COMP22111: Processor Architecture

Exercise 2 Optional: Stump Shifter

This part of the exercise is optional. It should only be completed if you are ahead, and you have submitted the ALU design. The same deadline applies as for the Stump ALU.

The Stump ISA contains bit-shift instructions, which are only applied to the first operand for a Type 1 instruction. Like the ARM (and unlike most microprocessors) the shifter is placed in series with the ALU and can be treated as a separate component.

In this part exercise you are required to produce a behavioural description of the Stump shifter. A file has been provided for you, Stump_shifter.v, with the interface given in Figure 1.

```
module Stump_shifter (input  wire [15:0] operand,
                      input  wire        c_in,
                      input  wire [1:0]  shift_op,
                      output reg  [15:0] shift_out,
                      output reg         c_out);
```

Figure 1: The Interface to the Stump Shifter

Your Stump processor will still be complete, even if you do not attempt this exercise, as a gate-level design of the Stump shifter is provided in Stump_shifter.v. If you are attempting this exercise then simply comment out the design provided for you (to make this easier remove the line comments '//' before the block comments '/*' and '*/' to make sure you comment out the correct code). Develop the shifter module as a combinatorial logic module similar to the ALU.

The Stump's shifter is simpler than the ARM's and, at most, shifts by one place. Shifts are only applied in the case of Type 1 instructions, with the type of shift being specified by bits [1:0] of the 16-bit instruction (see Table 1). The shift operation is performed on the data specified by srcA. The four operations provided by the shifter are:

- No shift – no shift is applied so the input is passed directly to the output,
- Arithmetic shift right (ASR) – shift of all bits one bit to the right, with the MSB being copied back into the MSB and bit 0 being copied into the shifter carry out,
- Rotate right (ROR) - shift of all bits one bit to the right, bit 0 being copied into the shifter carry out and the MSB,
- Rotate Right through Carry (RRC) - shift of all bits one bit to the right, bit 0 being copied into the carry shifter out register, and the old carry flag is copied to the MSB.

Figure 2 illustrates diagrammatically the three shifting operations.

Your design for the shifter will need testing and debugging (of course – but you know that by now!). You will have to create your own test harness using the test file provided for you - Stump_shifter_test.v. Just the test module has been defined for you. You need to instantiate the design under test and produce your own test stimulus.

shift_op	operation
0 0	No shift
0 1	ASR
1 0	ROR
1 1	RRC

Table 1: shift_op operations

Figure 2: The three Stump shifter operations

Please note: If you start the exercise but fail to produce a working solution, you should revert the design back to the gate-level design provided so that you have a fully working design available for later in the lab.

Submission: ex2b – Stump shifter

Once you have completed you design and tested its operation then you can submit it for marking using the submit script "22111submit" before the deadline for the exercise. Submit as ex2b.

You have now completed the optional exercise for Exercise 2