Stage 4

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 DP instructions and have checked flag updation.

Design Description:

I have used 9 states in the finite state machine. In the last stage I submitted a 10-state model. I have now removed state number 9.

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 each test case as a .vhd file, with the names being self-explanatory. To test them in the code, we must replace the complete code in mem.vhd with the code in the required 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
4. The 4 flags are named accordingly so as to be easily recognised.

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.

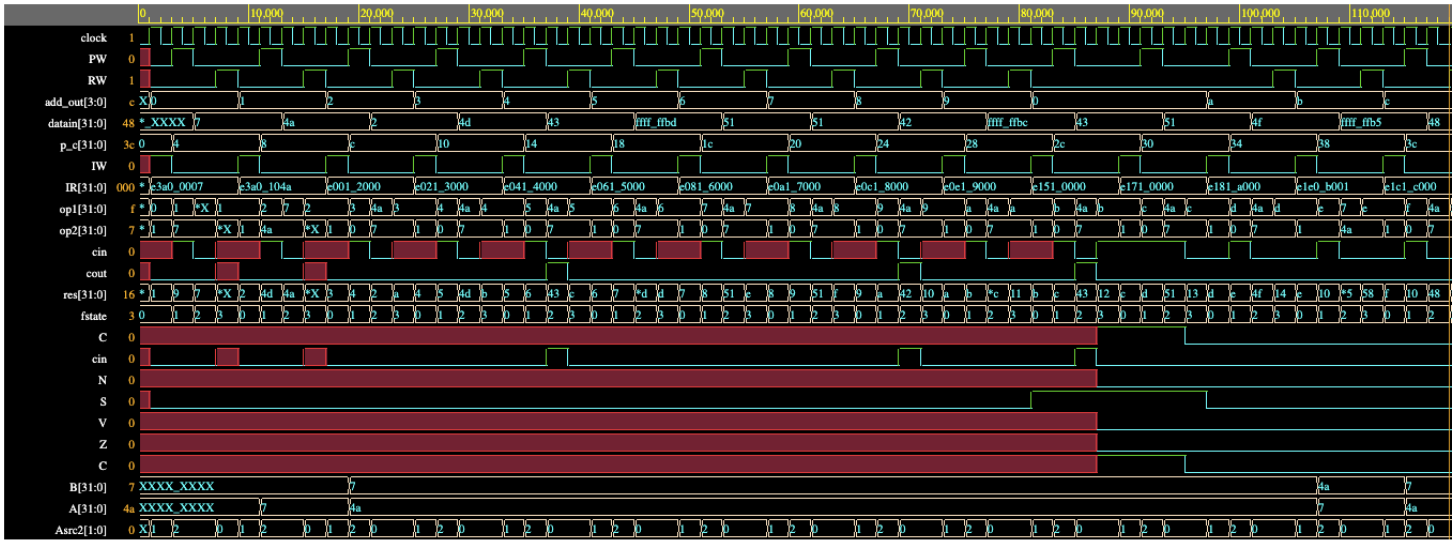
Test Cases and Output:

1. Testing All DP instructions:

Screenshot of my ARMSim program output:

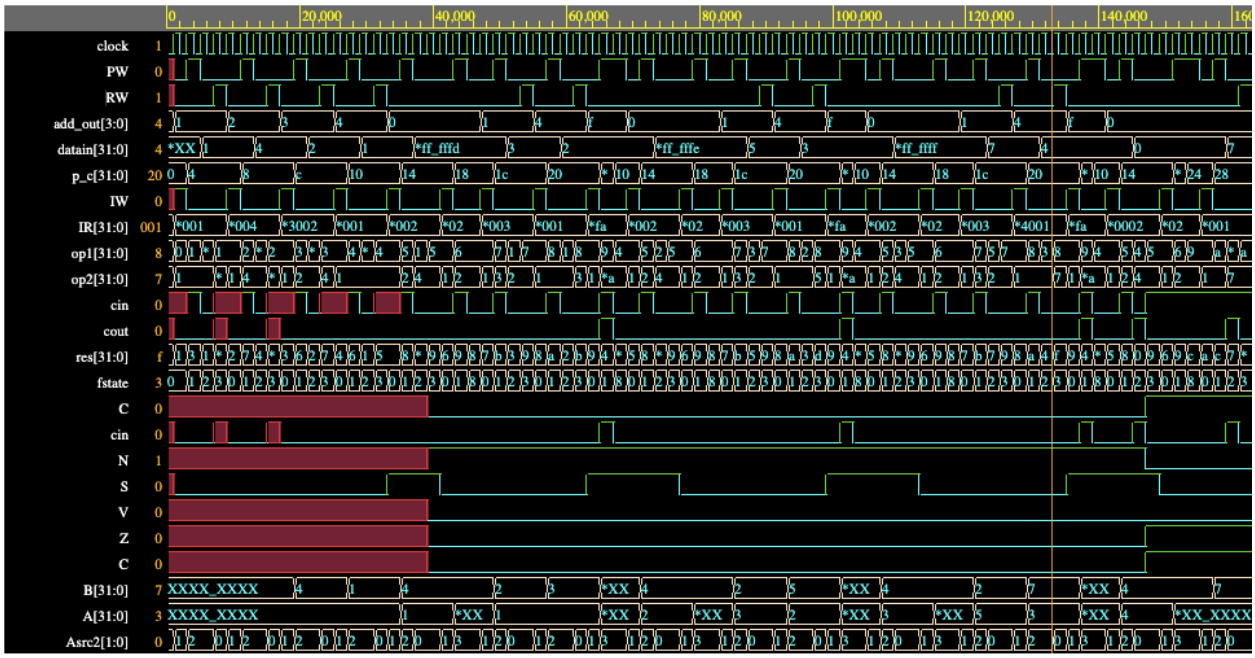
```
                        .text

00001000:E3A00007    mov r0,#7
00001004:E3A0104A    mov r1,#74

00001008:E0012000    and r2,r1,r0
0000100C:E0213000    eor r3,r1,r0
00001010:E0414000    sub r4,r1,r0
00001014:E0615000    rsb r5,r1,r0
00001018:E0816000    add r6,r1,r0
0000101C:E0A17000    adc r7,r1,r0
00001020:E0C18000    sbc r8,r1,r0
00001024:E0E19000    rsc r9,r1,r0
00001028:E1510000    cmp r1,r0
0000102C:E1710000    cmn r1,r0
00001030:E181A000    orr r10,r1,r0
00001034:E1E0B001    mvn r11,r1
00001038:E1C1C000    bic r12,r1,r0
```

Epwave output:



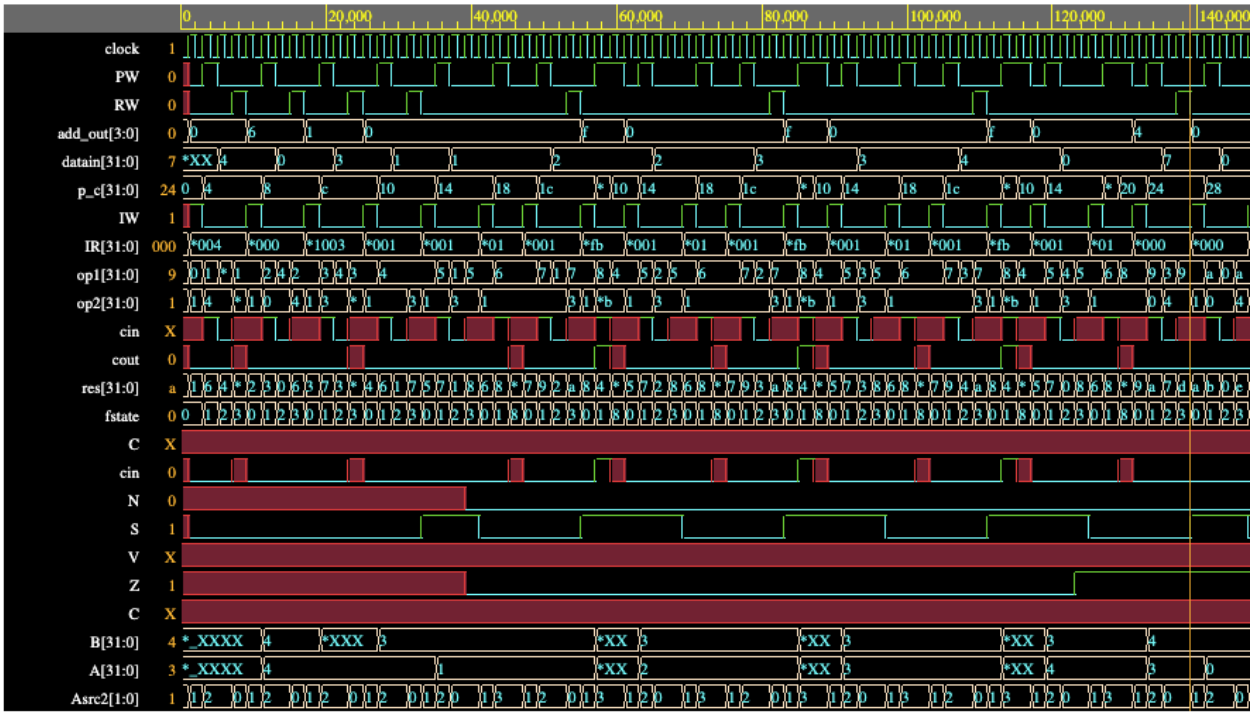
2. nth term of an AP:-

```
00001000:E3A01001    mov r1,#1        @starting term
00001004:E3A02004    mov r2,#4        @number of terms in the ap
00001008:E3A03002    mov r3,#2        @common difference

0000100C:E3A04001    mov r4,#1        @loop variable

00001010:E1540002    Loop:   cmp r4,r2
00001014:0A000002            beq out

00001018:E0811003            add r1,r1,r3
0000101C:E2844001            add r4,r4,#1
00001020:EAFFFFFA            b Loop

00001024:E1A00001    out: mov r0,r1
```



3. Tst and teq

```
00001000:E3A00004          mov r0, #4
00001004:E3A06000          mov r6, #0
00001008:E3A01003          mov r1, #3
0000100C:E3A00001          mov r0,#1
00001010:E1100001    L:    tst r0, r1
00001014:0A000001          beq END
00001018:E2800001          add  r0,r0,#1
0000101C:EAFFFFFB          b L
00001020:E0814000    END:  add r4, r1, r0
00001024:E3360000          teq r6,#0
```



4. Flags checker

```
                        .text

00001000:E3A00007    mov r0,#7
00001004:E3A0104A    mov r1,#74

00001008:E0715000    rsbs r5,r1,r0
0000100C:E2956045    adds r6,r5,#69
00001010:E2567002    subs r7,r6,#2
```