IPA Project Report

Team ATmicro

Things we have implemented so far

1. Fetch block

    - It contains a 1 KB instruction memory block.
    - The fetch block takes 10 bytes of data from the instruction memory and builds a 64 bit instruction.
    - The icode, ifun, rA, rB, valC and valP values are then extracted from this instruction and given as outputs to the later stages of the processor.
    - At each positive edge of the clock, a new instruction is generated based on the value of the PC.
    - This new instruction is then used to obtain the required output variables by splitting it into appropriate lengths.
    - This block also outputs 2 status variables namely instruction valid (instr_valid) and memory error (imem_error) which denote the proper/improper working of the fetch stage.

2. Decode and Writeback Block

    - It contains an array of 15 registers which form the register memory.
      (out of these 15 registers we have specified the 4^{th} one (%rsp) as the stack pointer register)
    - At each positive clock edge, the decode block takes icode, rA and rB and accesses the register memory using these values in order to generate actual numerical values valA and valB and store them in registers for use in successive stages.
    - At each negative clock edge, the writeback block takes valE (output of execute block), valM (output of memory block) and icode and updates the values stored in the register memory using the input values.
    - In case of call, ret, pushq and popq instructions, the writeback block updates the stack pointer to point to another location based on the input values.
    - We have implemented Decode and Writeback blocks in the same Verilog module because both of them share a common register memory and hence can access it easily if they are defined inside the same module.

3. Execute Block

    - It contains the 64 bit ALU which was coded earlier.
    - It takes icode, ifun, valA and valB as inputs and returns valE as the result of the requested operation.

- The execute block also evaluates 3 condition code flags namely ZF (zero flag), SF (sign flag) and OF (overflow flag) which are used to implement conditional instructions like conditional move and conditional jump.
- The execute block also combines the values stored in valB and valC in order to perform memory based operations.
- It also evaluates a variable 'jmpcnd' which specifies whether a jump is to be taken or not based on the conditional codes evaluated earlier.
- In case of call, ret, pushq and popq instructions, the execute block updates the value of valB by either incrementing/decrementing it by 8, hence giving the location of the new stack pointer.

4. Memory Block

    - It contains a 8 KB data memory block.
    - At each positive edge of the clock, it takes icode, valE and valA and based on their value, it either updates the data memory, or reads a value from the data memory and stores it in a variable valM which is used for successive stages.

5. PC Update block

    - At each positive edge of the clock, this block takes values like icode, valP, valC, valM and jmpcnd, generated by the previous stages of the processor and uses them to update the value of the program counter.
    - This step makes the processor ready for the next cycle of execution.

We have implemented the above mentioned block independently and tested them with dedicated testbenches.

Objectives we plan to accomplish

We plan to accomplish the following objectives in the coming time

1. Integrate all the independent blocks to finish the sequential implementation of the processor.
2. Add 5-stage pipelining in order to improve upon performance