Digital Design: Typical Digital Design Flow

Weijie Jiang, Ce Ma, Wim Dehaene

March 3, 2025

1 Goal of this exercise session

This exercise session aims to help you build a good understanding of the typical digital IC design flow. Most focus is on the backend part of the flow. In the sessions, you will start with an ALU and a RISC-V CPU written in Verilog. You will go through their synthesis, place & route and verification steps till eventually generating the GDSII file.

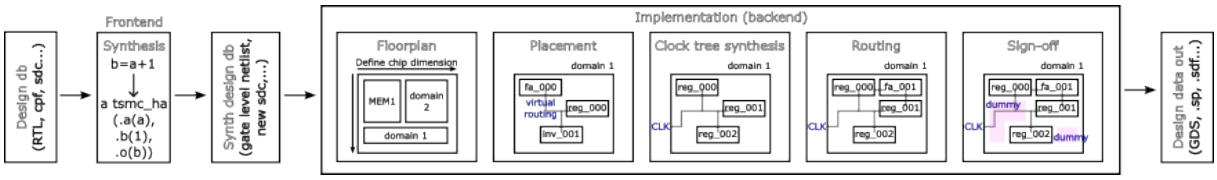

Figure 1: Diagram of a typical Digital Design Flow

- First session. Based on the ALU design to get familiar with the steps in Fig.1.
- Second session. Based on the RISC-V CPU design according to different requirements. You will use the same RTL as in Computer Architecture sessions. Due the increased complexity of the RISC-V block, running the flow can take up to 3h. Therefore, we provide the design after synthesis and after place and route. These results are then the start for your further analysis.

1.1 Preperation steps

Copy the design files to your directory by

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

The ALU module contains a logic unit, which does addition, subtraction, multiplication, division on different configuration. The outputs of the logic unit is sampled by flip flops. A mux selects between the bypassing input and flip flop stored value, and output to the other modules. In this session, we will think of this ALU as a complete chip, and apply the whole digital flow on it. For more details, refer to

```
./DesignDataIn/src/rtl/sparc_exu_alu.v
```

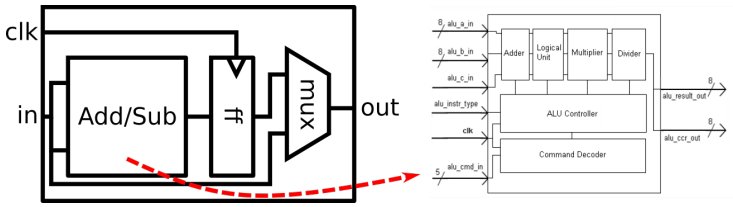

Figure 2: ALU diagram

2 Exercise: Front end

2.1 Synthesis

We start at Register Transfer Level (RTL). Once your design has been verified at RTL, the next step in the digital design flow is synthesis. Synthesis is a critical process where the RTL description is transformed into a gate-level representation using a synthesis tool. This transformation involves several key steps:

Elaboration: During elaboration, the synthesis tool constructs internal data structures and infers the hardware elements, such as registers, based on the RTL code. It also performs dead code elimination and semantic checks to ensure the RTL conforms to expected design principles. For example, if you declare a register in RTL with reg a, and the tool recognizes it as a register, it would be replaced by a specific instance, like CDN-flop a-reg(...), indicating a flip-flop is used in the hardware design.

Applying Constraints: The design is then subjected to user-defined timing and power constraints to ensure it meets specified performance and efficiency criteria. These constraints guide the synthesis tool in optimizing the design for speed, area, and power consumption.

Synthesis: In this phase, the tool translates RTL expressions into gate-level implementations using the digital cells available in the target technology library. For instance, an arithmetic operation like b = a + 1 in RTL would be implemented with hardware components, such as a simple gates, half adders or full adders, from the library. Post-synthesis, the tool verifies if the design meets the imposed timing and power constraints. If the design does not meet these criteria, further iterations of synthesis may be required to optimize and meet the constraints.

To start synthesis, simply type in the command line

```
cd 01_SYNTHESIS
source ./make.sh
```

The synthesis flow will start. Read the log on the terminal which contains quite a lot useful information. Let's walk through the log together to understand the synthesis in detail. Try to start by answering the following questions.

```
Info : Created nominal operating condition. [LBR-412]
     : Operating condition '_nominal_' was created for the PVT values...
```

Question: What are the difference on PVT sweeping for digital circuit and analog circuit? Hint: modeling process.

```
Info : Elaborating Design. [ELAB-1]
```

Question: Which design is being elaborated? What are done during elaboration?

```
Leakage Power: 0.0695220000 nW
Dynamic Power: 220.4366640000 nW
Total Power: 11.0878791000 nW
Leakage Power is contributing 0.596% to the Total Power.
```

Question: Why is the Total Power much lower than the Dynamic Power?

If everything goes well, you should have your synthesized design now. Go to the reports folder and check the available reports. First check the timing report. The report_timing command calculates the setup time of every path in the design. This includes the delay of in2reg, reg2out, reg2reg, in2out paths. By default, the delay of reg2reg paths will be analyzed and dumped to the report file. This is because in most cases we are more interested in the critical path inside the module. Try to answer the following questions:

- Explain the meaning of 'Start-point' and 'End-point'. Why is the 'Start-point' the 'CK' pin of a register while the 'End-point' the 'D' pin of a register?
- Did the synthesis result meet the timing? (Hint: check the 'Timing slack'.)
- What delay is missing? (Hint: do you know the information about wire now?)

You can also check the reports for power, area, etc.. For simplicity we will leave them to you.

3 Exercise: Back end

Now that we have the synthesized Verilog netlist of our ALU, we can now turn them into a layout and in the end a real silicon circuit! This part, with its many steps, is usually denoted as back-end design.

3.1 Floorplan

In the project home folder, type the following,

```
cd 02_FLOORPLAN
source ./make.sh
```

Innovus will start floor planning the ALU. After it's done, type

```
win
```

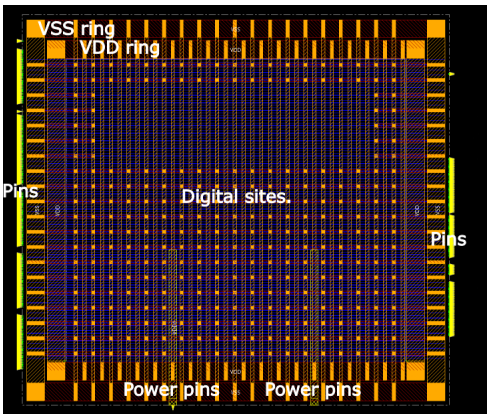

Figure 3: Explaining the floorplan results

The GUI will show with your floorplan. Press "f" to fit the design on your screen. Scroll up and down to zoom in and out. The floorplan contains a digital sites array, signal pins on the left and right, power pins at the bottom, and power rings surrounding the digital sites. Answer these questions:

- How big is this chip?
- Where are the power pins connected to?
- Which metals are used for signal pins, power pins, and power ring?
- Which other metal layers is used? Why are they there?

Unlike analog IC, where transistors can be placed anywhere, the digital gates are only allowed at specific places. The allowed locations are called sites. Zoom in to see these sites. Answer these questions:

- What's the spacing between two blue VSS metals?
- Will every digital gate have the same orientation?

Naturally, this floorplan does not come from nowhere. It's generated by the "innoFloorplan.tcl" file in the script folder. We encourage you to read this code to understand how the floorplan is derived (you don't have to understand codes in detail.)

3.2 Placement

We will now start placing the design.

```
cd 03_PnR
source ./placement.sh
```

Type "win" in the command line to open the GUI. Press "f" to fit the design on the screen. You should see your placed design (and pre-routed) Note that during placement, the nets are

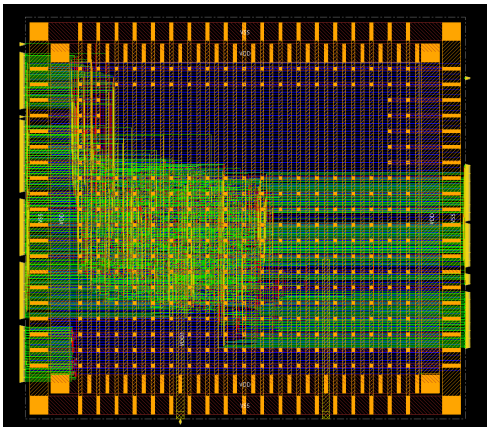

Figure 4: Placement result

pre-routed to roughly meet the timing requirement. Select any wire and press "q". You should see that the wire status as Unknown. The main differences between pre-routing and actual routing is that pre-routing does not strictly follow the To hide the metals and show the digital cells only, uncheck the layer box on the right. Look at the placement reuslts and answer the following questions.

- Where are the cells located? Are they closer to the input pins or the output pins or the power pins?

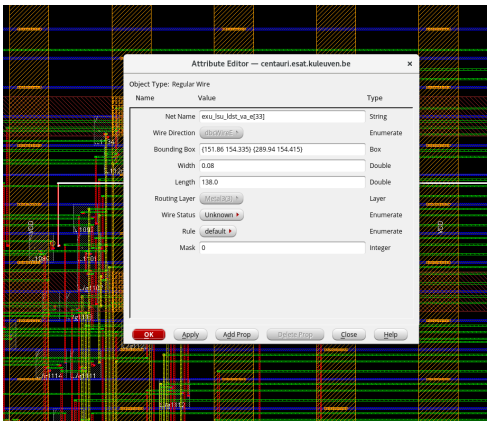

Figure 5: Status of routed nets during placement stage

- Are cells scattered or clustered? What are the advantages and disadvantages of a clustered placement result?
- What is missing from this placement result? Will you be able to pass DRC and with this? (hint, are there empty space on the floorplan?)

At this moment, you can start timing of the design using the report_timing command. This will show you the information on the critical path. For example, at the beginning you shall see things like

```
= Required Time 3.941
- Arrival Time 3.884
= Slack Time 0.057
```

Apparently, the Arrival Time is smaller than the Required Time, which means that the placement meets the timing requirement. Note that many "FE_RC_*" shows up the critical path. They are created by Innovus automatically for timing optimization.

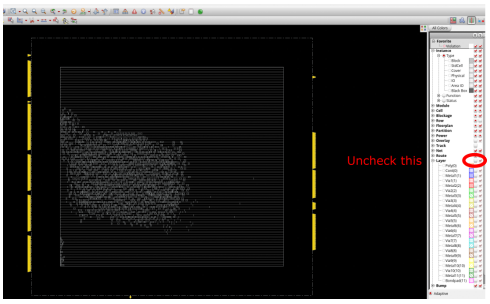

Figure 6: Hide all metal layers to view the placement result

3.3 Clock Tree Synthesis CTS

In a digital design, the clock is a critical net. It is also a difficult net because it is connected to every sequential element, with huge fan-out. As a consequence, the implementation of the clock net is completed in a separate step called clock tree synthesis. Before we continue this step, try to answer a few questions:

- What is clock skew? Is it always a bad thing?

- Is the delay from the source of the clock to the sequential elements important?
- Clock power is usually high, why? How can you reduce clock power?

To do CTS, source the "cts.sh". After it's completed, type "win" to open the GUI. Follow the steps to open the clock tree window. The clock starts from the top of the tree. The distance on

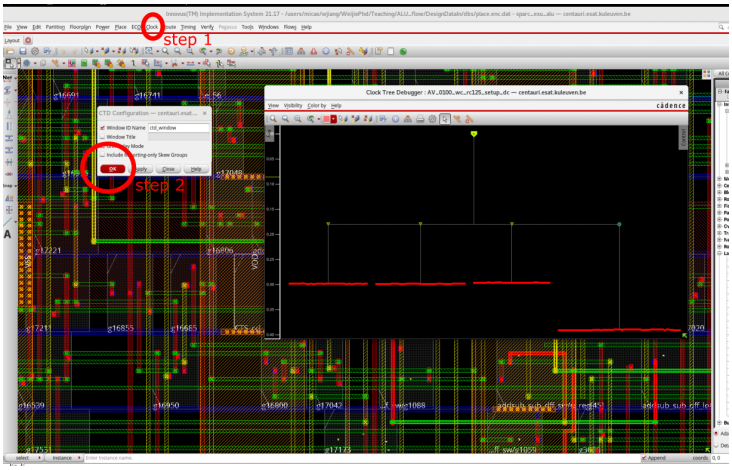

Figure 7: Steps to open the CTS result

the y axis represents the delay from a point to the next point. End points at the bottom are registers in the design. Examine the clock tree and answer a few questions:

- Are there buffers in the clock tree? What are they used for?
- Is the clock tree globally balanced? Will this be a problem? (hint: compare the delay from clock source to each flip flop.)

To inspect the clock tree visibly, drag your mouse and select all the items in the Clock Tree Debugger. After that you can move back to the main Innovus window. You should see that all clock related instances are selected and highlighted. Can you tell how clock tree is built by the Innovus? (Hint: think about letter 'H'.)

3.4 Routing

After CTS, we can start routing the design. Source the "route.sh". You can open the GUI to see the routed design. Open the timing report in the "route" folder of "timingReports" and answer a few questions.

- Did you meet the timing requirements?
- Which path is the critical path?
- Which timing is still missing in these reports? Why don't we do this at this step?

The generated reports are about setup timing. To check hold timing, type

```
timeDesign -hold -postRoute
```

Hold time report will be printed on the command line. Try to answer:

- Did you meet the hold timing requirement?
- Which path is the critical path?
- Is clock skew your friend or foe in terms of hold time violation?

3.5 Sign-off and optimizations

Now we are almost ready to sign off a full-blown chip. Before sign-off, a few optimization steps are generally applied on digital IC. During this step, we use more accurate parasitic extraction for timing check.s DRC and antenna errors are also fixed during this step. Source the "opt.sh". Open the reports and answer:

- Did you meet the hold timing requirement this time?
- What path violates the hold time check?
- How can you fix the existing hold time violation?

Before we finish this exercise, let's count the cells in the design. Open the design browser by clicking Tools/Design Browser. You can now see every existing cells in the design. Note that the StdCells contains all cells in the top level hierarchy, while standard cells in submodules are located in the Modules. We will end this exercise with a selection of all cells inserted by the Innovus. In the search bar, type 'FE*' and click enter. You should be able to filter out all those cells. Select all of them, right click the mouse, and select Attribute Editor. How many cells are inserted?

4 Conclusions

As said in the beginning, this exercise session aims to get you familiar with the flow. It's important to understand what each step is doing. You might also notice that the Front-end and floorplan are more explained than the rest. This is because during early stage of design, manual design efforts play a more important role than in late stage. For instance, in real digital circuit design, there could be thousands of hold-time violated paths during optimization stage. Such complexity basically prohibits manual fixing (and sometims is even beyond the tool's capability). Nevertheless, with a more appropriate floorplan, most of the hold-time violations could go away (think about a scan chain). In other words, tool manipulating is less important than problem analysis. In the next session, we will look at a more complicated design, a RISCV processor, and dive deep into the reports generated from each step.