

ECE 316 – HW#3 - A

Simple ALU
with
Hardware Implementation

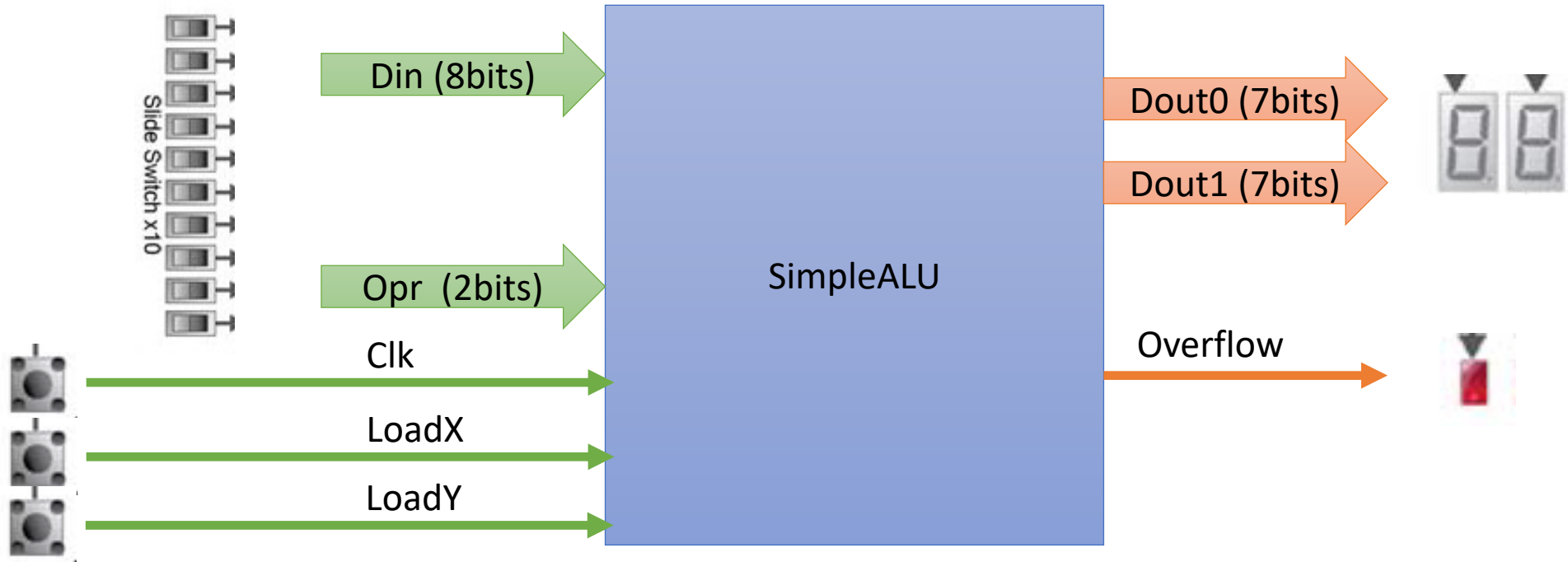
Objective

- 8-bit unsigned adder
 - X, Y
- Multiple operations
 - X
 - $X + 1$
 - $X - 1$
 - $X + Y$
- Binary input
- Decimal Output

SimpleALU – Inputs/Outputs

Pin/Bus	I/O	Size	Name
Din	Input	8-bits	Data load inputs
LoadX	Input	1-bit	Load X Register (active LOW)
LoadY	Input	1-bit	Load Y Register (active LOW)
Opr	Input	2-bit	Operation
Clk	Input	1-bit	Register Clock (Pulse, FallingEdge)
Dout0	Output	7-bits	7-segment Low Order digit
Dout1	Output	7-bits	7-segment High Order digit
Negative	Output	1-bit	7-Segment Bit 6 (0=ON, 1=OFF)
Overflow	Output	1-bit	Overflow flag

SimpleALU – External View



SimpleALU - Operations

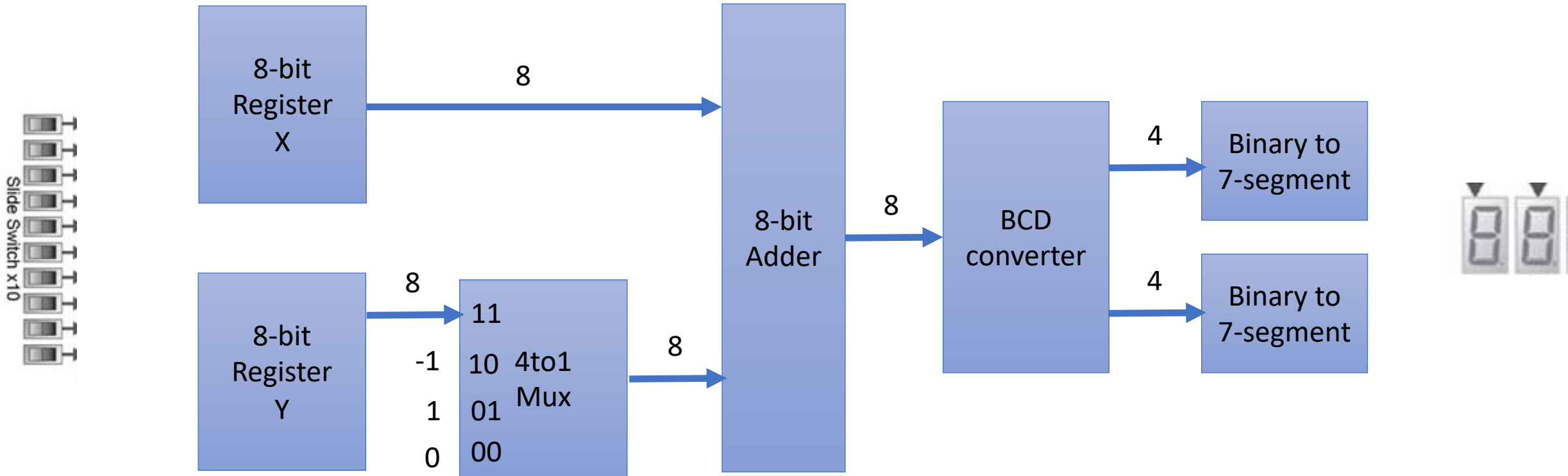
LoadX LoadY	Operation
11	Disable Load (default)
01	Load Register X
10	Load Register Y
00	Load X and Y

Opr	Operation/Output
00	$X (+ 0)$
01	$X + 1$
10	$X - 1$
11	$X + Y$

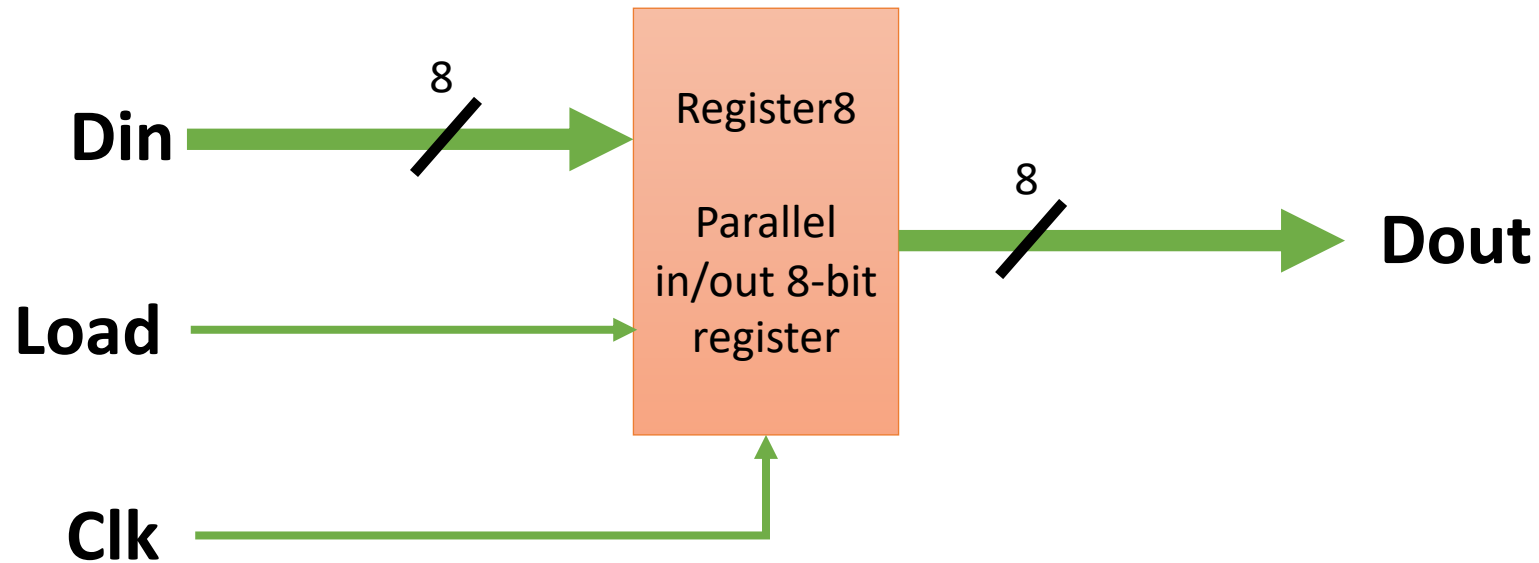
Constraints:

- All values are in range -99 to +99
- Overflow otherwise
- Numbers are 2's complement
- MSb is sign bit (0 – pos, 1 – neg)

Structural Design



SimpleALU – Internals – register8



Operation:

Load = 0:

Load Din to Internal on Clk falling edge
Load internal to Dout on Clk rising edge

Load = 1:

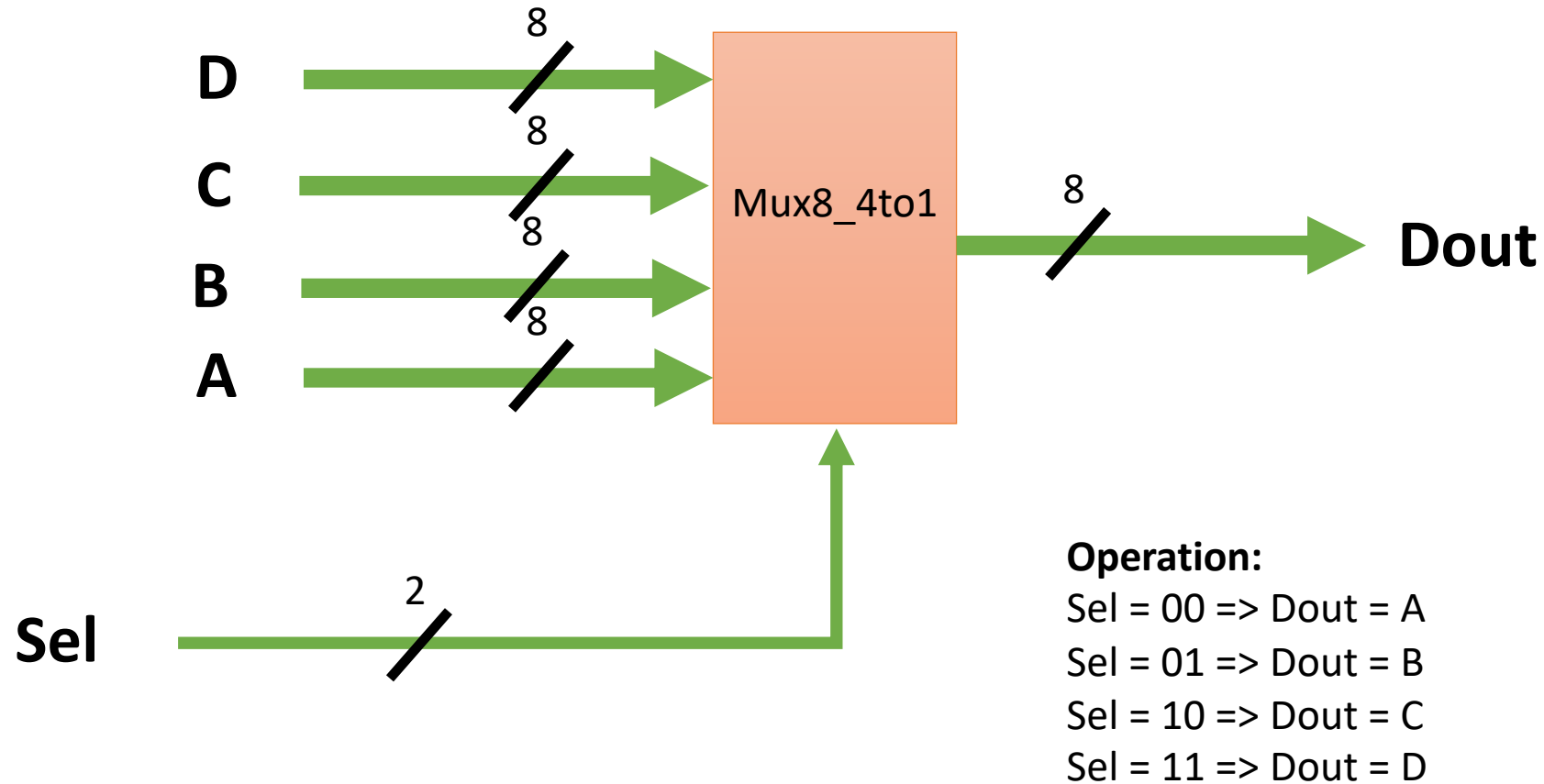
Hold Dout, ignore Din

Adjustments

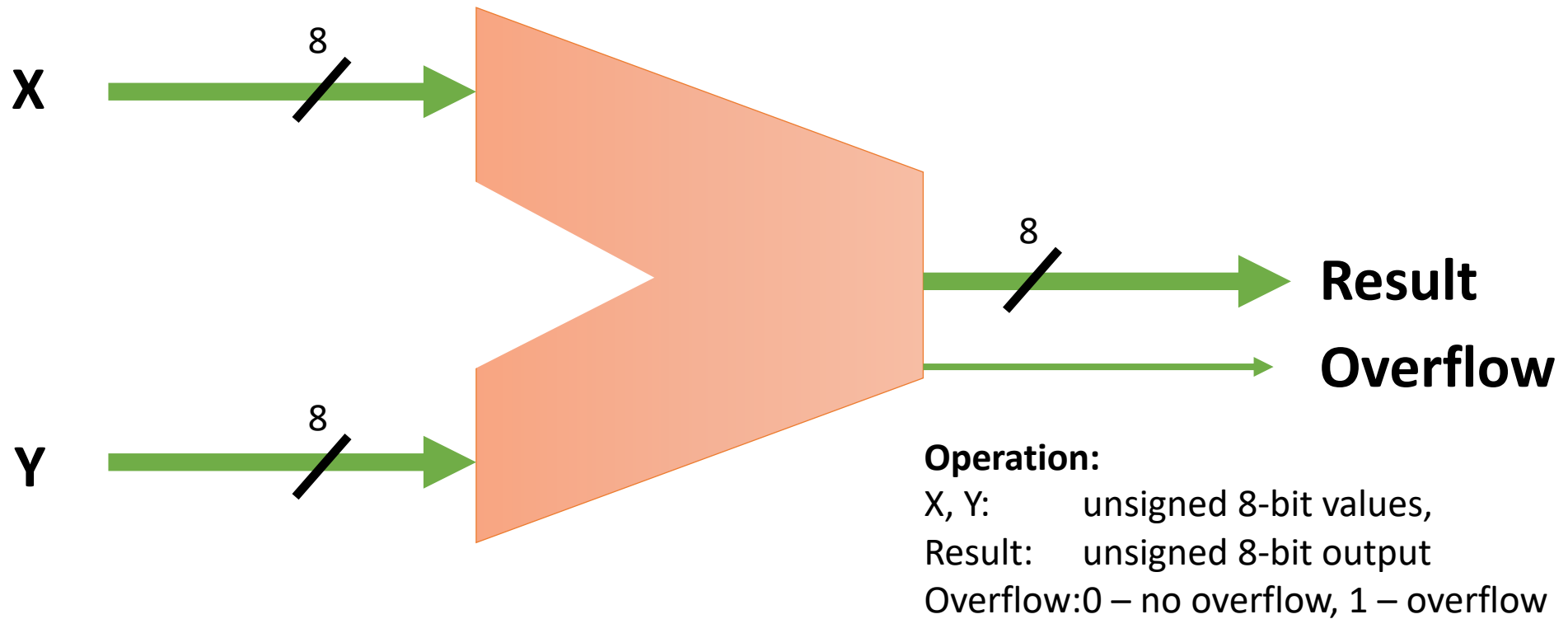
- Load is active LOW
- Clock is active LOW

Din	Load	<i>Store</i>	Clk	Dout	Note
X	1	Store	X	Dout	Hold last value
X	0	Din	↓	Dout	Din to internal storage
X	0	Store	↑	Store	Internal storage to Dout

SimpleALU – Internals – mux8_4to1



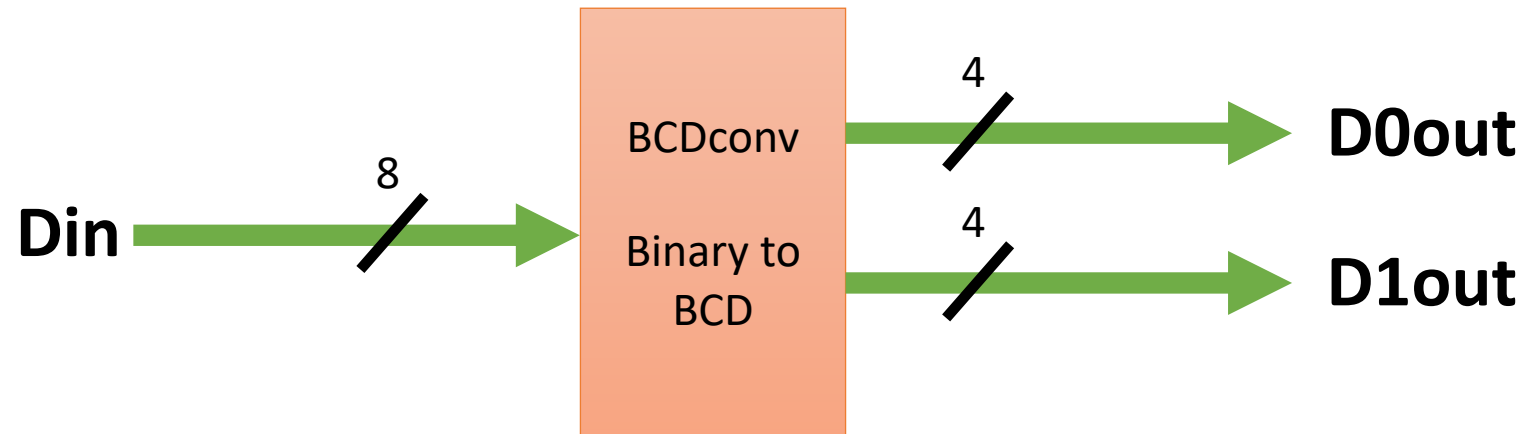
SimpleALU – Internals – add8



8-bit Adder

- Either Ripple-Carry or CLA 8-bit adder
- For RC:
 - Can increase 1-bit adder components to 8
 - OR
 - Can use 2 x 4-bit adders
- For CLA
 - Use 2 x 4-bit adders

SimpleALU – Internals – BCDconv



Operation:

Converts a unsigned 8-bit binary number in the
Range: 0 to +99
to two decimal numbers


BCD: Binary Coded Decimal (ECE315)

Decimal	Binary Number	BCD Outputs	OverFlow
1	0000 0001	0000 0001	0
.	.		.
.	.		.
.	.		.
9	0000 1001	0000 1001	0
10	0000 1010	0001 0000	0
11	0000 1011	0001 0001	0
12	0000 1100	0001 0010	0
.	.		.
.	.		.
.	.		.
99	0110 0011	1001 1001	0
100	0110 0100	xxxx xxxx	1

Shift Add 3 Algorithm (ECE315)

- Do 8 times (for 8 bit binary input).
 - 1. Shift left one to the binary input.
 - 2. Add 3 to BCD if it's greater than 4.
 - 3. Go to 1.

Shift Add 3 Algorithm (ECE315)

- Input = HGFE DCBA
 - Take the first 3 MSB of inputs and add to output.
 - Output = 0000 0HGF
 - If 00HGF > 4
 - $I\ H1\ G1\ F1 = 0\ H\ G\ F + 0011.$
 - Else
 - $I\ H1G1F1 = 0\ H\ G\ F$
 - Shift left one.
 - Output = 000I H1 G1 F1 E
 - If H1 G1 F1 E > 4
 - $H2\ G2\ F2\ E1 = H1\ G1\ F1\ E + 0011.$
 - Else
 - $H2\ G2\ F2\ E1 = H1\ G1\ F1\ E$
 - Shift left one.
 - Output = 00I H2 G2 F2 E1 D
 - If G2 F2 E1 D > 4
 - $G3\ F3\ E2\ D1 = G2\ F2\ E1\ D + 0011.$
 - Else
 - $G3\ F3\ E2\ D1 = G2\ F2\ E1\ D$
 - Shift left one.
 - Output = 0I H2 G3 F3 E2 D1 C
- 
- If 0I H2 G3 > 4
 - $J\ I1\ H3\ G4 = 0\ I\ H2\ G3 + 0011.$
 - Else
 - $J\ I1\ H3\ G4 = 0\ I\ H2\ G3$
 - If F3 E2 D1 C > 4
 - $F4\ E3\ D2\ C1 = F3\ E2\ D1\ C + 0011.$
 - Else
 - $F4\ E3\ D2\ C1 = F3\ E2\ D1\ C$
 - Shift left one.
 - Output = I1 H3 G4 F4 E3 D2 C1 B
 - If I1 H3 G4 F4 > 4
 - $I2\ H4\ G5\ F5 = I1\ H3\ G4\ F4 + 0011.$
 - Else
 - $I2\ H4\ G5\ F5 = I1\ H3\ G4\ F4$
 - If E3 D2 C1 B > 4
 - $E4\ D3\ C2\ B1 = E3\ D2\ C1\ B + 0011.$
 - Else
 - $E4\ D3\ C2\ B1 = E3\ D2\ C1\ B$
 - Shift left one.
 - Output = H4 G5 F5 E4 D2 C2 B1 A

Add 3 Circuit (ECE315)

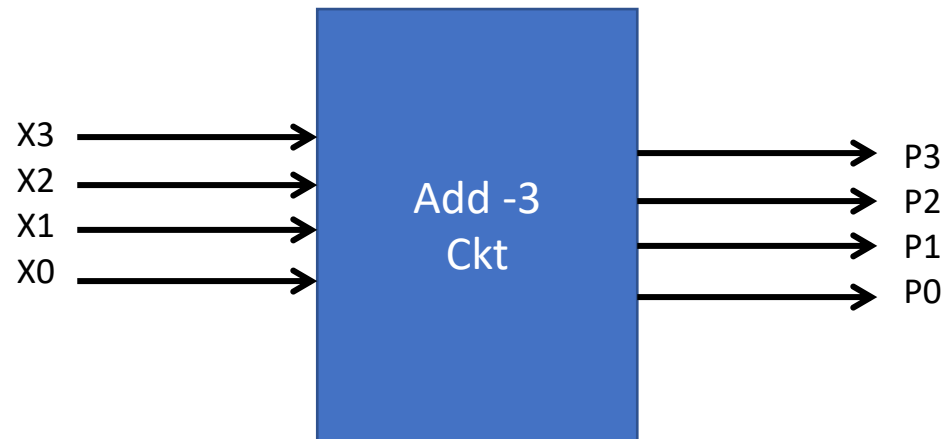
- If Input > 4
- Output = Input + 3;
- Else
- Output = Input;
- End

Input = 0101 (which is greater than 4)
Output = 0101 + 0011 = 1000

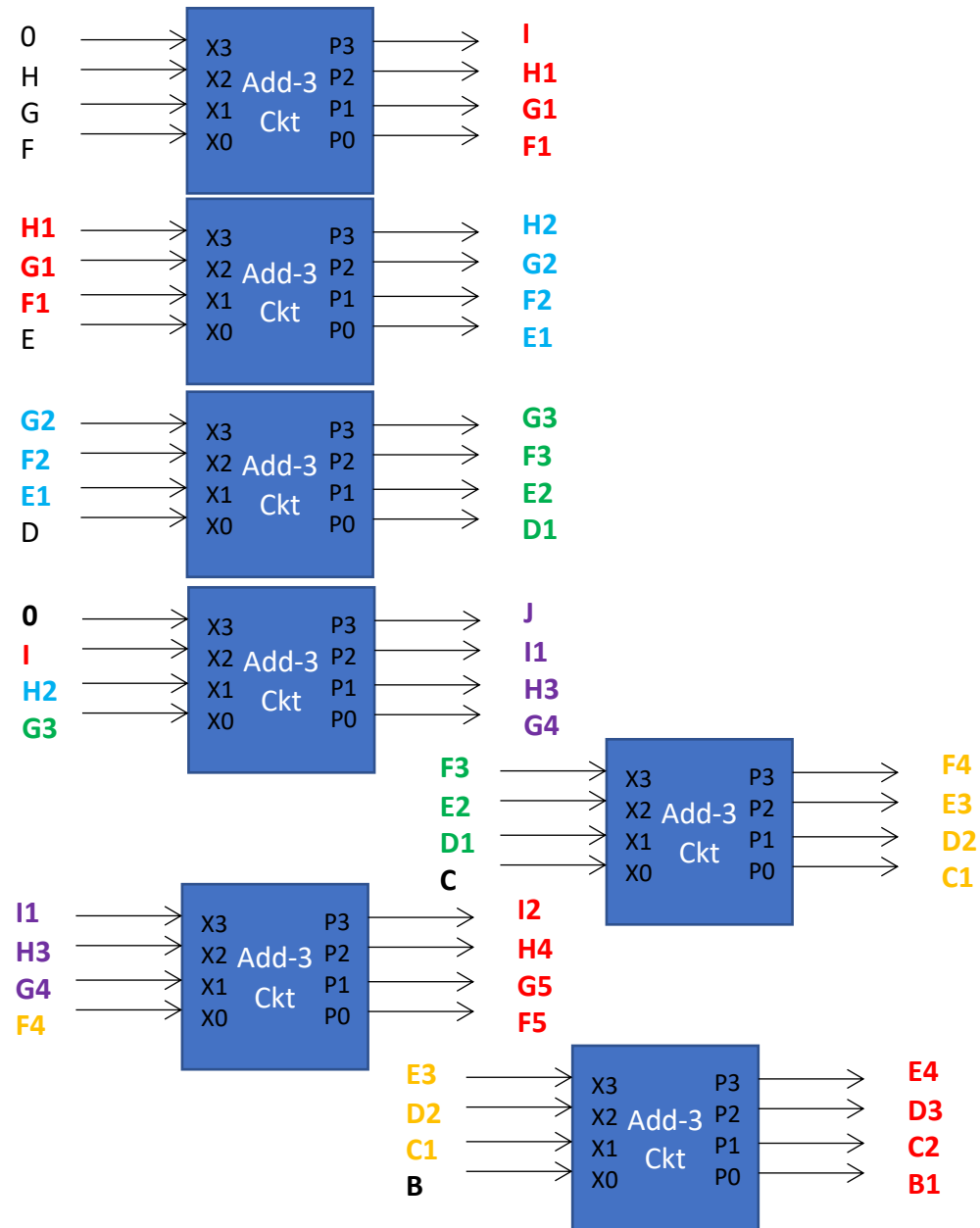
Input X3X2X1X0	Output P3P2P1P0	
0000	0000	When <i>input</i> ≤ 4 .
0001	0001	
0010	0010	
0011	0011	
0100	0100	
0101	1000	Fill these outputs
0110	1001	
0111		
1000		
1001		
1010	xxxx	When input > 9 , don't care.
.	.	
.	.	
1111	xxxx	

Add 3 Circuit (ECE315)

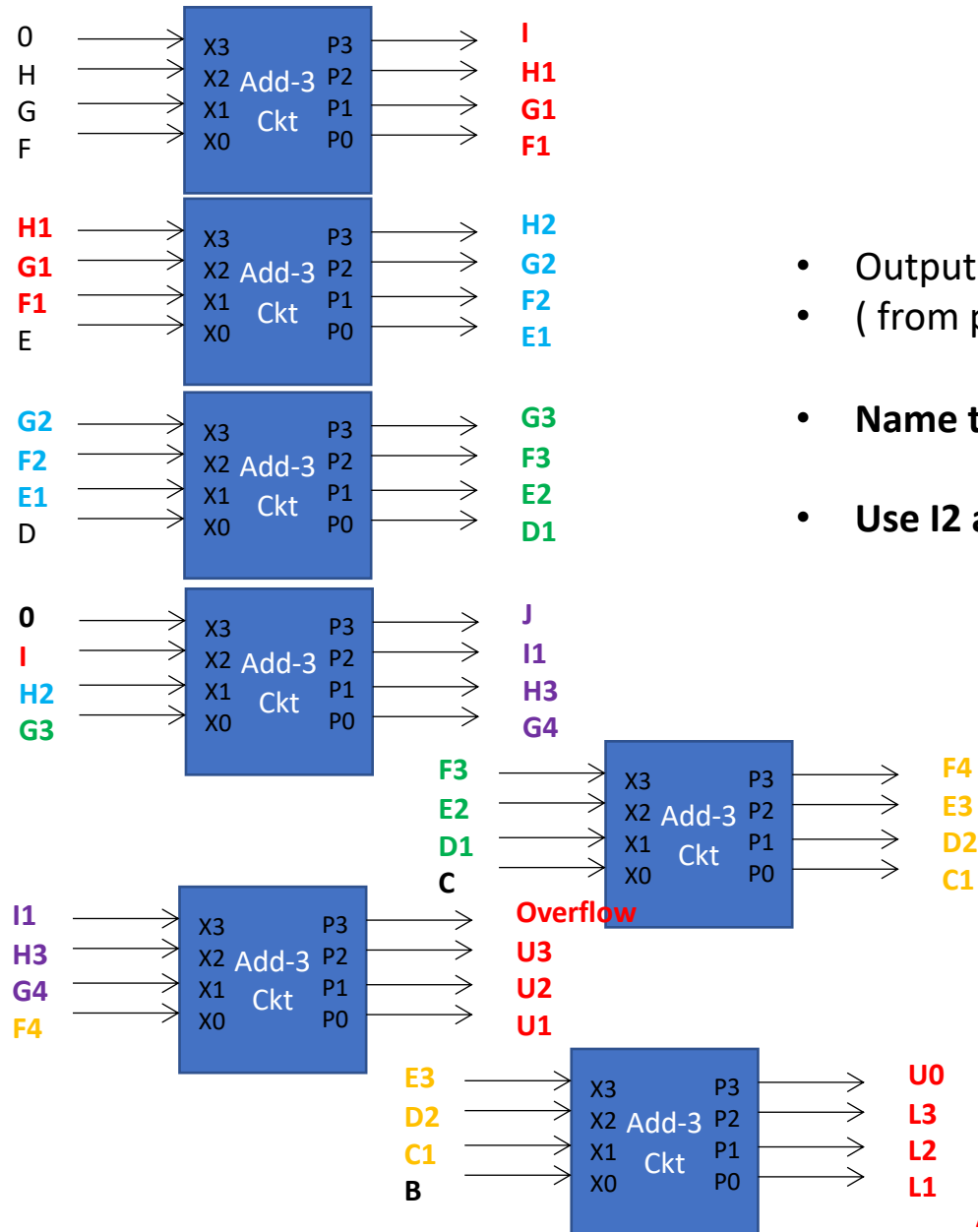
- Complete the Truth Table from prev slide
- Find the Boolean expression for P3, P2, P1 and P0.
- Implement in VHDK



- If $0HGF > 4$
 - $I\ H1\ G1\ F1 = 0\ H\ G\ F + 0011.$
- Else
 - $I\ H1G1F1 = 0\ H\ G\ F$
- Shift left one.
 - **Output = 000I H1 G1 F1 E**
- If $H1\ G1\ F1\ E > 4$
 - $H2\ G2\ F2\ E1 = H1\ G1\ F1\ E + 0011.$
- Else
 - $H2\ G2\ F2\ E1 = H1\ G1\ F1\ E$
- Shift left one.
 - **Output = 00I H2 G2 F2 E1 D**
- If $G2\ F2\ E1\ D > 4$
 - $G3\ F3\ E2\ D1 = G2\ F2\ E1\ D + 0011.$
- Else
 - $G3\ F3\ E2\ D1 = G2\ F2\ E1\ D$
- Shift left one.
 - **Output = 0I H2 G3 F3 E2 D1 C**
- If $0I\ H2\ G3 > 4$
 - $J\ I1\ H3\ G4 = 0\ I\ H2\ G3 + 0011.$
- Else
 - $J\ I1\ H3\ G4 = 0\ I\ H2\ G3$
- If $F3\ E2\ D1\ C > 4$
 - $F4\ E3\ D2\ C1 = F3\ E2\ D1\ C + 0011.$
- Else
 - $F4\ E3\ D2\ C1 = F3\ E2\ D1\ C$
- Shift left one.
 - **Output = I1 H3 G4 F4 E3 D2 C1 B**
- If $I1\ H3\ G4\ F4 > 4$
 - $I2\ H4\ G5\ F5 = I1\ H3\ G4\ F4 + 0011.$
- Else
 - $I2\ H4\ G5\ F5 = I1\ H3\ G4\ F4$
- If $E3\ D2\ C1\ B > 4$
 - $E4\ D3\ C2\ B1 = E3\ D2\ C1\ B + 0011.$
- Else
 - $E4\ D3\ C2\ B1 = E3\ D2\ C1\ B$
- Shift left one.
 - **Output= H4 G5 F5 E4 D2 C2 B1 A**
- **Let's the final output of BCD = U3U2U1U0 L3L2L1L0**



BCD

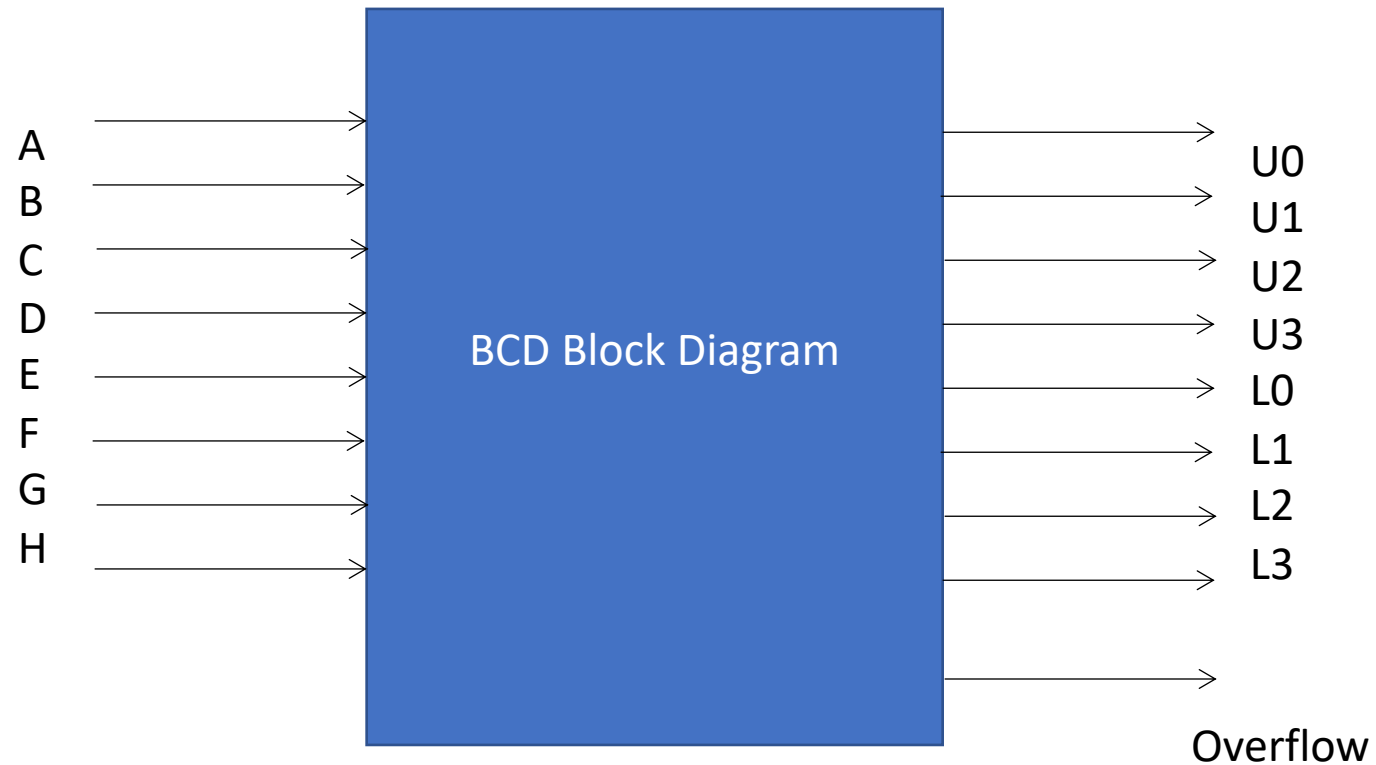


- Output= H4 G5 F5 E4 D3 C2 B1 A
- (from previous slide)
- Name the final output of BCD as
 - Output = U3U2U1U0 L3L2L1L0
- Use I2 as Overflow.

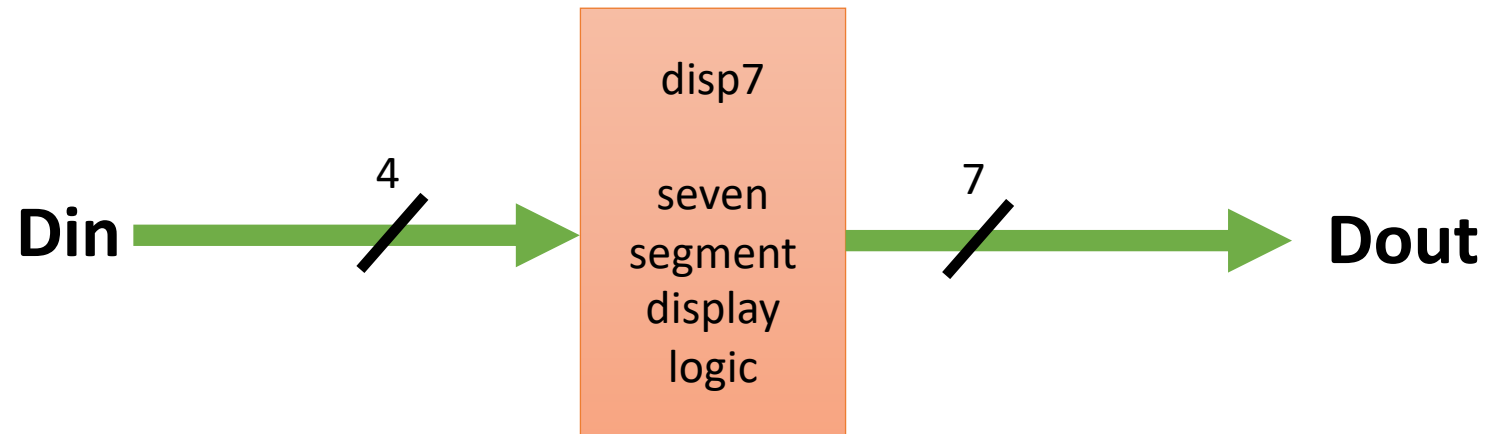
A = L0.

BCD Block Diagram (ECE315)

- Implement the BCD Ckt from previous slide in VHDL



SimpleALU – Internals – disp7



Operation:

Seven segment display logic

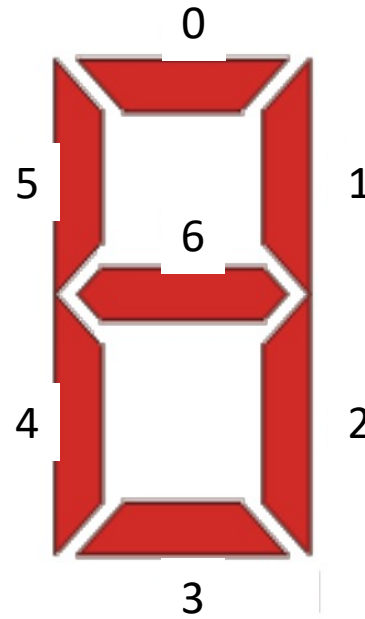
Converts 4-bit binary input to decimal value:

0,1,2,3,4,5,6,7,8,9

7-bits refer to segments 0-6 on display

7 Segment Display Patterns and Truth Table (ECE315)

	<i>Input</i> X3..X0	<i>Display</i> D6 ... D0
0	0 0 0 0	1 0 0 0 0 0 0
1	0 0 0 1	
2	0 0 1 0	
3	0 0 1 1	
4	0 1 0 0	
5	0 1 0 1	
6	0 1 1 0	
7	0 1 1 1	
8	1 0 0 0	
9	1 0 0 1	
10	1 0 1 0	X X X X X X X X
11	1 0 1 1	X X X X X X X X
12	1 1 0 0	X X X X X X X X
13	1 1 0 1	X X X X X X X X
14	1 1 1 0	X X X X X X X X
15	1 1 1 1	X X X X X X X X



The 7 segment displays on DE1 board are designed to light up when applying a low level voltage.

Applying a Vcc to display will turn off the LEDs.

K-Map (ECE315)

- We have 4 inputs and 7 outputs.
- Implement in VHDL as equations

	<i>Input</i> X3..X0	<i>Display</i> D6...D0
0	0 0 0 0	1 0 0 0 0 0 0
1	0 0 0 1	
2	0 0 1 0	
3	0 0 1 1	
4	0 1 0 0	
5	0 1 0 1	
6	0 1 1 0	
7	0 1 1 1	
8	1 0 0 0	
9	1 0 0 1	
10	1 0 1 0	X X X X X X X X
11	1 0 1 1	X X X X X X X X
12	1 1 0 0	X X X X X X X X
13	1 1 0 1	X X X X X X X X
14	1 1 1 0	X X X X X X X X
15	1 1 1 1	X X X X X X X X

To Demo - 3A

- Implement all components as VHDL
- Create a top-level design with these components
- Simulate in Modelsim
 - Show all operations for at least 2 values of X and Y
 - Show the overflow
- Note:
 - Show the 7-segment outputs as Hex values
 - Show the BCD as unsigned decimal