COL 216 Assignment 2 Stage 4

2020CS50429 - Manshi Sagar

March 2022

1 Introduction

In this stage, we are required to check if our Multi Cycle Processor supports all DP opcodes. To verify this, few testcases have been taken.

2 Program

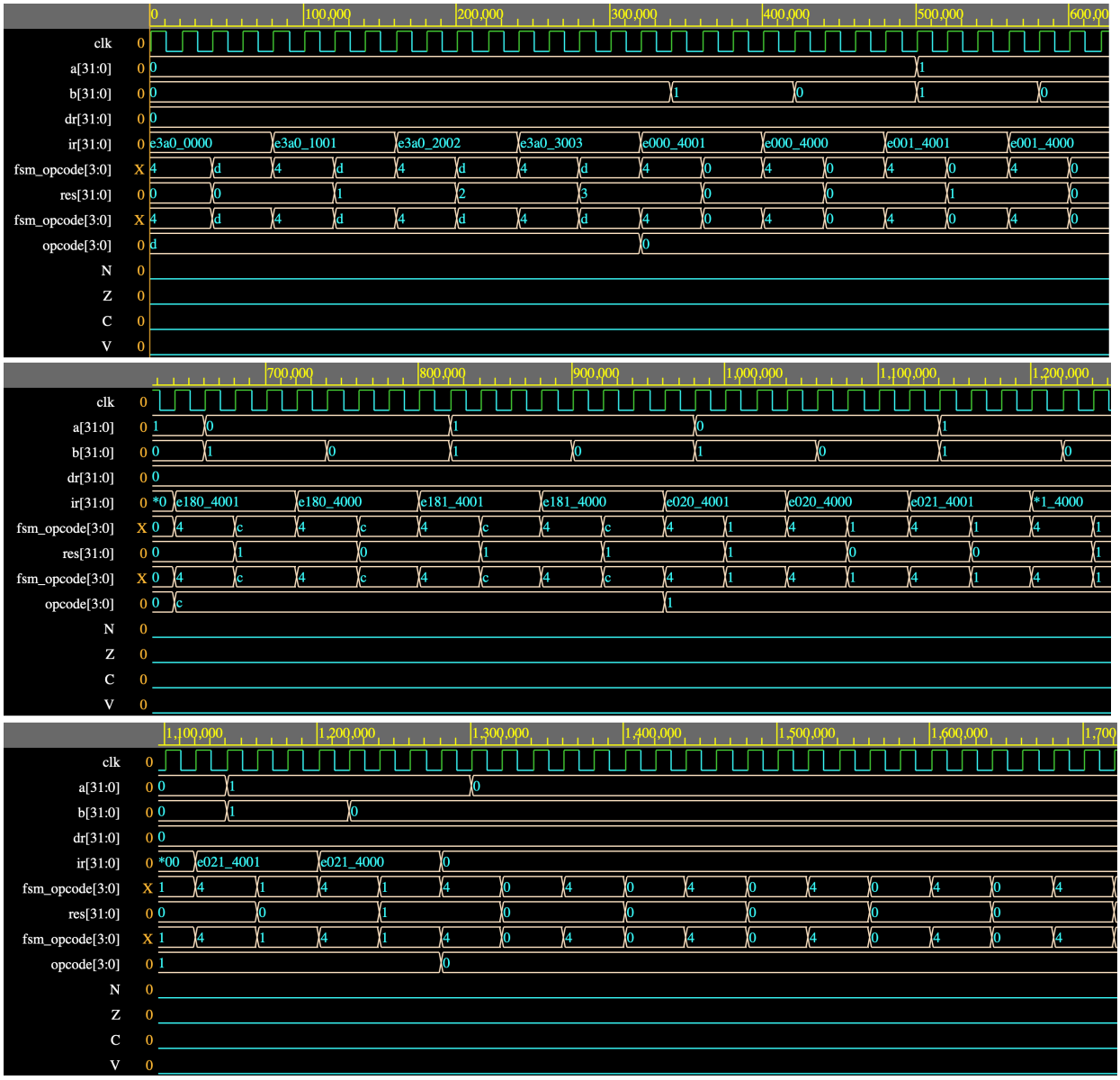
Submission contains 12 program files (and 1 pdf of report) : 1 testbench (testbench.tb) , 1 run.do file, 1 design file (gluelogic.vhd) , 1 MyTypes.vhd file (downloaded from moodle) and 11 other vhd files containing register file, pc, ALU, memory, flags, condition checker, decoder and FSM .

3 Testcases

3.1 Instructions involving logical operations

```
mov r0,#0
mov r1, #1
mov r2, #2
mov r3, #3
and r4, r0,r1
and r4, r0,r0
and r4, r1,r1
and r4, r1,r0
orr r4, r0,r1
orr r4, r0,r0
orr r4, r1,r1
orr r4, r1,r0
eor r4, r0,r1
eor r4, r0,r0
eor r4, r1,r1
eor r4, r1,r0

The following instructions can be copied in program memory to run these commands.
( 0=>X"E3A00000",1=>
X"E3A01001", 2=>
X"E3A02002",3=>
X"E3A03003",4=>
X"E0004001",5=>
X"E0004000",6=>
X"E0014001",7=>
X"E0014000",8=>
X"E1804001",9=>
X"E1804000",10=>
X"E1814001",11=>
X"E1814000",12=>
X"E0204001",13=>
X"E0204000",14=>
```

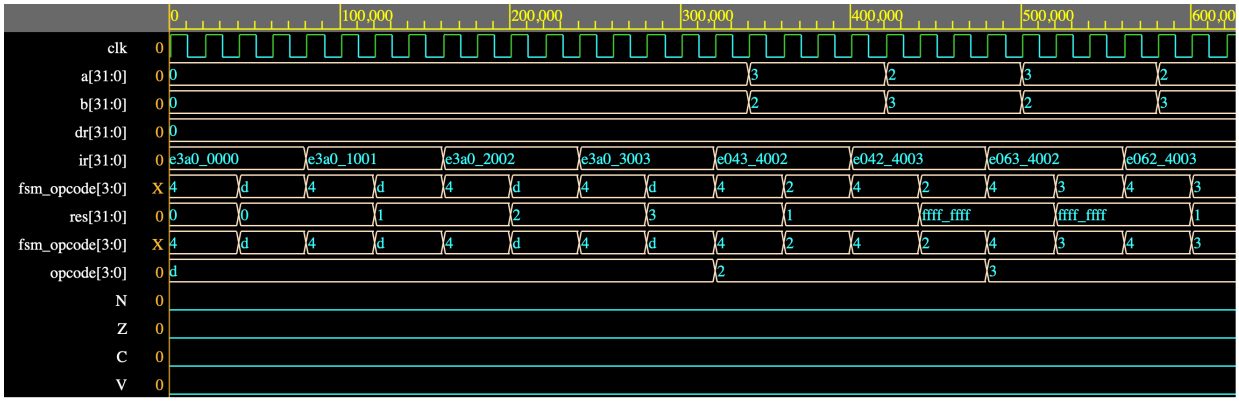
```
X"E0214001",15=>
X"E0214000",

others=>X"00000000");
```

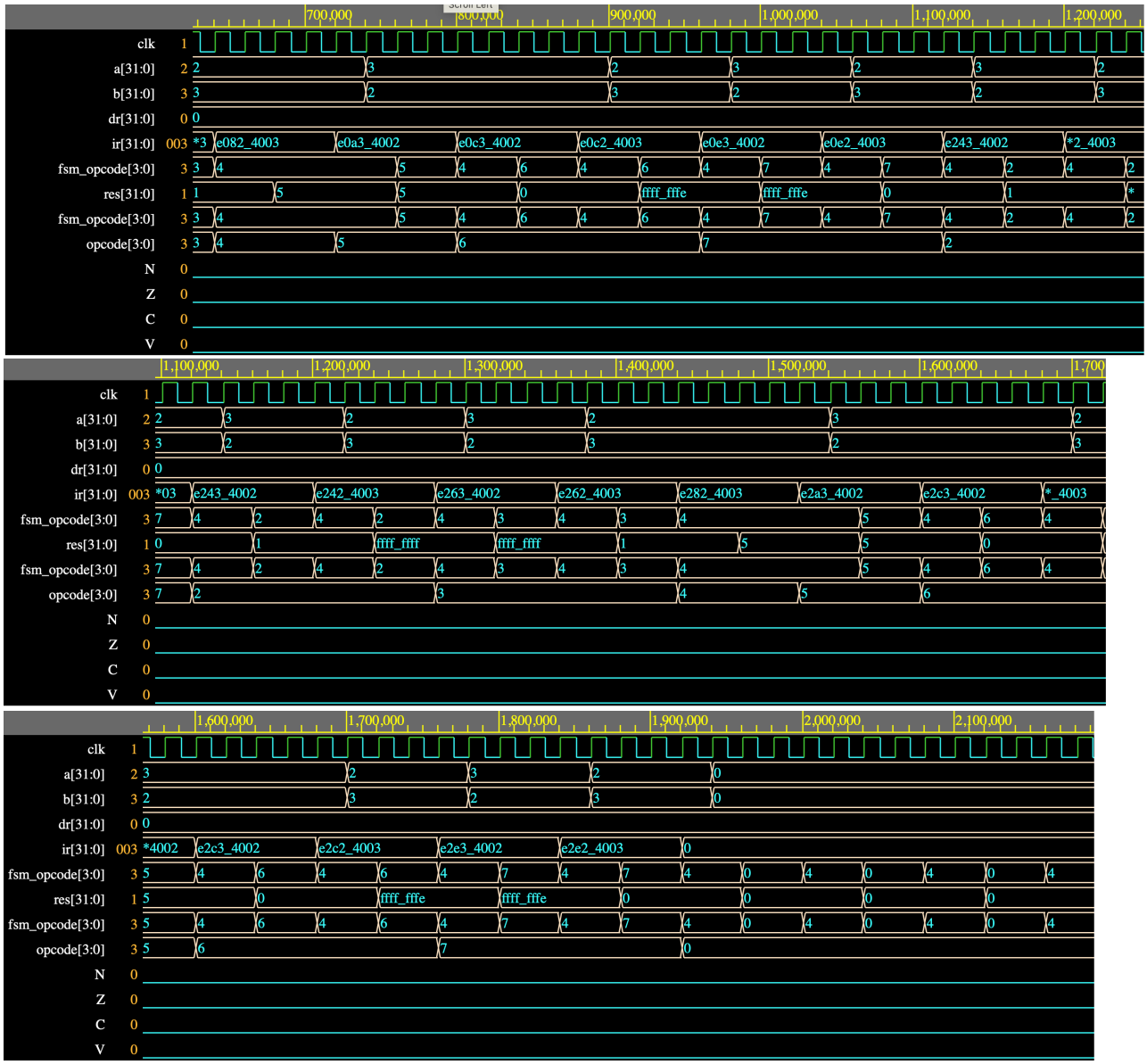


3.2 Instructions involving arithmetic operations

```
mov r0,#0
mov r1, #1
mov r2, #2
mov r3, #3
sub r4, r3,r2
sub r4,r2,r3
rsb r4,r3,r2
rsb r4,r2,r3
add r4,r2,r3
adc r4,r3,r2
sbc r4,r3,r2
sbc r4,r2,r3
rsc r4,r3,r2
rsc r4,r2,r3
```

```
sub r4, r3,#2
sub r4,r2,#3
rsb r4,r3,#2
rsb r4,r2,#3
add r4,r2,#3
adc r4,r3,#2
sbc r4,r3,#2
sbc r4,r2,#3
rsc r4,r3,#2
rsc r4,r2,#3

The following instructions can be copied in program memory to run these commands.
(0=>X"E3A00000",1=>
X"E3A01001",2=>
X"E3A02002",3=>
X"E3A03003",4=>
X"E0434002",5=>
X"E0424003",6=>
X"E0634002",7=>
X"E0624003",8=>
X"E0824003",9=>
X"E0A34002",10=>
X"E0C34002",11=>
X"E0C24003",12=>
X"E0E34002",13=>
X"E0E24003",14=>

X"E2434002",15=>
X"E2424003",16=>
X"E2634002",17=>
X"E2624003",18=>
X"E2824003",19=>
X"E2A34002",20=>
X"E2C34002",21=>
X"E2C24003",22=>
X"E2E34002",23=>
X"E2E24003",
others=>X"00000000");
```

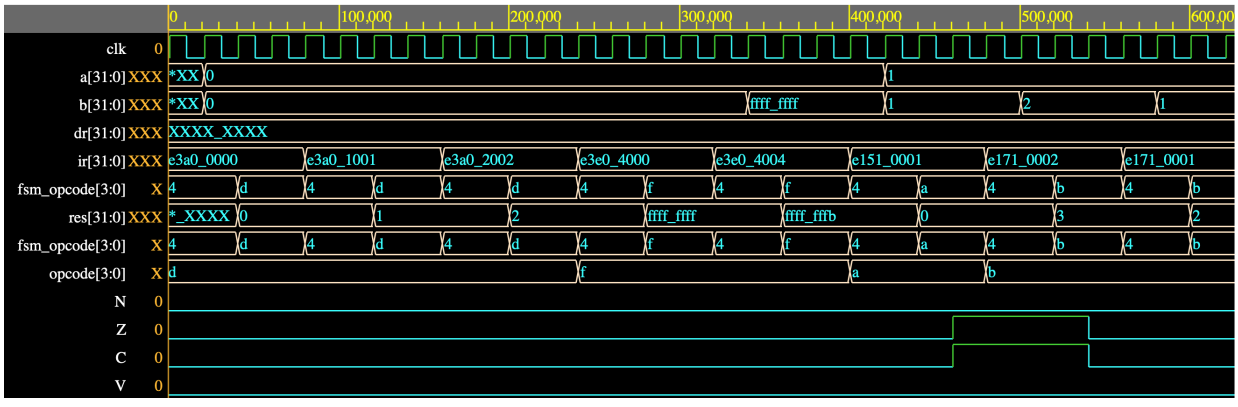


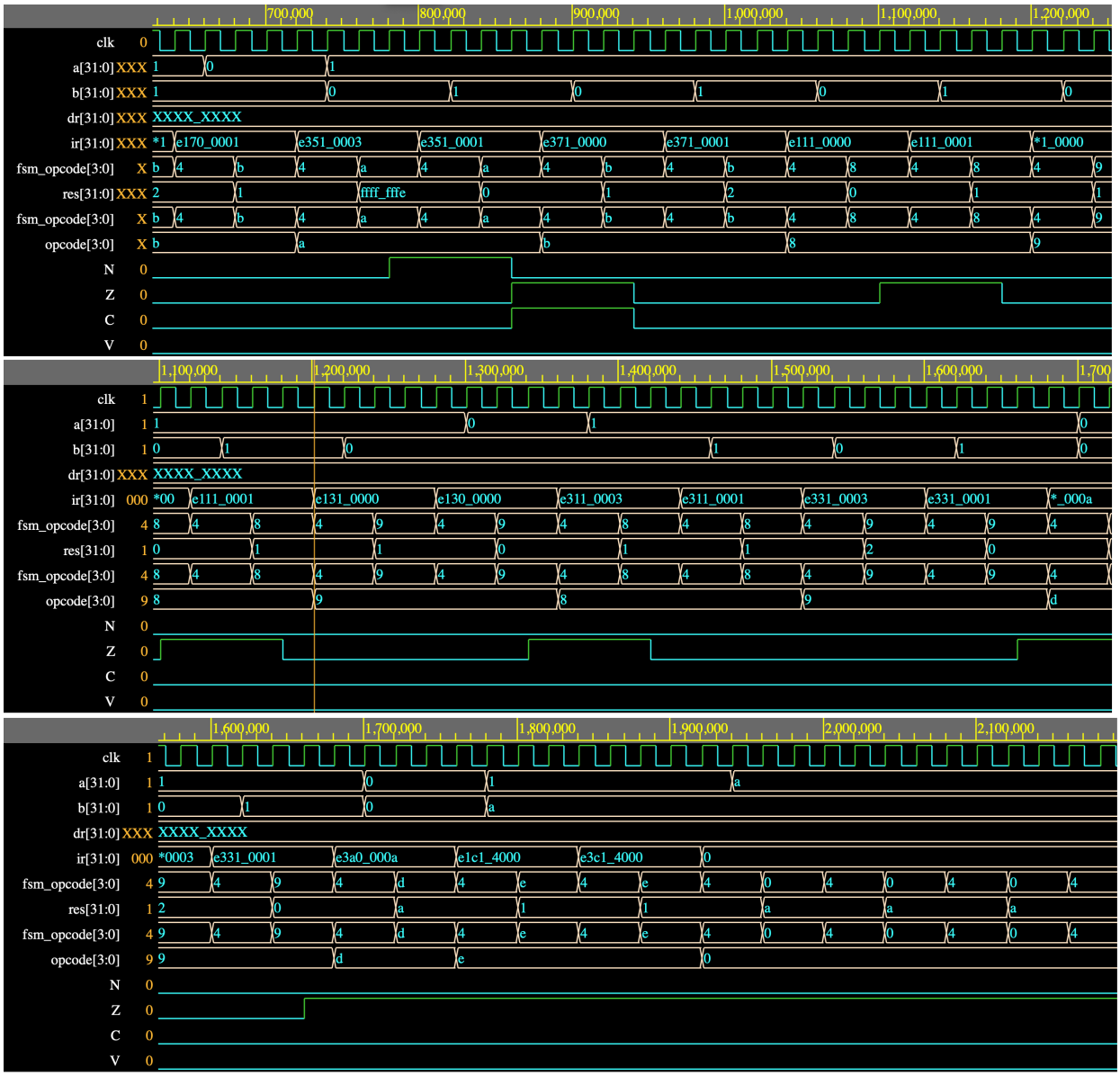


3.3 Instructions modifying flags

```
mov r0,#0
mov r1, #1
mov r2,#2
mvn r4,#0
add r4,r4,#1
mvn r4,#4
cmp r1,r1
cmn r1,r2
cmn r1,r1
cmn r0,r1
cmp r1,#3
cmp r1,#1
cmn r1,#0
cmn r1,#1
tst r1,r0
tst r1,r1
teq r1,r0
teq r0,r0
tst r1,#3
```

```
tst r1,#1
teq r1,#3
teq r1,#1
mov r0,#10
bic r4, r1,r0
bic r4, r1,#0

The following instructions can be copied in program memory to run these commands.
(0=>X"E3A00000",1=>
X"E3A01001",2=>
X"E3A02002" ,3=>
X"E3E04000",4=>
X"E3E04004",5=>

X"E1510001",6=>
X"E1710002",7=>
X"E1710001",8=>
X"E1700001",9=>
X"E3510003",10=>
X"E3510001",11=>
X"E3710000",12=>
X"E3710001",13=>

X"E1110000",14=>
X"E1110001",15=>
X"E1310000",16=>
X"E1300000",17=>
X"E3110003",18=>
X"E3110001",19=>
X"E3310003",20=>
X"E3310001",21=>

X"E3A0000A",22=>
X"E1C14000",23=>
X"E3C14000",
others => X"00000000");
```





4 Conclusion

All outputs and flags have been tallied with the values in ARMSim and everything works fine.