"/>	clk	clk	100.000	10.000	0.000	5.000	

Constraints for Pulse Width Check Only

<input type="checkbox"/>	Object	Name	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)	Jitter (ns)
--------------------------	--------	------	-----------------	-------------	--------------	--------------	-------------

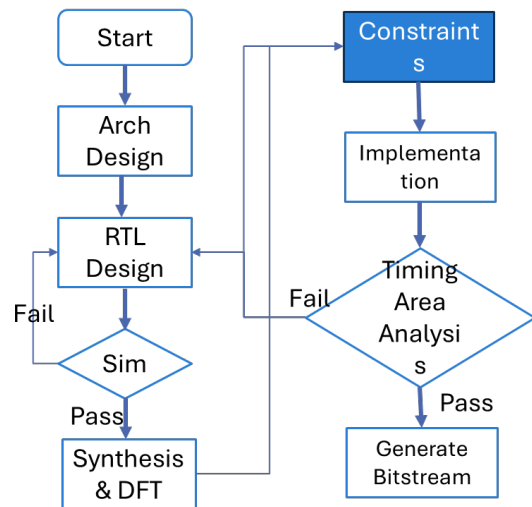
Tcl Command Preview (1) Existing Create Clock Constraints (0)

```
create_clock -period 10.000 -name clk -waveform {0.000 5.000} [get_ports {clk}]
```

Buttons: Back, Next, Skip to Finish >>, Cancel

Starting with Vivado Project - Constraints

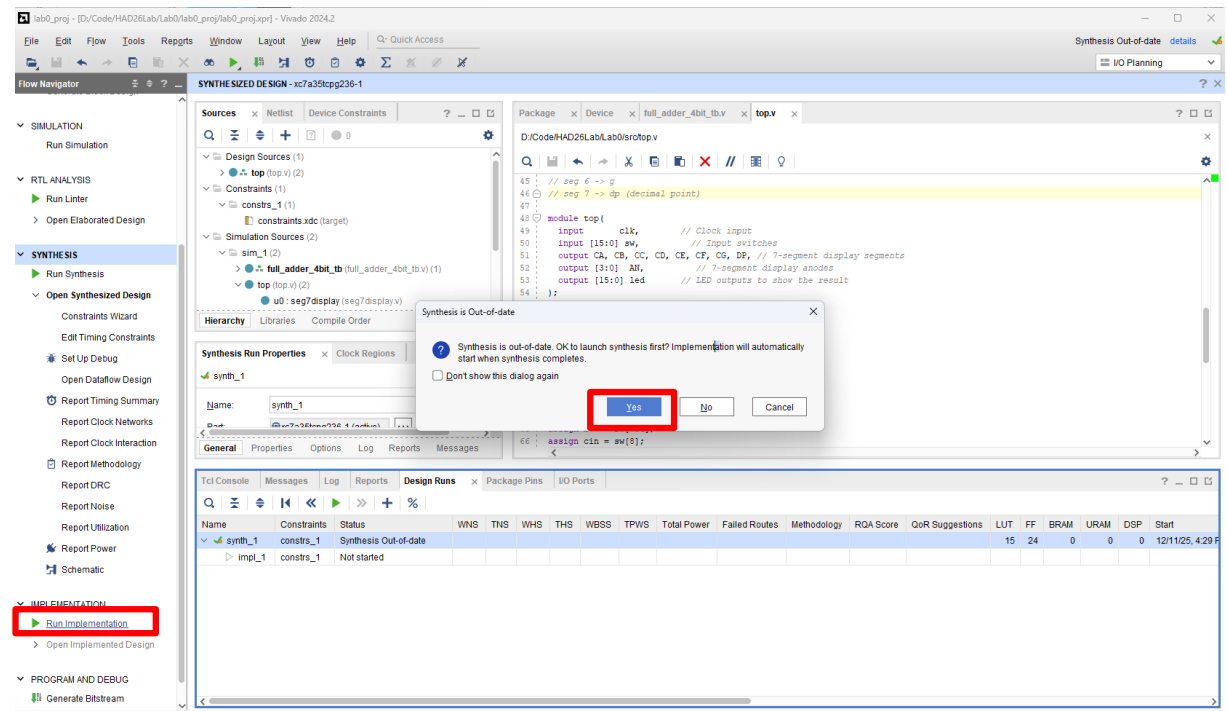
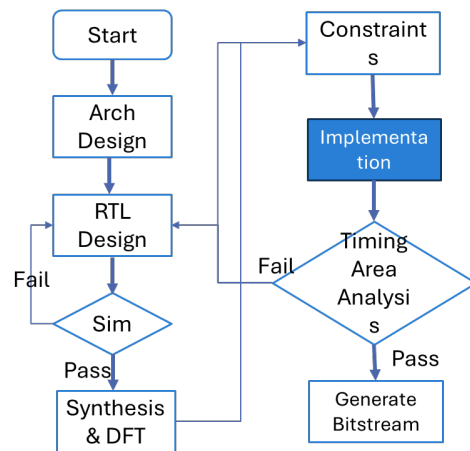
- ❑ Vivado recognize “clk” signal as a clock. Type 100 in the Frequency cell to create a 100 MHz clock, then click “Skip to finish”.



The screenshot shows the 'Timing Constraints Wizard' dialog box. It has a title bar with a close button. Below the title bar, there's a section for 'Primary Clocks' with a descriptive text: 'Primary clocks usually enter the design through input ports. Specify the period and optionally a name and waveform (rising and falling edge times) to describe the duty cycle if not 50%. More info'. Below this is a table of 'Recommended Constraints' with columns: Object, Name, Frequency (MHz), Period (ns), Rise At (ns), Fall At (ns), and Jitter (ns). A single row is visible with 'clk' as the object and name, and '100.000' as the frequency. Below the table is a section for 'Constraints for Pulse Width Check Only' with an empty table. At the bottom, there's a 'Tcl Command Preview' section showing the command: `create_clock -period 10.000 -name clk -waveform {0.000 5.000} [get_ports {clk}]`. At the very bottom, there are four buttons: 'Back', 'Next', 'Skip to Finish >>' (highlighted with a red box), and 'Cancel'.

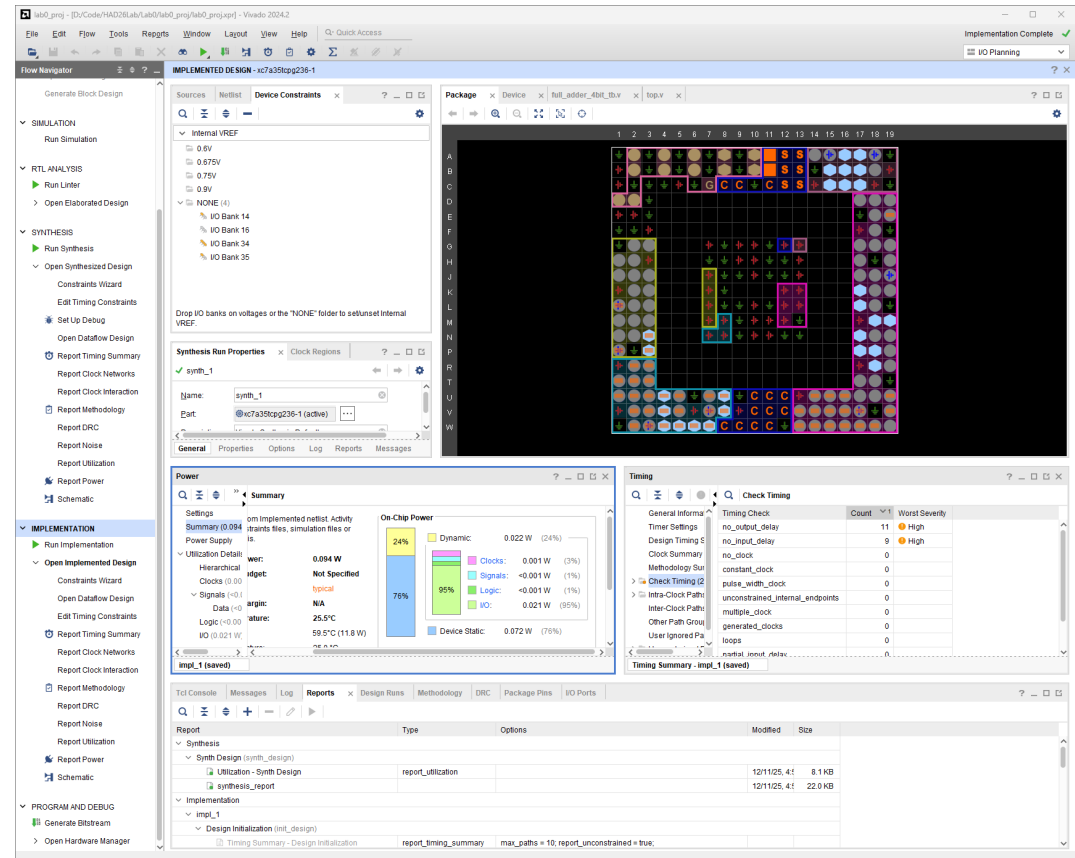
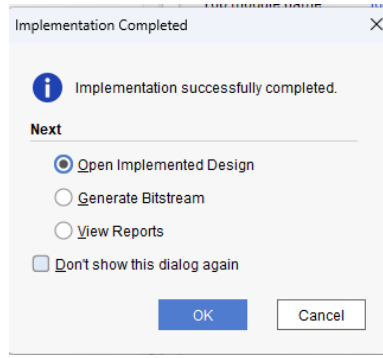
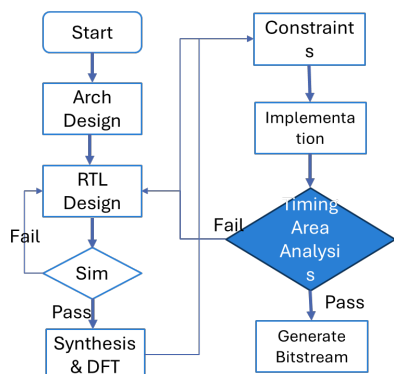
Starting with Vivado Project - Implementation

- ❑ After saving the constraints, click **“Run Implementation”**. Vivado will warn you the Synthesized Design is out-of-date since you add constraints. Click Yes to rerun Synthesize and Implementation together.



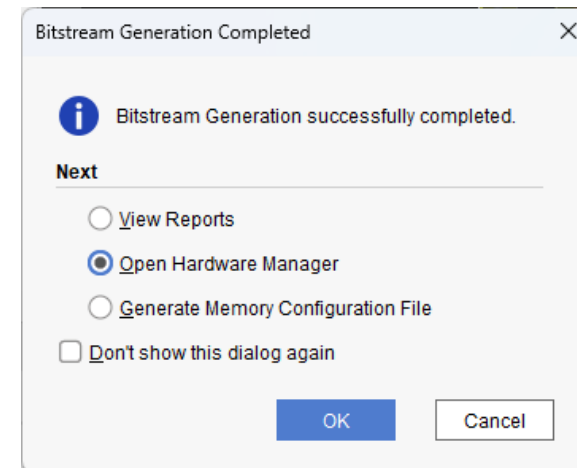
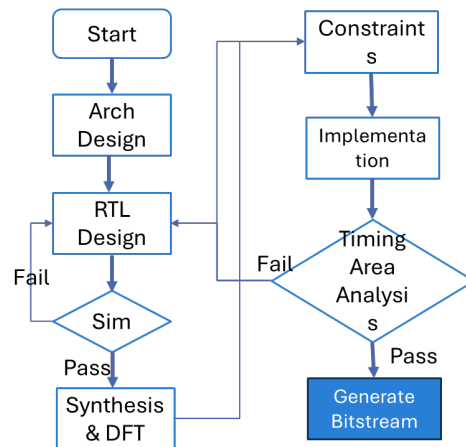
Starting with Vivado Project - Implementation

- ❑ After implementation is finished, you can “Open Implemented Design” to check the Timing, Power, and Area usage.
- ❑ In this simple example, we are unlikely to violate any timing/area/power restrictions.
- ❑ Then click “Generate Bitstream” in the flow navigator.



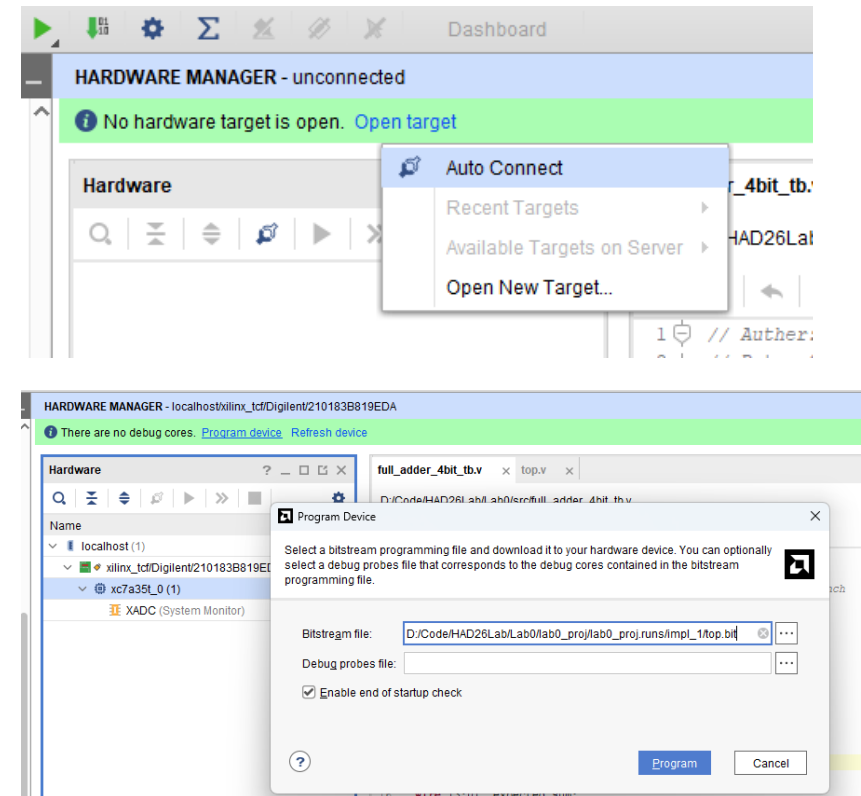
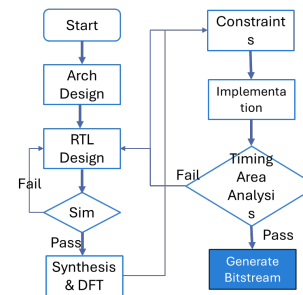
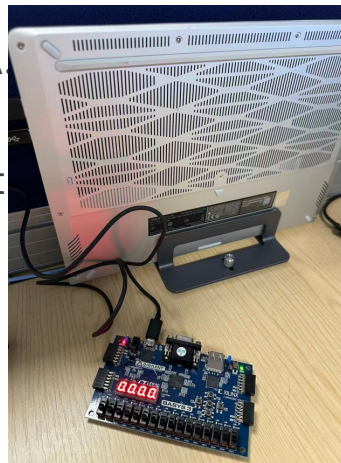
Starting with Vivado Project - Programming

- ❑ After generating the bitstream, you are ready to program on FPGA.
- ❑ Choose “Open Hardware Manager” when finished.



Starting with Vivado Project - Programming

- ❑ After generating the bitstream, you are ready to program on FPGA
- ❑ Choose “Open Hardware Manager” when finished. Connect BASYS3 with your Windows Laptop.
- ❑ In the hardware manager, click *Open Target -> Auto Connect*
- ❑ Then click *Program Device -> Program*



Pre-requisite: Design workstation you need

- **A windows/linux desktop with X86 CPU**

mac with Intel CPU –
OK. You need a
virtual machine for
Windows



mac with m-series
chip – NO! it will
never work (due to
driver issue)



PC with x86 CPU –
Good option



PC with arm CPU –
NO! Not going to
work



<https://www.xilinx.com/support/download.html>

Vivado ML Edition - 2025.2

Downloads

Licensing Help NIC Software & Drivers Alveo Packages

Vivado (HW Developer) Vitis (SW Developer) Vitis Embedded Platforms Embedded Software Power Design Manager Dev Tools

Version

- 2025.2
- 2025.1
- 2024.2
- 2024.1
- Vivado Archive
- ISE Archive
- CAE Vendor Libraries Archive

Vivado™ Edition - 2025.2 Full Product Installation

Important Information

Vivado™ 2025.2 is now available for download:

New production-ready devices supported:

- Versal AI Edge Series Gen 2 and Versal Prime Series Gen 2

Versal QoR Enhancements

Download Includes
Download Type
Last Updated
Answers
Documentation

Note:

- Download verification is only supported with Google Chrome and Microsoft Edge web browsers.
- Vivado ML 2021.1 and later versions require upgrading your license server tools to the Flex 11.17.2.0 versions.

AMD Unified Installer for FPGAs & Adaptive SoCs 2025.2: Windows Self Extracting Web Installer (EXE - 233.33 MB)

MD5 SUM Value : 1ecf89bb9f8f7d124637178c3e3ca396

Download Verification

Digests

Signature

Public Key

AMD Unified Installer for FPGAs & Adaptive SoCs 2025.2: Linux Self Extracting Web Installer (BIN - 346.7 MB)

MD5 SUM Value : abe838aa2e2d3d9b10fea94165e9a303

Download Verification

Digests

Signature

Public Key

AMD Unified Installer for FPGAs & Adaptive SoCs 2025.2 SFD (TAR/GZIP - 95.69 GB)

MD5 SUM Value : 5e793c6b88de5123a09f024253fc2527

Download Verification

Digests

Signature

Public Key

Create AMD account



Sign-In

E-mail Address

 This field cannot be left blank

Password

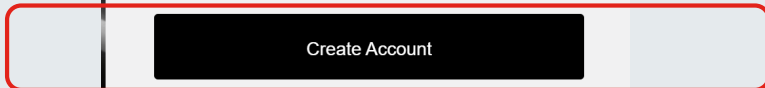
Sign in

OR

Create Account

[Forgot / Reset Password?](#)

First you need to create an account



Download and install

The screenshot shows the AMD Download Center interface. At the top is a navigation bar with the AMD logo and links for Products, Solutions, Resources & Support, and Shop. The main heading is "Download Center - Name and Address Verification". Below this is an information box titled "U.S. Government Export Approval" with two bullet points: "U.S. export regulations require that your First Name, Last Name, Company Name and Shipping Address be verified before AMD can fulfill your download request. **Please provide accurate and complete information.**" and "Addresses with Post Office Boxes and names/addresses with Non-Roman Characters with accents such as grave, tilde or colon **are not supported** by US export compliance systems." Below the information box is a form with three fields: "First Name" (containing "Haiyu"), "Last Name" (containing "Mao"), and "E-mail". A red box highlights these three fields. Below the form, the filename "FPGAs_AdaptiveSoCs_Unified_2024.2_1113_1001_Win64.exe" is displayed, followed by a note: "If you are downloading the Vivado / Vitis unified installer, you will receive a follow-up confirmation email with a notice regarding our Developer Program." At the bottom, there is a "Download" button and a link to the "privacy notice". A red box highlights the "Download" button. Two red callout boxes on the left provide instructions: "Fill your info in this form Use you email address for verification code Use King's College London address" with an arrow pointing to the form, and "Once the form completed, just press 'Download'" with an arrow pointing to the "Download" button.

AMD Products Solutions Resources & Support Shop

Download Center - Name and Address Verification

U.S. Government Export Approval

- U.S. export regulations require that your First Name, Last Name, Company Name and Shipping Address be verified before AMD can fulfill your download request. **Please provide accurate and complete information.**
- Addresses with Post Office Boxes and names/addresses with Non-Roman Characters with accents such as grave, tilde or colon **are not supported** by US export compliance systems.

First Name * Last Name *
Haiyu Mao
E-mail *

Filename:
FPGAs_AdaptiveSoCs_Unified_2024.2_1113_1001_Win64.exe

If you are downloading the Vivado / Vitis unified installer, you will receive a follow-up confirmation email with a notice regarding our Developer Program.

You can read about how we handle your personal data, your personal data rights, and how you can contact us in our [privacy notice](#).

Download

Fill your info in this form
Use you email address for verification code
Use King's College London address

Once the form completed, just press
"Download"

Install

AMD Unified Installer for FPGAs & Adaptive SoCs 2023.1 - Welcome

Welcome

We are glad you have chosen AMD as your platform development partner. This program can install the AMD products including Vitis, Vivado Design Environment, Lab Edition, Bootgen, HW_Server, Power Design Manager, and Documentation Navigator.

Supported operating systems for 2023.1 are:


- Windows 10 Professional and Enterprise versions 20H2, 21H1, 21H2, and 22H2: 64-bit
- Windows 11 Enterprise 21H2, 22H2: 64-bit

Note: Individual products in this installer may support a smaller subset of this list. Please refer to the relevant User Guides for the latest OS information.

Note: This release requires upgrading your license server tools to the Flex 11.17.2 versions. Please confirm with your license admin that the correct version of the license server tools are installed and available, before running the tools.

To reduce installation time, we recommend that you disable any anti-virus software before continuing. Please disable any power saving settings of your machine (automatic sleep mode) when running the installer.

Follow the installation prompt



AMD

Copyright © 1986-2022 Xilinx, Inc. All rights reserved.
Copyright © 2022-2023 Advanced Micro Devices, Inc. All rights reserved.

Preferences < Back **Next >** Cancel

Install

The "User Authentication" Section should be filled with your AMD account info.

Install vivado

AMD Unified Installer for FPGAs & Adaptive SoCs 2023.1 - Select Product to Install

Select Product to Install

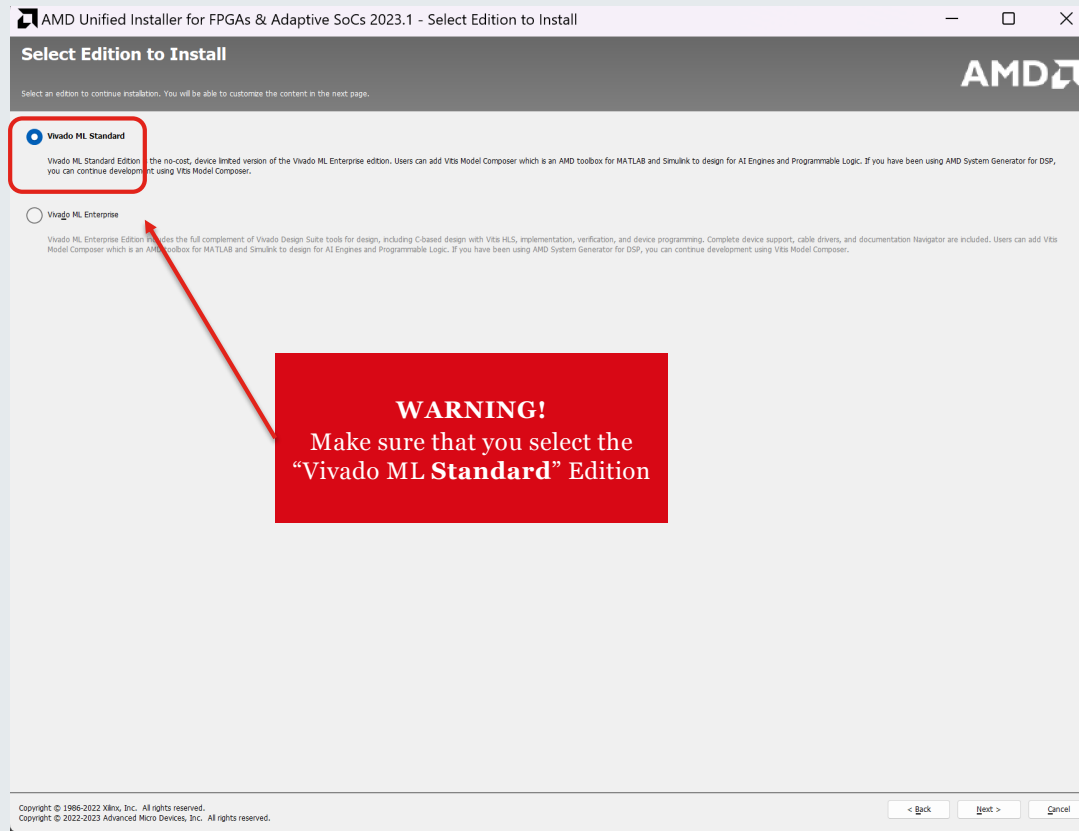
Select a product to continue installation. You will be able to customize the content in the next page.

- Vitis
Installs Vitis Core Development Kit for embedded software and application acceleration development on AMD platforms. Vitis installation includes Vivado Design Suite. Users can also install Vitis Model Composer to design for AI Engines and Programmable Logic in MATLAB and Simulink.
- Vivado**
Includes the full complement of Vivado Design Suite tools for design, including C-based design with Vitis High-Level Synthesis, implementation, verification and device programming. Complete device support, cable driver, and Document Navigator included. Users can also install Vitis Model Composer to design for AI Engines and Programmable Logic in MATLAB and Simulink.
- BootGen
Installs Bootgen for creating bootable images targeting AMD SoCs and FPGAs.
- Lab Edition
Installs only the Vivado Lab Edition. This standalone product includes Vivado Design Programmer, Vivado Logic Analyzer and UpdateMEM tools.
- Hardware Server
Installs hardware server and JTAG cable drivers for remote debugging.
- Power Design Manager (PDM)
Installs only the Power Design Manager (PDM). Power Design Manager is a standalone design tool used to estimate power requirements of Versal and Kria products. It supports the Xilinx Power Estimator (XPE) file exchange format for importing data from Vivado and XPE.
- Documentation Navigator (Standalone)
Documentation Navigator (DocNav) provides access to AMD FPGAs & Adaptive SoCs technical documentation both on the Web and on the Desktop. This is a standalone installation without Vivado Design Suite.

Copyright © 1986-2022 Xilinx, Inc. All rights reserved.
Copyright © 2022-2023 Advanced Micro Devices, Inc. All rights reserved.

< Back **Next >** Cancel

Install vivado



Install vivado

Select these options
Then go to the next prompt window

The screenshot shows the 'AMD Unified FPGAs & Adaptive SoCs Installer 2025.2 - Vivado ML Standard' window. The title bar includes the AMD logo. Below the title bar, the text reads: 'Customize your installation by (de)selecting items in the tree below. Moving cursor over selections below provide additional information.' The main area is a tree view titled 'Filter Devices'. Under 'Design Tools', 'Vivado Design Suite' is expanded, showing 'Vivado' (checked), 'Vitis HLS' (checked), 'Vitis Networking P4' (unchecked), 'Vitis Model Composer(A toolbox for Simulink)' (checked), 'Vitis Embedded Development' (unchecked), 'Power Design Manager (PDM)' (unchecked), and 'DocNav' (checked). Under 'Devices', '7 Series' is expanded, showing 'Virtex-7 FPGAs' (unchecked), 'Kintex-7 FPGAs' (checked), 'Artix-7 FPGAs' (checked), and 'Spartan-7 FPGAs' (checked). Other categories like 'SoCs' and 'Installation Options' are also visible. A 'Selected Devices' panel on the right lists 'Kintex-7 FPGAs', 'Artix-7 FPGAs', and 'Spartan-7 FPGAs'. At the bottom, it shows 'Download Size: 17.42 GB' and 'Disk Space Required: 63.46 GB'. A 'Reset to Defaults' button is present. The footer contains copyright information and navigation buttons: '< Back', 'Next >', and 'Cancel'.

Install vivado

Make sure you tick "Agree" to the License Agreements

AMD Unified Installer for FPGAs & Adaptive SoCs 2023.1 - Accept License Agreements

Accept License Agreements

Please read the following terms and conditions and indicate that you agree by checking the I Agree checkboxes.

End User License Agreement for Vivado

By checking "I Agree" below, or OTHERWISE ACCESSING, DOWNLOADING, INSTALLING or USING THE SOFTWARE, I AGREE on behalf of licensee to be bound by the agreement, which can be viewed by [clicking here](#).

I Agree

End User License Agreement for Doctav

By checking "I Agree" below, or OTHERWISE ACCESSING, DOWNLOADING, INSTALLING or USING THE SOFTWARE, I AGREE on behalf of licensee to be bound by the agreement, which can be viewed by [clicking here](#).

I Agree

End User License Agreement for Model Composer

By checking "I Agree" below, or OTHERWISE ACCESSING, DOWNLOADING, INSTALLING or USING THE SOFTWARE, I AGREE on behalf of licensee to be bound by the agreement, which can be viewed by [clicking here](#).

I Agree

Third Party Software End User License Agreement for Doctav

By checking "I AGREE" below, or OTHERWISE ACCESSING, DOWNLOADING, INSTALLING or USING THE SOFTWARE, YOU AGREE on behalf of licensee to be bound by the agreement, which can be viewed by [clicking here](#).

I Agree

Third Party Software End User License Agreement for Vivado

By checking "I AGREE" below, or OTHERWISE ACCESSING, DOWNLOADING, INSTALLING or USING THE SOFTWARE, YOU AGREE on behalf of licensee to be bound by the agreement, which can be viewed by [clicking here](#).

I Agree

Third Party Software End User License Agreement for Model Composer

By checking "I AGREE" below, or OTHERWISE ACCESSING, DOWNLOADING, INSTALLING or USING THE SOFTWARE, YOU AGREE on behalf of licensee to be bound by the agreement, which can be viewed by [clicking here](#).

Copyright © 1996-2022 Xilinx, Inc. All rights reserved.
Copyright © 2022-2023 Advanced Micro Devices, Inc. All rights reserved.

< Back Next > Cancel

Install vivado

Create a directory, e.g.,:
C:\Xilinx
then click "Yes"

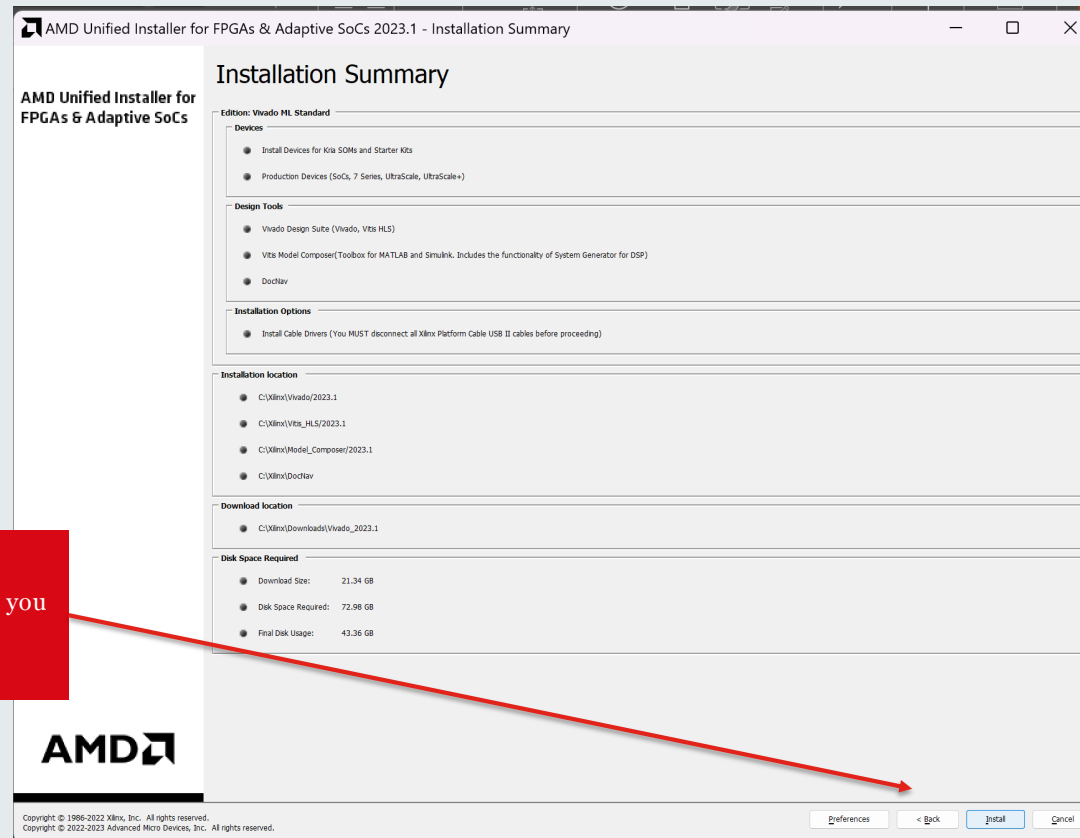
The screenshot shows the 'AMD Unified Installer for FPGAs & Adaptive SoCs 2023.1 - Select Destination Directory' window. The window title bar includes the AMD logo and window controls. The main content area is titled 'Select Destination Directory' and contains the following sections:

- Installation Options:**
 - Select the installation directory: C:\Xilinx
 - Installation location(s): C:\Xilinx\Vivado\2023.1, C:\Xilinx\Vitis_HLS\2023.1, C:\Xilinx\Model Composer\2023.1, C:\Xilinx\genrtav
 - Download location: C:\Xilinx\Downloads\Vivado_2023.1
 - Disk Space Required:
 - Download Size: 21.34 GB
 - Disk Space Required: 72.98 GB
 - Final Disk Usage: 43.36 GB
 - Disk Space Available: 266.84 GB
- Select shortcut and file association options:**
 - Create program group entries (Xilinx Design Tools)
 - Create desktop shortcuts
 - Create file associations
 - Apply shortcut & file association selections to:
 - Current user
 - All users

A dialog box titled 'The specified directory C:\Xilinx doe...' is overlaid on the main window. It contains a question mark icon and the text 'C:\Xilinx does not exist, do you want to create it?'. There are 'Yes' and 'No' buttons at the bottom of the dialog. A red arrow points from the text in the red box to the 'Yes' button in the dialog.

At the bottom of the main window, there are buttons for '< Back', 'Next >', and 'Cancel'. Copyright information is visible at the bottom left: 'Copyright © 1986-2022 Xilinx, Inc. All rights reserved. Copyright © 2022-2023 Advanced Micro Devices, Inc. All rights reserved.'

Install vivado



Once the directory is setup, you can press "install"

Install vivado

It might take several minutes to complete the installation

AMD Unified Installer for FPGAs & Adaptive SoCs 2023.1 - Installation Progress

Installation Progress

- ✓ It took 1 h and 27 m(s) to download files.
- ✓ It took 1 h and 54 m(s) to install files.
- ✓ Done Final Processing.

AMD FPGAs & Adaptive SoCs Software... Installation completed successfully. OK

AMD Vitis™
Unlocking a new design experience for all developers

< Back Next > Cancel

Copyright © 1986-2022 Xilinx, Inc. All rights reserved.
Copyright © 2022-2023 Advanced Micro Devices, Inc. All rights reserved.

Getting started with Vivado and Basys3

https://www.youtube.com/watch?v=6_GxkslqbcU

Introductory videos to vivado and basys3
can be found on youtube