Ian Miller, Christer Hoeflinger
ECE 425L
Lab 9: MIPS RTL Processor Model
April 10 2017

Marked up original test program

```
.data:00000000 80 02 00 44 lb     v0,68(zero)          ;Load 0x5 into v0
.data:00000004 80 07 00 40 lb     a3,64(zero)          ;Load 0x3 into a3
.data:00000008 80 e3 00 45 lb     v1,69(a3)            ;Load 0xc into v1
.data:0000000c 00 e2 20 25 or     a0,a3,v0             ;Or 0x3 and 0x5 into a0 (0x7)
.data:00000010 00 64 28 24 and    a1,v1,a0             ;And 0xc and 0x7 into a1 (0x4)
.data:00000014 00 a4 28 20 add    a1,a1,a0             ;Add 0x4 and 0x7 into a1 (0xb)
.data:00000018 10 a7 00 08 beq    a1,a3,0x0000003c     ;If a3==a1 (0x3 not 0xb), so not taken
.data:0000001c 00 64 30 2a slt    a2,v1,a0             ;If v1<a0 (0xc<0x7) then set a2, a2=0x0
.data:00000020 10 c0 00 01 beqz   a2,0x00000028        ;If a2=0, goto branch1:
.data:00000024 80 05 00 00 lb     a1,0(zero)           ;NOT EXECUTED

branch1:
.data:00000028 00 e2 30 2a slt    a2,a3,v0             ;If a3<v0 (0x3<0x5) then set a2, a2=0x1
.data:0000002c 00 c5 38 20 add    a3,a2,a1             ;Add 0x01 and 0xb into a3 (0xc)
.data:00000030 00 e2 38 22 sub    a3,a3,v0             ;Sub 0x5 from 0xc into a3 (0x7)
.data:00000034 08 00 00 0f j      0x0000003c           ;Jump to branch2:
.data:00000038 80 07 00 00 lb     a3,0(zero)           ;NOT EXECUTED

branch2:
.data:0000003c a0 47 00 47 sb     a3,71(v0)            ;Set address 0x4c to 0x7
.data:00000040 00 00 00 03 0x3
.data:00000044 00 00 00 05 0x5
.data:00000048 00 00 00 0c 0xc
```

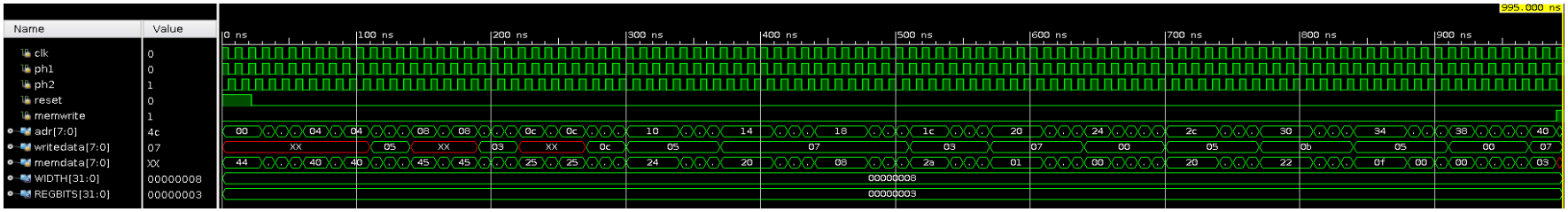

Illustration 1: Original Test Program Simulation

Note that at the very end (the cursor location, shown under “value”) writedata is 0x07 and adr is 0x4c, as is desired from the test program. Upon execution, the testbench also printed out “test completed successfully”, which checks the exact same thing

Marked XOR/NOR Test Program

```
0x0:   lb v0,20(zero)      ;Load 0xf0 into v0
0x4:   lb v1,24(zero)      ;Load 0xaa into v1
0x8:   xor a0, v0, v1      ;v0 xor v1 = 0x5a
0xc:   nor a1, v0, v1      ;v0 nor v1 = 0x05
0x10:  sb  a1, 0(a0)       ;store 0x05 at address 0x5a
0x14:  0xf0
0x18:  0xaa
```

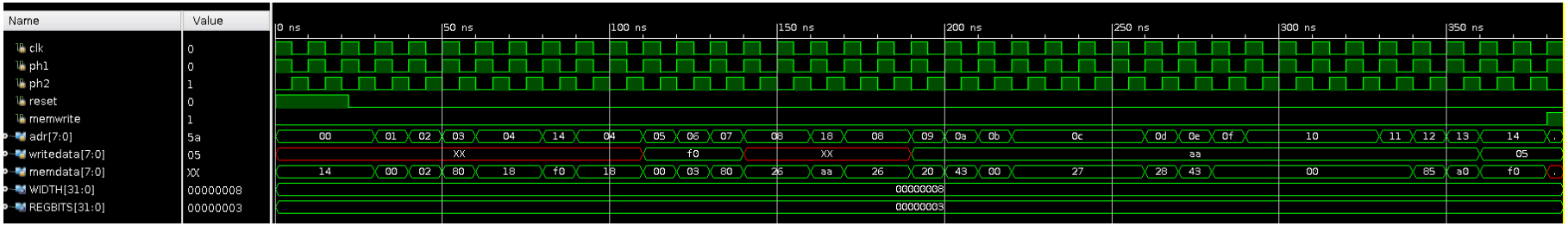

Illustration 2: XOR/NOR Test Program Simulation

Note that at the very end (the cursor location, shown under “value”) writedata is 0x05 and adr is 0x5a, as is desired from the test program. Upon execution, the testbench also printed out “test completed successfully”, which checks the exact same thing.

Marked Shifter Test Program

```
0x0:   lb v0,28(zero)      ;Load 0xf0 into v0
0x4:   lb v1,32(zero)      ;Load 0xaa into v1
0x8:   srl a0, v0, 4       ;a0 = 0xf0 shifted right logical 4 (0x0f)
0xc:   sll a1, v1, 4       ;a1 = 0xaa shifted left logical 4 (0xa0)
0x10:  beq a2 a1 0x50      ;a2 = 0? (no)
0x14:  sra a1, a1, 4       ;(0xfa)
0x18:  sb  a0, 0(a1)       ;store 0x0f at address 0xfa
0x1c:  0xf0
0x20:  0xaa
```

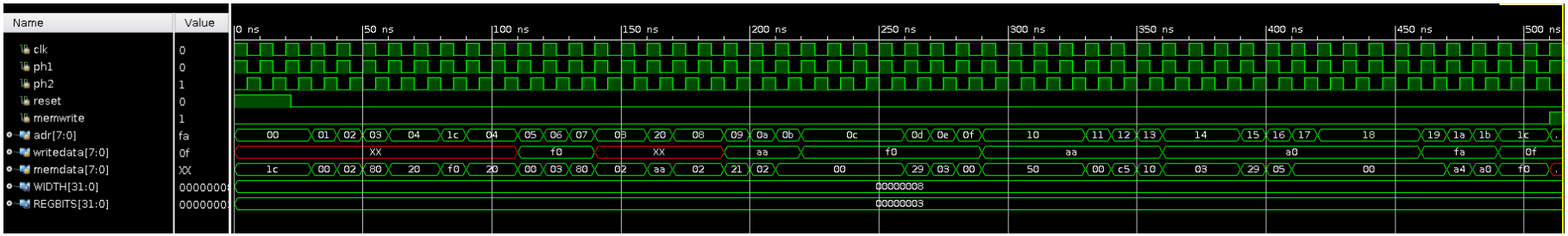

Illustration 3: Shifter Test Program Simulation

Note that at the very end (the cursor location, shown under “value”) writedata is 0x0f and adr is 0xfa, as is desired from the test program. Upon execution, the testbench also printed out “test completed successfully”, which checks the exact same thing.

All three tests were run on the final version of the MIPS processor (the simulations shown here)

The lab took about 4 hours. Problems encountered included needing to use a $signed macro to force Vivado to handle signed bit shifts properly. There was also some fiddly debugging in updating the ALU decoder, since a number of buses had to change in width to handle the new number of alu control signals.