Digital Design: Typical Digital Design Flow

Weijie Jiang, Ce Ma, Wim Dehaene

January 16, 2025

1 Goal of this exercise session

Previous exercise session gave a good understanding of the typical digital IC design flow depicted in Fig.1, mainly focusing on the backend part of the flow. Following the same steps and scripts, we provide for this session provides a RISC-V CPU design after place and route, since running the flow can take more than 3h. These results are the start for your further analysis.

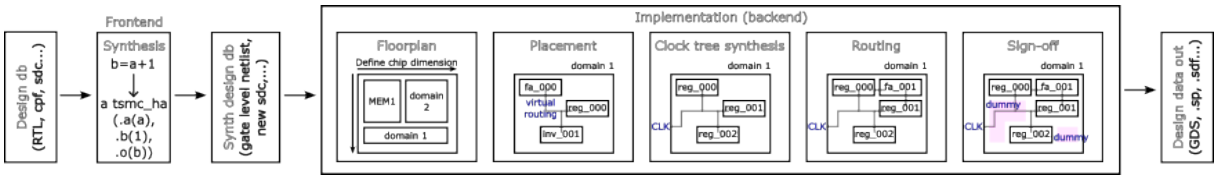


Figure 1: Diagram of a typical Digital Design Flow

The RISC-V RTL is the same as the used in Computer Architecture lab sessions, depicted in Fig. 2. We provide the RISC-V design using different requirements or specifications during synthesis and PnR. These specifications are parameters that as an engineer you need to feed as an input to the tools in the digital flow. Therefore, you will analyse how variations in those parameters affect power consumption, area and reliability.

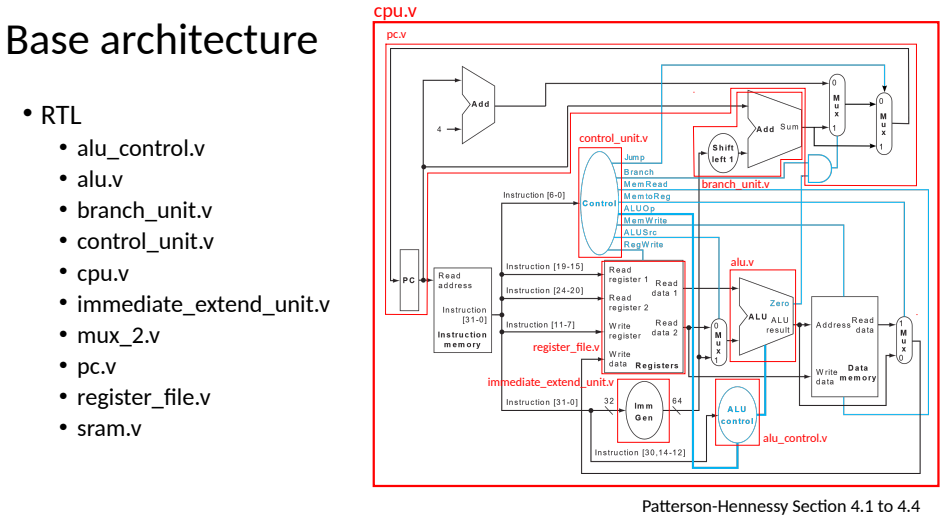


Figure 2: CPU base architecture [extracted from Computer Architecture manual]

1.1 Preparation steps

Copy the design files to your directory by

```
cd ~/Documents
cp -rP ~wdehaene/Public/DDIC/Project/RISCVCPU_flow RISCVCPU_flow
cd RISCVCPU_flow
```

2 Front end

2.1 Synthesis

Move to the associated directory:

```
cd 01_SYNTHESIS
```

The script used to run synthesis is scripts/genCompile.tcl, which loads the cpf file stored in DesignDataIn/cpf directory. This file specifies the standard cell flavours to be used during synthesis (ULVT, LVT, SVT, HVT...). In this exercise we are going to compare synthesizing the CPU using HVT+SVT cells and using SVT+LVT cells.

Uncomment in the ./make.sh file the FLAVOUR to be loaded. For instance, if using HVT cells, uncomment the "export FLAVOUR='HVT'" and comment the "export FLAVOUR='LVT'". Then, type the following commands.

```
source ./make.sh
# Wait until the design is loaded. In genus type:
report_power
report_timing
```

The timing report shows the critical path of the design. The *Pin* column tells the location of the signal, and the *Arrival* column tells the arrival time of that signal. The *Type* column tells the logic gates that the pin belongs to. You can tell from the suffix of the gate what flavor of the gate is. For instance, 'NAND2X8LVT' indicates a LVT (Low VT) cell, and 'NAND2X8SVT' indicate a SVT (Standard VT) cell.

EXERCISE: Execute the ./make.sh file and fill in Exercise 1 of the report.

3 Back end

We provide 5 different designs starting from the same synthesized RISC-V processor. Each design uses different input constraints, which as a designer you need to provide. have a look to the ./make.sh file. It defines first the parameters NB_METALS and TEST which specify the design to be loaded. Then, innovus executes the scripts/innoLoad.tcl file. Look at the top lines of this file. It loads the CPU design after signoff..

3.1 Chill Constraints

One of the main files you need to provide in a digital flow contains the timing information. It includes everything from the clock period and clock pins name to the restrictions on latency, skew and transitions delays of the clock and data ports. These files are stored in DesignDataIn/sdc and DesignDataIn/cts. This design loads the my_ctsSpec_ccopt_chill.tcl file, which constraints the max clock transition to 0.25ns and the skew to 0.4ns.

Go to 03_PnR folder. Open the ./make.sh file and uncomment the following lines:

```
export NB_METALS=4MetalLayers
export TEST=chillConstraint_NoDerate
innovus -overwrite -init scripts/innoLoad.tcl -log logs/innoLoad.log ...
```

EXERCISE: Execute the ./make.sh file and fill in Exercise 2 of the report.

3.2 Tight Constraints

Imagine that we want more clock precision, meaning faster transition delays (from 0.25ns to 0.1ns) and smaller clock skew (from 0.4ns to 0.25ns) in our design. The updated constraints are now defined in DesignDataIn/cts/my_ctsSpec_ccopt_tight.tcl.

Think as an engineer!

- Which step(s) of the flow do you think that are going to be altered by the new constraints?
- Where is the clock pin? Is it optimal?
- How do you think that this change is going to impact the final design (power, gates in the critical path, clock tree complexity...)?

Go to 03_PnR folder. Open the ./make.sh file and uncomment the following lines:

```
export NB_METALS=4MetalLayers
export TEST=tightConstraint_NoDerate
innovus -overwrite -init scripts/innoLoad.tcl -log logs/innoLoad.log ...
```

EXERCISE: Execute the ./make.sh file and fill in Exercise 3 of the report.

3.3 Add Derates

Imagine that you are in charge of designing a CPU for the driver assistance system in the next Tesla car. Considering the real case occurred in 2016 (see link), Elon Musk demands perfection, no room for even the tiniest error. Therefore, you opt to be more conservative in your design, adding some timing margins to ensure that your chip will work. These margins are inserted in the so-called derates in the innoTimingDerate.tcl file in the DesignDataIn/misc folder. Concretely, you add a 12% of the clock period margin for setup and 7.5% for hold.

Think as an engineer!

- Which step(s) of the flow do you think that are going to be altered by the new constraints?
- How do you think that this change is going to impact the final design (power, gates in the critical path, clock tree complexity...)?

Go to 03_PnR folder. Open the ./make.sh file and uncomment the following lines:

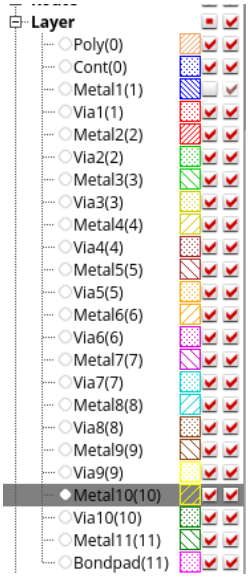
```
export NB_METALS=4MetalLayers
export TEST=tightConstraint_wDerate
innovus -overwrite -init scripts/innoLoad.tcl -log logs/innoLoad.log ...
```

EXERCISE: Execute the ./make.sh file and fill in Exercise 4 of the report.

3.4 Increase from 4 to 10 Metals for Routing

Until now, we used higher metal layers (Metal 6 and 7) in the power rings and grids and lower metals for routing (Metal 1 to 4). However, the metal stack that the foundry provides us includes up to 11 metal layers. Therefore, we update our flow and let the tool freely choose the layer for routing. This option is specified at the beginning of the floorplan script (innotFloorplan.tcl) with the command *setDesignMode -topRoutingLayer metalNameHere.*

Use the menu of Fig.3.4 to visualize and/or hide layers or type in the command line:




```
setLayerPreference Metal1 -isVisible 0 # not visible
setLayerPreference Metal10 -isVisible 1 # visible
```

Think as an engineer!

- Check the Library/lef/gsclib045_tech.lef file and compare the pitch, thickness and resistance of Metal1 and Metal10. What is the main difference between the different layers?
- Why do we use higher metals in the power rings?
- During routing, where and why would you use the lower metals? And the higher ones? Think in terms of cells density and wire length/delay.

Go to 03_PnR folder. Open the ./make.sh file and uncomment the following lines:

```
export NB_METALS=10MetalLayers
export TEST=tightConstraint_wDerate
innovus -overwrite -init scripts/innoLoad.tcl -log logs/innoLoad.log ...
```

EXERCISE: Execute the ./make.sh file and fill in Exercise 5 of the report.

3.5 Load Simulated/Real Activity

Once the layout is ready, we proceed to simulate the resultant netlist (some multiplications on a C++ program, see Computer Architecture exercise sessions). We verify the correctness operation and export the activity numbers. Then, we load the resultant activity information back in our flow to report more accurate power numbers. Edit the file scripts/innoLoad.tcl and uncomment the code lines below section LOAD ACTIVITY FILE.

Remember than during synthesis and PnR we estimated the input activity. Compute the new power numbers and compare with the power results of the previous exercise, before loading the real activity file of the cpu.

EXERCISE: Execute the ./make.sh file and fill in Exercise 6 of the report.

4 Conclusions

Congrats! You have now completed all basic steps for a typical digital flow. The most important takeaway of these sessions is the engineer's way of thinking. There is always a price to pay when you optimize your design in certain direction (speed/area/power/...). As engineers, our job is to decide how to meet the specs without overkill on some dimensions.