Design HW III

Prof. Mani Srivastava

EEM16/CSM51A - Logic Design of Digital Systems

VERSION 1.3

NOTE: You can chose to do this homework in a team of two, but in that case you need to do extra work. Also, there is an optional portion for extra credit, although the per-person credit is slightly less for teams of two.

SUBMISSION PROCEDURE: You need to submit your solution in electronic form at http://nesl.ee.ucla.edu/courses/eem16/2013w/submissions. You would need to use the user id and password handed out in the first lecture. If in a team of two, just one person should submit but make sure that the PDF file has names of both team members.

NO LATE SUBMISSION. Only exceptions would be for medical emergencies based on a certified note from a doctor.

Problem	Maximum Score	Your Score	Comments
BASELINE			
Correctness + Quality	50		
PDF and Video Report	20		
ADDITIONAL WORK FOR TEAMS OF 2			
Correctness + Quality	50		
PDF and Video Report	20		
Subtotal	**---**		
# of People (1 or 2)			
Subtotal divided by # of people			
Score out of 50+20	**70**		
EXTRA CREDIT FOR INDIVIDUALS			
/	8		
%	6		
EXTRA CREDIT FOR TEAMS OF TWO			
/	6		
%	6		

In this final design homework you will use the datapath created in the previous design homework, and combine with a control FSM and a user interface to create a simple programmable 16-bit calculator. You can choose to do this homework in a group of 2, but in that case you need to implement the additional functionality specified for teams of 2. Also, there is an optional component in this homework for extra credit.

The top level circuit is CALC, which uses subcircuit CALC_CORE and connects it to a clock source, a reset circuit, signals from a bunch of buttons that emulate keys of a calculator, and signals output to a bunch of seven-segment displays and LEDs. For your convenience we've provided a .circ file which already has a complete CALC and an empty CALC_CORE with all the necessary I/O pins. The file is available at :

https://www.dropbox.com/s/nt1rs10oe8h5c2m/libm16dhw3.circ

Additionally, in the file we have provided a circuit called Binary16to8BCD5, which converts 16-bit 2's complement numbers into 5-digit signed BCD numbers. You will need this functionality, and so this will save you some effort. In the file you'll also see a couple of other subcircuits that are used as subcircuits by other circuits in the file. ~~You should download this file and then load it as a library (see Project -> Load Library menu) so that you can easily update whenever we update this file to fix any bugs. Do not copy-and-paste as that will make life hard for you!~~

Your task is to design CALC_CORE, which is the heart of the calculator. You should use the DATAPATH from the previous design homework as the main part of CALC_CORE. Additionally, you can use any other gate or circuit from Logisim's library **except multiplier and divider**.

When using Logisim for this homework, you should simulate it with the clock set to tick continuously at a high frequency (Logisim permits as hight as 4.1 KHz).

CALC_CORE will take as inputs following signals which you must locate at the left side of your design in the following top-down order (this is how they already are in the file):

- CLK: clock
- RST: reset signal for you to initialize your circuit
- DIG[9:0]: i-th bit is asserted '1' for the duration the user presses the button with digit i.
- ADD, SUB, MUL, DIV, REM, NEG, INV, AND, OR, XOR, LSHFTL, LSHFTR, ASHFTR: These signals are '1' for the duration the user presses the +, -, *, /, %, [-], ~, &, |, ^, <<, >>, or >>> button respectively. They are intended for the user to do various arithmetic and logic operations.
- RUN, ENT: These signals are '1' for the duration the user presses the RUN or ENT buttons respectively. They are intended for user to run a custom program, and to enter an input value.
- MS[2:0]: These signals are '1' for the duration the user presses the MS3, MS2, and MS1 buttons respectively. They are intended for user to store the number that is displayed into one of three locations.

- MR[2:0]: These signals are '1' for the duration the user presses the MR3, MR2, and MR1 buttons respectively. They are intended for user to recall the number from one of the three locations and display it.
- DEC, HEX, TWC: These signals are '1' for the duration the user presses the DEC, HEX, or TWC buttons respectively, and intended to change the mode in which the 16-bit number is displayed. DEC means sign-magnitude decimal, HEX means sign-magnitude hexadecimal, and TWC means as 2's complement hexadecimal. So for example, -23 will be displayed as -23 in DEC mode, -17 in HEX mode, and FFE9 in TWC mode.

CALC_CORE will output the following signals, which you must locate at the right side of your design in the following top-down order (this is how they already are in the file):

- SGN: this signal is '1' to indicate that a minus sign should be displayed in the leftmost seven segment display.
- LED: this signal is '1' to indicate that the LED to the right of the display should be lit up.
- DECDISP, HEXDISP, TWCDISP: these signals are '1' depending on the mode in which the number is being displayed, and lit up the corresponding LED on the left side of the display.
- SSD4[6:0], SSD3[6:0], SSD2[6:0], SSD1[6:0], and SSD0[6:0]: these are five signals of 7-bit each to control the five seven-segment displays in which digits will be displayed.

You should look carefully at how the buttons and display elements are wired up to CALC_CORE in the CALC circuit.

Your calculator should function as described below. You should use this description to create a control FSM that will provide INSTR and DIN inputs to the DATAPATH, and look at its DOUT, FNEG, FZER, and FFOVF outputs.

In case of something that is not fully specified, you are expected to fill in the details yourself as you would do in a real design task.

1. Upon power up, the display should be all blank except the last digit which should be 0. The calculator display should be in DEC mode. All 14 registers in the REG module inside DATAPATH should be initialized to 0.
2. By default the display shows the results of the last operation, and the calculator waits for inputs from user.
3. CLR should clear the display (0 in the least significant digit, blank elsewhere, minus sign cleared)
4. User can enter a new input by pressing the digit keys. When the first digit key is pressed after an operation, the display should in effect clear, the calculator should go into DEC mode, and uses the first digit as the start of entering a new number. Each time the user presses a digit, it should go to the least-significant position on the display while at the same time existing digits on the display should shift one position.
   i. Exceptional conditions to handle when entering numbers:

a. If the entered digit would would make the number too large or too small for 16 bits signed representation, the display should read “Error”. After this the display should become unresponsive to any key other than CLR, DEC, HEX, and TWC (see below).
b. If the user presses DEL, then you should shift the digits on the display right by 1, and make the most significant digit display blank.
c. If the user presses CLR, the display should be cleared as described in step 2.
d. If the user presses the [-1] key, the sign of the number on the display should toggle. If doing so might result in too large or small a number, then do what is described in Step 4a.

ii. If the user presses DEC, HEX, or TWC button, the number displayed should from then on be represented according to the new mode and the corresponding LED lit up. The calculator display should stay in this mode until the next time DEC, HEX, or TWC button is pressed, or if the user starts entering new number (in which case the mode becomes DEC). If when these buttons are pressed the display is not showing a number (E.g. “Error”) then reset the display to 0 in addition to changing to the new display mode.

5. If the user presses one of the store keys (MS1, MS2, or MS3), you should arrange for the number on the display to be stored at address 1, 2, or 3 respectively in the REG module inside DATAPATH. This will require you to suitably control the INSTR input to DATAPATH.

6. If the user presses one of the recall keys (MR1, MR2, or MR3), you should arrange for the number stored at address 1, 2, or 3 respectively in the REG module inside DATAPATH to be displayed. This will require you to suitably control the INSTR input to DATAPATH.

7. If the user presses any of the other operation keys (+, -, *, /, %, &, |, ~ , ^, <<, >>, >>>) then the calculator should perform the corresponding operation and display the results. If the result is non-representable, the display should say “Error”. Note that most of these operations correspond to the one that you had implemented in the ALU. However, some such as / and % will require a sequence of operations on the DATAPATH.

   i. For operations requiring single operand, the operand value is the one which was on the display.
   ii. For operands requiring two operands, the first operand value is the one which was on the display and the second operand value is the one that is at address 1 in the REG module inside DATAPATH.
   iii. Implementing / and % is harder, and so is optional. You should do them only after the rest of your calculator is functional. Realizing there operations would get you extra credit as noted on the cover sheet. If you decide to not implement these two operations, then pressing these keys should have no effect. For teams of 2, wait till you read Step 10.

8. You might find it useful to use signals generated by buttons as clock signals to D flip-flops in order to detect their press. This can help you debug without needing to have a continuously running clock as would be the case with the alternative where you detect button press on edges of clocks (this is because in Logisim you cannot simultaneously press a button

and also poke the CLK to change its value). You may also find useful the technique in http://guanidene.blogspot.com/2012/01/simple-calculator-display-logic-circuit.html.

9. **[For teams of 2 only]:** If the user presses the RUN button then your calculator needs to run a microprogram stored in a ROM. Inside CALC_CORE you should instantiate a ROM with 12-bit address and 16-bit data output. Logisim stores ROM's contents in a file, and so you can even write you microprogram in a text editor! When RUN is pressed, you should execute the microinstructions starting from the address corresponding to the lower 12-bits of the number entered by the user on the display. To do this, you should create a microprogam counter, set its value to the start address, and then run microinstruction that you fetch and update the microprogam counter as necessary as per the following description:
   i. The microinstructions are 16-bit.
   ii. If first four bits [15:12] are 0000 through 1010 then the microinstruction does the operation on DATAPATH as was described in Design HW II.
   iii. If first two bits [15:14] are 11 then the microrinstruction is a branch instruction. The following two bits [13:12] indicate the condition depending on which branch is taken: 00 means branch is always taken, whereas 01, 10, and 11 correspond to branch being taken is FNEG, FZER, and FOVF from DATAPATH are 1 (as a result of the preceding microinstruction). Lastly, bits [11:0] give the 12-bit address from which the next microinstruction is read in case the branch is indeed taken.
   iv. If bits [15:12] of the microinstruction are 1011 then, depending on the next four bits [11:8] different actions are taken.

Bits [11:8]	opcode	
0000	HLT	Microprogram is halted, and the calculator should display the number corresponding to bits [7:0].
0001	OUT	If bits [7:4] = 0x0, the right LED is turned off and the display is left unchanged. If bits [7:4] = 0x1, the right LED is turned on and the display is left unchanged. If bits [7:4] = 0x2, the right LED is turned off and the display is updated according to the contents of [3:0] as described below. If bits [7:4] = 0x3, the right LED is turned on and the display is updated according to the contents of [3:0] as described below. If bits [7:4] = 0x4, the display is updated according to the contents of [3:0] as described below but LED is left unchanged. If [3:0] = 0x0 then 0 is sent to display, if [3:0] = 0x1 then 1 is displayed, and otherwise, using bits [3:0] as an address, the value stored at that address in register bank REG inside the DATAPATH is displayed. The microprogram should move to the next microinstruction.
0010	IN	Display the number corresponding to bits [7:0] preceded by the letter "E" (e.g. E255 if bits [7:0] were FF and the display mode was DEC), and then wait for user to enter a new number into the display and then press the ENT button. The microprogram should move to the next microinstruction.

v. Whenever microinstruction uses DIN of the DATAPATH, then the value it should get is the one on the display of the calculator.
vi. While the microprogram is running, the calculator display should not change unless the microprogram causes it to change due to IN, OUT, or HLT instructions.
vii. In case of invalid microinstructions, the microprogram should do nothing for that microinstruction (i.e. no operation), and move to the next microinstruction.

10. **[For teams of 2 only]:** Note that using the preceding one can write microprograms that can take interactive inputs. Write a test program that asks user for a sequence of numbers and finds their average. Assume that the numbers are all positive, and so the user can indicate the send of sequence by entering a negative number.
11. **[For teams of 2 only]:** Now that you have microprogram capability, you can easily implement the / and % as microprograms and get the extra credit for Step 8. Basically, when the user presses / or %, you can run a microprogram that you can put in a private area of the microcode ROM (which you can now make to be with 13 bit address).

What to submit?

Please follow these instructions carefully.

1. Save your design with the name *dhw3*. This will create a file called *dhw3.circ.*
2. Additionally create a single PDF file named *dhw3.pdf* with the following
   a. Your name and student id on the top of the first page
   b. Color screen capture of Logisim showing schematics of each of the circuits.
   c. For teams of two, how the work was divided.
3. Use the free screen video capture software at http://www.techsmith.com/jing.html to create a a short video (< 5 min, which is the limit of the free softwate) of you showing Logisim working with the circuit on some sample inputs. Include a URL to the video in the PDF. If you need to show a longer video, you can create multiple videos. Please do audio narration as necessary.

Put both *dhw3.circ* and *dhw3.pdf* (and any other additional file you deem necessary) into a single folder with the name *dhw3_yourlastname_yourfirstname* (e.g. *dhw3_smith_john*), and then zip that folder to get a file named *dhw3_yourlastname_yourfirstname.zip.* Upload this file by logging in at http://nesl.ee.ucla.edu/courses/eem16/2013w/submissions using the password and id that were given in the first lecture.