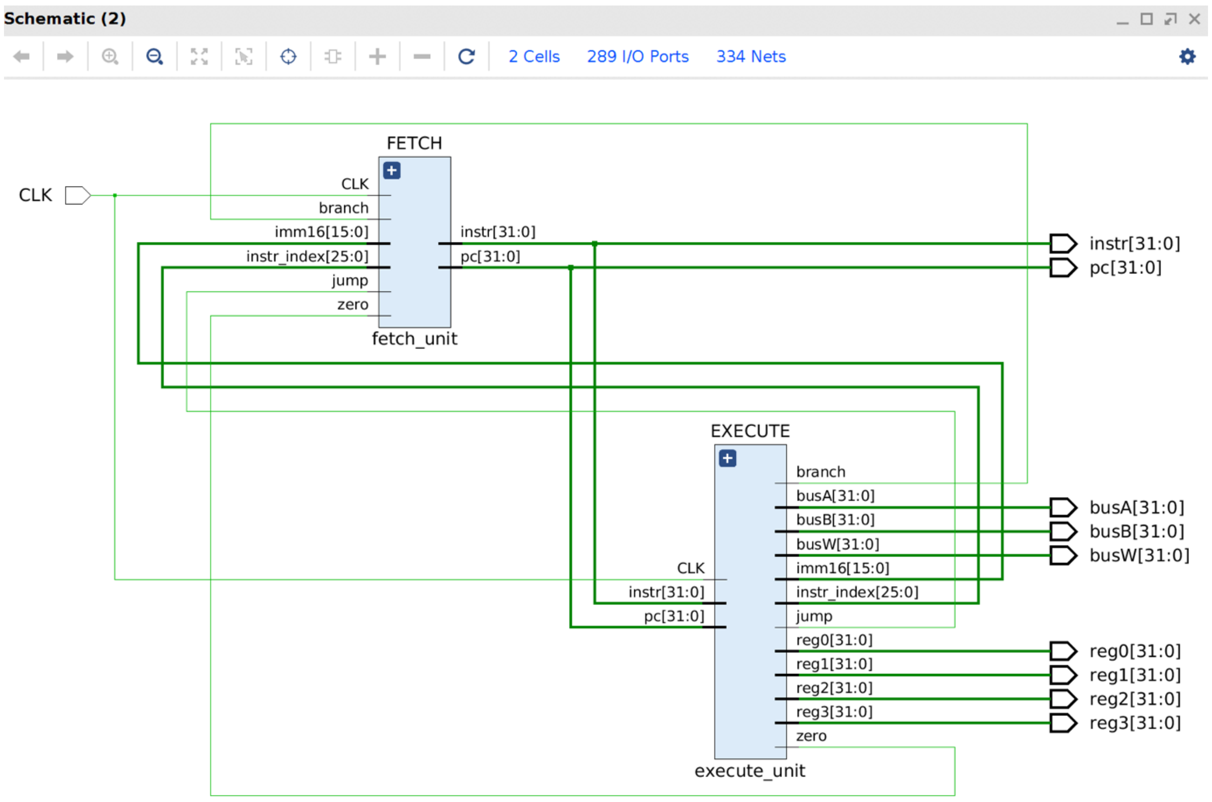
PROCESSOR DESIGN

Ahish Deshpande

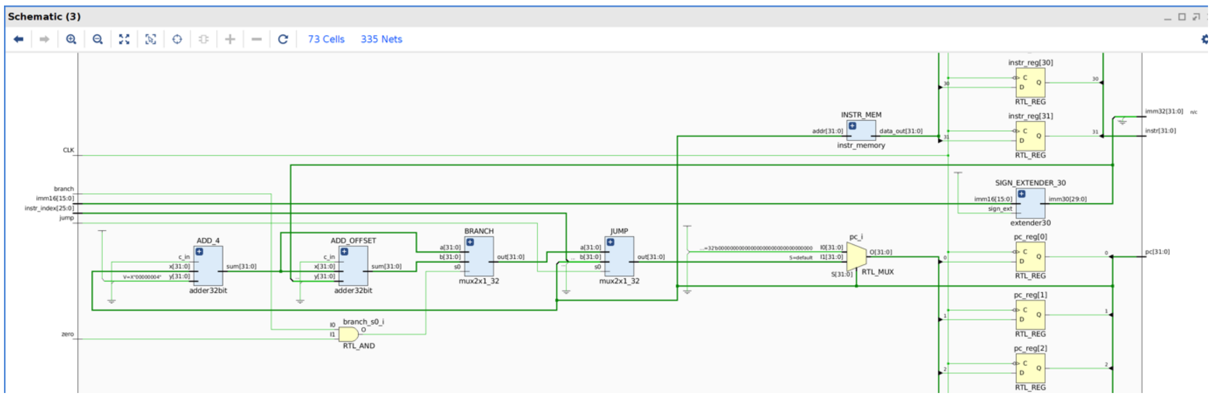
2018102022
ECE UG2k18

Architectural Diagram

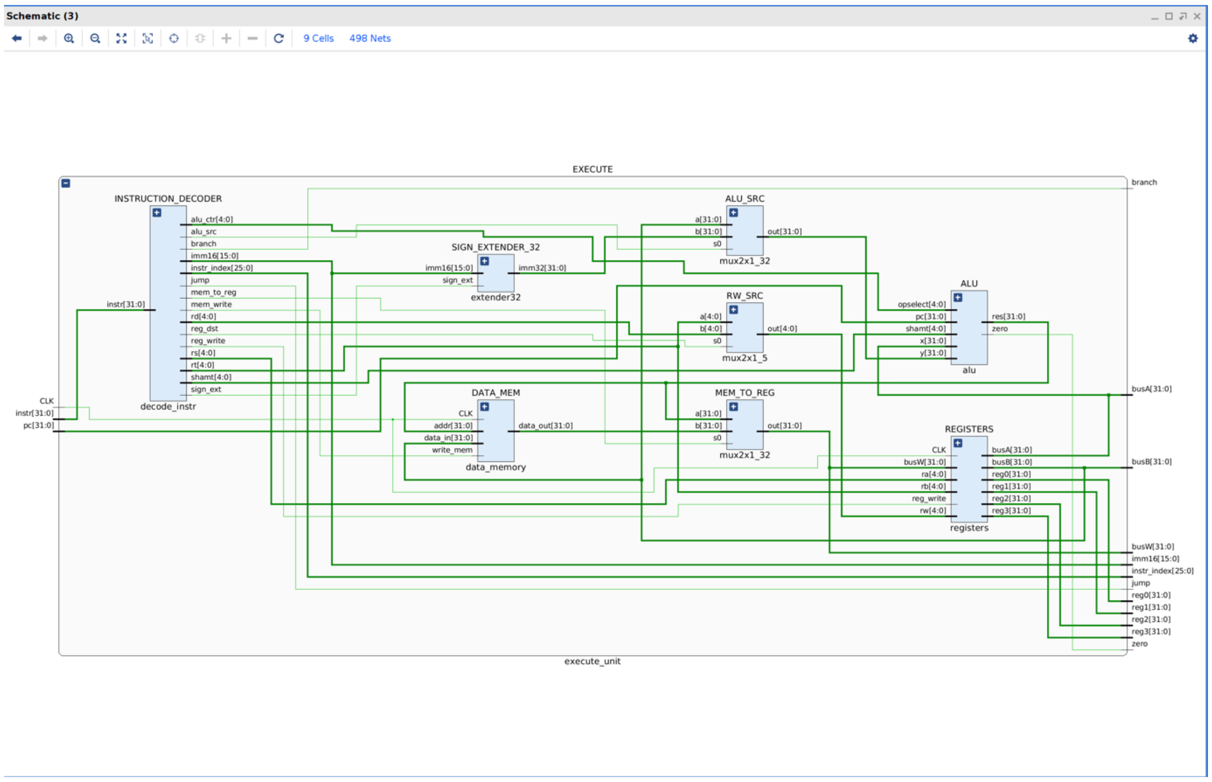
Overview of the processor:



Fetch block:



Execute block:



Main Memory Design

The main memory is currently of size 4096 bytes. It is an array of 4096 8-bit vectors. Information is stored in the Big Endian byte order. The entire processor uses the Harvard architecture, ie. the information memory and the data memory are separate. In smaller processors, this makes it easy to code, and reduces the need to separate instructions from data manually in one single block. The main memory is capable of loading a single byte(with the LB command), and storing a single byte(with the SB command). However, as the ALU output is 32-bit, only the 8 least significant bits are stored in the memory. Similarly, as the data out path from the main memory is 32-bit, 0's are appended to each byte.
The memory also supports LW and SW instructions, however both LW, LB and SW, SB will not run together. The code needs to be modified before running depending on which version is required.

Instructions Supported

The instructions supported are:

- R type:
  - ADD
  - AND
  - NOR
  - OR
  - SUB

    - XOR
    - SLT
    - SRA
    - SRAV
    - SRL
    - SRLV
    - SLL
    - SLTU
- I type:
    - ADDI
    - ANDI
    - XORI
    - ORI
    - SLTI
    - SLTIU
    - BEQ
    - BGTZ
    - BLEZ
    - BNE
    - LB
    - SB
- J type:
    - J
    - JAL

All the above instructions have been tested using the testbench written for the code.

Instructions Not Supported

All instructions mentioned in the document sent by the TA's are supported, however the remaining instructions given in the MIPS ISA are not supported.

Processor Clock Frequency

The minimum time period of the processor is 8ns. Thus, the clock frequency 1.25GHz(estimate).

Interesting Things

- The code is extremely modular and hence easy to debug and add features too
- During implementation, I realized that initializing the values of the different parameters could be difficult, as there are multiple 'feedback' loops in the circuit, with certain elements such as branch, zero, jump, etc. not having any initial values(as they are just wires). This was fixed by letting them be DON'T CARE for the first clock cycle, and not using them to load the second value of the PC, ie. 4. Thus, the PC always goes to 4 from 0(hardcoded). Initially, I thought that this would be a problem,

as I thought branch and jump instructions could not be placed at instruction 0, however, I realized that this is not true, since there would be one branch delay slot that would force the counter to reach value 4 anyways.