Overview of Chapter 2

Integrated Circuits (ICs)

SSI - several gates

MSI - 10-200 gates

LSI - 200-1000 gates

VLSI - 1,000s of gates

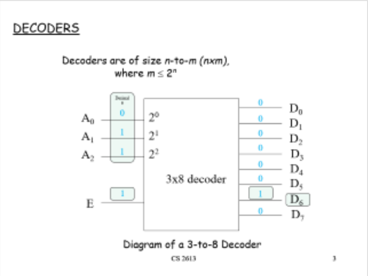
Technologies

TTL - long evolution from DTL

ECL - Emitter-coupled logic – high speed

MOS - metal oxide semiconductor - high density

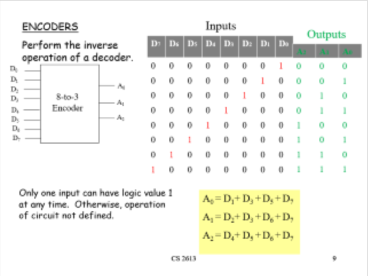
CMOS - low power



DECODERS

Decoders are of size *n*-to-*m* (*nxm*), where $m \leq 2^n$

Diagram of a 3-to-8 Decoder



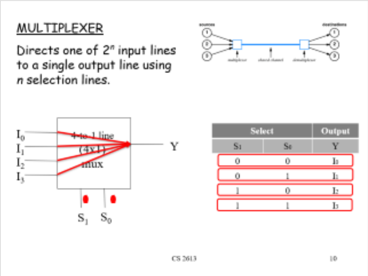
ENCODERS

Perform the inverse operation of a decoder.

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Only one input can have logic value 1 at any time. Otherwise, operation of circuit not defined.

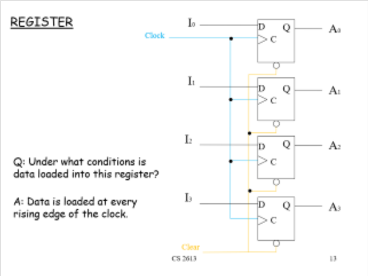
$$A_0 = D_1 + D_3 + D_5 + D_7$$
$$A_1 = D_2 + D_3 + D_6 + D_7$$
$$A_2 = D_4 + D_5 + D_6 + D_7$$



MULTIPLEXER

Directs one of 2^n input lines to a single output line using *n* selection lines.

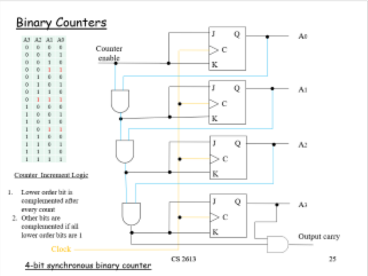
Select		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



REGISTER

Q: Under what conditions is data loaded into this register?

A: Data is loaded at every rising edge of the clock.



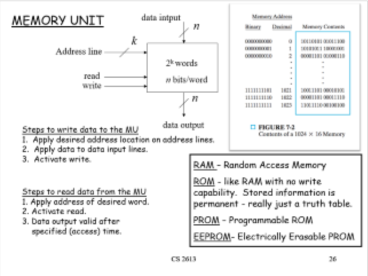
Binary Counters

A3	A2	A1	A0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Counter Increment Logic

1. Lower order bit is complemented after every count
2. Other bits are complemented if all lower order bits are 1

4-bit synchronous binary counter



MEMORY UNIT

Steps to write data to the MU
1. Apply desired address location on address lines.
2. Apply data to data input lines.
3. Activate write.

Steps to read data from the MU
1. Apply address of desired word.
2. Activate read.
3. Data output valid after specified (access) time.

Memory Address		
Binary	Decimal	Memory Contents
0000000000	0	10110101 01011100
0000000001	1	10101011 10001001
0000000010	2	00001101 01000110
.	.	.
1111111101	1021	10011101 00010101
1111111110	1022	00001101 00011110
1111111111	1023	11011110 00100100

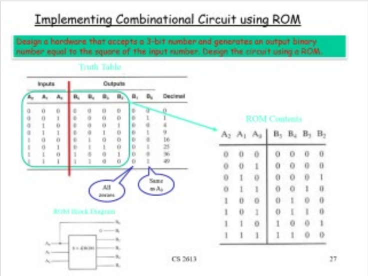
□ FIGURE 7-2
Contents of a 1024 × 16 Memory

RAM - Random Access Memory

ROM - like RAM with no write capability. Stored information is permanent - really just a truth table.

PROM - Programmable ROM

EEPROM- Electrically Erasable PROM



Implementing Combinational Circuit using ROM

Design a hardware that accepts a 3-bit number and generates an output binary number equal to the square of the input number. Design the circuit using a ROM.

Truth Table

Inputs			Outputs						
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

All zeroes

Same as A_0

ROM Contents

A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

ROM Block Diagram

DECODERS

Decoders are of size *n*-to-*m* (*nxm*),
where $m \leq 2^n$

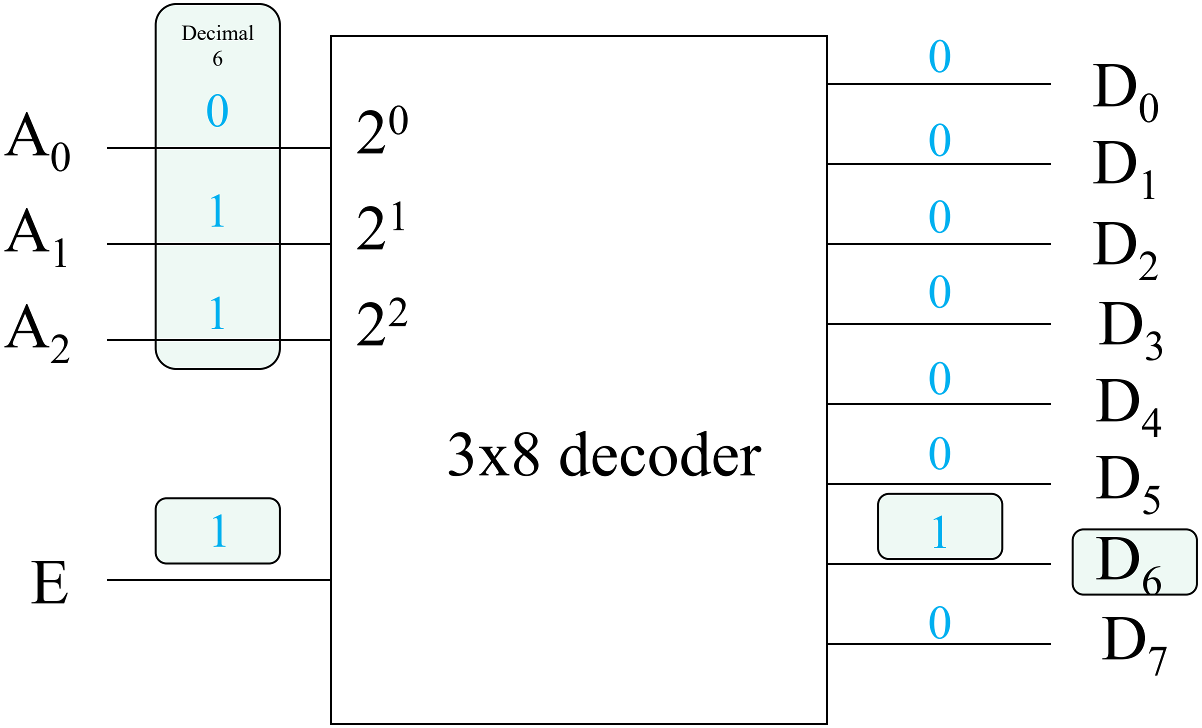
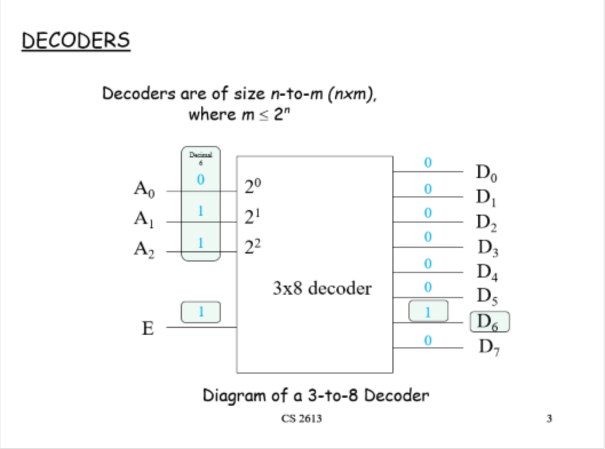


Diagram of a 3-to-8 Decoder



DECODERS

Decoders are of size *n*-to-*m* (*nxm*), where $m \leq 2^n$

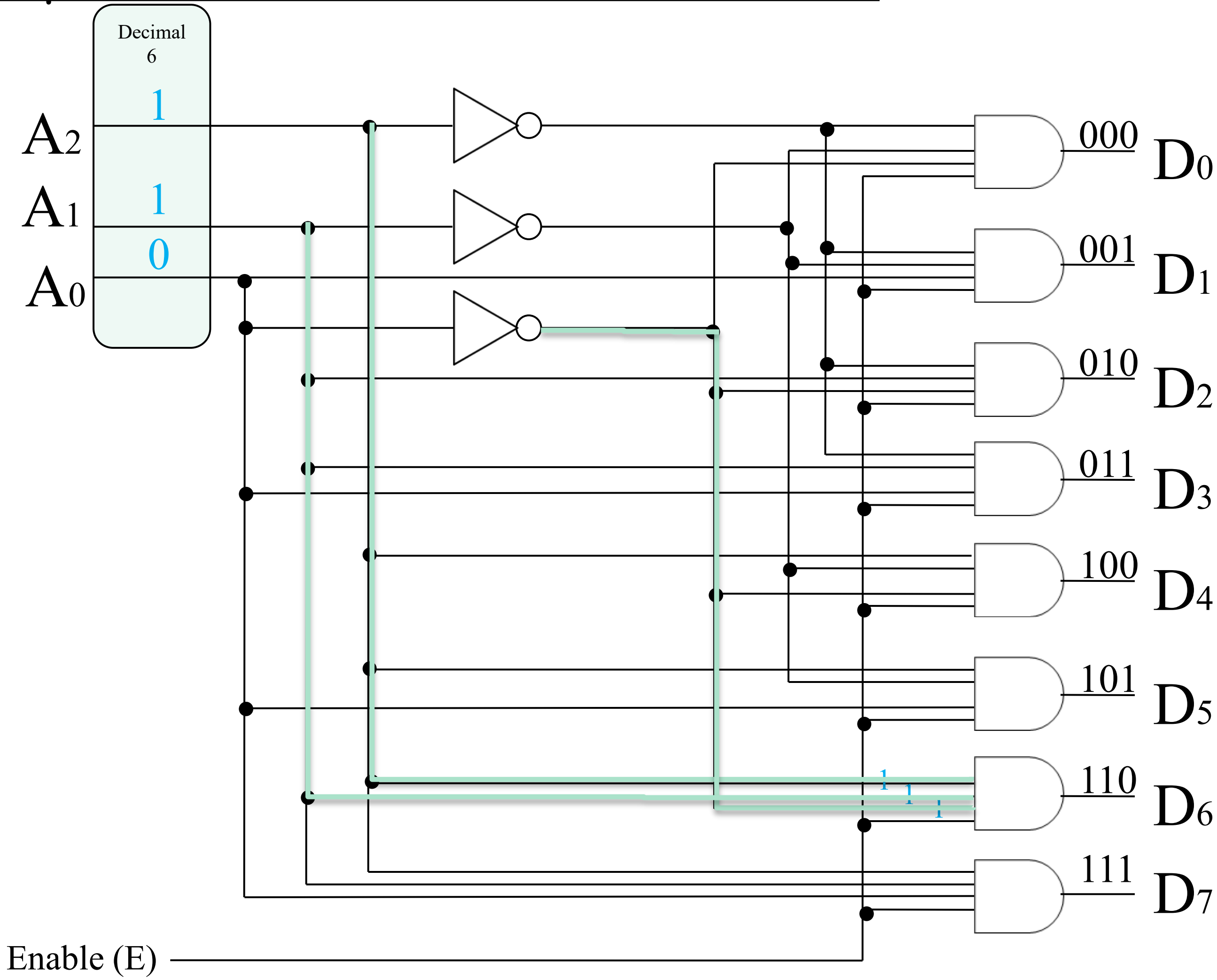
Diagram of a 3-to-8 Decoder

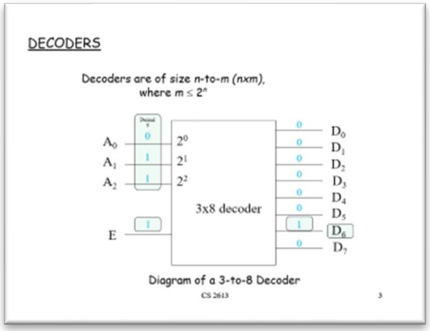
CS 2613 3

Input and Output Definition for 3-to-8 line Decoder

Enable	**Inputs**			**Outputs**							
E	A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Implementation of 3-to-8 line decoder





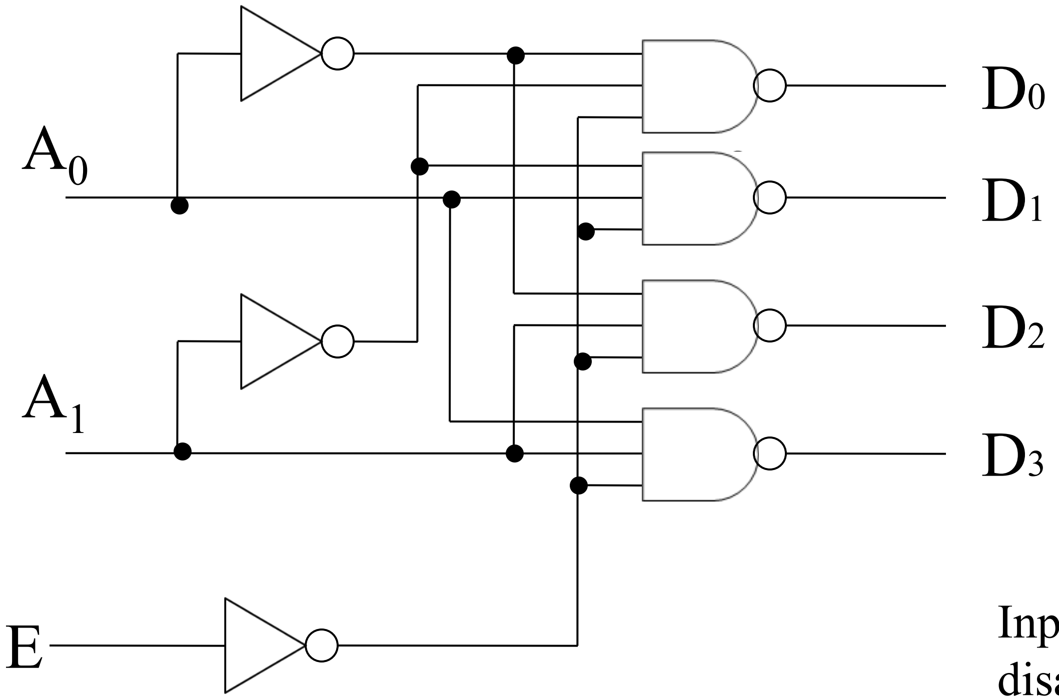
DECODERS

Decoders are of size *n*-to-*m* (*n*x*m*), where $m \leq 2^n$

Diagram of a 3-to-8 Decoder

CS 2613 3

2 x 4 Decoder implementation with NAND



(a) Logic diagram

E	A_1	A_0	D_0	D_1	D_2	D_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1

Inputs disabled

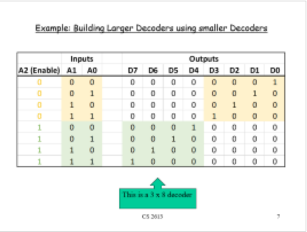
(b) Truth table

Example: Building Larger Decoders using smaller Decoders

	Inputs			Outputs							
A2 (Enable)	A1	A0		D7	D6	D5	D4	D3	D2	D1	D0
0	0	0		0	0	0	0	0	0	0	1
0	0	1		0	0	0	0	0	0	1	0
0	1	0		0	0	0	0	0	1	0	0
0	1	1		0	0	0	0	1	0	0	0
1	0	0		0	0	0	1	0	0	0	0
1	0	1		0	0	1	0	0	0	0	0
1	1	0		0	1	0	0	0	0	0	0
1	1	1		1	0	0	0	0	0	0	0

This is a 3 x 8 decoder

Example:Building Larger Decoders using smaller Decoders (2)

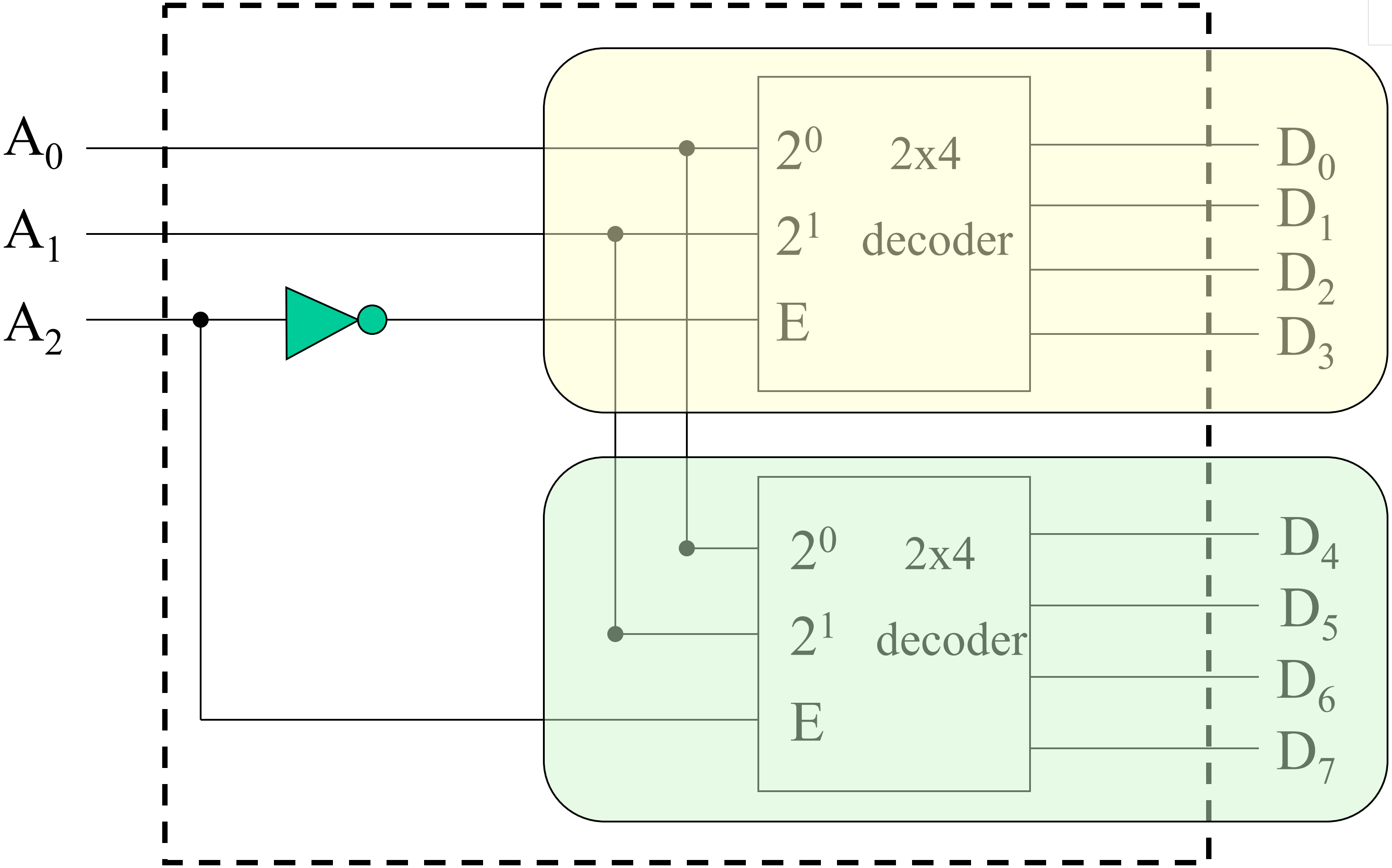


Example: Building Larger Decoders using smaller Decoders

	Inputs			Outputs							
A2 (Enable)	A1	A0		D7	D6	D5	D4	D3	D2	D1	D0
0	0	0		0	0	0	0	0	0	0	1
0	0	1		0	0	0	0	0	0	1	0
0	1	0		0	0	0	0	0	1	0	0
0	1	1		0	0	0	0	1	0	0	0
1	0	0		0	0	0	1	0	0	0	0
1	0	1		0	0	1	0	0	0	0	0
1	1	0		0	1	0	0	0	0	0	0
1	1	1		1	0	0	0	0	0	0	0

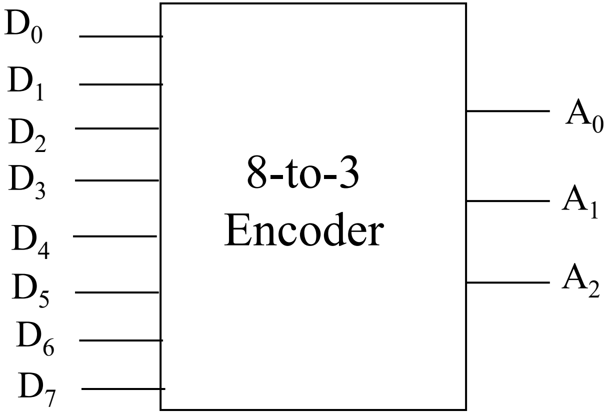
This is a 3 x 8 decoder

CS 2613 7



ENCODERS

Perform the inverse operation of a decoder.



Inputs								Outputs		
D7	D6	D5	D4	D3	D2	D1	D0	A2	A1	A0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Only one input can have logic value 1 at any time. Otherwise, operation of circuit not defined.

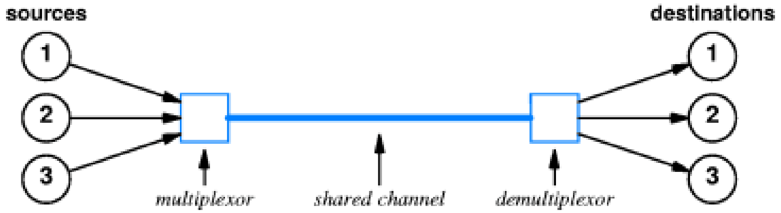
$$A_0 = D_1 + D_3 + D_5 + D_7$$

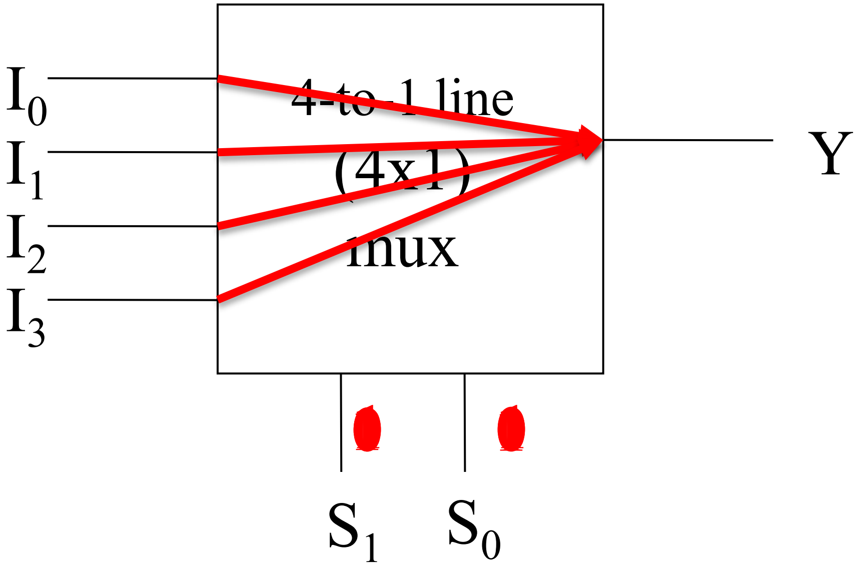
$$A_1 = D_2 + D_3 + D_6 + D_7$$

$$A_2 = D_4 + D_5 + D_6 + D_7$$

MULTIPLEXER

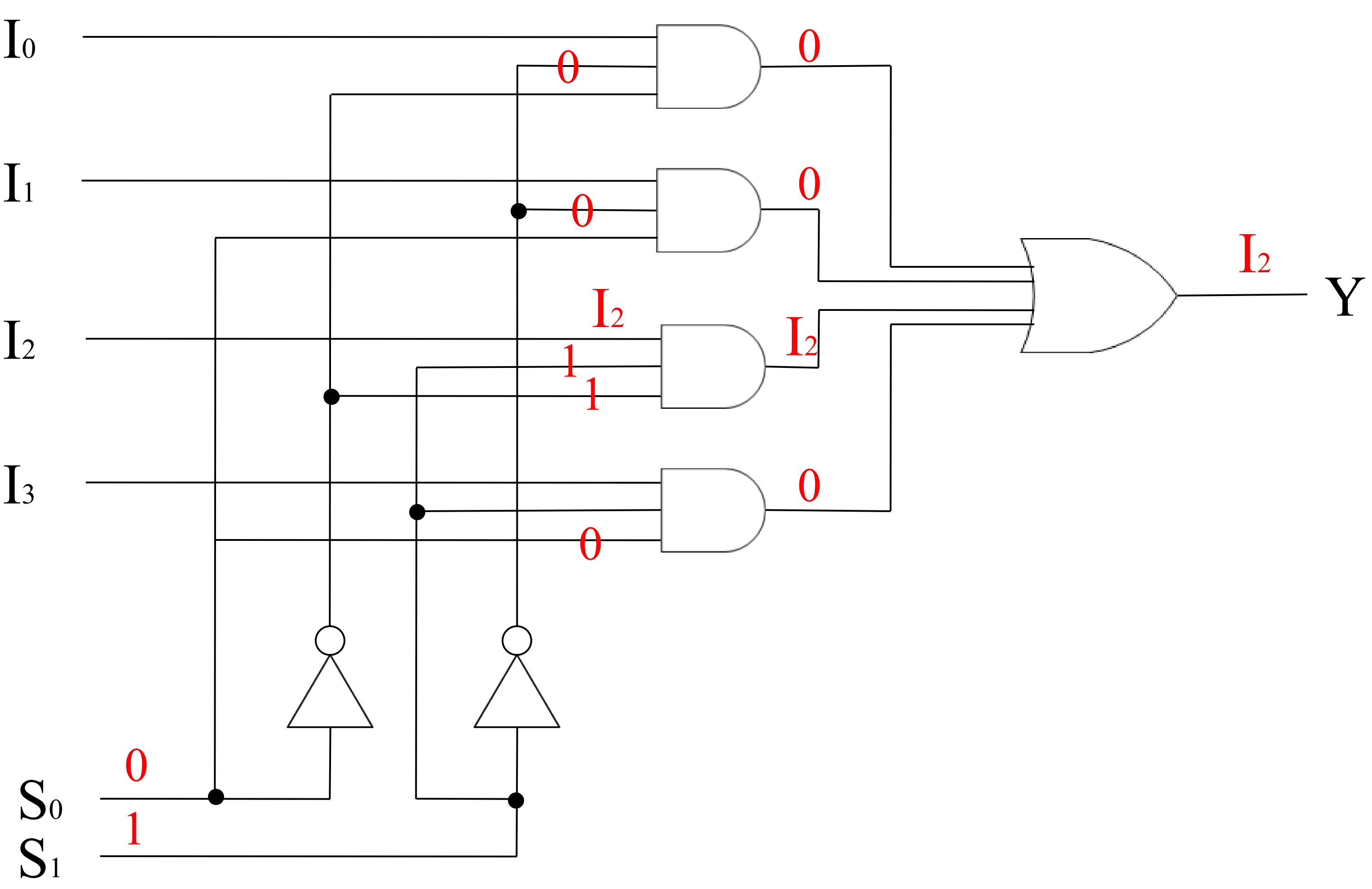
Directs one of 2^n input lines to a single output line using *n* selection lines.





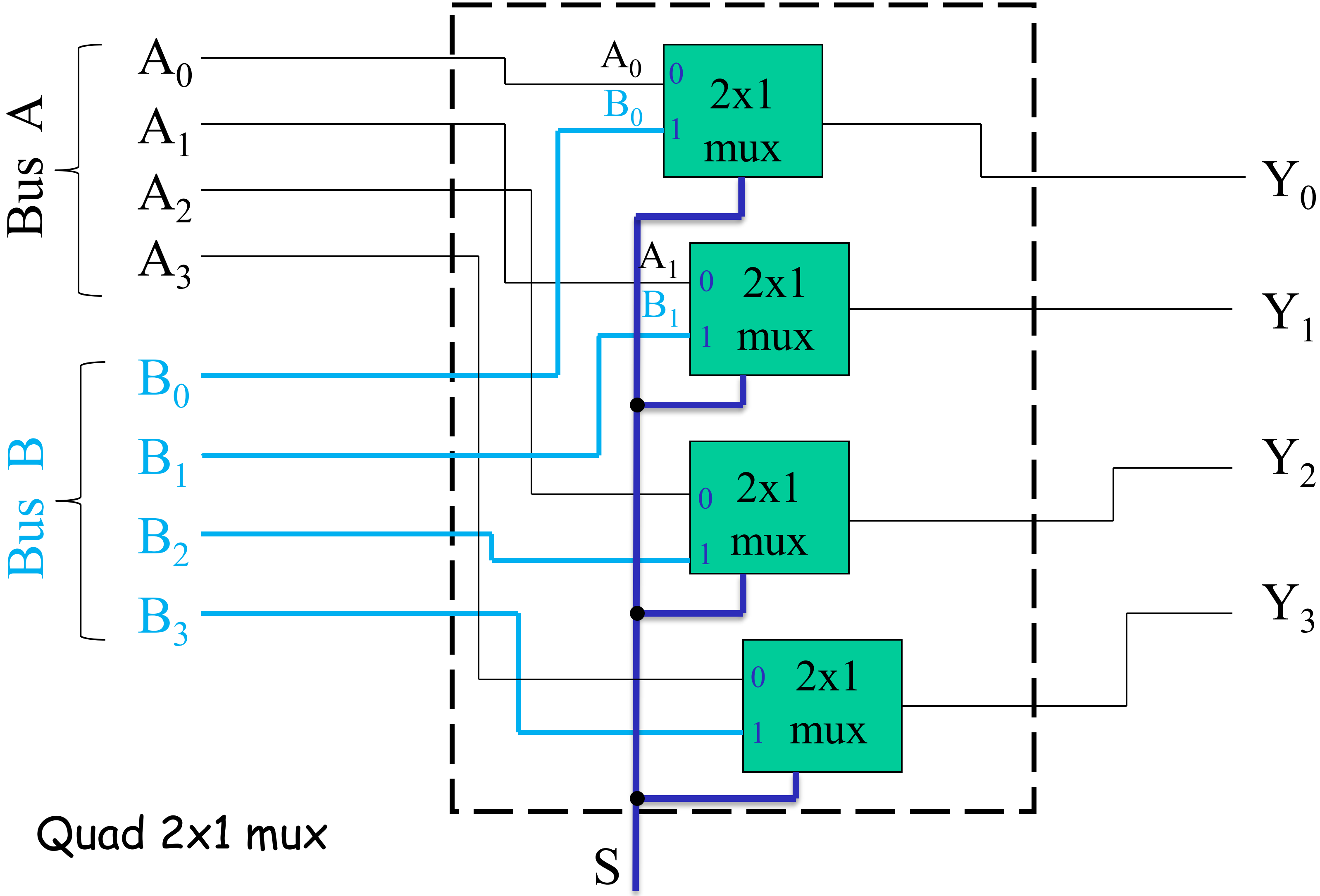
Select		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

4-to-1 line MUX Implementation



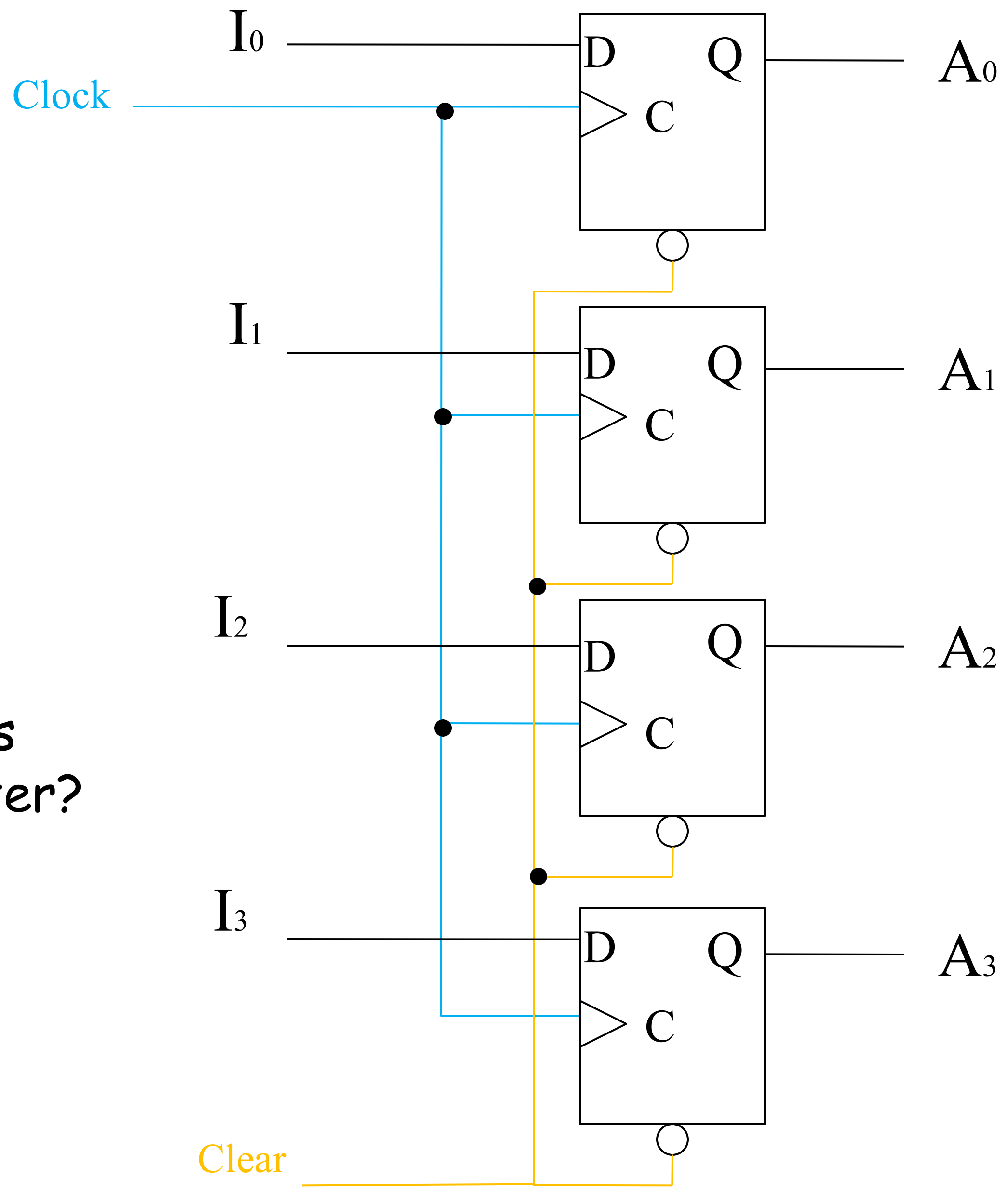
Example: Quad 2x1 mux.

How to construct a device that will select one of two 4-bit data lines (Buses).



Quad 2x1 mux

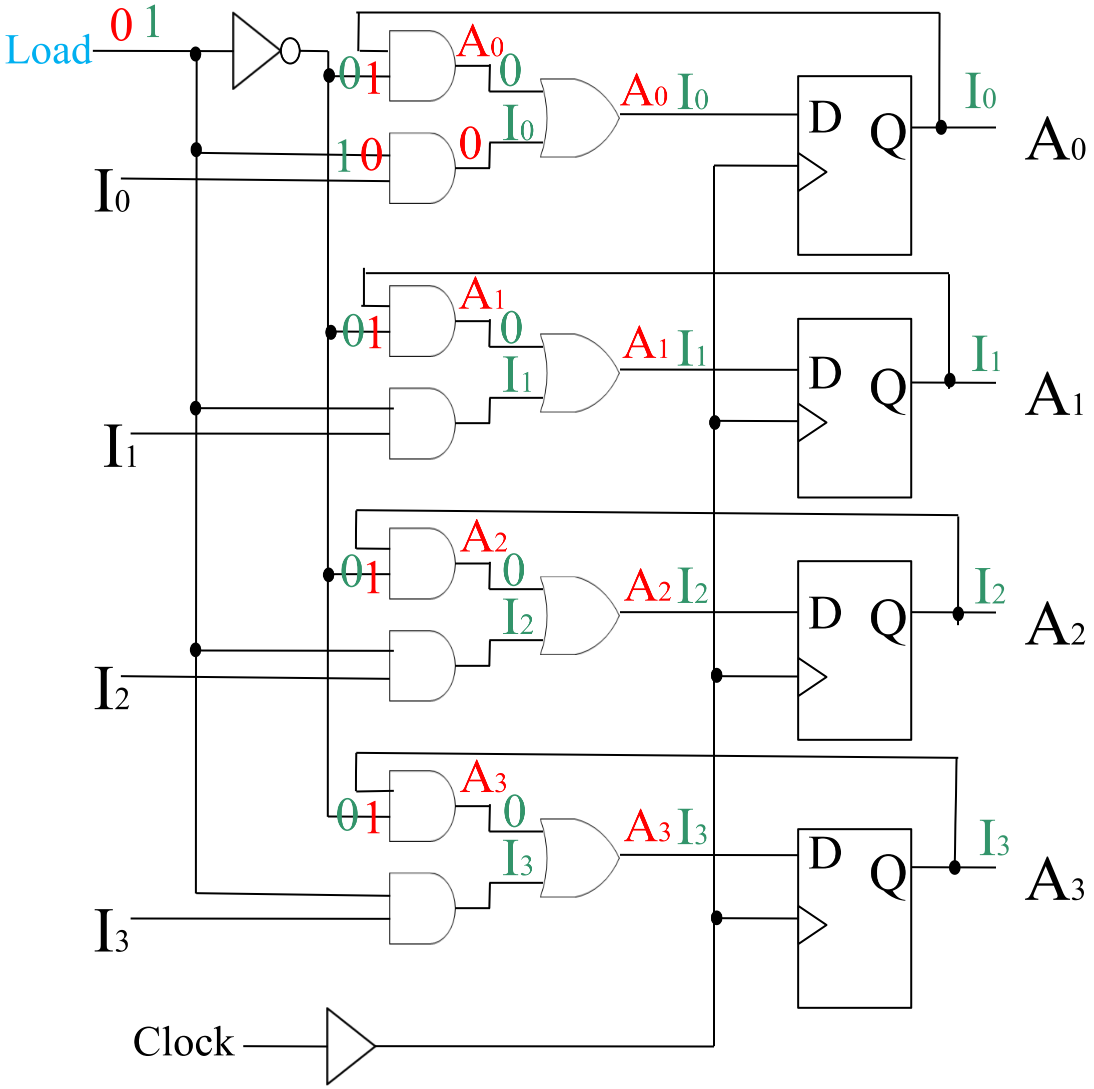
REGISTER



Q: Under what conditions is data loaded into this register?

A: Data is loaded at every rising edge of the clock.

4-bit register with parallel load



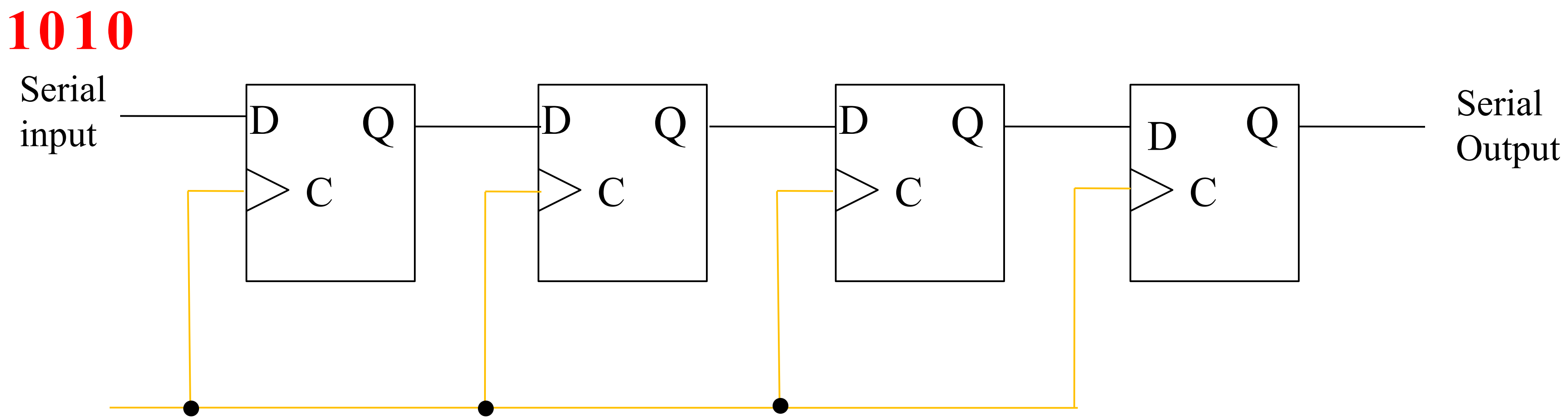
Q: Under what conditions is input data loaded into this register?

A: Input data is loaded when load line is high (on next rising edge).

Q: Why is the feedback required?

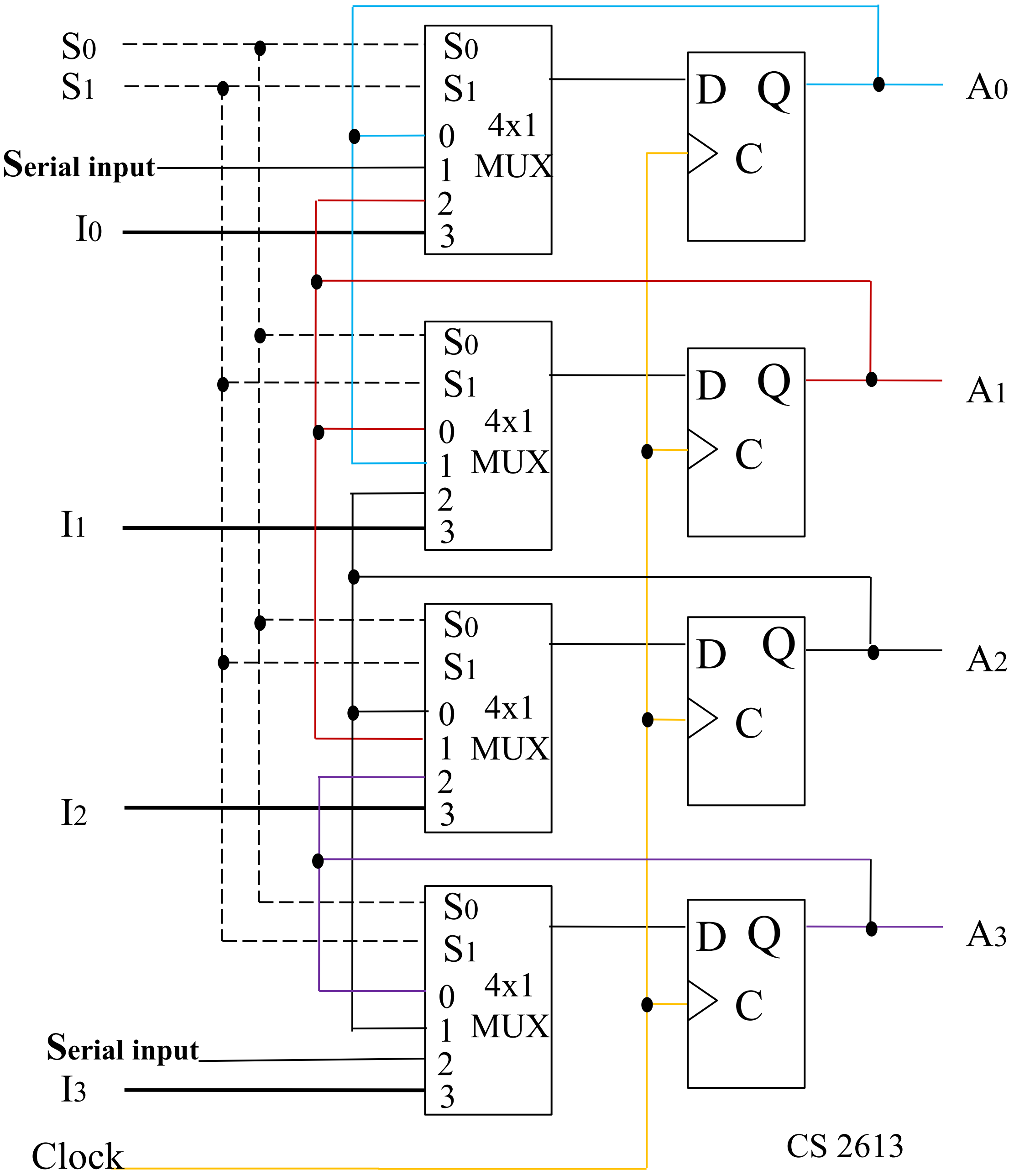
A: D Flip Flops do not have a "no change" input.

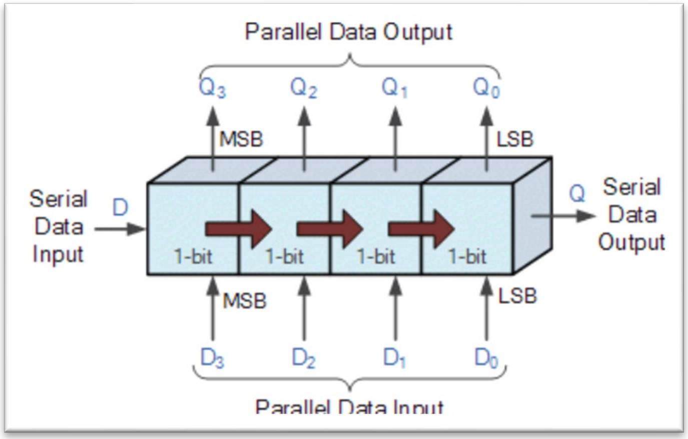
Shift Registers



4-bit shift register

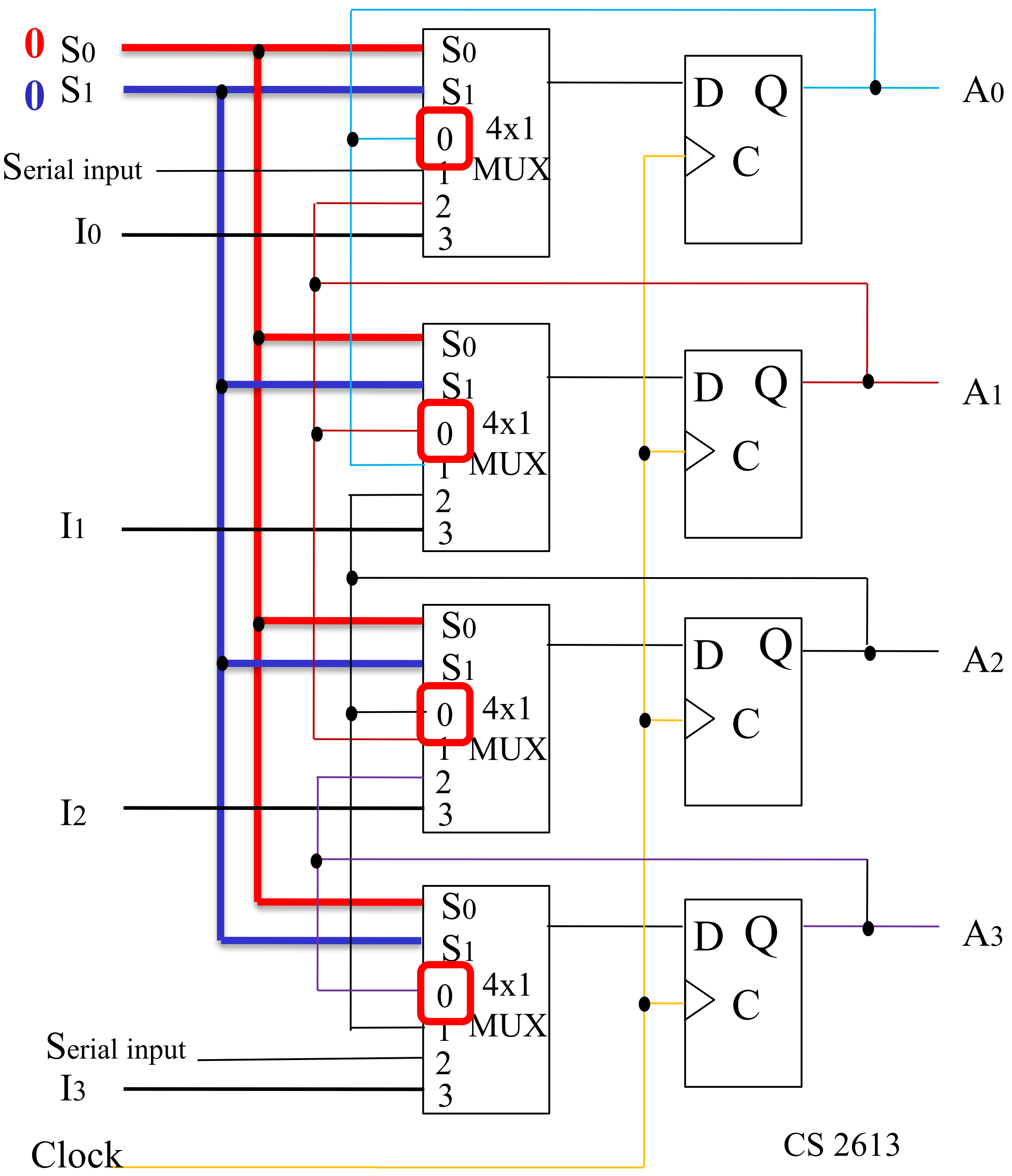
Bidirectional shift register with parallel load

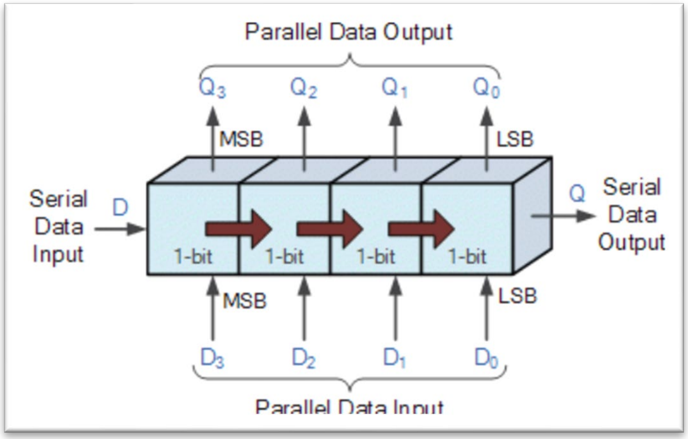




S1	S0	Register operation
0	0	No change
0	1	Shift right (down in figure)
1	0	Shift left (up in figure)
1	1	Parallel load

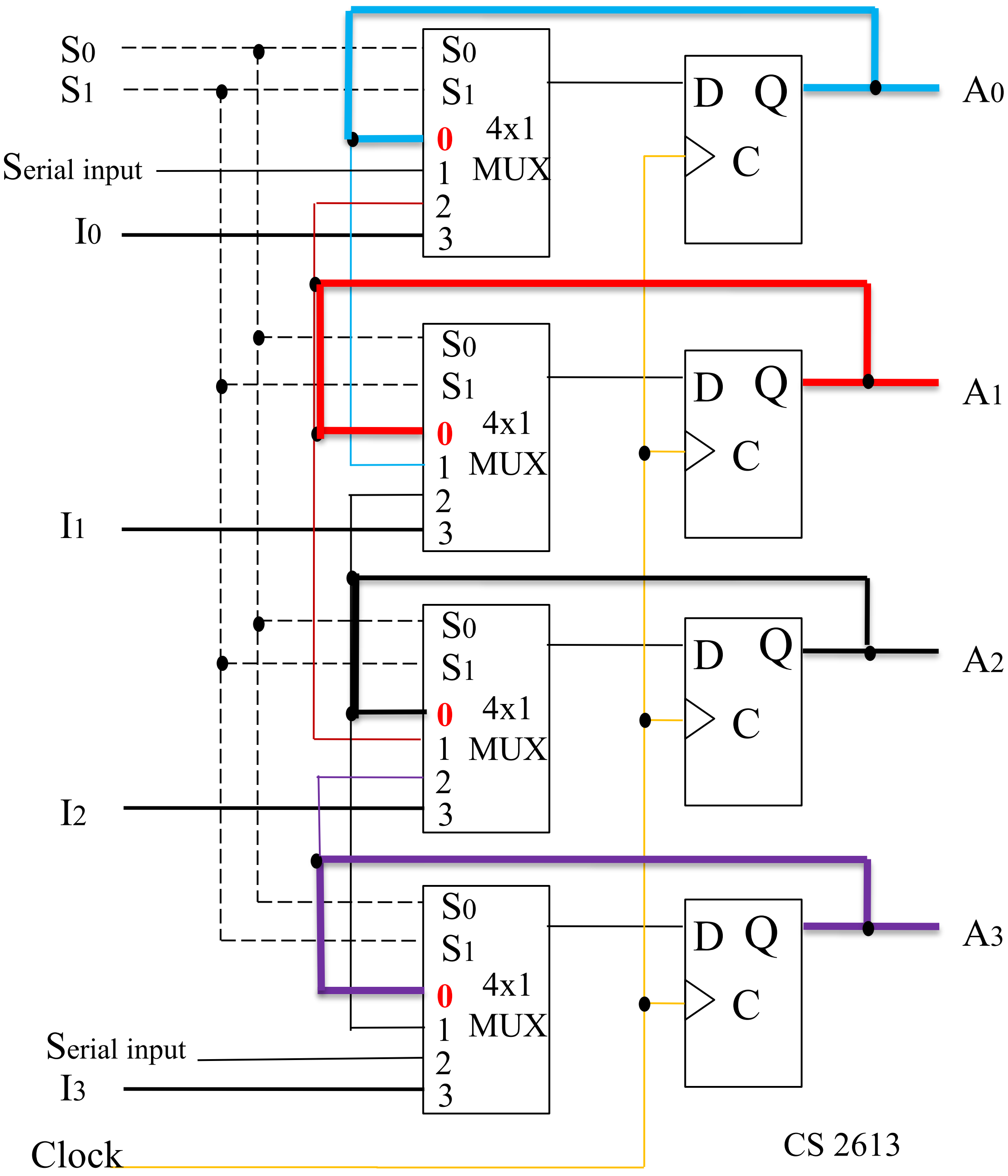
Bidirectional shift register with parallel load

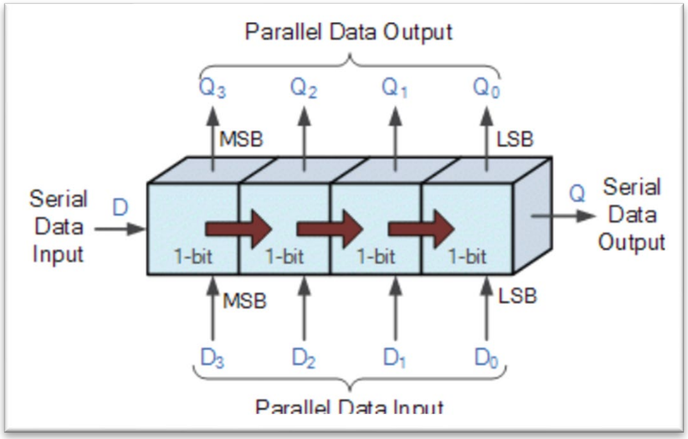




S1	S0	Register operation
0	0	No change
0	1	Shift right (down in figure)
1	0	Shift left (up in figure)
1	1	Parallel load

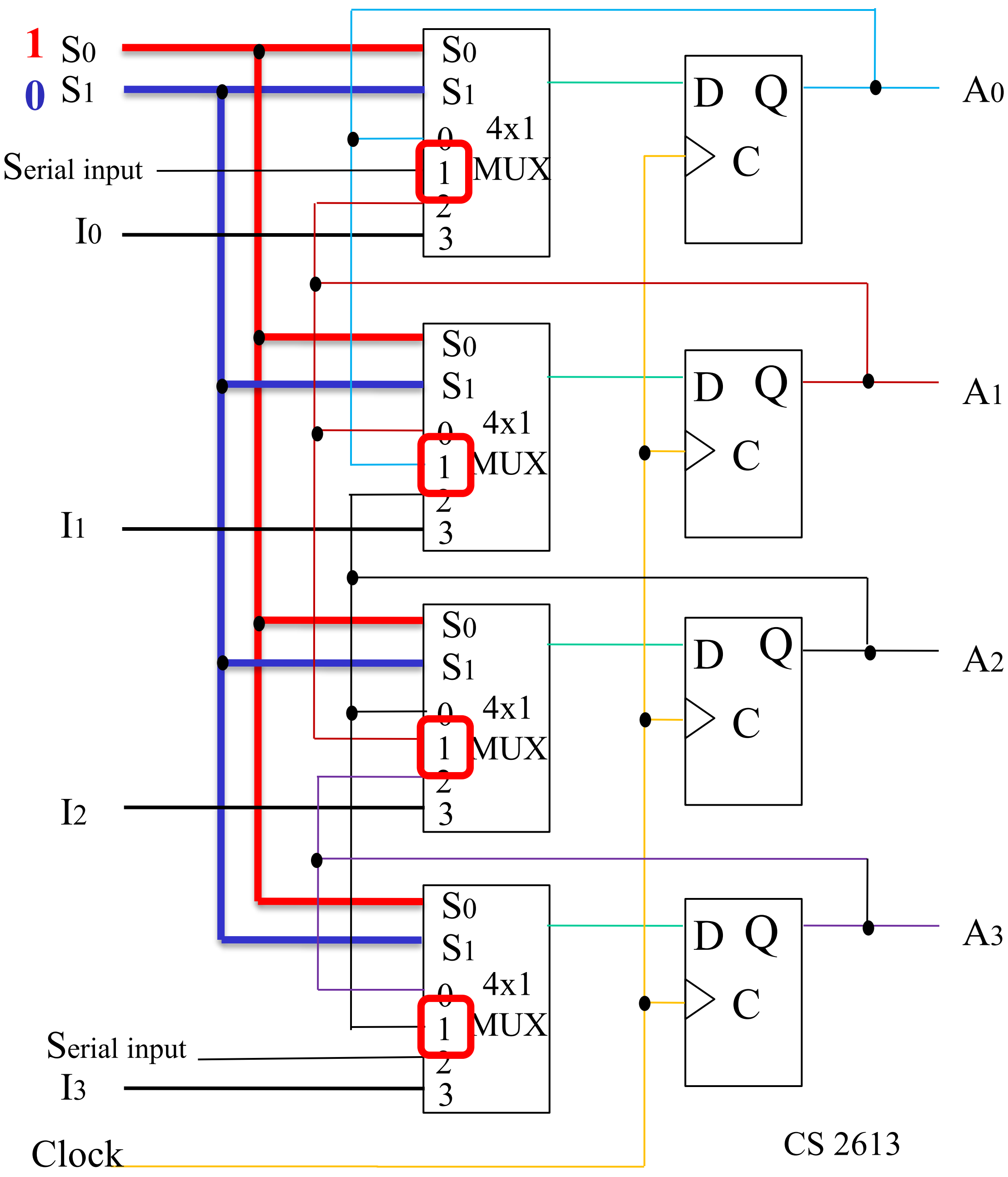
Bidirectional shift register with parallel load

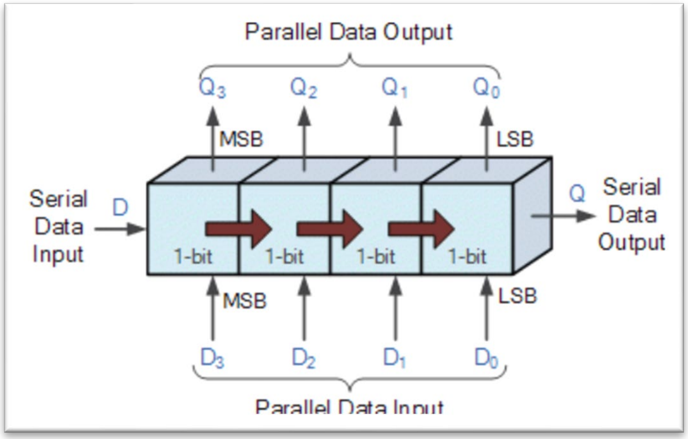




S1	S0	Register operation
0	0	No change
0	1	Shift right (down in figure)
1	0	Shift left (up in figure)
1	1	Parallel load

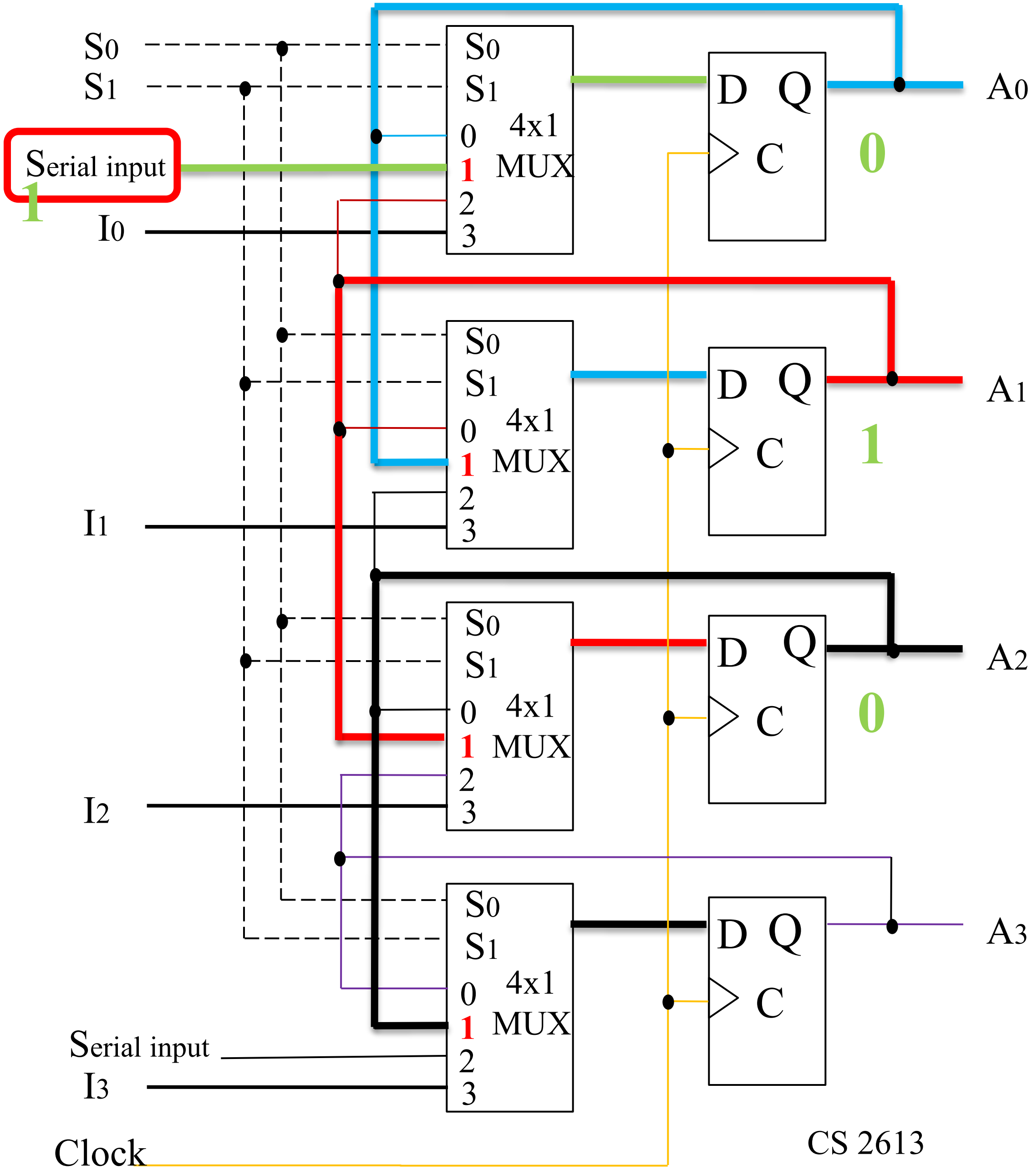
Bidirectional shift register with parallel load

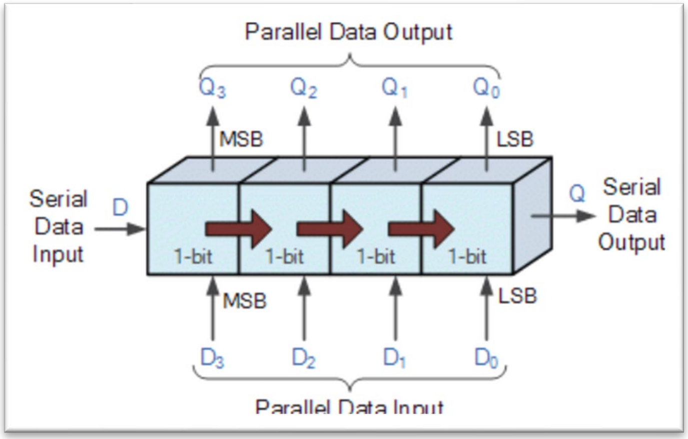




S1	S0	Register operation
0	0	No change
0	1	Shift right (down in figure)
1	0	Shift left (up in figure)
1	1	Parallel load

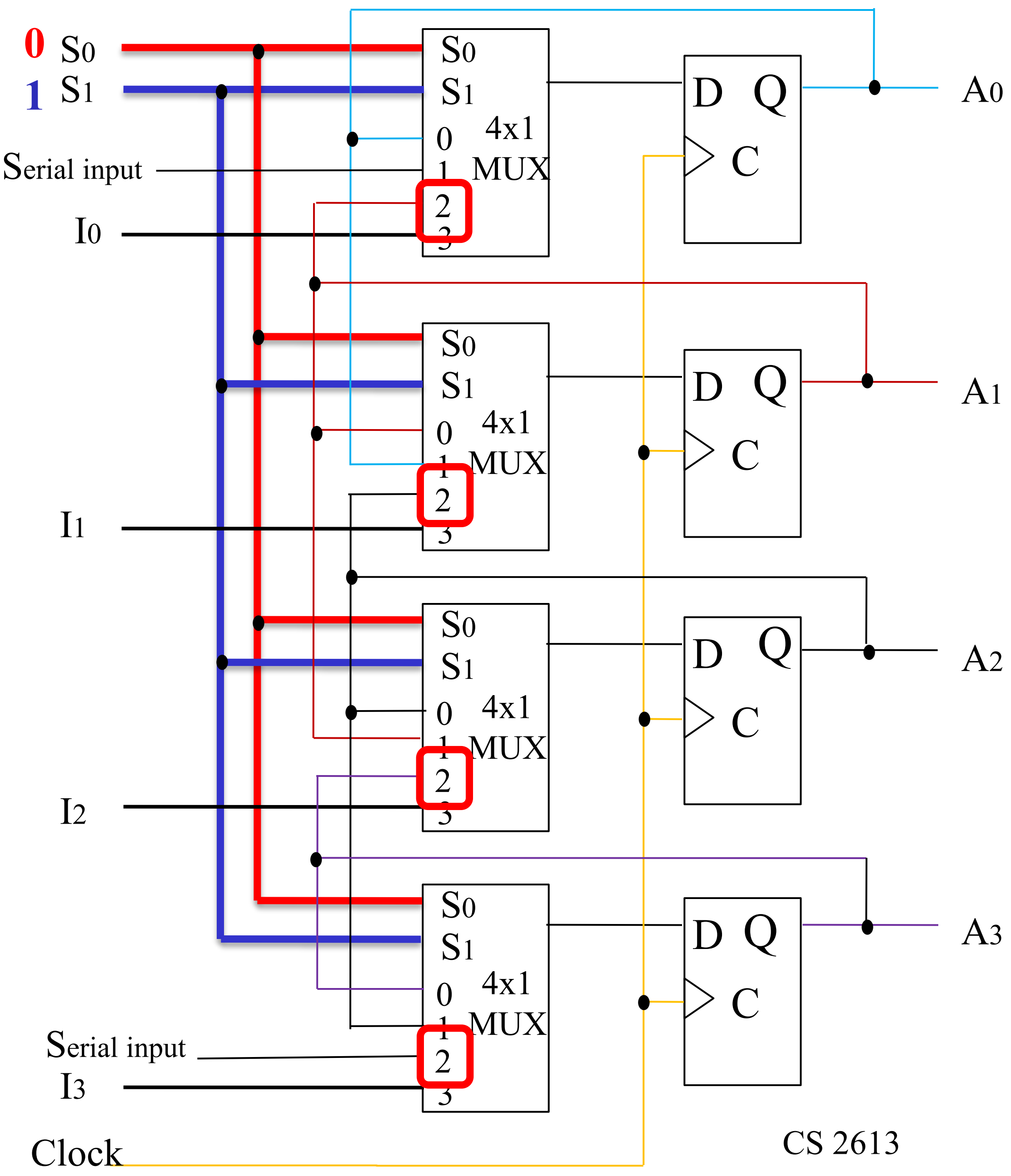
Bidirectional shift register with parallel load

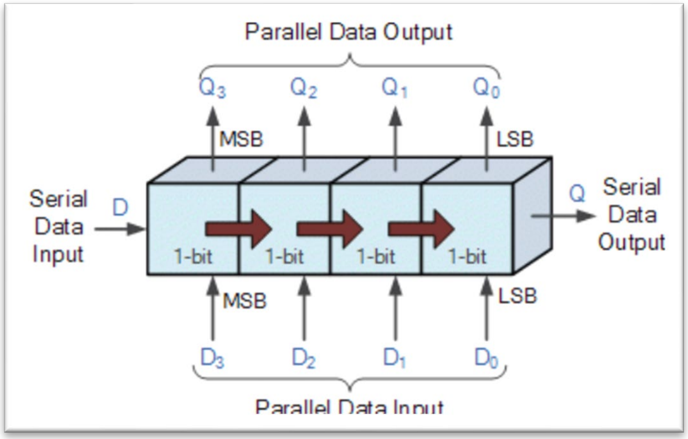




S1	S0	Register operation
0	0	No change
0	1	Shift right (down in figure)
1	0	Shift left (up in figure)
1	1	Parallel load

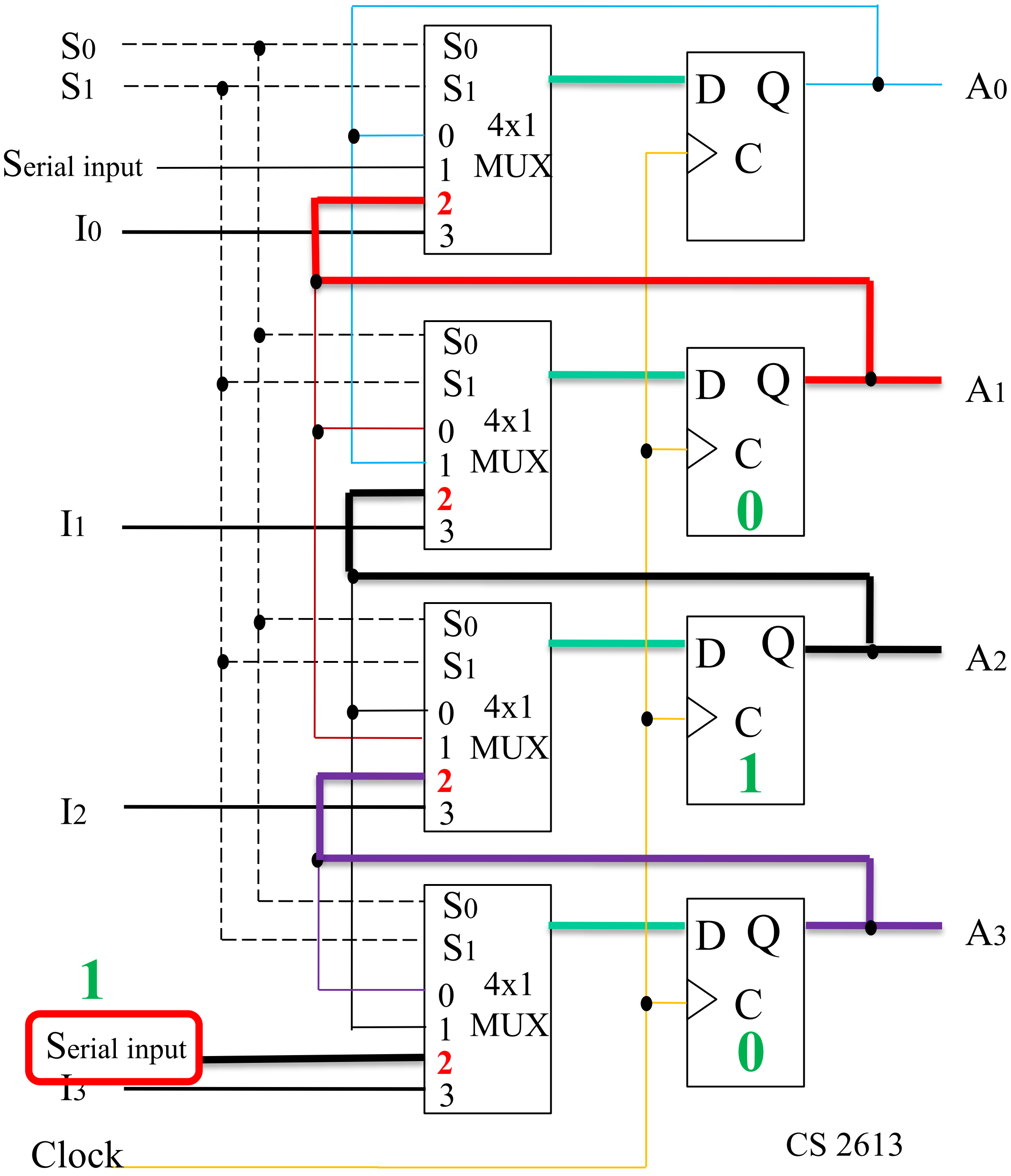
Bidirectional shift register with parallel load

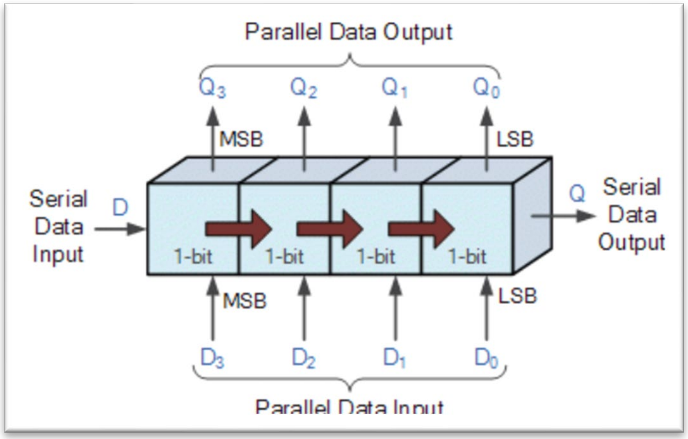




S1	S0	Register operation
0	0	No change
0	1	Shift right (down in figure)
1	0	Shift left (up in figure)
1	1	Parallel load

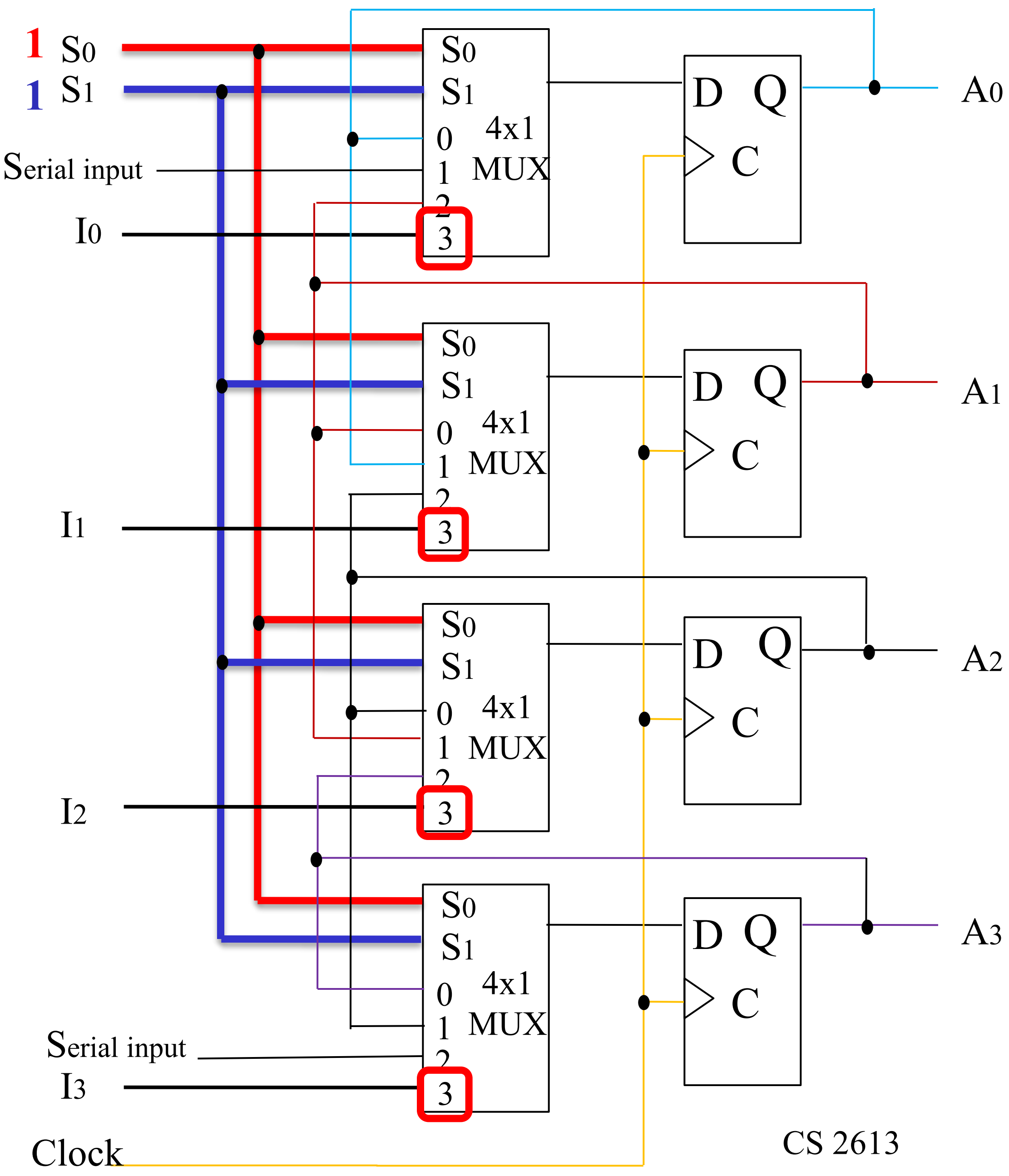
Bidirectional shift register with parallel load

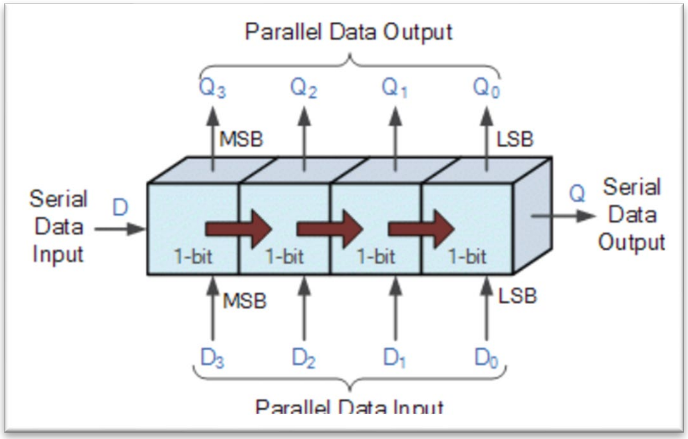




S1	S0	Register operation
0	0	No change
0	1	Shift right (down in figure)
1	0	Shift left (up in figure)
1	1	Parallel load

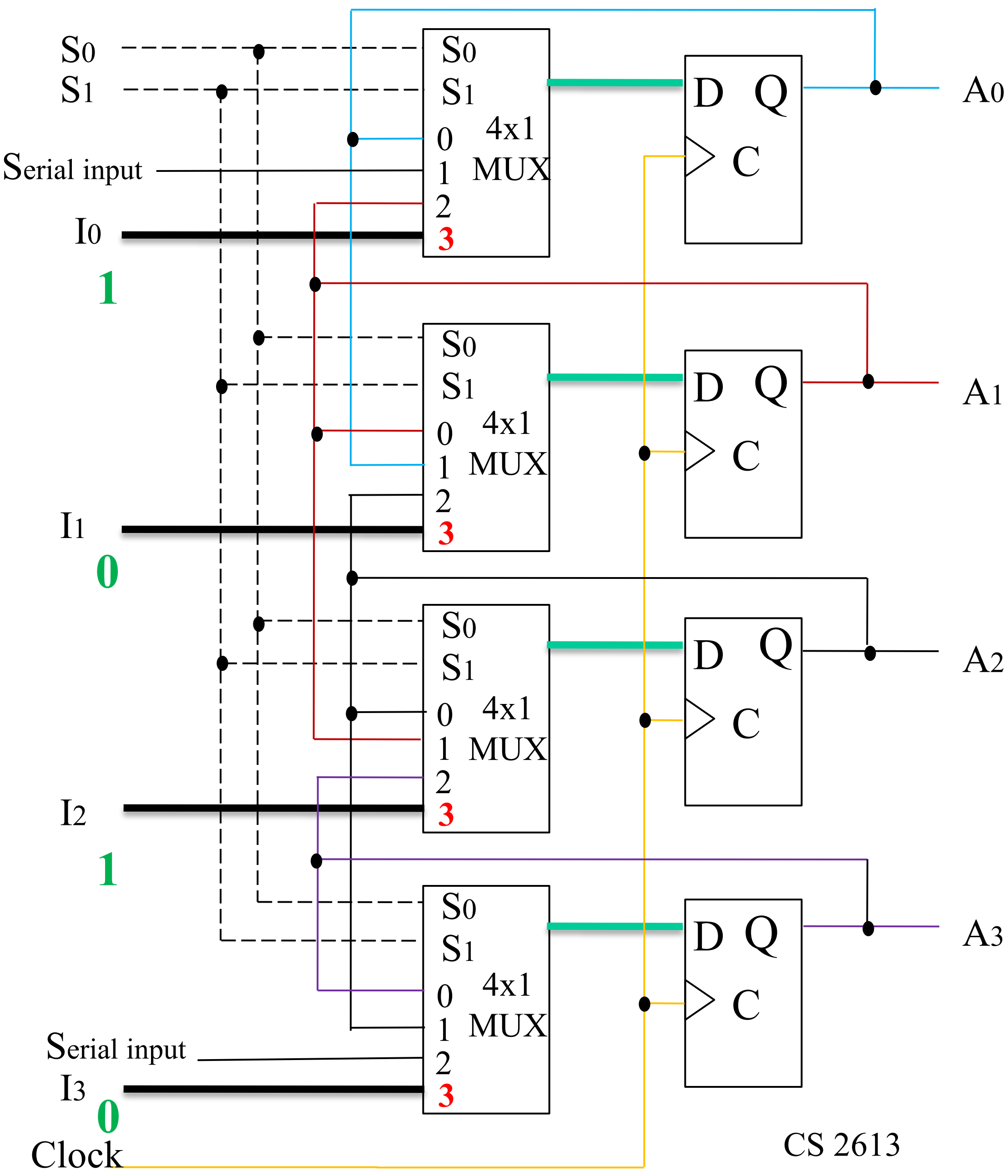
Bidirectional shift register with parallel load

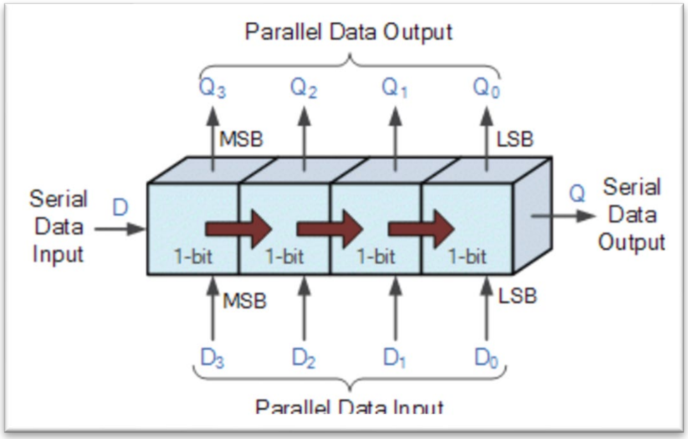




S1	S0	Register operation
0	0	No change
0	1	Shift right (down in figure)
1	0	Shift left (up in figure)
1	1	Parallel load

Bidirectional shift register with parallel load

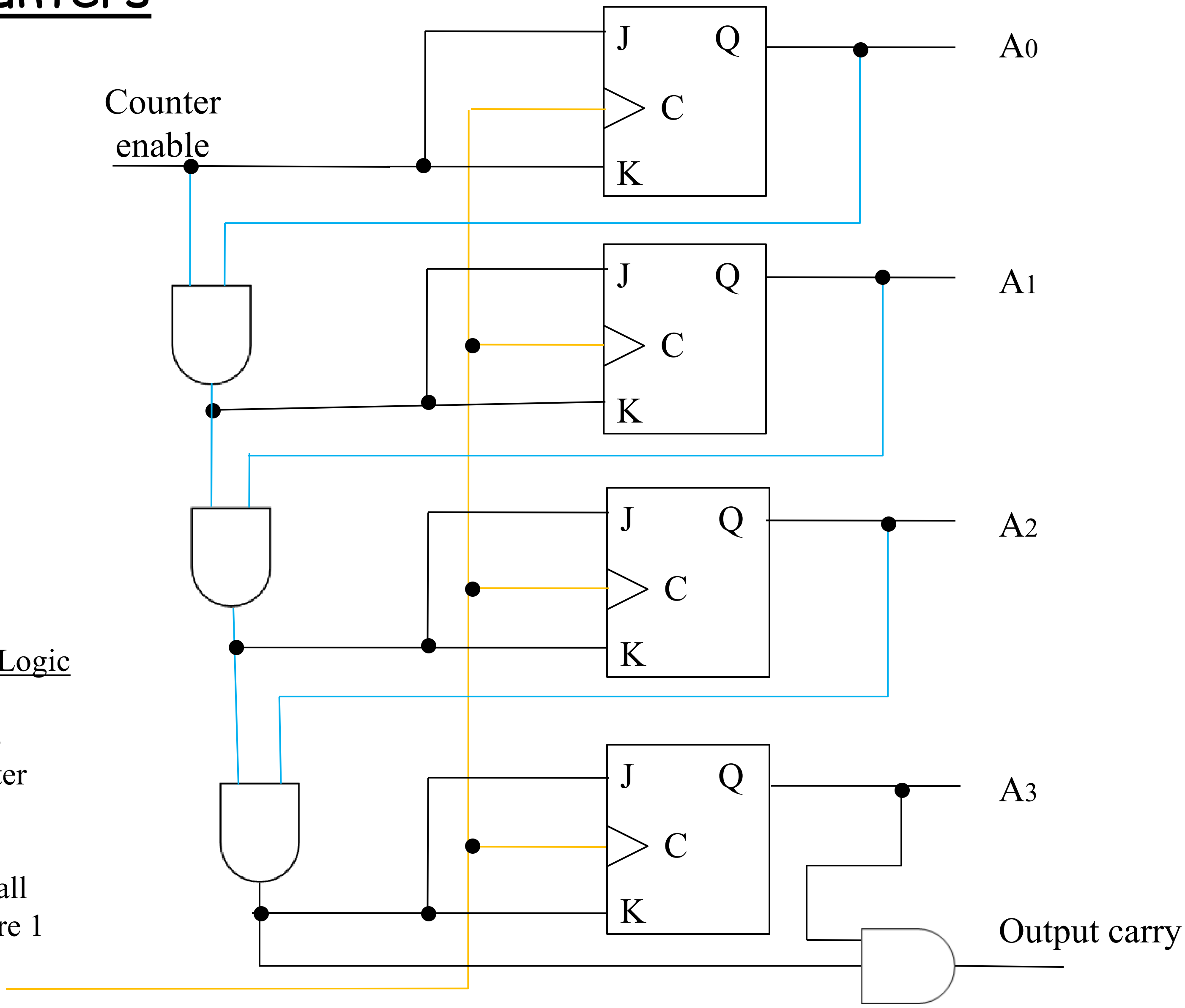




S1	S0	Register operation
0	0	No change
0	1	Shift right (down in figure)
1	0	Shift left (up in figure)
1	1	Parallel load

Binary Counters

A3	A2	A1	A0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

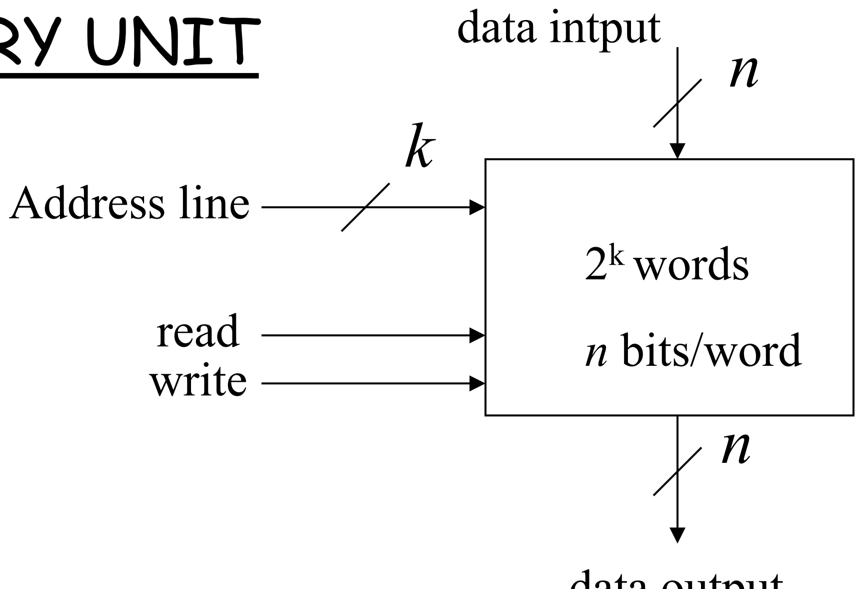


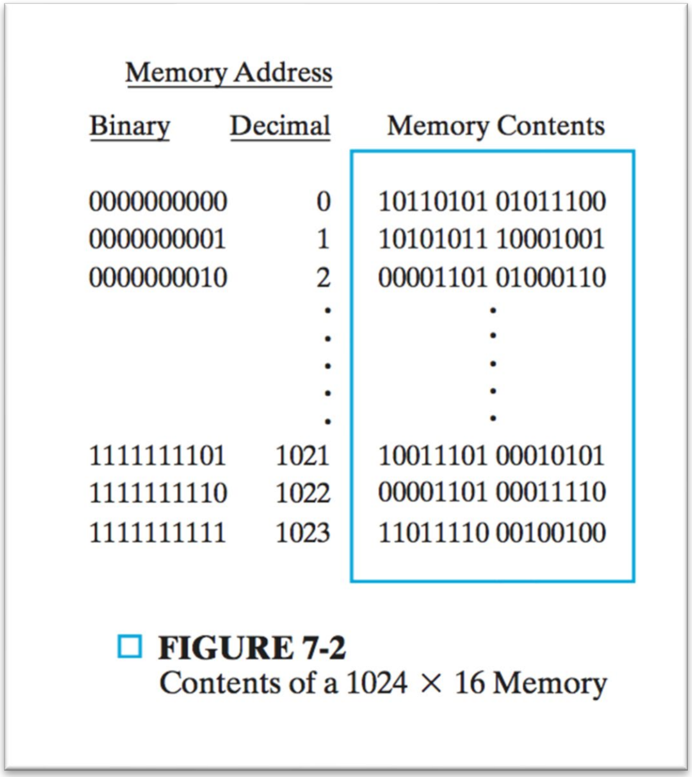
Counter Increment Logic

1. Lower order bit is complemented after every count
2. Other bits are complemented if all lower order bits are 1

4-bit synchronous binary counter

MEMORY UNIT





Memory Address		Memory Contents
Binary	Decimal	
0000000000	0	10110101 01011100
0000000001	1	10101011 10001001
0000000010	2	00001101 01000110
	⋮	⋮
1111111101	1021	10011101 00010101
1111111110	1022	00001101 00011110
1111111111	1023	11011110 00100100

□ **FIGURE 7-2**
Contents of a 1024 × 16 Memory

Steps to write data to the MU

1. Apply desired address location on address lines.
2. Apply data to data input lines.
3. Activate write.

Steps to read data from the MU

1. Apply address of desired word.
2. Activate read.
3. Data output valid after specified (access) time.

RAM – Random Access Memory

ROM - like RAM with no write capability. Stored information is permanent - really just a truth table.

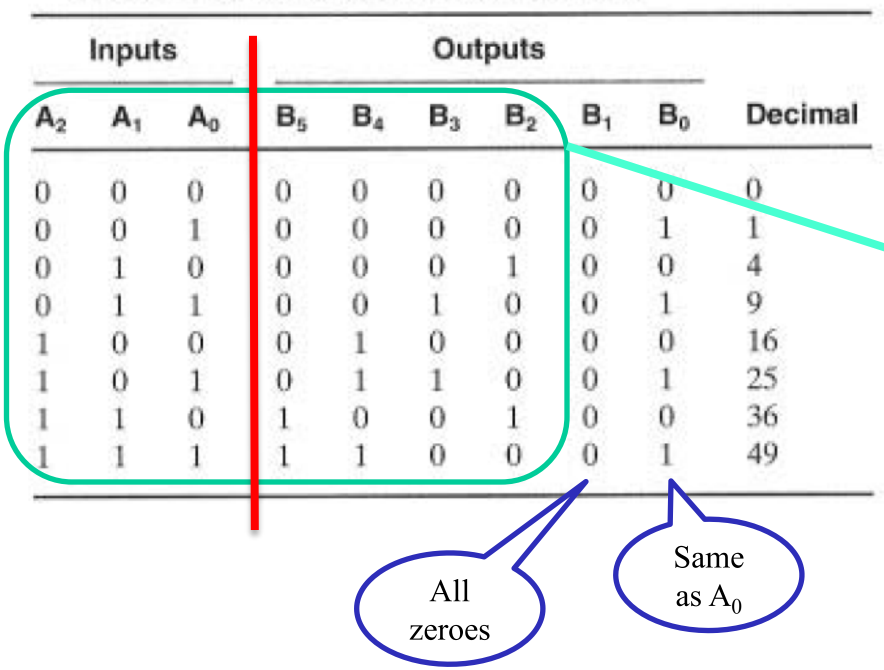
PROM – Programmable ROM

EEPROM- Electrically Erasable PROM

Implementing Combinational Circuit using ROM

Design a hardware that accepts a 3-bit number and generates an output binary number equal to the square of the input number. Design the circuit using a ROM.

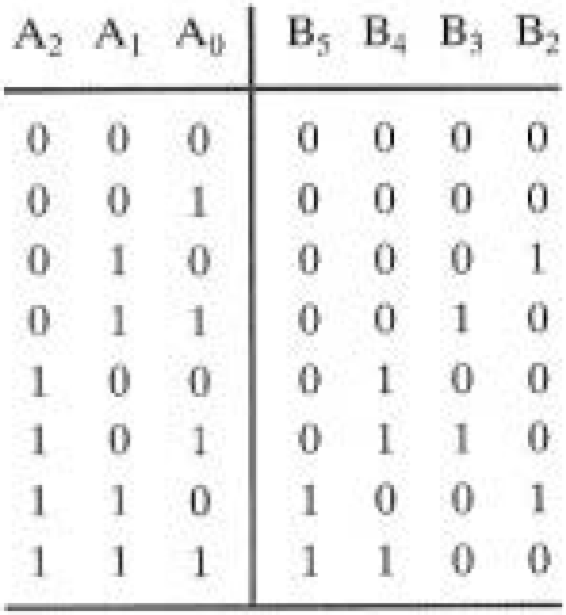
Truth Table



Inputs			Outputs						
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

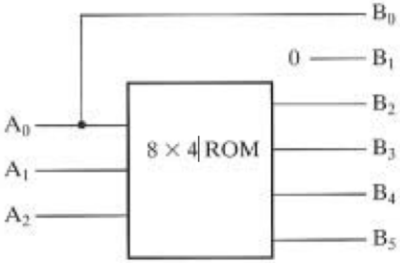
All zeroes
Same as A_0

ROM Contents



A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

ROM Block Diagram



B_0
0
B_1
B_2
B_3
B_4
B_5
A_0
A_1
A_2
8 × 4 ROM