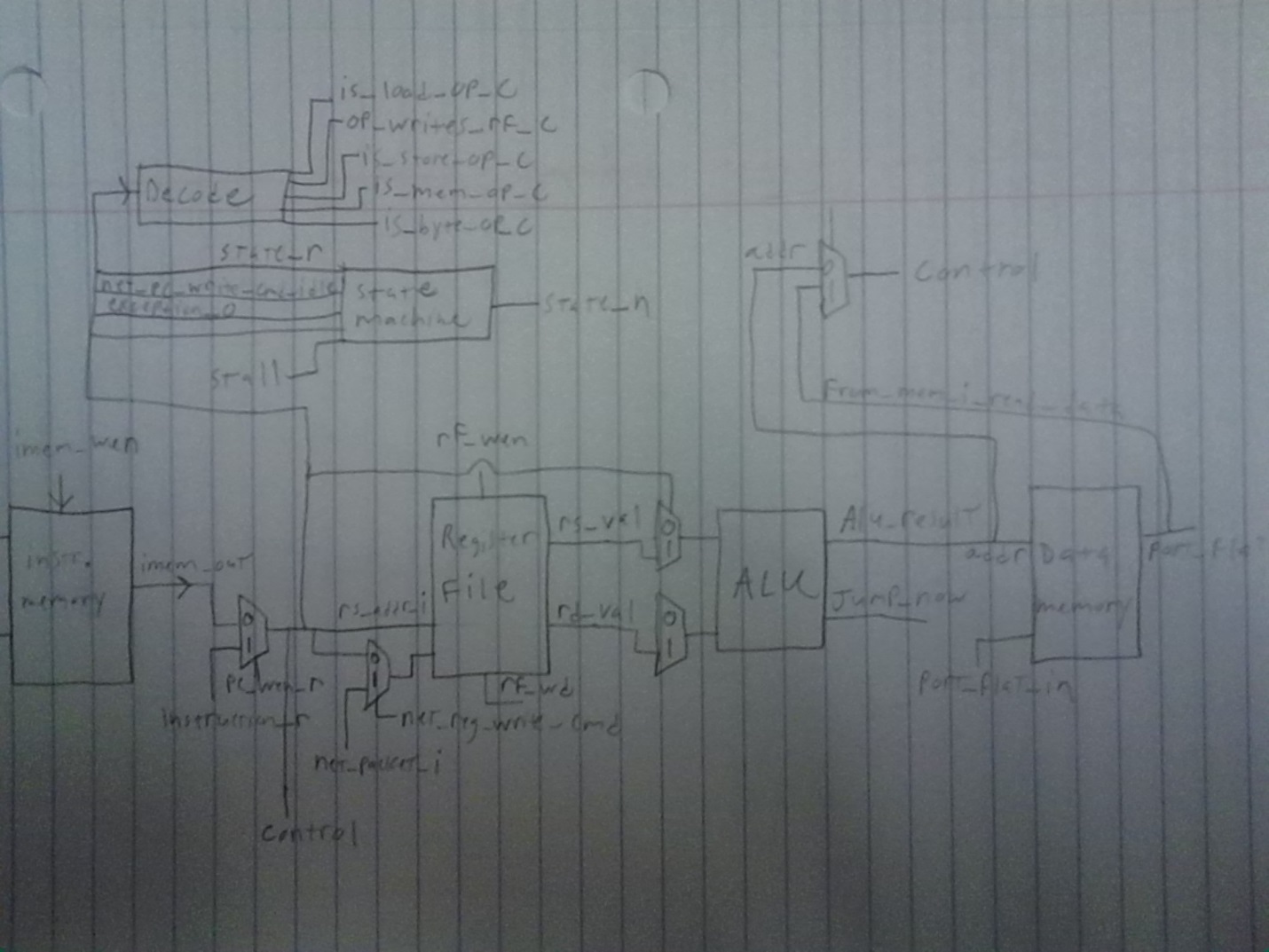
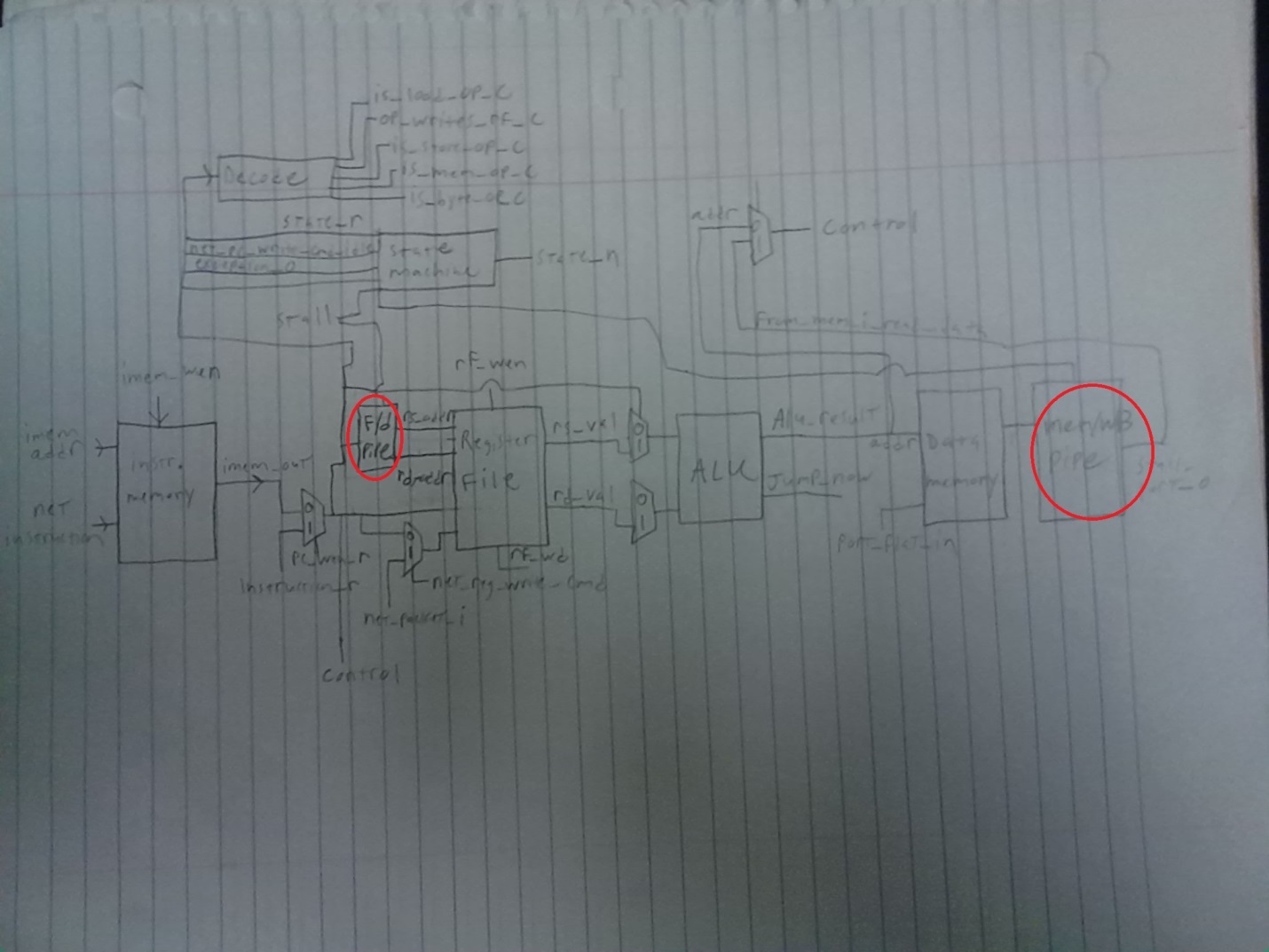
Q1) optimized lab2 miner baseline  
**fMax** = 54.78 MHz  
  
**Cycle Time**: 1 / fMax = 0.01825 microseconds  
**Cycle Count**: 160608  
**Instruction Count**: 152828  
**CPI**: [(160608 cycles) / (152828 instructions)] = 1.0509 average cycles per instruction  
**Absolute Time**: 0.01825 \* 160608 = 2931.906 microseconds = 2.931906 milliseconds  
  


Q2)  
a) pipe locations are circled in red  


b) A pipeline with more stages will be able to handle more instructions at a time. Unfortunately if a finished instruction relies on an instruction that is currently in the pipeline, the core must stall and wait for that instruction to complete. More stages will magnify this delay, less stages will alleviate it. More stages also increases the number of flip-flops amongst the modules, which increase cycle count. Also when a program branches, the entire pipeline must be cleared. Overall, the desired effect of pipelining is the increase on clock frequency.

Q3)  
Within the cl\_decode.sv file, the 5 instructions are shown. They are load, write, store, memory, and byte operations. Each always block shows their associated signals. Below is a chart that shows the signals that are relevant to each operation as well as their state at each stage in the pipeline.

For a particular instruction, the current operation being performed is determined. That operation is set to 1, all other operations are set to 0.

|  |  |
| --- | --- |
| is\_load\_op\_o | If instruction is (kLW, kLBU), then return 1, else return 0. |
| op\_writes\_rf\_o | If instruction is (kADDU, kSUBU, kSLLV, kSRAV, kSRLV, kAND, kOR, kNOR, kSLT, kSLTU, kMOV, kJALR, kLW, kLBU, kBLR, kXOR, kROR), then return 1, else return 0. |
| is\_store\_op\_o | If instruction is (kLW, kLBU, kSW, kSB), then return 1, else return 0. |
| is\_mem\_op\_o | If instruction is (kSW, kSB), then return 1, else return 0. |
| is\_byte\_op\_o | If instruction is (kLBU, kSB), then return 1, else return 0. |

Q4) Hazards  
  
**Data**: There is one data hazard  
1) Mathematical operations that rely on each other will cause the pipeline to stall. For example, if a math operation requires the results of an operation that is still in the pipeline and has yet to be executed.  
  
**Control**: There are two control hazards.  
1) If the system is not in the run state, it will stall.  
2) Branches and jumps will cause the pipeline to stall.  
  
**Structural**: There are four structural hazards.  
1) If an instruction is currently being decoded, the pipeline will stall.  
2) If the register file is currently being written to, the pipeline will stall.  
3) If the network writes to IMEM, the pipeline will stall.  
4) If the network is writing to the register, the pipeline will stall.

Q5) using NOPS to create the 3-stage core was optional. Decided to skip and instead just implement the pipeline.

Q6) done in Q2.

Q7) Lab-3 3-stage pipelined miner  
**fMax** = 83.61 MHz  
  
**Cycle Time**: 1 / fMax = 0.01196 microseconds  
**Cycle Count**: 185402  
**Instruction Count**: 152828  
**CPI**: [(185402 cycles) / (152828 instructions)] = 1.2131 average cycles per instruction  
**Absolute Time**: 0.01196 \* 185402 = 2217.40792 microseconds = 2.21740792 milliseconds

**Speedup of Lab-3 pipelined core versus Lab-2 non-pipelined optimized core  
83.61 MHz / 54.78 MHz = 1.526 speedup factor**