Ollscoil na Gaillimhe
University of Galway

CT101 Computing Systems

Dr. Bharathi Raja Chakravarthi
Lecturer-above-the-bar
Email: bharathi.raja@universityofgalway.ie

University *of* Galway.ie

Ollscoil na Gaillimhe
University of Galway

Combinational Logic

University
*of*Galway.ie

Introduction

- Digital logic circuits in digital systems can be categorized into two main types: combinational circuits and sequential circuits.

Combinational circuits:

- Combinational circuits are comprised of logic gates.
- The outputs of a combinational circuit are solely determined by the current combination of inputs.
- These circuits perform operations that can be logically specified by a set of Boolean functions.

Introduction

Sequential circuits:

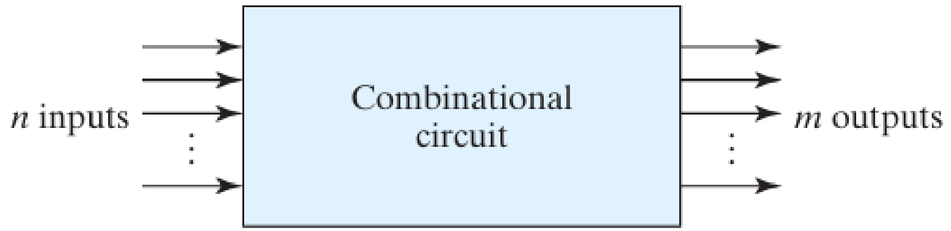
- Sequential circuits not only consist of logic gates but also include storage elements.
- The outputs of a sequential circuit depend on both the current inputs and the state of the storage elements.
- Unlike combinational circuits, sequential circuits exhibit behavior that relies on a time sequence of inputs and internal states, as the state of the storage elements is influenced by past inputs.

Combinational Circuits

- A combinational circuit is constructed by interconnecting logic gates.
- In combinational logic circuits, the behavior of logic gates depends solely on the values of the signals at their inputs.
- These circuits take input binary data and transform it into the required output data, reacting to the values of the input signals.

Combinational Circuits

- A block diagram of a combinational circuit is illustrated in Figure.
- The "n" input binary variables originate from an external source, while the "m" output variables are generated by the internal combinational logic circuit and are directed to an external destination.
- Physically, each input and output variable exists as an analog signal, but these values are interpreted as binary signals, representing logic 1 and logic 0.



Note:

Logic simulators show only 0's and 1's, not the actual analog signals.

Combinational Circuits

- In many practical applications, the source and destination points for data in combinational circuits are storage registers.
- When these registers are included along with the combinational gates, the entire circuit is considered to be a sequential circuit, as it combines logic operations with the use of storage elements.
- For a combinational circuit with "n" input variables, there are 2^n possible combinations of binary inputs.

Combinational Circuits

- Each possible input combination corresponds to a unique set of output values for each output variable.
- Combinational circuits can be precisely specified using a truth table that enumerates the output values for every combination of input variables.
- Alternatively, a combinational circuit can be described using "m" Boolean functions, where each function corresponds to one of the output variables, expressed in terms of the "n" input variables.

Combinational Circuits

- The purpose of the current topic is to leverage the knowledge acquired in previous topics to establish systematic analysis and design procedures for combinational circuits.

- These procedures will be essential for addressing three main tasks:

    1. Analyzing the behavior of a given logic circuit.
    2. Synthesizing a circuit that exhibits a specified behavior.
    3. Writing Hardware Description Language (HDL) models for common circuits.

Combinational Circuits

- Combinational circuits play a vital role in the design of digital systems, serving specific digital functions commonly required in digital system design.
- These circuits are available as integrated circuits and are categorized as standard components.
- Standard combinational circuits are widely used in various digital systems and are considered essential building blocks for digital design.

Combinational Circuits

- The most crucial standard combinational circuits, including:
    1. Adders
    2. Subtractors
    3. Comparators
    4. Decoders
    5. Encoders
    6. Multiplexers

- These components are available as medium-scale integration (MSI) circuits and are also employed as standard cells in complex Very Large Scale Integrated (VLSI) circuits like application-specific integrated circuits (ASICs).
- Within VLSI circuits, standard cell functions are interconnected in a manner similar to their use in multiple-IC MSI design.

Analysis Procedure

- The analysis of a combinational circuit requires that we determine the function that the circuit implements.
- This task starts with a given logic diagram and culminates with a set of Boolean functions, a truth table, or, possibly, an explanation of the circuit operation.
- If the logic diagram to be analyzed is accompanied by a function name or an explanation of what it is assumed to accomplish, then the analysis problem reduces to a verification of the stated function.
- The analysis can be performed manually by finding the Boolean functions or truth table or by using a computer simulation program.

Analysis Procedure

- **Initial Step:** Determine if the circuit is combinational or sequential.
  - Combinational circuits have no feedback paths or memory elements.

- **Feedback Path Definition:**
  - A loop from the output of one gate back to the input of another.
  - The presence of feedback paths indicates a sequential circuit.

- Sequential circuits require special methods for analysis and are not considered in this context.

Analysis Procedure

- **Verification of Combinational Circuit:**
    - Once verified, move on to deriving output Boolean functions or truth table.
- **Interpreting Circuit Functionality:**
    - Necessary if the function of the circuit is under investigation tables.
    - Experience with various digital circuits aids in successful interpretation.

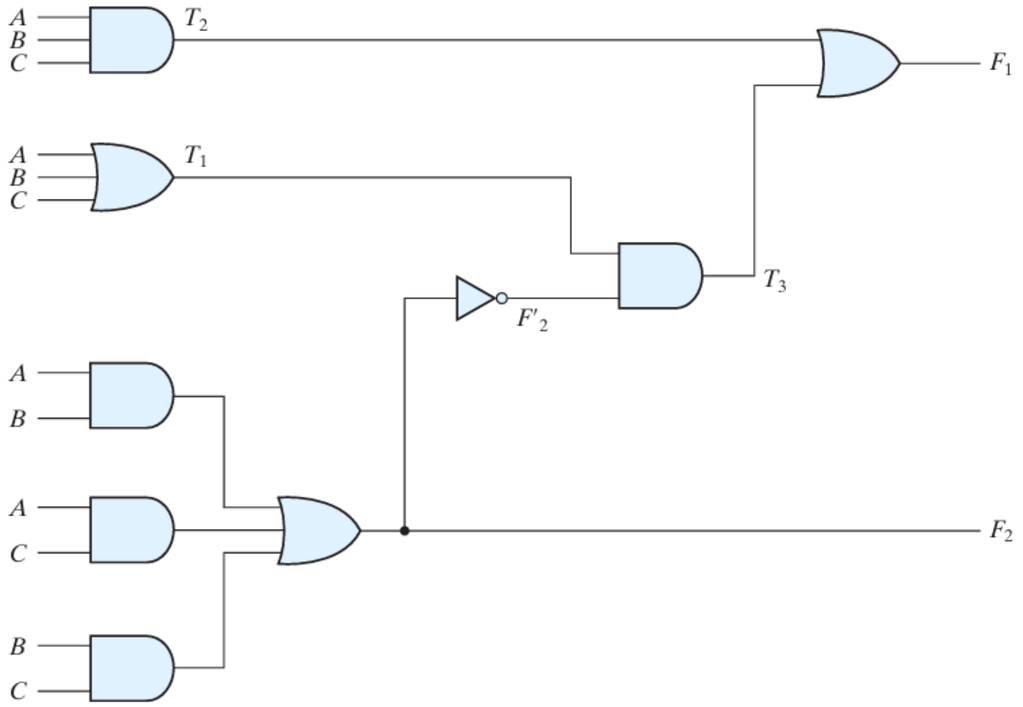
Analysis Procedure

Steps to Obtain Output Boolean Functions:

1. Label gate outputs that are functions of input variables.
2. Determine Boolean functions for each labeled gate output.
3. Label additional gates that are functions of input variables and previously labeled gates.
4. Find Boolean functions for these new labels.
5. Repeat until you reach the outputs of the circuit.
6. Derive output Boolean functions in terms of input variables through substitution.

Example Analysis:

- Circuit has three binary inputs: A, B, C.
- Circuit has two binary outputs: F1 and F2.
- Intermediate symbols are used for gate outputs.
- Outputs T1 and T2 are functions only of input variables.
- Output F2 can be directly derived from the input variables.



Analysis Procedure

- The Boolean functions for these three outputs are

$$F_2 = AB + AC + BC$$

$$T_1 = A + B + C$$

$$T_2 = ABC$$

- Next, we consider outputs of gates that are a function of already defined symbols:

$$T_3 = F'_2 T_1$$
$$F_1 = T_3 + T_2$$

Analysis Procedure

To obtain F_1 as a function of A, B , and C, we form a series of substitutions as follows:

$$
\begin{aligned}
F_1 &= T_3 + T_2 = F'_2 T_1 + ABC = (AB + AC + BC)'(A + B + C) + ABC \\
&= (A' + B')(A' + C')(B' + C')(A + B + C) + ABC \\
&= (A' + B'C')(AB' + AC' + BC' + B'C) + ABC \\
&= A'BC' + A'B'C + AB'C' + ABC
\end{aligned}
$$

Analysis Procedure

- **Investigating Information Transformation**:
    - Drawing the circuit based on derived Boolean expressions can help recognize familiar operations.
- **Example Insight into Behavior:**
    - Boolean functions for F_1 and F_2 match those describing a "full adder."
    - A Boolean representation alone doesn’t fully explain circuit behavior.
- **Deriving the Truth Table:**
    - Straightforward once output Boolean functions are known.

Analysis Procedure

Steps to Obtain Truth Table Directly from Logic Diagram:

1. Identify the number of input variables (n).
    - Create 2^n possible input combinations.
    - List binary numbers from 0 to **$(2^n - 1)$** in a table.
2. Label outputs of selected gates with arbitrary symbols.
3. Derive the truth table for outputs based only on input variables.
4. Continue to fill in the truth table based on outputs of gates that are functions of previously defined values.

Analysis Procedure

From the Truth Table below:

- Eight combinations for three input variables: A, B, and C.
- F_2 is 1 when any two or three of the inputs A, B, C are 1.
- F_2' (complement of F_2) is the inverse of F_2.
- T_1 and T_2 represent OR and AND functions of input variables, respectively.
- T_3 is 1 when both T_1 and F_2 are 1; otherwise, it's 0.
- F_1 is 1 when either T_2 or T_3 or both are 1.

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

Analysis Procedure

- **Comparison with Full Adder:**
    - Truth table for A, B, C, F_1, and F_2 matches that of the full adder (x, y, z, S, C).
- **Alternative Analysis Method: Logic Simulation:**
    - Not practical for circuits with large numbers of input patterns.
    - Useful for verifying if the functionality matches the specification.

Design Procedure

- **Design Process Overview for Combinational Circuits:**
    - Starts from a design specification.
    - Ends with a logic circuit diagram or set of Boolean functions.
- **Step-by-Step Procedure:**

1. **Determine Inputs and Outputs:**
    - Based on circuit specifications, find the number of required inputs and outputs.
    - Assign a symbol to each.

Design Procedure

2. **Create Truth Table:**

    - Define the required relationships between inputs and outputs.
3. **Simplified Boolean Functions:**

    - Find the simplified Boolean functions for each output as a function of input variables.
4. **Logic Diagram & Verification:**

    - Draw the logic diagram.
    - Verify the design's correctness either manually or via simulation.

Design Procedure

- A truth table for a combinational circuit consists of input columns and output columns.
- The input columns are obtained from the 2^n binary numbers for the n input variables.
- The binary values for the outputs are determined from the stated specifications.

Design Procedure

Importance of Accurate Specifications:

- Truth table gives an exact definition of the combinational circuit.
- Critical to interpret verbal specifications correctly.
- Incorrect interpretation can lead to a flawed truth table and, consequently, a flawed circuit.

Simplifying Output Binary Functions:

Methods for Simplification:

- Algebraic manipulation.
- Map method.
- Computer-based simplification programs.

Choice of Simplified Expressions:

- Various simplified expressions may be available.
- Choice is guided by specific application criteria.

Criteria for Choosing an Implementation:

1. **Number of Gates:** Limit on the total gates used.
2. **Number of Inputs to a Gate:** Limits on how many inputs a single gate can handle.
3. **Propagation Time:** Time it takes for a signal to travel through the gates.
4. **Number of Interconnections:** Complexity in terms of wiring and connections.
5. **Driving Capability:** Maximum number of gates that can be connected to the output of a given gate.
6. **Application-Specific Criteria:** Other limitations or requirements based on the specific use-case.

Design Procedure

Steps in Simplification:

- Start by achieving simplified Boolean functions in a standard form.
- Further optimize based on performance needs specific to the application.

Code Conversion Example

Why Code Converters are Needed?

- Different digital systems may use different binary codes for the same information.
- Sometimes, the output of one system needs to be used as the input to another system.

What is a Code Converter?

- A circuit that makes two different systems compatible by transforming one binary code into another.

Code Conversion Example

How It Works?

- Input Lines: Provide the bit combination according to the source code (e.g., Code A).
- Logic Gates: A combinational circuit made up of logic gates performs the actual transformation.
- Output Lines: Produce the bit combination conforming to the target code (e.g., Code B).

Example:

- Conversion Type: Binary Coded Decimal (BCD) to Excess-3 code.
- Design: Achieved through a combinational circuit using logic gates.

Code Conversion Example

Bit Combinations for Code Conversion:

- **Input Codes**: BCD with four input variables A,B,C,D
- **Output Codes**: Excess-3 with four output variables w,x,y,z

Input BCD				Output Excess-3 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

Code Conversion Example

Special Features of Truth Table:

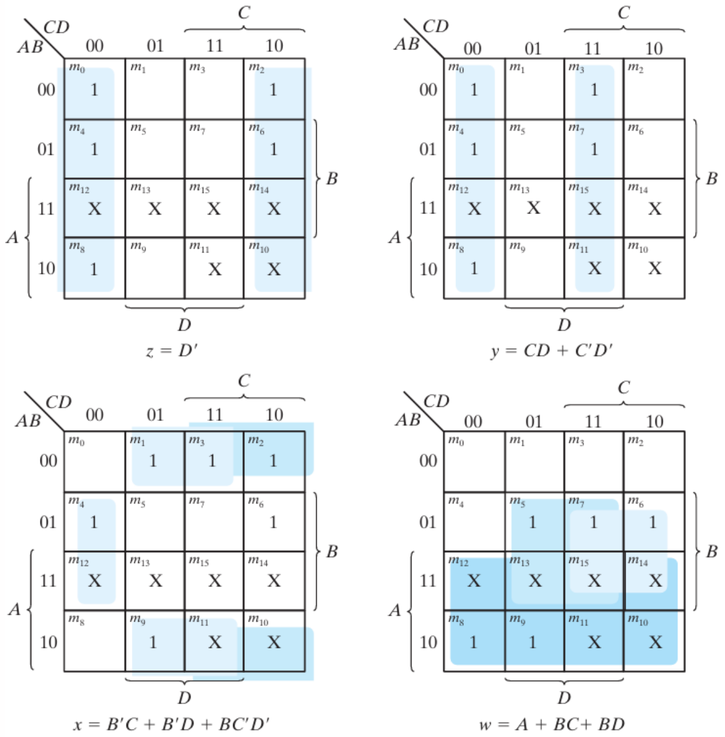
- **Valid Combinations**: 10 valid bit combinations to represent decimal digits 0-9.
- **Total Combinations**: 2^4 =16 possible combinations for four binary variables.
- **Don't-Care Combinations**: 6 remaining combinations have no meaning in BCD.

Design Flexibility:

- **Don't-Care Optimization**: Allowed to assign either 0 or 1 to output variables for "don't-care" combinations.
- **Simplification Goal**: Use "don't-care" conditions to optimize and simplify the circuit design.

Code Conversion Example

- The maps in Figure are plotted to obtain simplified Boolean functions for the outputs.
- Each one of the four maps represents one of the four outputs of the circuit as a function of the four input variables.
- The 1's marked inside the squares are obtained from the minterms that make the output equal to 1.



Code Conversion Example

- The 1’s are obtained from the truth table by going over the output columns one at a time.
- For example, the column under output z has five 1’s; therefore, the map for z has five 1’s, each being in a square corresponding to the minterm that makes z equal to 1.
- The six don’t-care minterms 10 through 15 are marked with an X .
- One possible way to simplify the functions into sum-of-products form is listed under the map of each variable.

Code Conversion Example

- A two-level logic diagram for each output may be obtained directly from the Boolean expressions derived from the maps.
- There are various other possibilities for a logic diagram that implements this circuit.
- The expressions obtained in Fig. 4.3 may be manipulated algebraically for the purpose of using common gates for two or more outputs.

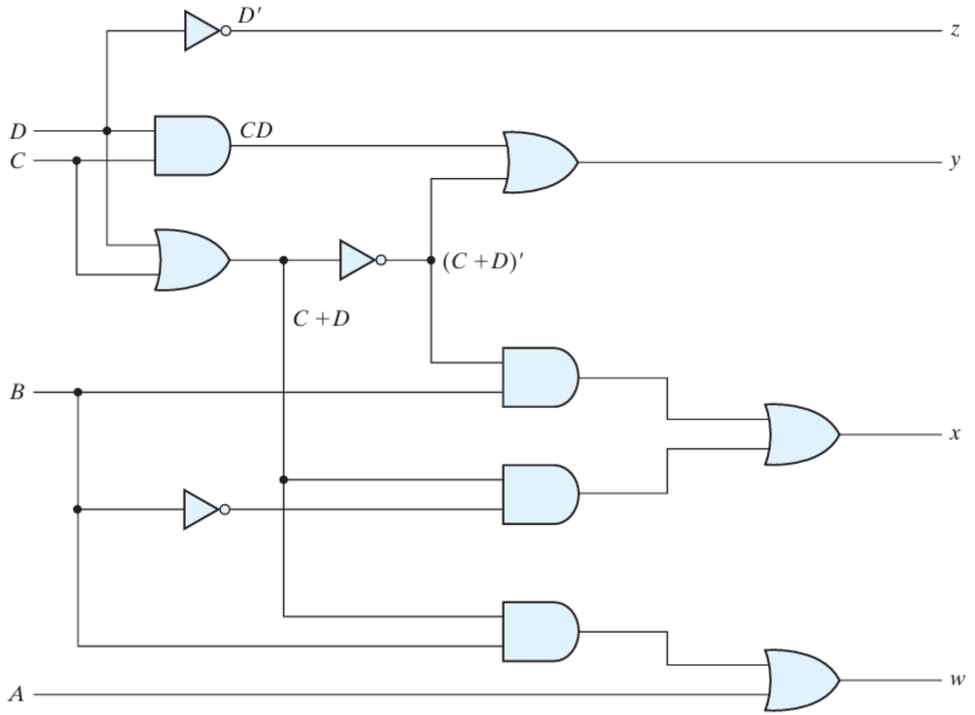
Code Conversion Example

- This manipulation, shown next, illustrates the flexibility obtained with multiple-output systems when implemented with three or more levels of gates:

$$
\begin{aligned}
z &= D' \\
y &= CD + C'D' = CD + (C + D)' \\
x &= B'C + B'D + BC'D' = B'(C + D) + BC'D' \\
&= B'(C + D) + B(C + D)' \\
w &= A + BC + BD = A + B(C + D)
\end{aligned}
$$

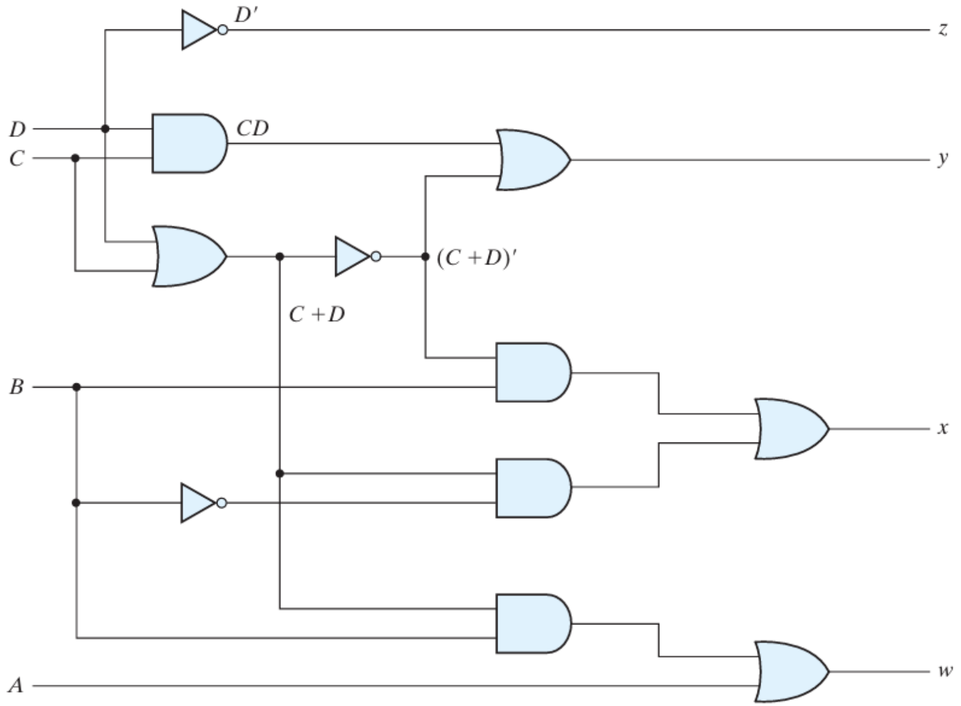
Code Conversion Example

- The logic diagram that implements these expressions is shown in Figure below.
- Note that the OR gate whose output is C+D has been used to implement partially each of three outputs.
- Not counting input inverters, the implementation in the sum-of-products form requires seven AND gates and three OR gates.



Code Conversion Example

- The implementation of below Figure requires four AND gates, four OR gates, and one inverter.
- If only the normal inputs are available, the first implementation will require inverters for variables B, C, and D, and the second implementation will require inverters for variables B and D.
- Thus, the three-level logic circuit requires fewer gates, all of which in turn require no more than two inputs.



References

- Computer Organization and Architecture Designing for Performance Tenth Edition by William Stallings
- Digital Design With an Introduction to the Verilog HDL FIFTH EDITION by M Morris, M. and Michael, D., 2013.

Ollscoil na Gaillimhe
University of Galway

Thank *you*

University *of* Galway.ie