Memory and Programmable Logic

Chapter 7

Steve Oldridge

Dr. Sidney Fels

Topics

- Random Access Memory (RAM)
- Memory Decoding
- Error Detection & Correction
- Read-Only Memory (ROM)
- Programmable Logic Array
- Programmable Array Logic
- Sequential Programmable Devices

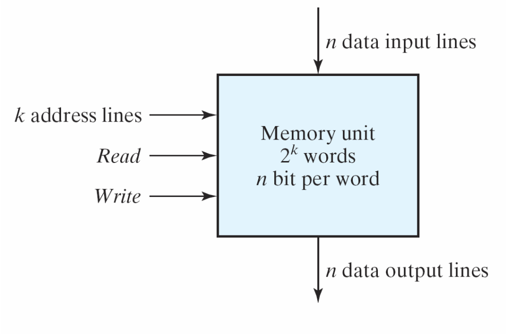
Memory

- RAM
  - Write and Read from the system
- ROM
  - Read from the system
  - Written during configuration
- Programmable Logic Device
  - ROM
  - PLA
  - FPGA

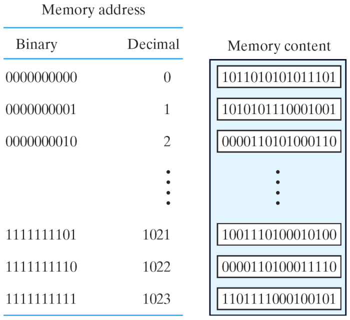
(a) Conventional symbol

(b) Array logic symbol

Random Access Memory



Memory Content



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

Read & Write Ops

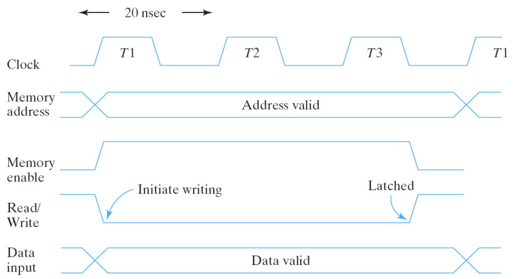
- Read
  - Set the binary address lines
  - Activate the Read signal
- Write
  - Set the binary address lines
  - Set the input
  - Activate Write

Memory Control

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

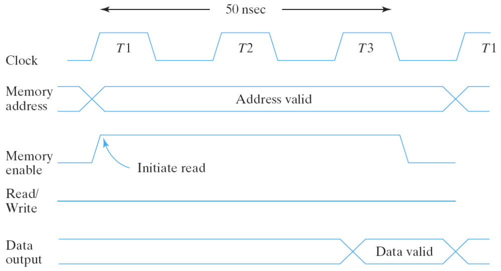
HDL Memory

Timing Waveforms



(a) Write cycle

Timing Waveforms (Read)

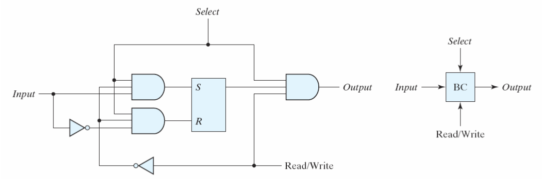


(b) Read cycle

Types of Memories

- Sequential Access Memory
- Static RAM
  - Latches store the values
- Dynamic RAM
  - Electric charge stored on capacitors
  - Capacitors must be periodically refreshed
- ROM
  - Nonvolatile memory

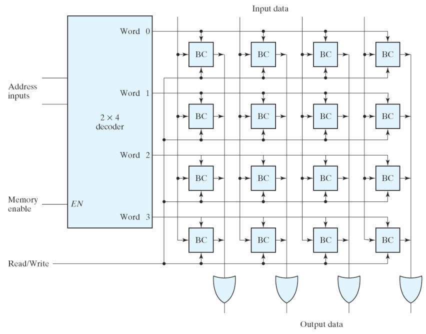
Binary Memory Cells



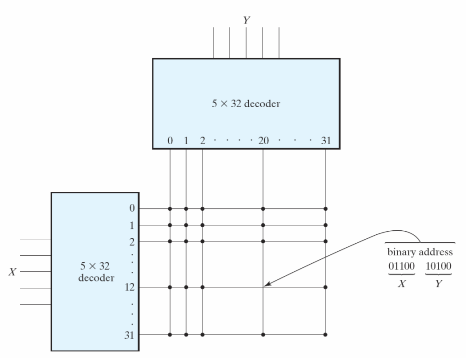
(a) Logic diagram

(b) Block diagram

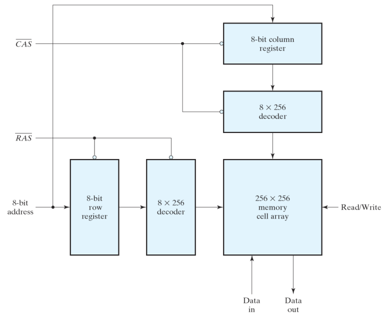
4x4 RAM



Coincident Decoder



Address Multiplexing



Error Detection and Correction

- Hamming Code
  - K ‘parity’ bits are added to an N bit word
  - Bits have a particular calculable value
- Positioned in Powers of 2
  - 1, 2, 4, 8

Hamming Code Example

- 11000100
- P_1 P_2 1 P_4 100 $P_8$0100

- P_1 = XOR (3,5,7,9,11)
- P_2 = XOR (3,6,7,10,11)
- P_3 = XOR (5,6,7,12)
- P_4 = XOR (9,10,11,12)

Hamming Code Check

- 001110010100

- C_1 = XOR (1,3,5,7,9,11)
- C_2 = XOR (2,3,6,7,10,11)
- C_3 = XOR (4,5,6,7,12)
- C_4 = XOR (8,9,10,11,12)

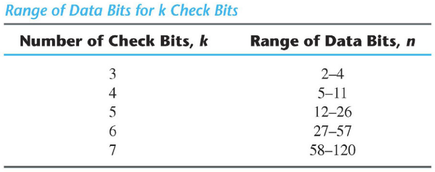
Hamming Code continued

- How do we determine Parity and Check equations for other size numbers?
- Determine the numbers from [0, 2^k-1]
  - Least significant bit is 1 for (1,3,5,7 etc.)
  - Bit 1 is a 1 for (2,3,6,7 etc.)

Single Error Correction, Double Error Detection

- 001110010100P_{13}
- P_{13}= XOR of all 12 bits
  - Tells us if there was only one error or not
- If C=0, P=0 No Error
- If C!=0, P=1 Only 1 error, Correct it
- If C!=0, P=0 More than one Error
- If C=0, P=1 Error occurred in P_{13}

Size of Parity

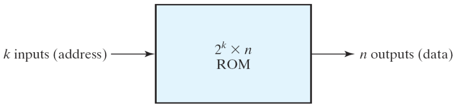


Range of Data Bits for k Check Bits

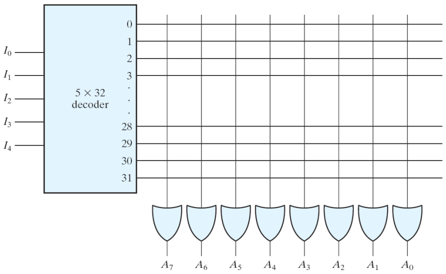
Number of Check Bits, *k*	**Range of Data Bits, *n***
3	2–4
4	5–11
5	12–26
6	27–57
7	58–120

Read Only Memory

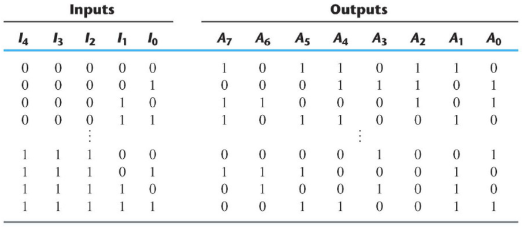
- Permanent Storage
- Allows for configuration of devices to be stored on device without requiring load



32x8 ROM

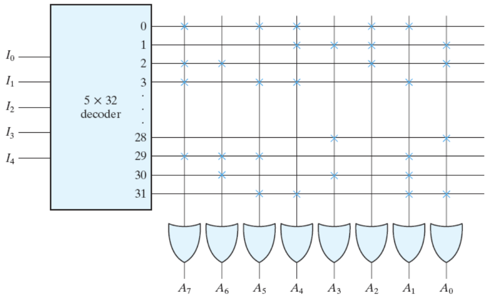


ROM Truth Table

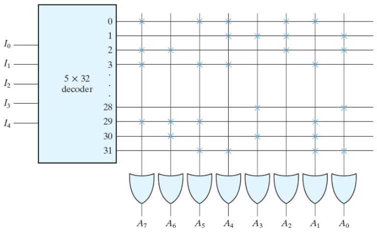


Inputs					Outputs							
I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		⋮						⋮				
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

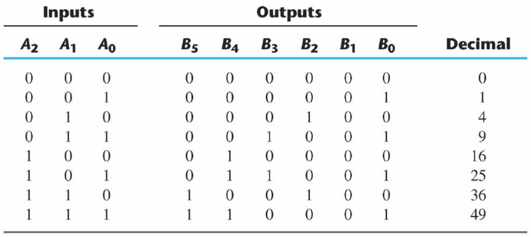
Configured ROM



ROM as a Combinational Circuit

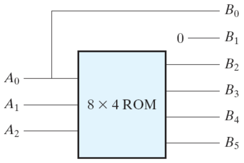


ROM ‘Circuit’ Example



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

ROM 'Circuit' Example



(a) Block diagram

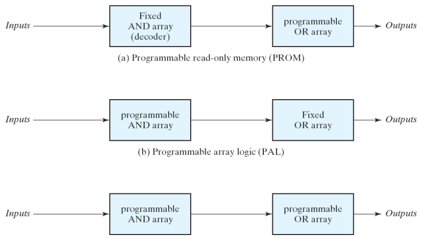
A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

(b) ROM truth table

4 Types of ROMs

- Mask Programming
  - Done during the fab process
- Programmable ROM (PROM)
  - All fuses are intact (set to 1) and are 'Blown'
- Erasable PROM (EPROM)
  - Ultraviolet light used to reprogram
- Electronically Erasable PROM (EEPROM)
  - Programmed connections can be erased via signal

Combinational PLDs

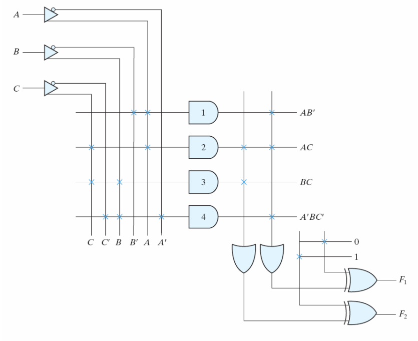


(a) Programmable read-only memory (PROM)

(b) Programmable array logic (PAL)

(c) Programmable logic array (PLA)

Programmable Logic Arrays



Programming PLAs

					Outputs	
		Inputs			(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	—

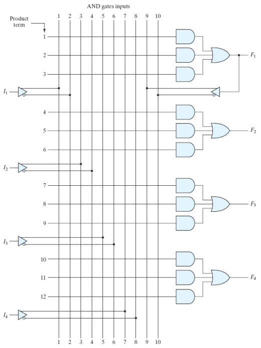
PLA Example

- F_1(ABC) = Sum(0,1,2,4)
- F_2(ABC) = Sum(0,5,6,7)

PLA programming table

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

Programmable Array Logic

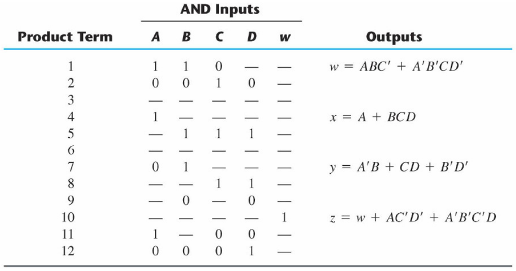


Example

- X = A + BCD
- Y = A’B + CD + B’D’
- Z = ABC’ + A’B’CD’ + AC’D’ + A’B’C’D

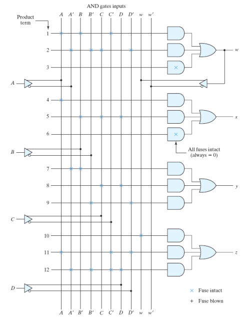
- W = ABC’ + A’B’CD’
- Z = W + AC’D’ + A’B’C’D

PAL Programming Table



	AND Inputs					
Product Term	*A*	*B*	*C*	*D*	*w*	Outputs
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	—	

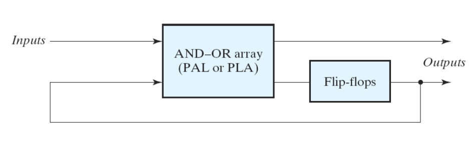
Configured PAL



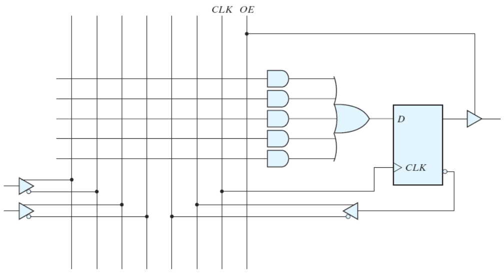
Sequential Programmable Devices

- Sequential Programmable Logic Device (SPLD)
- Complex Programmable Logic Device (CPLD)
- Field Programmable Gate Array (FPGA)

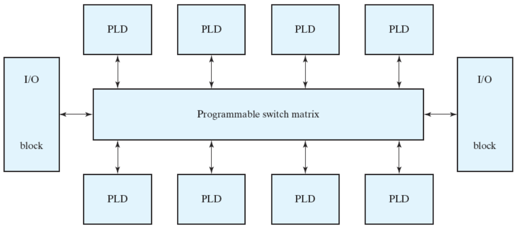
SPLDs



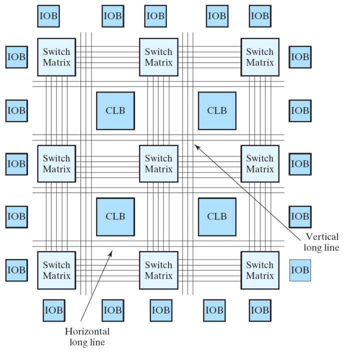
SPLD Macrocell



CPLD



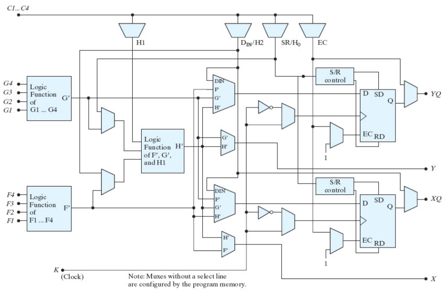
FPGA



FPGA CAD

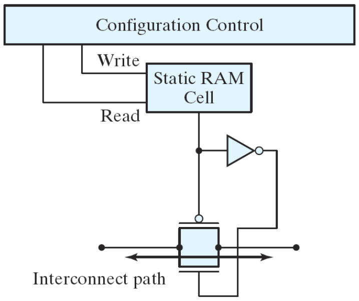
- Synthesis
  - How can we best fit our design onto the device?
- Placement
  - Which blocks should go next to each other?
- Routing
  - How can we connect the wires that need connecting?

Xilinx 4000X Logic Block

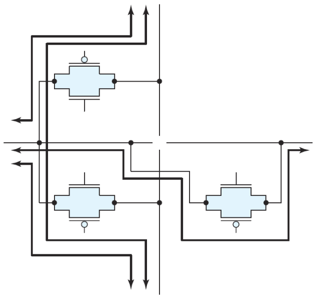


Note: Muxes without a select line are configured by the program memory.

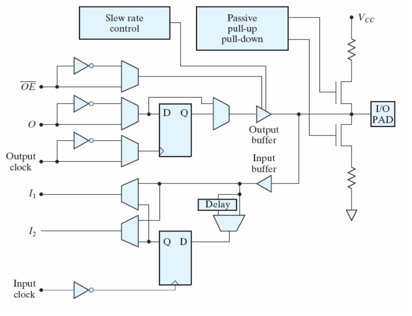
Switch Block Interconnect



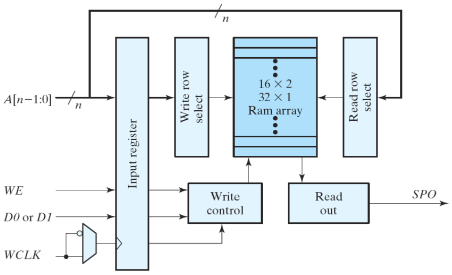
PIPs at the Switch



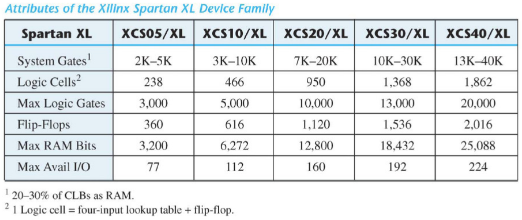
I/O Blocks



LUT as Memory



Spartan XL Device Family

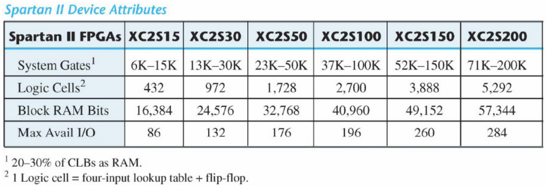


Attributes of the Xilinx Spartan XL Device Family

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.

Spartan II Family

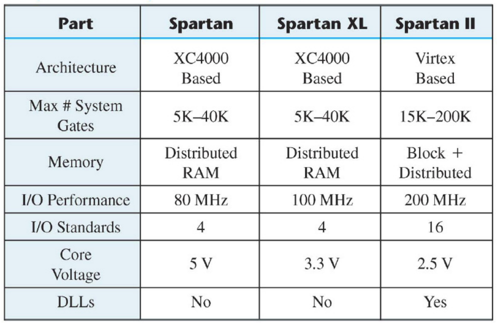


Spartan II Device Attributes

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

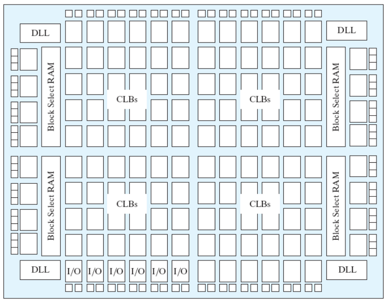
[1] 20–30% of CLBs as RAM.
[2] 1 Logic cell = four-input lookup table + flip-flop.

Family Comparisons

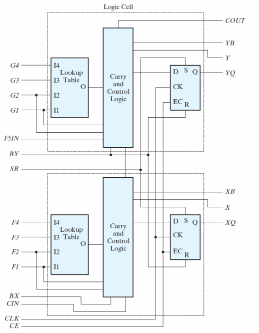


Part	Spartan	Spartan XL	Spartan II
Architecture	XC4000 Based	XC4000 Based	Virtex Based
Max # System Gates	5K–40K	5K–40K	15K–200K
Memory	Distributed RAM	Distributed RAM	Block + Distributed
I/O Performance	80 MHz	100 MHz	200 MHz
I/O Standards	4	4	16
Core Voltage	5 V	3.3 V	2.5 V
DLLs	No	No	Yes

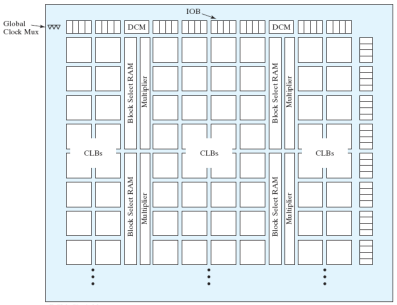
Spartan II Architecture



Spartan II CLBs



Virtex Family



DCM: Clock Manager