ECSE 318 **Due: Oct 13**

Problem 1 (50 points): In this assignment, you are asked to design the ALU of a 16-bit RISC microprocessor in Verilog. In designing the ALU, you need to follow the specification given below. Develop a hierarchical design by first identifying the modules which will implement the specification.

Top cell : alu `

I/O: bit 15 is the most significant bit. INPUTS: A(15:0) , B(15:0) , alu_code(4:0) ; OUTPUTS: C(15:0) overflow

The ALU control signals (alu_code) consist of five bits and define a total of 21 instructions.

- There are two fields in these five signals: operation type field, and operation field.
- The two high order bits denote the operation type and the lower three bits represent the operations which will take place.
- The bit patterns for the operation type field are:

arithmetic operation (00---)			**logic operations (01---)**		
add : 000	signed addition	(A+B -> C)	and : 000	A AND B	(A AND B -> C)
addu : 001	unsigned addition	(A+B -> C)	or : 001	A OR B	(A OR B -> C)
sub : 010	signed subtraction	(A-B -> C)	xor : 010	A XOR B	(A XOR B -> C)
subu : 011	unsigned subtraction	(A-B -> C)	not : 100	NOT A	(NOT A -> C)
inc : 100	signed increment	(A+1 -> C)			
dec : 101	signed decrement	(A-1 -> C)			

shift operations (10---)		**set condition operation (11---) (A and B are signed numbers)**	
sll : 000	logic left shift A by the amount of B	sle : 000	if A <= B then C(15:0) = <0...0001>
srl : 001	logic right shift A by the amount of B	slt : 001	if A < B then C(15:0) = <0...0001>
sla : 010	arithmetic left shift A by the amount of B	sge : 010	if A >= B then C(15:0) = <0...0001>
sra : 011	arithmetic right shift A by the amount of B	sgt : 011	if A > B then C(15:0) = <0...0001>
		seq : 100	if A = B then C(15:0) = <0...0001>
		sne : 101	if A != B then C(15:0) = <0...0001>
			otherwise, set C(15:0) = <0...0000>

- If there is an overflow in the ALU operations, the signal ``overflow" should be set to high, otherwise it is low. Therefore, when doing all the operations other than arithmetic ones, ``overflow" should always be set to low.
- Because this is a 16-bit ALU, the amount of shift will never go over 15. Only B(3:0) are used to define the shift amount.
- Find the critical path of your ALU. Try to optimize the circuit on the critical path so the delay of your ALU can be minimized.

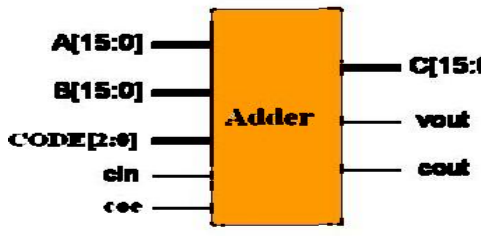
TOP MODULE: 16-bit Adder Module `

PRIMARY SIGNALS: BUS inputs/outputs and additional signals `

BUS signals: bit 15 is the most significant bit. All `
bus signals are named with upper case letters. `

INPUTS: A[15:0], B[15:0],CODE[2:0] OUTPUTS: C[15:0]

Additional signals: named with lower case letters	**3-bit CODE[2:0] control signals**	

vout = signed overflow
cout = carry output
cin = carry input
coe = carry output enable (active low)
	add : 000 signed addition (A+B -> C)
	addu : 001 unsigned addition (A+B -> C)
	sub : 010 signed subtraction (A-B -> C)
	subu : 011 unsigned subtraction (A-B -> C)
	inc : 100 signed increment (A+1 -> C)
	dec : 101 signed decrement (A-1 -> C)
		 A[15:0]
		B[15:0]
		CODE[2:0]
		cin
		coe
		Adder
		C[15:0]
		vout
		cout

```
Here are the input/output:
 A = A(15:0);          B  = B(15:0)
 i = cin               oe = coe
 f = function          C = C(15:0)
 v = vout              cout = cout
```

A	B	i	oe	f	C	v	cout
0000	0001	0	0	add	0001	0	0
000F	000F	1	0	add	001F	0	0
7F00	0300	0	0	add	8200	1	0
FF00	0100	1	0	add	0001	0	1
8100	8000	1	1	add	0101	1	d
0000	0001	1	0	sub	FFFF	0	0
000F	000F	1	0	sub	0000	0	1
7F00	0300	1	0	sub	7C00	0	1
FF00	0100	1	0	sub	FE00	0	1
8100	8000	1	1	sub	0100	0	d
0000	0100	0	0	inc	0001	0	0
0F00	0F00	1	0	inc	0F01	0	0
7FFF	0300	0	0	inc	8000	1	0
FF00	0100	1	0	inc	FF01	0	0
8100	8000	1	1	inc	8101	0	d

A	B	i	oe	f	C	v	cout
0000	0001	0	0	addu	0001	0	0
000F	000F	1	0	addu	001F	0	0
7F00	0300	0	0	addu	8200	0	0
FF00	0100	1	0	addu	0001	0	1
8100	8000	1	1	addu	0101	0	d
0000	0001	1	0	subu	FFFF	0	0
FF00	FCE0	1	0	subu	0220	0	1
7F00	0300	1	0	subu	7C00	0	1
FF00	0100	1	0	subu	FE00	0	1
8100	8000	1	1	subu	0100	0	d
0000	0100	0	0	dec	FFFF	0	0
000F	000F	1	0	dec	000E	0	1
7F00	0300	0	0	dec	7EFF	0	1
FF00	0100	1	0	dec	FEFF	0	1
8000	8000	1	1	dec	7FFF	1	d

```
 d = don't care
```

Signed overflow: When two operands of the same sign are added together and the result is of the opposite sign. Be sure that the signed overflow bit vout is only affected by signed operations.

```
 Here is an example of signed overflow. You must determine all the
 cases when signed overflow will occur.
 Ex:            01110000      8 bit signed int
               +01000001      8 bit signed int
                ----------
                10110001      vout=1
```

Problem 2(50 points):

Free Cell Game can be described as follow:

Let $R = \{H, D\}$ be the set of *redsuits* and $B = \{S,C\}$ be the set of *blacksuits*. Let $\Sigma = \{R \cup B\}$ be the set of all suits.

Let $\delta = \{1,2,3,4,5,6,7,8,9,10,J,Q,K\}$ be the ordered set of values $(1 \le 2 \le 3 \le 4 \le 5 \le 6 \le 7 \le 8 \le 9 \le 1 \le, J \le Q \le K)$.

The *card deck* is the Cartesian product $\Delta = \sum \times \Gamma$. Each element $\delta = (\sigma, \gamma)$ is a card. Two cards $\delta_1 = (\sigma_1, \gamma_1)$ and $\delta_2 = (\sigma_2, \gamma_2)$ are of the *same suit* if and only if $\sigma_1 = \sigma_2$. They are of the *same color* if $\gamma_1 \in R$ if and only if $\gamma_2 \in R$.

Freecell is a game played with a card deck according to the following rules:

- The cards are initially partitioned into eight ordered sets c_0 through c_7 (the *columns* of the *tableau*). Sets c_0, c_1, c_2, and c_3 initially contain seven cards each, while the remaining sets contain six cards each.
- There are four sets f_a, f_b, f_c, and f_d (the *freecells*), which are initially empty.
- There are four ordered sets h_H, h_D, h_S, and h_C (the *homecells*, one for each suit), which are initially empty.
- The game consists of a sequence of turns. At each turn, the player moves one card from one set to another according to the rules set forth below. The player wins the game when all cards have been moved to the h_γ sets.
- A card $\delta = (\sigma,\gamma)$ can be moved from its current position only if one of the following conditions holds:
    1. The current position is a free cell;
    2. The current position is the head of one of the c_i ordered sets.

- Card $\delta = (\sigma,\gamma)$ can be moved to a new position only if one of the following conditions holds:
    1. The new position is an empty free cell;
    2. The new position is the home cell h_σ and either h_σ is empty and $\gamma = \Lambda$ or the last element of h_σ is $\delta' = (\sigma,\gamma')$ and γ' immediately precedes γ in Γ.
    3. The new position is at the beginning of c_i and either c_i is empty, or the current first element of c_i, $\delta' = (\sigma,\gamma')$ satisfies the following two conditions:
        1. the color of δ' is different from the color of δ;
        2. δ' immediately follows δ in Γ.

Simple consequences of the constraints on legal moves are:

- The free cells can contain at most one card.

At any point in time during the game, h_γ is either empty or contains a set of consecutive cards of suit γ starting with $(\gamma,1)$.

Problem 3(50 points):

Free Cell Game can be described as follow:

Let $R = \{H, D\}$ be the set of *redsuits* and $B = \{S,C\}$ be the set of *blacksuits*. Let $\Sigma = \{R \cup B\}$ be the set of all suits.

Let $\delta = \{1,2,3,4,5,6,7,8,9,10,J,Q,K\}$ be the ordered set of values ($1 \leq 2 \leq 3 \leq 4 \leq 5 \leq 6 \leq 7 \leq 8 \leq 9 \leq 1 \leq, J \leq Q \leq K$).

The *card deck* is the Cartesian product $\Delta = \sum \times \Gamma$. Each element $\delta = (\sigma, \gamma)$ is a card. Two cards $\delta_1 = (\sigma_1, \gamma_1)$ and $\delta_2 = (\sigma_2,\gamma_2)$ are of the *same suit* if and only if $\sigma_1 = \sigma_2$. They are of the *same color* if $\gamma_1 \in R$ if and only if $\gamma_2 \in R$.

Freecell is a game played with a card deck according to the following rules:

- The cards are initially partitioned into eight ordered sets c_0 through c_7 (the *columns* of the *tableau*). Sets c_0, c_1, c_2, and c_3 initially contain seven cards each, while the remaining sets contain six cards each.
- There are four sets f_a, f_b, f_c, and f_d (the *freecells*), which are initially empty.
- There are four ordered sets h_H, h_D, h_S, and h_C (the *homecells*, one for each suit), which are initially empty.
- The game consists of a sequence of turns. At each turn, the player moves one card from one set to another according to the rules set forth below. The player wins the game when all cards have been moved to the h_γ sets.
- A card $\delta = (\sigma,\gamma)$ can be moved from its current position only if one of the following conditions holds:
    1. The current position is a free cell;
    2. The current position is the head of one of the c_i ordered sets.
- Card $\delta = (\sigma,\gamma)$ can be moved to a new position only if one of the following conditions holds:
    1. The new position is an empty free cell;
    2. The new position is the home cell h_σ and either h_σ is empty and $\gamma = \Lambda$ or the last element of h_σ is $\delta' = (\sigma,\gamma')$ and γ' immediately precedes γ in Γ.
    3. The new position is at the beginning of c_i and either c_i is empty, or the current first element of c_i, $\delta' =$

(σ,γ') satisfies the following two conditions:

1. the color of δ' is different from the color of δ;
2. δ' immediately follows δ in Γ.

Simple consequences of the constraints on legal moves are:

- The free cells can contain at most one card.
- At any point in time during the game, h_γ

Every initial partition of the cards into four groups of seven and four groups of six corresponds to one deal of the freecell game. Not all games terminates, and not all games that terminate are won.

Problem Description:
It is convenient to introduce a shorthand notation for moves. We shall use 1 through 8 to designate the columns of the tableau; *a* through *d* to designate the free cells; and *h* to designate the home cells. (Knowing what card is being moved allows one to uniquely identify which home cell is the destination.) With this notation, each move can be represented by a pair of characters. For instance, 4*b* designates the move of the first card of Column c_3 to the second free cell. (Notice the difference by 1. It is advantageous to stick to the standard notation, but it also advantageous to start the column indices from 0. The testbench should take care of this detail.)
Cards can also be designated by a combination of characters. For instance, *HA* stands for the Ace of Hearts.
You are supposed to write a Verilog description of a machine with the following features:

- The machine has the following interface.
```
module frecellPlayer(clock,source,dest,win);
input      clock;   input [3:0] source, dest; output      win;
```

- At each clock cycle, the machine processes one move, which is specified by *source* and *dest*. The encoding of source and destination is the following.
  1. Encoding for sources:
```
source[3]   = 0   =>  source[2:0] = column of tableau
source[3:2] = 10  =>  source[1:0] = free cell
source[3:2] = 11  =>  illegal
```
  2. Encoding for destinations:
```
dest[3]     = 0   =>  dest[2:0]   = column of tableau
dest[3:2]   = 10  =>  dest[1:0]   = free cell
dest[3:2]   = 11  =>  home cell   (dest[1:0] are ignored)
```
  An illegal source makes the move illegal, of course. When the destination is a home cell, the correct home cell is determined by examining the source.
- The output *win* should go to one when all cards reach the home cells. Since the freecell player has no reset input, and since there are no legal moves from the state in which all cards are in the home cells, *win* stays at 1 indefinitely once it transitions from 0.
- The initial state of the machine should correspond to the deal of freecell given below, which will be also posted on the Web site of the course. This is known as game 8321. In this tableau, the columns run from top to bottom. That is, the card that can be moved from Column c_0 is *HA*.
```
S4  S5  SJ  H4  DQ  D5  H5  CJ
DJ  S10 C7  SA  HJ  DK  D3  D4
D10 H8  C9  CQ  SQ  C3  HQ  H10
D6  C4  C6  C5  S6  D9  D7  C8
S3  H6  C2  S7  D2  H3  CK  H7
DA  HK  SK  H9  S9  S2  C10 D8
HA  H2  CA  S8
```
  To check the correctness of your machine you may want to experiment with different deals as well. The one above should correspond to the initial state of the description that you turn in.

You also have to write a testbench for your machine that conveniently exercises the design. Further details on the testbench, and a sample module for it will be posted on the course Web site. The sequence of moves will use the notation descried above.

Finally, write a short report (one or two pages) detailing the major design decisions. These should include the choice of the representation of the state of the game

The criteria according to which you should evaluate your design are: correctness, thoroughness of documentation, efficiency (e.g., how many latches), and simplicity.

(**This description is from Prof. Somenzi at Boulder**)