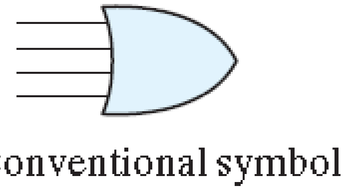
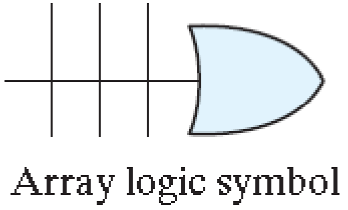
Chapter 7 Memory and Programmable Logic

7-1 Introduction

- Memories are for binary information storage.
- Two main memory types: *random-access memory* (*RAM*) and *read-only memory* (*ROM*). [actually both of them are random accessible]
- *RAM* can perform both *write* and *read* operations, while *ROM* can perform only the *read* operation.
  - *Write*: storing new information into memory
  - *Read*: transferring the stored information out of memory
- ROM is a *programmable logic device* (*PLD*) which stores the binary information via a hardware *programming* process.
- A PLD is an IC with internal logic gates connected through electronic paths that behave similarly to fuses that will be burned out to obtain a specified function.
- Programmable logic array (PLA), programmable array logic (PAL), and the field-programmable gate array (FPGA) are other commonly seen PLDs.



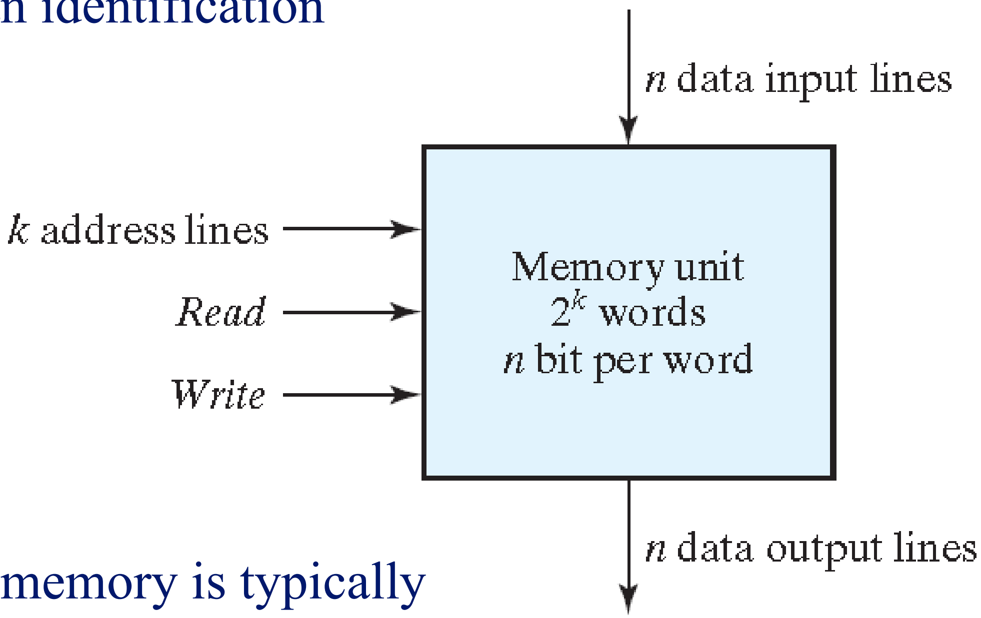


7-2 Random-Access Memory

- A memory unit includes an array of storage cells, together with associated read/write control circuits.
- The architecture of memory is such that information can be selectively retrieved from any of its internal locations.
- A memory unit stores binary information in groups of bits called *words* that move in and out of storage as a unit.
- A memory word is a group of 1's and 0's and may represent a number, an instruction, one or more alphanumeric characters, or any other binary-coded information.
- Most computer memory uses words that are multiples of 8 bits in length.
- The capacity of a memory unit is usually stated as the total number of bytes that the unit can store.

A Memory Unit of RAM

- Memory is accessed through *data input* and *output* lines, *address* selection lines, and *control* lines that specify the direction of transfer.
- The *n data input* lines provide the information to be stored in memory, and the *n data output* lines supply the information coming out of memory.
- The *k address* lines specify the particular word chosen.
- Read/Write are the control inputs specify the transfer direction.
- Each word in memory is assigned an identification *address* ranging from 0 up to 2^k - 1.



- The number of words (or bytes) in memory is typically with one of the letters K (kilo), M (mega), and G (giga).

Example: 1K x 16 Memory

- A 1024 × 16 Memory

Memory address		
Binary	Decimal	Memory content
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

Write and Read Operations

- Write operation
  - Apply the binary address to the address lines
  - Apply the data bits to the data input lines
  - Activate the *write* input
- Read operation
  - Apply the binary address to the address lines
  - Activate the *read* input
- Read/Write share a same pin; when memory is enabled, 0 for read and 1 for write.

Control Inputs to Memory Chip

Memory Enable	Read/Write	Memory Operation
0	X	None
1	0	Write to selected word
1	1	Read from selected word

RAM in HDL

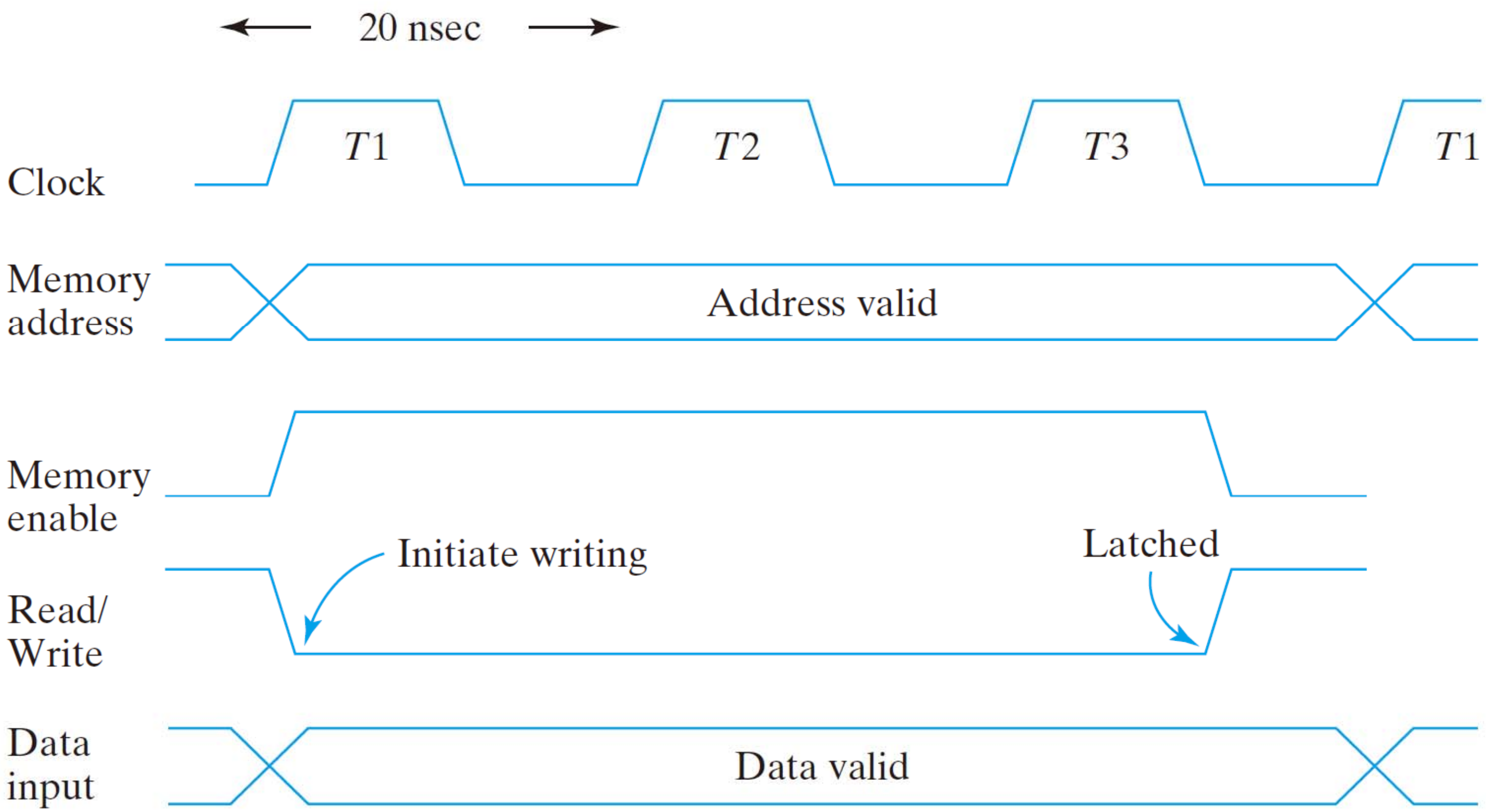
```
// Read and write operations of memory
// Memory size is 64 words of four bits each.

module memory (Enable, ReadWrite, Address, DataIn, DataOut);
  input   Enable, ReadWrite;
  input   [3: 0] DataIn;
  input   [5: 0] Address;
  output  [3: 0] DataOut;
  reg [3: 0]     DataOut;
  reg [3: 0]     Mem [0: 63];                     // 64 x 4 memory
  always @ (Enable or ReadWrite)
   if (Enable)
    if (ReadWrite) DataOut = Mem [Address];       // Read
   else Mem [Address] = DataIn;                   // Write
  else DataOut = 4'bz;                            // High impedance state
endmodule
```

Timing Waveforms

- The memory operation is controlled by an external device such as a central processing unit (CPU) which is usually synchronized by a clock.
- Instead, the memory's read and write operations are specified by control inputs.
- The *access time* of memory is the time required to select a word and read it.
- The *cycle time* of memory is the time required to complete a write operation.
- The CPU must provide the memory control signals to synchronize its internal clocked operations with the read and write operations.
- This means that the access time and cycle time of the memory must be within a time equal to a fixed number of CPU clock cycles.

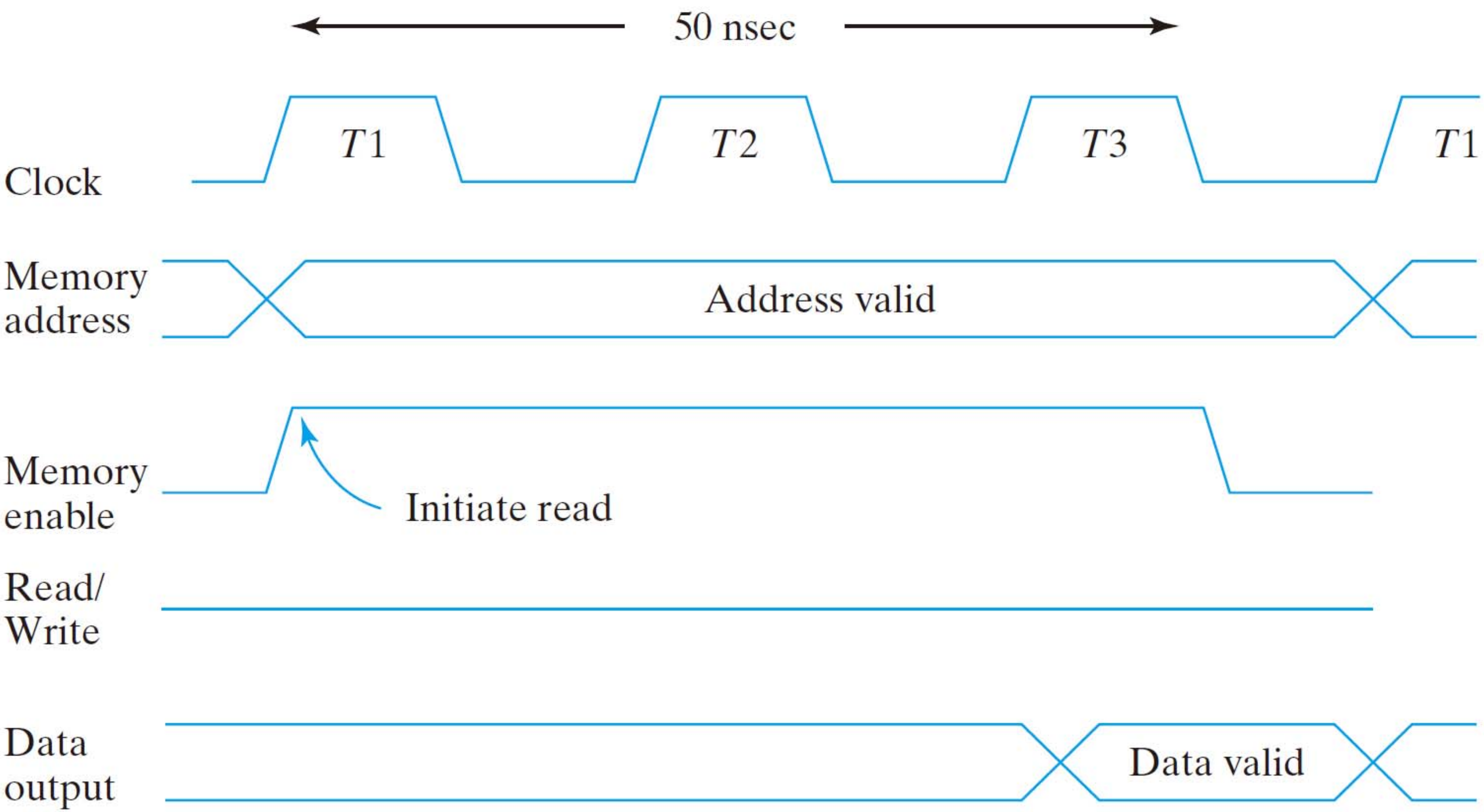
Write Cycle

- CPU clock is 50 MHz; a period of 20 ns
- The access/cycle time < 50 ns



(a) Write cycle

Read Cycle



(b) Read cycle

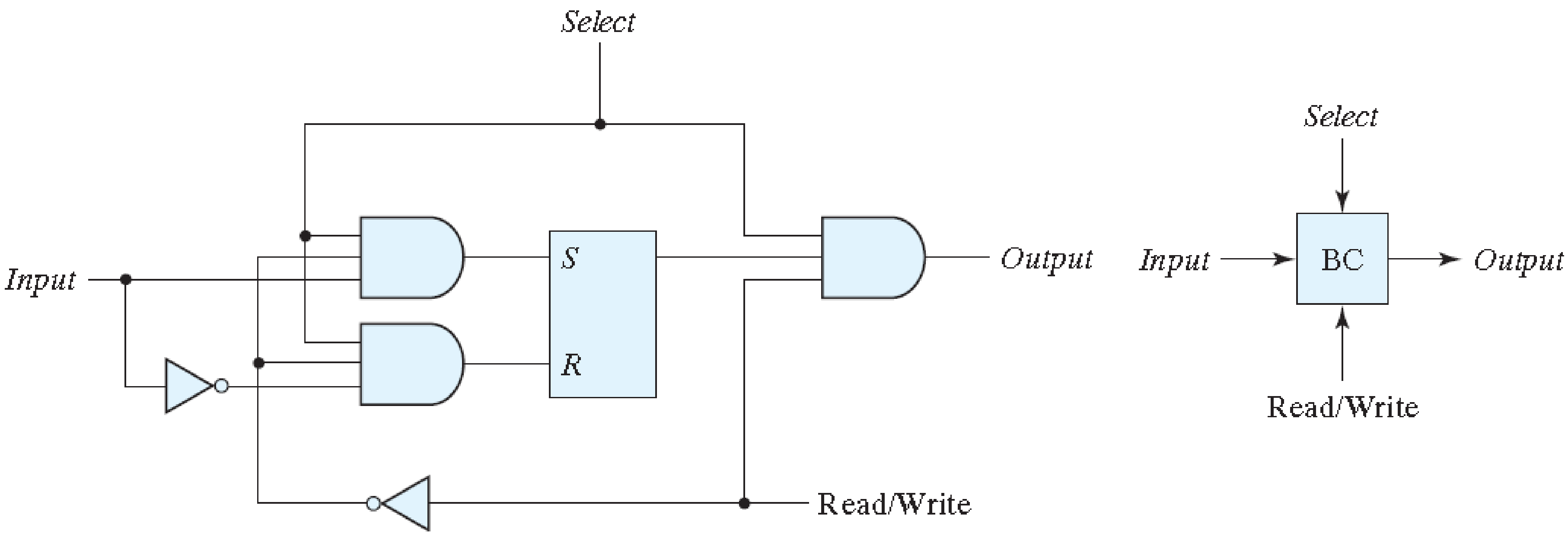
Types of Memories

- Static
  - information are stored in latches
  - remains valid as long as power is applied
  - short read/write cycle
- Dynamic
  - information are stored in the form of charges on capacitors
  - the stored charge tends to discharge with time
  - need to be refreshed (read and write back)
  - reduced power consumption
  - larger memory capacity

- Volatile
  - lose stored information when power is turned off
  - SRAM, DRAM
- Non-volatile
  - retain stored information after the removal of power
  - ROM
  - EPROM, EEPROM
  - Flash memory

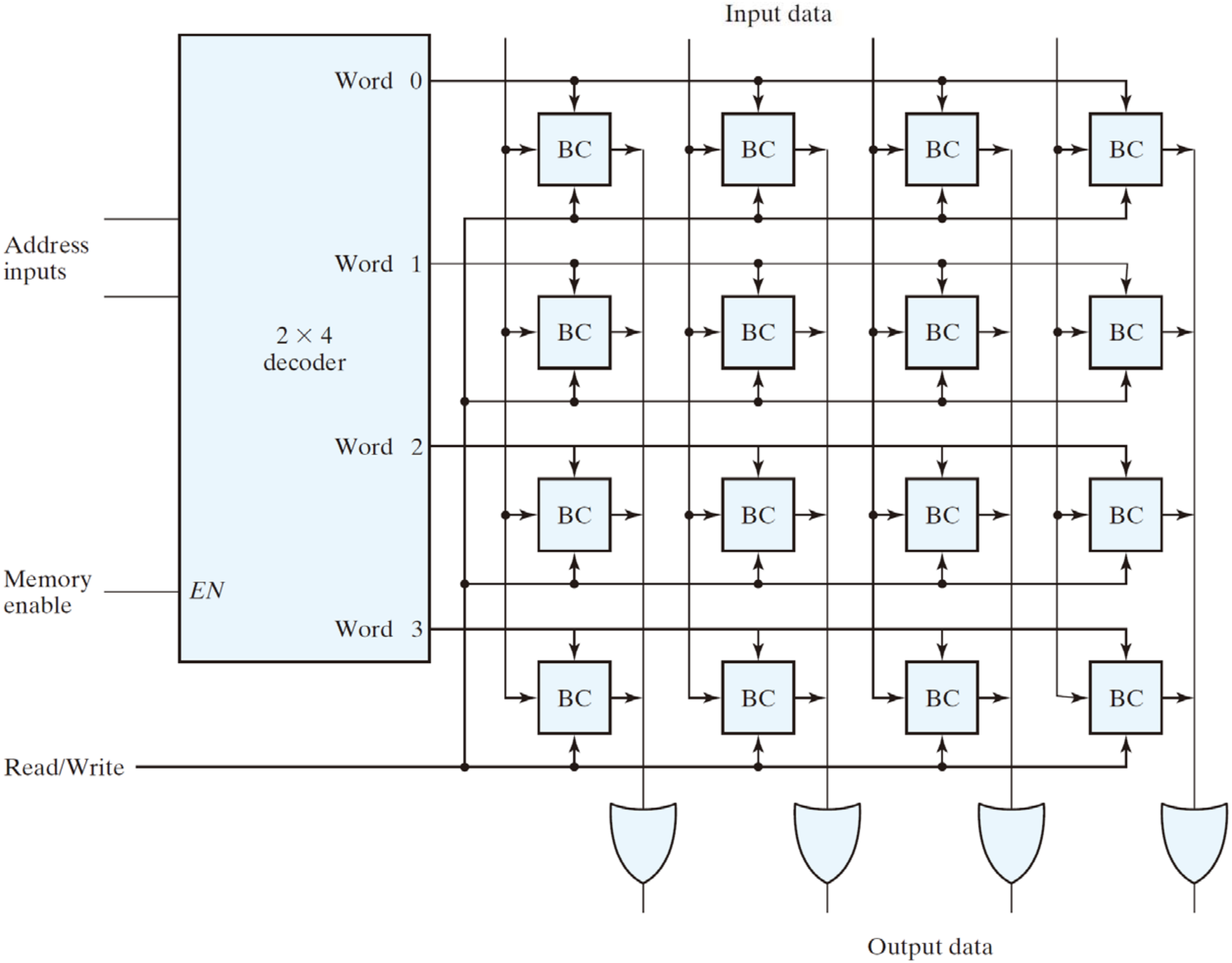
7-3 Memory Decoding

- A memory unit contains memory cells and decoding circuits.
- Decoding circuits may select the memory word specified by the input address.
- A bit memory cell is shown below:



- The cell is modeled by an *SR* latch with associated gates to form a *D* latch. **Actually, the cell is an electronic circuit with four to six transistors.**
- The cell must be very small in order to be able to pack as many cells as possible in the small area available in an IC chip. Think about 10^9 cells crowed in 1 cm^2.

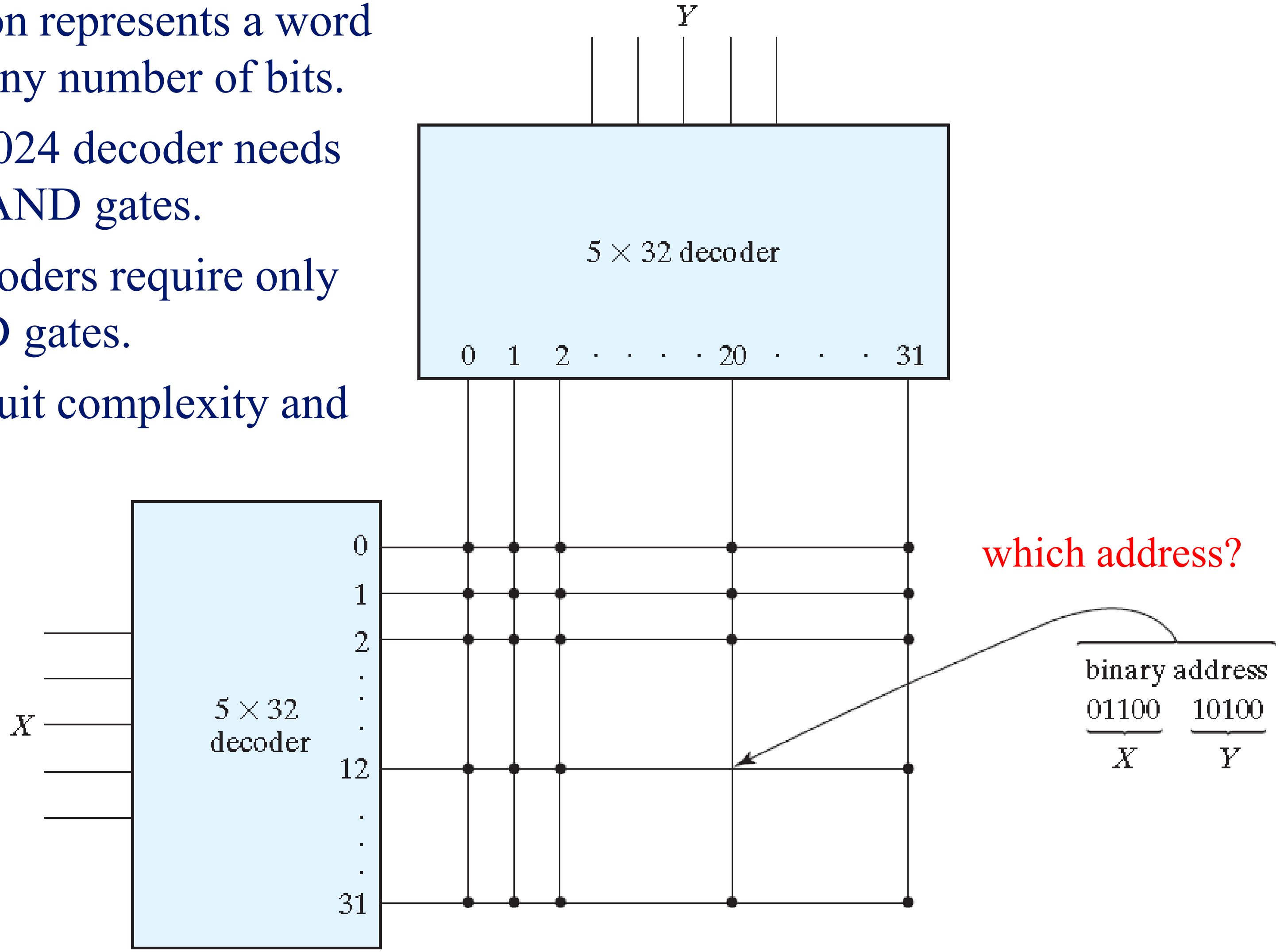
A 4 x 4 RAM


Input data
Word 0
Word 1
Word 2
Word 3
2 × 4
decoder
EN
Address
inputs
Memory
enable
Read/Write
BC
Output data

- Two address lines are needed to go through a 2 x 4 decoder to select one of the four words.
- The decoder is enabled with the *memory-enable* input.
- Once a word has been selected (*addressed*), the read/write input determines the operation.
- During the read operation, the four bits of the selected word go through the multi-input OR gates to the output terminals.
- During the write operation, the data available in the input lines are transferred into the four binary cells of the selected word.
- Commercial RAMs may have a capacity of millions to trillions of words, and each word may range from 1 to 64 bits.
- A memory with 2*k* words of *n* bits per word requires *k* address lines that go into a k x 2^k decoder.
- Each one of the decoder outputs selects one word of *n* bits for reading or writing.
- A decoder with *k* inputs and 2^k outputs requires 2^k AND gates with *k* inputs per gate.

Coincident Decoding

- A two-dimensional selection scheme can reduce the complexity of the decoding circuits; for example, 1-k memory is shown.
- Each intersection represents a word that may have any number of bits.
- A single 10 x 1024 decoder needs 1024 10-input AND gates.
- Two 5 x 32 decoders require only 64 5-input AND gates.
- Reduce the circuit complexity and the cycle time.

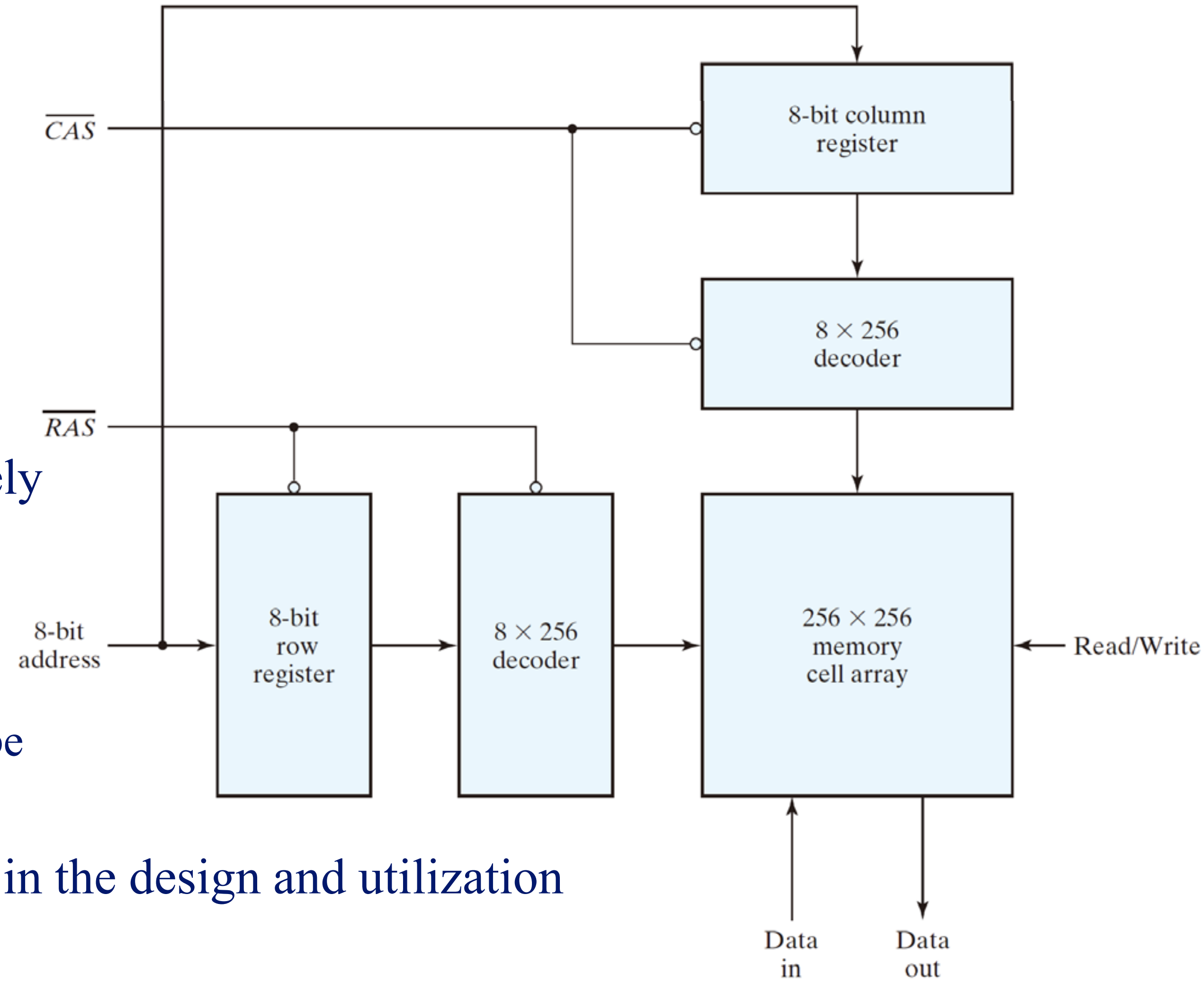


Address Multiplexing

- Address multiplexing is used in DRAM design to reduce the number of pins in IC package.
- In a two-dimensional array, the address is applied in two parts at different times, with the row address first and the column address second.
- Addresses are respectively latched into the 8-bit registers.

  RAS: row address strobe

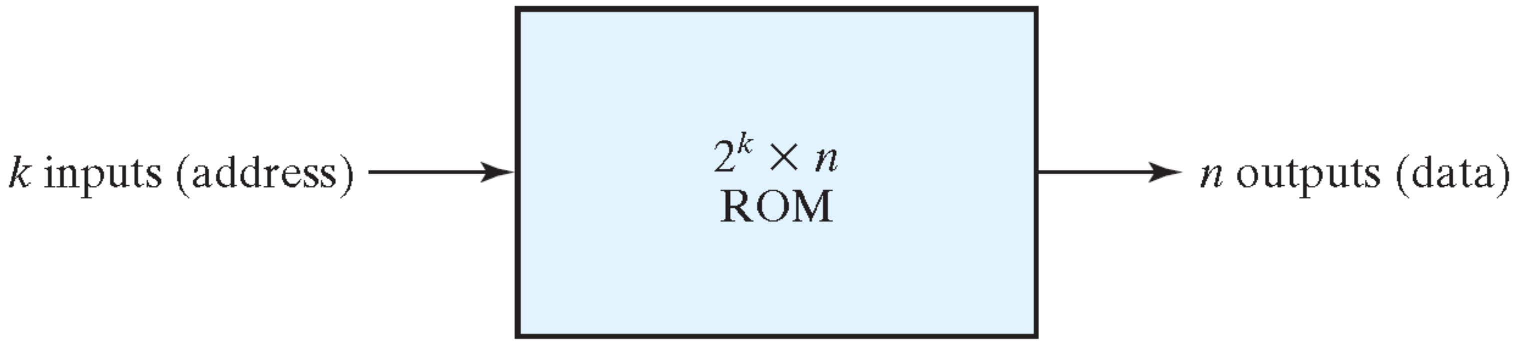
  CAS: column address strobe



- The timing is important in the design and utilization of RAM.

7-5 Read-Only Memory (ROM)

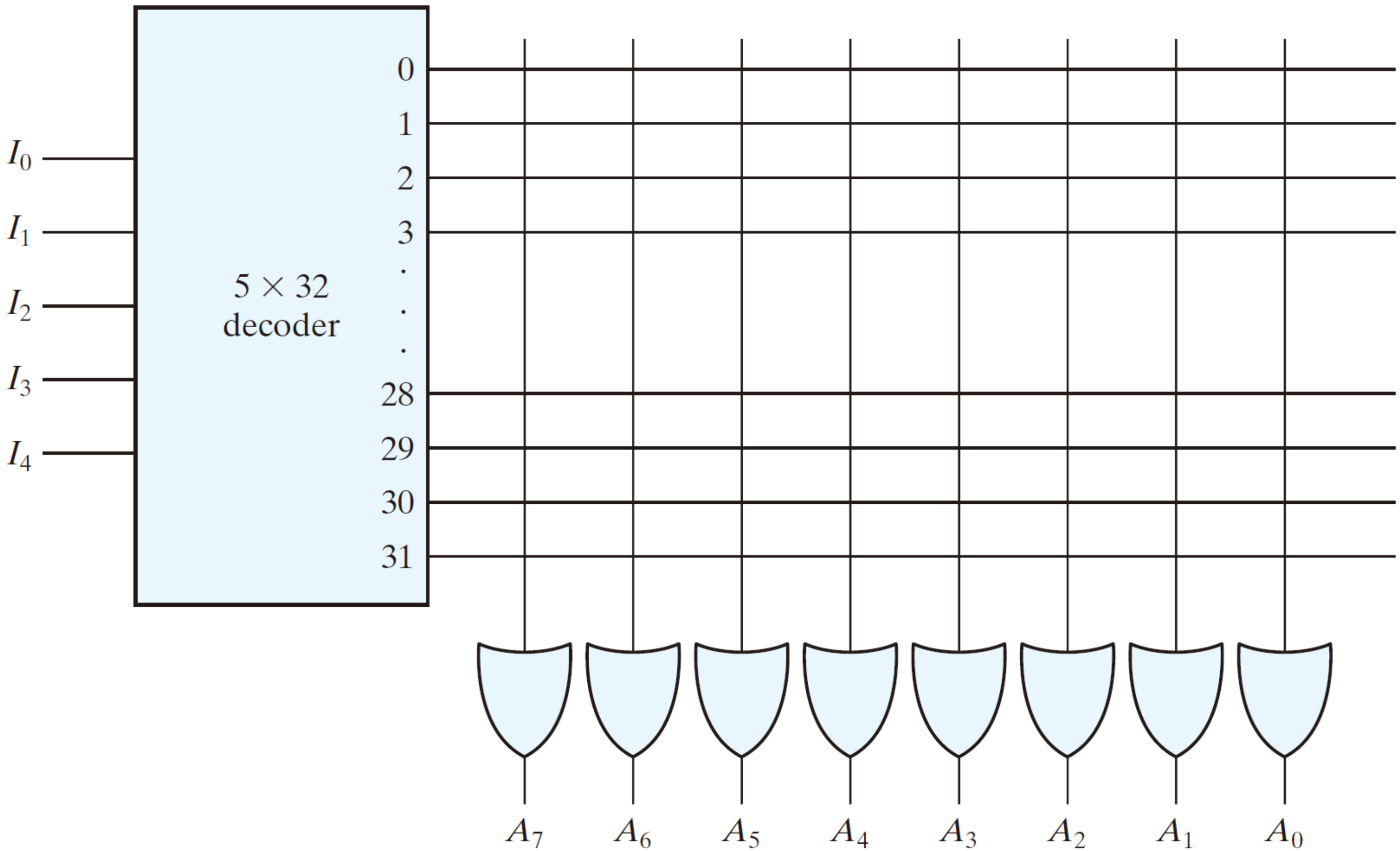
- ROM stores binary information permanently even when power is turned off and on again.
- A block diagram of a ROM consisting of k inputs and n outputs is shown.



- k address input lines can specify 2^k words.
- The inputs provide the address, and the outputs give the data word selected by the address.
- Integrated circuit ROM chips have one or more enable inputs and sometimes come with three-state outputs to facilitate the construction of large ROM arrays.

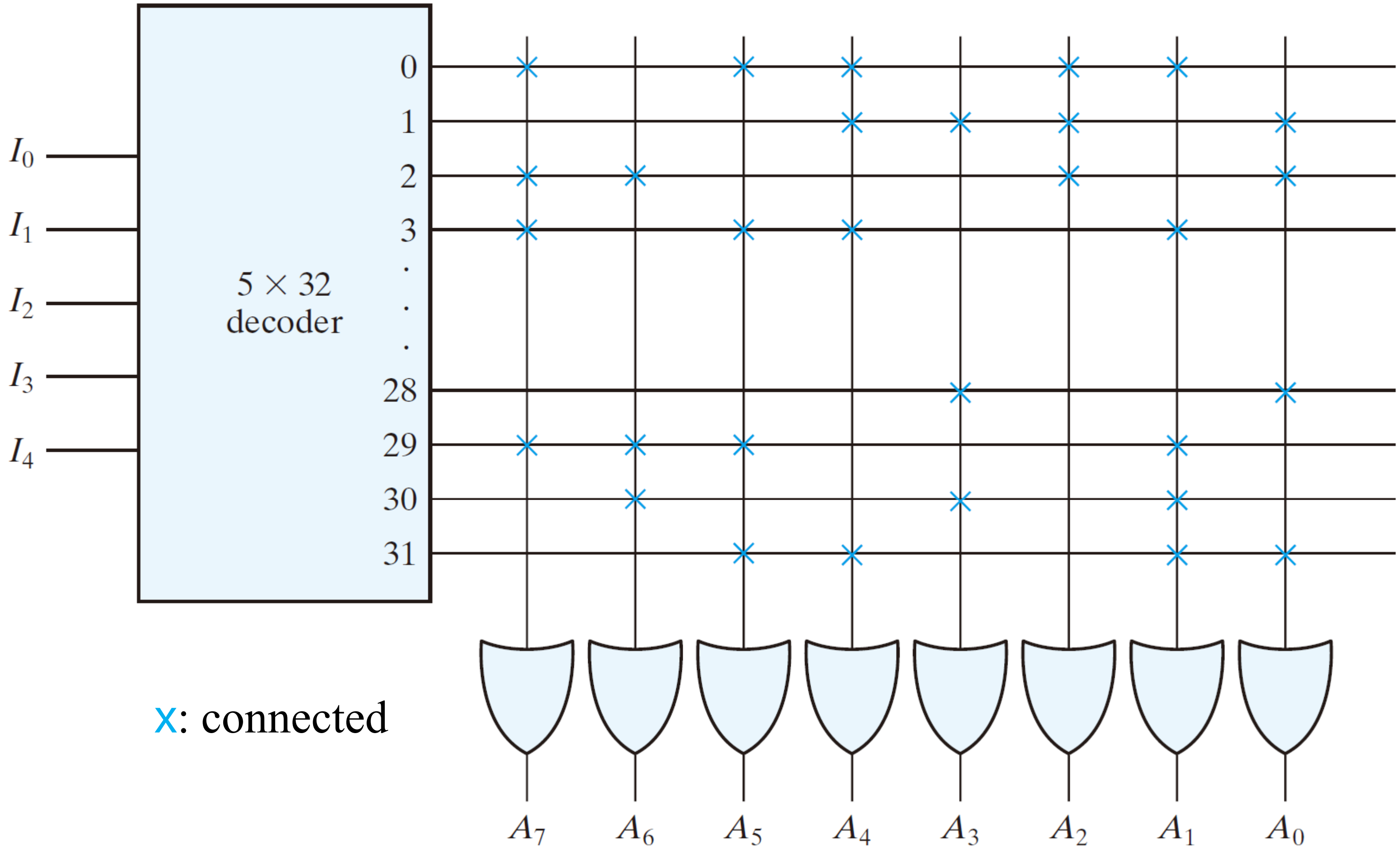
Example: 32 x 8 ROM

- 5-to-32 decoder
- 8 OR gates each has 32 inputs
- 32 x 8 internal programmable connections



Programming ROM

- A fused-switch in each cross-point: close (two lines are connected) or open
- The fuse that can be blown by applying a high voltage pulse.



what are stored in these words? see next slide

ROM Table

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

Combinational Circuit Implementation

- ROM: a decoder + OR gates
  - sum of minterms
  - a Boolean function => sum of minterms
  - for an *n*-input, *m*-output combinational circuit
    $\Rightarrow 2^n \times m$ ROM
  - the decoder is fixed, the inputs to OR gate is programmable
- Design procedure:
  1. determine the size of ROM
  2. obtain the programming truth table of the ROM
  3. the truth table = the fuse pattern

Example 7-1

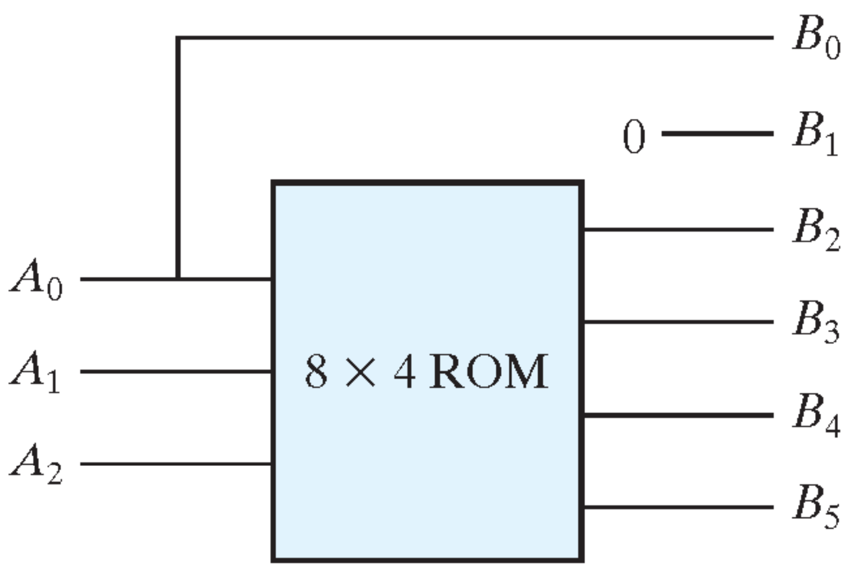
- Design a combinational circuit using a ROM. The circuit accepts a three-bit number and outputs a binary number equal to the square of the input number.
- 3 inputs and 6 outputs are required
- A partial truth table for the ROM

Truth Table for Circuit of Example 7.1

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

- $B_1 = 0$
- $B_0 = A_0$
- so, 8 x 4 ROM is enough

ROM Implementation for Example 7.1



ROM table

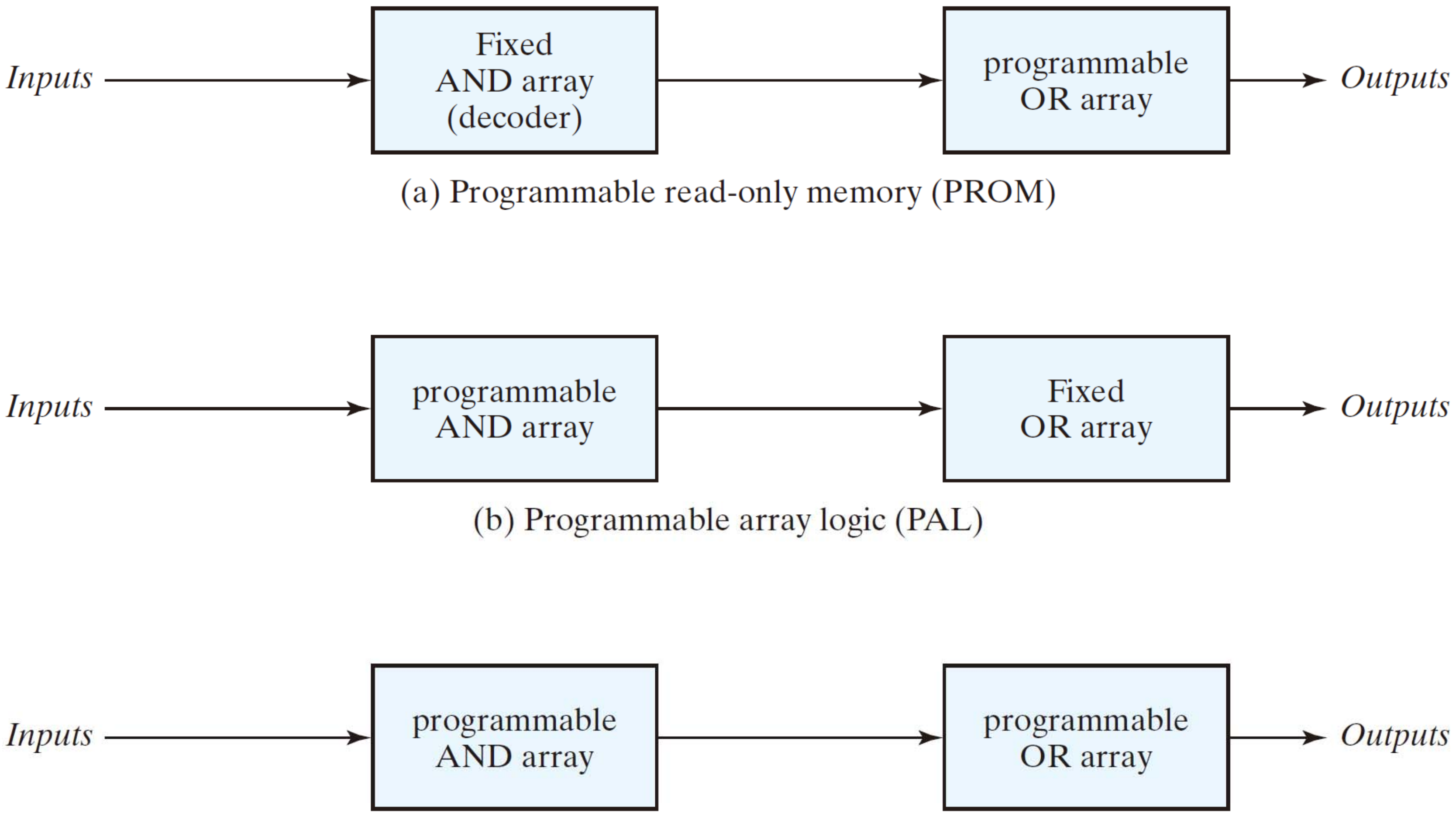
A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

Types of ROM

- Mask programming ROM
  - IC manufacturers
  - is economical only if large quantities
- PROM: Programmable ROM
  - fuses, one-time programmable (OTP)
  - universal programmer
- EPROM: Erasable PROM
  - floating gate (tunneling)
  - ultraviolet light erasable with transparent window
- EEPROM: Electrically Erasable PROM
  - longer time is needed to write
  - in system programmable (ISP)
- Flash memory
  - allowing simultaneous erasing of blocks of memory (flash erasable)
  - current mainstream product
  - NAND type and NOR type
  - very high density
- High voltage (> 10V) is required for non-mask programming

Combinational PLDs

- Three major types of combinational PLDs, differing in the placement of the programmable connections in the AND–OR array.



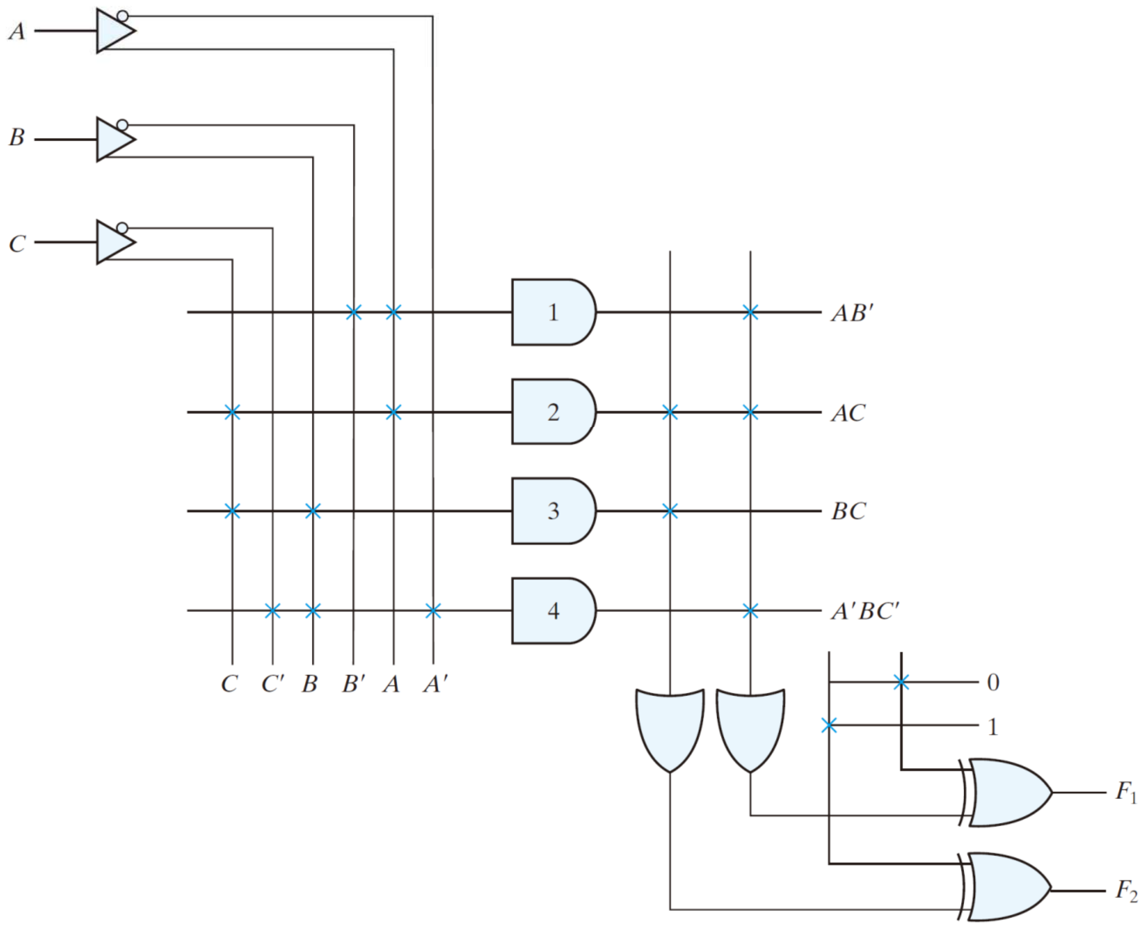
(a) Programmable read-only memory (PROM)

(b) Programmable array logic (PAL)

(c) Programmable logic array (PLA)

7.6 Programmable Logic Array (PLA)

- An array of programmable AND gates can generate any product terms of the inputs; and an array of programmable OR gates can generate the sums of the products.
- More flexible use less circuits than ROM.
- Example:
  - 3 inputs
  - 4 product terms
  - 2 outputs
  - XOR for inverting
  - $F_1 = AB' + AC + A'BC'$
  - $F_2 = (AC + BC)'$



PLA Programming Table

$F_1 = AB' + AC + A'BC'$ $F_2 = (AC + BC)'$

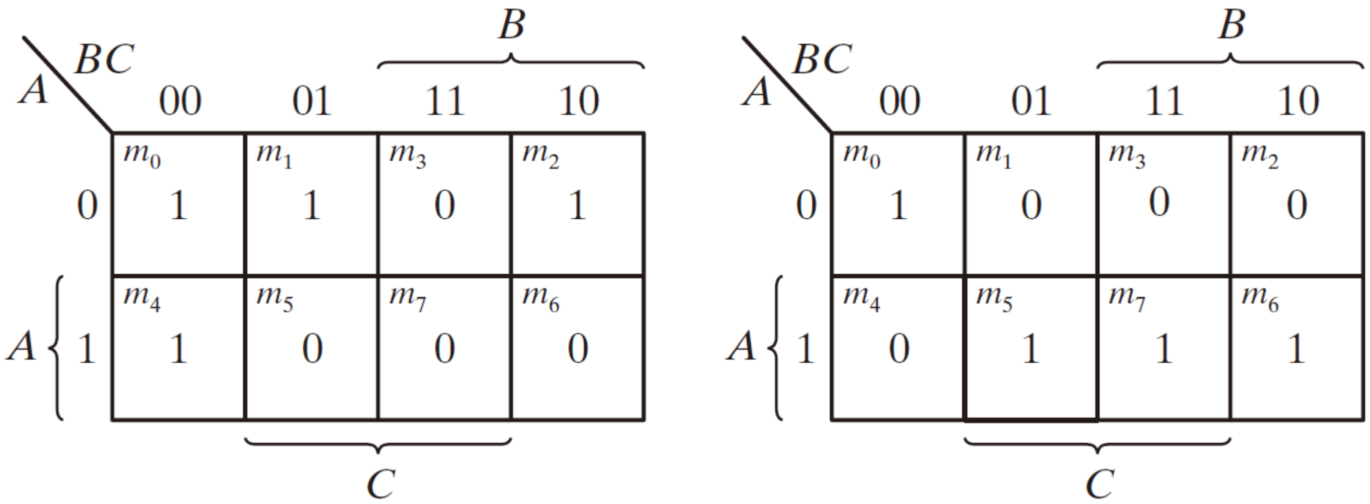
PLA Programming Table

		Inputs			Outputs (T)	Outputs (C)
	Product Term	A	B	C	F_1	F_2
AB'	1	1	0	—	1	—
AC	2	1	—	1	1	1
BC	3	—	1	1	—	1
$A'BC'$	4	0	1	0	1	—

- 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 the PLA consists of *n* buffer–inverter gates, *k* AND gates, *m* OR gates, and *m* XOR gates.
- There are 2*n* x *k* connections between the inputs and the AND array, *k* x *m* connections between the AND and OR arrays, and *m* connections associated with the XOR gates.

Examples 7-2

- Use PLA to implement $F_1(A, B, C) = \Sigma\ (0, 1, 2, 4)$; $F_2(A, B, C) = \Sigma\ (0, 5, 6, 7)$
- Both the true value and the complement of the function should be simplified to check to reduce the number of distinct product terms (AND gates needed).
- The number of literals in a term is not important in PLA design
- The K-map for the simplification



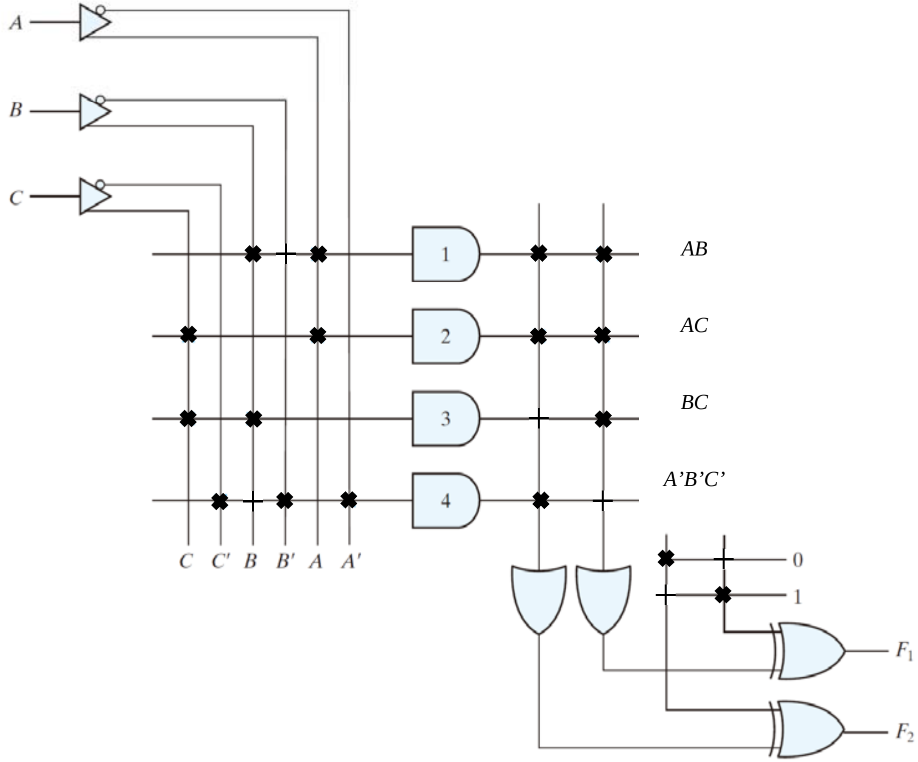
$F_1 = (AB + AC + BC)'$

$F_2 = AB + AC + A'B'C'$

- Only four distinct product terms: *AB, AC, BC, and A'B'C'*.

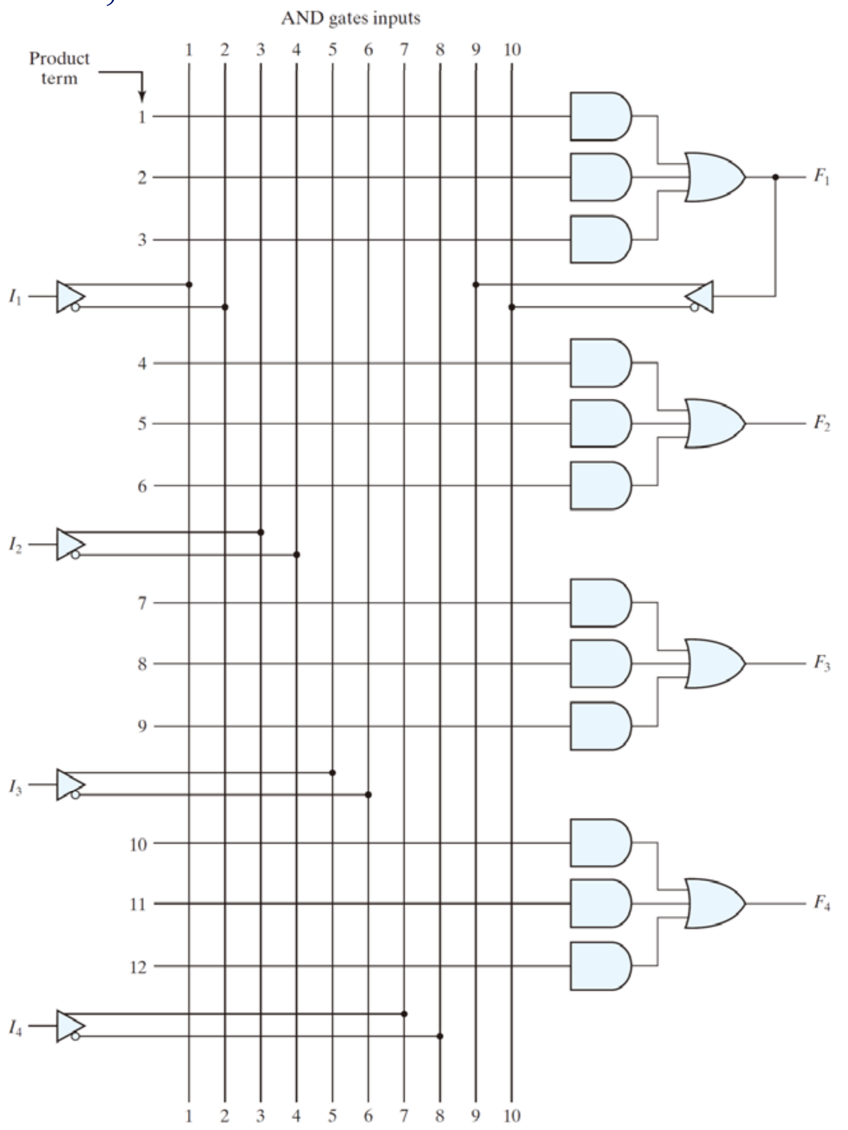
PLA programming table

	Product term	Inputs			Outputs (C)	Outputs (T)
		A	B	C	F_1	F_2
AB	1	1	1	–	1	1
AC	2	1	–	1	1	1
BC	3	–	1	1	1	–
$A'B'C'$	4	0	0	0	–	1



7-7 Programmable Array Logic (PAL)

- PAL has a fixed OR array and a programmable AND array.
- Easier to program than, but is not as flexible as, the PLA.
- Example: PAL with 4 inputs and 4 outputs.
- There are four sections of *three wide* AND–OR array, i.e. there are three programmable AND gates in each section and one fixed OR gate.
- The Boolean functions must be simplified to fit into each section.
- One of the outputs is fed back into the AND gates for implementing functions with a large number of product terms.
- Unlike PLA, a product term cannot be shared among two or more OR gates; therefore, each function can be simplified by itself.



An Example Implementation

- Using a PAL to design the following Boolean functions

  $w(A,B,C,D) = \Sigma(2,12,13)$

  $x(A,B,C,D) = \Sigma(7,8,9,10,11,12,13,14)$

  $y(A,B,C,D) = \Sigma(0,2,3,4,5,6,7,8,10,11,15)$

  $z(A,B,C,D) = \Sigma(1,2,8,12,13)$

- Simplify the functions

  $w = ABC' + A'B'CD'$

  $x = A + BCD$

  $y = A'B + CD + B'D'$

  $z = ABC' + A'B'CD' + AC'D' + A'B'C'D$

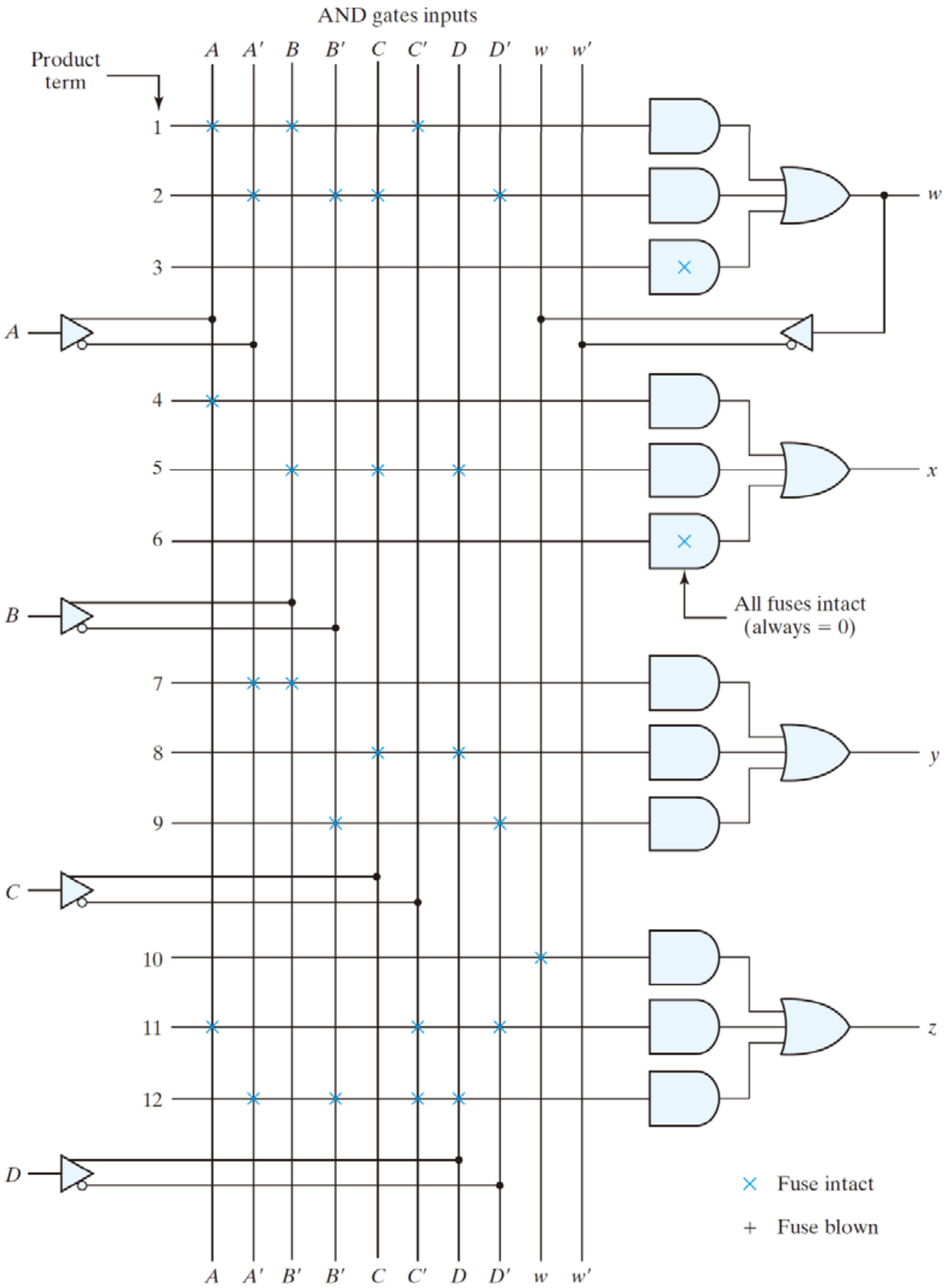
  $\quad = w + AC'D' + A'B'C'D$

  z has four product terms and can be reduced by using w.

PAL Programming Table

PAL Programming Table

Product Term	AND Inputs					Outputs
	A	*B*	*C*	*D*	*w*	
1	1	1	0	—	—	$w = ABC' + A'B'CD'$
2	0	0	1	0	—	
3	—	—	—	—	—	
4	1	—	—	—	—	$x = A + BCD$
5	—	1	1	1	—	
6	—	—	—	—	—	
7	0	1	—	—	—	$y = A'B + CD + B'D'$
8	—	—	1	1	—	
9	—	0	—	0	—	
10	—	—	—	—	1	$z = w + AC'D' + A'B'C'D$
11	1	—	0	0	—	
12	0	0	0	1	—	



$$w = ABC' + A'B'CD'$$

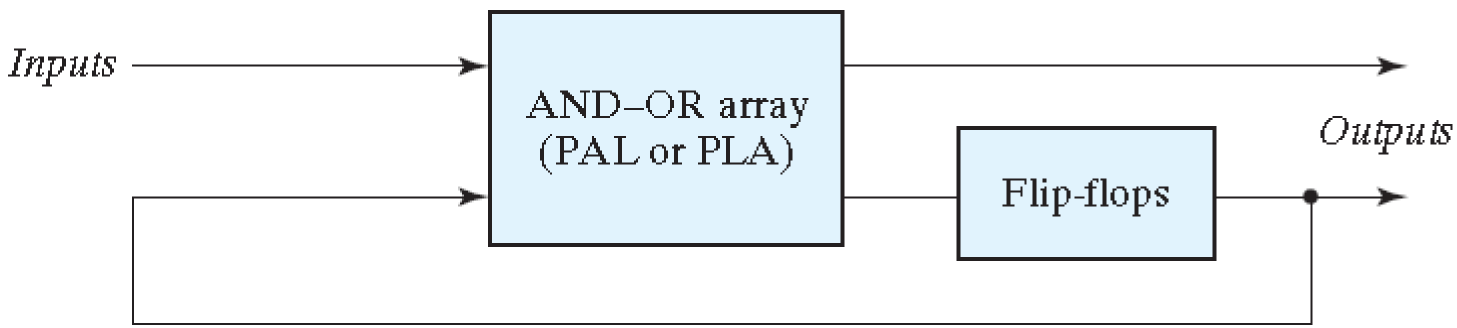
$$x = A + BCD$$

$$y = A'B + CD + B'D'$$

$$z = w + AC'D' + A'B'C'D$$

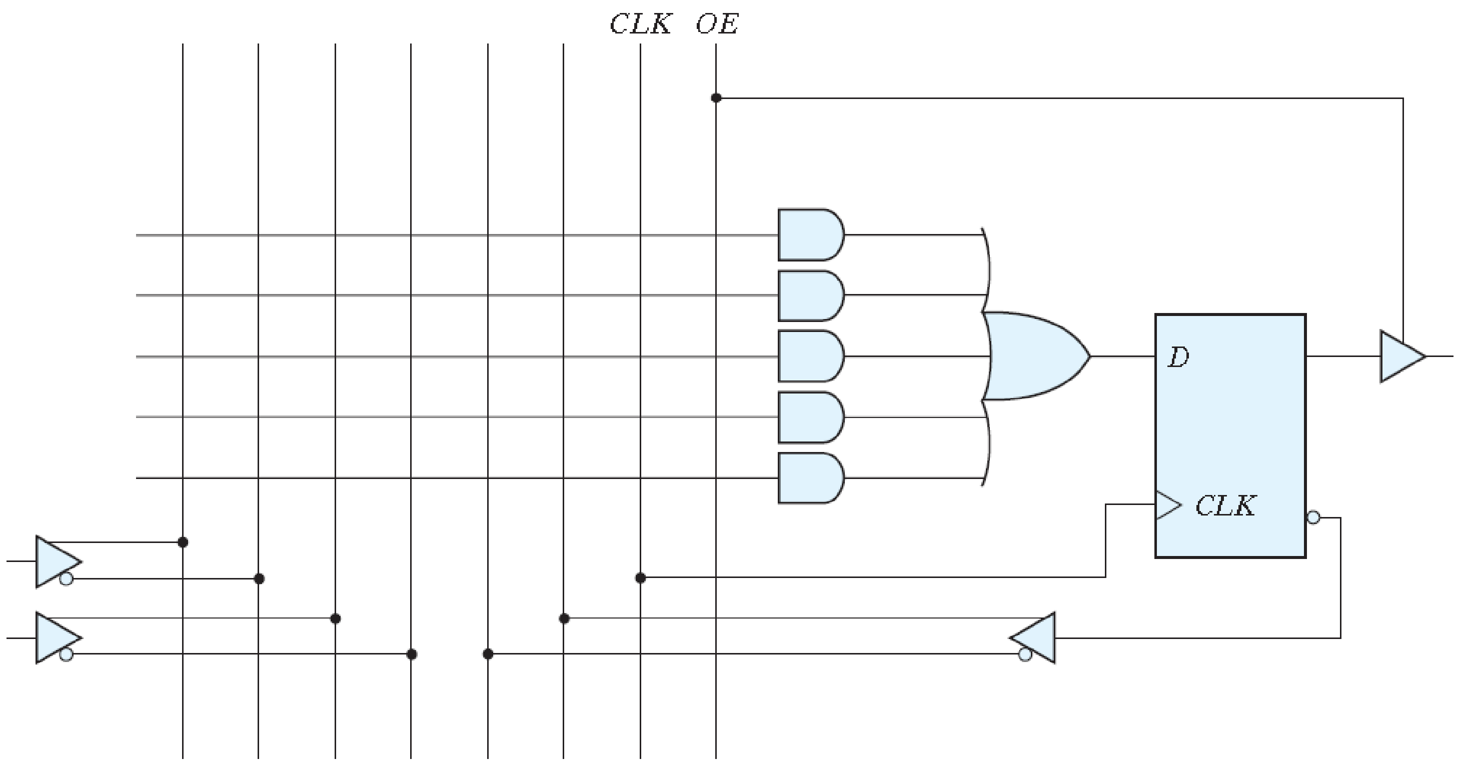
7-8 Sequential Programmable Devices

- Sequential programmable devices include both gates and flip-flops.
- Three major types:
  1. Sequential (or simple) programmable logic device (SPLD)
  2. Complex programmable logic device (CPLD)
  3. Field-programmable gate array (FPGA)



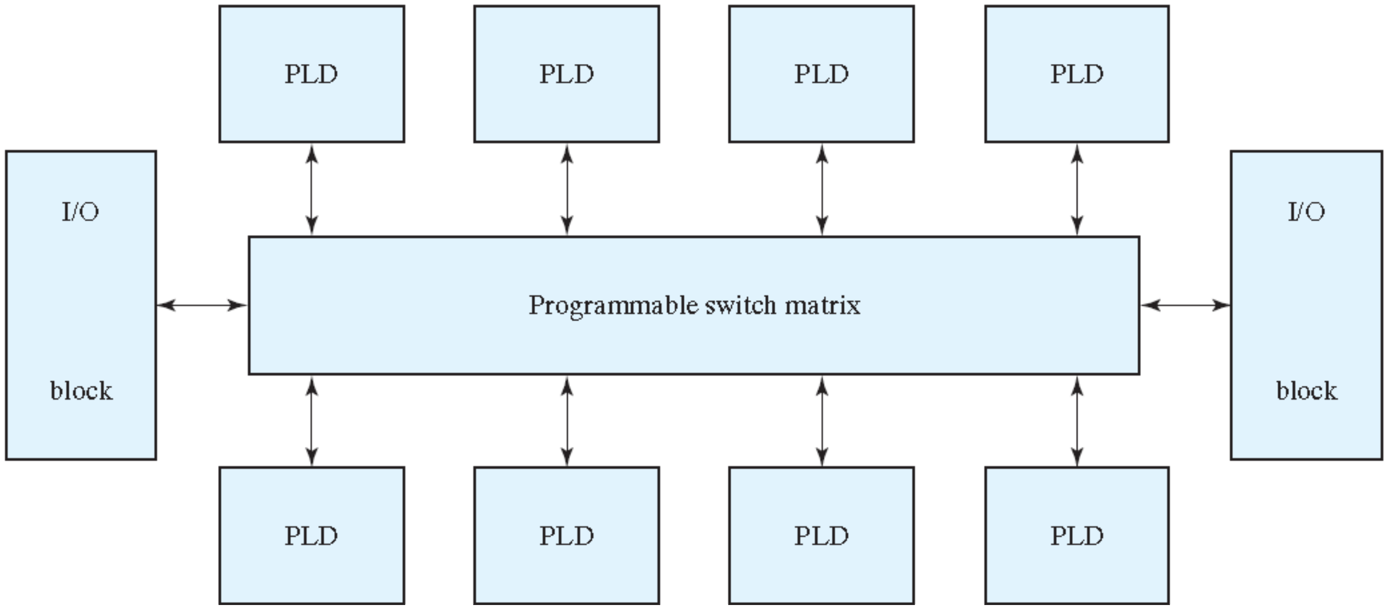
SPLD and its Macrocell

- The sequential PLD is sometimes referred to as a simple PLD to differentiate it from the complex PLD.
- A PAL that includes flip-flops is referred to as a *registered* PAL.
- Each section of an SPLD is called a *macrocell*, which is a circuit that contains a sum-of-products combinational logic function and an optional flip-flop.
- A basic macrocell: PAL + edge-triggered *D*-FF + tri-state output



Complex PLD

- CPLD, which is a collection of individual PLDs on a single IC, is more economical to use than SPLD.
- A programmable interconnection structure (switch matrix) allows the PLDs to be connected to each other.
- The input–output (I/O) blocks provide the connections to the IC pins which are driven by a three state buffer and can be programmed to act as input or output.

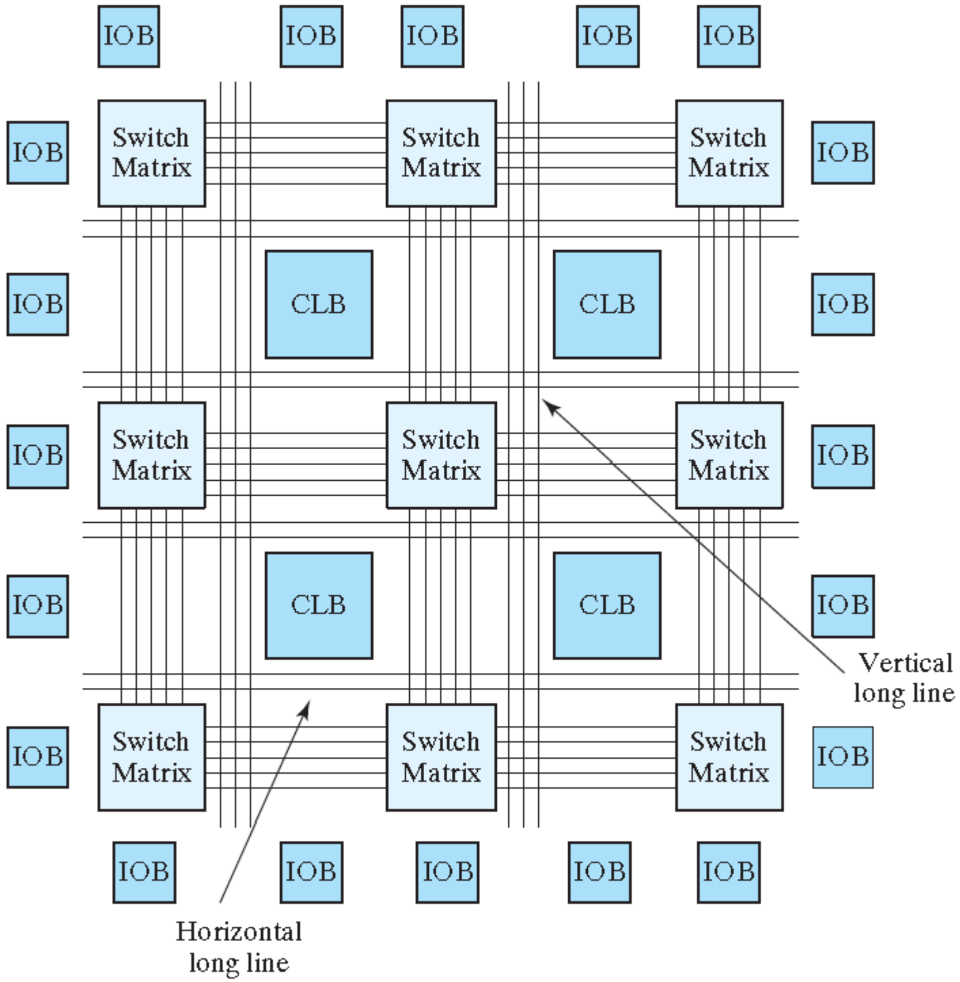


Field-Programmable Gate Array (FPGA)

- Gate array is an IC design methodology with several hundred thousand gates (transistors) fabricated in a single chip while their interconnections are determined from the designer's specification in the last steps of foundry service.
- A field-programmable gate array (FPGA) is a gate array like IC chip that can be programmed at the user's location.
- A typical FPGA consists of an array of millions of logic blocks (lookup tables, multiplexers, gates, and flip-flops), surrounded by programmable input and output blocks and connected together via programmable interconnections.
- A lookup table is a truth table stored in an SRAM and provides the combinational circuit functions for the logic block, in the same way that combinational circuit functions are implemented with ROM. For example, a 16 x 2 SRAM can store the truth table of a combinational circuit that has four inputs and two outputs.
- The combinational logic section, along with a number of programmable multiplexers, is used to configure the input equations for the flip-flop and the output of the logic block.
- There are different ways to store the truth table including SRAM, PROM, EEPROM, Flash, … .

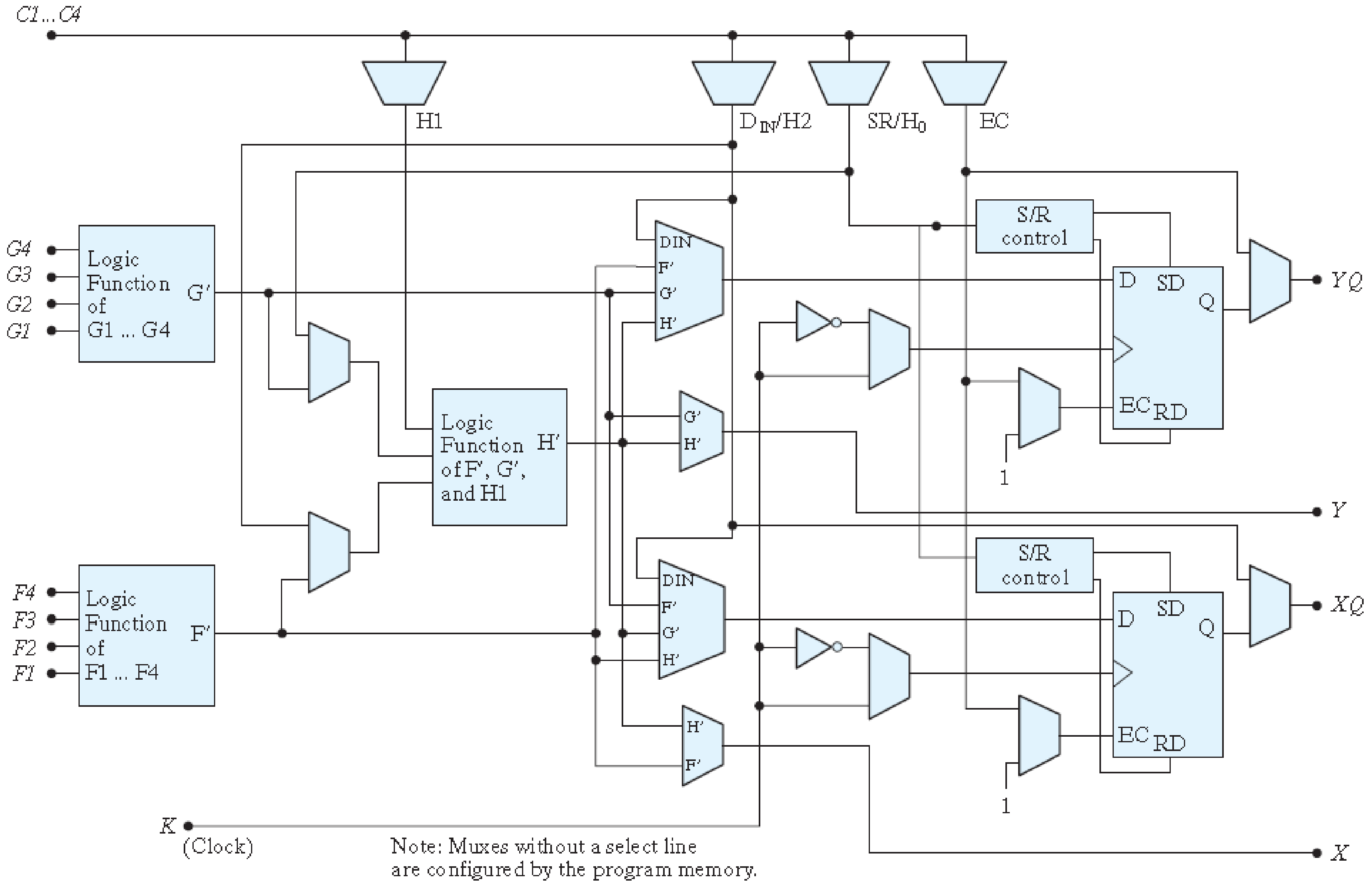
Basic Xilinx Architecture

- Xilinx launched the world's first commercial FPGA in 1985, with the vintage XC2000 device family.
- The XC3000 and XC4000 families soon followed, setting the stage for today's Spartan™, and Virtex™ device families.
- Each evolution improves in density, performance, power consumption, voltage levels, pin counts, and functionality.
- For example, the Spartan family of devices initially offered a maximum of 40K system gates, but today's Spartan-6 offers 150,000 logic cells plus 4.8Mb block RAM.
- Basic architecture of Spartan:
  CLB: configurable logic blocks
  IOB: I/O blocks



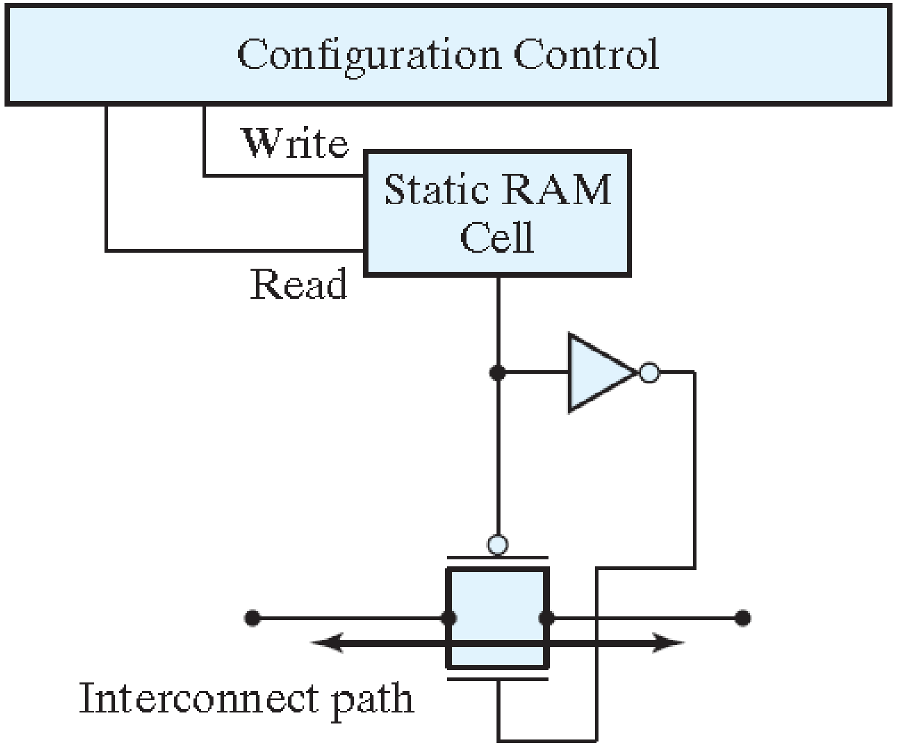
Configurable Logic Block (CLB)

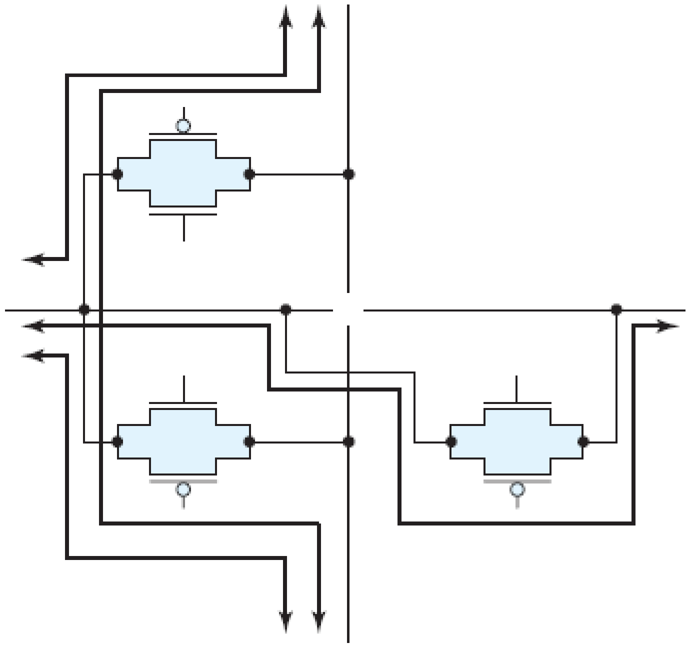
- Each CLB consists of a programmable lookup table, multiplexers, registers, and paths for control signals



Interconnect Resources

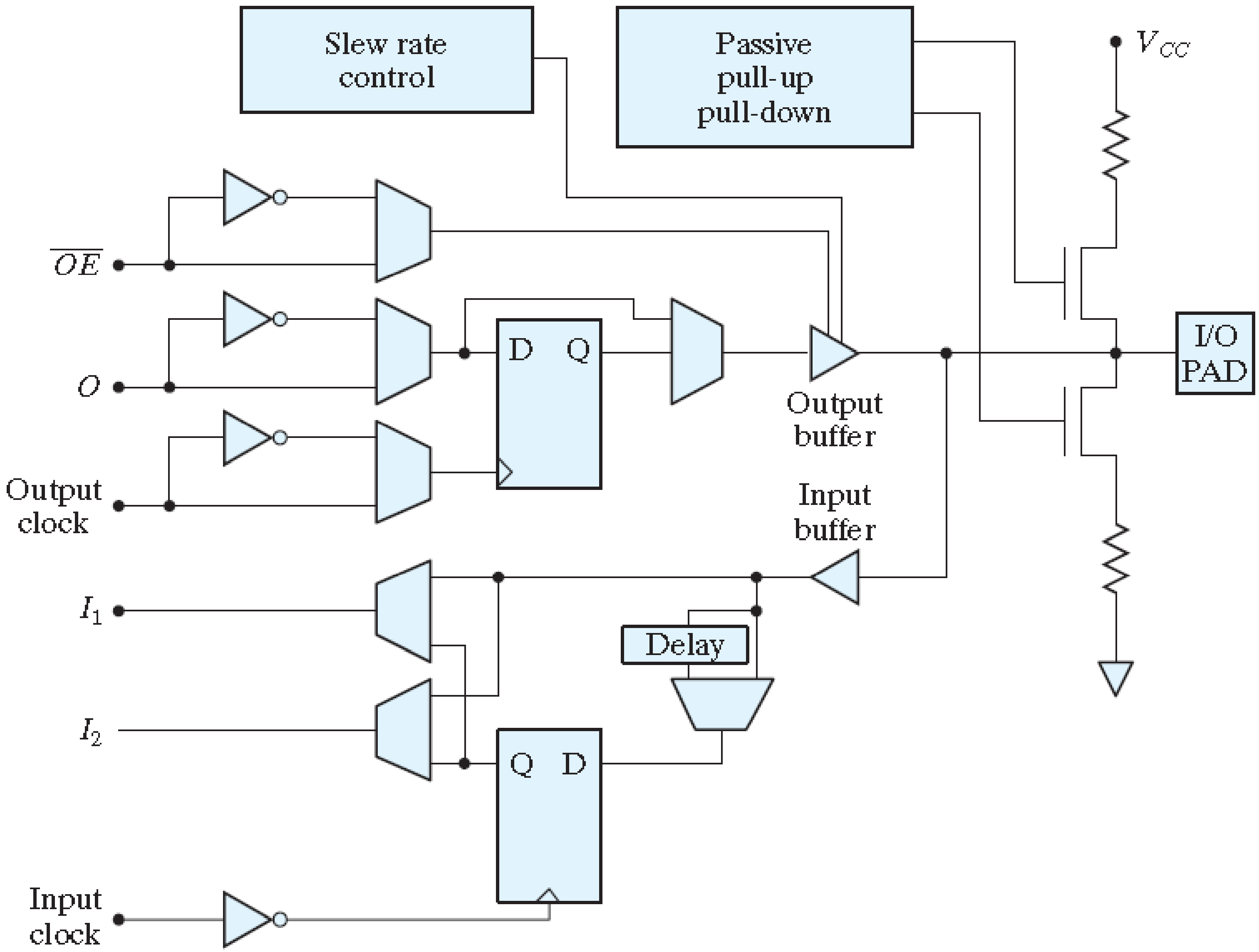
- A grid of switch matrices overlays the architecture of CLBs to provide general-purpose interconnect for branching and routing throughout the device.





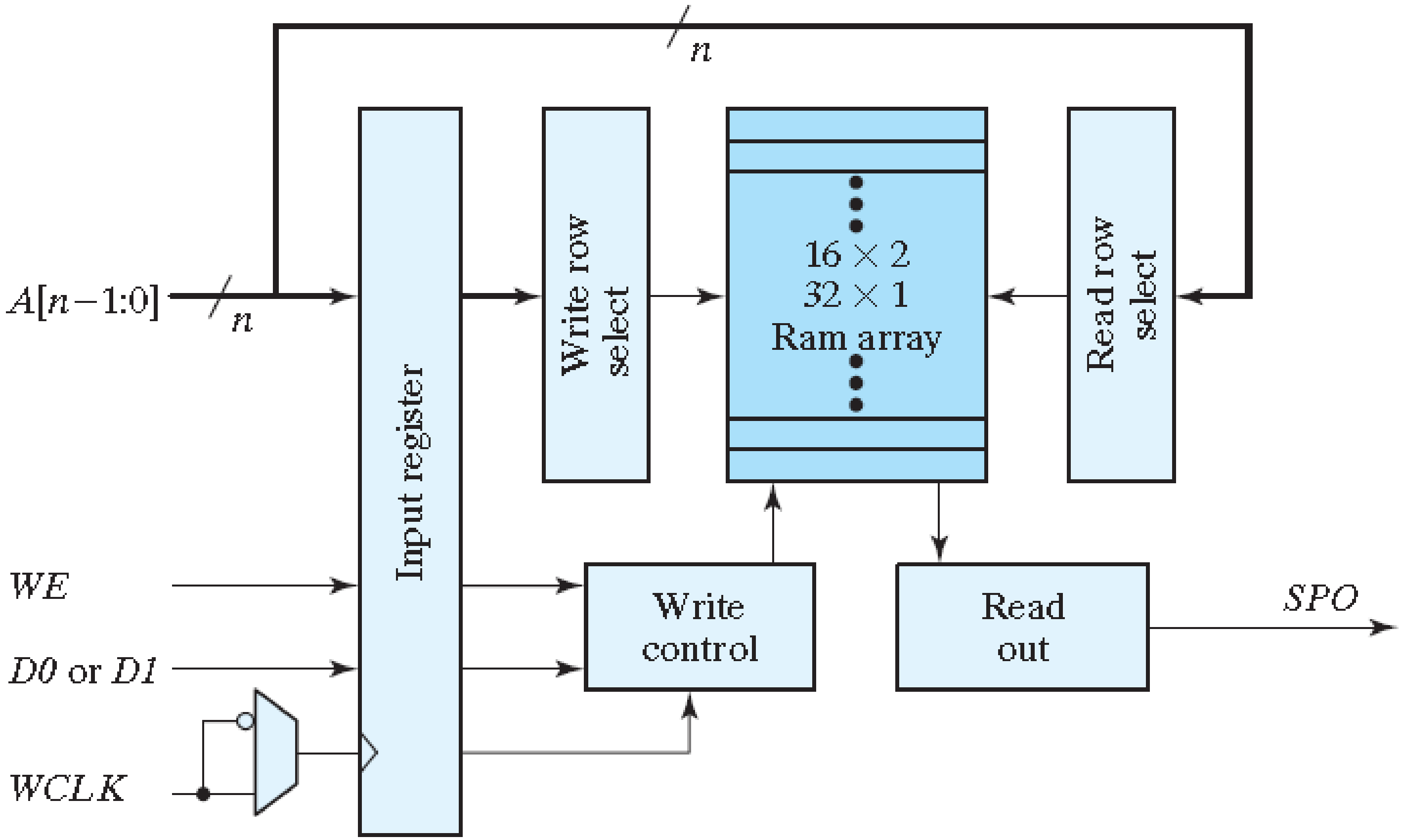
programmable interconnect points (PIPs)

I/O Block (IOB)

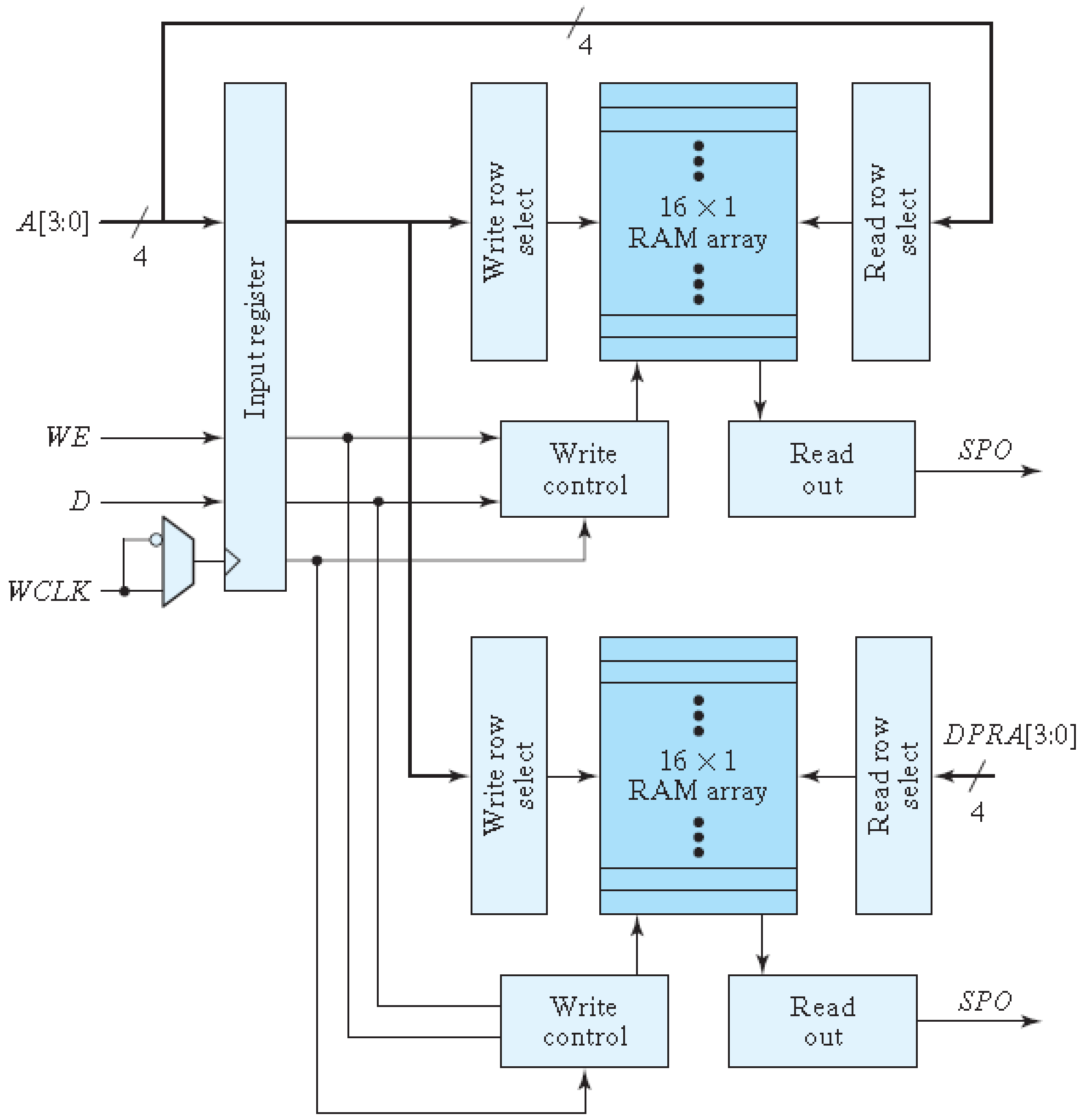

Slew rate control
Passive pull-up pull-down
V_{CC}
$\overline{OE}$
O
D Q
Output buffer
I/O PAD
Output clock
Input buffer
I_1
Delay
I_2
Q D
Input clock

Distributed RAM Cell from a Lookup Table

- Spartan can accommodate their on-chip distributed, dual-port, synchronous RAM (SelectRAM) to implement first-in, first-out register files (FIFOs), shift registers, and scratchpad memories, but their use reduces the CLBs available for logic.



Spartan Dual-port RAM



Xilinx Spartan XL FPGA Family

Attributes of the Xilinx Spartan XL Device Family

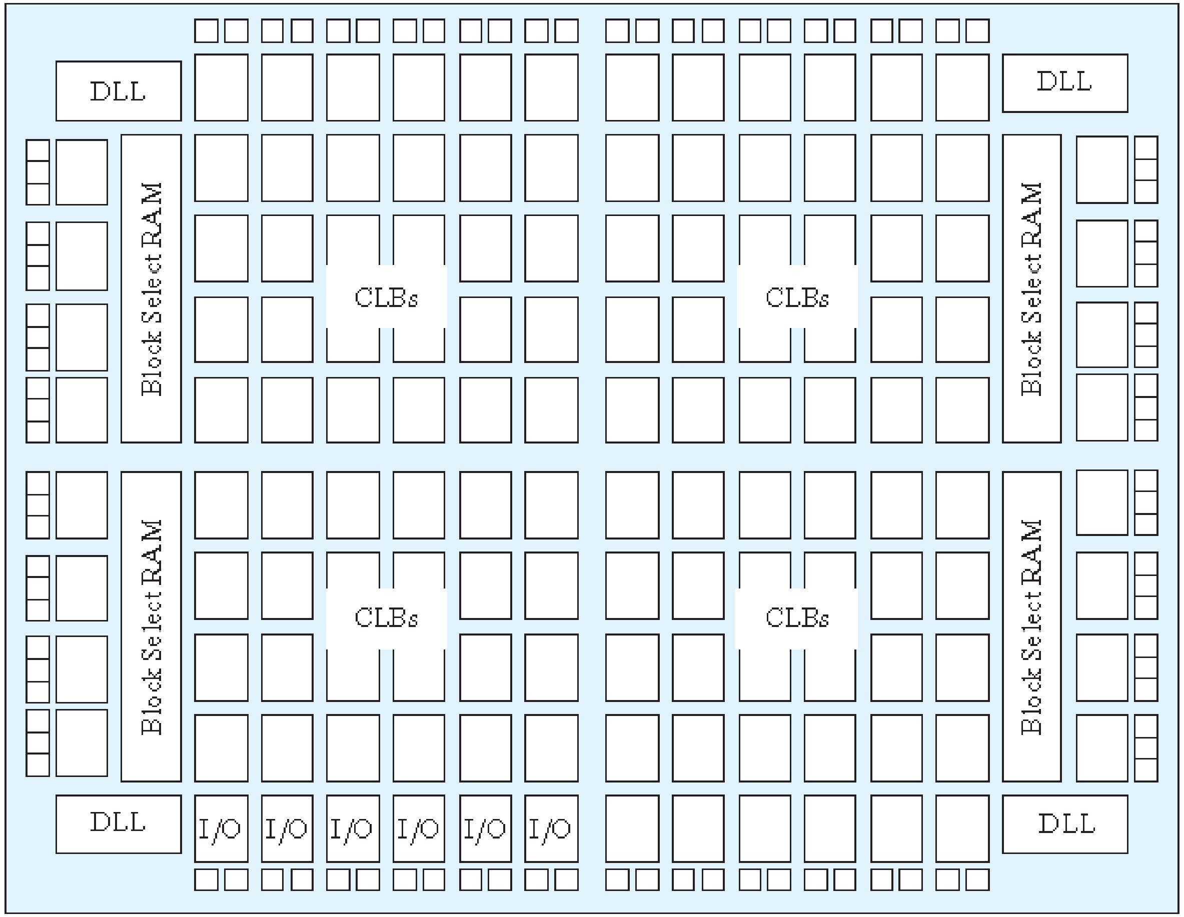
Spartan XL	XCS05/XL	XCS10/XL	XCS20/XL	XCS30/XL	XCS40/XL
System Gates[1]	2K–5K	3K–10K	7K–20K	10K–30K	13K–40K
Logic Cells[2]	238	466	950	1,368	1,862
Max Logic Gates	3,000	5,000	10,000	13,000	20,000
Flip-Flops	360	616	1,120	1,536	2,016
Max RAM Bits	3,200	6,272	12,800	18,432	25,088
Max Avail I/O	77	112	160	192	224

[1] 20–30% of CLBs as RAM.
[2] 1 Logic cell = four-input lookup table + flip-flop.

- Offer up to 80-MHz system performance, depending on the number of cascaded lookup tables, which reduce performance by introducing longer paths.

Xilinx Spartan II FPGAs

- Spartan II improves in speed (200-MHz), density (200,000 system gates) and operating voltage (2.5 V); as well as four other distinguished features: (1) on-chip block memory, (2) a novel architecture, (3) support for multiple I/O standards, and (4) delay locked loops (DLLs).



Xilinx Spartan II FPGAs

Spartan II Device Attributes

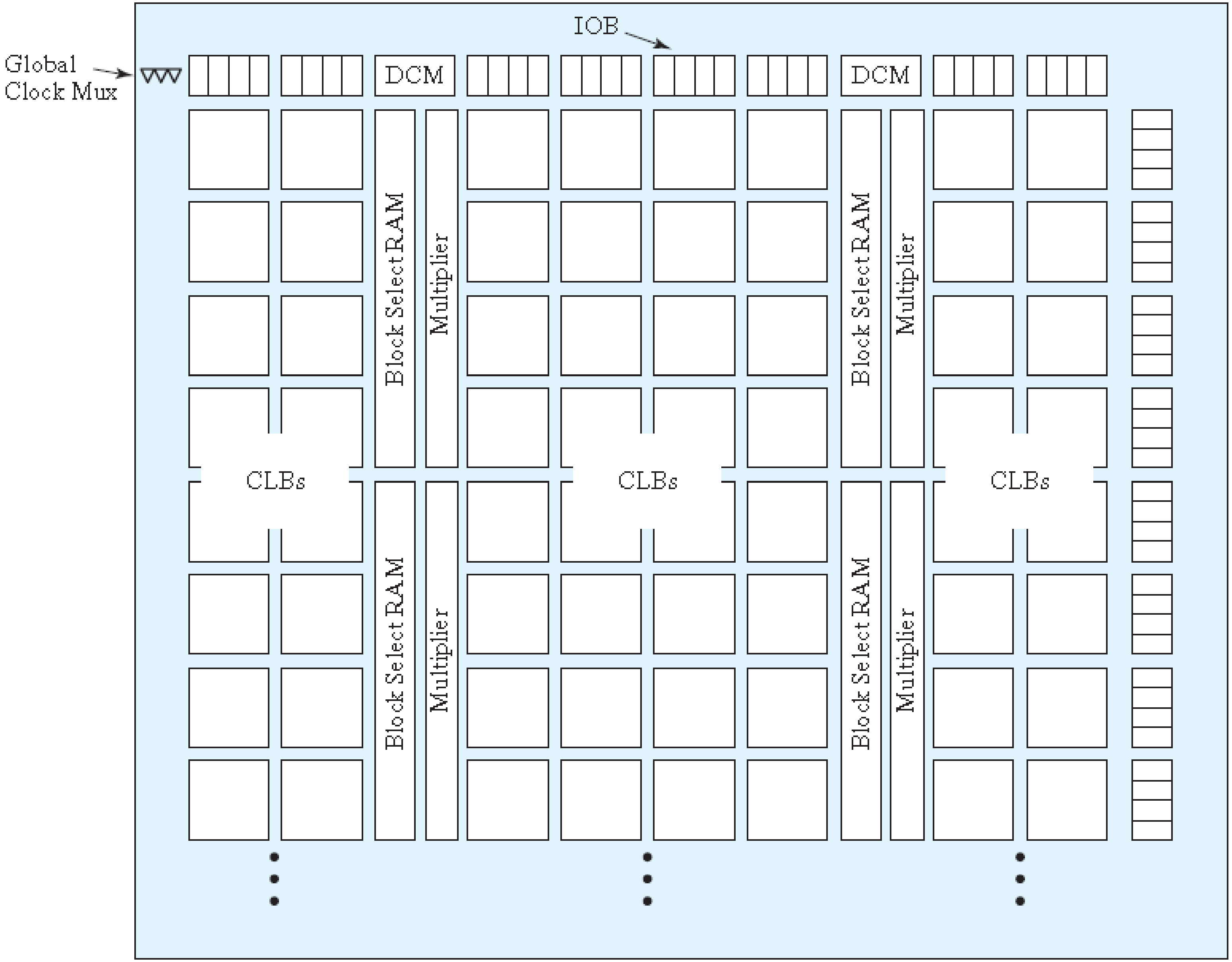
Spartan II FPGAs	XC2S15	XC2S30	XC2S50	XC2S100	XC2S150	XC2S200
System Gates[1]	6K–15K	13K–30K	23K–50K	37K–100K	52K–150K	71K–200K
Logic Cells[2]	432	972	1,728	2,700	3,888	5,292
Block RAM Bits	16,384	24,576	32,768	40,960	49,152	57,344
Max Avail I/O	86	132	176	196	260	284

[1] 20–30% of CLBs as RAM.
[2] 1 Logic cell = four-input lookup table + flip-flop.

Comparison of the Spartan Device Families

Part	Spartan	Spartan XL	Spartan II
Architecture	XC4000 Based	XC4000 Based	Virtex Based
Max # System Gates	5K–40K	5K–40K	15K–200K
Memory	Distributed RAM	Distributed RAM	Block + Distributed
I/O Performance	80 MHz	100 MHz	200 MHz
I/O Standards	4	4	16
Core Voltage	5 V	3.3 V	2.5 V
DLLs	No	No	Yes

Xilinx Virtex FPGAs



DCM: Clock Manager

ARM-Based Excalibur Embedded Processor

- 200-MHz ARM922T™ Processor
- Up to 3.3 Mbits of Memory
- Up to 1M Gates of Programmable Logic