PROGRAMMABLE MODULES

- SPECIFICATION OF PROGRAMMABLE COMBINATIONAL
  AND SEQUENTIAL MODULES

  1. PSA

  2. ROM

  3. FPGA

- THE WAY THE MODULES ARE PROGRAMMED

- NETWORKS OF PROGRAMMABLE MODULES

- EXAMPLES OF USES

PROGRAMMABLE SEQUENTIAL ARRAYS (PSA)
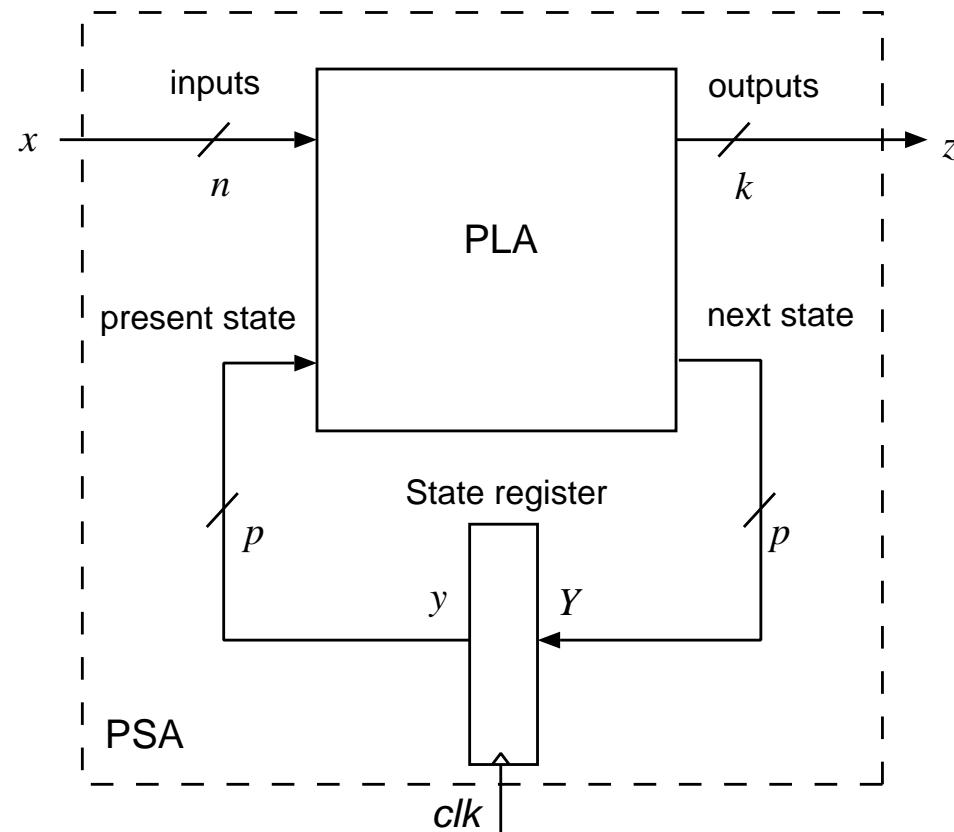
Figure 12.1: PROGRAMMABLE SEQUENTIAL ARRAY (PSA).

- SEQUENCE GENERATOR

  INPUTS: $x \in \{0, 1\}$
  OUTPUTS: $z \in \{0, 1, 3, 6, 7, 10, 14\}$

  FUNCTION: The transition and output functions

$$x = 0 \ : \ z = 0 \rightarrow 10 \rightarrow 14 \rightarrow 7 \rightarrow 0 \cdots$$

$$x = 1 \ : \ z = 1 \rightarrow 10 \rightarrow 3 \rightarrow 6 \rightarrow 1 \cdots$$

$$x = 0 \ : \ z = 0000 \rightarrow 1010 \rightarrow 1110 \rightarrow 0111 \rightarrow 0000 \cdots$$

$$x = 1 \ : \ z = 0001 \rightarrow 1010 \rightarrow 0011 \rightarrow 0110 \rightarrow 0001 \cdots$$
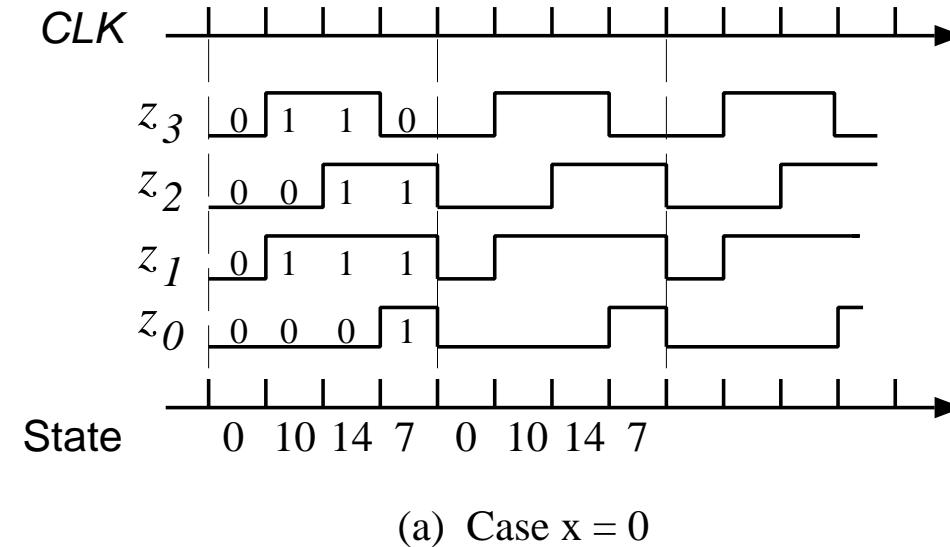
(a) Case x = 0
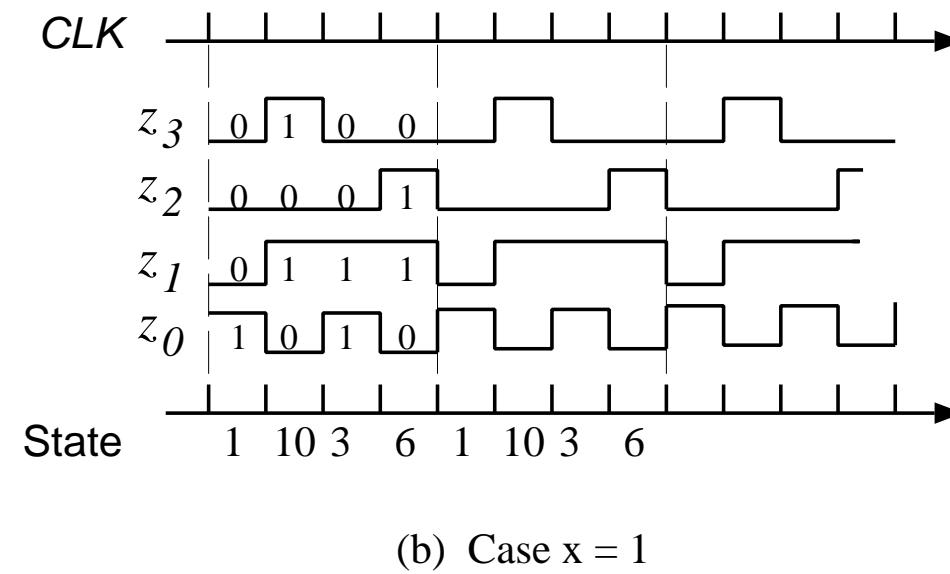
(b) Case x = 1

Figure 12.2: TIMING SEQUENCES IN Example 12.1.

Example 12.1 (cont.)

y	$k = 0$	$k = 1$	
0	10	—	k
1	—	10	k
3	—	6	x
6	0	1	k
7	0	1	k
10	14	3	k
14	7	—	x
	Y		K

$y \in \{2, 4, 5, 8, 9, 11, 12, 13, 15\}$ – don't care states

$$
\begin{aligned}
K &= xy_3y_2 + xy_3'y_2'y_1 + ky_1' + ky_3'y_2 + ky_3y_2' \\
Y_3 &= y_1' + y_2'k' \\
Y_2 &= y_3'y_2'y_1 + y_3k' \\
Y_1 &= y_2' + y_3 \\
Y_0 &= y_3k + y_2k + y_3y_2
\end{aligned}
$$

Figure 12.3: PSA IMPLEMENTATION IN Example 12.1.

READ-ONLY MEMORIES (ROM)

E

En

x_0

x_1

Address Inputs

2^n X k

ROM

x_{n-1}

z_{k-1} z_0

Outputs
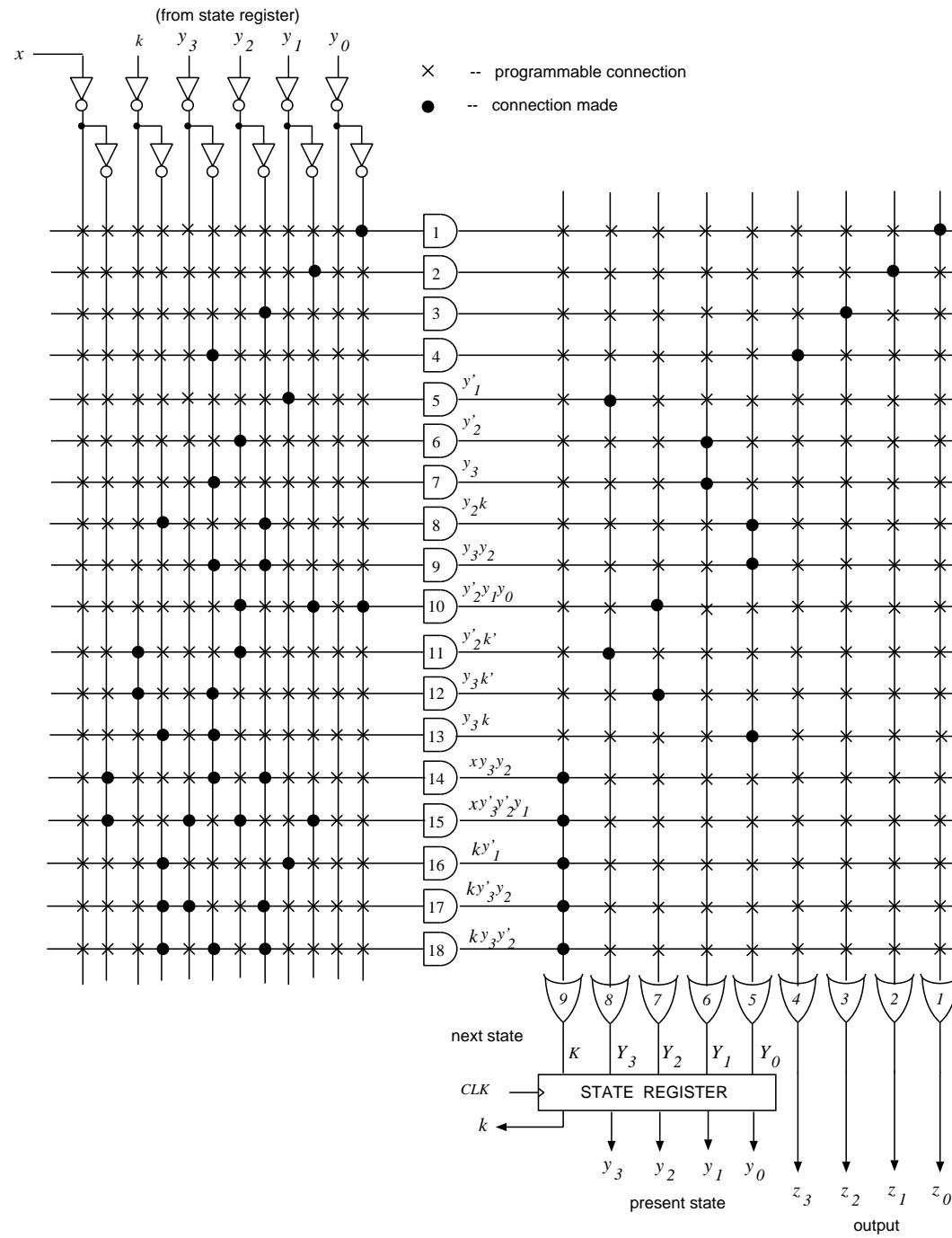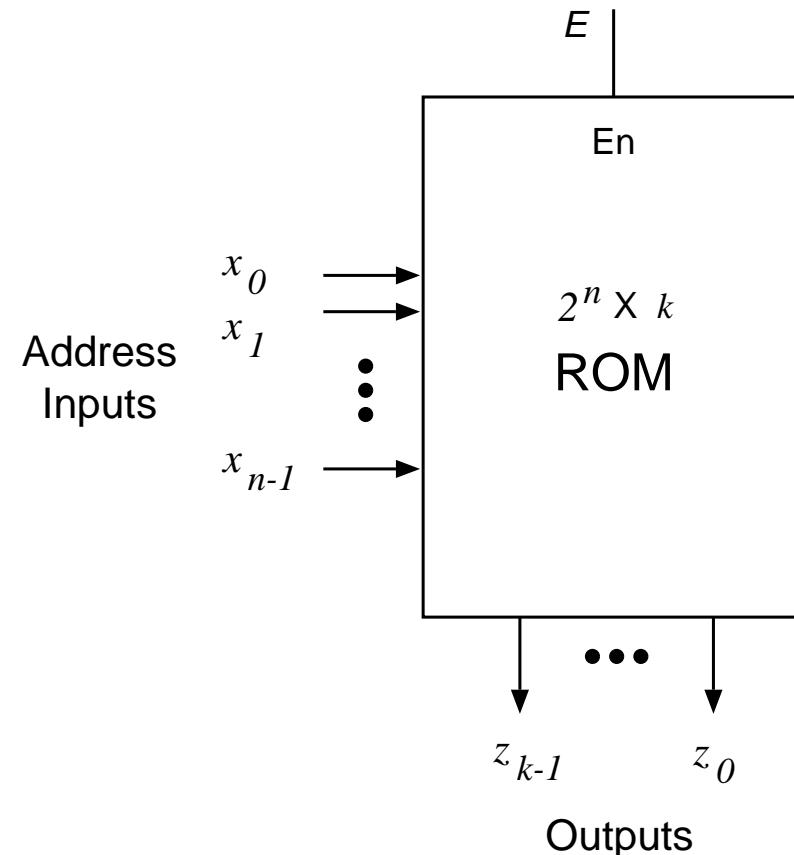
Figure 12.4: READ-ONLY MEMORY (ROM)

EXAMPLE 12.2

Address $\underline{x}$	Contents $\underline{z}$
000	1011
001	1101
010	**0111**
011	1000
100	0000
101	1111
110	1111
111	1011

E	x_1	x_0	z_3	z_2	z_1	z_0
1	0	0	0	1	0	1
1	0	1	1	1	0	0
1	1	0	1	0	1	1
1	1	1	0	0	1	0
0	-	-	Z	Z	Z	Z

(a)
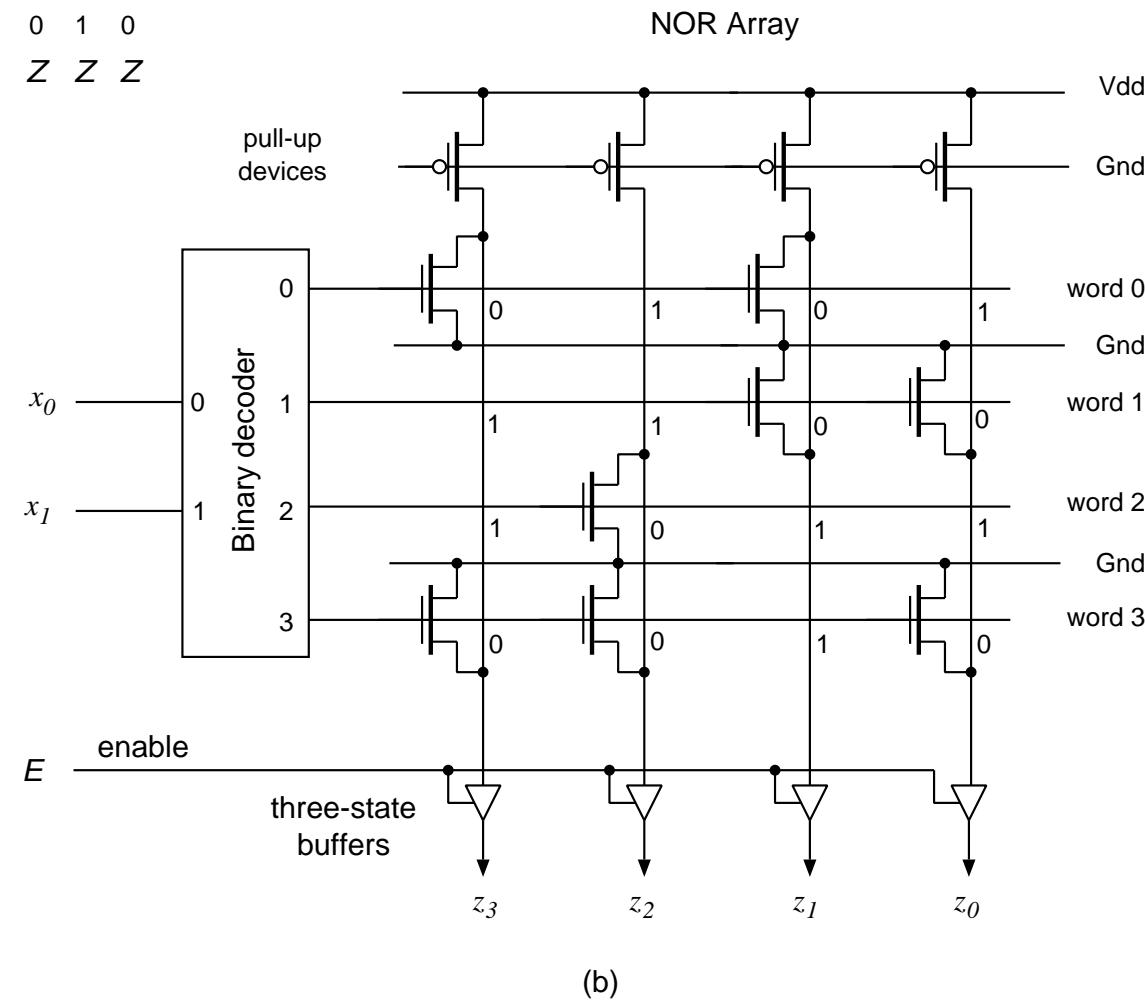
(b)

Figure 12.5: MOS IMPLEMENTATION OF A 4×4 READ-ONLY MEMORY: a) THE FUNCTION; b) THE CIRCUIT.

IMPLEMENTATION OF SWITCHING FUNCTIONS USING ROMs

ROM (512x5)

0	0 0 0 0 0
1	0 0 0 0 1

c_{in} — 0

x_0 — 1

BINARY DECODER

349 — 1 1 0 0 1

x_1

x_2

x_3

y_0

356 — 0 1 1 0 1

y_1

y_2 — 7

y_3 — 8

511 — 1 1 1 1 1

1
1110
+ 1010
—————
11001

Input/output
mapping:

$$c_{in}$$
$$x_3\ x_2\ x_1\ x_0$$
$$+\quad y_3\ y_2\ y_1\ y_0$$
$$\overline{c_{out}\ z_3\ z_2\ z_1\ z_0}$$

1
0
1
1
1
0
1
0
1

$z_3\ z_2\ z_1\ z_0$

c_{out}
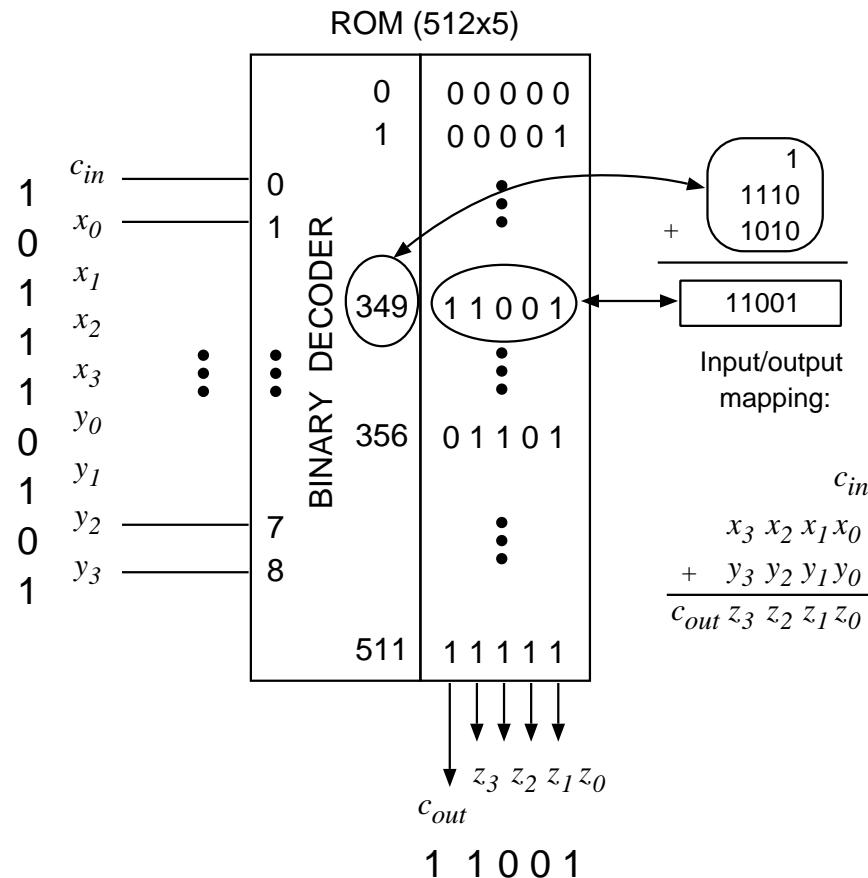
1 1 0 0 1

Figure 12.6: ROM-BASED IMPLEMENTATION OF A 4-BIT ADDER.

IMPLEMENTATION OF SEQUENTIAL SYSTEMS USING ROMs

INPUTS: $\quad \underline{x} = (x_1, x_0), \quad x_i \in \{0, 1\}$

OUTPUTS: $\quad z \in \{0, 1, \}$

STATE: $\quad \underline{y} = (y_1, y_0), \quad y_i \in \{0, 1, \}$

FUNCTION: The transition and output function

PS	$x_1 x_0$		
$y_1 y_0$	01	10	11
00	01,0	10,1	10,0
01	00,0	11,1	11,0
10	11,0	10,0	00,1
11	10,0	00,0	11,1
	$Y_1 Y_0, \ z$		
	NS, Output		

ROM address	ROM contents	
$y_1 y_0 x_1 x_0$	$Y_1 Y_0$	z
0000	- -	-
0001	0 1	0
0010	1 0	1
0011	1 0	0
0100	- -	-
0101	0 0	0
0110	1 1	1
0111	1 1	0
1000	- -	-
1001	1 1	0
1010	1 0	0
1011	0 0	1
1100	- -	-
1101	1 0	0
1110	0 0	0
1111	1 1	1

next state   output

(a)   (b)
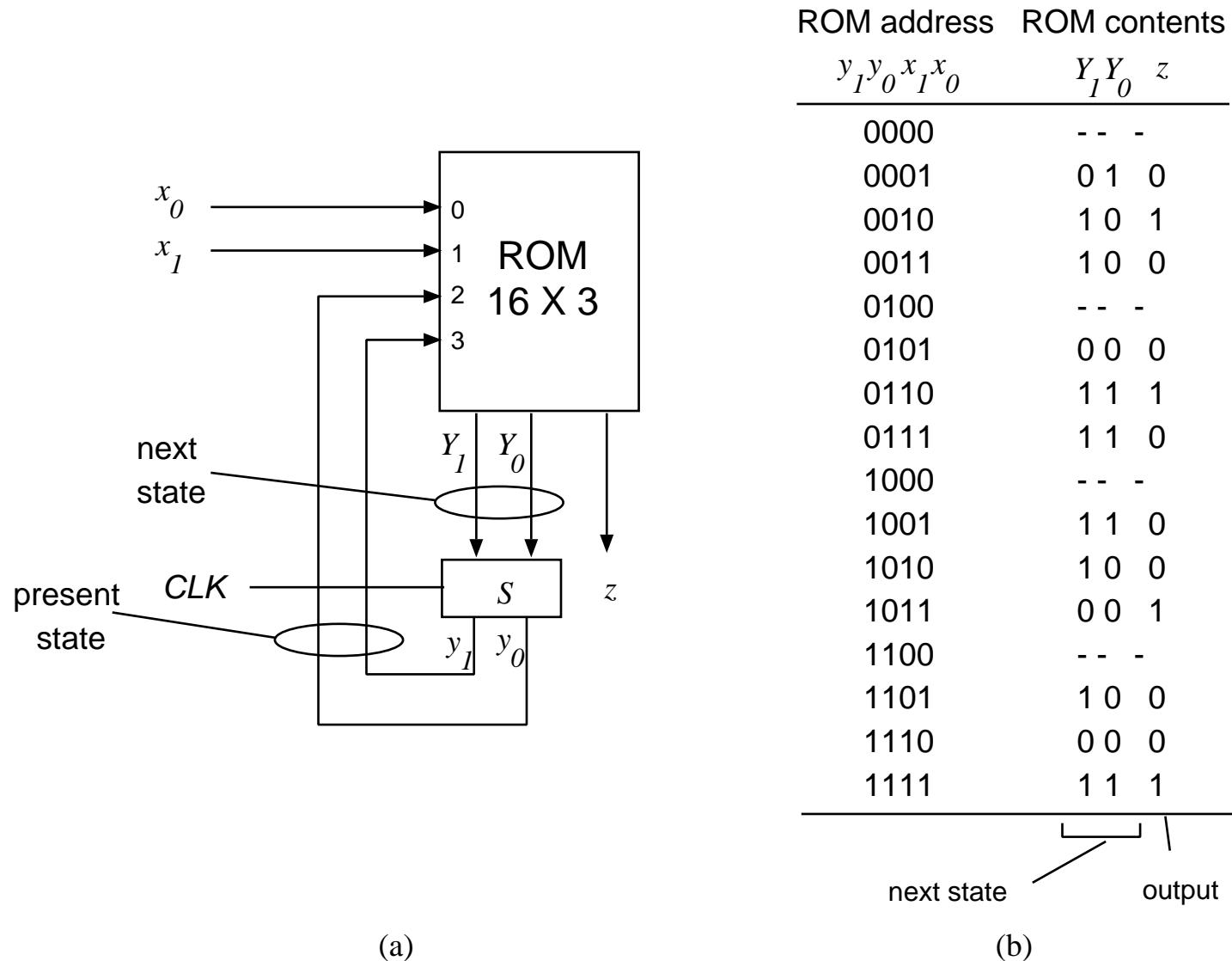
Figure 12.7: ROM-based implementation of a sequential system: a) network; b) ROM contents.

TYPES OF ROM MODULES

- MASK-PROGRAMMED ROM

- FIELD-PROGRAMMABLE ROM (PROMs)

- ERASABLE ROM (EPROM)

- ELECTRICALLY ERASABLE ROM(*flash-memory*) or EEPROM

NETWORKS OF PROGRAMMABLE MODULES

$$f_1(x_4, x_3, x_2, x_1, x_0) = \textit{one-set}(0,3,11,12,16,23,27)$$
$$f_0(x_4, x_3, x_2, x_1, x_0) = \textit{one-set}(5,7,19,21,31)$$

ROM MODULE: 8×2

$$\underline{x} = (\underline{x}^{(0)}, \underline{x}^{(1)})$$
$$\underline{x}^{(0)} = (x_4, x_3)$$
$$\underline{x}^{(1)} = (x_2, x_1, x_0)$$
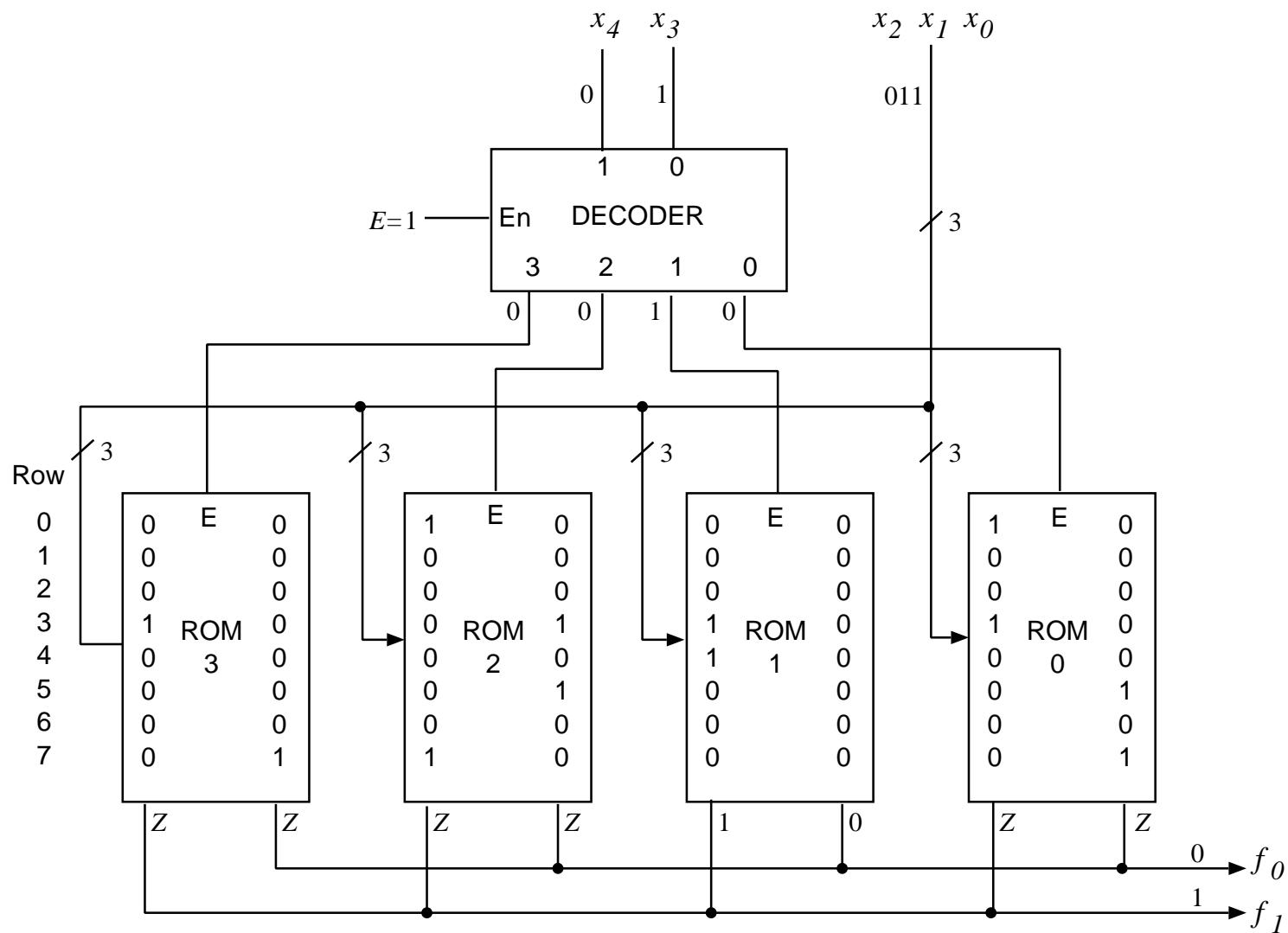
Figure 12.8: ROM-BASED NETWORK FOR THE IMPLEMENTATION OF TWO FUNCTIONS.

$x_{n-1}, \ldots, x_k$ $x_{k-1}, \ldots, x_0$

$N = 2^{n-k}$

E — En DECODER

$N-1$ $N-2$ 0

En En En

ROM
N-1

ROM
N-2

ROM
0

* three-state outputs

f

(a)

$N = 2^{n-k}$

1

k En k En k En

$x_{k-1}, \ldots, x_0$

ROM
N-1

ROM
N-2

ROM
0

k

$x_{n-1}, \ldots, x_k$ $n-k$ $N-1$ $N-2$ 0

E — En MULTIPLEXER

f
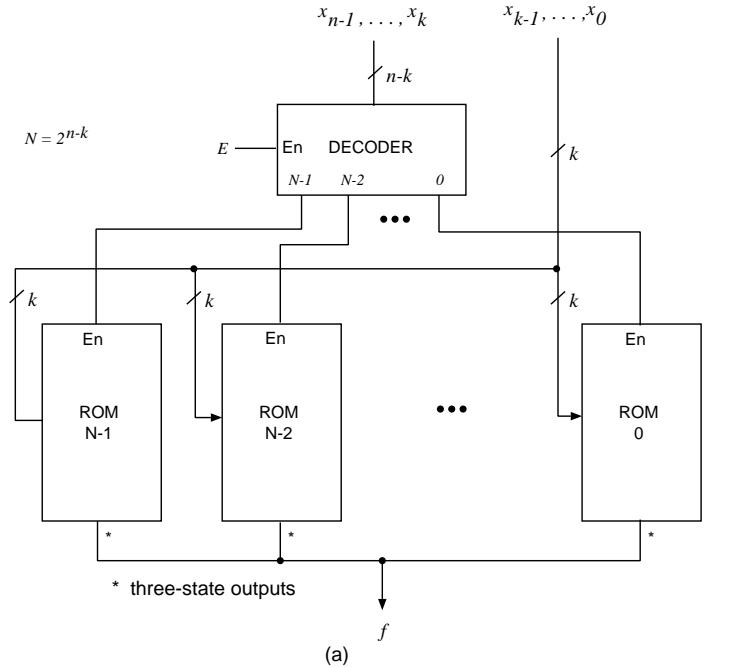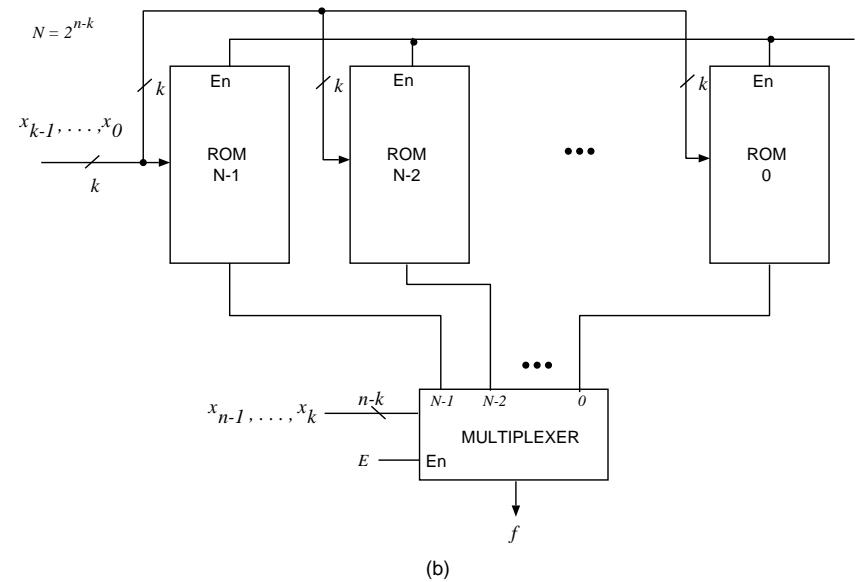
(b)

Figure 12.9: IMPLEMENTATIONS OF FUNCTIONS WITH n VARIABLES: a) ROMs AND DECODER; b) ROMs AND MULTIPLEXER
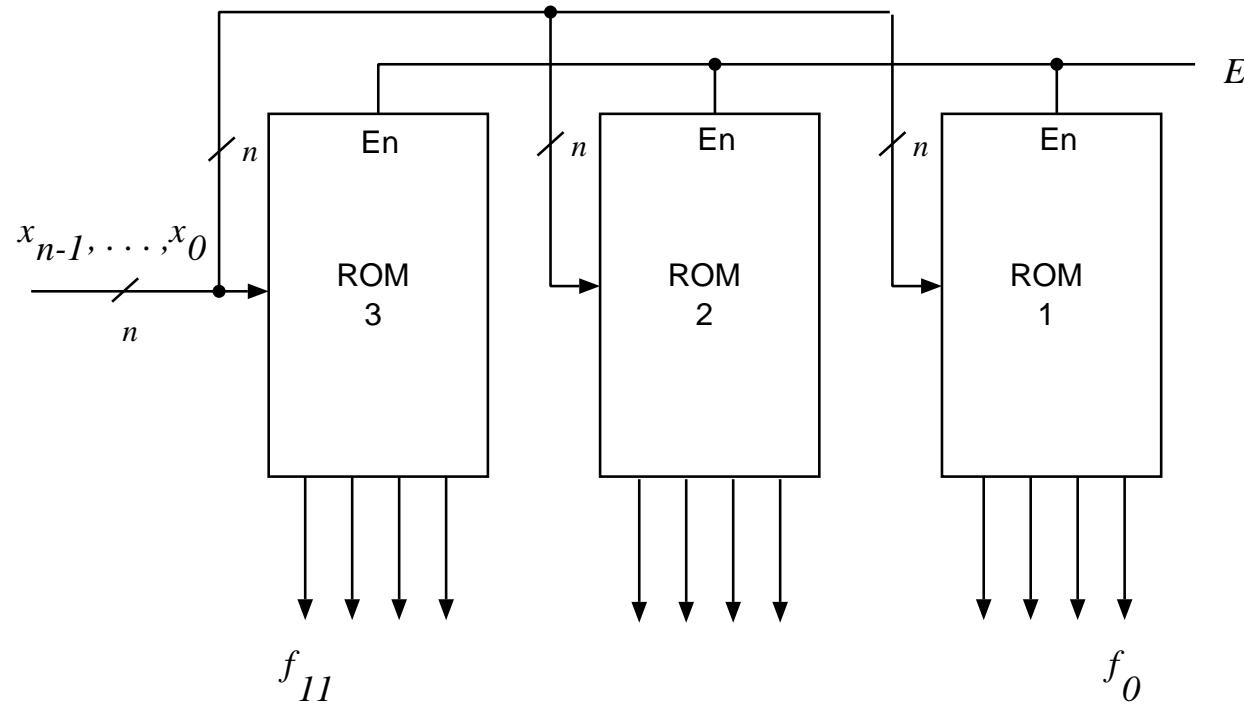
Figure 12.10: ROM-BASED IMPLEMENTATION OF LARGE NUMBER OF SWITCHING FUNCTIONS.

FIELD PROGRAMMABLE GATE ARRAYS (FPGA)
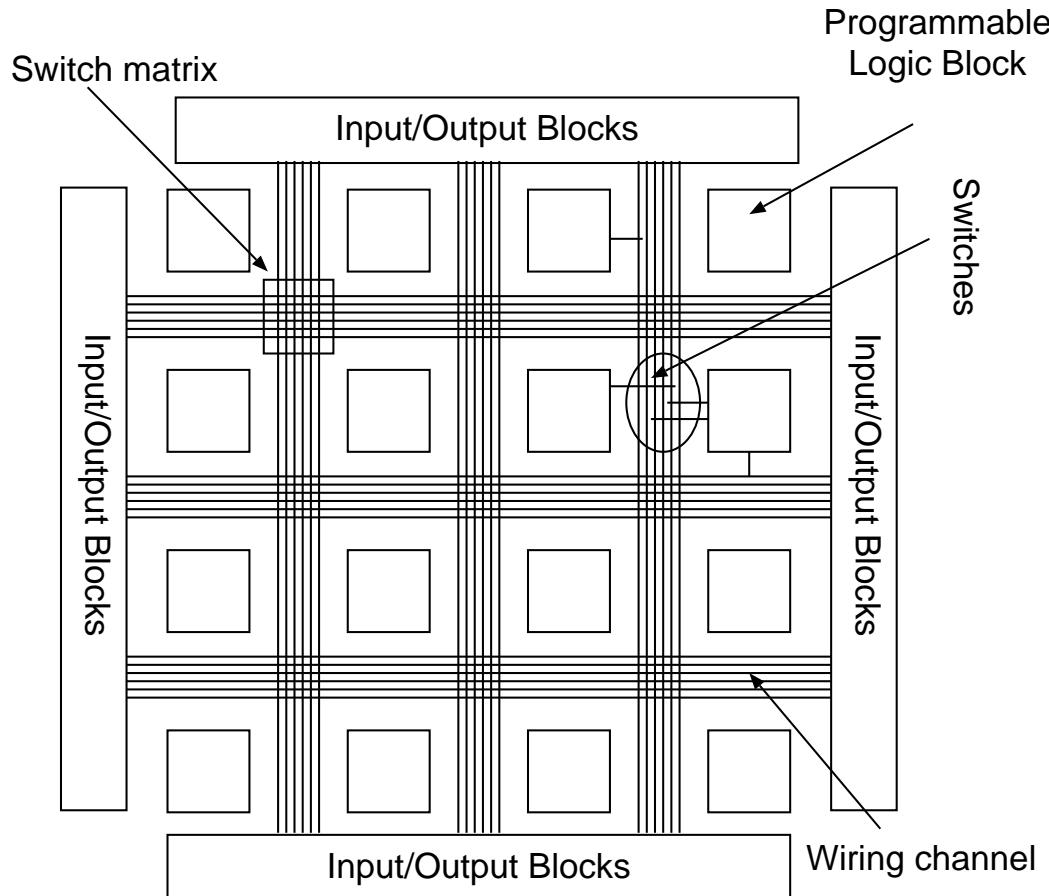
Figure 12.11: ORGANIZATION OF AN FPGA chip.

BASIC APPROACHES IN PROGRAMMING OF FPGAs

- ON-CHIP STATIC RAM LOADED WITH
  CONFIGURATION BIT PATTERNS (SRAM-FPGAs). (volatile)

- ANTIFUSE-PROGRAMMED DEVICES PROGRAMMED
  ELECTRICALLY TO PROVIDE CONNECTIONS THAT DEFINE CHIP CON-
  FIGURATION

- ARRAY-STYLE EPROM and EEPROM PROGRAMMED DEVICES
 USING SEVERAL PLAs AND A SHARED INTERCONNECT
  MECHANISM

SRAM cell

Transistor

1

Closed switch

0

Open switch

(a)

LUT (Look-Up Table)

a ——— 0
b ——— 1
c ——— 2   MUX ——→ y=d
d ——— 3
          1 0

1 1

SRAM cells

(b)

a ———
b ———   Decoder
c ———

0
1
1
0
1
0
0
1

——→ y=f(a,b,c)

SRAM cells

(c)
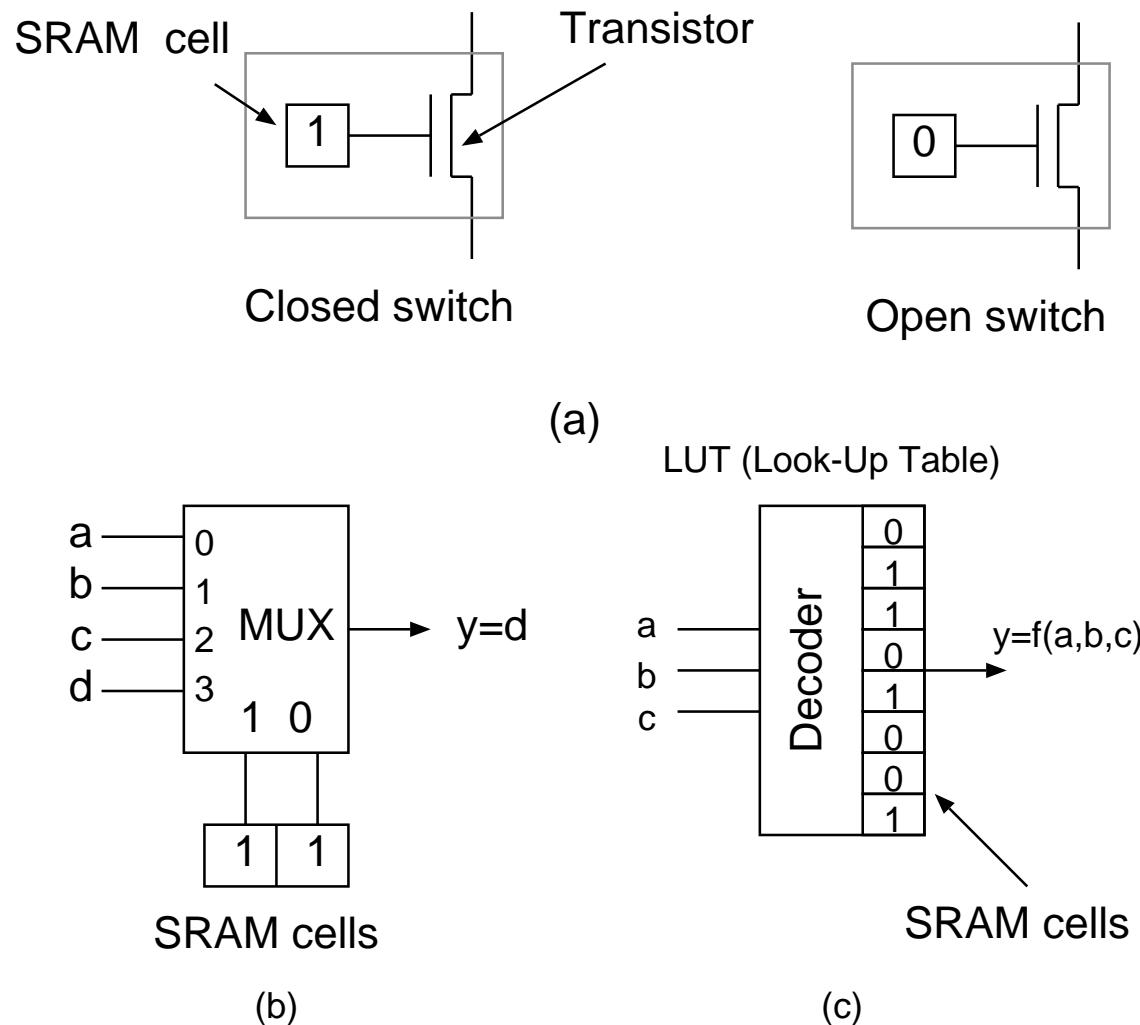
Figure 12.12: SRAM FPGA PROGRAMMABLE COMPONENTS: (a) Switch. (b) 4-input multiplexer. (c) Look-up table (LUT).

Example: XILINX XC2000
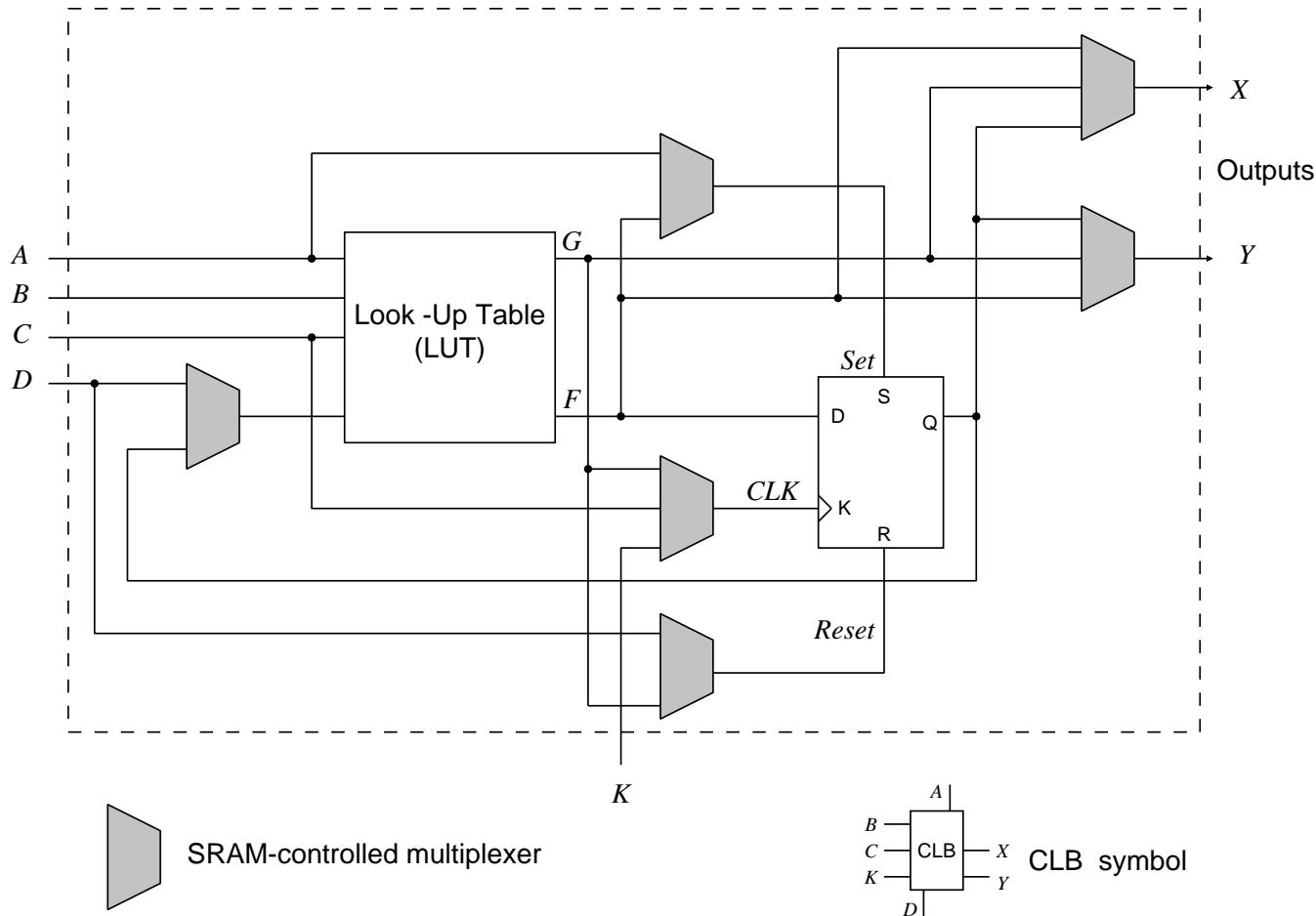
Figure 12.13: A CONFIGURABLE LOGIC BLOCK (CLB) (Courtesy of Xilinx, Inc.)
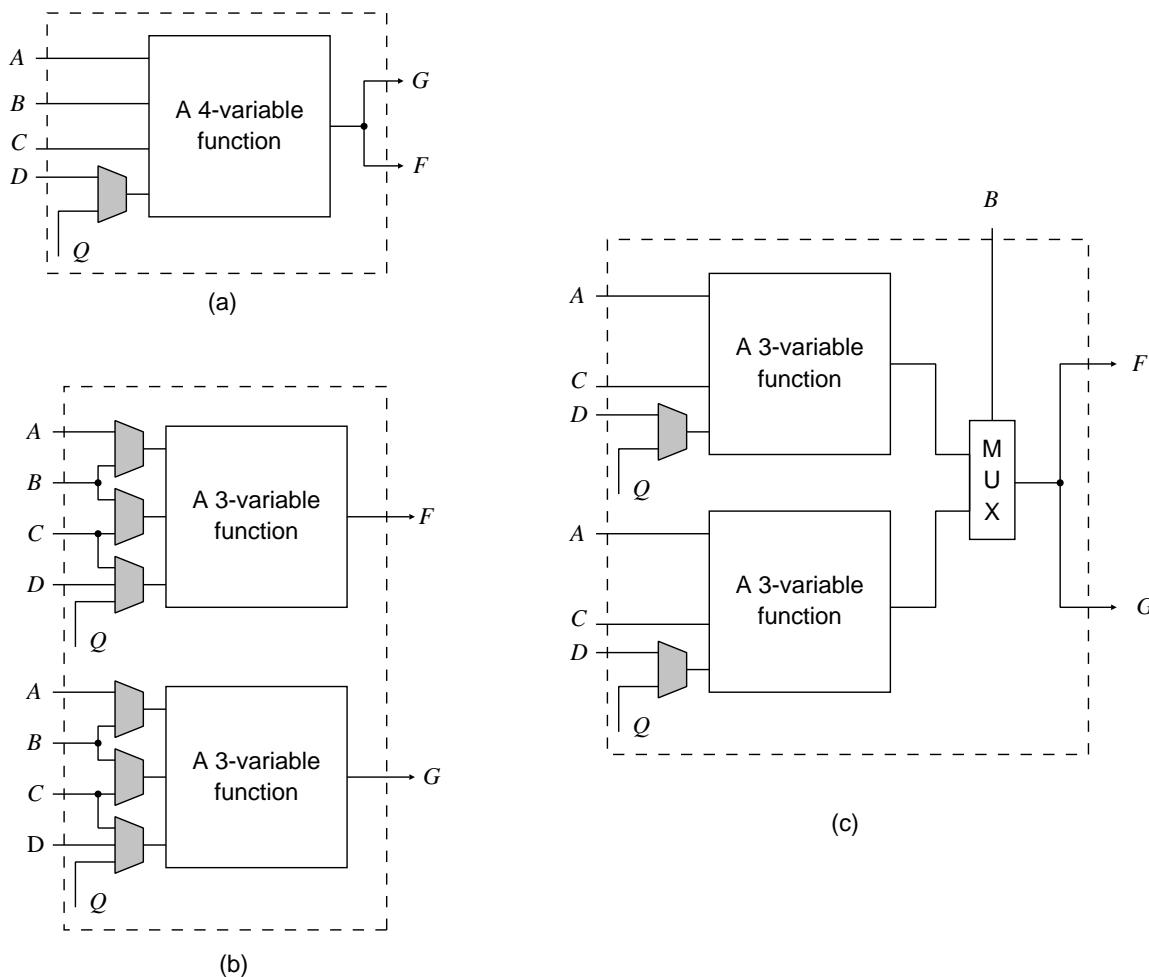
Figure 12.14: SRAM-FPGA options in generating functions: (a) One 4-variable function. (b) Two 3-variable functions. (c) Selection between two functions of 3 variables. (Courtesy of Xilinx, Inc.)

PROGRAMMABLE INTERCONNECT

1. DIRECT INTERCONNECTIONS BETWEEN HORIZONTALLY AND VERTICALLY ADJACENT CLBs– PROVIDE FAST SIGNAL PATHS BETWEEN ADJACENT MODULES

2. GENERAL-PURPOSE INTERCONNECT CONSISTS OF VERTICAL AND HORIZONTAL WIRING SEGMENTS BETWEEN SWITCH MATRICES

3. LONG VERTICAL AND HORIZONTAL LINES SPAN THE WHOLE CLBARRAY
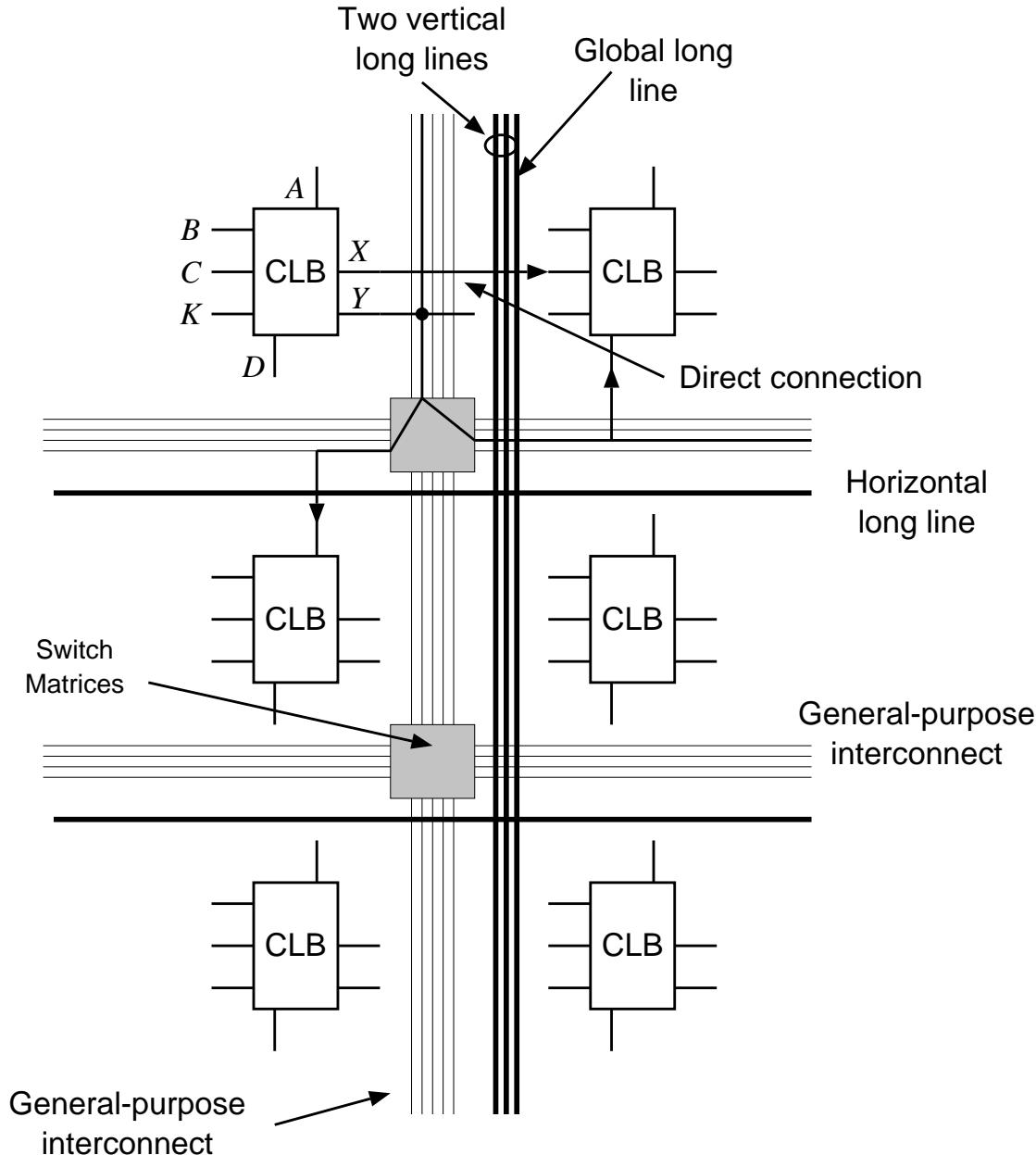
Figure 12.15: PROGRAMMABLE INTERCONNECT. (Courtesy of Xilinx, Inc.)

Example 12.5: BCD ADDER MODULE

- IMPLEMENT A ONE-DIGIT BCD ADDER USING A SRAM-FPGA MODULE OF XC2000 TYPE

  INPUTS:
  $$\underline{x} = (x_3, x_2, x_1, x_0), \quad x_j \in \{0, 1\}, \quad x \in \{0, \dots, 9\}$$
  $$\underline{y} = (y_3, y_2, y_1, y_0), \quad y_j \in \{0, 1\}, \quad y \in \{0, \dots, 9\}$$
  $$c_{in} \in \{0, 1\}$$

  OUTPUTS:
  $$\underline{s} = (s_3, s_2, s_1, s_0), \quad s_j \in \{0, 1\}, \quad s \in \{0, \dots, 9\}$$
  $$c_{out} \in \{0, 1\}$$

  FUNCTION: $x + y + c_{in} = 10c_{out} + s$

- COMPUTE $16u + v = x + y + c_{in} \in \{0, \dots, 19\}$ using a 4-bit binary adder

Example 12.5 (cont.)

- THREE CASES:

$$u = 0 \quad v \leq 9 \quad s = v \qquad\qquad\qquad\qquad c_{out} = 0$$
$$u = 0 \quad v > 9 \quad s = v - 10 = (v + 6) \ mod \ 16 \quad c_{out} = 1$$
$$u = 1 \qquad\qquad s = v + 16 - 10 = v + 6 \qquad c_{out} = 1$$

$\Rightarrow$ BCD OUTPUT

$$s = \begin{cases} (v + 6) mod16 & \textbf{if} \quad u = 1 \ or \ v \geq 10 \\ v & \textbf{otherwise} \end{cases}$$

$$c_{out} = \begin{cases} 1 & \textbf{if} \quad u = 1 \ or \ v \geq 10 \\ 0 & \textbf{otherwise} \end{cases}$$

THE CONDITION $u = 1 \ or \ v \geq 10$ CORRESPONDS TO SWITCHING EXPRESSION

$$t = u + v_3 v_2 + v_3 v_1$$

Example 12.5 (cont.)
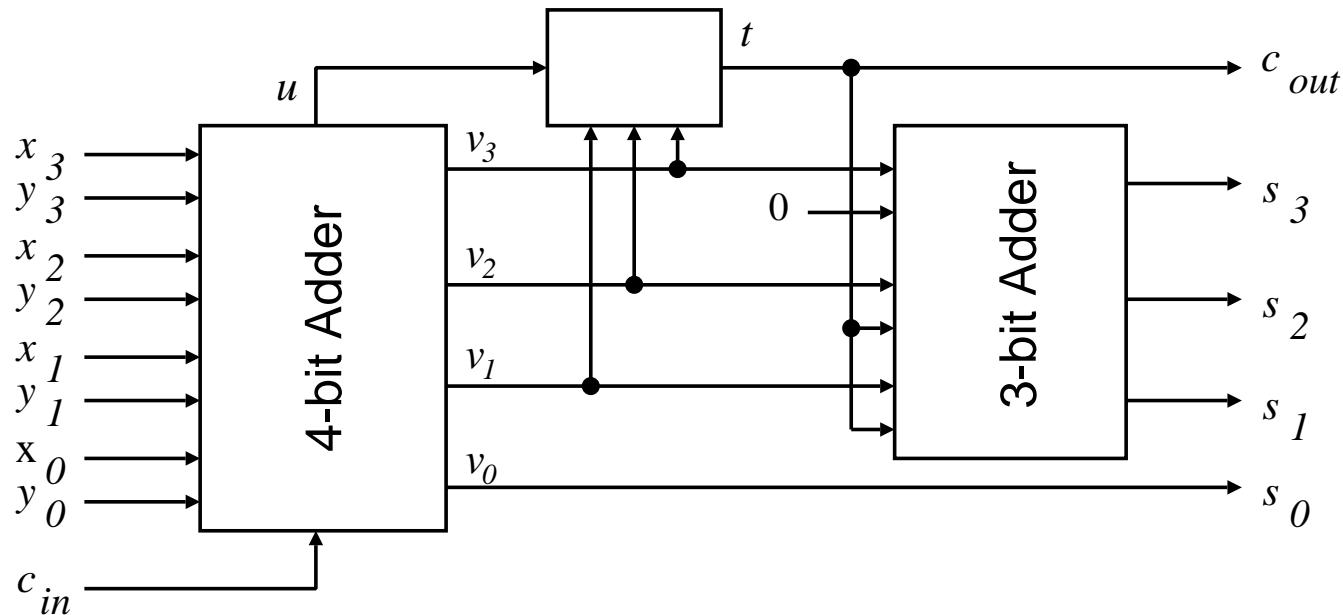
Figure 12.16: IMPLEMENTATION OF BCD ADDER MODULE

Example 12.5 (cont.)

- SIMPLIFICATION OF THE 3-BIT ADDER

$$s_3 = v_3 \oplus t(v_2 + v_1)$$
$$s_2 = v_2 \oplus tv_1'$$
$$s_1 = v_1 \oplus t$$

MOREOVER,

$$s_0 = v_0$$
$$c_{out} = t$$

x_3
y_3

u

CLB u

v_3 B A
v_2 C X
D Y t
K

v_3

c_{out}

v_1

x_2
y_2

t

CLB v_3 CLB s_3
v_2

v_2 v_1

x_1
y_1

t

CLB v_2 CLB s_2
v_1

v_1

x_0
y_0

t

CLB v_1 CLB s_1

c_{in} v_0

s_0
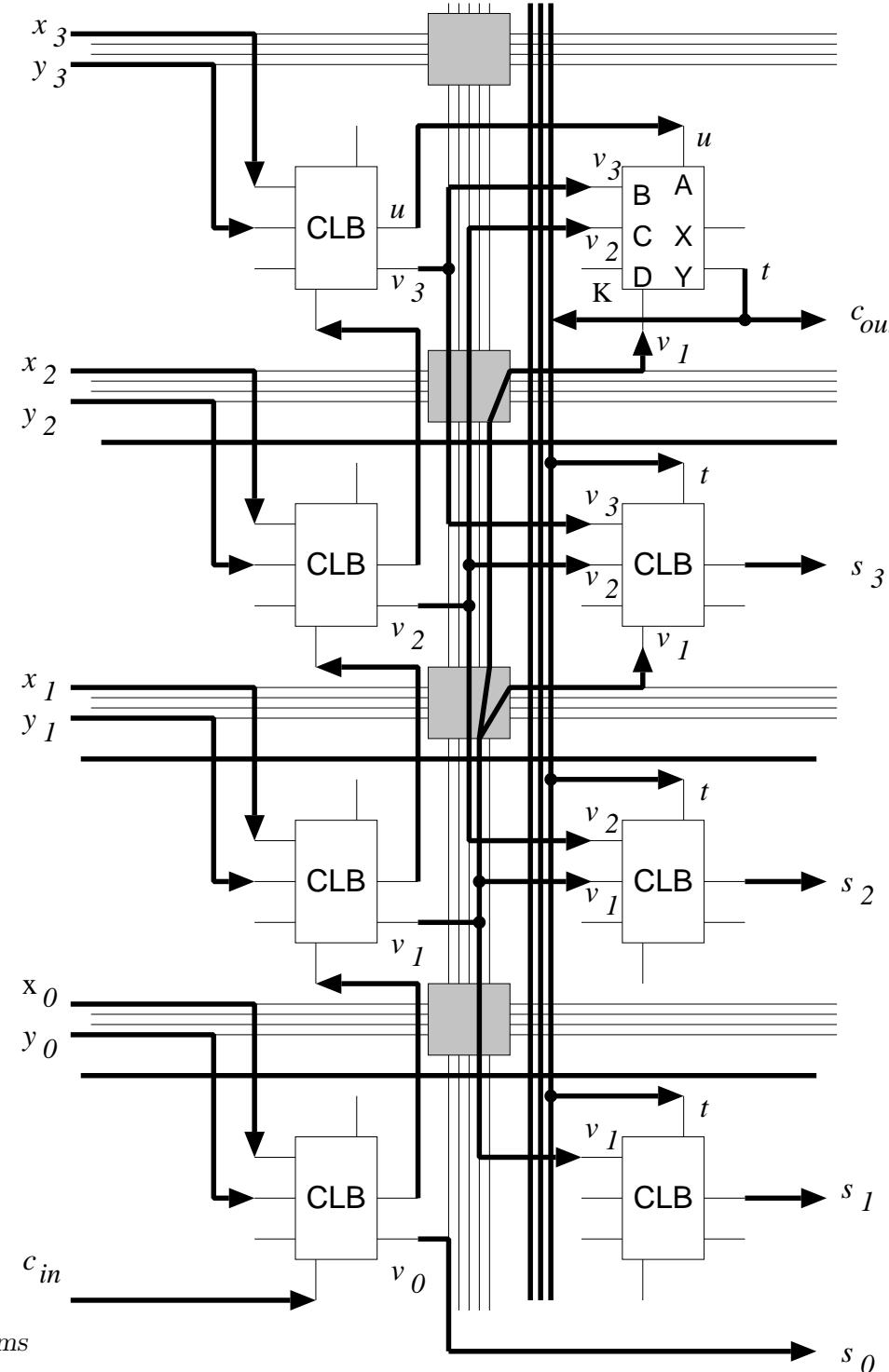
DESIGN WITH FPGAs

INVOLVES INTENSIVE USE OF CAD TOOLS AND MODULE LIBRARIES

Design entry : A SCHEMATIC ENTRY OR A BEHAVIORAL DESCRIPTION

Implementation :

- PARTITION OF DESIGN INTO SUBMODULES THAT CAN BE MAPPED ONTO CLBs,

- PLACEMENT OF SUBMODULES ONTO CHIP, AND

- ROUTING OF SIGNALS TO CONNECT THE SUBMODULES

Design verification :

- IN-CIRCUIT TESTING
- SIMULATION, AND
- TIMING ANALYSIS