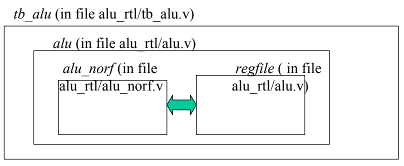
Update on Simple Pipelined System Homework

- I have updated the archive with my complete solution to this homework
  - I feel comfortable giving you my solution because it is such a poor quality solution – I expect you to do a much better job
  - The solution does illustrate the complete methodology and has the spice testbench
- I used logic synthesis to get a quick and dirty solution
  - Synthesis library only supported basic cells – most complex cell was a 2/1 mux
  - Longest register to register path had 16 gates on it!!!
  - My spice netlist did not buffer the clock network!
  - Meager testing showed my spice netlist to work with a clock period of 3 ns, but fail for 2 ns.

Modified *alu.v* RTL

- I modifed the *alu.v* RTL to have another level of hierarchy. The 'alu' module now has two modules in it
  - module *alu_norf* has all logic except the register file. This module contains all of the logic that YOU are supposed to specify.
  - This has same functionality as previous RTL, but will be easier to generate spice netlist once you have specified the logic.


tb_alu (in file alu_rtl/tb_alu.v)
alu (in file alu_rtl/alu.v)
alu_norf (in file alu_rtl/alu_norf.v
regfile (in file alu_rtl/alu.v)

Simulating your Gate Level alu_norf.v

- I have created another *modelsim* library for your gate level simulation
  - modelsim/src/alu_gate
  - The library directory includes a makefile.
- The file that contains the Verilog models of all of your gates should be placed in *modelsim/src/alu_gate/libcells.v*
- Your gate level solution for *alu_norf.v* (all logic except the register file) should also be placed in the same directory
  - Your gate level simulation (alu_gate) and the RTL simulation (alu_rtl) should produce the same results.
- To simulate, compile do:
  - gmake –f Makefile/alu_gate
  - qhsim –lib alu_gate tb_alu –c –do "run 20 us;quit"

Producing the Spice netlist

- I am assuming that your gate level solution (alu_norf.v) will be hierarchical
  - Correction: your gate level solution SHOULD BE hierarchical!!! Writing a flat netlist for a design of this complexity is too error prone!)
  - It will have a lot of modules in it for various common functions like a 4-bit DFF module, a 5-bit DFF module, a 2(?)-to-1 Mux module with 5-bit inputs, etc.
  - Other modules will tie these together to form the complete datapath.
  - The gate modules are in *libcells.v*
- As another example of a hierarchical gate level netlist, look at synopsy/rtl/test_flat.v
  - The top level module of this hierarchy is called 'control' and it calls several other modules defined in this file.

Flattening the Hierarchy

- The method I used to get a spice netlist first required flattening the gate level model
  - 'Flattening' removes all hierarchy and produces one module that just has all gate-level modules in it.
- The synopsys shell_script *synopsys/flatten.script* reads the file *rtl/test_flat.v* and produces a flattened version in *gate/test_flat.v*
  - Look at the two files and understand the differences
  - The shell script also makes all Verilog bus definitions to be written out as single net connections (I.e data[3:0] is written out as data[3], data[2], ..etc. This is important as Spectre does not understand busses.
- The file *synopsys/gate/alu_norf_gate.v* is my gate level solution.

Verilog to Spice

- The directory "*./spectre*" contains all the spice files for the testbench and a couple of perl scripts
- The perl script *v2sp.pl* can be used to convert a file containing Verilog instance declarations to Spectre instances
  - The file *spectre/alu_norf_gate.v* is my gate level solution with everything removed except instance statements
  - Do "cat alu_norf_gate.v | v2sp >alu_norf.sp" to convert to Spectre format
  - Look at the files *alu_norf_gate.v* and *alu_norf.sp* and understand the differences

Spice Netlist of Complete System

- The complete spice netlist is in the file *spectre/alu_netlist.sp*
  - Defines a subcircuit for alu_norf and includes the file *alu_norf.sp*. Creates an instance of the alu_norf subcircuit.
  - Includes the Verilog-A register file model "regfile.va" and creates an instance of this.
  - Includes the Verilog-A stimulus model "alustim.va" and creates an instance of this.
- The top level spice testbench is *spectre/tb_alu.sp*
  - Defines the clock driver, includes 'alu_netlist.sp', defines transient analysis, includes transistor models, etc.
- The file *spectre/lib_generic.sp* contains the spice subcircuit definitions for my gates using the N_def, P_def transistor models.

Stimulus model *alustim.va*

- The Verilog-A model *alustim.va* provides the stimulus for the testbench inputs
  - It includes a header file called 'alustim.h' that defines the 'program' to be executed
- The perl script *spectre/assemble_spdat.pl* can be used to produce a header file from a *.asm* file
  - '*assemble_spdat.pl add_prog.asm*' will create '*add_prog.h*'
  - Must copy '*add_prog.h*' to file '*alustim.h*' before executing spectre ('*spectre tb_alu.sp*')

Speeding Up Spectre

- Initially, you just want to know if your spectre netlist works or not
  - Can use the transistor models defined in 'tsmc018_fast.m'
  - These are MOS0 models so transistors are switches. Simulation time significantly reduced, but timing information is bogus
- You can also substitute Verilog-A models for some subcircuit calls to reduce the number of transistors
  - The file lib_generic_va.sp replaces the DFF, NAND3 subcircuits with Verilog-A models (dff.va, nand3.va).
  - This would be used to speed up your simulation to check functionality
  - By looking at 'dff.va', 'nand3.va' easy to write other models like this.

Other Comments on Spectre

- The file 'tb_alu.sp' has disabled saving of any simulation data for disk space reasons
  - 'opts save=none'
  - The *regfile.va* model prints message on each register file write
- If you want to save some signal values, try either
  - use explicit 'save' statements (see docs) or
  - use 'opts save=lvlpub nestlvl=1' (saves all public signals at first level of heirarchy).
  - If you save ALL signals at all levels of the heirarchy, the output file will be large
- The transient analysis statement uses 'errpreset=liberal'
  - This speeds up the simulation at the cost of some accuracy
  - I will use this option in testing your designs, so don't change it.