

HOMEWORK VI

DUE : December 8, 2011

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

ASSIGNMENT : There are nine questions.

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

1) Solve Problem 4.14 (d). Draw by hand the corresponding 2-level AND-OR gate network, assuming that single-rail inputs are available. Determine the TTL LS **SSI chip usage** for the case of developing a new **PCB**. That is, which TTL LS SSI chips are used, the number of chips used for each kind and the number of unused SSI gates for this 2-level AND-OR gate network.

You will point out the distinguished 1-cell(s). Do that by giving their minterm numbers as done in class. Remember also to state whether a term is an **essential prime implicant** or a **secondary essential prime implicant** or a **prime implicant**.

2) By using the minterm list in Problem 4.14 (d), obtain the minimal **product-of-sums** expression. Draw by hand the corresponding 2-level OR-AND gate network, assuming that single-rail inputs are available. Determine the TTL LS **SSI chip usage** for the case of developing a new **PCB**.

You will point out the distinguished 0-cell(s). Do that by giving their maxterm numbers. Remember also to state whether a term is an **essential prime implicant** or a **secondary essential prime implicant** or a **prime implicant**.

3) Solve Problem 4.18 (d). Draw by hand the minimal 2-level NAND-NAND gate network, assuming that single-rail inputs are available. Determine the TTL LS **SSI chip usage** for the case of developing a new **PCB**. You will **not** use the method given in Section 4.3.6 in the textbook.

You will point out the distinguished 1-cell(s). Do that by giving their minterm numbers. Note that all the don't cares need **not** be covered. Remember also to state whether a term is an **essential prime implicant** or a **secondary essential prime implicant** or a **prime implicant**.

4) Consider the following minimal SOP expression : $f(a, b, c, d) = bd + \bar{b}\bar{d}$

i) Draw the corresponding 2-level AND-OR gate network, assuming that single-rail inputs are available. Determine the TTL LS SSI chip usage for the case of developing a new PCB. That is, which TTL LS SSI chips are used, the number of chips used for each kind and the number of unused SSI gates.

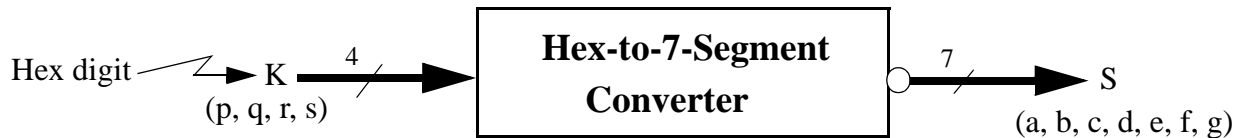
ii) Draw the corresponding 2-level NAND-NAND gate network, assuming that single-rail inputs are available. Determine the TTL LS SSI chip usage for the case of developing a new PCB.

iii) The 2-level minimal AND-OR and NAND-NAND circuits require more than one chip each. However, a close analysis of the minimal circuit indicates that only **one TTL SSI chip** is enough to implement the minimal circuit. Draw that circuit by hand. To determine which chip it is, you can check TTL manuals, such as the On Semiconductor manual or the Motorola manual.

5) Consider Problem 1 of Homework 3 that analyzes a sequential circuit.

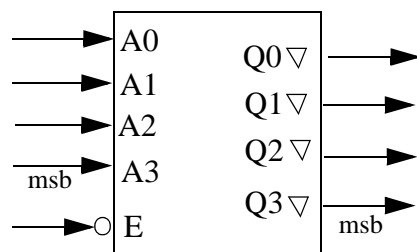
Assuming that single-rail inputs are available, determine the TTL LS SSI chip usage for the case of developing a new PCB.

6) Consider the following combinational circuit that converts a Hexadecimal digit to the 7-Segment format :



The converter outputs (S) are **active-low**.

Implement the circuit by using **generic** 16x4-bit ROM chips whose black-box view is as follows :



Note that this is what is implemented in **Block 2** of the Term Project (**ppm2.SCH**), except that the Xilinx ROMs do **not** have Enable inputs and have only one bit per location.

Write down the minterm lists of the outputs, by using the *Advanced Xilinx and Digilent Features* handout. Draw the circuit containing ROM chips and clearly indicate what is connected to the inputs and outputs of the ROM chips. Then, show the content of each ROM in terms of bits as done in class. Finally, write down the number of unused ROM bits.

7) Solve Problem 9.1 of Chapter IX, by working on figure **6-37** only (**not** Exercise 6.31 **nor** figures 6.73, 6.93, X6.44). **In addition**, apply Problem 9.1 on the three minterm lists given on page 222 of the Wakerly book. The circuit has three inputs (X, Y, Z) and three outputs (F, G, H).

Overall, Problem 9.1 wants you to assume that there is a *generic matching* ROM chip (**not** a commercial chip **nor** a Xilinx ROM) that exactly fits each implementation. Therefore, the number of unused bits is 0.

For this problem, there are **two** circuits to work on and for **each** circuit :

- i) Draw the black box view of the circuit showing its inputs and outputs,
- ii) State the number of ROM address inputs, the number of ROM data outputs, the ROM size (specified as k x m-bit) and the total number of bits the ROM has and finally,
- iii) For the circuit on page 222, show how the ROM is programmed, i.e. show the ROM content as discussed in class.

8) Consider the 10-to-4 Encoder designed in Question 2 of Homework V where you obtain the minimal expressions for all five (5) outputs of the encoder. Now, implement this encoder by using a single **PLS 100 PLA** chip whose description handout is given in class. **No** external (off-the-chip) connections nor external gates (no other chip) can be used. In order to help grade your answer, write down the five expressions you implement.

9) Implement **two** independent Full-Adder circuits (not connected to each other) by using a single Monolithic Memories **12H6 PAL** chip whose description handout is given in class. You may use off-the-chip (external) connections, but **no** external gate (**no** other chip) can be used.

RELEVANT QUESTIONS AND ANSWERS

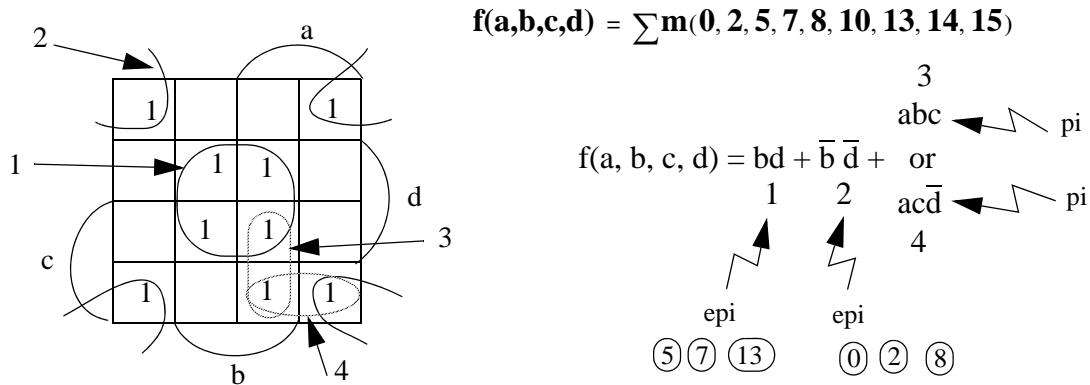
Q1) Consider the following minterm list :

$$f(a,b,c,d) = \sum m(0, 2, 5, 7, 8, 10, 13, 14, 15)$$

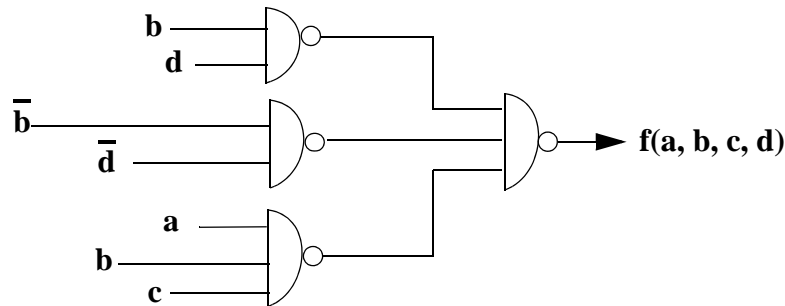
Obtain the minimal 2-level NAND-NAND circuit by using the Karnaugh-map method.

State whether a term is an essential prime implicant or a secondary essential prime implicant or a prime implicant. Assume that double-rail inputs are available.

A1) The Karnaugh map of this function is :



Either circuit is minimal. We choose the one with combinations 1, 2 and 3 : A two-level NAND-NAND circuit is directly obtained from a two-level AND-OR network which is in turn directly obtained from a 2-level SOP expression. Note that **double-rail** inputs eliminate the need for extra NAND gates for inversion.



Q2) Consider the following minterm list for a function :

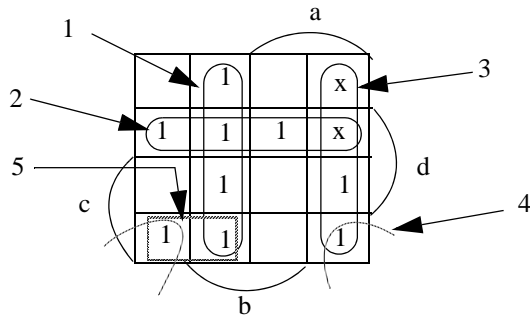
$$f(a,b,c,d) = \sum m(1,2,4,5,6,7,10,11,13) + d(8,9)$$

a) By using the Karnaugh-Map method, obtain the minimal SOP expression as shown in class. State whether a term is an essential prime implicant or a secondary essential prime implicant or a prime implicant.

b) Determine the minimal implementation of the expression obtained in part (a) by using only TTL LS NAND-gate chips, as shown in class. Assume that only single-rail inputs are available.

A2) a)

$$f(a,b,c,d) = \sum m(1,2,4,5,6,7,10,11,13) + d(8,9)$$

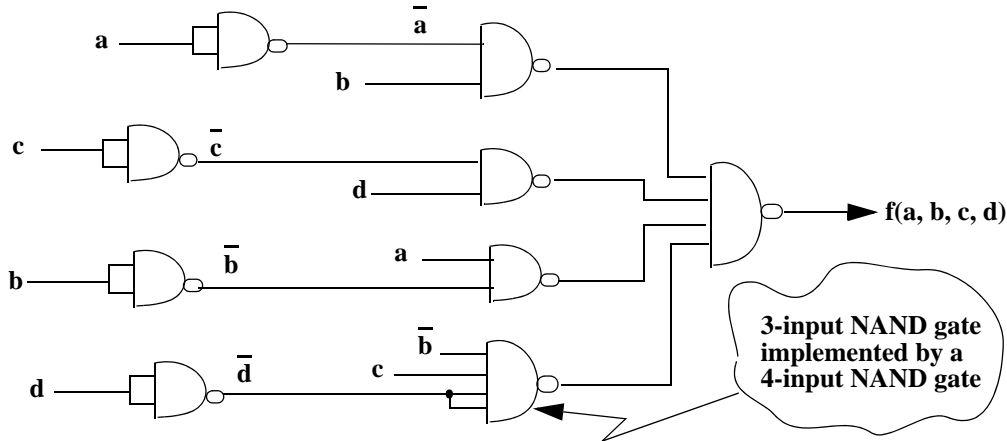


$$f(a, b, c, d) = \bar{a} b + \bar{c} d + a \bar{b} + \bar{a} c \bar{d} \text{ or } \bar{b} c \bar{d}$$

1 2 3 4
 epi epi epi pi
 (4) (7) (1) (13) (11)

Combinations 1, 2 and 3 are essential prime implicants, covering all the minterms except minterm 2. Combinations 4 and 5 are prime implicants. To cover minterm 2, we have two choices either 4 or 5.

b) We choose combinations 1, 2, 3 and 4 to implement. We know that any 2-level SOP expression can be directly converted to a 2-level NAND/NAND network :



2 74LS00, each with 4 2-input NAND-gates, one NAND gate **unused**
 1 74LS20, with 2 4-input NAND-gates. Zero **unused**
 3 chips used. One gate **unused**.

Q3) Consider the following switching function whose minterm list is given below :

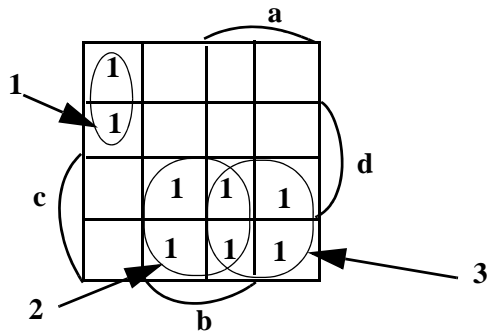
$$f(a,b,c,d) = \sum m(0, 1, 6, 7, 10, 11, 14, 15)$$

i) Obtain the minimal SOP expression by using the **K-map** method as done **in class**.

ii) Based on part (i), draw the 2-level AND-OR gate network, assuming **single-rail** inputs. Then, determine the TTL LS **SSI** chip usage of this 2-level AND-OR gate network as done **in class**.

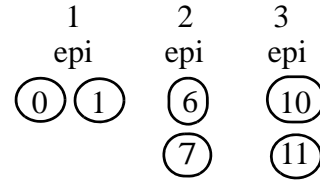
A3)

i)

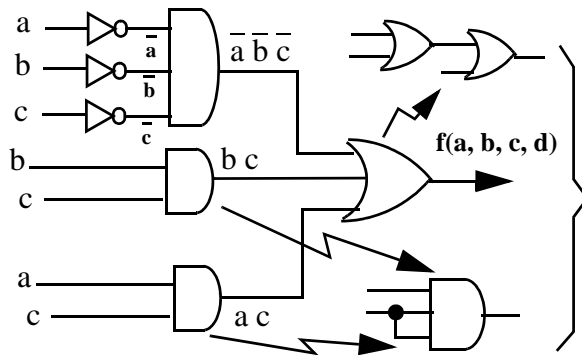


$$f(a,b,c,d) = \sum m(0, 1, 6, 7, 10, 11, 14, 15)$$

$$f(a, b, c, d) = \bar{a} \bar{b} \bar{c} + b c + a c$$



ii) The 2-level AND-OR gate network with single-rail inputs and the TTL LS SSI chip usage :

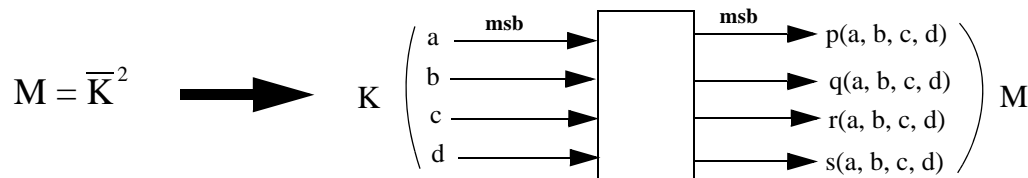


We need
 3 inverters
 2 2-input AND gates
 1 3-input AND gate
 1 3-input OR gate

1 74LS04 w/ 6 inverters, 3 inverters **unused**
 1 74LS11 w/ 3 3-input AND gates, 0 gates **unused**
 1 74LS32 w/ 4 2-input OR gates, 2 gates **unused**
 3 chips used. 5 gates **unused**

Q4) Design a digital circuit with four inputs, a, b, c and d. There are four outputs, p, q, r and s.

Inputs (a, b, c, d) represent a 4-bit 2's complement number, K, where "a" is the most significant bit. Outputs, (p, q, r, s) represent a 4-bit 2's complement number, M, where "p" is the most significant bit. The relationship between K and M is that the combinational circuit takes the 2's complement of input K as shown below:



Design the circuit by showing the following steps :

- the truth table of the four output functions,
- the minterm lists of the four output functions,
- the Karnaugh-map for only output function **r(a, b, c, d)**,
- the minimal 2-level SOP expression for only output function **r(a, b, c, d)**.

In the **Karnaugh-map** step, specify distinguished-1 cells of output function **r(a, b, c, d)**. Also, specify for each term of the minimal expression, if it is an epi or an sepi or a pi.

A4) The truth table of the four functions based on the given input-output relationship is below.

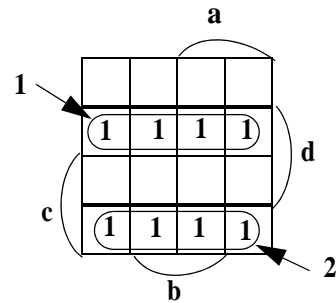
a b c d	p	q	r	s
0 0 0 0	0	0	0	0
0 0 0 1	1	1	1	1
0 0 1 0	1	1	1	0
0 0 1 1	1	1	0	1
0 1 0 0	1	1	0	0
0 1 0 1	1	0	1	1
0 1 1 0	1	0	1	0
0 1 1 1	1	0	0	1
1 0 0 0	1	0	0	0
1 0 0 1	0	1	1	1
1 0 1 0	0	1	1	0
1 0 1 1	0	1	0	1
1 1 0 0	0	1	0	0
1 1 0 1	0	0	1	1
1 1 1 0	0	0	1	0
1 1 1 1	0	0	0	1

$$p(a, b, c, d) = \sum m(1, 2, 3, 4, 5, 6, 7, 8)$$

$$q(a, b, c, d) = \sum m(1, 2, 3, 4, 9, 10, 11, 12)$$

$$r(a, b, c, d) = \sum m(1, 2, 5, 6, 9, 10, 13, 14)$$

$$s(a, b, c, d) = \sum m(1, 3, 5, 7, 9, 11, 13, 15)$$

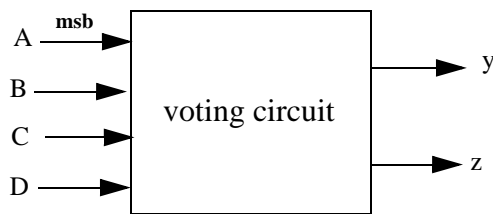


$$r(a, b, c, d) = \bar{c}d + c\bar{d}$$

$\begin{matrix} 1 & 2 \\ \text{epi} & \text{epi} \end{matrix}$

(1) (5) (9) (13) (2) (6) (10) (14)

Q5) Design a digital “voting” circuit with four inputs, A, B, C and D. Each input is assigned a “vote weight” as follows :



Input	Vote Weight
A	3
B	4
C	2
D	3

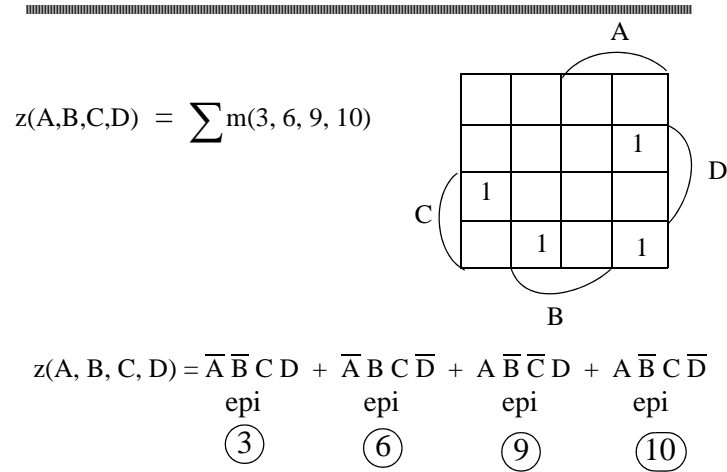
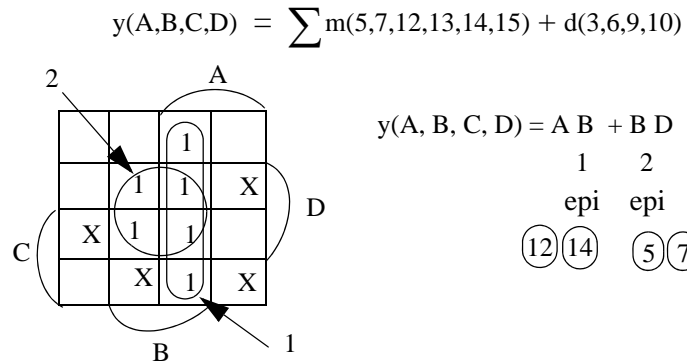
A decision is approved when the “y” output is **1** and the z output is **0**, which happens if

- i)** the sum of vote weights is 9 or more, OR,
- ii)** the sum is 7 or 8 and member B has voted for it, i.e. input B is 1.

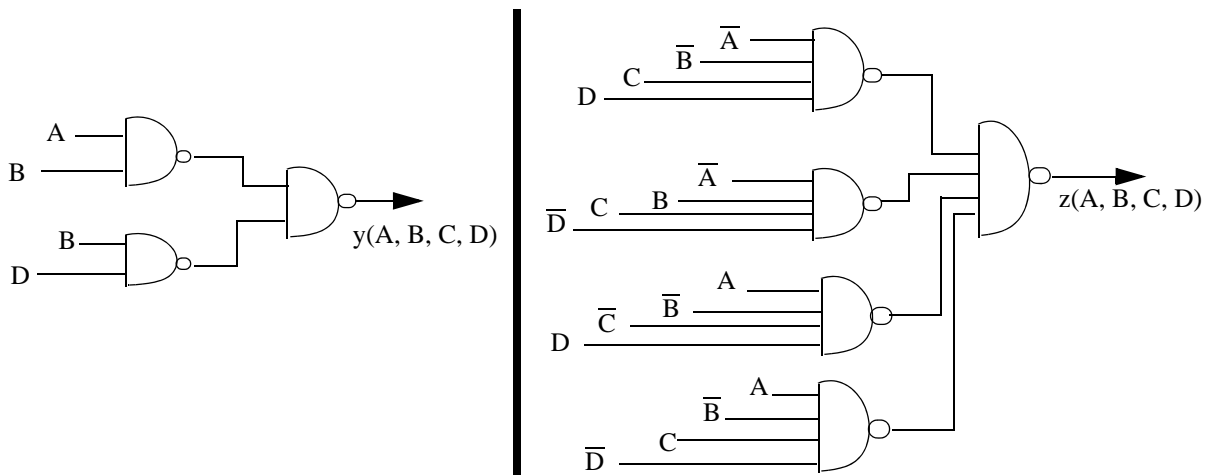
A new vote is asked for if the sum is 5 or 6 during which the “y” output is **don’t care** and the “z” output is **1**. Implement the circuit by using a **minimum** number of TTL LS SSI chips. Indicate the TTL LS SSI chip usage. Note that double-rail inputs are available.

A5) All possible sums of weights and corresponding “y” and “z” outputs are listed below :

ABCD	Vote sum	y	z
0000	0	0	0
0001	3	0	0
0010	2	0	0
0011	5	X	1
0100	4	0	0
0101	7	1	0
0110	6	X	1
0111	9	1	0
1000	3	0	0
1001	6	X	1
1010	5	X	1
1011	8	0	0
1100	7	1	0
1101	10	1	0
1110	9	1	0
1111	12	1	0



The circuit for the “y” output is a 2-level SOP circuit that can be directly implemented by a 2-level NAND- NAND network. It is the same for the “z” output :



We use the following chips :

- 3 74LS20 with 2 4-input NAND-gate chips, one 4-input NAND gate **unused**
- 1 74LS00 with 4 2-input NAND-gate chip, one 2-input NAND gate **unused**
- 4 chips used. Two gates **unused**.

Q6) Consider the following the maxterm list :

$$f(a,b,c,d) = \prod M(0,2,3,4,6,7,8,12,15)$$

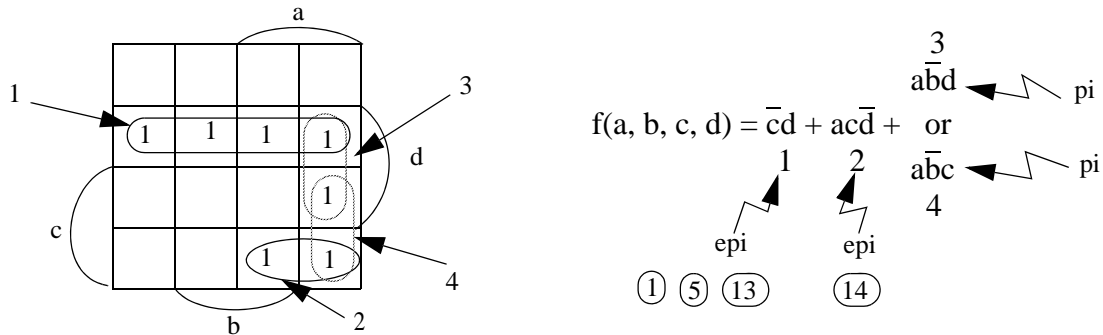
i) Obtain the minimal SOP expression by using the **K-map** method as done **in class**. You will point out the distinguished-1 cell(s). Do that by giving their **minterm numbers**. Remember also to state whether a term is an **essential prime implicant** or a **secondary essential prime implicant** or a **prime implicant**.

ii) Based on part (i), draw the 2-level AND-OR gate network, assuming **single-rail** inputs. Then, determine the TTL LS SSI chip usage of this 2-level AND-OR gate network as done **in class**.

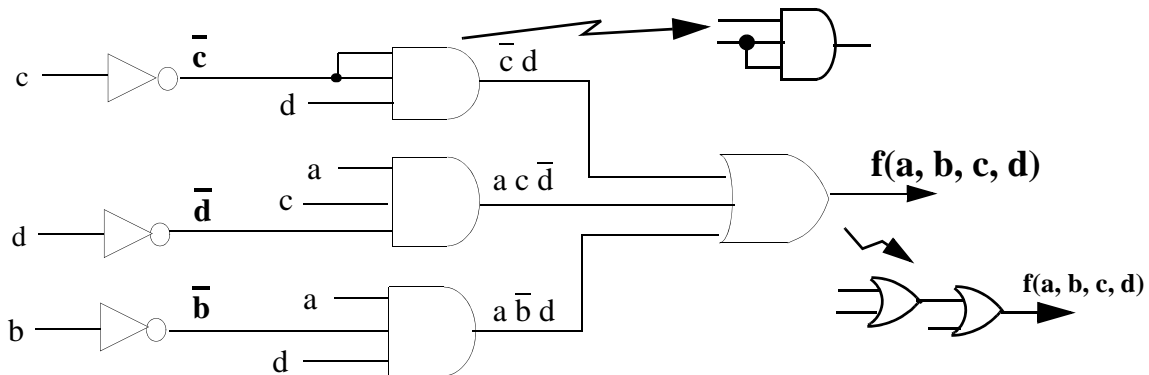
A6) We need to get the minterm list from the maxterm list for the minimal 2-level AND-OR gate network. One can immediately obtain the minterm list from a maxterm list :

$$f(a,b,c,d) = \prod M(0,2,3,4,6,7,8,12,15) = \sum m(1,5,9,10,11,13,14)$$

Then, the Karnaugh map of this function is :



Either circuit is minimal. We choose the one with combinations 1, 2 and 3 :



1 74LS04 w/ 6 inverters, 3 inverters **unused**
 1 74LS11 w/ 3 3-input AND gates, 0 gates **unused**
 1 74LS32 w/ 4 2-input OR gates, 2 gates **unused**
3 chips used. 5 gates **unused**

Q7) A switching function has been simplified and the following minimal expression is obtained :

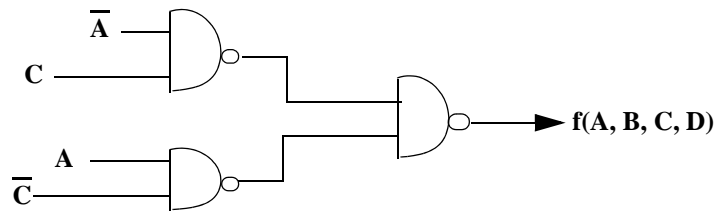
$$f(A, B, C, D) = \bar{A}C + A\bar{C}$$

Implement the minimal expression, by using as few TTL LS SSI chips as possible as shown in class. Assume that double-rail inputs are available.

A7) $f(A, B, C, D) = \bar{A}C + A\bar{C}$

There are three solutions each of which uses only **one** (1) chip :

i) The minimal SOP expression results in a 2-level AND/OR gate network. We know that any 2-level AND/OR gate network can be directly converted to a 2-level NAND/NAND network. Thus, we have the following 2-level NAND/NAND gate network :



We need three 2-input NAND gates !!!

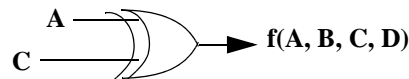
1 74LS00, with 4 2-input NAND-gates, 1 NAND gate unused

 1 chip used. 1 gate unused.

ii) We see that the circuit outputs a 1, if the two inputs (a and b) are different. Thus, this is the EX-OR function :

$$f(A, B, C, D) = \bar{A}C + A\bar{C} = A \oplus C$$

We need one 2-input XOR gate :



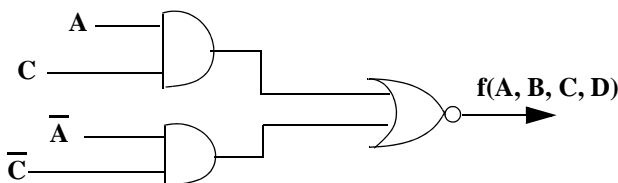
1 74LS86, with 4 2-input EX-OR-gates, 3 EX-OR gates unused

 1 chip used. 3 gates unused.

iii) We realize by using DeMorgan's theorems that

$$f(A, B, C, D) = \bar{A}C + A\bar{C} = \overline{\overline{\bar{A}C} + \overline{A\bar{C}}} = \overline{(\overline{\bar{A}C})(\overline{A\bar{C}})} = \overline{(A + \bar{C})(\bar{A} + C)} = \overline{AC + \bar{A}\bar{C}}$$

We realize that this is an AOI function implemented by the **74LS51 dual-AOI chip** :

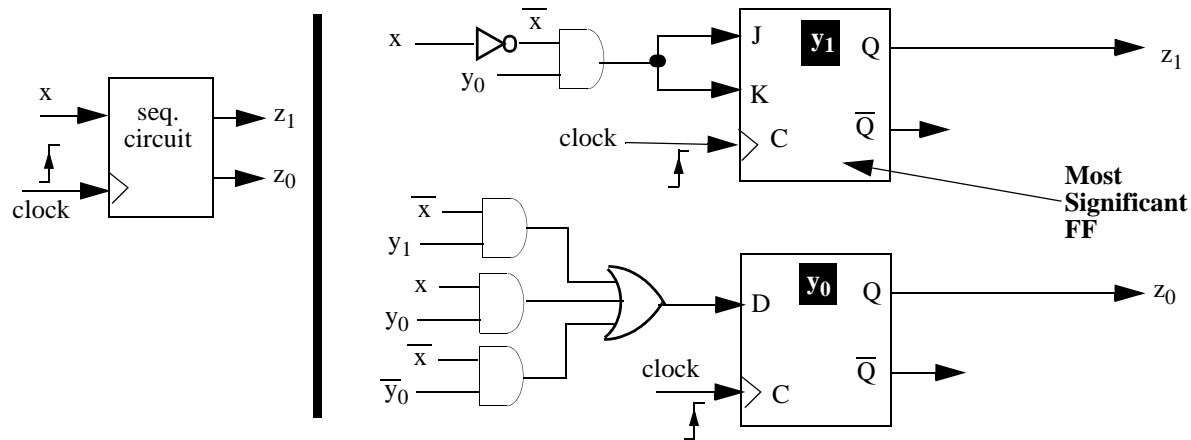


1 74LS51, with 2 AOI networks, one AOI network unused

 1 chip used. one AOI network unused.

The cheapest solution is the first one....

Q8) Consider the following **sequential** circuit :



Assuming that single-rail inputs are available, determine the TTL LS SSI **chip usage** for the case of developing a new **PCB**.

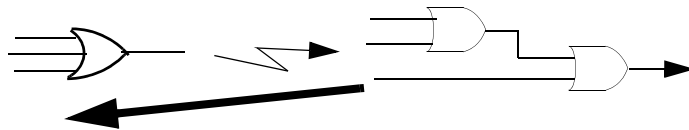
A8)

We need :

- 1 inverter
- 4 2-input AND gates
- 1 3-input OR gate
- 1 positive-edge triggered J-K FF
- 1 positive-edge triggered D FF

We realize that there is **no** positive-edge triggered J-K FF chip. Thus we need an **inverter** to invert the clock : **two** inverters are needed.

In addition there is no 3-input OR gate TTL chip. We implement it by using **two** 2-input OR gates as follows :



Then, we need :

- 2 inverters
- 4 2-input AND gates
- 2 2-input OR gates
- 1 negative-edge triggered J-K FF
- 1 positive-edge triggered D FF

The TTL LS SSI chip usage is then as follows :

- 1 74LS04, with 6 inverters, four inverters **unused**
- 1 74LS08, with 4 2-input AND gates, zero gate **unused**
- 1 74LS32, with 4 2-input OR gates, two gates **unused**
- 1 74LS74, with 2 positive edge-triggered D FFs, one FF **unused**
- 1 74LS112, with 2 negative-edge triggered FFs, one FF **unused**
- 5 chips used. Six gates **unused**, two FFs **unused**

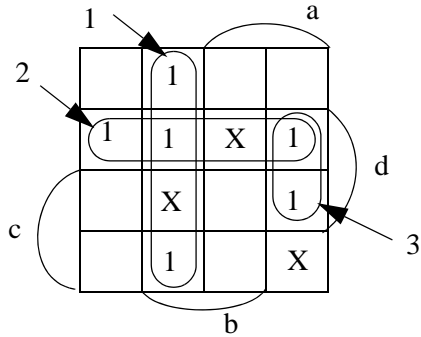
Q9) Consider the following minterm list :

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

a) Obtain the minimal SOP expression of the function by using the **K-map** method as done **in class**.

b) 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. Use no other chips/components.

A9) a) The minimal SOP expression is obtained by using the **K-map** method is as follows :



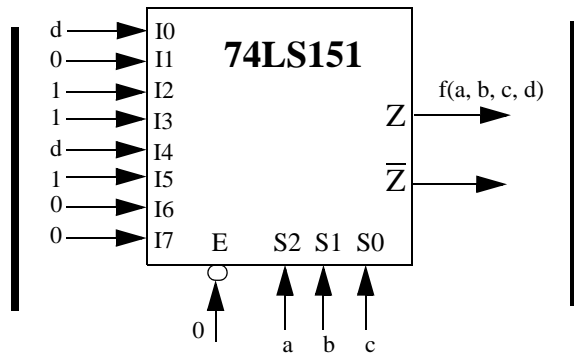
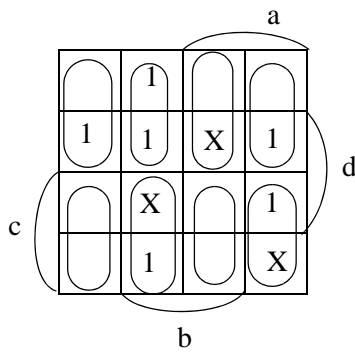
$$f(a, b, c, d) = \sum m(1, 4, 5, 6, 9, 11) + d(7, 10, 13)$$

$$f(a, b, c, d) = \bar{a} b + \bar{c} d + a \bar{b} d$$

1	2	3
epi	epi	epi
④ ⑥	①	⑪

b) The MUX implementation is as follows :

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



Chip Usage :
 1 74LS151 8-to-1 MUX
 Total : 1 chip used

Q10) Consider the following function in the SOP form (**not** necessarily minimized) :

$$F(A, B, C, D) = \bar{A} B C + A D + A C$$

Assuming that only single-rail inputs are available, implement the function using the following approaches and indicate the number of **chips** used :

- A generic 4-to-16 DCD-based
- A generic 8-to-1 MUX-based
- A generic matching ROM-based
- A generic matching PLA-based

All other chips used are **TTL** LS chips.

A10) We are given that $F(A, B, C, D) = \bar{A} B C + A D + A C$

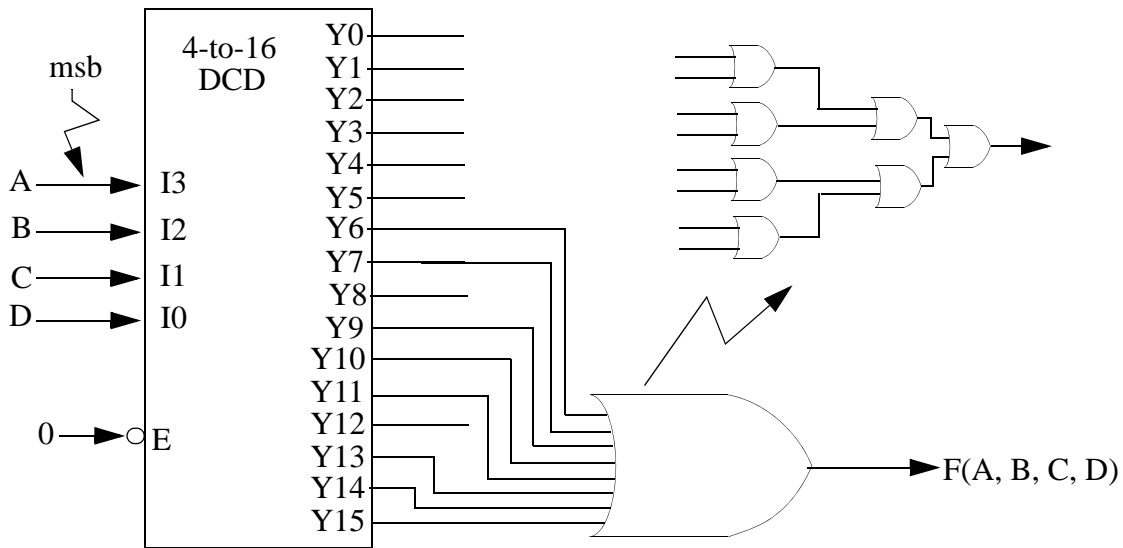
We need to obtain the minterms of the function from the above expression for the first three implementations. The method we have been using for that is by expanding the subexpressions to make them canonical from which it is easy to obtain the minterms :

$$\begin{aligned}
 F(A, B, C, D) &= \bar{A} B C (D + \bar{D}) + A D (B + \bar{B})(C + \bar{C}) + A C (B + \bar{B})(D + \bar{D}) \\
 &= \bar{A} B C D + \bar{A} B C \bar{D} + A B C D + A \bar{B} C D + A B \bar{C} D + A \bar{B} \bar{C} D + A B C D + A \bar{B} C D + A B C \bar{D} + A \bar{B} C \bar{D} \\
 &= \underbrace{\bar{A} B C D}_{0111} + \underbrace{\bar{A} B C \bar{D}}_{0110} + \underbrace{A B C D}_{1111} + \underbrace{A \bar{B} C D}_{1011} + \underbrace{A B \bar{C} D}_{1101} + \underbrace{A \bar{B} \bar{C} D}_{1001} + \underbrace{A B C \bar{D}}_{1110} + \underbrace{A \bar{B} C \bar{D}}_{1010} \\
 &\qquad\qquad\qquad 7 \qquad\qquad\qquad 6 \qquad\qquad\qquad 15 \qquad\qquad\qquad 11 \qquad\qquad\qquad 13 \qquad\qquad\qquad 9 \qquad\qquad\qquad 14 \qquad\qquad\qquad 10
 \end{aligned}$$

$$F(A,B,C,D) = \sum m(6,7,9,10,11,13,14,15)$$

i) A generic 4-to-16 DCD-based :

$$F(A,B,C,D) = \sum m(6,7,9,10,11,13,14,15)$$



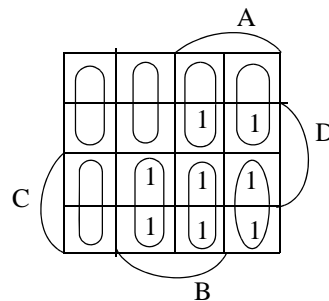
We need :
 1 4-to-16 generic DCD
 7 2-input OR gates

1 4-to-16 generic DCD
 2 74LS32 4 2-input OR gate chip, 1 gate unused

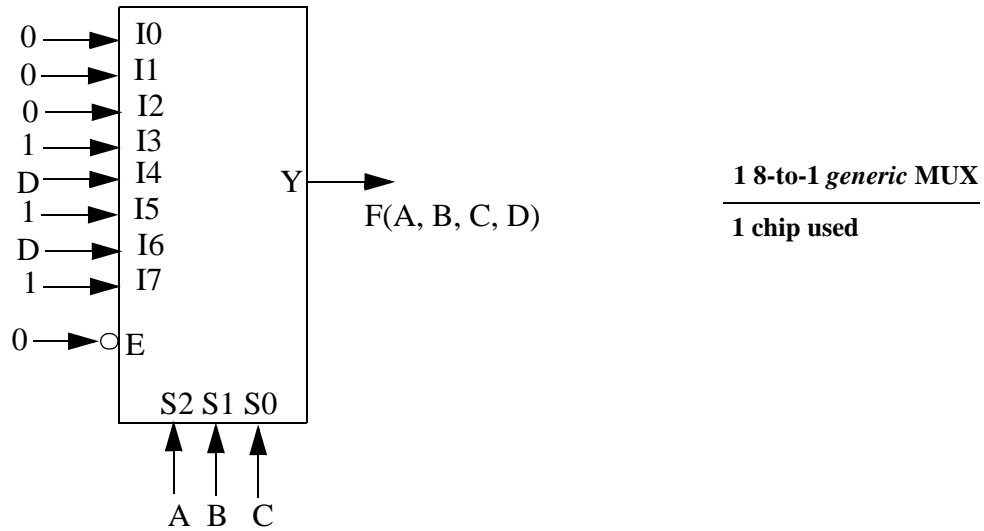
3 chips used, 1 gate unused

ii) A generic 8-to 1 MUX-based :

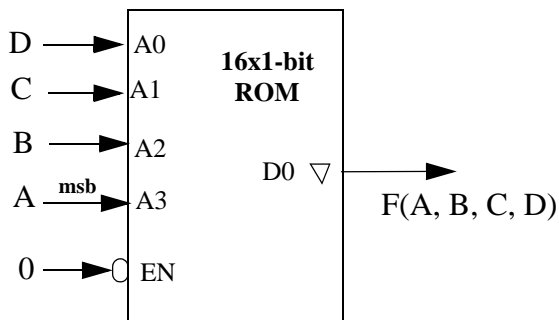
First, we get the K-map of the function :
 If (A,B,C) are used as select signals, the D input can be used without any complement operation :



The MUX circuit implementing the gate network :



iii) A generic matching ROM-based :



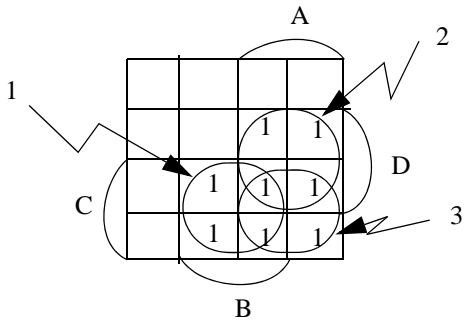
1 16 x 1-bit generic matching ROM

1 chip used

Location	Address				Content
	A A3	B A2	C A1	D A0	F(A, B, C, D) D0
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
A	1	0	1	0	1
B	1	0	1	1	1
C	1	1	0	0	0
D	1	1	0	1	1
E	1	1	1	0	1
F	1	1	1	0	1

iv) A generic matching PLA-based :

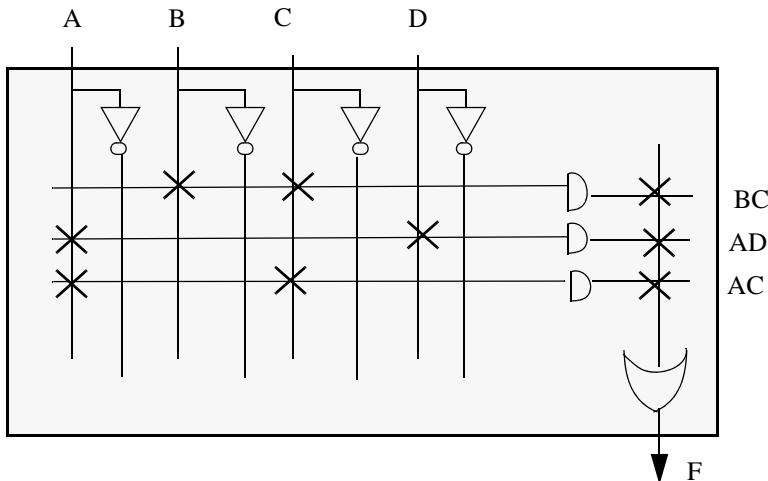
^We first minimize the function to get a minimal expression :



$$F(A, B, C, D) = BC + AD + AC$$

1 2 3
epi epi epi
⑥ ⑦ ⑨ ⑬ ⑩

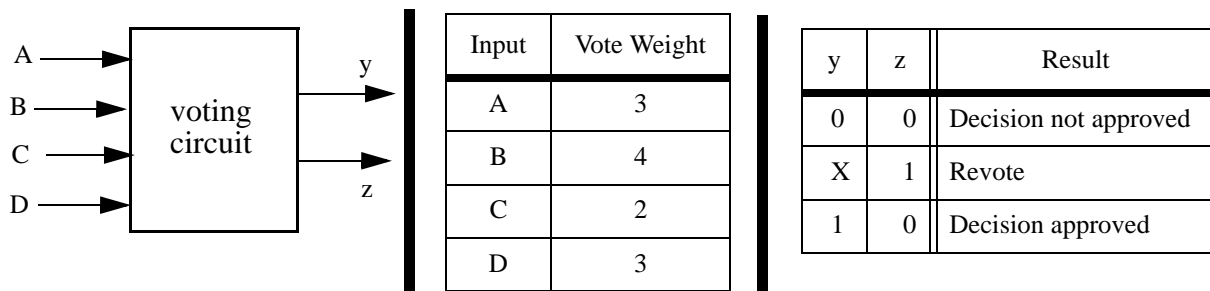
^Then, we program the PLA-based on the minimal SOP expression :



1 generic matching PLA

1 chip used

Q11) A digital “voting” circuit with four inputs, A, B, C and D has been designed, where each input is assigned a “vote weight” as follows :



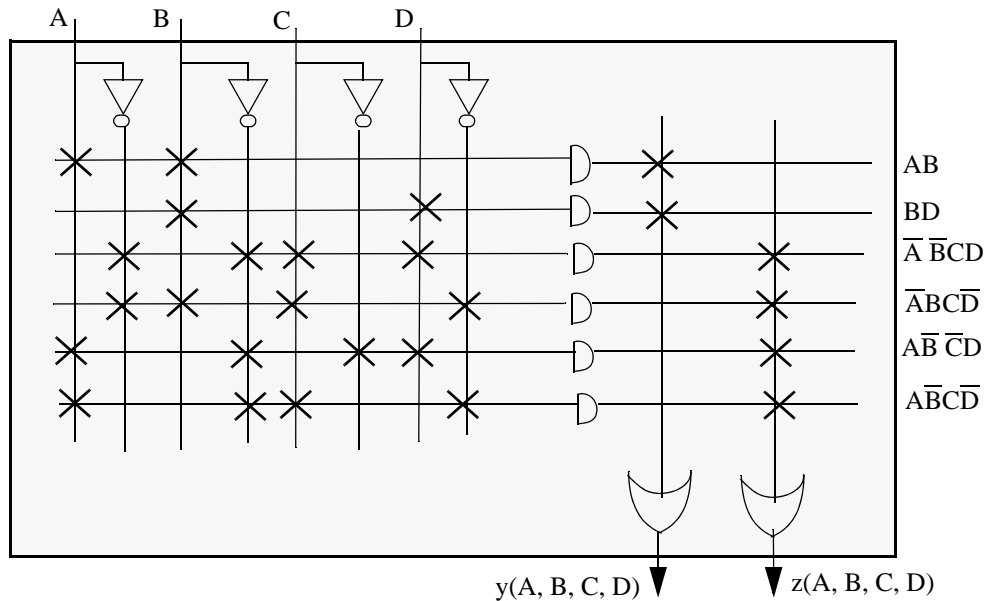
As the operation table of the circuit above shows, a decision is approved, that is the “y” output is 1 and the z output is 0, if **i)** the sum of vote weights is 9 or more, **OR, ii)** the sum is 7 or 8 and member B has voted for it, i.e. input B is 1. A new vote is asked for if the sum is 5 or 6 during which the “y” output is don’t care and the z output is 1.

The minimal expressions have been developed and by using them the TTL LS SSI implementation of the circuit with **double-rail** inputs and only NAND gates has resulted in four chips with **two** gates unused :

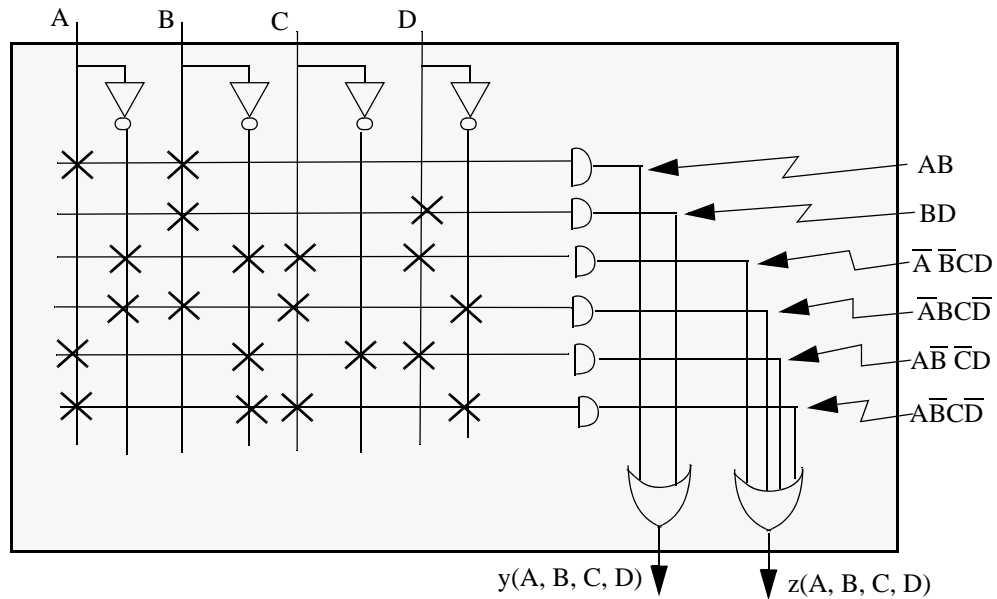
$$y(A, B, C, D) = A B + B D \quad \Bigg| \quad z(A, B, C, D) = \bar{A} \bar{B} C D + \bar{A} B C \bar{D} + A \bar{B} \bar{C} D + A \bar{B} C \bar{D}$$

For this problem, show the implementation with a *generic matching* PLA chip and then a *generic matching* PAL chip. Finally, indicate which chip is more appropriate for implementation.

A11) The circuit with a *generic matching* PLA chip is as follows :

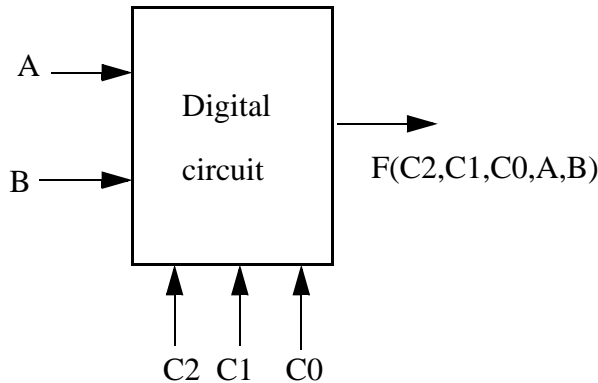


The circuit with a *generic matching* PAL chip is as follows :



The product terms of the two expressions are **not** the same : There is **no** sharing among the functions. Thus, a **PAL** is more appropriate for implementation.

Q12) By using a *generic matching* PLA, implement a digital circuit with five inputs and one output. Inputs A and B are data inputs and C2, C1 and C0 are control inputs. The output is named F. The digital circuit performs eight different operations based on the control signals :



Control signals C2 C1 C0	Function F
0 0 0	1
0 0 1	A v B (Or)
0 1 0	$\overline{A B}$ (Nand)
0 1 1	A xor B (Xor)
1 0 0	A xnor B (Xnor)
1 0 1	A B (And)
1 1 0	$\overline{A + B}$ (Nor)
1 1 1	0

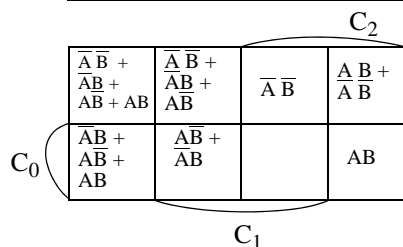
This digital circuit is a 1-bit Logic Unit (LU) as it performs only logical operations and has only a 1-bit output. If it performed also arithmetic operations, it would be called Arithmetic Logic Unit (ALU). **Every computer has an ALU which is typically a 32-bit or a 64-bit ALU.**

A12) The *generic matching* PLA has 5 inputs : C2, C1, C0, A and B. The operation table is as follows :

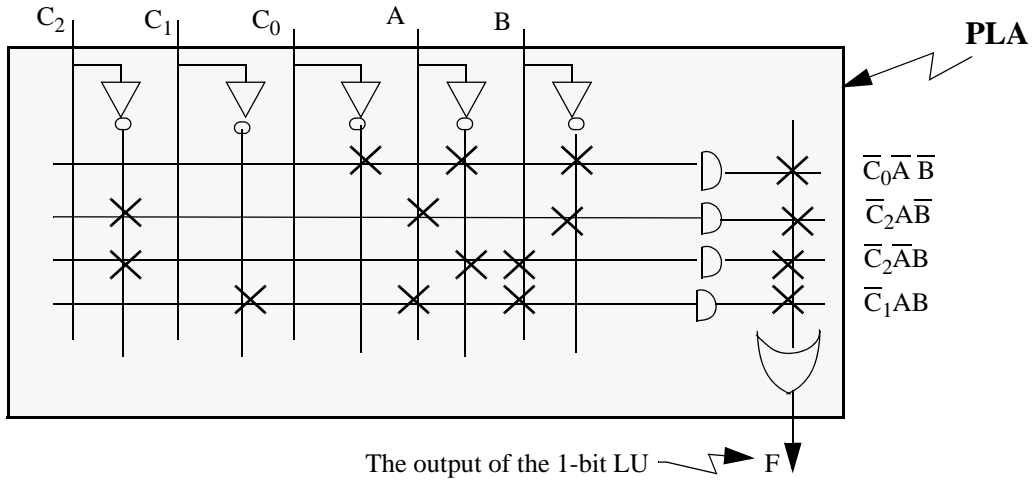
C2	C1	C0	F
0	0	0	1
0	0	1	A + B
0	1	0	$\overline{A B}$
0	1	1	A XOR B
1	0	0	A XNOR B
1	0	1	A.B
1	1	0	$\overline{A + B}$
1	1	1	0

F is 1 if

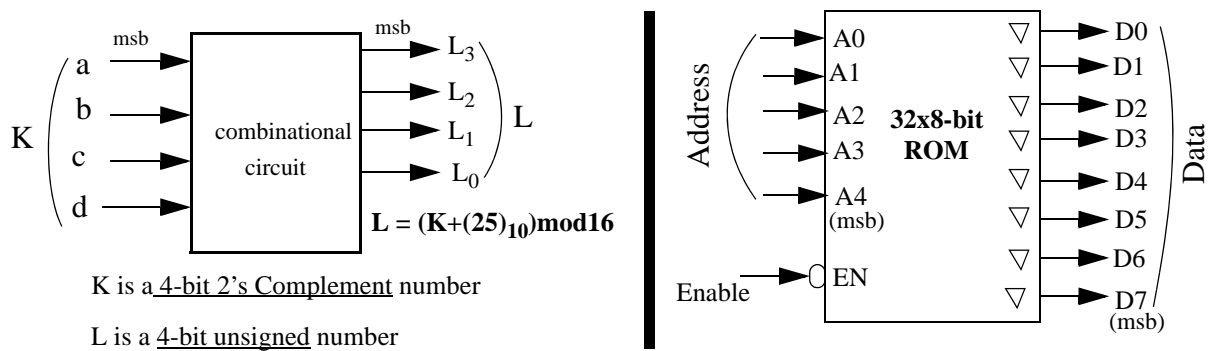
$$\begin{aligned} \rightarrow & \overline{C_0} \overline{C_1} \overline{C_2} 1 = \overline{C_0} \overline{C_1} \overline{C_2} (\overline{A B} + \overline{A} B + A \overline{B} + A B) \\ \rightarrow & C_0 \overline{C_1} \overline{C_2} (A+B) = C_0 \overline{C_1} \overline{C_2} (\overline{A B} + \overline{A B} + A B) \\ \rightarrow & \overline{C_0} C_1 \overline{C_2} (\overline{A} + \overline{B}) = \overline{C_0} C_1 \overline{C_2} (\overline{A B} + \overline{A B} + A \overline{B}) \\ \rightarrow & C_0 C_1 \overline{C_2} (A \overline{B} + \overline{A} B) \\ \rightarrow & \overline{C_0} \overline{C_1} C_2 (A B + \overline{A} \overline{B}) \\ \rightarrow & C_0 \overline{C_1} C_2 (A B) \\ \rightarrow & \overline{C_0} C_1 C_2 (\overline{A B}) \end{aligned}$$



$$F(C_2, C_1, C_0, A, B) = \overline{C_0} \overline{A B} + \overline{C_2} \overline{A B} + \overline{C_2} \overline{A B} + \overline{C_1} A B$$



Q13) Design a combinational circuit that has **four** inputs and **four** outputs :



The combinational circuit computes $(K + (25)_{10}) \bmod 16$. For example, if K is $(-6)_{10}$, the L output of the combinational circuit is $((-6)_{10} + (25)_{10}) \bmod 16 = (19)_{10} \bmod 16 = (3)_{10}$.

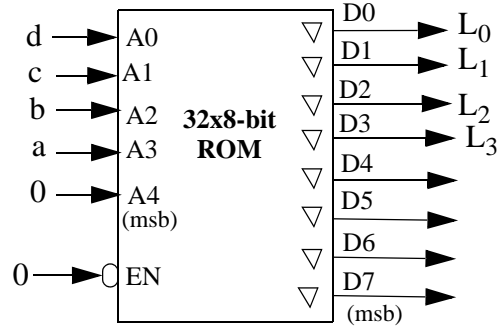
A ROM-based implementation is decided for the combinational circuit where a *generic* 32x8-bit ROM as shown above is used. No other chip or gate can be used.

Draw the circuit. Show how the ROM is programmed as discussed in class. Indicate how many ROM content bits are unused. Note again, your circuit must contain only one chip.

A13) The truth table of the combinational circuit and ROM connections are shown below :

a b c d	L ₃	L ₂	L ₁	L ₀
0000	1	0	0	1
0001	1	0	1	0
0010	1	0	1	1
0011	1	1	0	0
0100	1	1	0	1
0101	1	1	1	0
0110	1	1	1	1
0111	0	0	0	0
1000	0	0	0	1
1001	0	0	1	0
1010	0	0	1	1
1011	0	1	0	0
1100	0	1	0	1
1101	0	1	1	0
1110	0	1	1	1
1111	1	0	0	0

← (0 + 25) mod 16 = 25 mod 16 = (9)₁₀ = (1001)₂ unsigned



← (-1 + 25) mod 16 = 24 mod 16 = (8)₁₀ = (1000)₂ unsigned

The ROM content is as follows :

Loc	Address					Data							
	0 A4	a A3	b A2	c A1	d A0	D7	D6	D5	D4	L ₃ D3	L ₂ D2	L ₁ D1	L ₀ D0
0	0	0	0	0	0					1	0	0	1
1	0	0	0	0	1					1	0	1	0
2	0	0	0	1	0					1	0	1	1
3	0	0	0	1	1					1	1	0	0
4	0	0	1	0	0					1	1	0	1
5	0	0	1	0	1					1	1	1	0
6	0	0	1	1	0					1	1	1	1
7	0	0	1	1	1					0	0	0	0
8	0	1	0	0	0					0	0	0	1
9	0	1	0	0	1					0	0	1	0
A	0	1	0	1	0					0	0	1	1
B	0	1	0	1	1					0	1	0	0
C	0	1	1	0	0					0	1	0	1
D	0	1	1	0	1					0	1	1	0
E	0	1	1	1	0					0	1	1	1
F	0	1	1	1	1					1	0	0	0
10													
...													
1F													

Number of unused bits :

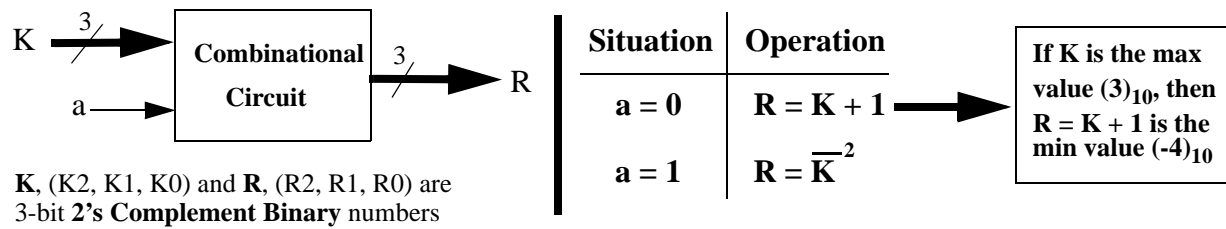
$$(16 \cdot 4) + (16 \cdot 8) =$$

$$64 + 128 = 192 \text{ bits}$$

The ROM has 32 locations but we do **not** use “bottom” 16 locations : locations 10 through 1F. Therefore, we do not use $16 \cdot 8 = 128$ bits. In addition, in the 16 locations used (locations 0 through F), most significant (leftmost) four bits are **not** used.

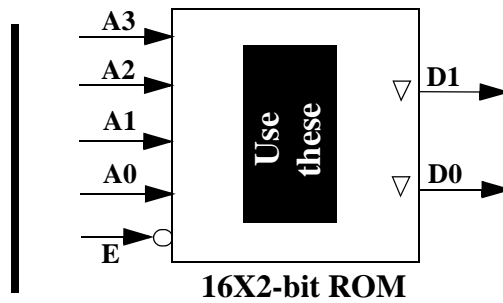
Thus, we do not use $16 \cdot 4 = 64$ bits. Overall then, 192 bits are unused.

Q14) Consider the following 4-input, 3-output combinational circuit and its operation table :



Implement the above circuit by using 16x2-bit ROM chips shown on the right side and as done in class. That is, show the following implementation steps :

- i) The truth table (a is the most significant input)
- ii) The ROM circuit,
- iii) The ROM content, and
- iv) The number of unused bits.



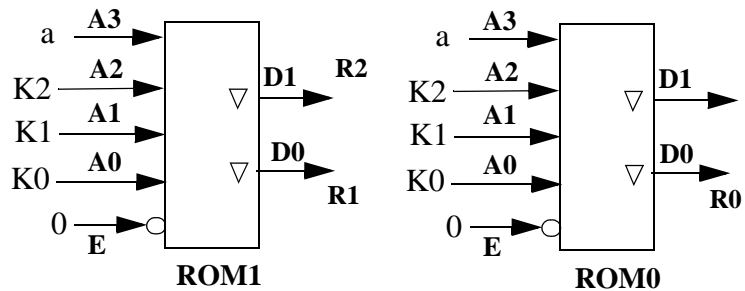
Assume that only **single-rail** inputs are available.

A14)

i) The truth table :

a	K2	K1	K0	R2	R1	R0
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	1	1	0
0	1	1	0	1	1	1
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	0	1	1	1	1
1	0	1	0	1	1	0
1	0	1	1	1	0	1
1	1	0	0	1	0	0
1	1	0	1	0	1	1
1	1	1	0	0	1	0
1	1	1	1	0	0	1

ii) The ROM circuit



iii) The ROM tables

a	K2	K1	K0	R2	R1
A3	A2	A1	A0	D1	D0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	1	0
0	1	0	0	1	0
0	1	0	1	1	1
0	1	1	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	1	0
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	0	0

ROM1 Table

a	K2	K1	K0	R0	Y0
A3	A2	A1	A0	D1	D0
0	0	0	0		1
0	0	0	1		0
0	0	1	0		1
0	0	1	1		0
0	1	0	0		1
0	1	0	1		0
0	1	1	0		1
0	1	1	1		0
1	0	0	0		0
1	0	0	1		1
1	0	1	0		0
1	0	1	1		1
1	1	0	0		0
1	1	0	1		1
1	1	1	0		0
1	1	1	1		1

ROM0 Table

iv) The number of unused bits :
 - ROM1 uses all of its bits.
 - ROM0 does not use 16 bits.

Therefore, the number of unused bits is 16

Q15) Consider the following three functions :

$$w(a, b, c, d) = c \bar{d} + \bar{c} d + a b$$

$$y(a, b, c, d) = \bar{a} c + a \bar{c} + \bar{b} d$$

$$z(a, b, c, d) = \bar{a} d + b \bar{c} + a c$$

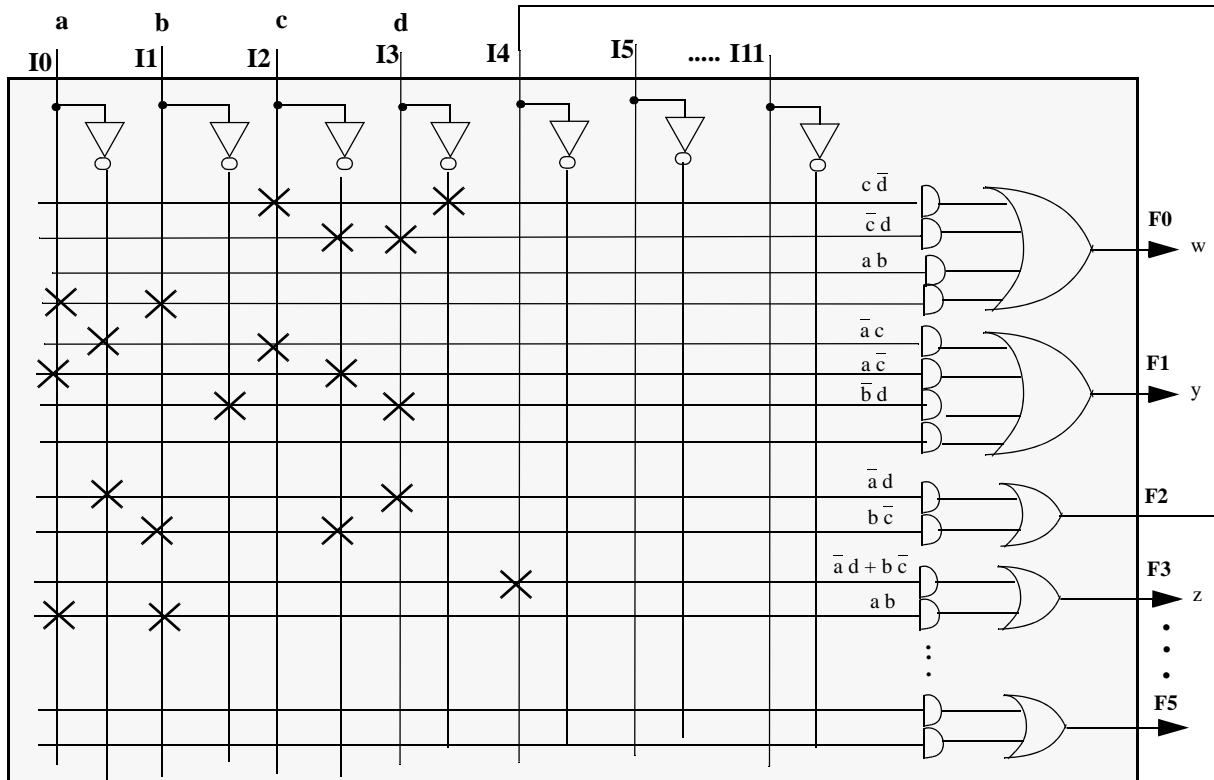
Implement the three functions by using a **single** Monolithic Memories **12H6 PAL chip** whose description handout is given **in class**. You may use off-chip (external) connections, but **no** external gate (no other chip) can be used. Assume that only **single-rail** inputs are available.

A15) We implement the three functions by using a **single** Monolithic Memories **12H6 PAL chip** :

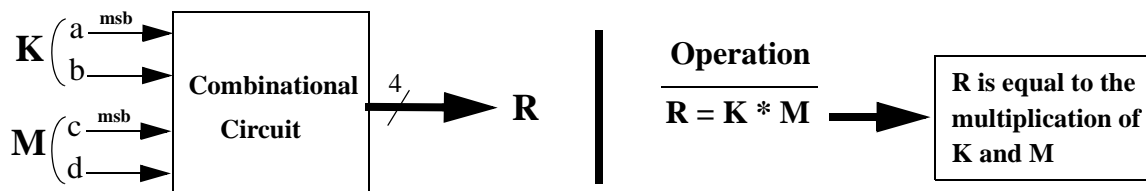
$$w(a, b, c, d) = c \bar{d} + \bar{c} d + a b$$

$$y(a, b, c, d) = \bar{a} c + a \bar{c} + \bar{b} d$$

$$z(a, b, c, d) = \bar{a} d + b \bar{c} + a c$$



Q16) Consider the following 4-input, 4-output **combinational** circuit and its operation table :



K (a, b), **M** (c, d) and **R** (R3, R2, R1, R0) are **Unsigned Binary** numbers

Implement the above circuit by using a **single generic matching** ROM chip and as done **in class**. That is, show the following implementation steps :

- i) The truth table (**a** is the most significant input)
- ii) The size of the generic matching ROM
- iii) The ROM circuit
- iv) The ROM content
- v) The number of unused bits

A16) The implementation is shown below :

i) The truth table :

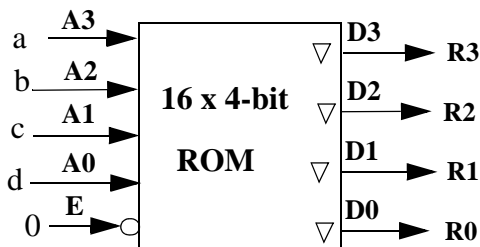
K	M	K		M		R				R
		a	b	c	d	R3	R2	R1	R0	
0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	2	0	0	1	0	0	0	0	0	0
0	3	0	0	1	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	1	1
1	2	0	1	1	0	0	0	1	0	2
1	3	0	1	1	1	0	0	1	1	3
2	0	1	0	0	0	0	0	0	0	0
2	1	1	0	0	1	0	0	1	0	2
2	2	1	0	1	0	0	1	0	0	4
2	3	1	0	1	1	0	1	1	0	6
3	0	1	1	0	0	0	0	0	0	0
3	1	1	1	0	1	0	0	1	1	3
3	2	1	1	1	0	0	1	1	0	6
3	3	1	1	1	1	1	0	0	1	9

iii) The ROM table

a	b	c	d	R3	R2	R1	R0
A3	A2	A1	A0	D3	D2	D1	D0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

ii) The ROM size and circuit

The matching ROM size is 16 x 4-bit since the circuit has 4 inputs and 4 outputs

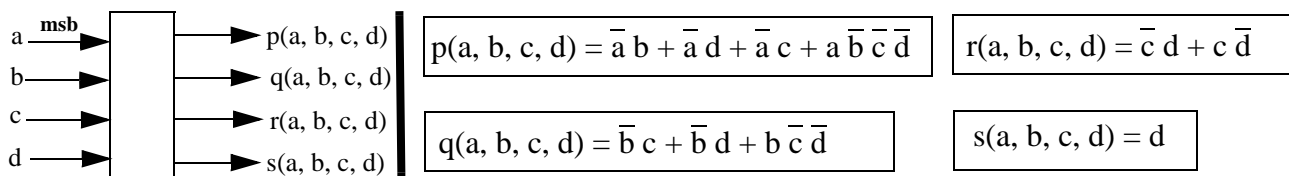


iv) The number of unused bits :

The ROM uses all of its bits, since it is a matching ROM

Therefore, the number of unused bits is 0

Q17) Design a digital circuit with four inputs, a, b, c and d. There are four outputs, p, q, r and s. The black-box view of the combinational circuit and the minimal SOP expressions for the four functions are shown below :

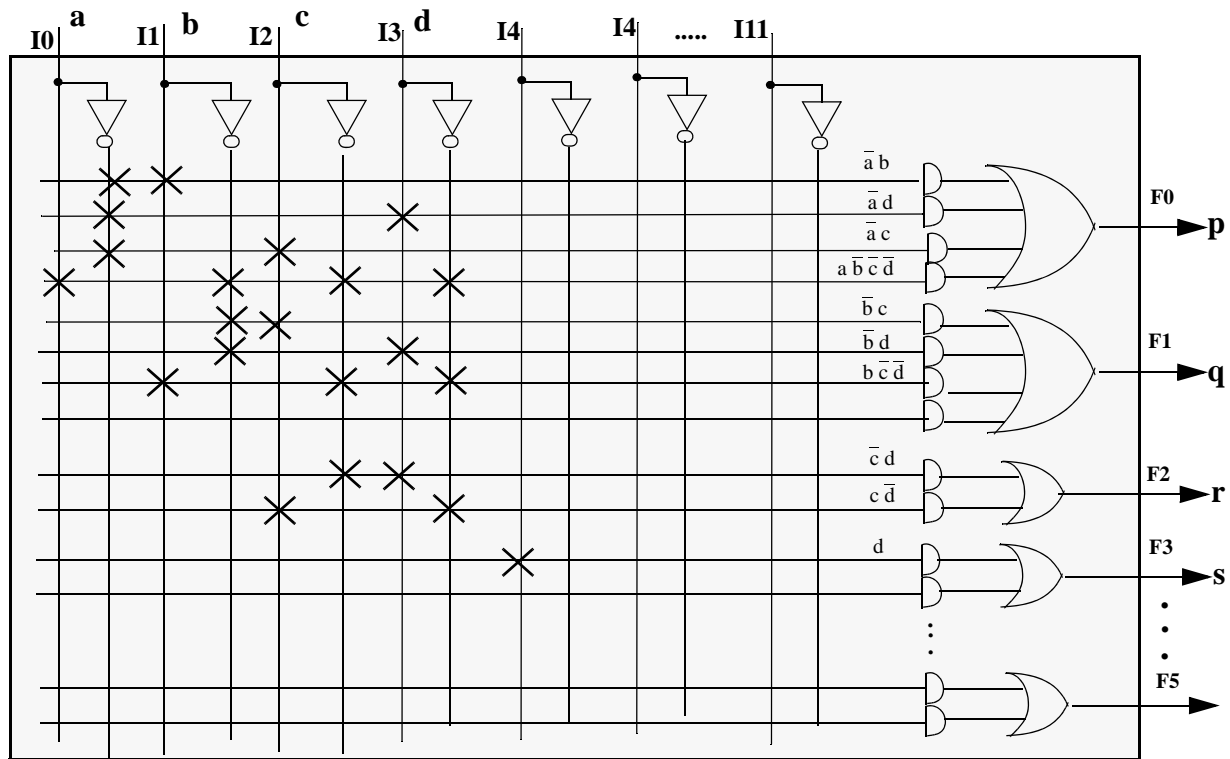


Implement the four functions by using a **single** Monolithic Memories **12H6 PAL chip** whose description handout is given **in class**. **No** off-chip (external) connections **nor** external gates (no other chip) can be used.

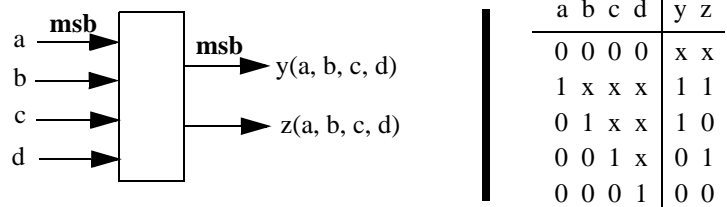
A17) We implement the three functions by using a **single** Monolithic Memories **12H6 PAL chip** :

$p(a, b, c, d) = \bar{a}b + \bar{a}d + \bar{a}c + a\bar{b}\bar{c}\bar{d}$	$r(a, b, c, d) = \bar{c}d + c\bar{d}$
$q(a, b, c, d) = \bar{b}c + \bar{b}d + b\bar{c}\bar{d}$	$s(a, b, c, d) = d$

The programming of the PAL chip is as follows :



Q18) Consider the following 4-input, 2-output combinational circuit and its operation table :



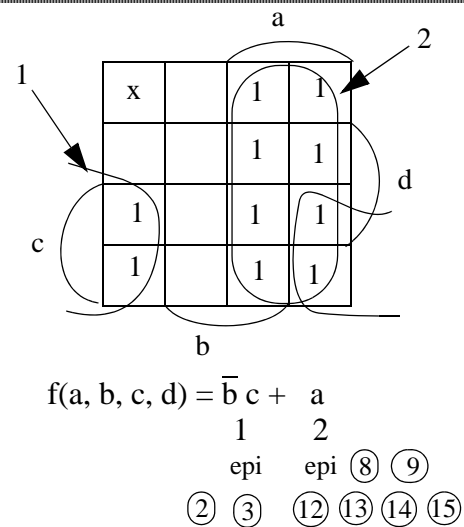
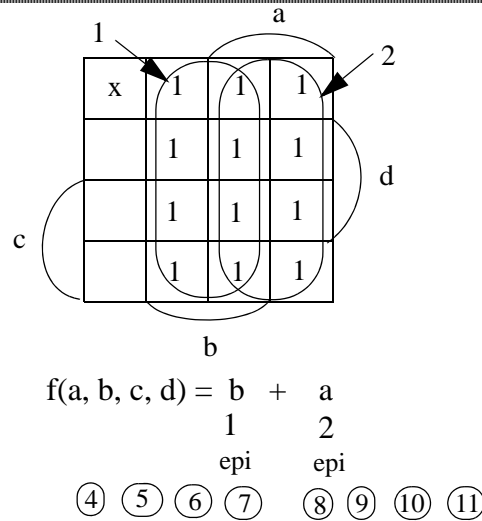
- Obtain the truth table of the combinational circuit.
- Obtain the minimal **SOP** expressions of the two functions by using the **K-map** method as done **in class**.
- Implement the two outputs by using minimum number of smallest **generic ROMs**. You may use other compo-

nents but a minimum number of them. You must use **at least** one ROM. Show the ROM implementation as done in class.

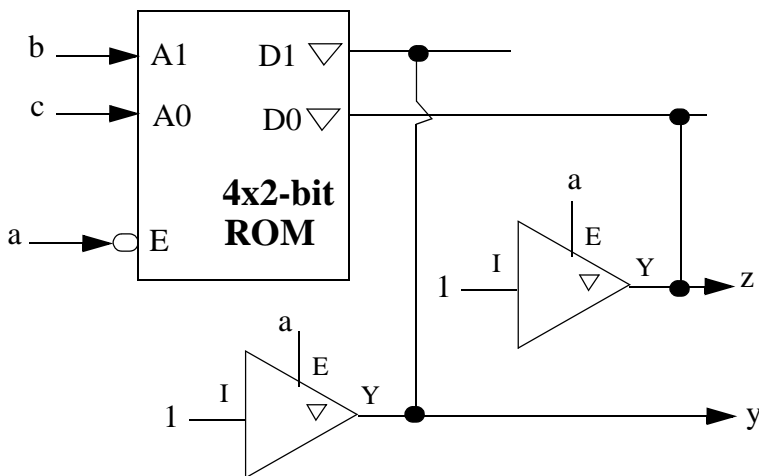
A18) The ROM implementation by using the **K-map** method is as follows :

	a b c d	y	z
0	0000	x	x
1	0001	0	0
2	0010	0	1
3	0011	0	1
4	0100	1	0
5	0101	1	0
6	0110	1	0
7	0111	1	0
8	1000	1	1
9	1001	1	1
10	1010	1	1
11	1011	1	1
12	1100	1	1
13	1101	1	1
14	1110	1	1
15	1111	1	1

The Minterm lists : $\begin{cases} y(a, b, c, d) = \sum m(4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15) + d(0) \\ z(a, b, c, d) = \sum m(2, 3, 8, 9, 10, 11, 12, 13, 14, 15) + d(0) \end{cases}$



Since both outputs are independent of input "d," and are 1 when "a" is 1, we can use a ROM of size 4x2-bit to output the values :

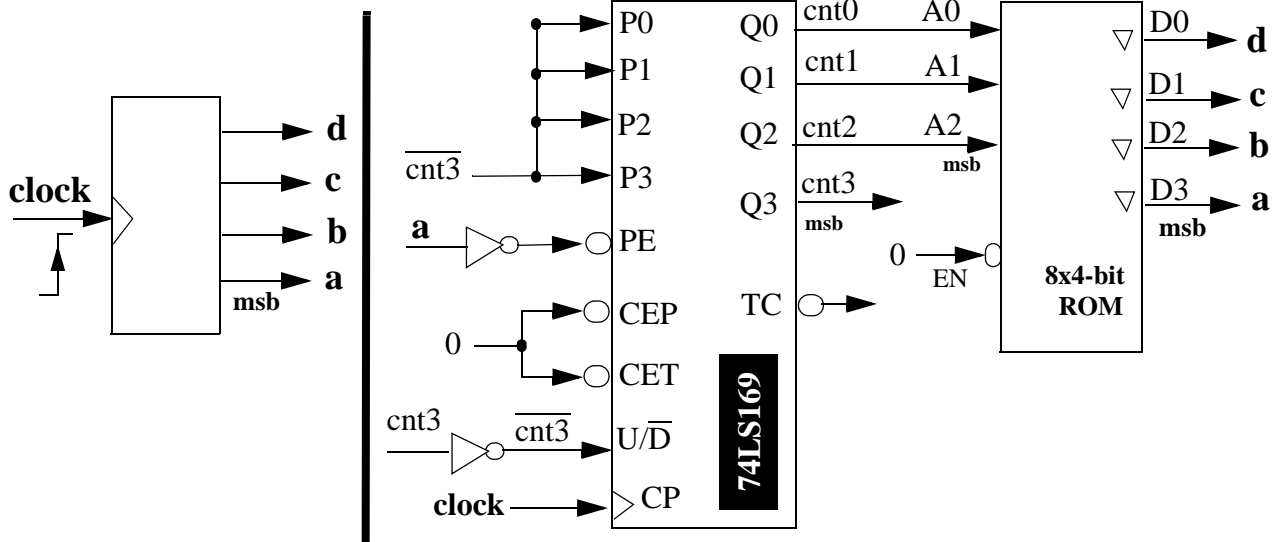


ROM Table

b	c	y	z
A1	A0	D1	D0
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	0

The number of unused bits is 0.

Q19) Consider a **sequential** circuit with a clock input and four outputs. The black-box view, the implementation of the black box with a **74LS169** counter and a **generic 8x4-bit ROM** and the content of the ROM are shown below. Determine what this sequential circuit does by continuing with the following table and showing the values for **12** clock periods :



The ROM content :

ROM Location	cnt2 A2	cnt1 A1	cnt0 A0	a D3	b D2	c D1	d D0
0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1
2	0	1	0	0	0	1	1
3	0	1	1	0	1	1	1
4	1	0	0	1	1	1	1
5	1	0	1	0	0	0	1
6	1	1	0	0	0	1	1
7	1	1	1	0	1	1	1

Continue with the table below :

Time	cnt3	cnt2	cnt1	cnt0	a b c d
t0	0	0	0	0	0 0 0 0
t1	0	0	0	1
....	
t11	

A19) We determine what this sequential circuit does, by completing the table for 12 clock periods :

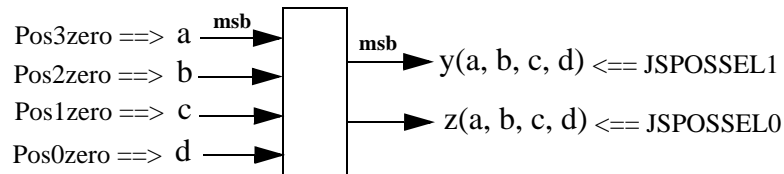
Time	cnt3	cnt2	cnt1	cnt0	a b c d
t0	0	0	0	0	0 0 0 0
t1	0	0	0	1	0 0 0 1
t2	0	0	1	0	0 0 1 1
t3	0	0	1	1	0 1 1 1
t4	0	1	0	0	1 1 1 1

Time	cnt3	cnt2	cnt1	cnt0	a	b	c	d
t5	1	1	1	1	0	1	1	1
t6	1	1	1	0	0	0	1	1
t7	1	1	0	1	0	0	0	1
t8	1	1	0	0	1	1	1	1
t9	0	0	0	0	0	0	0	0
t10	0	0	0	1	0	0	0	1
t11	0	0	1	0	0	0	1	1

We see that the counter counts in a cycle as follows : It counts from 0 4. Then, it loads 15 and counts down 12 and loads 0 to count up again.

The ROM outputs follow the rightmost three bits of the counter in such a way that when counting up at each count more ROM outputs become 1 : 0000-0001-0011-0111-1111. When counting down at each count more ROM outputs become 0, except when the counting down reaches 12 : 0111-0011-0001-1111.

Q20) Consider Macro 3, **M3**, of the term project studied in class and designed in the lab. It is the “Rightmost Zero Display” circuit that outputs the position number of the rightmost display that has a zero. The black box view below shows its inputs and outputs with **shorter** names :



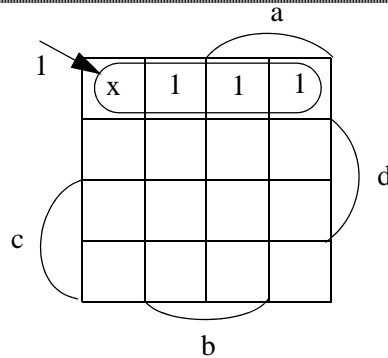
Implement the two functions (outputs) by using a **single** Monolithic Memories **12H6 PAL chip** whose description handout is given in class. **No** off-chip (external) connections **nor** external gates (no other chip) can be used. In order to use the PAL, show the following steps :

- i) Obtain the truth table of Macro 3.
- ii) Obtain the minimal SOP expressions of the two functions by using the **K-map** method as done **in class**.
- iii) Show the PAL implementation as done **in class**.

A20) The PAL implementation by using the **K-map** method is as follows :

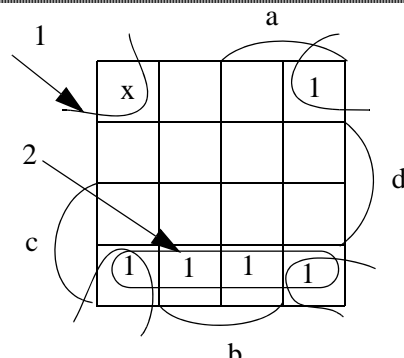
	a	b	c	d	y	z
0	0	0	0	0	x	x
1	0	0	0	1	0	0
2	0	0	1	0	0	1
3	0	0	1	1	0	0
4	0	1	0	0	1	0
5	0	1	0	1	0	0
6	0	1	1	0	0	1
7	0	1	1	1	0	0
8	1	0	0	0	1	1
9	1	0	0	1	0	0
10	1	0	1	0	0	1
11	1	0	1	1	0	0
12	1	1	0	0	1	0
13	1	1	0	1	0	0
14	1	1	1	0	0	1
15	1	1	1	1	0	0

The Minterm lists : $\begin{cases} y(a, b, c, d) = \sum m(4, 8, 12) + d(0) \\ z(a, b, c, d) = \sum m(2, 6, 8, 10, 14) + d(0) \end{cases}$



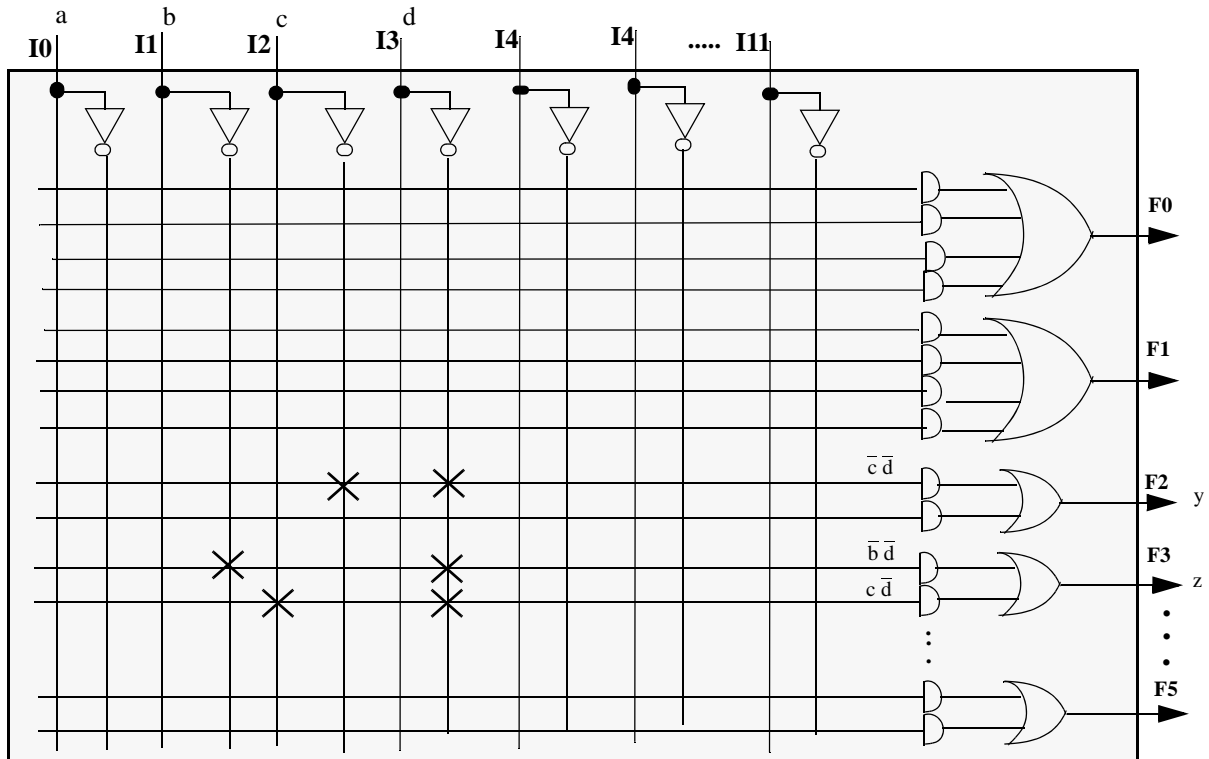
$$f(a, b, c, d) = \bar{c} \bar{d}$$

1
epi
④ ⑧ ⑫

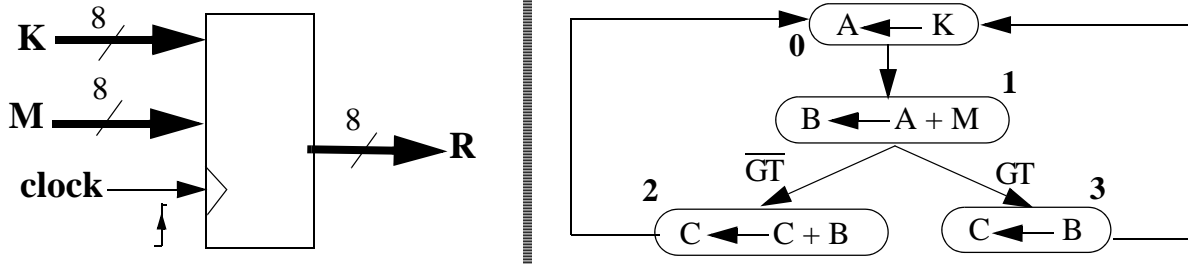


$$f(a, b, c, d) = \bar{b} \bar{d} + c \bar{d}$$

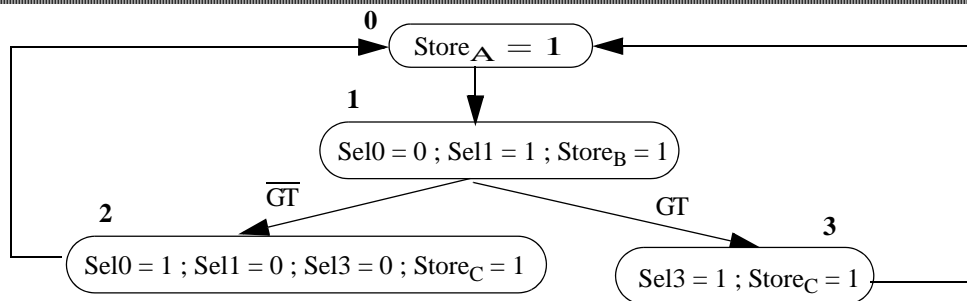
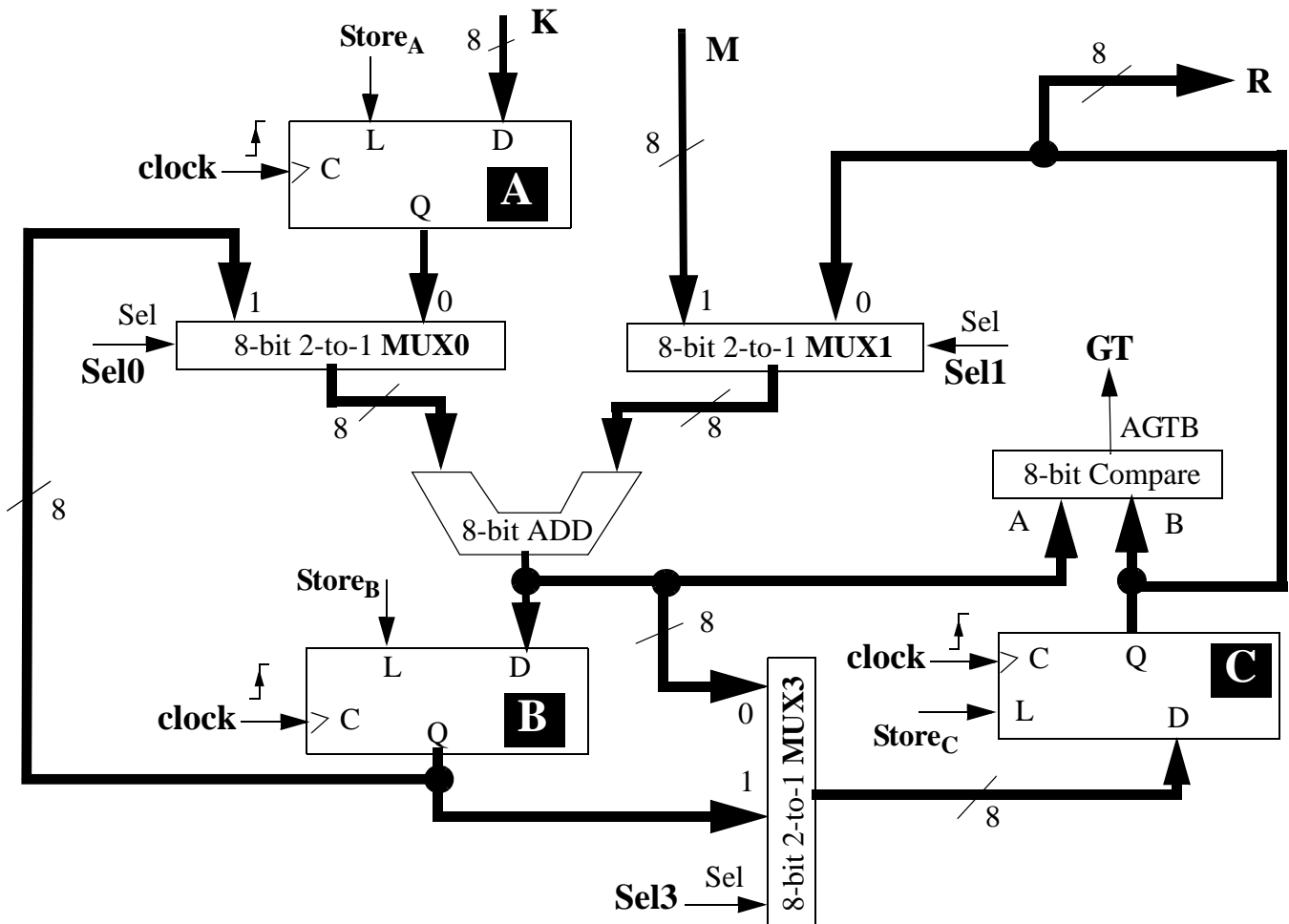
1 2
epi epi
⑧ ⑥ ⑭



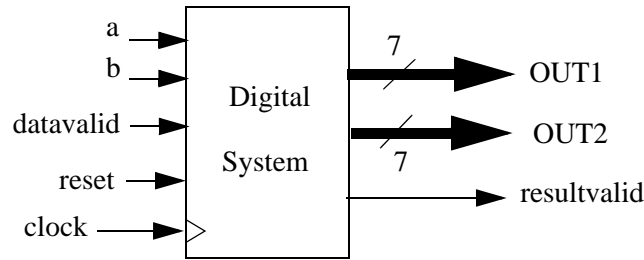
Q21) Consider the following **digital system** whose black-box view and **high-level** state diagram are shown below. Numbers **K**, **M** and **R** 8-bit **Unsigned Binary** numbers. Draw the **datapath** and **low-level** state diagram of the digital system.



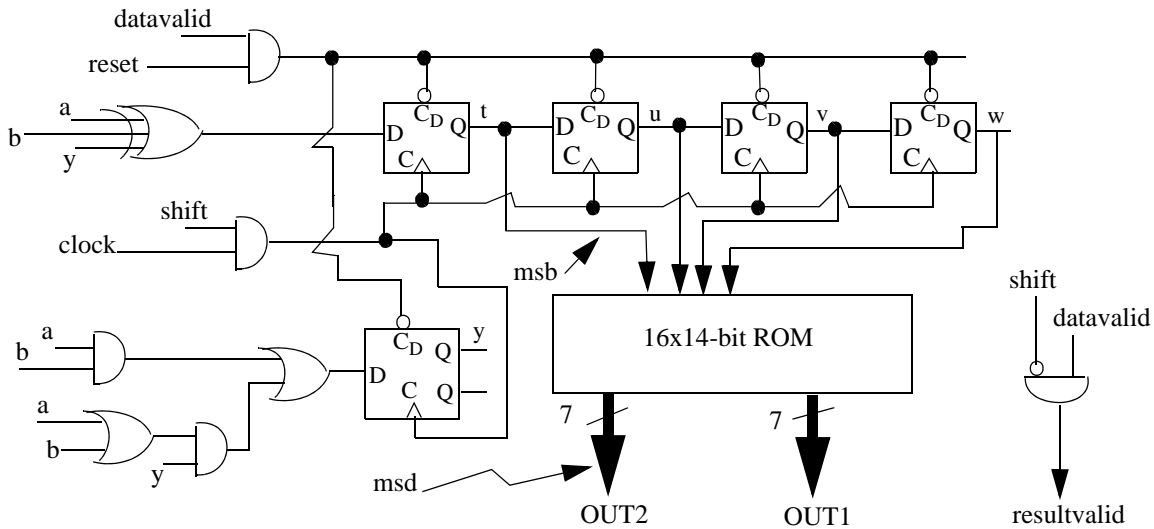
A21) The datapath and the low-level state diagram are shown below :



Q22) Consider the **digital system** below where all inputs except reset are synchronous with the clock signal :



The detailed internal circuit diagram of its **data unit** is shown below :



The reset signal is active during power on and is used to initialize the flip-flops to 0 ! After the reset goes to 1, the normal operation of this data unit starts. The normal operation starts when “a” and “b” are valid. When “a” and “b” are valid, the datavalid input is 1. The normal operation is indicated by a period of time the datavalid signal is 1 and then for at least one clock period it is 0. Then, datavalid is 1 for some time and 0 again, and so on. The clock period datavalid goes to 1, the shift signal goes to 1. The shift signal is generated by the control unit of this digital system.

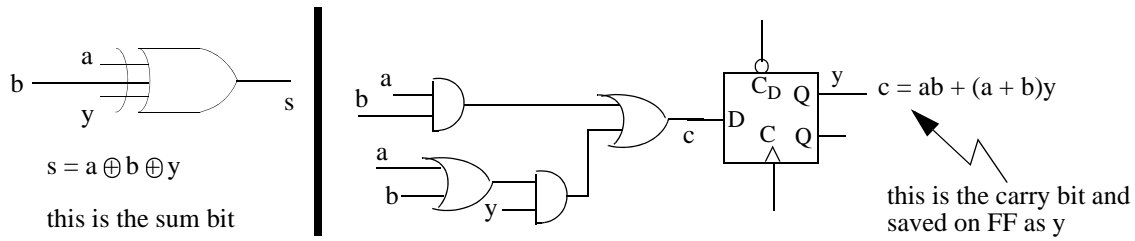
a) What does this circuit do ? What is its function ? Does it have any limitation ? Your analysis should include the timing of the internal control signal (shift) and the output control signal (resultvalid) with respect to the input control signals reset and datavalid. For example, for how long the shift is 1, given that the datavalid signal has been active ?

b) The ROM shown above functions as a 4-bit unsigned binary-to-7-segment decoder for **two** decimal digits. Indicate the addresses in Hexadecimal and the content of each ROM location in binary.

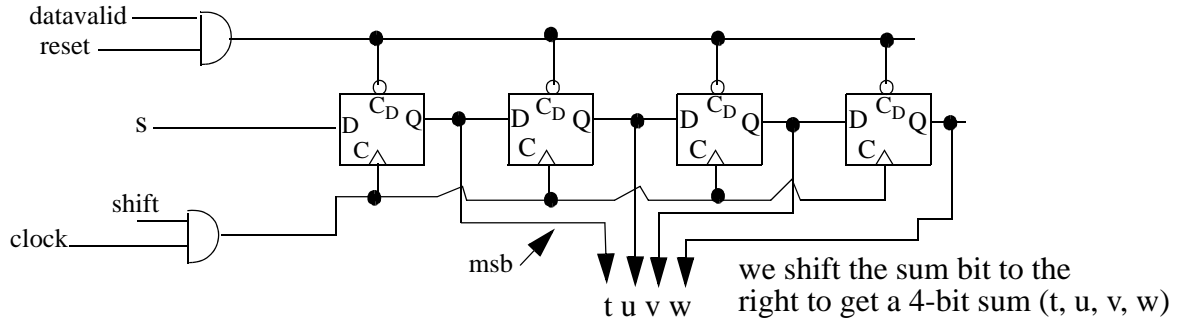
Hint : The sequential circuit is too complex to be analyzed by using the formal analysis rules. Thus, break up the circuit into blocks and analyze each block separately. Then, analyze the whole circuit based on the interaction of these blocks.

A22) a) There are **three** blocks in the data unit : the arithmetic block, the shift register block and the result block as shown below :

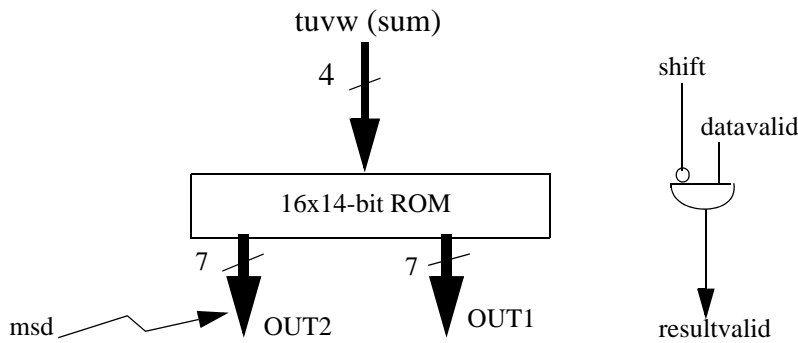
Arithmetic block



Shift register block



Result block



By looking at the arithmetic and shift register blocks, we see that when “a” and “b” are valid, they are added by the arithmetic block. The result of the addition is stored on the leftmost flip-flop of the shift register and the carry is stored on flip-flop “y.” In the next clock period, “a” and “b” are added by using the carry of the previous bit addition. This continues for four clock periods as the four sum bits are stored on four flip-flops. Thus, we add two 4-bit unsigned binary numbers input serially. Also, the shift signal is 1 four clock periods from the moment datavalid is 1 so as to shift in each one of the four sum bits.

According to the simple gate network in the result block, the validity of the result is indicated by no shifting and datavalid equal to 1. It is no shifting because, as shifting continues, the result is not valid yet (the result is valid when the shift is 0). We see that datavalid becomes 1 for the first addition and stays on until the next addition which starts after datavalid is 0 for one clock period at least.

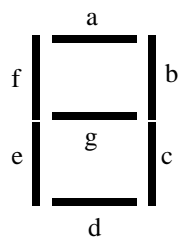
The ROM outputs the sum result by using **two** decimal digits. But, when datavalid is 1 and shift is 1, it outputs an intermediate addition result. When datavalid is 0, the output of the ROM is not valid either. Thus, there is a valid result as long as shift is 0 and datavalid is 1 (resultvalid is 1).

In summary, we add two 4-bit unsigned numbers and display the result on two 7-segment displays. There is a limitation of the circuit : the result is less than or equal to 15 as the ROM circuit receives four inputs !

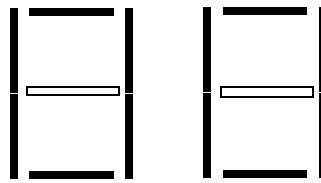
b) The 16x14-bit ROM circuit will show a result which can be between 0 and 15. The content is as follows :

Location	Contents	
	OUT2 a b c d e f g	OUT1 a b c d e f g
0	0000000	1111110
1	0000000	0110000
2	0000000	1101101
3	0000000	1111001
4	0000000	0110011
5	0000000	1011011
6	0000000	1011111
7	0000000	1110000

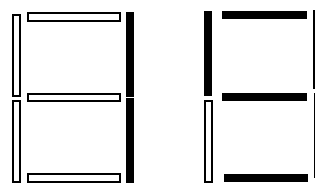
Location	Contents	
	OUT2 a b c d e f g	OUT1 a b c d e f g
8	0000000	1111111
9	0000000	1111011
A	0110000	1111110
B	0110000	0110000
C	0110000	1101101
D	0110000	1111001
E	0110000	0110011
F	0110000	1011011



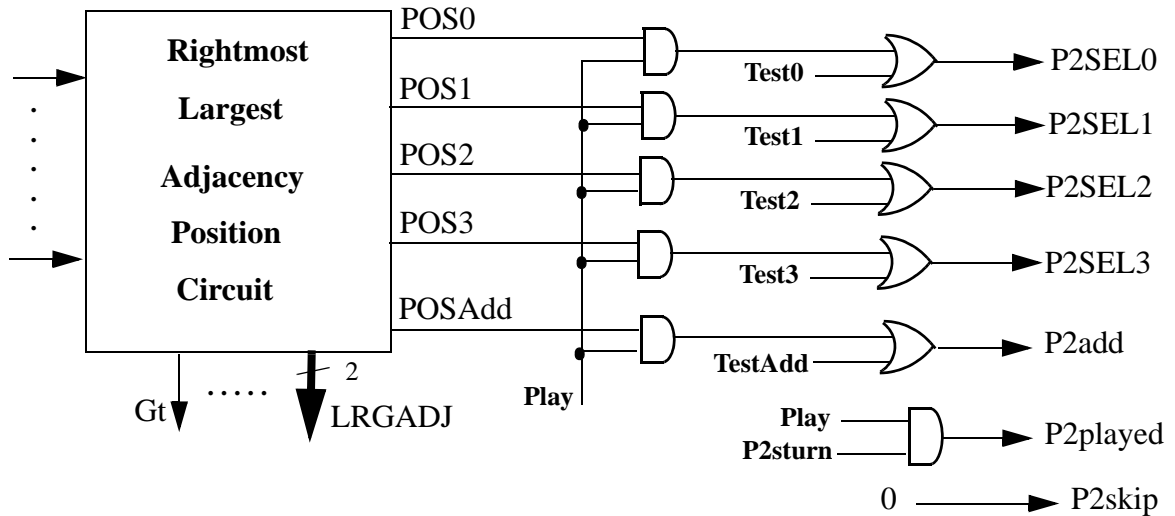
The smallest result shown is 0 :



The largest result shown is 15 :



Q23) Consider Block 6, the Machine Play Block, of the **term project**. The **datapath** of the circuit that implements Block 6 for the following specific playing strategy is shown below :



The strategy implemented : Play on the rightmost largest adjacency position. If direct playing and adding give the same largest adjacency, select direct playing.

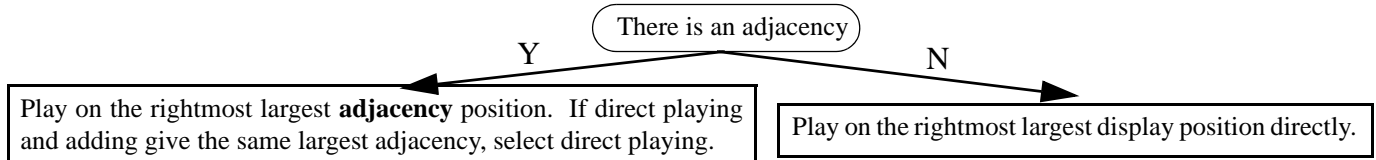
In the circuit, **POS0 - POS3** indicate which position has the largest adjacency (**LRGADJ**). **PosAdd** indicates whether the largest adjacency is with direct playing or an addition. **Play** is 1 in the last Player 2 state. **P2sturn** is equivalent to **S4** which means it is Player 2 that has the turn.

i) Assume that the code is 17. 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 remaining rows of the table. You will **circle** the position played as shown on the first row :

RD	Displays before play PD3 PD2 PD1 PD0	Displays after play PD3 PD2 PD1 PD0	D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
4	A B A B	A B A (4)	Direct	0	4	No
2	C 1 F 1					
7	7 E E E					
3	6 3 6 3					
9	F 9 9 9					

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.

ii) **Modify** the above circuit to implement the following strategy :



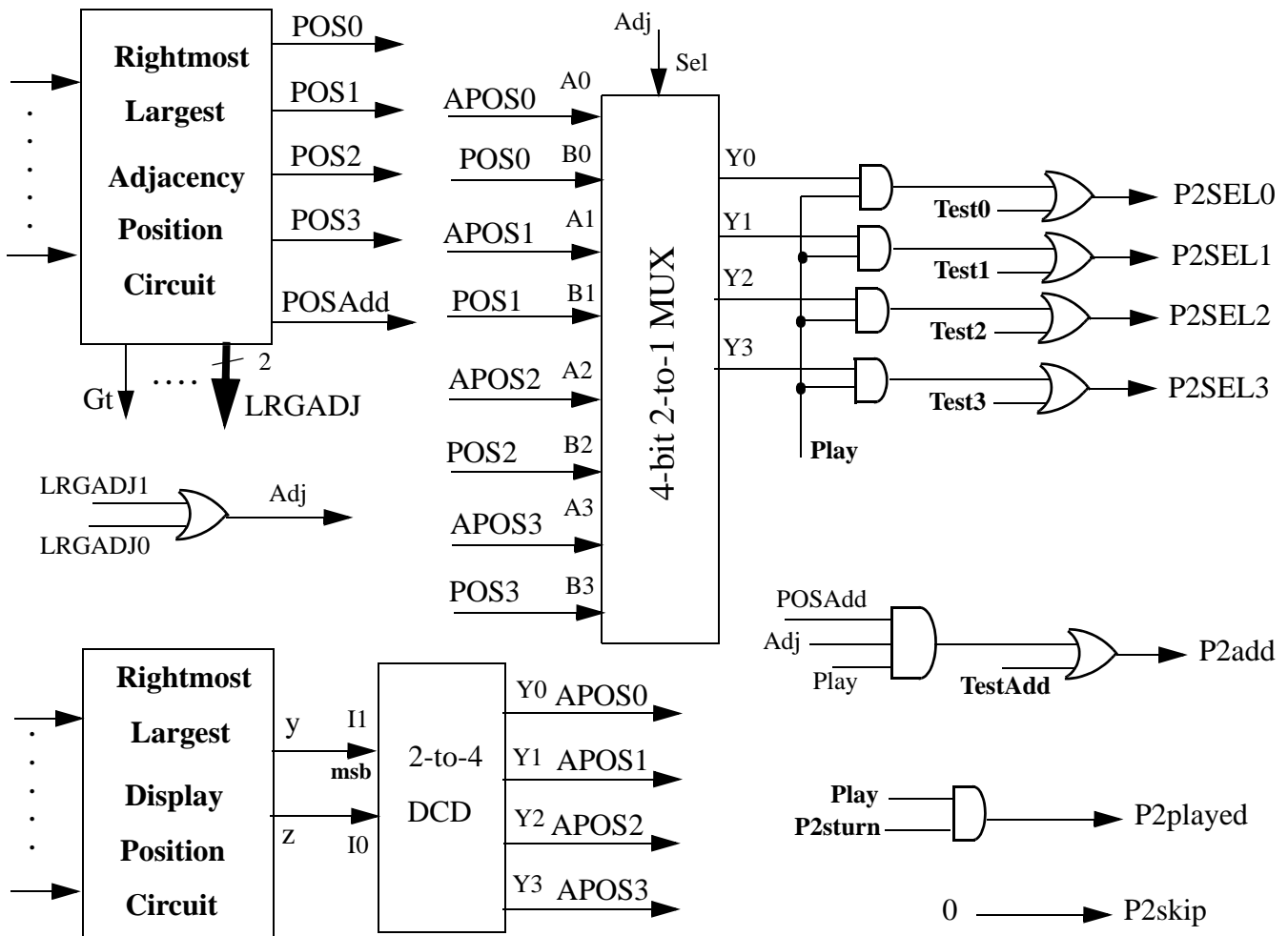
Use Macro 2, **M2**, which is the “Rightmost Largest Display” circuit as a black box with outputs (y, z) in your modified circuit. You can just show the **modified** portion of the circuit, **not** the whole circuit.

A23)

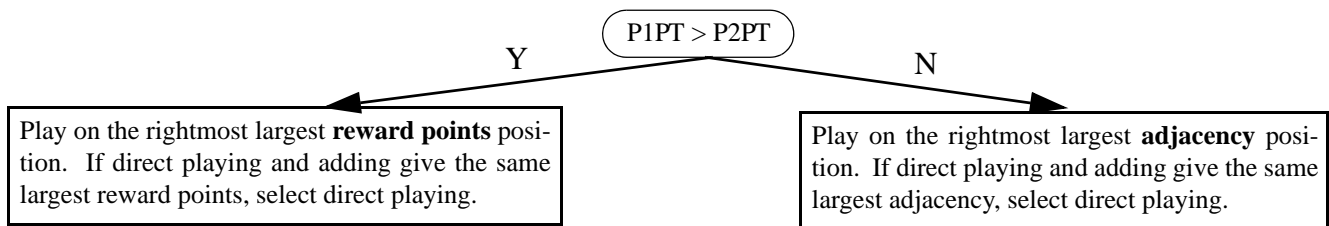
i) The table is completed as follows :

RD	Displays before play PD3 PD2 PD1 PD0	Displays after play PD3 PD2 PD1 PD0	D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
4	A B A B	A B A (4)	Direct	0	4	No
2	C 1 F 1	C 1 (1) 1	Add	2	12	Yes
7	7 E E E	(E) E E E	Add	3	112	Yes
3	6 3 6 3	6 3 (3) 3	Direct	2	12	Yes
9	F 9 9 9	(9) 9 9 9	Direct	3	72	Yes

ii) The modified portion of the datapath is as follows :



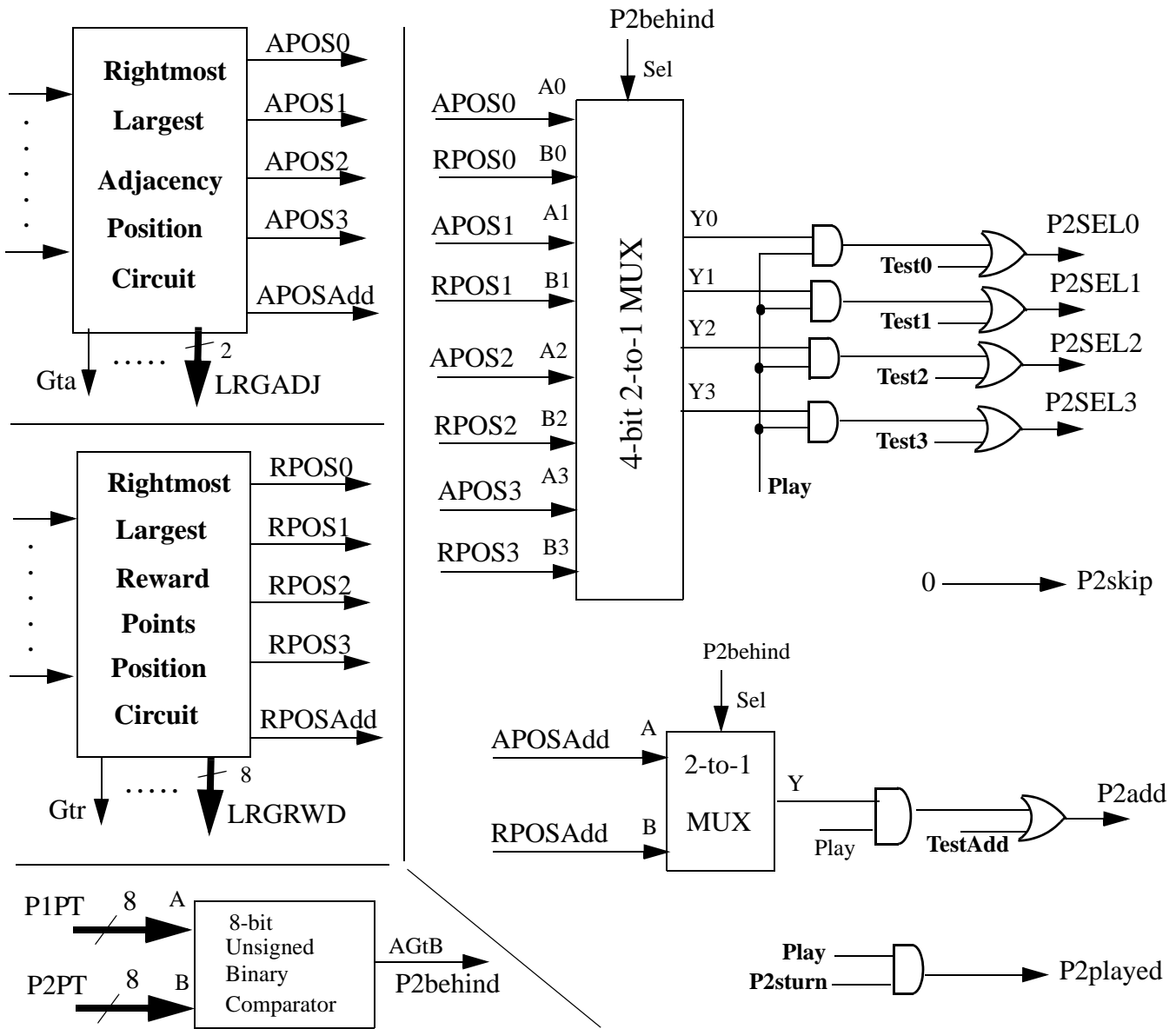
Q24) Consider Block 6, the Machine Play Block, of the **term project**. It has the following playing strategy :



a) Assume that the code is F1. The table below shows the random digit, position displays **before** and **after** the **machine** player plays, if the machine player is behind, 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.

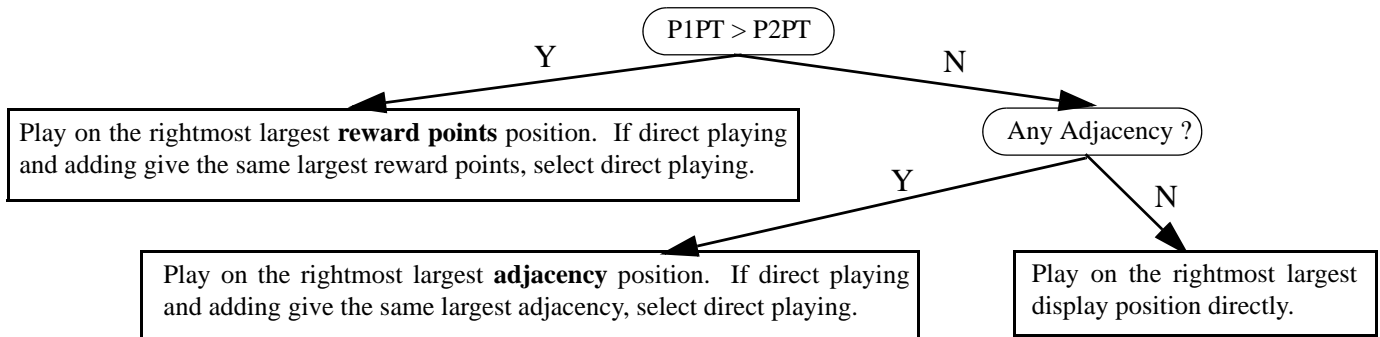
A large portion of the **datapath** of the circuit that implements Block 6 for the above specific playing strategy (two rectangles and one oval) is designed below :

RD	Displays before play				Displays after play				Player 2 is behind	D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
	PD3	PD2	PD1	PD0	PD3	PD2	PD1	PD0					
5	F	A	A	F					Yes				
2	A	E	8	7					No				
3	9	C	9	3					No				
6	9	F	6	6					Yes				
1	F	C	A	8					No				



Play is 1 in the last Player 2 state. **P2sturn** is equivalent to S4 which means it is Player 2 that has the turn to play.

b) Assume that the strategy is changed as follows (three rectangles and two ovals) :



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

RD	Displays before play				Displays after play				Player 2 is behind	D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
	PD3	PD2	PD1	PD0	PD3	PD2	PD1	PD0					
5	F	A	A	F					Yes				
2	A	E	8	7					No				
3	9	C	9	3					No				
6	9	F	6	6					Yes				
1	F	C	A	8					No				

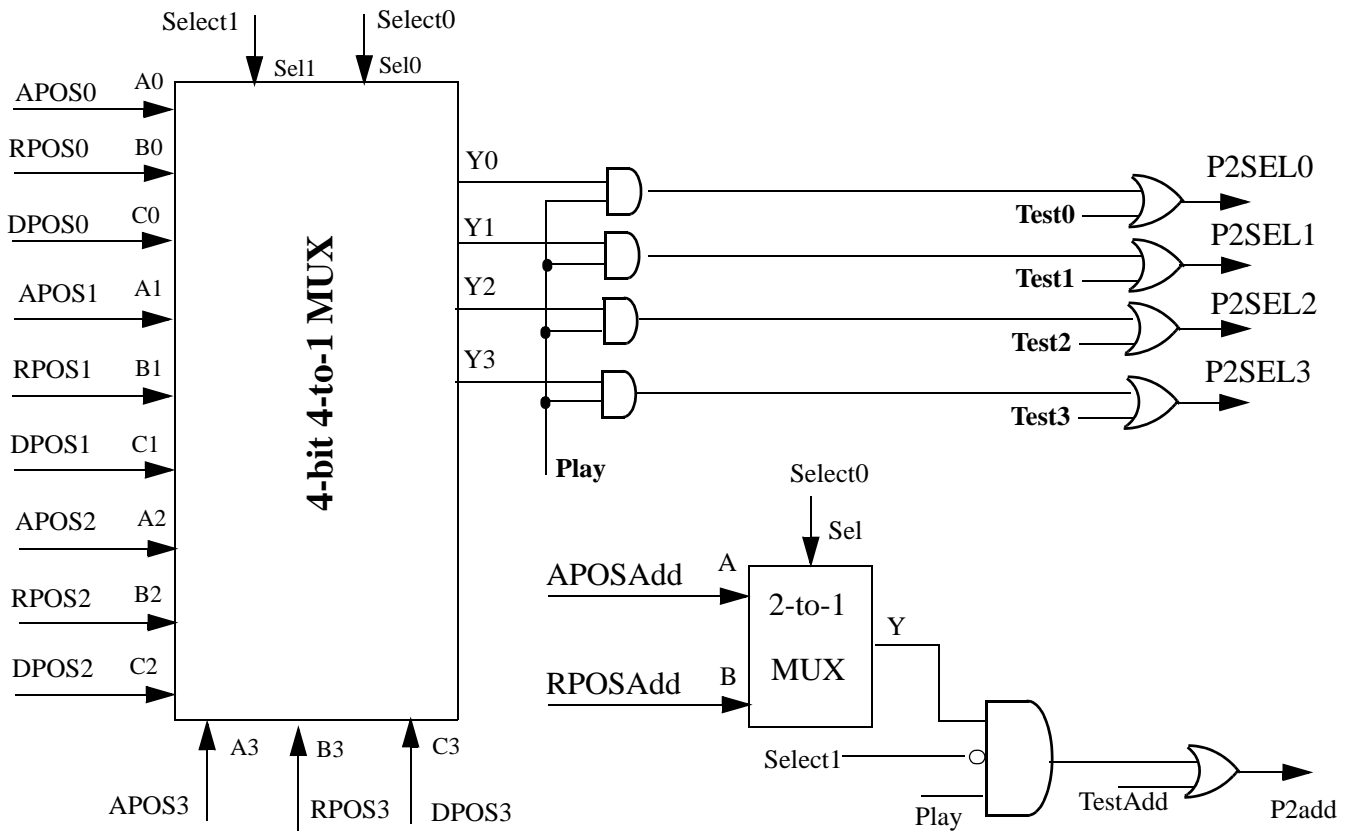
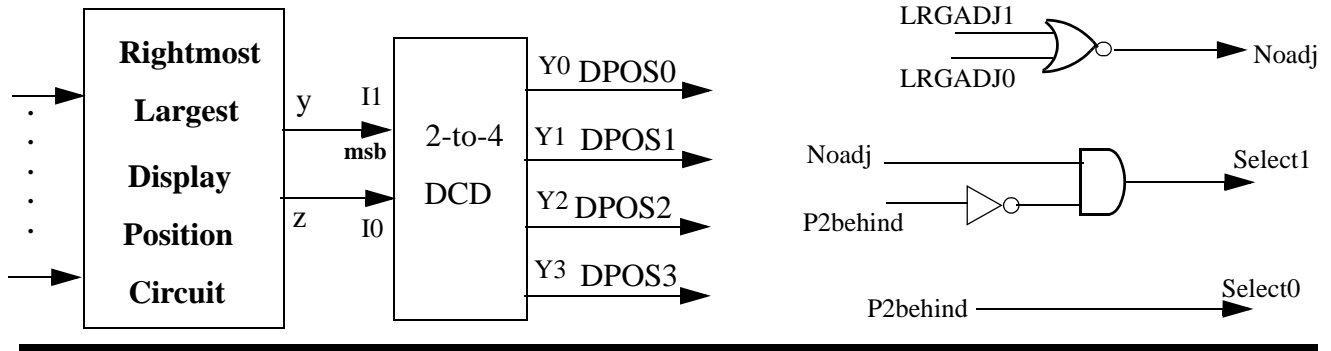
c) **Modify** the above circuit to implement the **new** strategy (three rectangles and two ovals).

Use Macro 2, **M2**, which is the “Rightmost Largest Display” circuit as a black box with outputs (y, z) in your modified circuit. You can just show the **modified** portion of the circuit, **not** the whole circuit.

A24) a) The table is completed as follows :

RD	Displays before play				Displays after play				Player 2 is behind	D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
	PD3	PD2	PD1	PD0	PD3	PD2	PD1	PD0					
5	F	A	A	F	F	A	(F)	F	Yes	A	1	150	Yes
2	A	E	8	7	A	E	8	(2)	No	D	0	2	No
3	9	C	9	3	9	C	(3)	3	No	D	1	6	Yes
6	9	F	6	6	(F)	F	6	6	Yes	A	1	30	Yes
1	F	C	A	8	F	C	A	(1)	No	D	0	9	No

c) The modified portion of the datapath is as follows :



b) The table is completed as follows :

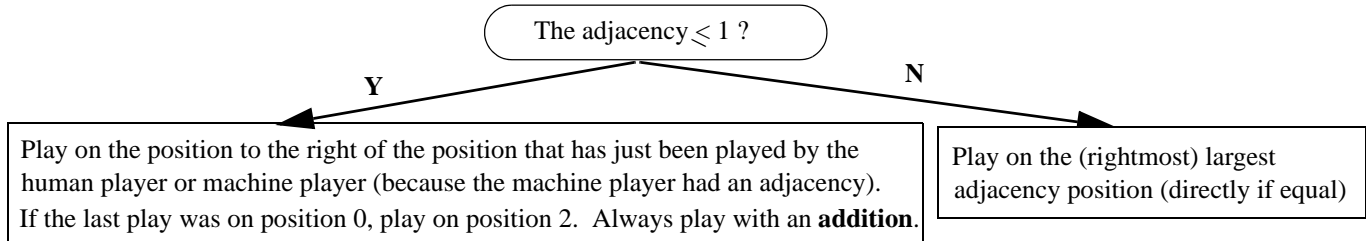
RD	Displays before play PD3 PD2 PD1 PD0	Displays after play PD3 PD2 PD1 PD0	Player 2 is behind	D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
5	F A A F	F A (F) F	Yes	A	1	150	Yes
2	A E 8 7	A (2) 8 7	No	D	0	2	No
3	9 C 9 3	9 C (3) 3	No	D	1	6	Yes
6	9 F 6 6	(F) F 6 6	Yes	A	1	30	Yes
1	F C A 8	(1) C A 8	No	D	0	1	No

Q25) 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 adjacency position (directly if equal)

a) Based on the discussions in the lab, how many clock periods does the machine player take to play ? **Explain**.

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



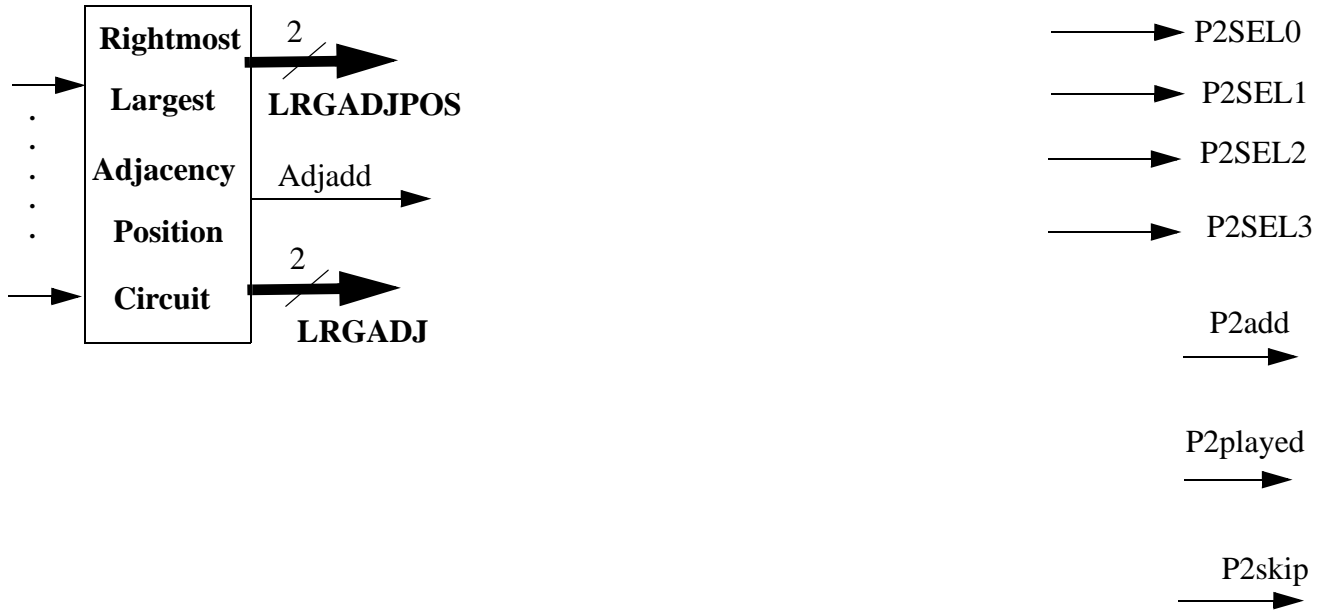
The table below shows the random digit, position displays **before** and **after** the **machine** player plays, the position the human/machine player has just played, whether the random digit is played directly or added, the number of adjacencies, the points earned by the **machine** player and if the machine player plays again. Note that each row is **independent**.

Assume that the code is **E9**. Complete the rows of the table. You will **circle** the position played :

RD	Displays before play				Last Play	Displays after play				D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
	PD3	PD2	PD1	PD0		PD3	PD2	PD1	PD0				
2	E	C	C	C	2								
7	1	8	7	0	1								
3	6	9	9	6	0								
8	4	C	4	8	1								
1	B	C	D	8	3								

c) i) **Design** the datapath of the machine player to implement the **new** strategy. In order to do that work on the figure below that shows portions of the datapath. In the figure, **LRGADJPOS** indicates the rightmost largest adjacency position, **Adjadd** indicates whether the largest adjacency is with direct playing or with an addition. **LRGADJ** indicates the amount of the largest adjacency. If you use **other** signals, **describe** what they are.

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



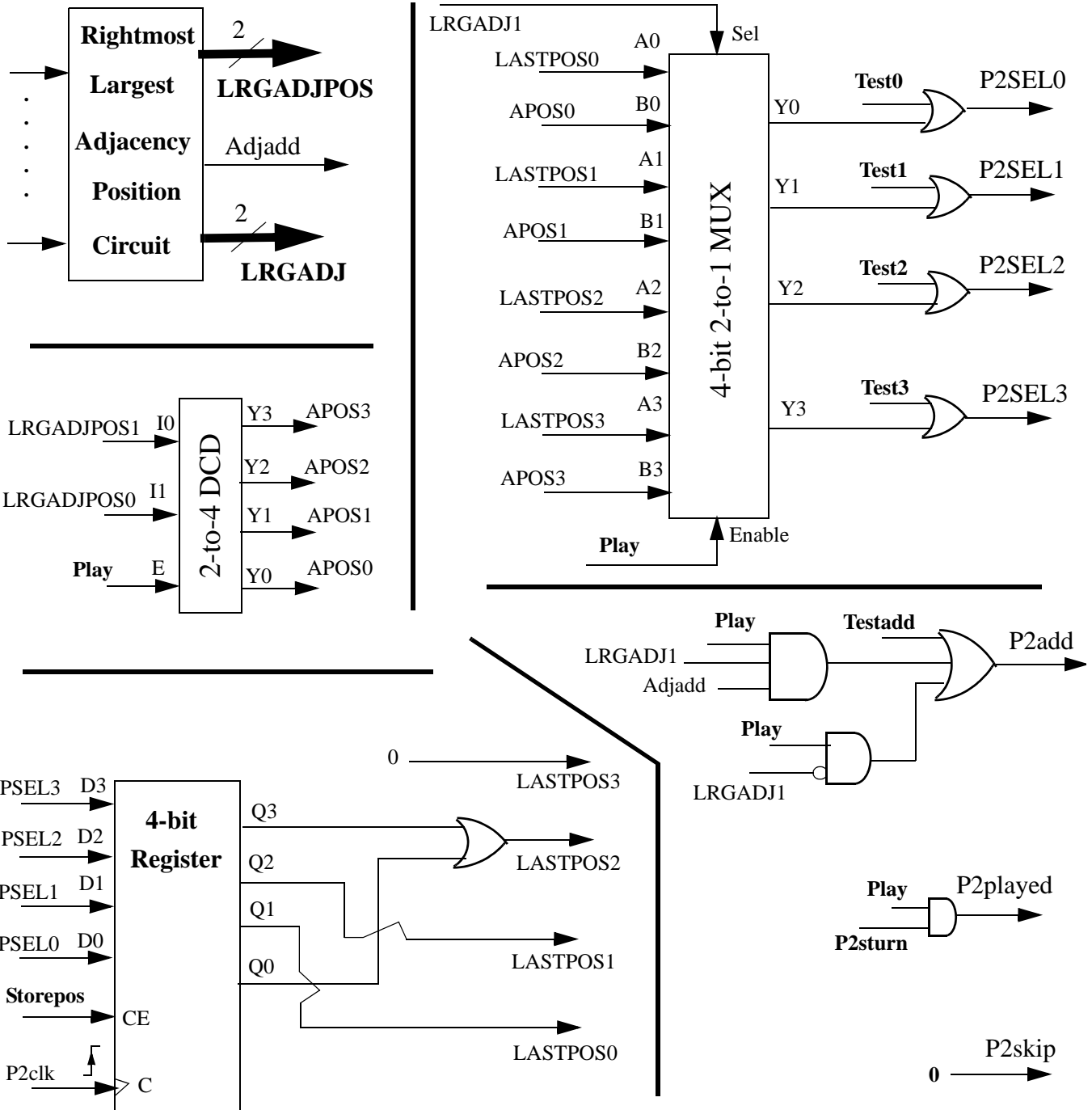
A25) a) It takes nine clock periods for the machine player to play since we have to collect all eight adjacencies in eight clock periods and then in the ninth clock periods we play. That is, it tests directly and with additions to collect eight adjacencies to determine the largest adjacency position, taking eight clock periods. Then it takes one clock period to play.

b) The table is completed as shown below.

RD	Displays before play				Last Play	Displays after play				D/A	Adjacency	Reward Points (Decimal)	Plays Again ?
	PD3	PD2	PD1	PD0		PD3	PD2	PD1	PD0				
2	E	C	C	C	2	E	C	(E)	C	A	0	126	N
7	1	8	7	0	1	1	8	7	(7)	A	1	14	Y
3	6	9	9	6	0	6	9	9	(9)	A	2	108	Y
8	4	C	4	8	1	4	(4)	4	8	A	2	16	Y
1	B	C	D	8	3	B	(D)	D	8	A	1	26	Y

The strategy does **not** check for code digits and so misses to earn code reward points when RD is 7 and 1. However, by chance it earns code reward points when RD is 2 and 3.

c) i) The modified Datapath is shown below.



ii) The machine player still takes **nine** clock periods to play since we have to collect the adjacency information taking 8 clock periods. Then, the next clock period we play.

The new signals used are as follows : **Play** is 1 in the last clock period. **P2sturn** is 1 when it is machine player's turn. That is when it is state 4 of the ppm operation diagram. **Testadd** is 1 when it is states 4 through 7 of the machine player operation diagram. **Test0, Test1, Test2** and **Test3** are active when we test positions 0, 1, 2 and 3, respectively. **Storepos** is 1 the clock period after the human player or machine player plays. That is it is 1 when it is state 2 or state 5 of the ppm operation diagram.