Combination Logic

Computer Organization
502046

ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

Acknowledgement

This slide show is intended for use in class, and is not a complete document. Students need to refer to the book to read more lessons and exercises. Students have the right to download and store lecture slides for reference purposes; Do not redistribute or use for purposes outside of the course.

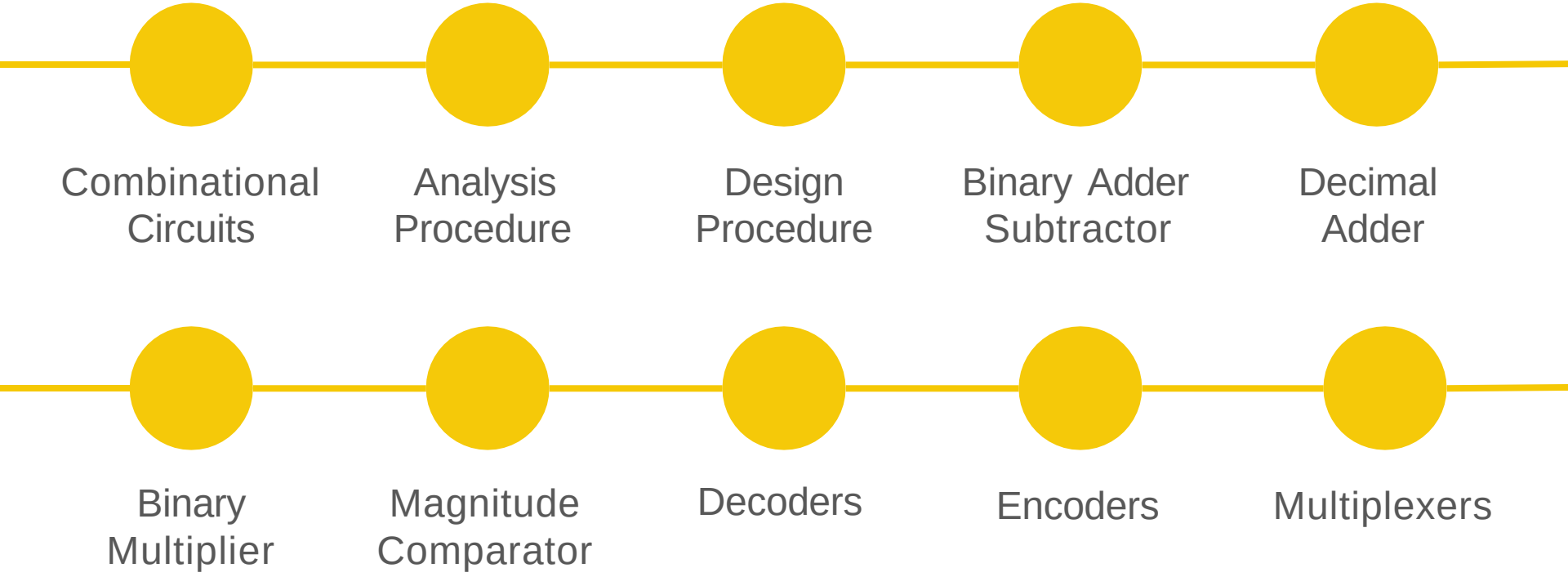
[1] Morris R. Mano (Author), Michael D. Ciletti, [2019] **Digital Design: With an Introduction to the Verilog HDL,** chapter 3 - Gate level minimization, 5th Edition.

✉ **trantrungtin.tdtu.edu.vn**

Chapter Objectives

1. Understand combination circuits.
2. Know how to analyse combination circuits.
3. Know how design a combination circuit.
4. Design by blocks

Outline


Combinational Circuits
Analysis Procedure
Design Procedure
Binary Adder Subtractor
Decimal Adder
Binary Multiplier
Magnitude Comparator
Decoders
Encoders
Multiplexers

Syllabus

4.1 Introduction

A combinational circuit:

- consists of logic gates whose outputs at any time are determined from only the present combination of inputs.
- performs an operation that can be specified logically by a set of Boolean functions.

Sequential circuits:

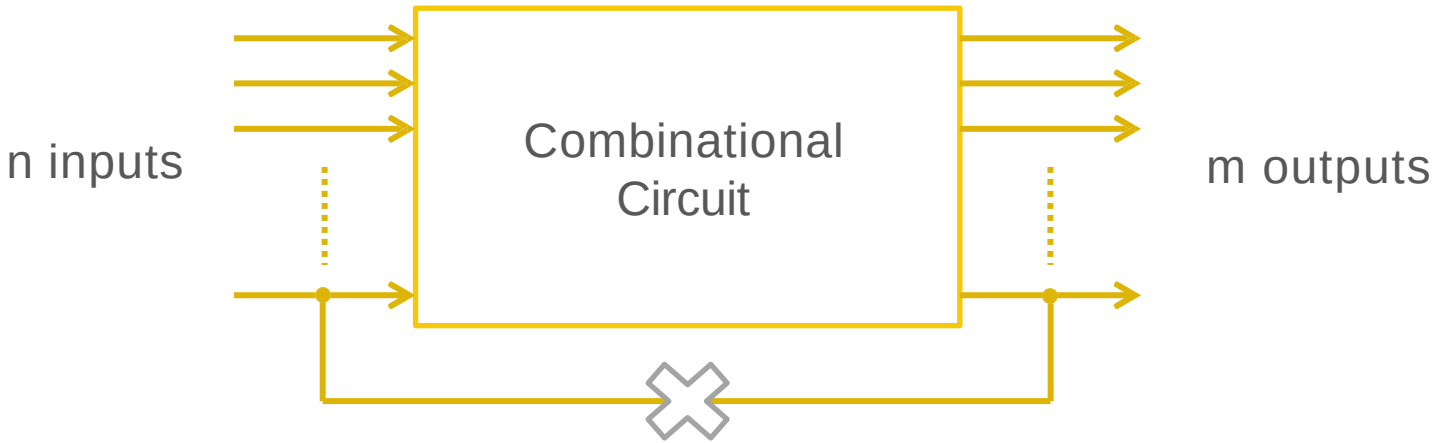
- employ storage elements in addition to logic gates.
- outputs are a function of the inputs and the state of the storage elements.
- The outputs depend not only on present values of inputs, but also on past inputs.

4.1 Introduction

- Combinational circuit consist of
  - Input variables
  - Logic gates
    - Accept signals from the inputs
    - Generate signals to the output
  - Output variables

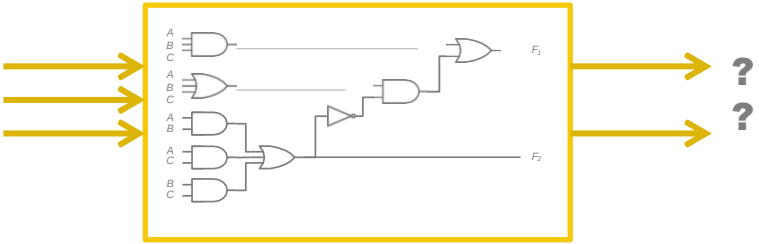
4.2 Combinational circuits

- For n input variable, there are 2n possible binary input combinations.
- For each possible input combination there is one possible output value.



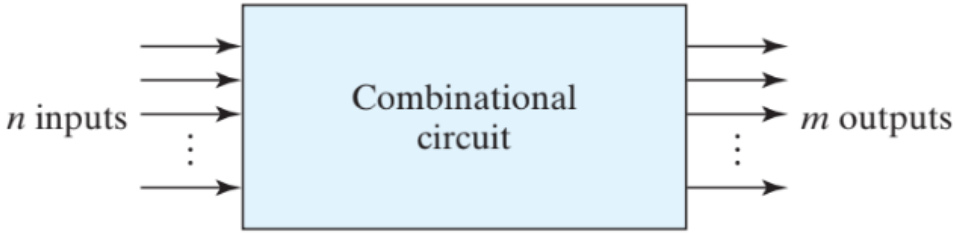
4.2 Combinational circuits

- Analysis
  - Given a circuit, find out its function
  - Function may be expressed as:
    - Boolean function
    - Truth table
- Design
  - Given a desired function, determine its circuit
  - Function may be expressed as:
    - Boolean function
    - Truth table



4.2 Combinational circuits

- Combinational logic gates react to the values of the signals at their inputs and produce the value of the output signal, transforming binary information from the given input data to a required output data.
- In many applications, the source and destination are storage registers.



4.2 Describe a combination circuit

- A truth table that lists the output values for each combination of input variables.
- By m Boolean functions, one for each output variable. Each output function is expressed in terms of the n input variables.

4.3 Analysis Procedure

- The analysis of a combinational circuit requires that we determine the function that the circuit implements. This task starts with one or more of the following:
- a given logic diagram and culminates with a set of Boolean functions,
- a truth table,
- an explanation of the circuit operation.
- The first step in the analysis is to make sure that the given circuit is combinational
    - No feedback paths or
    - No memory elements

4.3 Bool functions from a logic diagram

- 1. Label all gate outputs that are a function of input variables with arbitrary symbols but with meaningful names. Determine the Boolean functions for each gate output.
- 2. Label the gates that are a function of input variables and previously labeled gates with other arbitrary symbols. Find the Boolean functions for these gates.
- 3. Repeat the process outlined in step 2 until the outputs of the circuit are obtained.
- 4. By repeated substitution of previously defined functions, obtain the output. Boolean functions in terms of input variables.

4.3 Truth table from the logic diagram

- 1. Determine the number of input variables in the circuit. For n inputs, form the 2n possible input combinations and list the binary numbers from 0 to (2n - 1) in a table.
- 2. Label the outputs of selected gates with arbitrary symbols.
- 3. Obtain the truth table for the outputs of those gates which are a function of the input variables only.
- 4. Proceed to obtain the truth table for the outputs of those gates which are a function of previously defined values until the columns for all outputs are determined.

4.3 By means of logic simulation.

- This is not practical, however, because the number of input patterns that might be needed to generate meaningful outputs could be very large.
- But simulation has a very practical application in verifying that the functionality of a circuit actually matches its specification. In Section 4.12, we demonstrate the logic simulation and verification of the circuit of Fig. 4.2 , using Verilog HDL.

4.4 Design Procedure

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

4.4 Design Procedure

- A truth table for a combinational circuit consist of:
  - Input columns
    - Obtained from 2n binary numbers for the n input variables.
  - Output columns
    - Determined from the stated specifications.
    - Output functions specified in the truth table give exact definition of the combinational circuit.

4.4 Design Procedure

- The output binary functions listed in the truth table are simplified by any method:
  - Algebraic manipulation
  - The map method
  - Computer – based simplification program
- There is a variety of simplified expressions from which to choose.

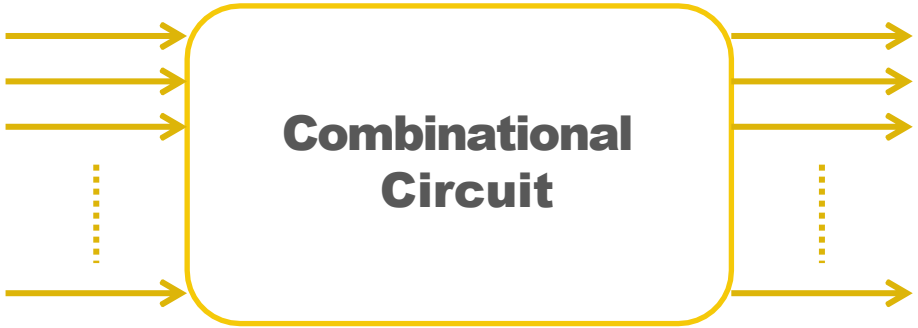
4.4 Design Procedure

- Practical design must consider such constraints:
  - The number of gates
  - Number of inputs to a gate
  - Propagation time of the signal through the gates
  - Number of interconnections
  - Limitations of the driving capability of each gate
  - Etc.

4.4 Design Procedure: Code conversion example

- Code converter is a circuit that makes the two systems compatible even though each uses a different binary code.
- To convert from binary code A to binary code B;

Input:
supply the bit combination of elements specified by code A

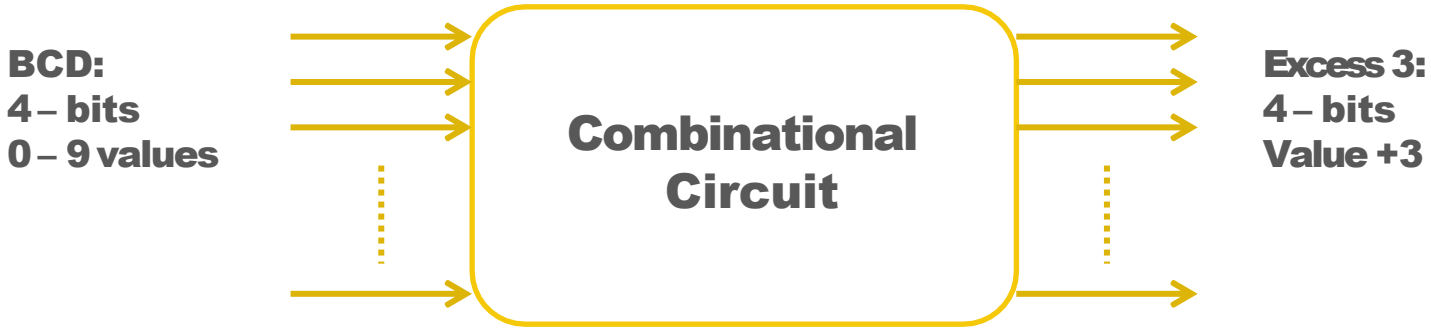


Output:
generate the corresponding bit combination of code B.

4.4 Design Procedure: Code conversion example

- Example:

- Design a circuit to convert a “BCD” code to “Excess 3” code.



4.4 Design Procedure: Code conversion example

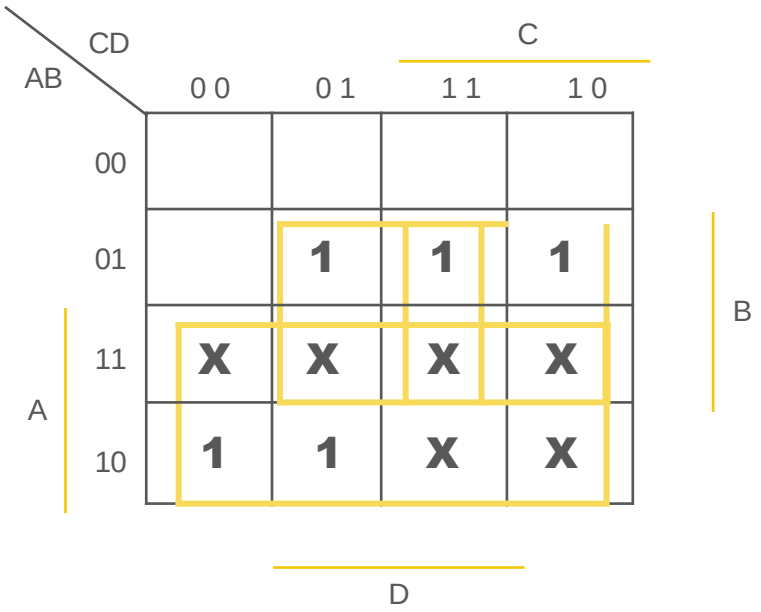
- Each code uses 4-bits to represent a decimal digit.
  - 4 input variables
    - A, B, C, D
  - 4 output variables
    - w, x, y, z
  - 2^4 = 16 bit combinations but only 10 have meaning in BCD.
  - Rest 6 bit combinations are don't-care combinations.

Input BCD	Output Excess-3
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
1 0 1 0	x x x x
1 0 1 1	x x x x
1 1 0 0	x x x x
1 1 0 1	x x x x
1 1 1 0	x x x x
1 1 1 1	x x x x

4.4 Design Procedure: Code conversion example

Input BCD	Output Excess-3
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
1 0 1 0	x x x x
1 0 1 1	x x x x
1 1 0 0	x x x x
1 1 0 1	x x x x
1 1 1 0	x x x x
1 1 1 1	x x x x

w's karnaugh map

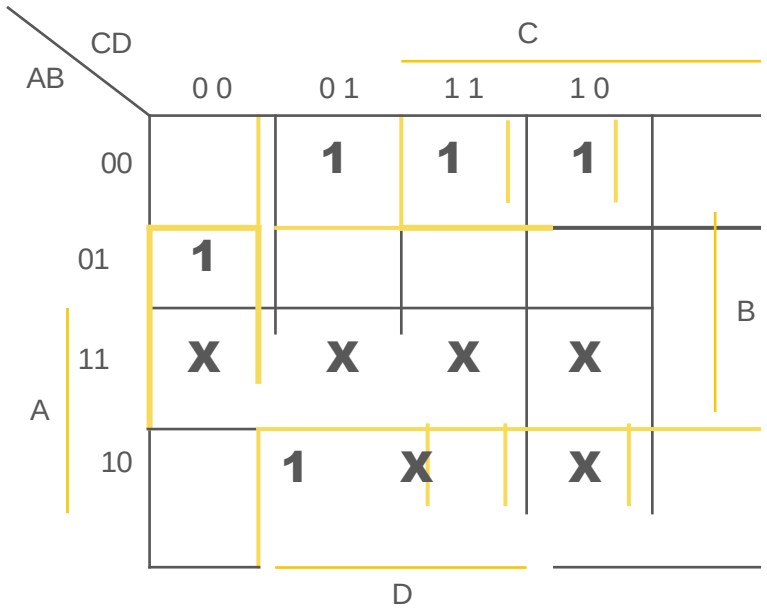


$w = A + BC + BD$

4.4 Design Procedure: Code conversion example

Input BCD	Output Excess-3
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
1 0 1 0	x x x x
1 0 1 1	x x x x
1 1 0 0	x x x x
	x x x x
	x x x x
	x x x x

x's karnaugh map

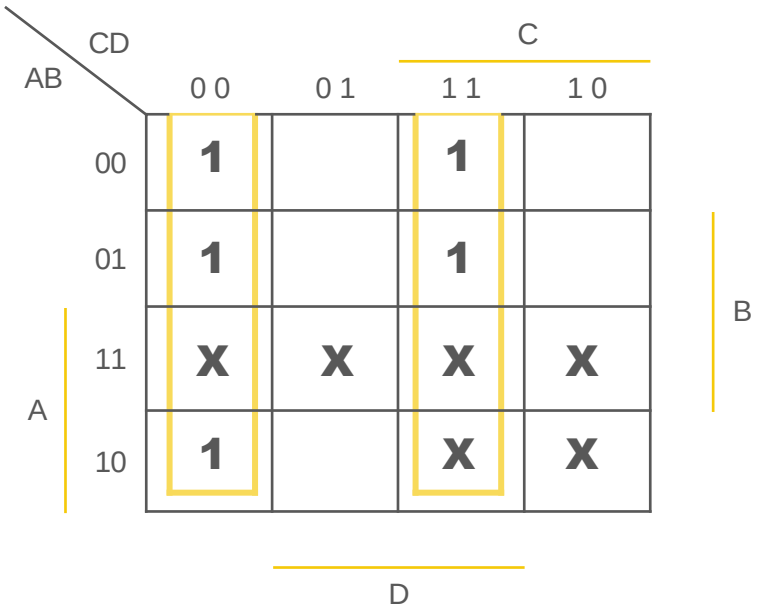


$x = B'C + B'D + BC'D'$

4.4 Design Procedure: Code conversion example

Input BCD	Output Excess-3
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
1 0 1 0	x x x x
1 0 1 1	x x x x
1 1 0 0	x x x x
1 1 0 1	x x x x
1 1 1 0	x x x x
1 1 1 1	x x x x

y's karnaugh map

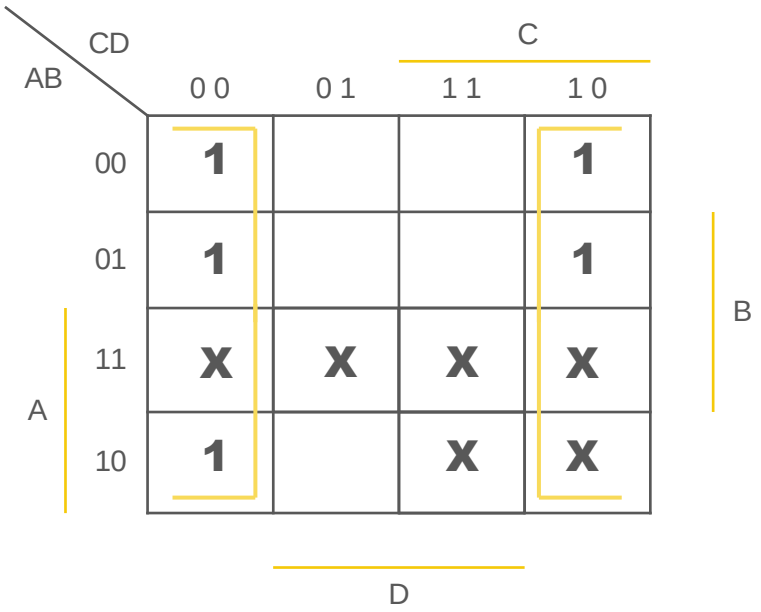


y = CD + C'D'

4.4 Design Procedure: Code conversion example

Input BCD	Output Excess-3
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
1 0 1 0	x x x x
1 0 1 1	x x x x
1 1 0 0	x x x x
1 1 0 1	x x x x
1 1 1 0	x x x x
1 1 1 1	x x x x

z's karnaugh map

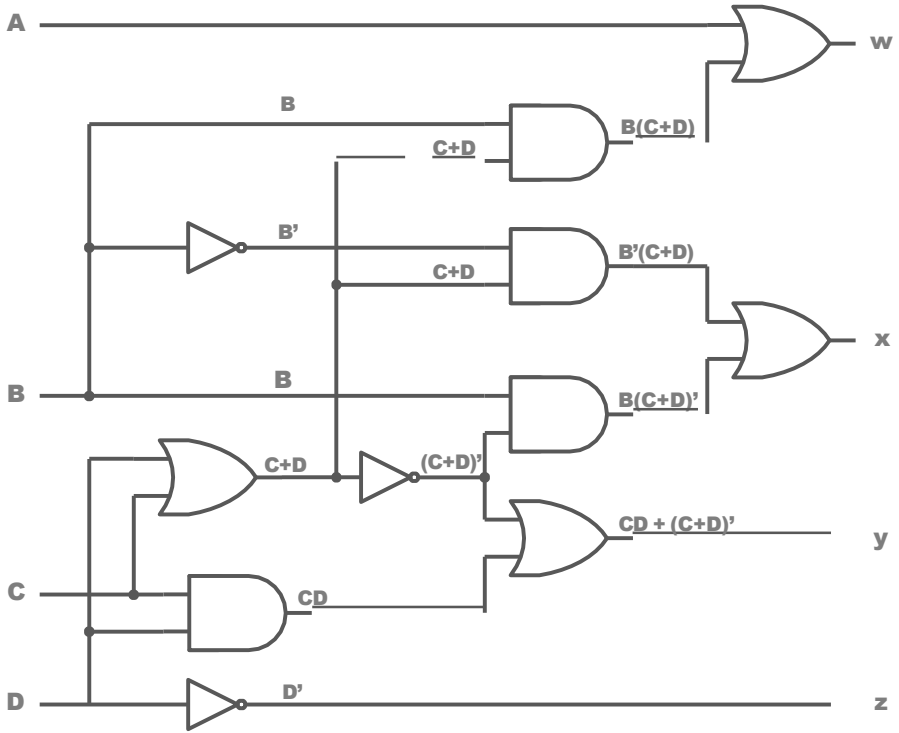


z = D'

4.4 Design Procedure: Code conversion example

- w = A + BC + BD = A + B (C + D)
- x = B'C + B'D + BC'D' = B' (C+D) + BC'D'

  x = B' (C + D) + B (C+D)'
- y = CD + C'D' = CD + (C + D)'
- z = D'



Notes: video

- https://www.youtube.com/watch?v=_QuDC2-IAME
- https://www.tinkercad.com/things/kbfLeffKM8a-copy-of-bcd-to-excess-3-/editel?tenant=circuits

Exercises – Check slide CH4B_self-study

4.5 Binary Adder – Subtractor

4.6 Decimal Adder

4.7 Binary Multiplier

4.8 Magnitude Comparator

4.9 Decoders

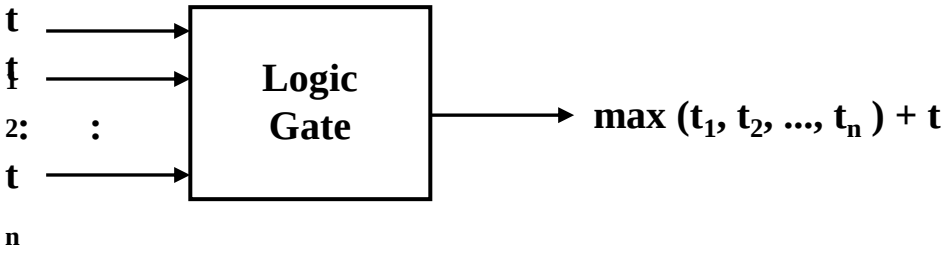
4.10 Encoders

4.11 Multiplexers

Circuit delays

- Given a logic gate with delay t. If inputs are stable at times t1, t2, ..., tn, then the earliest time in which the output will be stable is:

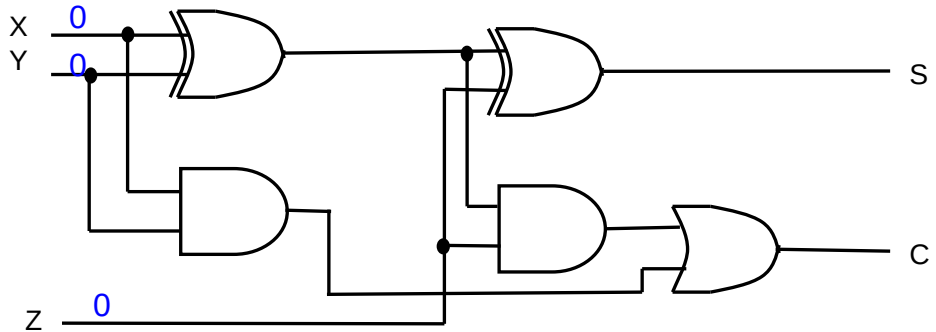
  max(t1, t2, ..., tn) + t



  - To calculate the delays of all outputs of a combinational circuit, repeat above rule for all gates.

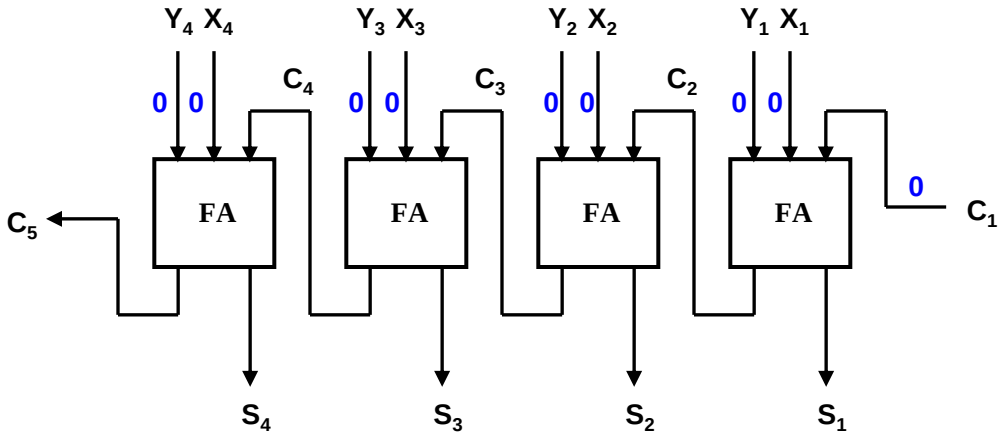
Circuit delays

- As a simple example, consider the full adder circuit where all inputs are available at time 0. Assume each gate has delay t.



Circuit delays

- More complex example: 4-bit parallel adder.

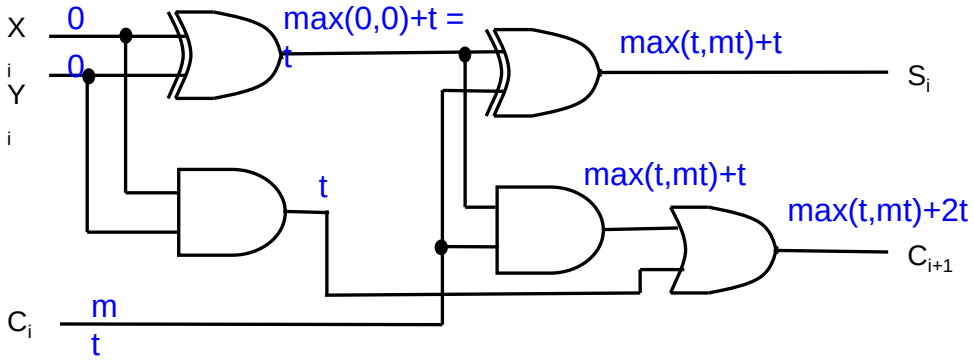


Circuit delays

- Analyse the delay for the repeated block.

where X_i, Y_i are stable at 0t, while C_i is assumed to be stable at mt.

  - Performing the delay calculation:



CIRCUIT DELAYS (5/5)

- Calculating:
  - When i=1, m=0; S1 = 2t and C2 = 3t
  - When i=2, m=3; S2 = 4t and C3 = 5t
  - When i=3, m=5; S3 = 6t and C4 = 7t
  - When i=4, m=7; S4 = 8t and C5 = 9t
- In general, an n-bit ripple-carry parallel adder will experience the following delay times:
  - Sn = ?
  - Cn+1 = ?
- Propagation delay of ripple-carry parallel adders is proportional to the number of bits it handles.
- Maximum delay: ?