Progress Report

Who worked on each part of the design

Jiamao Xu: control rom, datapath, ALU register, BR_en registers

Garen Hu: control rom, datapath, PC registers, ALU mux register, ctrl register

Jerry Wang: control rom, datapath, imm register, data register, pc sel register

The functionalities you implemented

Basic pipeline with components: control rom, datapath, PC, ALU unit, CMP unit, regfile, pc mux, regfile mux, alu mux, cmp mux. Can handle all of the RV32I instructions (with the exception of FENCE*, ECALL, EBREAK, and CSRR instructions).

The testing strategy you used to verify these functionalities

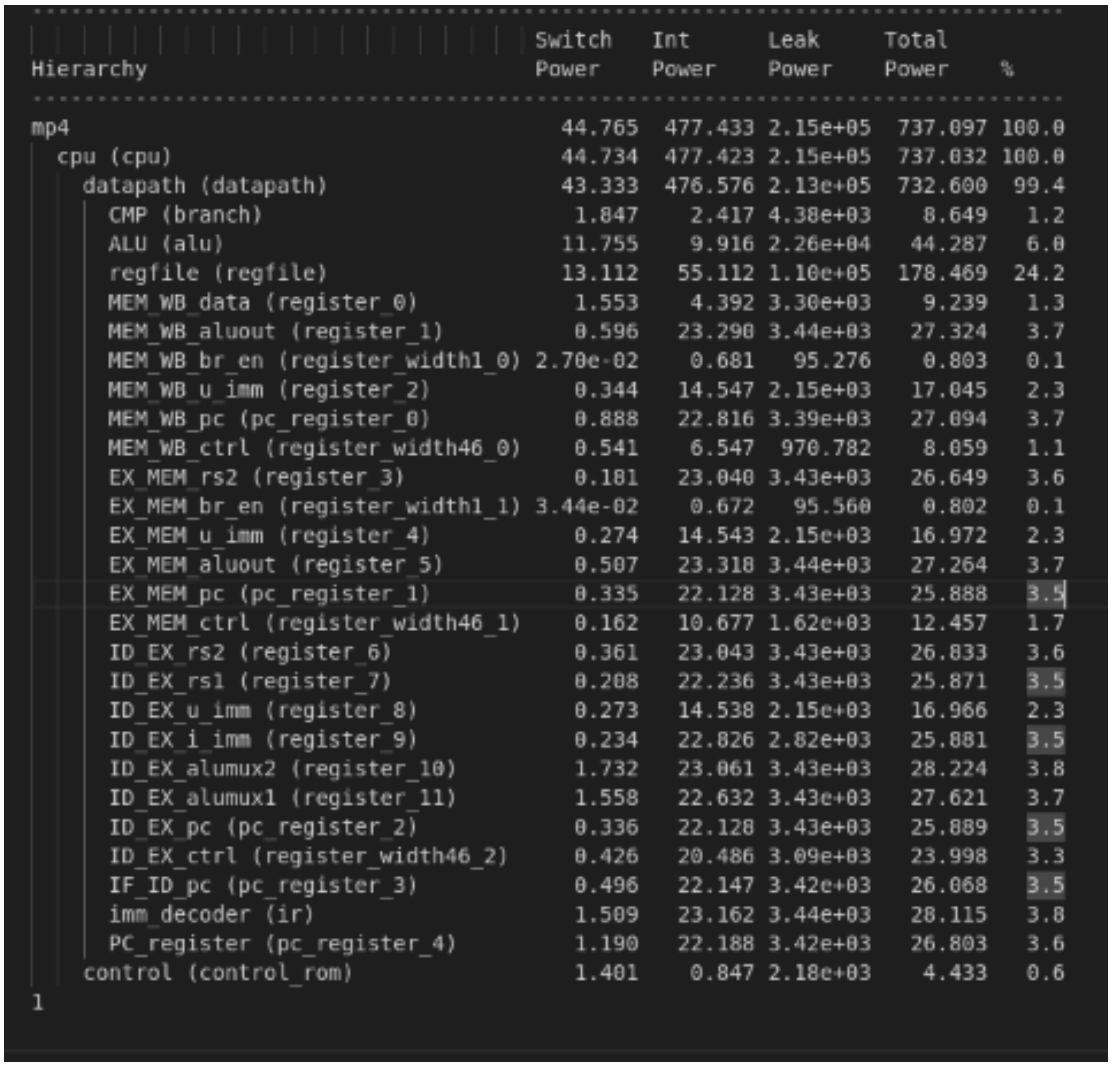
First, we run the provided test code, and we use Verdi to verify the mem address and the corresponding register value. If we find something goes wrong. We find the mem address and we manually change the halting condition to a few lines after the problem line. For example, if the problem occurs at memory address 80000198, we set up halting at 800001a0 to prevent execute code after it. And then we start with the instruction data and instruction address and trace the data along the datapath to see where is the problem

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                                                    -3.79
--------------------------------------------------------------------------
slack (MET)                                                           6.07
```

f_max=1/(10-slack)*1000=254.453

Power report:



Hierarchy	Switch Power	Int Power	Leak Power	Total Power	%
mp4	44.765	477.433	2.15e+05	737.097	100.0
cpu (cpu)	44.734	477.423	2.15e+05	737.032	100.0
datapath (datapath)	43.333	476.576	2.13e+05	732.600	99.4
CMP (branch)	1.847	2.417	4.38e+03	8.649	1.2
ALU (alu)	11.755	9.916	2.26e+04	44.287	6.0
regfile (regfile)	13.112	55.112	1.10e+05	178.469	24.2
MEM_WB_data (register_0)	1.553	4.392	3.30e+03	9.239	1.3
MEM_WB_aluout (register_1)	0.596	23.290	3.44e+03	27.324	3.7
MEM_WB_br_en (register_width1_0)	2.70e-02	0.681	95.276	0.803	0.1
MEM_WB_u_imm (register_2)	0.344	14.547	2.15e+03	17.045	2.3
MEM_WB_pc (pc_register_0)	0.888	22.816	3.39e+03	27.094	3.7
MEM_WB_ctrl (register_width46_0)	0.541	6.547	970.782	8.059	1.1
EX_MEM_rs2 (register_3)	0.181	23.040	3.43e+03	26.649	3.6
EX_MEM_br_en (register_width1_1)	3.44e-02	0.672	95.560	0.802	0.1
EX_MEM_u_imm (register_4)	0.274	14.543	2.15e+03	16.972	2.3
EX_MEM_aluout (register_5)	0.507	23.318	3.44e+03	27.264	3.7
EX_MEM_pc (pc_register_1)	0.335	22.128	3.43e+03	25.888	3.5
EX_MEM_ctrl (register_width46_1)	0.162	10.677	1.62e+03	12.457	1.7
ID_EX_rs2 (register_6)	0.361	23.043	3.43e+03	26.833	3.6
ID_EX_rs1 (register_7)	0.208	22.236	3.43e+03	25.871	3.5
ID_EX_u_imm (register_8)	0.273	14.538	2.15e+03	16.966	2.3
ID_EX_i_imm (register_9)	0.234	22.826	2.82e+03	25.881	3.5
ID_EX_alumux2 (register_10)	1.732	23.061	3.43e+03	28.224	3.8
ID_EX_alumux1 (register_11)	1.558	22.632	3.43e+03	27.621	3.7
ID_EX_pc (pc_register_2)	0.336	22.128	3.43e+03	25.889	3.5
ID_EX_ctrl (register_width46_2)	0.426	20.486	3.09e+03	23.998	3.3
IF_ID_pc (pc_register_3)	0.496	22.147	3.42e+03	26.068	3.5
imm_decoder (ir)	1.509	23.162	3.44e+03	28.115	3.8
PC_register (pc_register_4)	1.190	22.188	3.42e+03	26.803	3.6
control (control_rom)	1.401	0.847	2.18e+03	4.433	0.6

1

Road Map

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

Jiamao Xu: Hazard Detection, arbiter

Zihan Hu: Forwarding Unit, arbiter

Jerry Wang: Static-not-taken branch prediction, arbiter

What are those features or functionalities

Hazard detection is used to detect data hazard which occur when an instruction depends on the result of previous instruction and that result of instruction has not yet been computed. The Forwarding Unit is the solution to the data hazard. The idea of the forwarding unit is to pass the ALU value to alu mux directly rather than wait for the write back stage. Static-not-taken branch prediction is used to predict that the branch is always not taken. The arbiter is used to determine which cache can access the physical memory since the physical memory is single port.