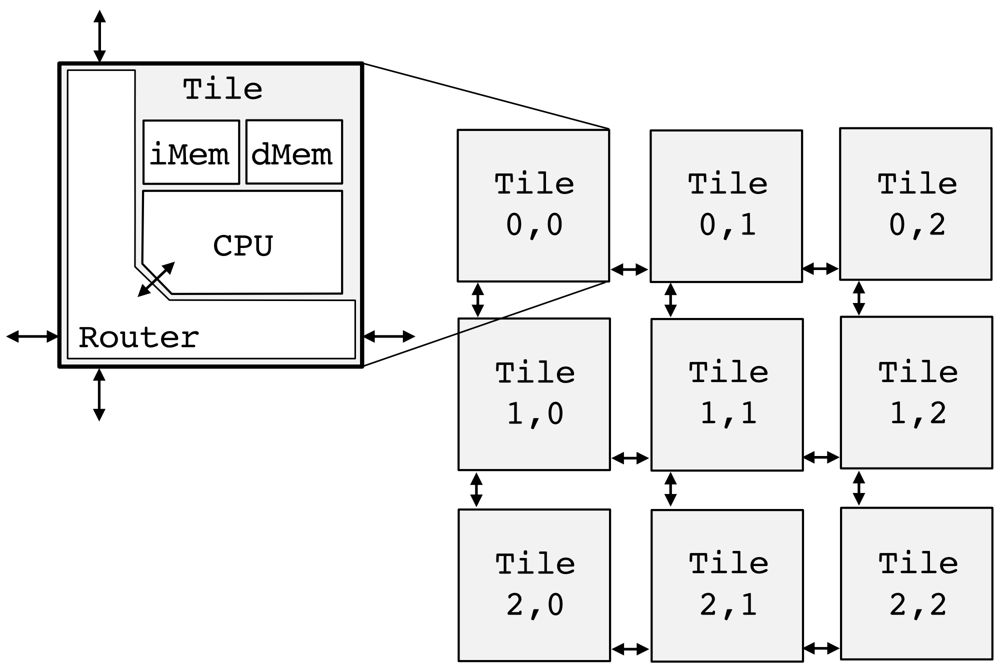
ECE 558/658 VLSI Design -- Fall 2021
Lab 5: Simulation, Synthesis, and Physical Design for a Simple NoC

In this lab you will gain experience of creating a digital design for a system that is much larger than the accumulator from the prior labs. Specifically, the design you implement is a simple Network-on-Chip comprising tiles each having a router and associated processing core. This is an example of a chip multiprocessor (CMP) NoC as discussed in lecture. You will work with this design to perform simulation with Mentor Graphics ModelSim, synthesis using Synopsys Design Compiler, and physical design (place & route) using Cadence Encounter. The synthesis and physical design steps of this lab mirror the corresponding steps in lab 4. The figure below shows the NoC system for the assignment. Each tile includes a single cycle MIPS processor and a simple NoC router. The Verilog HDL for this design is provided with the lab materials.



The flits transferred over the network channels are 12-bit wide, and the format of the flit is shown in the figure below. The upper 4 bits of the flit encode the destination address, with 2 bits for each the x and y fields of the address; the lower 8 bits carry payload. In a real NoC, a packet would be split into multiple flits as was discussed in lecture, and only the head flit would carry the destination address field.

Complete the steps described in the remainder of this document. Be sure to include each required item (indicated by **SUBMIT**) in your report. You must also explain what you did and why; images alone are not sufficient.

Part A: Analysis and Simulation

Step 1: Basics of Routing [5 pts]

The router implements an XY dimension-ordered routing scheme, which is a restricted turn routing algorithm, and hence guaranteed to never create routing deadlocks. Packets are routed horizontally (in the

X-dimension) to the column of their destination, then routed vertically (in the Y-dimension) within the column to reach the destination. Answer the following basic questions about the mesh NoC.

- Identify the path for a packet sent from the core (0, 0) to (2, 2) for a mesh topology. List all tiles on the path in the order they are traversed by the traffic.
- Similarly, identify the path that would be taken from core (2,2) to (1,1).

SUBMIT: Description of the two paths.

Step 2: Simulate 3x3 mesh [10 pts]

Now simulate the 3x3 mesh NoC system. Create a directory for this project on the vlsicad1 or vlsicad2 server and extract the provided Verilog hardware description language source files to that directory. All Verilog files have extension ".v". The top-level design file that instantiates the 9 tiles is system.v. Testbench file Testbench.v is used only for simulation; it loads hex files into the instruction and data memory of each processor and then monitors network traffic during execution. One processor is given a simple program that sends a few packets to different destinations, and the other eight processors are given programs that do not send traffic. You don't need to know the details of the processor for this lab, but you are encouraged to explore the design to know what you are implementing. The display statements in the testbench cause the simulator to print out all the routers that the packets pass through as they move through the network.

- Invoke ModelSim from the directory where you've placed the source files

  `> vsim &`

- Create a new ModelSim project. Click `File->New->Project…` Type in project name and location.
- Add Verilog source file to project. Right click within the Project panel, `Add to project-> Existing files…` and select all the Verilog files provided with the lab materials.
- Compile the HDL files. Right Click within project panel and click `compile-> compile all`. Now, all Verilog files are compiled to simulation libraries. You can find these under 'work' in the Library panel.
- Load and simulate the design. Expand work in Library panel. Right click Testbench and click simulate. The design will be loaded and the design hierarchy tree is shown in the new window.
- You can add waveforms by right clicking on a particular module or signal as desired. In this lab it may not be necessary to view signals in the waveform window because the output printed in the console window will report the movement of the packets.
- Run the simulation by clicking the run all button in the menu. The hex files should be in the current working directory or ModelSim will not find them.

SUBMIT: A screenshot of the console window showing the simulation outputs that print.
SUBMIT: Describe any difficulties encountered when simulating the design, and how you resolved them.

Part B: Synthesis

As you've done in the prior lab, synthesize the design to the library of standard cells. This will generate a mapped Verilog file that can be used with Encounter for physical design. In an NoC, a single tile would be designed, and then instantiated multiple times to create the overall mesh. You will follow that same approach here, and therefore will only synthesize one tile (step 3). You will then run a post-synthesis simulation (step 4) that instantiates the synthesized tile within the 3x3 mesh to check that it works correctly.

Step 3: Synthesize design [25 pts]

Use Design Compiler, with the tcl scripts from lab 4, to synthesize a single tile. You should follow here the instructions from lab 4, but will have to modify the scripts because the design has different module names, files, and so forth. Make sure you also copy the .db files from lab 4 to your working directory. We intentionally don't tell you the exact text changes to make to the tcl files; this is to encourage you to look through the scripts and to think about what each line may be doing. VLSI design engineers in industry become skilled in writing and modifying tcl scripts for these design automation tools.

- Change the top level module name in `setup.tcl` (remember: synthesize a tile, not the whole mesh)
- Change `read.tcl` so that all the Verilog files are read.
- You may wish to set the clock period in `Constraints.tcl`. This is not required, but a loose clock period can make for an easier synthesis job for design compiler. 10ns is reasonable. Synthesis will likely become slower when using a clock period below 5ns and the area of the design may increase.
- Apply the following change to ensure that the post-synthesis tile is a single flattened module with no hierarchy. Removing hierarchy in this lab simplifies post-synthesis simulation since there will be only one post-synthesis module to deal with.
    - In `CompileAnalyze.tcl`, after line `compile -only_design_rule -incremental` add new line `ungroup -all -flatten`
- Then run synthesis. If everything works correctly, a post-synthesis Verilog netlist will be created for the tile. Examine the reports and/or the netlist of the synthesized tile to answer the questions below.

SUBMIT: Describe precisely all the changes you made to tcl scripts and the purpose of each change
SUBMIT: Approximately how many cell instances (gates and flops) are in the synthesized design?
SUBMIT: What is the total cell area of the tile? Explain how you obtained the answer
SUBMIT: What is the maximum clock frequency at which your design will operate? Explain.
SUBMIT: How long is the critical path, in terms of number of logic gates? Explain how you found the answer

Step 4: Post-Synthesis Simulation [20 pts]

Now you will simulate the post-synthesis tile, which is a Verilog file that uses the cells from your library. The synthesized design has a large number of cells and nets that can make post-synthesis simulation very slow. To increase the speed of simulation, you will simulate the system with only one tile in the mesh replaced by its post-synthesis Verilog netlist. Create a modified `system.v` that instantiates the post-synthesis tile at position (1,2) on the east edge of the mesh. The other eight tiles are unchanged from the original `system.v` and will still use their pre-synthesis Verilog representations. Simulate this network using ModelSim and check the output in the console to see whether packets are routed through the network in the same way as they were in pre-synthesis simulation. When compiling the post-synthesis Tile, ModelSim will spend 10s of minutes print warnings about missing port connections which can be ignored; use this time to take a break, or to start on Part C.

- Find the synthesized Verilog output file (likely named `Tile_final.v`) that is generated in the synthesis directory. The same name "Tile" is used for both the pre-synthesis tile module (in `Tile.v`) and the post-synthesis tile module (in `Tile_final.v`). Since you will instantiate both of these modules in your system, they must have different names. Change the module name of the post-synthesis tile in file `Tile_final.v`. This is a large file, so you may want to change it before adding to ModelSim; it will be slow to edit the file in ModelSim's GUI.
- Create a new version of `system.v` in which tile (1,2) instantiates the new post-synthesis tile with the name that you specified, while keeping the rest of the tile instances unchanged. This is a very small change, since you are only replacing the instantiation of one module.

- Simulate using ModelSim as in step 2. Observe the output in the console to see the traffic moving through the mesh, including through the post-synthesized tile in the mesh. If this matches the pre-synthesis simulation, then it indicates that your synthesized design works correctly.

SUBMIT: Description of the modification you made to the system.v file.
SUBMIT: A screenshot of the console window showing the simulation results. Analysis of correctness.

Part C: Place and Route

To create the physical implementation of the tile, you must now perform place-and-route of the synthesized tile using Cadence Encounter. If you were creating the overall mesh network, you would instantiate multiple copies of the tile layout and connect them by abutment as you've done in lab 3 for the accumulator. If doing so, you would want to constrain the placement of the pins so that the N,S,E,W connections of the tile would be aligned on the appropriate edges to connect to neighbor tiles by abutment, but we do not require that.

Step 5: Placement [20 pts]

Use the approach from lab 4 to perform placement on the synthesized tile, using an aspect ratio of 1.0 (square) and a core utilization of 0.75. The steps here match the steps from lab 4, so you should consult the lab 4 instructions for a reminder. You will have to make a few changes to `design.conf` to specify the (post-synthesis) netlist file, and to indicate the top-level module name, and to provide the utilization and aspect ratio. Show the layout in both floorplan view and physical view. For physical view, show the layout <u>with wiring and vias</u>, and then again <u>without wiring and vias</u>. Measure the layout dimensions in physical view using the ruler tool. Recall that the routing performed in this step is just a trial route. Detailed routing is the next step.

SUBMIT: The layout images (floorplan + two physical views). Include ruler annotation on the physical view.
SUBMIT: Dimensions of the design.
SUBMIT: Are the dimensions consistent with the area reported by Design Compiler? Explain any differences

Step 6: Detailed Routing [20 pts]

Now perform detailed routing. Follow the same approach is used in lab 4 with `nanoroute`. Check the log file after detailed routing to see how many metal layers are used, and how much total wire length is routed in each layer. Create a new physical view in which all wiring and vias are turned off, <u>except for the uppermost metal layer that is used in your design.</u> Zoom in to measure the width of metal wires in that layer. Also check the widths of the metal 2, metal 4, and metal 6 wires in the same way. Measure the size of the vias between layers.

SUBMIT: Number of metal layers are used. Explain which metal layers have the most routing and whether you think that is reasonable.
SUBMIT: Image with all wiring turned off except the uppermost metal that is used.
SUBMIT: Give the wire width in each of the four specified metal layers, and support your answer with images. Do the widths of the wires across layers match your expectations from lecture? Explain.
SUBMIT: If you were producing a chip that is 1mm by 1mm (which is typically the smallest area that can be purchased on shared production runs), how large of a mesh network could you create on the chip? In other words, how many tiles could you fit?
SUBMIT [ECE658 only]: Submit a table showing the dimensions of vias between layers