CMPE 140 – Laboratory Assignment 1

Hyeran Jeon
Computer Engineering Department, San Jose State University
(This lab is created by Prof. Donald Hung)

System-Level Design Review

Purpose

Review system-level design by designing, functionally verifying, and FPGA prototyping a digital system for accelerating the factorial computation. The system should start execution upon receiving an external input "*Go*" and should output a "*Done*" signal when the execution is completed. In addition, an "*Error*" signal should be set when an input greater than 12 is entered.

Background

1) The algorithm for computing the factorial of *n*, i.e., *n*! = 1 x 2 x … x *n*, is shown below:

```
INPUT n
product = 1
WHILE (n > 1) {
    product = product * n
    n = n -1
    }
OUTPUT product
```

Figure 1. The Factorial of *n* Algorithm

2) Functional building blocks to be used in building the datapath of the system is provided in Fig.2 below. CNT is a down counter with parallel load control and an enable signal; REG is a data register with a load control signal; CMP is a comparator with a GT (greater-than) output; MUL is a combinational multiplier for unsigned integers; MUX is a 2-to-1 multiplexer; BUF is a tri-state buffer for output control, but is recommended to use another 2-to-1 MUX for this purpose.

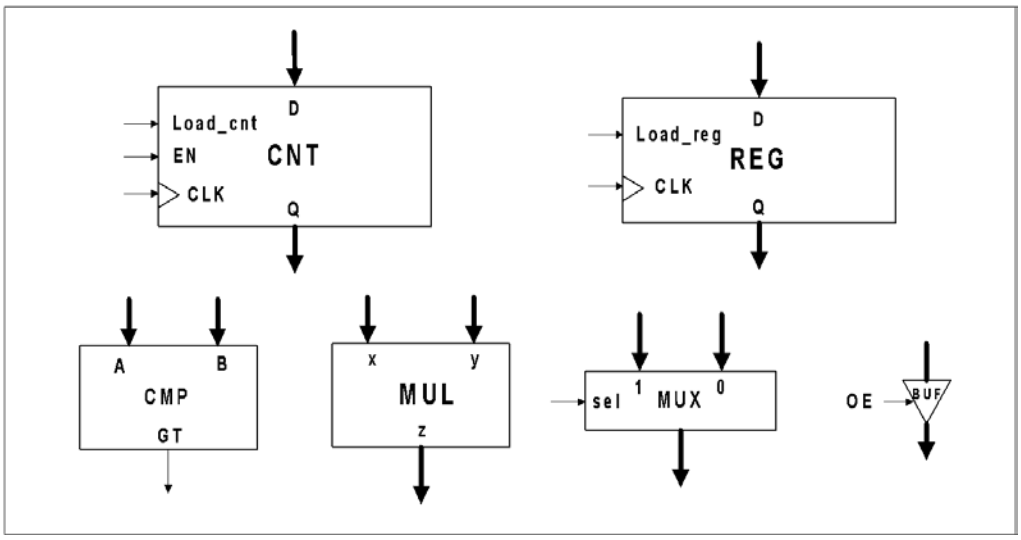

Figure 2. Functional Building Blocks of the Factorial Datapath Module

3) The factorial accelerator shall calculate and display results up to 12! (which is 0x1C8CFC00). Inputs for *n* should be unsigned. Use only one of each building block in Fig. 2 to build the accelerator's datapath (it is recommended to use a second 2-to-1 MUX to replace the output buffer BUF). In order to capture inputs that are greater than 12, an additional comparator (or some simple gate-level logic) will be needed. Except these, no other submodule should exist within the datapath. Parameterize the data path elements to make it easier for scaling the data width.

4) For FPGA validation, the accelerator's 32-bit result should be displayed in hexadecimal (8 digits) using the Basys3 board's four 7-segment LEDs, 4 digits at a time. Verilog source codes for utility modules needed by setting up the validation environment are provided in the Appendix.

Tasks

1) Design the system's datapath using the given building blocks
2) Draw the two-piece (CU-DP) system block diagram that shows all relevant signals
3) Draw the ASM chart which describes the cycle-by-cycle operations of the datapath
4) Draw bubble diagram for the "next state logic" (NS) part of the control unit
5) Construct an output table for the "output logic" part of the control unit
6) Based on the above, write Verilog design code for the factorial accelerator, as well as testbench code to functionally verify your design
7) Implement your design on the Basys 3 FPGA board and test it using the on-board resources.
8) Write a report based on your work, and including the following components:
   (1) Cover page (use photo copy of the cover page signed by the lab TA or the course instructor)
   (2) Purpose of the assignment
   (3) A list of tasks that you have successfully accomplished
   (4) A list of tasks that you cannot or did not accomplish (if there is any), with your explanations and/or analysis.
   (5) Appendices including the following (**Do not** cut and paste from this assignment):

- diagrams and table mentioned in Tasks 1 – 5, generated by Visio or other tools
- source code for both design and verification
- waveforms captured from your simulation results, and photos taken from FPGA validation results

Lab Checkup Schedule

Week	Due to be Checked by TA
Week #1	1. All diagrams/table completed (Tasks 1-5). 2. Verilog code drafted
Week #2	Demonstrate results of 1) Functional verification (Task 6) 2) FPGA validation (Task 7)

Appendix

```
module bin2hex32(
     input wire [31:0] value,
     output wire [3:0] dig0,
     output wire [3:0] dig1,
     output wire [3:0] dig2,
     output wire [3:0] dig3,
     output wire [3:0] dig4,
     output wire [3:0] dig5,
     output wire [3:0] dig6,
     output wire [3:0] dig7
 );

     assign dig0 = value       & 4'hFF;
     assign dig1 = value >>  4 & 4'hFF;
     assign dig2 = value >>  8 & 4'hFF;
     assign dig3 = value >> 12 & 4'hFF;
     assign dig4 = value >> 16 & 4'hFF;
     assign dig5 = value >> 20 & 4'hFF;
     assign dig6 = value >> 24 & 4'hFF;
     assign dig7 = value >> 28 & 4'hFF;
endmodule

module HILO_MUX(
        input  wire [3:0] HI_dig3,
        input  wire [3:0] HI_dig2,
        input  wire [3:0] HI_dig1,
        input  wire [3:0] HI_dig0,

        input  wire [3:0] LO_dig3,
        input  wire [3:0] LO_dig2,
```

```
        input  wire [3:0] LO_dig1,
        input  wire [3:0] LO_dig0,

        input  wire       HILO_sel,

        output wire [3:0] HW_dig3,
        output wire [3:0] HW_dig2,
        output wire [3:0] HW_dig1,
        output wire [3:0] HW_dig0
    );
    assign HW_dig3 = HILO_sel ? HI_dig3 : LO_dig3;
    assign HW_dig2 = HILO_sel ? HI_dig2 : LO_dig2;
    assign HW_dig1 = HILO_sel ? HI_dig1 : LO_dig1;
    assign HW_dig0 = HILO_sel ? HI_dig0 : LO_dig0;
endmodule

module hex_to_7seg(number, s0, s1, s2, s3, s4, s5, s6);
    output s0, s1, s2, s3, s4, s5, s6;
    input [3:0] number;
    reg s0, s1, s2, s3, s4, s5, s6;
    always @ (number)
    begin // BCD to 7-segment decoding
        case (number) // s0 - s6 are active low
            4'h0: begin s0=0; s1=0; s2=0; s3=0; s4=0; s5=0; s6=1; end
            4'h1: begin s0=1; s1=0; s2=0; s3=1; s4=1; s5=1; s6=1; end
            4'h2: begin s0=0; s1=0; s2=1; s3=0; s4=0; s5=1; s6=0; end
            4'h3: begin s0=0; s1=0; s2=0; s3=0; s4=1; s5=1; s6=0; end
            4'h4: begin s0=1; s1=0; s2=0; s3=1; s4=1; s5=0; s6=0; end
            4'h5: begin s0=0; s1=1; s2=0; s3=0; s4=1; s5=0; s6=0; end
            4'h6: begin s0=0; s1=1; s2=0; s3=0; s4=0; s5=0; s6=0; end
            4'h7: begin s0=0; s1=0; s2=0; s3=1; s4=1; s5=1; s6=1; end
            4'h8: begin s0=0; s1=0; s2=0; s3=0; s4=0; s5=0; s6=0; end
            4'h9: begin s0=0; s1=0; s2=0; s3=1; s4=1; s5=0; s6=0; end
            4'ha: begin s0=0; s1=0; s2=0; s3=0; s4=0; s5=1; s6=0; end
            4'hb: begin s0=1; s1=1; s2=0; s3=0; s4=0; s5=0; s6=0; end
            4'hc: begin s0=1; s1=1; s2=1; s3=0; s4=0; s5=1; s6=0; end
            4'hd: begin s0=1; s1=0; s2=0; s3=0; s4=0; s5=1; s6=0; end
            4'he: begin s0=0; s1=0; s2=1; s3=0; s4=0; s5=0; s6=0; end
            4'hf: begin s0=0; s1=1; s2=1; s3=1; s4=0; s5=0; s6=0; end
            default: begin s0=1; s1=1; s2=1; s3=1; s4=1; s5=1; s6=1; end
        endcase
    end
endmodule
```

```
## This file is a general .xdc for the Basys3 rev B board
## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
          set_property IOSTANDARD LVCMOS33 [get_ports clk]
          create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5}
[get_ports clk]

## Switches
set_property PACKAGE_PIN V17 [get_ports HILO_sel]
          set_property IOSTANDARD LVCMOS33 [get_ports HILO_sel]
#set_property PACKAGE_PIN V16 [get_ports {sw[1]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
#set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
#set_property PACKAGE_PIN W17 [get_ports {sw[3]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
#set_property PACKAGE_PIN W15 [get_ports {sw[4]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
#set_property PACKAGE_PIN V15 [get_ports {sw[5]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
#set_property PACKAGE_PIN W14 [get_ports {sw[6]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
#set_property PACKAGE_PIN W13 [get_ports {sw[7]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
#set_property PACKAGE_PIN V2 [get_ports {sw[8]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]}]
#set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
#set_property PACKAGE_PIN T2 [get_ports {sw[10]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
#set_property PACKAGE_PIN R3 [get_ports {sw[11]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]
#set_property PACKAGE_PIN W2 [get_ports {sw[12]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
#set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
#set_property PACKAGE_PIN T1 [get_ports {sw[14]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[14]}]
#set_property PACKAGE_PIN R2 [get_ports {sw[15]}]
          #set_property IOSTANDARD LVCMOS33 [get_ports {sw[15]}]
```

```
##7 segment display
set_property PACKAGE_PIN W7 [get_ports {LEDOUT[0]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {LEDOUT[0]}]
set_property PACKAGE_PIN W6 [get_ports {LEDOUT[1]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {LEDOUT[1]}]
set_property PACKAGE_PIN U8 [get_ports {LEDOUT[2]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {LEDOUT[2]}]
set_property PACKAGE_PIN V8 [get_ports {LEDOUT[3]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {LEDOUT[3]}]
set_property PACKAGE_PIN U5 [get_ports {LEDOUT[4]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {LEDOUT[4]}]
set_property PACKAGE_PIN V5 [get_ports {LEDOUT[5]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {LEDOUT[5]}]
set_property PACKAGE_PIN U7 [get_ports {LEDOUT[6]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {LEDOUT[6]}]

#set_property PACKAGE_PIN V7 [get_ports dp]
          #set_property IOSTANDARD LVCMOS33 [get_ports dp]

set_property PACKAGE_PIN U2 [get_ports {LEDSEL[0]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {LEDSEL[0]}]
set_property PACKAGE_PIN U4 [get_ports {LEDSEL[1]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {LEDSEL[1]}]
set_property PACKAGE_PIN V4 [get_ports {LEDSEL[2]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {LEDSEL[2]}]
set_property PACKAGE_PIN W4 [get_ports {LEDSEL[3]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {LEDSEL[3]}]

##Buttons
set_property PACKAGE_PIN U18 [get_ports rst]
          set_property IOSTANDARD LVCMOS33 [get_ports rst]
```

Top Binding for Prototyping on Basys3 Board:

```
module bin2hex32(
 input wire [31:0] value,
 output wire [3:0] dig0,
 output wire [3:0] dig1,
 output wire [3:0] dig2,
 output wire [3:0] dig3,
 output wire [3:0] dig4,
 output wire [3:0] dig5,
 output wire [3:0] dig6,
 output wire [3:0] dig7
 );

 assign dig0 = value       & 4'hFF;
 assign dig1 = value >>  4 & 4'hFF;
 assign dig2 = value >>  8 & 4'hFF;
 assign dig3 = value >> 12 & 4'hFF;
 assign dig4 = value >> 16 & 4'hFF;
 assign dig5 = value >> 20 & 4'hFF;
 assign dig6 = value >> 24 & 4'hFF;
 assign dig7 = value >> 28 & 4'hFF;

endmodule

module clk_gen(clk50MHz, rst, clksec4, clk_5KHz);
input clk50MHz, rst;
output clksec4, clk_5KHz;
reg clksec4, clk_5KHz;
integer count, count1;
always@(posedge clk50MHz)
begin
    if(rst)
    begin
        count = 0;
        count1 = 0;
        clksec4 = 0;
        clk_5KHz =0;
    end
    else
    begin
        if(count == 100000000)
        begin
            clksec4 = ~clksec4;
            count = 0;
        end
```

```
        if(count1 == 20000)
        begin
            clk_5KHz = ~clk_5KHz;
            count1 = 0;
        end
        count = count + 1;
        count1 = count1 + 1;
    end
end
endmodule

module hex2led(number, s0, s1, s2, s3, s4, s5, s6);
output s0, s1, s2, s3, s4, s5, s6;
input [3:0] number;
reg s0, s1, s2, s3, s4, s5, s6;
always @ (number)
begin // BCD to 7-segment decoding
case (number) // s0 - s6 are active low

4'h0: begin s0=0; s1=0; s2=0; s3=0; s4=0; s5=0; s6=1; end
4'h1: begin s0=1; s1=0; s2=0; s3=1; s4=1; s5=1; s6=1; end
4'h2: begin s0=0; s1=0; s2=1; s3=0; s4=0; s5=1; s6=0; end
4'h3: begin s0=0; s1=0; s2=0; s3=0; s4=1; s5=1; s6=0; end
4'h4: begin s0=1; s1=0; s2=0; s3=1; s4=1; s5=0; s6=0; end
4'h5: begin s0=0; s1=1; s2=0; s3=0; s4=1; s5=0; s6=0; end
4'h6: begin s0=0; s1=1; s2=0; s3=0; s4=0; s5=0; s6=0; end
4'h7: begin s0=0; s1=0; s2=0; s3=1; s4=1; s5=1; s6=1; end
4'h8: begin s0=0; s1=0; s2=0; s3=0; s4=0; s5=0; s6=0; end
4'h9: begin s0=0; s1=0; s2=0; s3=1; s4=1; s5=0; s6=0; end
4'ha: begin s0=0; s1=0; s2=0; s3=0; s4=0; s5=1; s6=0; end
4'hb: begin s0=1; s1=1; s2=0; s3=0; s4=0; s5=0; s6=0; end
4'hc: begin s0=1; s1=1; s2=1; s3=0; s4=0; s5=1; s6=0; end
4'hd: begin s0=1; s1=0; s2=0; s3=0; s4=0; s5=1; s6=0; end
4'he: begin s0=0; s1=0; s2=1; s3=0; s4=0; s5=0; s6=0; end
4'hf: begin s0=0; s1=1; s2=1; s3=1; s4=0; s5=0; s6=0; end
default: begin s0=1; s1=1; s2=1; s3=1; s4=1; s5=1; s6=1; end
endcase
end
endmodule // end led

module LED_MUX (clk, rst, LED0, LED1, LED2, LED3, LEDOUT, LEDSEL);
input clk, rst;
input [6:0] LED0, LED1, LED2, LED3;
output[3:0] LEDSEL;
output[6:0] LEDOUT;
reg [3:0] LEDSEL;
```

```
reg [6:0] LEDOUT;
reg [1:0] index;
always @(posedge clk)
begin
    if(rst)
        index = 0;
    else
        index = index + 1;
end
always @(index or LED0 or LED1 or LED2 or LED3)
begin
    case(index)
    0:  begin
        LEDSEL = 4'b1110;
        LEDOUT = LED0;
        end
    1:  begin
        LEDSEL = 4'b1101;
        LEDOUT = LED1;
        end
    2:  begin
        LEDSEL = 4'b1011;
        LEDOUT = LED2;
        end
    3:  begin
        LEDSEL = 4'b0111;
        LEDOUT = LED3;
        end
    default: begin
                LEDSEL = 0; LEDOUT = 0;
    end
    endcase
end
endmodule

module top(
    input clk,
    input rst,
    input HILO_sel,
    output [3:0] LEDSEL,
    output [6:0] LEDOUT
);
    reg [31:0] test_val;
//    initial test_val = 32'hfedc_3210;
    initial test_val = 32'hba98_7654;
```

```
wire clk_5kHz;
wire [3:0] dig0, dig1, dig2, dig3, dig4, dig5, dig6, dig7;
wire [3:0] HEX3, HEX2, HEX1, HEX0;
wire [6:0] LED3, LED2, LED1, LED0;

clk_gen U0 (
    .clk50MHz(clk),
    .rst(rst),
    //clksec4,
    .clk_5KHz(clk_5kHz)
);

bin2hex32 U1(
     .value(test_val),
     .dig0(dig0),
     .dig1(dig1),
     .dig2(dig2),
     .dig3(dig3),
     .dig4(dig4),
     .dig5(dig5),
     .dig6(dig6),
     .dig7(dig7)
 );

HILO_MUX DUT(
    .HI_dig3(dig7),
    .HI_dig2(dig6),
    .HI_dig1(dig5),
    .HI_dig0(dig4),
    .LO_dig3(dig3),
    .LO_dig2(dig2),
    .LO_dig1(dig1),
    .LO_dig0(dig0),
    .HILO_sel(HILO_sel),
    .HW_dig3(HEX3),
    .HW_dig2(HEX2),
    .HW_dig1(HEX1),
    .HW_dig0(HEX0)
);


hex2led U_LD_3(
    .number(HEX3),
    .s0(LED3[0]),
    .s1(LED3[1]),
    .s2(LED3[2]),
```

```
        .s3(LED3[3]),
        .s4(LED3[4]),
        .s5(LED3[5]),
        .s6(LED3[6])
    );
    hex2led U_LD_2(
        .number(HEX2),
        .s0(LED2[0]),
        .s1(LED2[1]),
        .s2(LED2[2]),
        .s3(LED2[3]),
        .s4(LED2[4]),
        .s5(LED2[5]),
        .s6(LED2[6])
    );
    hex2led U_LD_1(
        .number(HEX1),
        .s0(LED1[0]),
        .s1(LED1[1]),
        .s2(LED1[2]),
        .s3(LED1[3]),
        .s4(LED1[4]),
        .s5(LED1[5]),
        .s6(LED1[6])
    );
    hex2led U_LD_0(
        .number(HEX0),
        .s0(LED0[0]),
        .s1(LED0[1]),
        .s2(LED0[2]),
        .s3(LED0[3]),
        .s4(LED0[4]),
        .s5(LED0[5]),
        .s6(LED0[6])
    );

    LED_MUX U3(
        .clk(clk_5kHz),
        .rst(rst),
        .LED0(LED0),
        .LED1(LED1),
        .LED2(LED2),
        .LED3(LED3),
        .LEDOUT(LEDOUT),
        .LEDSEL(LEDSEL));
endmodule
```