CSE 30321 – General Tips and Pipelining Practicum

Chris Fallin

Nov 25, 2008

We will briefly take a whirlwind tour through some practical tips for the final project, with a specific focus on pipelining. The intent is to braindump some intuition and hard-earned experience to alleviate as much pain as possible for those of you who will be tackling formidable projects. As always, we (the TAs) are available in office hours for additional help and sometimes-sage advice. My office hours are Mon 3:00-4:30 pm and Weds 2:50-4:20 pm in Cushing 208 (the lab).

General Tips

- ModelSim >> ISE. If you are working on any serious project, eschew ISE entirely (except for area estimation!) and work with ModelSim. You can do a simple testbench directly in Verilog. ModelSim can save waveform views between sessions, can bookmark points in time, and is faster and more stable.
- When debugging, in general, it helps to show *all* available control signals. Get into the habit of tracking down problems by logical deduction, examining and verifying small pieces of your processor at a time, rather than treating your processor core like a black box with an unknowable fault. The tools allow very detailed analysis! For a pipeline, a uniform view organized by stage, with the instruction register in each stage displayed, can be very helpful.

Pipelining

- Throw away the original code. You may be tempted to take the existing datapath and chop it into pieces with pipeline registers. As with other vices in life, you must resist this temptation, because in the end it will *mess you up*. Make good decisions. In all seriousness, you will suffer much less brain-ache if you start with a clean-room implementation. We're looking at a ballpark figure of 1 KLOC of Verilog, which is a lot easier to get right if it was written for pipelining from the start.
- Think of the processor as a set of stages, with each stage purely combinational. (Ignore register files and memories for the moment.) Each stage takes input and produces

output, and all state is in the pipeline registers (Pregs). Then define the data that must pass between each stage. Construct the processor in a very regular manner using this paradigm. To avoid structural hazards, keep track of ownership explicitly – i.e., stage 2 owns the regfile read ports, stage 5 owns its write port, stage 4 owns data memory.

- More complicated setups – data forwarding and the like – work by extending this paradigm, so that each stage is a function of its own input *and* the input to other stages. For example, the ALU stage is also a function of the WriteBack stage, if writeback has a register value that it needs.

- Pass the whole instruction along the pipeline – don't try to generate all control signals at once. In actuality, this wastes some area (extra state bits) but reduces debugging effort by at least a factor of two and avoids the need to think about all stages at once.

- Build a Preg module and reuse it as a standard component. Each Preg takes two control signals: stall and bubble. A Preg is a latch with a 3-input MUX selecting the load value. In the normal case, the latch loads the output from the previous stage. When stalled, the latch loads its own value. When bubbled, the latch loads a "NOP" constant.

- Stall and bubble control must respect several constraints. If a stage is stalled, all prior stages must stall or else you lose data. The stage after a stalled stage must bubble, or else you duplicate an instruction. Be mindful of stalls in PC-update logic as well, or else you can skip instructions.

- Since instructions always stay in order, branches are simpler than you may think. We don't care if we start to fetch the instructions after a branch, because we will realize this mistake and flush the pipeline before the branch gets to the end, and the incorrect execution path will be (correctly) lost.

- TIMING, TIMING, TIMING. Preg on posedge and RegFile/Memories on negedge. It's black magic – trust us (but ask us if curious).

Best of luck, whatever your project! May your logic be correct, your run-times speedy and your bugs few as you embark upon what is surely to be the most enjoyable experience of your life.