CS M51A
Logic Design of Digital Systems
Winter 2021

Some slides borrowed and modified from:
M.D. Ercegovac, T. Lang and J. Moreno, Introduction to Digital Systems.

Standard Combinational Modules

- Decoder
- Encoder
- Multiplexer
- Demultiplexer
- Adder

TWO'S COMPLEMENT ARITHMETIC UNIT

INPUTS:

$\underline{x} = (x_{n-1}, \ldots, x_0), \quad x_j \in \{0, 1\}$
$\underline{y} = (y_{n-1}, \ldots, y_0), \quad y_j \in \{0, 1\}$
$c_{\text{in}} \in \{0, 1\}$
$F = (f_2, f_1, f_0)$

OUTPUTS:

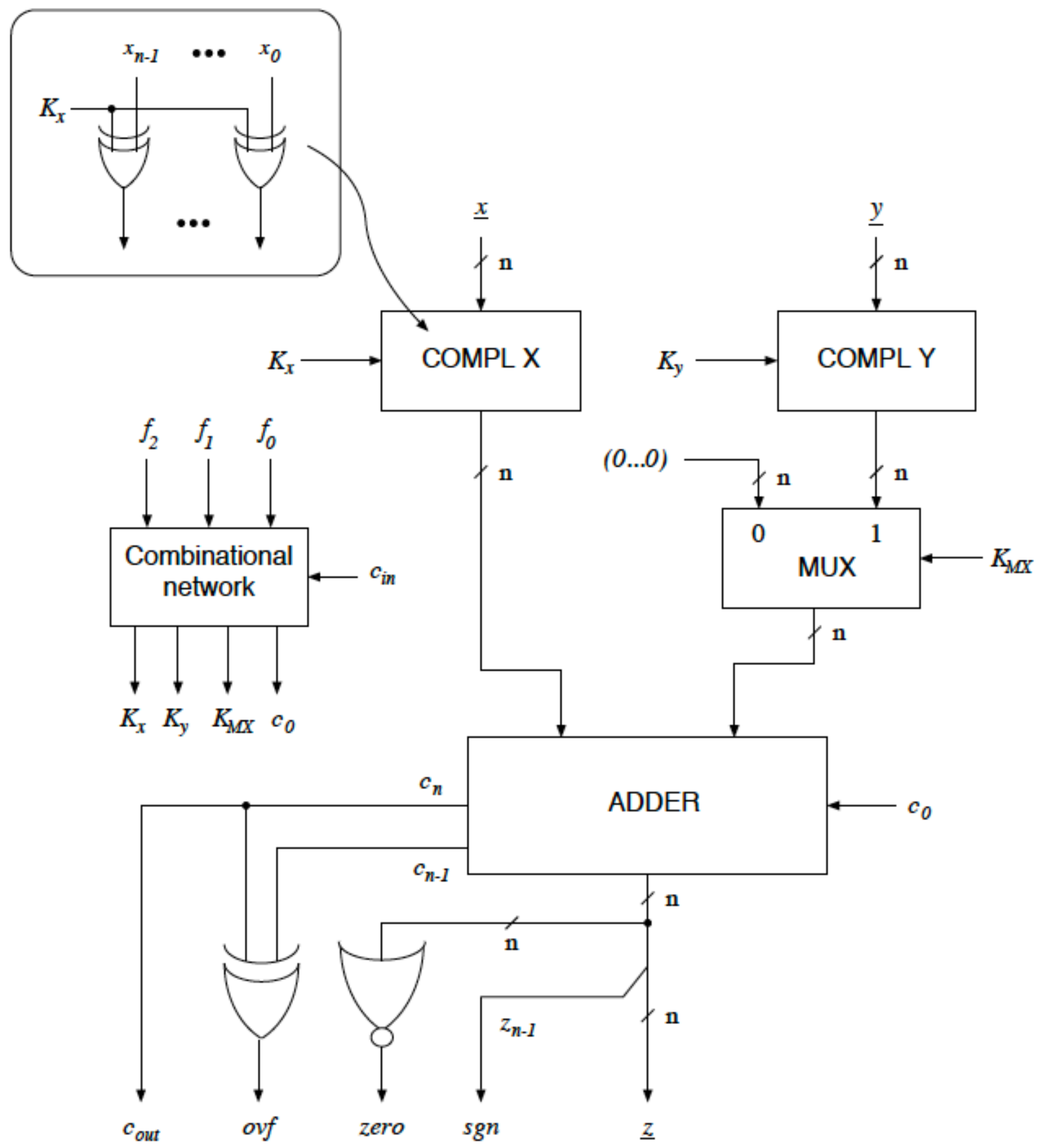
$\underline{z} = (z_{n-1}, \ldots, z_0), \quad z_j \in \{0, 1\}$
$c_{\text{out}}, sgn, zero, ovf \in \{0, 1\}$

FUNCTIONS:

F	Operation		
001	ADD	add	$z = x + y$
011	SUB	subtract	$z = x - y$
101	ADDC	add with carry	$z = x + y + c_{in}$
110	CS	change sign	$z = -x$
010	INC	increment	$z = x + 1$

$sgn = 1$ **if** $z < 0$, 0 **otherwise** (the sign)
$zero = 1$ **if** $z = 0$, 0 **otherwise**
$ovf = 1$ **if** z overflows, 0 **otherwise**

TWO'S COMPLEMENT ARITHMETIC UNIT



CONTROL OF TWO'S-COMPLEMENT ARITHMETIC OPERATIONS

- OPERATION IDENTIFIED BY BIT-VECTOR $F = (f_2, f_1, f_0)$

Operation	Op-code		Control Signals		
	$f_2 f_1 f_0$	$\underline{z}$	K_x	K_y	K_{MX}
ADD	001	$\mathrm{ADD}(\underline{x}, \underline{y}, 0)$	0	0	1
SUB	011	$\mathrm{ADD}(\underline{x}, \underline{y'}, 1)$	0	1	1
ADDC	101	$\mathrm{ADD}(\underline{x}, \underline{y}, c_{\mathrm{in}})$	0	0	1
CS	110	$\mathrm{ADD}(\underline{x'}, \underline{0}, 1)$	1	d.c.	0
INC	010	$\mathrm{ADD}(\underline{x}, \underline{0}, 1)$	0	d.c.	0

- CONTROL SIGNALS:

$$\begin{aligned} K_x &= f_2 f_1 \\ K_y &= f_1 \\ K_{MX} &= f_0 \end{aligned}$$

ALU MODULES AND NETWORKS

- *ARITHMETIC-LOGIC UNIT*

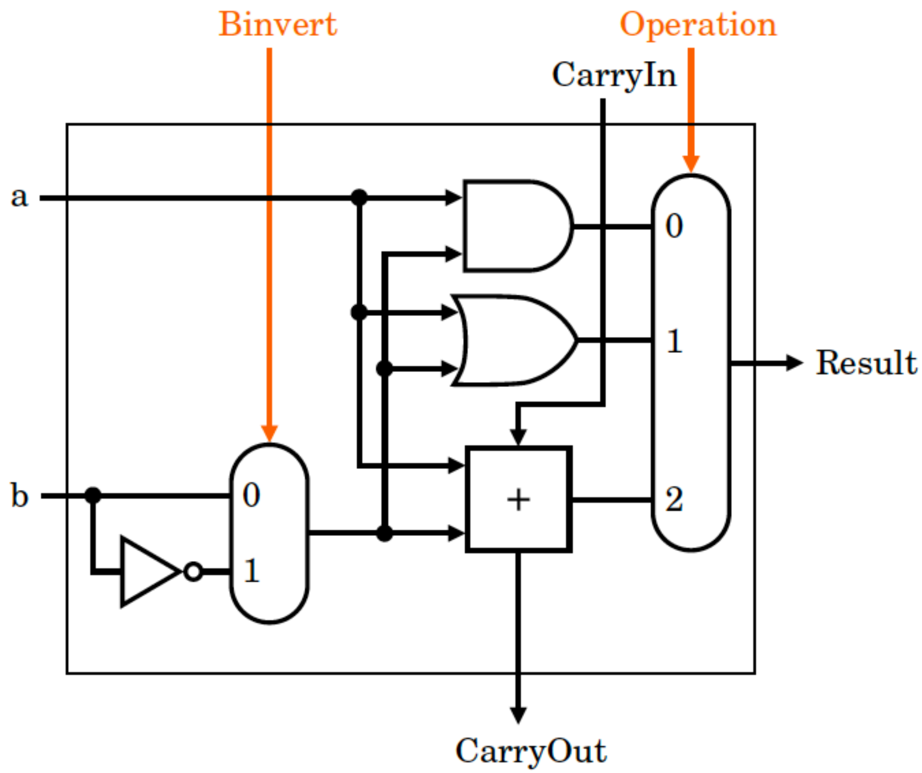
  module realizing set of arithmetic and logic functions

TYPICAL EXAMPLE OF ALU

Control (S)	Function
ZERO	$z = 0$
ADD	$z = (x + y + c_{\text{in}})$
SUB	$z = (x + y' + c_{\text{in}})$
EXSUB	$z = (x' + y + c_{\text{in}})$
AND	$\underline{z} = \underline{x} \cdot \underline{y}$
OR	$\underline{z} = \underline{x} + \underline{y}$
XOR	$\underline{z} = \underline{x} \oplus \underline{y}$
ONE	$\underline{z} = 1111$

Clicker Question

ALU



For $a = 0$, $b = 0$, CarryIn=1, Operation=01, Binvert=1,

a CarryOut=0,Result=0

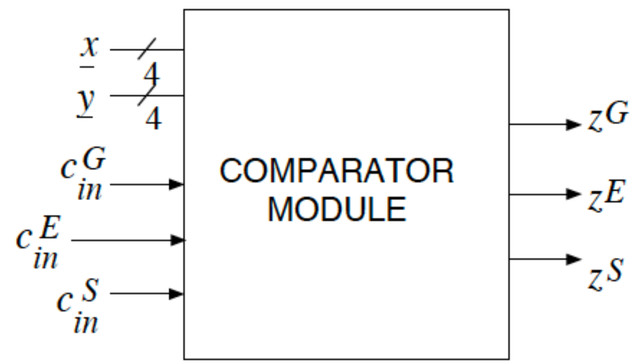
b CarryOut=0,Result=1

c CarryOut=1,Result=0

d CarryOut=1,Result=1

e None of the above

COMPARATOR MODULES



- HIGH-LEVEL DESCRIPTION OF AN n-BIT COMPARATOR:

INPUTS: $\underline{x} = (x_{n-1}, \ldots, x_0), \quad x_j \in \{0,1\}$

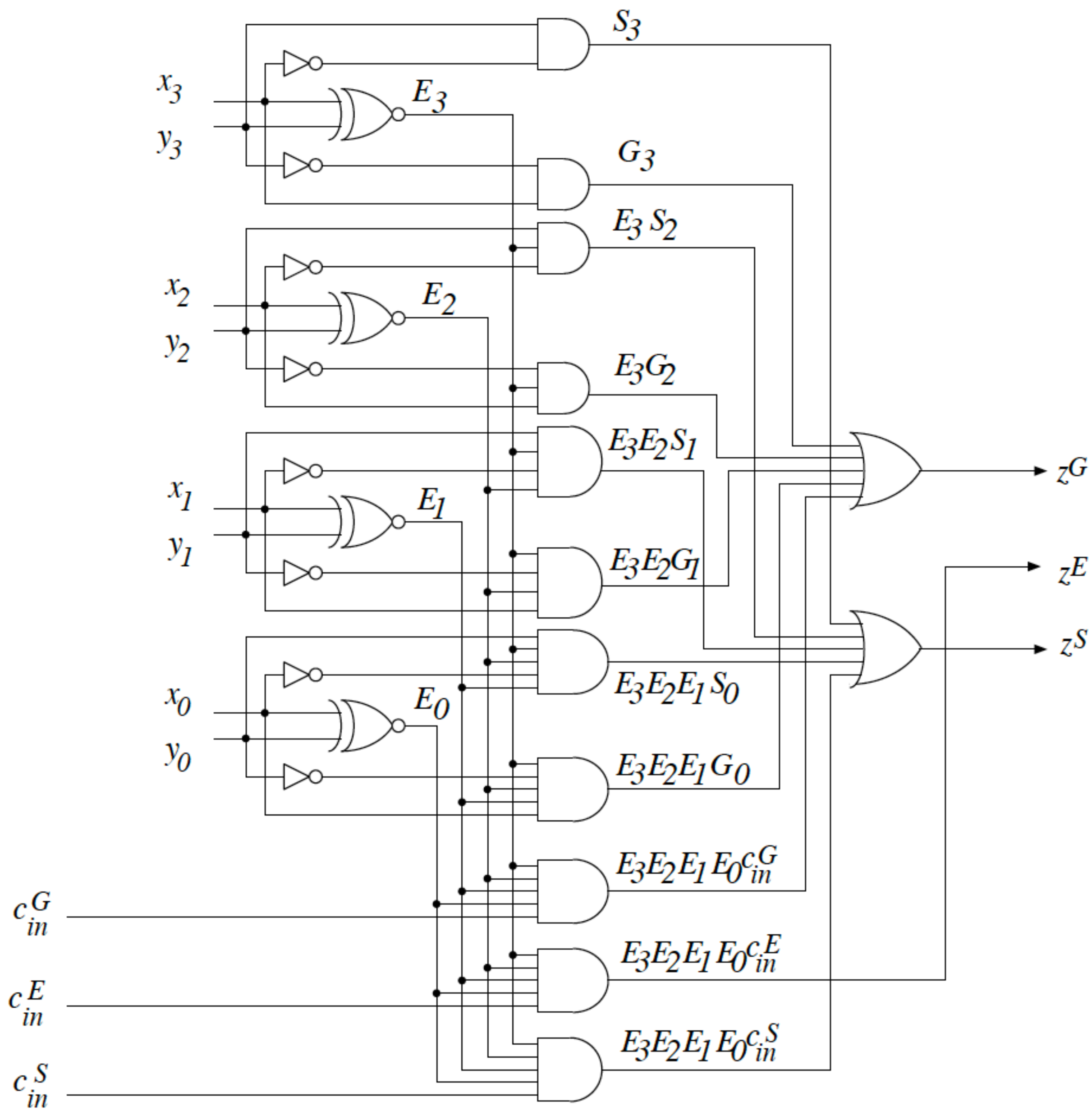
$\underline{y} = (y_{n-1}, \ldots, y_0), \quad y_j \in \{0,1\}$

$c_{\text{in}} \in \{\text{G,E,S}\}$

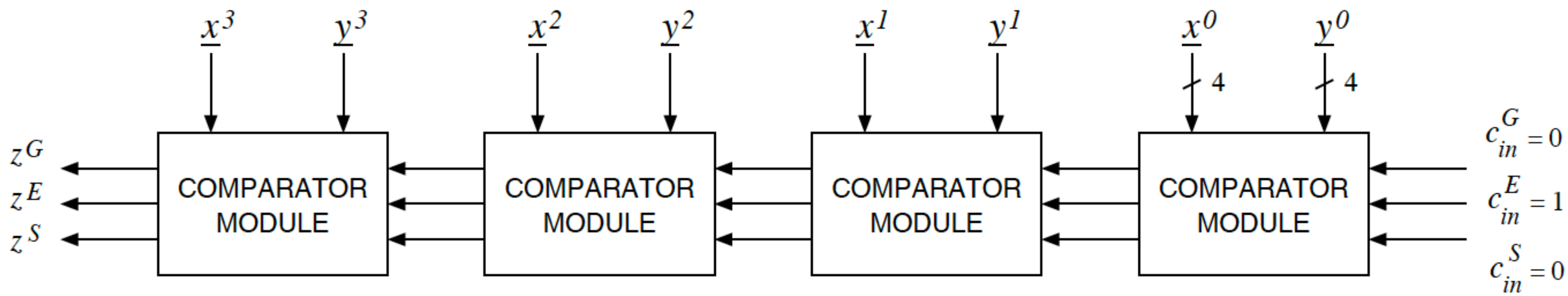
OUTPUT: $z \in \{\text{G,E,S}\}$

FUNCTION: $$z = \begin{cases} \text{G} & \textbf{if} \quad (x > y) \ \textbf{or} \ (x = y \ \textbf{and} \ c_{\text{in}} = \text{G}) \\ \text{E} & \textbf{if} \quad (x = y) \ \textbf{and} \ (c_{\text{in}} = \text{E}) \\ \text{S} & \textbf{if} \quad (x < y) \ \textbf{or} \ (x = y \ \textbf{and} \ c_{\text{in}} = \text{S}) \end{cases}$$

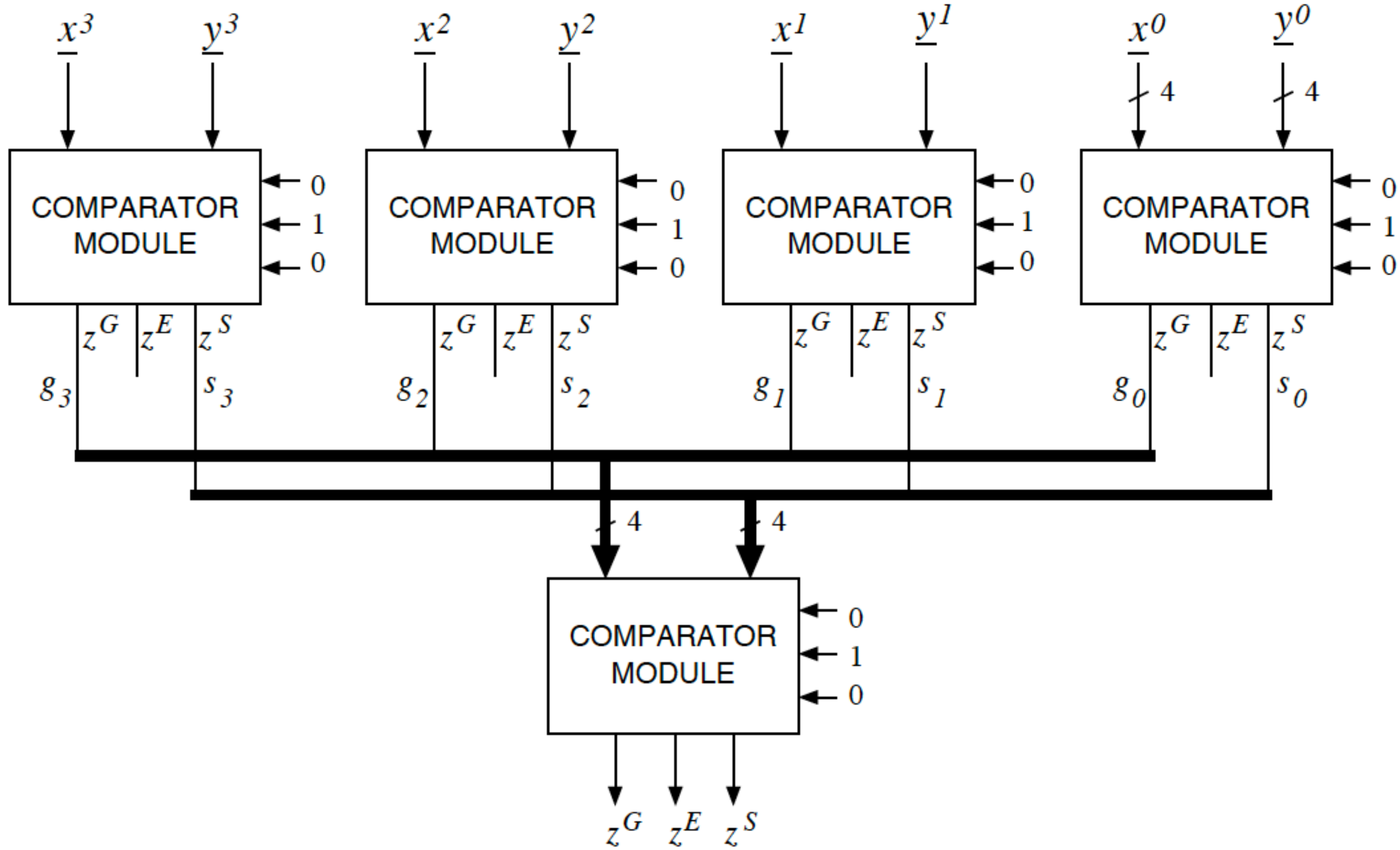
Implementation



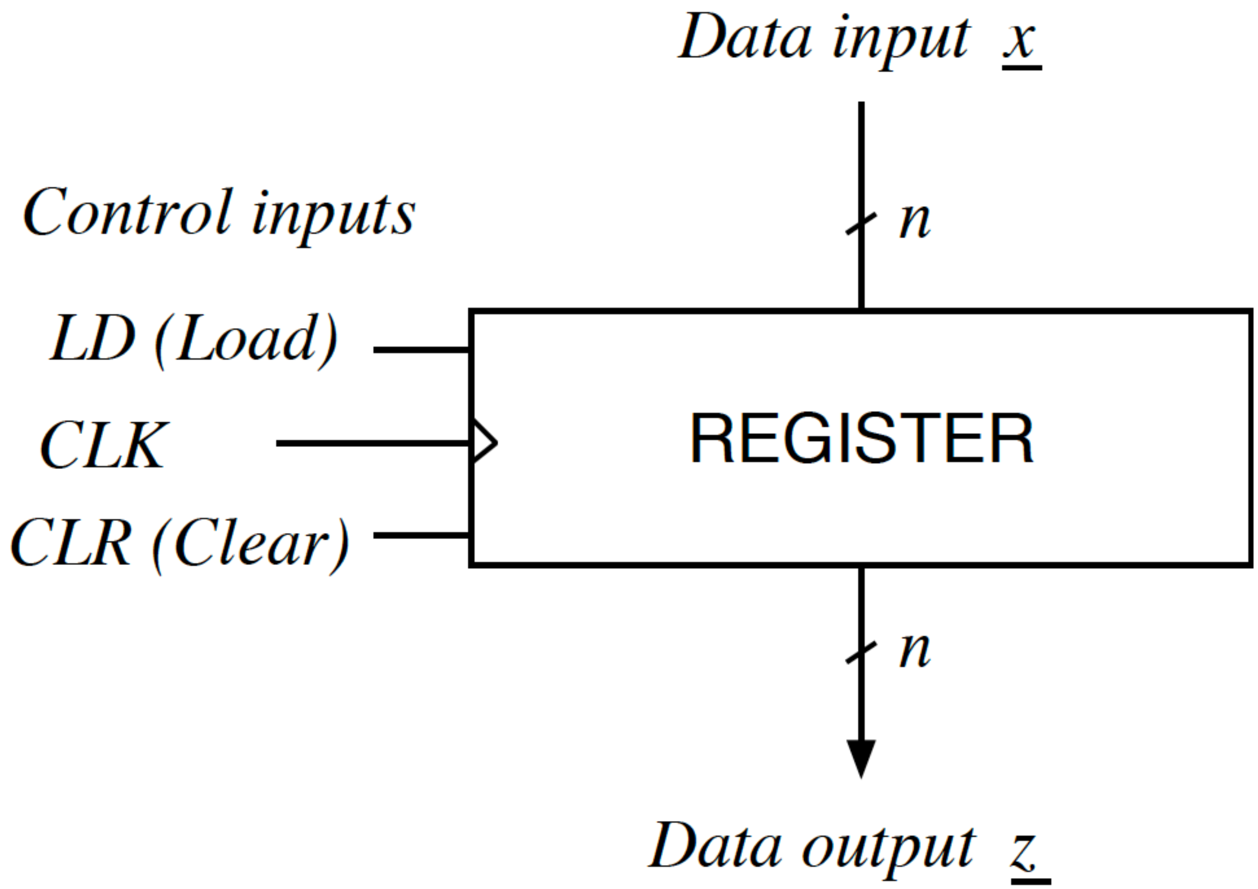
ITERATIVE COMPARATOR NETWORK



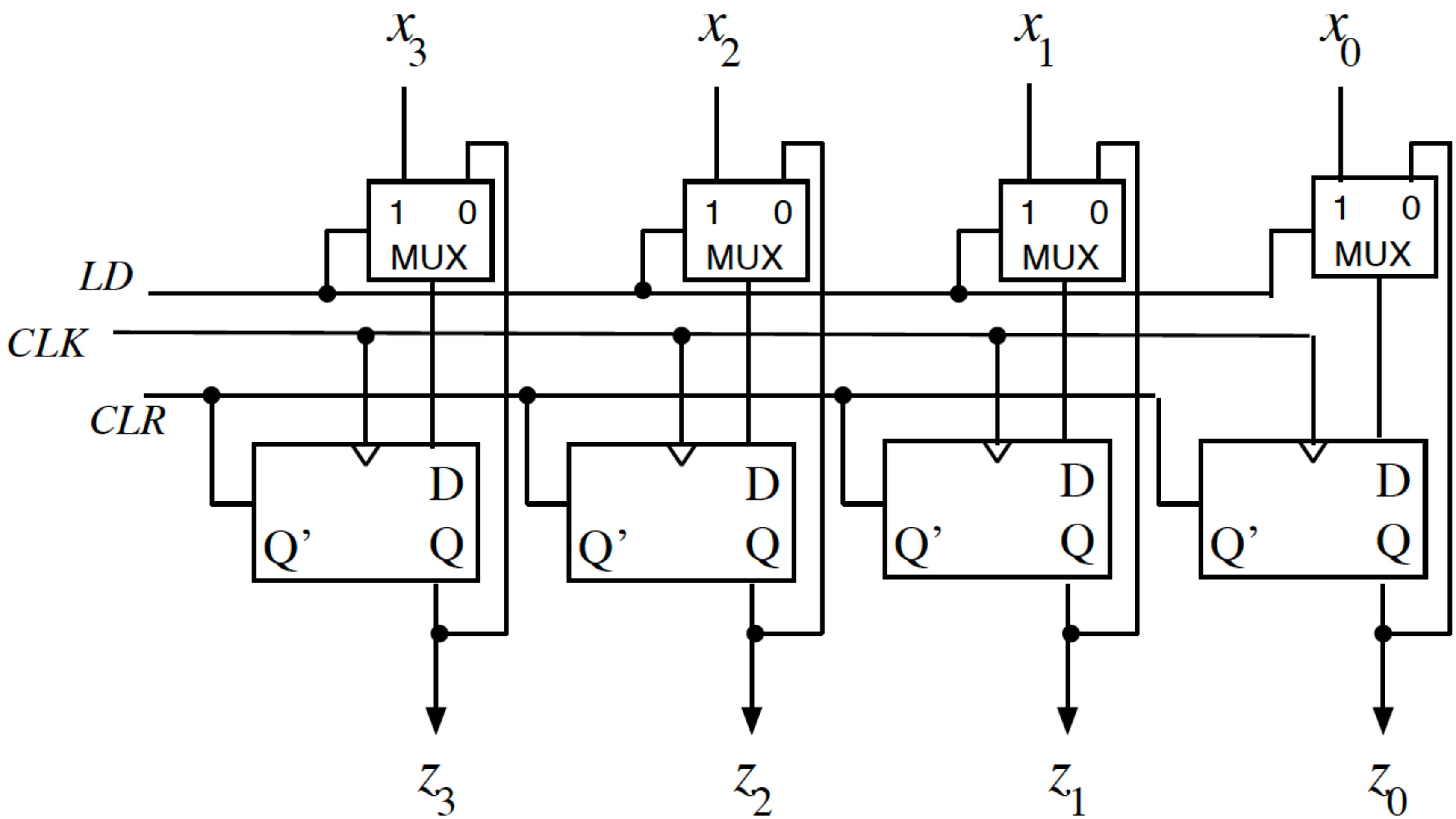
TREE COMPARATOR NETWORK



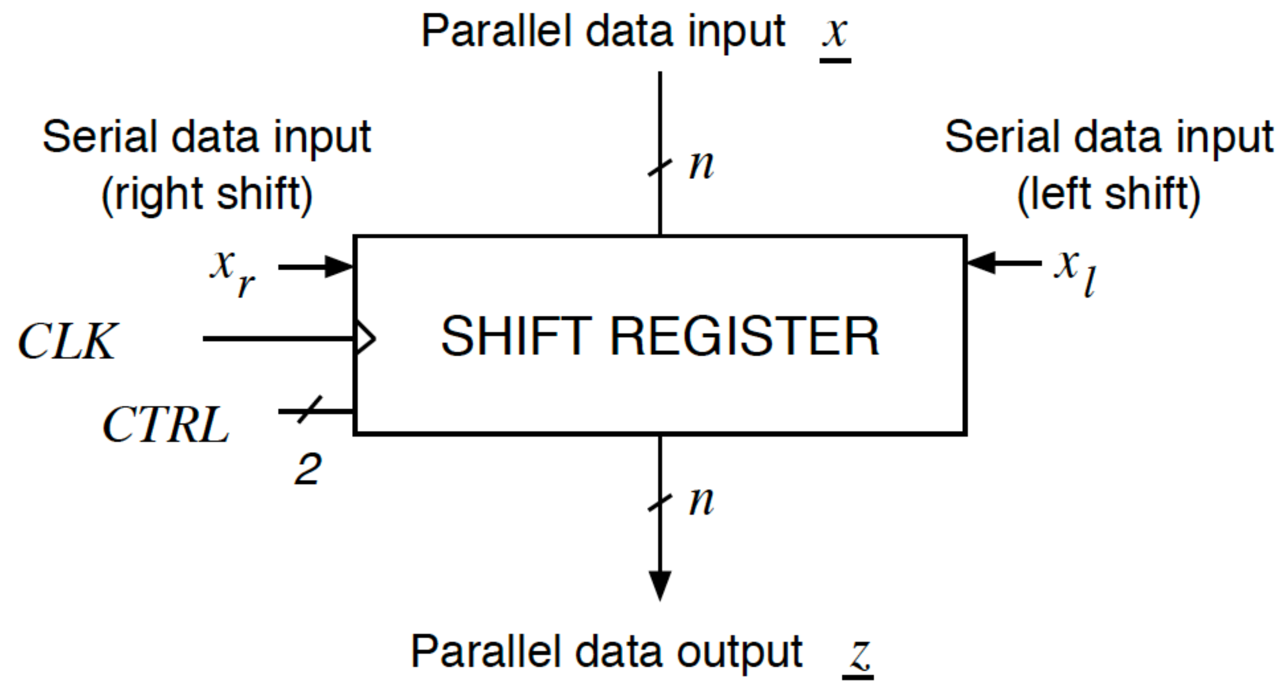
Standard Sequential Modules


Data input x
n
Control inputs
LD (Load)
CLK
CLR (Clear)
REGISTER
n
Data output z

IMPLEMENTATION OF 4-BIT REGISTER



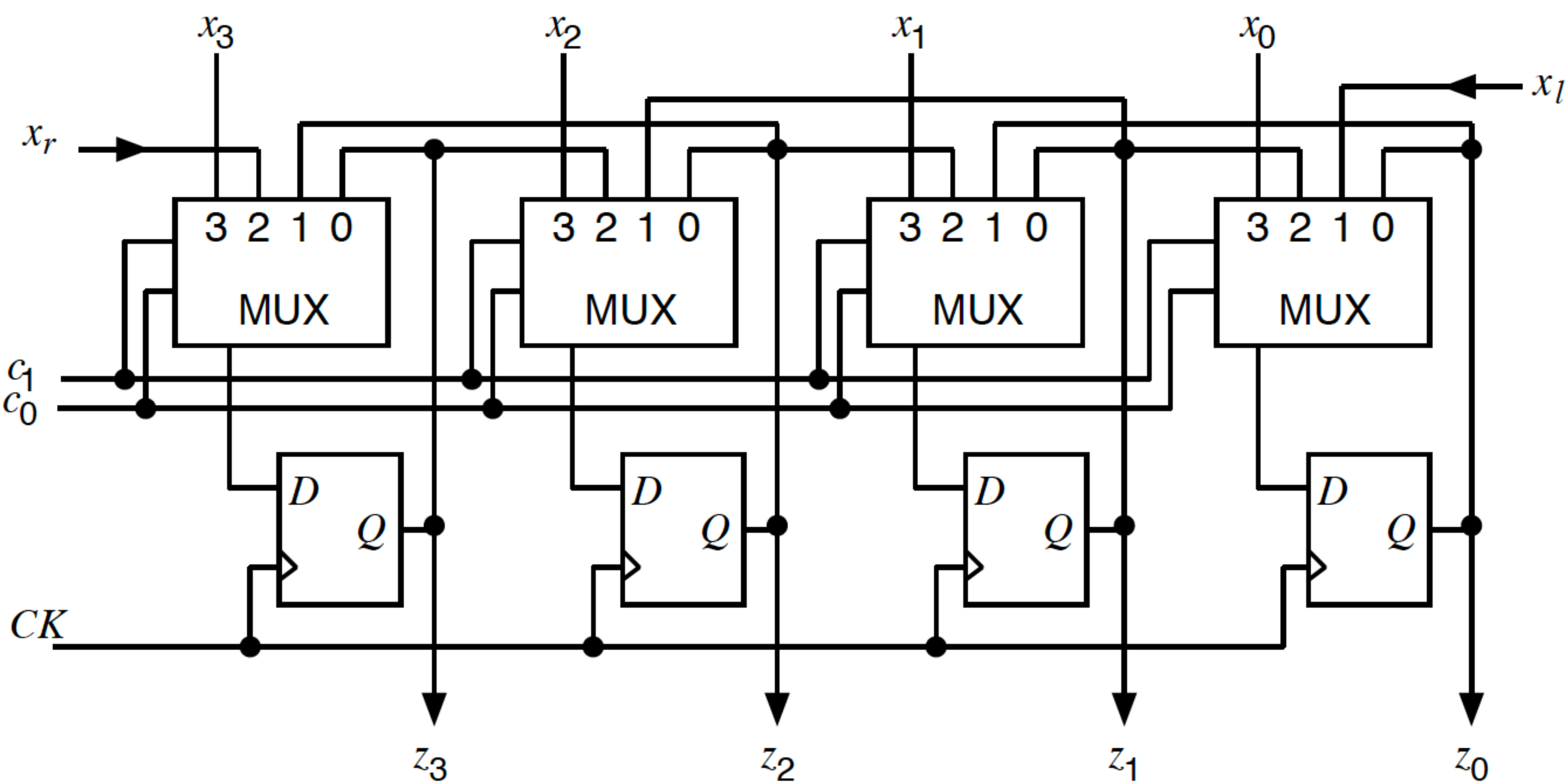
SHIFT REGISTERS



$$\underline{s}(t+1) = \begin{cases} \underline{s}(t) & \textbf{if} \quad CTRL = NONE \\ \underline{x}(t) & \textbf{if} \quad CTRL = LOAD \\ (s_{n-2}, \ldots, s_0, x_l) & \textbf{if} \quad CTRL = LEFT \\ (x_r, s_{n-1}, \ldots, s_1) & \textbf{if} \quad CTRL = RIGHT \end{cases}$$

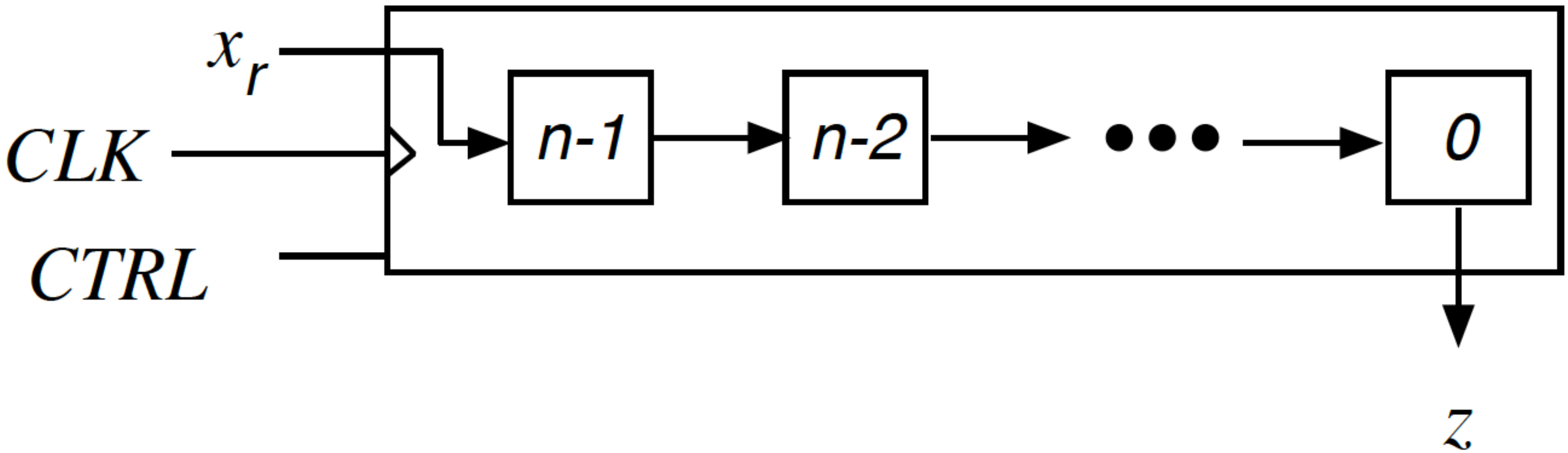
$$\underline{z} = \underline{s}$$

4-BIT BIDIRECTIONAL SHIFT-REGISTER

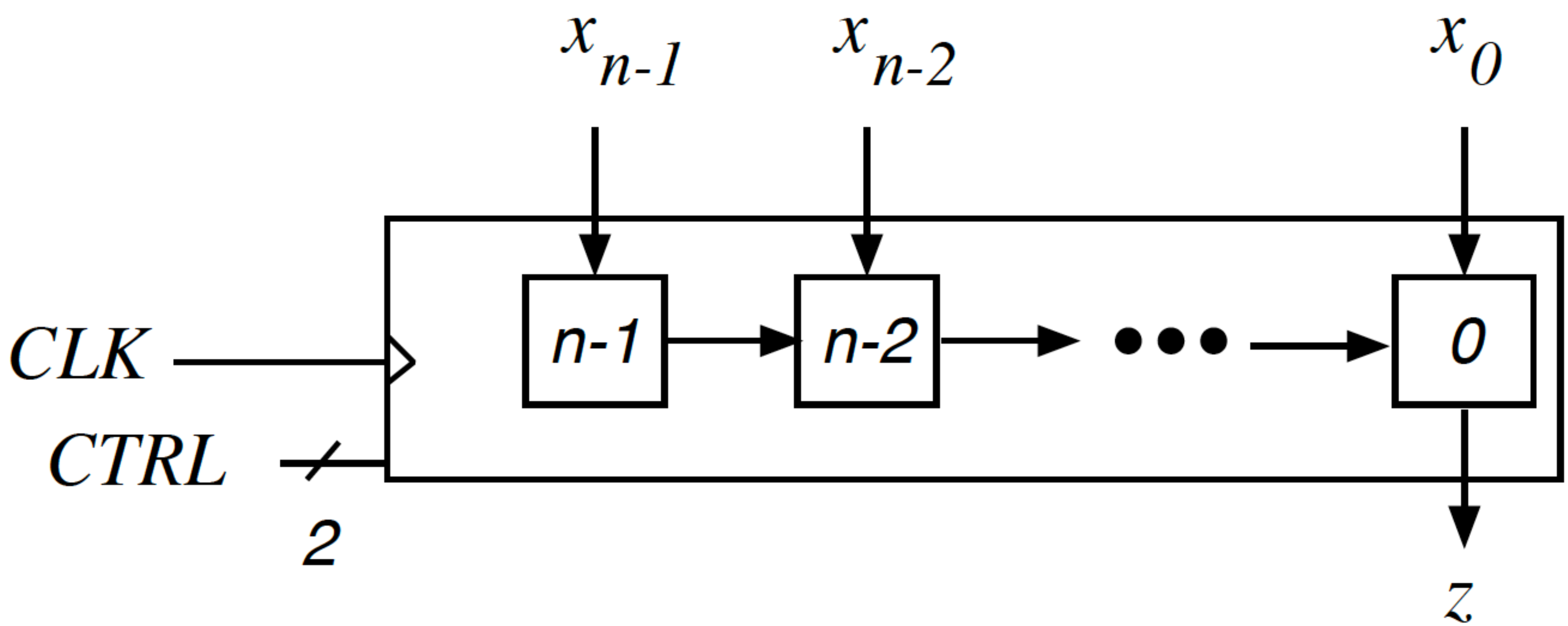


SERIAL-IN/SERIAL-OUT UNIDIRECTIONAL SHIFT REGISTER

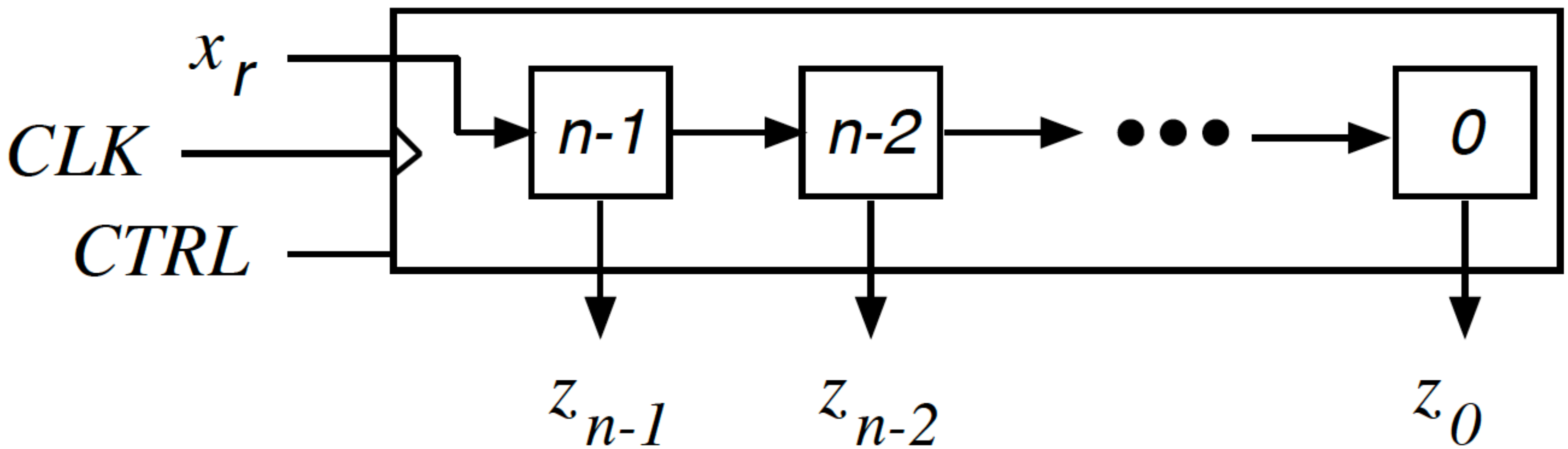
$$z(t) = x(t-n)$$



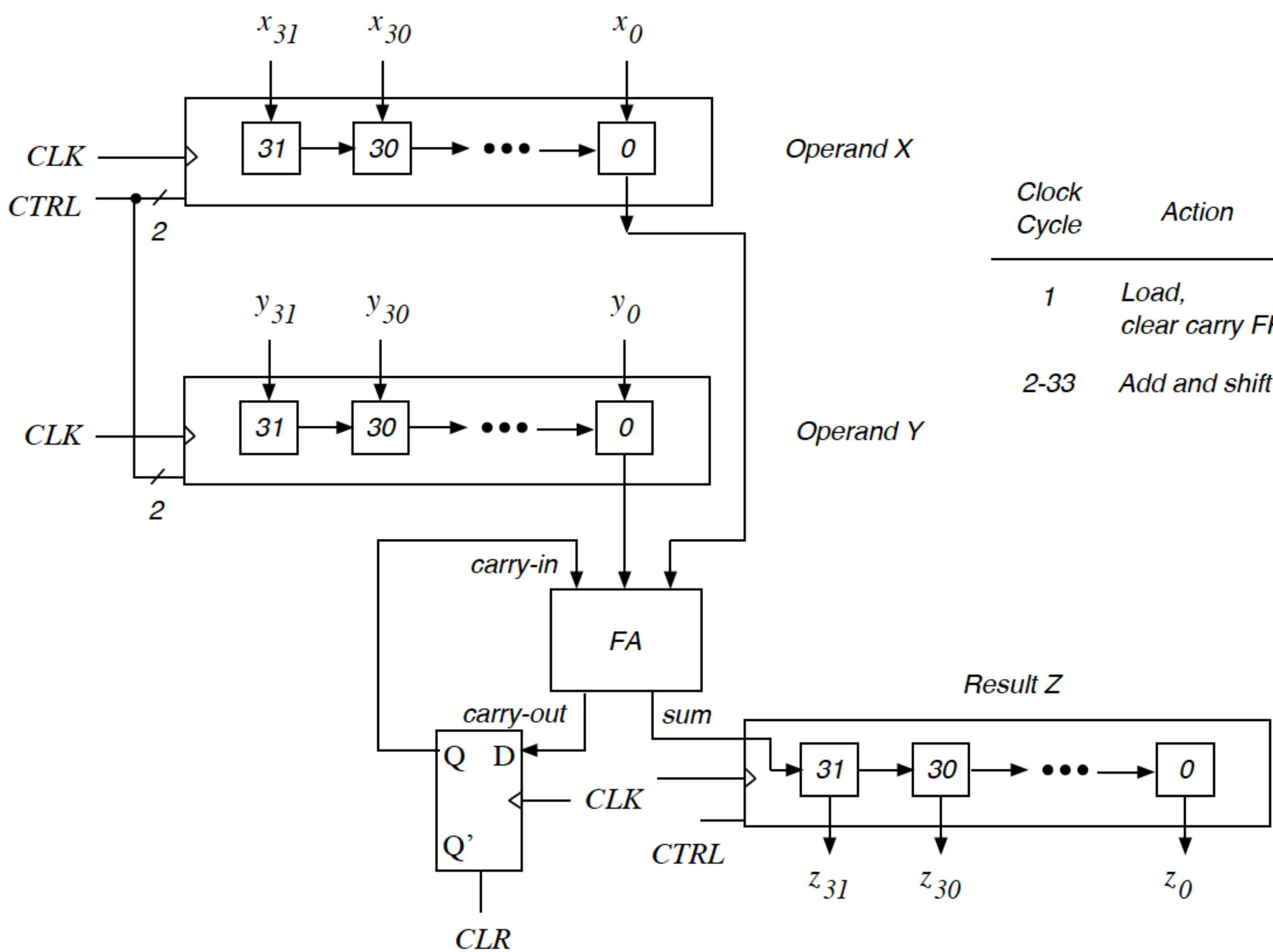
PARALLEL-IN/SERIAL-OUT UNIDIRECTIONAL SHIFT REGISTER



SERIAL-IN/PARALLEL-OUT UNIDIRECTIONAL SHIFT REGISTER



Example: Serial Adder



Clock Cycle	Action
1	Load, clear carry FF
2-33	Add and shift

Example

$$z(t) = \begin{cases} 1 & \textbf{if} \quad \underline{s}(t) = 01101110 \text{ and } x(t) = 1 \\ 0 & \textbf{otherwise} \end{cases}$$