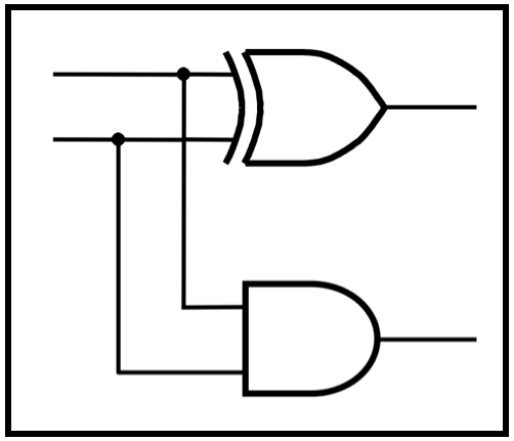


CprE 2810: Digital Logic

Instructor: Alexander Stoytchev

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

Register File

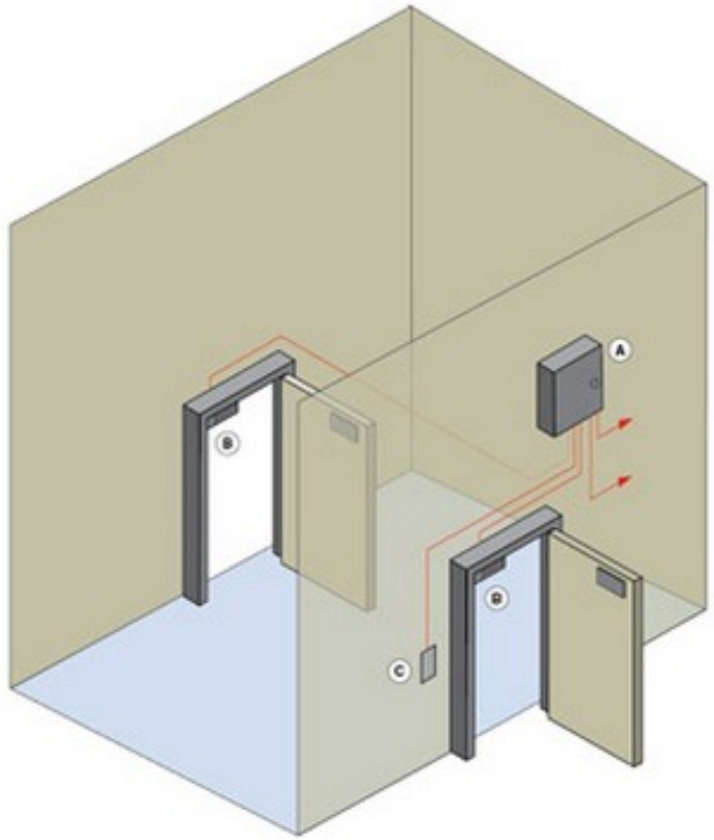
CprE 2810: Digital Logic
Iowa State University, Ames, IA

Quick Review

Flip-Flop Analogy

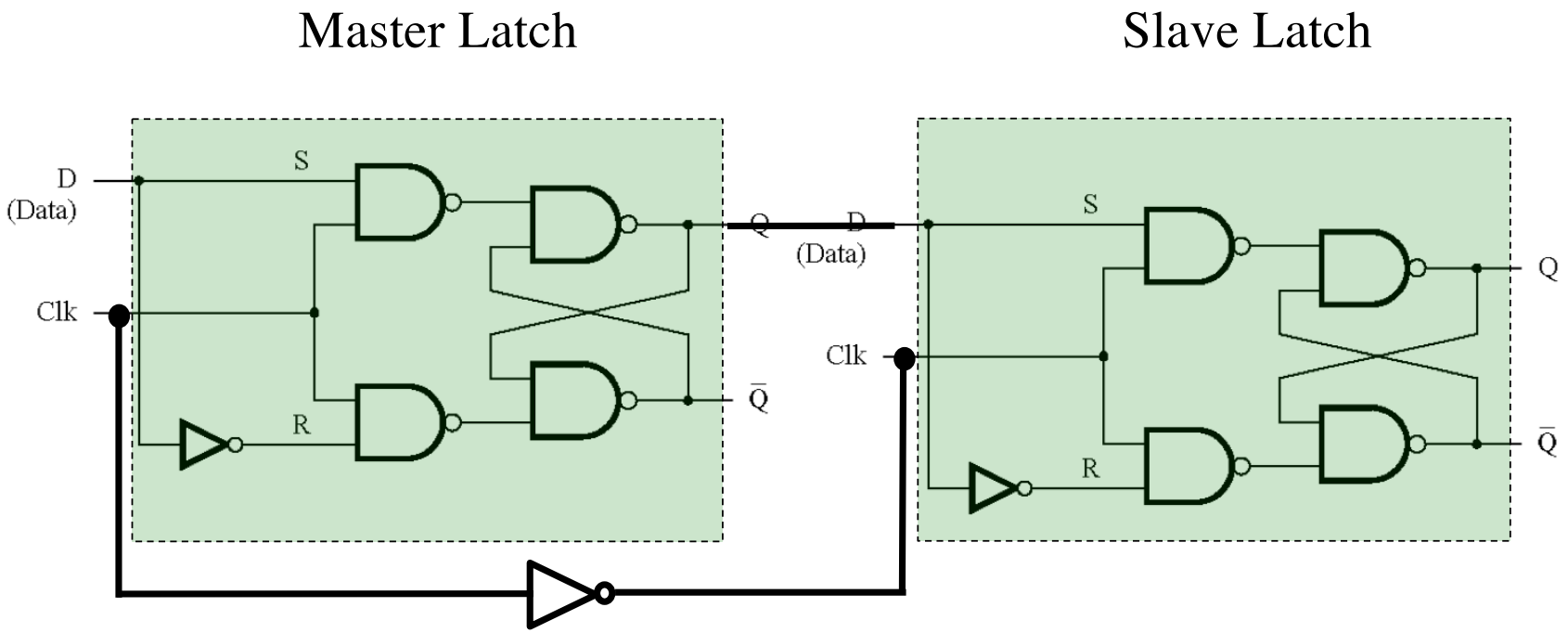
(double door)

D Flip-Flop Analogy

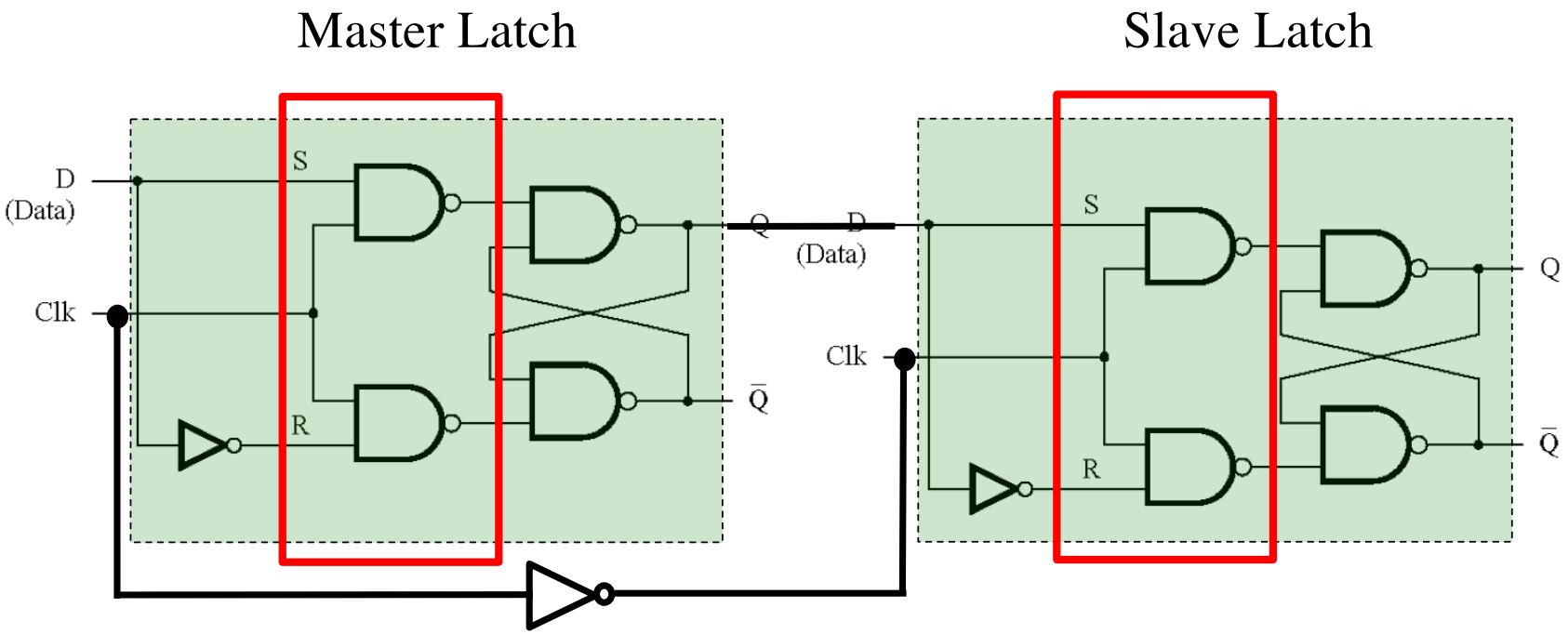


[https://www.maglocks.com/sdc-two-door-airlockmantrapcleanroom-level-1-kit.html]

D Flip-Flop Analogy



D Flip-Flop Analogy

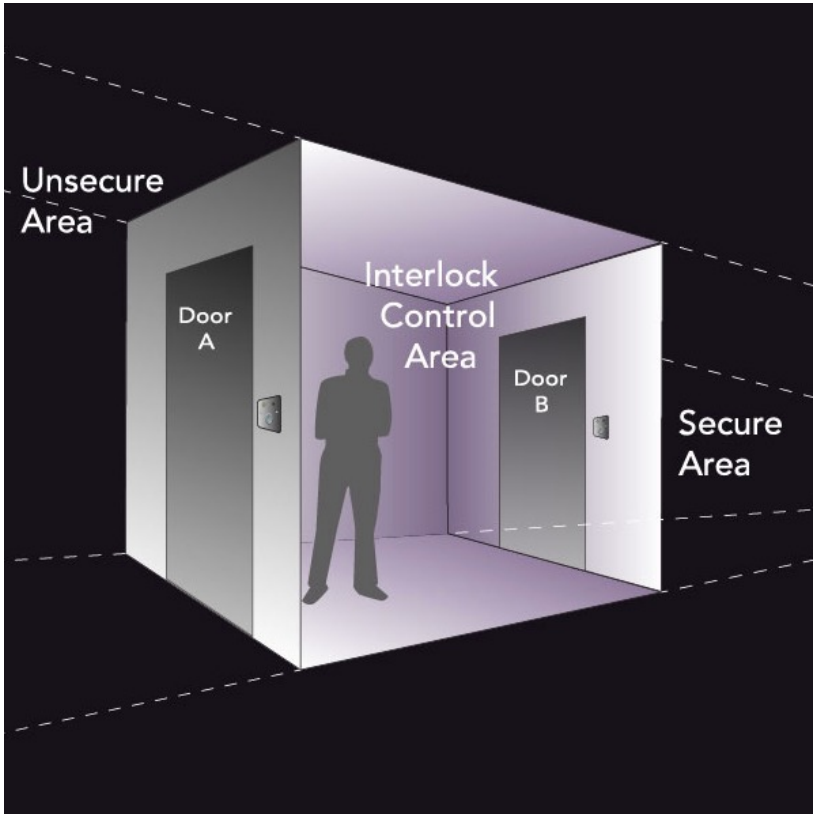


D Flip-Flop Analogy

Outer Door
Will Not Unlock When
Inner Door is Open

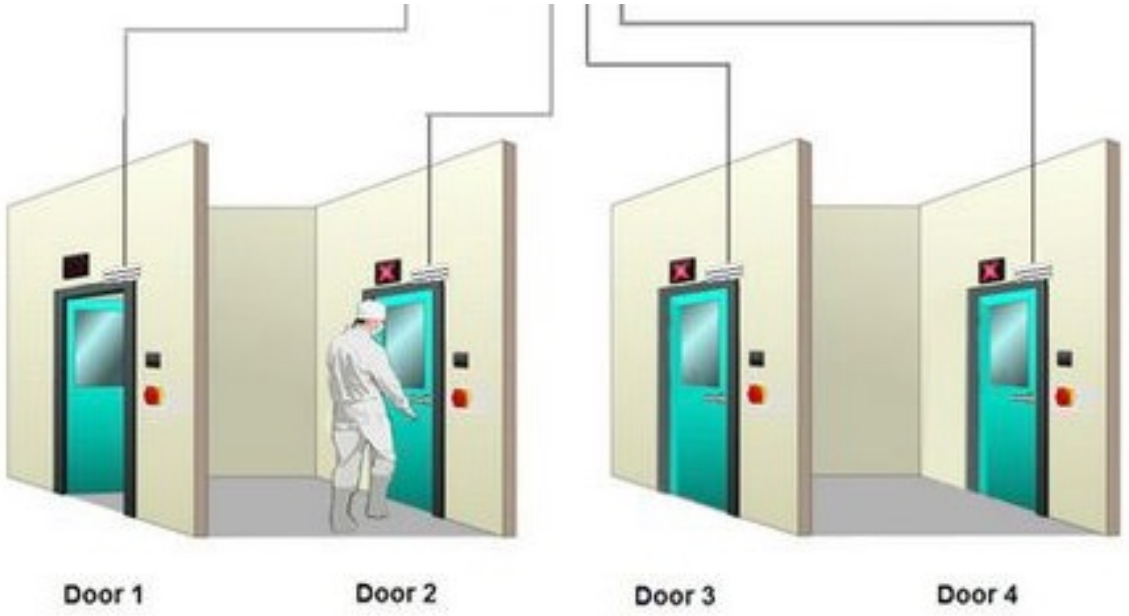
Inner Door
Will Not Unlock When
OuterDoor is Open

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



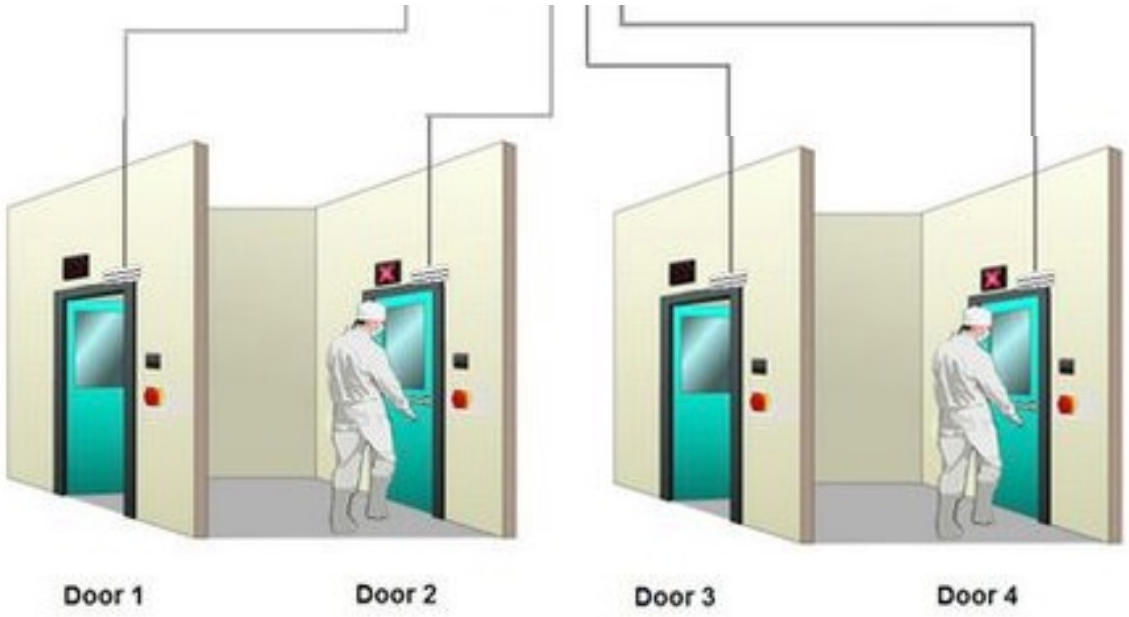
[https://www.insafe.com/activa-pro-interlock/]

Shift-Register Analogy



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

Shift-Register Analogy

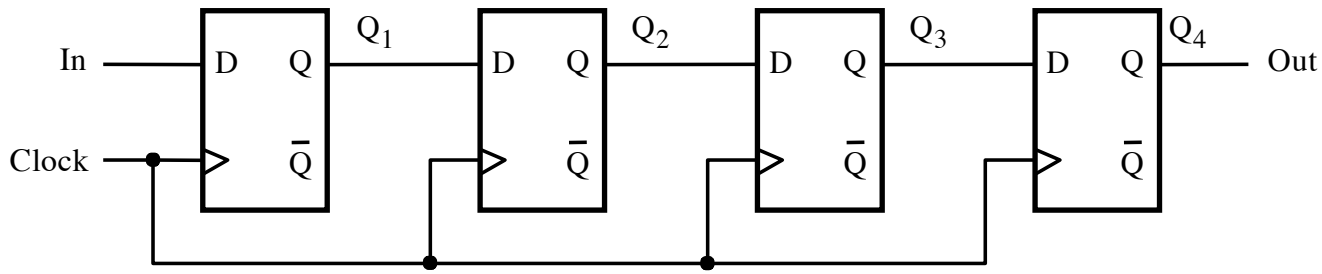


Doors 1 and 3 are always
in the same configuration

Doors 2 and 4 are always
in the same configuration

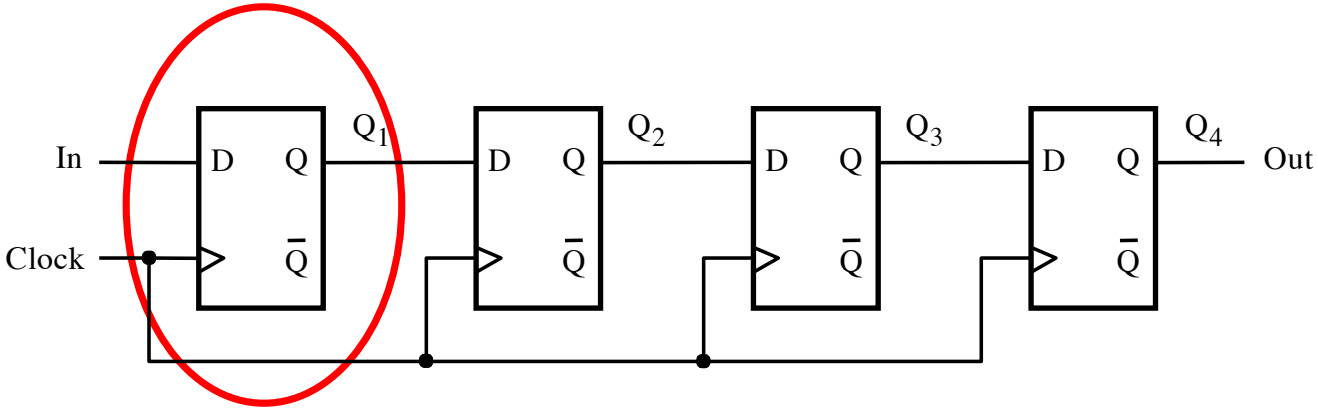
Shift Register

A simple shift register



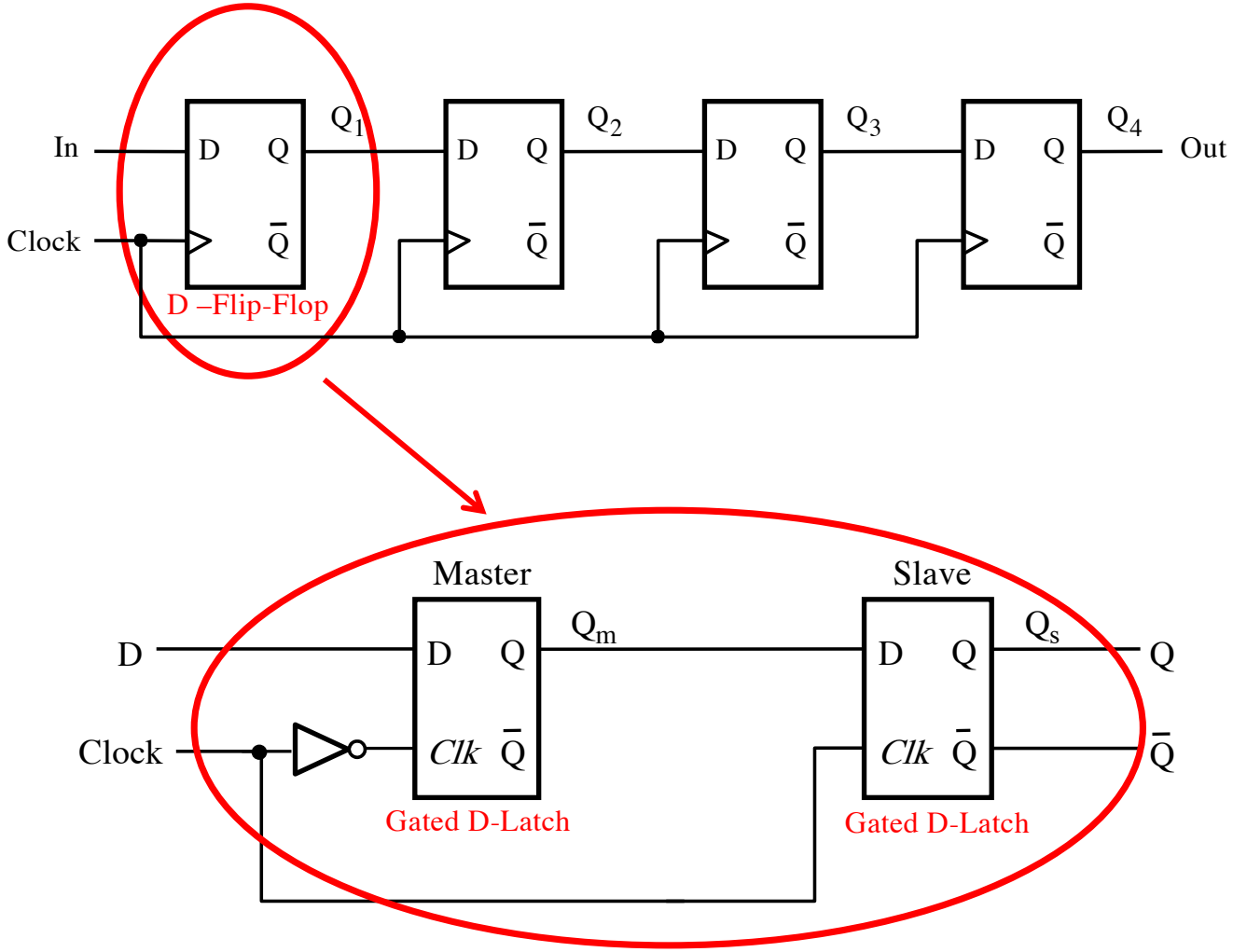
[Figure 5.17a from the textbook]

A simple shift register



Positive-edge-triggered
D Flip-Flop

A simple shift register



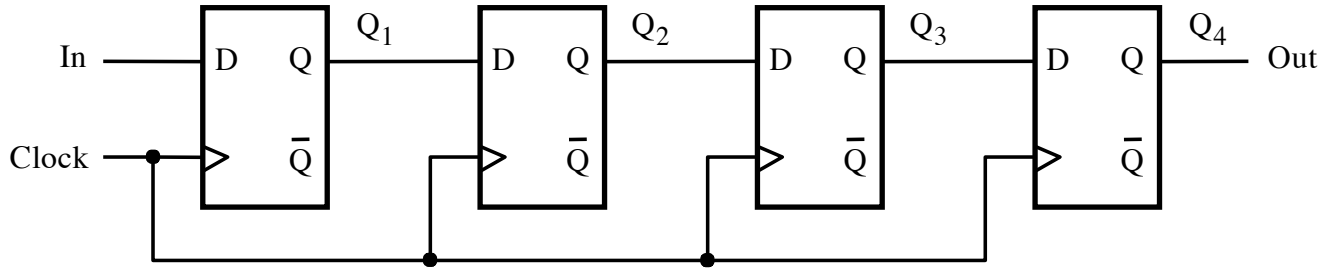
You need 2 latches to make a flip-flop

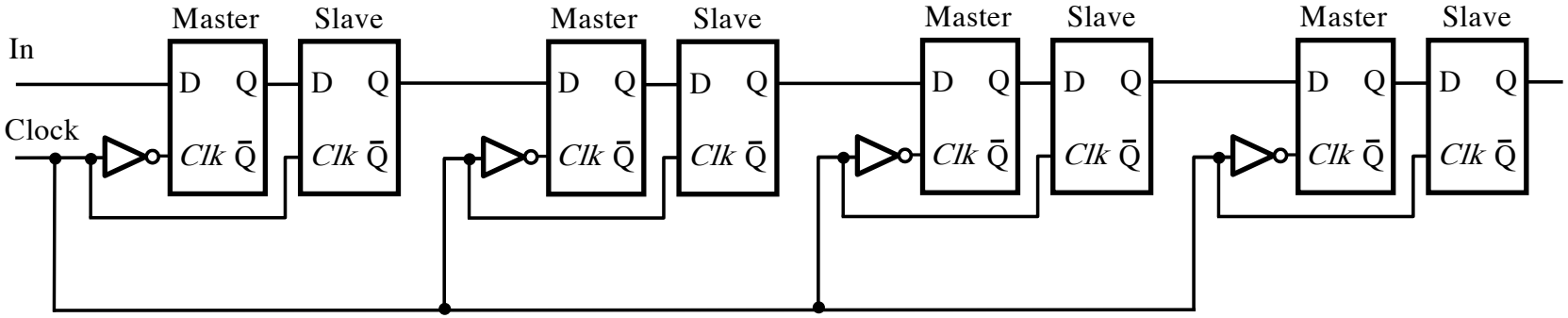
=

4-bit Register

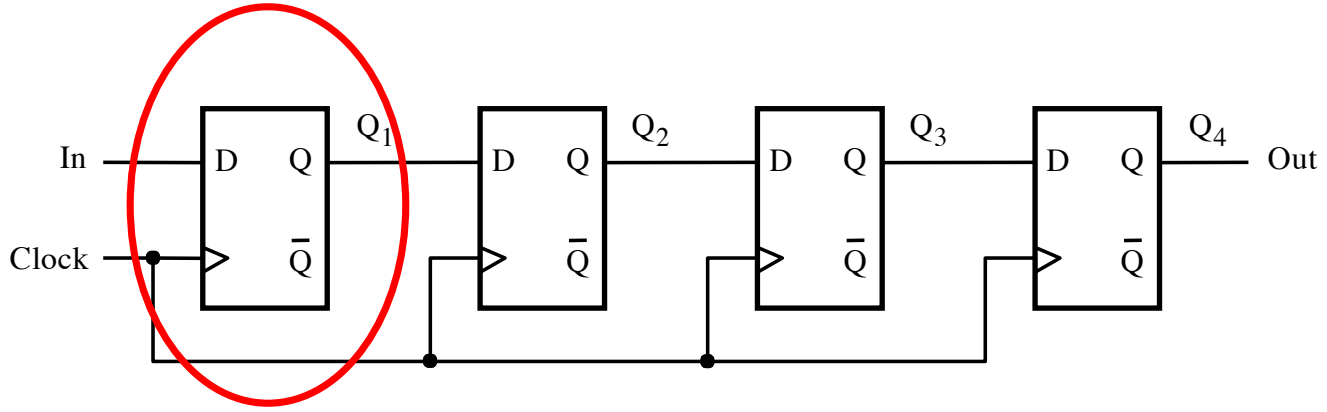
4-bit Register

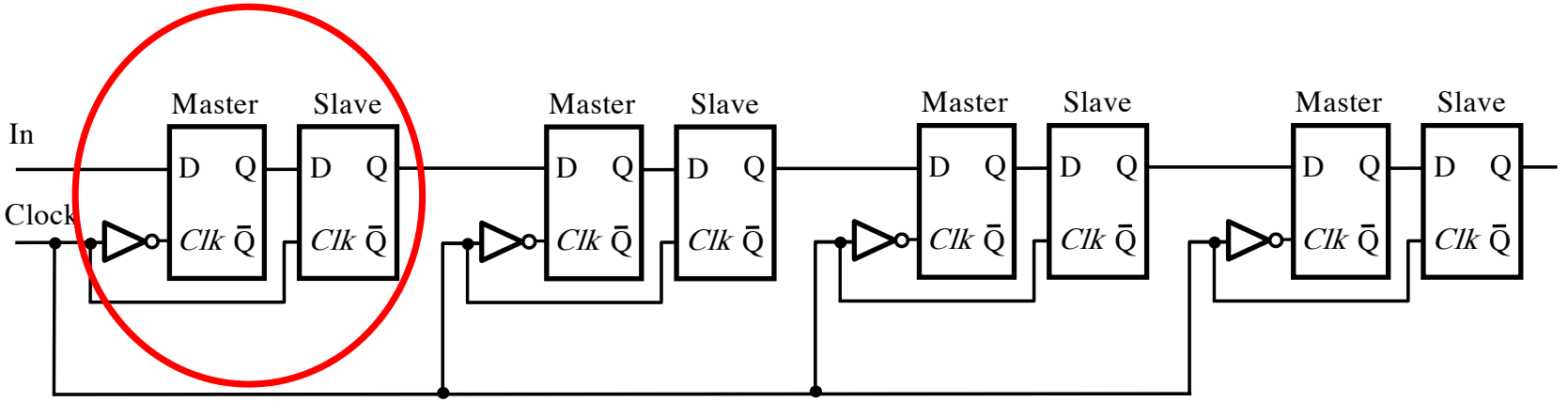
A simple shift register



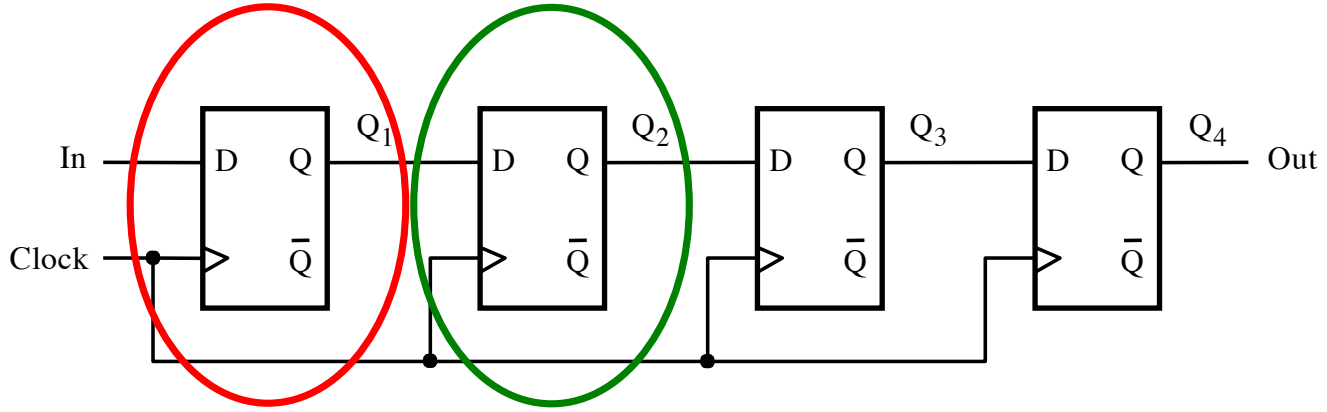


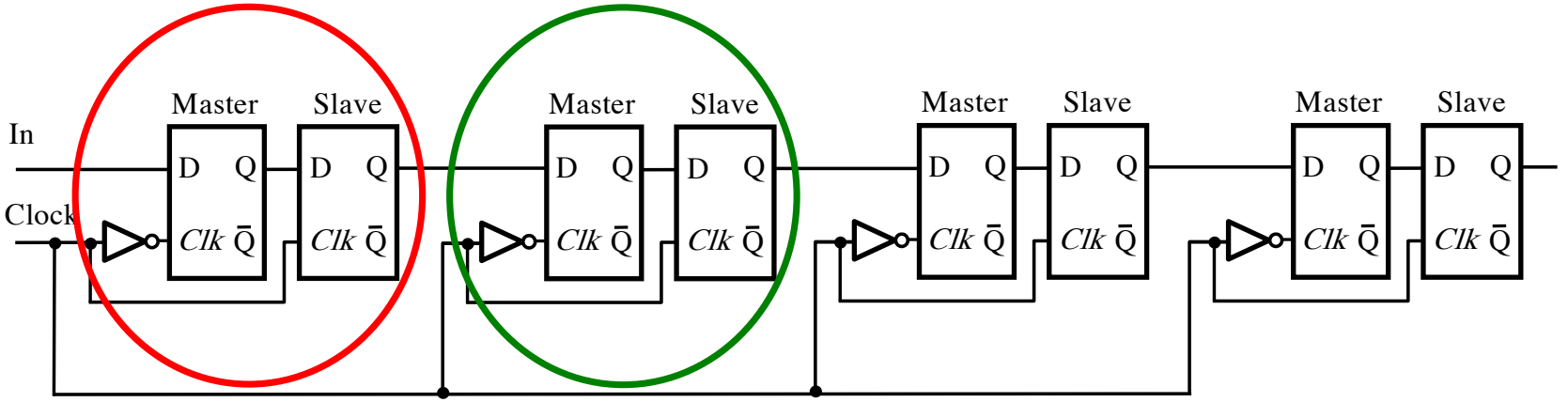
A simple shift register



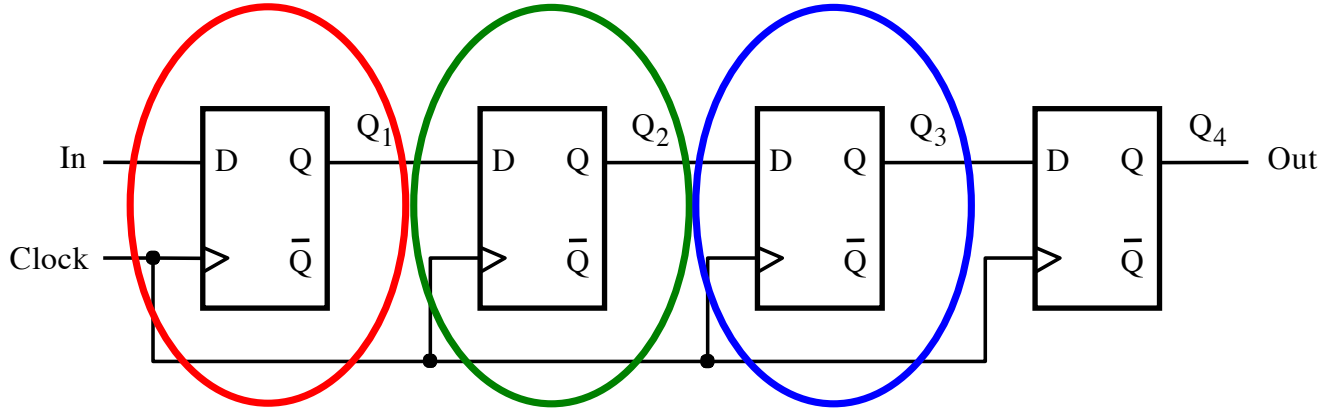


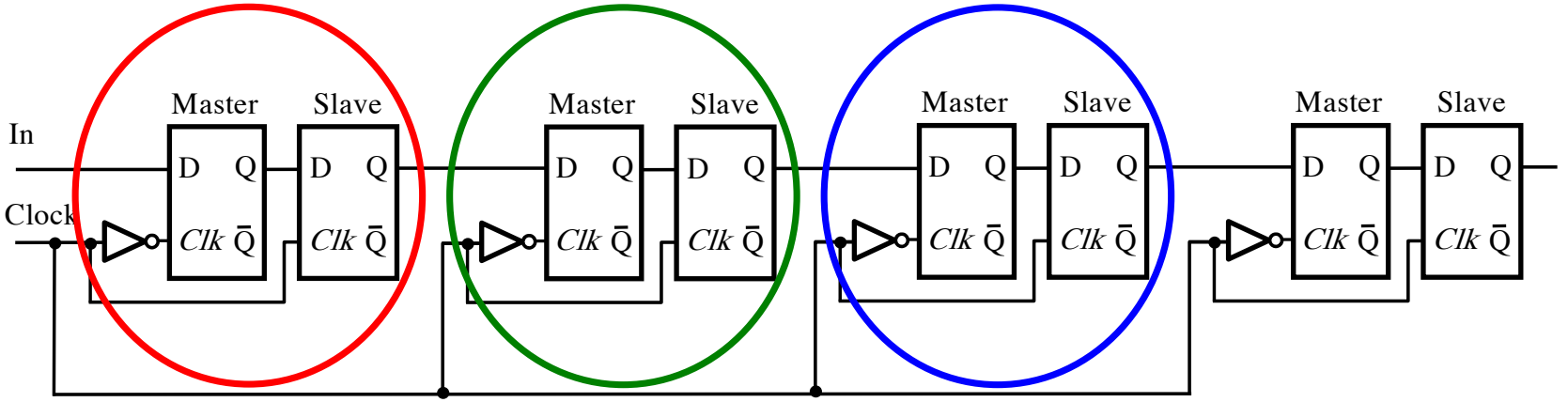
A simple shift register



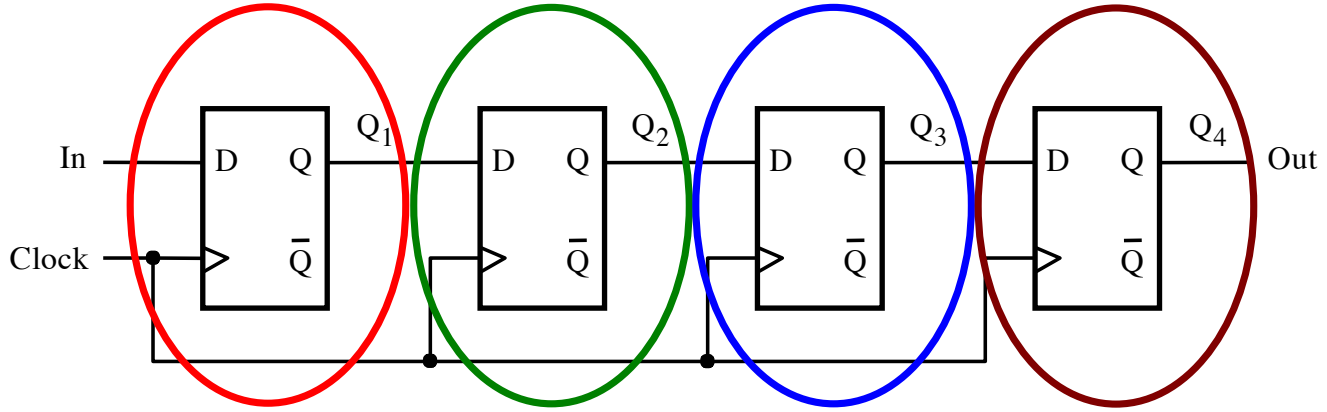


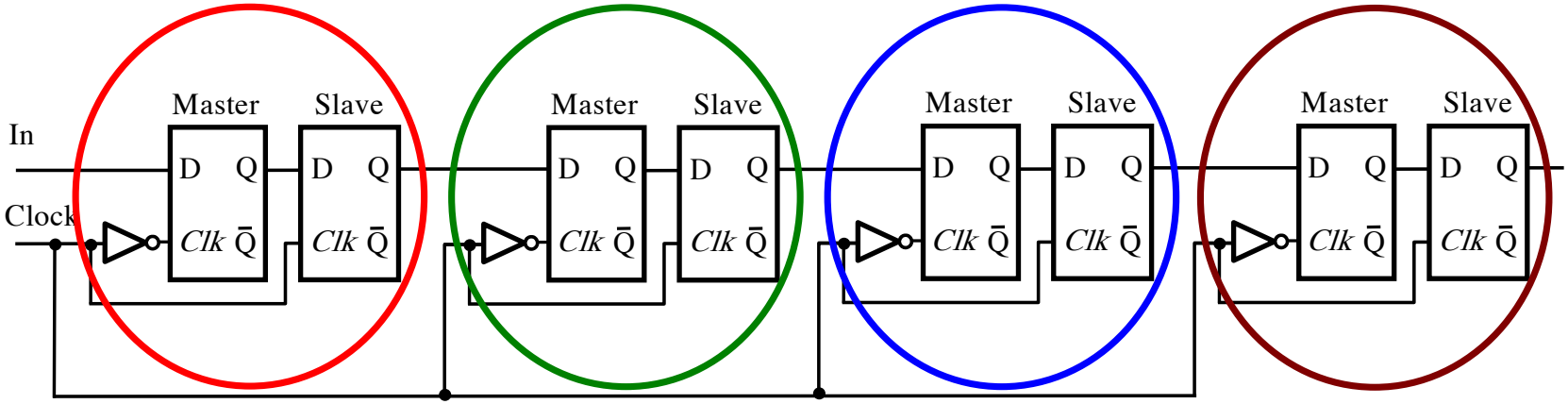
A simple shift register



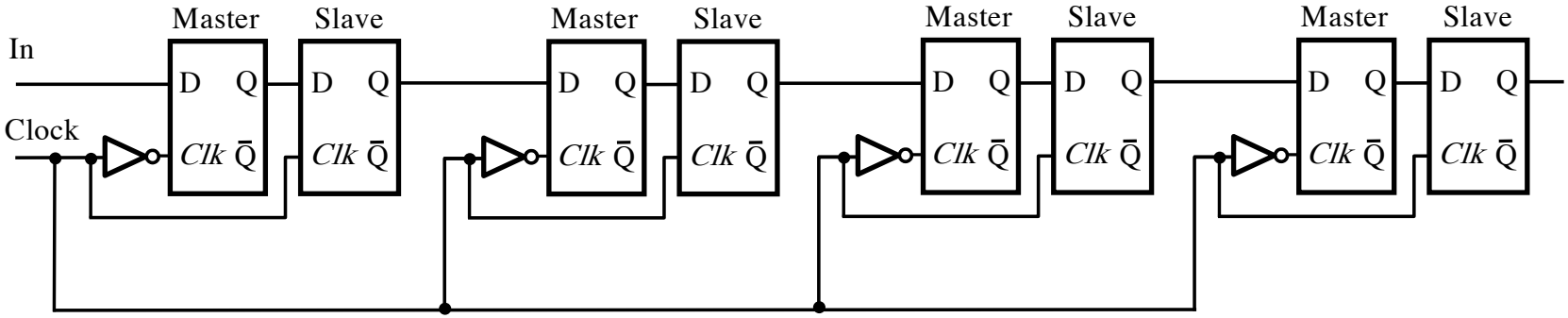


A simple shift register

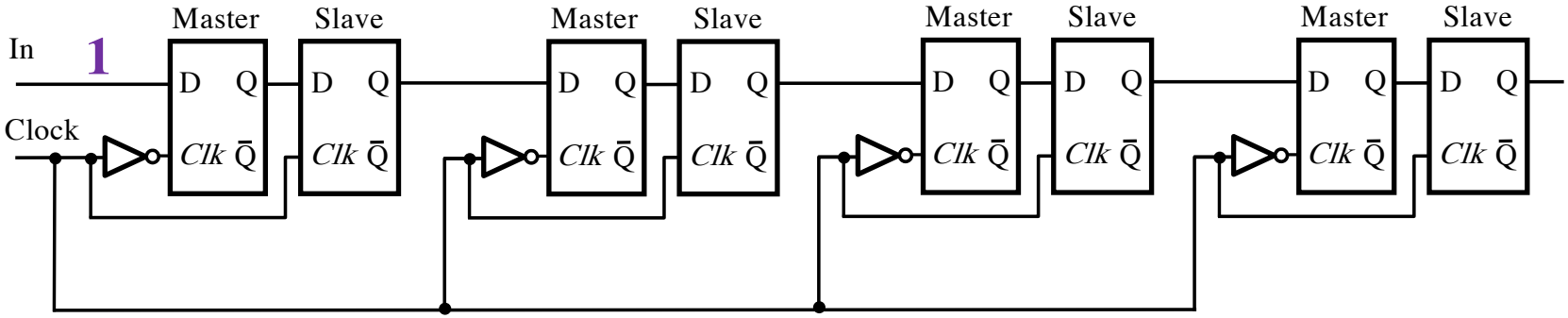




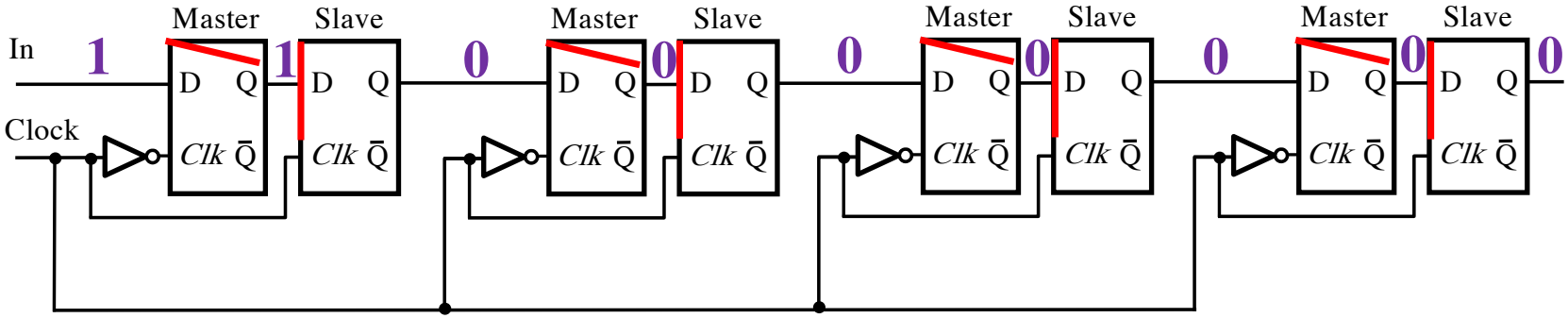
Simulating a shift register

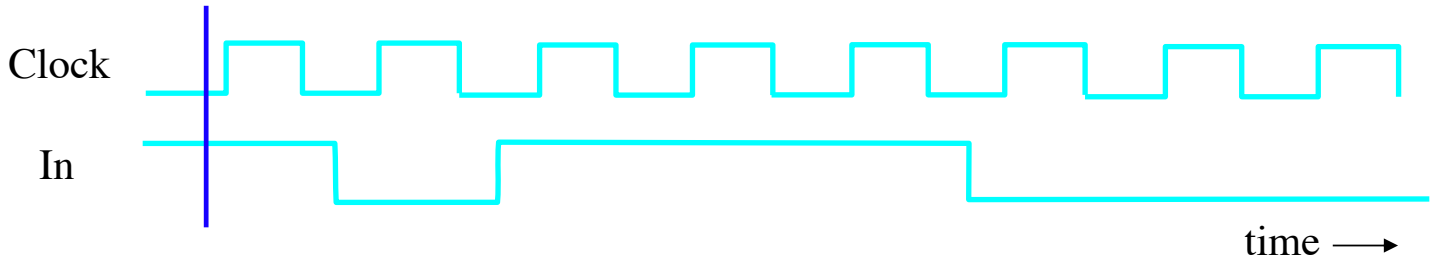


Simulating a shift register

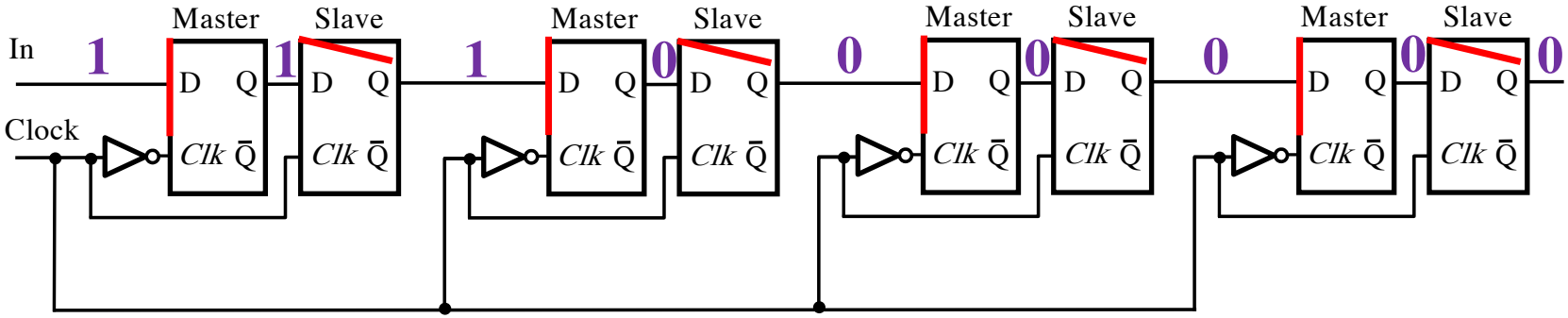


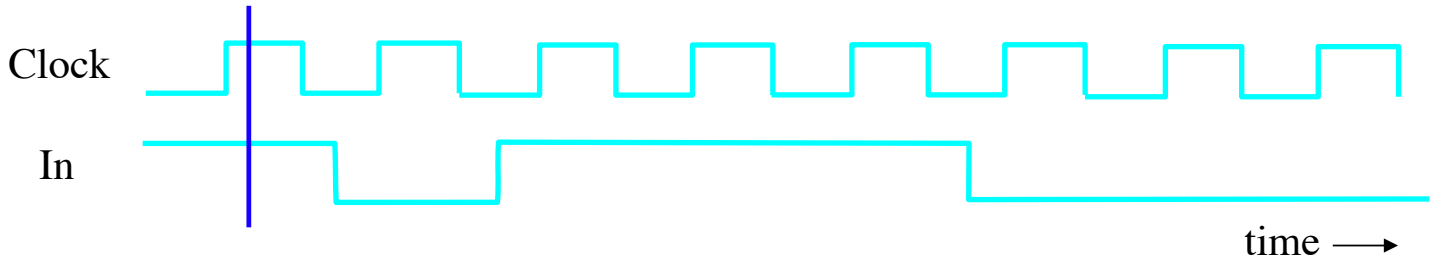
Simulating a shift register



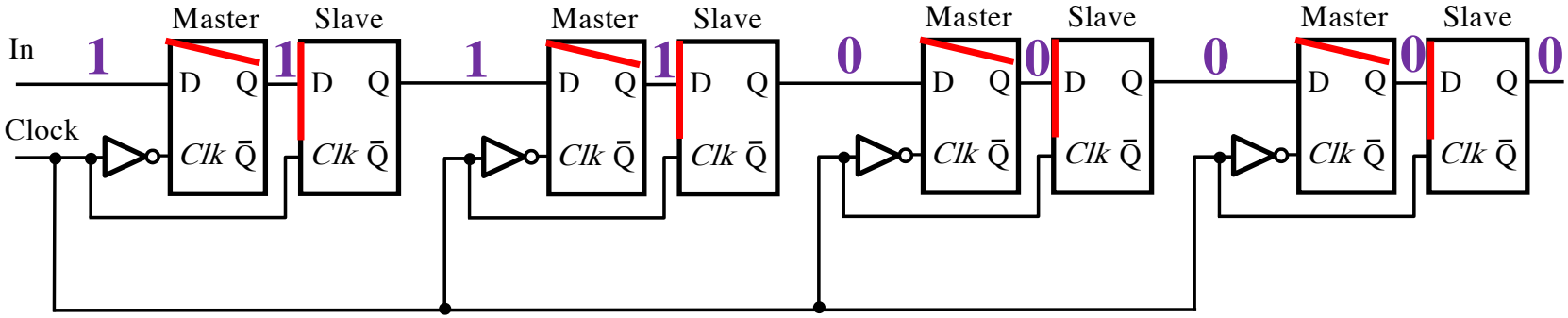


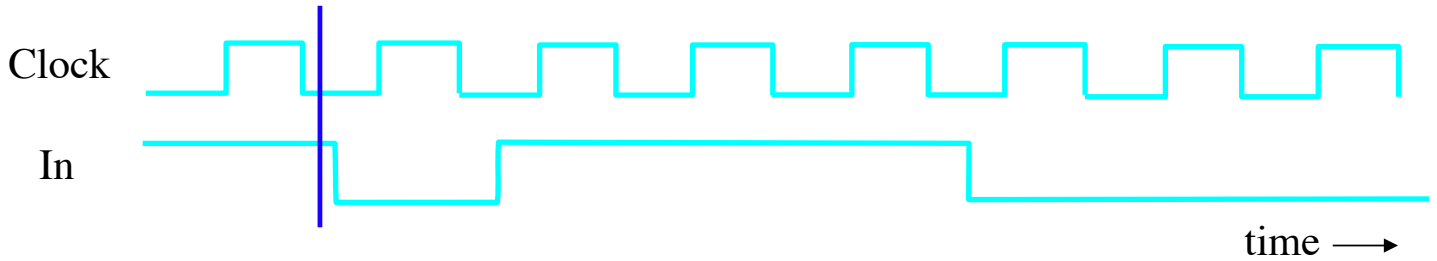
Simulating a shift register



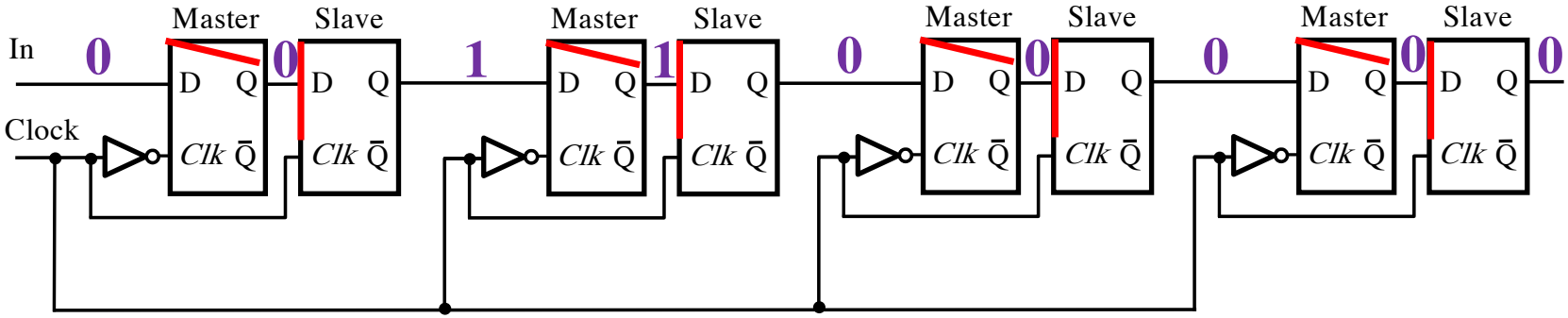


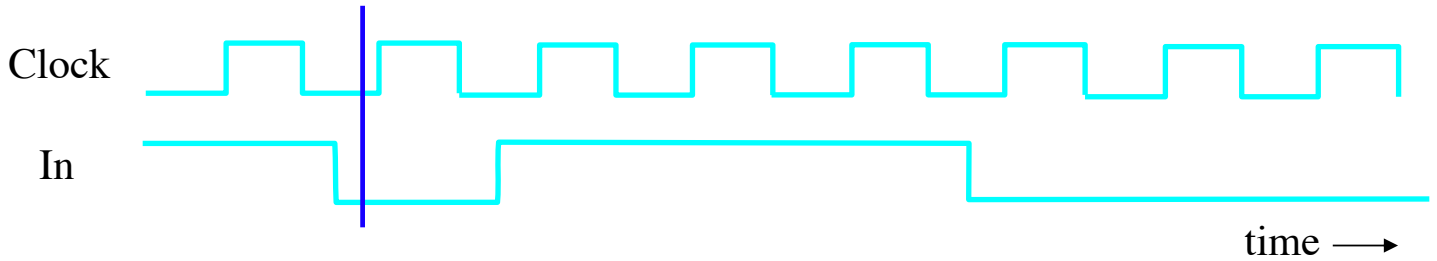
Simulating a shift register



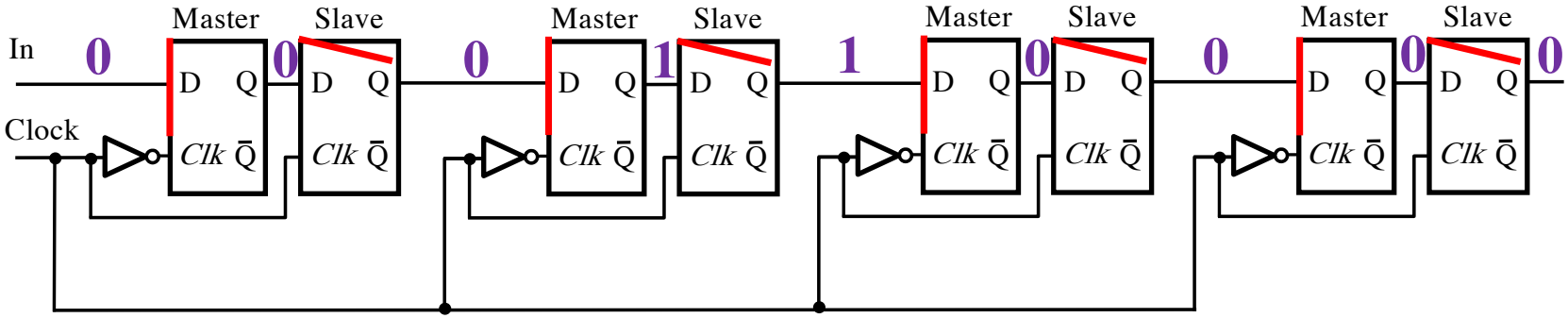


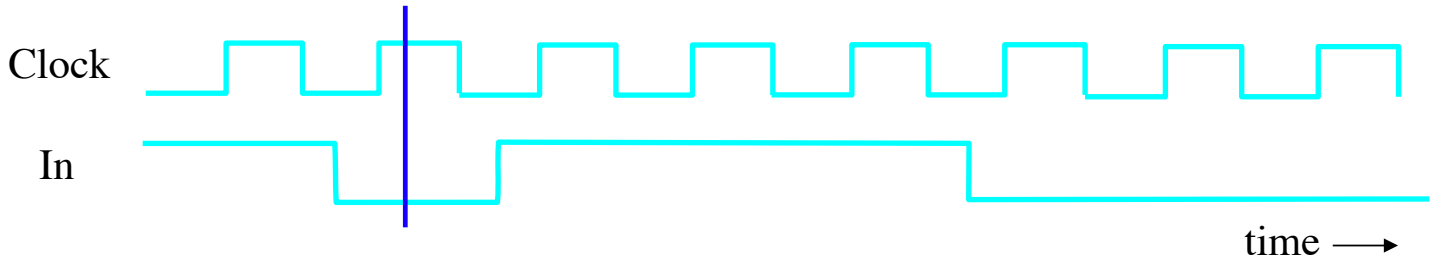
Simulating a shift register



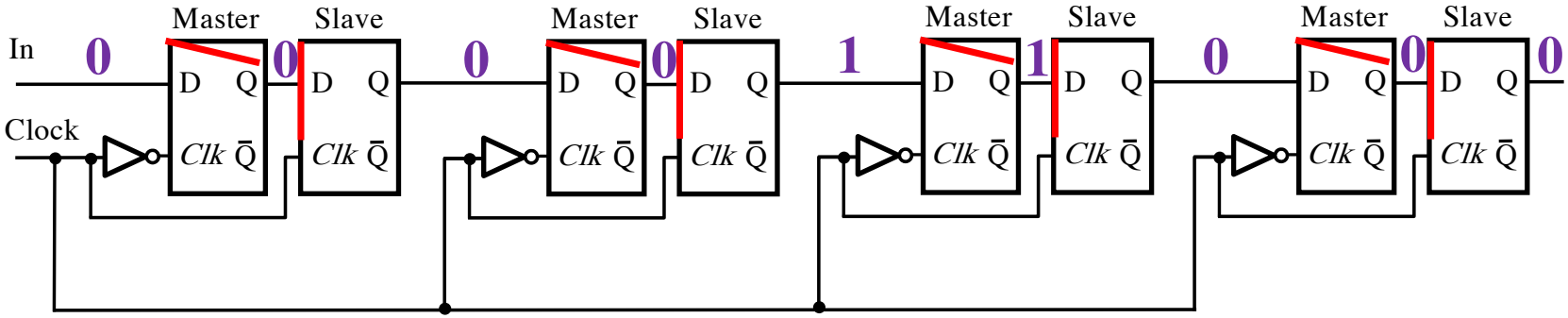


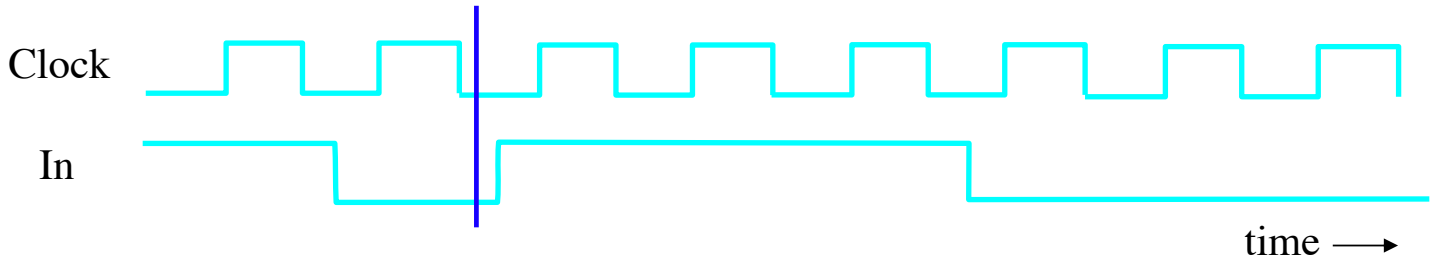
Simulating a shift register



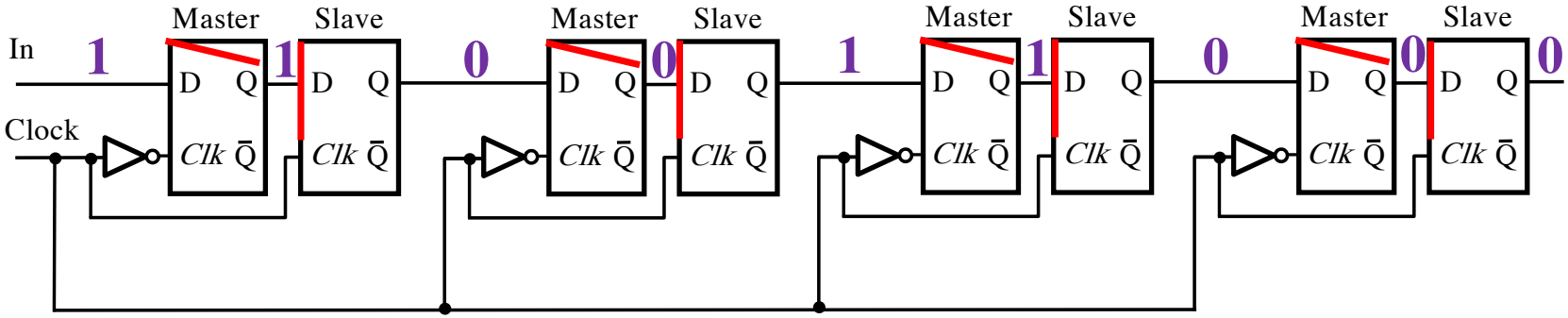


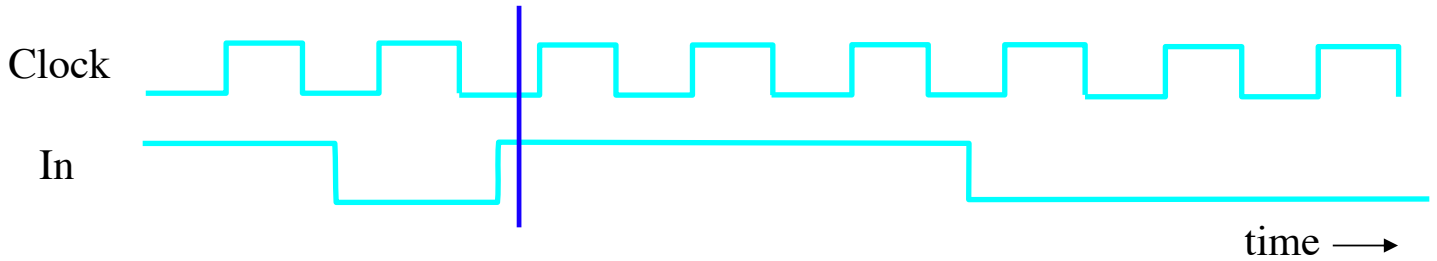
Simulating a shift register



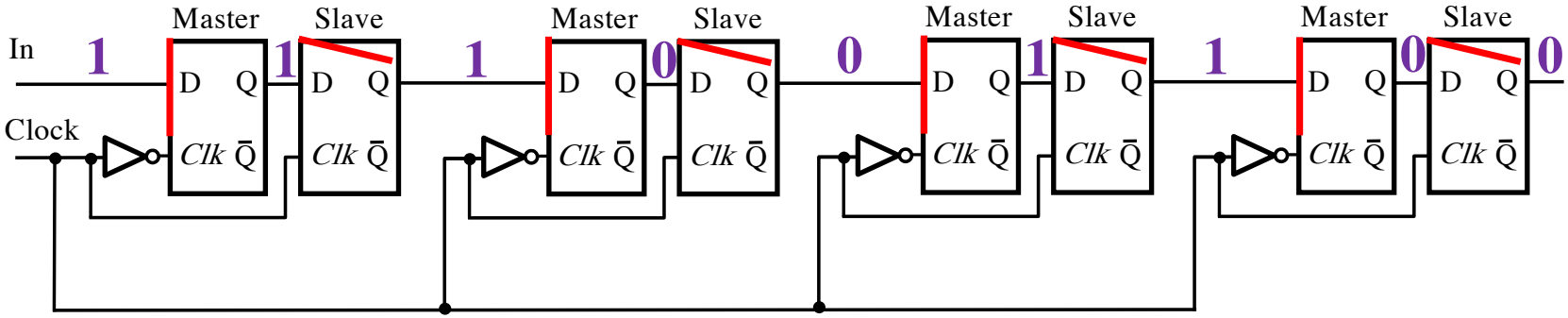


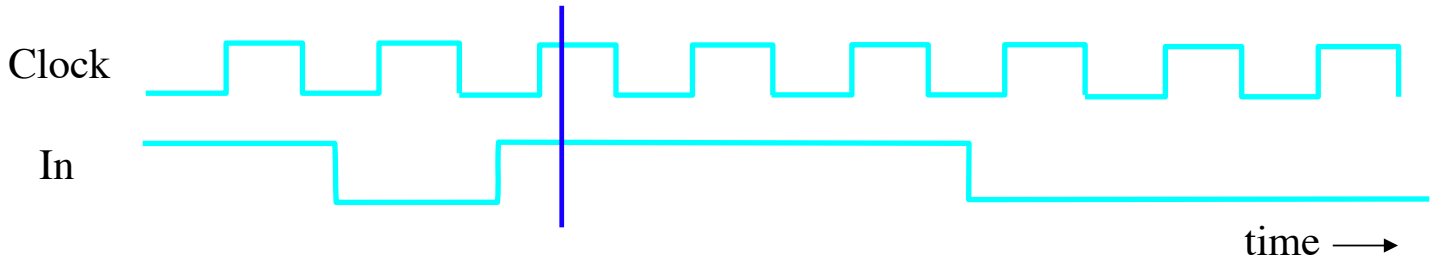
Simulating a shift register



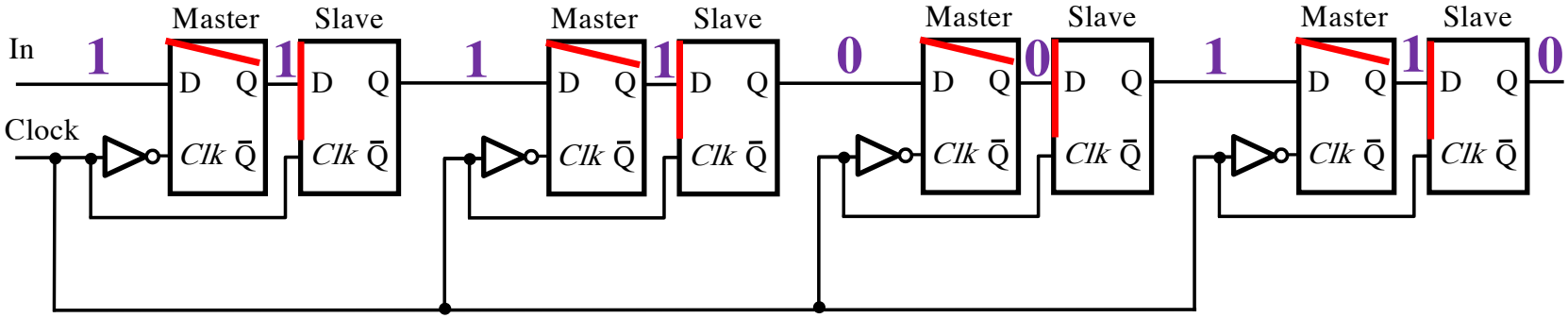


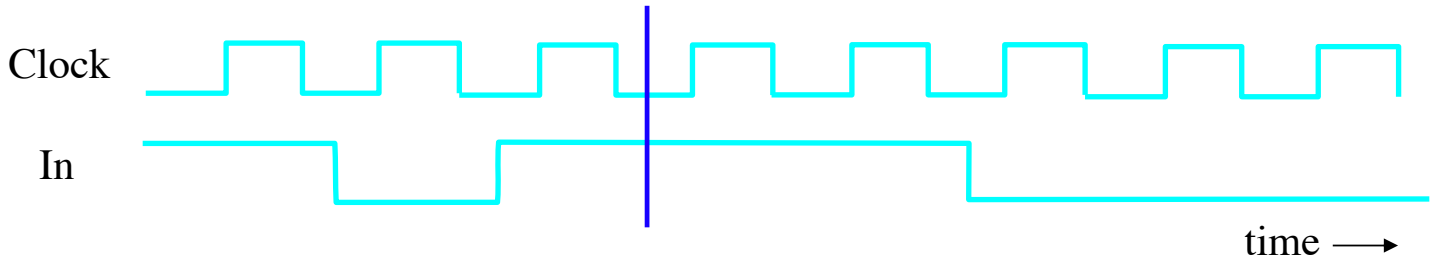
Simulating a shift register



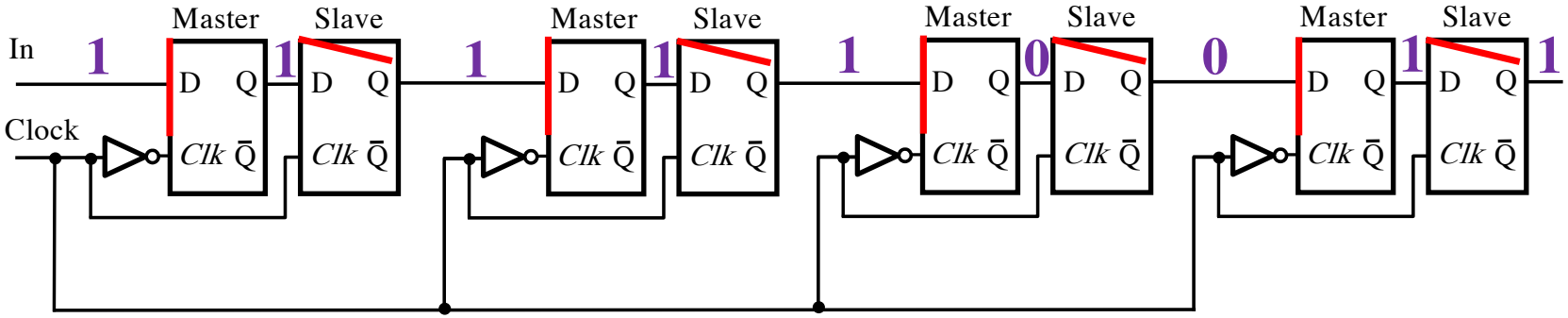


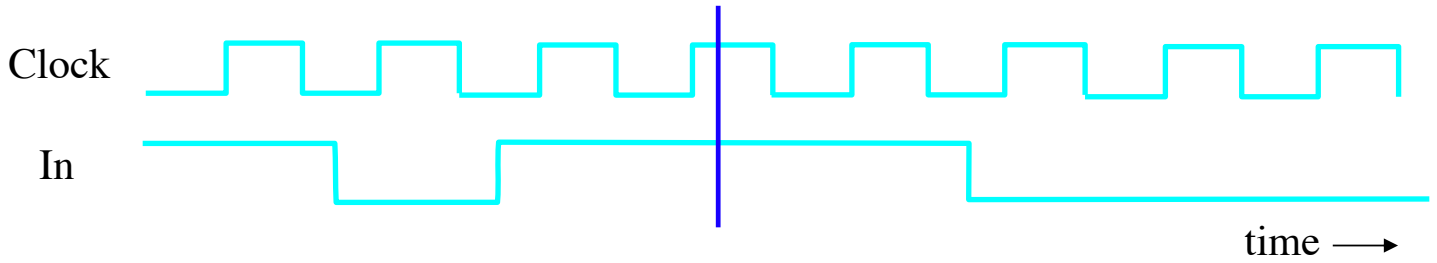
Simulating a shift register



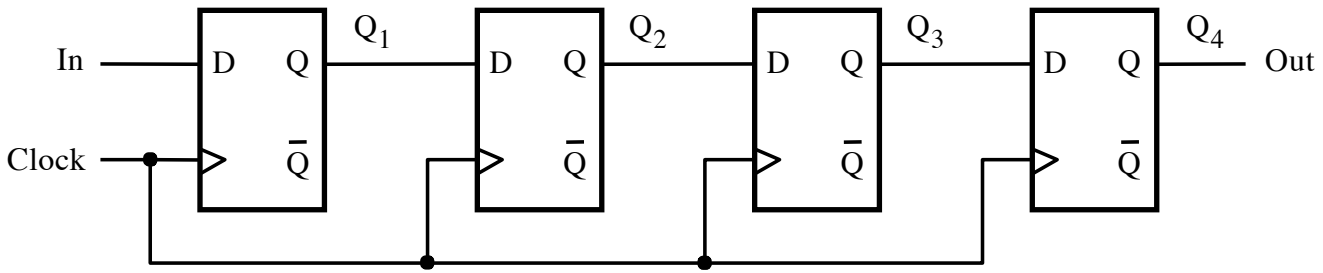


Simulating a shift register





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

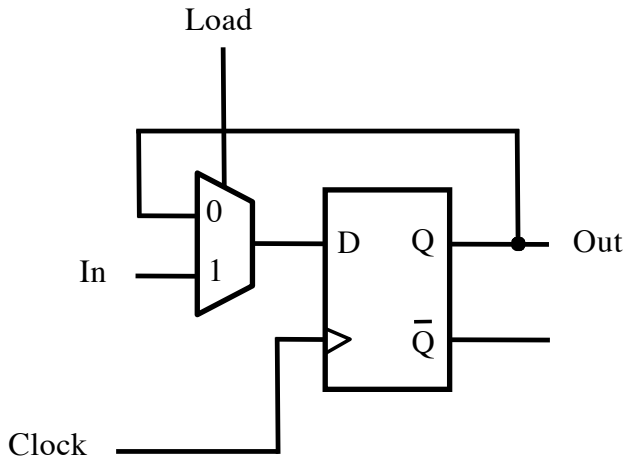
(b) A sample sequence

This simulation goes only up to here

[Figure 5.17 from the textbook]

Parallel-Access Register

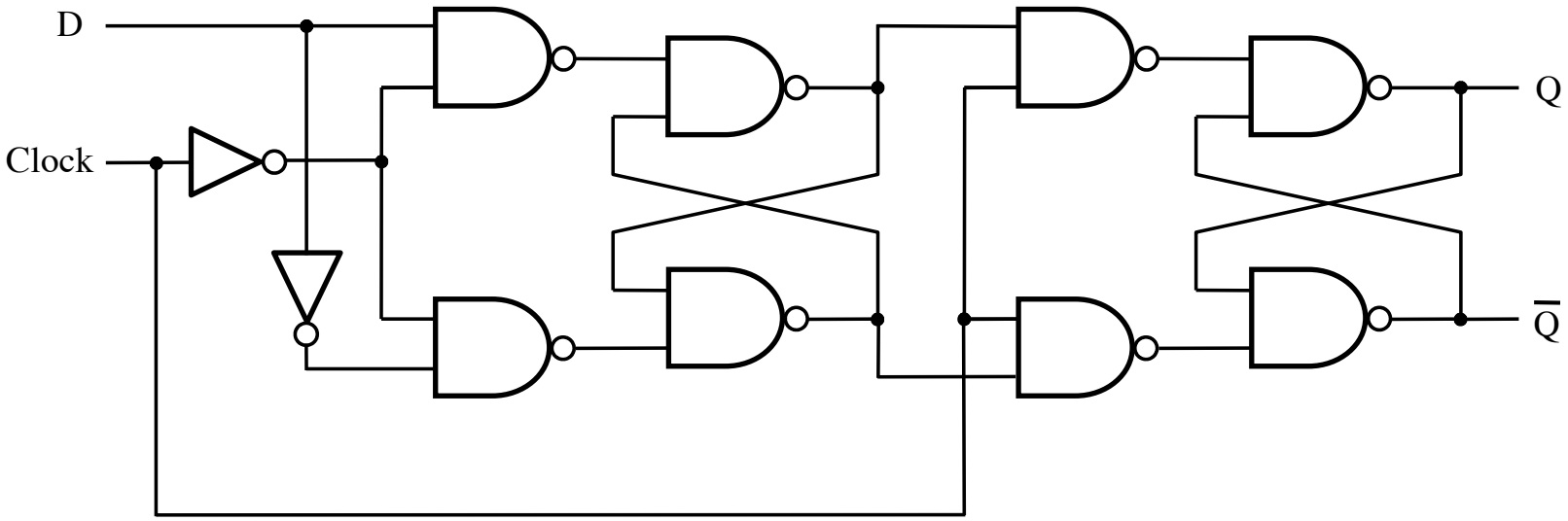
1-Bit Parallel-Access Register



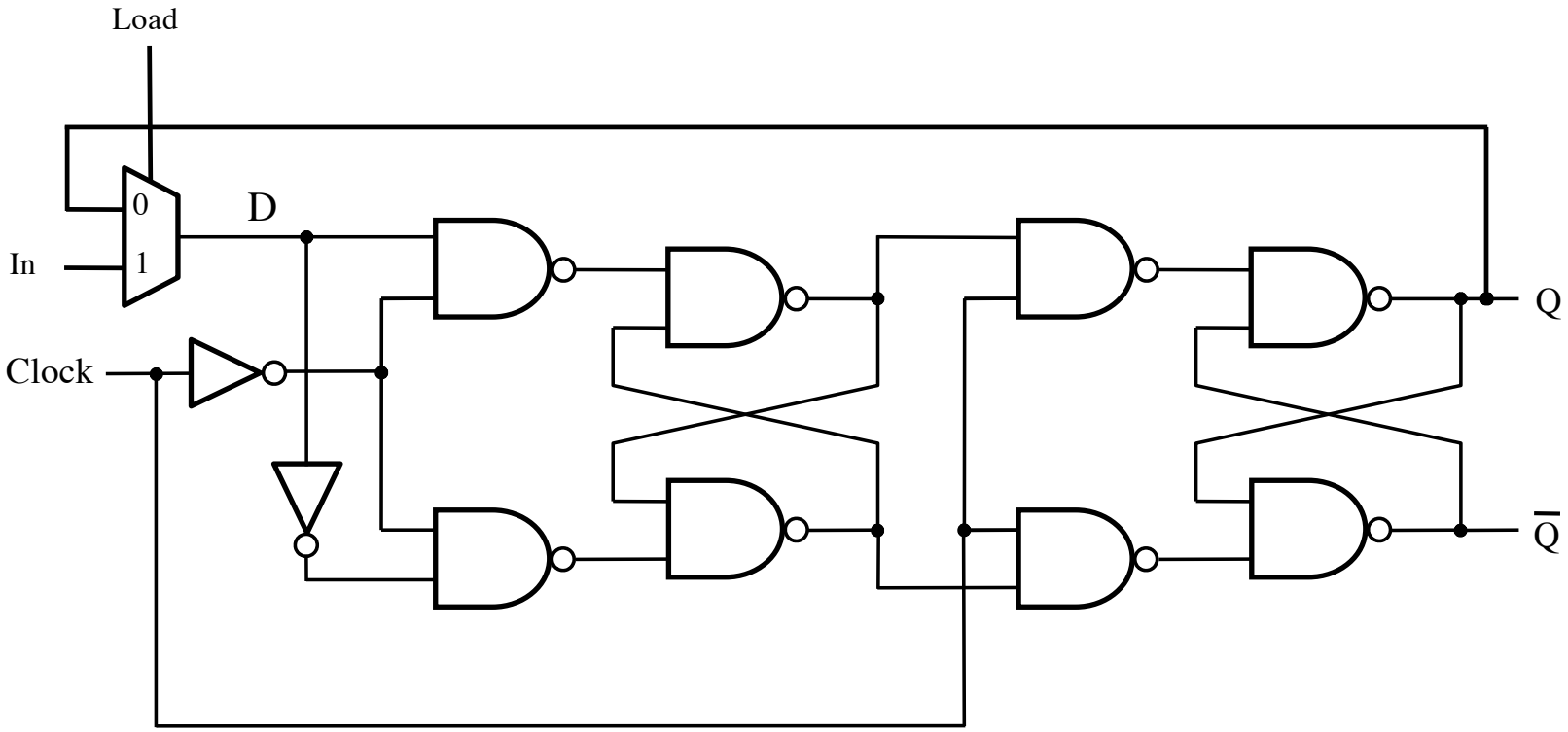
The 2-to-1 multiplexer is used to select whether to load a new value into the D flip-flop or to retain the old value.

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

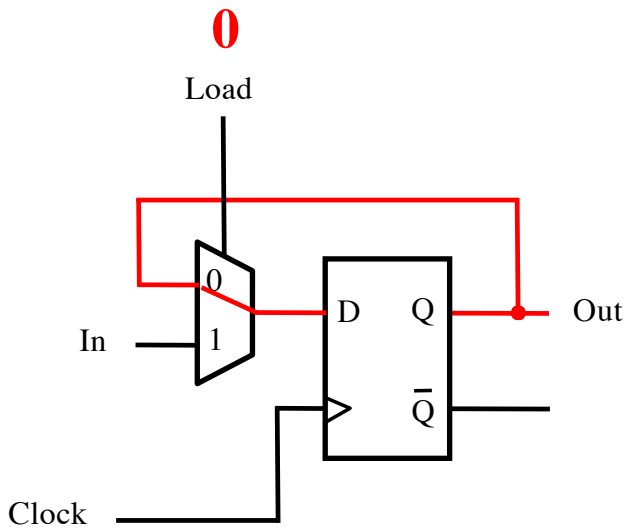
Positive-Edge-Triggered D Flip-Flop



1-bit Parallel-Access Register

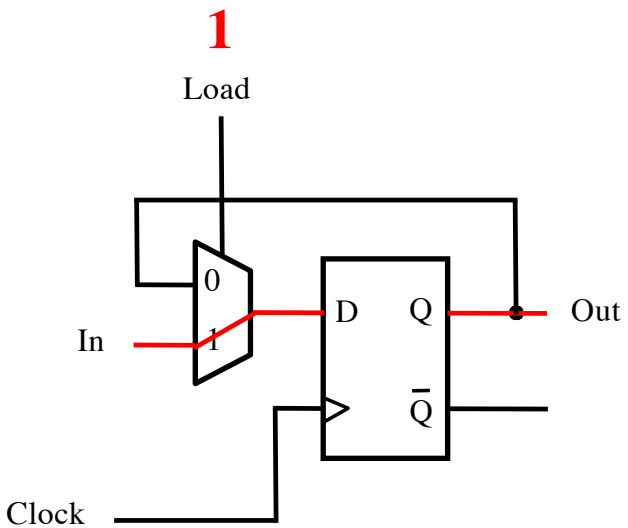


1-Bit Parallel-Access Register



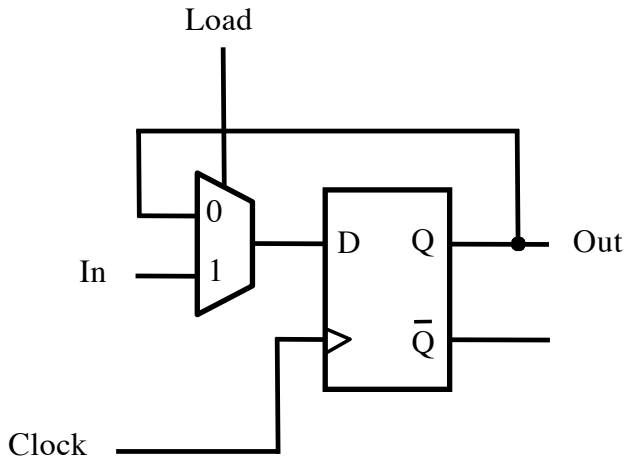
If Load = 0, then retain the old value.

1-Bit Parallel-Access Register



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

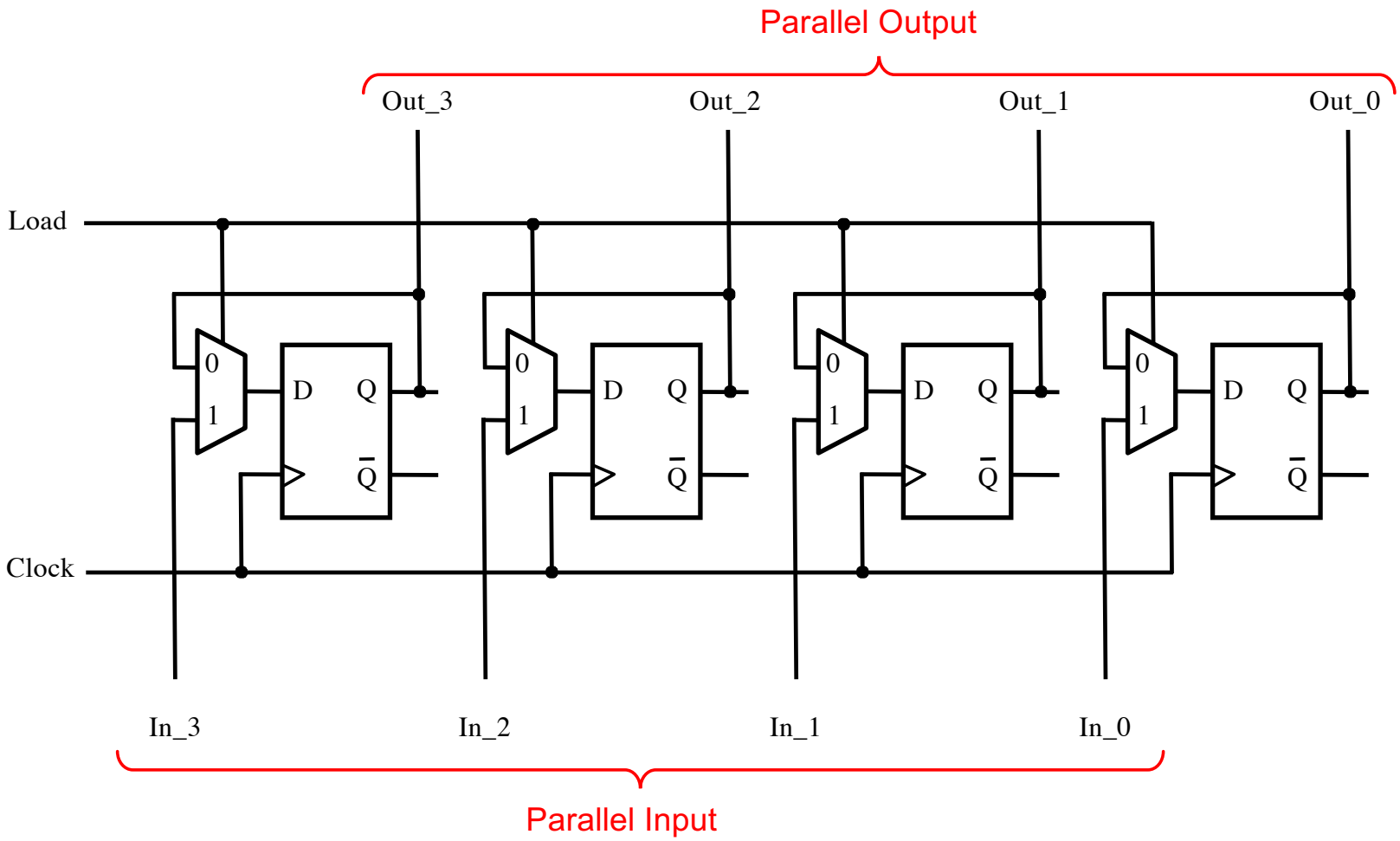
1-Bit Parallel-Access Register



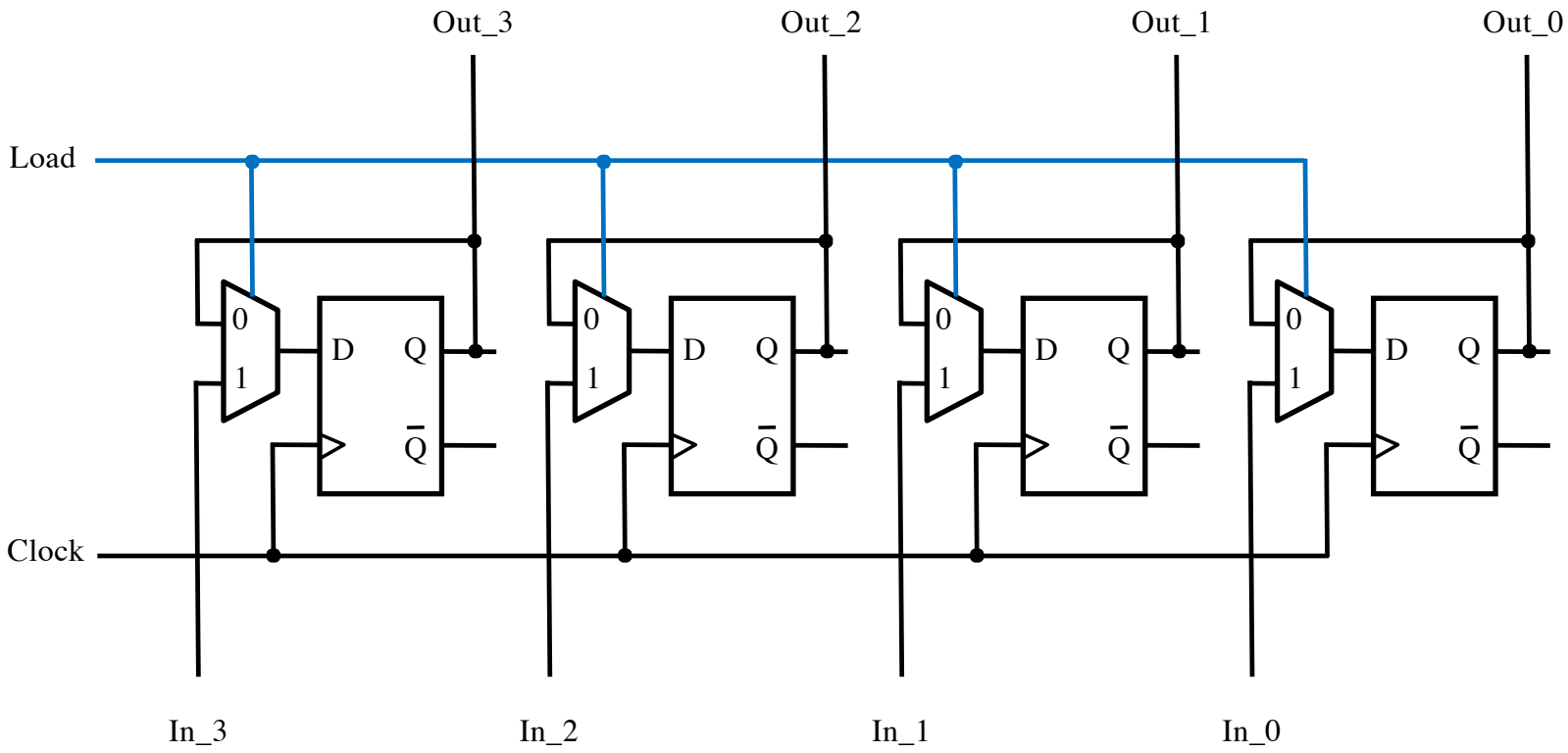
If Load = 0, then retain the old value.

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

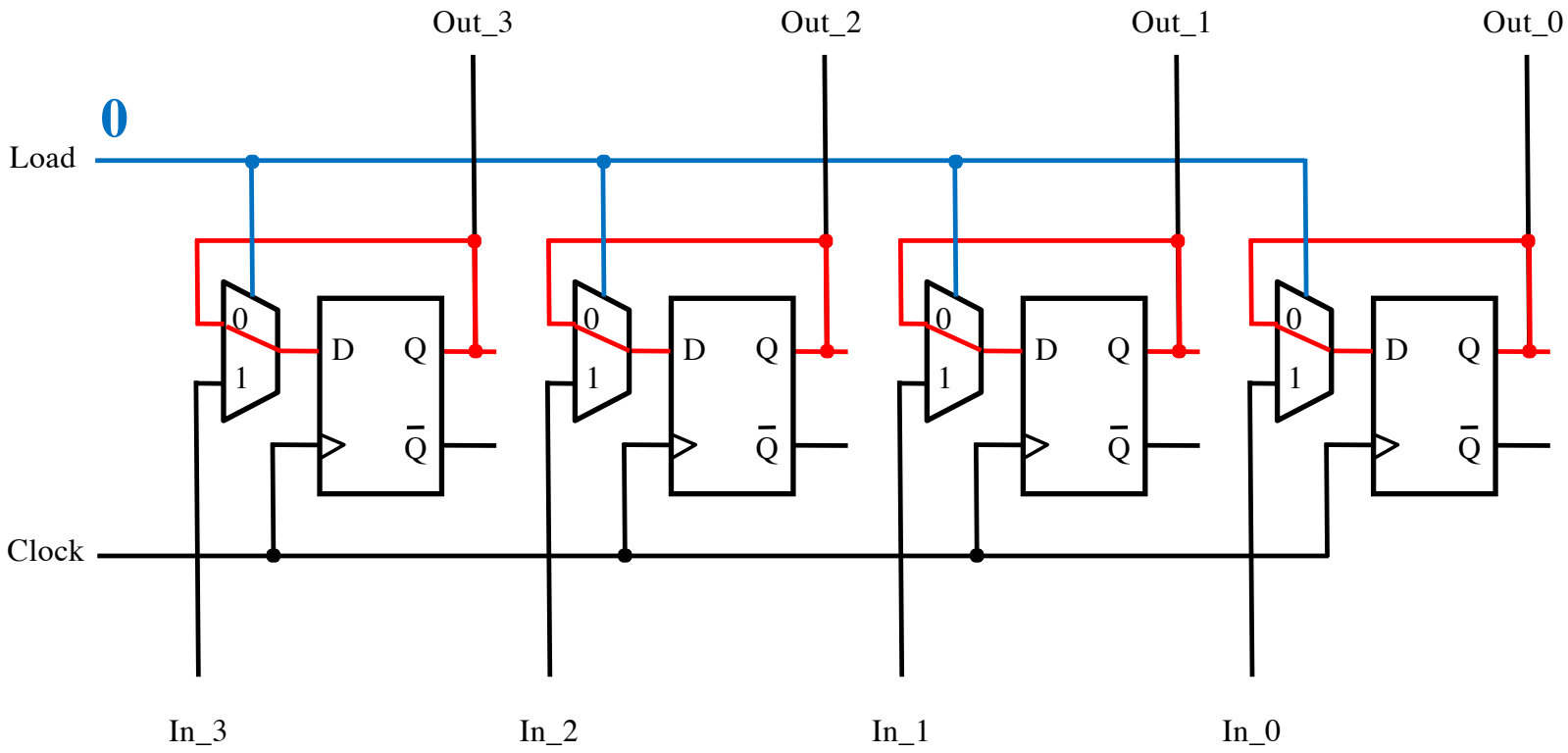
4-Bit Parallel-Access Register



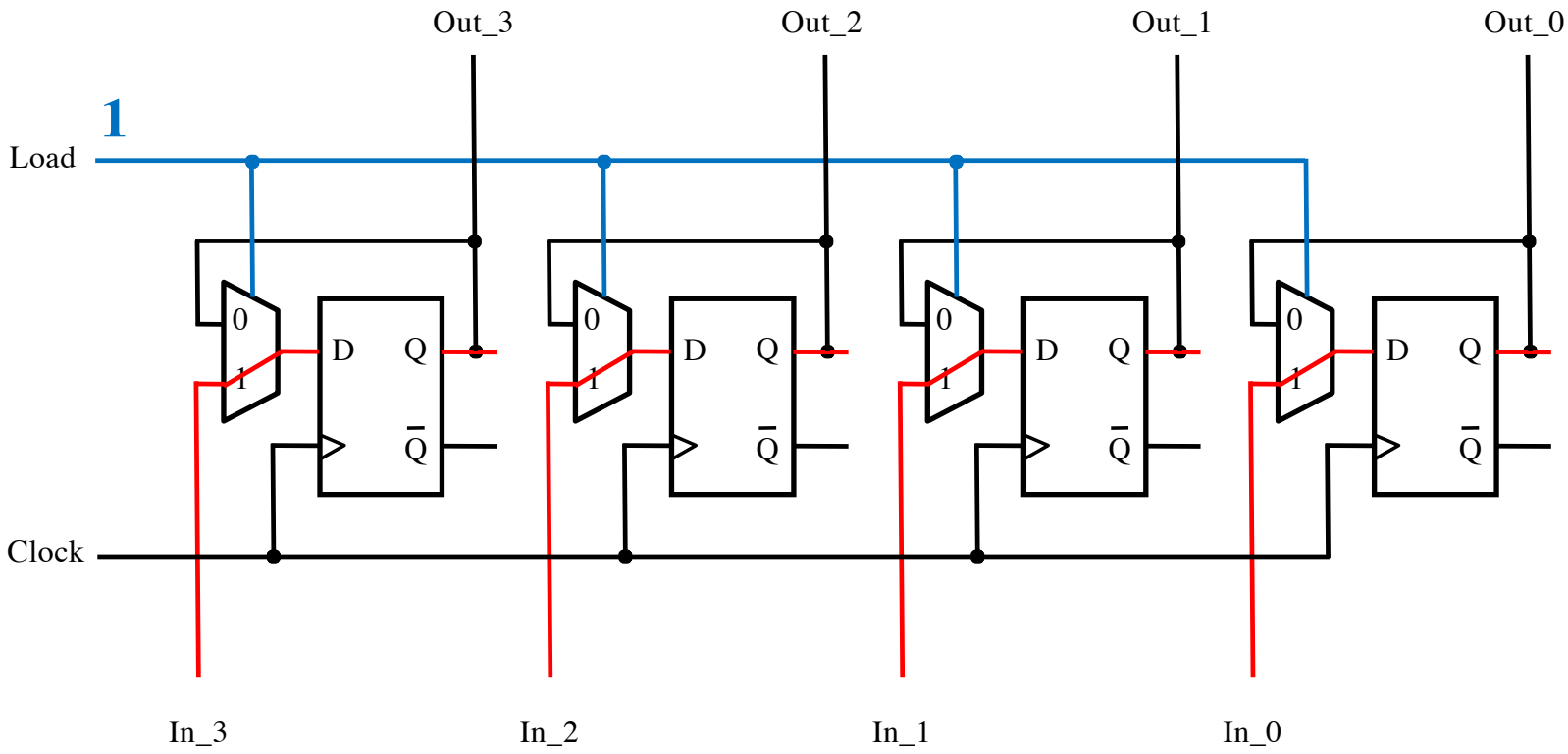
4-Bit Parallel-Access Register



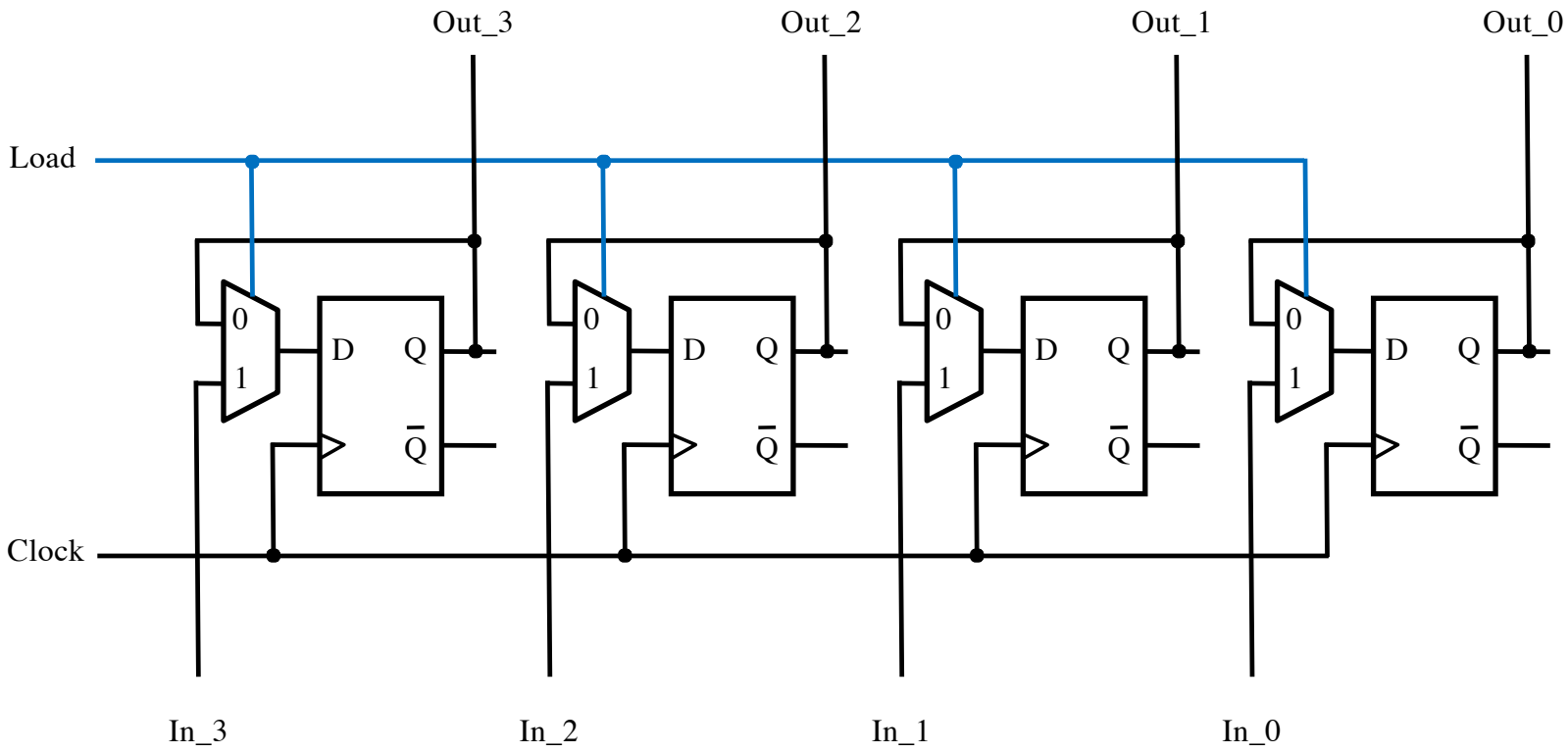
4-Bit Parallel-Access Register



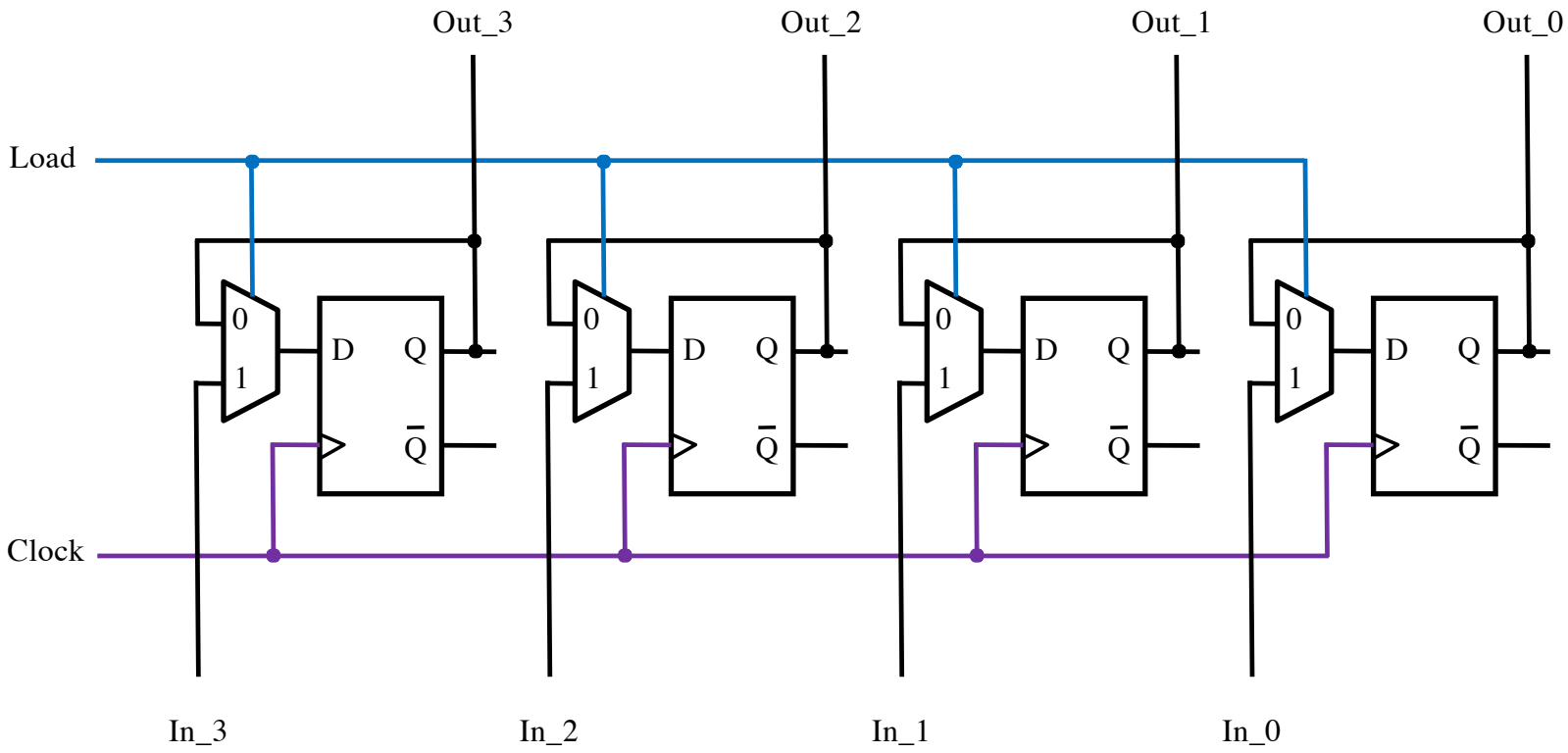
4-Bit Parallel-Access Register



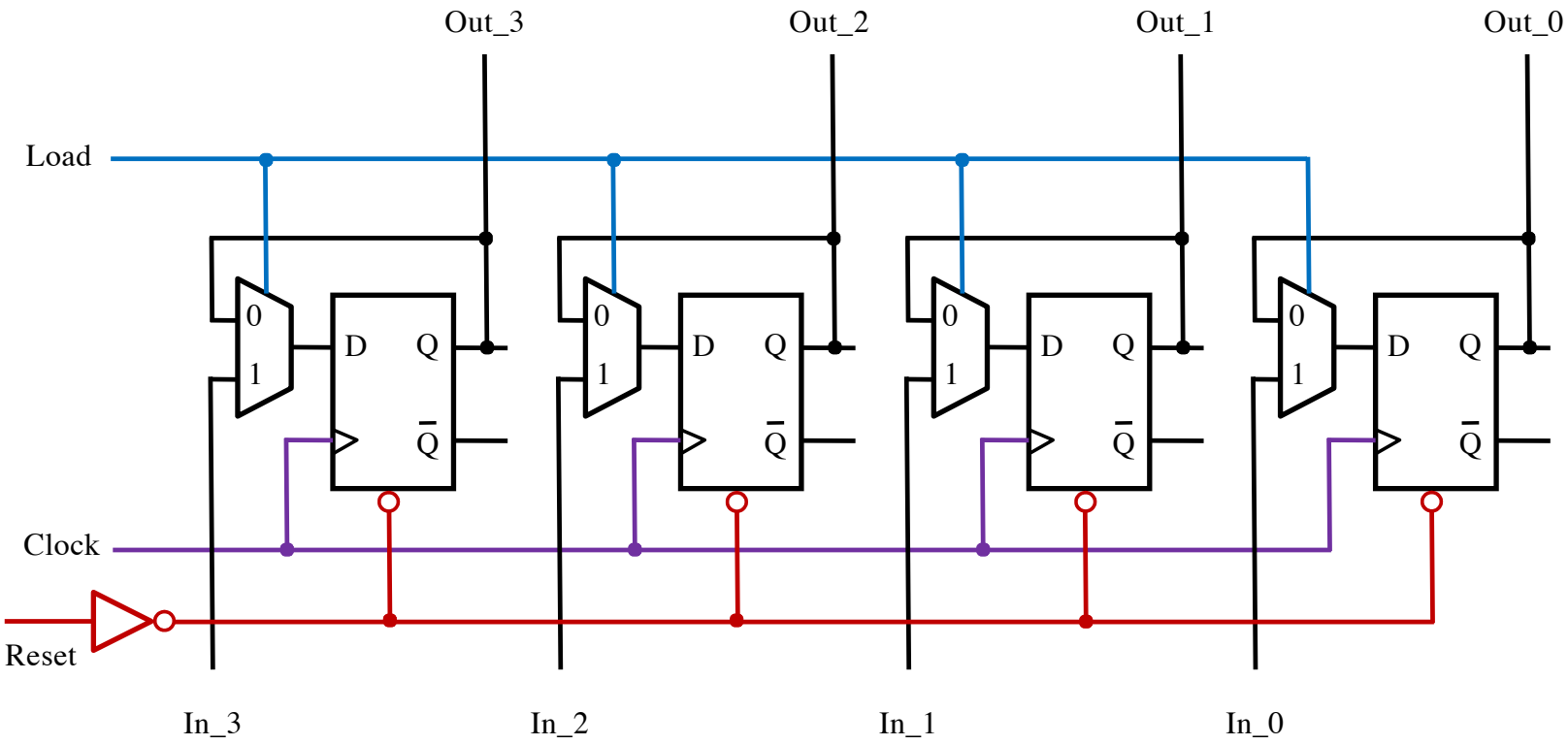
4-Bit Parallel-Access Register



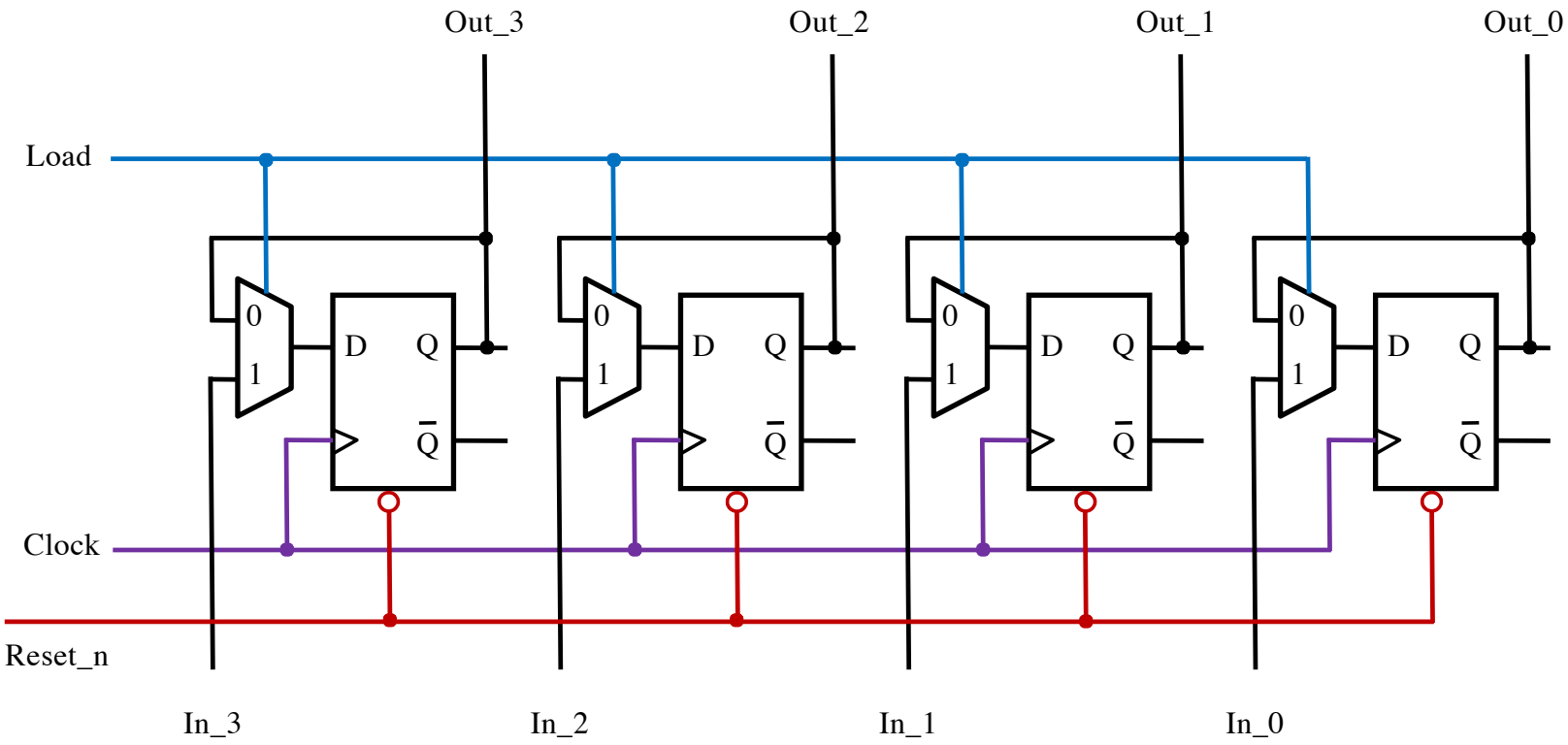
4-Bit Parallel-Access Register



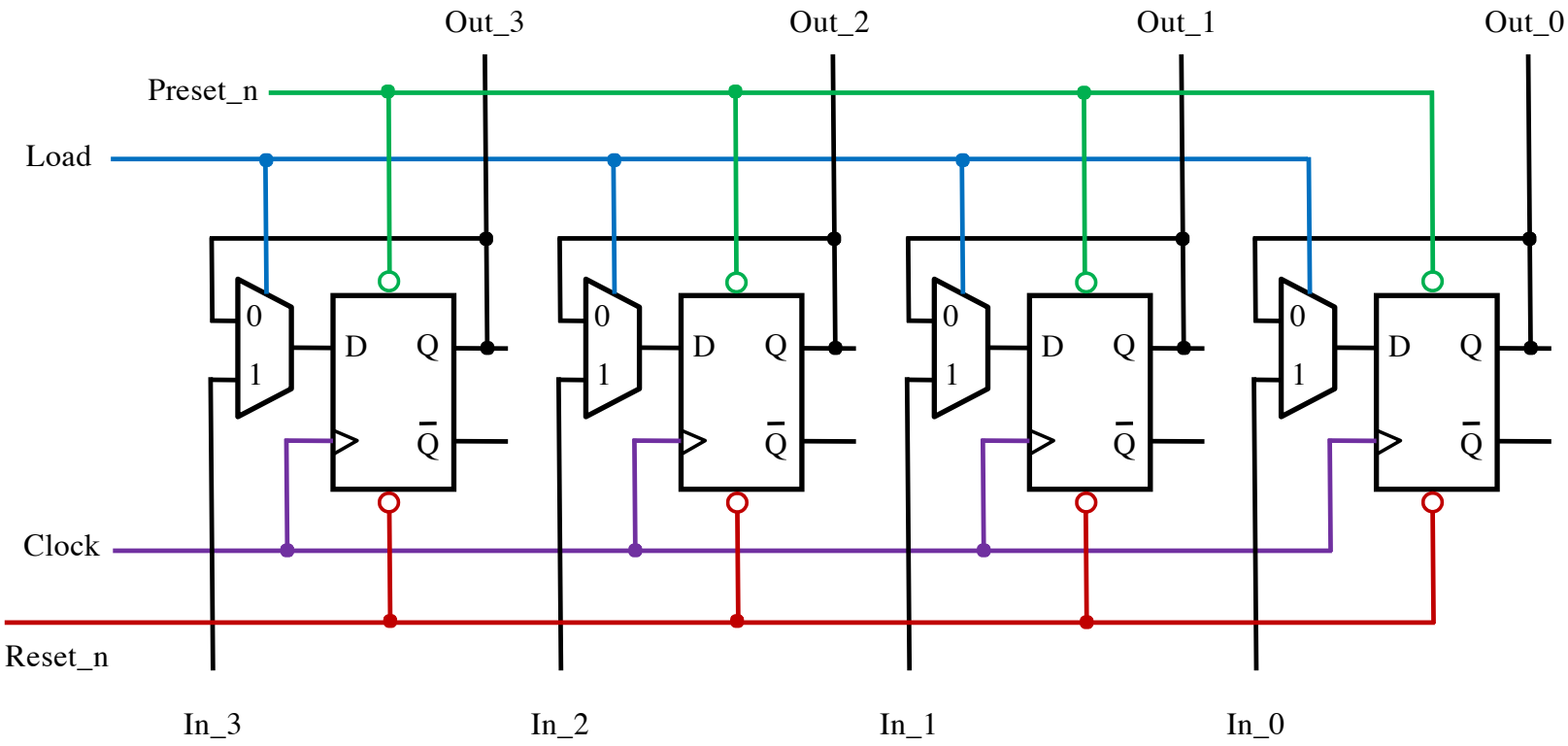
4-Bit Parallel-Access Register



4-Bit Parallel-Access Register

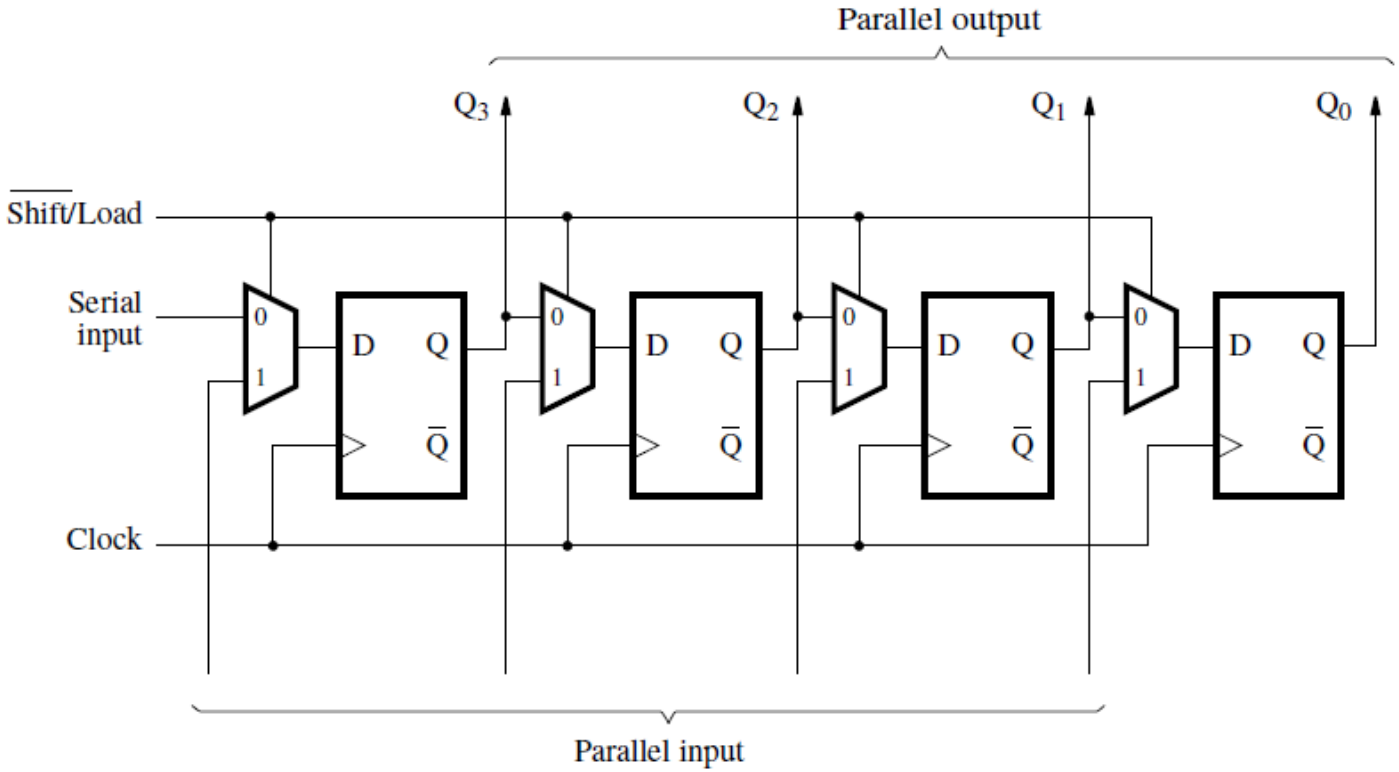


4-Bit Parallel-Access Register



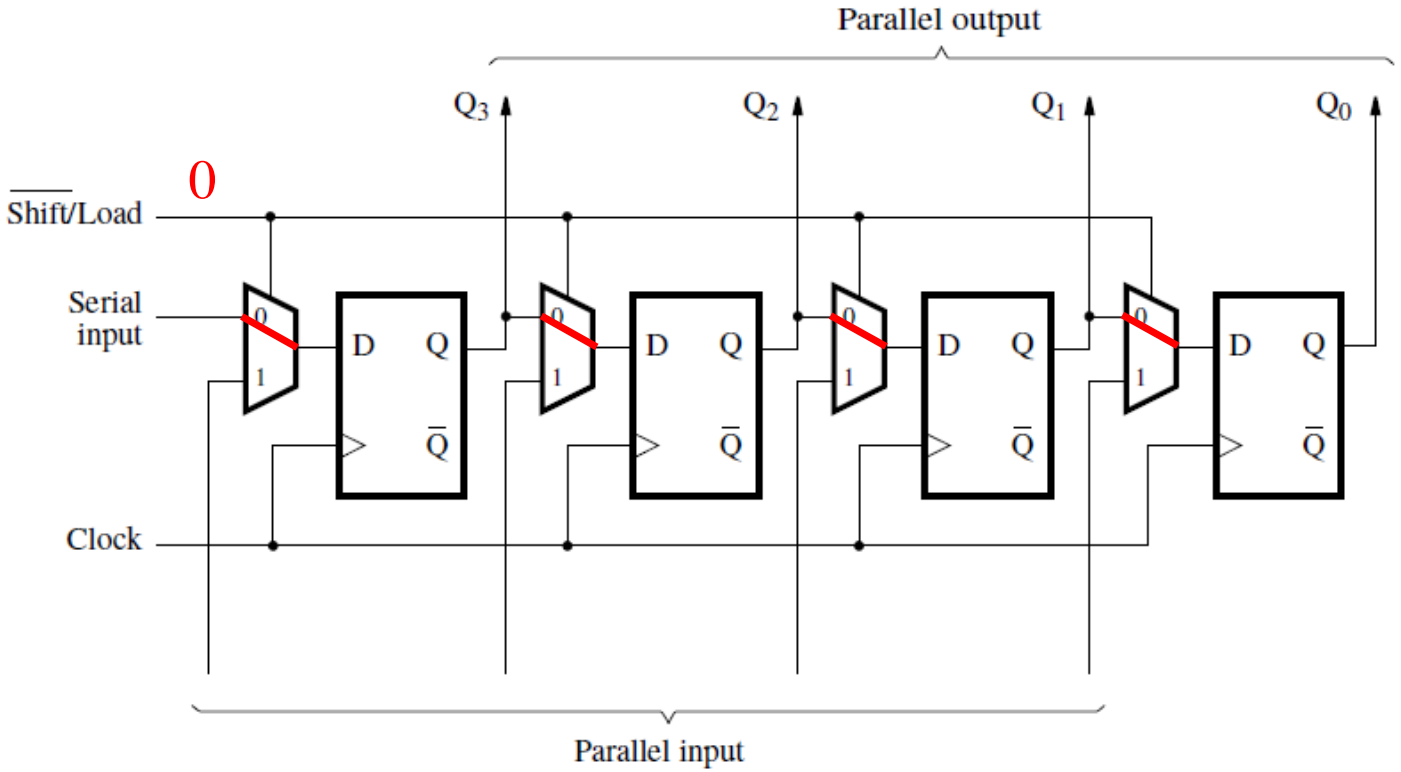
Parallel-Access Shift Register

Parallel-access shift register



[Figure 5.18 from the textbook]

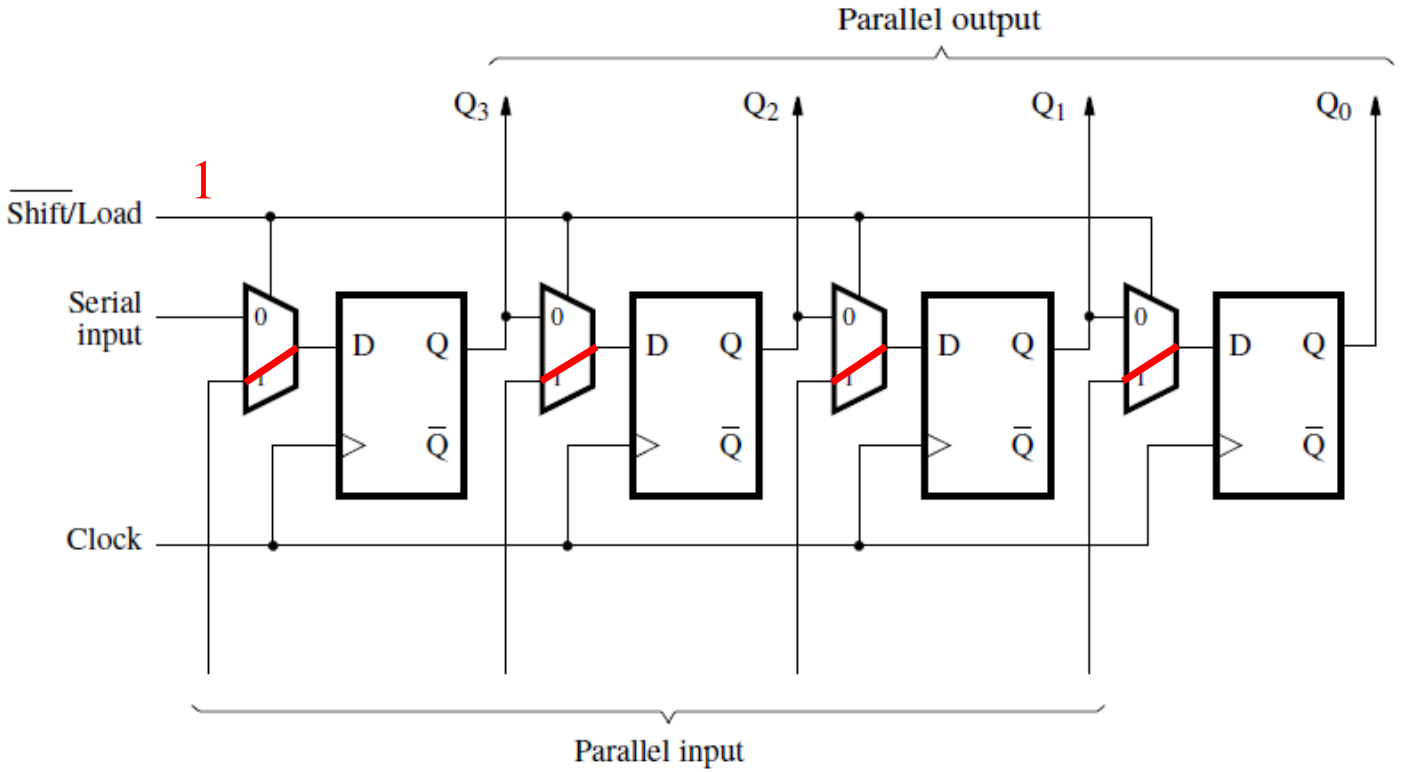
Parallel-access shift register



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

Register
(Definition)

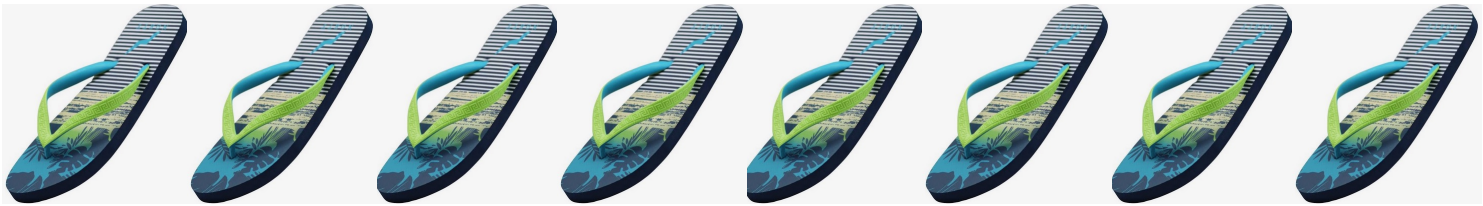
An n-bit structure consisting of flip-flops.

Flip-Flop

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

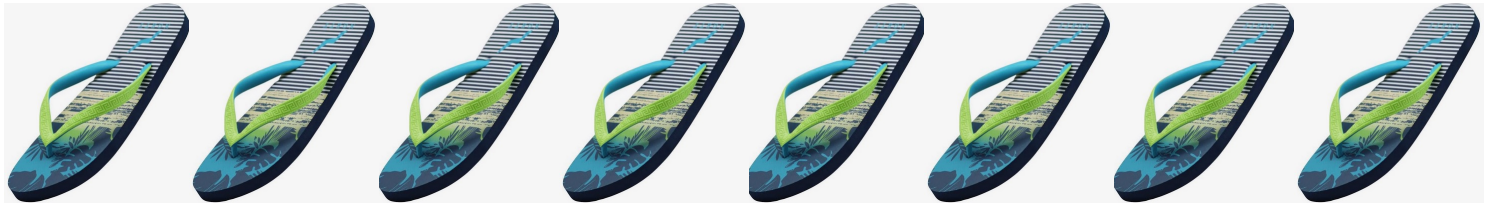
4-bit Register

8-bit Register

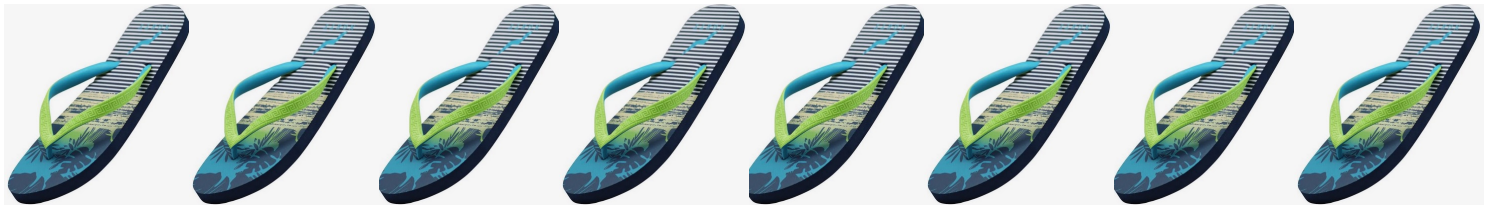


Register file with four 8-bit registers

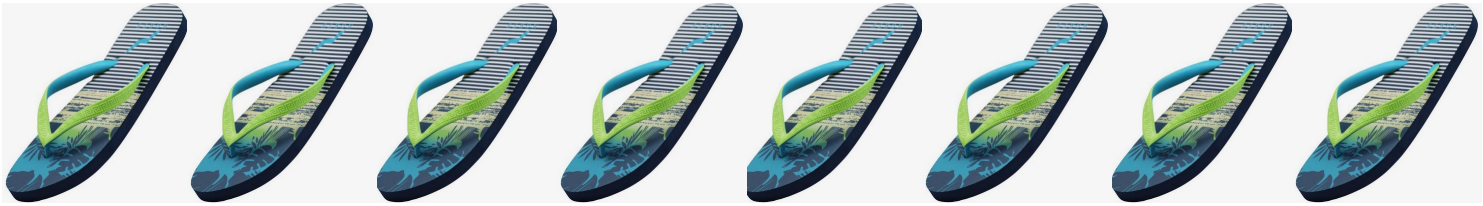
Register file with four 8-bit registers



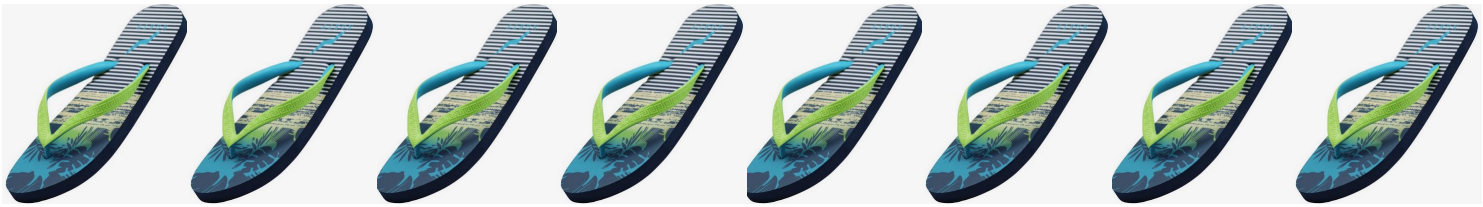
Register A



Register B

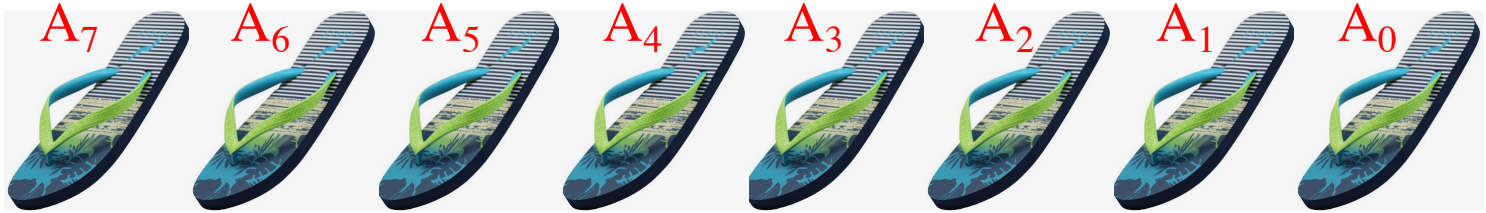


Register C

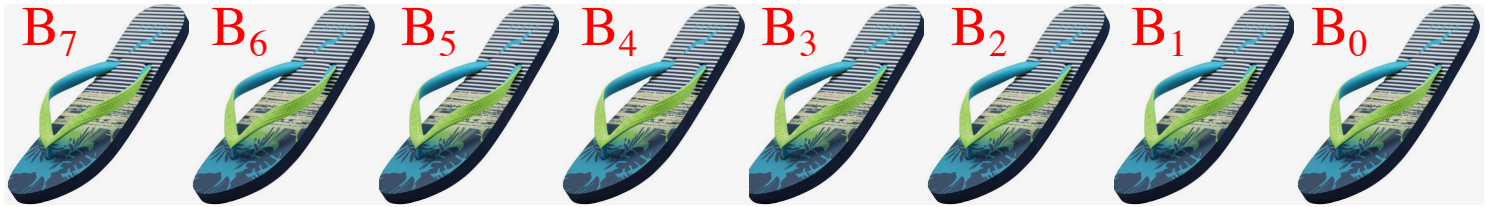


Register D

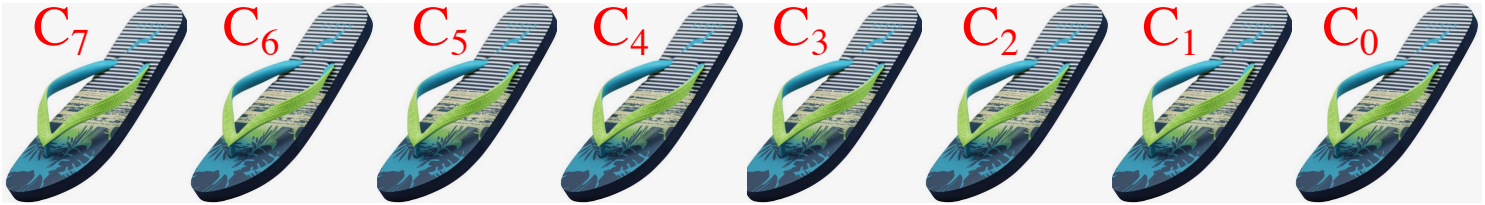
Register file with four 8-bit registers



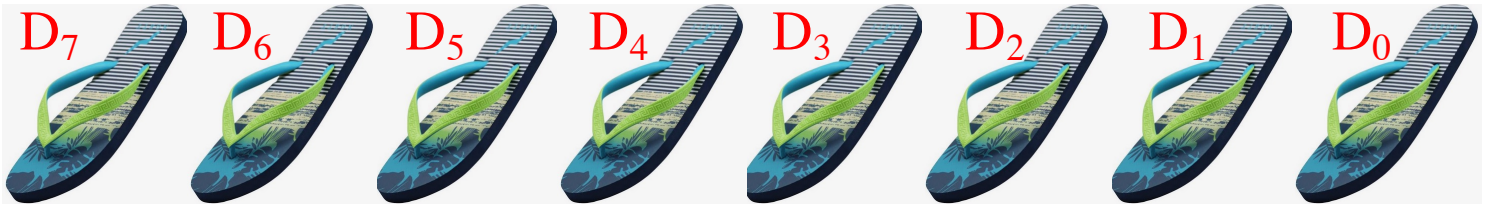
Register A



Register B



Register C

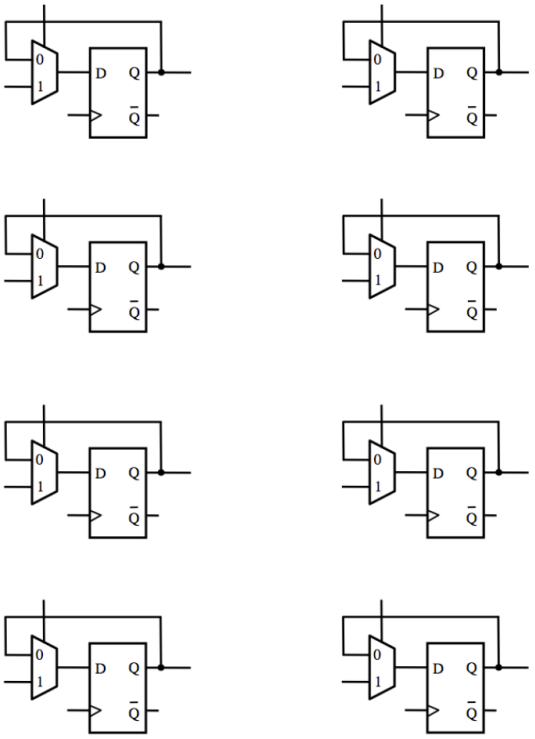


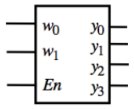
Register D

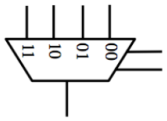
Register File (Definition)

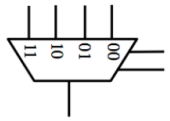
- A circuit that can store several binary numbers, which can be read or written independently.
- Each number is stored in a separate n-bit register.
- To write: a decoder selects which resister is enabled for writing. An input bus provides the values.
- To read: a set of multiplexers select which register 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.

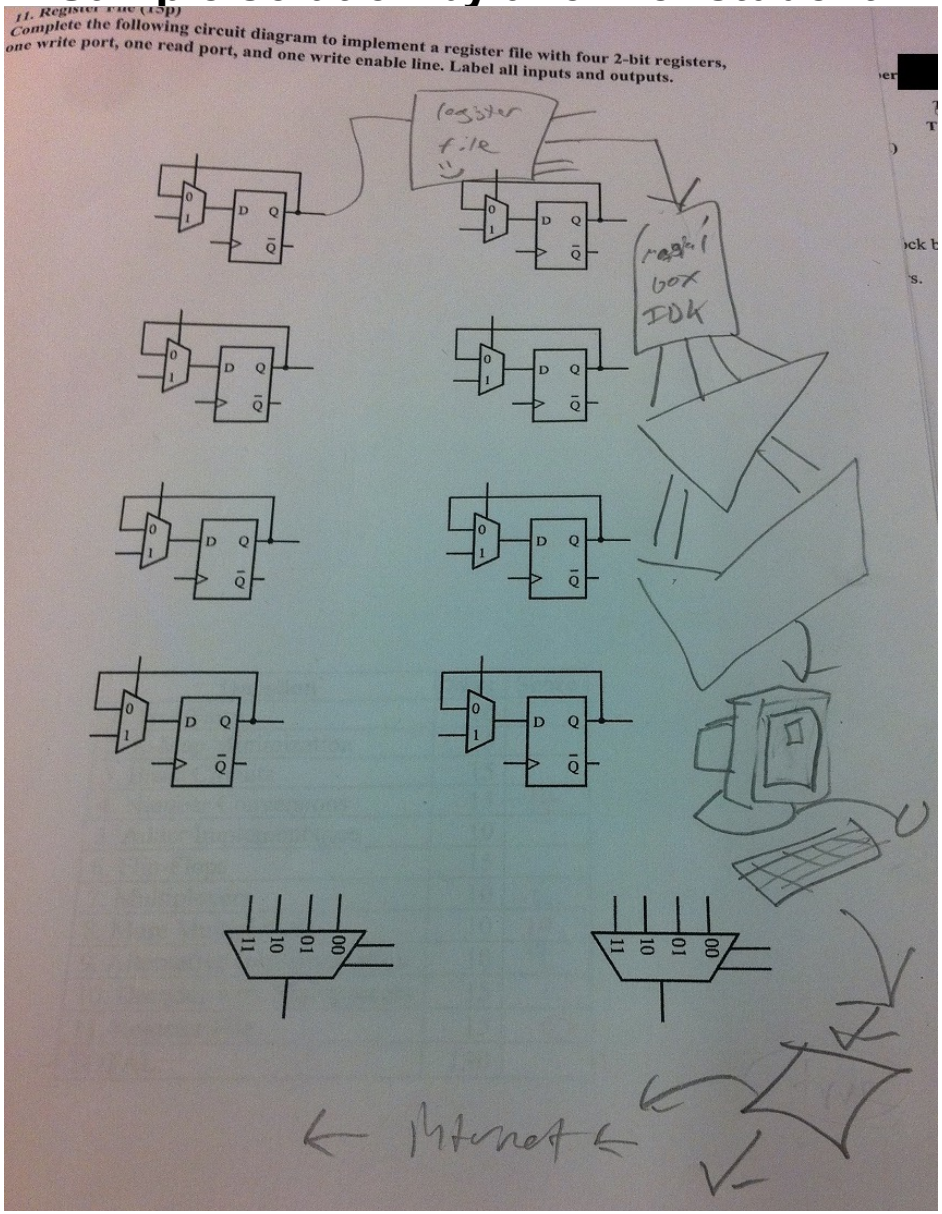


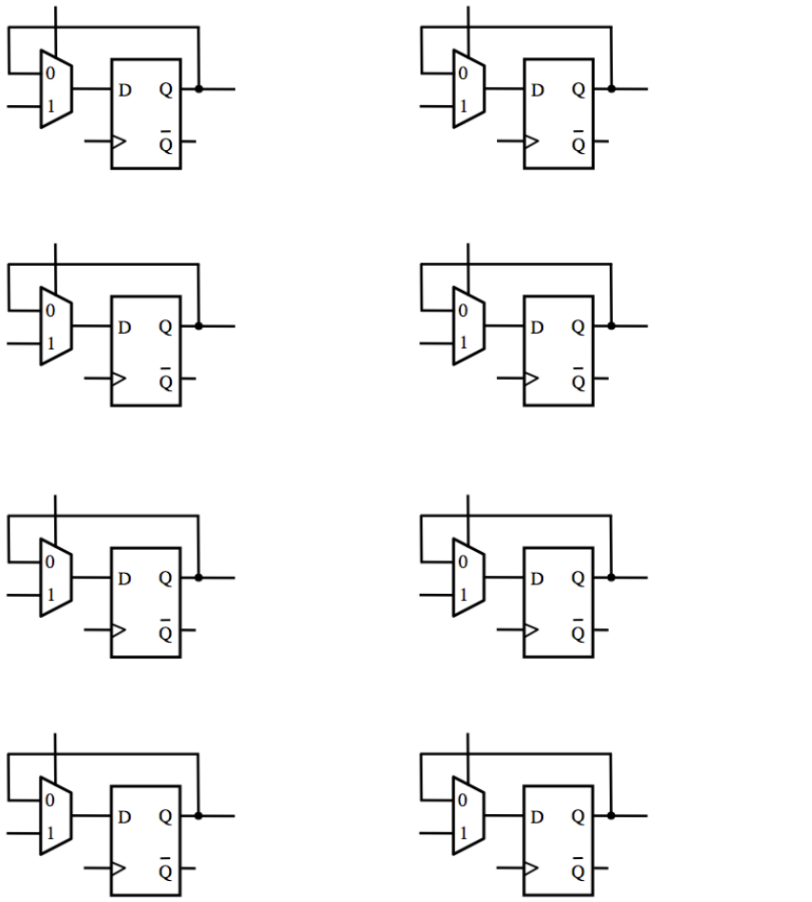


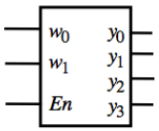


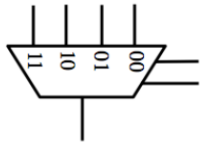


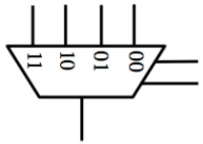
Sample solution by a former student

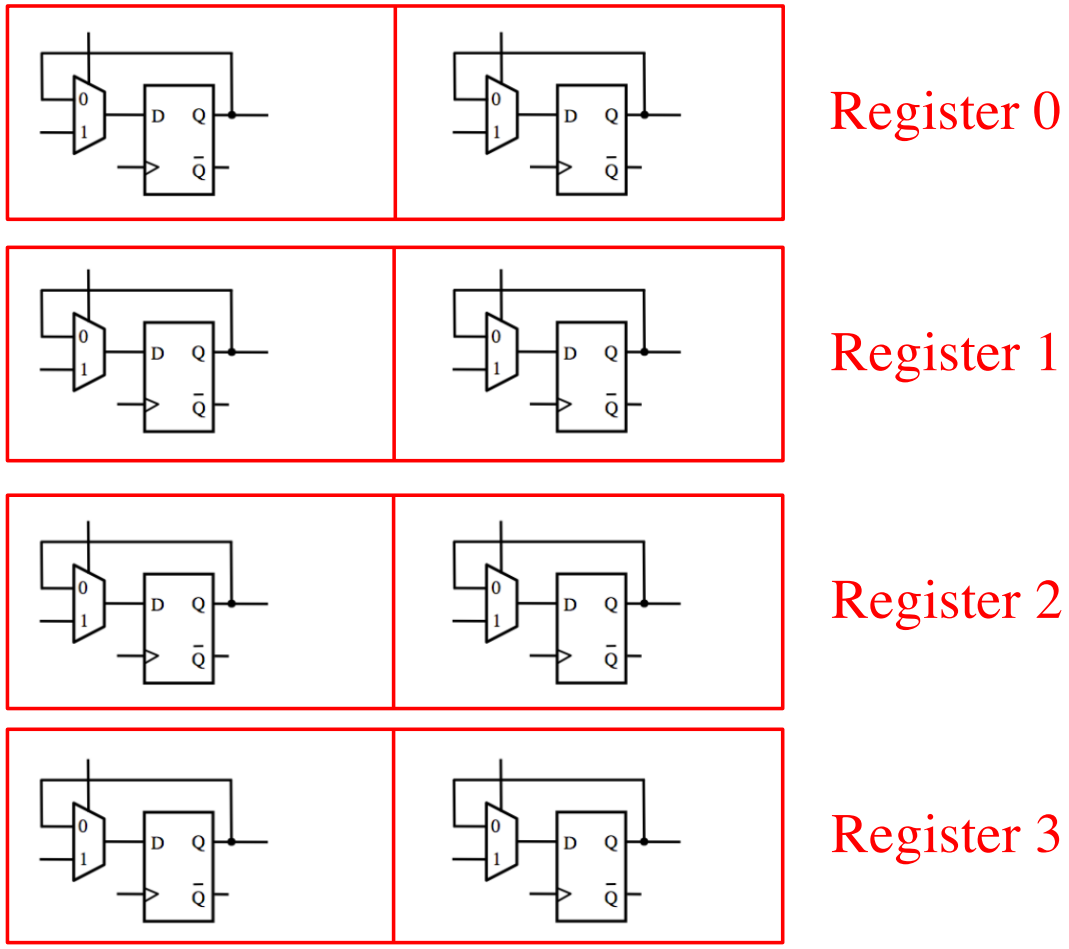


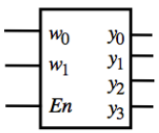

0
1
D
Q
Q

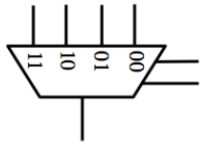

w_0
w_1
En
y_0
y_1
y_2
y_3

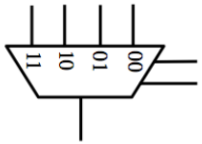

11
10
01
00

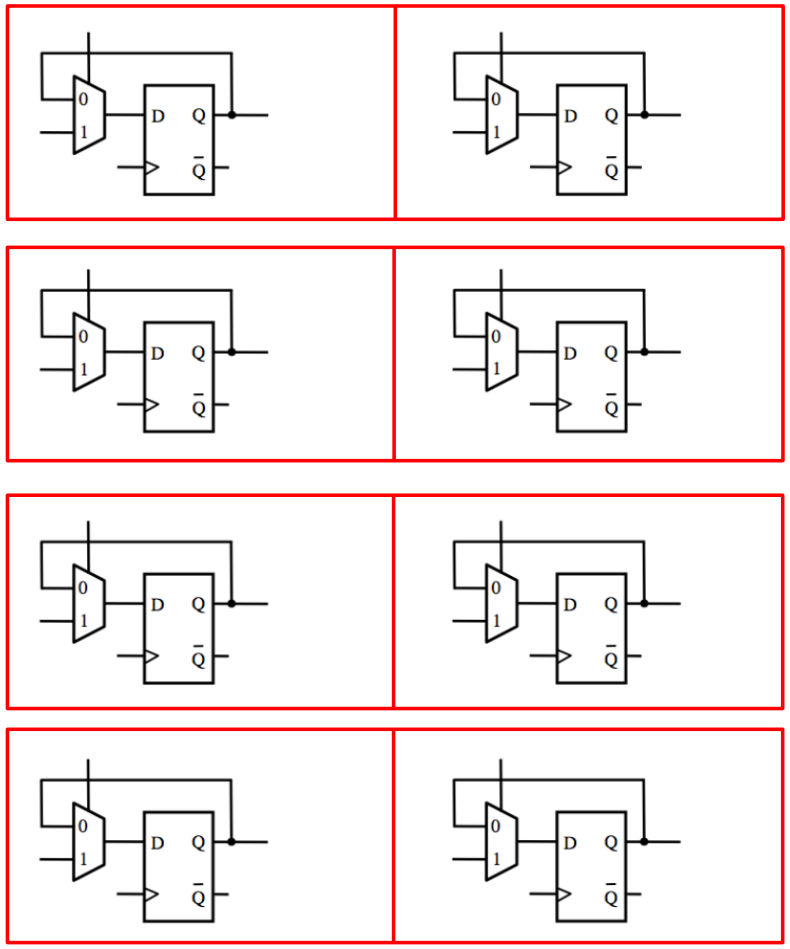

11
10
01
00

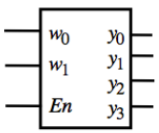

Register 0
Register 1
Register 2
Register 3

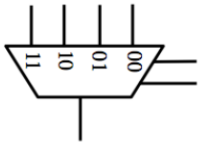

w_0
w_1
En
y_0
y_1
y_2
y_3

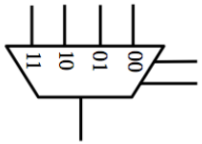

11
10
01
00

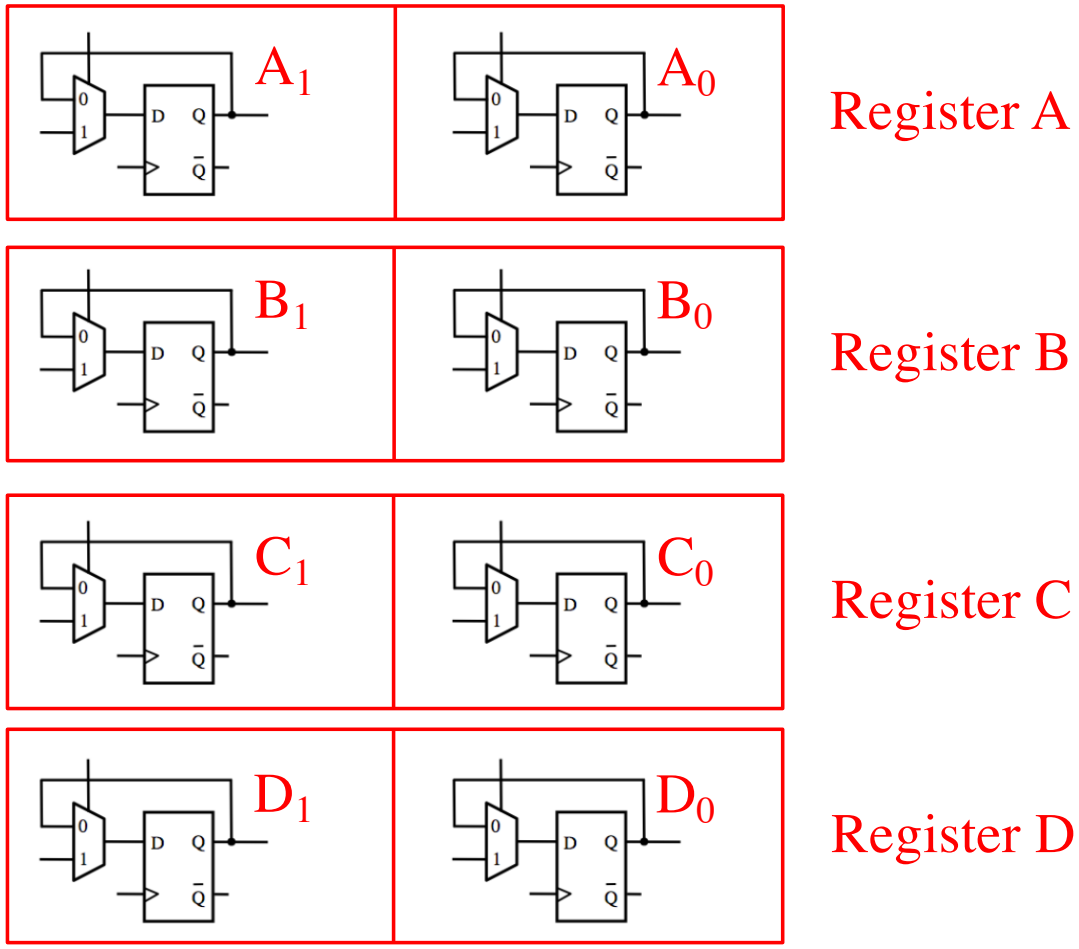

11
10
01
00

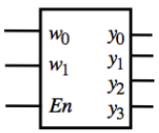

Register A
Register B
Register C
Register D

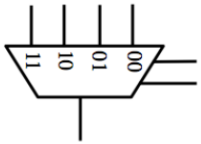

w_0
w_1
En
y_0
y_1
y_2
y_3

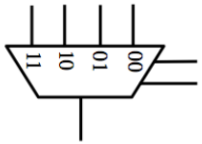

11
10
01
00

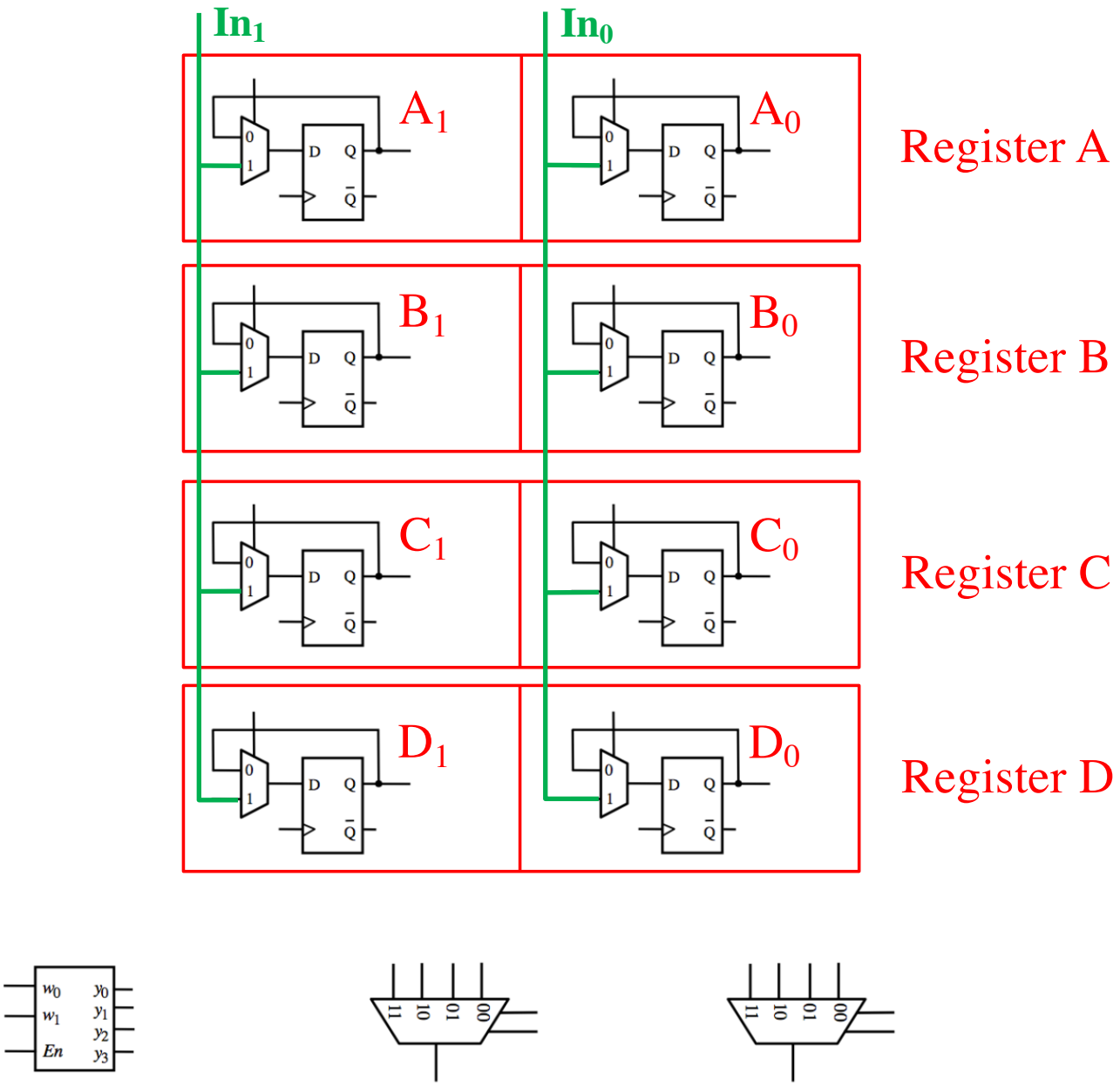

11
10
01
00

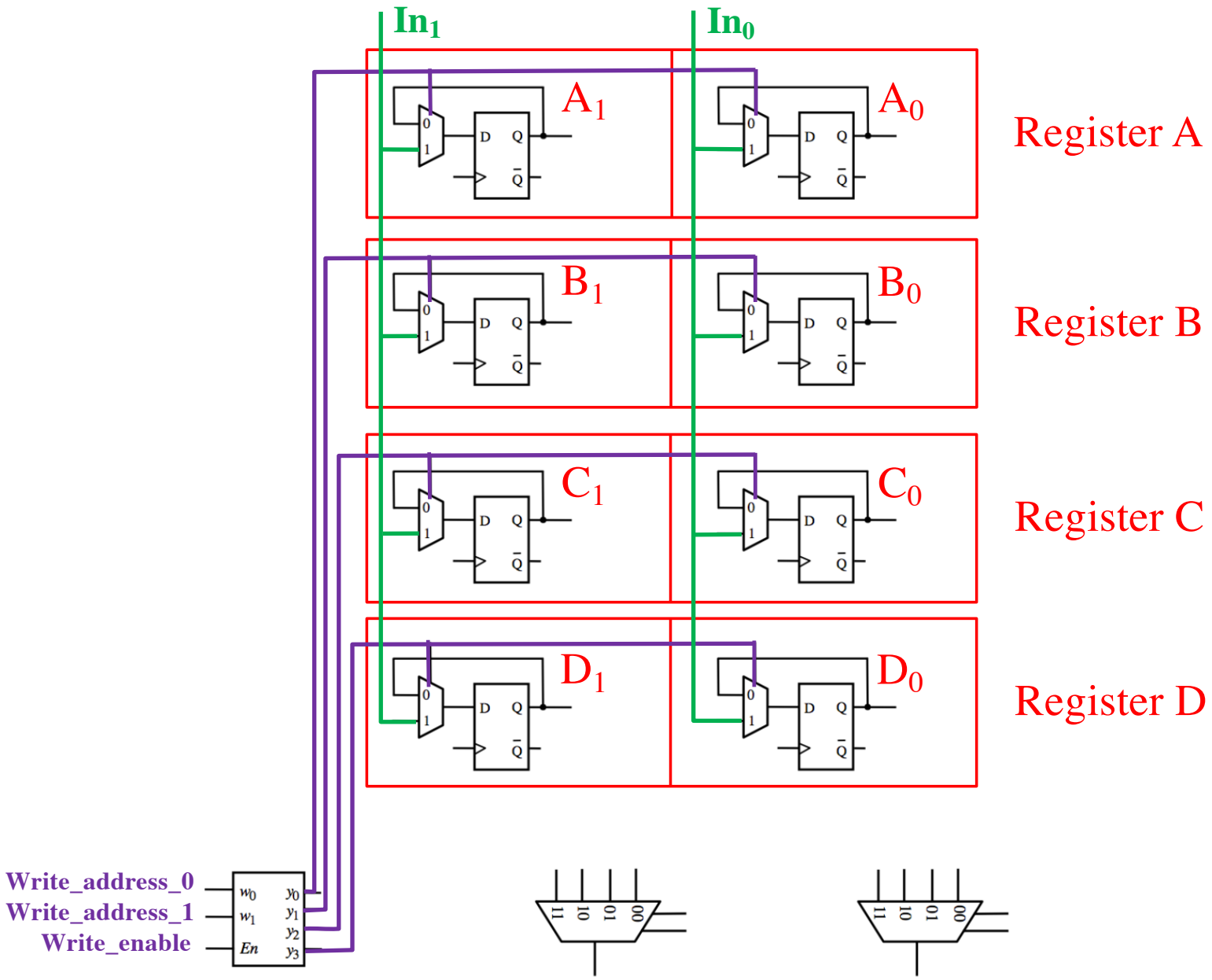

A1
A0
Register A
B1
B0
Register B
C1
C0
Register C
D1
D0
Register D
D
Q
0
1

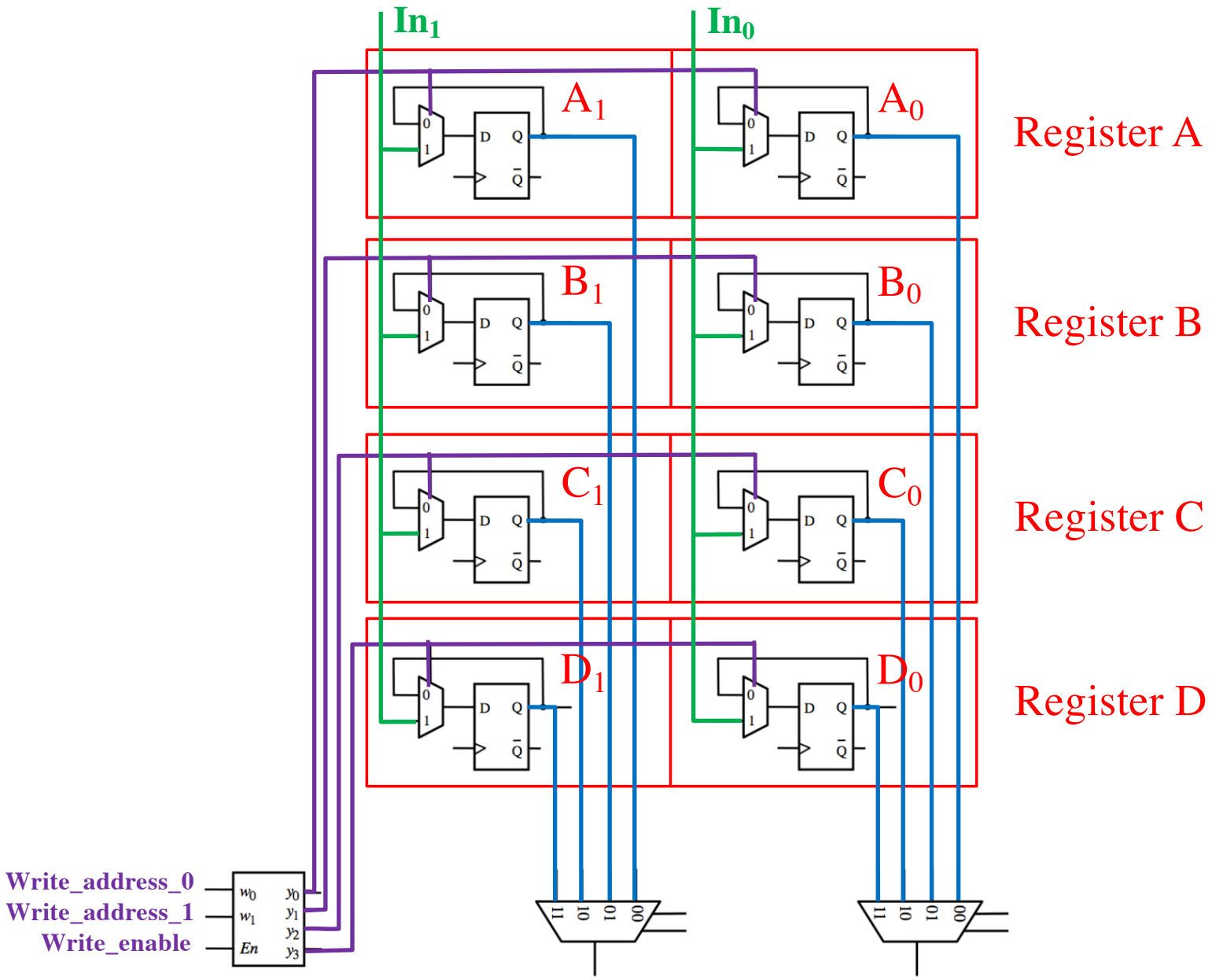

w0
w1
En
y0
y1
y2
y3

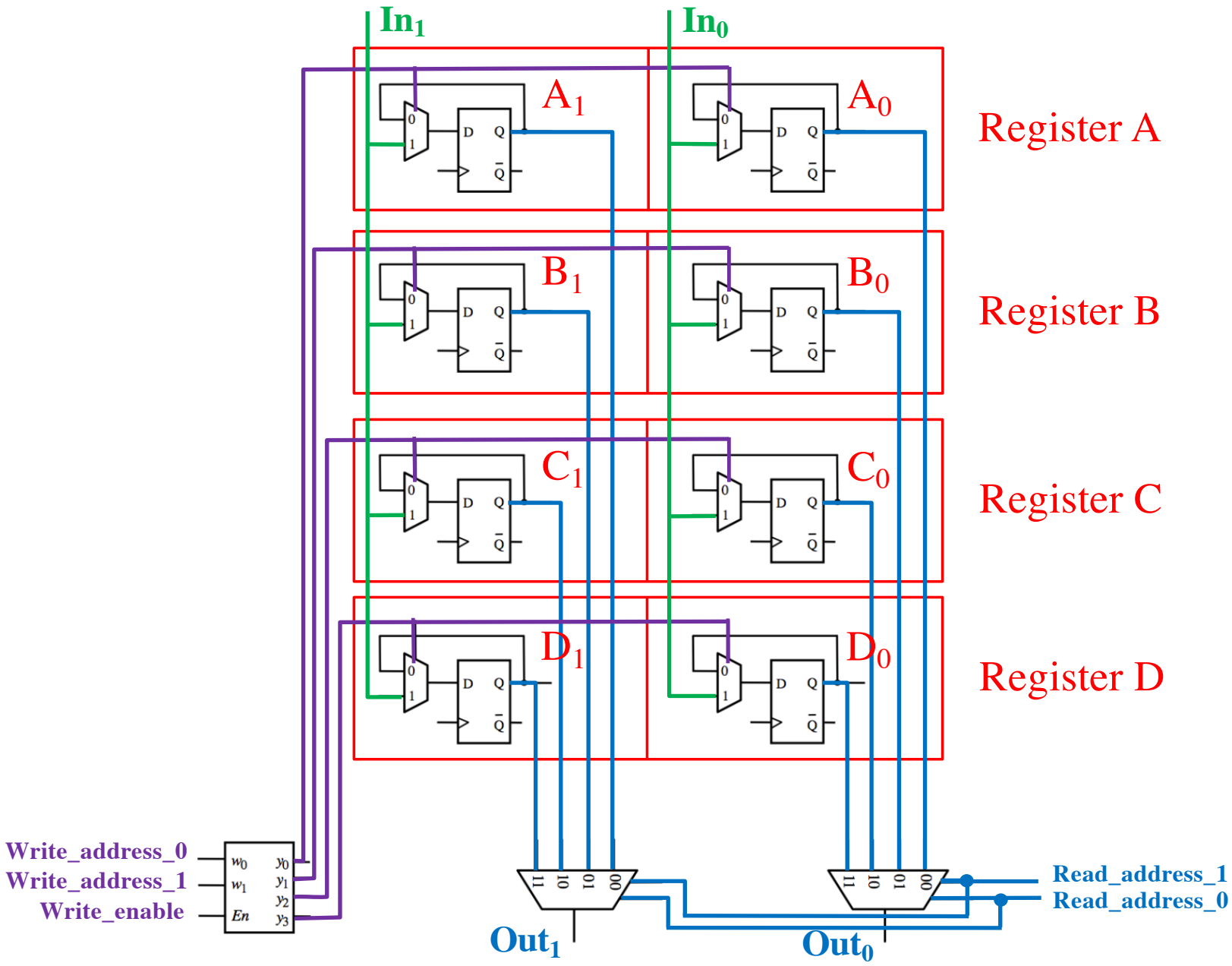

11
10
01
00

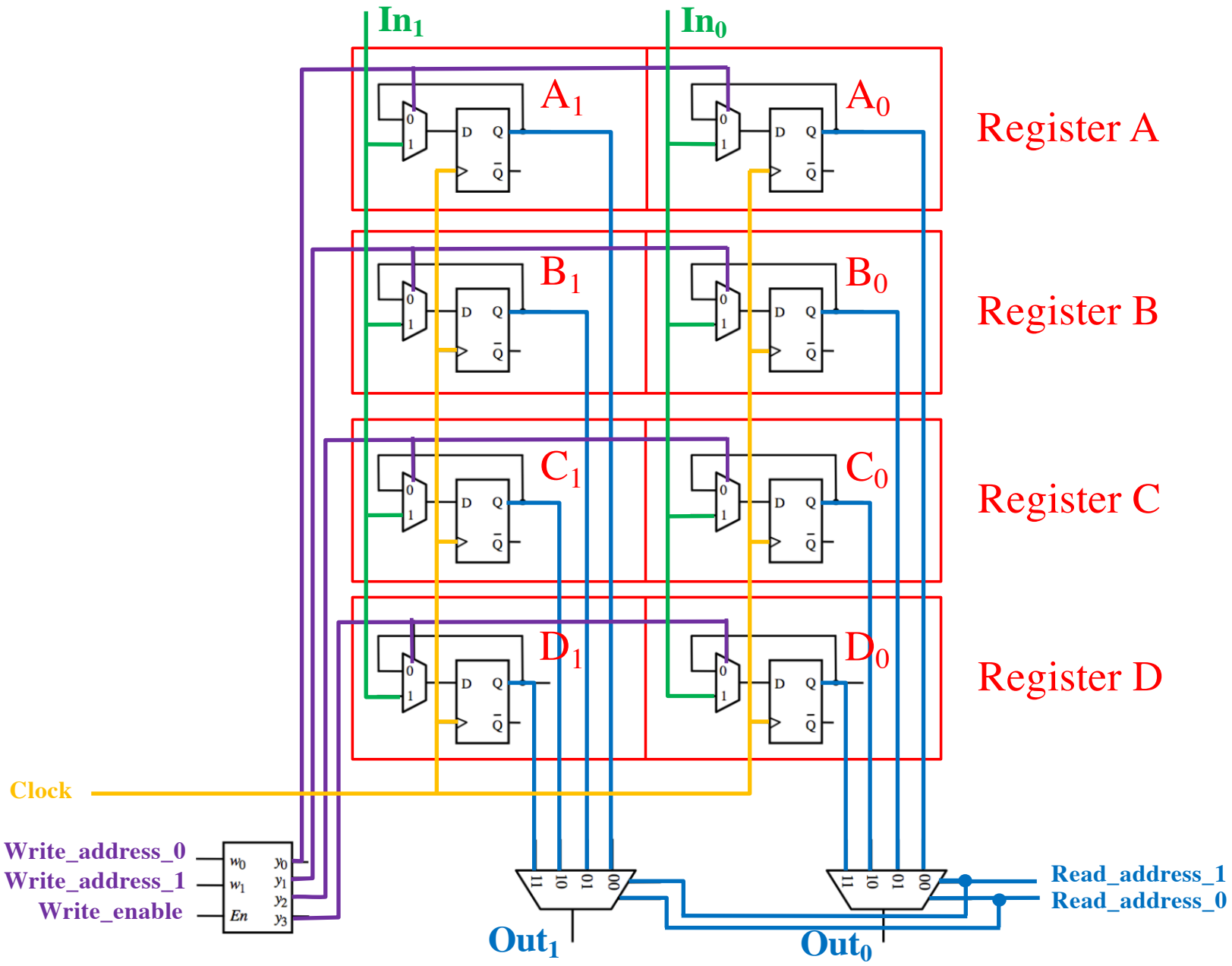

11
10
01
00

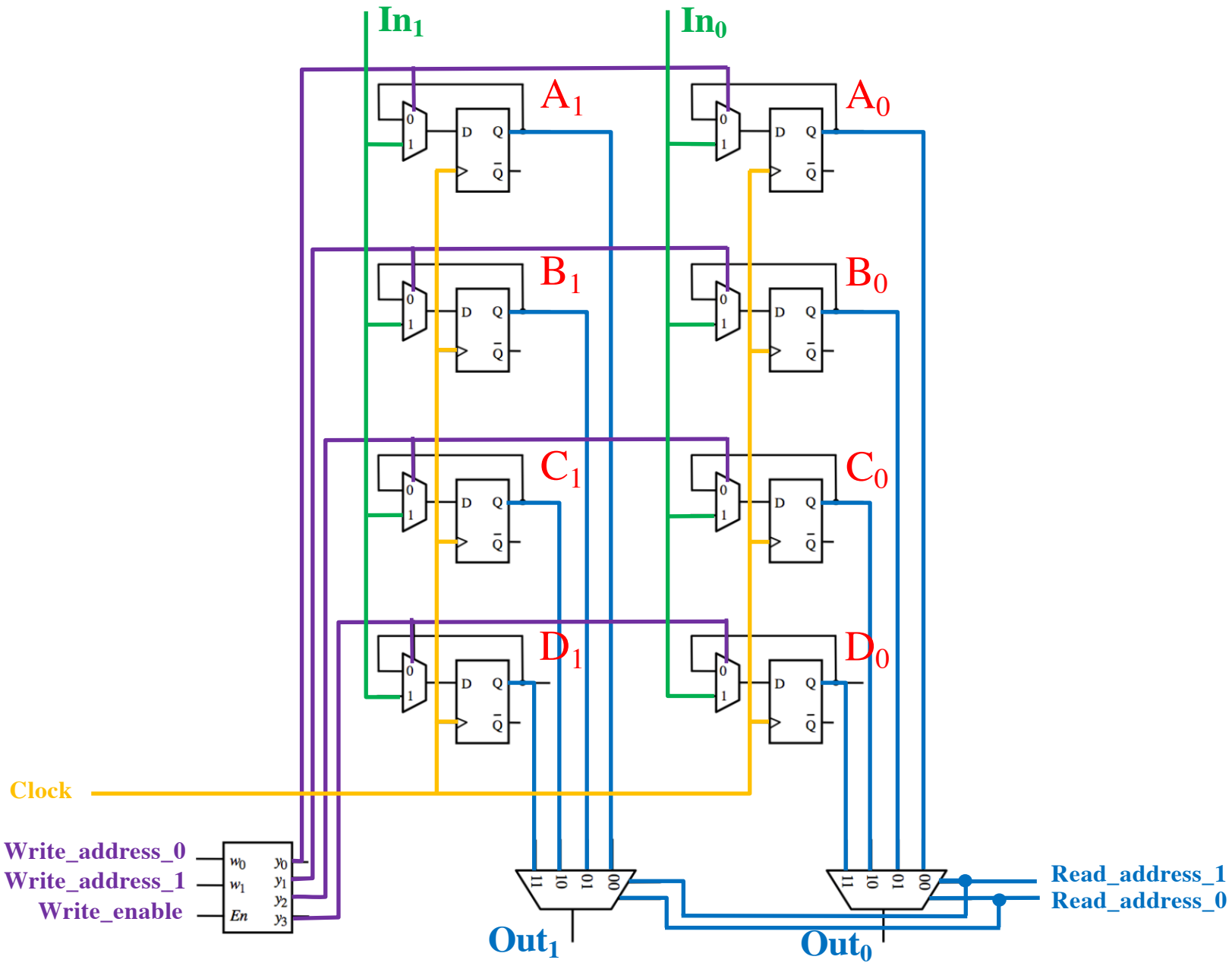

In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
Register A
Register B
Register C
Register D
0
1
D
Q
Q̄
w0
w1
En
y0
y1
y2
y3
11
10
01
00


In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
Register A
Register B
Register C
Register D
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
D
Q
Q̄
0
1
11
10
01
00

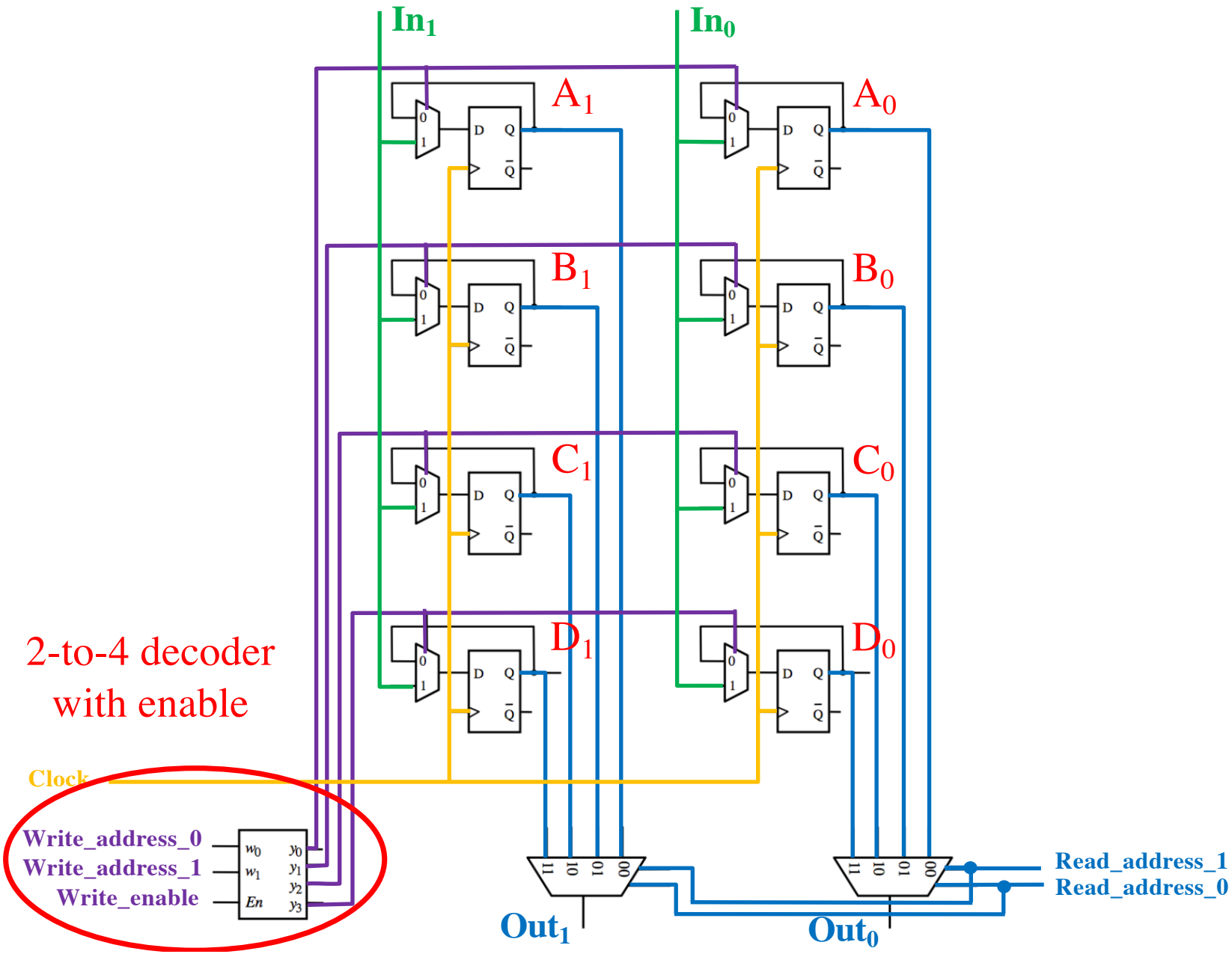

In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
Register A
Register B
Register C
Register D
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
Read_address_1
Read_address_0


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

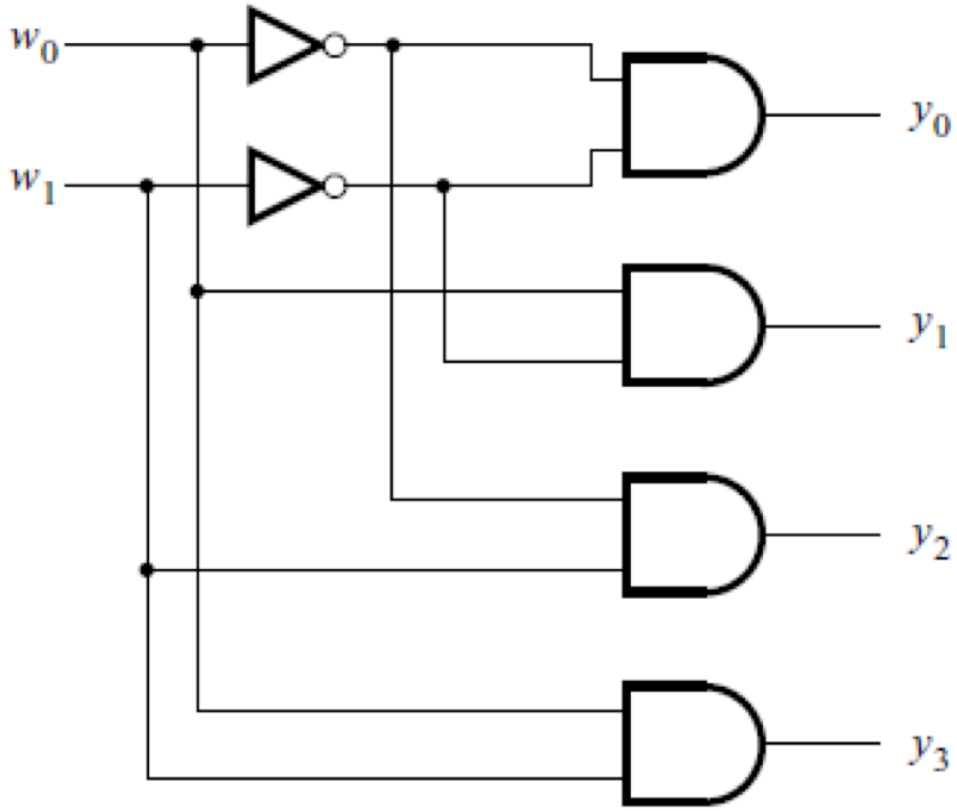

In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
D
Q
Q̄
0
1
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
Read_address_1
Read_address_0

Components of the Register File

2-to-4 Decoder

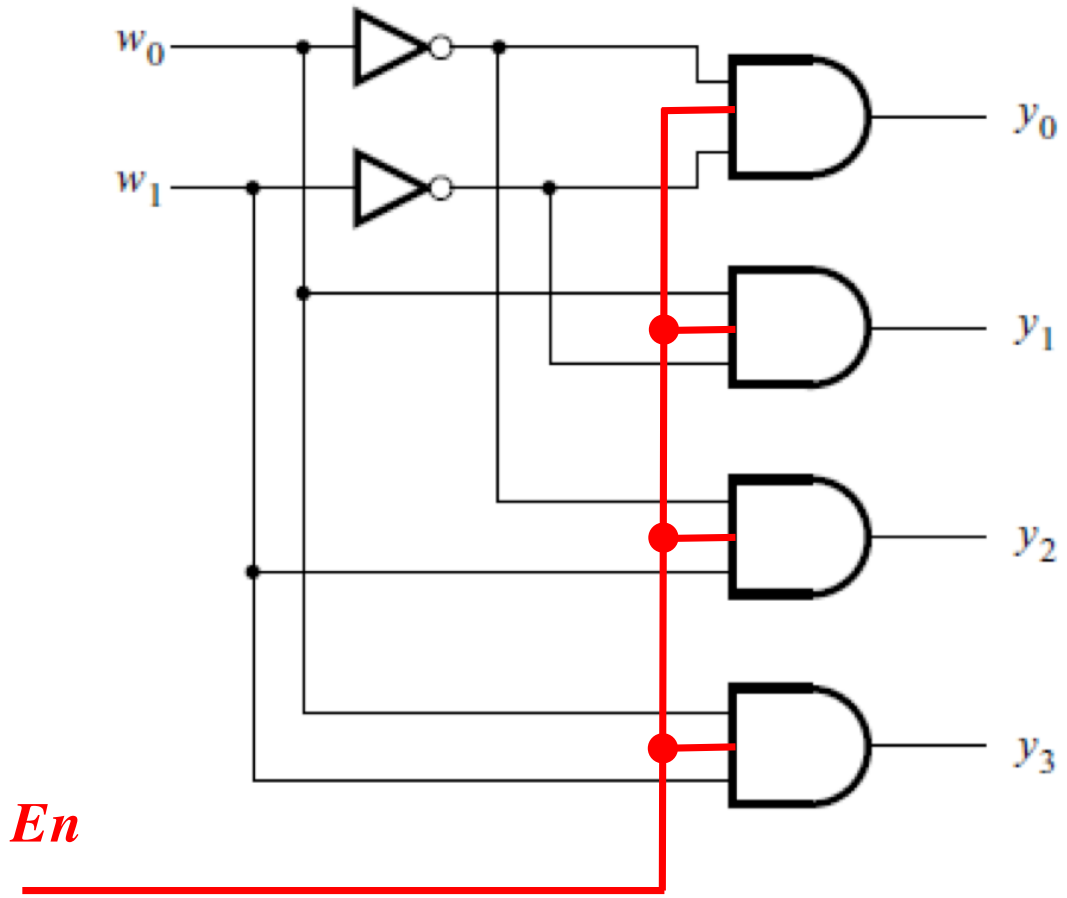

In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
2-to-4 decoder
with enable
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
Out1
Out0
Read_address_1
Read_address_0

2-to-4 Decoder

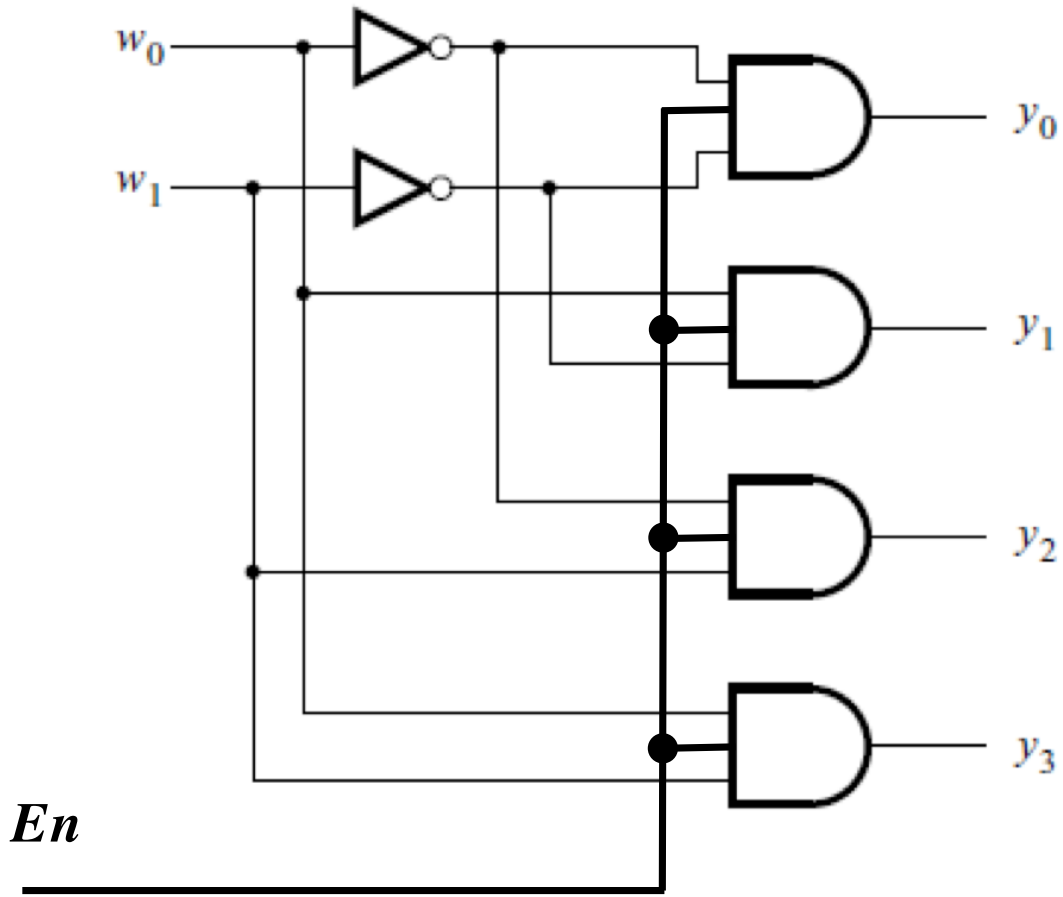


[Figure 4.14c from the textbook]

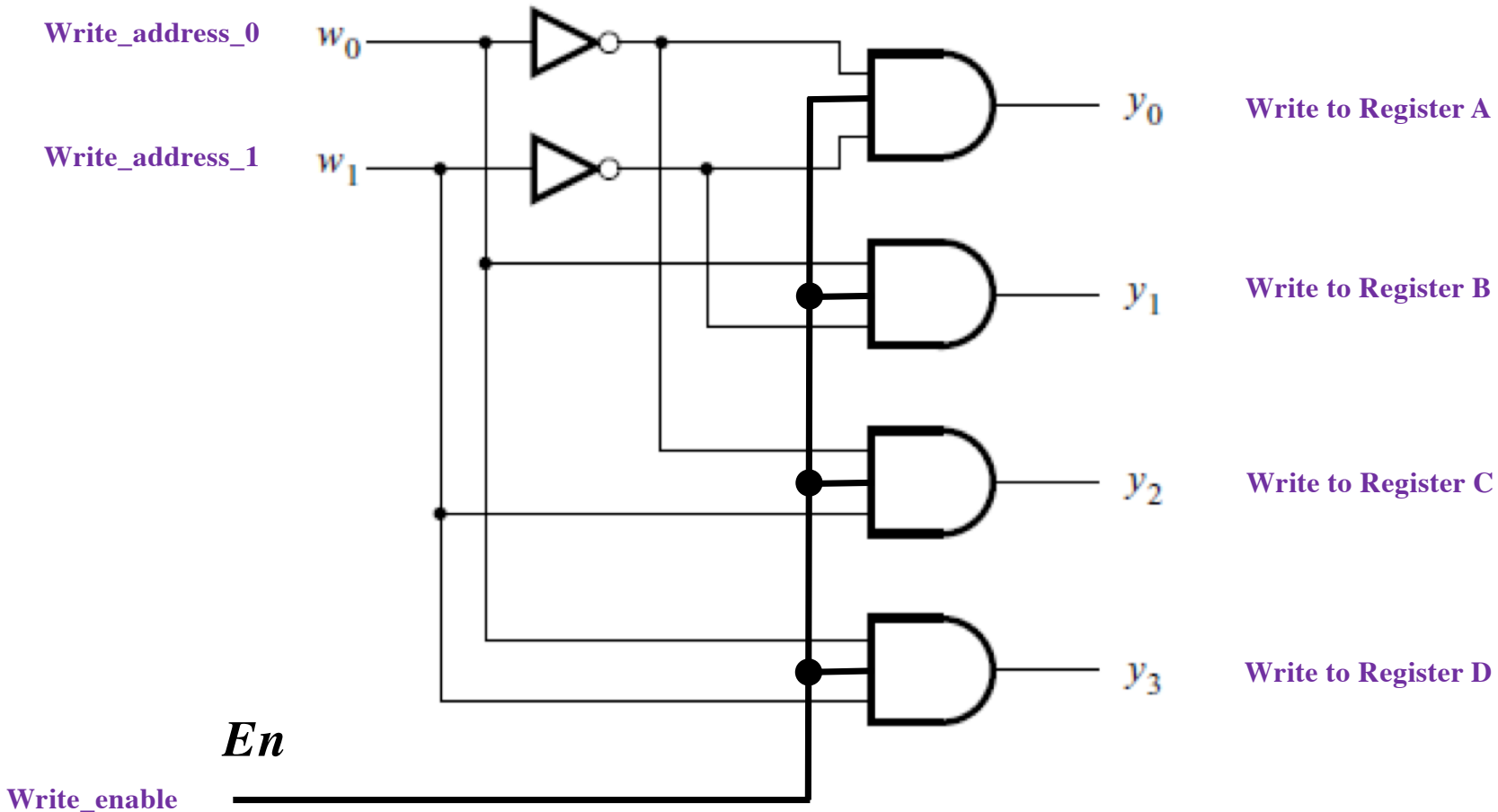
2-to-4 Decoder with Enable Input



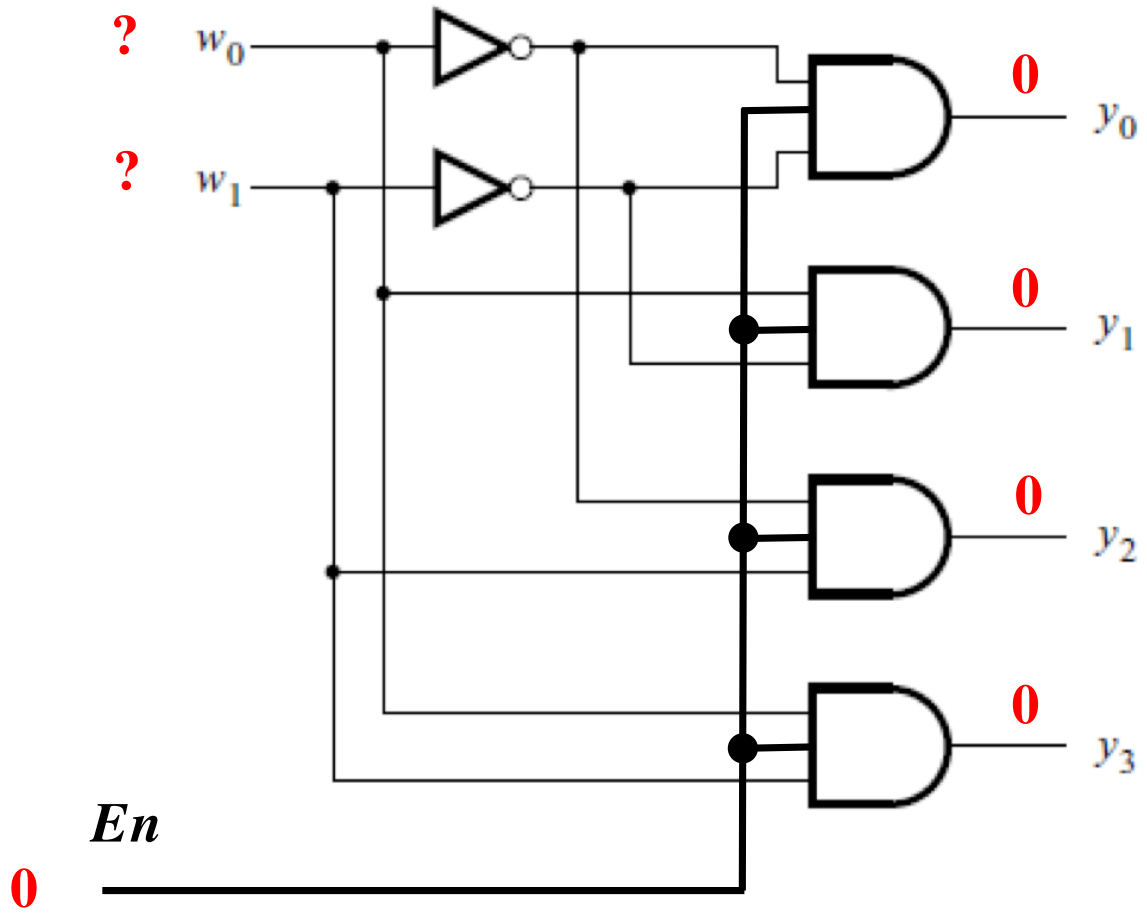
2-to-4 Decoder with Enable Input



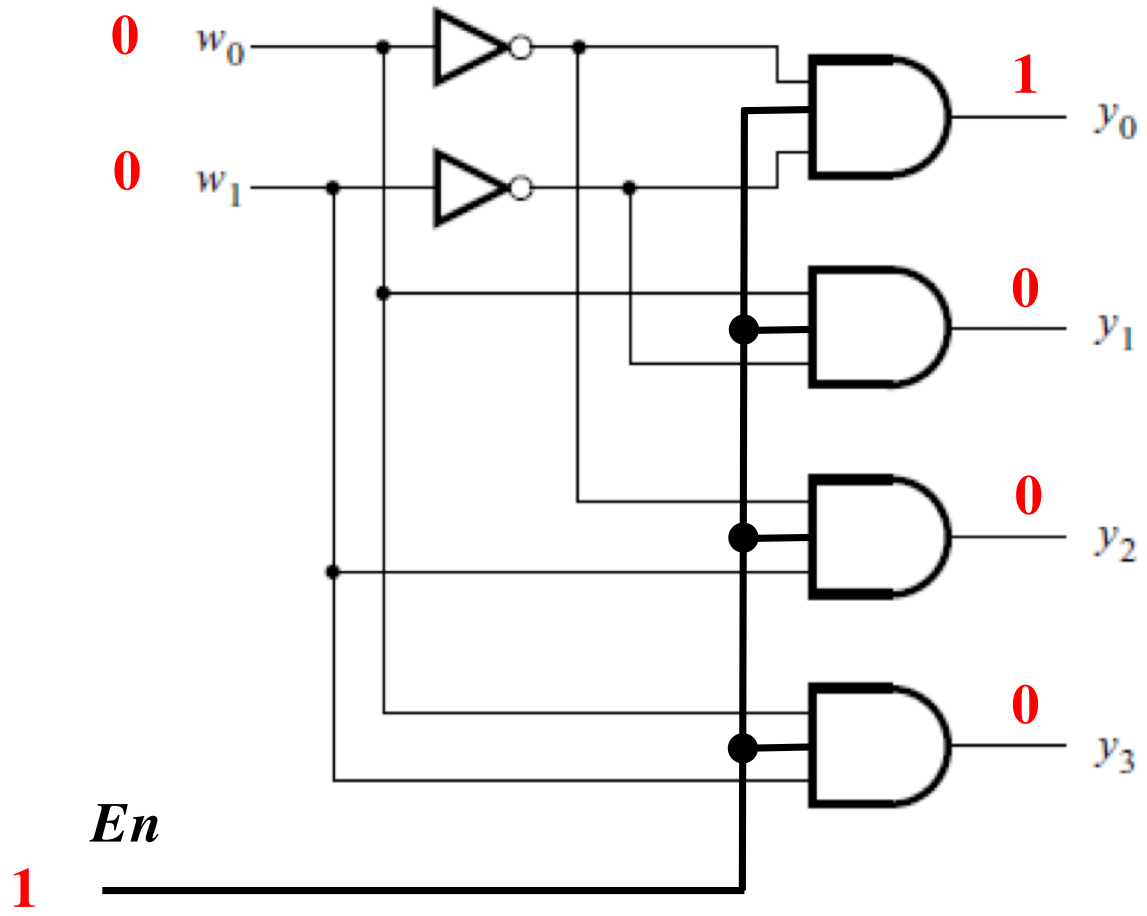
2-to-4 Decoder with Enable Input



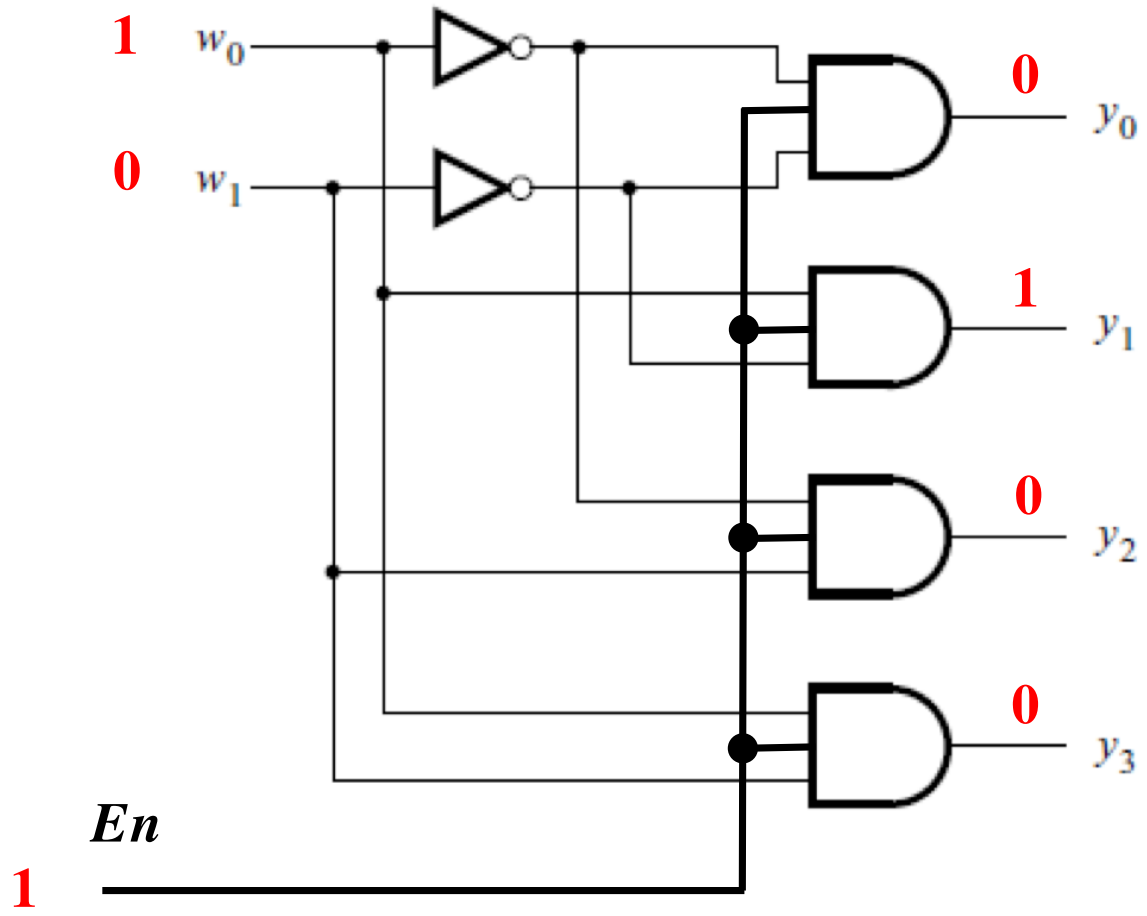
2-to-4 Decoder with Enable Input



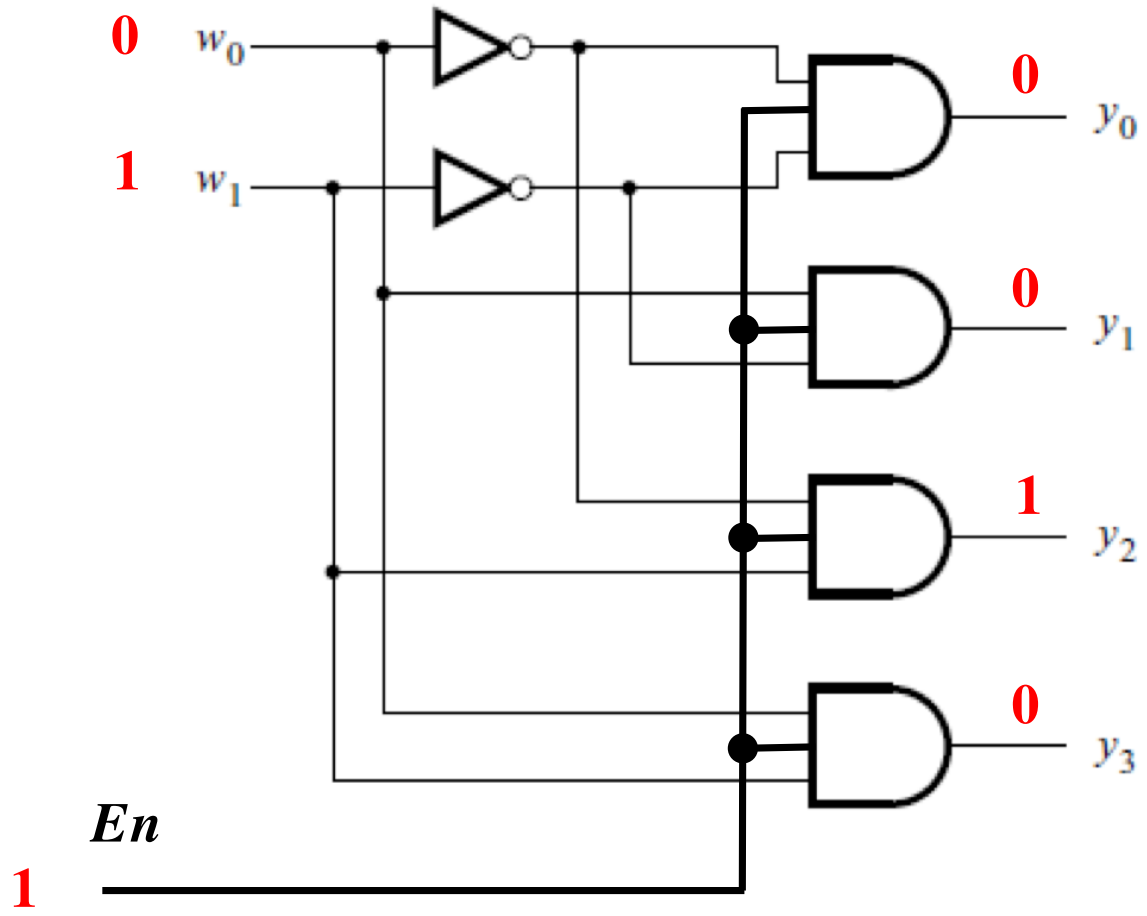
2-to-4 Decoder with Enable Input



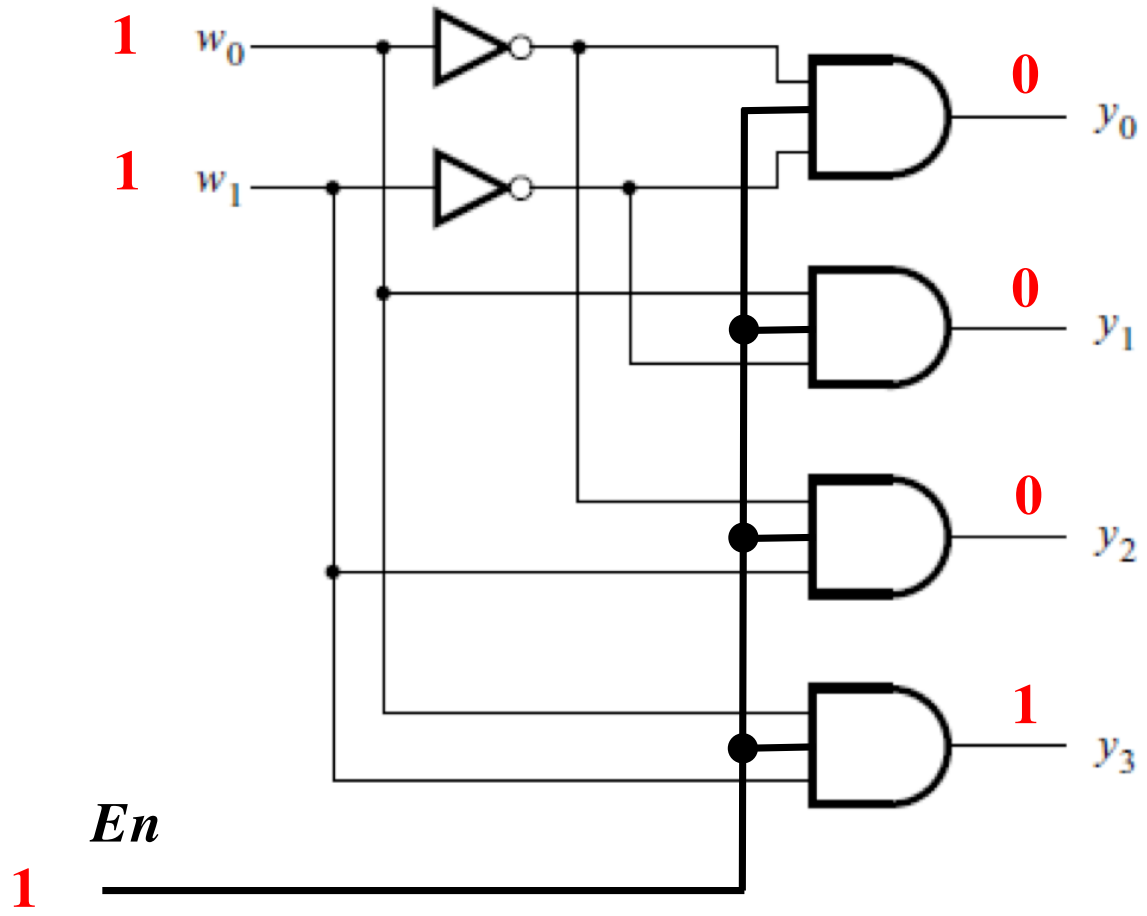
2-to-4 Decoder with Enable Input



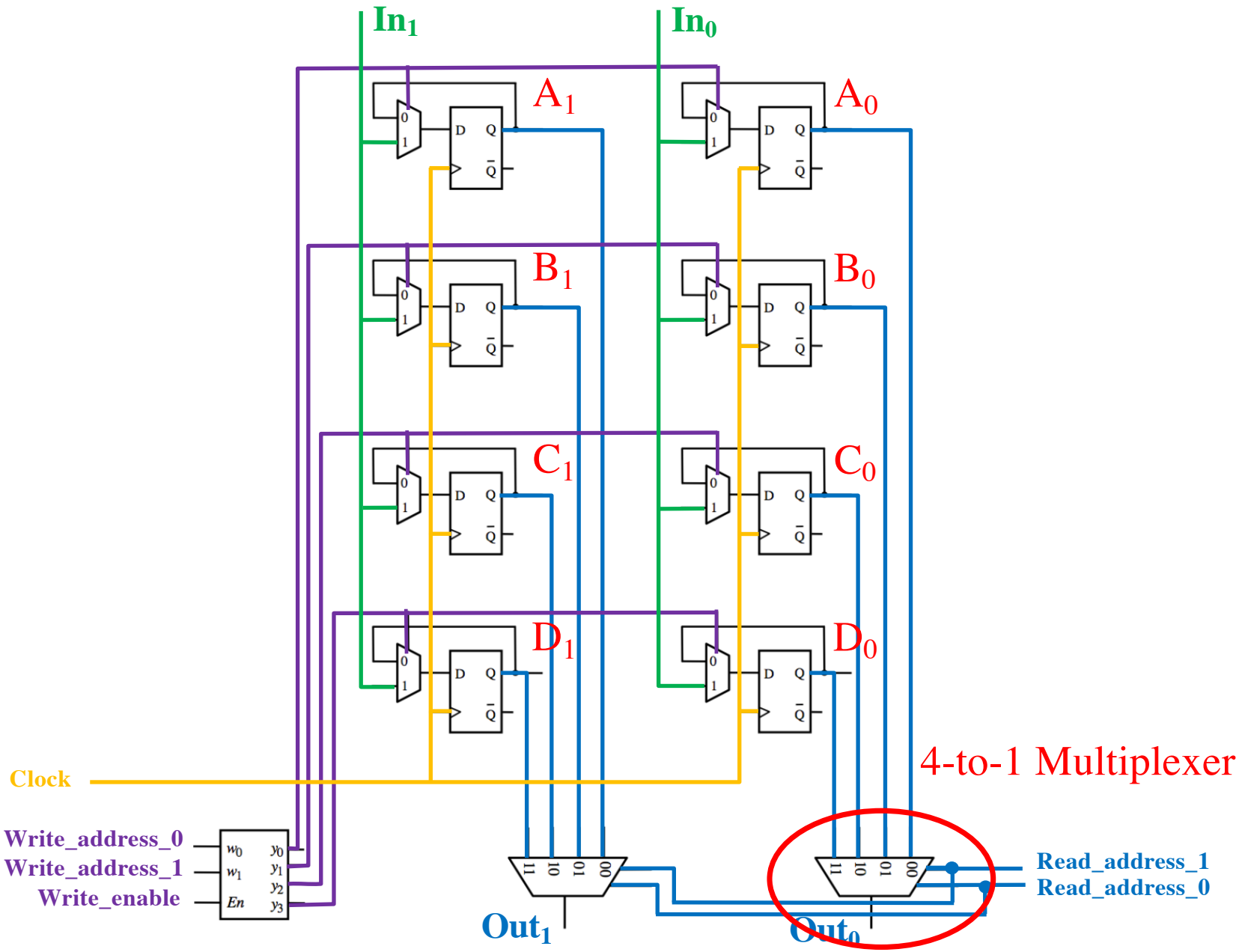
2-to-4 Decoder with Enable Input

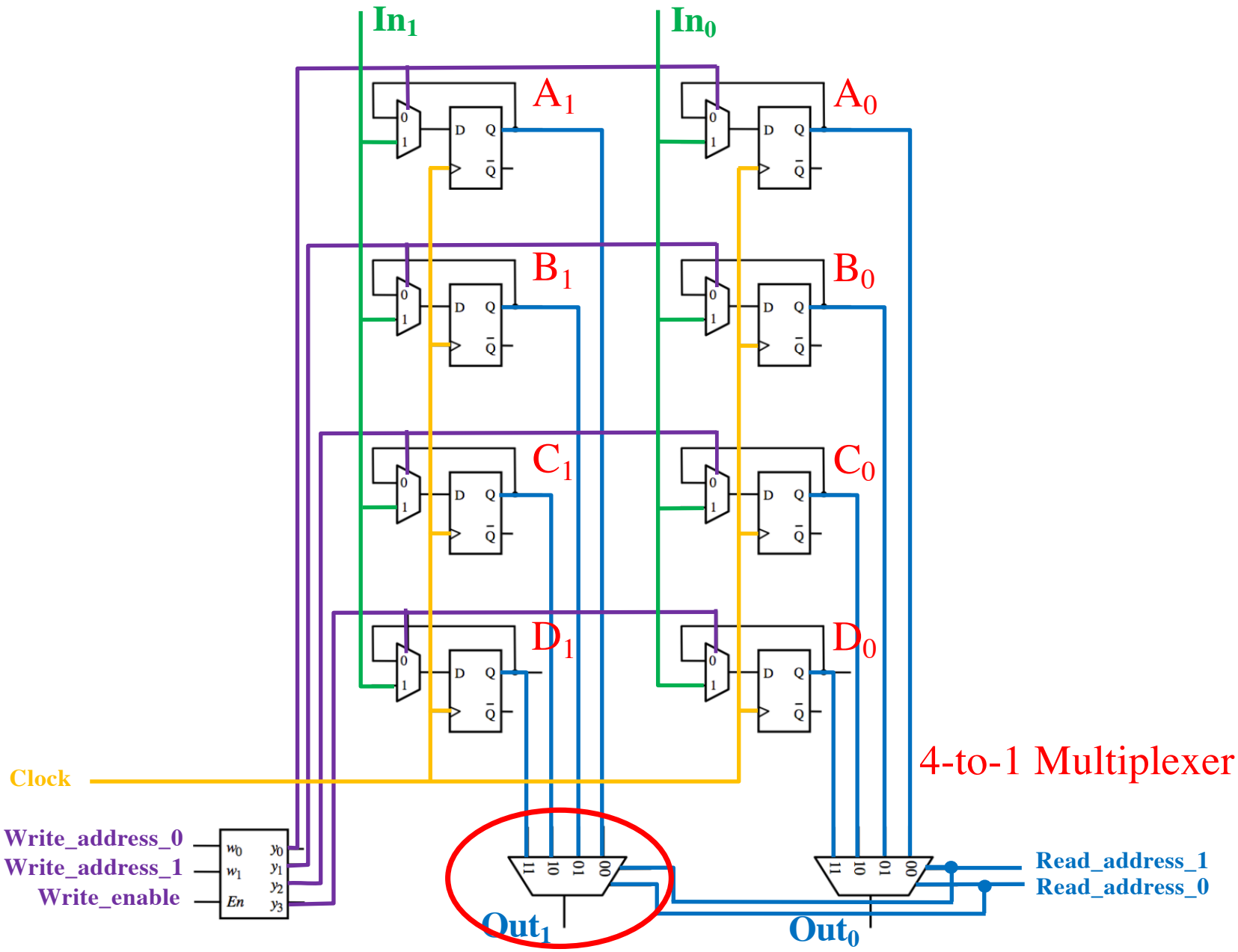


2-to-4 Decoder with Enable Input

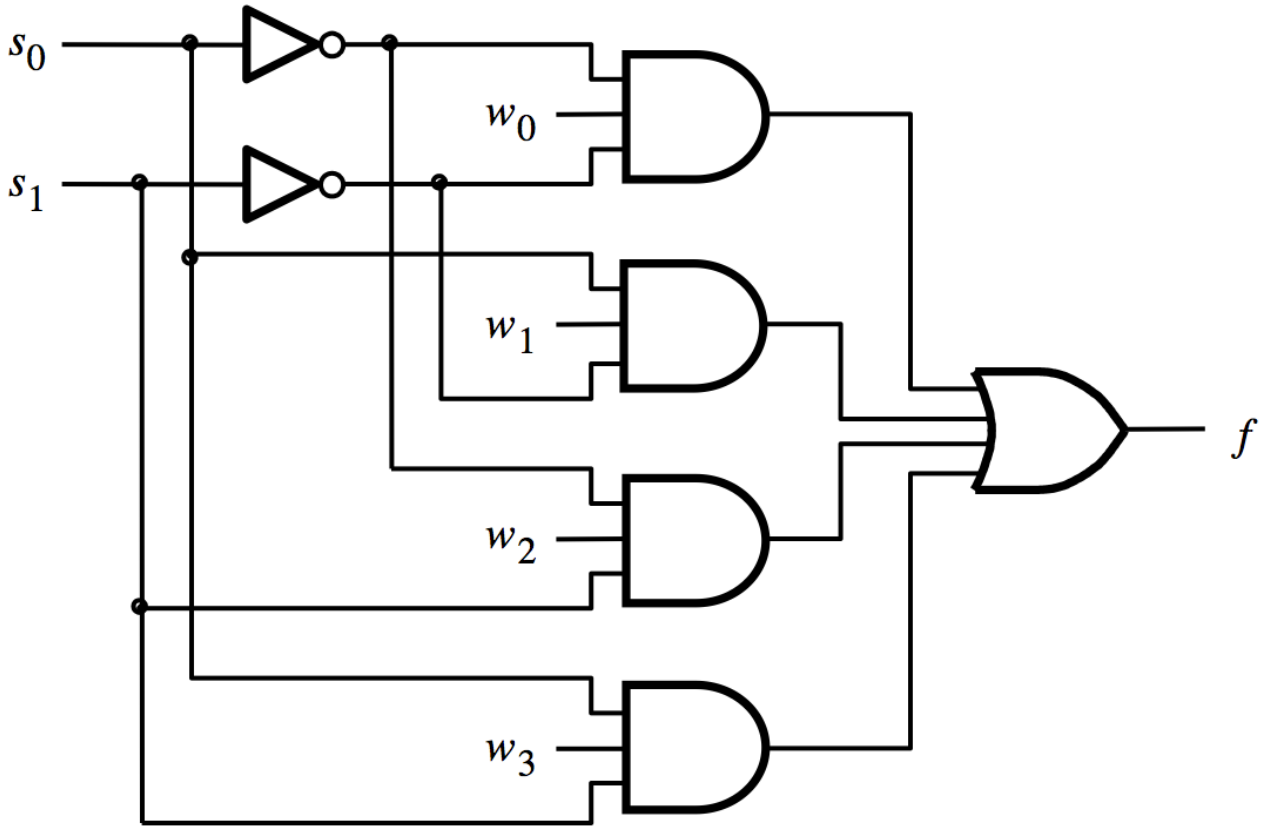


4-to-1 Multiplexer


In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
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
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

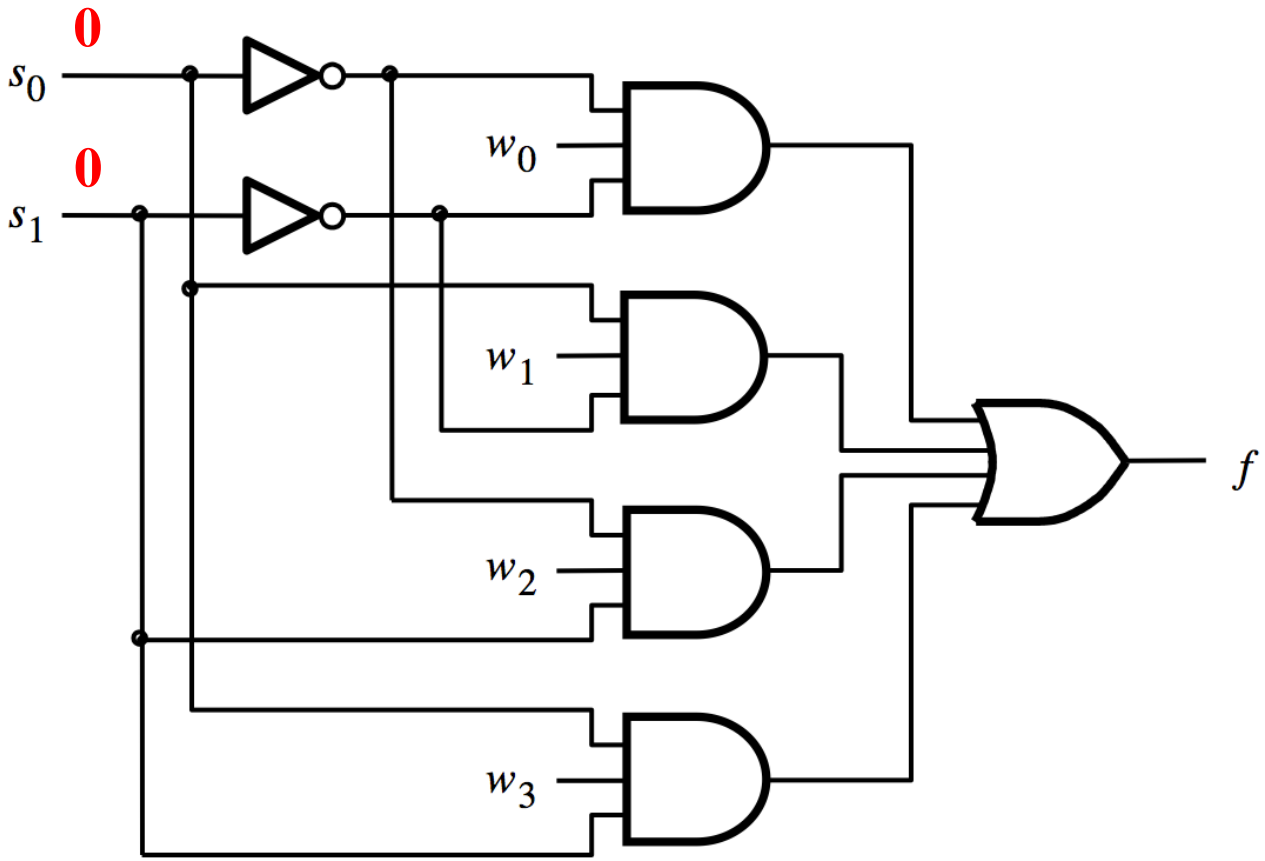
4-to-1 Multiplexer



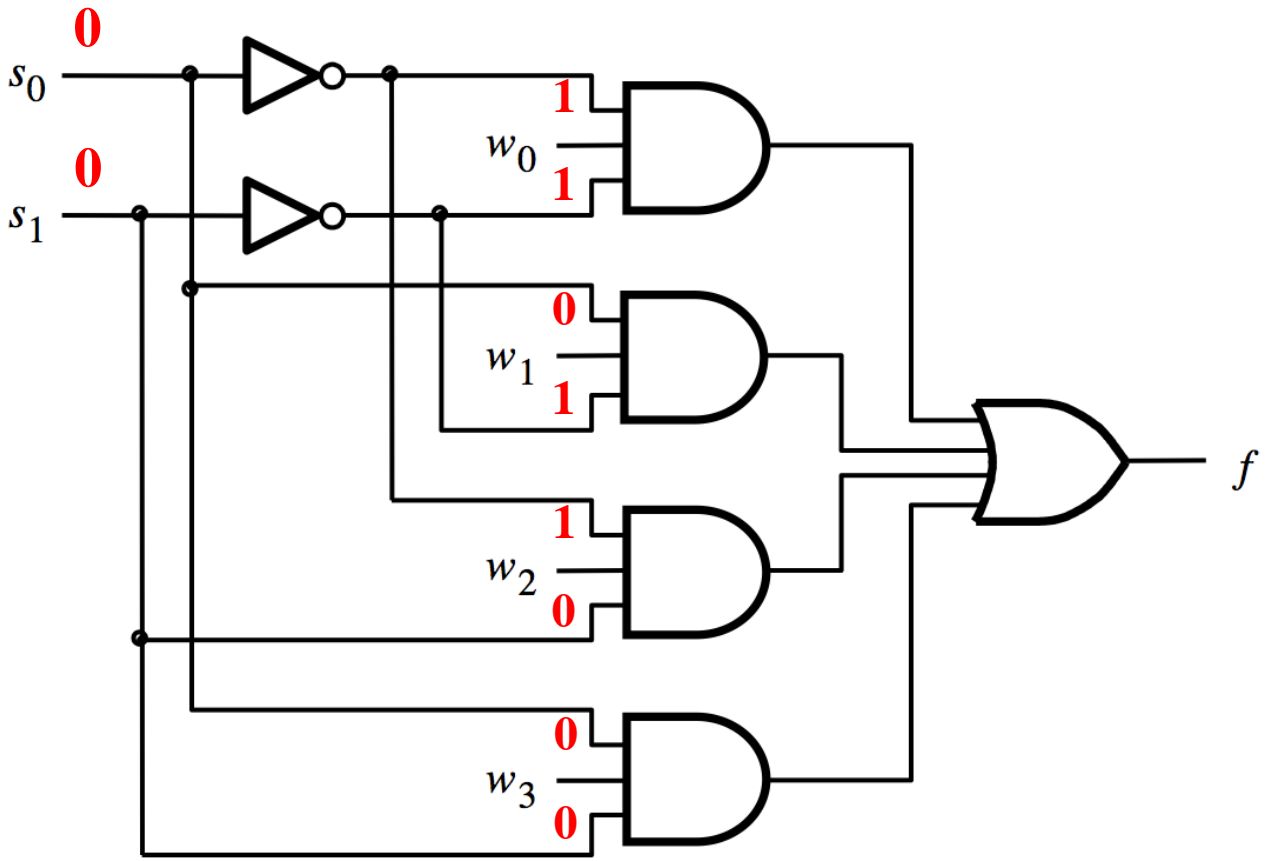
$$f = \overline{s_1}\,\overline{s_0}\,w_0 + \overline{s_1}\,s_0\,w_1 + s_1\,\overline{s_0}\,w_2 + s_1\,s_0\,w_3$$

[Figure 4.2c from the textbook]

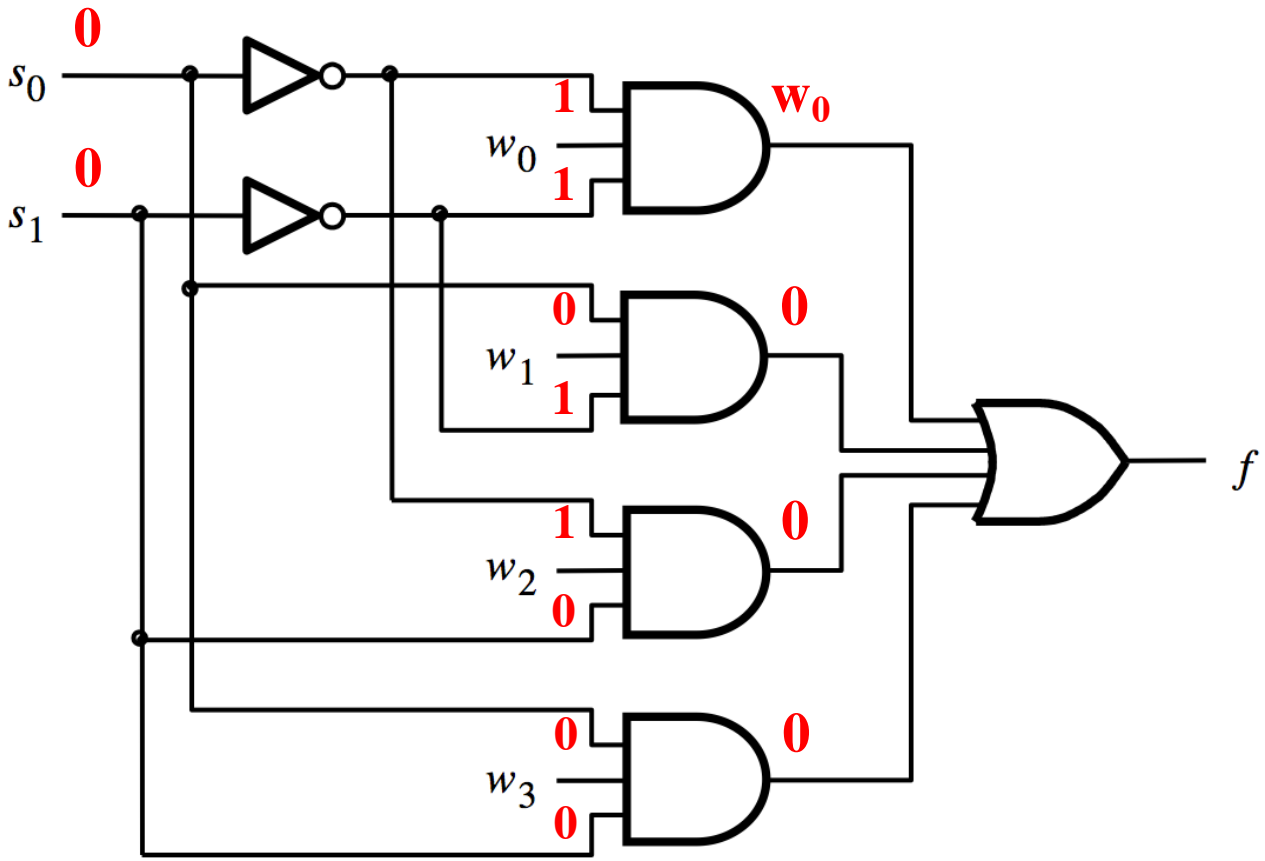
Analysis of the 4-to-1 Multiplexer (s_1=0 and s_0=0)



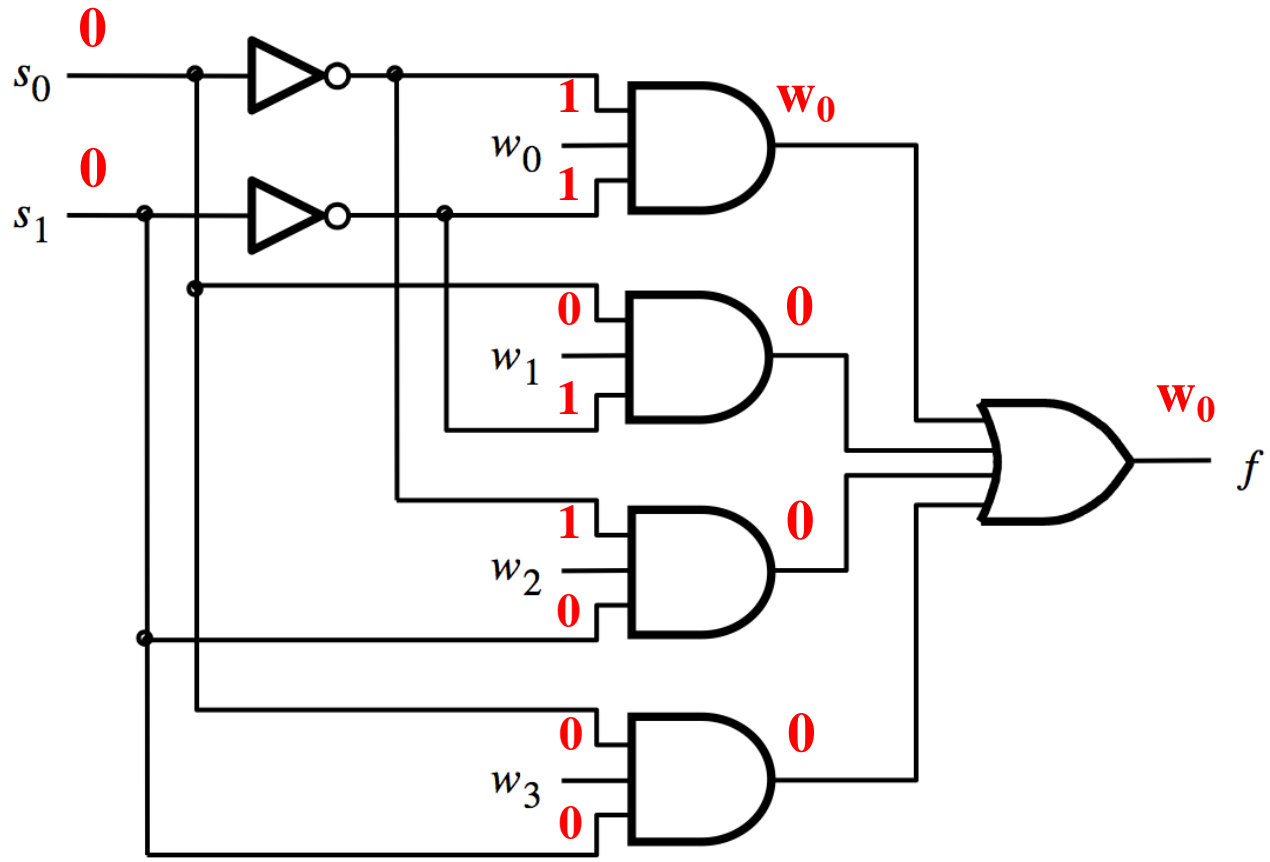
Analysis of the 4-to-1 Multiplexer (s_1=0 and s_0=0)



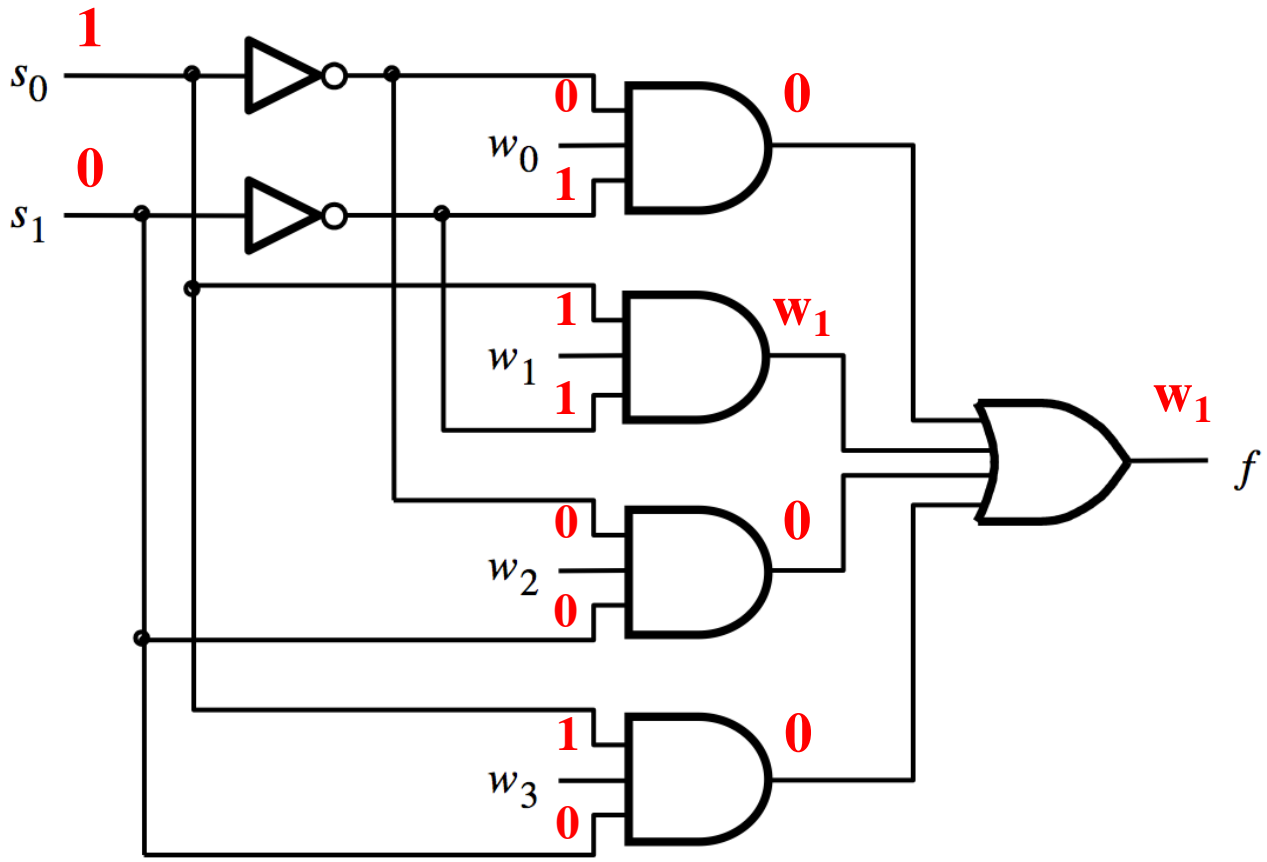
Analysis of the 4-to-1 Multiplexer (s_1=0 and s_0=0)



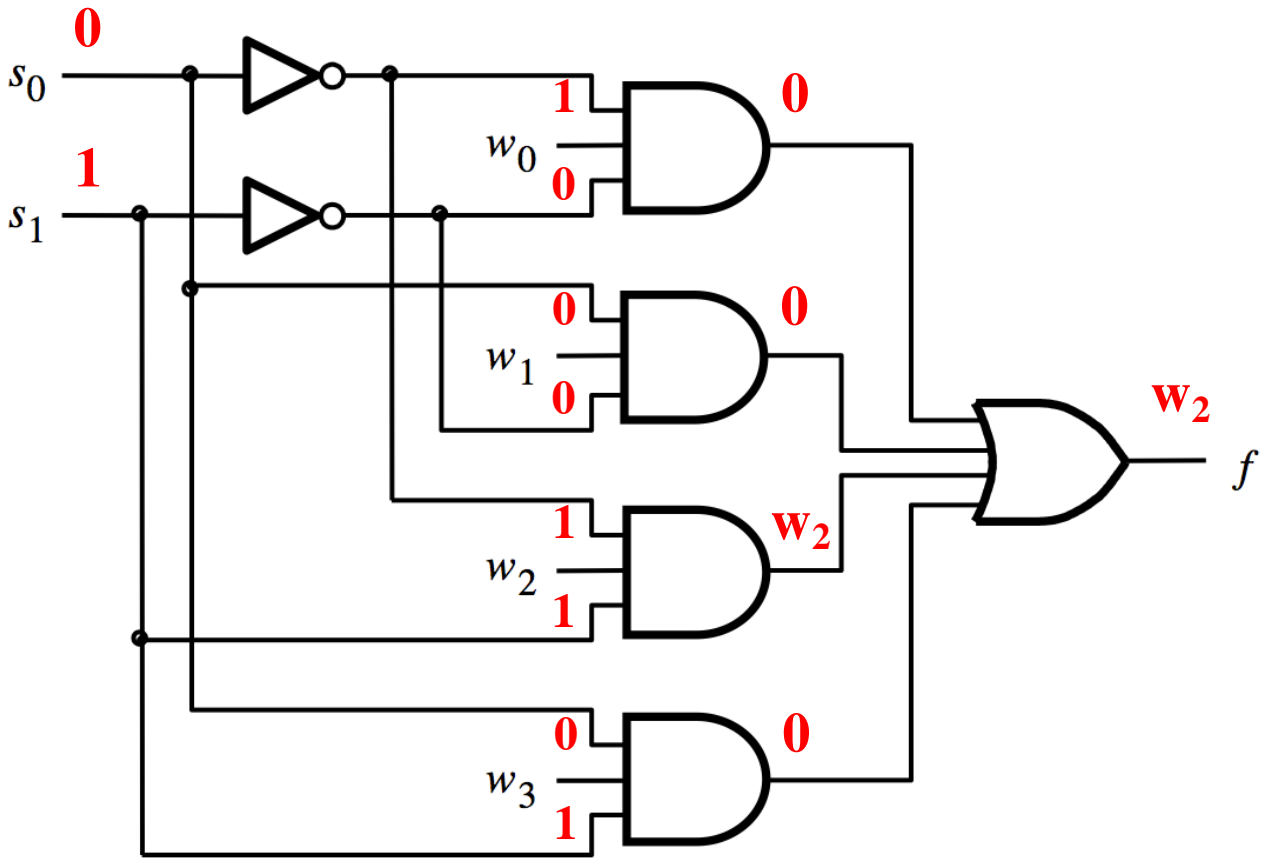
Analysis of the 4-to-1 Multiplexer (s_1=0 and s_0=0)



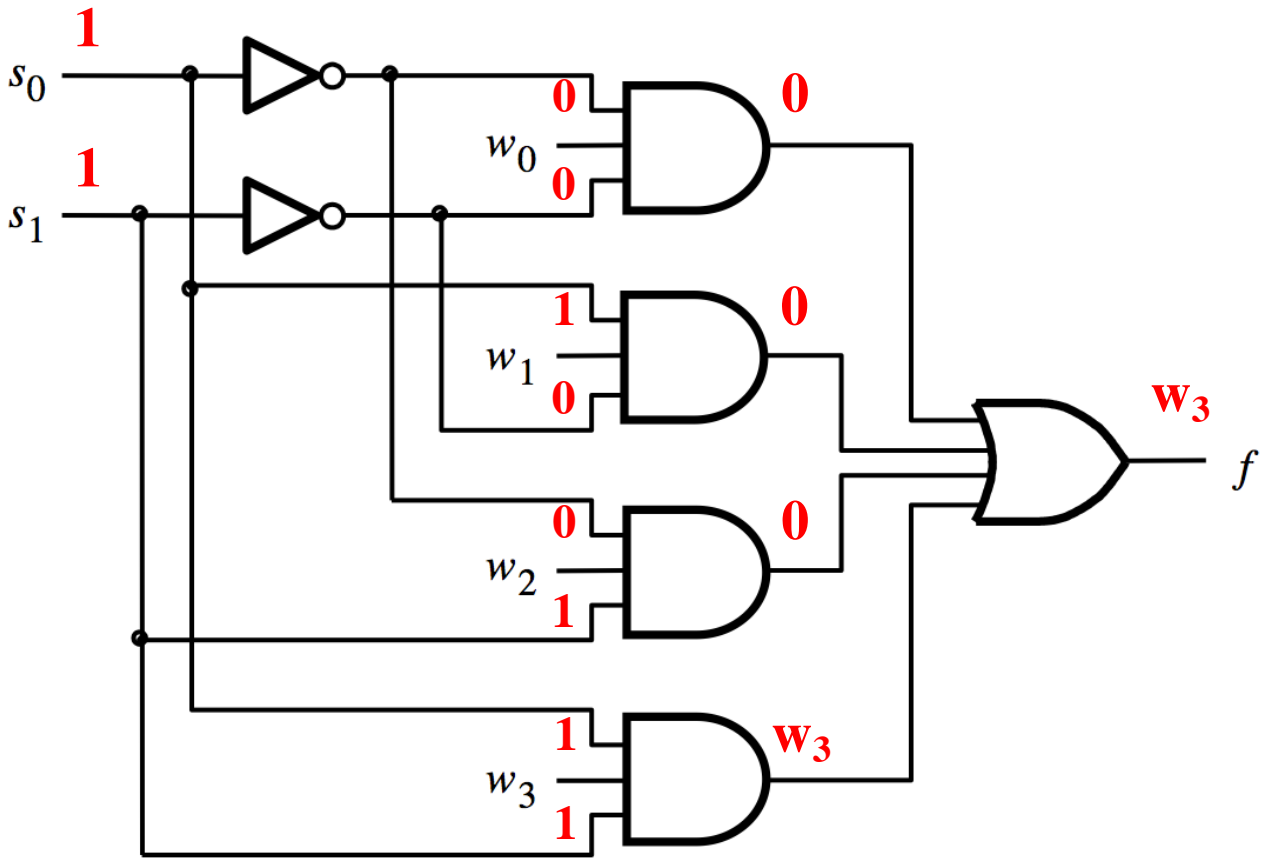
Analysis of the 4-to-1 Multiplexer (s_1=0 and s_0=1)



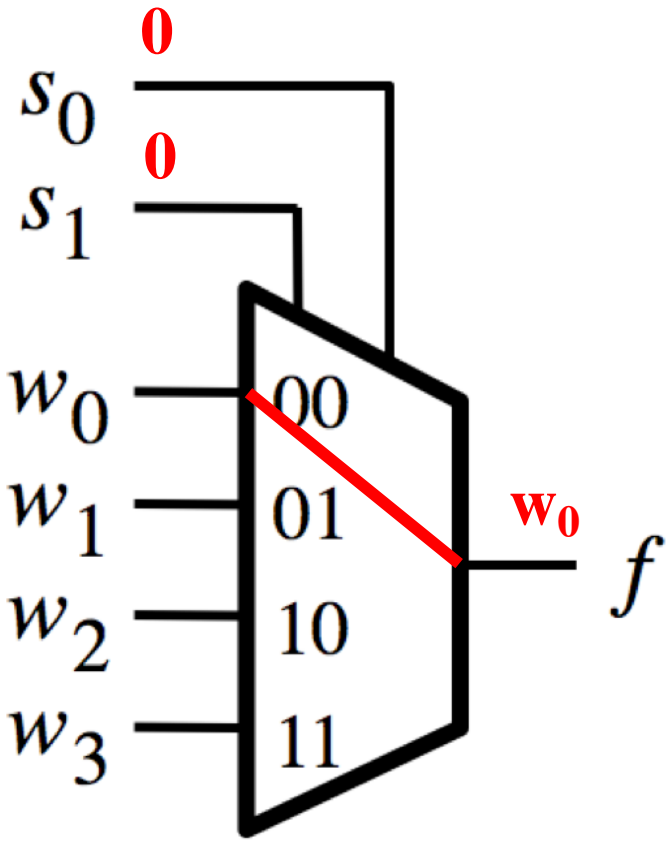
Analysis of the 4-to-1 Multiplexer
(s_1=1 and s_0=0)



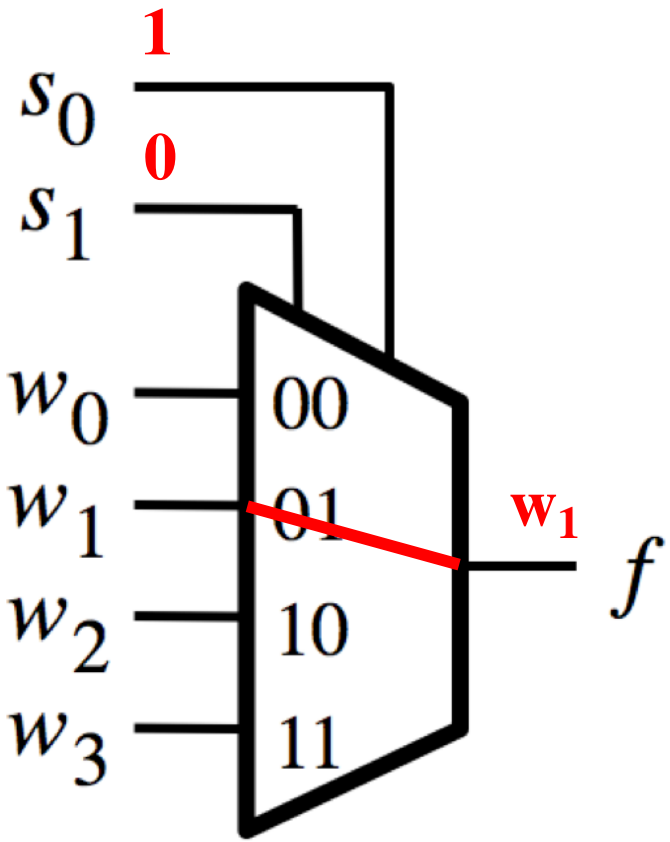
Analysis of the 4-to-1 Multiplexer
(s_1=1 and s_0=1)



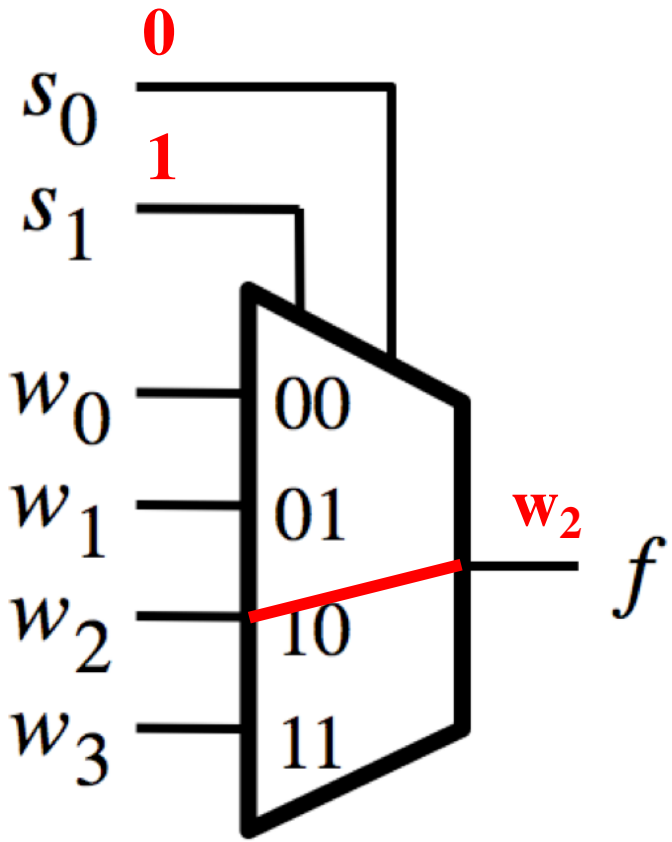
Analysis of the 4-to-1 Multiplexer
(s_1=0 and s_0=0)



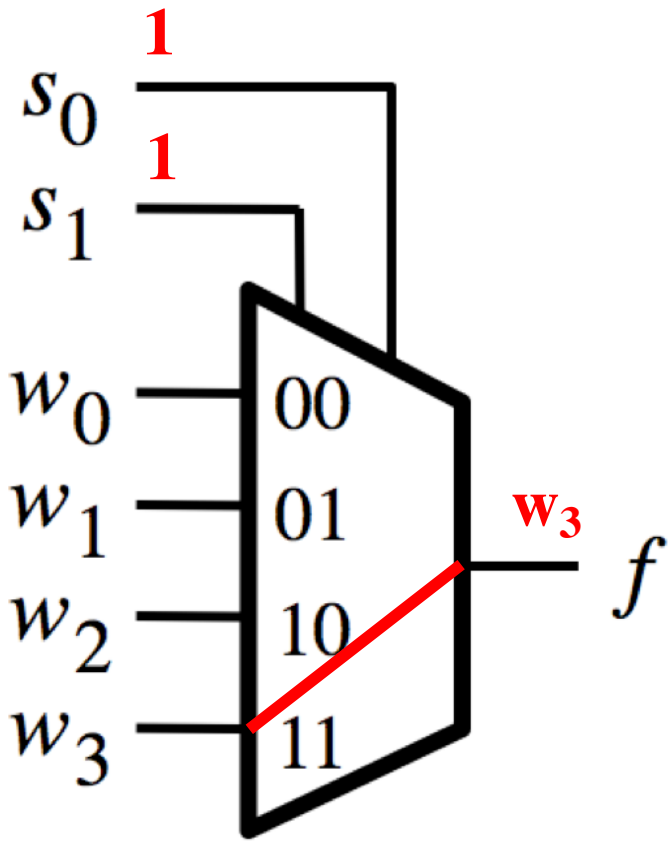
Analysis of the 4-to-1 Multiplexer
(s_1=0 and s_0=1)



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

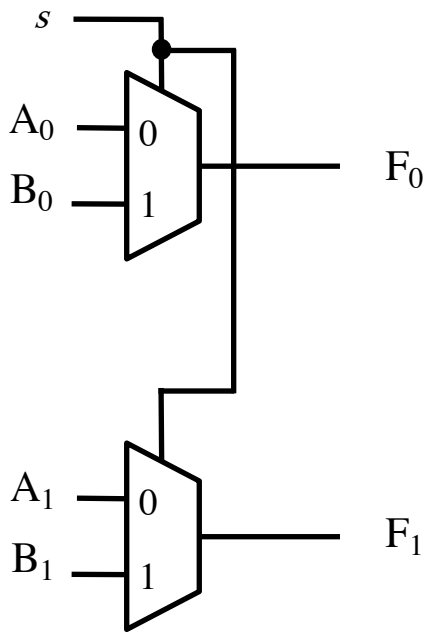


Analysis of the 4-to-1 Multiplexer (s_1=1 and s_0=1)

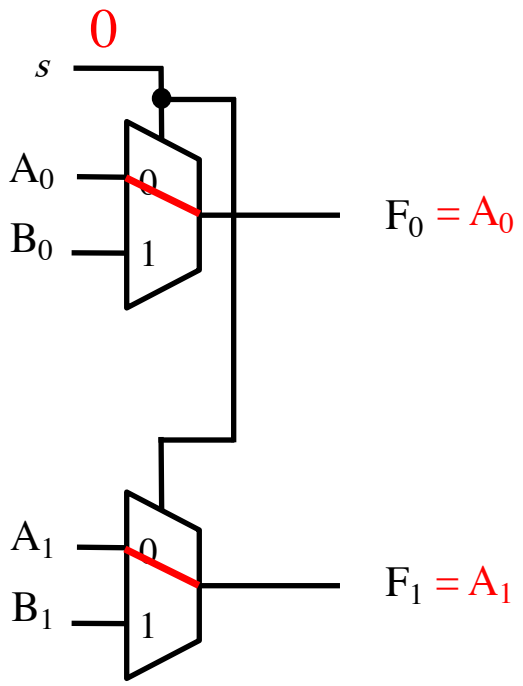


Multiplexer Tricks
(select one of two 2-bit numbers)

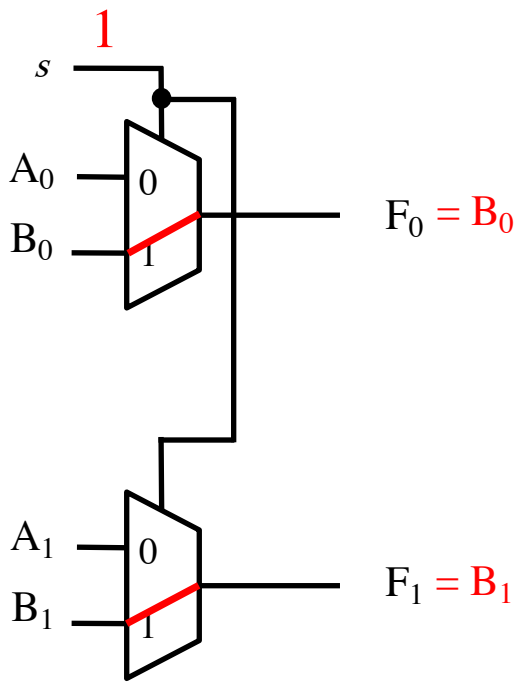
Select Either $A=A_1A_0$ or $B=B_1B_0$



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

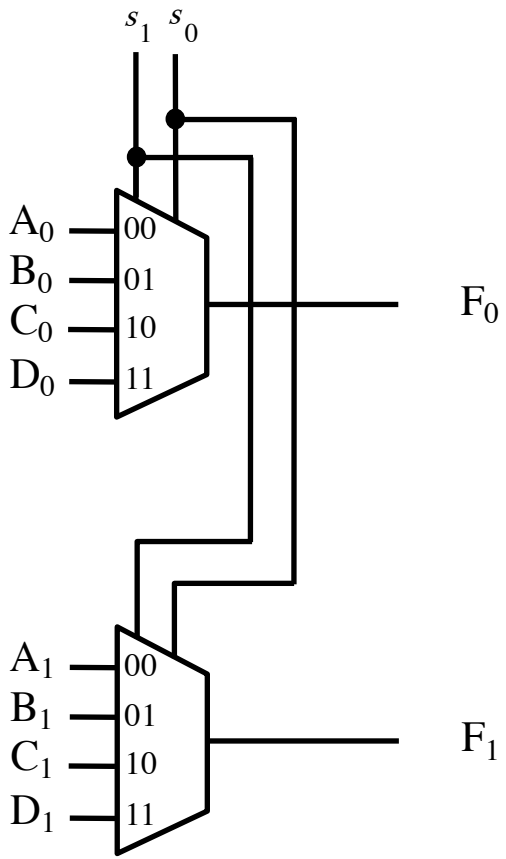


Select Either $A=A_1A_0$ or $B=B_1B_0$

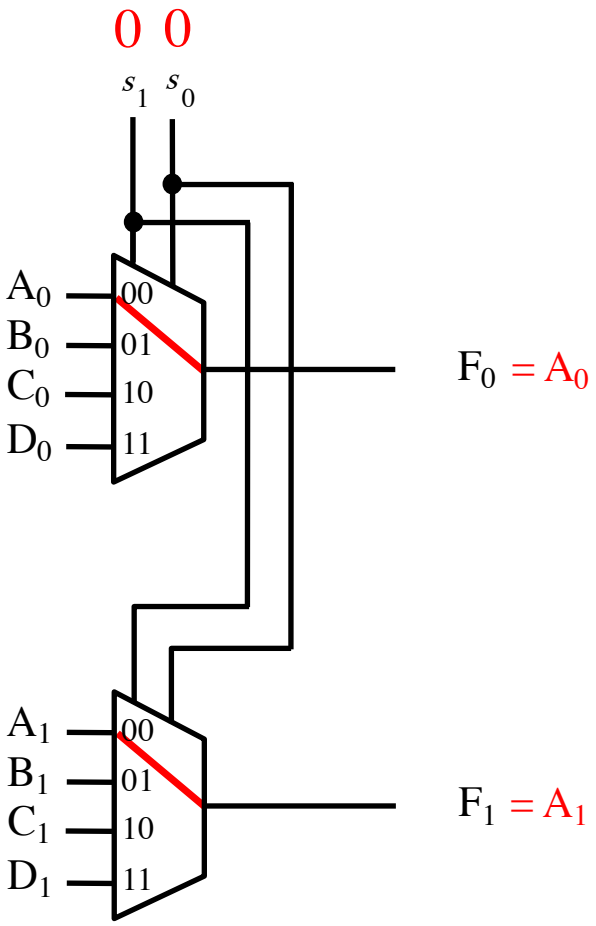


Multiplexer Tricks
(select one of four 2-bit numbers)

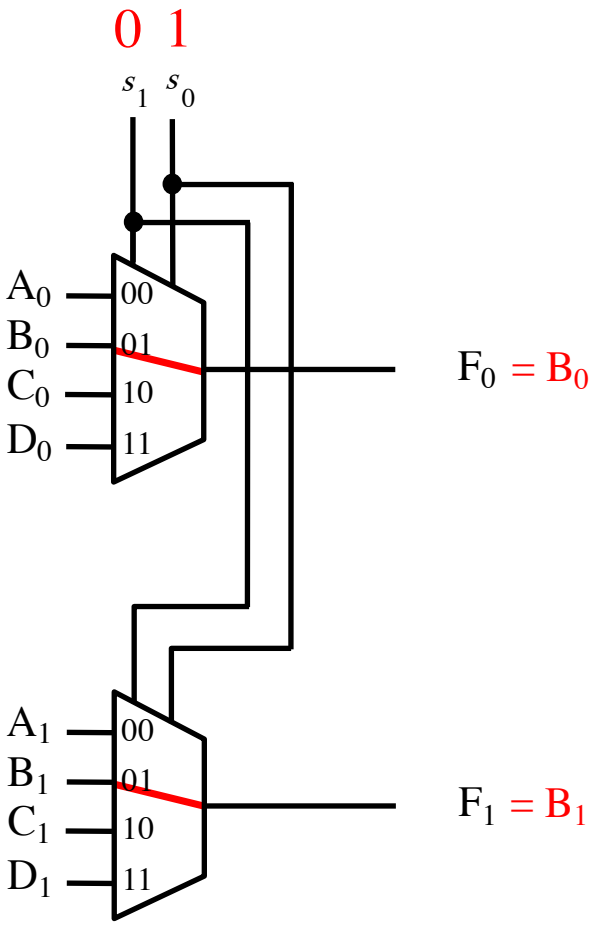
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



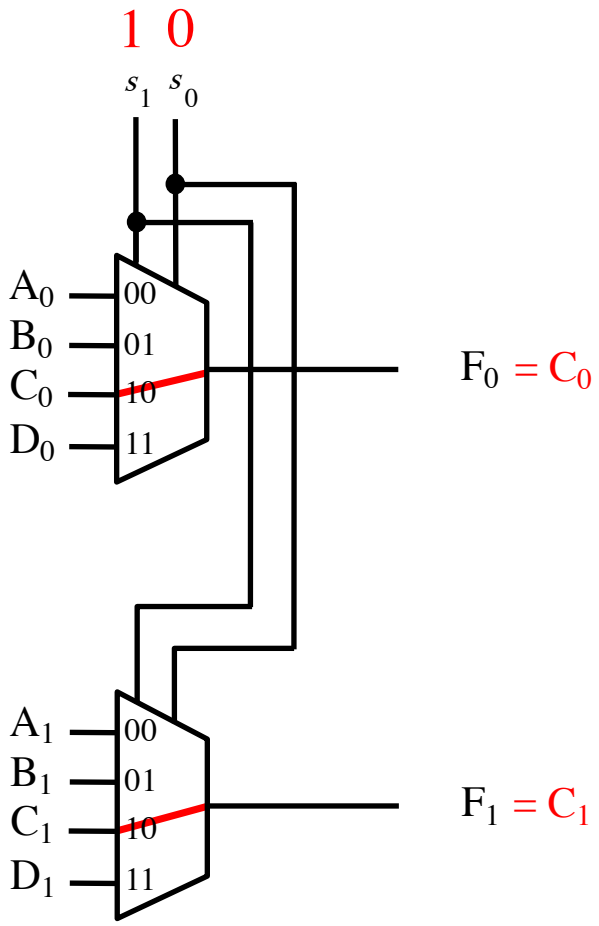
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



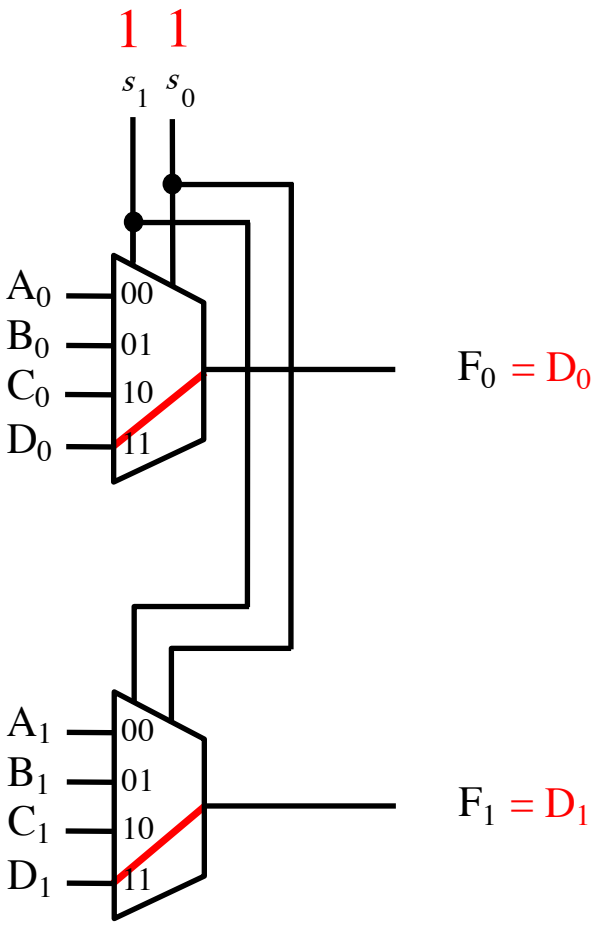
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



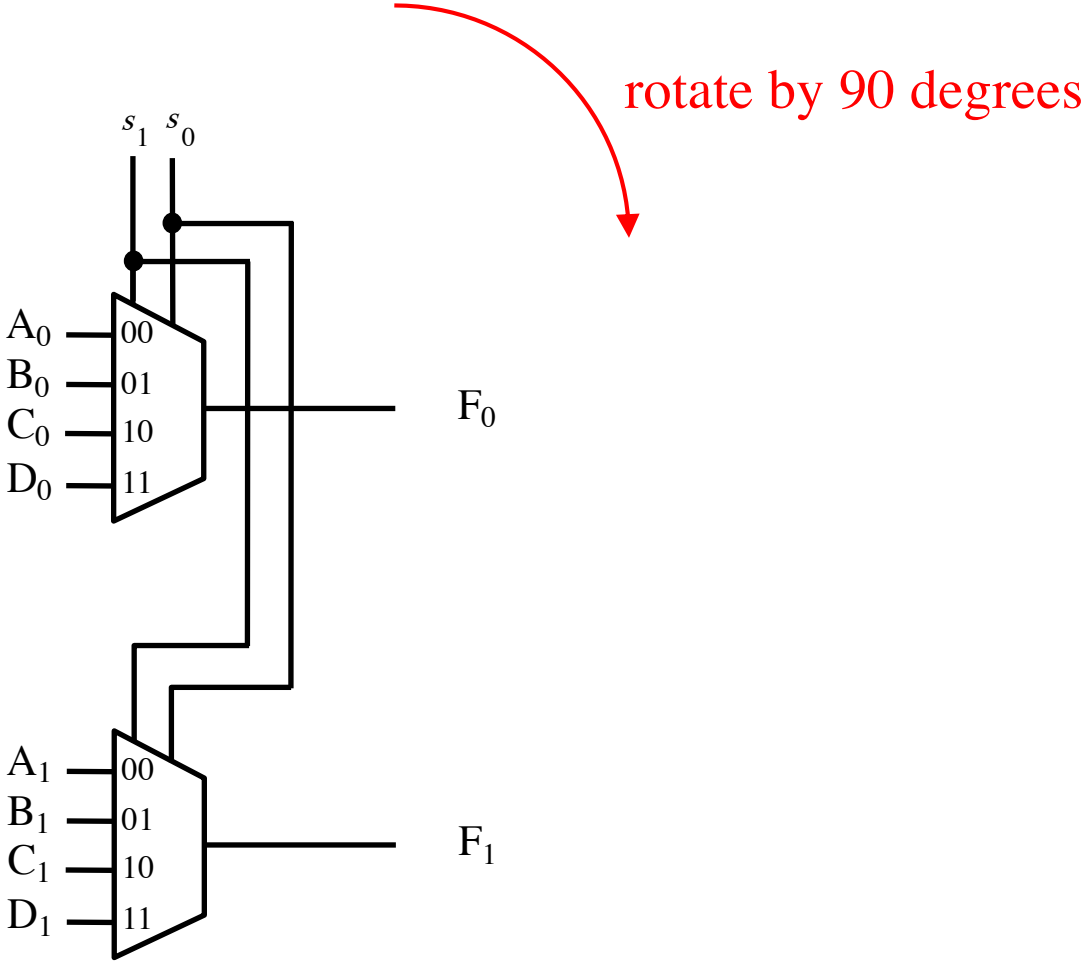
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



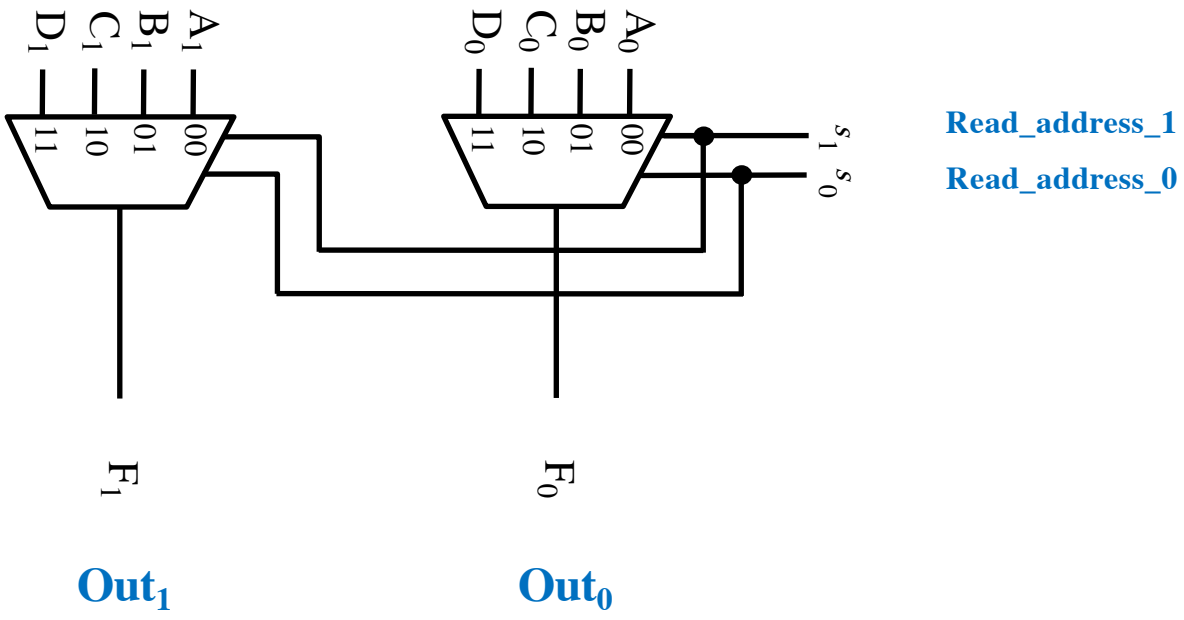
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$

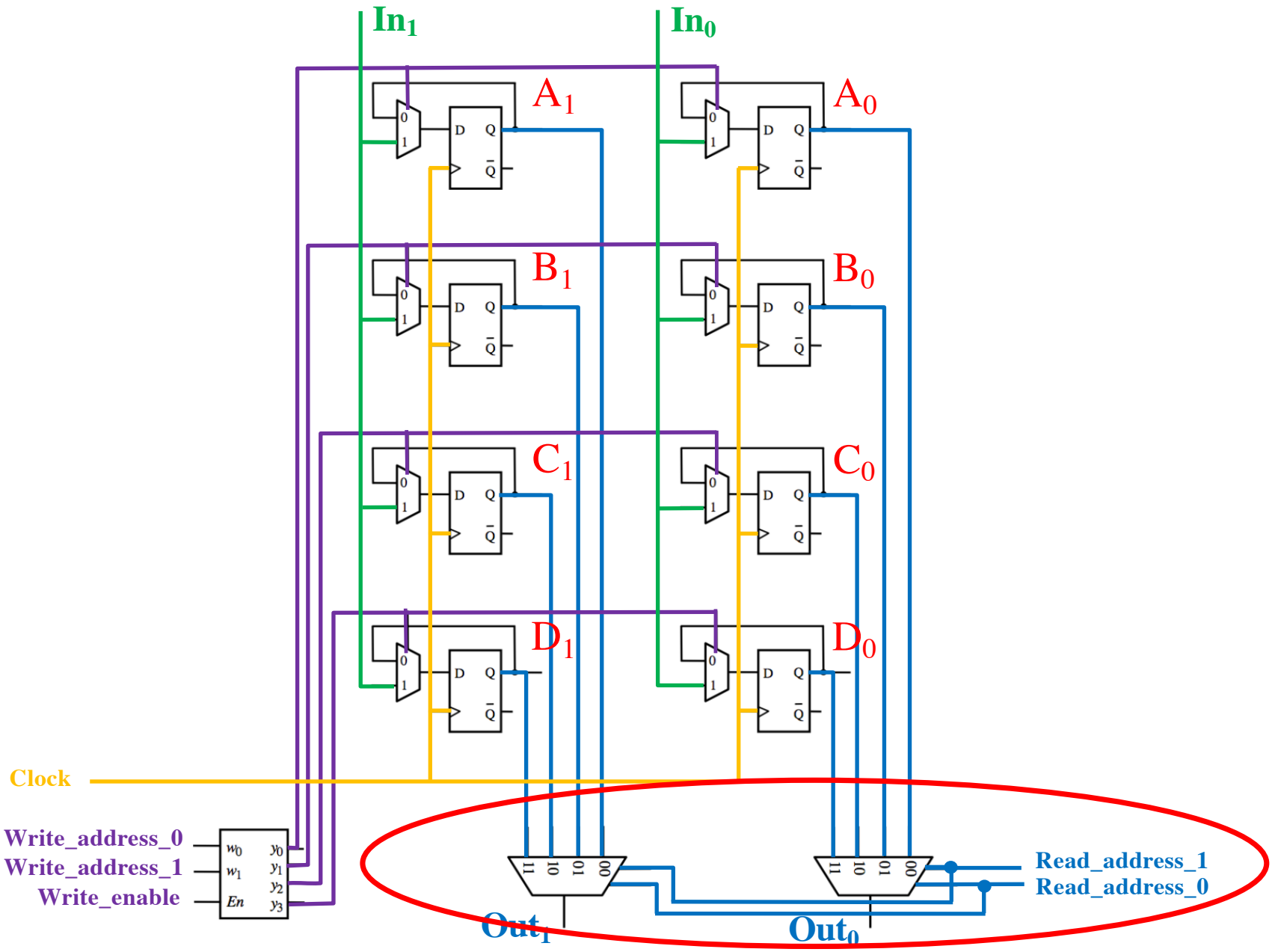


Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$

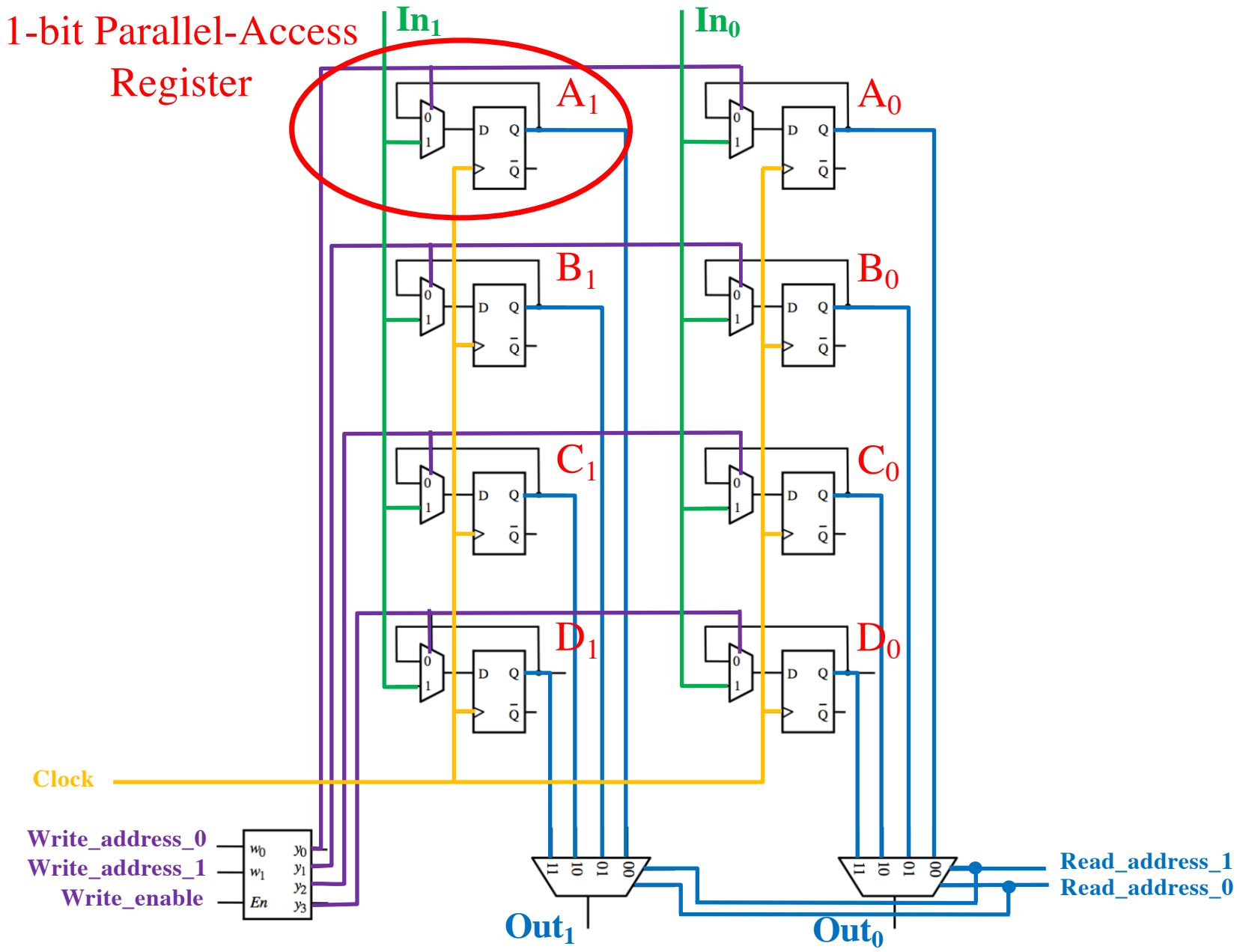


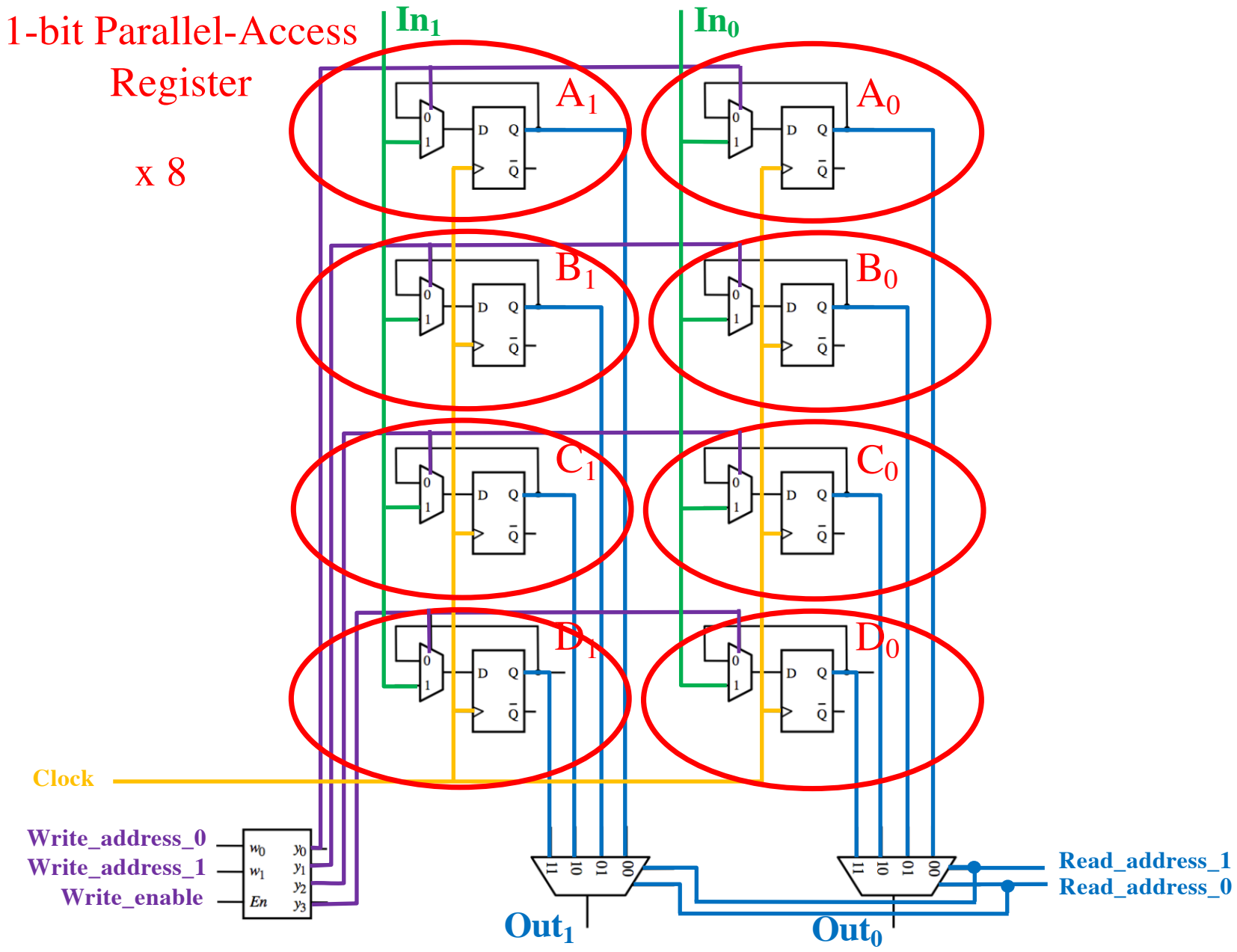
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



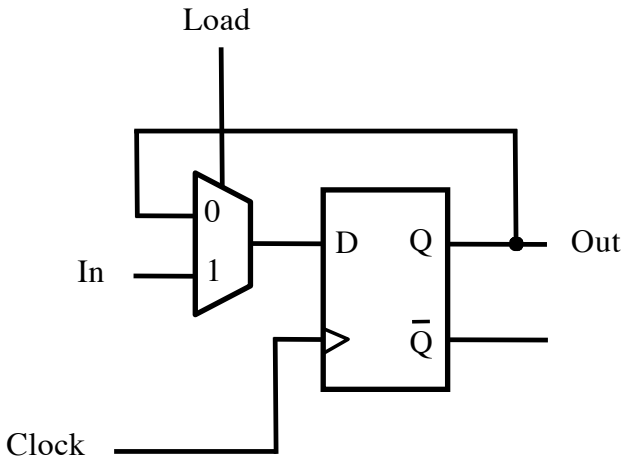

In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
D
Q
Q̄
0
1
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
Read_address_1
Read_address_0

1-bit Parallel-Access Register

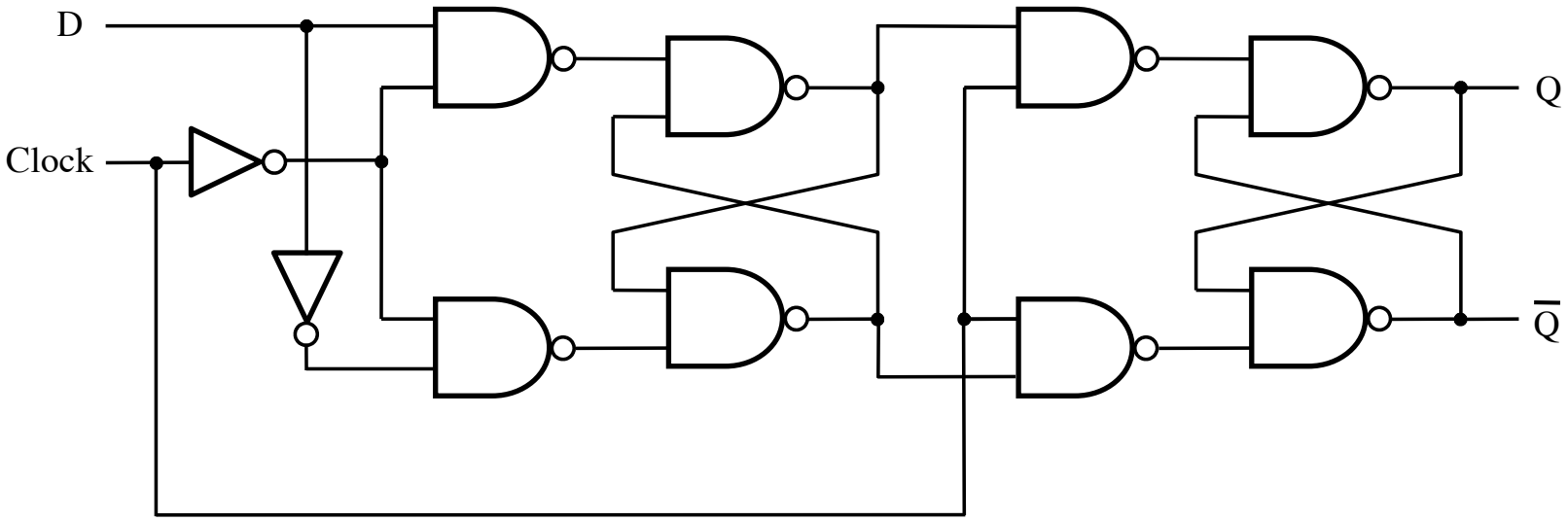

1-bit Parallel-Access Register
In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
D
Q
Q̄
0
1
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
Read_address_1
Read_address_0


1-bit Parallel-Access Register
x 8
In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
D
Q
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

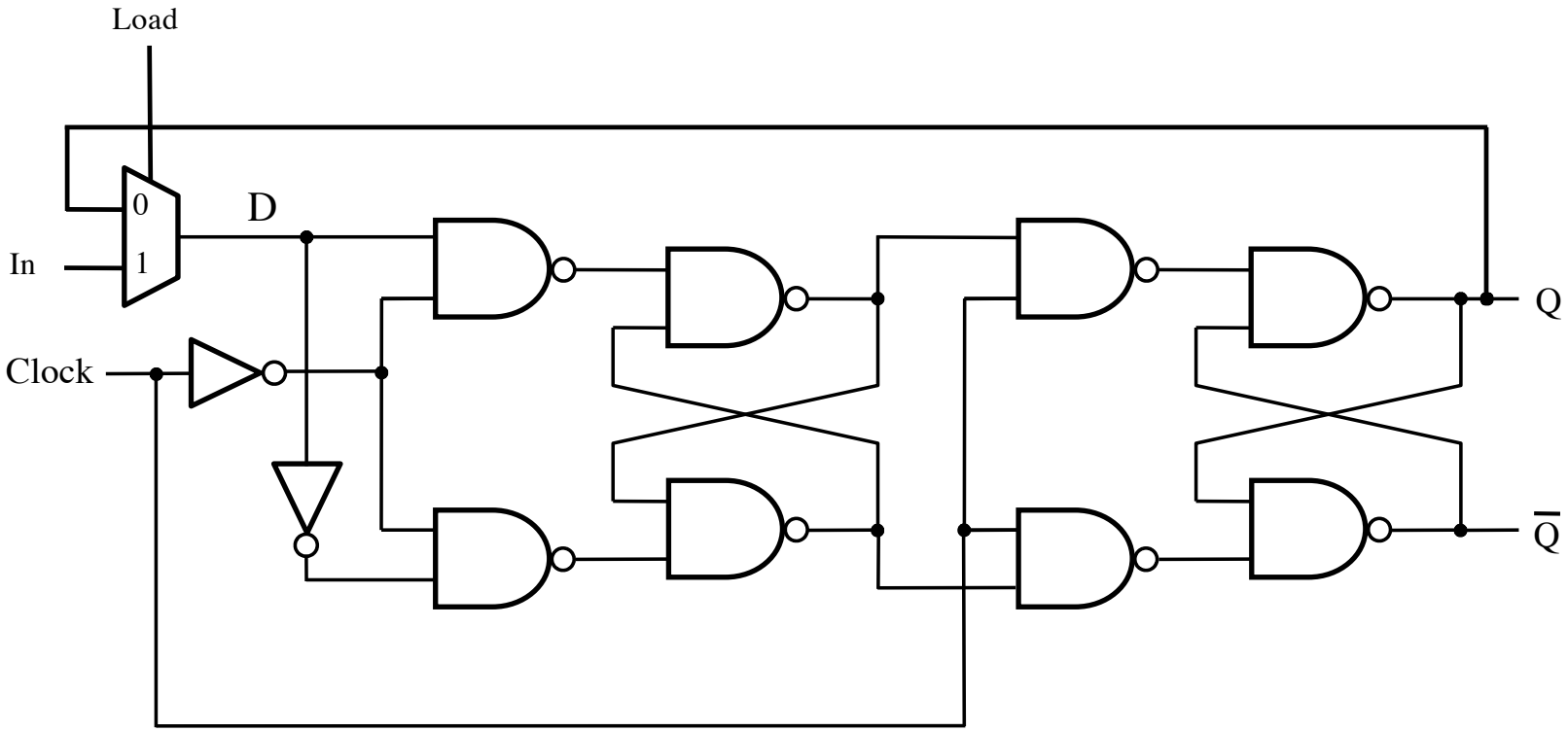
1-Bit Parallel-Access Register



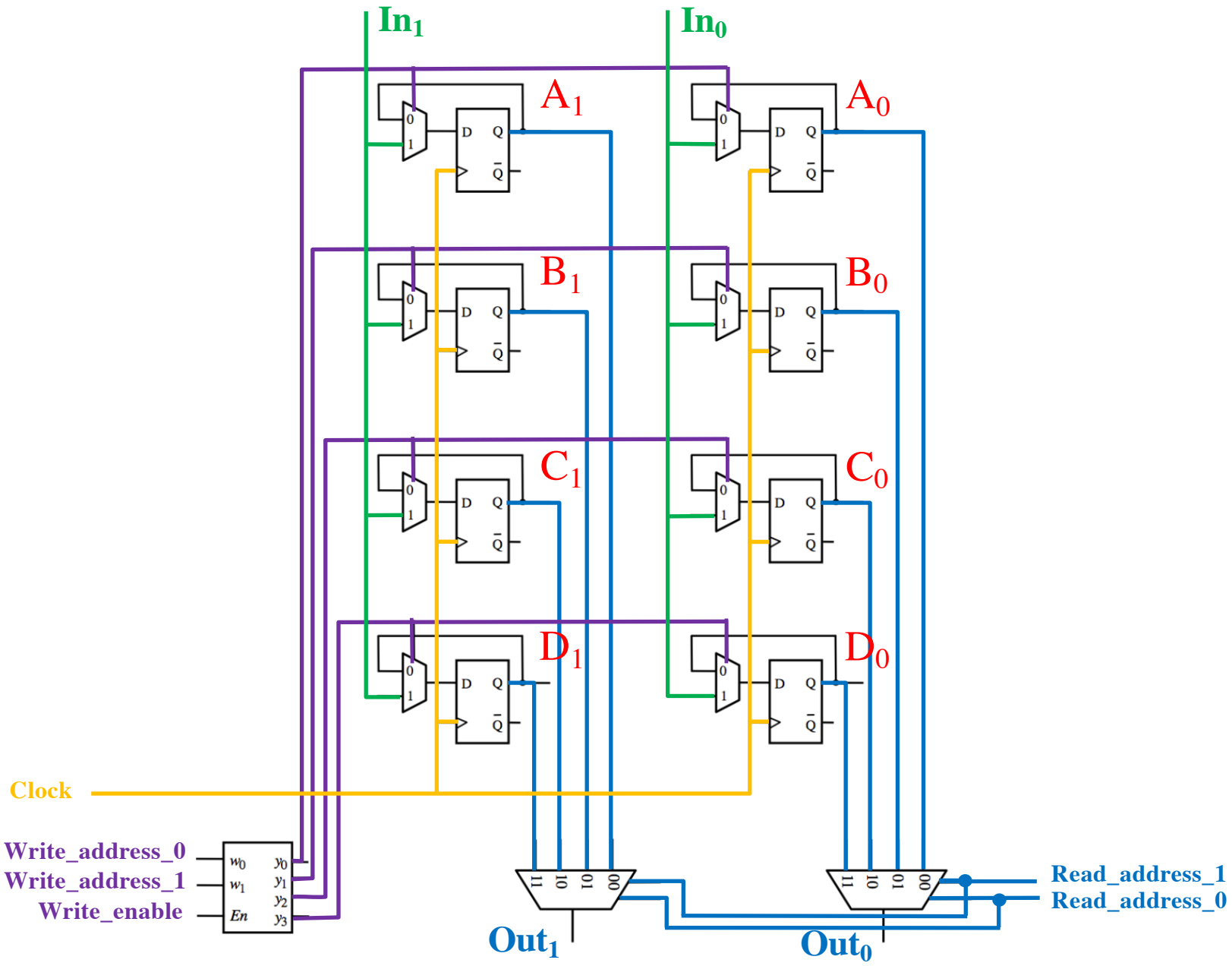
Positive-Edge-Triggered D Flip-Flop



1-bit Parallel-Access Register

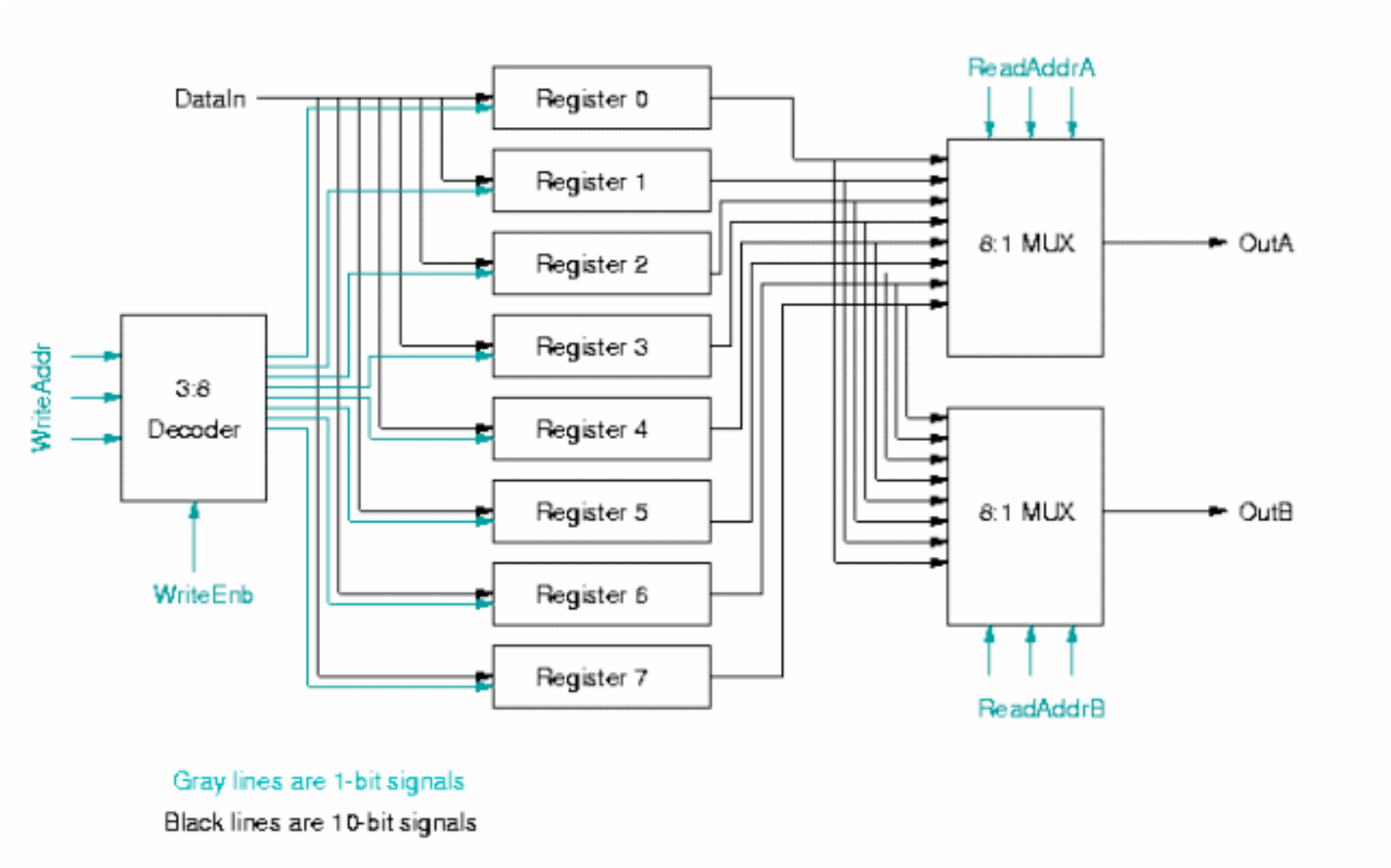


Putting it all Together


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

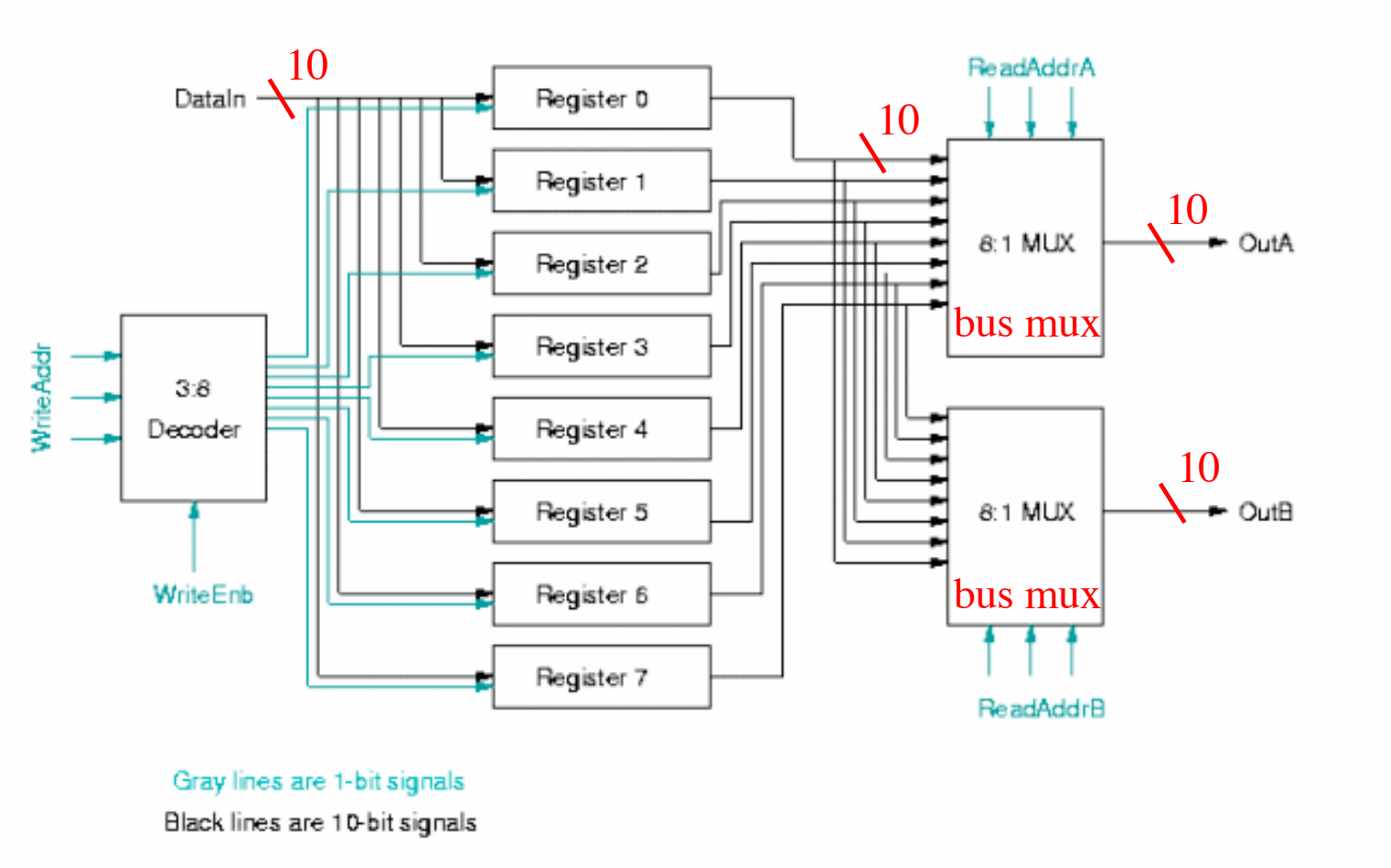
Register file with eight 10-bit registers and two read ports

Register file with 2 read ports



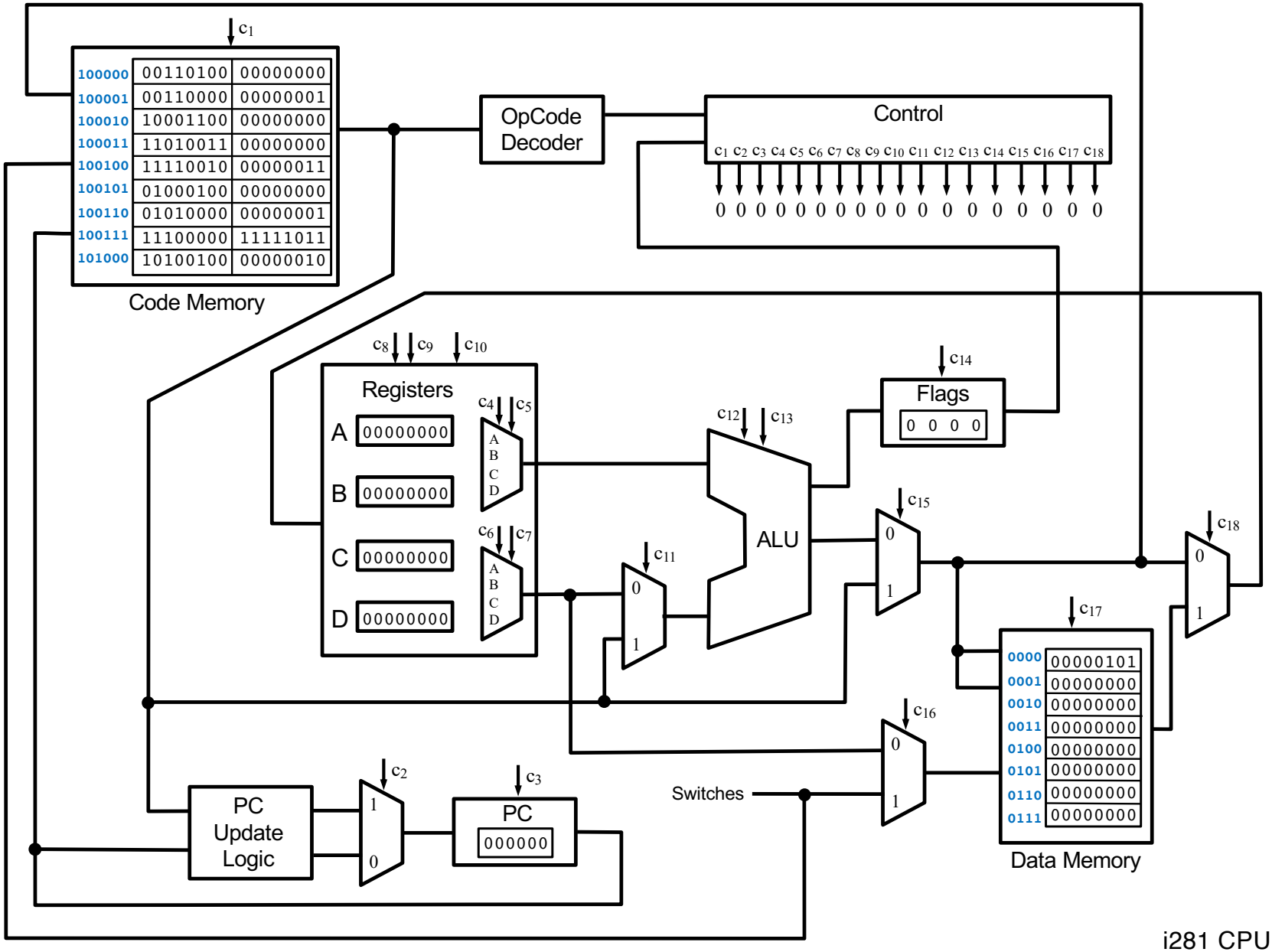
[https://jindongpu.wordpress.com/2012/03/07/register-file/]

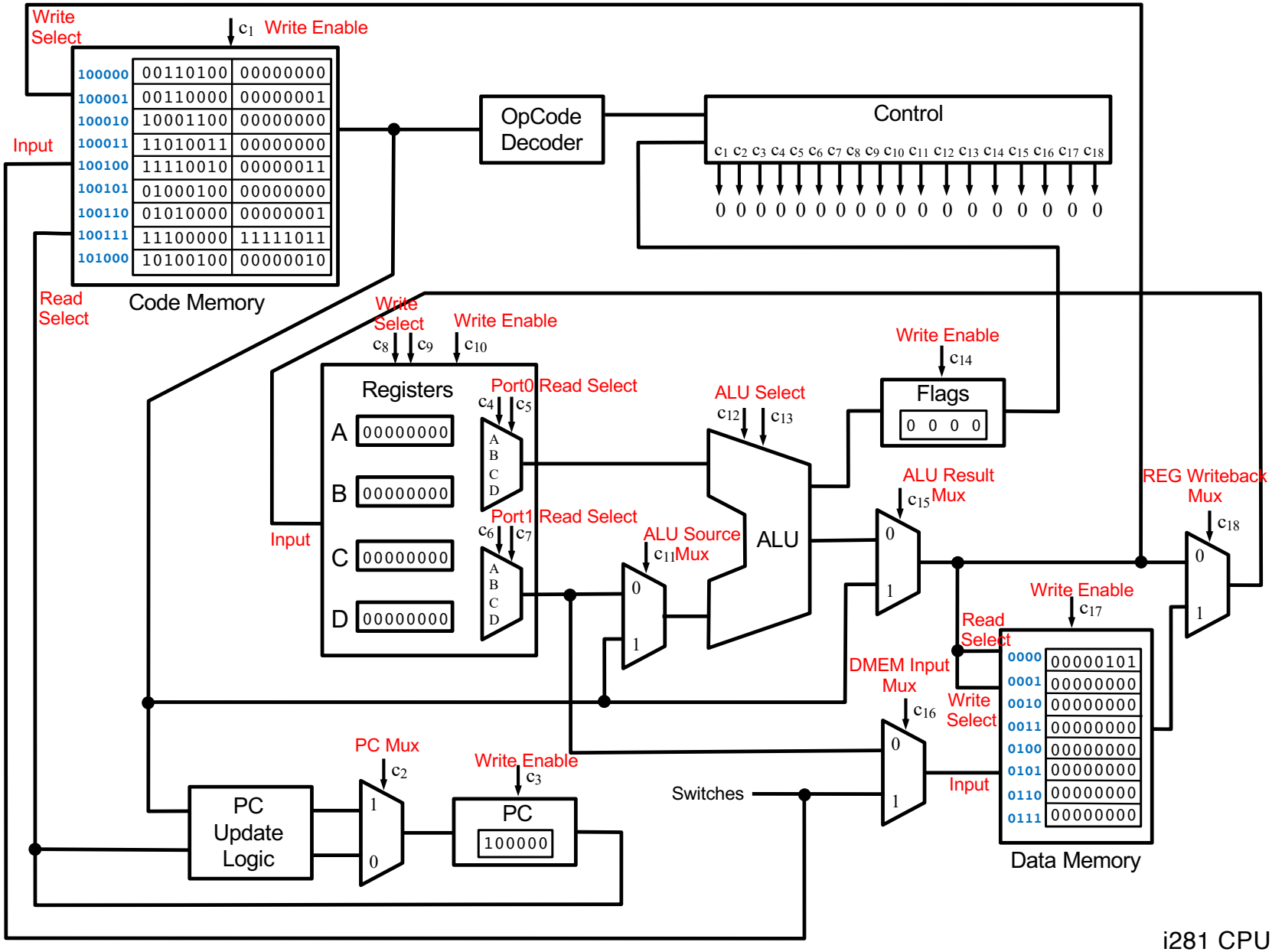
Register file with 2 read ports

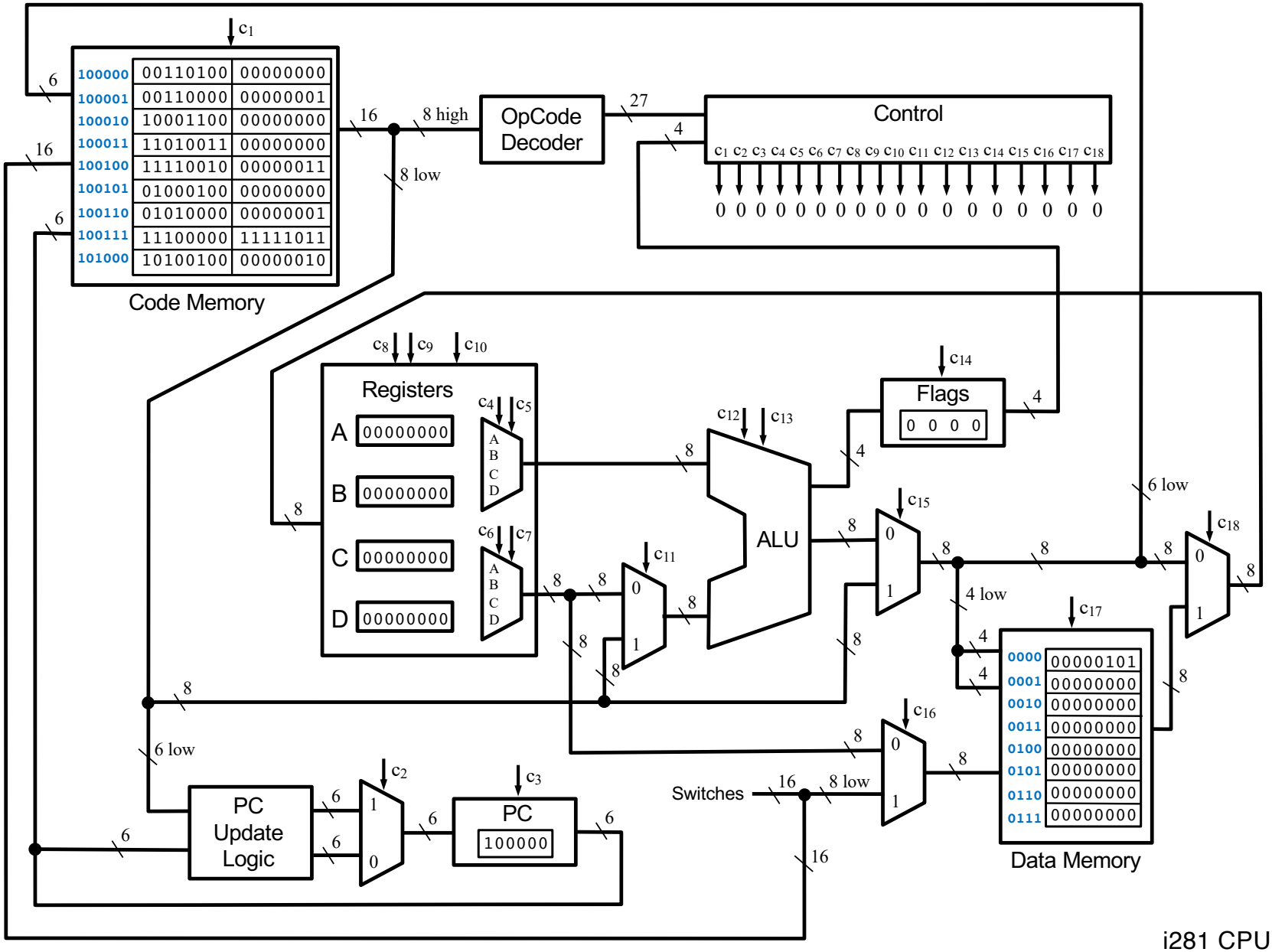


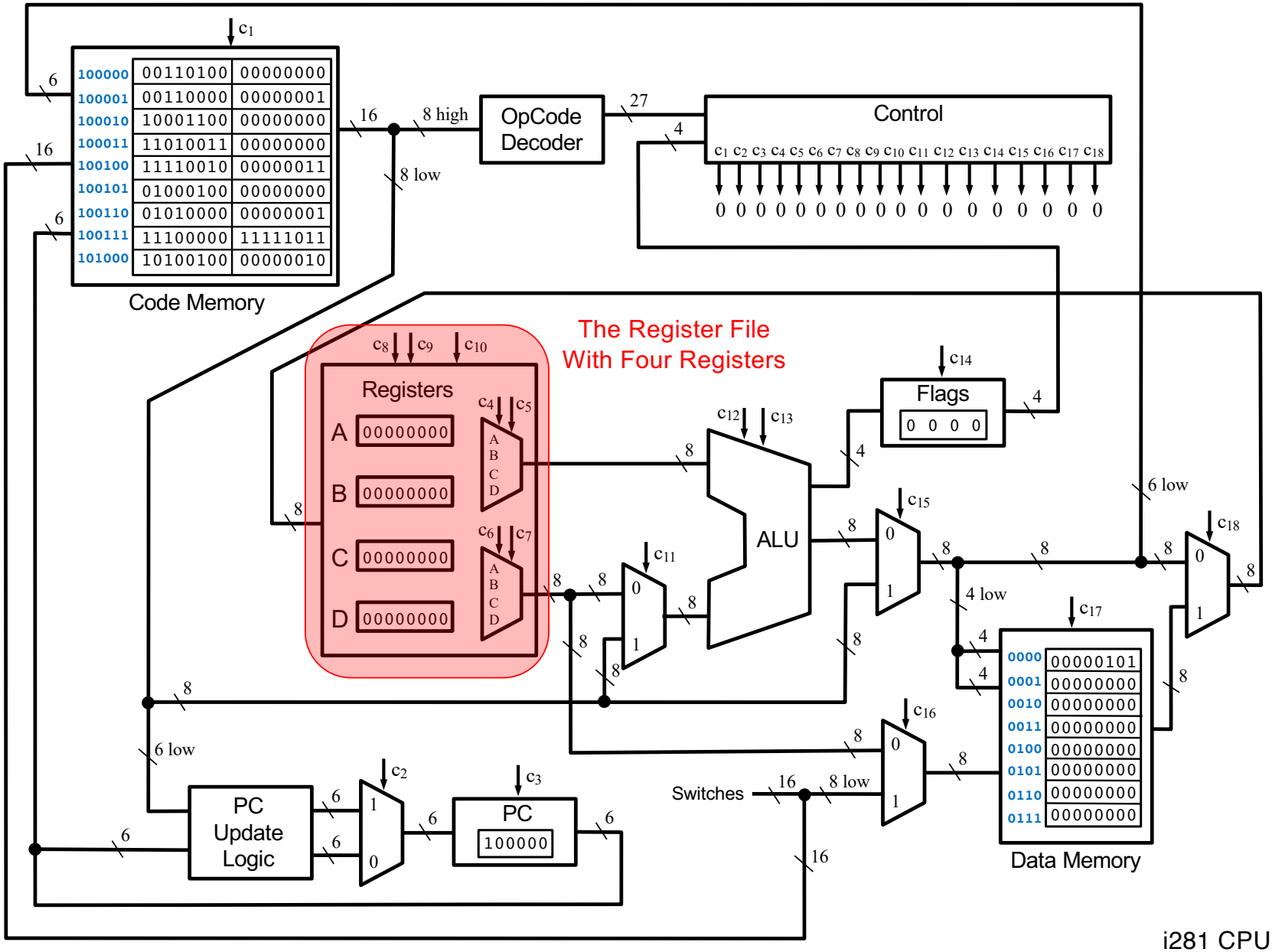
[https://jindongpu.wordpress.com/2012/03/07/register-file/]

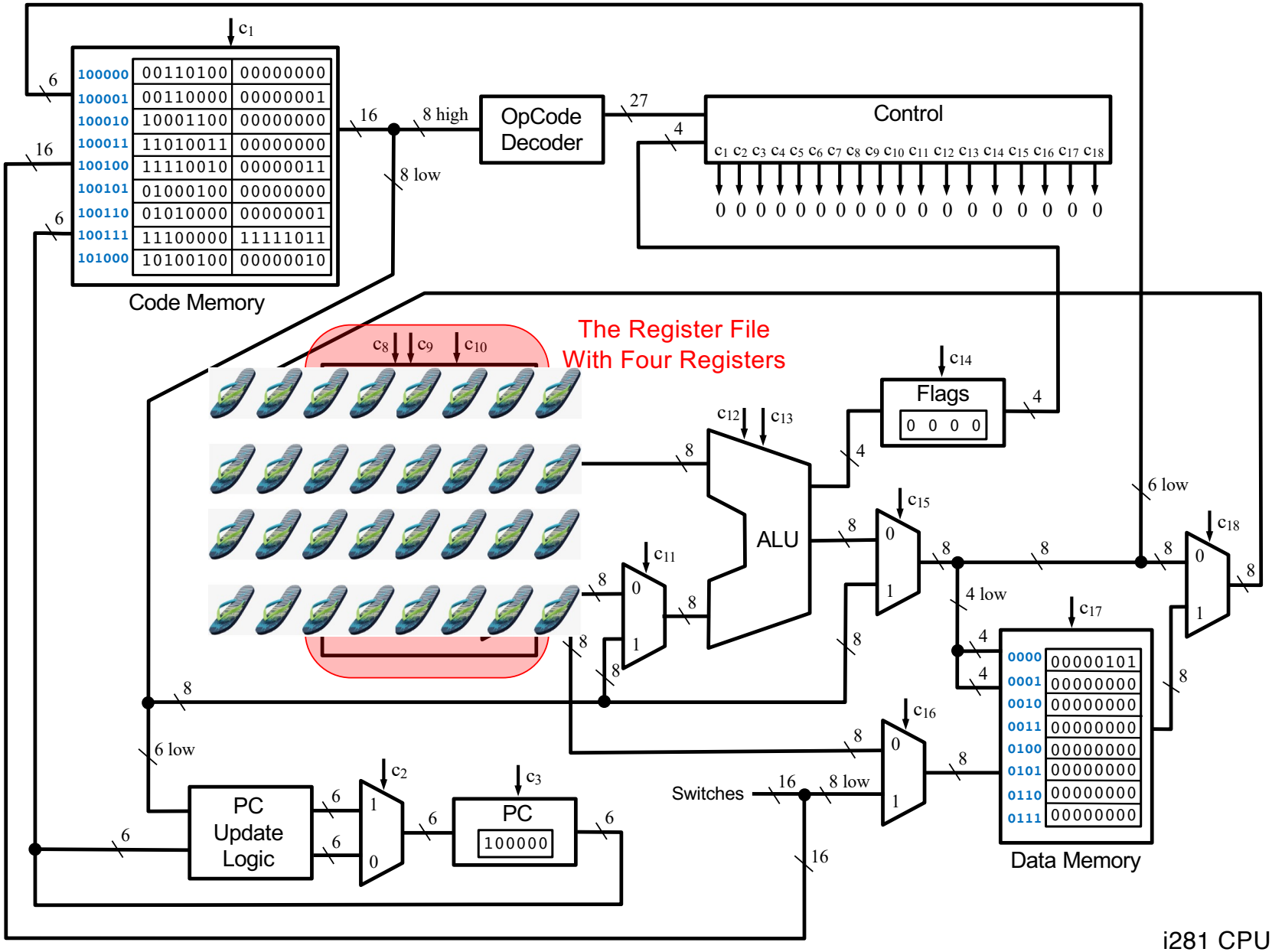
The i281 CPU has a register file with four 8-bit registers and two read ports

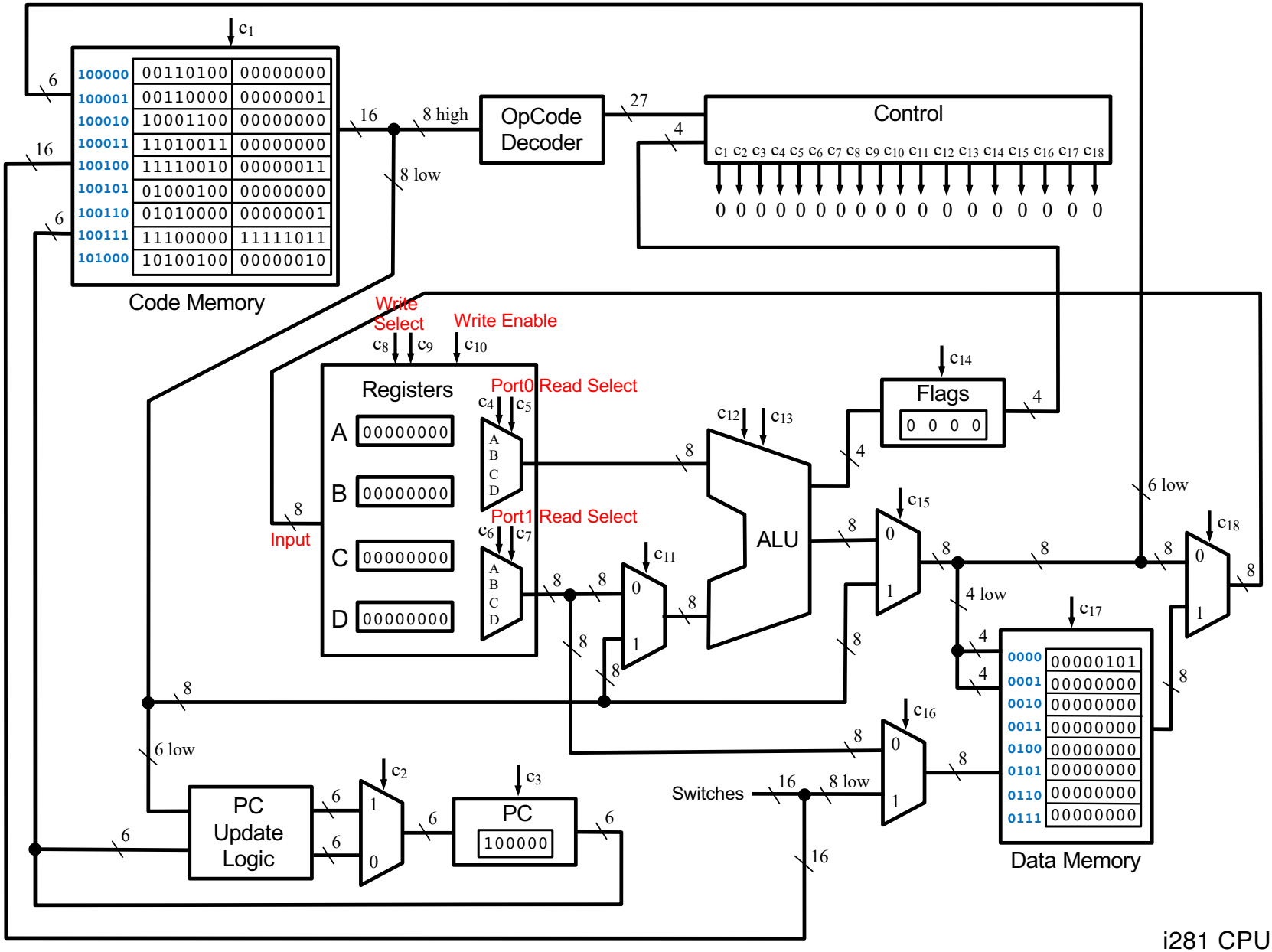

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
OpCode Decoder
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 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
c11
c12 c13
ALU
c14
Flags
0 0 0 0
c15
c18
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
c2
c3
PC Update Logic
PC
000000
i281 CPU

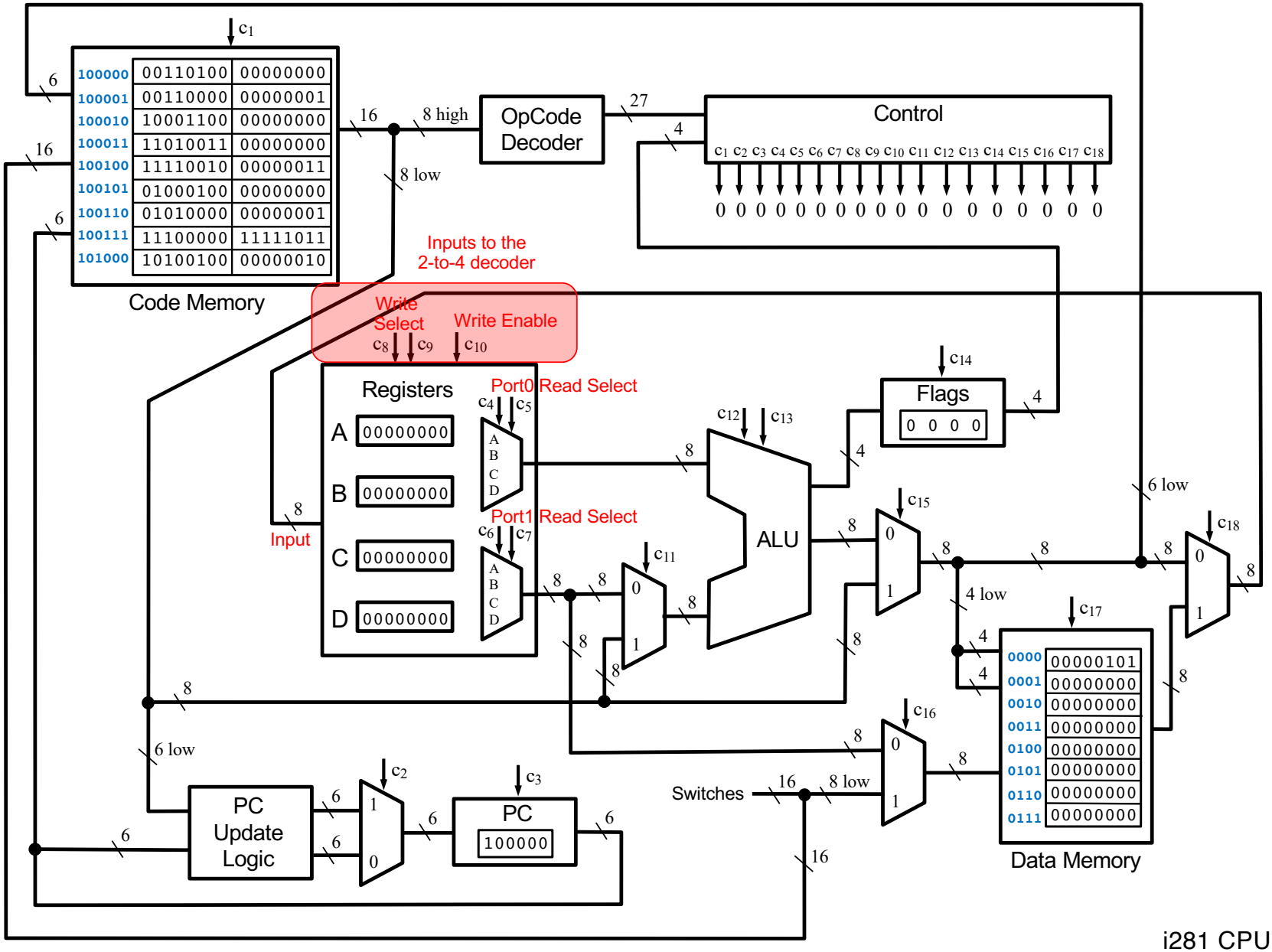

Write Select
c1 Write Enable
Input
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
Read Select
OpCode Decoder
Control
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
Write Select
c8 c9
Write Enable
c10
Registers
Port0 Read Select
c4 c5
A 00000000
B 00000000
C 00000000
D 00000000
Port1 Read Select
c6 c7
Input
ALU Select
c12 c13
ALU
ALU Source
c11 Mux
Write Enable
c14
Flags
0 0 0 0
ALU Result
c15 Mux
REG Writeback
Mux
c18
Write Enable
c17
Read Select
DMEM Input
Mux
c16
Write Select
Input
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
PC Mux
c2
Write Enable
c3
PC
100000
PC Update Logic
Switches
i281 CPU

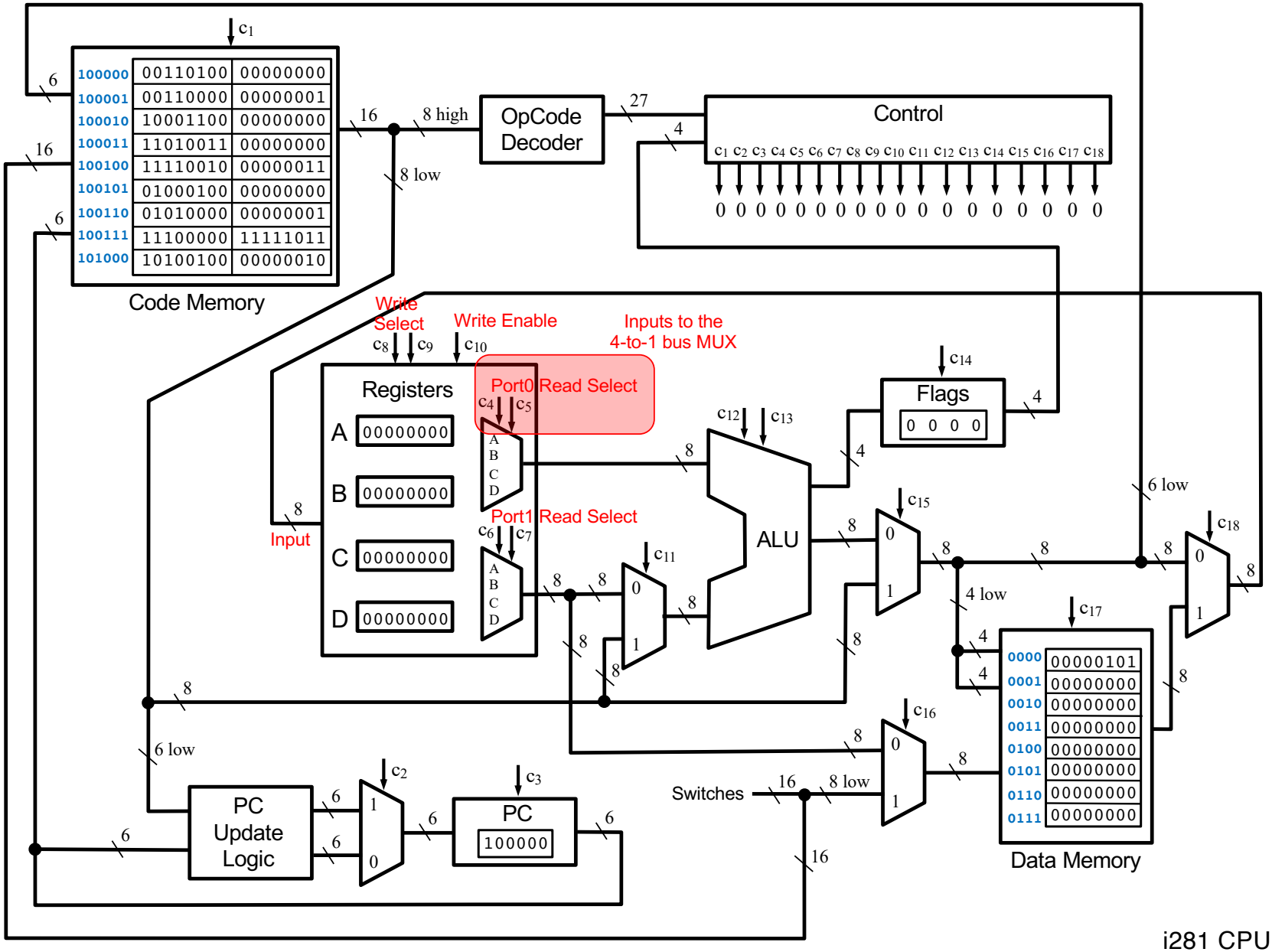

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
6
16
6
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 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
c15
c11
c16
c17
c18
6 low
4 low
8 low
Switches
16
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

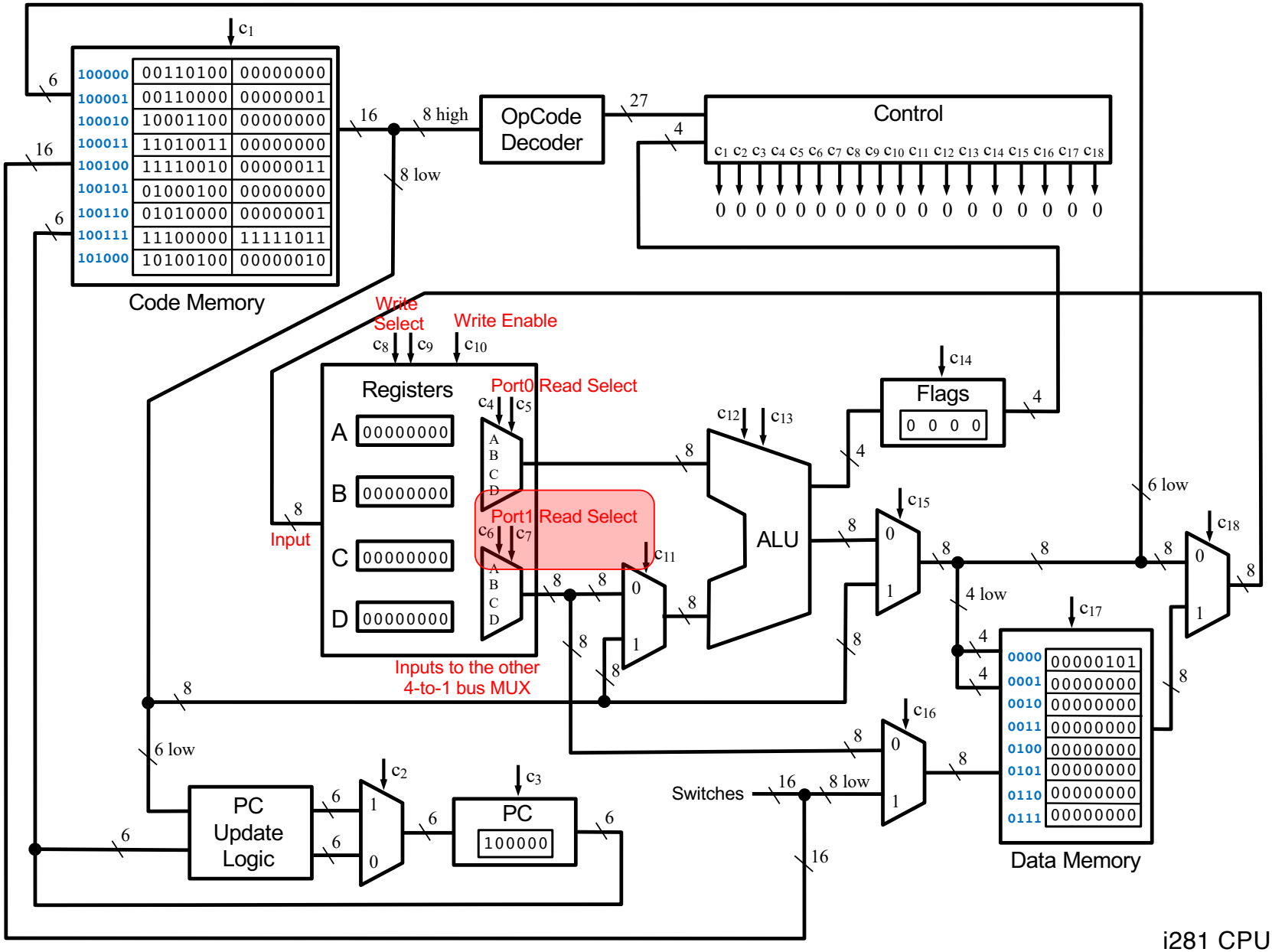

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 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
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
c12 c13
ALU
c14
Flags
0 0 0 0
c15
c11
c16
c17
c18
6 low
4 low
8 low
Switches
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

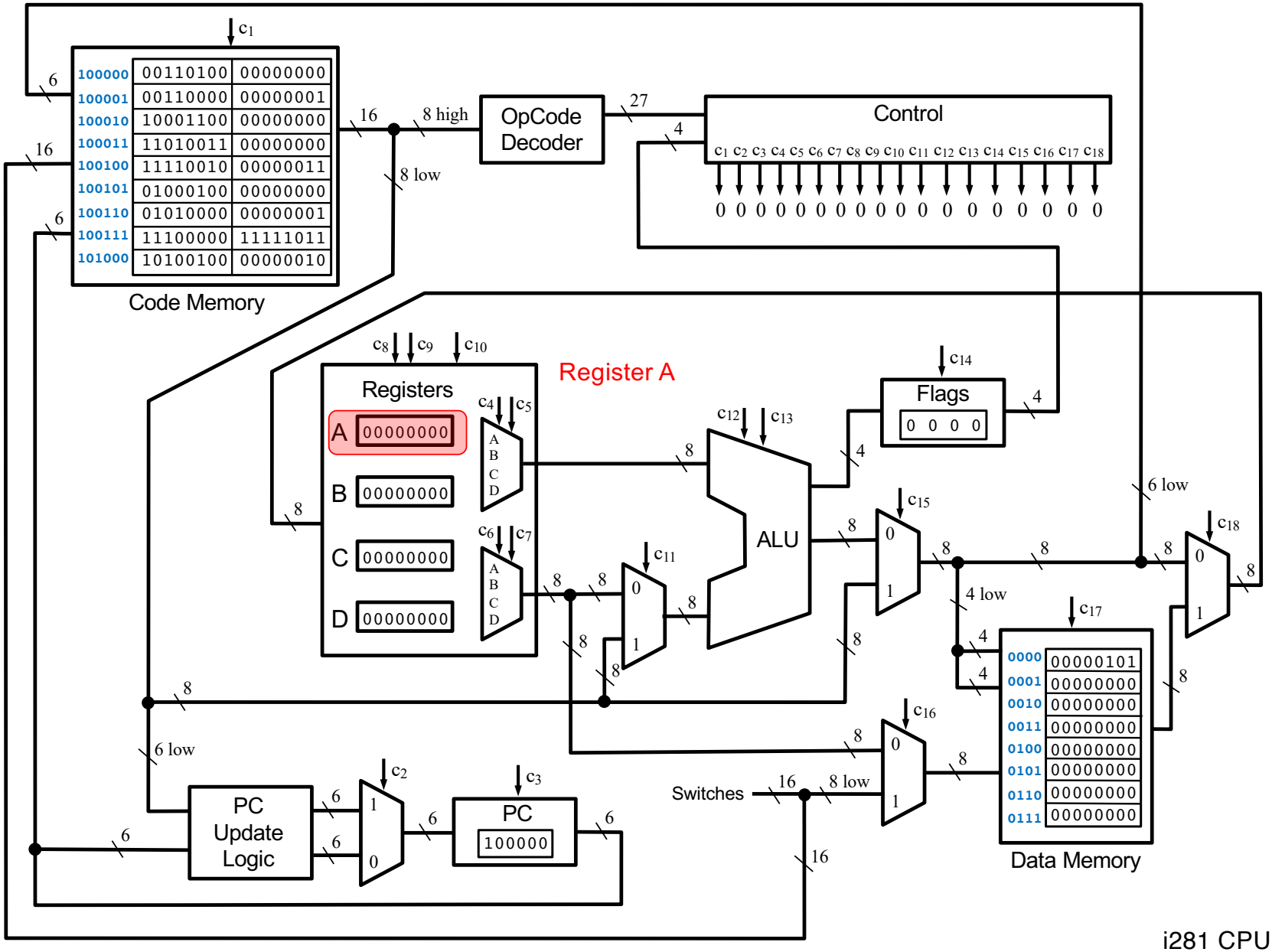

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
6
16
6
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 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
c12 c13
ALU
8
c14
Flags
0 0 0 0
4
c15
c11
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
100000
PC
Update
Logic
6 low
i281 CPU

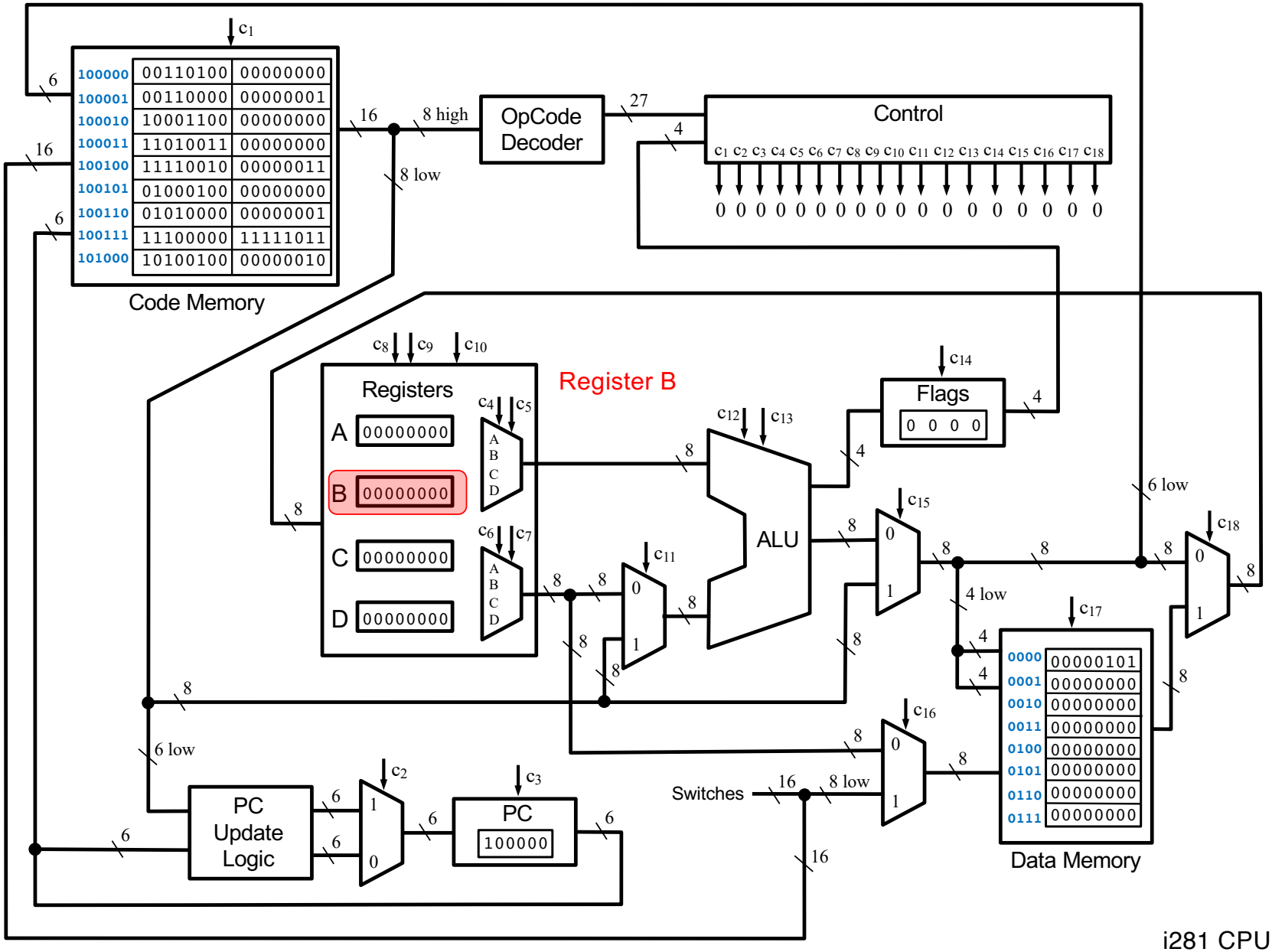

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
6
16
6
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 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
Write Select
c8 c9
Write Enable
c10
Registers
Port0 Read Select
c4 c5
A 00000000
B 00000000
C 00000000
D 00000000
A B C D
Port1 Read Select
c6 c7
8
Input
c12 c13
ALU
c14
Flags
0 0 0 0
4
c15
c11
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

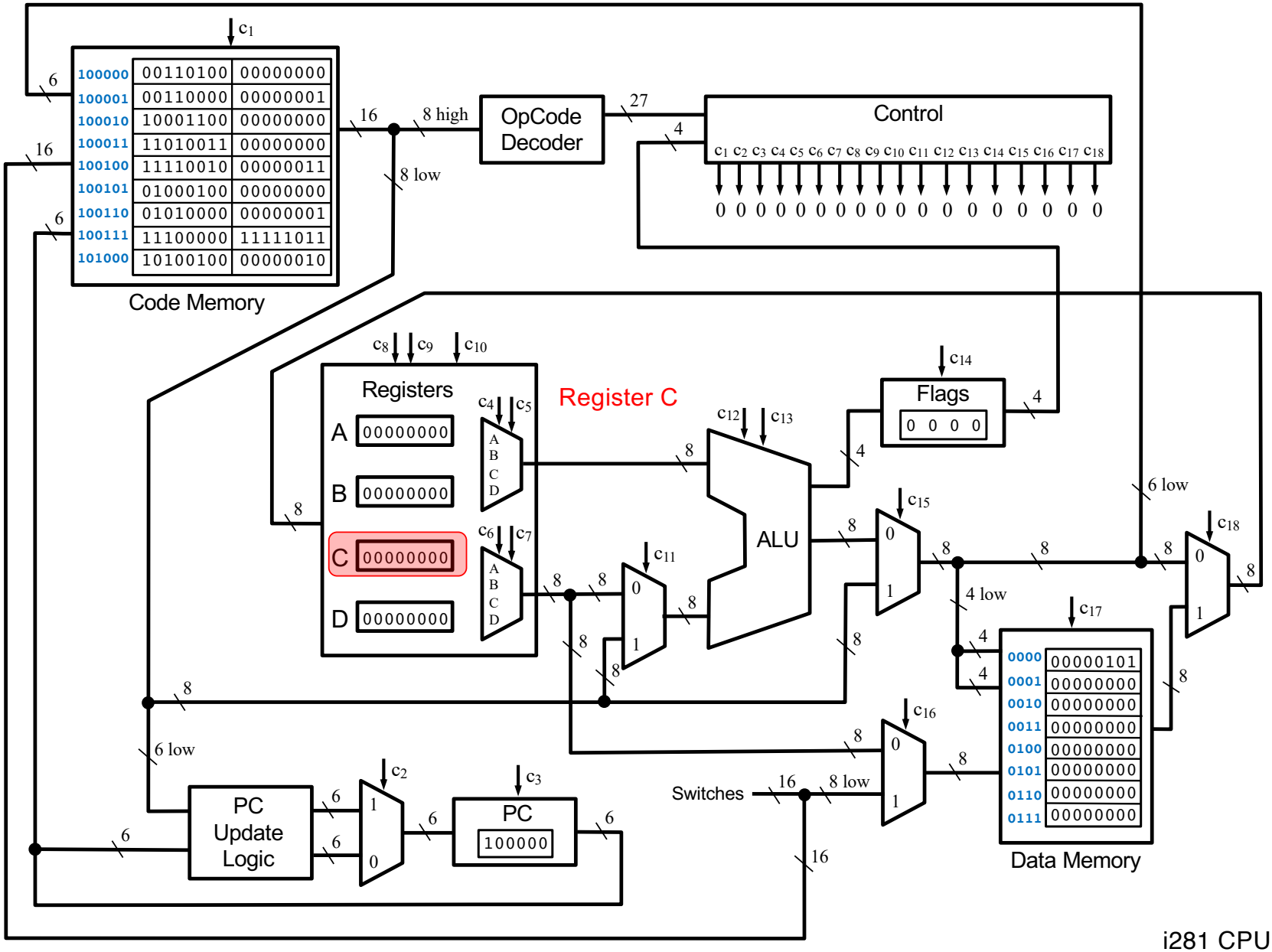

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 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
Inputs to the 2-to-4 decoder
Write Select
c8 c9
Write Enable
c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
Port0 Read Select
Port1 Read Select
c4 c5
c6 c7
Input
c12 c13
ALU
c14
Flags
0 0 0 0
c15
c11
c16
c17
c18
6 low
4 low
8 low
Switches
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

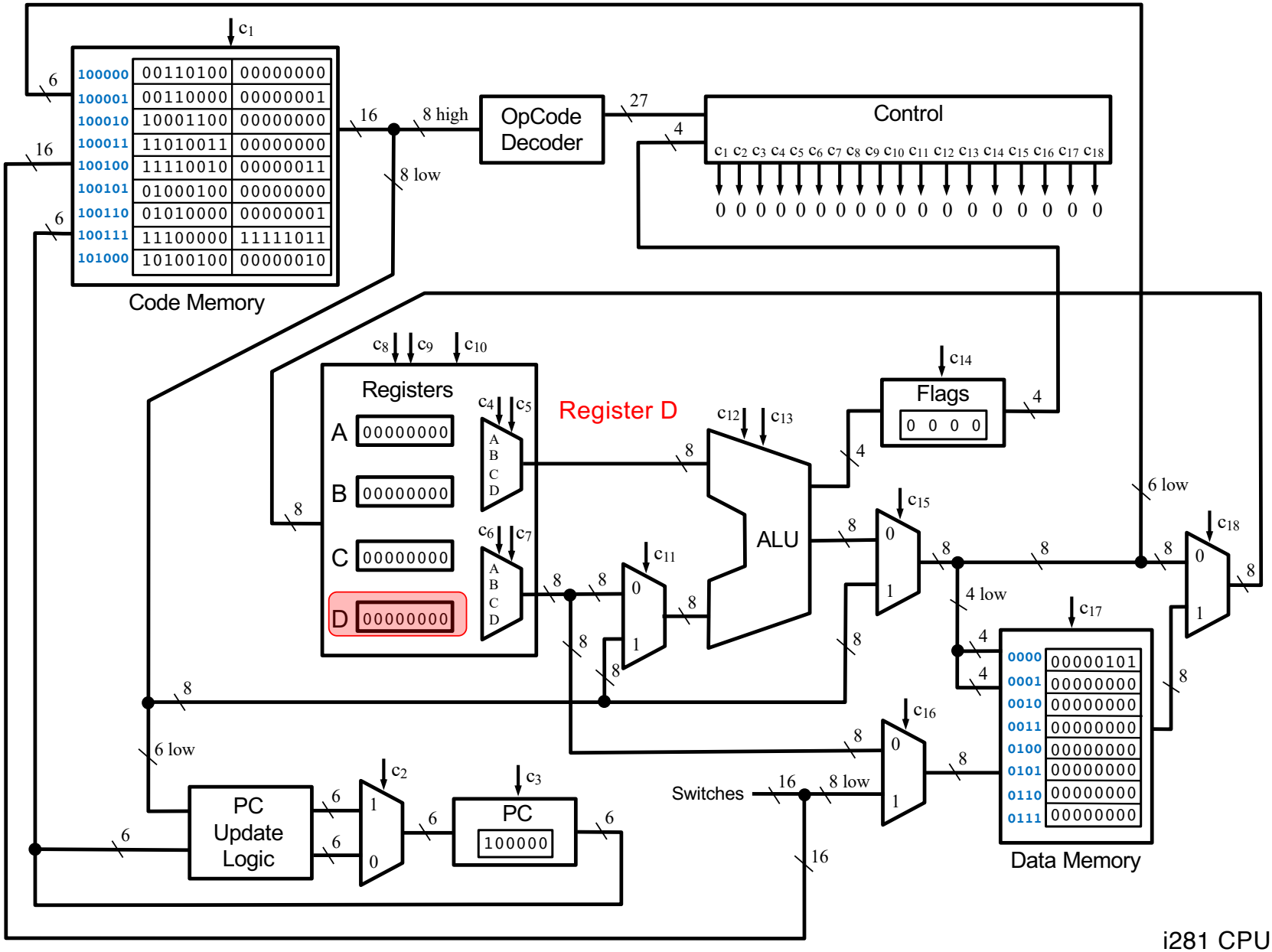

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
6
16
6
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 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
Write Select
c8 c9
Write Enable
c10
Inputs to the 4-to-1 bus MUX
Registers
Port0 Read Select
c4 c5
A 00000000
B 00000000
C 00000000
D 00000000
A B C D
Port1 Read Select
c6 c7
8
Input
c12 c13
ALU
c14
Flags
0 0 0 0
4
c15
c11
c16
c18
6 low
4 low
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
Switches
16
8 low
c2
c3
PC Update Logic
PC
100000
6
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
6
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
Write Select
c8 c9
Write Enable
c10
Registers
Port0 Read Select
c4 c5
A 00000000
B 00000000
C 00000000
D 00000000
A B C D
Port1 Read Select
c6 c7
c11
8
Input
Inputs to the other 4-to-1 bus MUX
c12 c13
ALU
c14
Flags
0 0 0 0
4
c15
c16
c17
c18
6 low
4 low
8 low
Switches
16
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

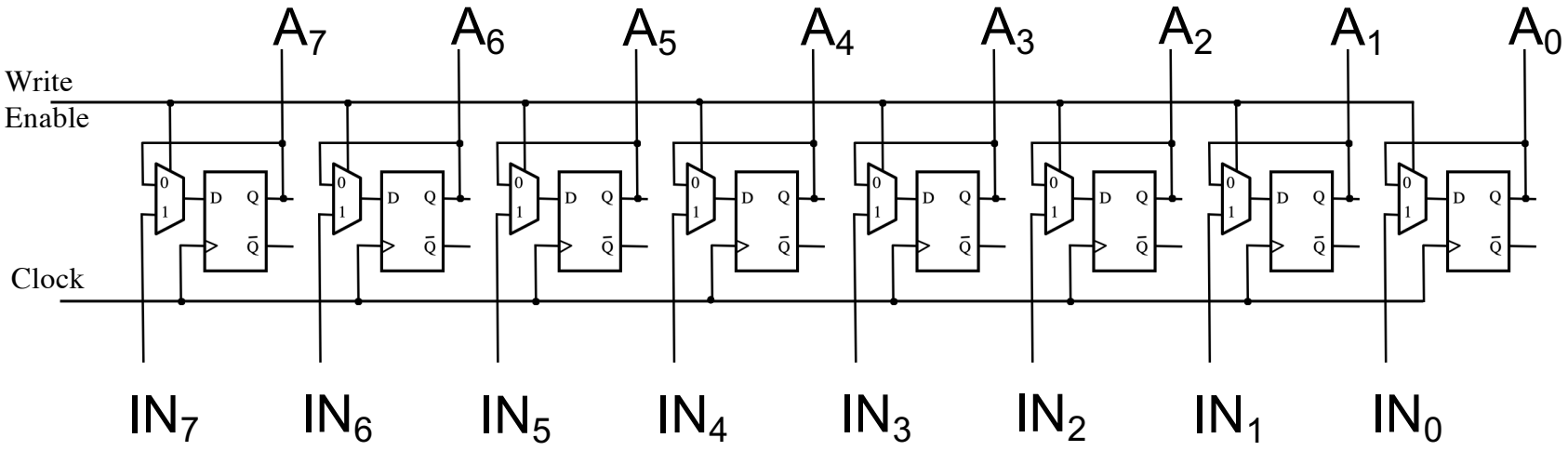

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
6
16
6
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 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
Register A
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
c15
c11
c16
c17
c18
6 low
4 low
8 low
Switches
16
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
PC Update Logic
c2
c3
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
6
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
Register B
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
c15
c11
c16
c17
c18
6 low
4 low
8 low
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
Switches
16
PC Update Logic
c2
c3
PC
100000
6
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
6
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
4
c15
c11
c16
c17
c18
6 low
4 low
8 high
Switches
16
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

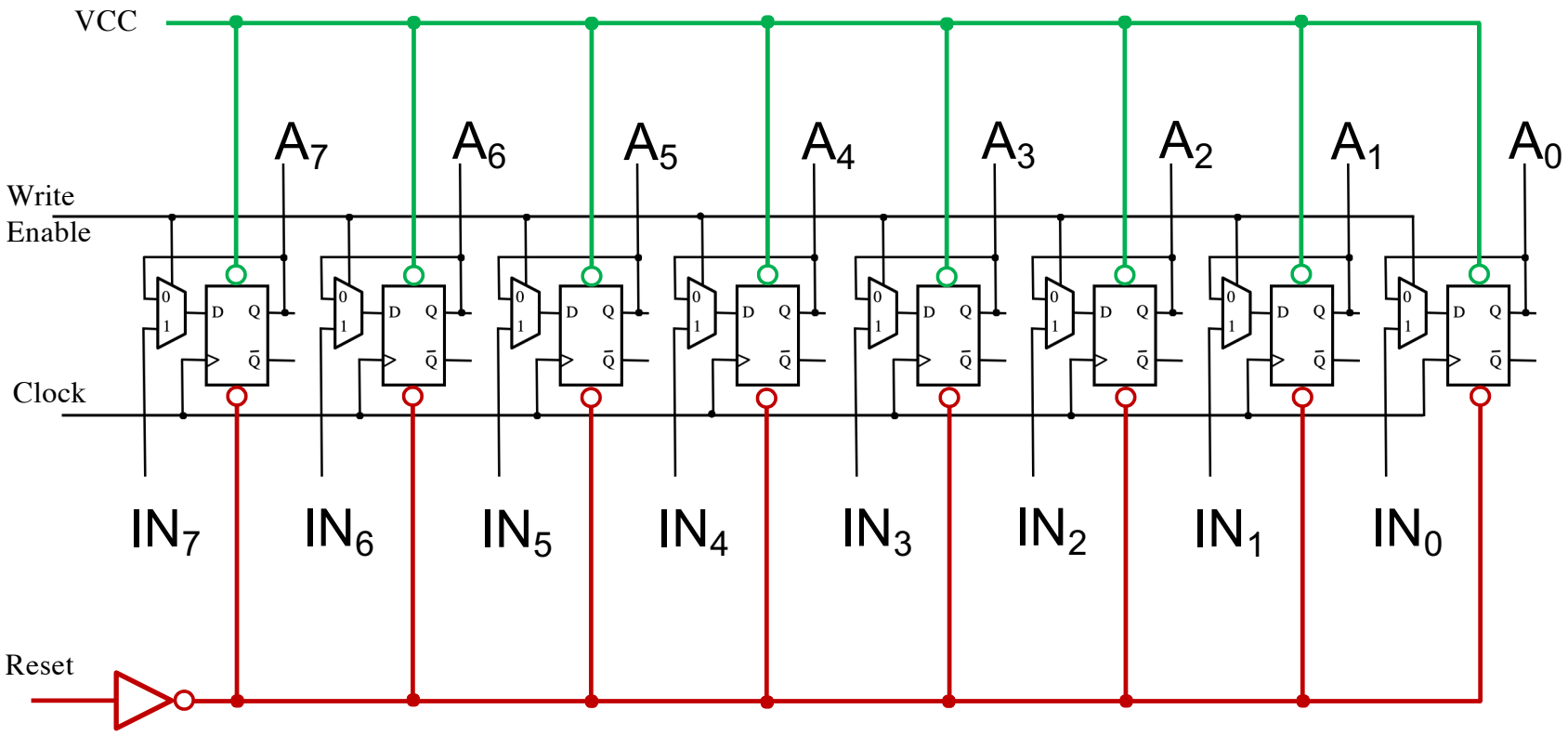

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
6
16
6
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 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
c15
c11
c16
c17
c18
6 low
4 low
8 high
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
6 low
i281 CPU

Register A

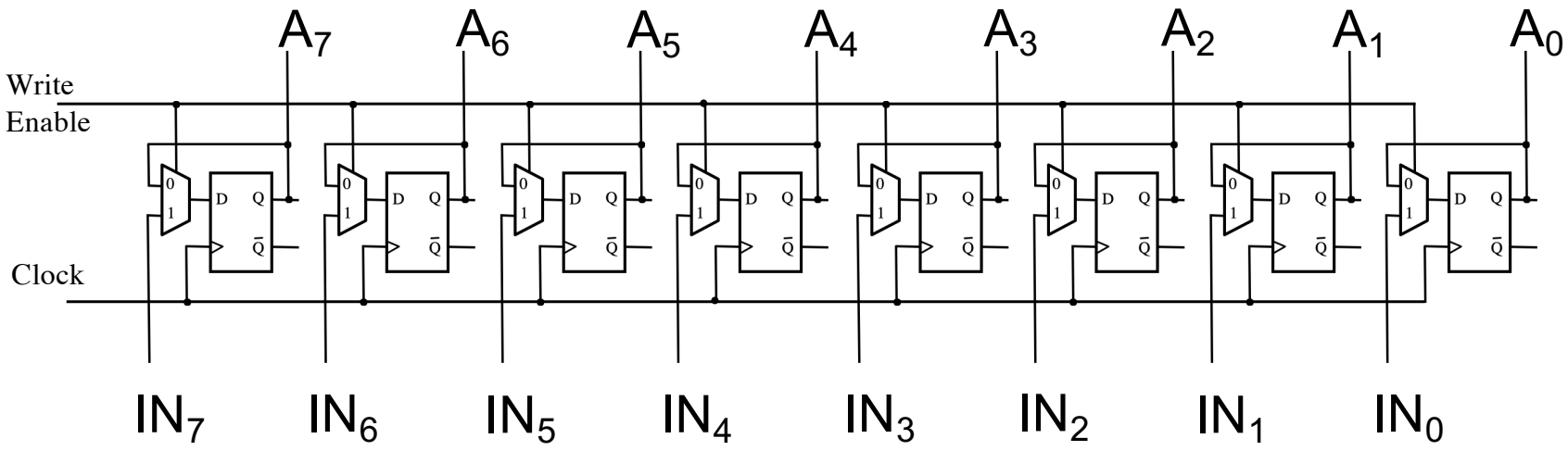


8-Bit Parallel-Access Register

Register A

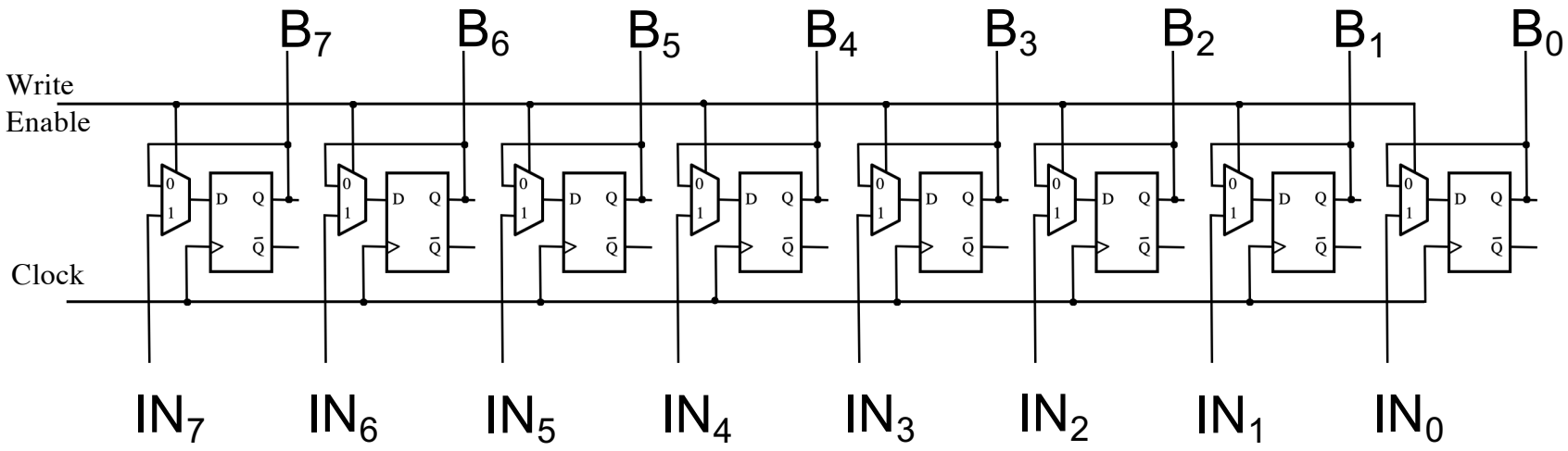

VCC
A7
A6
A5
A4
A3
A2
A1
A0
Write
Enable
Clock
IN7
IN6
IN5
IN4
IN3
IN2
IN1
IN0
Reset

Register A



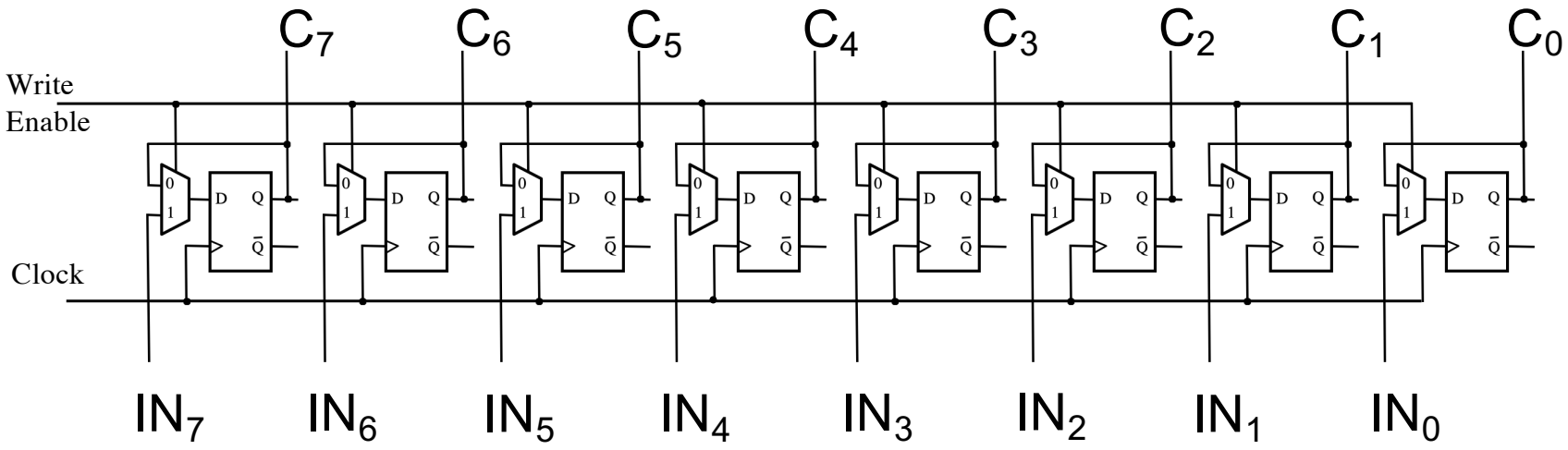
8-Bit Parallel-Access Register

Register B



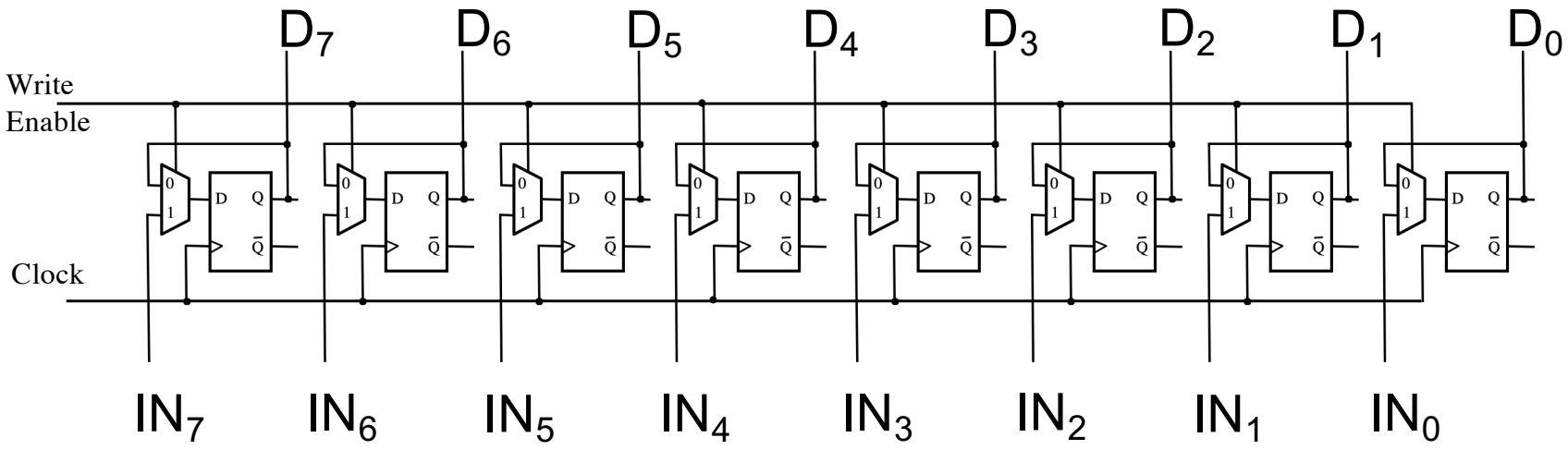
8-Bit Parallel-Access Register

Register C

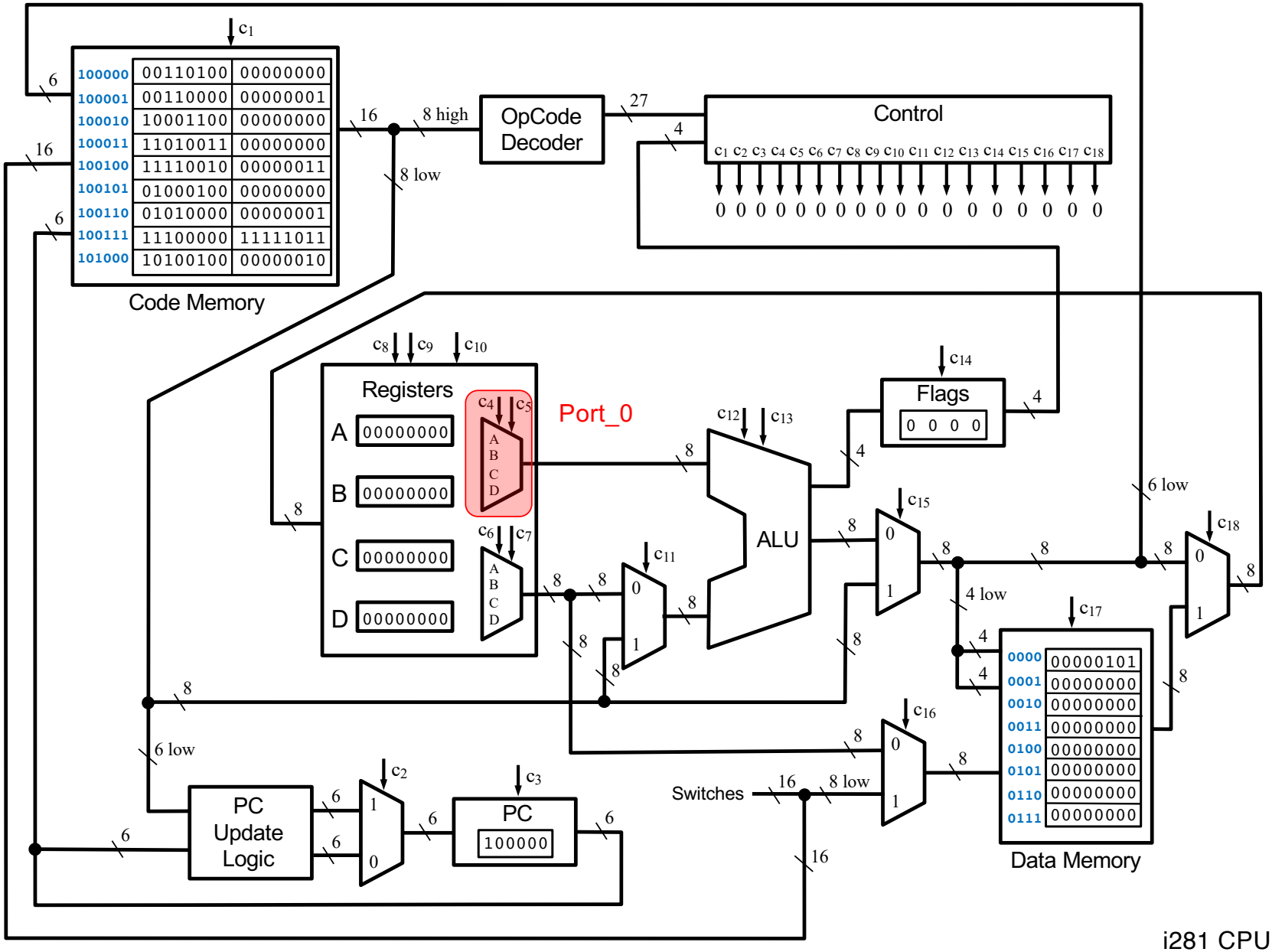


8-Bit Parallel-Access Register

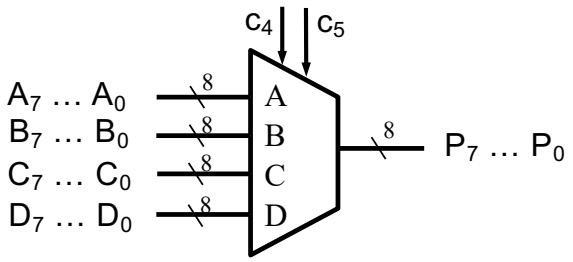
Register D

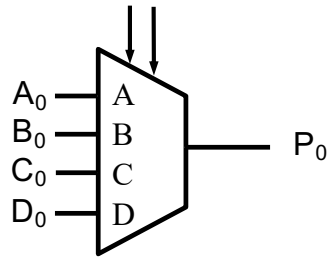


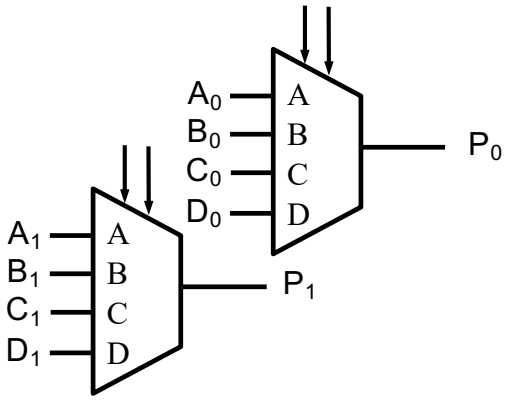
8-Bit Parallel-Access Register

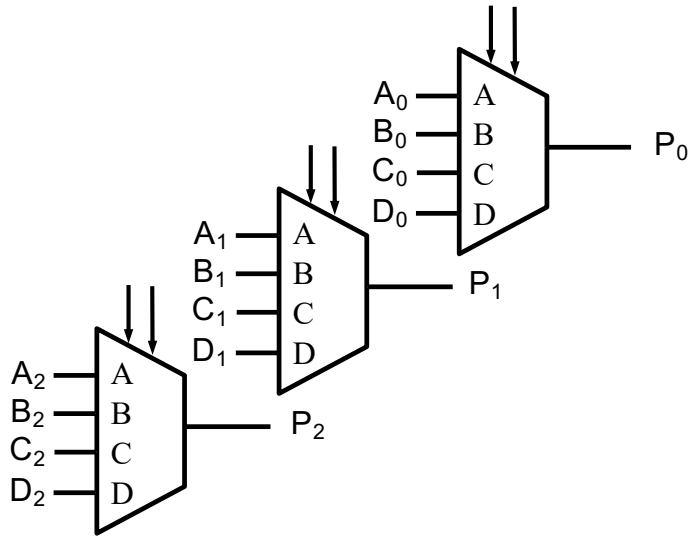

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
6
16
6
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4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

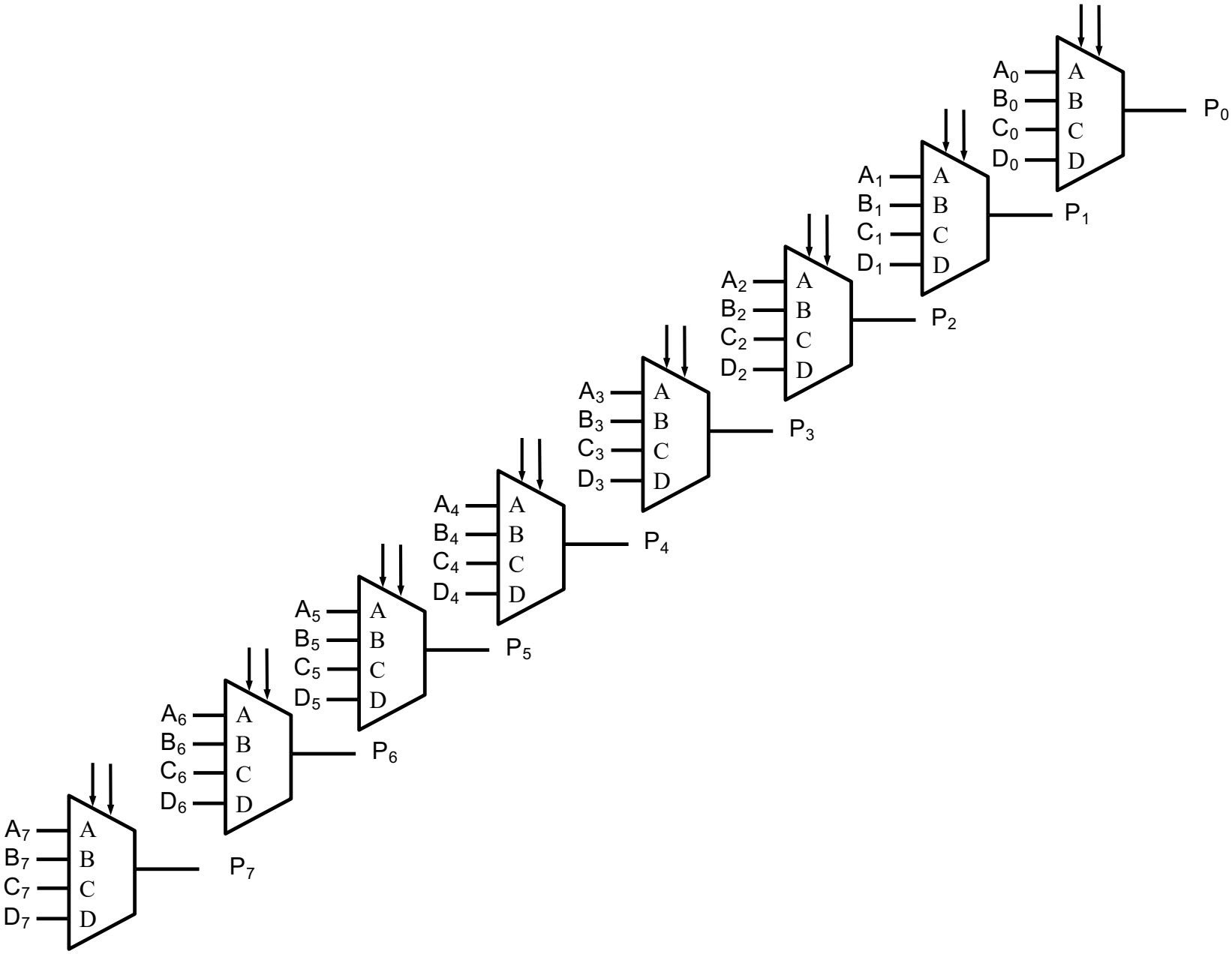
4-to-1 Bus Multiplexer (with 8-bit lines)

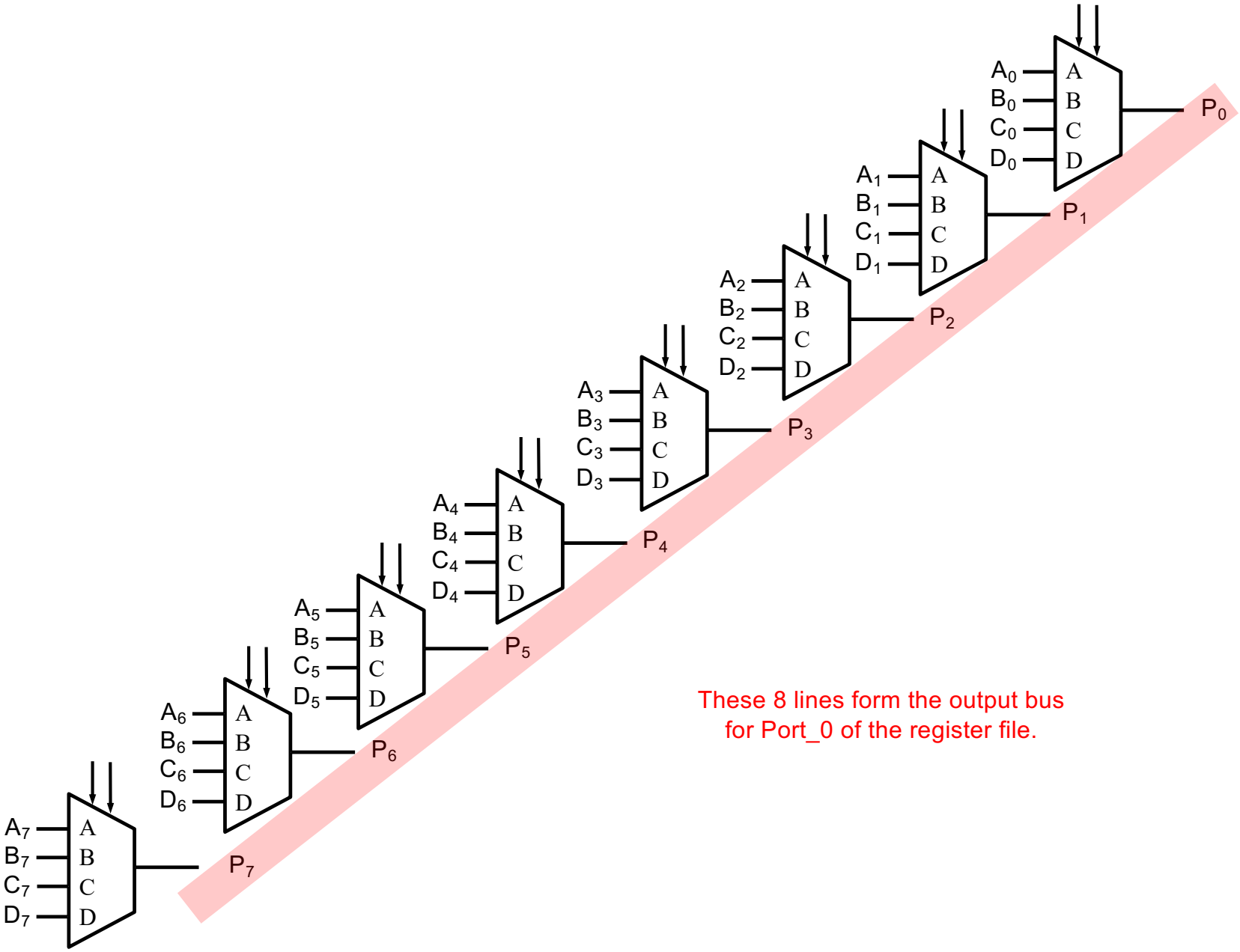


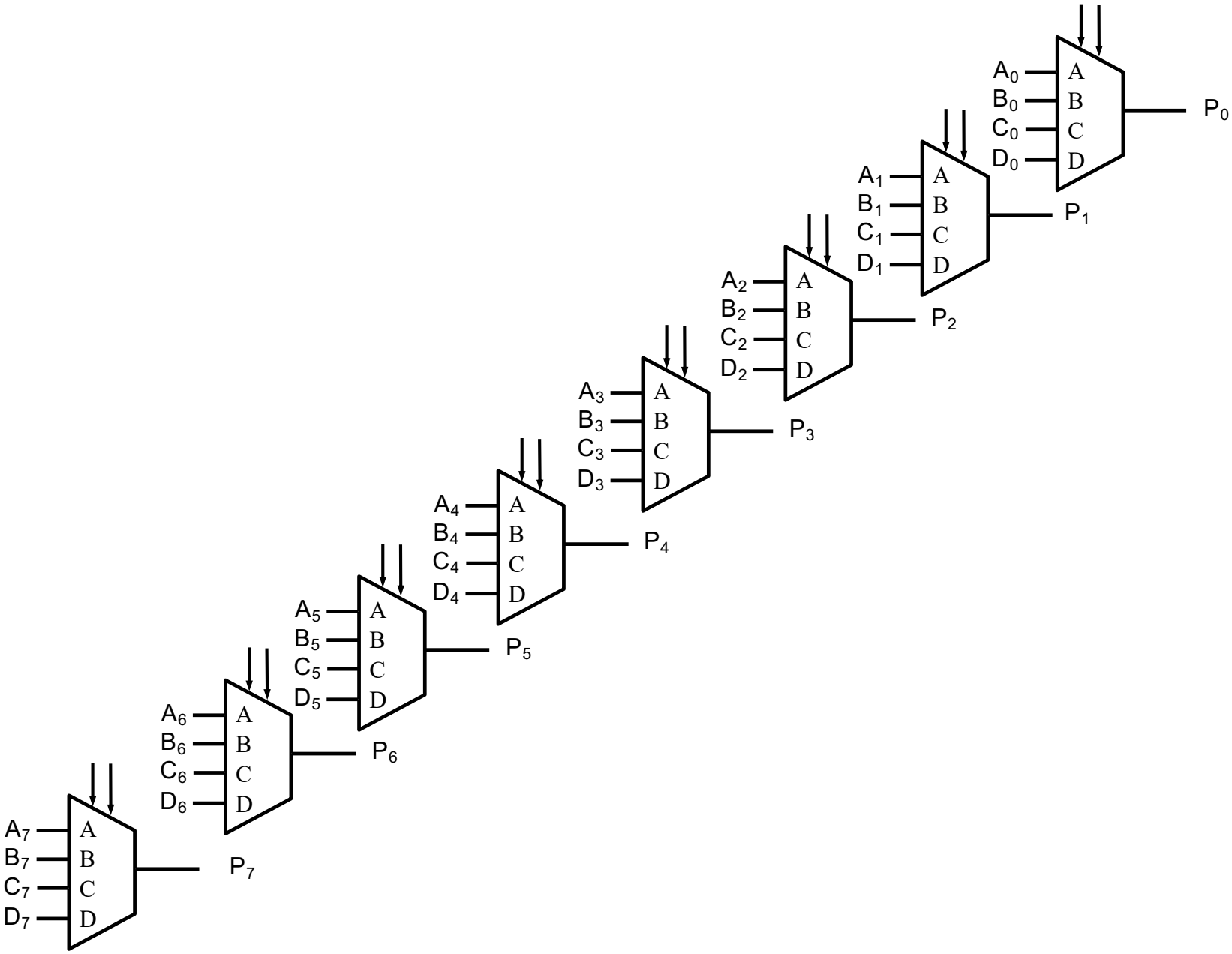

A0
B0
C0
D0
A
B
C
D
P0

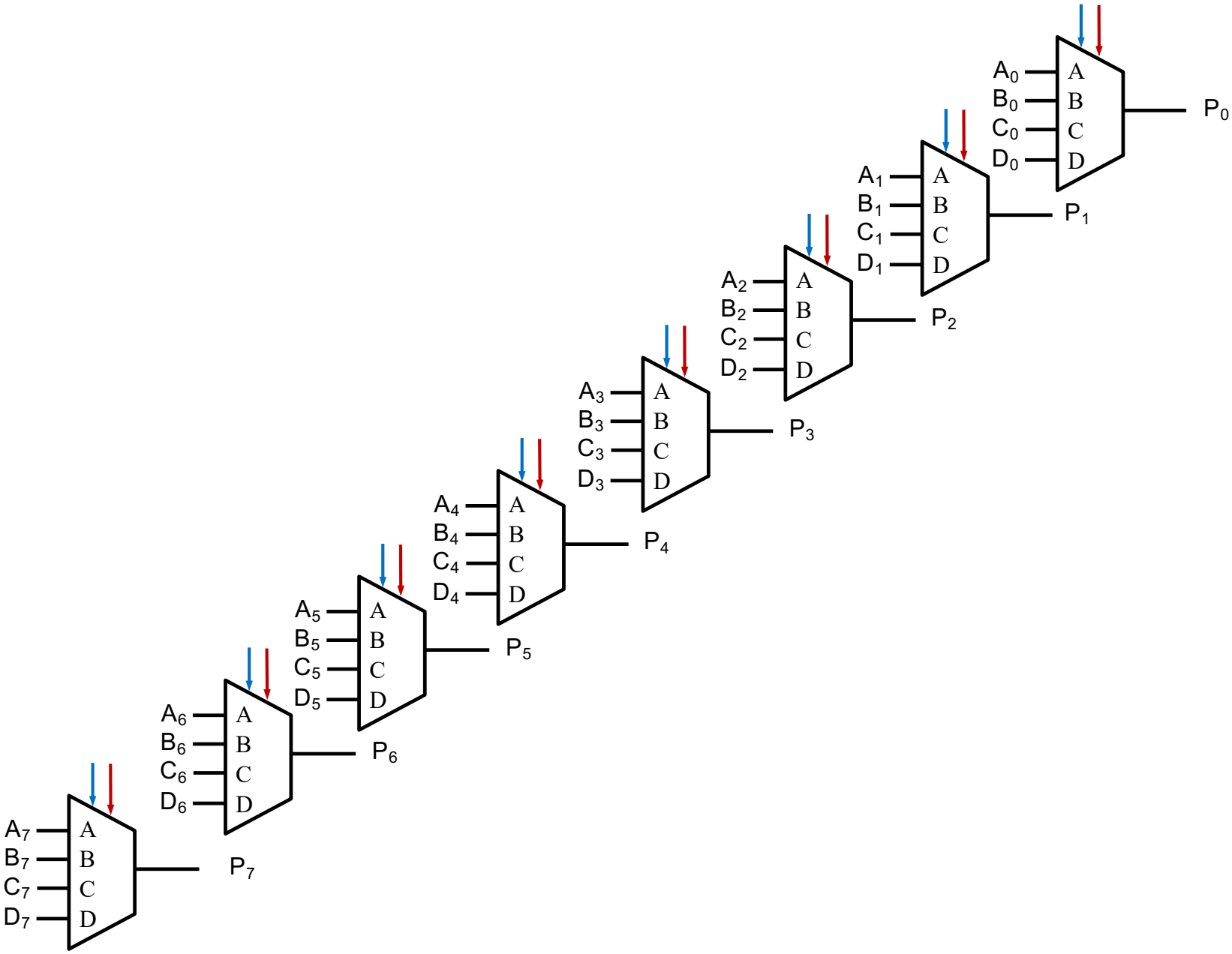

A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
A
B
C
D
P1

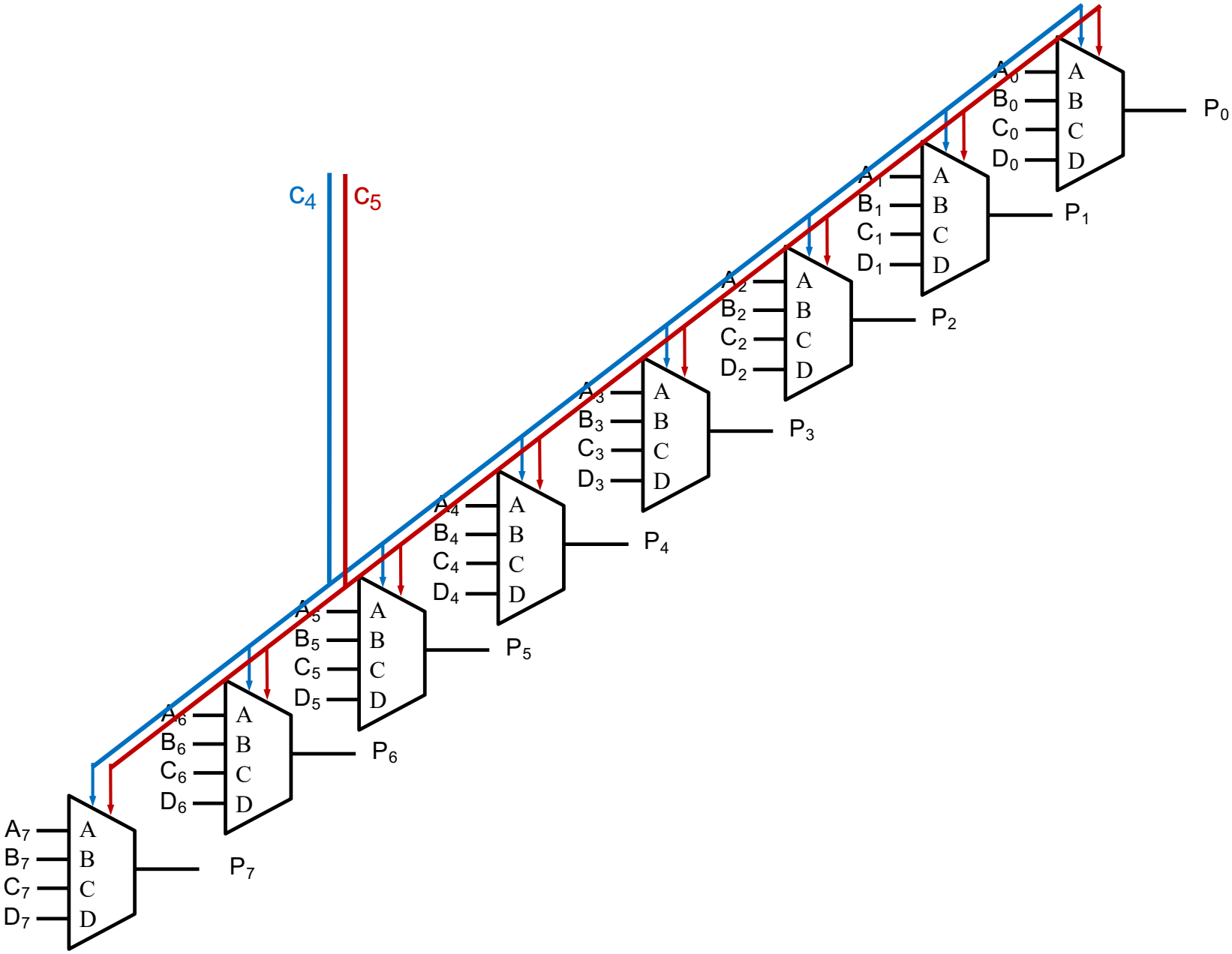

A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
A
B
C
D
P1
A2
B2
C2
D2
A
B
C
D
P2

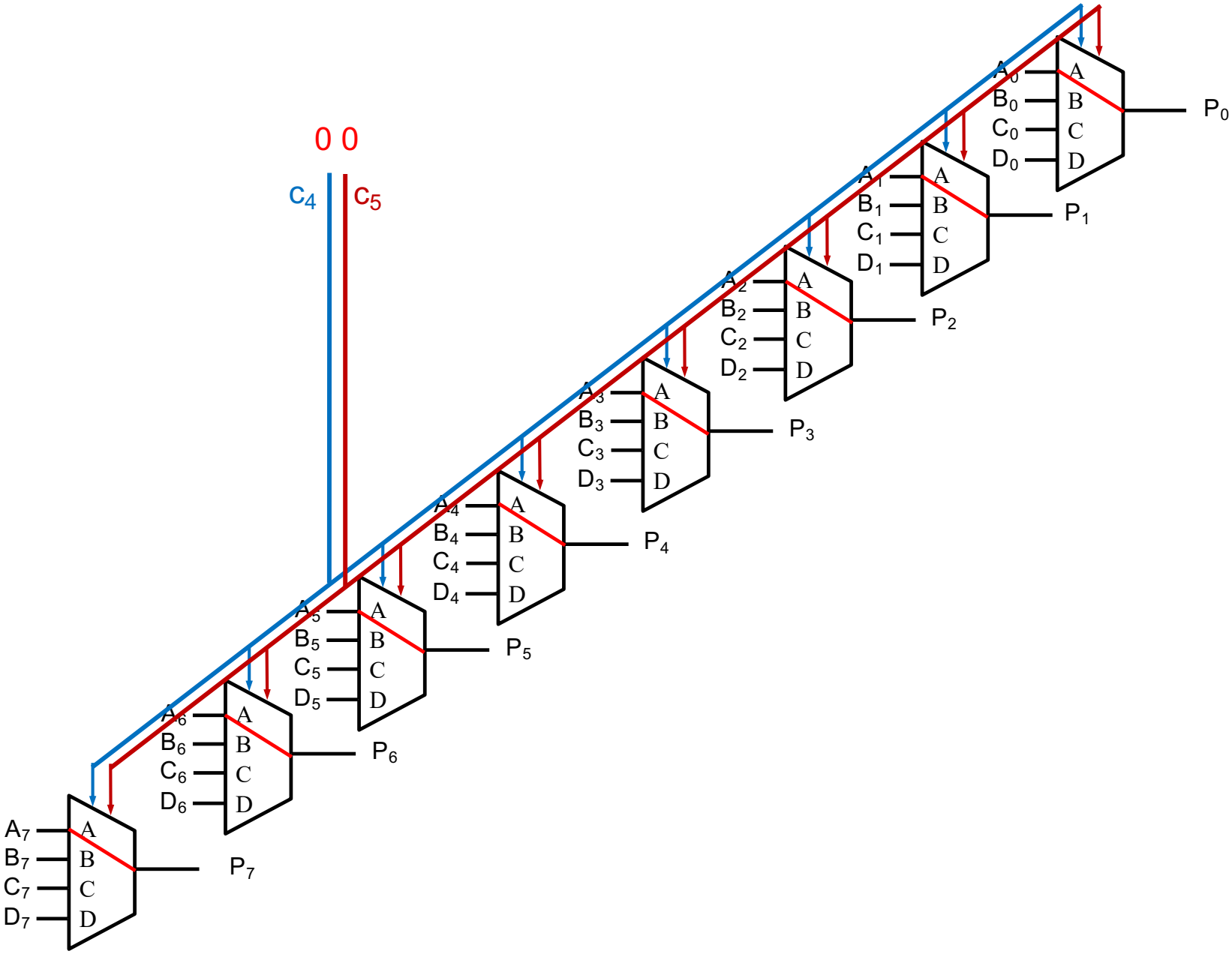

A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
P1
A2
B2
C2
D2
P2
A3
B3
C3
D3
P3
A4
B4
C4
D4
P4
A5
B5
C5
D5
P5
A6
B6
C6
D6
P6
A7
B7
C7
D7
P7

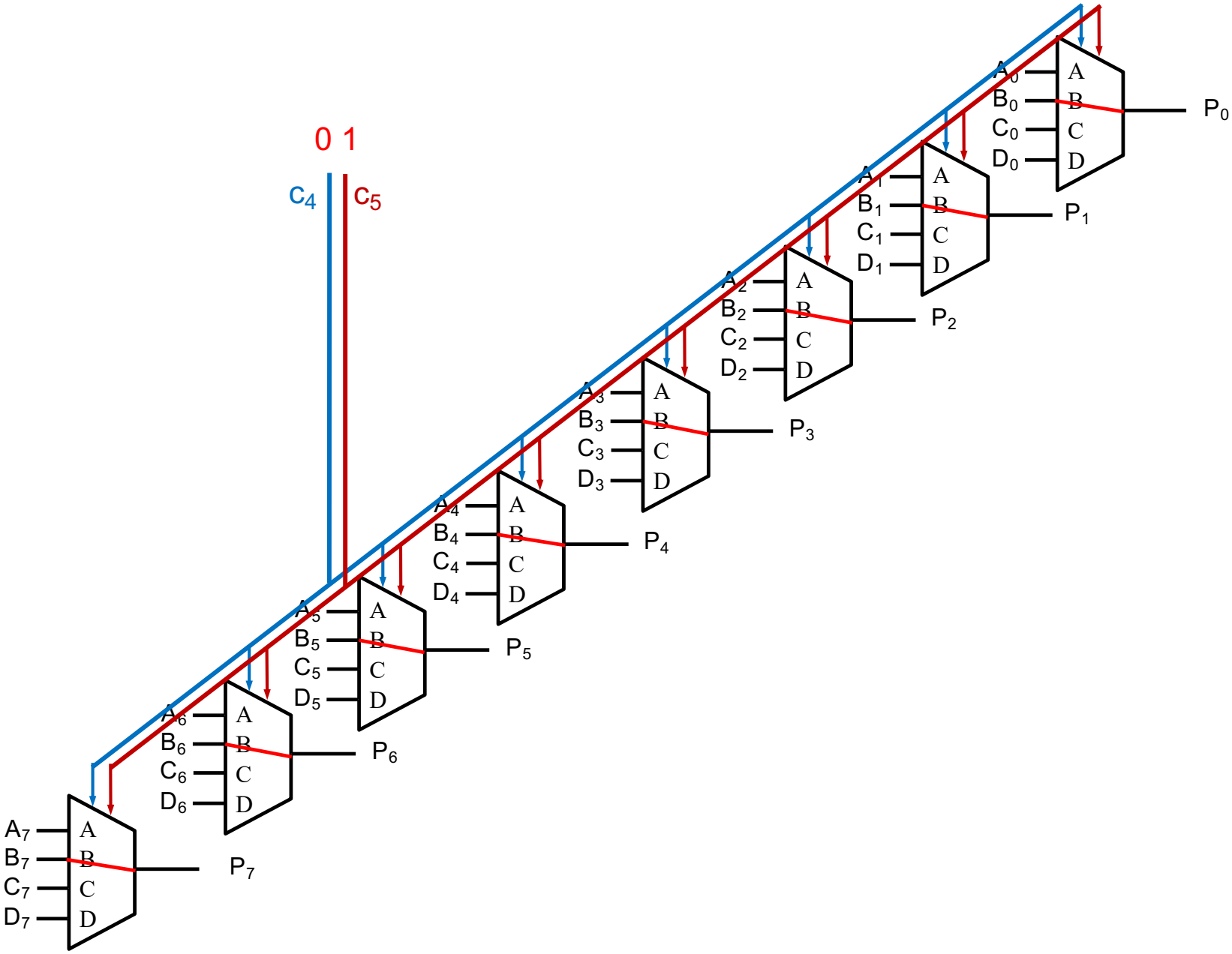

A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
P1
A2
B2
C2
D2
P2
A3
B3
C3
D3
P3
A4
B4
C4
D4
P4
A5
B5
C5
D5
P5
A6
B6
C6
D6
P6
A7
B7
C7
D7
P7
These 8 lines form the output bus
for Port_0 of the register file.

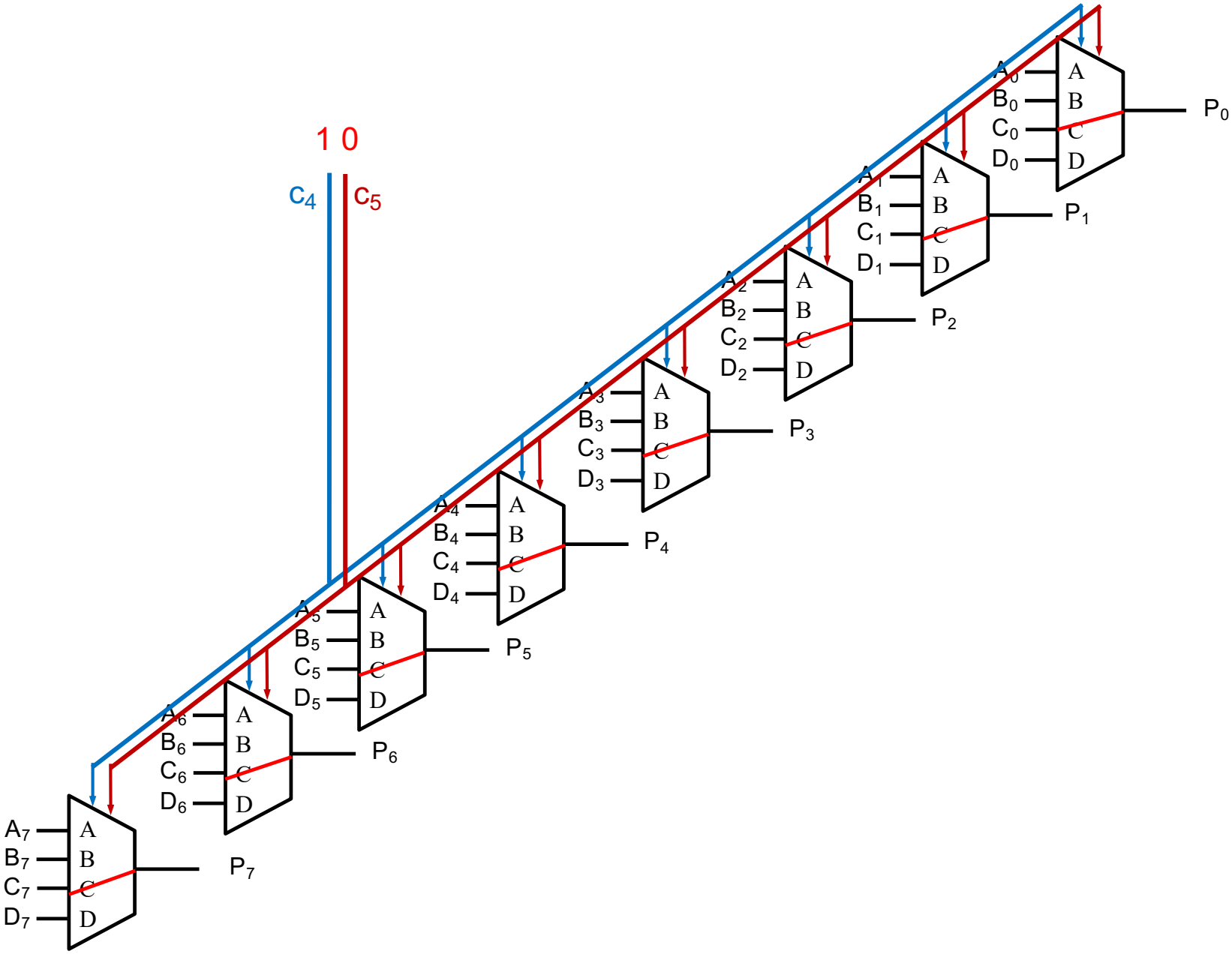

A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
P1
A2
B2
C2
D2
P2
A3
B3
C3
D3
P3
A4
B4
C4
D4
P4
A5
B5
C5
D5
P5
A6
B6
C6
D6
P6
A7
B7
C7
D7
P7

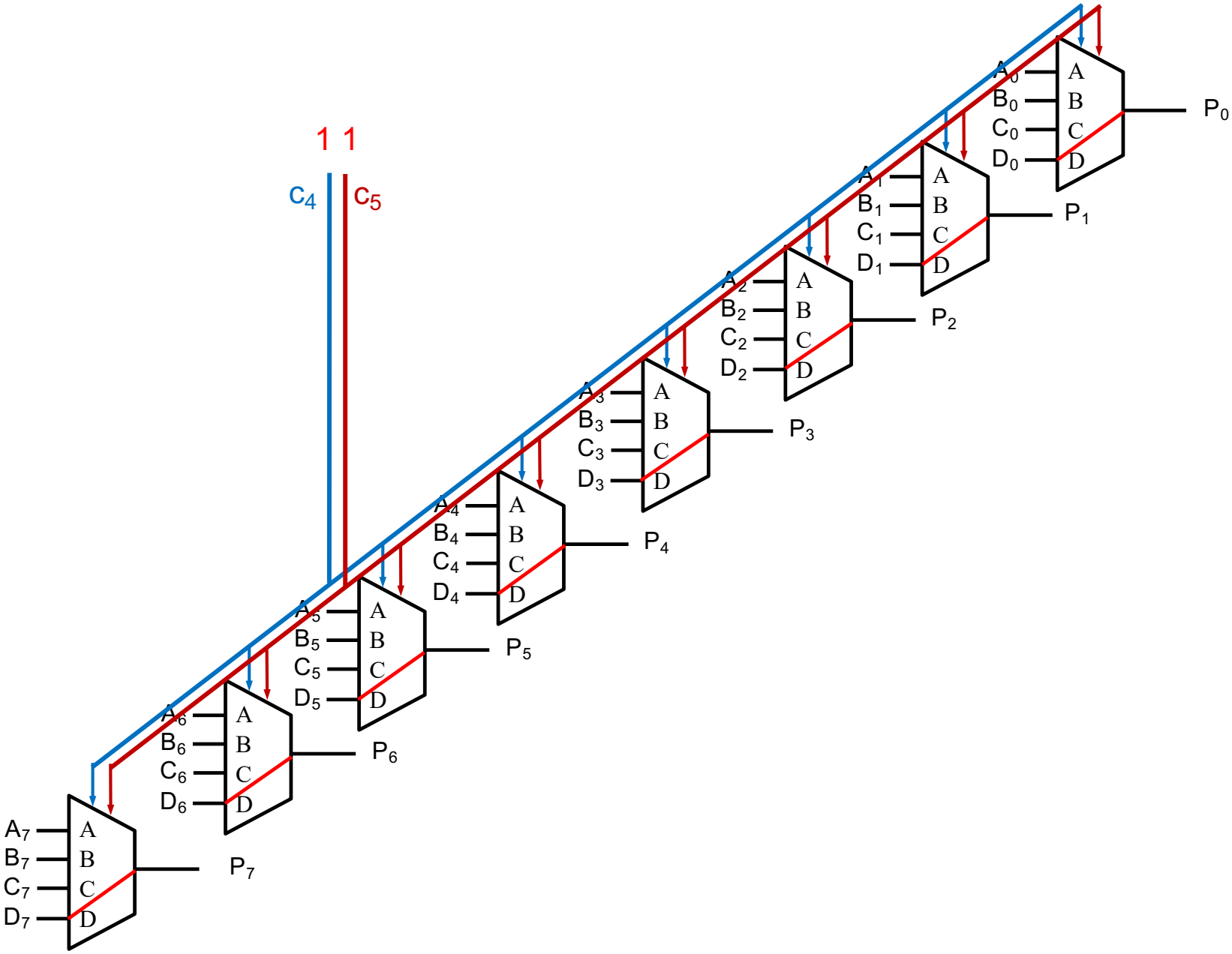

A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
P1
A2
B2
C2
D2
P2
A3
B3
C3
D3
P3
A4
B4
C4
D4
P4
A5
B5
C5
D5
P5
A6
B6
C6
D6
P6
A7
B7
C7
D7
P7

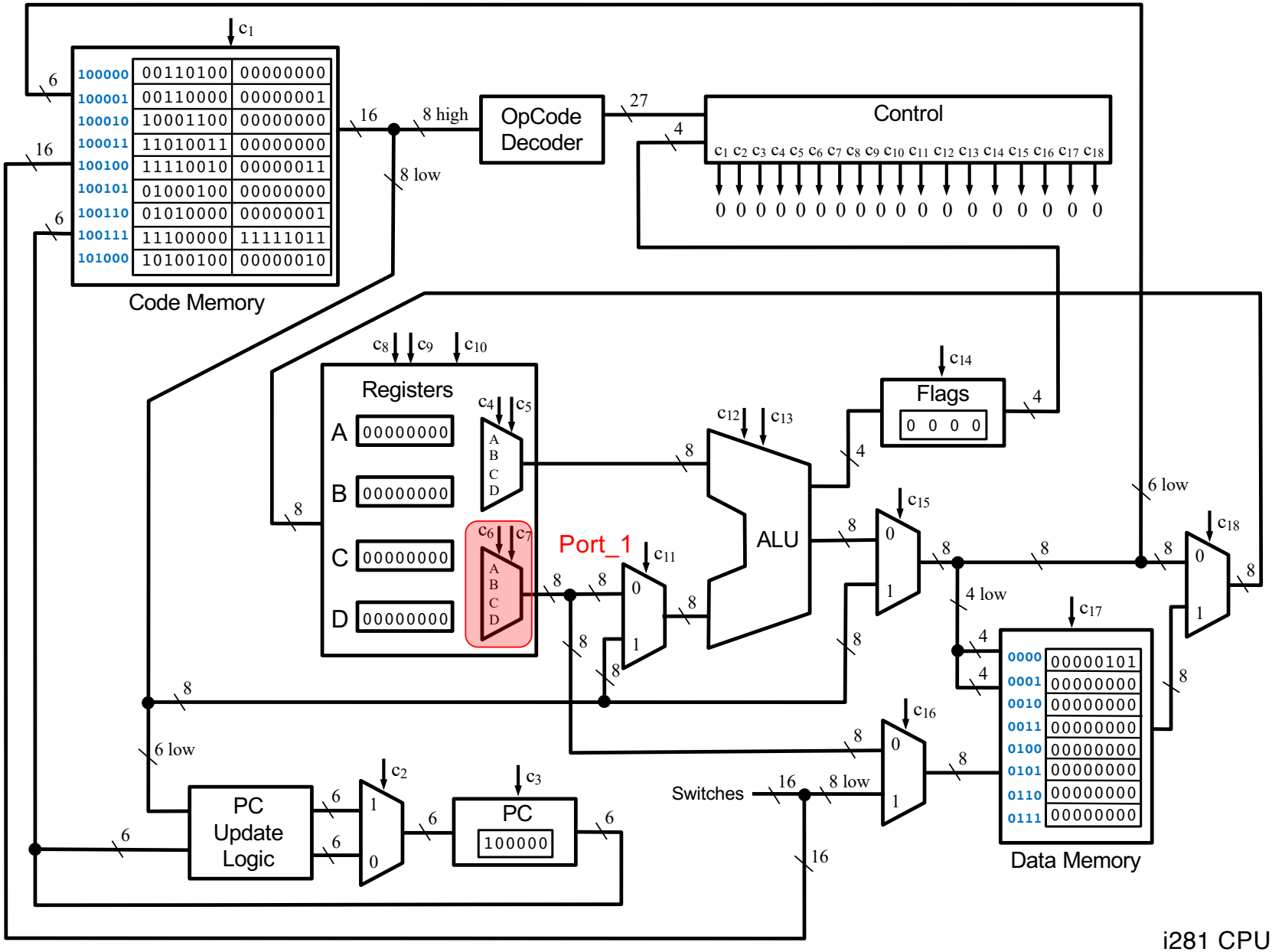

C4
C5
A0
B0
C0
D0
A
B
C
D
P0
A1
B1
C1
D1
P1
A2
B2
C2
D2
P2
A3
B3
C3
D3
P3
A4
B4
C4
D4
P4
A5
B5
C5
D5
P5
A6
B6
C6
D6
P6
A7
B7
C7
D7
P7


0 0
c_4
c_5
A_0
B_0
C_0
D_0
P_0
A_1
B_1
C_1
D_1
P_1
A_2
B_2
C_2
D_2
P_2
A_3
B_3
C_3
D_3
P_3
A_4
B_4
C_4
D_4
P_4
A_5
B_5
C_5
D_5
P_5
A_6
B_6
C_6
D_6
P_6
A_7
B_7
C_7
D_7
P_7
A
B
C
D

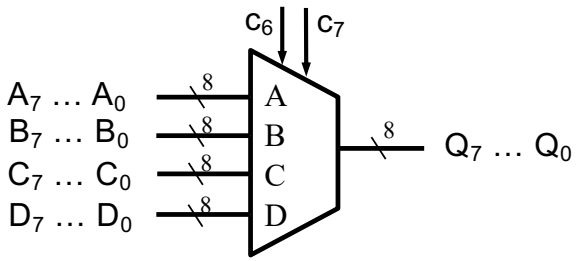

0 1
c_4
c_5
A_0
B_0
C_0
D_0
A
B
C
D
P_0
A_1
B_1
C_1
D_1
P_1
A_2
B_2
C_2
D_2
P_2
A_3
B_3
C_3
D_3
P_3
A_4
B_4
C_4
D_4
P_4
A_5
B_5
C_5
D_5
P_5
A_6
B_6
C_6
D_6
P_6
A_7
B_7
C_7
D_7
P_7

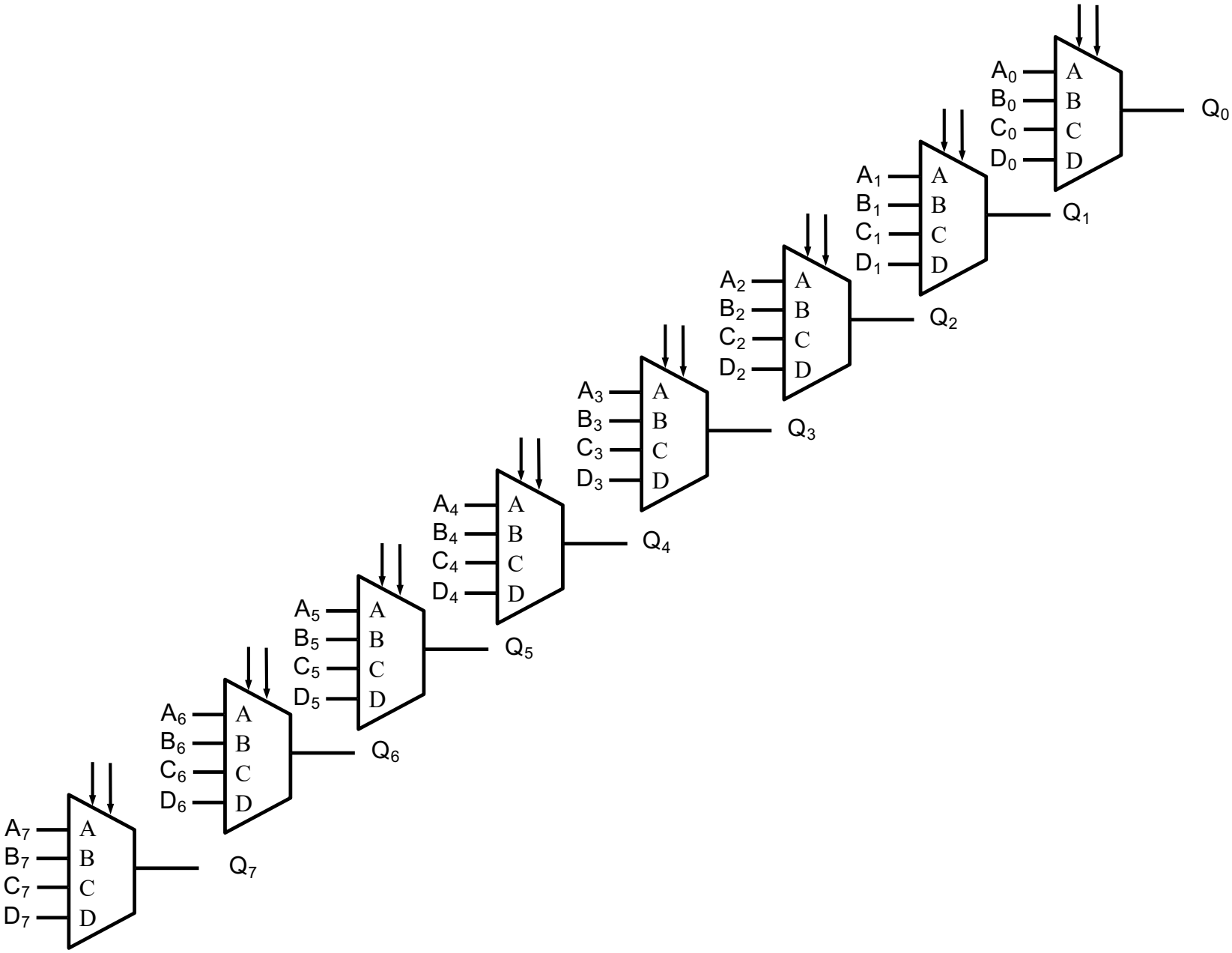

1 0
C4
C5
A7
B7
C7
D7
A
B
C
D
P7
A6
B6
C6
D6
P6
A5
B5
C5
D5
P5
A4
B4
C4
D4
P4
A3
B3
C3
D3
P3
A2
B2
C2
D2
P2
A1
B1
C1
D1
P1
A0
B0
C0
D0
P0

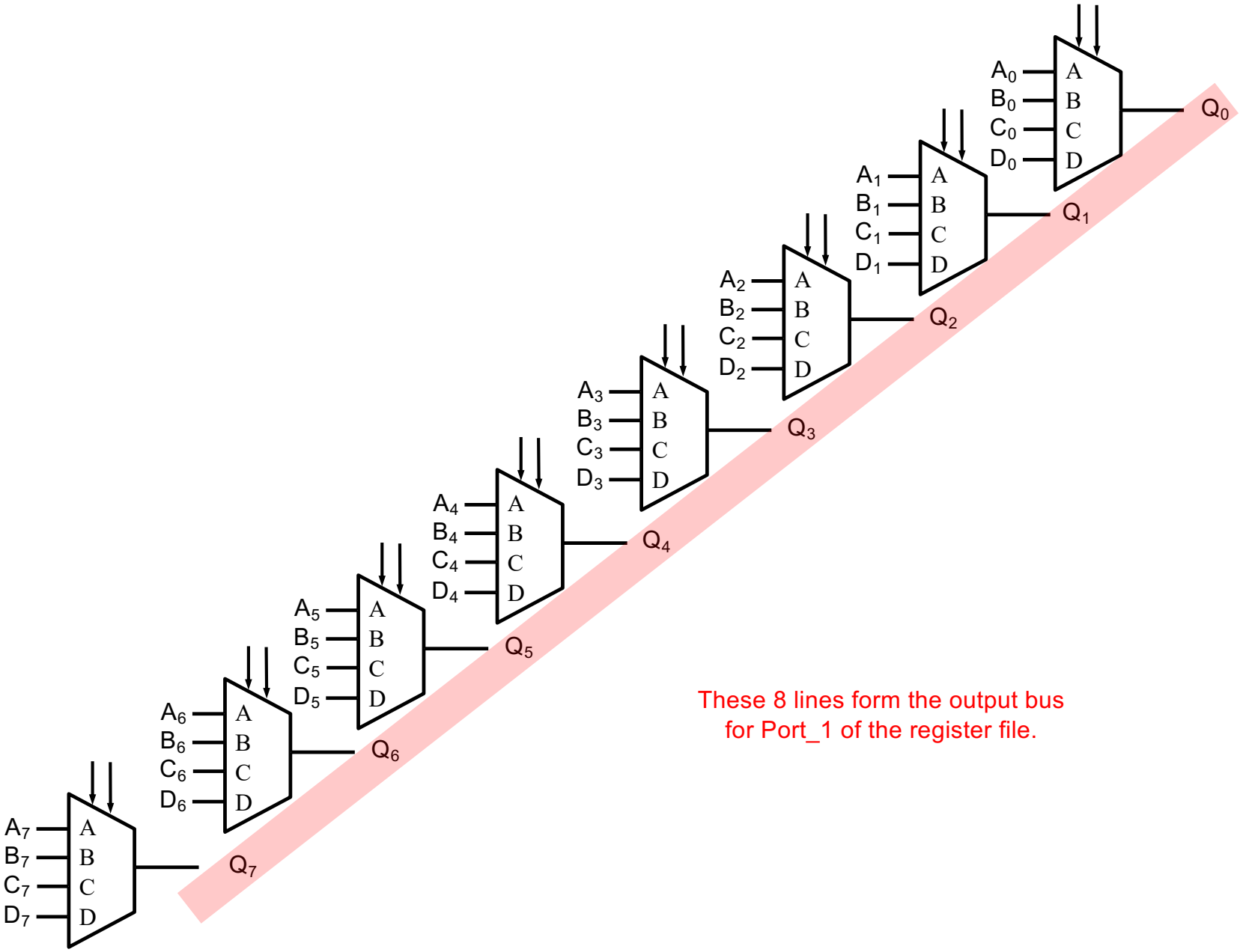

1 1
C4
C5
A7
B7
C7
D7
A
B
C
D
P7
A6
B6
C6
D6
P6
A5
B5
C5
D5
P5
A4
B4
C4
D4
P4
A3
B3
C3
D3
P3
A2
B2
C2
D2
P2
A1
B1
C1
D1
P1
A0
B0
C0
D0
P0

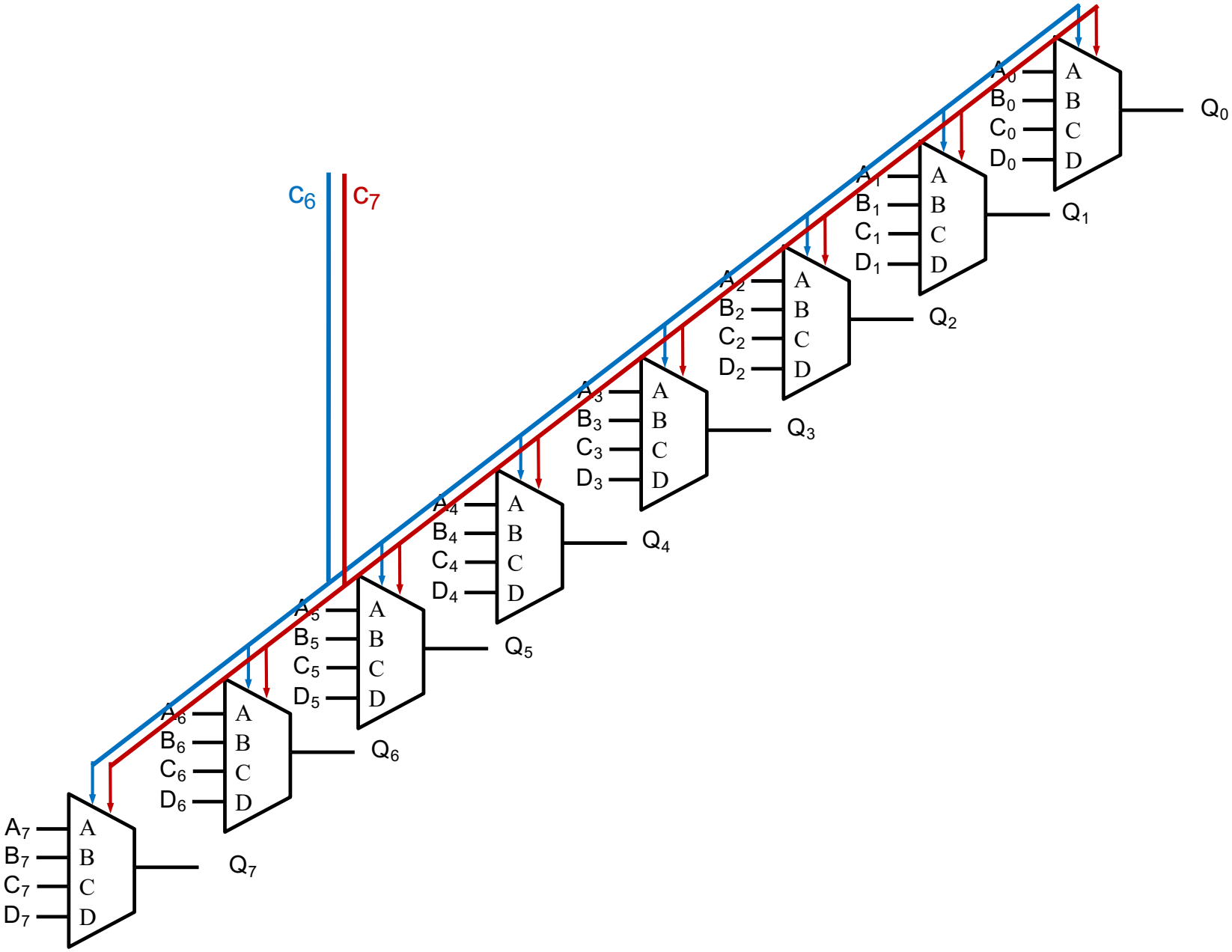

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
6
16
6
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 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
c15
Port_1
c11
0
1
8
6 low
c18
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
i281 CPU

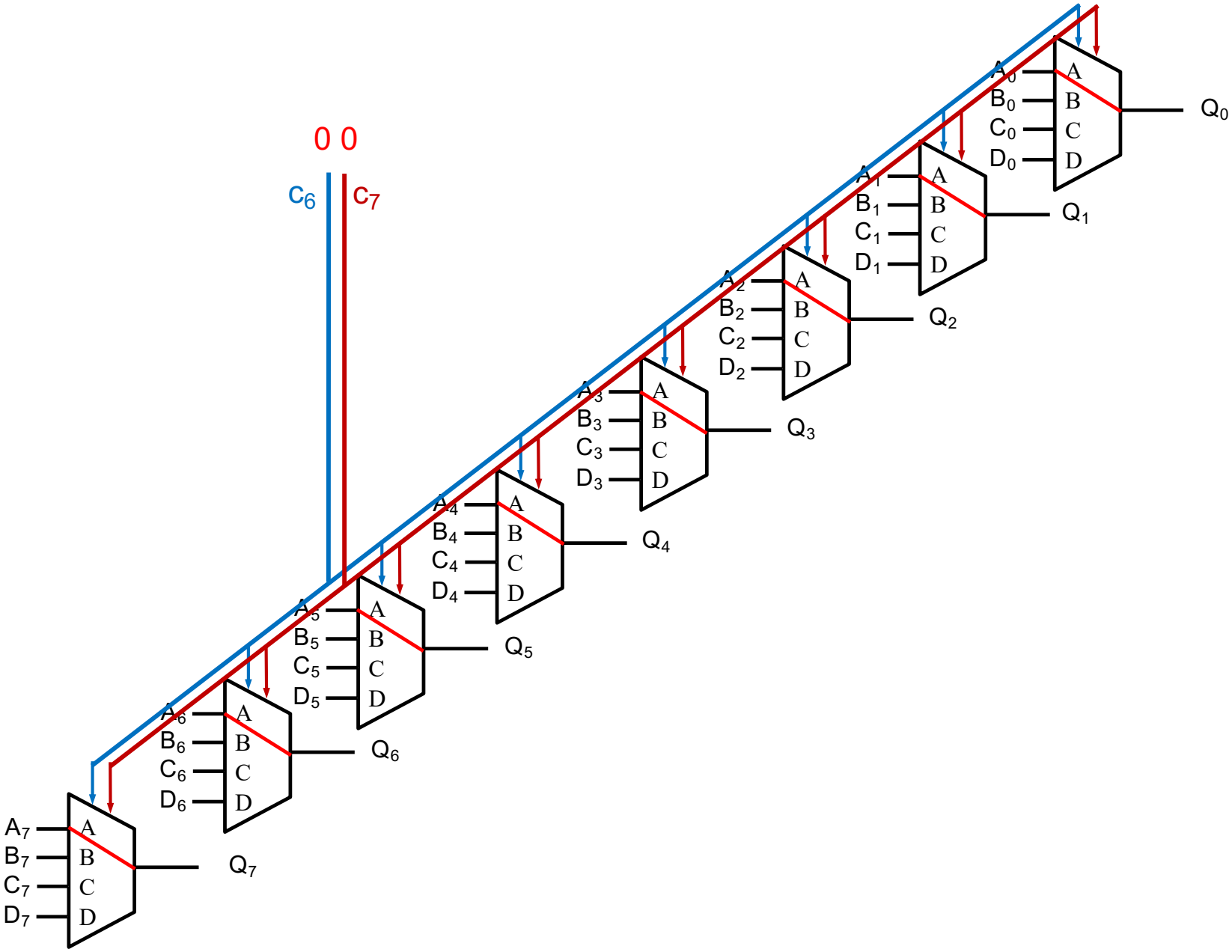
4-to-1 Bus Multiplexer (with 8-bit lines)

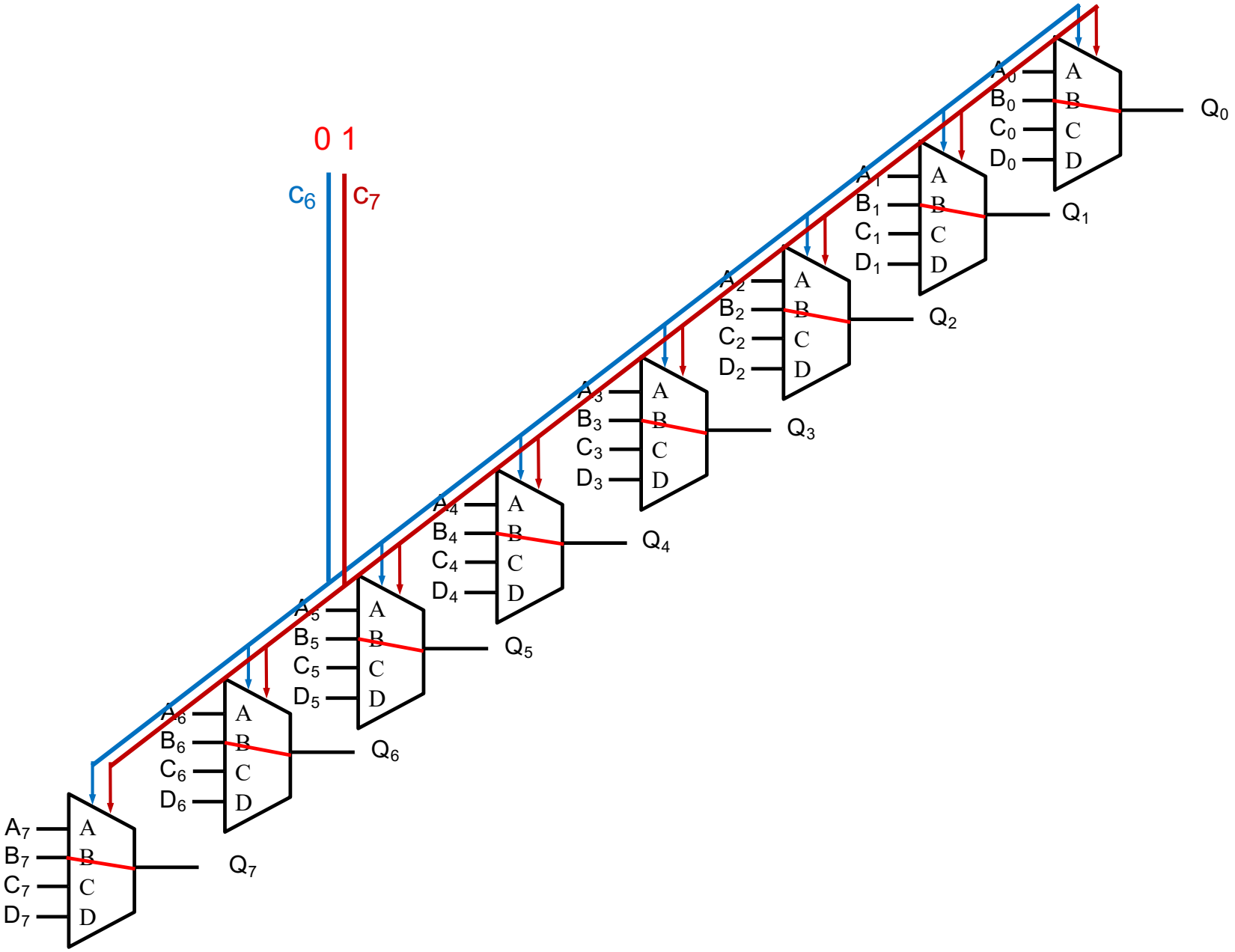


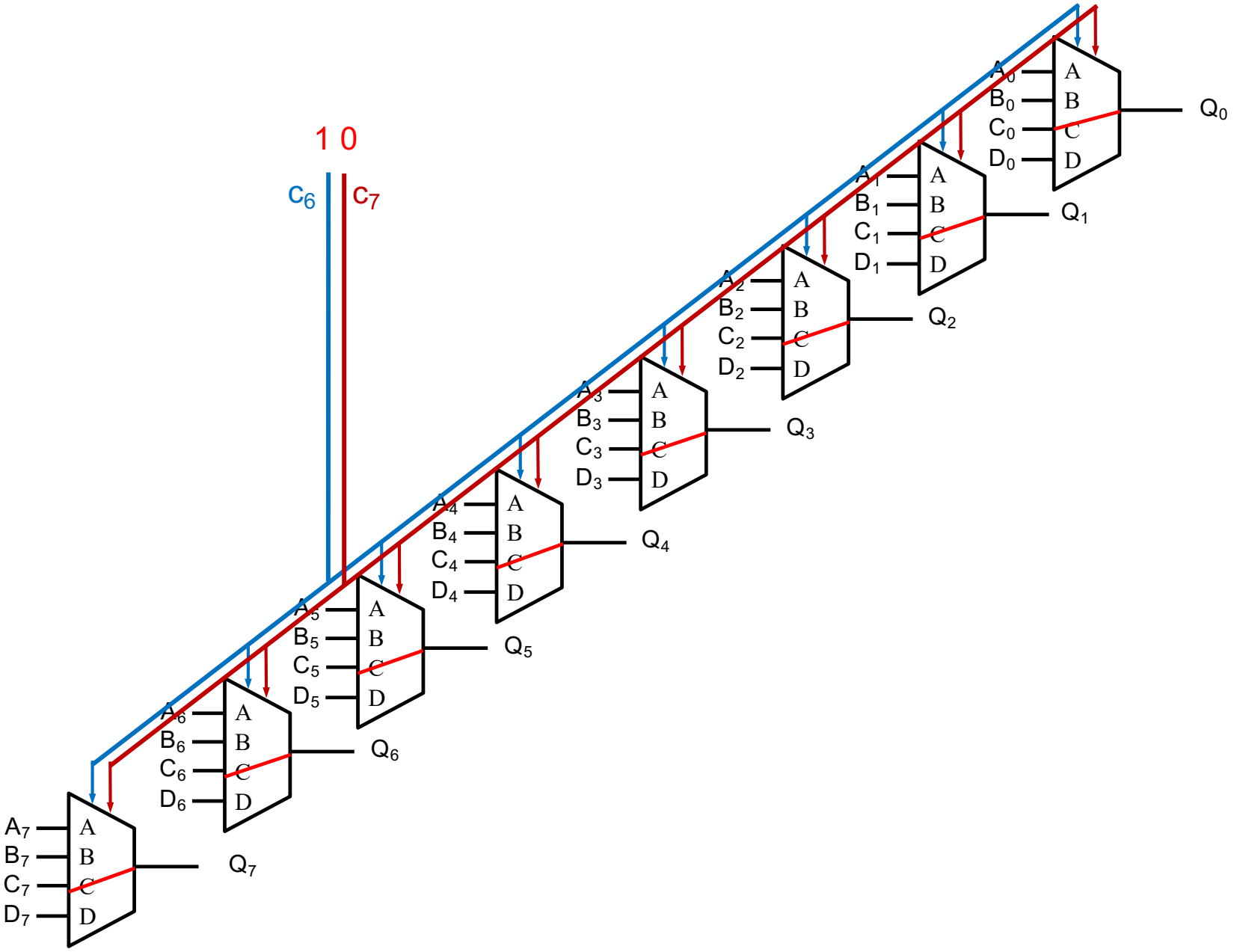

A0
B0
C0
D0
A
B
C
D
Q0
A1
B1
C1
D1
Q1
A2
B2
C2
D2
Q2
A3
B3
C3
D3
Q3
A4
B4
C4
D4
Q4
A5
B5
C5
D5
Q5
A6
B6
C6
D6
Q6
A7
B7
C7
D7
Q7

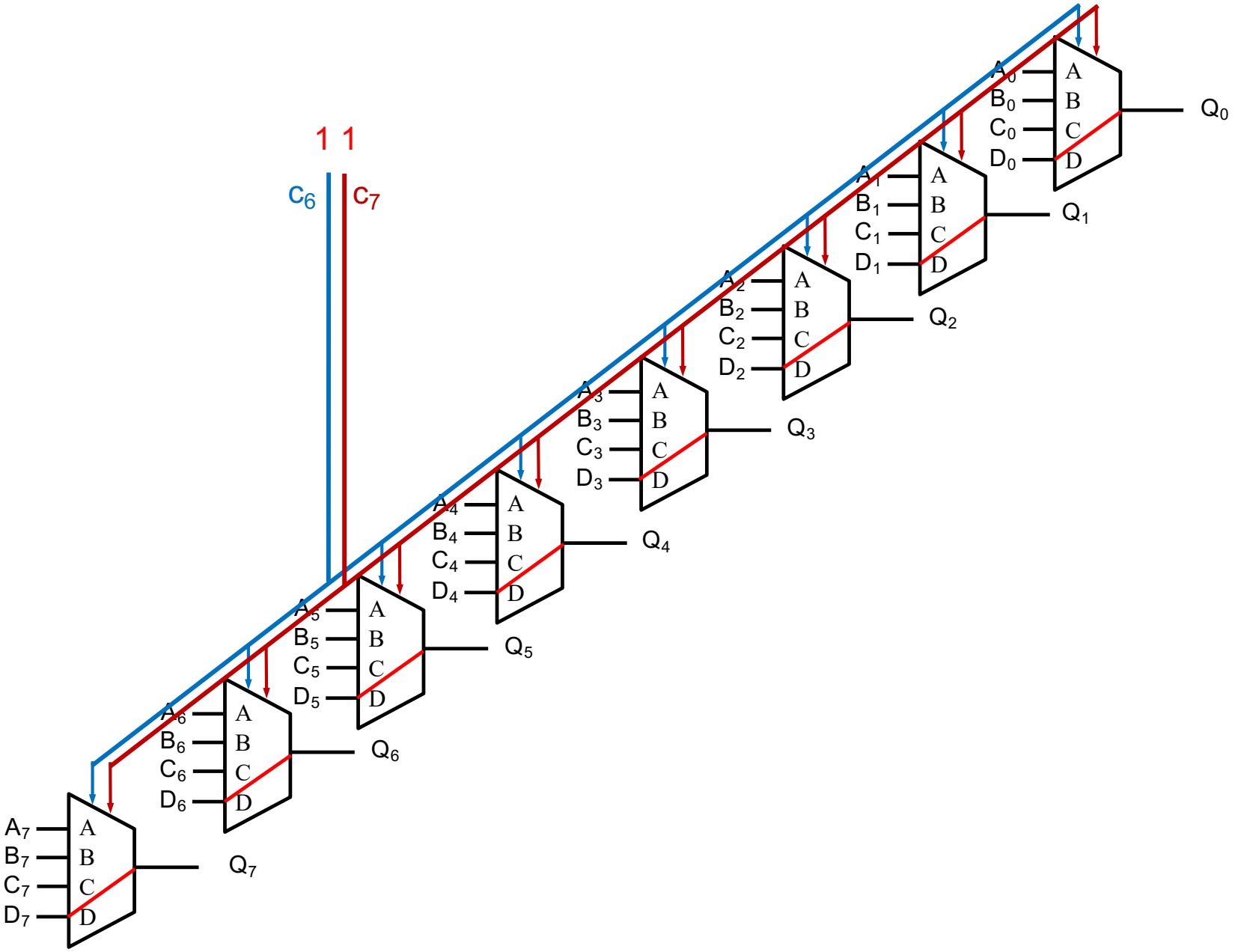

A0
B0
C0
D0
A
B
C
D
Q0
A1
B1
C1
D1
Q1
A2
B2
C2
D2
Q2
A3
B3
C3
D3
Q3
A4
B4
C4
D4
Q4
A5
B5
C5
D5
Q5
A6
B6
C6
D6
Q6
A7
B7
C7
D7
Q7
These 8 lines form the output bus
for Port_1 of the register file.

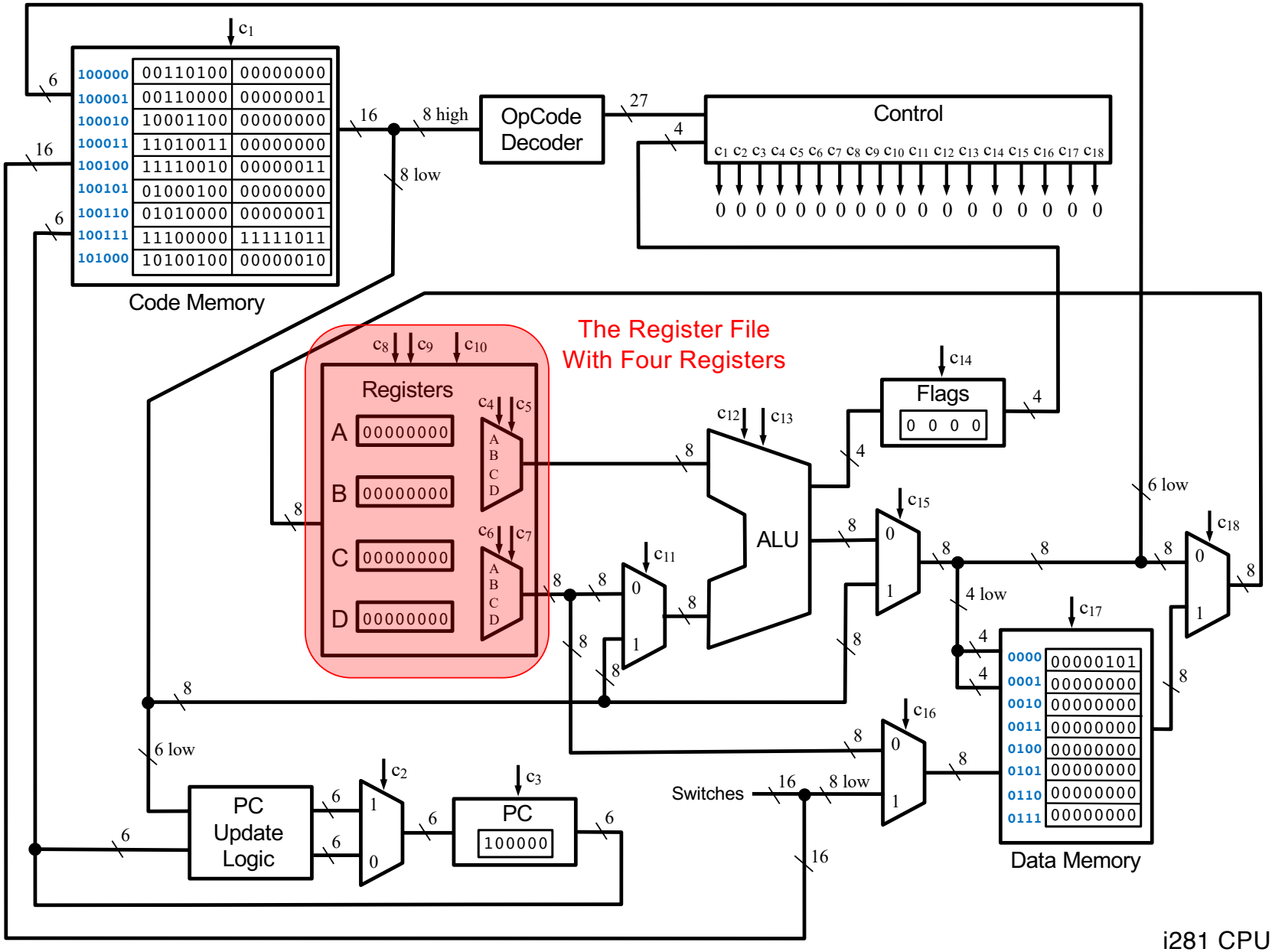

C6
C7
A7
B7
C7
D7
A
B
C
D
Q7
A6
B6
C6
D6
Q6
A5
B5
C5
D5
Q5
A4
B4
C4
D4
Q4
A3
B3
C3
D3
Q3
A2
B2
C2
D2
Q2
A1
B1
C1
D1
Q1
A0
B0
C0
D0
Q0


0 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


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

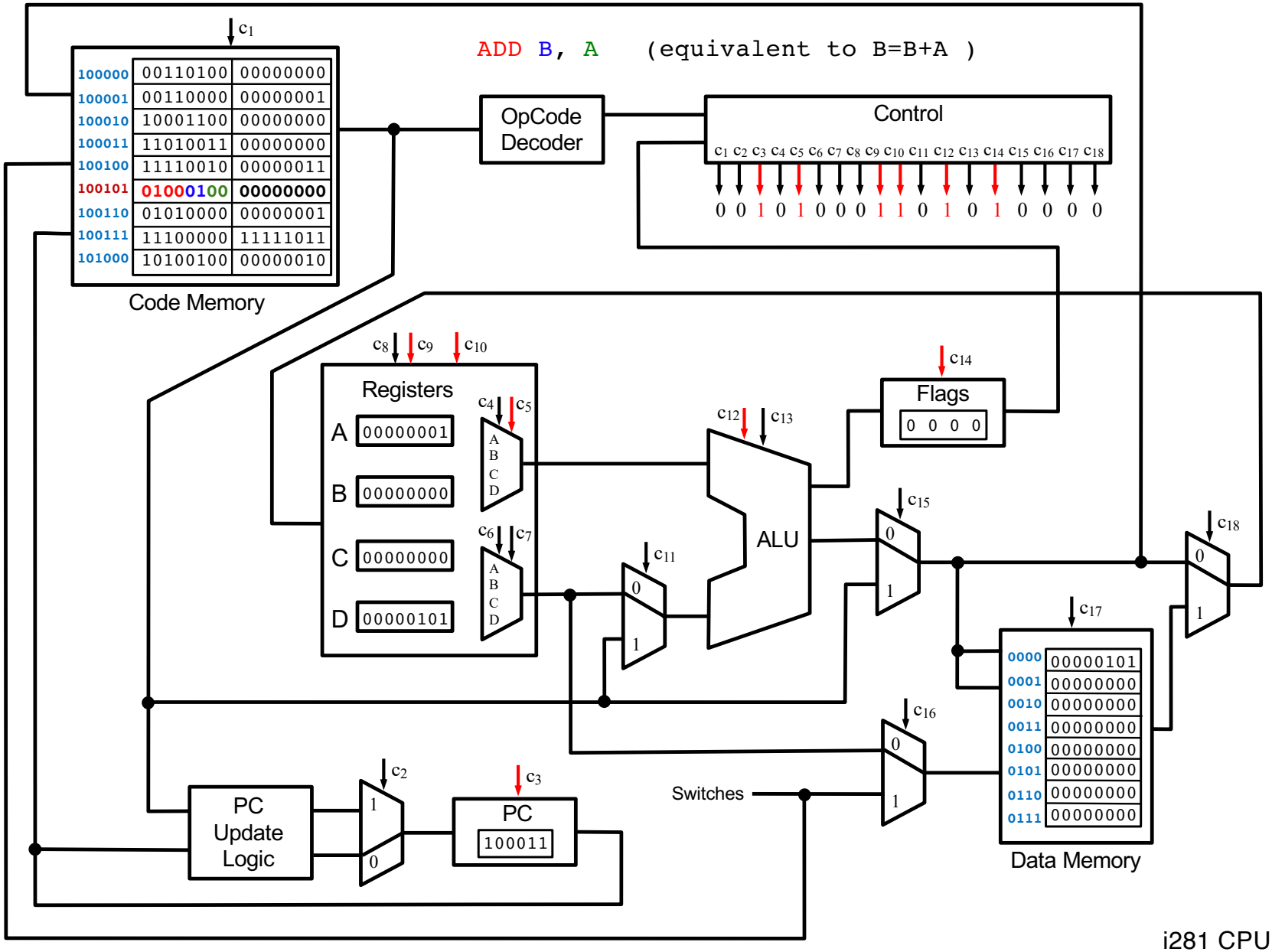

1 0
c_6
c_7
A_7
B_7
C_7
D_7
Q_7
A_6
B_6
C_6
D_6
Q_6
A_5
B_5
C_5
D_5
Q_5
A_4
B_4
C_4
D_4
Q_4
A_3
B_3
C_3
D_3
Q_3
A_2
B_2
C_2
D_2
Q_2
A_1
B_1
C_1
D_1
Q_1
A_0
B_0
C_0
D_0
Q_0
A
B
C
D

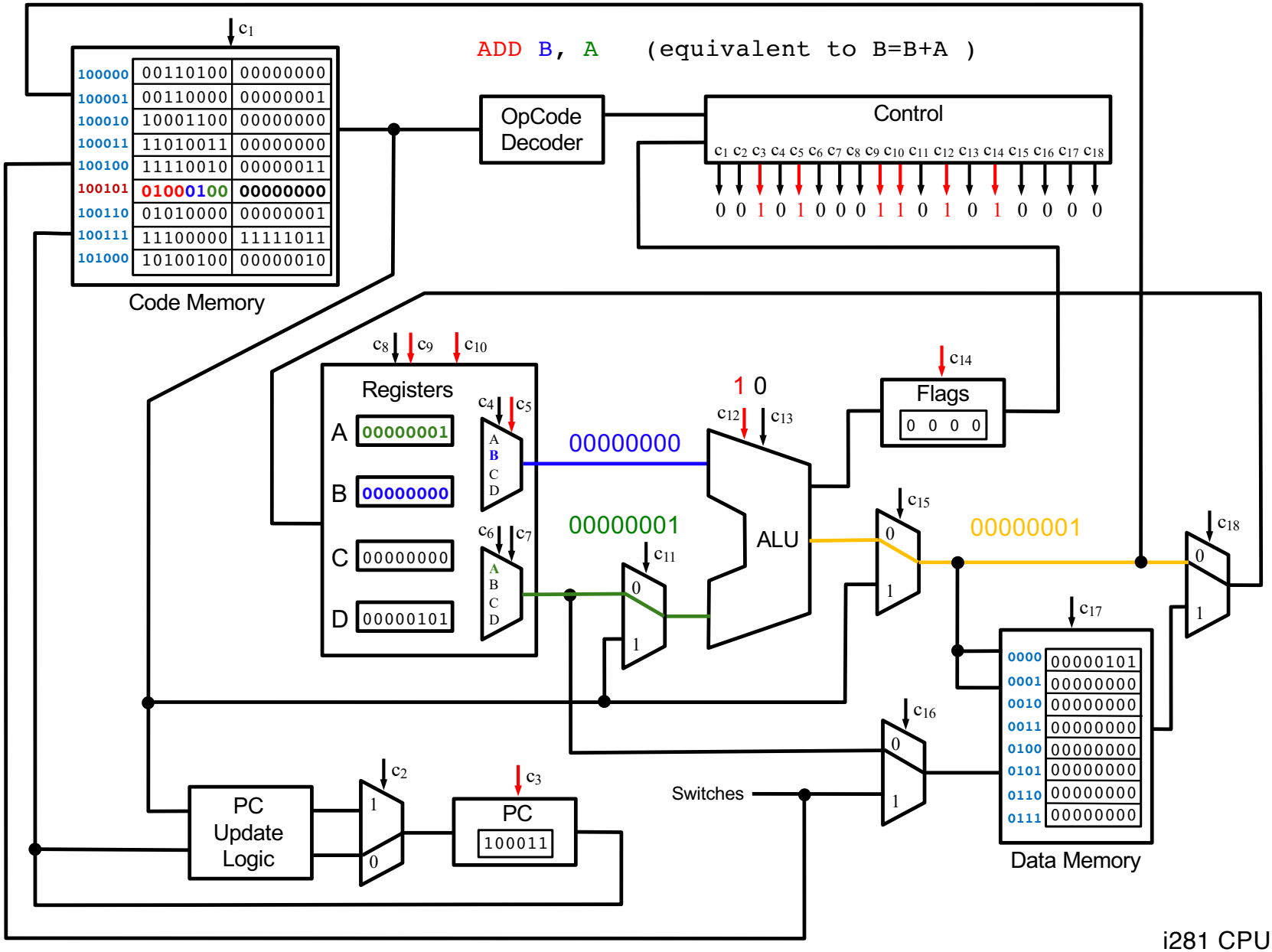

1 1
c_6
c_7
A_7
B_7
C_7
D_7
A_6
B_6
C_6
D_6
A_5
B_5
C_5
D_5
A_4
B_4
C_4
D_4
A_3
B_3
C_3
D_3
A_2
B_2
C_2
D_2
A_1
B_1
C_1
D_1
A_0
B_0
C_0
D_0
A
B
C
D
Q_7
Q_6
Q_5
Q_4
Q_3
Q_2
Q_1
Q_0

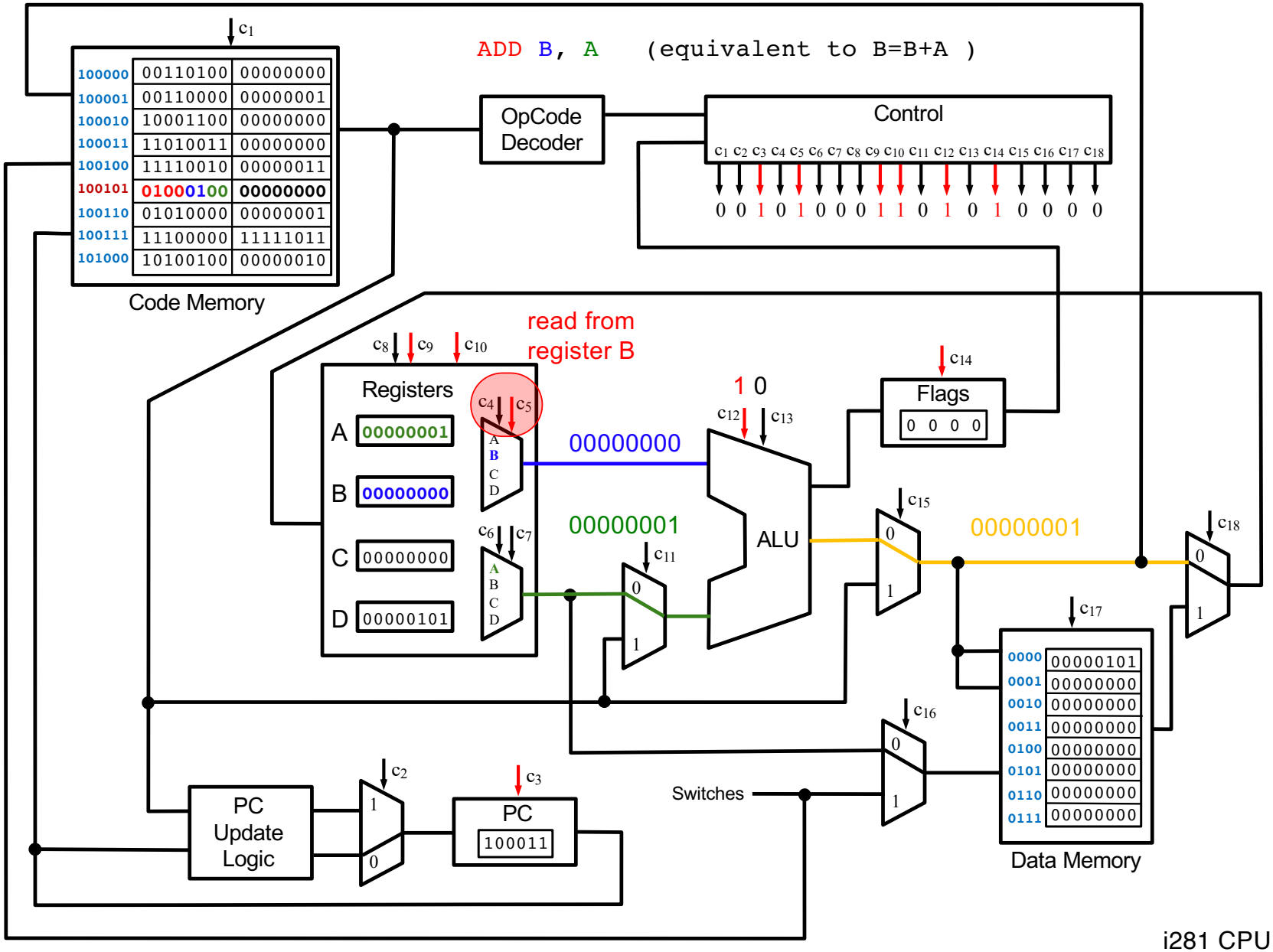

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 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
c12 c13
ALU
c14
Flags
0 0 0 0
c15
c11
c16
c17
c18
6 low
4 low
8 low
Switches
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

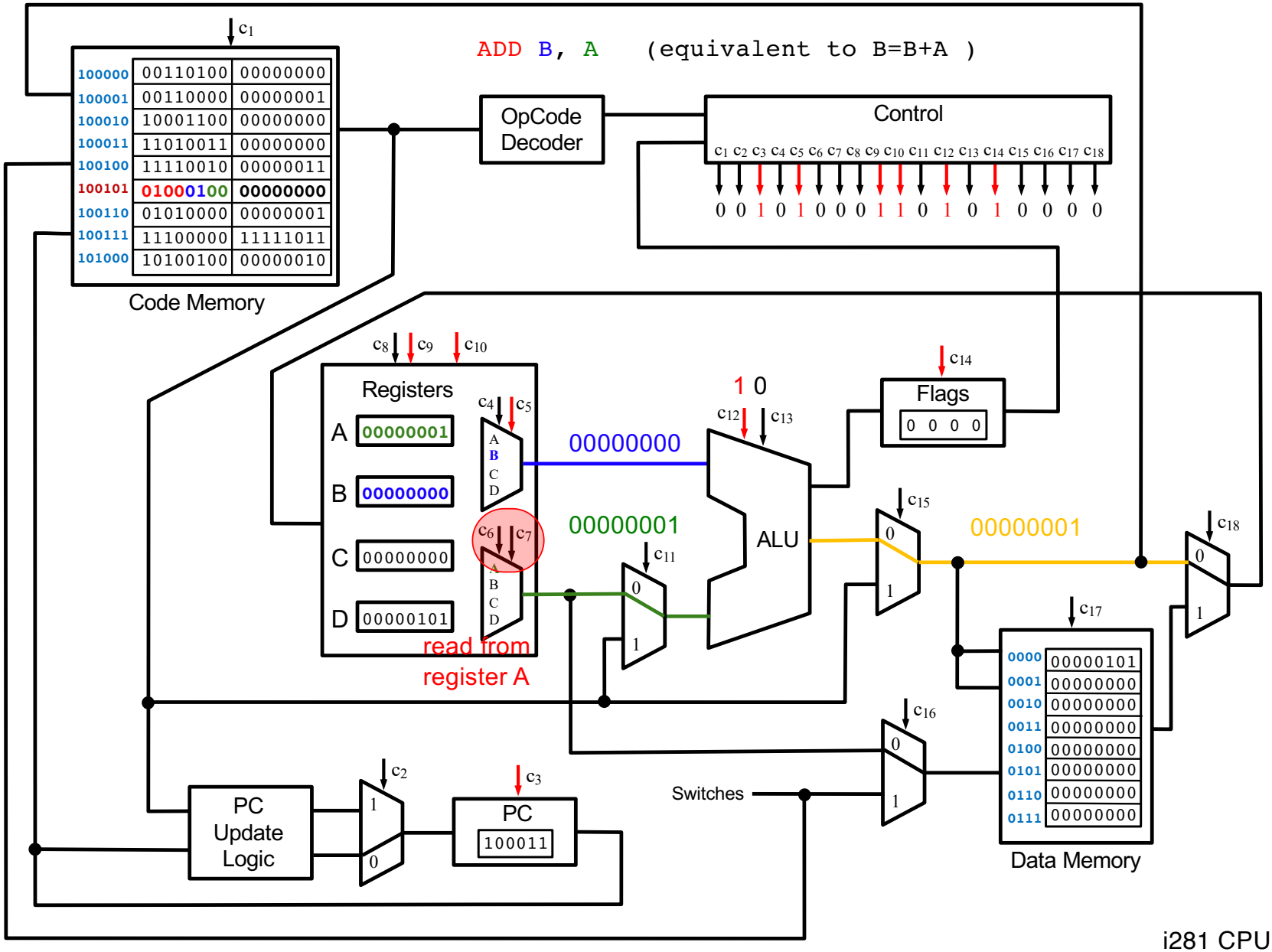
Simulation of the execution of one line of assembly code:

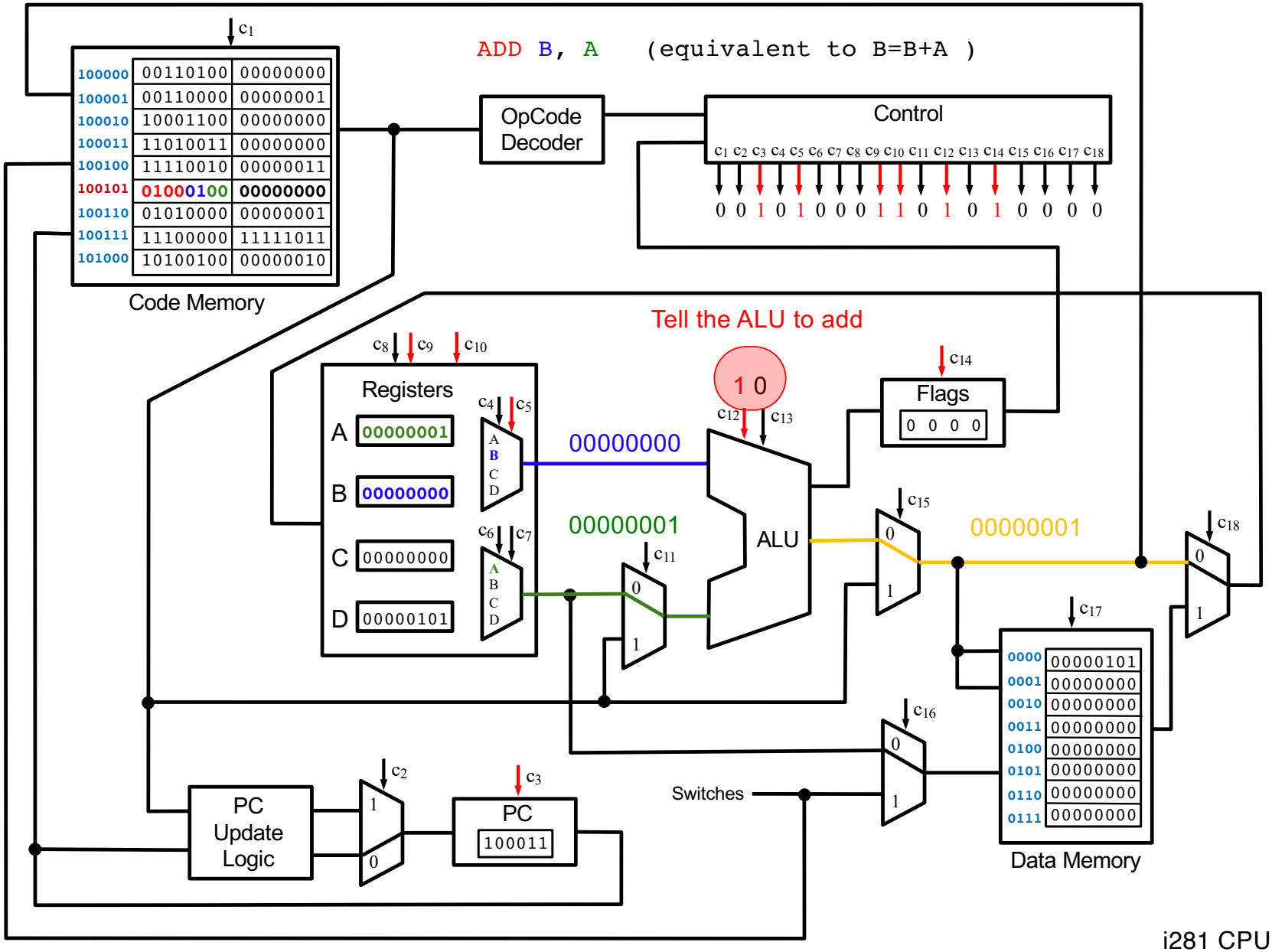
ADD B, A

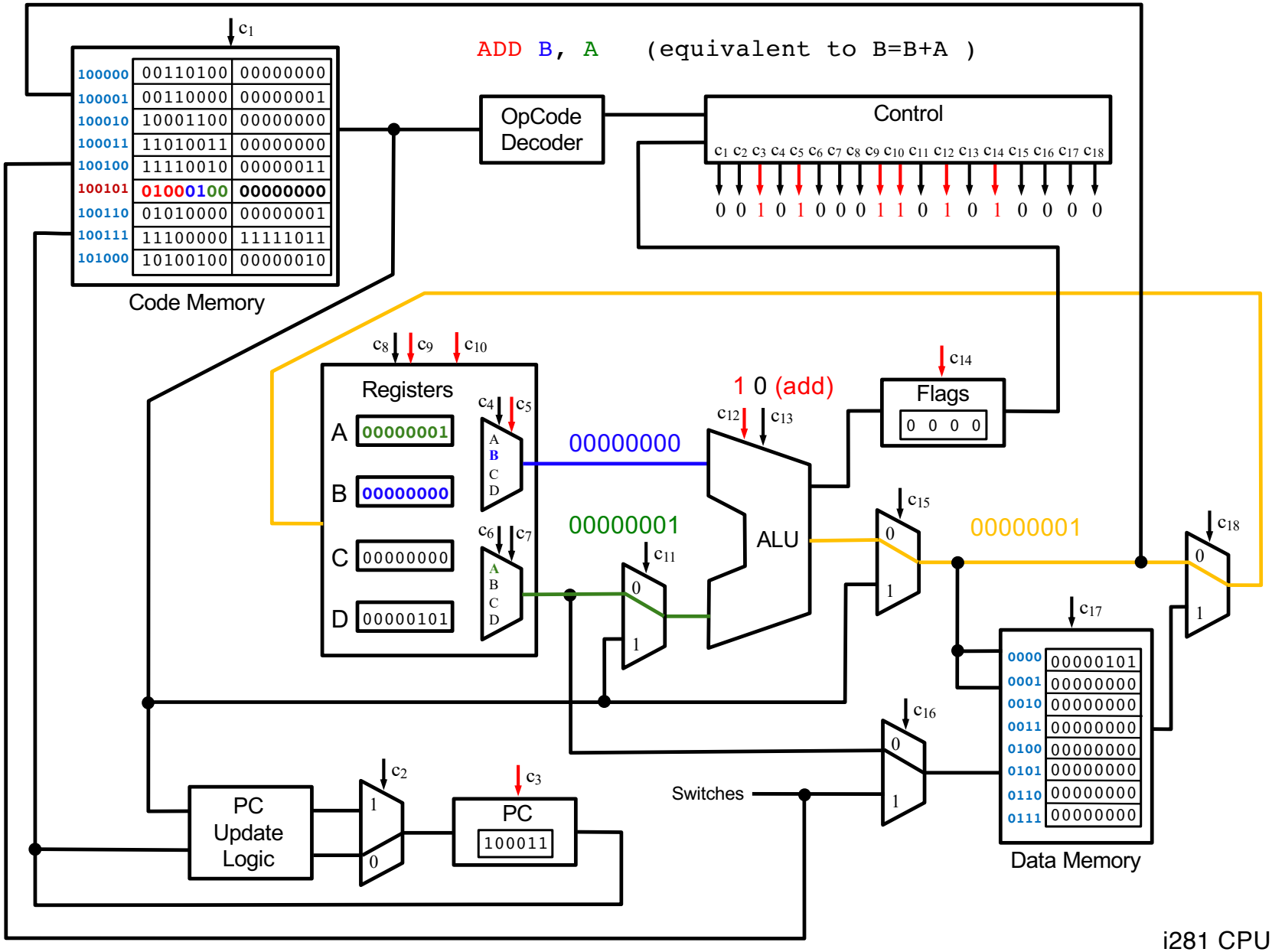

ADD B, A (equivalent to B=B+A)
c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
OpCode Decoder
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0
c8 c9 c10
Registers
A 00000001
B 00000000
C 00000000
D 00000101
c4 c5
c6 c7
A B C D
c14
Flags
0 0 0 0
c12 c13
ALU
c15
c11
c18
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
c2
c3
PC Update Logic
PC
100011
i281 CPU

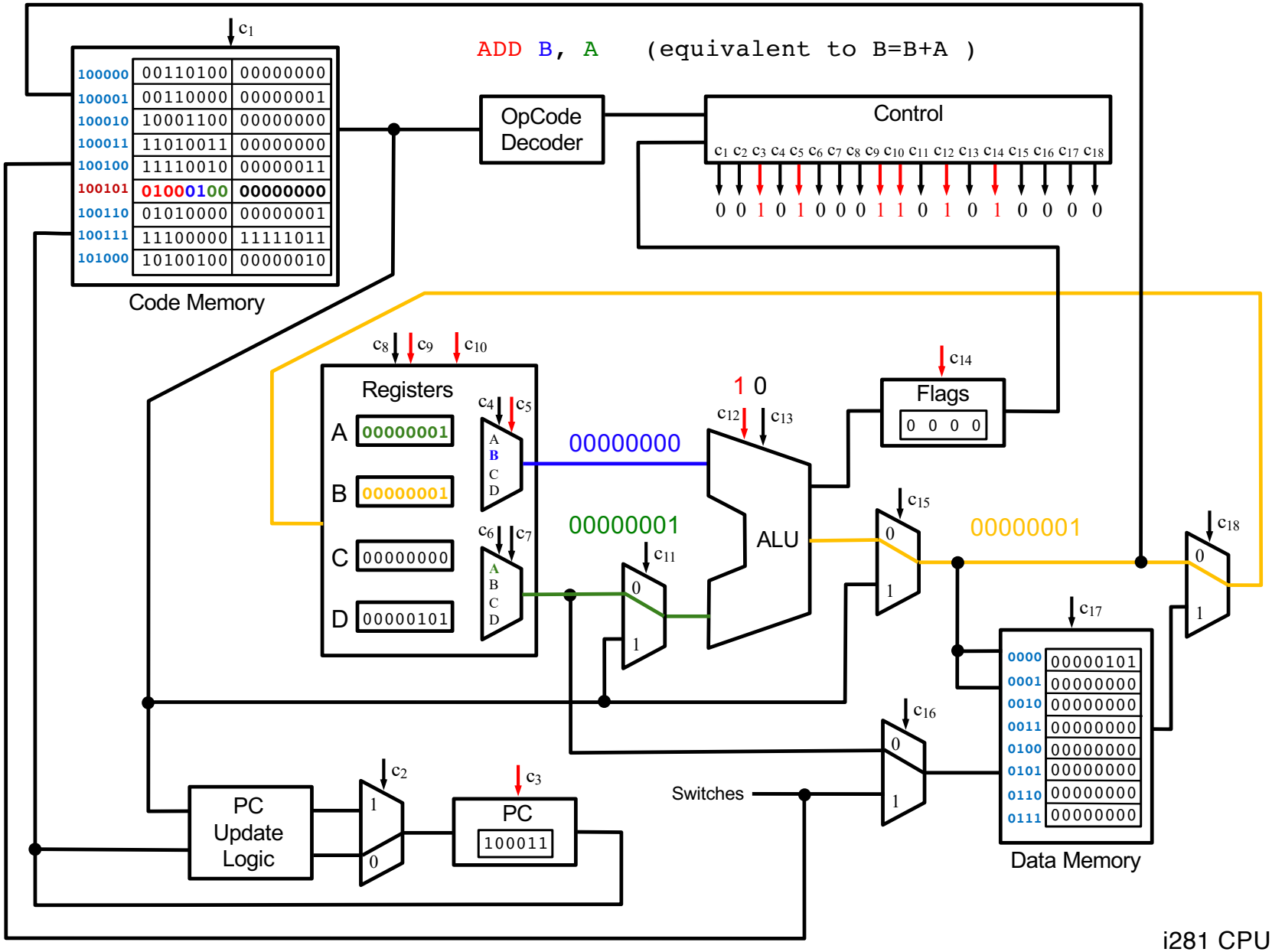

ADD B, A (equivalent to B=B+A)
c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
OpCode Decoder
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0
c8 c9 c10
Registers
A 00000001
B 00000000
C 00000000
D 00000101
c4 c5
c6 c7
A B C D
00000000
00000001
c11
1 0
c12 c13
ALU
c14
Flags
0 0 0 0
c15
00000001
c18
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
c2
c3
PC Update Logic
PC
100011
i281 CPU

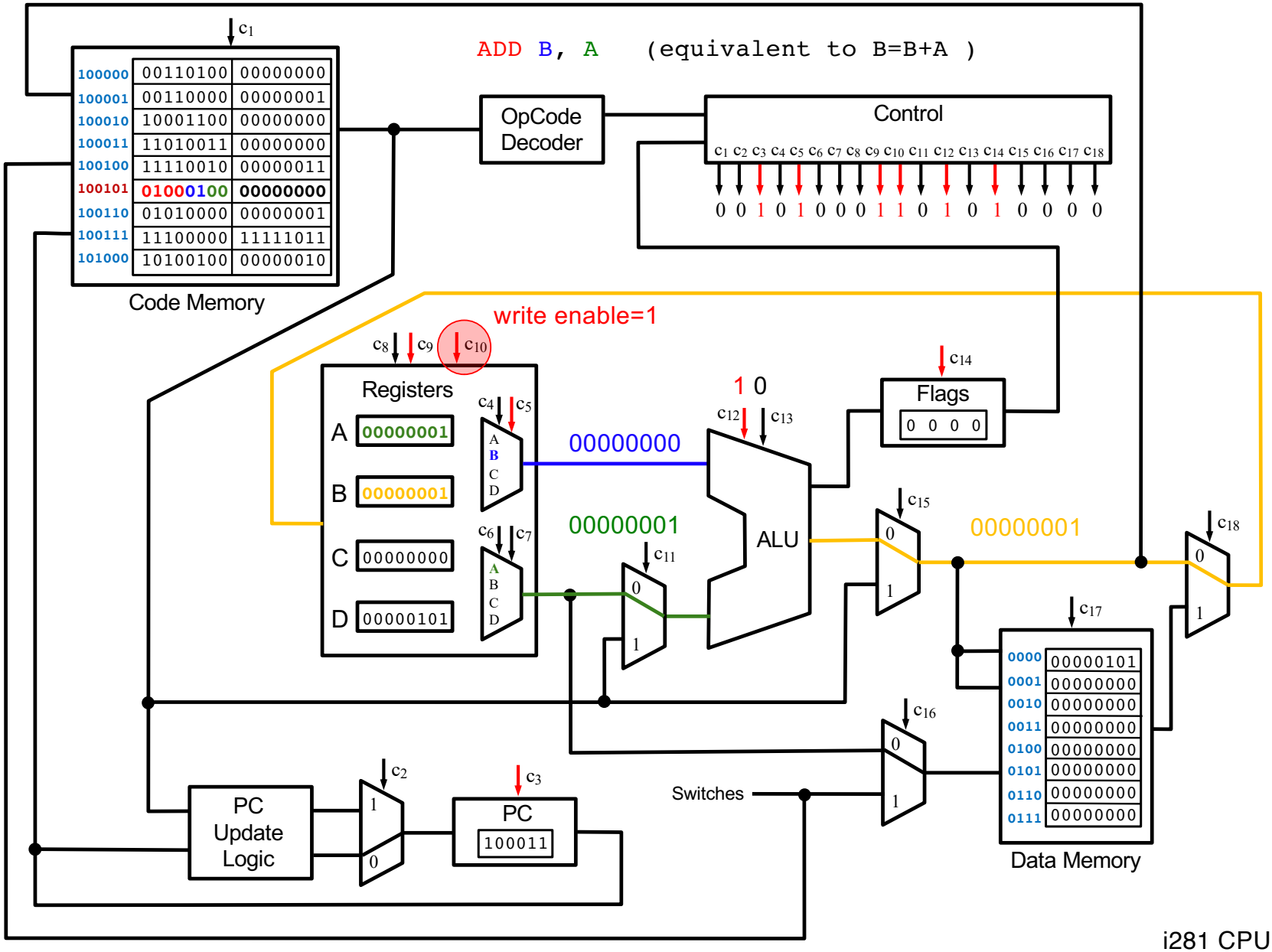

ADD B, A (equivalent to B=B+A)
c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
OpCode Decoder
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0
read from register B
c8 c9 c10
Registers
A 00000001
B 00000000
C 00000000
D 00000101
c4 c5
c6 c7
A B C D
00000000
00000001
c11
1 0
c12 c13
ALU
c14
Flags
0 0 0 0
c15
00000001
c18
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
c2
c3
PC Update Logic
PC
100011
i281 CPU

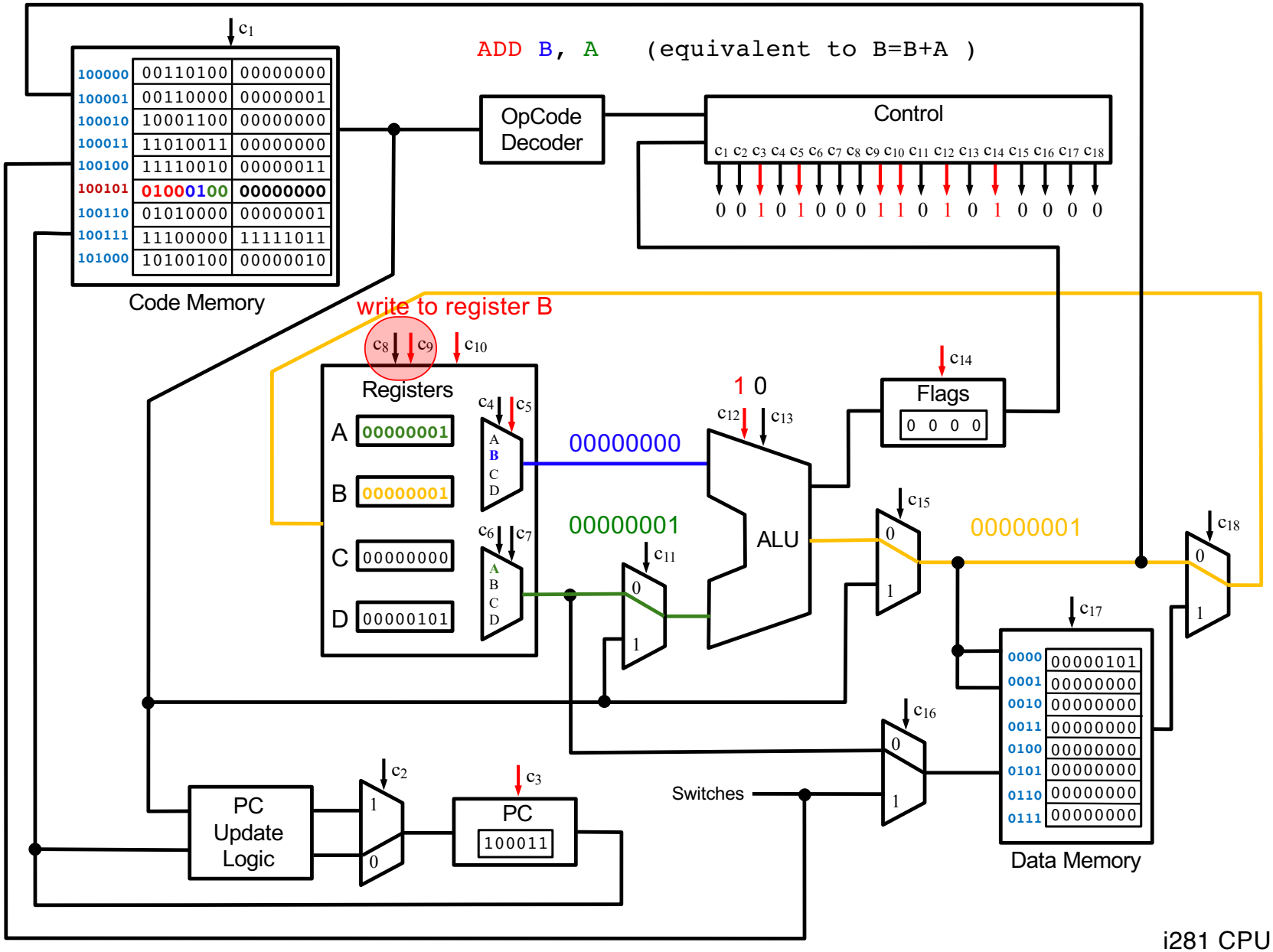

ADD B, A (equivalent to B=B+A)
c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
OpCode Decoder
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0
c8 c9 c10
Registers
A 00000001
B 00000000
C 00000000
D 00000101
c4 c5
A B C D
c6 c7
A B C D
read from register A
00000000
00000001
c11
0 1
1 0
c12 c13
ALU
c14
Flags
0 0 0 0
c15
0 1
00000001
c18
0 1
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
0 1
Switches
c2
1 0
c3
PC Update Logic
PC
100011
i281 CPU

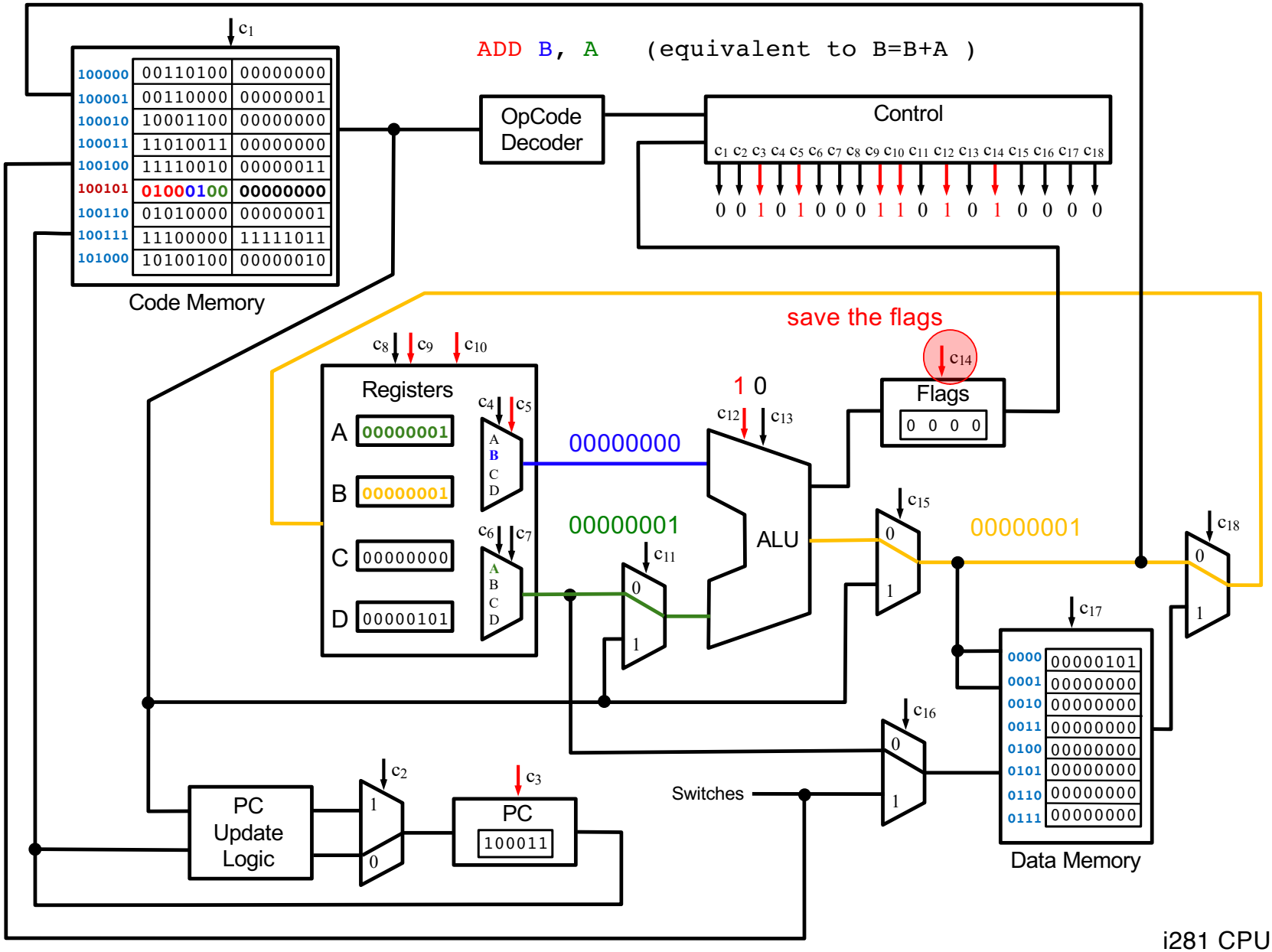

ADD B, A (equivalent to B=B+A)
c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
OpCode Decoder
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0
Tell the ALU to add
c8 c9 c10
Registers
A 00000001
B 00000000
C 00000000
D 00000101
c4 c5
c6 c7
A B C D
1 0
c12 c13
00000000
00000001
c11
ALU
c14
Flags
0 0 0 0
c15
00000001
c18
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
c2
c3
PC Update Logic
PC
100011
i281 CPU


ADD B, A (equivalent to B=B+A)
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
OpCode Decoder
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0
c8 c9 c10
Registers
A 00000001
B 00000000
C 00000000
D 00000101
c4 c5
c6 c7
A B C D
00000000
00000001
c11
1 0 (add)
c12 c13
ALU
c14
Flags
0 0 0 0
c15
00000001
c18
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
c2
c3
PC Update Logic
PC
100011
i281 CPU


ADD B, A (equivalent to B=B+A)
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
OpCode Decoder
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0
c8 c9 c10
Registers
A 00000001
B 00000001
C 00000000
D 00000101
c4 c5
c6 c7
A B C D
00000000
00000001
1 0
c12 c13
ALU
c14
Flags
0 0 0 0
c15
c11
00000001
c18
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
c2
c3
PC Update Logic
PC
100011
i281 CPU


ADD B, A (equivalent to B=B+A)
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
OpCode Decoder
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0
write enable=1
c8 c9 c10
Registers
A 00000001
B 00000001
C 00000000
D 00000101
c4 c5
c6 c7
A B C D
00000000
00000001
c11
1 0
c12 c13
ALU
c14
Flags
0 0 0 0
c15
00000001
c18
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
c2
c3
PC Update Logic
PC
100011
i281 CPU


ADD B, A (equivalent to B=B+A)
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
OpCode Decoder
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0
write to register B
c8 c9 c10
Registers
A 00000001
B 00000001
C 00000000
D 00000101
c4 c5
c6 c7
A B C D
00000000
00000001
c11
1 0
c12 c13
ALU
c14
Flags
0 0 0 0
c15
00000001
c18
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
c2
c3
PC Update Logic
PC
100011
i281 CPU


ADD B, A (equivalent to B=B+A)
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
OpCode Decoder
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0
save the flags
c14
Flags
0 0 0 0
c8 c9 c10
Registers
A 00000001
B 00000001
C 00000000
D 00000101
c4 c5
c6 c7
A B C D
00000000
00000001
1 0
c12 c13
ALU
c11
c15
00000001
c18
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
c2
c3
PC Update Logic
PC
100011
i281 CPU

Questions?

THE END