

**EE577A FINAL PROJECT** - Phase 1

**Design of a General Purpose CPU**

Youngseok Lee - 4930239194

Narayana Reddy Lekkala - 9623274062

Chirag Ahuja - 5920609598

Submitted on Apr/10/2015



1. Introduction

CPU, central processing unit, is a core circuitry for computers to process its information. In this project, a general purpose pipelined microprocessor has been designed to support simple instructions such as arithmetic/logical operations, Bitwise operation Store, and Load Word. The CPU is designed to have 5 stage pipelines so that every instruction is divided into 5 separate stages in order to increase the instruction throughouts, the number of instructions that can be executed in a unit of time.[[1]](#footnote-0)

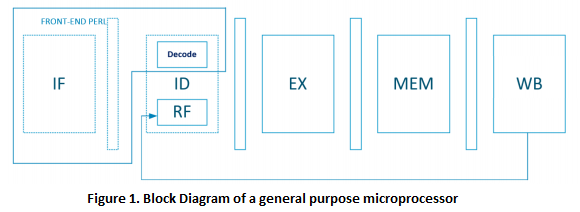


Figure 1 shows the overall picture of the proposed CPU design. This consists of two main parts; the software and hardware part.

Given each instruction format, the Perl script fetches and decodes the commanded instruction by generating a vector file. Then the vector file is put as inputs for Register Files(RF) in schematics. Based on the instruction, the function pipelined per each stage will be performed. In Execution stage, Arithmetic Logic Unit(ALU) performs simple operations such as AND, OR, XOR, ADD, and MUL. 512-bit SRAM is used in memory part supporting STORE and LOAD instructions. The Write-Back stage literally performs write the processed data into RF.

1. Software - Perl

* Perl generating vector file(front-end)
* Data dependency
* Perl scripting results(back-end)

1. Hardware - Schematics

* Overview

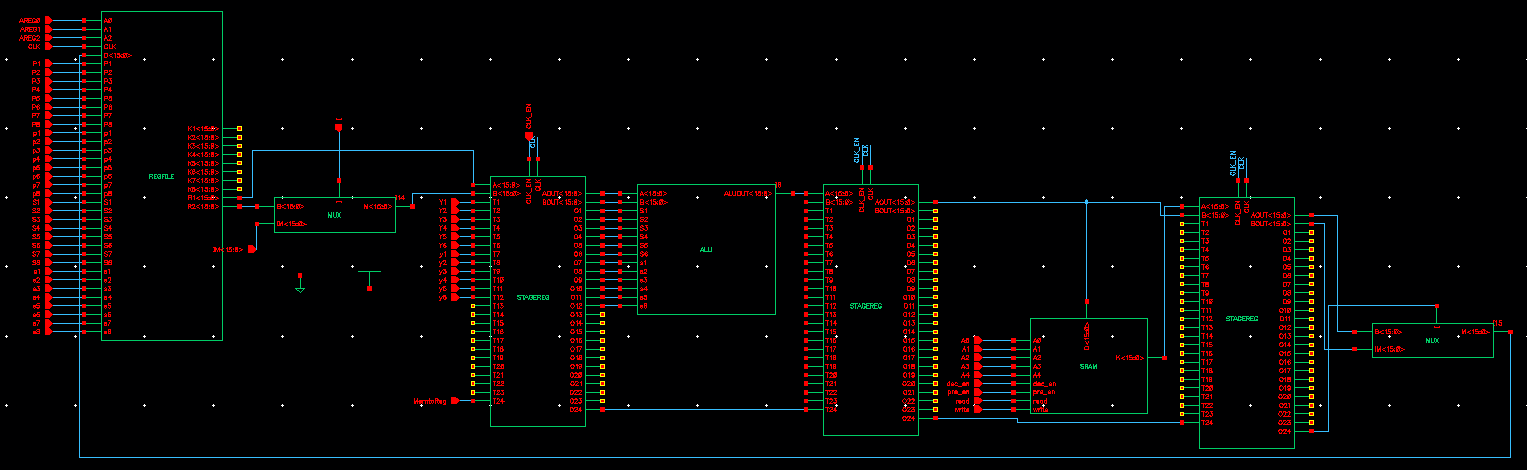
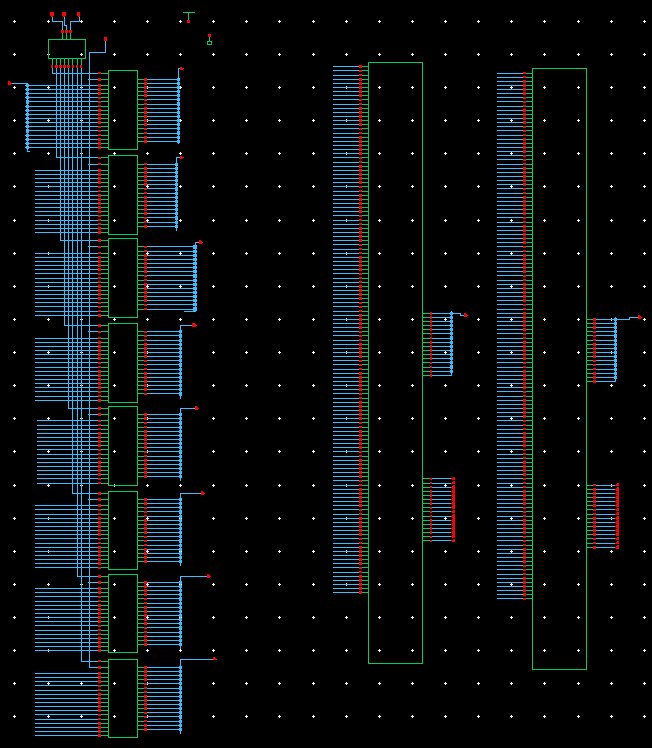
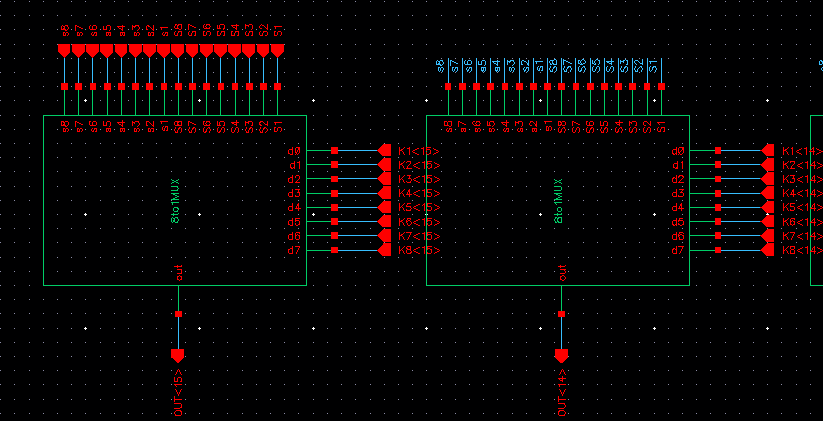
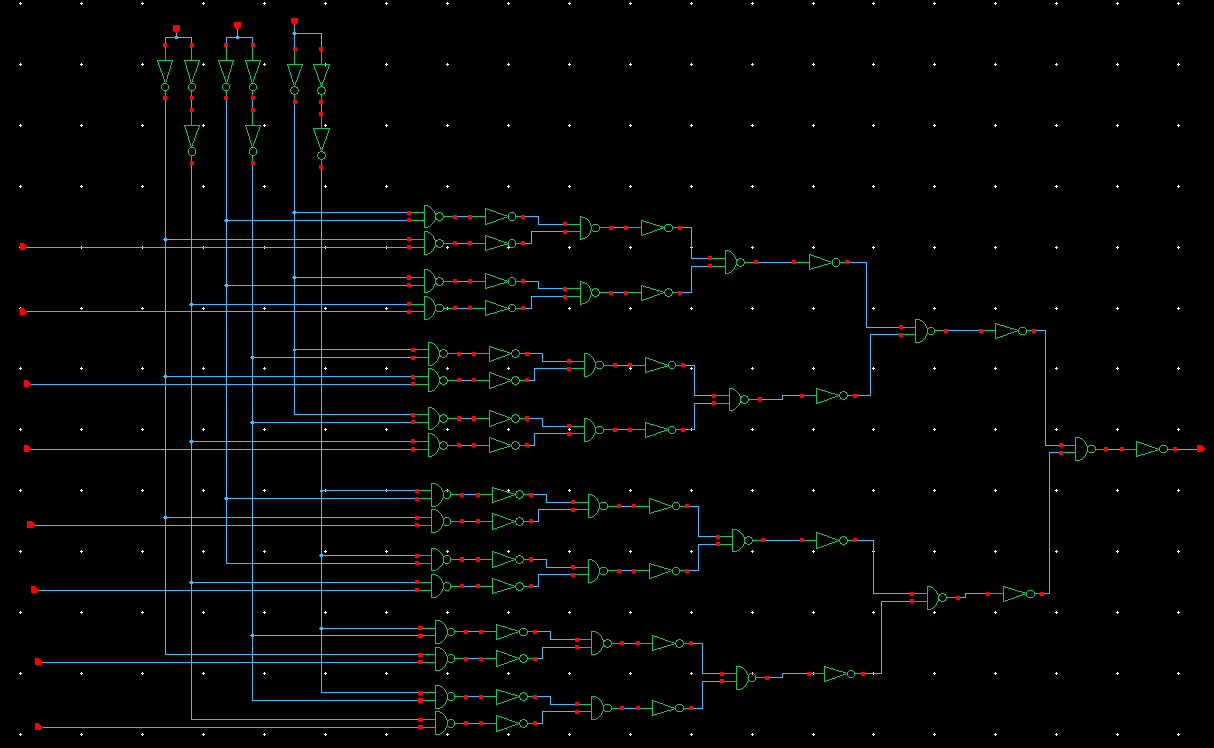


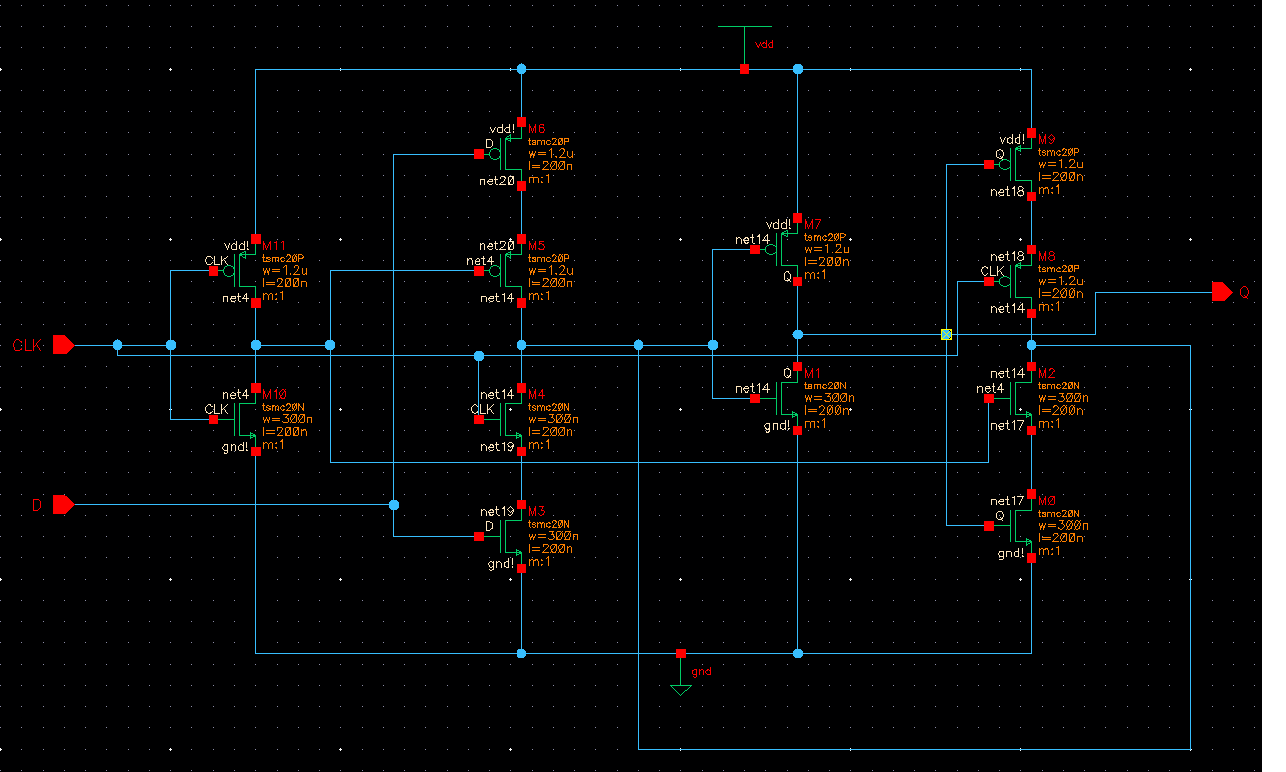
Figure 2 is the overview schematic design of the microprocessor. The first block is register file(RF) with multiple input pins where its data are fed by a Perl-generated vector file. The working principle is discussed in introduction. The time that takes per each cycle depends on the maximum clock value of any stage because the global clock should meet the minimum time for all circuitry.

* Register File



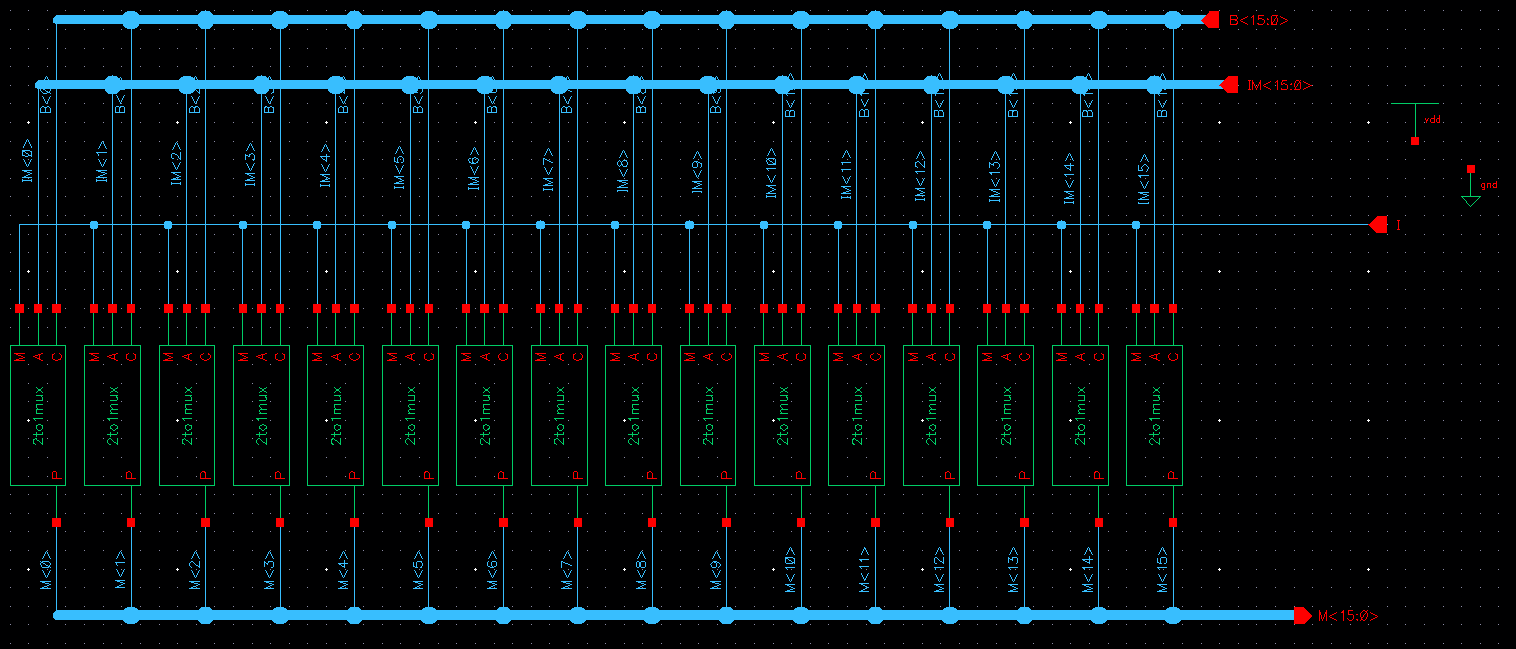






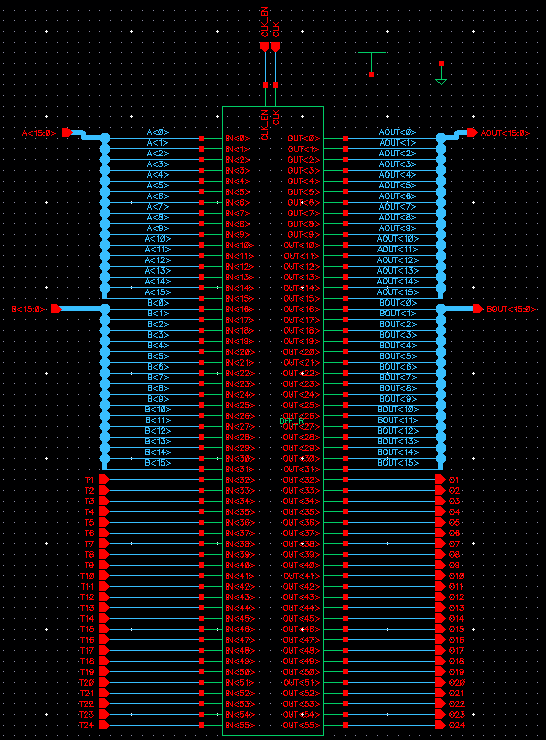
Register Files are 8 16-bit registers that are series of consecutive D flip flops with mux select lines. Upon loading any registers specified by the address bit, the 16-bit values will directly be inputted into stage registers for ALU. Figure X is the D-Flipflop.

* 6 to 1 MUX



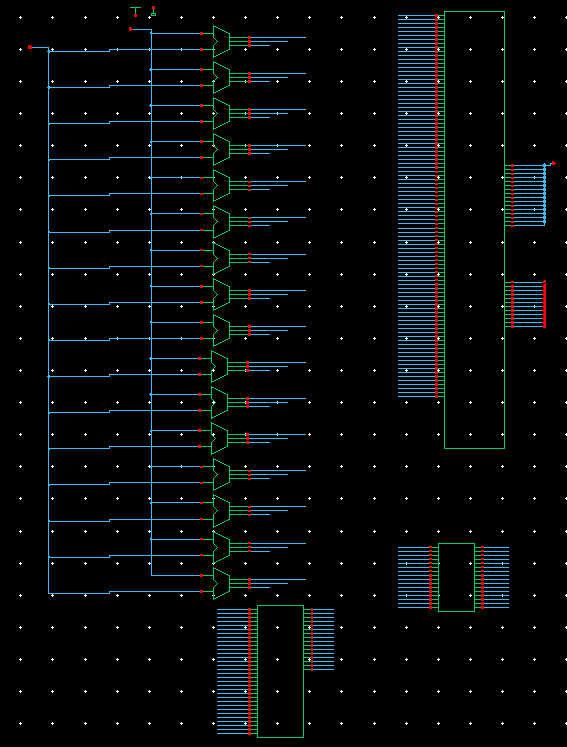
In case of the data comes directly from the perl scripting file, immediate 16-bit values are written into the stage registers of ALU throughout the 6-to-1 MUX designed in Figure X.

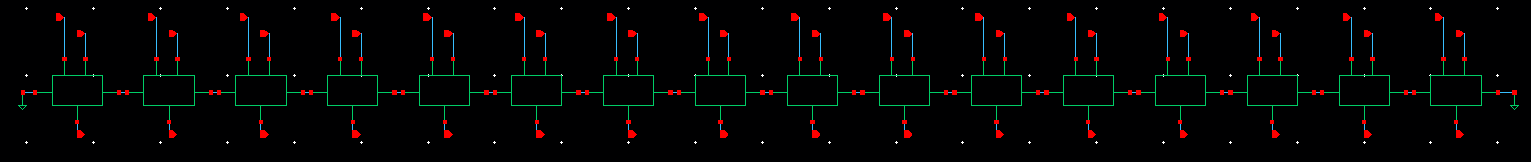
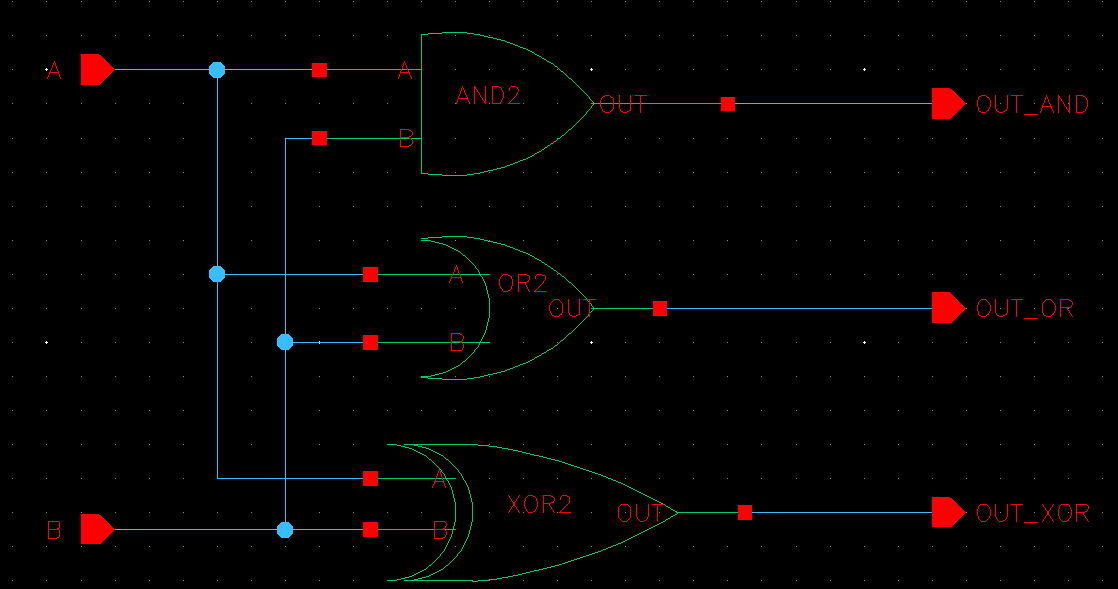
* Stage Register

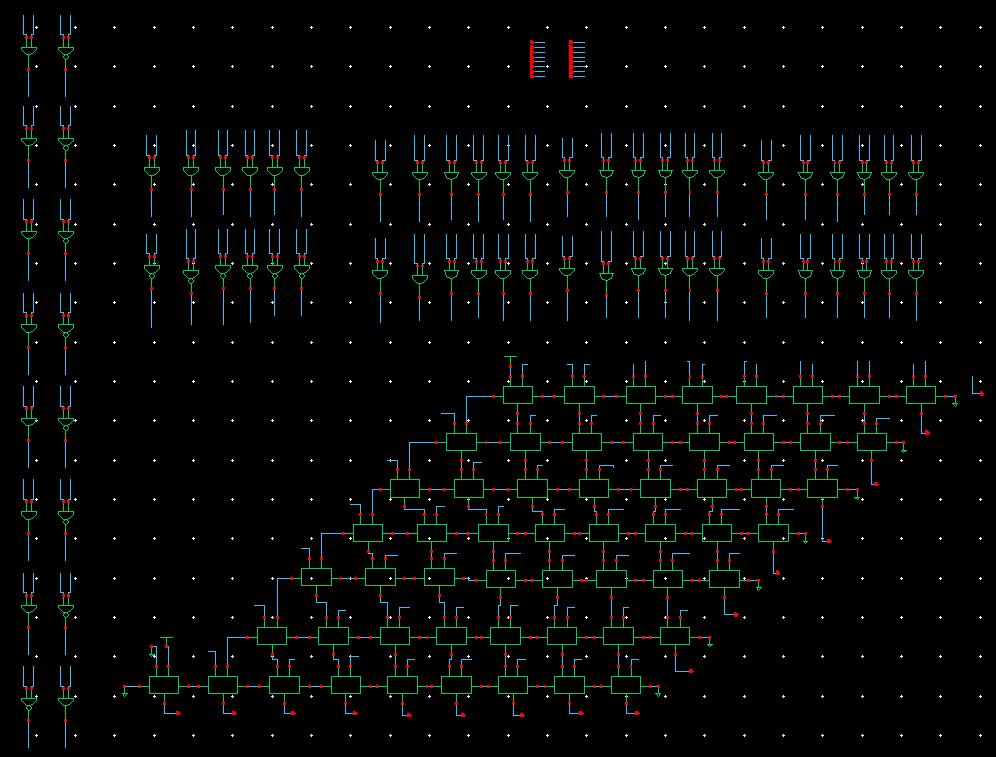


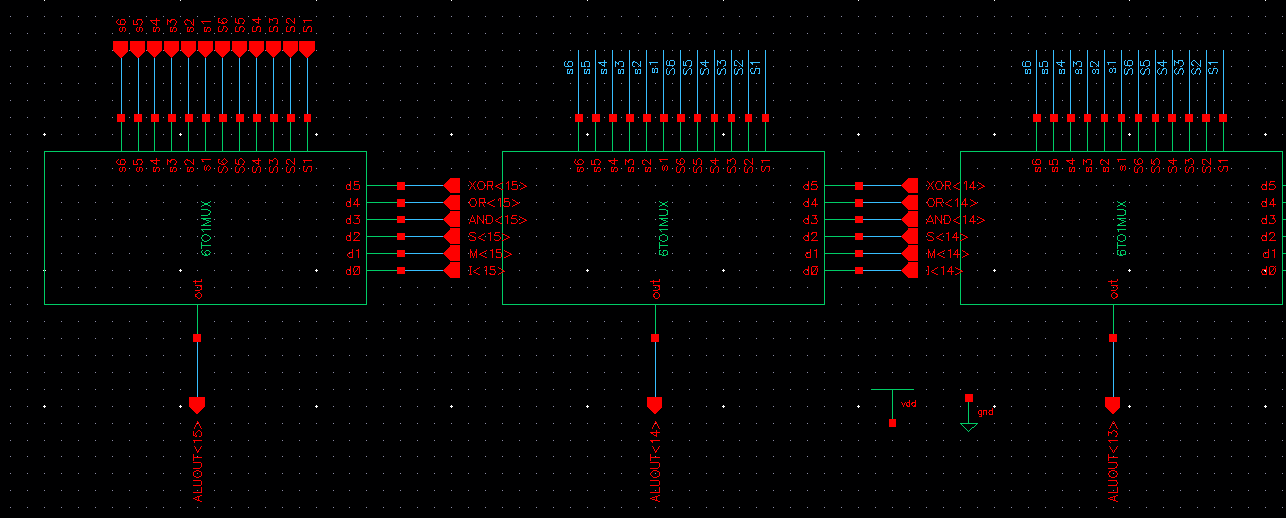
There are 3 stage registers whose capacities are 56 bits. The purpose of having the stage registers is to keep the input values from instruction(32-bits) and the address bit values(3-bits) that indicate where the processed output should be written. Freedom of 20 bits are added in order to specify the kind of instruction, the control signal of mux, and sram signals.

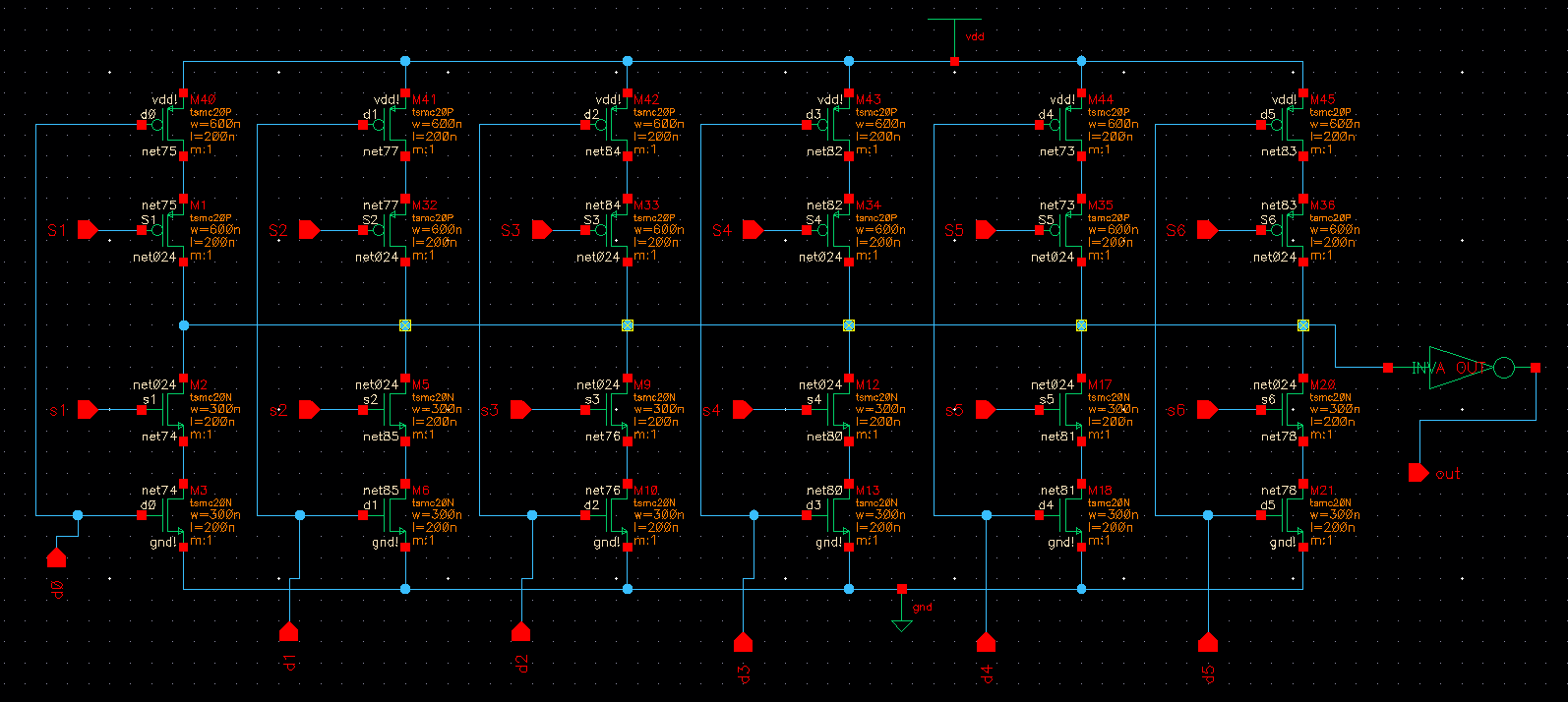
* ALU







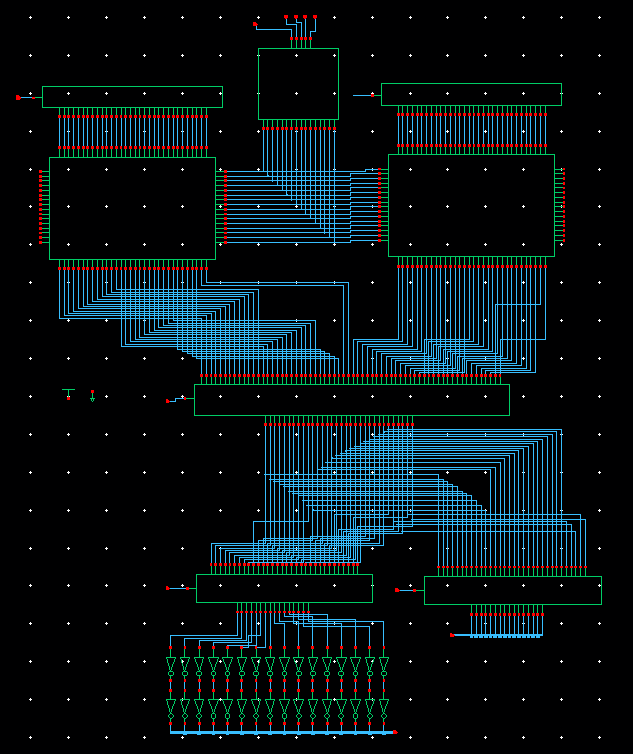




The designed ALU supports instructions such as bitwise AND, OR, XOR, ADD, and MUL with values from RF or/and immediate perl coding.

The three basic instructions AND, OR, and XOR are combined into one block. Later, this logic will be clock-gated to save power. 32-bit Carry Ripple Adder has been designed for adder. 2’s Compliment 8-bit Multiplier has been designed for multiplication. All arithmetic/logic results are inserted into ALU MUX, which only outputs the desired instruction results.

* SRAM



Detailed description of SRAM is discussed in Lab2.

1. Function Verification

* Perl verification (later)
* Waveform verification with vector file

1. Plan for phase 2

* Dynamic logic applied for max performance

In class, the techniques of delay optimization using dynamic logic are discussed. The designed CPU has a delay optimized components for ALU MUX. The adder will be optimized using the domino dynamic logic, and the multiplier will be optimized by inserting intermediate registers in the middle of the schematic for optimizing throughouts.

* DFF optimization and power optimization

DFF arrays are a large part of this project as it is widely used for RF, stage registers, and optimized ALU. Clock-gate technique for DFF will reduce the power consumption by reducing the switching activity in CMOS logic circuits.[[2]](#footnote-1)

Different kinds of DFF such as TGFFs, pulse triggered FFs, and other FF structures will be tested for maximum speed\*power\*performance product.

* Back-end perl scripting for verification

The execution results of each instruction will be verified using a Perl script. The script will compare the expected values and the simulation results for different instructions.

* layout for schematics

Phase 2 submission includes optimization and layout part. Customized floor plans for each modules will be proposed for minimum layout area.

* demo preparation(May 4th)

1. Challenges

* SRAM for slow clock

SRAM schematics itself run in a clock period of 2ns when tested individually. With the other blocks combined, SRAM only works in 6ns clock. Further investigation will be performed.

1. Appendix
2. Miscellaneous

1. Instruction pipeline - Wikipedia [↑](#footnote-ref-0)
2. Power estimation and optimization(Unit 13) - Shahin EE577A [↑](#footnote-ref-1)