MS108 Project Report

ZYHowell

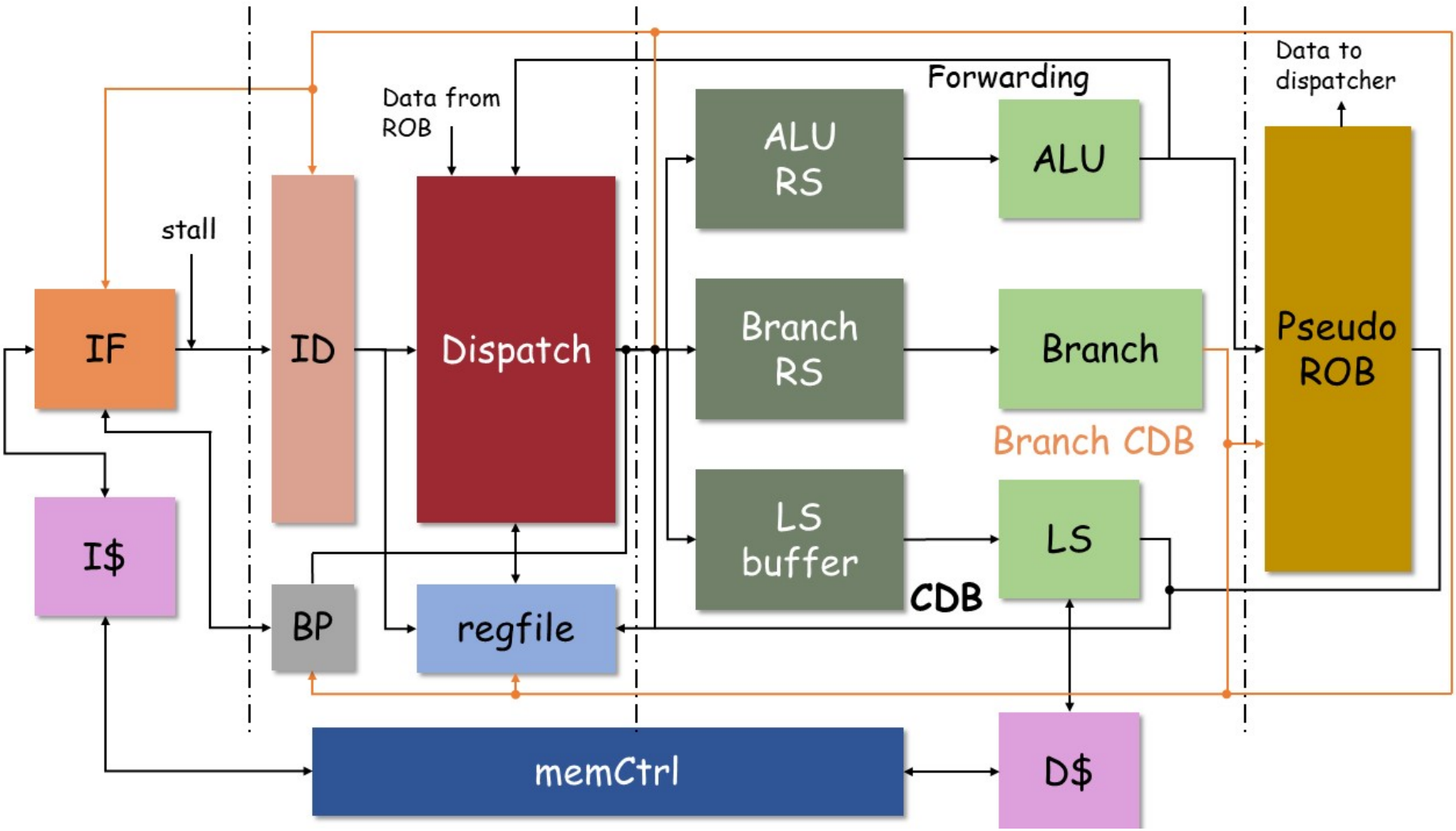
Pitfall, Stories and Great Ideas(from others)

January 4, 2020

Overview

- Tomasulo;
- iCache(DM and 2-way), dCache(write back);
- Branch Mask.
- GShare. (no more space to improve on FPGA)

Final Design



Branch Mask and Shadow Regfile

Consider the following code:

```
      lw  r10 0(r5)       #mask: 000
      bne r11 r12 jump;   #mask: 000, branch num:0
      add r10 r13 r14;    #mask: 001, so mask[0]=1
      ...
jump: add r11 r10 r12     #mask: 000
```

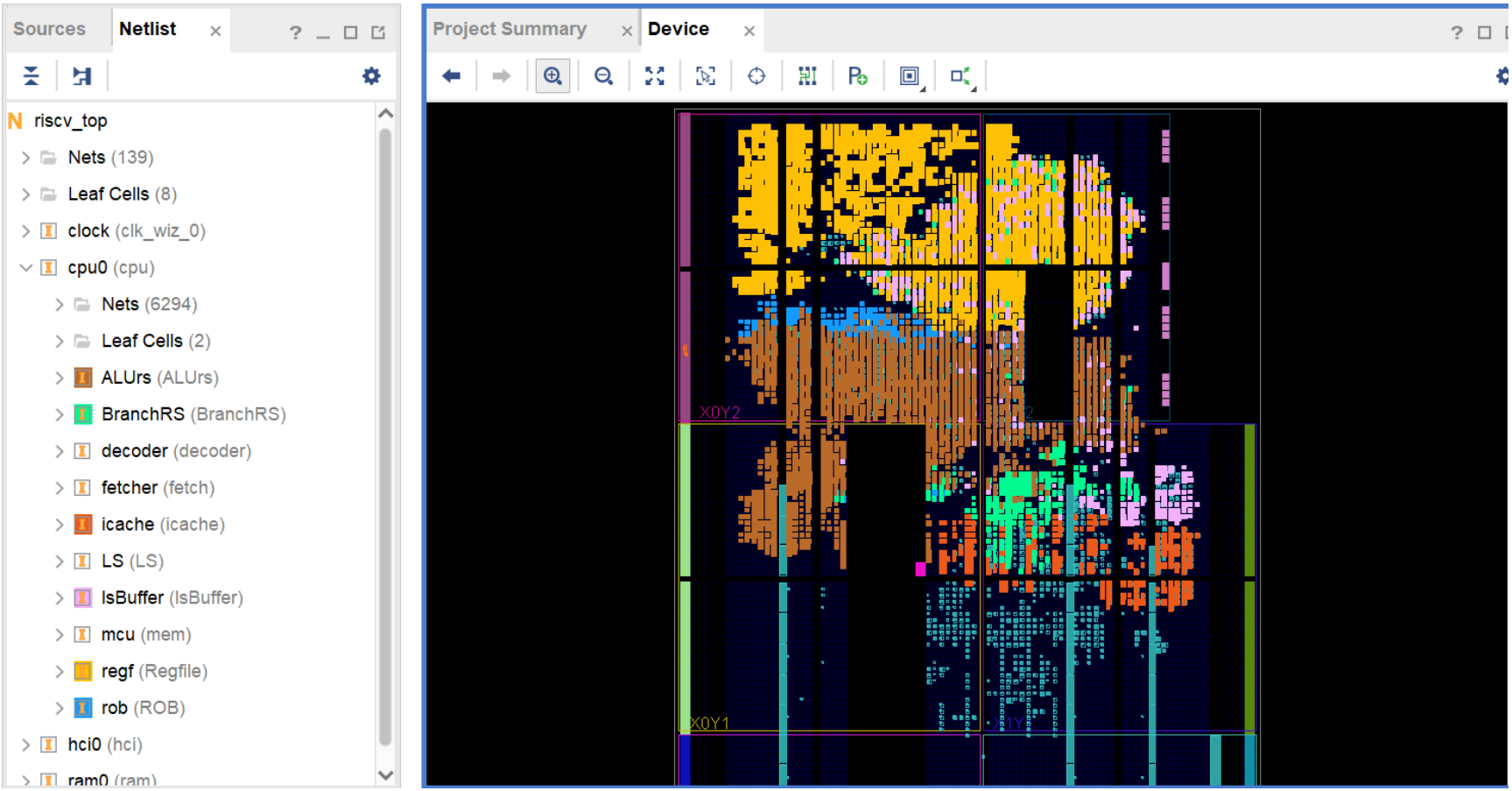
The jump needs to know it should receive from the "lw r10 0(r5)" instead of "addi r10 r13 r14"

Just like a stack:

|data tag1 (branch free,lw) tag2 (branch num0,add r10)

Branch Mask

- The reason why ALUrs and regfile so large is mainly logic for branch.



Sometimes bigger and dumber is better?

Pitfall, according to CAAQA ed.6, chapter three.

- Simply adding ROB line or RS has no improvement;
- It costs a lot to enlarge the shadow page but is obviously useless.

However,

- Enlarging the Cache seems useless, due to the limited testbench. (admitting that meeting the testbench is the most important for a design)
  Itanium 2(well, it is also a failure) and i7 vs. Pentium 4.
- Small cache may lead to worse performance(even write back)
  Load and replace, then done. So wait for a load more.

And sometimes smarter is better than bigger and dumber?

Pitfall, according to CAAQA ed.6, chapter three. Really?

- The story of write-back dCache.
  However:
- The result of branch masks;
- The result of special judge when entering RS and ROB. $\star\star\star$
  posedge 1: ID;
  posedge 2: in RS, then ready; (can let it work here)
  posedge 3: notice ready so do it;

CAAQA mentions to consider the situation.

My path

- Learn and design;(6th Sep - 1st assignment)
- One without ROB first; (after python interpreter - 23rd Nov, debug till 5th Dec)
- Add ROB, branch policy is stall; (23rd Nov - 9th Dec)
- Add branch mask, then make ROB a pseudo one to simplify. Shadow regfile also;(till 15th Dec)
- Add Dcache and Gshare. (1st Jan - 2nd Jan)

More stories

- Shadow register again: No shadow data: $99\% \rightarrow 50\%$ LUT; remark: considering the most complex logic, instead of the largest reg. (FF is no more than 20%)
- Misprediction judgement; (more LUT but not very, 1ns less delay)
- Not assign a=b, simply use b; (not elegant, but 0.7ns less delay)
- Only control Enable signal strictly and others loosely.

```
finalAddr = (op1 < op2) ? jumpAddr : nxtAddr;
assign mispredict = (finalAddr == jumpAddr) ^ predict;
//v.s.
begin
    finalAddr = (op1 < op2) ? jumpAddr : nxtAddr;
    mispredict = (op1 < op2) ^ predict;
end
```

How to judge a design?

- testbench? 2.2-2.25s for pi under 80MHz, delay is positive.
  -actually not enough this time(still wrong once after passing);
- CPI?
  -bad design can have good CPI but low frequency;
- Max Frequency and CPI?
  -based on the tech of chip creation.
  -1950s and early 1960s did not need Tomasulo.

So what is this project for? A 1960 CPU or a 2020 CPU? Is Tomasulo a 2020 CPU? Remind Pentium 4.

Thanks