Digital Design: SPICE Tutorial

Weijie Jiang, Ce Ma, Wim Dehaene

January 16, 2025

1 Introduction to Hspice

HSPICE is a program that takes in a netlist (a simple text file), containing a circuit description and analysis options, and outputs the analysis it has done on that circuit. An HSPICE netlist typically has a .sp extension, for example simpleInverter.sp. You can create netlists in any text editor that you like. From the output files, you will need to access **.lis file to check for errors or warnings regarding the circuit connections**. File .tr0 are used by other programs (in this session Ezwave) to graphically plot the results of the analysis that HSPICE did.

2 File description

Open "simpleInverter.sp" file as reference. In any Hspice file we differentiate three main blocks:

- Simulation or analysis setup.
- Circuitry. It comprises the circuit to be simulated and additional subcircuits. These subcircuits can be instantiated from the main one, so we only need to describe them once.
- Measurements.

3 Simulation setup

Among others, this block specifies the analysis to be performed, the supply sources and input signals.

- To run a transient analysis on the circuit we use the **.TRAN** statement. As transient analysis is dependent on time, we need to specify a range of time.

  ```
  .TRAN START=start1 STOP=stop1
  ```

- To insert the power supplies for both VDD and GND rails we use the following statements:

  ```
  VDD VoltageName vss value
  VSS GroundName 0 0
  ```

- To simulate your circuits you will need some way to tell Hspice what is "exciting" or supplying electrical power to the circuit. To define a constant voltage source we simply use: *Vxxx xxx Vvalue 0* , where xxx is a name we choose and that must be unique. In this session, we use a pulse input, which is defined with the following statement:

  ```
  Vxxx xxx vss pulse initV peakV delay riseTime fallTime pulseWidth period
  ```

4 Circuitry

The way we describe the circuit topology is by giving each node (a node being the point connecting any two or more elements) in the circuit a unique name and then connecting circuit elements between these nodes. This is the reason we call these circuits format a netlist.

4.1 Elements and connection

Passive elements use this syntax: *name node node value.* The first letter of the device's **name** MUST match its function:

```
Rxxx for resistor: Rxxx node node value
Lxxx for inductor: Lxxx node node value
Cxxx for capacitor: Cxxx node node value
```

In the general case of transistors, we use below syntax.

```
Mxxx for mosfet: Mxxx nodeSource nodeGate nodeDrain nodeBulk mostype parameter_list
```

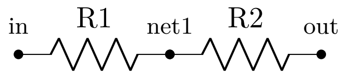
Additionally, a general purpose subcircuit can be made (see below). **Our transistors are wrapped into subcircuits.** Hence, to create transistors we use the following syntax:

```
for PMOS: Xxxx nodeSource nodeGate nodeDrain nodeBulk POSN w=width l=length
for MMOS: Xxxx nodeSource nodeGate nodeDrain nodeBulk MOSN w=width l=length
```

The **nodes** MUST acquire the name of the wire to which they are connected. If we want to connect two resistor in series, we need to use the same wire (or node name) and make sure the name is unique in the circuit:

```
R1 in net1 5M
R2 net1 out 10K
```

This is equivalent to the following netlist:



For the value units you can easily add next to the number f (femto), p (pico) , n (nano), u (micro), m (milli), k (kilo), meg (mega), g (giga) or t (tera).

4.2 Subcircuits design

To design a new subcircuit we use the following syntax:

```
.SUBCKT name node1 node2 ... nodeN [optional parameters]
   define circuitry here
.ENDS
```

For an example, look at *simpleInverter.sp* file. An inverter named MYNOT is created in a subcircuit, which nodes are named "input" and "output". Additionally, it includes a parameter named "multfact" with the default value 1. As you can see, this subcircuit is compound by a pmos and an nmos. The source of the nmos and the drain of the pmos are connected together with the wire named "output", which is a node of the subcircuit.

Subcircuits can be instantiated by starting with an X as the first letter of the name, following all node names, following the subcircuit name following optional paremeters:

```
   Xxxx nodea nodeb .... nodexx name param1=value1 param2=value2 ...
```

```
* Subcircuits
*-------------
.SUBCKT MYNOT input output vdd vss multfac='1'
xM1 output input vss vss MOSN w='multfac*120e-9' l='45e-9'
xM2 output input vdd vdd MOSP w='multfac*2*120e-9' l='45e-9'
.ENDS MYNOT
```

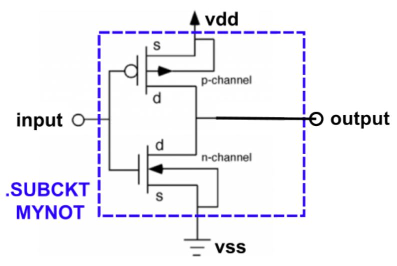


Figure 1: Hspice inverter design as a subcircuit.

Clearly, the node names do not need to be the same as in the subcircuit definition, and not all parameters need to be explicitly passed. If not passed, they retain the default value, specified in the subcircuit definition.

An instantiation of an MYNOT inverter, with the default parameter is shown as an example here:

```
Xdut in out vdd gnd MYNOT
```

4.3 Circuit to be simulated

This section contains the circuits that will be **simulated**. Each type of circuit element has its own command. You can either add transistors and passive elements as described above, or instantiate your own subcircuits. To do so, we use the following command (as an example, look again at *simpleInverter.sp* file):

```
Xname node1 node2 ... nodeN subcircuitName [optional parameters]
```

The circuit to be simulated in *simpleInverter.sp* is the gate described in MYNOT subcircuit, which input is the pulse voltage signal defined in section 3.

5 Measurements

The .MEASURE statement matches the last analysis command in the netlist before the .MEASURE statement (in our case the .tran statement). Fundamental measurement modes are:

- Rise, fall, and delay
- Find-when
- Equation evaluation
- Average, RMS, min, max, and peak-to-peak
- Integral evaluation
- Derivative evaluation
- Relative error

The results of the specified measurements are written into a **.mt0 file** as shown in figure 2. When a .MEASURE statement fails to execute, it writes "FAILED" in the output file. In *simpleInverter.sp*, we perform 4 measurements.

The first measurement is named delayNot1_FR and measures the delay from the second fall edge of voltage signal *in* to the second rise edge of voltage signal *out*. Similarly, delayNot1_RF measures from the rise edge to the fall edge.

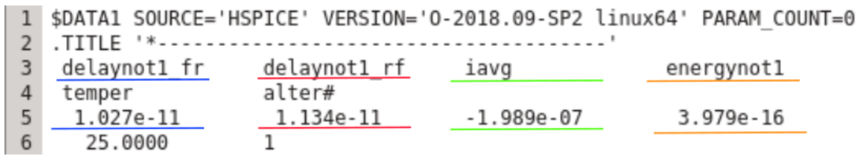
```
.MEASURE TRAN delayNot1_FR TRIG v(in) VAL='0.5*supply' FALL='edge'
+  TARG v(out) VAL='0.5*supply' RISE='edge'
```

The third measurement, measures the average current of the circuit in one clock cycle (2ns).

```
.MEASURE TRAN iavg avg i(vdd) from=0.5n to=2.5n
```

And finally, the average current is used to measure the energy per cycle of the circuit.

```
.MEASURE energyNot1 param='-supply*iavg*2n'
```



```
$DATA1 SOURCE='HSPICE' VERSION='O-2018.09-SP2 linux64' PARAM_COUNT=0
.TITLE '*-------------------------------------'
 delaynot1_fr     delaynot1_rf     iavg             energynot1
 temper           alter#
  1.027e-11        1.134e-11       -1.989e-07         3.979e-16
   25.0000        1
```

Figure 2: Hspice measurements results extracted from spicefiles/simpleInverter.sp.