

**HOMEWORK V**

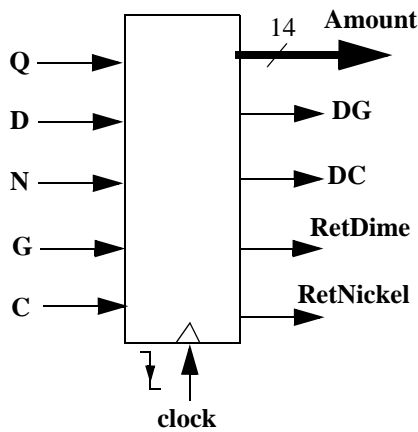
**DUE** : November 17, 2011

**READ** : Related portions of Chapters III, IV, VI, VII and VIII

**ASSIGNMENT** : There are seven questions.

**Solve all homework and exam problems as shown in class and past exam solutions.**

**1)** Consider the vending machine controller digital system designed in class. Assume that it is modified so that it accepts **quarters** :



**Inputs**

**Q** : Quarter is input  
**D** : Dime is input  
**N** : Nickel is input  
**G** : Gum is selected  
**C** : Chips is selected

**Outputs**

**Amount** : Value of gum and chips (35 cents) or the coin input so far shown on two 7-segment displays  
**DG** : Deliver Gum  
**DC** : Deliver chips  
**RetDime** : Return 10 cents  
**RetNickel** : Return 5 cents

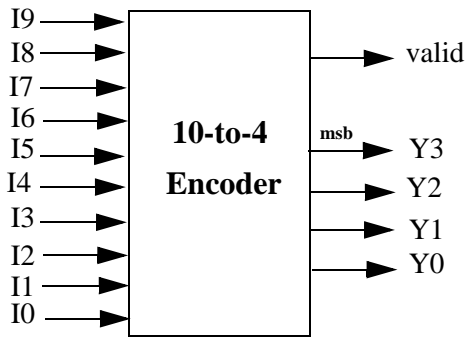
**Textual Input/Output Relationship**

After receiving the necessary amount (35 cents to 50 cents) and the selection is made deliver a gum or chips and return 5 or 10 or 25 cents if necessary

Modify the following in order to accept quarters :

- i)** The operation diagram.
- ii)** The high-level state diagram.
- iii)** The datapath.

**2)** Develop a **non-priority** 10-to-4 encoder shown below. “Non-priority” means that it is guaranteed that at any time either all inputs are 0 or only one input is 1. Provide an output named “valid” which is 1, when an encoder input is 1. The valid output is 0, when all 10 inputs are 0.



First, show the operation table of the encoder, which is a simplified truth-table since at any time no more than one input is 1. From the operation table, you will derive the minimal equations.

Then, draw the corresponding gate networks for the five outputs.

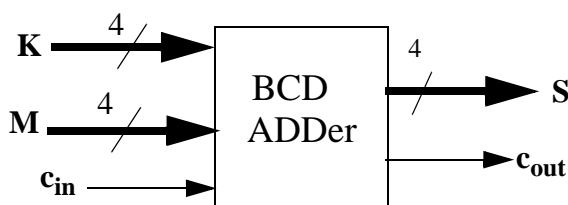
Then, indicate the **generic** gate usage, i.e. which gates, how many of each and the total generic gate count.

Students can take a look at the implementation of the **74LS147** TTL MSI chip which is a 10-to-4 **priority** encoder with active-low inputs, active-low data outputs and **no** valid output.

**3)** Consider Question 4 of Homework 4 where a complex combinational circuit is analyzed. Implement the two blocks (the ADDer and comparator) in the figure, by using **minimum** number of TTL LS chips.

Draw the full schematic by hand, showing all the connections. By using a **green** pen, outline the two blocks in your schematic.

**4)** Consider the black box view and input/output relationship of a **BCD ADDer**, a combinational circuit, shown below :



$$K + M + c_{in}$$

For example :

$$5 + 7 + 1 = (13)_{10}$$

$$S = 3 \text{ and } c_{out} \text{ is } 1$$

**Implement** the BCD ADDer on a PCB. That is, implement the BCD ADDer with chips. Use the following notes as you implement the BCD ADDer :

**i)** The circuit is a combinational circuit ! The *Digital Product Development* handout indicates that first the precise input/output relationship must be obtained : the truth table ! This is impractical for the BCD ADDer since there are **nine** inputs ! Therefore, we have to obtain the **operation table**. Then, we would proceed with the implementation step below.

ii) According to the *Digital Product Development* handout, we try to implement the BCD ADDer immediately. We answer the following questions (from *Digital Product Development*) :

- Is there a single **high-density** chip, or a few high-density chips that implements the above black box with the given operation table ? No !
- Any **programmable** chip or chips ? Yes, but for the sake of this problem, we will say “No,”
- A few **SSI** chips ? No !
- A **custom** chip implementation ? We could try it, but for the sake of this problem, we will say “No” too,
- Since there is **no** immediate implementation, we have to **partition** the BCD ADDer based on the major operations on the operation table. The BCD ADDer is not very complex which ensures that each block is immediately **implementable**.
- After the partitioning, for each block, do the following :
  - ◆ Get the input/output relationship, an operation table. Then, ask the implementation questions above. One of them will be “Yes,” so you will **not** partition anymore.

Again, you have *three* tasks : (i) you will get the **operation table** of the above black box, then (ii) determine the **major operations** and (iii) **partition** the black box into blocks that are immediately implementable by **chips** that we have discussed in class ! Draw the blocks by hand and clearly **label** wires. Precisely **indicate** which input of a block is connected to which output of a block. Do **not** implement the blocks (do **not** implement with chips).

When you partition, you will stop, if each block is immediately implementable by the first or third questions above. That is, each block is implementable by using several TTL LS chips and perhaps few SSI chips if necessary.

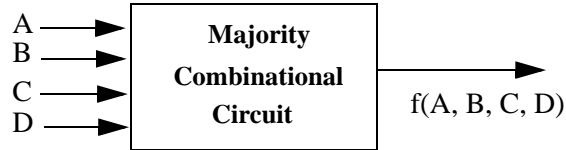
Hint : Try to make use of your answers to Homework 4 Question 4 and Question 3 above.

**5)** Solve Problem 6.70 (a) and (b).

For part (a), you will check your notes, search digital logic books and TTL data manuals to see which chip has the given internal circuit.

For Part (b), you will obtain an **operation table** to answer the textbook question : “What does the circuit do ?”

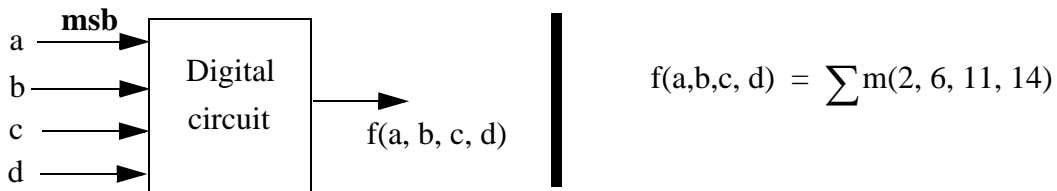
**6)** Develop a “majority” circuit with four single-rail inputs, A, B, C, D, and one output. It works such that if three or more inputs are 1, the output is 1. Otherwise, (all inputs are 0 or one input is 1 or two inputs are 1), the output is 0.



→ **First**, obtain the truth table of the function.

→ **Second**, implement the function by using a **single** 74LS151 chip. **No** other gate or chip can be used.

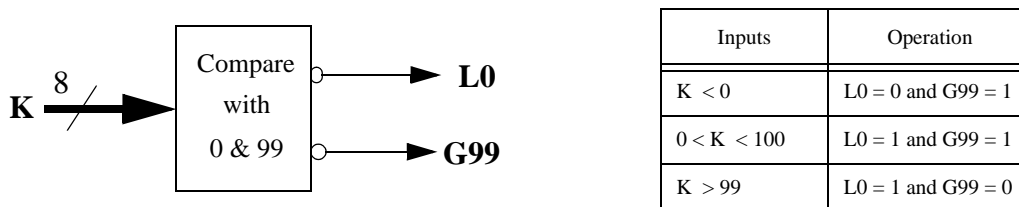
7) Consider the following circuit with four single-rail inputs :



Implement the circuit by using a single **74LS138** chip and **one** generic gate

## RELEVANT QUESTIONS AND ANSWERS

**Q1)** Consider the following circuit that compares its 8-bit 2's Complement input with 0 and 99 and has two **active-low** outputs. The black box view and operation table of this subblock are shown below :

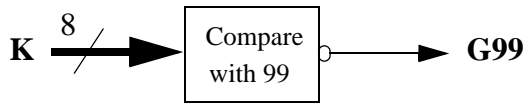


Output "L0" is 0 when input K is less than zero. Output "G99" is 0 when input K is greater than 99. Since there are eight inputs, we **partition** the black box into two blocks based on the above operation table above :



Implement the “Compare with 99” block. Follow the procedure given in the *Digital Product Development Handout* : You will implement each subsubblock of the circuit by using a few TTL LS chips for the case developing a new PCB. You will ask those questions mentioned in the *Digital Product Development Handout* to yourself during the implementation. As you recall, you are advised that you **not** use SSI chips unless it is necessary.

**A1)** We are asked to implement the following block with an 8-bit 2’s Complement input :



We know that for an 8-bit 2’s Complement number, the range is :  
 $(-128)_{10} \leq K \leq (+127)_{10}$

Below we show the 2’s Complement representation of decimal number 99 and the simplified truth table :

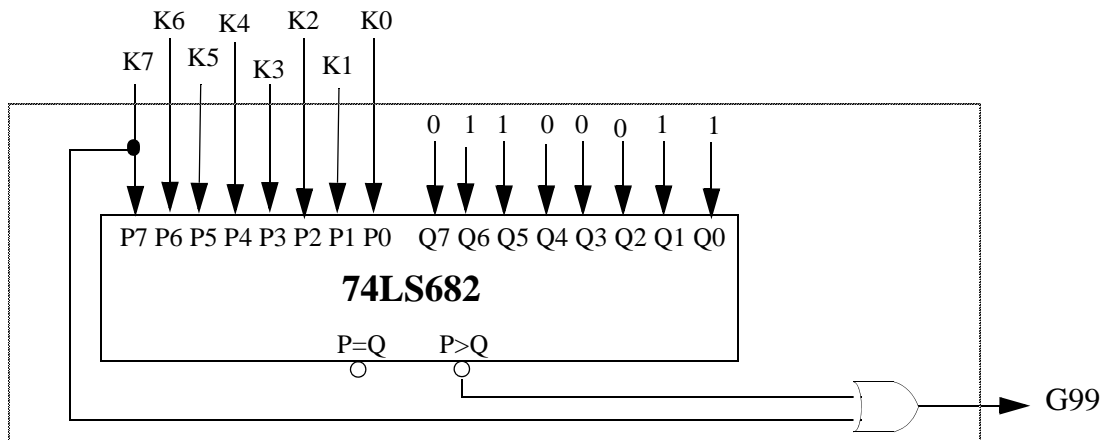
$99/2 = 49 \text{ \& } 1_{\text{lsb}}$   
 $49/2 = 24 \text{ \& } 1$   
 $24/2 = 12 \text{ \& } 0$   
 $12/2 = 6 \text{ \& } 0$   
 $6/2 = 3 \text{ \& } 0$   
 $3/2 = 1 \text{ \& } 1$   
 $1/1 = 0 \text{ \& } 1_{\text{msb}}$

$(1100011)_2$  Unsigned  
 $(01100011)_2$  2’s Complement

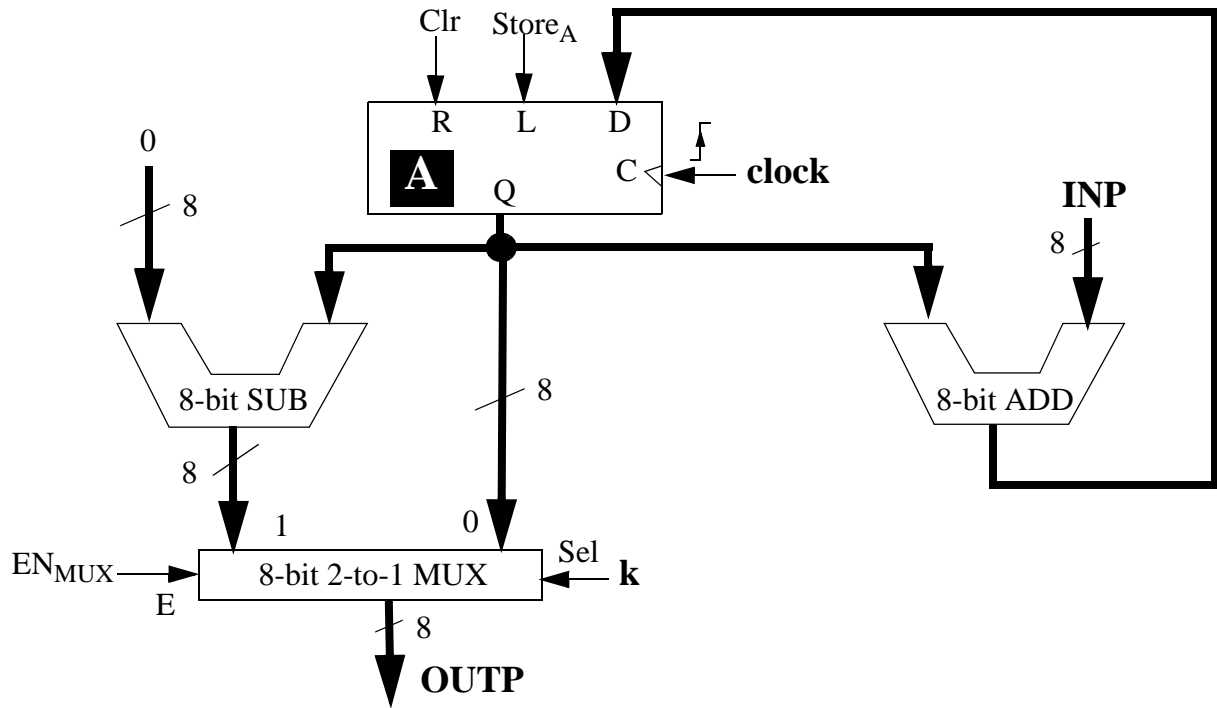
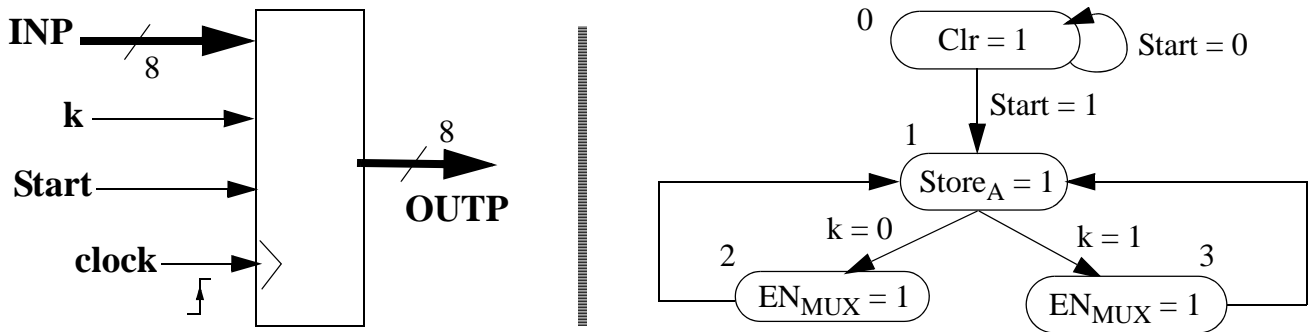
	K7	K6	K5	K4	K3	K2	K1	K0	K	G99
positive numbers	0	0	0	0	0	0	0	0	0	1
	...	...	...	...	...	...	...	...	...	...
	0	1	1	0	0	0	1	0	98	1
	0	1	1	0	0	0	1	1	99	1
	0	1	1	0	0	1	0	0	100	0
...	...	...	...	...	...	...	...	...	...	
negative numbers	0	1	1	1	1	1	1	1	127	0
	1	x	x	x	x	x	x	x	neg	1

Among the TTL chips we have, there is the 74LS682 chip which is an 8-bit unsigned comparator chip with **active-low** P=Q and P>Q outputs. While it can compare 8 bits, it cannot compare 2’s Complement numbers. For example, when K is negative, the comparator interprets it a number between 128 and 255 in decimal.

However, checking if the number is negative and bypassing the comparator chip is simple by using an OR gate. The gate output would be the G99 output. The gate would output 1 if K7 is 1 (K is negative), otherwise, its output would depend on the comparator output :



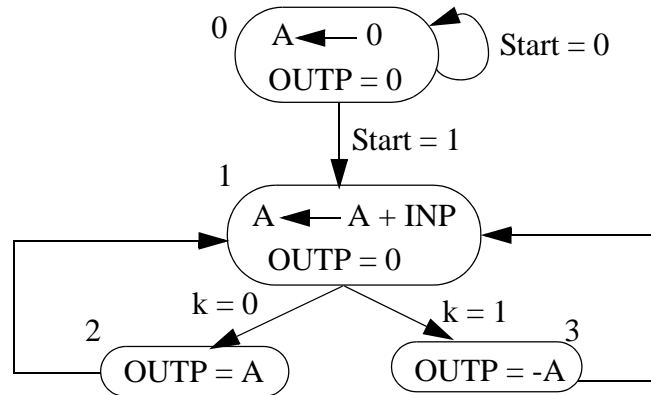
**Q2) a)** Consider the **digital system** below whose black-box view, **datapath** and **low-level** state diagram are shown. Draw the **high-level** state diagram of the digital system.



b) Show the content of the A register and OUTP for six (6) clock periods, by continuing with the table below :

Clock period	State	Start	INP	k	A	OUTP
Initial	----	?	4F	1	?	7
1	0	1	4F	1	NS	0
2	?	1	4F	1	....	....
3	?	1	4F	1	....	....
4	?	1	8	0	....	....
5	?	1	8	0	....	....
6	?	1	1E	1	....	....

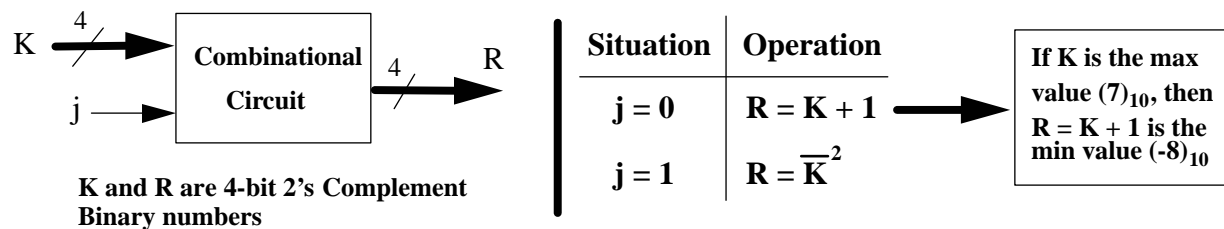
A2) a) We determine what this sequential circuit does, by completing the table shown below.



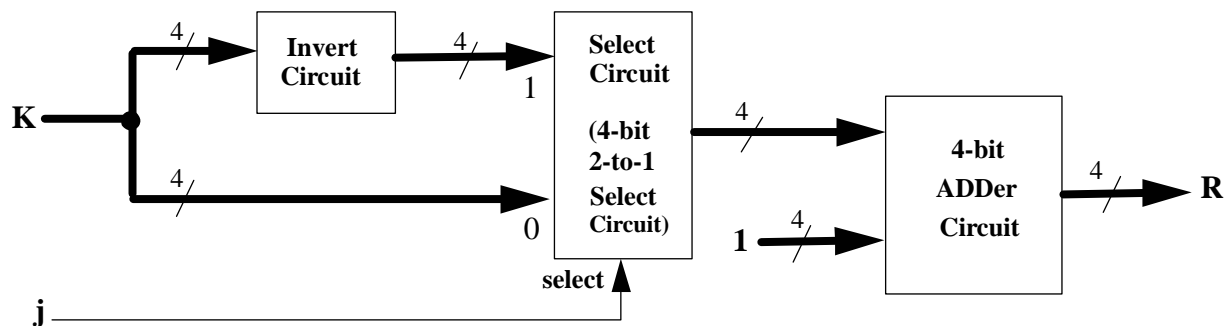
b) The table is continued below :

Clock period	State	Start	INP	k	A	OUTP
Initial	----	?	4F	1	?	7
1	0	1	4F	1	NS	0
2	1	1	4F	1	0	0
3	3	1	4F	1	4F	B1
4	1	1	8	0	NS	0
5	2	1	8	0	57	57
6	1	1	1E	1	NS	0

Q3) Consider the following 5-input, 4-output circuit and its operation table :



The following block partitioning is suggested based on the operation table :



i) Prove that this partitioning with **three** blocks is **correct**. That is, prove that the three blocks implement the operation table.

ii) Then, implement the three blocks by using **minimum** number of TTL LS chips. Draw the full circuit by showing **all** the connections. By using dotted lines, outline the three blocks in your circuit.

iii) Mention the TTL chip usage.

**A3) i)** We know that taking the 2's complement of a number is the same as complementing it and adding 1. Thus the operation table becomes as follows :

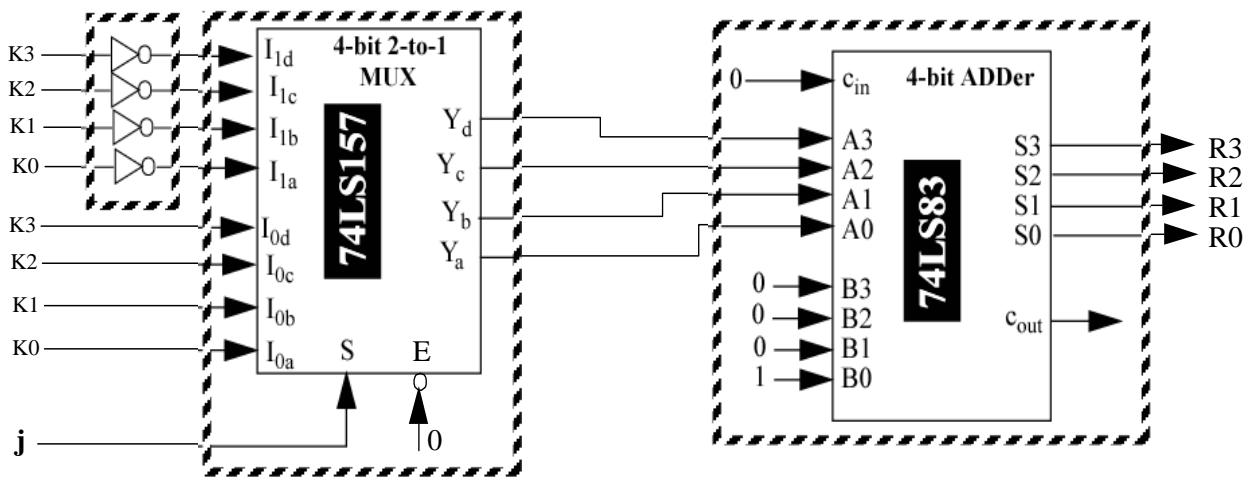
Situation	Operation
$j = 0$	$R = K + 1$
$j = 1$	$R = \overline{K}^2 = \overline{K} + 1$

a) When  $j$  is 0, the MUX selects  $K$  which is added 1 by the 4-bit ADDer to generate  $K + 1$

b) When  $j$  is 1, the MUX selects  $\overline{K}$ , generated by the invert circuit and a 1 is added by the 4-bit ADDer to generate the other output value  $\overline{K} + 1$

**Thus, the partitioning is correct, it implements the operation table.** Note that there is another partitioning given in Past Exam Question 1 above.

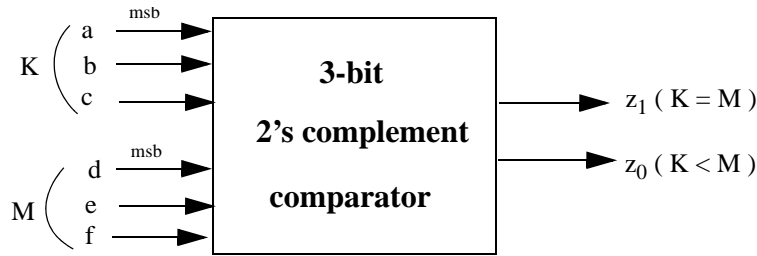
ii) The implementation of the combinational circuit by using TTL chips is shown below. We use a TTL MUX chip 74LS157 and a TTL ADDer chip 74LS83. The 74LS83 chip is a 4-bit chip with the same functionality as the 74LS283 chip. In addition, we use a TTL LS SSI chip, an inverter chip 74LS04.



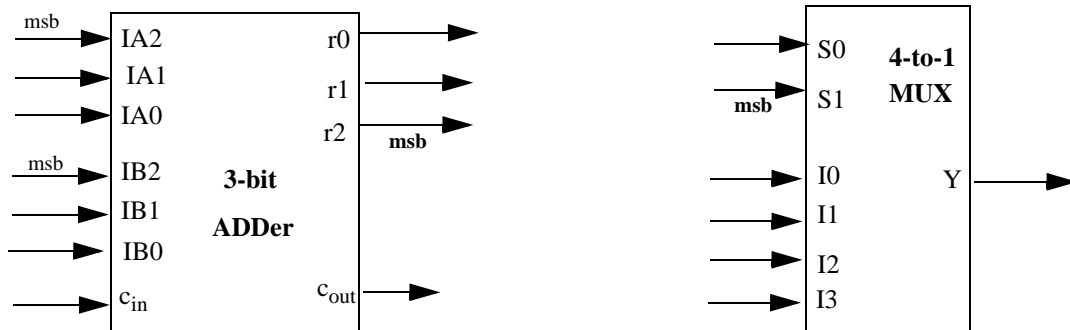
iii) The chip usage is as follows :

1 74LS04 with 6 inverters,	2 gates <b>unused</b>
1 74LS157 4-bit 2-to-1 MUX	
1 74LS83 4-bit ADDer	
<hr/>	
3 chips used. 2 gates <b>unused</b>	

**Q4)** A combinational circuit compares two 3-bit 2's complement numbers,  $K$  and  $M$ . Number  $K$  is represented by bits  $a$ ,  $b$  and  $c$ . Number  $M$  is represented by bits  $d$ ,  $e$  and  $f$ . The two active-high outputs,  $z_1$  and  $z_0$  are for “=” and “<,” respectively. The black box view of the comparator is as follows :



Design this 2's complement comparator by using only inverters, and the following two chips : a 3-bit ADDer and a 4-to-1 MUX :



Assume that **no overflow** will occur during the additions. Explain how you use the chips. Do not use any other/additional chips/gates/flip-flops.

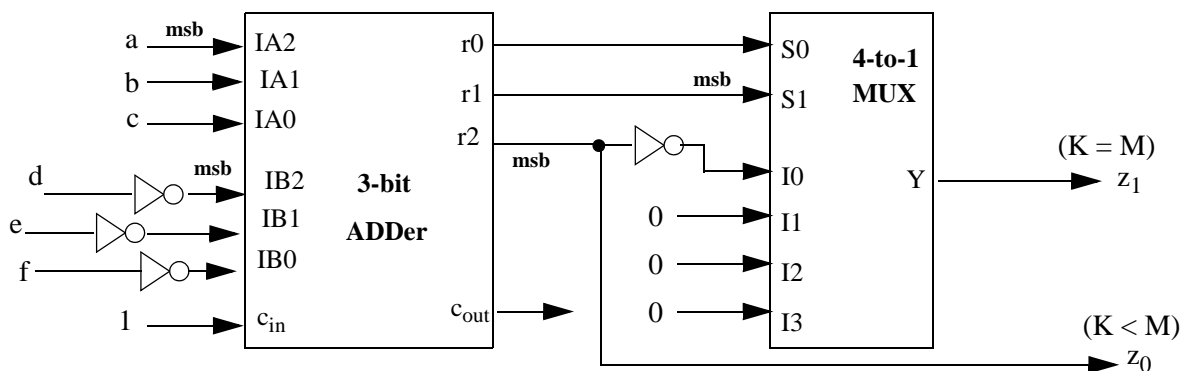
**A4)** To compare the two numbers, K and M, we have to subtract M from K :  $K - M$

$$K - M = K + (-M) = K + \overline{M} + 1$$

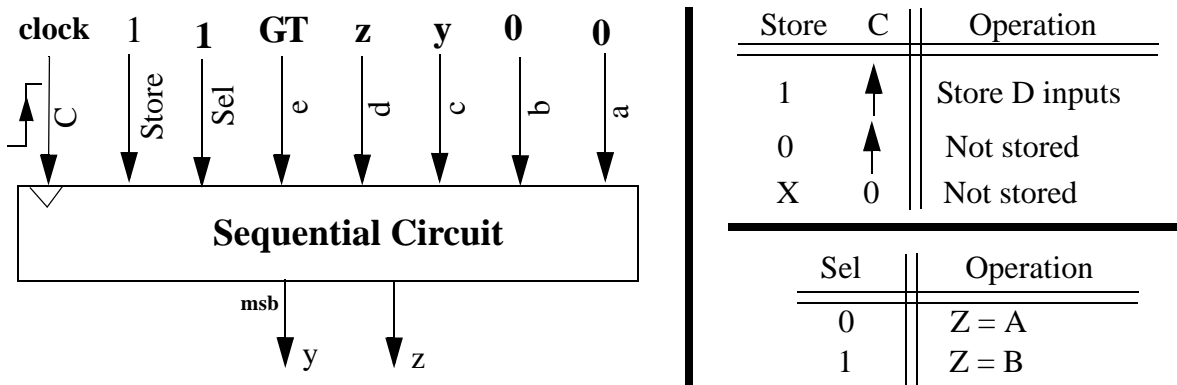
To subtract M, we complement M bits, d, e and f. Also, we input a 1 to the  $c_{in}$  input of the ADDer.

If  $K < M$ , the sign bit of the subtraction result is 1, since the result is negative. Thus, the r2 output of the ADDer is the  $z_0$  output of the comparator.

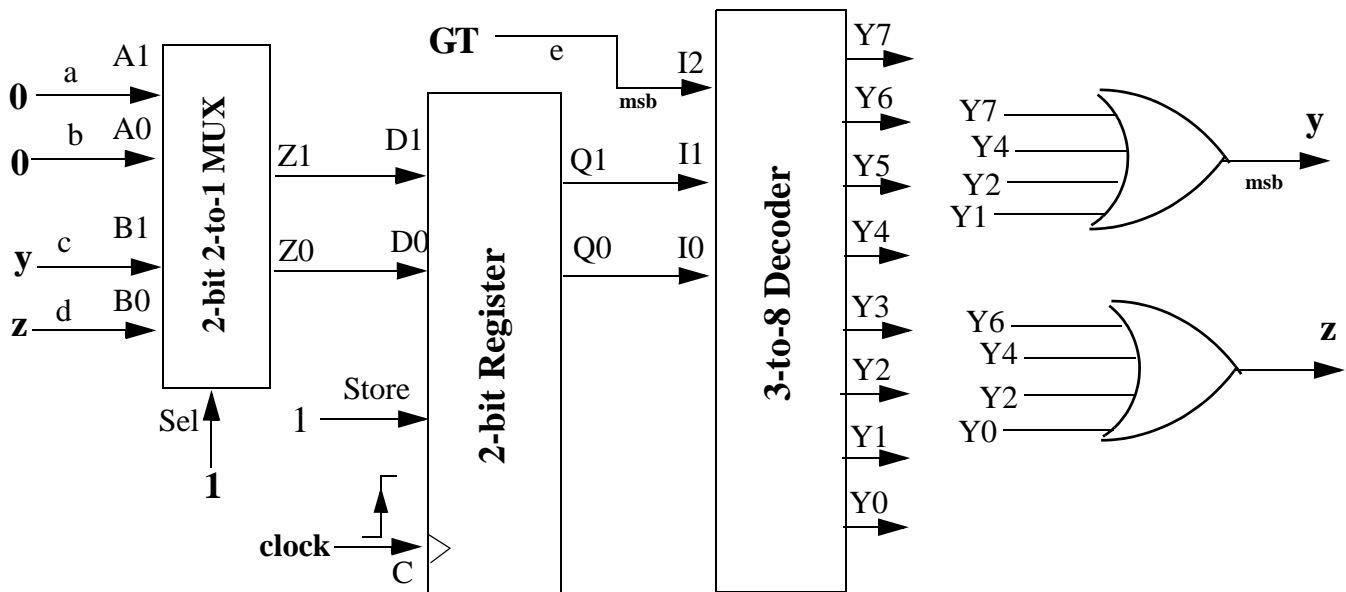
If K is equal to M, the subtraction result is 0. It is detected zero by connecting the r2, r1 and r0 outputs of the ADDer to the 4-to-1 MUX. The output of the MUX is the  $z_1$  output of the comparator. As seen below, the r1 and r0 outputs select the r2 output when they are zero. If r2 is also 0, the output of the MUX should be 1. That is why an inverter is needed to invert r2. The other data inputs of the MUX are connected 0, for cases where K is not equal to M :



**Q5)** A sequential circuit uses a 2-bit generic register and a 2-bit 2-to-1 generic MUX besides other components. Its black-box view and operation tables of the 2-bit generic register and 2-bit 2-to-1 generic MUX are shown below :



The implementation of the sequential circuit is shown below :



**Determine** what this sequential circuit does (the **purpose**) by continuing with the table below :

Time	GT	Q1 Q0	y z
t0	0	1 0	
t1	0		
t2	0		
t3	1		
t4	1		
t5	1		

Time	GT	Q1 Q0	y z
t6	1		
t7	0		
t8	1		
t9	1		
t10	0		
t11	0		

**A5)** We determine what this sequential circuit does, by completing the table shown below.

Time	GT	Q1 Q0	y z
t0	0	1 0	1 1
t1	0	1 1	0 0
t2	0	0 0	0 1
t3	1	0 1	0 0
t4	1	0 0	1 1
t5	1	1 1	1 0

Time	GT	Q1 Q0	y z
t6	1	1 0	0 1
t7	0	0 1	1 0
t8	1	1 0	0 1
t9	1	0 1	0 0
t10	0	0 0	0 1
t11	0	0 1	1 0

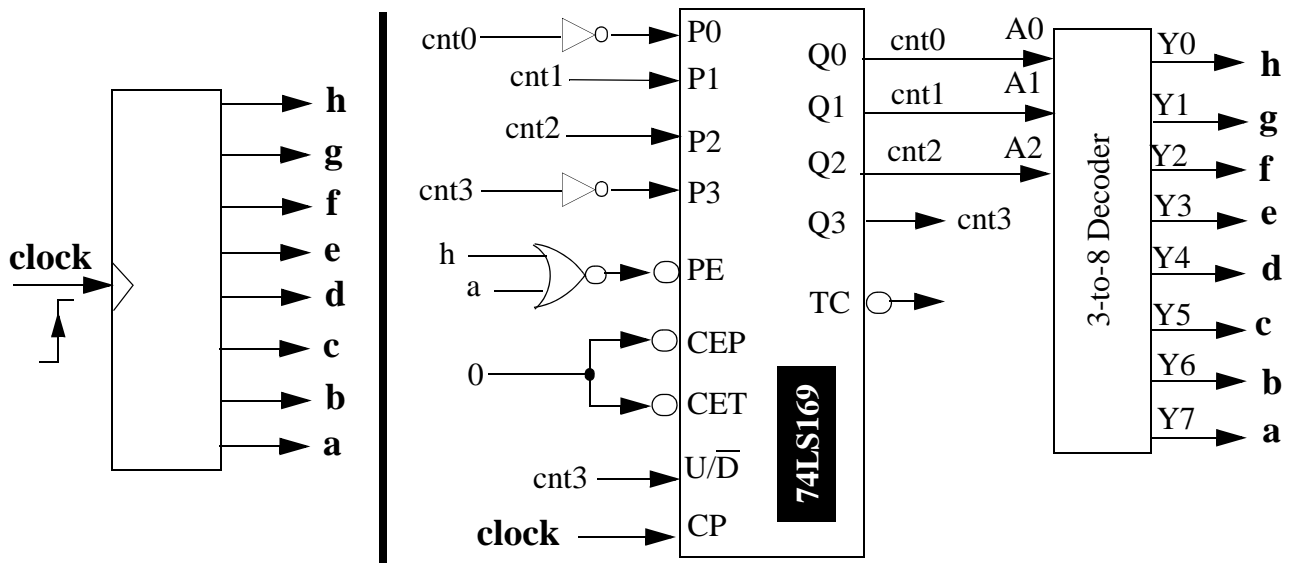
The MUX selects either (a, b) or (c, d) based on the Sel input. Since the Sel input is 1, only (c, d) inputs are used. These inputs are connected to the sequential circuit outputs (y, z).

The (y, z) values are stored every clock period since the Store signal is 1.

The 3-to-8 decoder decodes the stored (y, z) values together with the GT input. The decoder outputs are used for two OR gates to implement two combinational circuits that generate (y, z).

Altogether, the register, decoder and OR gates combination implements a 2-bit up/down counter that if GT is 0, it counts up by one otherwise, counts down by 1.

**Q6)** Consider a **sequential** circuit with a clock input and eight outputs. The black-box view and implementation of this sequential circuit are shown below. The circuit has a **74LS169** counter and a **generic** 3-to-8 decoder.



Determine what this sequential circuit does by continuing with the following table and showing the values for **16** clock periods :

Time	cnt3	cnt2	cnt1	cnt0	a	b	c	d	e	f	g	h
t0	0	1	0	1	0	0	1	0	0	0	0	0
t1	0	1	0	0	0	0	0	1	0	0	0	0
...			...									
t15			...									

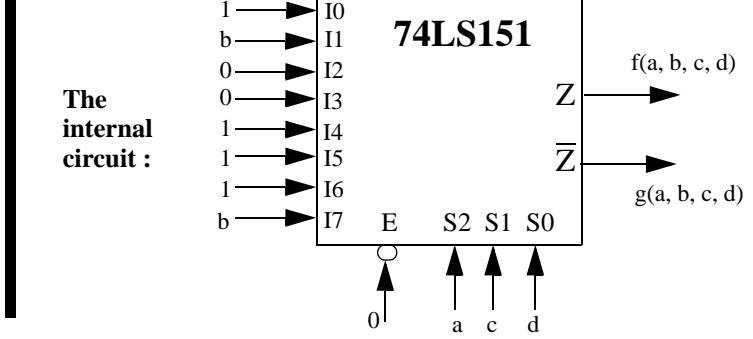
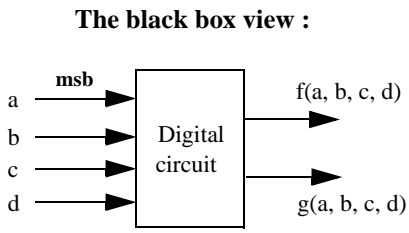
**A6)** We determine what this sequential circuit does, by completing the table for 16 clock periods :

Time	cnt3	cnt2	cnt1	cnt0	a	b	c	d	e	f	g	h
t0	0	1	0	1	0	0	1	0	0	0	0	0
t1	0	1	0	0	0	0	0	1	0	0	0	0
t2	0	0	1	1	0	0	0	0	1	0	0	0
t3	0	0	1	0	0	0	0	0	0	1	0	0
t4	0	0	0	1	0	0	0	0	0	0	1	0
t5	0	0	0	0	0	0	0	0	0	0	0	1
t6	1	0	0	1	0	0	0	0	0	0	1	0
t7	1	0	1	0	0	0	0	0	0	1	0	0
t8	1	0	1	1	0	0	0	0	1	0	0	0
t9	1	1	0	0	0	0	0	1	0	0	0	0
t10	1	1	0	1	0	0	1	0	0	0	0	0
t11	1	1	1	0	0	1	0	0	0	0	0	0
t12	1	1	1	1	1	0	0	0	0	0	0	0
t13	0	1	1	0	0	1	0	0	0	0	0	0
t14	0	1	0	1	0	0	1	0	0	0	0	0
t15	0	1	0	0	0	0	0	1	0	0	0	0

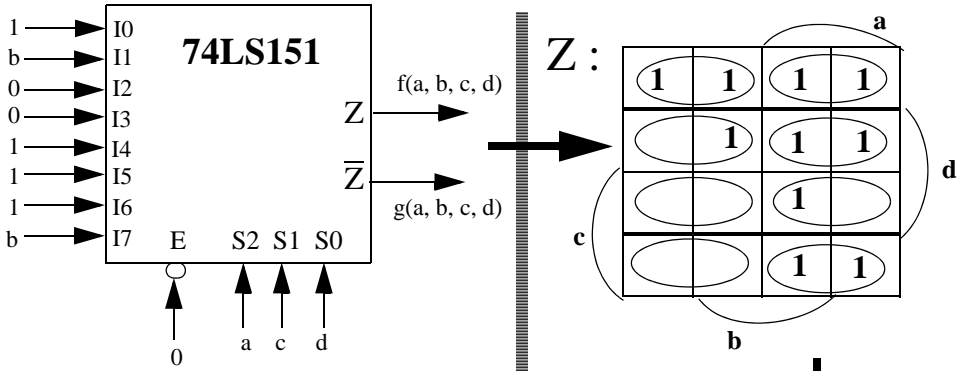
We see that the counter counts down from 6 to 0, then counts up 9 to 15 and then counts down 6 to 0 continuously. The decoder outputs follow the rightmost three bits of the counter and each becomes 1 left to right when counting down and right to left when counting up.

**Q7)** Analyze the digital circuit with four **single-rail** inputs and two outputs below.

Obtain the **truth table** of the **two functions**,  $f(a, b, c, d)$  and  $g(a, b, c, d)$ . **Then**, determine the purpose of the circuit.



**A7)** A digital circuit is given and its truth table and the purpose are asked. Since the 74LS151 is an 8-to-1 MUX, we can get the K-map of the function from which a truth table can be obtained. In addition Z and  $\bar{Z}$  are the complement of each other, therefore, obtaining the Karnaugh map of output Z is enough to obtain the complete truth table.



$$Z = f(a,b,c,d) = \sum m(0,4,5,8,9,10,12,13,14,15)$$

$$\bar{Z} = g(a,b,c,d) = \sum m(1,2,3,6,7,11)$$

The truth table of the functions :

a	b	c	d	f(a,b,c,d)	g(a,b,c,d)	a	b	c	d	f(a,b,c,d)	g(a,b,c,d)
0	0	0	0	1	0	1	0	0	0	1	0
0	0	0	1	0	1	1	0	0	1	1	0
0	0	1	0	0	1	1	0	1	0	1	0
0	0	1	1	0	1	1	0	1	1	0	1
0	1	0	0	1	0	1	1	0	0	1	0
0	1	0	1	1	0	1	1	0	1	1	0
0	1	1	0	0	1	1	1	0	1	1	0
0	1	1	1	0	1	1	1	1	1	1	0

The minterm list for f(a, b, c, d) is identical to the minterm list of a circuit studied in class : the 2-bit unsigned comparator. By studying the table, one confirms it is correct.

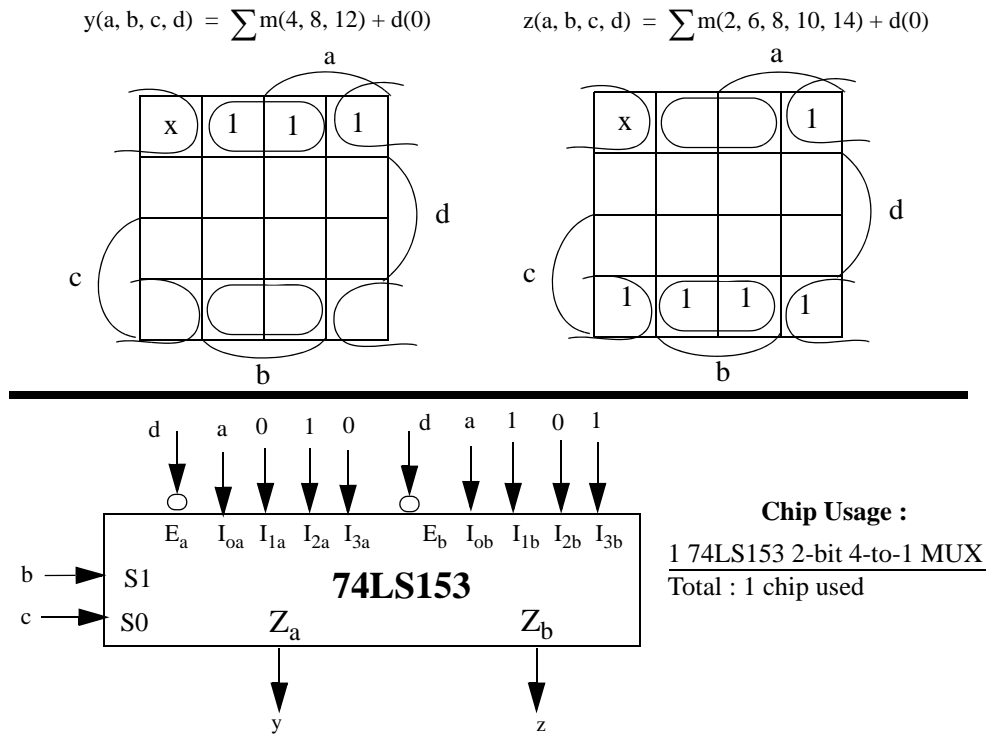
The circuit then compares two 2-bit unsigned numbers where :

- f(a, b, c, d) outputs 1
  - when G = (a, b) is **greater than or equal to** H = (c, d)
- g(a, b, c, d) outputs 1
  - when G = (a, b) is **less than** H = (c, d)

**Q8)** Consider Macro 3, **M3**, of Block 6 of the term project. It has 16 inputs and three outputs. In class it is partitioned into two pieces. One piece checks which of the four displays are zero. The other piece determines the right-most zero display and outputs the two ZERODISP lines. Implement the piece that generates the ZERODISP lines by using class notes and by using a **minimum** number of **smallest TTL MUX** chips as shown **in class**. Assume that there are only single-rail inputs. Again, you will use only **TTL MUX** chips. **No** other chip or gate can be used.

**A8)** We need to use a 16-to-1 MUX for each function to guarantee not to use an inverter. We can use an 8-to-1 MUX if we do not need an inverter. But, we see that both functions are zero when d is 1. Thus, we can use the d input as an enable input to disable the outputs when it is 1.

We can then use a 4-to-1 MUX if we carefully determine the select inputs. Since there is a TTL 4-to-1 MUX with two MUXes, we then need just one TTL MUX chip : 74LS153 :



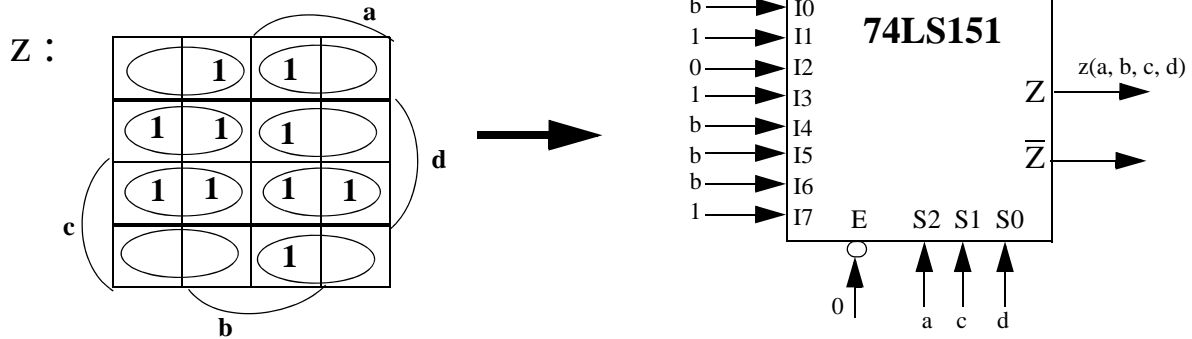
**Q9)** Consider the following switching function whose minterm list is given below :

$$y(a, b, c, d) = \sum m(1, 3, 4, 5, 7, 11, 12, 13, 14, 15)$$

Implement the function by using a **single** chip : just **one** 74LS151 MUX chip as done **in class**. Assume that there are single-rail inputs.

**A9)** The function is implemented by using a **single** 74LS151 chip :

$$z(a, b, c, d) = \sum m(1, 3, 4, 5, 7, 11, 12, 13, 14, 15)$$



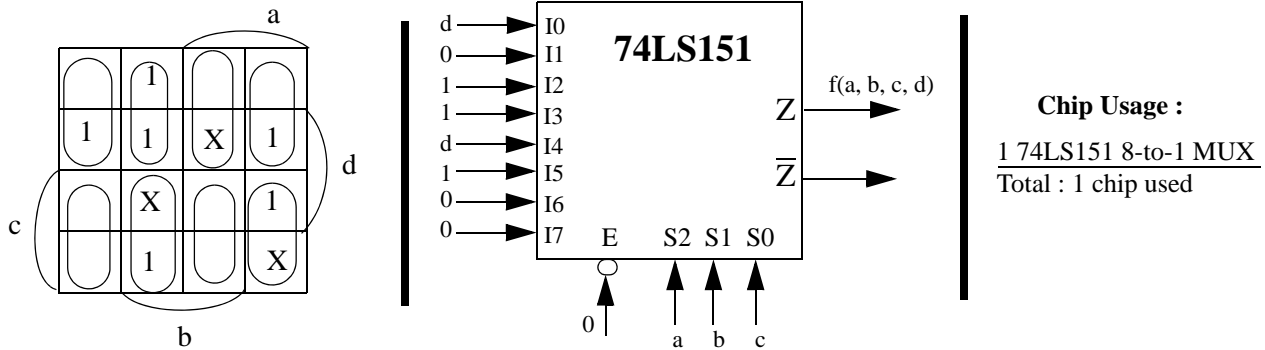
**Q10)** Consider the following minterm list :

$$f(a, b, c, d) = \sum m(1, 4, 5, 6, 9, 11) + d(7, 10, 13)$$

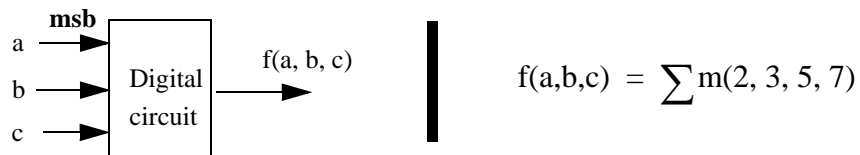
Implement the function by using a **single** chip : just **one** 74LS151 MUX chip as done **in class**. Assume that there are only **single-rail** inputs. **No** other chip/gate can be used.

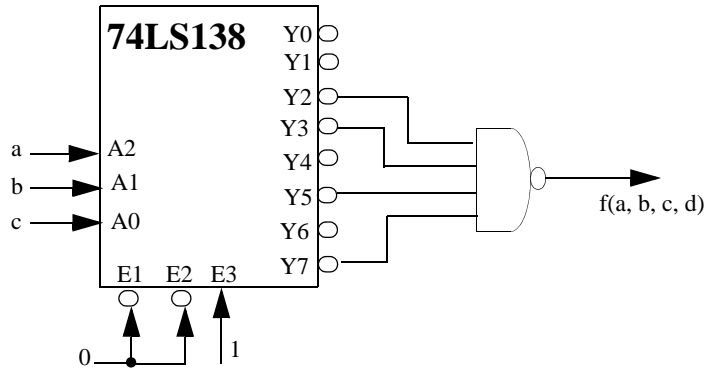
**A10)**

$$f(a, b, c, d) = \sum m(1, 4, 5, 6, 9, 11) + d(7, 10, 13)$$



**Q11)** Implement the following circuit with three **single-rail** inputs by using a **74LS138** chip and gates :





**A11)** The 74LS138 decoder chip has active-low outputs and so we need a NAND gate :

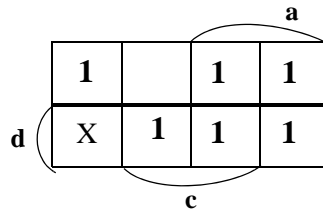
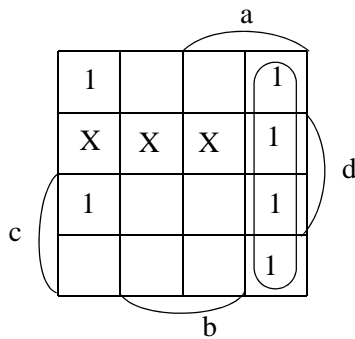
**Q12)** Consider the following minterm list :  $f(a, b, c, d) = \sum m(0, 3, 8, 9, 10, 11) + d(1, 5, 6, 13)$

Implement the function by using a **single** 74LS138 DCD chip and **one** TTL SSI chip as done **in class**. Indicate the TTL chip usage. Assume that there are only **single-rail** inputs.

**A12)** Since we have four inputs, we need a 4-to-16 decoder. But, we are asked to use a 74LS138 chip which is a 3-to-8 decoder chip ! In order to figure out how we can use the 3-to-8 decoder for this application, we need to examine its minimal SOP expression. We see that the function is 0 when input “b” is 1 and so a 3-to-8 Decoder would work. We see that the function must output 0 when (a, c, d) is (010) or combination 2. The circuit is then as follows :

$$f(a, b, c, d) = \sum m(0, 3, 8, 9, 10, 11) + d(1, 5, 6, 13)$$

The function can be described in terms of **a**, **c** and **d** when **b** is 0 as follows :

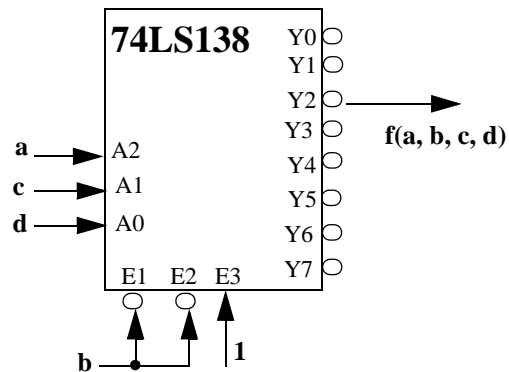


$$f(a, c, d) = \sum m(0, 3, 4, 5, 6, 7) + d(1)$$

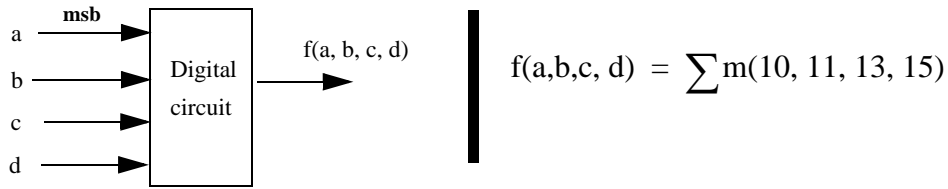
Then, the circuit with a 74LS138 decoder chip and one inverter is as follows :

**The chip usage :**

- 1 74LS138 3-to-8 Decoder
- 1 74LS04 w/ 6 inverters, 5 gates **unused**
- 2 chips used. 5 gates **unused**



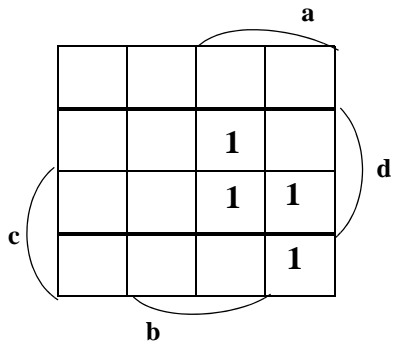
**Q13)** Implement the following circuit with four single-rail inputs by using a **74LS138** chip and **one** generic gate :



**A13)** Since we have four inputs, we need a **4-to-16** decoder. However, we are asked to use a **74LS138** chip which is a **3-to-8** decoder chip !

In order to figure out how we can use the 3-to-8 decoder for this application, we need to examine its Karnaugh map :

$$f(a,b,c, d) = \sum m(10, 11, 13, 15)$$

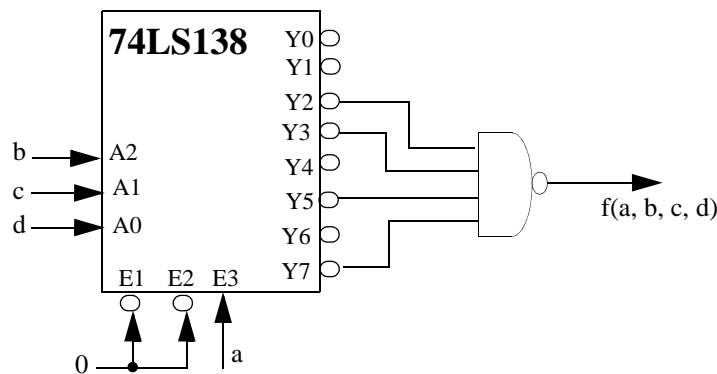


We realize that the function is zero when **a** is 0. That is, for input combinations 0, 1, 2, 3, 4, 5, 6 and 7, the output is zero.

When **a** is 1, the output depends on the other inputs **b**, **c** and **d**. That is, when a is 1, the function can be described in terms of b, c and d as follows :

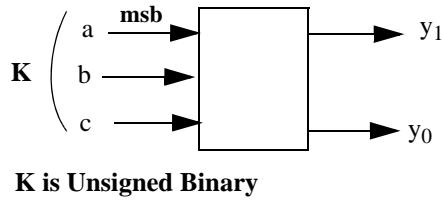
$$f(b, c, d) = \sum m(2, 3, 5, 7)$$

We can then use input “**a**” to **enable** the decoder when a is 1. The remaining three inputs control the decoder **data** inputs. Then, the circuit with a 74LS138 decoder chip and one NAND gate is as follows :



**Q14)** Consider the combinational circuit whose black-box view and input/output relationship are shown below.

Implement the circuit by using **one minimum** size TTL **Decoder** chip and minimum number of TTL **SSI** chips. Your answer will include **(i)** the truth table, **(ii)** the minterm lists, **(iii)** the circuit, and **(iv)** the TTL chip usage.



$y_1$  is 1, if  $K$  is divisible by three (3)  
 $y_0$  is 1, if  $K$  is divisible by five (5)

**A14)** The truth table is as follows :

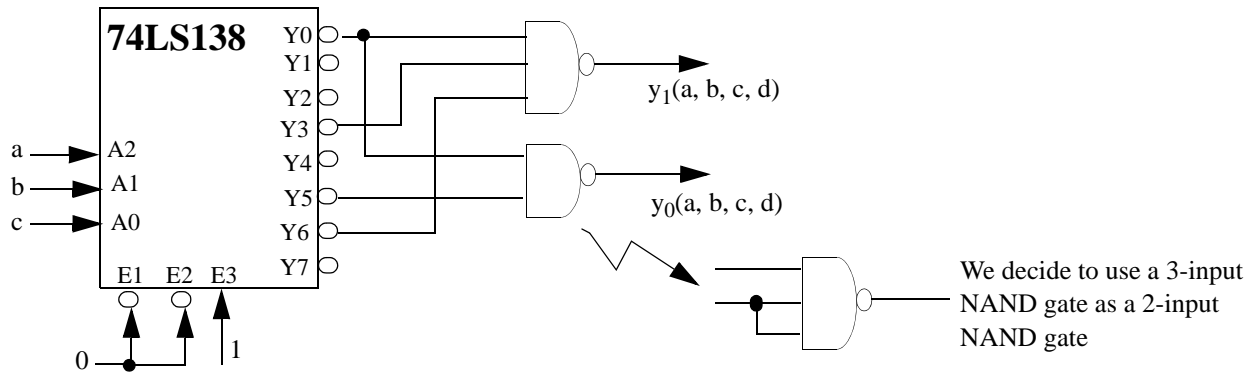
a b c	$y_1$	$y_0$
0 0 0	1	1
0 0 1	0	0
0 1 0	0	0
0 1 1	1	0
1 0 0	0	0
1 0 1	0	1
1 1 0	1	0
1 1 1	0	0

The minterm lists are as follows :

$$y_1(a, b, c) = \sum m(0, 3, 6)$$

$$y_0(a, b, c) = \sum m(0, 5)$$

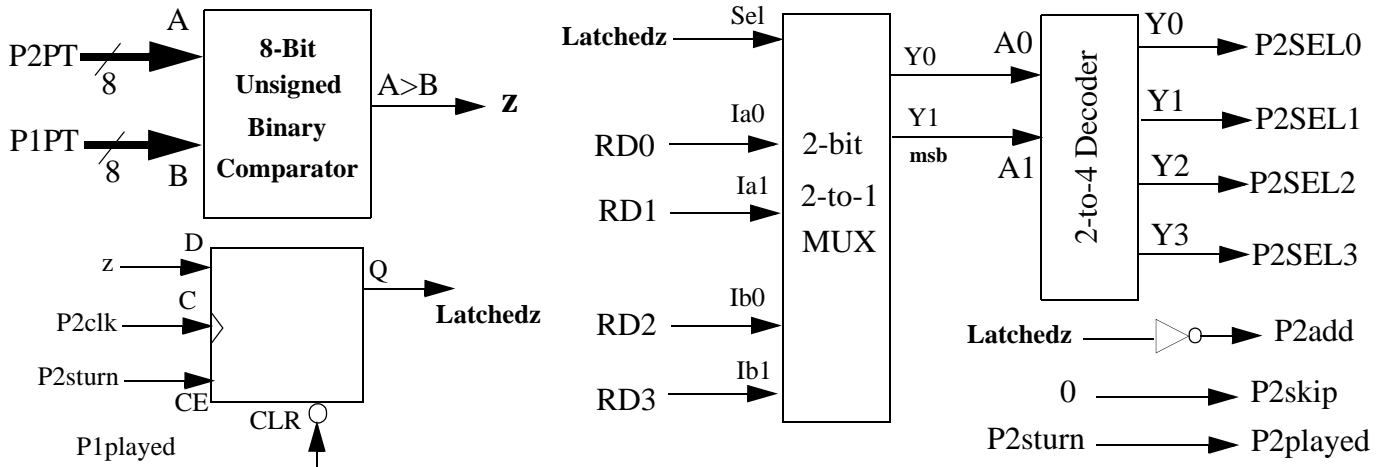
The circuit with the TTL chip usage is as follows :



**The chip usage :**

- 1 74LS138 3-to-8 Decoder
- 1 74LS10 w/3 3-input NANDs, 1 gate **unused**
- 2 chips used. 1 gate **unused**

**Q15)** Consider the following implementation of **Block 6**, the Machine Play Block, of the **term project** :



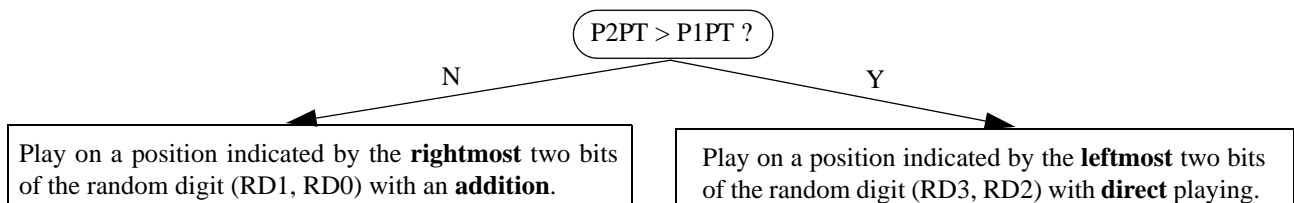
**RD3, RD2, RD1 and RD0** are the four bits of random digit, **RD**. **P2sturn** is 1 in State 4 of the Ppm operation diagram. Multiplexer inputs Ia0 and Ia1 are selected if Sel is 0. Output “z” is stored on a FF so that it can be used in State 5 and State 6 of the term project.

- i) Draw the **flowchart** of the playing strategy of the machine player.
- ii) How many clock periods does the machine player take to play ? **Explain**.
- iii) Assume that the code is **98**. Below, a list of values is given. The first row shows how the machine player plays by using the strategy implemented by the circuit above. Complete the remaining rows of the table. You will **circle** the position played as shown on the first row :

RD	Displays <b>before</b> play				P2PT	P1PT	Displays <b>after</b> play				D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
	PD3	PD2	PD1	PD0			PD3	PD2	PD1	PD0				
7	2	A	0	7	5C	4A	2	A	7	7	Direct	1	14	Yes
3	C	F	F	F	38	62								
0	2	8	8	8	4A	E6								
9	9	A	0	9	A5	CF								
8	8	C	6	A	9A	75								

Note that the cases on the table are **independent** of each other. That is, they do **not** follow each other with respect to time. Above, on the table, D/A means **Direct/Add** which means the player plays either directly or with an addition.

**A15) i)** The graph of the machine player strategy is as follows :

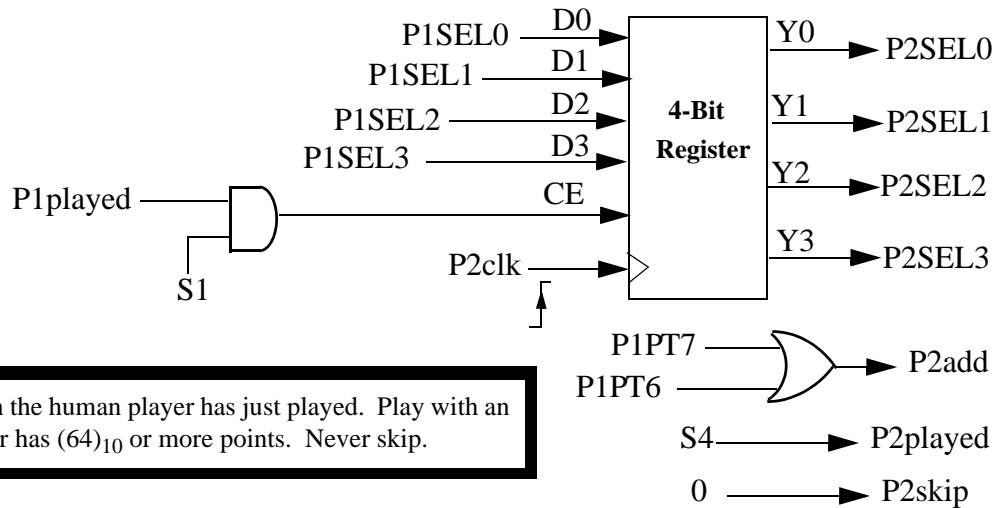


ii) The machine player takes **one** clock period since P2played is 1 as soon as P2sturn is 1.

iii) The table is completed as follows :

RD	Displays <b>before</b> play				P2PT	P1PT	Displays <b>after</b> play				D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
	PD3	PD2	PD1	PD0			PD3	PD2	PD1	PD0				
7	2	A	0	7	5C	4A	2	A	7	7	Direct	1	14	Yes
3	C	F	F	F	38	62	F	F	F	F	Add	3	120	Yes
0	2	8	8	8	4A	E6	2	8	8	8	Add	2	96	Yes
9	9	A	0	9	A5	CF	9	A	9	9	Add	1	90	Yes
8	8	C	6	A	9A	75	8	6	A	A	Direct	1	16	Yes

**Q16)** Consider Block 6, the Machine Play Block, of the **term project**. The circuit that implements Block 6 for a specific playing strategy is shown below.



Always play on the position the human player has just played. Play with an addition if the human player has  $(64)_{10}$  or more points. Never skip.

The strategy implemented is as follows : Always play on the position the human player has just played. Play with an addition if the human player has  $(64)_{10}$  or more points. Never skip.

**NSD1** and **NSD0** indicate the adjacency. **S1** and **S4** indicate State 1 and State 4 of the Ppm operation diagram, respectively. **P1SEL** signals indicate where Player 1 has just played. **P1PT7** and **P1PT6** are the leftmost two bits of the Player 1 points. **P1playsynch** and **P2playsynch** are the synchronized P1play and P2play signals connected to BTN1 and BTN2, respectively.

i) How many clock periods does the machine player take to play ? **Explain**.

ii) Assume that the code is **A8**. The table below shows the random digit, position displays **before** and **after** the above **machine** player plays, the Player 1 points, the Player 1 last play, whether the random digit is played directly or added, the number of adjacencies, the random reward, the points earned by the **machine** player and whether the machine player plays again. Complete the remaining rows of the table. You will **circle** the position played as shown on the first row. The meaning of **D/A** is Direct/Add which is whether the machine player plays the random digit **directly** on

a position or by **adding** to a position. Note that the cases on the table are **independent** of each other. That is, they do **not** follow each other with respect to time.

**iii)** Modify the above circuit so that the machine player plays with an addition only if there is no display overflow on the selected position by the machine player. Just show the **modified** portion of the circuit, **not** the whole circuit.

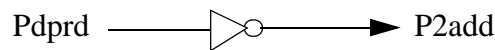
RD	Displays <b>before</b> play PD3 PD2 PD1 PD0				P1PT	P1SEL	Displays <b>after</b> play PD3 PD2 PD1 PD0				D/A	NSD	Reward Points (Decimal)	Plays Again ?
2	5	F	F	D	5C	0001	5	F	F	(F)	Add	2	60	Yes
7	7	E	E	E	8A	1000								
5	5	A	A	A	29	0100								
1	7	3	C	D	C8	0010								
8	C	8	8	F	3E	0001								

**A16) i)** The machine player takes **one** clock period as there is no sequencer with a counter+decoder combination.

**ii)** The table is completed as follows :

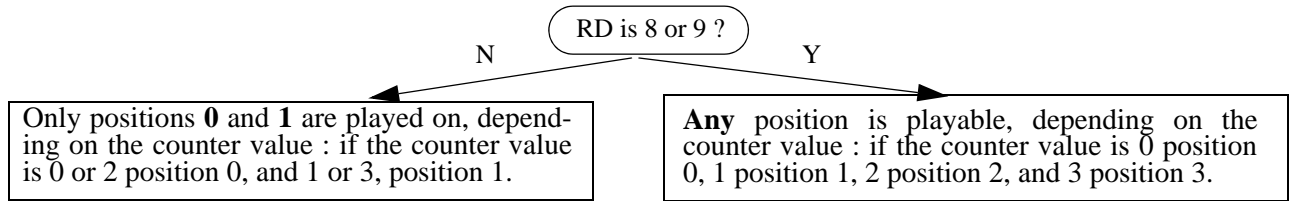
RD	Displays <b>before</b> play PD3 PD2 PD1 PD0				P1PT	P1SEL	Displays <b>after</b> play PD3 PD2 PD1 PD0				D/A	NSD	Reward Points (Decimal)	Plays Again ?
2	5	F	F	D	5C	0001	5	F	F	(F)	Add	2	60	Yes
7	7	E	E	E	8A	1000	(E)	E	E	E	Add	3	112	Yes
5	5	A	A	A	29	0100	5	(5)	A	A	Direct	1	10	Yes
1	7	3	C	D	C8	0010	7	3	(D)	D	Add	1	26	Yes
8	C	8	8	F	3E	0001	C	8	8	(8)	Direct	2	96	Yes

**iii)** The modified circuit is as follows :



**Q17)** Consider **Block 6**, the Machine Play Block, of the **term project**. Its playing strategy is described by the graph below.

As the flowchart shows it **never** skips **nor** plays directly. It has a **random** playing strategy that when it is machine player's turn, a freely running counter is stopped. The output of the counter and the rightmost bit of RD (RD3) are used to decide on which position to play.

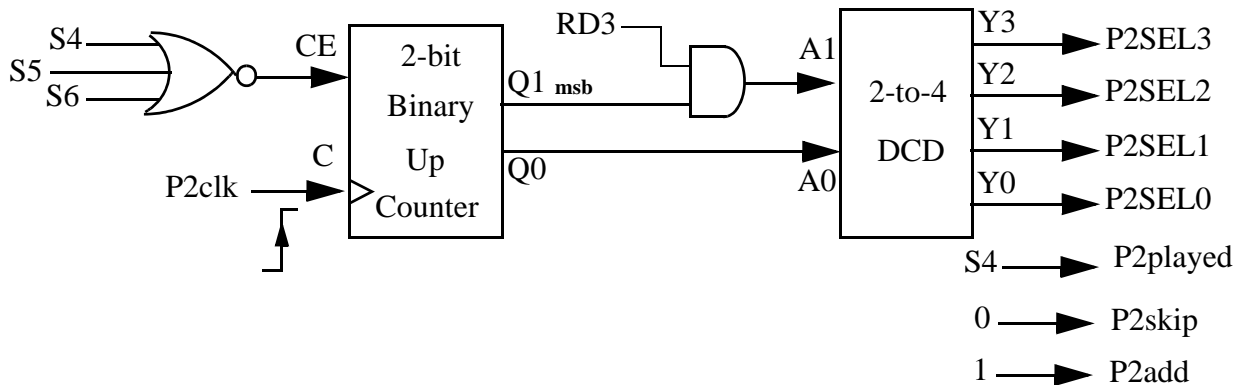


i) Implement Block 6 based on the playing strategy above.

ii) Assume that the code is **3A**. Below, a list of RD and DISP values is given. The first row shows how the machine player plays by using the strategy implemented by the circuit above. You will **circle** the position played as shown on the first row. Note that the cases on the table are **independent** of each other. That is, they do **not** follow each other with respect to time. Above, on the table, D/A means **Direct/Add** which means the player either plays directly or with an addition.

RD	Q1 Q0 in State 4	RD3	A1 A0 In State 4	Displays <b>before</b> play PD3 PD2 PD1 PD0	Displays <b>after</b> play PD3 PD2 PD1 PD0	D/A	Adjacency	Reward Points <b>(Decimal)</b>	Plays again ?
7	1 1	0	0 1	A 2 3 F	A 2 (A) F	Add	0	10	No
4	1 0			C 6 A 6					
2	0 1			C C A 8					
9	1 1			F 8 8 8					
0	0 0			5 9 9 9					

**A17)** i) Below, **RD3** is the most significant bit of random digit RD. **S4, S5** and **S6** indicate State 4, State 5 and State 6 of the Ppm **operation diagram**, respectively. The **CE** input of the counter is the clock enable input. **P2clk** is the high-speed Player 2 clock.



ii) The table is completed as follows :

RD	Q1 Q0 in State 4	RD3	A1 A0 In State 4	Displays <b>before</b> play PD3 PD2 PD1 PD0	Displays <b>after</b> play PD3 PD2 PD1 PD0	D/A	Adjacency	Reward Points <b>(Decimal)</b>	Plays again ?
7	1 1	0	0 1	A 2 3 F	A 2 (A) F	Add	0	10	No
4	1 0	0	0 0	C 6 A 6	C 6 A (A)	Add	1	100	Yes
2	0 1	0	0 1	C C A 8	C C (C) 8	Add	2	48	Yes
9	1 1	1	1 1	F 8 8 8	(8) 8 8 8	Add	3	64	Yes
0	0 0	0	0 0	5 9 9 9	5 9 9 (9)	Add	2	36	Yes

**Q18)** Consider Block 6, the Machine Play Block, of the **term project**. Assume that the machine player has the following playing strategy :

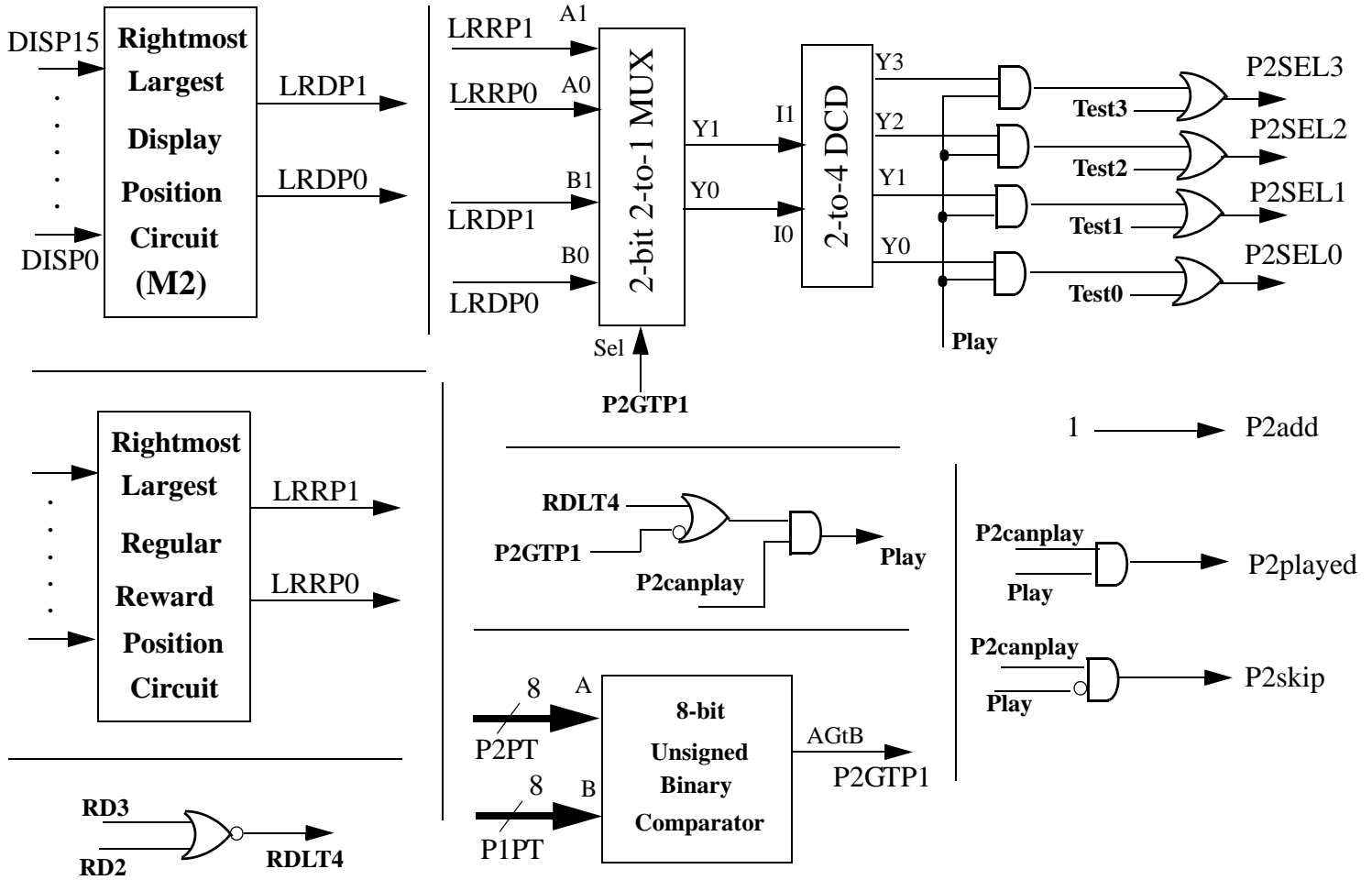
Play on the  
(rightmost)  
largest display  
position with  
an addition

**a)** Assume that the code is **58**. The table below shows the random digit, position displays **before** and **after** the **machine** player plays, if the machine player is ahead (P2GTP1 = Yes) **before** the play, whether the random digit is played directly or added, the number of adjacencies, the points earned by the **machine** player and if the machine plays again. Complete the rows of the table. You will **circle** the position played :

RD	Displays <b>before</b> play PD3 PD2 PD1 PD0	P2GTP1	Displays <b>after</b> play PD3 PD2 PD1 PD0	D/A	Adjacency	Reward Points <b>(Decimal)</b>	Plays Again ?
5	0 7 0 0	Yes					
2	8 C 0 0	No					
6	8 E 6 0	No					
3	A E 6 3	Yes					
7	F E 8 3	No					

**b)** Assume that the above machine player is modified to have a different strategy. The **datapath** of the circuit that implements the modified machine player for the new playing strategy is below.

In the figure, M2 is Macro 2 of Block 6, as designed in the lab. P2GTP1 means P2PT > P1PT. This is the case **before** the play. **P2canplay** is 1 in the last Player 2 state. RD3 and RD 2 are the leftmost two bits of the random digit.



i) Draw the **flowchart** of the playing strategy of the modified machine player.

ii) How many clock periods does the machine player take to play ? **Explain.**

iii) Assume again that the code is **58**. The table below shows the same values as the table above : The random digit, position displays **before** and **after** the **machine** player plays, if the machine player is ahead (P2GTP1 = Yes) **before** the play, whether the random digit is played directly or added, the number of adjacencies, the points earned by the **machine** player and if the machine player plays again.

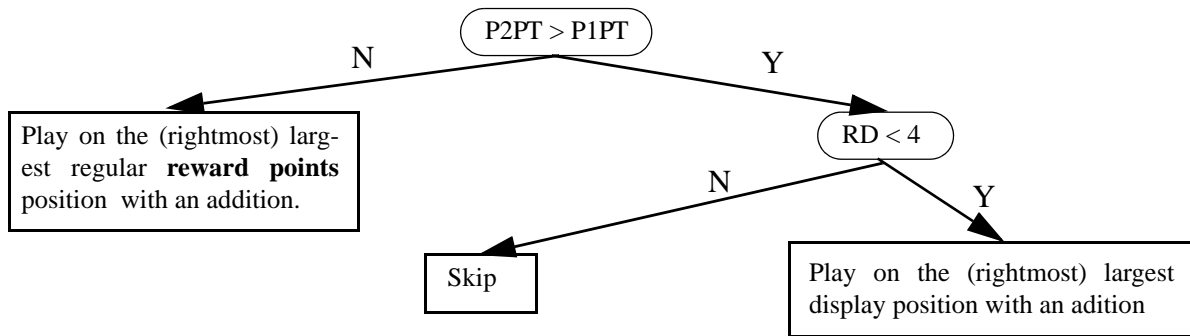
RD	Displays <b>before</b> play				P2GTP1	Displays <b>after</b> play				D/A	Adjacency	Reward Points <b>(Decimal)</b>	Plays Again ?
	PD3	PD2	PD1	PD0		PD3	PD2	PD1	PD0				
5	0	7	0	0	Yes								
2	8	C	0	0	No								
6	8	E	6	0	No								
3	A	E	6	3	Yes								
7	F	E	8	3	No								

Complete the rows of the table. You will **circle** the position played :

**A18) a)** The table is completed as follows :

RD	Displays <b>before</b> play				P2GTP1	Displays <b>after</b> play				D/A	Adjacency	Reward Points <b>(Decimal)</b>	Plays Again ?
	PD3	PD2	PD1	PD0		PD3	PD2	PD1	PD0				
5	0	7	0	0	Yes	0	(C)	0	0	A	0	12	N
2	8	C	0	0	No	8	(E)	0	0	A	0	14	N
6	8	E	6	0	No	8	(4)	6	0	A	0	4	N
3	A	E	6	3	Yes	A	(1)	6	3	A	0	1	N
7	F	E	8	3	No	(6)	E	8	3	A	0	6	N

**b) i)**



**ii)** The machine player takes **five** clock period to play. It does only additions, hence it takes four clock periods to collect the four regular reward points and then one clock period to play.

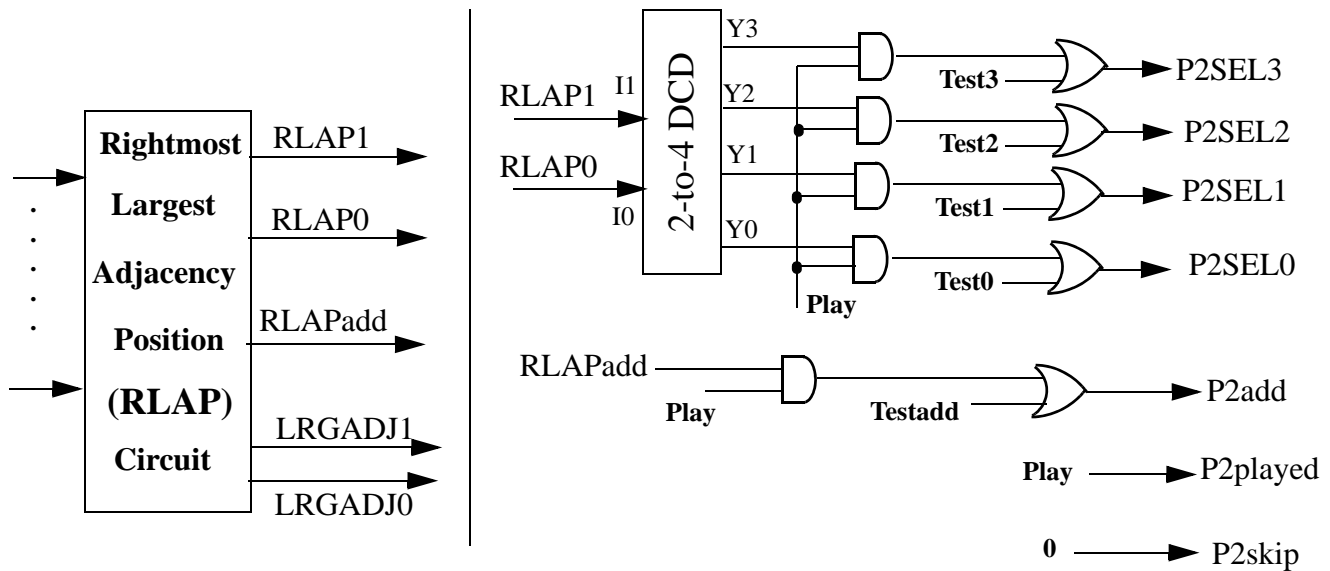
**iii)** The table is completed as follows :

RD	Displays <b>before</b> play				P2GTP1	Displays <b>after</b> play				D/A	Adjacency	Reward Points <b>(Decimal)</b>	Plays Again ?
	PD3	PD2	PD1	PD0		PD3	PD2	PD1	PD0				
5	0	7	0	0	Yes	0	7	0	0	Skip	Skip	Skip	Skip
2	8	C	0	0	No	8	(E)	0	0	A	0	14	N
6	8	E	6	0	No	(E)	E	6	0	A	1	28	Y
3	A	E	6	3	Yes	D	(1)	6	3	A	0	1	N
7	F	E	8	3	No	F	E	(F)	3	A	0	15	N

**Q19)** Consider Block 6, the Machine Play Block, of the **term project**. Assume that its strategy is a sfollow :

Play on the (rightmost) largest adjacency position (directly if equal)

The **datapath** of the circuit that implements the above strategy is shown below :



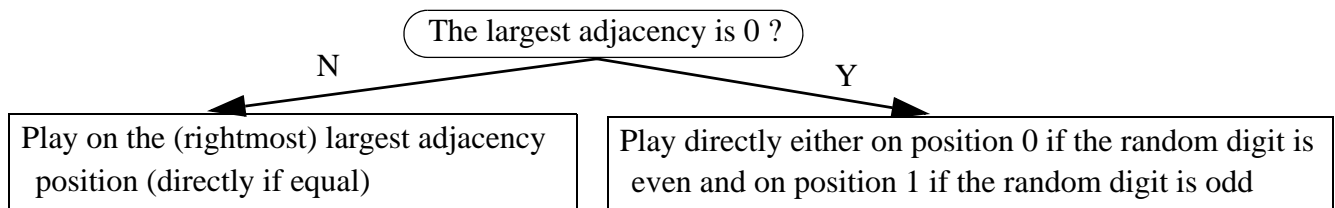
In the figure, **RLAPadd** is 1 if an addition is needed to play on the rightmost largest adjacency position. **Play** is 1 in the last Player 2 state.

**a) i)** Assume that the code is **A7**. The table below shows the random digit, position displays **before** and **after** the **machine** player plays, whether the random digit is played directly or added, the number of adjacencies, the points earned by the **machine** player and if the machine player plays again. Complete the rows of the table. You will **circle** the position played :

RD	Displays <b>before</b> play					Displays <b>after</b> play				D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
	PD3	PD2	PD1	PD0		PD3	PD2	PD1	PD0				
8	F	F	F	F									
6	F	5	5	5									
4	6	A	6	7									
7	0	0	0	0									
3	1	E	8	E									

**ii)** How many clock periods does the machine player take to play ? **Explain.**

**b) i)** Assume that the above machine player is modified to have the following **new** strategy :



Assume again that the code is **A7**. The table below shows the random digit, position displays **before** and **after** the **machine** player plays, whether the random digit is played directly or added, the number of adjacencies, the points earned by the **machine** player and if the machine player plays again. Complete the rows of the table. You will **circle** the position played :

RD	Displays <b>before</b> play PD3 PD2 PD1 PD0	Displays <b>after</b> play PD3 PD2 PD1 PD0	D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
8	F F F F					
6	F 5 5 5					
4	6 A 6 7					
7	0 0 0 0					
3	1 E 8 E					

**ii) Modify** the above circuit (the datapath) to implement the **new** strategy (two rectangles and one oval). You can just **show** the **modified** portion of the circuit, **not** the whole circuit.

**A19) a) i)** The table is completed below.

The strategy does not check for code digits and so misses to earn code reward points when the random digit is 8. However, it earns code reward points by coincidence when the random digit is 4 and 7.

RD	Displays <b>before</b> play PD3 PD2 PD1 PD0	Displays <b>after</b> play PD3 PD2 PD1 PD0	D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
8	F F F F	F F F (8)	D	0	8	N
6	F 5 5 5	(5) 5 5 5	A	3	40	Y
4	6 A 6 7	6 A (A) 7	A	1	187	Y
7	0 0 0 0	0 0 0 (7)	D	0	63	N
3	1 E 8 E	1 (1) 8 E	A	1	2	Y

**ii)** The machine player takes **nine** clock period to play. It does direct playing and additions, hence it takes eight clock periods to collect the eight adjacencies and then one clock period to play.

**b) i)** The table is completed as follows :

RD	Displays <b>before</b> play PD3 PD2 PD1 PD0	Displays <b>after</b> play PD3 PD2 PD1 PD0	D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
8	F F F F	F F F (8)	D	0	8	N
6	F 5 5 5	(5) 5 5 5	A	3	40	Y
4	6 A 6 7	6 A (A) 7	A	1	187	Y
7	0 0 0 0	0 0 (7) 0	D	0	7	N
3	1 E 8 E	1 (1) 8 E	A	1	2	Y

The strategy also does not check for code digits and so again misses to earn code reward points when the random digit is 8. It also misses to earn code reward points when the random digit is 7. However, it again earns code reward points by coincidence when the random digit is 4.

ii) We know that the Rightmost Largest Adjacency circuit keeps the largest adjacency when it determines which position has it. The largest adjacency lines, LRGADJ1 and LRGADJ0 are used to determine if the adjacency is 0 :

