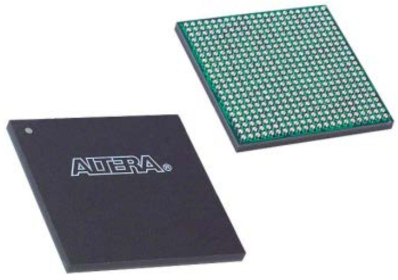
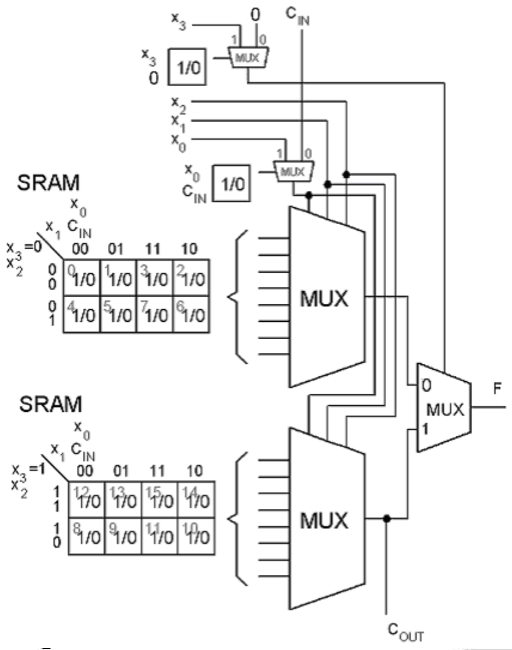
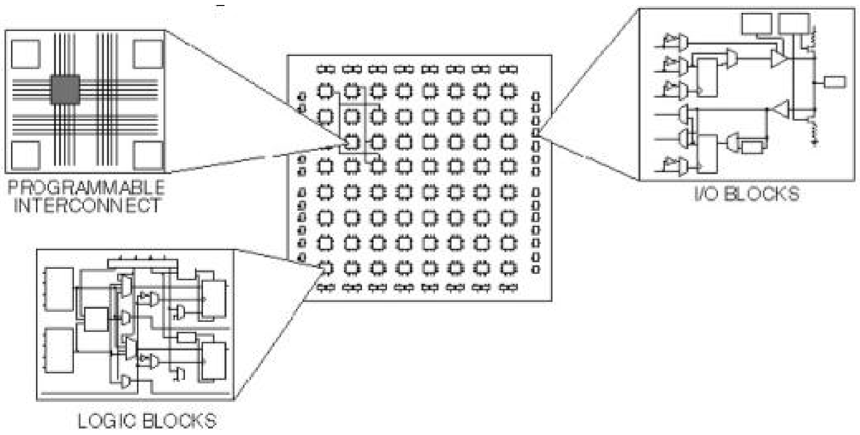
Digital Design   *IE1204/5*

Exercises







Compiled by   *William Sandqvist*   `william@kth.se`

ICT/ES   Electronic Systems

Number systems and codes

1.1
Enter the corresponding binary numbers for the following decimal numbers (base 10).
a) 9 b) 12 c) 71 d) 503

1.2
Convert the following binary number to decimal.
a) 101101001_2 b) 110100.010_2

1.3
Convert the following binary numbers (base=2) to the corresponding octal numbers (base=8) and hexadecimal numbers (base=16).
a) $01\ 1101_2$ b) $1000\ 1011_2$ c) $1\ 0011\ 0101_2$ d) $1101\ 1110\ 1001\ 0001_2$ e) $10\ 1001.001_2$

1.4
Convert the following hexadecimal numbers (base=16) to the corresponding octal numbers (base=8).
a) $94D_{16}$ b) $9E.7A_{16}$

1.5
Convert the octal (base=8) number 4515_8 to the corresponding hexadecimal number (base=16).

1.6
Write the hexadecimal (base=16) number BAC_{16} in decimal form (base=10).

1.7
What characterizes Gray codes, and how can they be constructed?

1.8
Write the following "signed" numbers with two's complement notation, $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.
a) -23 b) -1 c) 38 d) -64

1.9
Write the following "signed" numbers with one's complement notation, $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.
a) -23 b) -1 c) 38 d) -0

Digital arithmetic

2.1
Add by hand the following pair of binary numbers.
a) 110 + 010 b) 1110 + 1001 c) 11 0011.01 + 111.1 d) 0.1101 + 0.1110

2.2
Add or subtract (addition with the corresponding negative numbers) the following pair of numbers. The numbers shall be represented as binary 4-bit numbers (Nibble) in two's complement form.
a) 1 + 2 b) 4 – 1 c) 7 – 8 d) -3 – 5

2.3
Multiply by hand following pairs of unsigned binary numbers.
a) 110·010 b) 1110·1001 c) 11 0011.01·111.1 d) 0.1101·0.1110

2.4
Divide by hand following pairs of unsigned binary numbers.b.
a) 110/010 b) 1110/1001

2.5

IEEE-754 standard for storage of 32-bit float.
Assume that a 32-bit float is stored in a register: $40C80000_{16}$ What real decimal number is this?

(2.6)

Floating point format's principles becomes more transparent if one of pedagogical reasons "scales down" to a 4-bit register size (Nibble). However, a 4-bit format would be practically unusable.

Assume the following four bit floating point format: $[b_3b_2b_1b_0] = (-1^{b_3}) \cdot (1.b_0) \cdot (2^{b_2b_1-1})$

The sign is expressed with the bit b_3, the mantissa is represented by one bit b_0, and the exponent has two bits b_2b_1 expressed as exess -1.

a) Count up the number that can be represented with full precision. Mark them on the number line.
b) How big is the largest quantization error?
c) Can the number 0 be represented? If not, suggest a change in the format so that 0 can be represented.

(2.7)

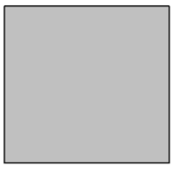
To examine the addition and multiplication of floating point, we assume now for pedagogical reasons a 6 bit format. (This is still to few bits to be practically usable).

$$[b_5b_4b_3b_2b_1b_0] = (-1^{b_5}) \cdot (1.b_2b_1b_0) \cdot (2^{b_4b_3-1})$$

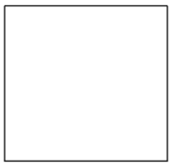
a) Which of the following numbers can be represented in this format? 0,25 0,8125 -1,375 4,25 7.5
b) Add the numbers ($b_5b_4b_3b_2b_1b_0$) 001111 and 010010. What is needed to avoid loss of precision?
c) Multiply the previous numbers with each other.

Sets and Cubes

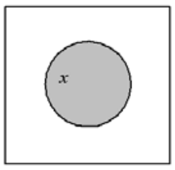
Venn-diagram representation



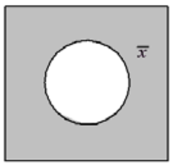
Constant ”1”



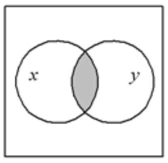
Constant ”0”



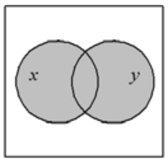
Variable x



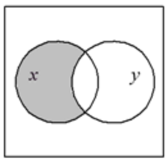
Variable $\overline{x}$



$x \cdot y$



$x + y$



$x \cdot \overline{y}$

3.1
Prove the distributive law with the help of Venn diagram.

$$x + y \cdot z = (x + y) \cdot (x + z)$$

3.2
Prove De Morgan's law using the Venn diagram.

$$\overline{x \cdot y} = \overline{x} + \overline{y}$$

3.3
a) Draw a Venn diagram for three variables and mark where the truth table all mintermer are placed.
b) Minimize the function using the Venn diagram.

$$f = \overline{x_2}\,\overline{x_1}\,x_0 + \overline{x_2}\,x_1\,x_0 + x_2\,\overline{x_1}\,x_0 + x_2\,x_1\,\overline{x_0} + x_2\,x_1\,x_0$$

Cube representation

3.4
a) Represent the following function of three variables as a 3-dimensional cube with Gray-coded corners.

$$f(x_2, x_1, x_0) = \sum m(0,2,3,4,6)$$

b) Use the cube to simplify the function.

Boolean algebra and gates

Boolean algebra

$A \cdot A = A \quad A \cdot 0 = 0 \quad A + 0 = A$

$A + A = A \quad A \cdot 1 = A \quad A + 1 = 1$

Distributive laws	$A\cdot(B + C) = A\cdot B + A\cdot C$ $A + (B\cdot C) = (A + B)\cdot(A + C)$
Kommutative laws	$A\cdot B = B\cdot A$ $A + B = B + A$
Associative laws	$(A\cdot B)\cdot C = A\cdot(B\cdot C)$ $(A + B) + C = A + (B + C)$

Absorbtion laws	$A + A\cdot B = A$ $A\cdot(A + B) = A$
Consensus laws	$A\cdot B + \overline{A}\cdot C =$ $A\cdot B + \overline{A}\cdot C + B\cdot C$
de Morgan laws	$\overline{(A + B)} = \overline{A}\cdot\overline{B}$ $\overline{(A\cdot B)} = \overline{A} + \overline{B}$

4.1

Use the laws of Boolean algebra to simplify the following logic expressions:

a) $f = a\cdot\overline{c}\cdot d + a\cdot d$

b) $f = a\cdot(\overline{b} + \overline{a}\cdot c + a\cdot b)$

c) $f = a + \overline{b} + \overline{a}\cdot b + \overline{c}$

d) $f = (a + b\cdot\overline{c})\cdot(\overline{a}\cdot\overline{b} + c)$

e) $f = (a + \overline{b})\cdot(\overline{a} + b)\cdot(a + b)$

f) $f = \overline{a}\cdot\overline{b}\cdot c + a\cdot b\cdot c + \overline{a}\cdot b\cdot c$

g) $f = \overline{a}\cdot b\cdot\overline{c} + \overline{a}\cdot b\cdot\overline{d} + c\cdot d$

h) $f = a + (\overline{\overline{a}\cdot\overline{b}})$

i) $f = \overline{\overline{a} + \overline{\overline{a}\cdot b + c}}$

4.2

Prove algebraically that the following relations are valid.

a) $(\overline{x_3}x_2 + 1 + x_3\overline{x_2})x_1 + x_2x_1 + x_2 + \overline{x_1} = 1$

b) $\overline{x_3}x_2x_1 + \overline{(x_3 + \overline{x_1})} = \overline{x_3}x_1$

c) $\overline{(x_2 + x_1)}\,\overline{x_2}\,\overline{x_1} + x_3 = \overline{x_2}\,\overline{x_1} + x_3$

d) $\overline{x_2 + x_1} + \overline{x_2}\,\overline{x_1} + x_3 = \overline{x_2}\,\overline{x_1} + x_3$

4.3

Simplify the following three expressions as much as possible.

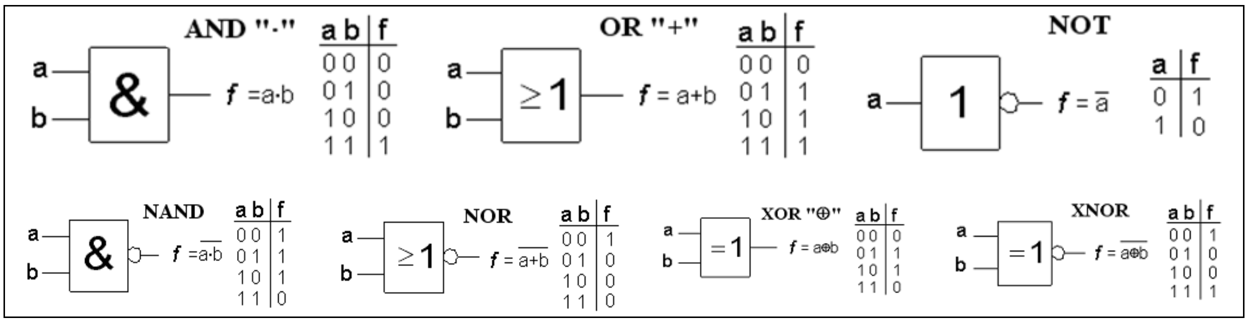
a) $(x + y)(\overline{x} + z)$ b) $(x + \overline{y} + xy)(x + \overline{y})\overline{x}y$ c) $x(1 + \overline{x}y) + \overline{x}$

4.4

Simplify the following expression as far as possible.

$$\overline{(a + b + \overline{c})(a + \overline{b} + \overline{c})(\overline{a} + \overline{\overline{b}c} + b\overline{c})}$$

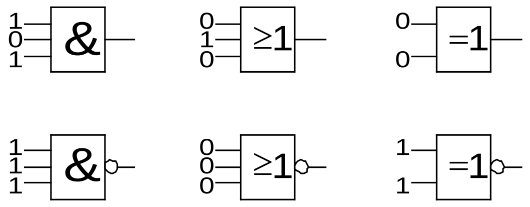
Gates

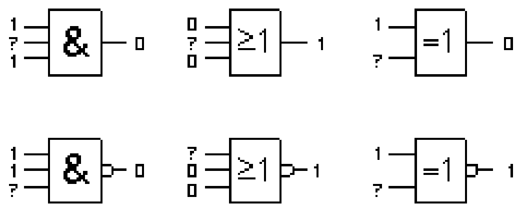


4.5

a) Specify the **output** 1/0 for the following six types of gates when the inputs are as given in the figure.

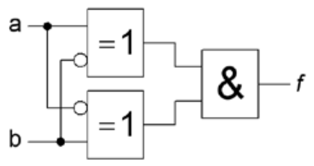
b) Specify the **input** 1/0 for the following six types of gates when the output signals are as given in the figure.





4.6

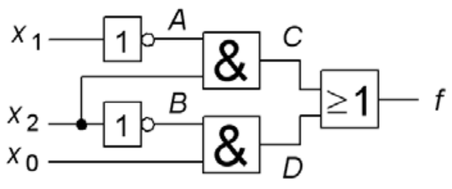
Simpify *f*(*a*,*b*) which are realized by the gate circuit, as much as possible, and specify the function name.

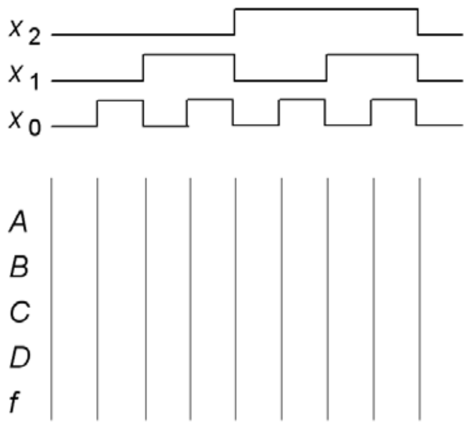


4.7

a) Draw timing diagram of signals *A*, *B*, *C*, *D*, *f*. The inputs x_0, x_1, and x_2 has the frequency ratio 4: 2:1 to "sweep" through the truth table combinations in the "right" order.

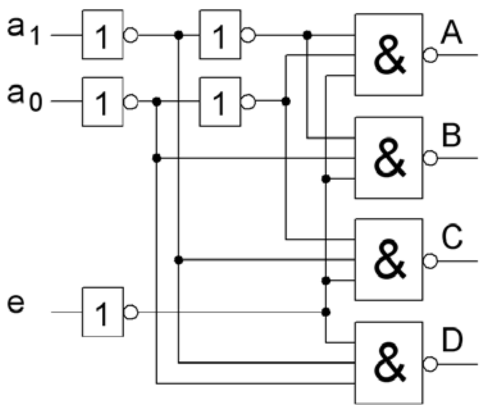
b) Write the truth table for the function *f*.





4.8

Specify the logical expressions for *A*, *B*, *C* and *D*.



4.9

Simplify the complex expressions below as much as possible.

a) $x_2 \oplus x_1 \oplus x_1 x_2$ b) $x_2 x_1 \oplus \overline{(x_2 + x_1)}$

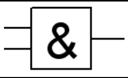
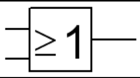
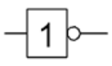
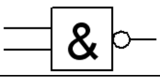
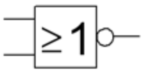
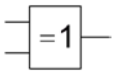
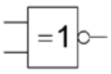
4.10

Show that

a) $\overline{x_2 \oplus x_1} = \overline{x_2} \oplus x_1 = x_2 \oplus \overline{x_1}$ b) $x_2 \oplus x_1 = \overline{x_2} \oplus \overline{x_1}$

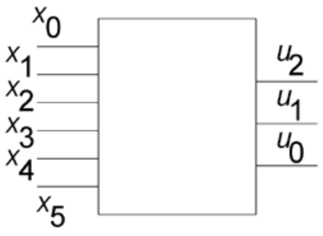
4.11

The figure shows the international standard gate symbols. America's dominance in the semiconductor area implies that one must also be familiar with the Americancan symbols. Name the gates and draw the corresponding American gate symbols.

4.12

A combinatorial network with six inputs x_5, x_4, x_3, x_2, x_1, x_0 and three outputs u_2, u_1, u_0, is described with text as follows:



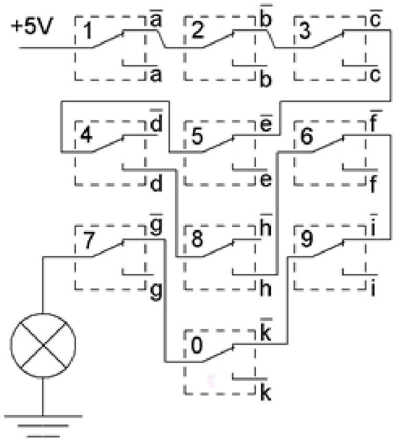
- $u_0 = 1$ if and only if ”either both x_0 and x_2 is 0 or x_4 and x_5 are different”
- $u_1 = 1$ if and only if ”x_0 and x_1 are the same and x_5 are the inverse of x_2”
- $u_2 = 0$ if and only if ”x_0 is 1 and some of x_1 ... x_5 is 0”

Describe the network with Boolean algebra and operations AND OR NOT XOR instead.

Truth table, SoP and PoS -form, Complete logic

5.1

The figure shows a simple "code lock" with 10 change-over contacts. The lamp will light for a certain combination of simultaneously pressed contacts,.



a) Which combination?

b) Specify the logical function for light up the lamp. Variables names stands in the figure (*a* … *k*).

f =

5.2

A logic function has the following State Table:

a b c	*f*
0 0 0	1
0 0 1	0
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	0
1 1 1	1

Specify the function of PoS-normal form (product of sums):
f(*a*, *b*, *c*) =

Specify the function of SoP-normal form (sum of products):
f(*a*, *b*, *c*) =

5.3

A minimal function is specified on the SoP form (sum of products).
Type the same function as SoP normal form, and as PoS normal form.

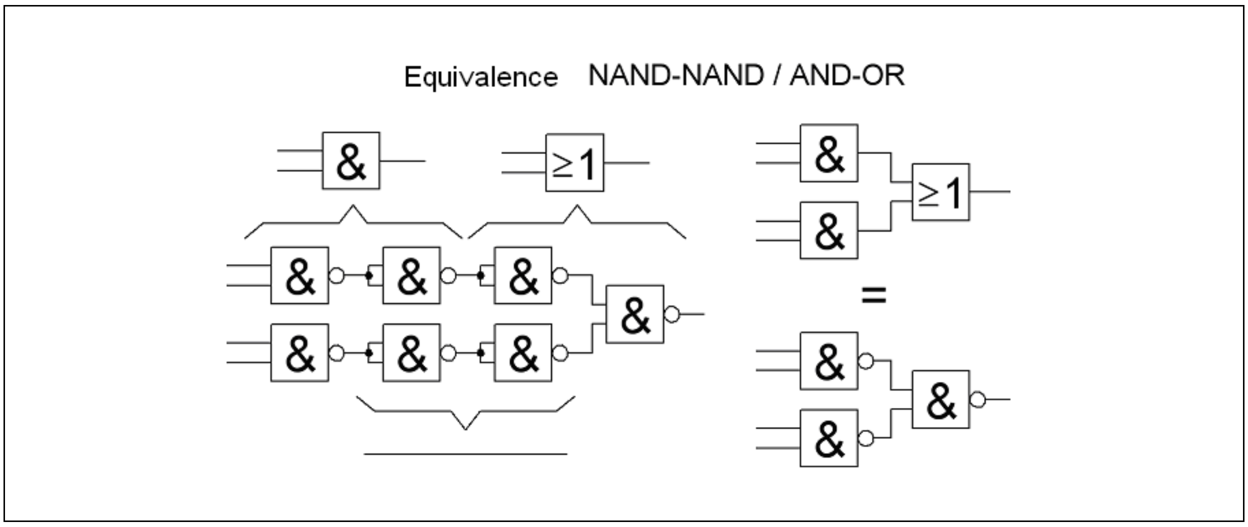
$$f(x,y,z) = x\bar{y} + y\bar{z} + \bar{x}z$$

5.4

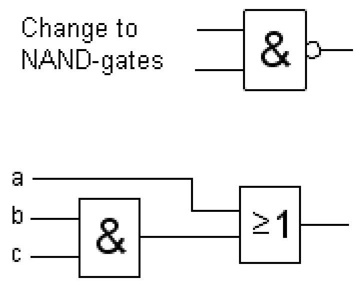
A function is denoted as a mixture of products and sums.
Type the same function as SoP normal, and as PoS normal.

$$f(x,y,z) = (x+\bar{y})\left(xyz + \bar{y}(x+z)\right) + xy\bar{z}(x+\bar{x}y)$$

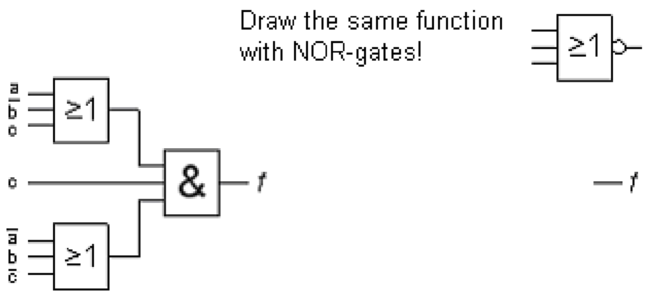
Equivalence AND-OR / NAND-NAND and OR-AND / NOR-NOR



5.5
Draw this AND/OR network as a NAND/NAND network.



5.6
Draw this OR /AND networks as a NOR/NOR network.



5.7
a) Write the truth table for a circuit with four inputs that define the even parity; ie circuit output is "1" when an even number of inputs are simultaneously "1".

b) Implement this function with as few NOR gates as possible.

The Karnaugh map

6.1

Make the best possible groups in the Karnaugh map. Enter the minimized function at the SoP form.

f =

ab \ cd	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	1	1	0
10	1	0	0	1

6.2

Make the best possible groups in the Karnaugh map. Enter the minimized function at the SoP form.

f =

ab \ cd	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	1	1	1
10	1	0	0	1

6.3

Place this function in the Karnaugh map.

$$f = \bar{a}b\bar{c} + ab\bar{c} + bc\bar{d}$$

Try to find better groups. Enter the minimized function at the SoP form.

f =

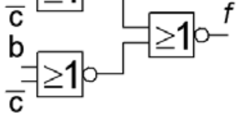
ab \ cd	00	01	11	10
00				
01				
11				
10				

6.4

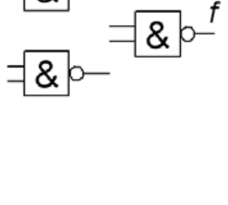
The top of the figure to the right is a NOR-NOR network.

Analyse this network and insert the truth table in the Karnaugh map.

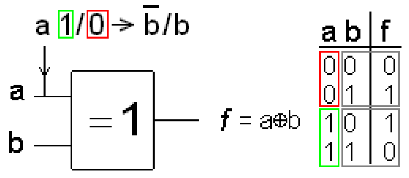
Make groups in yhe Karnaugh map and realize the function with NAND gates at the bottom of the figure.
Variables *a b* and *c* are available in both normal and inverted form.



a \ bc	00	01	11	10
0	0	1	3	2
1	4	5	7	6



PLD circuits often have an XOR gate at the output so that one is to be able to invert the function. One can then choose to group together 0s or 1s after what is the most advantageous.



6.5

A function with four variables are defined with minterms in the SoP form. Use Karnaugh map to minimize the function. Also minimize the function's inverse.

$$f(x_3, x_2, x_1, x_0) = \sum m(0, 2, 4, 8, 10, 12) \quad f = ? \quad \overline{f} = ?$$

6.6

A function with four variables are defined with minterms in the PoS form. Use Karnaugh map to minimize the function. Also minimize the function's inverse.

$$f(x_3, x_2, x_1, x_0) = \prod M(0, 1, 4, 5, 10, 11, 14, 15) \quad f = ? \quad \overline{f} = ?$$

6.7

A function with four variables are defined with mintermer the SoP form. Use Karnaugh map to minimize the function. Also minimize the function inverse.

$$f(x_3, x_2, x_1, x_0) = \sum m(0, 2, 3, 4, 6, 7, 8, 9, 10, 12, 13, 14) \quad f = ? \quad \overline{f} = ?$$

6.8

Sometimes the problem is such that certain input combinations are "impossible" and therefore can not occur. Such minterms (or maxterms) are denotet with d ("do not care") and could be used as ones or zeros depending on what works best to get as big as possible groups.

$$f(x_3, x_2, x_1, x_0) = \sum m(3, 5, 7, 11) + d(6, 15) \quad f = ? \quad \overline{f} = ?$$

6.9

$$f(x_3, x_2, x_1, x_0) = \sum m(1, 4, 5) + d(2, 3, 6, 7, 8, 9, 12, 13) \quad f = ? \quad \overline{f} = ?$$

6.10

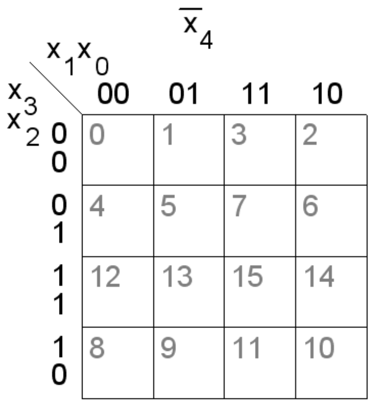
A function with **five** variables is defined as

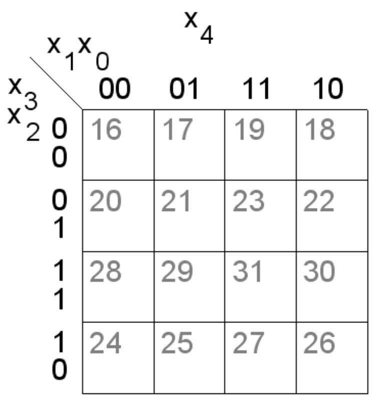
$$f(x_4, x_3, x_2, x_1, x_0) = \sum m(9, 11, 12, 13, 14, 15, 16, 18, 24, 25, 26, 27)$$

see the completed truth table.
Use Karnaugh map method for minimizing the function. Also minimize the function’s inverse.

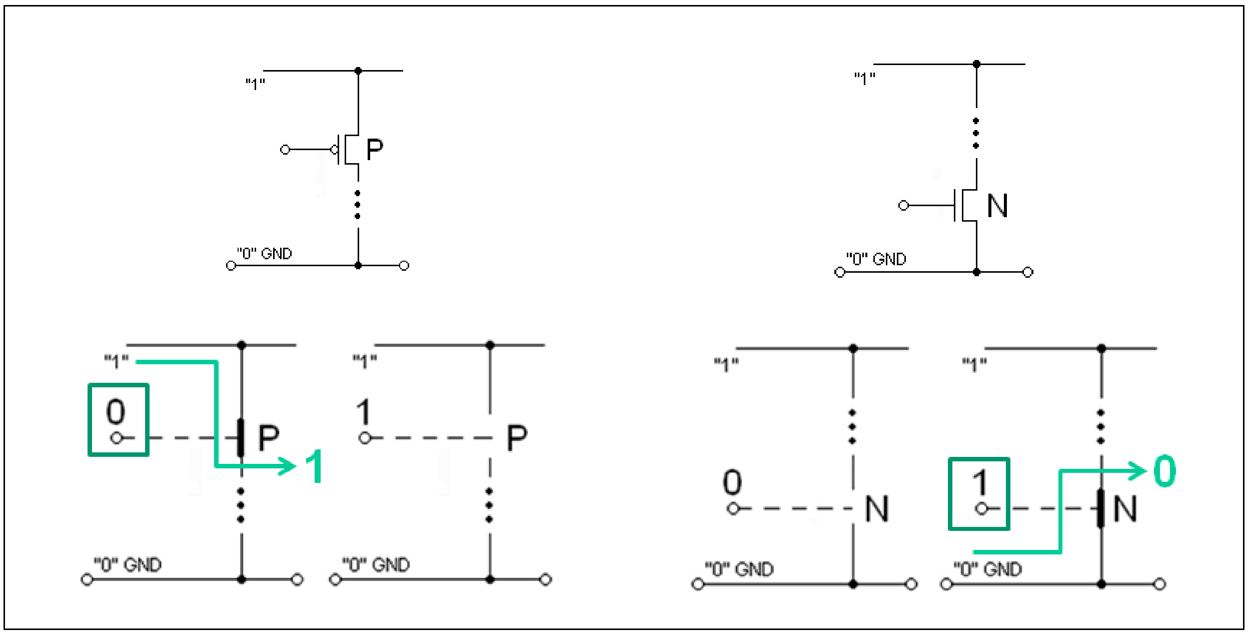
$f(x_4, x_3, x_2, x_1, x_0) \quad f = ? \quad \overline{f} = ?$

	x_4	x_3	x_2	x_1	x_0	f		x_4	x_3	x_2	x_1	x_0	f
0	0	0	0	0	0	0	16	1	0	0	0	0	1
1	0	0	0	0	1	0	17	1	0	0	0	1	0
2	0	0	0	1	0	0	18	1	0	0	1	0	1
3	0	0	0	1	1	0	19	1	0	0	1	1	0
4	0	0	1	0	0	0	20	1	0	1	0	0	0
5	0	0	1	0	1	0	21	1	0	1	0	1	0
6	0	0	1	1	0	0	22	1	0	1	1	0	0
7	0	0	1	1	1	0	23	1	0	1	1	1	0
8	0	1	0	0	0	0	24	1	1	0	0	0	1
9	0	1	0	0	1	1	25	1	1	0	0	1	1
10	0	1	0	1	0	0	26	1	1	0	1	0	1
11	0	1	0	1	1	1	27	1	1	0	1	1	1
12	0	1	1	0	0	1	28	1	1	1	0	0	0
13	0	1	1	0	1	1	29	1	1	1	0	1	0
14	0	1	1	1	0	1	30	1	1	1	1	0	0
15	0	1	1	1	1	1	31	1	1	1	1	1	0



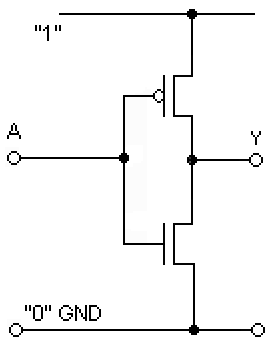


MOS-transistors and digital circuits



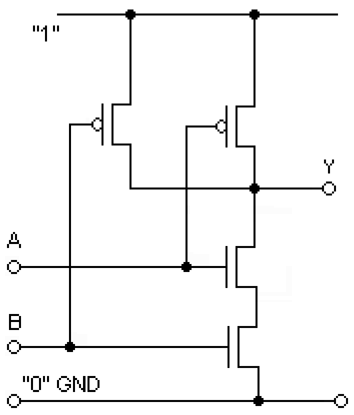
7.1

Identify transistors behavior, and write the truth table for *Y*(*A*).
Which logic function is it?



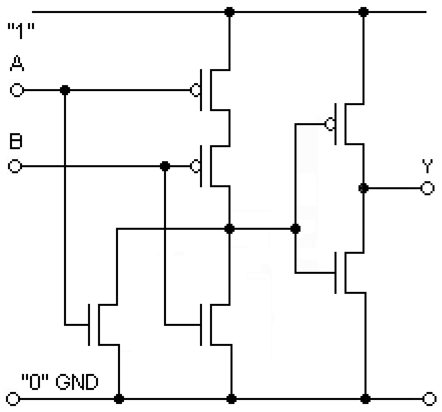
7.2

Identify transistors behavior, and write the truth table for *Y*(*A*,*B*).
Which logic function is it?



7.3

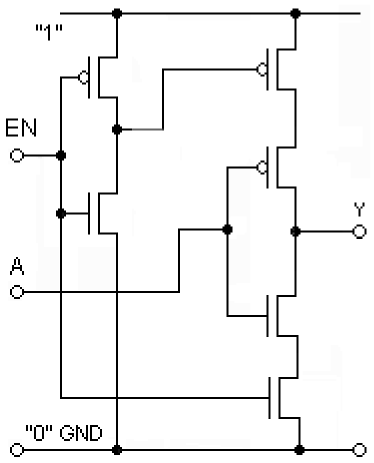
Identify transistors behavior, and write the truth table for *Y*(*A*,*B*). Which logic function is it?



7.4

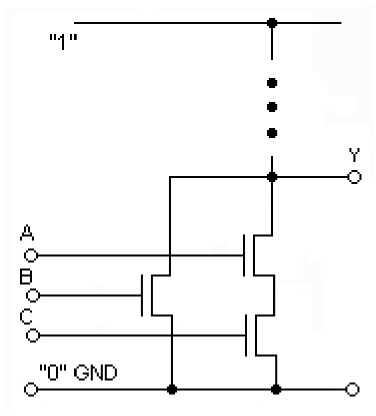
Study the circuit and describe the function. What role does the signal *EN* have? What relationship holds between *Y* and *A*? *Y*(A).

How many "states" can the output have?



7.5

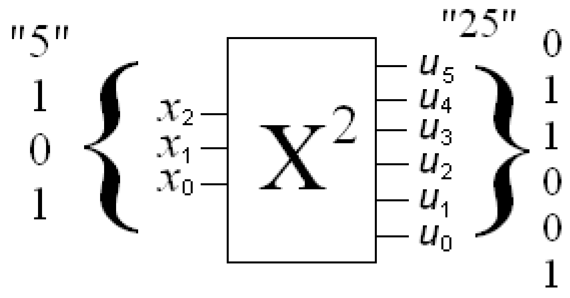
The figure shows one half of a CMOS circuit. Draw the other half, which contains the PMOS transistors. Enter the logical function *Y*(*A*,*B*,*C*).



Combinational circuits

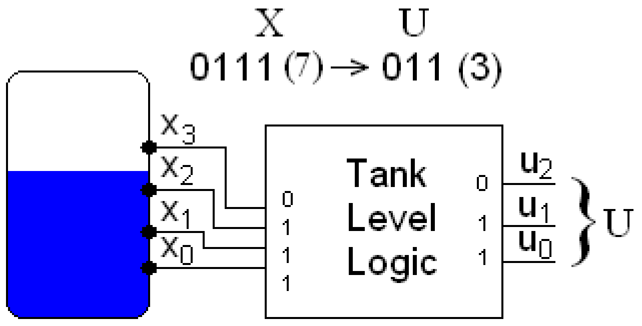
8.1.

Derive the Boolean expressions to minimized SoP form of a combinatorial network that converts a three-bit binary coded number X (x_2, x_1, x_0) to a binary coded six bit number U (u_5, u_4, u_3, u_2, u_1, u_0) which is equal to the square of the input $U = X^2$. Use Karnaugh maps.



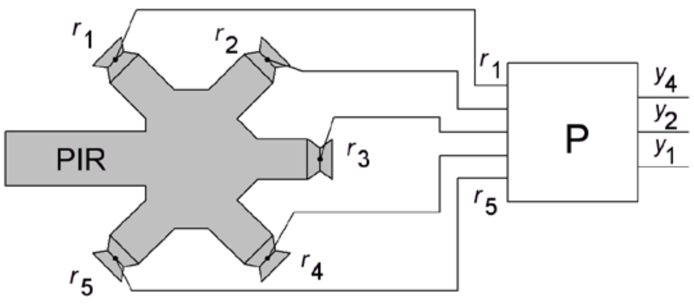
8.2

A monitoring system for a water tank consists of four level sensors x_3, x_2, x_1, x_0. The signals from these forms a binary four-bit number X . A logic circuit ”Tank Level Logic” transcodes X to a three bit number U (u_2, u_1, u_0) which presents the level as a binary number between 0 and 4.
Construct the logic network. Derive the Boolean expressions on minimized the SoP form. Take advantage that many of input signal combinations can *never* occur! The input variables are available in both inverted and not inverted form from the level sensors.
Use AND_OR to NAND_NAND equivalence to produce a logic network using only NAND gates.



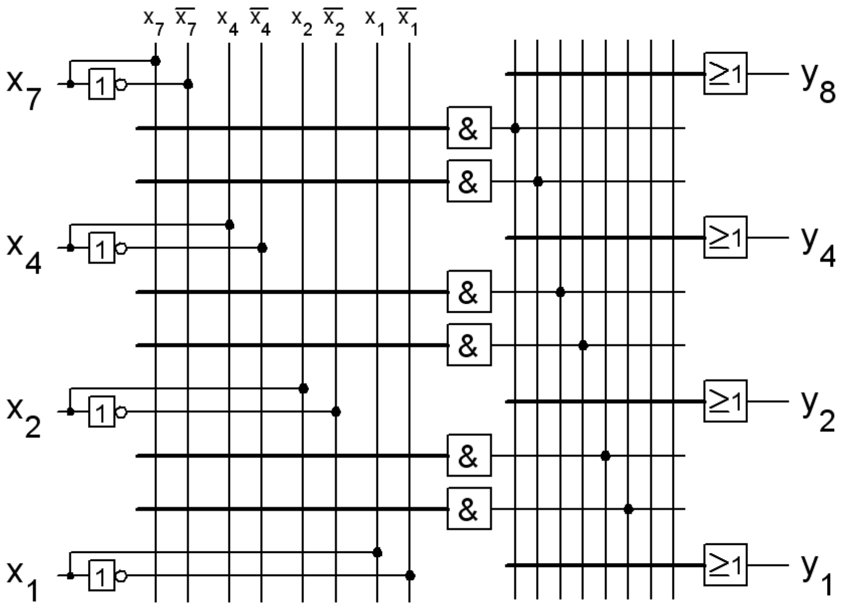
8.3

A pier at an airport has five connecting Gates (ramp). The Gates are numbered 1...5. At each Gate there are a sensor with the output signal $r_i = 0$ if an aircraft is connected to the Gate, otherwise 1. A combinatorial circuit, P, helps air traffic controller to direct arriving aircraft to available Gates. The circuit P has input signals r_1, r_2, r_3, r_4, r_5 and output signals y_4, y_2, y_1. The combination of the oututs y_4, y_2, y_1 should in binary give the number of the Gate with the maximum sequence number that is vacant. If no Gate is free the number (y_4, y_2, y_1) = (1, 1, 1) is used. Minimize each output separately.



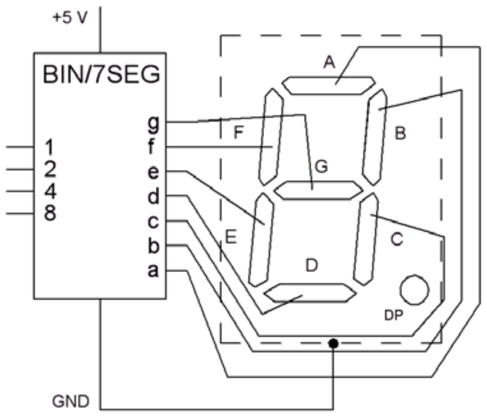
8.4

The decimal digits 0 to 9 can be encoded in the so-called 7421 code. It is a balanced binary position code with weights 7, 4, 2, and 1, where two combinations of weighted bits can provide the same value, the code word with the minimum number of ones is selected.
(7421-code has the property that it encodes the digits 0 to 9 with minimal number of ones, a total of 14 st).
Design a circuit that translates from the 7421 Code to the more conventional BCD code (code 8421).
Use a PLD circuit of the AND-OR type. Both the AND plane and the OR plane can be programmed individually.
Draw a cross in the figure below to show the programmed connections to be made. The Gates inputs are drawn in a "simplified" way.


x_7 $\overline{x_7}$ x_4 $\overline{x_4}$ x_2 $\overline{x_2}$ x_1 $\overline{x_1}$
x_7
x_4
x_2
x_1
1
&
≥1
y_8
y_4
y_2
y_1

8.5

A 7-segment encoder decodes a binary 4-bit number to the corresponding segment image for the numbers 0 ... 9 (or hexadecimal 0 ... F).
Set up the truth table, and enter a minimized logical circuit for one of the segments for example segment "G".

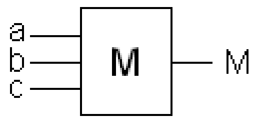


8.6

Show how a 4-1 multiplexer can be used as a function generator and as a such generate the OR function.

8.7

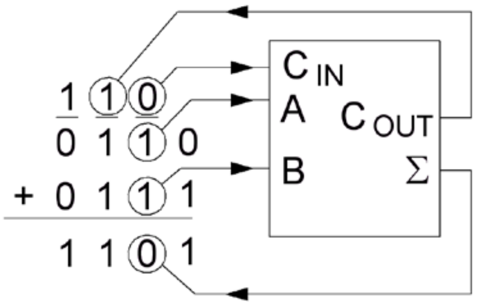
A majority gate adopt at output the same value as the majority of the inputs. The gate can for example be used in fault-tolerant logic, or for image processing circuits.

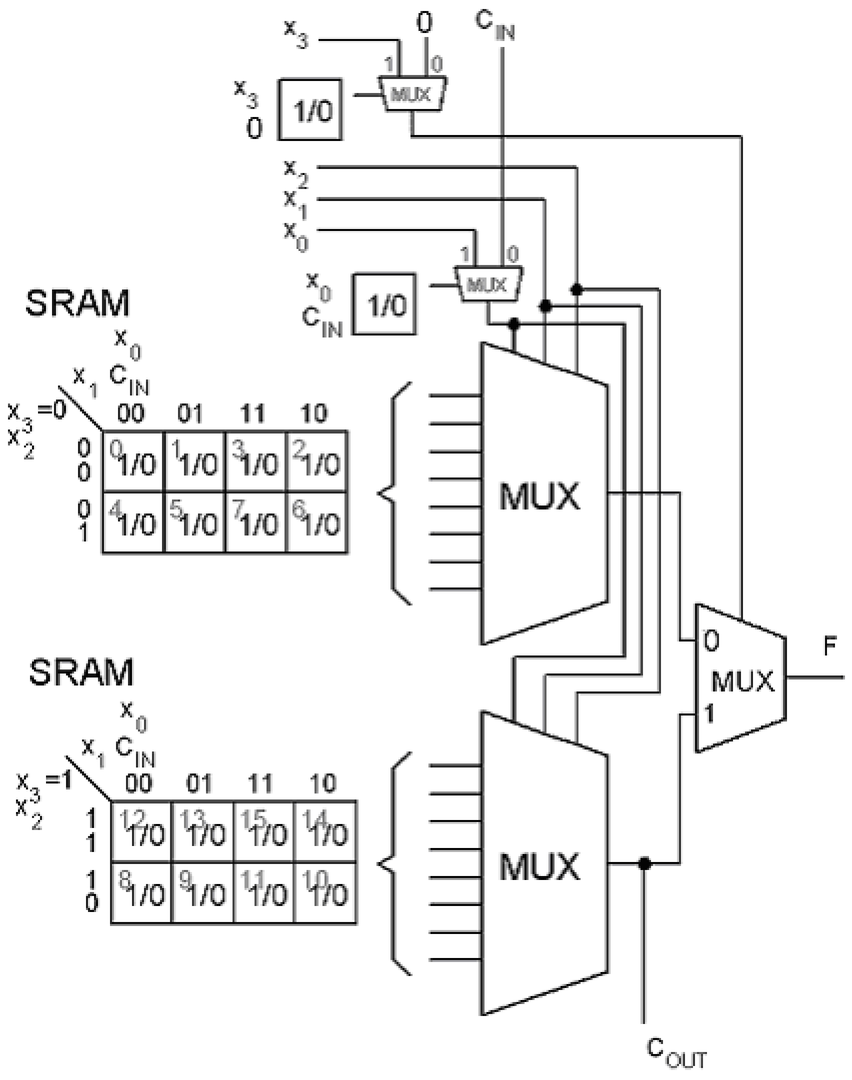


a) Derive the gate's truth table and minimize the function with Karnaugh maps. Realize the function with AND-OR gates.
b) Realize the majority gate with a 8:1 MUX.
c) Use Shannon decomposition and realize the majority gate with a 2:1 MUX and gates.
d) Realize the majority gate with only 2:1 MUXes.

8.8

Derive the Full Adder truth table. Show how a full adder is implemented in an FPGA chip. Logic elements in a FPGA is able to cascade C_{OUT} and C_{IN} between "stages". Show the contents of the SRAM cells (the LUT, Lookup Table).



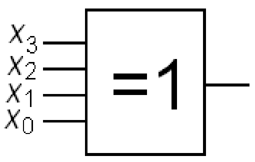


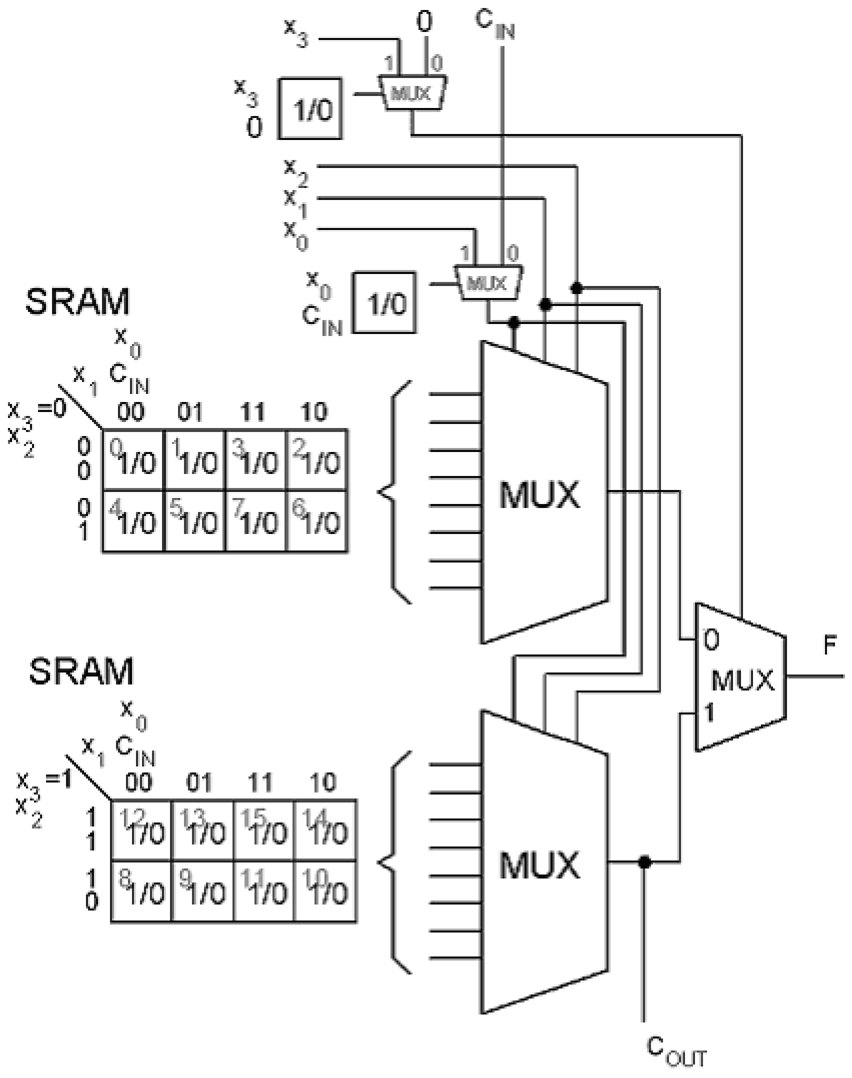
8.9

Show how one four-input XOR gate (XOR) are realized in a FPGA circuit. Show the contents of the SRAM cells (the LUT, Lookup Table).

XOR

$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	^{0}0	^{1}1	^{3}0	^{2}1
01	^{4}1	^{5}0	^{7}1	^{6}0
11	^{12}0	^{13}1	^{15}0	^{14}1
10	^{8}1	^{9}0	^{11}1	^{10}0





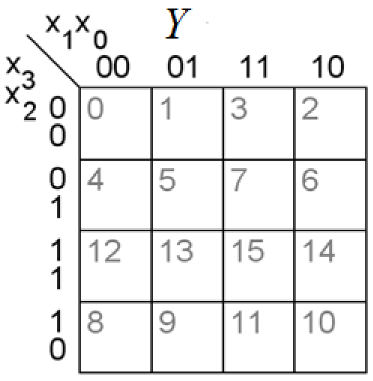
8.10

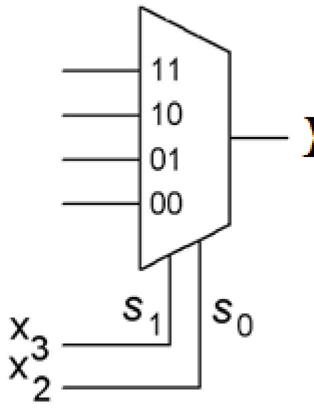
The Boolean function *Y* of four variables $x_3\, x_2\, x_1\, x_0$ is defined by it’s truth table.

a) Use the Karnaugh map to construct a minimal circuit for the function (use ”–” as don’t care). Choose any gates..

b) Realize the function *Y* with a 4:1 multiplexer and (any) gates. Use x_3 and x_2 as the multiplexer select signals.

	$x_3x_2x_1x_0$	Y		$x_3x_2x_1x_0$	Y
0	0000	–	8	1000	–
1	0001	–	9	1001	–
2	0010	1	10	1010	1
3	0011	0	11	1011	0
4	0100	0	12	1100	0
5	0101	1	13	1101	1
6	0110	0	14	1110	1
7	0111	1	15	1111	0



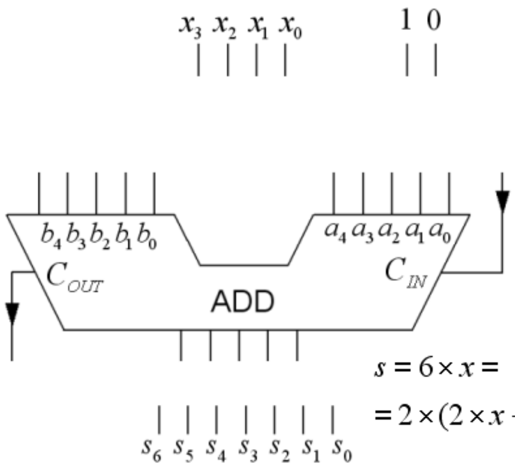


8.11

A four bit number *x* ($x_3x_2x_1x_0$) is to be multiplied by the constant 6.
The number *x* is fed into a five bit adder which is configured for the operation $6{\cdot}x = 2{\cdot}(2{\cdot}x + 1{\cdot}x)$.

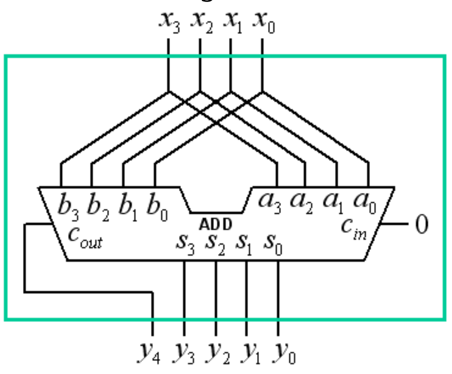
a) Draw how the adder has to be configured. Except the four bits in *x*, constants with the values 0 and 1 are also available.

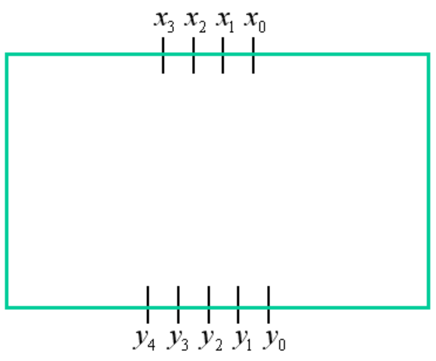
b) Which is the greatest binary number *s* ($s_6s_5s_4s_3s_2s_1s_0$) that can appear on the output when the circuit is configured for this operation? Answer with a binary number.



$$s = 6 \times x =$$
$$= 2 \times (2 \times x + 1 \times x)$$

8.12
A four bit unsigned integer *x* ($x_3x_2x_1x_0$) is connected to an 4-bit adder as in the figure. The result is a 5-bit number *y* ($y_4y_3y_2y_1y_0$). Draw the figure to the right how the same results can be obtained *without using the adder*. There are also bits with the values 0 and 1 if needed.

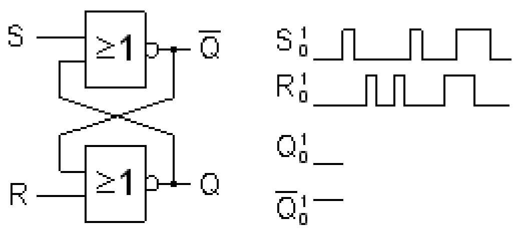




Sequential circuits, latches and clocked flip-flops

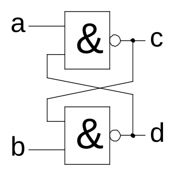
9.1

Complete the timing diagram for the output signals Q and $\overline{Q}$. The distance between the pulses is much longer than the gate delay.
(What is locking input signal for the NOR gates)



9.2

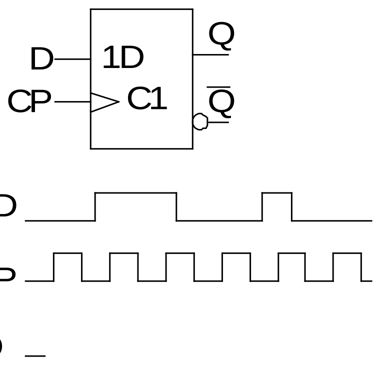
You probably know the latch to the right. The usual names are replaced with **a b c d**. Fill in the characteristic table.



a b	c d
0 0	
0 1	
1 0	
1 1	

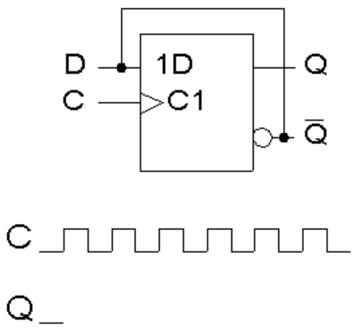
9.3

Draw in the timing diagram the output **Q**, for the D-flip-flop.



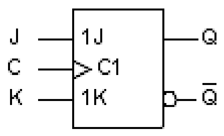
9.4

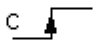
Draw Q in this timing diagram.



9.5

JK flip-flop was an older type of "universal flip-flop". Show how it can be used as a T flip-flop and a D flip-flop.

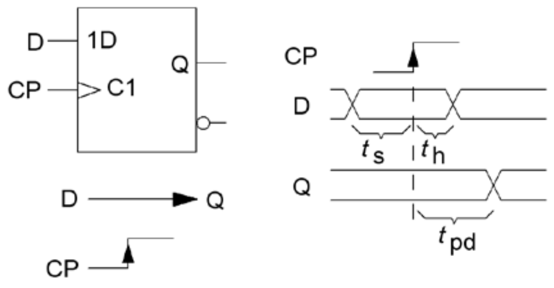




J	K	Q
0	0	Q, Same
0	1	0
1	0	1
1	1	$\overline{Q}$, Toggle

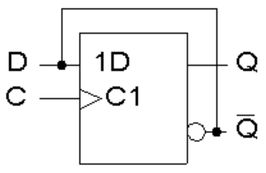
Flip-Flop Timing Parameters.
The fip-flop is loaded with data at the positive edge of the clockpulse, but data must be stable t_s before the clock edge and even the time t_h after.
The data can be found at output after the time t_{pd}.
(t_{pd} can be different for 0→1 respective 1→0 transitions).

If these times are not respected the flip-flop functioning becomes uncertain.



9.6

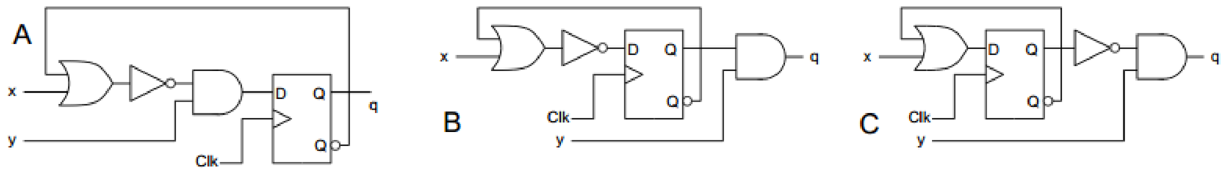
What is the maximum clock frequency that can be used to the circuit in the figure without risking malfunction?
Suppose
t_s = 20 ns t_h = 5 ns t_{pd} = 30 ns



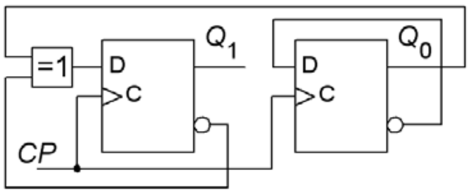
9.7

The figure shows three different state machines. Specify the state machine (A, B or C) that can operate at the highest clock speed. Highlight the critical path (the path that limits clock frequency) in this figure and calculate the period time for the clock signal Clk

t_{AND} = 0,4 ns, t_{OR} = 0,4 ns, t_{NOT} = 0,1 ns, t_{setup} = 0.3 ns, t_{dq} = 0,4ns



9.8

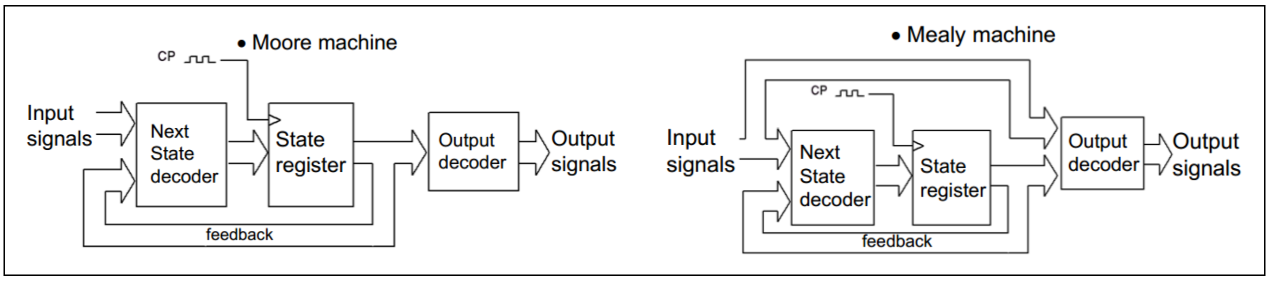


For the flip-flops: setup time t_{su} = 4 ns, delay time for the flip-flop outputs t_{pdQ} = 3 ns. The XOR-gate has delay time t_{pdXOR} = 5 ns.

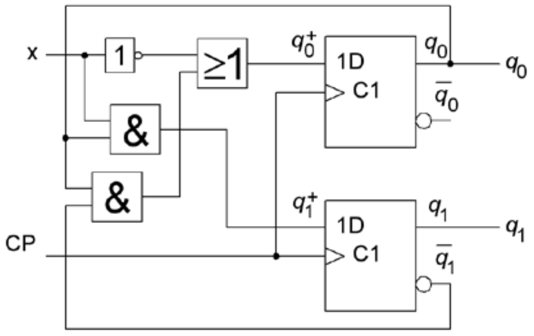
a) How long does it needs to be between the clock pulses T_{CP} > ?, for the counter function to be safe?

b) What value must the hold time have t_h for this circuit to work? t_h < ? ns.

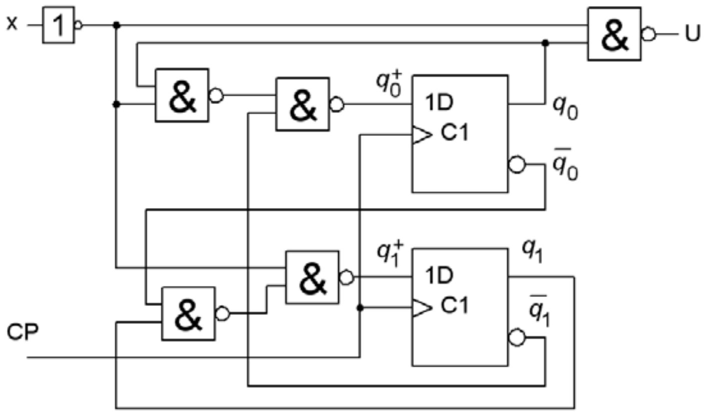
Sequential circuits



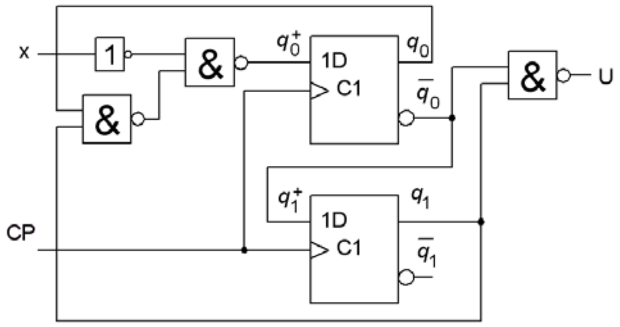
10.1
Determine the state diagram and state table for the sequence circuit. Which of the models Mealy or Moore fits the circuit?



10.2
Determine the state diagram and state table for the sequence circuit. Which of the models Mealy or Moore fits the circuit?

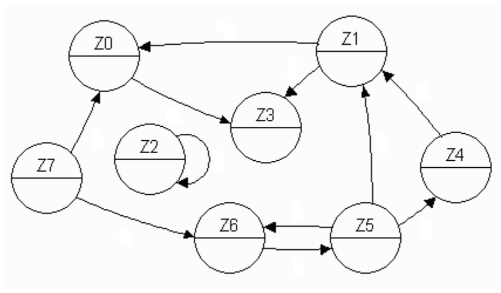


10.3
Determine the state diagram and state table for the sequence circuit. Which of the models Mealy or Moore fits the circuit?



(10.4)
Is there any stopping condition, loss condition or isolated states in the state diagram?

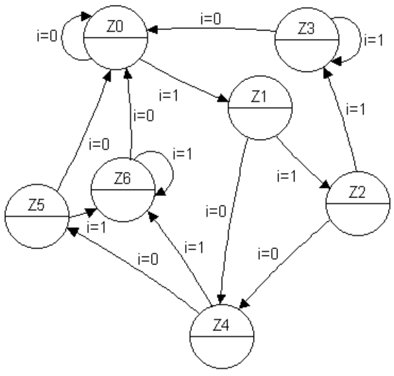
Stopping condition:
Loss condition:
• Isolated states:

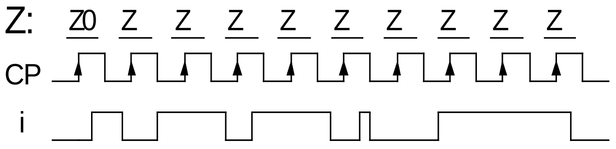


10.5
To the right is a state diagram for a Moore machine.
(it will detect a double tap).
A monkey accidentally get hold of the push-button input signal, and then presses according to the timing diagram below.

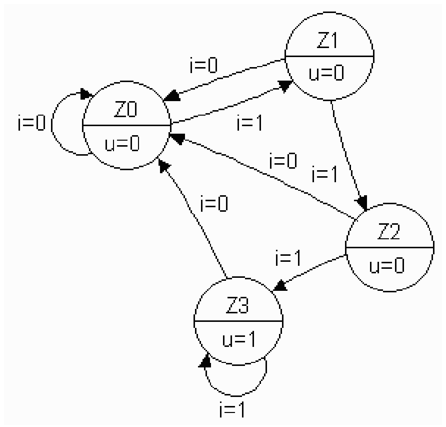
The Moore-machine has flip-flops that are triggered by the positive edge of the clock. Suppose that the initial state is Z0.

Fill in the states the machine enters..

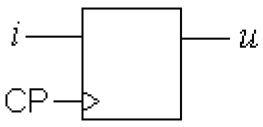




10.6
Construct a Moore machine which requires that the input signal is equal to one (i = 1) during three successive clock pulse interval, for the output to be one (u = 1).
As soon as the input signal becomes zero (i = 0) during a clock pulse interval, the circuit output should return to zero (u = 0). See the state diagram.
Choose Gray code for state encoding. (Z0=00, Z1=01, Z2=11, Z3=10). Use D-flip-flops and AND-OR gates.



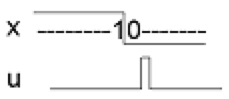
(This is a safety circuit to prevent "false alarms”)

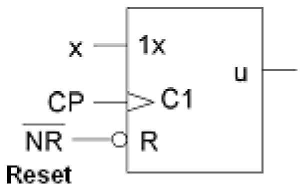


10.7

Construct a sequential circuit that detects when the input signal x has a transition $1 \rightarrow 0$ and then has the output $u = 1$ in the following clock-pulse interval, and then being 0 for the rest of the sequence.

The circuit should be able to "reset" with an asynchronous reset pulse (NR active low), so that it monitors the input signal again.





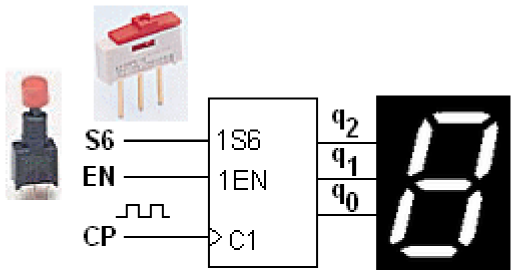
a) Draw state a diagram of a Moore machine type for the sequence network.

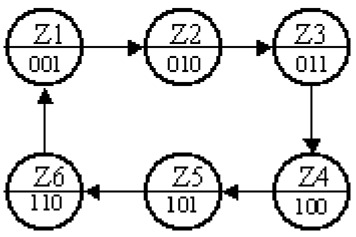
b) Derive the Boolean expressions for the next state and the output for three different state encoding::

1) ”Binarycode”
2) ”Graycode”
3) ”One hot” code

c) Show how the reset signal is connected to NR D-flip-flops PRE and CLR inputs.

10.8





Design a counter that counts {… 1, 2, 3, 4, 5, 6, 1 …}. The counting sequence,, $q_2q_1q_0$, is to be shown with a 7-segment display, as a roll of the dice.

a) State the expressions for the next state.

b) Complete the expressions with a variable EN which will ” freeze” the state when EN = 0 (unpressed button). The counter shold count for EN = 1 (pressed button).

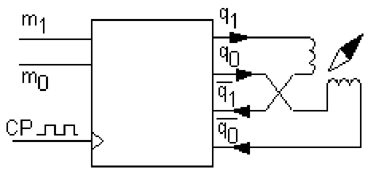
c) Complete the expressions with a variable S6 which forces the couter to state “6” when S6 = 1 (hidden button pressed). This is the “cheat-button”. S6 takes precedence over EN.

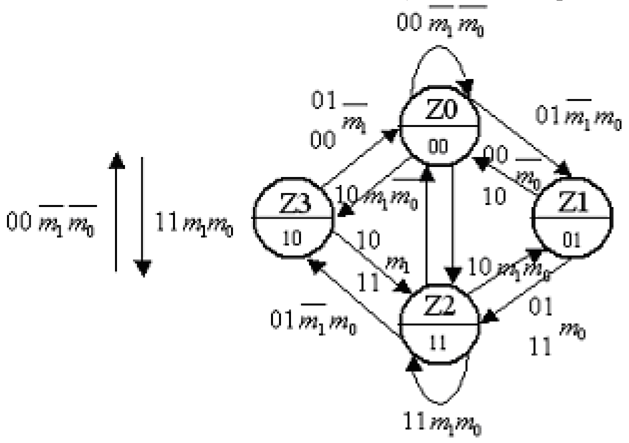
10.9

A stepper motor is a digital component that is driven by pulses. Stepper motors are usually connected to a counter counting Gray code.

The figures counter also has a mode-input, m_1m_0.

$m_1m_0 = 00 \rightarrow$ Reset (fixed position)
$m_1m_0 = 01 \rightarrow$ count up (cw)
$m_1m_0 = 10 \rightarrow$ count down (ccw)
$m_1m_0 = 11 \rightarrow$ Preset (another fixed position)





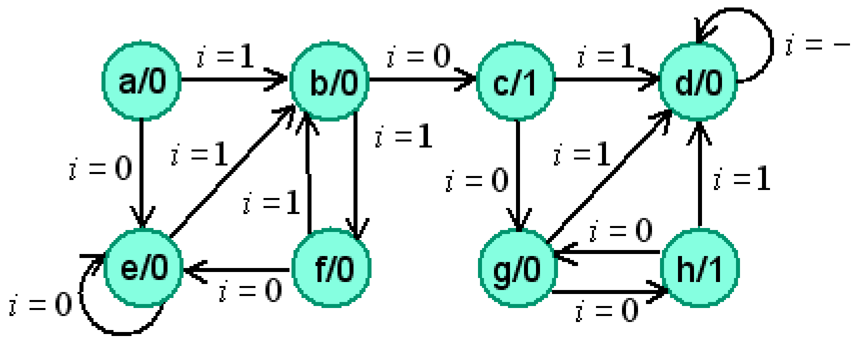
Sometimes you write boolean conditions instead of just the numbers at the arrows. In the figure, both the condition and numbers are used.

- Derive the minimized expressions of the counter next state decoder.

10.10

This state diagram applies to a synchronous sequential circuit.

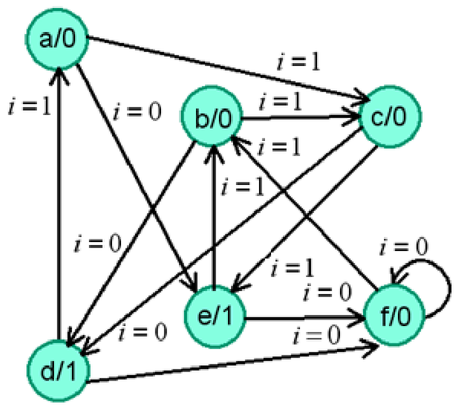
- Derive state table.
- Minimize the number of states.
- Derive the minimized state table
- Draw the minimized state diagram.



10.11

This state diagram applies to a synchronous sequential circuit.

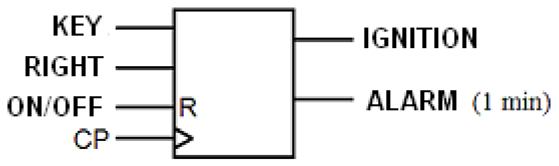
- Derive state table.
- Minimize the number of states.
- Derive the minimized state table
- Draw the minimized state diagram.



10.12

An engineering student builds a car thief alarm that is a synchronous Moore machine. The alarm gets its "security" of being secret and unique. To start the car you have to maneuver the car's controls in the following order:

1) Turn the ignition key (ignition on)
2) Set the turn signal to the right (right on)
3) Turn off the ignition key (ignition off)
4) Set the turn signal to neutral (right off)
5) Turn the ignition key (ignition on)



If, at any point in the list, you do the "wrong" thing you end up stucked in the an ALARM state. If you do everything right the car starts (= get stucked in the IGNITION coil on state).

Sequence circuit also has a "hidden" button that goes to the D-flip-flops reset, which means that the alarm can be switched ON / OFF.

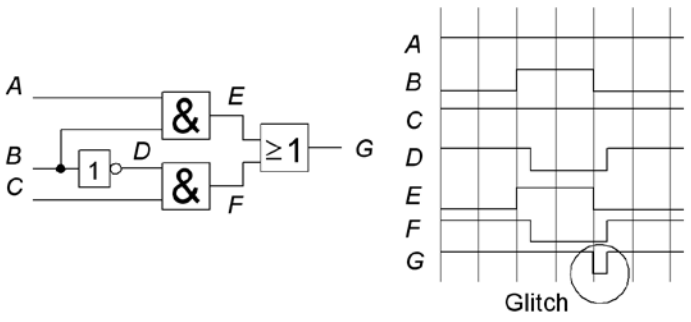
- Draw the state diagram for the alarm.

Asynchronous sequential circuits

11.1

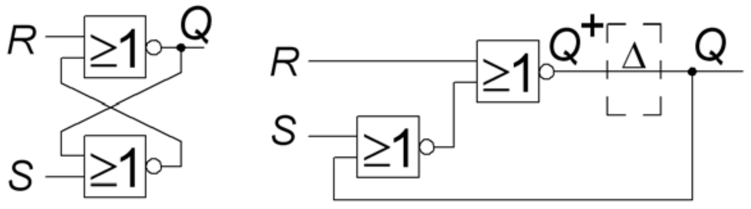
If the signals passes different amount of gate delays before they are combined at the utput, then momentary unwanted deviations from the truth table can occur, so-called "glitches".

Show in Karnaugh map how to avoid them.



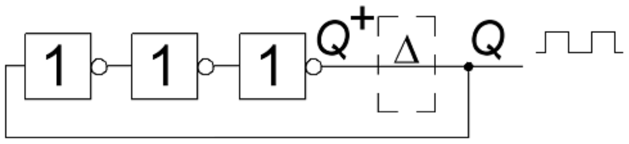
11.2

To the left in the figure, an SR latch has two gates with feed-back. To the right, the circuit is redrawn as a compatible "Moore" –machine. There is no clock signal, and no real state register. All gate delays that are present in the network is thought placed in the symbol Δ between Q^+ and Q getting a similar function to the flip-flops in a synchronous sequential circuit.
Analyze the SR-circuit in the same way as a Moore machine.



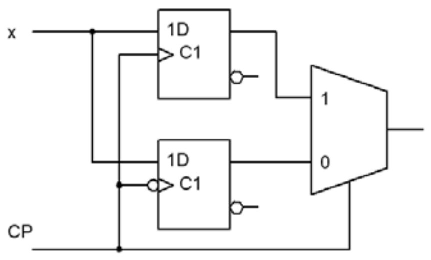
11.3

Show that there is an unstable network - an oscillator - if an odd number of inverters are connected in a circle.
Assume that the gate delay t_{pd} is 5 ns and that three gates are connected as in figure.
What value will the oscillation frequency get?



11.4

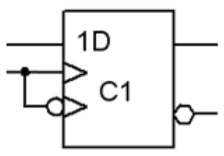
Analyze following circuit:
Draw a state diagram.
Consider the circuit as an asynchronous sequential circuit which has the clock pulse as one of the asynchronous inputs. What function does the circuit perform?



11.5

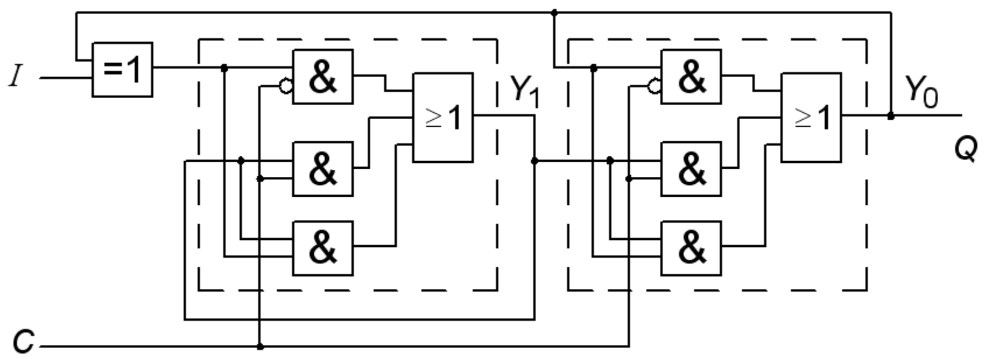
Construct an asynchronous state machine that functions as a double edge triggered D flip-flop (DETFF), wich means that the flip-flop will change value at both the positive and the negative edge of the clock.

a) Derive the FSM.
b) Construct the flow table and minimize it.
c) Assign states, transfer to Karnaugh maps and derive the Boolean expressions.
d) Draw the schematic for the circuit.



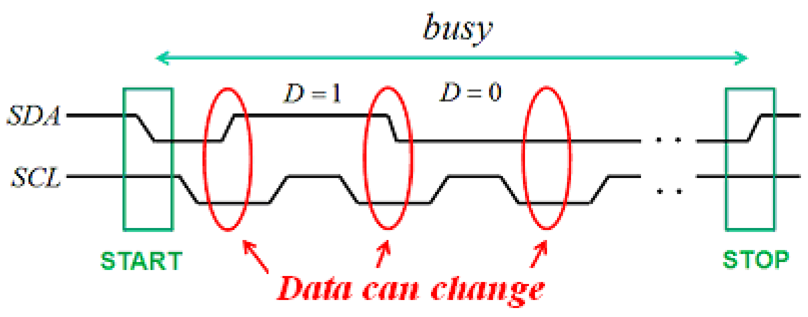
11.6

Analyze the following circuit.



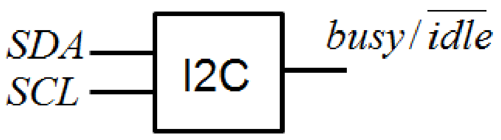
a) Derive the Boolean expressions for the state variables Y_1 and Y_0.
b) Derive the exitations table. Which function (dashed) are in the inner loops.
c) Derive the flow table, assign symbolic states and draw FSM.
d) Identify the function. Which flip-flop does this correspond to?

11.7



Data transfer between different chips in electronic equipments can be done with the so called I2C bus. It consists of two lines SDA and SCL. The figure above shows a principle diagram when a number of bits are transmitted. During transmissom Data *D* may only be changed when SCL=0.
Positive and negative SDA-edge when SCL=1 are used as **unique** start and stop signals for data transmission. (During transmission no such edges can occur).
(Before the stop pulse, the receiver can "acknowledge" the reception - in the figure we disregards this.)

In order to study the I2C data transfer wants to construct a Moore-equivalent asynchronous sequential circuits which provide output *busy* = 1 during the time from the start signal until the stop signal. When no data communication occurs is *busy* = 0.



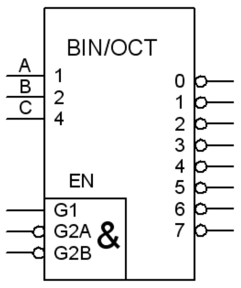
- Derive a primitive flowtable
- Minimize the flowtabele
- Choose code for state asignment
- derive the exitation table
(motivate that the design is free of critical race)
- Derive the minimized Boolean expressions
(motivate freedom of hazard)

Address decoding of memories and I/O circuits

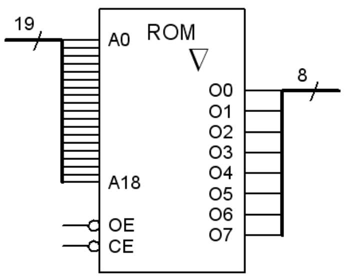
12.1

A dynamic RAM-memory consists of a number of 256Mbit memory chips organised as 32 M×8.
a) How many chips are needed for 256M×64?
b) How many chips are needed for 512M×72? (What can be the reason for the "strange" bit width "72"?)

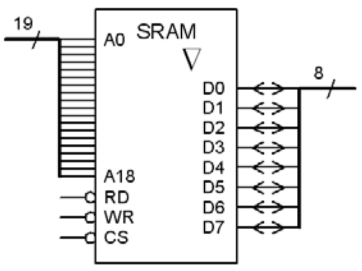
12.2



3:8-decoder

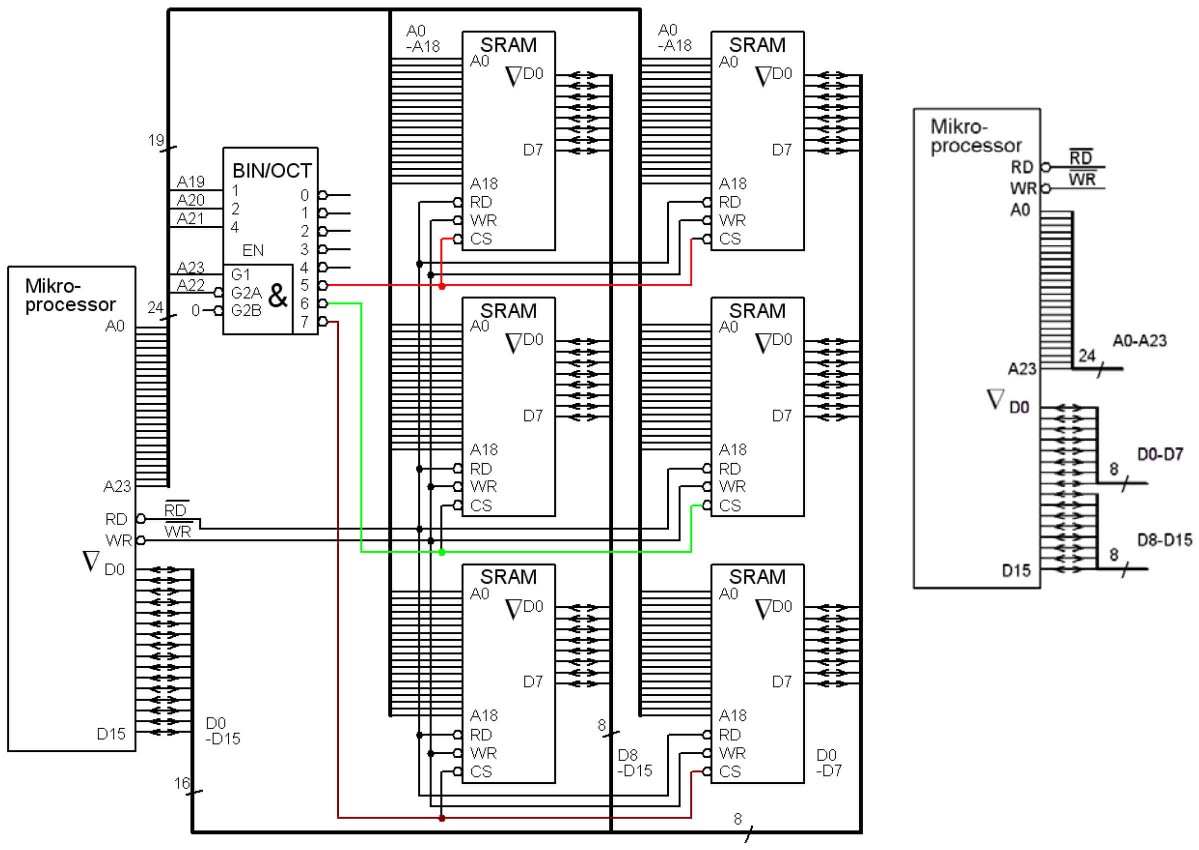


ROM 512k×8



SRAM 512k×8

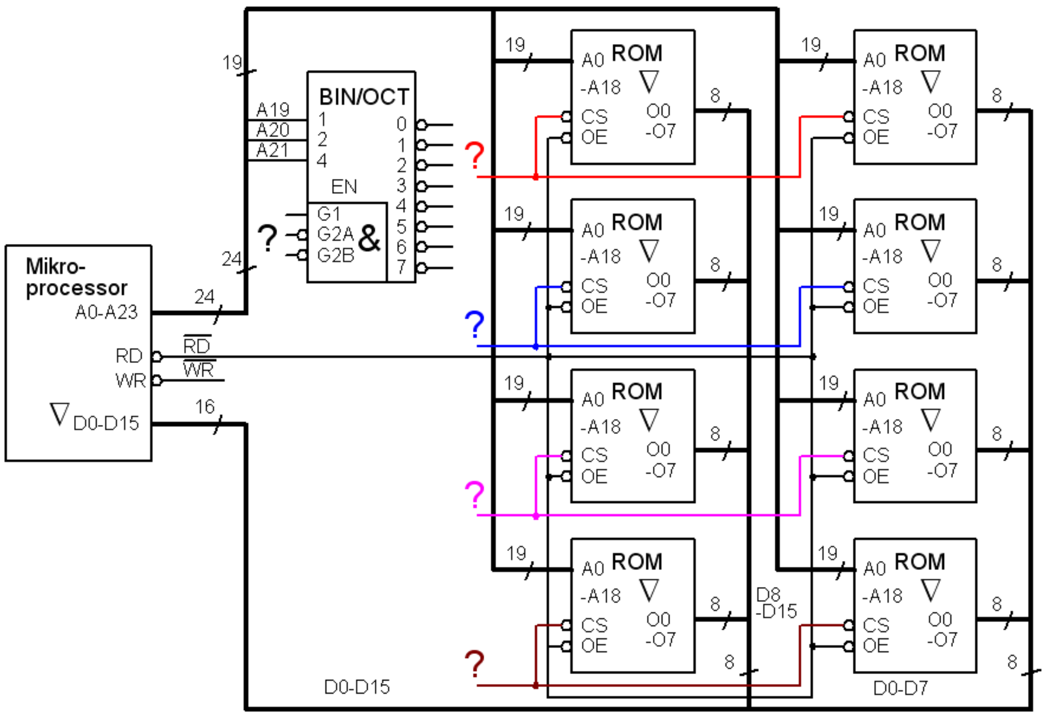
A certain 16 bit processor can address 24 bits. Memory Space is divided between ROM, SRAM and IO circuits. Address decoding is done using a 3:8-decoder.
a) How large is the RAM in the figure? What is the address range expressed in hexadecimal numbers?



b) How do you change the address range to 980000 – AFFFFF ?
c) How do you change the address range to 480000 – 5FFFFF ?

d) Typically a processor reads its first instruction from address 0 then there must be a read at that address. Assume that the ROM is 2M×16 bitar and that the address range is 000000 ... and beyond. ROM Chip is 512k×8.

- How many chips are needed?
- How should the decoder be connected?
- How shall the memorychips be connected?
- Specify address space of decoder outputs in hexadecimal. numbers.

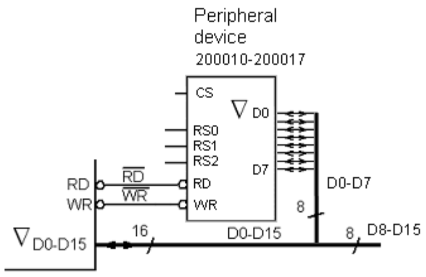


e) Which address space becomes available for SRAM and IO circuits?

12.3

Peripherals, I/O, are often connected to a CPU as if they were memory chips (though with only a few "memory cells"). Eg. a real time clock chip - keeps track of the time and date. It is controlled/read from the 8 built-in 8-bit registers.

a) Connect one eight registers memory-mapped peripheral device (I/O) to a CPU. The CPU has 16-bit data bus (we only use 8 bits), and a 24 bit address bus. Use a 3:8-decoder and if needed gates. The peripheral device must be connected so that the addresses are `0x200010` … `0x200017`. (Compare with the previous task).



b) What is incomplete decoding?

Solutions

Number systems and codes

1.1
Enter the corresponding binary number (base 2) to the following decimal numbers (base 10).
a) $9_{10} = (1{\cdot}2^3+0{\cdot}2^2+ 0{\cdot}2^1+1{\cdot}2^0) = (8+1) = 1001_2$
b) $12_{10} = (8+4) = 1100_2$ c) $71_{10} = (64+7 = 64+4+2+1) = 1000111_2$
d) $503_{10} = (512 – 9 = 511 – 8) = (111111111_2 - 1000_2) = 111110111_2$

1.2
Convert the following binary number to decimal.
a) $101101001_2 = (2^8+2^6+2^5+2^3+2^0 = 256+64+32+8+1) = 361_{10}$
b) $110100.010_2 = (2^5+2^4+2^2 + 2^{-2} = 32+16+4 + 0.25) = 52{,}25_{10}$

1.3
Convert the following binary numbers (base = 2) to the corresponding octal (base = 8) and hexadecimal (base = 16).
a) $01\ 1101_2 = 1D_{16} = (011\ 101_2) = 35_8$
b) $1000\ 1011_2 = 8B_{16} = (010\ 001\ 011_2) = 213_8$
c) $1\ 0011\ 0101_2 = 135_{16} = (100\ 110\ 101_2) = 465_8$
d) $1101\ 1110\ 1001\ 0001_2 = DE91_{16} = (001\ 101\ 111\ 010\ 010\ 001_2) = 157221_8$
e) $10\ 1001.001_2 = 29.2_{16} = (101\ 001\ .\ 001_2) = 51.1_8$

1.4
Convert the following hexadecimal numbers (base = 16) to the corresponding octal (base = 8).
a) $94D_{16} = (1001\ 0100\ 1101_2 = 100\ 101\ 001\ 101_2) = 4515_8$
b) $9E.7A_{16} = (1001\ 1110\ .\ 0111\ 1010_2 = 010\ 011\ 110\ .\ 011\ 110\ 100_2) = 236.364_8$

1.5
Convert the octal (base = 8) number 4515_8 to the corresponding hexadecimal number (bas=16).
$4515_8 = (100\ 101\ 001\ 101_2 = 1001\ 01\ 00\ 1101_2) = 94D_{16}$

1.6
Write the hexadecimal (base = 16) number BAC_{16} in decimal form (bas=10).
$BAC_{16} = (11{\cdot}16^2+10{\cdot}16^1+12{\cdot}16^0 = 11{\cdot}256+10{\cdot}16+12{\cdot}1) = 2988_{10}$

1.7
What characterizes Gray codes, and how can they be designed?
Gray codes have the distance "one" between codewords. There is never more than one bit at a time that changes in the transitions from one codeword to the next. If one wants to construct an N-bit Gray code can do this from the code for the N-1 bits. First, follow the N-1 bit code with "0" as the bit N, then continue to the next half of the N-1 code again but with code words in reverse order, with "1" as bit N. This is a "mirrored binary code."
This is how do you do 3-bit Gray code from 2-bit Gray code:
00 01 11 10 is a 2-bit Gray code. 000 001 011 010 is the code with "0" added as bit 3. 10 11 01 00 is the 2-bit code in reverse order. 110 111 101 100 is thereversed code with "1" added as bit 3. All together the 3-bit Gray code becomes:
000 001 011 010 110 111 101 100
This is not the only possible 3-bit Gray code, another possible code for the three bit are
000 010 011 001 101 111 110 100.
(In general it is the "reflected binary code" you mean when you talk about Gray code).

1.8
Type the following signed numbers with binary two's complement notation, $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.
a) $-23 = (+23_{10} = 0010111_2 \rightarrow -23_{10} = 1101000_2 + 1_2) = 1101001_2 = 105_{10}$

b) -1 = ($+1_{10} = 0000001_2 \rightarrow -1_{10} = 1111110_2 + 1_2$) = $1111111_2 = 127_{10}$
c) +38 = ($32_{10}+4_{10}+2_{10}$) = $0100110_2 = 38_{10}$
d) -64 = ($+64_{10} = 1000000_2$ is to big positive number! $-64_{10} = 0111111_2 + 1_2$) = $1000000_2 = 64_{10}$

1.9

Type the following signed numbers with binary one's complement notation, $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.
a) -23 = ($+23_{10} = 0010111_2 \rightarrow -23_{10} = 1101000_2$) = $1101000_2 = 104_{10}$
b) -1 $\rightarrow$ $1111110_2 = 126_{10}$ c) 38 = $0100110_2 = 38_{10}$ d) -0 $\rightarrow$ $1111111_2 = 127_{10}$

Digital arithmetic

2.1

Add by hand the following pair of binary numbers.
a) 110 + 010 b) 1110 + 1001 c) 11 0011.01 + 111.1 d) 0.1101 + 0.1110

a)
$$\begin{array}{ccccc}
 & {\scriptstyle 1} & {\scriptstyle 1} & & \\
 & & 1 & 1 & 0 \\
+ & & 0 & 1 & 0 \\ \hline
 & 1 & 0 & 0 & 0
\end{array}$$

b)
$$\begin{array}{cccccc}
 & {\scriptstyle 1} & & & & \\
 & & 1 & 1 & 1 & 0 \\
+ & & 1 & 0 & 0 & 1 \\ \hline
 & 1 & 0 & 1 & 1 & 1
\end{array}$$

c)
$$\begin{array}{cccccccc}
 & & & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & \\
 & 1 & 1 & 0 & 0 & 1 & 1.0 & 1 \\
+ & & & & 1 & 1 & 1.1 & \\ \hline
 & 1 & 1 & 1 & 0 & 1 & 0.1 & 1
\end{array}$$

d)
$$\begin{array}{cccccc}
 & {\scriptstyle 1} & {\scriptstyle 1} & & & \\
 & 0. & 1 & 1 & 0 & 1 \\
+ & 0. & 1 & 1 & 1 & 0 \\ \hline
 & 1. & 1 & 0 & 1 & 1
\end{array}$$

2.2

Add or subtract (addition with corresponding negative numbers) the following numbers. The numbers shall be represented as binary 4-bit numbers (Nibble) in two's complement form..
a) 1 + 2 b) 4 – 1 c) 7 – 8 d) -3 – 5
$-1_{10} = (+1_{10} = 0001_2 \rightarrow -1_{10} = 1110_2 + 1_2) = 1111_2$
$-8_{10} = (+8_{10} = 1000_2 \rightarrow -8_{10} = 0111_2 + 1_2) = 1000_2$
$-3_{10} = (+3_{10} = 0011_2 \rightarrow -3_{10} = 1100_2 + 1_2) = 1101_2$
$-5_{10} = (+5_{10} = 0101_2 \rightarrow -5_{10} = 1010_2 + 1_2) = 1011_2$

1+2=3 4-1=3 7-8=-1 -3-5=-8

a)
$$\begin{array}{c|ccccl}
 & 0 & 0 & 0 & 1 & =1 \\
+ & 0 & 0 & 1 & 0 & =2 \\ \hline
 & 0 & 0 & 1 & 1 & =3
\end{array}$$

b)
$$\begin{array}{c|ccccl}
{\scriptstyle 1} & {\scriptstyle 1} & & & & \\
 & 0 & 1 & 0 & 0 & =4 \\
+ & 1 & 1 & 1 & 1 & =-1 \\ \hline
\cancel{1} & 0 & 0 & 1 & 1 & =3
\end{array}$$

c)
$$\begin{array}{c|ccccl}
 & 0 & 1 & 1 & 1 & =7 \\
+ & 1 & 0 & 0 & 0 & =-8 \\ \hline
 & 1 & 1 & 1 & 1 & =-1
\end{array}$$

d)
$$\begin{array}{c|ccccl}
{\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & \\
 & 1 & 1 & 0 & 1 & =-3 \\
+ & 1 & 0 & 1 & 1 & =-5 \\ \hline
\cancel{1} & 1 & 0 & 0 & 0 & =-8
\end{array}$$

2.3

Multiply by hand following pairs of unsigned binary numbers.
a) 110·010 b) 1110·1001 c) 11 0011.01·111.1 d) 0.1101·0.1110

a) 110·010=(6·2=12)=1100
$$\begin{array}{cccccccl}
 & & & 1 & 1 & 0 & =6 \\
 & \times & & 0 & 1 & 0 & =2 \\ \hline
 & & & 0 & 0 & 0 & \\
 & & 1 & 1 & 0 & & \\
+ & 0 & 0 & 0 & & & \\ \hline
 & 0 & 1 & 1 & 0 & 0 & =12
\end{array}$$

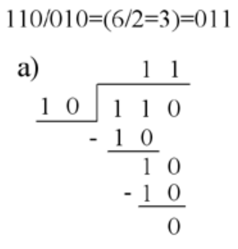
b) 1110·1001=1111110
$$\begin{array}{ccccccccl}
 & & & & 1 & 1 & 1 & 0 & =14 \\
 & & & \times & 1 & 0 & 0 & 1 & =9 \\ \hline
 & & & & 1 & 1 & 1 & 0 & \\
 & & & 0 & 0 & 0 & 0 & & \\
 & & 0 & 0 & 0 & 0 & & & \\
+ & 1 & 1 & 1 & 0 & & & & \\ \hline
 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & =126
\end{array}$$

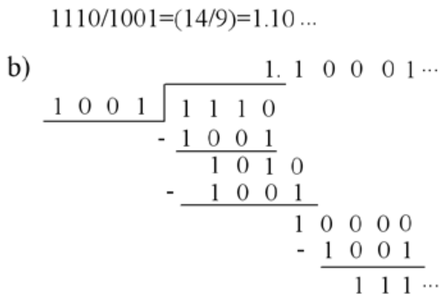
c) 110011.01·111.1=
=110000000.011
$$\begin{array}{r}
110011\boxed{01} \\
\times \quad 111\boxed{1} \\ \hline
11001101 \\
11001101 \\
11001101 \\
+11001101 \\ \hline
110000000\boxed{011}
\end{array}$$

d) 0.1101·0.1110=
=0.10110110
$$\begin{array}{r}
\boxed{1101} \\
\times \quad \boxed{1110} \\ \hline
0000 \\
1101 \\
1101 \\
+ \ 1101 \\ \hline
\boxed{10110110}
\end{array}$$

2.4

Divide by hand following pairs of unsigned binary numbers.
a) 110/010 b) 1110/1001
If the division is a **integer division** the answer of b) is the integer 1.

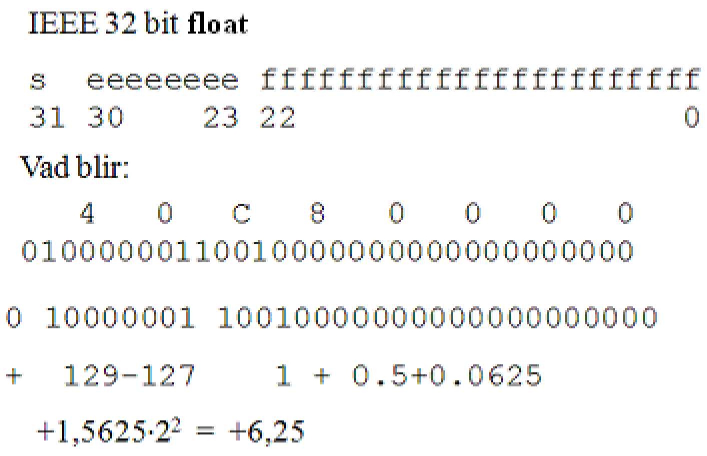




2.5

Assume that a 32-bit float is stored in a register: $40C80000_{16}$ what real decimal number is this?
32-bit floating point numbers are stored normalized as ”1.” One ”sign bit”, 8 bits for the 2-exponent (expressed as an exess-127), 23 bits for significand. Since all numbers are starting with "1" this “1” is not needed to be stored, it is implicitly understood.

Exess-127 means that number 127 is added to all exponents, they are therefore always stored as positive numbers. This has the advantage that floating point numbers can be sized as if they were integers!



IEEE 32 bit float

```
s  eeeeeeee fffffffffffffffffffffff
31 30     23 22                   0
```

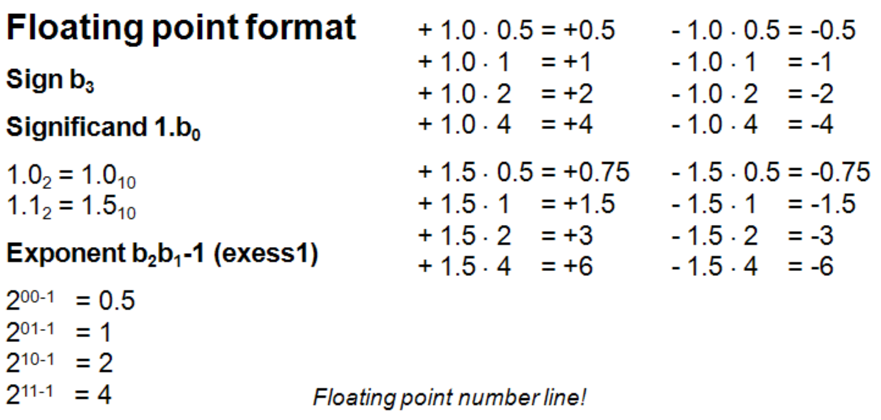
Vad blir:

```
   4    0    C    8    0    0    0    0
01000000110010000000000000000000

0 10000001 10010000000000000000000

+  129-127    1 + 0.5+0.0625
```

$+1{,}5625 \cdot 2^2 = +6{,}25$

2.6

a)



Floating point format

Sign b_3

Significand $1.b_0$

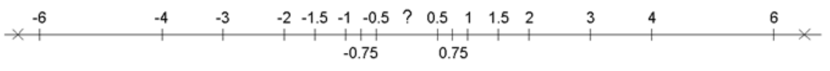
$1.0_2 = 1.0_{10}$
$1.1_2 = 1.5_{10}$

Exponent b_2b_1-1 (exess1)

$2^{00-1} = 0.5$
$2^{01-1} = 1$
$2^{10-1} = 2$
$2^{11-1} = 4$

$+1.0 \cdot 0.5 = +0.5$	$-1.0 \cdot 0.5 = -0.5$
$+1.0 \cdot 1 = +1$	$-1.0 \cdot 1 = -1$
$+1.0 \cdot 2 = +2$	$-1.0 \cdot 2 = -2$
$+1.0 \cdot 4 = +4$	$-1.0 \cdot 4 = -4$
$+1.5 \cdot 0.5 = +0.75$	$-1.5 \cdot 0.5 = -0.75$
$+1.5 \cdot 1 = +1.5$	$-1.5 \cdot 1 = -1.5$
$+1.5 \cdot 2 = +3$	$-1.5 \cdot 2 = -3$
$+1.5 \cdot 4 = +6$	$-1.5 \cdot 4 = -6$

Floating point number line!



b) The maximum quantization error occurs between 4 and 6 (or between -6 and -4). The error is (6-4)/2 = 1.
c) Any representation of the number 0 does not exist, you can choose to use the smallest positive and smallest negative numbers as +0 and -0. This is done in the IEEE standard.

2.7

$$[b_5b_4b_3b_2b_1b_0] = (-1^{b_5})\cdot(1.b_2b_1b_0)\cdot(2^{b_4b_3-1})$$

Significand $1.b_2b_1b_0$	**Exponent b_4b_3-1 (exess1)**	
$1.000_2 = 1.000_{10}$	00-1 = -1	$2^{00-1} = 0.5$
$1.001_2 = 1.125_{10}$	01-1 = 0	$2^{01-1} = 1$
$1.010_2 = 1.25_{10}$	10-1 = 1	$2^{10-1} = 2$
$1.011_2 = 1.375_{10}$	11-1 = 2	$2^{11-1} = 4$
$1.100_2 = 1.5_{10}$		
$1.101_2 = 1.625_{10}$		
$1.110_2 = 1.75_{10}$		
$1.111_2 = 1.875_{10}$		

- $0.25_{10} = 0.01_2 = 1.0_2 \cdot 2^{-2}$ exponent -2 *not* representable
- $0.8125_{10} = 0.1101_2 = 1.101_2 \cdot 2^{-1}$ representable
- $-1,375_{10} = -1.011_2 = -1.011_2 \cdot 2^0$ representable
- $4.25_{10} = 100.01_2 = 1.0001_2 \cdot 2^2$ significand 1.0001 *not* representable
- $7.5_{10} = 111.1_2 = 1.111_2 \cdot 2^2$ representable

Float addition

Algorithm:
1. Check for zeroes
2. Align significands
3. Add/Sub significands
4. Normalize result

a = 001111_2 $\rightarrow$ 0 01 111 $\rightarrow$ + **01** 1.111
b = 010010_2 $\rightarrow$ 0 10 010 $\rightarrow$ + **10** 1.010

Align significands. **10** > **01** shift smaller number *a* to right to get *same* exponent:
Significand 0.1111 exponent **10**

Add significands:

```
   0.1111
 + 1.010
  10.0011
```

Normalize result:

10.0011 exp **10** $\rightarrow$ 1.00011 exp **11**
Rounding: 1.000**11** ~ 1.001

Result: 0 11 001

a = 1.875 *b* = 2.5 *a*+*b* = 4.5 (4.375)

Foat multiplckation (this is simpler than addition!)

Simpler than addition!

Algorithm:
1. Check for zeroes
2. Add exponents and subtract *Bias*
3. Multiply significands
4. Normalize

Exponents: 01 10 Bias = 1
exp = 01+10 -1 = 10

Multiply significands

```
      1.010
    * 1.111
      1 010
     10 10
    101 0
 +  1 010
   10.010 110
```

Normalize result:

10.010110 exp **10** $\rightarrow$ 1.0010110 exp **11**

Rounding: 1.001**0110** ~ 1.001

Result: 0 11 001

a = 1.875 *b* = 2.5 *a***b* = 4.5 (4.6875)

Sets and Cubes

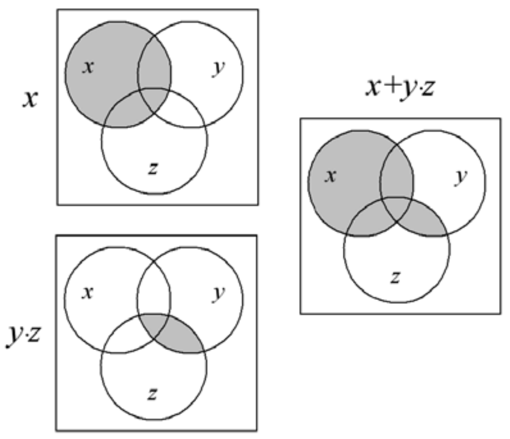
Venn-diagram representation

3.1

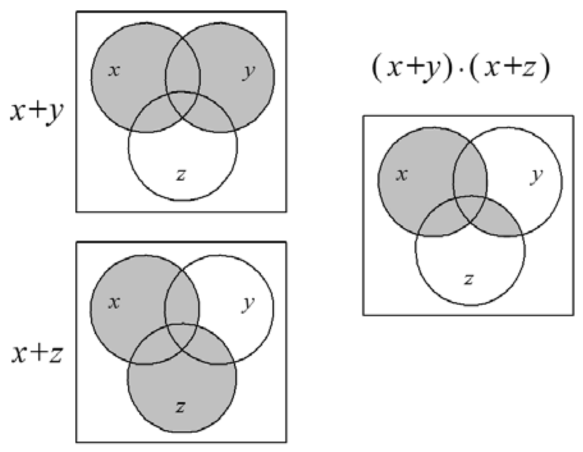
Proof of the distributive law with the help of Venn diagrams.

$x + y \cdot z = (x + y) \cdot (x + z)$

$x + y \cdot z$



$(x + y) \cdot (x + z)$

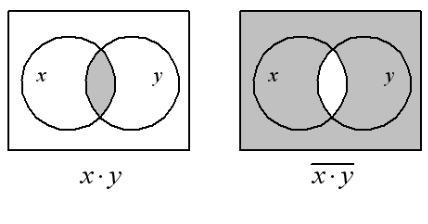


3.2

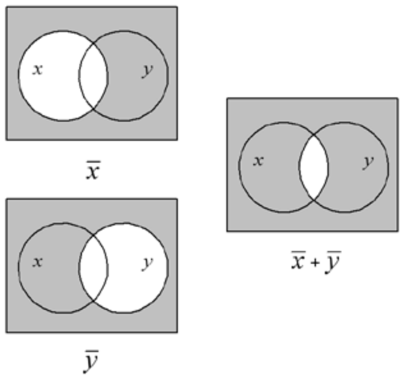
Proof of De Morgan's law using the Venn diagram.

$\overline{x \cdot y} = \overline{x} + \overline{y}$

$\overline{x \cdot y}$



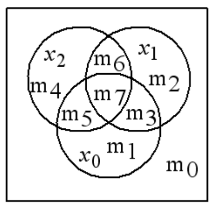
$\overline{x} + \overline{y}$



3.3

a) The minterms placement in a three variables Venn diagram.

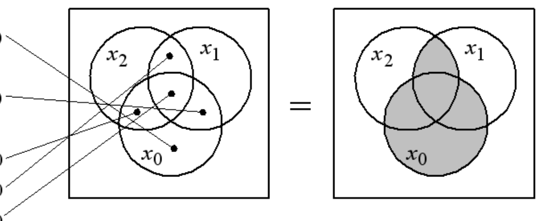
$x_2 x_1 x_0$	
0 0 0	m_0
0 0 1	m_1
0 1 0	m_2
0 1 1	m_3
1 0 0	m_4
1 0 1	m_5
1 1 0	m_6
1 1 1	m_7



b)

$$f = \bar{x}_2\bar{x}_1x_0 + \bar{x}_2x_1x_0 + x_2\bar{x}_1x_0 + x_2x_1\bar{x}_0 + x_2x_1x_0$$

$x_2x_1x_0$		
0 0 0	0	
0 0 1	1	$\bar{x}_2\bar{x}_1x_0$
0 1 0	0	
0 1 1	1	$\bar{x}_2x_1x_0$
1 0 0	0	
1 0 1	1	$x_2\bar{x}_1x_0$
1 1 0	1	$x_2x_1\bar{x}_0$
1 1 1	1	$x_2x_1x_0$



$f = x_2x_1 + x_0$

Venn diagram method clearly shows the boolean relationships, but are difficult to use for more than three variables. It is impractical to convert to a computer algorithm.

3.4

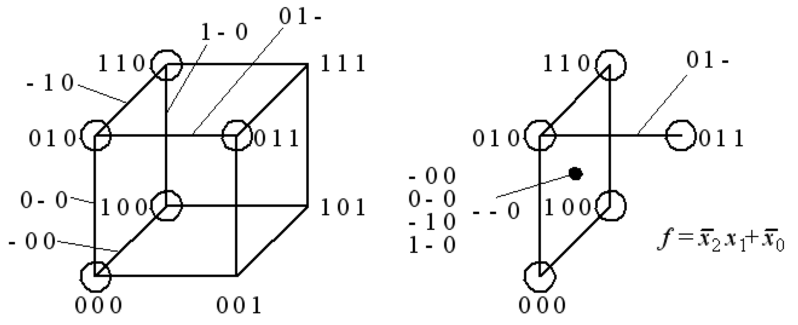
a) Represent the following function of three variables, as a 3-dimensional cube with Gray-coded corner.

$$f(x_2, x_1, x_0) = \sum m(0,2,3,4,6) = \bar{x}_2\,\bar{x}_1\,\bar{x}_0 + \bar{x}_2 x_1 \bar{x}_0 + \bar{x}_2\, x_1\, x_0 + x_2\,\bar{x}_1\,\bar{x}_0 + x_2 x_1 \bar{x}_0$$

b) Use the cube to to simplify the function..

$$f = \bar{x}_2\bar{x}_1\bar{x}_0 + \bar{x}_2x_1\bar{x}_0 + \bar{x}_2x_1x_0 + x_2\bar{x}_1\bar{x}_0 + x_2x_1\bar{x}_0$$

$x_2x_1x_0$		
0 0 0	1	$\bar{x}_2\bar{x}_1\bar{x}_0$
0 0 1	0	
0 1 0	1	$\bar{x}_2x_1\bar{x}_0$
0 1 1	1	$\bar{x}_2x_1x_0$
1 0 0	1	$x_2\bar{x}_1\bar{x}_0$
1 0 1	0	
1 1 0	1	$x_2x_1\bar{x}_0$
1 1 1	0	



The cubic representation is hard to visualize for more than three dimensions, but the minimization method can be easily defined for any number of variables and dimensions, and then form the basis for computer algorithms.

Boolean algebra and gates

4.1

a) $f = a\cdot\overline{c}\cdot d + a\cdot d \quad = a\cdot d\cdot(\overline{c}+1) = a\cdot d$

b) $f = a\cdot(\overline{b}+\overline{a}\cdot c + a\cdot b) \quad = a\cdot\overline{b} + a\cdot\overline{a}\cdot c + a\cdot a\cdot b = a\cdot\overline{b} + 0 + a\cdot b = a\cdot(\overline{b}+b) = a$

c) $f = a+\overline{b}+\overline{a}\cdot b+\overline{c} \quad = (1\cdot\overline{b}+\overline{a}\cdot b) \quad + a+\overline{c} = \{consensus\} =$

$= (1\cdot\overline{b}+\overline{a}\cdot b+1\cdot\overline{a}) \quad + a+\overline{c} = \ldots\,\overline{a}+a\,\ldots = 1$

d) $f = (a+b\cdot\overline{c})\cdot(\overline{a}\cdot\overline{b}+c) \quad = a\cdot\overline{a}\cdot\overline{b} + a\cdot c + \overline{a}\cdot b\cdot\overline{b}\cdot\overline{c} + b\cdot c\cdot\overline{c} = 0 + a\cdot c + 0 + 0 = a\cdot c$

e) $f = (a+\overline{b})\cdot(\overline{a}+b)\cdot(a+b) \quad = (a\cdot\overline{a}+a\cdot b+\overline{a}\cdot\overline{b}+b\cdot\overline{b})\cdot(a+b) =$

$(0+a\cdot b+\overline{a}\cdot\overline{b}+0)\cdot(a+b) = a\cdot a\cdot b + a\cdot\overline{a}\cdot\overline{b} + a\cdot b\cdot b + \overline{a}\cdot\overline{b}\cdot b = a\cdot b + a\cdot b = a\cdot b$

f) $f = \overline{a}\cdot\overline{b}\cdot c + a\cdot b\cdot c + \overline{a}\cdot b\cdot c \quad = c\cdot(\overline{a}\cdot\overline{b}+a\cdot b+\overline{a}\cdot b) = c\cdot(\overline{a}\cdot\overline{b}+b\cdot(a+\overline{a})) =$

$= c\cdot(\overline{a}\cdot\overline{b}+1\cdot b) = \{consensus\} = c\cdot(\overline{a}\cdot\overline{b}+1\cdot b+\overline{a}\cdot 1) = c\cdot(\overline{a}\cdot\overline{b}+b+\overline{a}) = c\cdot(\overline{a}\cdot(\overline{b}+1)+b) =$

$= c\cdot(\overline{a}+b) = \overline{a}\cdot c + b\cdot c$

g) $f = \overline{a}\cdot b\cdot\overline{c}+\overline{a}\cdot b\cdot\overline{d}+c\cdot d \quad = \overline{a}\cdot b\cdot(\overline{c}+\overline{d})+c\cdot d = \overline{a}\cdot b\cdot\overline{\overline{(\overline{c}+\overline{d})}}+c\cdot d =$

$\overline{a}\cdot b\cdot\overline{c\cdot d} + c\cdot d = \{x+\overline{x}y = x+y\} = \overline{a}\cdot b + c\cdot d$

h) $f = a + (\overline{\overline{a}\cdot\overline{b}}) \quad = \{deMorgan\} = a + \overline{\overline{a}} + \overline{\overline{b}} = a+b$

i) $f = \overline{\overline{a}+\overline{\overline{a}\cdot b+c}} \quad = \{deMorgan\} = a\cdot(\overline{a}\cdot b+c) = a\cdot\overline{a}\cdot b + a\cdot c = ac$

4.2

Prove algebraically that the following relationships are valid.

a) $(\overline{x_3}x_2 + 1 + x_3\overline{x_2})x_1 + x_2x_1 + x_2 + \overline{x_1} = 1$ **LHS:** $(\ldots+1)x_1 = x_1 \quad x_1 + \overline{x_1} + \ldots = 1$

b) $\overline{x_3}x_2x_1 + \overline{(x_3+\overline{x_1})} = \overline{x_3}x_1$ **LHS:** $\overline{x_3}x_2x_1 + \overline{(x_3+\overline{x_1})} = \overline{x_3}x_2x_1 + \overline{x_3}\,x_1 = \overline{x_3}\,x_1(x_2+1) = \overline{x_3}\,x_1$

c) $\overline{(x_2+x_1)}\,\overline{x_2}\,\overline{x_1} + x_3 = \overline{x_2}\,\overline{x_1} + x_3$ **LHS:** $\overline{(x_2+x_1)}\,\overline{x_2}\,\overline{x_1} + x_3 = \overline{x_2}\,\overline{x_1}\,\overline{x_2}\,\overline{x_1} + x_3 = \overline{x_2}\,\overline{x_1} + x_3$

d) $\overline{x_2+x_1} + \overline{x_2}\,\overline{x_1} + x_3 = \overline{x_2}\,\overline{x_1} + x_3$ **LHS:** $\overline{x_2+x_1} + \overline{x_2}\,\overline{x_1} + x_3 = \overline{x_2}\,\overline{x_1} + \overline{x_2}\,\overline{x_1} + x_3 = \overline{x_2}\,\overline{x_1} + x_3$

4.3

Simplify the following three expressions as much as possible.

a) $(x+y)(\overline{x}+z) = x\overline{x} + y\overline{x} + xz + yz = \overline{x}y + xz + yz = \{consensus\ remove\ yz\} = \overline{x}y + xz$

b) $(x+\overline{y}+xy)(x+\overline{y})\overline{x}y = (x+xy+x\overline{y}+\overline{y})\overline{x}y = 0+0+0+0$

c) $x(1+\overline{x}y)+\overline{x} = x(1)+\overline{x} = 1$

4.4

Simplify the following expressions as much as possible.

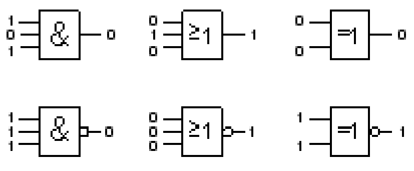
$\overline{(a+b+\overline{c})(a+\overline{b}+\overline{c})(\overline{a}+\overline{\overline{b}c}+b\overline{c})} = \overline{(a+b+\overline{c})(a+\overline{b}+\overline{c})(\overline{a}+b+\overline{c}+b\overline{c})} =$

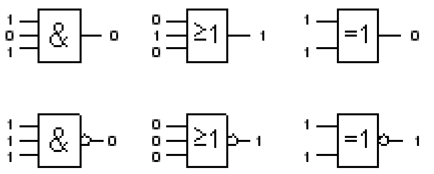
$= \overline{(a+b+\overline{c})(a+\overline{b}+\overline{c})(\overline{a}+b+\overline{c})} =$

$\overline{(a+b+\overline{c})} + \overline{(a+\overline{b}+\overline{c})} + \overline{(\overline{a}+b+\overline{c})} = \overline{a}\overline{b}c + \overline{a}bc + a\overline{b}c = \overline{a}\overline{b}c + \overline{a}\overline{b}c + \overline{a}bc + a\overline{b}c =$

$= \overline{a}c(\overline{b}+b) + \overline{b}c(a+\overline{a}) = \overline{a}c + \overline{b}c$

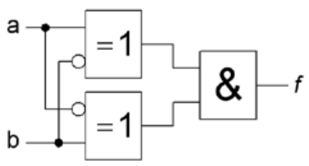
4.5


a)


b)

4.6

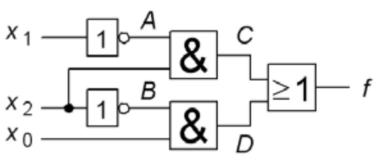
simplify *f*(*a*,*b*) realized by the figure gates, as far as possible, and give the name of the function.

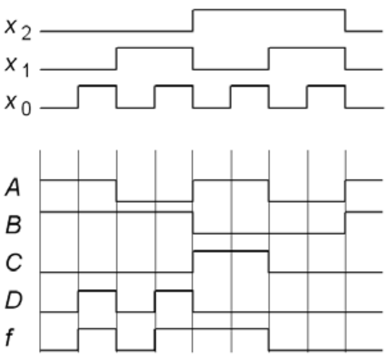


$$(a\oplus\overline{b})(b\oplus\overline{a}) = (a\overline{\overline{b}}+\overline{a}\,\overline{b})(b\overline{\overline{a}}+\overline{b}\,\overline{a}) = (ab+\overline{a}\,\overline{b})(ab+\overline{a}\,\overline{b}) = ab+\overline{a}\,\overline{b}$$

It will be an XNOR function.

4.7

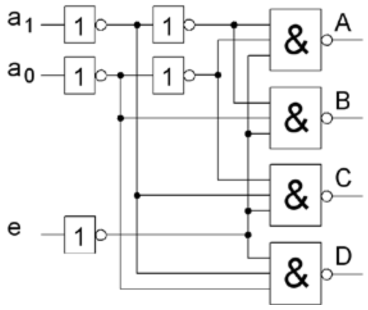




x_2	x_1	x_0	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

4.8

Indicate the logical expressions for *A*, *B*, *C* and *D*.



$$A = \overline{\overline{\overline{a_1}}\;\overline{\overline{a_0}}\;\overline{e}} = \overline{a_1}+\overline{a_0}+e$$

$$B = \overline{\overline{\overline{a_1}}\;\overline{a_0}\;\overline{e}} = \overline{a_1}+a_0+e$$

$$C = \overline{\overline{a_1}\;\overline{\overline{a_0}}\;\overline{e}} = a_1+\overline{a_0}+e$$

$$D = \overline{\overline{a_1}\;\overline{a_0}\;\overline{e}} = a_1+a_0+e$$

4.9

Simplify the complex expressions below as much as possible.

a)

$$XOR-\ function \quad a\oplus b = a\overline{b}+\overline{a}b$$

$$x_2\oplus x_1\oplus x_1x_2 \quad = (x_2\overline{x_1}+\overline{x_2}x_1)\overline{x_1x_2}+\overline{(x_2\overline{x_1}+\overline{x_2}x_1)}\,x_1x_2 =$$

$$= (x_2\overline{x_1}+\overline{x_2}x_1)(\overline{x_1}+\overline{x_2})+\overline{x_2\overline{x_1}}\;\overline{\overline{x_2}x_1}\;x_1x_2 = (x_2\overline{x_1}+0+0+\overline{x_2}x_1)+(\overline{x_2}+x_1)(x_2+\overline{x_1})x_1x_2 =$$

$$= +(0+x_1x_2+\overline{x_2}\;\overline{x_2}+0)x_1x_2 = \overline{x_1}+\overline{x_2}x_1+x_1x_2 = x_2\overline{x_1}+\overline{x_2}x_1+x_1x_2+x_1x_2 =$$

$$= x_2(\overline{x_1}+x_1)+x_1(\overline{x_2}+x_2) = x_2+x_1$$

b)

$$x_2x_1\oplus\overline{(x_2+x_1)} \quad = x_2x_1(x_2+x_1)+\overline{x_2x_1}\;\overline{(x_2+x_1)} = x_2x_1+(\overline{x_2}+\overline{x_1})\overline{x_2}\;\overline{x_1} =$$

$$= x_2x_1+\overline{x_2}\;\overline{x_1}$$

4.10

Show that

a) $\overline{x_2 \oplus x_1} = \overline{x_2} \oplus x_1 = x_2 \oplus \overline{x_1}$

This time we are proving the relationship with so-called "perfect induction." It involves directly inserting all four combinations of the two variables in the various expressions. If the expressions has the same truth table so they're equivalent. When the variables are few, this non algebraic method cold be used.

x_2	x_1	$\overline{x_2}$	$\overline{x_1}$	$\overline{x_2 \oplus x_1}$	$\overline{x_2} \oplus x_1$	$x_2 \oplus \overline{x_1}$
0	0	1	1	1	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	1	1

b)

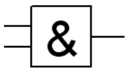
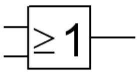
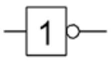
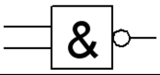
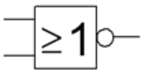
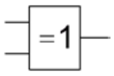
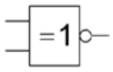
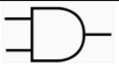
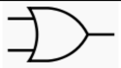
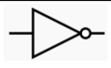
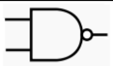
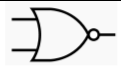
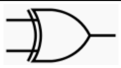
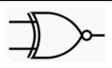
$$x_2 \oplus x_1 = \overline{x_2} \oplus \overline{x_1}$$

$$LHS: \quad x_2 \oplus x_1 = x_2\overline{x_1} + \overline{x_2}x_1$$

$$RHS: \quad \overline{x_2} \oplus \overline{x_1} = \overline{x_2}\,\overline{\overline{x_1}} + \overline{\overline{x_2}}\,\overline{x_1} = x_2\overline{x_1} + \overline{x_2}x_1 \quad LHS = RHS$$

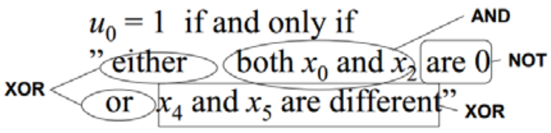
4.11

The figure shows the international standard gate symbols. Name the gates and draw the corresponding American gate symbols.

AND	OR	NOT	NAND	NOR	XOR	XNOR
						
						

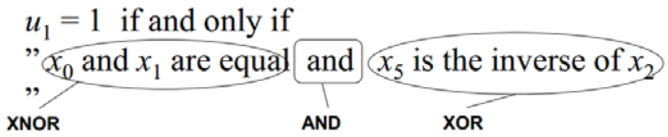
4.12

From text to Boolean expression.



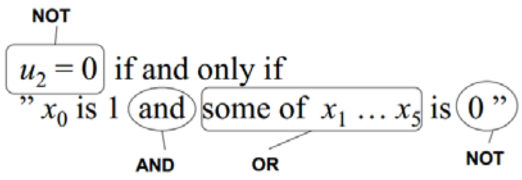
$u_0 = 1$ if and only if ”either both x_0 and x_2 are 0 or x_4 and x_5 are different”

$$u_0 = \overline{x_0} \cdot \overline{x_2} \oplus (x_4 \oplus x_5)$$



$u_1 = 1$ if and only if ”x_0 and x_1 are equal and x_5 is the inverse of x_2 ”

$$u_1 = \overline{x_0 \oplus x_1} \cdot (x_5 \oplus x_2)$$



$u_2 = 0$ if and only if ” x_0 is 1 and some of $x_1 \dots x_5$ is 0 ”

$$\overline{u_2} = x_0 \cdot (\overline{x_1} + \overline{x_2} + \overline{x_3} + \overline{x_4} + \overline{x_5}) \quad \Rightarrow \quad u_2 = \overline{x_0 \cdot (\overline{x_1} + \overline{x_2} + \overline{x_3} + \overline{x_4} + \overline{x_5})} = \overline{x_0} + x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5$$

The truth table, SoP and PoS form, complete logic

5.1

Contacts always depicted in the unaffected position. To get the light to shine you should simultaneously press the numbers "4" and "8" ie contact *d* and *h*. Please note that you must not press down any other contacts! The logical function (the light) becomes:

$$f = \overline{a}\cdot\overline{b}\cdot\overline{c}\cdot d\cdot\overline{e}\cdot\overline{f}\cdot\overline{g}\cdot h\cdot\overline{i}\cdot\overline{k}$$

Code lock is a decoder, that decodes a single minterm in the truth table.

5.2

a b c	*f*	*a b c*	*f*
0 0 0	1	100	1
0 0 1	0	1 0 1	1
0 1 0	0	1 1 0	0
0 1 1	0	1 1 1	1

Function on SoP-normal form:

$$f = \overline{a}\,\overline{b}\,\overline{c} + a\,\overline{b}\,\overline{c} + a\,\overline{b}\,c + a\,b\,c$$

Function on PoS-normal form:

$$f = (a+b+\overline{c})\cdot(a+\overline{b}+c)\cdot(a+\overline{b}+\overline{c})\cdot(\overline{a}+\overline{b}+c)$$

5.3

$$f(x,y,z) = x\overline{y} + y\overline{z} + \overline{x}z \quad = x\overline{y}(z+\overline{z}) + (x+\overline{x})y\overline{z} + \overline{x}(y+\overline{y})z =$$

$$= x\overline{y}z + x\overline{y}\,\overline{z} + xy\overline{z} + \overline{x}y\overline{z} + \overline{x}yz + \overline{x}\,\overline{y}z$$

$$\Rightarrow \quad f(x,y,z) = \sum m(001, 010, 011, 100\ , 101, 110) = \sum m(1, 2, 3, 4, 5, 6)$$

$$\Rightarrow \quad f(x,y,z) = \prod M(0,7) = (\overline{x}+\overline{y}+\overline{z})(x+y+z)$$

5.4

$$f(x,y,z) = (x+\overline{y})\Big(xyz + \overline{y}(x+z)\Big) + xy\overline{z}(x+\overline{x}y) \quad = (x+\overline{y})\Big(xyz + \overline{y}(x+z)\Big) + xy\overline{z}(x+y) =$$

$$= (x+\overline{y})\Big(xyz + \overline{y}(x+z)\Big) + xy\overline{z} =$$

$$= xyz + x\overline{y} + x\overline{y}z + xy\overline{y}z + x\overline{y} + \overline{y}z + xy\overline{z} =$$

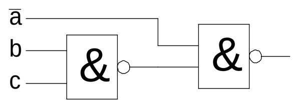
$$= xyz + x\overline{y}(z+\overline{z}) + x\overline{y}z + x\overline{y}(z+\overline{z}) + (x+\overline{x})\overline{y}z + xy\overline{z} =$$

$$= xyz + x\overline{y}z + x\overline{y}\,\overline{z} + x\overline{y}z + x\overline{y}z + x\overline{y}\,\overline{z} + x\overline{y}z + \overline{x}\,\overline{y}z + xy\overline{z} \quad = \overline{x}\,\overline{y}z + x\overline{y}\,\overline{z} + x\overline{y}z + xy\overline{z} + xyz$$

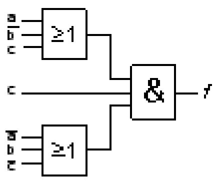
$$\Rightarrow \quad f(x,y,z) = \sum m(001, 100, 101, 110, 111) = \sum m(1, 4, 5, 6, 7)$$

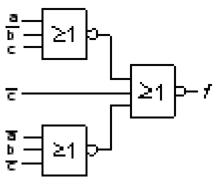
$$\Rightarrow \quad f(x,y,z) = \prod M(0,\ 2\ ,3) \quad = (x+y+z)(x+\overline{y}+z)(x+\overline{y}+\overline{z})$$

5.5



5.6





5.7

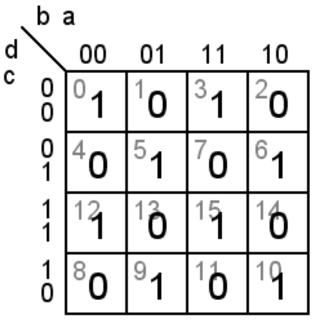
a) Parity circuit for even parity, the number of ones must be even (0, 2, or 4) for ”1” at the output.

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

Half of the rows in the truth table are “1”. This function is not possible to minimize, but all 8 minterms need to be included in the SP form!

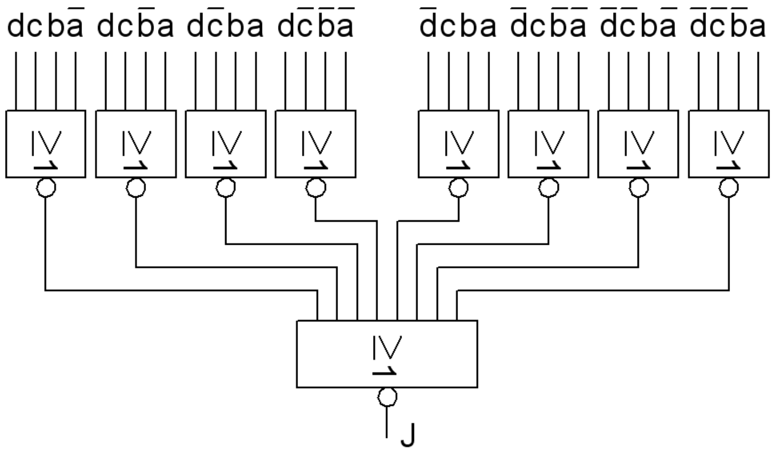
$$J = \overline{d}\,\overline{c}\,\overline{b}\,\overline{a} + \overline{d}\,\overline{c}ba + \overline{d}c\overline{b}a + \overline{d}cb\overline{a} + d\overline{c}\,\overline{b}a + d\overline{c}b\overline{a} + dc\overline{b}\,\overline{a} + dcba$$

(Anyone who already knows the Karnaugh map can directly see that no "groupings" are possible.)



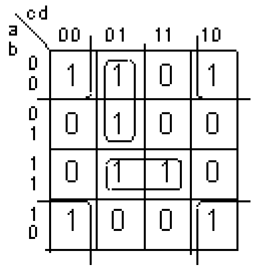
b) With NOR gates the PoS form is better suitable.

$$J = (d + c + b + \overline{a})(d + c + \overline{b} + a)(d + \overline{c} + b + a)(d + \overline{c} + \overline{b} + \overline{a})\cdot$$
$$\cdot(\overline{d} + c + b + a)(\overline{d} + c + \overline{b} + \overline{a})(\overline{d} + \overline{c} + b + \overline{a})(\overline{d} + \overline{c} + \overline{b} + a)$$



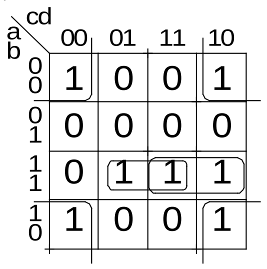
Karnaugh map

6.1



$$f = \overline{a}\,\overline{c}\,d + abd + \overline{b}\,\overline{d}$$

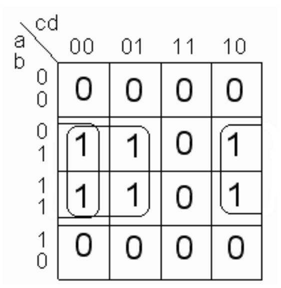
6.2



$$f = abd + abc + \overline{bd}$$

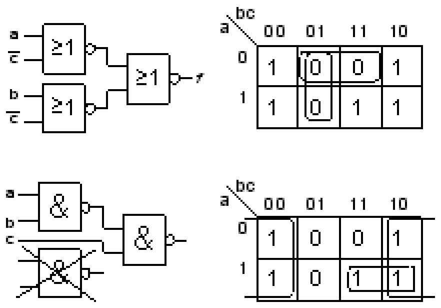
6.3

$$f = \overline{a}b\overline{c} + ab\overline{c} + bc\overline{d}$$



$$f = b\overline{c} + b\overline{d}$$

6.4

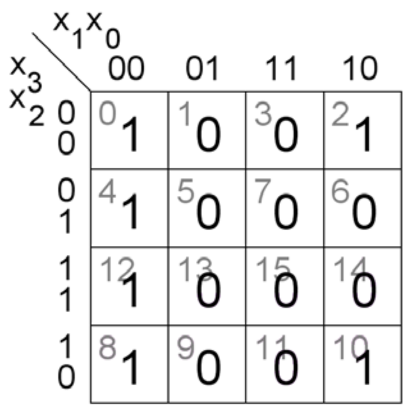


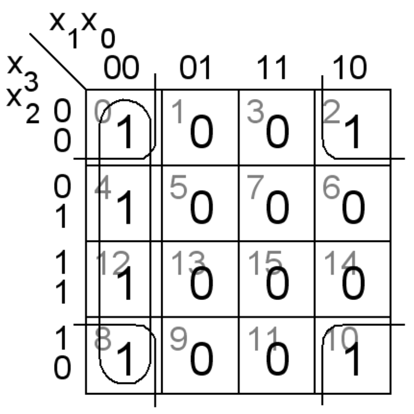
6.5

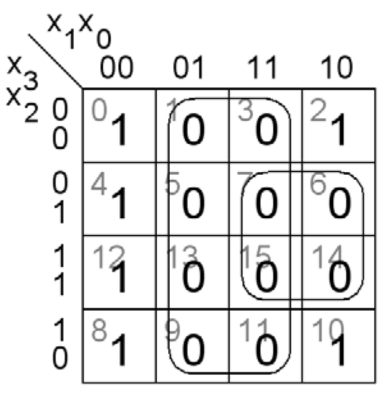
Truth table and Karnaugh map. The minimized function is obtained by grouping of the 1's in the Karnaugh map. The function inverse is obtained if 0:s are grouped together "wrongly" as if they were 1's.

$$f(x_3, x_2, x_1, x_0) = \sum m(0, 2, 4, 8, 10, 12) \qquad f = ? \qquad \overline{f} = ?$$

	x_3	x_2	x_1	x_0	f
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0







$f = \bar{x}_1 \bar{x}_0 + \bar{x}_2 \bar{x}_0$

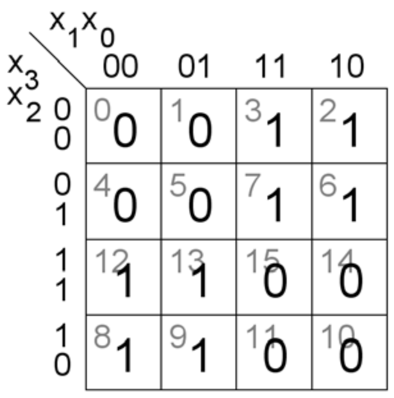
$\bar{f} = \{\, 0:\text{s} \;\; \text{as} \, 1:\text{s} \,\} = x_0 + x_2 x_1$

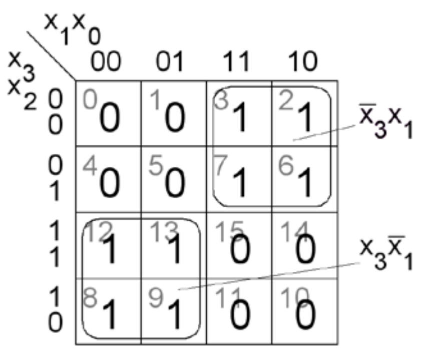
6.6

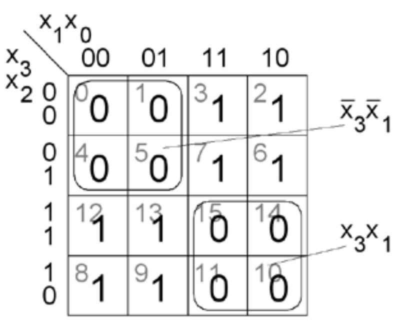
Truth table and Karnaugh map. The minimized function is obtained by grouping of the 1's in the Karnaugh map. The function inverse is obtained if 0:s are grouped together "wrongly" as if they were 1's.

$f(x_3, x_2, x_1, x_0) = \prod M(0, 1, 4, 5, 10, 11, 14, 15) \quad f = ? \quad \bar{f} = ?$

	x_3	x_2	x_1	x_0	f
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
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	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0







$f = x_3 \bar{x}_1 + \bar{x}_3 x_1 = x_3 \oplus x_1$

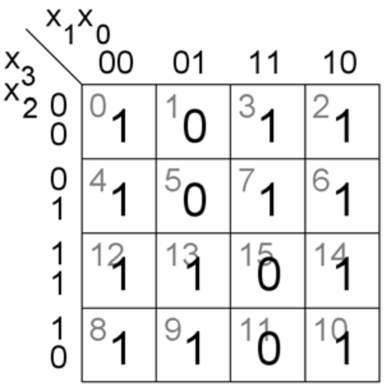
$\bar{f} = \{\, 0:\text{s as } 1:\text{s} \,\} = \bar{x}_3 \bar{x}_1 + x_3 x_1 = \overline{x_3 \oplus x_1}$

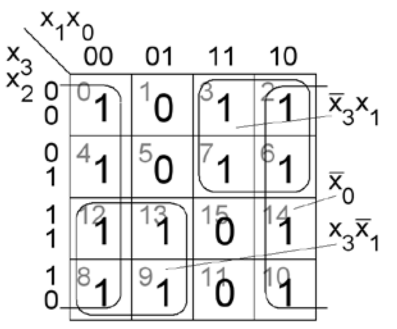
6.7

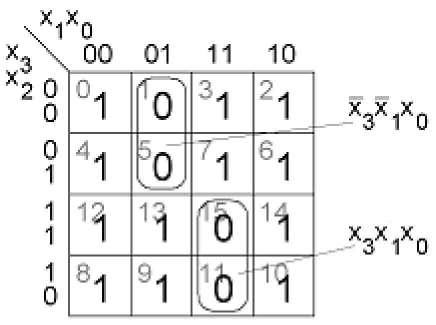
Truth table and Karnaugh map. The minimized function is obtained by grouping of the 1's in the Karnaugh map. The function inverse is obtained if 0:s are grouped together "wrongly" as if they were 1's.

$f(x_3, x_2, x_1, x_0) = \sum m(0, 2, 3, 4, 6, 7, 8, 9, 10, 12, 13, 14) \quad f = ? \quad \overline{f} = ?$

	x_3	x_2	x_1	x_0	f
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0





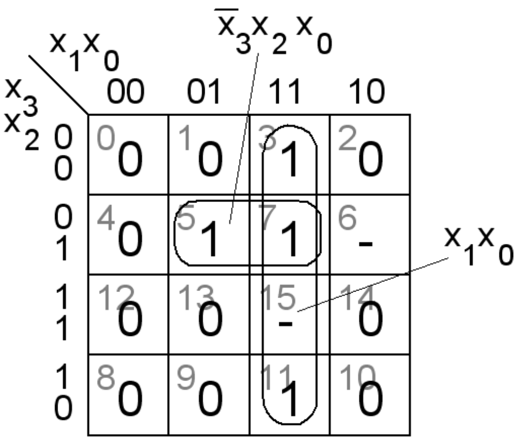


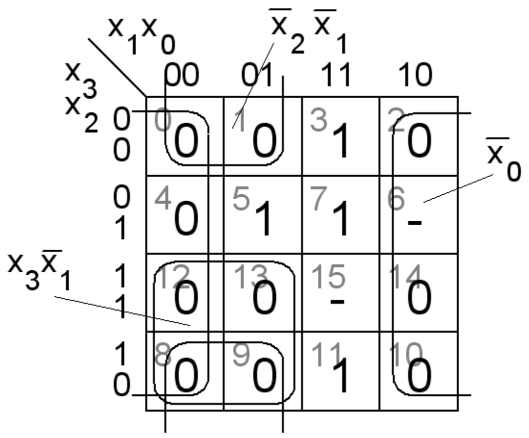
$f = \overline{x_0} + x_3\overline{x_1} + \overline{x_3}\,x_1$

$\overline{f} = \{\,0:\text{s as }1:\text{s}\,\} = \overline{x_3}\,\overline{x_1}\,x_0 + x_3x_1x_0$

6.8

$f(x_3,x_2,x_1,x_0)=\sum m(3,5,7,11)+d(6,15) \quad f=? \quad \overline{f}=?$

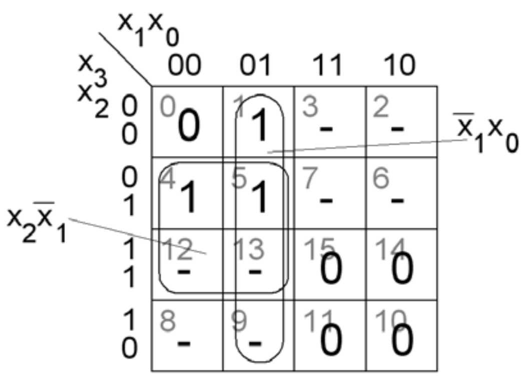


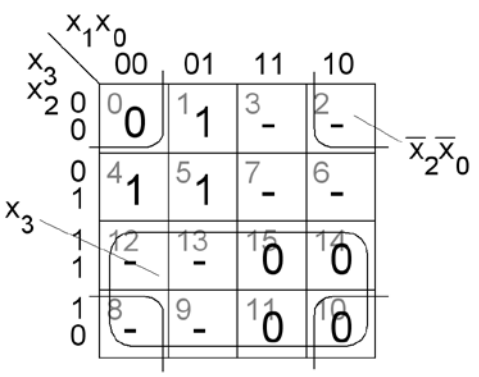


$f=x_1x_0+\overline{x_3}\,x_2\,x_0$ $\qquad$ $\overline{f}=\overline{x_0}+\overline{x_2}\,\overline{x_1}+x_3\,\overline{x_1}$

6.9

$f(x_3,x_2,x_1,x_0)=\sum m(1,4,5)+d(2,3,6,7,8,9,12,13) \quad f=? \quad \overline{f}=?$



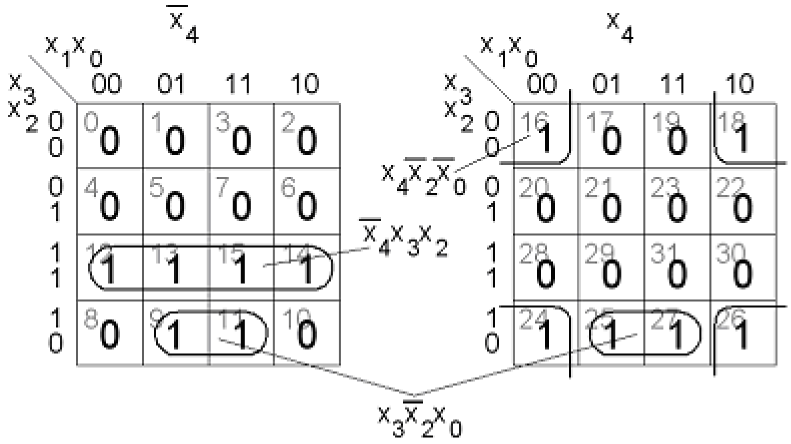


$f=x_2\,\overline{x_1}+\overline{x_1}\,x_0$ $\qquad$ $\overline{f}=x_3+\overline{x_2}\,\overline{x_0}$ or $\overline{f}=x_1+\overline{x_2}\,\overline{x_0}$

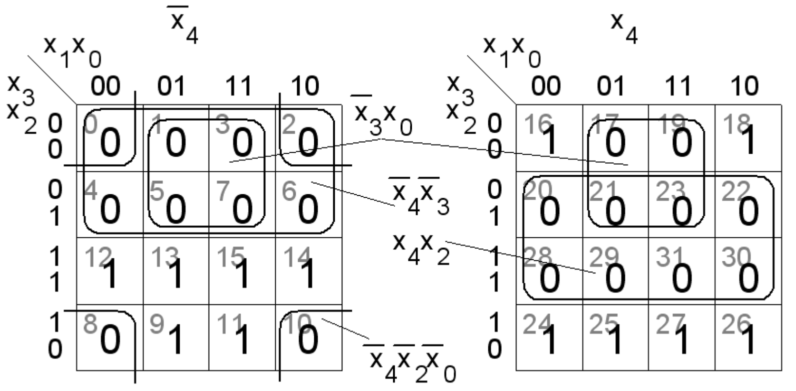
6.10

Karnaugh map for **five** variables. The left diagram is for $\overline{x}_4$ and the right for x_4. If the same grouping can be made in *both* diagrams then the variable $\overline{x}_4$ or x_4 is omitted, otherwise it has to be included.

$f(x_4, x_3, x_2, x_1, x_0) \quad f = ? \quad \overline{f} = ?$



$f = \overline{x_4}\, x_3\, x_2 + x_3\, \overline{x_2}\, x_0 + x_4\, \overline{x_2}\, \overline{x_0}$

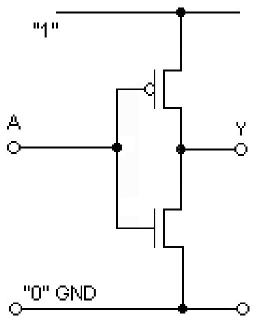


$\overline{x}_4$
x_4
x_1x_0
x_3x_2
$\overline{x}_3x_0$
$\overline{x}_4\overline{x}_3$
x_4x_2
$\overline{x}_4x\overline{x}_2\overline{x}_0$

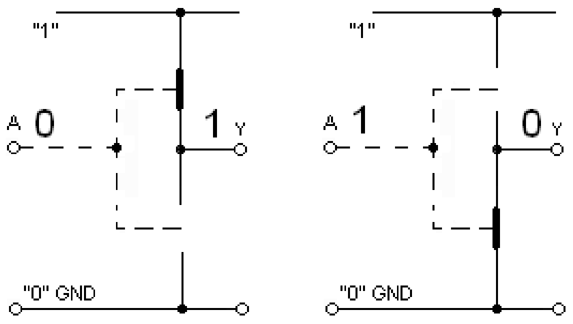
$\overline{f} = \overline{x_4}\, \overline{x_3} + \overline{x_3}\, x_0 + x_4\, x_2 + \overline{x_4}\, \overline{x_2}\, \overline{x_0}$

MOS-transistors and digital circuits

7.1

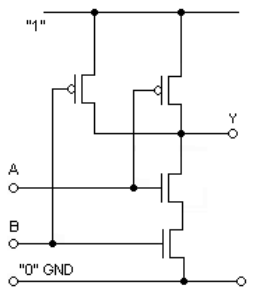


A	Y
0	1
1	0

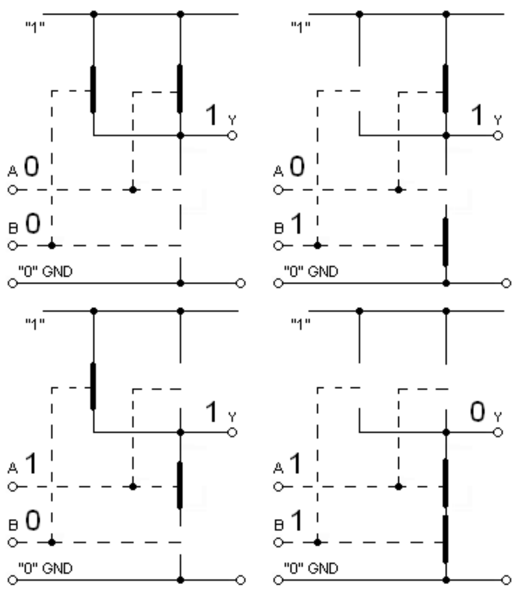


This is a CMOS-inverter. $Y = \overline{A}$.

7.2

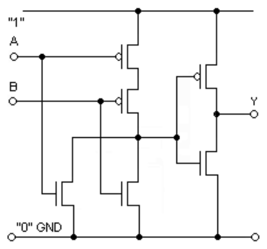


A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

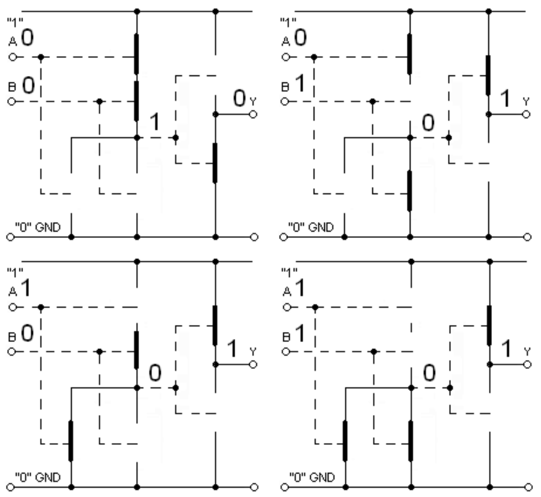


This is a CMOS-NAND-gate. $Y = \overline{A \cdot B}$.

7.3



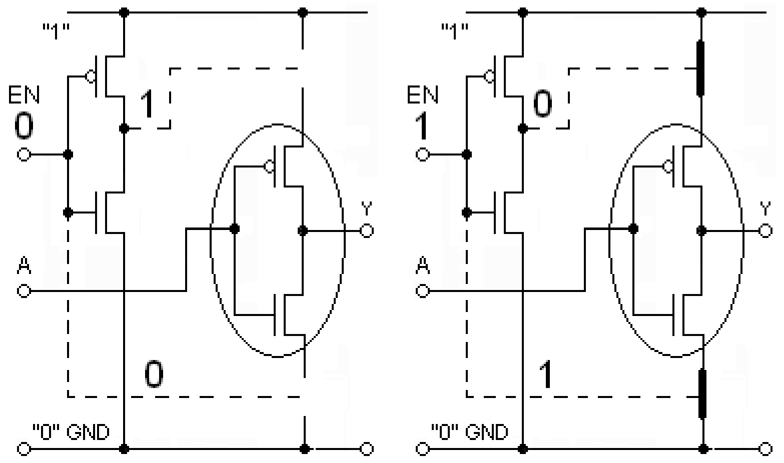
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



This is a CMOS-OR-gate. $Y = A + B$.

7.4

The circuit is an inverter with THREE-state output. When *EN* = 1 becomes the circled portion "plugged in" and the relationship between *Y* and *A* is then inverted, $Y = \overline{A}$. When EN = 0 becomes the circled portion "unplugged". Output becomes disconnected and in a third state, exept ”1” and ”0” there is a state "disconnected". Since the output *Y* now is disconnected it is no longer affected by the input *A*.

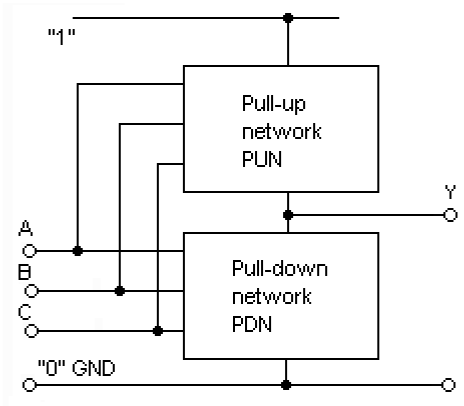


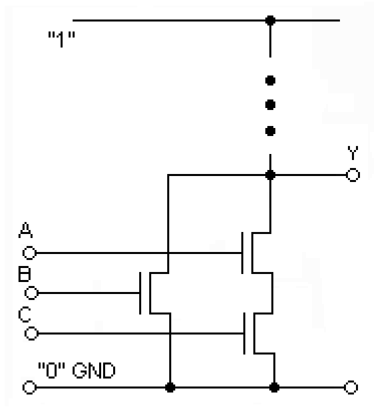
THREE-state outputs are used to make it possible to connect many outputs to a single line (**bus**). Several circuits outputs can utilize a common input line, provided that only one of the circuits are active (*EN* = 1) at a time – the others has *EN* = 0 and are ”disconnected”.

7.5

CMOS-circuits cosists of two subcircuits that each other's inverses. The Pull-up-net, PUN, transfers ”1” to the output while the pull-down network transfers ”0”. If one analyzes the Pull-down network, one therefore get the function *Y* inverted.

$$\overline{Y} = A \cdot C + B \quad \Rightarrow \quad Y = \overline{A \cdot C + B} = \overline{A \cdot C} \cdot \overline{B} = (\overline{A} + \overline{C})\overline{B}$$

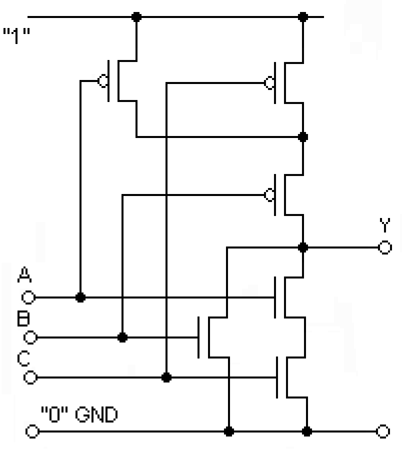




$$(\overline{A} + \overline{C})\overline{B}$$

Pull-up-net shall have *A* and *C* in parallell (+) and then in series (·) with B. The use of PMOS-transistors inverts the variables A, B and C.

$$(\overline{A} + \overline{C})\overline{B}$$

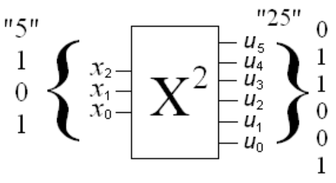


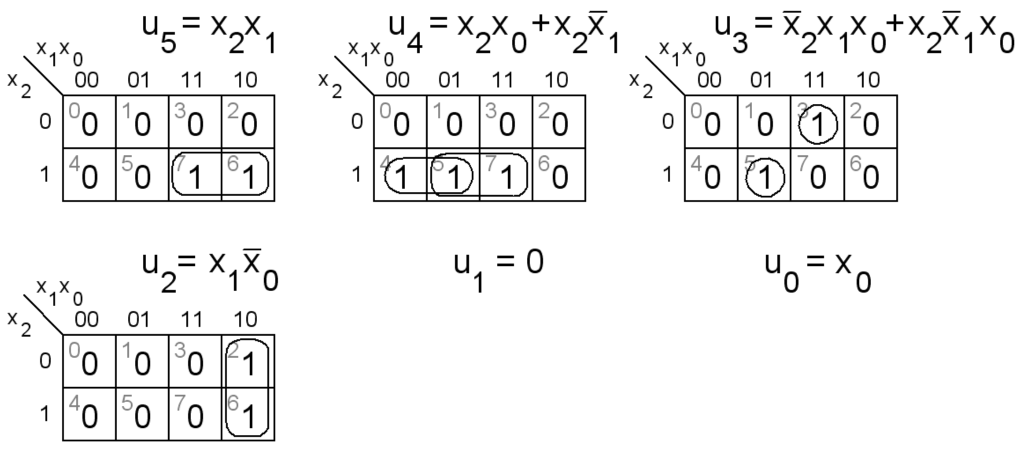
Combinatorial circuits

8.1

X	x_2	x_1	x_0	$U = X^2$	u_5	u_4	u_3	u_2	u_1	u_0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	1
2	0	1	0	4	0	0	0	1	0	0
3	0	1	1	9	0	0	1	0	0	1
4	1	0	0	16	0	1	0	0	0	0
5	1	0	1	25	0	1	1	0	0	1
6	1	1	0	36	1	0	0	1	0	0
7	1	1	1	49	1	1	0	0	0	1

In the truth table we can see that u_1 allways is equal to 0. u_1 output can therefore be directly connected 0V (ground) so it will have the output constant 0. We can also see that u_0 allways is equal to x_0. u_0 output can therefore be directly connected to the x_0 input. The other expressions are obtained by using their Karnaugh maps.

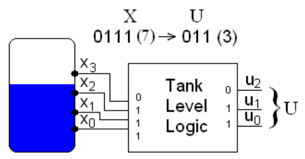


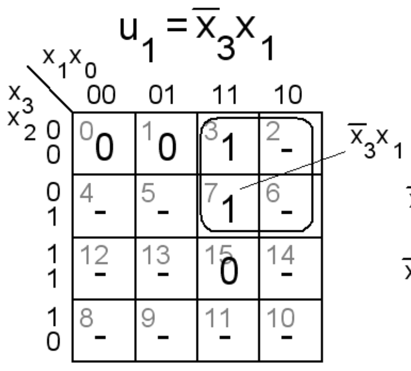


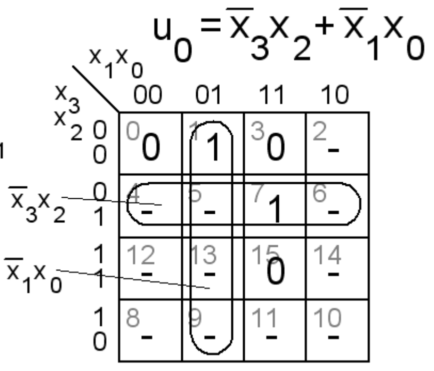
8.2

X	x_3	x_2	x_1	x_0	U	u_2	u_1	u_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	0	0	1
3	0	0	1	1	2	0	1	0
7	0	1	1	1	3	0	1	1
15	1	1	1	1	4	1	0	0

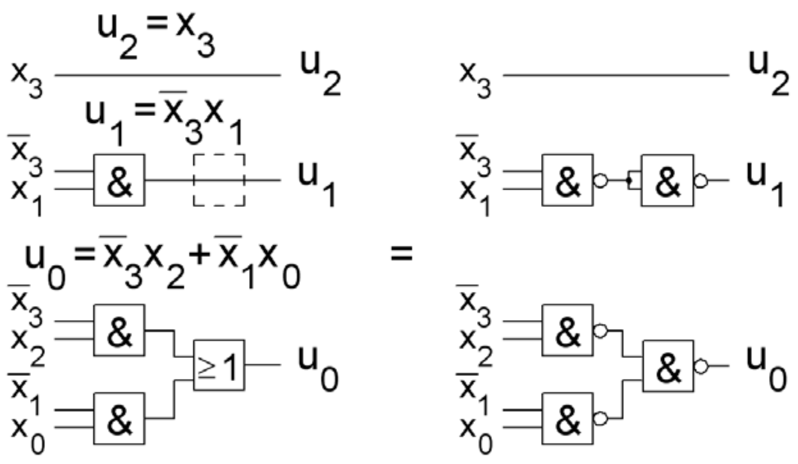
In the truth table we can see that u_2 and x_3 are equal why u_2 directly can be connected to x_3. $u_2 = x_3$.
The other expressions are obtained by using their Karnaugh maps.







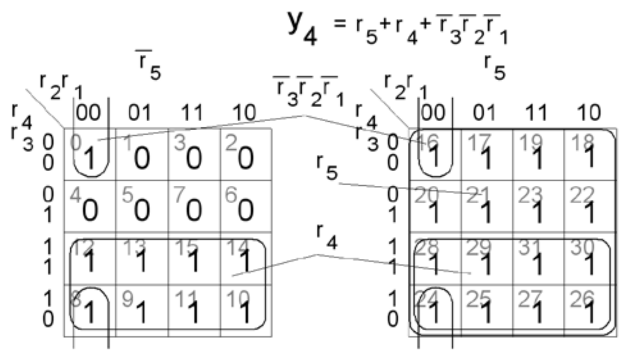
Gate circuits:

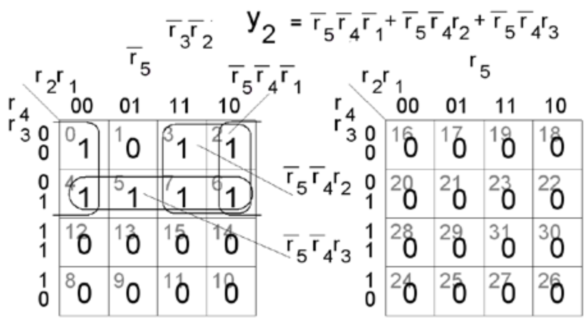


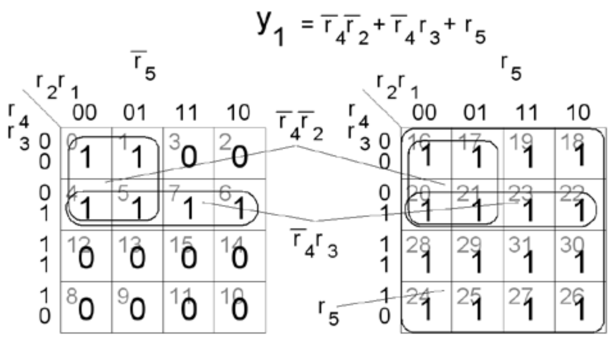
8.3

Truth table

r_5	r_4	r_3	r_2	r_1	y_4	y_2	y_1
1	-	-	-	-	1	0	1
0	1	-	-	-	1	0	0
0	0	1	-	-	0	1	1
0	0	0	1	-	0	1	0
0	0	0	0	1	0	0	1
0	0	0	0	0	1	1	1







8.4

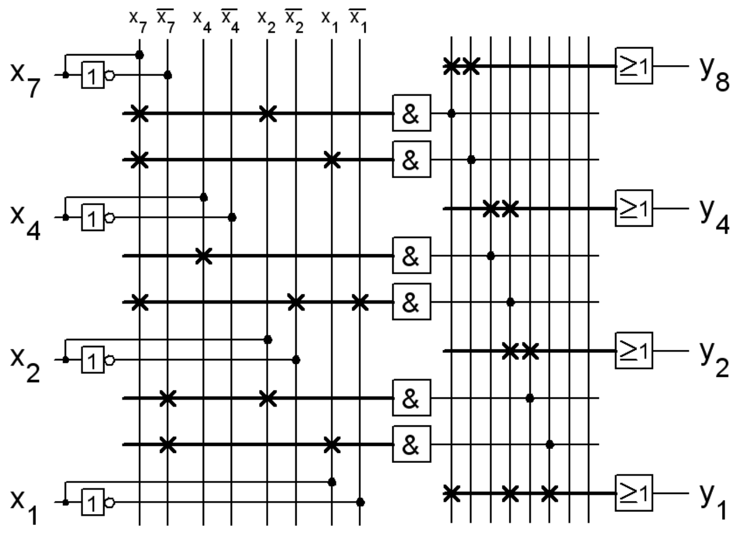
	7	4	2	1		8	4	2	1
	x_7	x_4	x_2	x_1		y_8	y_4	y_2	y_1
(0)	0	0	0	0	0	0	0	0	0
(1)	0	0	0	1	1	0	0	0	1
(2)	0	0	1	0	2	0	0	1	0
(3)	0	0	1	1	3	0	0	1	1
(4)	0	1	0	0	4	0	1	0	0
(5)	0	1	0	1	5	0	1	0	1
(6)	0	1	1	0	6	0	1	1	0
(8)	1	0	0	0	7	0	1	1	1
(9)	1	0	0	1	8	1	0	0	0
(10)	1	0	1	0	9	1	0	0	1

$$y_8 = x_7x_2 + x_7x_1$$

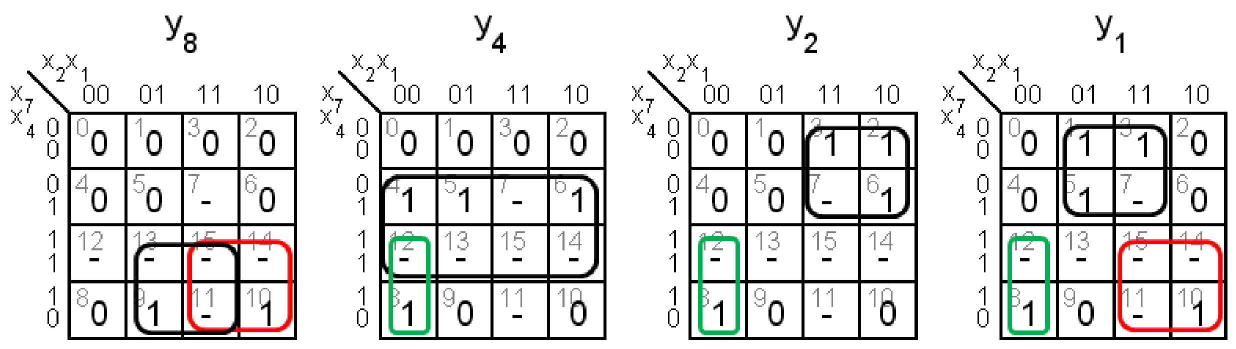
$$y_4 = x_4 + x_7\overline{x_2}\,\overline{x_1}$$

$$y_2 = \overline{x_7}\,x_2 + x_7\overline{x_2}\,\overline{x_1}$$

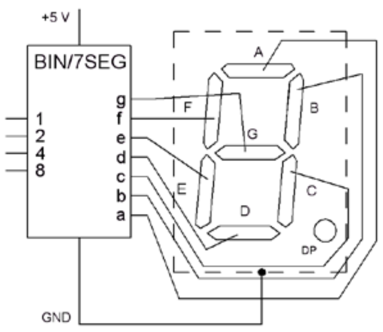
$$y_1 = \overline{x_7}\,x_1 + x_7x_2 + x_7\overline{x_2}\,\overline{x_1}$$



Two of the AND-gates can be common to the y_8,y_1 and y_8,y_2,y_1 -circuits.



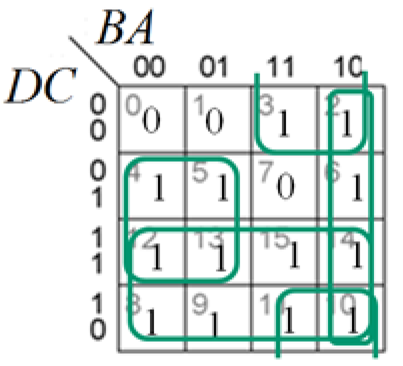
8.5



D C B A	a b c d e f g
0 0 0 0	1 1 1 1 1 1 0
0 0 0 1	0 1 1 0 0 0 0
0 0 1 0	1 1 0 1 1 0 1
0 0 1 1	1 1 1 1 0 0 1
0 1 0 0	0 1 1 0 0 1 1
0 1 0 1	1 0 1 1 0 1 1
0 1 1 0	1 0 1 1 1 1 1
0 1 1 1	1 1 1 0 0 0 0
1 0 0 0	1 1 1 1 1 1 1
1 0 0 1	1 1 1 0 0 1 1
1 0 1 0	1 1 1 0 1 1 1
1 0 1 1	0 0 1 1 1 1 1
1 1 0 0	0 0 0 1 1 0 1
1 1 0 1	0 1 1 1 1 0 1
1 1 1 0	1 0 0 1 1 1 1
1 1 1 1	1 0 0 0 1 1 1

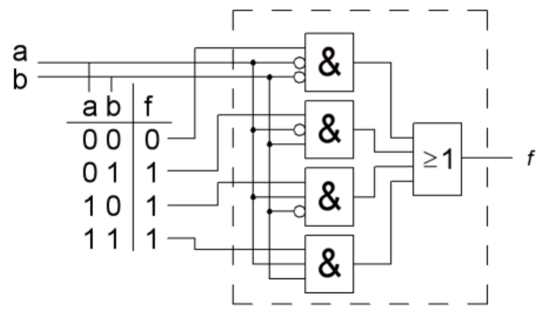
segment

$$g = D + \overline{B}C + B\overline{C} + B\overline{A}$$



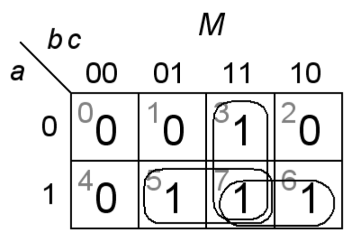
8.6

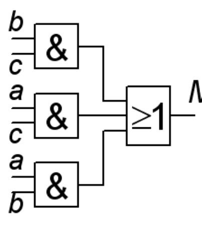
A 4-1 multiplexer used as function generator for the OR-function.

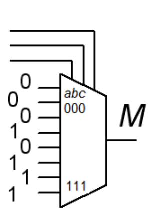


8.7

	a	b	c	M	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	$\bar{a}bc$
4	1	0	0	0	
5	1	0	1	1	$a\bar{b}c$
6	1	1	0	1	$ab\bar{c}$
7	1	1	1	1	abc

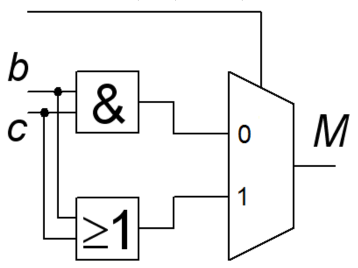






$$M = bc + ac + ab$$

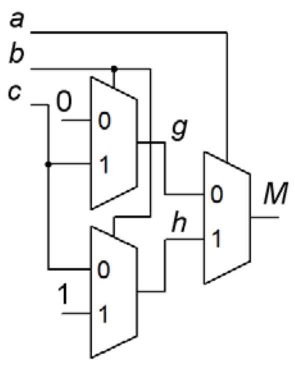
$$M = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc = \bar{a}(bc) + a(\bar{b}c + b\bar{c} + bc) = \bar{a}(bc) + a(b+c)$$



$$M = \bar{a}(bc) + a(b+c) \quad g = bc \quad h = b + c$$

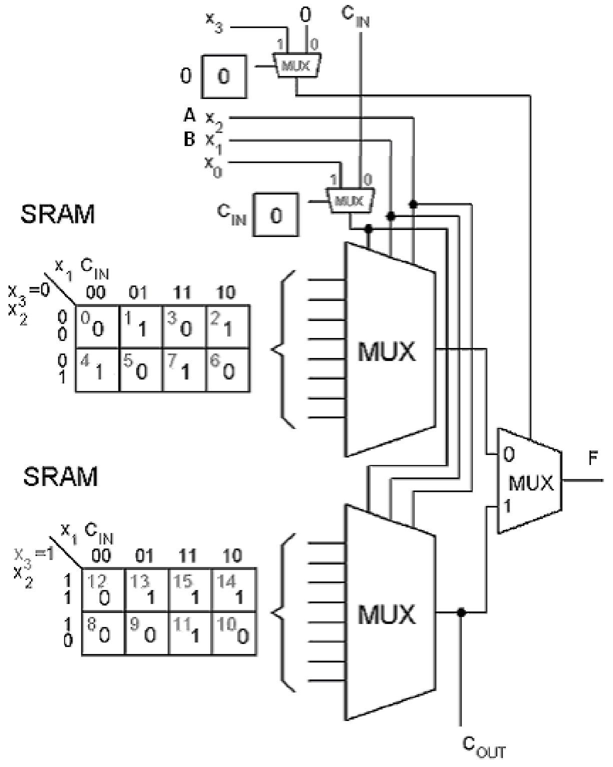
$$g = \bar{b}(0) + b(c) \quad = bc$$

$$h = \bar{b}(c) + b(1) \quad = \bar{b}c + b(\bar{c} + c) = \bar{b}c + b\bar{c} + bc = c(b + \bar{b}) + b(c + \bar{c}) = b + c$$

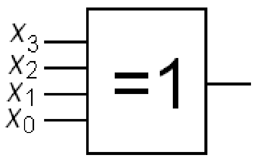


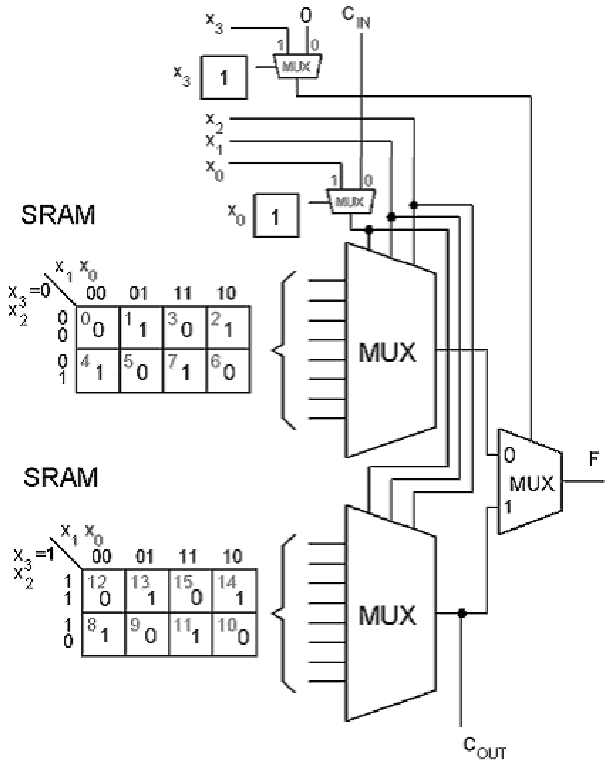
8.8
In order to make a full adder we need to use the the upper MUX to the sum function, and the bottom MUX, which is connected to C_{OUT} , is used for the Carry-function. In stead of x_0 we choose C_{IN}. In order for the upper MUX to be connected to the logic element output the output mux must be controlled with 0 instead of with x_3.

x_2	x_1	x_0		
A	B	C_{IN}	∑	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

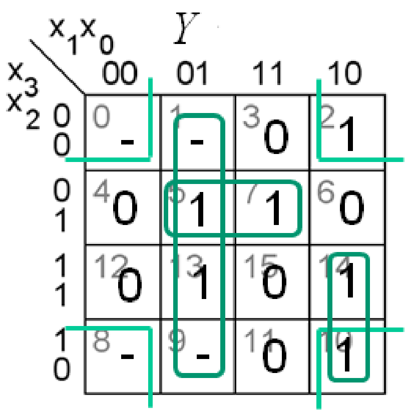


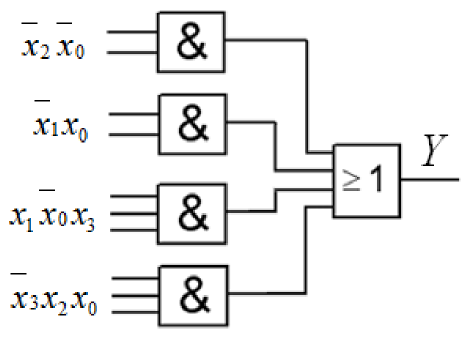
8.9



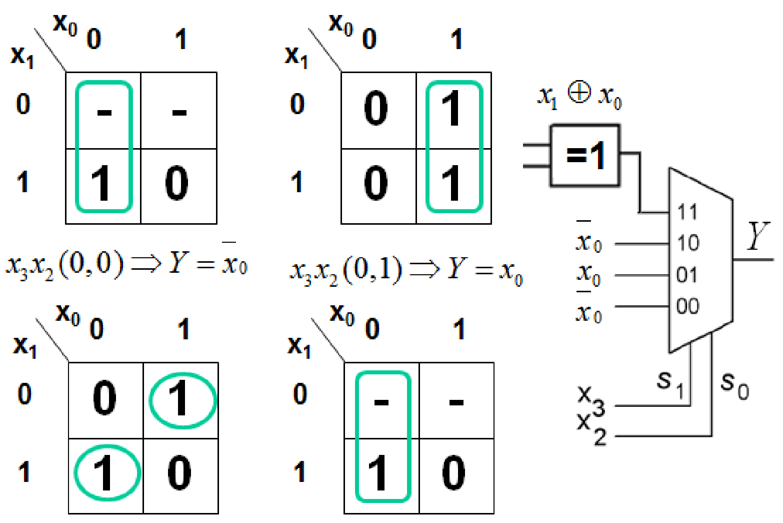


8.10





$$Y = \bar{x}_2\bar{x}_0 + \bar{x}_1 x_0 + x_1\bar{x}_0 x_3 + \bar{x}_3 x_2 x_0$$

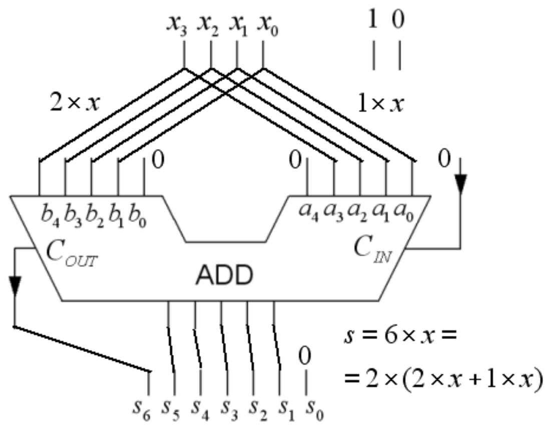


$x_3x_2(0,0) \Rightarrow Y = \bar{x}_0$ $x_3x_2(0,1) \Rightarrow Y = x_0$

$x_3x_2(1,1) \Rightarrow Y = x_1 \oplus x_0$ $x_3x_2(1,0) \Rightarrow Y = \bar{x}_0$

Alternatively, the XOR gate is also used to MUX input 00 and 01.

8.11 a)

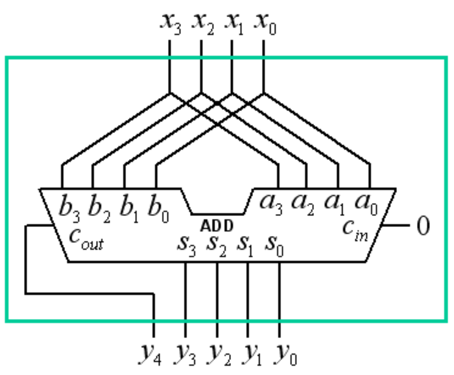


$s = 6\times x =$
$= 2\times(2\times x + 1\times x)$

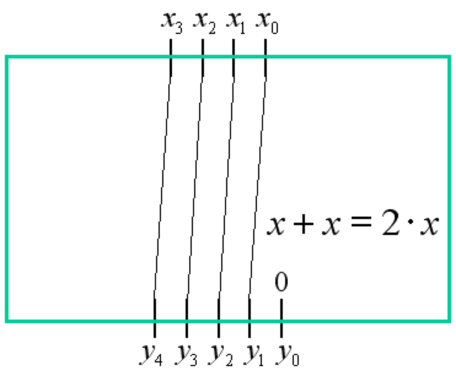
b) The greatest number will be $s_{max} = 6\cdot 15 = 90$. Since the calculator is not allowed at the exam, this time we can choose to transform 90 to a **binary number** in the same way as in a) (also other conversion method are ok):

$$\begin{array}{rcccccccl}
 & & \underline{1} & \underline{1} & \underline{1} & \underline{1} & \underline{0} & \underline{0} & \\
 & & 0 & 1 & 1 & 1 & 1 & 0 & 15\times 2 \\
 & + & 0 & 0 & 1 & 1 & 1 & 1 & 15\times 1 \\
\hline
 & & 1 & 0 & 1 & 1 & 0 & 1 & \\
1 & & 0 & 1 & 1 & 0 & 1 & 0 & \times 2
\end{array}$$

8.12



x_3 x_2 x_1 x_0
b_3 b_2 b_1 b_0
a_3 a_2 a_1 a_0
ADD
c_{out}
c_{in}
0
s_3 s_2 s_1 s_0
y_4 y_3 y_2 y_1 y_0

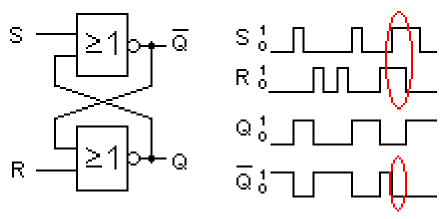


x_3 x_2 x_1 x_0
$x + x = 2 \cdot x$
0
y_4 y_3 y_2 y_1 y_0

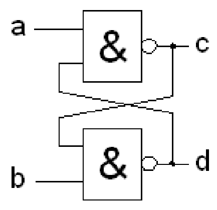
Sequential circuits, latches and clocked flip-flops

9.1

The figure shows a *SR*-latch, but at the end of the input-sequence the “forbidden” input combination $S = 1$, $R = 1$ will occur. The outputs Utgångarna will not be each others inverses for this combination. For NOR-gates is 1 a locking input signal, therefore $\overline{Q} = 0$ as long as *S* is left at 1.

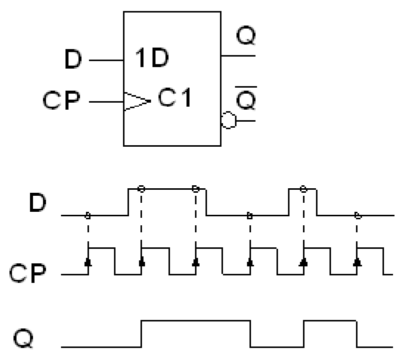


9.2

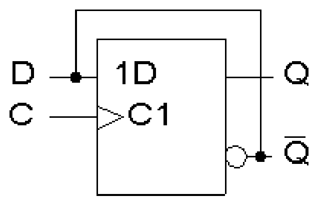


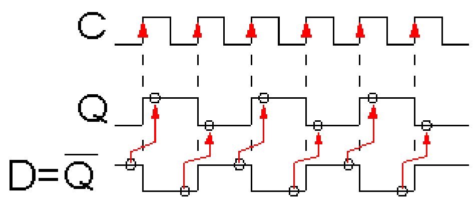
a b	c d
0 0	1 1
0 1	1 0
1 0	0 1
1 1	c d

9.3

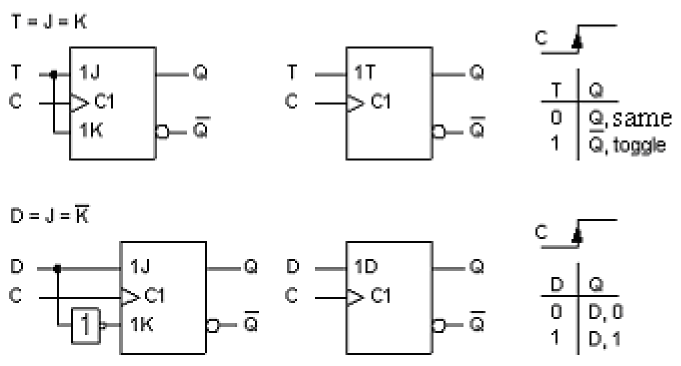


9.4



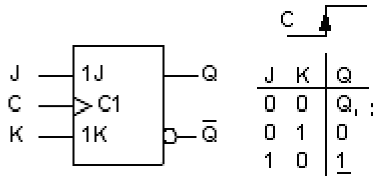


9.5



T	Q
0	Q, same
1	Q̄, toggle

D	Q
0	D, 0
1	D, 1



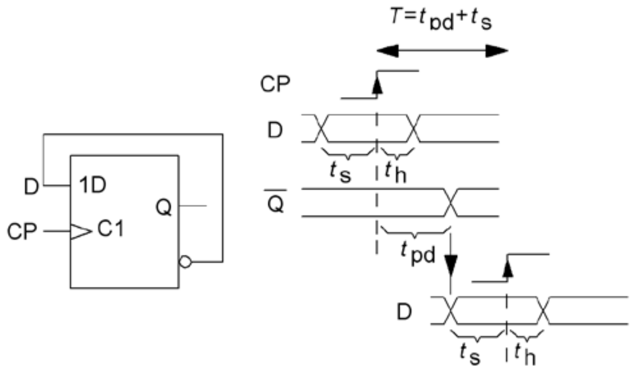
J	K	Q
0	0	Q, same
0	1	0
1	0	1
1	1	Q̄, toggle

JK-flip-flop can be used as T-flip-flop or as D-flip-flop. (When flip-flops are connected to each other there are usually the inverted outputs available, you will then not require the inverter to make the JK flip flop to D flip-flop.)

9.6

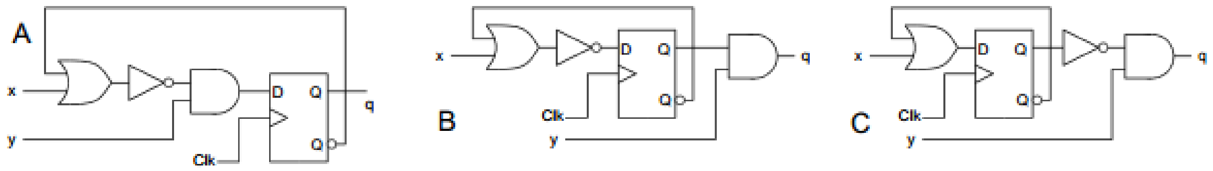
$$T = t_{pd} + t_s$$

$$f = \frac{1}{T} = \frac{1}{t_{pd} + t_s} = \frac{1}{(20+30)[\text{ns}]} = 20\text{ MHz}$$

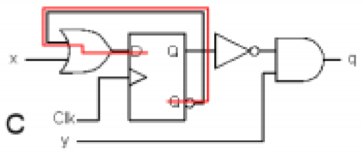


9.7

$t_{AND} = 0{,}4$ ns, $t_{OR} = 0{,}4$ ns, $t_{NOT} = 0{,}1$ ns, $t_{setup} = 0.3$ ns, $t_{dq} = 0{,}4$ns



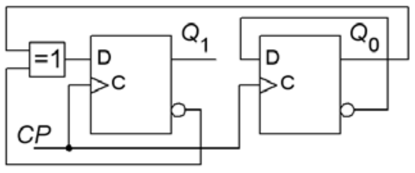
$$T = T_{OR} + T_{setup} + T_{dq} = 0{,}4 + 0{,}3 + 0{,}4 = 1{,}1\text{ ns}$$



9.8

a) The clock period is determined by the longest path, the one with the XOR-gate.
$T_{CP} > t_{pdQ} + t_{su} + t_{pdXOR} = 3+4+5 = \mathbf{12}$ **ns**.

b) Hold time is the time the data input must be stable after the clock edge. The flip-flop that has its *D*-input directly connected to the $\overline{Q}_0$ will get it's D-input changed first, directly after $t_{pdQ} = 3$ ns. $t_h < \mathbf{3}$ **ns**.



Sequential circuits

10.1

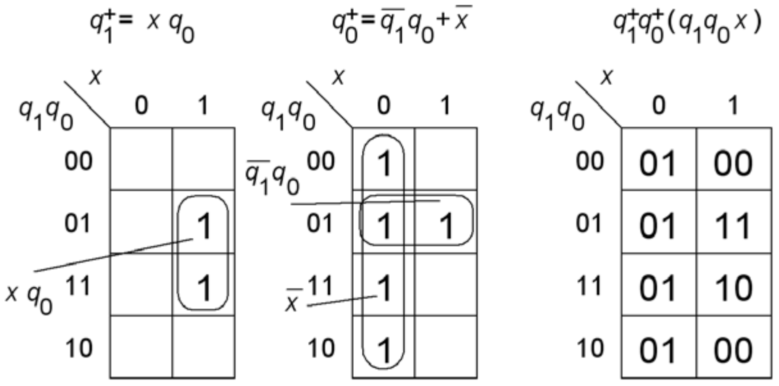
From the circuit diagram one can be derive the following expressions:

$q_1 \quad q_0$

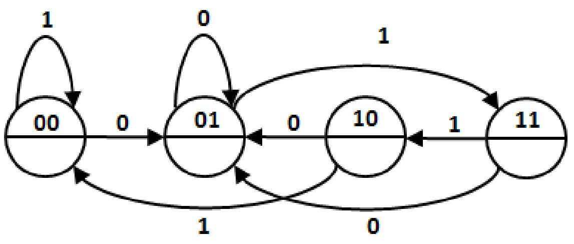
$$q_1^+ = x \cdot q_0$$

$$q_0^+ = \overline{x} + \overline{q_1} \cdot q_0$$

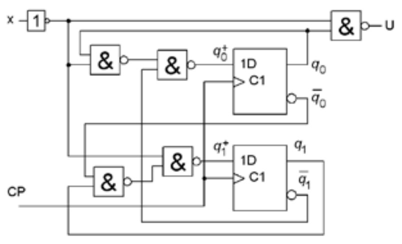
No output decoder exists the flip-flop state is directly used as output. Moore model is to be used.



q_1q_0 \ x	0	1
00	01	00
01	01	11
11	01	10
10	01	00



10.2



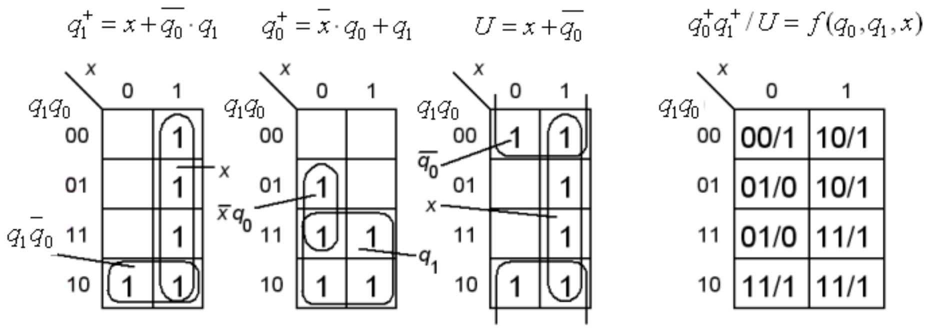
From the circuit diagram one can be derive the following expressions:

$$U = \overline{\overline{x} \cdot q_0} = x + \overline{q_0}$$

$$q_1^+ = \overline{\overline{q_1} \cdot \overline{(\overline{q_0} \cdot x)}} = x + \overline{q_0} \cdot q_1$$

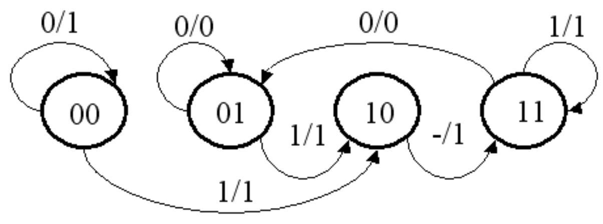
$$q_0^+ = \overline{\overline{(q_1 \cdot \overline{\overline{q_0}})} \cdot \overline{x}} = \overline{x} \cdot q_0 + q_1$$

Since U depends directly of x must Mealy model be used.



$q_1^+ = x + \overline{q_0} \cdot q_1$
$q_0^+ = \overline{x} \cdot q_0 + q_1$
$U = x + \overline{q_0}$
$q_0^+ q_1^+ / U = f(q_0, q_1, x)$
x
$q_1\overline{q_0}$
$\overline{x}\,q_0$
q_1
$\overline{q_0}$
x

q_1q_0 \ x	0	1
00	00/1	10/1
01	01/0	10/1
11	01/0	11/1
10	11/1	11/1



10.3

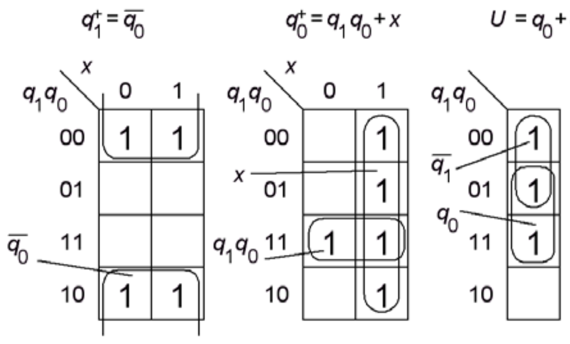
From the circuit diagram one can be derive the following expressions:

$$U = \overline{\overline{q_0} \cdot q_1} = q_0 + \overline{q_1}$$

$$q_1^+ = \overline{q_0}$$

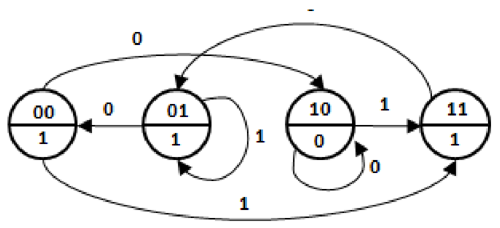
$$q_0^+ = \overline{\overline{x} \cdot \overline{q_0 \cdot q_1}} = q_1 q_0 + x$$

Since *U* only depends on the state and is is independent of *x* must Moore model be used.

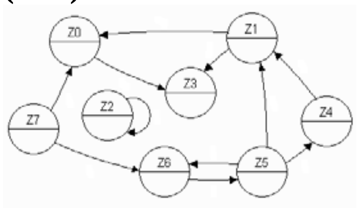


$q_1^+ q_0^+ (q_1 q_0 x)$

q_1q_0 \ x	0	1
00	10	11
01	00	01
11	01	01
10	10	11

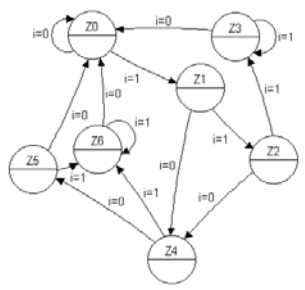


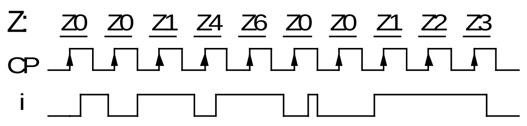
(10.4)



- Stopping condition: Z3
- Loss condition: Z7
- Isolated states: Z2

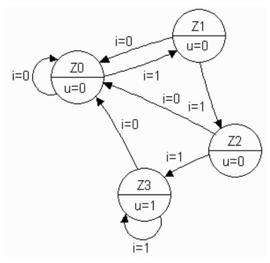
10.5





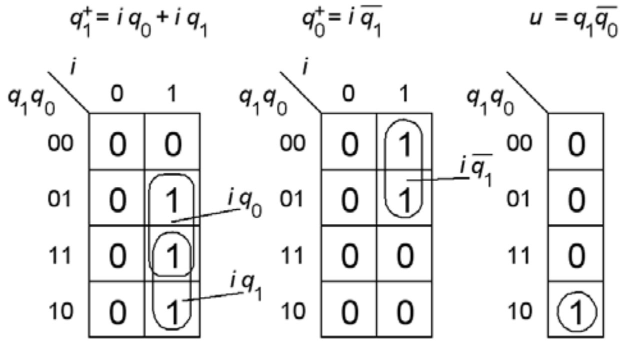
10.6

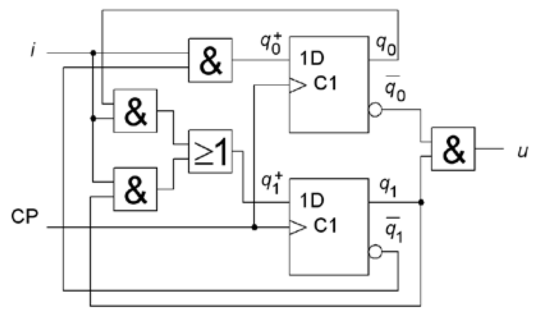
From state diagram to coded state table.



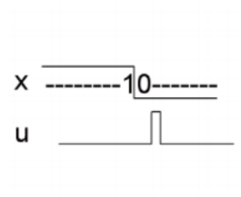
$q_1^+ q_0^+ (q_1 q_0 i)$

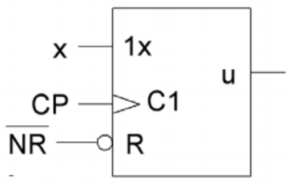
q_1q_0 \ i	0	1
Z0: 00	00	01
Z1: 01	00	11
Z2: 11	00	10
Z3: 10	00	10

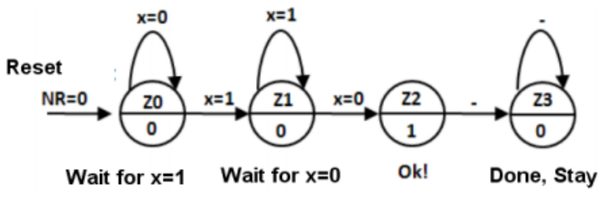




10.7







State encoding Binary:

Bin $q_1^+q_0^+(q_1q_0x)$

q_1q_0 \ x	0	1
Z0: 00	00	01
Z1: 01	10	01
Z3: 11	11	11
Z2: 10	11	11

$q_1^+ = q_0\overline{x} + q_1$

q_1q_0 \ x	0	1
00	0	0
$q_0\overline{x}$ 01	1	0
q_1 11	1	1
10	1	1

$q_0^+ = q_1 + x$

q_1q_0 \ x	0	1
00	0	1
x 01	0	1
q_1 11	1	1
10	1	1

$U = q_1\overline{q}_0$

q_1q_0	
00	0
01	0
$q_1\overline{q}_0$ 11	0
10	1

State encoding Gray:

Gray $q_1^+q_0^+(q_1q_0x)$

q_1q_0 \ x	0	1
Z0: 00	00	01
Z1: 01	11	01
Z2: 11	10	10
Z3: 10	10	10

$q_1^+ = q_0\overline{x} + q_1$

q_1q_0 \ x	0	1
00	0	0
$q_0\overline{x}$ 01	1	0
q_1 11	1	1
10	1	1

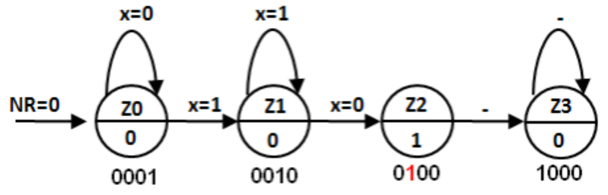
$q_0^+ = \overline{q}_1q_0 + \overline{q}_1x$

q_1q_0 \ x	0	1
$\overline{q}_1x$ 00	0	1
$\overline{q}_1q_0$ 01	1	1
11	0	0
10	0	0

$U = q_1q_0$

q_1q_0	
00	0
q_1q_0 01	0
11	1
10	0

State encoding One hot:



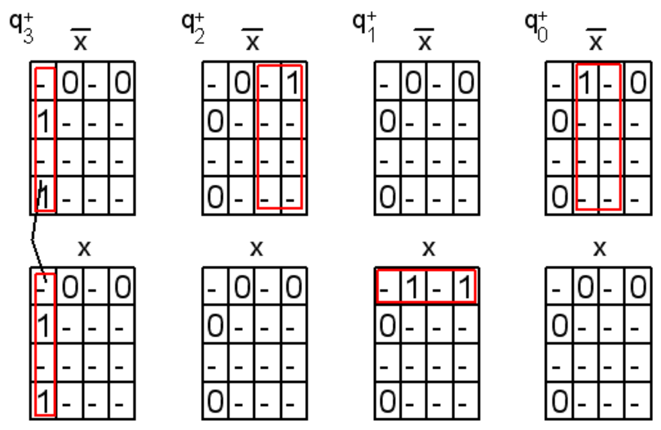
$q_3^+q_2^+q_1^+q_0^+(x\,q_3\,q_2\,q_1\,q_0)$

$\overline{x}$

$q_3q_2 \backslash q_1q_0$	00	01	11	10
00	^{0} -	^{1} 0001	^{3} -	^{2} 0100
01	^{4} 1000	^{5} -	^{7} -	^{6} -
11	^{12} -	^{13} -	^{15} -	^{14} -
10	^{8} 1000	^{9} -	^{11} -	^{10} -

x

$q_3q_2 \backslash q_1q_0$	00	01	11	10
00	^{0} -	^{17} 0010	^{19} -	^{18} 0010
01	^{20} 1000	^{21} -	^{23} -	^{22} -
11	^{28} -	^{29} -	^{31} -	^{30} -
10	^{24} 1000	^{25} -	^{27} -	^{26} -



q_3^+ $\overline{x}$

-	0	-	0
1	-	-	-
-	-	-	-
1	-	-	-

q_3^+ x

-	0	-	0
1	-	-	-
-	-	-	-
1	-	-	-

q_2^+ $\overline{x}$

-	0	-	1
0	-	-	-
-	-	-	-
0	-	-	-

q_2^+ x

-	0	-	0
0	-	-	-
-	-	-	-
0	-	-	-

q_1^+ $\overline{x}$

-	0	-	0
0	-	-	-
-	-	-	-
0	-	-	-

q_1^+ x

-	1	-	1
0	-	-	-
-	-	-	-
0	-	-	-

q_0^+ $\overline{x}$

-	1	-	0
0	-	-	-
-	-	-	-
0	-	-	-

q_0^+ x

-	0	-	0
0	-	-	-
-	-	-	-
0	-	-	-

$$q_3^+ = \overline{q}_1\overline{q}_0$$

$$q_2^+ = q_1\overline{x}$$

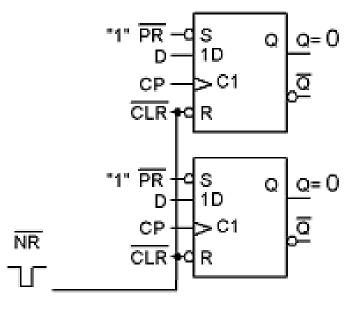
$$q_1^+ = \overline{q}_3\overline{q}_2x$$

$$q_0^+ = q_0\overline{x}$$

$$u = q_2$$

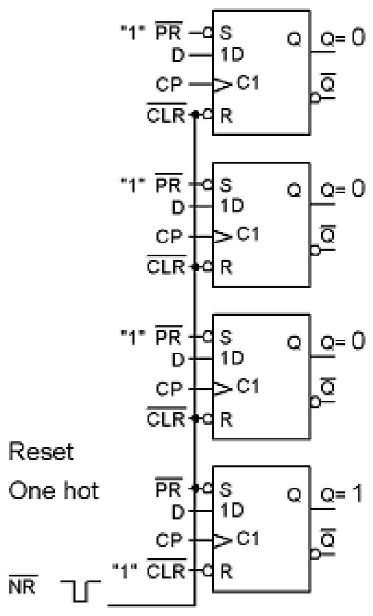
This is how to do reset to ”00” with CLR inputs for the Bin/Gray state encoding, and to ”0001” with PRE/CLR inputs for the ”one hot” state encoding.

Reset Bin/Gray



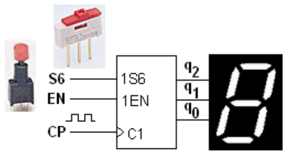
Reset

One hot



10.8

Six state requires three flip-flops. There are 8 states in total, two states which are not included in the sequence. To be on the safe side we specify what should happen with these states, so that the counter will not get "stuck" at Z0 or Z7.



$q_2^+q_1^+q_0^+(q_2q_1q_0)$

q_2q_1 \ q_0	0	1
00	001	010
01	011	100
11	001	001
10	101	110

q_2^+

0	0
0	1
0	0
1	1

q_1^+

0	1
1	0
0	0
0	1

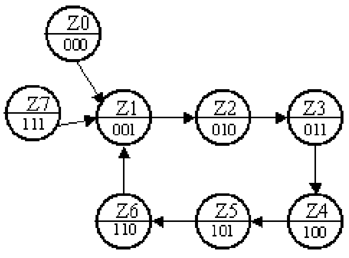
q_0^+

1	0
1	0
1	1
1	0

$$q_2^+ = q_2\bar{q}_1 + \bar{q}_2 q_1 q_0$$

$$q_1^+ = \bar{q}_1 q_0 + \bar{q}_2 q_1 \bar{q}_0$$

$$q_0^+ = \bar{q}_0 + q_2 q_1$$



Equations with EN (EN=0 → next state same) :

$$(q_2^+)' = EN\cdot(q_2^+) + \overline{EN}\cdot(q_2)$$

$$(q_1^+)' = EN\cdot(q_1^+) + \overline{EN}\cdot(q_1)$$

$$(q_0^+)' = EN\cdot(q_0^+) + \overline{EN}\cdot(q_0)$$

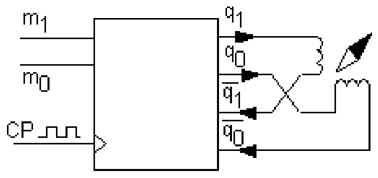
Equations with S6 (S6 = 1 → next state is 110) :

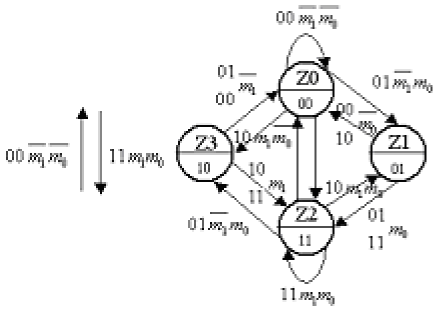
$$(q_2^+)'' = (q_2^+)' + S6$$

$$(q_1^+)'' = (q_1^+)' + S6$$

$$(q_0^+)'' = (q_0^+)'\cdot\overline{S6}$$

10.9





$q_1^+q_0^+(q_1q_0m_1m_0)$

q_1q_0 \ m_1m_0	00	01	11	10
00	00	01	11	10
01	00	11	11	00
11	00	10	11	01
10	00	00	11	11

q_1^+

0	0	1	1
0	1	1	0
0	1	1	0
0	0	1	1

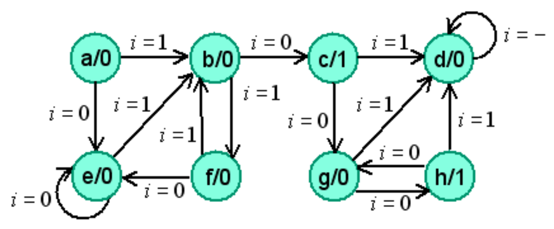
q_0^+

0	1	1	0
0	1	1	0
0	0	1	1
0	0	1	1

$$q_1^+ = q_0 m_0 + \bar{q}_0 m_1$$

$$q_0^+ = q_1 m_1 + \bar{q}_1 m_0$$

10.10



Write down the state table

now	next		out
$i=$	0	1	z
a	e	b	0
b	c	f	0
c	g	d	1
d	d	d	0
e	e	b	0
f	e	b	0
g	h	d	0
h	g	d	1

Two states can not be equivalent if the output is different or if subsequent state output is different.

Groups with the same ouput:

$$P_1 = (\boxed{a,b,d,e,f,g})(c,h)$$

Examine subsequent state:

$$a_{i=0} \to (a,b,d,\mathrm{e},f,g) \quad a_{i=1} \to (a,\mathrm{b},d,e,f,g)$$
$$\boxed{b_{i=0} \to (\mathrm{c},h) \quad b_{i=1} \to (a,b,d,e,\mathrm{f},g)}$$
$$d_{i=0} \to (a,b,\mathrm{d},e,f,g) \quad d_{i=1} \to (a,b,\mathrm{d},e,f,g)$$
$$e_{i=0} \to (a,b,d,\mathbf{e},f,g) \quad e_{i=1} \to (a,\mathbf{b},d,e,f,g)$$
$$f_{i=0} \to (a,b,d,\mathbf{e},f,g) \quad f_{i=1} \to (a,\mathbf{b},d,e,f,g)$$
$$\boxed{g_{i=0} \to (c,\mathbf{h}) \quad g_{i=1} \to (a,b,\mathbf{d},e,f,g)}$$

$(b,\ g)$ forms a group of them self.

$$P_2 = (a,d,e,f)(\boxed{b,g})(c,h)$$

$$P_2 = (\boxed{a,d,e,f})(b,g)(c,h)$$

Examine subsequent state :

$$a_{i=0} \to (a,d,\mathbf{e},f) \quad a_{i=1} \to (\mathbf{b},g)$$
$$\boxed{d_{i=0} \to (a,\mathbf{d},e,f) \quad d_{i=1} \to (a,\mathbf{d},e,f)}$$
$$e_{i=0} \to (a,d,\mathbf{e},f) \quad e_{i=1} \to (\mathbf{b},g)$$
$$f_{i=0} \to (a,d,\mathbf{e},f) \quad f_{i=1} \to (\mathbf{b},g)$$

(d) Forms a group of it self.

$$P_3 = (a,e,f)(b,g)(\boxed{d})(c,h)$$

$$P_3 = (a,e,f)(\boxed{b,g})(d)(c,h)$$

Examine subsequent state :

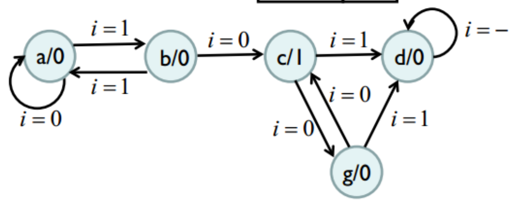
$$\boxed{\begin{array}{l} \boxed{b_{i=0} \rightarrow (\mathbf{c},h) \quad b_{i=1} \rightarrow (a,e,\mathbf{f})} \\ \boxed{g_{i=0} \rightarrow (c,\mathbf{h}) \quad g_{i=1} \rightarrow (\mathbf{d})} \end{array}}$$

(b) (g) forms own groups.

$$P_4 = (a,e,f)(\boxed{b})(d)(\boxed{g})(c,h)$$

$$P_4 = (a,e,f)(b)(d)(g)(c,h)]$$

Changing names	now	next		out
	$i=$	0	1	z
$(a,e,f) \Rightarrow a$	a	a	b	0
$(b) \Rightarrow b$	b	c	a	0
$(c,h) \Rightarrow c$	c	g	d	1
$(d) \Rightarrow d$	d	d	d	0
$(g) \Rightarrow g$	g	c	d	0



10.11

	next		out
$i:$	0	1	z
a	e	c	0
b	d	c	0
c	d	e	0
d	f	a	1
e	f	b	1
f	f	b	0

Groups with same output:

$P_1 = (a,b,c,f)(d,e)$

Examine next states:

$a_{i=0} \rightarrow (d,e) \quad a_{i=1} \rightarrow (a,b,c,f)$

$b_{i=0} \rightarrow (d,e) \quad b_{i=1} \rightarrow (a,b,c,f)$

$c_{i=0} \rightarrow (d,e) \quad c_{i=1} \rightarrow (d,e)$

$f_{i=0} \rightarrow (a,b,c,f) \quad f_{i=1} \rightarrow (a,b,c,f)$

$d_{i=0} \rightarrow (a,b,c,f) \quad d_{i=1} \rightarrow (a,b,c,f)$

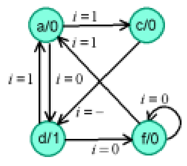
$e_{i=0} \rightarrow (a,b,c,f) \quad e_{i=1} \rightarrow (a,b,c,f)$

Groups with same subsequent state output:

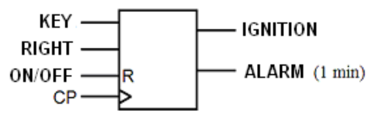
$P_2 = (a,b)(c)(f)(d,e)$

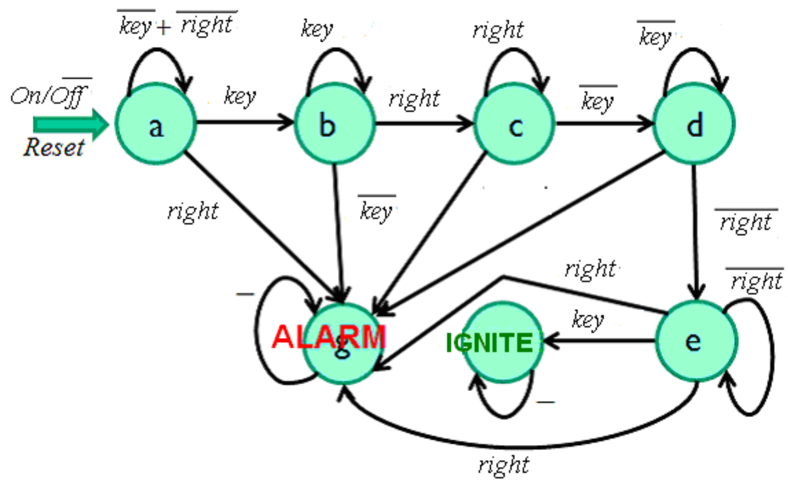
$P_3 = P_2$

		next		out
		0	1	z
$(a,b) \rightarrow a$	a	d	c	0
$(c) \rightarrow c$	c	d	d	0
$(f) \rightarrow f$	d	f	a	1
$(d,e) \rightarrow d$	f	f	a	0



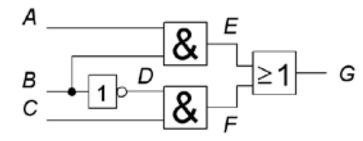
10.12

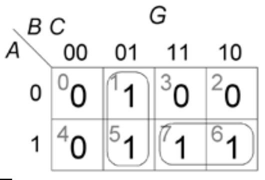


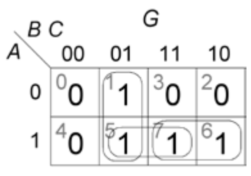


Asynchronous sequential circuits

11.1



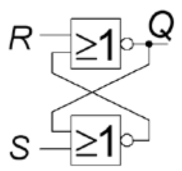


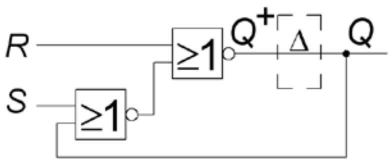


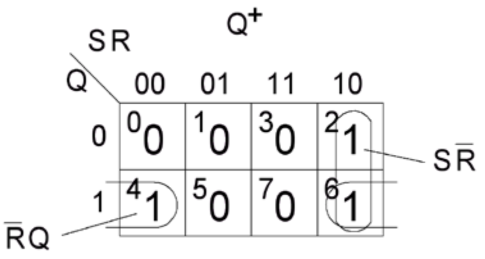
$G = \bar{B}C + AB$ $\{Hazardfree\}$ $G = \bar{B}C + AB + AC$

11.2

To the left is a SR-latch made by two gates with feedback. To the right the circuit is drawn as a Moore-compatible statemachine.
Moore-machine.



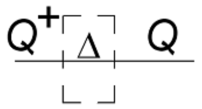




$$Q^+ = \overline{R + \overline{S+Q}} = \bar{R} \cdot \overline{\overline{(S+Q)}} = \bar{R} \cdot (S+Q) = S\bar{R} + \bar{R}Q$$

When dealing with asynchronous statemachines the coded state table is used to be named **excitation table**.

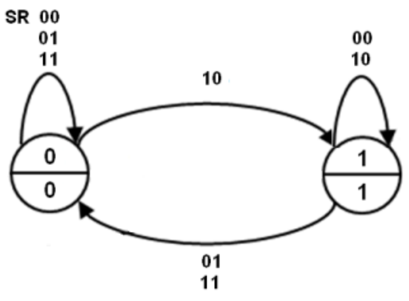
Present state Q	Next state Q^+			
	Input signals SR			
	00	01	11	10
0	0	0	0	1
1	1	0	0	1



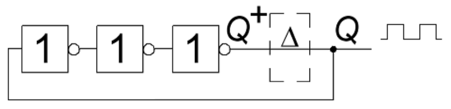
For each input (column), there must be at least one state where $Q = Q^+$. Such conditions are stable and they are usually marked by a circle.
The state diagram follows from the exitation table.

The state table is named **flow table** when working with asynchronous state machines..

Present state Q	Next state Q^+			
	Input signals SR			
	00	01	11	10
A	A	A	A	B
B	B	A	A	B



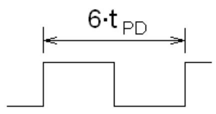
11.3



$Q^+ = \bar{Q}$

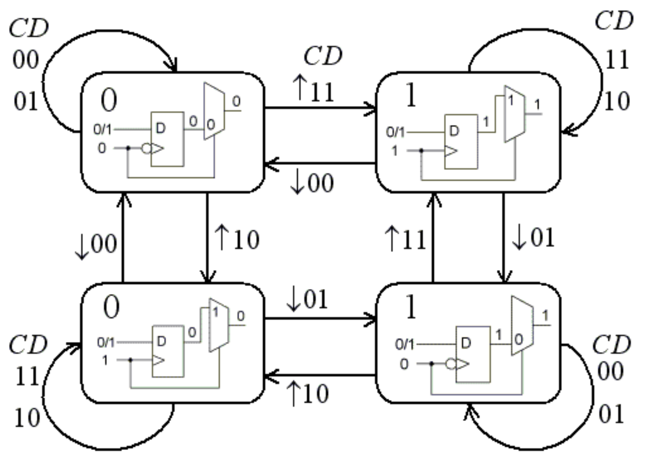
Q	Q^+
0	1
1	0

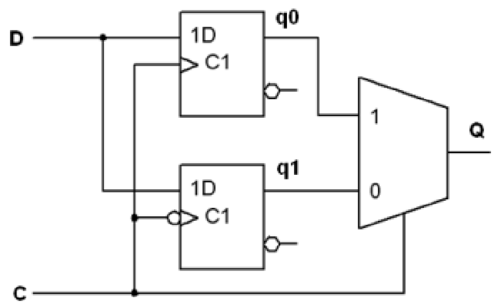
No stable states!

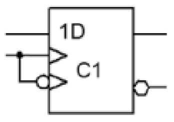


$$T = 6 \cdot t_{PD} \Rightarrow f = \frac{1}{6 \cdot t_{PD}}$$

11.4







- At positive edge ↑ **C** changes from 0 to 1 and when **C**=1 the MUX connects the upper flip-flop **q0** to the output.
- At negative edge ↓ **C** changes from 1 to 0 and **C**=0 the MUX connects the lower flip-flop **q1** to the output.

The result is a D-flip-flop that reacts on both edges of the clock.

11.5

There are four input combinations (CD) and two output combinations (Q). A total of 8 possible states (CDQ).:

Possible input/output combinations

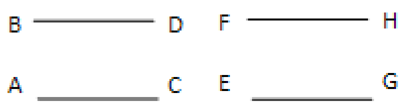
Present state		Next state		Comment
State tag	CDO	(CDO)+	(CDO)+	
A	000	010	100	Output O gets D input value when C changes value
B	001	011	100	
C	010	000	111	No change of O when D changes value
D	011	001	111	
E	100	000	110	
F	101	000	111	
G	110	011	100	
H	111	011	101	

Flow table (stabile states marked in bold font)

Present state	Next State (CD)				Output
	00	01	11	10	
A	**A**	C	-	E	0
B	**B**	D	-	E	1
C	A	**C**	H	-	0
D	B	**D**	H	-	1
E	A	-	G	**E**	0
F	A	-	H	**F**	1
G	-	D	**G**	E	0
H	-	D	**H**	F	1

We see immediately that no minimization may be done by state equivalence classes, because all eight states have different outputs where they have stable states, and where they have do not cares in the table.

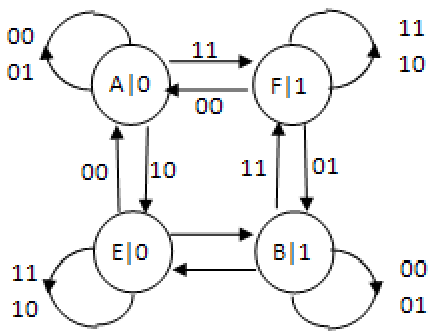
Merger-diagram:



The minized flow table

Flow Table (stable states marked as bold)

Present state	Next State (CD)				Output
	00	01	11	10	
A	**A**	**A**	F	E	0
B	**B**	**B**	F	E	1
E	A	B	**E**	**E**	0
F	A	B	**F**	**F**	1



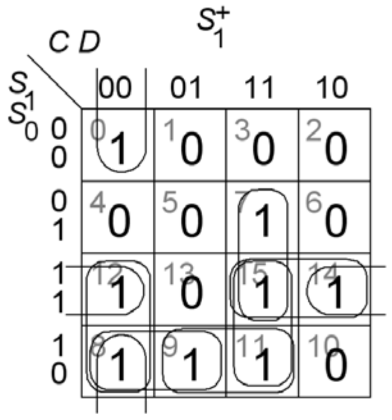
c) Assign states, do Karnaugh-minimization and derive the boolean equations. Possible state assignements are (E=00, B=01, A=10, F=11) and their rotations and mirror solutions:

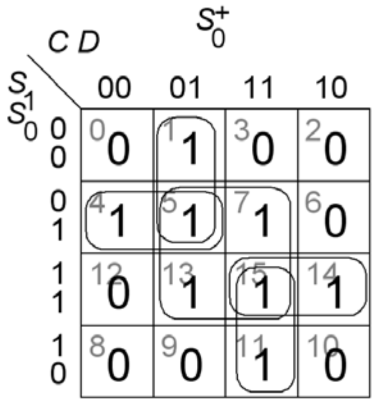
Possible state assignments			
A	F	B	E
00	01	11	10
01	11	10	00
11	10	00	01
10	00	01	11
11	01	00	10
01	00	10	11
00	10	11	01
10	11	01	00

One of the resulting state tables

Flow Table (stable states marked as bold)					
Present state	Next State (CD)				Output
	00	01	11	10	
10	**10**	**10**	11	00	0
01	**01**	**01**	11	00	1
00	10	01	**00**	**00**	0
11	10	01	**11**	**11**	1

And the corresponding Karnaugh diagrams and Boolean expressions becomes:





$$S_1^+ = CD(S_1 + S_0) + \overline{C}\,\overline{D}(S_1 + \overline{S_0}) + S_1 S_0 (C + \overline{D})$$

$$S_0^+ = S_0 D + \overline{S_1}\,S_0 \overline{C} + \overline{S_1}\,\overline{C}\,D + S_1 CD + S_1 S_0 C$$

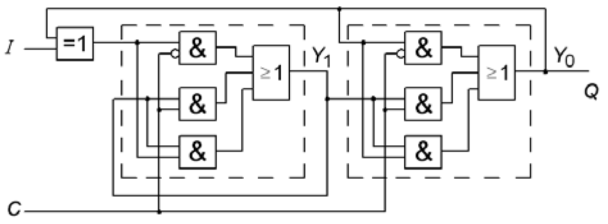
$$O = S_0$$

11.6

a) Derive the Boolean expressions for the state variables. Answer:

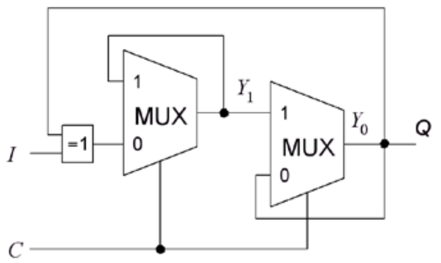
$$Y_0^+ = Y_0Y_1 + Y_0\overline{C} + Y_1C$$

$$Y_1^+ = Y_1(Y_0 \oplus I) + (Y_0 \oplus I)\overline{C} + \overline{Y_1}C$$



b Derive the exitations table. Which function (dashed) are in the inner loops.

The two inner loops are hazard-free MUX:es!



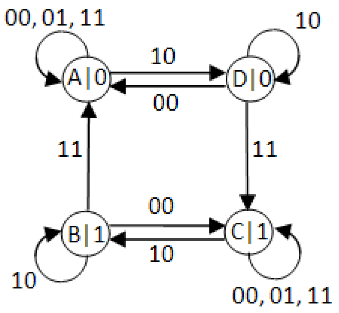
Pres. State	Next State				Q
	IC=				
Y_1Y_0	00	01	11	10	
00	**00**	**00**	**00**	10	0
01	11	~~00~~	00	**01**	1
11	**11**	**11**	**11**	01	1
10	00	~~11~~	11	**10**	0

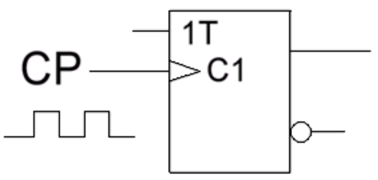
Stable states marked in ***Bold***
Impossible transitions marked as ~~*strikethrough*~~

Derive the flow table, assign symbolic states and drav FSM.

Pres. State	Next State				Q
	IC=				
Y_1Y_0	00	01	11	10	
A	**A**	**A**	**A**	D	0
B	C	~~A~~	A	**B**	1
C	**C**	**C**	**C**	B	1
D	A	~~C~~	C	**D**	0

Stable states marked in ***Bold***
Impossible transitions marked as ~~*strikethrough*~~

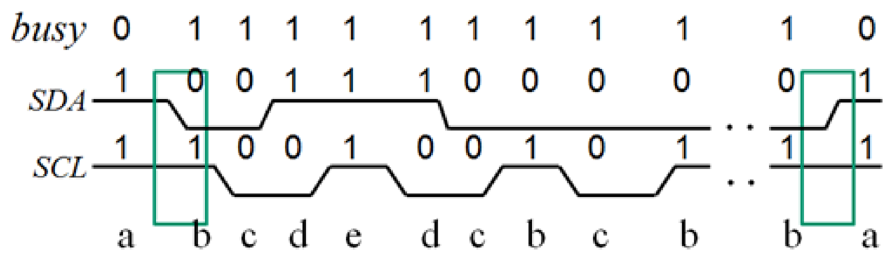




Identify the function of the asynchronous circuit. Which flip-flop is it?

- Positive edge-triggered T-flip-flop.

11.7

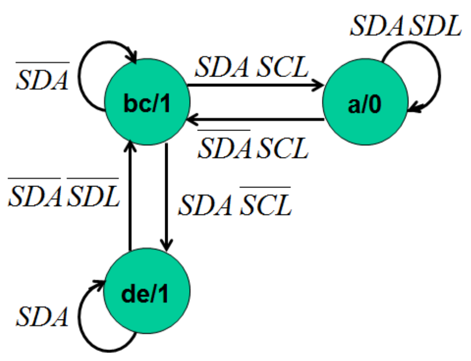


busy 0 1 1 1 1 1 1 1 1 1 1 0
SDA 1 0 0 1 1 1 0 0 0 0 0 1
SCL 1 1 0 0 1 0 0 1 0 1 1 1
a b c d e d c b c b b a

	00	01	11	10	q
	SDA	SCL			
a	–	b	a	*	0
b	c	b	a	–	1
c	c	b	–	d	1
d	c	–	e	d	1
e	–	*	e	d	1

Folow the timing diagram and create a new state for every combination that has not been before. In state *a* we ”waits” for the startedge (*b*), then input 10 is impossible (marked with *). The Protocol prohibits change of data SDA when SCL is **high**. Therefore input 01 is impossible in state *e* (marked with *). This gives us two extra don’t care positions in the table. You can directly see which states that can be merged.

	00	01	11	10	q
	SDA	SCL			
a	–	bc	a	–	0
bc	bc	bc	a	de	1
de	bc	–	de	de	1



As state code assignement the Gray-code can be used. *a* 00, *bc* 01, *de* 11, and *x* 10. *x* can be used as don’t care exept from 10.

q_1q_0	00	01	11	10	$busy$
	SDA	SCL			
00	–	01	00	–	0
01	01	01	00	11	1
11	01	–	11	11	1
10	≠10	≠10	≠10	≠10	–

$q_1^+ q_0^+$

$busy = q_0$

q_1q_0	00	01	11	10
	SDA	SCL		
00	–	0	0	–
01	0	0	0	1
11	0	–	1	1
10	–	–	–	–

$$q_1^+ = SDA(\overline{SCL} + q_1)$$

q_1q_0	00	01	11	10
	SDA	SCL		
00	–	1	0	–
01	1	1	0	1
11	1	–	1	1
10	–	– ≠10	–	–

$$q_0^+ = (\overline{SDA} + \overline{SCL} + q_1)$$

The groups are forming contiguous areas in Karnaughdiagram and therefore hazard free (if the networks have two levels). Realising with optional gates.

Address decoding of memories and I/O circuits

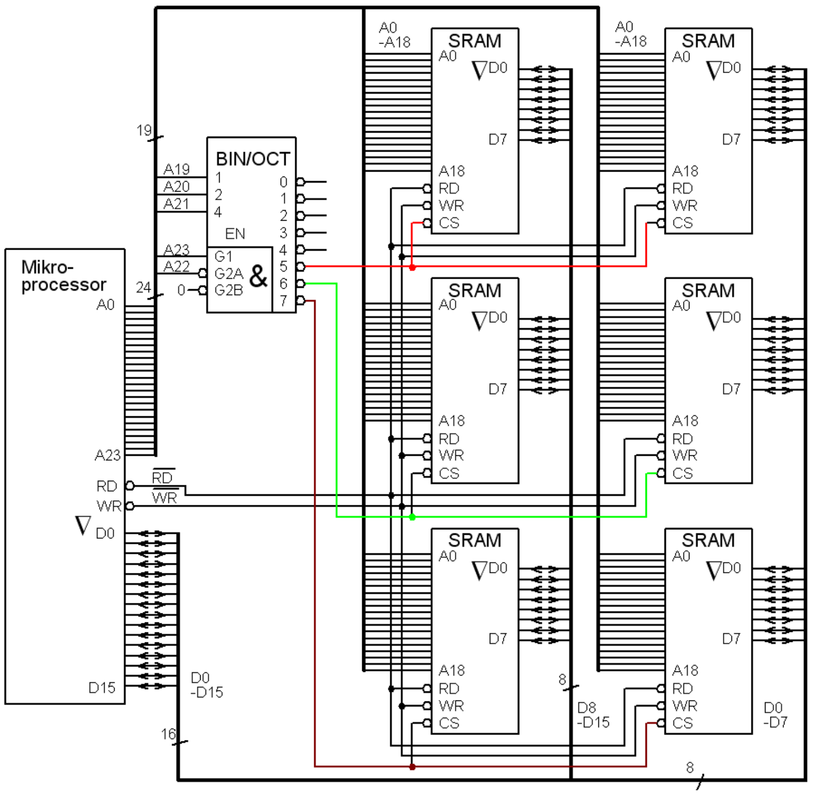
12.1

A dynamic RAM-memory consits of some 256Mbit memory chips organised as 32 M×8.
a) How many chips are needed for 256M×64?
Memory N = 256M M = 64 bit. **Chip** p = 32M q = 8 bit.
Number of columns $k = M/q = 64/8 = 8$.
Number of rows $r = N/p$ = 256M/32M = 8.
Total number of chips $K = r \times k = 8 \times 8 = 64$.

b) How many memory chips are needed for 512M×72? (what can be the reson for the unusual widh ”72” of the memory?)
Memory N = 512M M = 72 bit. **Chip** p = 32M q = 8 bit.
Number of columns $k = M/q = 72/8 = 9$.
Number of rows $r = N/p$ = 512M/32M = 16.
Total number of chips $K = r \times k = 9 \times 16 = 144$.
The unusual “width” 72 (= 64 + 8). The 8 *extra* bits are used for correcting single errors and detecting double errors. (Not shown in this course).

12.2

A certain 16 bit processor can address 24 bits. Memory Space is divided between ROM, SRAM and IO circuits. Address decoding is done using a 3:8-decoder.
a) How large is the RAM in the figure? What is the address range expressed in hexadecimal numbers?



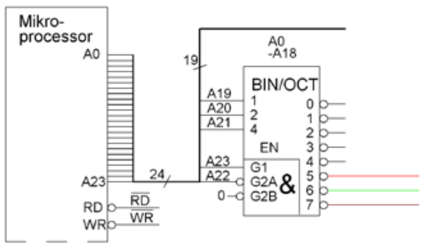
- **Memory chip**:

p = 512k q = 8 bitar

- **Memory**:

$r = 3$ $k = 2$ $K = 2 \times 3 = 6$
$M = k \times q = 2 \times 8 = 16$ bitar
$N = p \times r$ = 512k × 3 = 1,5M

Address range:



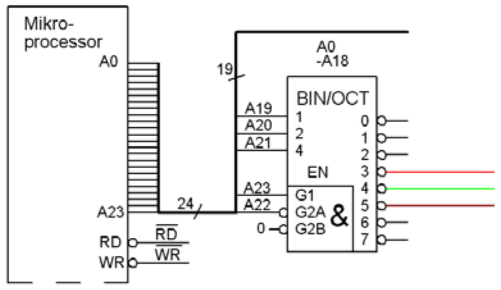
Computer:	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decoder:	1	0	0 … 7																					
Mem start:	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
● Begin hex	A				8				0				0				0				0			
Mem end:	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
● End hex	B				F				F				F				F				F			

b) How do you change the address range to 980000 – AFFFFF ?

980000
1001|1000|0000|0000|0000|0000|

AFFFFF
1010|1111|1111|1111|1111|1111|

"10|011" $\longrightarrow$ "3"
"10|101" $\longrightarrow$ "5"

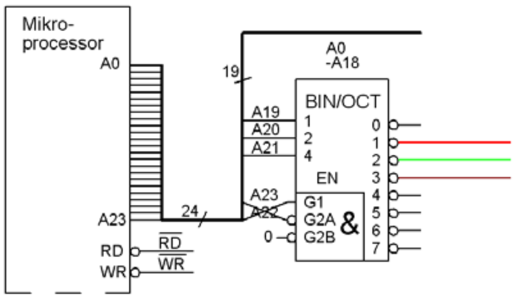


c) Change the address range to 480000 – 5FFFFF ?

480000
0100|1000|0000|0000|0000|0000|

5FFFFF
0101|1111|1111|1111|1111|1111|

"01|001" $\longrightarrow$ "1"
"01|011" $\longrightarrow$ "3"

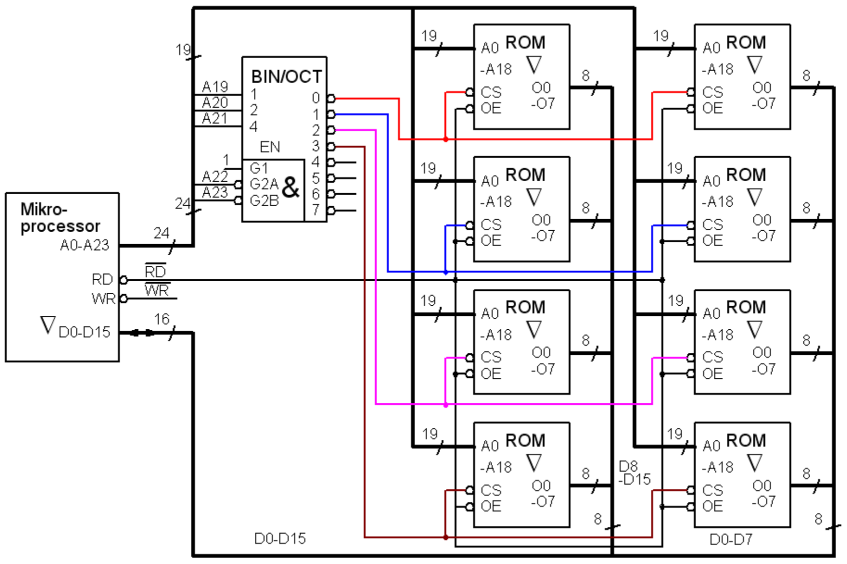


We Interchanges A23 and A22 !

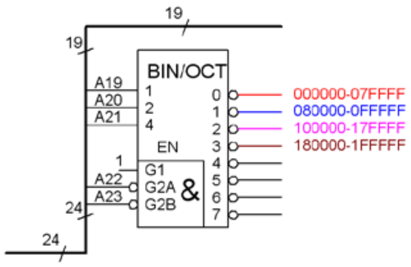
d) ROM-memory is 2M×16 bit and the address range is 000000 … and forward. ROM Chip is 512k×8.

- How many chips are needed?
- How is the decoder connected?
- How are the memory chips connected?
- Which is the address area for the ROM expressed in hexadecimal numbers.

Memory:
N = 2 M (4·512k) word is M = 16 bit
Memory chip:
p = 512 k byte width q = 8 bit
- Number of chip rows $r \leq N/p$ = 4·512k/512k = 4
- Number of chip columns $k \geq M/q$ = 16/8 = 2
- Total of chips $K = r \times k = 4 \times 2 = 8$



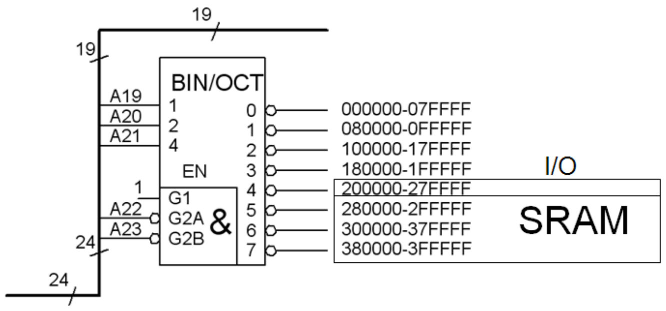
```
00ab|cmmm|mmmm|mmmm|mmmm|mmmm
0000|0000|0|0|0|0 - 0000|0111|F|F|F|F    000000-07FFFF
0000|1000|0|0|0|0 - 0000|1111|F|F|F|F    080000-0FFFFF
0001|0000|0|0|0|0 - 0001|0111|F|F|F|F    100000-17FFFF
0001|1000|0|0|0|0 - 0001|1111|F|F|F|F    180000-1FFFFF
```



Total ROM 000000 – 1FFFFF

e) Which address range is free for SRAM and IO-circuits?

```
00ab|cmmm|mmmm|mmmm|mmmm|mmmm
0010|0000|0|0|0|0 - 0010|0111|F|F|F|F    200000-27FFFF
0010|1000|0|0|0|0 - 0010|1111|F|F|F|F    280000-2FFFFF
0011|0000|0|0|0|0 - 0011|0111|F|F|F|F    300000-37FFFF
0011|1000|0|0|0|0 - 0011|1111|F|F|F|F    380000-3FFFFF
```



Possible SRAM+I/O addresses 200000 – 3FFFFF

12.3

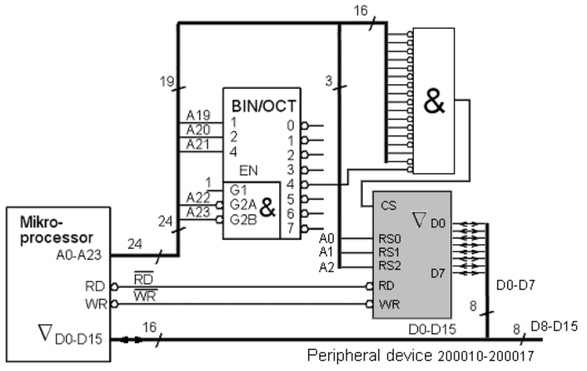
a) Connect a 8 register memory-mapped peripheral device (I/O) to a CPU. The CPU has 16-bit data bus (only 8 bits are used by the chip), and a 24 bit address bus. Use a 3:8-decoder and if needed gates. The peripheral device must be connected so that it has register addresses `0x200010` … `0x200017`.

```
0x200010 = 0010|0.000|0000|0000|0001|0.000
0x200011 = 0010|0.000|0000|0000|0001|0.001
0x200012 = 0010|0.000|0000|0000|0001|0.010
0x200013 = 0010|0.000|0000|0000|0001|0.011
```

I/O addresses, 200000 – 27FFFF is to be found, acording to the previous exercise, at the 3:8-decoder output ”4”. It decodes A23... A19, the peripheral itself decodes A2 ... A0,

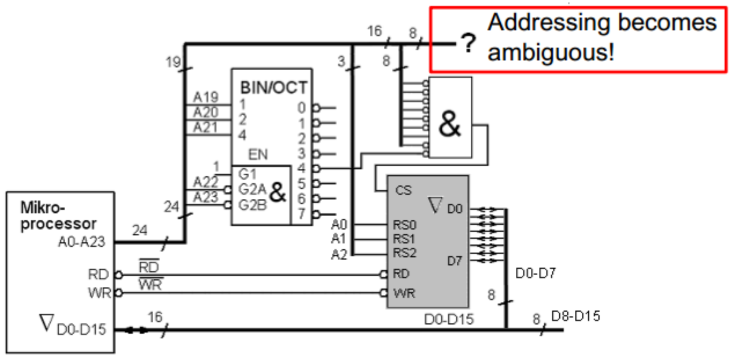
```
0x200014 = 0010|0.000|0000|0000|0001|0.100
0x200015 = 0010|0.000|0000|0000|0001|0.101
0x200016 = 0010|0.000|0000|0000|0001|0.110
0x200017 = 0010|0.000|0000|0000|0001|0.111
```

the rest we have to decode with an and-gate



b) what is meant by incomplete decoding?

For full decoding, we used a &-gate with 17 inputs! Sometimes you make a partial decoding. Then you omits address signals and thus can use a gate with fewer inputs
I/O device addressing is ambiguous, it can be addressed with many different addresses, but the one who writes the program code determines which addresses to use. The main thing is to ensure that the I/O device addresses do not collide with any other device addresses.



16
8
?
Addressing becomes ambiguous!
19
3
8
BIN/OCT
A19
A20
A21
EN
A22
A23
G1
G2A
G2B
&
&
Mikro-processor
A0-A23
24
24
RD
WR
A0
A1
A2
CS
RS0
RS1
RS2
RD
WR
D0
D7
D0-D15
D0-D7
8
8
D8-D15
16