



Cadence tools

Brandon Rumberg

Tools to cover

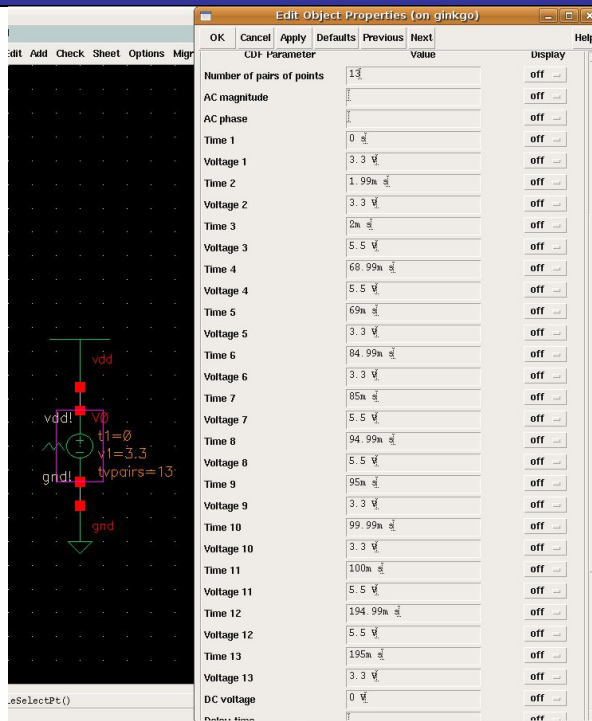
- Creating piecewise linear (PWL) files
- OCEAN scripts
- Verilog-A



Why use PWL files?

- Creating arbitrary waveforms in Cadence is tedious & changes are difficult

Piecewise linear source

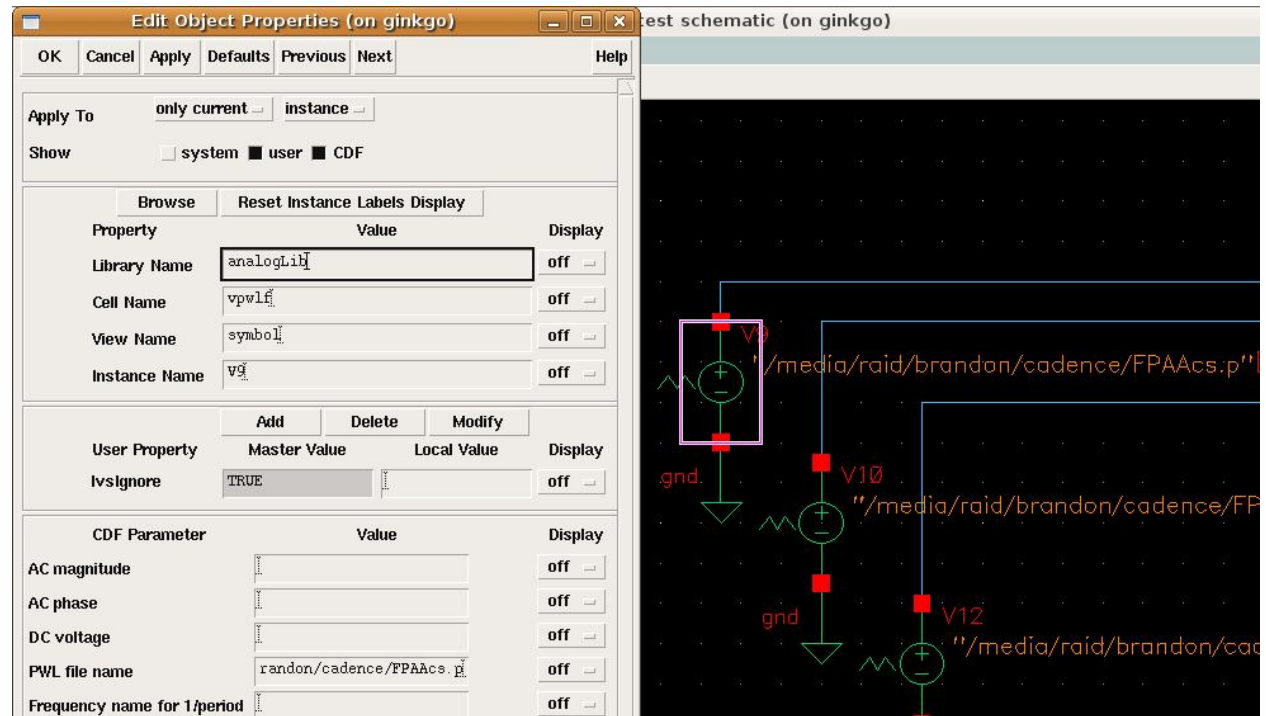


Combining sources



PWLF Source

- PWLF sources read from a piecewise linear file
 - ‘vpwlf’ in ‘analogLib’ library
- The only necessary parameter is the file path/name



Note: I've had trouble with the PWLF sources in the 'NCSU_Analog_Parts' library. They seem to have trouble finding the PWL file at times. So I recommend using the sources in 'analogLib'

When PWL files are useful



- Testing a circuit with a realistic input acquired from elsewhere
 - Such as a speech recording
- Testing chip-level configuration logic
 - Such as a serial interface that controls parameters/connections within the chip
 - You can reuse these files to simplify post-fab testing
- Performing a sequence of operations in one transient simulation
 - Particularly when the result of one operation affects the next operation

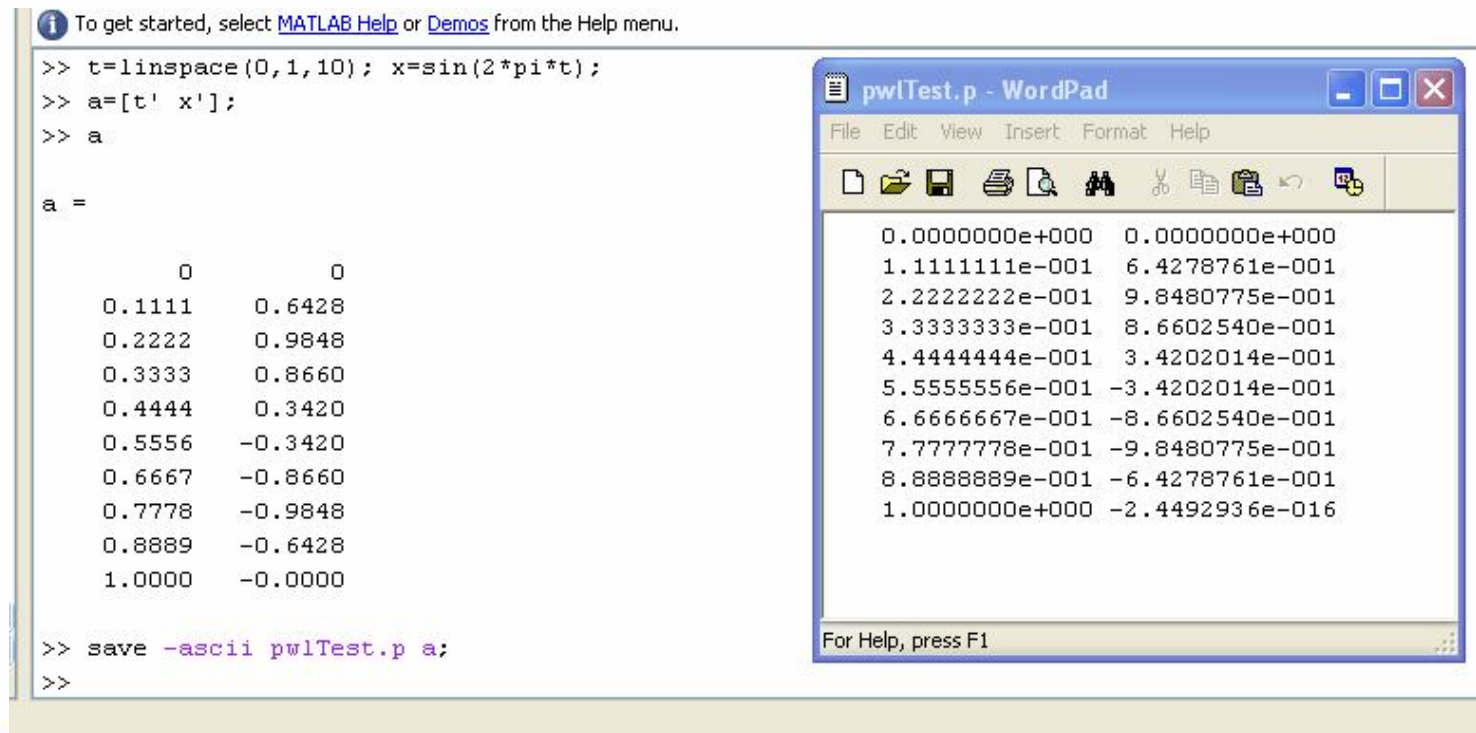
PWL files

- PWL files are text files with rows of time/value pairs
 - ‘Time’ and ‘Value’ are separated by a space
 - Each pair is on a separate line
- Such files can easily be generated with
 - Matlab/Octave
 - Excel (save as txt file)

Time	Value
0.00000000e+00	1.50000000e+00
1.17200000e-03	1.50000000e+00
1.17300000e-03	1.50000000e+00
1.27200000e-03	1.50000000e+00
1.27300000e-03	1.50000000e+00
3.35900000e-03	1.50000000e+00
3.36900000e-03	5.10000000e+00
4.36900000e-03	5.10000000e+00
4.37900000e-03	5.10000000e+00
1.54379000e-01	5.10000000e+00

Generating PWL files

- Create matrix with 'time' in the first column and 'value' in the second column
- Save using `save -ascii <filename> <matrix>`
- The file extension is arbitrary



The image shows two windows side-by-side. The left window is a MATLAB command window with the following text:

```
To get started, select MATLAB Help or Demos from the Help menu.  
>> t=linspace(0,1,10); x=sin(2*pi*t);  
>> a=[t' x'];  
>> a  
  
a =  
  
      0      0  
  0.1111  0.6428  
  0.2222  0.9848  
  0.3333  0.8660  
  0.4444  0.3420  
  0.5556 -0.3420  
  0.6667 -0.8660  
  0.7778 -0.9848  
  0.8889 -0.6428  
  1.0000 -0.0000  
  
>> save -ascii pwlTest.p a;  
>>
```

The right window is a WordPad window titled "pwlTest.p - WordPad". It contains the following text:

```
0.0000000e+000 0.0000000e+000  
1.1111111e-001 6.4278761e-001  
2.2222222e-001 9.8480775e-001  
3.3333333e-001 8.6602540e-001  
4.4444444e-001 3.4202014e-001  
5.5555556e-001 -3.4202014e-001  
6.6666667e-001 -8.6602540e-001  
7.7777778e-001 -9.8480775e-001  
8.8888889e-001 -6.4278761e-001  
1.0000000e+000 -2.4492936e-016
```

At the bottom of the WordPad window, it says "For Help, press F1".

Notes about PWLF

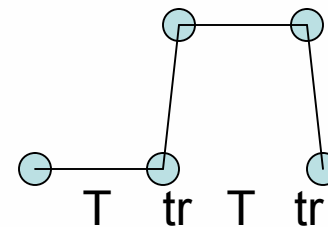
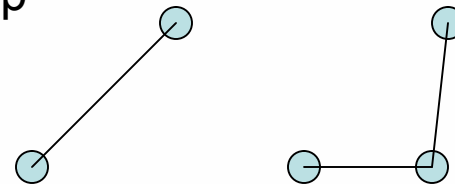


- Cadence seems to read the PWL file at the instance that the schematic is saved
 - So if you generate a new PWL file, then you need to resave your schematic before starting a new simulation

Tips for generating PWL files

- Keep in mind that PWL will be interpreted by connecting the dots
- To simplify the creation of a bitstream
 - Define the hold (T) and rise/fall times (tr), then
 - Write a function that turns a string of bits into the desired waveform

To create a step, you need to specify the point before the step



Tools to cover

- Creating piecewise linear (PWL) files
- OCEAN scripts
- Verilog-A

OCEAN scripts



- OCEAN is a simulator scripting language included in Cadence
- Can be thought of as
 - Parametric sweeps on steroids, or
 - A cross between Matlab and a simulator
- OCEAN
 - Exposes all simulator, graph, and calculator functions
 - Includes standard programming language functionality
 - File I/O
 - for/while loops
 - if/else branching
 - User-defined functions (called 'procedures')
 - Lisp syntax

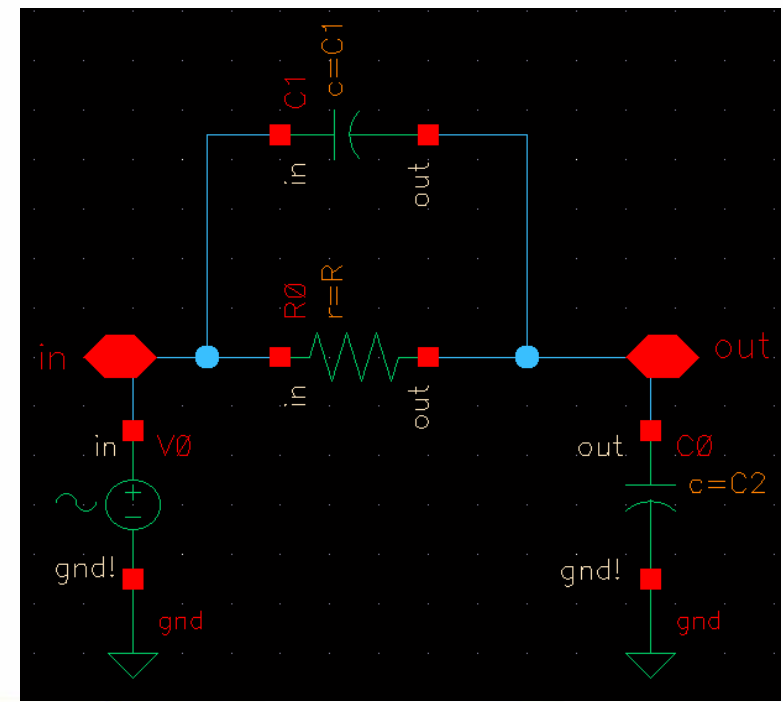
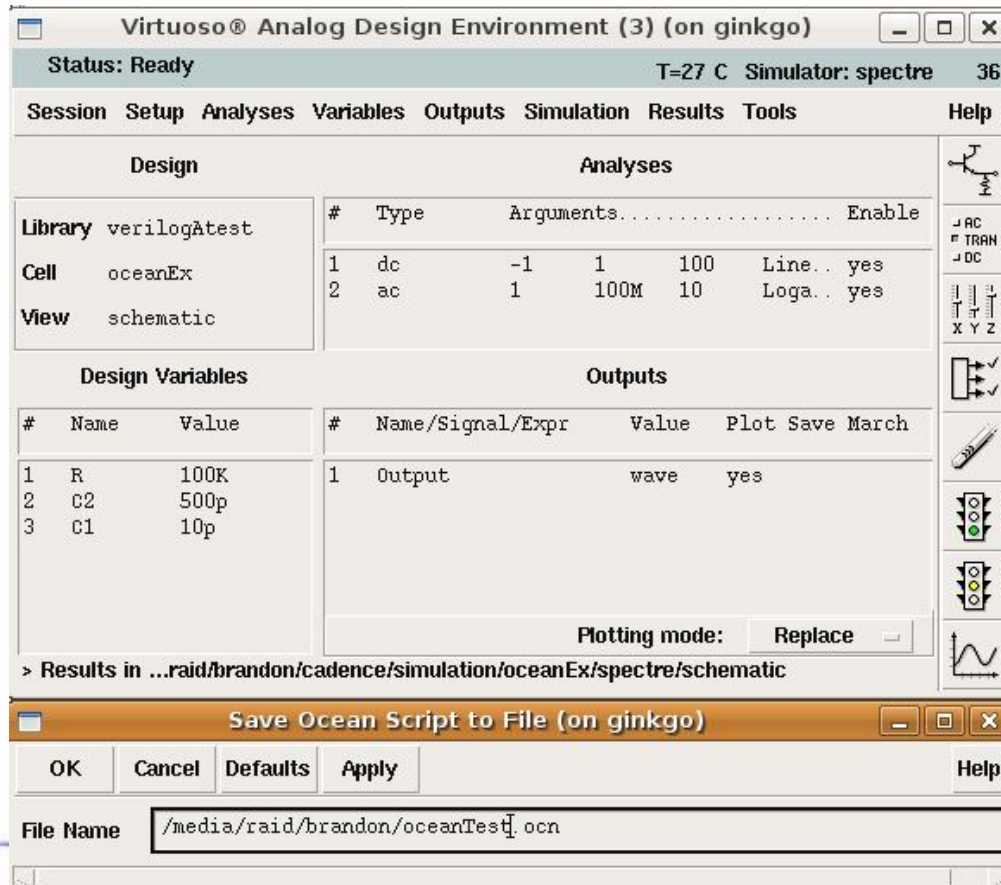
An OCEAN of possibility



- Circuit comparison
 - Create one OCEAN testbench and then automatically swap in/out different netlists
- Algorithmic circuit tuning
 - Rather than using parametric sweeps, create an OCEAN script that automatically tunes the circuit
- Parameter extraction
 - Have OCEAN extract the important circuit performance parameters and save them in a file

Creating an OCEAN script

- The easiest way to get started is to set up an initial simulation in the Virtuoso environment, then 'Session-Save Script'



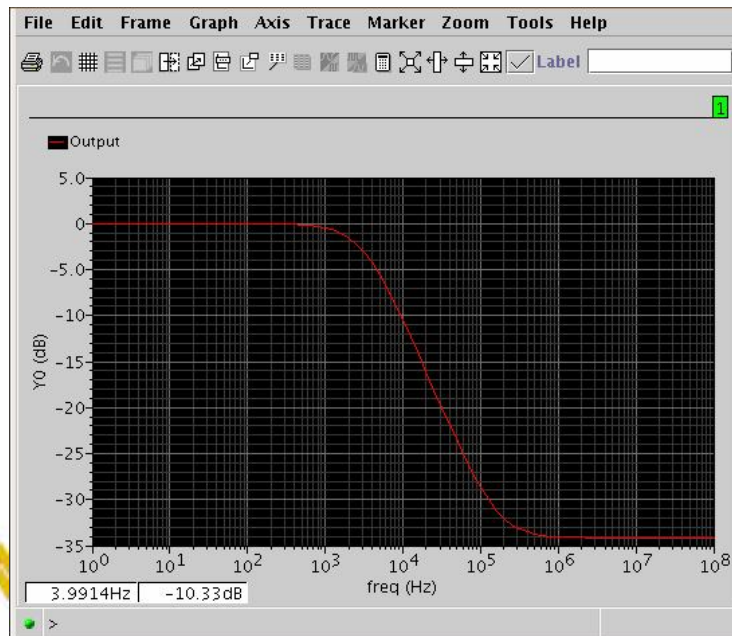
Sample OCEAN script

```
ocnWaveformTool( 'wavescan )
simulator( 'spectre )
design( "/media/raid/brandon/cadence/simulation/oceanEx/spectre/schematic/netlist/netlist" )
resultsDir( "/media/raid/brandon/cadence/simulation/oceanEx/spectre/schematic" )
analysis('ac ?start "1" ?stop "100M" ?dec "10" )      Analysis
analysis('dc ?dev "/V0" ?param "dc" ?start "-1"
           ?stop "1" ?lin "100" )
desVar(  "R" 100k )
desVar(  "C2" 500p )      Design variables
desVar(  "C1" 10p )
temp( 27 )
run()                    Simulate and results
Output = dB20(VF("/out"))
plot( Output ?expr '( "Output" ) )
```

- Edit the script with a regular text editor
- You can run the script using a different circuit by changing the path in `design ()`

Running an OCEAN script

- You can start ocean by typing `ocean` at the command line
- Then by typing `load("<script>.ocn")`
- To avoid retyping full commands, use
 - `!<first letters of command><Enter>`
 - e.g. `!l<Enter>` will rerun the last script



```
File Edit View Terminal Tabs Help
dc: dc = -240 mV (38 %), step = 20 mV (1 %)
dc: dc = -140 mV (43 %), step = 20 mV (1 %)
dc: dc = -40 mV (48 %), step = 20 mV (1 %)
dc: dc = 60 mV (53 %), step = 20 mV (1 %)
dc: dc = 160 mV (58 %), step = 20 mV (1 %)
dc: dc = 260 mV (63 %), step = 20 mV (1 %)
dc: dc = 360 mV (68 %), step = 20 mV (1 %)
dc: dc = 460 mV (73 %), step = 20 mV (1 %)
dc: dc = 560 mV (78 %), step = 20 mV (1 %)
dc: dc = 660 mV (83 %), step = 20 mV (1 %)
dc: dc = 760 mV (88 %), step = 20 mV (1 %)
dc: dc = 860 mV (93 %), step = 20 mV (1 %)
dc: dc = 960 mV (98 %), step = 20 mV (1 %)
Total time required for dc analysis 'dc' was 0 s.
modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.
simulation completed successfully.
reading simulation data...
...successful.
t
ocean>
```


Modifying an OCEAN script

- Use simulation result to calculate capacitor value that gives -20dB at high frequency

```
ocnWaveformTool( 'wavescan' )
simulator( 'spectre' )
design( "/media/raid/brandon/cadence/simulation/oceanEx/spectre/schematic/netlist/netlist" )
resultsDir( "/media/raid/brandon/cadence/simulation/oceanEx/spectre/schematic" )
analysis('ac ?start "1" ?stop "100M" ?dec "10" )
|
Rvar=100e3;
C2var=500e-12;
C1var=10e-12;
desVar( "R" Rvar )
desVar( "C2" C2var )
desVar( "C1" C1var )
temp( 27 )
run()

Output = dB20(VF("/out"))
plot( Output ?expr '( "Output" ) )

targetGain=-20;
actualGain=value(db20(VF("/out"))) 1M);
ratio=10**((targetGain-actualGain)/20);
C1var=C1var*ratio;

desVar("C1" C1var)

run()
Output = dB20(VF("/out"))
plot( Output ?expr '( "Output" ) )
```

Define design variables as variables so we can work with them

Run first with arbitrary starting values

Calculate high-frequency gain using standard calculator functions

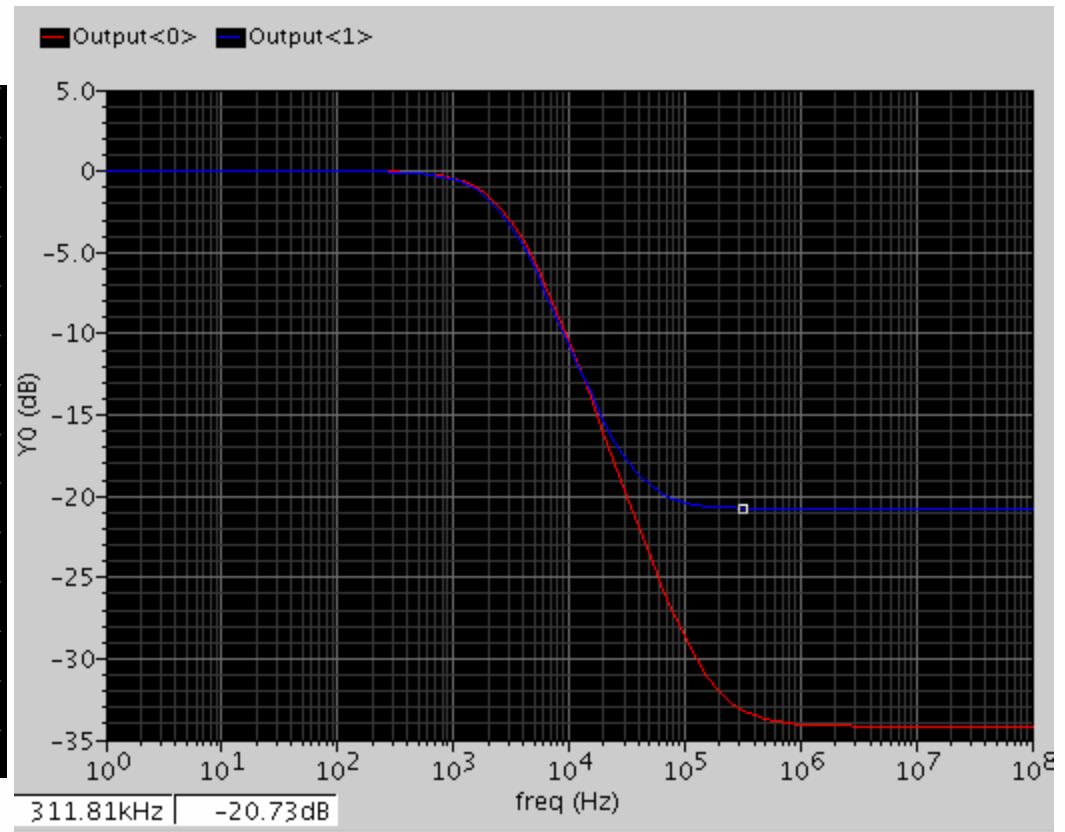
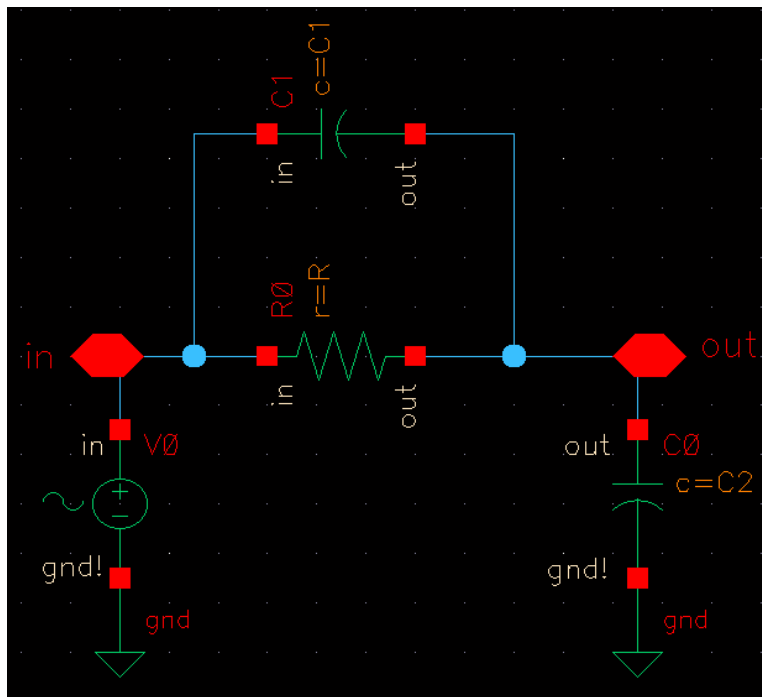
Calculate capacitor scaling to achieve target gain of -20dB
(note that exponentials are done with **, not ^)

Change the capacitor value based on the results

Resimulate



Results of previous slide



Tools to cover

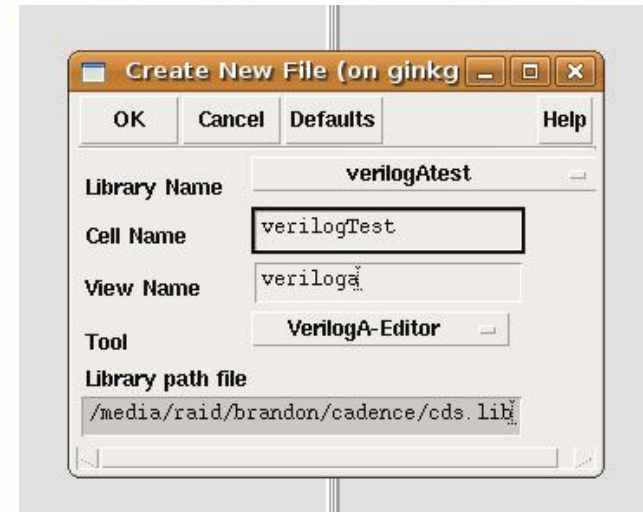
- Creating piecewise linear (PWL) files
- OCEAN scripts
- Verilog-A

Verilog-A

- A modeling language for analog simulation
- Uses for Verilog-A
 - Replace transistor-level circuits
 - Simulate top-level before all circuits are finished
 - Evaluate top-level impact of circuit nonidealities
 - Speed up simulation
 - Modeling non-standard circuit elements

Creating a Verilog-A cell

- Create a cell as normal, but choose 'VerilogA-Editor' for the tool

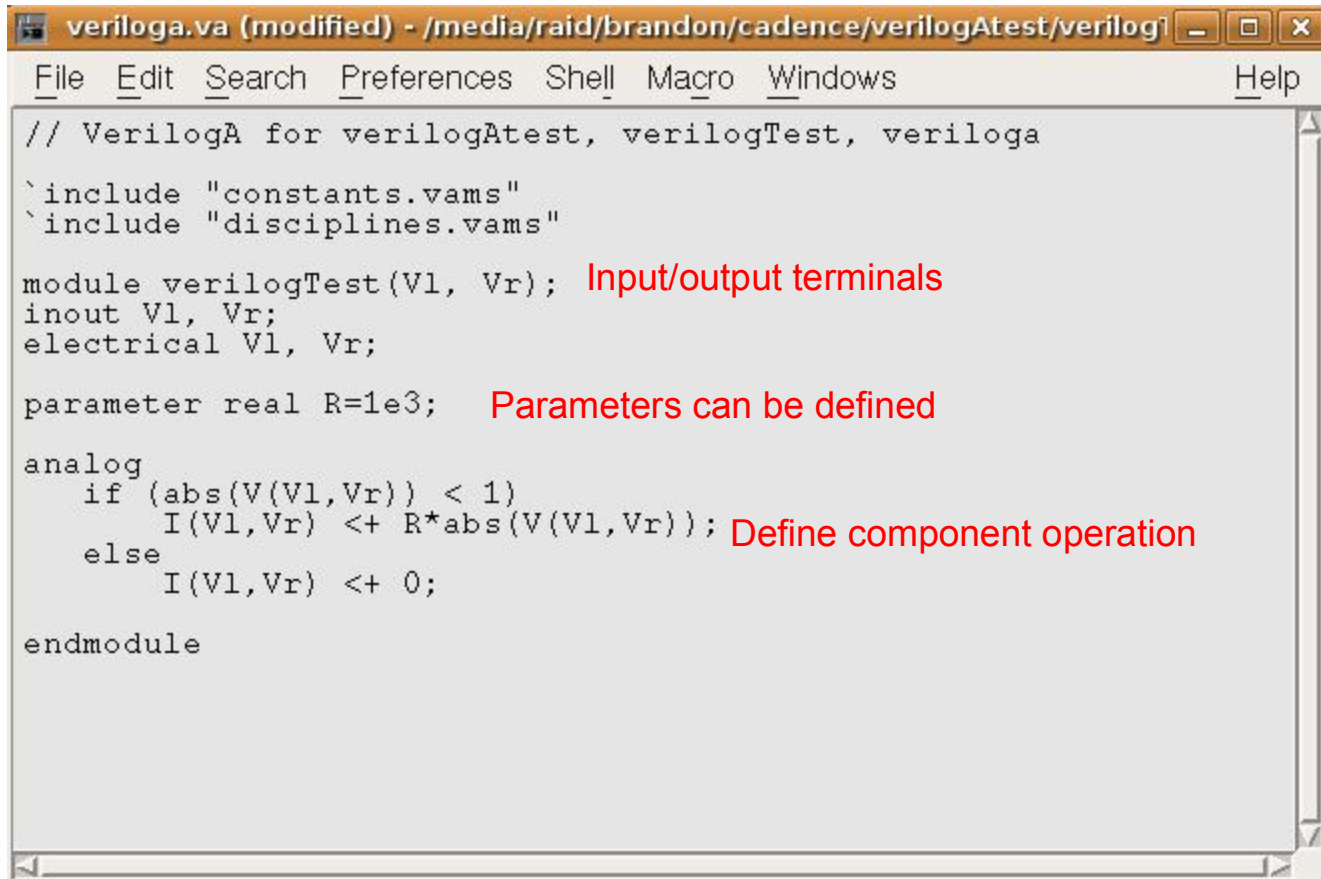


- This creates and opens a Verilog template

```
veriloga.va - /media/raid/brandon/cadence/verilogAtest/verilogTest/veriloga/ (on ginkgo)
File Edit Search Preferences Shell Macro Windows Help
// VerilogA for verilogAtest, verilogTest, veriloga
`include "constants.vams"
`include "disciplines.vams"
module verilogTest;
endmodule
```

Note: You can change the default text editor by typing `editor="<editor name>"` in the icfb window
The default is `vi`. You may want to change to `nedit` for a more conventional text editor.

Insert your Verilog-A code



```
veriloga.va (modified) - /media/raid/brandon/cadence/verilogAtest/verilog1
File Edit Search Preferences Shell Macro Windows Help
// VerilogA for verilogaTest, verilogaTest, veriloga
`include "constants.vams"
`include "disciplines.vams"

module verilogaTest(Vl, Vr); Input/output terminals
inout Vl, Vr;
electrical Vl, Vr;

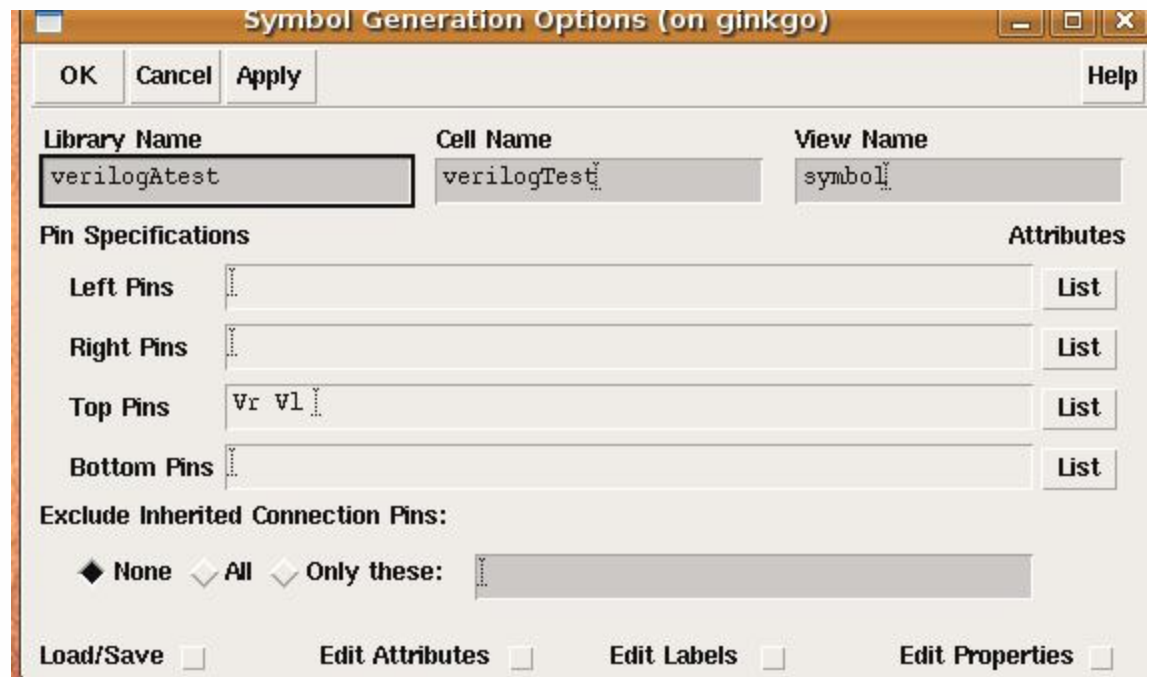
parameter real R=1e3; Parameters can be defined

analog
  if (abs(V(Vl,Vr)) < 1)
    I(Vl,Vr) <+ R*abs(V(Vl,Vr)); Define component operation
  else
    I(Vl,Vr) <+ 0;

endmodule
```

Symbol Creation & Compilation

- When you close the editor window, you will be asked if you want to create a symbol for you Verilog-A code
 - Select 'yes' so that it will automatically generate your pins
- The code is automatically compiled when you compile
 - If there is a syntax error you will receive a notification
- Next is the symbol generation dialog box shown to the write
- Create symbol as usual



The screenshot shows a dialog box titled "Symbol Generation Options (on ginkgo)". It has a standard Windows-style title bar with minimize, maximize, and close buttons. Below the title bar are three buttons: "OK", "Cancel", and "Apply", and a "Help" button in the top right corner. The dialog is divided into several sections:

- Library Name:** A text field containing "verilogAtest".
- Cell Name:** A text field containing "verilogTest".
- View Name:** A text field containing "symbol".
- Pin Specifications:** A section with four rows, each with a label and a text field, and a "List" button to the right:
 - Left Pins:** Empty text field.
 - Right Pins:** Empty text field.
 - Top Pins:** Text field containing "Vr V1".
 - Bottom Pins:** Empty text field.
- Exclude Inherited Connection Pins:** A section with three radio buttons: "None" (selected), "All", and "Only these:". To the right of "Only these:" is an empty text field.
- Bottom Row:** Four checkboxes: "Load/Save", "Edit Attributes", "Edit Labels", and "Edit Properties", all of which are currently unchecked.

Insert symbol for Verilog part & run sim

