COL216 Assignment 2

Stage 8 report

Aniruddha Deb
2020CS10869

Summary

This stage completes the CPU: partitioning the memory into user and protected mode, and adding SWI instructions to read bytes from an input port. Reset is also included.

The memory is partitioned into 64 words of protected memory (addresses `0x00-0xFF`) and user memory (addresses `0x100-0x1FF`). The reset instruction is placed at 0x00, and the swi instruction at 0x08. The input port is addressed at 0x0C. The swi subroutine is at 0x10, and we use the swi subroutine to read a byte from the input port and place it in r0, and then return to our user program.

In addition, the flags and full predication has been implemented and debugged. Branch and link has also been implemented using r14 of the register file as the link register. We use `mov pc, lr` as the replacement for ret, so as to make the program compileable by the arm assembler. For the protected mode instructions, `rte` as specified in the assignment note, is used (as we don't compile protected mode instructions repeatedly). `gentest.c` has also been modified to generate programs according to the new memory mapping.

File Structure

```
2020CS10869.zip
├── Makefile
├── alu.vhdl
├── commons.vhdl
├── cond_tb.vhdl
├── cpu.vhdl
├── cpu_controller.vhdl
├── cpu_datapath.vhdl
├── cpu_tb.vhdl
├── gen_datapath.py
├── instr_decoder.vhdl
├── logs
│   └── cpu_synth_log.txt
├── mem.vhdl
├── mult_tb_v2.vhdl
├── multiplier.vhdl
├── mytypes.vhdl
├── pmconnect.vhdl
├── predicator.vhdl
├── protected.s
├── regfile.vhdl
├── report.pdf
```

```
├── run.do
├── shifter.vhdl
├── swi_tb.vhdl
├── test_progs
│   ├── cond_test.s
│   ├── gentest
│   ├── gentest.c
│   ├── mult_test.s
│   ├── ret_test.s
│   └── swi_test.s
├── wave_imgs
│   ├── cond_test.png
│   └── swi_test.png
└── waves
    ├── cond_tb.ghw
    ├── mul_tb_v2.ghw
    └── swi_tb.ghw
```

Testing

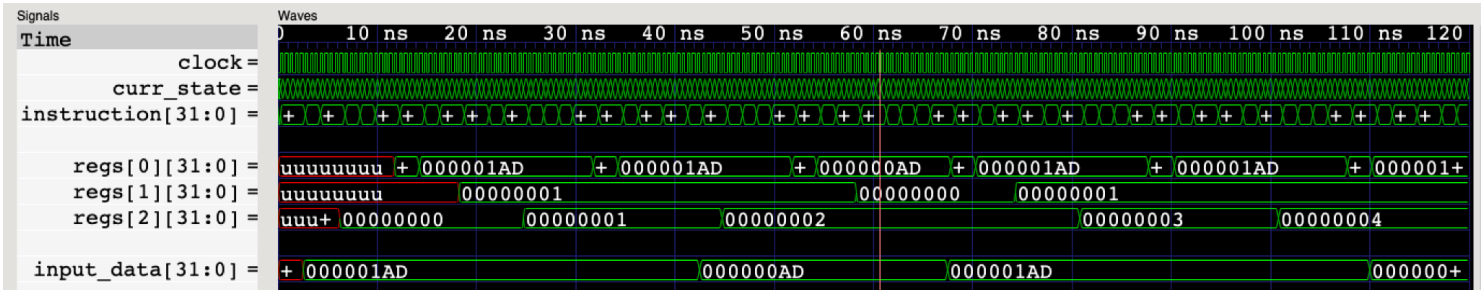
SWI tests

```
    0 => X"E3A0EC01",
    1 => X"E6000011",
    2 => X"EA000000",
    3 => X"00000000",
    4 => X"E3A0000C",
    5 => X"E5900000",
    6 => X"E6000011",

    64 => X"E3A02000",

    65 => X"EF000000",
    66 => X"E1A01420",
    67 => X"E3510001",
    68 => X"1AFFFFFB",
    69 => X"E2822001",
    70 => X"EAFFFFF9",
    others => X"00000000"
```

```
@ swi_test.s
@ check for swi interrupts
@
@
@ protected zone code
@
@
    .text
    mov r2, #0
ol:
il: swi 0x00
    mov r1, r0, LSR #8
    cmp r1, #1
    bne il
    add r2, #1
    b ol
    .end
```



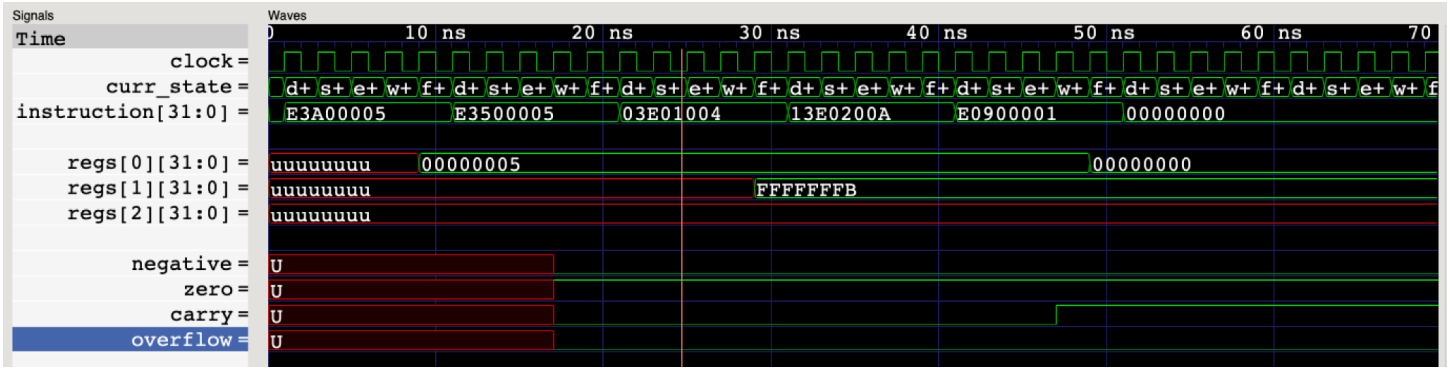
SWI tests

```
0 => X"E3A0EC01",
1 => X"E6000011",
2 => X"EA000000",
3 => X"00000000",
4 => X"E3A0000C",
5 => X"E5900000",
6 => X"E6000011",

64 => X"E3A02000",

65 => X"EF000000",
66 => X"E1A01420",
67 => X"E3510001",
68 => X"1AFFFFFB",
69 => X"E2822001",
70 => X"EAFFFFF9",
others => X"00000000"
```

```
@ swi_test.s
@ check for swi interrupts
@
@
@ protected zone code
@
@
    .text
    mov r2, #0
ol:
il: swi 0x00
    mov r1, r0, LSR #8
    cmp r1, #1
    bne il
    add r2, #1
    b ol
    .end
```



Synthesis

CPU

The CPU entity with the PMConnect module successfully synthesized using Mentor Precision on EDAPlayground. The logs are in cpu_synth_log.txt in logs.

```
# Info: ************************************************************
# Info: Device Utilization for 7A100TCSG324
# Info: ************************************************************
# Info: Resource                          Used    Avail   Utilization
# Info: ------------------------------------------------------------
# Info: IOs                               33      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_multicycle_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_multicycle_arch
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | Port                    | Direction |   INFF   |  OUTFF   |  TRIFF   |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | clock                   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(31)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(30)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(29)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(28)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(27)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(26)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(25)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(24)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(23)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
```

```
# Info: | memory_input_data(22)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(21)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(20)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(19)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(18)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(17)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(16)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(15)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(14)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(13)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(12)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(11)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(10)   | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(9)    | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(8)    | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(7)    | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(6)    | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(5)    | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(4)    | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(3)    | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(2)    | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(1)    | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: | memory_input_data(0)    | Input     |          |          |          |
# Info: +-------------------------+-----------+----------+----------+----------+
# Info: Total registers mapped: 0
# Info: [12022]: Design has no timing constraint and no timing information.
```

Design and Verification

Adding SWI and RTE instructions required the introduction of two new states: execute_swi and execute_rte. Both of these are single cycle changes which change the processor mode, write to either pc or the register file (or both) and then return to fetch, to fetch the next instruction and carry on execution as usual.