Logic and Computer Design Fundamentals

Chapter 3 – Combinational Logic Design

Part 1 – Implementation Technology and Logic Design

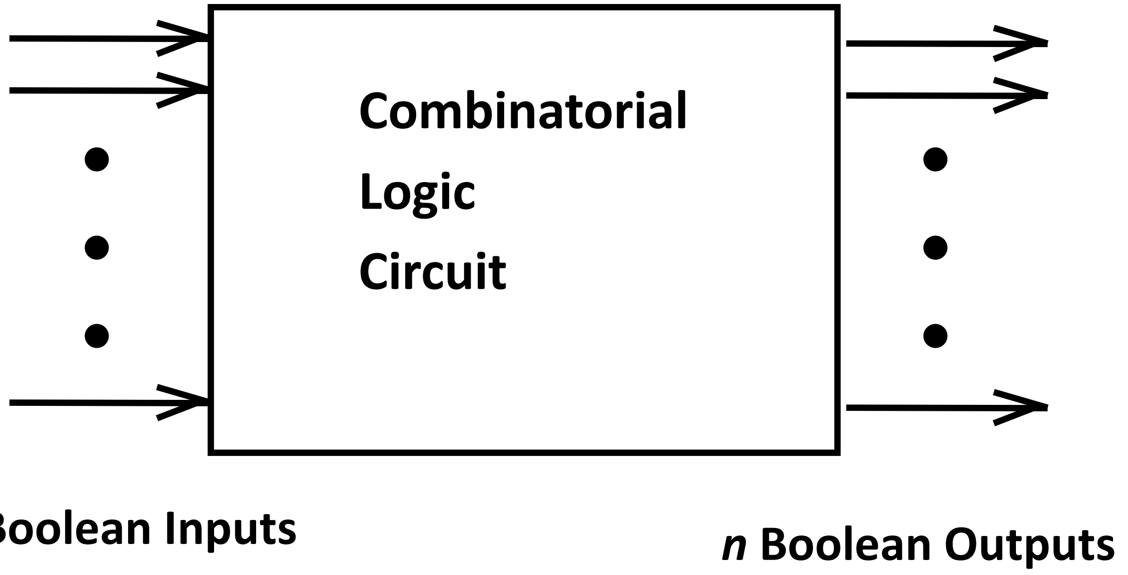
Charles Kime & Thomas Kaminski

Overview

- **Part 1 - Implementation Technology and Logic Design**
  - **Design Concepts and Automation**
    - **Fundamental concepts of design and computer-aided design techniques**
  - **The Design Space**
    - **Technology parameters for gates, positive and negative logic and design tradeoffs**
  - **Design Procedure**
    - **The major design steps: specification, formulation, optimization, technology mapping, and verification**
  - **Technology Mapping**
    - **From AND, OR, and NOT to other gate types**
  - **Verification**
    - **Does the designed circuit meet the specifications?**
- **Part 2 – Rudimentary Logic Functions, Decoding, Encoding, and Selecting**

Combinational Circuits

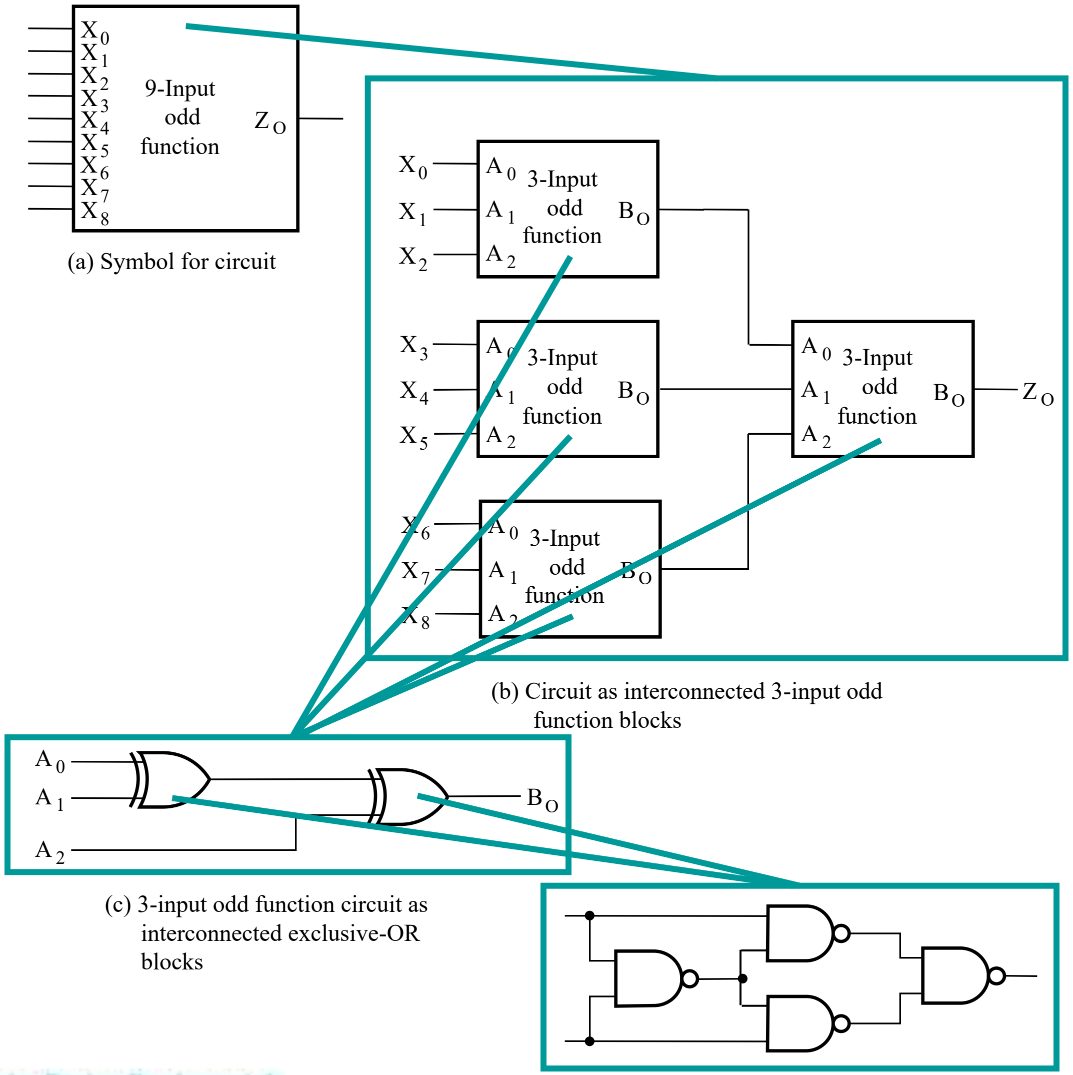
- **A combinational logic circuit has:**
  - **A set of *m* Boolean inputs,**
  - **A set of *n* Boolean outputs, and**
  - ***n* switching functions, each mapping the 2^m input combinations to an output such that the current output depends only on the current input values**
- **A block diagram:**



Hierarchical Design

- To control the complexity of the function mapping inputs to outputs:
  - Decompose the function into smaller pieces called *blocks*
  - Decompose each block's function into smaller blocks, repeating as necessary until all blocks are small enough
  - Any block not decomposed is called a *primitive block*
  - The collection of all blocks including the decomposed ones is a *hierarchy*
- Example: 9-input parity tree (see next slide)
  - Top Level: 9 inputs, one output
  - 2nd Level: Four 3-bit odd parity trees in two levels
  - 3rd Level: Two 2-bit exclusive-OR functions
  - Primitives: Four 2-input NAND gates
  - Design requires 4 X 2 X 4 = 32 2-input NAND gates

Hierarchy for Parity Tree Example



(a) Symbol for circuit

(b) Circuit as interconnected 3-input odd function blocks

(c) 3-input odd function circuit as interconnected exclusive-OR blocks

(d) Exclusive-OR block as interconnected NANDs

Reusable Functions and CAD

- Whenever possible, we try to decompose a complex design into common, *reusable* function blocks
- These blocks are
  - verified and well-documented
  - placed in libraries for future use
- Representative Computer-Aided Design Tools:
  - Schematic Capture
  - Logic Simulators
  - Timing Verifiers
  - Hardware Description Languages
    - Verilog and VHDL
  - Logic Synthesizers
  - Integrated Circuit Layout

Top-Down versus Bottom-Up

- **A *top-down design* proceeds from an abstract, high-level specification to a more and more detailed design by decomposition and successive refinement**
- **A *bottom-up design* starts with detailed primitive blocks and combines them into larger and more complex functional blocks**
- **Designs usually proceed from both directions simultaneously**
  - **Top-down design answers: What are we building?**
  - **Bottom-up design answers: How do we build it?**
- **Top-down controls complexity while bottom-up focuses on the details**

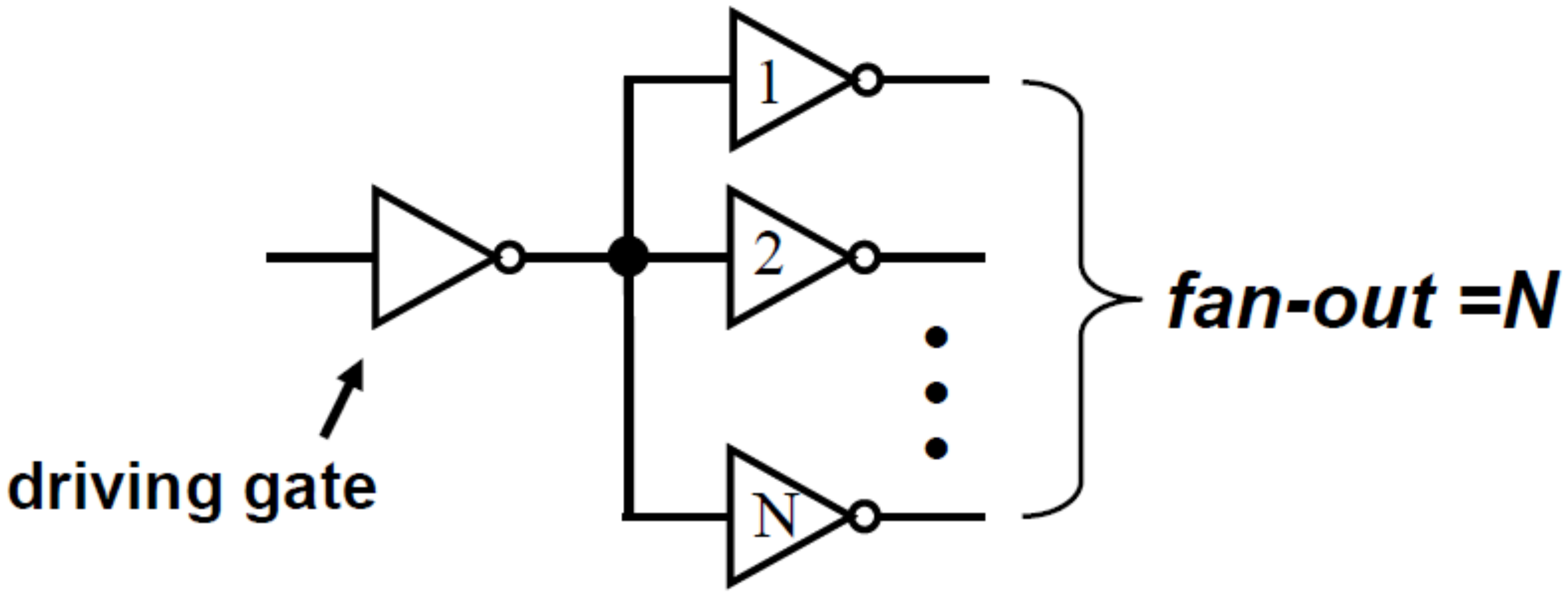
Integrated Circuits

- **Integrated circuit (informally, a "chip") is a semiconductor crystal (most often silicon) containing the electronic components for the digital gates and storage elements which are interconnected on the chip.**
- **Terminology - Levels of chip integration**
  - ***SSI* (*small-scale integrated*) - fewer than 10 gates**
  - ***MSI* (*medium-scale integrated*) - 10 to 100 gates**
  - ***LSI* (*large-scale integrated*) - 100 to thousands of gates**
  - ***VLSI* (*very large-scale integrated*) - thousands to 100s of millions of gates**

Technology Parameters

- Specific gate implementation technologies are characterized by the following parameters:
  - *Fan-in* – the number of inputs available on a gate
  - *Fan-out* – the number of gates that are connected to the output of the driving gate
  - *Logic Levels* – the signal value ranges for 1 and 0 on the inputs and 1 and 0 on the outputs (see Figure 1-1)
  - *Noise Margin* – the maximum external noise voltage superimposed on a normal input value that will not cause an undesirable change in the circuit output
  - *Cost for a gate* - a measure of the contribution by the gate to the cost of the integrated circuit
  - *Propagation Delay* – The time required for a change in the value of a signal to propagate from an input to an output
  - *Power Dissipation* – the amount of power drawn from the power supply and consumed by the gate

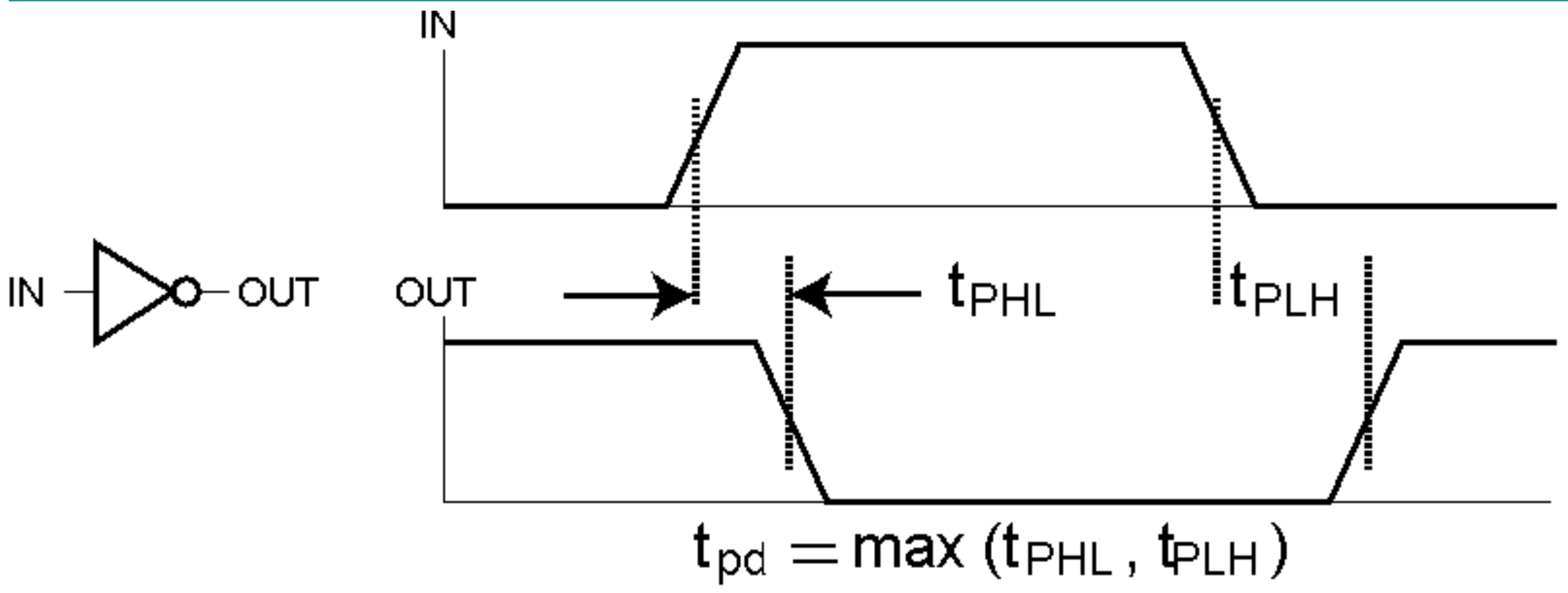
Fan-Out



Propagation Delay

- **Propagation delay is the time for a change on an input of a gate to propagate to the output.**
- **Delay is usually measured at the 50% point with respect to the H and L output voltage levels.**
- **High-to-low (t_{PHL}) and low-to-high (t_{PLH}) output signal changes may have different propagation delays.**
- **High-to-low (HL) and low-to-high (LH) transitions are defined with respect to the output, <u>not</u> the input.**
- **An HL input transition causes:**
  - **an LH output transition if the gate inverts and**
  - **an HL output transition if the gate does not invert.**

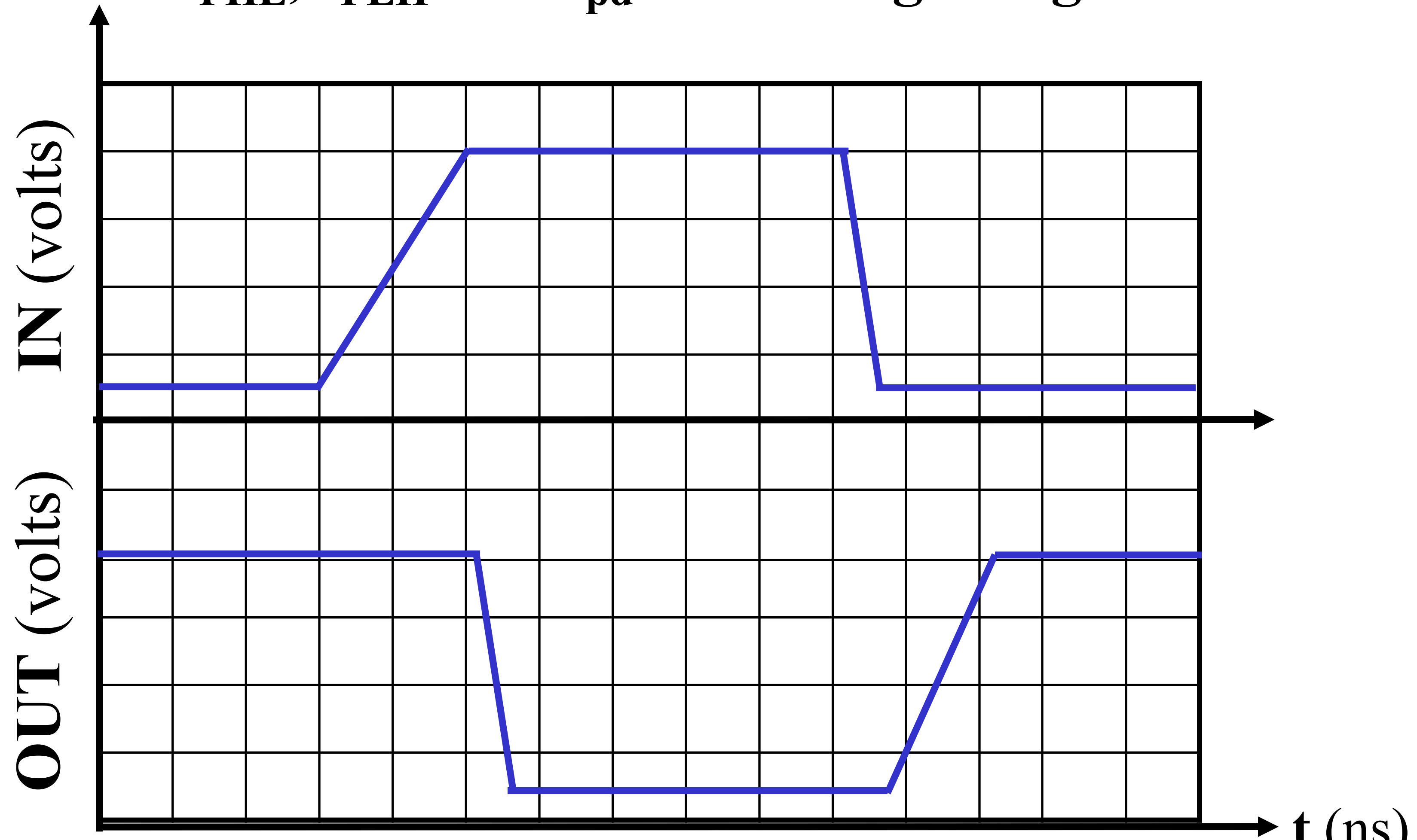
Propagation Delay (continued)



- **Propagation delays measured at the midpoint between the L and H values**

Propagation Delay Example

- Find t_{PHL}, t_{PLH} and t_{pd} for the signals given

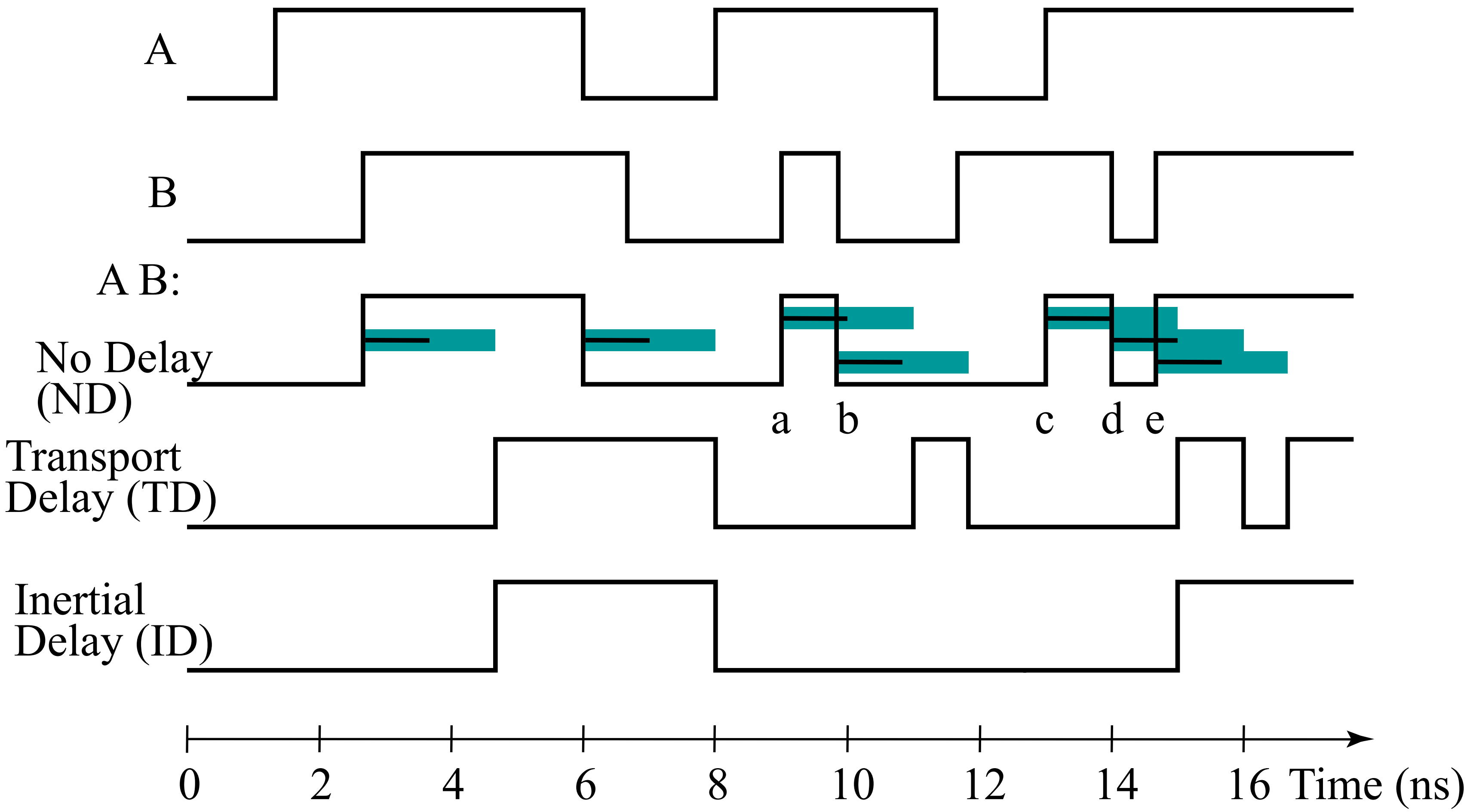


1.0 ns per division

Delay Models

- *Transport delay* - a change in the output in response to a change on the inputs occurs after a fixed specified delay
- *Inertial delay* - similar to transport delay, except that if the input changes such that the output is to change twice in a time interval less than the *rejection time*, the output changes do not occur. Models typical electronic circuit behavior, namely, rejects narrow "pulses" on the outputs

Delay Model Example



Propagation Delay = 2.0 ns Rejection Time = 1 .0 ns

Cost

- In an integrated circuit:
  - The cost of a gate is proportional to the chip area occupied by the gate
  - The gate area is roughly proportional to the number and size of the transistors and the amount of wiring connecting them
  - Ignoring the wiring area, the gate area is roughly proportional to the gate input count
  - So gate input count is a rough measure of gate cost
- If the actual chip layout area occupied by the gate is known, it is a far more accurate measure

Design Trade-Offs

- **Cost - performance tradeoffs**
- **Gate-Level Example:**
  - **NAND gate G with 20 standard loads on its output has a delay of 0.45 ns and has a normalized cost of 2.0**
  - **A buffer H has a normalized cost of 1.5. The NAND gate driving the buffer with 20 standard loads gives a total delay of 0.33 ns**
  - **In which of the following cases should the buffer be added?**
    1. **The cost of this portion of the circuit cannot be more than 2.5**
    2. **The delay of this portion of the circuit cannot be more than 0.40 ns**
    3. **The delay of this portion of the circuit must be less than 0.40 ns and the cost less than 3.0**
- **Tradeoffs can also be accomplished much higher in the design hierarchy**
- **Constraints on cost and performance have a major role in making tradeoffs**

Design Procedure

1. **Specification**
   - Write a specification for the circuit if one is not already available
2. **Formulation**
   - Derive a truth table or initial Boolean equations that define the required relationships between the inputs and outputs, if not in the specification
3. **Optimization**
   - Apply 2-level and multiple-level optimization
   - Draw a logic diagram or provide a netlist for the resulting circuit using ANDs, ORs, and inverters

Design Procedure

4. Technology Mapping
   - Map the logic diagram or netlist to the implementation technology selected
5. Verification
   - Verify the correctness of the final design

Design Example

1. Specification
   - <u>BCD to Excess-3 code converter</u>
   - Transforms BCD code for the decimal digits to Excess-3 code for the decimal digits
   - BCD code words for digits 0 through 9: 4-bit patterns 0000 to 1001, respectively
   - Excess-3 code words for digits 0 through 9: 4-bit patterns consisting of 3 (binary 0011) added to each BCD code word
   - Implementation:
     - multiple-level circuit
     - NAND gates (including inverters)

Design Example (continued)

2. Formulation

- Conversion of 4-bit codes can be most easily formulated by a truth table
- Variables
  - BCD: A,B,C,D
- Variables
  - Excess-3 W,X,Y,Z
- Don't Cares
  - BCD 1010 to 1111

Input BCD A B C D	Output Excess-3 WXYZ
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

Design Example (continued)

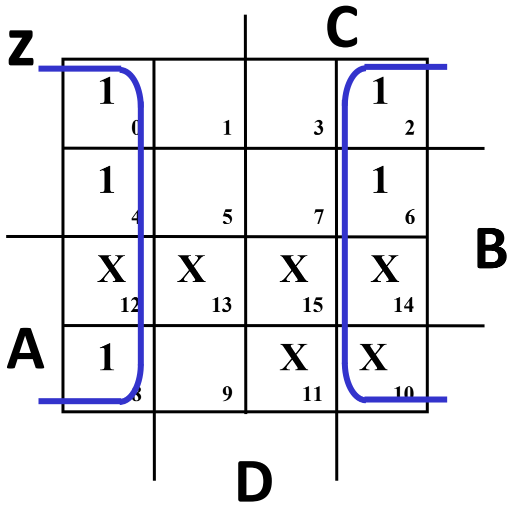
3. **Optimization**
   a. **2-level using K-maps**

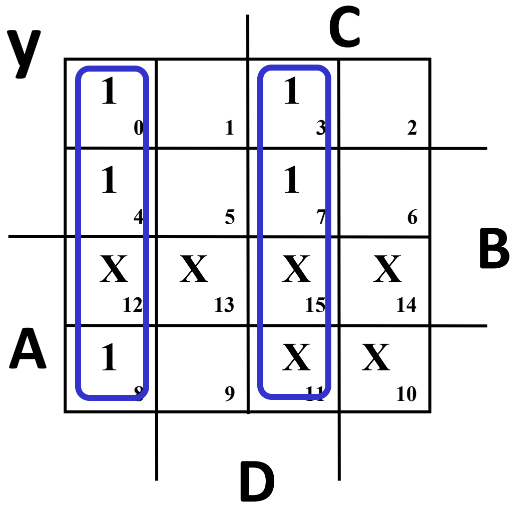
$W = A + BC + BD$

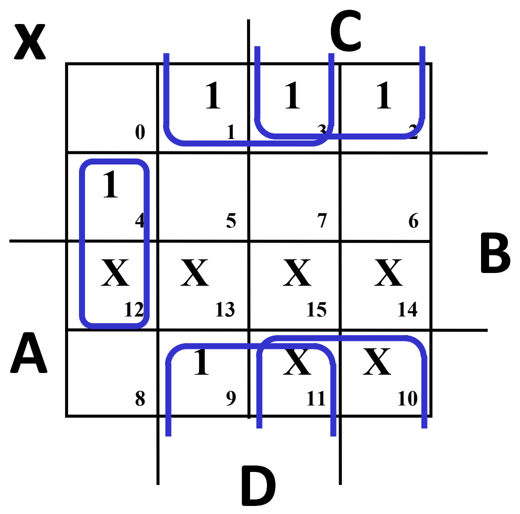
$X = \bar{B}C + \bar{B}D + B\bar{C}\bar{D}$

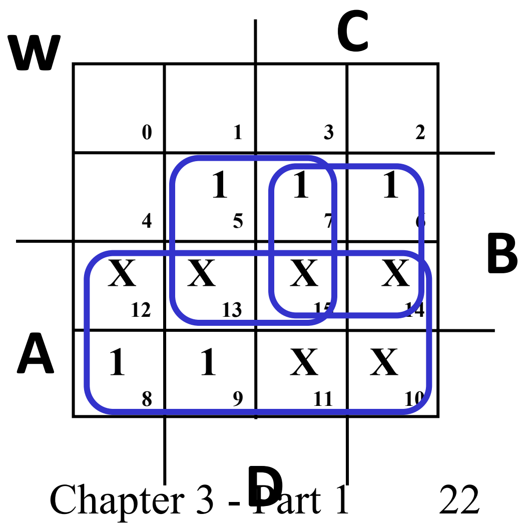
$Y = CD + \bar{C}\bar{D}$

$Z = \bar{D}$









Design Example (continued)

4. Technology Mapping

- Mapping with a library containing inverters and 2-input NAND, 2-input NOR, and 2-2 AOI gates