Assignment # 3
EC-423 Digital System Design

Submission: 8 Oct 2012
TA

Question 1: Verilog Coding

Design a system which gets a 16-bit signed input on every clock cycle and keeps accumulating these values in a 32-bit accumulator register. The design should also provide the following:

- Output flag indicating the overflow and underflow condition of addition operation
- Two 16-bit registers that keep the updated minimum and maximum values every clock cycle
- A count register that stores the number of cycles it takes the accumulator register to first overflow or underflow.

a. Draw RTL diagram for the architecture
b. Implement the design RTL Verilog and test it for multiple scenarios

Question 2: Verilog Coding Inference

Draw an RTL diagram for the following Verilog code. Clearly specify the data widths of all the wires, and show multiplexers, registers, reset and clock signals.

```
module rtl_design (input [31:0] x0, input [1:0] sel, input clk,rst_n,
output [31:0] out);

reg [31:0] x1, x2, x3;

reg [31:0] y0, y1;

wire [31:0] out1, out2;


assign out1 = x0 + x1;

assign out2 = x2 + x3

assign out = y0 + y1;
```

```
always @(posedge clk or negedge rst_n) begin
     if(!rst_n) begin
          x1 <= 0;
          x2 <= 0;
          x3 <= 0;
     end
     else if (sel==0) begin
          x3 <= x2;
          x2 <= x1;
          x1 <= x0;
     end
     else if (sel == 01) begin
          x3 <= x1;
          x2 <= x0;
          x1 <= x2;
     end
     else begin
          x3 <= x3;
          x2 <= x2;
          x1 <= x0;
     end
end
always @ (posedge clk or negedge rst_n) begin
     if(!rst_n) begin
          y1 <= 0;
          y0 <= 0;
```

```
      end
      else begin
            y1 <= out1 + y0;
            y0 <= y1 + out2;
end
      end
endmodule
```

Question 3: RTL Verilog Coding

Write RTL Verilog Code and stimulus to implement the Multiply Accumulator (MAC) architecture that implements the following:

$$Acc = A \times B + Acc$$

The signals A, B, and Acc are 8, 8, 32-bit wide respectively.

Using the MAC architecture, extend the design to implement the following opcodes

$$Acc = A \times B$$

$$Acc = A + Acc$$

$$Acc = B + Acc$$