

## CS/EE 5830/6830 VLSI Architecture

### Possible project: SRT divider

This potential project is to design a 24-bit SRT Radix-4 divider. The specs for the divider are:

- ❑ 24-bit fractional inputs (both divisor and dividend) producing a 12-bit quotient and remainder. The reason to go to 24 bits is that the algorithm doesn't change when going to more bits, but it's slightly more realistic because single-precision floating point uses 24-bit mantissas, and the opportunities for optimization change by opening up more possibilities of combinational and sequential versions of the algorithm (see Figure 5.3 for details). (Of course, it's actually not really realistic because "real" floating point units would be built for at least double-precision with 53-bit mantissas!)
- ❑ Assume that the fractional inputs are already normalized before the start of the division. This normalization puts the arguments in the range  $[0.5, 1)$ . That is, the bit to the left of the binary point is 0, and the first bit to the right of the binary point is 1.
- ❑ Assume that in addition to the extra quotient bits required for whatever initialization you do, you should include one additional quotient digit for rounding (section 5.22 has details).
- ❑ Use radix-4 to reduce the number of iterations. I expect you will want to use  $\{-2, -1, 0, 1, 2\}$  for your quotient digit set, although you might want to be quirky and try something else.
- ❑ Speed is the main criterion that you're shooting for, with area being the next most important. (Of course, correctness is assumed before you make it go fast!)
- ❑ You can choose to implement on-the-fly quotient conversion or not, but the timing you report must be from the start of the division to the delivery of the correct quotient in conventional form.
- ❑ You should deliver a conventional-form remainder, but this doesn't have to be counted in the divider speed. Report the delay if the remainder takes longer to produce than the quotient.
- ❑ Use gates from the UofU\_Digital\_v1\_1 library and/or nmos and pmos transistors from the NUofU\_Analog\_Parts library.
- ❑ Another option would be to use Xilinx ISE and demonstrate your divider on a Xilinx chip.
- ❑ You can use Verilog and Synopsys to generate parts of your circuit if you like, but you cannot simply use the DesignWare divide function! You should be the designer for all circuit and architectural details, but you can use Verilog to assemble them if you like.
- ❑ The final timing results must be simulated using SpectreS on a transistor netlist. That is, no behavioral elements can be left in the circuit that you

measure. I assume that you'll want to use SpectreSVerilog in mixed analog/digital mode.

- There are lots of ways of doing quotient digit selection, but they are all based on a function that takes some bits of the scaled partial residual and some bits of the dividend to return a quotient digit. You can build this logic any way you like (gates, PLAs, and ROMs are some of the choices). Using case statements in Verilog is a great way to build a "ROM" out of standard cells. Synopsys does a good job of building an optimized combinational circuit for the ROM function.
- If you'd like to try a different version of the algorithm like radix-8, radix-16 with overlapped radix-4 stages, or r512 with quotient digit selection by rounding, go for it!

What to turn in:

- You should write a short report on the decisions you made in your divider implementation. Include architectural decisions, circuit implementation decisions, testing decisions and anything else we might find interesting.
- You should test your divider on a number of values using switch-level simulation in Verilog. Let us know how you're choosing your test cases and what you're trying to test. Report the results of the simulation, but don't turn in pages and pages of outputs. A few test outputs that demonstrate correct functionality are all that are needed.
- You should measure the speed of your divider using SpectreS and measuring the worst-case speed as best you can determine. Describe your test setup and your measurements.
- You should measure the area of your divider in terms of total transistor count.
- Turn in design documents for your divider including schematics, Verilog code, test code, etc.