6.375 Tutorial 5

RISC-V 6-Stage with Caches

Ming Liu

Overview

- Interface as parameters
- 6 Stage Pipeline
- Caches

Notes

- In Lab 6, you should use the `Fifo::*;` library defined in `includes/`
  - Do NOT use the BSV library `FIFO::*;`
- Code shown in lecture is meant to be a guideline
  - Make sure you understand it because the code you write in the lab may be similar but not identical

Interfaces as Module Parameters

- Entire interfaces/subinterfaces (not just static types) can be passed as a parameter into a module

```
module mkTb();
  Integer init = 0;
  IMemory iMem <- mkIMemory;
  Proc#(2) p <- mkProc(iMem, init);
endmodule
```

Last parameter is always interface of module being defined

```
module mkProc(IMemory mem,
              Integer init,
              Proc#(ncores) ifc);
  rule doFetch;
    mem.req(addr);
    ..
  endrule
  method ... //define Proc interface
  endmethod
endmodule
```

"ifc" is optional when no other parameters

```
module mkProc(Proc);


endmodule
```

Interfaces as Module Parameters

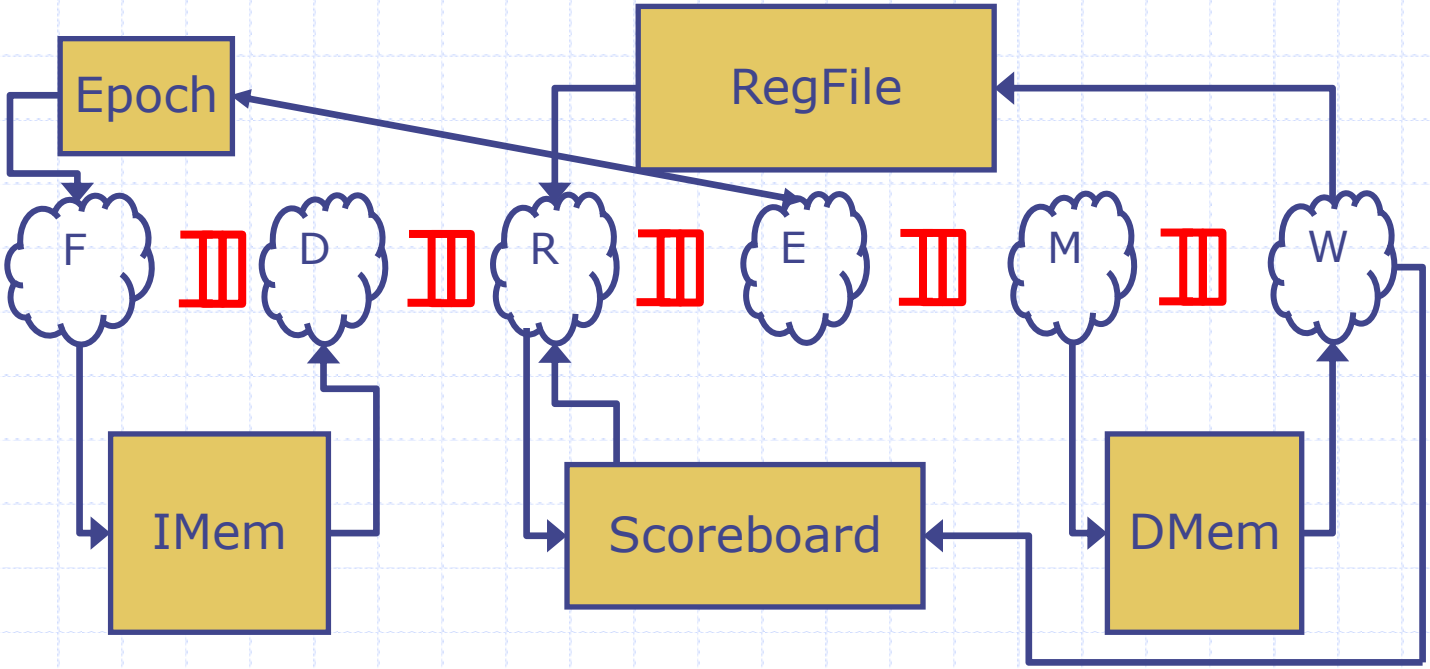
- Passing interfaces to a module breaks synthesis boundary (*synthesize*)
  - Compiler currently cannot create a separate Verilog file for this module due to scheduling difficulties
- Beware that all the scheduling/rule legality checks are still in place. Even across modules

```
module mkTb();
  IMemory iMem <- mkIMemory;
  Proc p0 <- mkProc(iMem);
  Proc p1 <- mkProc(iMem);
endmodule
```

If both p0 and p1 contain rules that calls iMem.req, then they may conflict.

Six Stage Pipeline

- Use the 2-stage pipeline as a starting point. Subdivide the rules and create Fifos (elastic pipeline) between stages.
  - Not necessary to try and modularize the stages

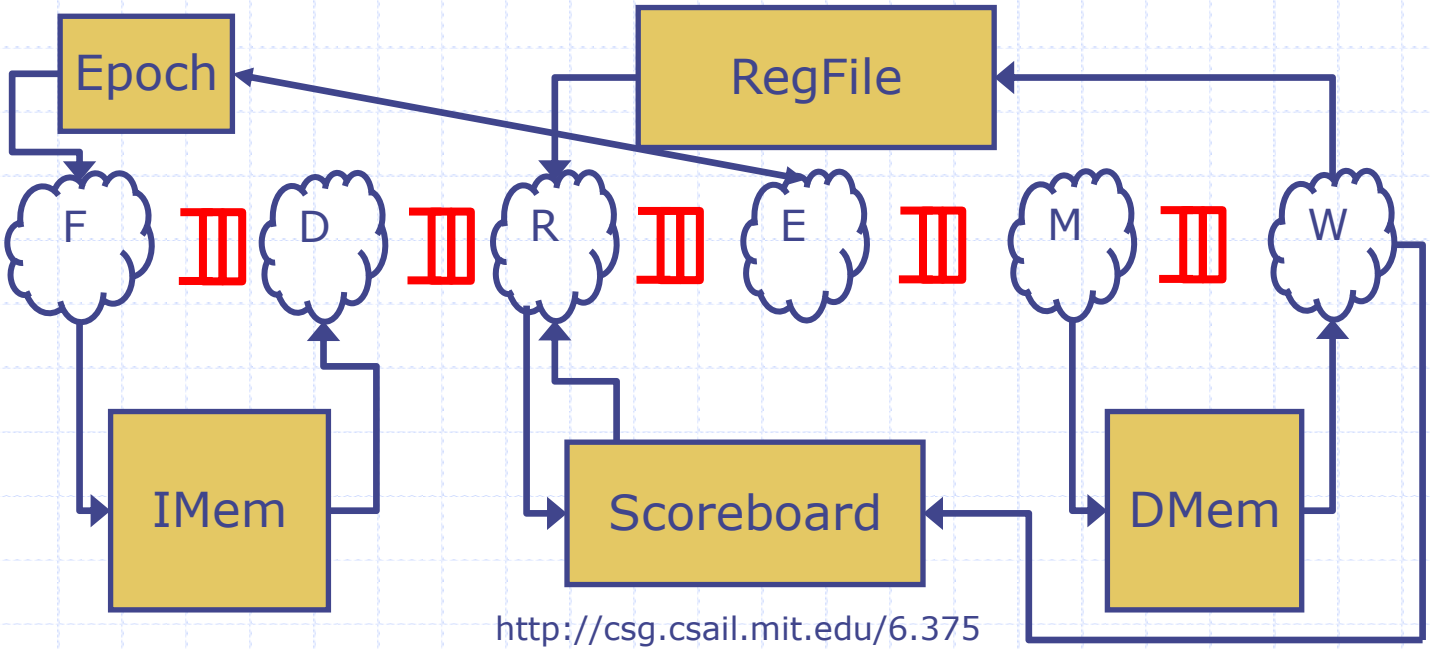


Six Stage Pipeline

- At Execute stage, we filter out instructions with mismatched epochs (caused by a prior branch insn)
- However we can't just "kill" or toss out the instruction as we did in the 2 stage. Why?

RegFetch stage has changed state of our processor → Scoreboard insert!

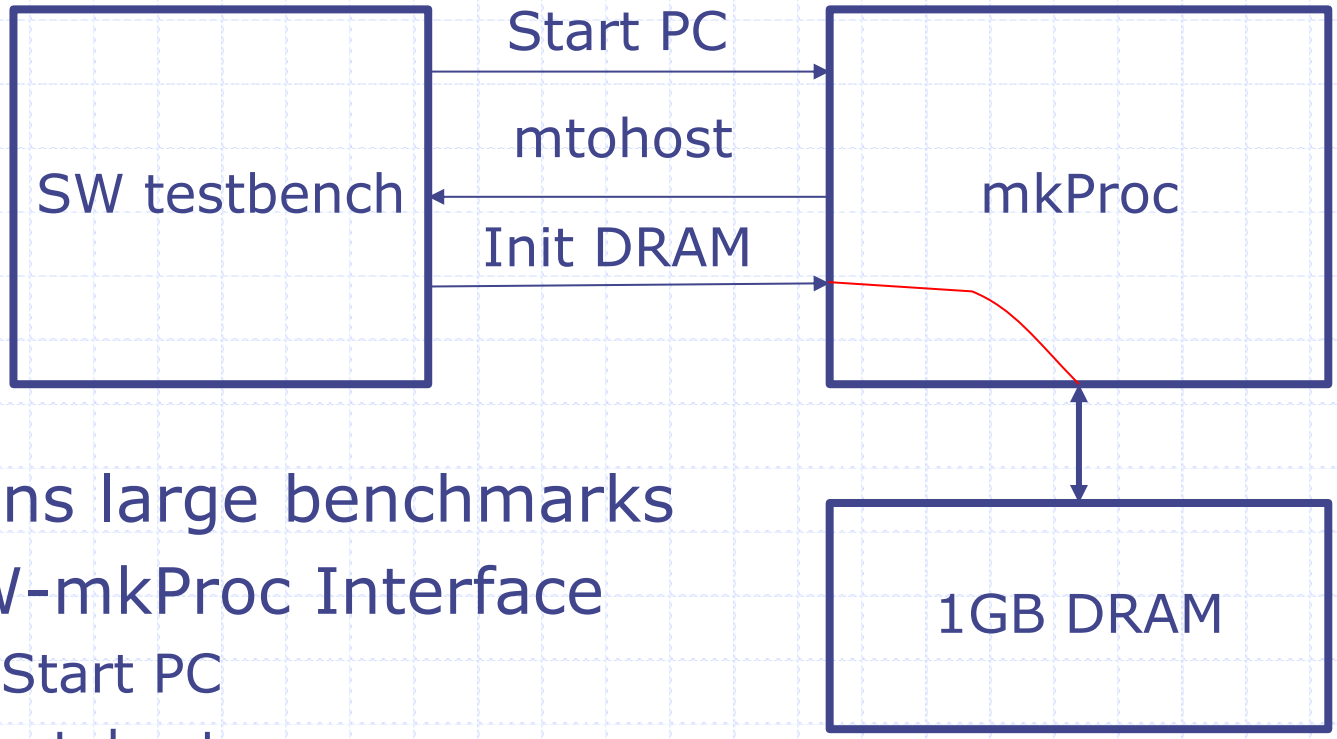
Solution: Mark instruction as "poisoned" and pass it on (but do not process it). Remove from SB in Writeback stage



Other Notes

- Play with the type/size of RegFile and Scoreboard to see its effect on performance
- Look at `info_dut/mkProc.sched` to see if the schedule is what you expect
- Pay attention to scheduling warnings related to your processor

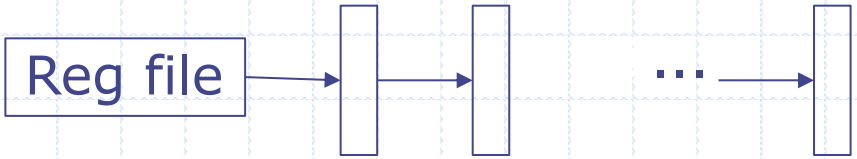
Part B: FPGA Infrastructure



- Runs large benchmarks
- SW-mkProc Interface
  - Start PC
  - mtohost
  - Init DRAM
- Avoid re-programming FPGA
  - Reset FPGA after each test

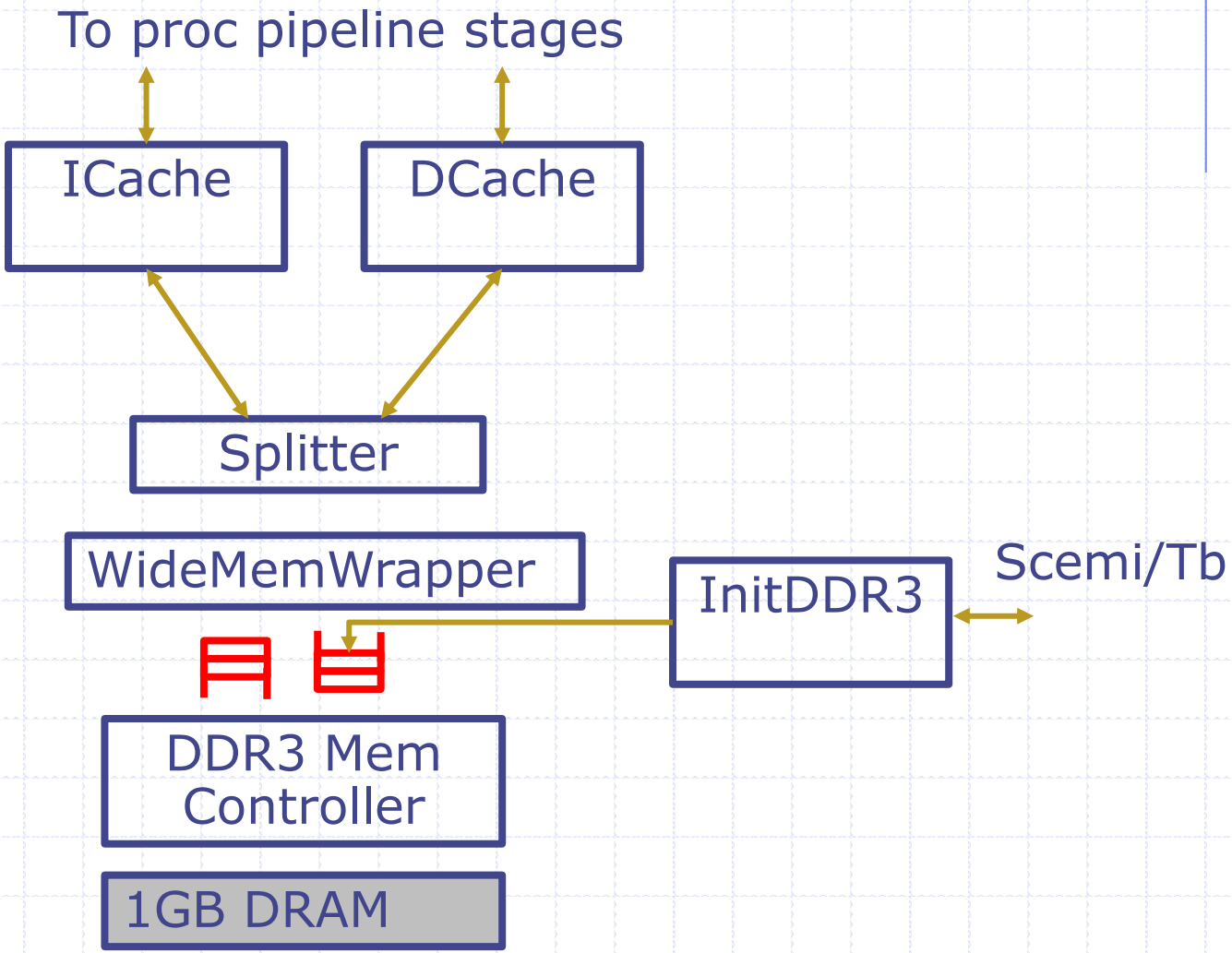
Simulation

- DRAM
  - RegFile + pipelined delay of resp



- We also simulate DRAM initialization
  - Longer simulation time

Memory Interface



Implementing Cache

- MemUtil.bsv and CacheTypes.bsv has a lot of useful utilities functions and constants you should use
- Port code from DDR3Example.bsv to your 6-stage pipeline
  - Only very minor changes to the pipeline itself is needed
- Guard all your rules in processor with (csrf.started)
  - Proc should execute only when the host tells it to start