Progress Report

Who worked on each part of the design

Jiamao Xu: Hazard detection & forwarding & Static branch predictor & RVFI

Garen Hu: Arbiter Hazard detection & forwarding & Static branch predictor

Jerry Wang: Integration of L1 cache Hazard detection & forwarding & RVFI

The functionalities you implemented

Based on the function we came up with from CP1, we added more advanced implementations including L1 cache system, hazard detection & forwarding, and static branch predictor. We also integrate the RVFI monitor to compare with the true result.

The testing strategy you used to verify these functionalities

First, we run the provided test code, and we use Verdi to verify the mem address and the corresponding register value. If we find something goes wrong. We find the mem address and we manually change the halting condition to a few lines after the problem line. For example, if the problem occurs at memory address 80000198, we set up halting at 800001a0 to prevent executing code after it. And then we start with the instruction data and instruction address and trace the data along the datapath to see where the problem is. In addition, we spent a huge time debugging our forwarding unit, we kept tracing the key registers from the end to where it went wrong and tried to fix our code.

The timing and energy analysis of your design: fmax & energy report from Design Compiler

```
clock my_clk (rise edge)                                  10.00      10.00
clock network delay (ideal)                                0.00      10.00
clock uncertainty                                         -0.10       9.90
cpu/datapath/PC_register/data_reg[31]/CK (DFF_X1)          0.00       9.90 r
library setup time                                        -0.04       9.86
data required time                                                    9.86
--------------------------------------------------------------------------
data required time                                                    9.86
data arrival time                                                    -5.78
--------------------------------------------------------------------------
slack (MET)                                                           4.08
```

f_max=1/(10-slack)*1000= 1/(10-4.08)*1000=168.918…

Power report:

```
--------------------------------------------------------------------------------
                                    Switch   Int      Leak     Total
Hierarchy                           Power    Power    Power    Power    %
--------------------------------------------------------------------------------
mp4                                190.017  554.036 7.68e+05 1.51e+03 100.0
  cache_adaptor (cacheline_adaptor)  17.178   33.625 3.66e+04   87.376   5.8
  arbiter (arbiter)                   5.157    2.041 2.34e+04   30.609   2.0
  d_cache (cache_0)                  69.017  209.797 2.44e+05  522.419  34.5
    bus (line_adapter_0)              0.813    0.246 4.95e+03    6.009   0.4
    datapath (cache_datapath_0)      68.061  209.092 2.38e+05  515.462  34.1
      dirty (array_width1_0)          0.601    2.562 1.57e+03    4.729   0.3
      valid (array_width1_1)          0.197    0.765 1.50e+03    2.464   0.2
      tag (array_width24_0)           0.766    5.713 1.93e+04   25.801   1.7
      DM_cache (data_array_0)        65.032  199.081 1.96e+05  460.384  30.4
    control (cache_control_0)         0.143    0.459  345.840    0.948   0.1
  i_cache (cache_1)                  68.656  213.908 2.37e+05  519.712  34.4
    bus (line_adapter_1)              1.408    0.901 4.60e+03    6.911   0.5
    datapath (cache_datapath_1)      67.203  212.590 2.32e+05  512.190  33.9
      valid (array_width1_3)          0.109    0.779 1.58e+03    2.466   0.2
      tag (array_width24_1)           0.874    6.022 1.94e+04   26.267   1.7
      DM_cache (data_array_1)        64.045  204.845 2.03e+05  471.973  31.2
    control (cache_control_1)       4.39e-02    0.417  148.749    0.610   0.0
  cpu (cpu)                          29.331   94.459 2.26e+05  349.565  23.1
    hzard_control (hazard_control) 4.38e-03 4.97e-03   45.430 5.48e-02   0.0
    datapath (datapath)              28.844   94.171 2.23e+05  346.447  22.9
      forward (forwarding)            0.352    0.378 1.75e+03    2.476   0.2
      CMP (branch)                    0.273    0.295 4.29e+03    4.855   0.3
      ALU (alu)                       5.165    4.221 2.26e+04   32.000   2.1
      regfile (regfile)               1.954   22.658 1.11e+05  135.225   8.9
      MEM_WB_mem_rdata (register_3)   0.208    1.083 3.24e+03    4.535   0.3
      MEM_WB_aluout (register_4)      0.927    3.328 3.21e+03    7.466   0.5
      MEM_WB_br_en (register_width1_0) 2.90e-02  0.445  147.346    0.621   0.0
      MEM_WB_u_imm (register_5)       0.578    2.403 2.07e+03    5.046   0.3
      MEM_WB_pc (pc_register_1)       0.946    3.374 3.27e+03    7.586   0.5
      MEM_WB_ctrl (register_width46_0) 0.376   1.548 1.09e+03    3.019   0.2
      EX_MEM_wdata (register_7)       0.867    3.317 3.16e+03    7.341   0.5
      EX_MEM_br_en (register_width1_1) 2.99e-02  0.450  144.320    0.624   0.0
      EX_MEM_u_imm (register_10)      0.563    2.460 2.07e+03    5.097   0.3
      EX_MEM_aluout (register_11)     1.401    3.352 3.20e+03    7.953   0.5
      EX_MEM_pc (pc_register_3)       0.892    3.284 3.23e+03    7.411   0.5
      EX_MEM_ctrl (register_width46_1) 0.450   1.940 1.68e+03    4.069   0.3
      ID_EX_rs2 (register_13)         0.977    3.682 3.19e+03    7.848   0.5
      ID_EX_rs1 (register_14)         0.971    2.965 3.25e+03    7.186   0.5
      ID_EX_u_imm (register_15)       0.604    2.497 2.07e+03    5.169   0.3
      ID_EX_i_imm (register_16)       0.913    3.684 2.76e+03    7.360   0.5
      ID_EX_alumux2 (register_17)     1.006    3.736 3.19e+03    7.931   0.5
      ID_EX_alumux1 (register_18)     0.998    3.107 3.26e+03    7.369   0.5
      ID_EX_pc (pc_register_5)        0.972    3.554 3.24e+03    7.762   0.5
      ID_EX_ctrl (register_width46_2) 1.347    4.190 3.99e+03    9.527   0.6
      IF_ID_pc (pc_register_7)        0.990    3.556 3.23e+03    7.779   0.5
      imm_decoder (ir)                1.368    3.807 3.26e+03    8.433   0.6
      PC_register (pc_register_8)     1.160    3.317 3.23e+03    7.711   0.5
    control (control_rom)             0.483    0.284 2.30e+03    3.064   0.2
1
```

Road Map

Who is going to implement and verify each feature or functionality you must complete

Jiamao Xu: L2 cache system, Basic Hardware Prefetching

Zihan Hu: 4-way set associative cache, Parameterized cache

Jerry Wang: Victim Cache, Local Branch History Table

What are those features or functionalities

L2 cache system: L2 cache is a type of memory cache to improve our system performance. Since large L1 caches will form a large critical path, we need a L2 cache to prevent such issues

Basic Hardware Prefetching: The OBL (One block lookahead) prefetch is a sequential prefetching technique that utilizes spatial locality and is straightforward to execute. This method triggers a prefetch for line i+1 upon accessing line i and encountering a cache miss. However, if line i+1 is already present in the cache, no memory access is initiated.

4-way set associative cache(may be 8-way set associative cache): we will implement a 4-way or 8-way set associative cache with pseudo-LRU replacement policy to make our cache system more efficient and have large cache size.

Parameterized cache: Parameterized cache is a cache memory design that allows for flexibility in its configuration by allowing users to specify the cache parameters such as cache size, block size, and associativity.

Victim Cache: A victim cache is a small, fully associative cache that stores blocks that have been evicted from a larger, more frequently accessed cache, known as the primary cache. When a cache miss occurs in the primary cache, the block that is evicted from the cache is stored in the victim cache.

Local Branch History Table: The 2-bit local branch history table is a component of a branch predictor used in computer processors to improve performance. It stores information about the outcome of recent branch instructions and is used to predict whether a branch is likely to be taken or not taken in the future.