ECE 440 - Homework #3

Collin Heist

3rd February 2020

1 Exercise 3.1

1.1 RegWithBenefits Module

Listing 1.1: Register 'With Benefits' Module

```
`timescale 1ns / 1ps

module regWithBenefits
  #(parameter W = 8)
  (input logic [W-1:0] d,
   input logic rstN, ck, clr, ld, shl, shIn,
   output logic [W-1:0] q);

always_ff @(posedge ck or negedge rstN) begin
  if (~rstN | clr) begin
    q <= 0;
    end
  else if (ld)
    q <= d;
  else if (shl)
    q <= {register[W-2:0], shIn};
end

endmodule
```

1.2 Testbench

Listing 1.2: 8-bit Register Testbench

```
`timescale 1ns / 1ps

module testbench();

// Global Parameters
parameter CLK_PRD = 100;
parameter HOLD_TIME = (CLK_PRD * 0.3);
parameter MAX_SIM_TIME = (100 * CLK_PRD);

logic clock, async_reset, sync_reset, load, shift_left,
   shift_in;
logic [7:0] inputs, reg_out;

regWithBenefits dut(
  .d(inputs),
  .rstN(async_reset),
  .ck(clock),
  .clr(sync_reset),
  .ld(load),
  .shl(shift_left),
  .shIn(shift_in),
  .q(reg_out));

// Prevent simulating longer than MAX_SIM_TIME
initial #(MAX_SIM_TIME) $finish;

// Generate Clock Signal
initial begin
  clock <= 0;
  forever #(CLK_PRD / 2) clock = ~clock;
end

// Main Simulation
initial begin
  inputs = 8'bx; async_reset = 0; sync_reset = 0; load = 0;
     shift_left = 0; shift_in = 0;

  #100; @(posedge clock); #HOLD_TIME; #CLK_PRD;

  // Reset is done, start the simulation
  async_reset = 1; inputs = 8'b11110001; load = 1; #CLK_PRD;
       // Let the register load the inputs

  async_reset = 0; #HOLD_TIME;  // Randomly reset, reg_out
     should be zeros!
  $display(”%2d:␣reg_out=%b”, $stime, reg_out);

  @(posedge clock); #HOLD_TIME; // Sync again

  async_reset = 1; inputs = 8'b10110001; load = 1; #CLK_PRD;
       // Load the inputs once more
  $display(”%2d:␣reg_out=%b”, $stime, reg_out);
```

```
    shift_in = 1; load = 0; shift_left = 1; #CLK_PRD; //
       reg_out should be 0b01100011
    $display("%2d: reg_out=%b", $stime, reg_out);

    $finish;
  end

  endmodule
```

1.3 Simulation Results

Although it might be hard to see, the behavioral simulation results are exactly as expected. The asynchronous reset going low (as it is an active low signal) immediately results in **reg_out** becoming all zeros, and the shifting operation works as expected.

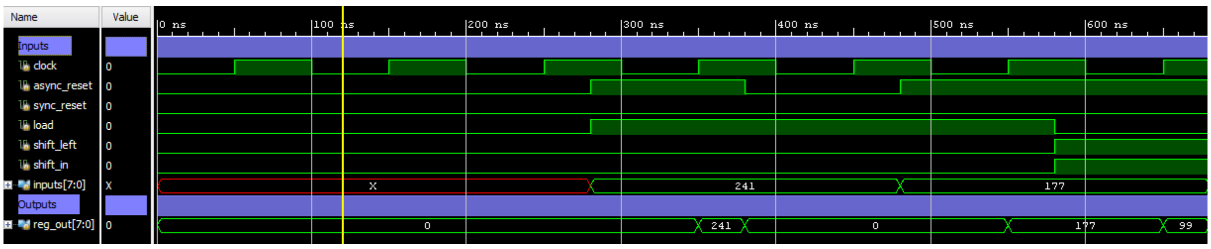
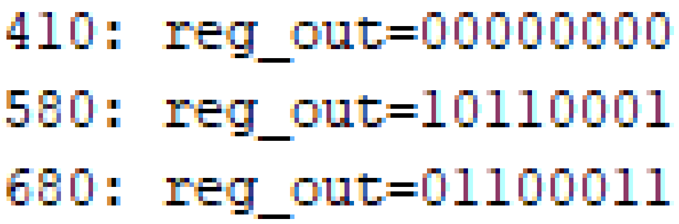


Figure 1.1: Results of the Behavioral Simulation.

The TCL console print statements show the shifting in and shifting left operations working as expected.



```
410: reg_out=00000000
580: reg_out=10110001
680: reg_out=01100011
```

Figure 1.2: Results of the Behavioral Simulation.