



Department of Electrical Engineering  
Indian Institute of Technology, Bombay

**Wadhvani Electronics Lab**

**EE214 - 2022**

**Scan Chain Demo**

***Naef Ahmad***

- Need for Scan Chain
- Basic understanding of JTAG and Scan Chain
- What is Virtual JTAG IP
- How Scan Chain works
- How to Run Scan Chain using virtual JTAG

Can We use this (Pin Planning ) method to verify any design?

- Possible Issues?

1. Number of Switches and LEDs are Limited
2. Amount of time to verify all input combinations

- Already written a Testbench for verification
- Verified Design through RTL Simulation
- Hardware may have some timing Differences/ small faults.
- Design may not be synthesizable(works only in simulation )
- All simulated design may not work perfectly on board
- Before releasing your design, need to confirm if it works on actual hardware or not.

Till now -

1. Written VHDL code for Full Adder, Multiplexer, Decoder etc.
2. Verified through a Testbench using a TRACEFILE (RTL Simulation)
3. Verified on Xenon board using Switches and LEDs
4. Number of input- output pins for Full Adder = 3/2, Prime/Fibonacci Detector = 4/1.

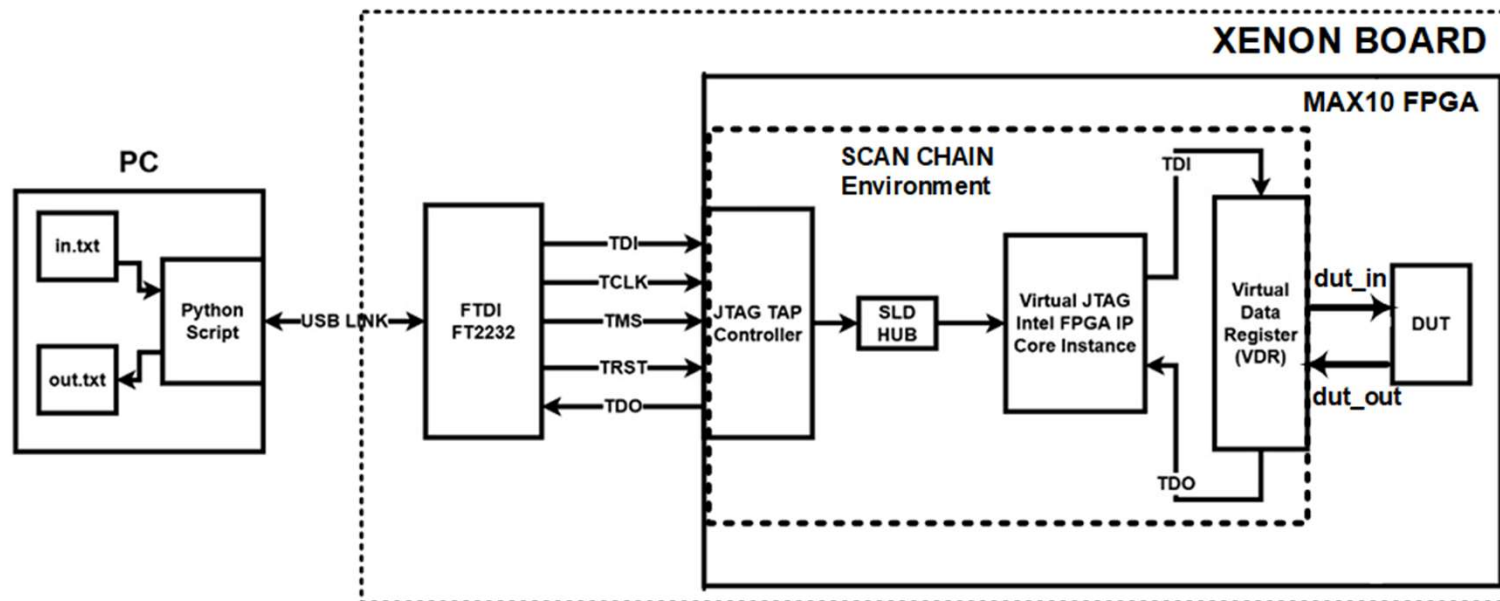
- 4-bit Adder-Subtractor
- Number of input/output Pins = 8/9
- Available switches/LEDs on krypton Board = 8/8
- How to give inputs and Check output?
- Time required to check for all input combinations? - 256 combinations for adder
- Solution – Scan Chain
- For a design with any number of input and output pins

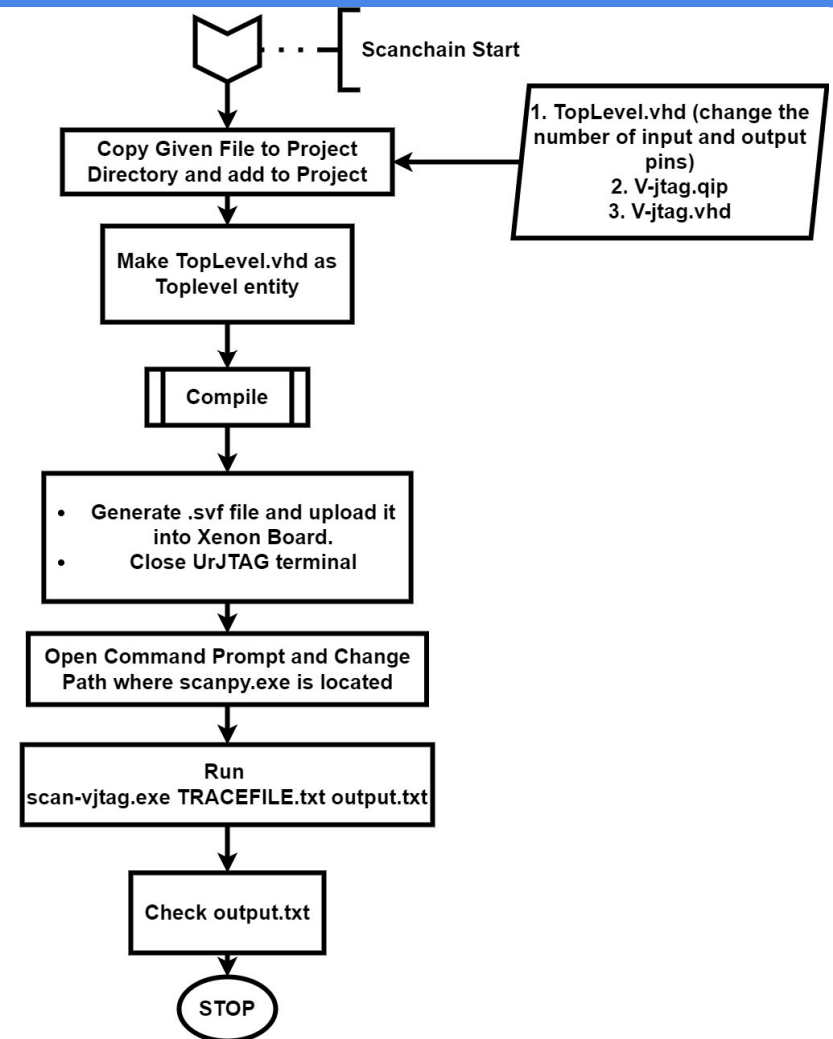
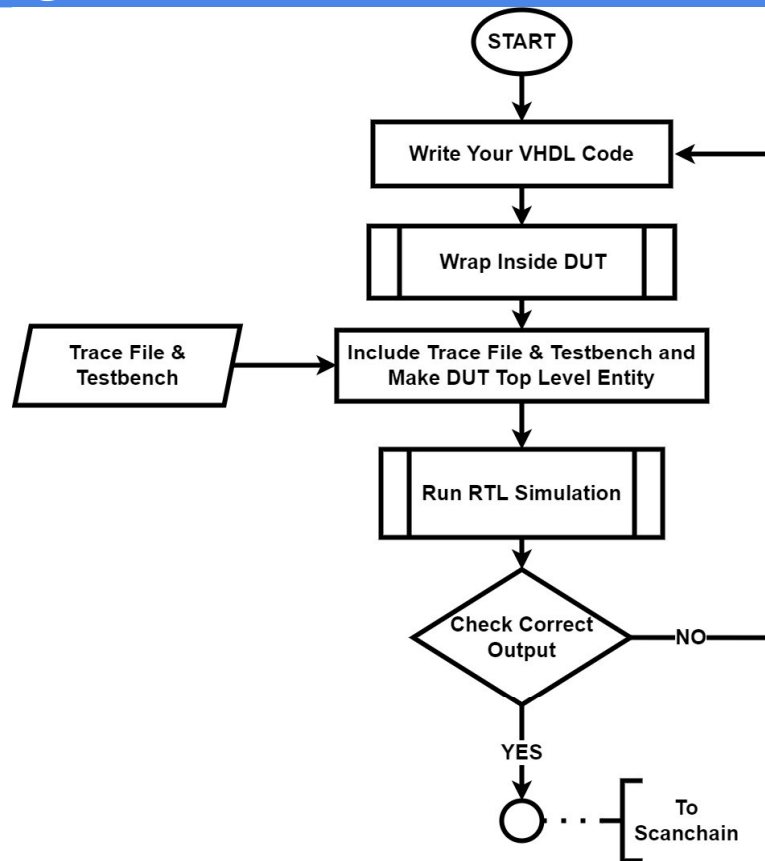
- Can we use existing Tracefile?
- Possible Solution :
- send inputs through USB to FPGA
- Allow FPGA to react to the inputs
- Read outputs generated by FPGA through USB
- Compare generated output with golden output

- Scan Chain based testing method became very popular
- Industry standard for verifying designs and testing PCBs
- Subsequently, used for uploading design on FPGA/CPLD boards.
- So much that it became an IEEE standard – IEEE-1149.1
- Joint Test Action Group - 1990
- One stop Solution for Programming and testing.
- Every IC has Basic JTAG hardware Built in it.
- Different Manufacturers follow the same standard



- FPGAs already have the basic JTAG Hardware built-in
- To Control the basic hardware, need to write some VHDL code
- Alternative - Virtual JTAG Intel FPGA IP Core
- Intellectual Property: source code not visible





# Steps to run Scan Chain

IIT Bombay

Install Python, PIP, Python Packages

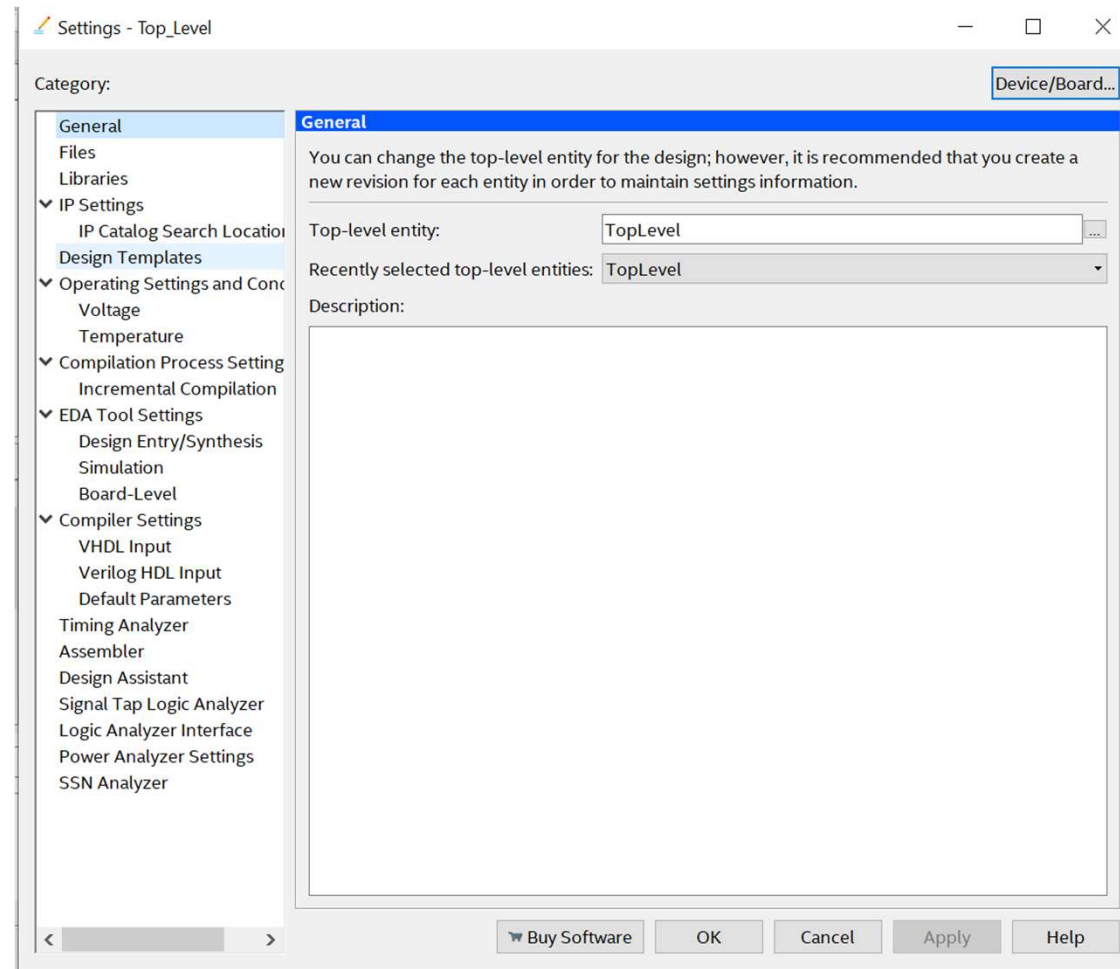
- Install Python 3.7 or Python 3.8
- Install PIP
- Install the following packages using the commands as shown
  - `pip install bitstring`
  - `pip install ftd2xx`
  - `pip install pyusb`

- Necessary files
  - DUT wrapped design file and other dependent design files with working RTL simulation
  - Tracefile
- Files Required for Scan Chain (Shared)
  - TopLevel.vhd
  - Folder v\_jatg
  - scan\_vjtag.exe file
  - scan\_vjtag.py
- Copy these files and paste them in the project directory

# Modifications to Quartus Project

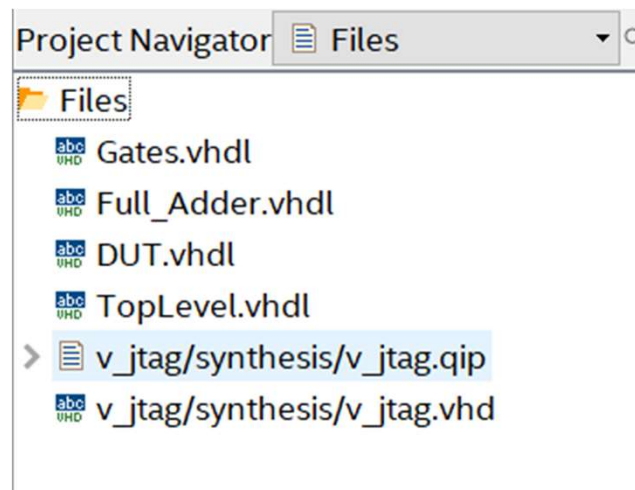
## Modifications to Quartus Project

- Add TopLevel.vhdl to the Quartus Project
- Change the number of inputs and outputs to the design
- This change is the same as what was done for testbench
- Go to Assignments -> Settings -> General
- Set top level entity as TopLevel as shown



# Modifications To Project

- Add the following files in the path: v\_jtag\synthesis
  - v\_jatg.vhd
  - v\_jtag.qip
- For a sample full adder project, these are the list of files that have to be present
- Note that the presence or absence of testbench in the project does not matter during scan chain



- Note – Pin Planning is not needed.
- Note – Keeping Testbench in project doesn't matter
- Do a full compilation of the design (note you need to repeat the task since the top level entity has changed)
- Generate SVF file again
- Dump the SVF file on the Xenon Board
- Close the Urjtag Terminal

- Open Command Prompt and change path using cd command
- In case the project is on C drive, use the following
  - cd ..\..\
  - cd <absolute path of folder>
- This path can be copied from windows explorer
- In case you are in D/E/F drives, enter the following
  - D: or E: or F:
  - cd <absolute path of folder>

```
C:\Users\Ahmad>cd ..\..\
C:\>cd C:\Scan_Chain_Files
```

```
C:\Users\Ahmad>D:
D:\>cd 214_summer_22\Scan_Chain_Files
D:\214_summer_22\Scan_Chain_Files>
```



- Run the following command to observe the output
  - `scan_vjtag.exe TRACEFILE.txt out.txt`
- `scan_vjtag.exe` is the name of the file to run
- `TRACEFILE.txt` needs to be a text file in the specified format. It is an input to the exe file
- In case you have renamed it, write the new file name in the command
- `out.txt` is a file that is generated by the code. There need not be an existing file with the name. In case there is, the program will overwrite the file without warnings
- Check `out.txt` to find whether your design has successfully passed all test cases or not

```
D:\IIT_Bombay\WEL\NPTEL\Scan_Chain_Test\Full_Adder_code_files>scan_vjtag.exe TRACEFILE.txt out.txt
{'type': 6, 'id': 67330064, 'description': b'Dual RS232-HS A', 'serial': b'A'}
```

- In case the .exe file is treated as a virus by your antivirus software, then give it permission by allowing the file in settings
- If the issue still persists, the scan\_vjtag.py file is provided as backup
- Verify installation of the pip packages in slide 12 (enter installation command again and you will get something similar to the following if it is installed successfully)

```
D:\IIT_Bombay\WEL\NPTEL\Scan_Chain_Test\Full_Adder_code_files>pip install pyusb  
Requirement already satisfied: pyusb in c:\users\abhin\appdata\local\programs\python\python38-32\lib\site-packages (1.0.2)
```

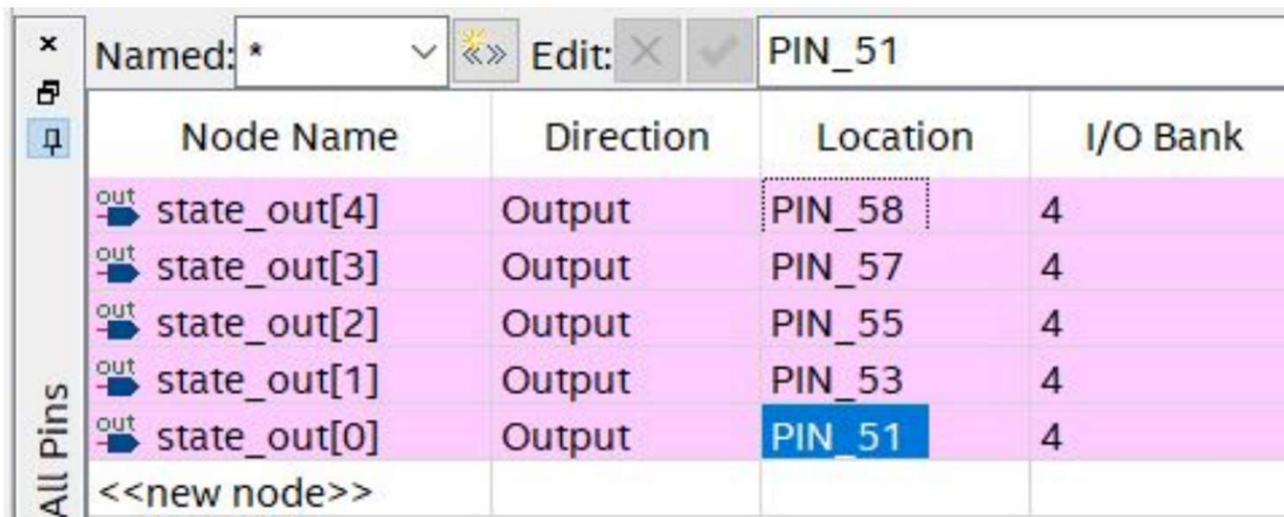
Repeat all steps until the command scan\_vjtag.exe TRACEFILE.txt out.txt

- Enter the following command instead of that
  - python scan\_vjtag.py TRACEFILE.txt out.txt

```
D:\IIT_Bombay\WEL\NPTEL\Scan_Chain_Test\Full_Adder_code_files>python scan_vjtag.py TRACEFILE.txt out.txt  
{'type': 6, 'id': 67330064, 'description': b'Dual RS232-HS A', 'serial': b'A'}
```

In case Quartus throws an error unrelated to your design, then do the following

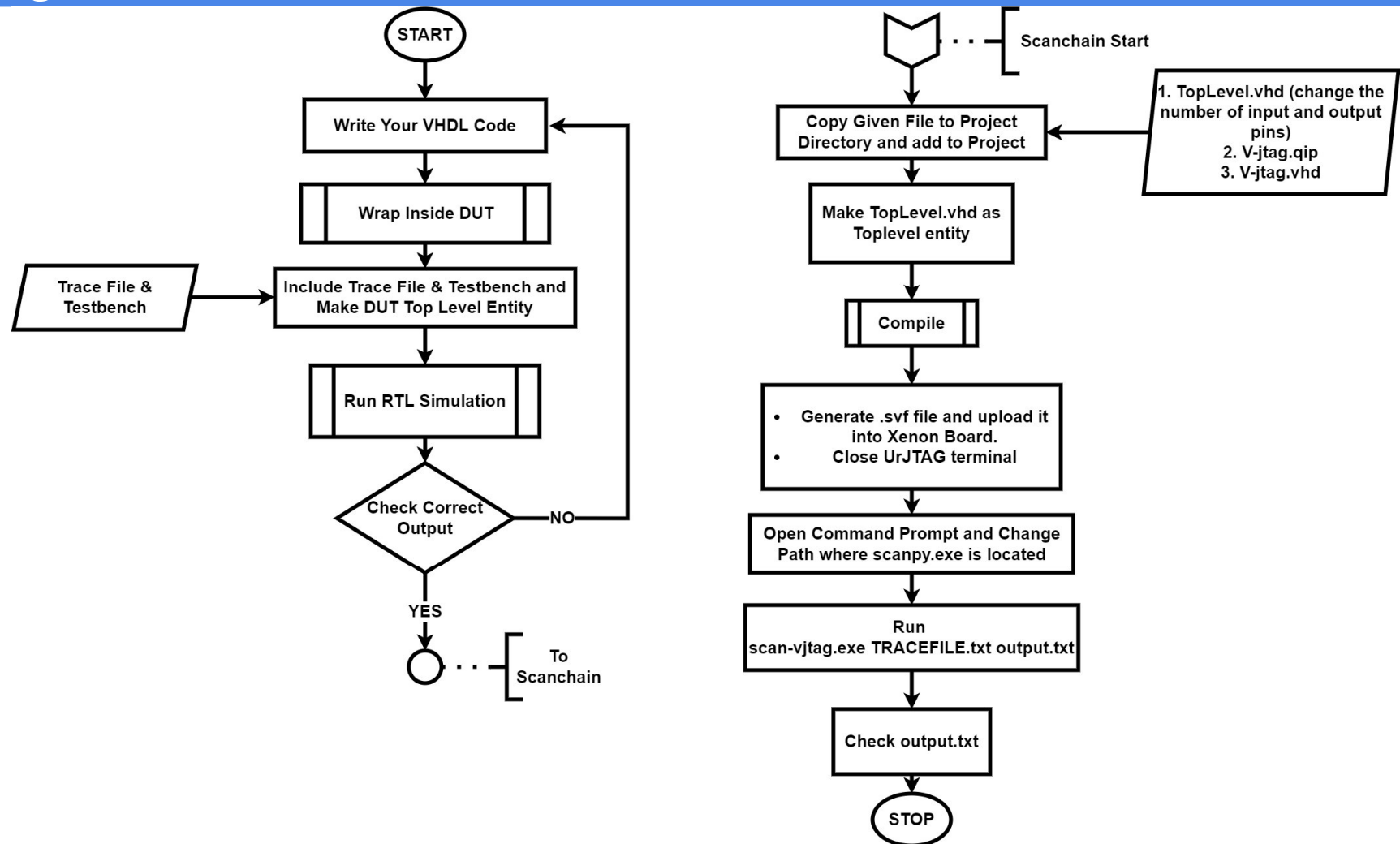
- Verify the port names in v\_jtag.vhd with the component declared in TopLevel.vhd
- Before dumping the SVF file, open pin planner and map it as shown in image (also mentioned in comments of TopLevel.vhdl)
- Compile design again, and proceed from slide 16



The screenshot shows the Quartus Pin Planner window. The 'Named:' dropdown is set to '\*'. The 'Edit:' buttons are visible. The table below lists the node names, their directions, locations, and I/O banks. The signal 'state\_out[0]' is mapped to 'PIN\_51' in I/O Bank 4, which is highlighted in blue. Other signals are mapped to pins 58, 57, 55, and 53, all in I/O Bank 4. A new node entry is at the bottom.

Node Name	Direction	Location	I/O Bank
state_out[4]	Output	PIN_58	4
state_out[3]	Output	PIN_57	4
state_out[2]	Output	PIN_55	4
state_out[1]	Output	PIN_53	4
state_out[0]	Output	PIN_51	4
<<new node>>			

- Testbench -> Hardware is Simulated and design's functionality is checked
- Scan-Chain -> Actual hardware is given the inputs and Outputs are then compared



# Questions?