CIRCUIT IMPLEMENTATION STRATEGIES LOGIC ARRAYS

- PROGRAMMING LOGIC DEVICES
  - PROGRAMMABLE LOGIC ARRAY (PLA)
  - PROGRAMMABLE ARRAY LOGIC (PAL)
  - COMMERCIAL PROGRAMMABLE LOGIC DEVICES (CPLD's)
  - FIELD-PROGRAMMABLE GATE ARRAYS
- IMPLEMENTATION EXAMPLES
  - COMBINATIONAL CIRCUIT DESIGN USING PLA'S AND PAL'S
  - COMBINATIONAL CIRCUIT DESIGN USING MSI BUILDING BLOCKS
  - DESIGN EXAMPLE USING SEVERAL IMPLEMENTATION STRATEGIES

Revised 2005-02-03

PROGRAMMING LOGIC DEVICES

PROGRAMMABLE LOGIC ARRAY (PLA)

- CONTAINS AN AND-ARRAY AND AN OR-ARRAY
  - AND-ARRAY IS USED TO IMPLEMENT BOOLEAN TERMS
  - OR-ARRAY IS USED TO PRODUCE THE DESIRED BOOLEAN FUNCTION BY OR'ING THE TERMS GENERATED BY THE AND-ARRAY
  - ALL TERMS PRODUCED BY THE AND-ARRAY ARE ACCESSIBLE TO ALL OR-OUTPUT GATES OF THE OR-ARRAY, MEANING THAT TERMS CAN BE SHARED AMONG OUTPUT FUNCTIONS.
  - COMMERCIAL PLA'S USE TWO NOR PLAINS TO IMPLEMENT THE AND- AND OR-ARRAYS

PROGRAMMING LOGIC DEVICES

PROGRAMMABLE LOGIC ARRAY (PLA)

- PLA'S ARE PROGRAMMED BY 'BURNING' UNWANTED CONNECTIONS
- EXAMPLES FROM BROWN'S TEXTBOOK PAGES 96 AND 97:

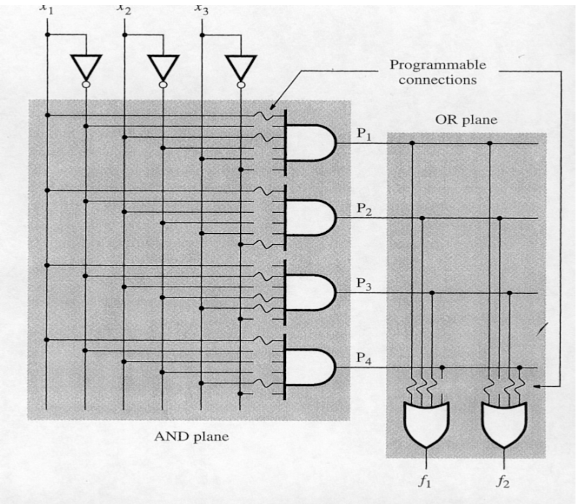


Figure 3.26 Gate-level diagram of a PLA.

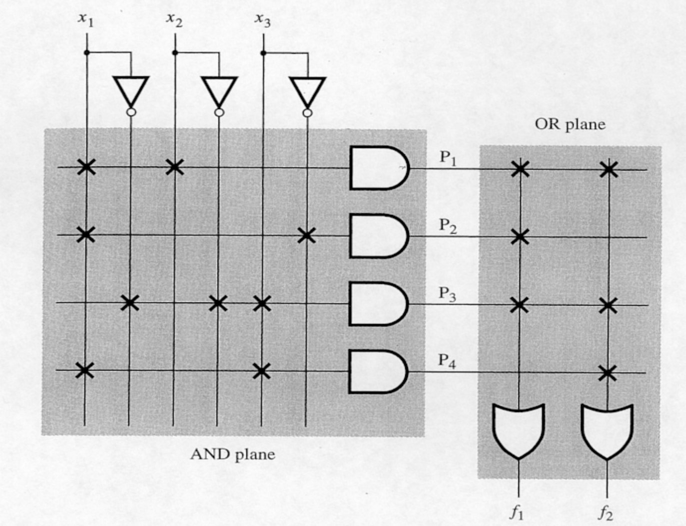


Figure 3.27 Customary schematic for the PLA in Figure 3.26.

PROGRAMMING LOGIC DEVICES

PROGRAMMABLE ARRAY LOGIC (PAL)

- CONTAINS AN AND-ARRAY AND AN OR-ARRAY
  - AND-ARRAY IS USED TO IMPLEMENT BOOLEAN TERMS
  - OR-ARRAY IS USED TO PRODUCE THE DESIRED BOOLEAN FUNCTION BY OR'ING THE TERMS GENERATED BY THE AND-ARRAY
  - A FIXED NUMBER OF TERMS PRODUCED BY THE AND-ARRAY ARE ACCESSIBLE TO EACH ONE OF THE OR-OUTPUT GATES OF THE OR-ARRAY, MEANING THAT TERMS CANNOT BE SHARED AMONG OUTPUT FUNCTIONS.
  - COMMERCIAL PAL'S USE TWO NOR PLAINS TO IMPLEMENT THE AND- AND OR-ARRAYS

PROGRAMMING LOGIC DEVICES

PROGRAMMABLE ARRAY LOGIC (PAL)

- PAL'S ARE PROGRAMMED BY 'BURNING' UNWANTED CONNECTIONS
- EXAMPLE FROM BROWN'S TEXTBOOK PAGE 98:

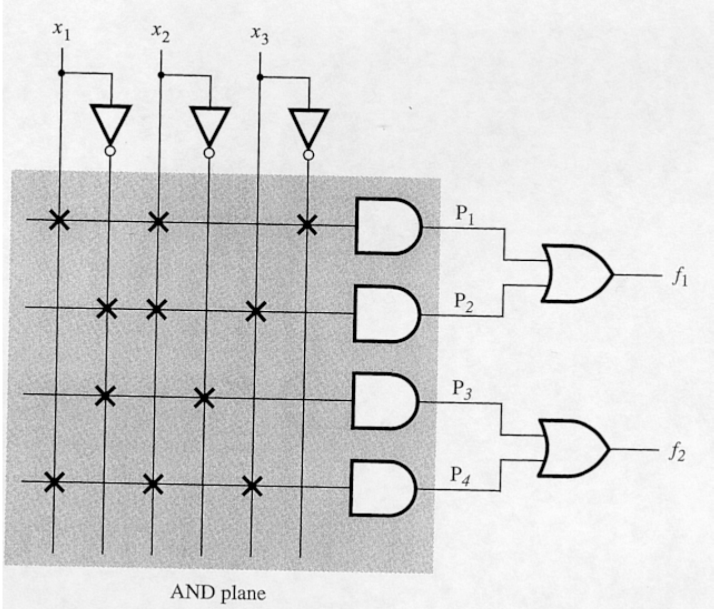


Figure 3.28 An example of a PAL.

PROGRAMMING LOGIC DEVICES

SIMPLE COMMERCIAL PROGRAMMABLE LOGIC DEVICES (CPLD’s)

- THEY ARE GENERALLY PAL DEVICES WHERE ADDITIONAL FEATURES HAVE BEEN ADDED, SUCH AS:
  - BIDIRECTIONAL OUTPUTS BY USING A TRI-STATE GATE AT THE OUTPUT OF EACH OR-GATE.
  - FEEDBACK LINES
  - A STORAGE DEVICE (D-FLIPFLOP) AT EACH OUTPUT, TO STORE OR NOT, THE CORRESPONDING OUTPUT FUNCTION OR ITS COMPLEMENT.
  - THESE ADDITIONAL FEATURES ARE SHOWN ON THE NEXT SLIDE

PROGRAMMING LOGIC DEVICES

SIMPLE COMMERCIAL PROGRAMMABLE LOGIC DEVICES (CPLD’s)

- EXAMPLE: Extra circuitry added to OR-gate outputs of a PAL example (From Brown’s Textbook page 99)

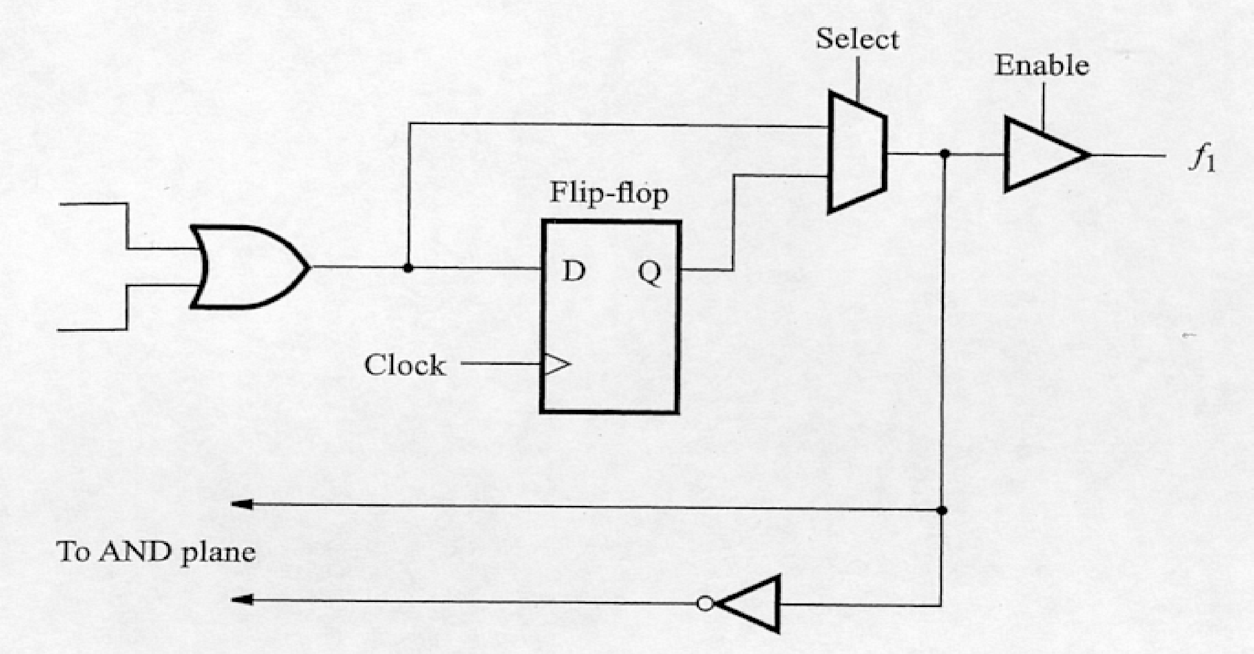


Figure 3.29 Extra circuitry added to OR-gate outputs from Figure 3.28.

PROGRAMMING LOGIC DEVICES

COMPLEX COMMERCIAL PROGRAMMABLE LOGIC DEVICES (CPLD's)

- EXAMPLE: (Page 102 of Brown's Textbook)

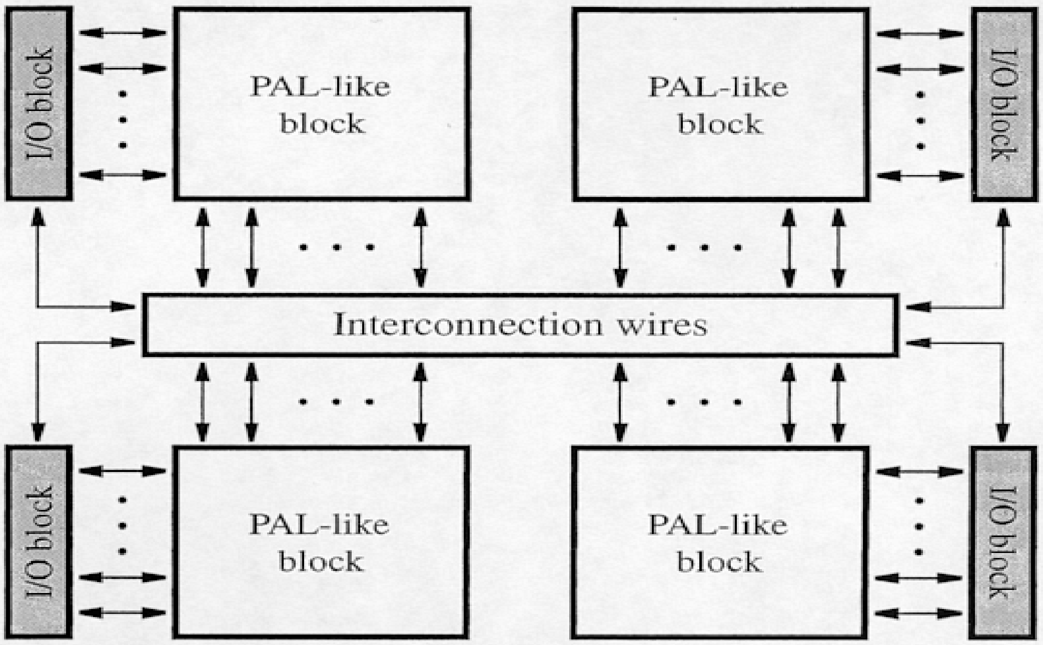


Figure 3.32 Structure of a complex programmable logic device (CPLD).

PROGRAMING LOGIC DEVICES

COMPLEX COMMERCIAL PROGRAMMABLE LOGIC DEVICES (CPLD's)

- EXAMPLE: A Section of the CPLD in previous slide (Page 103 of Brown's Textbook)

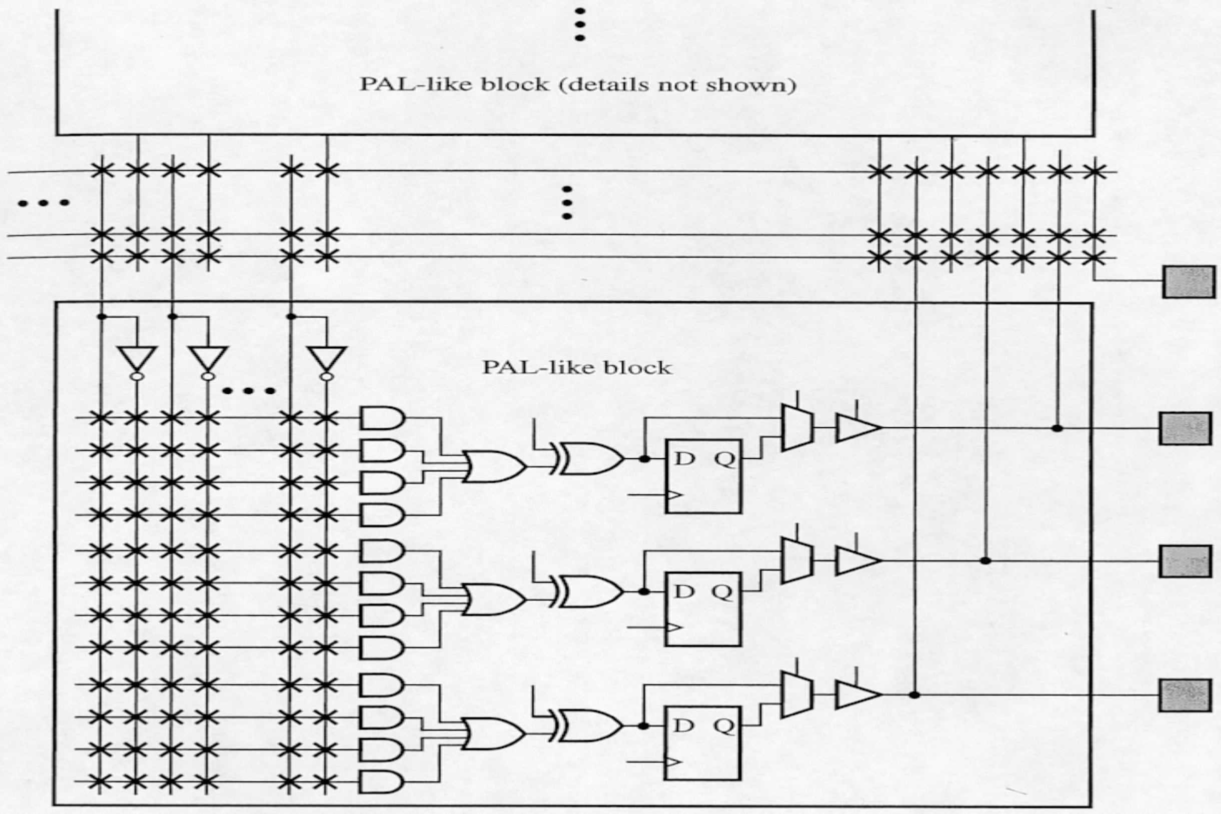


Figure 3.33 A section of the CPLD in Figure 3.32.

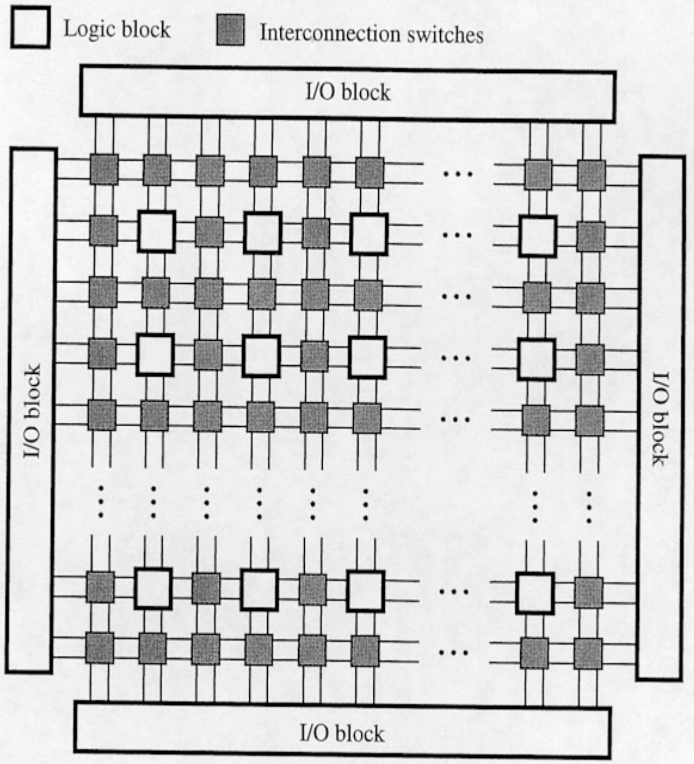
PROGRAMMING LOGIC DEVICES

FIELD-PROGRAMMABLE GATE ARRAYS (FPGA'S)

- FPGA'S USE LUT'S INSTEAD OF THE AND-OR ARRAYS OF PLA'S AND PAL'S.
- THE LUT'S ARE CONNECTED (PROGRAMMED) THROUGH INTERCONNECTION SWITCHES.
- THEY HAVE EXTRA CIRCUITRY,SUCH AS STORAGE CELLS AND MUX's, AS IN PAL'S.
- STORAGE CELLS ARE VOLATILE. AT POWER-ON, A PROM LOADS AUTOMATICLY THE CELLS CONTENTS.
- THE PROM IS HOUSED ON THE SAME CIRCUIT BOARD.

PROGRAMING LOGIC DEVICES

FIELD-PROGRAMMABLE GATE ARRAYS: EXAMPLE FROM BROWN’S TEXTBOOK, PAGES 106, 108.



(a) General structure of an FPGA

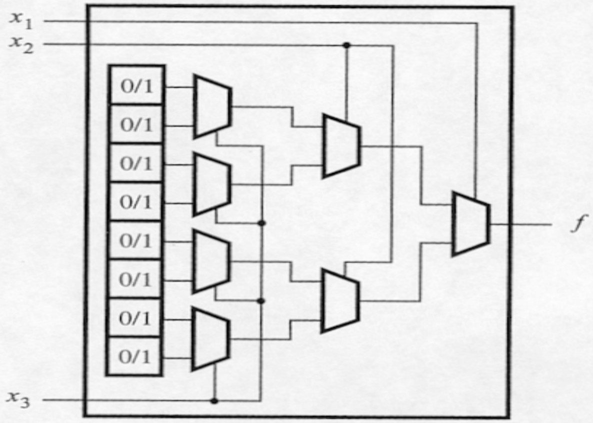
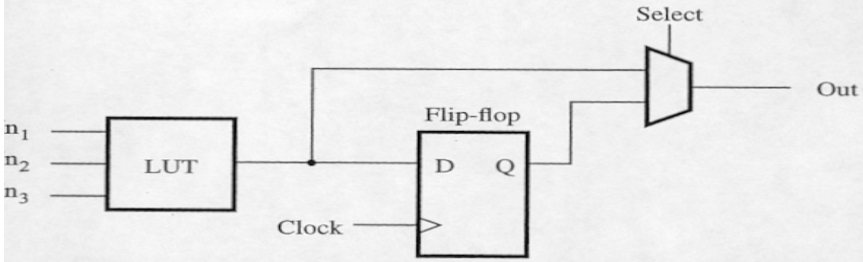


Figure 3.37 A three-input LUT.



igure 3.38 Inclusion of a flip-flop in an FPGA logic block.

IMPLEMENTATION EXAMPLES

FPG’S: EXAMPLE CONTINUES FROM BROWN’S TEXTBOOK, PAGES 109.

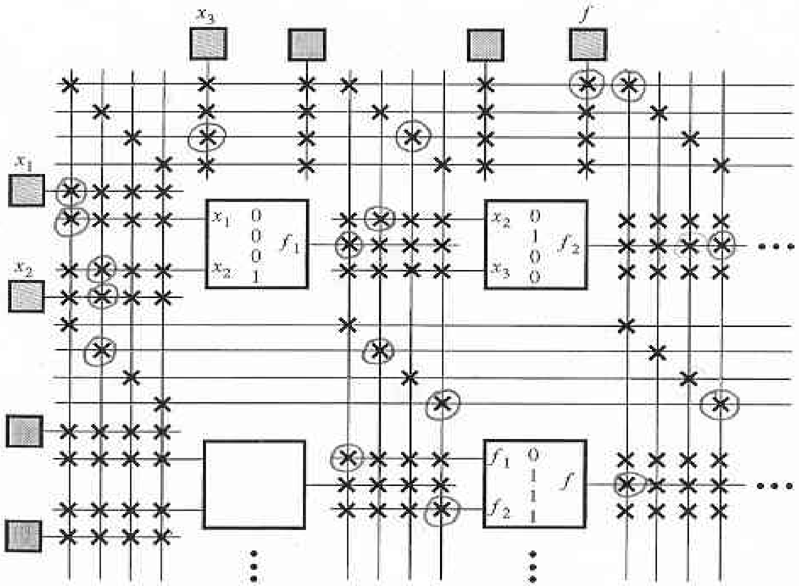


Figure 3.39 A section of a programmed FPGA.

NOTE: BLUE CONNECTIONS IN TEXTBOOK ARE SHOWN HERE WITH A CIRCLE

IMPLEMENTATION EXAMPLES

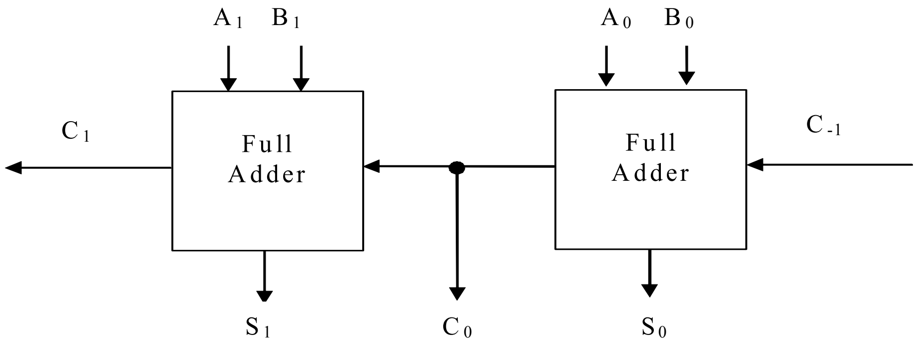
COMBINATIONAL CIRCUIT DESIGN USING PLA'S

Example 1: The PLA device shown on the next slide is to be used in the design of a combinational circuit that will produce the following set of switching functions:

$S_i = A_i\ !B_i\ !C_{i-1} + !A_i\ B_i\ !C_{i-1} + !A_i\ !B_i\ C_{i-1} + A_i\ B_i\ C_{i-1}$

$C_i = A_i\ B_i\ + A_i\ C_{i-1} + B_i\ C_{i-1}$

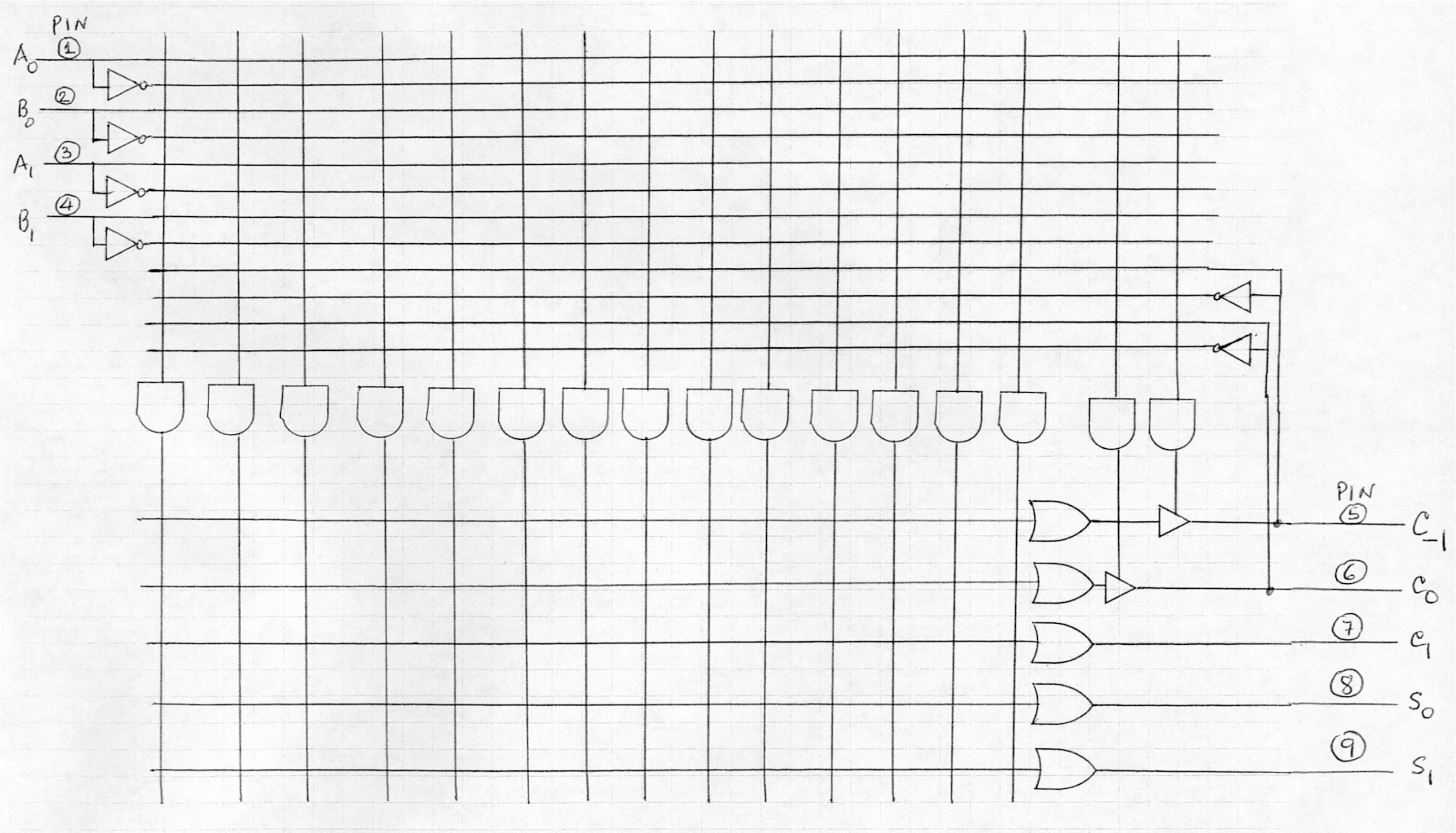
where i = 0,1, which corresponds to the block diagram shown below



IMPLEMENTATION EXAMPLES

COMBINATIONAL CIRCUIT DESIGN USING PLA'S

Example 1 (Continues): Commercial PLA device to be used in this design example



IMPLEMENTATION EXAMPLES

COMBINATIONAL CIRCUIT DESIGN USING PLA'S

Example 1 (continues):

- a) Give the algebraic expression of the 4 functions in such a way that they can be produced by the given PLA device.
- b) On the drawing of the PLA device, indicate the correct connections to produce the given 4 functions.

Solution: a) The equations are

$S_0 = A_0\ !B_0\ !C_{-1} + !A_0\ B_0\ !C_{-1} + !A_0\ !B_0\ C_{-1} + A_0\ B_0\ C_{-1}$

$C_0 = A_0\ B_0 + A_0\ C_{-1} + B_0\ C_{-1}$

$S_1 = A_1\ !B_1\ !C_0 + !A_1\ B_1\ !C_0 + !A_1\ !B_1\ C_0 + A_1\ B_1\ C_0$

$C_1 = A_1\ B_1 + A_1\ C_0 + B_1\ C_0$

Revised 2005-02-03

IMPLEMENTATION EXAMPLES

COMBINATIONAL CIRCUIT DESIGN USING PLA'S

Example 1 (continues):

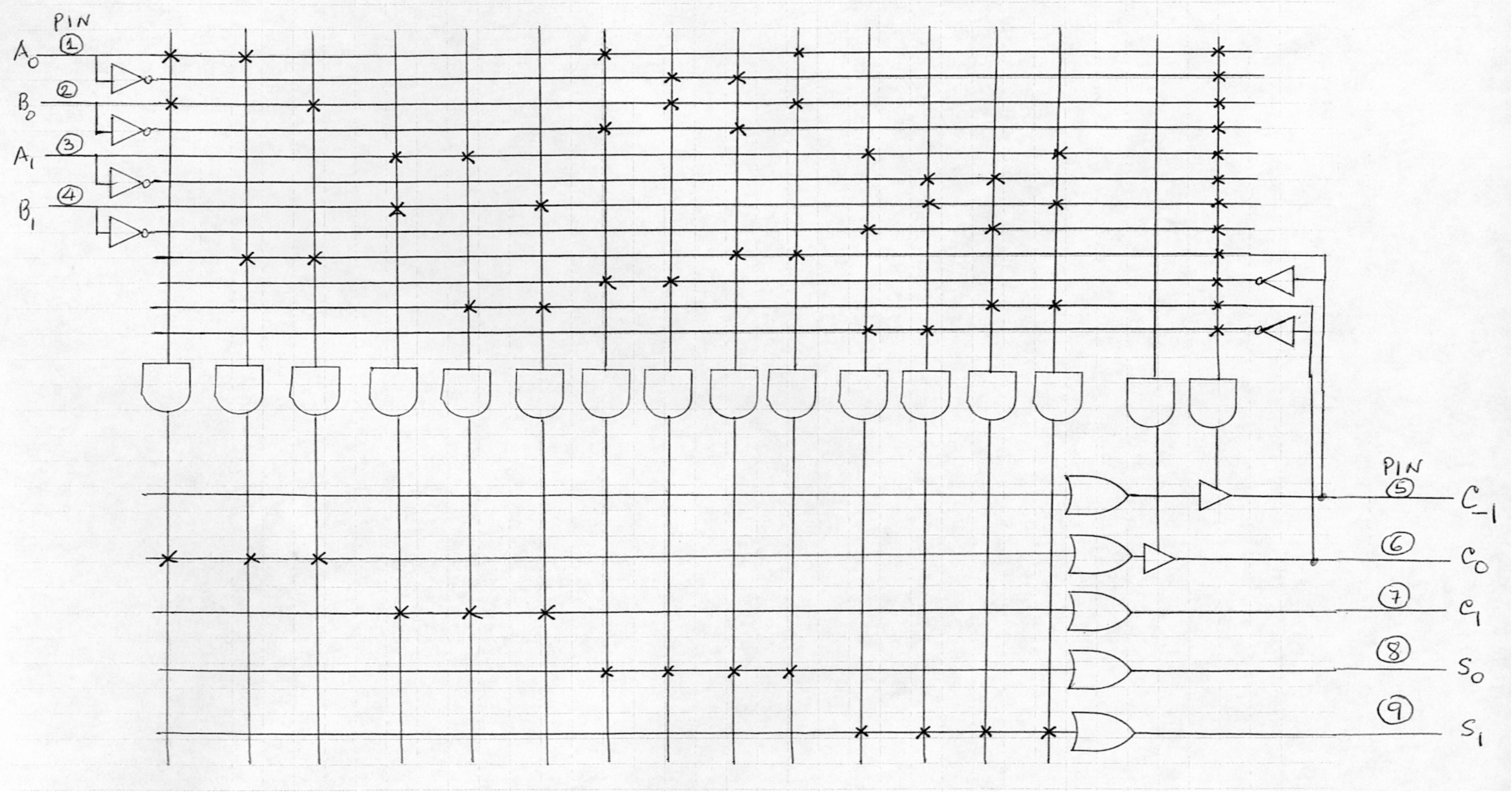
b) Note that the adder requires 5 inputs and that there are only 4 dedicated input pins in the given PLA device. Bidirectional pin 5 is used as another input by conveniently programming its tri-state gate. The carry term C_0 is used to compute S_1 and C_1 through the feedback line from pin 6, thus allowing C_0 to be combined with A_1 and B_1 as required by the equations shown in part a).

The final circuit is shown on the next slide.

IMPLEMENTATION EXAMPLES

COMBINATIONAL CIRCUIT DESIGN USING PLA'S

Example (continues): Final PLA circuit



IMPLEMENTATION EXAMPLES

COMBINATIONAL CIRCUIT DESIGN USING PAL'S

Example 2: On the given PAL device drawing, shown on the next slide, indicate the correct connections that will produce the following set of equations:

F_1 = !E F !D + !E F C + B

F_2 = !D + C + !E F B

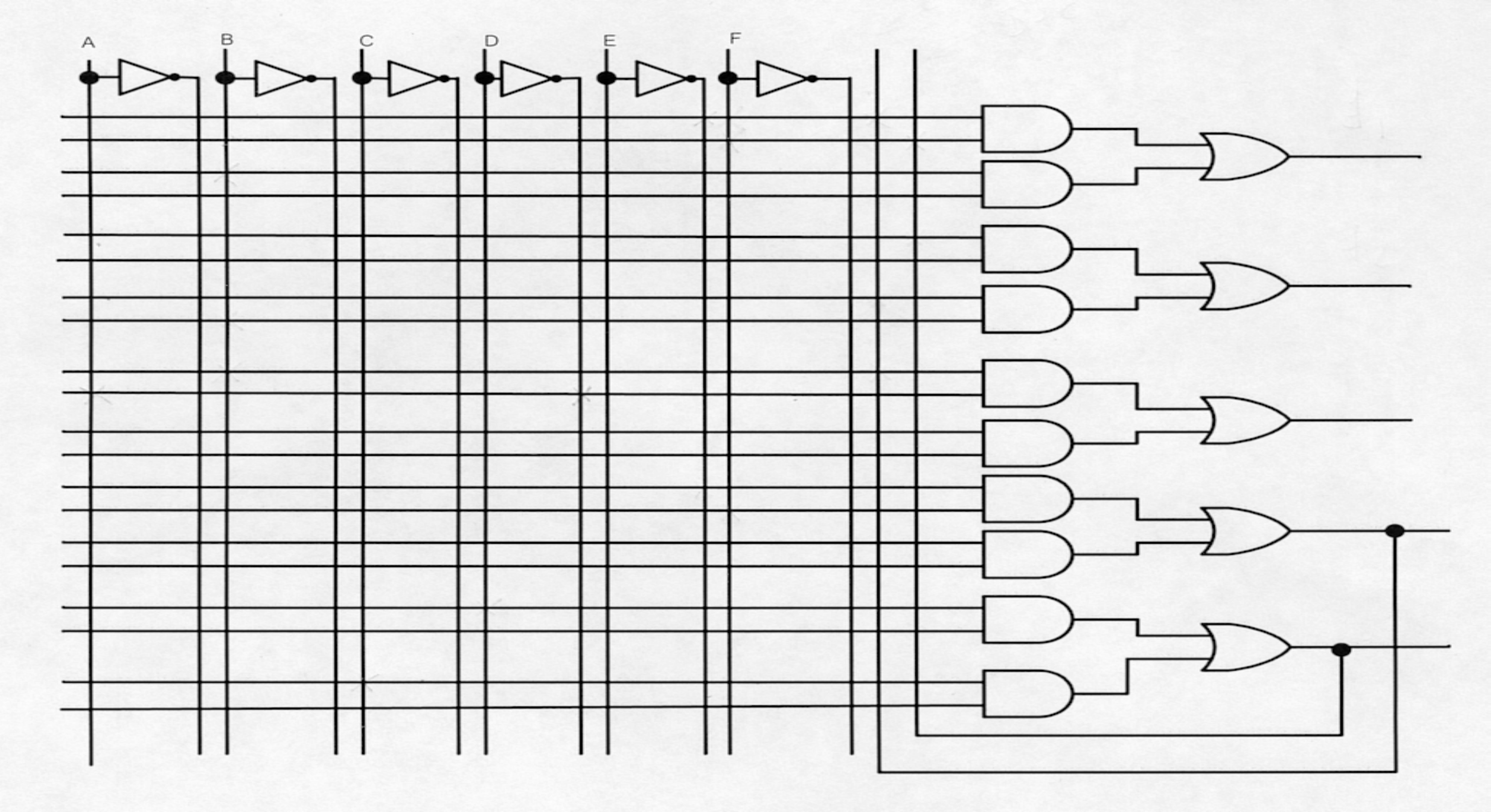
F_3 = !E F B A !D

Revised 2005-02-03

IMPLEMENTATION EXAMPLES

COMBINATIONAL CIRCUIT DESIGN USING PAL'S

Example 2 (Continues) : Drawing of the PAL to be used in this example



IMPLEMENTATION EXAMPLES

COMBINATIONAL CIRCUIT DESIGN USING PAL'S

Example 2 (Continues):

SOLUTION:

F_1 = !E F !D + !E F C + B = (!E F) (!D + C) + B

F_2 = !D + C + !E F B = (!D + C) + (!E F) B

F_3 = !E F B A !D = (!E F) B A !D

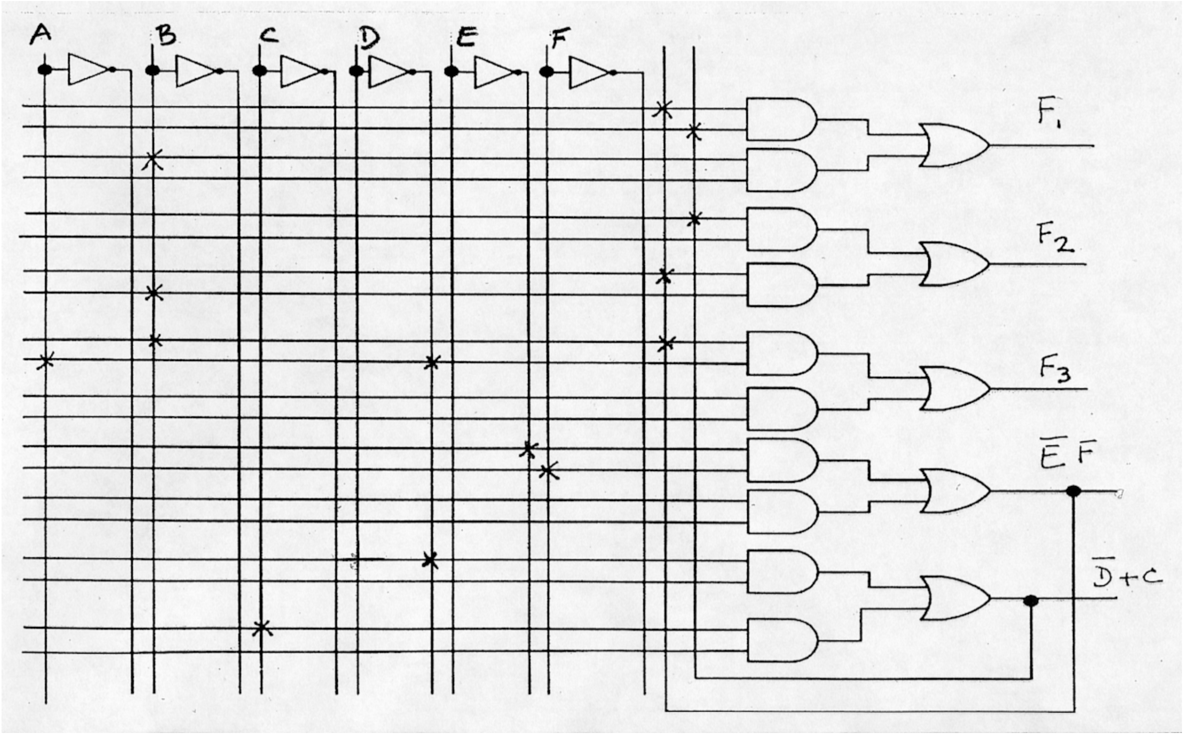
By factoring, the feedback functions (!E F) and (!D + C) were found, making the implementation feasible.

The complete circuit is shown on the next slide

IMPLEMENTATION EXAMPLES

COMBINATIONAL CIRCUIT DESIGN USING PAL'S

Example 2 (Continues): Complete circuit



IMPLEMENTATION EXAMPLES

COMBINATIONAL CIRCUIT DESIGN USING MSI BUILDING BLOCKS

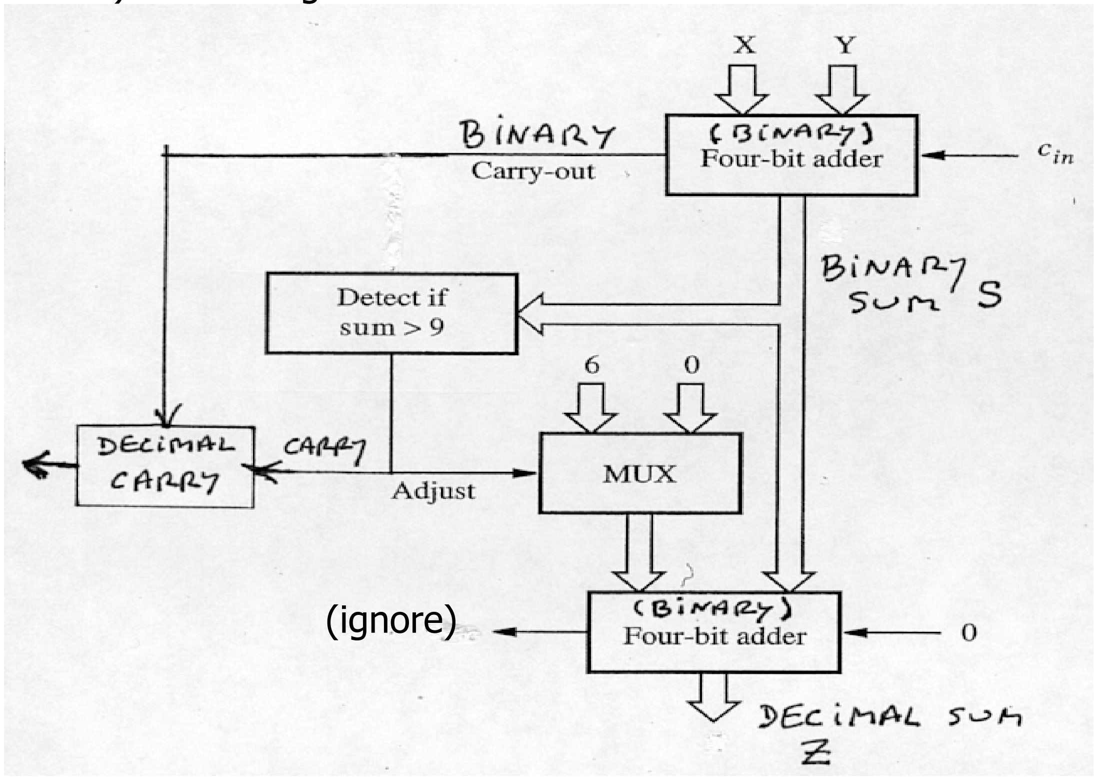
- A block diagram is produced using the available MSI BUILDING BLOCKS and showing the behavior of the circuit to be designed.
- Example: Design a ONE-DIGIT Binary-Coded-Decimal (BCD)-Adder using 4-bit binary adders.
  - Solution: There are two cases to consider:
    - Case I : Let the result be Z = X + Y , where X and Y are the two decimal digits to be added. If Z ≤ 9, then the binary sum S = Z and the binary carry-out = 0. If Z > 9, then Z = S + 6 and decimal carry-out = 1.
    - Case II : When X + Y > 15 (Example: 9 + 8 =17), then the binary sum S must be corrected by adding 6 to S to produce Z = S + 6; the binary carry-out = 1 and the decimal carry-out =1.

THE BLOCK DIAGRAM IS SHOWN ON NEXT SLIDE

IMPLEMENTATION EXAMPLES

COMBINATIONAL CIRCUIT DESIGN USING MSI BUILDING BLOCKS

- Example (Continues): Block diagram of the ONE-DIGIT BCD-adder

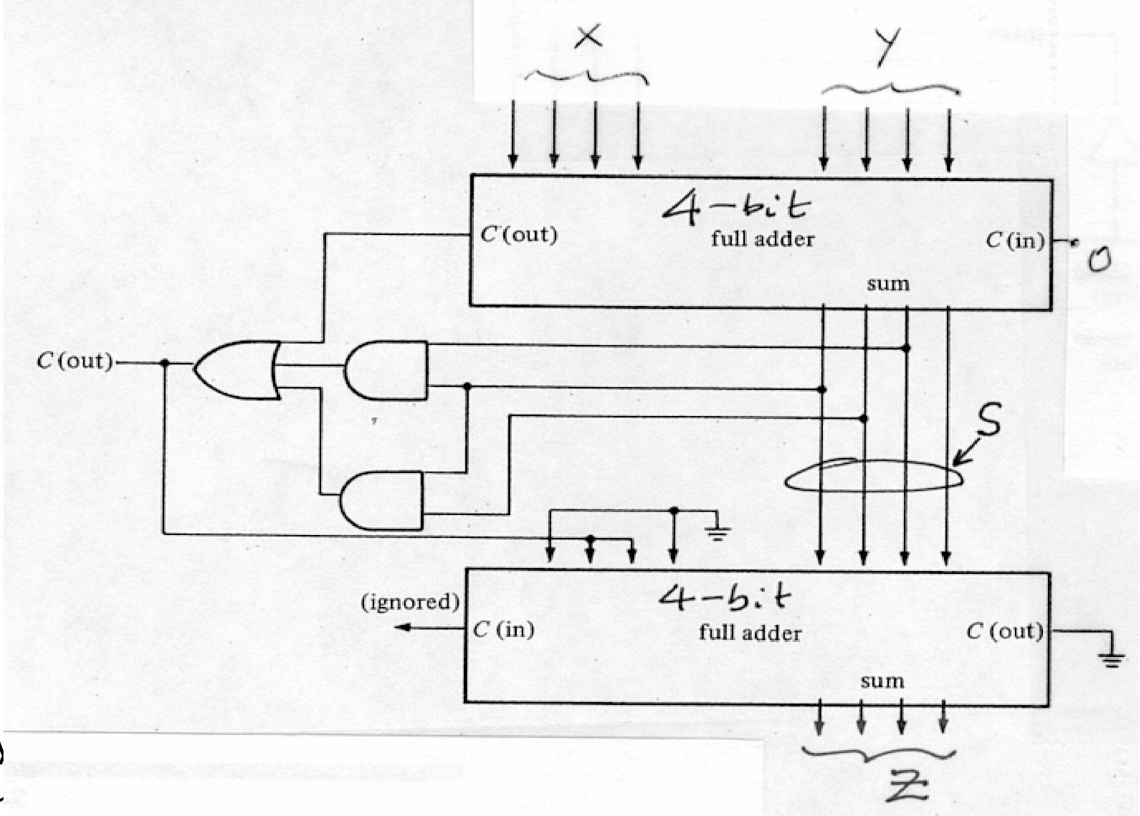


decimal carry-out

IMPLEMENTATION EXAMPLES

COMBINATIONAL CIRCUIT DESIGN USING MSI BUILDING BLOCKS

- Example (Continues): Complete Circuit



IMPLEMENTATION EXAMPLES

- DESIGN EXAMPLE USING SEVERAL IMPLEMENTATION STRATEGIES
  - Design a 2-bit Binary-Coded-Base-3 Multiplier using five different implementation strategies:
    - Classic, two-level gate design
    - MUX-based design
    - ROM-based design
    - PLA-based design
    - PAL-based design

  Show on a table the gate count and packages count for each strategy.

IMPLEMENTATION EXAMPLES

- Design a 2-bit Binary-Coded-Base-3 Multiplier using five different implementation strategies:

Classic, two-level gate design.

Solution: The input variables are:

$\underline{a} = \{a_1, a_0\}$, $\underline{b} = \{b_1, b_0\}$

The output variables are:

$\underline{c} = \{c_3, c_2, c_1, c_0\}$, such that

$\underline{c} = \underline{a} \times \underline{b}$, where x represents arithmetic multiplication.

The base-3 digits, “0”, “1” and “2”are coded 00, 01, 10 respectively.

The multiplication table is

$\underline{b}$ \ $\underline{a}$	0	1	2
0	00	00	00
1	00	01	02
2	00	02	11

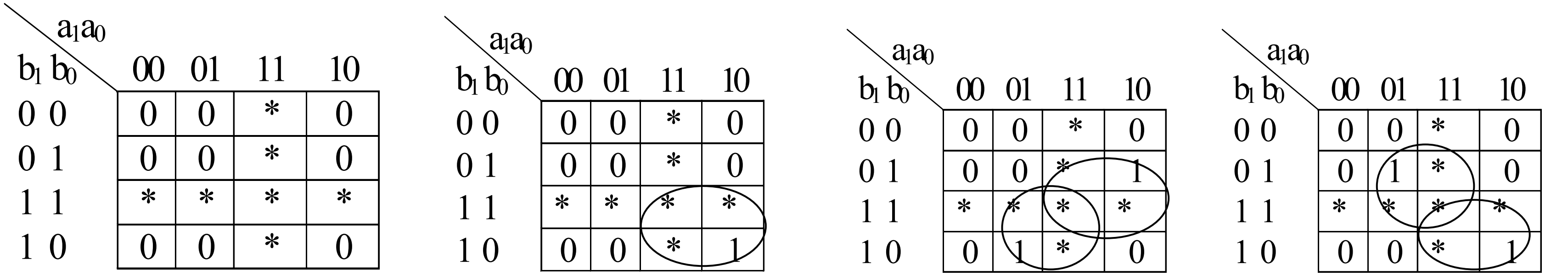
$\underline{C} = \{c_3, c_2, c_1, c_0\}$

IMPLEMENTATION EXAMPLES

- Design a 2-bit Binary-Coded-Base-3 Multiplier

Classic, two-level gate design continues.

The K-maps of the output variables are



$b_1 b_0$ \ $a_1 a_0$	00	01	11	10
0 0	0	0	*	0
0 1	0	0	*	0
1 1	*	*	*	*
1 0	0	0	*	0

$C_3 = 0$

$b_1 b_0$ \ $a_1 a_0$	00	01	11	10
0 0	0	0	*	0
0 1	0	0	*	0
1 1	*	*	*	*
1 0	0	0	*	1

$C_2 = a_1 b_1$

$b_1 b_0$ \ $a_1 a_0$	00	01	11	10
0 0	0	0	*	0
0 1	0	0	*	1
1 1	*	*	*	*
1 0	0	1	*	0

$C_1 = a_1 b_0 + a_0 b_1$

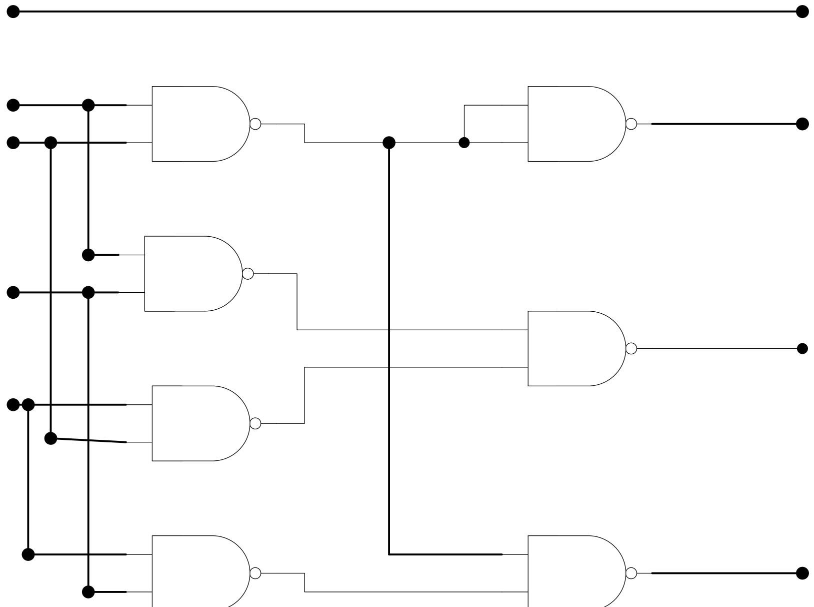
$b_1 b_0$ \ $a_1 a_0$	00	01	11	10
0 0	0	0	*	0
0 1	0	1	*	0
1 1	*	*	*	*
1 0	0	0	*	1

$C_0 = a_1 b_1 + a_0 b_0$

IMPLEMENTATION EXAMPLES

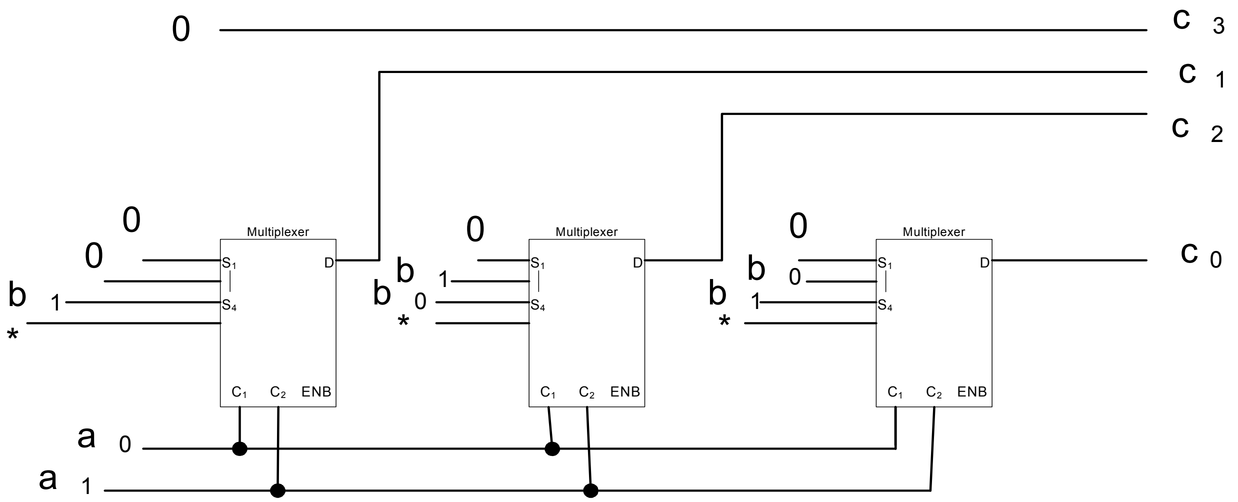
- Design a 2-bit Binary-Coded-Base-3 Multiplier

Classic, two-level gate design continues. NAND-NAND circuit.



IMPLEMENTATION EXAMPLES

- Design a 2-bit Binary-Coded-Base-3 Multiplier
  - MUX-based design:
  - Selecting a_0 , a_1 , as the control variables to the MUX' s, the data line functions can be read directly from the K-maps on the previous slide.
  - The resulting circuit is shown below.



Revised 2005-02-03

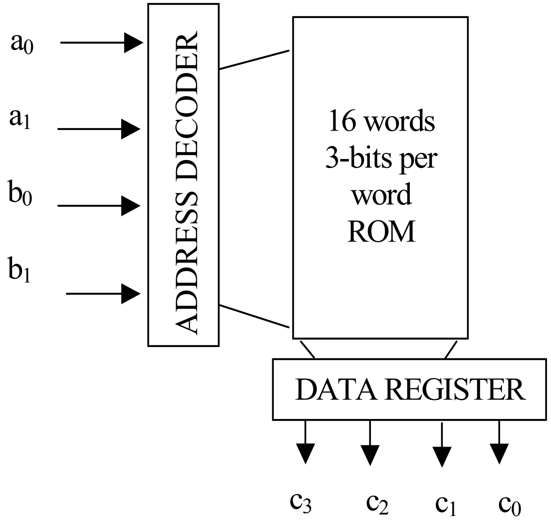
IMPLEMENTATION EXAMPLES

- Design a 2-bit Binary-Coded-Base-3 Multiplier
  - ROM-based design
  - The address of the ROM represents the valuation of the input variables and the contents, at that address, corresponds to the value of the output variables for that valuation of the input variables.
  - An address/contents table is produced and the ROM is programmed accordingly.

BASE-3 DIGITS		ADDRESS				CONTENTS			
b	a	b_1	b_0	a_1	a_0	c_3	c_2	c_1	c_0
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0
0	2	0	0	1	0	0	0	0	0
0	*	0	0	1	1	*	*	*	*
1	0	0	1	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	1
1	2	0	1	1	0	0	0	1	0
1	*	0	1	1	1	*	*	*	*
2	0	1	0	0	0	0	0	0	0
2	1	1	0	0	1	0	0	1	0
2	2	1	0	1	0	0	1	0	1
2	*	1	0	1	1	*	*	*	*
*	0	1	1	0	0	0	0	0	0
*	1	1	1	0	1	*	*	*	*
*	2	1	1	1	0	*	*	*	*
*	*	1	1	1	1	*	*	*	*

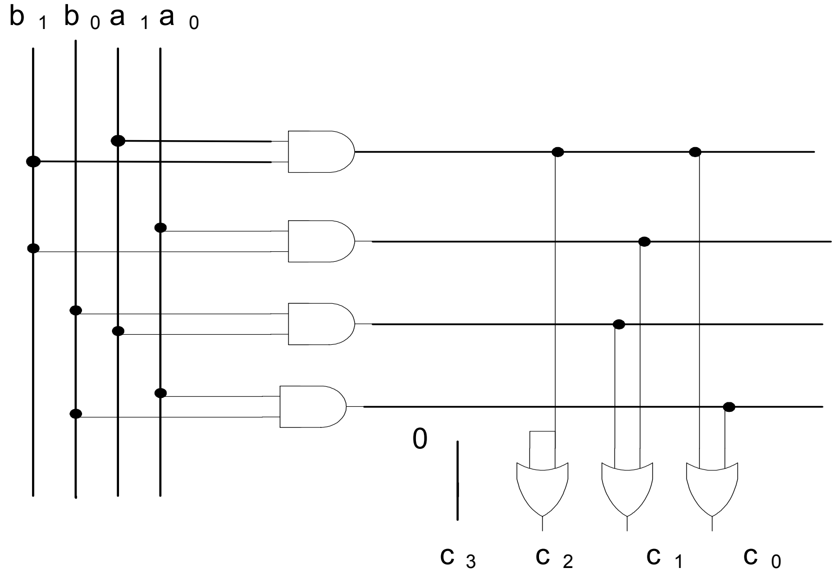
IMPLEMENTATION EXAMPLES

- Design a 2-bit Binary-Coded-Base-3 Multiplier
  - ROM-based design: Final circuit



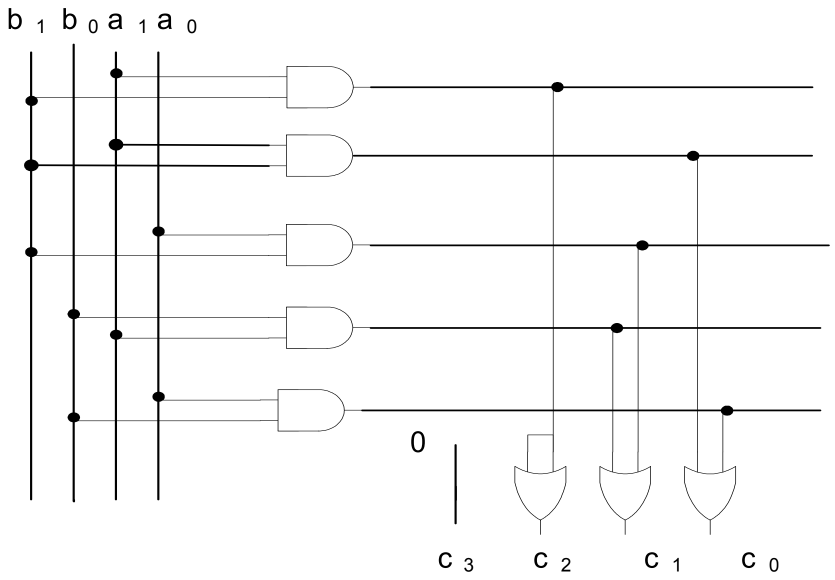
IMPLEMENTATION EXAMPLES

- Design a 2-bit Binary-Coded-Base-3 Multiplier
  - PLA-based design: Note that the term a_1 b_1 can be shared among output functions c_0 and c_2 .



IMPLEMENTATION EXAMPLES

- Design a 2-bit Binary-Coded-Base-3 Multiplier
  - PAL-based design: Note that NO SHARING of terms is allowed in PAL-based design and, therefore, term a_1 b_1 must be implemented twice



Revised 2005-02-03

IMPLEMENTATION EXAMPLES

- Design a 2-bit Binary-Coded-Base-3 Multiplier
- IMPLEMENTATION STATEGIES COMPARISON:
  - In order to compare the different strategies used in this design we choose the following metrics:
    - 1) The number of gate-equivalences and
    - 2) The number of packages.
  - We assume that the 7400 Series is used for the classic, two-level NAND-NAND design. For example: the 7400 IC package, called QUAD, contains 4 2-input NAND gates).
  - The number of gates used in a 4 x 1 MUX is equal to 7 (cf. Brown's Textbook, 2nd Ed. Page 317). One package houses 2 MUX's.
  - Finally we assume that in the ROM-based design, the decoder requires 19 gates, 16 for each minterm and 3 gates for the output level.
  - The following table shows the results.

IMPLEMENTATION EXAMPLES

- Design a 2-bit Binary-Coded-Base-3 Multiplier
  - IMPLEMENTATION STATEGIES COMPARISON RESULTS

Design Approach	Number of Packages	Total number of gate equivalences
Gate	2	7
MUX	2	21
ROM	1	19
PLA	1	7
PAL	1	8

  - The design using the least number of packages and the least number of gates is the PLA-based design.