Introduction to hardware design tools and methods

Digital Circuits Lab (CS-220)

Spring ‘25

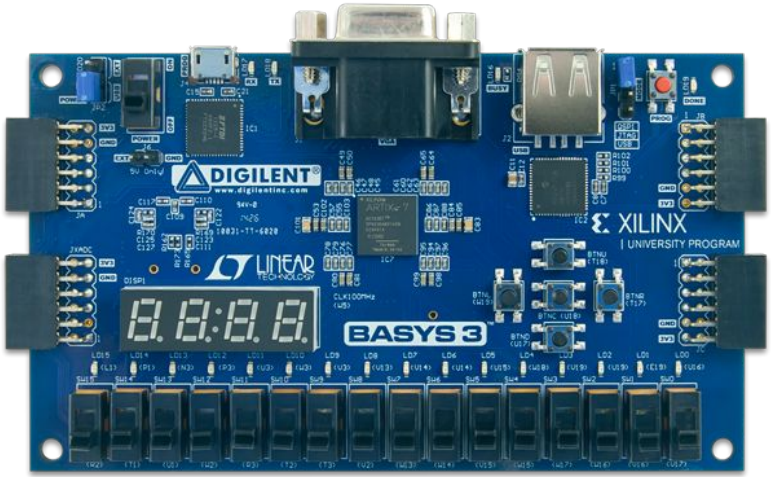
Sotiris Totomis & Alexandros Fourtounis

Overview

- Field Programmable Gate Arrays (FPGAs)
- Simulation
- Synthesis
- Implementation
- Tools installation
- Vivado tool flow & walkthrough
  - Simulation & Synthesis steps
- Verilator
  - Simulation steps
- Hands-on SystemVerilog demo
  - How to develop
  - How to debug

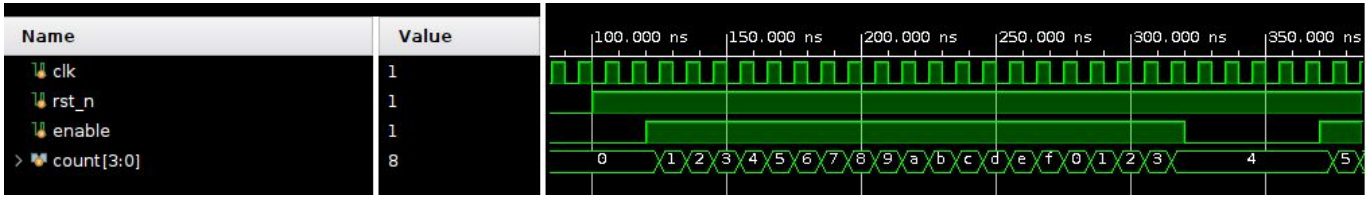
Field Programmable Gate Arrays (FPGAs)

- FPGAs are programmable integrated circuits
- Consist of configurable logic blocks (CLBs)
  - Look-Up Tables (LUTs) for combinational logic
  - Flip-Flops (FFs) for sequential logic
  - Multiplexers (MUXs) for routing
- Advantages
  - Flexibility to reprogram for different applications
  - Lower development costs compared to ASICs
- They are used in various fields
  - Signal processing, aerospace, AI/ML
- FPGA vs. ASIC
  - ASIC is application specific (duh!)
  - FPGAs are better options for prototyping
- Some well-known vendors
  - **Digilent**
  - **Xilinx (AMD)**
  - Altera (Intel)



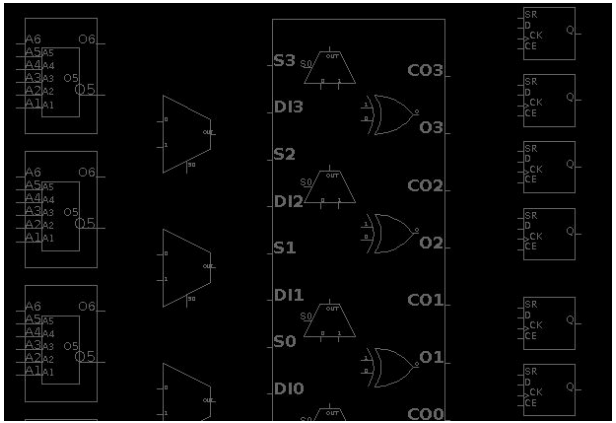
Simulation

- The process of modeling and analyzing the behavior of digital circuits using software tools
  - A critical step in the design and verification before the hardware implementation (FPGA)
  - Register Transfer Level (RTL) simulation tests the code written in SystemVerilog
  - Simulation waveforms are our main debugging tool!
- Simulator alternatives
  - Vivado Design Suite includes a built-in simulator
    - Provides a graphical waveform for signal inspection
  - Verilator is an open-source simulator
    - Produces waveforms (vcd files), it requires a viewer to open them (GTKWave)
- Why do we need a testbench (TB) for simulation? TB is not synthesizable!



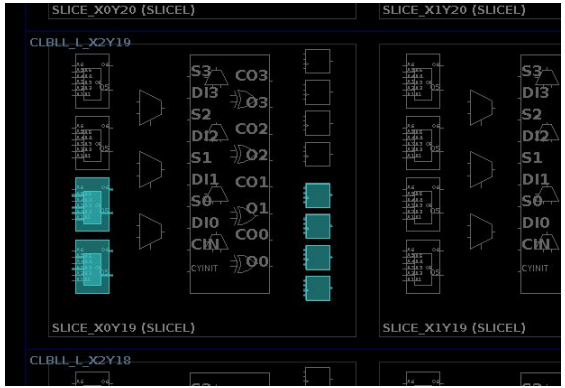
Synthesis

- The process of converting high-level Hardware Description Language (HDL) code into gate-level netlist (logic gates, FFs, etc.)
- Reports
  - Space utilization
  - Timing & critical path
- What do we need to synthesize our code?
  - A top module, acts like a wrapper for our synthesizable code
    - It instantiates the synthesizable modules
    - It connects them to the FPGA’s I/O pins via constraint files (XDC)



Implementation

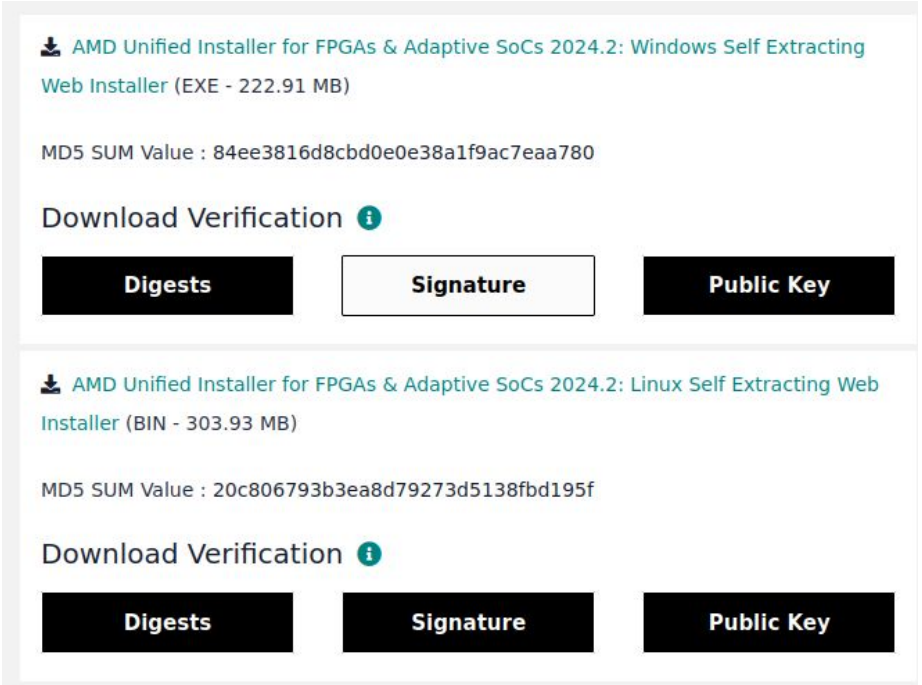
- The process of converting synthesized design into a physical layout that can be finally programmed onto an FPGA
- Places the design and routes the connections, meeting timing and resource constraints
- Timing, and congestion challenges - but not our problem for now!



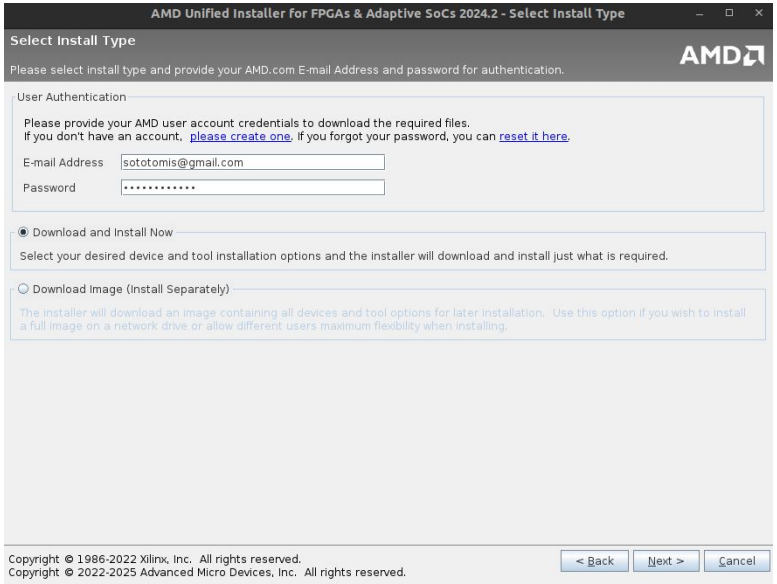
Tools installation

- Simulation options
  - Vivado, but it has high disk space requirements (~70 GB)
  - Alternatives for Linux/MacOS
    - Verilator + GTKWave (follow the site's instructions)
    - Icarus Verilog + GTKWave (follow the site's instructions)
  - You are able to use the department's machines for simulations
    - Don't forget ssh -X for display forwarding!
- For the FPGA flow (synthesis & implementation) we will use only Vivado during lab hours

Vivado installation - Version/OS/Account

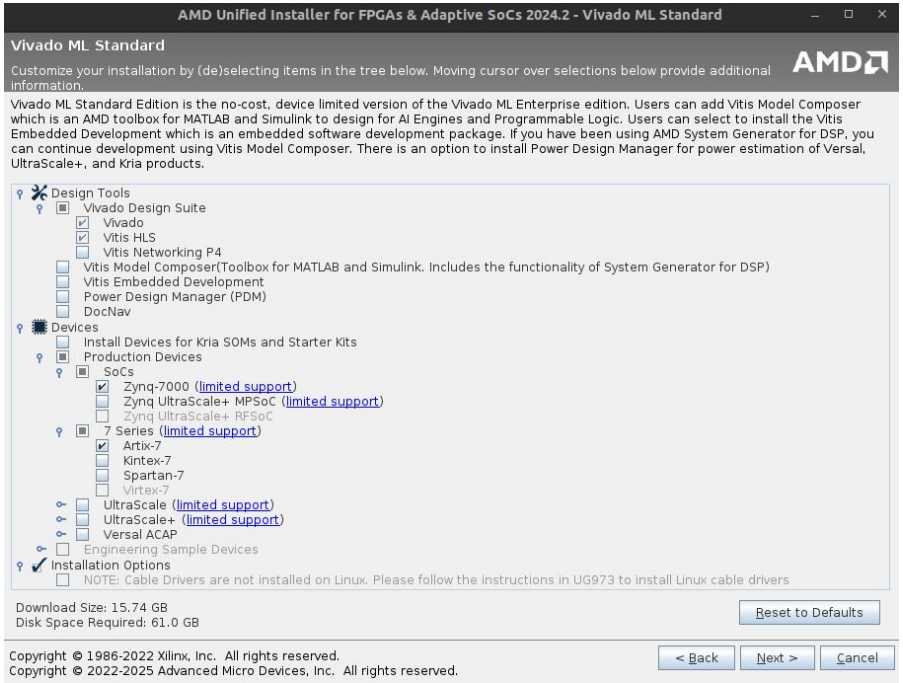


Vivado installation - Initial setup



Linux: If you install Vivado in a system's directory, you have to execute the installer binary with sudo permissions

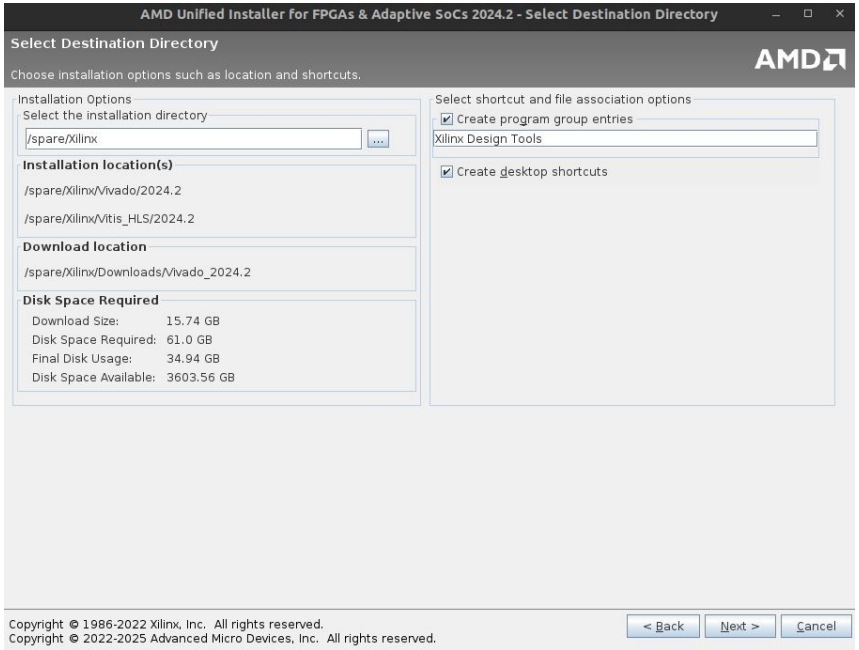
Vivado installation - Devices & Basys 3 board files



*If you would like to test synthesis and implementation locally, you have to copy to your vivado directory the associated board files. Do this step **after** Vivado installation completes.*

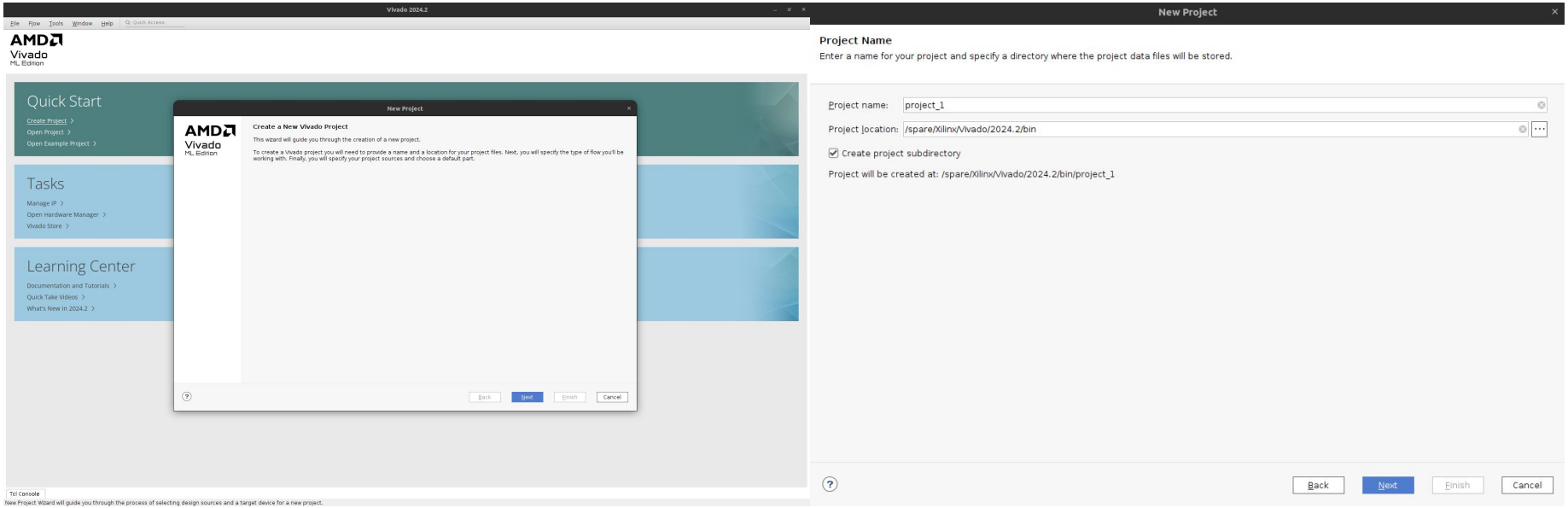
https://digilent.com/reference/software/vivado/board-files

Vivado installation - Final step, wait & run

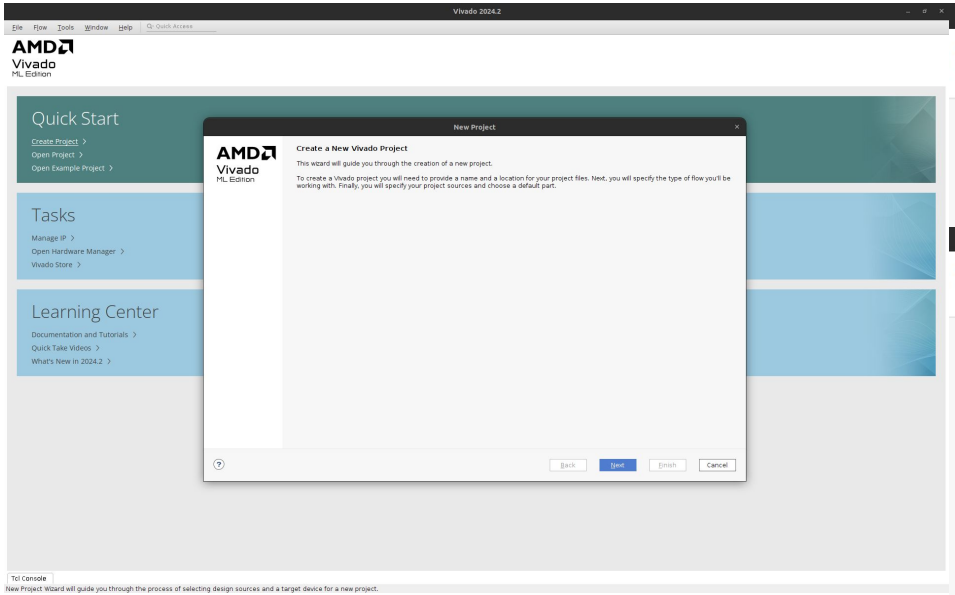


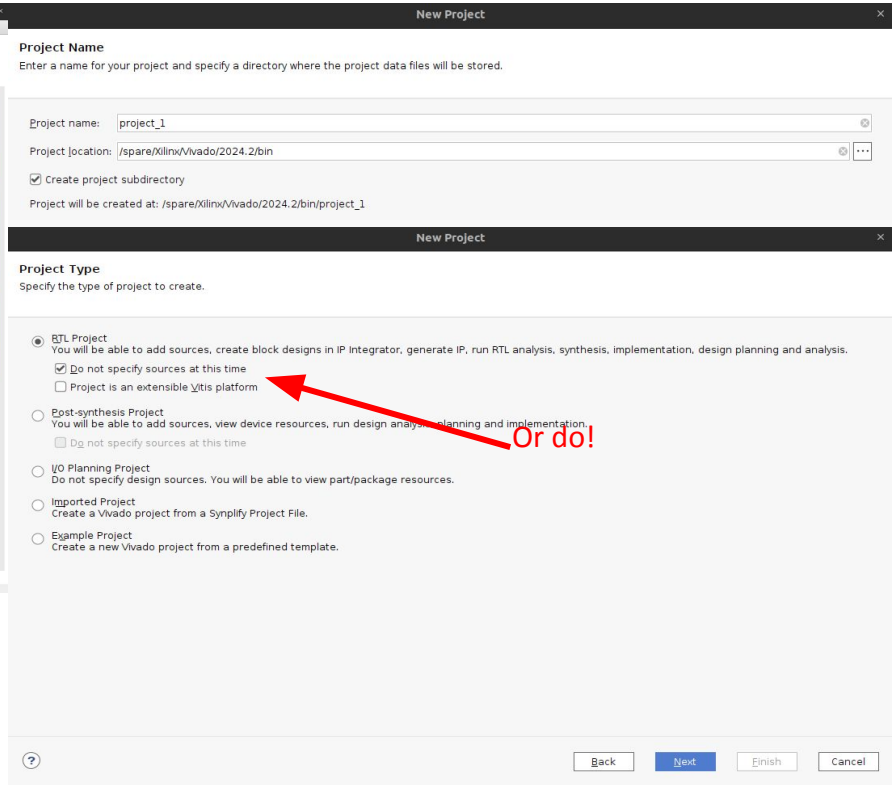

```
sototo@rvila:/spare/Xilinx$ pwd
/spare/Xilinx
sototo@rvila:/spare/Xilinx$ cd Vivado/
sototo@rvila:/spare/Xilinx/Vivado$ ls
2024.2
sototo@rvila:/spare/Xilinx/Vivado$ cd 2024.2/
sototo@rvila:/spare/Xilinx/Vivado/2024.2$ source settings64.sh
sototo@rvila:/spare/Xilinx/Vivado/2024.2$ cd bin/
sototo@rvila:/spare/Xilinx/Vivado/2024.2/bin$ sudo ./vivado
```

Vivado - Project creation

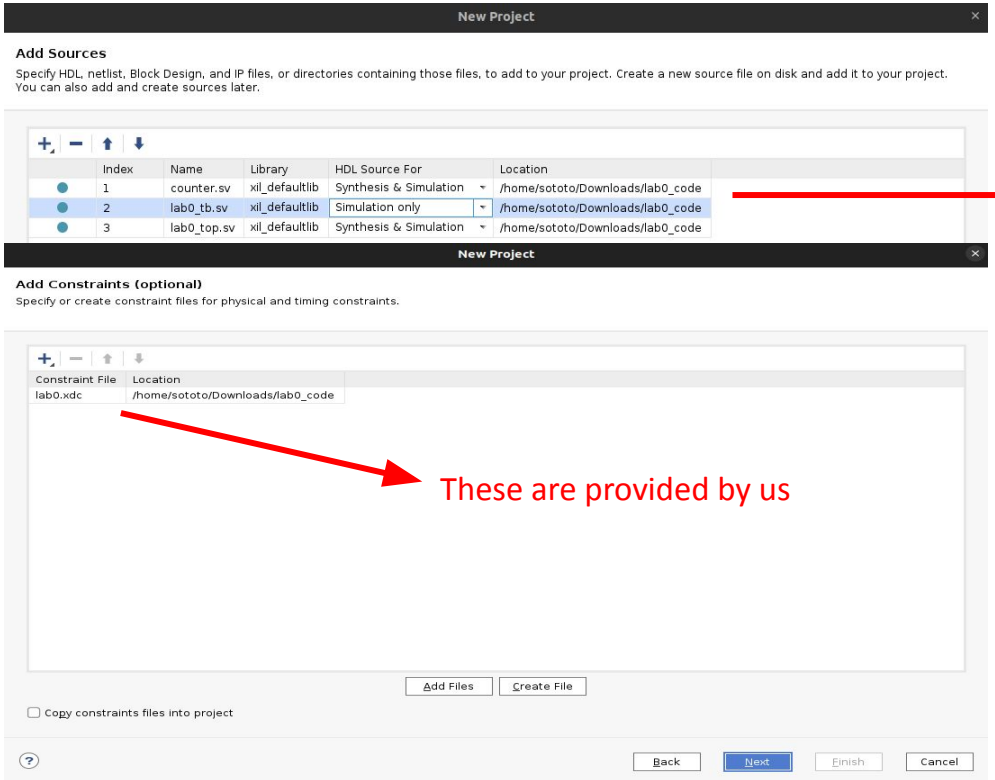


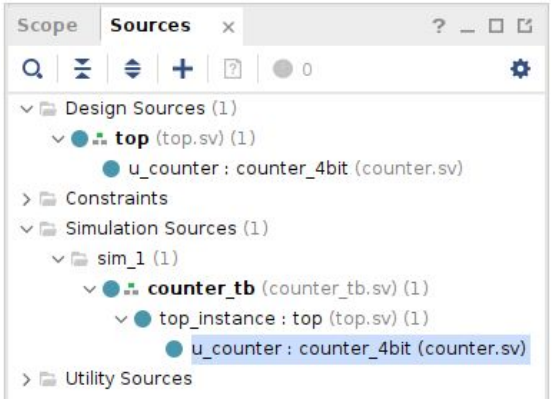
Vivado - Project creation



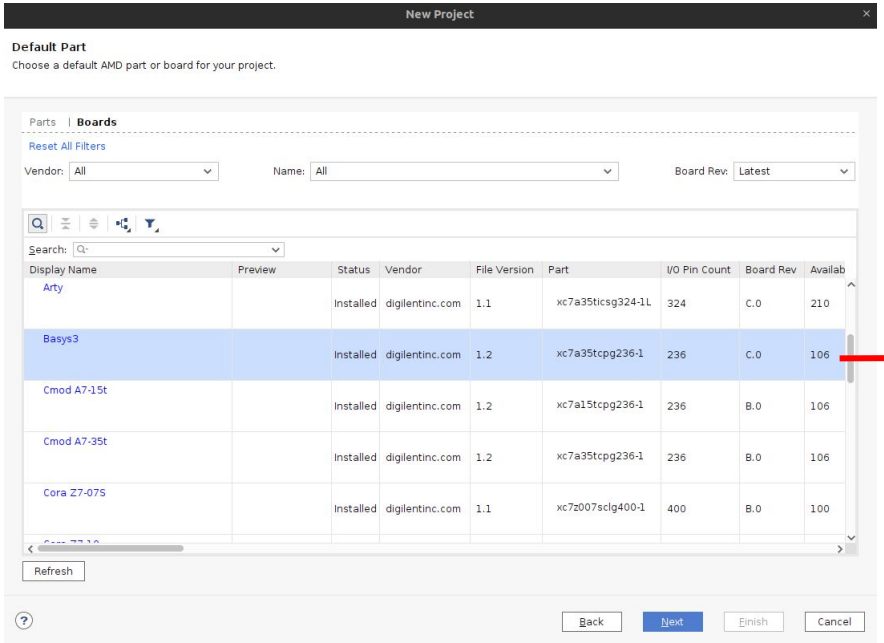


Vivado - Import files & constraints



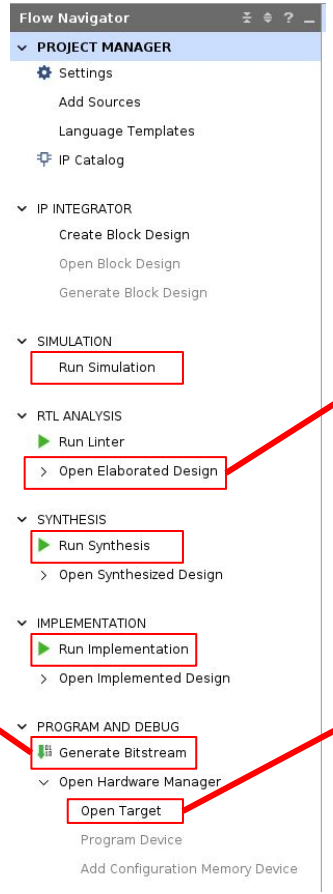


Vivado - Board selection



https://digilent.com/reference/software/vivado/board-files

Vivado - Flow Navigator



Displays an abstract schematic of your design

Generates the binary file that contains the configuration data to program an FPGA

Opens the connected FPGA device to program it. Don’t forget to switch on the FPGA!

Verilator

- ssh -X [csd_host]
- Follow the site instructions
- Modify the Makefile prototype for each new simulation
- Must check simulator flags in source files!

```
SRCS=lab0_tb.sv lab0_top.sv counter.sv
TOP_MODULE=lab0_tb
#TOP_MODULE=example
HY220_TOOLS_DIR=~hy220/tools
include $(HY220_TOOLS_DIR)/common/Makefile.include
```

```
// increase with timer
counter #(
  .N(27),
`ifdef XILINX_SIMULATOR
  .MAX(9)
`else
  .MAX(99999999)
`endif
)
```

```
// increase with timer
counter #(
  .N(27),
`ifdef XILINX_SIMULATOR
  .MAX(9)
`else
  .MAX(9)
`endif
)
```

Switch to demo

- Various module examples
- Hierarchy of files for design and simulation
- always_comb vs always_ff
- synchronous vs. asynchronous clock
- Debugging and waveforms
- Synthesis & implementation steps