University of Maryland Baltimore County

CMPE/ENEE 691
Hardware Security
Spring 2023

Tutorial: Synopsys HSPICE

HSPICE is an analog circuit simulator by Synopsis which can perform transient, steady state and frequency domain analysis. HSPICE simulates a circuit netlist file that has a '.sp' extension. The netlist file consists of SPICE statements in the form of a script.

A SPICE script has the following topology:

Title statement
Data statements
Control statements
Output statements
End statement

Some important notes about HSPICE scripting

- The data, control and output statements can be in any order.
- Comment statements begin with '*' and can be placed anywhere in the script.
- A statement can be continued in the following line by starting the continuation line with a '+'.
- The first line is ignored, so always start your netlist file with a title or a comment
- HSPICE syntax is not case sensitive.

Sources and Passive Devices

Here are the statements to describe common sources and devices

- Voltage Source: V*name* N+ N- <qualifier> value
- Current Source: I*name* N+ N- <qualifier> value
- Resistor: R*name* N+ N- value
- Capacitor: C*name* N+ N- value
- Inductor: L*name* N+ N- value

Where *name* is a number that identifies the element, e.g. V*name* could be V14, V15 etc. and R*name* could be R3 or R199 etc. N+ is the positive node and N- is the negative node to which the source connects. Anything appearing between <> is optional. For the sources, <qualifier> can be DC or AC, in which case value that follows is the magnitude. e.g.

Vdd dd gnd DC 2.5

is a DC voltage source called Vdd with magnitude of 2.5V, with its +ve node connected to node 'dd' and its -ve node connected to 'gnd'.

Note: 0, GND, GND!, and GROUND node names all refer to the global HSPICE ground.

Apart from DC and AC, a transient function can be used as the value, in which case no qualifier is needed.
2 Transient functions of interest are:

1) Pulse Train: PULSE(V1 V2 TD Tr Tf PW Period)
   e.g.
   Vname N1 N2 PULSE(V1 V2 TD Tr Tf PW Period)
   where
   V1: initial voltage
   V2: peak voltage

TD: initial delay time
Tr: rise time
Tf: fall time
PW: pulse width
Period: period
The pulse source is plotted in Figure 1.

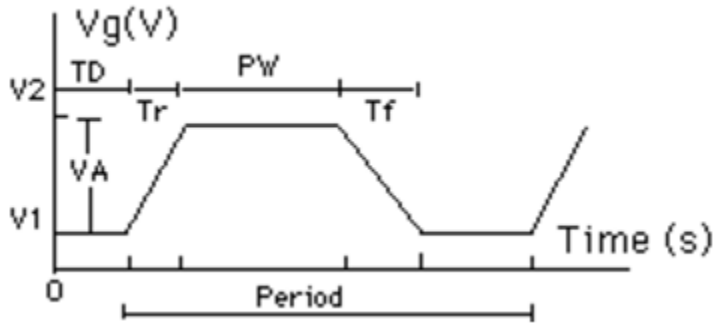

Figure 1: Pulse Voltage Source

2) Piecewise Linear: PWL(T1 V1 T2 V2 T3 V3 ...)
where (Ti Vi) specifies the value Vi of the source at time Ti.
e.g.
Vgpwl 1 2 PWL(0 0 10U 5 100U 5 110U 0)
The generated waveform is shown in Figure 2.

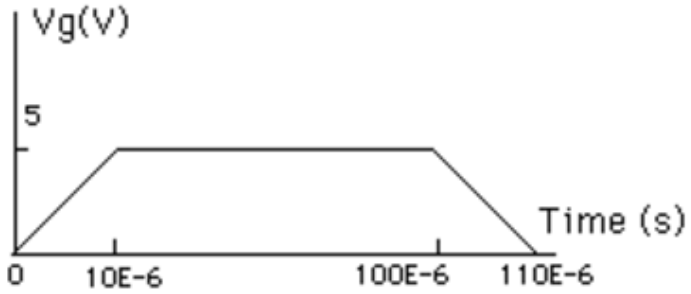

Figure 2: PWL source

Figure 3 shows an RC circuit.

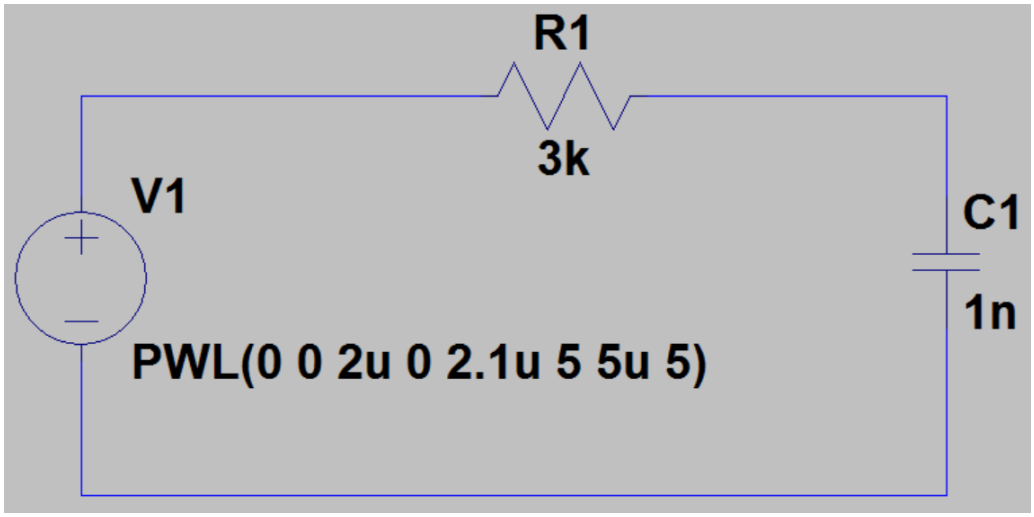

Figure 3: RC Circuit

The HSPICE script in Figure 4 describes and simulates the transient response of the RC circuit from Figure 3.

```
******** RC Circuit ***********

******** Sources *********
V1 N1 GND PWL(0 0 2u 0 2.1u 5 5u 5)

****** Componenets *******
R1 N1 N2 3k
C1 N2 GND 1n

******* Transient Analysis *********
.TRAN 500n 20u START=3n

******* Output ********
.PRINT TRAN V(N1) V(N2)
.PLOT TRAN V(N1) V(N2)

.END
```

Figure 4: RC Circuit Netlist File

The .TRAN command used in Figure 4 is used to start a transient analysis, using the following generic syntax:
.TRAN <step> <stop-time> START=<start-time>
i.e. The circuit will be simulated between 3n and 20u and the values to be printed and plotted will be sampled every 500n.

.PRINT specifies output variables to write to the output listings (.lis) file, using the following syntax:
.PRINT <analysis type> <variable1> …

To simplify parsing the output listings file, HSPICE prints a single 'x' in the first column, to indicate the beginning of the .PRINT output data, and a single 'y' in the first column to indicate the end of the .PRINT output data. The output written by the .PRINT command in the output (.lis) file is shown in Figure 5.

```
x

 time          voltage     voltage
              n1          n2
  500.00000n    0.          0.
    1.00000u    0.          0.
    1.50000u    0.          0.
    2.00000u   21.1758f   105.3524a
    2.50000u    5.0000    688.1766m
    3.00000u    5.0000      1.3474
    3.50000u    5.0000      1.8531
    4.00000u    5.0000      2.3140
    4.50000u    5.0000      2.7748
    5.00000u    5.0000      3.1569
    5.50000u    5.0000      3.4220
    6.00000u    5.0000      3.6576
    6.50000u    5.0000      3.8545
    7.00000u    5.0000      4.0440
    7.50000u    5.0000      4.1653
    8.00000u    5.0000      4.2866
    8.50000u    5.0000      4.4079
    9.00000u    5.0000      4.5203
    9.50000u    5.0000      4.5776
   10.00000u    5.0000      4.6350
   10.50000u    5.0000      4.6924
   11.00000u    5.0000      4.7498
   11.50000u    5.0000      4.8025
```

```
    12.00000u     5.0000      4.8261
    12.50000u     5.0000      4.8497
    13.00000u     5.0000      4.8733
    13.50000u     5.0000      4.8970
    14.00000u     5.0000      4.9187
    14.50000u     5.0000      4.9284
    15.00000u     5.0000      4.9381
    15.50000u     5.0000      4.9478
    16.00000u     5.0000      4.9576
    16.50000u     5.0000      4.9665
    17.00000u     5.0000      4.9705
    17.50000u     5.0000      4.9745
    18.00000u     5.0000      4.9785
    18.50000u     5.0000      4.9825
    19.00000u     5.0000      4.9863
    19.50000u     5.0000      4.9882
    20.00000u     5.0000      4.9902
y
```

Figure 5: Signals printed with .PRINT for the .TRAN analysis

The .PLOT command generates a low resolution printer plot in the output (.lis) file. The output file can then be opened with a wave viewer like "waveview" and the signals plotted as shown in Figure 6. Plotting using "waveview" will be described later in the tutorial.

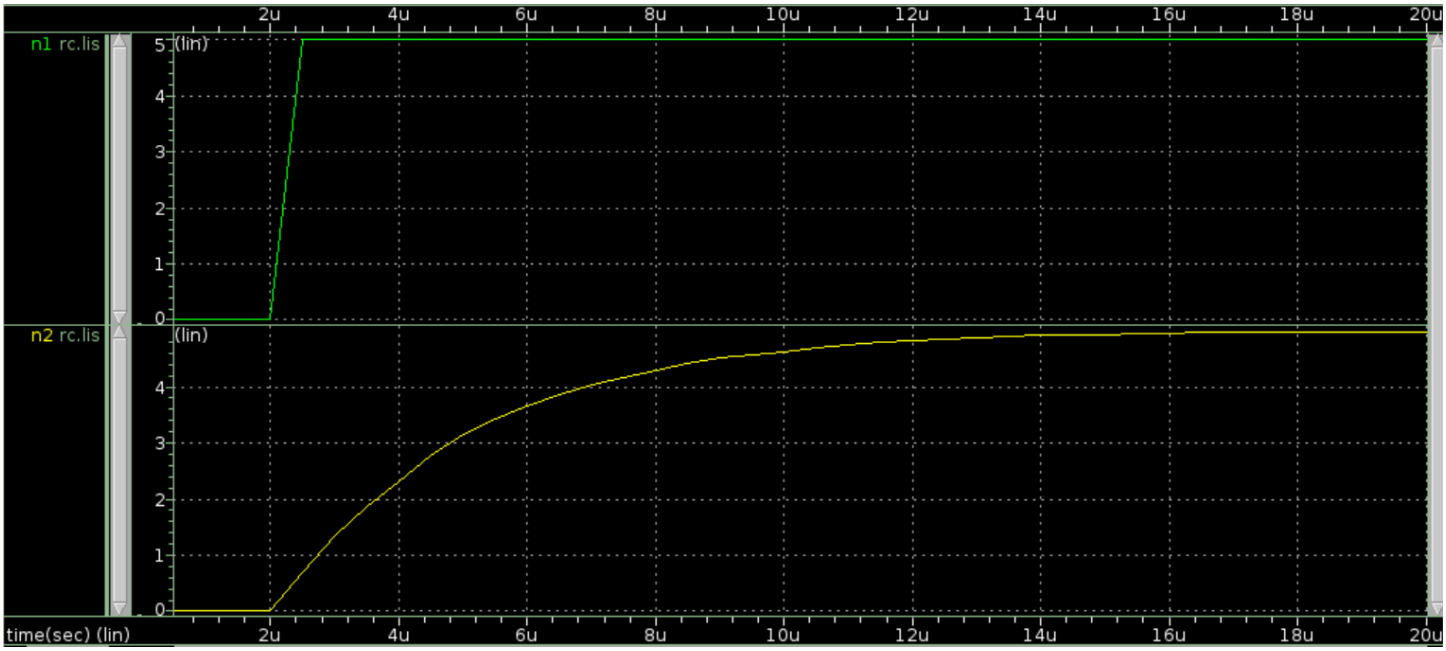

Figure 6: Waveform of RC Circuit Transient Response

MOSFETS

The generic statement to describe a MOSFET transistor is as follows:

M*name* Nd Ng Ns Nb model L=value W=value <AD=value AS=value PD=value PS=value>

After the name comes 4 connections: drain, gate, source, and body, followed by the model (a set of specifications describing the transistor. In the example provided, the models are described in the 'transistor_model' file which is called in the main script using the .INCLUDE command), followed by the transistor length and width, followed by optional values that you can ignore.

Figure 7 shows a schematic for a CMOS inverter.

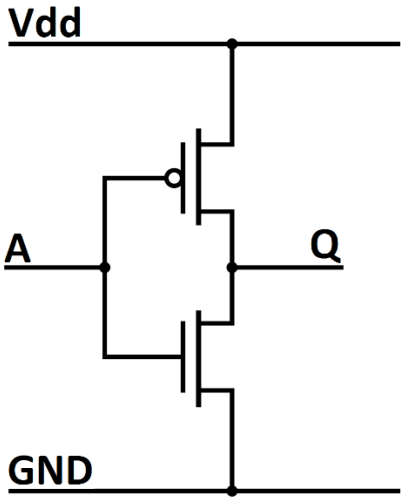

Figure 7: CMOS Inverter

Figure 8 shows the SPICE script for the inverter in Figure 7.

```
******** Inverter ***********
.INCLUDE './transistor_model'

.PARAM vth_pmos=-0.3021, vth_nmos=0.322, toxm_pmos=1.26e-009, toxm_nmos=1.14e-09

******** Sources *********
Vdd N1 GND 1.1
Vin A GND PULSE(0 1.1 1n 100p 100p 5n 10.2n)

****** Netlist *******
Mpmos Q A N1 N1 PMOS_VTL W=0.630000U L=50E-09
Mnmos Q A GND GND NMOS_VTL W=0.415000U L=50E-09

******* Transient Analysis *********
.TRAN 1n 25n START=0n

******* Output ********
.PRINT TRAN V(A) V(Q)
.PLOT TRAN V(A) V(Q)

.END
```

Figure 8: Inverter Netlist File

The output generated by the .PRINT command in the output (.lis) file is shown in Figure 9.

```
x

 time          voltage     voltage
               a           q
    0.           0.          1.1000
    1.00000n     0.          1.1000
    2.00000n     1.1000   -238.2061u
    3.00000n     1.1000    245.3837u
    4.00000n     1.1000   -270.2833u
    5.00000n     1.1000     13.7133u
    6.00000n     1.1000    382.7020u
    7.00000n     0.          1.0999
    8.00000n     0.          1.0999
    9.00000n     0.          1.0998
   10.00000n     0.          1.1000
```

```
    11.00000n     0.          1.1000
    12.00000n     1.1000     56.1294u
    13.00000n     1.1000     43.8452u
    14.00000n     1.1000     46.7282u
    15.00000n     1.1000     51.4339u
    16.00000n     1.1000     56.1396u
    17.00000n     0.          1.0999
    18.00000n     0.          1.1001
    19.00000n     0.          1.1000
    20.00000n     0.          1.0998
    21.00000n     0.          1.1000
    22.00000n     1.1000     42.3806u
    23.00000n     1.1000     62.2143u
    24.00000n     1.1000     55.7865u
    25.00000n     1.1000     21.6660u
y
```

Figure 9: Signals printed with .PRINT for the .TRAN analysis

The waveforms of the inverter simulation are shown in Figure 10. Plotting waveforms will be described later in the tutorial.

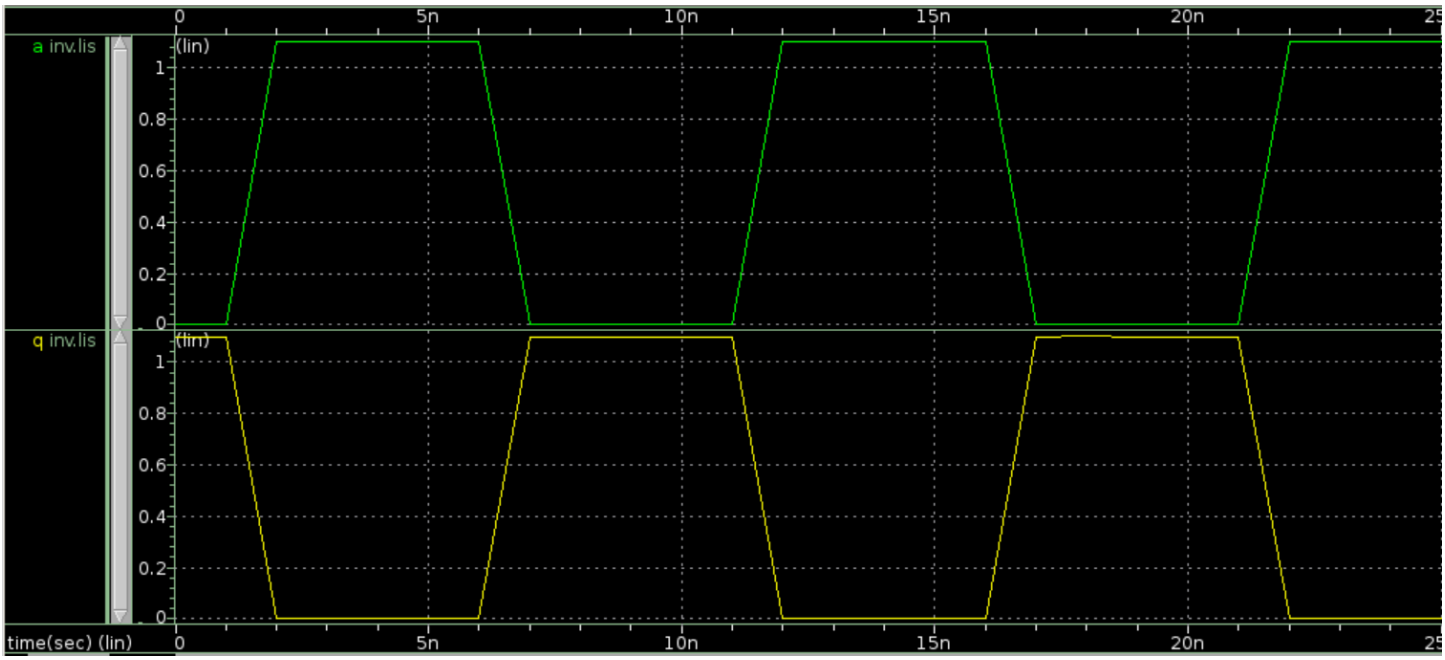

Figure 10: Waveform of the Inverter described in Figure 8

Subcircuits (.SUBCKT)

Subcircuits are used for commonly-used modules so that they can be instantiated in the netlist as needed. A subcircuit has the general form shown in Figure 11.

```
.SUBCKT <subcircuit name> <node 1> <node 2> …
<subcircuit netlist>
.ENDS
```

Figure 11: Generic Subcircuit Statement

For example, a 4-bit adder can be described by interconnecting 4 full adder subcircuit instances.

Figure 12 shows a schematic of a full adder.

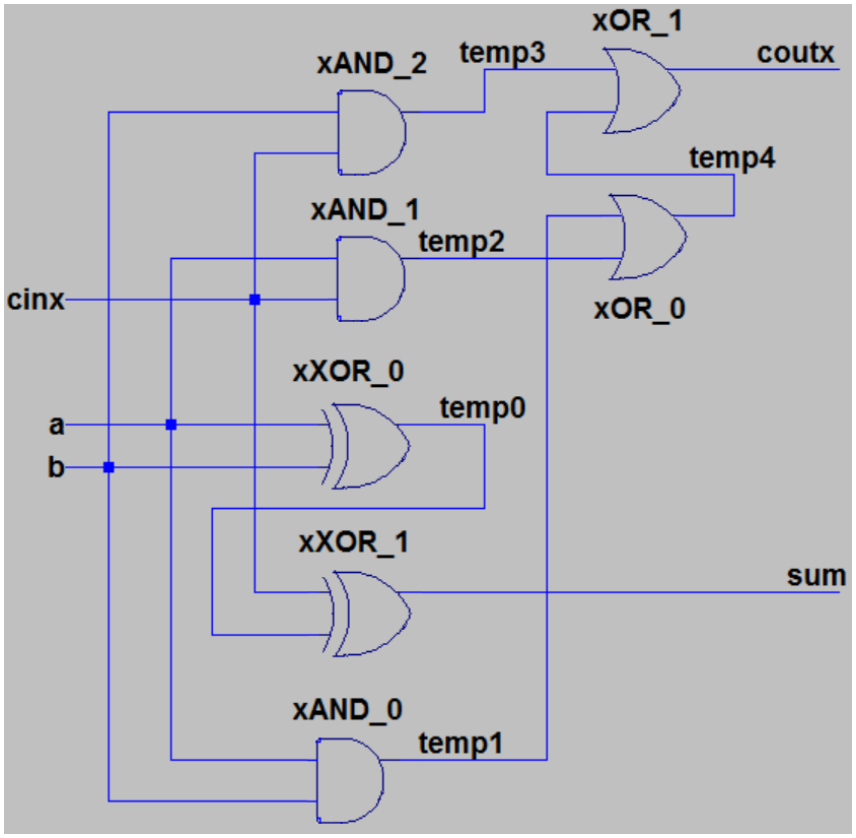

Figure 12: Full Adder Schematic

The subcircuit statement for a full adder is shown in Figure 13. Note that in Figure 13, the instantiated gates (XOR_X1, AND2_X1,OR2_X1) are described in the library file 'all_gates' which is called in the script using the .INCLUDE command, as will be shown later in Figure 16.

```
.SUBCKT ADDER_X1 a b cinx sum coutx VDDx GNDx
xXOR_0 a b temp0 VDDx GNDx XOR_X1
xXOR_1 cinx temp0 sum VDDx GNDx XOR_X1
xAND_0 a b temp1 VDDx GNDx AND2_X1
xAND_1 a cinx temp2 VDDx GNDx AND2_X1
xAND_2 b cinx temp3 VDDx GNDx AND2_X1
xOR_0 temp1 temp2 temp4 VDDx GNDx OR2_X1
xOR_1 temp3 temp4 coutxx VDDx GNDx OR2_X1
.ENDS
```

Figure 13: Full Adder Subcircuit Statement

A 4-bit adder constructed by interconnecting 4 full adder subcircuit instances is shown in Figure 14.

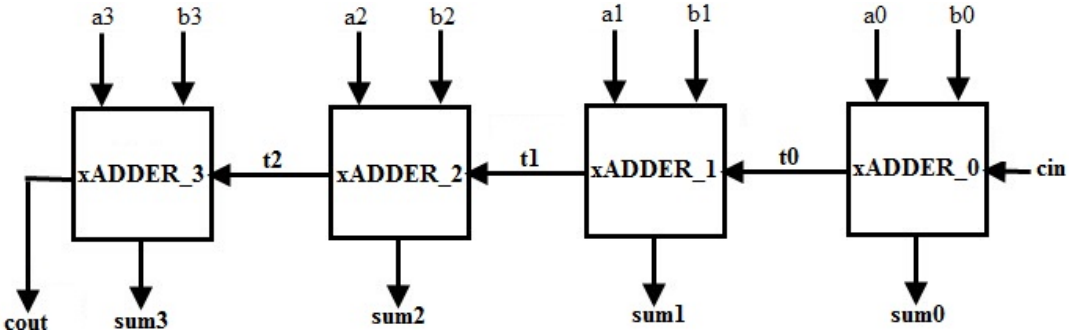

Figure 14: 4-bit Adder

The 4-bit Adder netlist made from 4 interconnected instantiations of the Full Adder Subcircuit is shown in Figure 15.

```
xADDER_0 a0 b0 cin sum0 t0 VDD GND ADDER_X1
xADDER_1 a1 b1 t0 sum1 t1 VDD GND ADDER_X1
xADDER_2 a2 b2 t1 sum2 t2 VDD GND ADDER_X1
xADDER_3 a3 b3 t2 sum3 cout VDD GND ADDER_X1
```

Figure 15: Full Adder Netlist

As can be seen in Figures 13 and 15, any instantiated modules (whether from subcircuits or from included library files) should be given names that start with 'X' or 'x'. An instantiation statement starts with the instance name, followed by the nodes in the same order as in the module description.

A complete netlist file for the full adder can be seen in Figure 16, with all control and output statements necessary to run the simulation and view the simulation results. Read the comments for a description of some of the commands.

```
******** 4-bit Adder ***********

******* Library Files *********
.INCLUDE './transistor_model'
.INCLUDE './ALL_GATES_HSPICE'

******* Simulation Control Options *********
.OPTION POST BRIEF PROBE ACCURATE INGOLD=2 MEASDGT=8

******* Temperature *********
***.TEMP sets the circuit temperature
.TEMP 45

******* Global Node *********
***.GLOBAL is used to define nodes and sources to enable their use in subckt nodes
.GLOBAL VDD

******* Parameters *********
***.PARAM is used to assign values to a parameters
.PARAM LMIN=50E-09, vth_pmos=-0.3021, vth_nmos=0.322, toxm_pmos=1.26e-009,
toxm_nmos=1.14e-09

******* Transient Analysis *********
.TRAN 2n 200n START=0n
.PRINT TRAN V(a0) V(a1) V(a2) V(a3) V(b0) V(b1) V(b2) V(b3) V(cin)
+ V(sum0) V(sum1) V(sum2) V(sum3) V(cout)
.MEASURE TRAN rise_difference TRIG V(b0) VAL=0.55 RISE=1 TARG V(sum0) VAL=0.55 RISE=2
.MEASURE TRAN fall_difference TRIG V(a0) VAL=0.55 FALL=1 TARG V(sum0) VAL=0.55 FALL=1
.MEASURE TRAN rise_time TRIG V(sum0) VAL=0.11 RISE=2 TARG V(sum0) VAL=0.99 RISE=2
.MEASURE TRAN fall_time TRIG V(sum0) VAL=0.99 FALL=1 TARG V(sum0) VAL=0.11 FALL=1

******** Sources ***********
Vsupply VDD 0 1.1
Va0 a0 GND PULSE(0 1.1 0n 100p 100p 40n 80.2n)
Va1 a1 GND PULSE(0 1.1 10n 100p 100p 40n 80.2n)
Va2 a2 GND PULSE(0 1.1 30n 100p 100p 40n 80.2n)
Va3 a3 GND PULSE(0 1.1 50n 100p 100p 40n 80.2n)
Vb0 b0 GND PULSE(0 1.1 60n 100p 100p 50n 100.2n)
Vb1 b1 GND PULSE(0 1.1 80n 100p 100p 50n 100.2n)
Vb2 b2 GND PULSE(0 1.1 100n 100p 100p 50n 100.2n)
Vb3 b3 GND PULSE(0 1.1 120n 100p 100p 50n 100.2n)
Vcin cin GND PWL(0n 0 110n 0 110.1n 1.1 160n 1.1)
```

```
******** Model *********
*** Subcircuit: Full Adder
.SUBCKT ADDER_X1 a b cinx sum coutx VDDx GNDx
xXOR_0 a b temp0 VDDx GNDx XOR_X1
xXOR_1 cinx temp0 sum VDDx GNDx XOR_X1
xAND_0 a b temp1 VDDx GNDx AND2_X1
xAND_1 a cinx temp2 VDDx GNDx AND2_X1
xAND_2 b cinx temp3 VDDx GNDx AND2_X1
xOR_0 temp1 temp2 temp4 VDDx GNDx OR2_X1
xOR_1 temp3 temp4 coutx VDDx GNDx OR2_X1
.ENDS
*** Netlist: 4-bit Adder
xADDER_0 a0 b0 cin sum0 t0 VDD GND ADDER_X1
xADDER_1 a1 b1 t0 sum1 t1 VDD GND ADDER_X1
xADDER_2 a2 b2 t1 sum2 t2 VDD GND ADDER_X1
xADDER_3 a3 b3 t2 sum3 cout VDD GND ADDER_X1

.END
```

Figure 16: Full Adder Netlist File

More Netlist Command Descriptions

.INCLUDE is used to include files like subcircuit libraries and specification files. e.g. in Figure 16, 'all_gates' contains transistor-level implementation of gate primitives (gate library) and 'transistor_model' contains transistor specifications (transistor library).

.OPTION is used to set simulation control options, with the following generic syntax:
.OPTION OPTION1 <OPTION2 OPTION3 …>
Table 1 describes the options in used in the Full Adder netlist of Figure 16.

Option	Description
POST	Stores simulation results for analysis by an interactive wave viewer.
BRIEF	Prevents printing to the output file until the .END statement in case a netlist or syntax error occurs.
PROBE	Gives a neater simulation output by excluding node voltages and source currents from the output leaving only variables specified in .PRINT and .PROBE
ACCURATE	Increases transient and AC simulation accuracy for a longer simulation time. Setting ACURATE is equivalent to setting RUNLVL to 5 or 6.
INGOLD	Specifies the printout data format, setting it to 2 makes the format exponential.
MEASDGT	Formats the .MEASURE statement output, can be between 1 and 10 to specify the decimal places. It is used in conjunction with INGOLD to control the output data format.

Table 1: Option Descriptions

When .PRINT is used with .OPTION POST, the plot data is written in the '.tr0' file and the numeirical values are written in the 'lis' file.
.PROBE writes the plot data in the '.tr0' file without writing any numerical data to the '.lis' file.
.PLOT writes the plot data to the '.lis' file.

.MEASURE is used to print many types of user defined specifications. The syntax for rise, fall and delay measurements is:
.MEASURE <analysis type> <variable name> <trig_spec> <targ_spec >
where <analysis-type> in our *Full Adder* example is TRAN
<variable name> is the user-defined variable name

<trig_spec> has the following syntax:

TRIG <signal> VAL=<value> <RISE or FALL>=<number of edge>

<targ_spec> has the same syntax as <trig_spec>

The following statement from Figure 16 can be used to measure the Rise Difference between 2 signals

```
.MEASURE TRAN rise_difference TRIG V(b0) VAL=0.55 RISE=1 TARG V(sum0) VAL=0.55 RISE=2
```

This statement measures the time difference from when V(b0) reaches its half value (0.55) on its first rising edge to when V(sum0) reaches its half value (0.55) on its 2nd rising edge (notice that the 2nd rising edge is used because the signal has a rising edge at time 0ns).

Fall Difference can be measured similarly

```
.MEASURE TRAN fall_difference TRIG V(a0) VAL=0.55 FALL=1 TARG V(sum0) VAL=0.55 FALL=1
```

Rise and Fall Time for a signal (sum3 from Figure 16) can also be measured in a similar manner, using the 10% and 90% values accordingly.

```
.MEASURE TRAN rise_time TRIG V(sum0) VAL=0.11 RISE=2 TARG V(sum0) VAL=0.99 RISE=2
.MEASURE TRAN fall_time TRIG V(sum0) VAL=0.99 FALL=1 TARG V(sum0) VAL=0.11 FALL=1
```

The outputs of the .MEASURE statements can be seen in the output listings file as shown in Figure 17.

```
 rise_difference= 4.56465976e-11  targ= 6.00956466e-08   trig= 6.00500000e-08
 fall_difference= 5.64084378e-11  targ= 4.02064084e-08   trig= 4.01500000e-08
 rise_time= 1.20731271e-11  targ= 6.01030953e-08   trig= 6.00910222e-08
 fall_time= 1.33315639e-11  targ= 4.02119037e-08   trig= 4.01985721e-08
```

Figure 17: The Output of .MEAURE in the '.lis' File

Process Variations

The script in Figure 18 adds process variations to the simulation by causing the Lmin, Vth and Tox for NMOS and PMOS transistors to vary according to a Gaussian distribution function (notice that in 'transisot_model', these parameters are assigned variables as their values, so they can be specified in the main script). To enable Monte Carlo sweeping, SWEEP MONTE = … is a added to the .TRAN statement as shown in Figure 18. Running the script will simulate 30 versions of the circuit, each with a varied set of the parameters Lmin, Vth and Tox, to mimic the process variations arising from the imperfect fabrication process. The results for each run are printed to the output '.lis' file headed with 'monte carlo index = x', where x is the number of the monte carlo run. Open the file with any text editor and search for the word 'monte', you will find the results for monte = 1, 2, 3 … 30, each with its corresponding output data generated from .PRINT and .MEAURE

```
******** 4-bit Adder ***********

******* Library Files *********
.INCLUDE './transistor_model'
.INCLUDE './ALL_GATES_HSPICE'

******* Simulation Control Options *********
.OPTION POST BRIEF PROBE ACCURATE INGOLD=2 MEASDGT=8

******* Temperature *********
***.TEMP sets the circuit temperature
.TEMP 45

******* Global Node *********
***.GLOBAL is used to define nodes and sources to enable their use in subckt nodes
.GLOBAL VDD

******* Parameters *********
***.PARAM is used to assign values to a parameters
```

```
*.PARAM LMIN=50E-09, vth_pmos=-0.3021, vth_nmos=0.322, toxm_pmos=1.26e-009,
toxm_nmos=1.14e-09
.PARAM LMIN=GAUSS(50E-09,0.1,3), vth_pmos=GAUSS(-0.3021,0.1,1),
vth_nmos=GAUSS(0.322,0.1,1), toxm_pmos=GAUSS(1.26E-009,0.01,1), toxm_nmos=GAUSS(1.14E-
09,0.01,1)

******* Transient Analysis *********
.TRAN 2n 200n START=0n SWEEP MONTE=30
.PRINT TRAN V(a0) V(a1) V(a2) V(a3) V(b0) V(b1) V(b2) V(b3) V(cin)
+ V(sum0) V(sum1) V(sum2) V(sum3) V(cout)
.MEASURE TRAN rise_difference TRIG V(b0) VAL=0.55 RISE=1 TARG V(sum0) VAL=0.55 RISE=2
.MEASURE TRAN fall_difference TRIG V(a0) VAL=0.55 FALL=1 TARG V(sum0) VAL=0.55 FALL=1
.MEASURE TRAN rise_time TRIG V(sum0) VAL=0.11 RISE=2 TARG V(sum0) VAL=0.99 RISE=2
.MEASURE TRAN fall_time TRIG V(sum0) VAL=0.99 FALL=1 TARG V(sum0) VAL=0.11 FALL=1

******** Sources ***********
Vsupply VDD 0 1.1
Va0 a0 GND PULSE(0 1.1 0n 100p 100p 40n 80.2n)
Va1 a1 GND PULSE(0 1.1 10n 100p 100p 40n 80.2n)
Va2 a2 GND PULSE(0 1.1 30n 100p 100p 40n 80.2n)
Va3 a3 GND PULSE(0 1.1 50n 100p 100p 40n 80.2n)
Vb0 b0 GND PULSE(0 1.1 60n 100p 100p 50n 100.2n)
Vb1 b1 GND PULSE(0 1.1 80n 100p 100p 50n 100.2n)
Vb2 b2 GND PULSE(0 1.1 100n 100p 100p 50n 100.2n)
Vb3 b3 GND PULSE(0 1.1 120n 100p 100p 50n 100.2n)
Vcin cin GND PWL(0n 0 110n 0 110.1n 1.1 160n 1.1)

******** Model *********
*** Subcircuit: Full Adder
.SUBCKT ADDER_X1 a b cinx sum coutx VDDx GNDx
xXOR_0 a b temp0 VDDx GNDx XOR_X1
xXOR_1 cinx temp0 sum VDDx GNDx XOR_X1
xAND_0 a b temp1 VDDx GNDx AND2_X1
xAND_1 a cinx temp2 VDDx GNDx AND2_X1
xAND_2 b cinx temp3 VDDx GNDx AND2_X1
xOR_0 temp1 temp2 temp4 VDDx GNDx OR2_X1
xOR_1 temp3 temp4 coutx VDDx GNDx OR2_X1
.ENDS
*** Netlist: 4-bit Adder
xADDER_0 a0 b0 cin sum0 t0 VDD GND ADDER_X1
xADDER_1 a1 b1 t0 sum1 t1 VDD GND ADDER_X1
xADDER_2 a2 b2 t1 sum2 t2 VDD GND ADDER_X1
xADDER_3 a3 b3 t2 sum3 cout VDD GND ADDER_X1

.END
```

Figure 18: Full Adder Netlist File with Process Variations

Simulating the Netlist File

You can simulate a netlist file by typing:

$ hspice <netlist>.sp -o

The '-o' option writes the results of the simulation to an output listing file with the same name as the input netlist file but with a '.lis' suffix. Exclude '-o' if you want to print the simulation output on the terminal. You can specify the output file name by typing it after '-o', like the following:

$ hspice <netlist>.sp -o <output-file>

Plotting the Waveforms

To plot your simulation waveforms, run Synopsys WaveView by typing:

$ wv &

The main Waveview window is shown in Figure 19, click on ‘Open Waveform’.

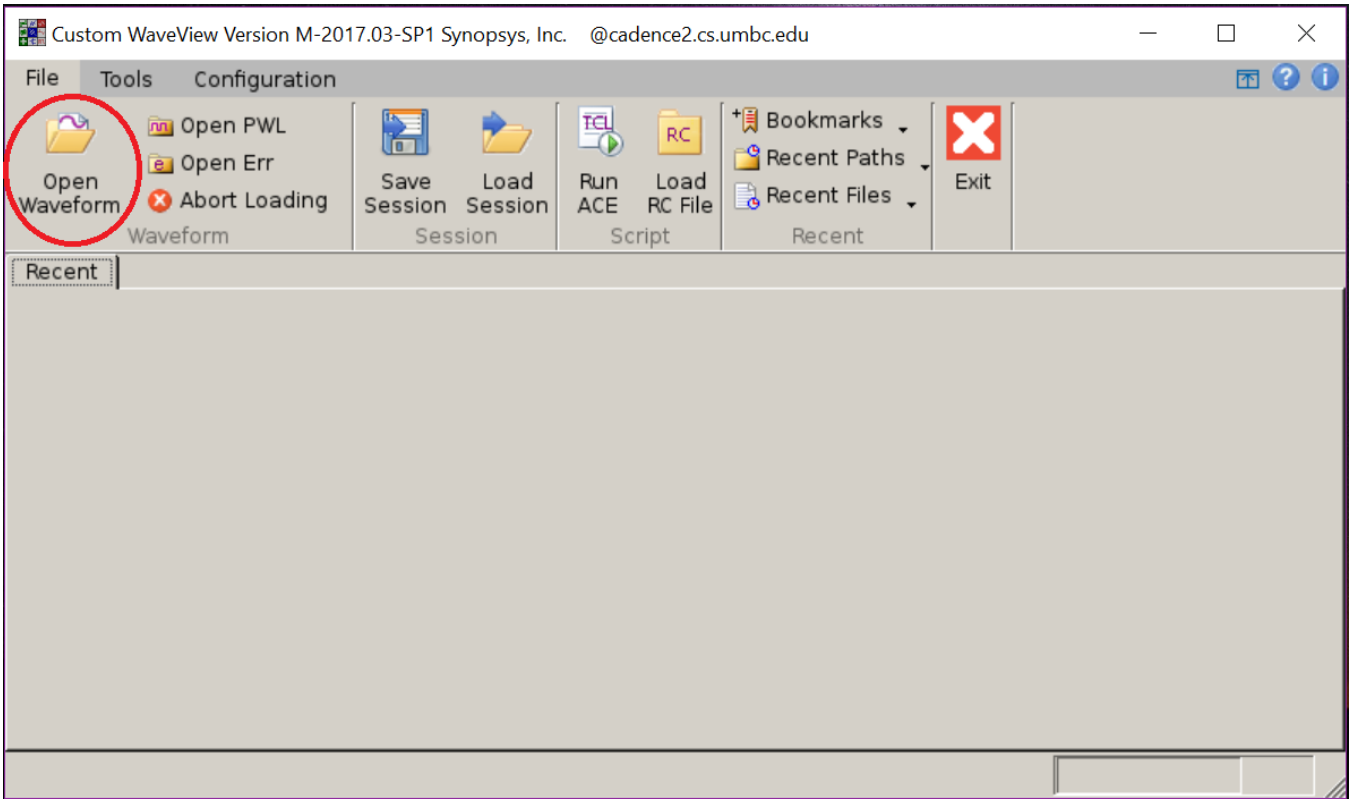

Figure 19: Waveview Window - the ‘Open Waveform’ button is encircled in red.

When you click on ‘Open Waveform’, the window shown in Figure 20 will appear, where you can choose and open the file containing the graphical data (‘.tr’ or ‘.lis’ depending on the command you used to dump the data). Use the file browser to select your file and click ‘OK’.

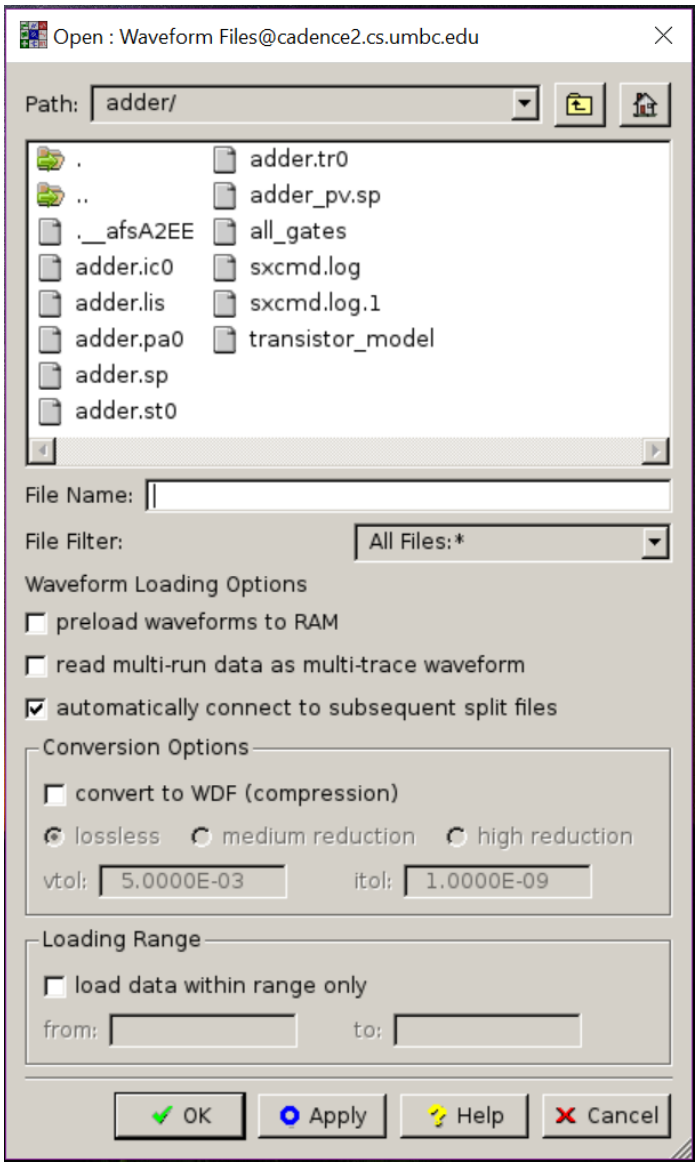

Figure 20: ‘Waveform Files’ Window

The file will appear in the ‘Waveform’ menu to the left. Expand your file by clicking on the small ‘+’ box, and click on the toplevel module. All the signals will appear in the box below it as shown in Figure 21. You can further expand your submodules to see their local signals as well.

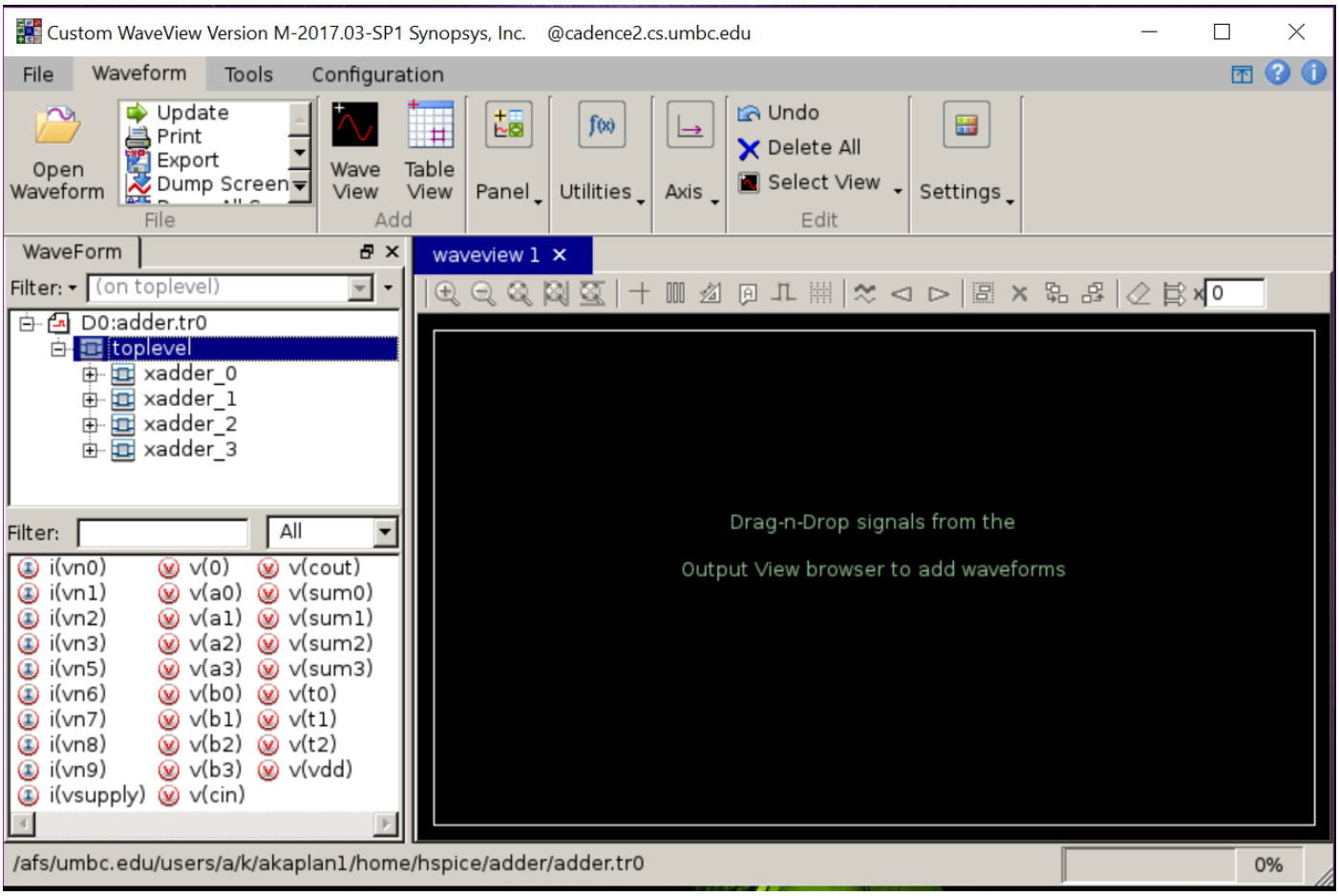

Figure 21: Waveview with the toplevel highlighted and the signals appearing in the box below it.

Plot signals by dragging and dropping them to the black area or double-clicking on them. Figure 22 shows a plot of all input and output signals of the 4-bit adder.

Figure 22: 4-bit Adder Input and Output Waveforms

For more information on how to use HSPICE, refer to the HSPICE User Guide which is uploaded on blackboard.

Other useful HSPICE references:
http://web.mit.edu/6.012/www/SPICEtutorial.pdf
http://www.seas.upenn.edu/~jan/spice/spice.overview.html
https://web.engr.oregonstate.edu/~moon/ece323/hspice/gstart.html