Chapter 7

Memory and Programmable Logic

7-1

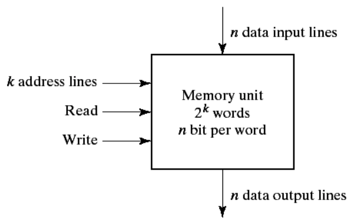
Outline

- Introduction
- Random-Access Memory
- Memory Decoding
- Error Detection and Correction
- Read-Only Memory
- Programmable Devices
- Sequential Programmable Devices

7-2

Mass Memory Elements

- Memory is a collection of binary cells together with associated circuits needed to transfer information to or from any desired location



- Two primary categories of memory:
  - Random access memory (RAM)
  - Read only memory (ROM)

7-3

Programmable Logic Device

- The binary information within the device can be specified in some fashion and then embedded within the hardware
  - Most of them are programmed by breaking the fuses of unnecessary connections
- Four kinds of PLD are introduced
  - Read-only memory (ROM)
  - Programmable logic array (PLA)
  - Programmable array logic (PAL)
  - Field-programmable gate array (FPGA)

7-4

Outline

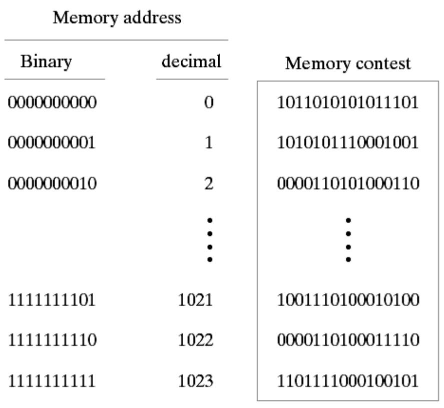
- Introduction
- Random-Access Memory
- Memory Decoding
- Error Detection and Correction
- Read-Only Memory
- Combinational Programmable Devices
- Sequential Programmable Devices

7-5

Random Access Memory

- A **word** is the basic unit that moves in and out of memory
  - The length of a word is often multiples of a byte (=8 bits)
- Memory units are specified by its **number of words** and the **number of bits** in each word
  - Ex: 1024(words) x 16(bits)
  - Each word is assigned a particular **address**, starting from 0 up to $2^k - 1$

    (k = number of address lines)



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

Fig. 7-3 Content of a 1024 × 16 Memory

7-6

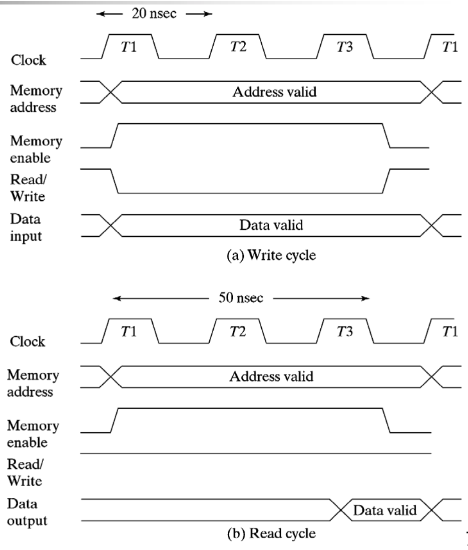
Write and Read Operations

- Write to RAM
  - Apply the binary address of the desired word to the **address lines**
  - Apply the data bits that must be stored in memory to the **data input lines**
  - Activate the **write control**
- Read from RAM
  - Apply the binary address of the desired word to the **address lines**
  - Activate the **read control**

7-7

Timing Waveforms

- CPU clock = 50 MHz
  - cycle time = 20 ns
- Memory access time = 50 ns
  - The time required to complete a read or write operation
- The control signals must stay active for at least 50 ns
  - 3 CPU cycles are required



7-8

Types of Memories

- Access mode:
  - Random access: any locations can be accessed in any order
  - Sequential access: accessed only when the requested word has been reached (ex: hard disk)
- Operating mode:
  - Static RAM (SRAM)
  - Dynamic RAM (DRAM)
- Volatile mode:
  - Volatile memory: lose stored information when power is turned off (ex: RAM)
  - Non-volatile memory: retain its storage after removal of power (ex: flash, ROM, hard-disk, …)

7-9

SRAM vs. DRAM

- Static RAM:
  - Use internal latch to store the binary information
  - Stored information remains valid as long as power is on
  - Shorter read and write cycles
  - Larger cell area and power consumption
- Dynamic RAM:
  - Use a capacitor to store the binary information
  - Need periodically refreshing to hold the stored info.
  - Longer read and write cycles
  - Smaller cell area and power consumption

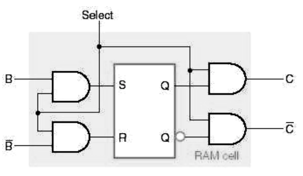


Fig. 6-5 Static RAM Cell

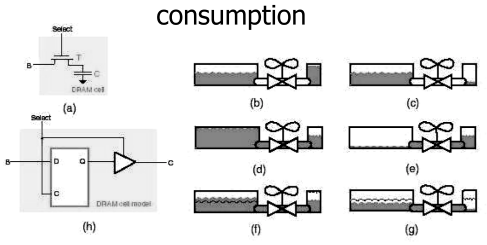


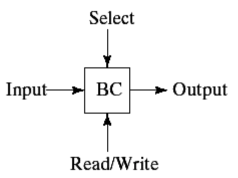
Fig. 6-12 Dynamic RAM Cell, Hydraulic Analogy of Cell Operation, and Cell Model

Outline

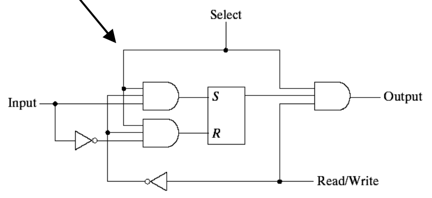
- Introduction
- Random-Access Memory
- Memory Decoding
- Error Detection and Correction
- Read-Only Memory
- Combinational Programmable Devices
- Sequential Programmable Devices

7-11

Memory Construction



A SRAM Cell



(a) Logic diagram

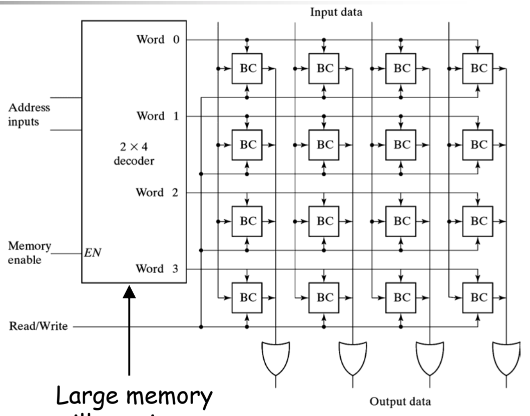


Fig. 7-6 Diagram of a 4 × 4 RAM

Large memory will require large decoder

7-12

Coincident Decoding

- Address decoders are often divided into two parts
  - A two-dimensional scheme
- The total number of gates in decoders can be reduced
- Can arrange the memory cells to a square shape
- EX: 10-bit address
  404 = 0110010100
  X = 01100 (first five)
  Y = 10100 (last five)

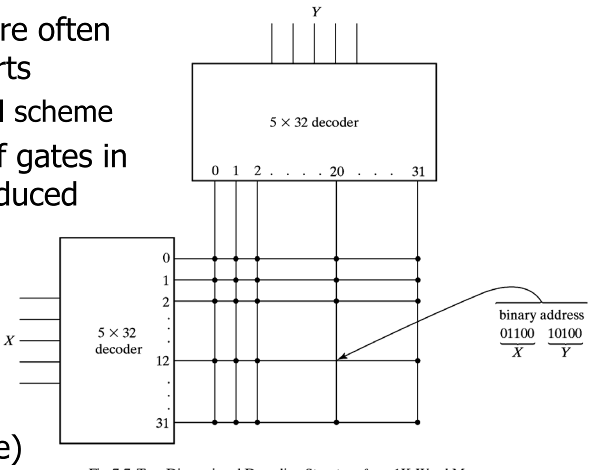


Fig. 7-7 Two-Dimensional Decoding Structure for a 1K-Word Memory

Address Multiplexing

- Memory address lines often occupy too much I/O pads
  - 64K = 16 lines
  - 256M = 28 lines
- Share the address lines of X and Y domains
  - Reduce the number of lines to a half
  - An extra register is required for both domain to store the address
- Two steps to send address
  - RAS=0: send row address
  - CAS=0: send column address

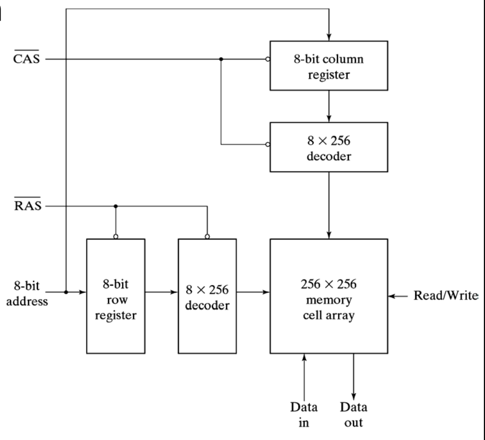


Fig. 7-8 Address Multiplexing for a 64K DRAM

Outline

- Introduction
- Random-Access Memory
- Memory Decoding
- $\boxed{\text{Error Detection and Correction}}$
- Read-Only Memory
- Combinational Programmable Devices
- Sequential Programmable Devices

7-15

Error Detection & Correction

- Memory arrays are often very huge
  - May cause occasional errors in data access
- Reliability of memory can be improved by employing error-detecting and correcting codes
- Error-detecting code: only check for the **existence** of errors
  - Most common scheme is the parity bit
- Error-correcting code: check the existence and **locations** of errors
  - Use multiple parity check bits to generate a **syndrome** that can indicate the erroneous bits
  - Complement the erroneous bits can correct the errors

7-16

Hamming Code (1/2)

- k parity bits are added to an n-bit data word
- The positions numbered as a **power of 2** are reserved for the parity bits
  - Ex: original data is 11000100 (8-bit)

  ⇒ Bit position: **1** **2** 3 **4** 5 6 7 **8** 9 10 11 12

  P_1 P_2 1 P_4 1 0 0 P_8 0 1 0 0
  - P1 = XOR of bits (3,5,7,9,11) = 0
    P2 = XOR of bits (3,6,7,10,11) =0
    P4 = XOR of bits (5,6,7,12) = 1
    P8 = XOR of bits (9,10,11,12) = 1
  - The composite word is 001110010100 (12-bit)

7-17

Hamming Code (2/2)

- When the 12 bits are read from memory, the parity is checked over the **same combination** of bits **including** the parity bit
  - C1 = XOR of bits (1,3,5,7,9,11)
    C2 = XOR of bits (2,3,6,7,10,11)
    C4 = XOR of bits (4,5,6,7,12)
    C8 = XOR of bits (8,9,10,11,12)
- (001110010100) → C = $C_8C_4C_2C_1$ = 0000 : no error
  (101110010100) → C = $C_8C_4C_2C_1$ = 0001 : bit 1 error
  (001100010100) → C = $C_8C_4C_2C_1$ = 0101 : bit 5 error

viewed as a binary number

7-18

General Rules of Hamming Code

- The number of parity bits:
  - The syndrome C with k bits can represent $2^k - 1$ error locations (0 indicates no error)
  - $2^k - 1 \geq n + k \rightarrow 2^k - 1 - k \geq n$

Number of Check Bits, k	Range of Data Bits, n
3	2-4
4	5-11
5	12-26
6	27-57
7	58-120

- The members of each parity bit:
  - C1(P1): have a "1" in **bit 1** of their location numbers
    1(000**1**), 3(001**1**), 5(010**1**), 7(011**1**), 9(100**1**), …
  - C2(P2): have a "1" in **bit 2** of their location numbers
    2(00**1**0), 3(00**1**1), 6(01**1**0), 7(01**1**1), 10(10**1**0), …
  - C: with parity bit;   P: without parity bit itself

7-19

Extension of Hamming Code

- Original Hamming code can detect and correct only a **single error**
  - Multiple errors are not detected
- Add an extra bit as the parity of total coded word
  - Ex: 001110010100P_{13} (P_{13}=XOR of bits 1 to 12)
  - Still single-error correction but double-error detection
- Four cases can occur:
  - If C=0 and P=0, no error occurred
  - If C$\neq$0 and P=1, single error occurred (can be fixed)
  - If C$\neq$0 and P=0, double error occurred (cannot be fixed)
  - If C=0 and P=1, an error occurred in the P_{13} bit

7-20

Outline

- Introduction
- Random-Access Memory
- Memory Decoding
- Error Detection and Correction
- Read-Only Memory
- Combinational Programmable Devices
- Sequential Programmable Devices

7-21

Read Only Memory

- A memory device that can permanently keep binary data
  - Even when power is turned off and on again
- For a 2^k x n ROM, it consists of
  - k inputs (address line) and n outputs (data)
  - 2^k words of n-bit each
  - A k x 2^k decoder (generate all minterms)
  - n OR gates with 2^k inputs
  - Initially, all inputs of OR gates and all outputs of the decoder are fully connected

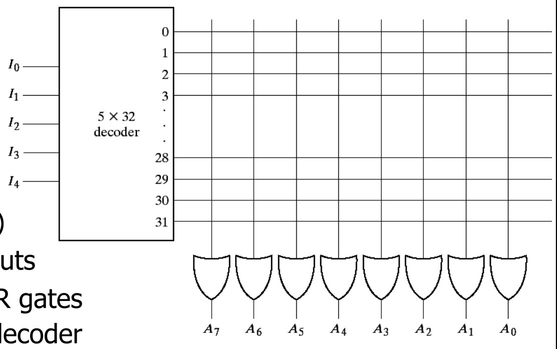


Fig. 7-10 Internal Logic of a 32 × 8 ROM

7-22

Programming the ROM

- Each intersection (crosspoint) in the ROM is often implemented with a fuse
- Blow out unnecessary connections according to the truth table
  - "1" means connected (marked as X)
  - "0" means unconnected
- Cannot recovered after programmed

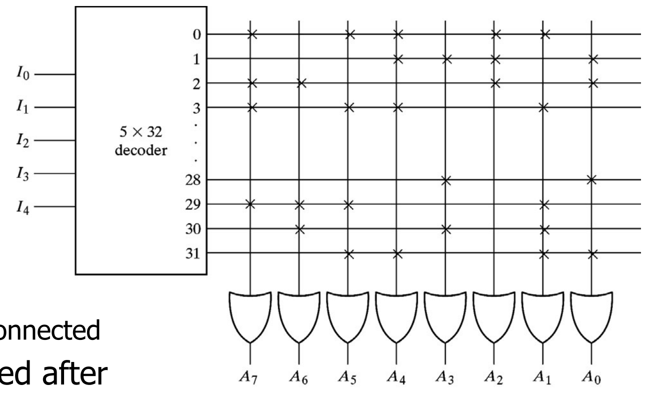


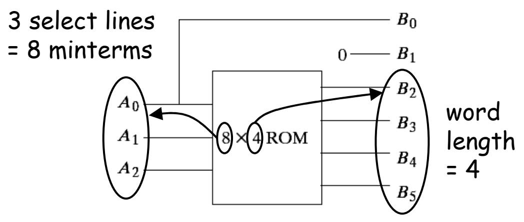
Fig. 7-11 Programming the ROM According to Table 7-3

7-23

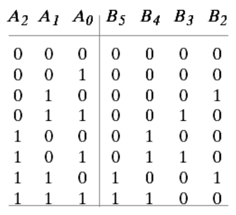
Design Comb. Circuit with ROM

- Derive the truth table of the circuit
- Determine minimum size of ROM
- Program the ROM

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



(a) Block diagram



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

(b) ROM truth table

7-24

Types of ROMs

- Mask programming
  - Program the ROM in the semiconductor factory
  - Economic for large quantity of the same ROM
- Programmable ROM (PROM)
  - Contain all fuses at the factory
  - Program the ROM by burning out the undesired fuses (irreversible process)
- Erasable PROM (EPROM)
  - Can be restructured to the initial state under a special ultra-violet light for a given period of time
- Electrically erasable PROM (EEPROM or E^{2}PROM)
  - Like the EPROM except being erased with electrical signals

7-25

Programmable Logic Devices

- ROM provides full decoding of variables
  - Waste hardware if the functions are given
- For known combinational functions, Programmable Logic Devices (PLD) are often used
  - Programmable read-only memory (PROM)
  - Programmable array logic (PAL)
  - Programmable logic array (PLA)
- For sequential functions, we can use
  - Sequential (simple) programmable logic device (SPLD)
  - Complex programmable logic device (CPLD)
  - Field programmable gate array (FPGA) $\leftarrow$ most popular

7-26

Outline

- Introduction
- Random-Access Memory
- Memory Decoding
- Error Detection and Correction
- Read-Only Memory
- Combinational Programmable Devices
- Sequential Programmable Devices

Configurations of Three PLDs

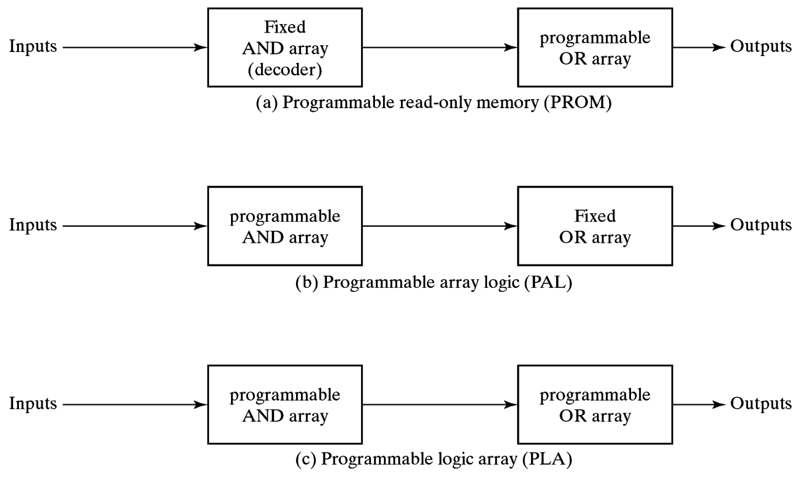


Fig. 7-13 Basic Configuration of Three PLDs

Programmable Logic Array

- PLA does not provide full decoding of the variables
  - Only generate the terms you need
- The decoder is replaced by an array of AND gates that can be programmed

F1 = AB′ + AC + A′BC′
F2 = (AC + BC)′

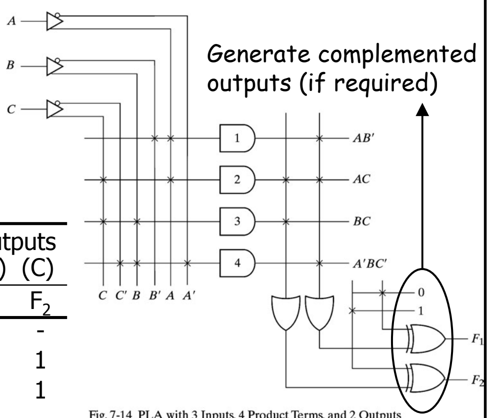


Fig. 7-14 PLA with 3 Inputs, 4 Product Terms, and 2 Outputs

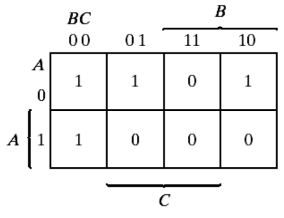
		Inputs			Outputs	
					(T)	(C)
	Product Term	A	B	C	F_1	F_2
AB’	1	1	0	-	1	-
AC	2	1	-	1	1	1
BC	3	-	1	1	-	1
A’BC’	4	0	1	0	1	-

Implementation with PLA

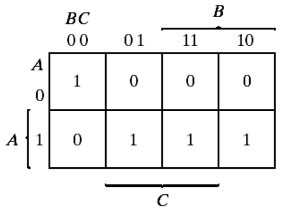
- Example 7-2: implement the two functions with PLA
  $F_1(A, B, C) = \Sigma\ (0, 1, 2, 4)$
  $F_2(A, B, C) = \Sigma\ (0, 5, 6, 7)$
- Goal: minimize the number of **distinct** product terms between two functions

PLA programming table

	Product term	Inputs A B C	Outputs (C) F_1	Outputs (T) 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



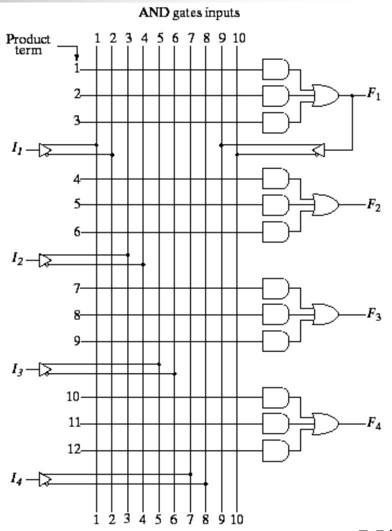
$F_1 = A'B' + A'C' + B'C'$
$F_1 = (AB + AC + BC)'$



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

Programmable Array Logic

- PAL has a fixed OR array and a programmable AND array
  - Easier to program but not as flexible as PLA
- Each input has a buffer-inverter gate
- One of the outputs is fed back as two inputs of the AND gates
- Unlike PLA, a product term cannot be shared among gates
  - Each function can be simplified by itself without common terms

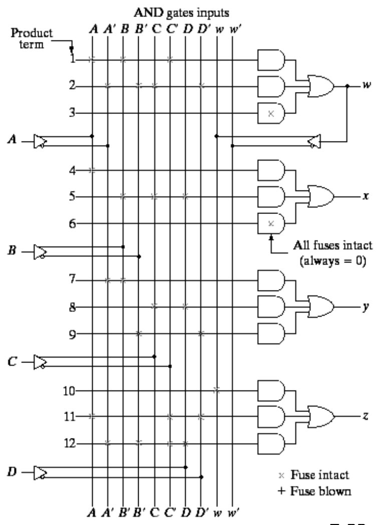


7-31

Implementation with PAL

w=Σ(2,12,13) x=Σ(7,8,9,10,11,12,13,14,15)
y=Σ(0,2,3,4,5,6,7,8,10,11,15) z=Σ(1,2,8,12,13)

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



7-32

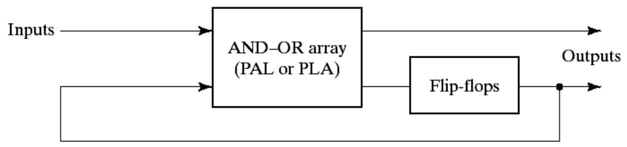
Outline

- Introduction
- Random-Access Memory
- Memory Decoding
- Error Detection and Correction
- Read-Only Memory
- Combinational Programmable Devices
- Sequential Programmable Devices

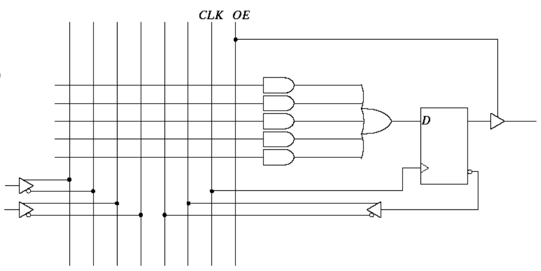
7-33

Sequential PLD

- The most simple sequential PLD = PLA (PAL) + Flip-Flops



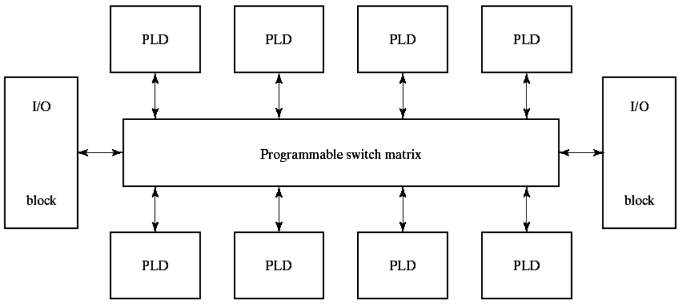
- The mostly used configuration for SPLD is constructed with 8 to 10 macrocells as shown right



7-34

Complex PLD

- Complex digital systems often require the connection of several devices to produce the complex specification
  - More economical to use a complex PLD (CPLD)
- CPLD is a collection of individual PLDs on a single IC with programmable interconnection structure



7-35

Field Programmable Gate Array

- Gate array: a VLSI circuit with some pre-fabricated gates repeated thousands of times
  - Designers have to provide the desired interconnection patterns to the manufacturer (factory)
- A field programmable gate array (FPGA) is a VLSI circuit that can be programmed in the user’s location
  - Easier to use and modify
  - Getting popular for fast and reusable prototyping
- There are various implementations for FPGA
  - More introductions are adopted from “Logic and Computer Design Fundamentals”, 2nd Edition Updated, by M. Morris Mano and Charles R. Kime, Prentice-Hall, 2001

7-36

FPGA Structure (Altera)

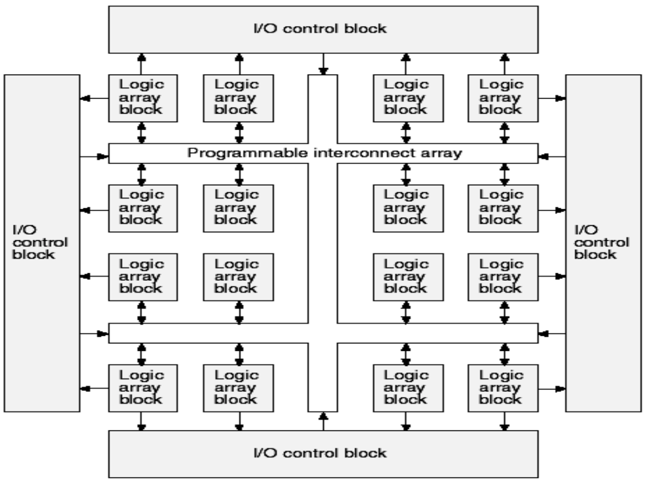


Fig. 6-28 Altera® MAX 7000™ Structure (Reprinted with Permission of Altera Corporation,© Altera Corp., 1991)

FPGA Structure (Xilinx)

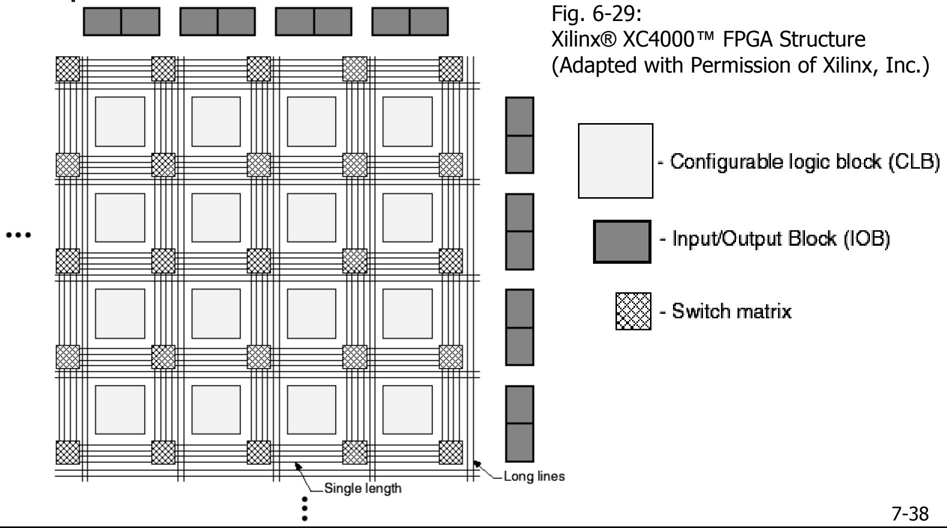
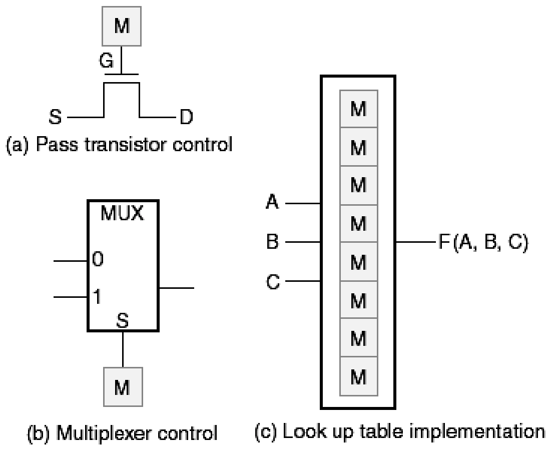


Fig. 6-29:
Xilinx® XC4000™ FPGA Structure
(Adapted with Permission of Xilinx, Inc.)

Store the Programming Info.

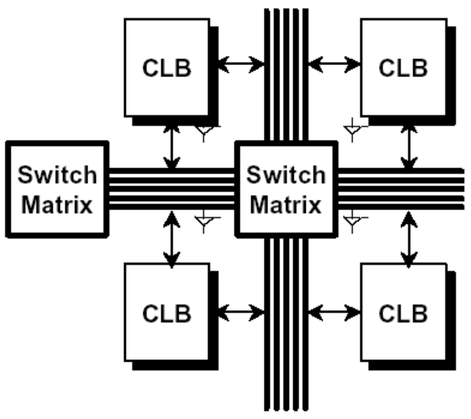
- SRAM technology is used
  - M = 1-bit SRAM
  - Loaded from the PROM after power on
- Store control values
  - Control pass transistor
  - Control multiplexer
- Store logic functions
  - Store the value of each minterm in the truth table



7-39

Xilinx FPGA Routing

- Fast direct interconnect
  - Adjacent CLBs
- General purpose interconnect
  - CLB – CLB or CLB – IOB
  - Through switch matrix
- Long lines
  - Across whole chip
  - High fan-out, low skew
  - Suitable for global signals (CLK) and buses
  - 2 tri-states per CLB for busses



7-40

Xilinx Switch Matrix

- Six pass transistors to control each switch node
- The two lines at point 1 are joined together
- At point 2, two distinct signal paths pass through one switch node

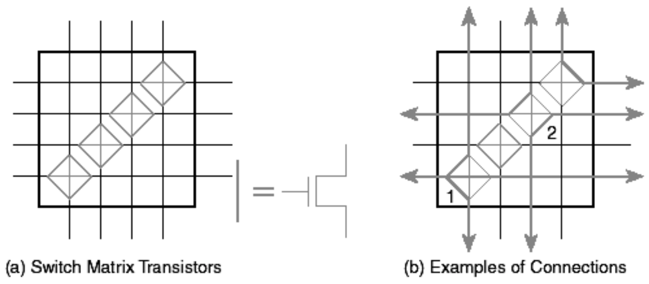
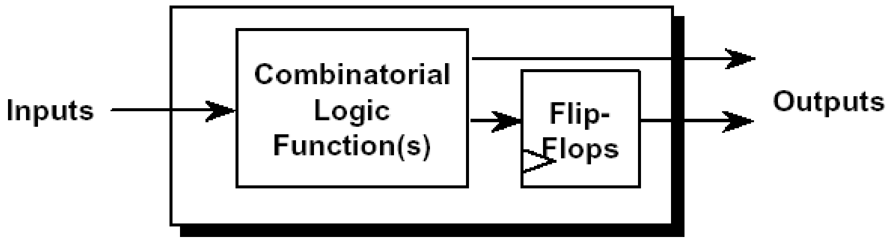


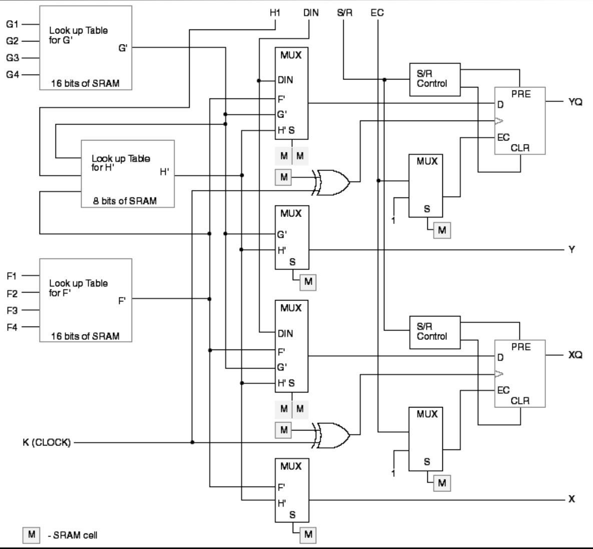
Fig. 6-31 Example of Xilinx® Switch Matrix (Adapted with Permission of Xilinx®, Inc.)

Configurable Logic Block (CLB)

- Combinational logic via lookup table
  - Any function(s) of available inputs
- Output registered and/or combinational



Simplified CLB Structure



7-43

Internal Functions of a CLB

- Two 4-input tables implement two distinct functions (F′ and G′)
- F’ and G’ with another control (H1) feed into a third lookup table (H′)
- Two arbitrary functions of up to four variables and selected functions of up to nine variables can be implemented
- Properly setting the two MUXes can assign any pair of F′, G′, and H′ to the two combinational outputs (X and Y)

7-44

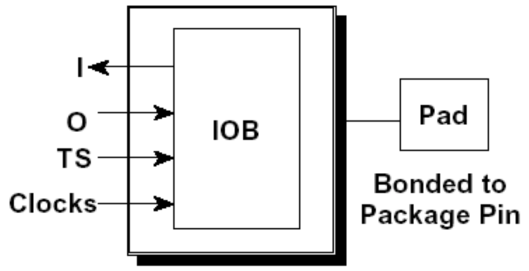
Internal Functions of a CLB

- Two D flip-flops directly drive outputs XQ and YQ
- Each of the D inputs can be selected from F′, G′, H′ and input DIN
- Two XORs select each flip-flop individually to be positive or negative edge triggered
- Two SR controls select the signal S/R to be an asynchronous Set or Reset for the flip-flops
- Two multiplexers allow the input EC to optionally act as a clock ENABLE signal for each flip-flop

7-45

I/O Block (IOB)

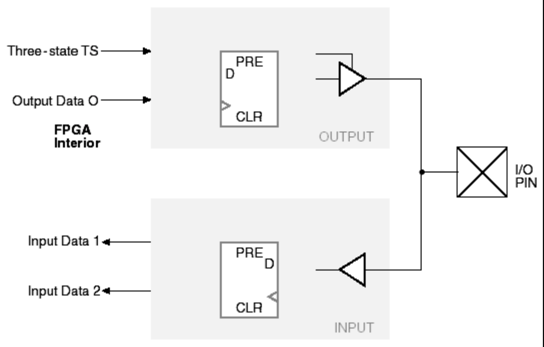
- Periphery of identical I/O blocks
  - Input, output, or bidirectional
  - Registered, latched, or combinational
  - Three-state output
  - Programmable output slew rate



7-46

Input/Output Mode of an IOB

- Input
  - 3-state control places the output buffer into high impedance
  - Direct in and/or registered in
- Output
  - 3-state driver should be enabled by TS signal
  - Direct output or registered output



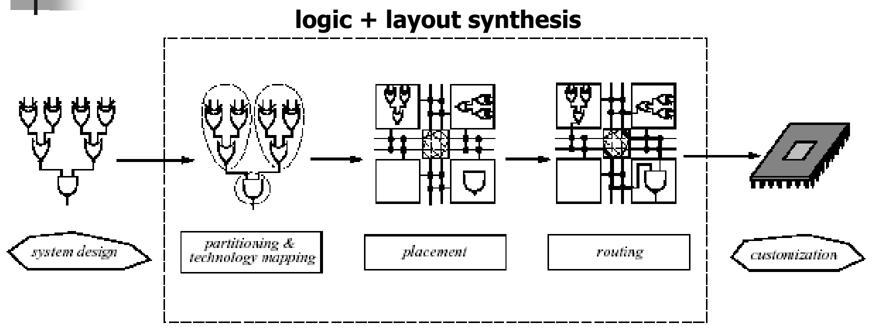
7-47

Design with FPGA

- Using HDL, schematic editor, SM chart or FSM diagram to capture the design
- Simulate and debug the design
- Work out detail logic and feed the logic into CLBs and IOBs
  - Completed by a CAD tool
- Generate bit pattern for programming the FPGA and download into the internal configurable memory cells
- Test the operations

7-48

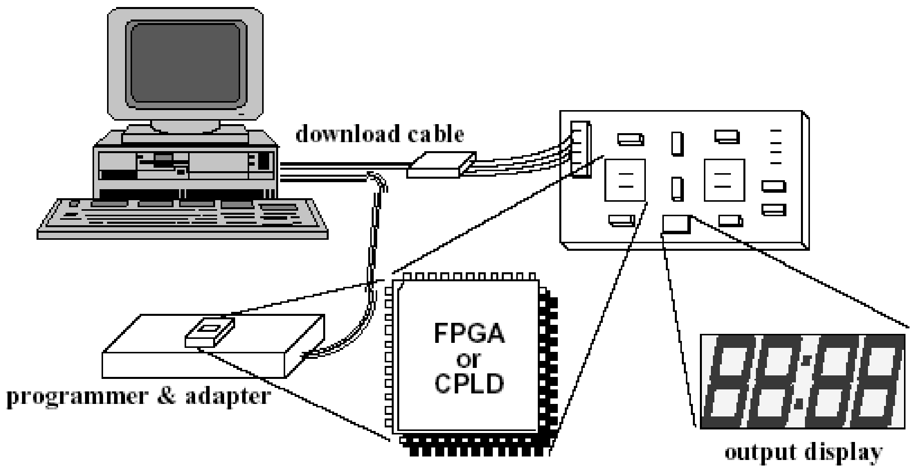
FPGA Design Flow



- Advantages: Fast and reusable prototyping
  - Can be reprogrammed and reused
  - Implementation time is very short
- Disadvantages: Expensive and high volume

7-49

Download to a FPGA Demo Board



Source: CIC training manual

7-50