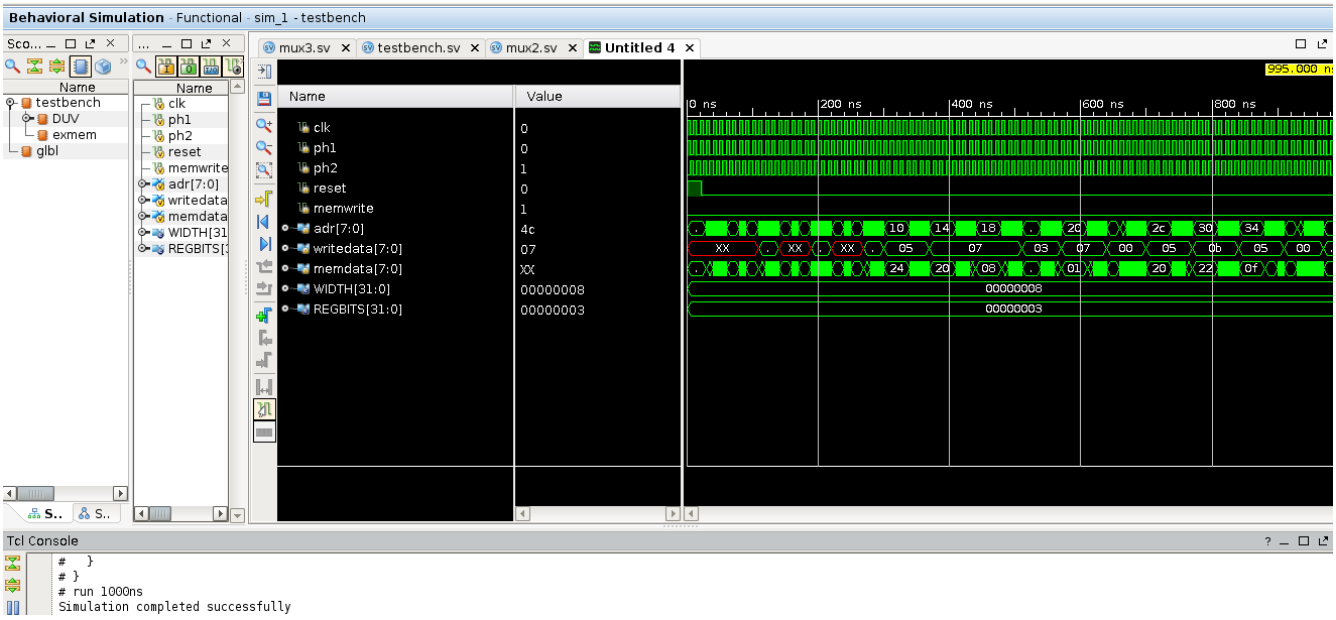
Adam Ness and Marty Townley
VLSI
Lab 9
4/8/17

Original assembly program, commented:

```
.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
```

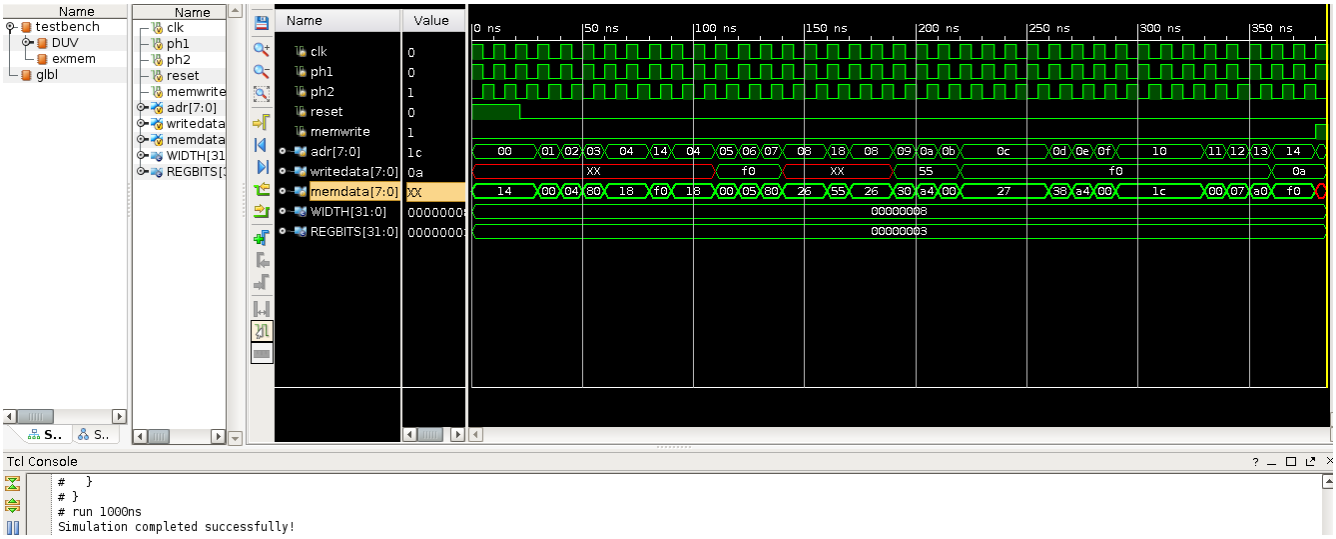
After some modifications, the original program executed properly on the mips processor.
Simulation screenshot:



Next, the following assembly program was written to test xor and nor functionality:

```
.data:00000000 80040014 lb a0,20(zero) a0 = 0xf0
.data:00000004 80050018 lb a1,24(zero) a1 = 0x55
.data:00000008 00a43026 xor a2,a1,a0   a2 = (0xf0)xor(0x55) = 0xa5
.data:0000000c 00a43827 nor a3,a1,a0   a3 = (0xf0)nor(0x55) = 0x0a
.data:00000010 a007001c sb a3,28(zero) write 0x0a to memory
.data:00000014 000000f0
.data:00000018 00000055
```

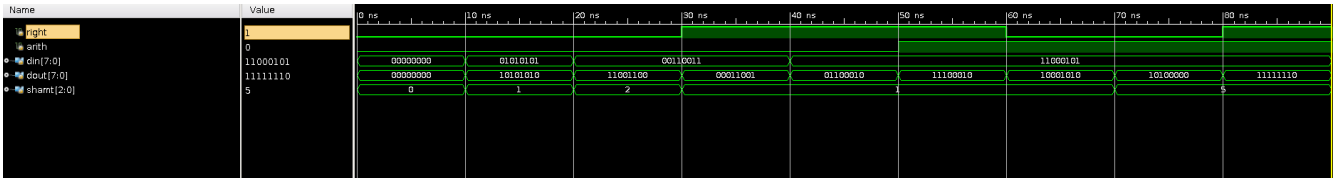
Simulating the circuit indicated proper operation:



Additionally, re-running the original simulation indicated the circuit still works.

Finally, the shifter was added and simulated.

Standalone shifter simulation:



Finally, the shifter was added to and simulated in the mips processor. For ease of integration, the shifter was added to the alu module, and was integrated. A markup of the resulting structure is shown in the following block diagram:

1. A PDF report file that includes assembly listings of the original test program and your modified program along with captures simulation showing that it completed correctly. Include a summary that describes any problems you encountered and provides an estimate of how long it took you to complete this lab
2. A tarball containing your SystemVerilog files including testbenches and memory data files.

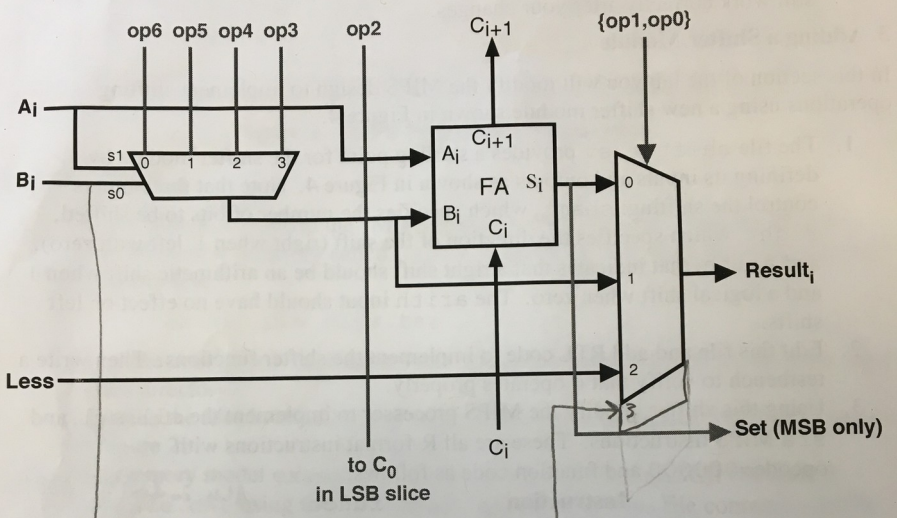


Figure 3 – Extended ALU Slice (From Lab 4)

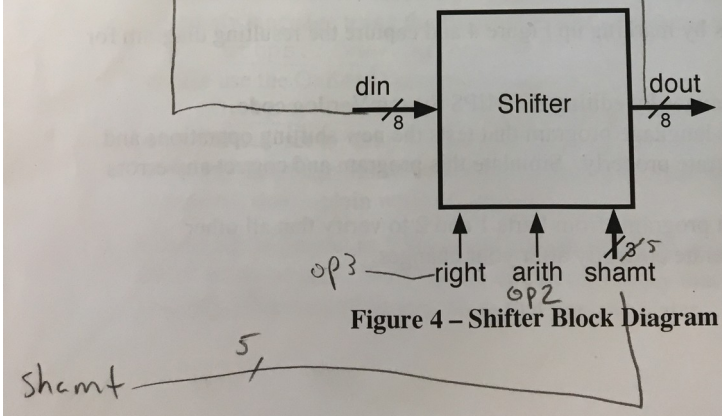
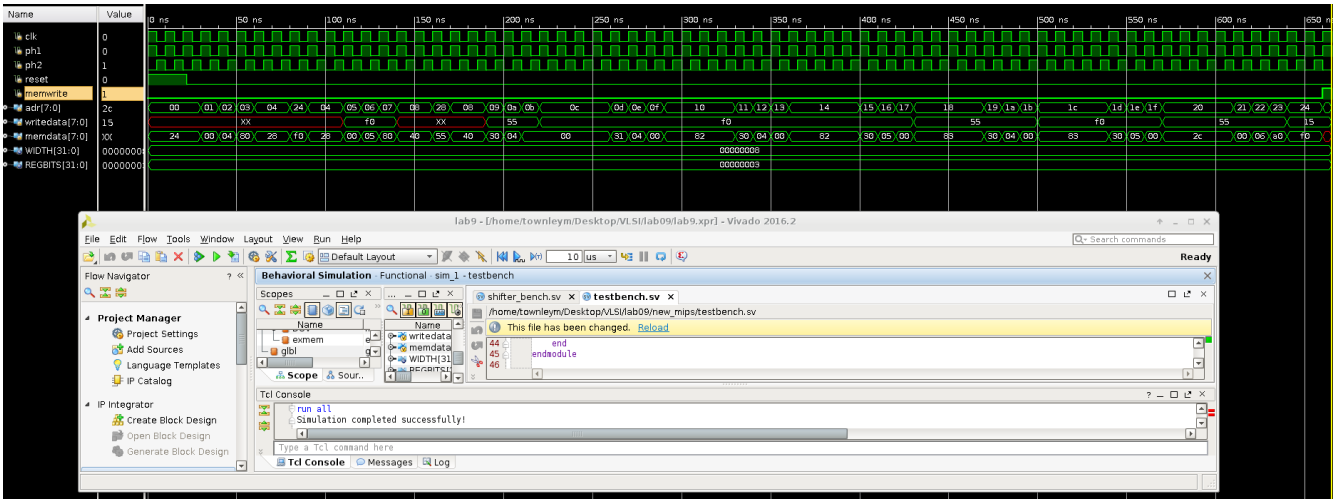


Figure 4 – Shifter Block Diagram

A new mips assembly program was created to test shifting functionality in the processor:

```
.data:00000000 80040024 lb a0,36(zero)  a0 = 0xf0
.data:00000004 80050028 lb a1,40(zero)  a1 = 0x55
.data:00000008 00043040 sll a2,a0,0x1   a2 = 0xf0 << 2
.data:0000000c 00043100 sll a2,a0,0x4   a2 = 0xf0 << 4
.data:00000010 00043082 srl a2,a0,0x2   a2 = 0xf0 >> 2
.data:00000014 00053082 srl a2,a1,0x2   a2 = 0x55 << 2
.data:00000018 00043083 sra a2,a0,0x2   a2 = 0xf0 >>> 2
.data:0000001c 00053083 sra a2,a1,0x2   a2 = 0x55 >>> 2
.data:00000020 a006002c sb a2,44(zero)  store 0x15 into memory
.data:00000024 000000f0
.data:00000028 00000055
```

This program was simulated to ensure correct operation, and we found that it works correctly:



Finally, the previous two programs were run again to ensure correct operation of all instructions, and we found that everything still works fine.

No major problems were encountered. The only issues involved an unsigned operation encoded in the alu_decoder, which needed fixing for the first part of the lab. For the xor and nor simulation, we found that the online assembler encoded an instruction wrong, which also had to be fixed.

It took approximately 7 hours to complete this lab.