

Logic Design – Lab 4: “Multiplexers, decoders and code converters”

For the report, please complete the tasks marked in green boxes only! (ver. 2020)

1. MULTIPLEXERS and DEMULTIPLEXERS

A multiplexer (or mux) is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of particular input line is controlled by a set of selection lines (address lines). Normally, there are 2^N input lines and N selection lines whose bit combinations determine which input is selected. An electronic multiplexer can be considered as a multiple-input, single-output switch, and is also called a data selector.

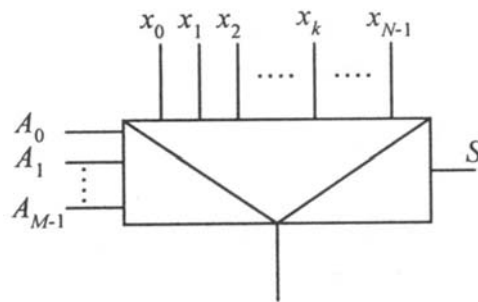


Fig. 1. Symbol of multiplexer.

where: x_0, x_1, \dots, x_{N-1} – input lines,
 A_0, A_1, \dots, A_{M-1} – selection lines,
 S – strobe input (enable input),
 Y – output.

Conversely, a demultiplexer (or demux) is a device taking a single input signal and selecting one of many data-output-lines, which is connected to the single input. An electronic demultiplexer can be considered as a single-input, multiple-output switch. A multiplexer is often used with a complementary demultiplexer on the receiving end.

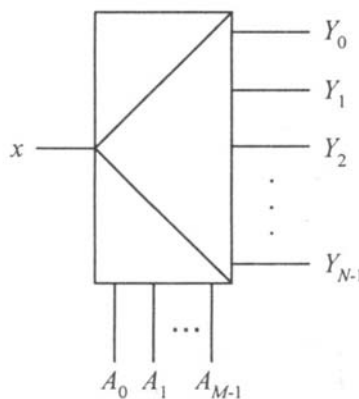


Fig. 2. Symbol of demultiplexer.

where: x – input,
 A_0, A_1, \dots, A_{M-1} – selection lines,
 Y_0, Y_1, \dots, Y_{N-1} – output lines.

LAB TASKS:

1. Construct the logic diagram of a 4 x 1 Mux (4-inputs, 1-bit):

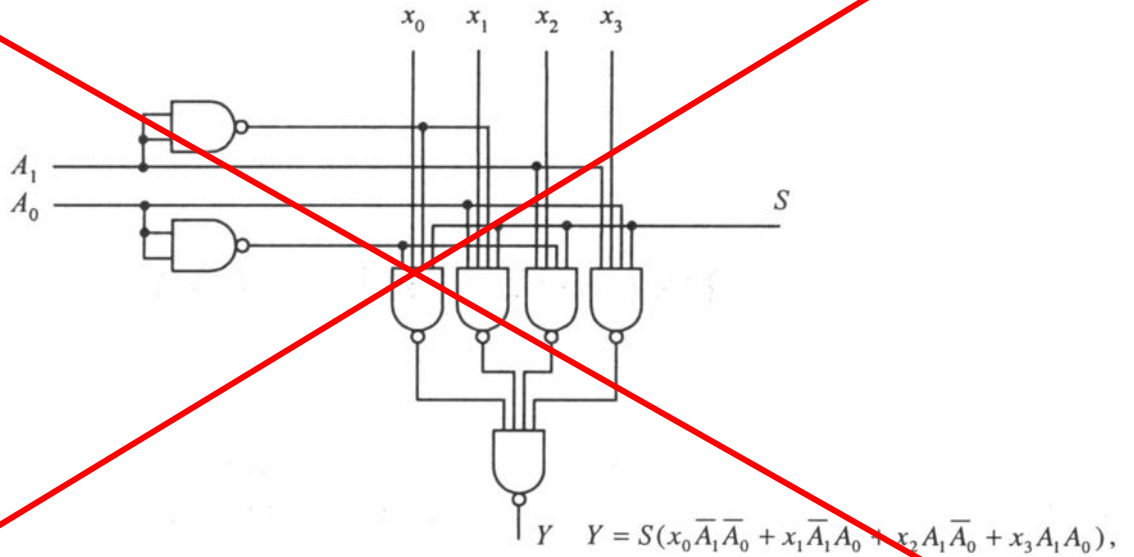


Fig. 3. Diagram of the 4-inputs, 1-bit multiplexer ant its logic output equation.

2. Construct the logic diagram of a 1 x 4 Demux (1-bit, 4-outputs):

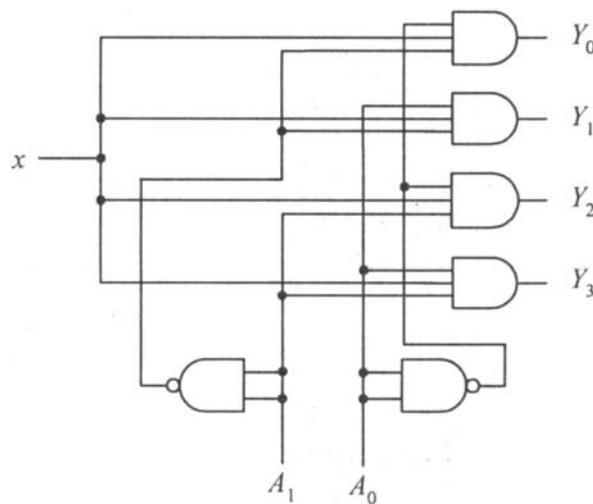


Fig. 4. Diagram of the 1-bit, 4-outputs demultiplexer.

Simulate both of logic diagrams as above using Electronic Workbench software to verify that designs work correctly according to specifications.

2. Combinational Circuits Implementation using Multiplexers

Multiplexers can also be used to implement Boolean functions of multiple variables by the way of the multiplexer selection inputs. The individual minterms can be selected by the data inputs, thereby providing a method of implementing a Boolean function of n variables with a multiplexer that has n selection inputs and 2^n data inputs, one for each minterms.

The following shows an efficient method for implementing a Boolean function of n variables with a multiplexer that has $n-1$ selection inputs. The first $n-1$ variables of the function are connected to the selection inputs of the multiplexer. The remaining single variable of the function is used for the data inputs. If a single variable is denoted by z , each data input of the multiplexer will be z , z' , 1 , or 0 . To demonstrate this procedure, consider the Boolean function:

$$F(x,y,z) = \Sigma(1,2,6,7).$$

This function of three variables can be implemented with a four-to-one-line multiplexer as shown in Figure A3.1. The two variables X and Y are applied to the selection lines in that order; X is connected to the S_1 input and y to the S_0 input. The values for the data input lines are determined from the truth table of the function. When $XY = 00$, output F is equal to z because $F = 0$ when $Z = 0$ and $F = 1$ when $Z = 1$. This requires that variable z be applied to data input 0. The operation of the multiplexer is such that when $XY = 00$, data input 0 has a path to the output, and that makes F equal to Z .

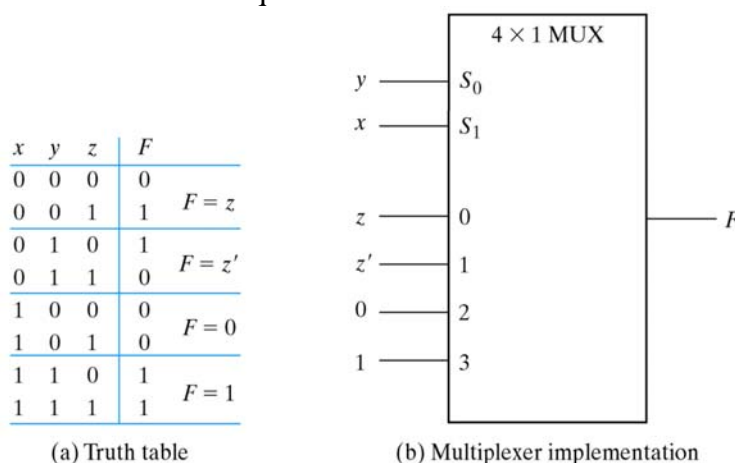


Fig. 5. Implementation of a Boolean function using a Multiplexer.

In a similar fashion, we can determine the required input to data lines 1, 2, and 3 from the value of F when $xy=01$, 10 , and 11 , respectively. This example shows all four possibilities that can be obtained for the data inputs.

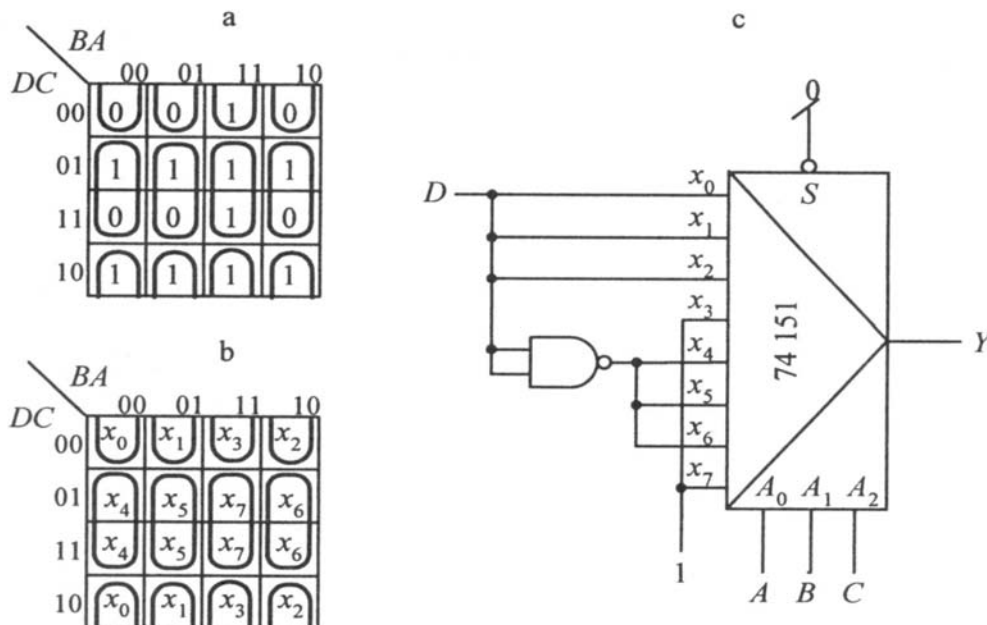
The general procedure for implementing any Boolean function of n variables with a multiplexer with $n-1$ selection inputs and 2^{n-1} data inputs follows from the example above. To begin with, Boolean function is listed in a truth table. Then first $n-1$ variables in the table are applied to the selection inputs of the multiplexer. For each combination of the selection variables, we evaluate the output as a function of the last variable. This function can be 0, 1, the variables, or the complement of the variable. These values are then applied to the data inputs in the proper order.

LAB TASK:

3. Use 8-to-1 multiplexer to design a circuit for the following function of 4 inputs:

$$Y = AB + \bar{C}D + C\bar{D}$$

Solution:



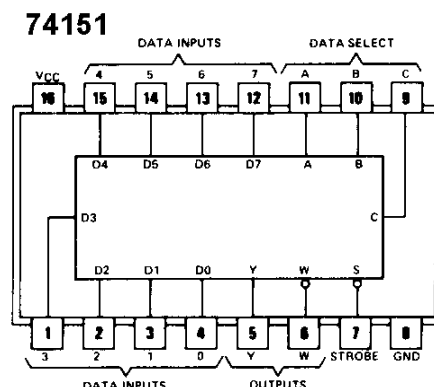
Multiplexer equation:

$$Y = x_0 \bar{C} \bar{B} \bar{A} + x_1 \bar{C} \bar{B} A + x_2 \bar{C} B \bar{A} + x_3 \bar{C} B A + x_4 C \bar{B} \bar{A} + x_5 C \bar{B} A + x_6 C B \bar{A} + x_7 C B A$$

Fig. 6. Combinational Circuits Implementation using Multiplexers. a) Karnaugh map of given boolean function, b) Karnaugh map of multiplexer equation, c) implementation with use of 74151 TTL circuit.

Simulate logic diagram as above using Electronic Workbench software to verify that design works correctly according to specifications.

Tip: see example file: “example_lab4 - 74151 multiplexer.ewb”



3. BCD and Gray codes conversion

Decimal	Binary-Coded Decimal (BCD) / 8421 code	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0100
5	0101	0110
6	0110	0111
7	0111	0101
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

LAB TASK:

4. Construct the BCD/8421 to Gray code converter which implements the following equations:

$$W(a, b, c, d) = a,$$

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

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

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

where: a, b, c, d – inputs (data in BCD code),
 W, X, Y, Z – outputs (in Gray code).

BCD to Gray code converter Karnaugh maps:

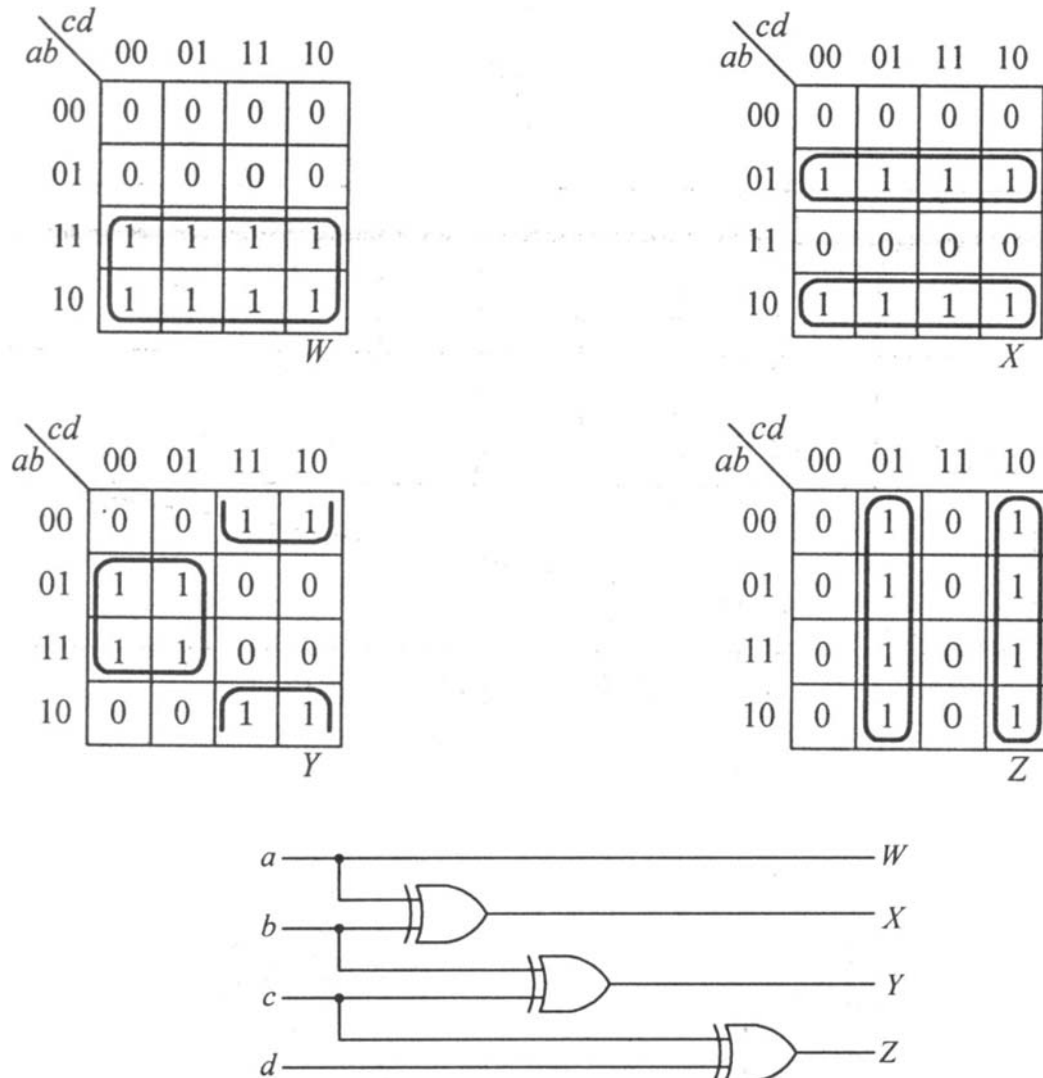


Fig. 7. XOR implementation of BCD to Gray code converter.

Simulate logic diagram as above using Electronic Workbench software to verify that design works correctly according to specifications (check all possible inputs states).

LAB TASK:

5. Construct the Gray code to the BCD/8421 code converter which implements the following equations:

$$A(w, x, y, z) = w,$$

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

$$C(w, x, y, z) = \bar{w}\bar{x}y + \bar{w}x\bar{y} + wxy + w\bar{x}\bar{y},$$

$$D(w, x, y, z) = \bar{w}\bar{x}yz + \bar{w}\bar{x}y\bar{z} + \bar{w}x\bar{y}z + wxy\bar{z} + wxy\bar{z} + w\bar{x}yz + w\bar{x}yz.$$

where: w, x, y, z – inputs (Gray code)
 A, B, C, D – outputs (BCD/8421 code),

Gray code to BCD/8421 converter Karnaugh maps:

wz	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

A

wz	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

B

wz	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

C

wz	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

D

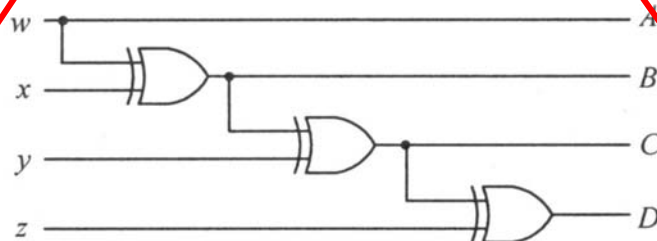


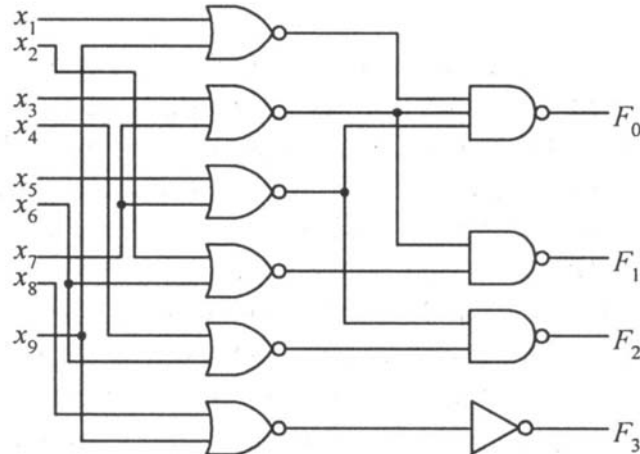
Fig. 8. Implementation of Gray to BCD code converter with XOR gates.

Simulate logic diagram as above using Electronic Workbench software to verify that design works correctly according to specifications (check all possible inputs states).

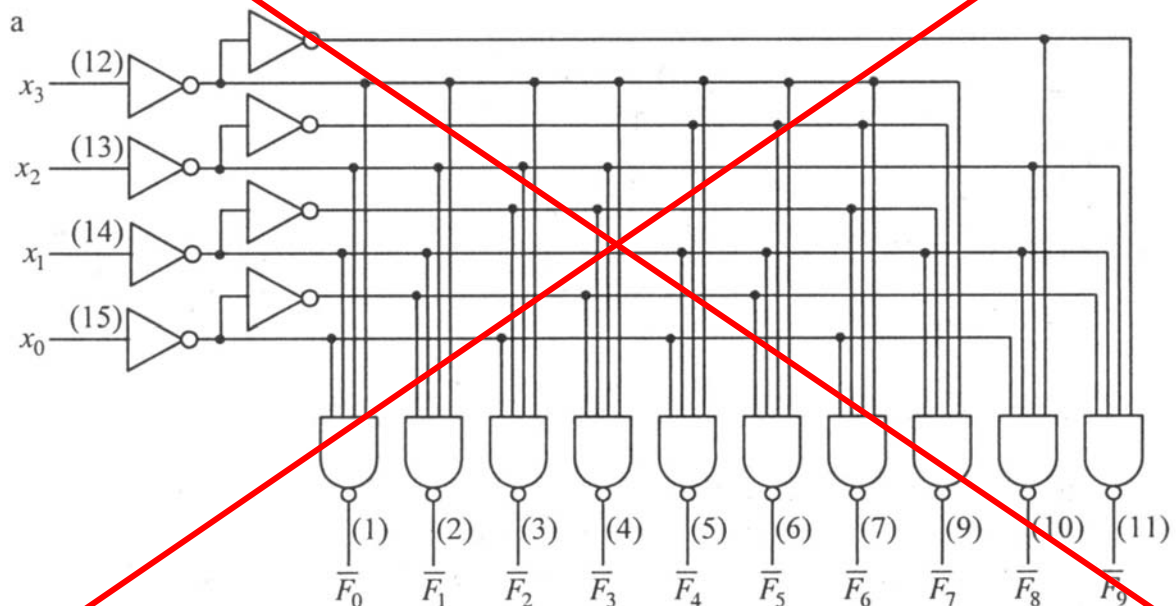
4. ENCODERS/DECODERS

LAB TASK:

6. Construct the **Decimal ("1 of 10") to BCD encoder** and simulate it using Electronic Workbench software to verify that it operates correctly according to truth table (check all possible input conditions):

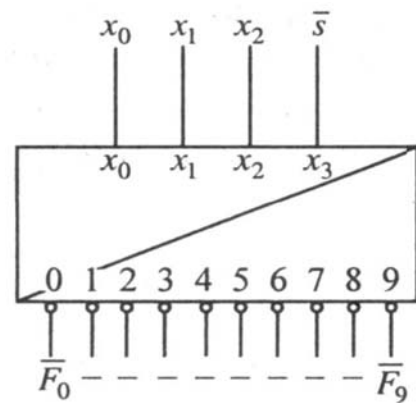


7. Construct the **BCD to Decimal decoder ("1 of 10")** and simulate it using EWB software to verify that it operates correctly according to truth table (check all possible input conditions). It may be used circuit build with simple gates only as shown on diagram as below, or with use of 7442 TTL:



BCD to Decimal decoder (“1 of 10”) truth table:

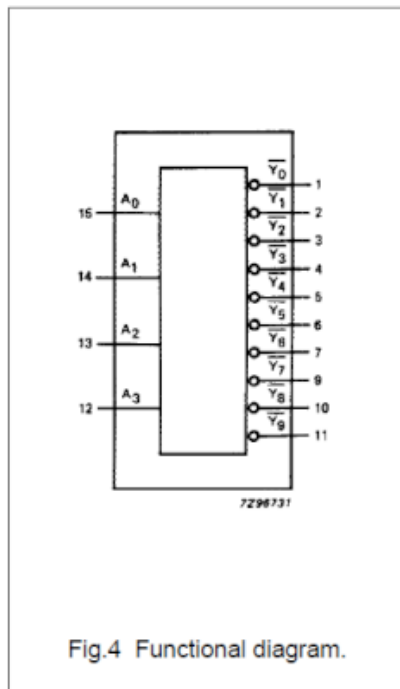
$x_3x_2x_1x_0$	$\overline{F_0}\overline{F_1}\overline{F_2}\overline{F_3}\overline{F_4}\overline{F_5}\overline{F_6}\overline{F_7}\overline{F_8}\overline{F_9}$
0000	0 1 1 1 1 1 1 1 1 1
0001	1 0 1 1 1 1 1 1 1 1
0010	1 1 0 1 1 1 1 1 1 1
0011	1 1 1 0 1 1 1 1 1 1
0100	1 1 1 1 0 1 1 1 1 1
0101	1 1 1 1 1 0 1 1 1 1
0110	1 1 1 1 1 1 0 1 1 1
0111	1 1 1 1 1 1 1 0 1 1
1000	1 1 1 1 1 1 1 1 0 1
1001	1 1 1 1 1 1 1 1 1 0
1010	1 1 1 1 1 1 1 1 1 1
1011	1 1 1 1 1 1 1 1 1 1
1100	1 1 1 1 1 1 1 1 1 1
1101	1 1 1 1 1 1 1 1 1 1
1110	1 1 1 1 1 1 1 1 1 1
1111	1 1 1 1 1 1 1 1 1 1



BCD to decimal decoder (1-of-10)

7442 (TTL)

74HC/HCT42



FUNCTION TABLE

INPUTS				OUTPUTS									
A_3	A_2	A_1	A_0	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$	$\overline{Y_4}$	$\overline{Y_5}$	$\overline{Y_6}$	$\overline{Y_7}$	$\overline{Y_8}$	$\overline{Y_9}$
L	L	L	L	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	H	H	L	H	H	H	H	H	H	H
L	L	H	H	H	H	H	L	H	H	H	H	H	H
L	H	L	L	H	H	H	H	L	H	H	H	H	H
L	H	L	H	H	H	H	H	H	L	H	H	H	H
L	H	H	L	H	H	H	H	H	H	L	H	H	H
L	H	H	H	H	H	H	H	H	H	H	L	H	H
H	L	L	L	H	H	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L
H	L	H	L	H	H	H	H	H	H	H	H	H	H
H	L	H	H	H	H	H	H	H	H	H	H	H	H
H	H	L	L	H	H	H	H	H	H	H	H	H	H
H	H	L	H	H	H	H	H	H	H	H	H	H	H
H	H	H	L	H	H	H	H	H	H	H	H	H	H
H	H	H	H	H	H	H	H	H	H	H	H	H	H

Note

1. H = HIGH voltage level
L = LOW voltage level