How We Implemented SRL

This document will walk you through how we added SRL instruction to the MIPS architecture, in the hope of helping you in your implementations. The document will begin with the datapath and controller changes and end with the corresponding code changes. Consider this document as a tutorial for adding instructions to MIPS architecture with System Verilog.

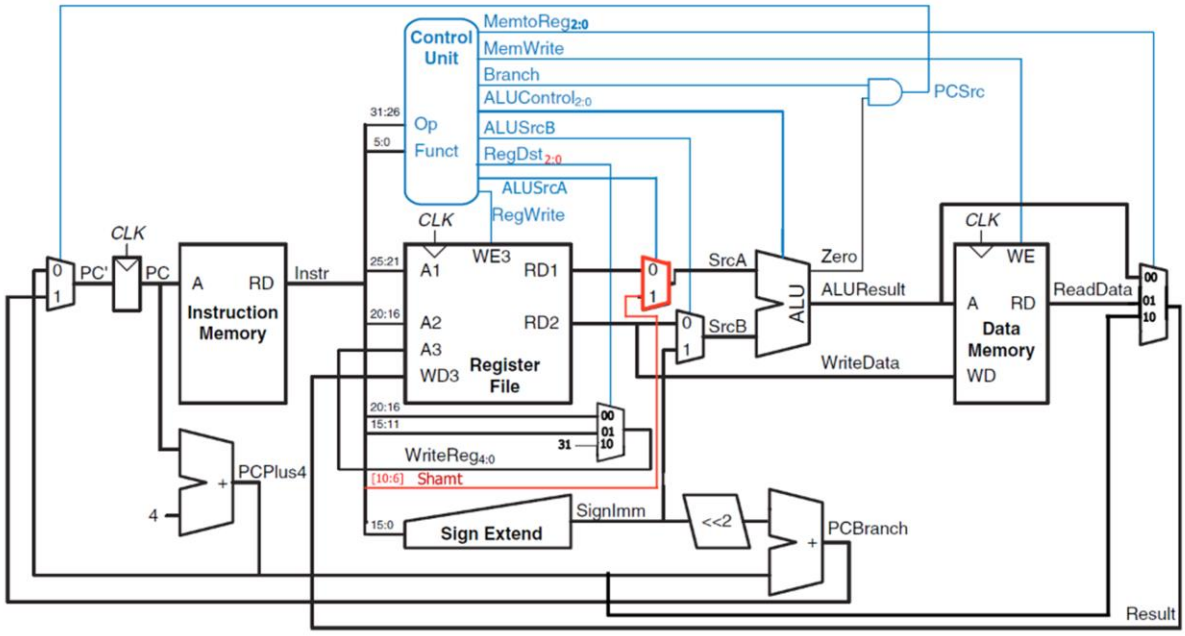

Figure 1: SRL Datapath Changes (marked in red)

ALUOp	Funct	ALUControl
00	X	010 (add)
X1	X	110 (subtract)
1X	100000 (add)	010 (add)
1X	100010 (sub)	110 (subtract)
1X	100100 (and)	000 (and)
1X	100101 (or)	001 (or)
1X	101010 (slt)	111 (set less than)
1X	000010 (srl)	011 (srl)

Table 1: ALU Decoder Changes (marked in red)

SRL is an R-type instruction whose RTL expression is:
IM[PC], RF[rd] <= RF[rt] >> shamt

Inclusion of SRL instruction brings about some changes in the datapath. First, realize that RD2 corresponds to RF[rt]. We needed a new MUX to select SrcA value. For the other instructions, SrcA is always equal to RD1. Yet, for SRL instruction we must be able to give shamt signal as an input to ALU. Since SrcB accepts RD2, we can only use SrcA to represent the shamt value. ALUSrcA signal is created to control this new MUX. To avoid confusion, signal that controls the MUX for SrcB is renamed to ALUSrcB. Note that main decoder table is not modified this time as SRL instruction is

fully represented in the R-type row of the main decoder table. Nevertheless, ALU Decoder Table is altered so that logical right shift operation is supported.

With the datapath and decoder changes described. Let’s move on the code changes.

To begin with, overall you can overview the module parameters and notice that ALUSrcA and ALUSrcB variables are defined when needed.

Firstly, to the controller module, the following line is added:

```
assign alusrca = (funct == 6'b000010); // If it is shift right logic
```

This line makes sure that if the R-type instruction has a funct value of 000010 (i.e. it is the srl instruction, it is the exact same funct value as specified in the MIPS manual), alusrca bit will be 1 and MUX will select the shamt (shift-amount) bits.

Secondly, the following case is added to the switch-case structure in the aludec module:

```
6'b000010: alucontrol  = 3'b011; // SRL
```

Change in the ALU decoder is reflected with that line.

Finally, ALU module is updated to support logical right shift operation with the inclusion of the following snippet.

```
3'b011:
begin
  result = b >> a;
  zero = 0;
end
```