COL216 Assignment 2

Stage 2 report

Aniruddha Deb
2020CS10869

Summary

This project combined the components made in stage 1 with a CPSR register and a condition checker to create a CPU that operates on the specified subset of ARM instructions. Testing was done partly in GHDL/GtkWave (easy to use on Mac, better waveform viewer) and on EDAPlayground (class-mandated, used for synthesis). The code uses the IEEE libraries `numeric_std` and `std_logic_1164`, as well as the `mytypes` library and the `decoder` entity uploaded on moodle.

Note that most files from stage 1 have been retained. New files have been shown in green, while folders remain in blue.

File Structure

```
2020CS10869.zip
├── alu.vhdl
├── alu_tb.vhdl
├── cpu.vhdl
├── cpu_tb.vhdl
├── decoder.vhdl
├── mem.vhdl
├── mem_tb.vhdl
├── mytypes.vhdl
├── regfile.vhdl
├── regfile_tb.vhdl
├── report.pd
├── run.do
├── logs
│   ├── alu_synth_log.txt
│   ├── cpu_synth_log.txt
│   ├── data_mem_synth_log.txt
│   ├── prog_mem_synth_log.txt
│   └── regfile_synth_log.txt
├── test_progs
│   ├── test_prog_1.s
│   ├── test_prog_2.s
│   ├── test_prog_3.s
│   └── test_prog_4.s
├── wave_imgs
```

```
│   ├── alu_test.png
│   ├── cpu_test_prog_1.png
│   ├── cpu_test_prog_2.png
│   ├── cpu_test_prog_3.png
│   ├── cpu_test_prog_4.png
│   ├── memfile_byte_test.png
│   ├── memfile_test.png
│   ├── regfile_altera_test.png
│   └── regfile_test.png
└─ waves
    ├── alu_tb.ghw
    ├── cpu_prog_1.ghw
    ├── cpu_prog_2.ghw
    ├── cpu_prog_3.ghw
    ├── cpu_prog_4.ghw
    ├── data_mem_tb.ghw
    ├── prog_mem_tb.ghw
    └── regfile_tb.ghw
```

Program Details

- `cpu.vhdl` - contains the CPU entity and architecture, along with the CPSR register and condition checker entities
- `mytypes.vhdl` - a collection of useful types, uploaded on moodle
- `decoder.vhdl` - decodes instructions and generates a set of useful flags that are used in the glue logic

Testing

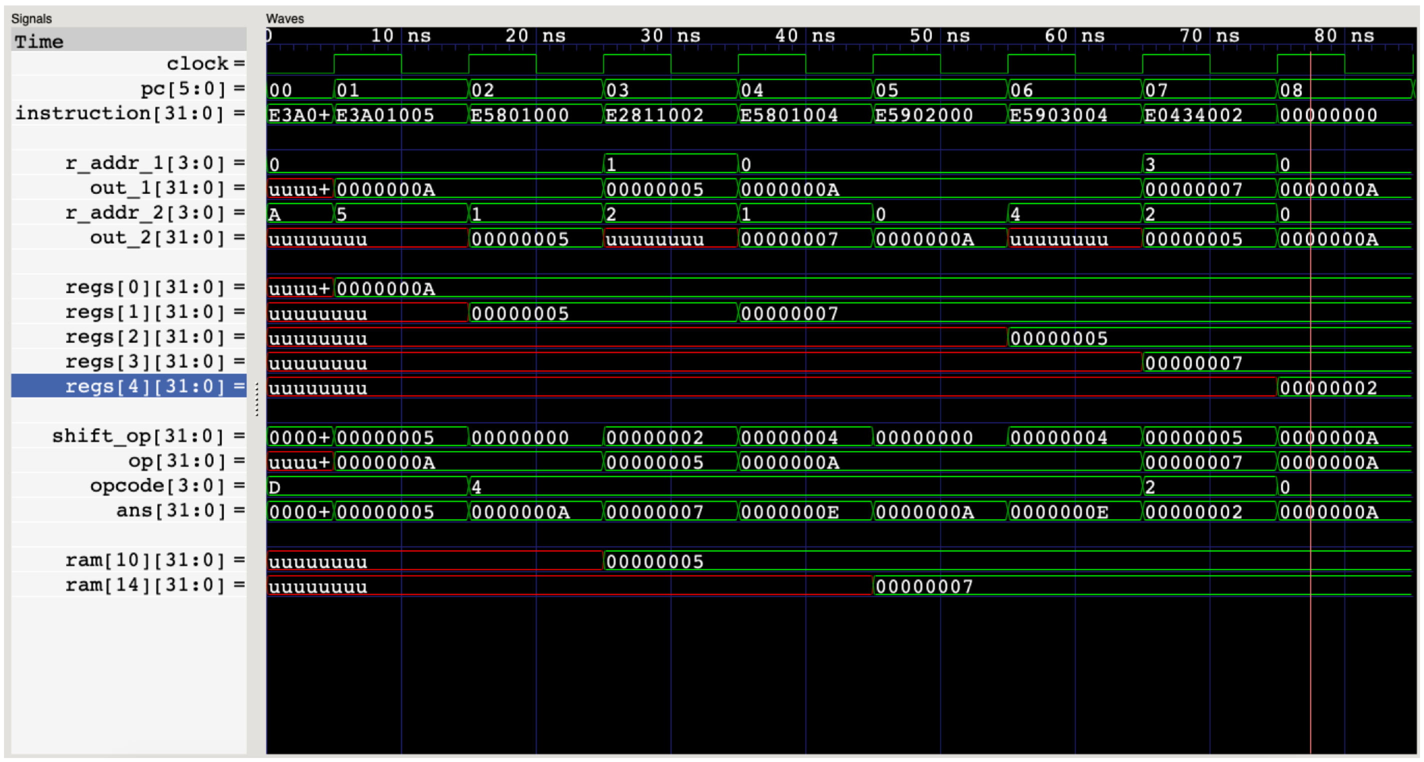
Associated testbenches are marked with `<filename>_tb.vhdl`. The component testbenches are the same as last stage, while the CPU testbench only provides a clock to the CPU component.

The CPU is tested on four programs: each of whose code and instruction sets is shown side by side here, along with a graph of the relevant CPU signals. The graph highlights the internal working of the CPU and serves as the verification for the correctness of the CPU logic

Test 1

Taken From Assignment Description

`0 => X"E3A0000A",`
`1 => X"E3A01005",`
`2 => X"E5801000",`
`3 => X"E2811002",`
`4 => X"E5801004",`
`5 => X"E5902000",`
`6 => X"E5903004",`
`7 => X"E0434002",`
`others => X"00000000"`
	`.text`
	`mov r0, #10`
	`mov r1, #5`
	`str r1, [r0]`
	`add r1, r1, #2`
	`str r1, [r0, #4]`
	`ldr r2, [r0]`
	`ldr r3, [r0, #4]`
	`sub r4, r3, r2`
	`.end`



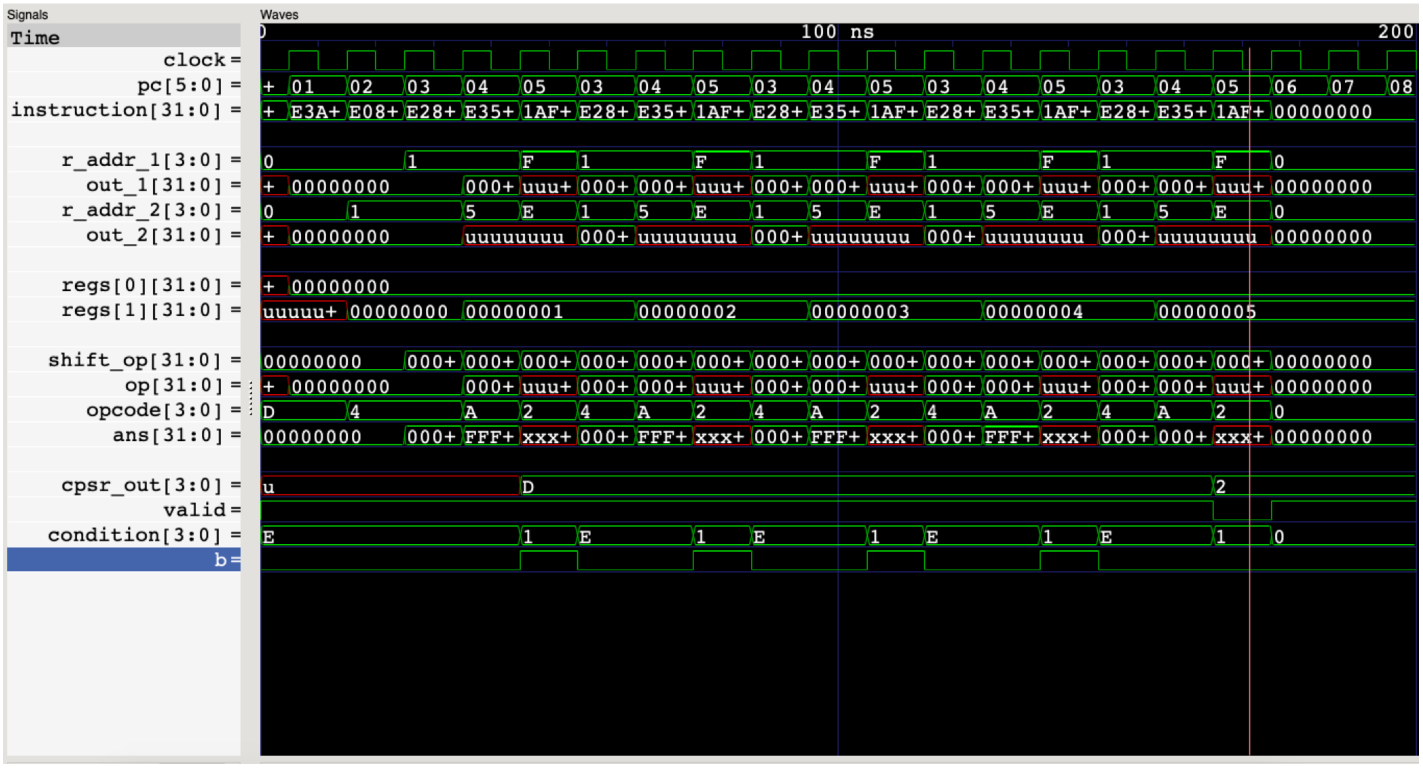
```
Time                0  10 ns  20 ns  30 ns  40 ns  50 ns  60 ns  70 ns  80 ns
clock =
pc[5:0] =           00 01 02 03 04 05 06 07 08
instruction[31:0] = E3A0+ E3A01005 E5801000 E2811002 E5801004 E5902000 E5903004 E0434002 00000000

r_addr_1[3:0] =     0 1 0 3 0
out_1[31:0] =       uuuu+ 0000000A 00000005 0000000A 00000007 0000000A
r_addr_2[3:0] =     A 5 1 2 1 0 4 2 0
out_2[31:0] =       uuuuuuuu 00000005 uuuuuuuu 00000007 0000000A uuuuuuuu 00000005 0000000A

regs[0][31:0] =     uuuu+ 0000000A
regs[1][31:0] =     uuuuuuuu 00000005 00000007
regs[2][31:0] =     uuuuuuuu 00000005
regs[3][31:0] =     uuuuuuuu 00000007
regs[4][31:0] =     uuuuuuuu 00000002

shift_op[31:0] =    0000+ 00000005 00000000 00000002 00000004 00000000 00000004 00000005 0000000A
op[31:0] =          uuuu+ 0000000A 00000005 0000000A 00000007 0000000A
opcode[3:0] =       D 4 2 0
ans[31:0] =         0000+ 00000005 0000000A 00000007 0000000E 0000000A 0000000E 00000002 0000000A

ram[10][31:0] =     uuuuuuuu 00000005
ram[14][31:0] =     uuuuuuuu 00000007
```

Test 2

Taken From Assignment Description

```
0 => X"E3A00000",
1 => X"E3A01000",
2 => X"E0800001",
3 => X"E2811001",
4 => X"E3510005",
5 => X"1AFFFFFB",
others => X"00000000"
```

```
    .text
    mov r0, #0
    mov r1, #0
Loop: add r0, r0, r1
    add r1, r1, #1
    cmp r1, #5
    bne Loop
    .end
```

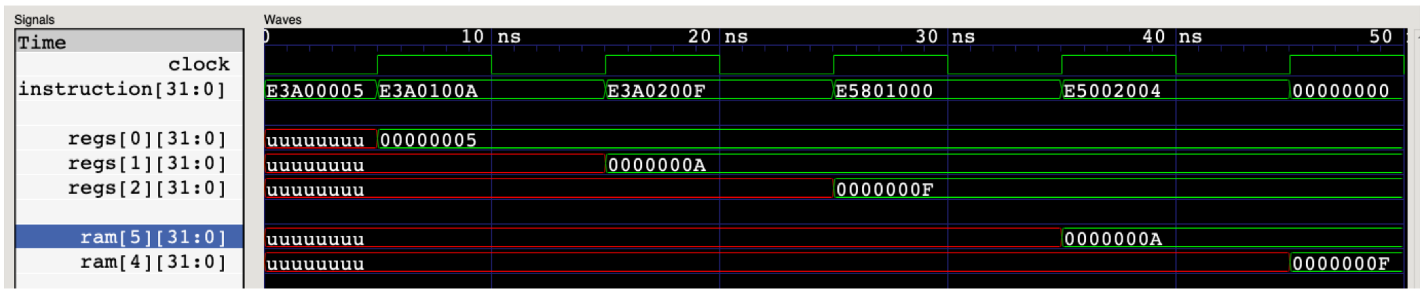


Test 3

Created by Self: Testing negative indexes for load/store operations

```
0 => x"E3A00005",
1 => x"E3A0100A",
2 => x"E3A0200F",
3 => x"E5801000",
4 => x"E5002004",
others => x"00000000"
```

```
    .text
    mov r0, #5
    mov r1, #10
    mov r2, #15
    str r1, [r0]
    str r2, [r0, #-4]
    .end
```

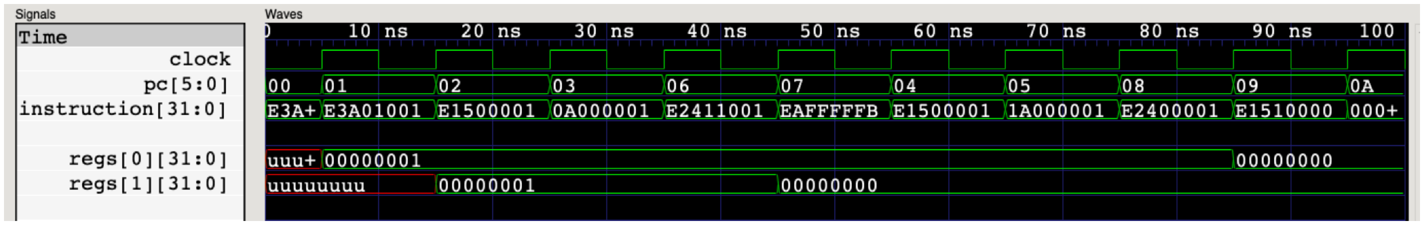


Test 4

Created by Self: Testing all three different branches

```
0 => x"E3A00001",
1 => x"E3A01001",
2 => x"E1500001",
3 => x"0A000001",

4 => x"E1500001",
5 => x"1A000001",

6 => x"E2411001",
7 => x"EAFFFFFB",

8 => x"E2400001",
9 => x"E1510000",
others => x"00000000"
```

```
    .text
    mov r0, #1
    mov r1, #1
    cmp r0, r1
    beq equal
back:
    cmp r0, r1
    bne end
equal:
    sub r1, #1
    b back
end:
    sub r0, #1
    cmp r1, r0
    .end
```



Synthesis

The CPU entity successfully synthesized using Mentor Precision on EDAPlayground. The logs are in cpu_synth_log.txt in logs. **Note that the CPU had only one IO pin, and that was synthesized. Mentor did not synthesize any of the other entities declared and port mapped inside the CPU (ALU, register file, program and data memory etc).**

A excerpt of the synthesis log is shown here

```
# Info: ***************************************************************
# Info: Device Utilization for 7A100TCSG324
# Info: ***************************************************************
# Info: Resource                             Used    Avail   Utilization
# Info: ---------------------------------------------------------------
# Info: IOs                                  1       210       0.48%
# Info: Global Buffers                       0       32        0.00%
# Info: LUTs                                 0       63400     0.00%
# Info: CLB Slices                           0       15850     0.00%
# Info: Dffs or Latches                      0       126800    0.00%
# Info: Block RAMs                           0       135       0.00%
# Info: DSP48E1s                             0       240       0.00%
# Info: ---------------------------------------------------------------
# Info: ******************************************************
# Info: Library: work    Cell: cpu    View: cpu_arch
# Info: ******************************************************
# Info:  Number of ports :                                1
# Info:  Number of nets :                                 0
# Info:  Number of instances :                            0
# Info:  Number of references to this view :              0
# Info: Total accumulated area :
# Info:  Number of gates :                                0
# Info:  Number of accumulated instances :                0
# Info: *****************************
# Info:  IO Register Mapping Report
# Info: *****************************
# Info: Design: work.cpu.cpu_arch
# Info: +----------+-----------+----------+----------+----------+
# Info: | Port     | Direction |   INFF   |  OUTFF   |  TRIFF   |
# Info: +----------+-----------+----------+----------+----------+
# Info: | clock    | Input     |          |          |          |
# Info: +----------+-----------+----------+----------+----------+
# Info: Total registers mapped: 0
```