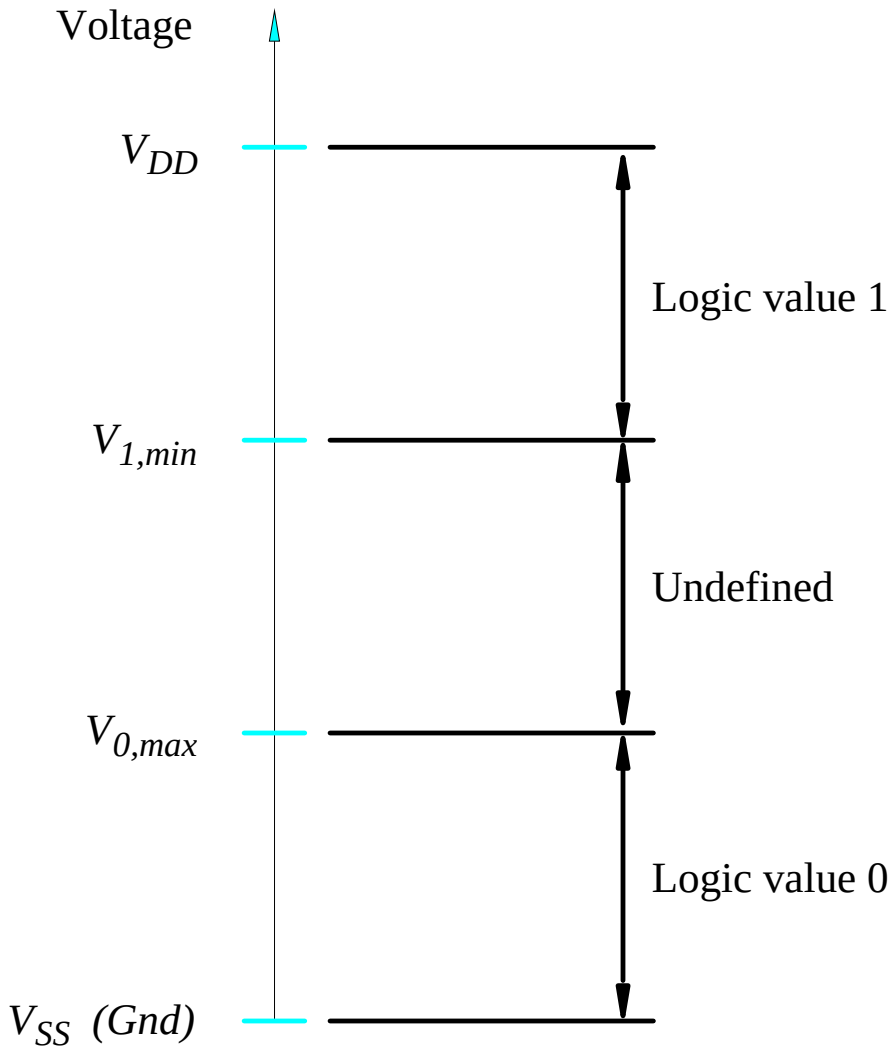
Integrated Circuits Technology (Instructor: Saraju P. Mohanty)

- Binary Logic
- Basic Logic Gates
- Operation of Transistors
- Tri-state Buffer and Transmission Gate
- Positive Logic Vs Negative Logic
- Integrated Circuits Categories
- Levels of Integration
- Digital Logic Families
- Characteristics of digital logic families
- Nomenclature of ICs
- Delay of real gates
- Implementation Approaches for Digital ICs
- Digital IC Design Flow
- Hierarchical Design
- Power Dissipation/Consumption in Logic Gates

Digital Systems: Information Representation

Digital system → manipulates *discrete elements of information* represented by physical quantities called signals (voltages, currents), mostly using two discrete values – binary signals. (HIGH – LOW, TRUE-FALSE, 1-0, positive / negative logic).



Logic values as voltage levels.

Why is Binary Used ??

- It is easy to distinguish between two states: high or low voltage, presence or absence of electric charge or a switch in the on or off position of a transistor.
- To reduce cost of electronic circuits.
- High noise margin compared to the multi-valued logic circuits.

Binary Logic

• Binary logic deals with binary variables (that assumes two discrete values) and with the operations applied to these variables.

•Binary variables are designated by alphabets, such as A, B, C,….., Z.

•The three basic logic operations are AND, OR, and NOT.

X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

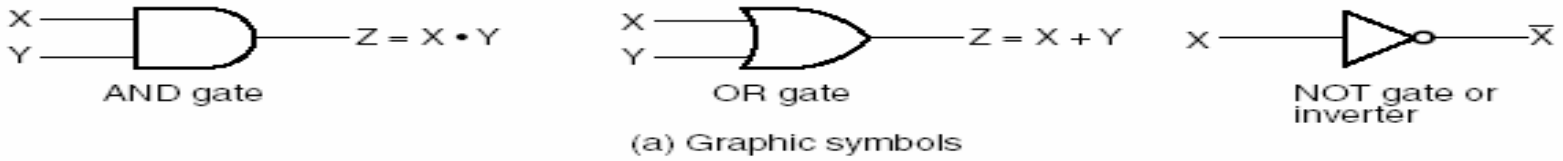
X	Y	$Z = X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

X	$Z = \overline{X}$
0	1
1	0

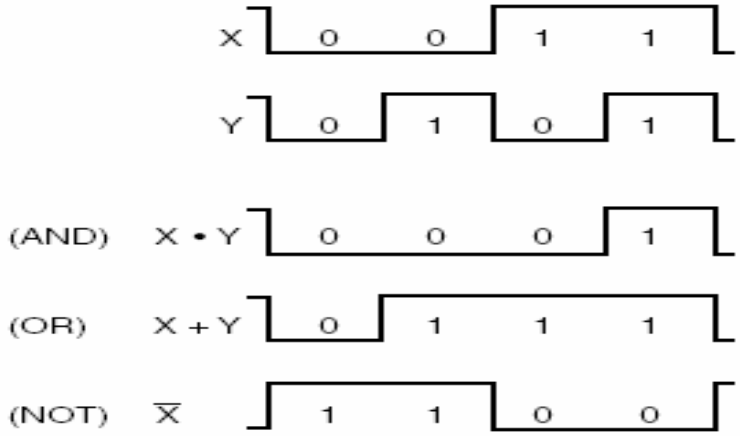
Table 2-1 Truth Tables for the Three Basic Logical Operations

Logic Gates

- Logic gates are electronic circuits that operate on one or more input signal to produce an output signal.
- Gates can be specified in three different ways: Graphic Symbols, Truth Tables, and Timing Diagrams.



(a) Graphic symbols

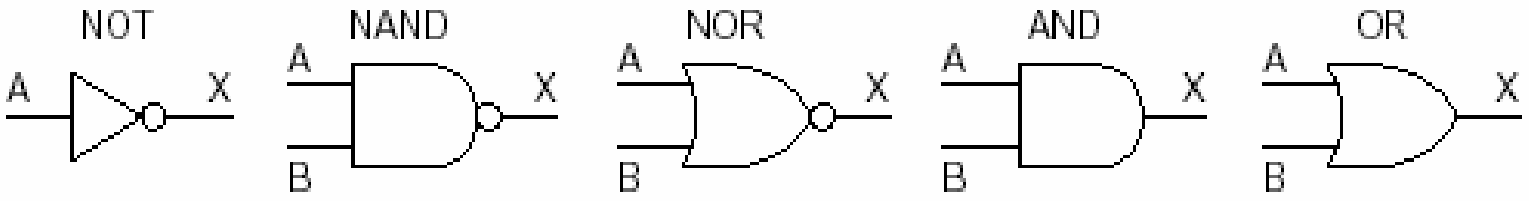


(b) Timing diagram

Fig. 2-1 Digital Logic Gates

- "Ideal" noise-free signals
- Horizontal axis : Time
- Vertical axis : Signal

Logic Gates: Basic Gates



A	X
0	1
1	0

(a)

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

Figure 3-2. The symbols and functional behavior for the five basic gates.

Logic Gates with more than two inputs

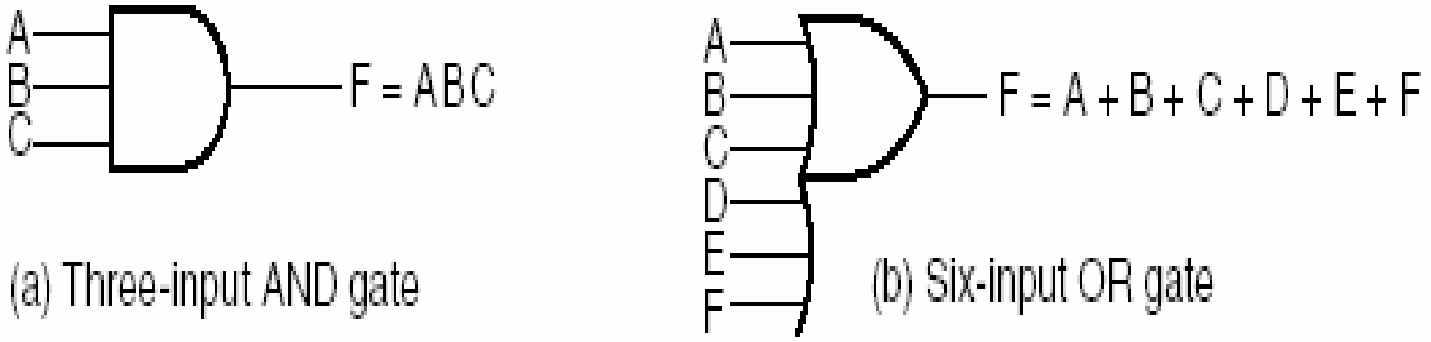
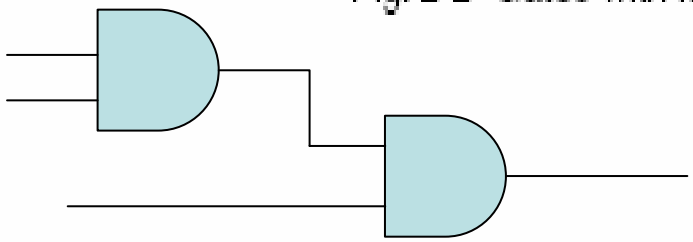


Fig. 2-2 Gates with More than Two Inputs



Alternative two input gate design for F=ABC

Logic Gates: More on gates…

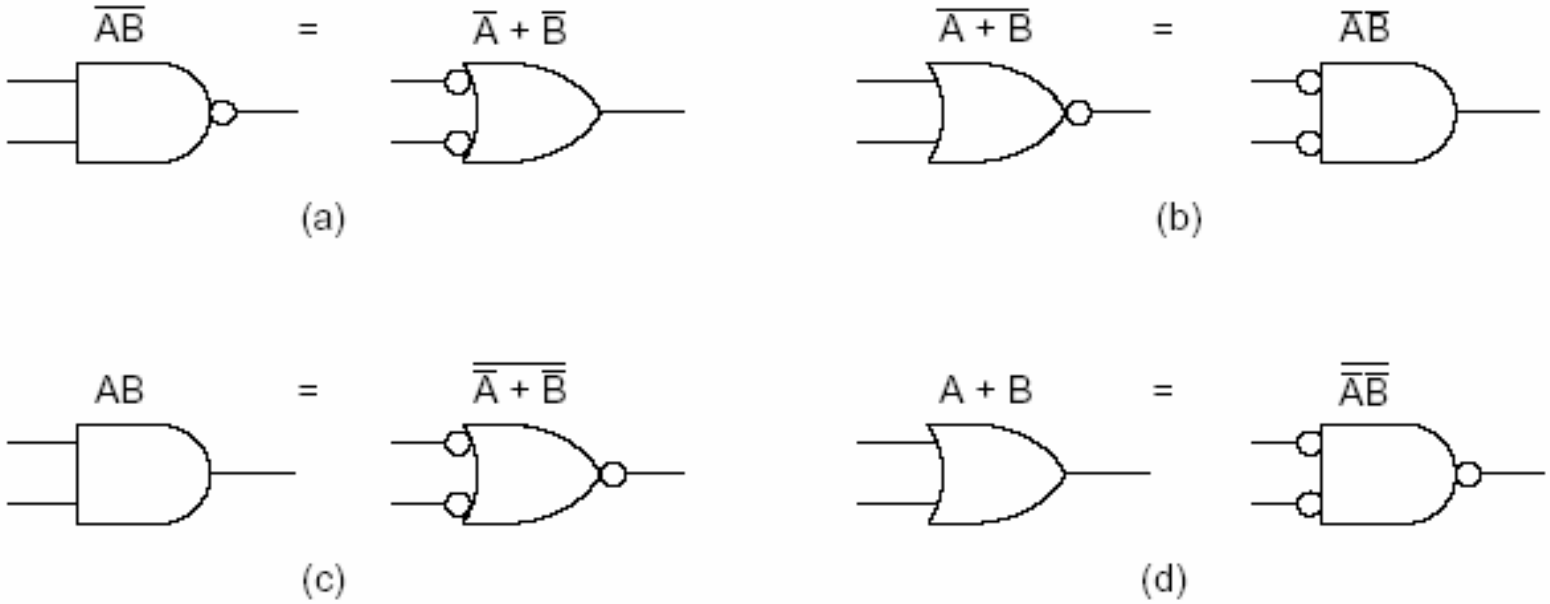
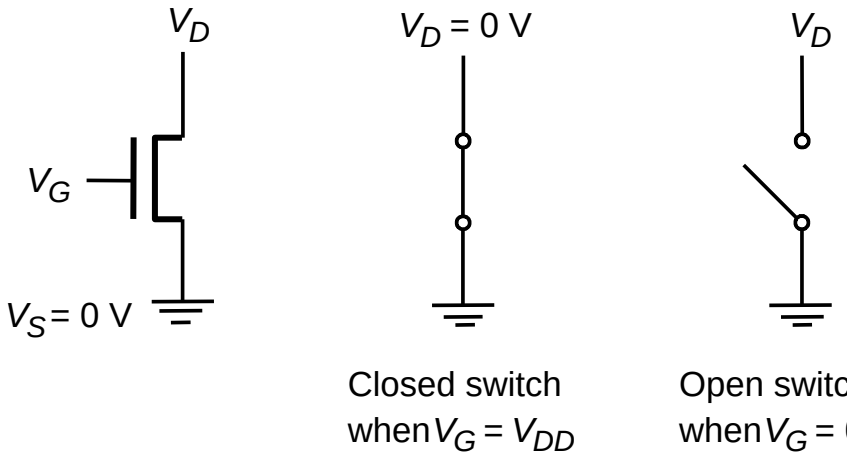


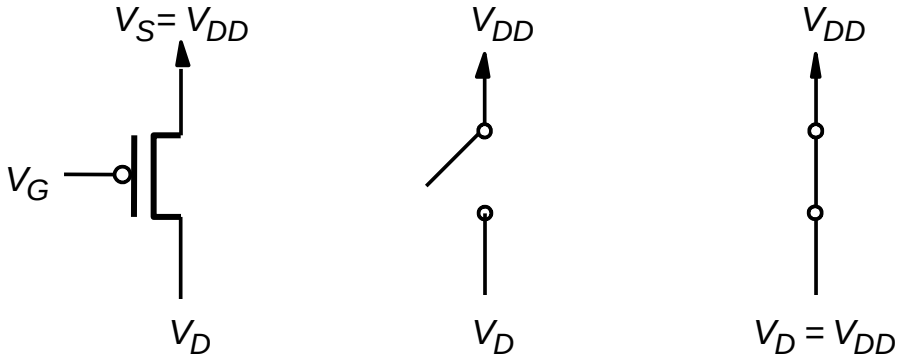
Figure 3-7. Alternative symbols for some gates: (a) NAND. (b) NOR. (c) AND. (d) OR.

NOTE: Operationally, **NOR = AND** and **NAND = OR**

Operation of Transistors: NMOS Vs PMOS

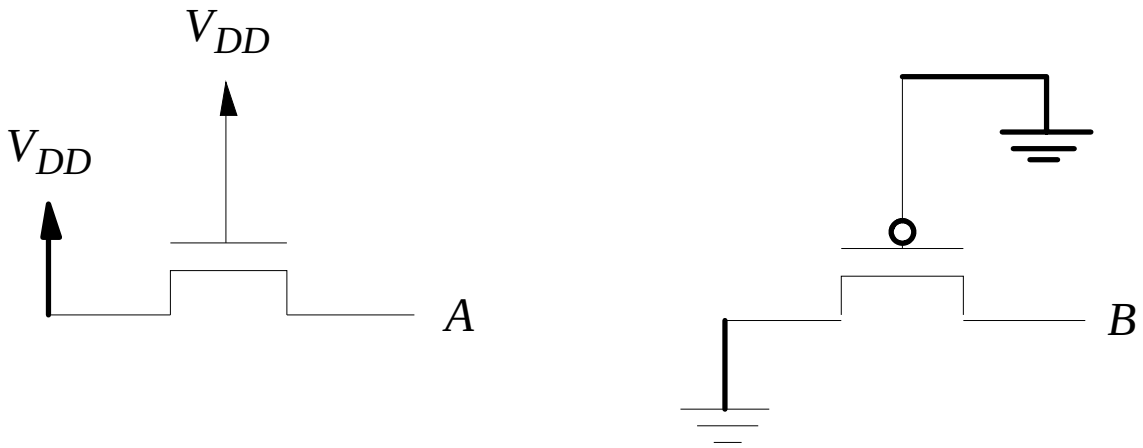


(a) NMOS transistor



(b) PMOS transistor

Passing 1 and 0 Through NMOS and PMOS

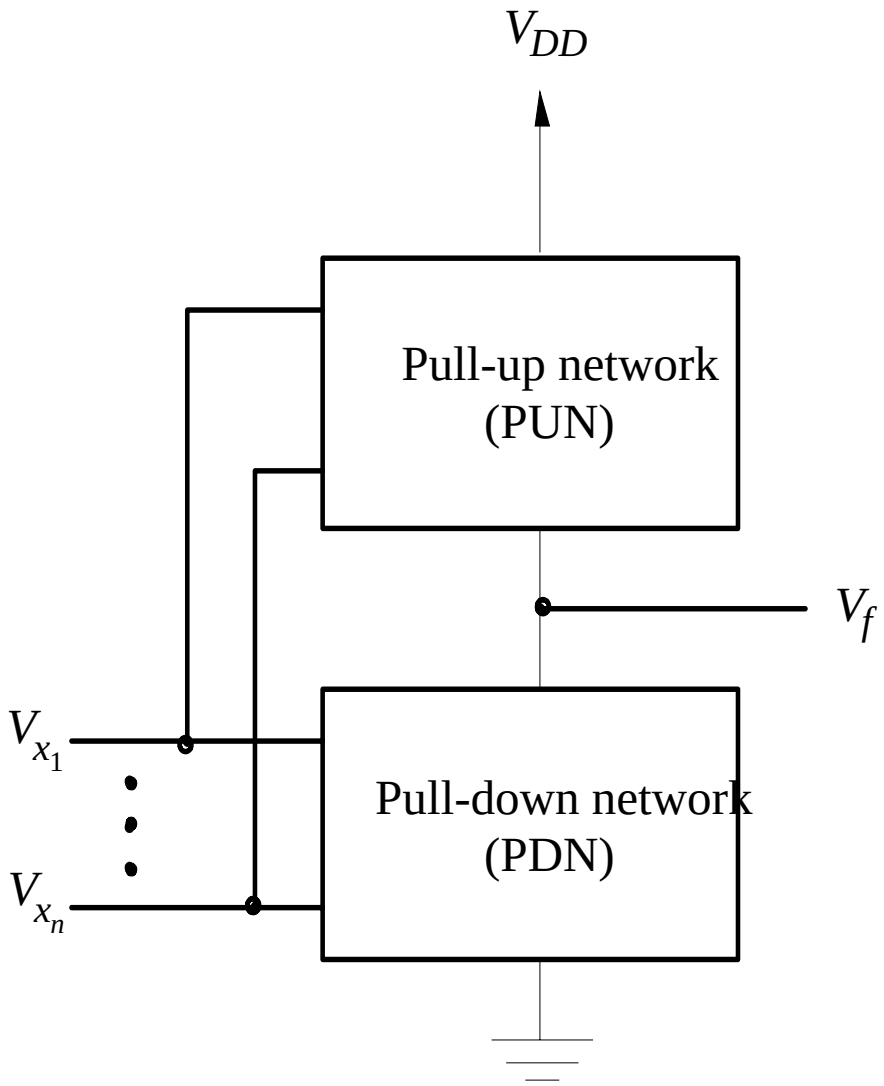


(a) NMOS transistor (b) PMOS transistor

•For NMOS, at steady state, VA = VDD – VT, but VA= VDD was ideally needed, thus not reproducing 1 properly.

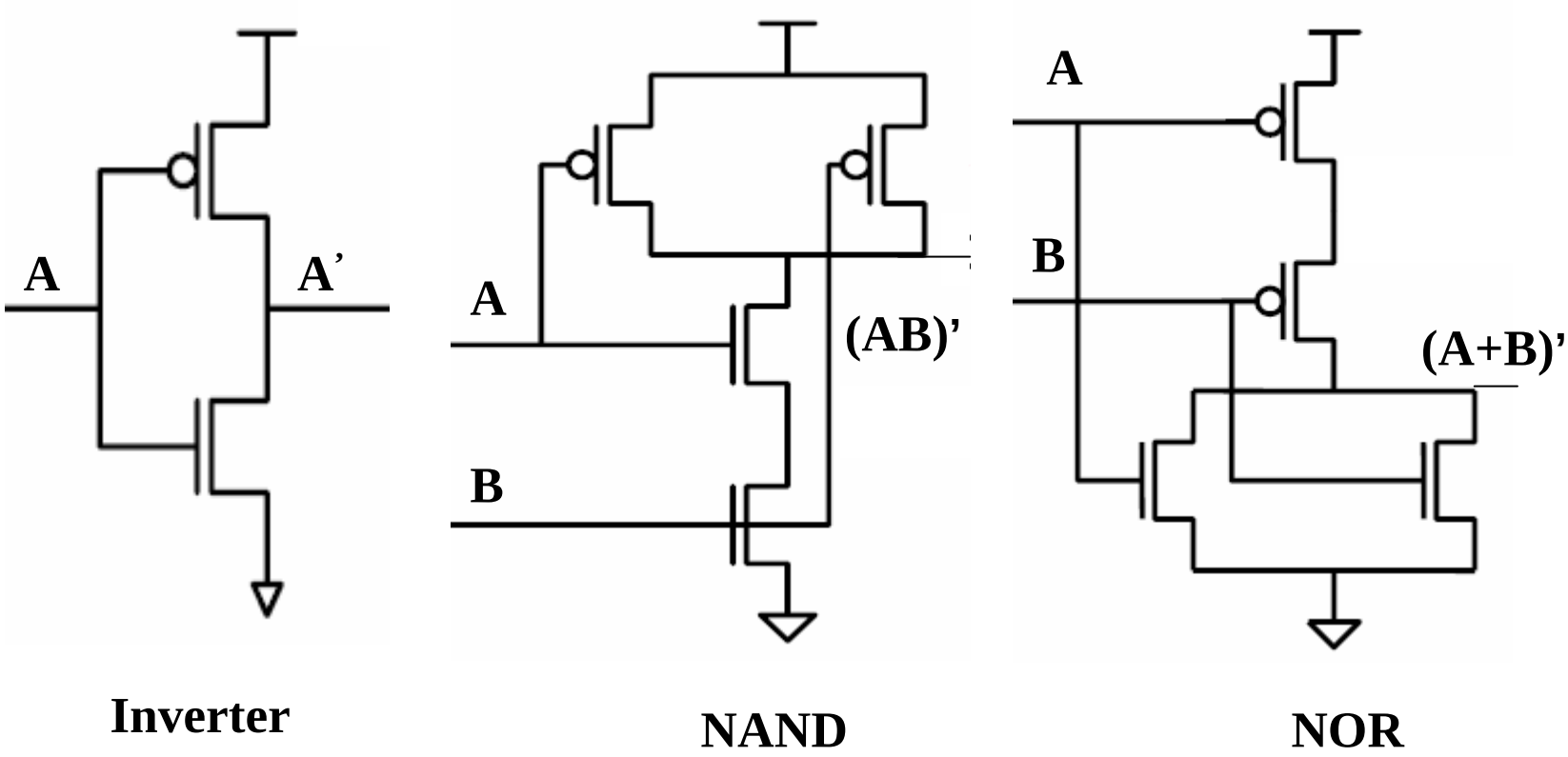
• For PMOS, at steady state, VB = VT, but VA= 0 Volt was ideally needed, thus not reproducing 0 properly.

CMOS Logic Gates



- PDN: NMOS network
- PUN: PMOS network
- PUN and PDN network are dual of each other.
- CMOS is necessary for faithful 0 and 1 logic.

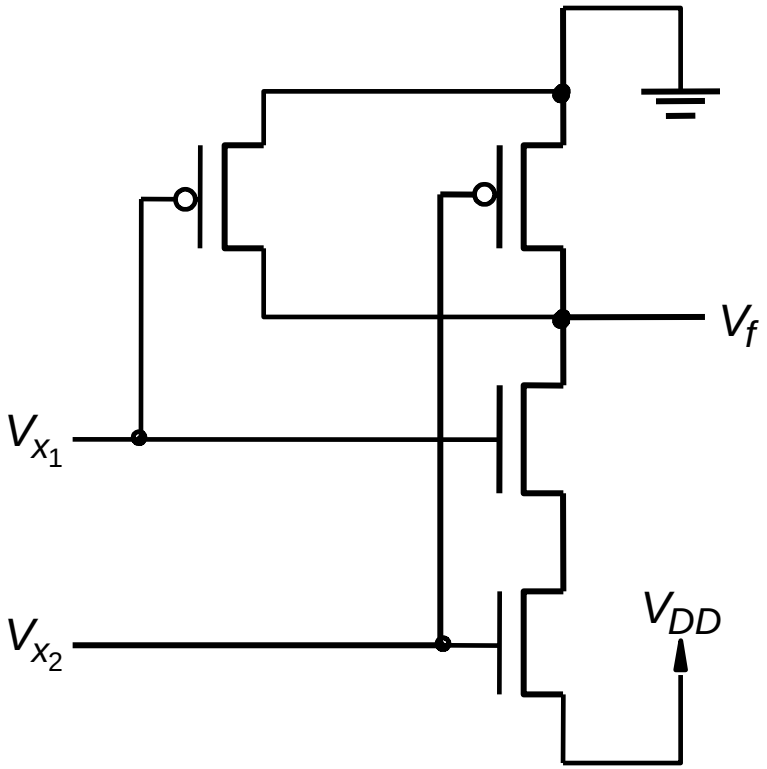
Gates at Transistor Level : CMOS FET



Inverter **NAND** **NOR**

CMOS FET : Complementary Metal Oxide Semiconductor Field Effect Transistor

A Poor implementation of a CMOS AND gate.

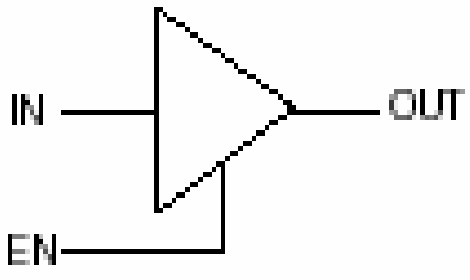


(a) An AND gate circuit

Logic value		Voltage	Logic value
x_1	x_2	V_f	f
0	0	1.5 V	0
0	1	1.5 V	0
1	0	1.5 V	0
1	1	3.5 V	1

(b) Truth table and voltage levels

Three-state buffers



(a) Logic symbol

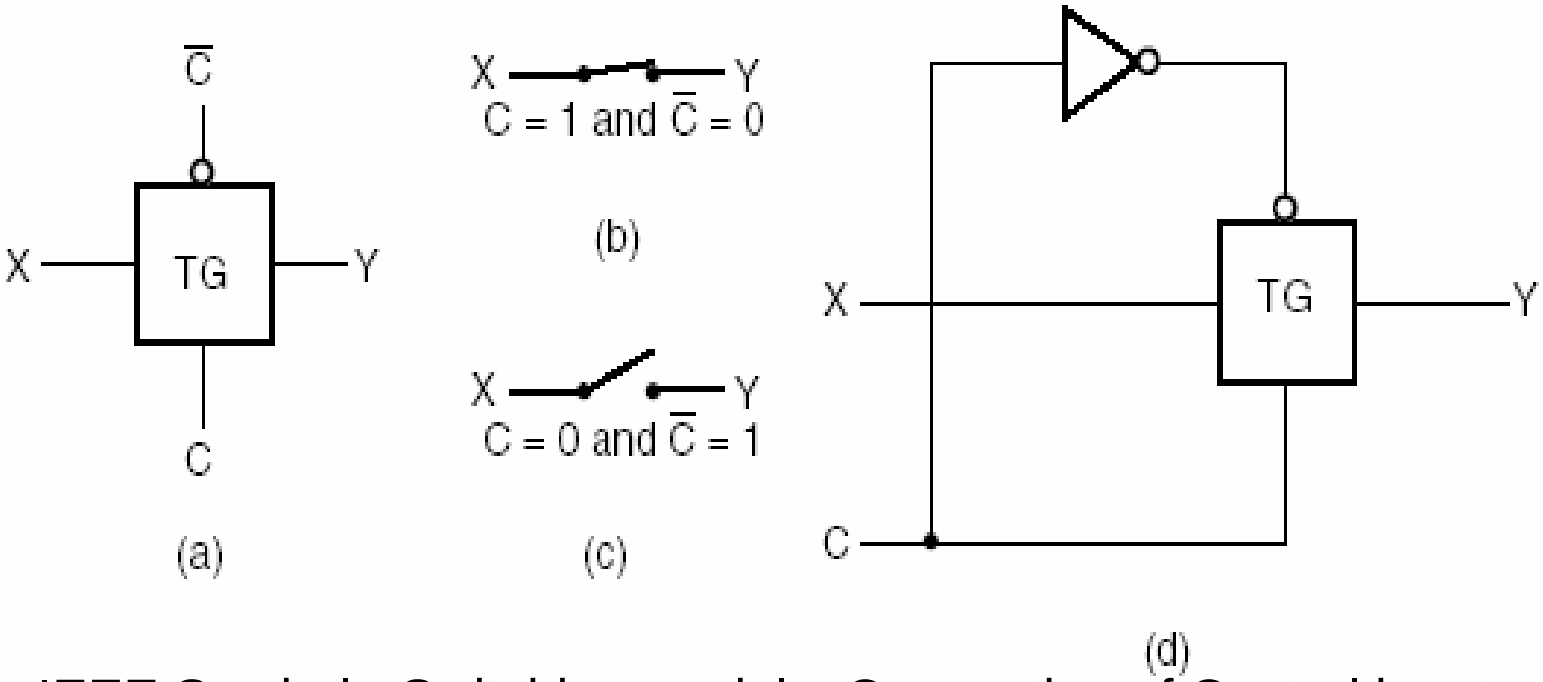
EN	IN	OUT
0	X	HI-Z
1	0	0
1	1	1

(b) Truth table

Fig. 6-8 Three-state Buffer

- Hi-Z is the hi-impedance state
- Hi-Z behaves like an open circuit → appears that the output is disconnected.
- When connected together can form a multiplexed line.

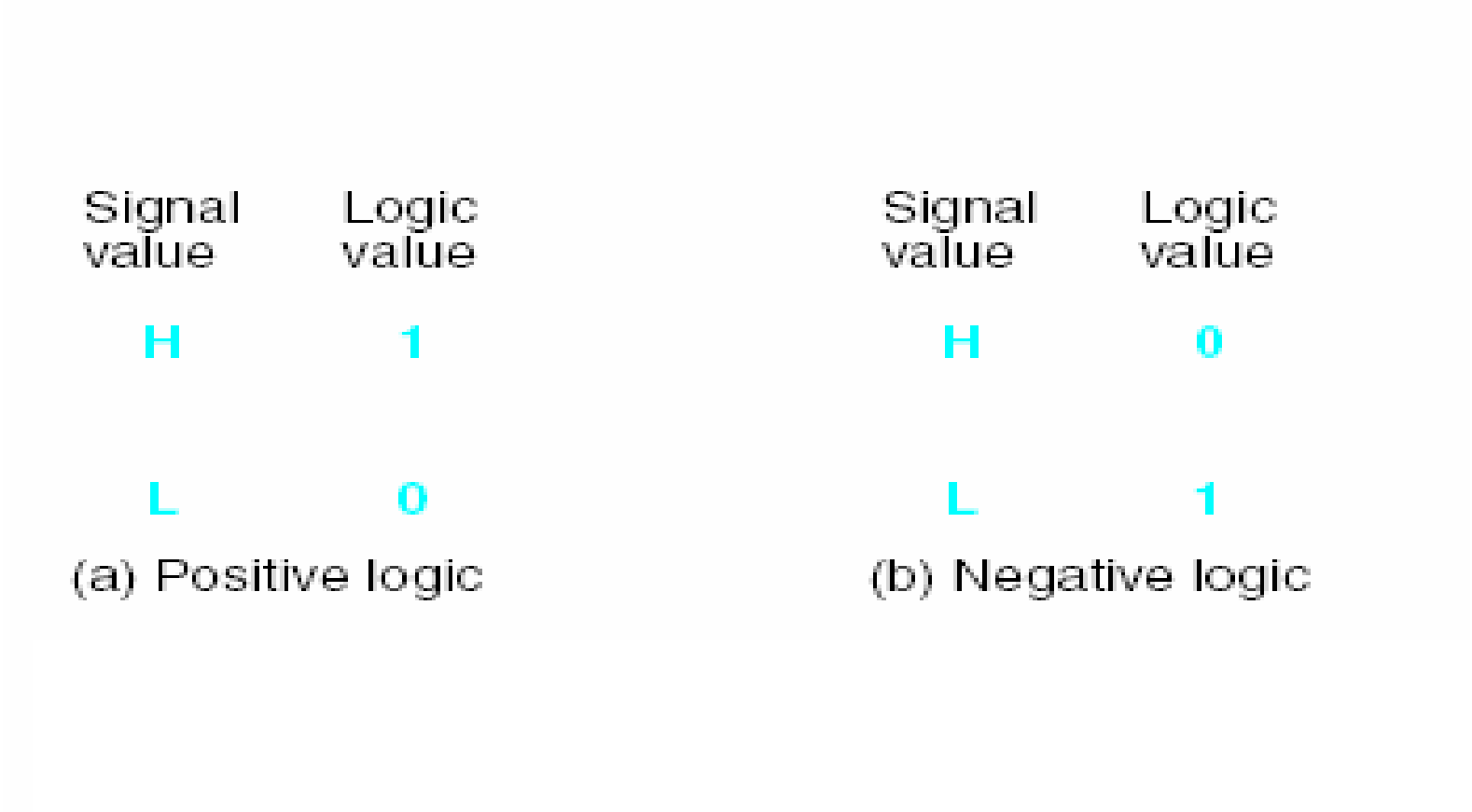
Transmission Gates



IEEE Symbol Switching model Connection of Control inputs

It is an electronic switch for connecting and disconnecting two points in a circuit.

Positive Vs Negative Logic



Positive Logic : High Level represent logic 1

Negative Logic: Low Level represent logic 1

Positive Vs Negative Logic (Formal ….)

- Positive Logic: The signal that activates the circuit (the 1 state) has a voltage level that is more POSITIVE than the 0 state.
- Negative logic: The signal that activates the circuit (the 1 state) has a voltage level that is more NEGATIVE than the 0 state

EXAMPLE 1	Active signal — TRUE, 1, HIGH	=	+10 volts
	Complement — FALSE, 0, LOW	=	0 volts
EXAMPLE 2	Active signal — TRUE, 1, HIGH	=	0 volts
	Complement — FALSE, 0, LOW	=	−10 volts

(Examples of positive logic)

Positive Vs Negative Logic …..

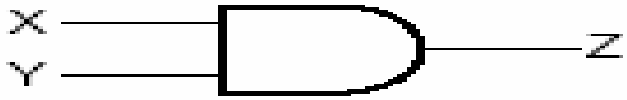
X	Y	Z
L	L	L
L	H	L
H	L	L
H	H	H

(a) Truth table with H and L

(b) Gate block diagram

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

(c) Truth table for positive logic



(d) Positive-logic AND gate

X	Y	Z
1	1	1
1	0	1
0	1	1
0	0	0

(e) Truth table for negative logic

(f) Negative-logic OR gate

Fig. 2-43 Demonstration of Positive and Negative Logic

Positive Vs Negative Logic : Observations

- Small triangles on the inputs and outputs are polarity indicators.
- Presence of polarity indicator implies negative logic.
- Same physical gate can operate as positive AND gate or negative OR gate.
- The actual signal values (such as 5.0V) do not determine the type of logic, rather the relative amplitude of two signals.
- Conversion of positive to negative logic or vice versa: change 1's to 0's and 0's to 1's at inputs and outputs.

What is an Integrated Circuit ?

- An integrated circuits is a silicon semiconductor crystal containing the electronic components for digital gates.
- Integrated Circuit is abbreviated as IC.
- The digital gates are interconnected to implement a Boolean function in a IC .
- The crystal is mounted in a ceramic/plastic material and external connections called “pins” are made available.
- ICs are informally called chips.

Integrated Circuits Categories

There are many different types of ICs as listed below.

IC Categories	**Functions**
Analog ICs	Amplifiers
	Filters
Digital ICs	Boolean Gates
	Encoders/Decoders
	Multiplexers / Demultiplexers
	Flip-flops
	Counters
	Shift Registers
Hybrid ICs	Mixed Signal Processors
Interface ICs	Analog-Digital Converters
	Digital-Analog Converters

Levels of Integration (Chip Complexity)

Categorized by the number of gates contained in the chip.

IC Complexity	**Number of Gates**	**Functional Complexity**	**Examples**
SSI	<10	Basic gates	Inverters, AND gates, OR gates, NAND gates, NOR gates
MSI	10-100	Basic gates	Exclusive OR/NOR
		Sub-modules	Adders, subtractors, encoders, decoders, multiplexers, demultiplexers, counters, flip-flops
LSI	100-1000s	Functional modules	Shift registers, stacks
VLSI	1000s-100,000	Major building blocks	Microprocessors, memories
ULSI	>100,000	Complete systems	Single chip computers, digital signal processors
WSI	>10,000,000	Distributed systems	Microprocessor systems

Levels of Integration (Chip Complexity) …..

- ICs with less than 10 equivalent gates are grouped together and identified as Small Scale Integration (SSI). Chips in this group are mostly simple gates used as glue logic for tying larger components.
- ICs with 10 to100 equivalent gates are grouped together and identified as Medium Scale Integration (MSI). Chips in this group are simple modules performing a specific elementary function.
- ICs with 100 to few thousand equivalent gates are grouped together and identified as Large Scale Integration (LSI). Chips in this group are also functional modules.
- ICs with several thousand to millions gates are termed Very Large Scale Integration (VLSI), and Ultra Large Scale Integration (ULSI). Chips of these types include microprocessors, microcontrollers, etc.
- Wafer Scale Integration (WSI) refers to devices with many VLSI components connected within a single wafer. This is ideal for constructing devices with many parallel processors

IC Nomenclature

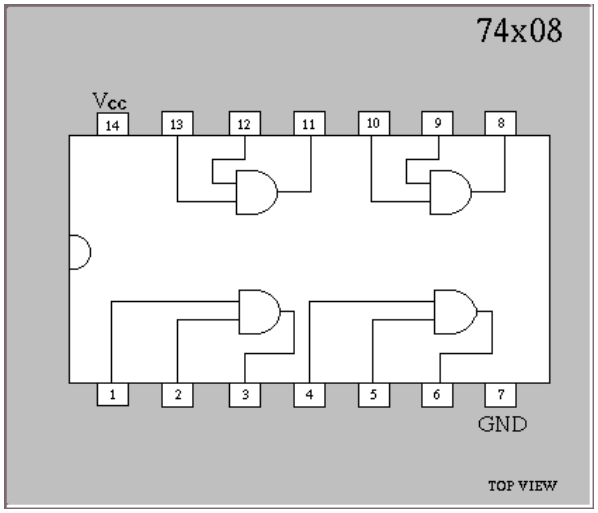
Most chips are identified in a similar way regardless of vendors. Adoption of this notation allows user to select chips with similar functions from a variety of vendors.

- Manufacturer Code:
  (i) SN - Texas Instruments, (ii) DM - National Semiconductor, (iii) MC – Motorola, (iv) MM - Monolithic Memories
- Product Line:
  (i) 5 - Military grade TTL products (40C to 100C)
  (ii) 7 - Commercial grade TTL products (0C to 70C), ….., and so on.
- Performance:
  (i) (blank) - Regular, (ii) H - High-speed, (iii) L - Low-power, (iii) S –Schottky, (iv) LS - Low-power Schottky, (v) AS - Advanced Schottky, (vi) ALS - Advanced low-power Schottky
- Logic Family:
  (i) 00 - Quad 2-input positive NAND, (ii) 02 - Quad 2-input positive NOR
  (iii) 0 - Hex inverters, (iv) 08 - Quad 2-input positive AND
- Package Type:
  (i) J - Ceramic Dual-In-Line (DIP), (ii) N - Plastic DIP, (iii) T -Tin can, (iv) W - Ceramic flat package

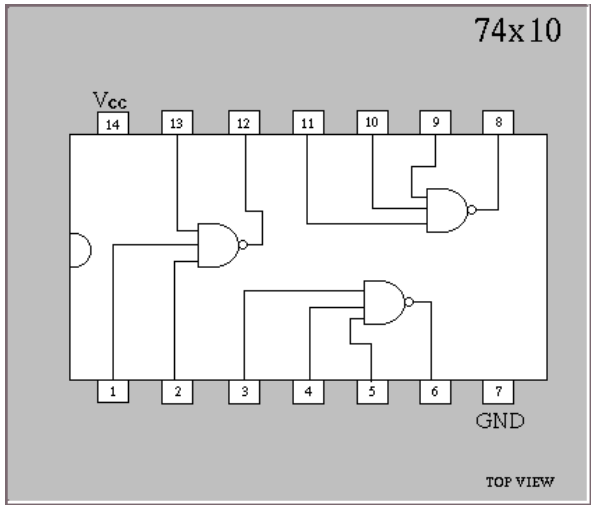
IC Nomenclature : Example ….

	Manufacturer Code	Product Line	Performance Class	Logic Family	Package Type
1	SN	74	LS	00	N
2	MC	54	S	190	T

Chip 1: Texas Instruments, Low-power Schottky, Quad 2-input positive NAND, Plastic DIP, and Chip 2: Motorola, Schottky, Tin can



Qaud 2-input Positive AND Gates



Triple 3-input Positive NAND Gates

Digital Logic Families

- Various circuit technology used to implement an IC at lower level of abstraction.
- The circuit technology is referred to as a digital logic family.

RTL	Resistor-transistor Logic
DTL	Diode-transistor logic
TTL	Transistor-transistor logic
ECL	Emitter-coupled logic
MOS	Metal-oxide semiconductor
CMOS	Complementary Metal-oxide semiconductor
BiCMOS	Bipolar Complementary Metal-oxide semiconductor
GaAs	Gallium-Arsenide

Digital Logic Families ……

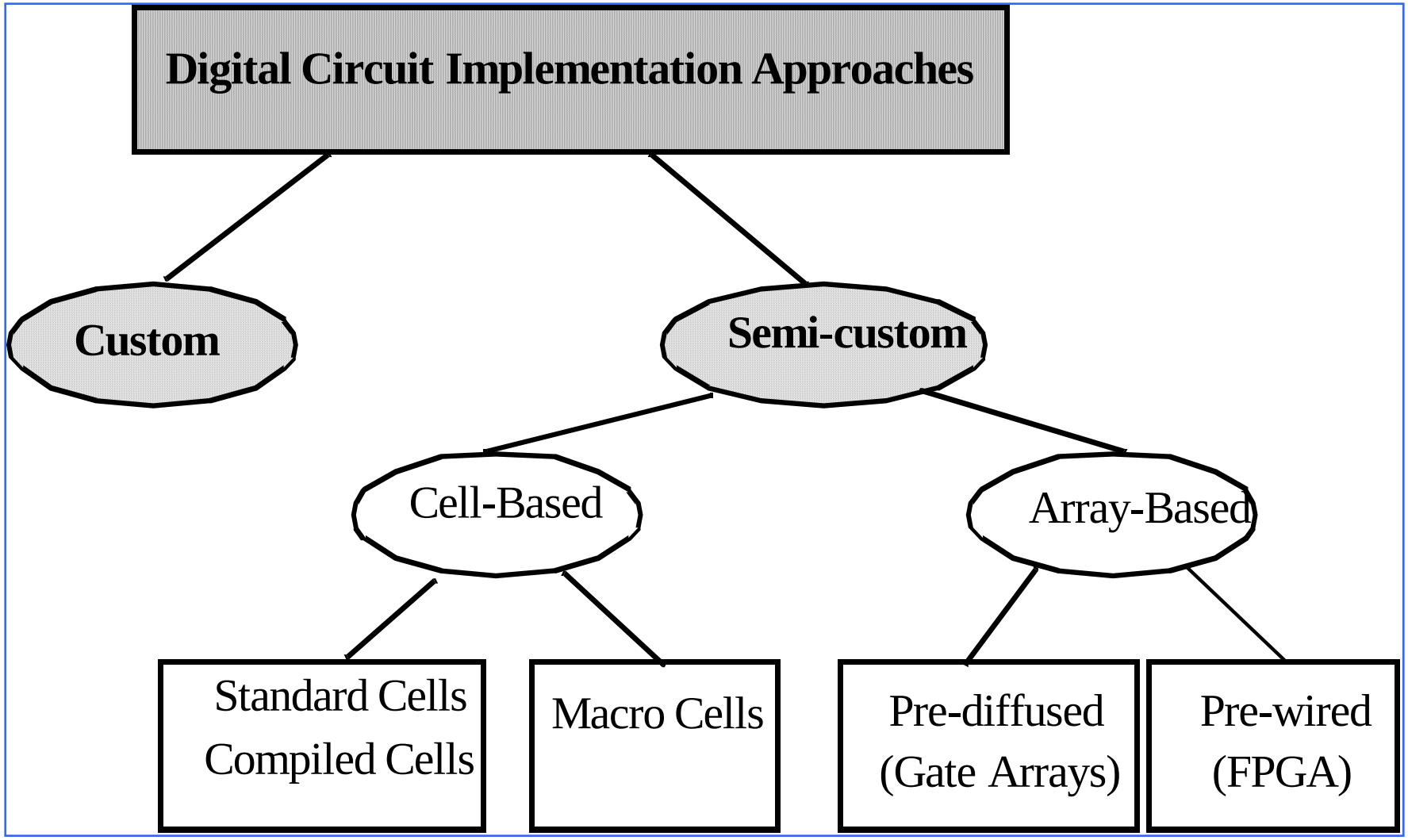
- RTL, DTL, and obsolete.
- TTL are not much used now-a-days.
- ECL can provide high-speed ICs.
- MOS is suitable for high-component density.
- CMOS is preferred for low-power high-performance and high-packing density IC implementation.
- CMOS is widely used technology at present.
- BiCMOS is used where high current and high-speed is necessary.
- GaAs is used for very high speed circuits.

Characteristics of Logic Families

The characteristics of a logic family is expressed in terms of the parameters, such as Fan-in, Fan-out, Noise-Margin, Power Dissipation, etc.

- Fan-in: the number of inputs available on a gate.
- Fan-out: the number of standard loads that the output of a typical gate can drive without impairing its performance.
- Noise-Margin: the maximum external noise voltage superimposed on a normal input voltage value that will not cause an undesirable change in the circuit output.
- Power Dissipation: Power consumed / dissipated by a gate.
- Propagation delay: the time for change in value of a signal to propagate from input to output.

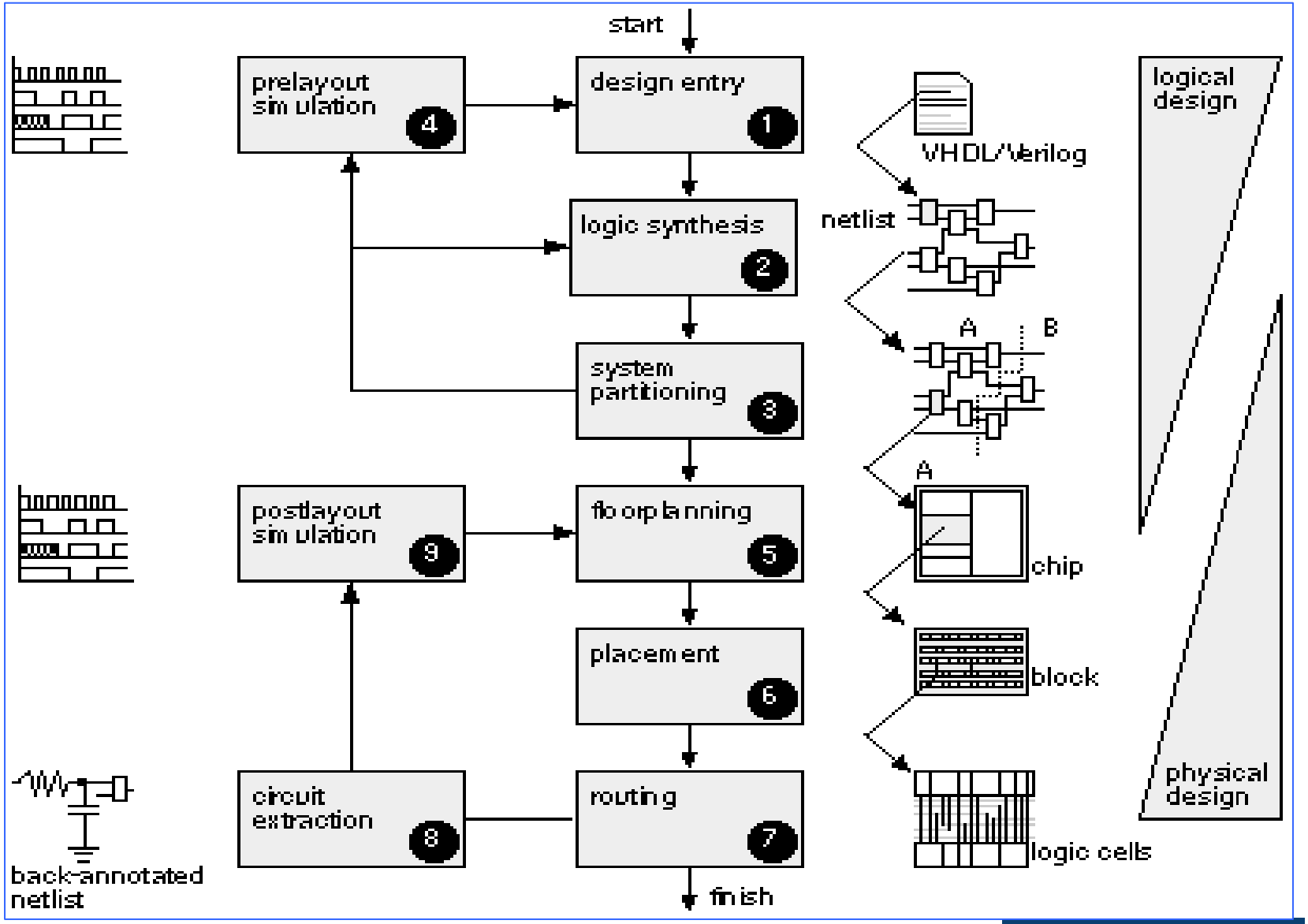
Implementation Approaches for Digital ICs



Implementation Approaches for Digital ICs

- Full-custom: all logic cells are customized. A general purpose microprocessor is designed this way.
- Semi-custom: all of the logic cells are from predesigned cell libraries (reduces the manufacture lead time of the IC)
- Standard-cell based IC uses predesigned logic cells such as AND gates, OR gates, MUXs, FFs,.., etc.
- Macrocells (also called megacells) are larger predesigned cells, such as microcontrollers, even microprocessors, etc.
- Gate-Array, Sea-of-Gates or prediffused arrays contains array of transistors or gates which can be connected by wires to implement the chip.
- Programmable-Logic-Array (PLA) is an example of fuse-based FPGA design. (NOTE: Fuse-based, nonvolatile and volatile are three types of FPGAs)

Digital IC Design Flow



Digital IC Design Flow

- Design entry : Enter the design into an ASIC design system, either using a hardware description language (HDL) or schematic entry .
- Logic synthesis : Use an HDL (VHDL or Verilog) and a logic synthesis tool to produce a netlist—a description of the logic cells and their connections.
- System partitioning : Divide a large system into ASIC-sized pieces.
- Prelayout simulation : Check to see if the design functions correctly.
- Floorplanning : Arrange the blocks of the netlist on the chip.
- Placement : Decide the locations of cells in a block.
- Routing : Make the connections between cells and blocks.
- Extraction : Determine the resistance and capacitance of the interconnect.
- Postlayout simulation : Check to see the design still works with the added loads of the interconnect.

Digital IC Design Flow : Tools Used

Tools	Purpose
Cadence NClaunch	VHDL simulator
Synopsys Design Analyzer	Verilog netlist generation
Cadence Silicon Ensemble	Layout, Placement and routing
Cadence Virtuose tool	Layout Editing
Cadence Abstract Generator	Abstract generation
Synopsys Nanosim	Power and delay calculations

Design Flow Example: VHDL Code

File Edit Window Tools Syntax Help

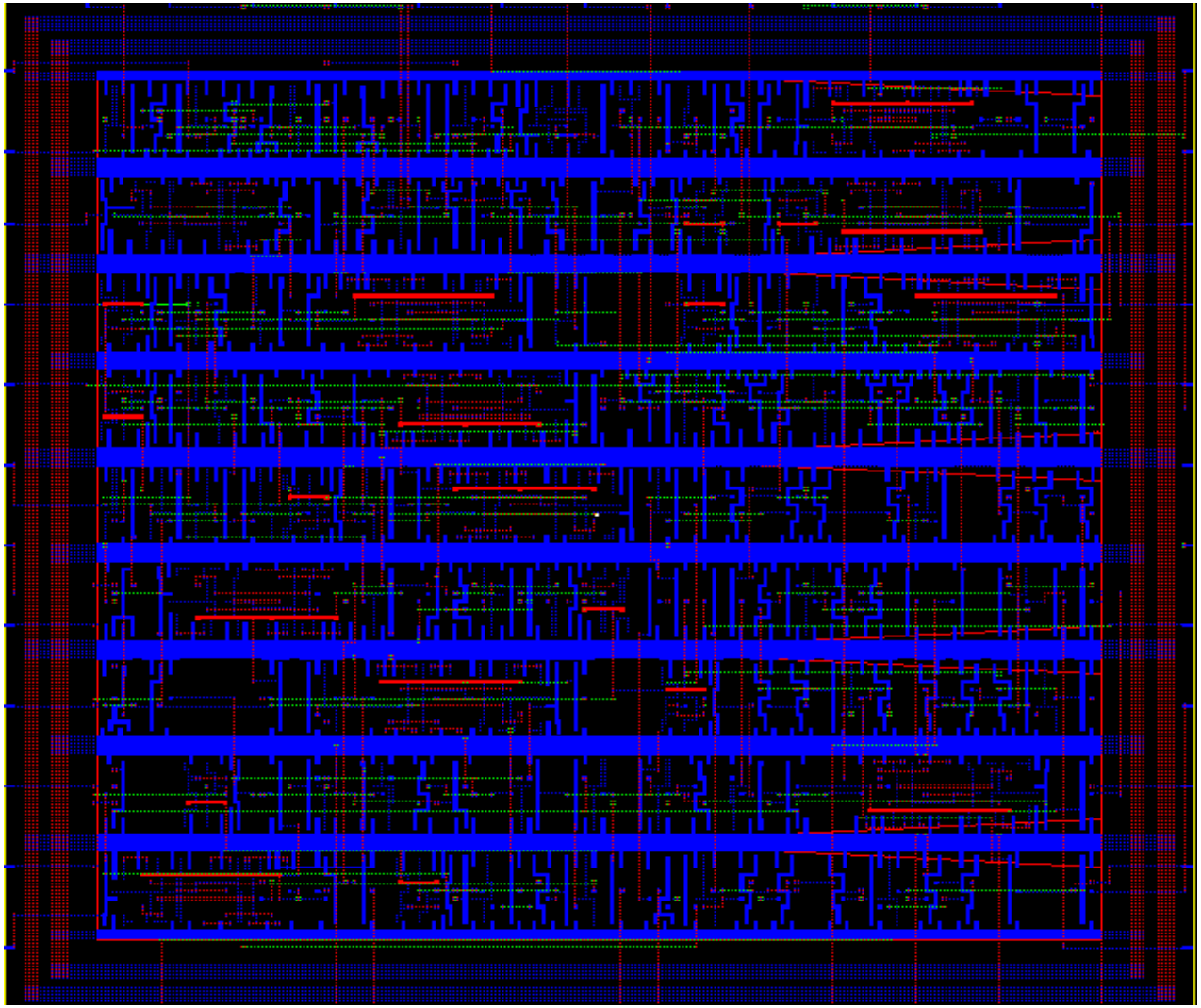
```
entity edm3 is
port (clk, reset, vdd1, enable, vss1 : in std_logic;
      AnMax, An : in std_logic_vector(16 downto 0);
      edge_block, done, write_edm3 : out std_logic;
      countout : out std_logic_vector(7 downto 0)
      );
end entity edm3;
--------------------------------------------------------------------------------
-

architecture behav of edm3 is

component counter8 is
port (clk : in std_logic;
      reset, vdd1, enable : in std_logic;
      q   : inout std_logic_vector(7 downto 0)
      );
end component counter8;

signal An_max, AnMax_by_2 : std_logic_vector(16 downto 0);
signal count_out : std_logic_vector(7 downto 0);
signal At, A, B, Bt, count, edgeblock, en_count, write, proces,
tempedgeblock : std_logic;

begin

COUNTER: counter8 port map (clk=>clk, reset=>reset, enable=>write, vdd1=>vdd1,
 q=>count_out);

countout<=count_out;

counting: process(count_out) is
          begin
          if (count_out="11111111") then
          count<='1';
          else
          count<='0';
          end if;
          end process;


--------------------------------------------------------------------------------
-
```

Design Flow Example: Synthesized Netlist

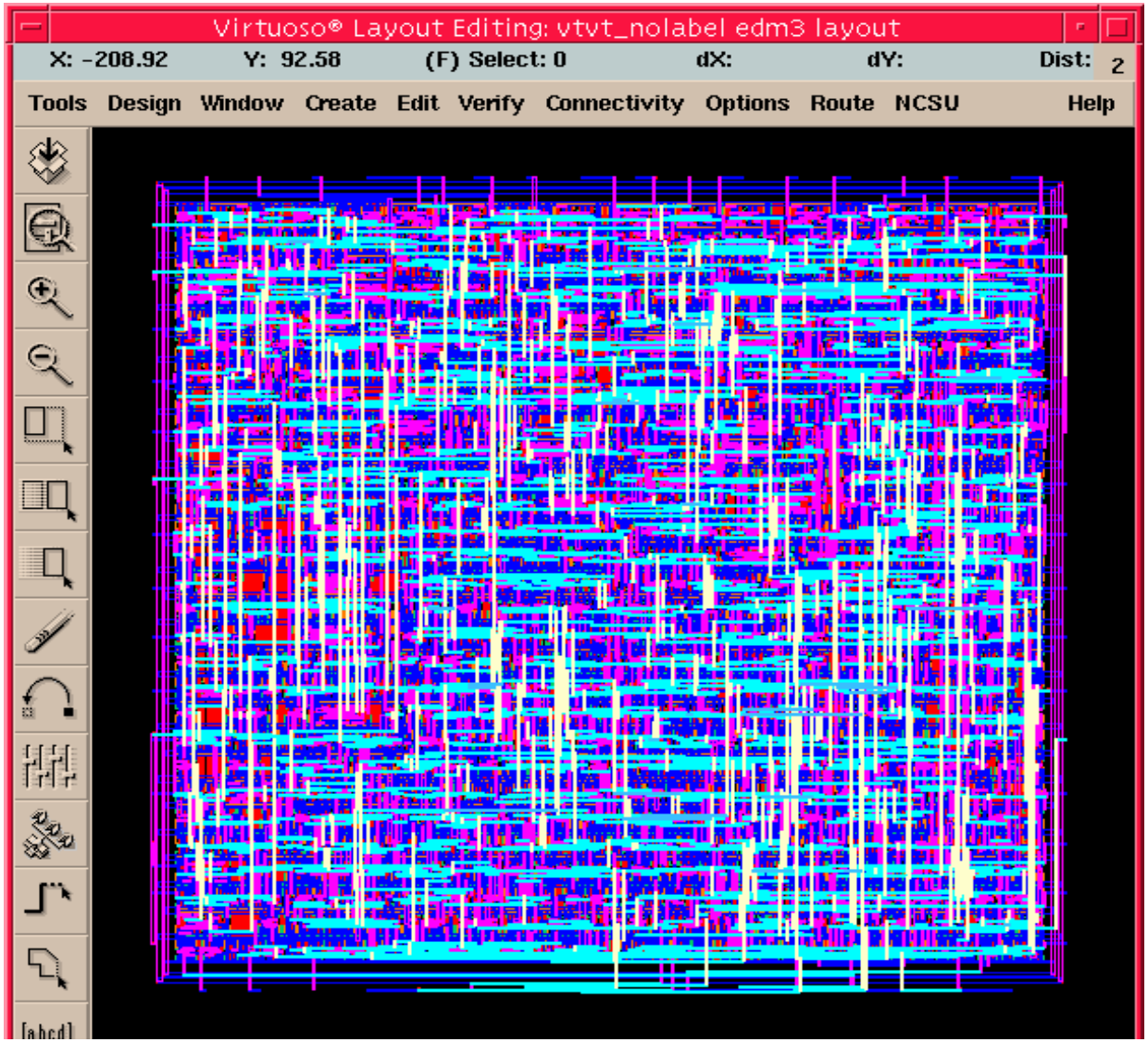
File Edit Window Tools Syntax Help

```
module edm3 ( clk, reset, vdd1, enable, vss1, AnMax, An, edge_block, done,
    write_edm3, countout );
input  [16:0] An;
input  [16:0] AnMax;
output [7:0] countout;
input  clk, reset, vdd1, enable, vss1;
output edge_block, done, write_edm3;
    wire Bt, n_133, At88, n_134, At, \"<"-return148 , count, Bt100, n195, n196,
        n197, n198, n199, n200, n201, n202, n203, n204, n205, n206,
        \*cell*78/U5/Z_0 ;
    counter8 COUNTER ( .clk(clk), .reset(reset), .vdd1(vdd1), .enable(
        write_edm3), .q(countout) );
    and3_1 U39 ( .ip1(n195), .ip2(n_133), .ip3(n196), .op(\*cell*78/U5/Z_0 )
        );
    or2_1 U40 ( .ip1(n197), .ip2(n198), .op(At88) );
    and2_1 U41 ( .ip1(\"<"-return148 ), .ip2(n_133), .op(n_134) );
    inv_1 U42 ( .ip(reset), .op(n_133) );
    nand2_1 U43 ( .ip1(n199), .ip2(n200), .op(Bt100) );
    nor2_1 U44 ( .ip1(n201), .ip2(n199), .op(n197) );
    nor2_1 U45 ( .ip1(n198), .ip2(n201), .op(n202) );
    nor3_1 U46 ( .ip1(n203), .ip2(n204), .ip3(n205), .op(count) );
    nand2_1 U47 ( .ip1(At), .ip2(n_133), .op(n199) );
    mux2_2 U48 ( .ip1(n202), .ip2(n198), .s(n199), .op(write_edm3) );
    mux2_2 U49 ( .ip1(n201), .ip2(enable), .s(n199), .op(n195) );
    and2_1 U50 ( .ip1(countout[1]), .ip2(countout[3]), .op(n206) );
    nand3_1 U51 ( .ip1(countout[0]), .ip2(countout[2]), .ip3(n206), .op(n205)
        );
    nand2_1 U52 ( .ip1(countout[5]), .ip2(countout[4]), .op(n203) );
    nand2_1 U53 ( .ip1(countout[7]), .ip2(countout[6]), .op(n204) );
    inv_1 U54 ( .ip(count), .op(n201) );
    nand2_1 U55 ( .ip1(Bt), .ip2(n_133), .op(n196) );
    inv_1 U56 ( .ip(n196), .op(n198) );
    nand2_1 U57 ( .ip1(enable), .ip2(n196), .op(n200) );
    drp_2 At_reg ( .ck(clk), .ip(At88), .rb(n_133), .q(At) );
    lp_2 edgeblock_reg ( .ck(\*cell*78/U5/Z_0 ), .ip(n_134), .q(edge_block) );
    dp_2 done_reg ( .ck(clk), .ip(count), .q(done) );
    drp_2 Bt_reg ( .ck(clk), .ip(Bt100), .rb(n_133), .q(Bt) );
    edm3_DW01_cmp2_17_0 \lt_100/lt/lt  ( .A(An), .B({vss1, AnMax[16],
        AnMax[15], AnMax[14], AnMax[13], AnMax[12], AnMax[11], AnMax[10],
        AnMax[9], AnMax[8], AnMax[7], AnMax[6], AnMax[5], AnMax[4], AnMax[3],
        AnMax[2], AnMax[1]}), .LEQ(1'b0), .TC(1'b0), .LT_LE(\"<"-return148 )
        );
endmodule
```

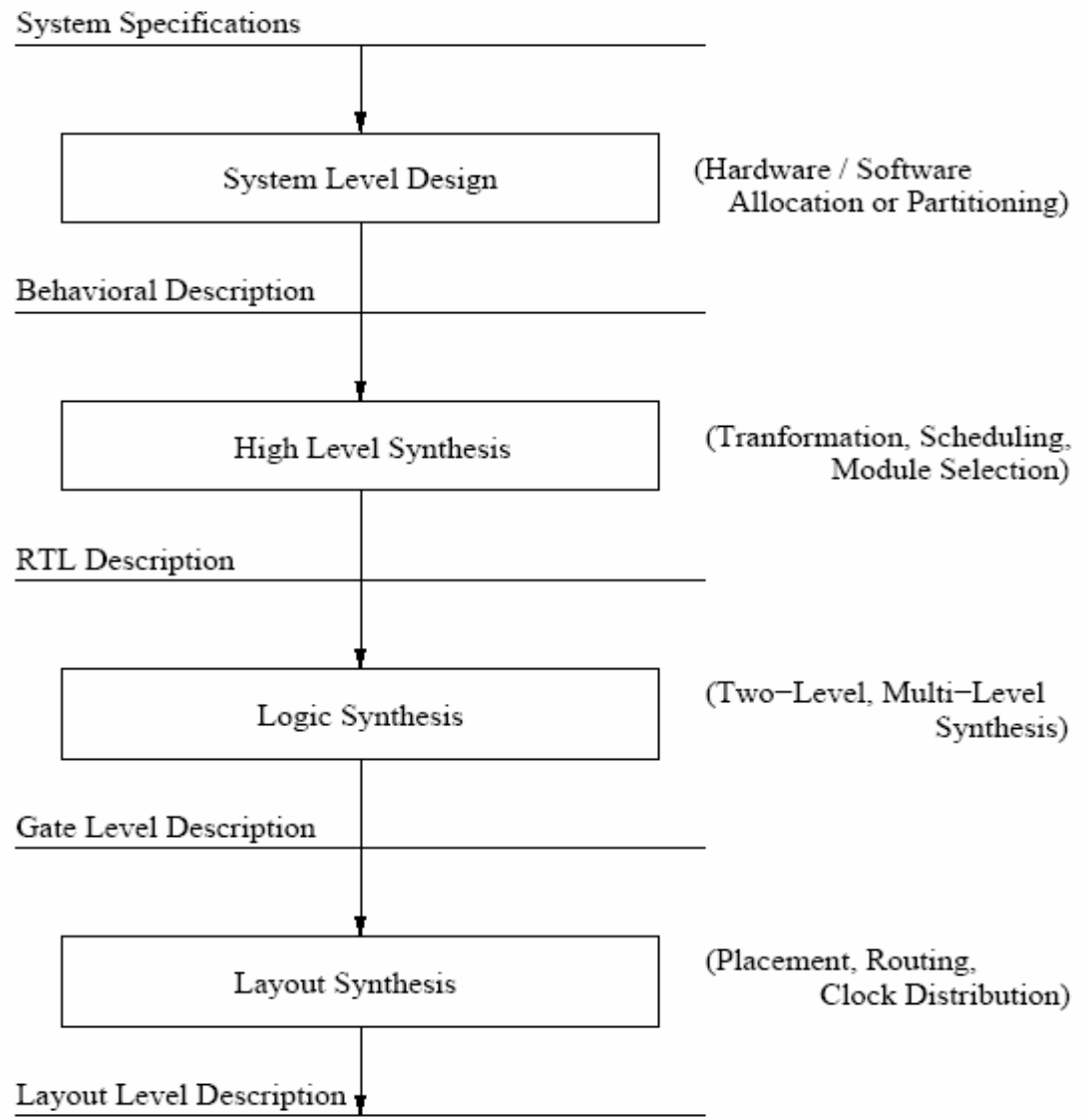
Design Flow Example: Placement and Routing



Design Flow Example: Layout



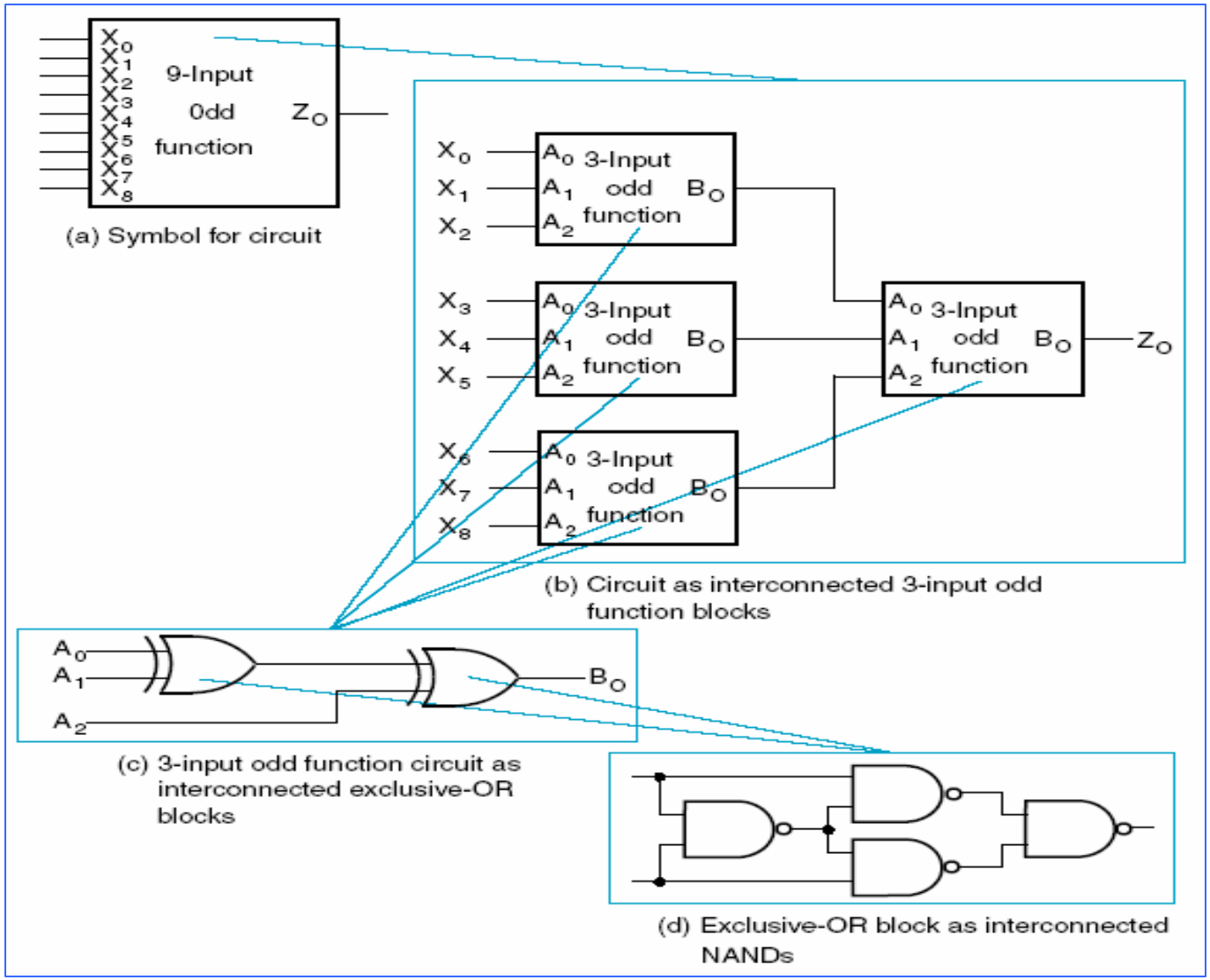
Synthesis Flow: A Complete Picture



Hierarchical Design of Digital ICs

- In order to deal with large and complex circuits design-and-conquer approach is used.
- The design is broken into blocks and the blocks are connected to get the overall chip.
- The design-and-conquer approach is referred to as hierarchical design.
- There are several advantages of hierarchical design.
  - Reduction of complexity of the design
  - At the lowest level of hierarchy a program or description can serve as model and schematics not needed.
  - Block reusability

Hierarchical Design of Digital ICs : Example



Hierarchical Design of Digital ICs : Example

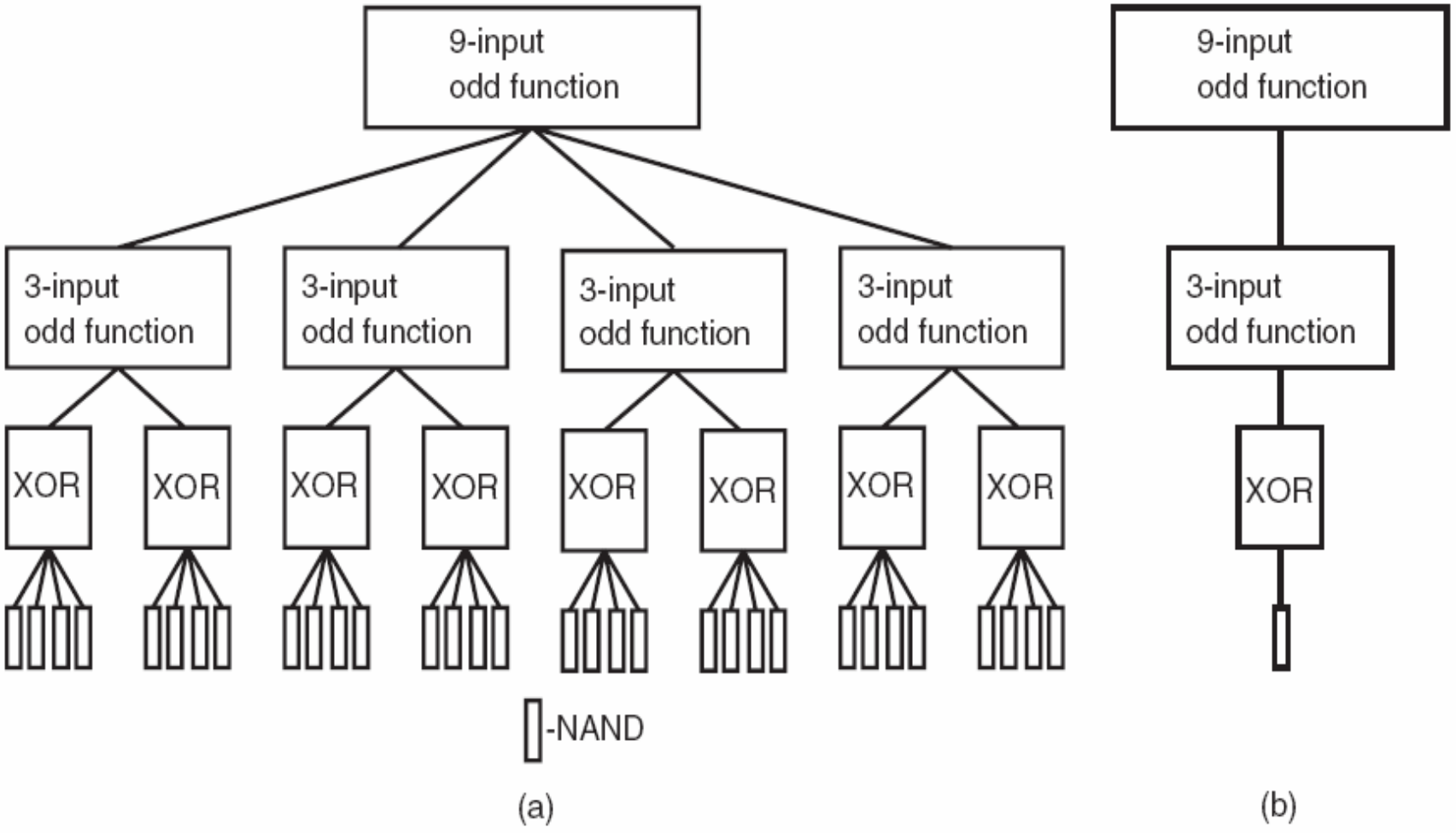


Fig. 3-3 Diagrams Representing the Hierarchy for Figure 3-2

Ideal Vs Real : Practical Consideration

- The ideal gates are perfect switches that provide the output at the same instance when the input is applied.

- In practice, a gate does have a finite nonzero delay.

- The three propagation delay parameters are :
  - high-to-low propagation time (t_{PHL})
  - low-to-high propagation time (t_{PLH})
  - (average) propagation delay (t_{pd})

- Two delay models used during simulation:
  - transport delay
  - inertial delay

Real Gates: Propagation delay

- High-to-low propagation time (t_{PHL}): time delay from the reference voltage at the input to the reference voltage at the output, when output voltage is going from high-to-low.

- Low-to-high propagation time (t_{PLH}): time delay from the reference voltage at the input to the reference voltage at the output, when output voltage is going from low-to-high.

- (Average) propagation delay (t_{pd}): defined as either maximum of t_{PHL} and t_{PLH} or average of the two.

- Usually 50% point on the voltage signal is used as reference, but other references are possible.

Real Gates : Propagation delay ……

NOTE : The 50% point on the voltage signal is used as reference.

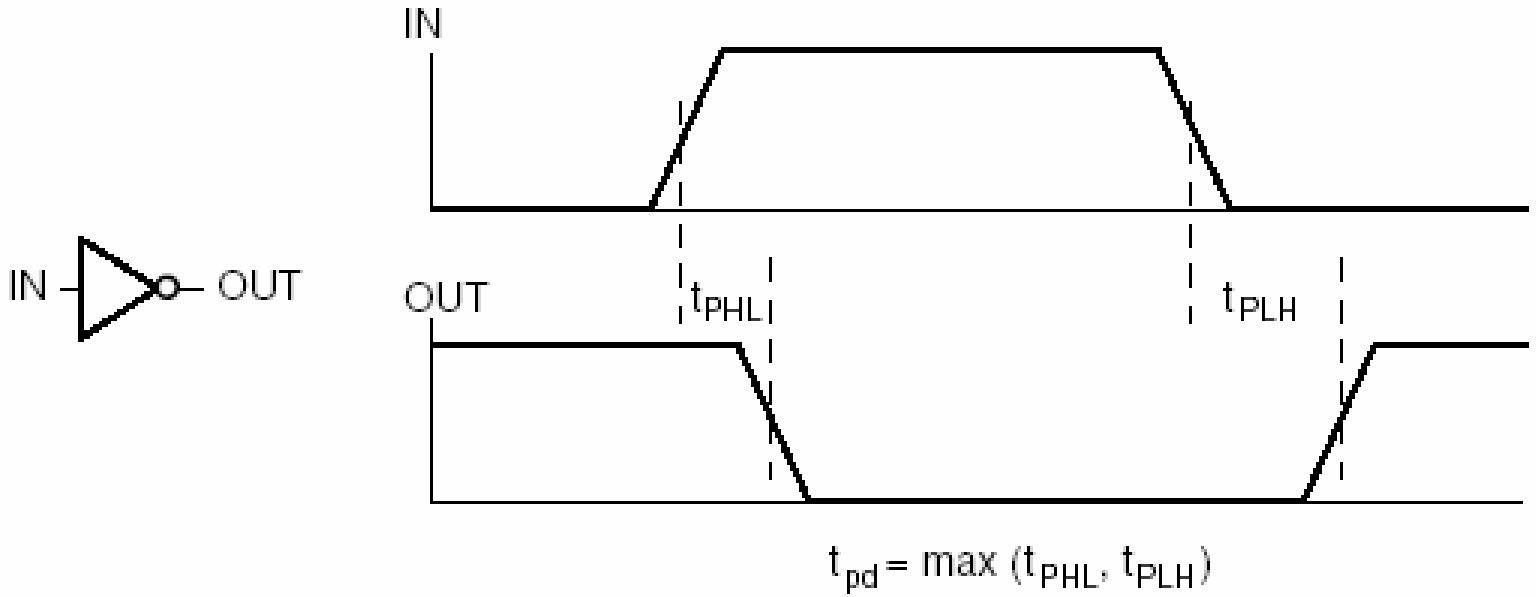


Fig. 2-40 Propagation Delay for an Inverter

Real Gates: Transport Vs Inertial delay

- Inertial delay is intended to model the delay through a gate, in which there is some minimum pulse length that must be maintained before an event is propagated. This minimum pulse length is the rejection time.

- Transport delay, on the other hand, models the delay on a wire, so pulses of any width are propagated.

Real Gates: Transport Vs Inertial delay …….

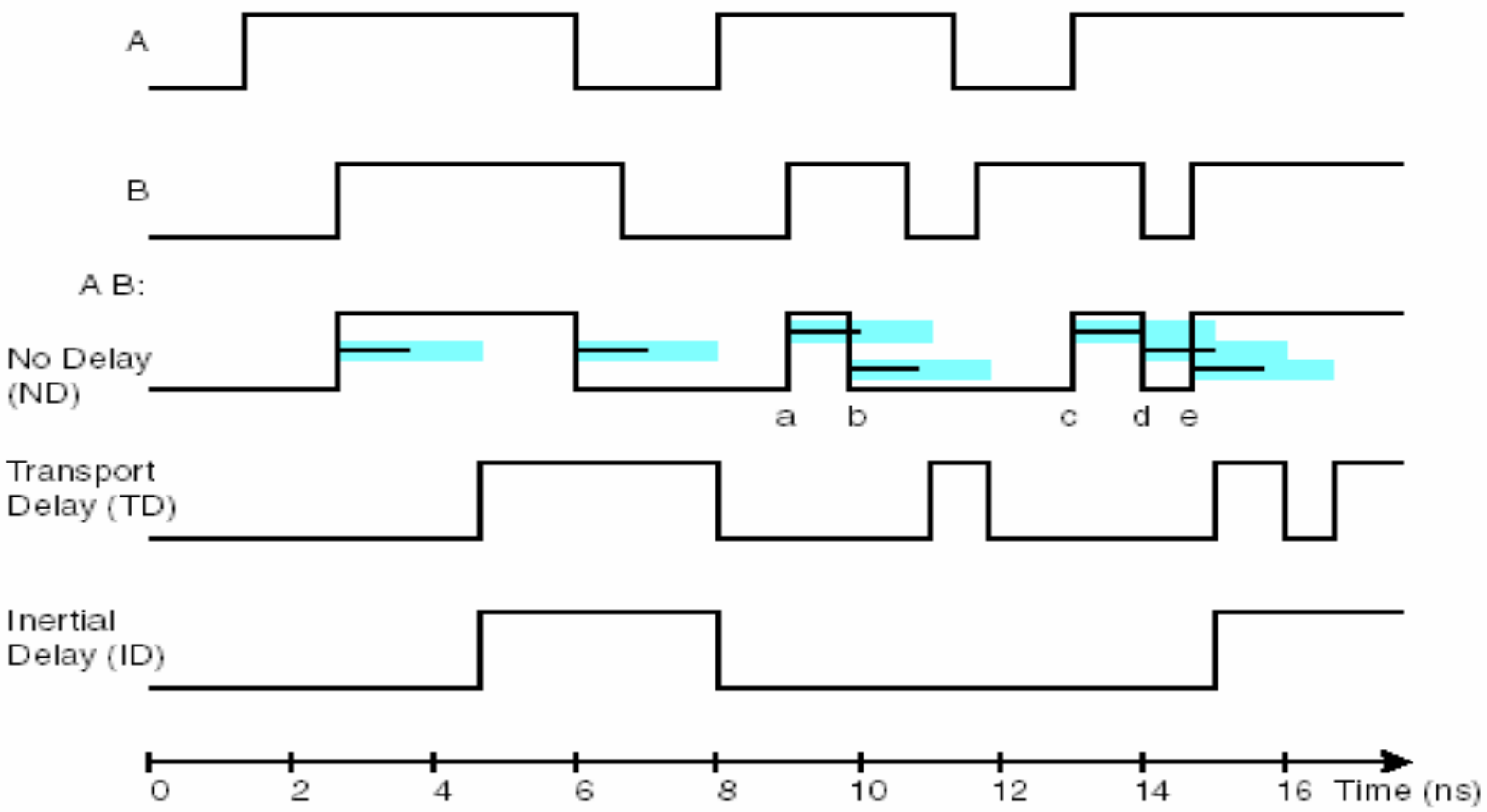


Fig. 2-41 Examples of Behavior of Transport and Inertial Delays

Programmable Logic Technologies

- Five types of PLDs, programmable logic devices:
  - ROM
  - PLA, programmable logic array
  - PAL, programmable array logic
  - CPLD, complex PLD
  - FPGA, field programmable gate array
- Programming technologies to control connections. First three are permanent.
- Use of fuse in each of the programmable points of the PLD having two states CLOSED and OPEN. Oldest programming technology. The fuse is blow by passing high current through application of high voltage.

Programmable Logic Technologies

- Mask programming, done at the last steps of chip fabrication process, where connections are made or not made in the metal layers serving as chip conductors.
- Antifuse, the opposite of a fuse, two conductors are separated by a material having a high resistance. This acts as an open circuit. By applying high voltage the antifuse is melted to provide low resistance, thus a closed path.
- Connection control using SRAM: A SRAM bit driving the gate of an MOS n-channel transistor at the programming point. If SRAM bit stores 1, then transistor is on / connection CLOSED. Otherwise, it is off / connection OPEN. This is volatile technology.

Programmable Logic Technologies

Programming technologies to build look-up tables:

Logic inputs are address inputs for reading the SRAM, logic outputs are the stored values for the addressed word that appears on the SRAM data outputs, store truth table in the SRAM, hence lookup table.

Programming technologies to control transistor switching:

Based on storing charge on a floating gate. Negative charge makes the transistor impossible to turn ON. Absence of that makes it possible to turn ON if a HIGH is applied to its regular gate. These technologies allow for reprogramming, thus ***erasable*** and ***electrically erasable*** technologies.

NOTE: Floating gate is located below the regular gate within an MOS transistor and is completely isolated by an insulating dielectric.

Read Only Memory: ROM

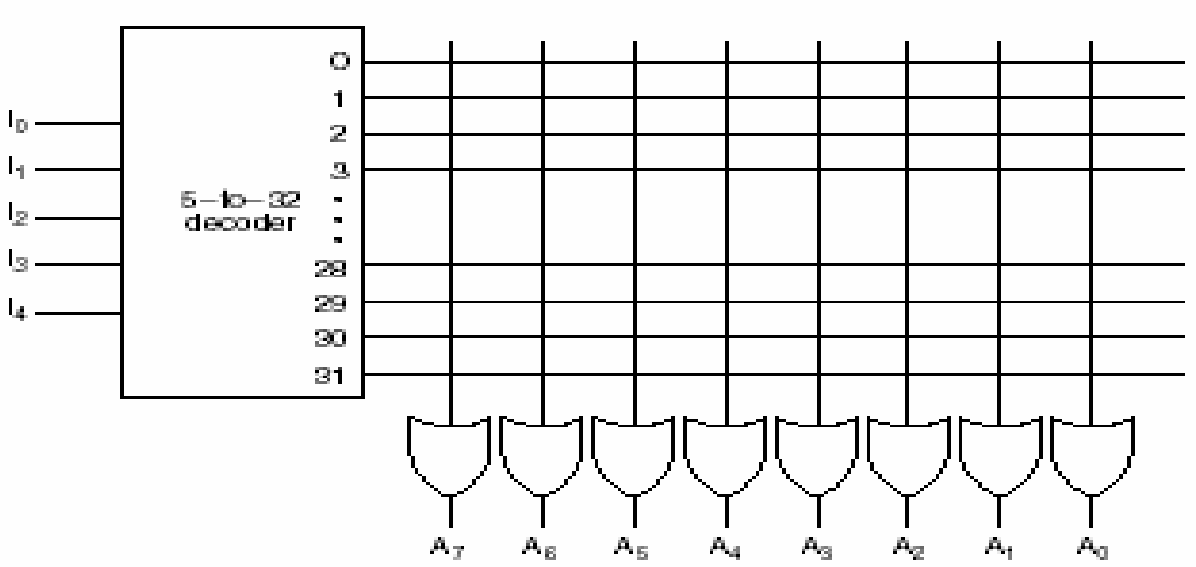
- ROM is a device where permanent binary information is stored.
- The information is specified by the designer and is then embedded into the ROM to form required interconnection or electronic device pattern.
- ROM is non volatile and hence information stays even if power supply is off.
- Inputs provide the memory address and the outputs give the data bits of the stored word that is selected by the address. **k** address input lines can specify $\mathbf{2^k}$ words. ROM does not have data inputs.
- ROM that can be programmed using fuse is called programmable ROM (PROM): EPROM and EEPROM.

ROM Block Diagram

Read Only Memory: Example

32x8 ROM, 32 words, 8 bits each. Five inputs to form binary numbers from 0– 31.

Each decoder output represents a memory address. The 32 outputs are connected through programmable connections to each of the 8 OR gates. Each OR gate must be considered as having 32 inputs. The ROM contains 32x8 = 256 programmable connections.



In general, a **2^kxn** ROM will have an internal **k-to-2^k** line decoder and **n** OR gates, each having **2^k** inputs.

Read Only Memory: Truth Table

- The internal storage of a ROM is specified by a truth table.
- The truth table shows the contents of a word in each address.

ROM Truth Table (Partial)

Inputs					Outputs							
I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		.							.			
		.							.			
		.							.			
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

Table 6-2 ROM Truth Table (Partial)

Read Only Memory: Programming

- Hardware procedure programs the ROM with connections that follow the truth table.
- Every 0 specifies an OPEN circuit and every 1 specifies a CLOSED circuit.
- Example: the 8-bit word 10110010 for permanent storage at input address 00011. (1's are marked with x in diagram)

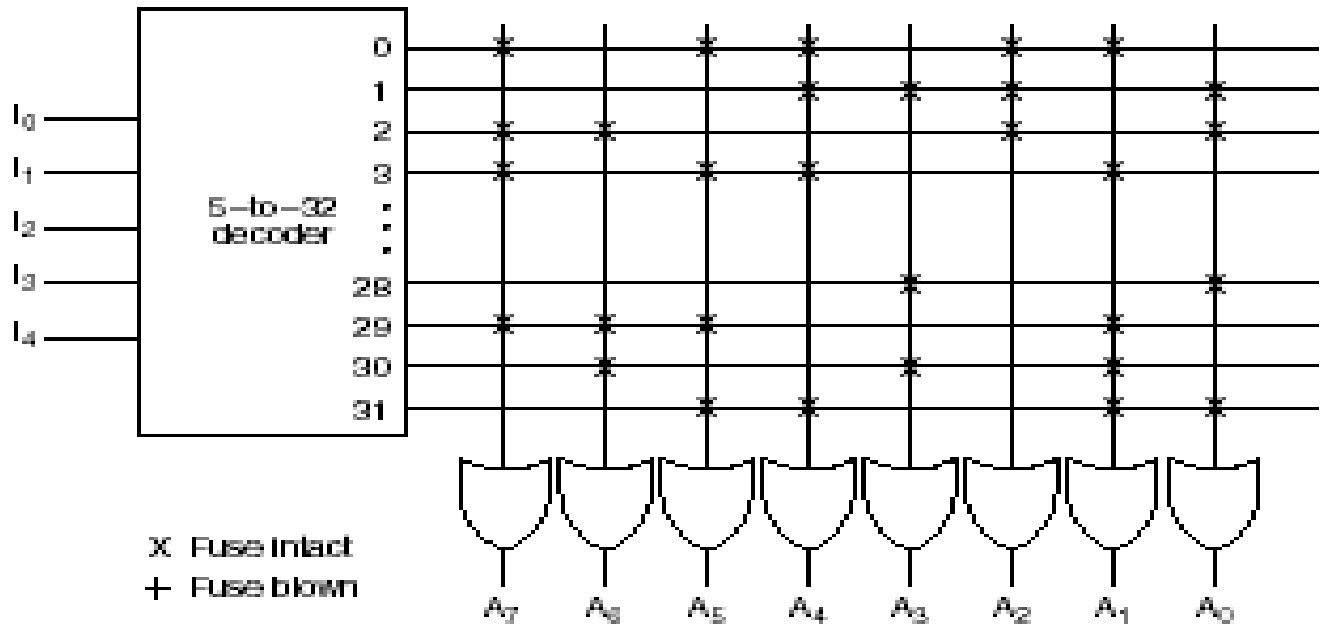
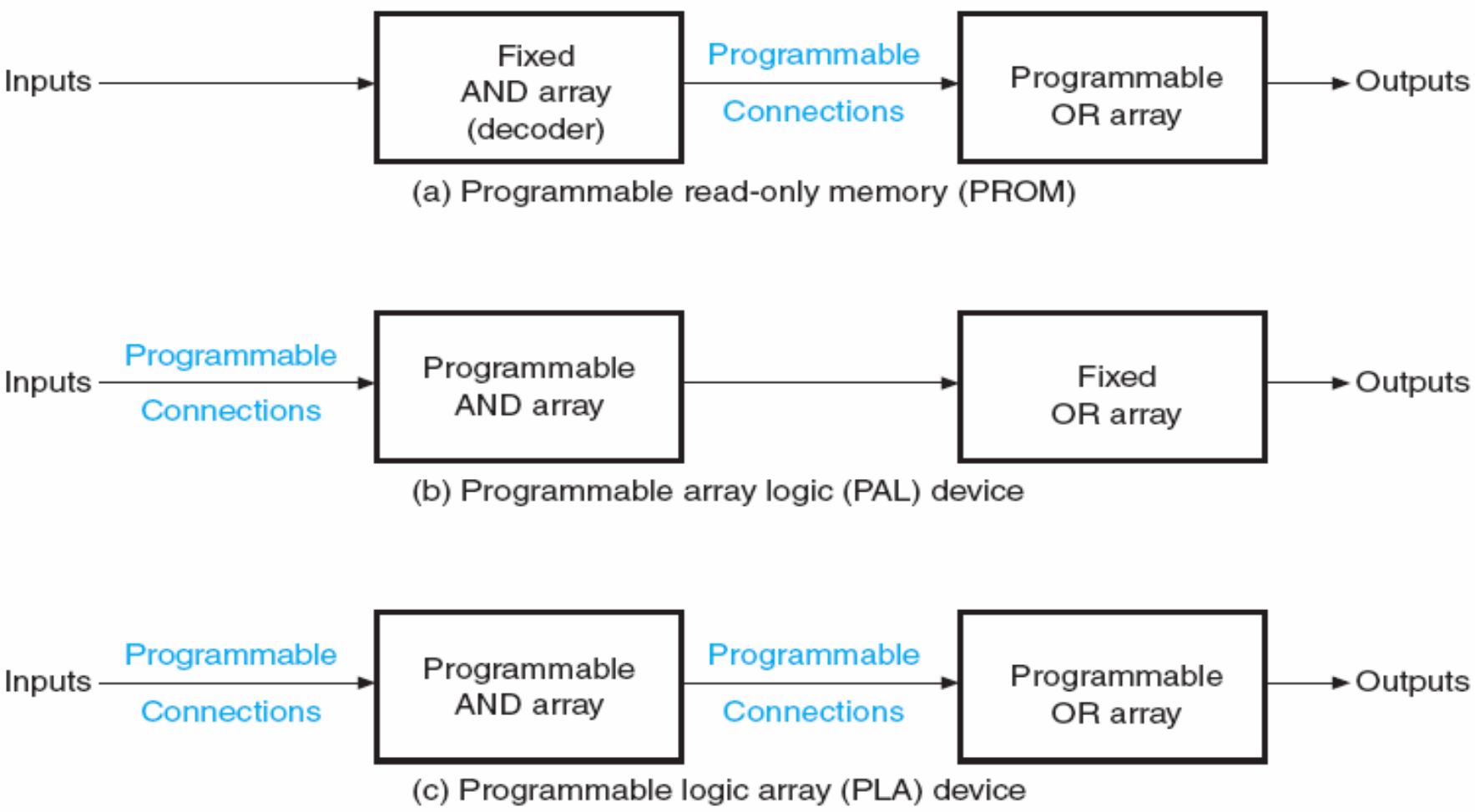


Fig. 6-21 Programming the ROM According to Table 6-2

Three Majors Types of PLDs



The three major types of PLDs differ in the placement of programmable connections in the AND-OR array.

Three Majors Types of PLDs

- Programmable Read-Only Memory (PROM): Has fixed AND array constructed as decoder and programmable connection for the output OR gates. PROM implements Boolean functions in sum-of-minterms form.
- Programmable Array Logic (PAL): Has fixed OR array and programmable connection AND array. The AND gates are programmed to provide the product terms for the Boolean functions, which are logically summed in each OR gate.
- Programmable Logic Array (PLA): Has programmable connections for both AND and OR arrays. The product terms in the AND array may be shared by any OR gate to provide the required sum-of-products implementation.

Programmable Logic Array (PLA)

- Similar to PROM, but does not provide full deocding of the variables and does not generate all the minterms.
- The decoder is replaced by an array of AND gates that can be programmed to generate product terms of the input variables.
- The product terms are then connected to OR gates to provide the sum-of-products for the Boolean function.

Programmable Logic Array : Example

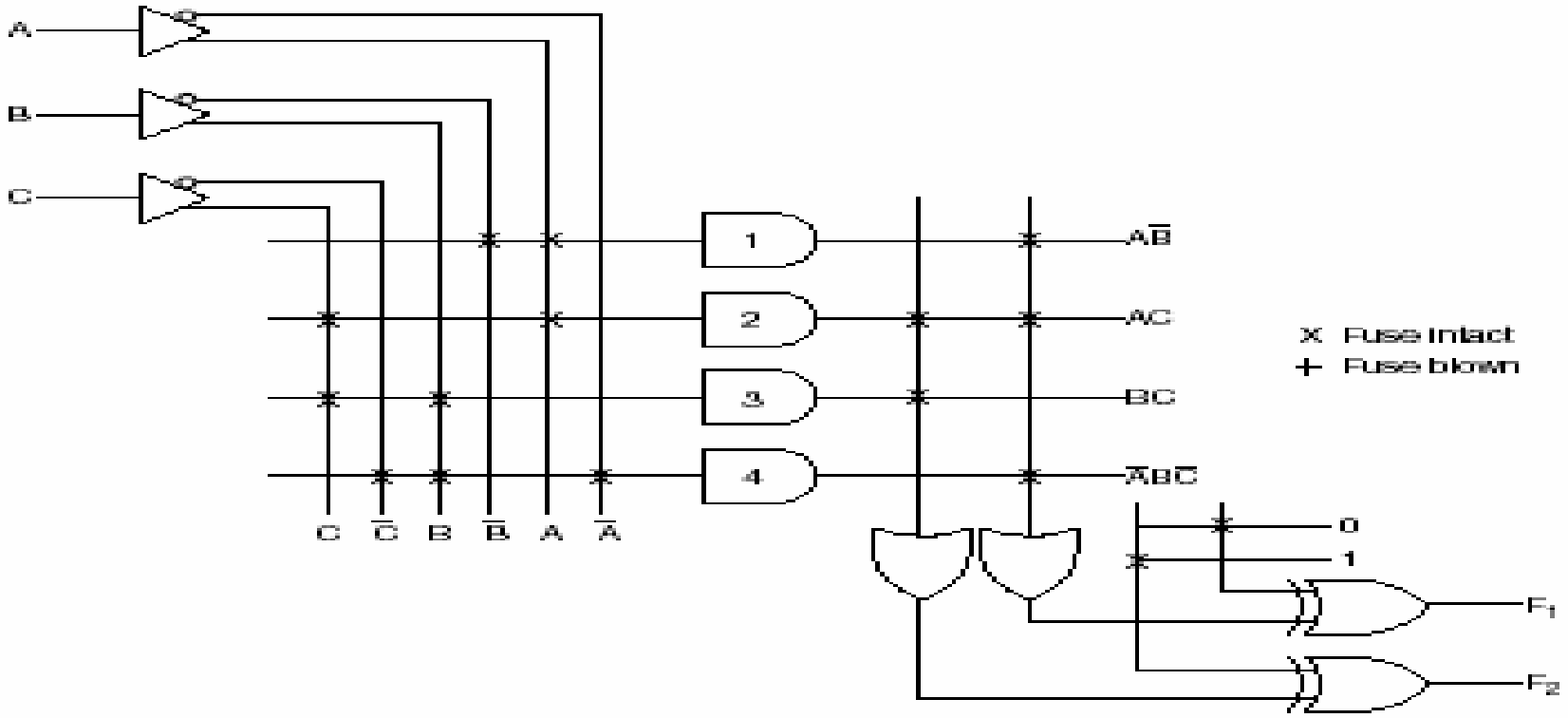


Fig. 6-24 PLA with Three Inputs, Four Product Terms, and Two Outputs

The Boolean functions implemented in the above PLA are: F_1=AB'+AC+A'BC' and F_2=(AC+BC)'. The output is complemented or left in its true form, depending on the programming of the connection associated with the XOR gate.

Programmable Logic Array: Example…

- The fuse map of a PLA is specified in a tabular form.
- The table consists of three sections. The 1st section lists the product term numbers. The 2nd section specifies the paths between inputs and AND gates. The 3rd section specifies the paths between the AND and OR gates.
- For each product term the inputs are marked 1, 0, or -, depending on its

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

Table 6-4 Programming Table for the PLA in Figure 6-24

For each output variable we may have T (true) or C (complement) for controlling the output XOR gate.

Programmable Logic Array: Size

- The size of a PLA is specified by the number of inputs, the number of product terms, and the number of outputs.
- For n inputs, k product terms, and m outputs, the internal logic of PLA consists of n buffer-inverters gates, k AND gates, m OR gates, and m XOR gates. There are 2n*k programmable connections, between the inputs, and the AND arrays, k*m programmable connections between the AND OR arrays, and m programmable connections associated with the XOR gates.
- In designing a digital system with PLA, there is no need to show the internal connections of the unit. All that is needed is a PLA programming table from which PLA can be programmed to supply the required logic.

Programmable Array Logic (PAL) Devices

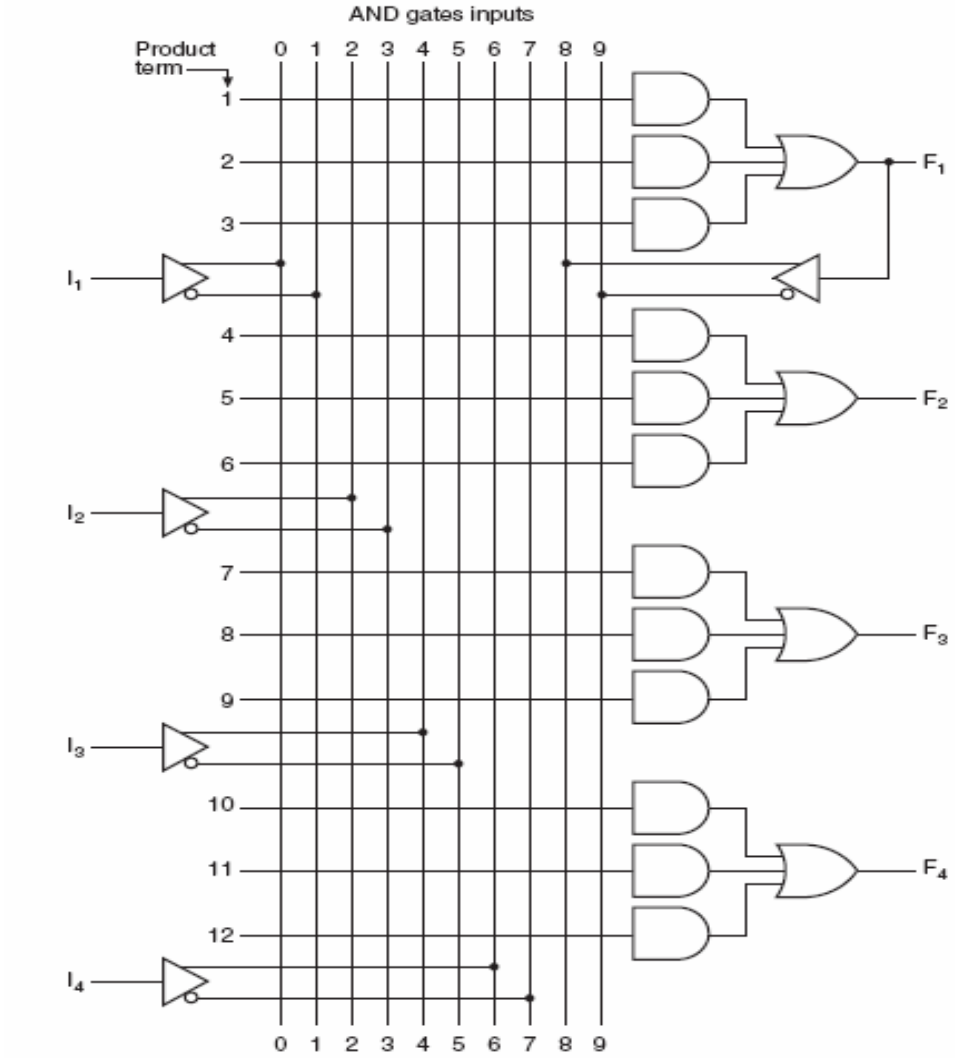


Fig. 6-26 PAL® Device with Four Inputs, Four Outputs, and a Three-wide AND-OR Structure

The PAL devices are easy to program, as only the AND gates are programmable, but it is not as flexible as the PLA.

Each input has a buffer-inverter gate and each output is generated by a fixed OR gate. Device has four sections, each composed of a three-wide AND-OR array, meaning that there are three programmable AND gates in each section. Each AND gate has 10 programmable input connections, indicated in the diagram by the 10 vertical lines.

VLSI Programmable Logic Device

- The gate arrays have gates in the range 1000-1,000,000, typically called complex programming logic devices (CPLDs) or field-programming gate arrays (FPGAs).
- Properties:
  - Substantial amount of uncommitted combinational logic
  - Pre-implemented flip-flops, and
  - Programmable interconnections between the combinational logic, flip-flops, and the chip input/output.

VLSI Programmable Logic Device ….

- The VLSI PLDs, differ significantly from vendor to vendor.
- Three vendors:
  - Actel
  - Xilinx
  - Altera
- Reference: Brown, S., and Rose, J., "*FPGA and CPLD Architectures: A Tutorial*", IEEE Design & Test of Computers, pp. 42-57, 1996.

VLSI Programmable Logic Device: Altera

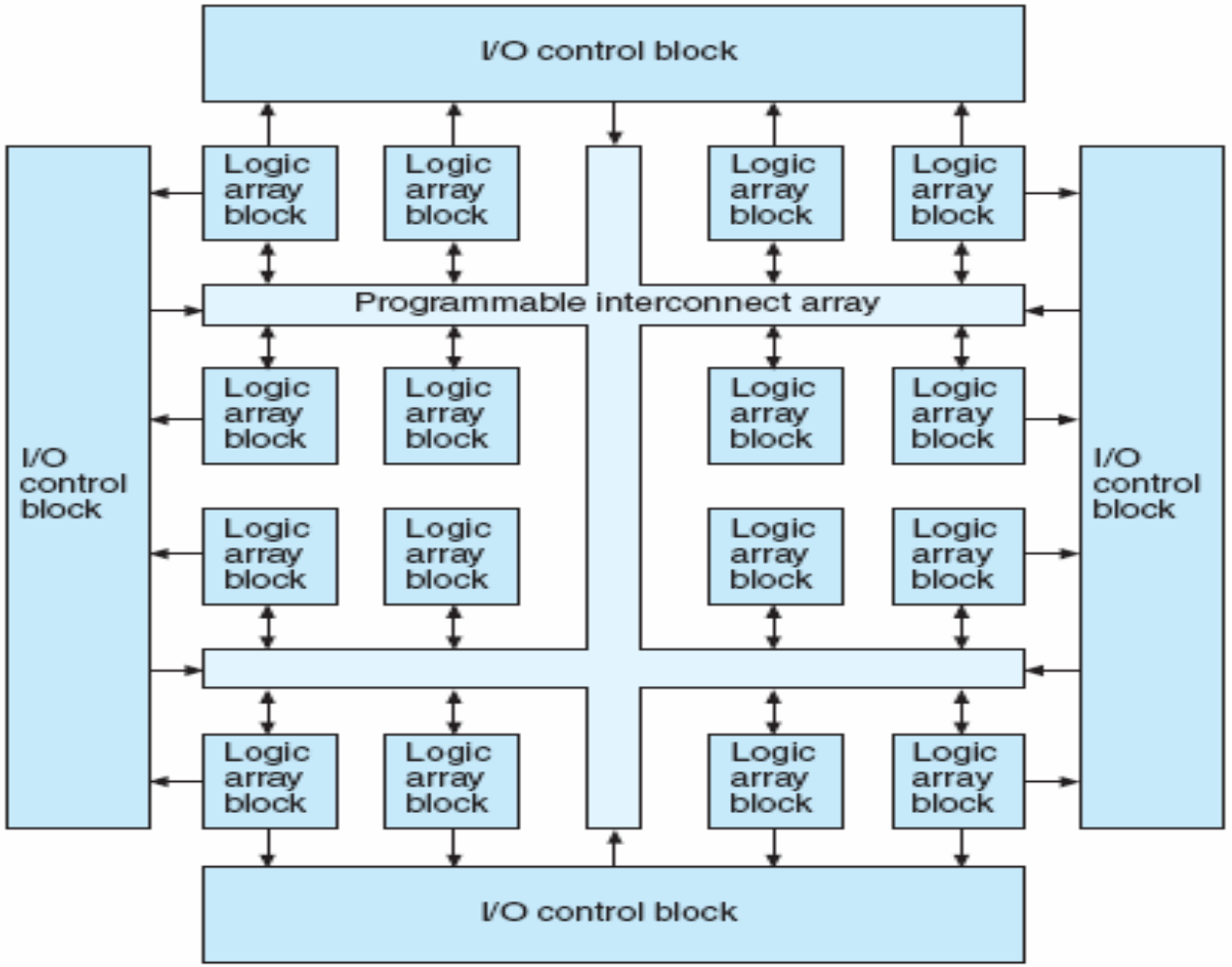


Fig. 6-28 Altera® MAX 7000™ Structure (Reprinted with Permission of Altera Corporation,© Altera Corp., 1991)

VLSI Programmable Logic Device: Altera

- Based on an EEPROM floating gate technology.
- There are 16 identical logic array blocks, all of whose outputs feed into the programmable interconnect array.
- The programmable interconnect arrays also receive inputs from the I/O control blocks.
- Connections can be programmed from all signals within the programmable interconnection array to the inputs of the logic array block.
- Each logic array block has 16 macrocells, each with a flip-flop and basic combinational circuits.
- Each macrocell in the logic array blocks around the outer edge is connected to I/O control blocks.

VLSI Programmable Logic Device: Xilinx

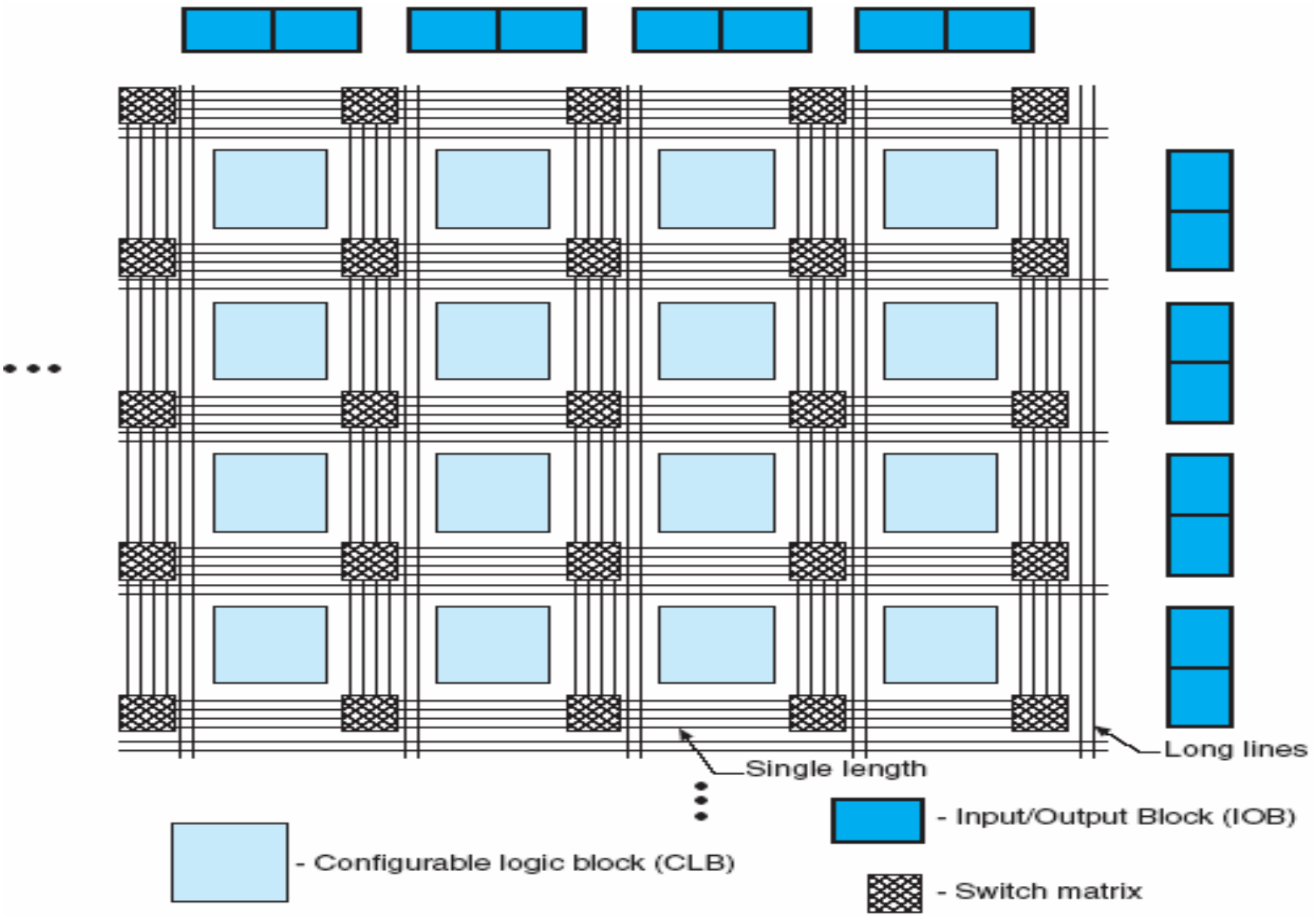
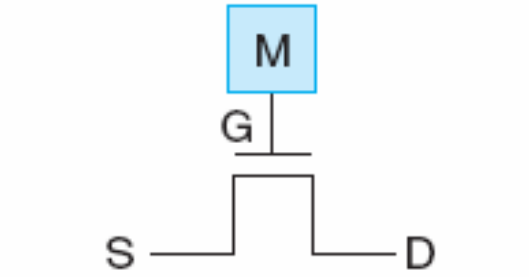


Fig. 6-29 Xilinx® XC4000™ FPGA Structure

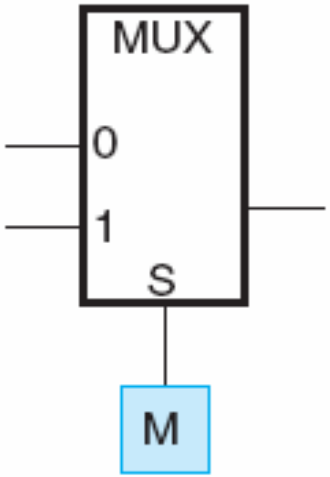
VLSI Programmable Logic Device: Xilinx …

- The logic is implemented in an array of programmable blocks of logic called configurable logic blocks (CLBs).
- The input to and output from the array is handled by input or output blocks (IOBs).
- Connections among CLBs and IOBs are programmed.
- Xilinx uses SRAM technology to store the programming information.
- Three techniques used to control the SRAM bits:
  - Pass transistor control
  - Multiplexer control
  - Lookup table implementation

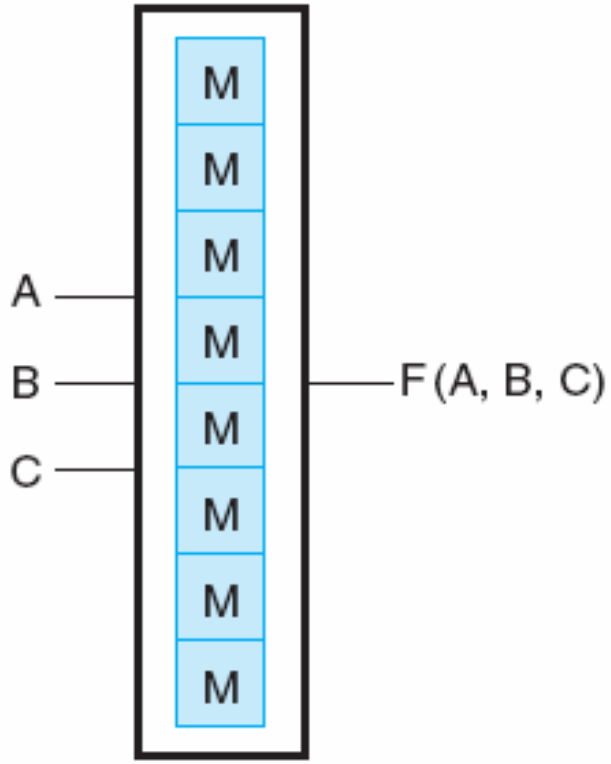
VLSI Programmable Logic Device: Xilinx …



(a) Pass transistor control



(b) Multiplexer control

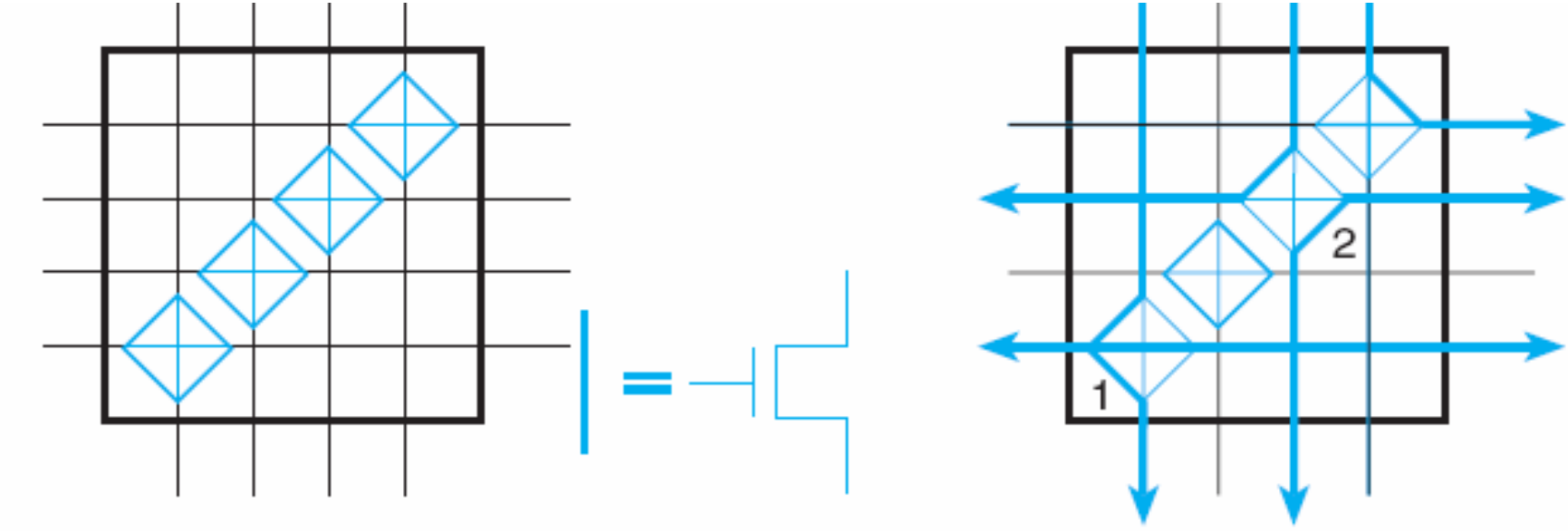


(c) Look up table implementation

Fig. 6-30 SRAM Bit Use in Xilinx® FPGAs

VLSI Programmable Logic Device: Xilinx …

- Long Lines: Wiring segments spanning entire length of the array
- Single Length Wiring: Span a single CLB (similarly, double length wiring)
- Interconnection is done using switching matrix.
- Fig a: Four segments, six transistors each, one vertical one horizontal, and four diagonal.



(a) Switch Matrix Transistors (b) Examples of Connections

Fig. 6-31 Example of Xilinx® Switch Matrix (Adapted

VLSI Programmable Logic Device: Xilinx …

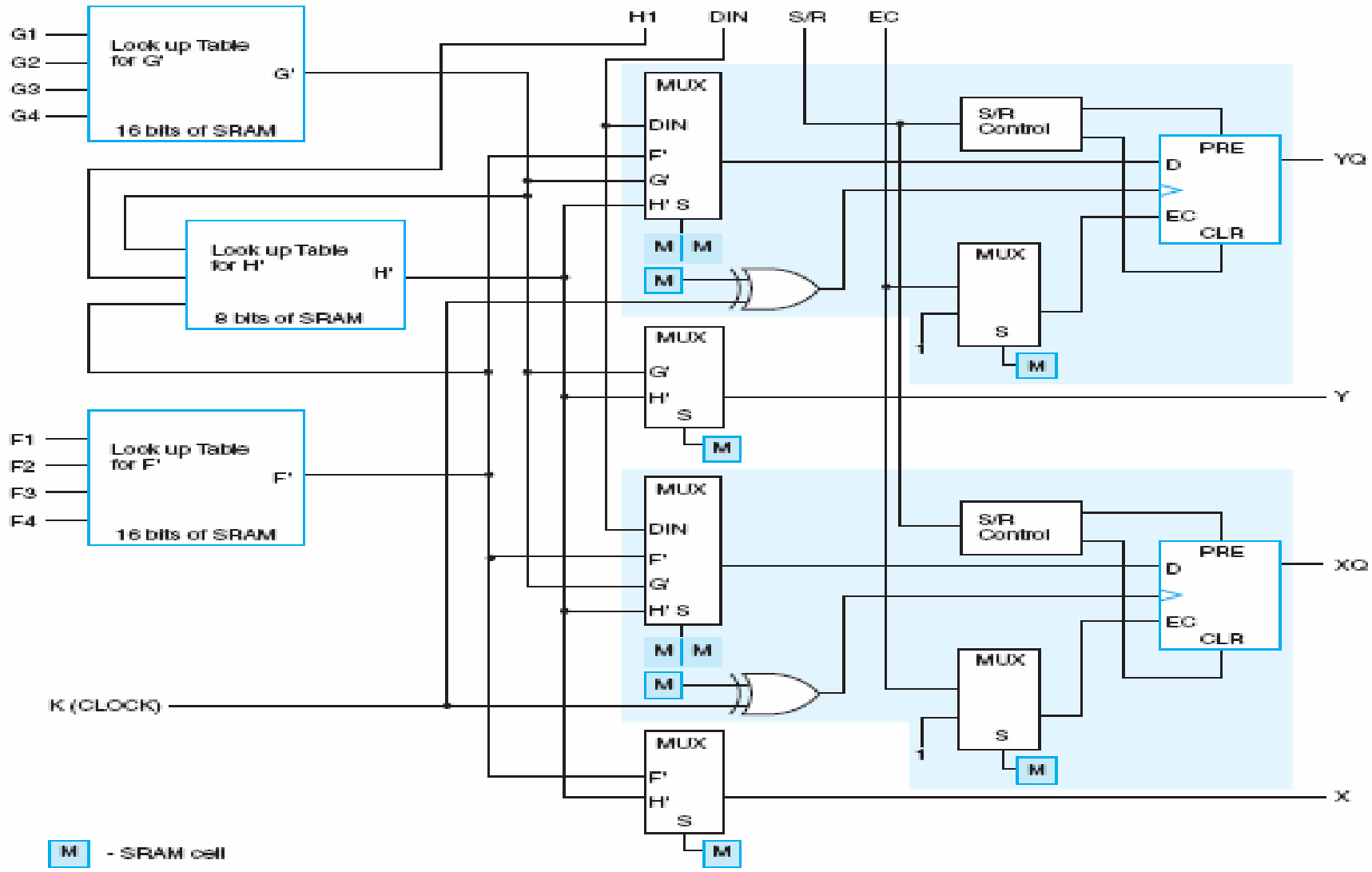
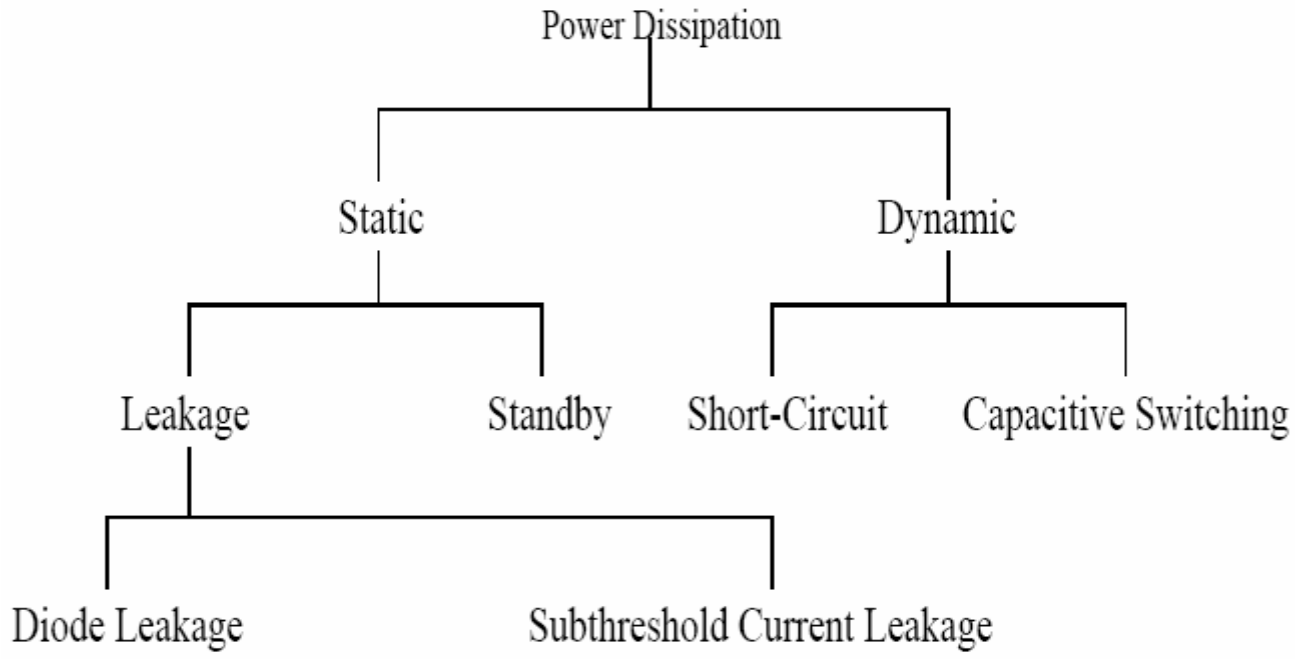


Fig. 6-32 Simplified Diagram of a Xilinx® Configurable Logic Block

Power Dissipation/Consumption in CMOS Circuits



Leakage Current: Reverse biased current in the parasitic diode and subthreshold current due to charge inversion existing at gate below VT.

Standby Current: Continuous DC current from Vdd to ground

Short-Circuit Current: DC current from Vdd to ground during output transition

Capacitive Current: Flows to charge discharge capacitive loads during logic change.