Combinational Circuits

Chapter 3

S. Dandamudi

Outline

- Introduction
- Multiplexers and demultiplexers
  - Implementing logical functions
  - Efficient implementation
- Decoders and encoders
  - Decoder-OR implementations
- Comparators
- Adders
  - Half-adders
  - Full-adders
- Programmable logic devices
  - Programmable logic arrays (PLAs)
  - Programmable array logic (PALs)
- Arithmetic and logic units (ALUs)

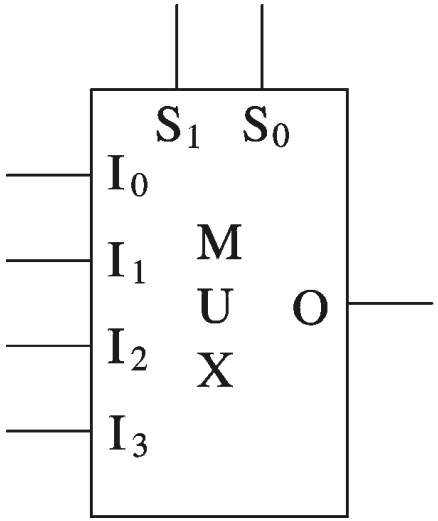
Introduction

- Combinational circuits
  - » Output depends only on the current inputs
- Combinational circuits provide a higher level of abstraction
  - * Helps in reducing design complexity
  - * Reduces chip count
  - * Example: 8-input NAND gate
    - » Requires 1 chip if we use 7430
    - » Several 7400 chips (How many?)
- We look at some useful combinational circuits

Multiplexers

- Multiplexer
  - 2^n data inputs
  - n selection inputs
  - a single output
- Selection input determines the input that should be connected to the output

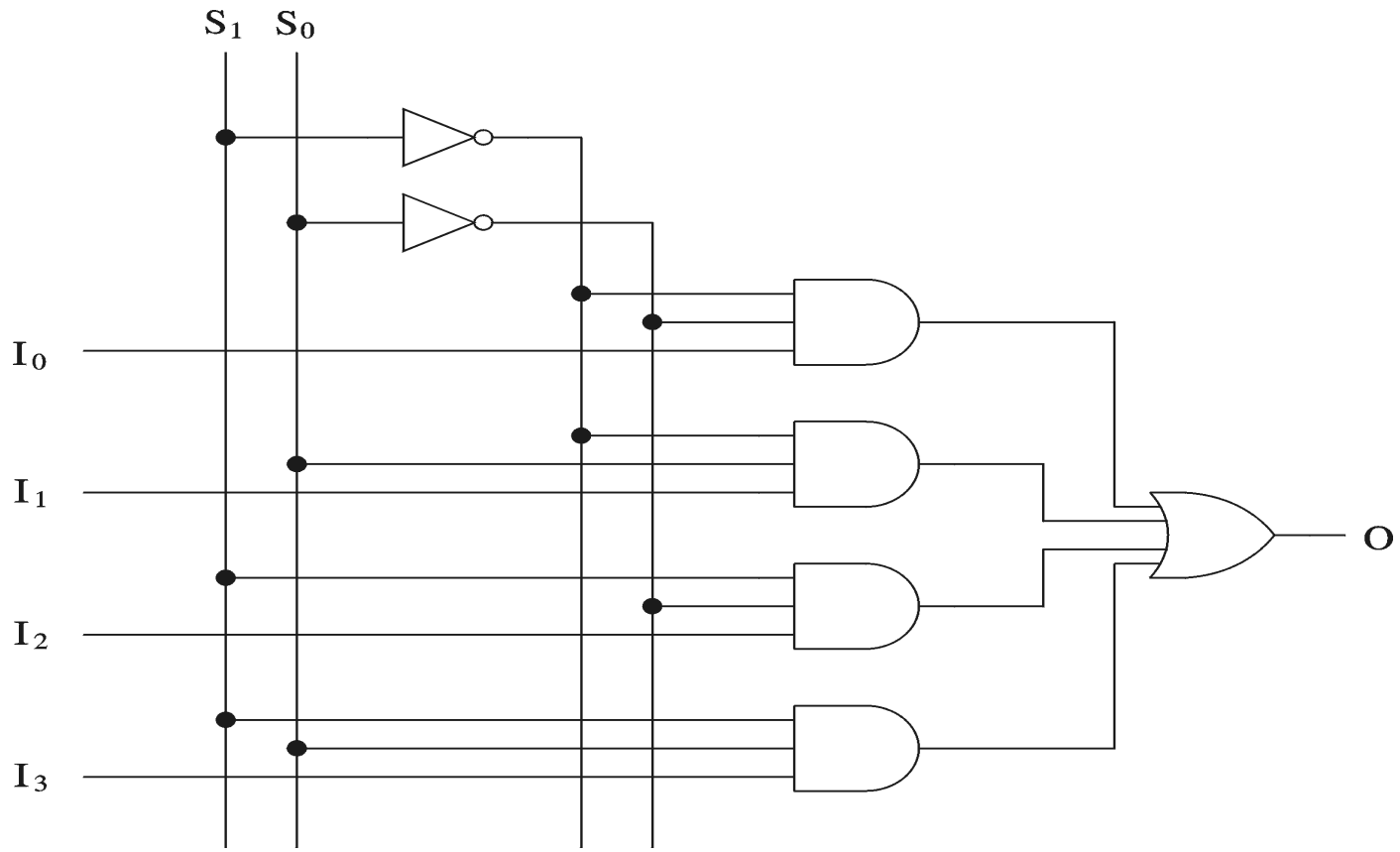
4-data input MUX



S_1	S_0	O
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

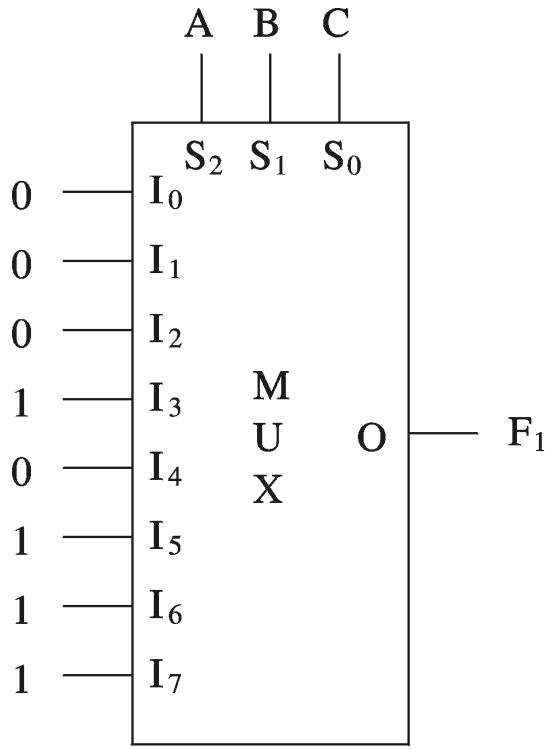
Multiplexers (cont'd)

4-data input MUX implementation

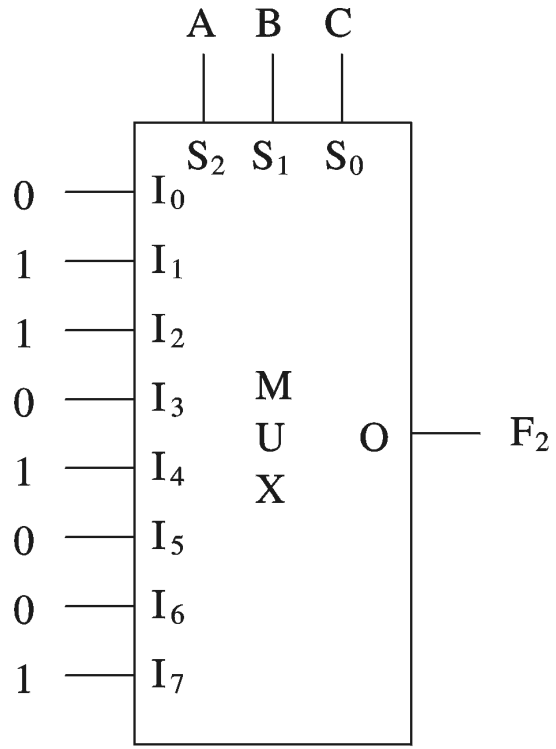


Multiplexers (cont'd)

MUX implementations



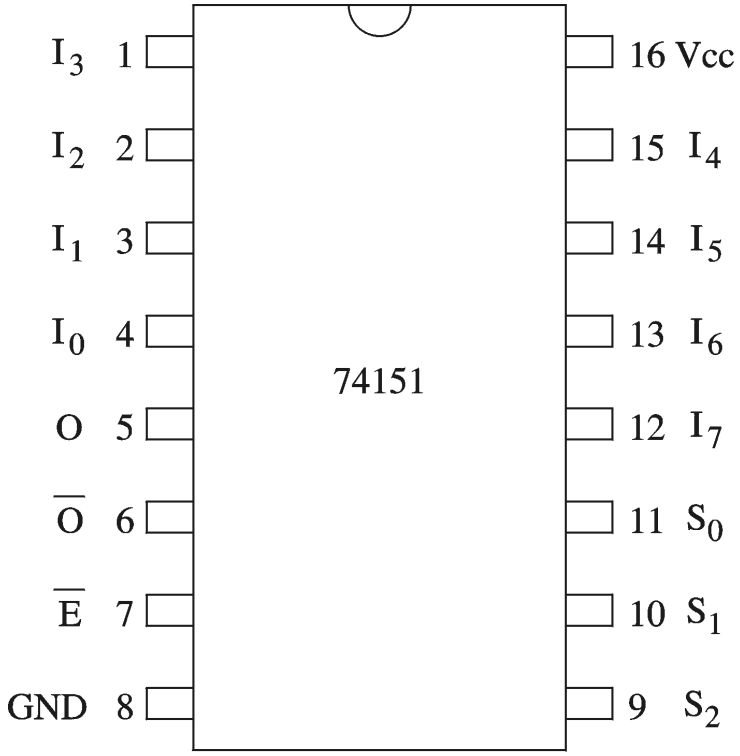
Majority function



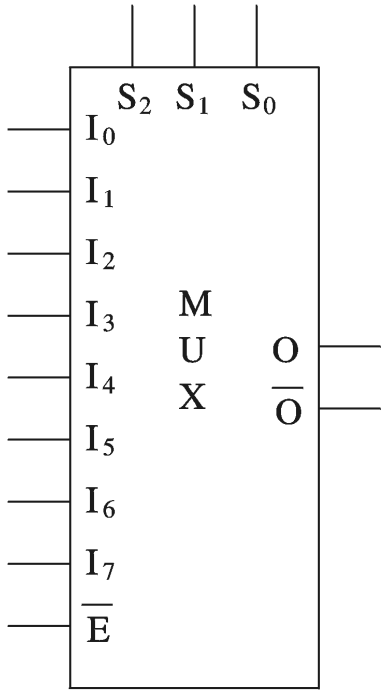
Even-parity function

Multiplexers (cont'd)

Example chip: 8-to-1 MUX



(a) Connection diagram



(b) Logic symbol

Multiplexers (cont'd)

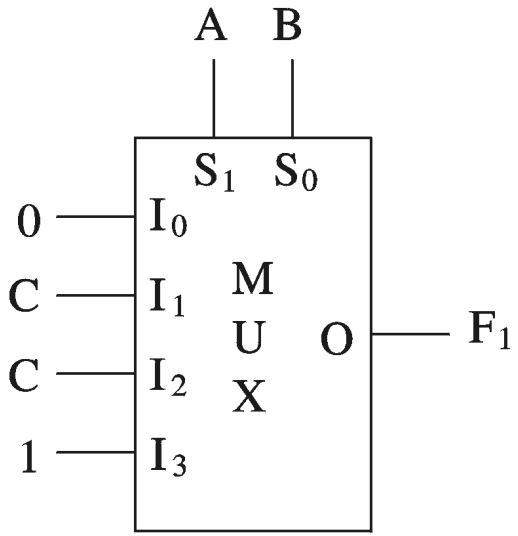
Efficient implementation: Majority function

Original truth table

A	B	C	F_1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

New truth table

A	B	F_1
0	0	0
0	1	C
1	0	C
1	1	1



Multiplexers (cont'd)

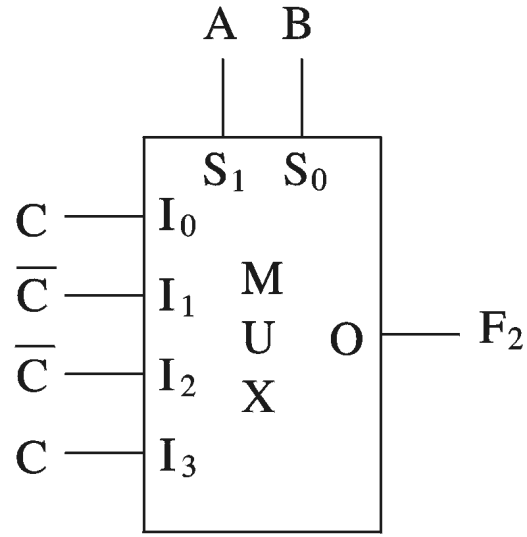
Efficient implementation: Even-parity function

Original truth table

A	B	C	F_1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

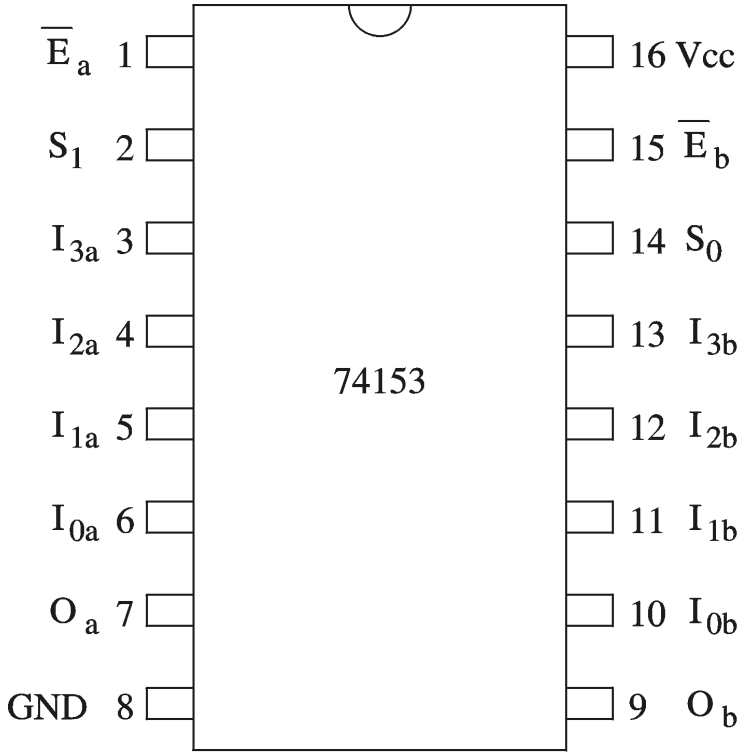
New truth table

A	B	F_1
0	0	C
0	1	$\overline{C}$
1	0	$\overline{C}$
1	1	C

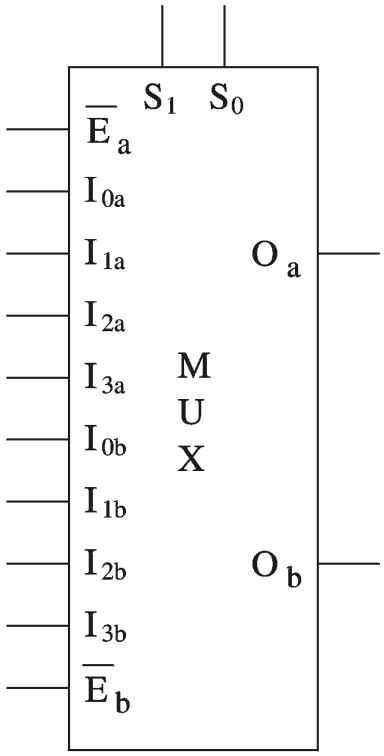


Multiplexers (cont'd)

74153 can used to implement two output functions



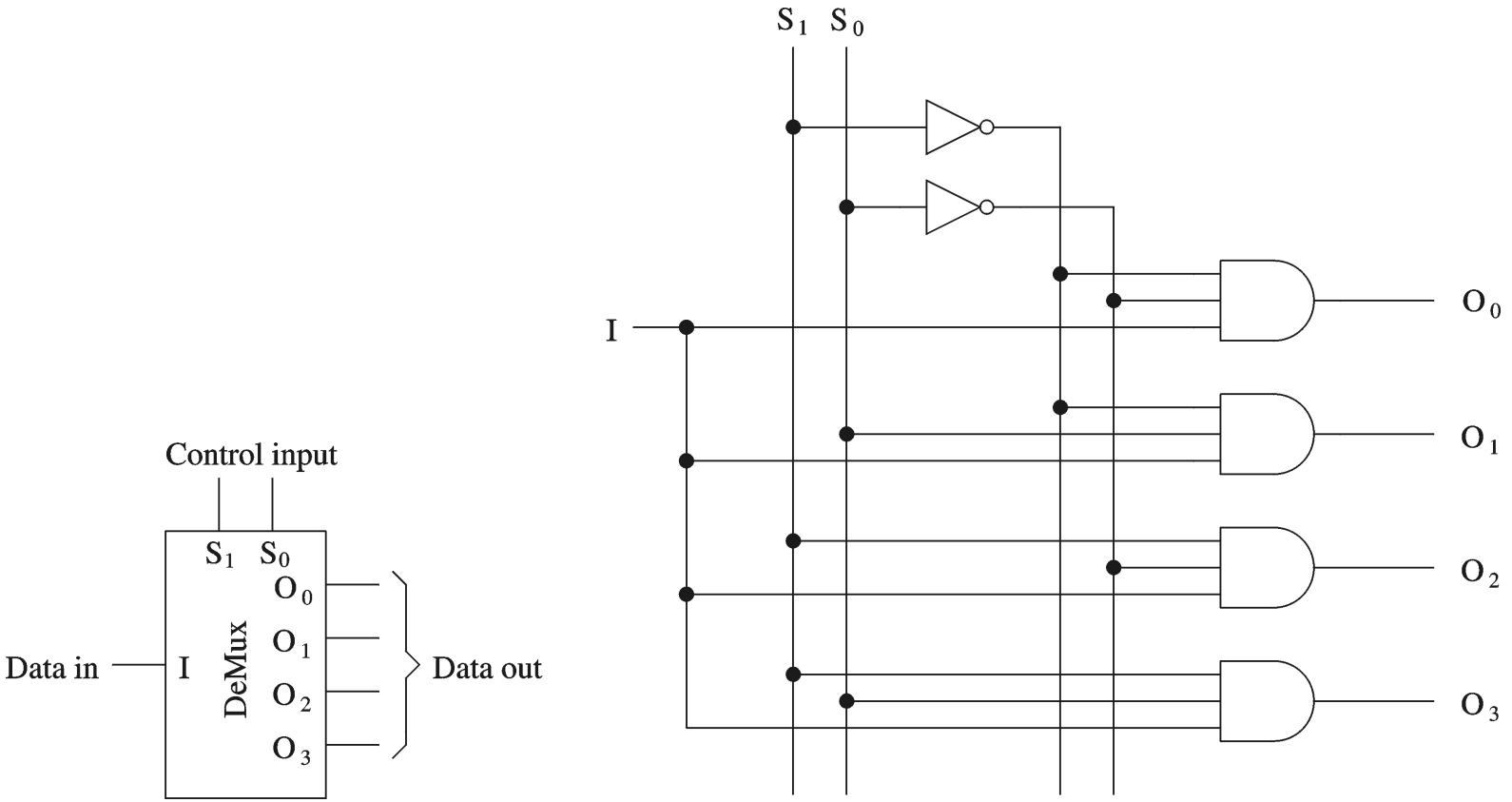
(a) Connection diagram



(b) Logic symbol

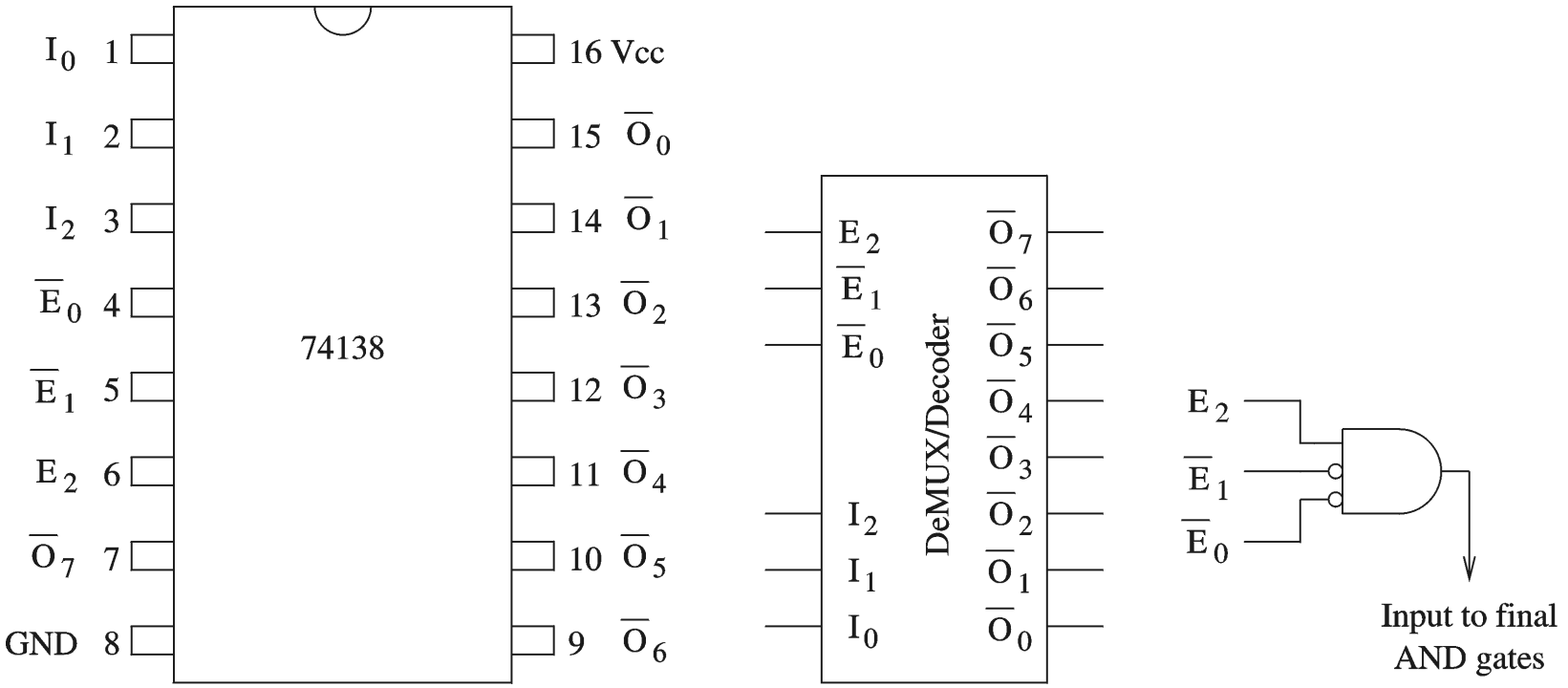
Demultiplexers

Demultiplexer (DeMUX)



Demultiplexers (cont'd)

74138 can used as DeMUX and decoder

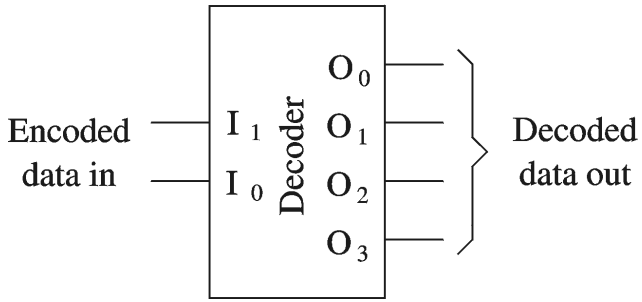


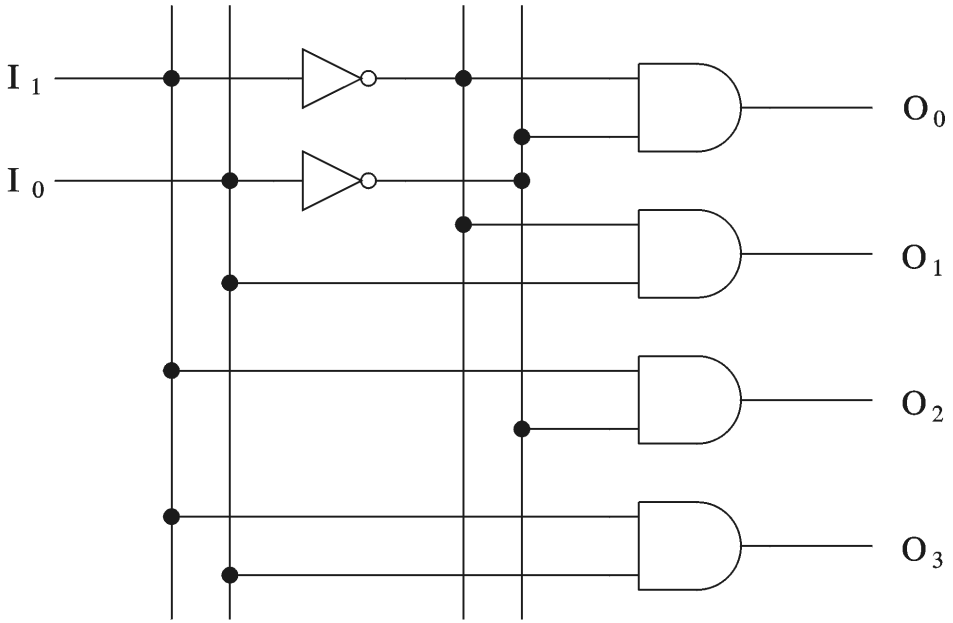
(a) Connection diagram (b) Logic symbol (c) Enable input logic details

Decoders

- Decoder selects one-out-of-N inputs

I_1	I_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

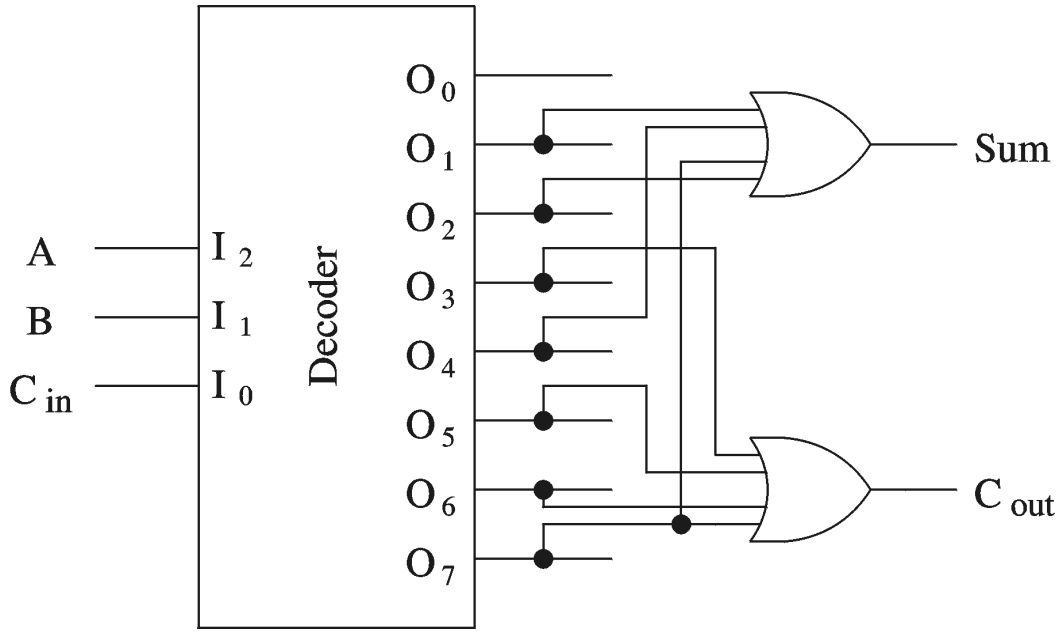




Decoders (cont'd)

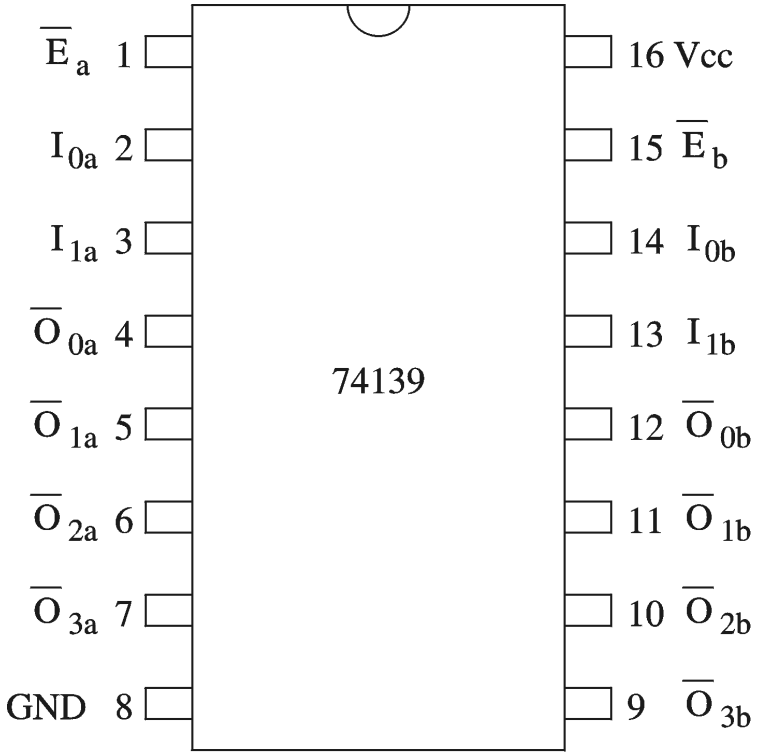
Logic function implementation

A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

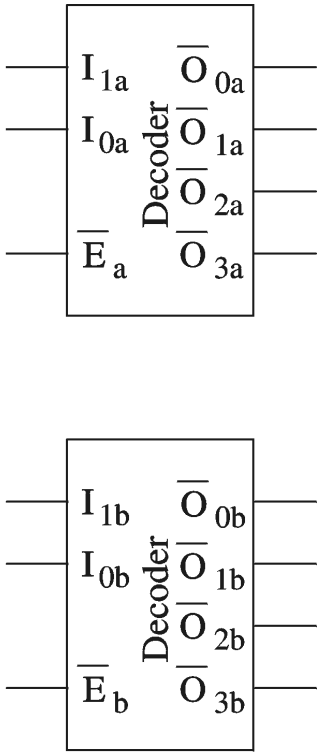


Decoders (cont'd)

74139: Dual decoder chip



(a) Connection diagram

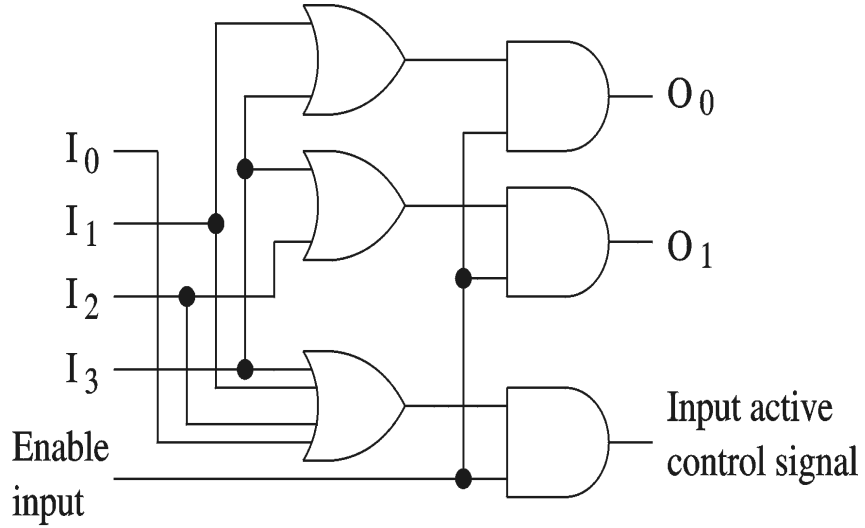


(b) Logic symbol

Encoders

- Encoders
  - Take 2^B input lines and generate a B-bit binary number on B output lines
  - Cannot handle more than one input with 1

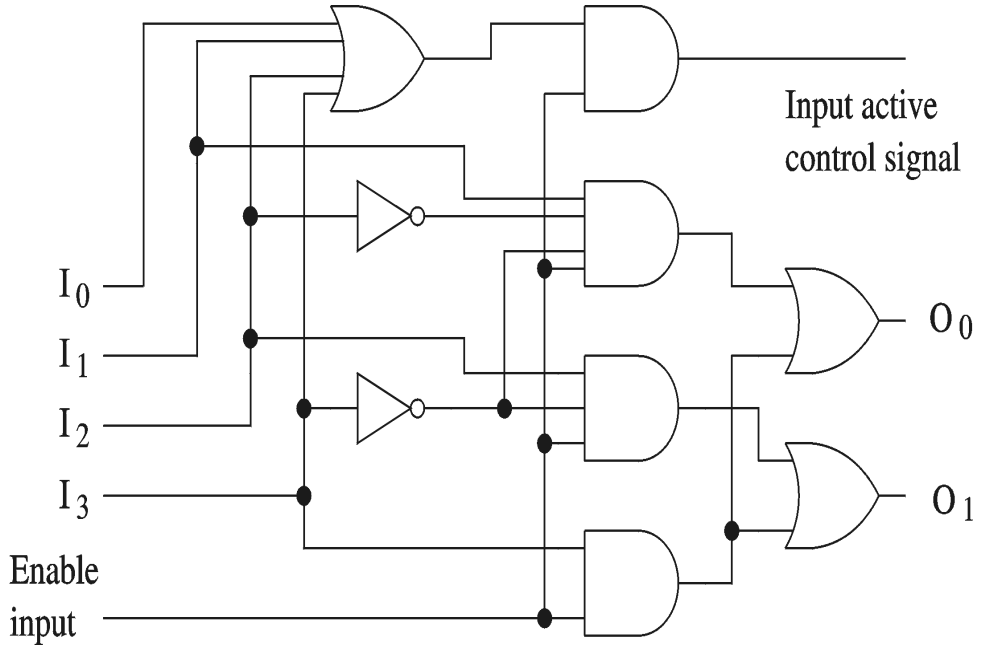
Enable input	I_3	I_2	I_1	I_0	O_1	O_0	Input active control signal
0	X	X	X	X	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	1	1
1	0	1	0	0	1	0	1
1	1	0	0	0	1	1	1



Encoders (cont'd)

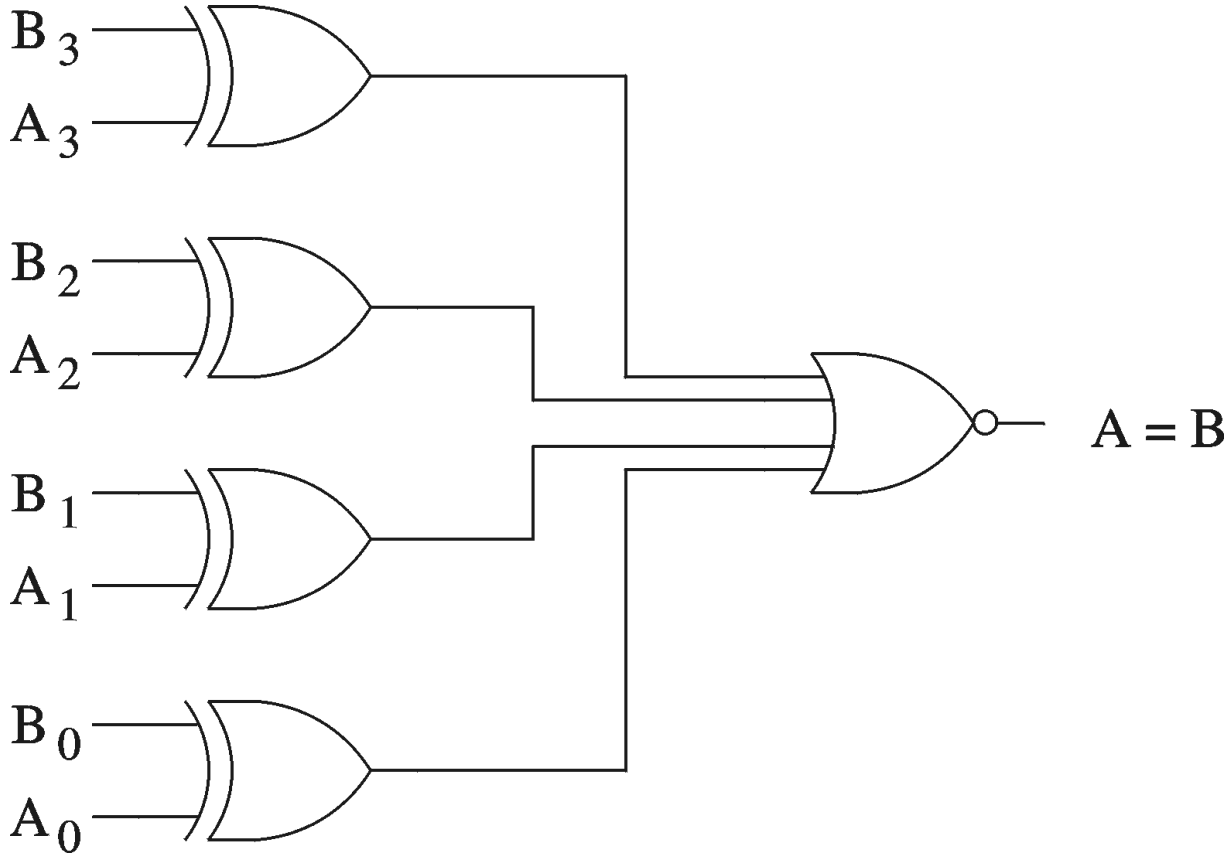
- Priority encoders
  - Handles inputs with more than one 1

Enable input	I_3	I_2	I_1	I_0	O_1	O_0	Input active control signal
0	X	X	X	X	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	X	0	1	1
1	0	1	X	X	1	0	1
1	1	X	X	X	1	1	1



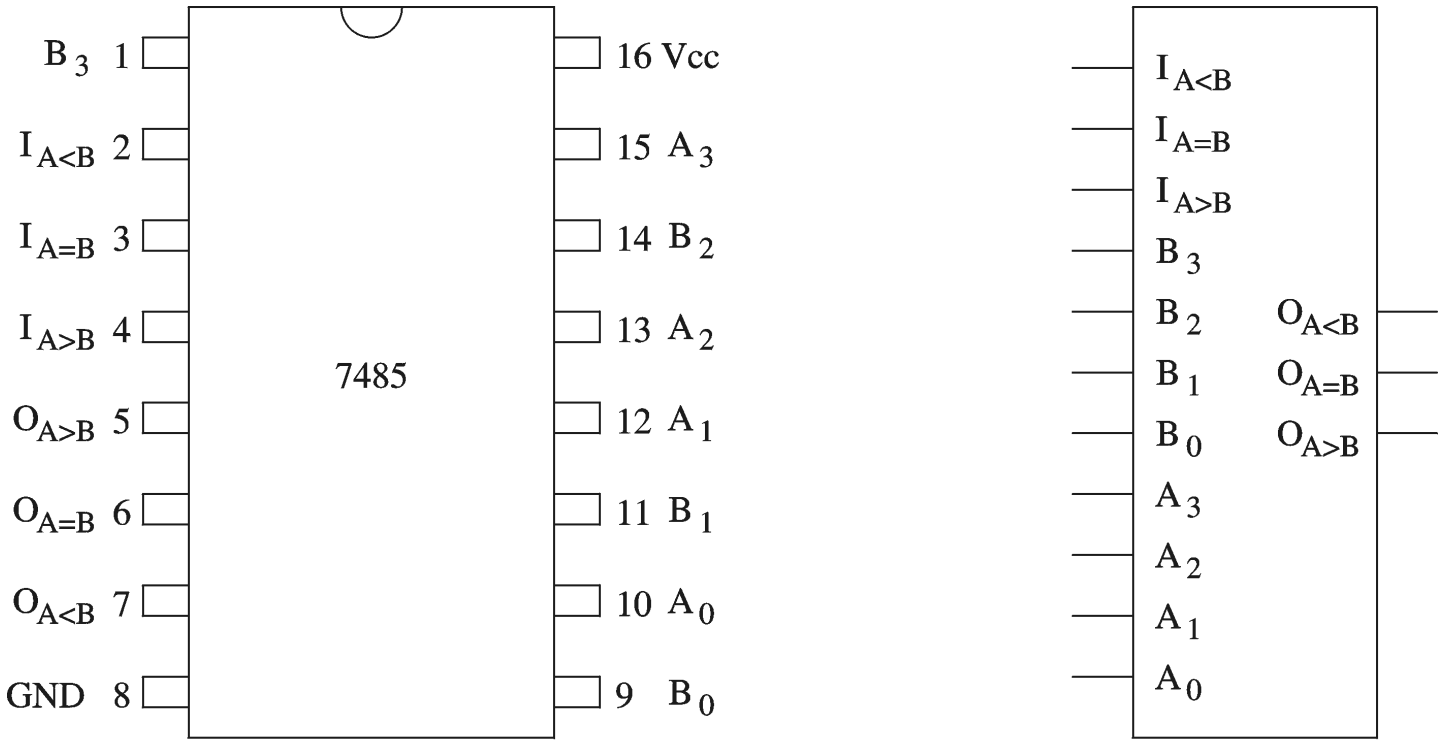
Comparator

- Used to implement comparison operators (= , > , < , ≥ , ≤)



Comparator (cont'd)

4-bit magnitude comparator chip

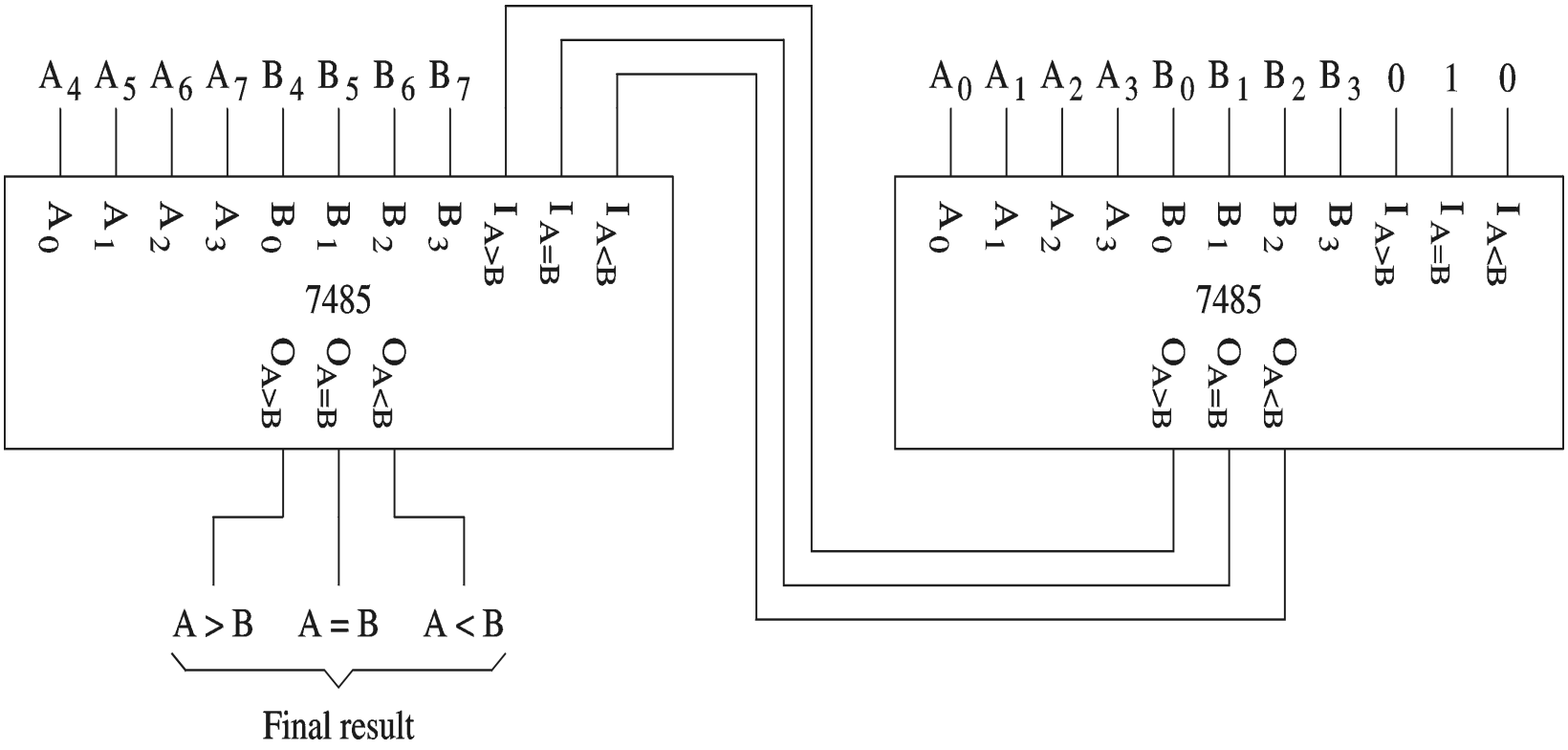


(a) Connection diagram

(b) Logic symbol

Comparator (cont'd)

Serial construction of an 8-bit comparator

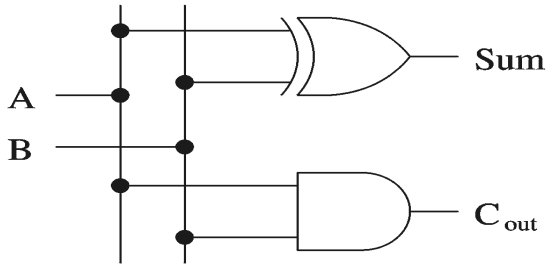


Adders

- Half-adder
  - Adds two bits
    - Produces a *sum* and *carry*
  - Problem: Cannot use it to build larger inputs
- Full-adder
  - Adds three 1-bit values
    - Like half-adder, produces a *sum* and *carry*
  - Allows building N-bit adders
    - Simple technique
      - Connect C_{out} of one adder to C_{in} of the next
    - These are called *ripple-carry adders*

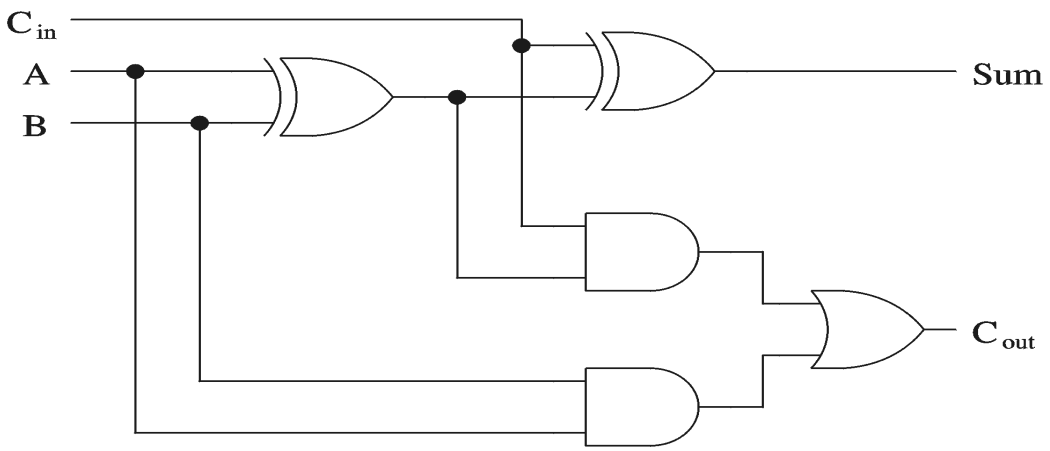
Adders (cont'd)

A	B	Sum	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



(a) Half-adder truth table and implementation

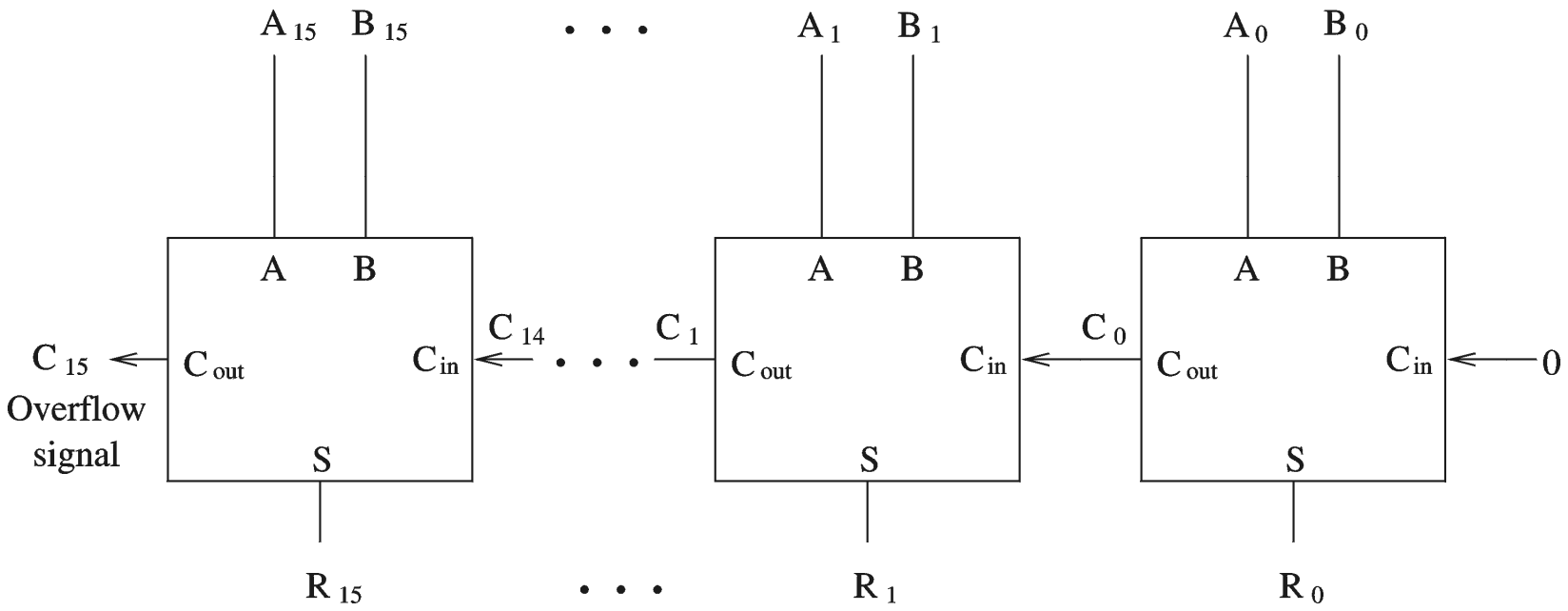
A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



(b) Full-adder truth table and implementation

Adders (cont'd)

A 16-bit ripple-carry adder

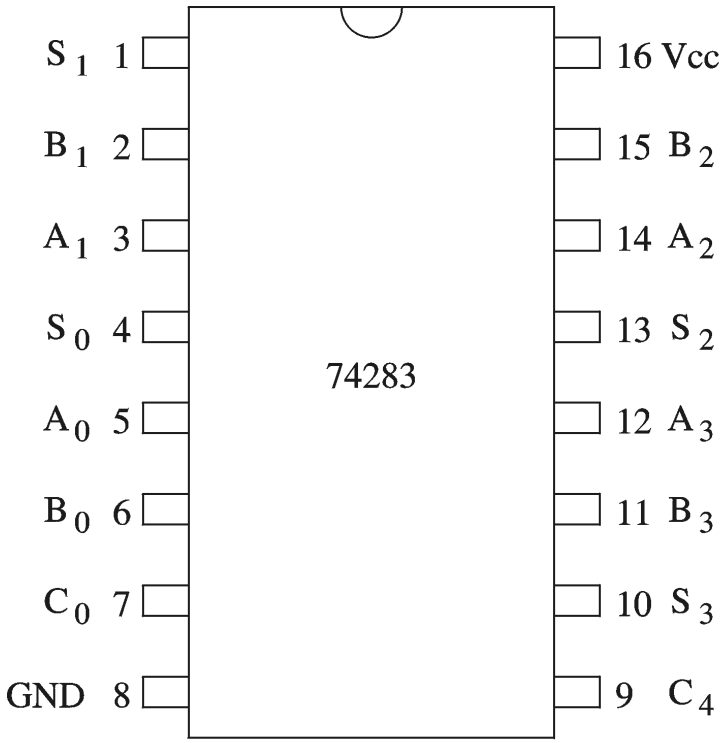


Adders (cont'd)

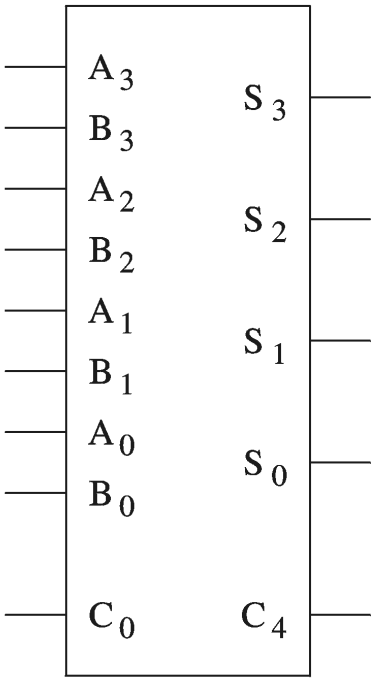
- Ripple-carry adders can be slow
  - Delay proportional to number of bits
- Carry lookahead adders
  - Eliminate the delay of ripple-carry adders
  - Carry-ins are generated independently
    - $C_0 = A_0 B_0$
    - $C_1 = A_0 B_0 A_1 + A_0 B_0 B_1 + A_1 B_1$
    - . . .
  - Requires complex circuits
  - Usually, a combination carry lookahead and ripple-carry techniques are used

Adders (cont'd)

4-bit carry lookahead adder



(a) Connection diagram



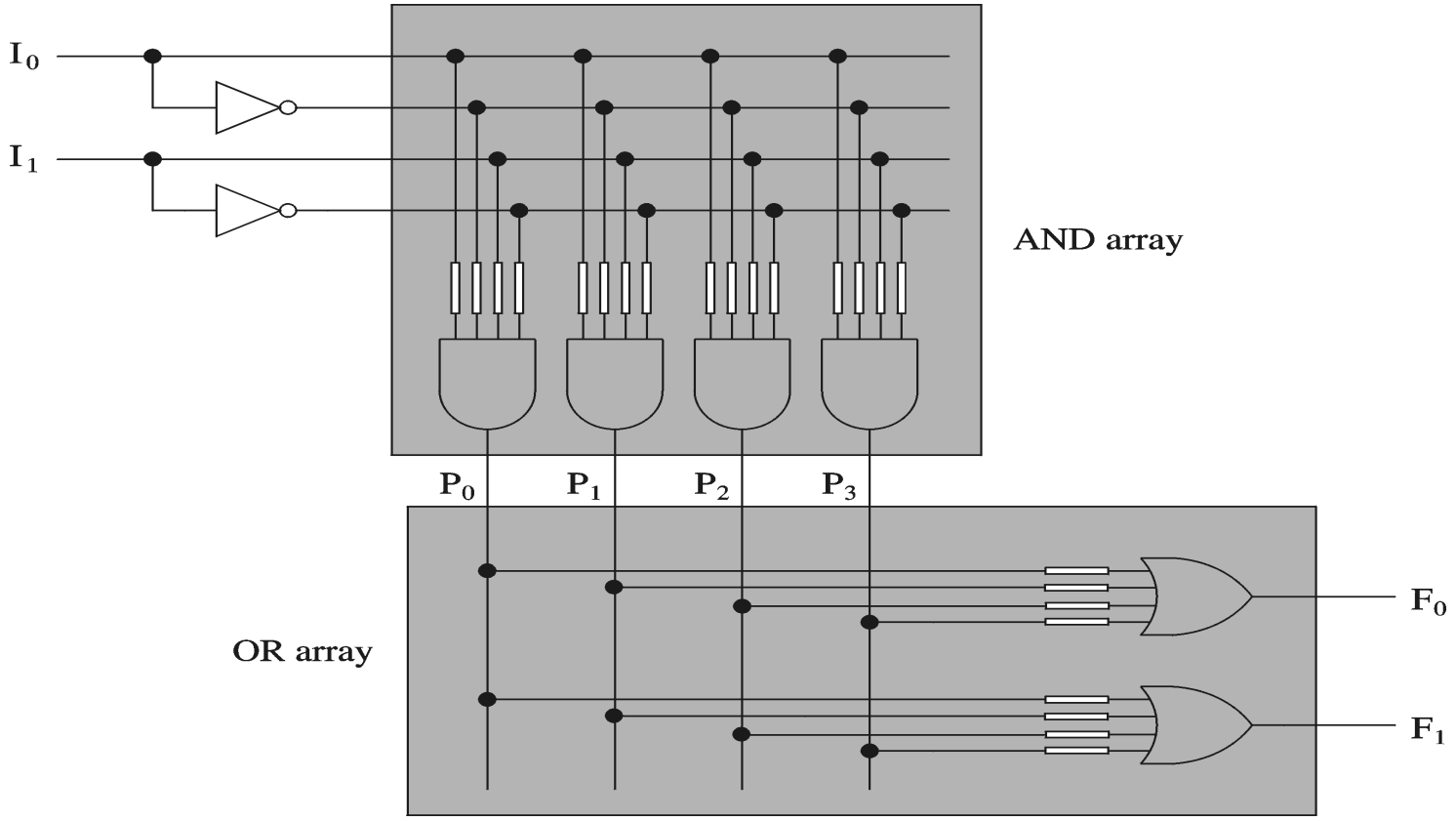
(b) Logic symbol

Programmable Logic Arrays

- PLAs
  - Implement sum-of-product expressions
    - No need to simplify the logical expressions
  - Take *N* inputs and produce *M* outputs
    - Each input represents a logical variable
    - Each output represents a logical function output
  - Internally uses
    - An AND array
      - Each AND gate receives 2*N* inputs
        - → *N* inputs and their complements
    - An OR array

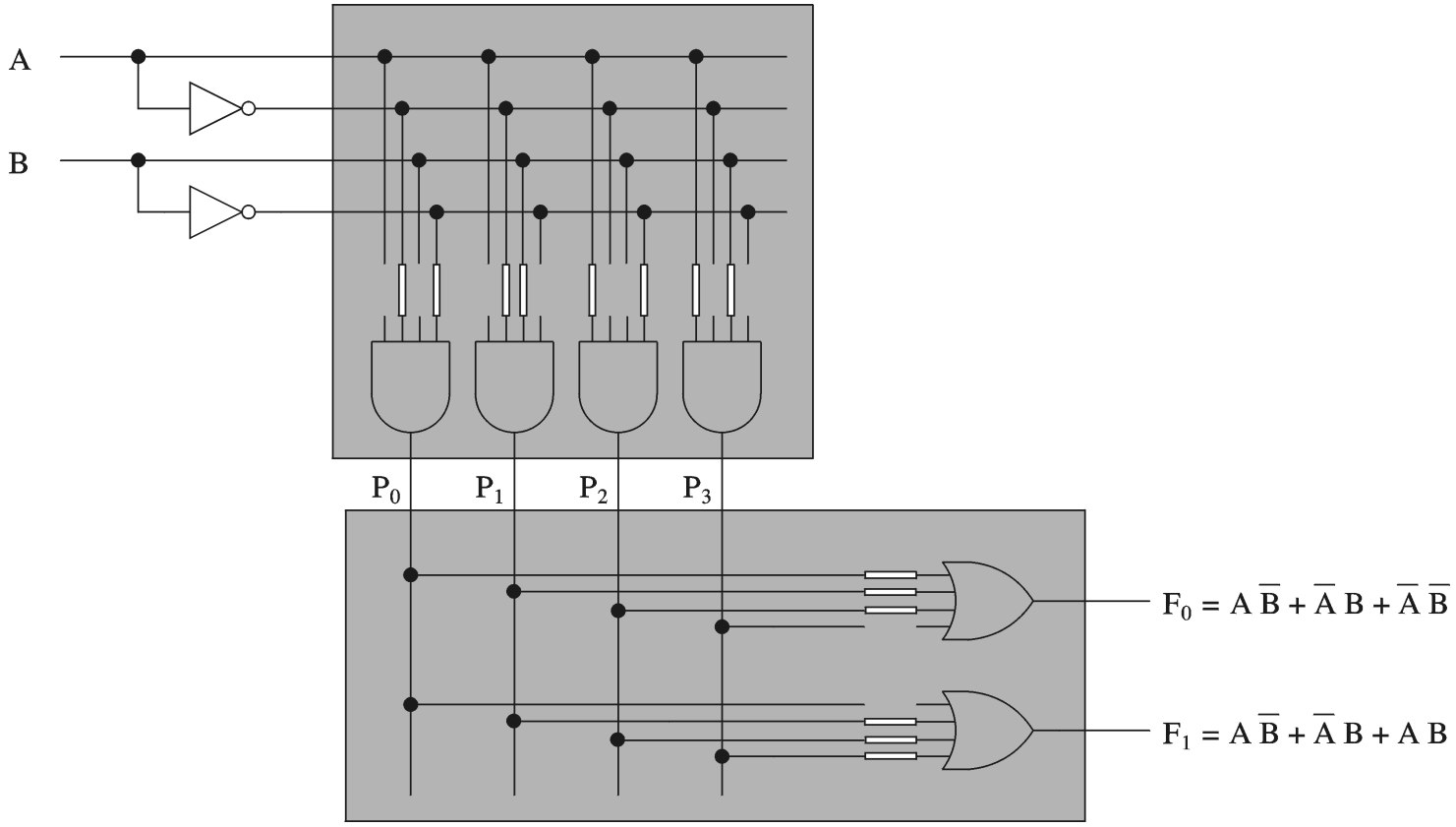
Programmable Logic Arrays (cont'd)

A blank PLA with 2 inputs and 2 outputs



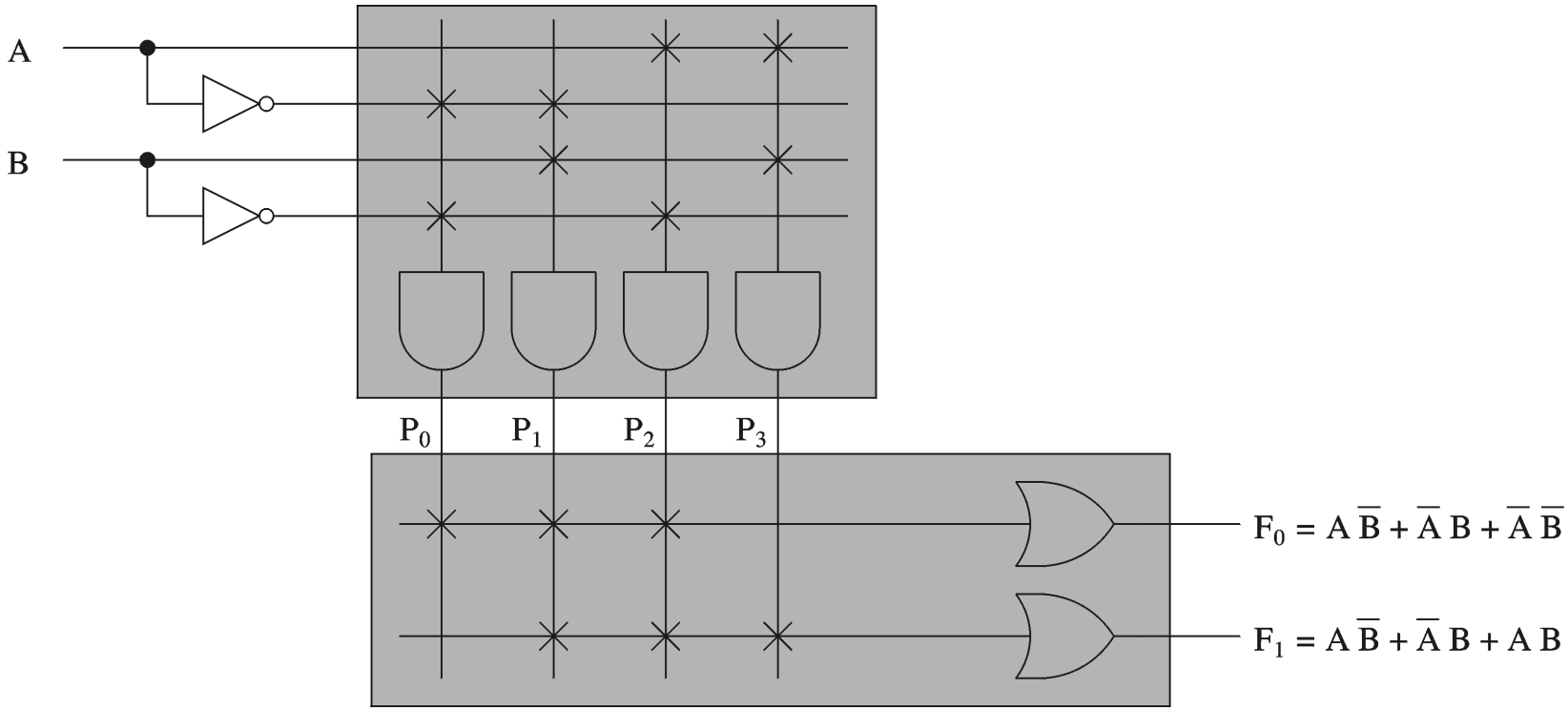
Programmable Logic Arrays (cont'd)

Implementation examples



Programmable Logic Arrays (cont'd)

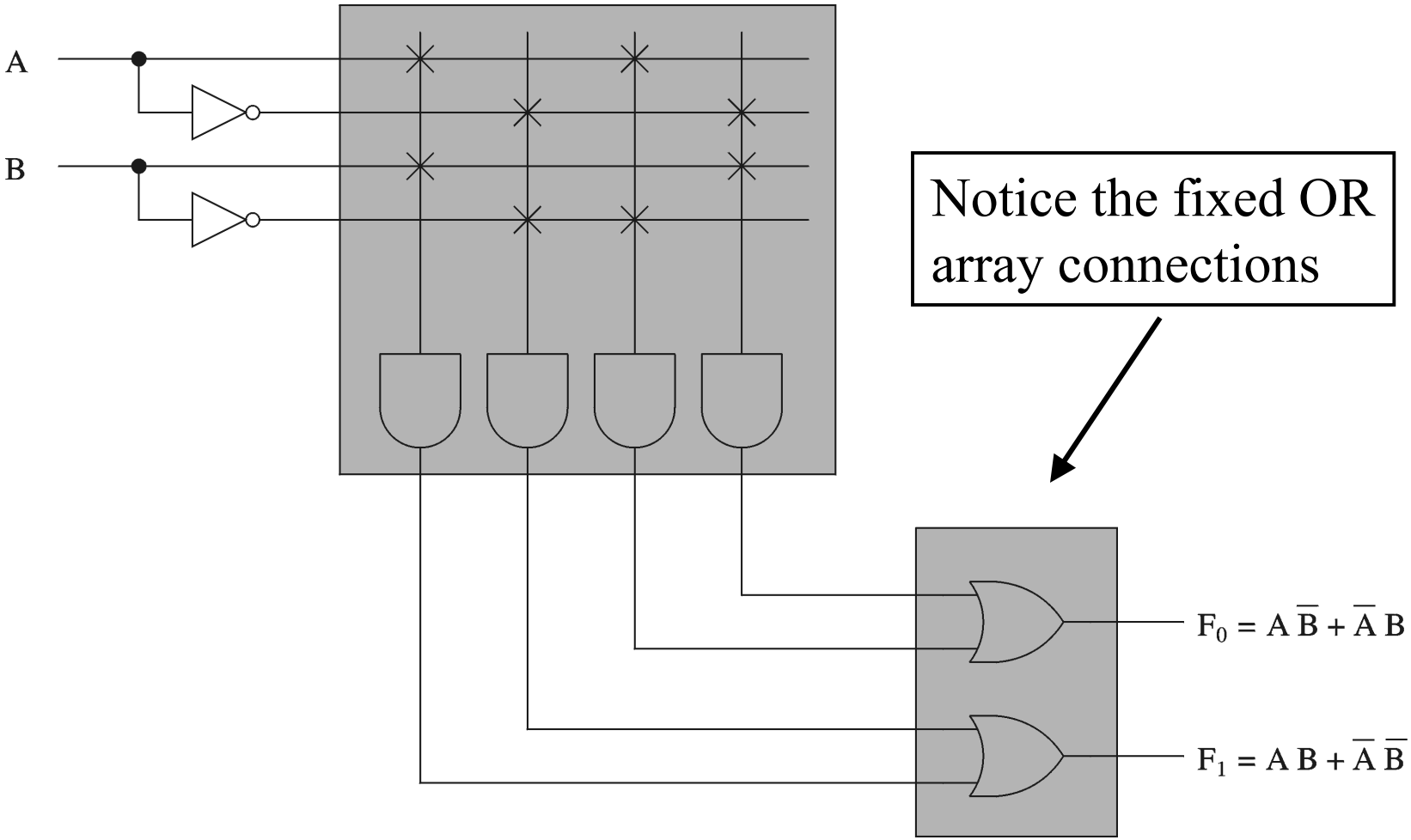
Simplified notation



Programmable Array Logic Devices

- Problem with PLAs
  - * Flexible but expensive
  - * Example
    - » 12 X 12 PLA with
      - – 50-gate AND array
      - – 12-gate OR array
    - » Requires 1800 fuses
      - – 24 X 50 = 1200 fuses for the AND array
      - – 50 X 12 = 600 fuses for the OR array
- PALs reduce this complexity by using fixed OR connections
  - * Reduces flexibility compared PLAs

Programmable Array Logic Devices (cont'd)

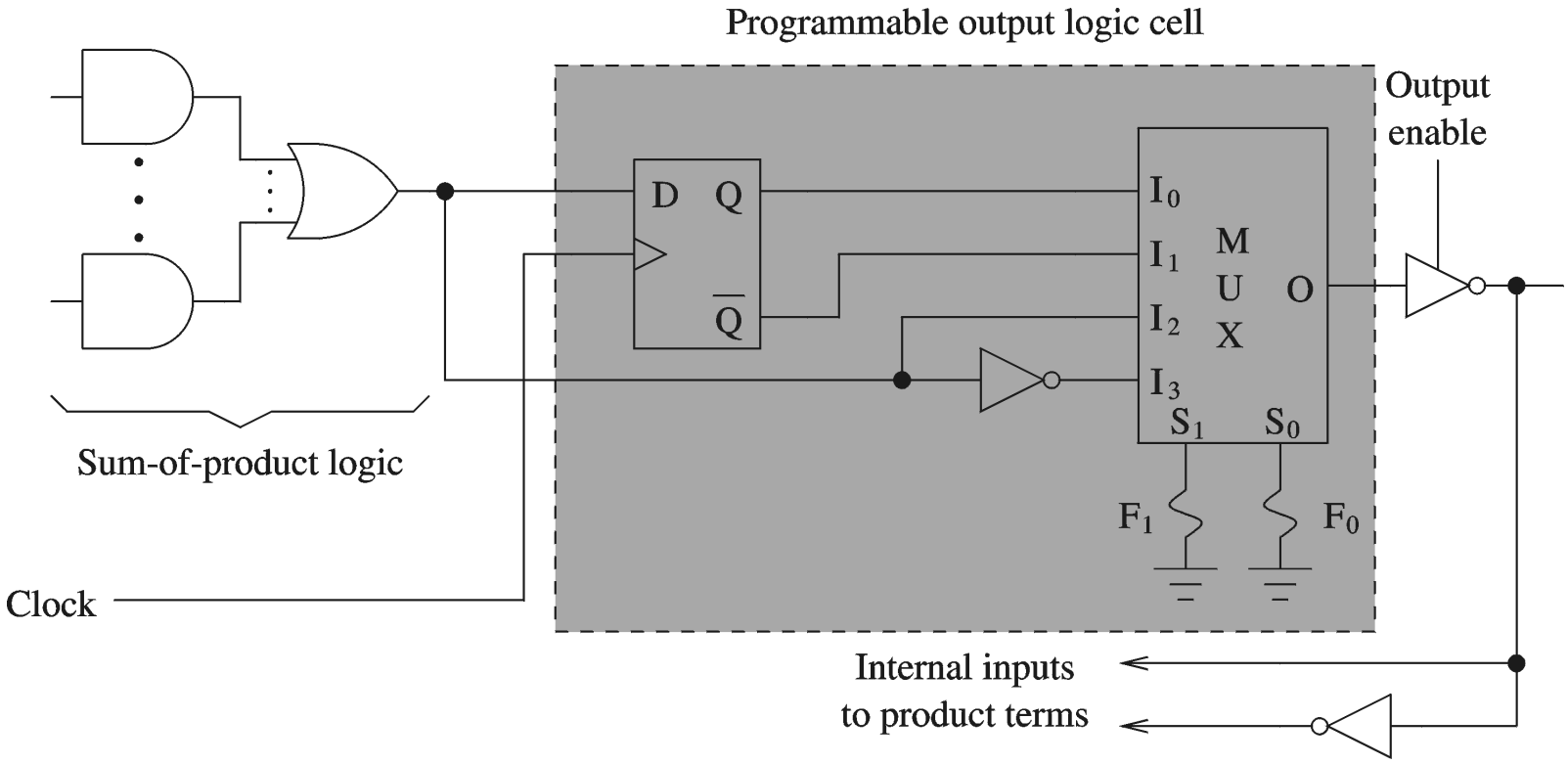


Programmable Array Logic Devices (cont'd)

- An example PAL (Texas Instruments TIBPAL22V10-10C)
  - 22 X 10 PAL (24-pin DIP package)
    - 120-gate AND array
    - 10-gate OR array
  - 44 X 120 = 5280 fuses
    - Just for the AND array
      - OR array does not use any fuses
  - Uses variable number of connections for the OR gates
    - Two each of 8-, 10-, 12-, 14-, and 16-input OR gates
  - Uses internal feedback through a programmable output cell

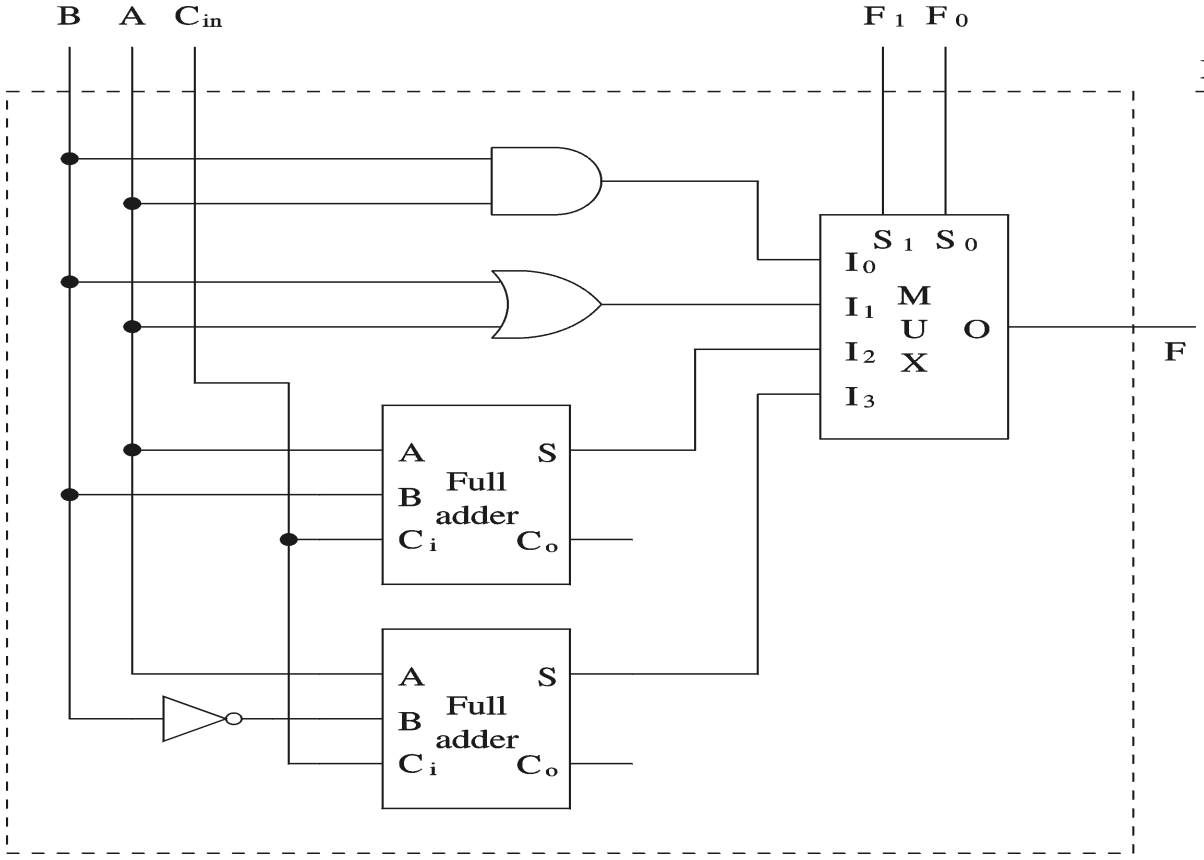
Programmable Array Logic Devices (cont'd)

- MUX selects the input
  - S_0 and S_1 are programmed through fuses F_0 and F_1



Arithmetic and Logic Unit

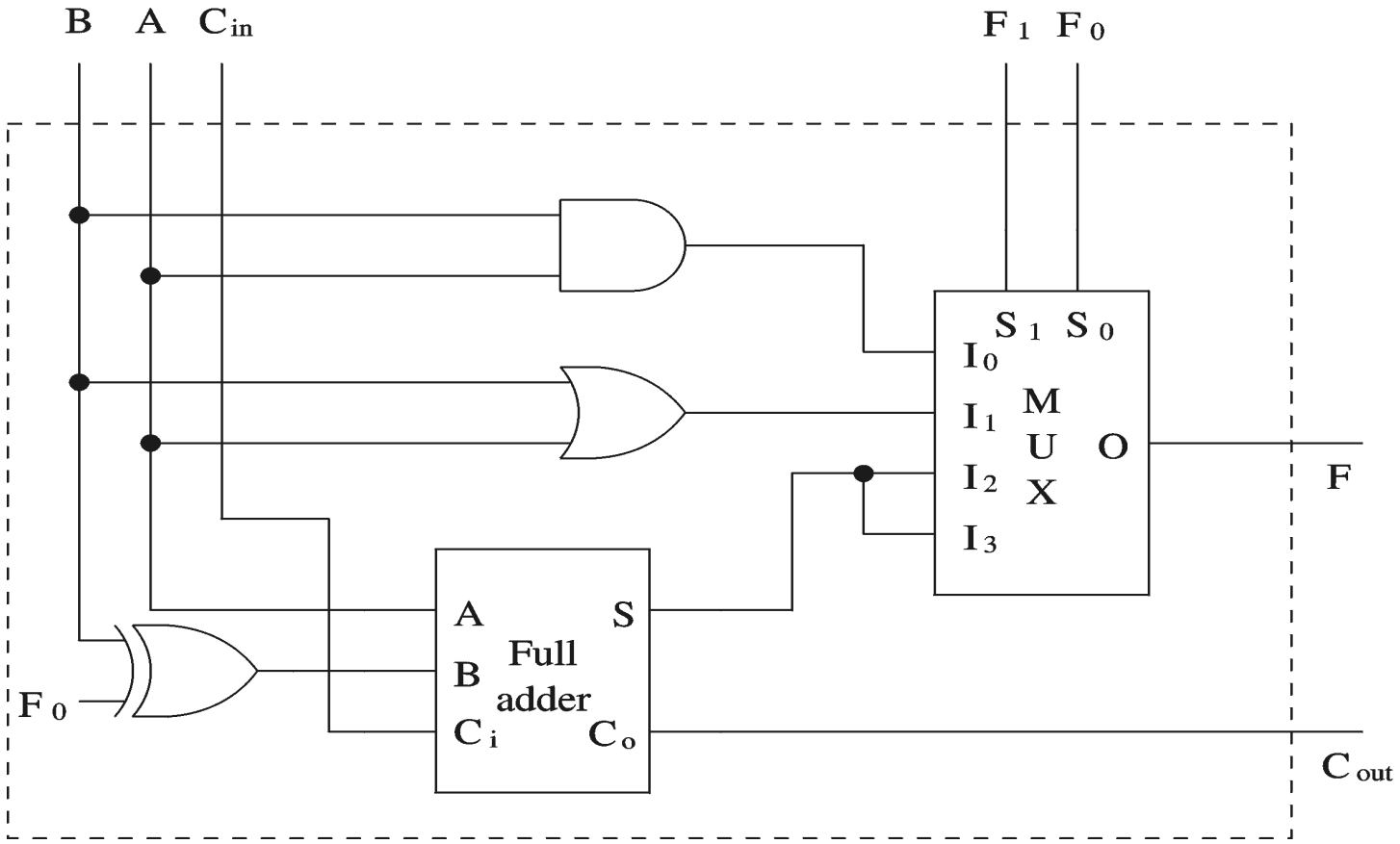
Preliminary ALU design



F1	F0	F
0	0	A and B
0	1	A or B
1	0	A + B
1	1	A − B

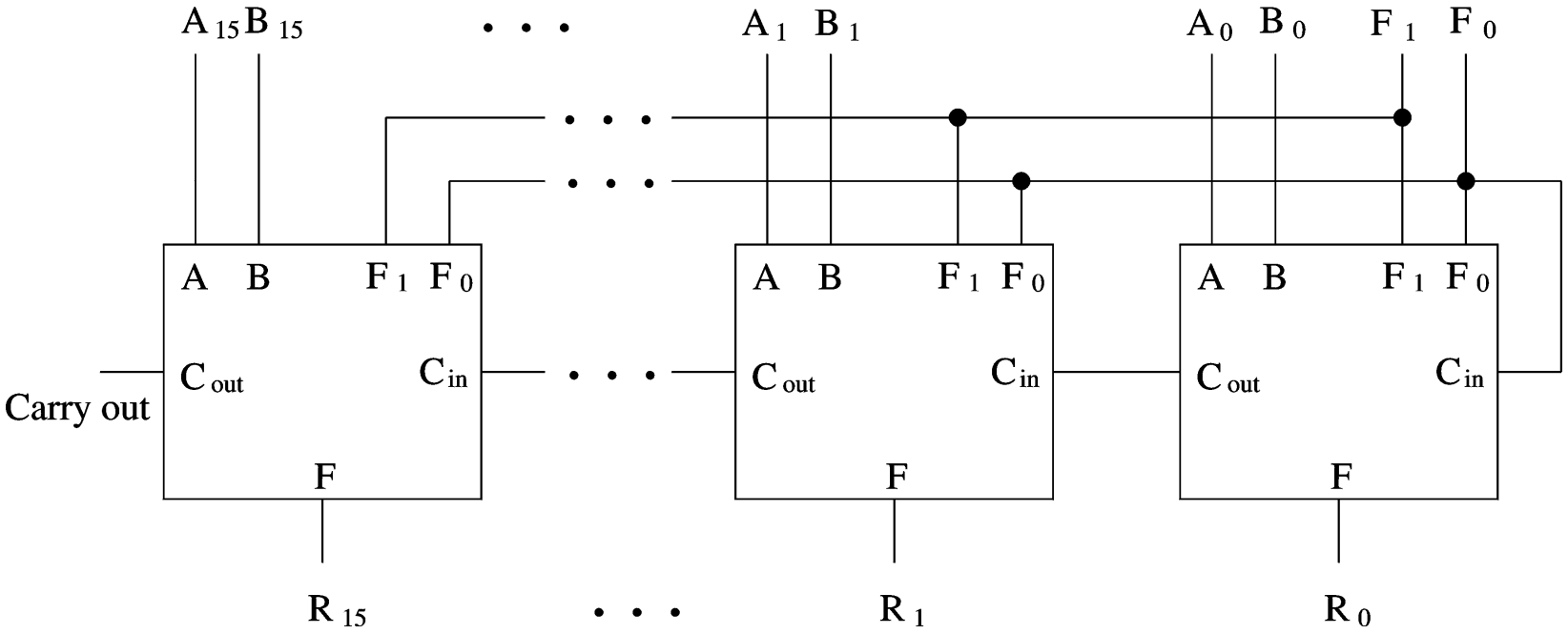
Arithmetic and Logic Unit (cont'd)

Final design



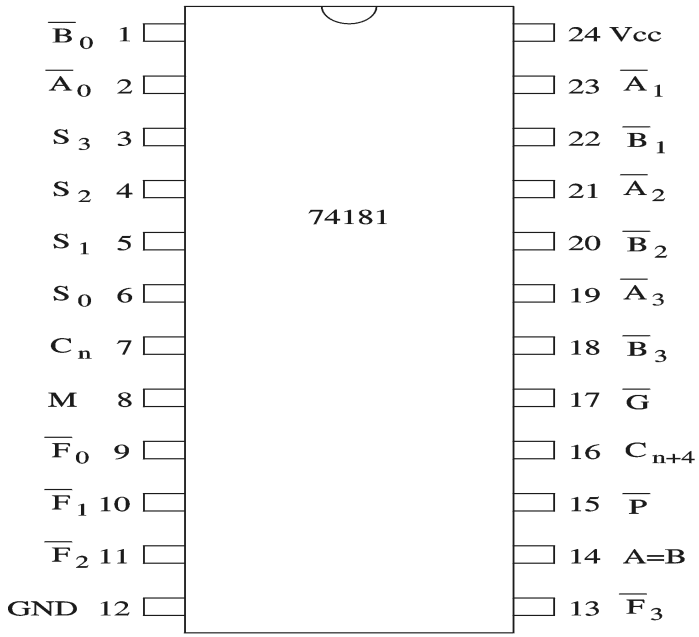
Arithmetic and Logic Unit (cont'd)

16-bit ALU

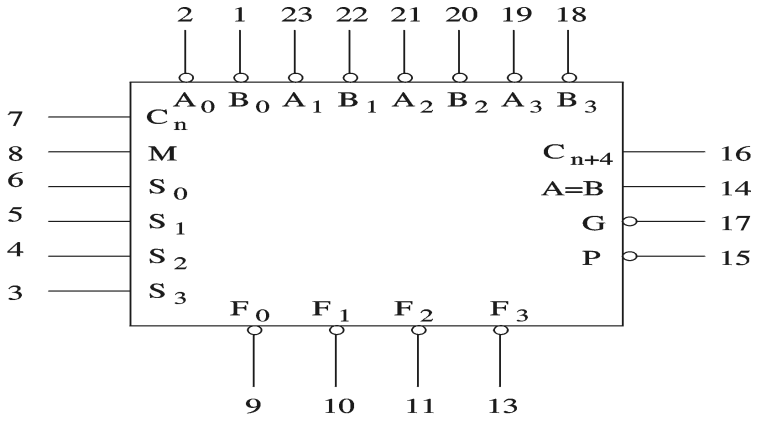


Arithmetic and Logic Unit (cont'd)

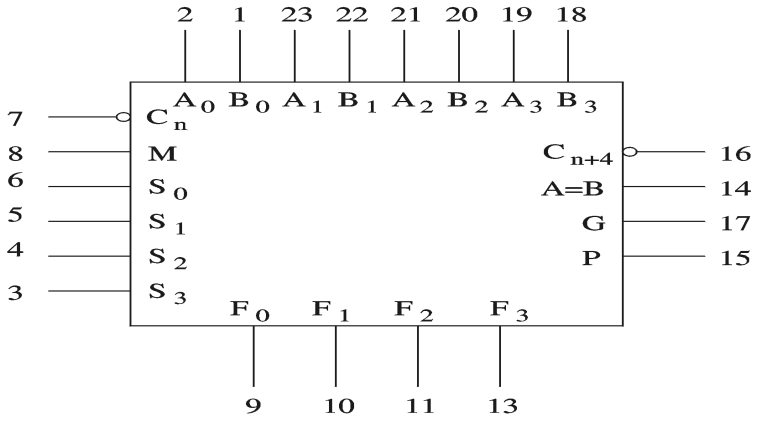
4-bit ALU



(a) Connection diagram



(b) Active low operands



(c) Active high operands

Summary

- Combinational circuits provide a higher level of abstraction
  - Output depends only on the current inputs
- Sample combinational circuits
  - Multiplexers and demultiplexers
  - Decoders and encoders
  - Comparators
  - Adders
    - Half-adder
    - Full-adder

Summary (cont'd)

- Programmable logic devices
  - PLAs
  - PALs
- Some more complete sets
  - Multiplexers
  - Decoder-OR
  - PLAs
  - PALs
- Looked at a very simple ALU design

Last slide