جامعة الملك سعود
King Saud University

CSC 220: Computer Organization

Unit 8
Registers and RTL

Prepared by:
Md Saiful Islam, PhD

Updated by:
Isra Al-Turaiki, PhD

Department of Computer Science

College of Computer and Information Sciences

Overview

- Registers Construction
  - Basic Registers
  - Shift Registers
- Register Transfer Language
- Micro operations
- Multiplexer-Based Transfers
- Bus Construction

Chapter-6

M. Morris Mano, Charles R. Kime and Tom Martin, **Logic and Computer Design Fundamentals**, Global (5th) Edition, Pearson Education Limited, 2016. ISBN: 9781292096124

Registers

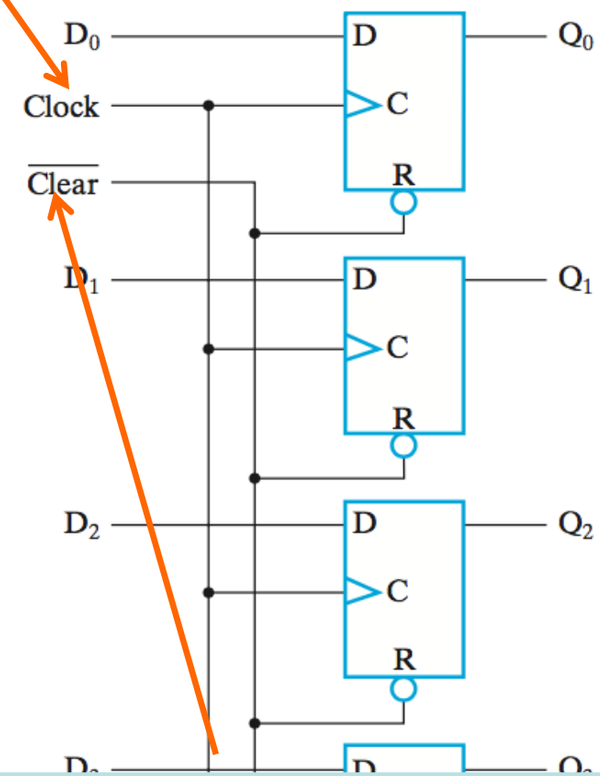
- *Register* –a set of flip- flops, together with gates that implement their state transitions.
- Each flip-flop capable of storing one bit.
- In theory, a register is sequential logic which can be defined by a state table
- Stores a vector of binary values
- Used to perform simple data storage and data movement and processing operations

Parallel Loading

- *Loading* is the transfer of new information into the register
- If all the bits of the registers are loaded simultaneously with a common clock pulse transition, we say that loading is done in parallel.

A basic Register : 4-bit register

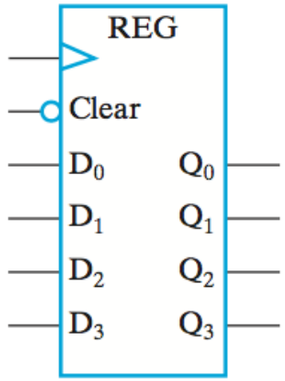
triggers all flip-flops on the rising edge of each pulse



clear the register to all 0s prior to its clocked operation. 0 must be applied to reset the flip-flops asynchronously.

(a) Logic diagram

The simplest possible register is one that consists of only flip-flops without any external gates.



(b) Symbol

A basic Register : 4-bit register

Example: a 4-bit register

– This register uses D flip-flops, so it's easy to store data without worrying about flip-flop input equations.

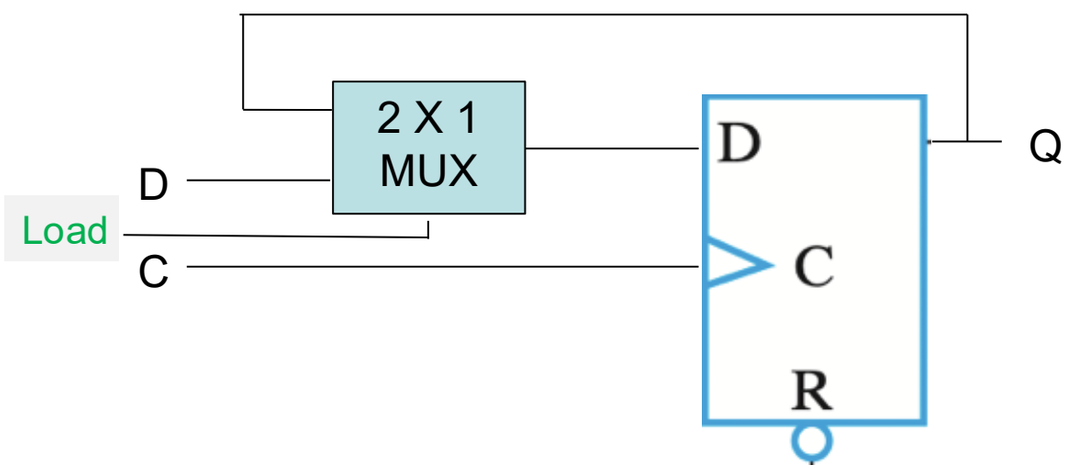
- All the flip-flops share a common CLK and CLR signal.

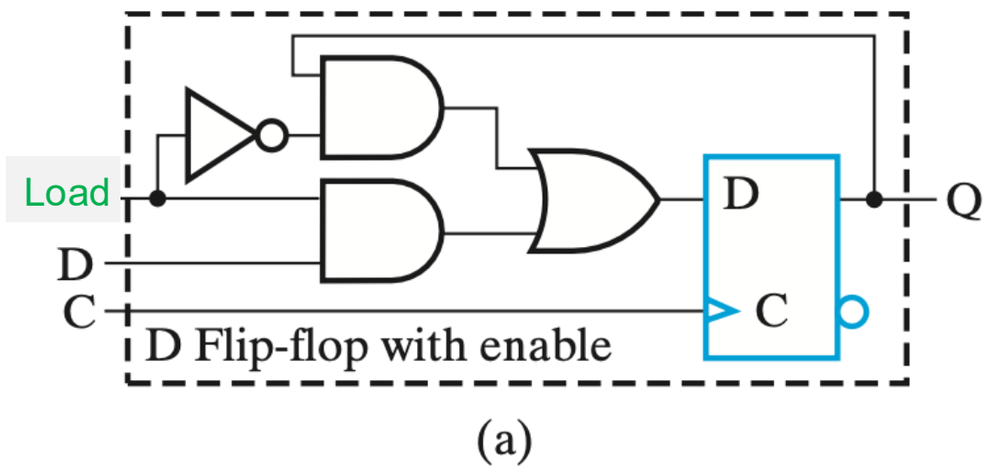
Register Storage

- Expectations:
  - A register can store information for multiple clock cycles
- Reality:
  - A D flip-flop register loads information on every clock cycle
- Realizing expectations:
  - Use a signal to control feedback of the output of the register back to its inputs,

Register with Load Signal

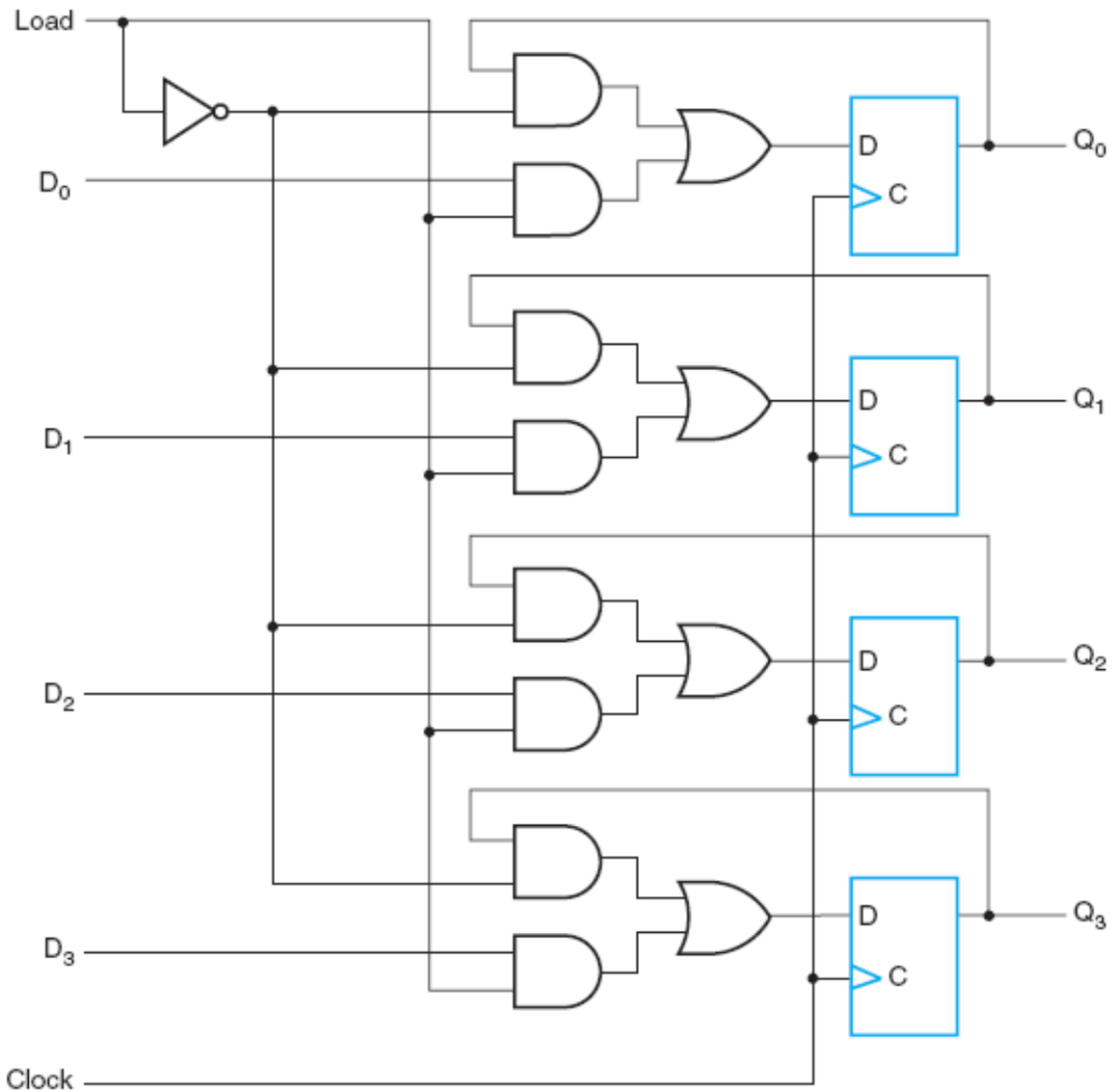
- Modify the D Flip-Flop input using Load signal.
- Load is a frequent name for the signal that controls register storage and loading
  - Load = 1: Load new values
  - Load = 0: hold current values





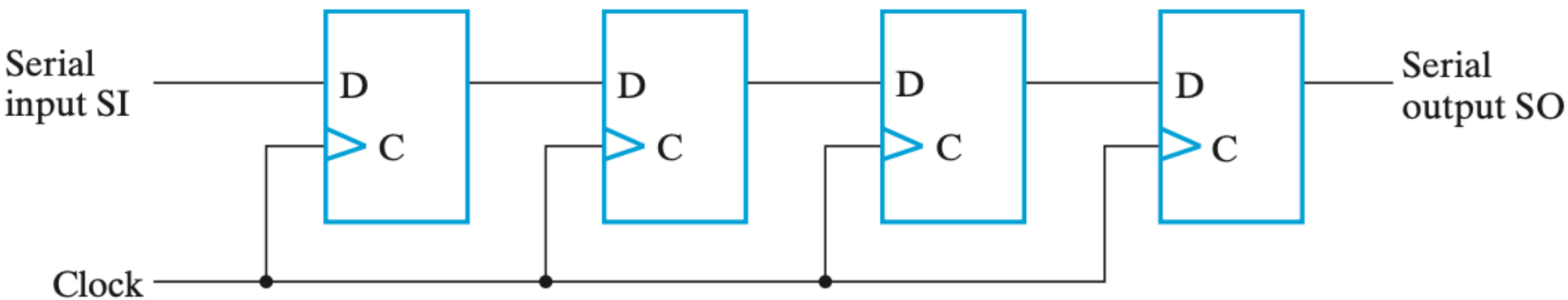
(a)

4-Bit Register with Parallel Load

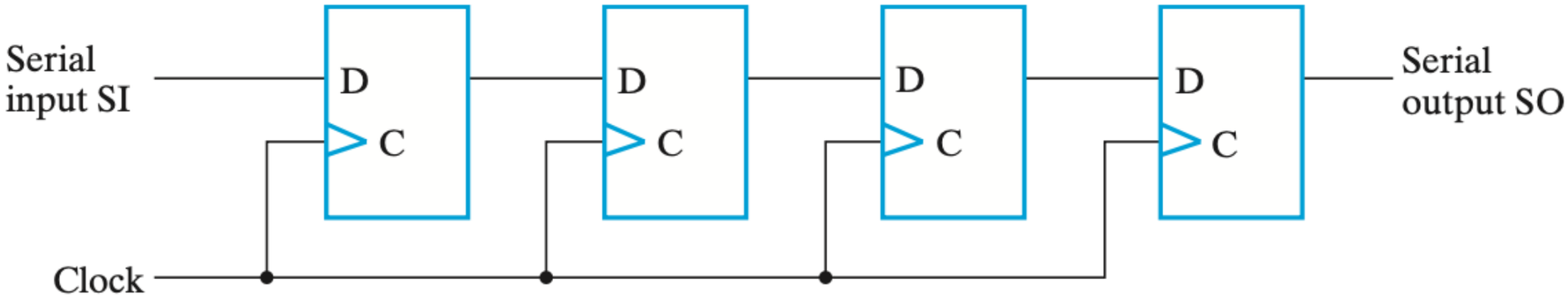


Shift Registers

- A register capable of shifting its stored bits laterally in one or both directions is called a *shift register*.
  - a chain of flip-flops, with the output of one flip-flop connected to the input of the next flip-flop.
  - All flip-flops have a common clock-pulse input that activates the shift.



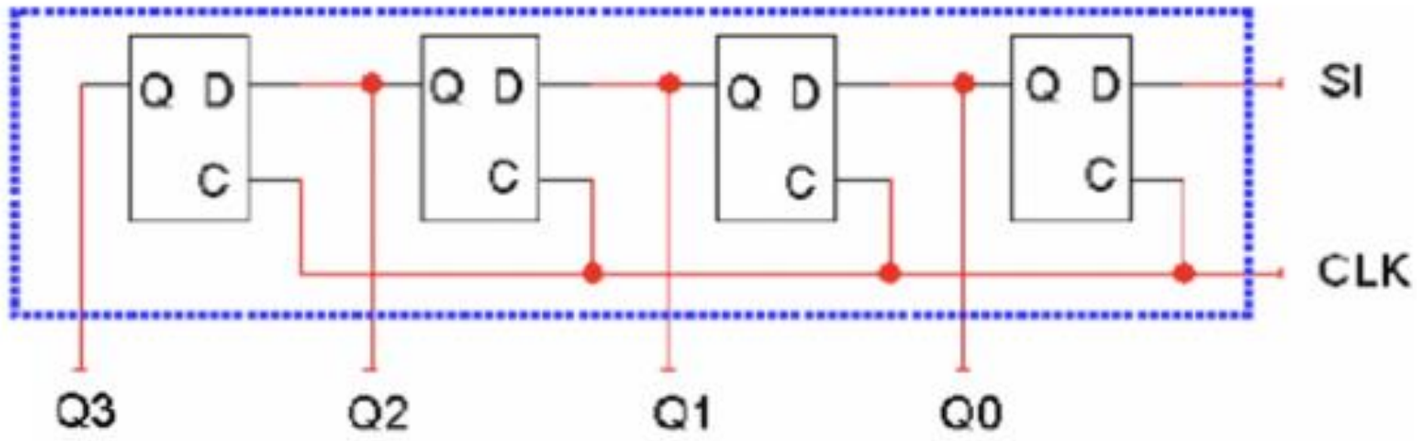
Shift Right



SI	Present State	Next State
1	0110	1011

- A shift register shifts its output once every clock cycle.
- SI is the input that supplies a new bit to shift into the register.
- The current Q0 (0 in this example) will be lost on the next cycle.

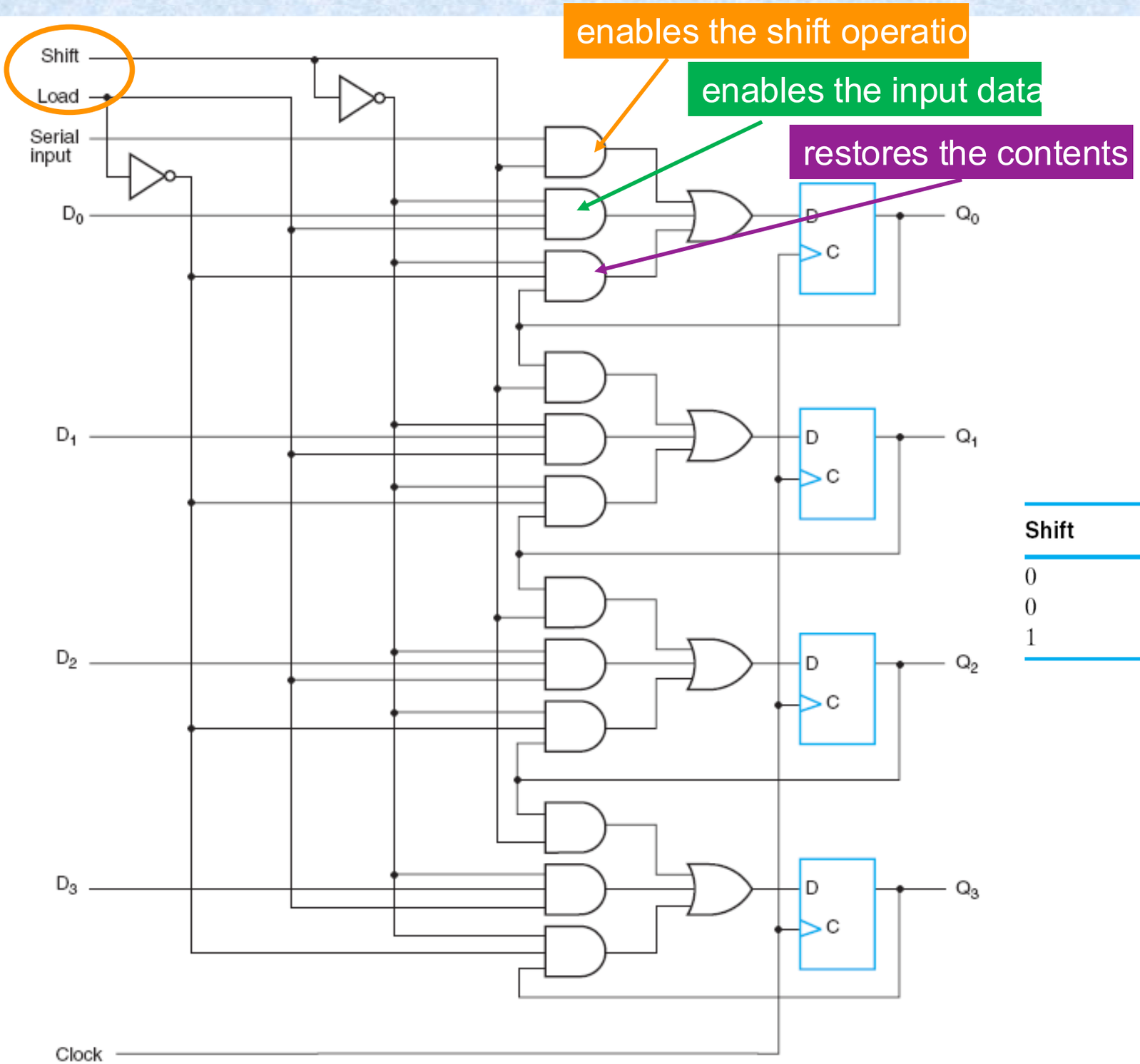
Shift Left

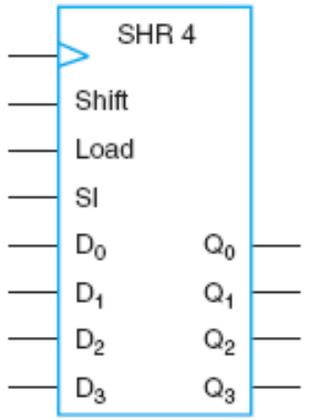


SI	Present State	Next State
1	0110	1101

- A shift register shifts its output once every clock cycle.
- SI is the input that supplies a new bit to shift into the register.
- The current Q3 (0 in this example) will be lost on the next cycle.

Shift Register with Parallel Load





(b) Symbol

Shift	Load	Operation
0	0	No change
0	1	Load parallel data
1	×	Shift down from Q_0 to Q_3

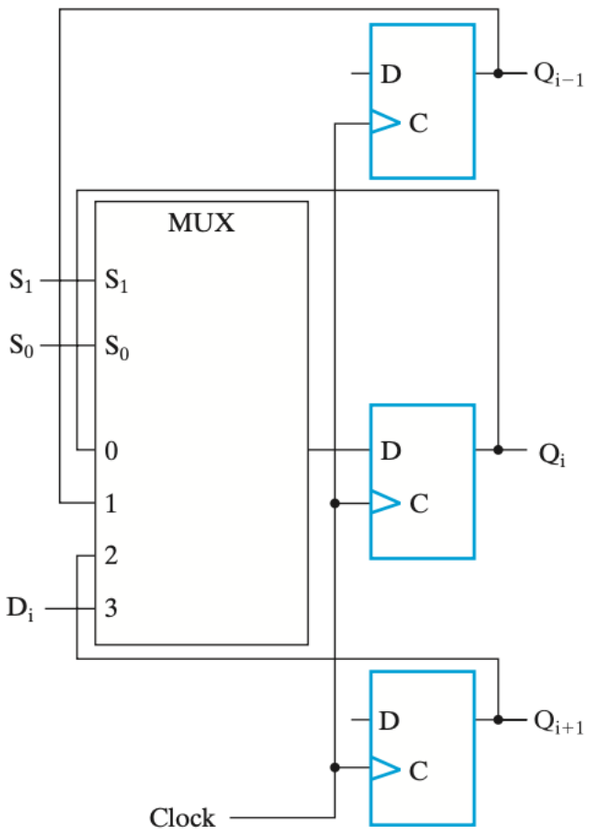
Function Table

A Bidirectional Shift Register with Parallel Load

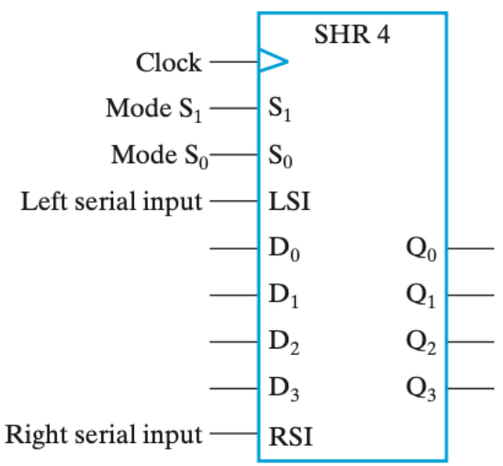
- A register capable of shifting in only one direction is called a *unidirectional* shift register.
- A register that can shift in both directions is a *bidirectional* shift register

Function Table for the Register of Figure 6-11

Mode Control		Register Operation
S_1	S_0	
0	0	No change (Hold)
0	1	Shift left
1	0	Shift right
1	1	Parallel load



(a) Logic diagram of one typical stage

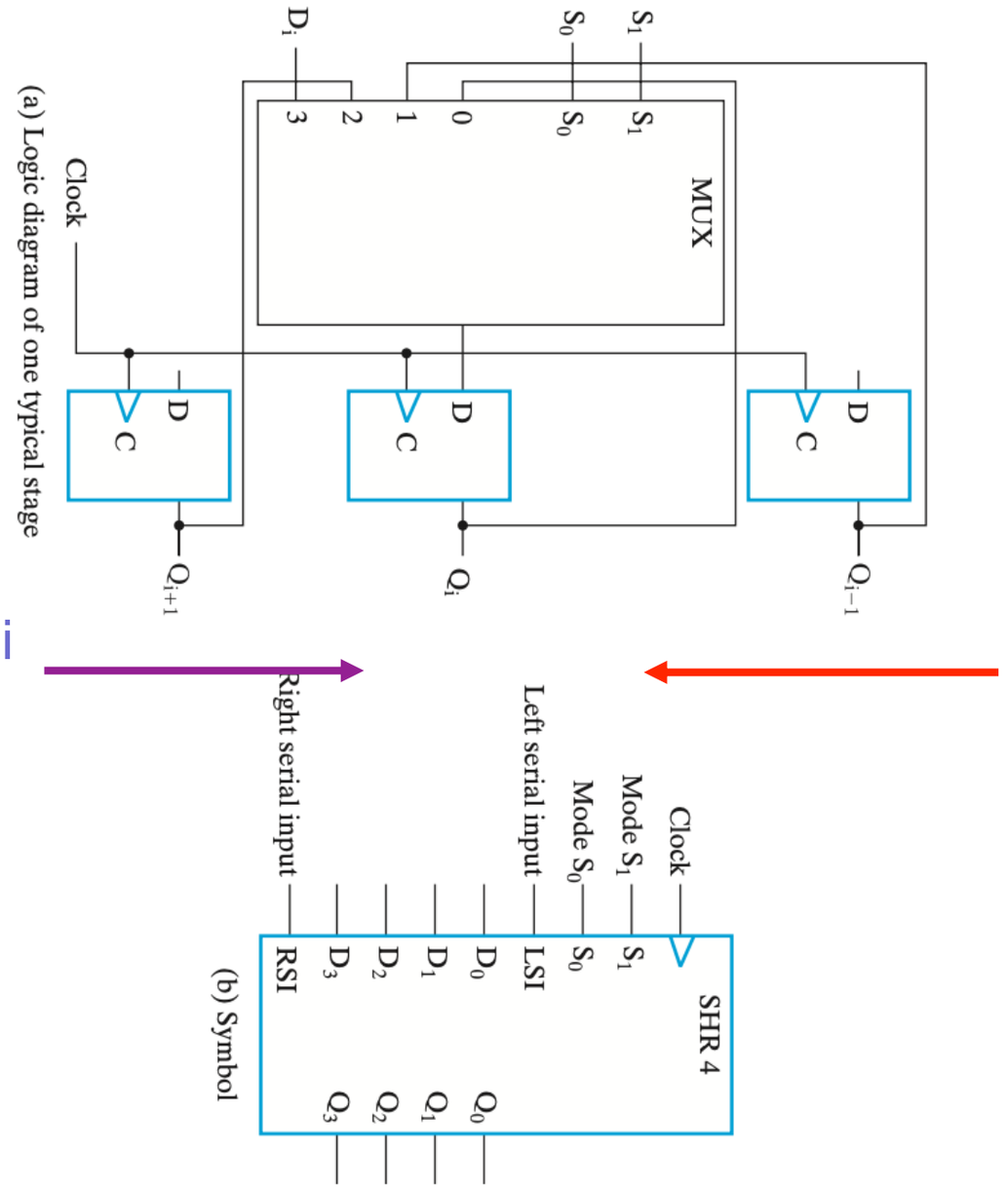


(b) Symbol

One stage of a bidirectional shift register with parallel load

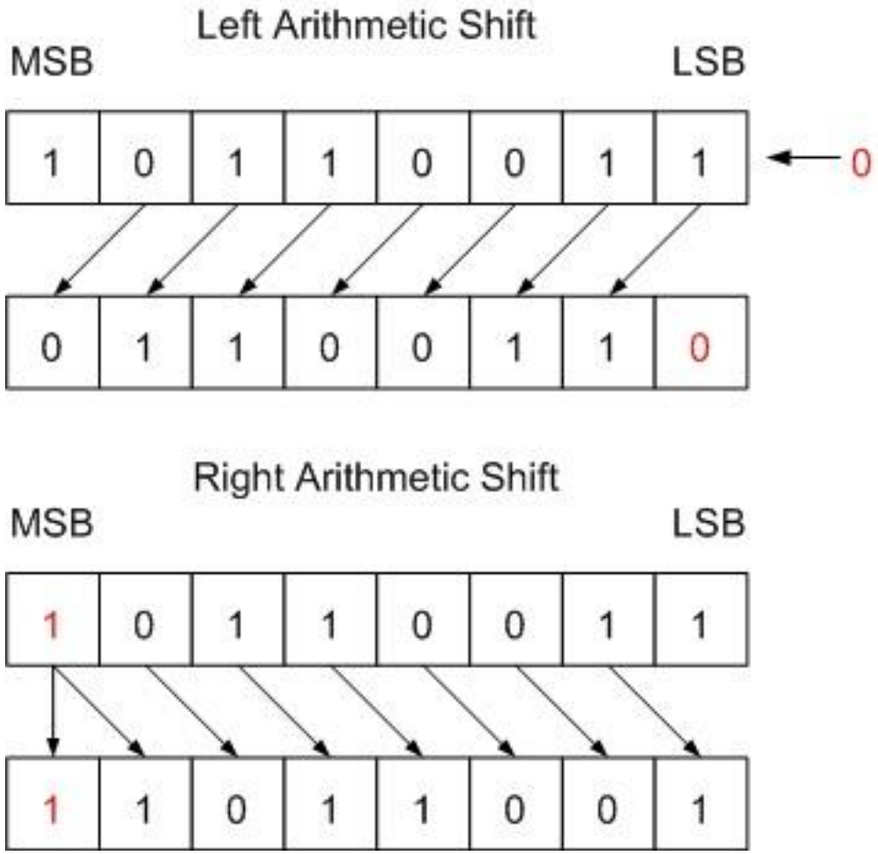
From Qi+1 to Qi (right, up)

From Qi-1 to Qi (left, down)



Other Types of Shift Registers

- Logical shifts – Standard shifts like we just saw. In the absence of a SI input, 0 occupies the vacant position.
  - Left: 0110 -> 1100
  - Right: 0110 -> 0011
- Circular shifts (also called ring counters or rotates) – The shifted out bit wraps around to the vacant position.
  - Left: 1001 -> 0011
  - Right: 1001 -> 1100
- Switch-tail ring counter (aka Johnson counter) – Similar to the ring counter, but the serial input is the complement of the serial output.
  - Left: 1001 -> 0010
  - Right: 1001 -> 0100
- Arithmetical shifts – Left shifting is the same as a logical shift. Right shifting however maintains the MSB.
  - Left: 0110 -> 1100
  - Right: 0110 -> 0011; 1011 -> 1101


Left Arithmetic Shift
MSB
LSB
1 0 1 1 0 0 1 1
0
0 1 1 0 0 1 1 0
Right Arithmetic Shift
MSB
LSB
1 0 1 1 0 0 1 1
1 1 0 1 1 0 0 1

Example

- Consider a 4-bit register with the following inputs
  - parallel data inputs ABCD = 0011
  - left serial input LSI = 1 (This is the serial input for a left shift.)
  - right serial input RSI = 0 (This is the serial input for a right shift.)
  - The three control inputs S_2, S_1, S_0 operate as shown in the table to the right.
  - Let the initial register content be $Q_A Q_B Q_C Q_D$ = 0101

S2	S1	S0	Operation
0	0	0	left shift
0	0	1	circular left shift
0	1	0	right shift
0	1	1	circular right shift
1	0	0	no change
1	0	1	parallel load
1	1	0	complement each bit
1	1	1	set to 1111

Example

S2	S1	S0	Q_A	Q_B	Q_C	Q_D
			0	1	0	1
0	0	0				
0	0	1				
0	1	0				
1	0	0				
0	1	1				
1	0	1				
1	1	1				
0	0	0				
1	1	0				

S2	S1	S0	Operation
0	0	0	left shift
0	0	1	circular left shift
0	1	0	right shift
0	1	1	circular right shift
1	0	0	no change
1	0	1	parallel load
1	1	0	complement each bit
1	1	1	set to 1111
Other Information			
Parallel Load			ABCD = 0011
LS1			1
RSI			0

Example

S2	S1	S0	Q_A	Q_B	Q_C	Q_D
			0	1	0	1
0	0	0				
0	0	1				
0	1	0				
1	0	0				
0	1	1				
1	0	1				
1	1	1				
0	0	0				
1	1	0				

S2	S1	S0	Operation
0	0	0	left shift
0	0	1	circular left shift
0	1	0	right shift
0	1	1	circular right shift
1	0	0	no change
1	0	1	parallel load
1	1	0	complement each bit
1	1	1	set to 1111
Other Information			
Parallel Load			ABCD = 0011
LS1			1
RSI			0

Example

S2	S1	S0	Q_A	Q_B	Q_C	Q_D
			0	1	0	1
0	0	0	**1**	**0**	**1**	**1**
0	0	1				
0	1	0				
1	0	0				
0	1	1				
1	0	1				
1	1	1				
0	0	0				
1	1	0				

S2	S1	S0	Operation
0	0	0	left shift
0	0	1	circular left shift
0	1	0	right shift
0	1	1	circular right shift
1	0	0	no change
1	0	1	parallel load
1	1	0	complement each bit
1	1	1	set to 1111

Other Information	
Parallel Load	ABCD = 0011
LS1	1
RSI	0

Example

S2	S1	S0	Q_A	Q_B	Q_C	Q_D
			0	1	0	1
0	0	0	1	0	1	1
0	0	1	0	1	1	1
0	1	0				
1	0	0				
0	1	1				
1	0	1				
1	1	1				
0	0	0				
1	1	0				

S2	S1	S0	Operation
0	0	0	left shift
0	0	1	circular left shift
0	1	0	right shift
0	1	1	circular right shift
1	0	0	no change
1	0	1	parallel load
1	1	0	complement each bit
1	1	1	set to 1111
Other Information			
Parallel Load			ABCD = 0011
LS1			1
RSI			0

Example

S2	S1	S0	Q_A	Q_B	Q_C	Q_D
			0	1	0	1
0	0	0	1	0	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	1
1	0	0				
0	1	1				
1	0	1				
1	1	1				
0	0	0				
1	1	0				

S2	S1	S0	Operation
0	0	0	left shift
0	0	1	circular left shift
0	1	0	right shift
0	1	1	circular right shift
1	0	0	no change
1	0	1	parallel load
1	1	0	complement each bit
1	1	1	set to 1111

Other Information	
Parallel Load	ABCD = 0011
LS1	1
RSI	0

Example

S2	S1	S0	Q_A	Q_B	Q_C	Q_D
			0	1	0	1
0	0	0	1	0	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	1
1	0	0	0	0	1	1
0	1	1				
1	0	1				
1	1	1				
0	0	0				
1	1	0				

S2	S1	S0	Operation
0	0	0	left shift
0	0	1	circular left shift
0	1	0	right shift
0	1	1	circular right shift
1	0	0	no change
1	0	1	parallel load
1	1	0	complement each bit
1	1	1	set to 1111

Other Information	
Parallel Load	ABCD = 0011
LS1	1
RSI	0

Example

S2	S1	S0	Q_A	Q_B	Q_C	Q_D
			0	1	0	1
0	0	0	1	0	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	1
1	0	0	0	0	1	1
0	1	1	1	0	0	1
1	0	1				
1	1	1				
0	0	0				
1	1	0				

S2	S1	S0	Operation
0	0	0	left shift
0	0	1	circular left shift
0	1	0	right shift
0	1	1	circular right shift
1	0	0	no change
1	0	1	parallel load
1	1	0	complement each bit
1	1	1	set to 1111

Other Information	
Parallel Load	ABCD = 0011
LS1	1
RSI	0

Example

S2	S1	S0	Q_A	Q_B	Q_C	Q_D
			0	1	0	1
0	0	0	1	0	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	1
1	0	0	0	0	1	1
0	1	1	1	0	0	1
1	0	1	0	0	1	1
1	1	1				
0	0	0				
1	1	0				

S2	S1	S0	Operation
0	0	0	left shift
0	0	1	circular left shift
0	1	0	right shift
0	1	1	circular right shift
1	0	0	no change
1	0	1	parallel load
1	1	0	complement each bit
1	1	1	set to 1111
Other Information			
Parallel Load			ABCD = 0011
LS1			1
RSI			0

Example

S2	S1	S0	Q_A	Q_B	Q_C	Q_D
			0	1	0	1
0	0	0	1	0	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	1
1	0	0	0	0	1	1
0	1	1	1	0	0	1
1	0	1	0	0	1	1
1	1	1	1	1	1	1
0	0	0				
1	1	0				

S2	S1	S0	Operation
0	0	0	left shift
0	0	1	circular left shift
0	1	0	right shift
0	1	1	circular right shift
1	0	0	no change
1	0	1	parallel load
1	1	0	complement each bit
1	1	1	set to 1111
Other Information			
Parallel Load			ABCD = 0011
LS1			1
RSI			0

Example

S2	S1	S0	Q_A	Q_B	Q_C	Q_D
			0	1	0	1
0	0	0	1	0	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	1
1	0	0	0	0	1	1
0	1	1	1	0	0	1
1	0	1	0	0	1	1
1	1	1	1	1	1	1
0	0	0	1	1	1	1
1	1	0				

S2	S1	S0	Operation
0	0	0	left shift
0	0	1	circular left shift
0	1	0	right shift
0	1	1	circular right shift
1	0	0	no change
1	0	1	parallel load
1	1	0	complement each bit
1	1	1	set to 1111

Other Information	
Parallel Load	ABCD = 0011
LS1	1
RSI	0

Example

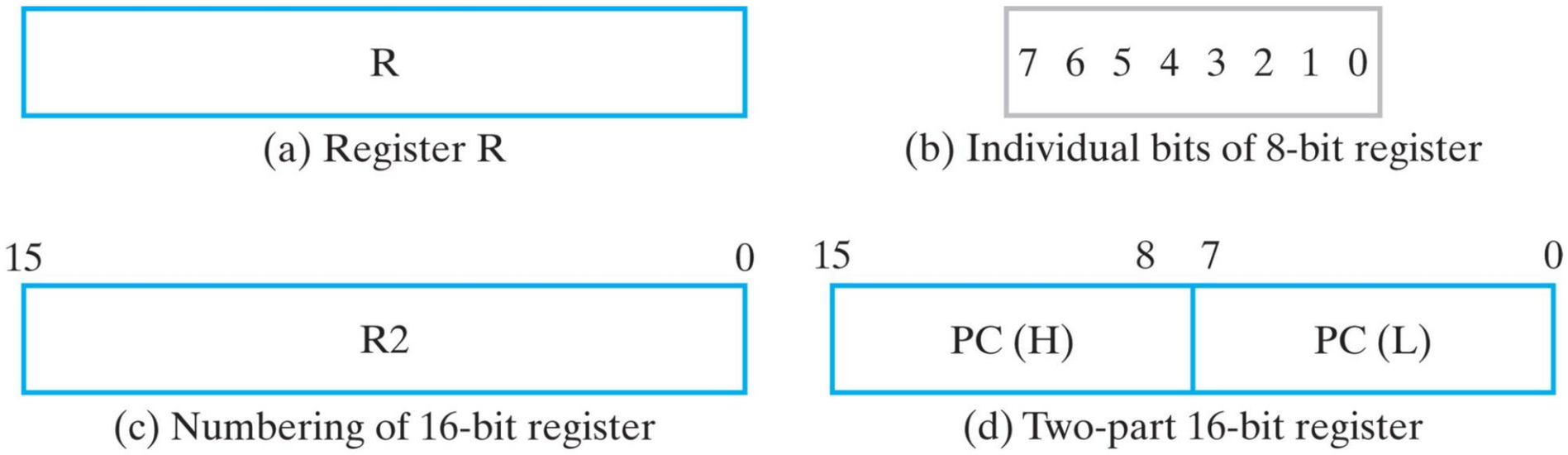
S2	S1	S0	Q_A	Q_B	Q_C	Q_D
			0	1	0	1
0	0	0	1	0	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	1
1	0	0	0	0	1	1
0	1	1	1	0	0	1
1	0	1	0	0	1	1
1	1	1	1	1	1	1
0	0	0	1	1	1	1
1	1	0	0	0	0	0

S2	S1	S0	Operation
0	0	0	left shift
0	0	1	circular left shift
0	1	0	right shift
0	1	1	circular right shift
1	0	0	no change
1	0	1	parallel load
1	1	0	complement each bit
1	1	1	set to 1111
Other Information			
Parallel Load			ABCD = 0011
LS1			1
RSI			0

Register Transfer Operations

- Micro operations (micro-ops) are operations on data stored in registers
- Register transfer language (RTL) is a concise and precise means of describing those operations
- RTL expressions are made up of elements which describe the registers being manipulated, and the micro-ops being performed on them
- Registers are denoted by uppercase letters (sometimes followed by numbers) that indicate the function of the register
  - e.g. R0, R1, AR, PC, MAR, et al.
  - The individual bits can be denoted using parenthesis and bit numbers or labels
    - e.g. R0(0), R0(7:0), PC(L), PC(H)

Register Transfer Operations



(a) Register R

(b) Individual bits of 8-bit register

(c) Numbering of 16-bit register

(d) Two-part 16-bit register

Register Transfer Language

TABLE 6-1
Basic Symbols for Register Transfers

Symbol	Description	Examples
Letters (and numerals)	Denotes a register	$AR, R2, DR, IR$
Parentheses	Denotes a part of a register	$R2(1), R2(7{:}0), AR(L)$
Arrow	Denotes transfer of data	$R1 \leftarrow R2$
Comma	Separates simultaneous transfers	$R1 \leftarrow R2, R2 \leftarrow R1$
Square brackets	Specifies an address for memory	$DR \leftarrow M[AR]$

Destination register ← Source register:

- Data in source register does not change
- A datapath are available from the outputs of the source register to the inputs of the destination register
- The destination register has a parallel load capability
- All RTL statements occur in response to a clock tick

Register Transfer Language

- Normally we want a given transfer to occur not for every clock pulse, but only for specific values of the control signals.
- This can be specified by a *conditional statement*.

  e.g. If (K1 = 1) Then (R2 ← R1)
- Control function notation (Colon, :)

  e.g. K1: R2 ← R1

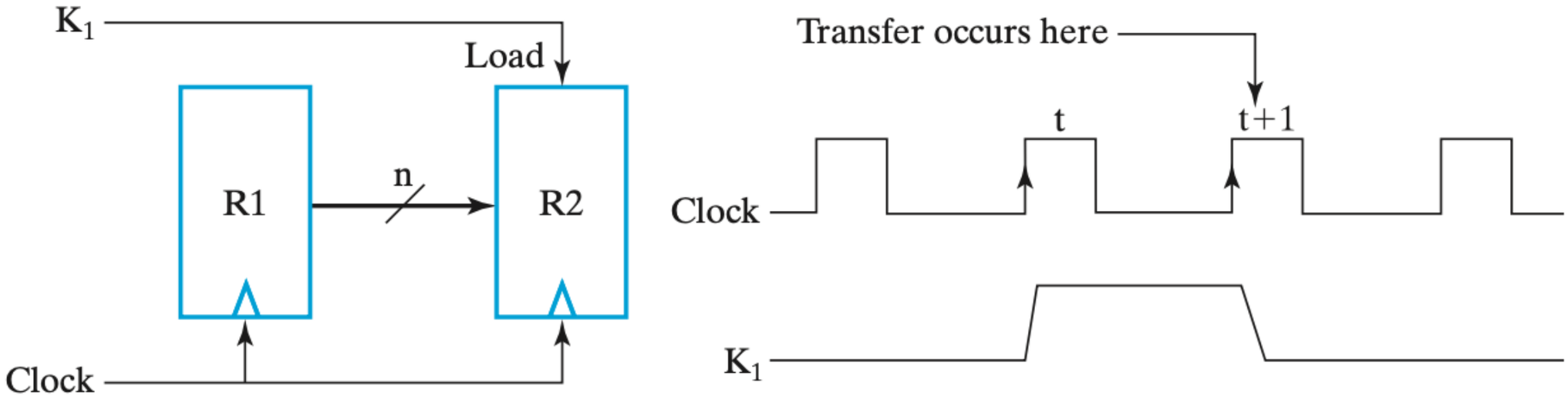


FIGURE 6-5
Transfer from *R*1 to *R*2 when $K_1 = 1$

Microoperations

- In digital systems there are 4 types of microoperation
  - Transfer - move data from one set of registers to another
  - Arithmetic - perform arithmetic on data in registers
  - Logic – perform bit manipulation on data in registers
  - Shift - shift data in registers

Arithmetic Microoperations

- Every RTL statement written in register-transfer notation presupposes a hardware construct for implementing the transfer.

Arithmetic Microoperations

Symbolic designation	Description
$R0 \leftarrow R1 + R2$	Contents of $R1$ plus $R2$ transferred to $R0$
$R2 \leftarrow \overline{R2}$	Complement of the contents of $R2$ (1's complement)
$R2 \leftarrow \overline{R2} + 1$	2's complement of the contents of $R2$
$R0 \leftarrow R1 + \overline{R2} + 1$	$R1$ plus 2's complement of $R2$ transferred to $R0$ (subtraction)
$R1 \leftarrow R1 + 1$	Increment the contents of $R1$ (count up)
$R1 \leftarrow R1 - 1$	Decrement the contents of $R1$ (count down)

Logical Microoperations

These operations treat each bit in the register as a binary variable.

Symbolic designation	Description
$R0 \leftarrow \overline{R1}$	Logical bitwise NOT (1's complement)
$R0 \leftarrow R1 \wedge R2$	Logical bitwise AND (clears bits)
$R0 \leftarrow R1 \vee R2$	Logical bitwise OR (sets bits)
$R0 \leftarrow R1 \oplus R2$	Logical bitwise XOR (complements bits)

Logical Microoperations

- Let R1 = 10101010 and R2 = 11110000
- Then after the operation, R0 becomes:

R0	Operation
01010101	$R0 \leftarrow \overline{R1}$
11111010	$R0 \leftarrow R1 \vee R2$
10100000	$R0 \leftarrow R1 \wedge R2$
01011010	$R0 \leftarrow R1 \oplus R2$

Shift Microoperations

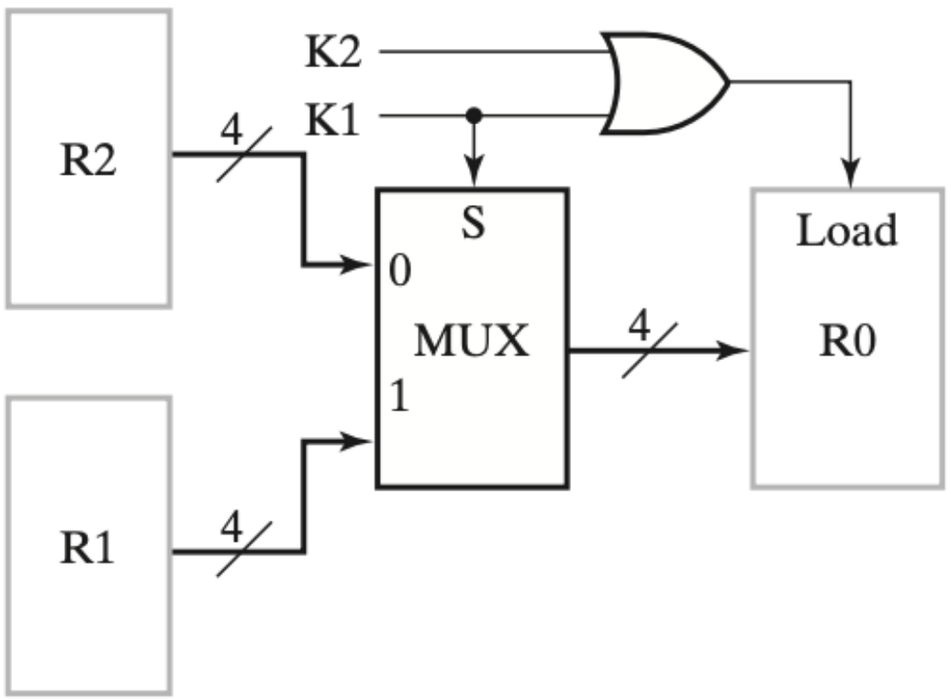
- We can shift values left or right by one bit.
- The source register is not modified, and we assume that the shift input is always 0.

TABLE 6-5
Examples of Shifts

Type	Symbolic Designation	Eight-Bit Examples	
		Source *R2*	After Shift: Destination *R1*
Shift left	$R1 \leftarrow \text{sl } R2$	10011110	00111100
Shift right	$R1 \leftarrow \text{sr } R2$	11100101	01110010

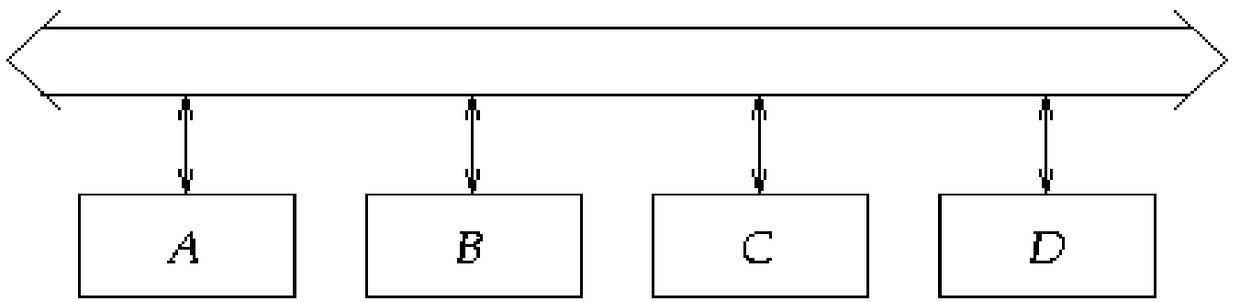
Multiplexer-Based Transfers

- There are occasions when a register receives data from two or more different sources at different times.
- Example $K_1: R0 \leftarrow R1, \quad \overline{K}_1 K_2: R0 \leftarrow R2$



Bus Construction

A bus consists of a set of parallel data lines



Multiplexer-based Bus Construction

Figure 4-3 Bus system for four registers.

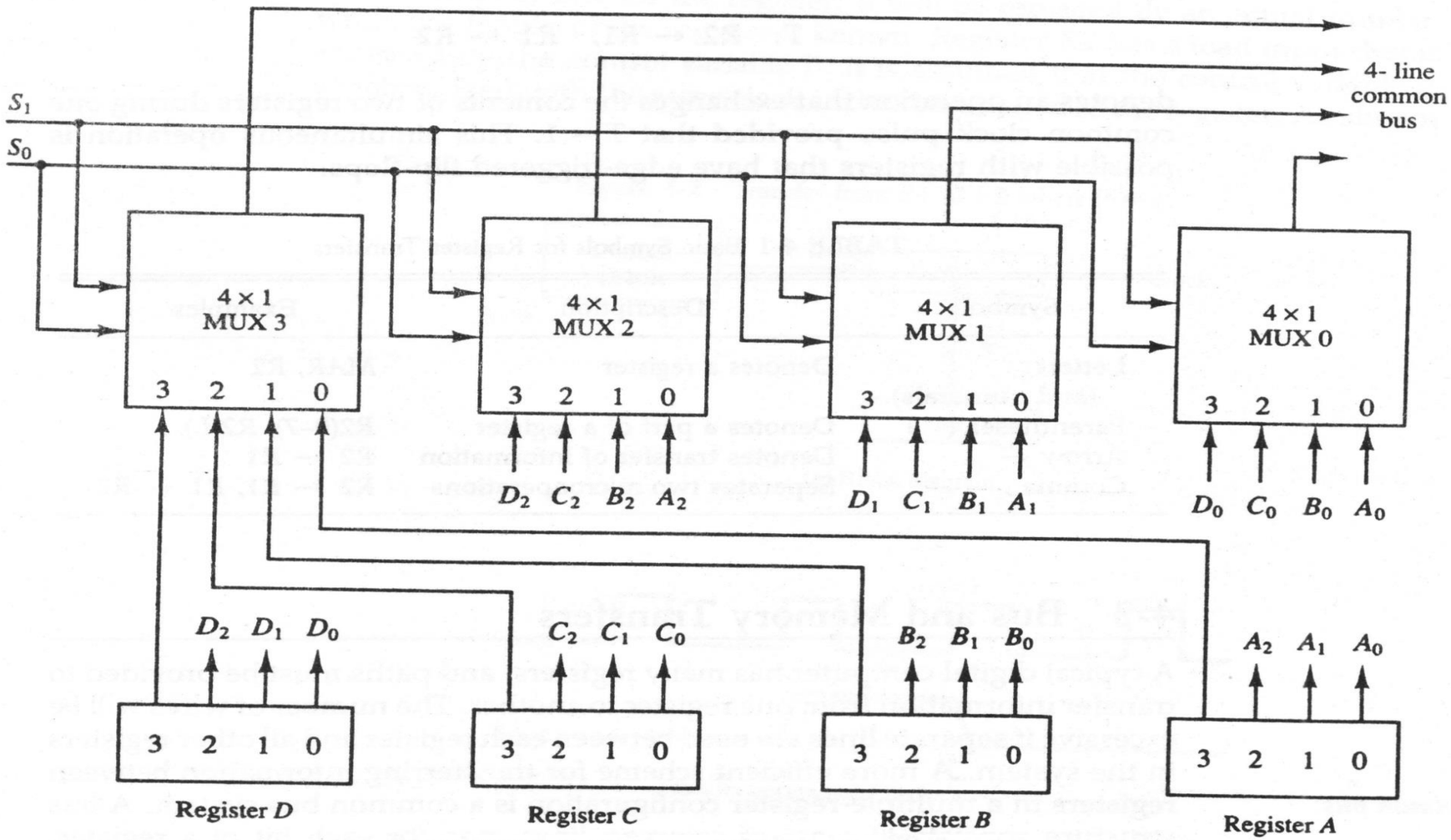
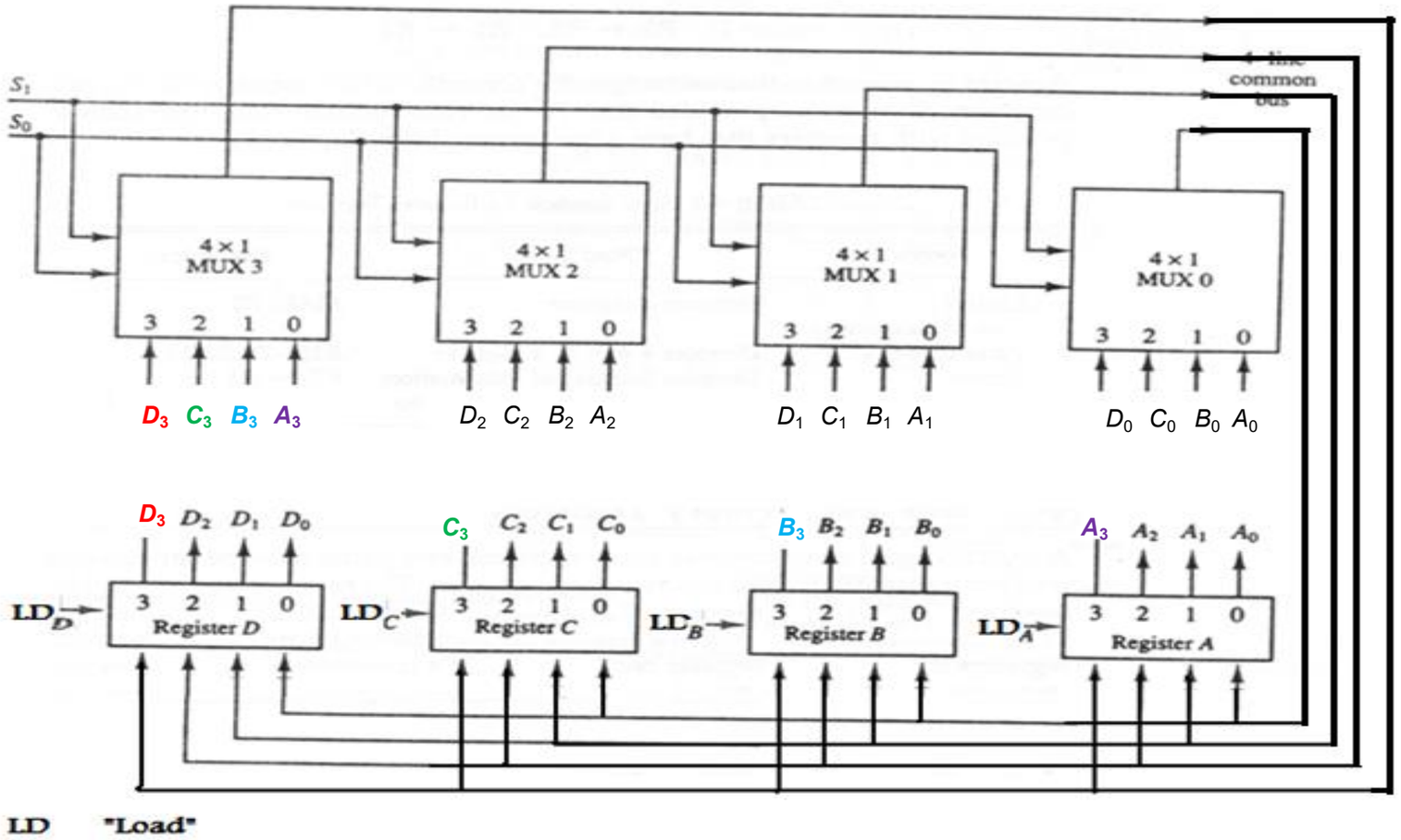


TABLE 4-2 Function Table for Bus of Fig. 4-3

S_1	S_0	Register selected
0	0	A
0	1	B
1	0	C
1	1	D

Bus-based Transfers for Multiple Registers



To transfer data using a bus:

- connect the output of the source register to the bus;
- connect the input of the target register to the bus;
- when the clock pulse arrives, the transfer occurs

Examples: Bus-based transfers

S1	S0	LD_A	LD_B	LD_C	LD_D	Operation
0	0	0	1	0	0	Register B ← Register A
1	1	1	0	0	0	Register A ← Register D
0	0	0	1	1	0	Register B ← Register A; Register C ← Register A