Extended MIPS Processor

Marty Townley and Adam Ness

ECE425

Spring 2017

Introduction

Throughout the course of the semester, we having been working with the Harvey Mudd MIPS processor. For this project, we added additional functionality to this processor by altering specific parts of the given controller and datapath. Through this work we were able to add XOR and NOR instruction ability, as well as several shift operations, namely SLL, SRL, and SRA.

System Description

Features Added

Extended ALU

In order to execute all bitwise operations in the MIPS instruction set, the original ALU had to be extended to handle XOR and NOR operations. This was done by removing several logic gates prior to the Full Adder, and replacing them with a 4-1 multiplexer. This expanded the number of ALU control signals to seven. These changes were made on a slice level, then using Magic “array” commands to form the eight ALU bitslices into the full ALU used within the MIPS datapath. The changes to the ALU control signals then required a rework of the ALU decoder, which was later included in the Control Unit (discussed in Section IV).

Funnel Shifter

Similar to the extension of the ALU, the existing datapath required additional hardware in order to handle shift operations. To facilitate this functionality, we added an Array Funnel Shifter below the existing datapath. The shifter designed allows for three additional instructions to be performed, Shift Left Logical (SLL), Shift Right Logical (SRL), as well as Shift Right Arithmetic (SRA). This new hardware required additional alterations to the Extended ALU, namely replacing the 3-1 “result” multiplexer, with a 4-1 multiplexer. This allowed the outputs of the shifter to be propagated through the remainder of the datapath as required.

Operation of ALU

Operation of the extended ALU was done in multiple stages. Simulation in IRSIM was done after the addition of the XOR and NOR commands, as well as after the replacement of the 3-1 multiplexer for the 4-1 multiplexer required by the Funnel Shifter. The original ALU was removed from the datapath, and replaced with our extended ALU, which was slightly wider, due to the changed multiplexer. Functionality was again verified once the finished ALU was placed within the MIPS datapath. The results of these tests may be found in the Extended ALU section of the Simulation Plots.

Operation of Funnel Shifter

The Funnel Shifter was exhaustively simulated using IRSIM as a standalone cell (sll, slr, sra operations). Placing the shifter below the datapath allowed us to maintain the 3000 lambda width requirement, however, this caused difficulties due to the necessity of running the control signals up through the datapath. It was simulated again once placed within the datapath, alongside the extended ALU. The results of these tests may be found in the Funnel Shifter section of the Simulation Plots.

Operation of Control Unit

The Control Unit used was generated from the controller.sv module that included the new ALU control signals for the extended ALU. The aspect ratio and control signals had to manipulated so that the controller would fit above the datapath, with signals routed down to facilitate ease of wiring to the datapath. The clock of the controller was tied to ph1 of the datapath. Correct operation of the Control Unit was confirmed using IRSIM, and executing add, sra, lb, sq, beq, and j. The results of these tests may be found in the Control Unit section of the Simulation Plots.

Operation of Overall Chip

The complete chip was simulated in IRSIM as well. The memfile.dat, which includes a MIPS program represented by eight 32-bit hexadecimal numbers was ran through the chip, as shown below:

```
.data:00000000 80 02 00 44    lb   v0,68(zero)       v0=5
.data:00000004 80 07 00 40    lb   a3,64(zero)       a3=3
.data:00000008 80 e3 00 45    lb   v1,69(a3)         v1=c
.data:0000000c 00 e2 20 25    or   a0,a3,v0          a0=7
.data:00000010 00 64 28 24    and  a1,v1,a0          a1=0100, 4
.data:00000014 00 a4 28 20    add  a1,a1,a0          a1=1011, 11
.data:00000018 10 a7 00 08    beq  a1,a3,0x0000003c  branch if a1=a3, doesnt
.data:0000001c 00 64 30 2a    slt  a2,v1,a0          a2 = v1 < a0 ? 0
.data:00000020 10 c0 00 01    beqz a2,0x00000028     not equal
.data:00000024 80 05 00 00    lb   a1,0(zero)        a1=68 (44h)
.data:00000028 00 e2 30 2a    slt  a2,a3,v0          a2= a3 < v0 ? 1
.data:0000002c 00 c5 38 20    add  a3,a2,a1          a3=a2+a1 = 1+68, 45h
.data:00000030 00 e2 38 22    sub  a3,a3,v0          a3=a3-v0=64, 40h
.data:00000034 08 00 00 0f    j    0x0000003c        jump to address 0x3c
.data:00000038 80 07 00 00    lb   a3,0(zero)        a3=44h
.data:0000003c a0 47 00 47    sb   a3,71(v0)         stores a3 into 4c
.data:00000040 00 00 00 03    sra  zero,zero,0x0
.data:00000044 00 00 00 05    0x5
.data:00000048 00 00 00 0c    syscall
```

The program in its entirety was not able to simulate successfully, however, we believe that we traced the issue back to a flip-flop within the datapath. The data register prior to the Register File did not correctly handle the bits fed to the 2-1 multiplexer on the write data port. The results of these tests may be found in the Overall Chip section of the Simulation Plots.

Conclusion

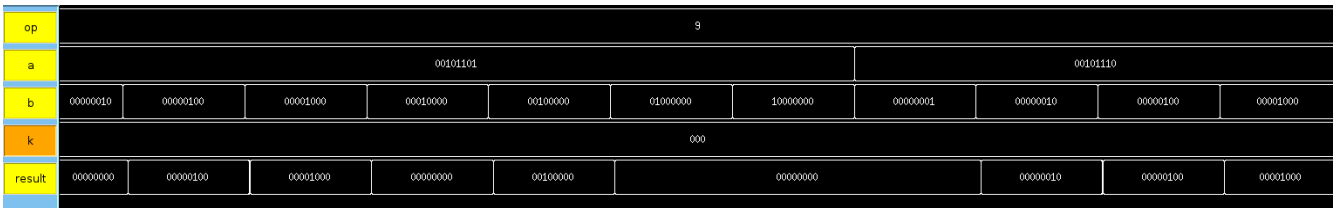
In summary, we came close to producing a functioning 8-bit MIPS processor. The alterations we made to add XOR, NOR, as well as, SLL, SRL, and SRA instruction ability were verified through individual simulation of the respective parts. The extended ALU, Array Funnel Shifter, and Control Unit were all simulated as standalone cells, then once again after placed within the Pad Frame. The final layout was checked for design errors, then simulated again using the memfile.dat MIPS program outlined above.

Simulation Plots

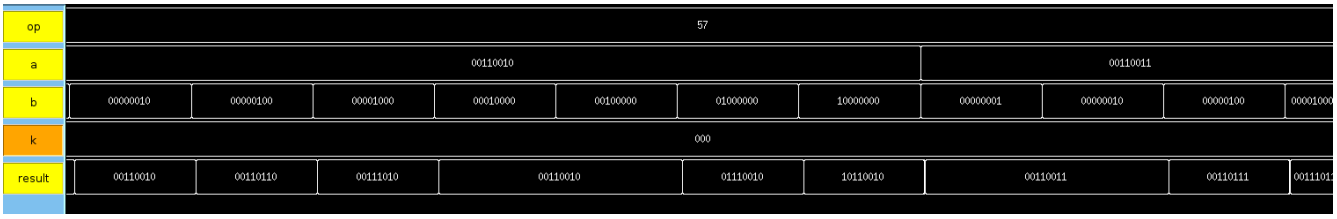
Extended ALU

The following ALU tests were done after standalone testing. The waveforms below correspond to the specified instruction type, within the datapath.

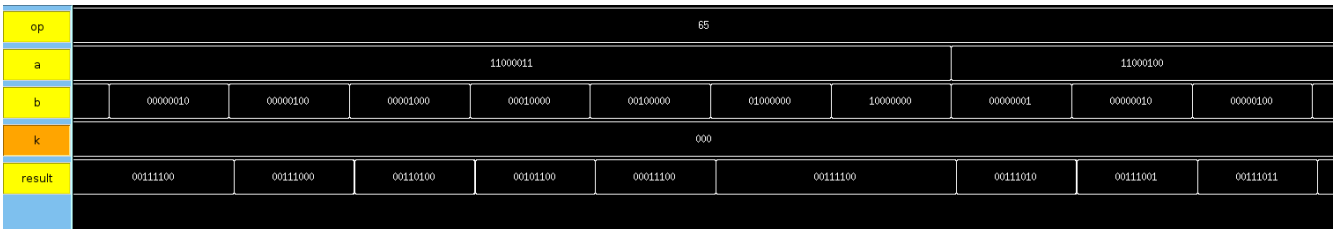
AND:



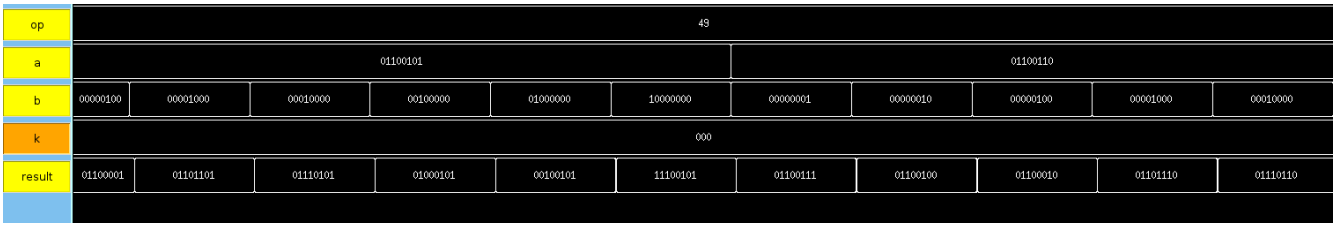
OR:



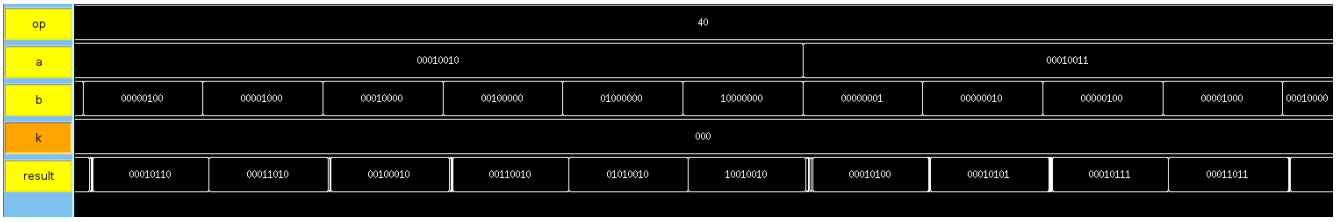
NOR:



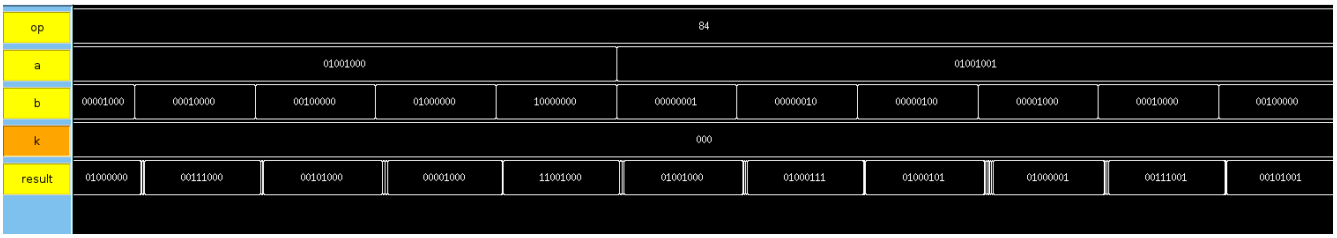
XOR:



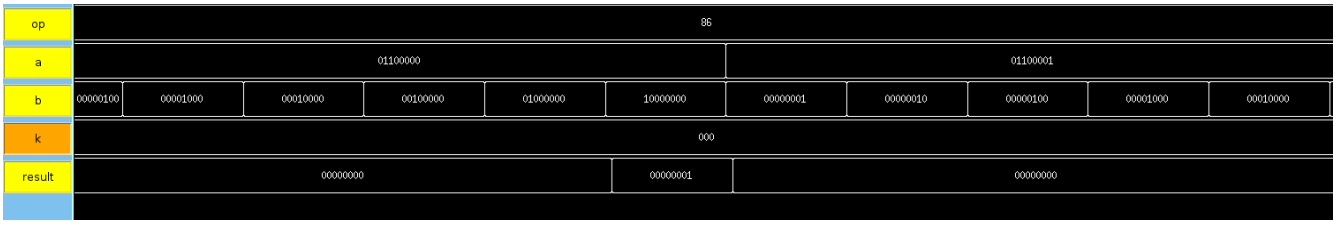
ADD:



SUB:



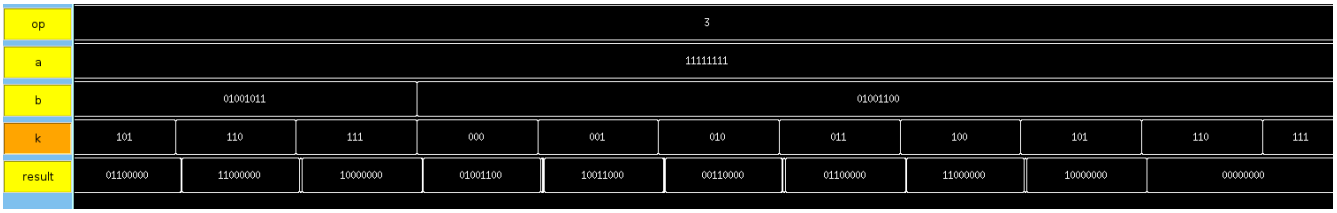
SLT:



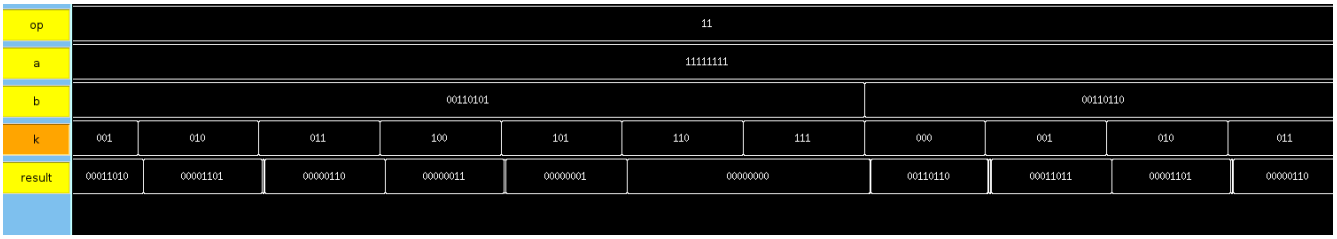
Funnel Shifter

Similar to the tests of the extended ALU shown above, the following shifter tests were of the shifter within the datapath. Standalone tests were done prior to integration as well, and the .cmd TCL file is included in the .tar file.

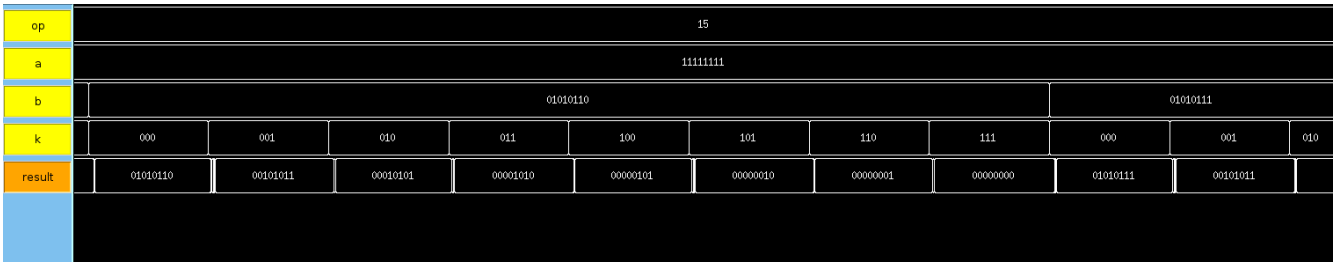
SLL:



SRL:

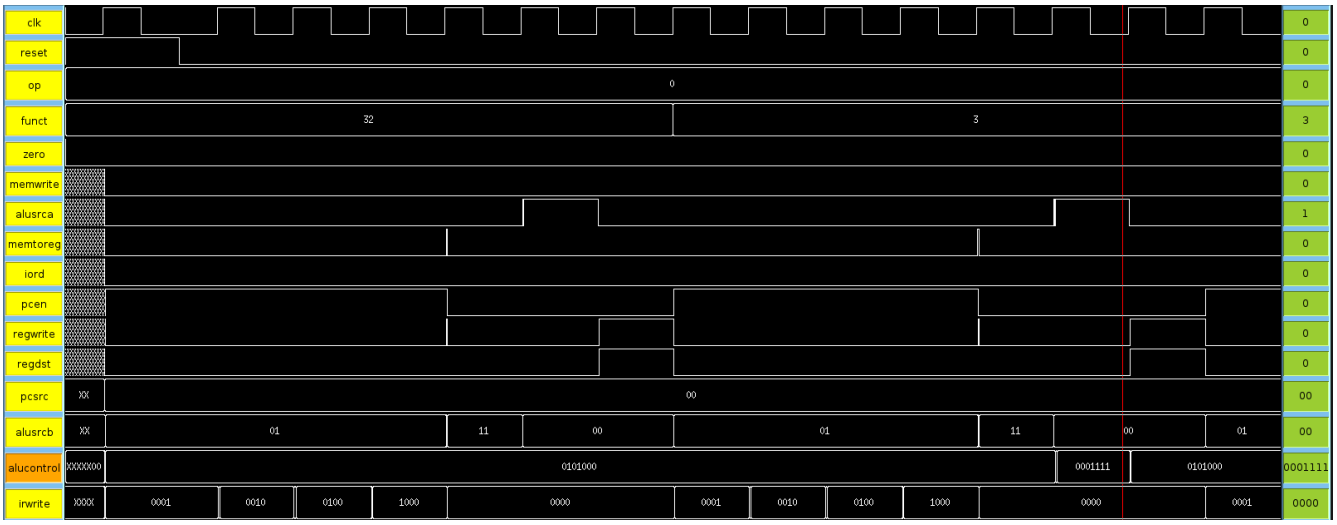


SRA:

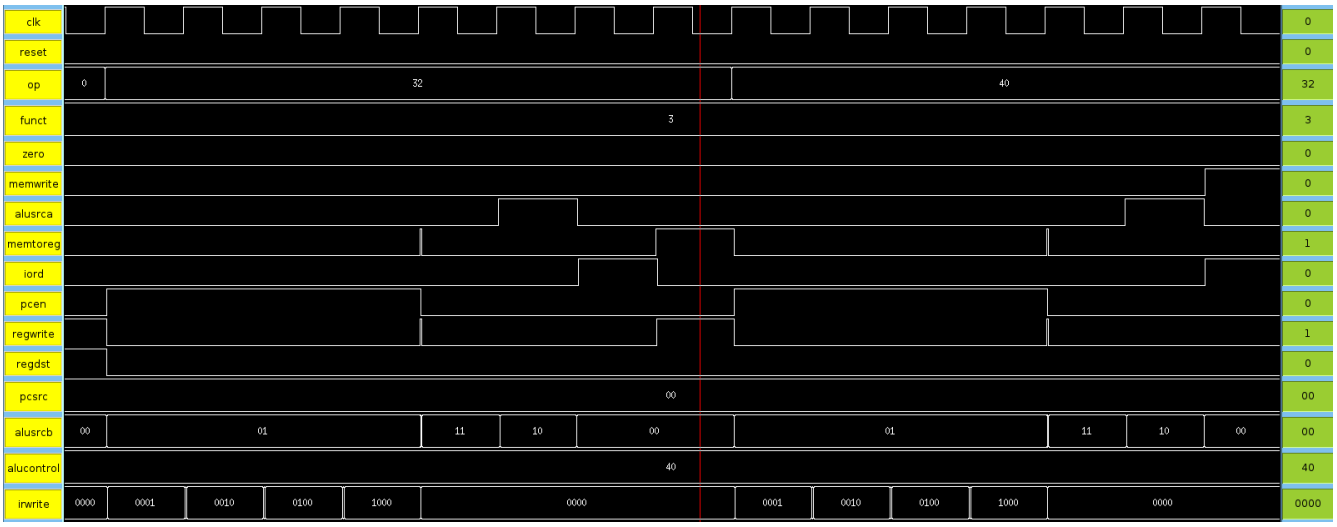


Control Unit

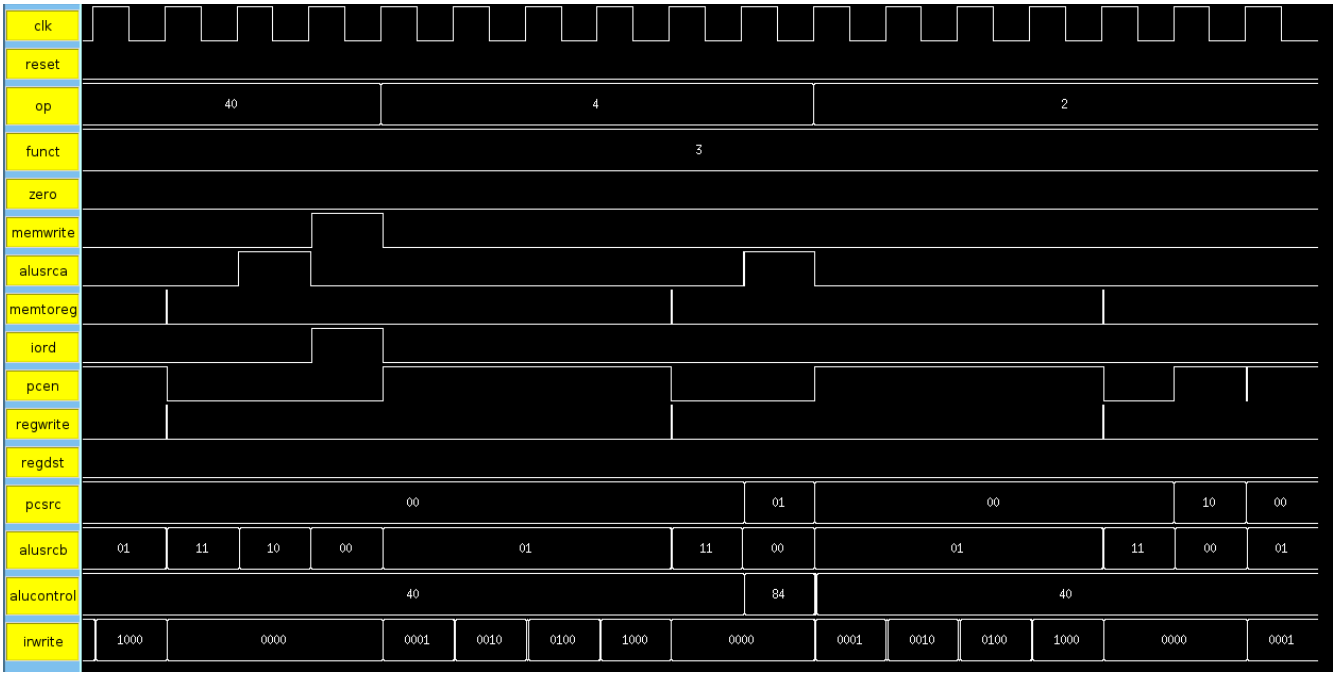
R-Type:



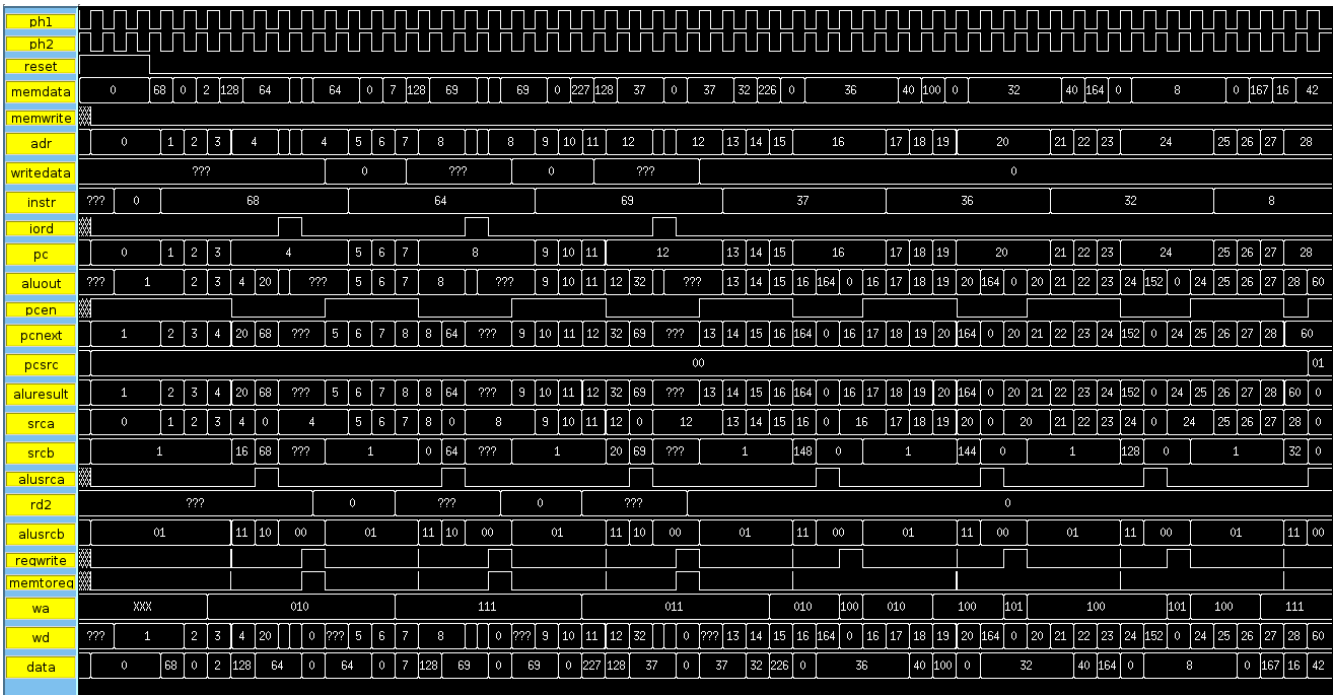
LB, SB:



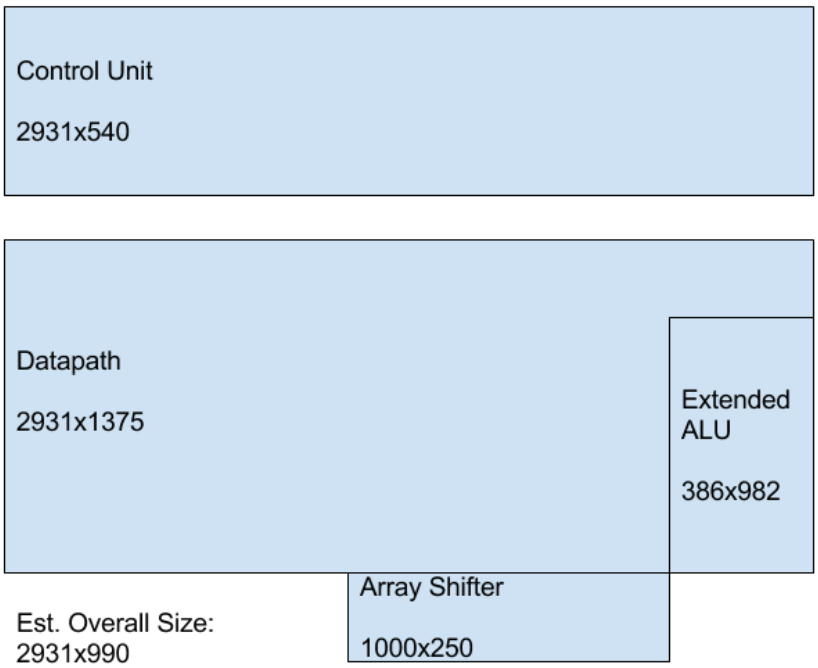
BEQ, J:



Overall Chip



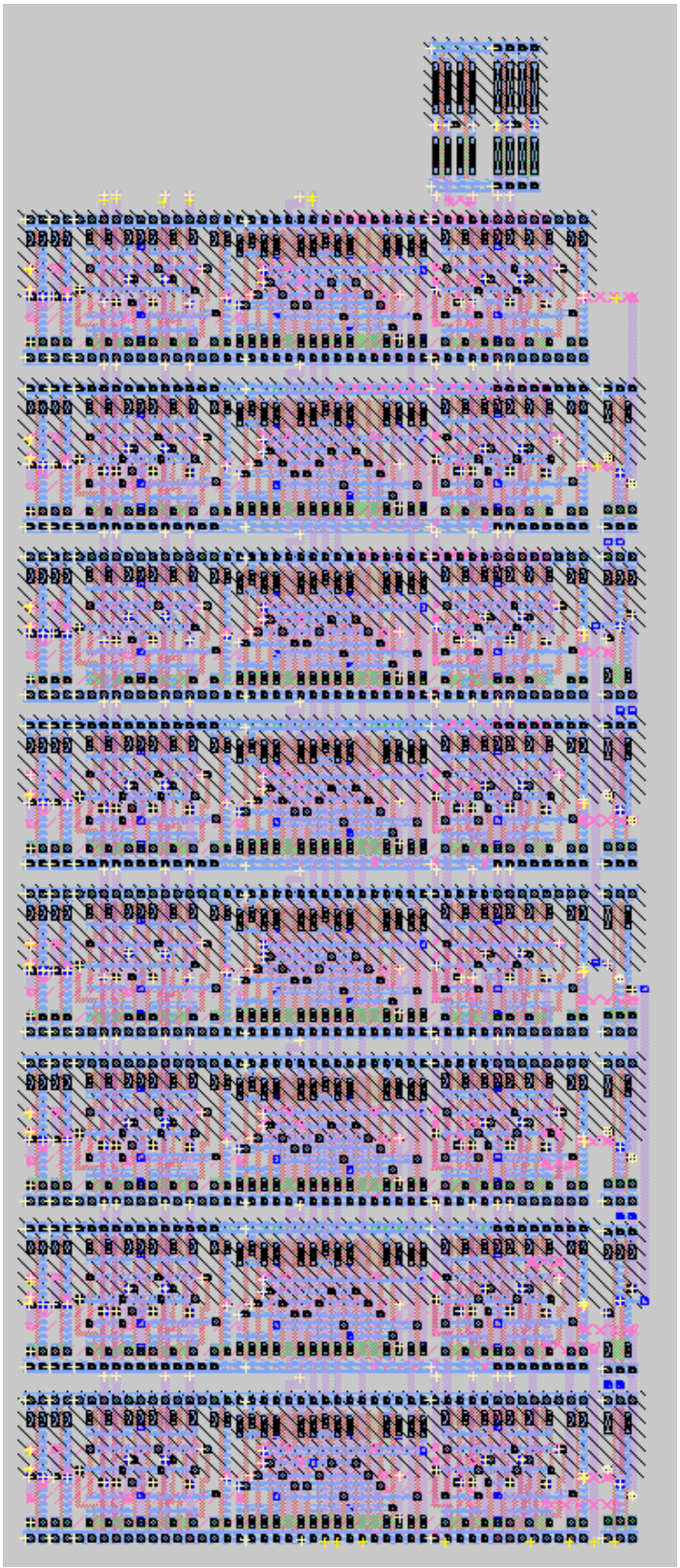
Chip Floorplan



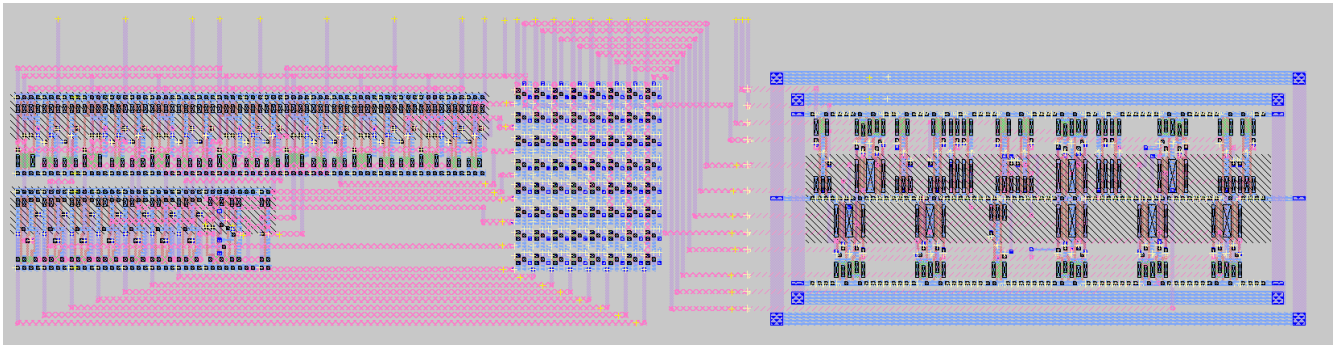
Note: Shifter shown includes source gen. + decoder logic
Dimensions listed in lambda

Layout Plots

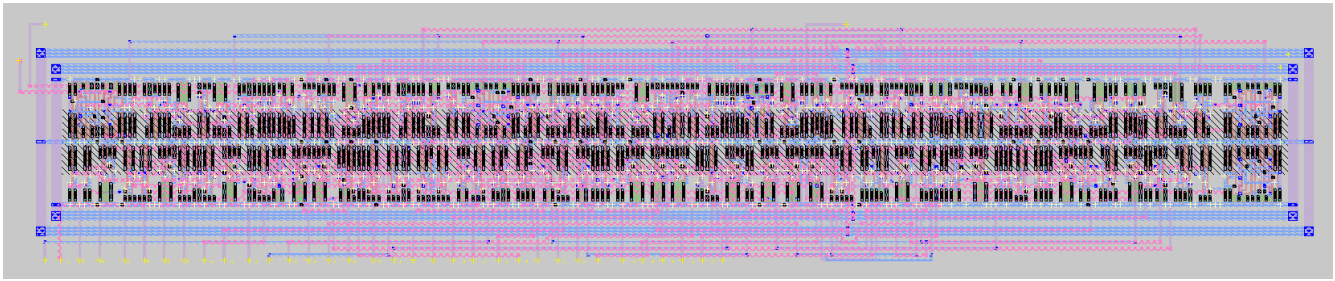
Extended ALU



Funnel Shifter



Control Unit



Core Cell

Overall Chip