# USING VIO AND ILA IN VIVADO

Stefan Johannesson

# OBJECTIVES OF LAB

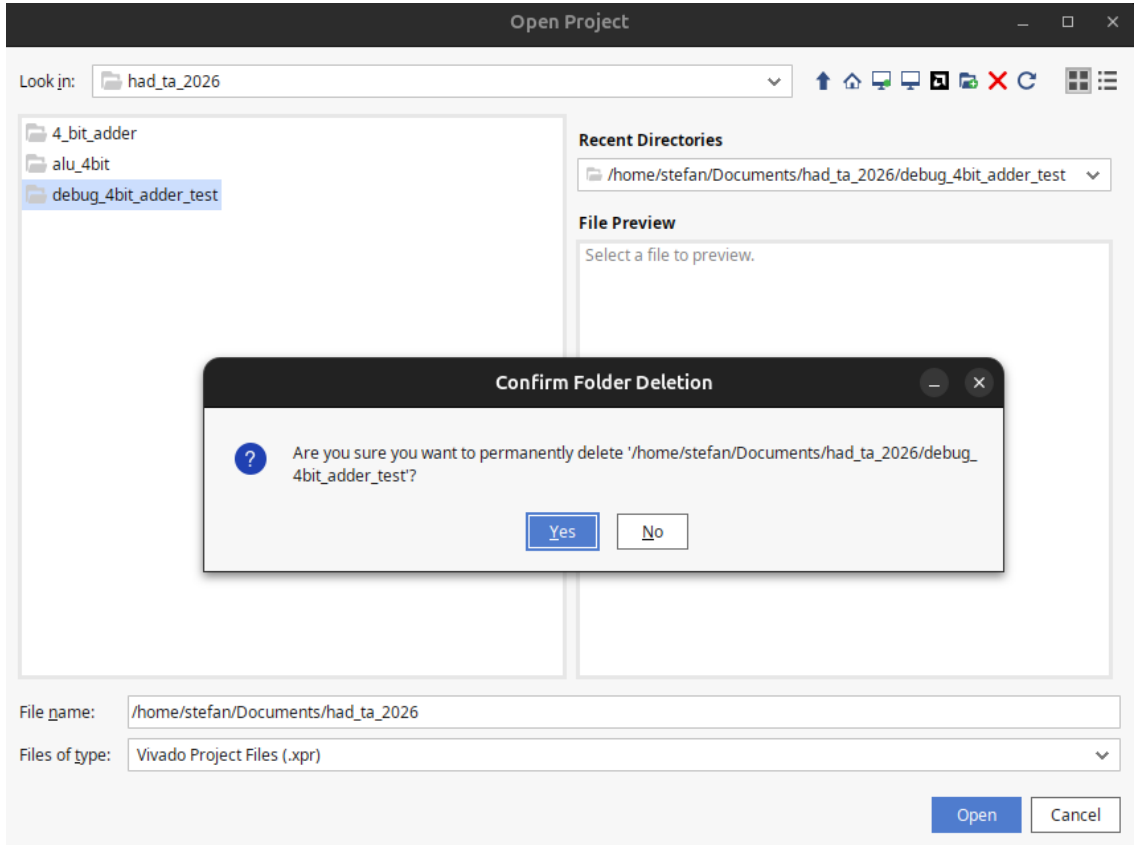
- Go over common headaches with Vivado
- Learn about VIO and ILA in Vivado
- Modify 4-bit adder project to leverage VIOs and ILAs
- Learn about ILA triggers

# HEADACHE 1: NAMING CONVENTION

- Vivado has a very strict naming scheme. Only allowed letters:
  - Alphanumeric characters (A-Z, a-z, 0-9)
  - Underscores (_)
- Even folder names is NOT exempt from naming scheme
  - Saving under "OneDrive – YOUR_NAME\...\..." will cause same issue
  - Save locally, like "C:\Documents", or "C:\Desktop"

# HEADACHE 2: PHANTOM PROJECTS

- When deleting project folders using a file explorer, **Vivado will still see the 'deleted' project**
- How to solve?
  - Manually delete the projects **within Vivado.**
  - Under File -> Project -> Open, you should have a pop-up window
  - Within this window, delete the project folder you want removed.

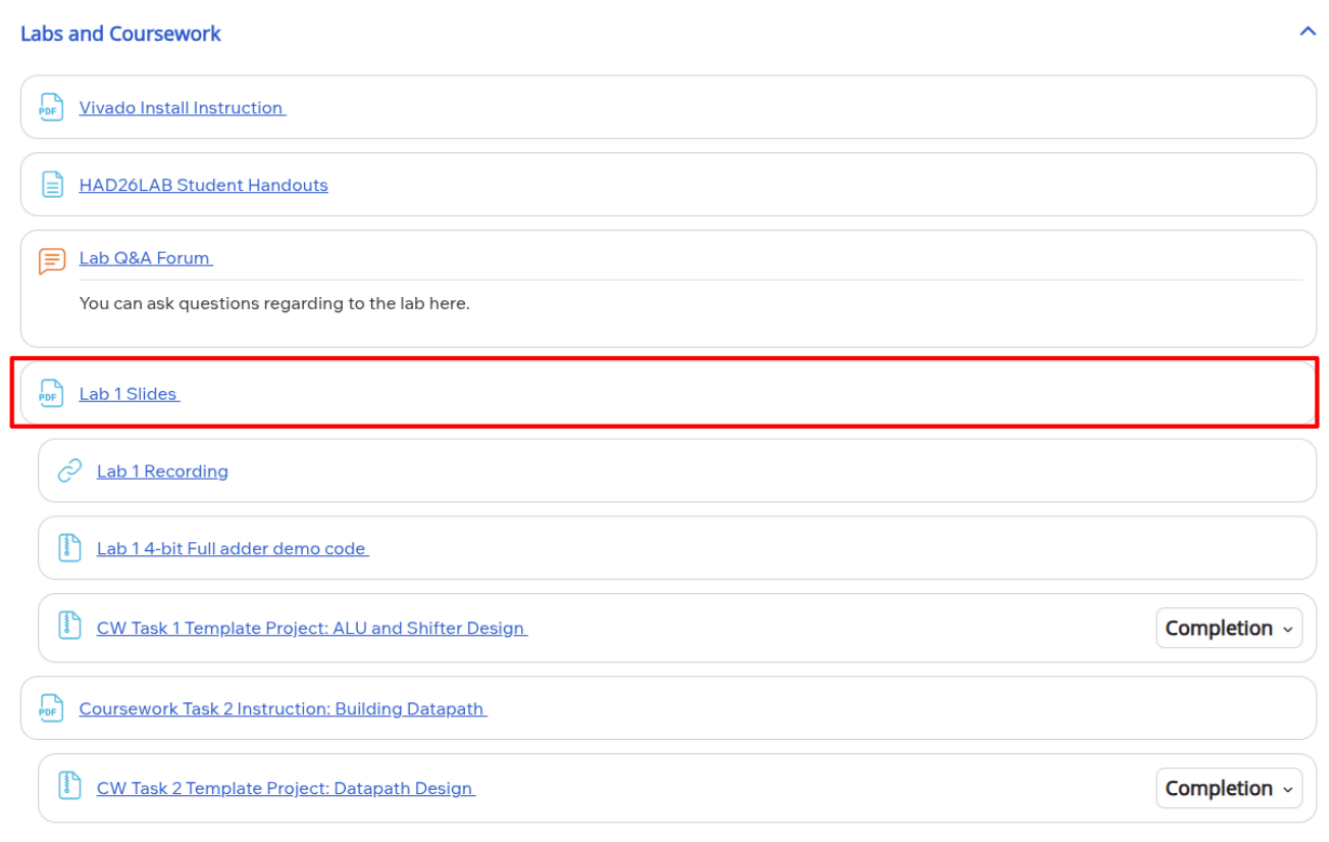


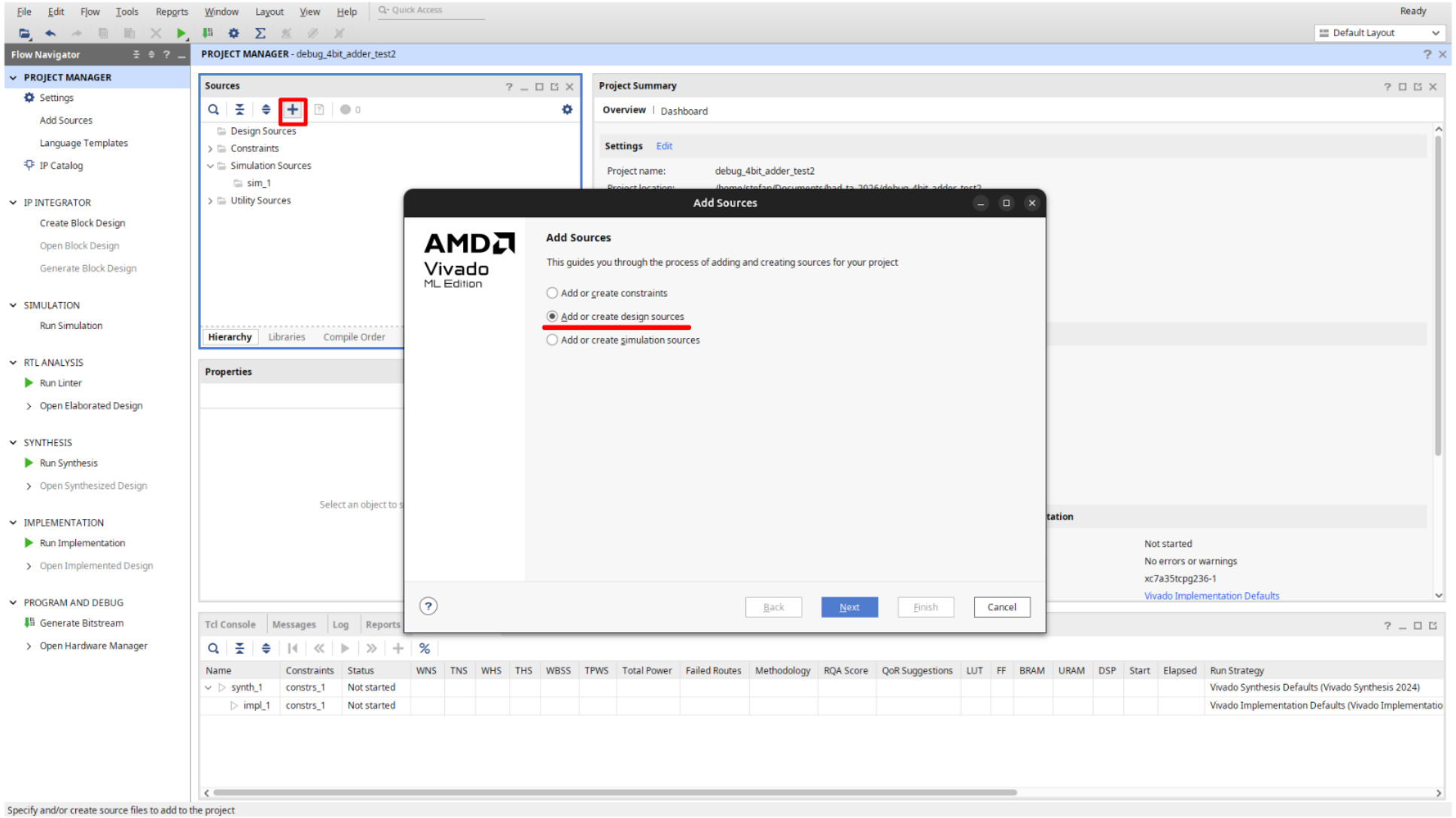
# WHAT IS VIO AND ILA?

- **VIO -** Virtual Input/Output
  - Virtual control panel of FPGA on your computer
  - Virtually drive signals inside FPGA via USB
- **ILA -** Integrated Logic Analyzer
  - Digital Oscilloscope built directly into FPGA logic
  - Records internal signals at full system speed (100 MHz for Basys 3)

# PRELIMINARY: RE-IMPORT 4-BIT ADDER

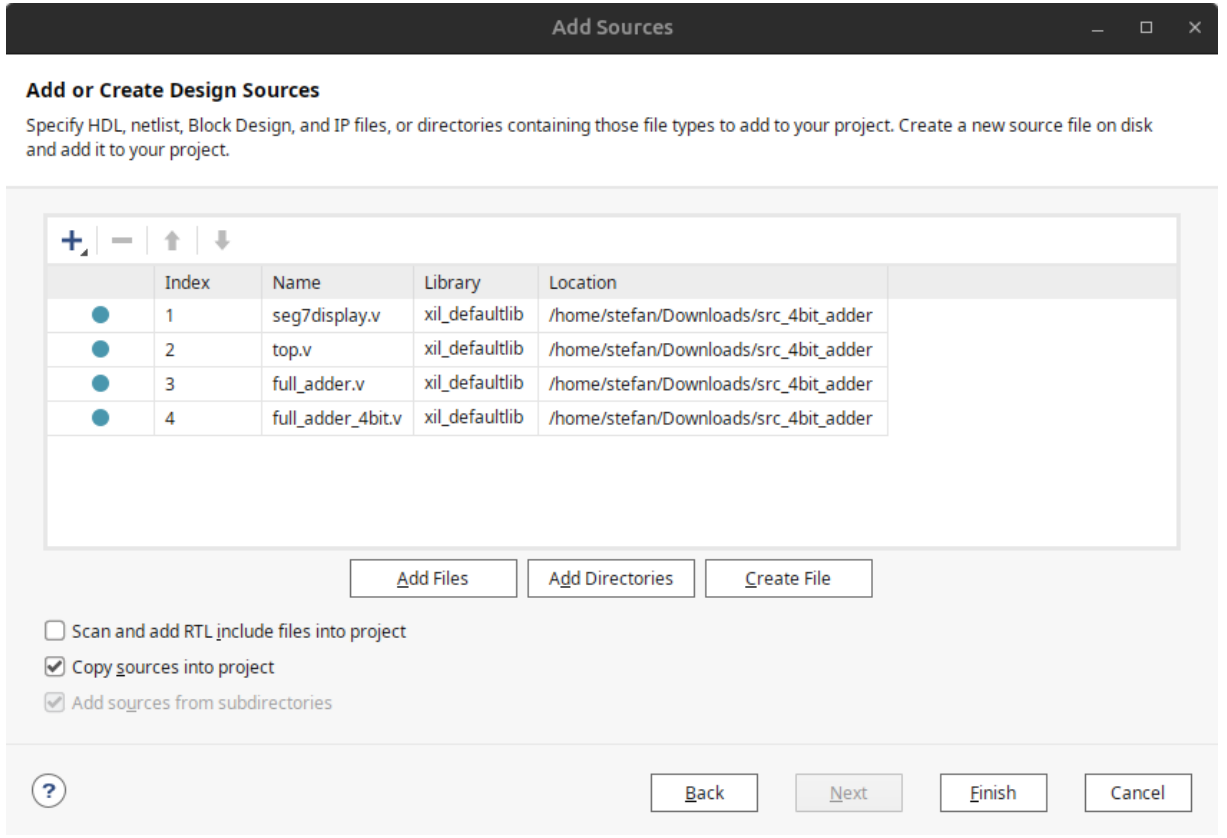
- Make a new Vivado project called "debug_4bit_adder"
- Import the 4 bit adder files from KEATS
- Extract the ZIP file to access individual Verilog files



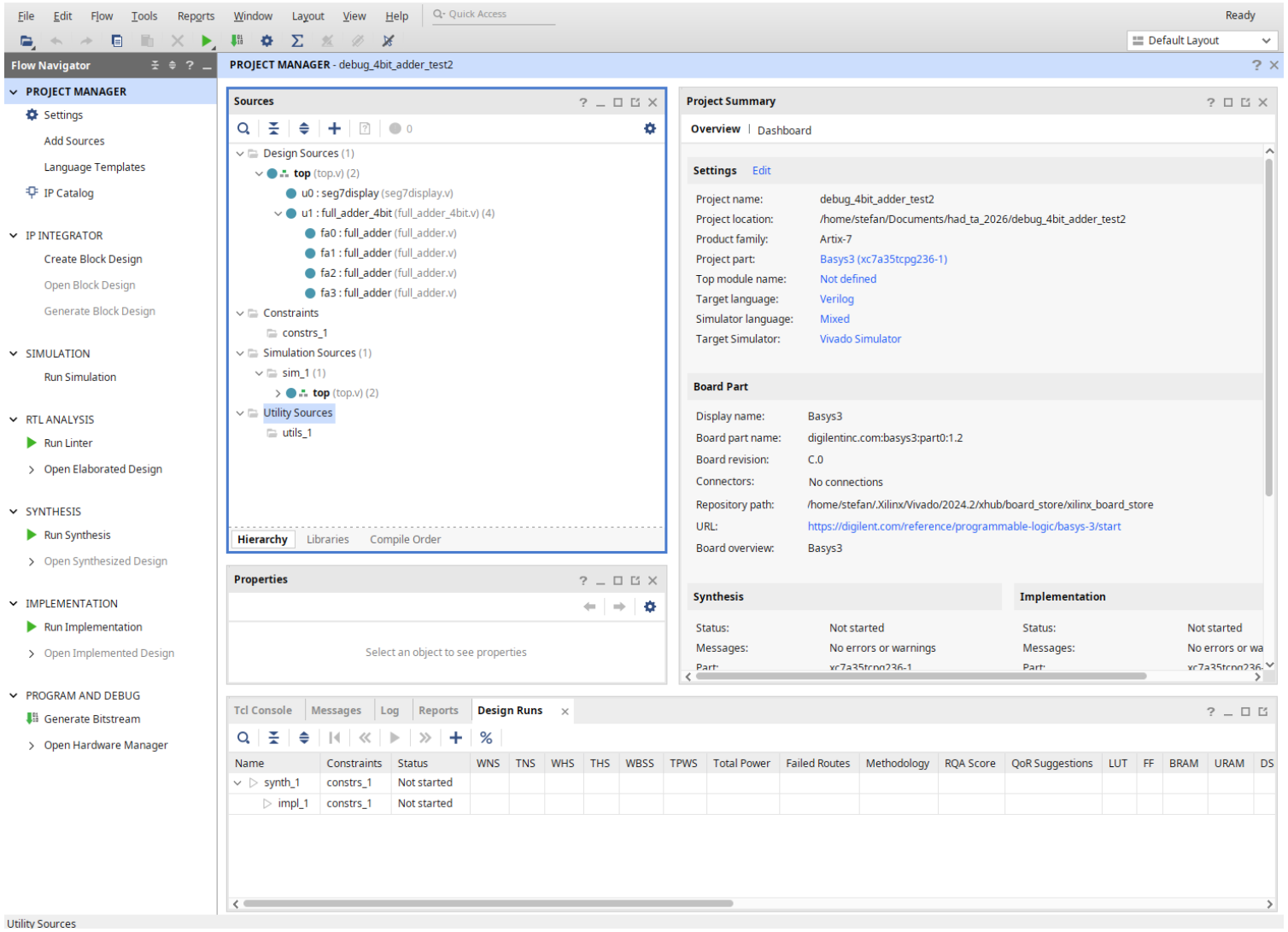

File Edit Flow Tools Reports Window Layout View Help
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Run Linter
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
PROJECT MANAGER - debug_4bit_adder_test2
Sources
Design Sources
Constraints
Simulation Sources
sim_1
Utility Sources
Hierarchy Libraries Compile Order
Properties
Project Summary
Overview | Dashboard
Settings Edit
Project name: debug_4bit_adder_test2
Add Sources
AMD
Vivado
ML Edition
Add Sources
This guides you through the process of adding and creating sources for your project
Add or create constraints
Add or create design sources
Add or create simulation sources
Back Next Finish Cancel
Not started
No errors or warnings
xc7a35tcpg236-1
Vivado Implementation Defaults
Tcl Console Messages Log Reports
Name Constraints Status WNS TNS WHS THS WBSS TPWS Total Power Failed Routes Methodology RQA Score QoR Suggestions LUT FF BRAM URAM DSP Start Elapsed Run Strategy
synth_1 constrs_1 Not started Vivado Synthesis Defaults (Vivado Synthesis 2024)
impl_1 constrs_1 Not started Vivado Implementation Defaults (Vivado Implementatio
Specify and/or create source files to add to the project

# RE-IMPORT 4-BIT ADDER

- Import your Verilog files from the extracted ZIP file
  - You can omit importing the testbench
- Make sure that "Copy sources into project" is selected (makes file management easier for your coursework!)
- Useful tip: if you have done Lab 1 as a separate project, can directly import constraints file into current project (will show how!)



- Project Hierarchy should now look like this



- Possible to repeat steps in Lab 1 again to define pinouts and voltages…
- … but if you already did Lab 1, you can simply use the same constraints file!

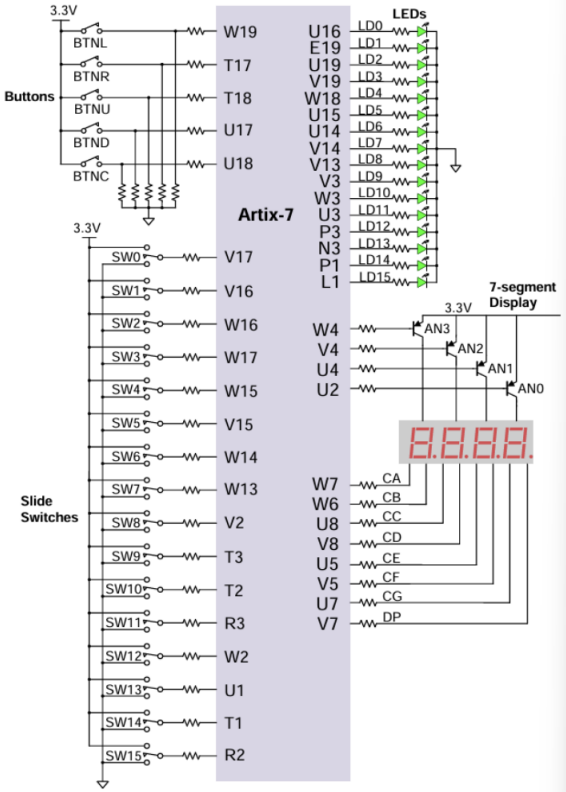
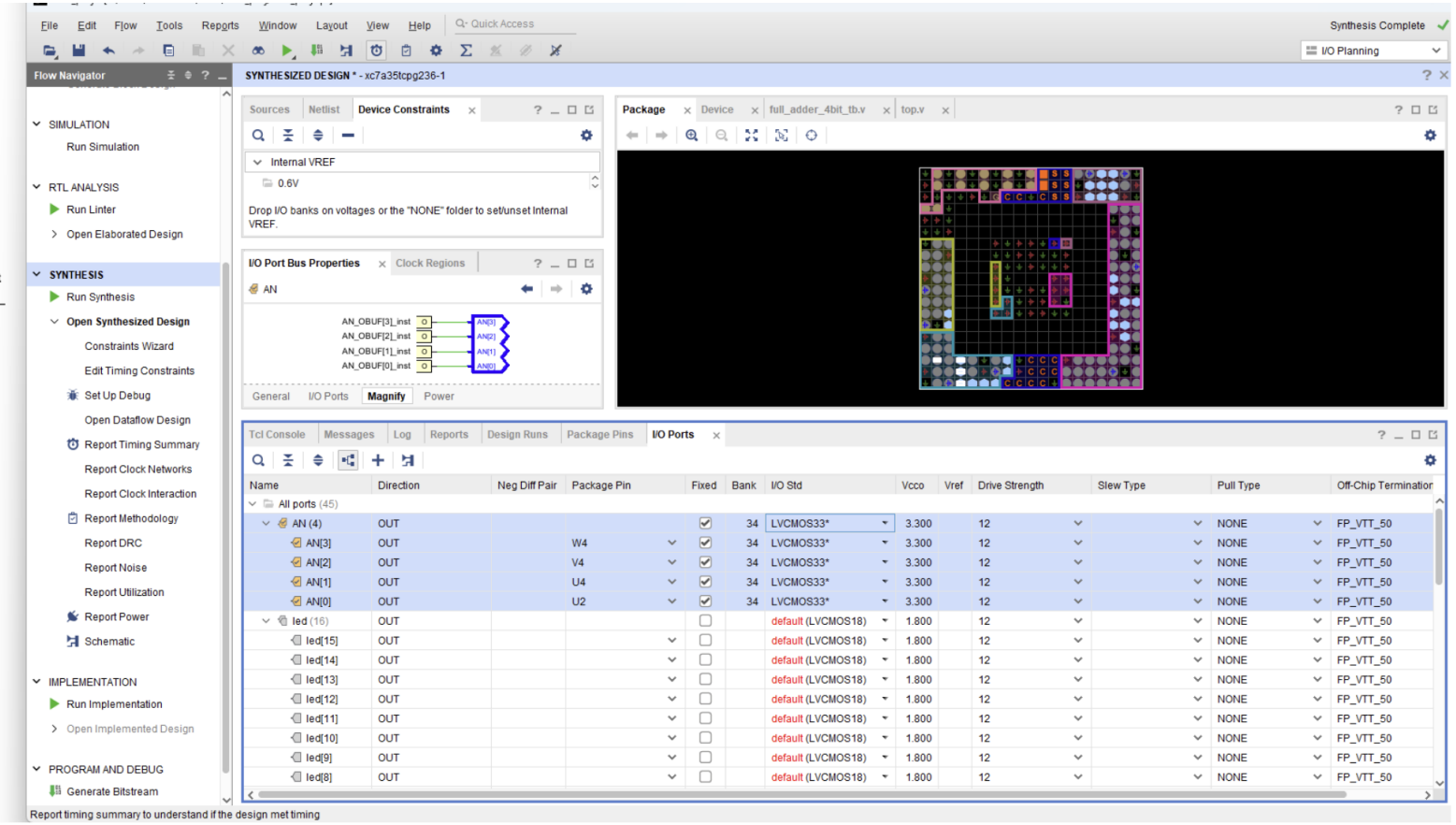
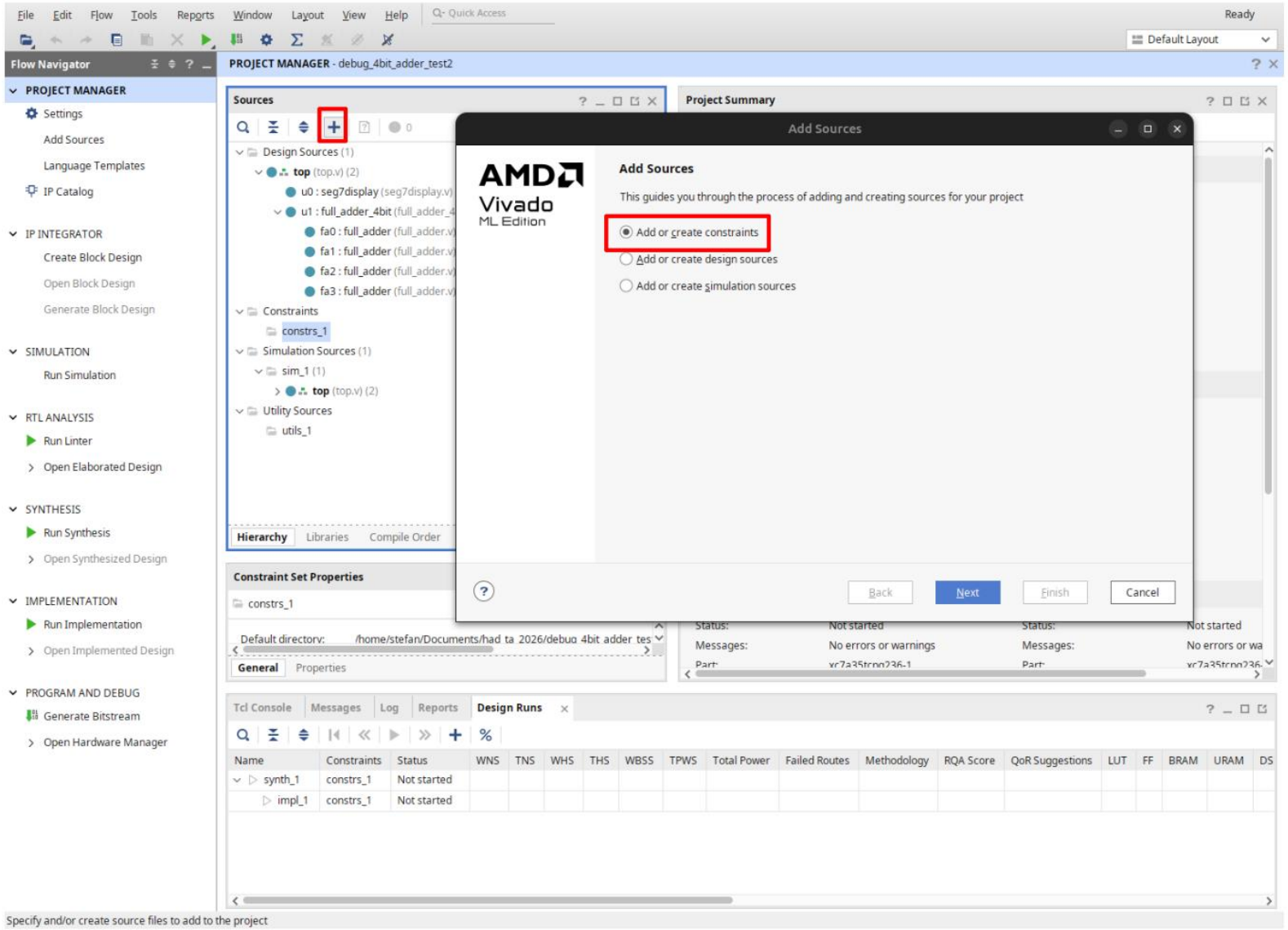
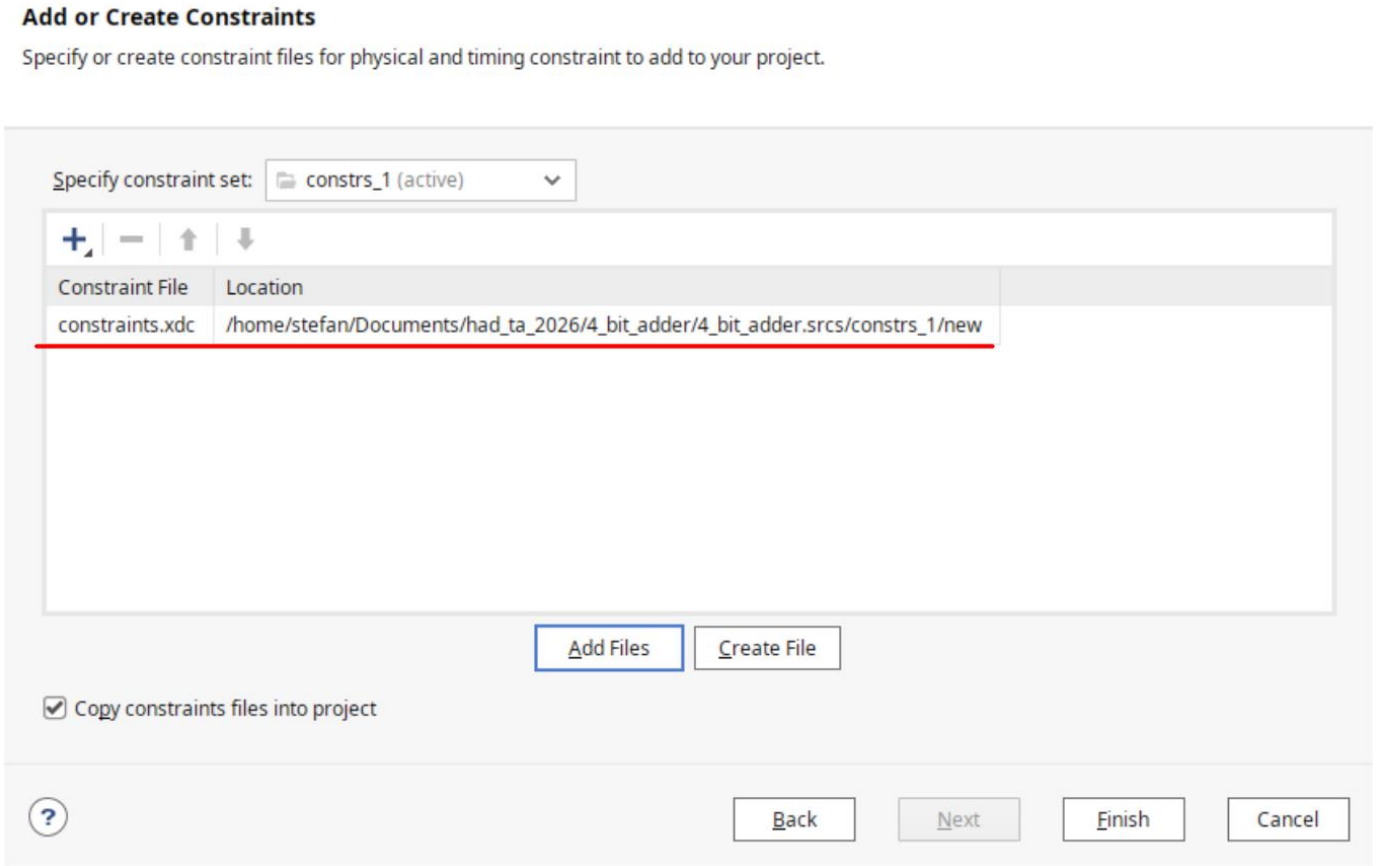



*Figure 16. General purpose I/O devices on the Basys3.*

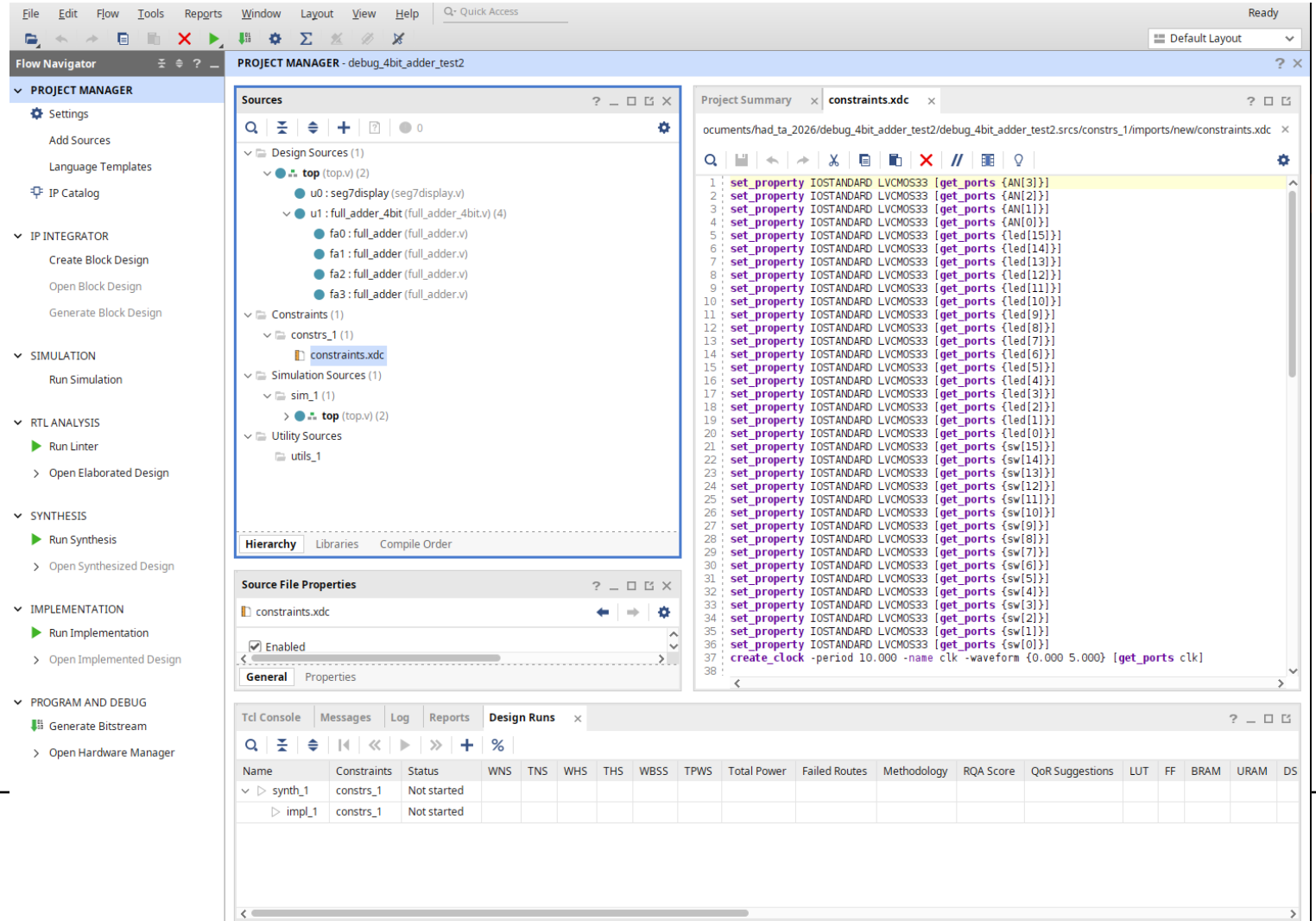



File Edit Flow Tools Reports Window Layout View Help
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Run Linter
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
PROJECT MANAGER - debug_4bit_adder_test2
Sources
Design Sources (1)
top (top.v) (2)
u0 : seg7display (seg7display.v)
u1 : full_adder_4bit (full_adder_4
fa0 : full_adder (full_adder.v)
fa1 : full_adder (full_adder.v)
fa2 : full_adder (full_adder.v)
fa3 : full_adder (full_adder.v)
Constraints
constrs_1
Simulation Sources (1)
sim_1 (1)
top (top.v) (2)
Utility Sources
utils_1
Hierarchy Libraries Compile Order
Project Summary
Add Sources
AMD
Vivado
ML Edition
Add Sources
This guides you through the process of adding and creating sources for your project
Add or create constraints
Add or create design sources
Add or create simulation sources
Back Next Finish Cancel
Constraint Set Properties
constrs_1
Default directory: /home/stefan/Documents/had ta 2026/debug 4bit adder tes
General Properties
Status: Not started
Messages: No errors or warnings
Part:
Tcl Console Messages Log Reports Design Runs
Name Constraints Status WNS TNS WHS THS WBSS TPWS Total Power Failed Routes Methodology RQA Score QoR Suggestions LUT FF BRAM URAM DS
synth_1 constrs_1 Not started
impl_1 constrs_1 Not started
Specify and/or create source files to add to the project

- Go to where you saved Lab 1
- Your constraints file should be saved under "{PROJ_NAME}.srcs\constrs1\new\"
- Import it into you project



# WHAT YOU SHOULD HAVE



# STEP 1: SETTING UP THE ILA

- We want to explicitly tell the ILA what specific wires in our design we want it to watch
- We can do this by adding the `(* mark_debug = "true" *)` attribute right before the wire declaration:
- In "top.v", replace:

```
//Wires and signals
wire [3:0] a;
wire [3:0] b;
wire       cin;
wire [3:0] sum;
wire       cout;
```

# STEP 1

```
// Wires and signals, add mark_debug so ILA can "see" them
(* mark_debug = "true" *) wire [3:0] a;
(* mark_debug = "true" *) wire [3:0] b;
(* mark_debug = "true" *) wire       cin;
(* mark_debug = "true" *) wire [3:0] sum;
(* mark_debug = "true" *) wire       cout;
```

# STEP 2: SETTING UP  THE VIO

- We want to setup additional, virtual inputs, which can control the FPGA
- However, we may occasionally want to make use of the switches

- We can multiplex the swiches and the virtual I/O, and have an additional, virtual signal vio_sel, to control which is the input
  - vio_sel = 0 => using physical switches
  - vio_sel = 1 => using VIO signals

# REPLACE INPUTS WITH MULTIPLEXED VERSION

```
// Assign inputs from switches
assign a   = sw[3:0];
assign b   = sw[7:4];
assign cin = sw[8];
```

```
// Added VIO signals for debugging
wire [3:0] vio_a;
wire [3:0] vio_b;
wire vio_cin;
wire vio_sel; // VIO control switch (0 =
Physical, 1 = Virtual)

// Added Multiplexer to allow swapping
between hardware and software inputs
// If vio_sel is 1, use VIO. If 0, use
physical switches.
assign a   = vio_sel ? vio_a   : sw[3:0];
assign b   = vio_sel ? vio_b   : sw[7:4];
assign cin = vio_sel ? vio_cin : sw[8];
```

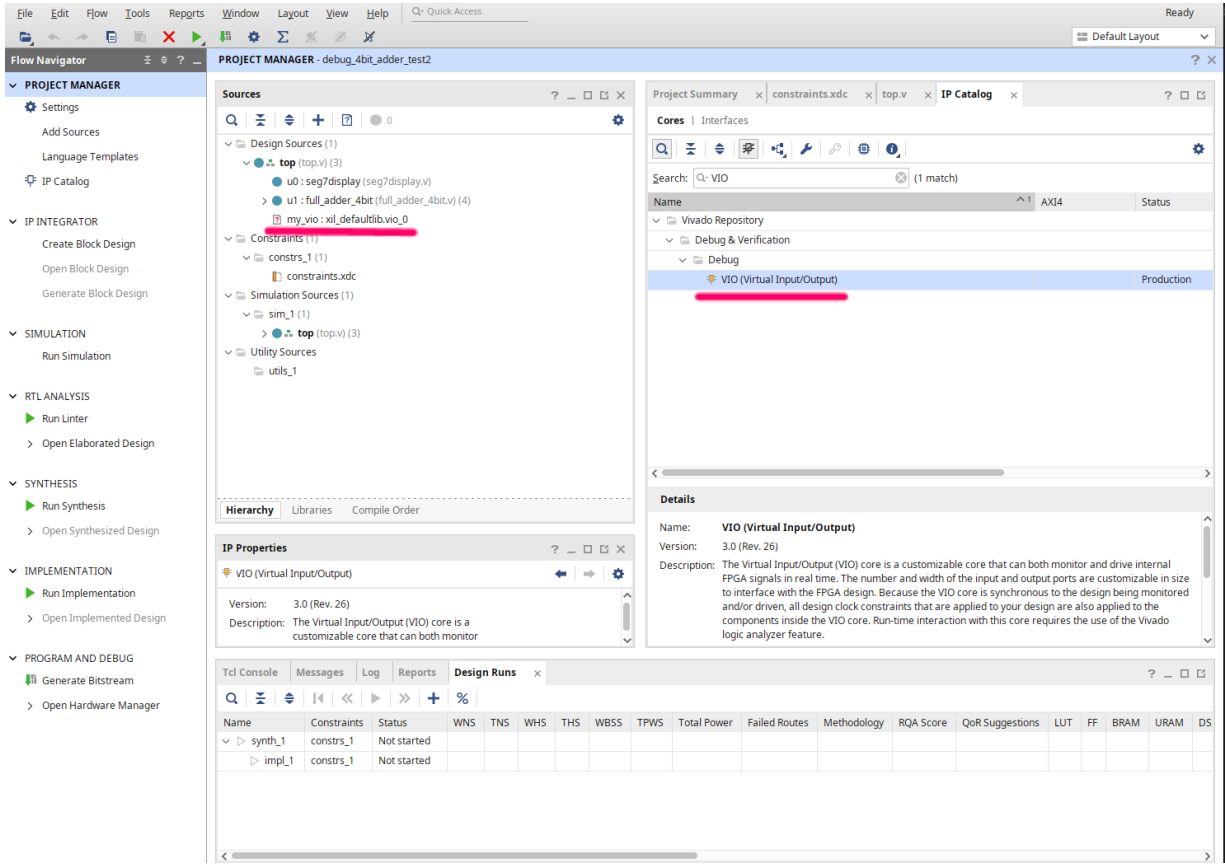
# ADD VIO CORE

- At the bottom of "top.v", add the following code to instantiate the VIO core

```
// --- NEW: Instantiate VIO Core ---
vio_0 my_vio (
  .clk(clk),                // input wire clk
  .probe_in0(sum),          // input wire [3 : 0] probe_in0
  .probe_in1(cout),         // input wire [0 : 0] probe_in1
  .probe_out0(vio_a),       // output wire [3 : 0] probe_out0
  .probe_out1(vio_b),       // output wire [3 : 0] probe_out1
  .probe_out2(vio_cin),     // output wire [0 : 0] probe_out2
  .probe_out3(vio_sel)      // output wire [0 : 0] probe_out3
);
```

# ADDING THE VIO CORE

1) Open the IP catalogue on the left menu
2) Search "VIO (Virtual Input/Output)" and select it
3) In the configuration window:
   - `PROBE_IN` Ports: Set to 2 (we are reading the outputs sum and cout)
   - `PROBE_OUT` Ports: Set to 4 (we are controlling `vio_a`, `vio_b`, `vio_cin`, and `vio_sel`)



# ADDING THE VIO CORE

4) Go to the `PROBE_IN` Ports tab

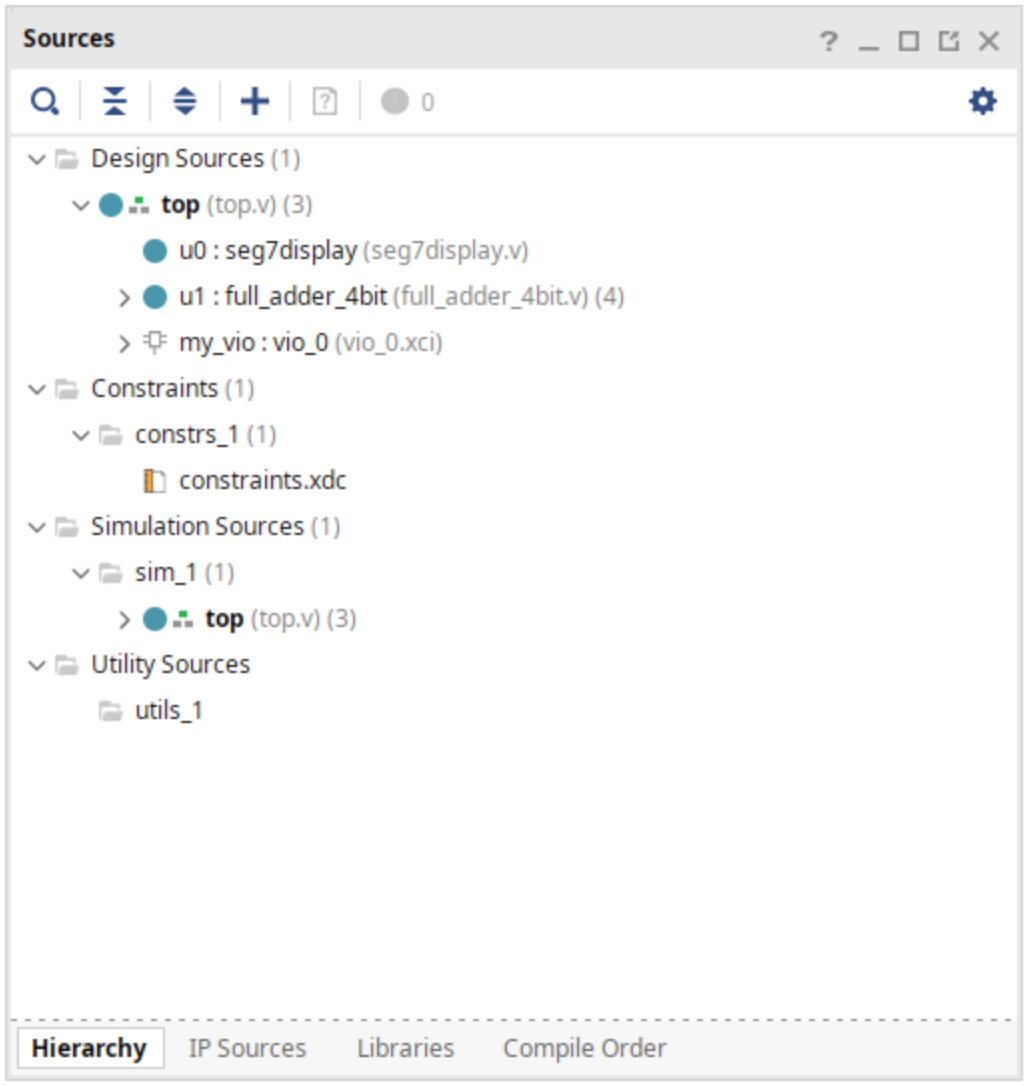
- `PROBE_IN0` width = 4 (for 4-bit output of sum)
- `PROBE_IN1` width = 1 (for cout)

5) Go to the `PROBE_OUT` Ports tab

- `PROBE_OUT0` width = 4 (for 4-bit input vio_a)
- `PROBE_OUT1` width = 4 (for 4-bit input vio_B)
- `PROBE_OUT2`width = 1 (for 1-bit input vio_cin)
- `PROBE_OUT3` width = 1 (for 1-bit input vio_sel)

6) Under Synthesis Options, click Global.

7) Click OK -> Generate

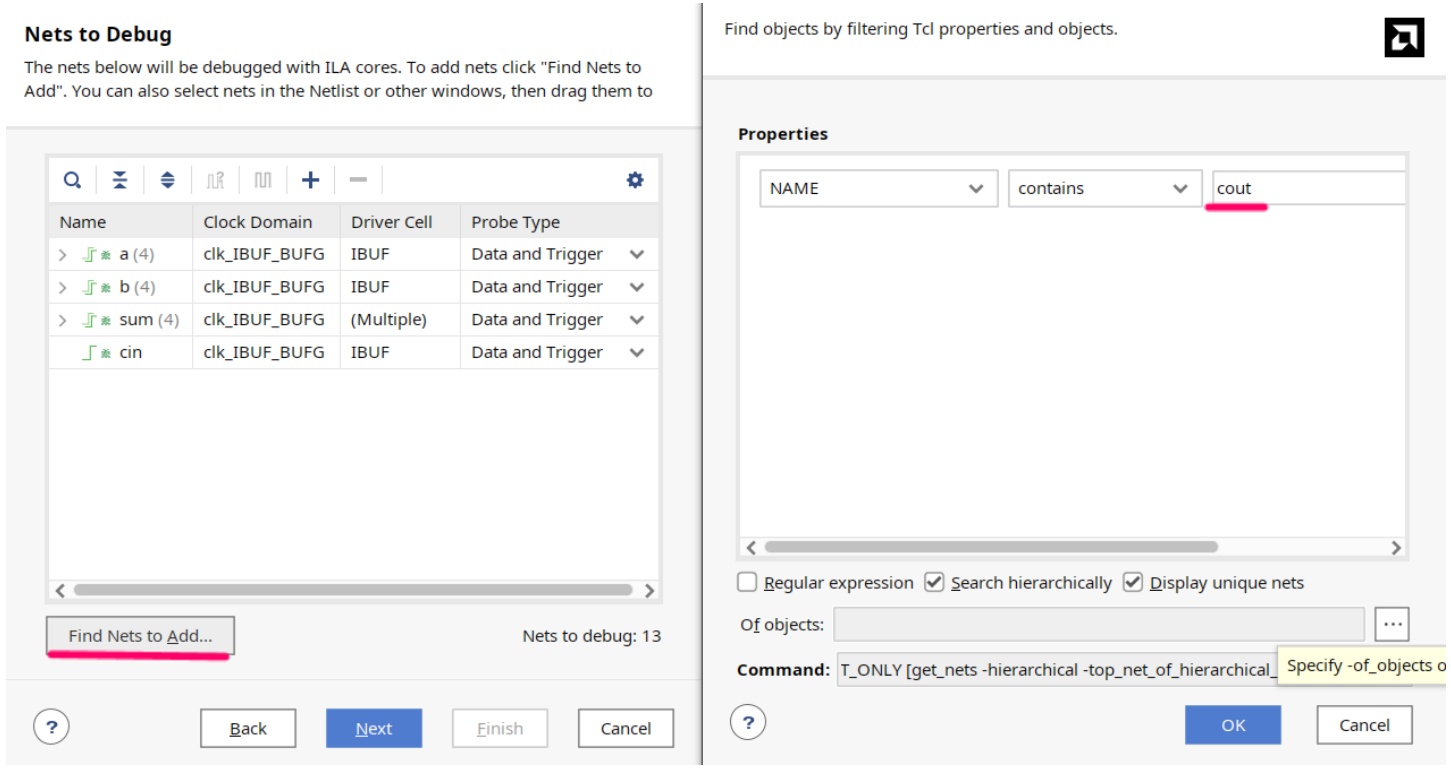


# SETTING UP THE ILA CORE FOR DEBUGGING

- On the left menu, **Run Synthesis**
- Click **Open Synthesized Design**
- In the top menu, select **Tools -> Set Up Debug.**
- Click through the wizard. Ideally, you should see all wires (a, b, cin, sum, cout) listed automatically
  - If not (as shown on the right), follow the following steps…

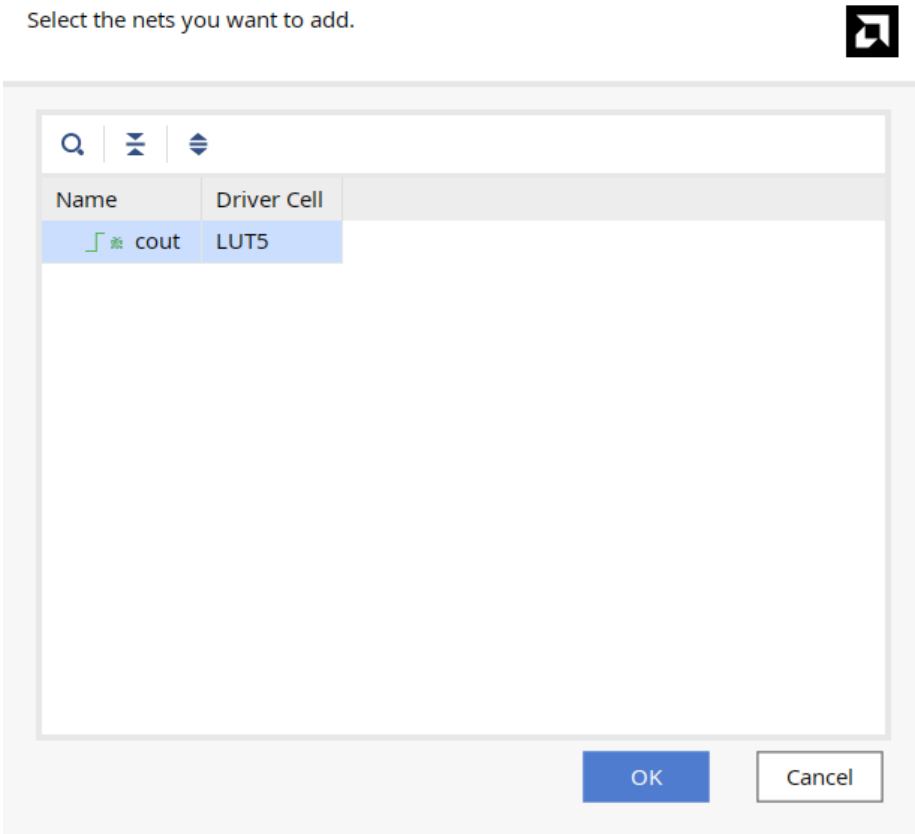
# IF NOT ALL WIRES SHOW UP IN DEBUGGING WIZARD...

1) At the bottom of the wizard, click on "Find Nets to Add..."
2) In the new window, type "sum" and press enter
3) Select the wire and press "OK"
4) Repeat steps for cout



# IF NOT ALL WIRES SHOW UP IN DEBUGGING WIZARD...

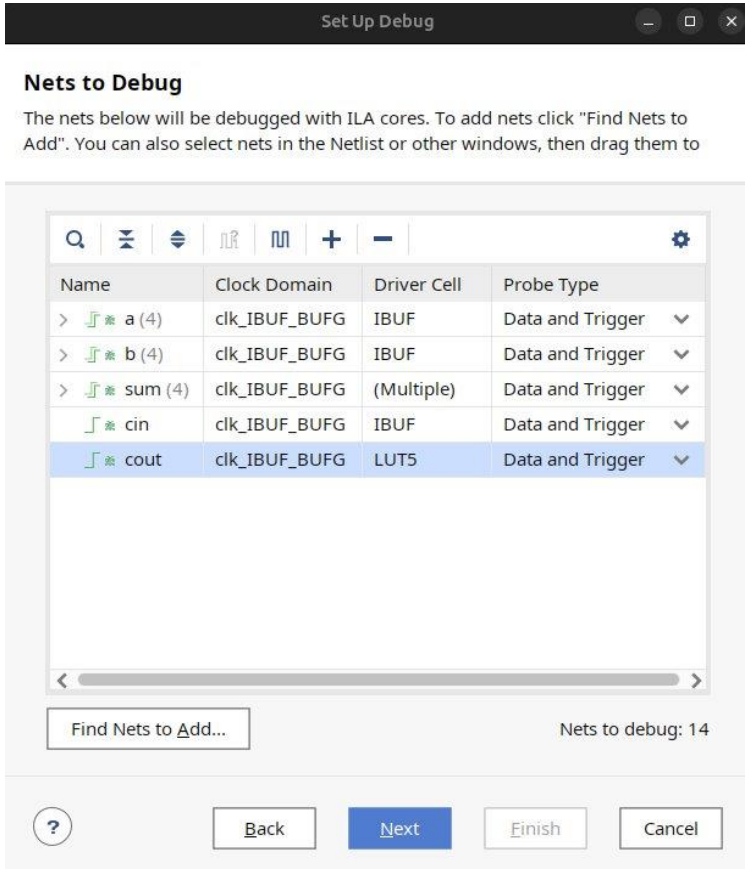
1) At the bottom of the wizard, click on "Find Nets to Add..."
2) In the new window, type "sum" and press enter
3) Select the wire and press "OK"
4) Repeat steps for cout



# FINISHING SETTING UP DEBUGGING CORE

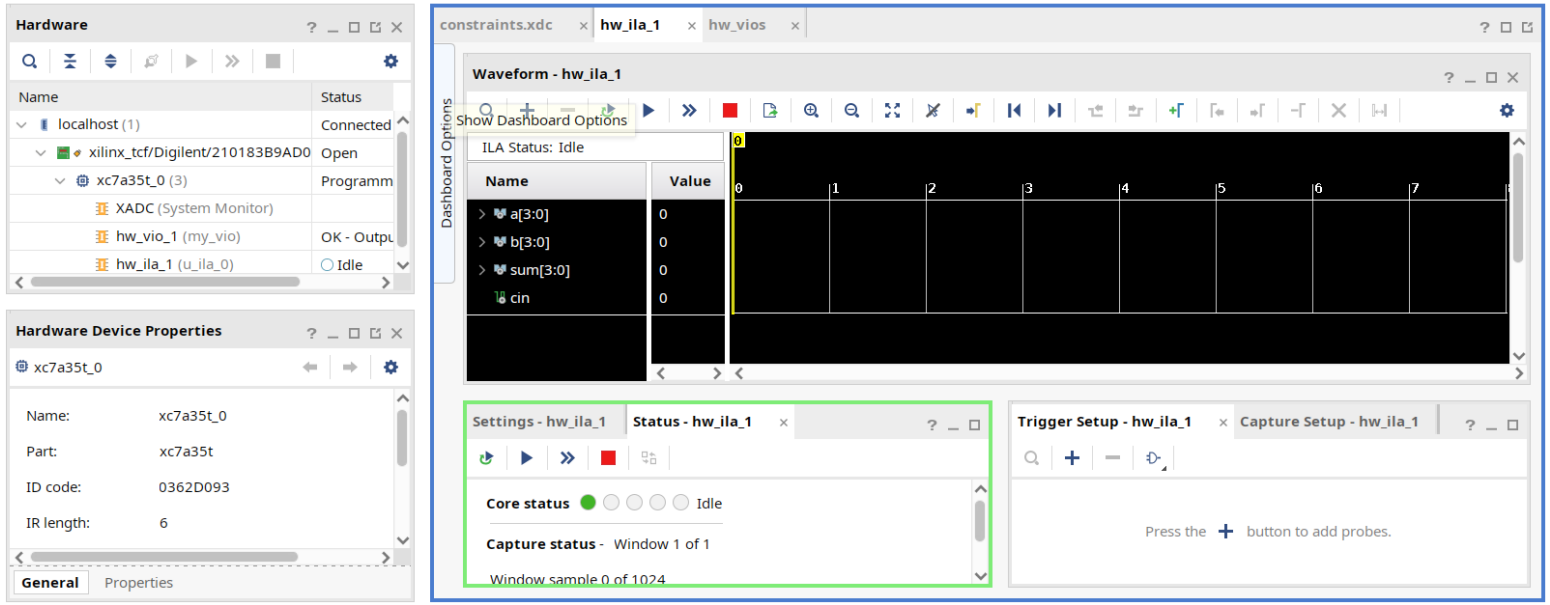
- You should now have the following on the right
- Click through and finish the Debugging Wizard
- Once done, hit CTRL+S to save the debug constraints to your imported .xdc file

- Your constraints file should now have both the settings of the pinouts, and the setup for the debugging core



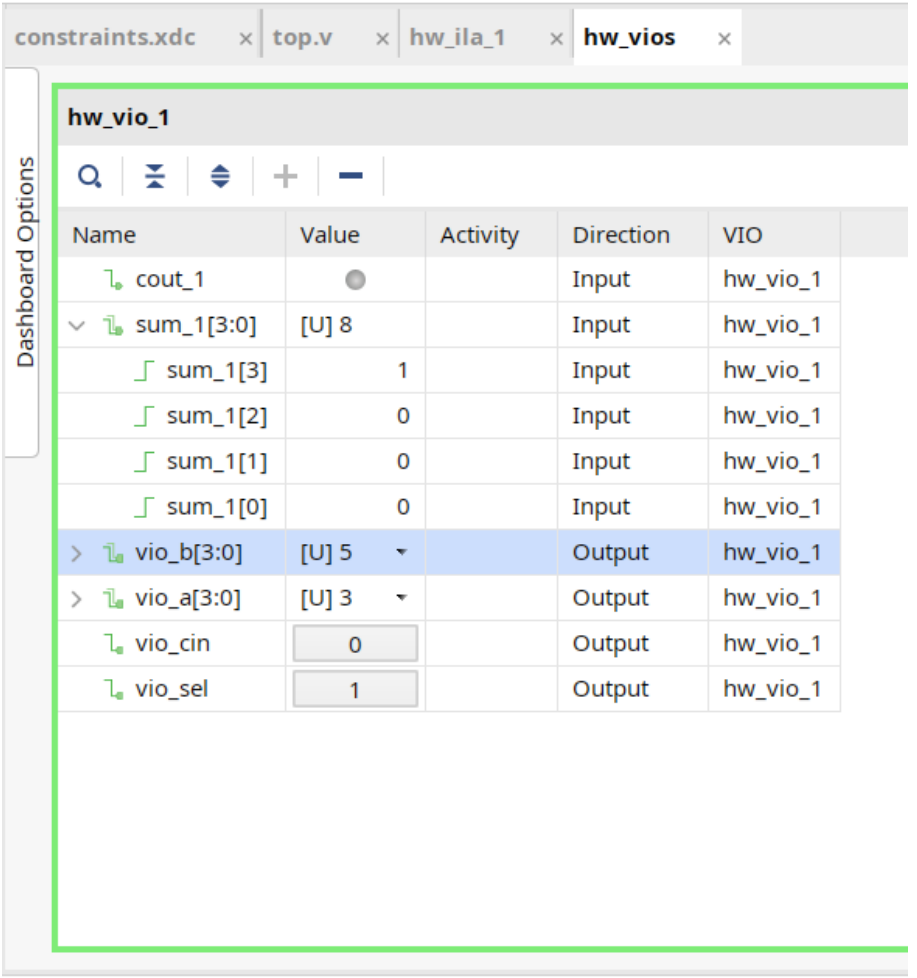
# GENERATE THE BITSTREAM

- On the left menu, select **Generate Bitstream**
- Program your Basys 3 board in the Hardware Manager.
  - The VIO dashboard and ILA dashboard will appear automatically in Vivado



# CONFIGURING THE VIO

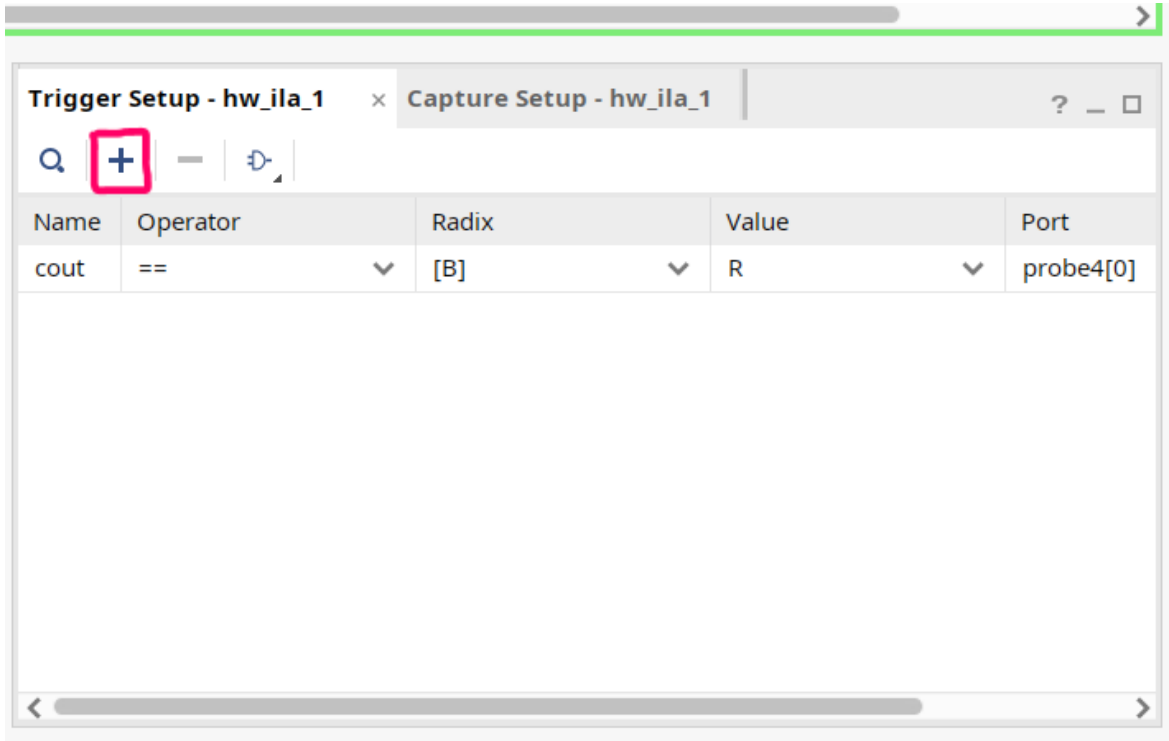
1) Go to the **hw_vios tab** in the Hardware Manager
2) At the top, press '+'
3) Select all of the wires present and press 'OK' (if the previous steps were done correctly, you should have the listed signals on the right). These will be your probes
4) Configure your probes by right-clicking on them
5) Make sure to have vio_sel = 1 for the virtual I/O to work

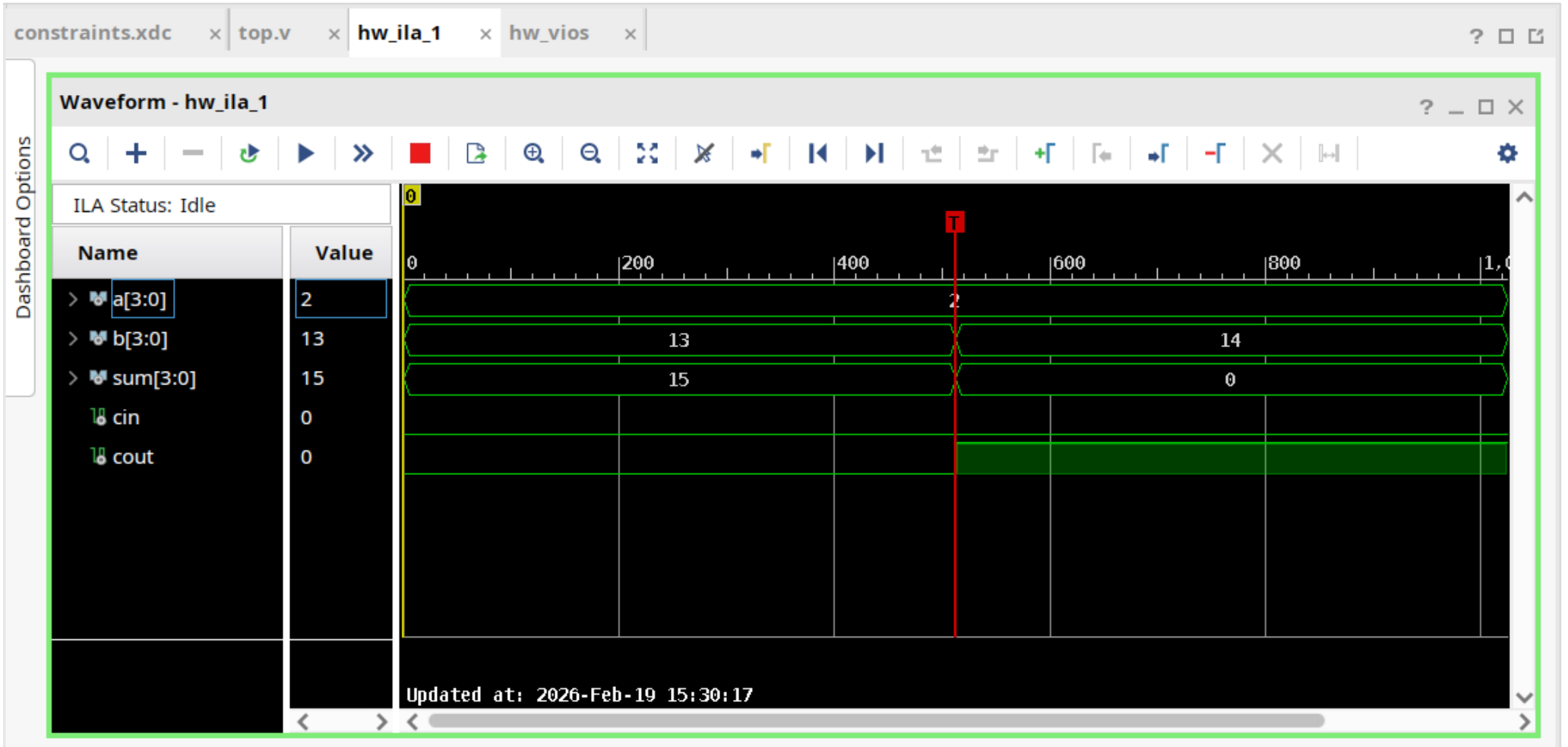


# SETTING UP ILA TRIGGERS 1

- Let's say we want to capture the exact clock cycle when overflow happens. We can setup a ILA trigger, to show when exactly overflow occurred in our setup

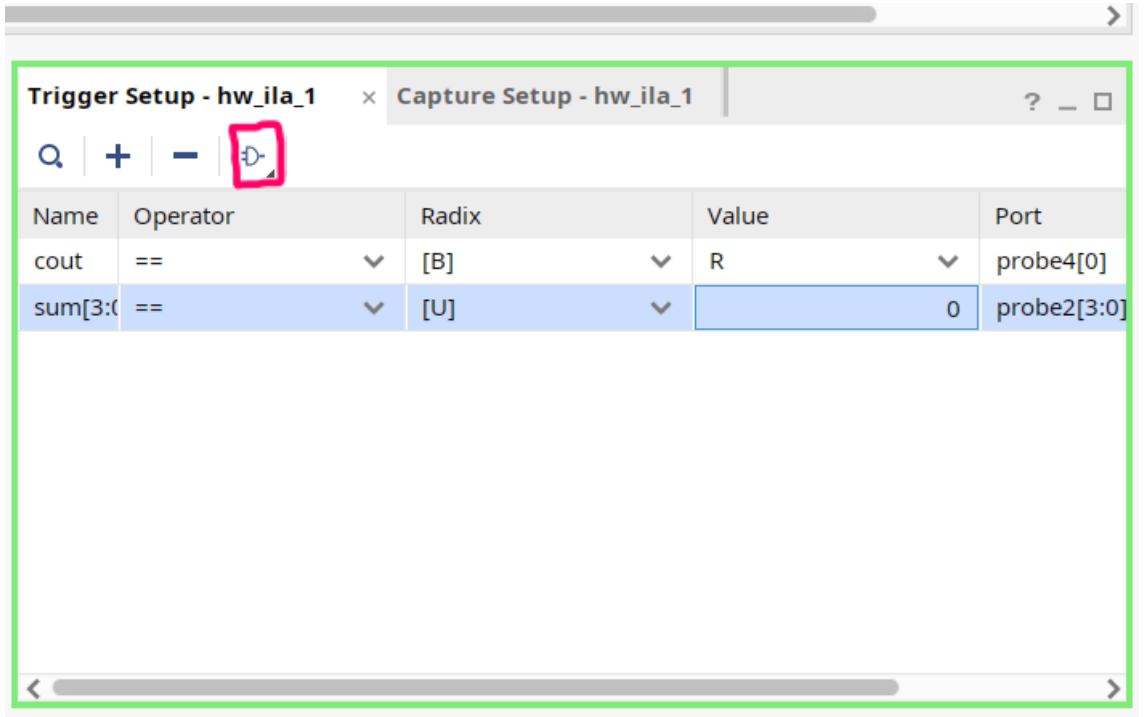
- In Hardware Manager, go to the **hw_ila_1** tab, and press the '+' in the **Trigger Setup** window
- Select **cout**
- Under **Operator,** ensure it's set to '=='
- Under **Value**, ensure it's set to 'R (0-to-1 transition)'
- Press the play button at the top of the **Waveform window** to arm the trigger, and play around with the VIO

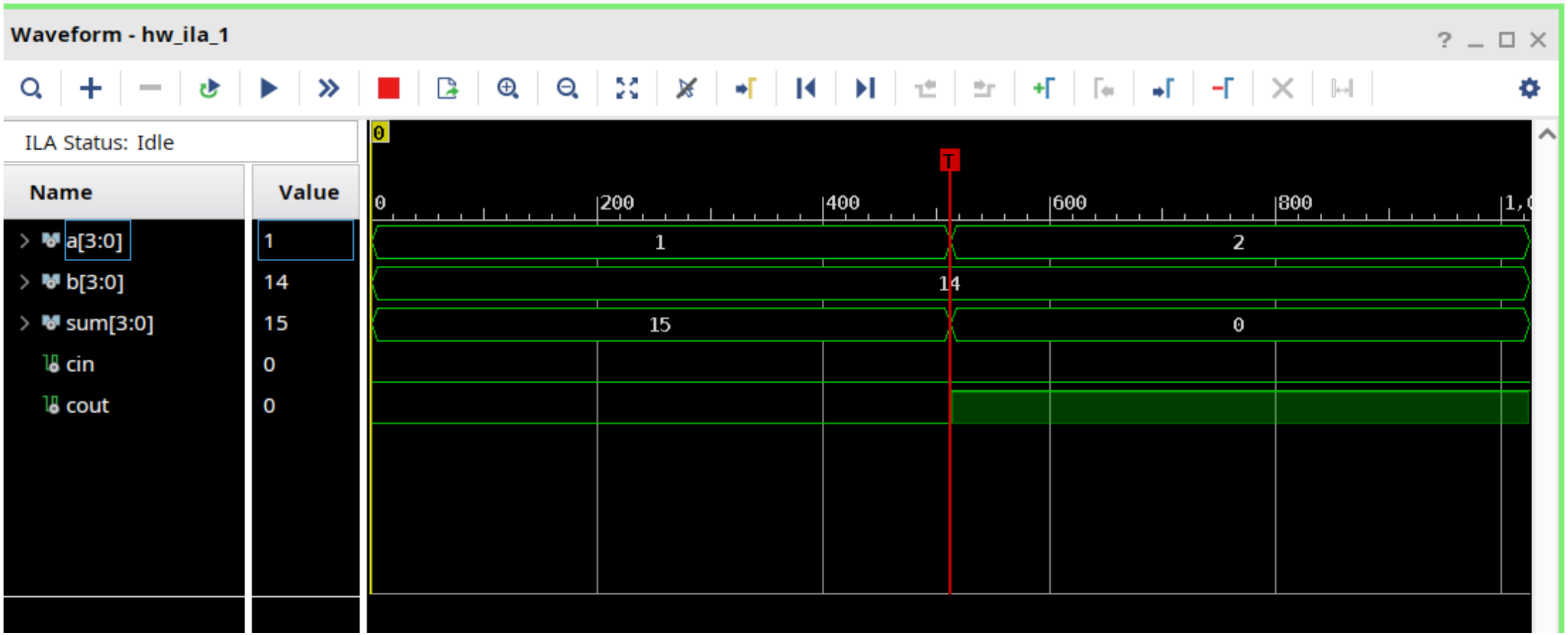



constraints.xdc
top.v
hw_ila_1
hw_vios
Dashboard Options
Waveform - hw_ila_1
ILA Status: Idle
Name
Value
a[3:0]
2
b[3:0]
13
sum[3:0]
15
cin
0
cout
0
0
200
400
600
800
T
2
13
14
15
0
Updated at: 2026-Feb-19 15:30:17

# SETTING UP ILA TRIGGERS 2

- We can set up compound/multiple trigger conditions in the ILA. Let's say we want to only want the ILA to trigger when BOTH cout = 1 and sum = 0
- Go back to the **Trigger Setup,** and add 'sum' as a trigger condition
- For 'sum', set **Operator** to '==' and **Value** to 0
- Under **Trigger Mode** button (red box), ensure its set to 'Global AND'.
- Press the play button on the Waveform window and play around with the VIO




Waveform - hw_ila_1
ILA Status: Idle
Name
Value
a[3:0]
1
b[3:0]
14
sum[3:0]
15
cin
0
cout
0
0
T
0
200
400
600
800
1
2
14
15
0