Logic and Computer Design Fundamentals

Chapter 6 – Selected Design Topics

Part 2 Propagation Delay and Timing
Part 4 Programmable Implementation Technologies

Yueming Wang (王跃明)

ymingwang@zju.edu.cn

2017

College of Computer Science, Zhejiang University

Qiushi Academy for Advanced Studies, Zhejiang University

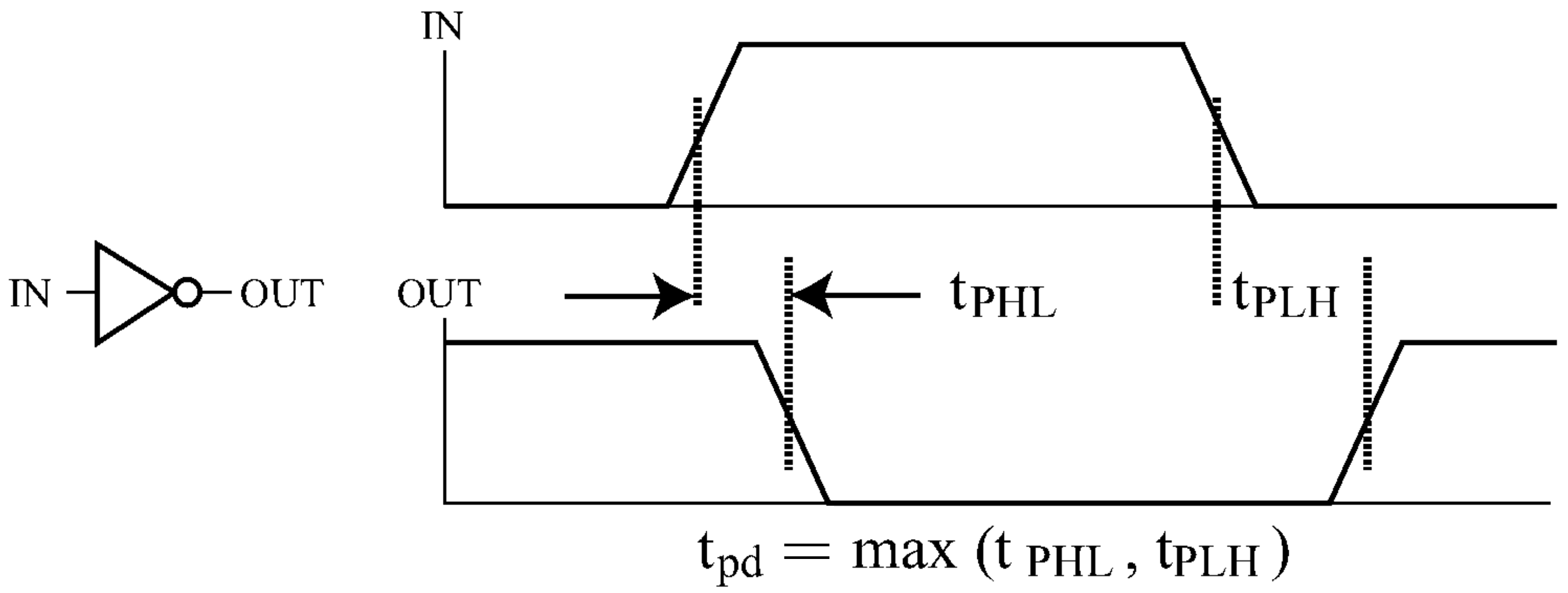
Overview

- 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 – Asynchronous Interactions
- Part 4 - 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, <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.

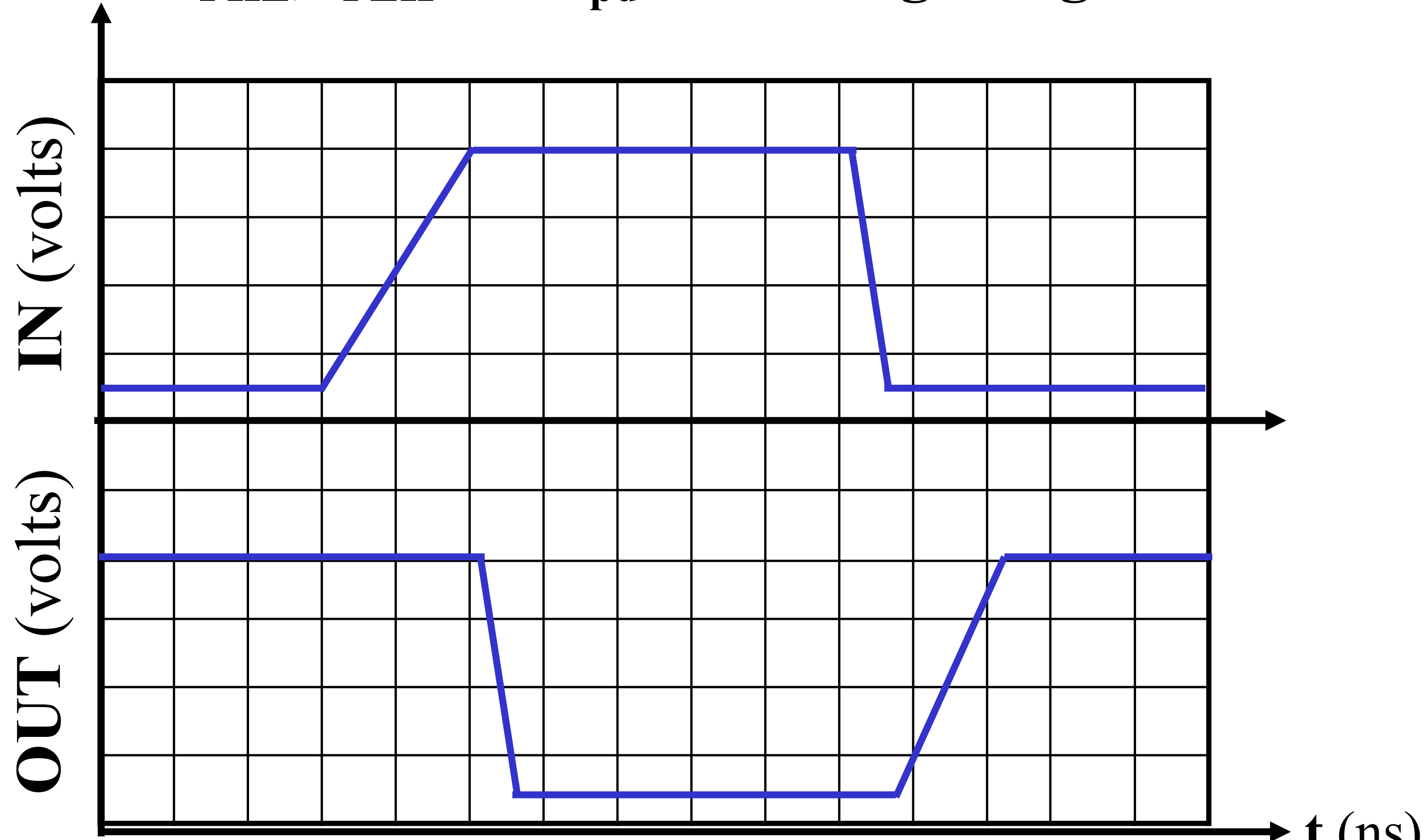
Propagation Delay (continued)



- **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

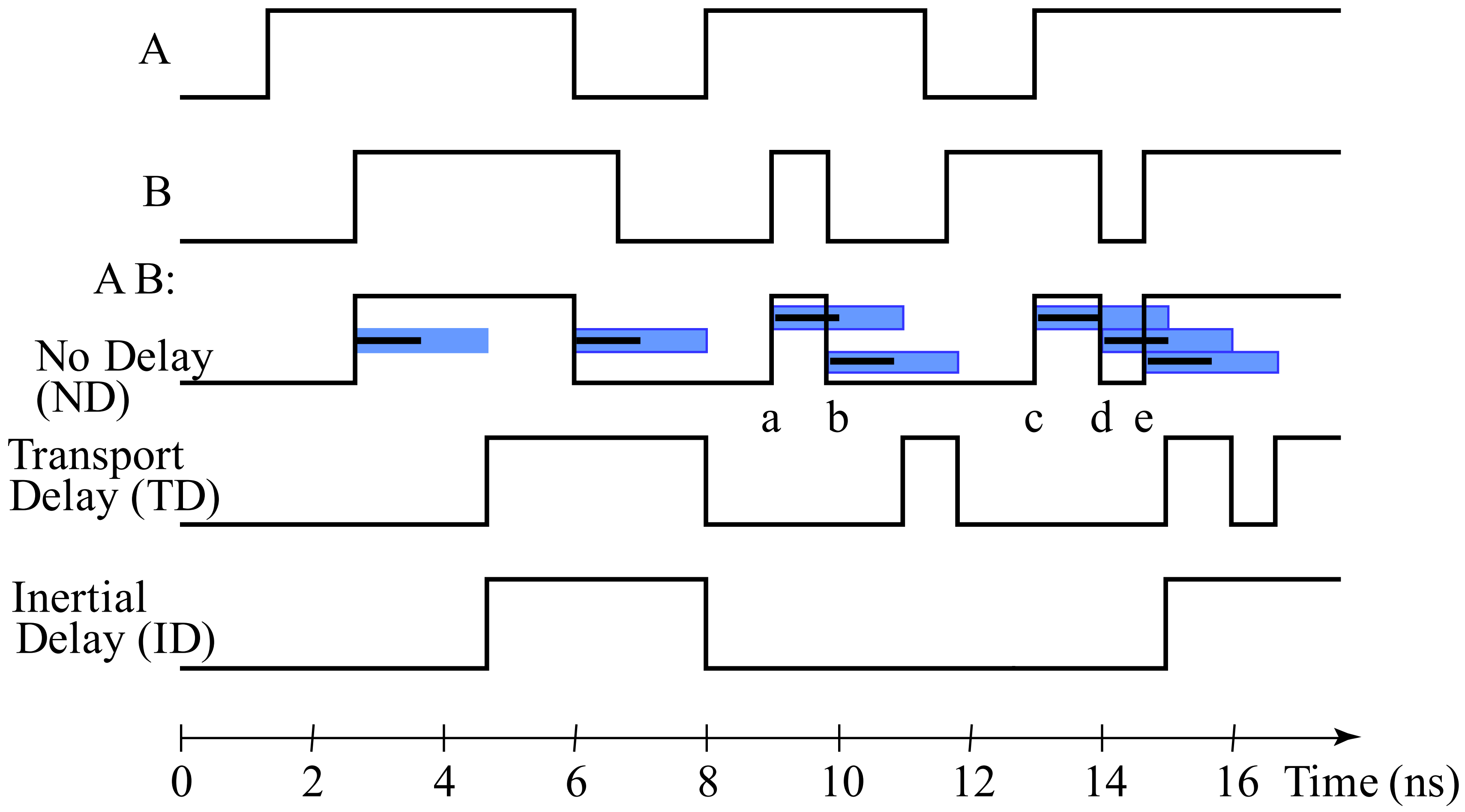


1.0 ns per division

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**

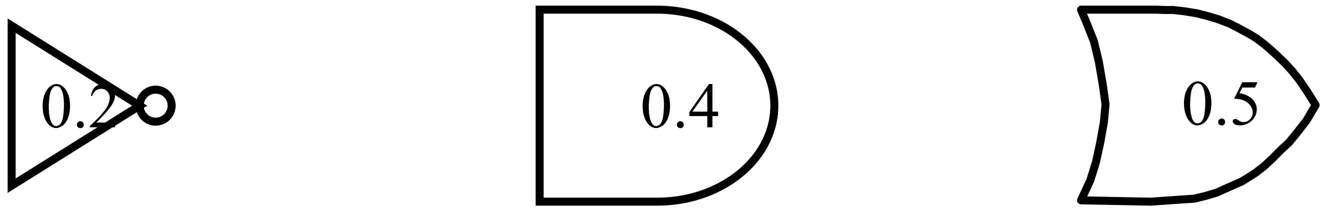
Delay Model Example



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

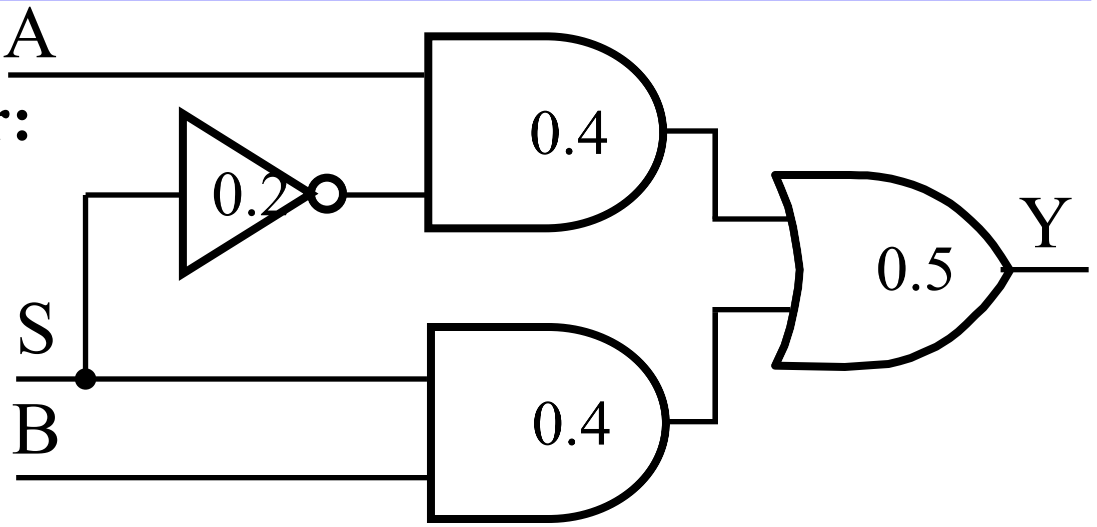
Circuit Delay

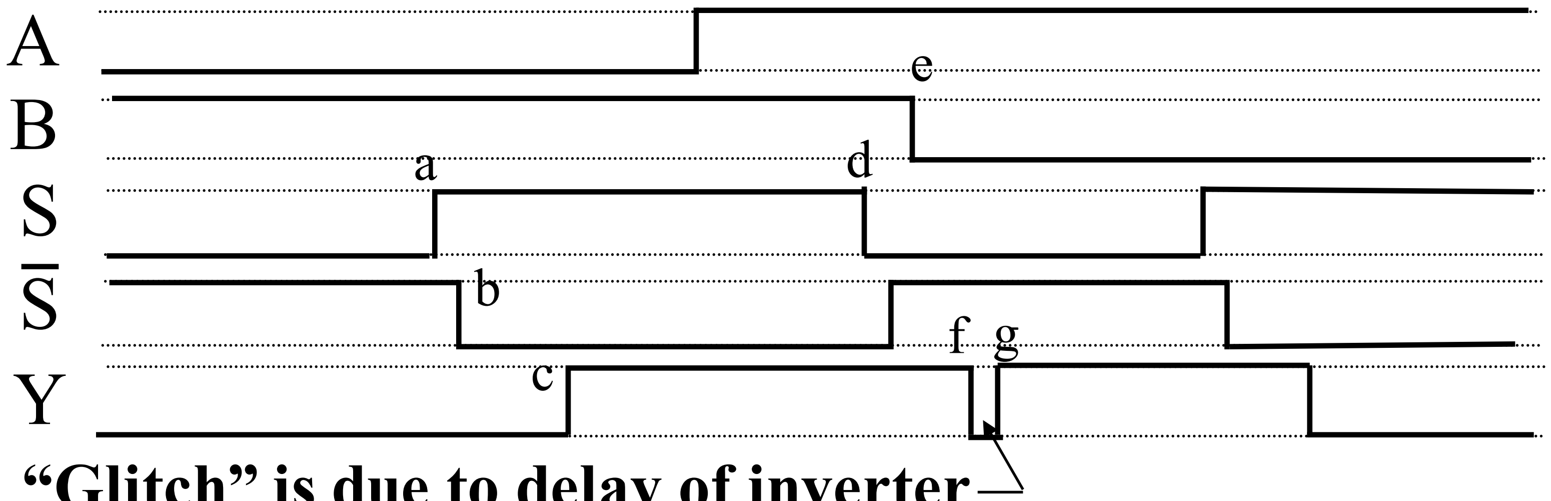
- **Suppose gates with delay *n* ns are represented for *n* = 0.2 ns, *n* = 0.4 ns, *n* = 0.5 ns, respectively:**



Circuit Delay

- Consider a simple 2-input multiplexer:
- With function:
  - Y = A for S = 0
  - Y = B for S = 1





- "Glitch" is due to delay of inverter

Fan-out and Delay

- **The fan-out loading a gate's output affects the gate's propagation delay**
- **Example:**
  - **One realistic equation for t_{pd} for a NAND gate with 4 inputs is:**

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

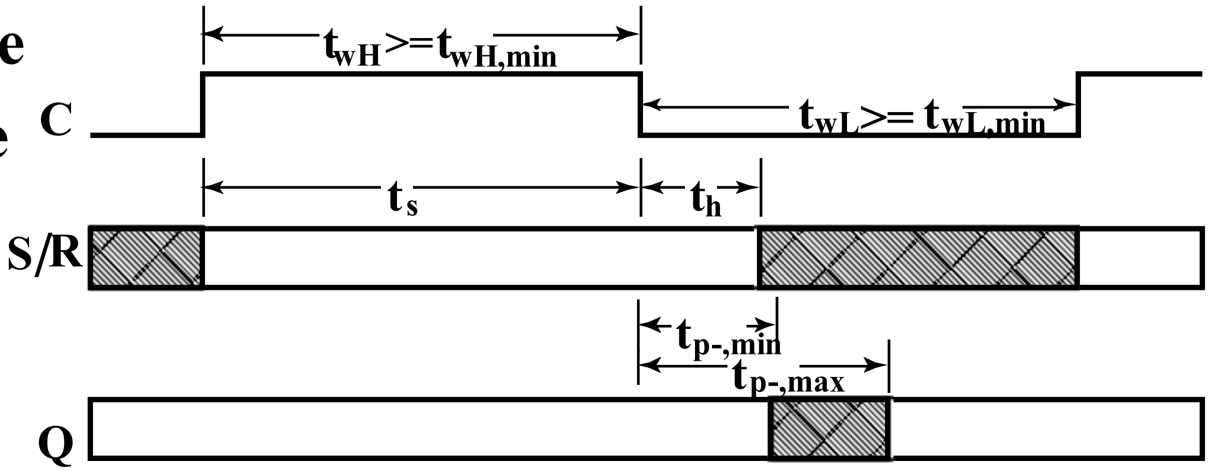
  - **SL is the number of standard loads the gate is driving, i. e., its fan-out in standard loads**
  - **For SL = 4.5, $t_{pd} = 0.165$ ns**
- **If this effect is considered, the delay of a gate in a circuit takes on different values depending on the circuit load on its output.**

Cost/Performance Tradeoffs

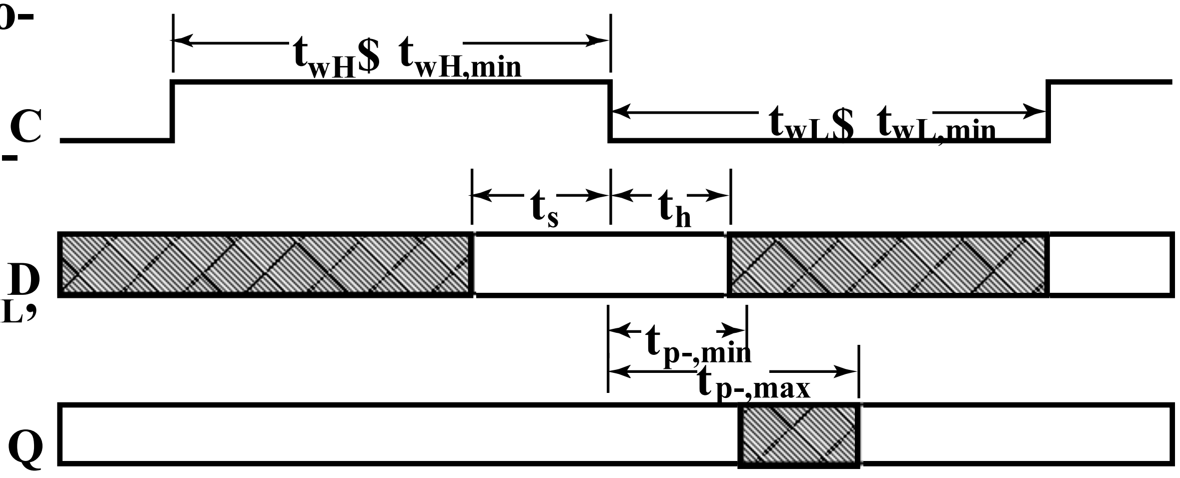
- Gate-Level Example:
  - NAND gate G with 20 standard loads on its output has a delay of 0.45 ns and has a normalized cost of 2.0
  - A buffer H has a normalized cost of 1.5. The NAND gate driving the buffer with 20 standard loads gives a total delay of 0.33 ns
  - In which if the following cases should 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

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)



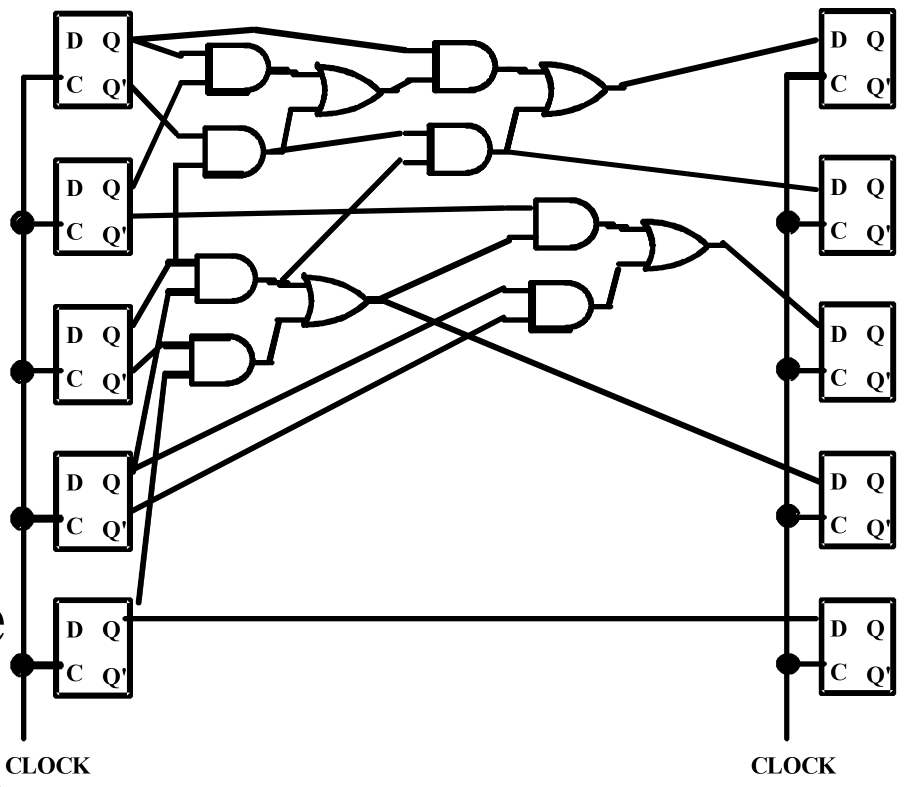
(b) Edge-triggered (negative edge)

Flip-Flop Timing Parameters

- t_s - setup time
  - Master-slave - Equal to the width of the triggering pulse
  - 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 <u>except</u>
  - 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 <u>clock period</u> is too short, some data changes will not propagate through the circuit to flip-flop inputs before the setup time interval begins

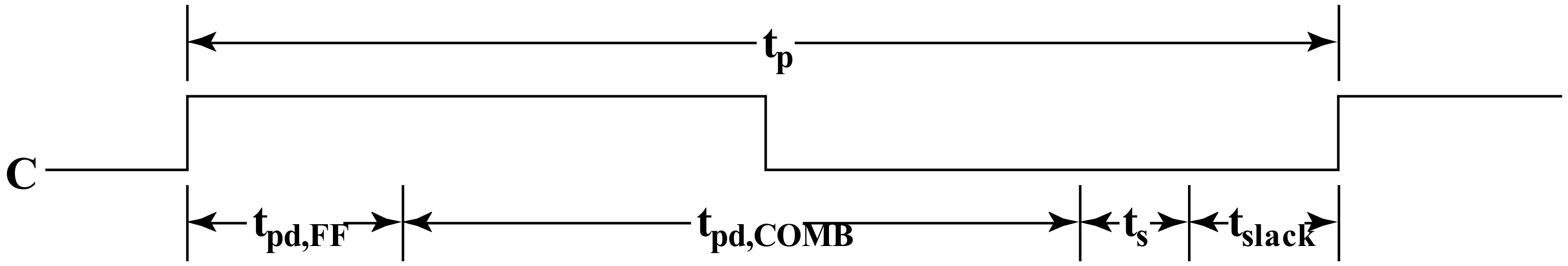


Circuit and System Level Timing

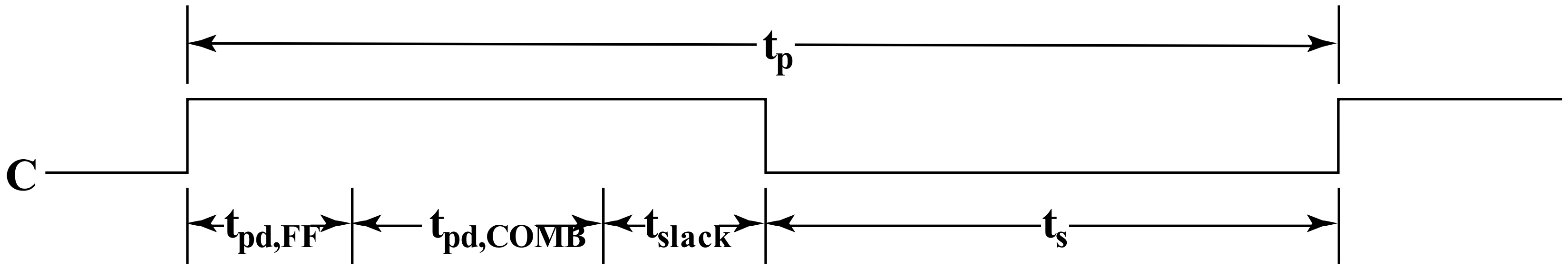
- New Timing Components
  - t_p - clock period - The interval between occurrences of a specific clock edge in a periodic clock
  - $t_{pd,COMB}$ - total delay of combinational logic along the path from flip-flop output to flip-flop input
  - t_{slack} - extra time in the clock period in addition to the sum of the delays and setup time on a path
    - Can be either positive or negative
    - 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

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

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**

Overview

- Part 1 – The Design Space
- Part 2 – Propagation Delay and Timing
- Part 3 – Asynchronous Interactions
- Part 4 - 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 - 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 are used to:
  - Control connections
  - Build lookup tables
  - Control transistor switching
- The technologies
  - Control connections
    - Mask programming
    - Fuse
    - Antifuse
    - Single-bit storage element

Programming Technologies

- The technologies (continued)
  - Build lookup tables
    - Storage elements (as in a memory)
  - 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

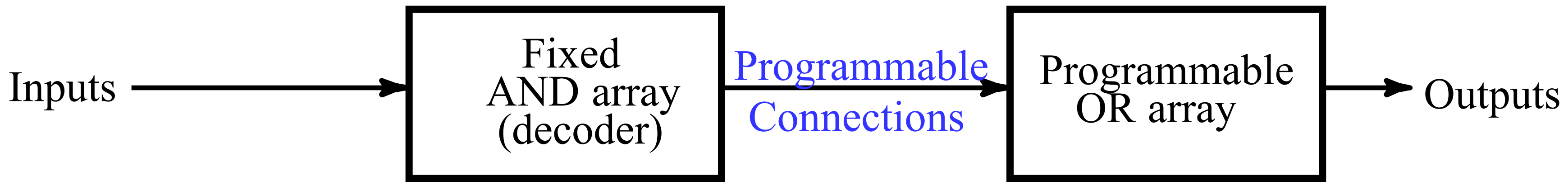
- **Permanent - Cannot be erased and reprogrammed**
    - **Mask programming**
    - **Fuse**
    - **Antifuse**
- **Reprogrammable**
    - **Volatile - Programming lost if chip power lost**
        - **Single-bit storage element**
    - **Non-Volatile**
        - **Erasable**
        - **Electrically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

Programmable Configurations

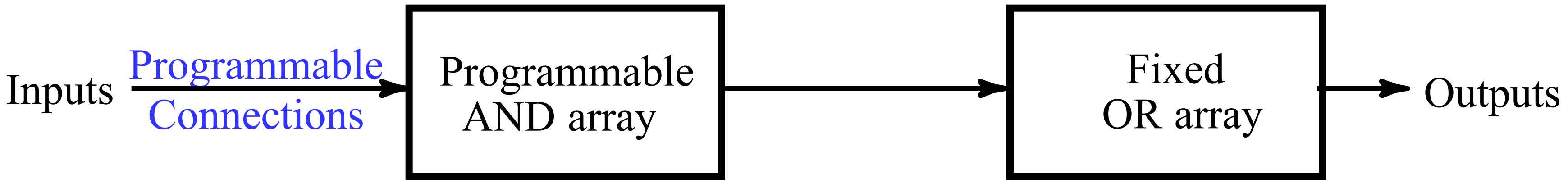
- ***Read Only Memory* (*ROM*) - a fixed array of AND gates and a programmable array of OR gates**
- ***Programmable Array Logic* (*PAL*)® - a programmable array of AND gates feeding a fixed array of OR gates.**
- ***Programmable Logic Array* (*PLA*) - a programmable array of AND gates feeding a programmable 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**

PAL is a registered trademark of Lattice Semiconductor Corp.

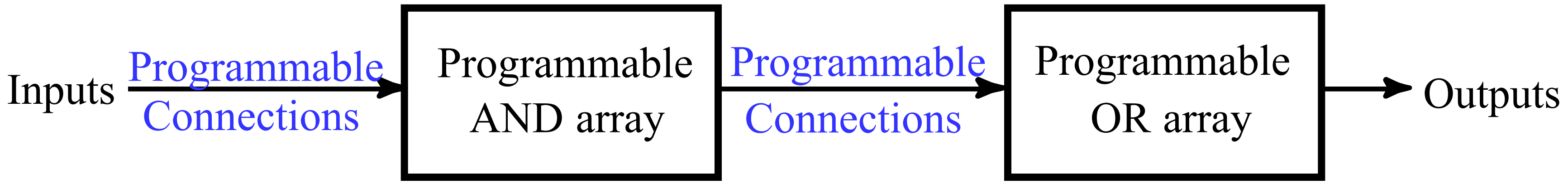
ROM, PAL and PLA Configurations



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL) device



(c) Programmable logic array (PLA) device

Read Only Memory

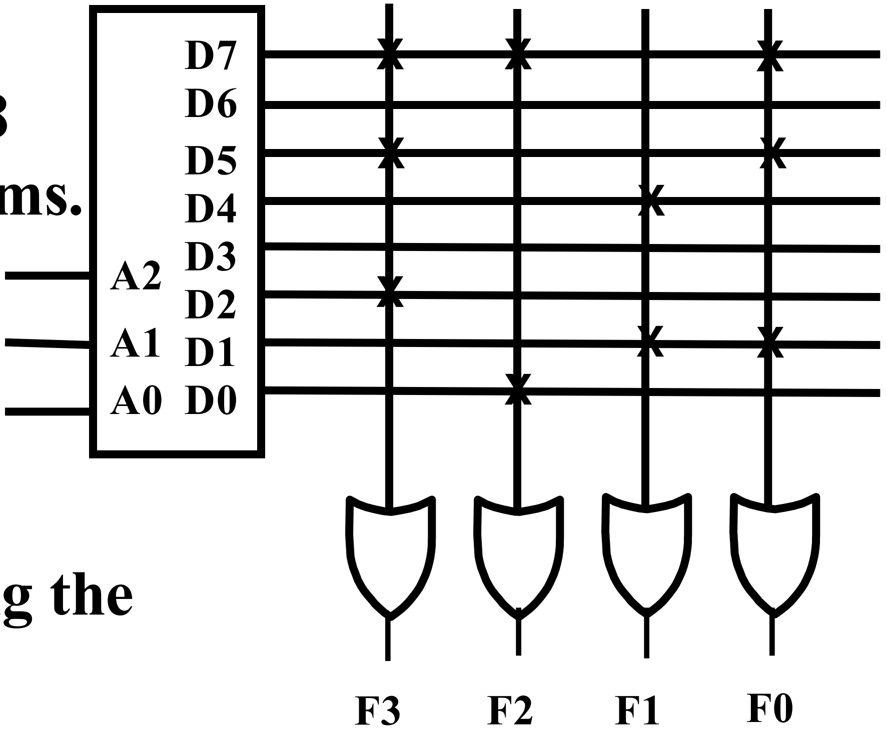
- Read Only Memories (ROM) or Programmable Read Only Memories (PROM) have:
  - N input lines,
  - M output lines, and
  - 2^N decoded minterms.
- <u>Fixed</u> AND array with 2^N outputs implementing all N-literal minterms.
- <u>Programmable</u> OR Array with M outputs lines to form up to M sum of minterm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

Example: Square of 3-bit input number

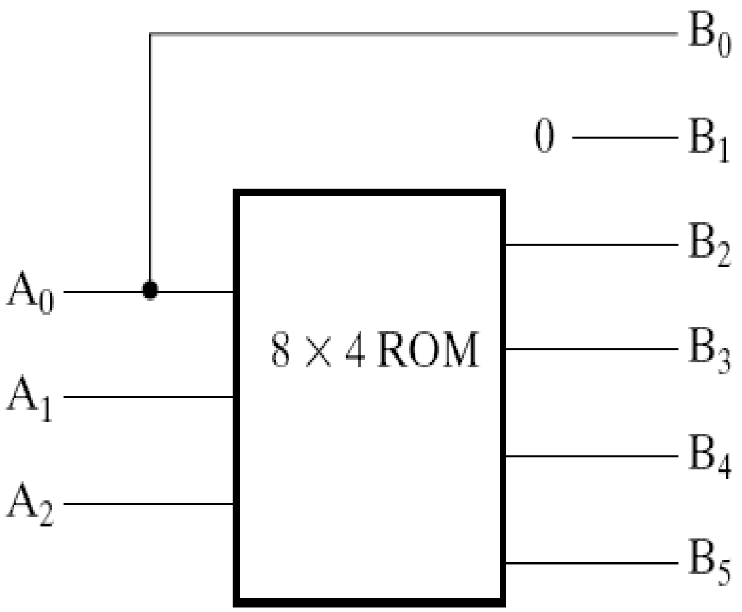
Inputs			Outputs						
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

$B_0 = A_0$

$B_1 = 0$

Example: Square of 3-bit input number

- **8 × 4 ROM are selected**



ROM Truth Table

A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

Programmable Array Logic (PAL)

- The PAL is the opposite of the ROM, having a programmable set of ANDs combined with fixed ORs.
- Advantages
  - 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
  - ROM guaranteed to implement any M functions of N inputs. PAL may have too few inputs to the OR gates.

Programmable Array Logic Example

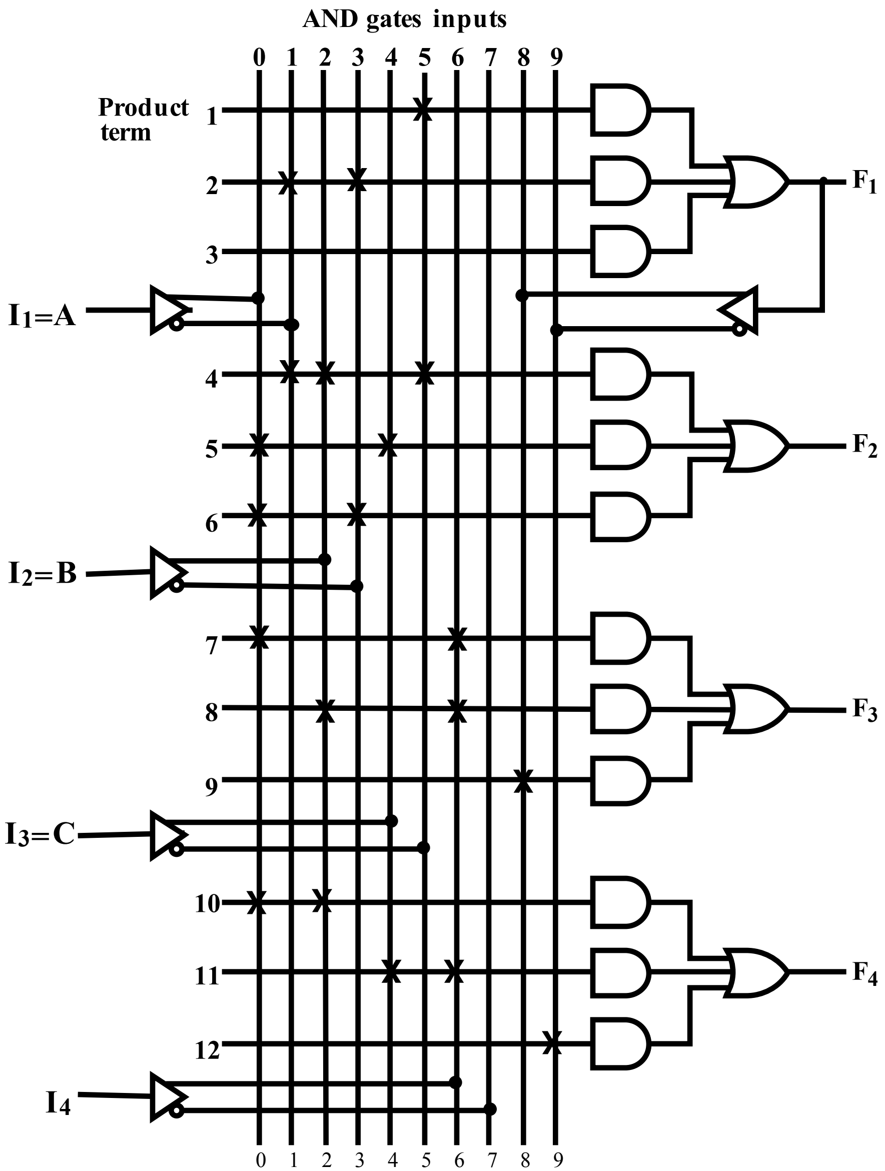
- 4-input, 4-output PAL with fixed, 3-input OR terms
- What are the equations for F1 through F4?

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

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

$$F3 =$$

$$F4 =$$



Programmable Array Logic

- **Design requires fitting functions within the limited number of ANDs per OR gate**
- **Single function optimization is the first step to fitting**
- **Otherwise, if the number of terms in a function is greater than the number of ANDs per OR gate, then factoring is necessary**

Programmable Array Logic Example

- Equations: $F1 = A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + \overline{A}\,\overline{B}C + ABC$
  $F2 = AB + BC + AC$
- F1 must be factored since four terms
- Factor out last two terms as W

Product term	AND Inputs A	B	C	D	W	Outputs
1	0	0	1	—	—	$W = \overline{A}\,\overline{B}C + ABC$
2	1	1	1	—	—	
3	—	—	—	—	—	
4	1	0	0	—	—	$F1 = X = A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + W$
5	0	1	0	—	—	
6	—	—	—	—	1	
7	1	1	—	—	—	$F2 = Y = AB + BC + AC$
8	—	1	1	—	—	
9	1	—	1	—	—	
10	—	—	—	—	—	
11	—	—	—	—	—	
12	—	—	—	—	—	

Programmable Array Logic Example

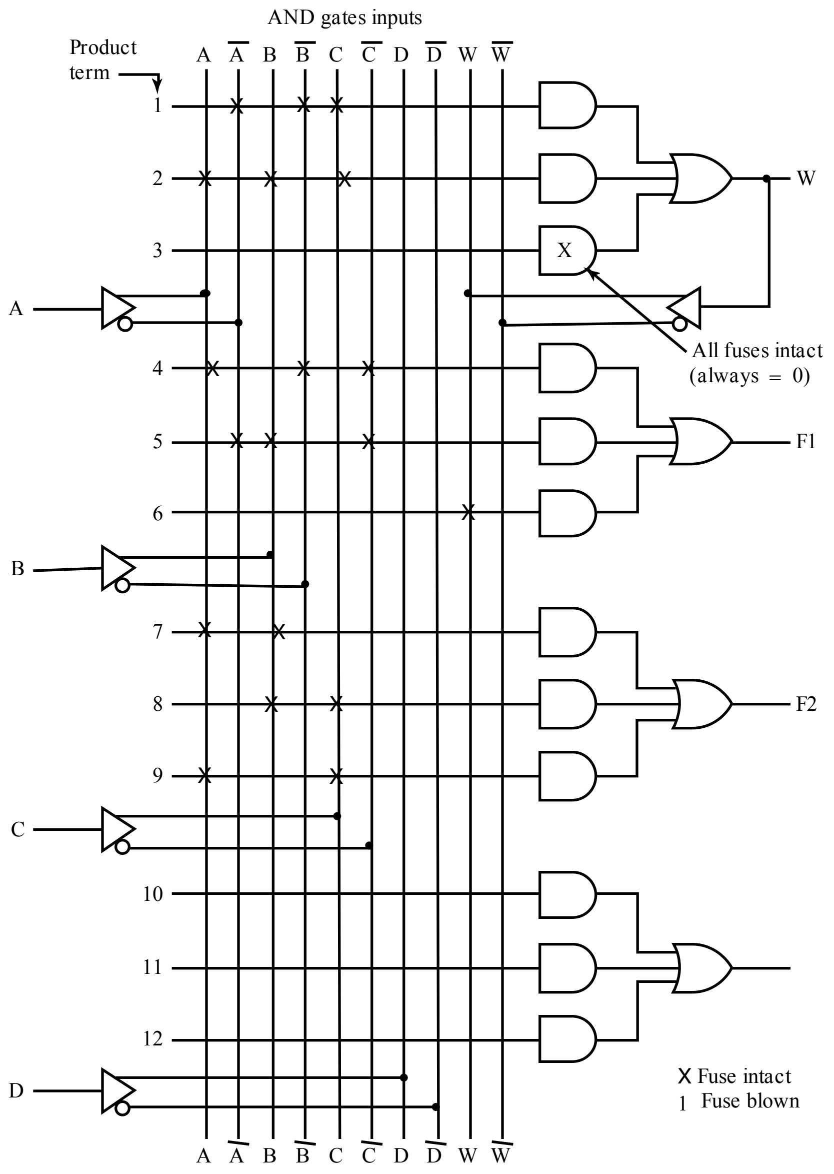
$W = \overline{A}\,\overline{B}C + ABC$

$F1 = X =$

$A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + W$

$F2 = Y =$

$AB + BC + AC$



AND gates inputs
Product term
A $\overline{A}$ B $\overline{B}$ C $\overline{C}$ D $\overline{D}$ W $\overline{W}$
1
2
3
4
5
6
7
8
9
10
11
12
A
B
C
D
X
W
F1
F2
All fuses intact (always = 0)
X Fuse intact
1 Fuse blown

Programmable Logic Array (PLA)

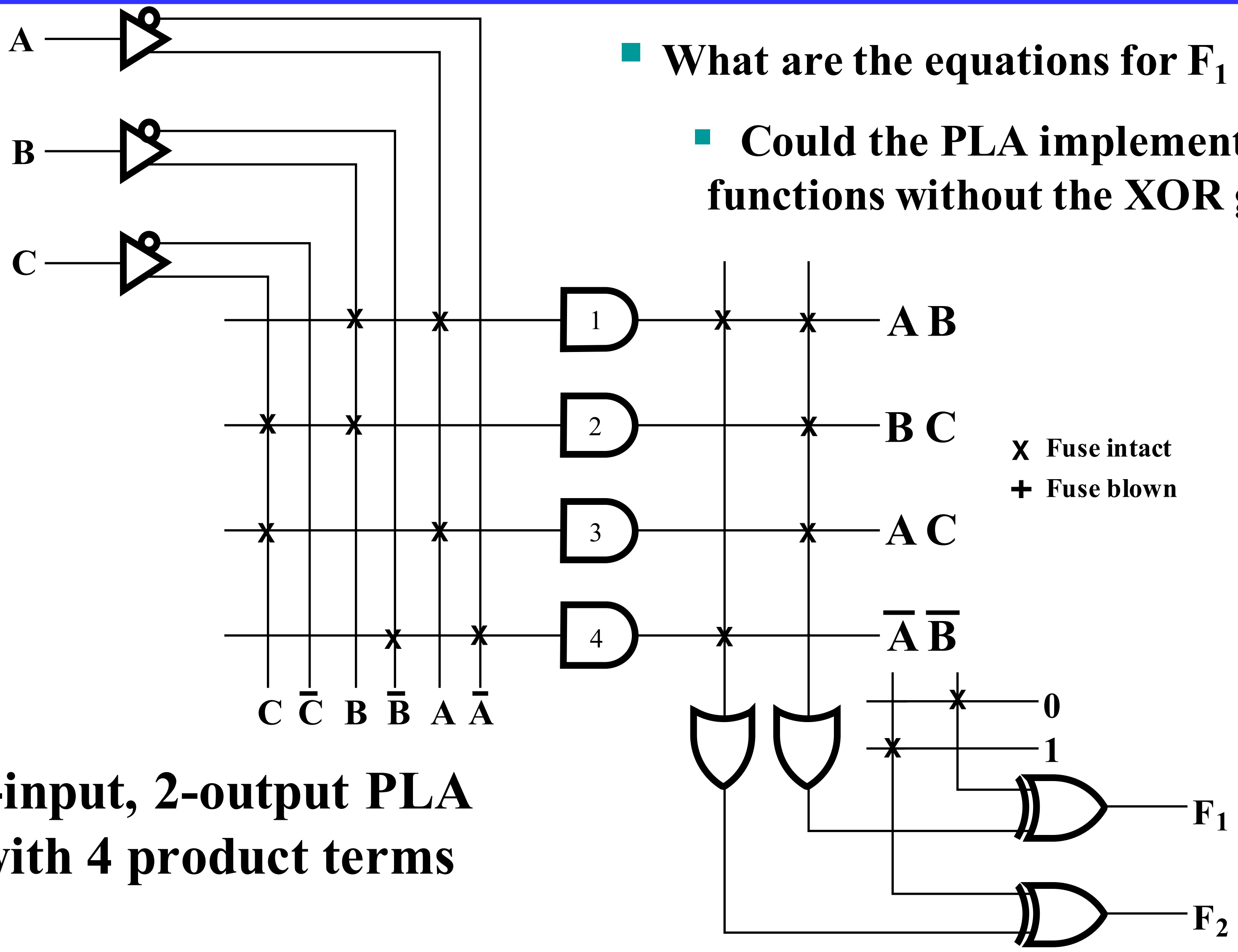
- Compared to a ROM and a PAL, a PLA is the most flexible having a programmable set of ANDs combined with a programmable set of ORs.
- Advantages
  - A PLA can have large N and M permitting implementation of equations that are impractical for a ROM (because of the number of inputs, N, required
  - A 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?
  - Could the PLA implement the functions without the XOR gates?



A
B
C
1
2
3
4
A B
B C
A C
$\overline{A}\,\overline{B}$
C $\overline{C}$ B $\overline{B}$ A $\overline{A}$
X Fuse intact
+ Fuse blown
0
1
F_1
F_2

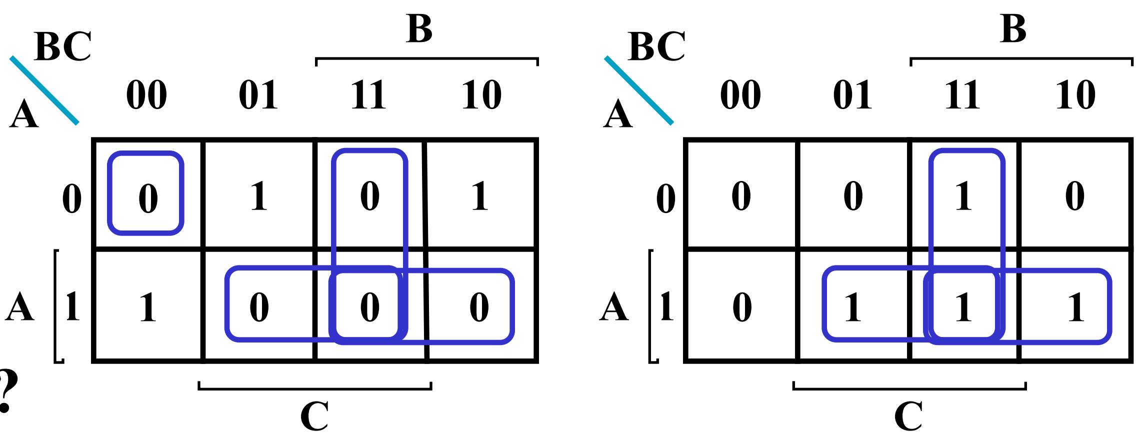
- 3-input, 2-output PLA with 4 product terms

Programmable Logic Array

- The set of functions to be implemented must fit the available number of product terms
- The number of literals per term is less important in fitting
- Since output inversion is available, terms can implement either a function or its complement
- For small circuits, K-maps can be used to visualize product term sharing and use of complements
  - The best approach to fitting is multiple-output, two-level optimization (which has not been discussed)
- For larger circuits, software is used to do the optimization including use of complemented functions

Programmable Logic Array Example

- K-map specification
- How can this be implemented with four terms?
- Complete the programming table



$F_1 = \bar{A}\,\bar{B}C + \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C}$

$\bar{F}_1 = AB + AC + BC + \bar{A}\,\bar{B}\,\bar{C}$

$F_2 = AB + AC + BC$

$\bar{F}_2 = \bar{A}\bar{C} + \bar{A}\bar{B} + \bar{B}\,\bar{C}$

PLA programming table

	Product term	Inputs A	B	C	Outputs (C) F_1	(T) F_2
AB	1	1	1	–	1	1
AC	2	1	–	1	1	1
BC	3	–	1	1	1	1
$\bar{A}\bar{B}\bar{C}$	4	0	0	0	1	–

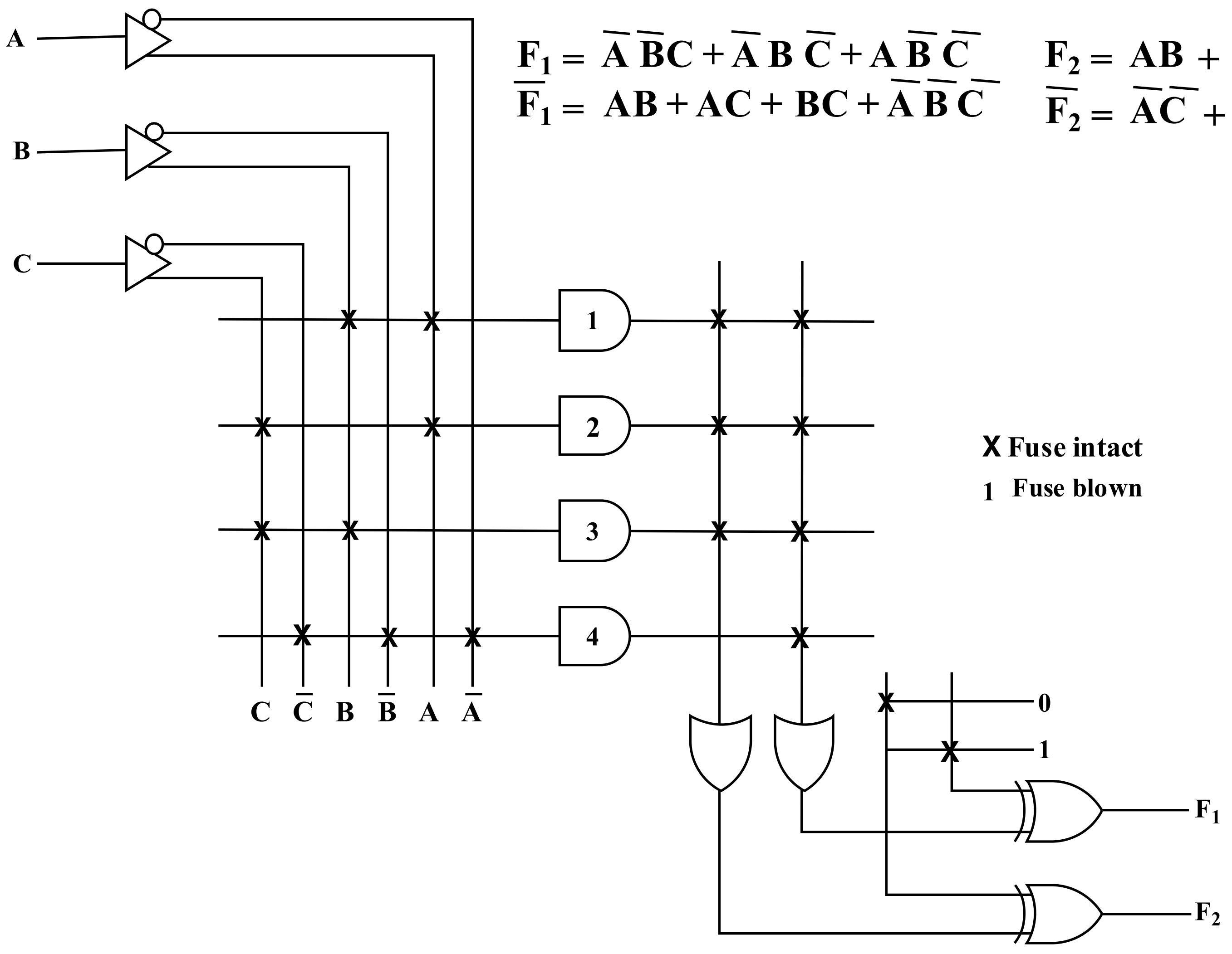
Programmable Logic Array Example

$$F_1 = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C}$$

$$\overline{F_1} = AB + AC + BC + \overline{A}\,\overline{B}\,\overline{C}$$

$$F_2 = AB + AC + BC$$

$$\overline{F_2} = \overline{A}\,\overline{C} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C}$$



X Fuse intact

1 Fuse blown

Assignments

- 6-9, 6-10, 6-12, 6-20