Combinational Logic

CS211 Chapter 5

James YU
yujq3@sustech.edu.cn

Department of Computer Science and Engineering
Southern University of Science and Technology

Jul. 12, 2022

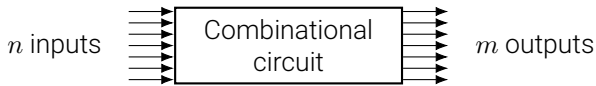
南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Combinational logic

- Logic circuits for digital systems:
  - combinational logic,
  - sequential logic (next lectures).
- *Combinational*?
  - Output determined by the combination of inputs.
  - Perform an operation specified by a set (combination) of Boolean functions.

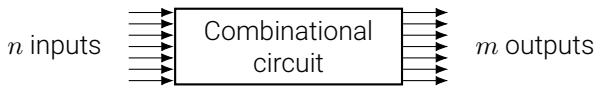
Combinational logic

- An interconnection of logic gates that
  - react to values of input signals,
  - produce output signal values,
  - n inputs from external sources, and
  - m outputs go to external destinations.



Combinational logic

- Two representations of a combinational circuit:
  - 2^n possible input combinations: truth table, or
  - m outputs: m Boolean functions, each expressed with the n inputs.



Analysis of combinational logic

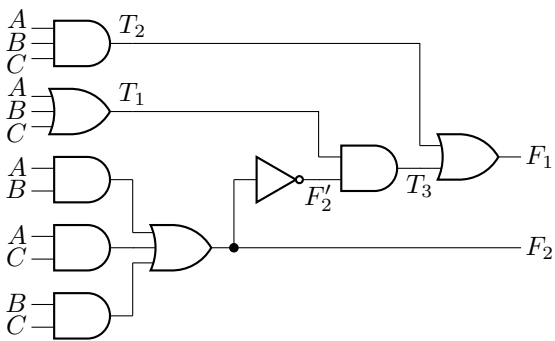
- Analysis of a combinational circuit: determine the function of the circuit.
  - Given logic diagram,
  - develop a set of Boolean functions, a truth table, an optional explanation of the circuit operation.
- If a function name or an explanation is given along the circuit, just verify if the given information is correct.

Analysis of combinational logic

- Obtain the output Boolean functions:
  - Label all gate outputs that are a function of only inputs, no other intermediate variables. Determine their Boolean functions.
  - Label all gates that are a function of inputs and the gates in the previous step. Determine their Boolean functions.
  - List output Boolean functions, squeeze the intermediate variables.

Analysis of combinational logic

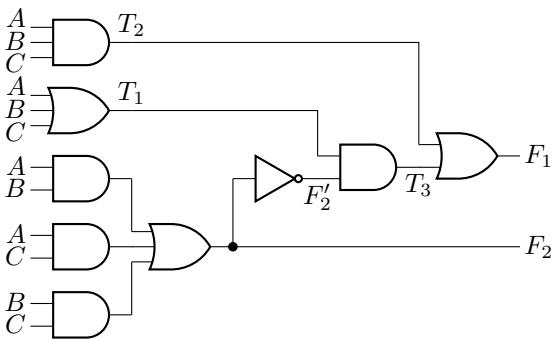
- Obtain the output Boolean functions:
  - Label all gate outputs that are a function of only inputs, no other intermediate variables. Determine their Boolean functions.



- $T_1 = A + B + C.$
- $T_2 = ABC.$
- $F_2 = AB + AC + BC.$

Analysis of combinational logic

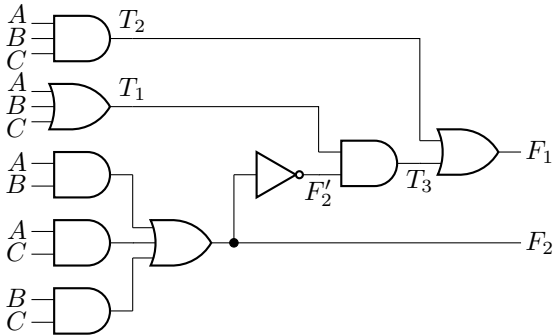
- Obtain the output Boolean functions:
  - Label all gates that are a function of inputs and the gates in the previous step. Determine their Boolean functions.



- $T_3 = F_2' + T_1$.
- $F_1 = T_2 + T_3$.

Analysis of combinational logic

- Obtain the output Boolean functions:
  - List output Boolean functions, squeeze the intermediate variables.



$$
\begin{aligned}
F_1 &= T_2 + T_3 \\
&= ABC + F_2' T_1 \\
&= ABC + \\
&\quad (AB + AC + BC)' \\
&\quad (A + B + C) \\
&= ABC + A'B'C + \\
&\quad AB'C' + A'BC'
\end{aligned}
$$

Analysis of combinational logic

- Truth table is simple with Boolean function
  - Determine the number of input variables. For n inputs, form the 2^n combinations from 0 to 2^n-1.
  - Label the outputs of the intermediate gates.
  - Obtain the truth table for these outputs.
  - Obtain the truth table for the remaining outputs.

A	B	C	F_2	F_2'	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1

Design of combinational logic

- Design of a combinational circuits: develop a logic circuit diagram or a set of Boolean functions.
  - From specification of the design objective
- Involves the following steps:
  - Determine required number of inputs and outputs.
  - Derive the truth table.
  - Obtain simplified Boolean function for each output.
  - Draw logic diagram and verify the correctness.

Design of combinational logic

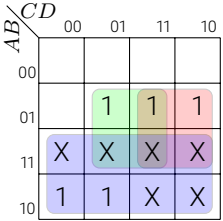
- Example: Convert from BCD decimal code to excess-3 code.

Input BCD				Output Code			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

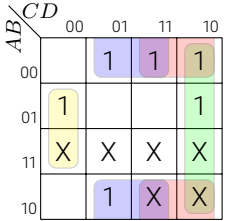
- Obviously, four inputs and four outputs
- And we already have the truth table
- Then the Boolean functions. How?
  - Remember K-maps?

Design of combinational logic

$$w = A + BC + BD$$

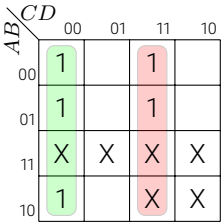


$$x = B'C + B'D + BC'D' + CD'$$

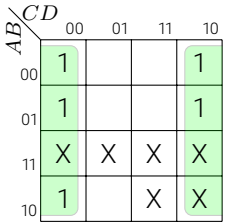


Design of combinational logic

$$y = CD + C'D'$$



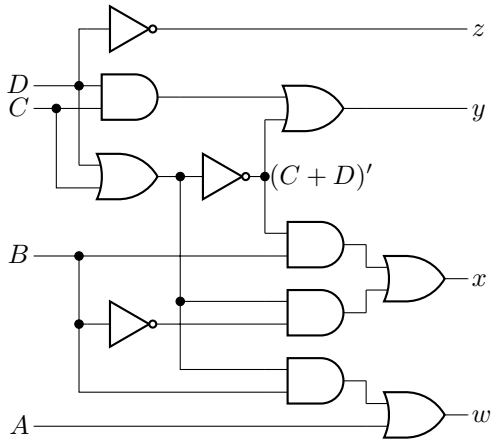
$$z = D'$$



Design of combinational logic

- We manipulate these Boolean functions to reuse common gates:
  - $w = A + BC + BD = A + B(C + D)$.
  - $x = B'C + B'D + BC'D' = B'(C + D) + BC'D'$.
  - $y = CD + C'D' = CD + (C + D)'$.
  - $z = D'$.

Design of combinational logic

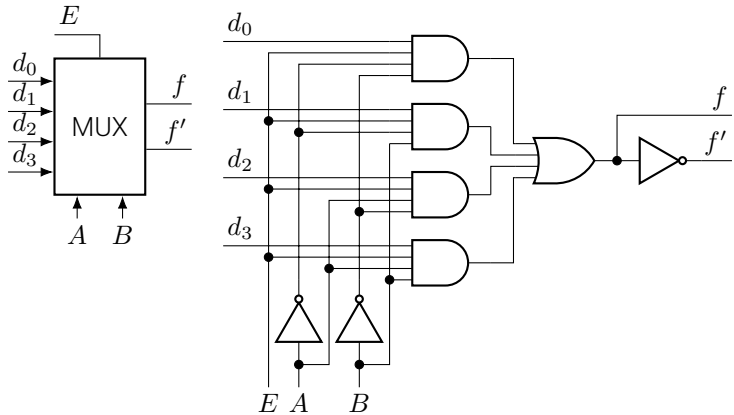


Combinational logic design with MSI circuits

- Since the introduction of MSI and LSI circuits, the traditional methods of logic design have largely been superseded.
  - Traditionally, the design engineer has developed a Boolean equation as the solution to a particular problem.
  - This function has then been minimised and implemented using SSI circuits.
- In practice, many combinational circuits may have a large number of inputs and outputs.
  - Consequently the use of truth tables in the design of such circuits is impractical.
- The development of MSI circuits has led to the technique of splitting a complex design into a number of sub-systems.

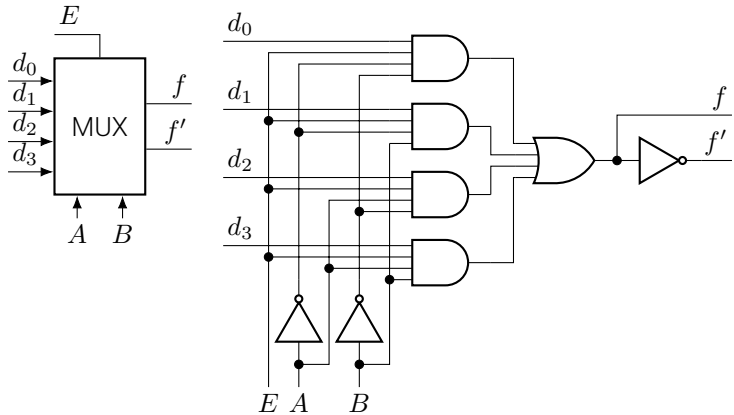
Multiplexer

- A multiplexer (MUX) selects 1-out-of-n lines where n is usually 2, 4, 8, or 16.
- A block diagram of a multiplexer having four input data lines `d0`, `d1`, `d2`, and `d3` and complementary outputs `f` and `f'`.



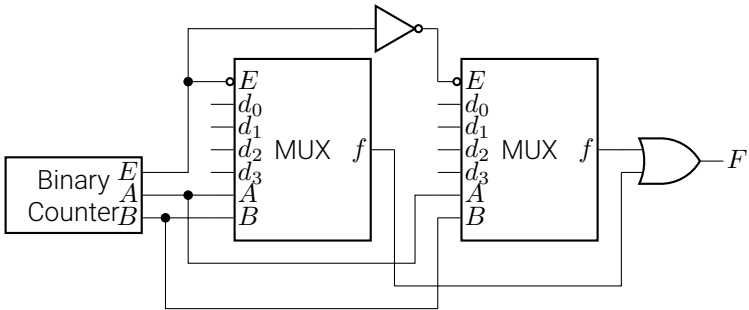
Multiplexer

- The device has two control or selection lines A and B and an enable line E.
- The characteristic equation of the multiplexer is
  $f = A'B'd_0 + A'Bd_1 + AB'd_2 + ABd_3$.



Interconnecting multiplexers

- Data within a digital system is normally processed in parallel form in order to increase the speed of operation.
- If the output of the system has to be transmitted over a relatively long distance then a parallel-to-serial conversion will take place.
- Example: An 8-bit word is presented in parallel at the data inputs.

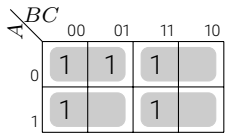


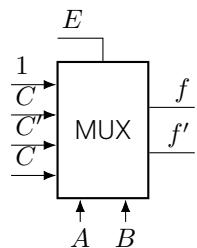
Interconnecting multiplexers

- The principle of data selection can be extended to allow the selection of 1-out-of-64 lines.
  - Using nine 8-to-1 multiplexers arranged in two levels of multiplexing.
  - **How?**

Multiplexer as a Boolean function generator

- For a 4-to-1 MUX the characteristic equation is $f = A'B'd_0 + A'Bd_1 + AB'd_2 + ABd_3$.
  - A and B are Boolean variables, applied at the select inputs, which can be factored out of any Boolean function of n variables.
  - The remaining $n - 2$ variables, referred to as the *residue variables*, can be formed into residue functions which can then be applied at the data inputs.
- Example: $f(A, B, C) = \sum(0, 1, 3, 4, 7)$.

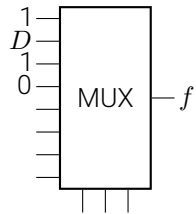




Multiplexer as a Boolean function generator

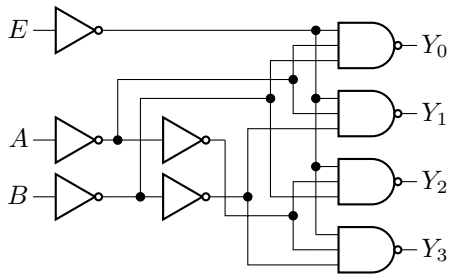
- Example: $f(A, B, C, D) = \sum(0, 1, 3, 4, 5, 9, 10, 11, 14, 15)$.

A	B	C	D	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
		...		



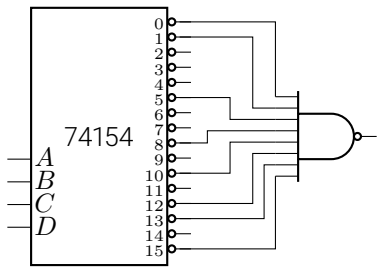
Decoder

- The basic function of an MSI decoder having n inputs is to select 1-out-of-2^n output lines.
  - The selected output is identified either by a 1, when all other outputs are 0, or by a 0 when all other outputs are 1.



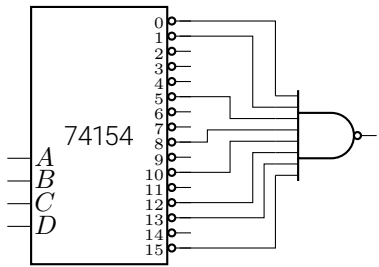
Decoder

- It will be seen that the logic diagram of the basic decoder is identical to that of the basic demultiplexer.
  - Provided the data line is used to enable the decoder.
- The decoder may also be regarded as a minterm generator. Each output generates one minterm.



Decoder

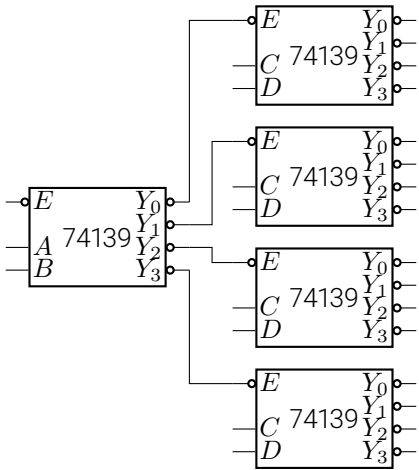
- If $A = B = C = D = 0$ the output 0 of the decoder is active low while all other outputs are 1.
  - The decoder can generate the inverse of the 16 minterms.
  - Example: $f(A, B, C, D) = \sum(0, 1, 5, 8, 10, 12, 13, 15)$.



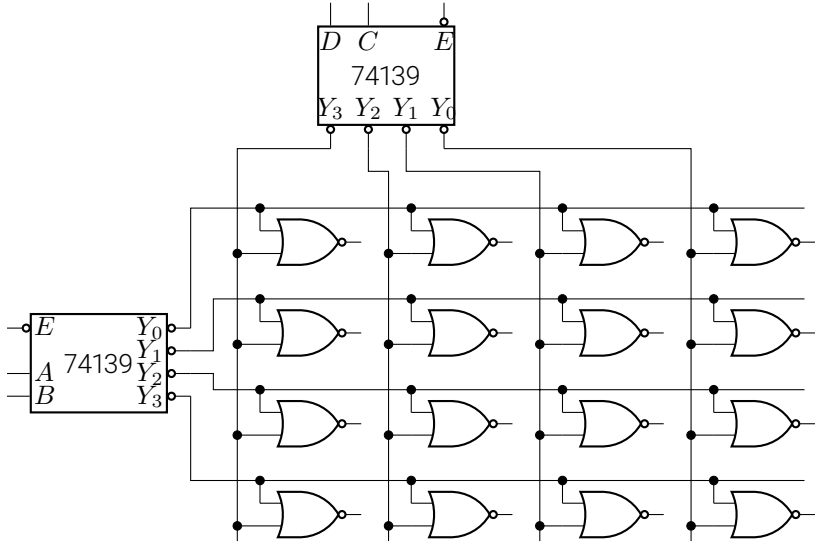
Decoder network

- When a large decoding network is required it cannot be implemented in a single MSI package.
  - Mainly because of the large number of pins needed.
- The decoding range can be extended by interconnecting decoder chips. Two schemes:
  - *Tree decoding*.
  - *Coincident decoding*.

Decoder network

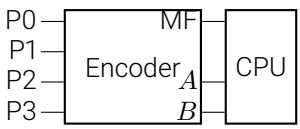


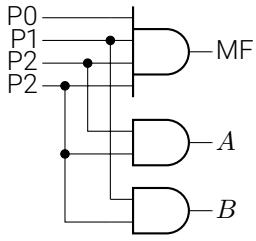
Decoder network



Encoder

- An encoder performs the inverse operation to that of a decoder.
- Example: CPU master flag





- The encoder is designed to identify one, and only one, of the peripherals at any given instant.
  - In practice, there is nothing to prevent two or more peripherals requesting service at the same time.
  - To deal with this situation a system of priorities can be attached to the peripheral flags.

Encoder

- Example: Octal-to-binary encoder.

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

- Eight inputs and three outputs connected with OR.

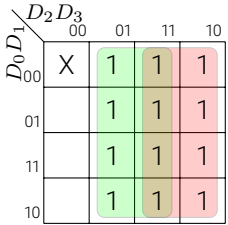
Encoder

- $x = D_4 + D_5 + D_6 + D_7$.
- $y = D_2 + D_3 + D_6 + D_7$.
- $z = D_1 + D_3 + D_5 + D_7$.
- Only one input can be active for one time.
  - D_3 and D_6 are 1 simultaneously: outputs $(111)_2 = 7$.
  - Encoder circuit must have priority: *Priority encoder*.

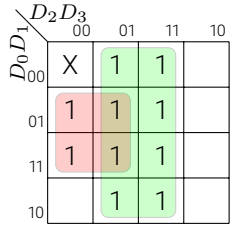
Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	x	x	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Encoder

$$x = D_2 + D_3$$

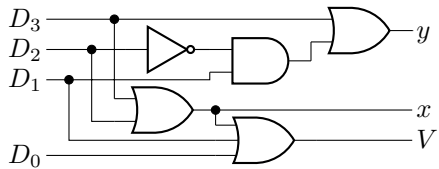


$$y = D_3 + D_1D_2'$$



Encoder

- $x = D_2 + D_3$.
- $y = D_3 + D_1 D_2'$.
- $V = D_0 + D_1 + D_2 + D_3$.

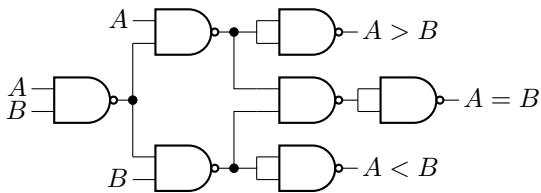


Digital comparator

- The usual problem for a comparator is the comparison of two multi-digit words such as $A = A_2A_1A_0$ and $B = B_2B_1B_0$.
  - Start from most to least significant bit.
  - $A = B$ if all bits are equal: $A_i = B_i$.
  - $x_i = A_iB_i + A_i'B_i'$.
- $A = B$ if $x_2x_1x_0 = 1$.
- $A > B$ if $A_2B_2' + x_2A_1B_1' + x_2x_1A_0B_0' = 1$.
- $A < B$ if $A_2'B_2 + x_2A_1'B_1 + x_2x_1A_0'B_0 = 1$.

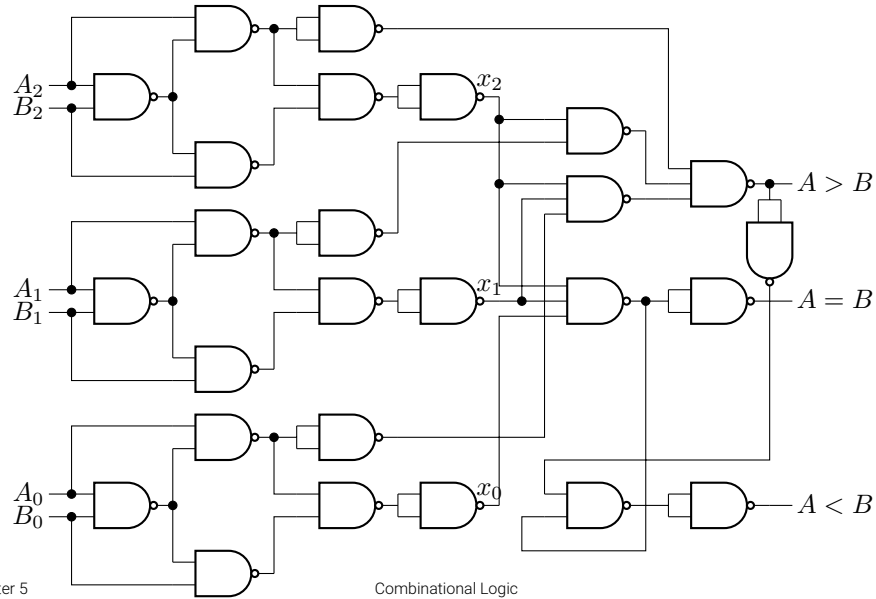
Digital comparator

- The implementation of a 3-bit comparator is based on a single bit comparator.



- Using the equations developed in the last page, we can have a 3-bit comparator.

Digital comparator



Exercise

- Design a combinational circuit with three inputs and one output.
  - The output is 1 when the binary value of the inputs is less than 3. The output is 0 otherwise.
  - The output is 1 when the binary value of the inputs is an even number.

Exercise

- An 8×1 multiplexer has inputs A, B, and C connected to the selection inputs S_2, S_1, and S_0, respectively. The data inputs I_0 through I_7 are as follows: $I1 = I2 = 0$; $I3 = I7 = 1$; $I4 = I5 = D$; and $I0 = I6 = D'$. Determine the Boolean function that the multiplexer implements.

Exercise

- A combinational circuit is defined by the equations
  $F_1(A, B, C) = AB + A'B'C'$
  $F_2(A, B, C) = A + B + C$
  $F_3(A, B, C) = A'B + AB'$
  Design a circuit which will implement these three equations using a decoder and NAND gates external to the decoder.