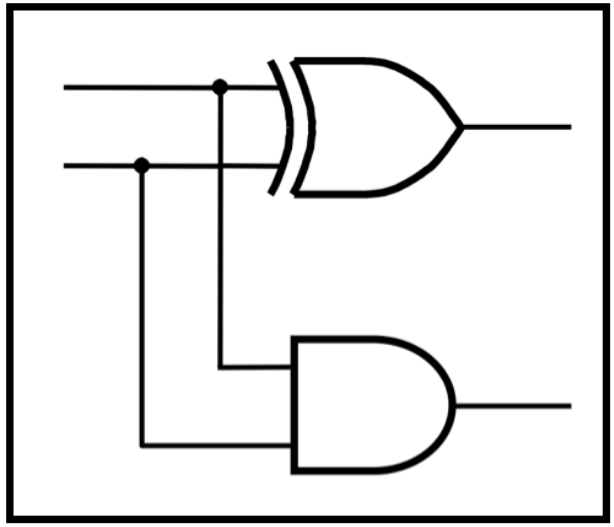


CprE 281: Digital Logic

Instructor: Alexander Stoytchev

http://www.ece.iastate.edu/~alexs/classes/

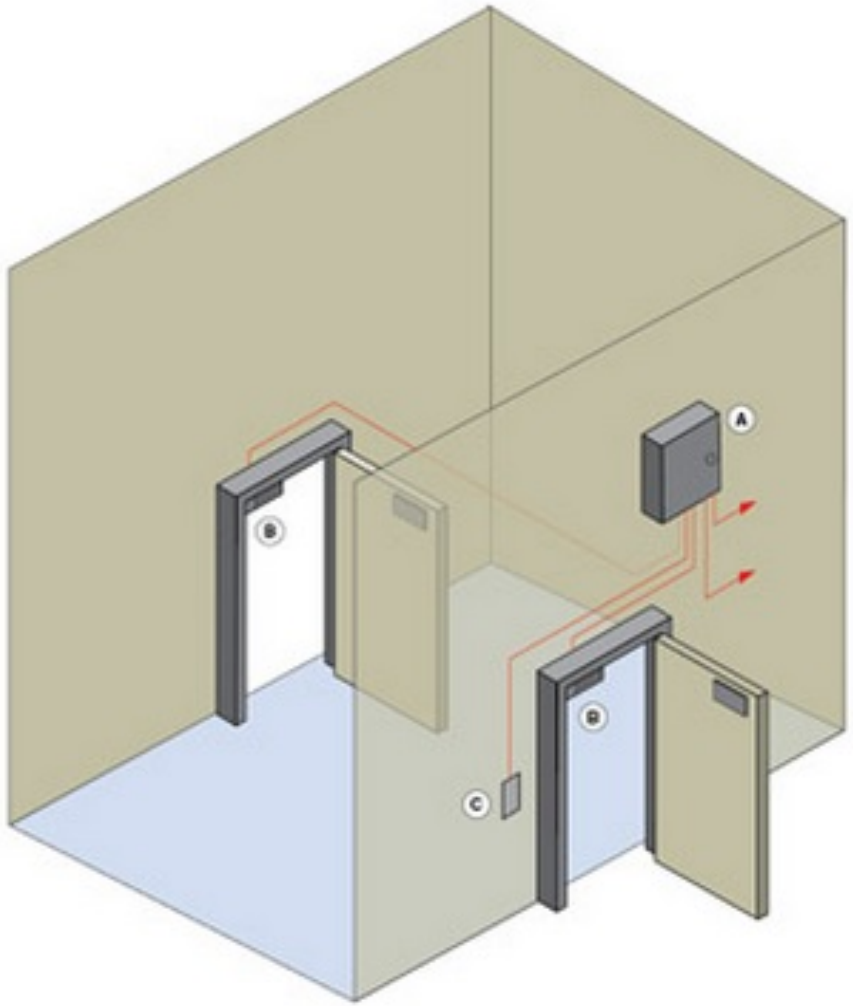
Registers and Counters

CprE 281: Digital Logic
Iowa State University, Ames, IA

Flip-Flop Analogy

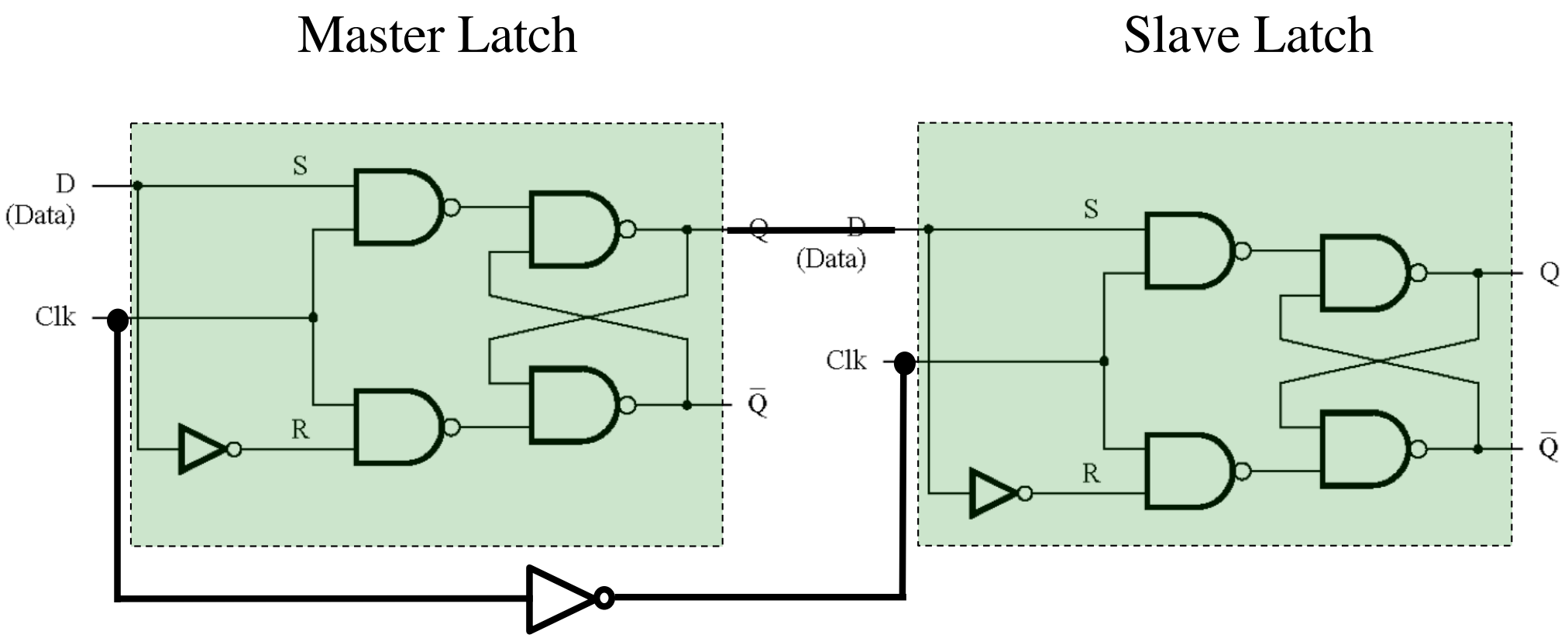
(double door)

D Flip-Flop Analogy

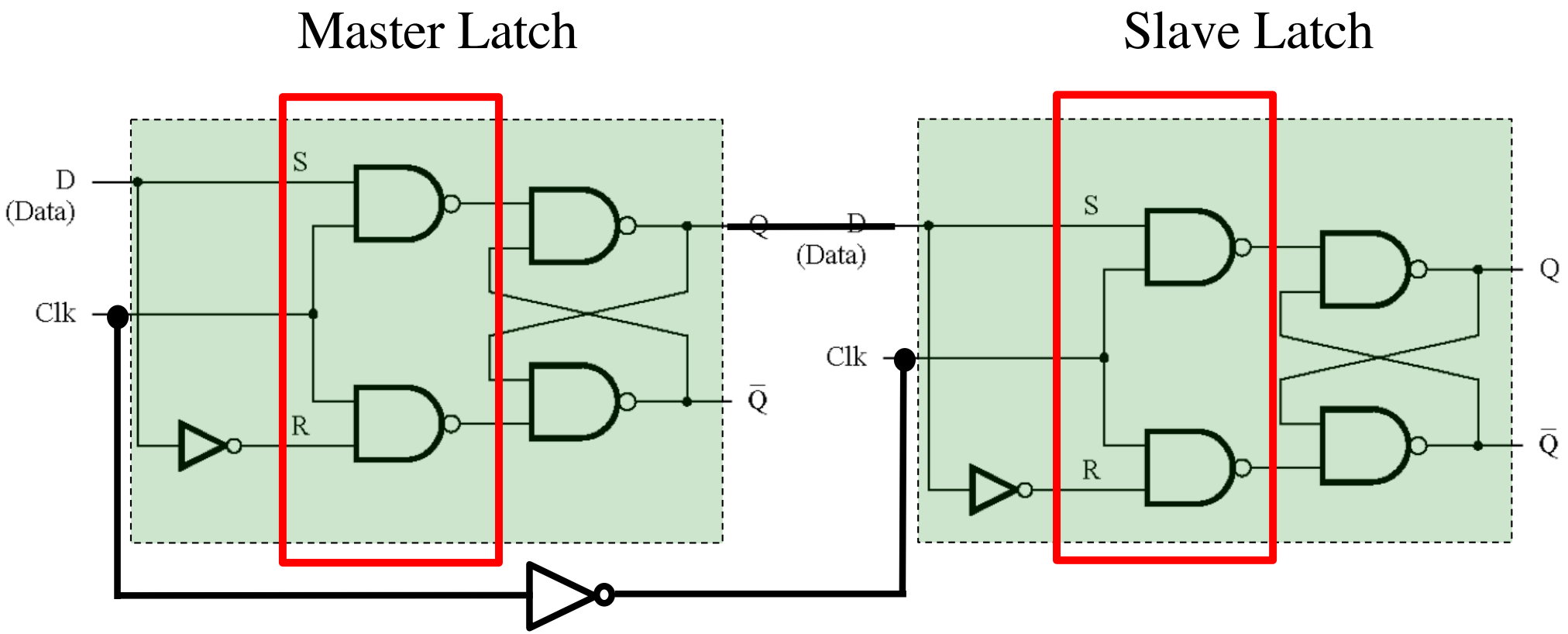


[https://www.maglocks.com/sdc-two-door-airlockmantrapcleanroom-level-1-kit.html]

D Flip-Flop Analogy



D Flip-Flop Analogy

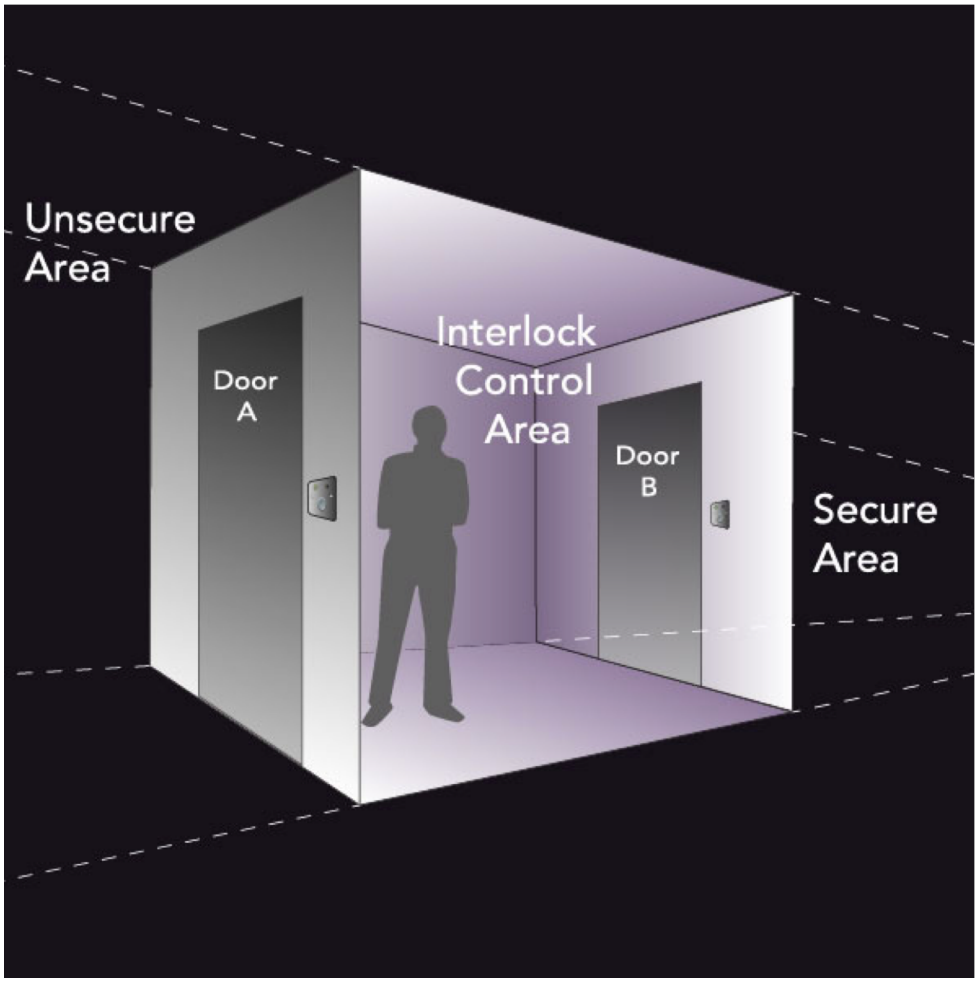


D Flip-Flop Analogy

Outer Door
Will Not Unlock When
Inner Door is Open

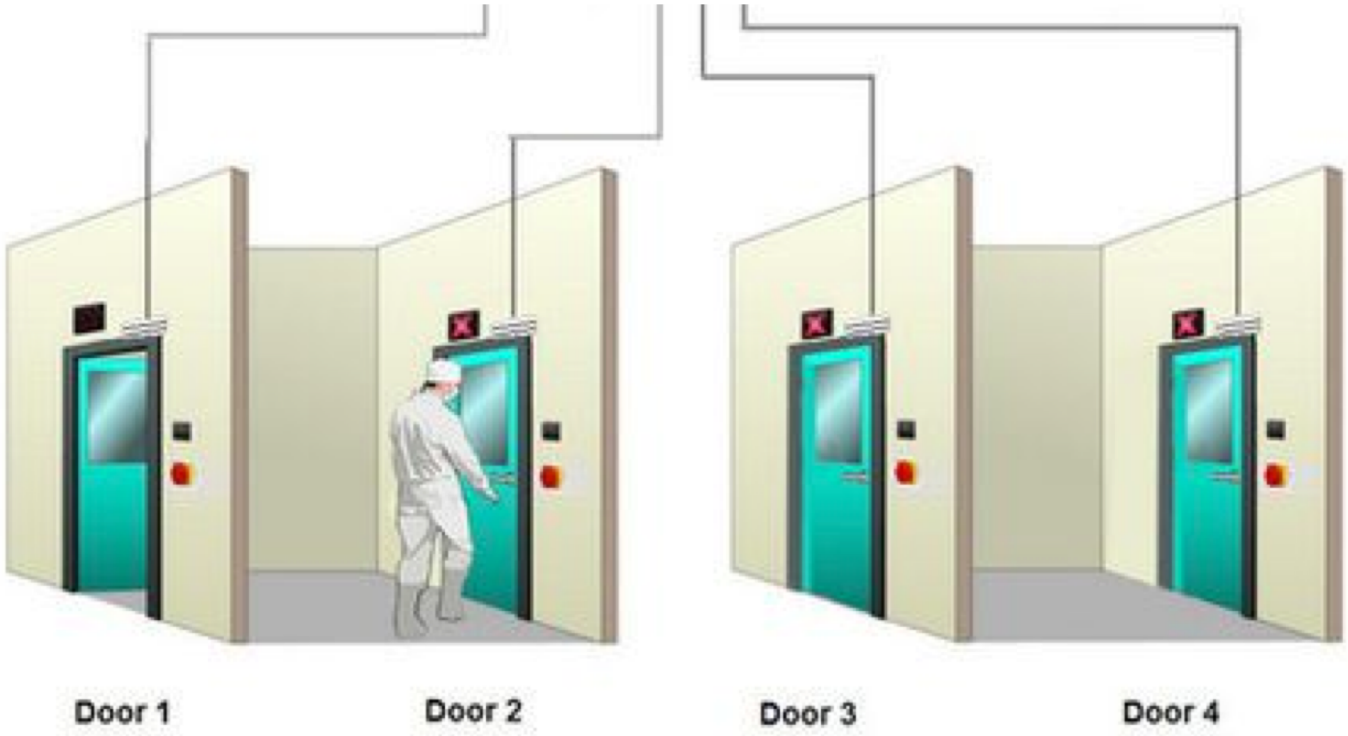
Inner Door
Will Not Unlock When
OuterDoor is Open

[https://www.nortechcontrol.com/solutions/people/door-interlock/]



[https://www.insafe.com/activa-pro-interlock/]

Shift-Register Analogy



[https://tiimg.tistatic.com/fp/1/004/777/door-interlock-controllers-by-radix-566.jpg]

Registers

Register
(Definition)

An n-bit structure consisting of flip-flops

Flip-Flop

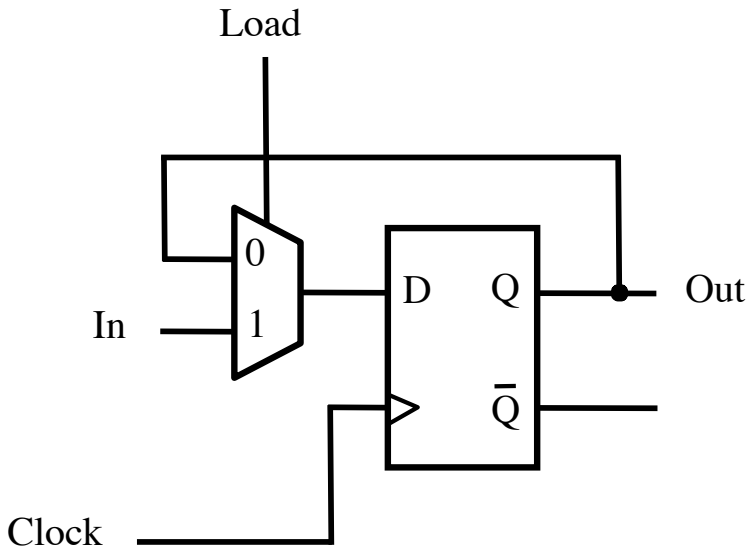
[https://www.ebay.com/itm/223956811053?chn=ps&mkevt=1&mkcid=28&var=522710005842]

4-bit Register

8-bit Register

Parallel-Access Register

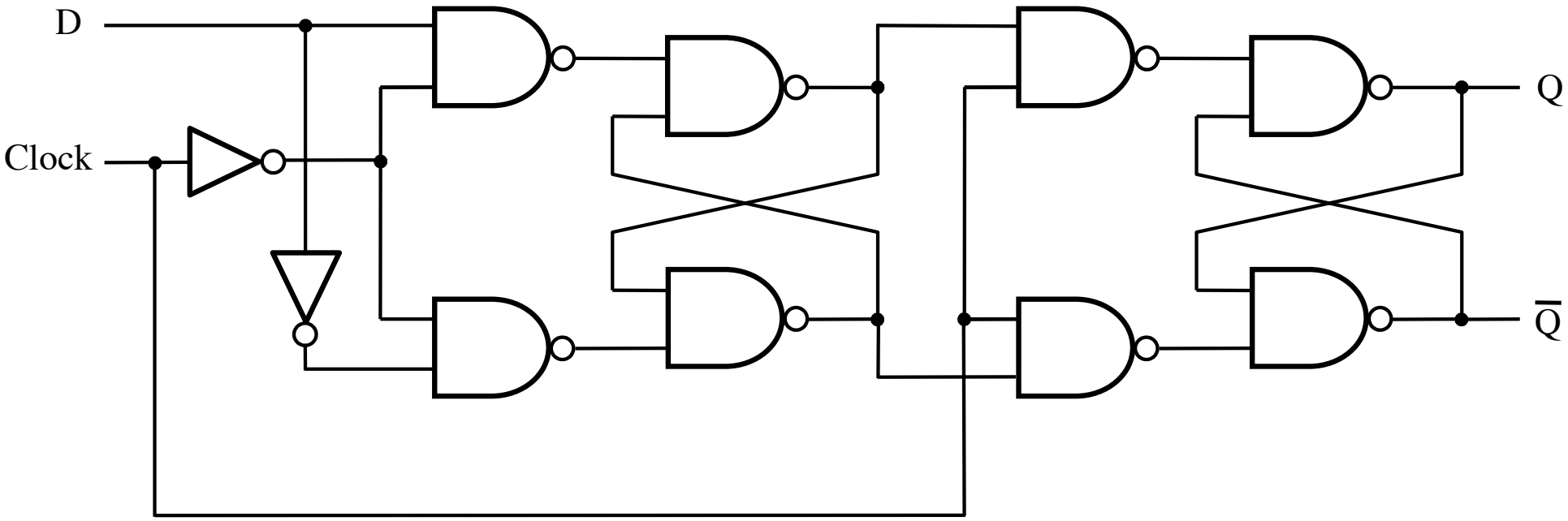
1-Bit Parallel-Access Register



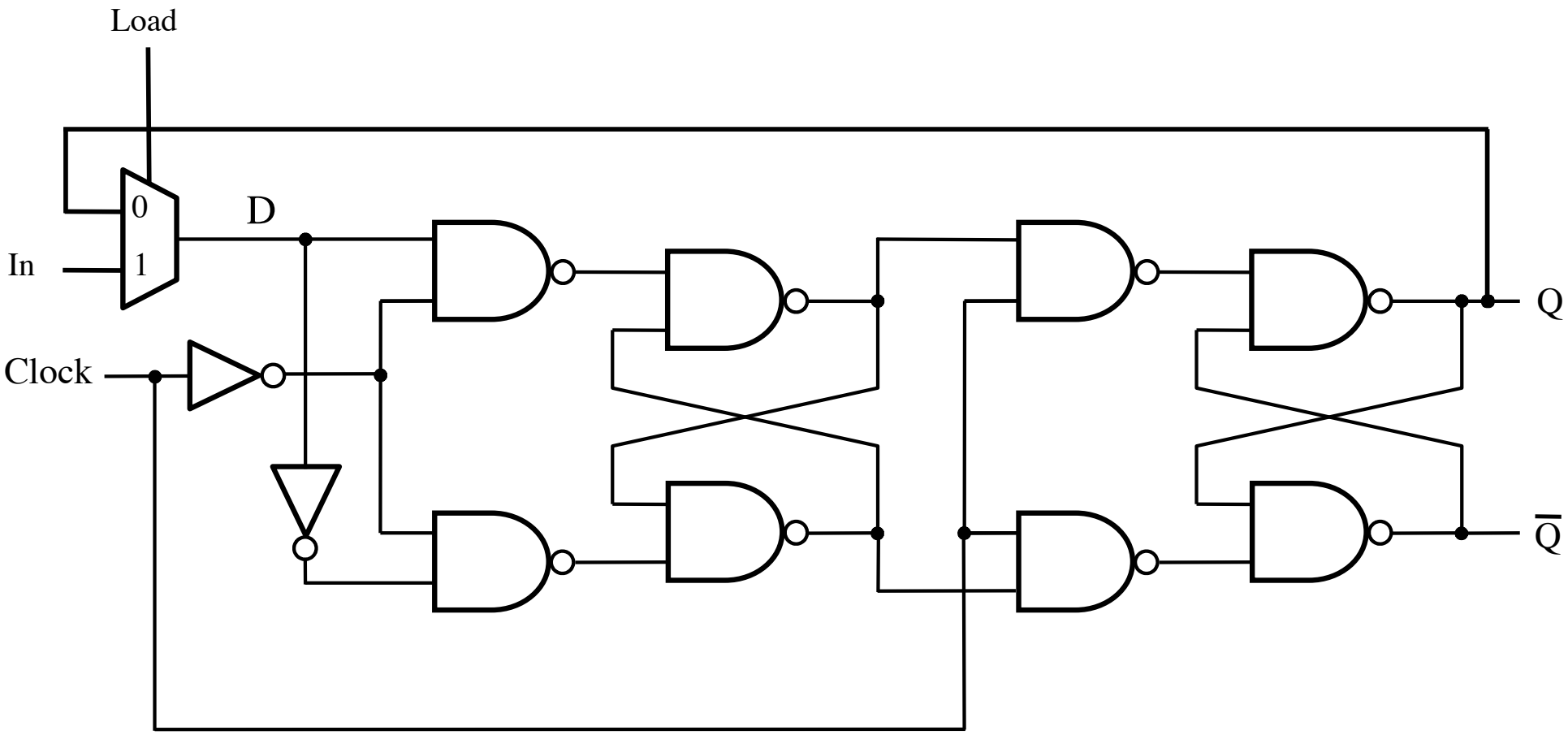
The 2-to-1 multiplexer is used to select whether to load a new value into the D flip-flop or to retain the old value.

The output of this circuit is the Q output of the flip-flop.

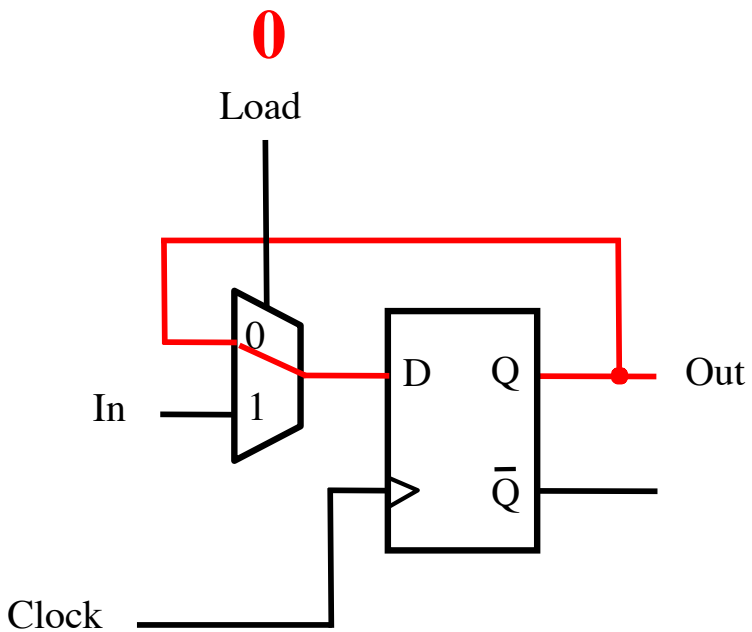
Positive-Edge-Triggered D Flip-Flop



1-bit Parallel-Access Register

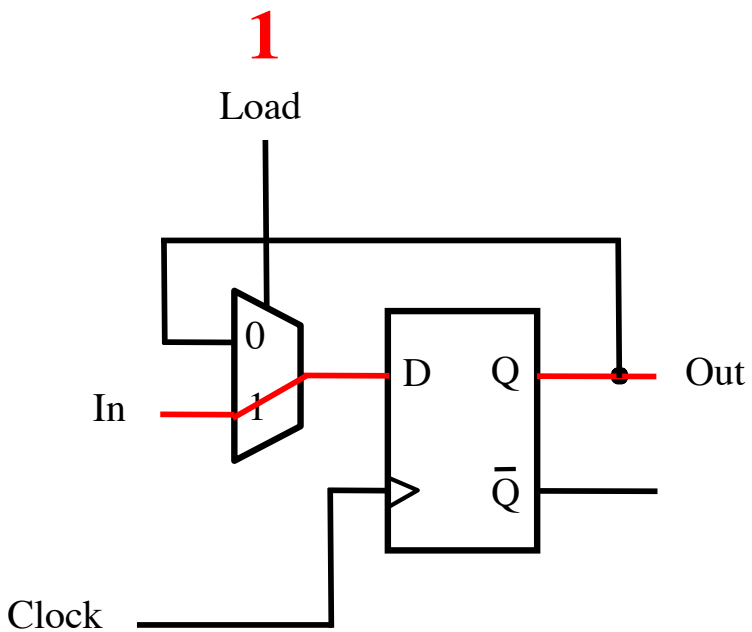


1-Bit Parallel-Access Register



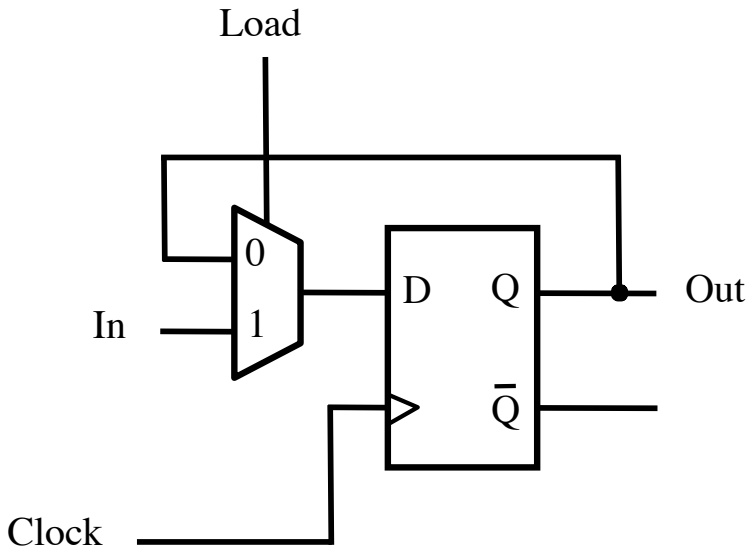
If Load = 0, then retain the old value.

1-Bit Parallel-Access Register



If Load = 1, then load the new value from In.

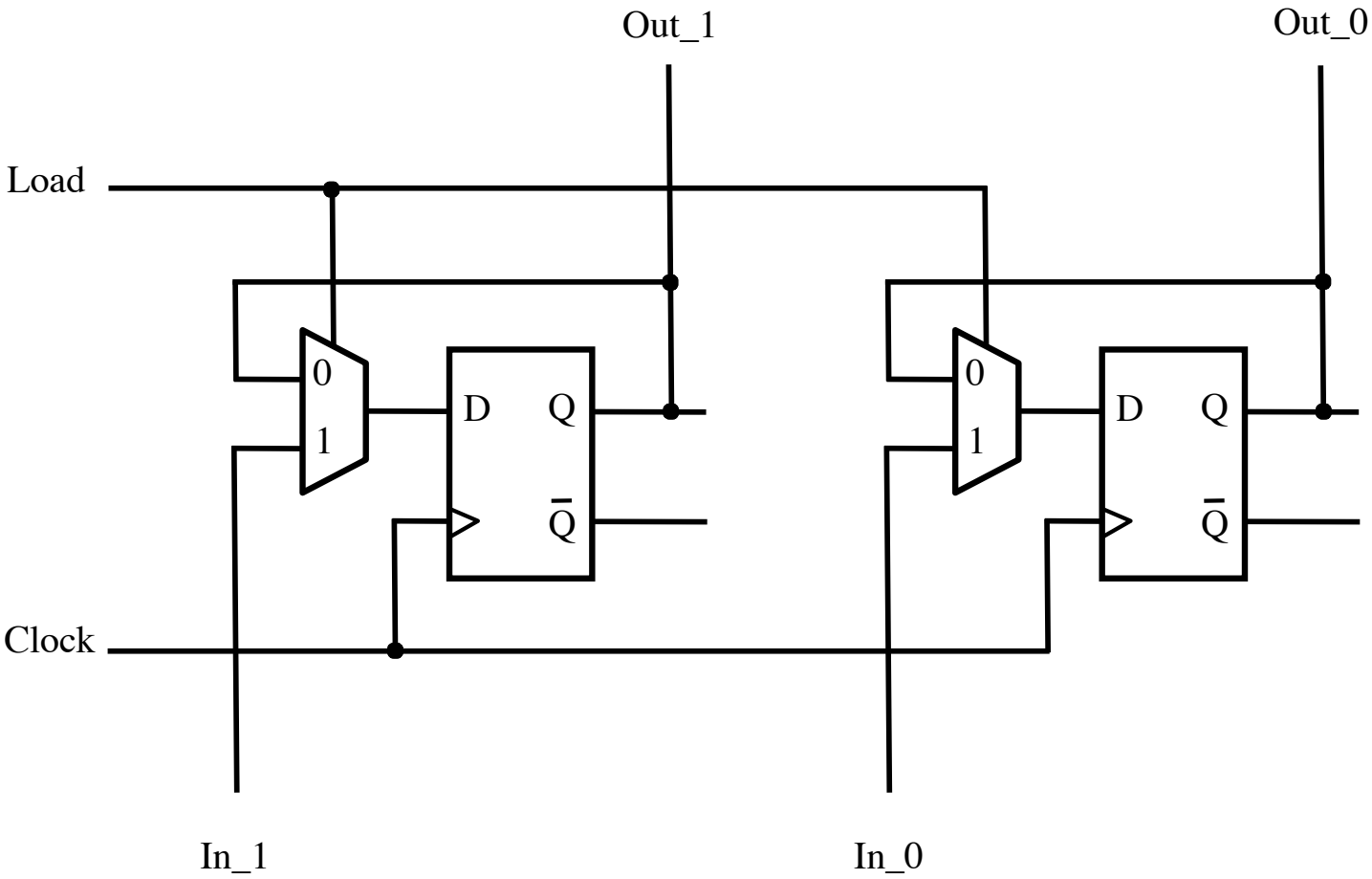
1-Bit Parallel-Access Register



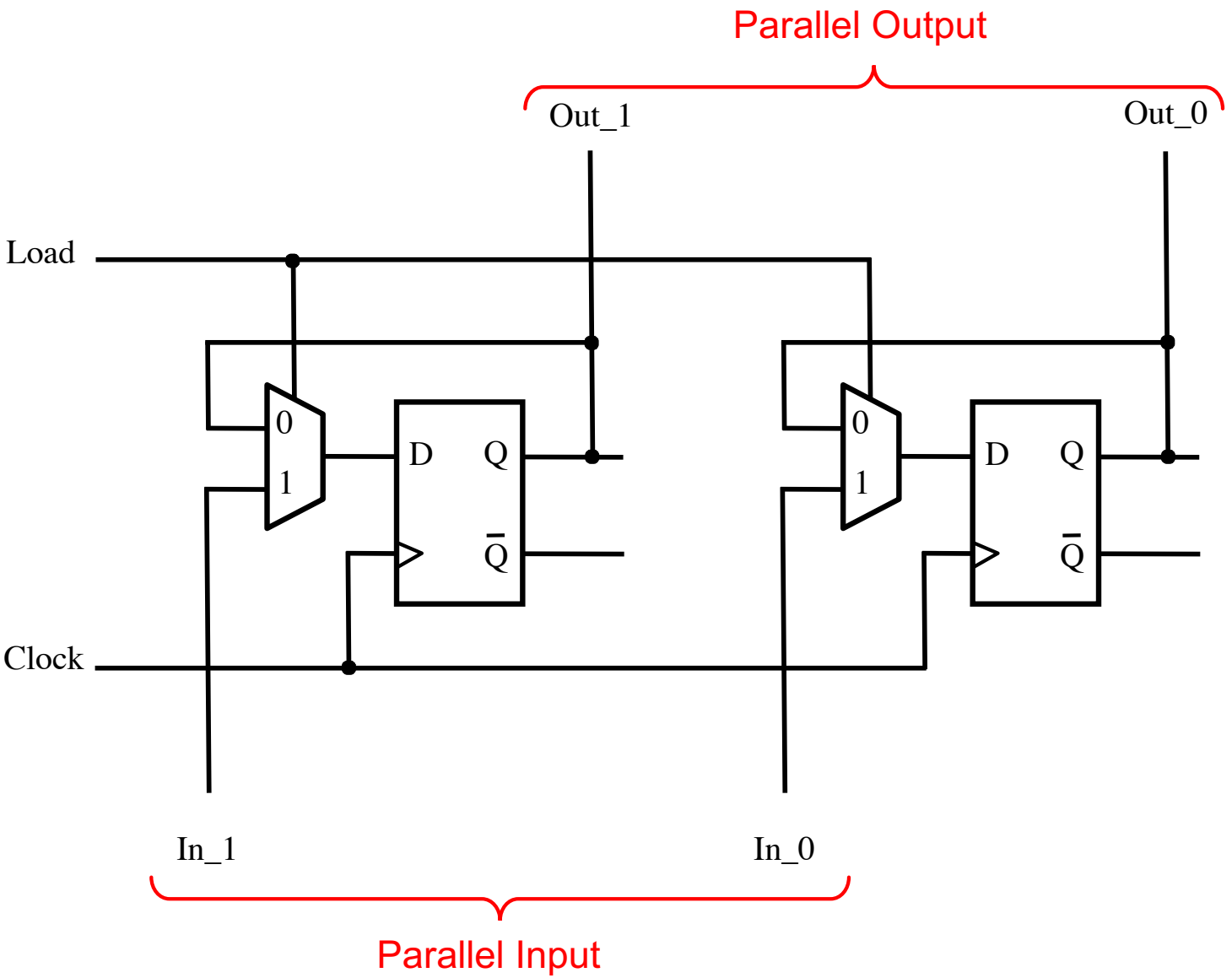
If Load = 0, then retain the old value.

If Load = 1, then load the new value from In.

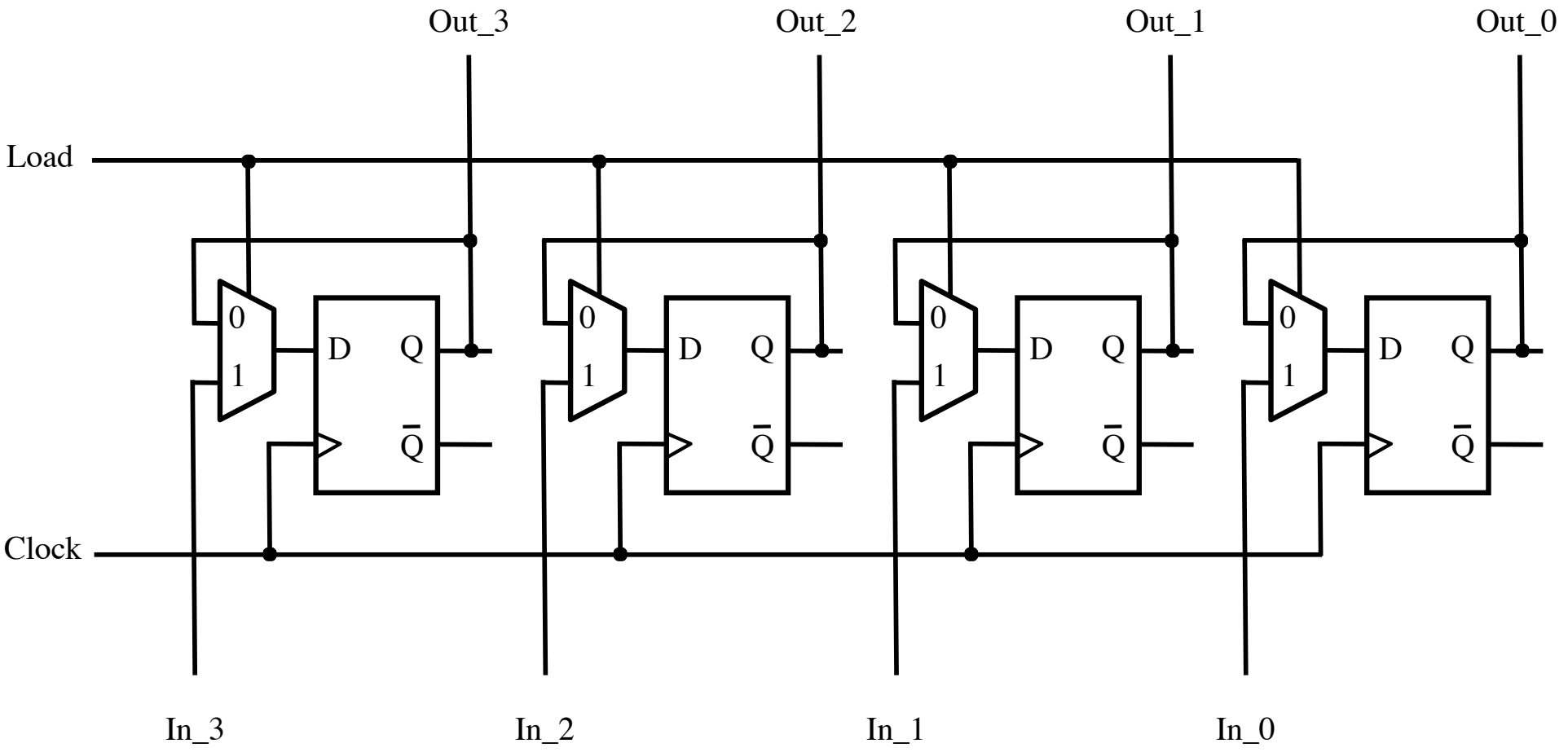
2-Bit Parallel-Access Register



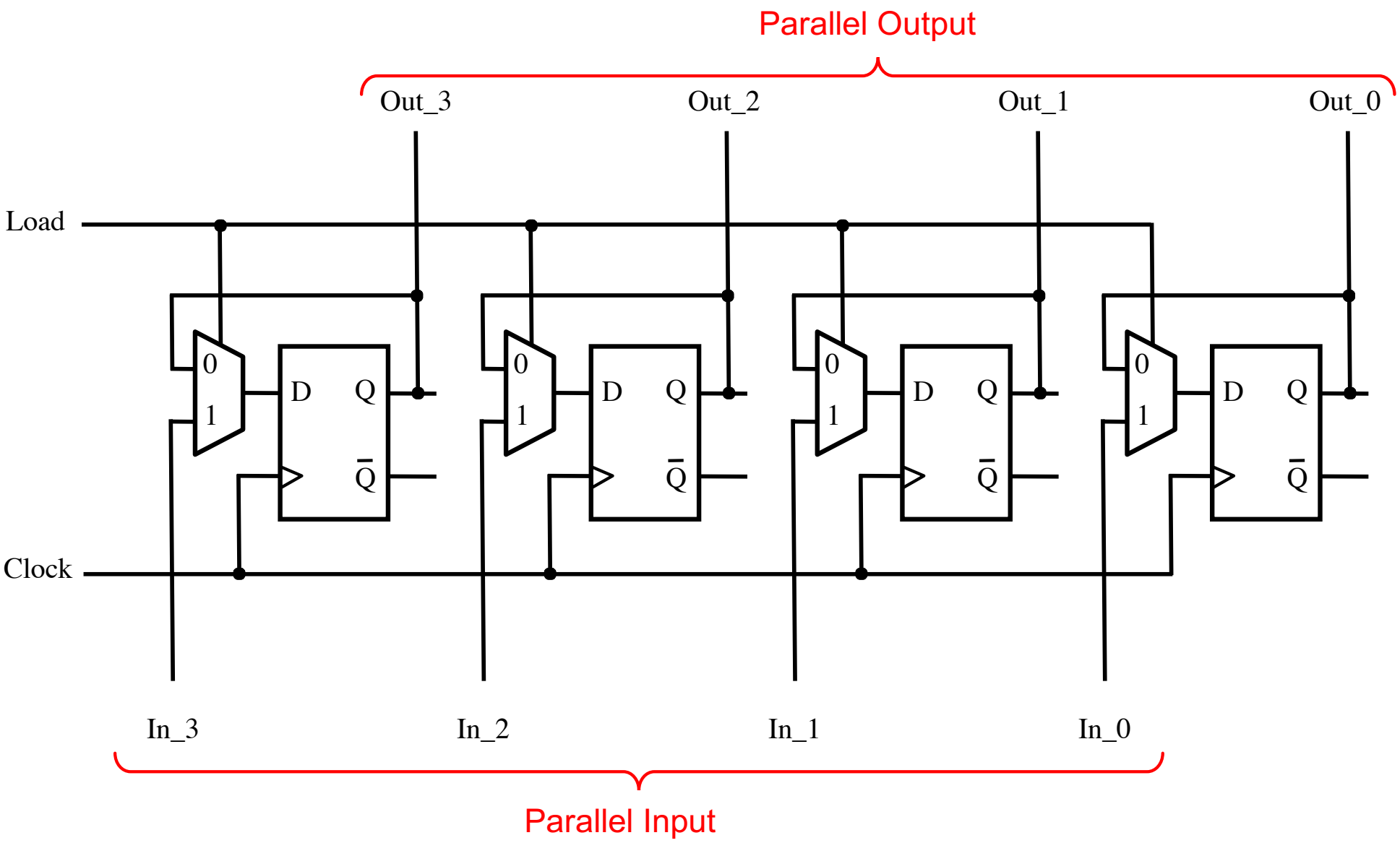
2-Bit Parallel-Access Register



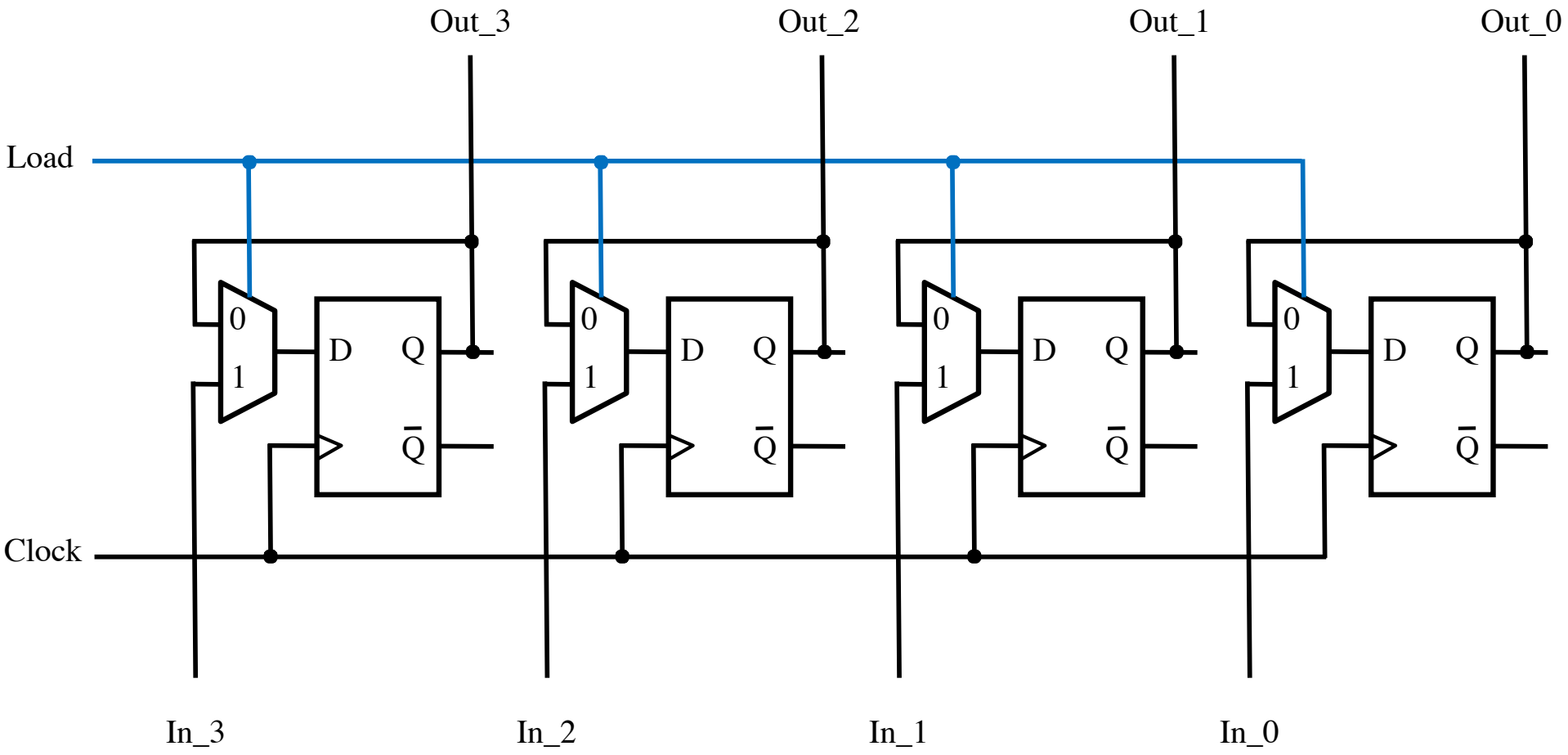
4-Bit Parallel-Access Register



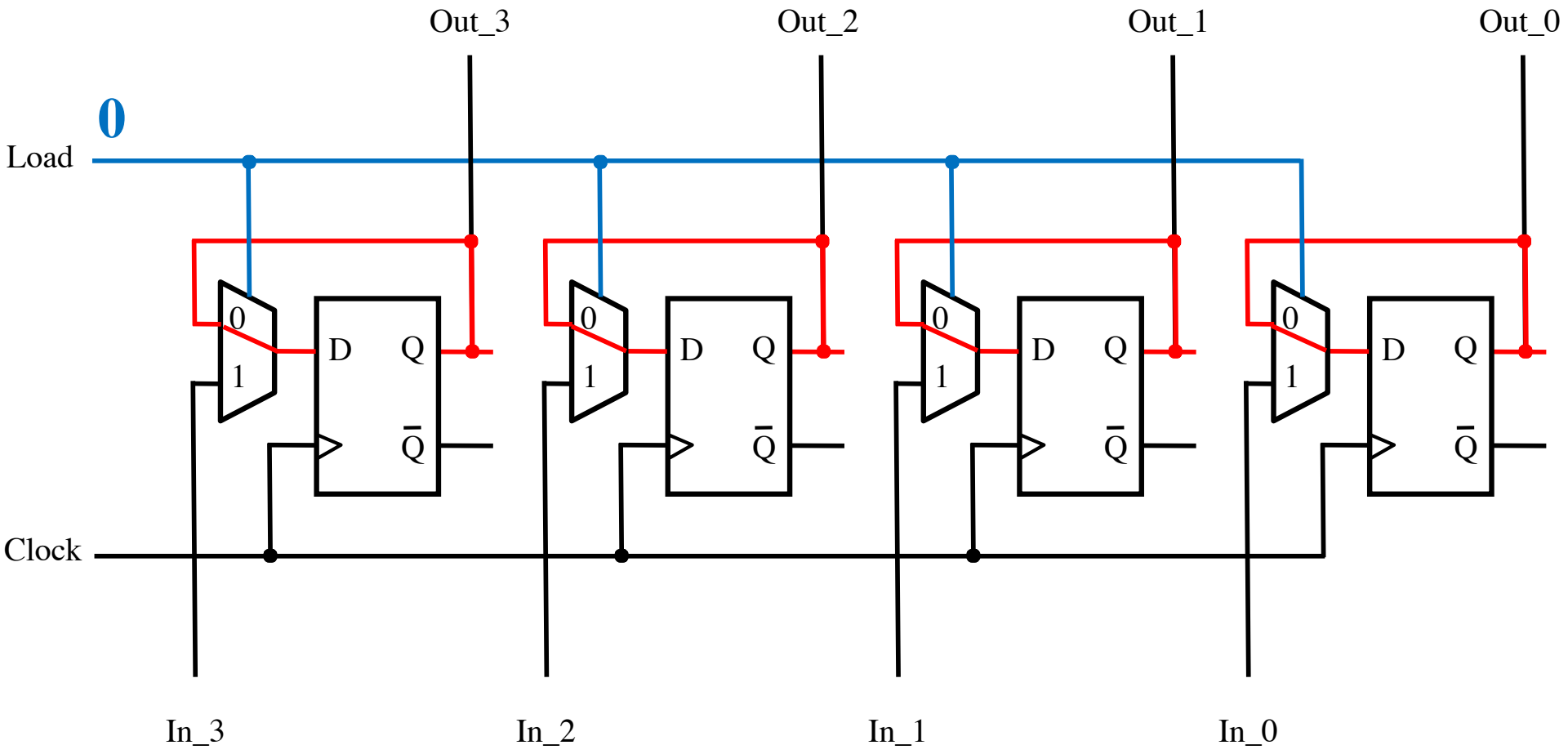
4-Bit Parallel-Access Register



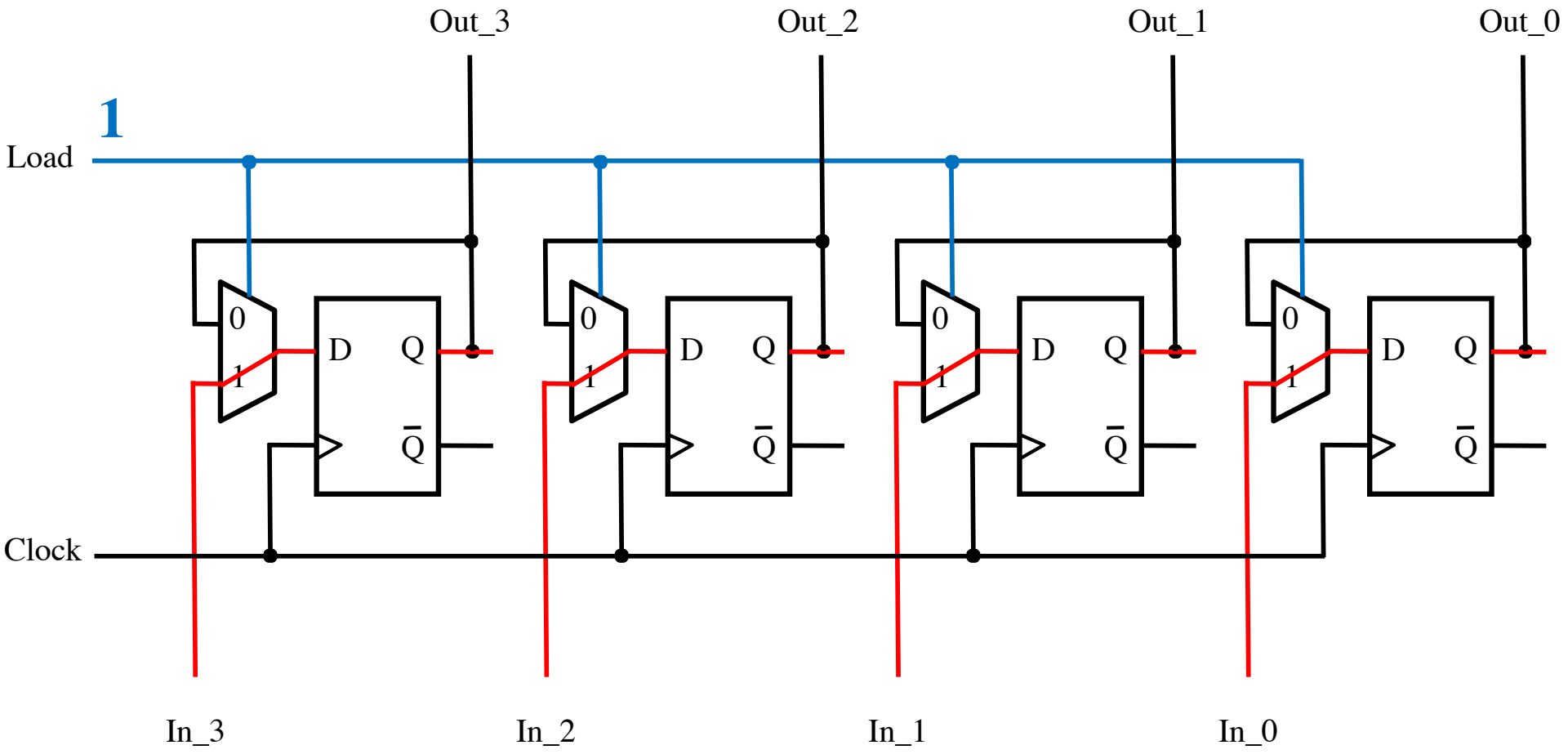
4-Bit Parallel-Access Register



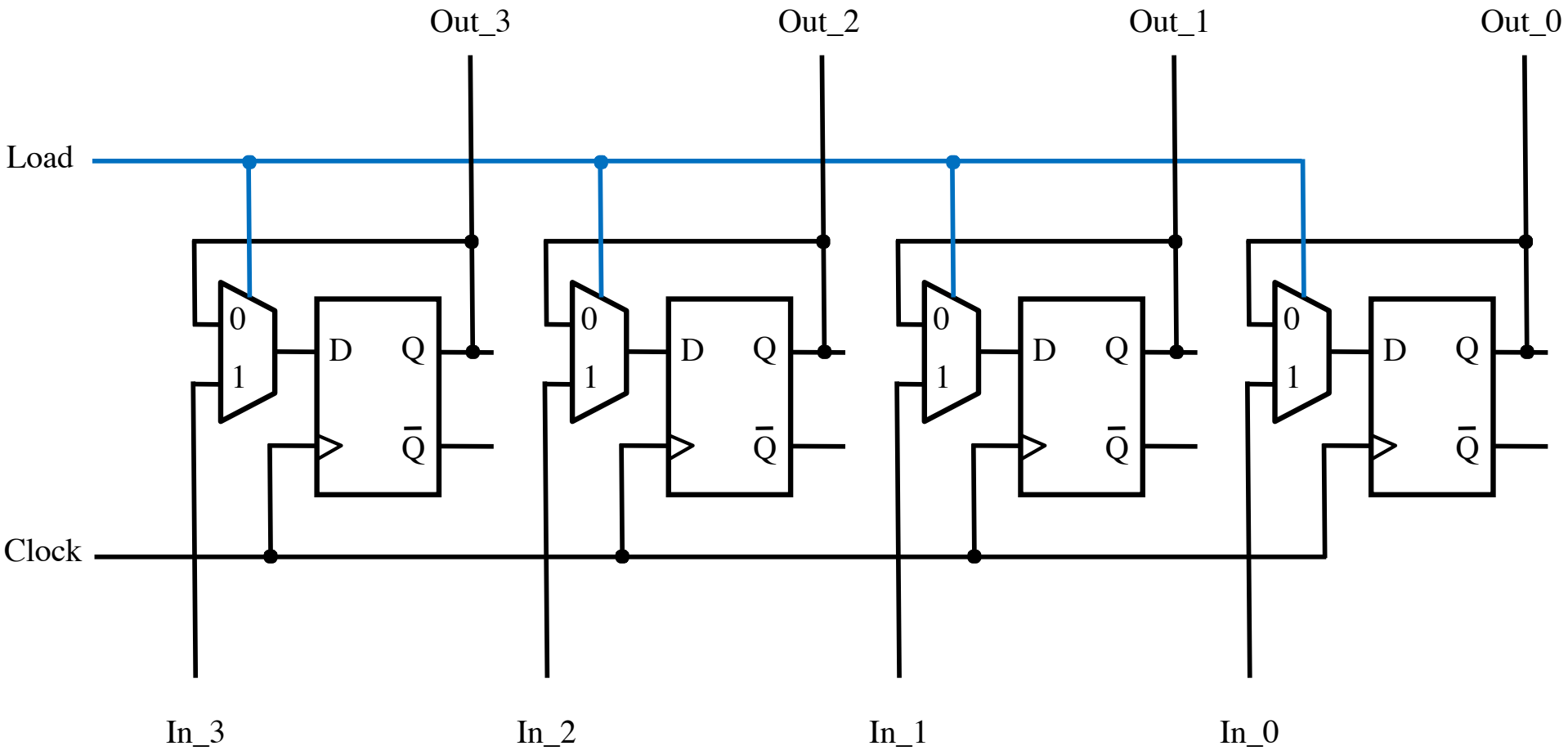
4-Bit Parallel-Access Register



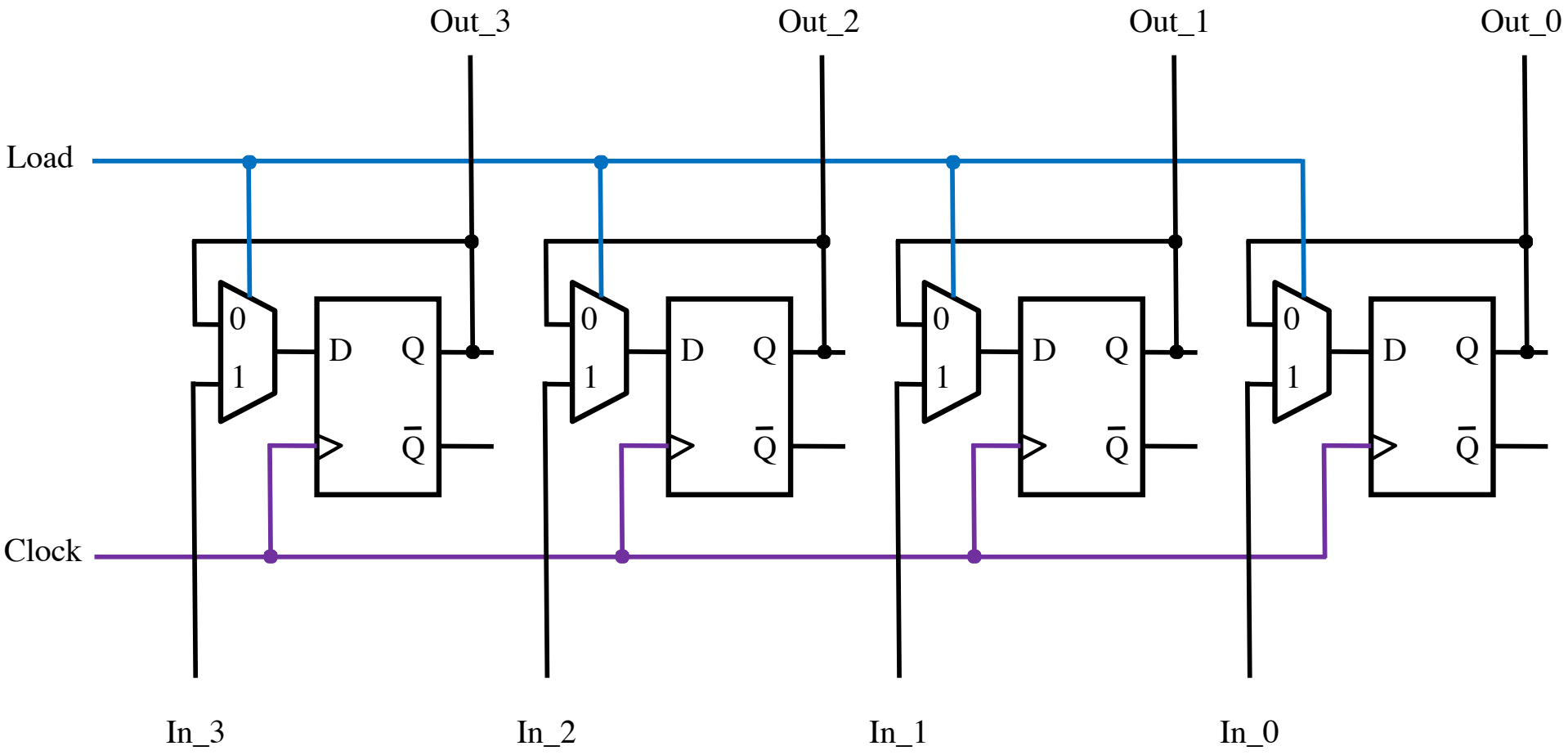
4-Bit Parallel-Access Register



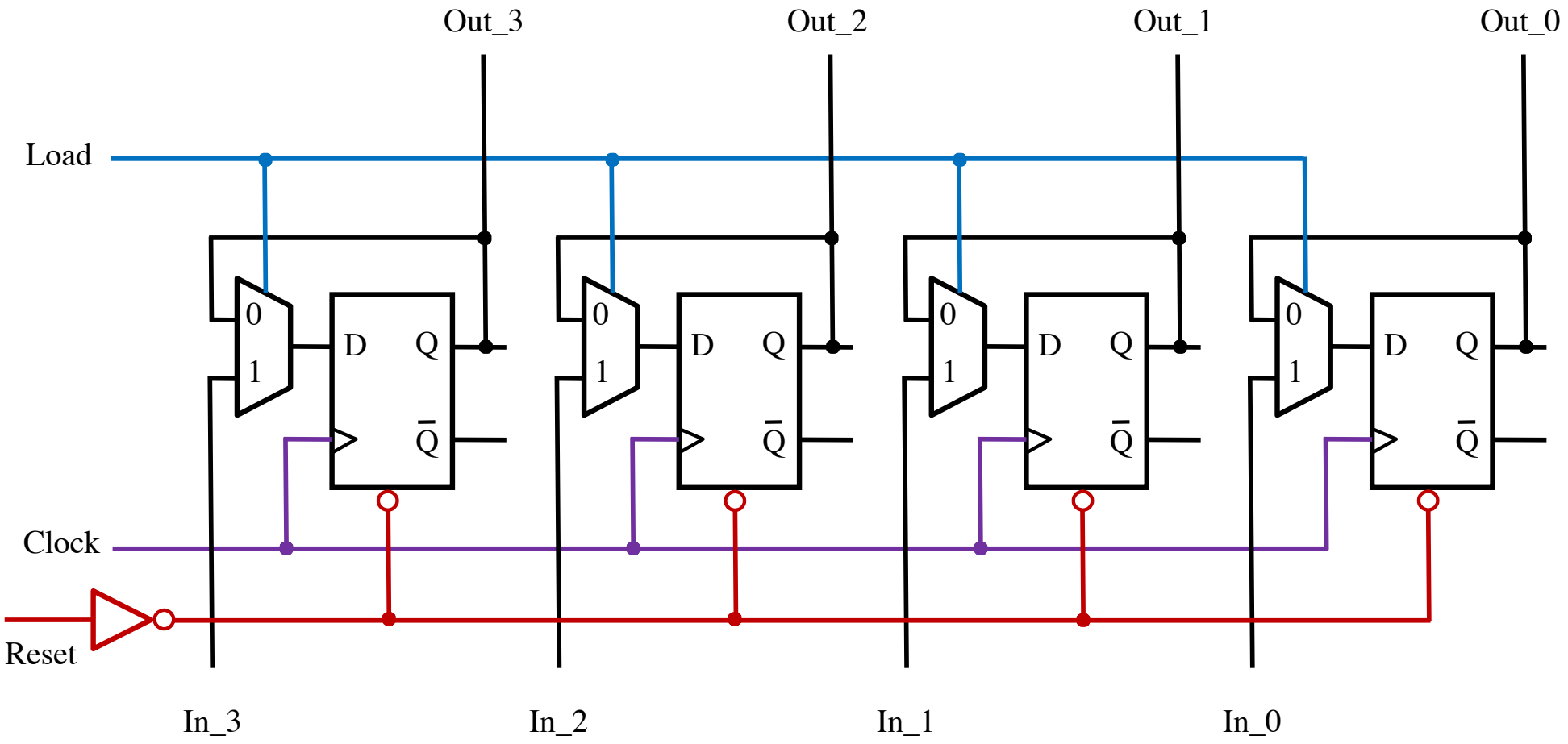
4-Bit Parallel-Access Register



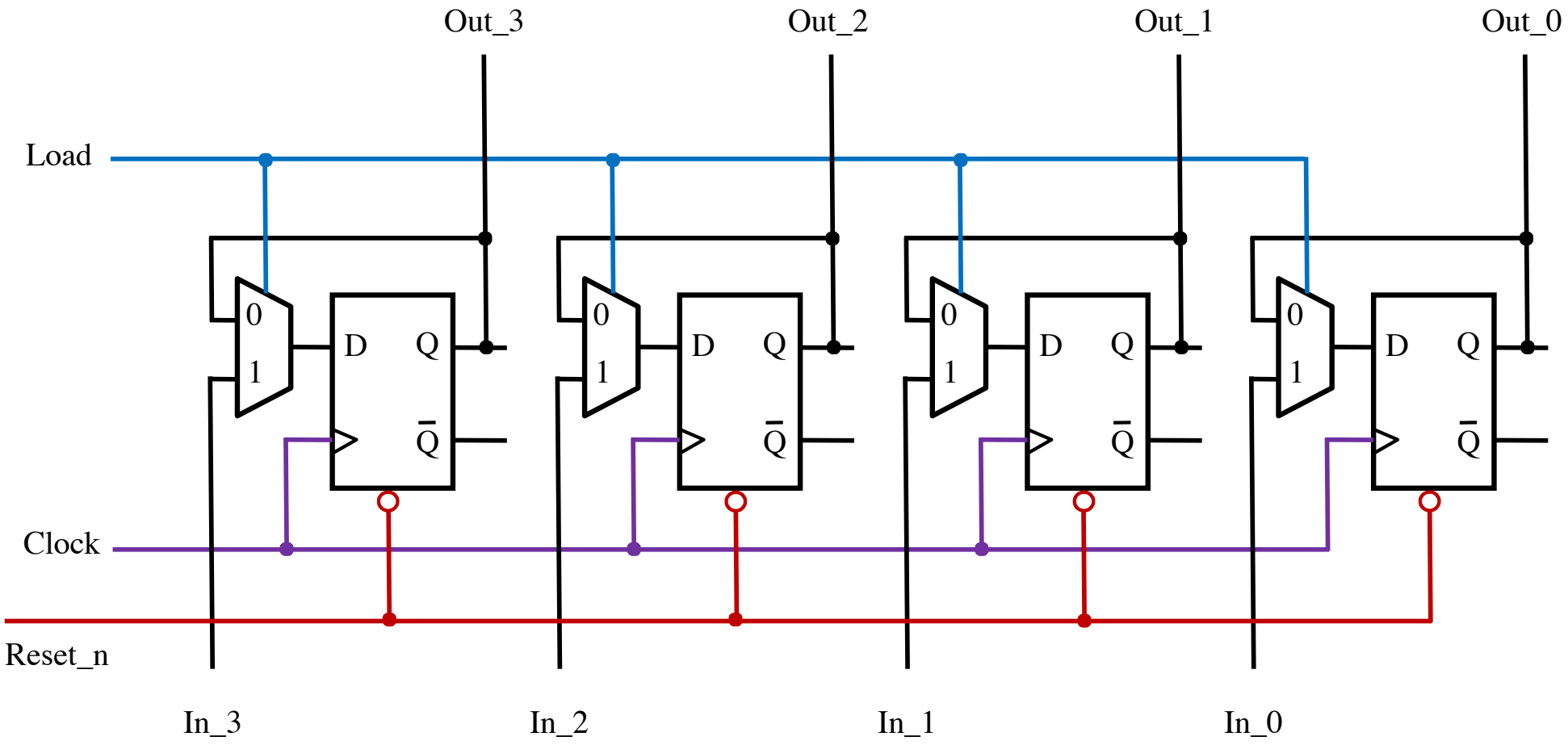
4-Bit Parallel-Access Register



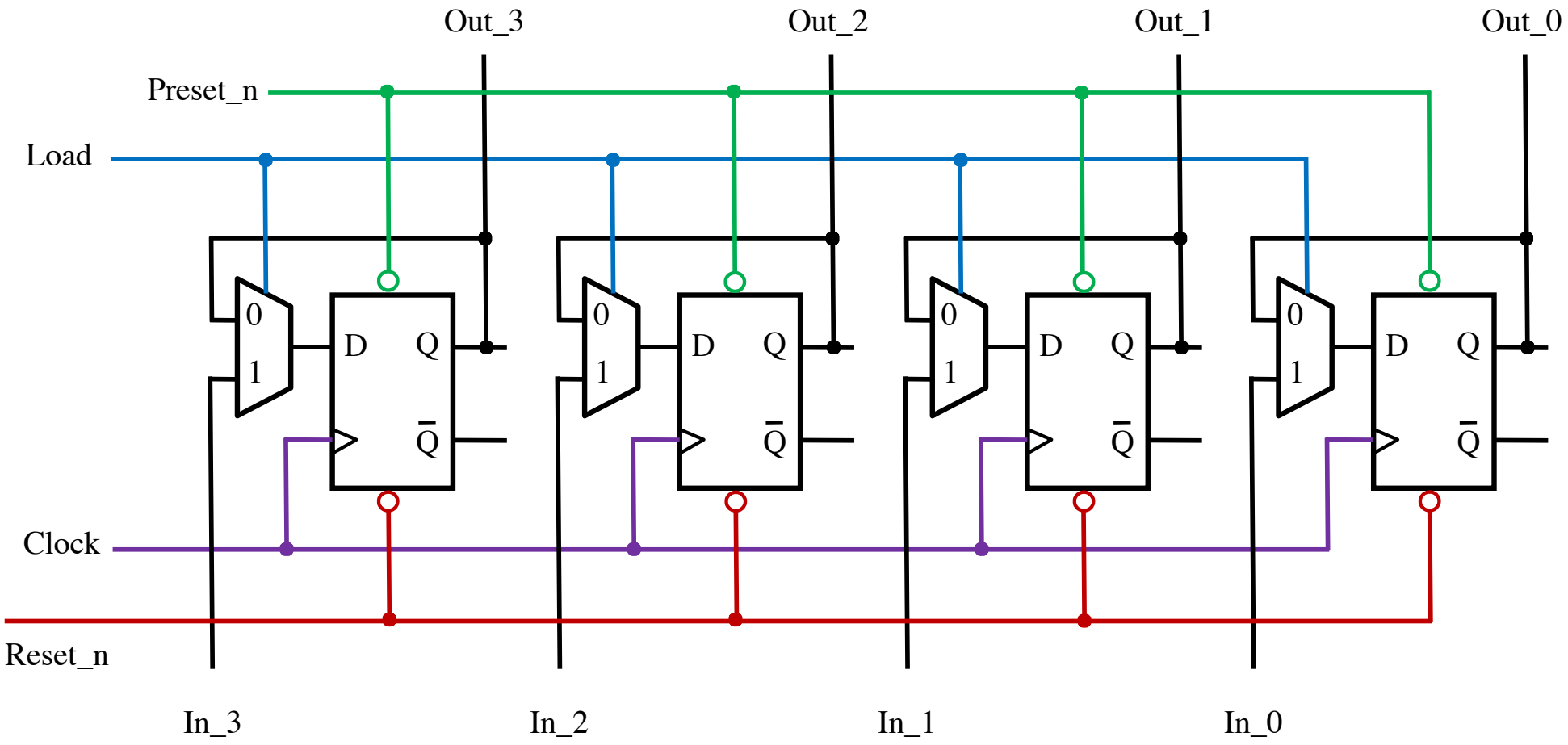
4-Bit Parallel-Access Register



4-Bit Parallel-Access Register

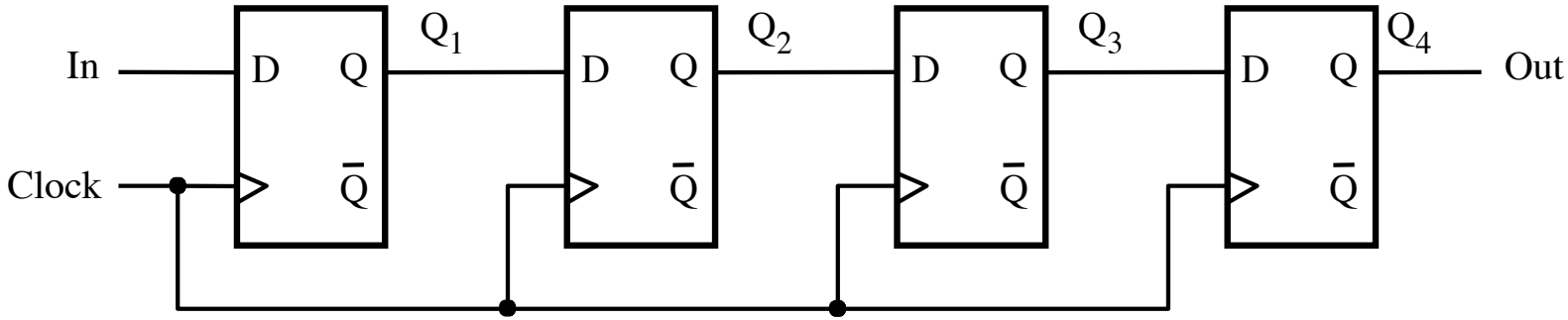


4-Bit Parallel-Access Register



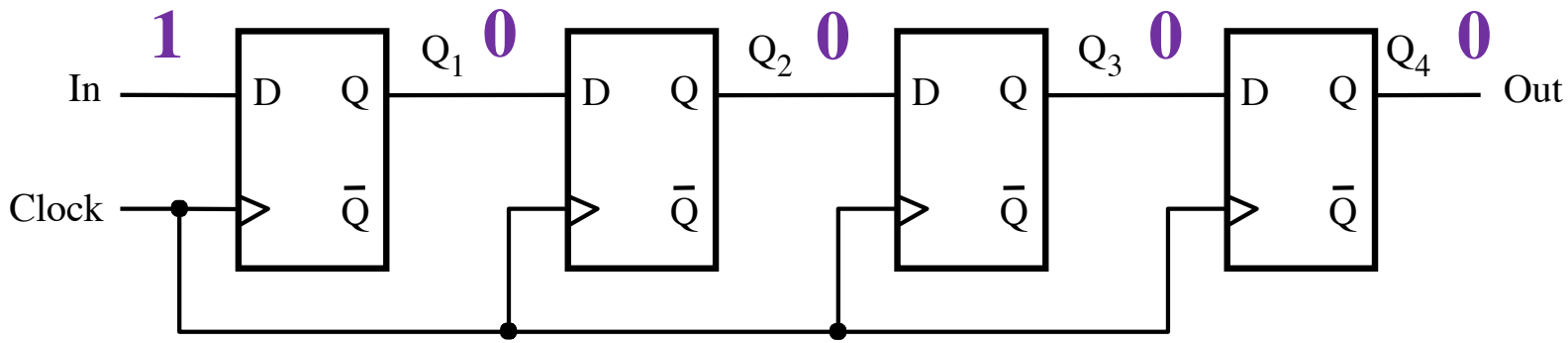
Shift Register

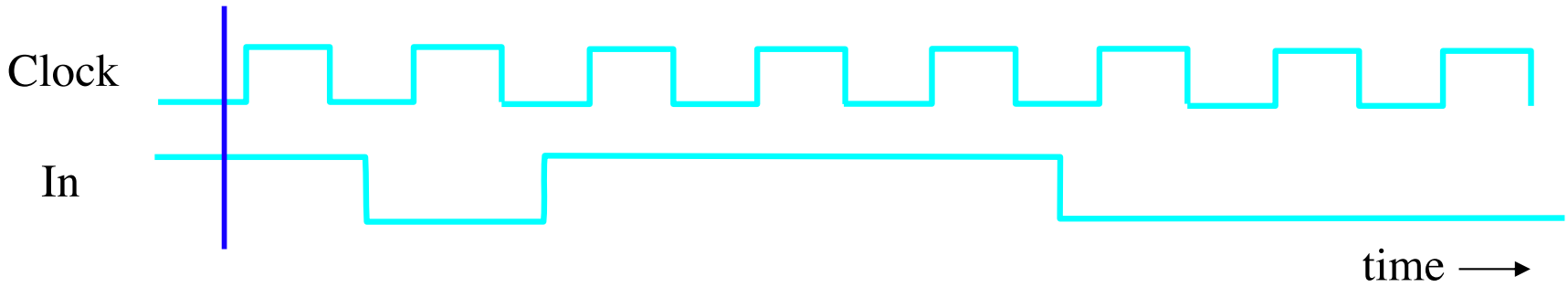
A simple shift register



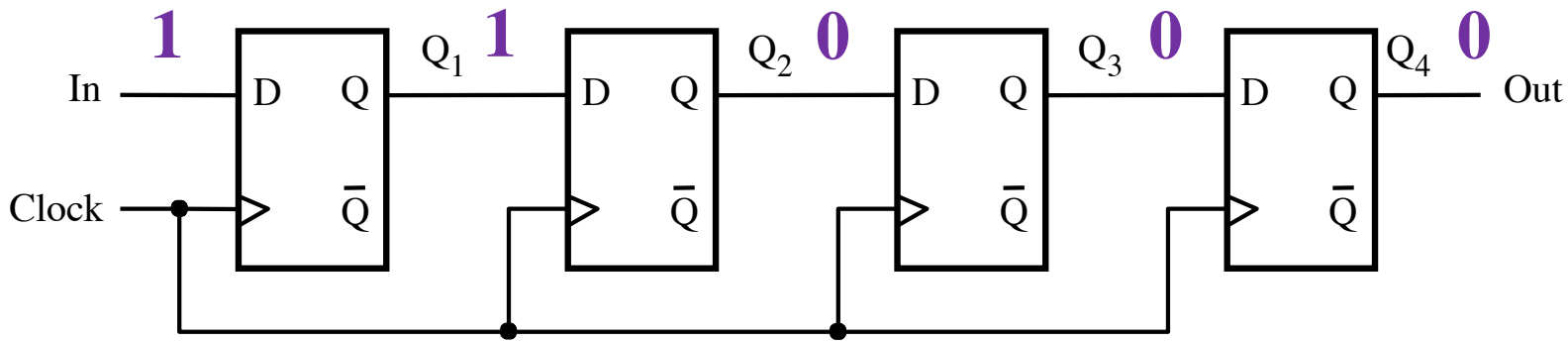
[Figure 5.17a from the textbook]

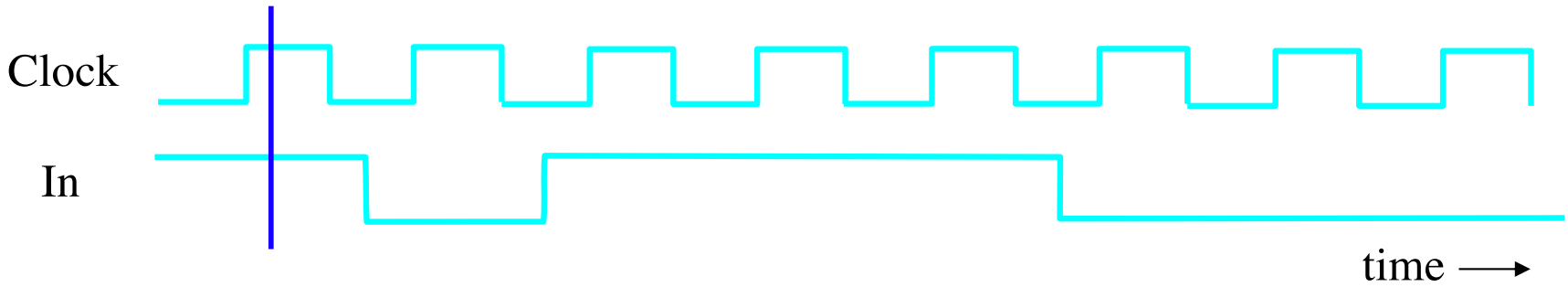
Shift Register Simulation



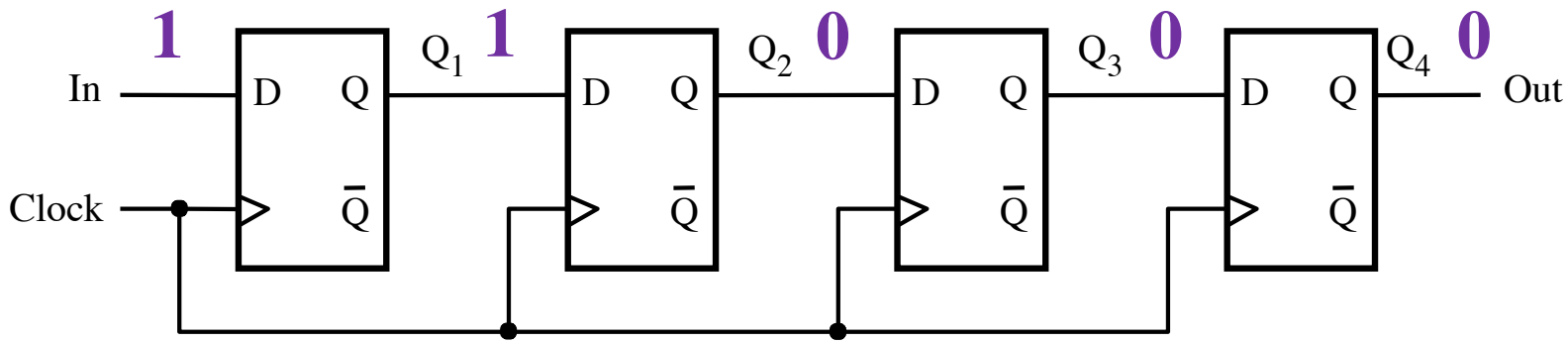


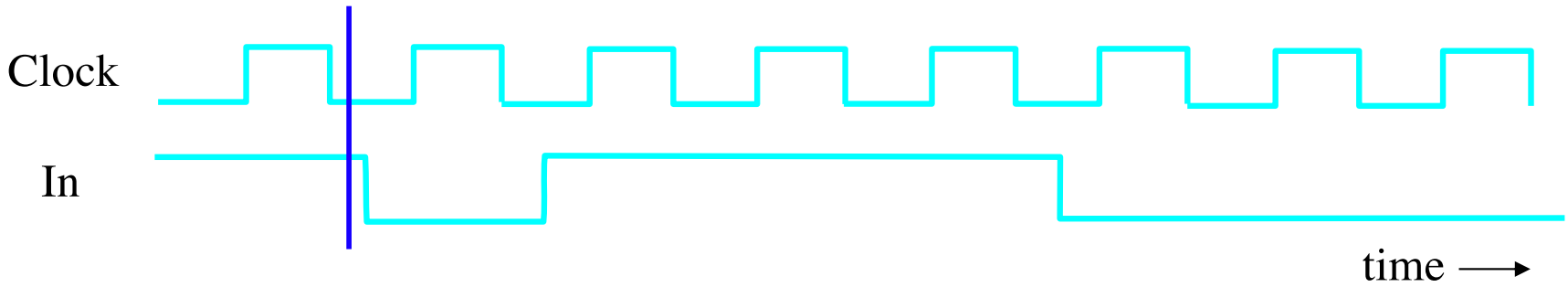
Shift Register Simulation



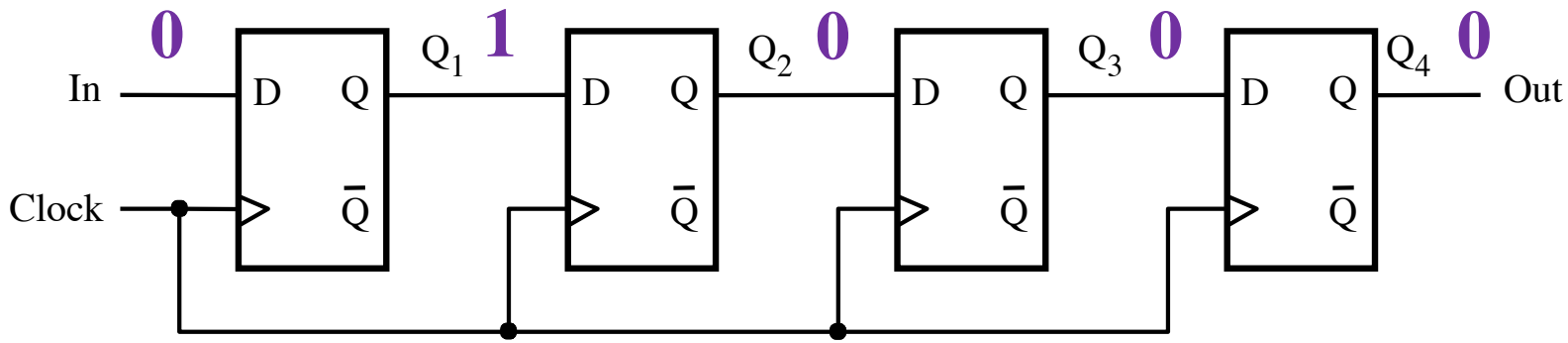


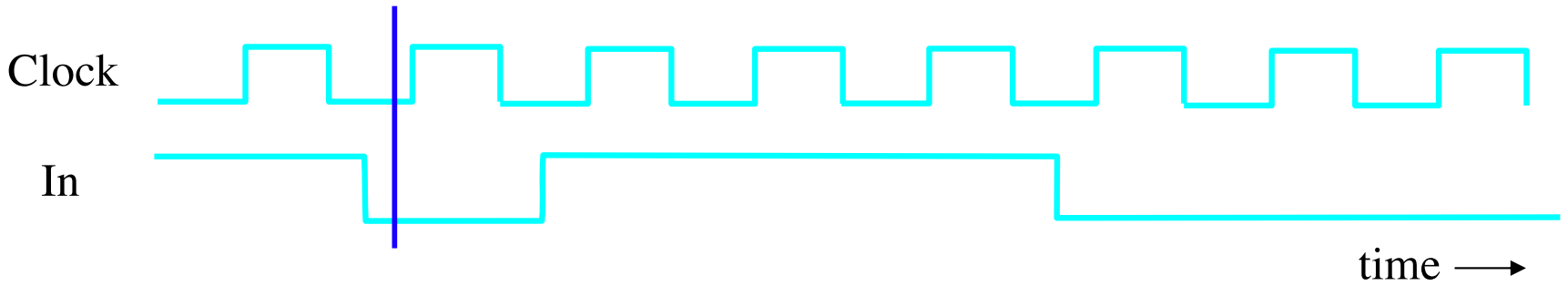
Shift Register Simulation



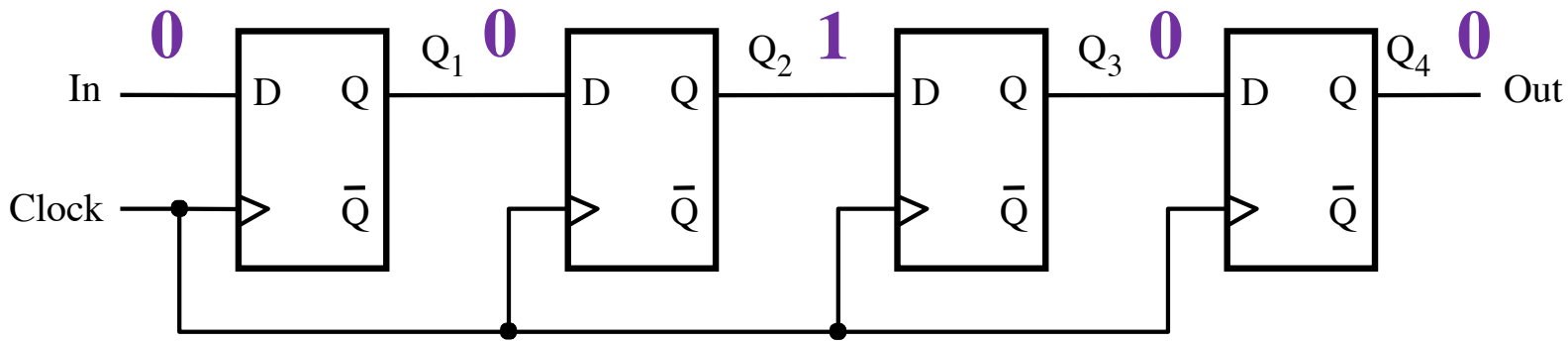


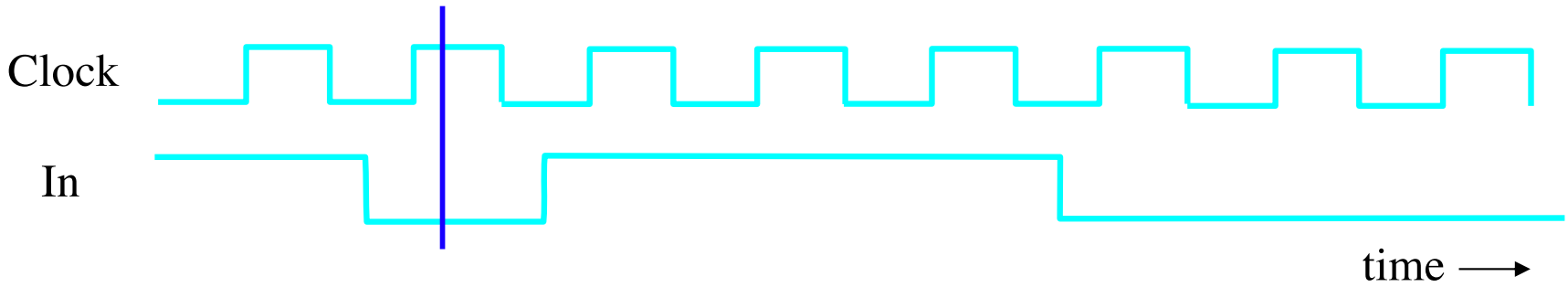
Shift Register Simulation



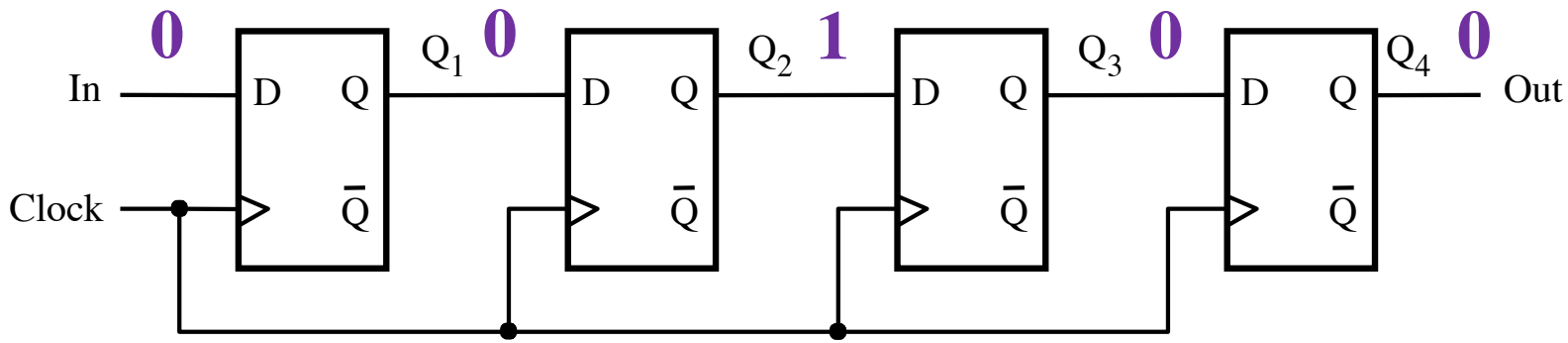


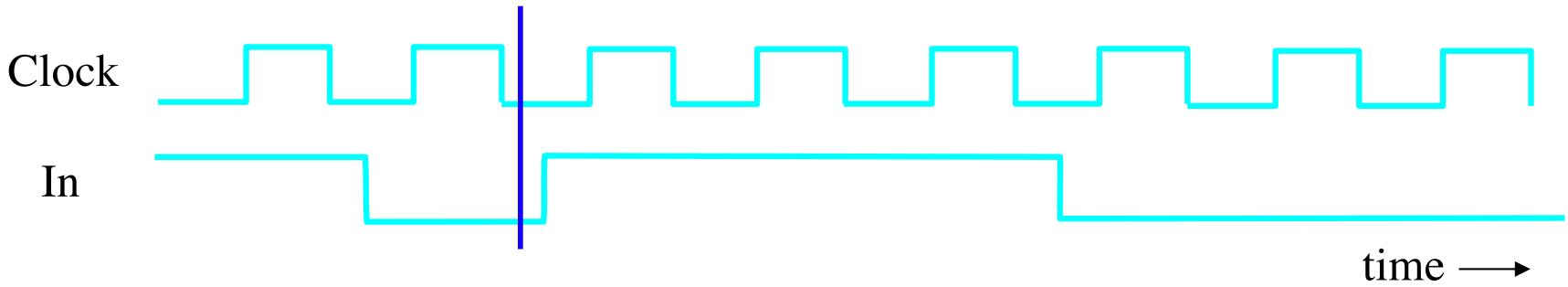
Shift Register Simulation



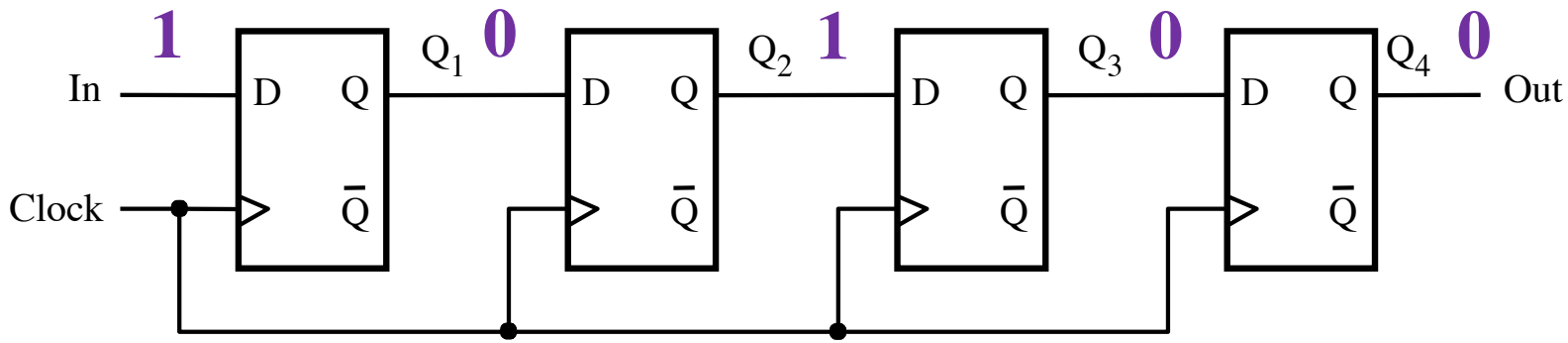


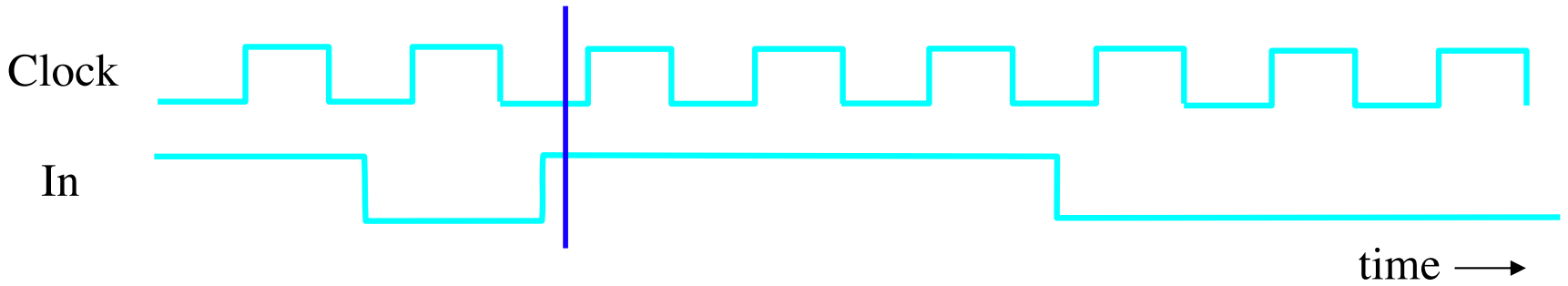
Shift Register Simulation



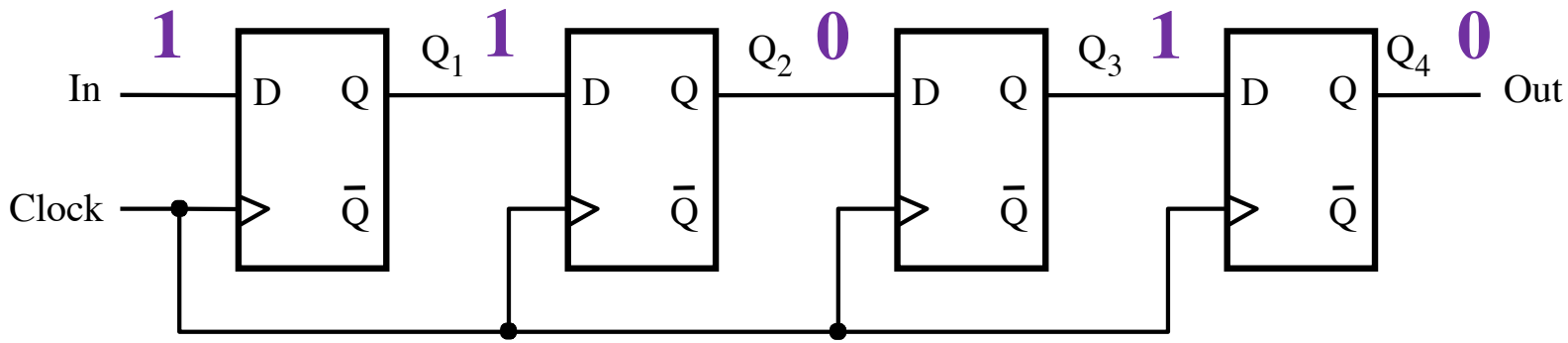


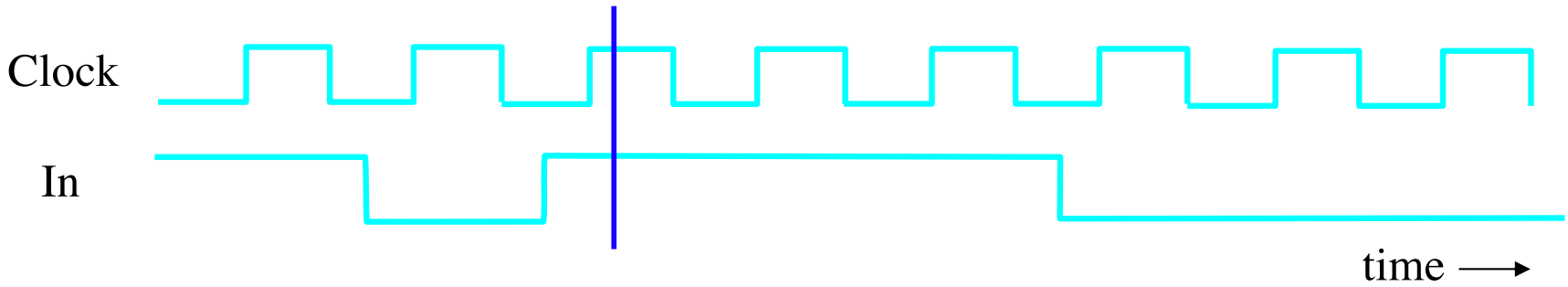
Shift Register Simulation



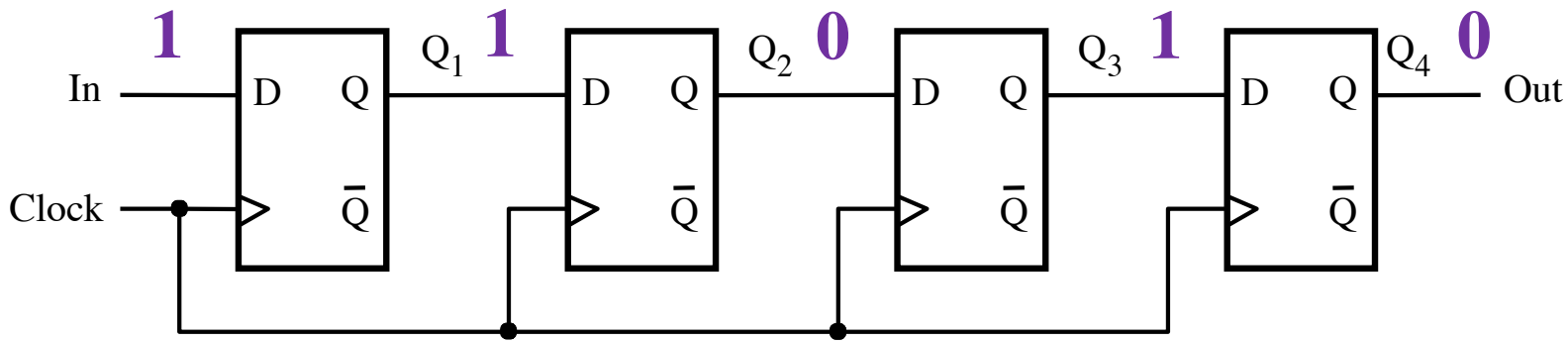


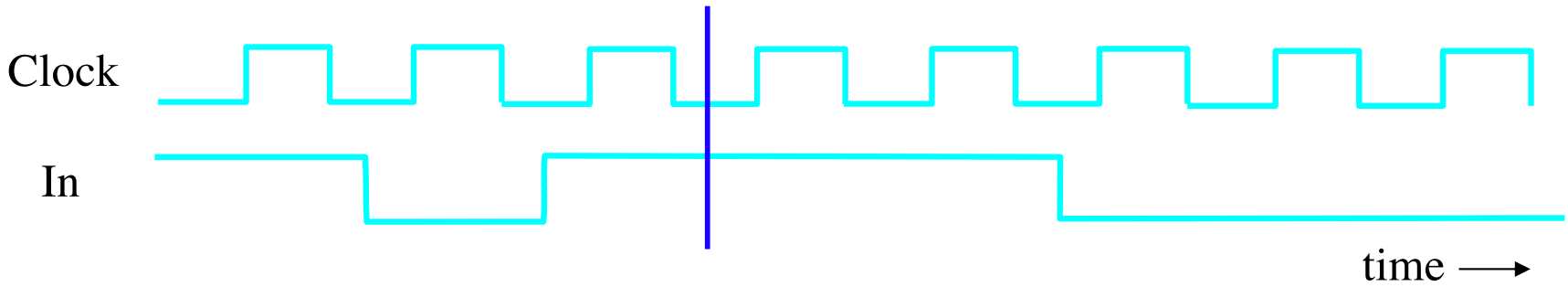
Shift Register Simulation



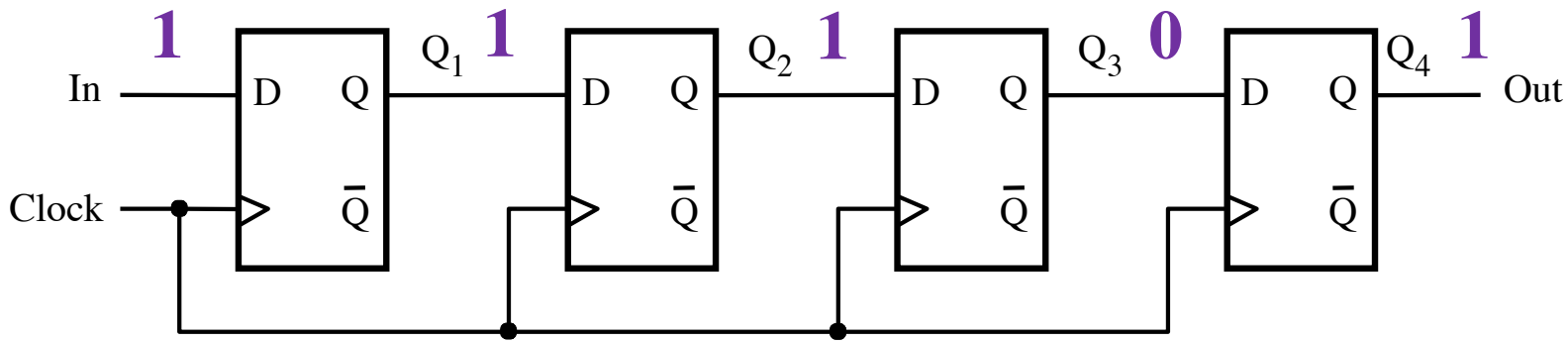


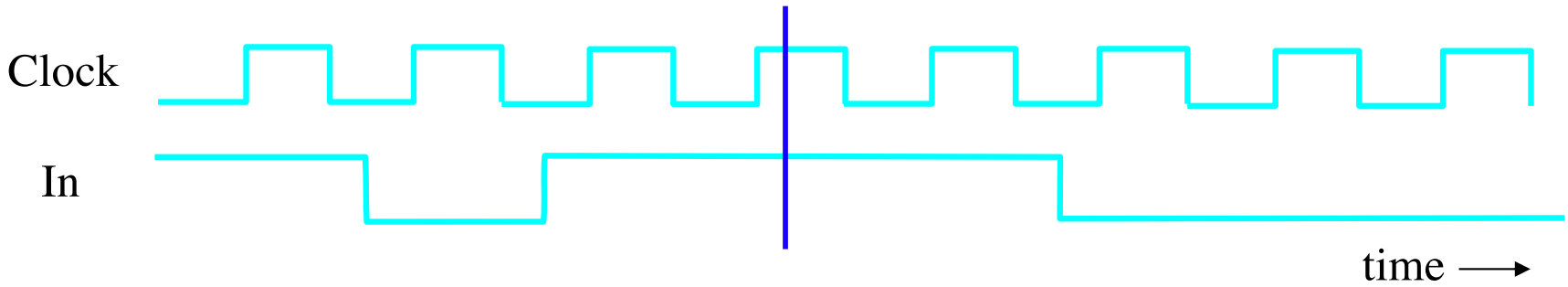
Shift Register Simulation



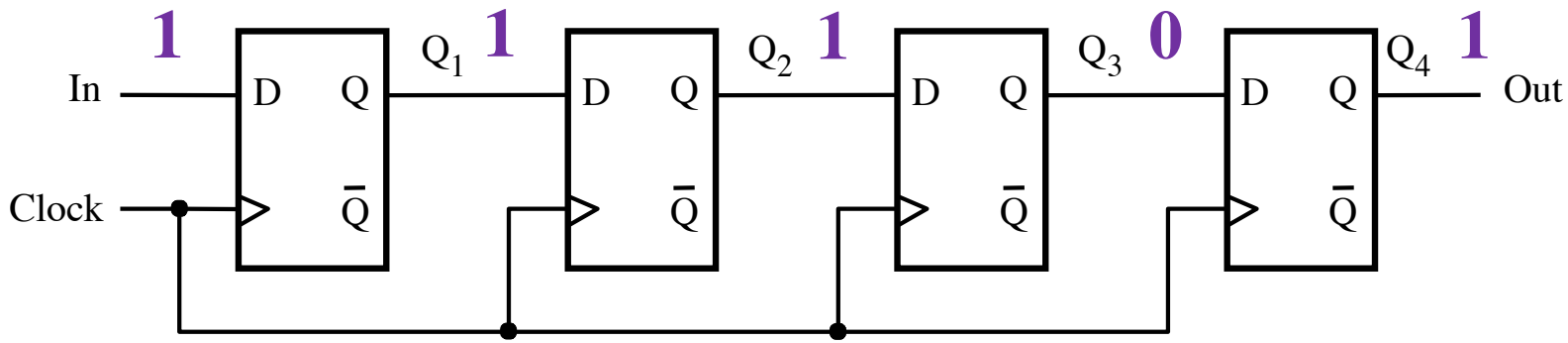


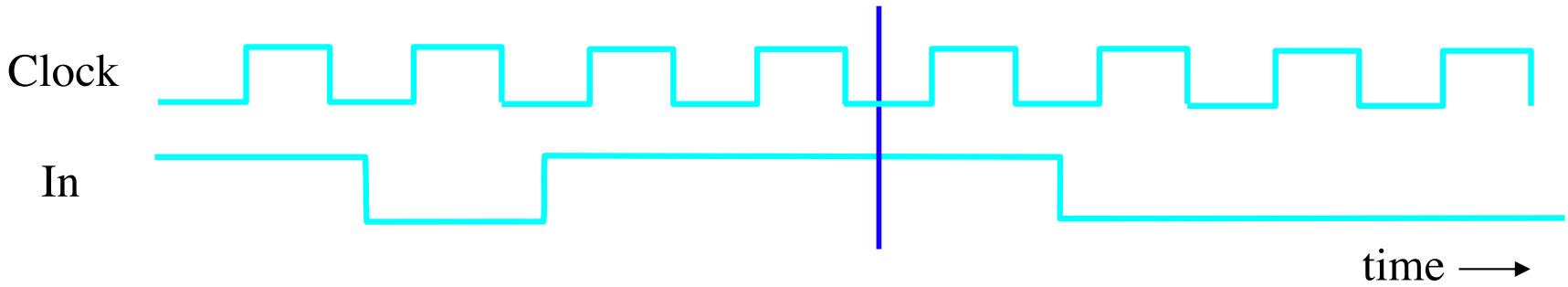
Shift Register Simulation



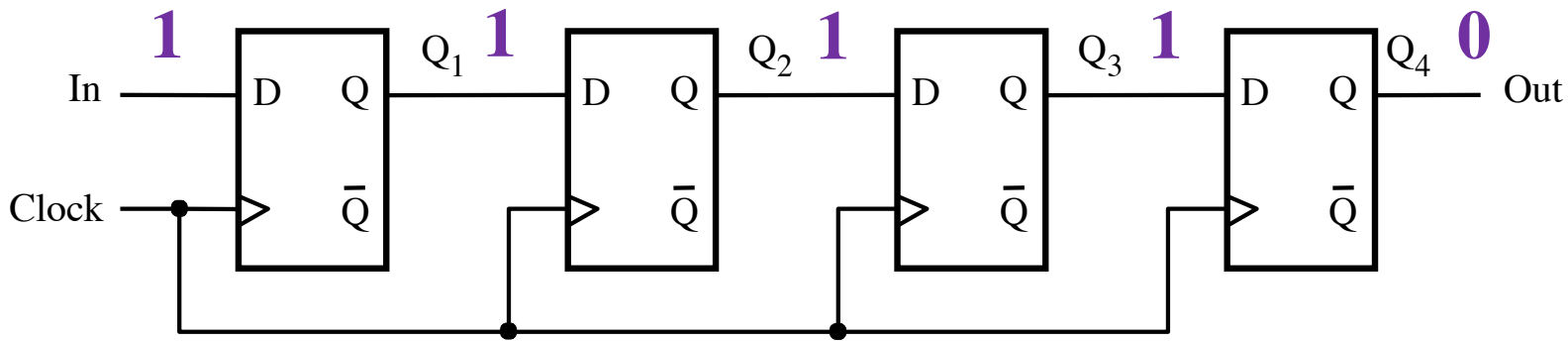


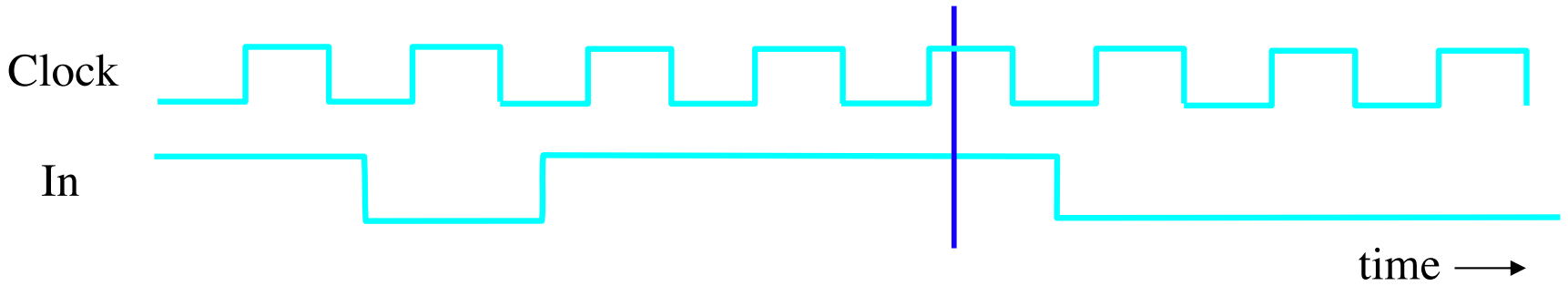
Shift Register Simulation



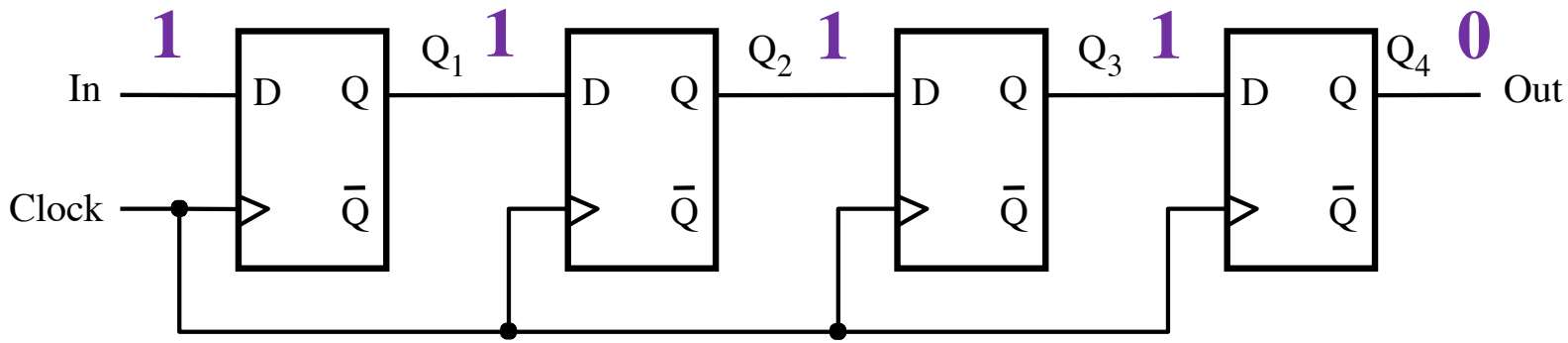


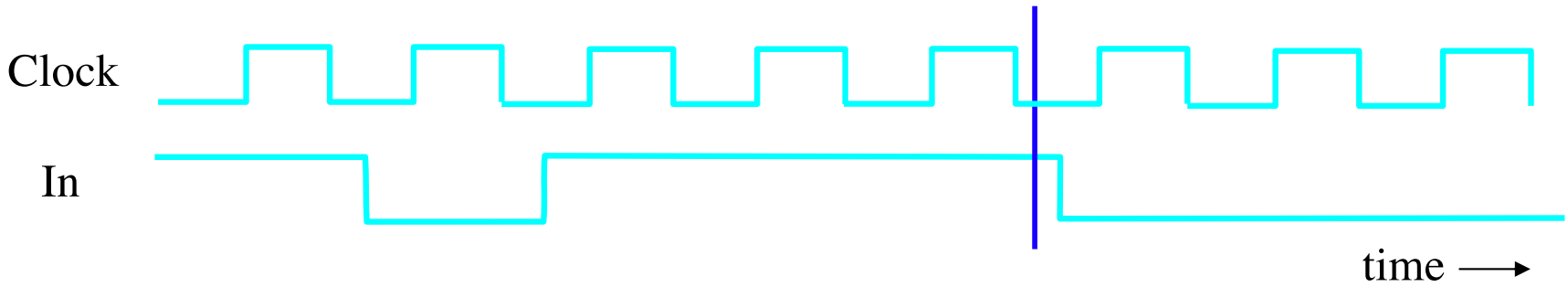
Shift Register Simulation



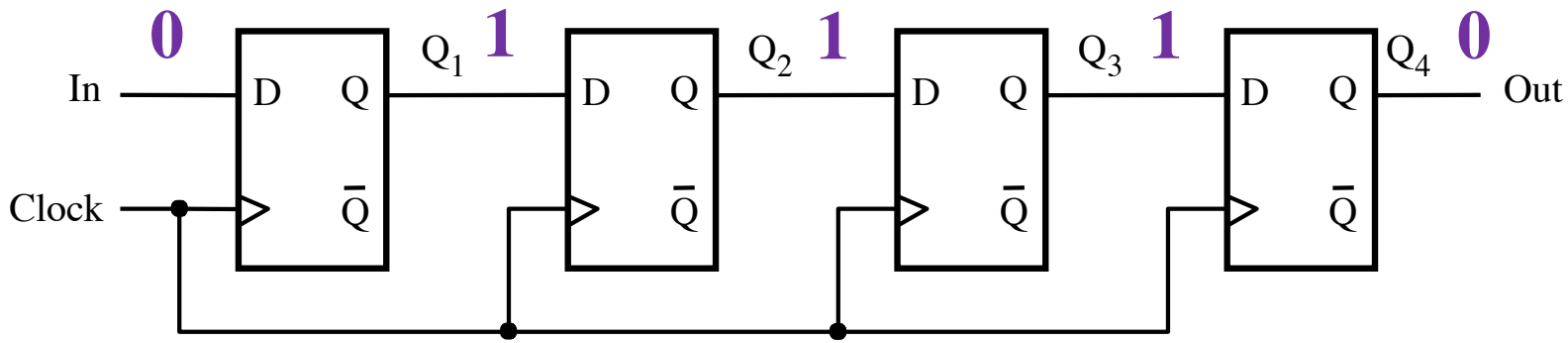


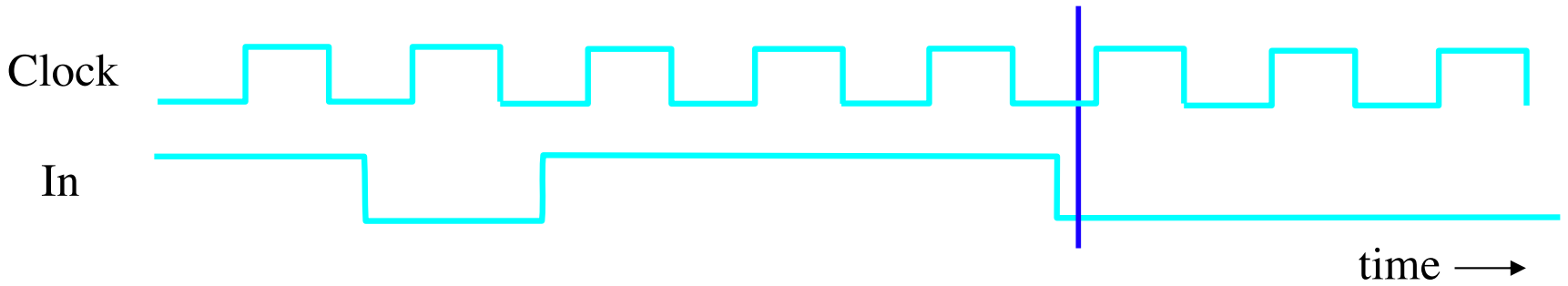
Shift Register Simulation



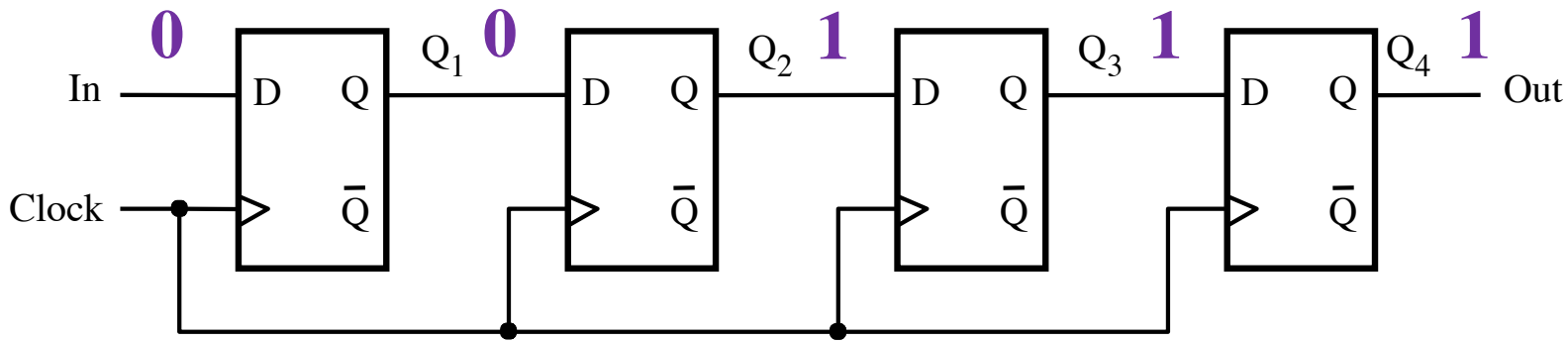


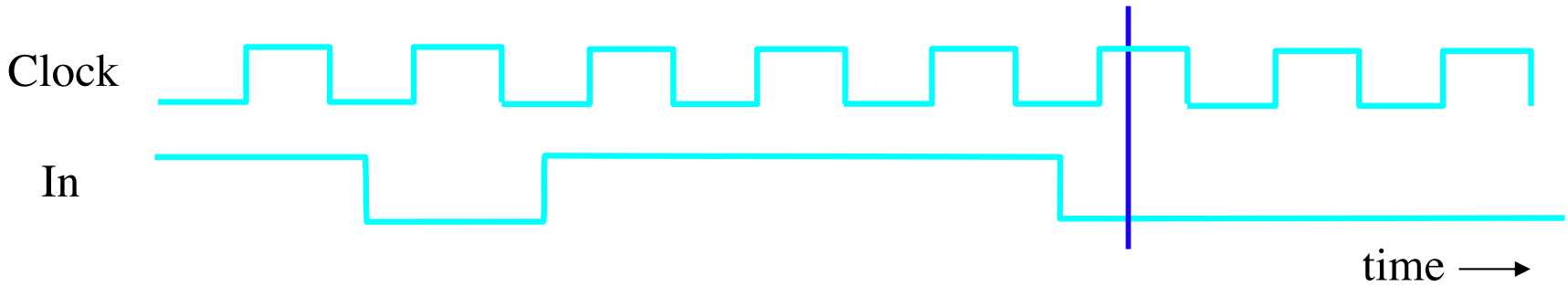
Shift Register Simulation



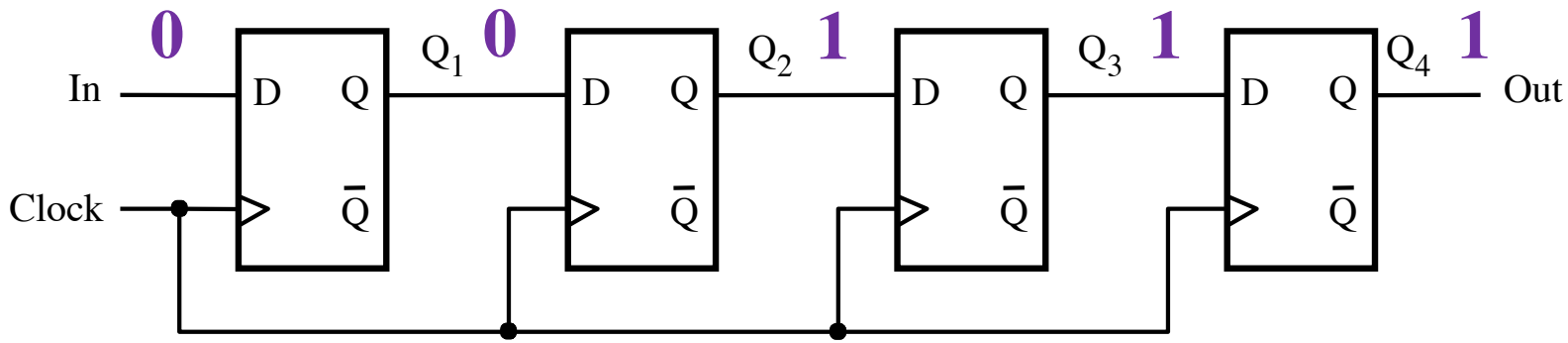


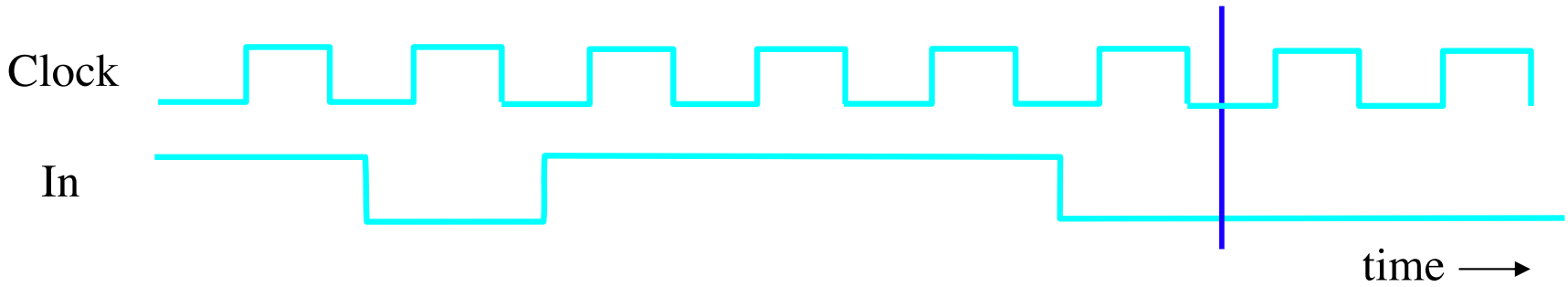
Shift Register Simulation



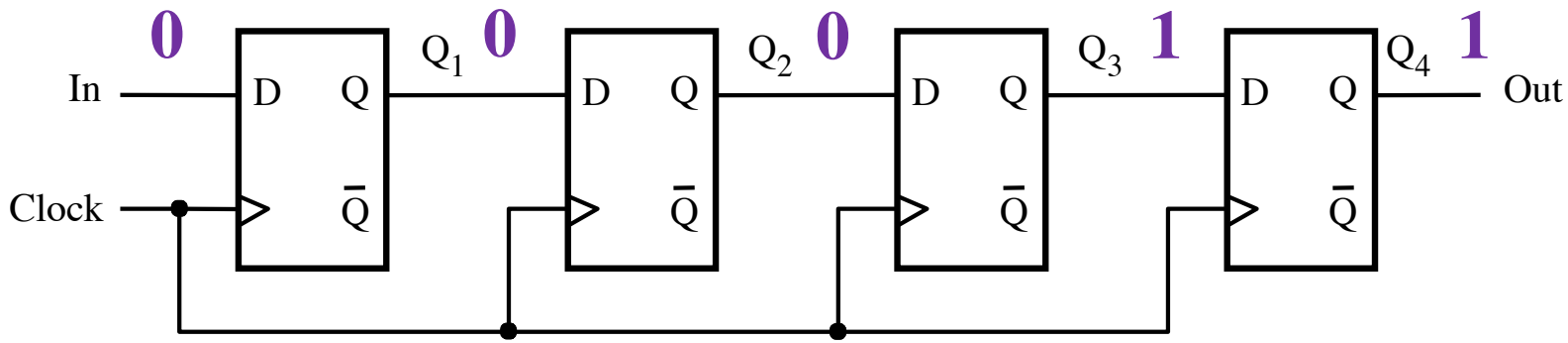


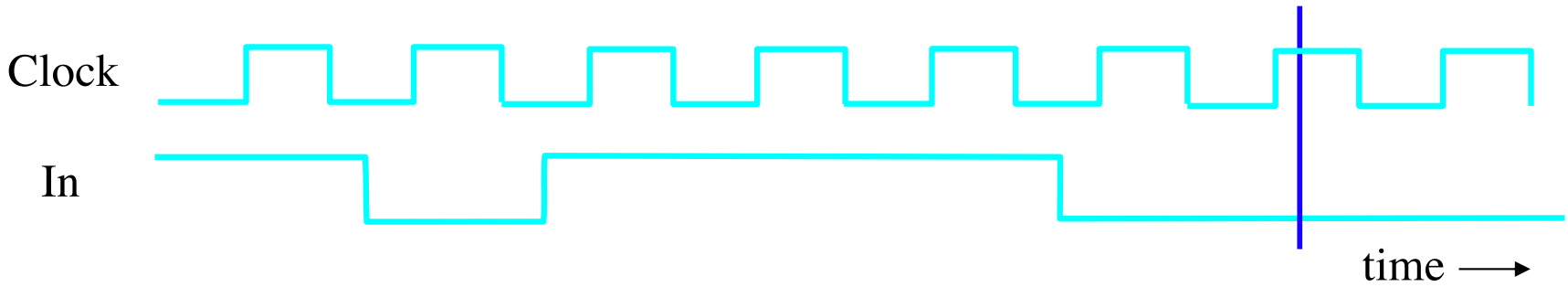
Shift Register Simulation



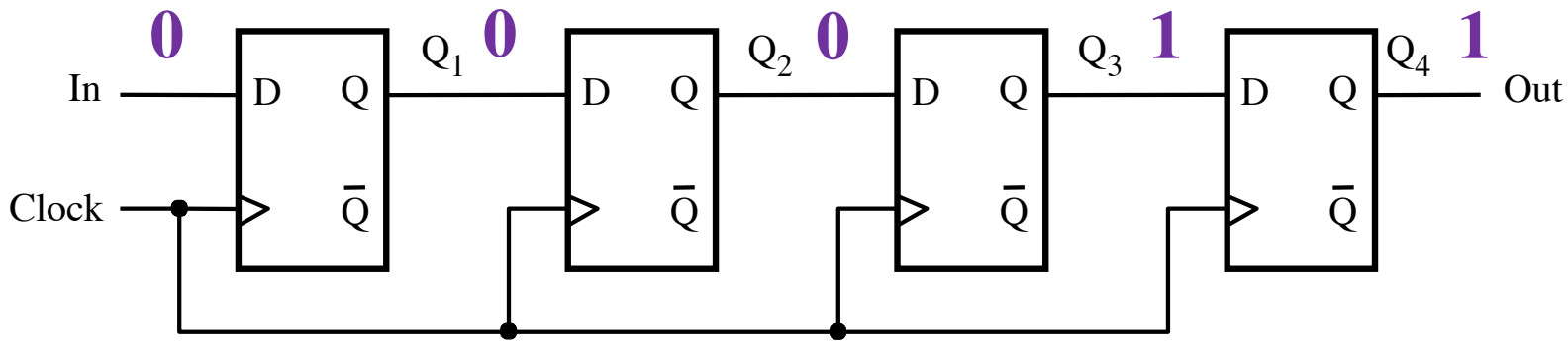


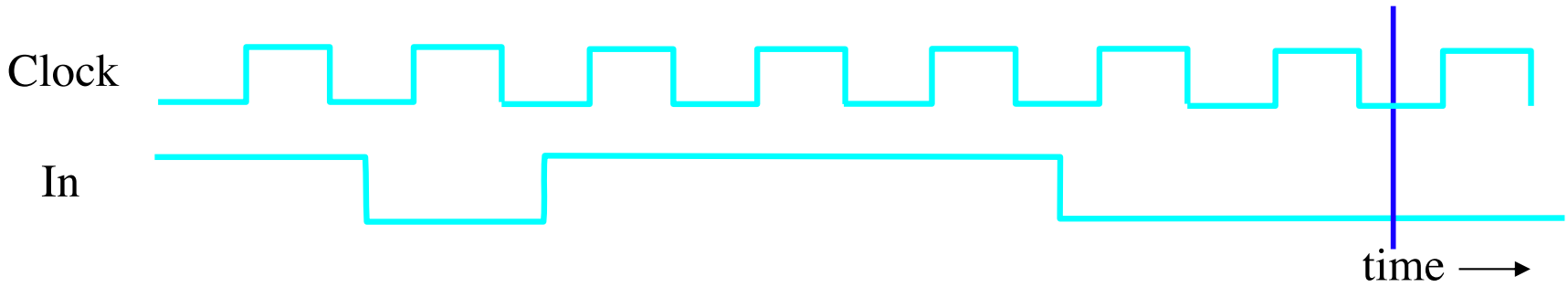
Shift Register Simulation



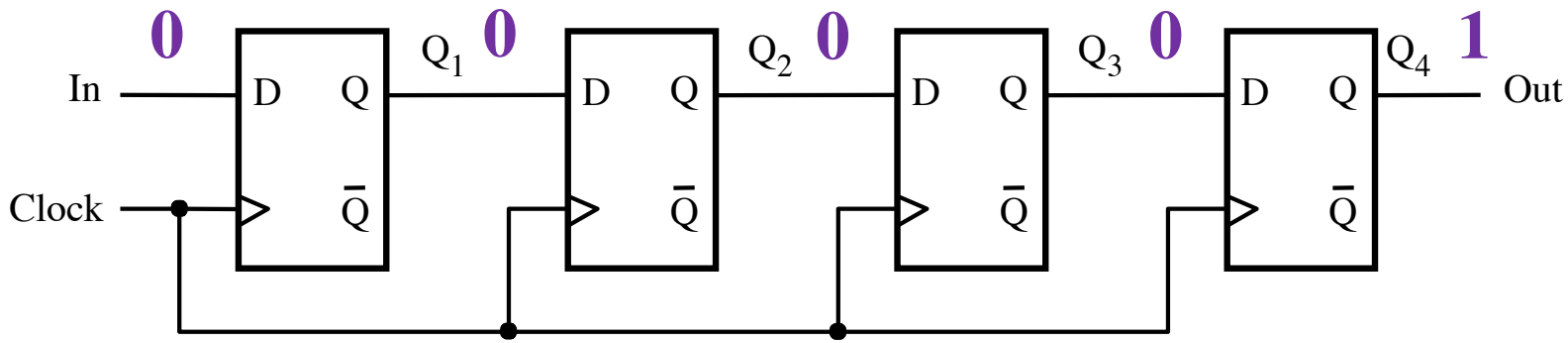


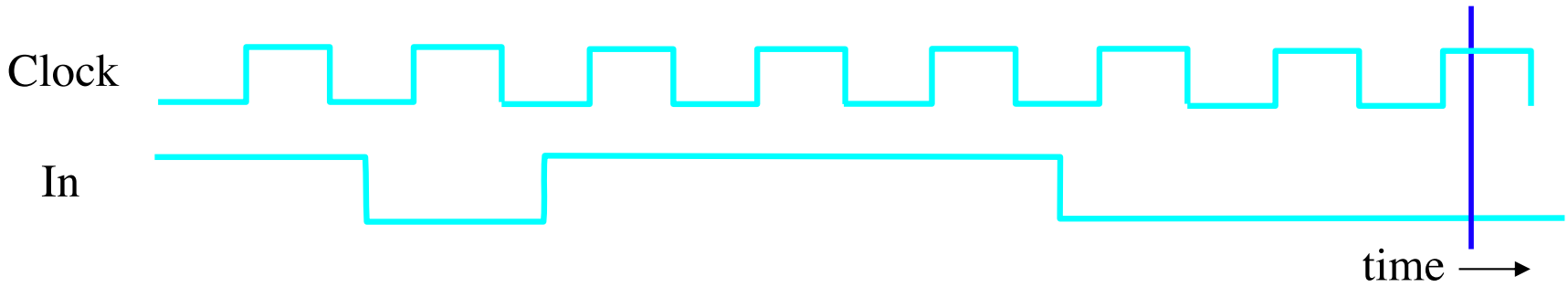
Shift Register Simulation



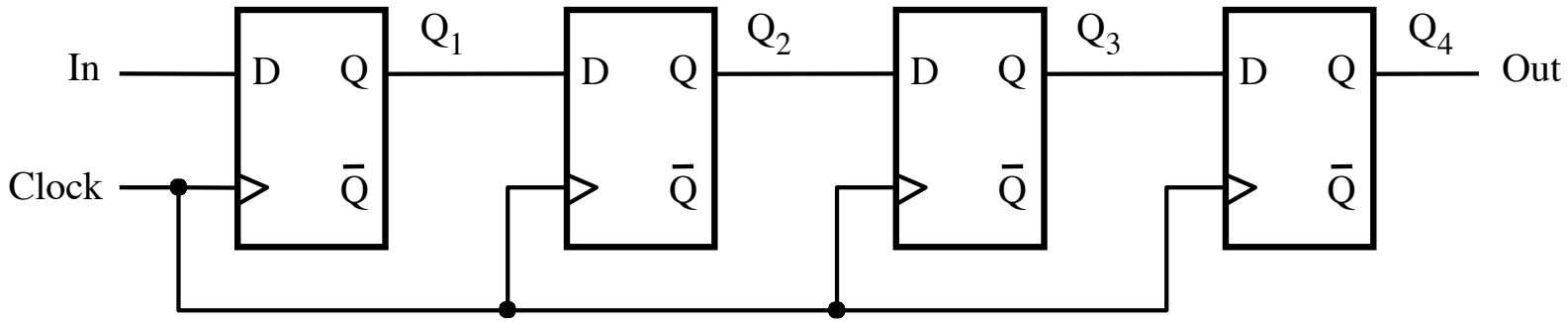


Shift Register Simulation





A simple shift register



(a) Circuit

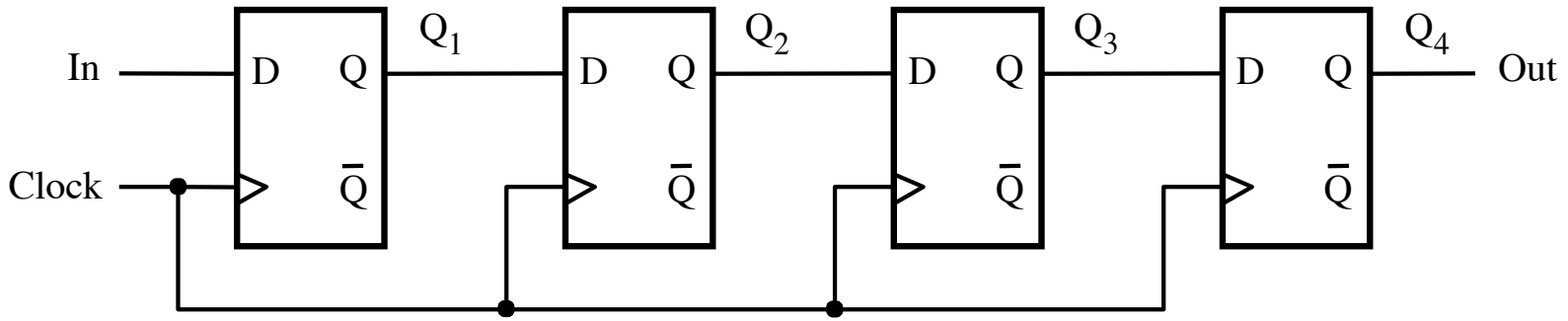
time

	In	Q_1	Q_2	Q_3	Q_4 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

(b) A sample sequence

[Figure 5.17 from the textbook]

A simple shift register



(a) Circuit

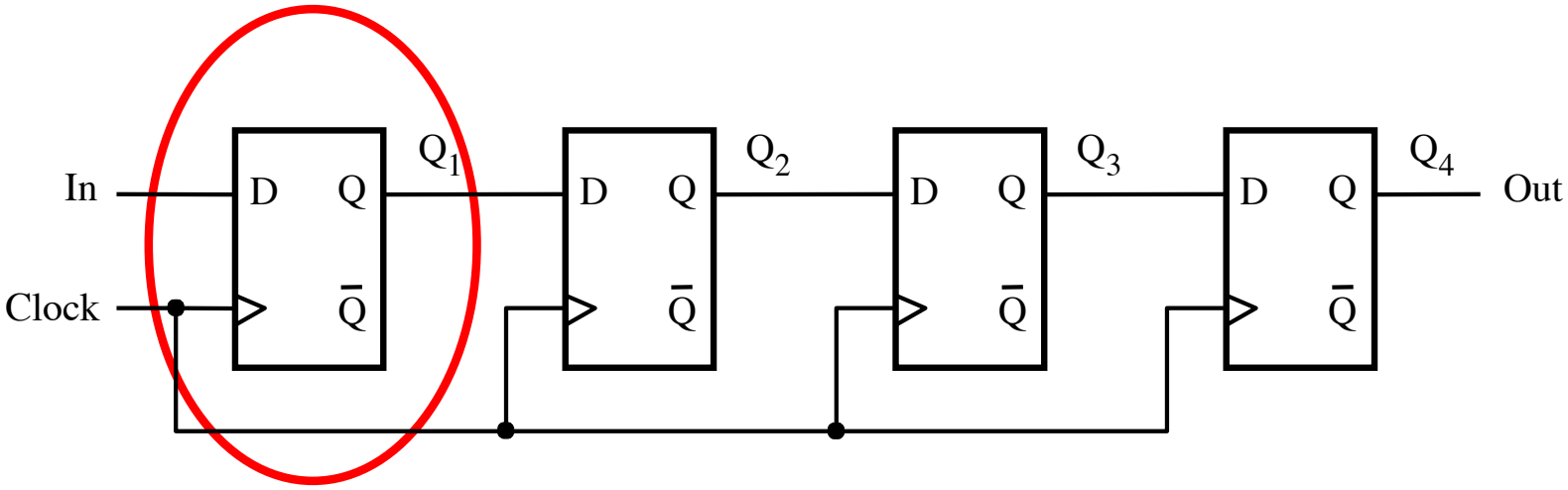
time

	In	Q_1	Q_2	Q_3	Q_4 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1
	0	0	0	0	1

The simulation goes one step further

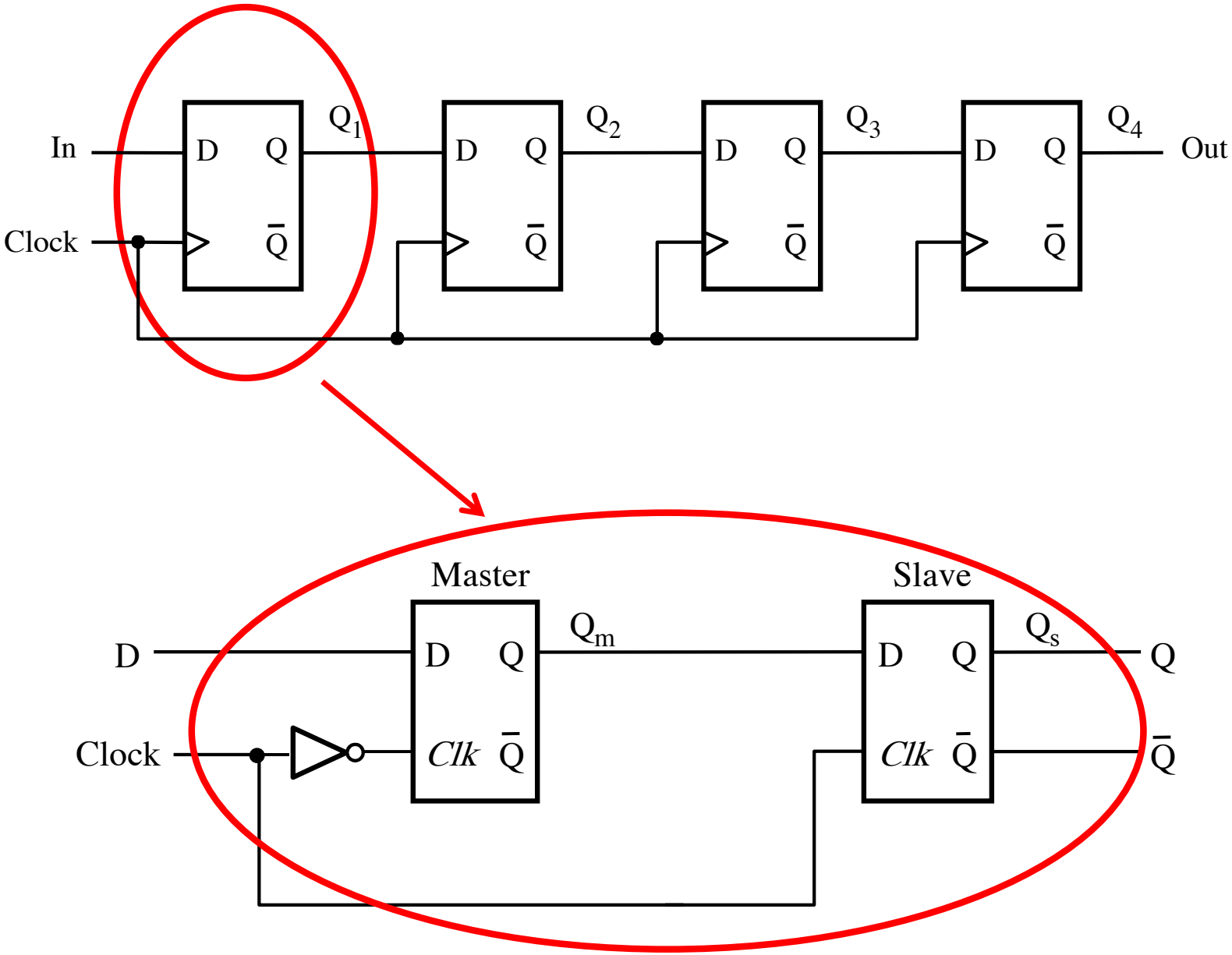
[Figure 5.17 from the textbook]

A simple shift register

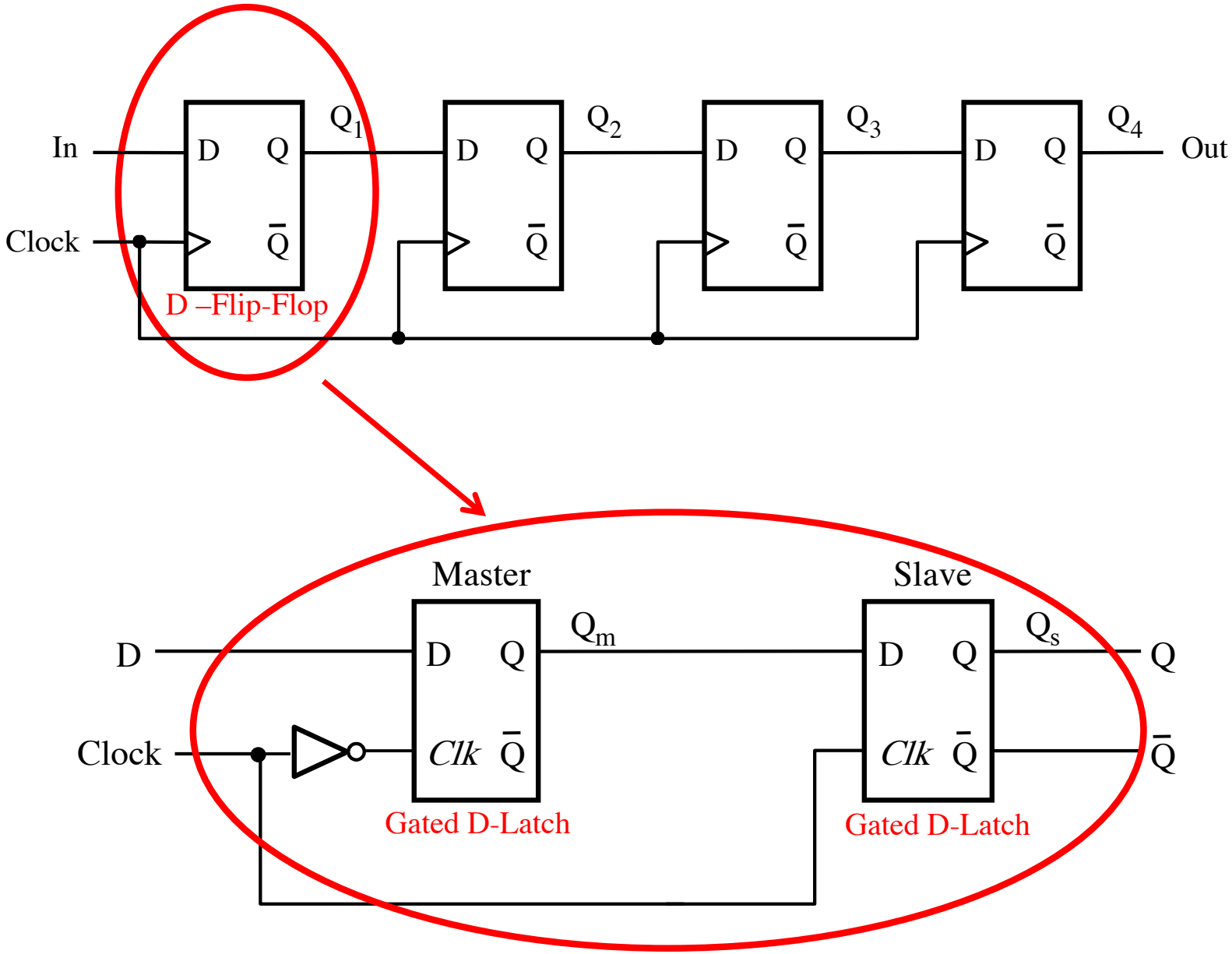


Positive-edge-triggered
D Flip-Flop

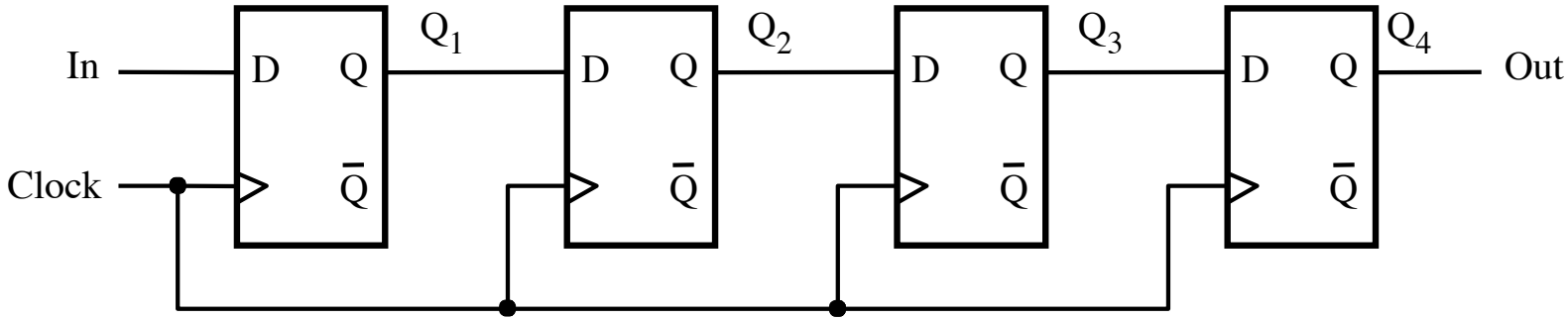
A simple shift register



A simple shift register



A simple shift register



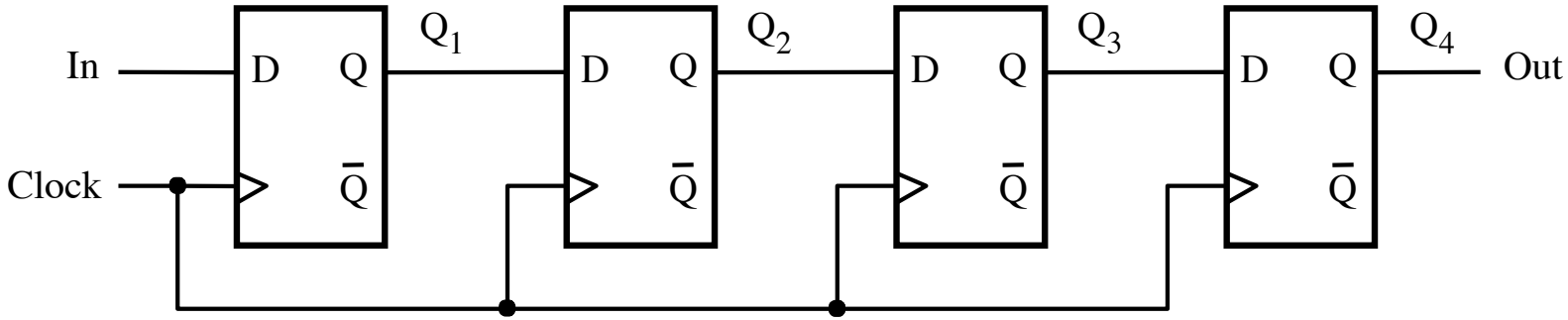
You need 2 latches to make a flip-flop

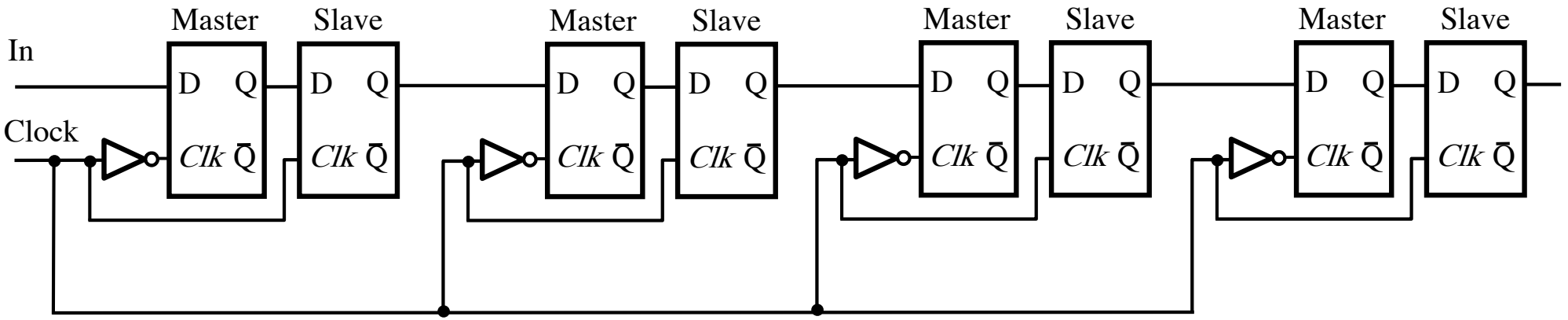
=

4-bit Register

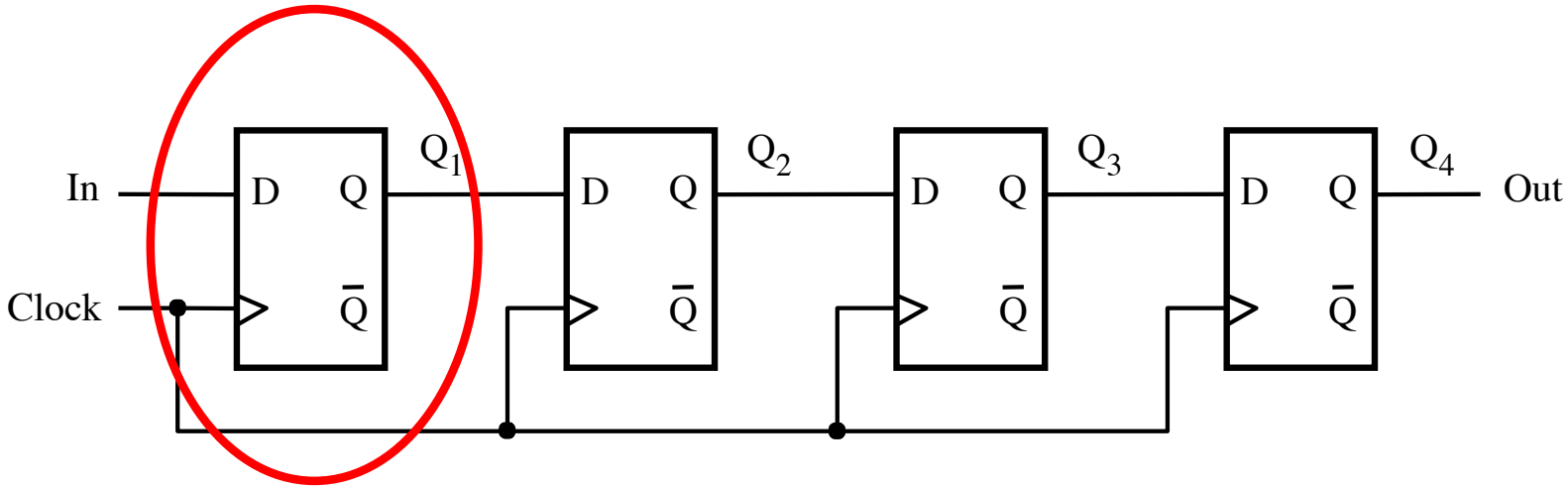
4-bit Register

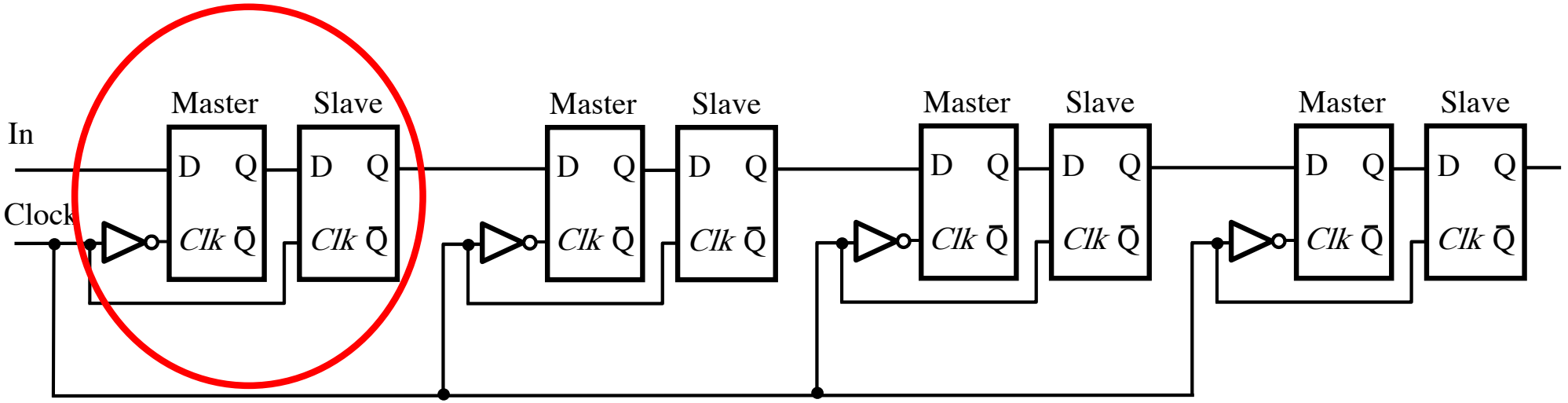
A simple shift register



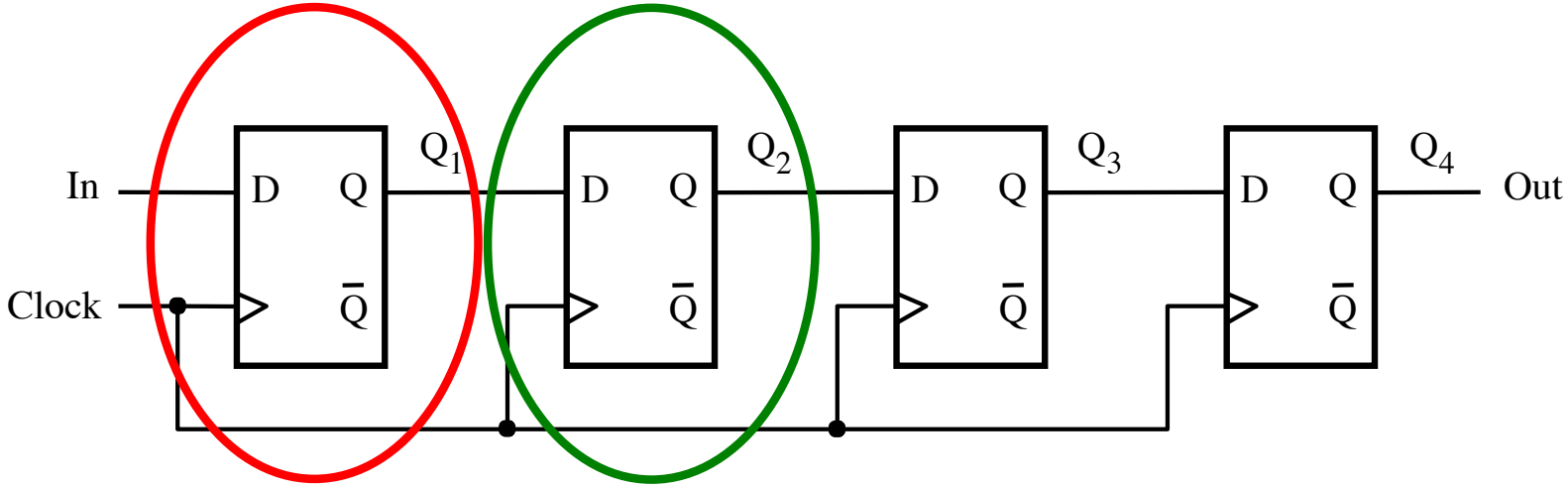


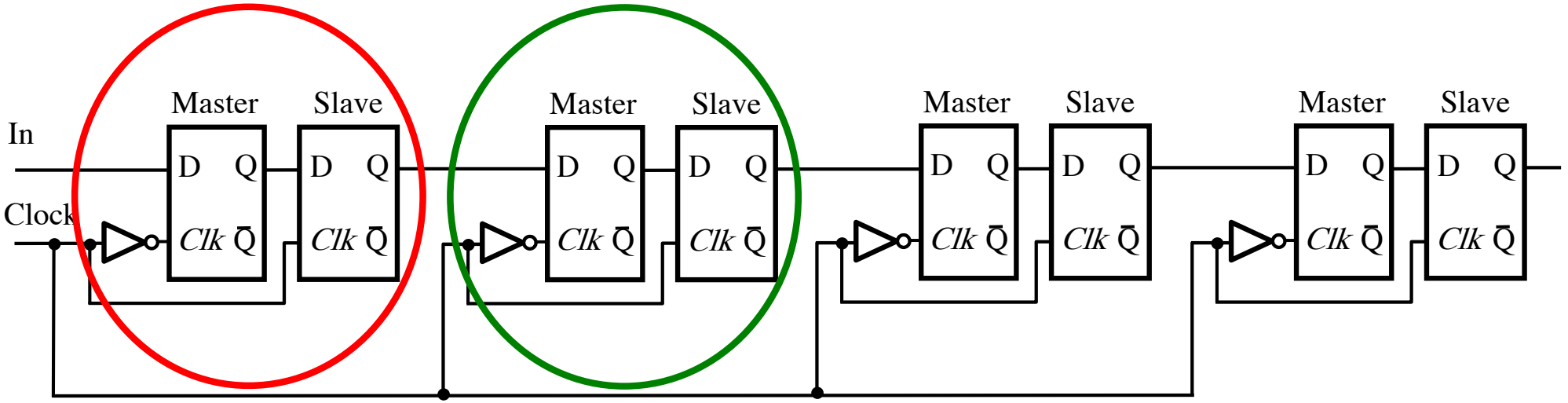
A simple shift register



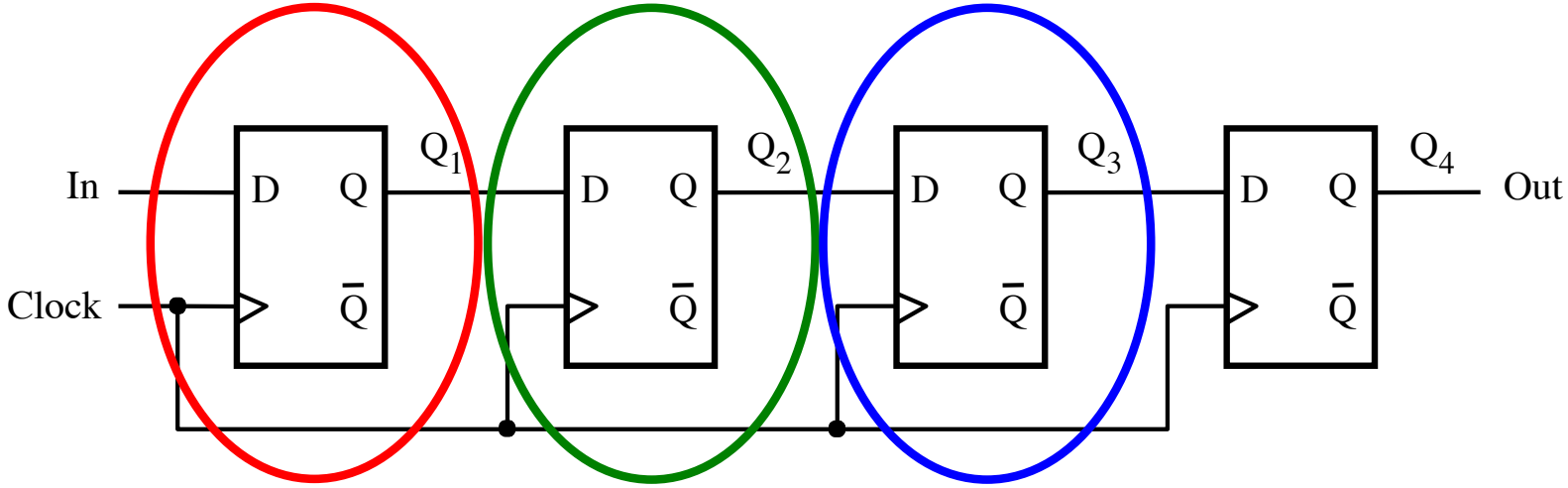


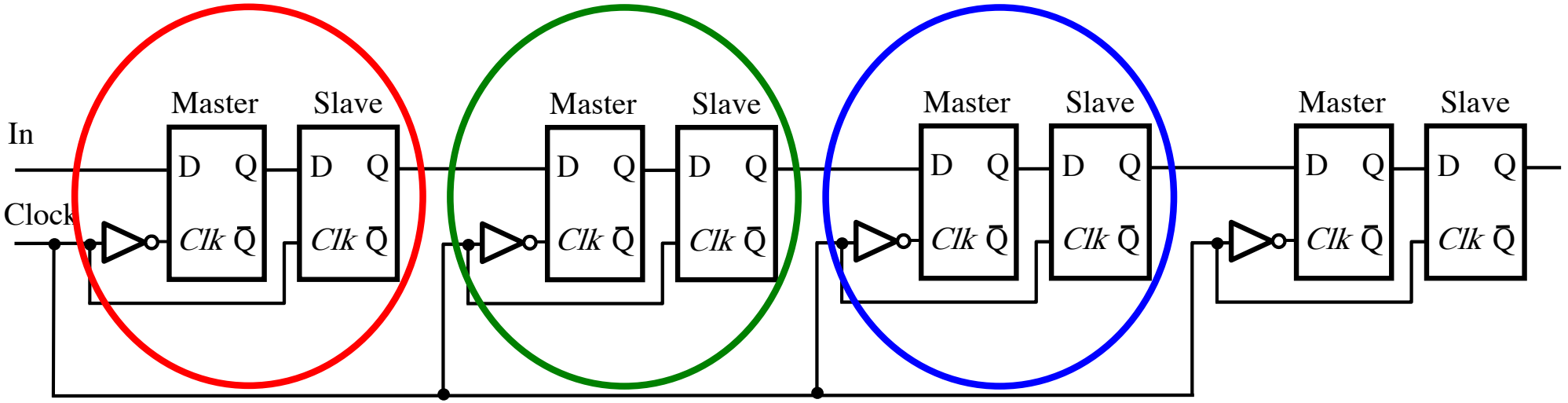
A simple shift register



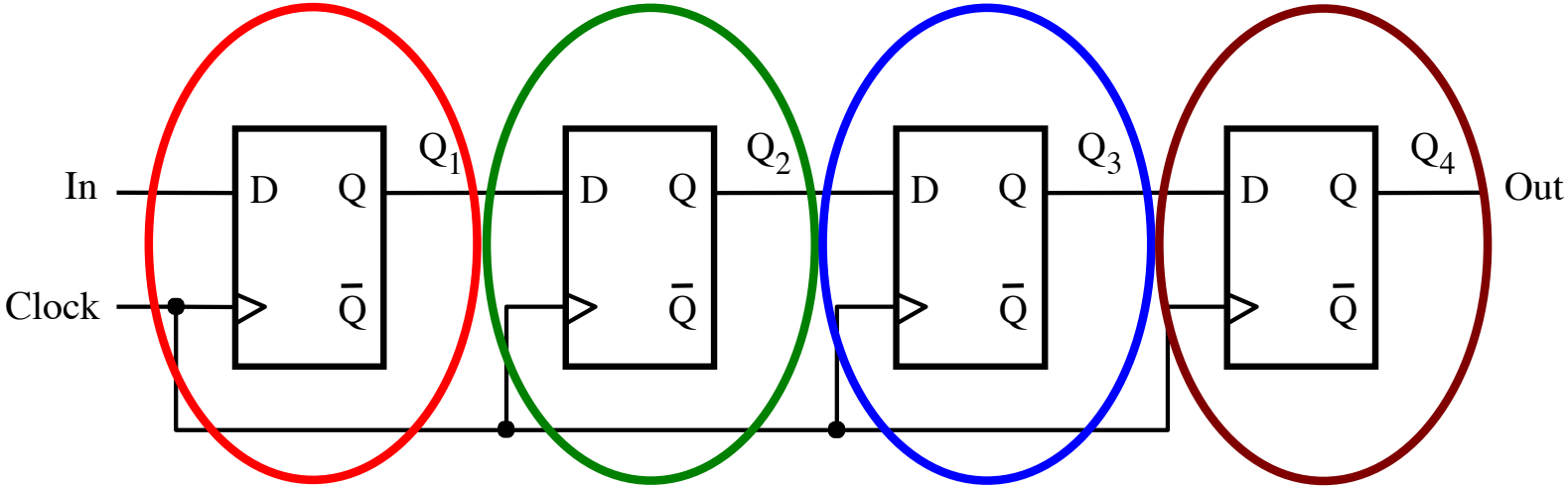


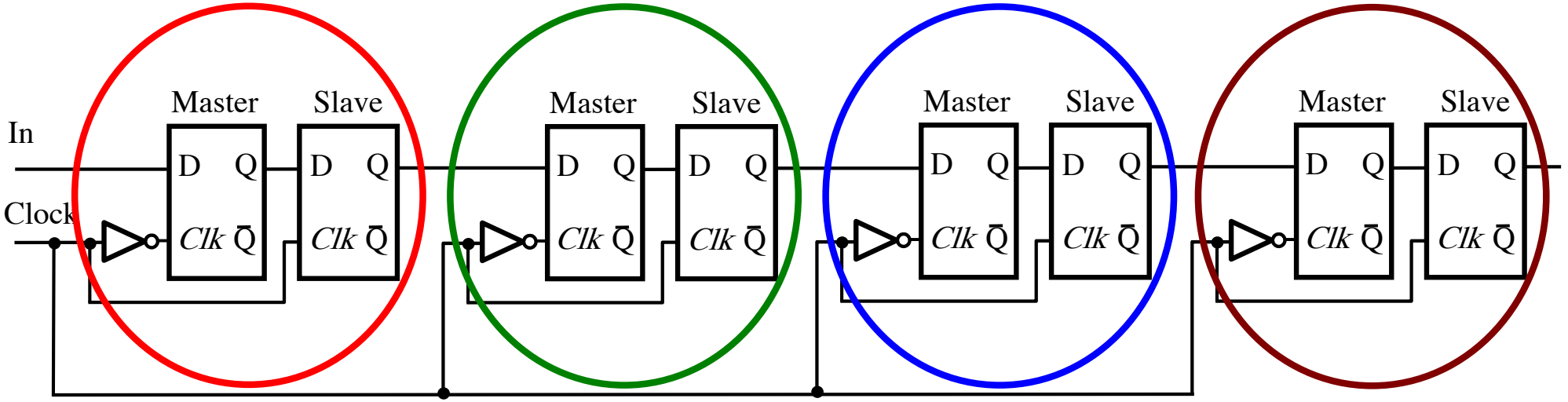
A simple shift register



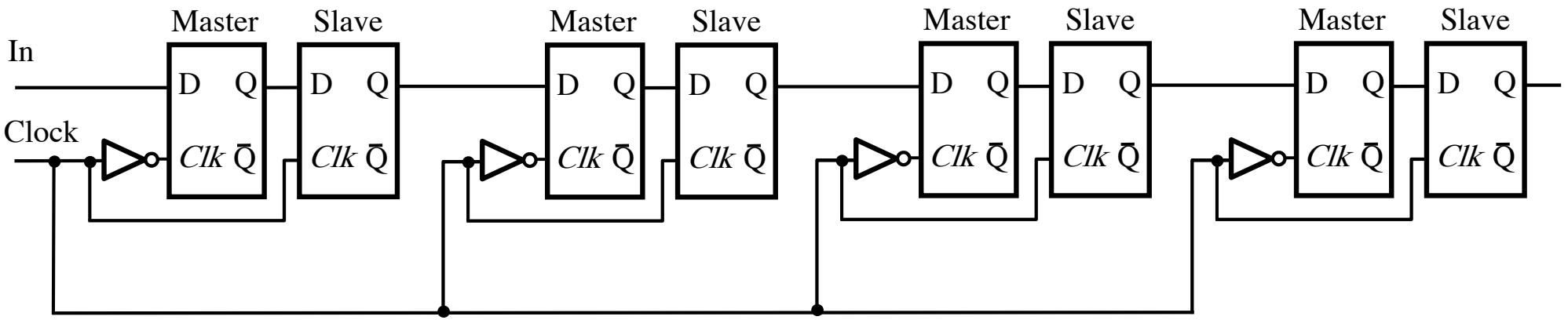


A simple shift register

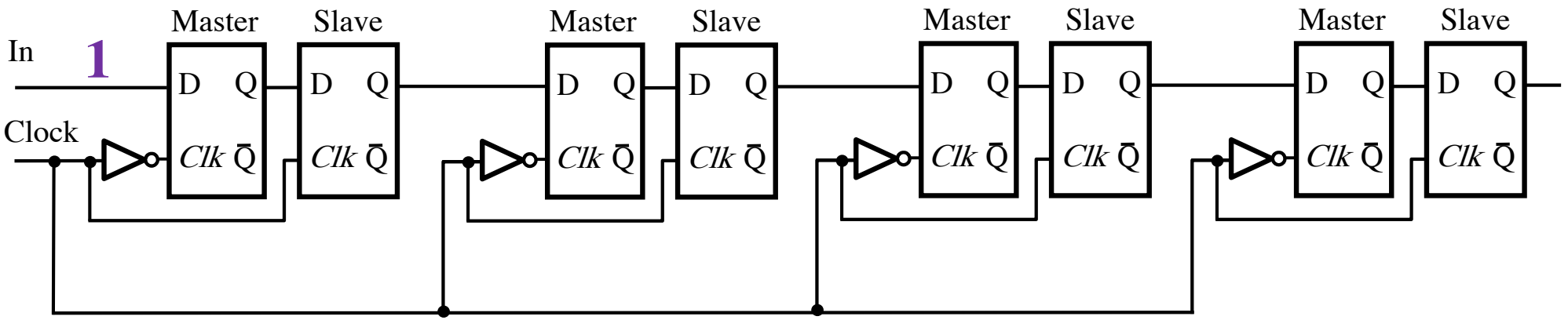




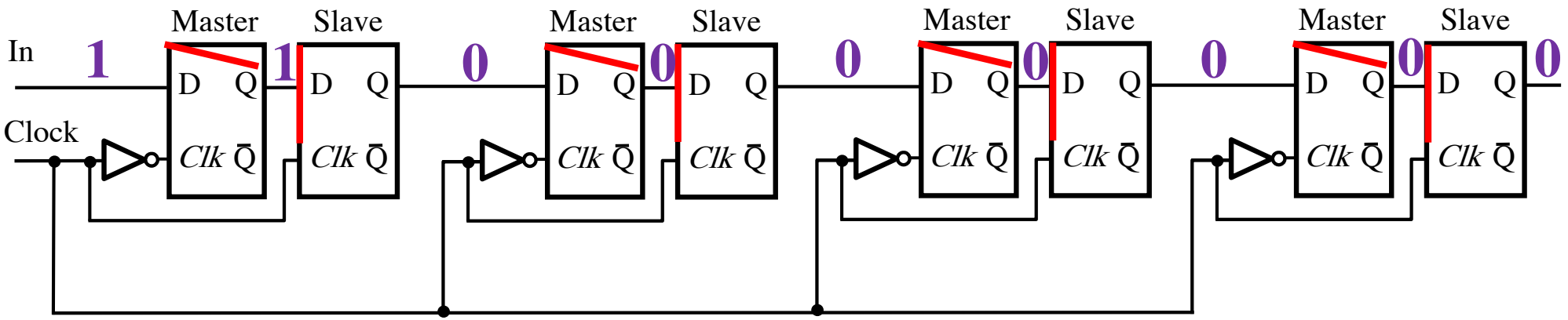
Simulating a shift register

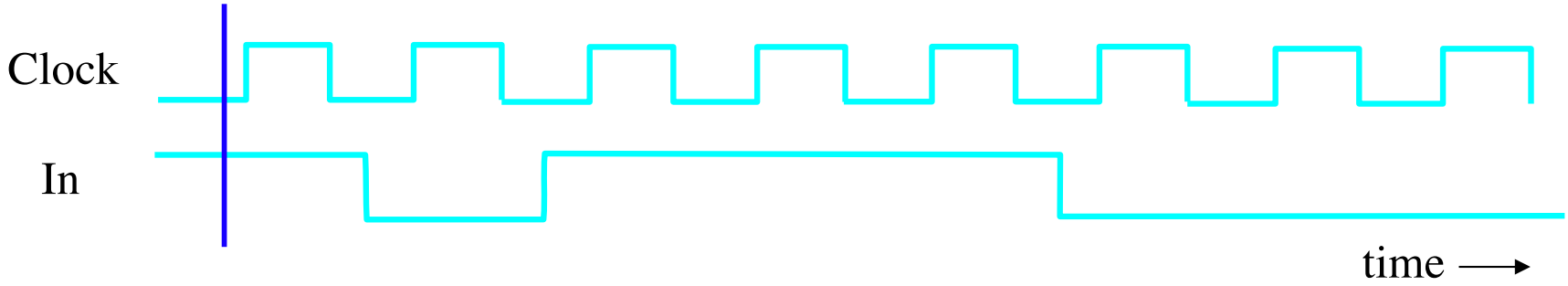


Simulating a shift register

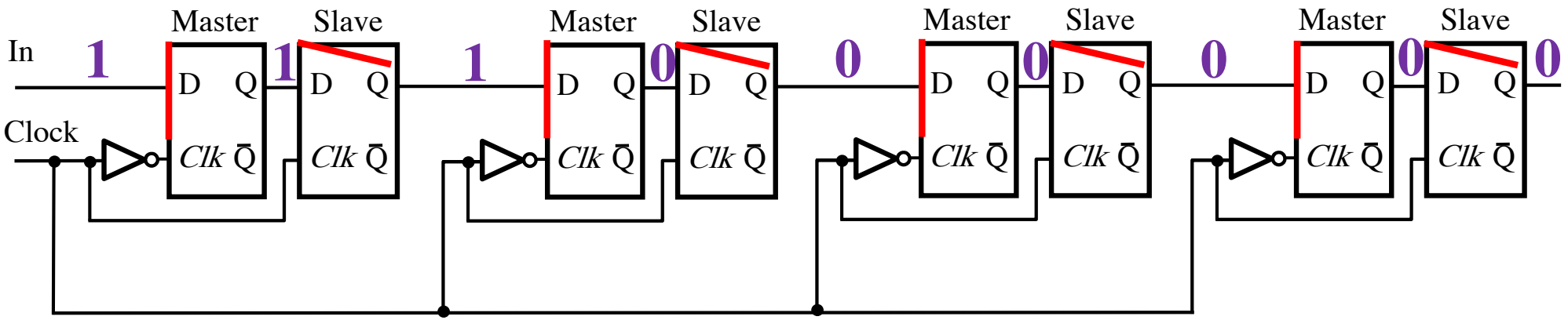


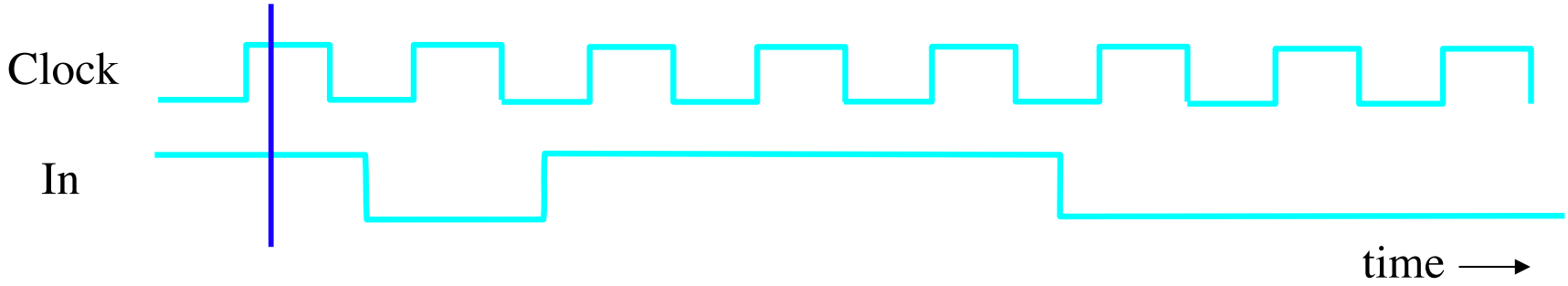
Simulating a shift register



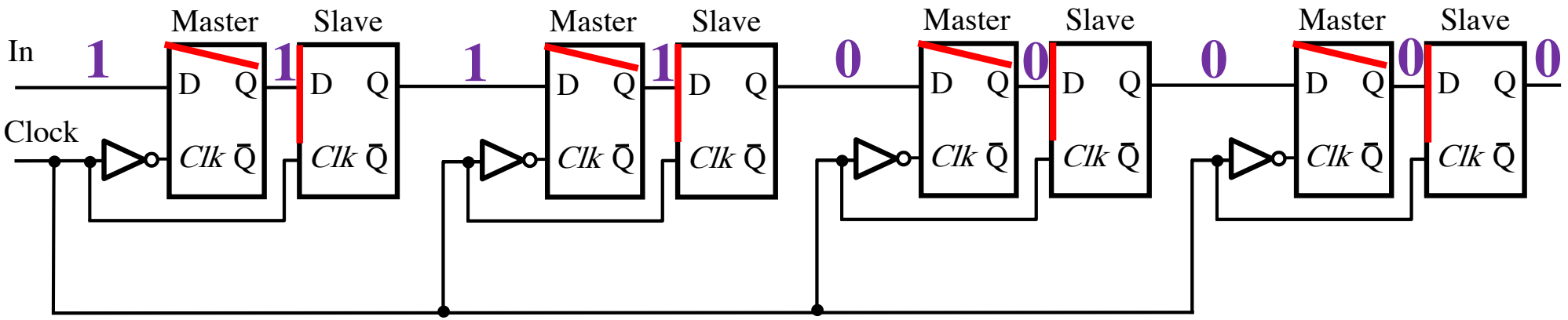


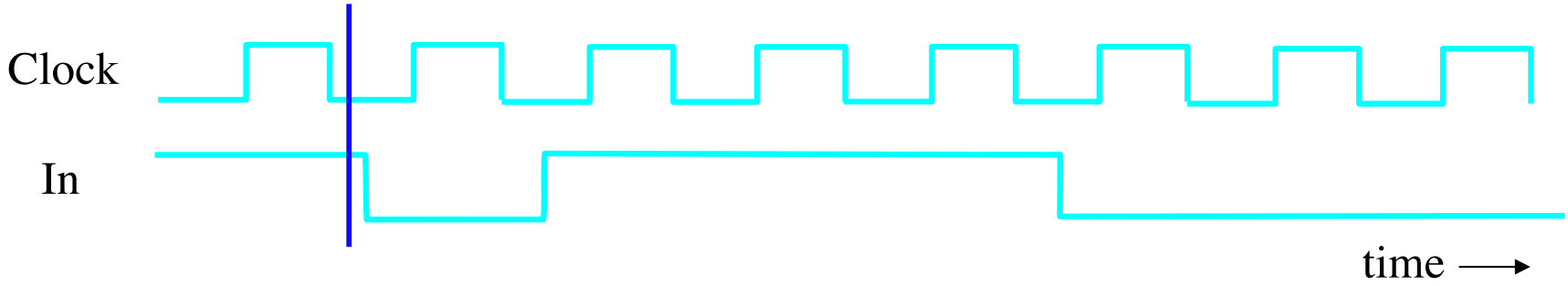
Simulating a shift register



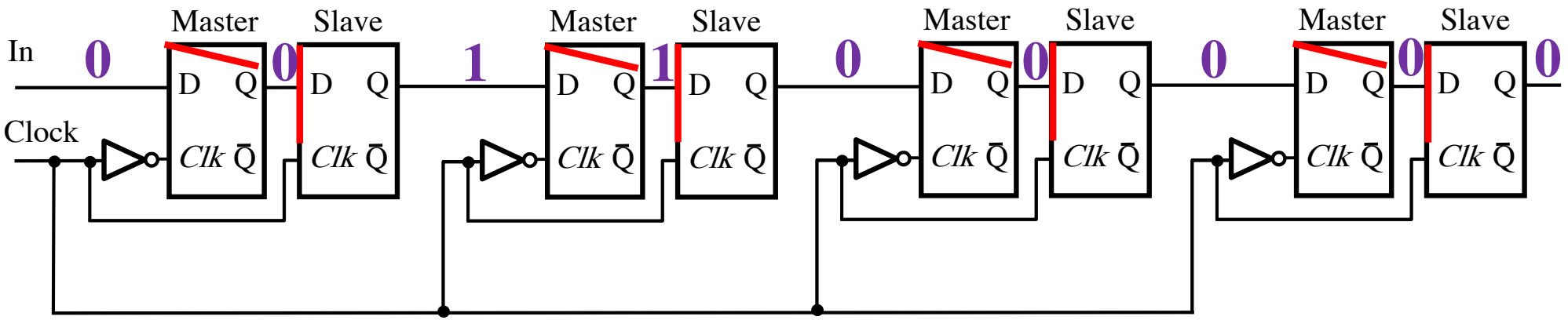


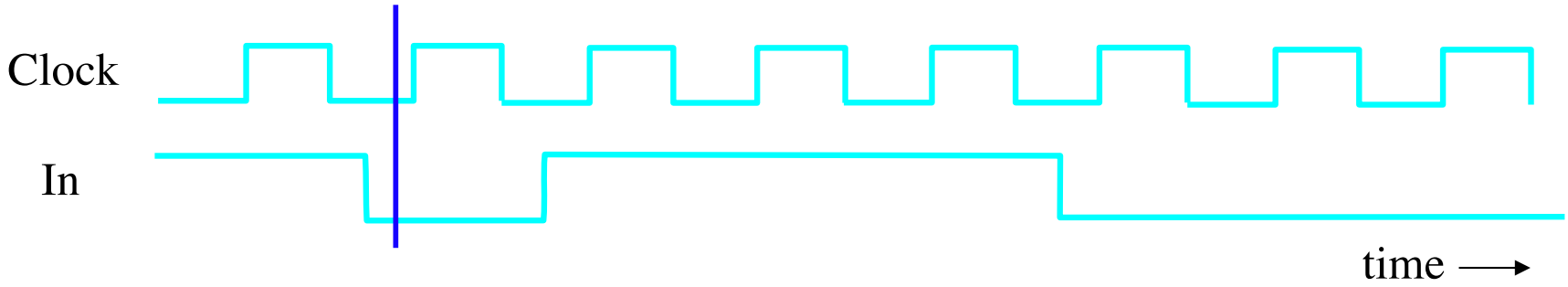
Simulating a shift register



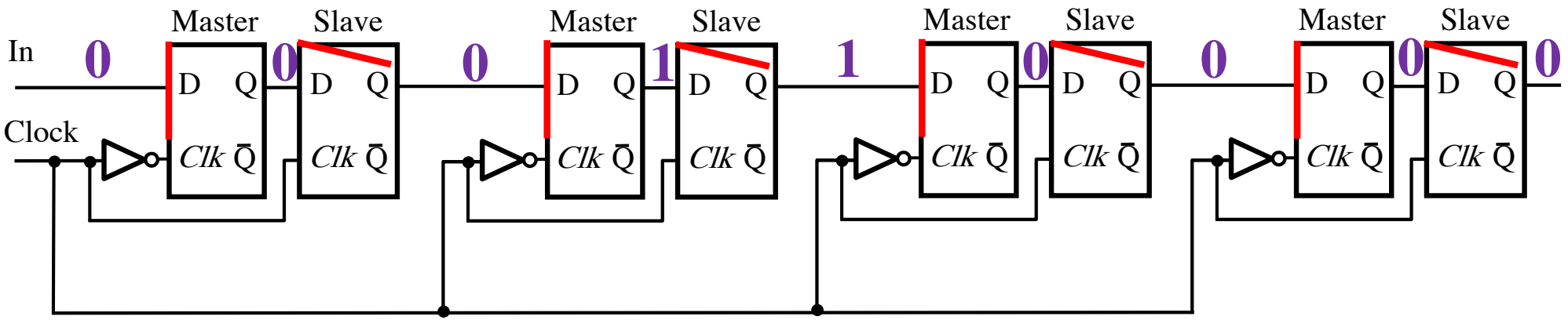


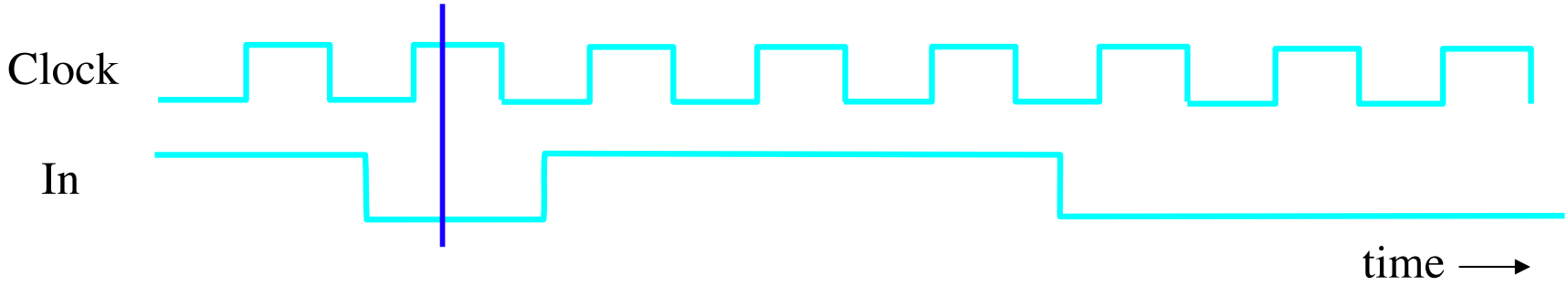
Simulating a shift register



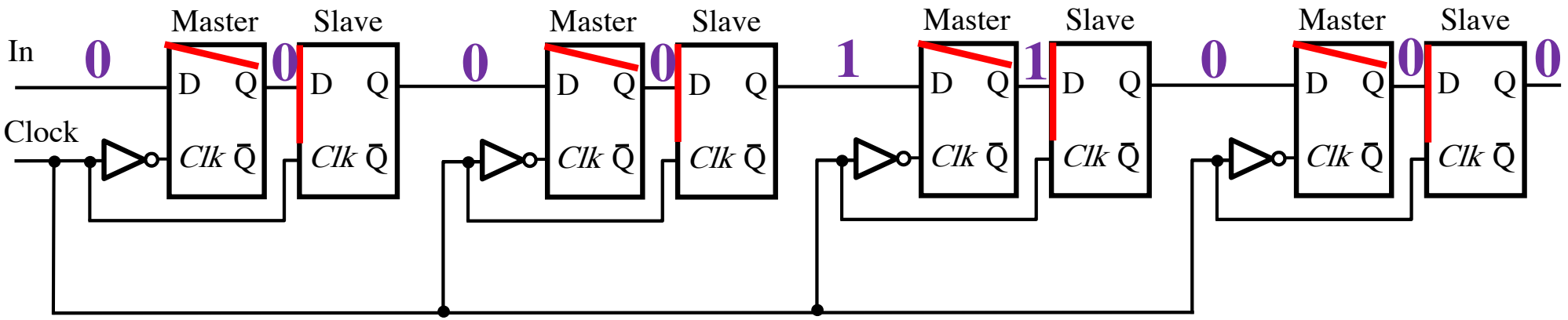


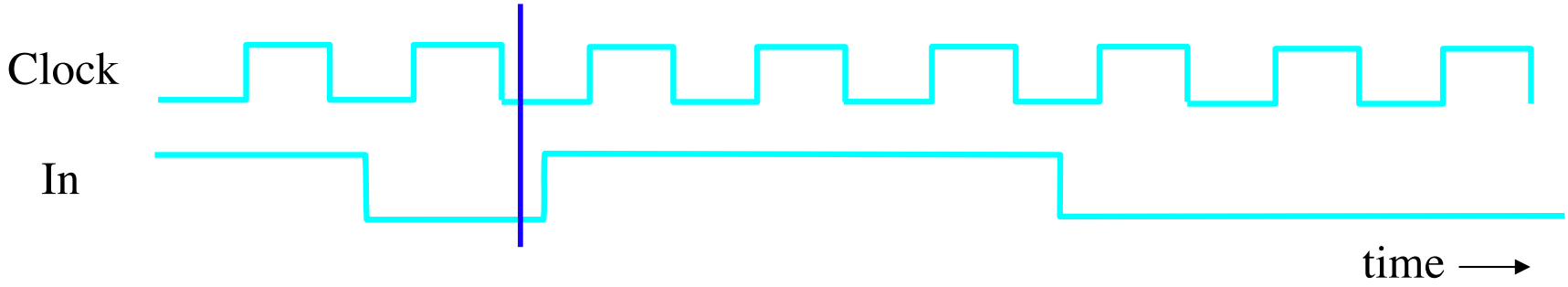
Simulating a shift register



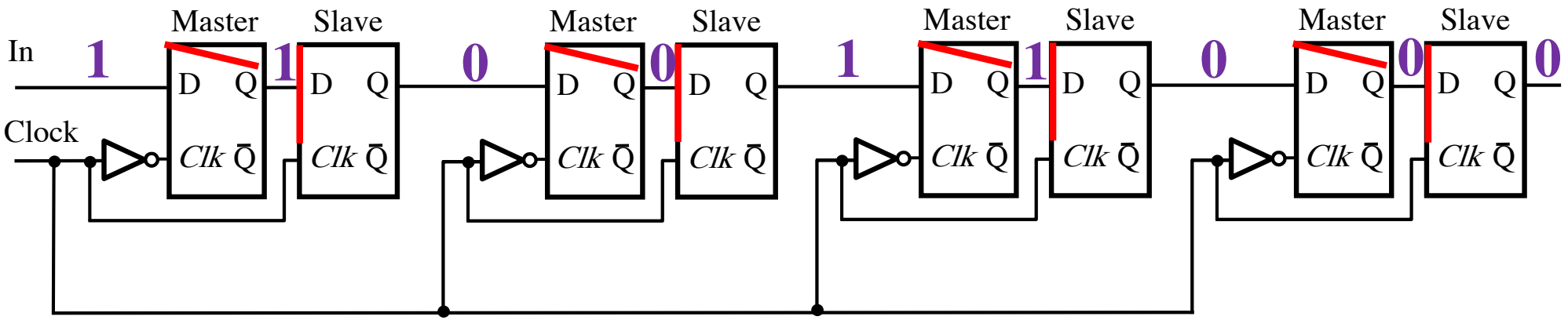


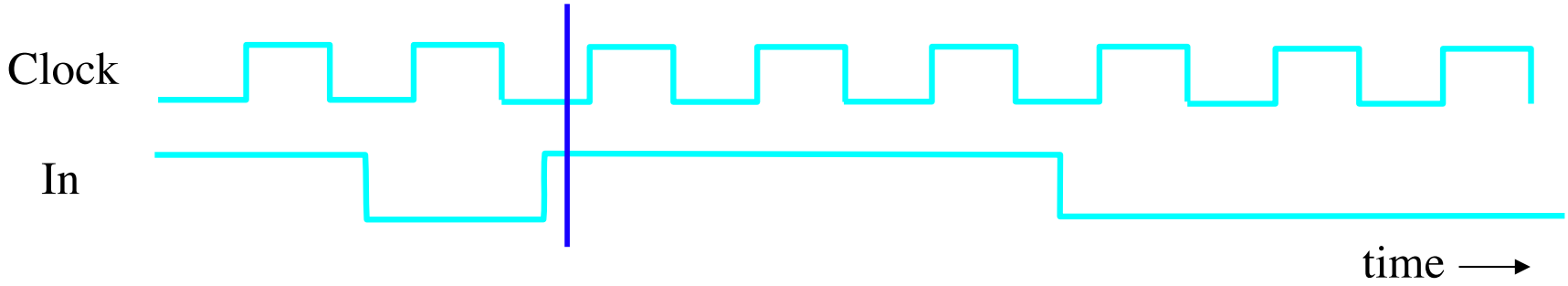
Simulating a shift register



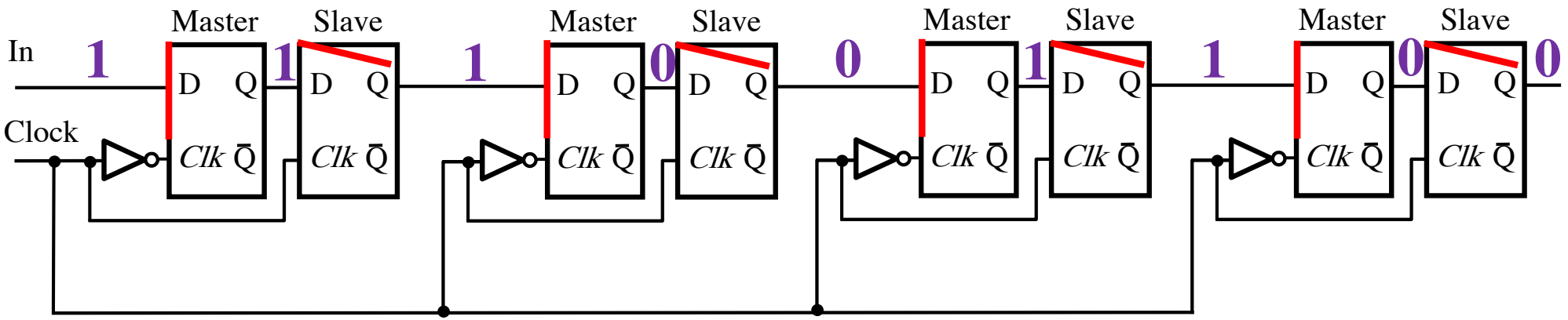


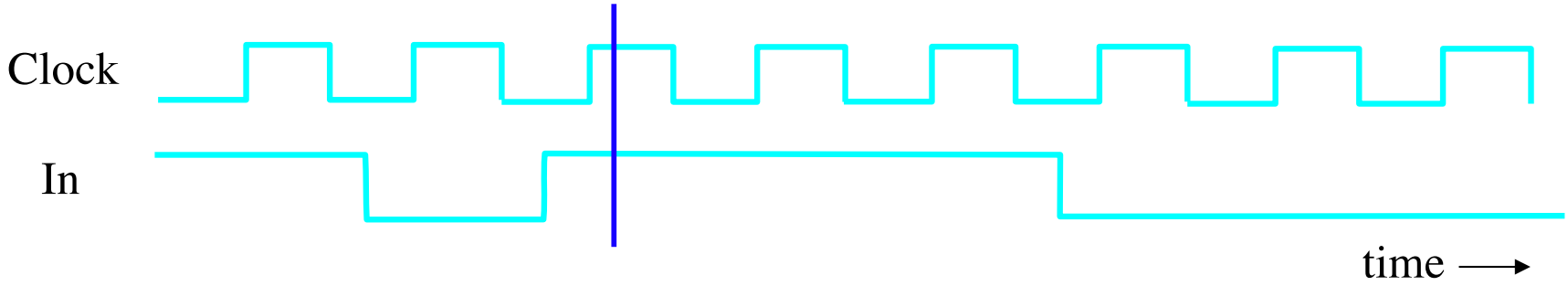
Simulating a shift register



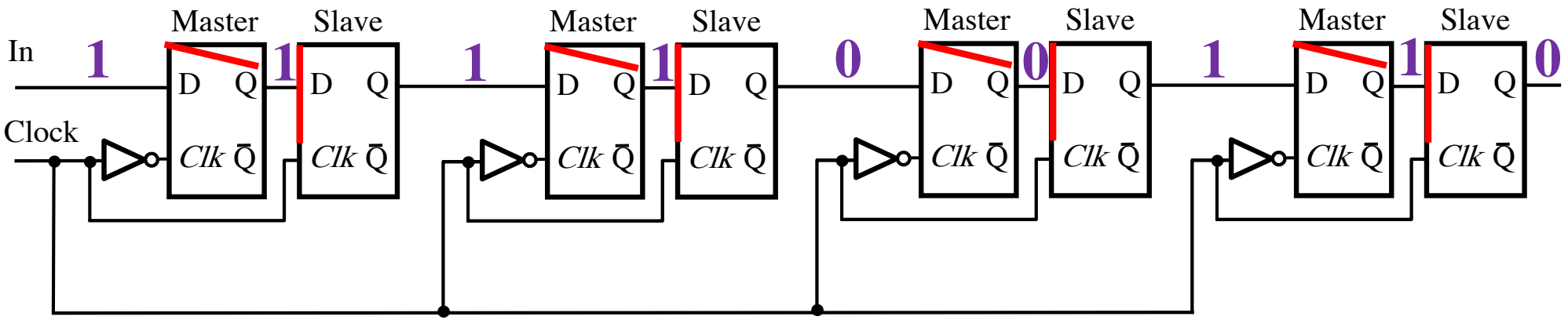


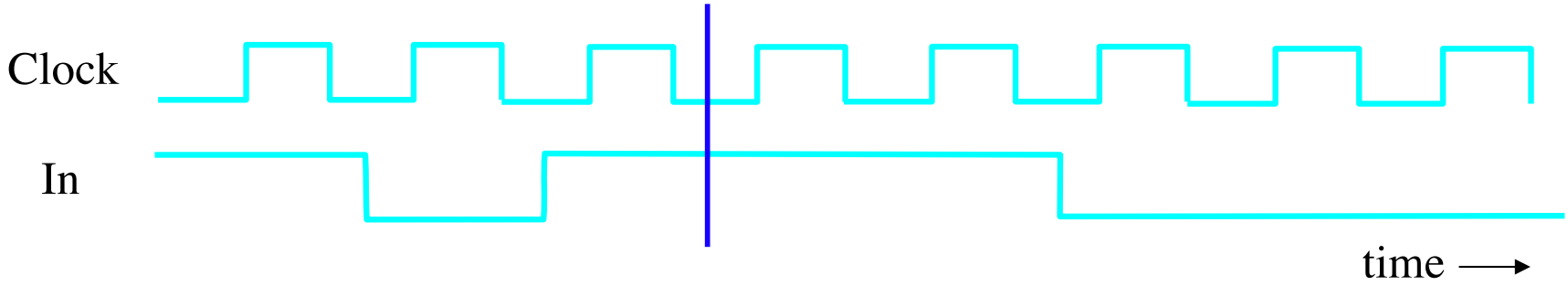
Simulating a shift register



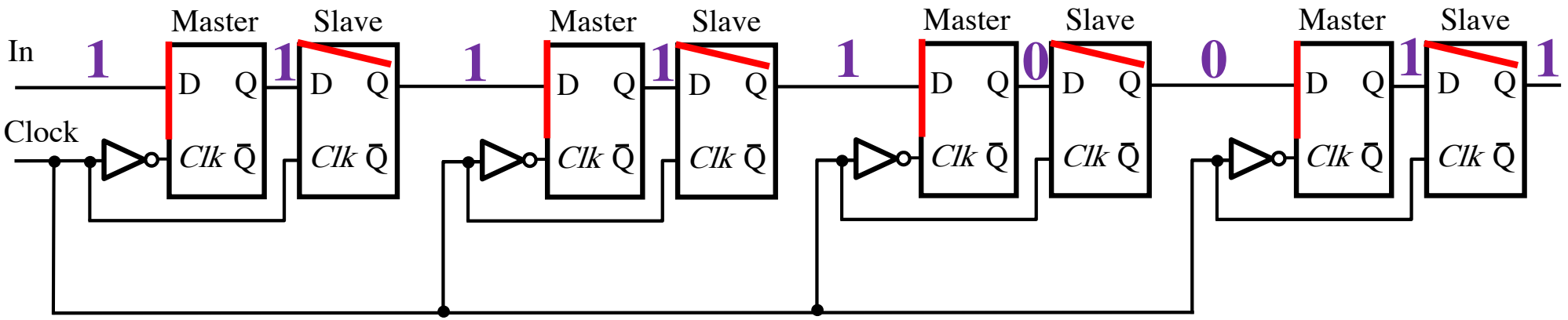


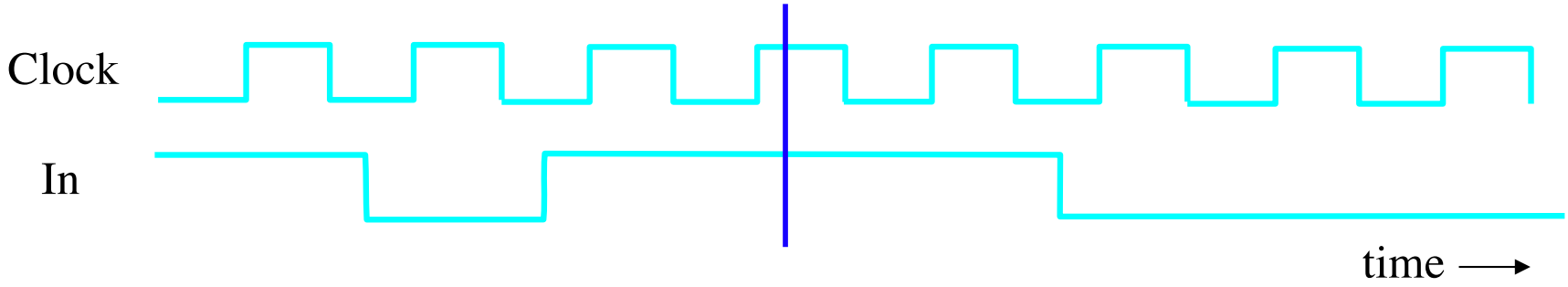
Simulating a shift register



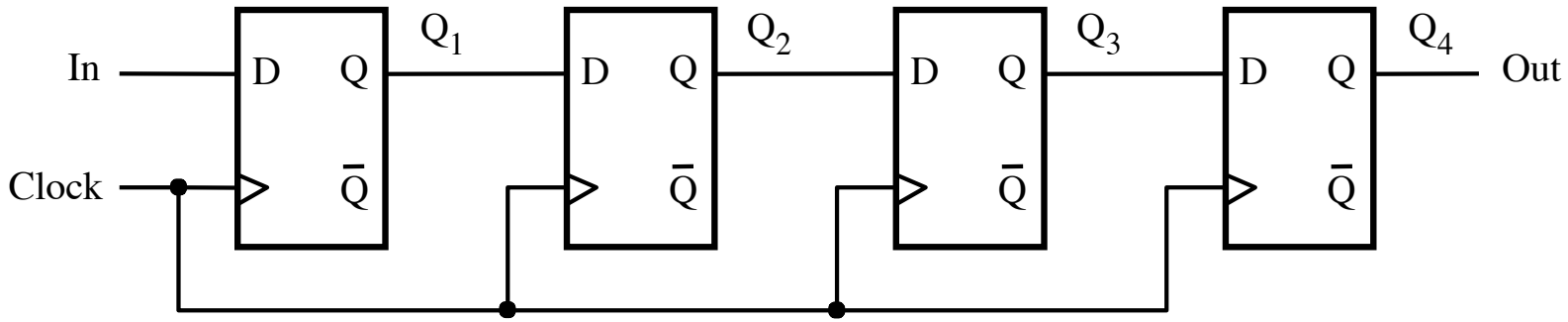


Simulating a shift register





A simple shift register



(a) Circuit

	In	Q_1	Q_2	Q_3	Q_4 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

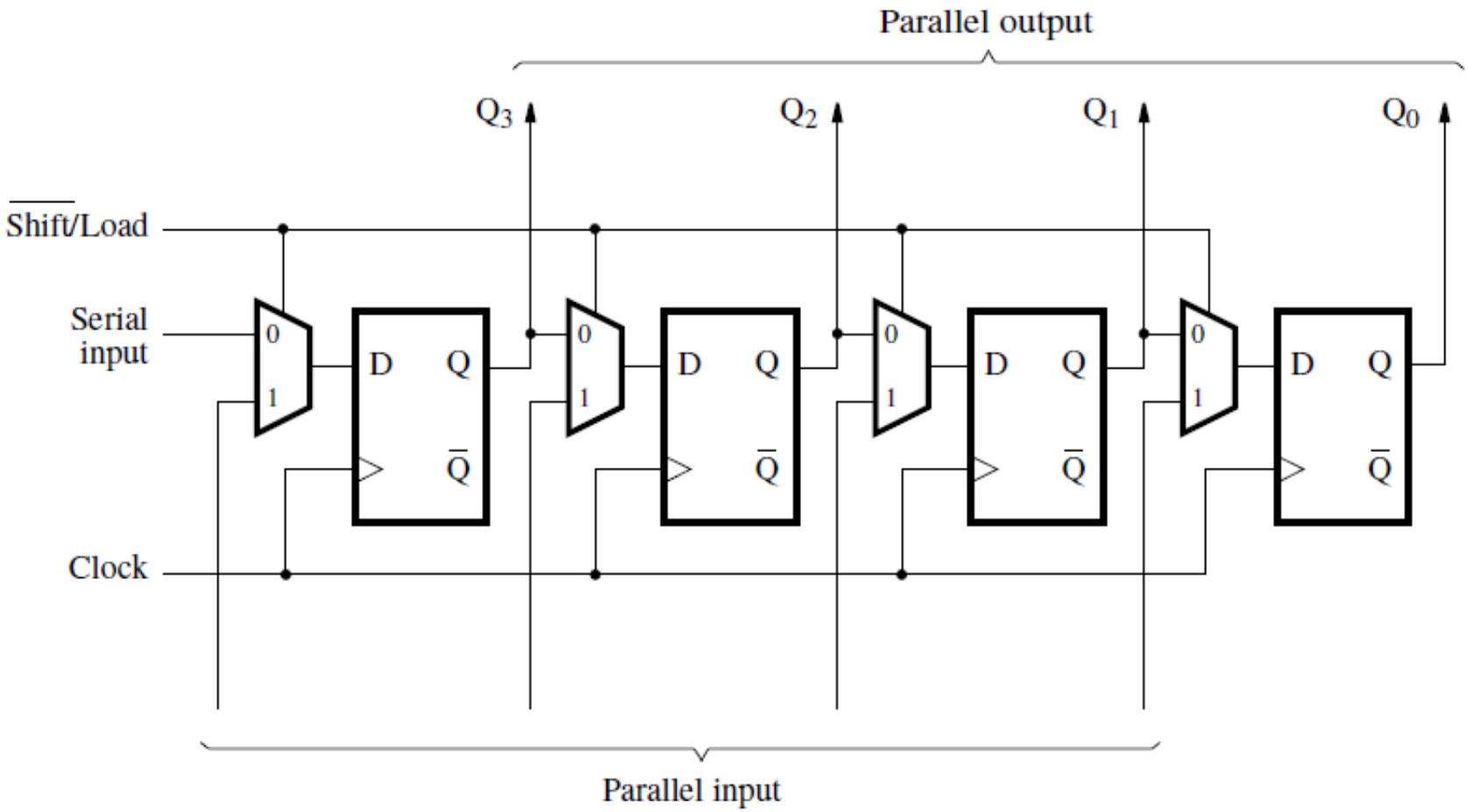
(b) A sample sequence

This simulation goes only up to here

[Figure 5.17 from the textbook]

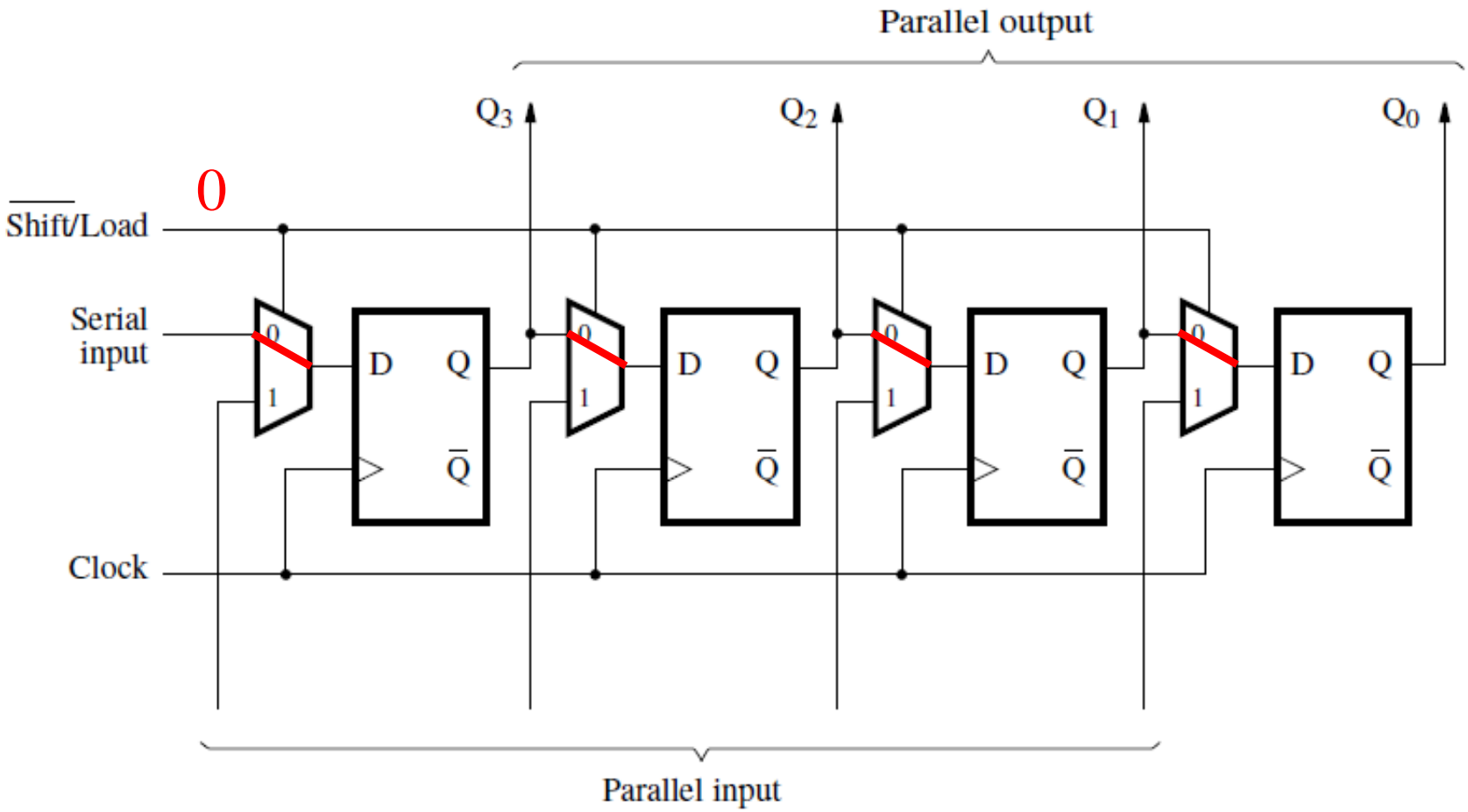
Parallel-Access Shift Register

Parallel-access shift register



[Figure 5.18 from the textbook]

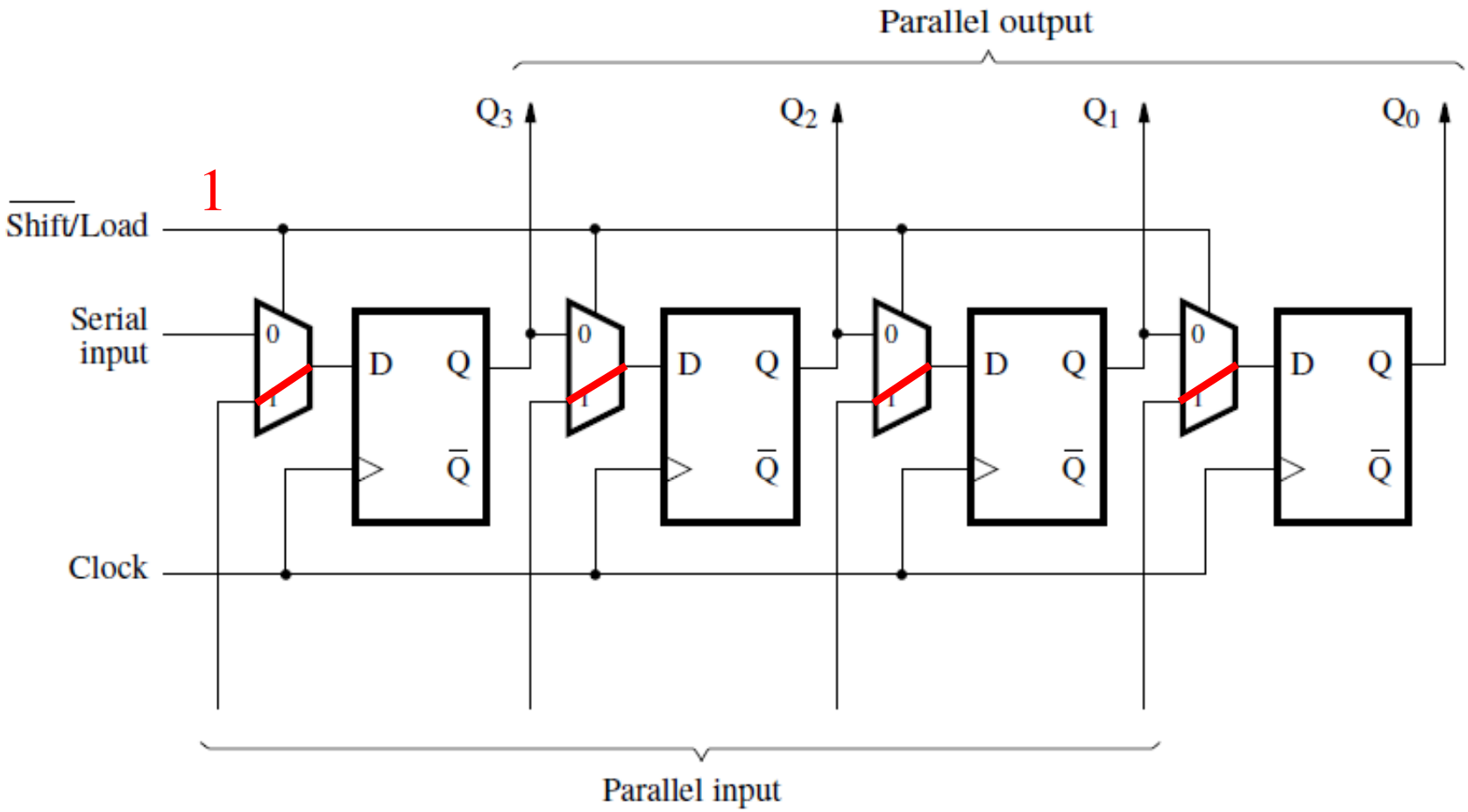
Parallel-access shift register



When Load=0, this behaves like a shift register.

[Figure 5.18 from the textbook]

Parallel-access shift register



When Load=1, this behaves like a parallel-access register.

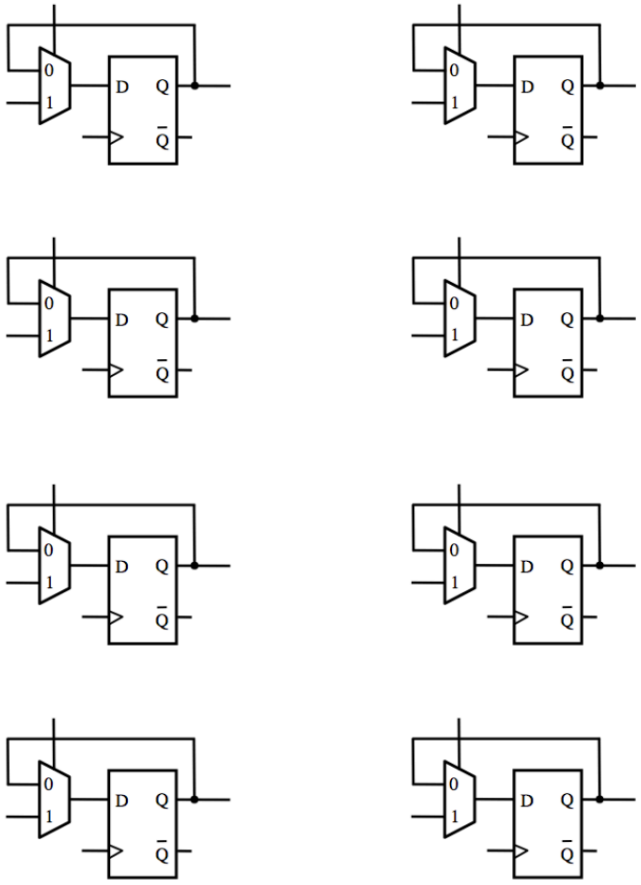
[Figure 5.18 from the textbook]

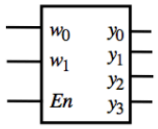
Register File

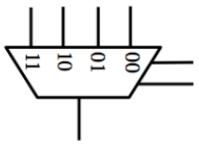
Motivation

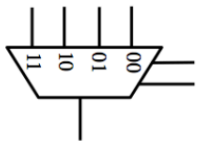
- We would like to build a circuit that can store several numbers which can be read or written independently.
- Each number is stored in a separate n-bit register.
- To write: a decoder selects which resister is enabled for writing. An input bus provides the values.
- To read: a set of multiplexers select which register will be read and copied to the output bus.
- Some register files come with two read ports.
  In those designs the multiplexer circuitry is doubled.

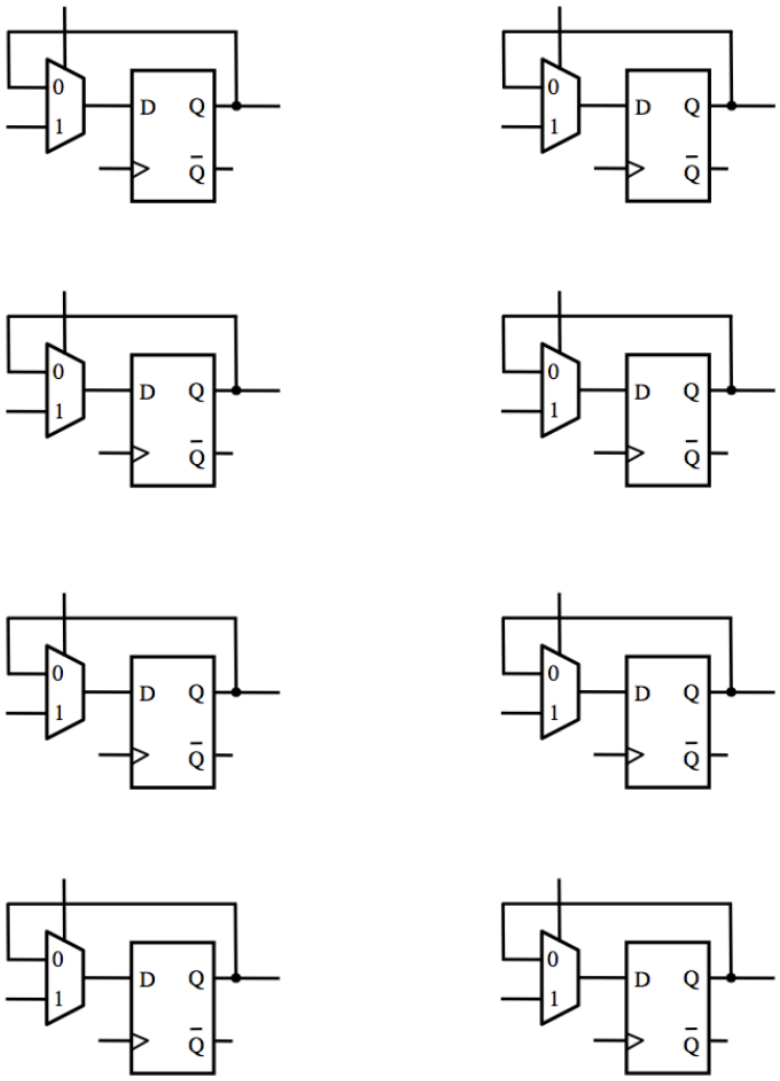
Complete the following circuit diagram to implement a register file with four 2-bit registers, one write port, one read port, and one write enable line.

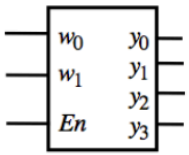


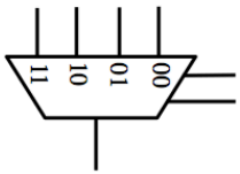


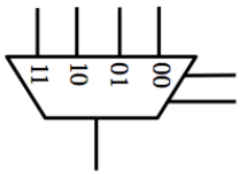


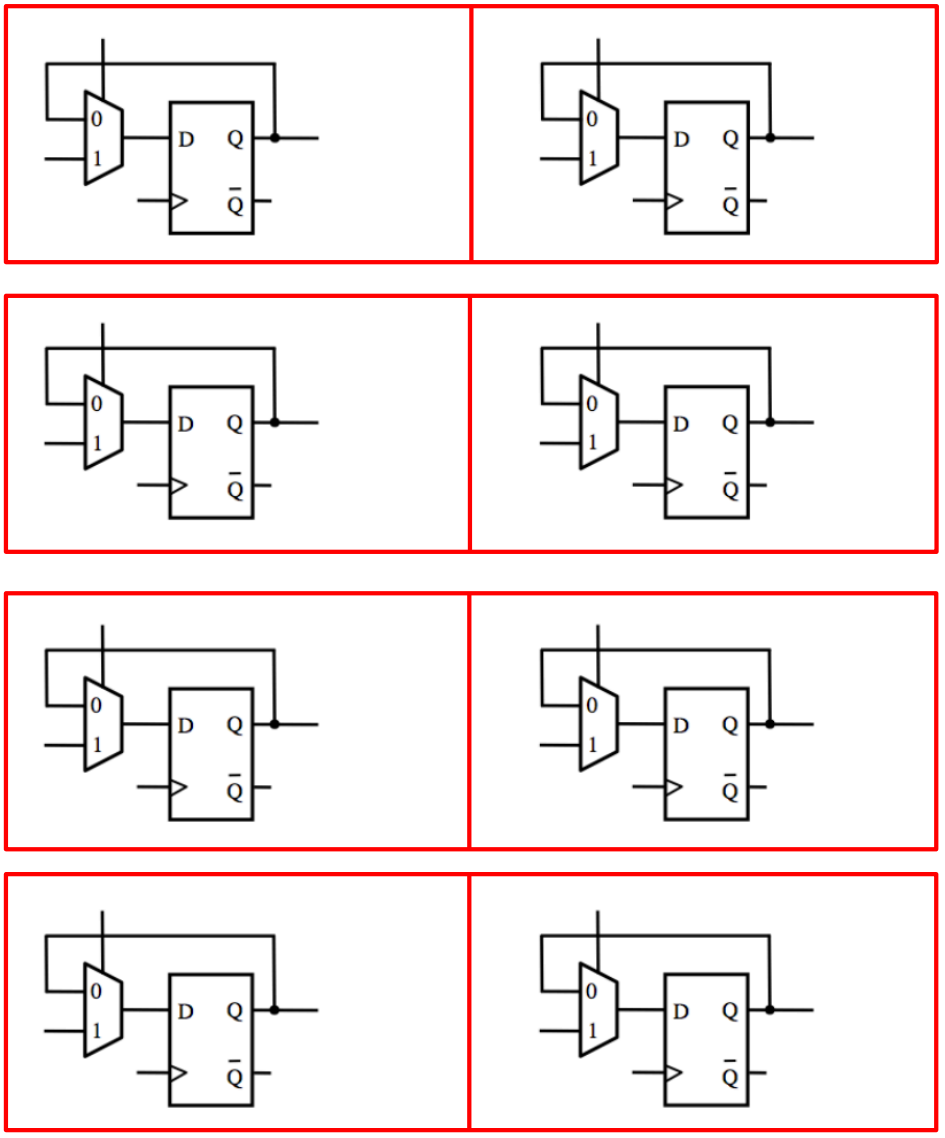


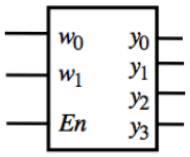

0
1
D
Q
Q

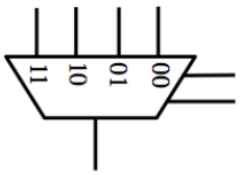

w_0
w_1
En
y_0
y_1
y_2
y_3

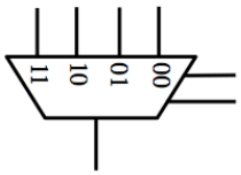

11
10
01
00

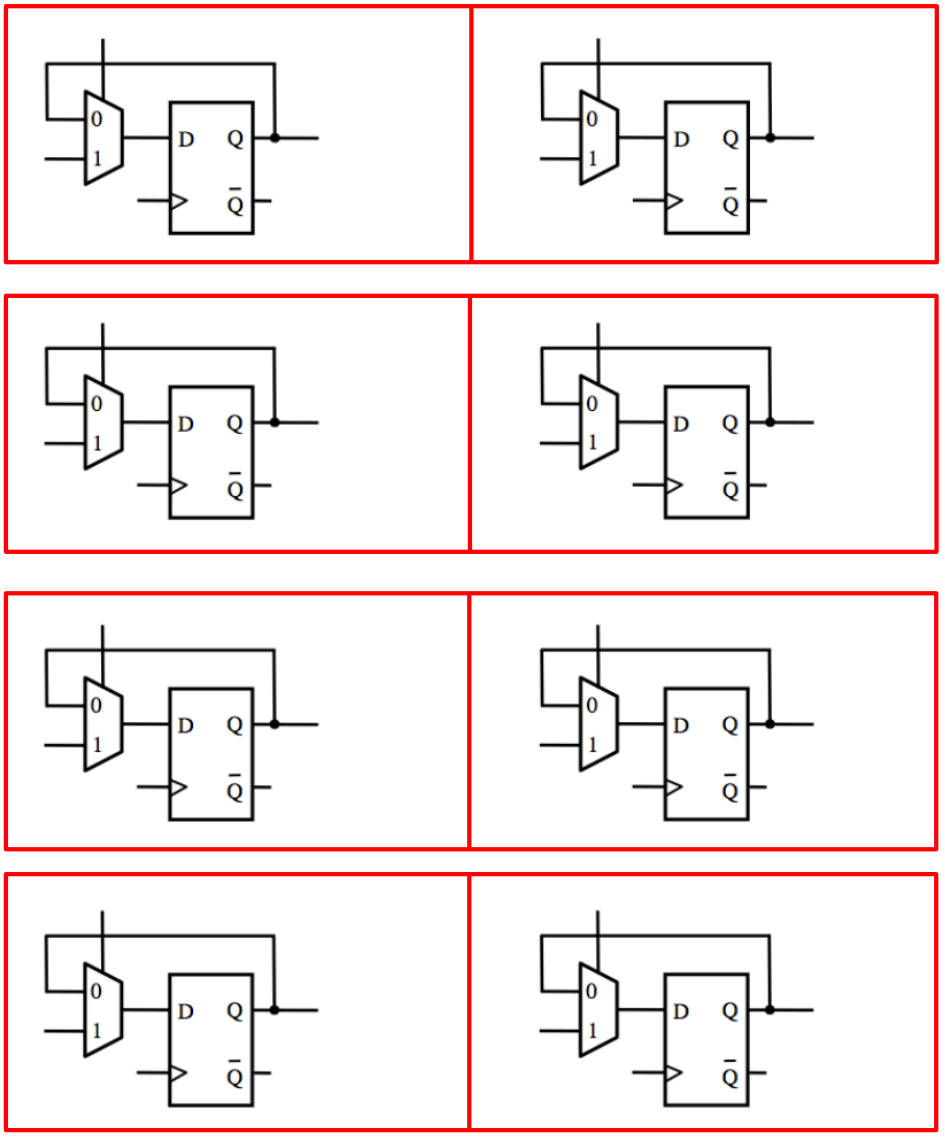

11
10
01
00

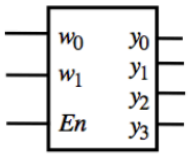

Register 0
Register 1
Register 2
Register 3
D
Q
0
1

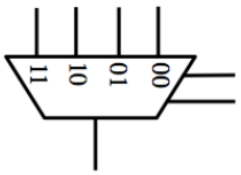

w_0
w_1
En
y_0
y_1
y_2
y_3

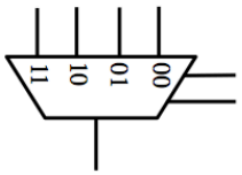

11
10
01
00

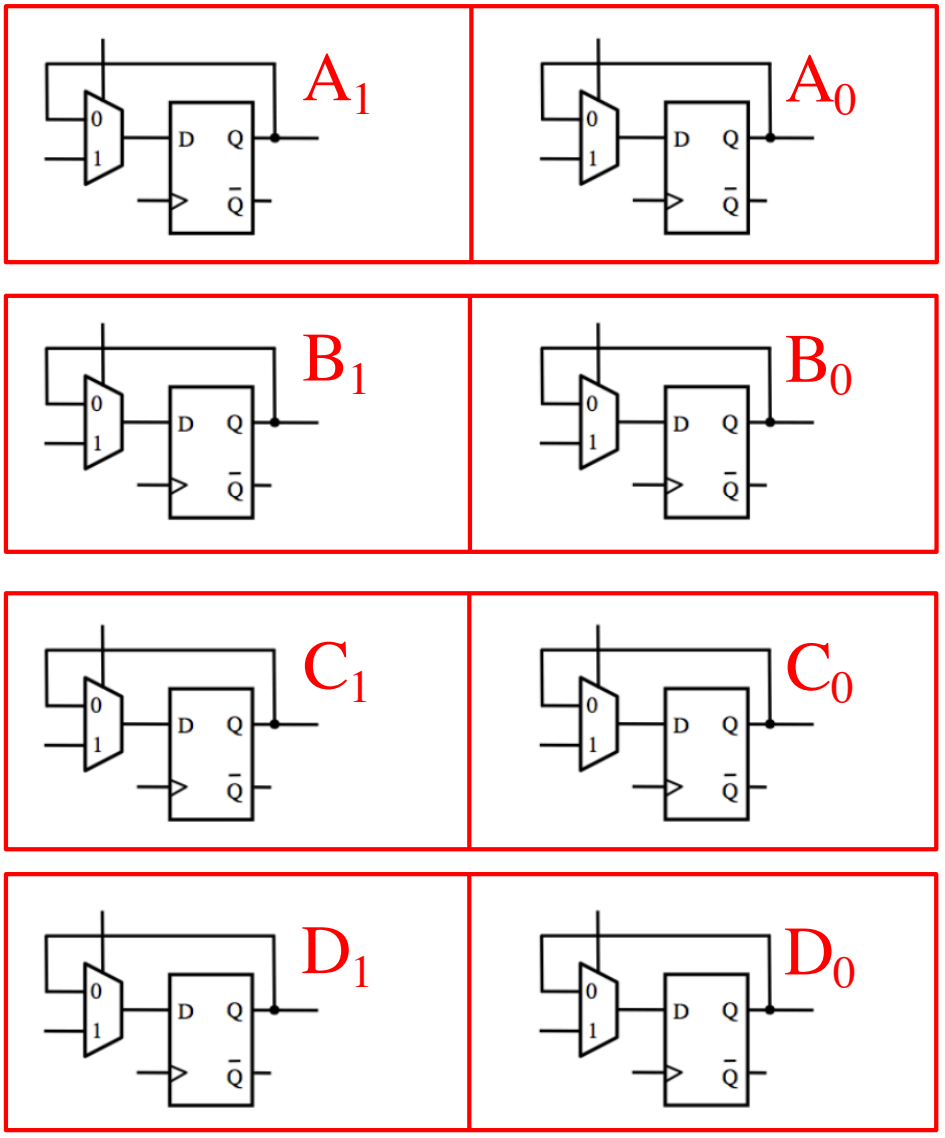

11
10
01
00

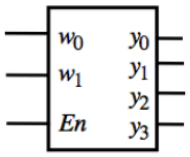

Register A
Register B
Register C
Register D
0
1
D
Q
Q̄

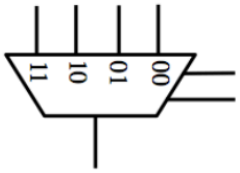

w_0
w_1
En
y_0
y_1
y_2
y_3

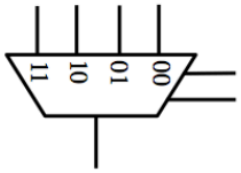

11
10
01
00

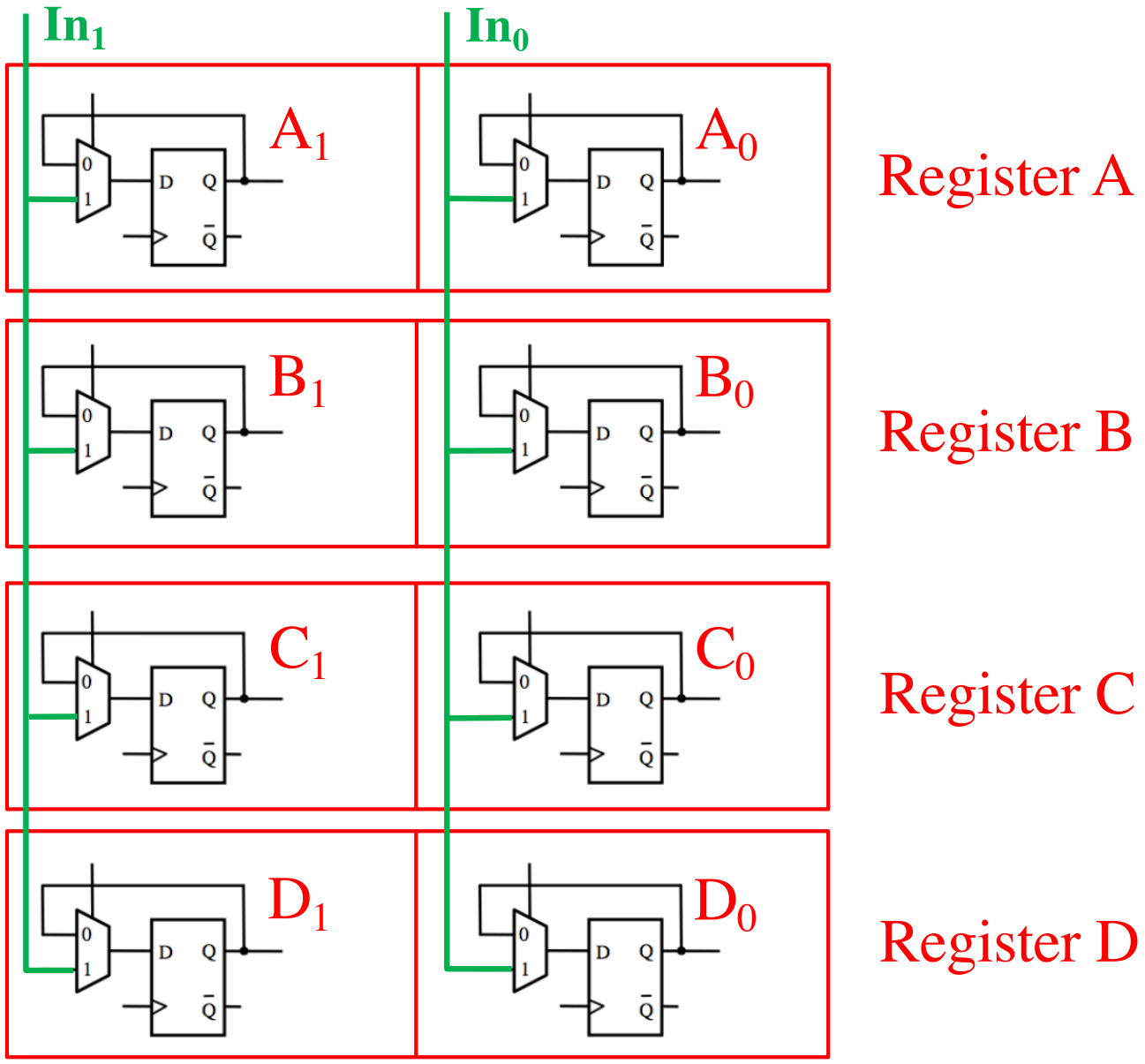

11
10
01
00

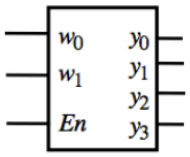

A1
A0
Register A
B1
B0
Register B
C1
C0
Register C
D1
D0
Register D
D
Q
0
1

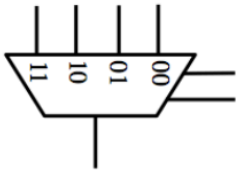

w0
w1
En
y0
y1
y2
y3

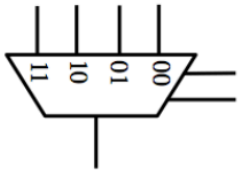

11
10
01
00

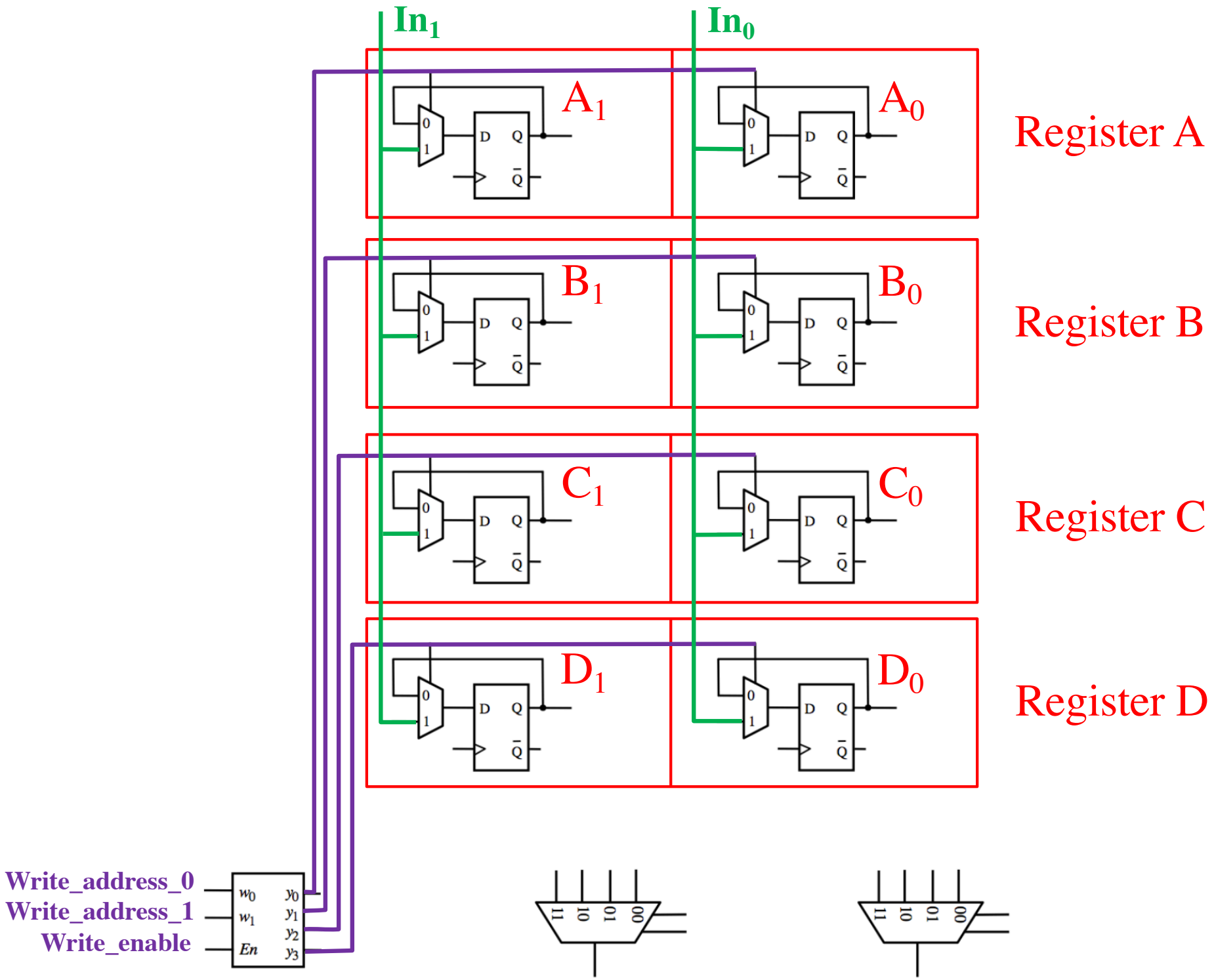

11
10
01
00

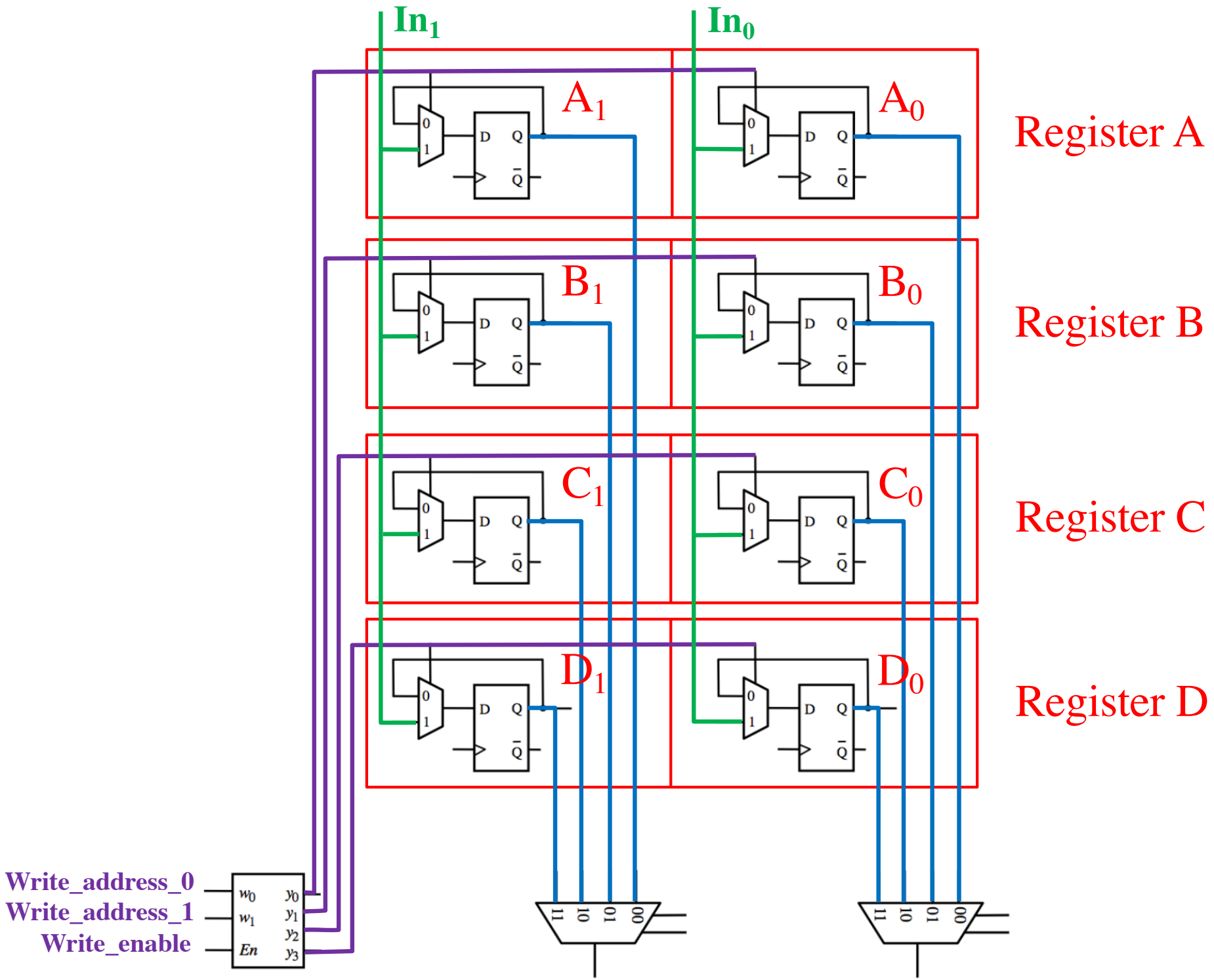

In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
D
Q
Q̄
0
1
Register A
Register B
Register C
Register D

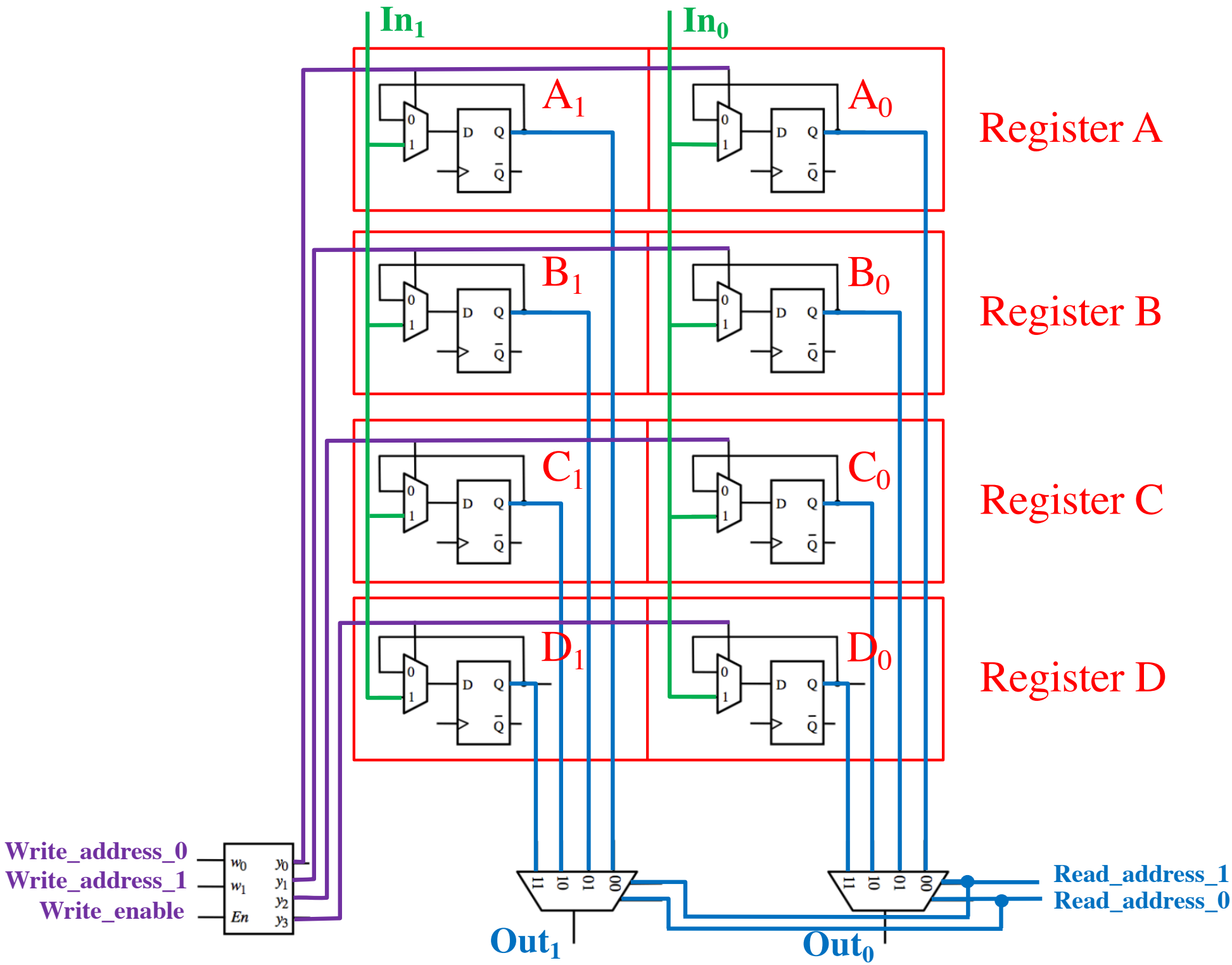

w0
w1
En
y0
y1
y2
y3

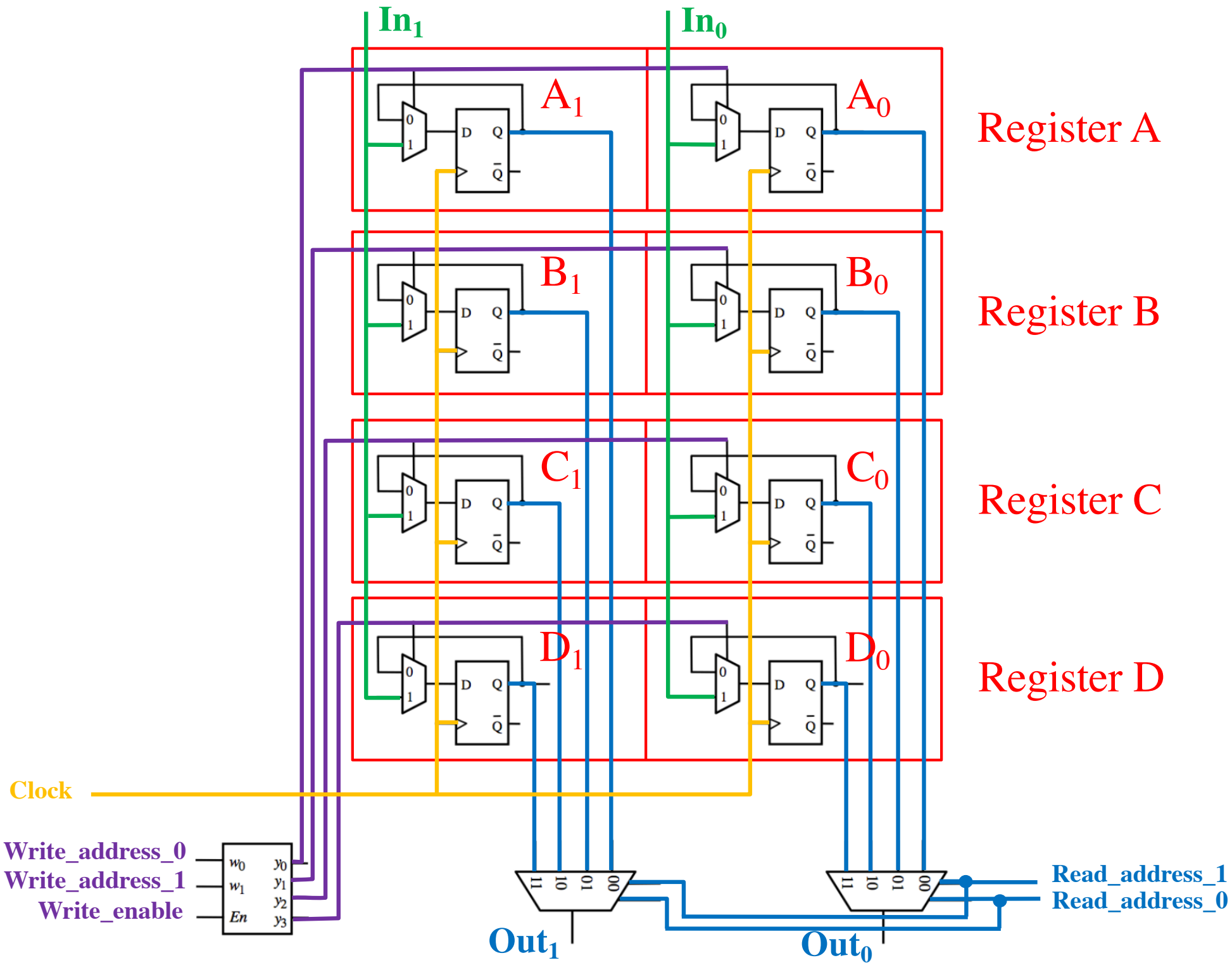

11
10
01
00

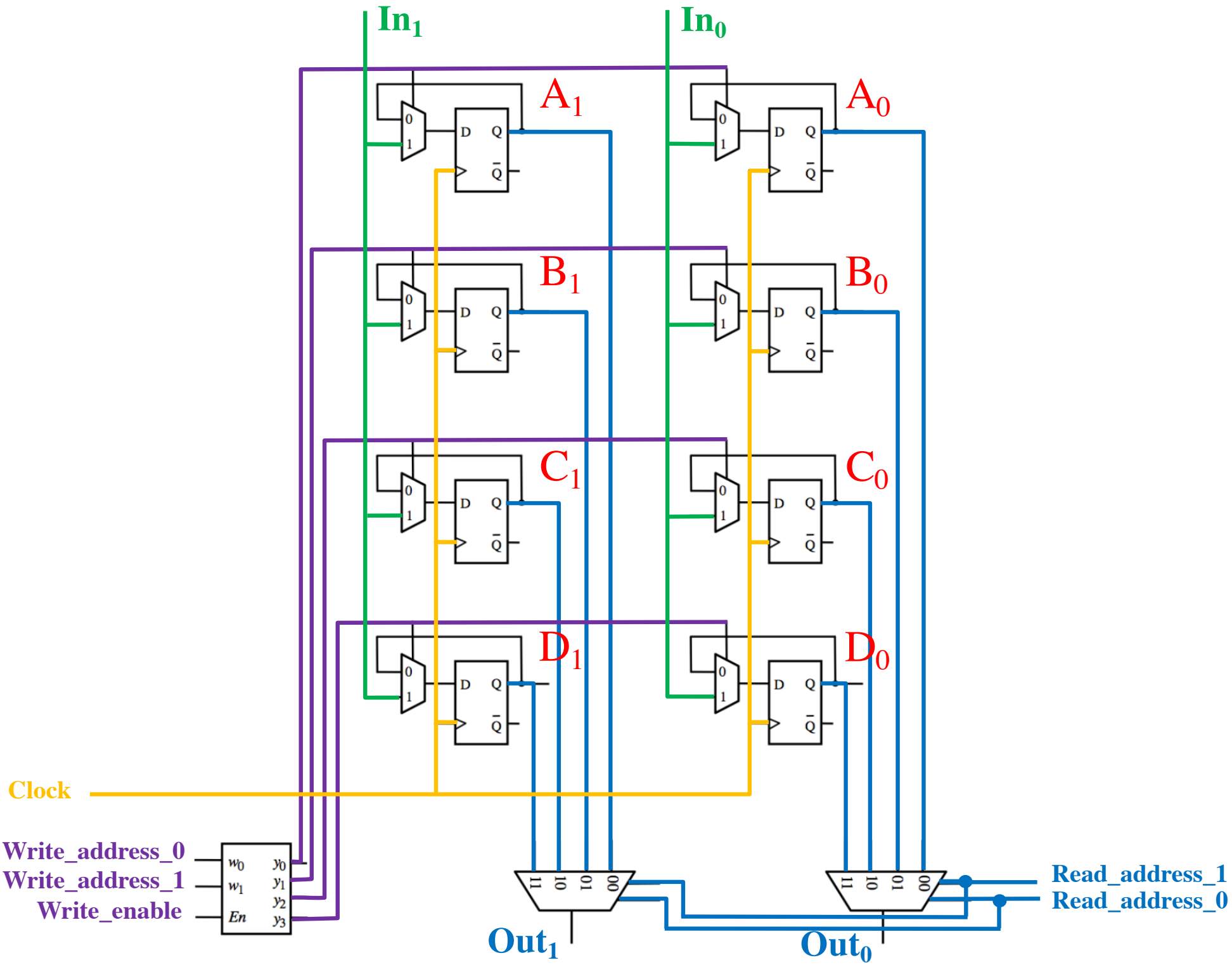

11
10
01
00


In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
Register A
Register B
Register C
Register D
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
11
10
01
00


In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
Register A
Register B
Register C
Register D
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00

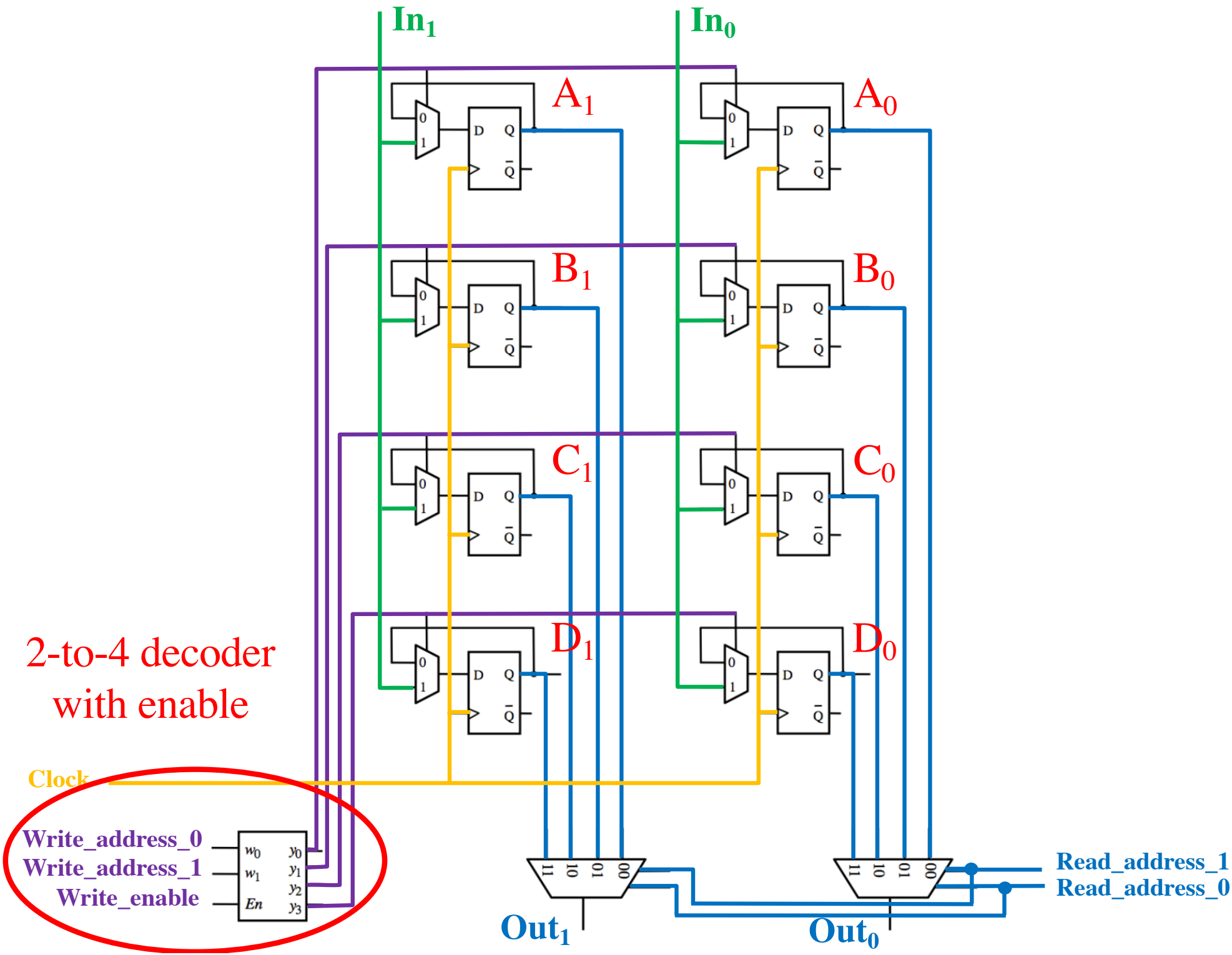

In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
Register A
Register B
Register C
Register D
D
Q
Q̄
0
1
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
Read_address_1
Read_address_0


In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
Register A
Register B
Register C
Register D
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
Read_address_1
Read_address_0

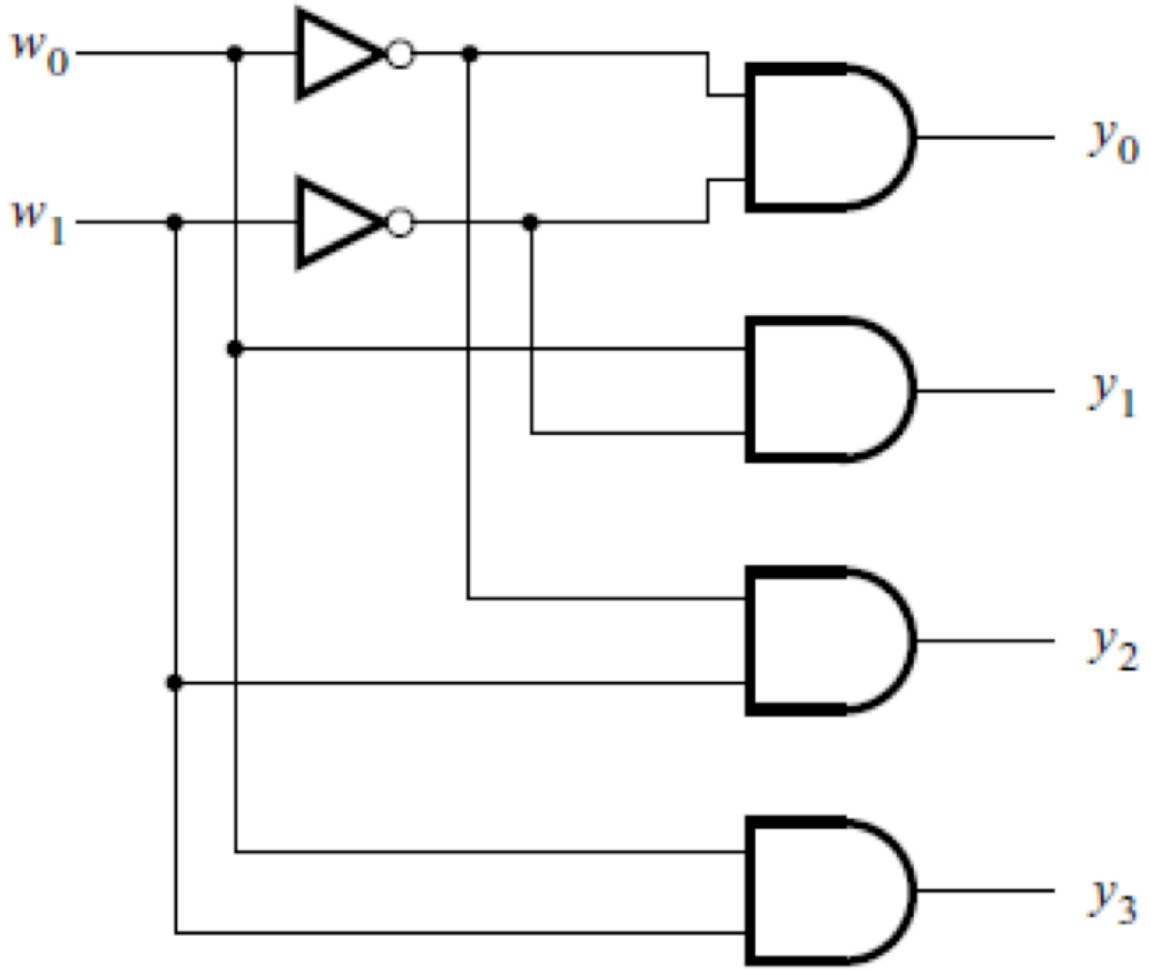

In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
D
Q
Q̄
0
1
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
Read_address_1
Read_address_0

Components of the Register File

2-to-4 Decoder

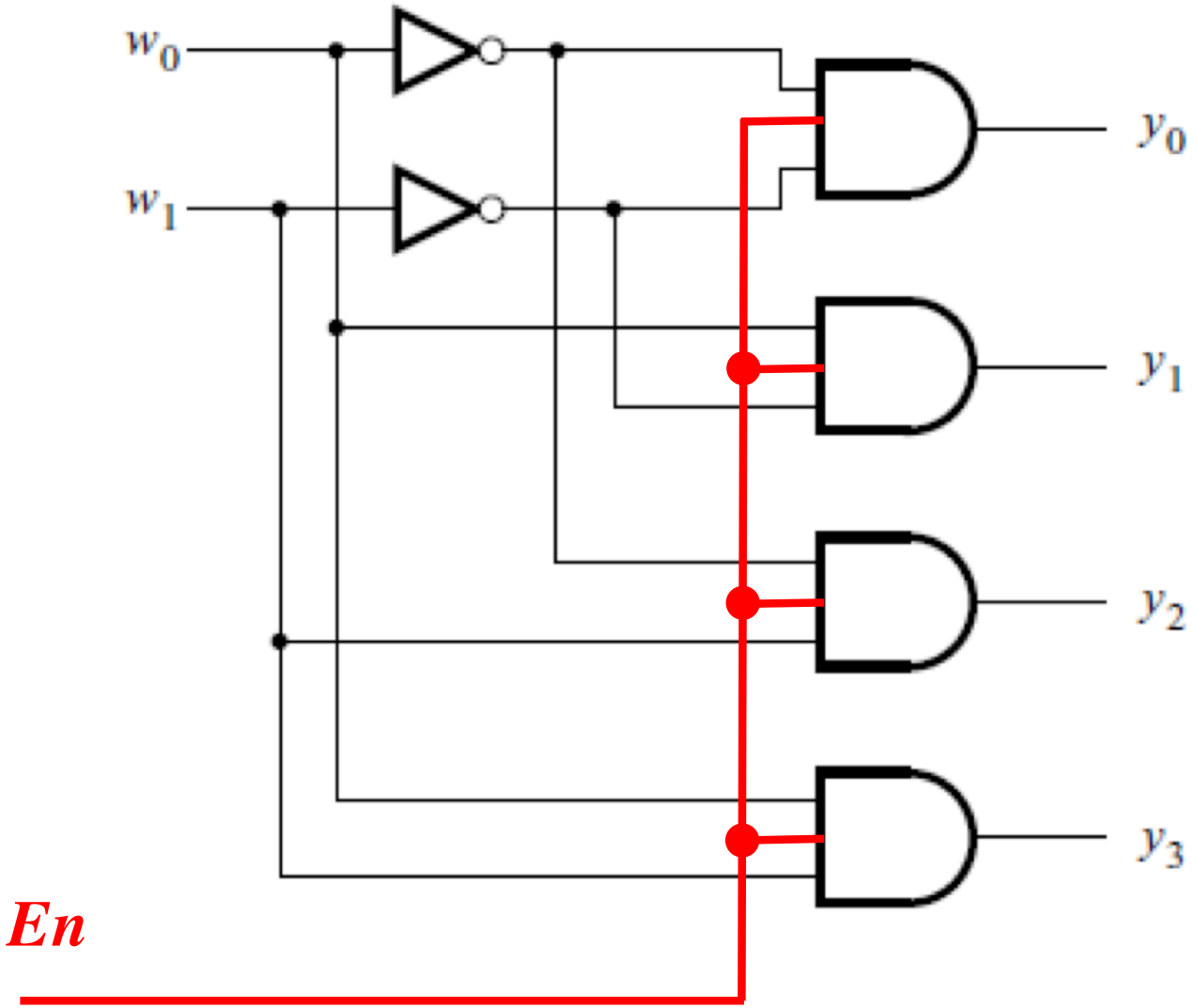

In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
2-to-4 decoder
with enable
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
Out1
Out0
Read_address_1
Read_address_0

2-to-4 Decoder

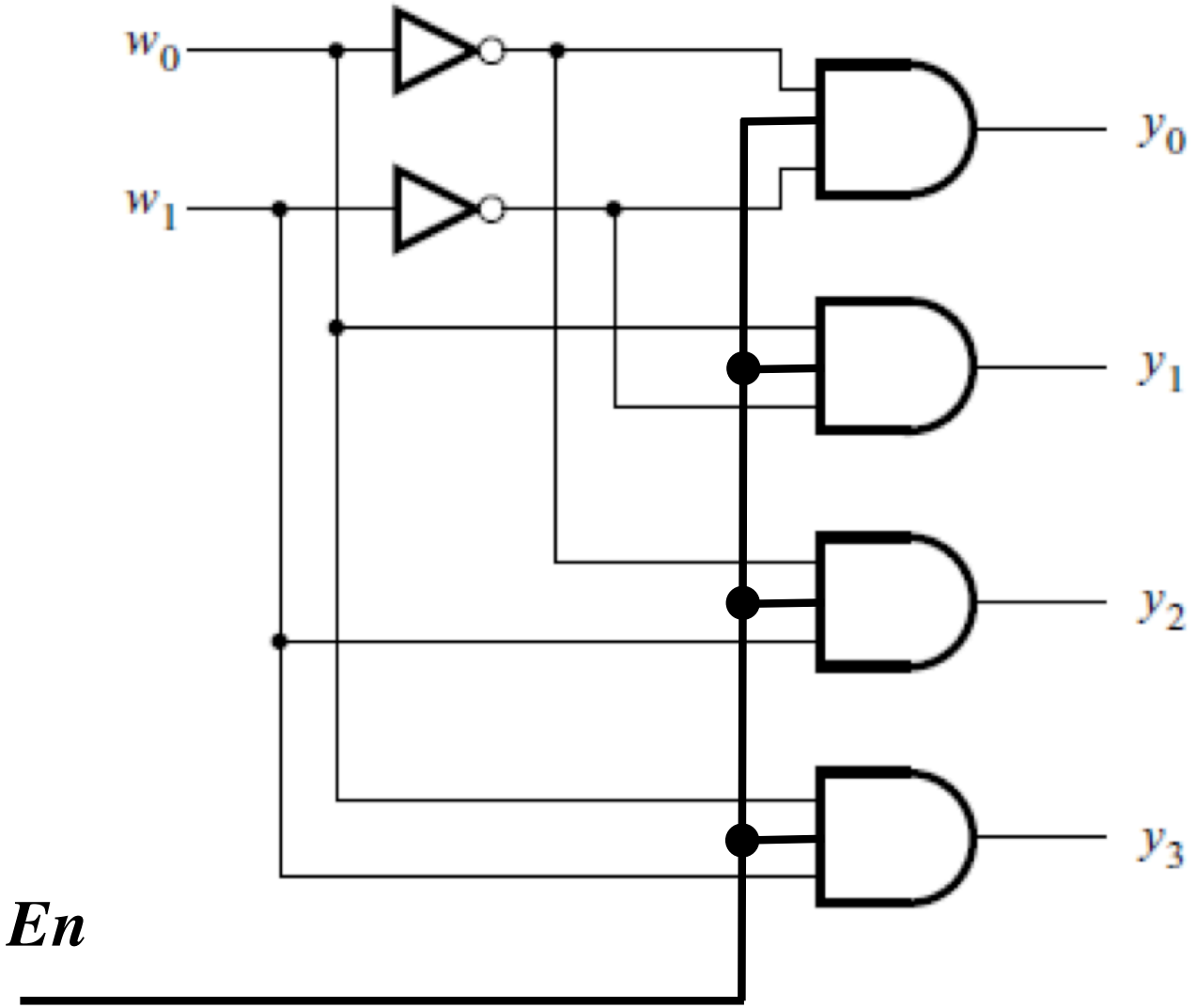


[Figure 4.14c from the textbook]

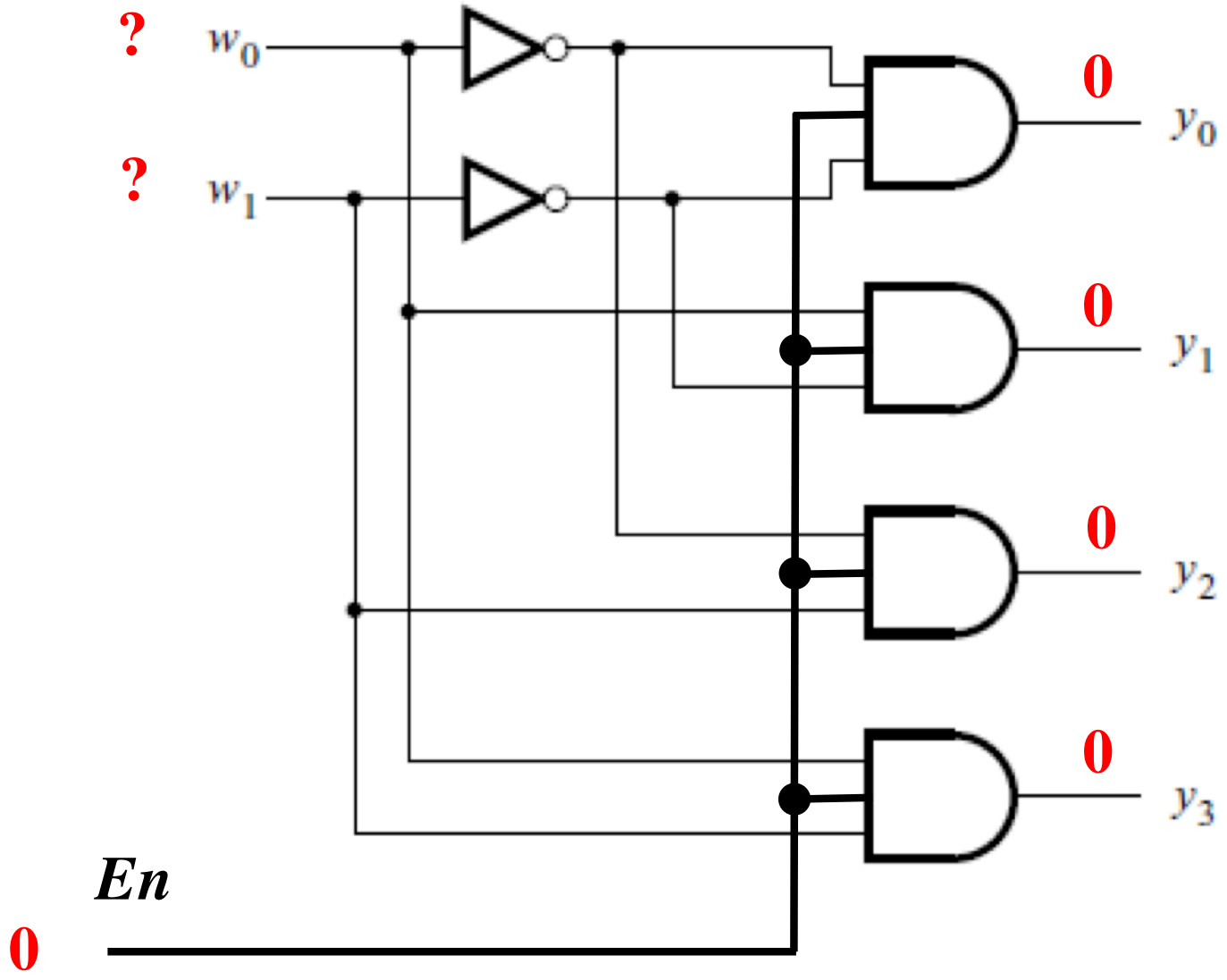
2-to-4 Decoder with Enable Input



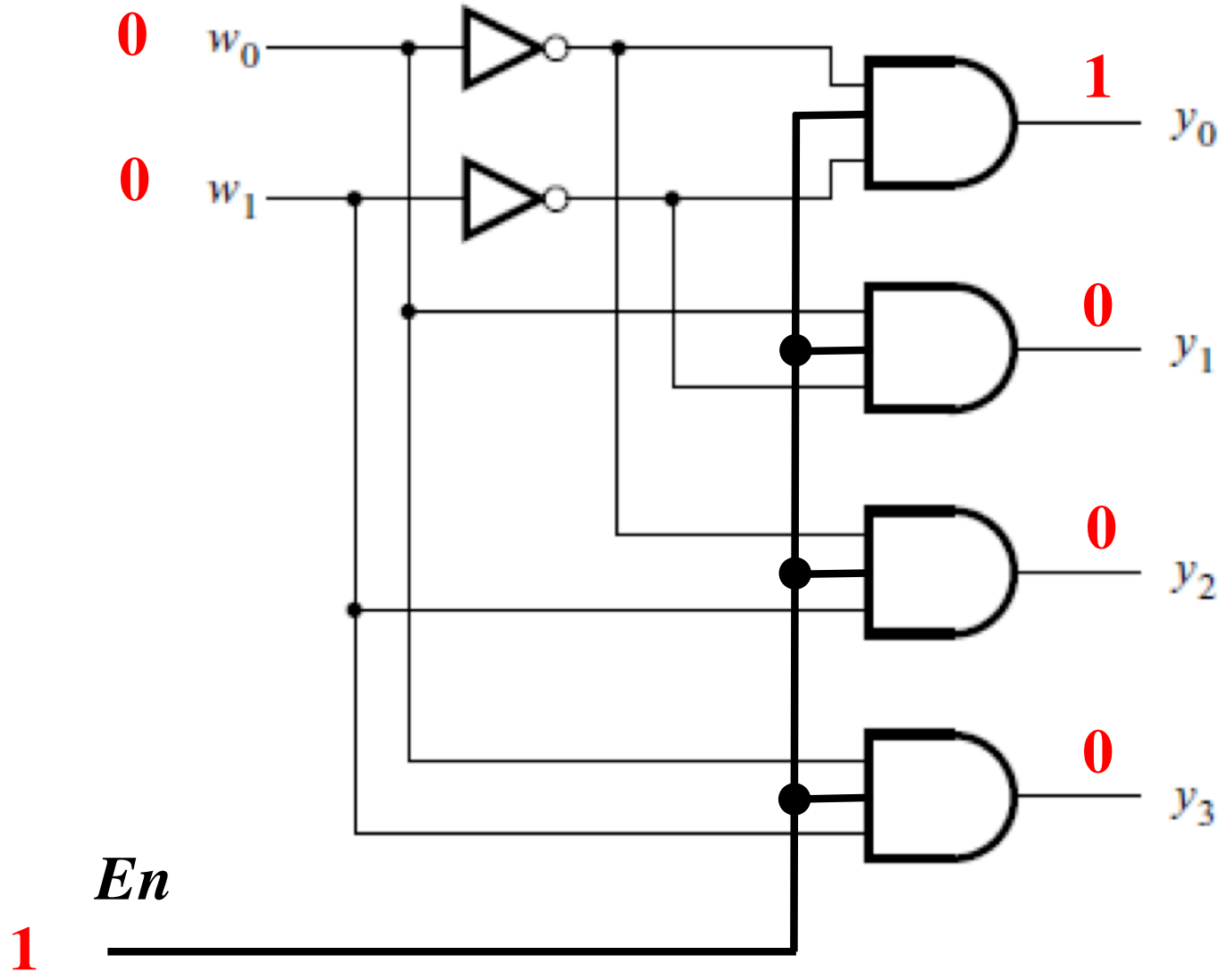
2-to-4 Decoder with Enable Input



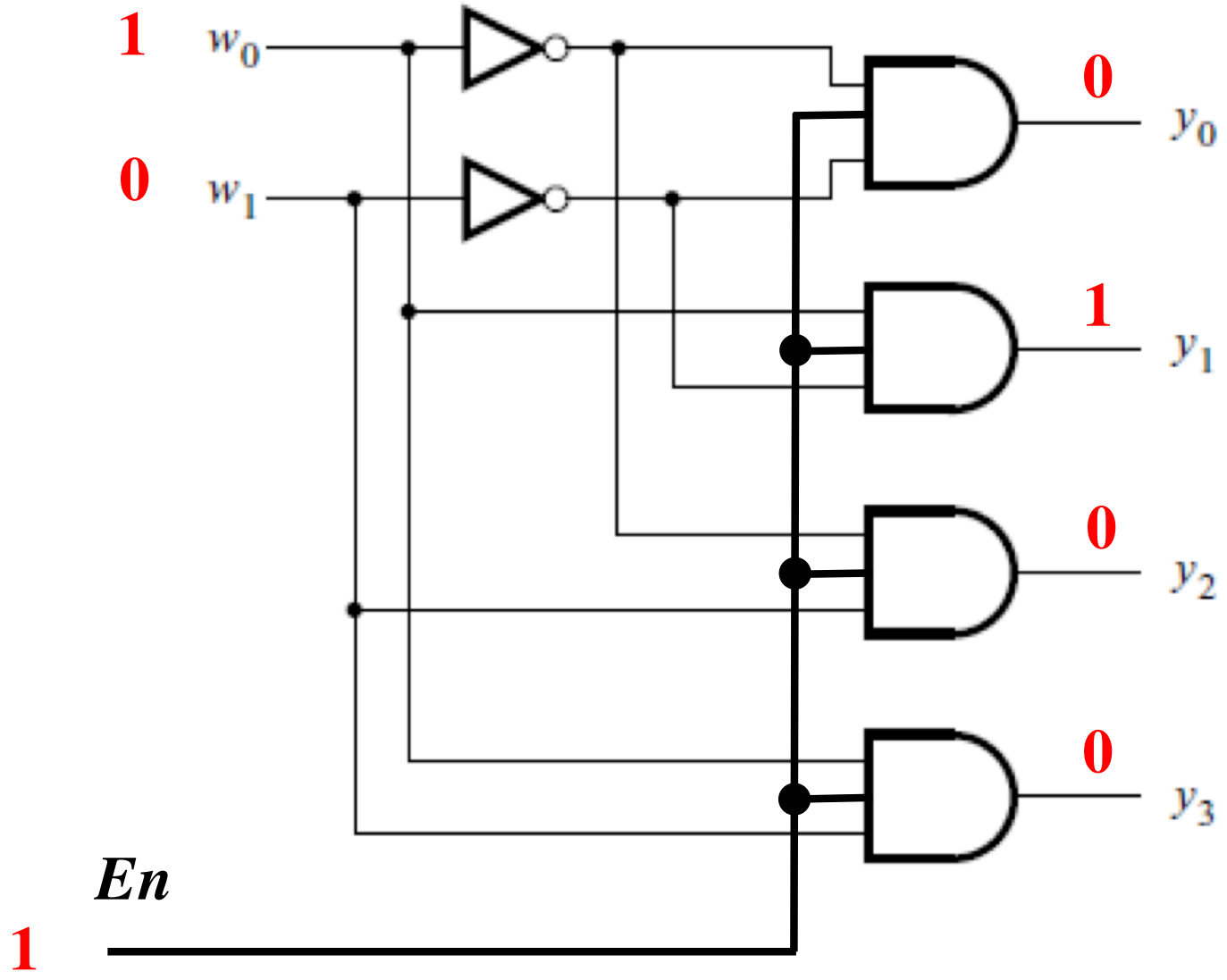
2-to-4 Decoder with Enable Input



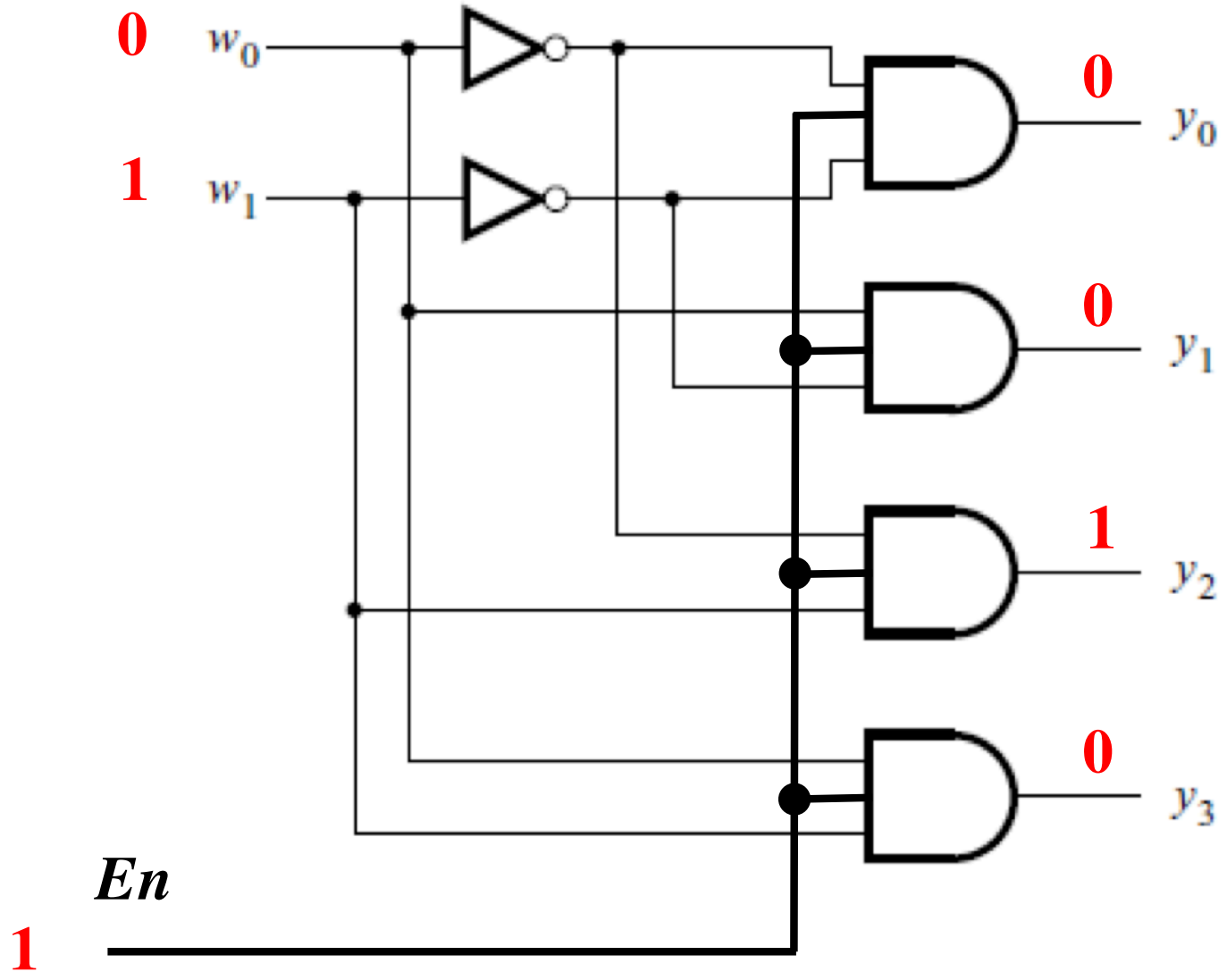
2-to-4 Decoder with Enable Input



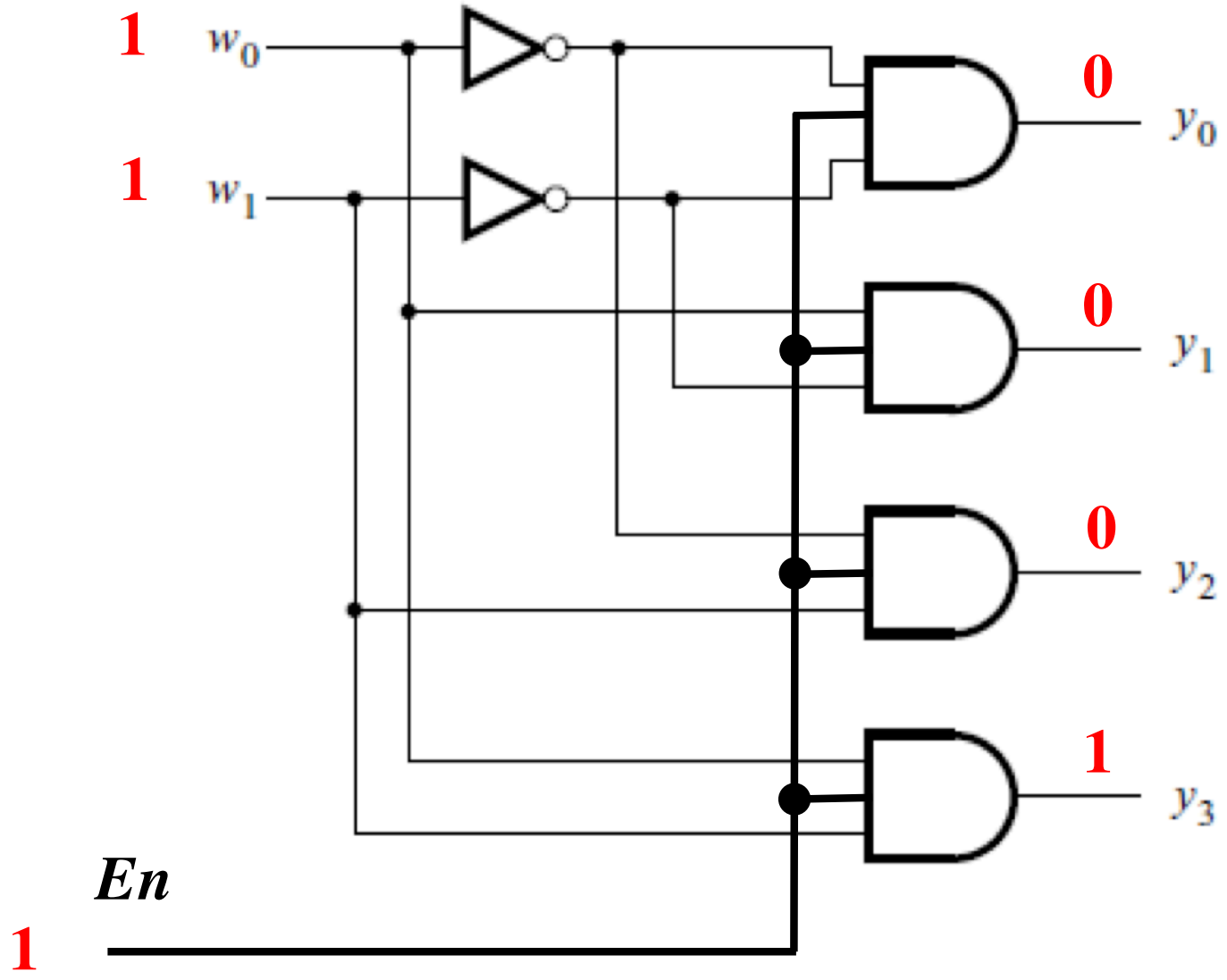
2-to-4 Decoder with Enable Input



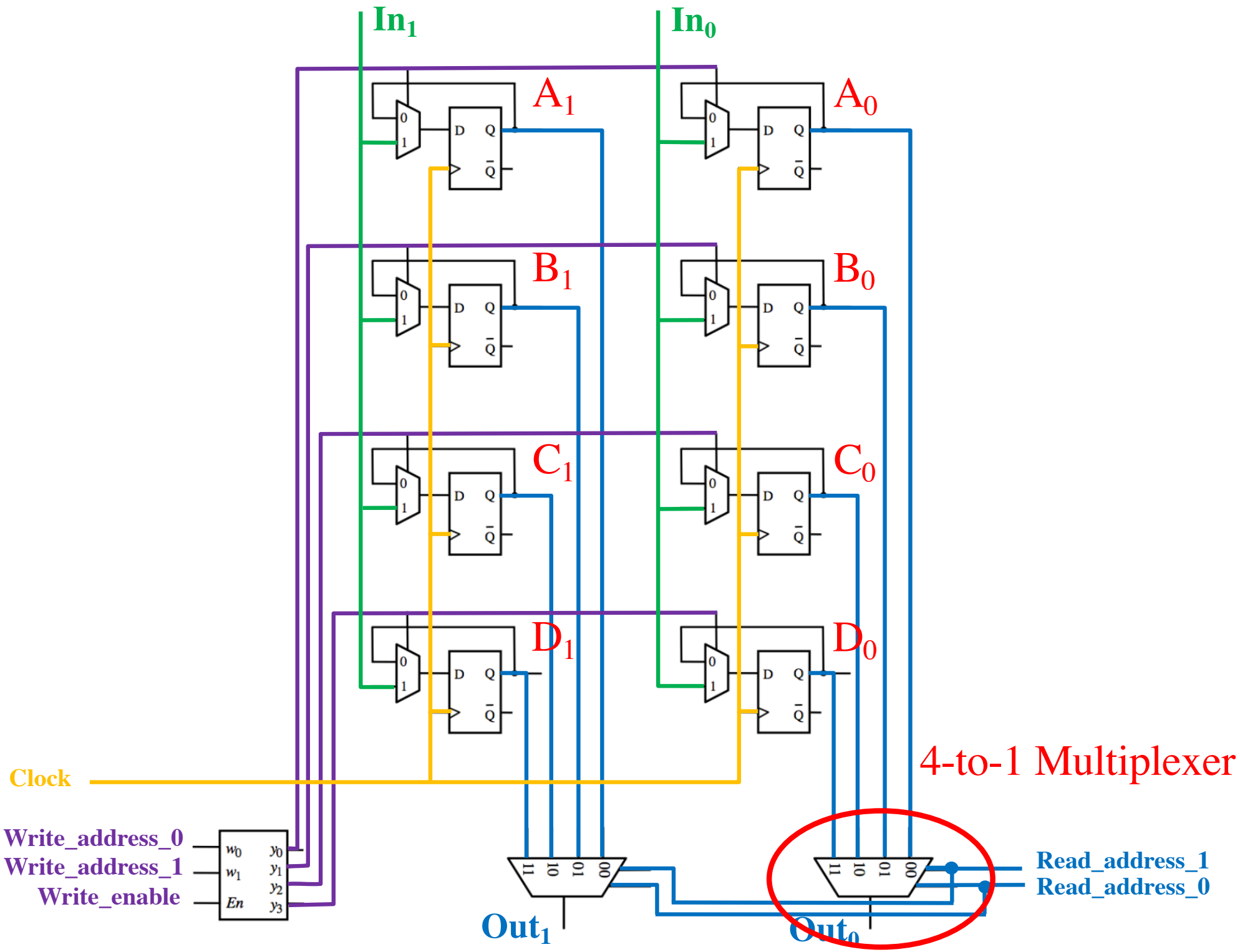
2-to-4 Decoder with Enable Input

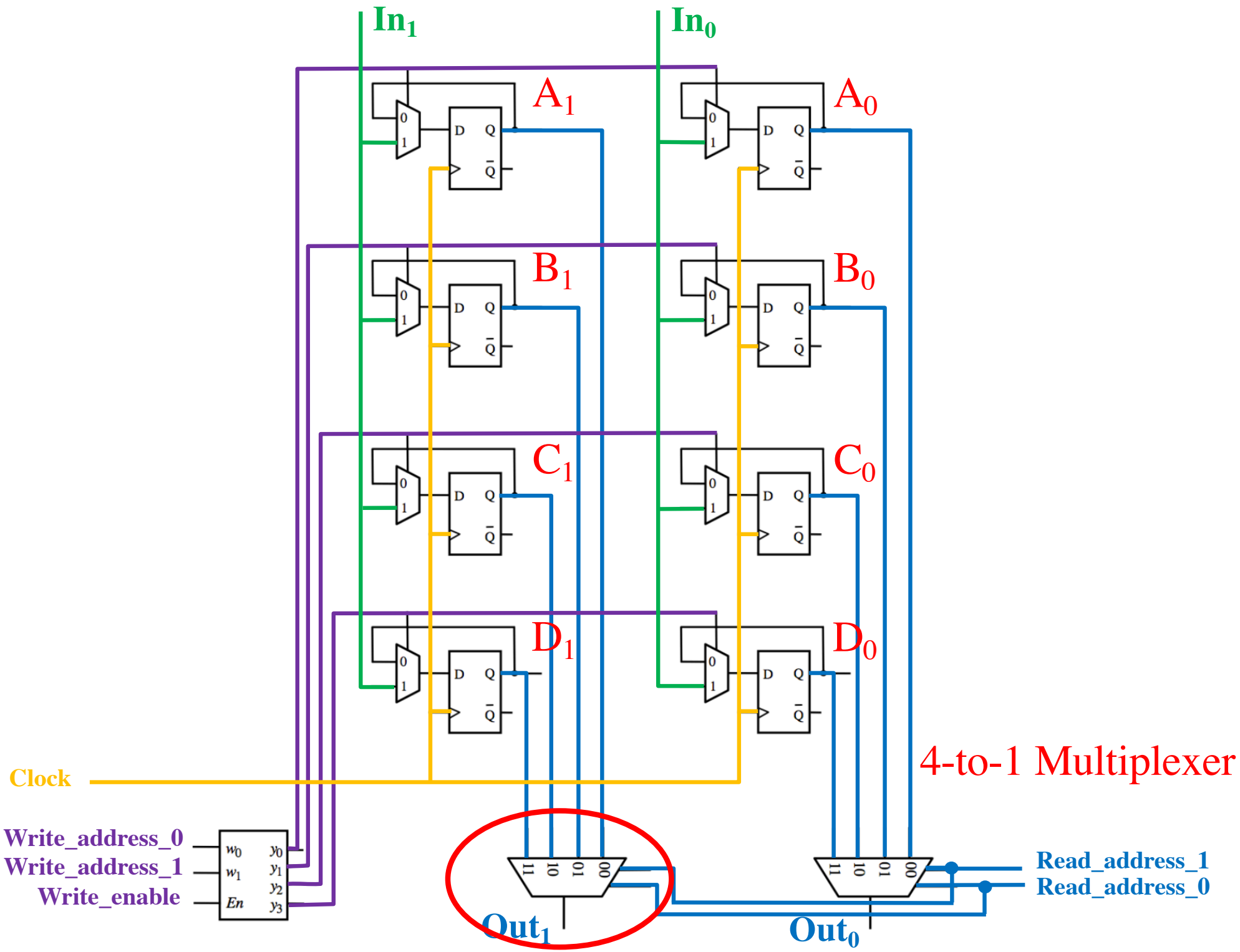


2-to-4 Decoder with Enable Input

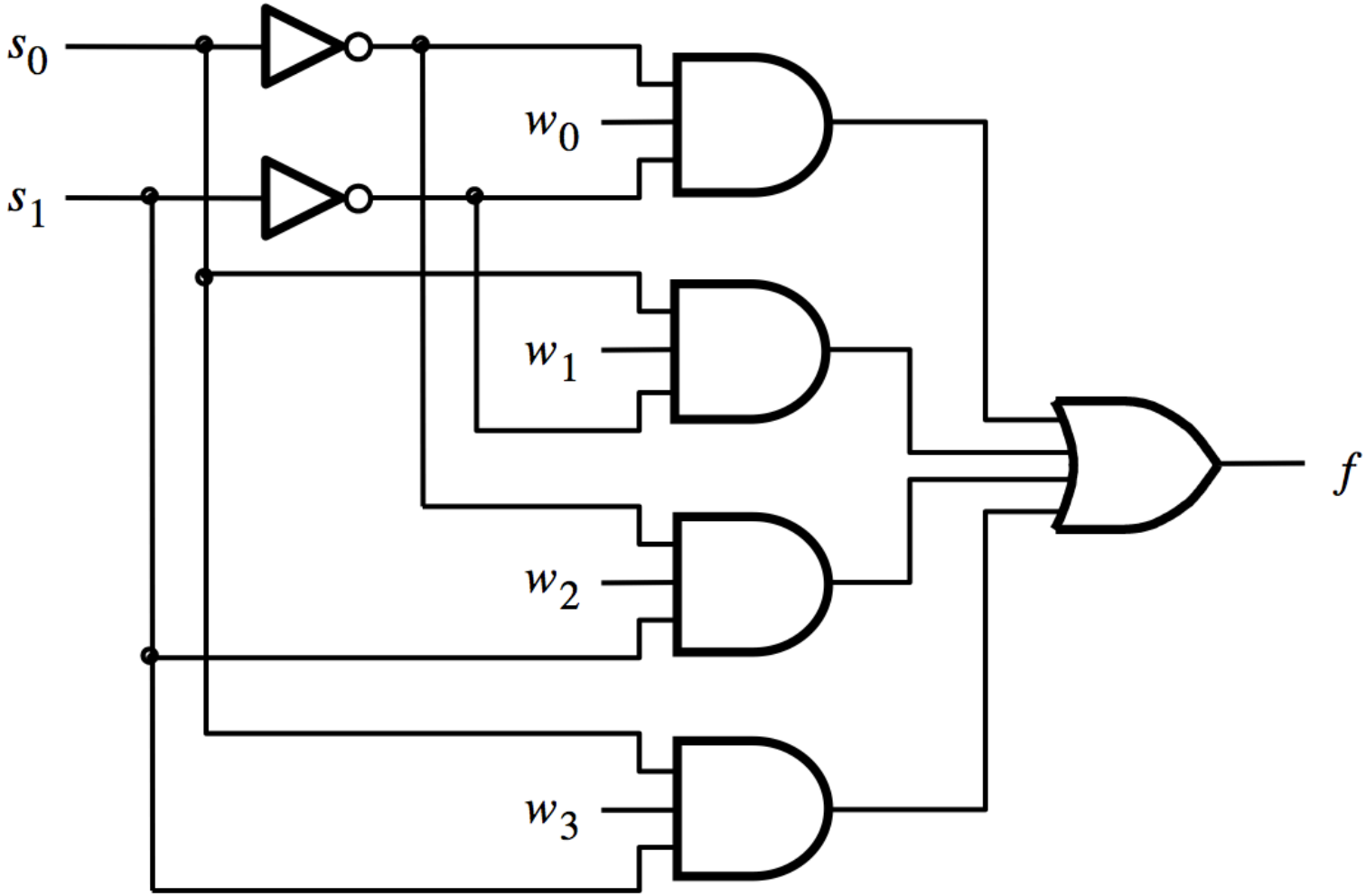


4-to-1 Multiplexer


In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
D
Q
Q̄
0
1
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
4-to-1 Multiplexer
Read_address_1
Read_address_0


In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
D
Q
Q̄
0
1
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
4-to-1 Multiplexer
Read_address_1
Read_address_0

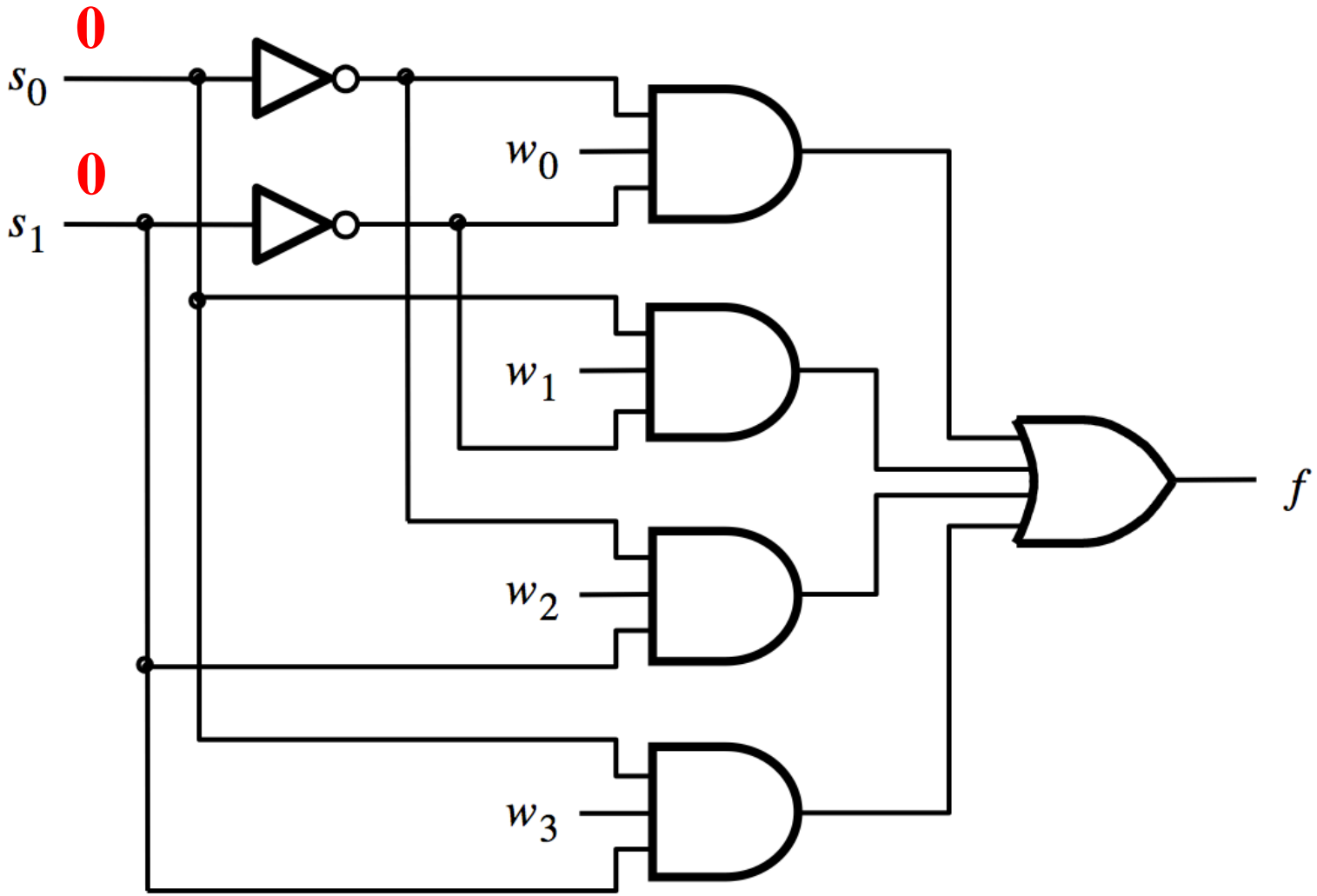
4-to-1 Multiplexer



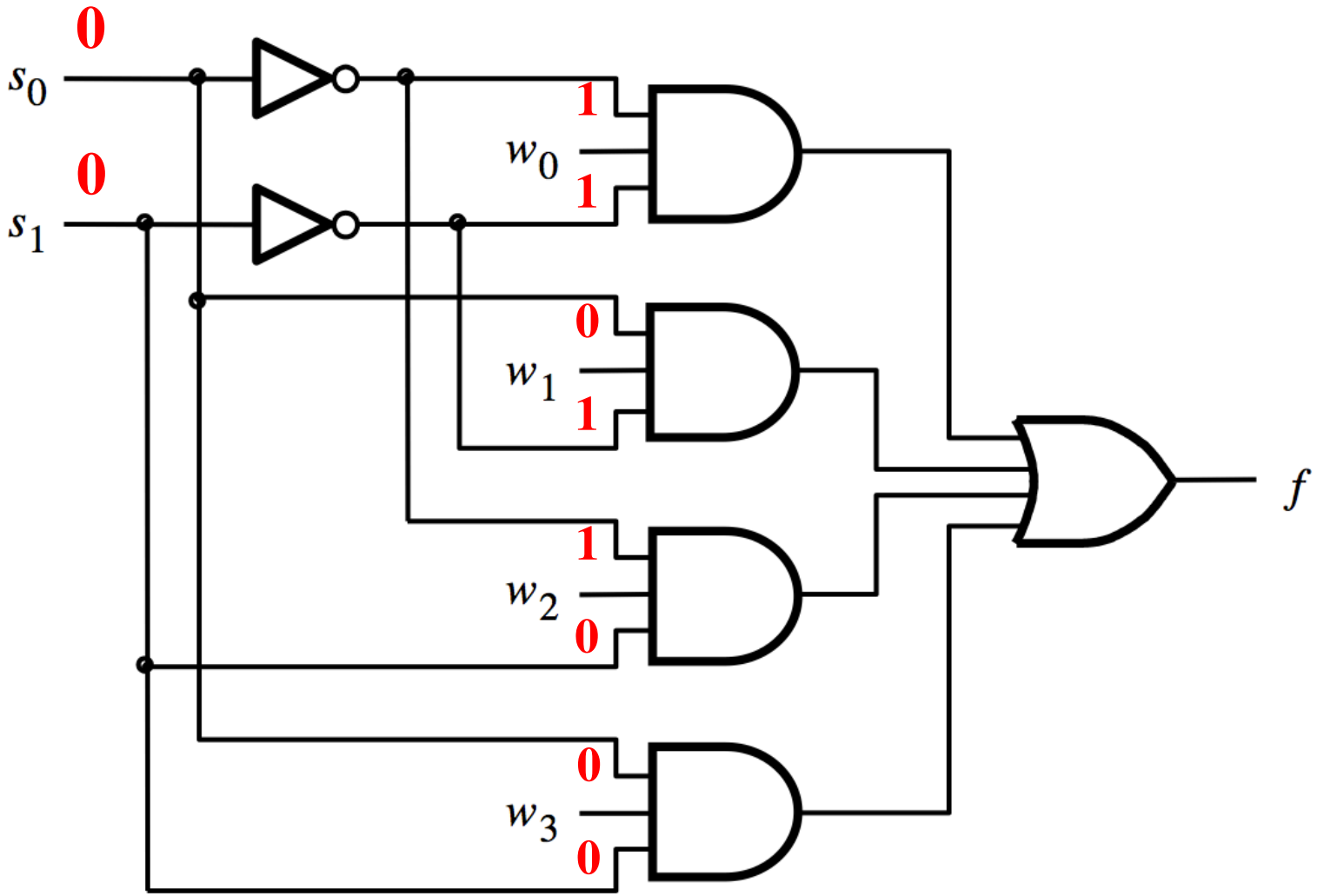
$$f = \overline{s_1}\,\overline{s_0}\,w_0 + \overline{s_1}\,s_0\,w_1 + s_1\,\overline{s_0}\,w_2 + s_1\,s_0\,w_3$$

[Figure 4.2c from the textbook]

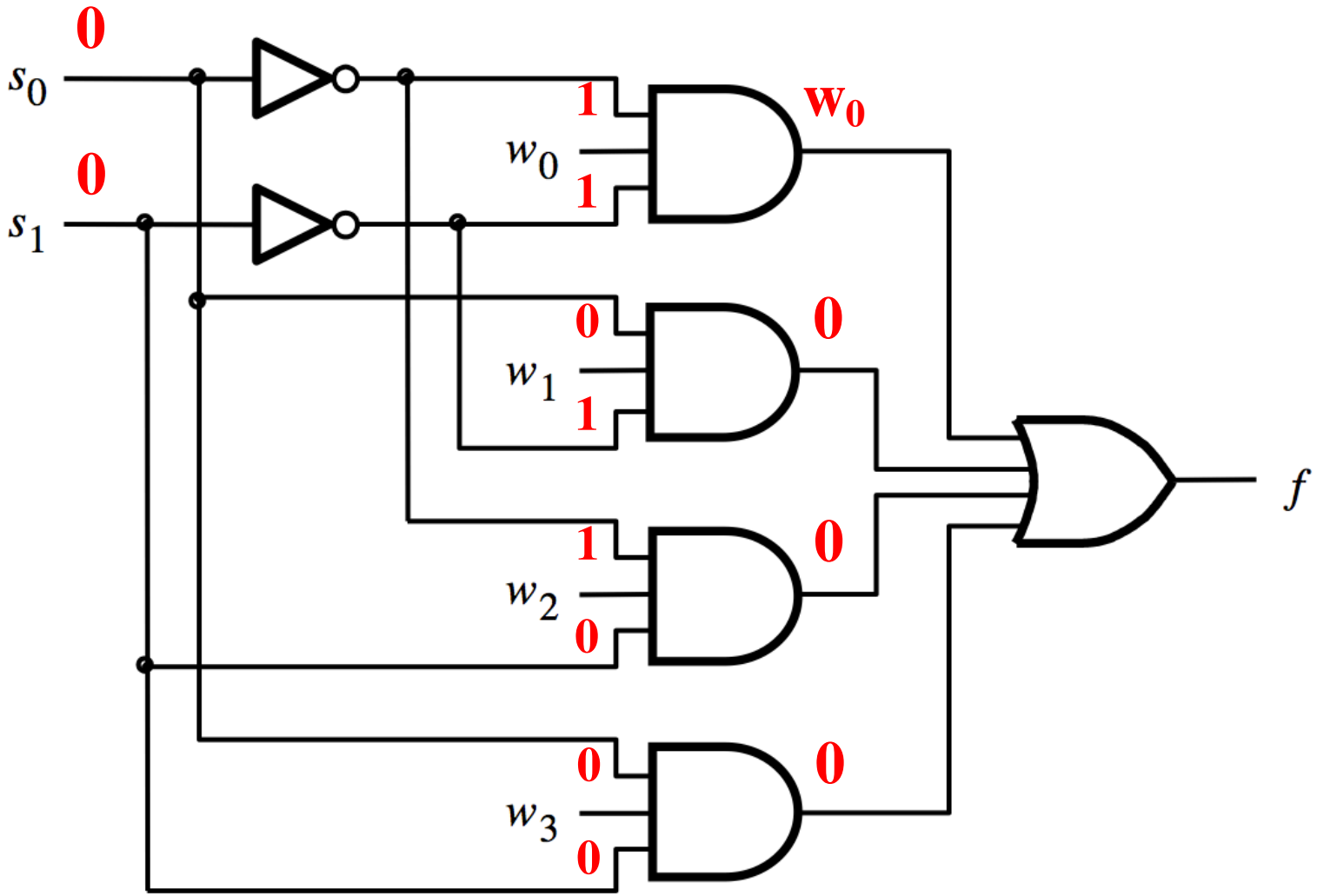
Analysis of the 4-to-1 Multiplexer (s_1=0 and s_0=0)



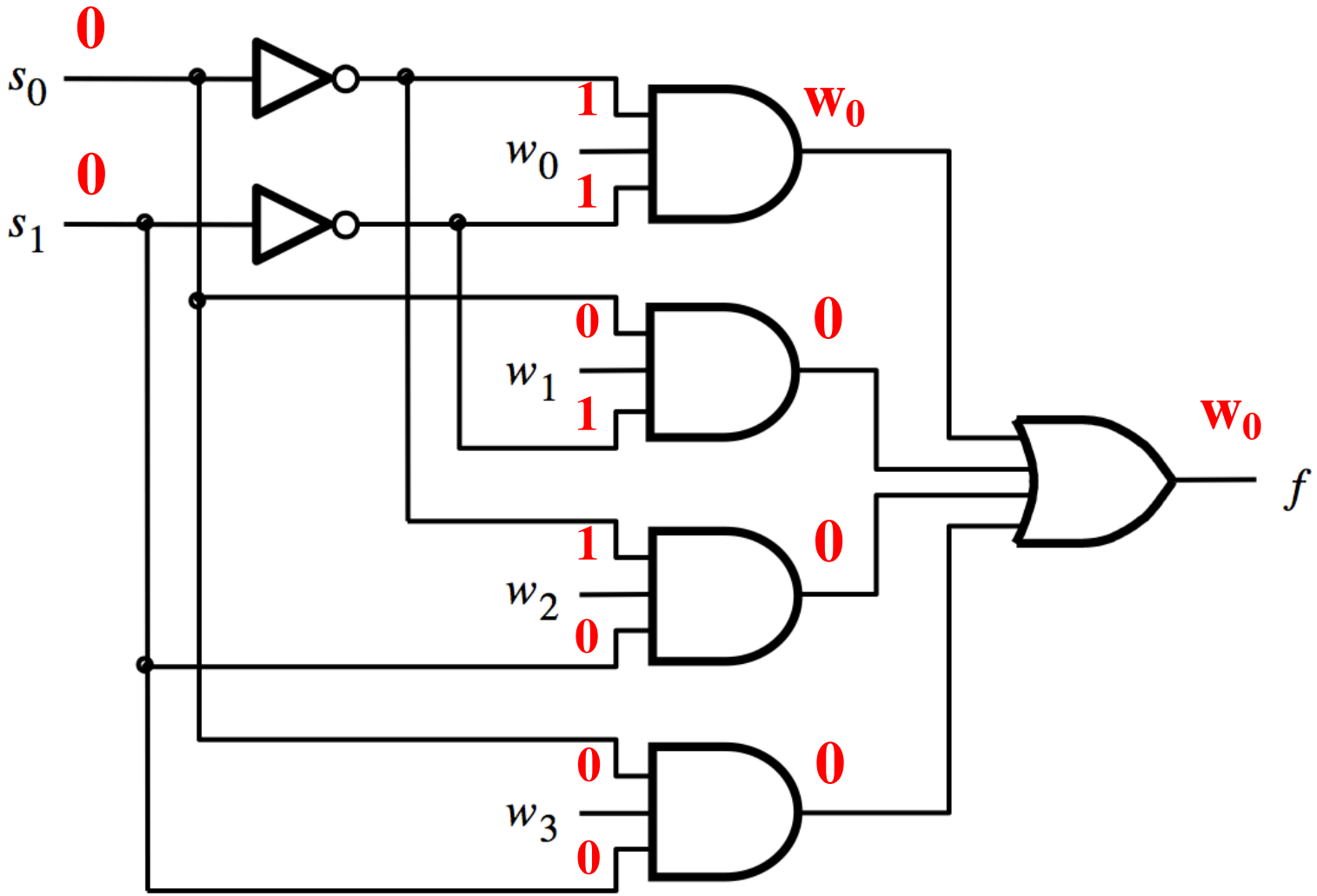
Analysis of the 4-to-1 Multiplexer (s_1=0 and s_0=0)



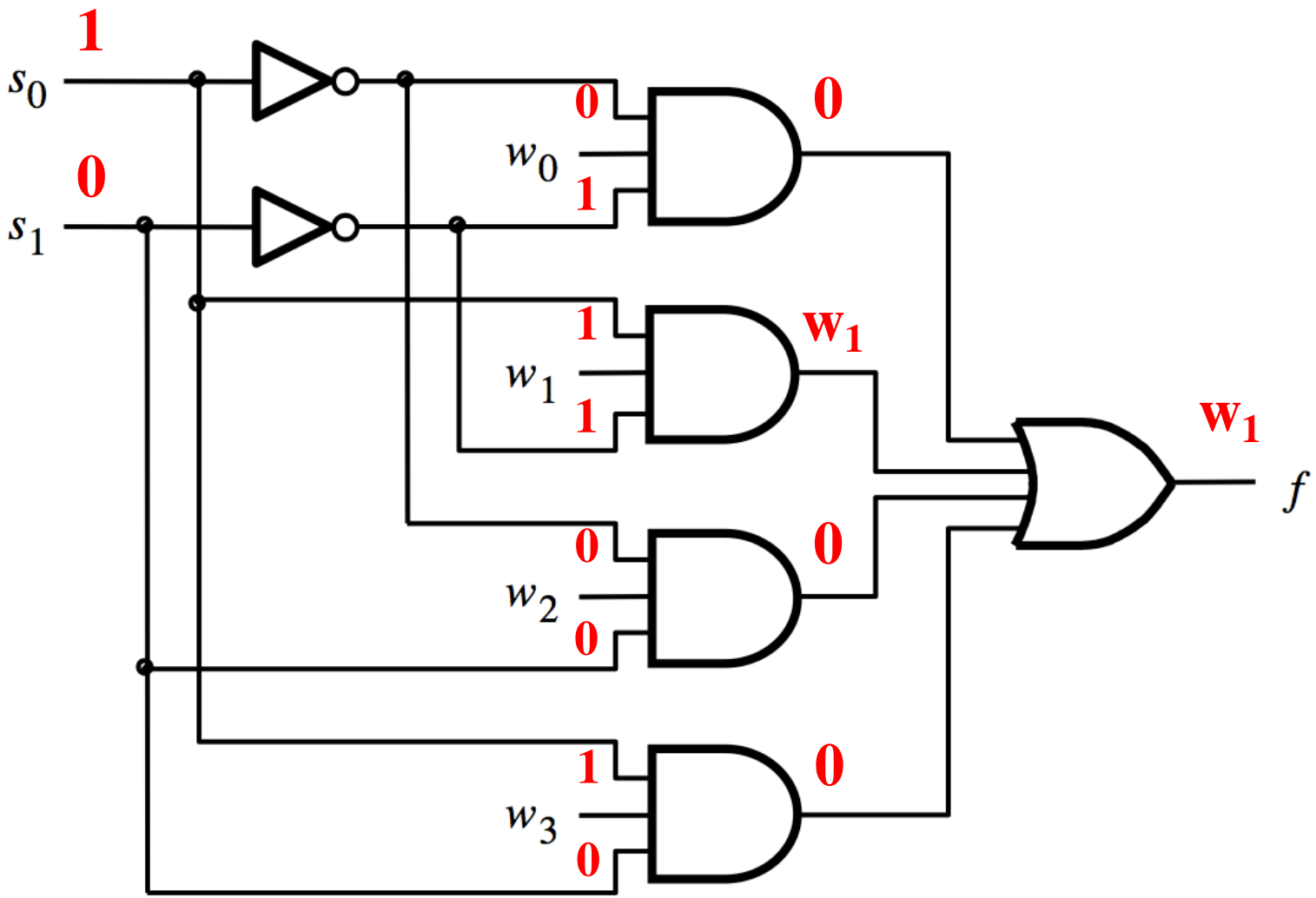
Analysis of the 4-to-1 Multiplexer (s_1=0 and s_0=0)



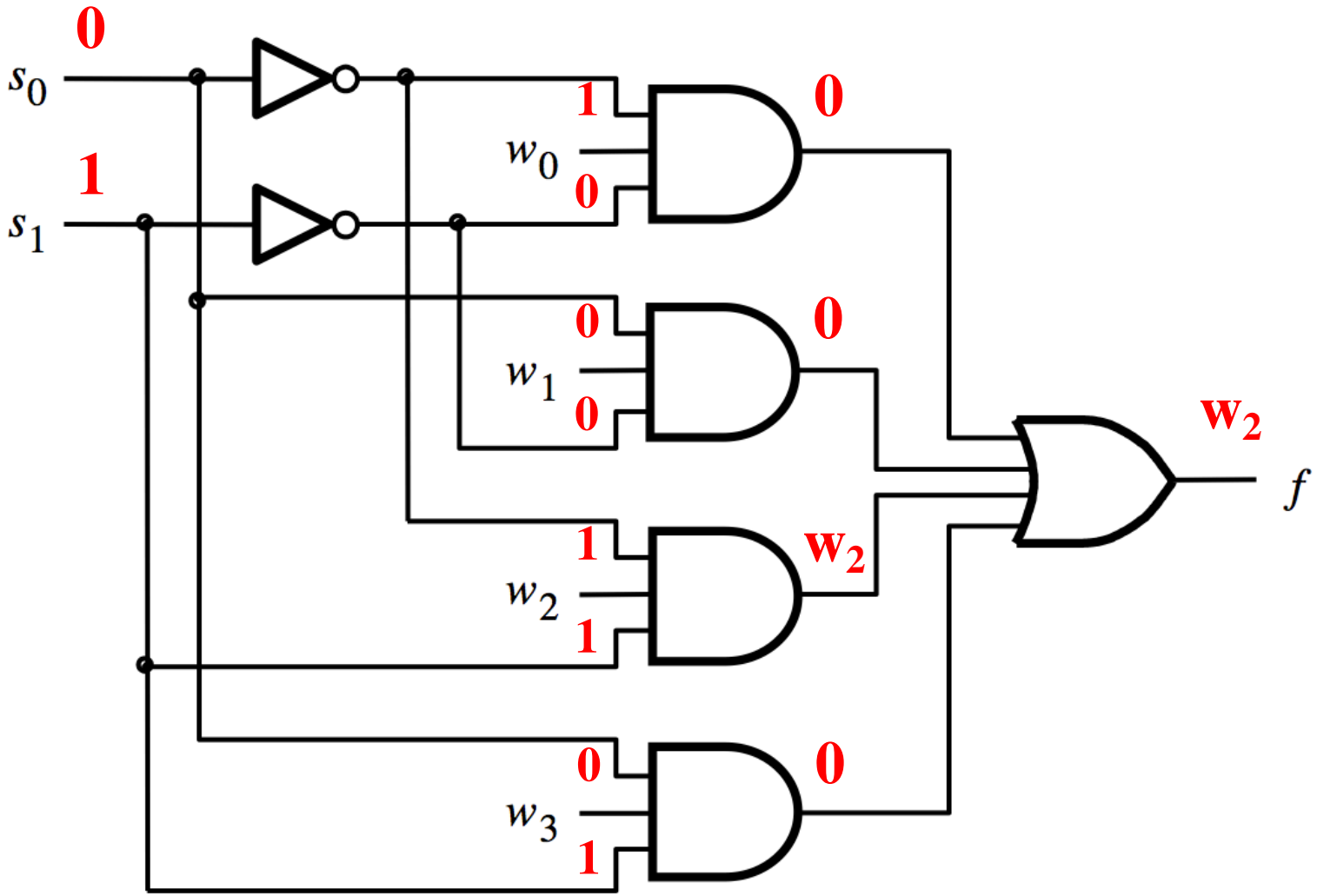
Analysis of the 4-to-1 Multiplexer (s_1=0 and s_0=0)



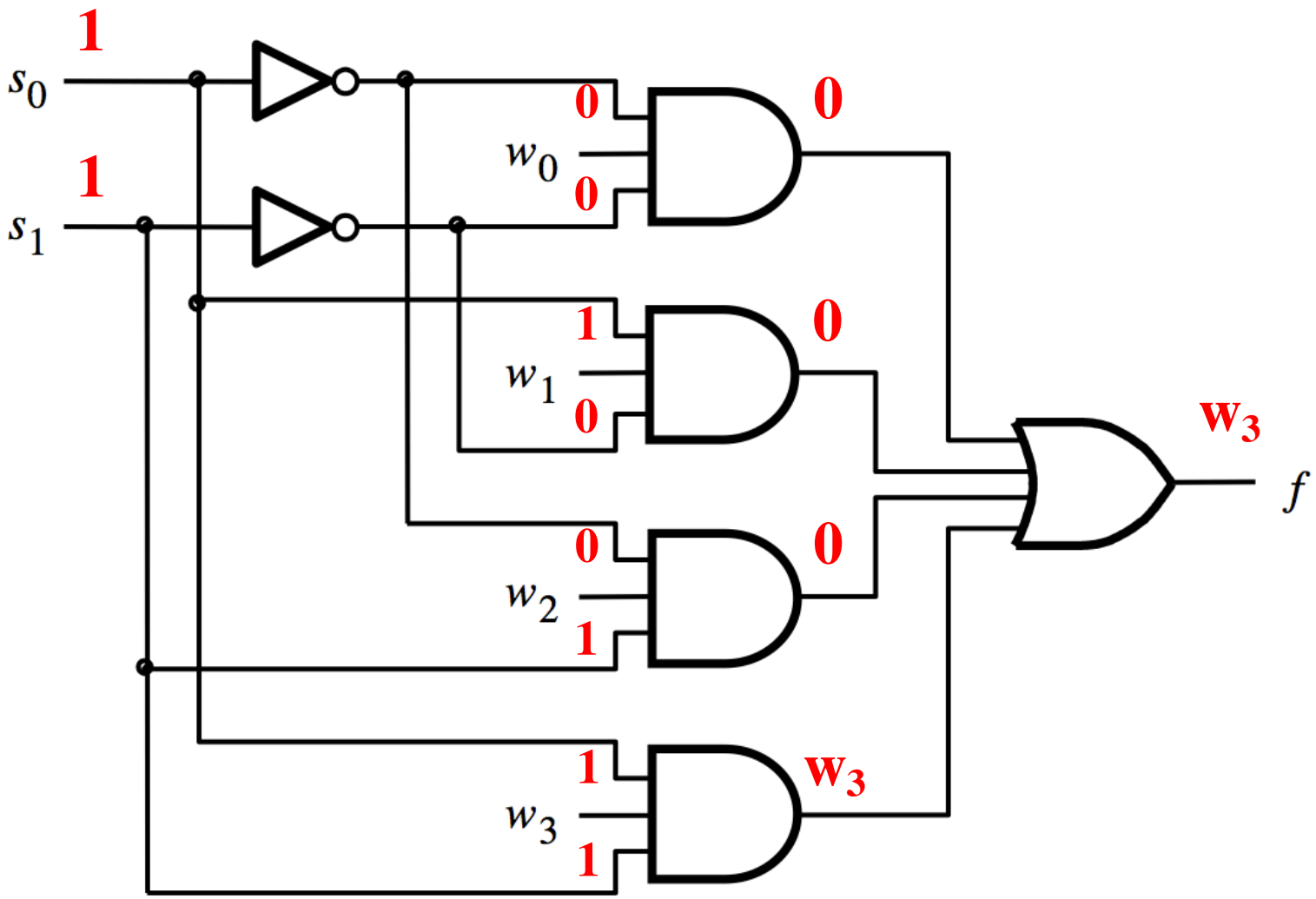
Analysis of the 4-to-1 Multiplexer (s_1=0 and s_0=1)



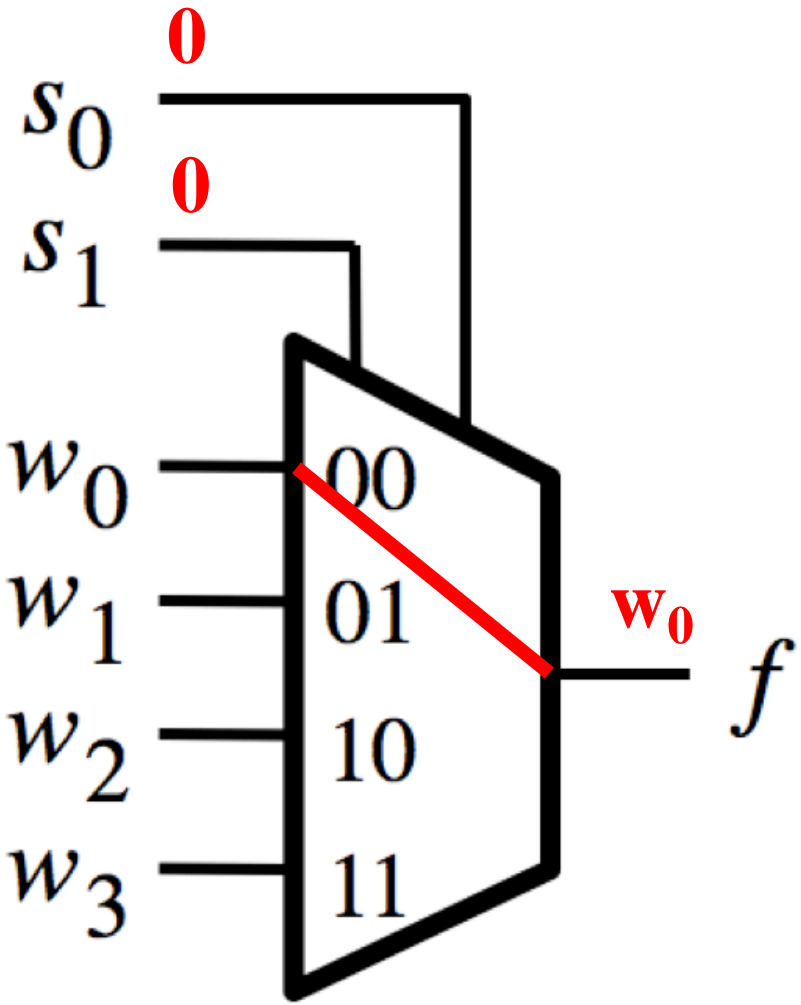
Analysis of the 4-to-1 Multiplexer (s_1=1 and s_0=0)



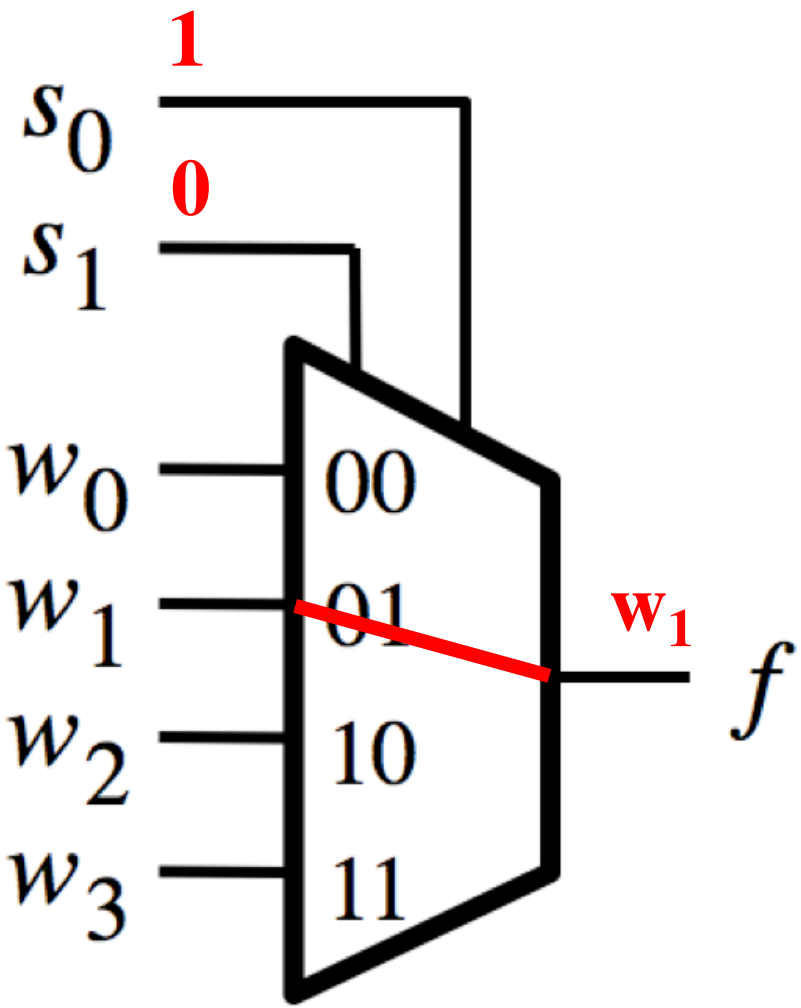
Analysis of the 4-to-1 Multiplexer (s_1=1 and s_0=1)



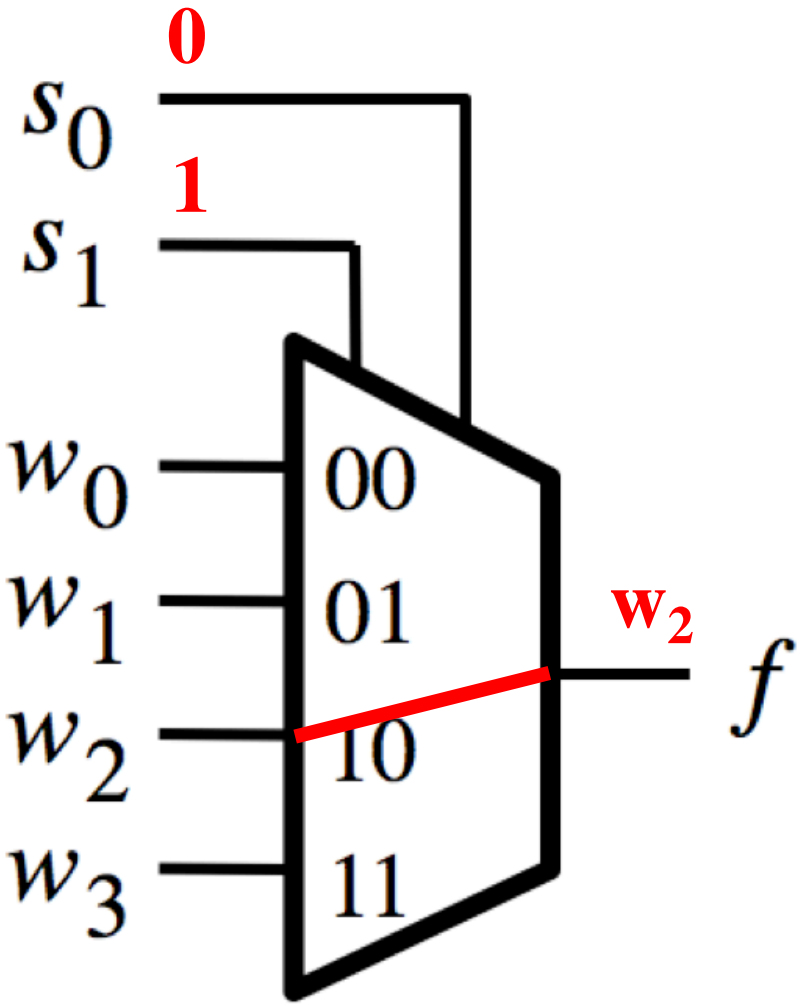
Analysis of the 4-to-1 Multiplexer (s_1=0 and s_0=0)



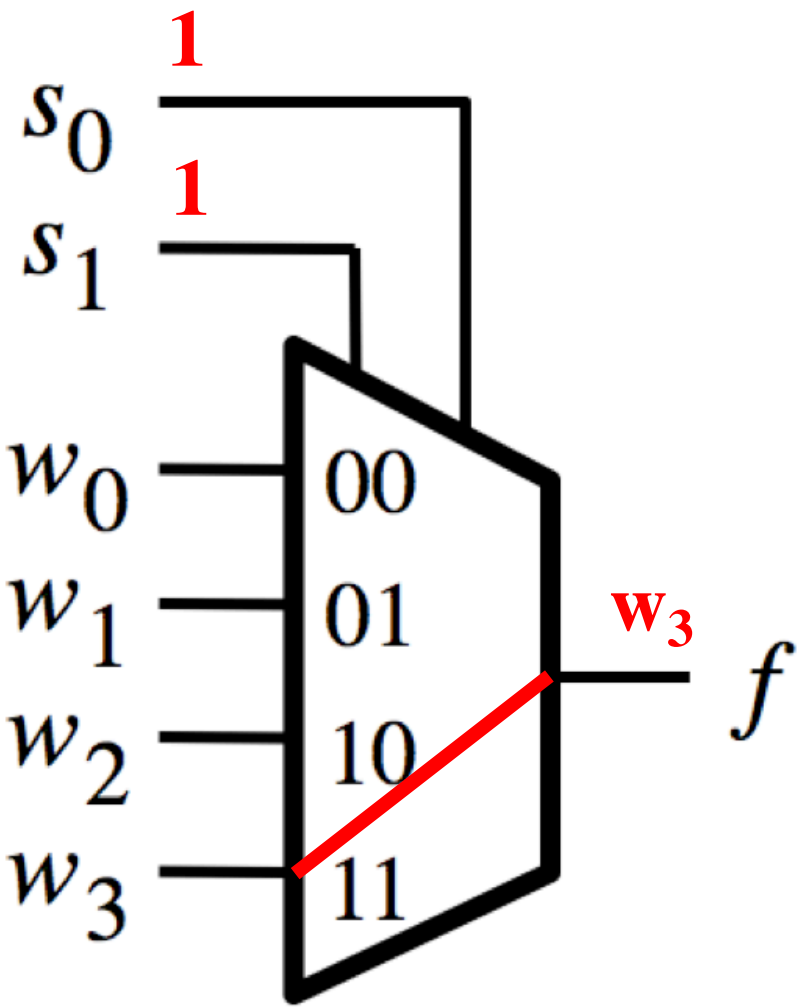
Analysis of the 4-to-1 Multiplexer (s_1=0 and s_0=1)



Analysis of the 4-to-1 Multiplexer (s_1=1 and s_0=0)

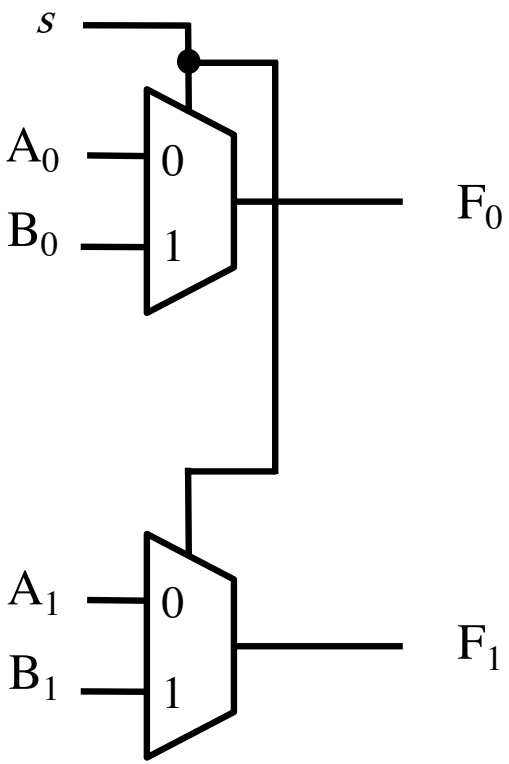


Analysis of the 4-to-1 Multiplexer
(s_1=1 and s_0=1)

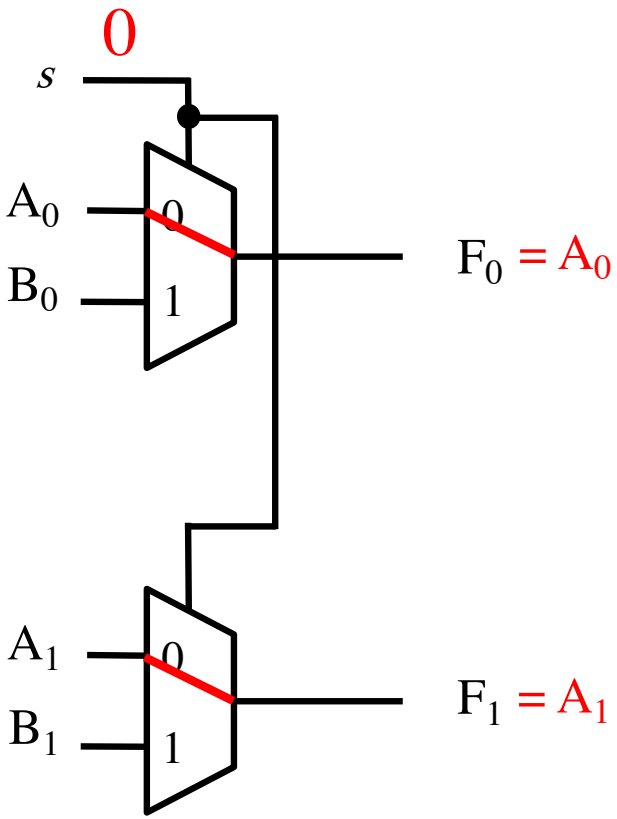


Multiplexer Tricks
(select one of two 2-bit numbers)

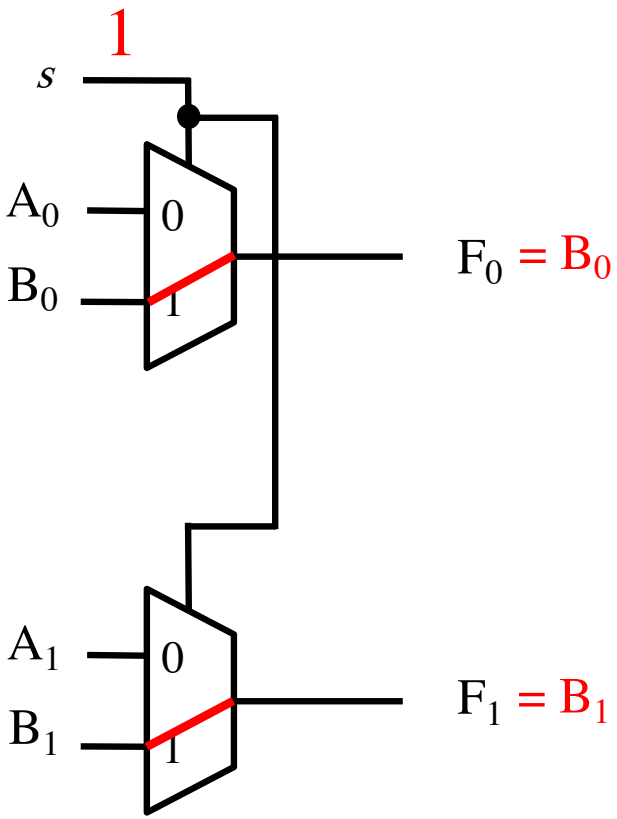
Select Either $A=A_1A_0$ or $B=B_1B_0$



Select Either $A=A_1A_0$ or $B=B_1B_0$

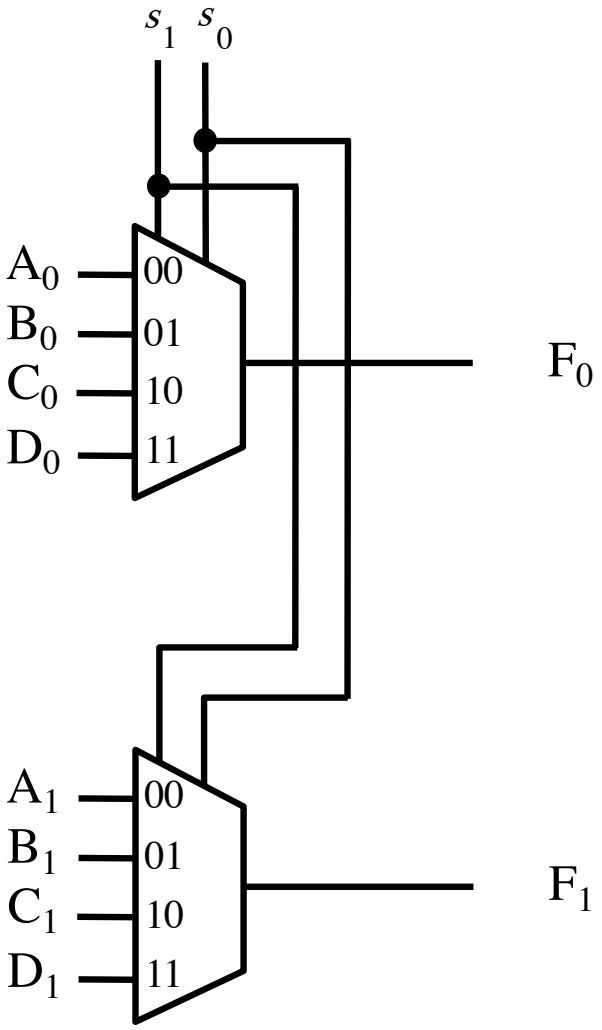


Select Either $A=A_1A_0$ or $B=B_1B_0$

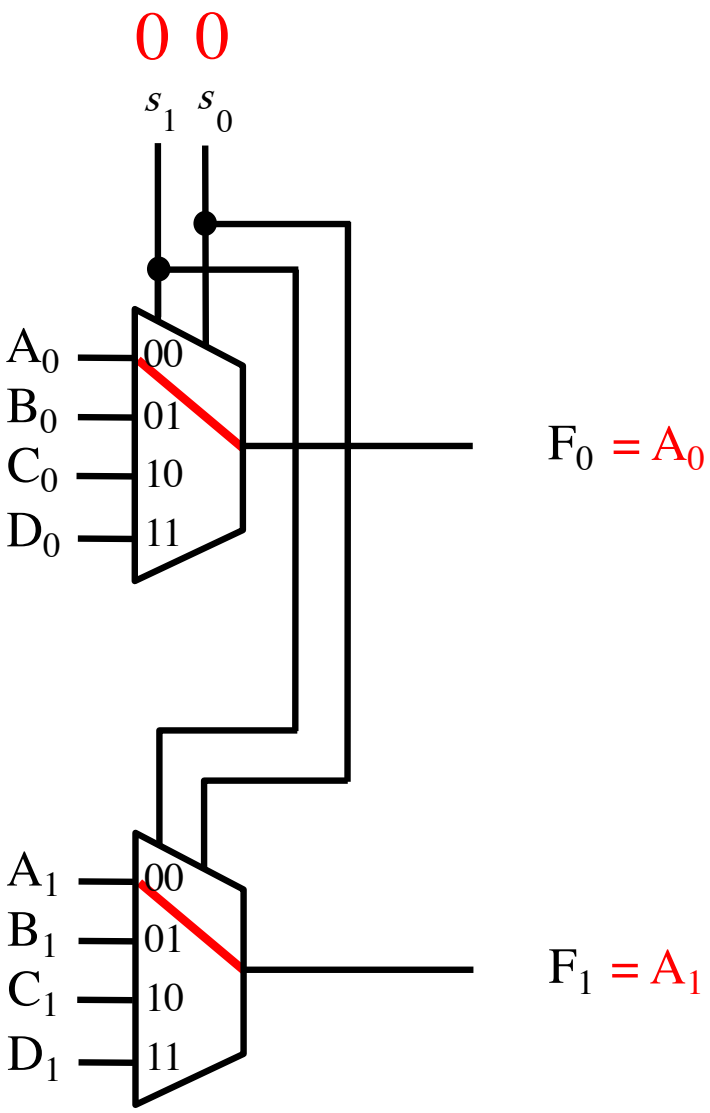


Multiplexer Tricks
(select one of four 2-bit numbers)

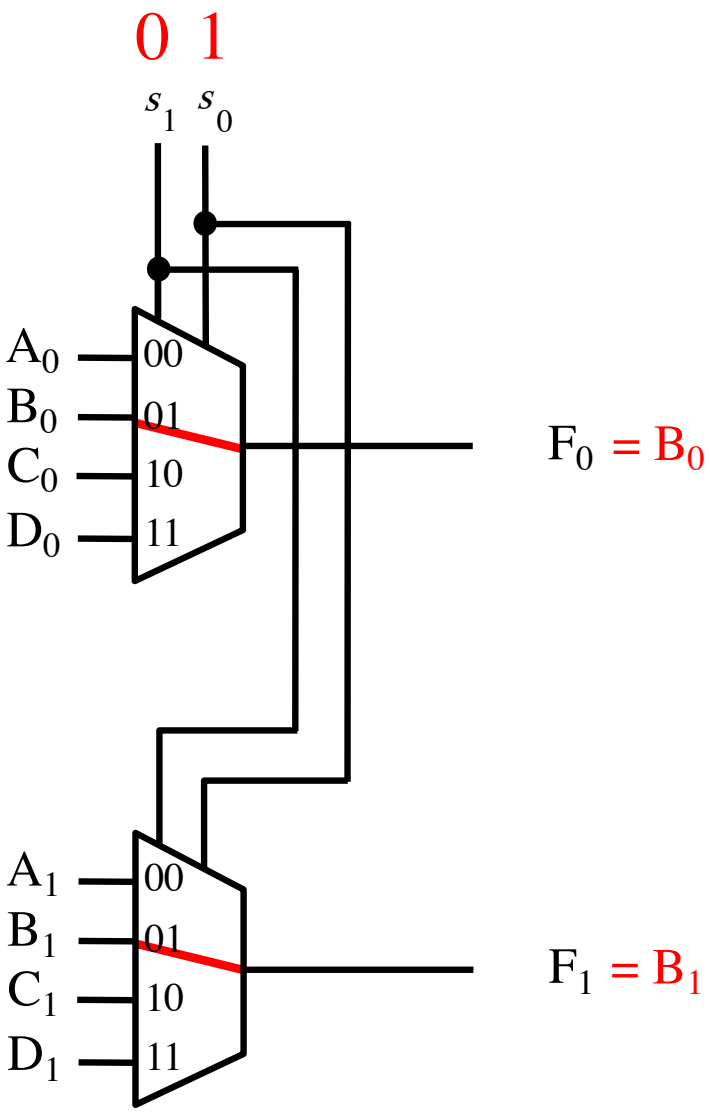
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



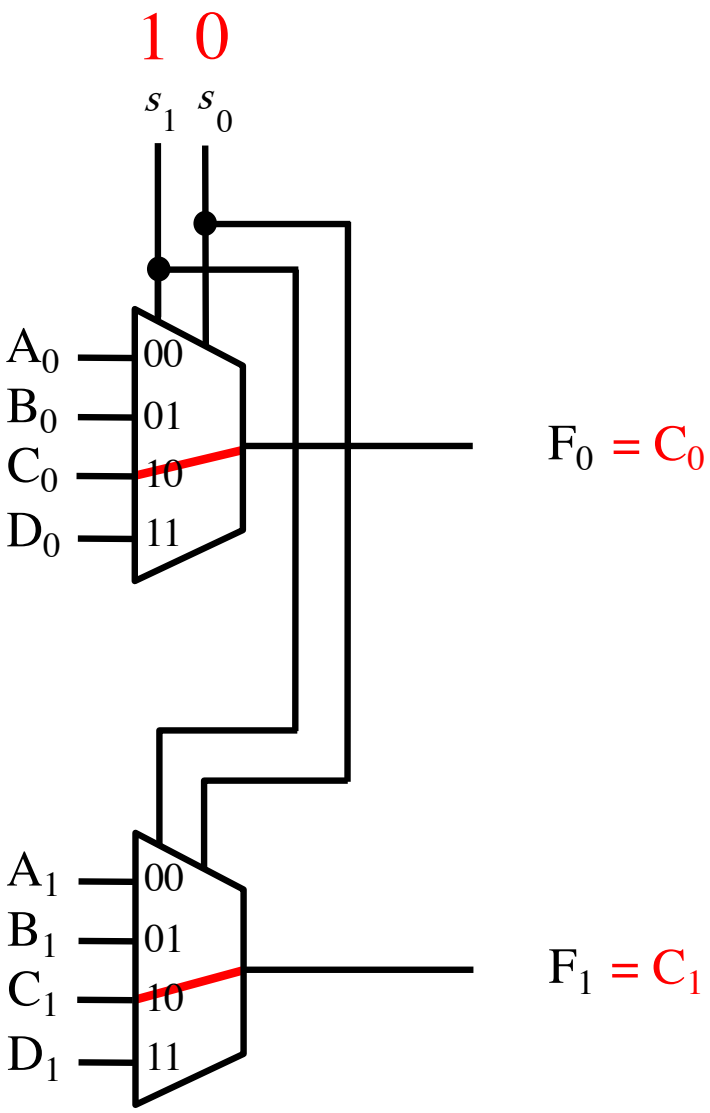
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



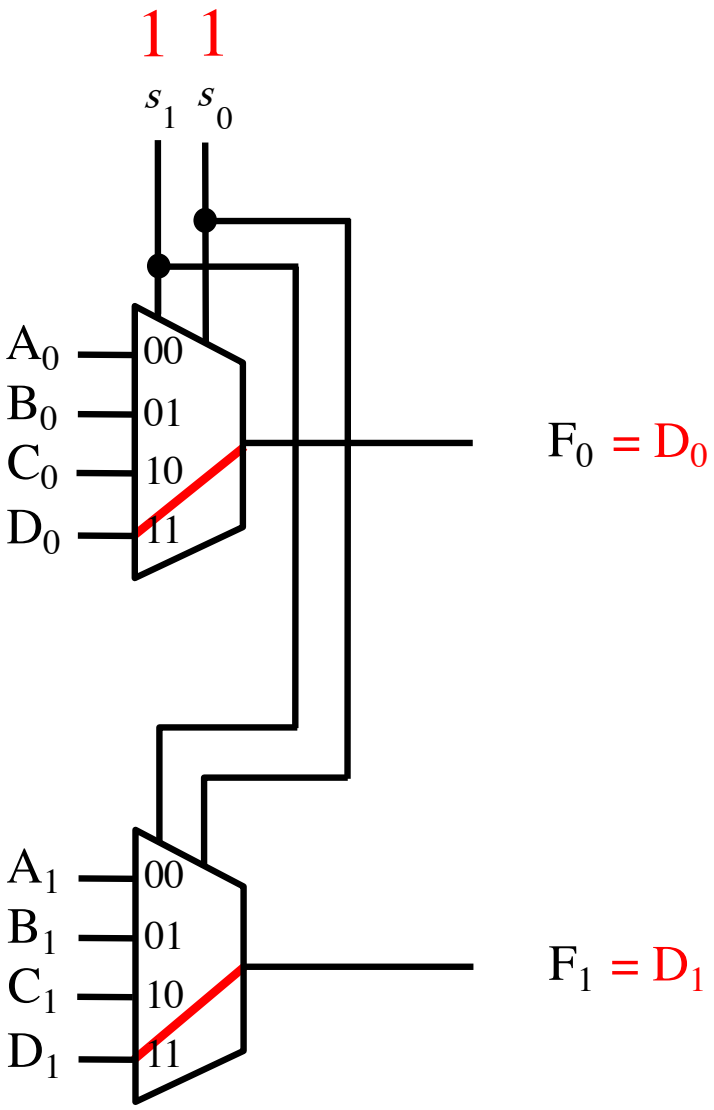
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



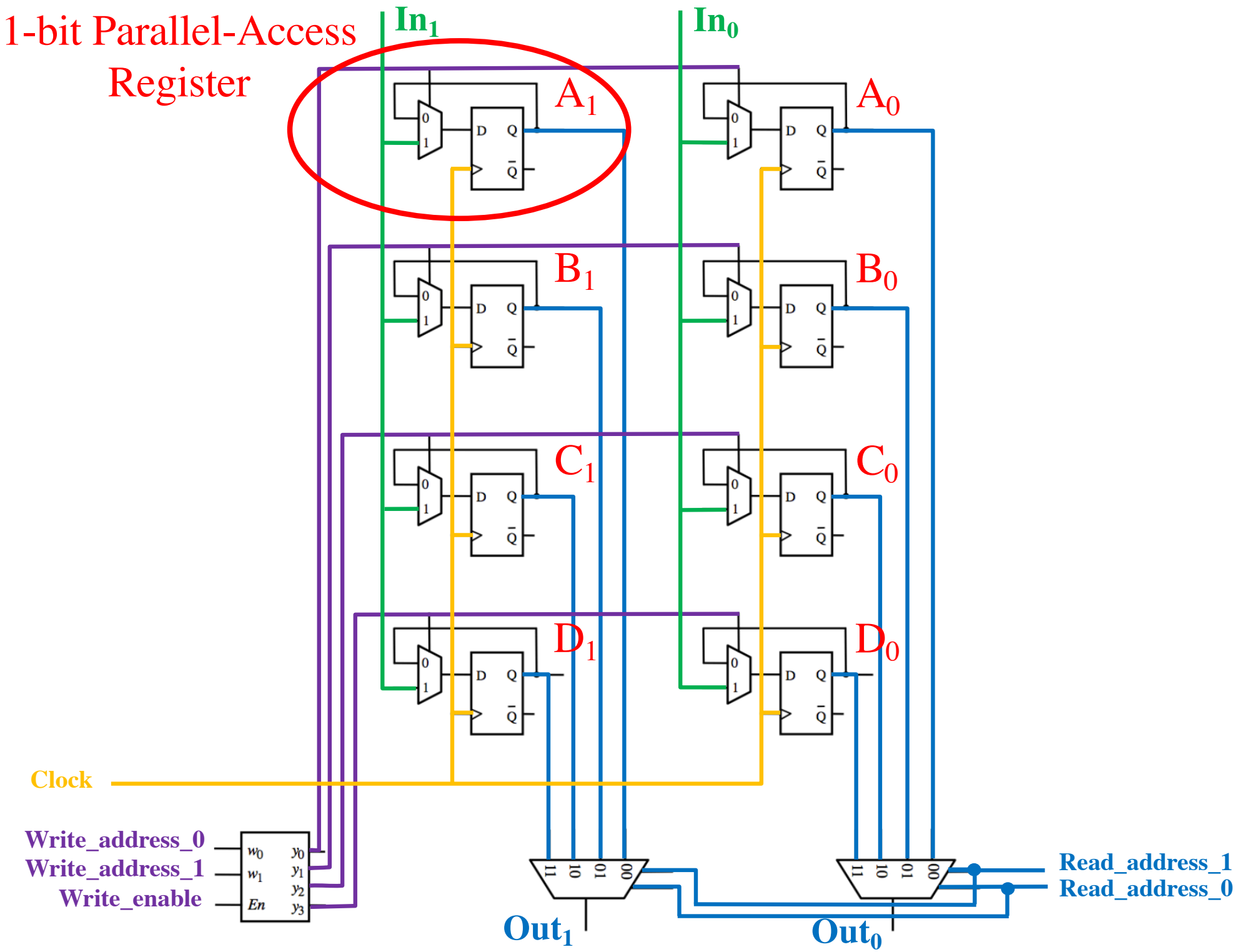
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$

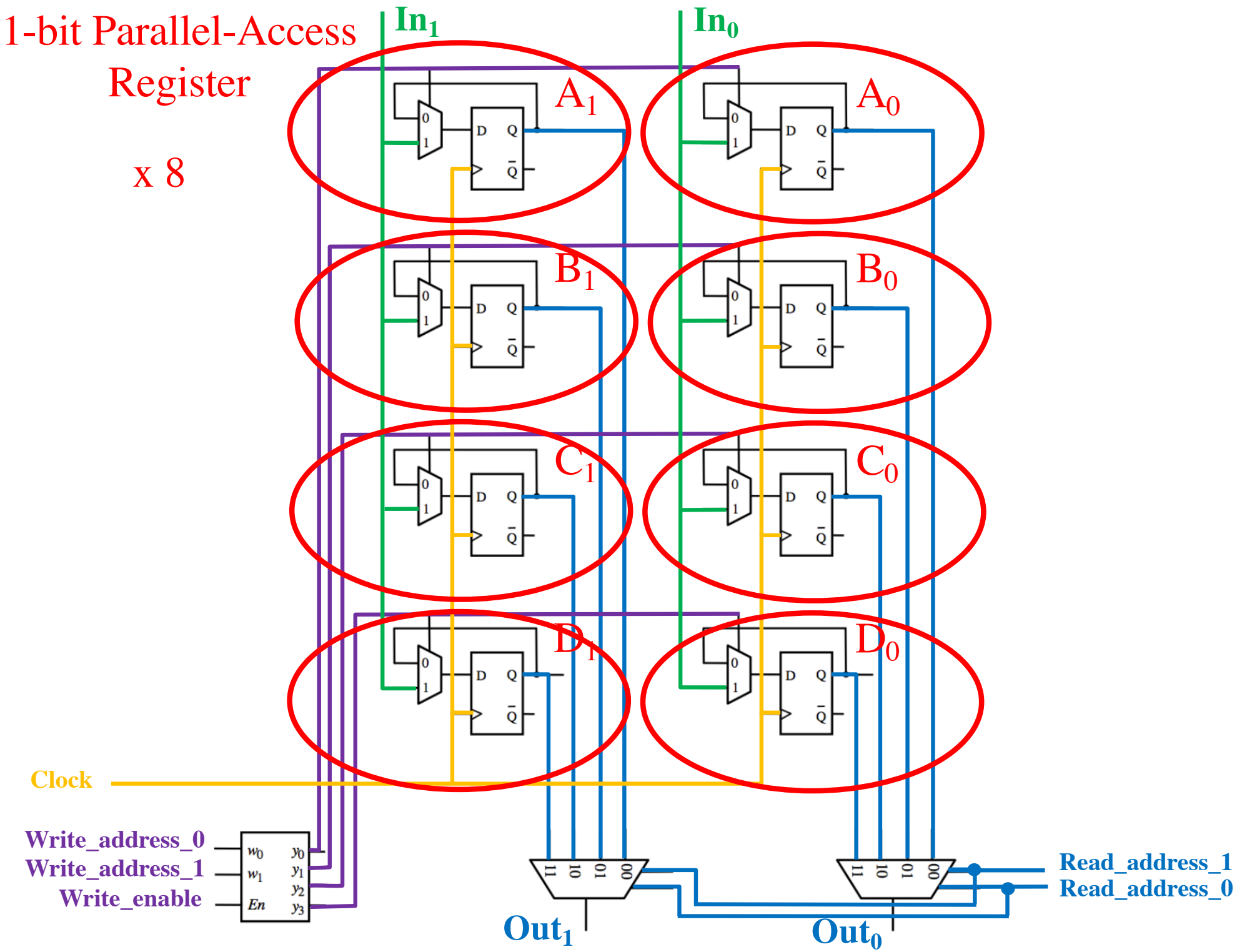


Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$

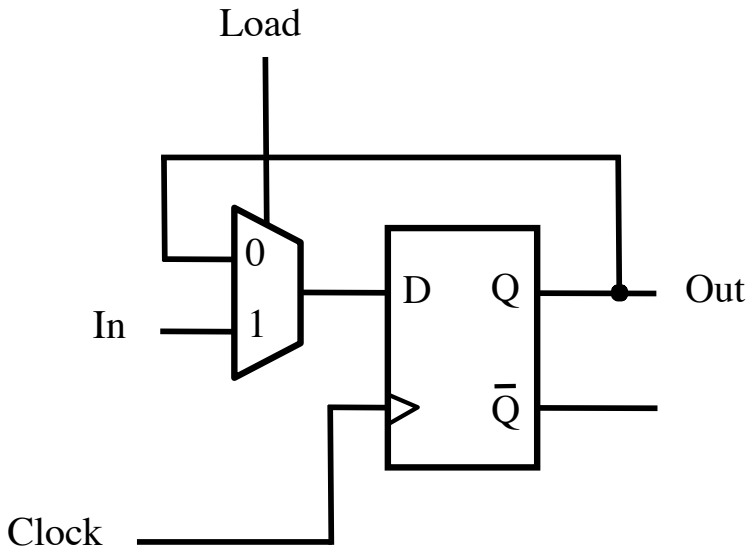


1-bit Parallel-Access Register

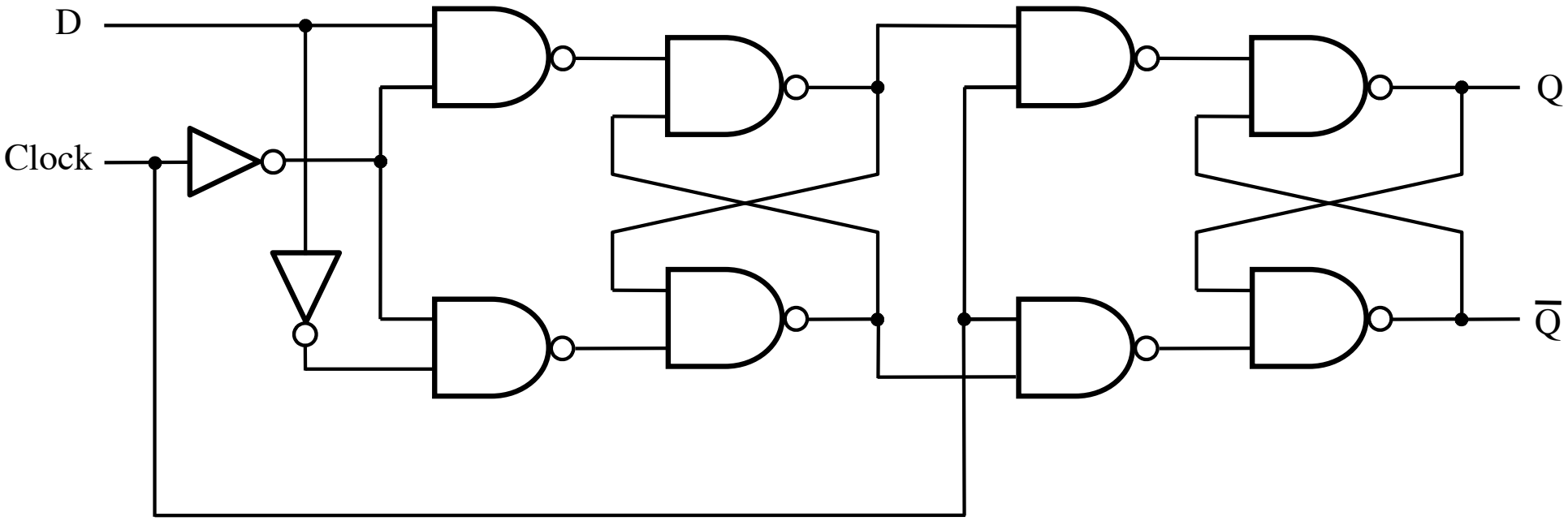

1-bit Parallel-Access Register
In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
D
Q
Q̄
0
1
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
Read_address_1
Read_address_0


1-bit Parallel-Access Register
x 8
In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
D
Q
Q̄
0
1
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
Read_address_1
Read_address_0

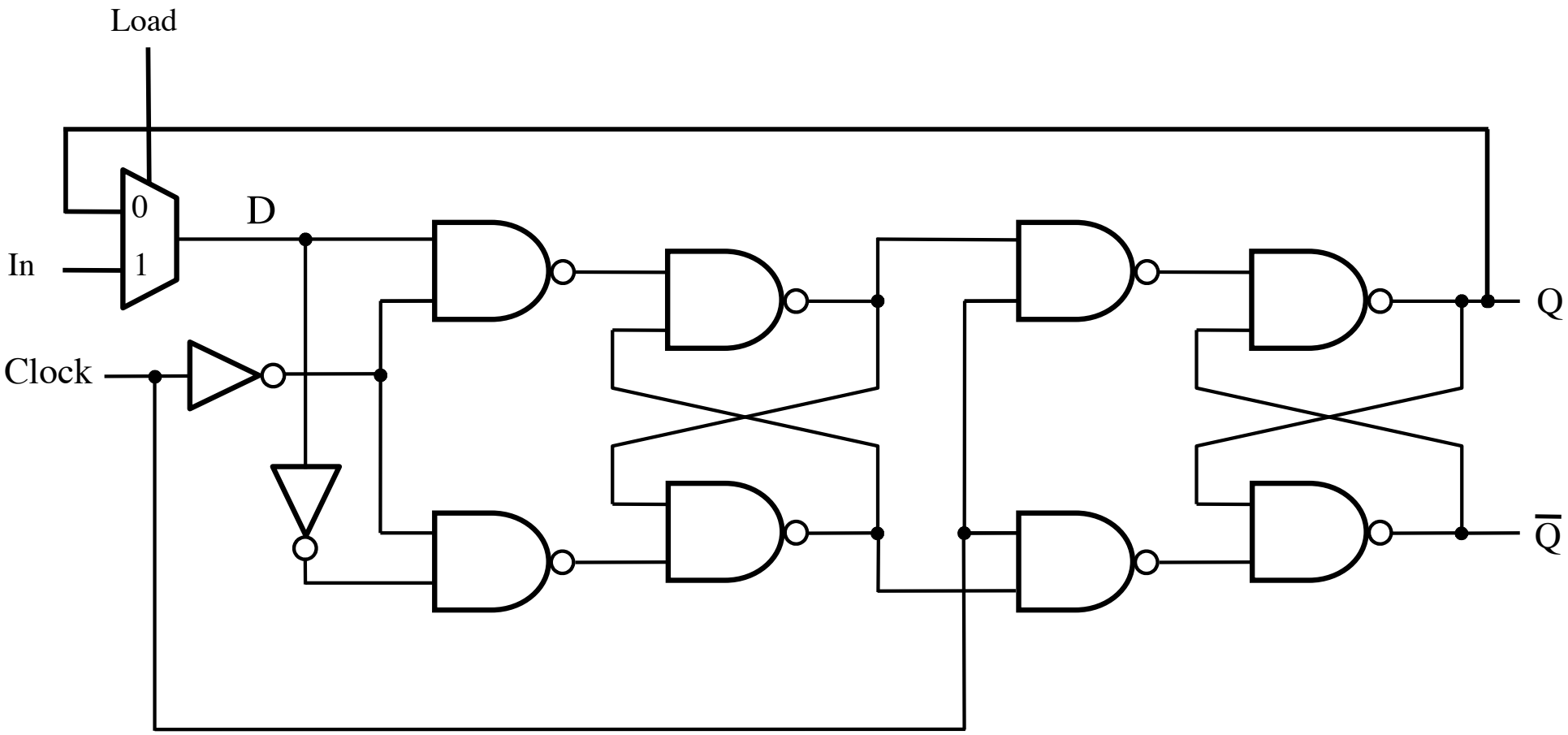
1-Bit Parallel-Access Register



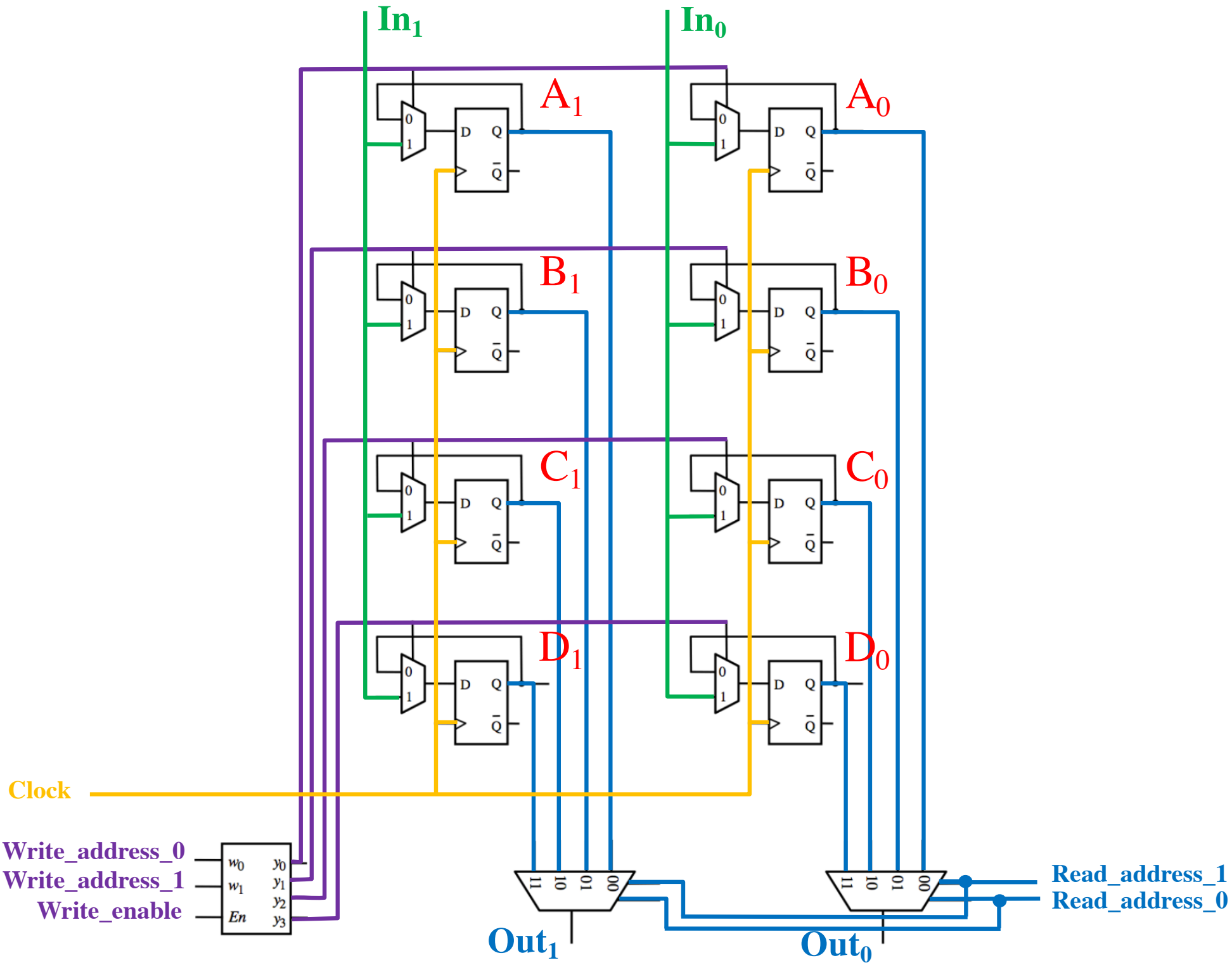
Positive-Edge-Triggered D Flip-Flop



1-bit Parallel-Access Register

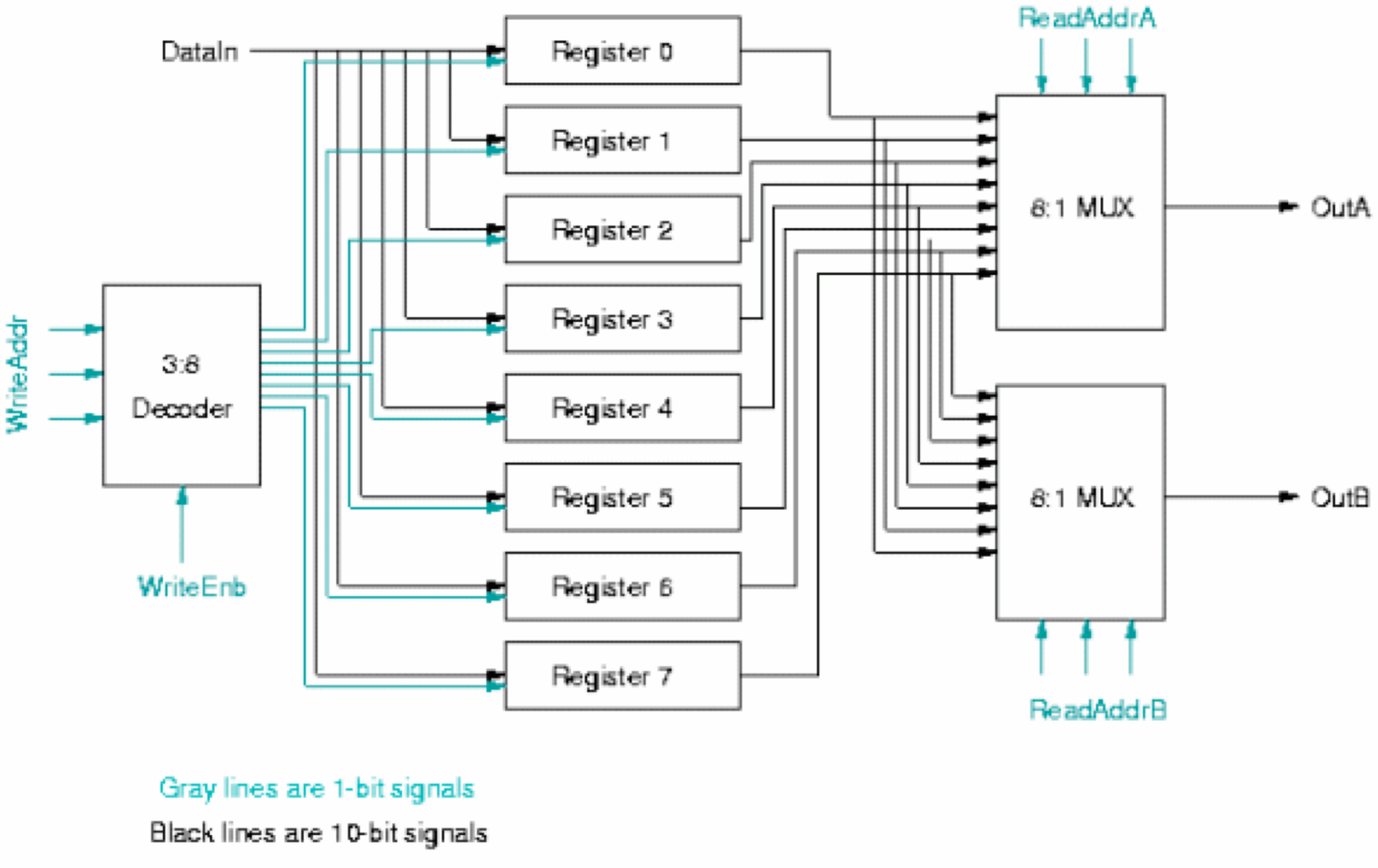


Putting it all Together


In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
D
Q
Q̄
0
1
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
Read_address_1
Read_address_0

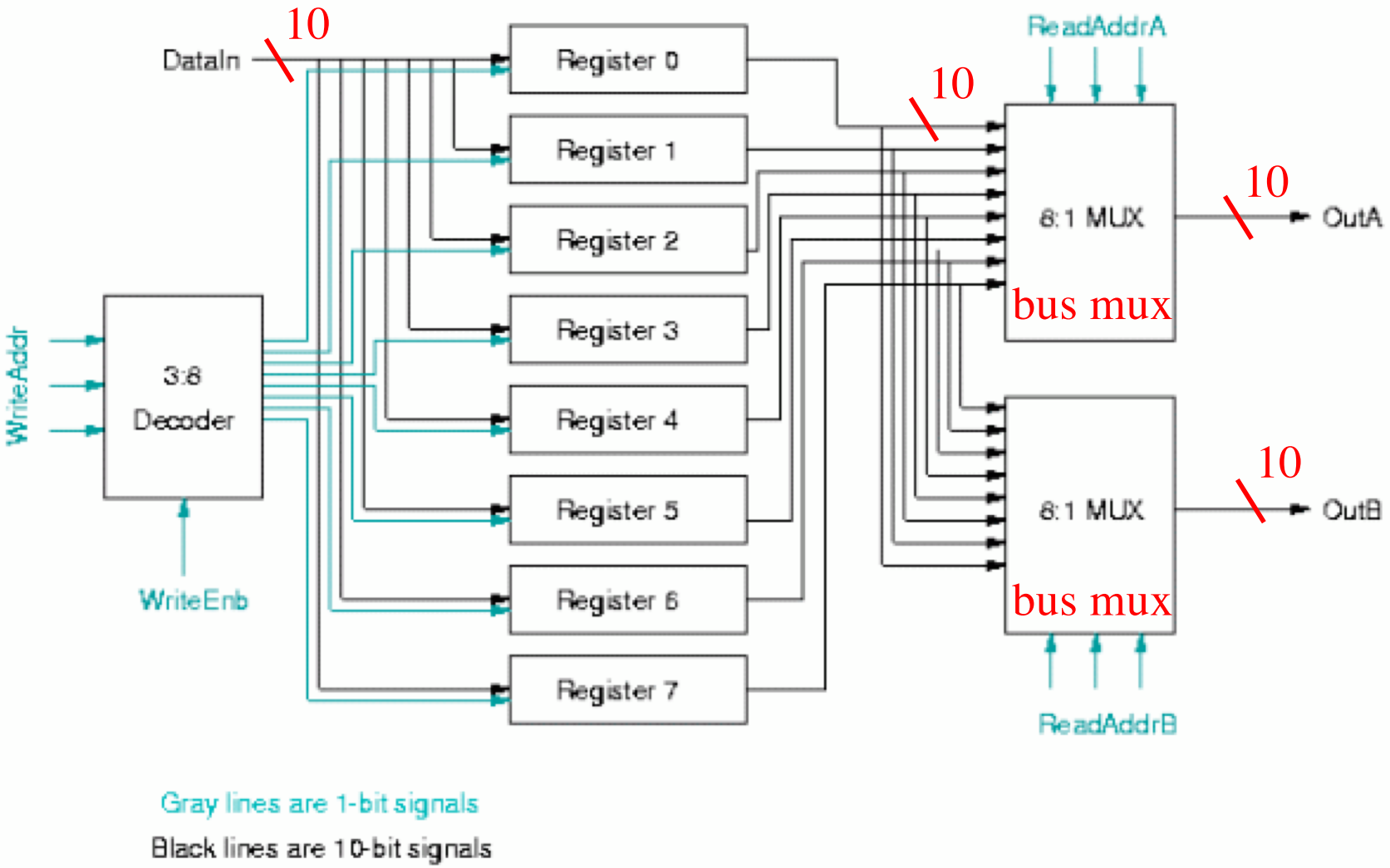
Register file with eight 10-bit registers and two read ports

Register file with 2 read ports



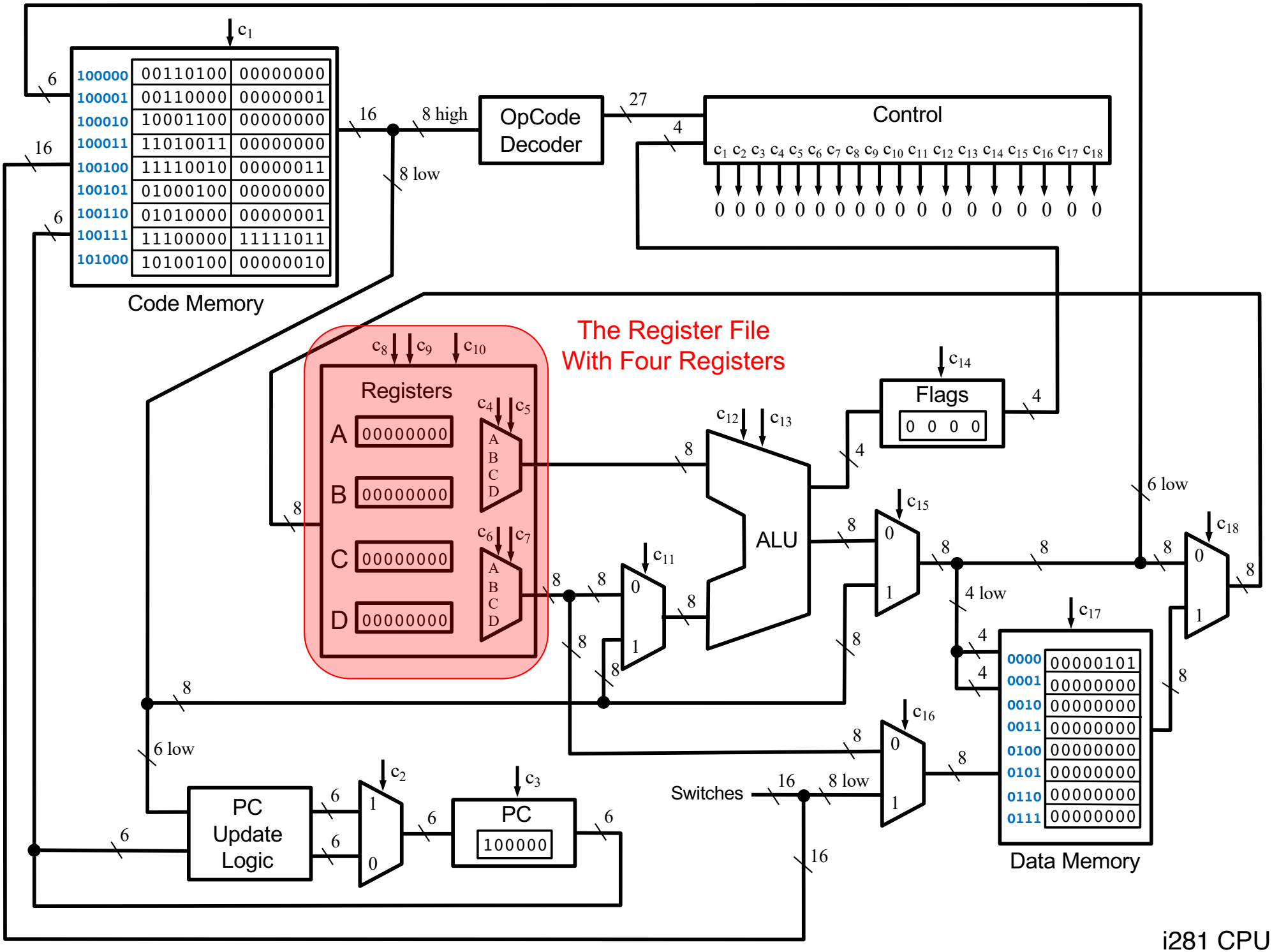
[https://jindongpu.wordpress.com/2012/03/07/register-file/]

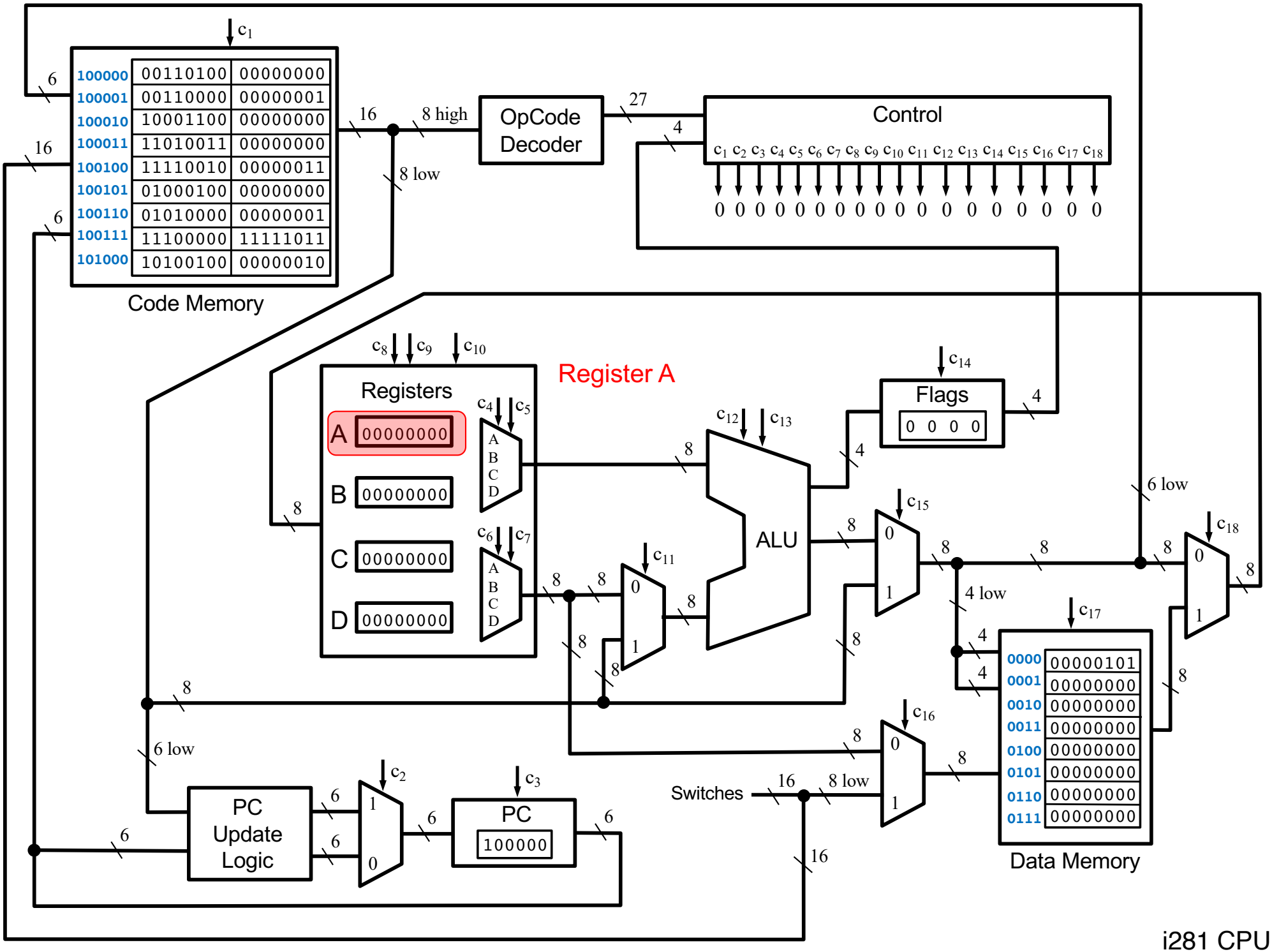
Register file with 2 read ports

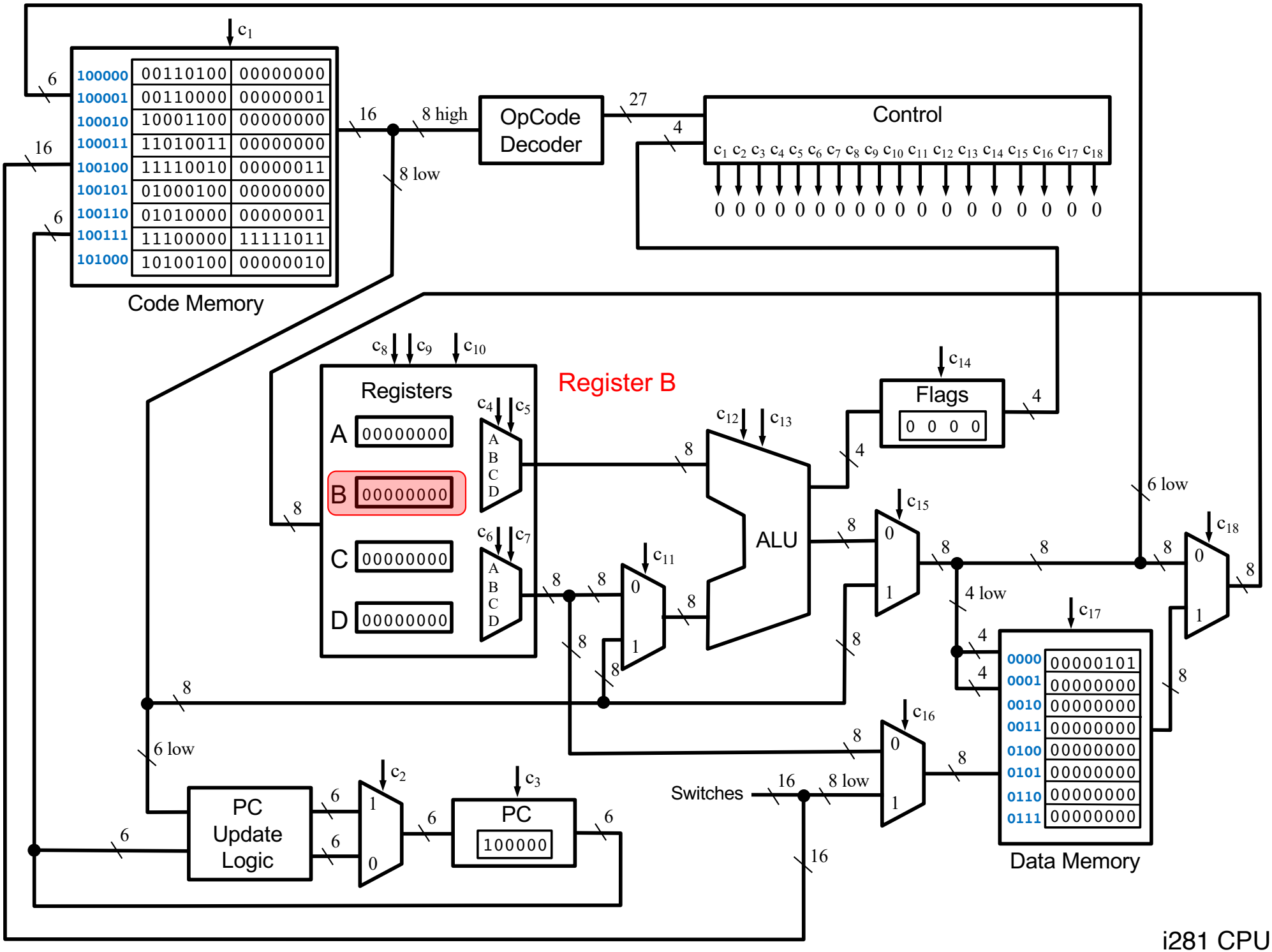


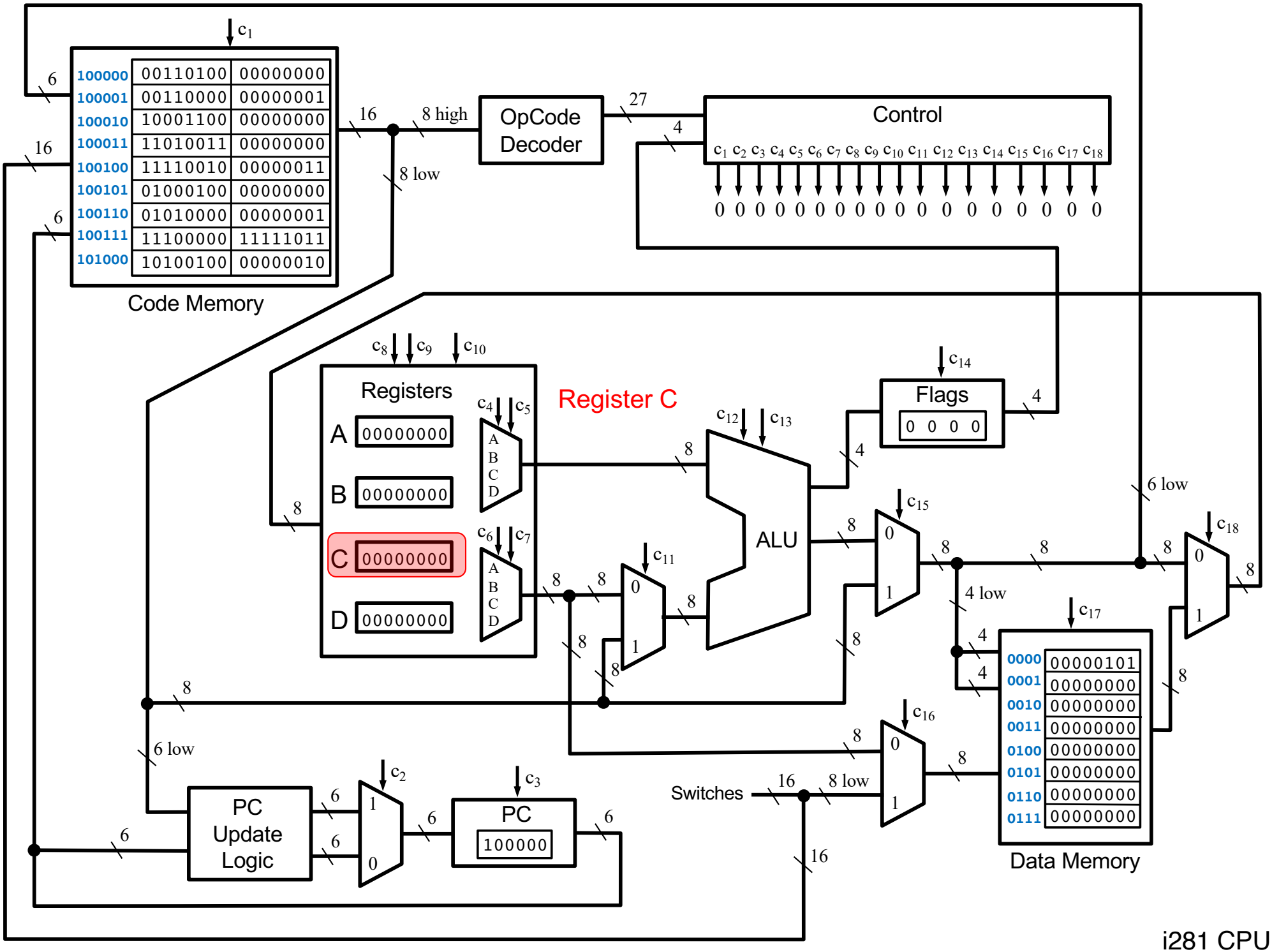
[https://jindongpu.wordpress.com/2012/03/07/register-file/]

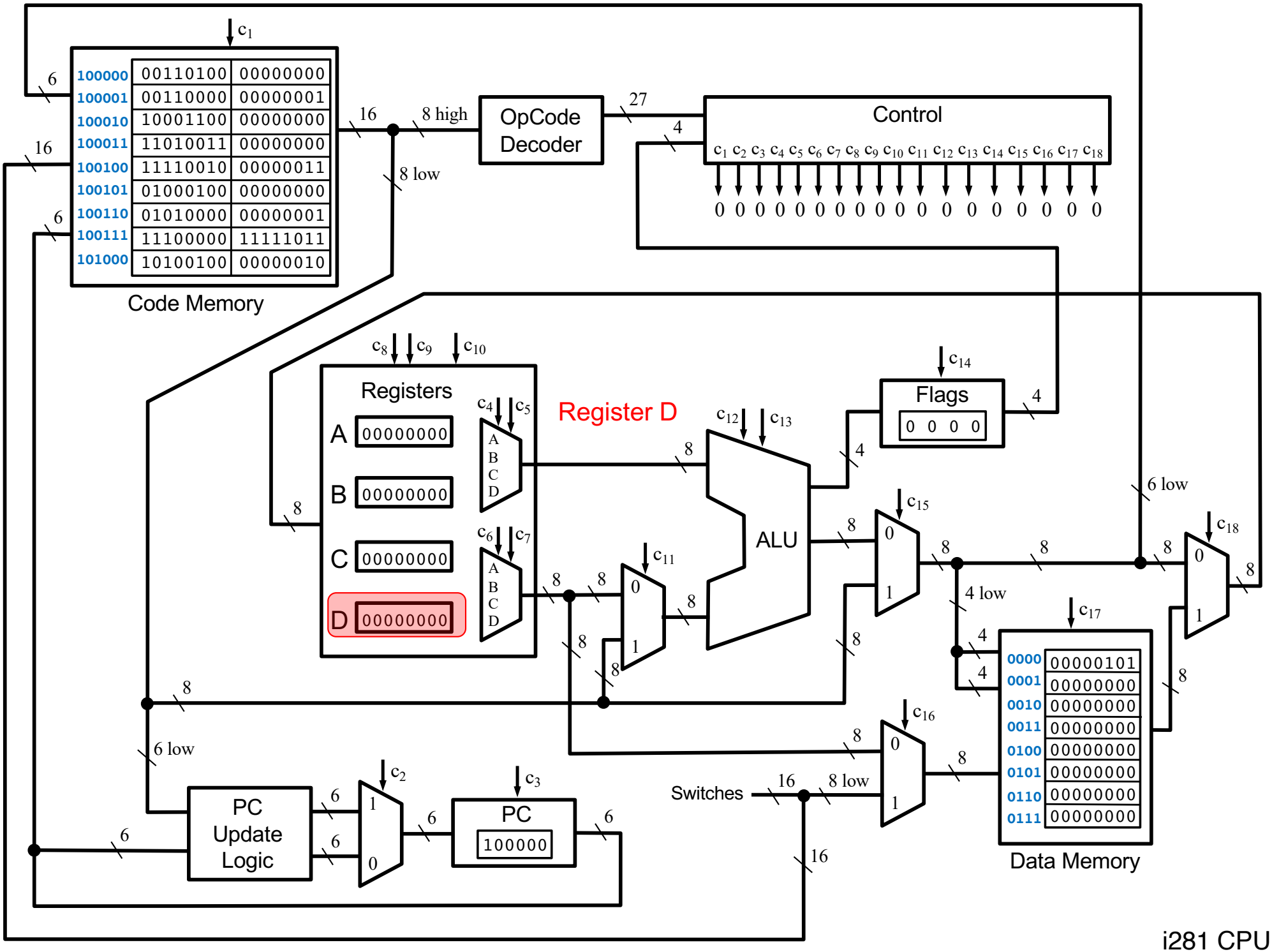
The i281 CPU has a register file with four 8-bit registers and two read ports


c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
6
16
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
The Register File
With Four Registers
c8 c9 c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
c4 c5
c6 c7
A B C D
c14
Flags
0 0 0 0
4
c12 c13
ALU
c11
c15
c16
c17
c18
6 low
4 low
8 low
8 high
Switches
PC Update Logic
c2
c3
PC
100000
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
i281 CPU

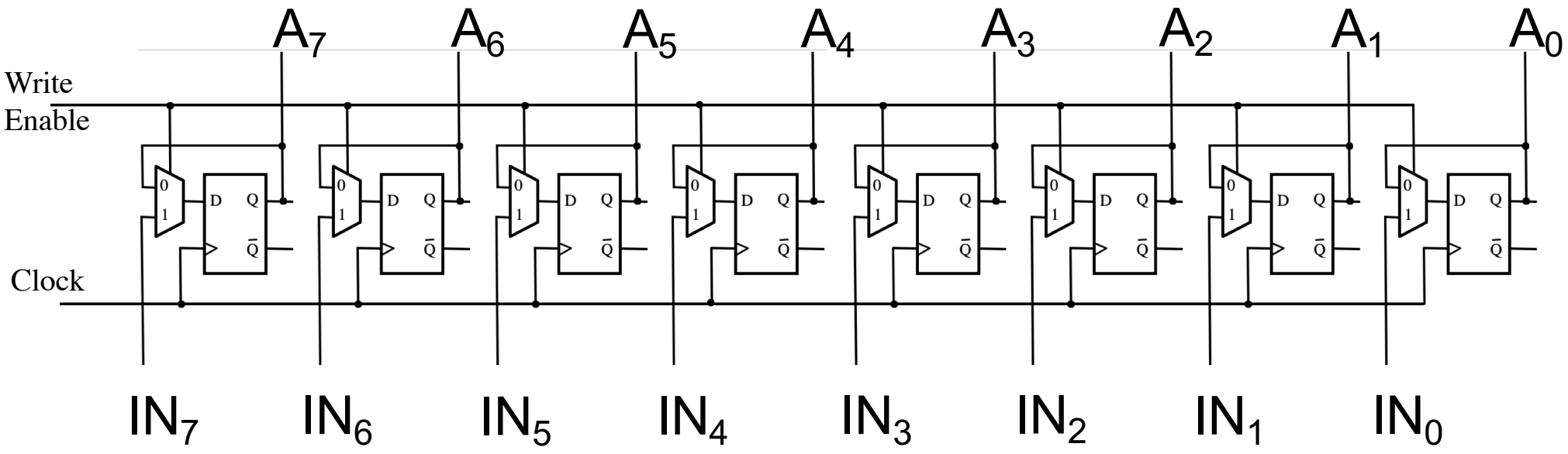

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
6
16
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Register A
c8 c9 c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
c4 c5
c6 c7
A B C D
c11
c12 c13
ALU
c14
Flags
0 0 0 0
4
c15
c18
6 low
4 low
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
8 low
c2
c3
PC Update Logic
PC
100000
i281 CPU


c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
6
16
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Register B
c8 c9 c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
c4 c5
c6 c7
A B C D
c11
c12 c13
ALU
c14
Flags
0 0 0 0
4
c15
c18
6 low
4 low
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
8 low
c2
c3
PC Update Logic
PC
100000
i281 CPU


c1
6
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
c8 c9 c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
c4 c5
c6 c7
A B C D
Register C
c12 c13
ALU
c14
Flags
0 0 0 0
c11
c15
c16
c17
c18
6 low
4 low
Switches
16
8 low
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c2
c3
PC Update Logic
PC
100000
i281 CPU

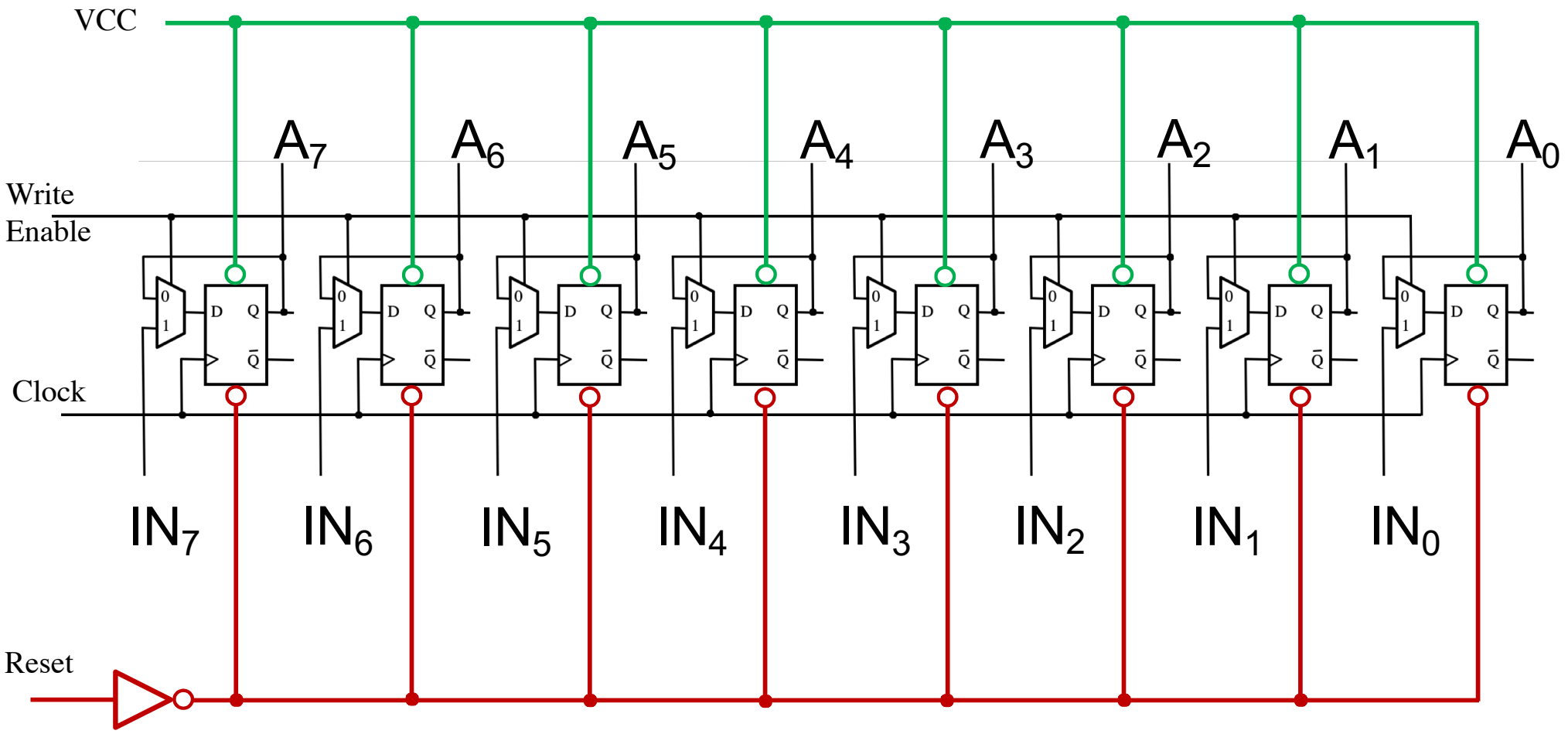

c1
6
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
c8 c9 c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
c4 c5
c6 c7
A B C D
Register D
c12 c13
ALU
c14
Flags
0 0 0 0
4
8
c11
c15
c18
6 low
4 low
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
8 low
c2
c3
PC Update Logic
PC
100000
i281 CPU

Register A

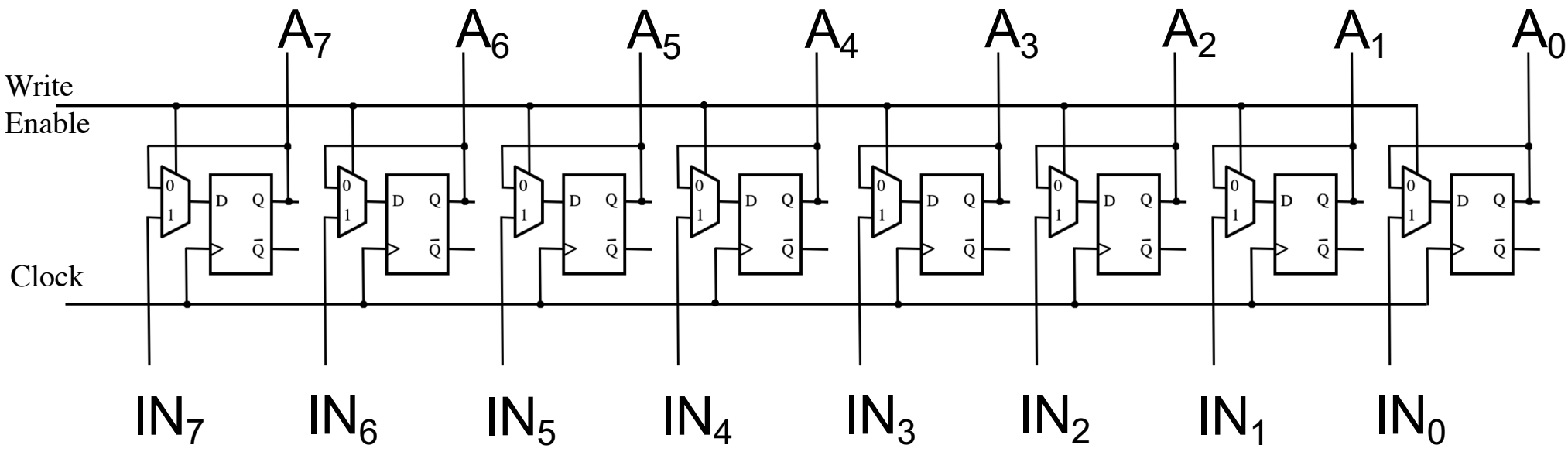


8-Bit Parallel-Access Register

Register A

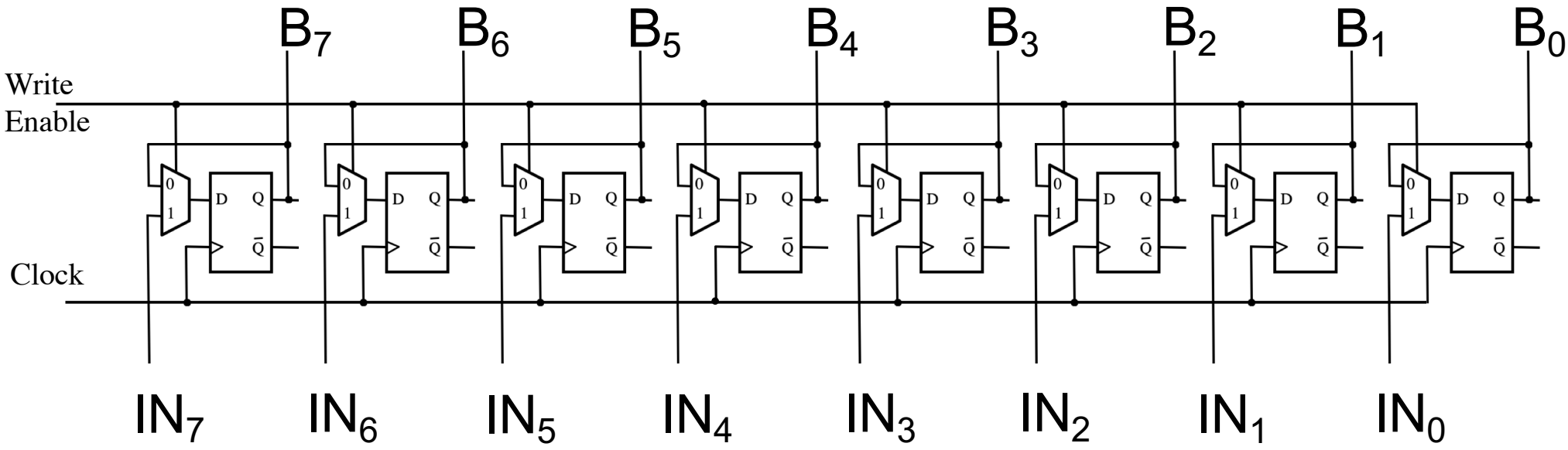

VCC
A7
A6
A5
A4
A3
A2
A1
A0
Write Enable
Clock
IN7
IN6
IN5
IN4
IN3
IN2
IN1
IN0
Reset
D
Q
Q̄
0
1

Register A



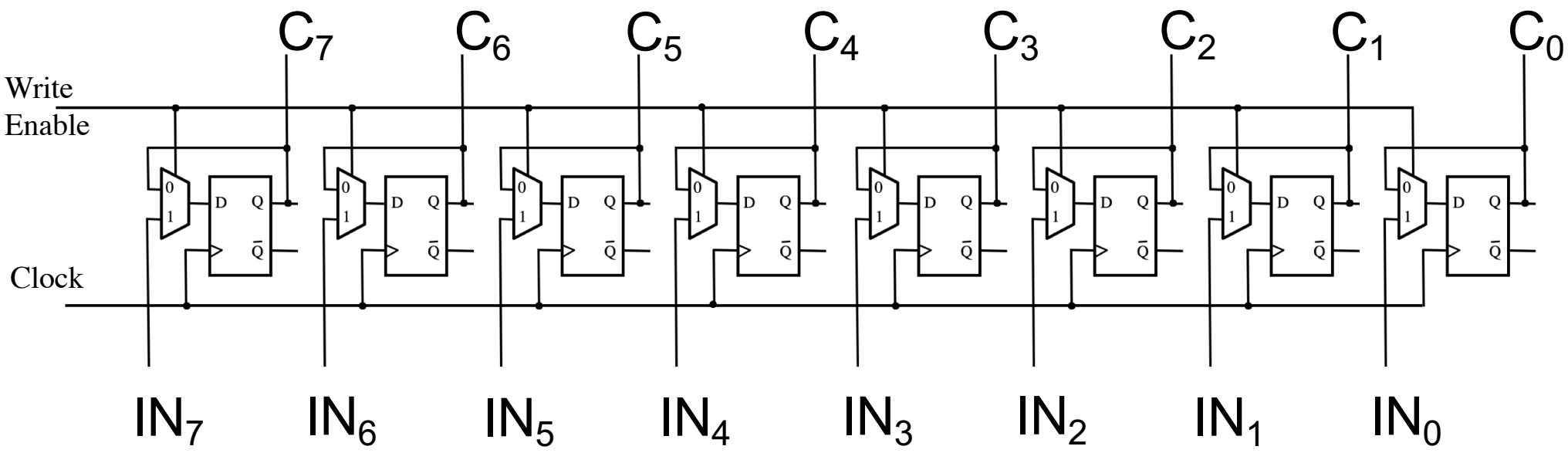
8-Bit Parallel-Access Register

Register B



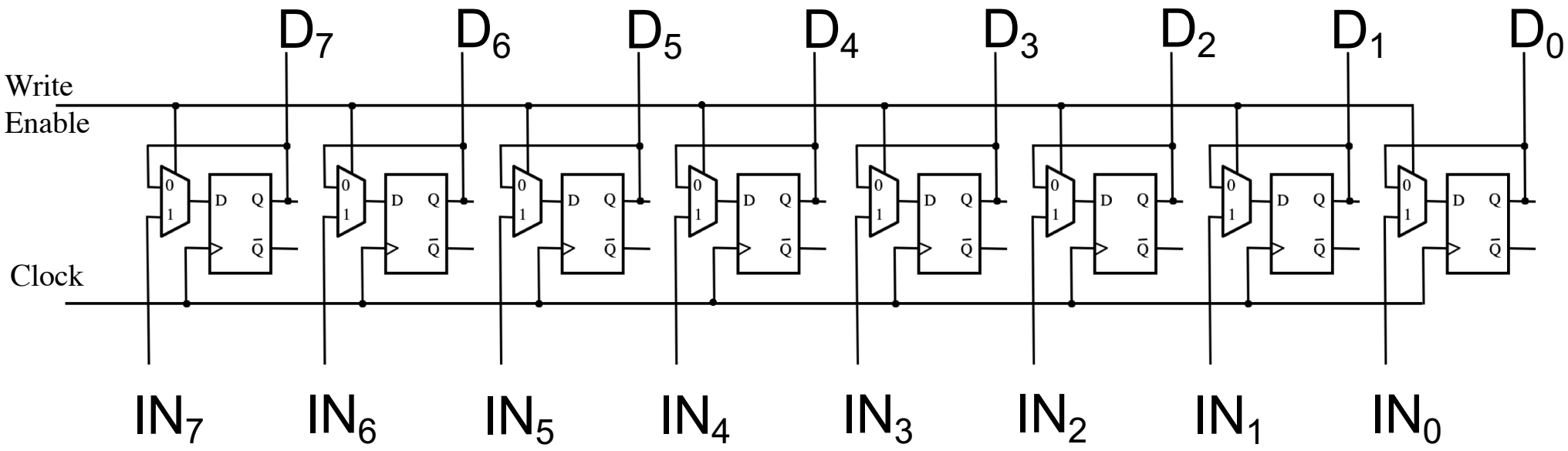
8-Bit Parallel-Access Register

Register C

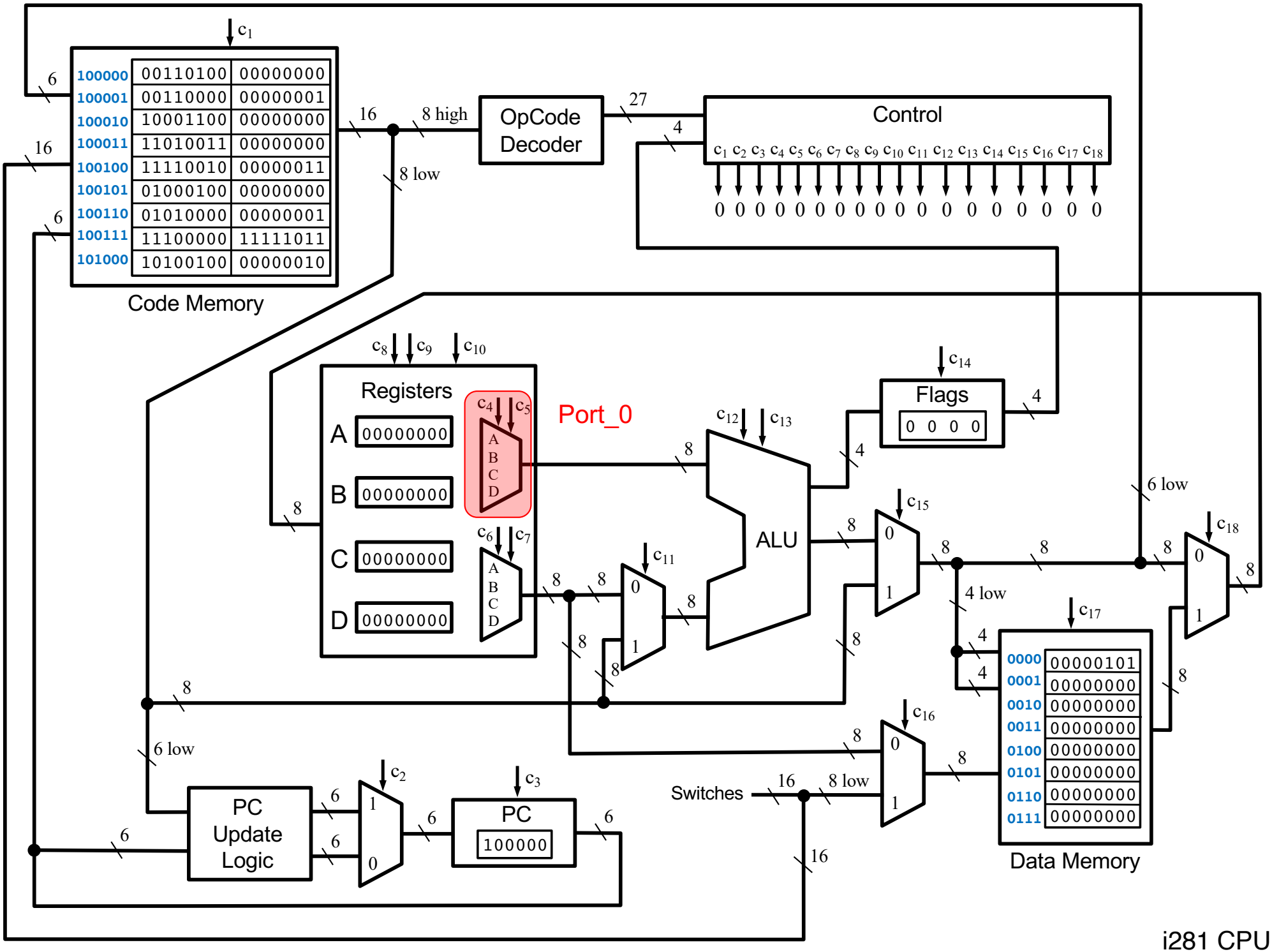


8-Bit Parallel-Access Register

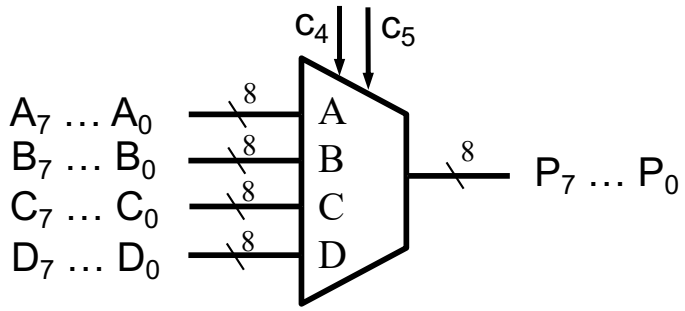
Register D

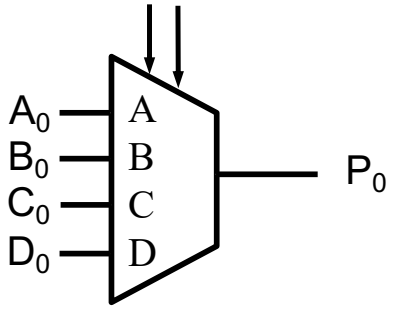


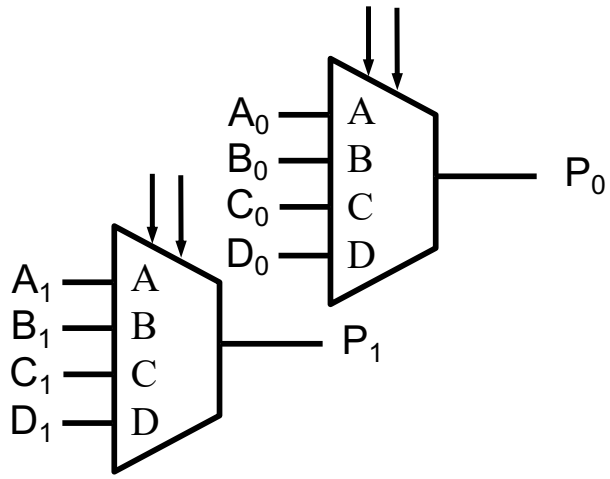
8-Bit Parallel-Access Register

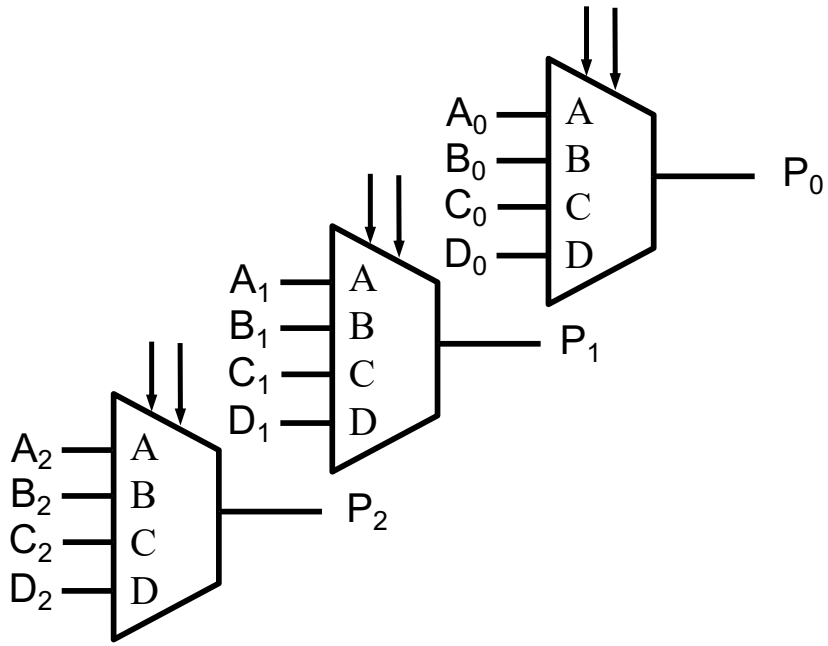

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
6
16
16
6
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
c8 c9 c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
c4 c5
c6 c7
A B C D
Port_0
c12 c13
ALU
c14
Flags
0 0 0 0
4
8
c11
c15
c18
6 low
4 low
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
16
8 low
c2
c3
PC Update Logic
PC
100000
6 low
i281 CPU

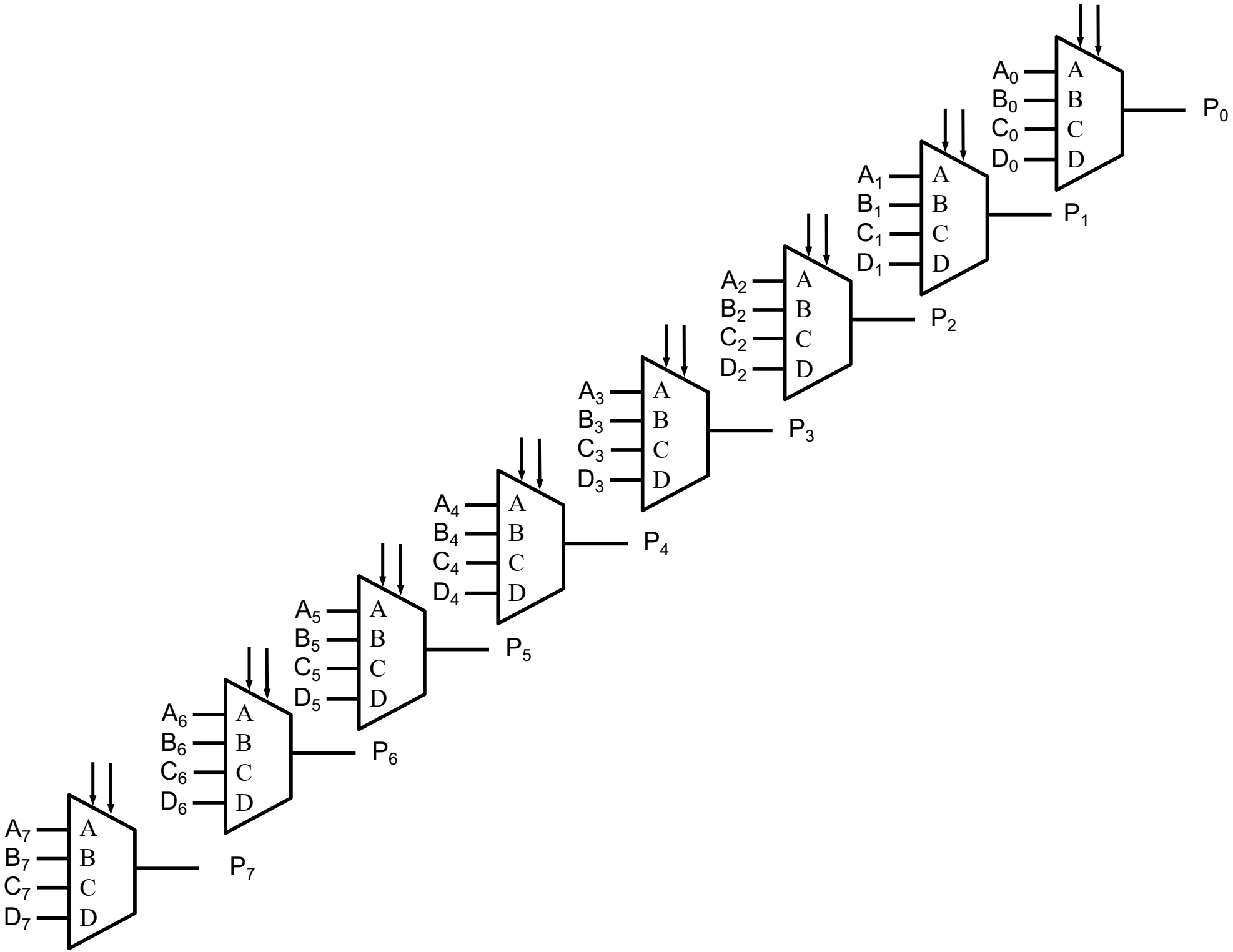
4-to-1 Bus Multiplexer (with 8-bit lines)

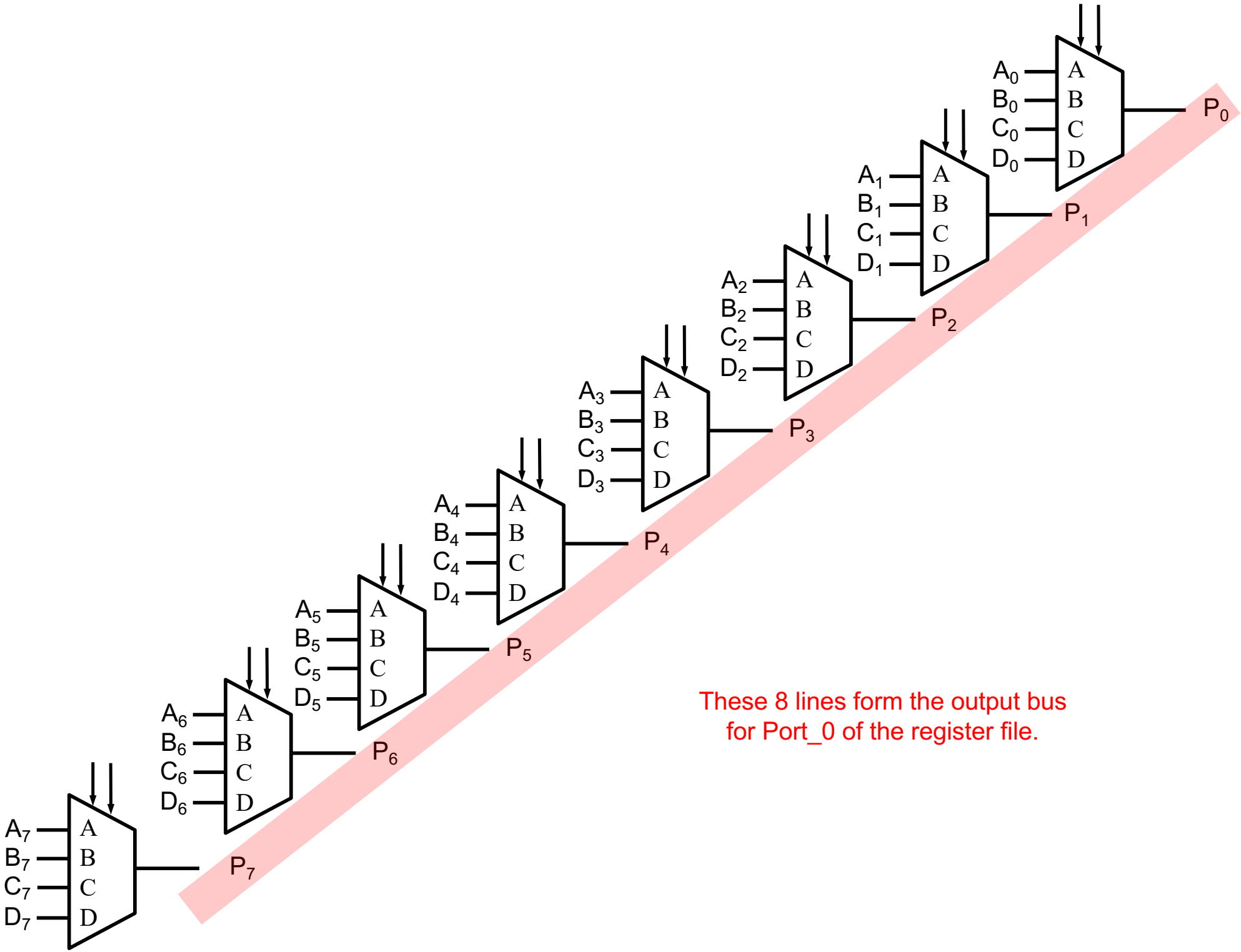


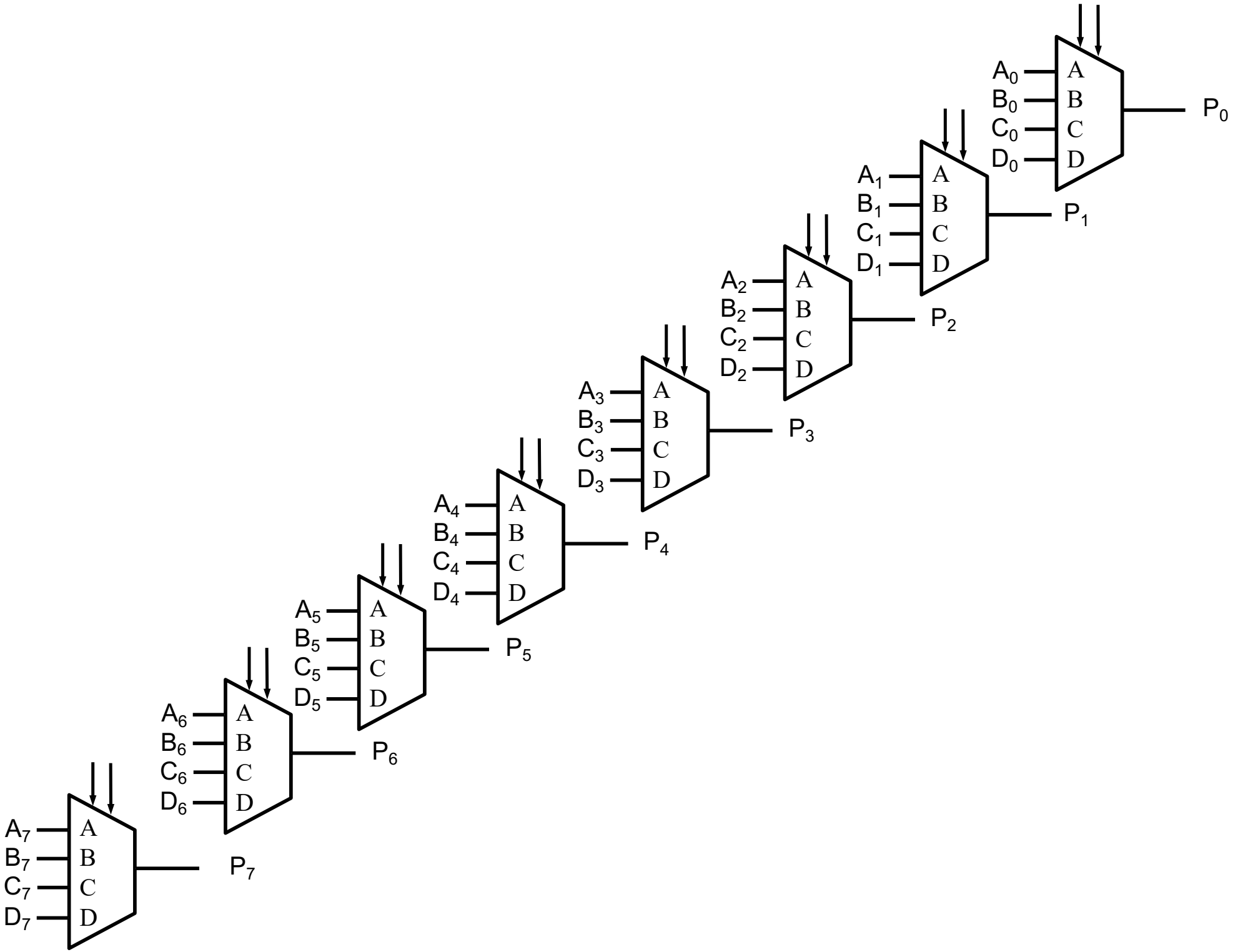

A0
B0
C0
D0
A
B
C
D
P0

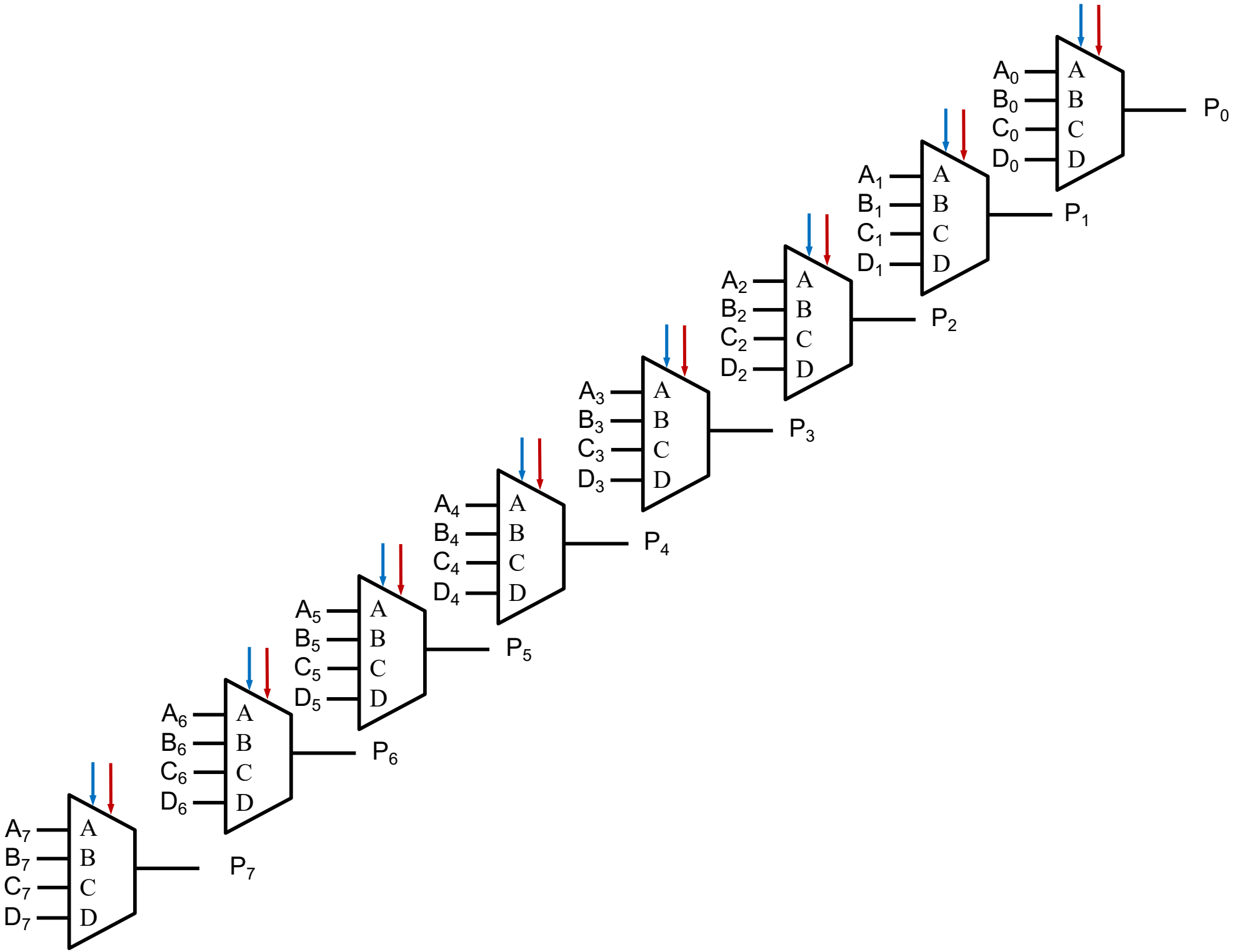

A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
A
B
C
D
P1

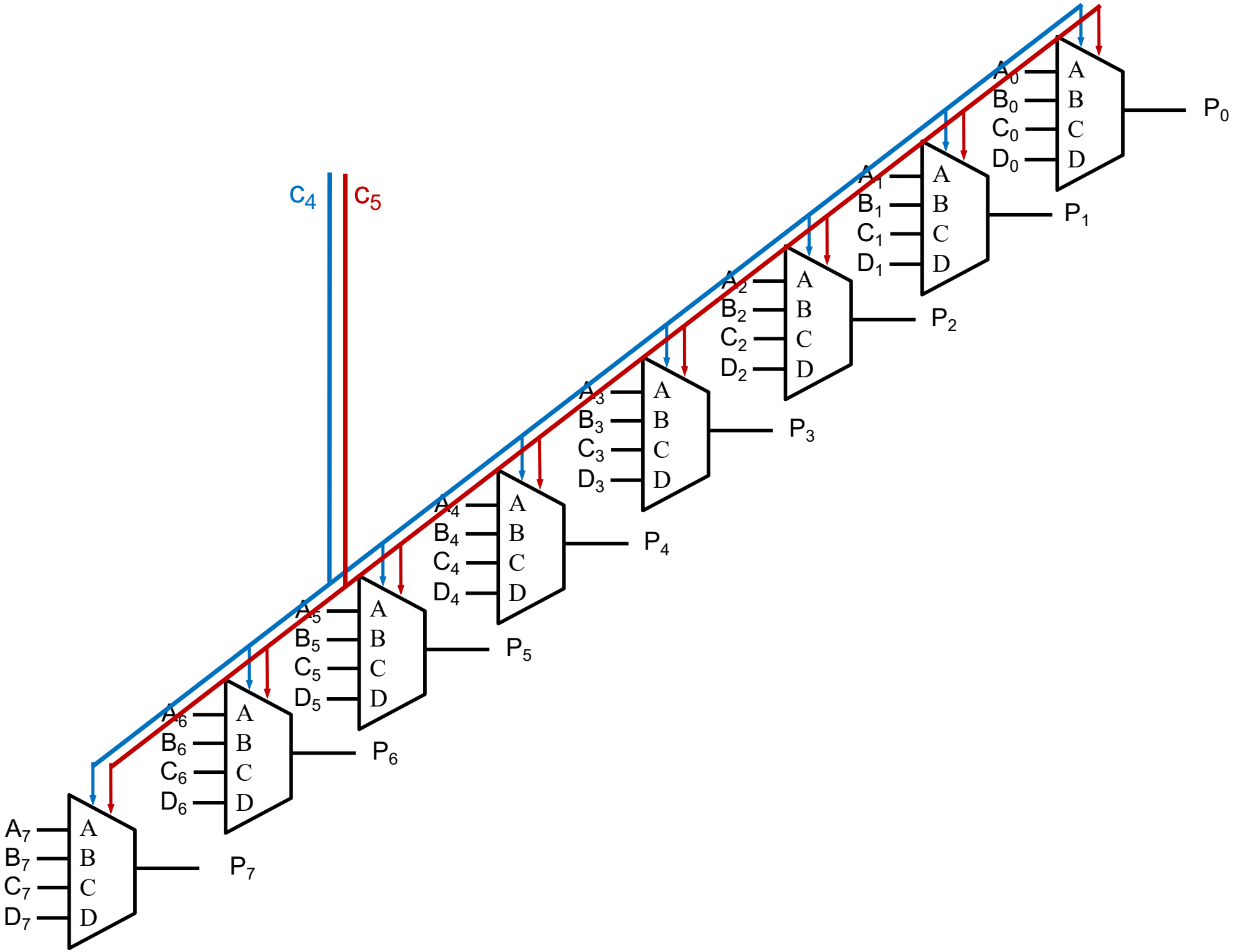

A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
A
B
C
D
P1
A2
B2
C2
D2
A
B
C
D
P2

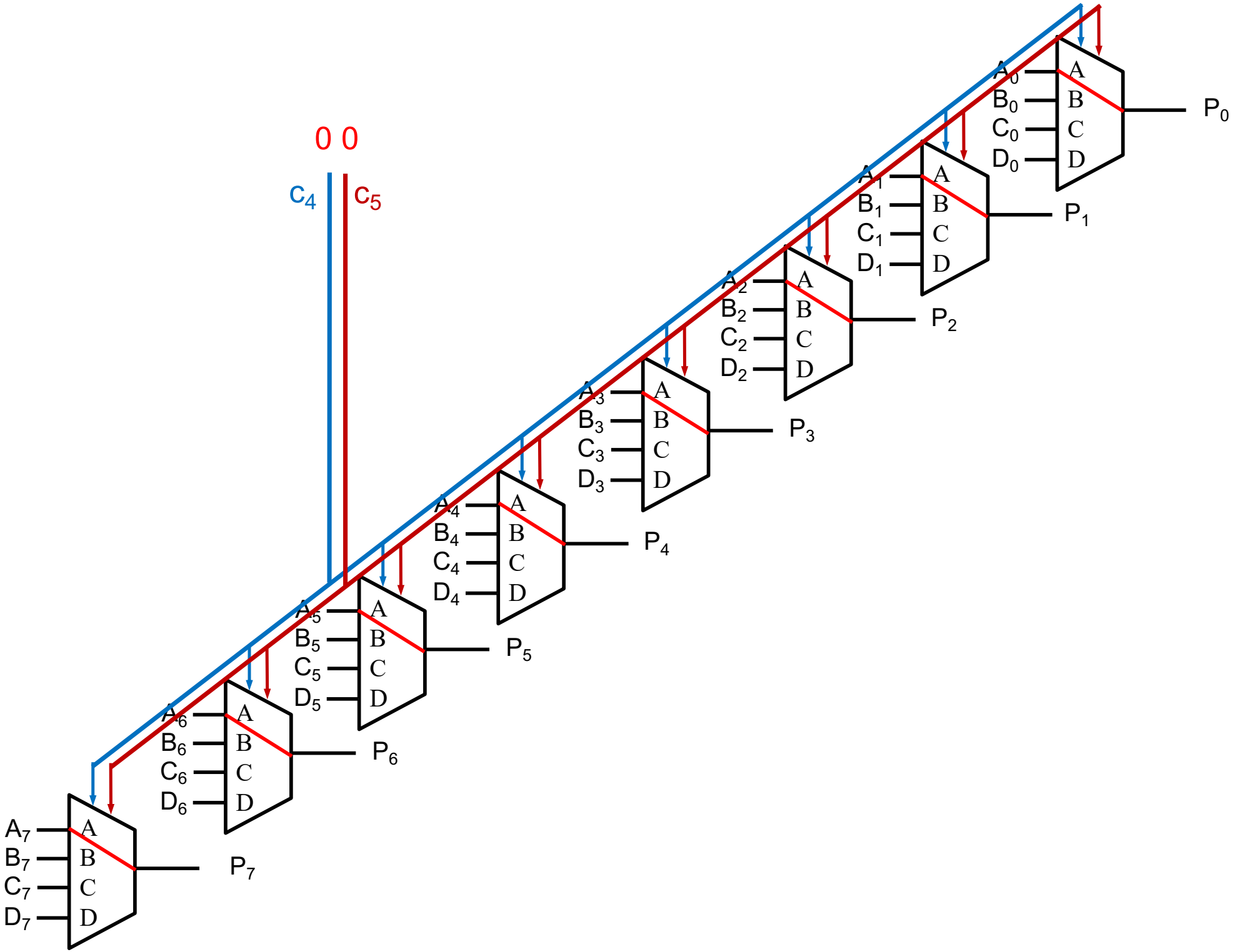

A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
P1
A2
B2
C2
D2
P2
A3
B3
C3
D3
P3
A4
B4
C4
D4
P4
A5
B5
C5
D5
P5
A6
B6
C6
D6
P6
A7
B7
C7
D7
P7

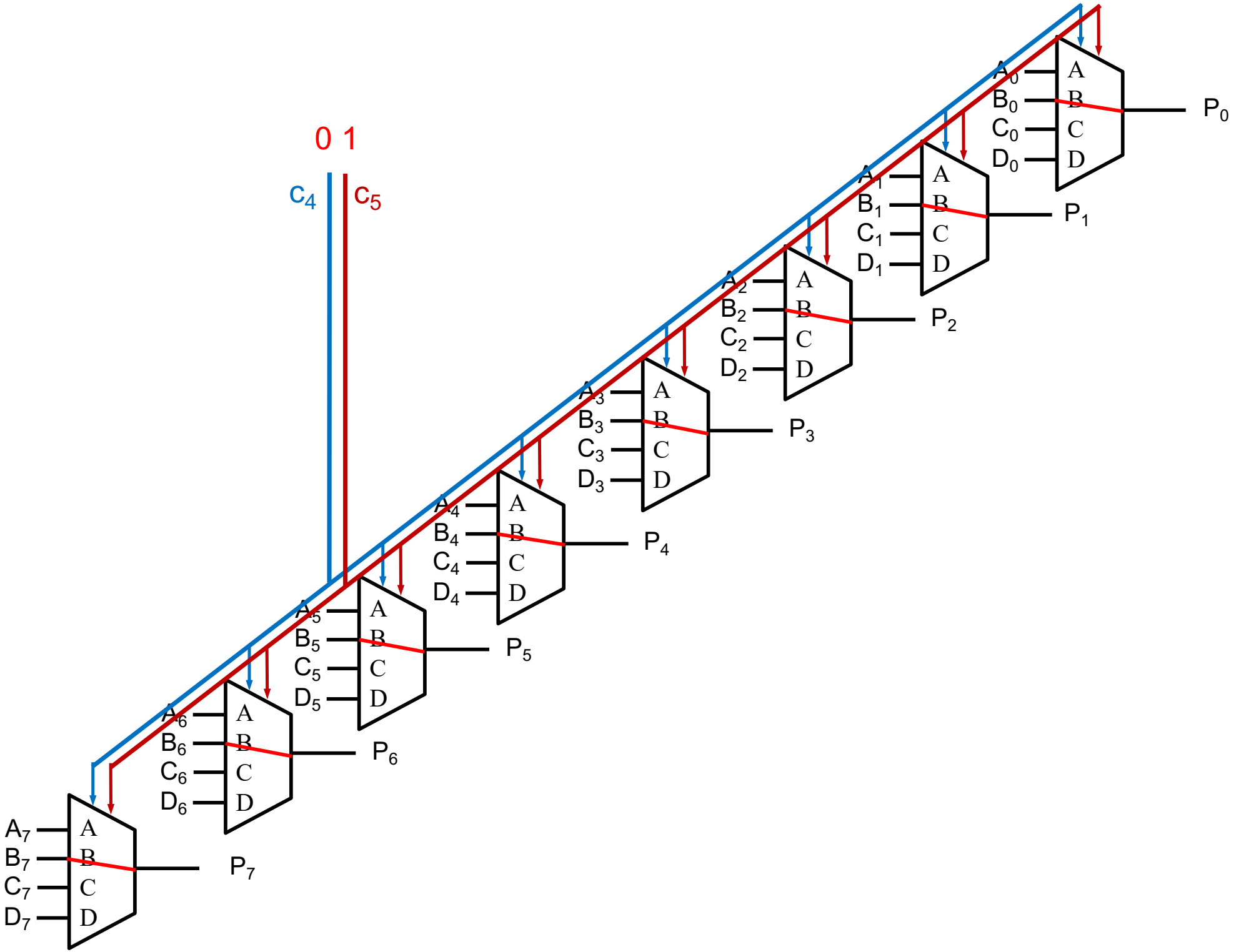

A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
P1
A2
B2
C2
D2
P2
A3
B3
C3
D3
P3
A4
B4
C4
D4
P4
A5
B5
C5
D5
P5
A6
B6
C6
D6
P6
A7
B7
C7
D7
P7
These 8 lines form the output bus
for Port_0 of the register file.

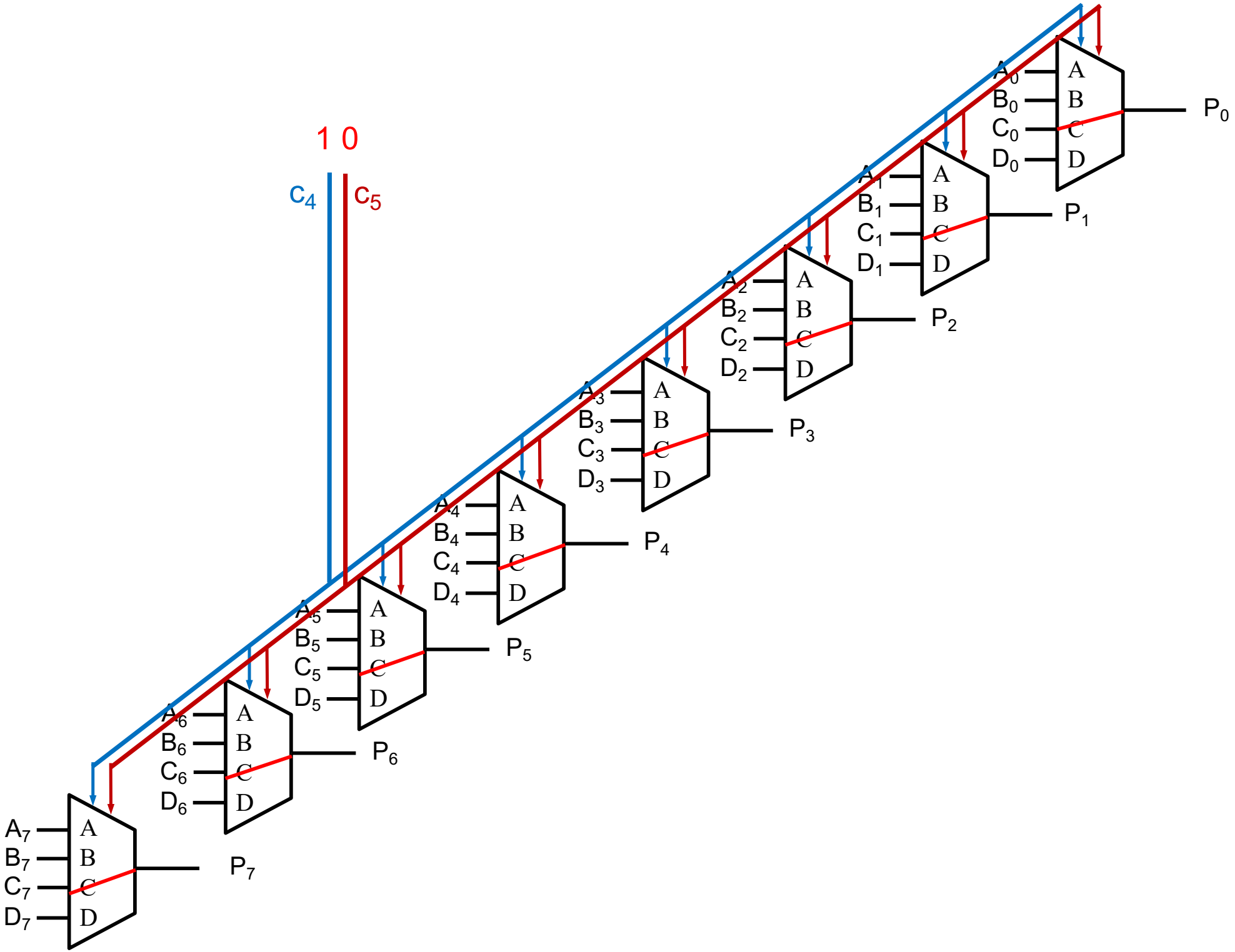

A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
P1
A2
B2
C2
D2
P2
A3
B3
C3
D3
P3
A4
B4
C4
D4
P4
A5
B5
C5
D5
P5
A6
B6
C6
D6
P6
A7
B7
C7
D7
P7

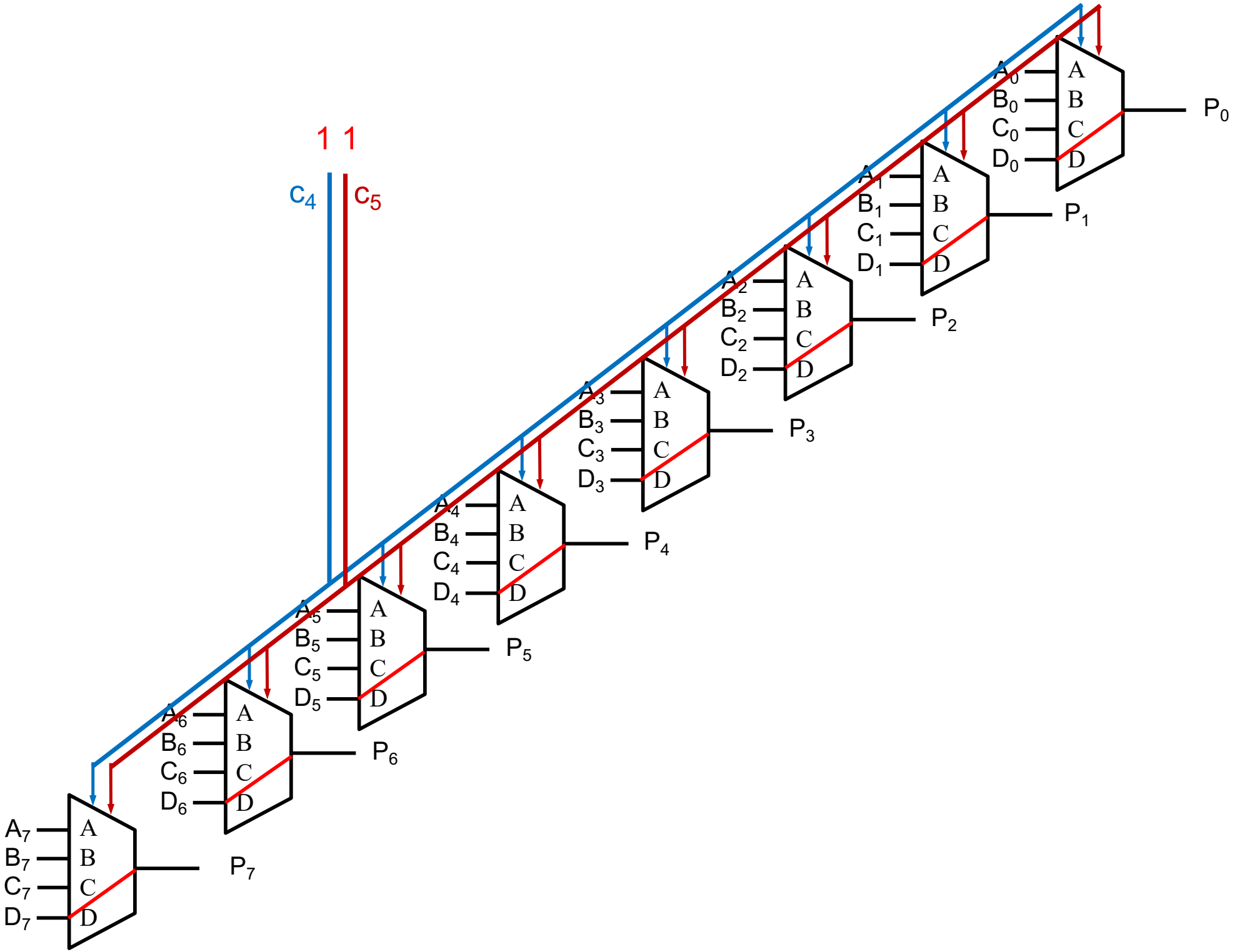

A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
P1
A2
B2
C2
D2
P2
A3
B3
C3
D3
P3
A4
B4
C4
D4
P4
A5
B5
C5
D5
P5
A6
B6
C6
D6
P6
A7
B7
C7
D7
P7

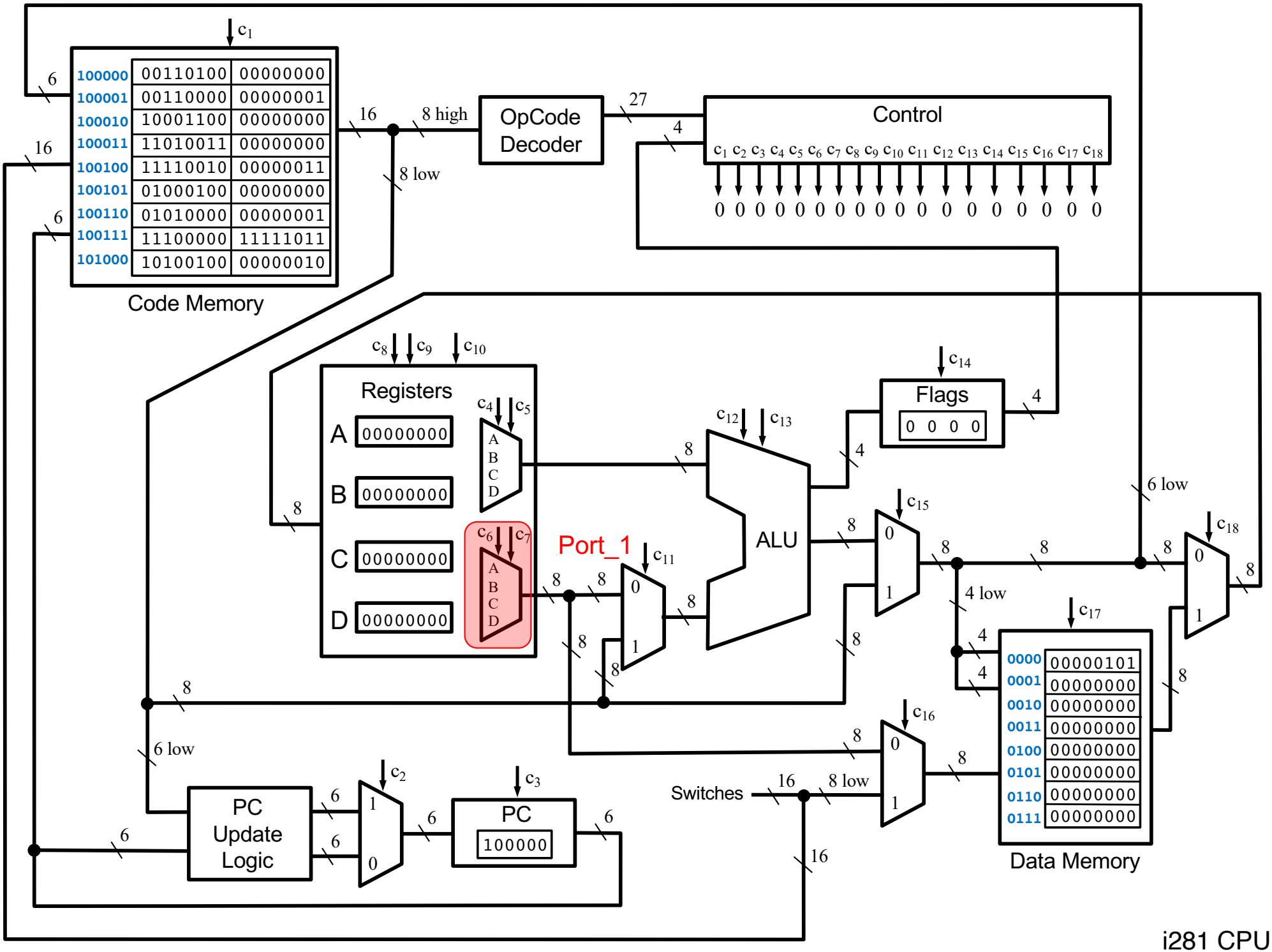

C4
C5
A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
P1
A2
B2
C2
D2
P2
A3
B3
C3
D3
P3
A4
B4
C4
D4
P4
A5
B5
C5
D5
P5
A6
B6
C6
D6
P6
A7
B7
C7
D7
P7


0 0
C_4
C_5
A_0 B_0 C_0 D_0 A B C D P_0
A_1 B_1 C_1 D_1 A B C D P_1
A_2 B_2 C_2 D_2 A B C D P_2
A_3 B_3 C_3 D_3 A B C D P_3
A_4 B_4 C_4 D_4 A B C D P_4
A_5 B_5 C_5 D_5 A B C D P_5
A_6 B_6 C_6 D_6 A B C D P_6
A_7 B_7 C_7 D_7 A B C D P_7

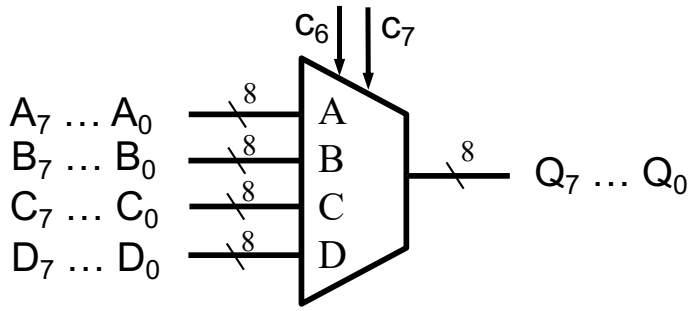

0 1
C_4
C_5
A_0 B_0 C_0 D_0 A B C D P_0
A_1 B_1 C_1 D_1 A B C D P_1
A_2 B_2 C_2 D_2 A B C D P_2
A_3 B_3 C_3 D_3 A B C D P_3
A_4 B_4 C_4 D_4 A B C D P_4
A_5 B_5 C_5 D_5 A B C D P_5
A_6 B_6 C_6 D_6 A B C D P_6
A_7 B_7 C_7 D_7 A B C D P_7

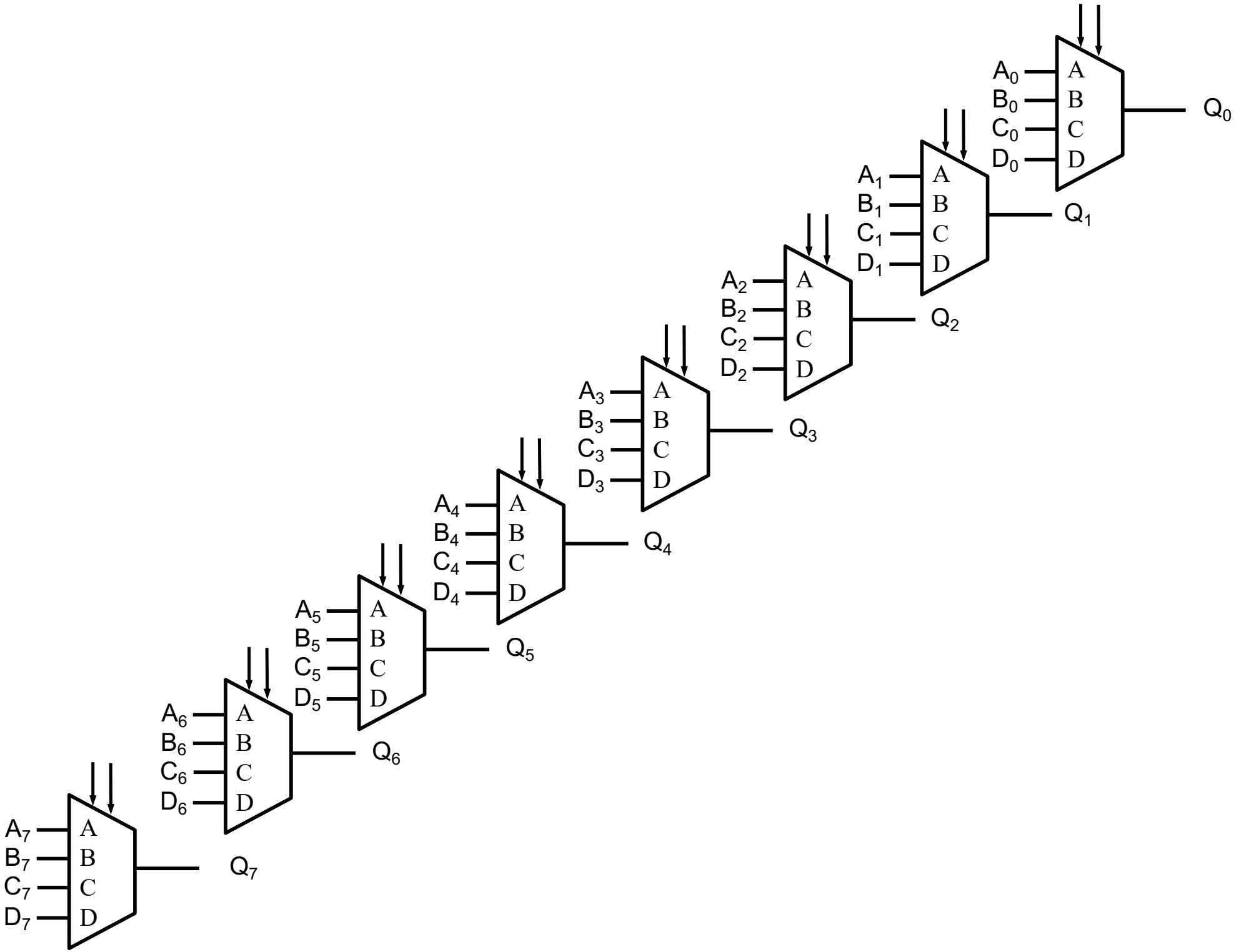

1 0
C4
C5
A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
P1
A2
B2
C2
D2
P2
A3
B3
C3
D3
P3
A4
B4
C4
D4
P4
A5
B5
C5
D5
P5
A6
B6
C6
D6
P6
A7
B7
C7
D7
P7

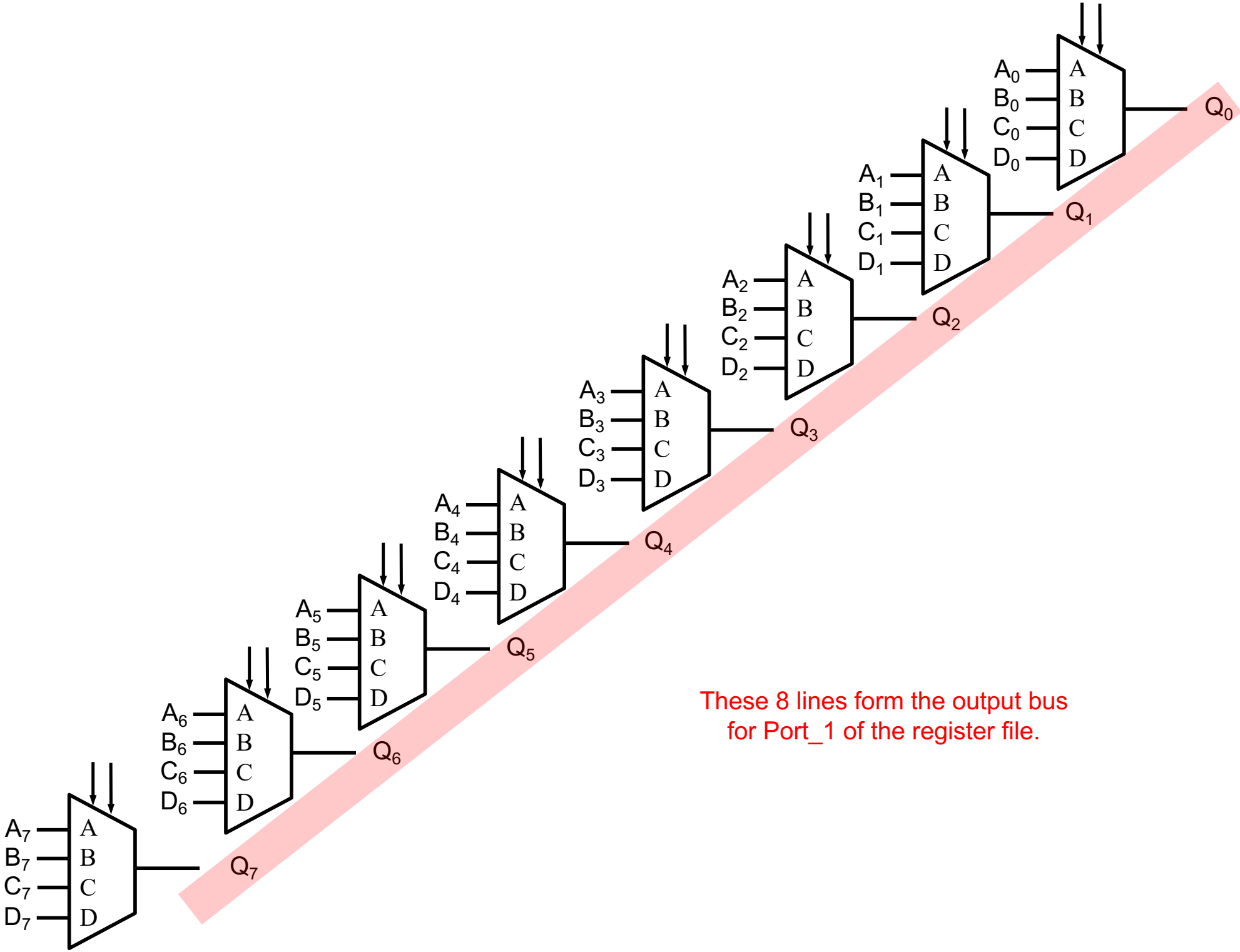

1 1
C4
C5
A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
P1
A2
B2
C2
D2
P2
A3
B3
C3
D3
P3
A4
B4
C4
D4
P4
A5
B5
C5
D5
P5
A6
B6
C6
D6
P6
A7
B7
C7
D7
P7

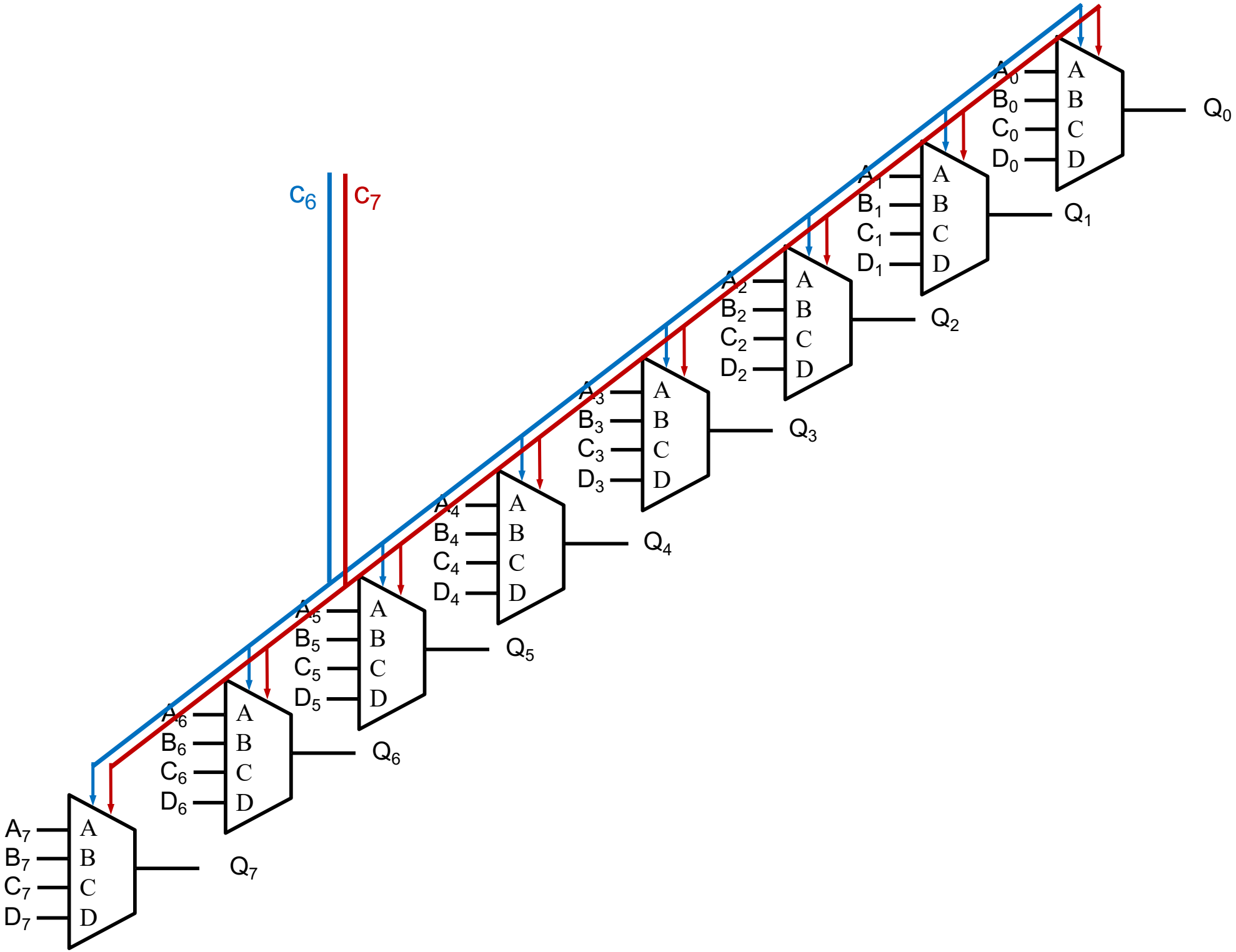

c1
6
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
c8 c9 c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
c4 c5
c6 c7
A B C D
8
c12 c13
ALU
c14
Flags
0 0 0 0
4
Port_1
c11
c15
c18
6 low
4 low
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
8 low
c2
c3
PC Update Logic
PC
100000
i281 CPU

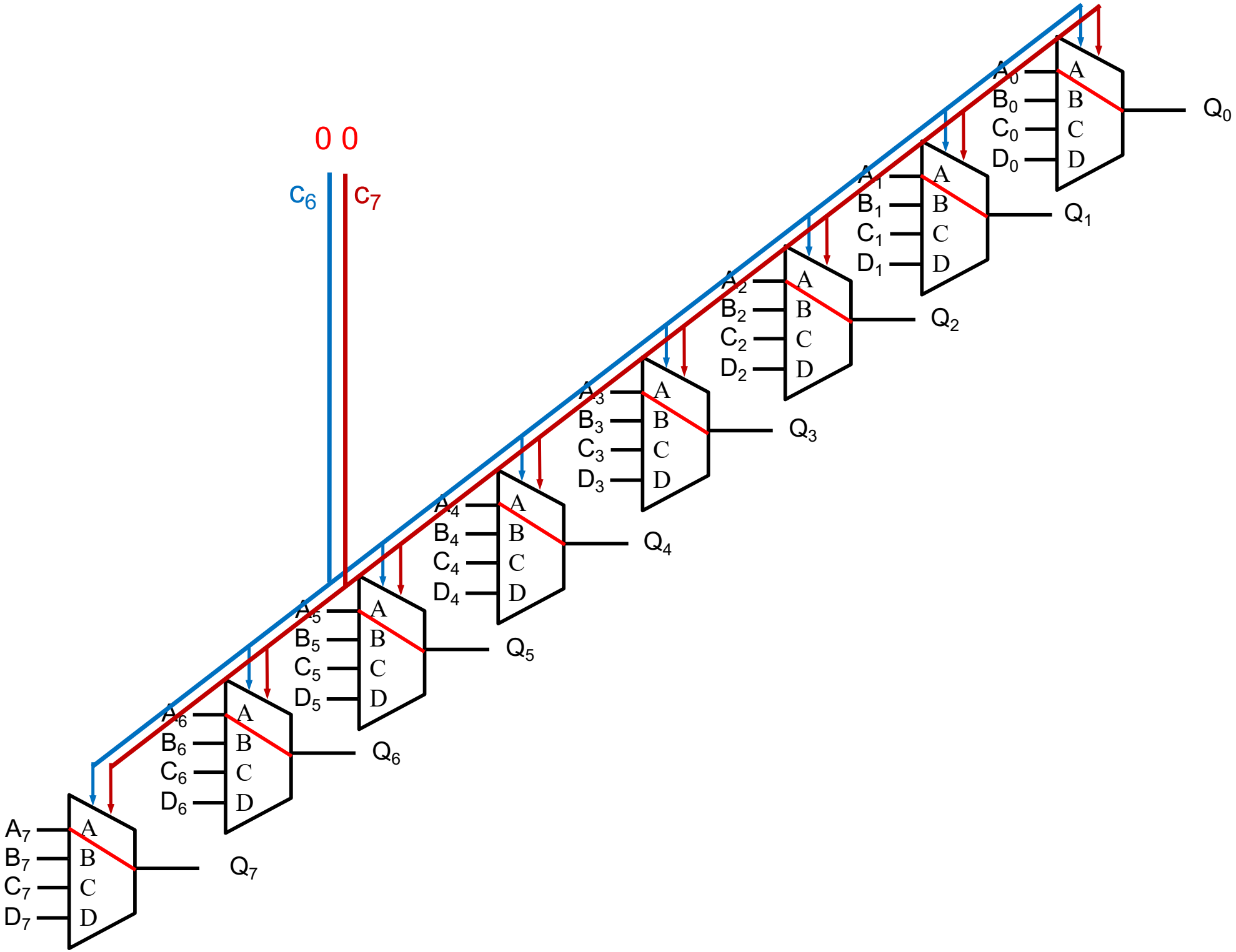
4-to-1 Bus Multiplexer (with 8-bit lines)

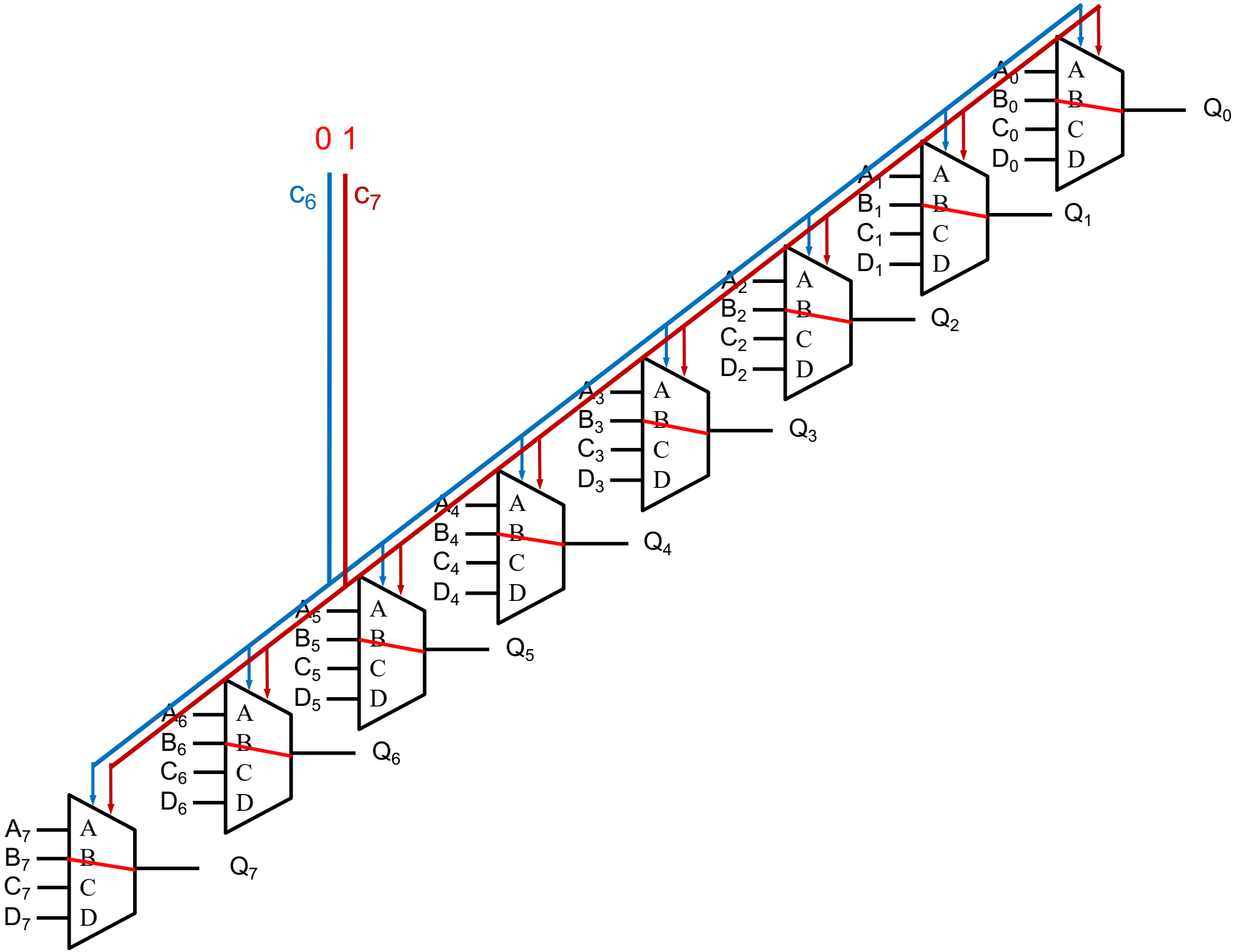


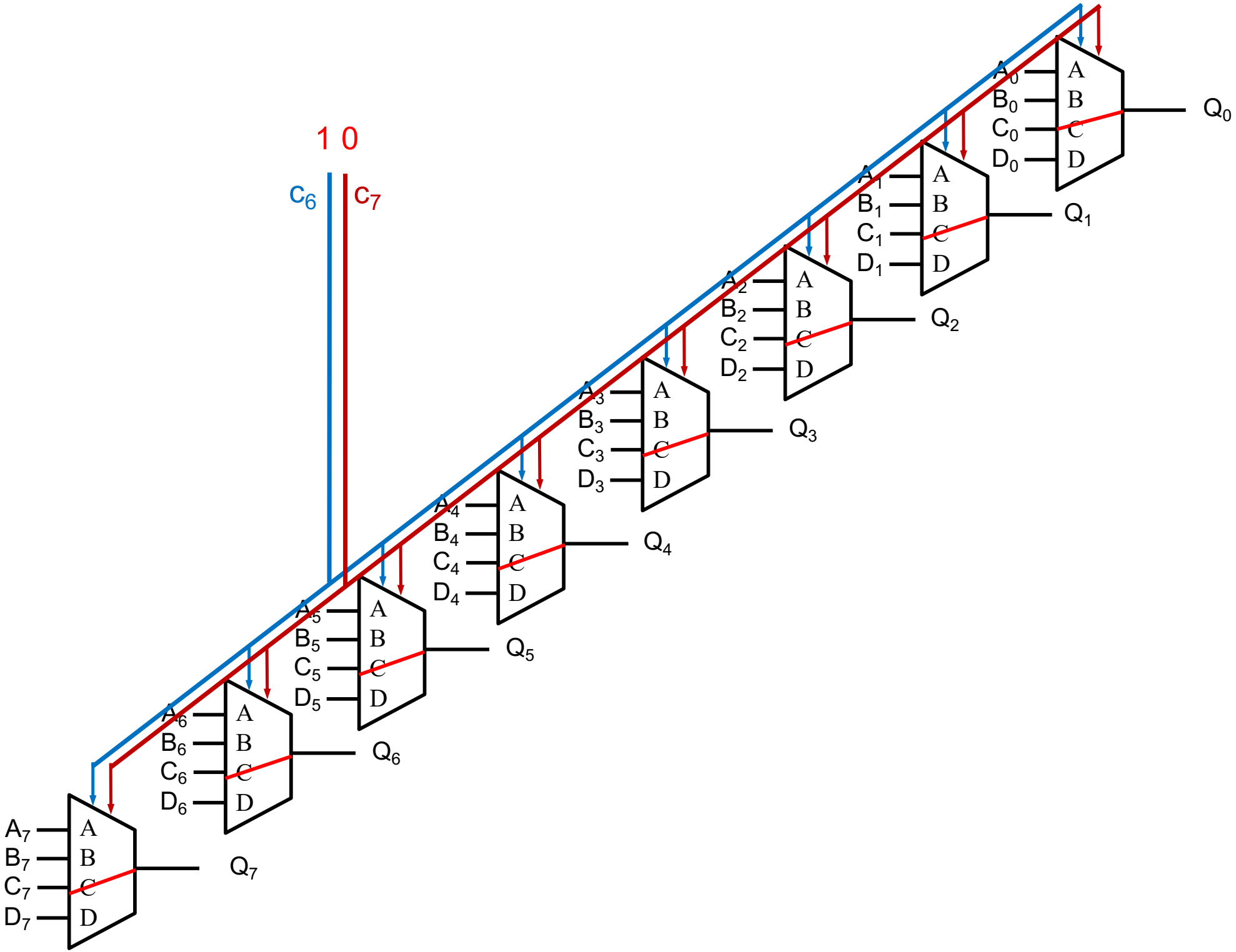

A0
B0
C0
D0
A
B
C
D
Q0
A1
B1
C1
D1
Q1
A2
B2
C2
D2
Q2
A3
B3
C3
D3
Q3
A4
B4
C4
D4
Q4
A5
B5
C5
D5
Q5
A6
B6
C6
D6
Q6
A7
B7
C7
D7
Q7

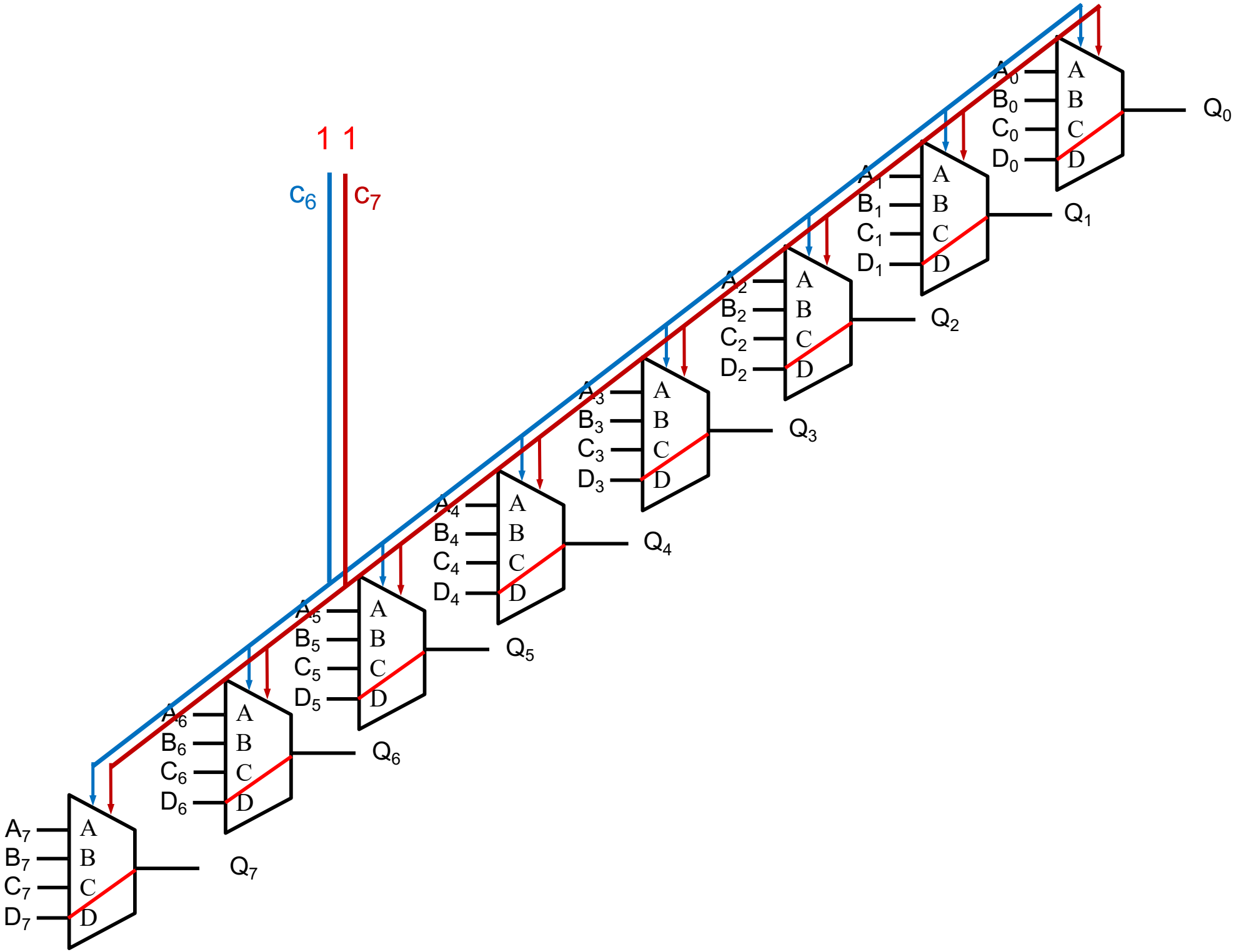

A0
B0
C0
D0
A
B
C
D
Q0
A1
B1
C1
D1
Q1
A2
B2
C2
D2
Q2
A3
B3
C3
D3
Q3
A4
B4
C4
D4
Q4
A5
B5
C5
D5
Q5
A6
B6
C6
D6
Q6
A7
B7
C7
D7
Q7
These 8 lines form the output bus
for Port_1 of the register file.

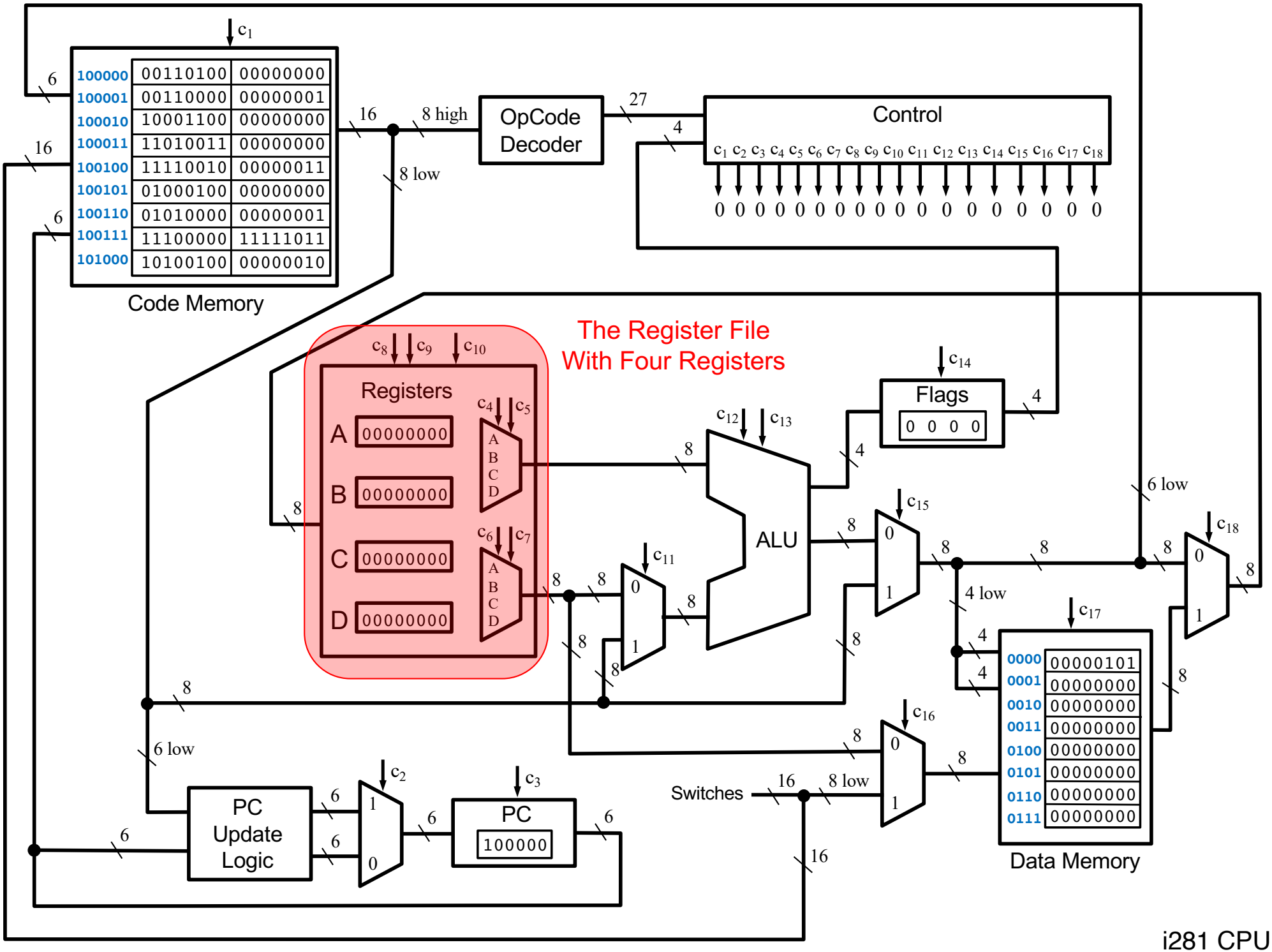

C6
C7
A0
B0
C0
D0
A
B
C
D
Q0
A1
B1
C1
D1
Q1
A2
B2
C2
D2
Q2
A3
B3
C3
D3
Q3
A4
B4
C4
D4
Q4
A5
B5
C5
D5
Q5
A6
B6
C6
D6
Q6
A7
B7
C7
D7
Q7


0 0
C6
C7
A0
B0
C0
D0
A
B
C
D
Q0
A1
B1
C1
D1
Q1
A2
B2
C2
D2
Q2
A3
B3
C3
D3
Q3
A4
B4
C4
D4
Q4
A5
B5
C5
D5
Q5
A6
B6
C6
D6
Q6
A7
B7
C7
D7
Q7


0 1
C6
C7
A7
B7
C7
D7
A
B
C
D
Q7
A6
B6
C6
D6
Q6
A5
B5
C5
D5
Q5
A4
B4
C4
D4
Q4
A3
B3
C3
D3
Q3
A2
B2
C2
D2
Q2
A1
B1
C1
D1
Q1
A0
B0
C0
D0
Q0

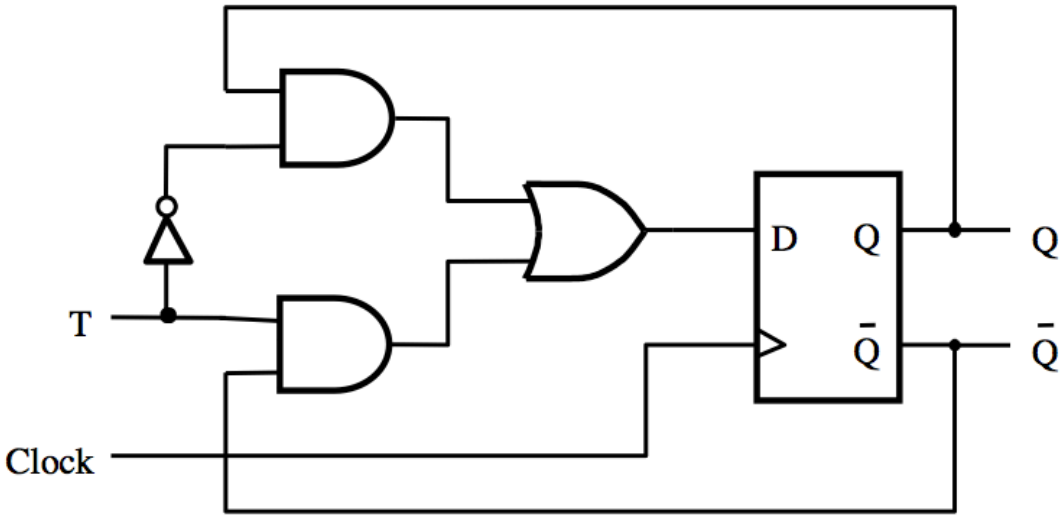

1 0
C6
C7
A7
B7
C7
D7
A
B
C
D
Q7
A6
B6
C6
D6
Q6
A5
B5
C5
D5
Q5
A4
B4
C4
D4
Q4
A3
B3
C3
D3
Q3
A2
B2
C2
D2
Q2
A1
B1
C1
D1
Q1
A0
B0
C0
D0
Q0

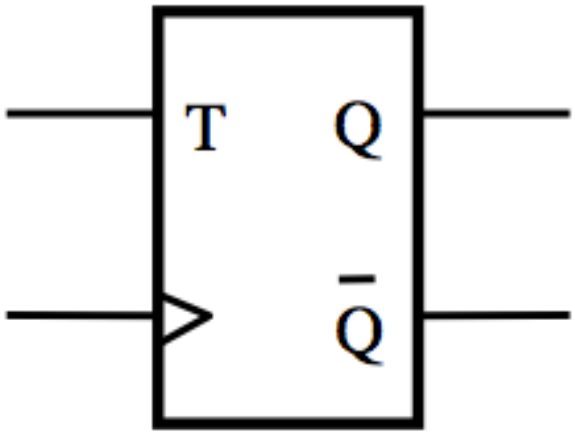

1 1
C6
C7
A7
B7
C7
D7
A
B
C
D
Q7
A6
B6
C6
D6
Q6
A5
B5
C5
D5
Q5
A4
B4
C4
D4
Q4
A3
B3
C3
D3
Q3
A2
B2
C2
D2
Q2
A1
B1
C1
D1
Q1
A0
B0
C0
D0
Q0


c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
6
16
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
The Register File
With Four Registers
c8 c9 c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
c4 c5
c6 c7
A B C D
c14
Flags
0 0 0 0
4
c12 c13
ALU
c11
c15
c16
c17
c18
6 low
4 low
8 low
8 high
Switches
PC Update Logic
c2
c3
PC
100000
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
i281 CPU

Counters

T Flip-Flop
(circuit and graphical symbol)

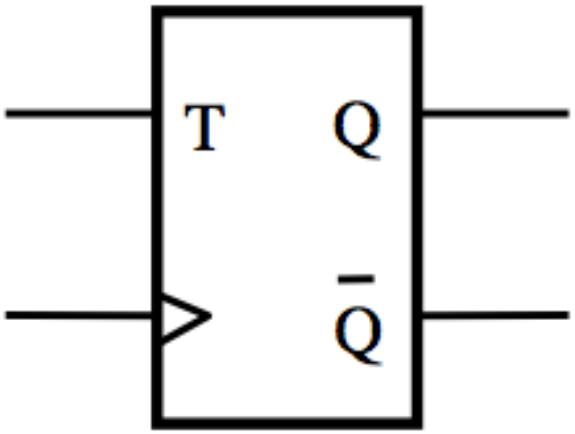




[Figure 5.15a,c from the textbook]

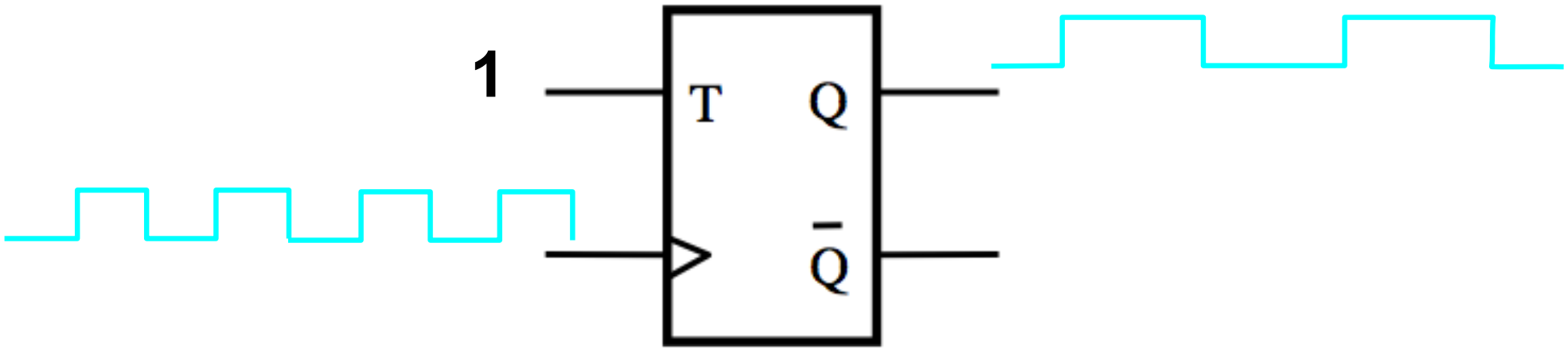
The output of the T Flip-Flop

divides the frequency of the clock by 2

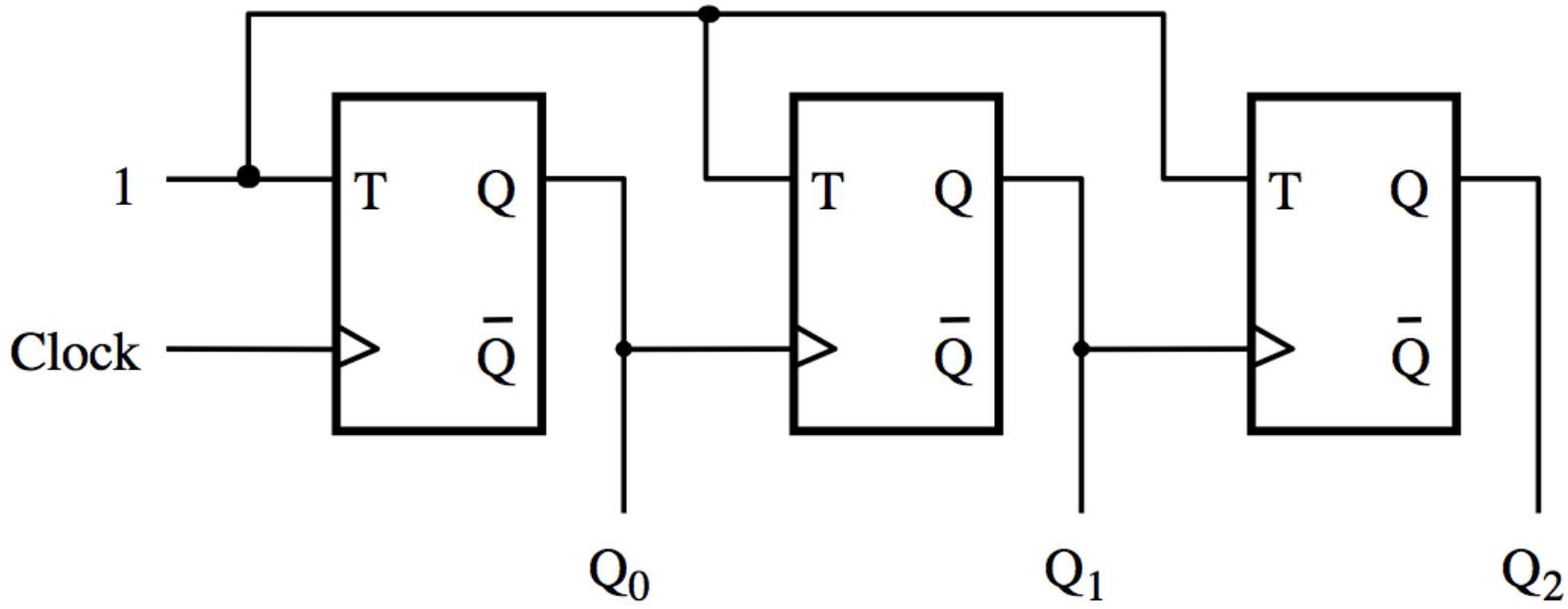


The output of the T Flip-Flop

divides the frequency of the clock by 2

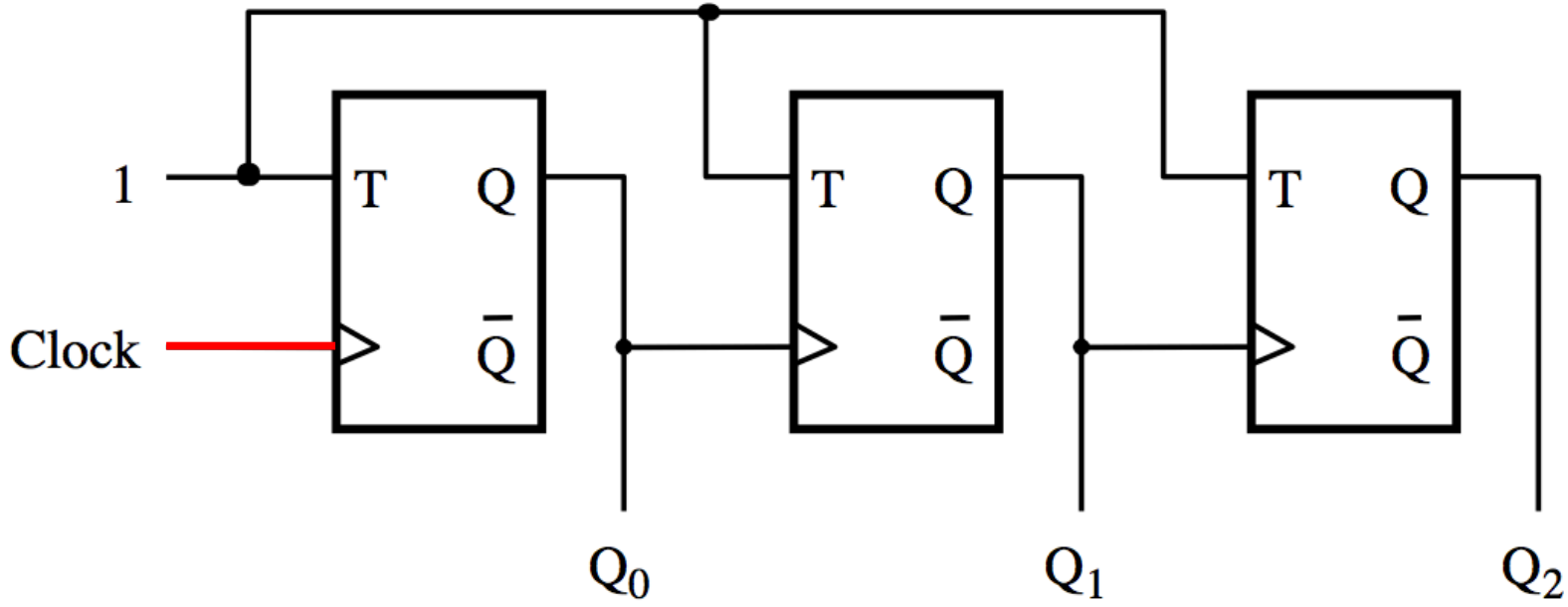


A three-bit down-counter



[Figure 5.20 from the textbook]

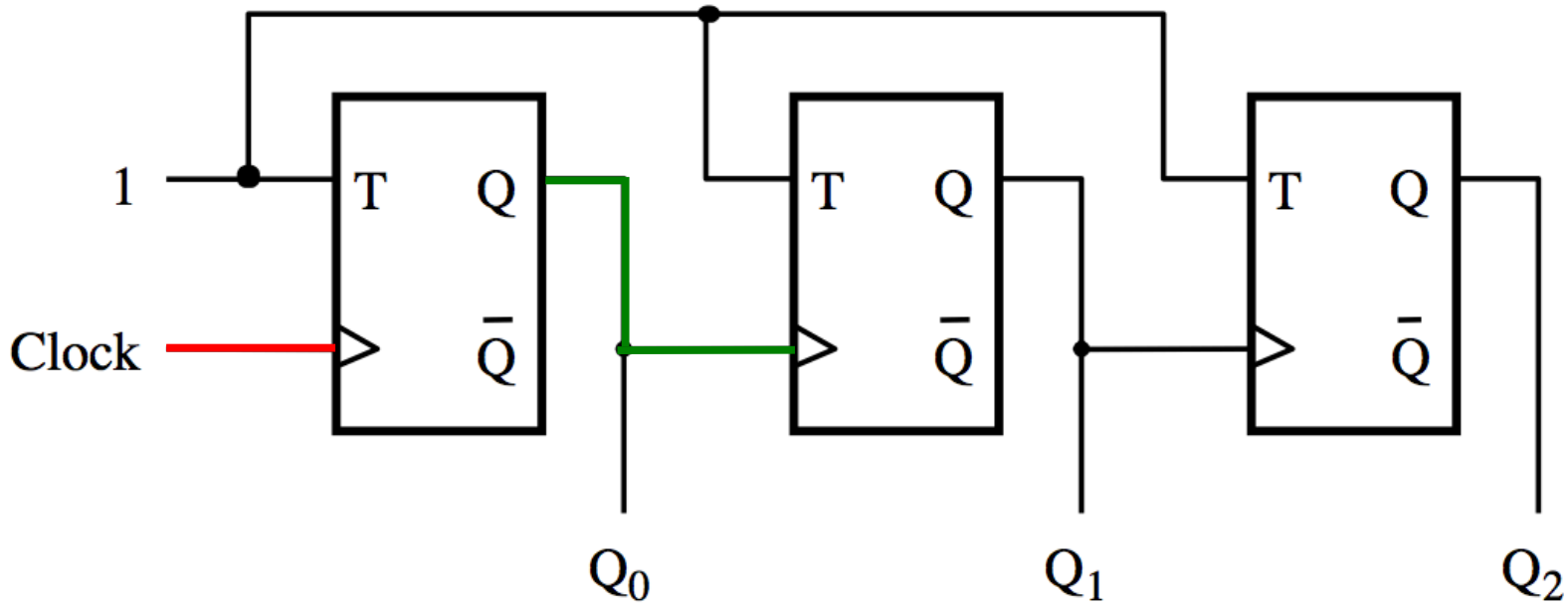
A three-bit down-counter



The first flip-flop changes
on the positive edge of the clock

[Figure 5.20 from the textbook]

A three-bit down-counter

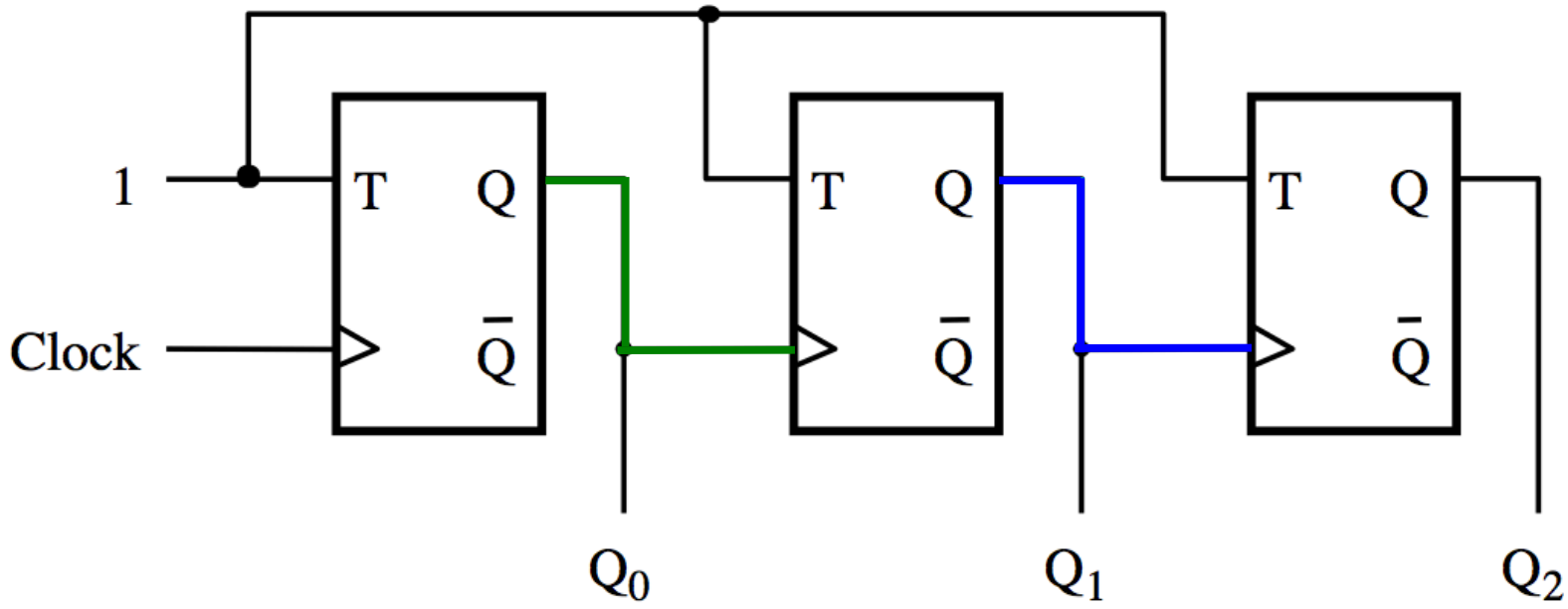


The first flip-flop changes on the positive edge of the clock

The second flip-flop changes on the positive edge of Q_0

[Figure 5.20 from the textbook]

A three-bit down-counter



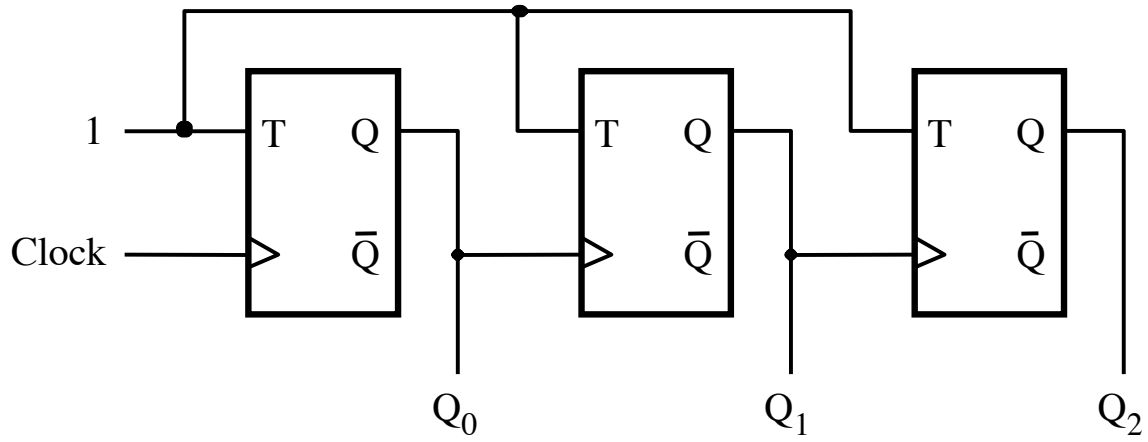
The first flip-flop changes on the positive edge of the clock

The second flip-flop changes on the positive edge of Q_0

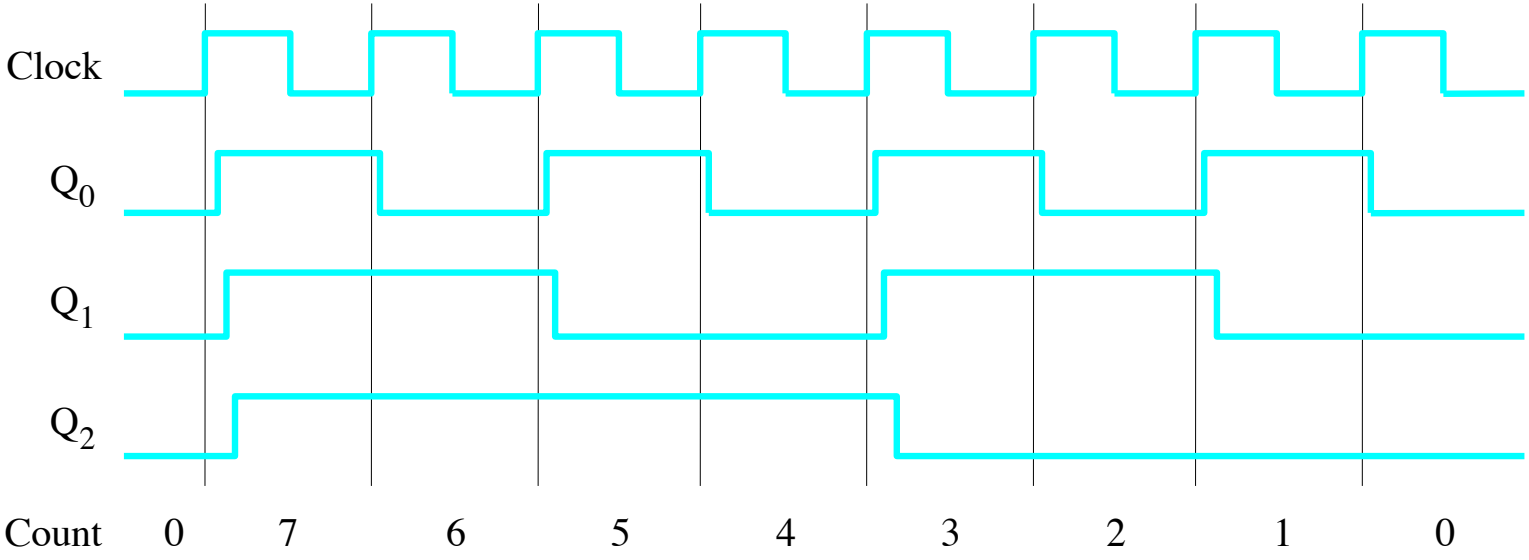
The third flip-flop changes on the positive edge of Q_1

[Figure 5.20 from the textbook]

A three-bit down-counter



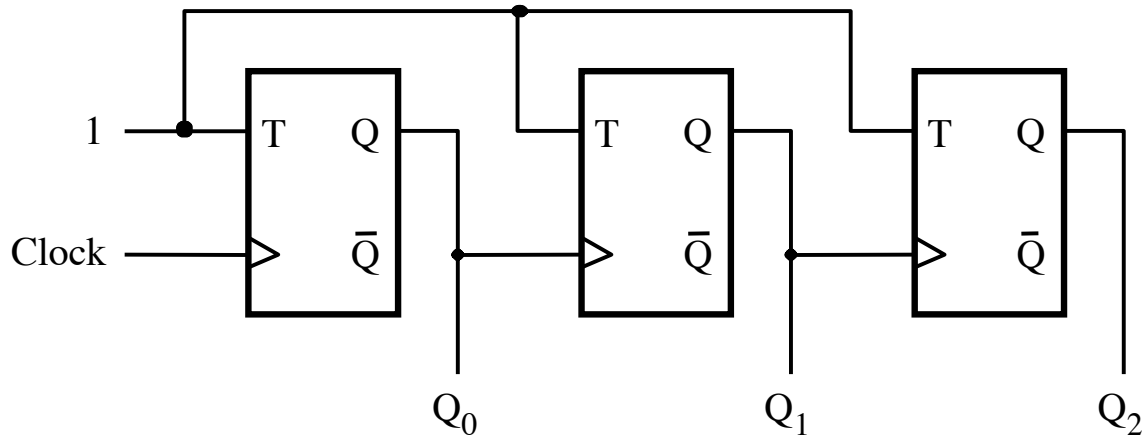
(a) Circuit



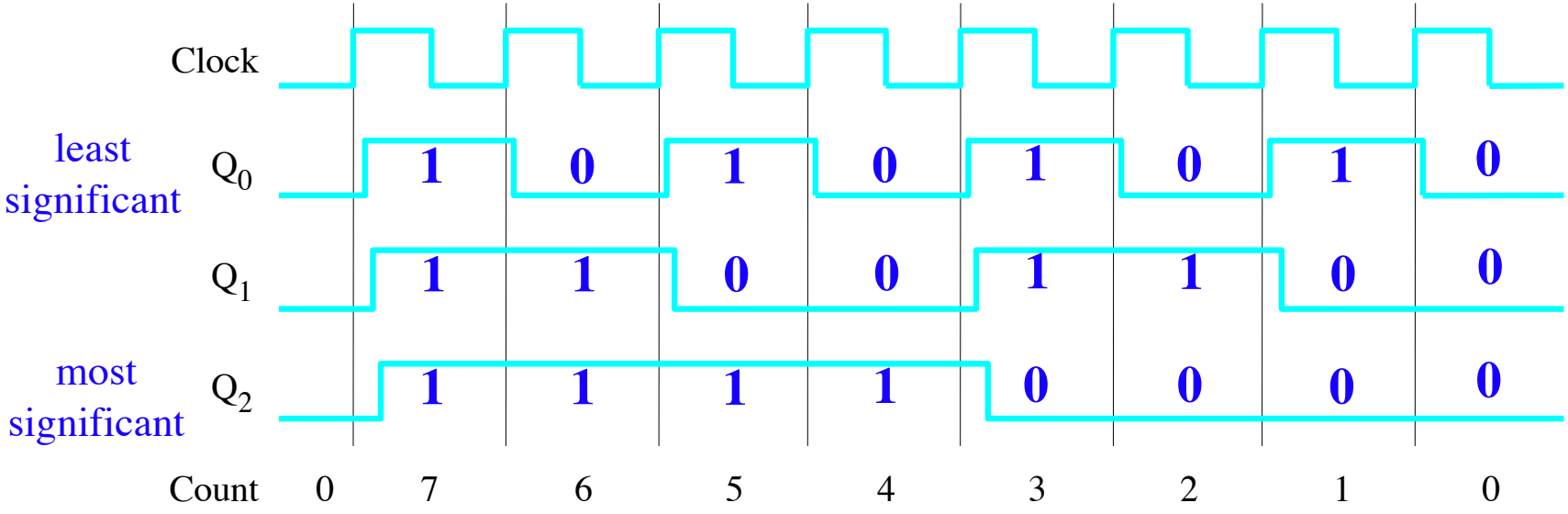
(b) Timing diagram

[Figure 5.20 from the textbook]

A three-bit down-counter

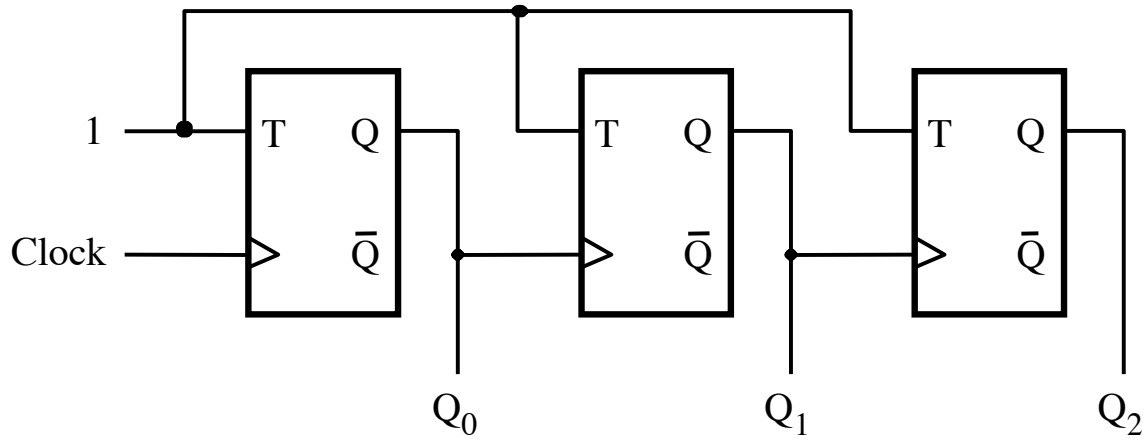


(a) Circuit

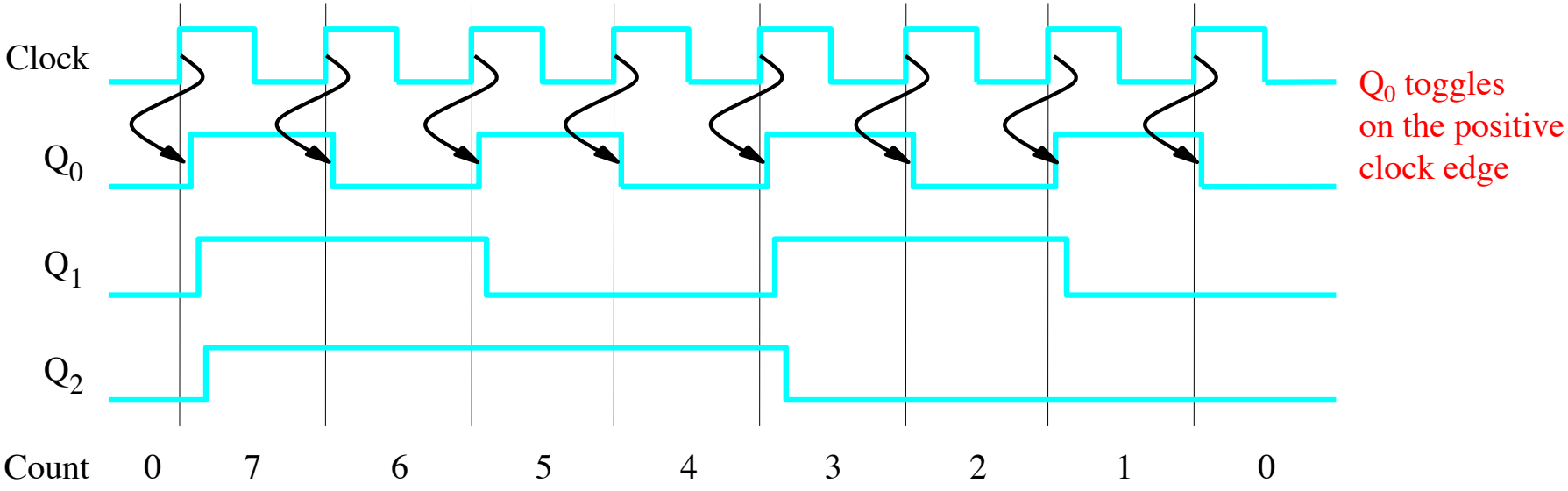


(b) Timing diagram

A three-bit down-counter

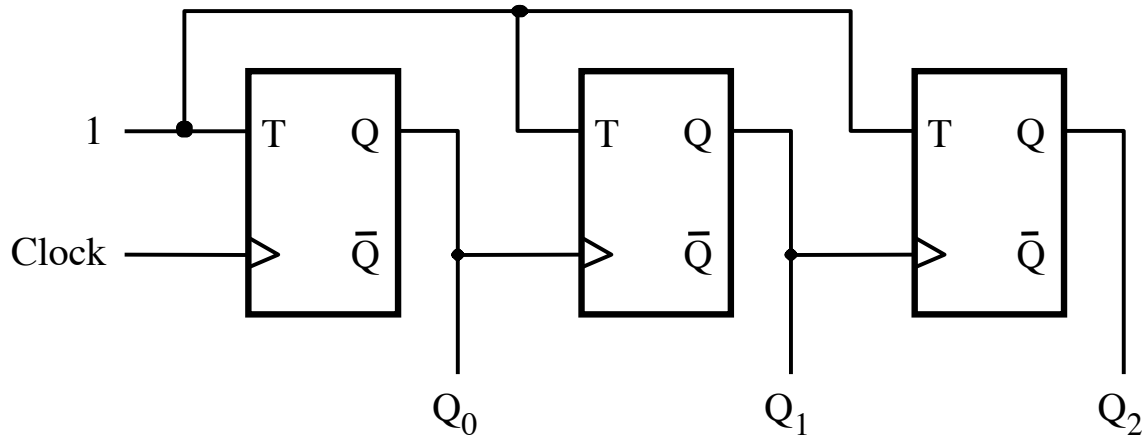


(a) Circuit

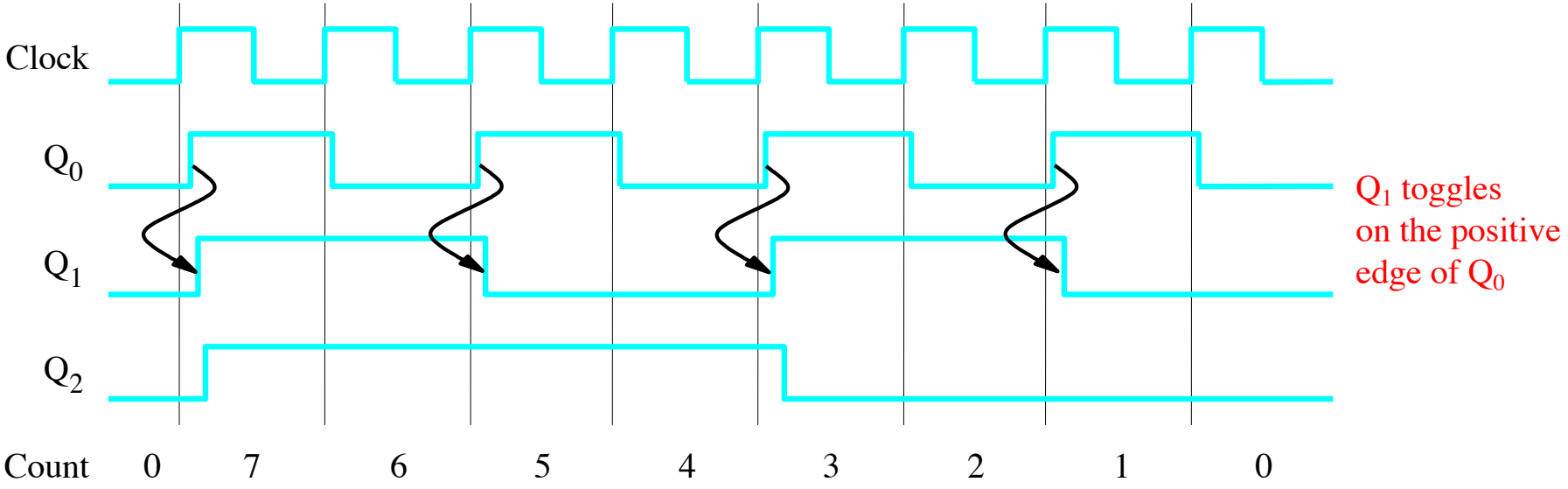


(b) Timing diagram

A three-bit down-counter

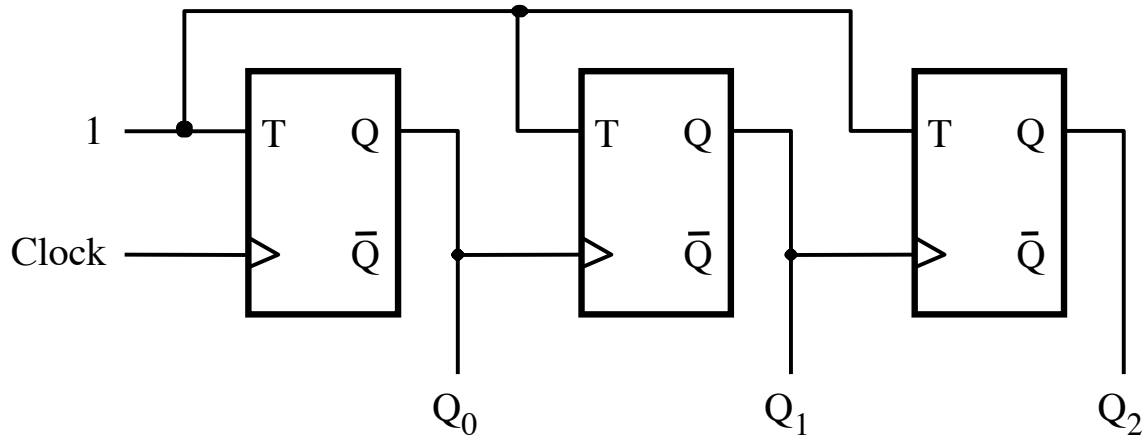


(a) Circuit

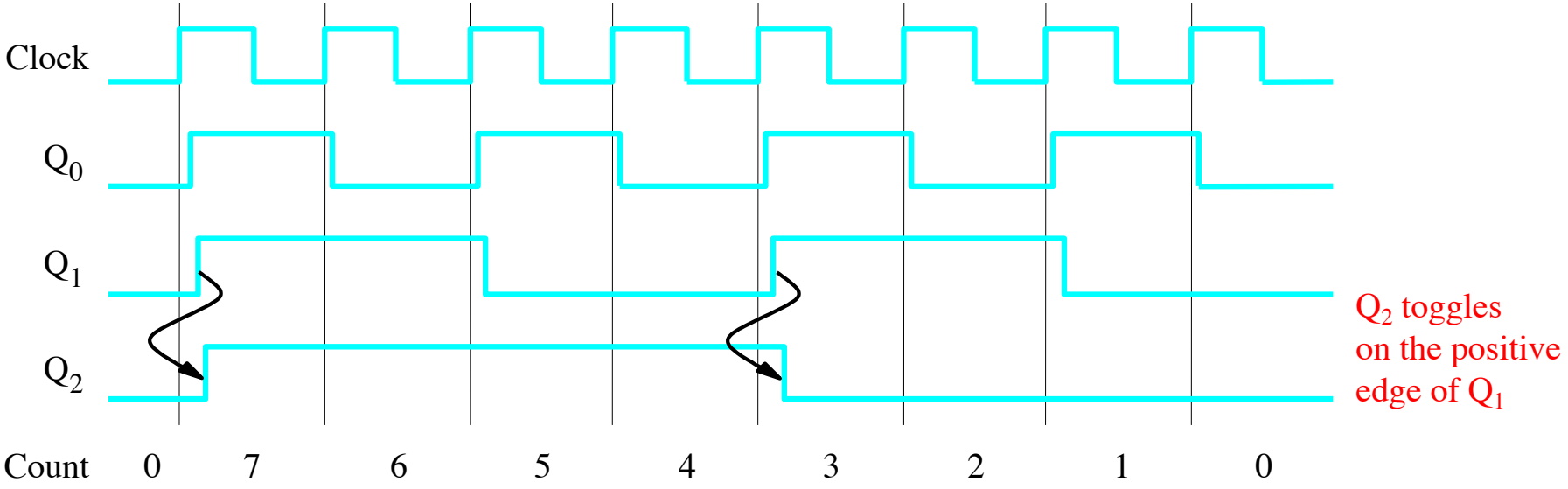


(b) Timing diagram

A three-bit down-counter

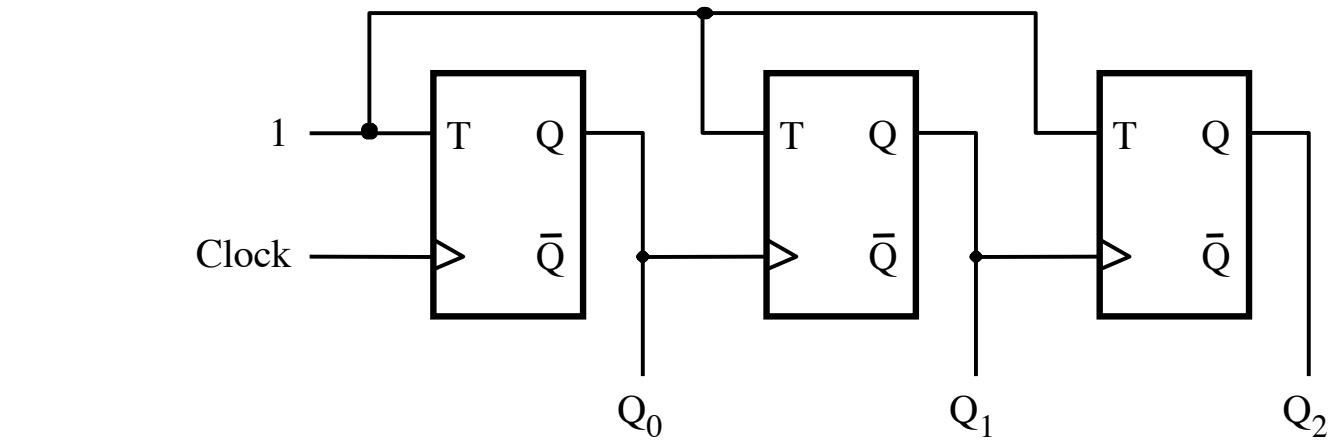


(a) Circuit

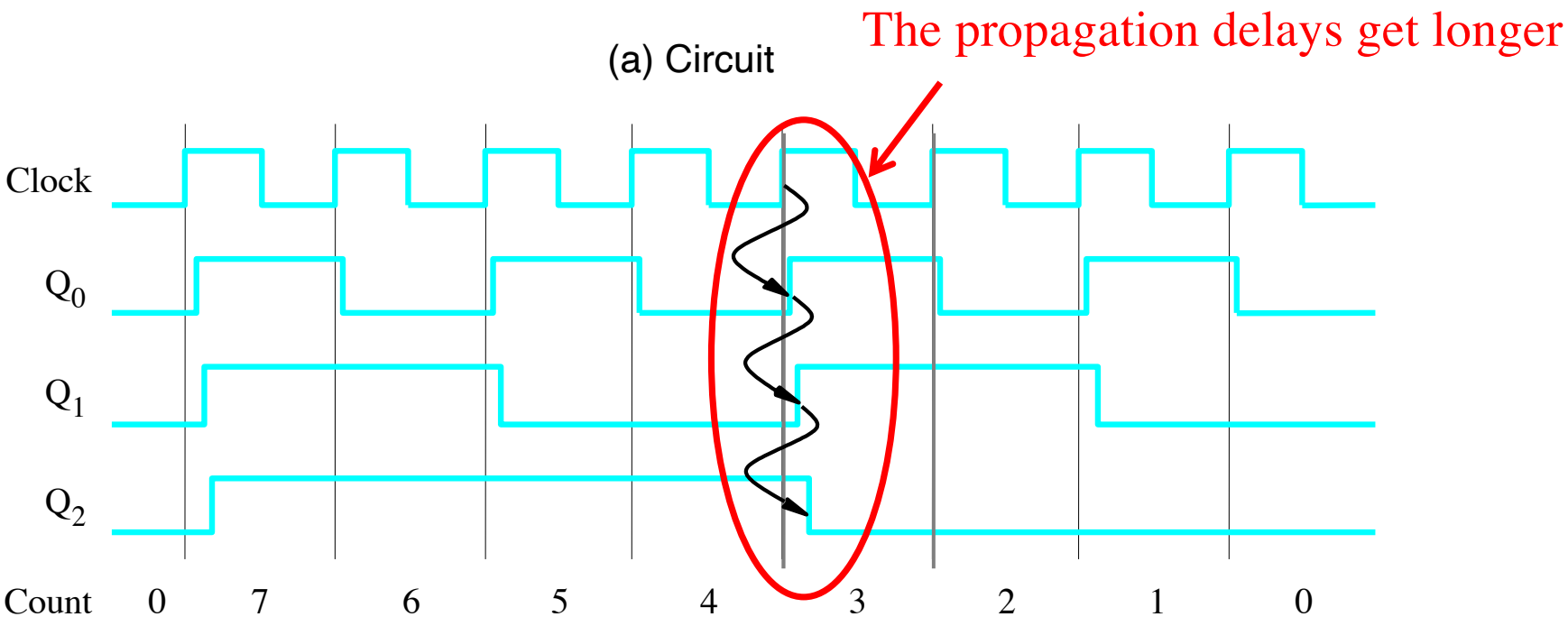


(b) Timing diagram

A three-bit down-counter

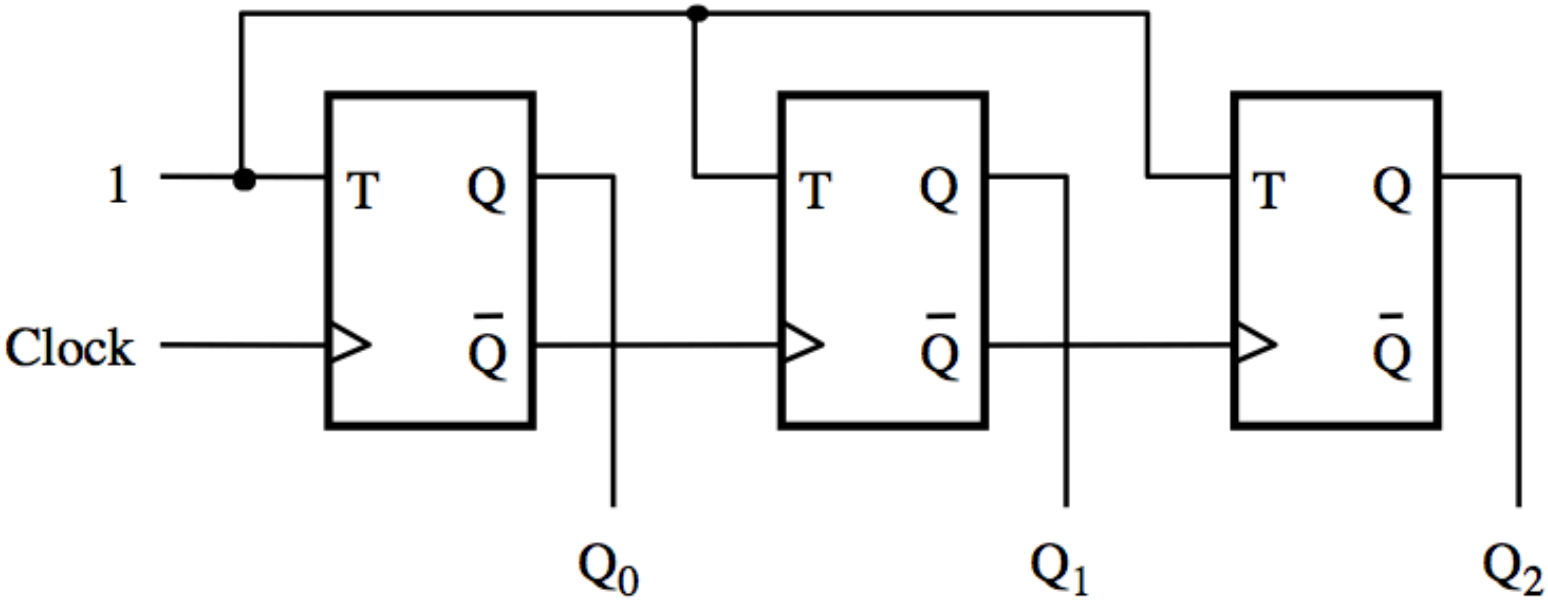


(a) Circuit



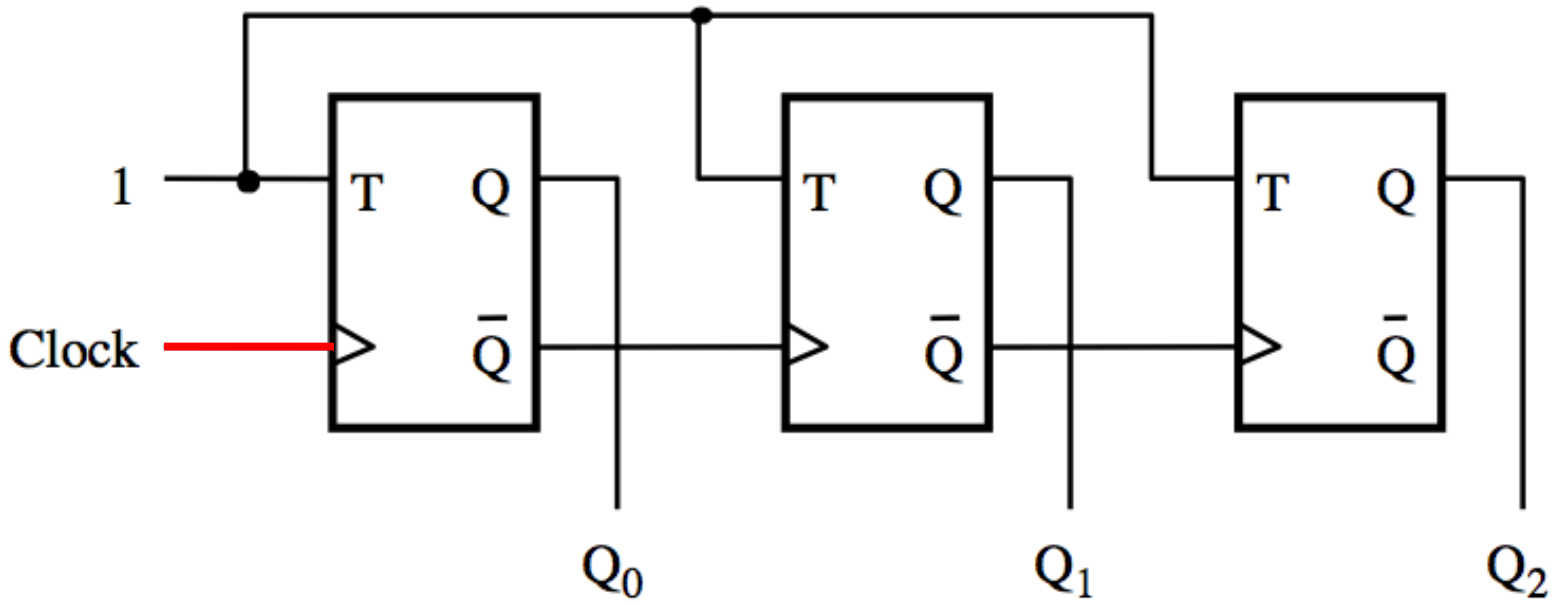
(b) Timing diagram

A three-bit up-counter



[Figure 5.19 from the textbook]

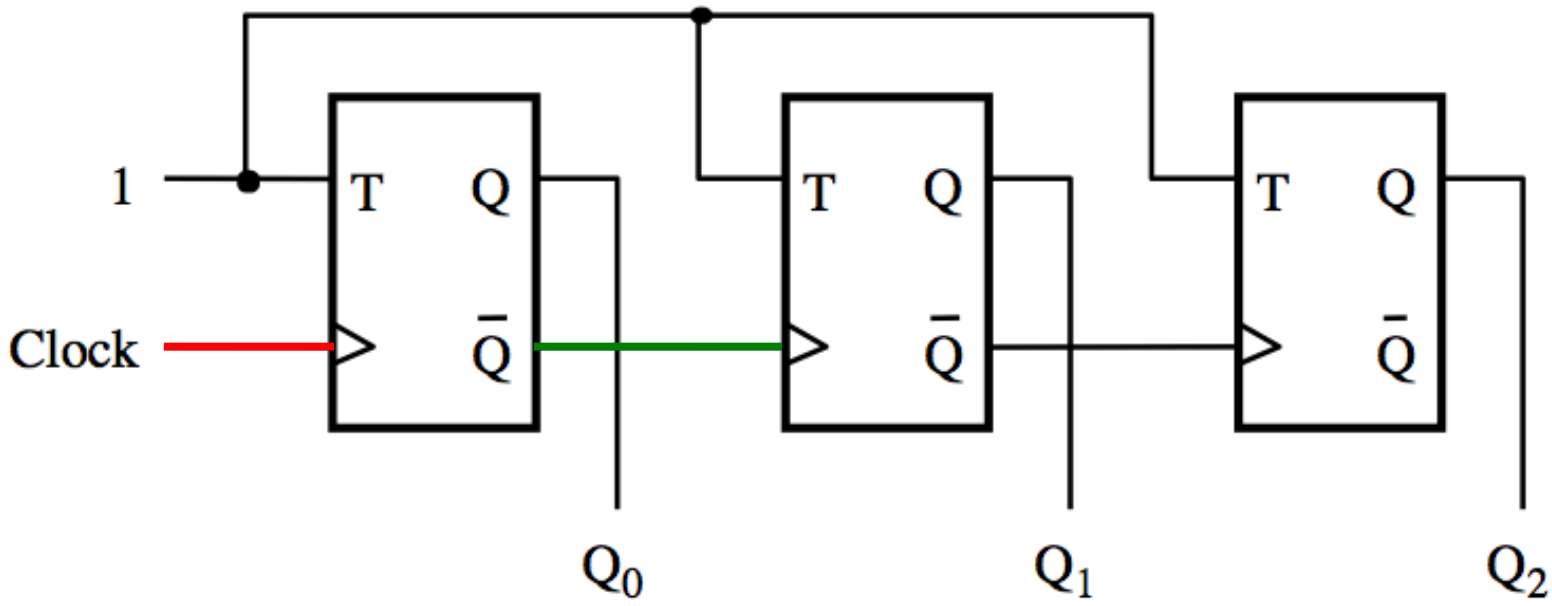
A three-bit up-counter



The first flip-flop changes
on the positive edge of the clock

[Figure 5.19 from the textbook]

A three-bit up-counter

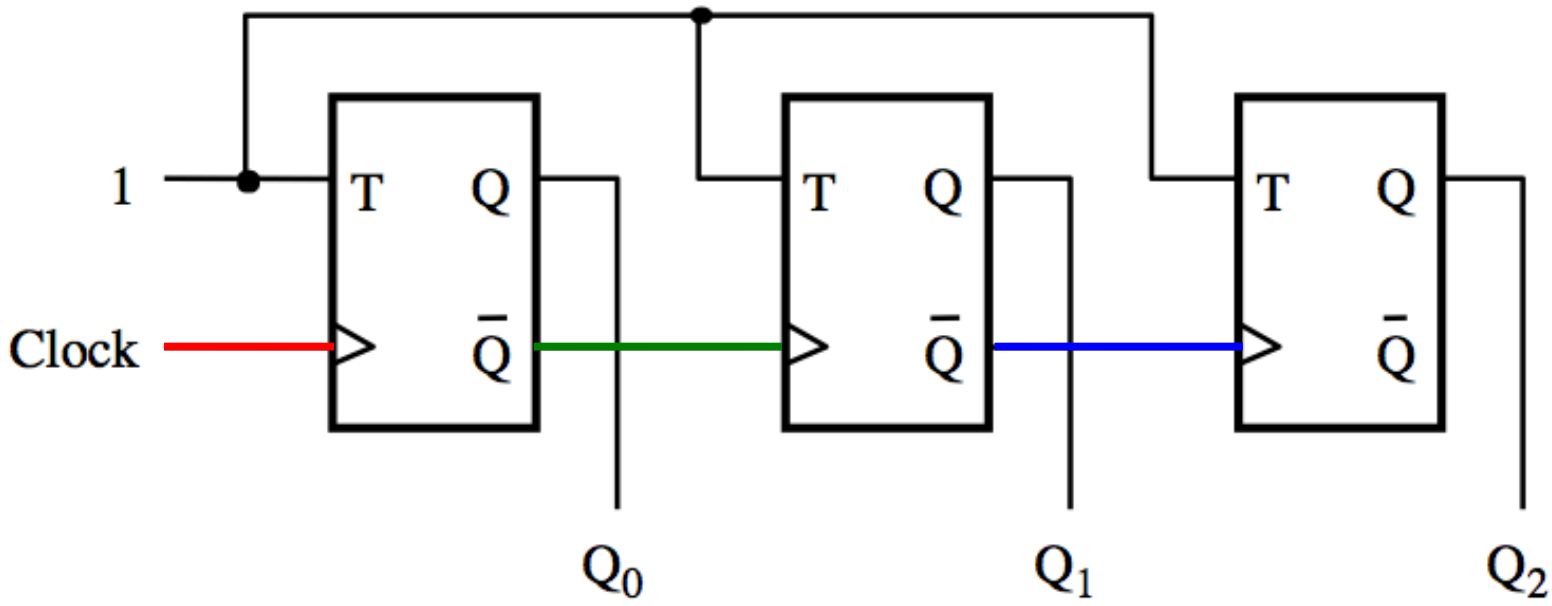


The first flip-flop changes
on the positive edge of the clock

The second flip-flop changes
on the positive edge of $\overline{Q}_0$

[Figure 5.19 from the textbook]

A three-bit up-counter



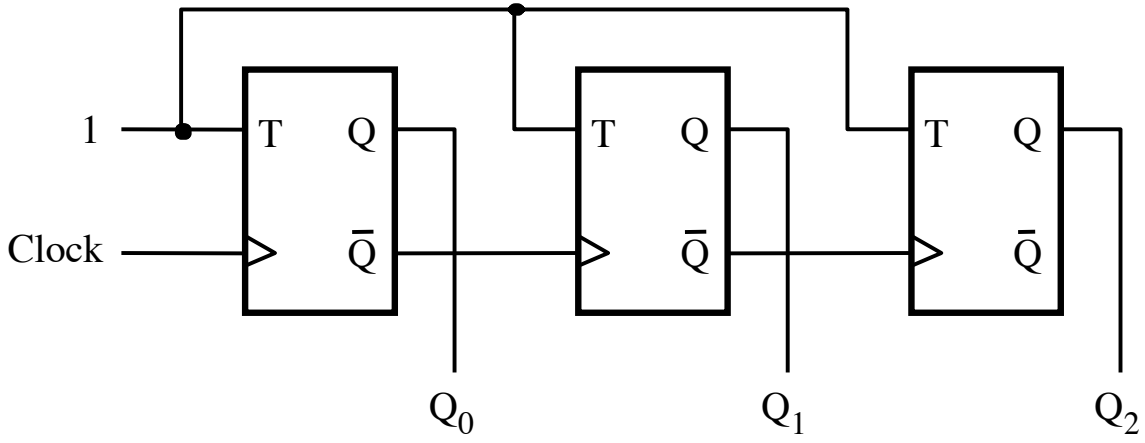
The first flip-flop changes on the positive edge of the clock

The second flip-flop changes on the positive edge of $\overline{Q}_0$

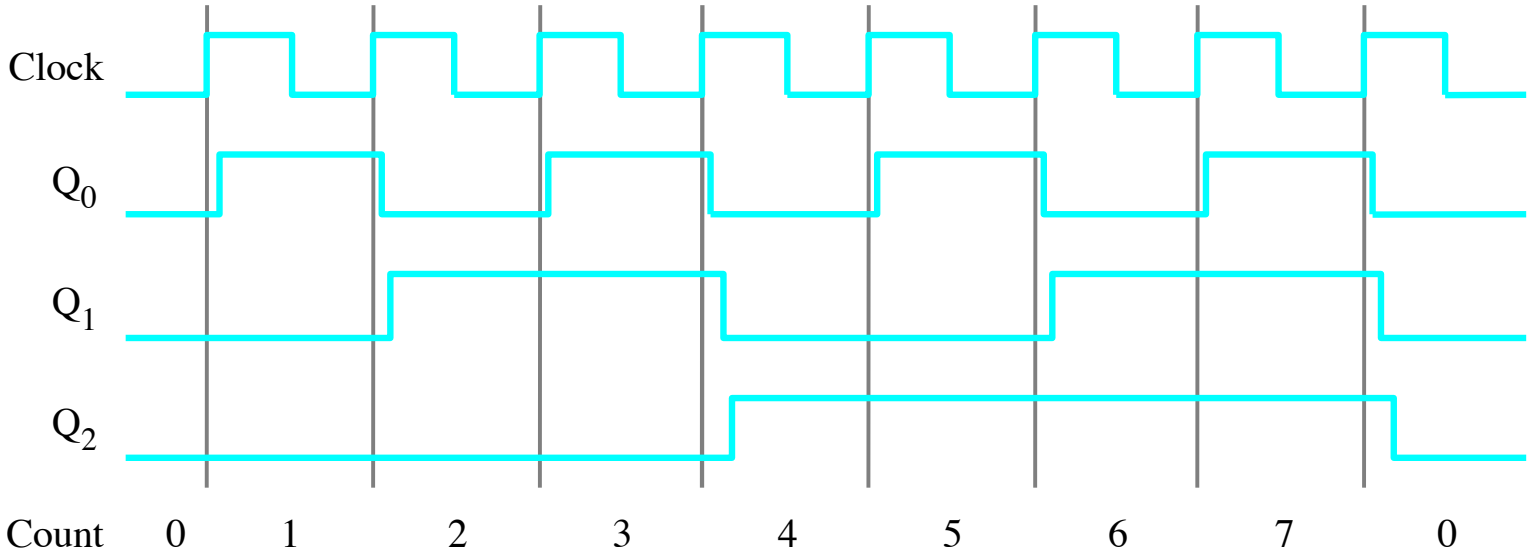
The third flip-flop changes on the positive edge of $\overline{Q}_1$

[Figure 5.19 from the textbook]

A three-bit up-counter



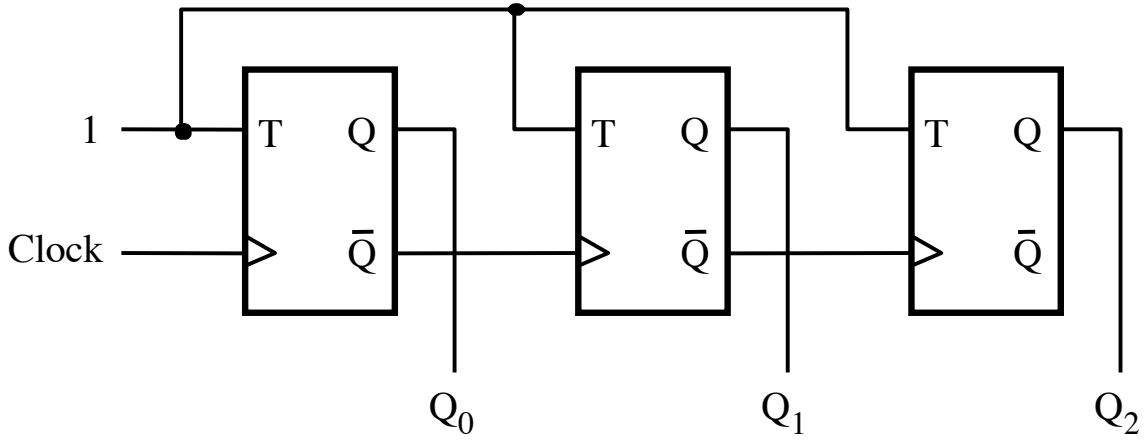
(a) Circuit



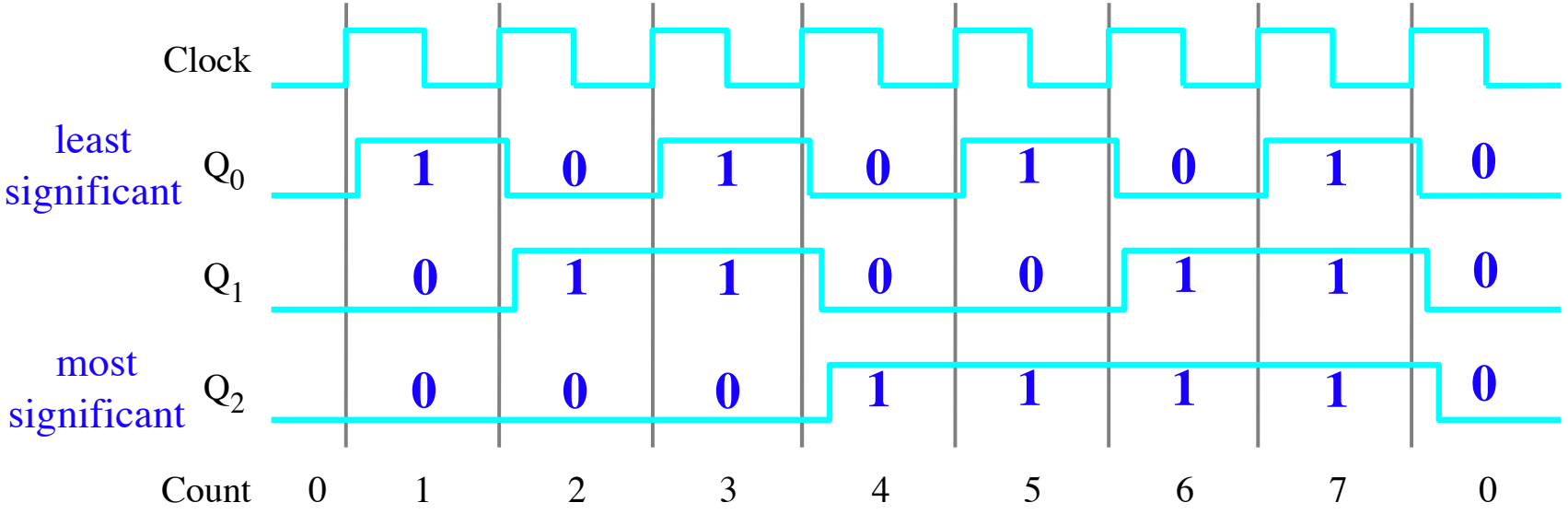
(b) Timing diagram

[Figure 5.19 from the textbook]

A three-bit up-counter



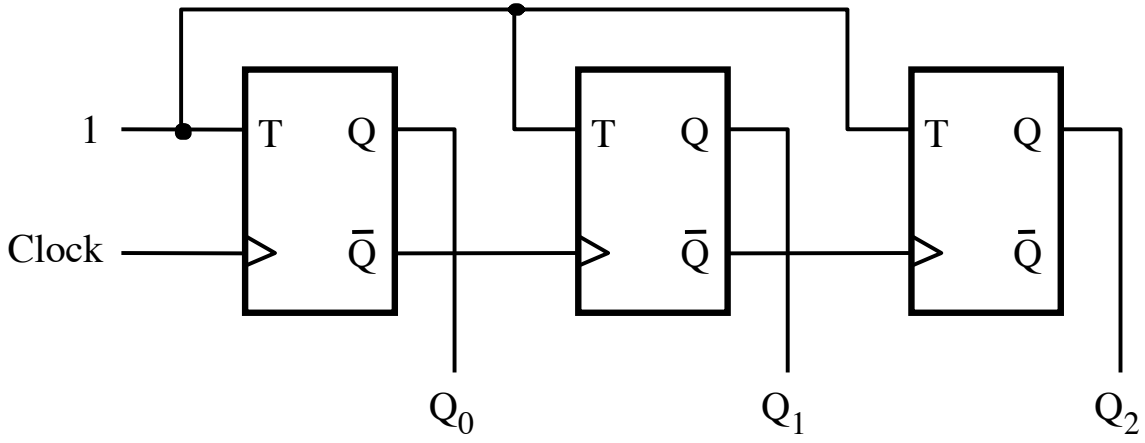
(a) Circuit



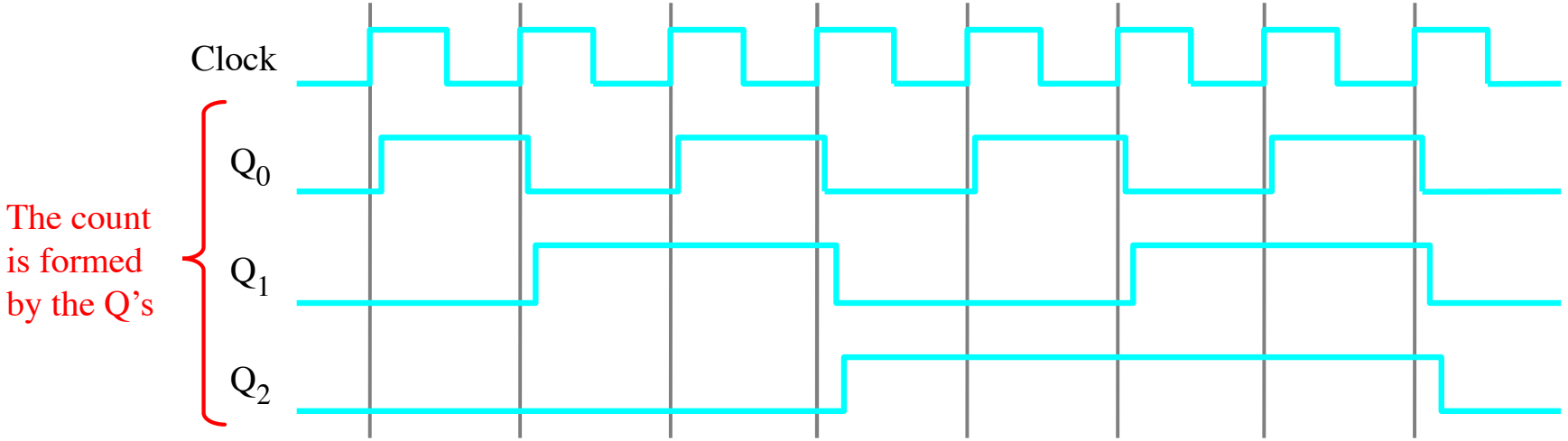
(b) Timing diagram

[Figure 5.19 from the textbook]

A three-bit up-counter



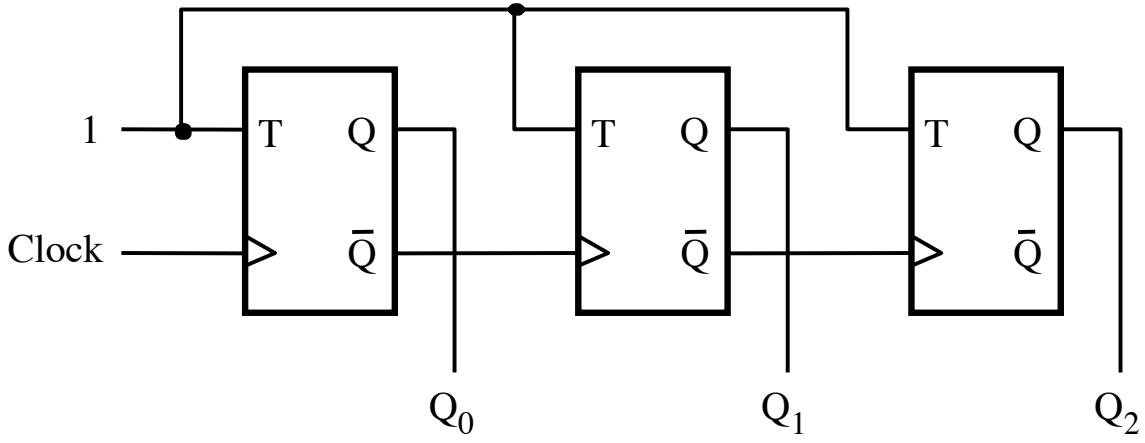
(a) Circuit



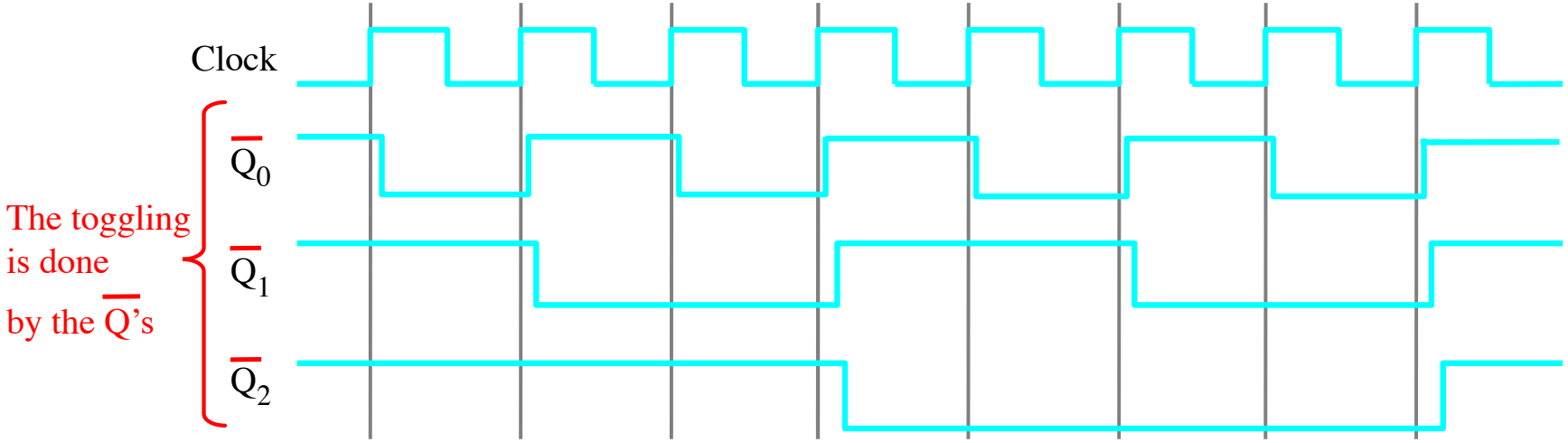
(b) Timing diagram

[Figure 5.19 from the textbook]

A three-bit up-counter

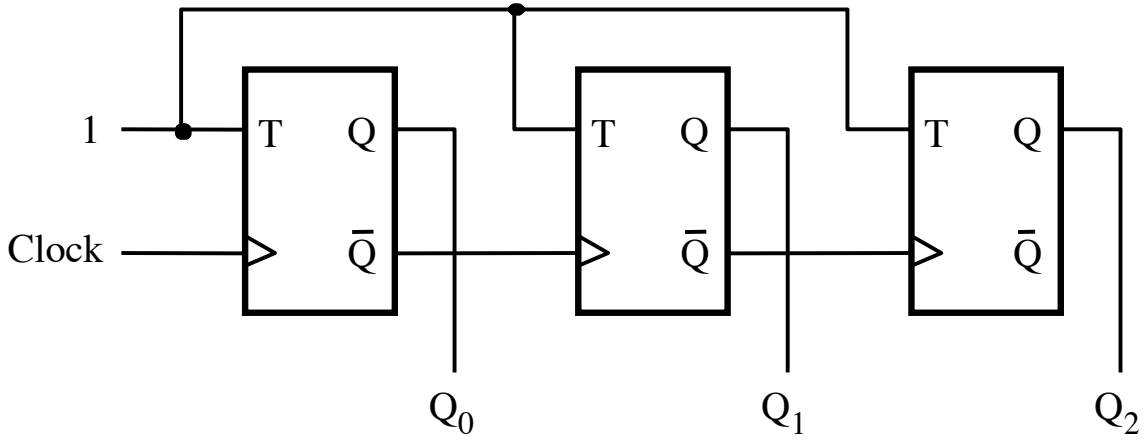


(a) Circuit

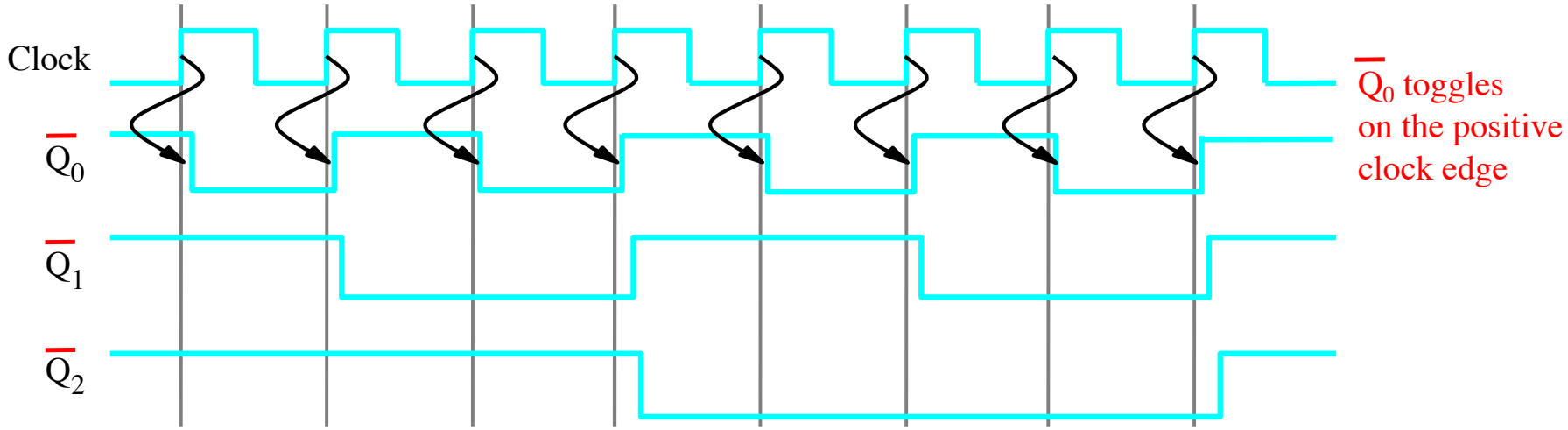


(b) Timing diagram

A three-bit up-counter

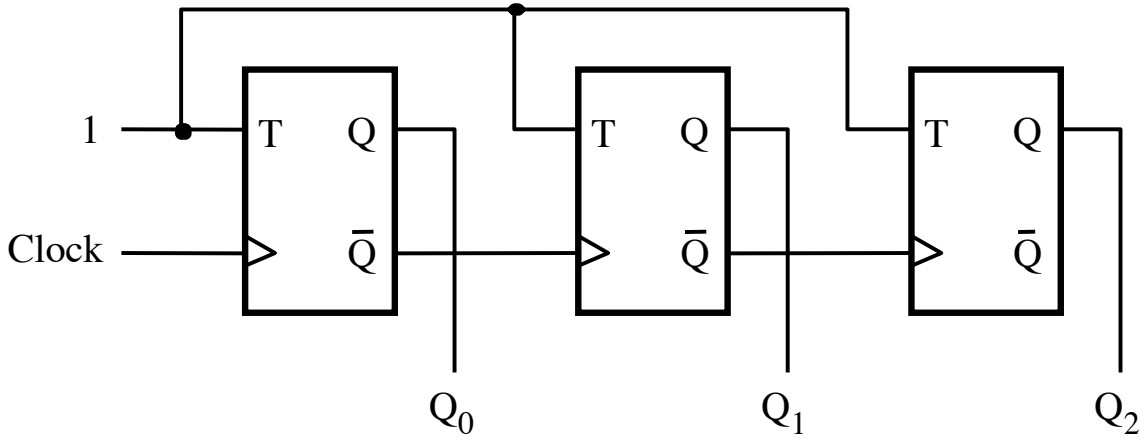


(a) Circuit

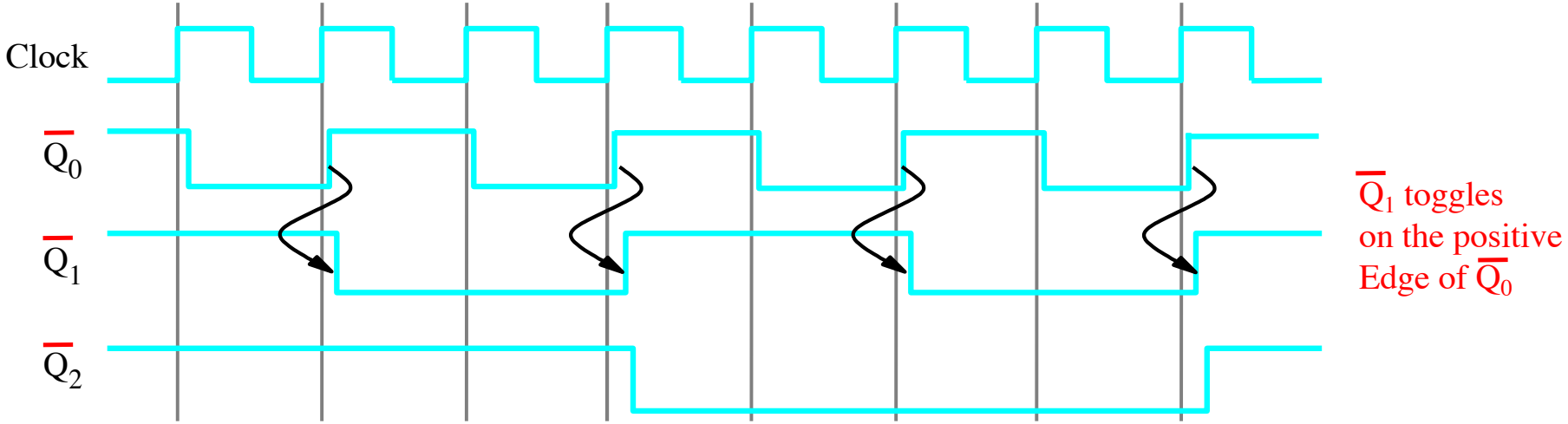


(b) Timing diagram

A three-bit up-counter

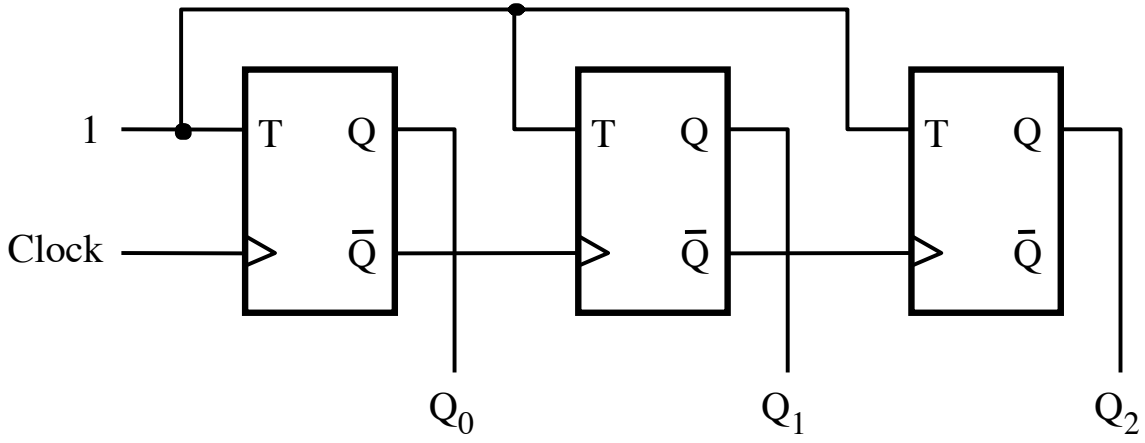


(a) Circuit

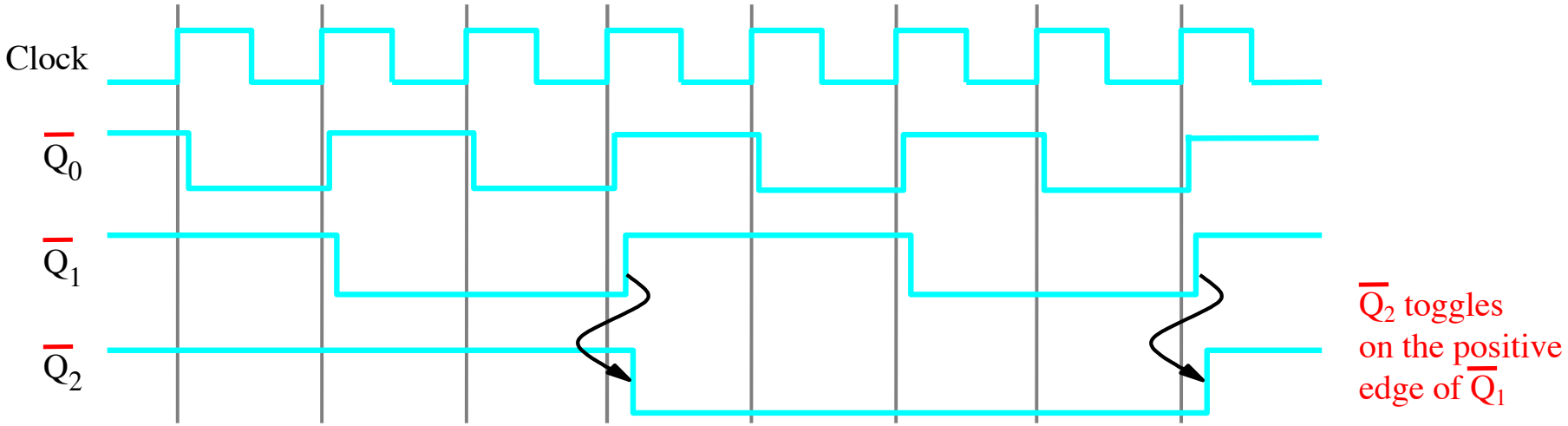


(b) Timing diagram

A three-bit up-counter

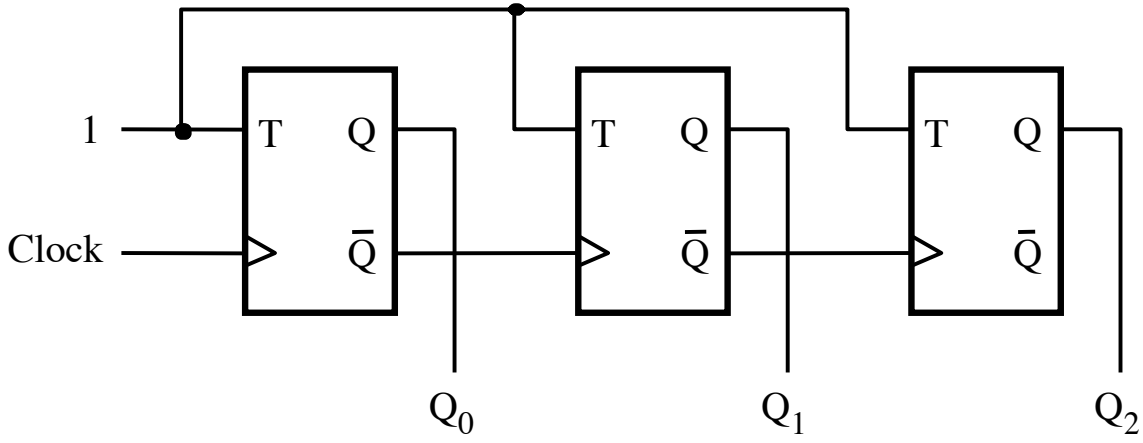


(a) Circuit



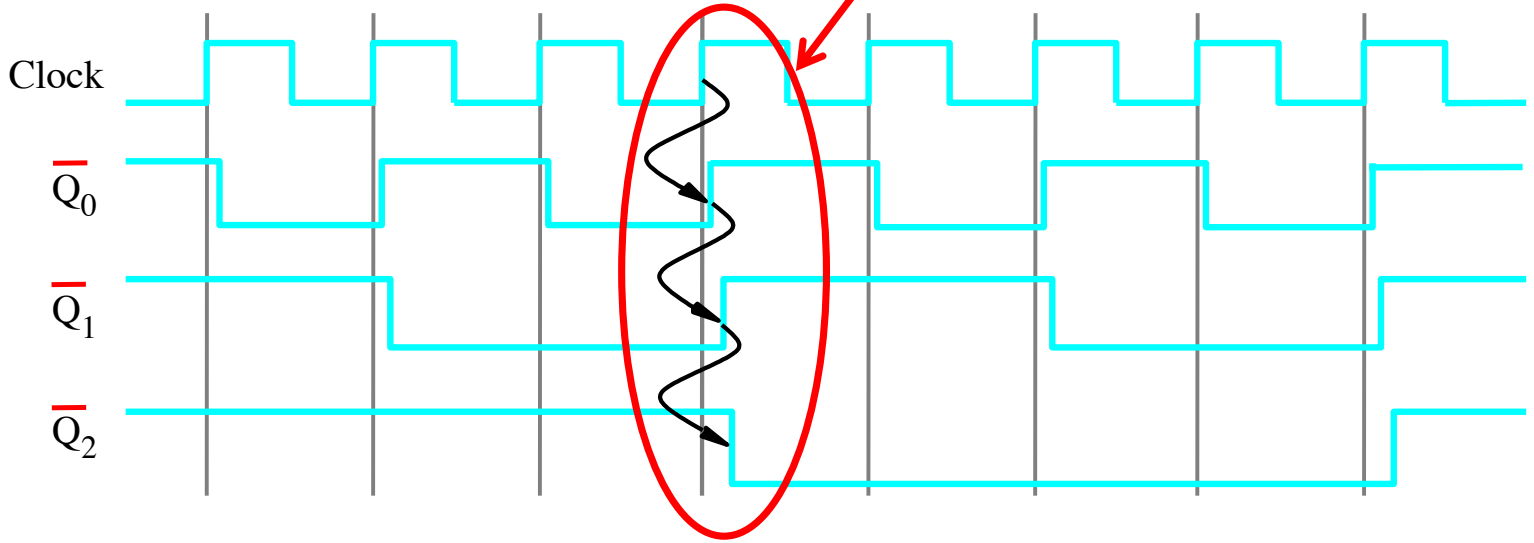
(b) Timing diagram

A three-bit up-counter



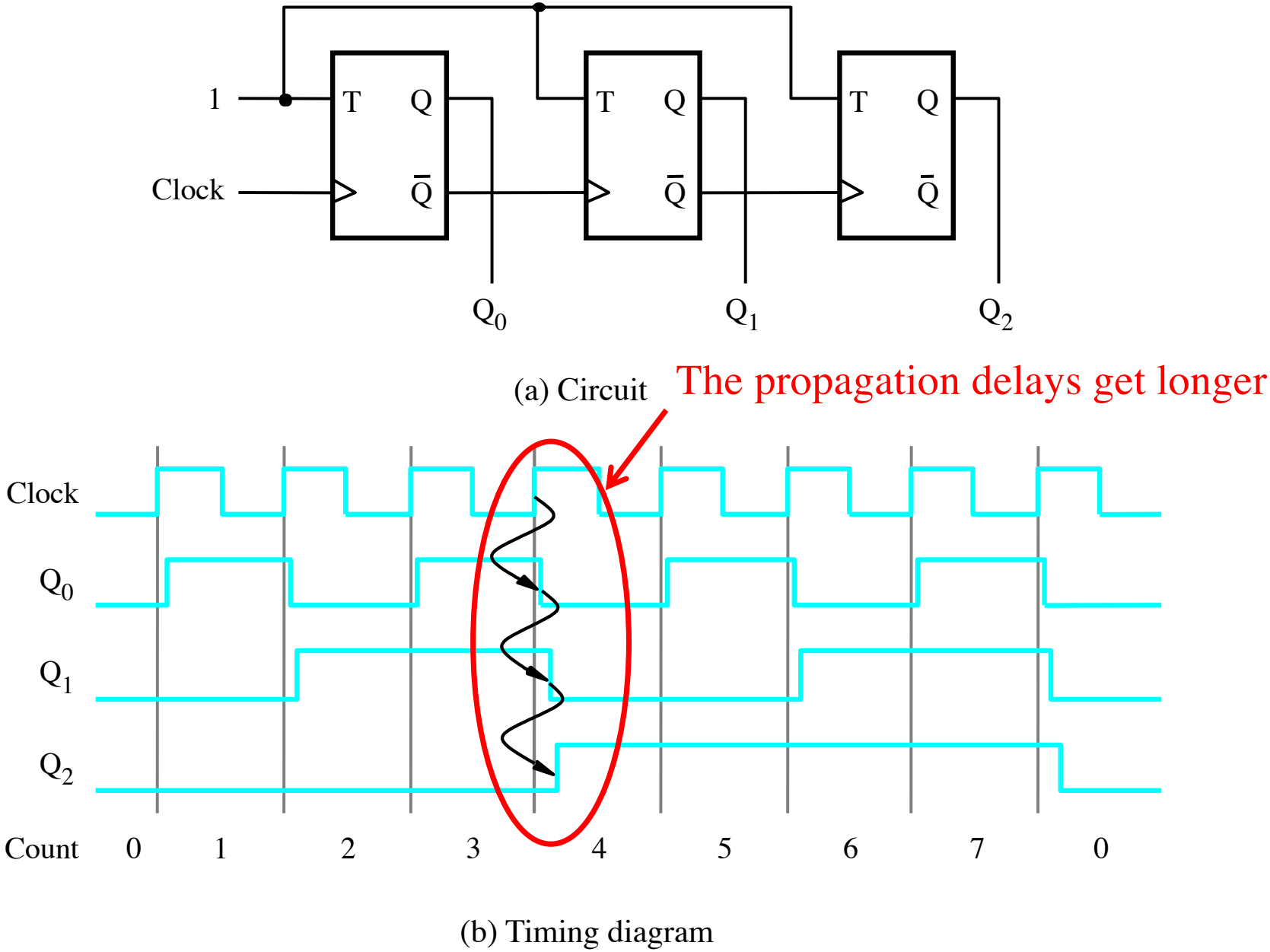
(a) Circuit

The propagation delays get longer



(b) Timing diagram

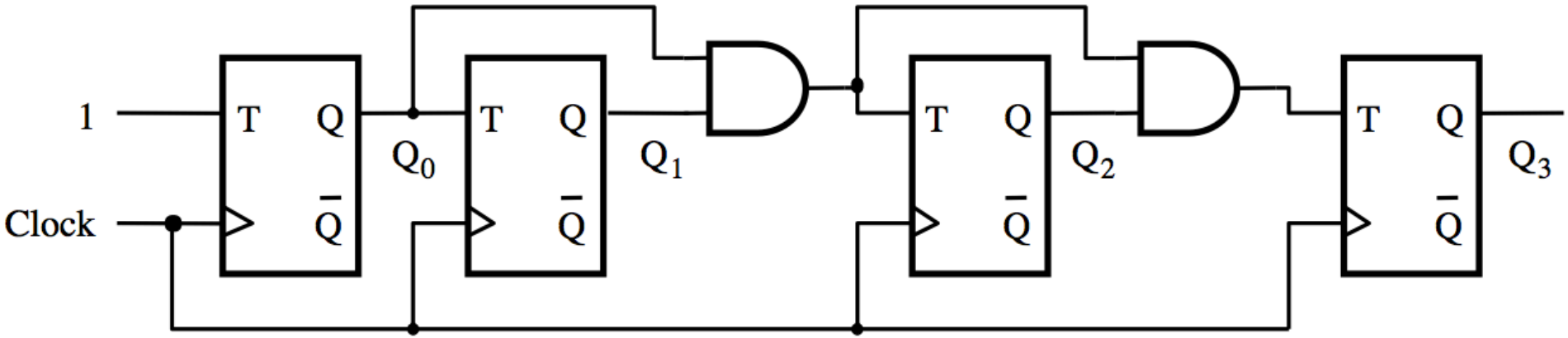
A three-bit up-counter



[Figure 5.19 from the textbook]

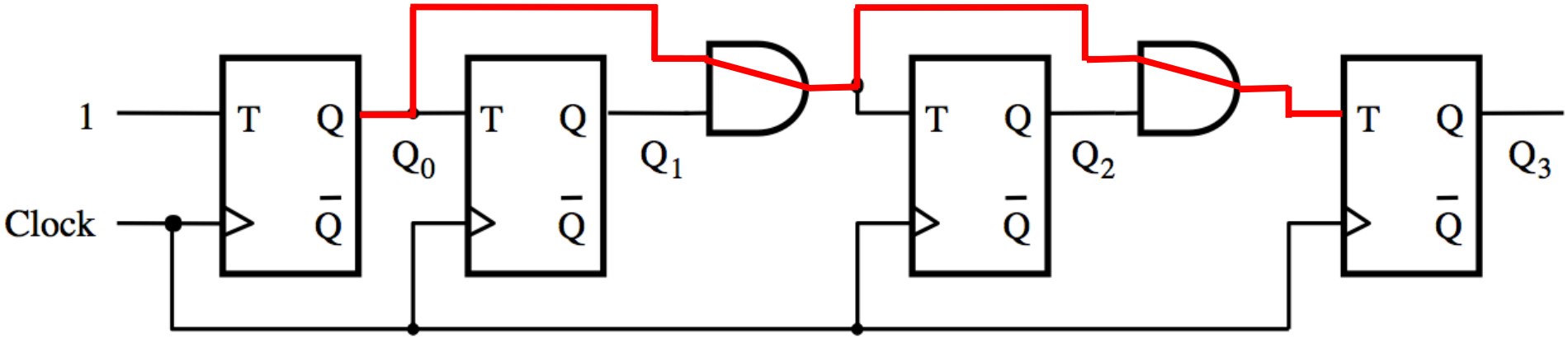
Synchronous Counters

A four-bit synchronous up-counter



[Figure 5.21 from the textbook]

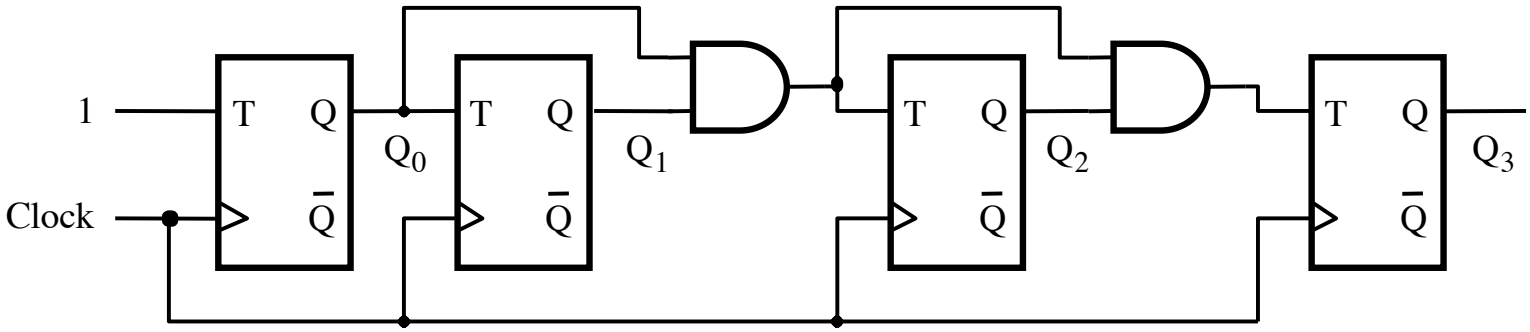
A four-bit synchronous up-counter



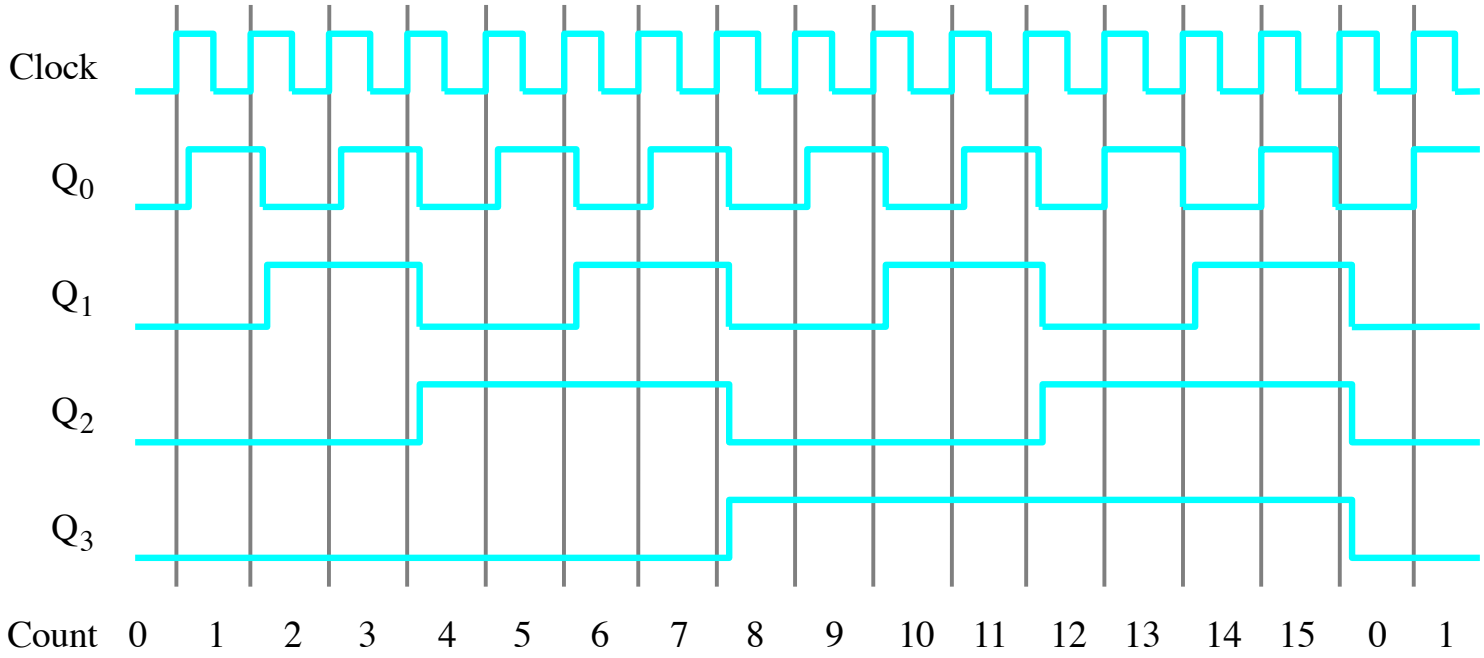
The propagation delay through all AND gates combined must not exceed the clock period minus the setup time for the flip-flops

[Figure 5.21 from the textbook]

A four-bit synchronous up-counter



(a) Circuit

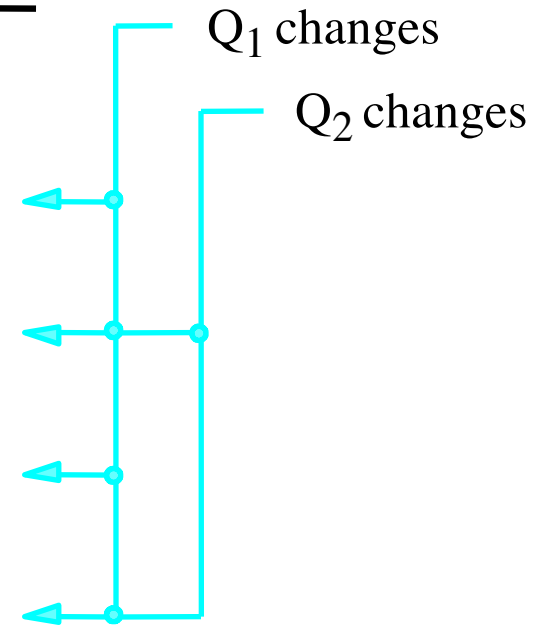


(b) Timing diagram

[Figure 5.21 from the textbook]

Derivation of the synchronous up-counter

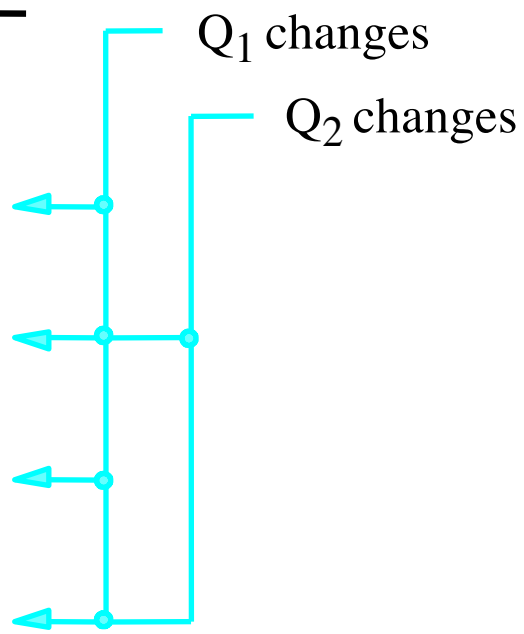
Clock cycle	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0



[Table 5.1 from the textbook]

Derivation of the synchronous up-counter

Clock cycle	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0



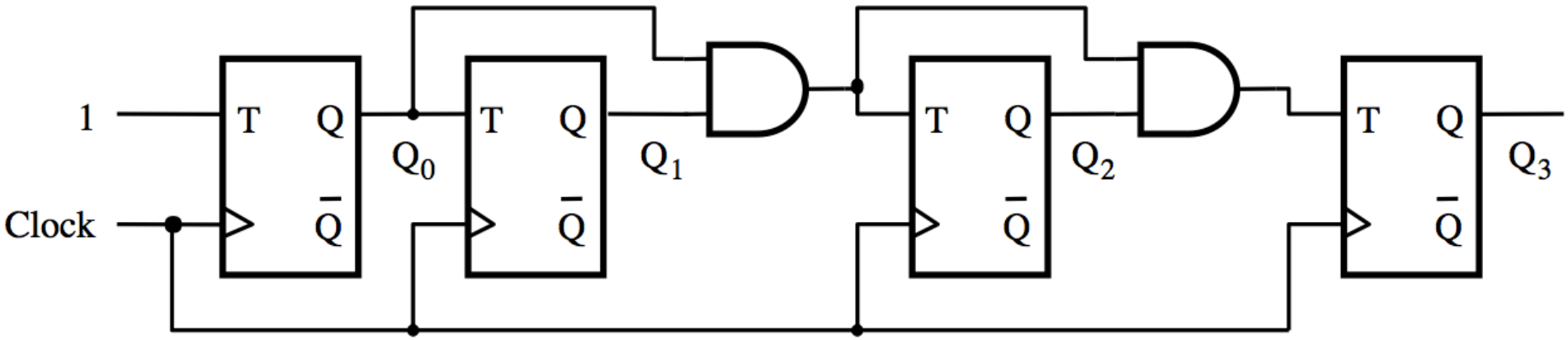
$T_0 = 1$

$T_1 = Q_0$

$T_2 = Q_0\ Q_1$

[Table 5.1 from the textbook]

A four-bit synchronous up-counter



$T_0 = 1$

$T_1 = Q_0$

$T_2 = Q_0 Q_1$

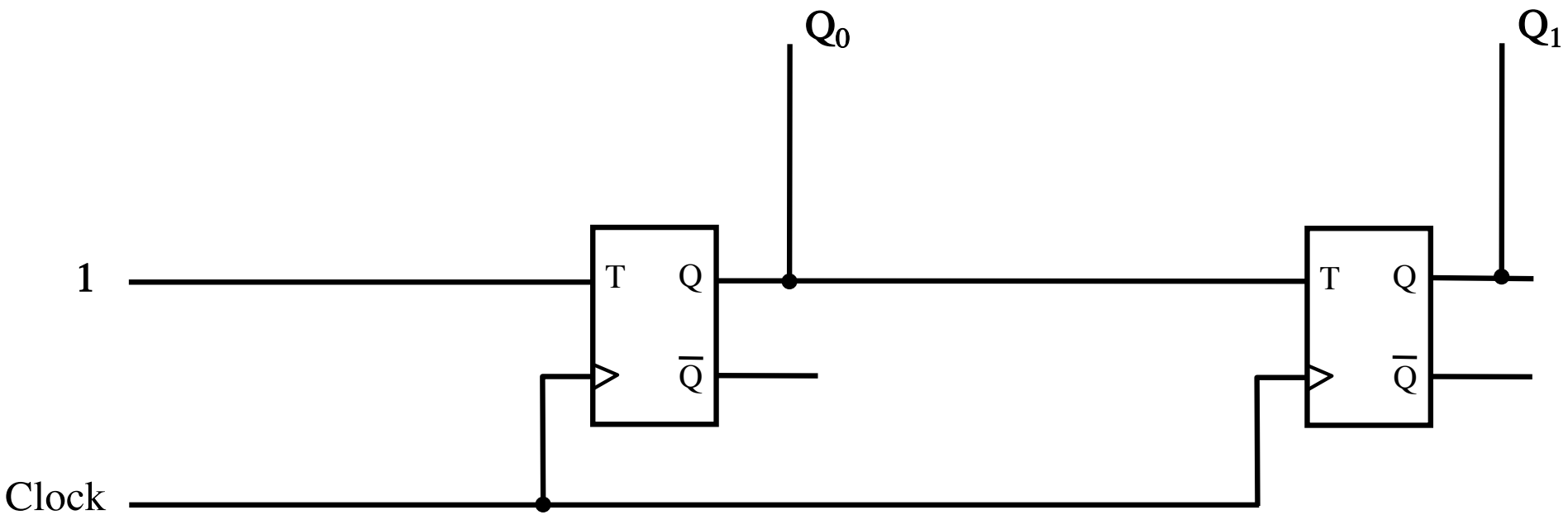
[Figure 5.21 from the textbook]

In general we have

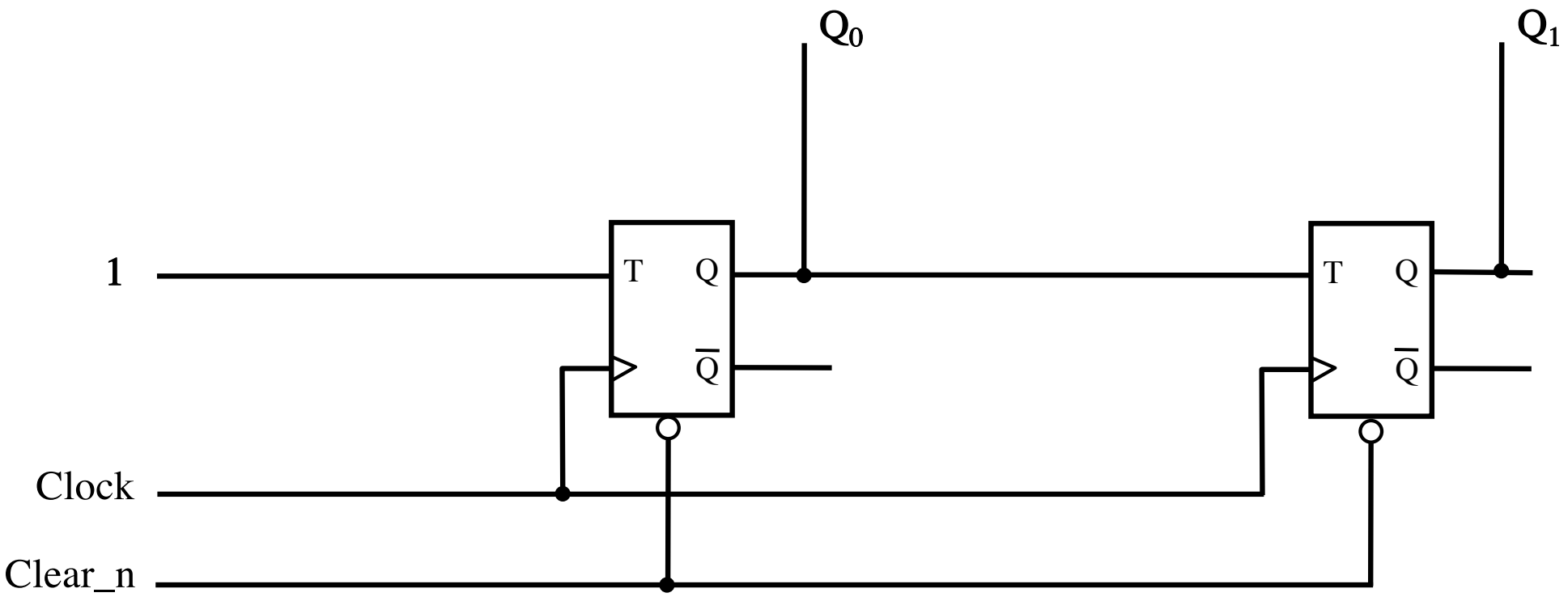
$$T_0 = 1$$
$$T_1 = Q_0$$
$$T_2 = Q_0\, Q_1$$
$$T_3 = Q_0\, Q_1\, Q_2$$
$$\cdots$$
$$T_n = Q_0\, Q_1\, Q_2 \cdots Q_{n-1}$$

Synchronous v.s. Asynchronous Clear

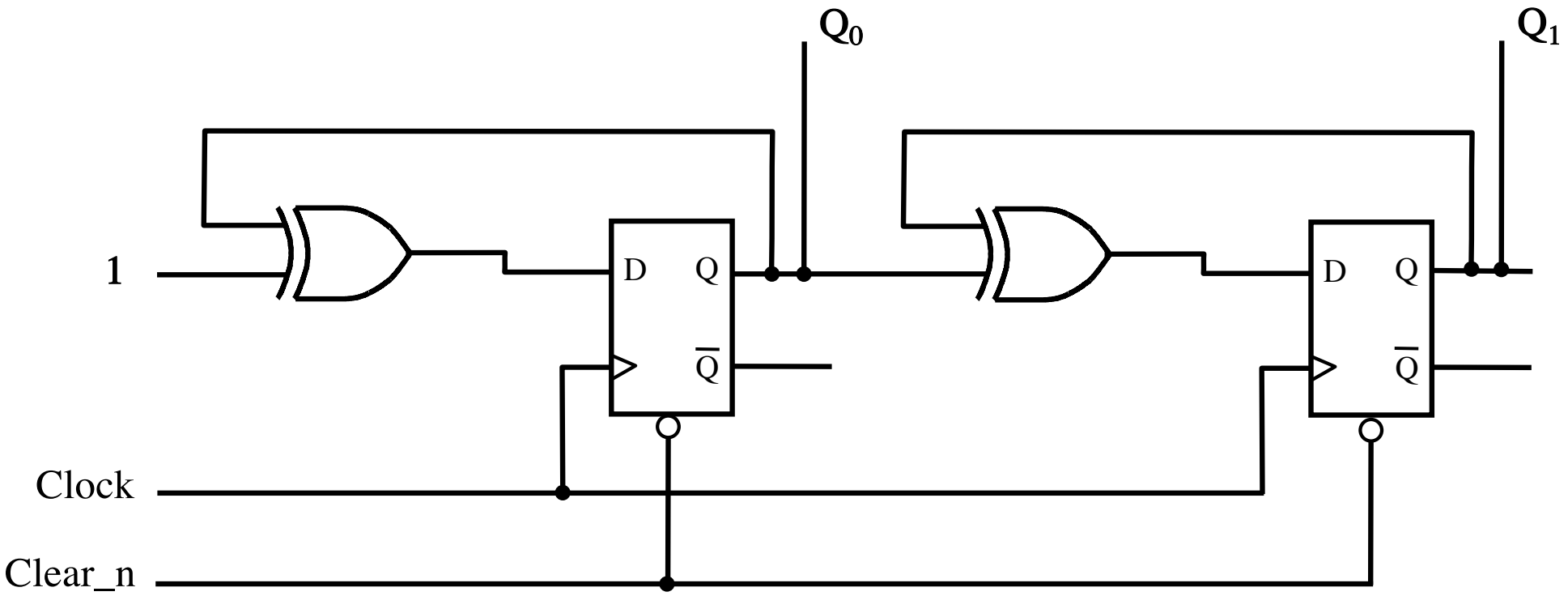
2-Bit Synchronous Up-Counter (without clear capability)



2-Bit Synchronous Up-Counter (with asynchronous clear)

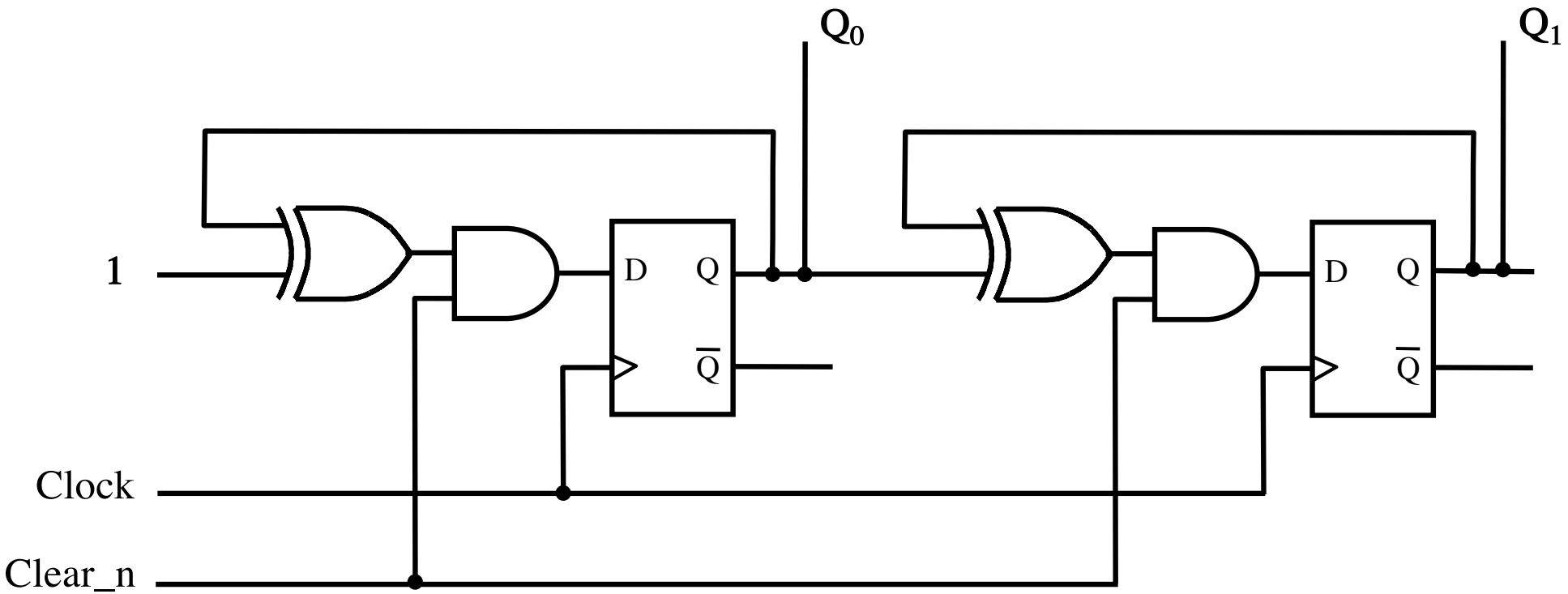


2-Bit Synchronous Up-Counter (with asynchronous clear)



This is the same circuit but uses D Flip-Flops.

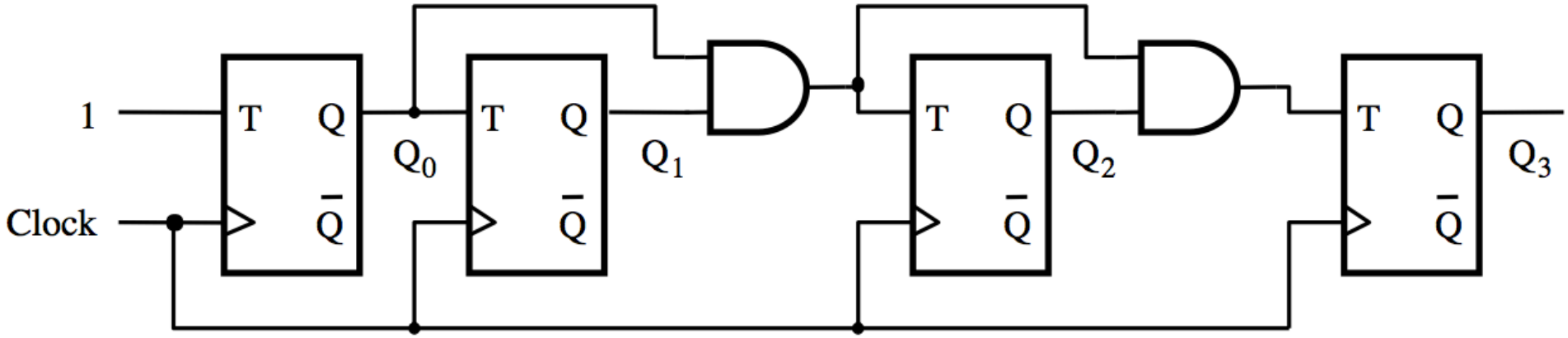
2-Bit Synchronous Up-Counter (with synchronous clear)



This counter can be cleared only on the positive clock edge.

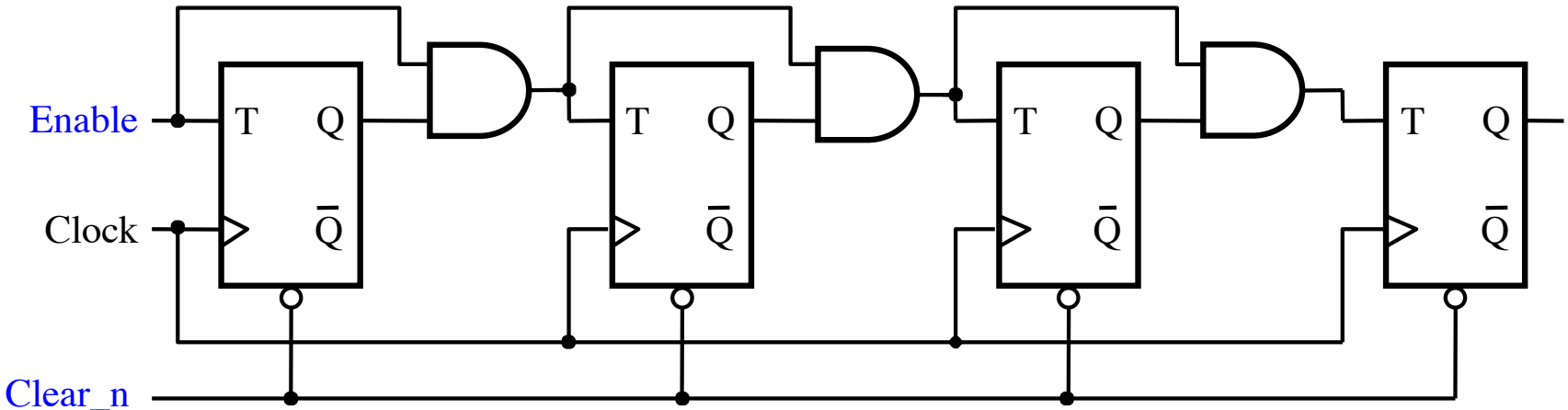
Adding Enable Capability

A four-bit synchronous up-counter



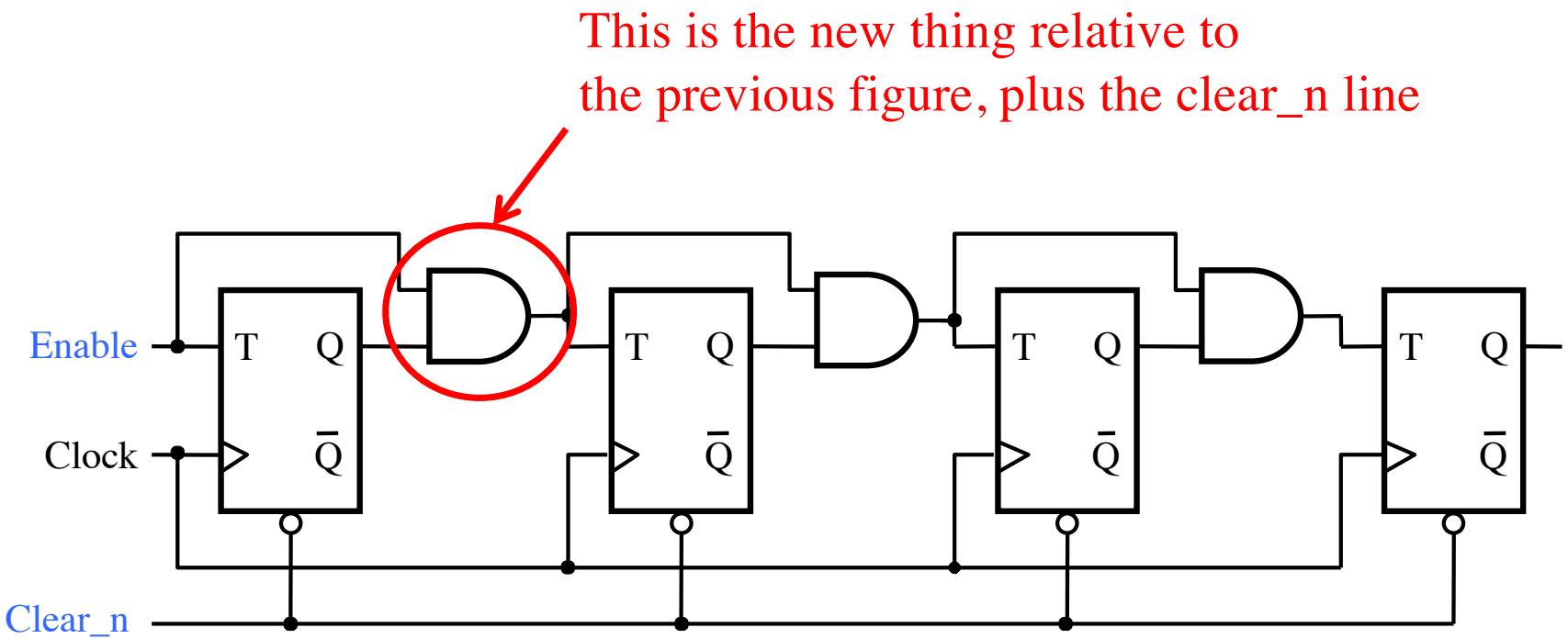
[Figure 5.21 from the textbook]

Inclusion of Enable and Clear Capability



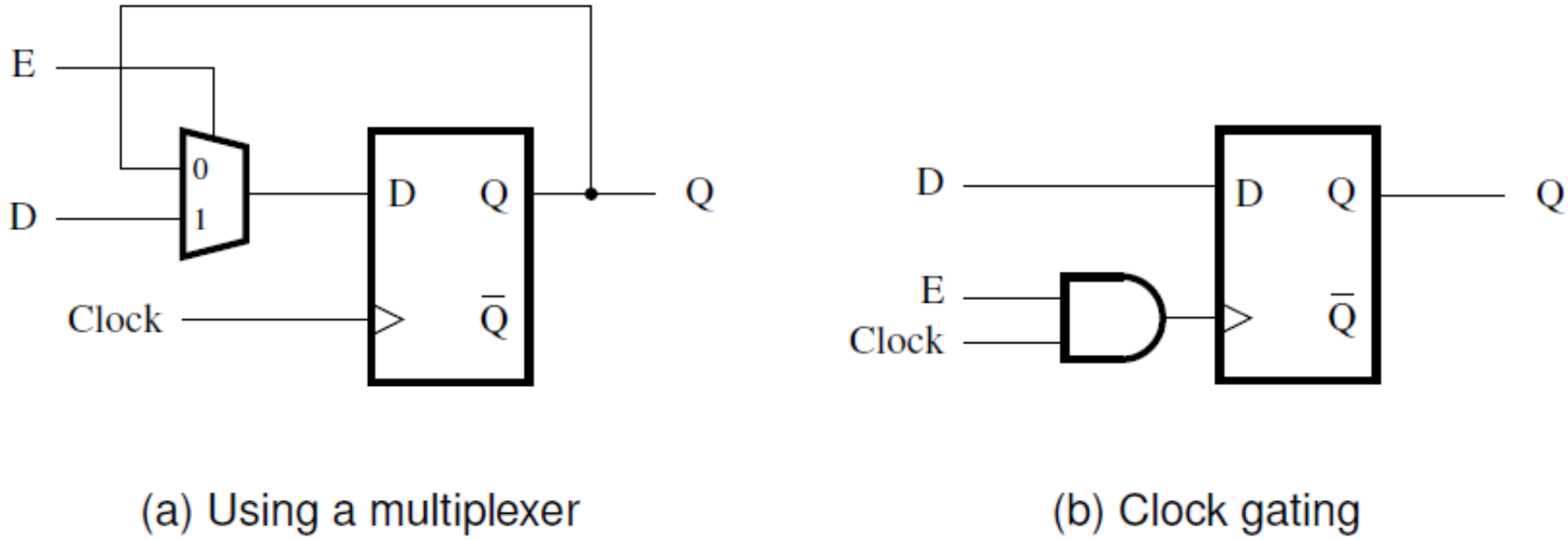
[Figure 5.22 from the textbook]

Inclusion of Enable and Clear Capability



[Figure 5.22 from the textbook]

Providing an enable input for a D flip-flop



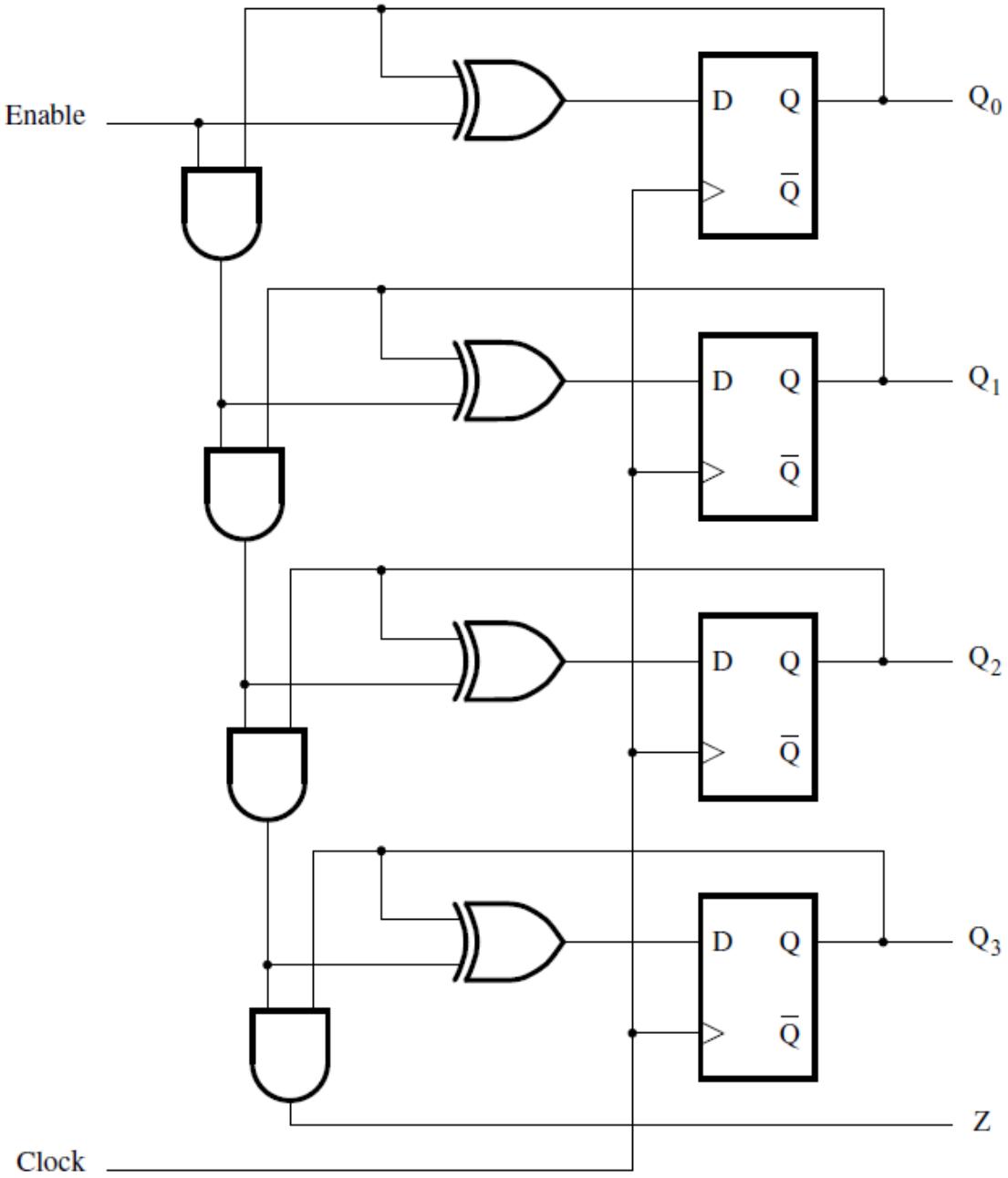
(a) Using a multiplexer

(b) Clock gating

[Figure 5.56 from the textbook]

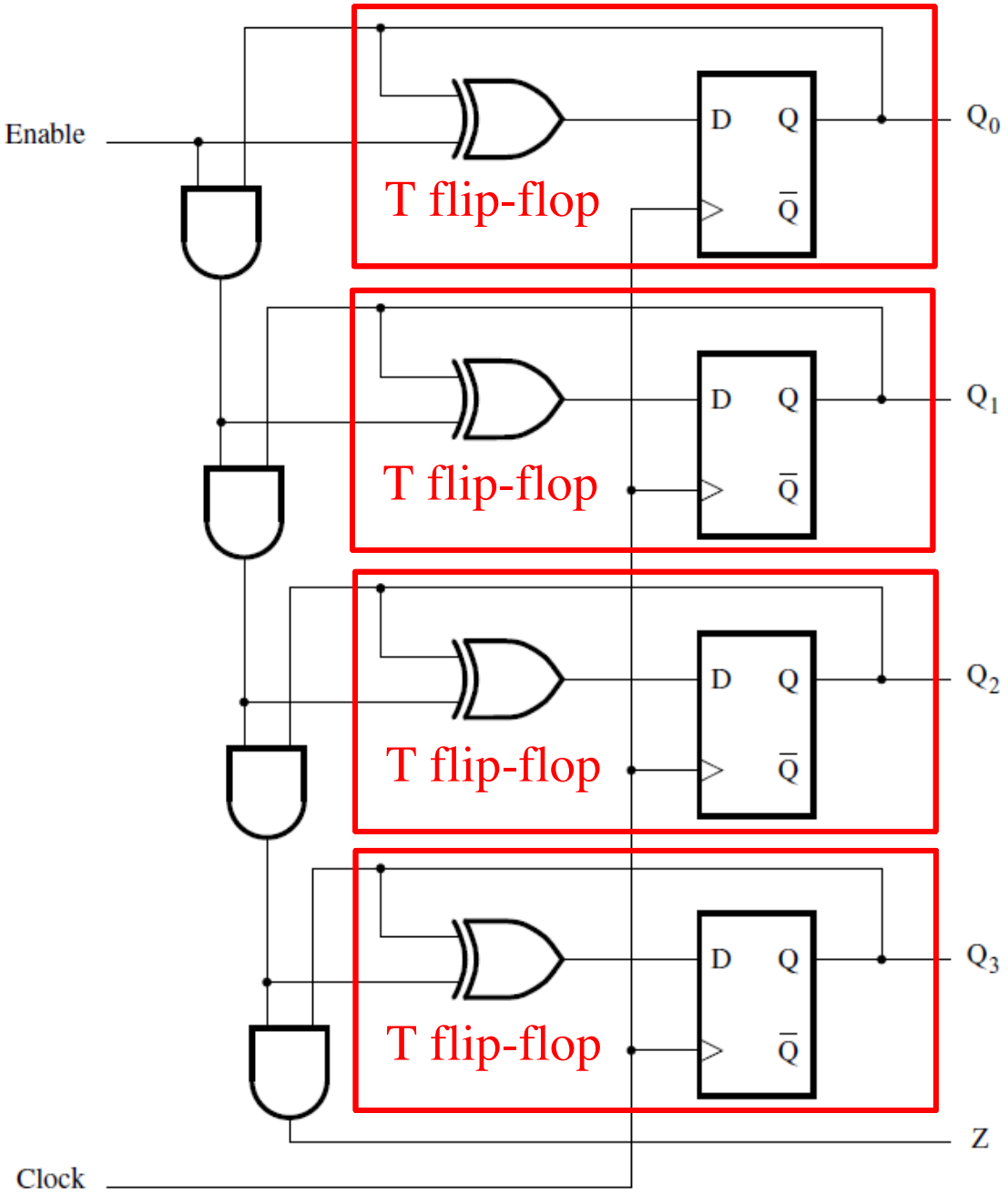
Synchronous Counter
(with D Flip-Flops)

A 4-bit up-counter with D flip-flops



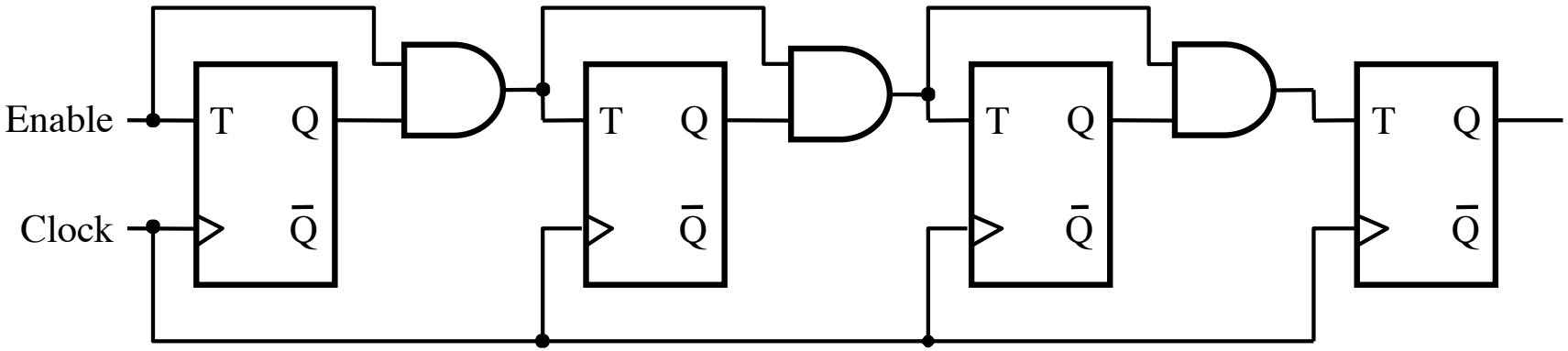
[Figure 5.23 from the textbook]

A 4-bit up-counter with D flip-flops

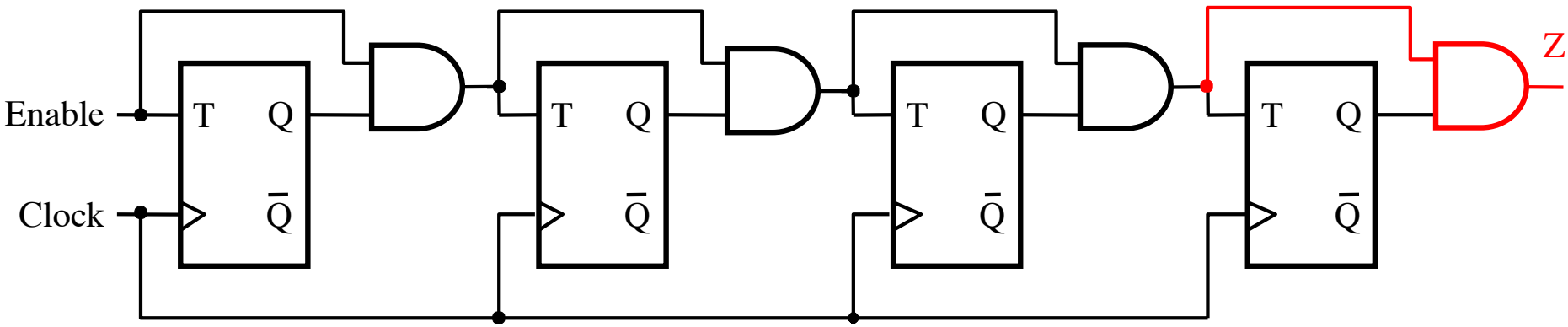


[Figure 5.23 from the textbook]

Equivalent to this circuit with T flip-flops



Equivalent to this circuit with T flip-flops

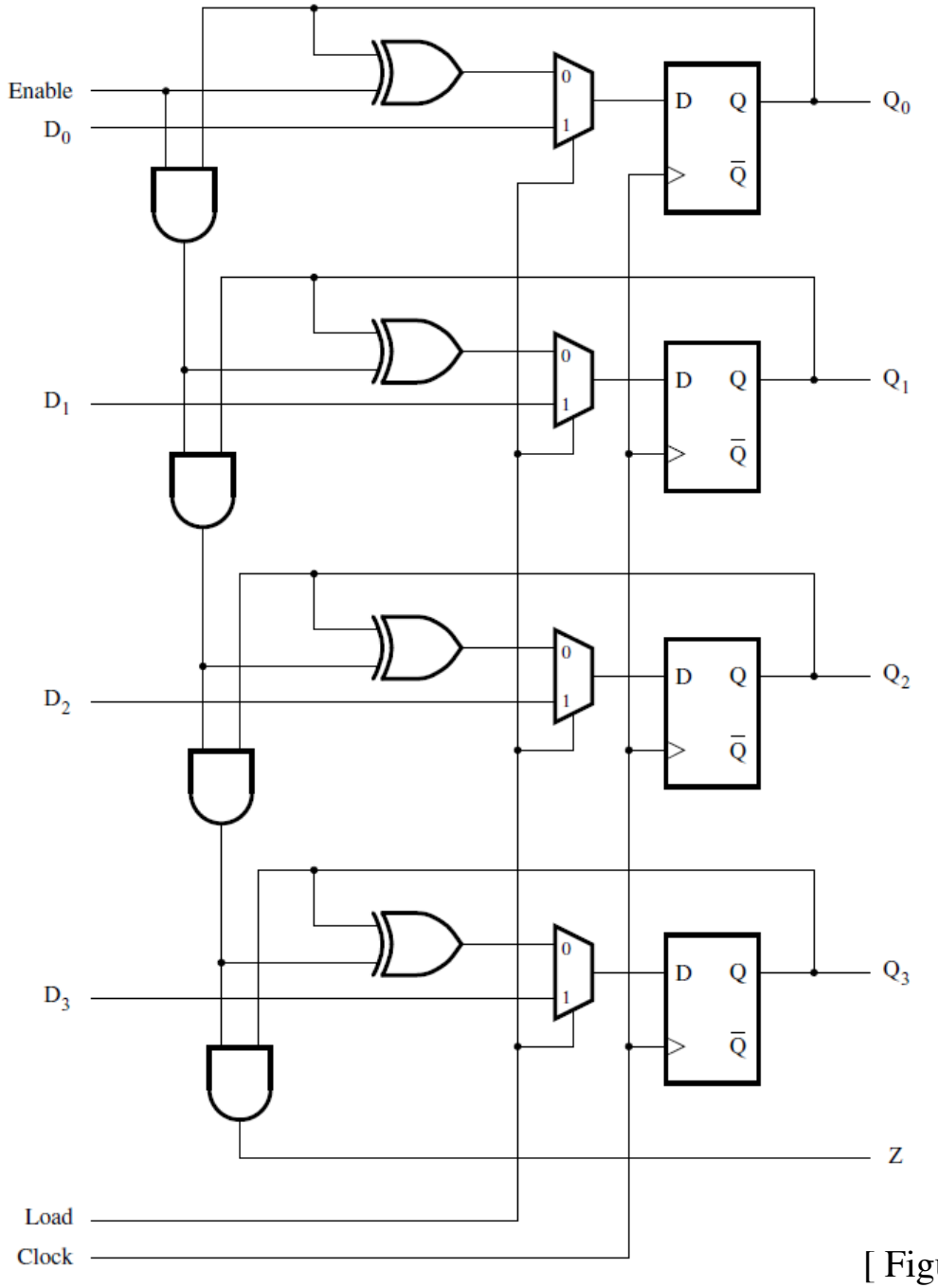


But has one extra output called Z, which can be used to connect two 4-bit counters to make an 8-bit counter.

When Z=1 the counter will go to 0000 on the next clock edge, i.e., the outputs of all flip-flops are currently 1 (maximum count value).

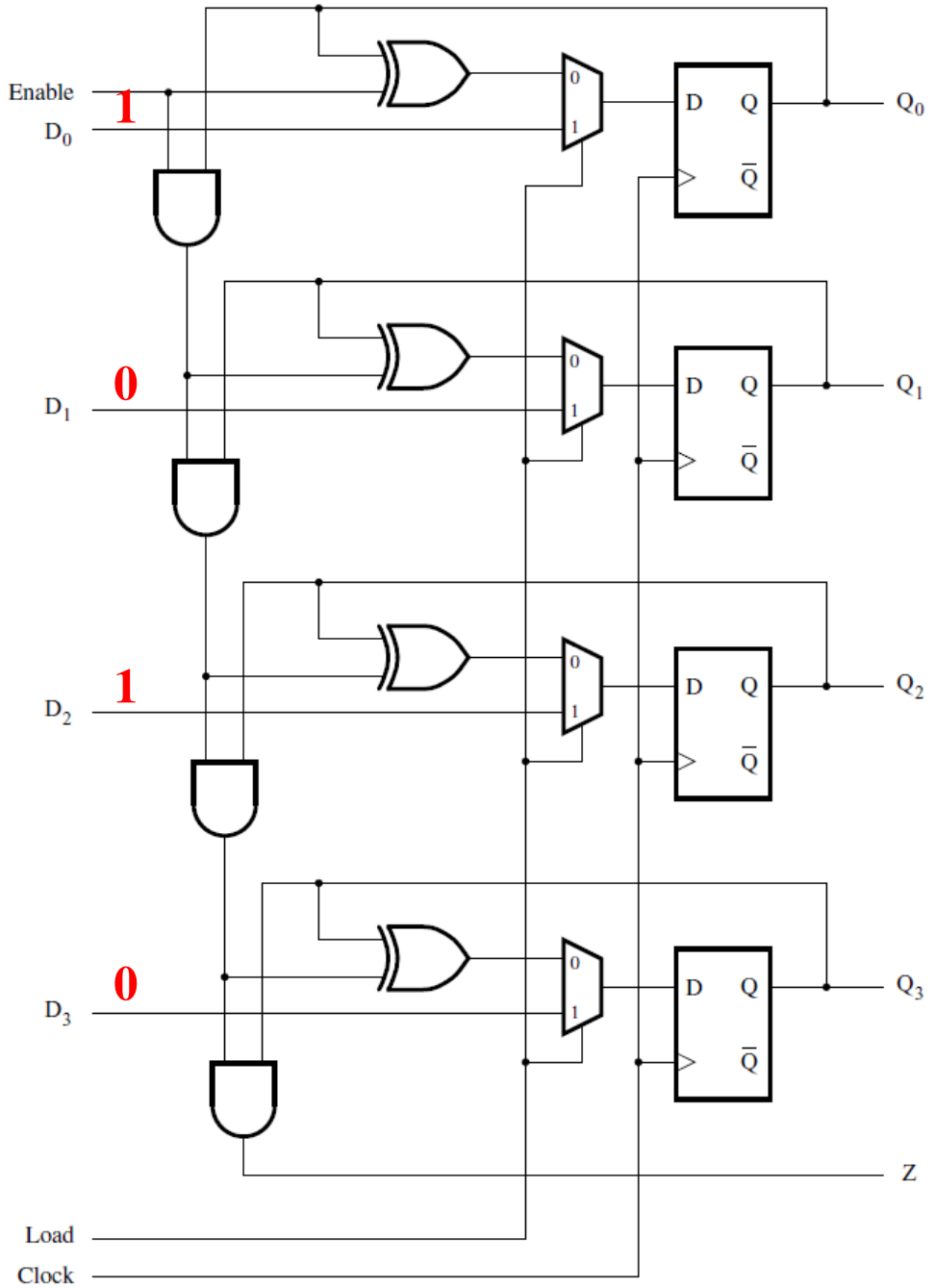
Counters with Parallel Load

A counter with parallel-load capability



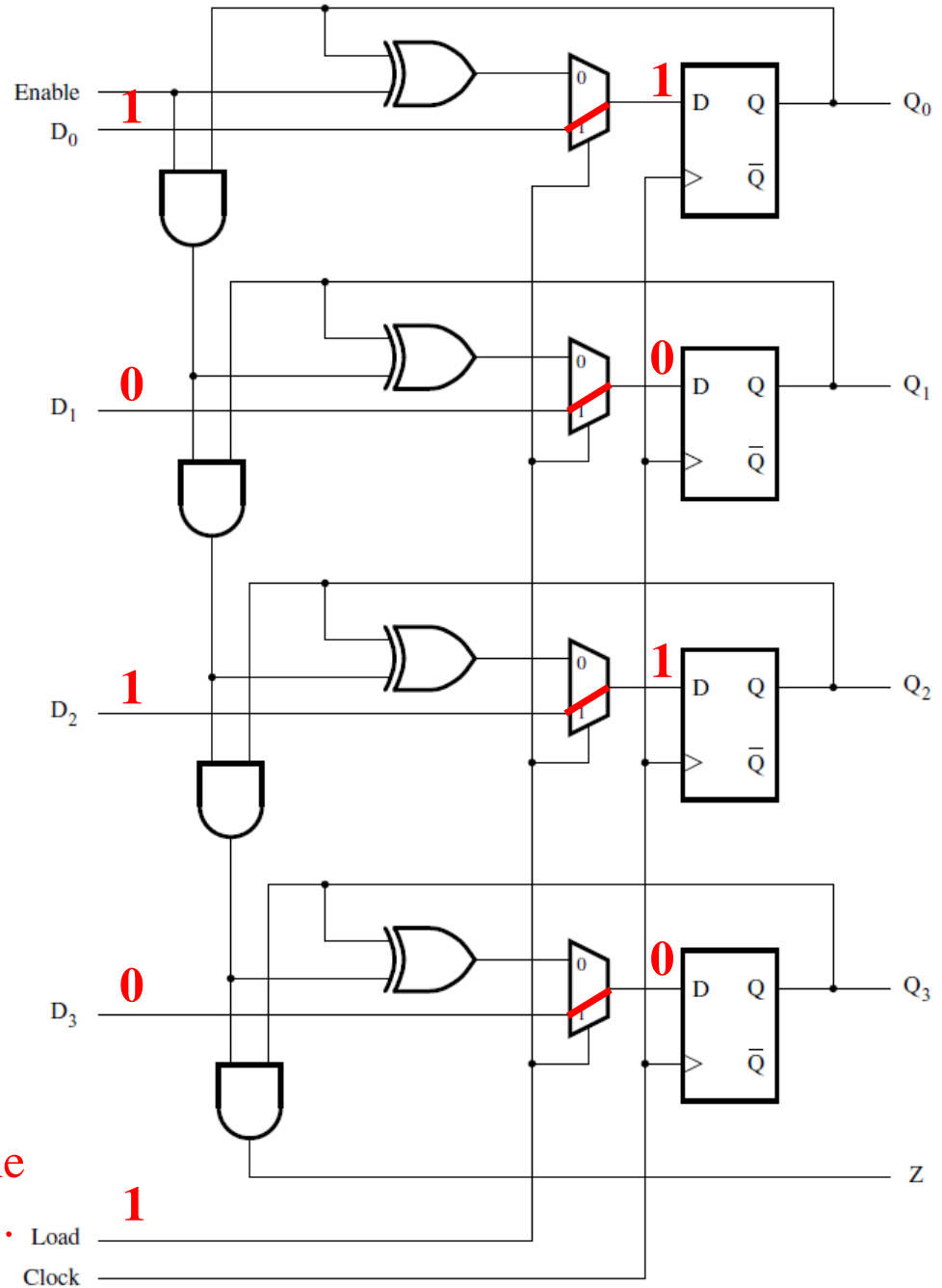
[Figure 5.24 from the textbook]

How to load the initial count value



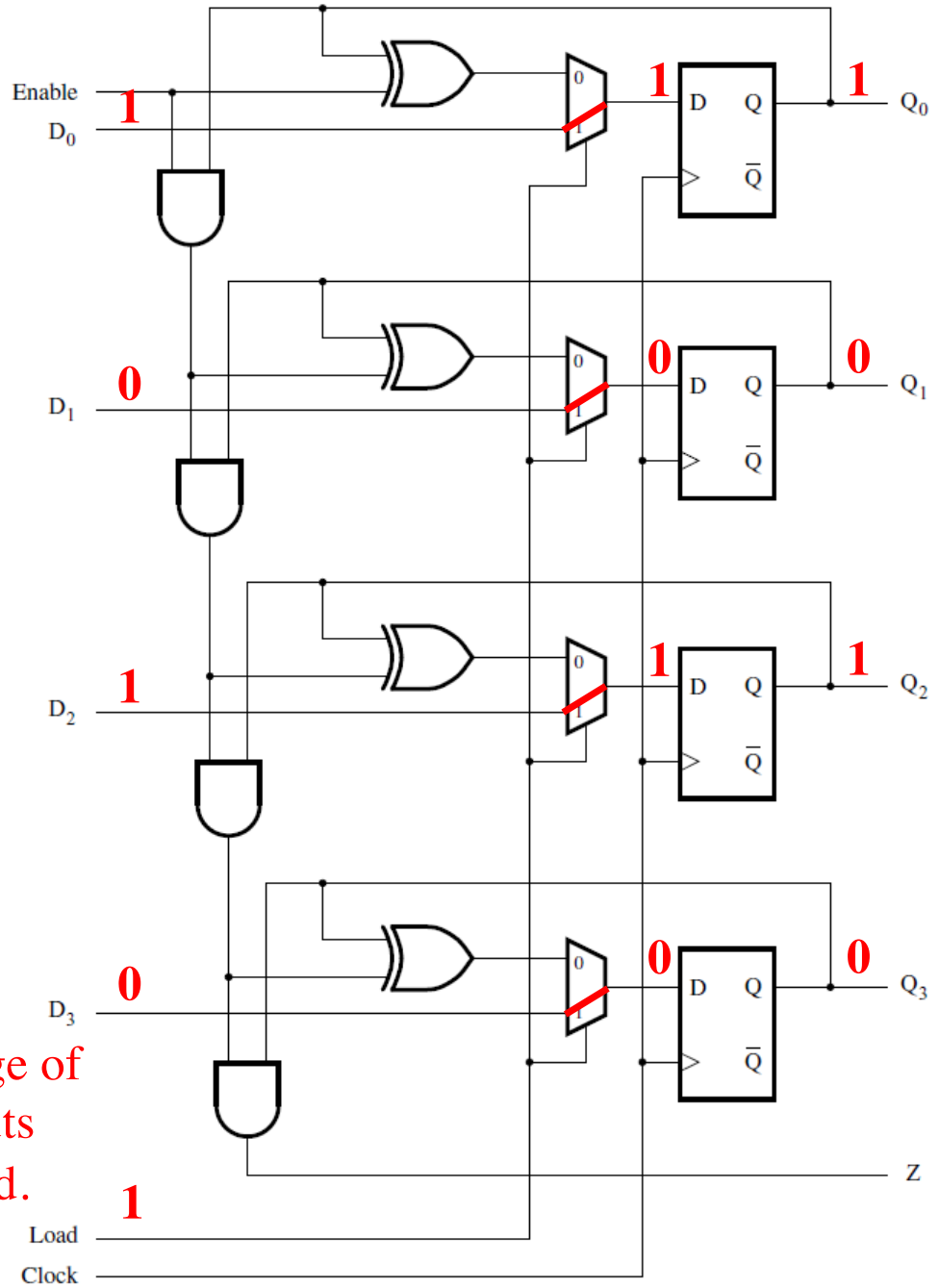
Set the initial count on the parallel load lines (in this case 5).

How to zero a counter



Set "Load" to 1, to open the "1" line of the multiplexers.

How to zero a counter



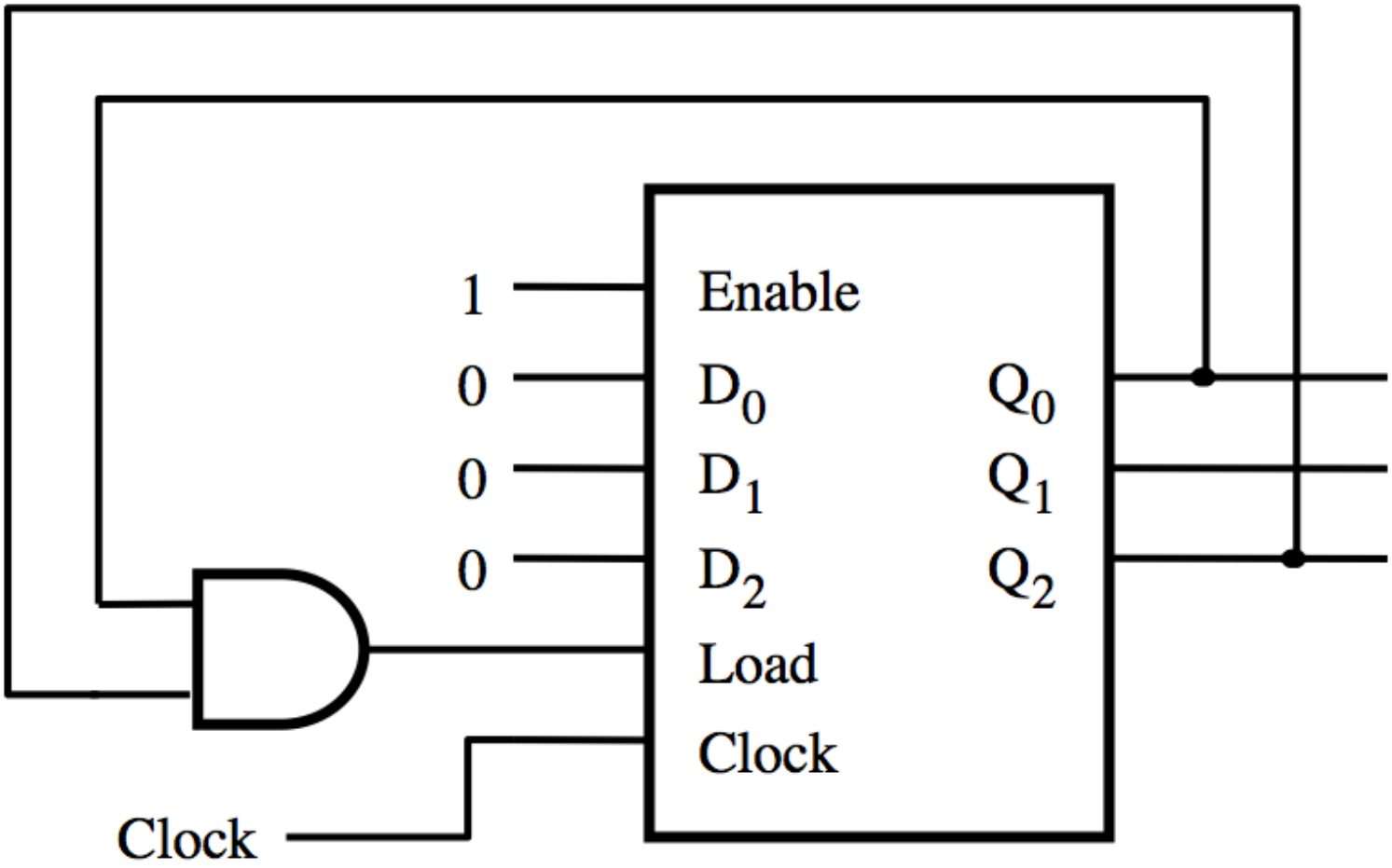
When the next positive edge of the clock arrives, the outputs of the flip-flops are updated.

Reset Synchronization

Motivation

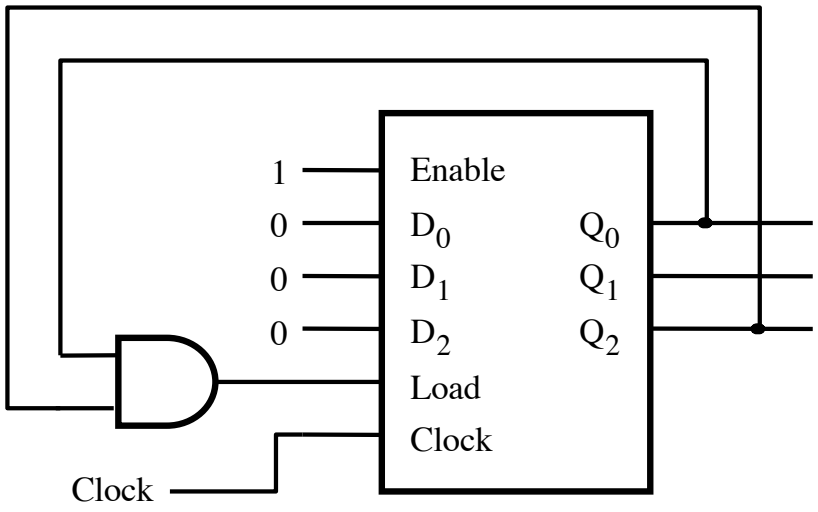
- An n-bit counter counts from 0, 1, …, 2^n-1
- For example a 3-bit counter counts up as follow
  - 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, …
- What if we want it to count like this
  - 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 0, 1, …
- In other words, what is the cycle is not a power of 2?

What does this circuit do?

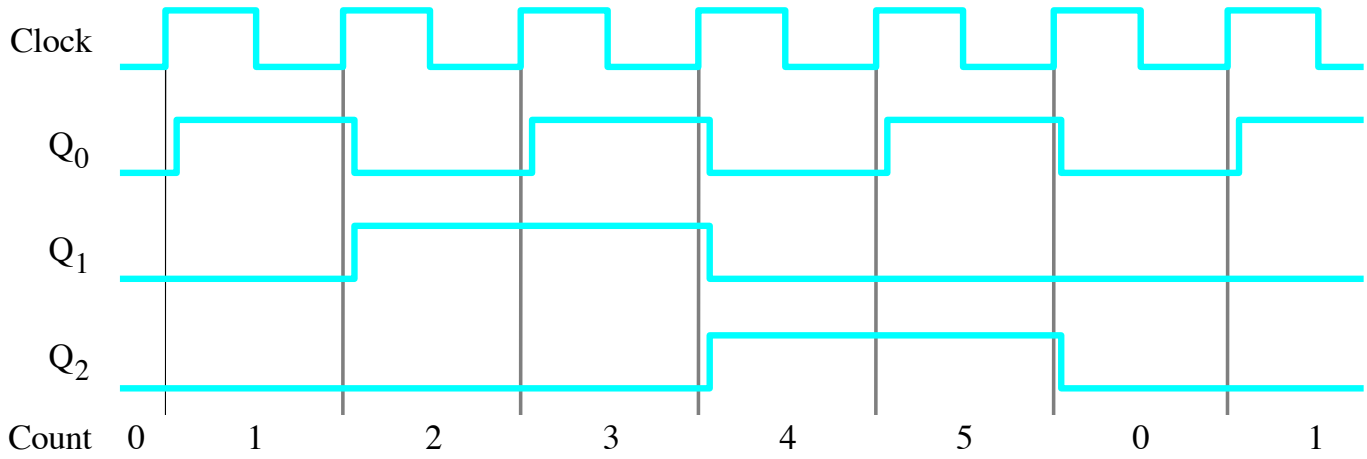


[Figure 5.25a from the textbook]

A modulo-6 counter with synchronous reset



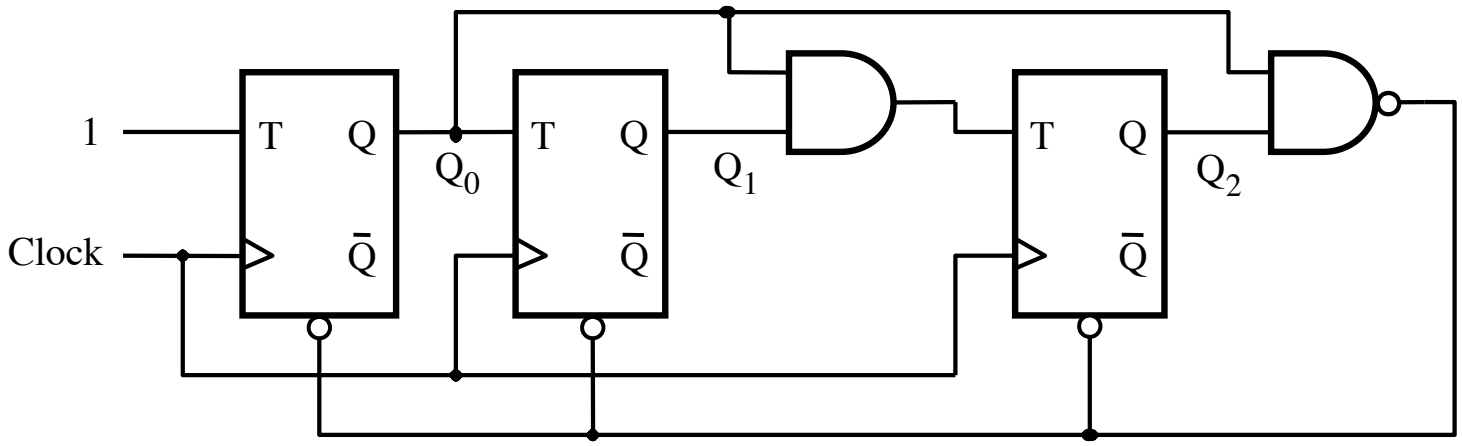
(a) Circuit



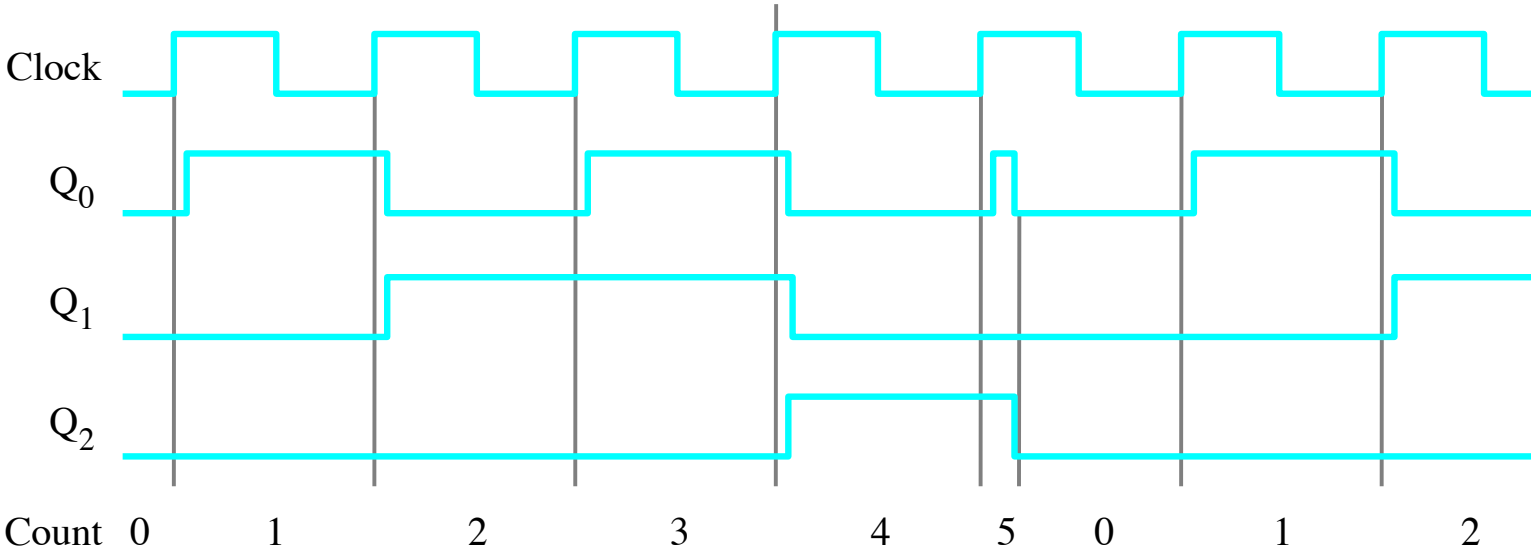
(b) Timing diagram

[Figure 5.25 from the textbook]

A modulo-6 counter with asynchronous reset



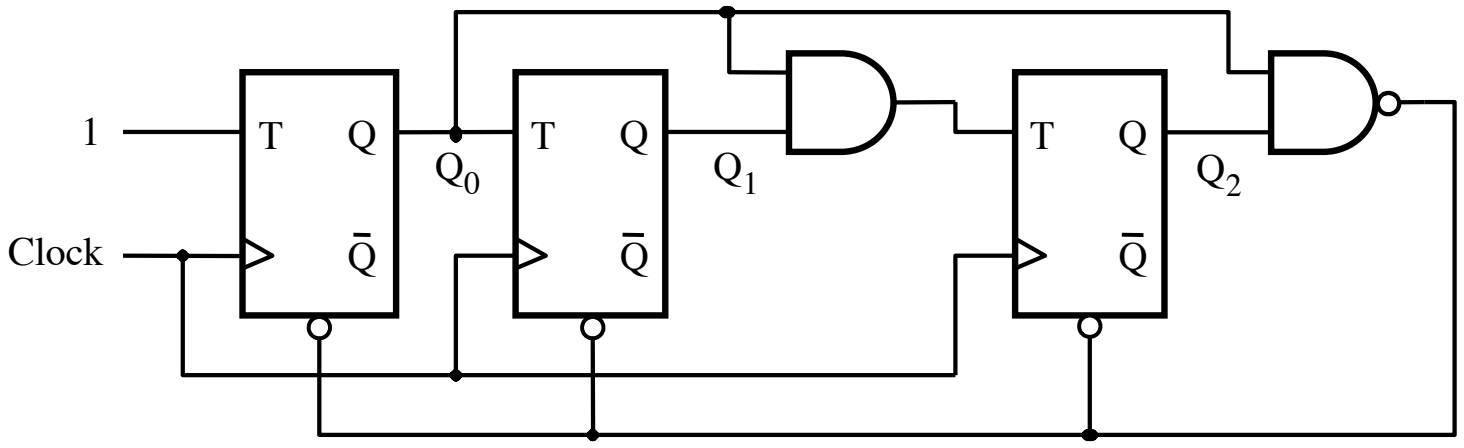
(a) Circuit



(b) Timing diagram

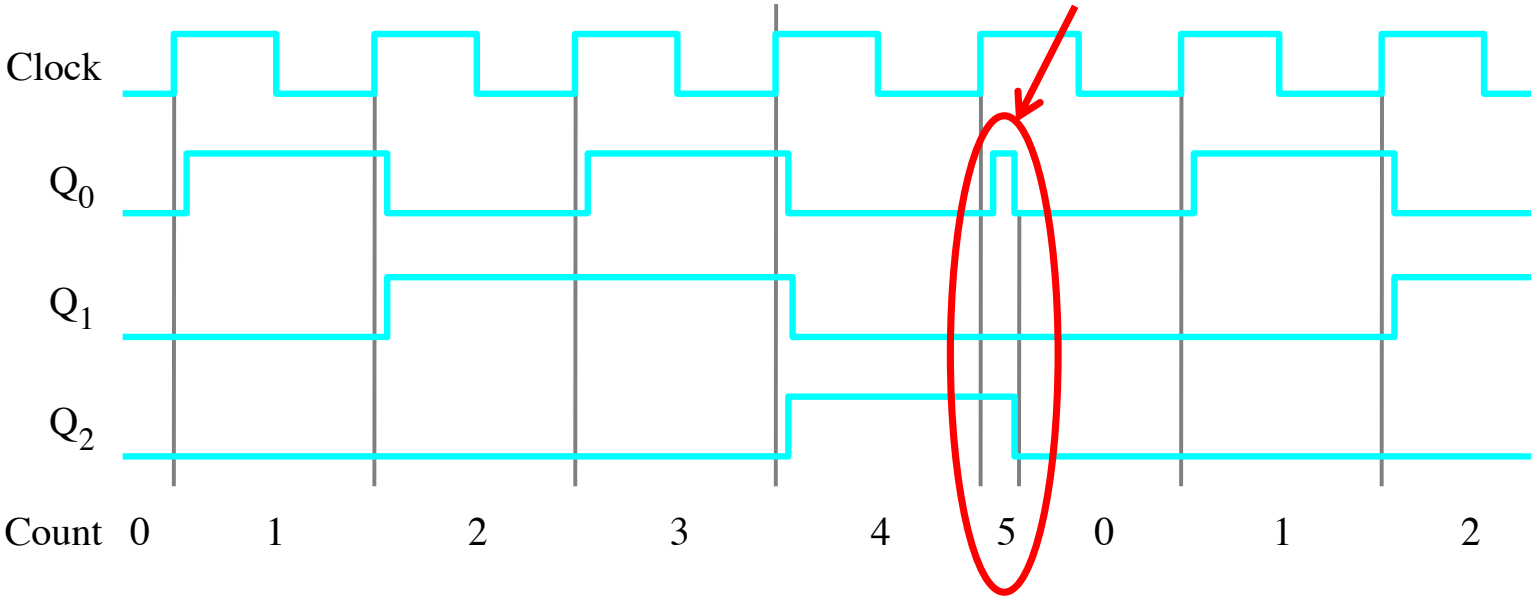
[Figure 5.26 from the textbook]

A modulo-6 counter with asynchronous reset



(a) Circuit

The number 5 is displayed
for a very short amount of time



(b) Timing diagram

[Figure 5.26 from the textbook]

Questions?

THE END