Design of Digital Integrated Systems: exercise session 1

Florian De Roose

October 7, 2015

1 General introduction

In this series of four exercise sessions, we will implement and optimise a 16-bit adder on transistor level. We will use a predictive 45 nm technology, as provided by Arizona State University (http://ptm.asu.edu/). We will use Hspice, a textbased simulator, to verify the behaviour of the ports, etc. We will use several MATLAB scripts to simplify the coding and interpretation of the simulations. The final deliverables will be a short report and some spicefiles.

If you have any questions, try using the manuals for the different tools and the book of Rabaey for the theory. If that doesn't help, don't hesitate to ask during the sessions or to send me an e-mail on florian.deroose@esat.kuleuven.be.

2 Introduction to session 1

In this session we will try to do the following things:

- Set up the MATLAB/Spice environment
- Explore the possibilities of the provided tools
- Optimise a buffer chain
- Design a switchable inverter

2.1 Tools

We will use several software tools to design the adder and to verify the design. Here follows a short discription of the most important ones. Small note: all double quotation marks (") are used as to indicate where the code starts and stops. It is not a part of the code.

2.1.1 mat2spice

mat2spice is a MATLAB-based tool that converts an m2s-file into an actual spice file. m2s-files are a hybrid of both MATLAB and spice: all normal text is considered spice traditional text, but by using escape symbols like $, MATLAB code can be included. Perhaps an example is easiest to demonstrate. This piece of code

```
.TRAN 10n
$ for i=1:2
Xnot$i n$i n$i+1 vdd vss MYNOT multfac=$< 2^i >$
$ end
```

will be converted to

```
.TRAN 10n

Xnot1 n1 n2 vdd vss MYNOT multfac=2
Xnot2 n2 n3 vdd vss MYNOT multfac=4
```

By using iterations in MATLAB the amount of m2s code required for a certain digital circuit can be vastly reduced. The tool works by taking the code and copying it in a new file.

- Lines without special escape symbols will simply be pasted like the ".TRAN 10n" statement.
- Lines that start with dollar-space will be considered MATLAB and executed as such. This can include the calculation of values (e.g. "$ ResVal = 1/2/Pi/CapVal/fcutoff;"), looping control, importing data or all other MATLAB statements.
- Inline evaluation of MATLAB values can be done in two ways. The shorthand notation starts with a dollar and no spaces. All consecutive code until the next space will be considered MATLAB code. Code that follows that space will again simply be pasted in spice. An example is the "n$i+1" statement: when i = 1 in the loop this transforms to "n2". Alternatively, if you want to include spaces in the command an inline MATLAB environment can be created by opening with "$<" and closing with ">$".

These are just the highlights of the tool. More details can be found in the documentation of the mat2spice tool, which can be found at "*installdir*/Resources/mat2spice/docs/index.html".

2.1.2 Hspice

Hspice is the spicesimulator we will use. It accepts the regular spice files, with regular spice syntax. The syntax could be deduced from the examples provided or found online. Hspice has an excellent manual, which can be found by typing in the terminal:

```
source ~micasusr/design/scripts/hspice.rc && hspice -help
```

2.1.3 Hspice toolbox

The Hspice toolbox is a set of MATLAB functions which can interpret the output of Hspice. The most important functions are:

- "x = loadsig(*file*)" imports a spice output file to a struct x. This struct can be used further by the toolbox.
- "y = evalsig(x, '*name*')" loads the signal with spicename "*name*" into the variable y. x is the struct generated by the loadsig function. The time axis of a transient analysis has "TIME" as name and the frequency in an ac analysis has "FREQ" as name.

- "plotsig(x, '*name*')" plots the signal with spicename "*name*". More options for the axis are available.

More information about the toolbox can be found in the manual, which can be opened by typing in the terminal:

```
evince installdir/Resources/HSpiceToolbox/hspice_toolbox.pdf
```

3 Setup of the tools

Boot in linux. Start by getting the required files from Toledo: DDIS1.zip

Extract this folder in a logical place, like for example the '~/Master2/DDIS' folder. This can by done by opening a terminal and writing:

```
mkdir ~/Master2
cd ~/Master2
mkdir DDIS
cd DDIS
cp ~/Downloads/DDIS1.zip ./DDIS1.zip
unzip DDIS1.zip -d ./
```

Do not use spaces in the name of the folders! The provided scripts will not work if you use spaces.

Once unzipped, you will notice the different files and folders:

- m2sfiles: contains all mat2spice files. In order for the scripts to work, the .m2s files should always be saved in this folder.
- Doc: contains this document
- Resources: contains all the externally provided MATLAB scripts (mat2spice and the Hspice toolbox) and the model files for the 45 nm technology. No files should be added or removed, the folder should be left as is.
- Spicefiles: contains the actual spice files and the generated spice outputs. Normally, no self-written files should be saved here, only the output of mat2spice and Hspice. Initially, this folder should be empty.
- auxiliaryMatlabScripts: contains some auxiliary scripts we added to simplify certain operations. More info later on.
- startup.m: a startupscript that initialises the MATLAB session by adding the correct folders to the MATLAB search path.
- Some demo scripts for later

All operations can be done from within MATLAB. In order to initialise all the tools, MATLAB should be opened by using the following command:

```
cd ~/Master2/DDIS/DDIS1/
matlab
```

This ensures that the startup script (startup.m) is automatically executed. This script adds hspice to the bash (so that the hspice-command can be used), sets the correct home folder and adds all the tools to the MATLAB search path.

Small sidenote: usually the standard matlab install at ESAT uses the shortcuts of EMACS, which means that "ctrl+C" and "ctrl+V" do not work. You can change this by going to “Preferences” in the main window, select the tab "Keyboard", go to "Shortcuts" and select the "Windows Default Set".

4 Exploration of the tools

4.1 Basic spice run

To simplify the use of the tools, a few scripts have been provided. In order for them to work, a variable called "inputfile" should be provided. As a first example, try the following code.

```
inputfile = 'simpleInverter';
runSpice;
```

runSpice calls three scripts.You can go and have a look at them:

convertToSpice is a script that converts the m2s-file to a regular spice file and stores it in the correct folder. If no supply voltage is defined, it will be set to 1 V.

execSpice is a script that calls hspice. This will do the actual simulation and store the result in "spicefiles". The most important files are the .tr0 (transient analysis) and .ac0 (ac analysis) files.

loadHSpiceResults is a script to import these outputs in the MATLAB workspace. The transient analysis will be stored in "transientsim" and the ac analysis will be stored in "acsim".

To view the waveforms of this simulation, type

```
plotsig(transientsim,'in;out')
```

As you had guessed from the title, the behaviour is that of an inverter.

4.1.1 Experiments

Have a look at the file "simpleInverter.m2s" in the m2sfiles folder by typing

```
open m2sfiles/simpleInverter.m2s
```

in MATLAB. Have a look at what the code tries to say. Make a small drawing of the circuit represented in this file. Then run the script "simpleInverter.m" and study its behaviour. Notice how the delay is calculated and displayed in the MATLAB terminal. Here are a few experiments that you could do to get more familiar with the basic spice files and their outputs.

- Add a load capacitor of 10 fF to the output.
- Remove the load capacitor and add another inverter as a load. Is 10 fF a big or a small capacitance compared to the minimal inverter load?

- Scale the load inverter without changing the subcircuit. Make the size 16x a minimal inverter.
- 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.
- Can you calculate the energy used for a single switching event from the simulation? Can you make a rough estimate by hand based on the nodal capacitance? Hints:
  - The current of voltage sources can be extracted by using "I_", followed by the name of the source, all small letters (e.g. to get the current for the Vdd source, use "evalsig(transientsim, 'I_vdd')"). For voltages, only use small letters, even if you used capitals in the spice name. Spice does not make a distinction between nodes with and without capitals.
  - Integrating can be done using the "trapz" function. Use trapz(time,power) because the datapoints for the current are not uniformly sampled.
  - The function "getIndexNumber" returns the index of the element closest to the time in a time array. E.g. if the time array was [1,3,7,12,20,21] and the input time would be 8, it would return 3 because the element closest to 8 is 7, wich has index 3.
  - The nodal capacitance table can be calculated by adding ".options captab" to the spicefile.
- Calculate the leakage power for a single gate.
- What happens to delay and energy if you lower the power supply? Change the power supply by changing the "globals.supply" variable in MATLAB.

4.2 Introducing .vec files

Now we can simulate individual transistors and display their analog outputs using the tools. It would be useful to have a tool that better handles digital signals for our application. The .vec file can solve such a problem. It provides a list of input ports and adds a piecewise linear voltage input signal to them. Additionally, a list of output ports for which the value and timing is checked can be added. Have a look at the "simpleInverterVec.m2s" and "simpleInverterVec.vec" to see what has changed and how they work. Both have clear comments and more info about the commands can be found in the Hspice manual.

Once you have an idea of what you can define in a .vec file, run the "simpleInverterVec.m" file. Note how, in addition to the correctness, the delay of the signals is verified. In the provided example, the maximum delay for the output of the inverter was put at 100 ps. As you can see from the delay calculation, this requirement is not met.

Please note that the .vec file just defines voltage sources and checks the output for errors. This means that you can add 10 vectors of 1ns, but if you only simulate 5ns, only the first 5 vectors can be applied and only the outputs during the first 5ns can be checked. So make sure that you simulate long enough!

4.2.1 Experiments

- Try to increase the maximum allowed delay for the output so that no error messages are generated.

- Make a new subcircuit for a standard NAND gate and a NOR gate: first make a sketch on paper and then implement it. Try to match their driving capability by scaling the transistors appropriately. Compare the resulting input capacitance to that of an inverter.
- Instantiate a NAND and a NOR gate, and write the corresponding test vectors in the .vec file. Start with a very loose timing and verify the correct behaviour of the gate. Change the delay constraint so that the .vec test starts failing.

4.3 Using arrays of data

Using larger digital words efficiently will be important for the design of the 16-bit adder. In order to introduce the idea of busses, "inverterArray.m2s" and "inverterArray.vec" contain examples of how to make a larger bus, both in the .m2s file and in the .vec file. The files describe an array of 16 inverters which inverts the data on a bus in0 .. in15 and puts the result on a bus out0 .. out15.

Experiment 1: try to change the code to make an array of buffers instead of inverters. Check the output by modifying the expected output vectors.

Experiment 2: implement a bus of switchable inverter: if control is high, it should behave as an inverter, if control is low, it should behave as a buffer. Start from the switchableInverter code.

5 Design case

In this design case you should design a buffer chain for minimal delay & energy. The input signal comes from a minimum inverter (modeling a logic gate with unit size), which should be included in the delay and power calculations. The output is loaded with a 100x inverter, which should not be included in the delay and power calculations. Figure 1 provides a schematic for the design. To help you focus on the interesting part (the design), templates have been provided as "bufferChain" with extensions ".m", ".m2s" and ".vec". Design for the following specs:

- Minimum delay.
- Minimum switching energy with a maximum delay of 200 ps.

Your design parameters are the number of inverters, the size of the inverters and the power supply.

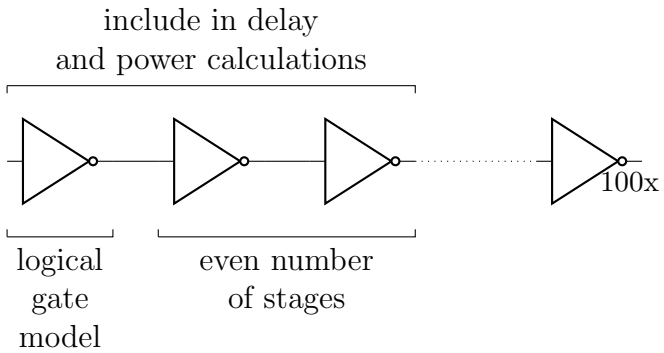

Figure 1: Schematic of the buffer chain