Digital Design: exercise session 1

Weijie Jiang, Ce Ma, Wim Dehaene

March 3, 2025

1 General overview

Welcome to the exercise session of DDIC. We will show you how to do low level digital circuit design in four sessions, not just Verilog/VHDL coding but with actual transistors. These processes are practiced mostly by foundry or PhD students who need to setup his or her own digital library. They are also very helpful for digital/mixed signal designers because they are tightly related to parameters and concepts in front-end or back-end process (Front-end: Verilog/VHDL code writing, testing, and synthesizing... Back-end: floor plan, place and route, drc check...).

In this series of exercise sessions, we will optimize speed and power consumption of different building blocks of a simple 16-bit microprocessor, shown in Figure 1. We will focus on two memory elements, the registers and SRAM cells, and on a 16-bit adder, used in the ALU and address calculation unit. We will explore all different design levels: transistor level, gate level, architecture level, and system level. We will use a predictive 45 nm technology, as provided by Arizona State University (http://ptm.asu.edu/).

To do so, we will introduce the following industry-standard tools in the first 3 exercise sessions:

- **Cadence**: Graphical circuit design.
- **HSPICE**: Textbase simulator where circuits are described in a netlist format.
- **Ezwave**: Graphical waveform environment for displaying and analyzing signals.
- **Matlab**: Graphical interface to embed Hspice netlists and simplify coding and interpretation of the simulations.

The fourth session is somewhat more theoretical. It contains some insightful questions which can help to prepare for the exam.

2 Introduction to the session

In this session we will get familiar with a typical industrial custom gate design. It includes their design, simulation, results displaying and characterization. To do so, we are going to use two main tools:

- Cadence
- Hspice

We will start designing and simulating a simple inverter in a familiar way, the analog way, using **Cadence's graphical user interface** (gui). We will see how in the background Cadence translates your circuit drawing and simulation parameters into a text file using a netlist language.

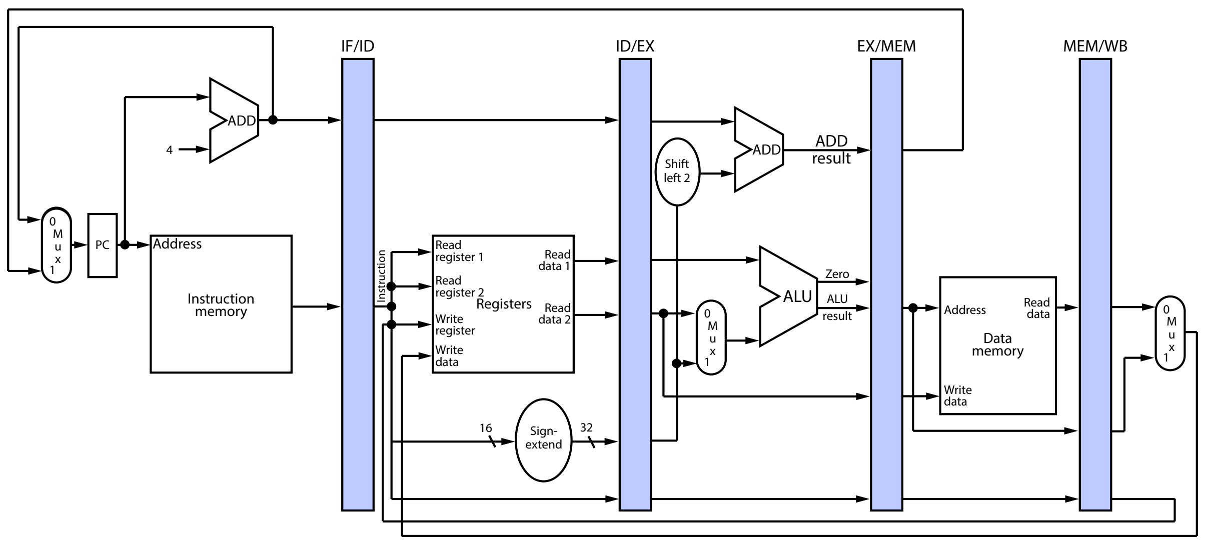


Figure 1: 5-stage pipelined microprocessor architecture

Then, we will repeat the inverter simulation using Hspice. **Any simulator gui that you will typically use, runs in the background a textbased simulator (like Hspice) to describe the circuit**. Therefore, working directly in Hspice allows to first, remove the abstraction of any gui. And second, from a research point of view, it permits digging into the lowest design level and push the design to the limits. The text format of the circuit is known as a **netlist**. A netlist, is the starting point of any industrial digital flow. We will plot the results using **ezwave** graphical interface, although any other is valid. We will work directly from the command line.

3 Environment setup

1. Boot/login in linux
2. Open a terminal and type the following lines:

```
mkdir ~/Documents/DigitalDesign
cd ~/Documents/DigitalDesign/
cp -r ~wdehaene/Public/DDIC/Project/doc .
cp -r ~wdehaene/Public/DDIC/Project/Resources .
cp -r ~wdehaene/Public/DDIC/Project/auxiliaryMatlabScripts .
cp -rP ~wdehaene/Public/DDIC/Project/DD_Es1 .
cd DD_Es1/
```

4 Exercise 1: Cadence

4.1 Environment setup

1. Go to the Cadence folder in DD_ES1. In the terminal type:

   ```
   cd ~/Documents/DigitalDesign/DD_Es1/Cadence
   ```

2. Run the setup script to copy or link all necessary files for you to design and simulate in 45nm process.

   ```
   sh setupGPDK45.sh
   ```

3. Launch Virtuoso:

   ```
   sh launchVirtuoso.sh
   ```

4.2 Exercise: Design a minimum size inverter

Use Cadence's Virtuoso tool to design a minimum size inverter and characterize it in terms of speed and power. To do so, follow the steps described in the Tutorial_Cadence.pdf.

Working with Virtuoso and ADE is straightforward. However, it involves a lot of clicking and GUI operations, which is inefficient for large scale digital circuits. Meanwhile, many analog features are not used by digital designers. To do more efficient design, we will move to more scripted version of this exercise.

5 Exercise 2: Hspice

5.1 Environment setup

1. In the terminal type:

```
cd ~/Documents/DigitalDesign/DD_Es1/spice
source setup.rc
```

5.2 Exercise: Design a minimum size inverter

For this exercise we provide the design of the same minimum size inverter of section 4 but with Hspice. You will find it in **"simpleInverter.sp"** file. Open it and read simultaneously **Tutorial_Spice.pdf** to understand how the circuit, simulation and measurements are defined. Then, run the simulation to characterize it in terms of speed and power. Finally, use a gui to plot the signals. Below are a series of commands to achieve this:

```
# run simulation with hspice
hspice -i simpleInverter.sp -o spicefiles simpleInverter.sp
# view results in ezwave
ezwave spicefiles/simpleInverter.tr0
```

Now, you can check the log output for errors or warnings in the **spicefiles/simpleInverter.lis** file and a simulation output is created in the binary **spicefiles/simpleInverter.tr** file. If everything went well, follow the gui and main steps to plot and analyze the desired signals as shown in figures 2 and 3. The measurements are stored in **spicefiles/simpleInverter.mt** file.

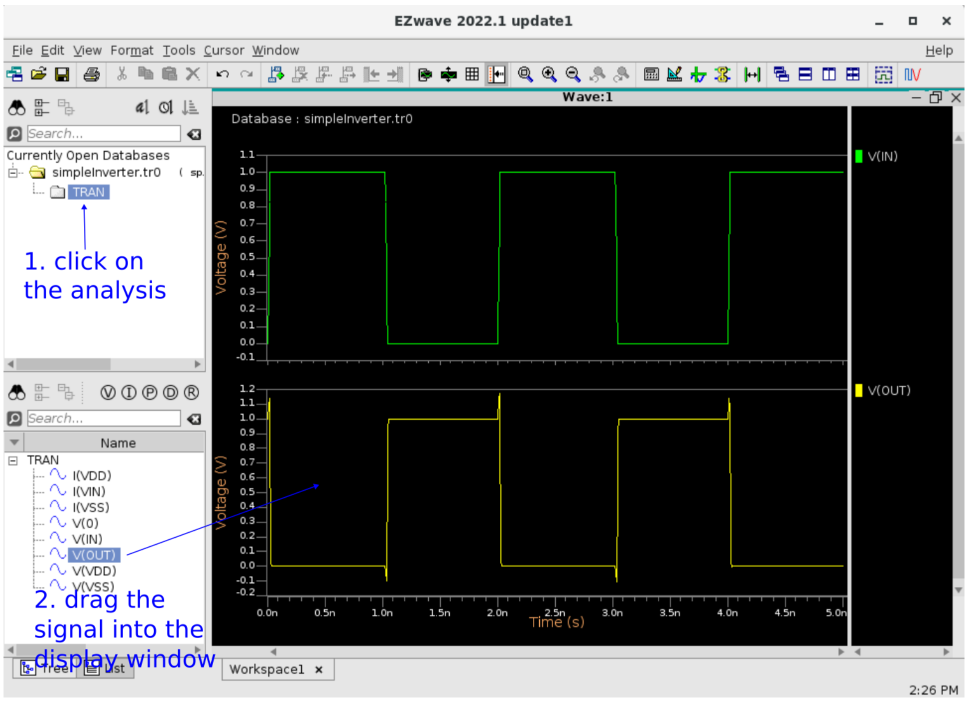
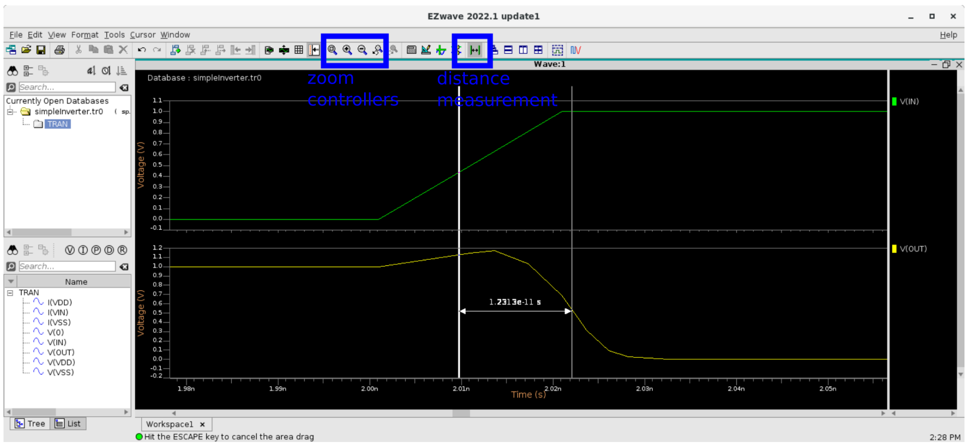


Figure 2: Ezwave graphical user interface. Inverter's input and output display.



Hit the ESCAPE key to cancel the area drag

Figure 3: Ezwave graphical user interface. Manual calculation of the inverter's delay.

6 Exercise 3: Experiments

Here you should have an inverter and should know how to simulate it, see the waveforms and measure delays and energy.

1.027e-11

1. Write down the delay of this inverter, which corresponds to the intrinsic delay (tp0).
2. Add a load capacitor of 10 fF to the output and re-characterize the inverter in terms of speed. 1.101e-10
3. Scale the inverter to 2x the size using the multfac parameter. What changes to the output behavior? Why does this happen?
   6.134e-11, doubled width -> doubled Ids -> halved output resistance -> halfed time constance
4. Set the inverter back to minimal size and remove the load capacitor. Add another inverter as a load. Is 10 fF a big or a small capacitance compared to the minimal inverter load? 2.846e-11
5. Scale the load inverter without changing the subcircuit. Make the size 16x a minimal inverter and observe the output. 1.417e-10
6. Tweak the inverter so that the pMOS is 4 times bigger than the nMOS. Is there a difference between the delay of positive edges and negative edges? Return to the standard 2/1 values.
   2* 1.417e-10 1.409e-10 4* neg 1.295e-10 rise 2.045e-10
7. Add again the load to 10fF. Make a rough estimate of the energy consumption by hand based on the nodal capacitance and compare both results. The values should be very close to each other. (The nodal capacitance table can be calculated by adding ".options captab" to the spicefile).
8. What happens to delay and energy if you lower the power supply?
9. Put the supply back to 1V. Calculate the leakage power for a single gate by setting a dc voltage on the input (uncomment and modify line 39). Can you find two different values of the leakage power? Why do they differ?