CMPE 125 Lab 7 Assignments

Dr. Donald Hung
Computer Engineering Department, San Jose State University

System-level Design

Purpose:

1) To be familiar with system-level design methodologies
2) To be familiar with the procedure and tools for system-level development

Preparation:

Review lecture notes on finite state machine (FSM) and system-level design

Description:

The attached diagram (see Attachment 1) shows the control (FSM) and datapath (in dotted box) of a small calculator capable to perform ADD, SUBTRACT, AND, and XOR operations. The calculator will operate based on the system clock *CLK_db*. When idling, the calculator constantly watches for the *Go* signal and once the signal is detected, the calculator will do the following:

1) Load 3-bit input data (operands) *In1* and *In2* into the register file:

```
R1 ← In1;
R2 ← In2;
```

2) Executes the algorithm shown below, depending on the 2-bit operation control input *op*:

```
if (Op = 2’b11)          //Add
     R3 ← R1 + R2;
else if(Op = 2’b10)      //Subtract
     R3 ← R1 - R2;
else if(Op = 2’b01)      //And
     R3 ← R1 & R2;
else                     //Xor
     R3 ← R1 ^ R2;
```

3) Output the 3-bit result *Out* along with the flag signal *Done*, then return to the idle mode.

Tasks:

1. Design and functionally verify the datapath (DP).

   Top-level Verilog source code for the DP is attached (see Attachment 2). It connects all the building blocks in the DP (via module instantiation and port mapping). You need to complete the design code for each building block in the DP, and then write a Verilog testbench to functionally verify the DP.

2. Design and functionally verify the control unit (CU).

   You should use the systematic design methodology lectured in class to design the CU (the FSM), i.e., starting with constructing an ASM chart that describes the cycle-by-cycle operation of the calculator, then extract the *state transition diagram* and *output table* for coding the FSM's *next-state logic* and *output logic* respectively. The FSM must be tested thoroughly via functional verification

3. Integrate and functionally verify the entire system.

   You should write a top-level Verilog design code for the calculator by connecting the FSM with the DP, and write a Verilog testbench to verify the overall system's functional correctness.

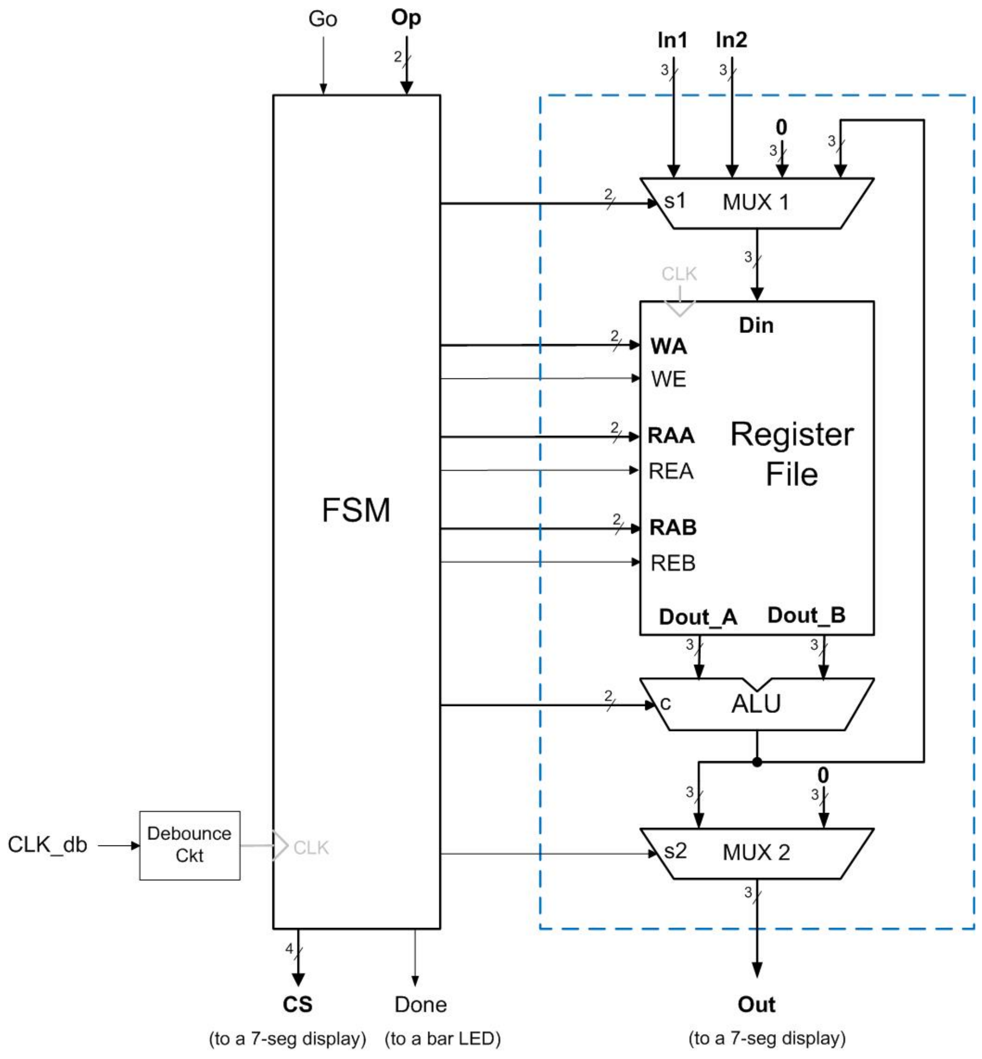
4. Validate the designed calculator using the Nexys2 FPGA board. Set the FGPA pin constraints based on the I/O arrangement specified below:
   - use an on-board push buttons to generate the start signal *Go*,
   - use another on-board push button for a manually generated system clock *CLK_db* (must be de-bounced, as shown in Attachment 1),
   - use six on-board DIP switches to enter the two 3-bit binary operands *In1* and *In2*,
   - use the two remaining on-board DIP switches to enter the 2-bit operation control input *op*,
   - use one of the on-board barcode LEDs to display the output signal *Done*,
   - use one on-board 7-segment LED to display the FSM's current state *CS* (this is for diagnosis purpose),
   - use another on-board 7-segment LED to display the calculated result *Out*. (**Note:** since the result *Out* is in binary, you need to write Verilog code for a *binary to 7-segment decoding* module **led4binary**, attach it to the DP's output, and integrate it to the overall system before launching the FPGA implementation tool.

Contents of Report:

1) Cover page.
2) A list of successfully accomplished tasks.
3) For Tasks 1 – 3, you should include the following:
   a) A test plan (for functional verification)
   b) Commented source code (design and verification)
   c) Captured verification results (waveforms/simulation log files)

   In addition, for Task 2 (the control unit), you should also include the following:
   d) The ASM chart describing the system's cycle-by-cyle operation
   e) The *state transition diagram* and the *output table* extracted from the ASM chart
4) For Task 4 (validation), you should describe the hardware prototyping setup and your test plan (for hardware validation)
5) Descriptions and analysis on your observations (for all tasks)
6) Detailed descriptions of the task(s) that you cannot, or did not accomplish, including reason(s) and/or your analysis.

Attachment 1: System Schematics



Attachment 2 : Top-level Verilog Source Code for the Datapath (DP)

```
module DP(in1, in2, s1, clk, wa, we, raa, rea, rab, reb, c, s2,
out);

     parameter word_width = 3;
     parameter rf_size = 4;
     parameter addr_size = 2;

input [word_width-1:0] in1, in2;
input [addr_size-1:0] s1, wa, raa, rab, c;
input we, rea, reb, s2, clk;
output [word_width-1:0] out;
```

```
wire [word_width-1:0] m1out;
wire [word_width-1:0] douta;
wire [word_width-1:0] doutb;
wire [word_width-1:0] aluout;

MUX1 U0(.in1(in1), .in2(in2), .in3(3'b000), .in4(aluout),
.s1(s1), .m1out(m1out));
RF U1(.clk(clk), .rea(rea), .reb(reb),  .raa(raa), .rab(rab),
.we(we), .wa(wa), .din(m1out), .douta(douta), .doutb(doutb));
ALU U2(.in1(douta), .in2(doutb), .c(c), .aluout(aluout));
MUX2 U3(.in1(aluout), .in2(3'b000), .s2(s2), .m2out(out));

endmodule
```