Lecture 5: Combinational Logic - Part I

CS207: Digital Logic

Jialin Liu

Department of Computer Science and Engineering (CSE)
Southern University of Science and Technology (SUSTech)

9 October 2022

Acknowledgement

These slides were prepared based on the slides by Dr. Jianqiao Yu and the ones by Prof. Georgios Theodoropoulos of the Department of CSE at the SUSTech, as well as the contents of the following book:
M. M. Mano and M. Ciletti, *Digital design: with an introduction to the Verilog HDL*. Pearson, 2013

Recap: Gate-level Minimisation II

- Universal gates: NAND and NOR gates.
- Procedure of two-level and multi-level implementations using NAND or NOR gates.
- Exclusive-OR gate for constructing error detection circuits.

- Lecture 1: Binary Numbers
- Lecture 2: Boolean Algebra and Logic Gates
- Lecture 3 Gate-Level Minimisation
- Lecture 4: Gate-Level Implementation

→ Now we will use the knowledge acquired in previous weeks to formulate systematic analysis and design procedures for combinational circuits.

Types of Logic Circuits

- "A **combinational circuit** consists of logic gates whose outputs *at any time* are determined from only the present combination of inputs."
  → Combinational logic (Week 5 and Week 6).

- "**Sequential circuits** employ storage elements in addition to logic gates. Their outputs are a function of the inputs and the state of the storage elements."
  - State of the storage elements: a function of previous inputs.
  - Outputs of a sequential circuit depend on: values of present inputs and past inputs.
  - Its behaviour must be specified by a time sequence of inputs and internal states.

  → Sequential logic (Week 7 and Week 8).

Lecture 5: Combinational Circuit

Outline of This Lecture

Combinational Circuit

- A combinational circuit consists of an interconnection of logic gates.
  → No feedback paths or memory elements.
- Combinational logic gates
  - react to values of input signals,
  - produce output signal values,
  - transform binary information from the given input data to a required output data.
- Remark: Each input and output variable exists physically as an analog signal whose values are interpreted to be a binary signal.

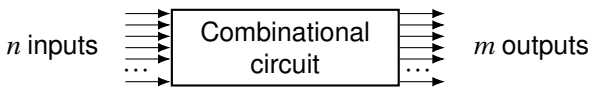
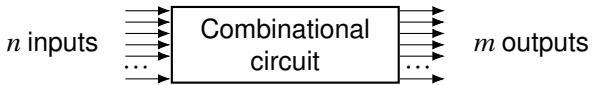


Figure: A block diagram of a combinational circuit. n input binary variables come from external sources, and m output variables (produced by the circuit) go to external destinations.

Representations of Combinational Circuits

- Two representations of a combinational circuit:
  1. 2^n possible input combinations: truth table
  2. m outputs: m Boolean functions, each expressed with the n inputs.



What We Will Learn about Combinational Circuits

- Formulate systematic analysis and design procedures for combinational circuits:
  1. Analyse the behaviour of a given logic circuit.
  2. Synthesise a circuit that will have a given behaviour.
  3. Write hardware description language (HDL) models for some common circuits.

- Standard components: combinational circuits that are employed extensively in the design of digital systems.
  - Adders, subtractors, comparators, decoders, encoders, and multiplexers.
  - Available in integrated circuits as medium-scale integration (MSI) circuits.
  - Also used as standard cells in complex very large-scale integrated (VLSI) circuits such as application-specific integrated circuits (ASICs).

Outline of This Lecture

Analysis of Combinational Circuits

- **Analysis of a combinational circuit**: determine the function of the circuit.
  - Input of the procedure: a logic diagram.
  - Output of the procedure:
    - a set of Boolean functions,
    - a truth table,
    - or, possibly, an 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.
- The analysis can be performed manually or by using a computer simulation program.

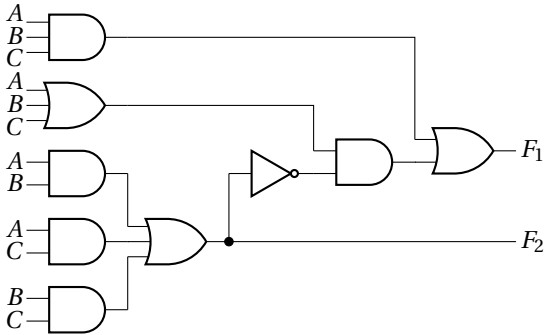
Outline

Analysis of Combinational Circuits: Steps

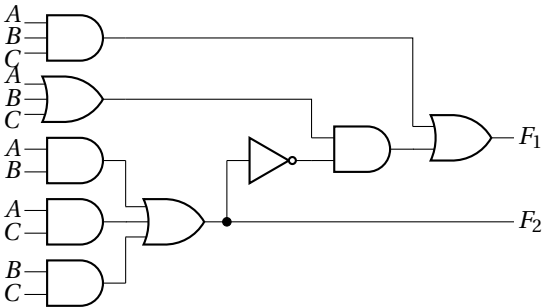
- **Input**: a logic diagram.
  1. **Make sure that the given circuit is combinational and not sequential.**
     - The diagram of a combinational circuit has logic gates with no feedback paths or memory elements.
     - **Feedback path**: a connection from the output of one gate to the input of a second gate whose output forms part of the input to the first gate.
  2. **Obtain the output Boolean functions from the given diagram:**
     - 2.1 Label all gate outputs that are a function of input variables with arbitrary symbols – but with meaningful names (i.e., only inputs, no other intermediate variables). Determine the Boolean functions for each gate output.
     - 2.2 Label the gates that are a function of input variables and previously labeled gates with other arbitrary symbols. Determine their Boolean functions.
     - 2.3 Repeat the process outlined in step 2.2 until the outputs of the circuit are obtained.
     - 2.4 By repeated substitution of previously defined functions, obtain the output Boolean functions in terms of input variables.
- **Output**: a set of Boolean functions.

Analysis of Combinational Circuits: Example

- Input: a logic diagram.
- Output: a set of Boolean functions.

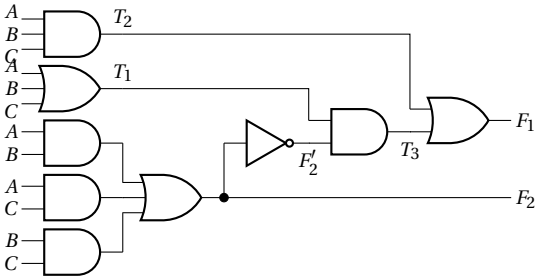


1. Make sure that the given circuit is combinational and not sequential.
   - The diagram of a combinational circuit has logic gates with no feedback paths or memory elements.
   - **Feedback path**: a connection from the output of one gate to the input of a second gate whose output forms part of the input to the first gate.



- There is no feedback path!
- It is a combinatorial circuit :)

2. Obtain the output Boolean functions from the given diagram:
   - 2.1 Label all gate outputs that are a function of input variables with arbitrary symbols (i.e., only inputs, no other intermediate variables). Determine the Boolean functions for each gate output.
   - 2.2 Label the gates that are a function of input variables and previously labeled gates with other arbitrary symbols. Determine their Boolean functions.
   - 2.3 Repeat the process outlined in step 2.2 until the outputs of the circuit are obtained.
   - 2.4 By repeated substitution of previously defined functions, obtain the output Boolean functions in terms of input variables.



- $T_1 = A + B + C.$
- $T_2 = ABC.$
- $F_2 = AB + AC + BC.$
- $T_3 = F_2' T_1.$
- $F_1 = T_2 + T_3.$

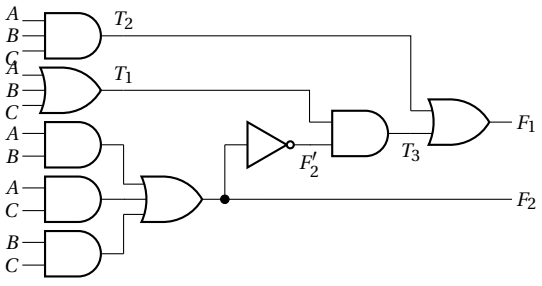
- Form a series of substitutions:

$$F_1 = T_2 + T_3 = ABC + F_2' T_1 = ABC + (AB + AC + BC)'(A + B + C) = ABC + A'BC' + A'B'C + AB'C'$$

Outline

Output A Truth Table: Steps

- Truth table is simple with Boolean function
  1. Determine the number of input variables. For n inputs, form the 2^n combinations from 0 to $2^n - 1$.
  2. Label the outputs of the intermediate gates.
  3. Obtain the truth table for these outputs.
  4. 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

Outline of This Lecture

Design of Combinational Circuits: Steps

- Design of combinational circuits: develop a logic circuit diagram or a set of Boolean functions from specification of the design objective.
  - Input: a specification of the design objective.
  - Output: a logic circuit diagram or a set of Boolean functions.

- Steps:
  - From the specifications of the circuit, determine the required number of inputs and outputs and assign a symbol to each.
  - Derive the truth table that defines the required relationship between inputs and outputs.
  - Obtain the simplified Boolean functions for each output as a function of the input variables.
  - Draw the logic diagram and verify the correctness of the design (manually or by simulation).

Example: A Conversion Circuit (1/5)

- Convert from BCD code to excess-3 code (c.f. page 23 of [1])[1].

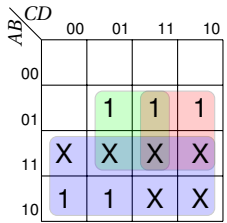
Input BCD				Output Code			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

- Each code uses 4 bits, therefore 4 inputs and 4 outputs.
- And we already have the truth table.
- Then the Boolean functions. How?
  - Remember K-maps?

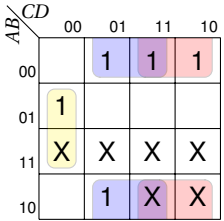
[1] Self-complementary BCD code used to represent the decimal numbers.

Example: A Conversion Circuit (2/5)

$$w = A + BC + BD$$

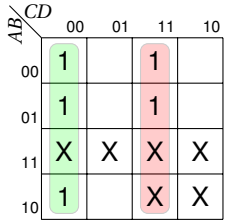


$$x = B'C + B'D + BC'D'$$

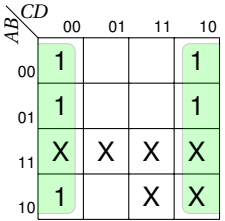


Example: A Conversion Circuit (3/5)

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



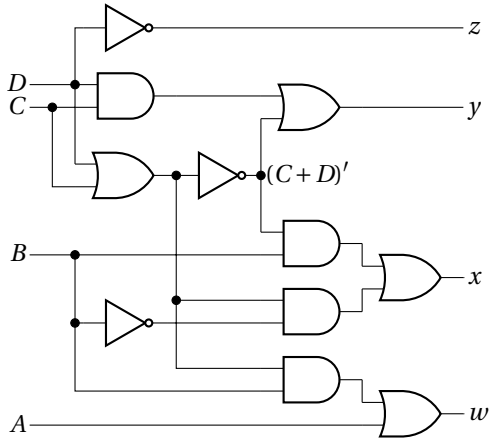
$$z = D'$$



Example: A Conversion Circuit (4/5)

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

Example: A Conversion Circuit (5/5)



Outline of This Lecture

Combinational Logic Design with MSI Circuits

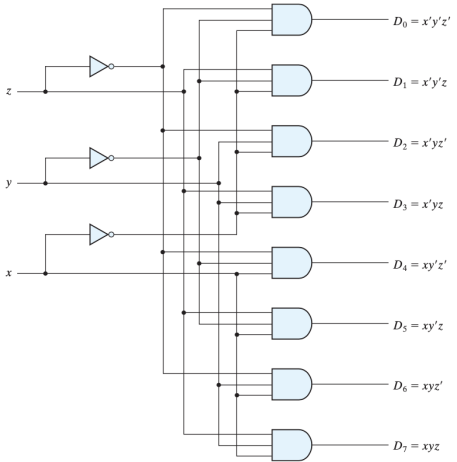
- Since the introduction of MSI and LSI circuits, the traditional methods of logic design have largely been superseded.
  - Traditionally, the design engineer has developed a Boolean equation as the solution to a particular problem.
  - This function has then been minimised and implemented using SSI 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.

Outline

Decoder (译码器)

- Digital systems
  - Discrete quantities of information are represented by binary codes.
  - A binary code of n bits → up to 2^n distinct elements of coded information.
- Decoder
  - A **decoder** is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique 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.
  - The basic function of an MSI decoder having n inputs is to select 1-out-of-2^n output lines.

Example



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

Figure: Three-to-eight-line decoder. Example application: binary-to-octal conversion.

Figure: Figure 4.18 and Table 4.16 in [1].

Related Terms

- **n-to-m-line decoder**
  - A n-to-m-line decoder is a combinational circuit that converts binary information from n input lines to m unique output lines with $m \leq 2^n$.

- "Decoder" is also used in conjunction with other code converters.
  - Example: BCD-to-seven-segment decoder.

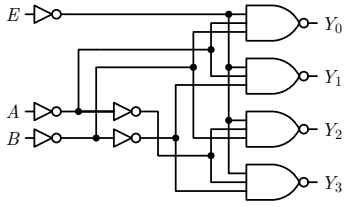
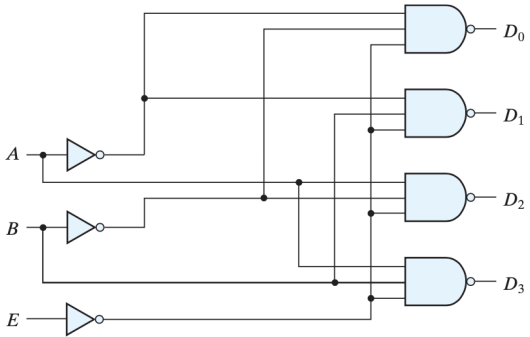


Figure: Three-to-four-line decoder.

Design Decoders

- Some decoders are constructed with NAND gates.
  - A NAND gate produces the AND operation with an inverted output.
    → More economical to generate minterms in their complemented form.
- Decoders include one or more **enable inputs** (使能端) to control the circuit operation.



E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Figure: Two-to-four-line decoder with enable input. Figure 4.19 in [1].

Main Applications of Decoders

- Minterm generator (函数最小项发生器): Generate the 2^n (or fewer) minterms of n input variables.
- Demultiplexer (数据分配器): A decoder with enable input can function as a **demultiplexer** – a circuit that receives information from a single line and directs it to one of 2^n possible output lines.
- Address decoder (地址解码器): Identify a memory cell, disk sector, or other memory or storage device, to ensure one device can communicate with the processor at one time.

Decoder as Minterm Generator

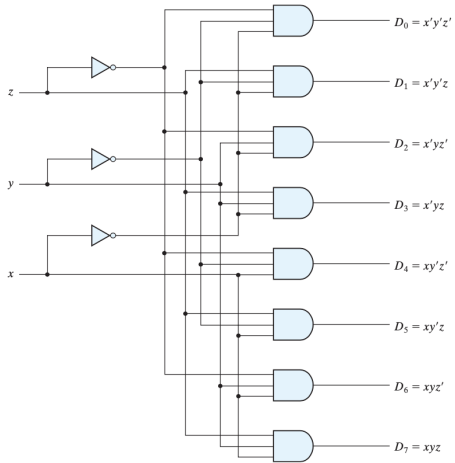
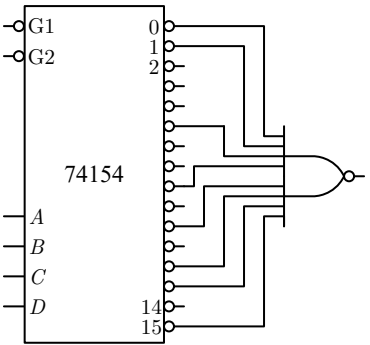


Figure: Three-to-eight-line decoder. Figure 4.18 in [1].

Decoder as Minterm Generator (函数最小项发生器)

- 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 as Demultiplexer (数据分配器)

- A decoder with enable input can function as a **demultiplexer** – a circuit that receives information from a single line and directs it to one of 2^n possible output lines.
- Because decoder and demultiplexer operations are obtained from the same circuit, a decoder with an enable input is referred to as a **decoder-demultiplexer**.

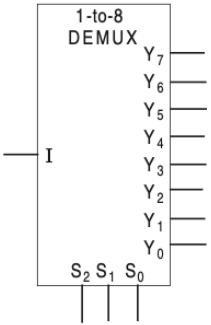


Figure: Figure 5.95 in [2].

Interconnect Decoders with Enable Inputs

- Decoders with enable inputs can be connected together to form a larger decoder circuit.
- Enable inputs are a convenient feature for interconnecting two or more standard components.

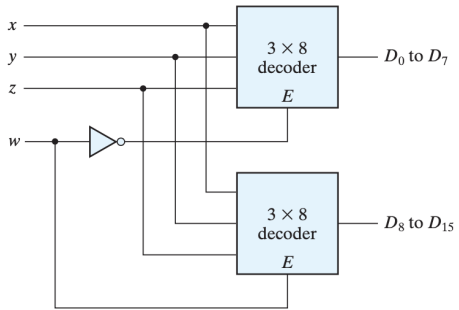


Figure: A 4×16 decoder constructed with two 3×8 decoders. Figure 4.20 [1].

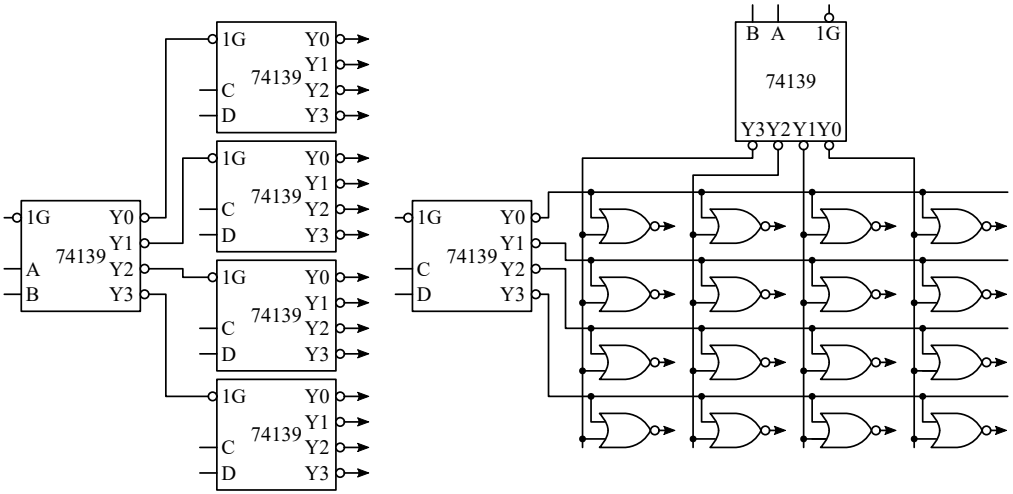
Decoder Network

- 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**.
    - Reduce the total number of gates by using two-dimensional decoding: arrange memory cells in a (as close as possible to) square configuration. Use two $n/2$ input decoders instead of one n input decoder.

Decoder Network

Example



Outline

Encoder (编码器)

- An **encoder** performs the inverse operation to that of a decoder.
- Example: Octal-to-binary encoder.
  - Eight inputs and three outputs connected with OR.
  - Only one input can be active for one time.

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

Encoder

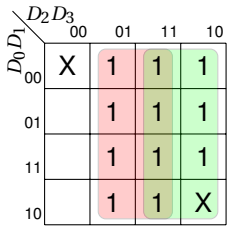
- Only one input can be active for one time.
- D_3 and D_6 are 1 simultaneously: outputs $(111)_2 = 7$. ← Neither 3 or 6 !
  - $x = D_4 + D_5 + D_6 + D_7$.
  - $y = D_2 + D_3 + D_6 + D_7$.
  - $z = D_1 + D_3 + D_5 + D_7$.
- Encoder circuit must have priority: **Priority encoder**.

Priority Encoder (优先编码器)

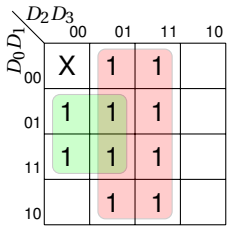
- Encoder circuits must establish an input priority to ensure that only one input is encoded.
- Example:

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

$$x = D_2 + D_3$$



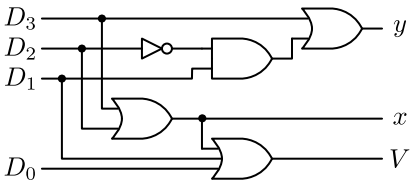
$$y = D_3 + D_1 D_2'$$



Priority Encoder

Example

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



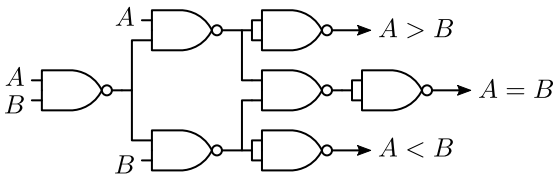
Outline

Magnitude Comparator

- **Magnitude comparator** compares two binary numbers and determines if one number is greater than, less than, or equal to the other number.
- 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 $A_2'B_2 + x_2A_1'B_1 + x_2x_1A_0'B_0 = 1$.

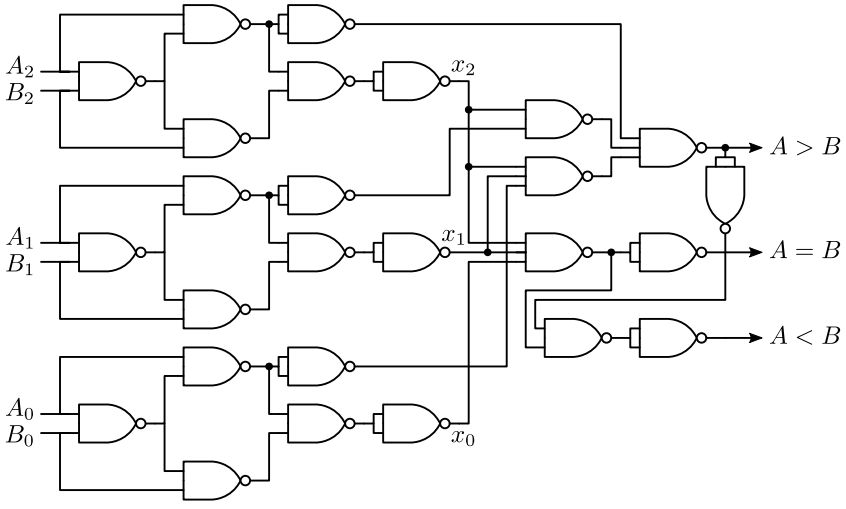
Magnitude Comparator

- 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.

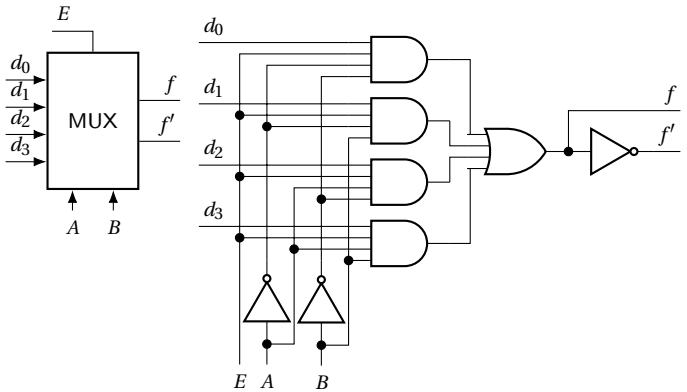
Magnitude Comparator



Outline

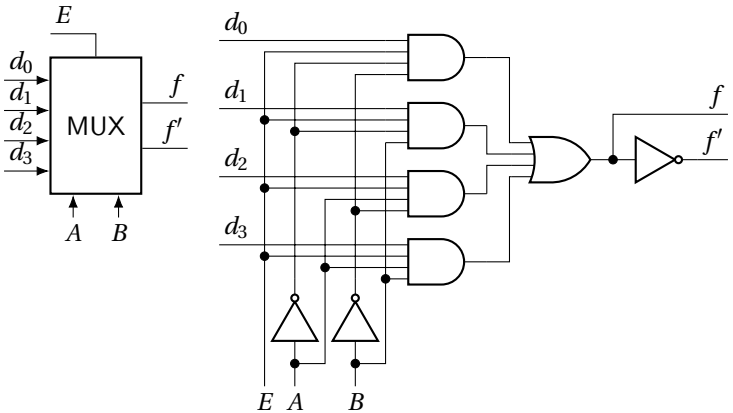
Multiplexer

- A **multiplexer (MUX)** (多路(复用)器) is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.
- It selects 1-out-of-n lines where n is usually `2`, `4`, `8`, or `16`.
- A block diagram of a multiplexer having 4 input data lines `d0`, `d1`, `d2`, and `d3` and complementary outputs `f` and `f'`.



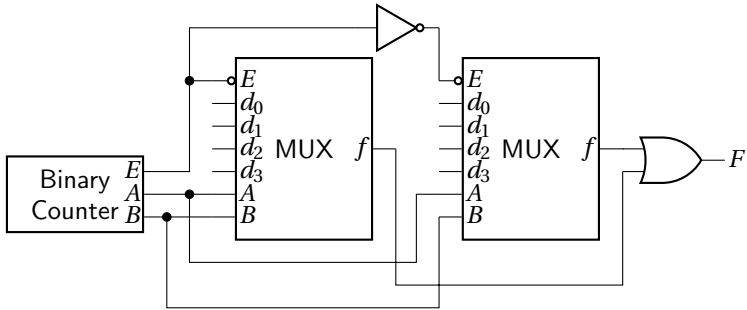
Multiplexer

- 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.

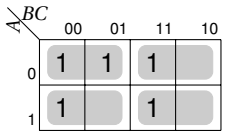


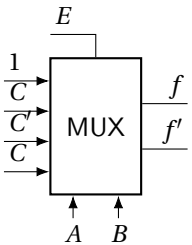
Interconnecting Multiplexers

- 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.

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: $f(A,B,C) = \sum(0,1,3,4,7)$.

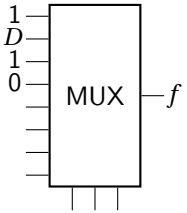




Multiplexer as a Boolean Function Generator

- 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
		...		



Outline of This Lecture

Summary

- Combinational circuit: consists of logic gates whose outputs at any time are determined from only the present combination of inputs.
  - Procedure of analysing combinational circuits.
  - Procedure of designing combinational circuits.
- Standard components: combinational circuits that are employed extensively in the design of digital systems.
  - Adders, subtractors, comparators, decoders, encoders, and multiplexers.
  - Available in integrated circuits as medium-scale integration (MSI) circuits.
  - Also used as standard cells in complex very large-scale integrated (VLSI) circuits such as application-specific integrated circuits (ASICs).

Essential Reading

- Essential reading for this lecture: pages 125-132, 148–158 of the textbook.
- Essential reading for next lecture: pages 133-148 of the textbook.

[1] M. M. Mano and M. Ciletti, *Digital design: with an introduction to the Verilog HDL*. Pearson, 2013