COL216 Assignment 2

Stage 5 report

Aniruddha Deb
2020CS10869

Summary

This stage involved creation of a barrel shifter which supports the four shift/rotate operations in ARM assembly: LSL, LSR, ASR and ROR. Along with this, the datapath was restructured and a new state was introduced in the cpu controller. The shifter, although a small component, has several nuances which are hidden away in the ARM reference manual, such as the behavior when shifting from a register (the LSB is taken and if it's greater than 32, some extra operations are done), and other nuances such as setting of the carry bit. I've successfully implemented all of these in the logic, which makes the current implementation ARM compliant. The shifter has also been thoroughly tested with a testbench covering all normal and edge cases.

File Structure

```
.
├── Assignment.jpeg
├── Makefile
├── alu.vhdl
├── commons.vhdl
├── commons_tb.vhdl
├── cpu.vhdl
├── cpu_multicycle.vhdl
├── cpu_synth.v
├── cpu_tb.vhdl
├── fact_tb.vhdl
├── ldr_str_tb.vhdl
├── logs
│   ├── cpu_synth_log.txt
│   └── shifter_synth_log.txt
├── mem.vhdl
├── mult_tb.vhdl
├── mytypes.vhdl
├── predicator.vhdl
├── regfile.vhdl
├── run.do
├── report.pdf
├── shift_tb.vhdl
├── shifter.vhdl
├── shifter_tb.vhdl
├── test_dp_tb.vhdl
├── test_progs
│   ├── fact.s
│   ├── gentest
```

```
│   ├── gentest.c
│   ├── mult.s
│   ├── shift_test.s
│   └── test_dp.s
├── wave_imgs
│   ├── fact.png
│   ├── mult.png
│   ├── shift_test_1.png
│   ├── shift_test_2.png
│   ├── shift_test_3.png
│   └── test_dp.png
└── waves
    ├── ldr_str_tb.ghw
    ├── shift_tb.ghw
    ├── shifter.ghw
    ├── test_dp.ghw
    ├── test_fact.ghw
    └── test_mult.ghw
```

Program Details

- `shifter.vhdl` - implements the shifter
- `shift_tb.vhdl` - testbench for shifter logic (instructions defined in `test_progs/shift_test.s`)

Other program details are the same as previous stages.

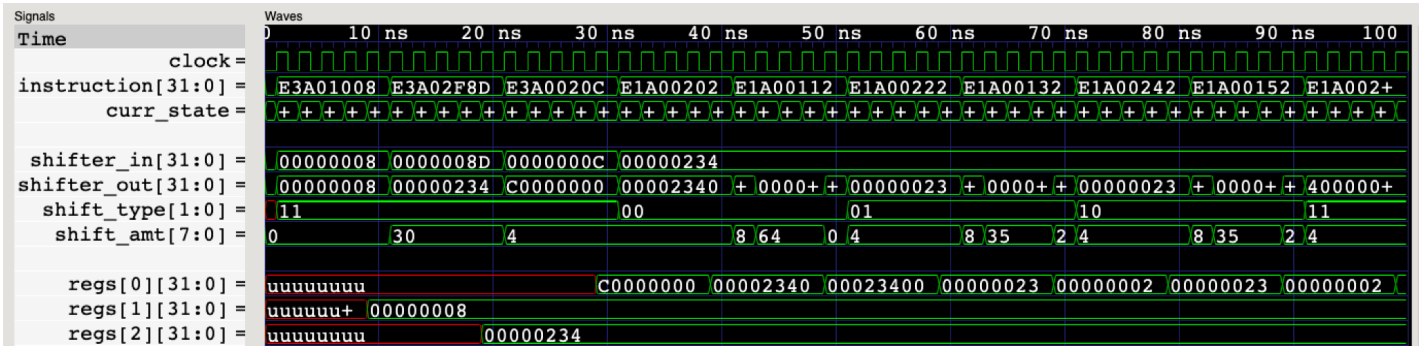
Testing

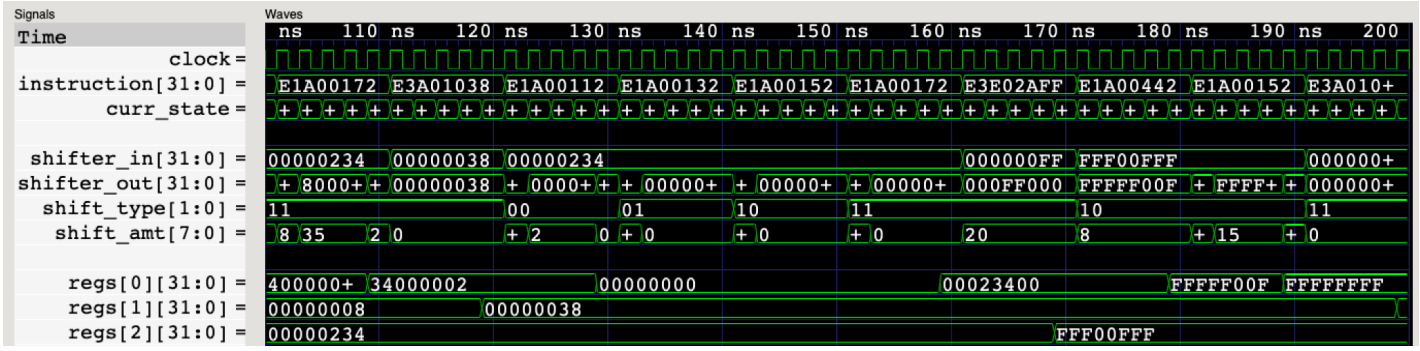
This section used the same tests as stage 4 to check the validity of the CPU, that is, whether it produces the same output as before. In addition, a new test, `shift_test.s`, implementing all the shift operations for DP/DT instructions that are implemented, is used to test the CPU. The description of this test is given below, Along with a line-by-line compiled instruction.

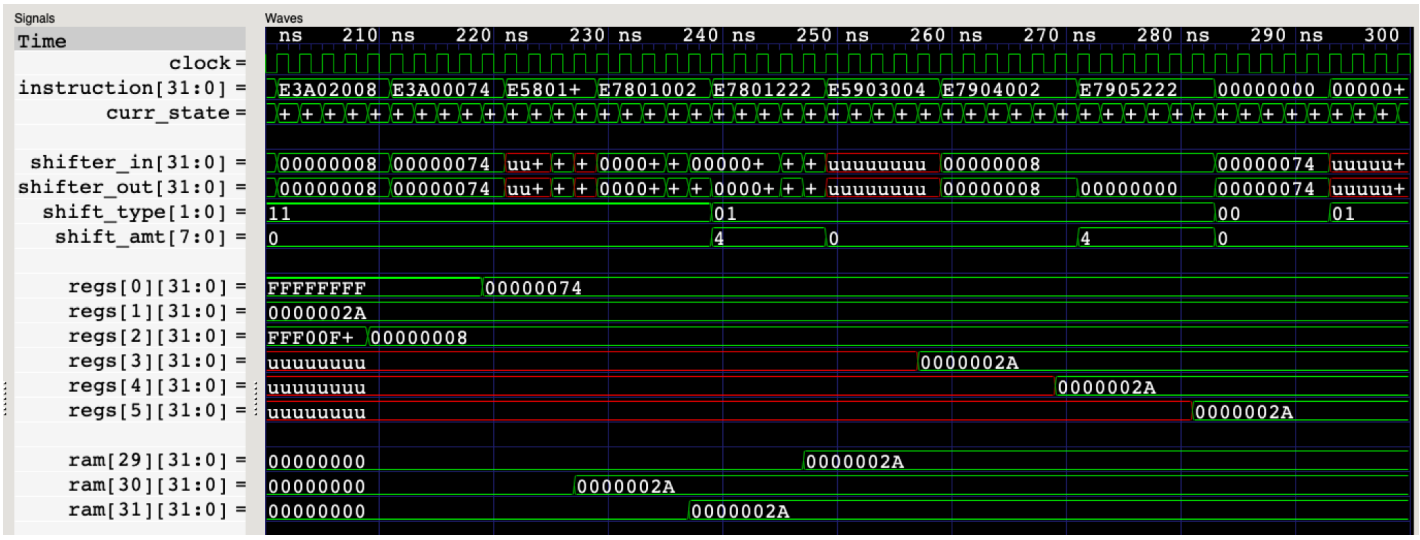
	`@ shift_test.s`
	`@ test file for the shifter`
	
	`    .text`
	
	`@ Data Processsing Shifts:`
	`@ the following types of data processing shifts exist in ARM`
	`@`
	`@ - #<immediate>`
	`@ - <Rm>`
	`@ - <Rm>, LSL #<shift_imm>`
	`@ - <Rm>, LSL <Rs>`
	`@ - <Rm>, LSR #<shift_imm>`
	`@ - <Rm>, LSR <Rs>`
	`@ - <Rm>, ASR #<shift_imm>`
	`@ - <Rm>, ASR <Rs>`
	`@ - <Rm>, ROR #<shift_imm>`
	`@ - <Rm>, ROR <Rs>`
	`@ - <Rm>, RRX`

```
                        @
                        @ most of these are self explanatory, and all have been implemented,
                        except for
                        @ RRX. For Data processing, encoding is in one of three ways:
                        @ 1. 32-bit immediate - an 8 bit constant rotated right by twice of a
                        @    four-bit constant. #<immed_8>, <rotate_amount>
                        @ 2. Immediate Shift - Rm shifted by an immediate constant
                        @ 3. Register Shift - Rm shifted by a register Rs

0 => X"E3A01008",           mov r1, #8 @ we'll use this for register shifts
1 => X"E3A02F8D",           mov r2, #0x234 @ shows immediate shift working, because 0x234
                                    @ won't fit in 8 bits

2 => X"E3A0020C",           mov r0, #12, 4 @ r0 = 1100 >> 4 = 110000...00 (immediate)
3 => X"E1A00202",           mov r0, r2, LSL #4 @ r0 = 0x2340
4 => X"E1A00112",           mov r0, r2, LSL r1 @ r0 = 0x23400
5 => X"E1A00222",           mov r0, r2, LSR #4 @ r0 = 0x23
6 => X"E1A00132",           mov r0, r2, LSR r1 @ r0 = 0x2
7 => X"E1A00242",           mov r0, r2, ASR #4 @ same as above
8 => X"E1A00152",           mov r0, r2, ASR r1 @ same as above
9 => X"E1A00262",           mov r0, r2, ROR #4 @ r0 = 0x40000023
10 => X"E1A00172",          mov r0, r2, ROR r1 @ r0 = 0x34000002

                        @ edge cases: ASR for negatives, shifting out by a value greater than 32

11 => X"E3A01038",          mov r1, #56

12 => X"E1A00112",          mov r0, r2, LSL r1 @ r0 = 0
13 => X"E1A00132",          mov r0, r2, LSR r1 @ r0 = 0
14 => X"E1A00152",          mov r0, r2, ASR r1 @ r0 = 0
15 => X"E1A00172",          mov r0, r2, ROR r1 @ r0 = 0x23400
                                    @ (56 = -8 mod 32, so equiv. to LSL #8)

16 => X"E3E02AFF",          mvn r2, #0x000FF000
17 => X"E1A00442",          mov r0, r2, ASR #8 @ r0 = 0xFFFFF00F
18 => X"E1A00152",          mov r0, r2, ASR r1 @ r0 = 0xFFFFFFFF

                        @ data transfer:
                        @ the following types of data transfer operators exist in ARM
                        @
                        @ 1. [<Rn>, #+/-<offset_12>]
                        @ 2. [<Rn>, +/-<Rm>]
                        @ 3. [<Rn>, +/-<Rm>, <shift> #<shift_imm>]
                        @ 4. [<Rn>, #+/-<offset_12>]!
                        @ 5. [<Rn>, +/-<Rm>]!
                        @ 6. [<Rn>, +/-<Rm>, <shift> #<shift_imm>]!
                        @ 7. [<Rn>], #+/-<offset_12>
                        @ 8. [<Rn>], +/-<Rm>
                        @ 9. [<Rn>], +/-<Rm>, <shift> #<shift_imm>
                        @
                        @ From these operations, only (1), (2), (3) have been implemented in
                        @ this version. We test these implementations below:

19 => X"E3A0102A",          mov r1, #42
20 => X"E3A02008",          mov r2, #8
21 => X"E3A00074",          mov r0, #116
22 => X"E5801004",
23 => X"E7801002",          str r1, [r0, #4]
```

```
24 => X"E7801222",          str r1, [r0, r2]
25 => X"E5903004",          str r1, [r0, r2, LSR #4]

26 => X"E7904002",          ldr r3, [r0, #4]
27 => X"E7905222",          ldr r4, [r0, r2]
                            ldr r5, [r0, r2, LSR #4]



                            .data
28 => X"00000000",      pos: .skip 20
29 => X"00000000",
30 => X"00000000",
31 => X"00000000",
32 => X"00000000",
others => X"00000000"       .end
```







Synthesis

CPU

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).**

```
# 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
```

Shifter

The shifter was also separately designed, synthesized and tested.

```
# Info: ************************************************************
# Info: Device Utilization for 7A100TCSG324
# Info: ************************************************************
# Info: Resource                          Used    Avail   Utilization
# Info: ------------------------------------------------------------
# Info: IOs                               76      210      36.19%
# Info: Global Buffers                    0       32        0.00%
# Info: LUTs                              289     63400     0.46%
# Info: CLB Slices                        59      15850     0.37%
# 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: shifter    View: shifter_arc
# Info: ************************************************************
# Info:  Number of ports :                            76
# Info:  Number of nets :                            426
# Info:  Number of instances :                       383
# Info:  Number of references to this view :           0
# Info: Total accumulated area :
# Info:  Number of LUTs :                            289
# Info:  Number of Primitive LUTs :                  307
# Info:  Number of LUTs with LUTNM/HLUTNM :           36
# Info:  Number of accumulated instances :           383
# Info: ****************************
# Info:  IO Register Mapping Report
# Info: ****************************
# Info: Design: work.shifter.shifter_arc
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | Port               | Direction |   INFF   |  OUTFF   |  TRIFF   |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(31)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(30)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(29)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(28)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
```

```
# Info: | shifter_in(27)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(26)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(25)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(24)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(23)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(22)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(21)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(20)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(19)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(18)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(17)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(16)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(15)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(14)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(13)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(12)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(11)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(10)     | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(9)      | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(8)      | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(7)      | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
```

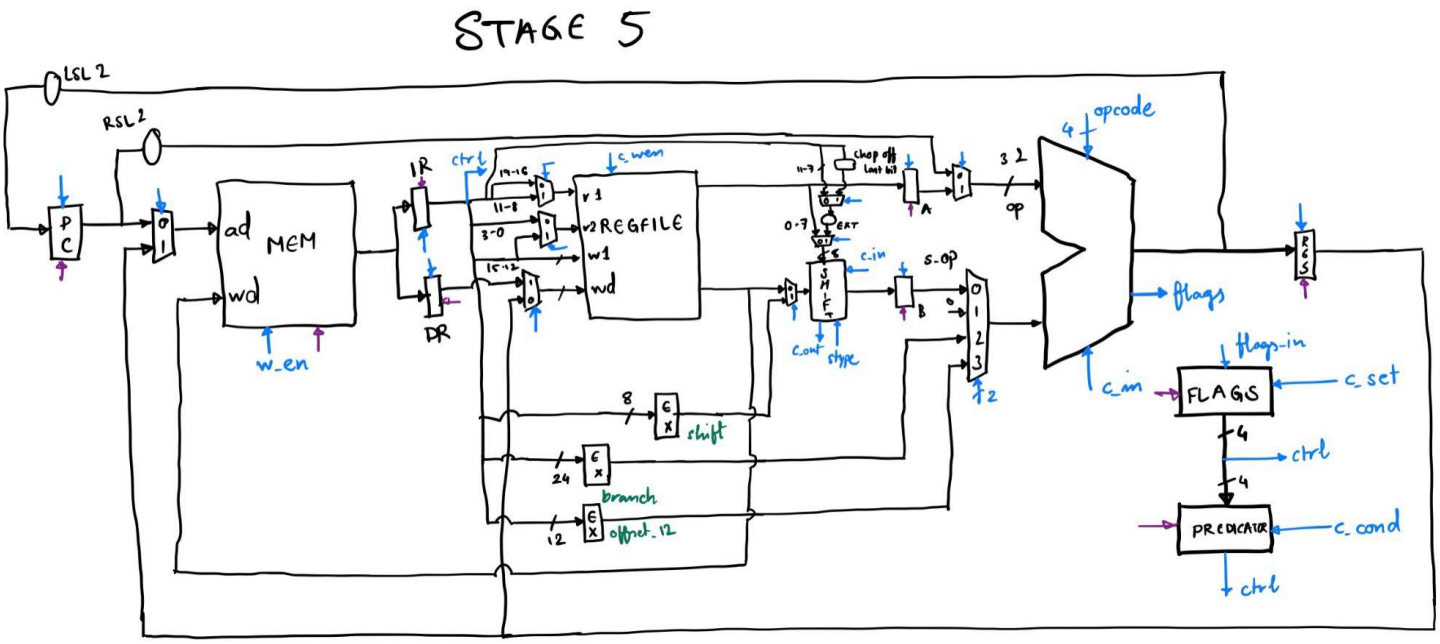
```
# Info: | shifter_in(6)      | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(5)      | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(4)      | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(3)      | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(2)      | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(1)      | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_in(0)      | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(31)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(30)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(29)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(28)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(27)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(26)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(25)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(24)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(23)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(22)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(21)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(20)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(19)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(18)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
```

```
# Info: | shifter_out(17)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(16)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(15)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(14)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(13)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(12)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(11)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(10)    | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(9)     | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(8)     | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(7)     | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(6)     | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(5)     | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(4)     | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(3)     | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(2)     | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(1)     | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shifter_out(0)     | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | carry_in           | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | carry_out          | Output    |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shift_type(1)      | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
```

```
# Info: | shift_type(0)      | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shift_amt(7)       | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shift_amt(6)       | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shift_amt(5)       | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shift_amt(4)       | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shift_amt(3)       | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shift_amt(2)       | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shift_amt(1)       | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: | shift_amt(0)       | Input     |          |          |          |
# Info: +--------------------+-----------+----------+----------+----------+
# Info: Total registers mapped: 0
# Info: [12022]: Design has no timing constraint and no timing information.
```

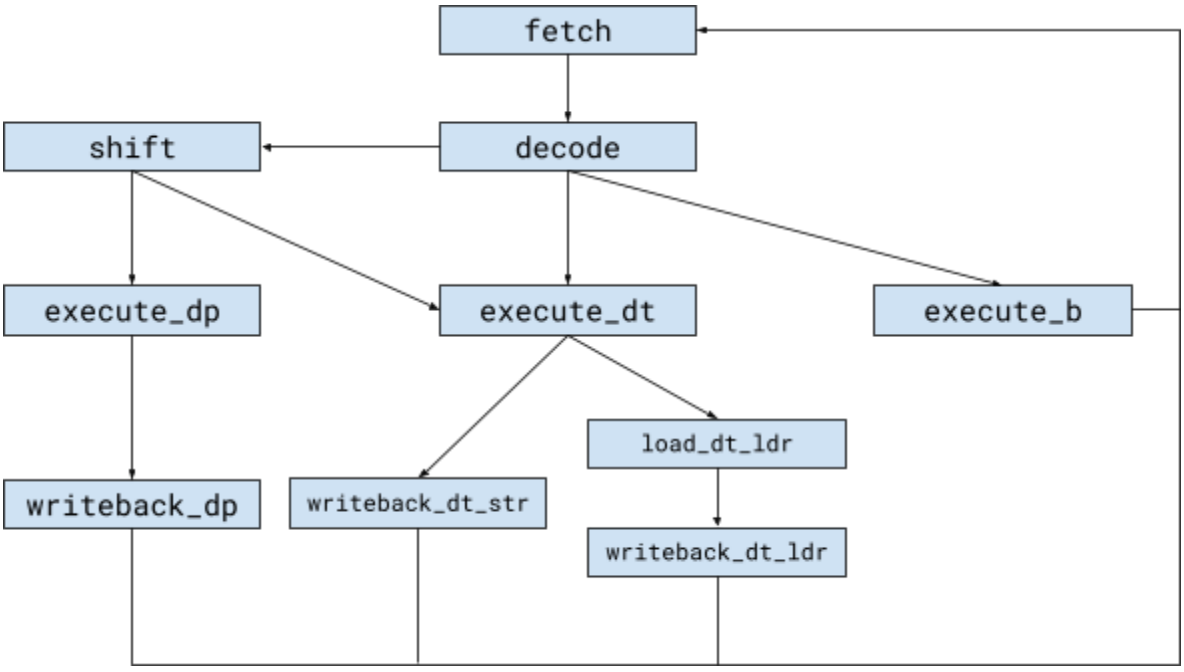
Design and Verification

The design to be implemented was as follows:



Notice the reworking of the datapath post the register file: the datapath includes two sequential multiplexers to allow for changing the shift amount, based on the instruction specified. The flags have also been detached from the ALU to the register file, because we need to account for the carry from the shifter as well for some instructions such as MOV, AND, EOR etc. This will be implemented in a later stage, but the detachment makes it easier to use the same datapath (or with minimum modifications) in the future stages.

The control FSM implemented includes one extra shift state: this allows us to read the shift value from the register file, if needed (in case of a shift by value stored in register, we need to read three registers, and the register file only has two outputs), and also accommodates any delay caused due to the shifter module.


fetch
shift
decode
execute_dp
execute_dt
execute_b
load_dt_ldr
writeback_dp
writeback_dt_str
writeback_dt_ldr