Lecture 6: Combinational Logic - Part II: Standard Components

CS207: Digital Logic

Jialin Liu

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

14 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: 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** (Lecture 5 and Lecture 6).
- "**Sequential circuits** employ memory elements in addition to logic gates. Their outputs are a function of the inputs and the state of the memory elements."
  - State of the memory elements: a function of previous inputs.
  - Outputs of a sequential circuit depend on values of present inputs and past inputs.

  → **Memory elements** (Lecture 7).
  → **Sequential logic** (from Lecture 8).

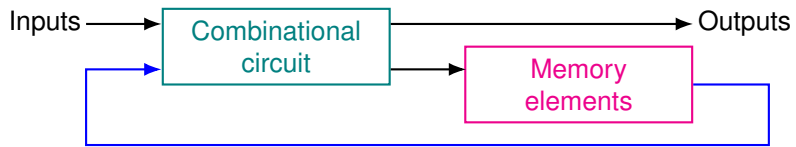


Figure: Block diagram of sequential circuit.

Recap: Analysis of Combinational Circuits

- **Analysis of a combinational circuit**: determine the function of the circuit.
  - Given a logic diagram.
  - Output 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.

Recap: Analysis of Combinational Circuits: Steps

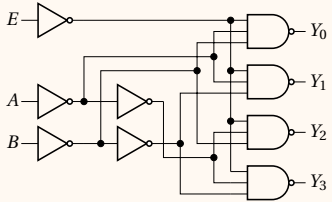
- **Input**: a logic diagram.
  1. **Make sure that the given circuit is combinational and not sequential.**
     - The diagram of a combinational circuit has logic gates with no feedback paths or memory elements.
     - **Feedback path**: a connection from the output of one gate to the input of a second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

Quiz

- You will be given 5 minutes in total.
- **Count as attendance in your final mark of this course.**
  - A wrong answer will be treated equally as a correct answer.
  - A meaningless answer receives 0 mark.
- **Requirement:** Take a piece of paper, write down
  1. your name,
  2. your student number,
  3. your answer on a piece of paper.
- **Submission:**
  - **Onsite students**: Submit your paper during the break (11:10-11:20am today).
  - **International students**: Send a photo of your solution to *liujl@sustech.edu.cn* during the break.
  - Late submissions receive 0 mark.

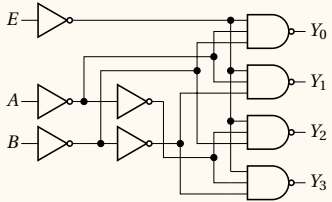
Quiz

Given the following logic diagram, write down (1) its output Boolean functions and (2) its truth table.



Quiz

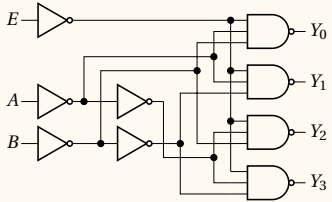
Given the following logic diagram, write down (1) its output Boolean functions and (2) its truth table.



$$Y_0 = (E'A'B')' = E + A + B$$
$$Y_1 = (E'A'B)' = E + A + B'$$
$$Y_2 = (E'AB')' = E + A' + B$$
$$Y_3 = (E'AB)' = E + A' + B'$$

Quiz

Given the following logic diagram, write down (1) its output Boolean functions and (2) its truth table.



$$Y_0 = (E'A'B')' = E + A + B$$
$$Y_1 = (E'A'B)' = E + A + B'$$
$$Y_2 = (E'AB')' = E + A' + B$$
$$Y_3 = (E'AB)' = E + A' + B'$$

E	A	B	Y_0	Y_1	Y_2	Y_3
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
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

E	A	B	Y_0	Y_1	Y_2	Y_3
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
1	X	X	1	1	1	1

This is a 2-to-4-line decoder with enable input !

Recap: 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).

Outline of This Lecture

Standard Components
- Decoder
- Encoder
- Multiplexer
- Magnitude Comparator

Iterative Combinational Circuits, Gate Delay & Timing hazard

Summary

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

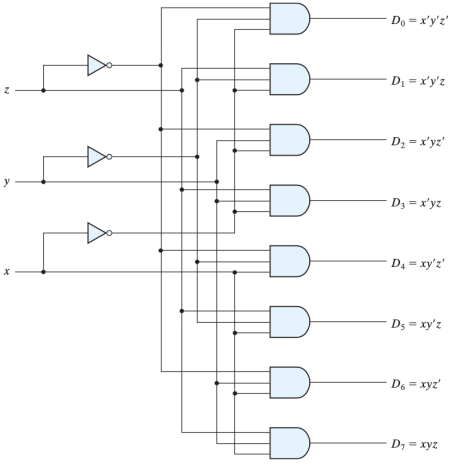
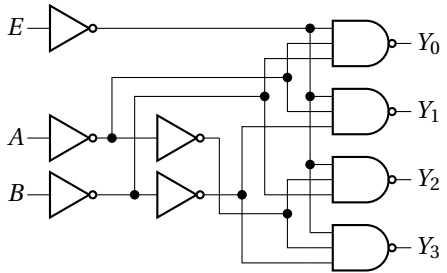


Figure: Figure 4.18 and Table 4.16 in [1]. Three-to-eight-line decoder. Example application: binary-to-octal conversion.

Related Terms

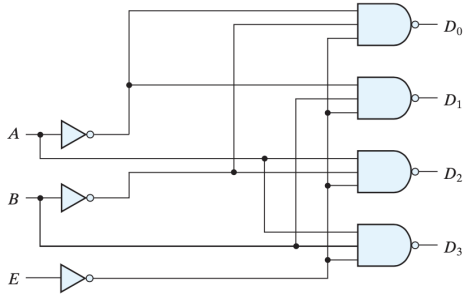
- **n-to-m-line decoder**
  - A n-to-m-line decoder is a combinational circuit that converts binary information from n input lines to m unique output lines with $m \leq 2^n$.



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

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

Main Applications of Decoders

- Minterm generator (最小项生成器):
  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 (最小项生成器)

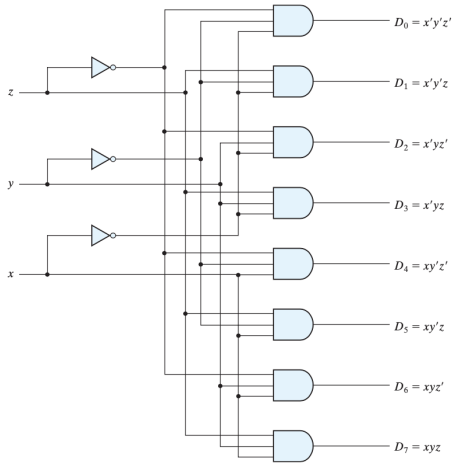
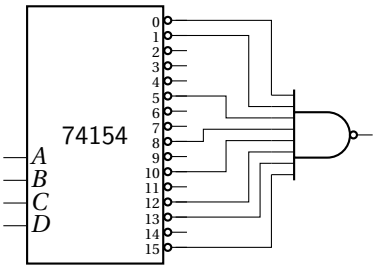


Figure: Figure 4.18 in [1]. Three-to-eight-line decoder as a minterm generator.

Decoder as Minterm Generator (最小项生成器)

- 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**.

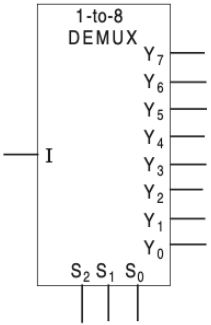


Figure: Figure 5.95 in [2].

Interconnect Decoders with Enable Inputs

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

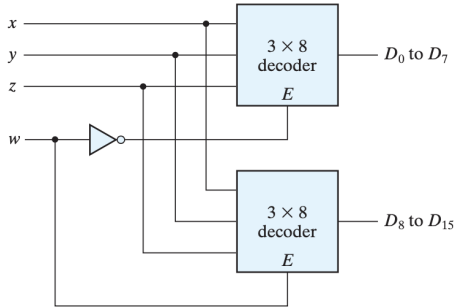


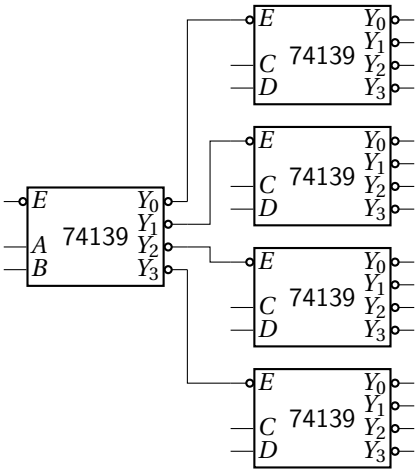
Figure: A 4×16 decoder constructed with two 3×8 decoders. Figure 4.20 [1].

Decoder Network

- When a large decoding network is required, it cannot be implemented in a single medium-scale integration (MSI) circuit.
  - 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

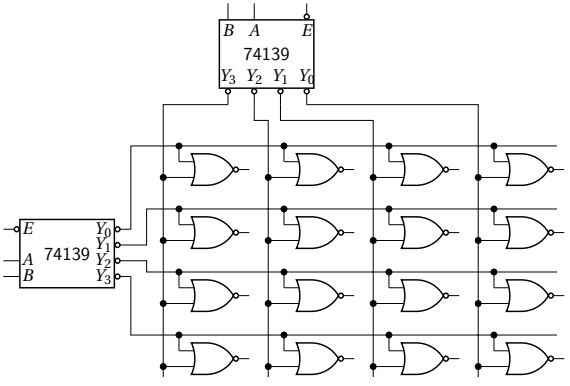
Tree Decoding: Example



Decoder Network

Coincident Decoding: Example

- A decoder with n inputs and 2^n outputs requires 2^n `AND` gates with n inputs per gate.
- 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.



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` gates.
  - 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 !
  - $x = D_4 + D_5 + D_6 + D_7$.
  - $y = D_2 + D_3 + D_6 + D_7$.
  - $z = D_1 + D_3 + D_5 + D_7$.

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

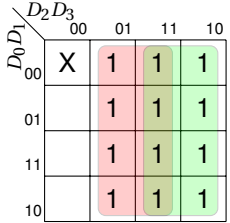
- Encoder circuit must have priority: **Priority encoder**.

Priority Encoder (优先编码器)

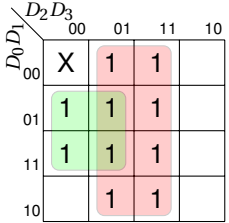
- **Priority encoder**: Encoder circuits must establish an input priority to ensure that only one input is encoded.
- Example:

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

$$x = D_2 + D_3$$



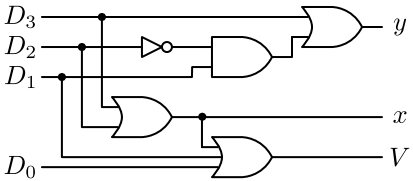
$$y = D_3 + D_1 D_2'$$



Priority Encoder

Example

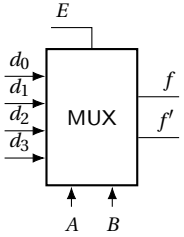
- $x = D_2 + D_3$.
- $y = D_3 + D_1 D_2'$.
- $V = D_0 + D_1 + D_2 + D_3$.

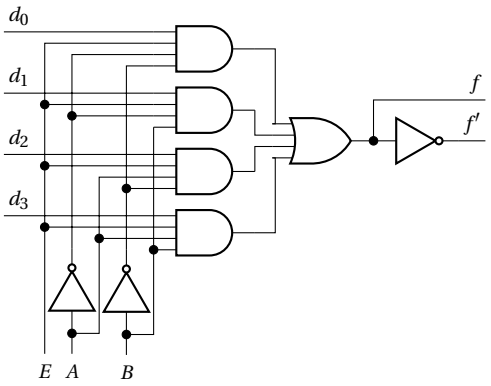


Outline

Multiplexer

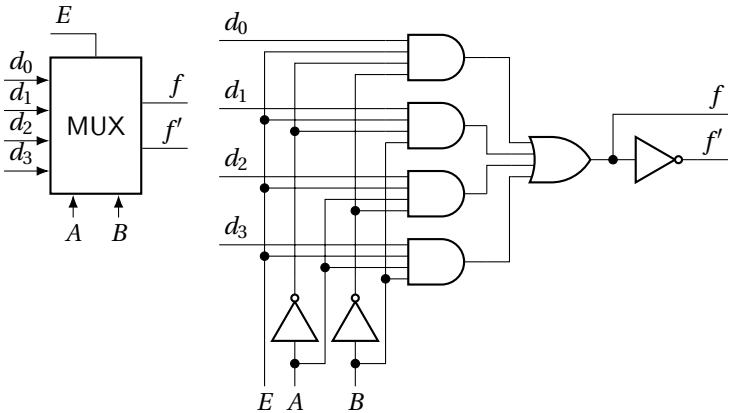
- A **multiplexer (MUX)** (多路复用器) is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.
- It selects 1-out-of-n lines where n is usually 2, 4, 8, or 16.
- Below is a multiplexer of 4 input data lines and 2 complementary outputs.





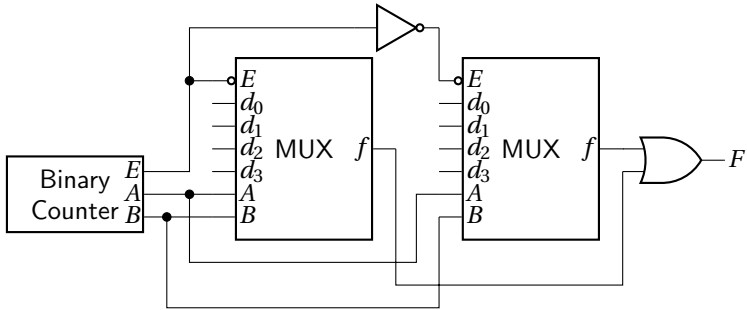
Multiplexer

- 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.

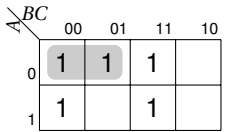


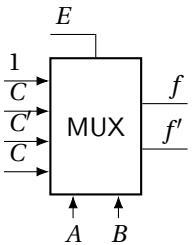
Interconnecting Multiplexers

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

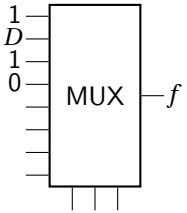




Multiplexer as a Boolean Function Generator

- 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
		...		



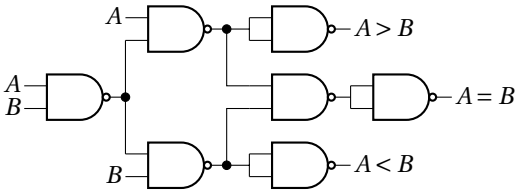
Outline

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: $A_i = B_i$.
  - $x_i = A_iB_i + A_i'B_i'$.
- Therefore:
  - $A = B$ 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 $A_2'B_2 + x_2A_1'B_1 + x_2x_1A_0'B_0 = 1$.

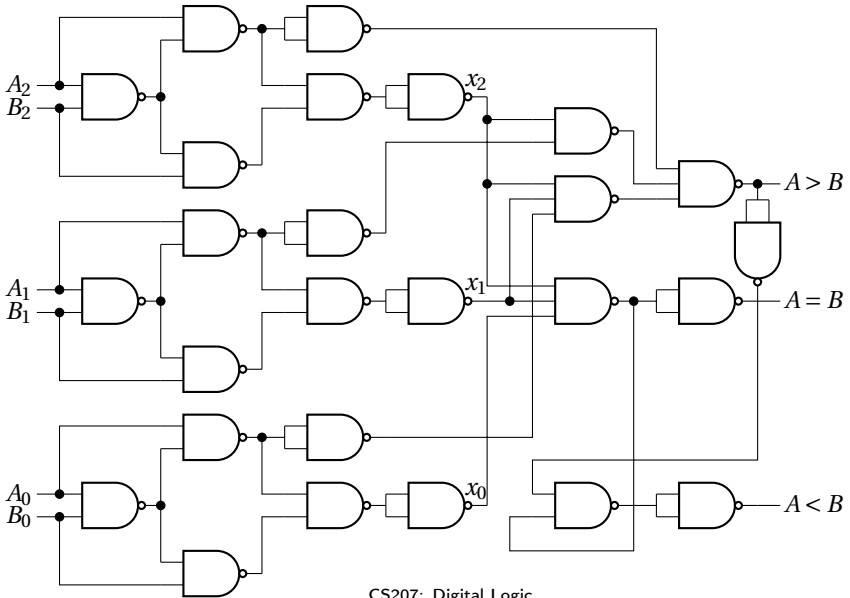
Magnitude Comparator

- 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.

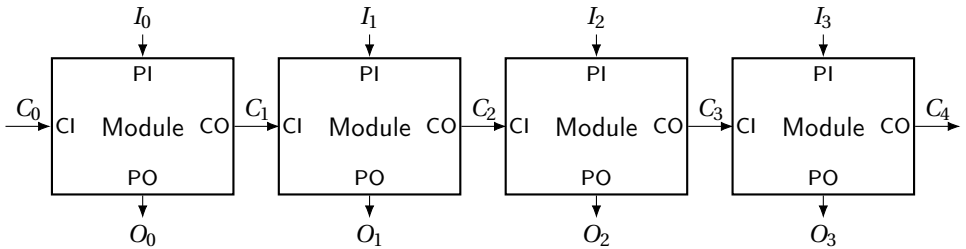
Magnitude Comparator



Outline of This Lecture

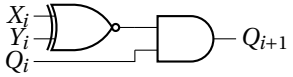
Iterative Combinational Circuits

- General structure: n identical modules for the same functionality.
- For problems that can be solved by an iterative algorithm:
  - Set C_0 to its initial value and set i to 0.
  - While $i < n$, repeat:
    - Use C_i and I_i to determine the values of O_i and C_{i+1}.
    - Increase i.
- Primary inputs/outputs and cascading inputs/outputs.

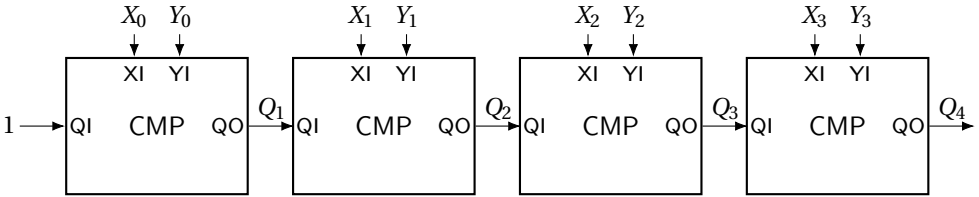


Iterative Comparator Circuits

- Complete circuit for comparing whether two n-bit values X and Y are the same:
  - Set Q_0 to 1 and set i to 0.
  - While $i < n$, repeat:
    - If Q_i is 1 and X_i equals Y_i, set Q_{i+1} to 1.
    - Increase i.
- Module: one-bit comparator.



$$Q_{i+1} = (X_i \oplus Y_i)' Q_i$$

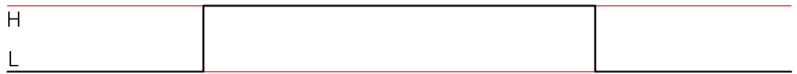


Gate Delays

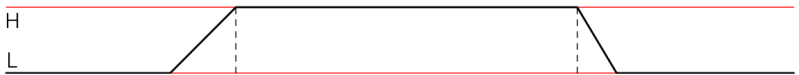
- When the input to a logic gate is changed, the output will not change immediately.
- The switching elements within a gate take a finite time to react to a change (transition) in input.
- As a result the change in the gate output is delayed w.r.t. to the input change.
- Such delay is called the **propagation delay** of the **logic gate delay**.
  - The propagation delay for a 0-to-1 output change may be different from the delay for a 1-to-0 change.

Gate Delays

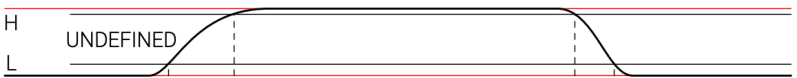
- Ideal case of zero-time switching:



- A more realistic approximation



- Actual timing for rise and fall times:

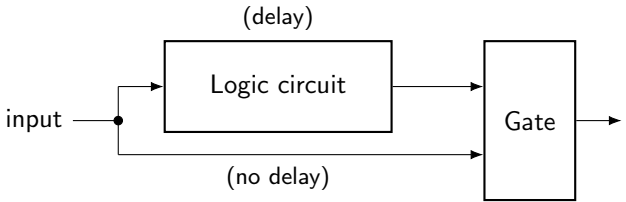


Effect of Gate Delays

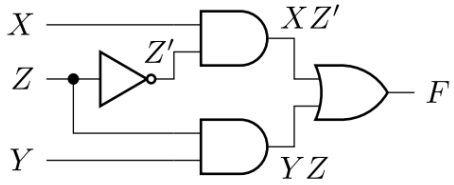
- The analysis of a combinational circuit ignoring delays can predict only its **steady-state behavior**.
  - Predicts a circuit's output as a function of its inputs assuming that the inputs have been stable for a long time, relative to the delays in the circuit's electronics.
- Because of circuit delays, the **transient behavior** of a combinational logic circuit may differ from what is predicted by steady-state analysis.
- **Timing hazard**: a circuit's output may produce a short pulse (**glitch**) at a time when steady state analysis predicts that the output should not change.

Timing Hazard

- t_{pLH} = low-to-high, t_{pHL} = high-to-low propagation time.
- A gate has measurable response time t_{pLH} and t_{pHL}. Around 10ns per gate.
- Delays through transmission gates can add up and introduce timing hazards.
- **static-1 hazard** is a short `0` glitch when for a changed input, we expect (by logic theorems) the output to remain constant `1`.
- **static-0 hazard** is a short `1` glitch when we expect the output to remain constant `0`.

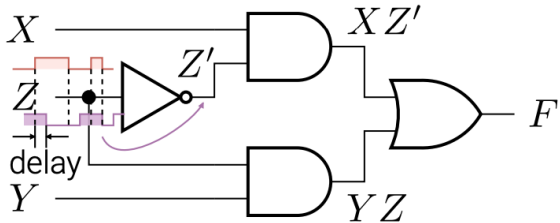


Circuit with A Static-1 Hazard



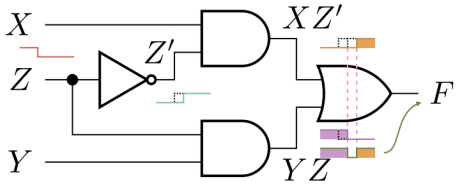
- Assume $XYZ = 111$, consider a transition to $XYZ = 110$.
- What we logically expect:
  - $F = XZ' + YZ$
- Before: $F = 1$, after: $F = 1$.
- `The output does not change!`

Circuit with A Static-1 Hazard



- But real world gates introduce delays.
- Input signal of each gate is shifted in the output by a constant delay.

Circuit with A Static-1 Hazard



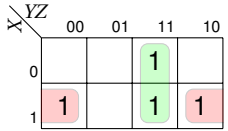
- Change occurs at input Z and propagates to output F along two paths with different delays.

Timing Hazards and Karnaugh Maps

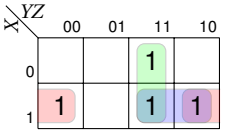
- Recall that in sum-of-products (`AND-OR`) circuits, AND gates correspond to prime implicants of the Karnaugh map.
- A potential hazard exists wherever two adjacent 1-cells in a Karnaugh map are not covered by a single product term (prime implicant).
- A hazard occurs when there is a transition between adjacent prime implicants.
- To eliminate hazards, find a cover in which all adjacent 1-cells are covered by a prime implicant.
  - Define **consensus** prime implicants, as product terms not covered in F.
  - Therefore, include an extra product term to cover the hazardous input combination

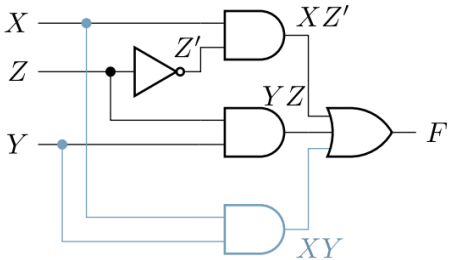
Eliminating The Timing Hazard

Minimal cost: $F = XZ' + YZ$



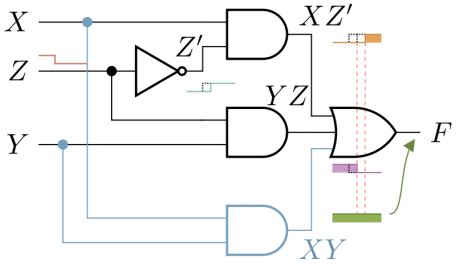
Eliminate hazard: $F = XZ' + YZ + XY$





Eliminating The Timing Hazard

- So, how does this help?



- Unexpected glitch is eliminated!
  - A change in input variable Z causes a transition between two adjacent 1-cells but now these 1-cells are included in the product term XY.

Dynamic Hazards

- Dynamic hazard: Output signal is supposed to change 1-0 or 0-1 but a short oscillation occurs before the output settles to its new logic value.
- Occurs if there are multiple paths with different delays from the changing input to the changing output.
- In practice many input variables, multiple outputs; solved by computer programs.

Outline of This Lecture

Summary

- 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).
- Iterative combinational circuits are useful, but the gate delay is not negligible.
  - The analysis of a combinational circuit ignoring delays can predict only its steady-state behaviour.
  - Because of circuit delays, the transient behaviour of a combinational logic circuit may differ from what is predicted by steady-state analysis.
  - Timing hazard: a circuit's output may produce a short pulse (glitch) at a time when steady state analysis predicts that the output should not change.

Essential Reading

- Essential reading for this lecture: pages148–158 of the textbook.
- Essential reading for next lecture: pages 190–196 of the textbook.

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