Stage 5

Divyanshu Pabia - RA2111003011373

The epwaves shown are the result of the testbench that I have submitted along with my code in this zip file. I have tested my design on a few custom-made examples. They are present in a folder named test_cases. I have tested all the methods of using Shift and rotate as given in the assignment pdf.

Design Description:

I have added another module named shifter. It comprises 5 more smaller modules:- shifter1, shifter2, shifter4, shifter8, shifter16. The names are self-explanatory. The other changes in this stage were the changes that needed to be made in the datapath and 2 new states were added to the finite state machine, thus bringing the total number of states to 11. The first state added was for reading a third register from the register file, and the second state added was to clock the shifter result. The carry from the shifter has not been used as of now, and will be integrated with the rest of the design in subsequent stages. I have shown the output of the carry signal in my second test case.

The design works up to the expectation and nothing has been hardcoded. I have attached the epwave outputs of as many signals as I could/are relevant.

How to test my test cases:

I have attached the test cases as a .vhd files. To test them in the code, we must replace the complete code in mem.vhd with the code in the test case file, and then run on edaplayground.com. To check if the required output is being stored correctly, read below:

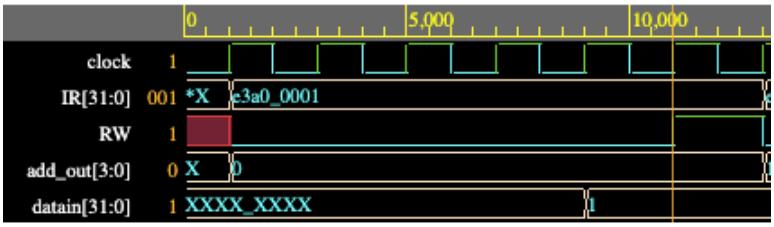
Guide to my epwave output:

1. Check datain[31:0] and add_out when RW = 1.
2. Datain[31:0] is the value going into the register file.
3. Add_out is the address at which the value is being entered

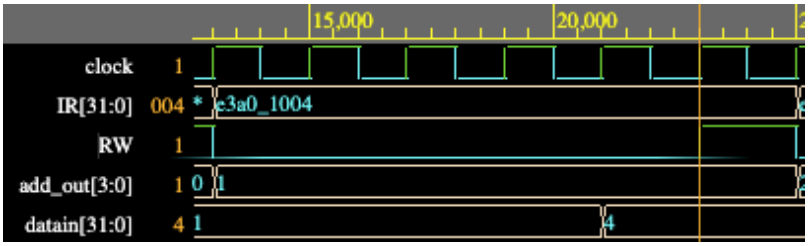
All these programs work in my design. Nothing has been hardcoded, or taken from anybody else. I have included all essentials signals in the epwaves that I have submitted. If needed, I can submit more as well. I have extensively tested all commands.

Test Cases and Output:

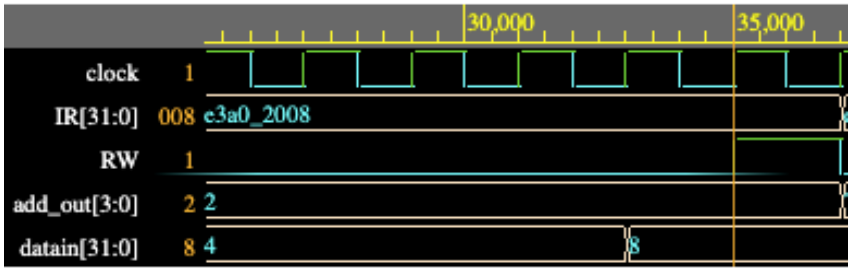
1. mov r0,#0x1



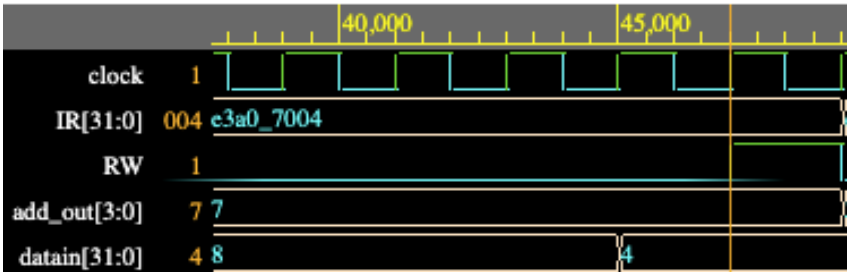
2. mov r1, #0x4



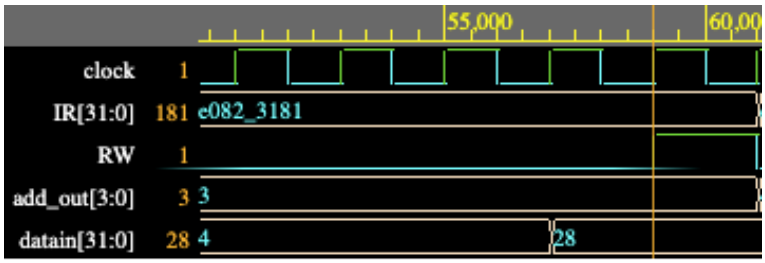
3. mov r2, #8



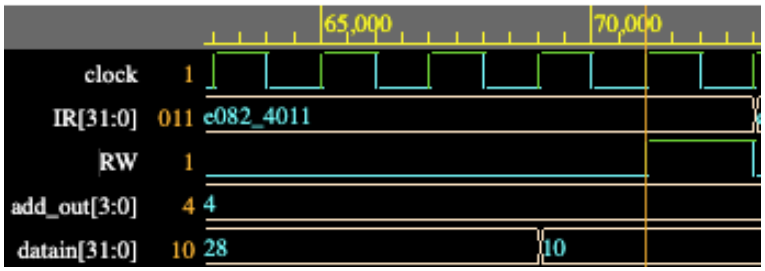
4. mov r7,#4



5. add r3, r2, r1, LSL #3

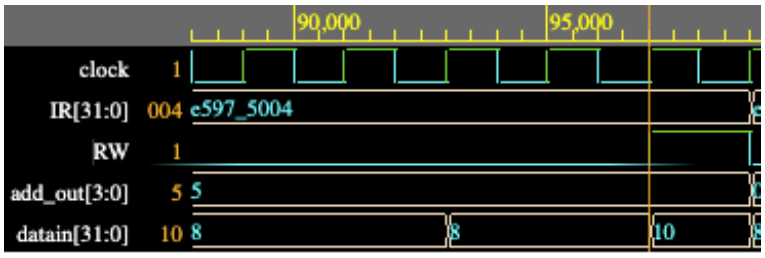


6. add r4, r2, r1, LSL r0

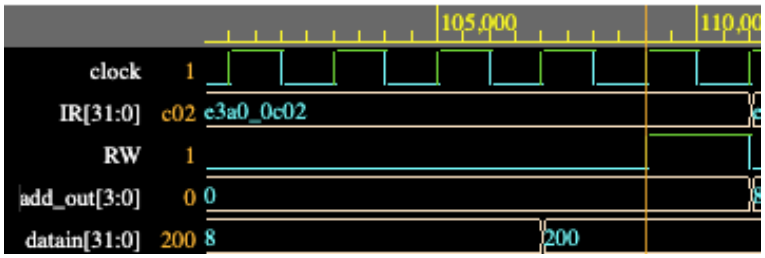


7. str r4, [r7, r2, LSR #1]

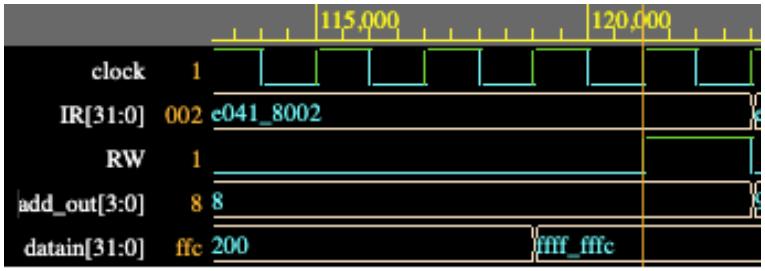
8. ldr r5, [r7, #4] (retrieved the correct value from the data memory)



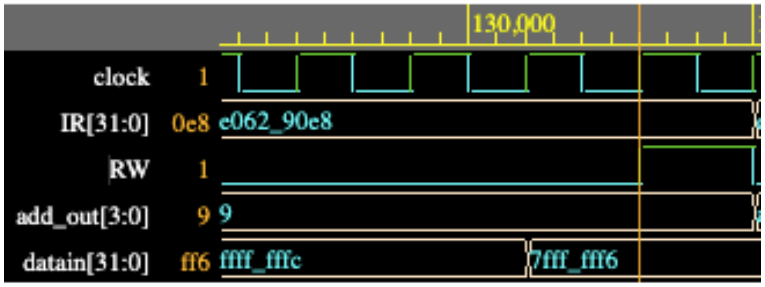
9. mov r0,#512



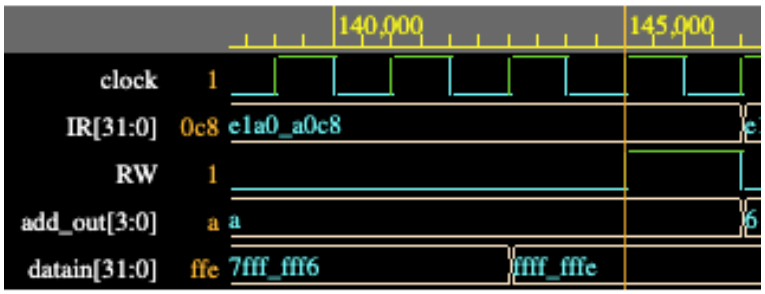
10. sub r8, r1,r2



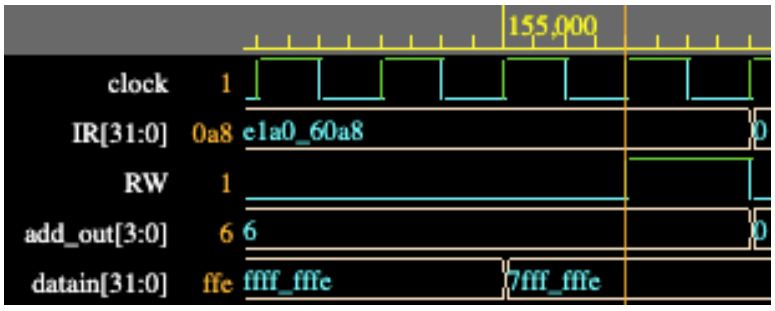
11. rsb r9,r2,r8, ROR #1



12. mov r10,r8, ASR #1

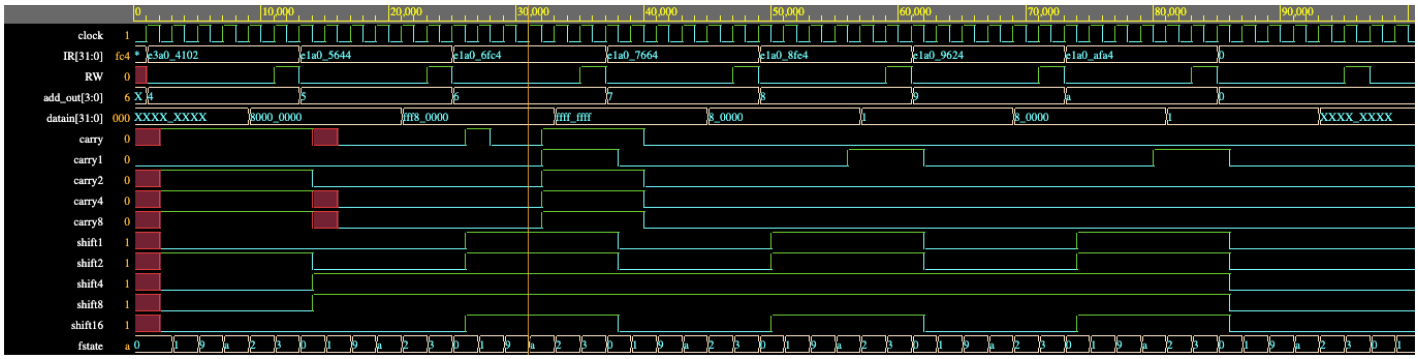


13. mov r6,r8, LSR #1



All the shift and rotate instructions have been tested above.

Output of shift2.vhd



Here, I have shown the working of the carry flag as well. Note that, I am going to use the shifter result and carry only at the start of state 10(a). The values of the result and carry in all other states is **"don't care".**

This is the code in shift2.vhd:

```
mov r4,#0x80000000
mov r5,r4, ASR #12
mov r6,r4, ASR #31
mov r7,r4, ROR #12
mov r8,r4, ROR #31
mov r9,r4, LSR #12
mov r10,r4, LSR #31
```

You can check the output in datain[31:0], it should match with: