Logic and Computer Design Fundamentals

Integrated Circuits

Charles Kime & Thomas Kaminski

(Hyperlinks are active in View Show mode)

Overview

- Part 1 – The Design Space
  - Integrated Circuits
    - Levels of Integration
    - Technology Parameters
- Part 2 – Propagation Delay and Timing
- Part 3 - Programmable Implementation Technologies

Integrated Circuits

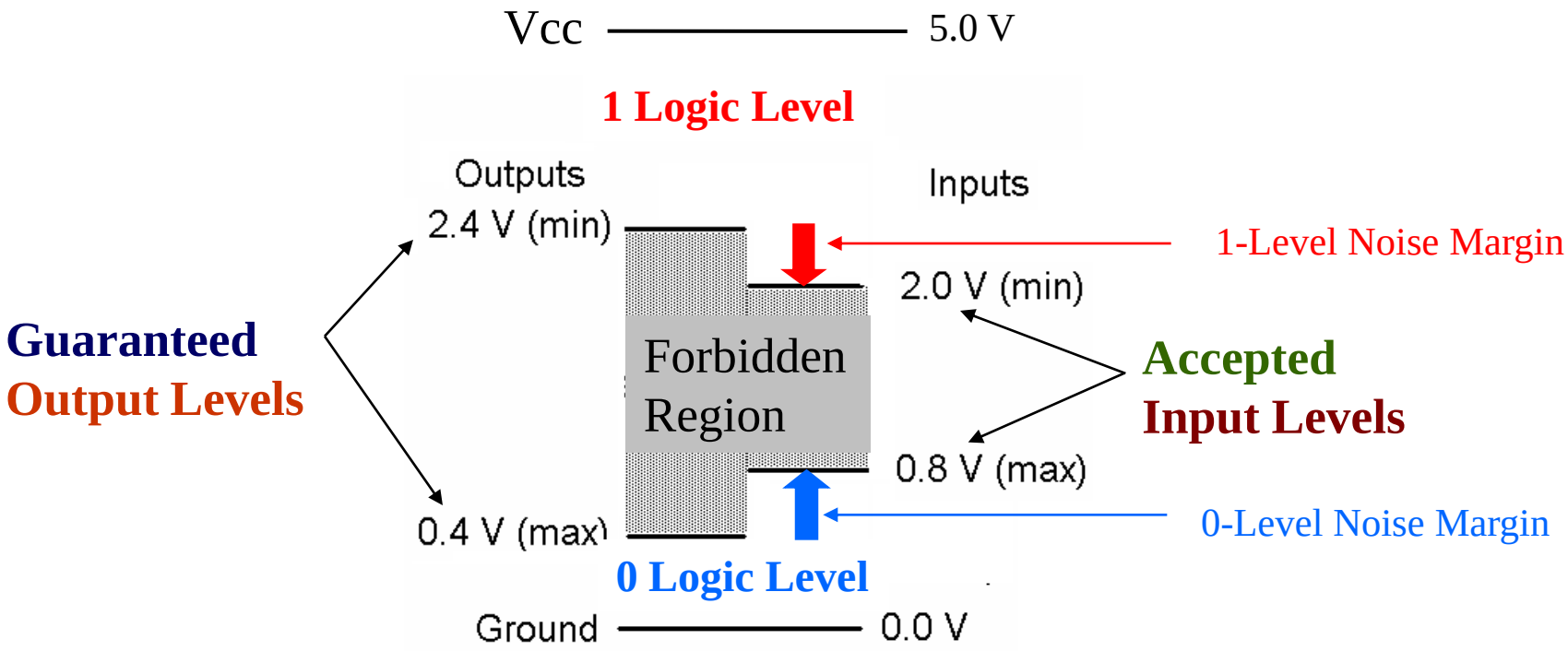
- **Integrated circuit (informally, a "chip") is a semiconductor crystal (most often silicon) containing the electronic components for the digital gates and storage elements which are interconnected on the chip.**
- **Terminology - Levels of chip integration**
  - ***SSI* (*small-scale integrated*) - 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

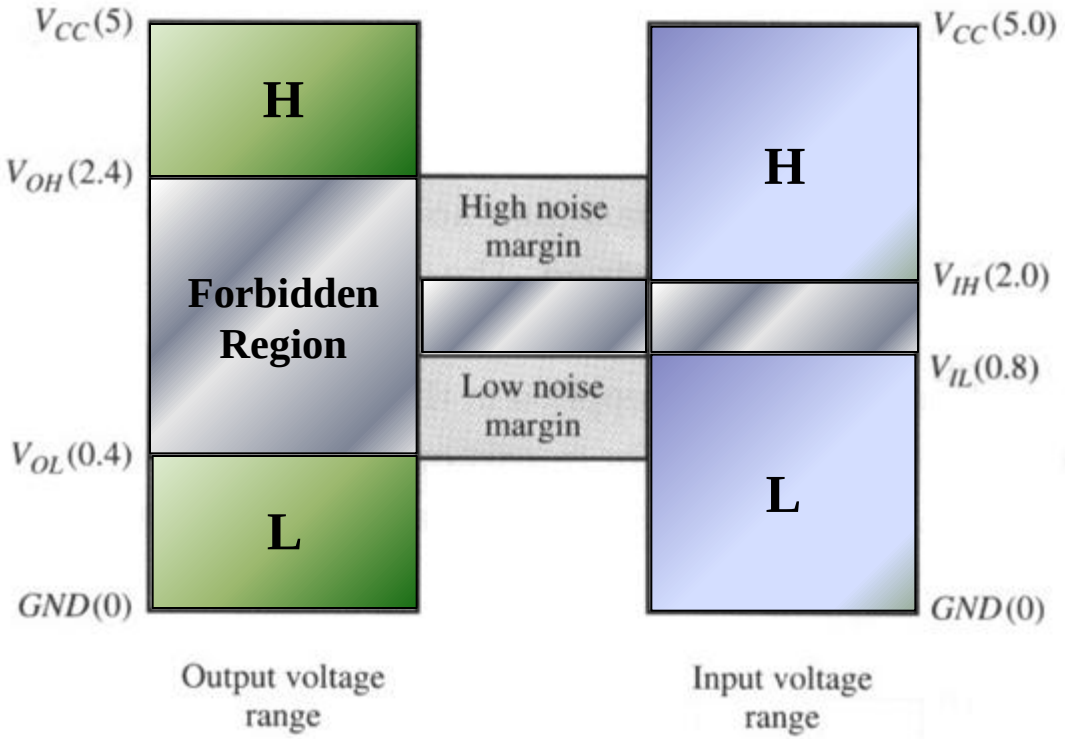
- **Specific gate implementation technologies are characterized by the following parameters:**
  - ***Fan-in*** **– the number of inputs available on a gate**
  - ***Fan-out*** **– the number of standard loads driven by a gate output**
  - ***Logic Levels*** **– the signal value ranges for 1 and 0 on the inputs and 1 and 0 on the outputs**
  - ***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**

Logic Levels & Noise Margin

Standard TTL Output and Inputs Voltage Levels



Logic Levels & Noise Margin



High and low noise margins

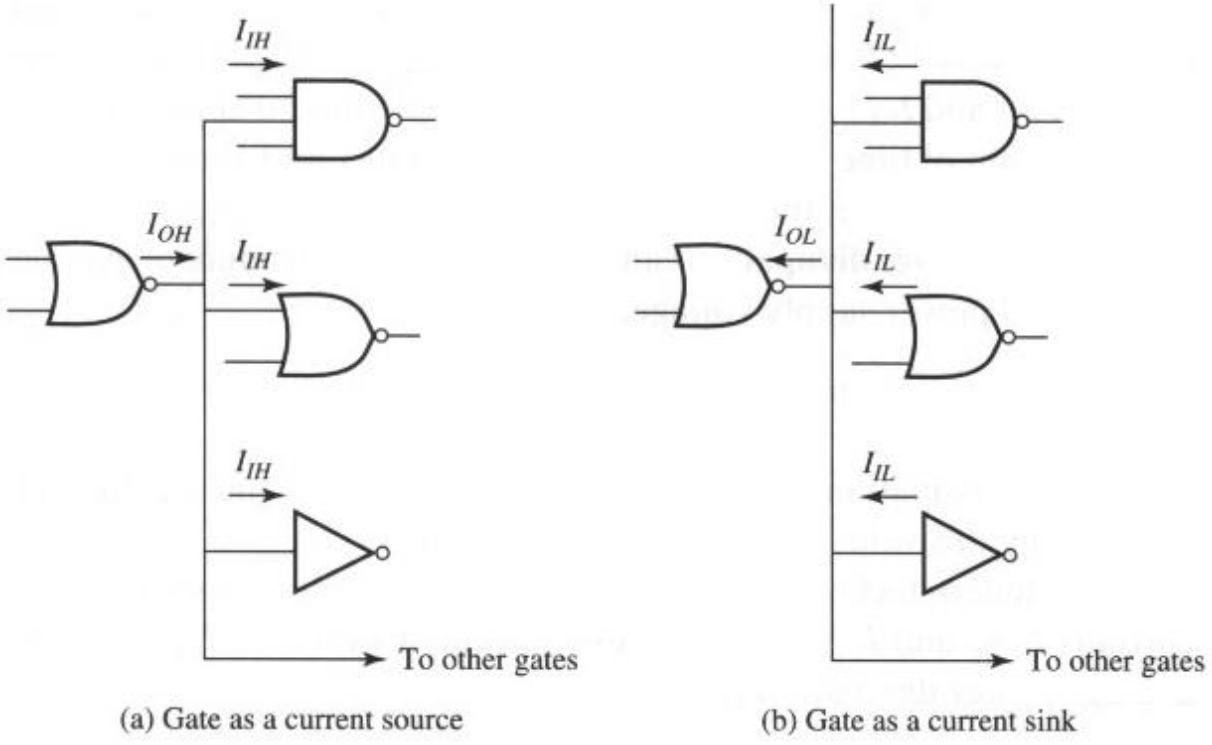
High-Level Noise Margin = V_{OH} - V_{IH}

Low-Level Noise margin = V_{IL} - V_{OL}

Fan-out

- **Fan-out can be defined in terms of a standard load**
  - **Example: 1 standard load equals the load contributed by the input of 1 inverter.**

Fan-out



(a) Gate as a current source

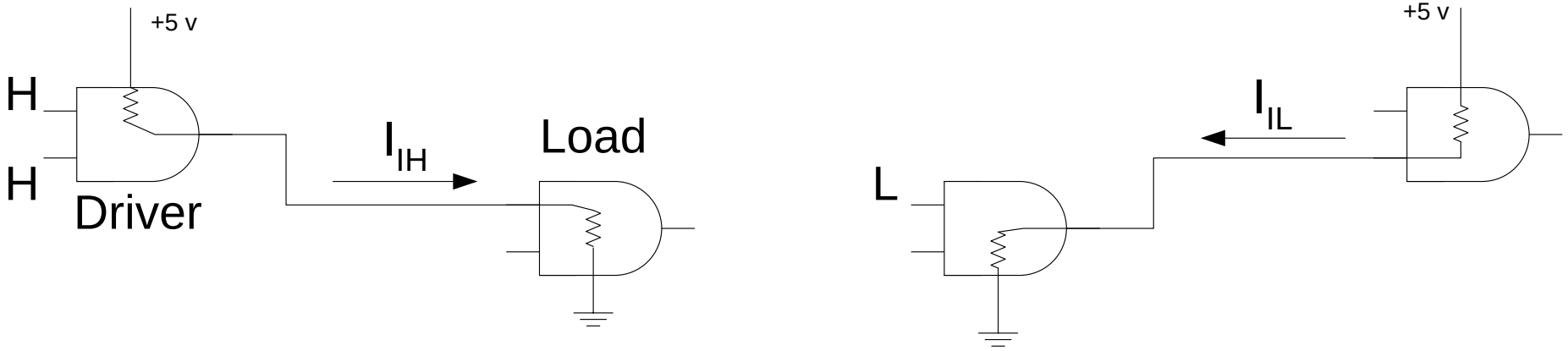
(b) Gate as a current sink

Current flow in a typical logic circuit

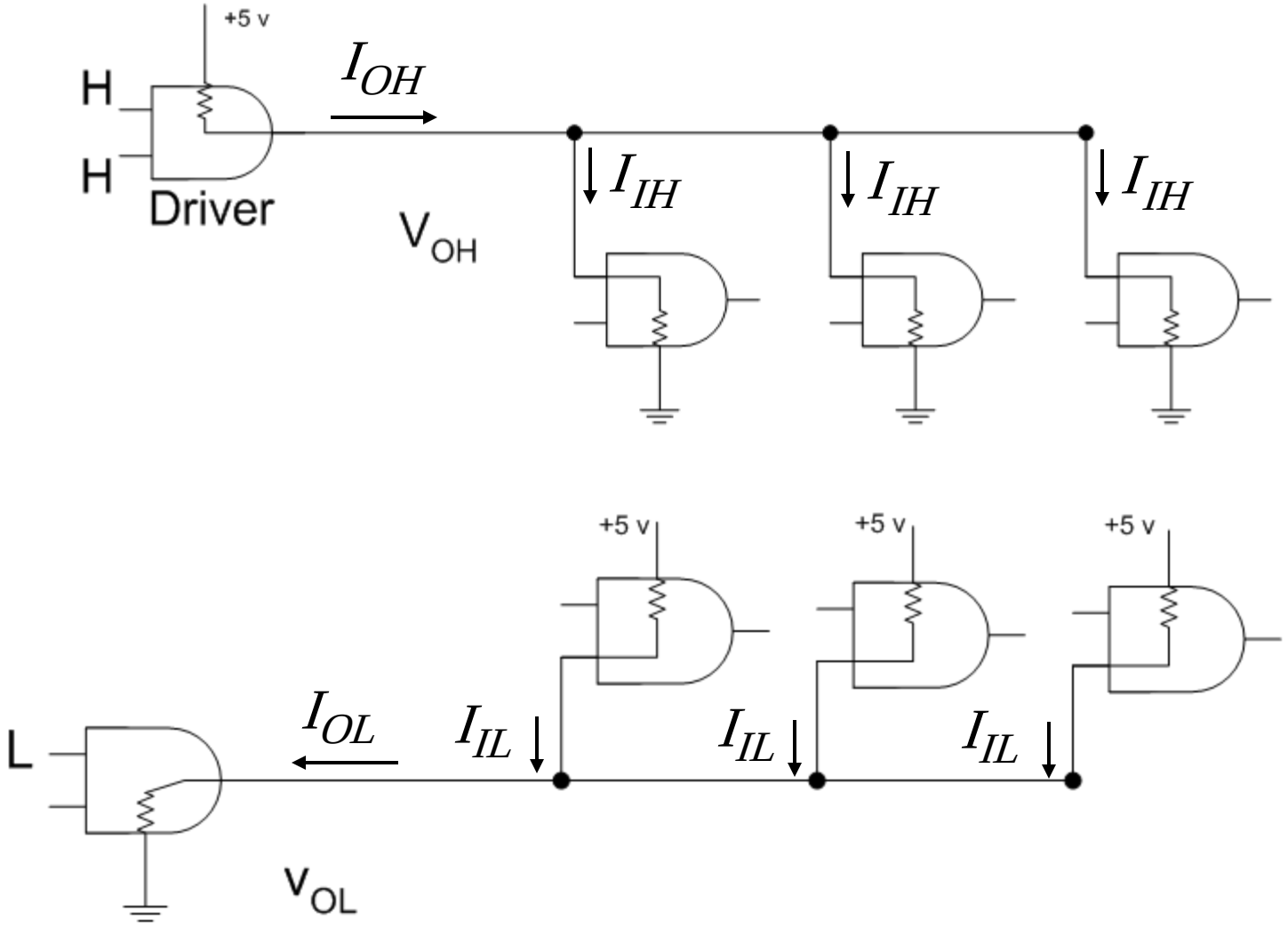
Fan-out = min (I_{OH}/ I_{IH} , I_{OL}/I_{IL})

Fan-Out for TTL Loads

- **Fan Out = $I_{OH}/I_{IH} = I_{OL}/I_{IL}$**
- **= 400 µA/40 µA = 16 mA/1.6 mA = 10**

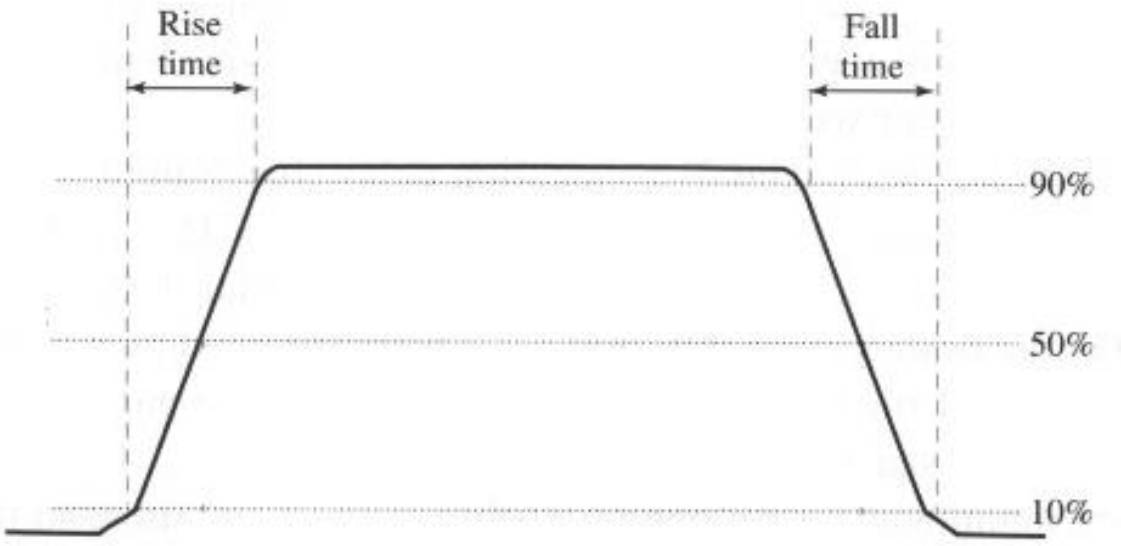


Fan-Out for TTL Loads



Transition Time

- ***Transition time*** **-the time required for a signal to change from H to L, t_{HL}, or from L to H, t_{LH}**



Rise Time : a signal to switch from 10% to 90%

Fall Time : a signal to switch from 90% to 10%

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**

Power Dissipation

In TTL :

Gate Output High (I_{CCH})
Gate Output Low (I_{CCL})

Average Power Dissipated:

$$P_{AVG} = V_{cc}\, I_{cc}$$
$$P_{AVG} = V_{cc}\, (I_{CCH} + I_{CCL})/2$$

Part 2
Propagation Delay and Timing

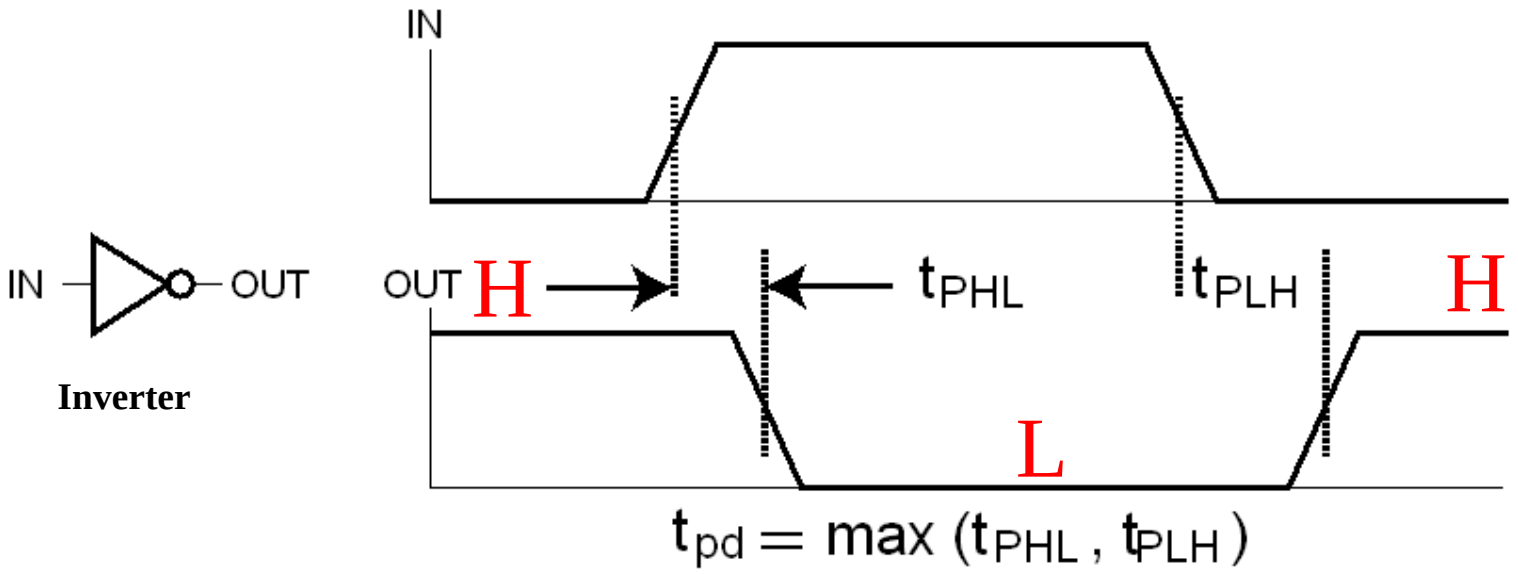
Overview

- **Part 1 – The Design Space**
- **Part 2 – Propagation Delay and Timing**
  - **Propagation Delay**
  - **Delay Models**
  - **Cost/Performance Tradeoffs**
  - **Flip-Flop Timing**
  - **Circuit & System Level Timing**
- **Part 3 - Programmable Implementation Technologies**

Propagation Delay

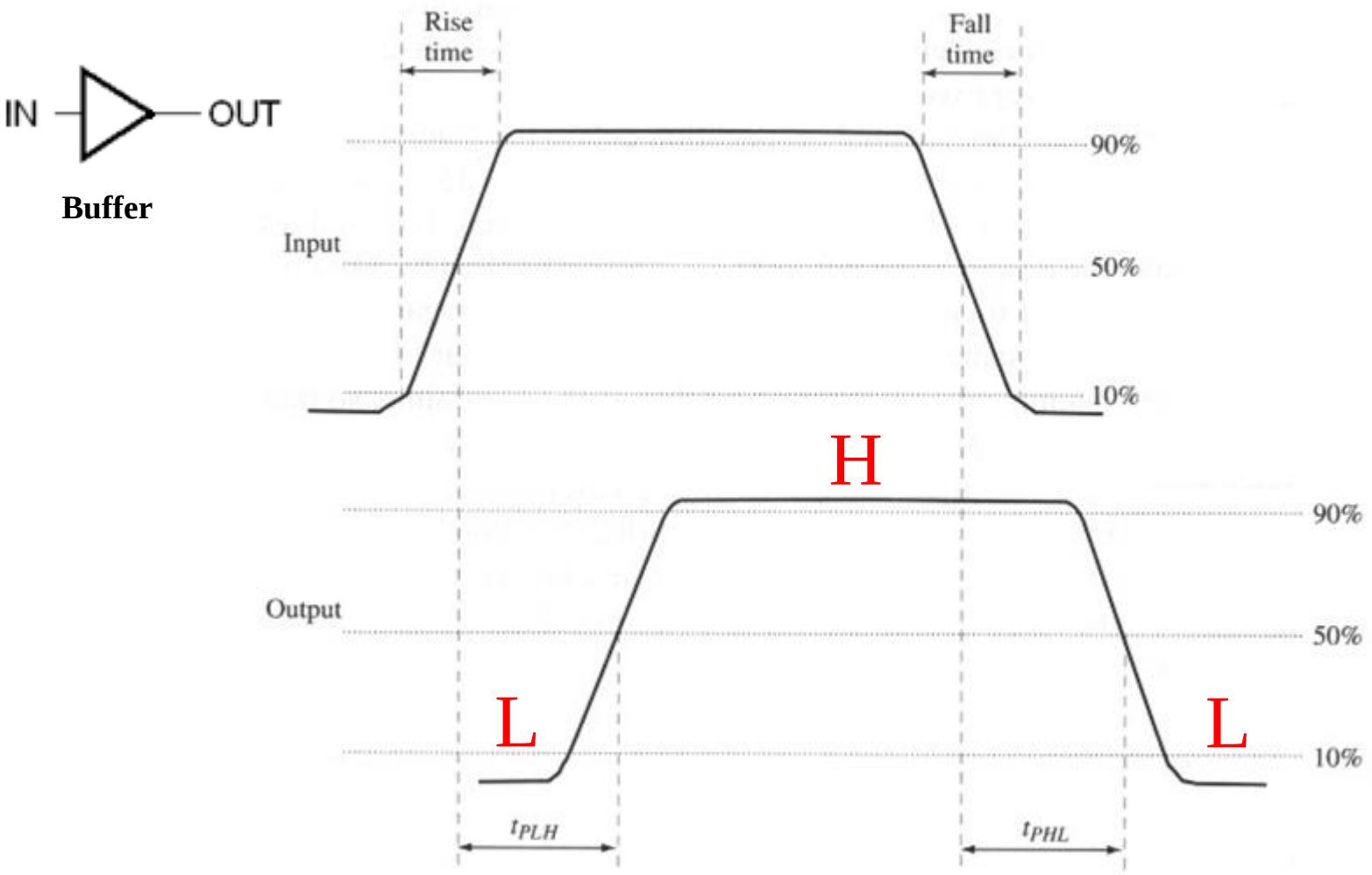
- **Propagation delay is the time for a change on an input of a gate to propagate to the output.**
- **Delay is usually measured at the 50% point with respect to the H and L output voltage levels.**
- **High-to-low (t_{PHL}) and low-to-high (t_{PLH}) output signal changes may have different propagation delays.**
- **High-to-low (HL) and low-to-high (LH) transitions are defined with respect to the output, not 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.**
- **t_{PLH} = The low-to-high propagation time.**
- **t_{PHL} = The high-to-low propagation time.**
- **t_{pd} = Propagation delay = Max(t_{PLH} , t_{PHL})**

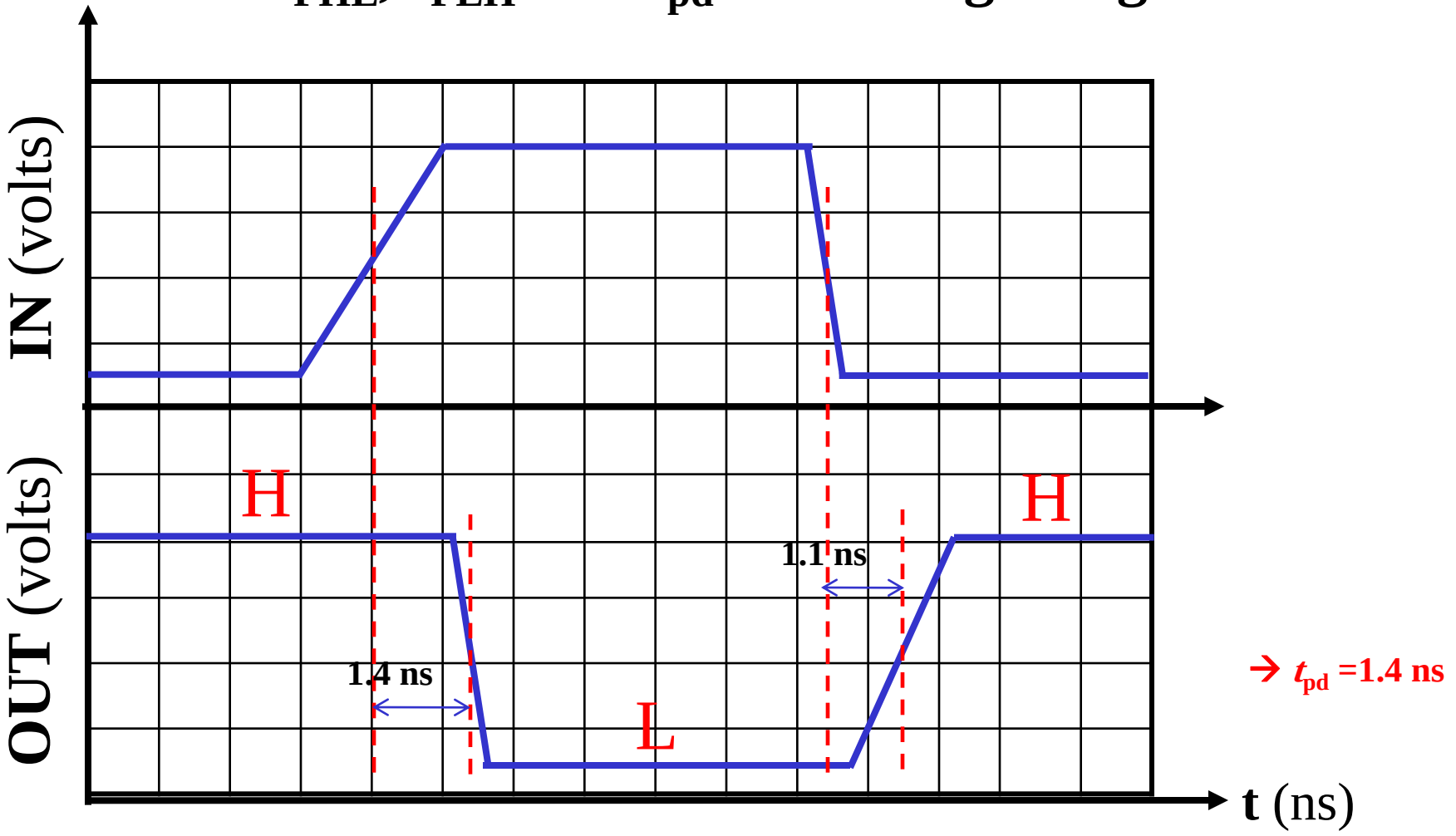
Propagation Delay



Propagation Delay: $t_P = \max(t_{PLH}, t_{PHL})$

Propagation Delay Example

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



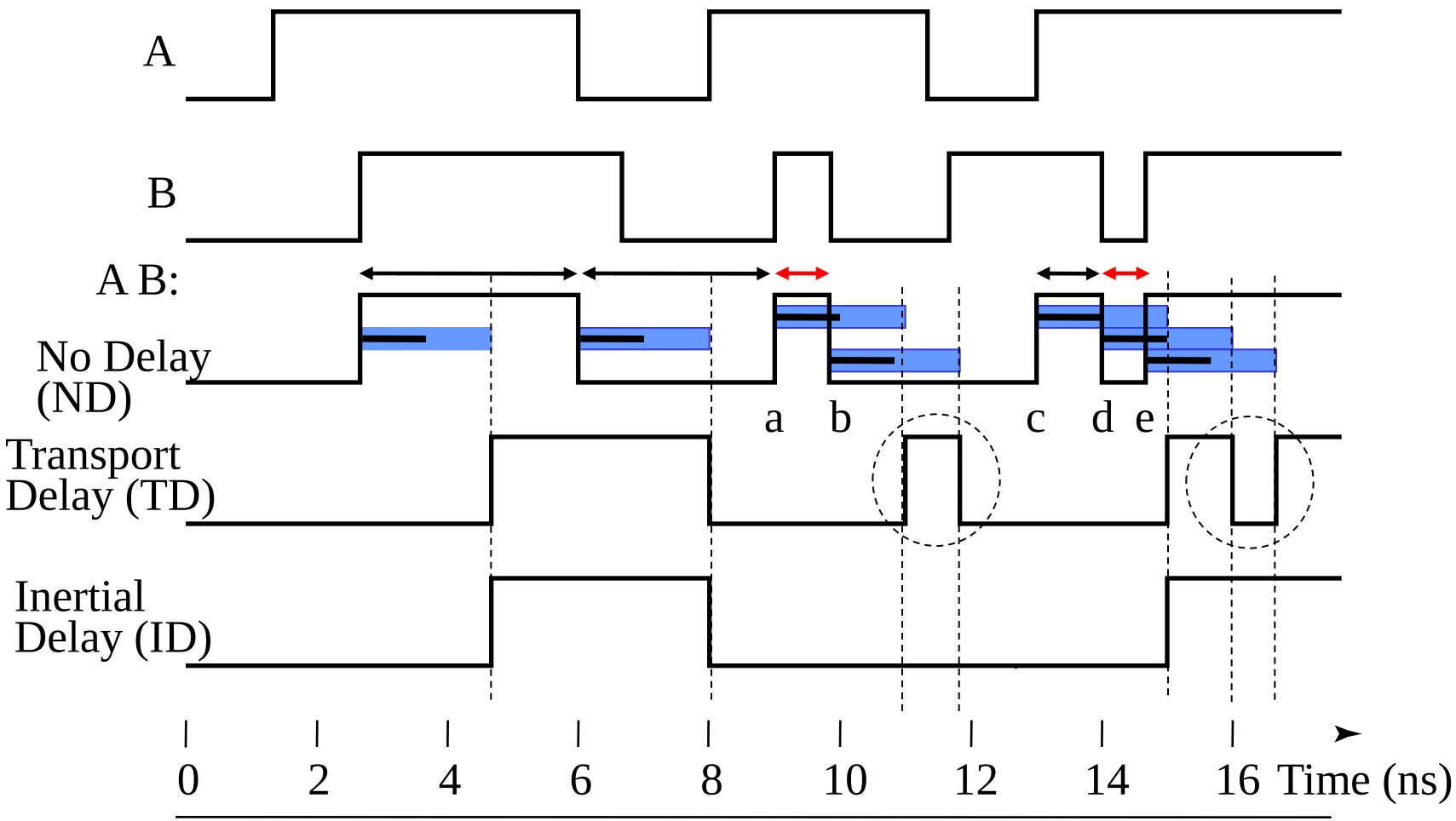
Delay Models

- **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 – AND Gate



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

Delay Models

- To determine whether an edge appears in the ID output, it must be determined whether the second edge occurs in the ND output before the end of the rejection time, and whether the edge will result in a change in the ID output.
- Since edge b occurs before the end of the rejection time for edge a in the ND output, edge a does not appear in the ID output.
- Since edge b does not change the state of ID, it is ignored.
- Since edge d occurs at the rejection time after edge c in the ND output, edge c appears.
- Edge e occurs within the rejection time after edge d, so edge d does not appear.
- Since edge c appeared and edge d did not appear, edge e does not cause a change.

Fan-out and Delay

- The fan-out loading a gate's output affects the gate's propagation delay

- Example 1:
  - One realistic equation for t_{pd} for a NAND gate with 4 inputs is:

    $$t_{pd} = 0.07 + 0.021 \text{ SL ns}$$

  - SL is the number of standard loads the gate is driving, i. e., its fan-out in standard loads
  - For SL = 4.5, $t_{pd} = 0.1645$ ns

- If this effect is considered, the delay of a gate in a circuit takes on different values depending on the circuit load on its output.

Fan-out and Delay

- **Example 2:**
  - **One realistic equation for t_{pd} for a NAND gate with 4 inputs is:**

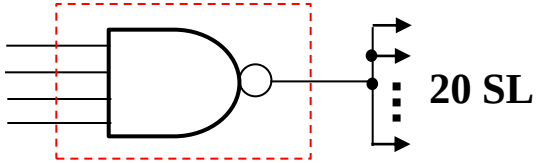
    $$t_{pd} = 0.07 + 0.021 \text{ SL ns}$$

  - **4-input NAND output is attached to the inputs of the following gates with the given number of standard loads representing the inputs:**
    - **4-input NOR gate – 0.80 standard load**
    - **3-input NAND gate – 1.00 standard load, and**
    - **Inverter – 1.00 standard load.**

  - $t_{pd} = 0.07 + 0.021\ (0.80 + 1.00 + 1.00) = 0.1288$ ns

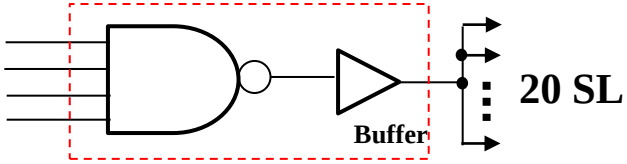
Cost/Performance Tradeoffs

- **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**



Delay	0.45 ns
Cost	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**



Delay	0.33 ns
Cost	2.0+1.5 = 3.5

  - **In which if the following cases should the buffer be added?**

1. **The cost of this portion of the circuit cannot be more than 2.5**

   → **No Buffer**

2. **The delay of this portion of the circuit cannot be more than 0.40 ns**

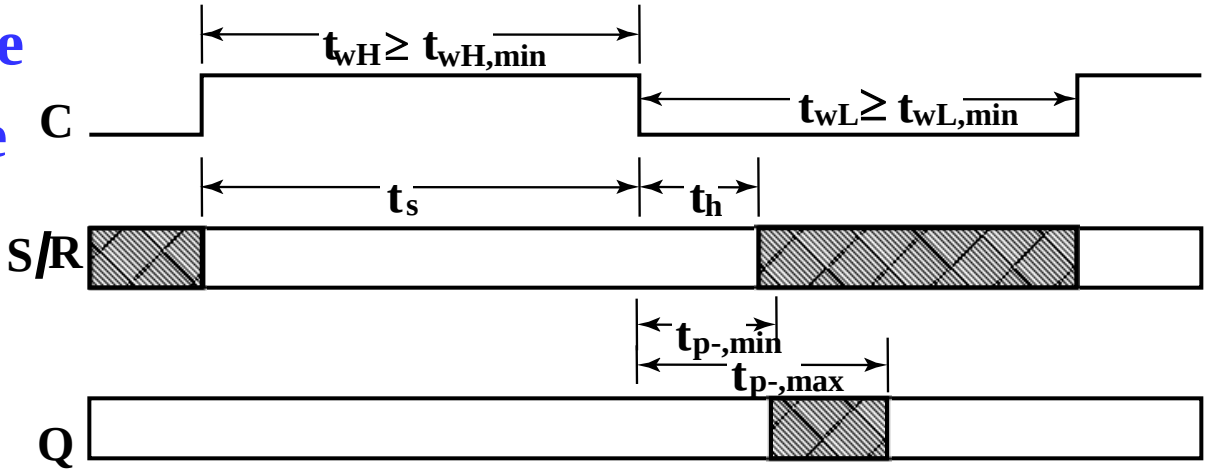
   → **Use Buffer**

3. **The delay of this portion of the circuit must be less than 0.35 ns and the cost less than 3.0**

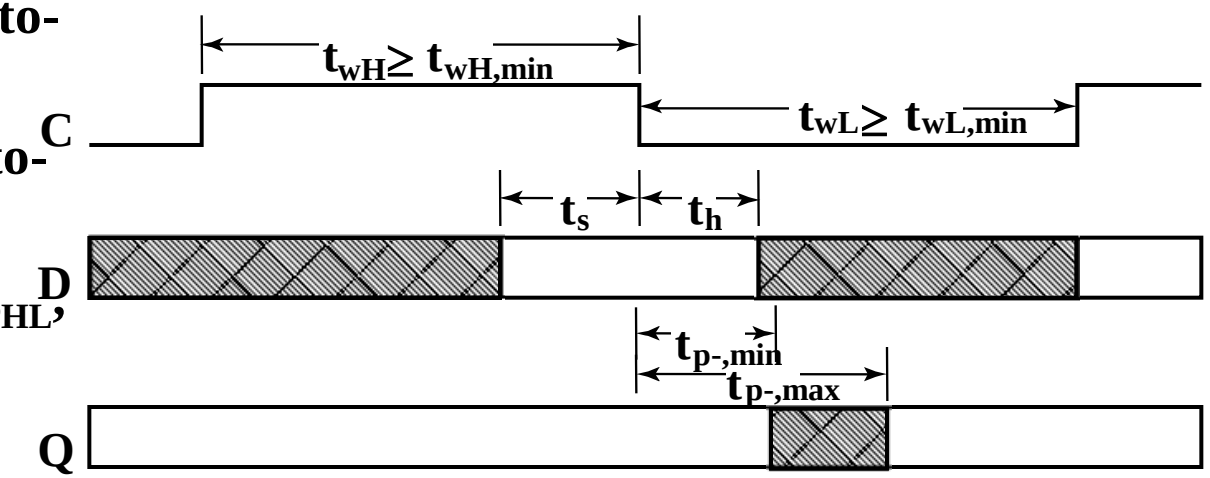
   → **Irrelevant - Buffer is needed to satisfy delay constraint, but cannot satisfy cost constraint.**

Flip-Flop Timing Parameters

- t_s - setup time
- t_h - hold time
- t_w - clock pulse width
- t_{px} - propagation delay
  - t_{PHL} - High-to-Low
  - t_{PLH} - Low-to-High
  - t_{pd} - max (t_{PHL}, t_{PLH})



(a) Pulse-triggered (positive pulse)



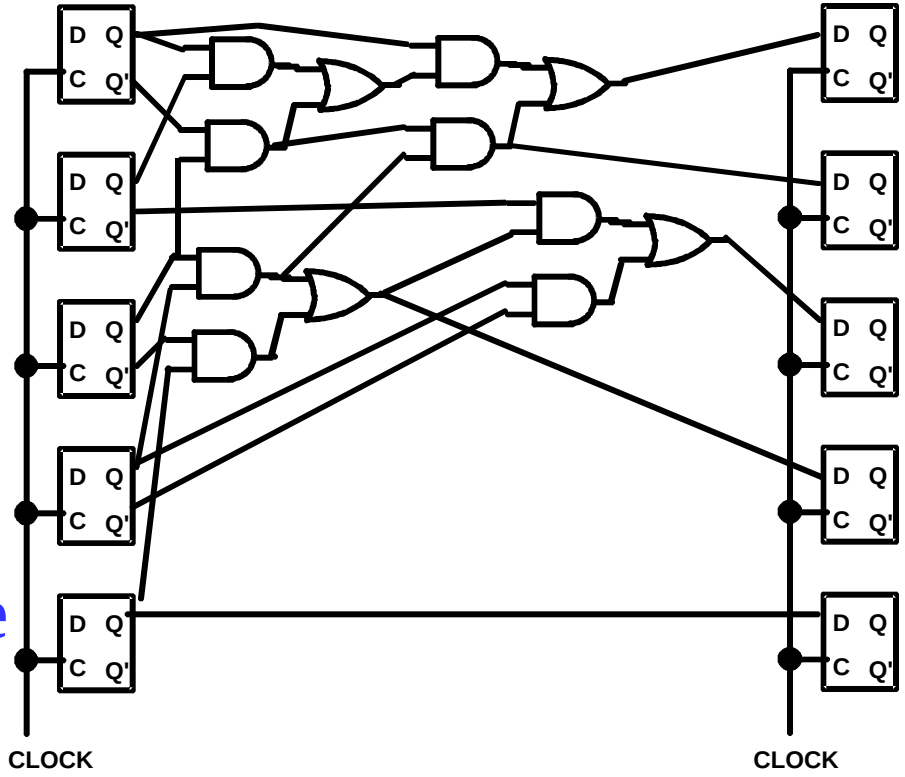
(b) Edge-triggered (negative edge)

Flip-Flop Timing Parameters

- t_s - setup time
  - Master-slave - Equal to the width of the triggering pulse (Master)
  - Edge-triggered - Equal to a time interval that is generally much less than the width of the triggering pulse
- t_h - hold time - Often equal to zero
- t_{px} - propagation delay
  - Same parameters as for gates except
  - Measured from clock edge that triggers the output change to the output change

Circuit and System Level Timing

- Consider a system comprised of ranks of flip-flops connected by logic:
- If the clock period is too short, some data changes will not propagate through the circuit to flip-flop inputs before the setup time interval begins

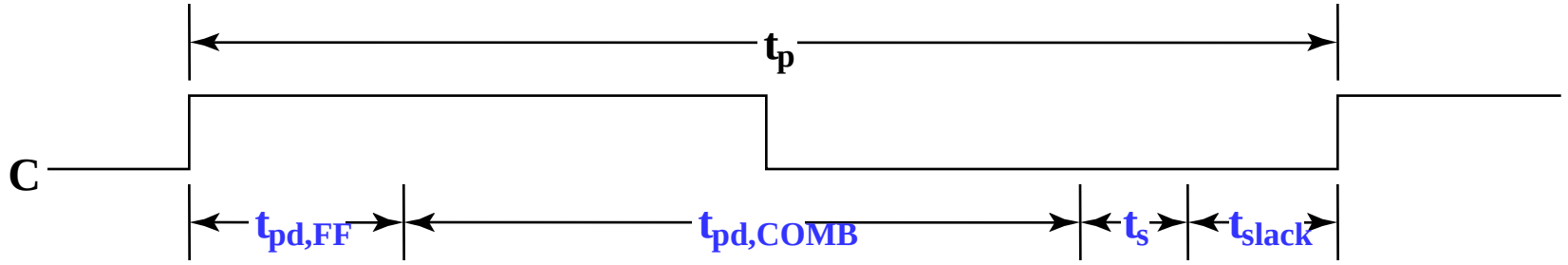


Circuit and System Level Timing

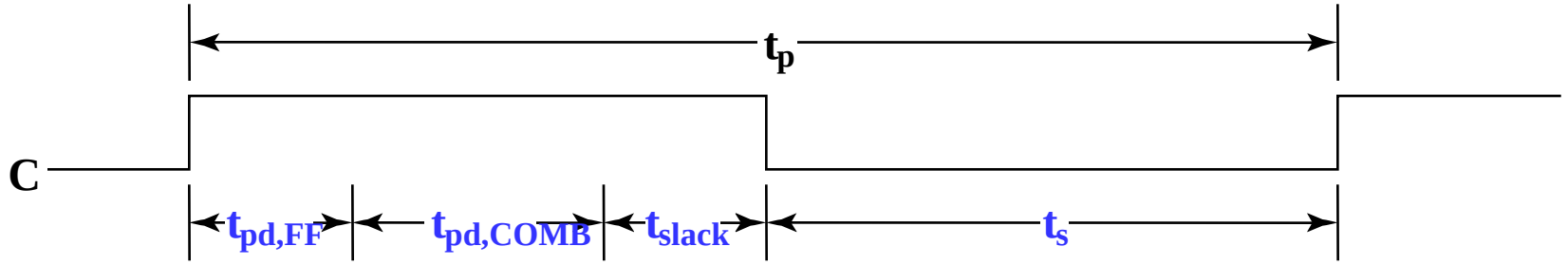
- **New Timing Components**
  - **t_p - clock period - The interval between occurrences of a specific clock edge in a periodic clock**
  - **$t_{pd,COMB}$ - total delay of combinational logic along the path from flip-flop output to flip-flop input**
  - **t_{slack} - extra time allowed in the clock period in beyond that required by the path**
    - **Must be greater than or equal to zero on all paths for correct operation**

Circuit and System Level Timing

- **Timing components along a path from flip-flop to flip-flop**



(a) Edge-triggered (positive edge)



(b) Pulse-triggered (negative pulse)

Circuit and System Level Timing

- **Timing Equations**

  $$t_p = t_{slack} + (t_{pd,FF} + t_{pd,COMB} + t_s)$$

  - **For t_{slack} greater than or equal to zero,**

    $$t_p \geq \max (t_{pd,FF} + t_{pd,COMB} + t_s)$$

    for all paths from flip-flop output to flip-flop input

- **Can be calculated more precisely by using t_{PHL} and t_{PLH} values instead of t_{pd} values, but requires consideration of inversions on paths**

Example 1: Calculation of Allowable $t_{pd,COMB}$

- **Compare the allowable combinational delay for a specific circuit:**
  - **a) Using edge-triggered flip-flops**
  - **b) Using master-slave flip-flops**

- **Parameters**
  - $t_{pd,FF}(max) = 1.0$ ns
  - $t_s(max) = 0.3$ ns for edge-triggered flip-flops
  - $t_s = t_{wH} = 1.0$ ns for master-slave flip-flops
  - Clock frequency = 250 MHz

Example 1: Calculation of Allowable $t_{pd,COMB}$

- **Calculations: t_p = 1/clock frequency = 4.0 ns**
  - **Edge-triggered: $4.0 \geq 1.0 + t_{pd,COMB} + 0.3$, $t_{pd,COMB} \leq 2.7$ ns**
  - **Master-slave: $4.0 \geq 1.0 + t_{pd,COMB} + 1.0$, $t_{pd,COMB} \leq 2.0$ ns**

- **Comparison: Suppose that for a gate, average t_{pd} = 0.3 ns**
  - **Edge-triggered: Approximately 9 gates allowed on a path**
  - **Master-slave: Approximately 6 to 7 gates allowed on a path**

Example 2: Clock Period & Frequency Calculations

- **Parameters**
  - $t_{pd,FF}(max) = 0.2$ ns
  - $t_s(max) = 0.1$ ns for edge-triggered flip-flops
  - $t_{pd,COMB}(max) = 1.3$ ns
  - $t_p = 1.5$ ns
- **Solution**
  - $t_p = t_{slack} + (t_{pd,FF} + t_{pd,COMB} + t_s)$
  - → $1.5 = t_{slack} + (0.2 + 1.3 + 0.1)$
  - → $t_{slack} = -0.1$ ns
  - t_{slack} has to be greater than or equal to zero for the longest path
  - → t_p is too small
  - → $t_p \geq t_{p,\,min} = 1.6$ ns
  - $f_{max} = 1 / 1.6$ ns $= 625$ MHz

Part 3
Programmable Implementation Technologies

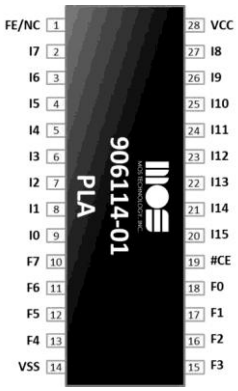
Overview

- **Part 1 – The Design Space**
- **Part 2 – Propagation Delay and Timing**
- **Part 3 - Programmable Implementation Technologies**
  - **Why Programmable Logic?**
  - **Programming Technologies**
  - **Read-Only Memories (ROMs)**
  - **Programmable Logic Arrays (PLAs)**
  - **Programmable Array Logic (PALs)**

Why Programmable Logic?

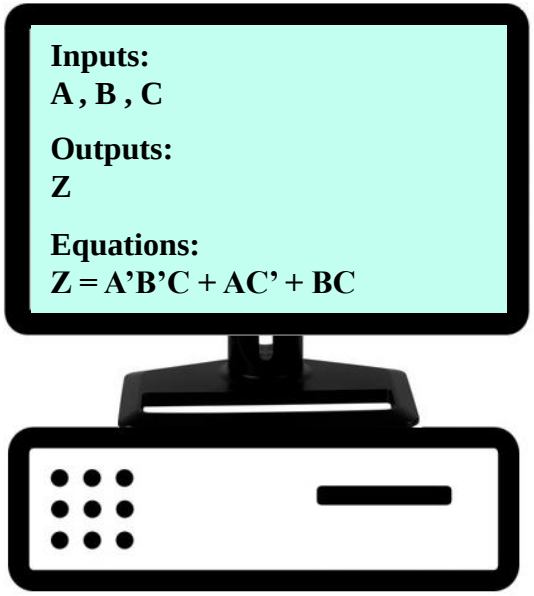
- **Facts:**
  - **It is most economical to produce an IC in large volumes**
  - **Many designs required only small volumes of ICs**
- **Need an IC that can be:**
  - **Produced in large volumes**
  - **Handle many designs required in small volumes**
- **A programmable logic part can be:**
  - **made in large volumes**
  - **programmed to implement large numbers of different low-volume designs**

Programmable Logic



1

Universal Programmer



2

Programmable Logic - More Advantages

- **Many programmable logic devices are *field-programmable*, i. e., can be programmed outside of the manufacturing environment**
- **Most programmable logic devices are *erasable* and *reprogrammable*.**
  - **Allows "updating" a device or correction of errors**
  - **Allows reuse the device for a different design - the ultimate in re-usability!**
  - **Ideal for course laboratories**
- **Programmable logic devices can be used to prototype design that will be implemented for sale in regular ICs.**
  - **Complete Intel Pentium designs were actually prototyped with specialized systems based on large numbers of VLSI programmable devices!**

Programming Technologies

- **Programming technologies:**

1. **Fuse:**
   - **Fuses which are initially CLOSED are selectively "blown out" by a higher than normal voltage to establish OPEN connections.**
   - **The pattern of OPEN and CLOSED fuses defines the logic.**
   - **Permanent: The devices cannot be reprogrammed because irreversible physical changes have occurred as a result of device programming .**

2. **Antifuse:**
   - **Antifuses contain a material that is initially OPEN.**
   - **Antifuses are selectively CLOSED by applying a higher than normal voltage.**
   - **The pattern of OPEN and CLOSED fuses defines the logic.**
   - **Permanent: The devices cannot be reprogrammed because irreversible physical changes have occurred as a result of device programming .**

Programming Technologies

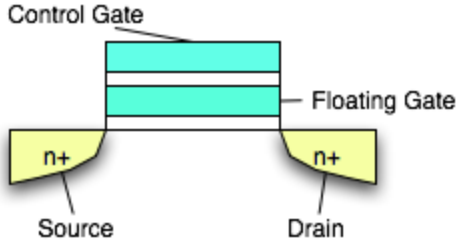
- **Programming technologies (continued):**

3. Single-bit storage element

- **If the stored bit value is a 1, then the transistor is turned ON, and the connection between its source and drain forms a CLOSED path.**
- **If the stored bit value is a 0, then the transistor is turned OFF, and the connection between its source and drain forms a OPEN path.**
- **Reprogrammable: The storage element content can be changed electronically, the device can be easily reprogrammed.**
- **Volatile: The programmed logic is lost in the absence of the power supply voltage.**

4. Transistor Switching Control

- **Stored charge on a floating transistor gate**
  - **Erasable**
  - **Electrically erasable**
  - **Flash (as in Flash Memory)**
- **Storage elements (as in a memory)**



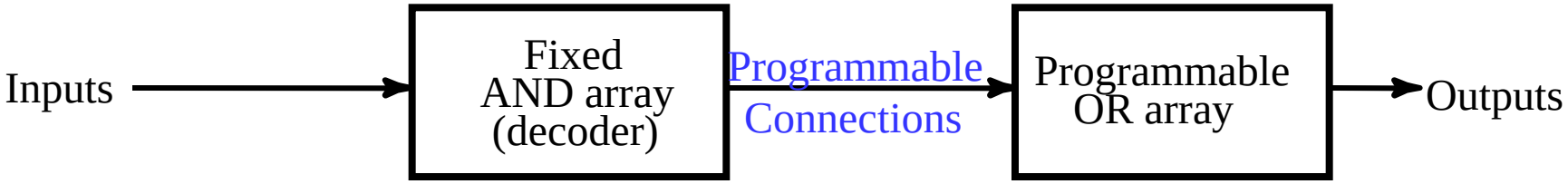
Technology Characteristics

- **Permanent - Cannot be erased and reprogrammed**
  - **Fuse**
  - **Antifuse**
- **Reprogrammable:**
  - **Volatile - Programming lost if chip power lost**
    - **Single-bit storage element**
  - **Non-Volatile**
    - **Erasable**
    - **Electrically erasable**
    - **Flash (as in Flash Memory)**

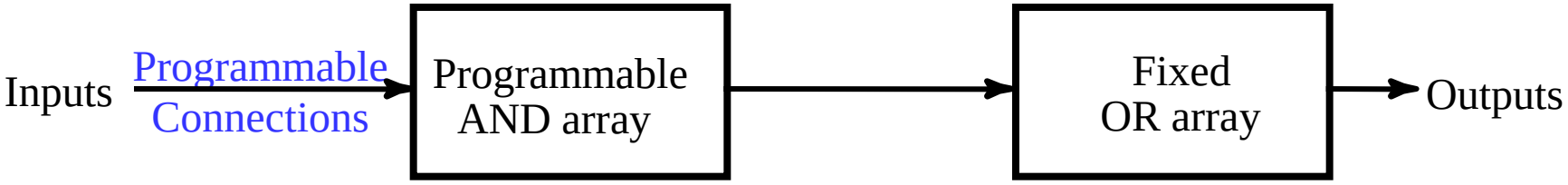
Programmable Configurations

- **Read Only Memory (ROM) - a fixed array of AND gates and a programmable array of OR gates**
- **Programmable Array Logic (PAL)® - a programmable array of AND gates feeding a fixed array of OR gates.**
- **Programmable Logic Array (PLA) - a programmable array of AND gates feeding a programmable array of OR gates.**
- **Complex Programmable Logic Device (CPLD) /Field-Programmable Gate Array (FPGA) - complex enough to be called "architectures" - See VLSI Programmable Logic Devices reading supplement**

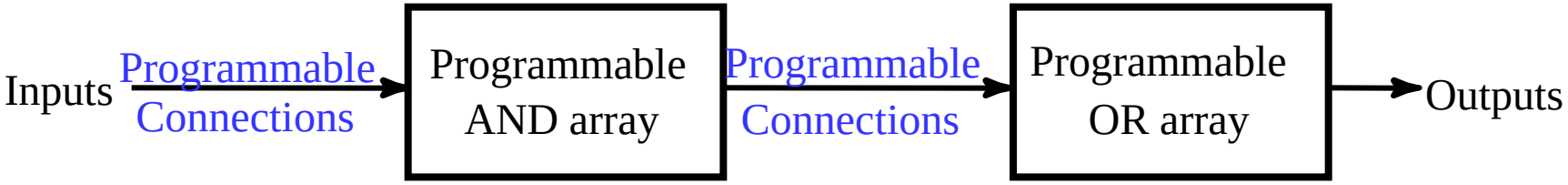
ROM, PAL and PLA Configurations



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL) device



(c) Programmable logic array (PLA) device

Read Only Memory

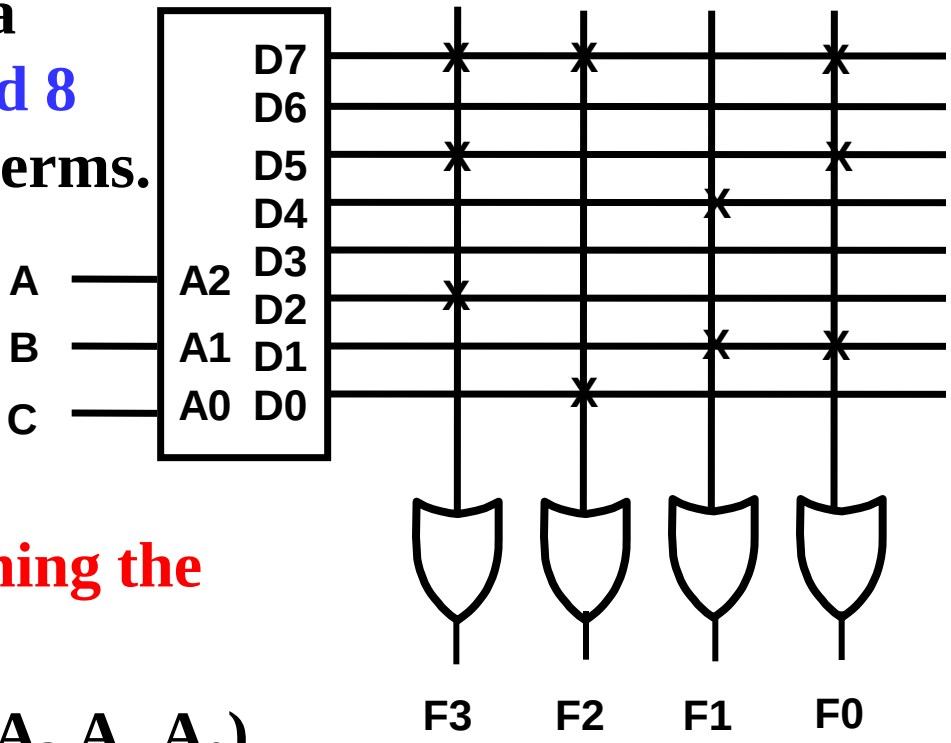
- **Read Only Memories (ROM) or Programmable Read Only Memories (PROM) have:**
  - **N input lines (Address Lines),**
  - **M output lines, and**
  - **2^N decoded minterms.**
- **Fixed AND array with 2^N outputs implementing all N-literal minterms.**
- **Programmable OR Array with M outputs lines to form up to M sum of minterm expressions.**

Read Only Memory

- **A program for a ROM or PROM is simply a multiple-output truth table**
  - **If a 1 entry, a connection is made to the corresponding minterm for the corresponding output**
  - **If a 0, no connection is made**
- **Can be viewed as a *memory* with the inputs as *addresses* of *data* (output values), hence ROM or PROM names!**

Read Only Memory Example

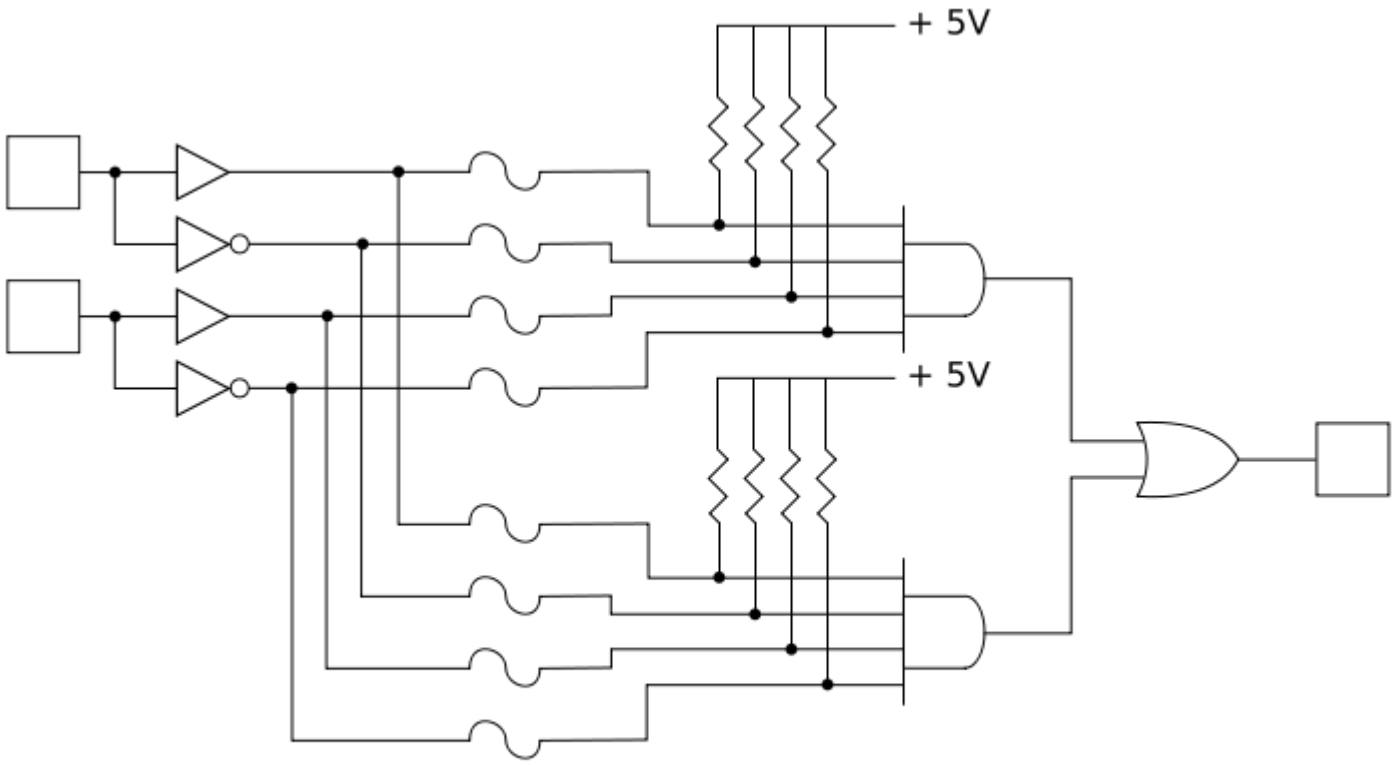
- Example: A 8 X 4 ROM (N = 3 input lines, M= 4 output lines)
- The fixed "AND" array is a "decoder" with 3 inputs and 8 outputs implementing minterms.
- The programmable "OR" array uses a single line to represent all inputs to an OR gate. An "X" in the array corresponds to attaching the minterm to the OR



- Read Example: For input (A_2,A_1,A_0) = 001, output is (F_3,F_2,F_1,F_0) = 0011.
- What are functions F_3, F_2 , F_1 and F_0 in terms of (A_2, A_1, A_0)?

Programmable Array Logic (PAL)

- **The PAL is the opposite of the ROM, having a programmable set of ANDs combined with fixed ORs.**



Simplified programmable logic device

Programmable Array Logic Example

- **4-input, 3-output PAL with fixed, 3-input OR terms**
- **What are the equations for F1 through F4?**

$F1 = \overline{A}\,\overline{B} + \overline{C}$

$F2 = \overline{A}\,B\,\overline{C} + A\,C + A\,\overline{B}$

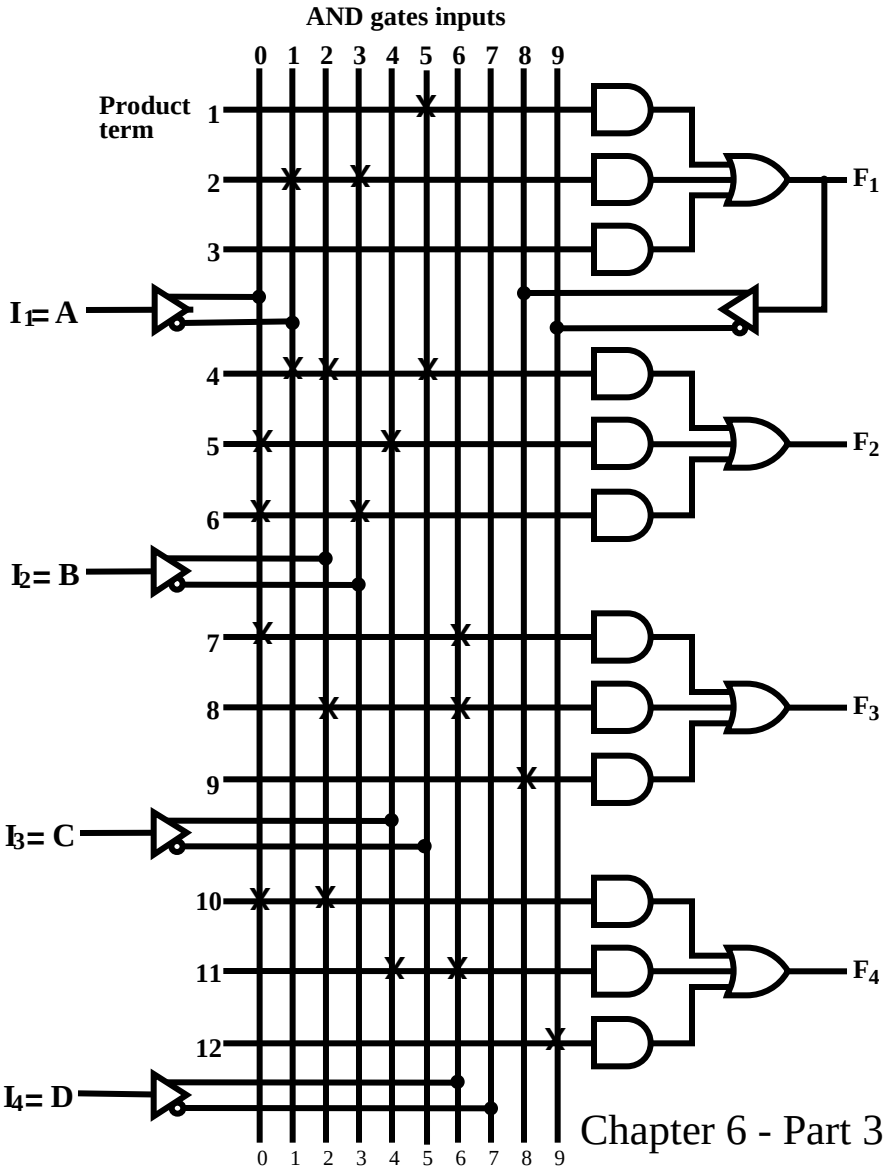
$F3 = AD + BD + F1$

$= AD + BD + \overline{A}\,\overline{B} + \overline{C}$

$F4 = AB + CD + \overline{F1}$

$= AB + CD + (A+B)C$

$= AB + CD + AC + BC$



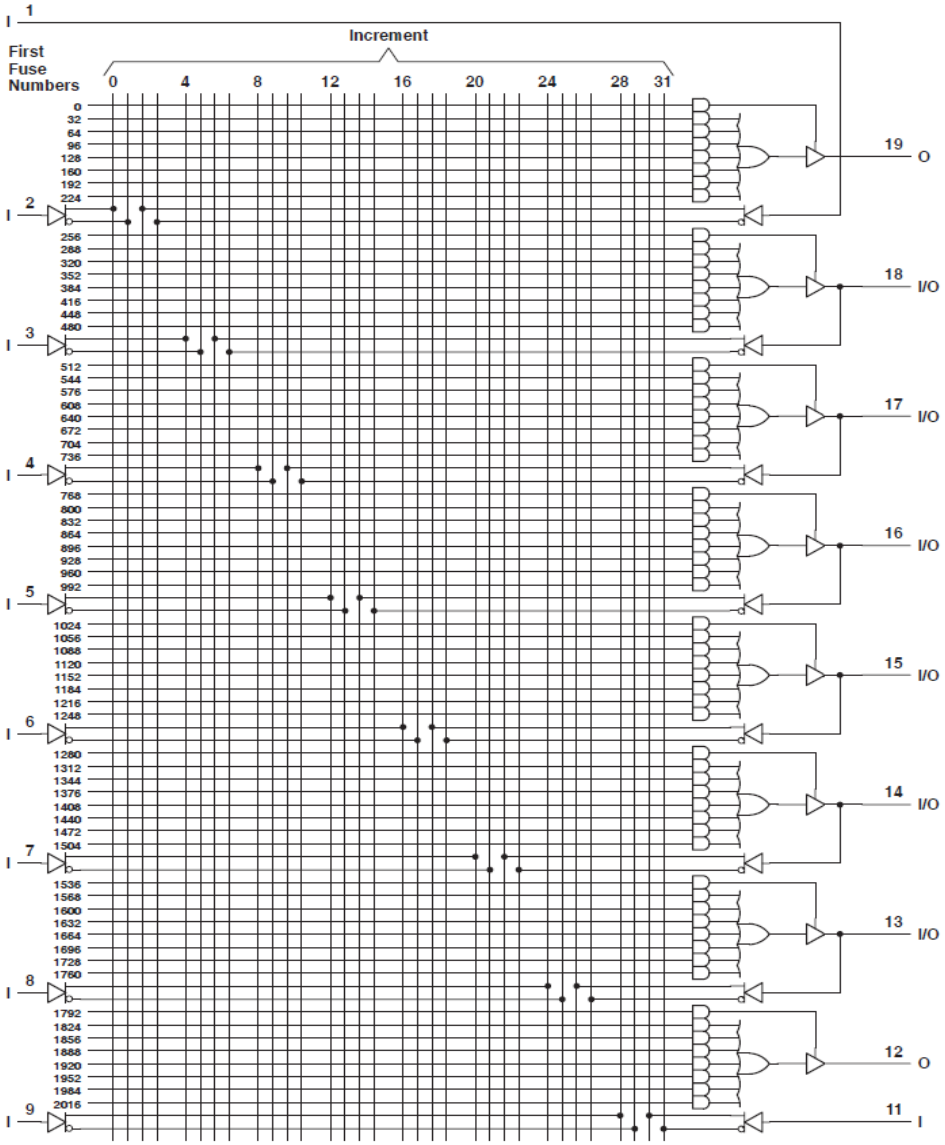
Programmable Array Logic (PAL)

- **Advantages of PAL:**
  - **For given internal complexity, a PAL can have larger N and M**
  - **Some PALs have outputs that can be complemented, adding POS functions**
  - **No multilevel circuit implementations in ROM (without external connections from output to input). PAL has outputs from OR terms as internal inputs to all AND terms, making implementation of multi-level circuits easier.**

- **Disadvantage of PAL:**
  - **ROM guaranteed to implement any M functions of N inputs. PAL may have too few inputs to the OR gates.**

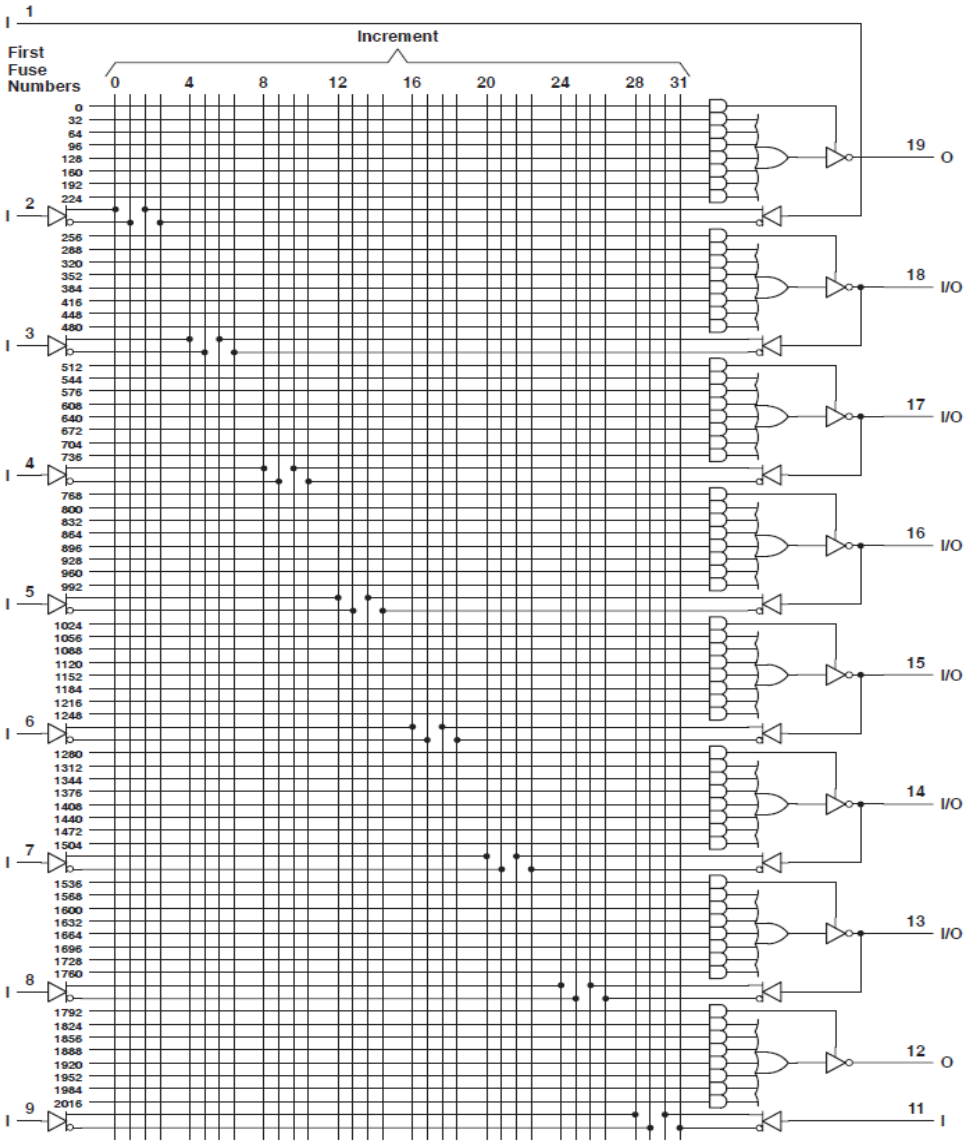
PAL16H8

http://www.datasheets.org.uk/pdf-datasheets/Databooks-2/Book291-186.html



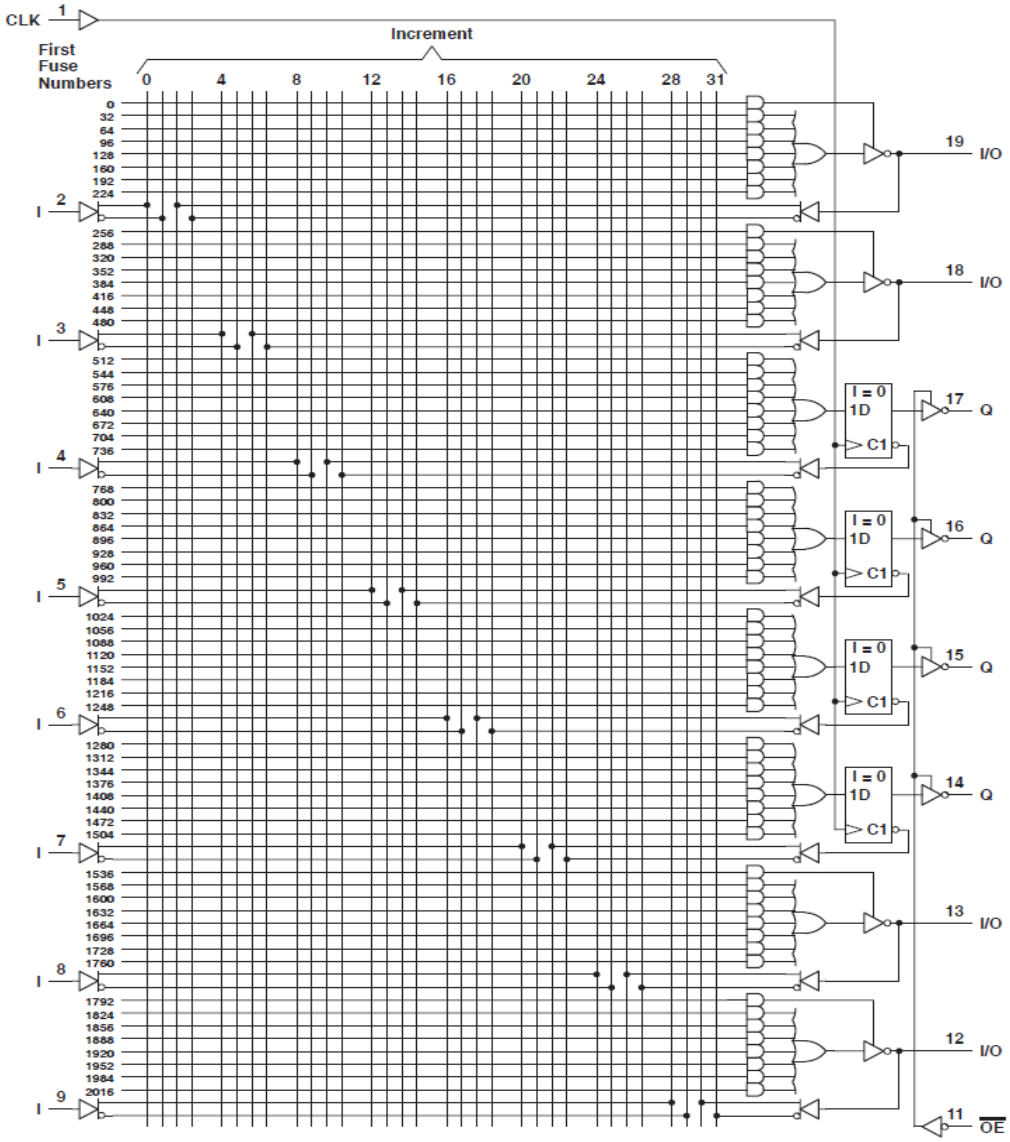
PAL16L8

http://www.datasheetcatalog.org/datasheets/166/34602_DS.pdf



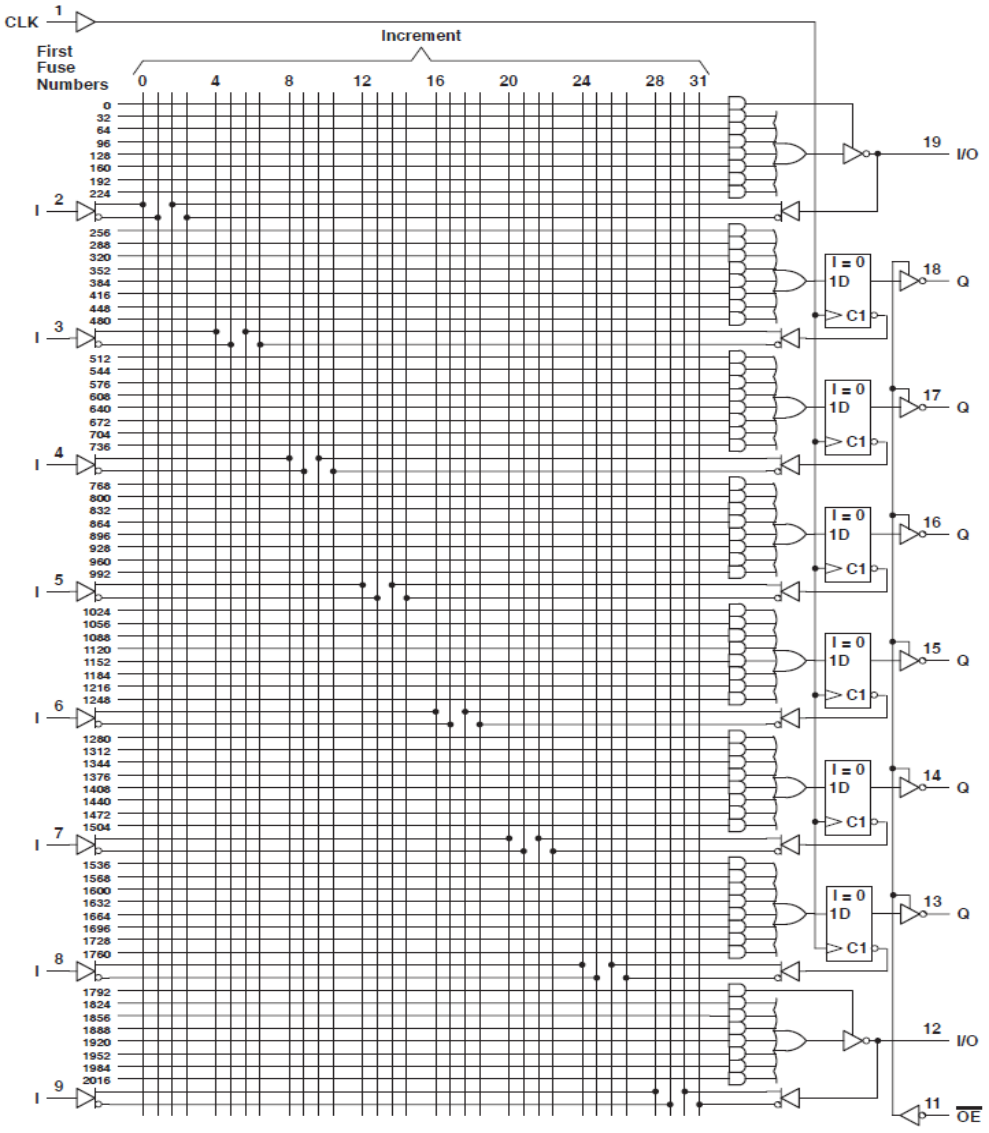
PAL16R4

http://www.datasheetcatalog.org/datasheets/166/34602_DS.pdf



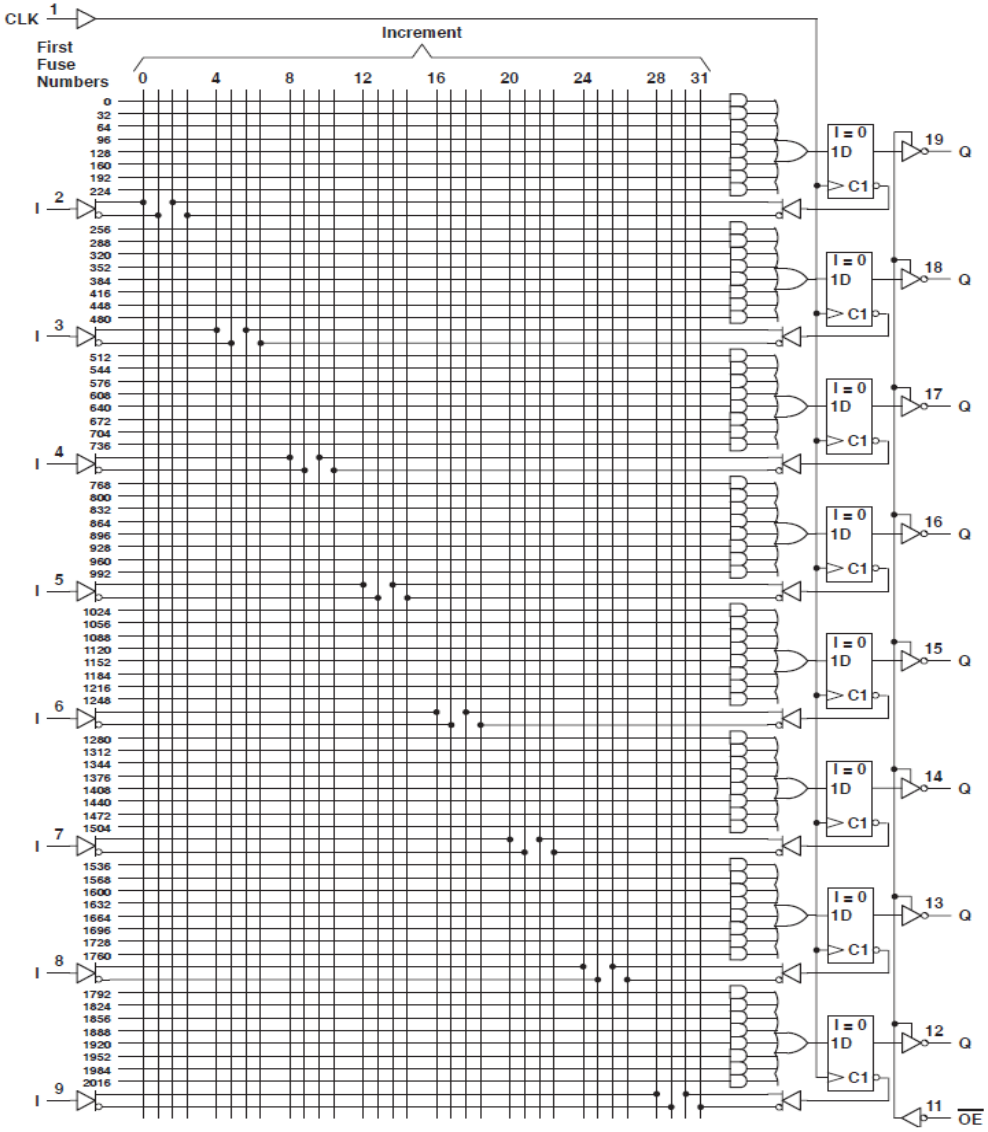
PAL16R6

http://www.datasheetcatalog.org/datasheets/166/34602_DS.pdf



PAL16R8

http://www.datasheetcatalog.org/datasheets/166/34602_DS.pdf



Programmable Logic Array (PLA)

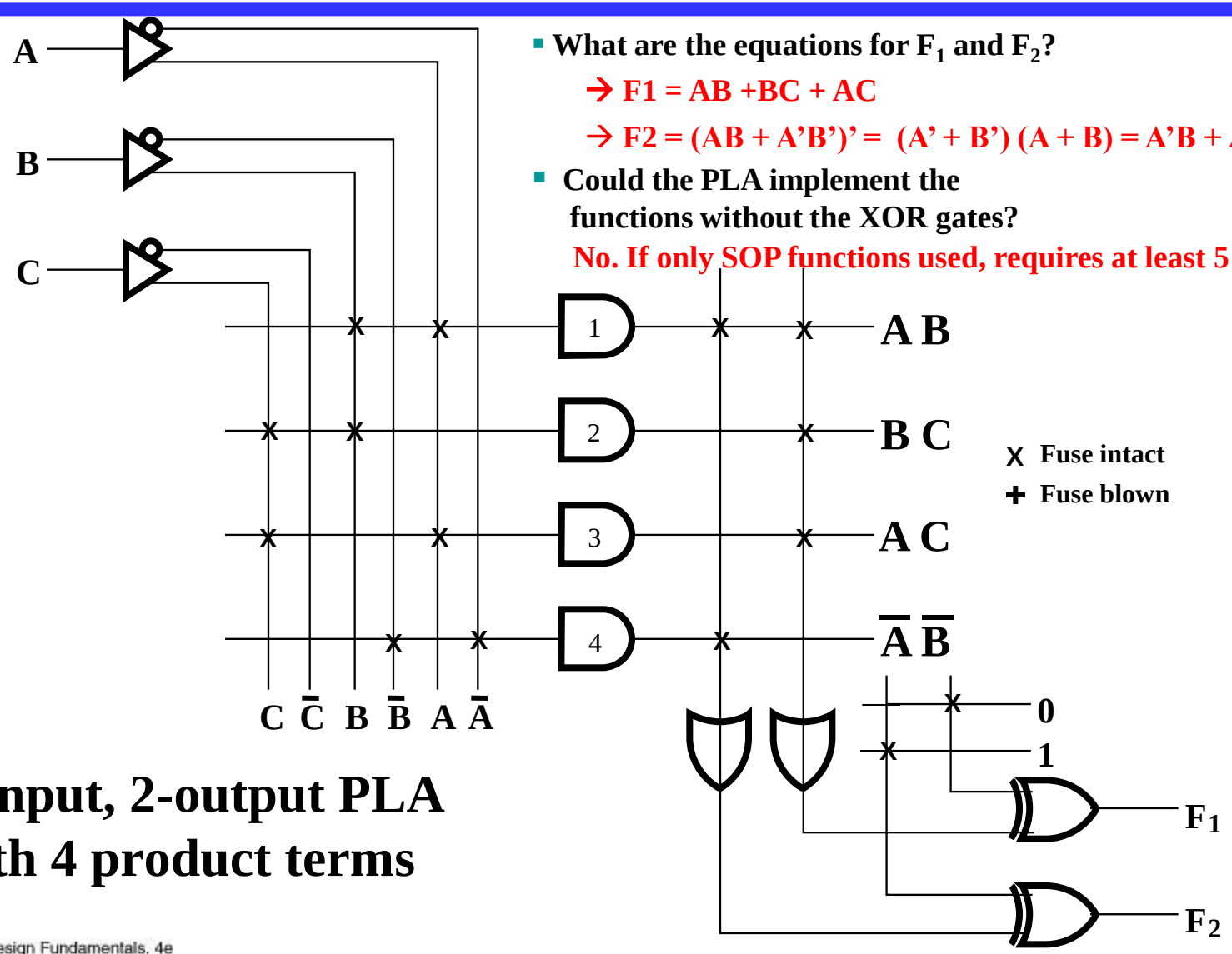
- **Compared to a ROM and a PAL, a PLA is the most flexible having a programmable set of ANDs combined with a programmable set of ORs.**

- **Advantages**
  - **A PLA can have large N and M permitting implementation of equations that are impractical for a ROM (because of the number of inputs, N, required**
  - **A PLA has all of its product terms connectable to all outputs, overcoming the problem of the limited inputs to the PAL ORs**
  - **Some PLAs have outputs that can be complemented, adding POS functions**

Programmable Logic Array (PLA)

- **Disadvantages**
  - **Often, the product term count limits the application of a PLA.**
  - **Two-level multiple-output optimization is required to reduce the number of product terms in an implementation, helping to fit it into a PLA.**
  - **Multi-level circuit capability available in PAL not available in PLA. PLA requires external connections to do multi-level circuits.**

Programmable Logic Array Example



- What are the equations for F_1 and F_2?
  - → F1 = AB +BC + AC
  - → F2 = (AB + A'B')' = (A' + B') (A + B) = A'B + AB'
- Could the PLA implement the functions without the XOR gates?
  No. If only SOP functions used, requires at least 5 AND gates.

- **3-input, 2-output PLA with 4 product terms**

Terms of Use

- **All (or portions) of this material © 2008 by Pearson Education, Inc.**
- **Permission is given to incorporate this material or adaptations thereof into classroom presentations and handouts to instructors in courses adopting the latest edition of Logic and Computer Design Fundamentals as the course textbook.**
- **These materials or adaptations thereof are not to be sold or otherwise offered for consideration.**
- **This Terms of Use slide or page is to be included within the original materials or any adaptations thereof.**