ASSIGNMENT 2 STAGE 4 REPORT

Sreemanti Dey

January 2022

1 Objective

This stage is mainly to test all DP opcodes for the multi-cycle processor we have designed in stage 3.

2 Assumptions

VHDL
edaplayground
Aldec Riviera Pro 2020.04 used for simulation
Mentor Precision 2021.1 used for synthesis

3 Implementation details

I had previously given support for only a subset of instructions (as mentioned in the specifications of the previous 2 stages), so my processor in stages 2 and 3 was designed for only that subset. Thus, in this stage, I have added support for all the DP instructions, hence I have made some changes in my processor.vhd and flagupd.vhd.
The changes include the following:

1. In my processor.vhd, I have made changes to my read-write signal for the register and the carry-in for the ALU, to add support for all the DP opcodes.

2. In my flagupd.vhd, I have added support for all the DP opcodes.

I have written the following assembly program files in my program:

1. p1.s and, eor

2. p2.s add, sub, rsb

3. p3.s adc, sbc, rsc

4. p4a.s cmp
5. p4b.s cmn
6. p5.s orr, bic
7. p6.s mov, mvn
8. p7.s tst, teq

4 p1.s - Testing and, eor

4.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave

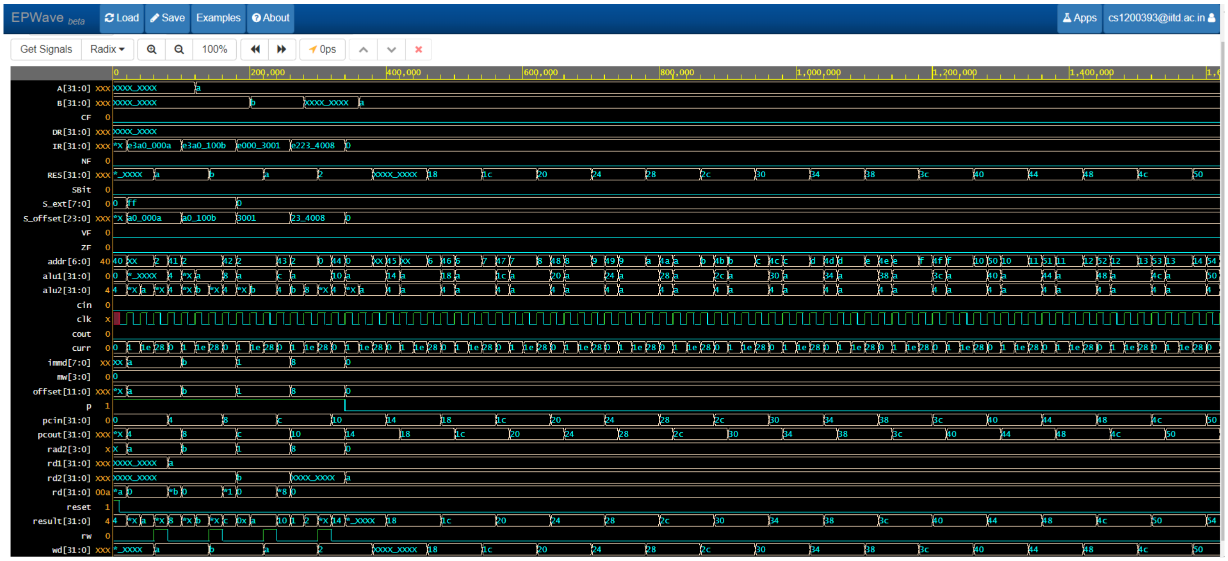

Figure 1: Program 1

We can see that when rw becomes ’1’ then I write 1011 = 10 into the register and then I write 10 eor 8 = 2 into the register, thus verifying and, eor work correctly.

5 p2.s - Testing add, sub, rsb

5.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave

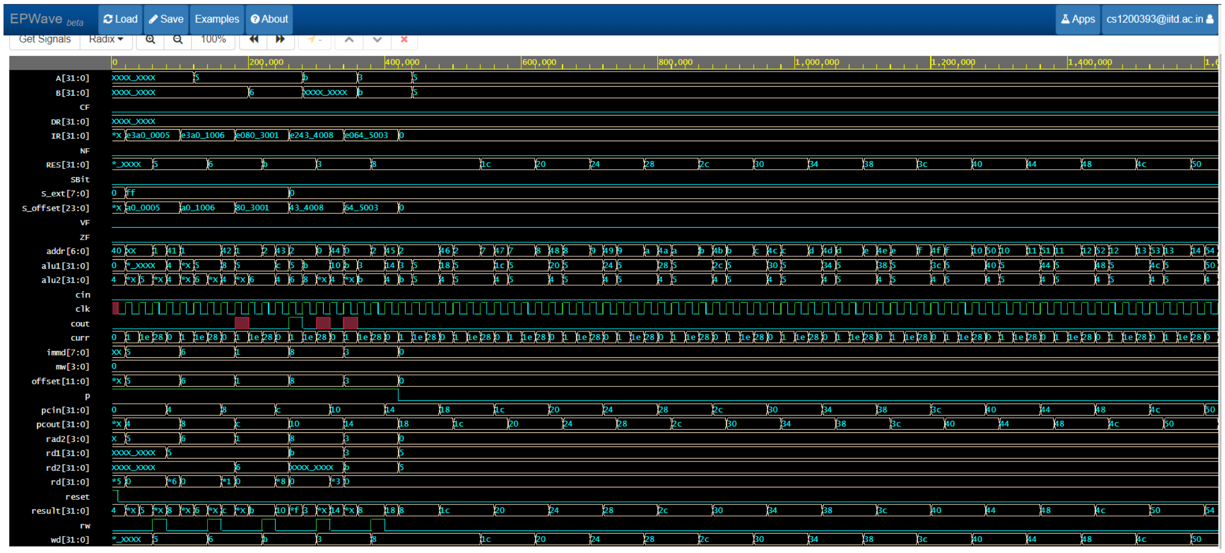

Figure 2: Program 2

We can see that when rw becomes '1' then I write 5+6 = 11 (add) into the register and then I write 11 - 8 = 3 (sub) into the register and then I write 11-3 = 8 (rsb) into the register, thus verifying add, sub, rsb work correctly.

6 p3.s - Testing adc, sbc, rsc

6.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave

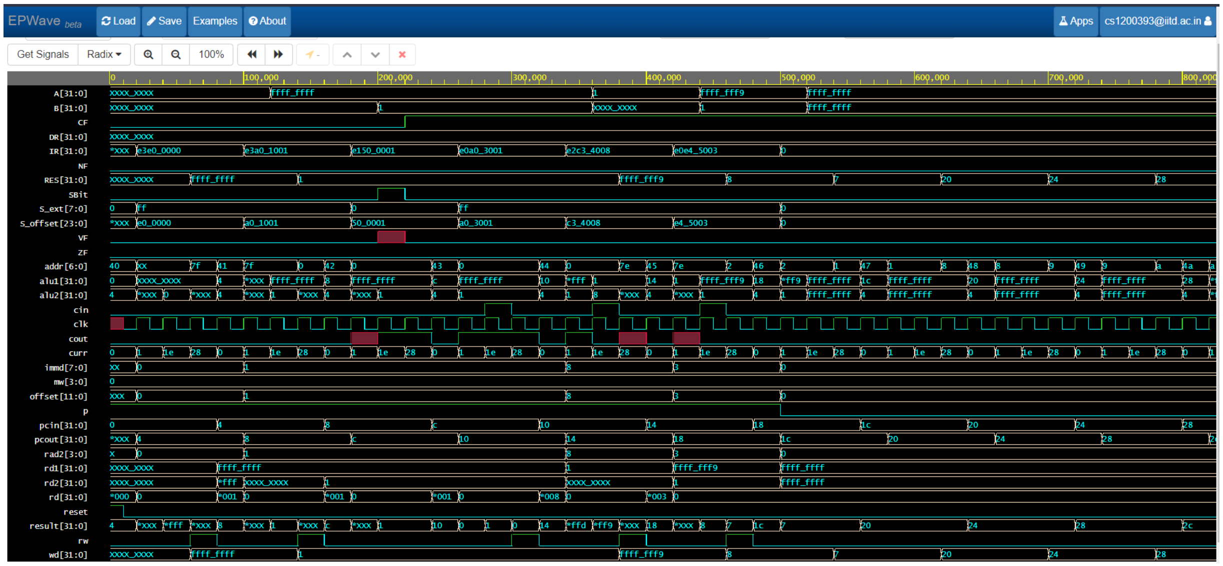

Figure 3: Program 3

Here, we can see that carry flag gets set when I do cmp 1, -1 since it is a subtraction, hence we get a carry-out and I use this carry out in adc, sbc and rsc instructions, and thus I get 1, -7 and 8 as the answers which are seen in the wd signal when rw is 1.

7 p4a.s - Testing cmp

7.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave

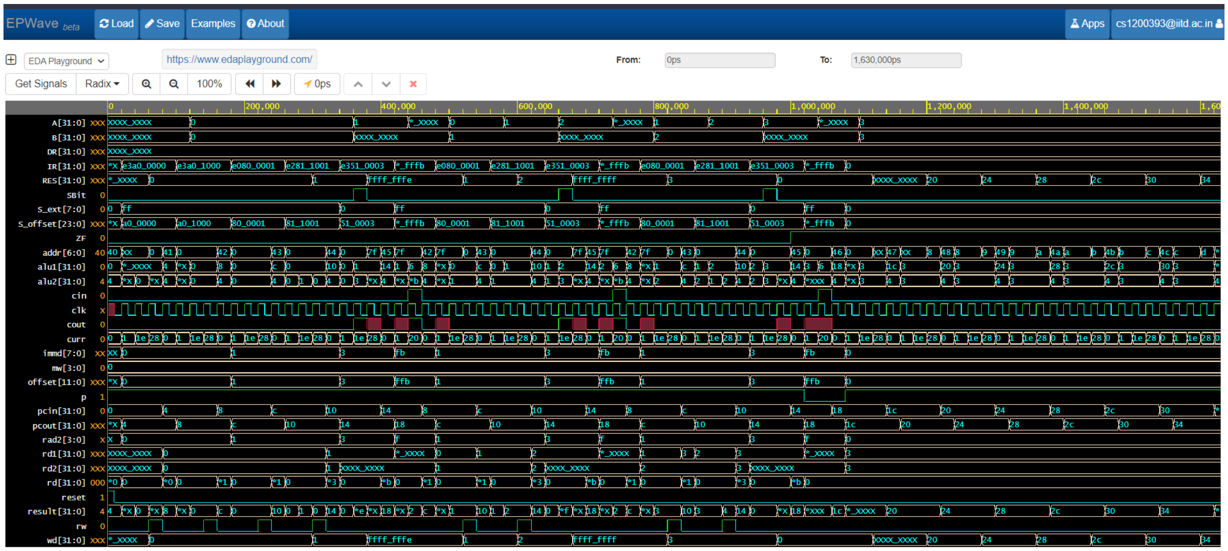

Figure 4: Program 4a

Here, I have used loop to check for correctness of cmp. I found that the times before I branch, r0,r1 have the correct value thus proving that cmp is working correctly.

8 p4b.s - Testing cmn

8.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave

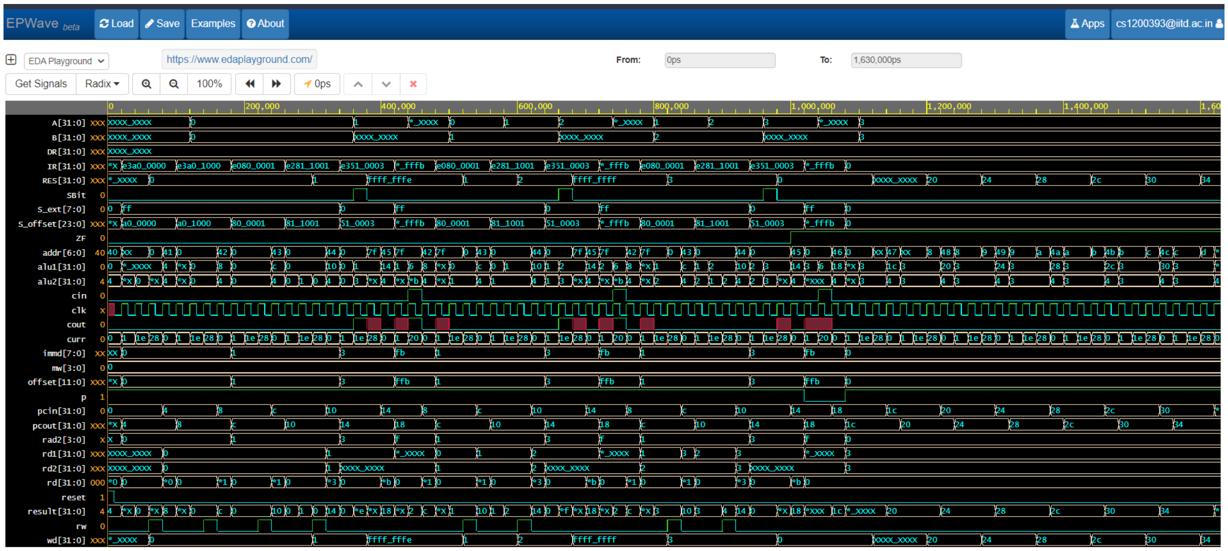

Figure 5: Program 4b

Here, I have used loop to check for correctness of cmp. I found that the times before I branch, r0,r1 have the correct value thus proving that cmp is working correctly.

9 p5.s - Testing orr, bic

9.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave

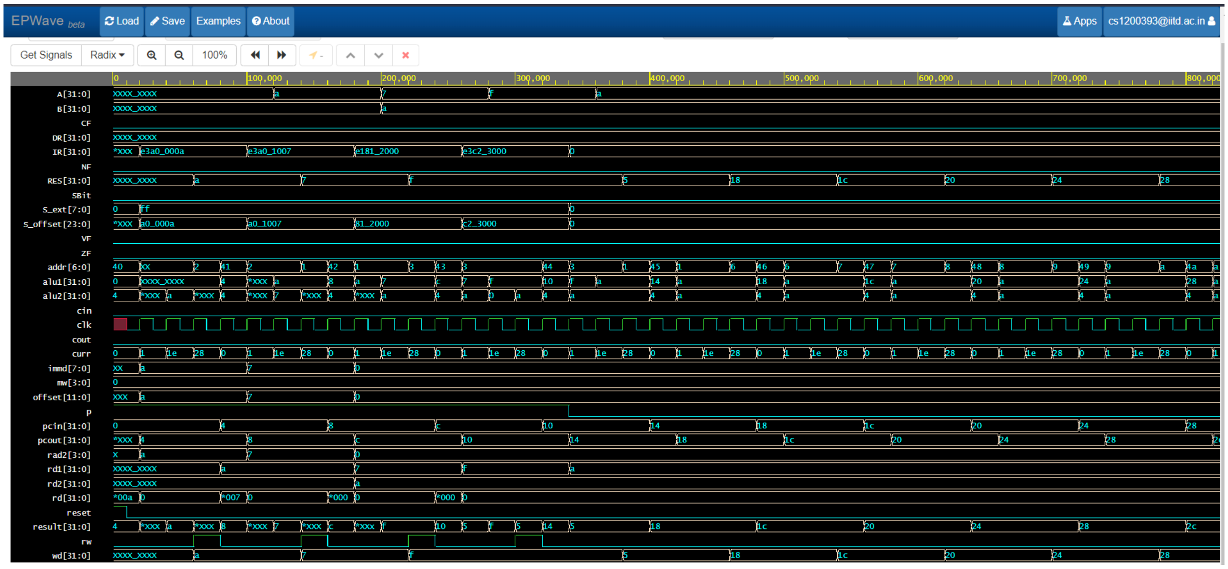

Figure 6: Program 5

Here, I do 10 — 7 which is 15, then I do 15 and not 0 = 15, and these are the places when rw = 1 and wd contains these data thus verifying orr and bic work correctly.

10 p6.s - Testing mov, mvn

10.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave.

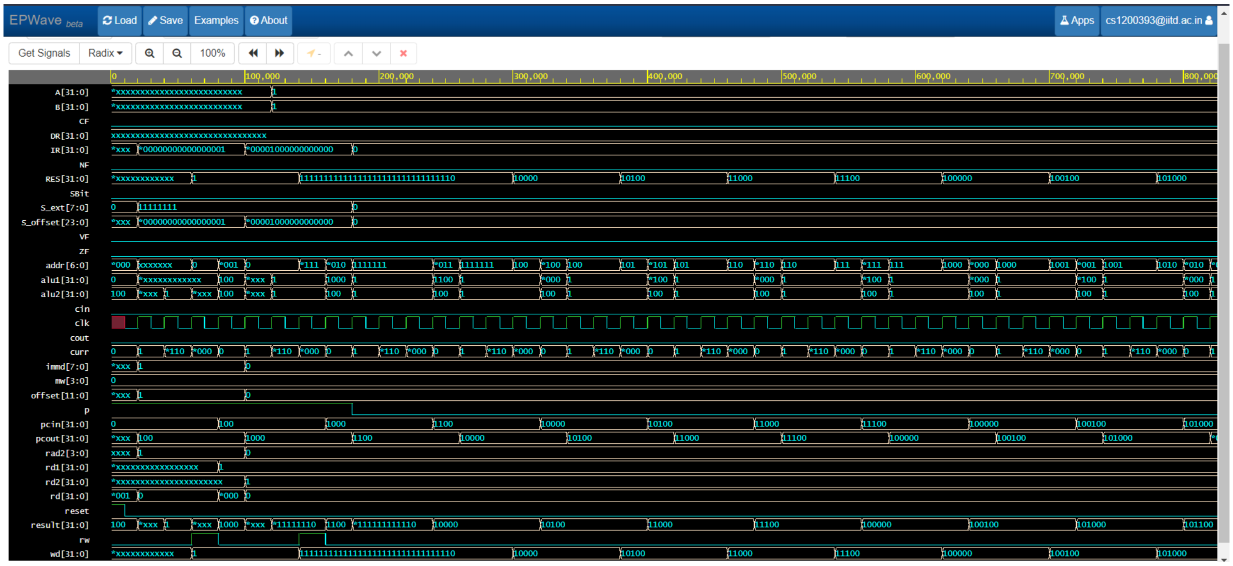

Figure 7: Program 6

Here, I have moved the value 1 into r0(mov), which is shown by the first time rw becomes 1 and then I have moved the complement of 1 into r0(mvn), which is shown the second time rw becomes 1. I have used representation binary here so as to see the values correctly.

11 p7.s - Testing tst, teq

11.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave.

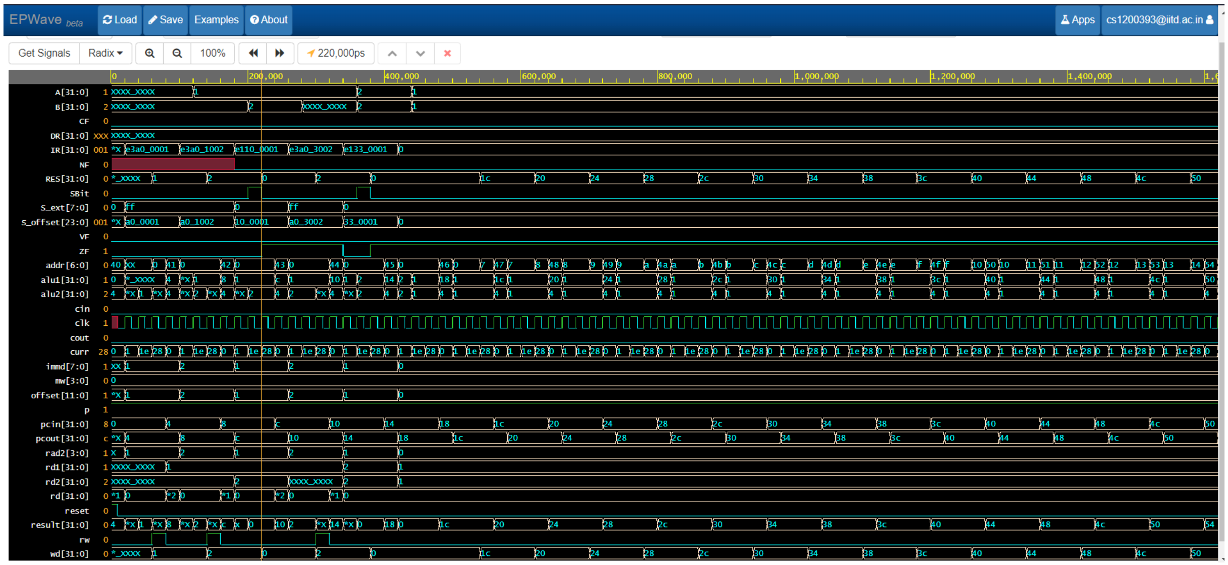

Figure 8: Program 7

Here, I have 1 and 2 = 0 hence tst should set the ZFlag to 1 and similarly for teq, we have 2 xor 2 = 0, hence teq should set the ZFlag to 1, which is seen in my EPwave hence verifying that the tst, teq commands work correctly.