CS150 Project

Tips and Advice

Dan Yeager, Daiwei Li, James Parker

Testing, testing, testing

Unit testing

- Requires you to modularize your design
- Example modules:
  - Branch decoder
  - Memory mapping read/write
  - Mem mask in/out
  - Hazard unit
  - Control/Datapath split

Testing, testing, testing

Unit testing

- Write tests before writing modules - test-driven development
- See `CacheTestBench.v, CacheTestTasks.vh, ALUTestBench.v` for examples
- Use tasks as "functions"
- `$random` to generate random numbers

Testing, testing, testing

Integration testing

- Difficult to write a testbench for the entire CPU, too many signals!
- Verify instruction-to-instruction behavior
- Write software (assembly files)

Testing, testing, testing

Writing assembly tests:

- ld file: (reference: http://www.math.utah.edu/docs/info/ld_3.html)

```
SECTIONS
{
    . = 0x10000000;
    .text : {
        * (.start);
        * (.text);
    }
}
```

Testing, testing, testing

Writing assembly tests:

- Makefile:

```
TARGET := <ld file>
include ../Makefile.llvm.in
```

Testing, testing, testing

Writing assembly tests:

- Assembly code:

```
_start:
li $s0, 0x00000020
addi $t0, $t0, 20
bne $t0, $s0, End
End:
```

Avoiding bugs

- Issues with understanding?
  - How does the UART work?
  - What does it mean for memories to be synchronous?
- Design document messy
  - Do the design before writing the Verilog!
- Bugs translating from design to code
  - Don't code when tired
  - Testbenches will help catch these bugs

Fixing bugs

- Use make report - check for warnings/errors
- Check for default values in case or if/else if statements
- Check reset values
- Check wire widths - these errors should be caught in synthesis
- Use Modelsim
  - Navigate to part of signal you care about
  - e.g. EchoTestbench -> go to DataOutValid = 1

Style

Verilog header file (.vh)

- Constants: `Opcode.vh`

```
`ifndef OPCODE
`define OPCODE

// Opcode
`define RTYPE 6'b000000
// Load/store
`define LB 6'b100000
`define LH 6'b100001
`define LW 6'b100011
`define LBU 6'b100100
`define LHU 6'b100101
...
`endif //OPCODE
```

Style

Verilog header file (.vh)

- Tasks

```
task SingleCacheWrite;
  input isDcache;
  input [31:0] task_addr;
  input [31:0] task_cache_din;
  input [3:0]  task_cache_we;
  input        task_expect_hit;
begin
  SetupWrite(isDcache, task_addr, task_cache_din, task_cache_we);
  ClockInRequest();
  WaitForStall();
  VerifyWrite(isDcache, task_expect_hit);
end
endtask
```

Style

- Specify wire widths: 0 vs 32'b0
  - Better yet, use a define for hardwired values
- Multiple signal assignments
  - For something complicated like control, should have spreadsheet
    - Even better, generate Verilog from spreadsheet!
  - {Signal1, Signal2, etc...} = {Signal1Val, Signal2Val, etc...};
    - e.g. assign {opcode, rs, rt, rd, etc... } = {instruction[31:26], instruction[25:21], instruction [20:16], etc...}

Style

- always@(*) + case/if else instead of mux module instantiations and ? : statements
  - Makes sense when you have a lot of muxes driven by the same signal
  - Why? Easier to read

Style

- Group "like" signals
  - e.g. Put all pipeline registers in one always@ (posedge clk) block
- Label signals
  - I_{}, X_{}, M_{} for signals from different stages