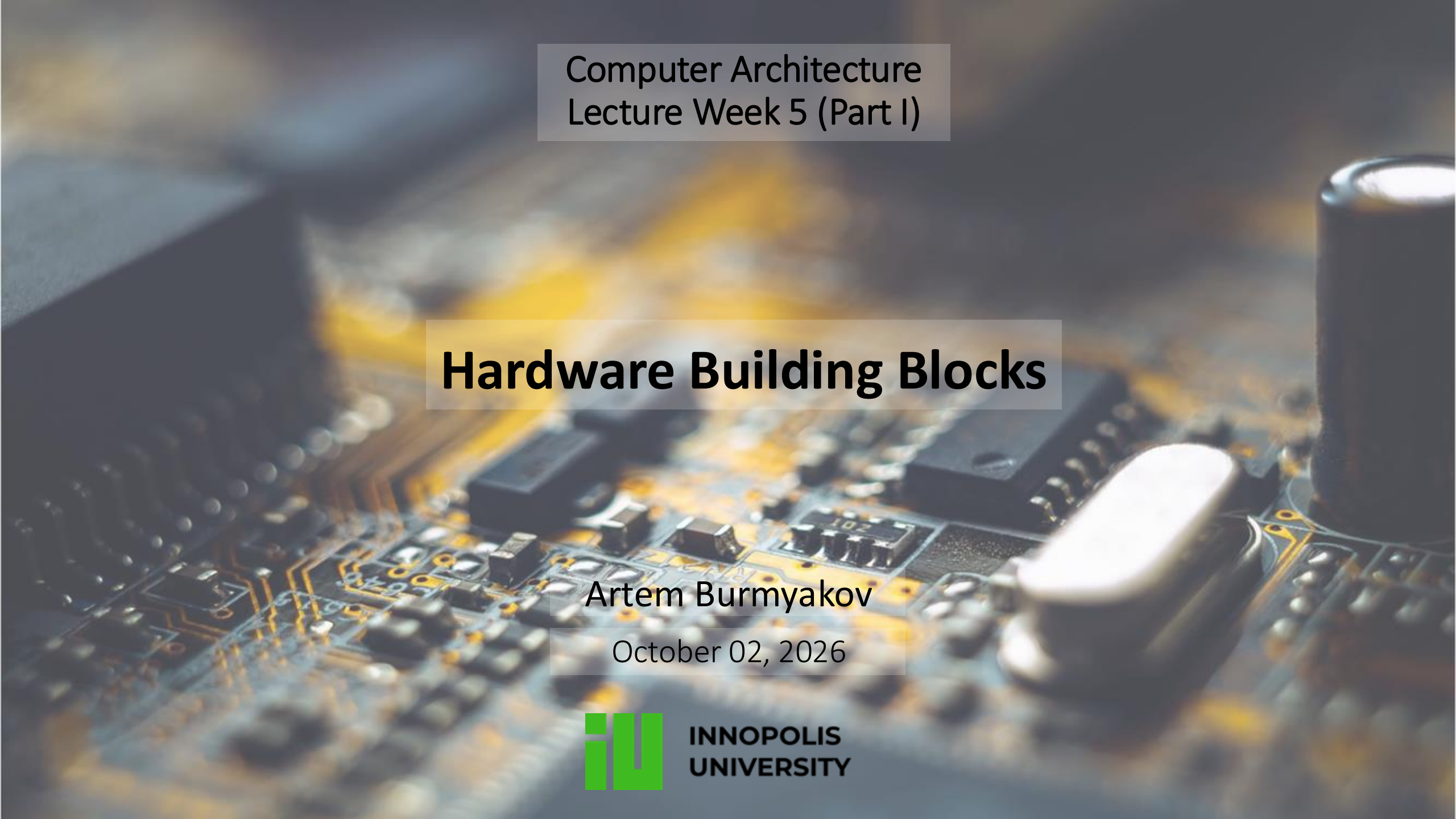




Computer Architecture
Lecture Week 5 (Part I)

Hardware Building Blocks

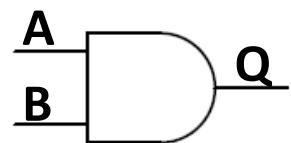
Artem Burmyakov

October 02, 2026



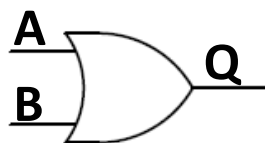
INNOPOLIS
UNIVERSITY

Recap: Basic Logic Gates



AND

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



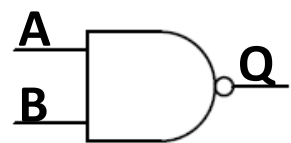
OR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1



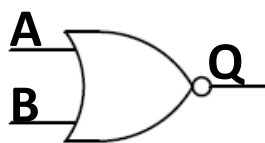
XOR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0



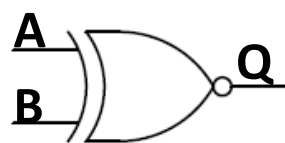
NAND

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



NOR

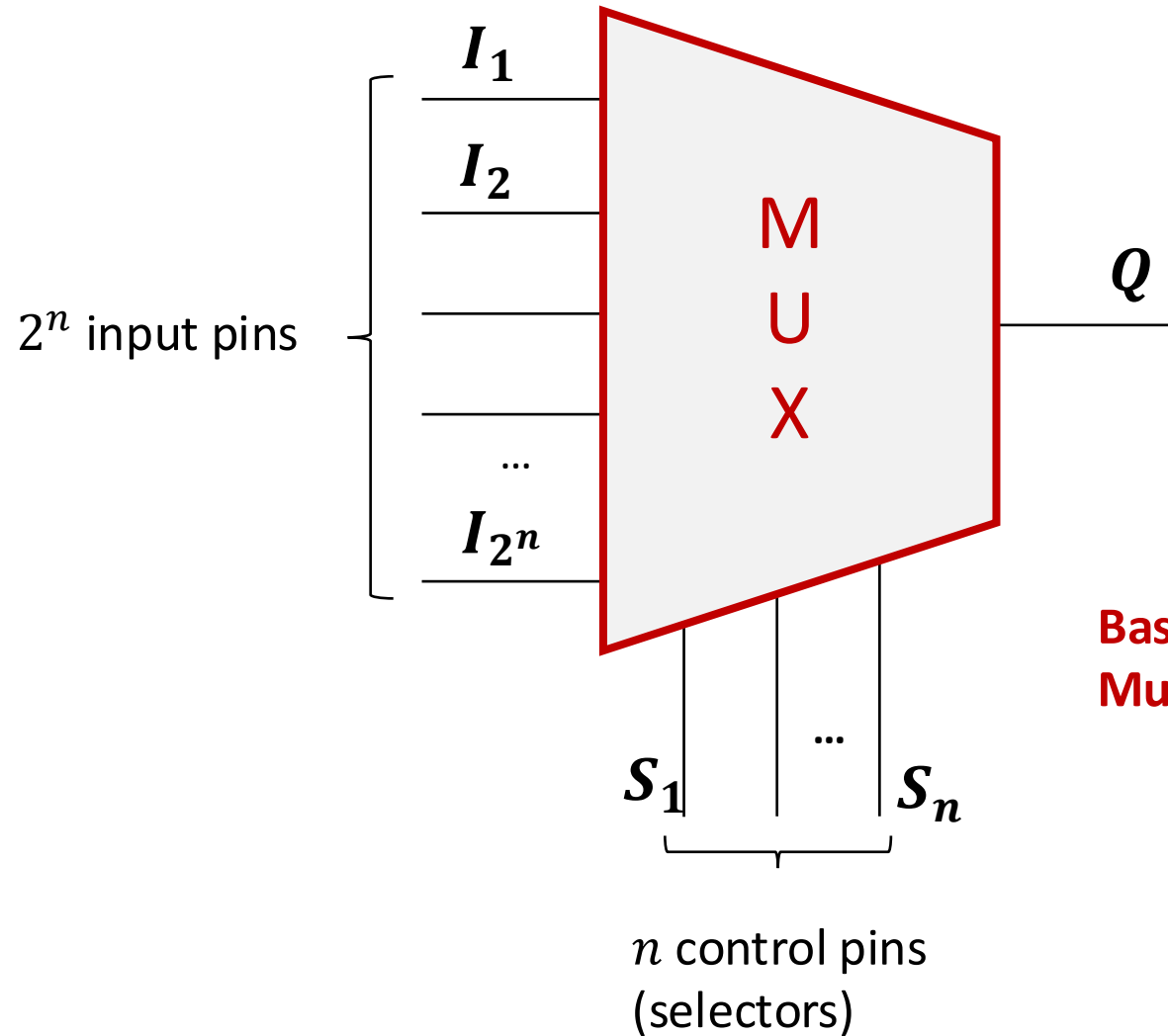
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



XNOR

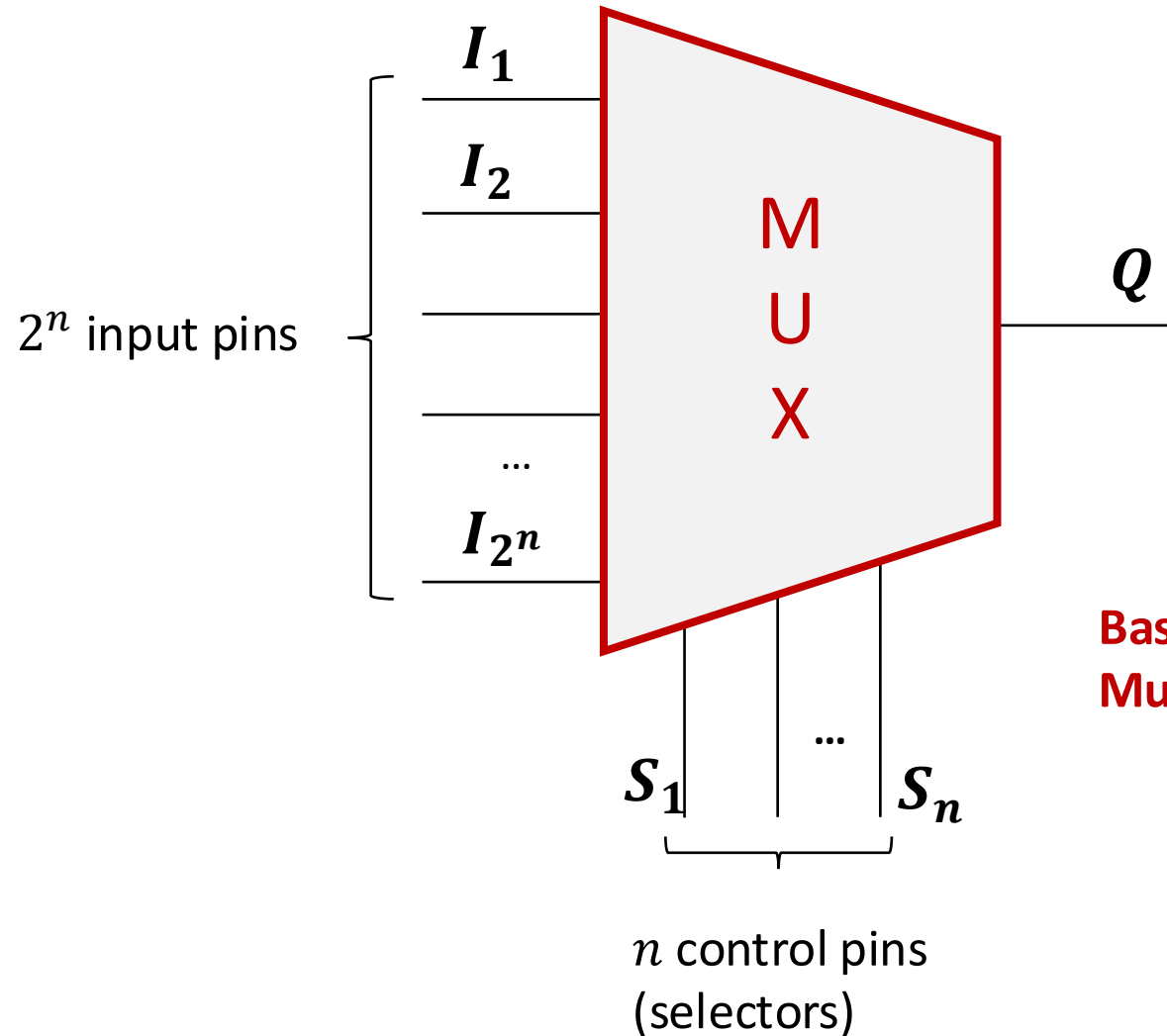
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Consider the following circuit:



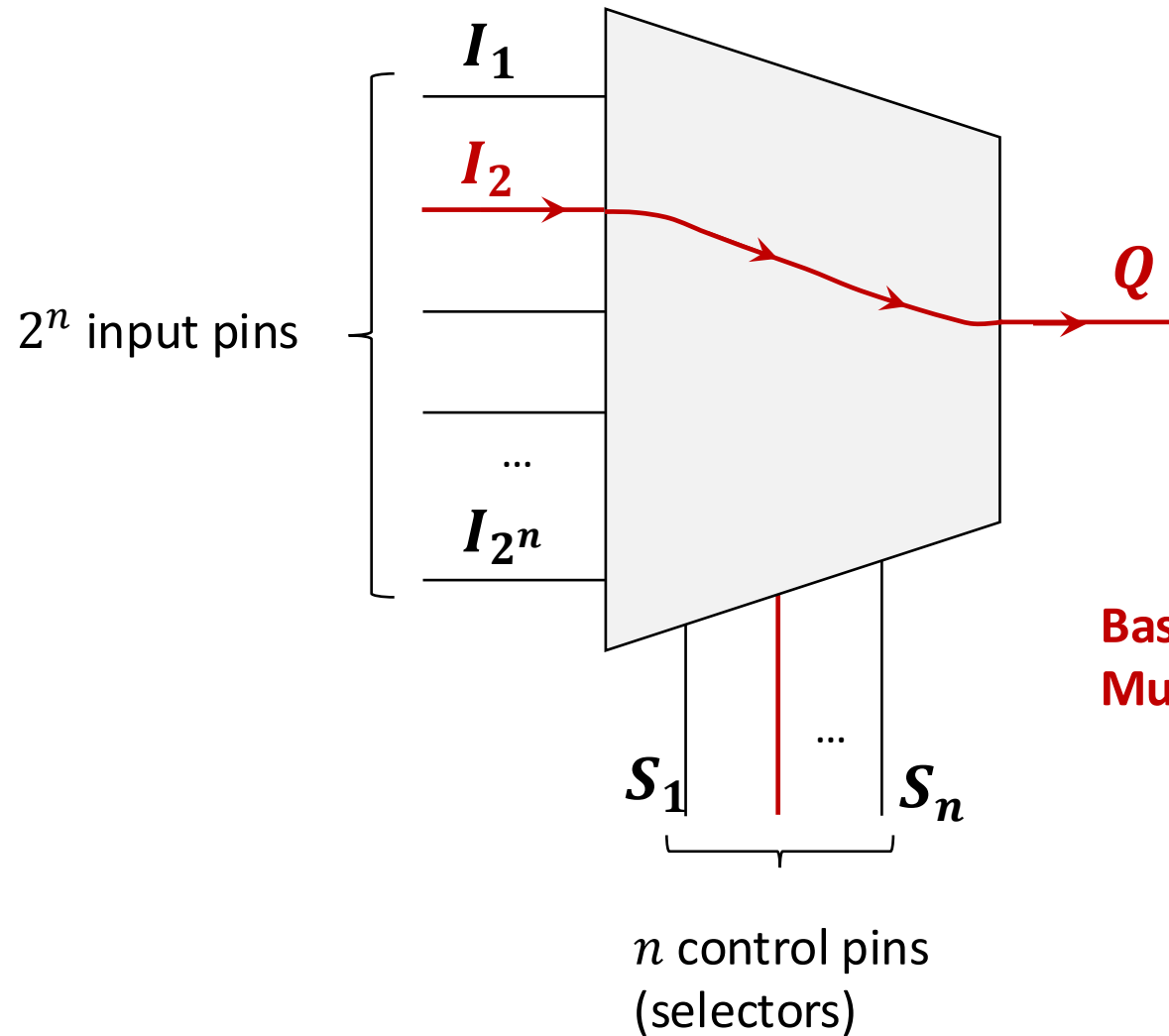
**Based on control pin signals,
Multiplexer sets its output to one of its inputs**

Multiplexer/Selector: Combinational Logic Circuit



**Based on control pin signals,
Multiplexer sets its output to one of its inputs**

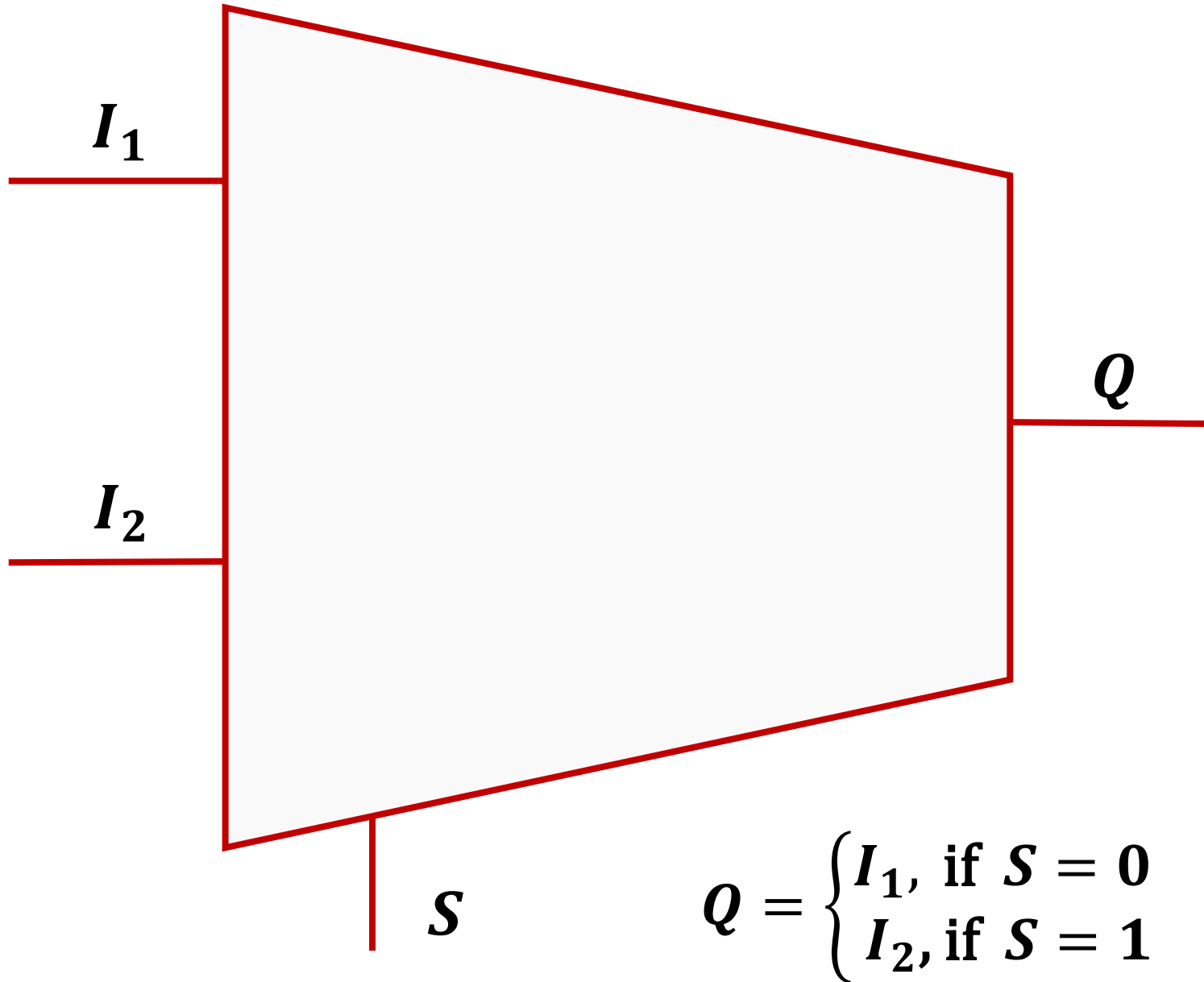
Multiplexer/Selector: Combinational Logic Circuit



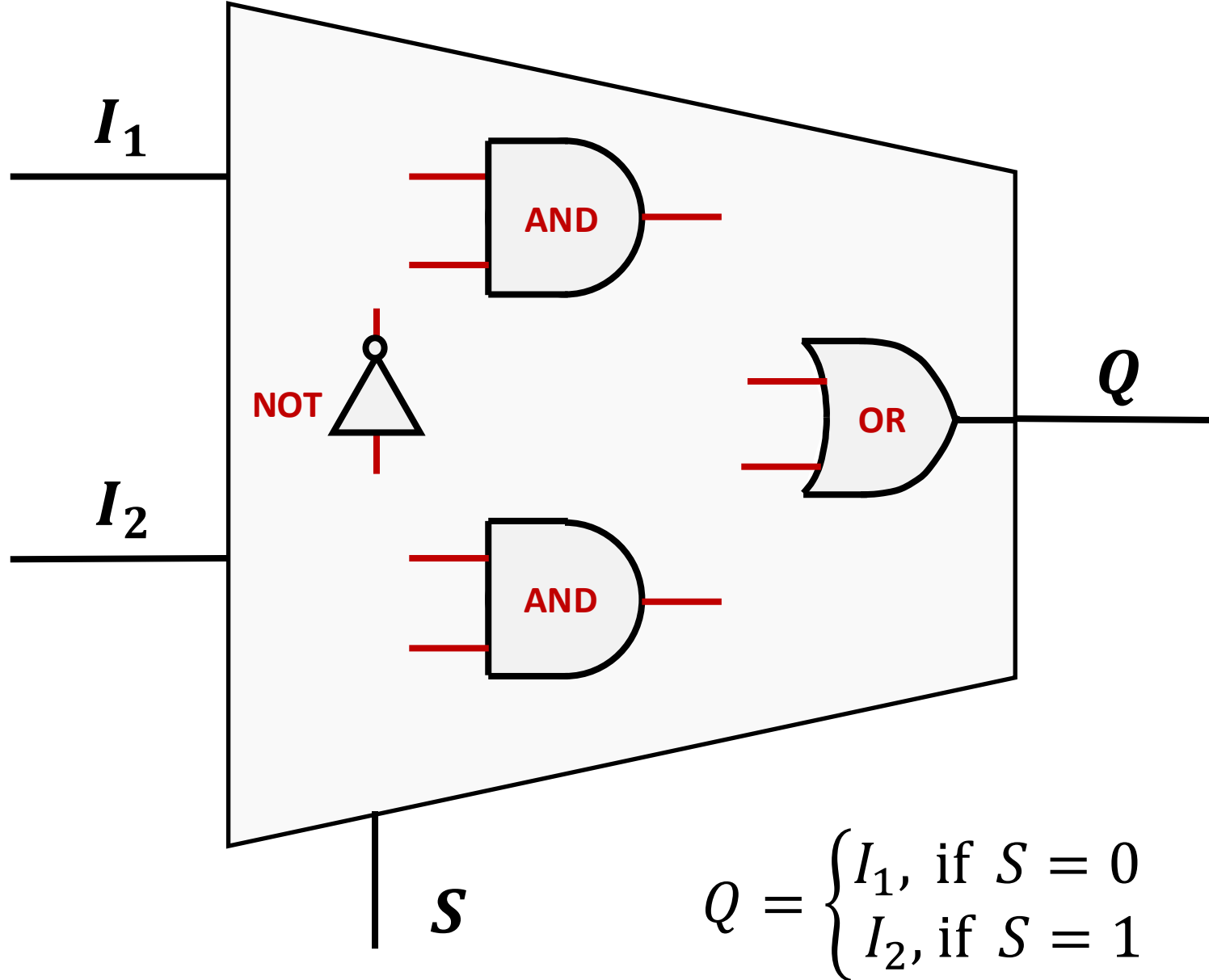
**Based on control pin signals,
Multiplexor sets its output to one of its inputs**

2-to-1 Multiplexer Implementation

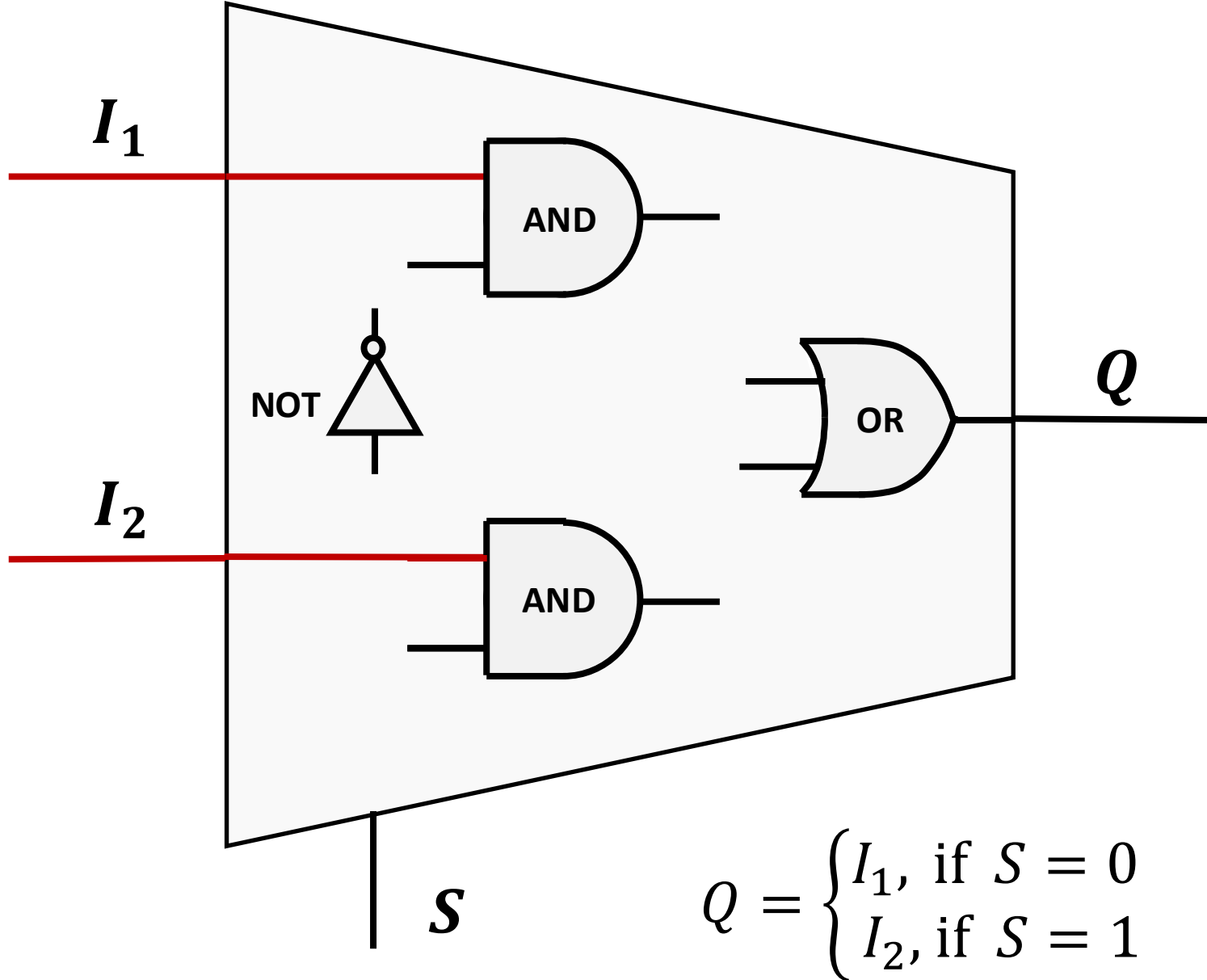
2-to-1 Multiplexer Implementation



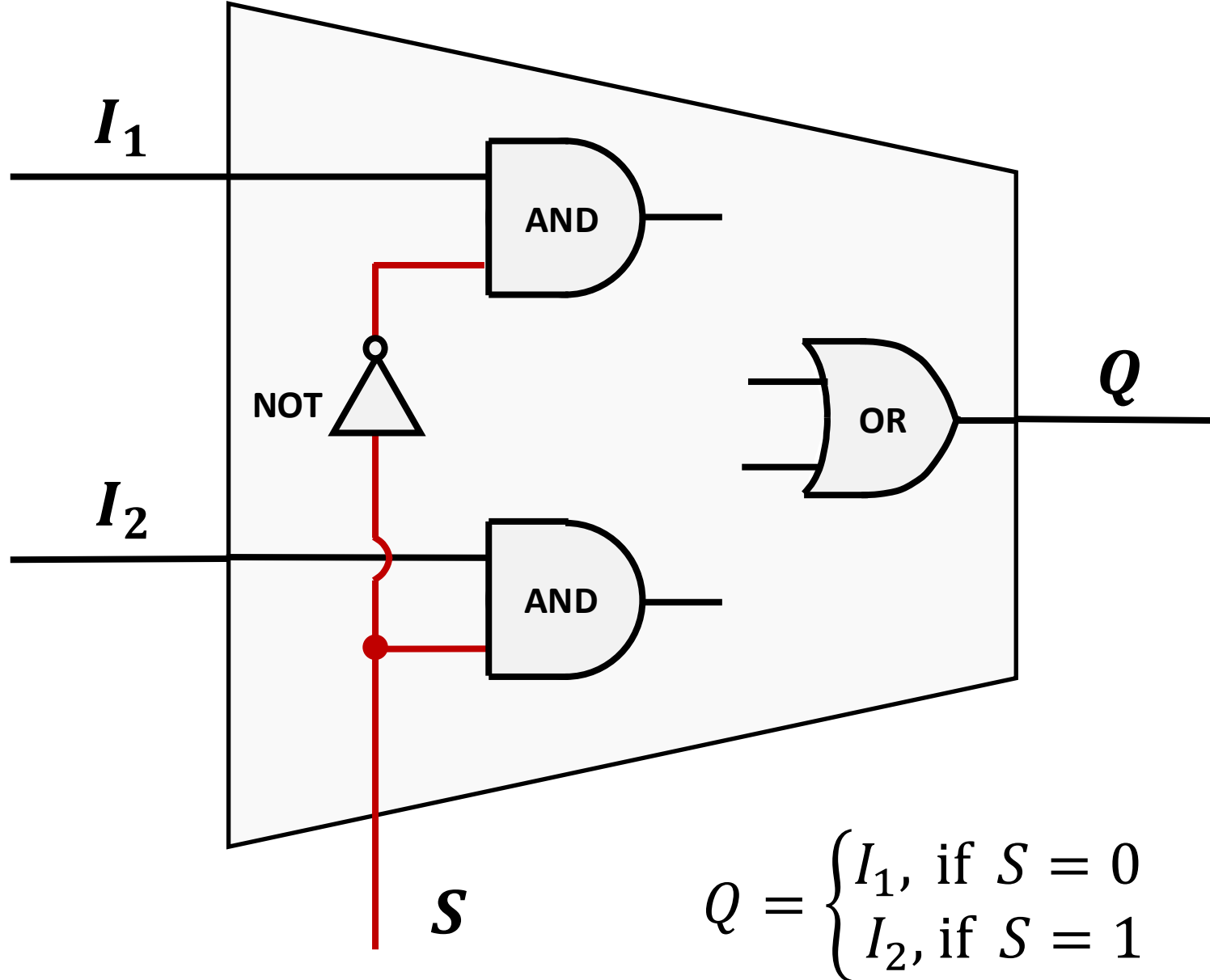
2-to-1 Multiplexer Implementation



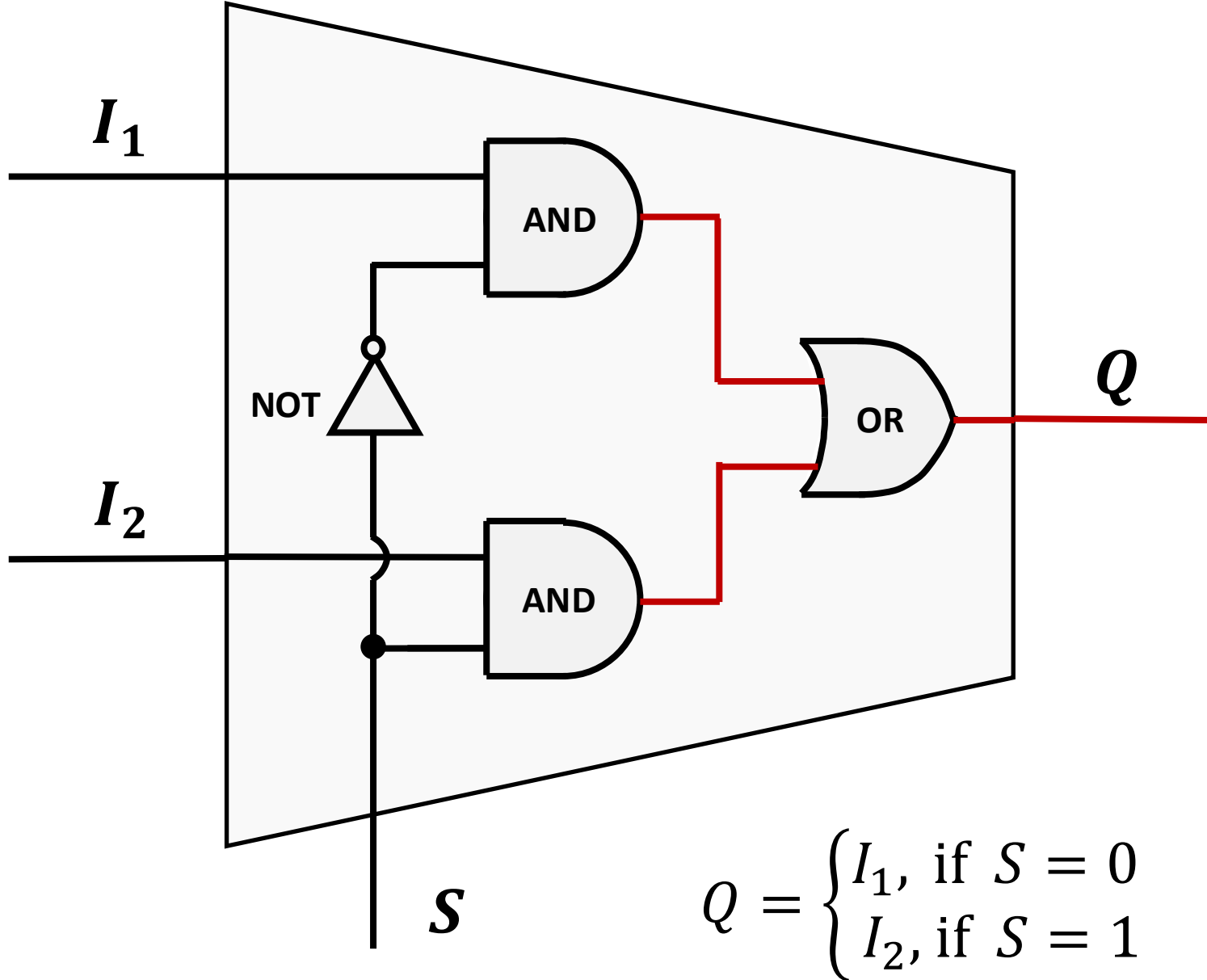
2-to-1 Multiplexer Implementation



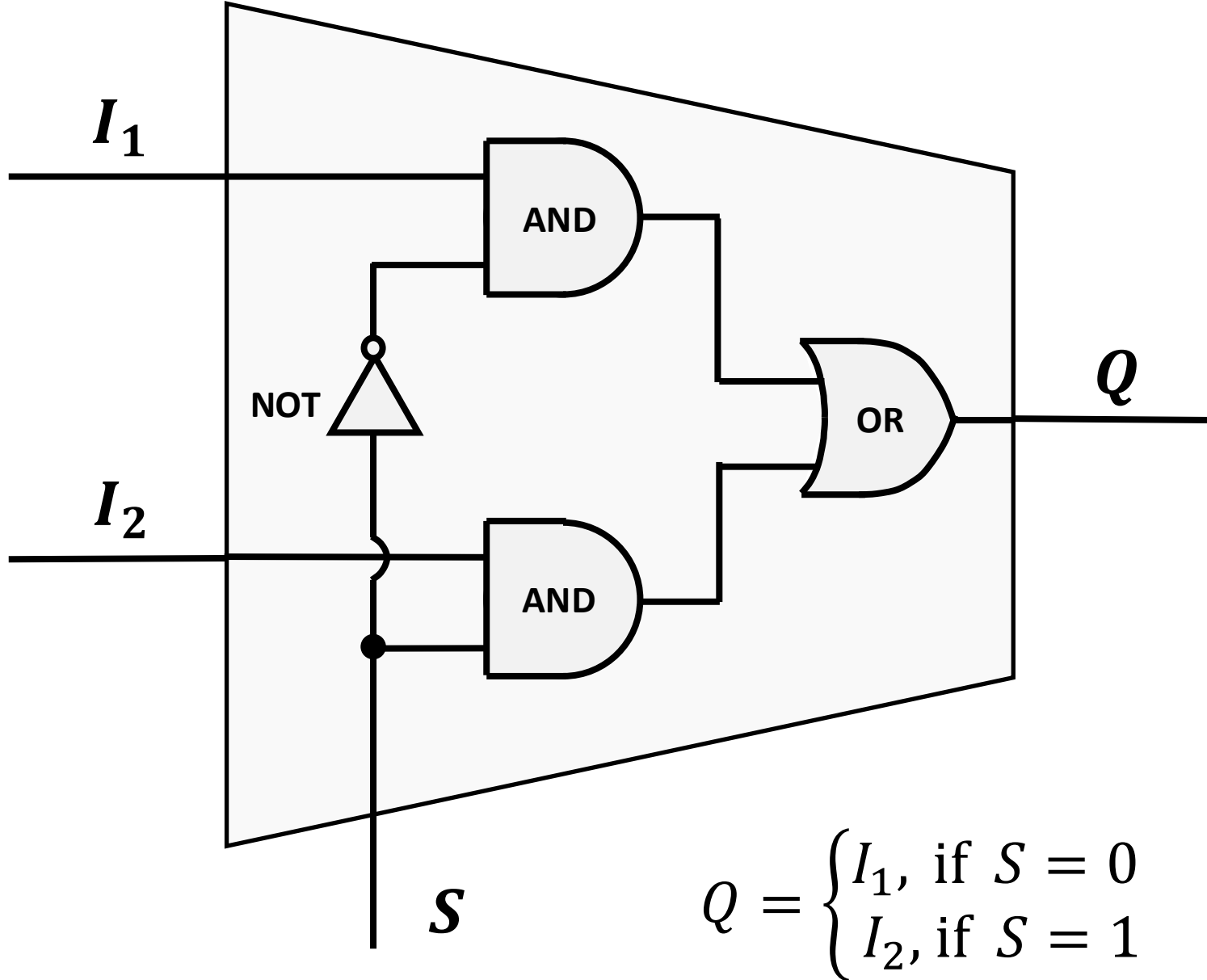
2-to-1 Multiplexer Implementation



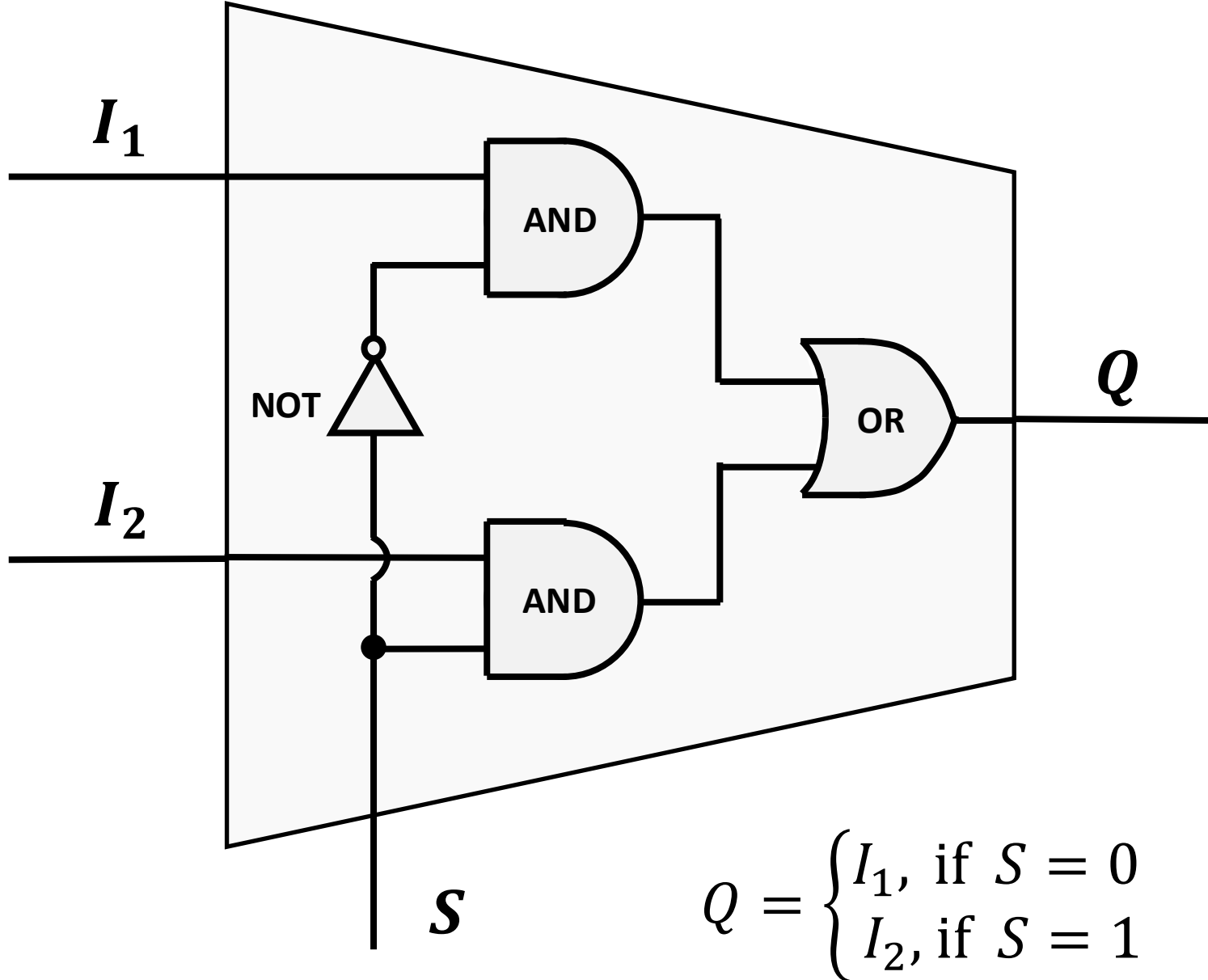
2-to-1 Multiplexer Implementation



2-to-1 Multiplexer Implementation



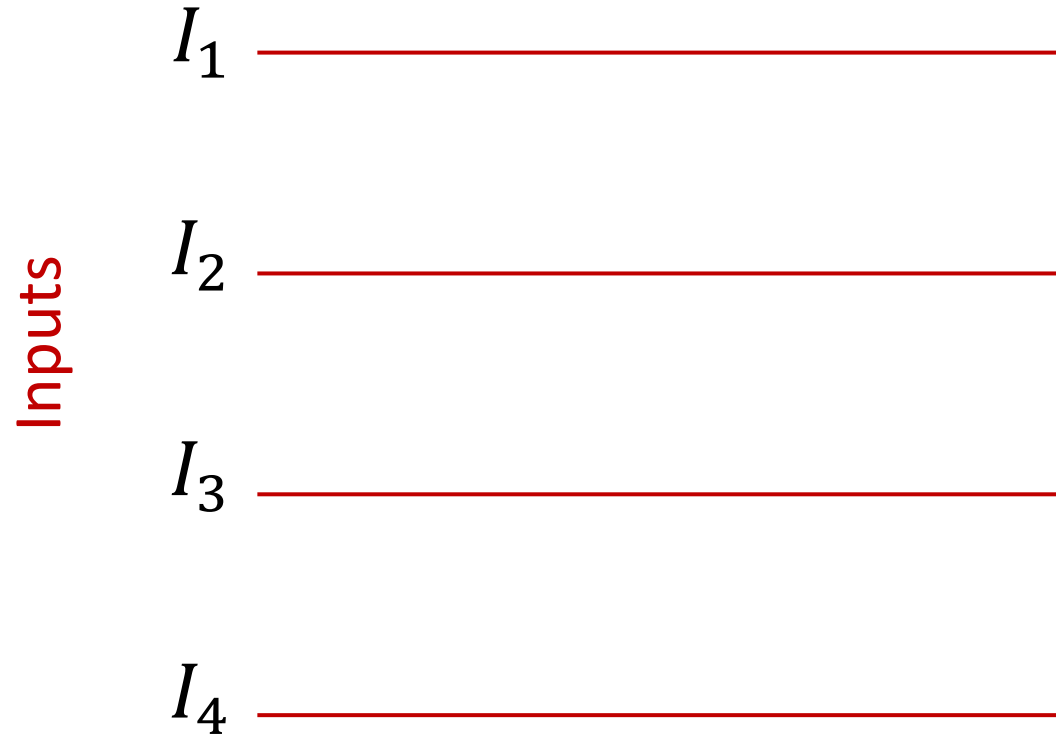
2-to-1 Multiplexer Implementation



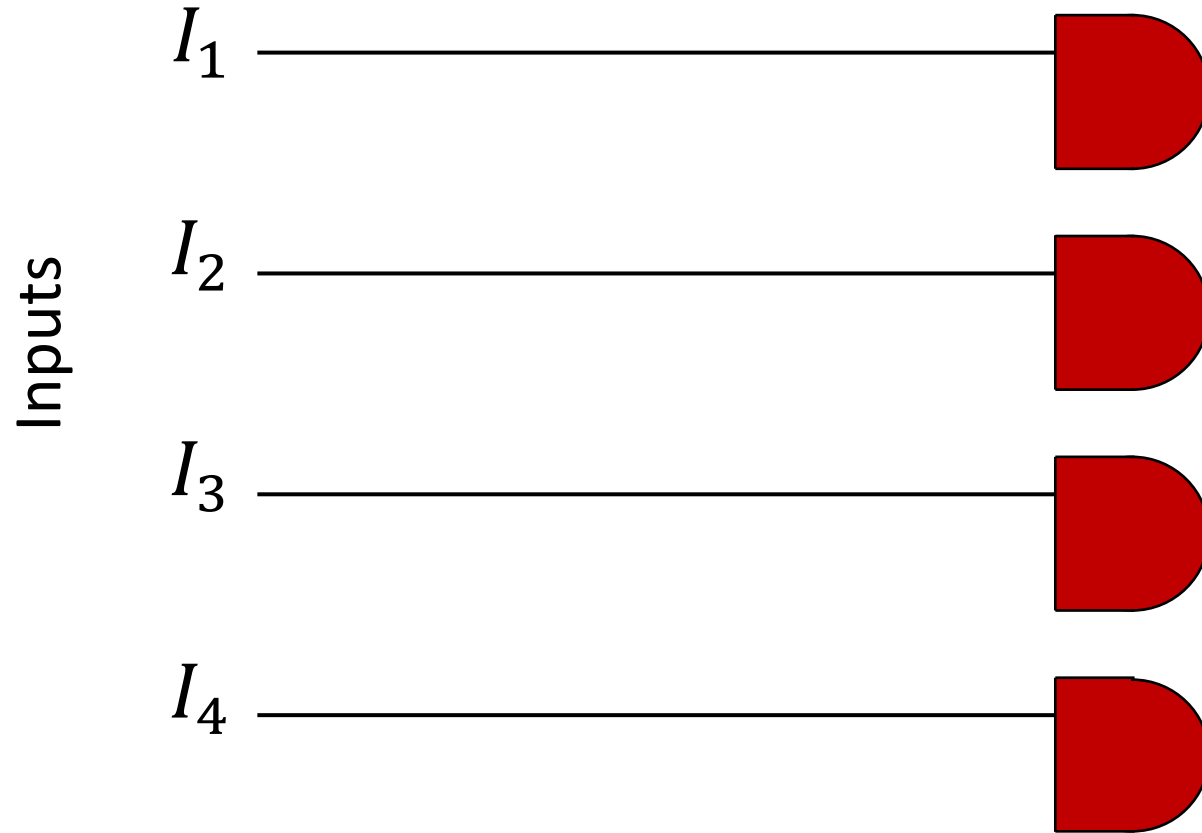
Verification of correctness:

I_1	I_2	S	Q
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	0
0	1	1	1
1	1	1	1

4-to-1 Multiplexer Implementation

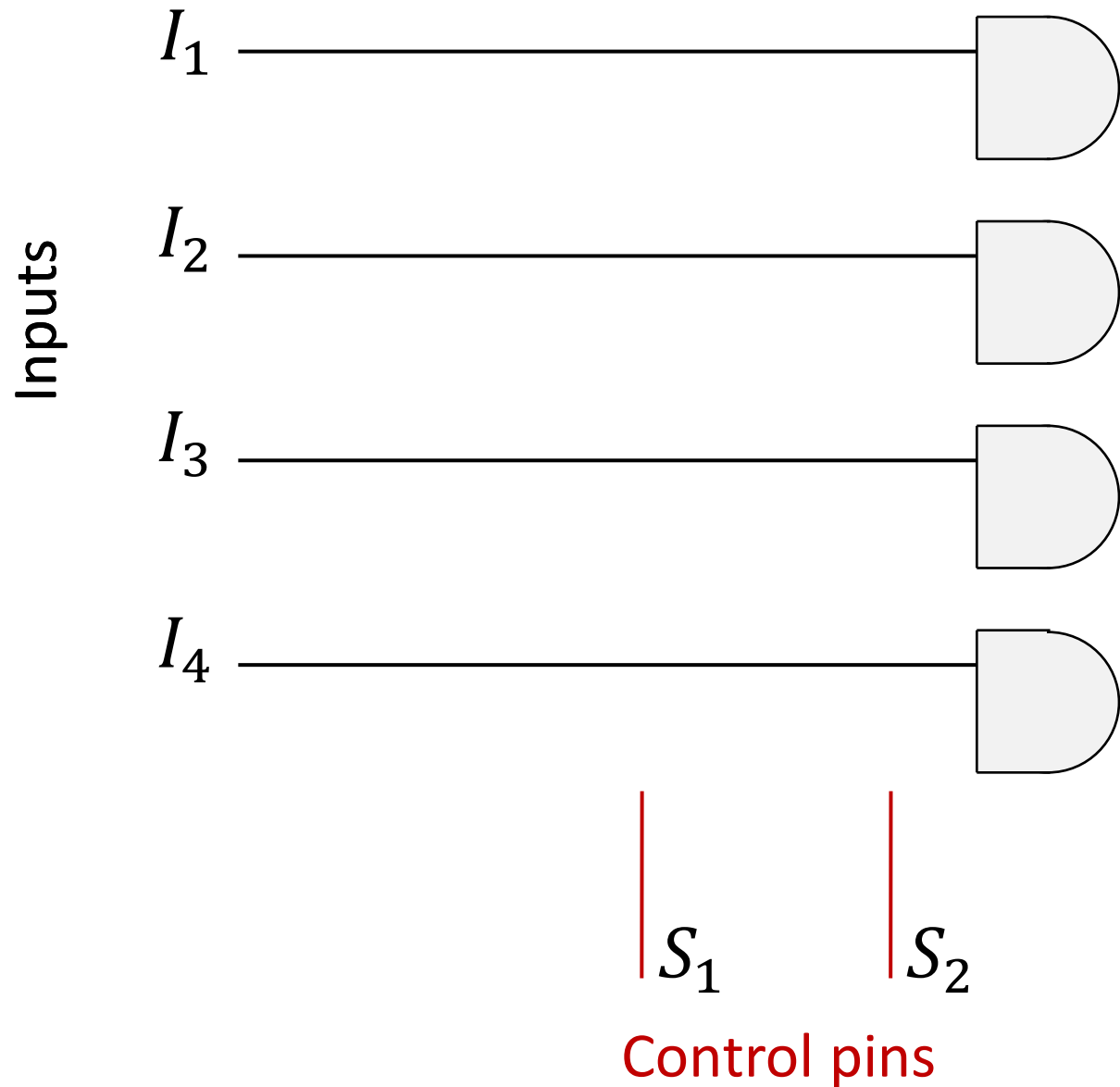


4-to-1 Multiplexer Implementation

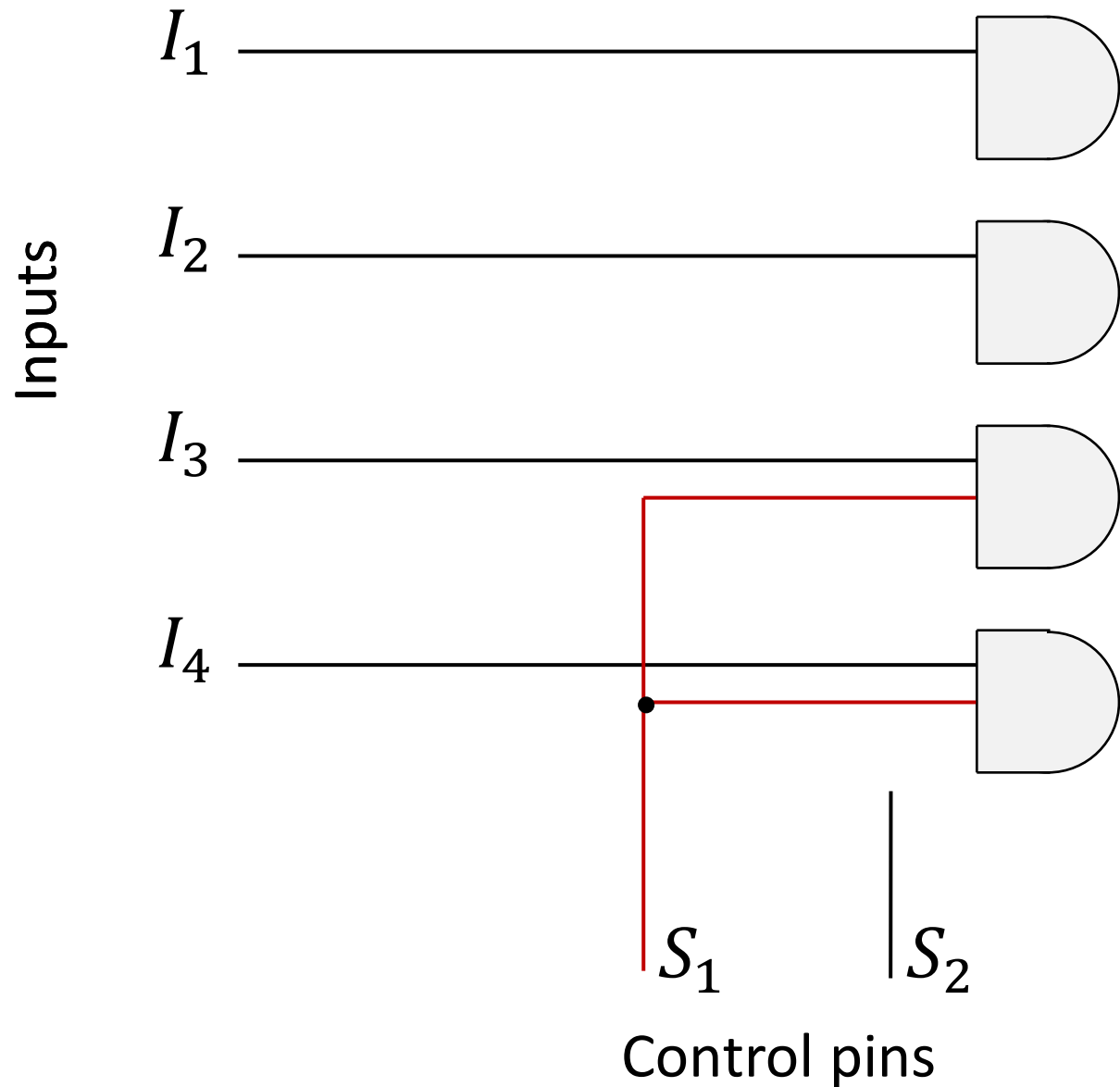


Each input pin is connected to
an individual AND logic gate

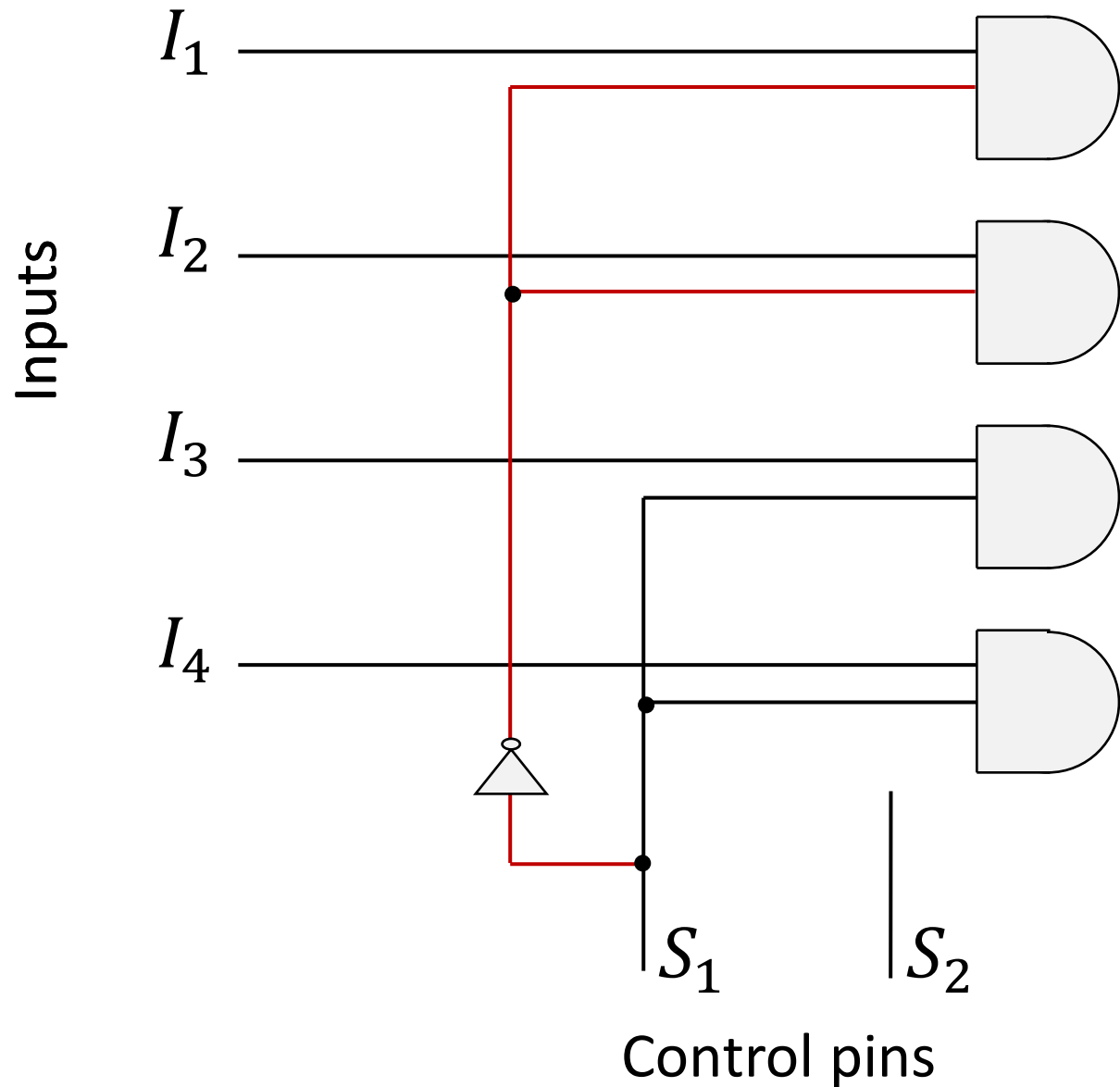
4-to-1 Multiplexer Implementation



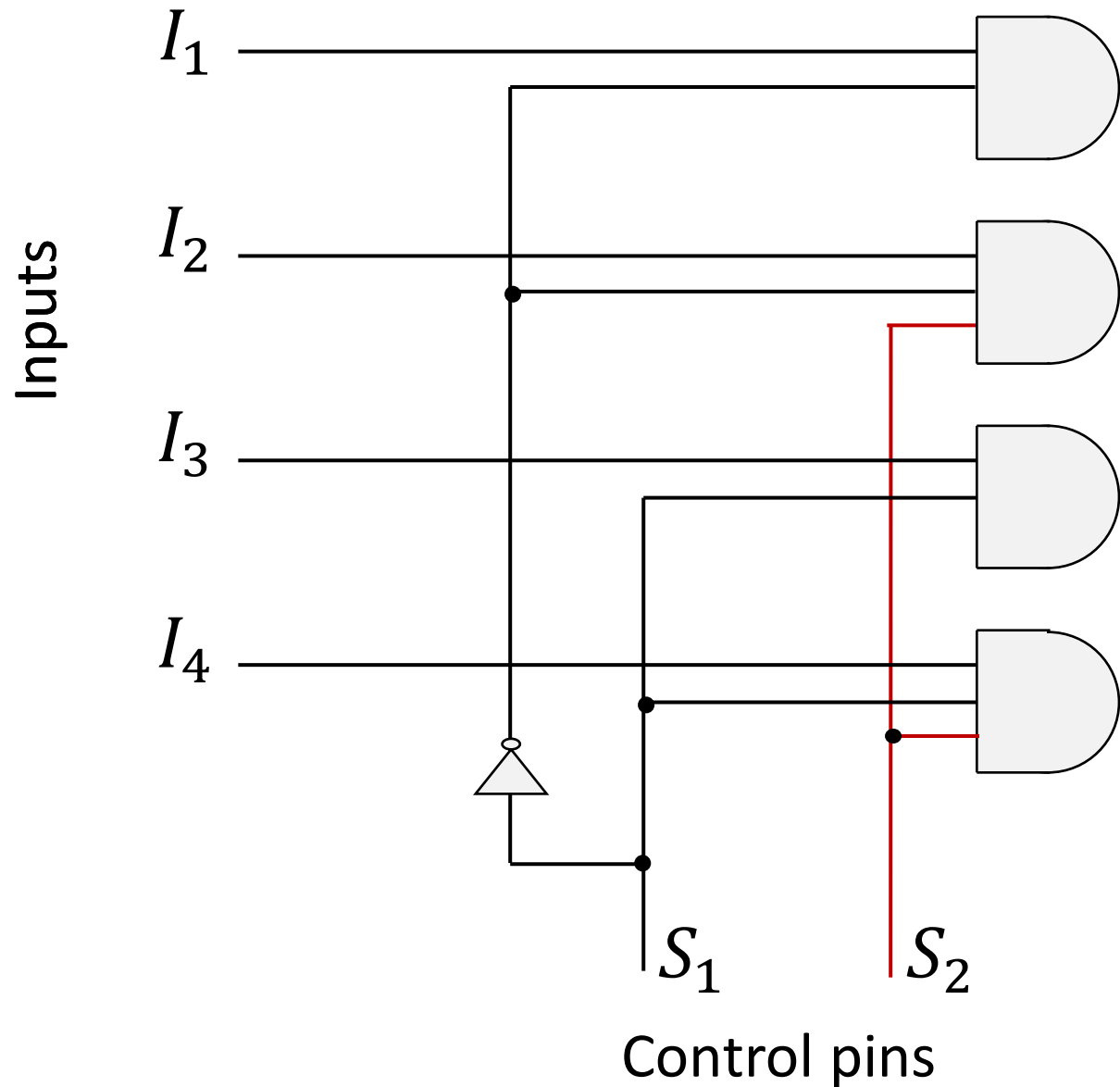
4-to-1 Multiplexer Implementation



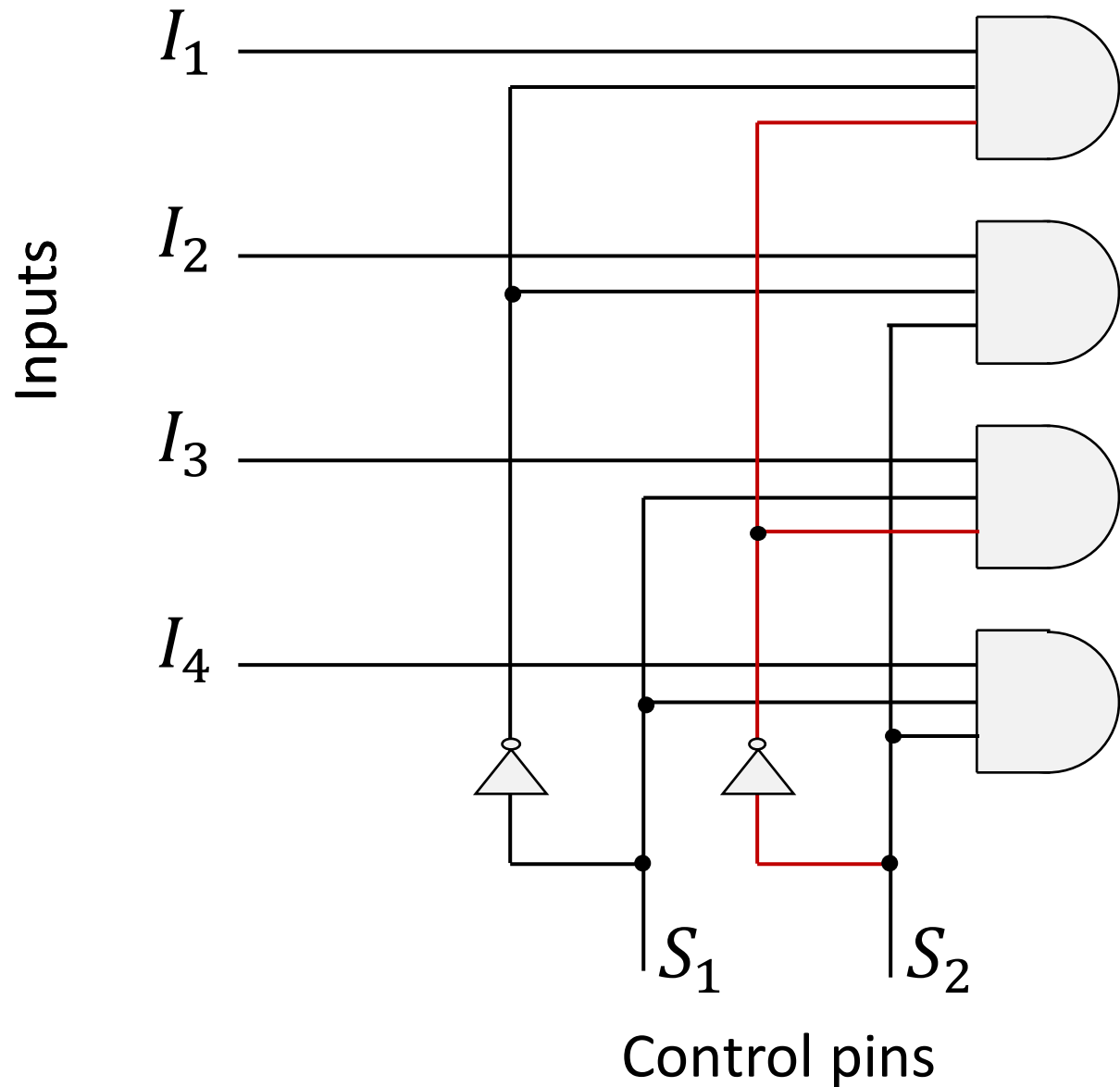
4-to-1 Multiplexer Implementation



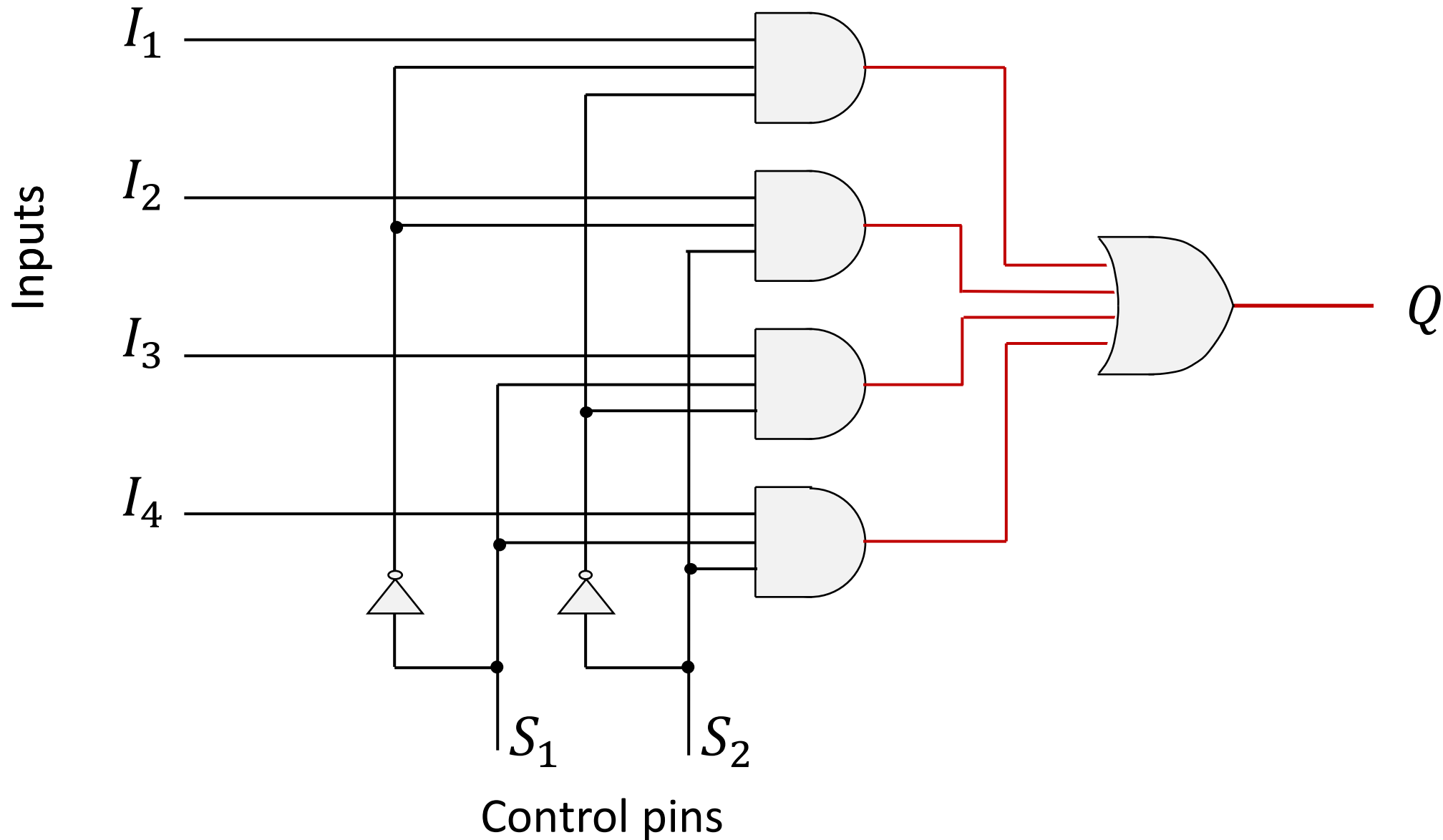
4-to-1 Multiplexer Implementation



4-to-1 Multiplexer Implementation

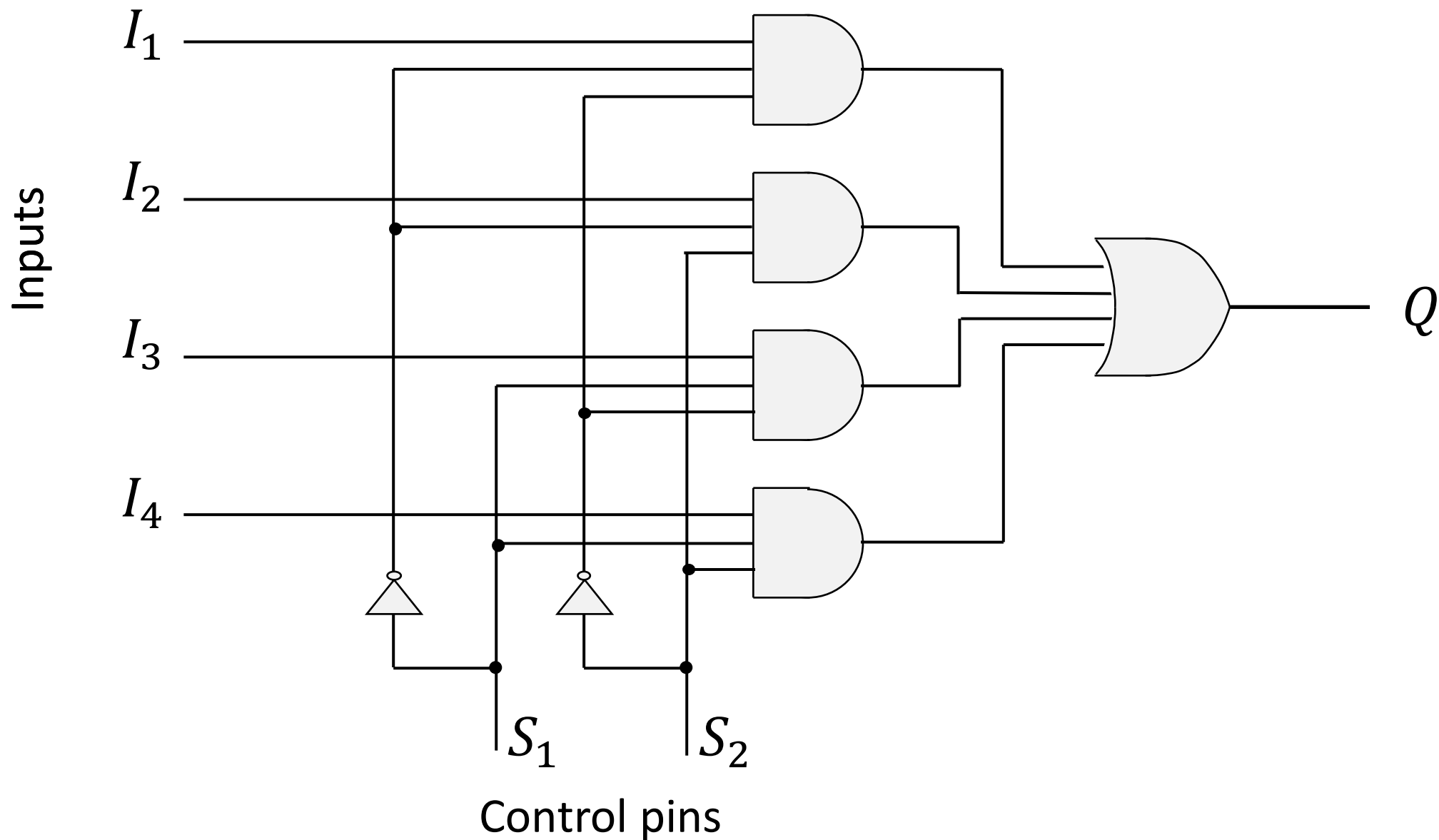


4-to-1 Multiplexer Implementation

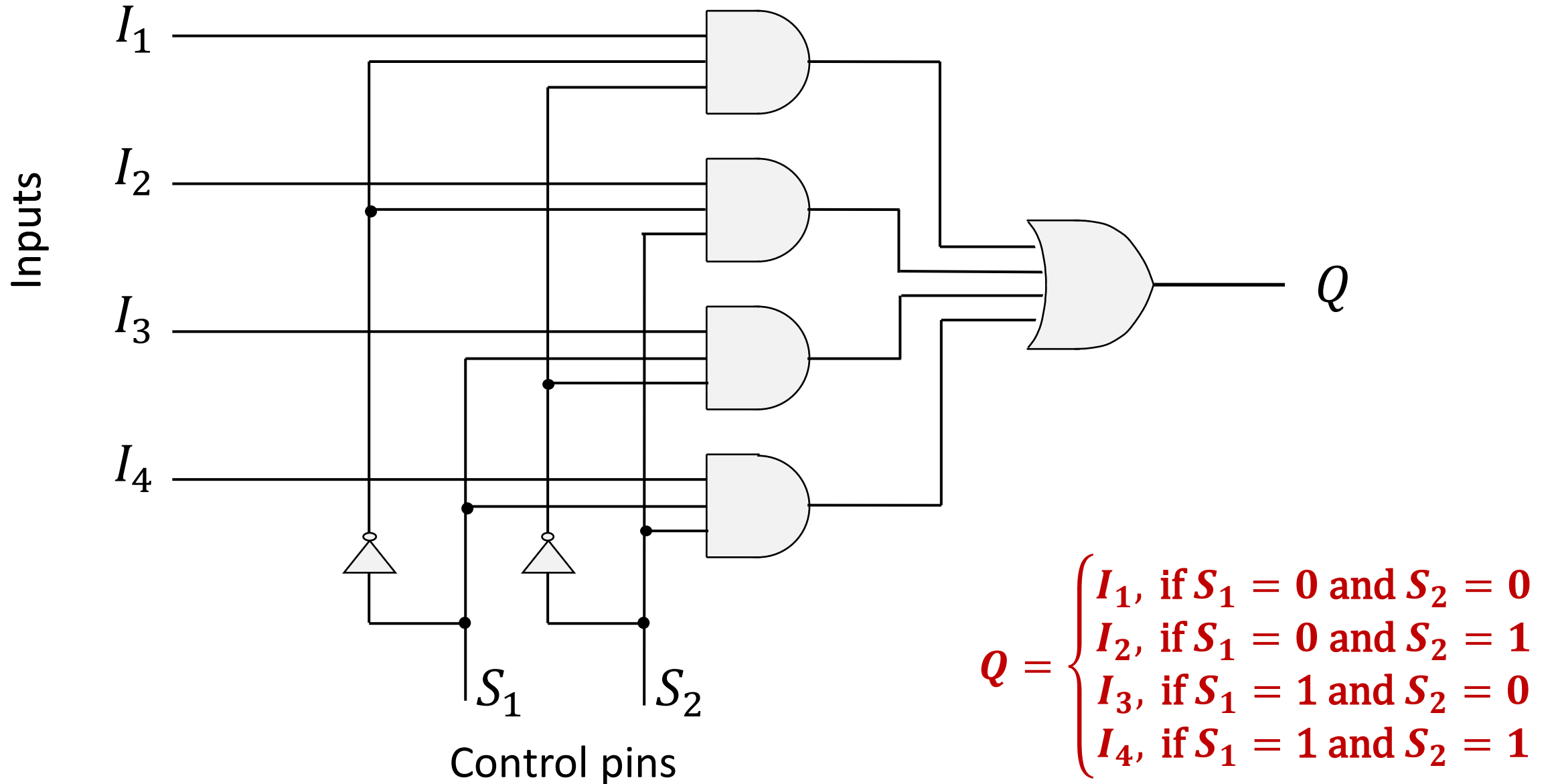


4-to-1 Multiplexer Implementation

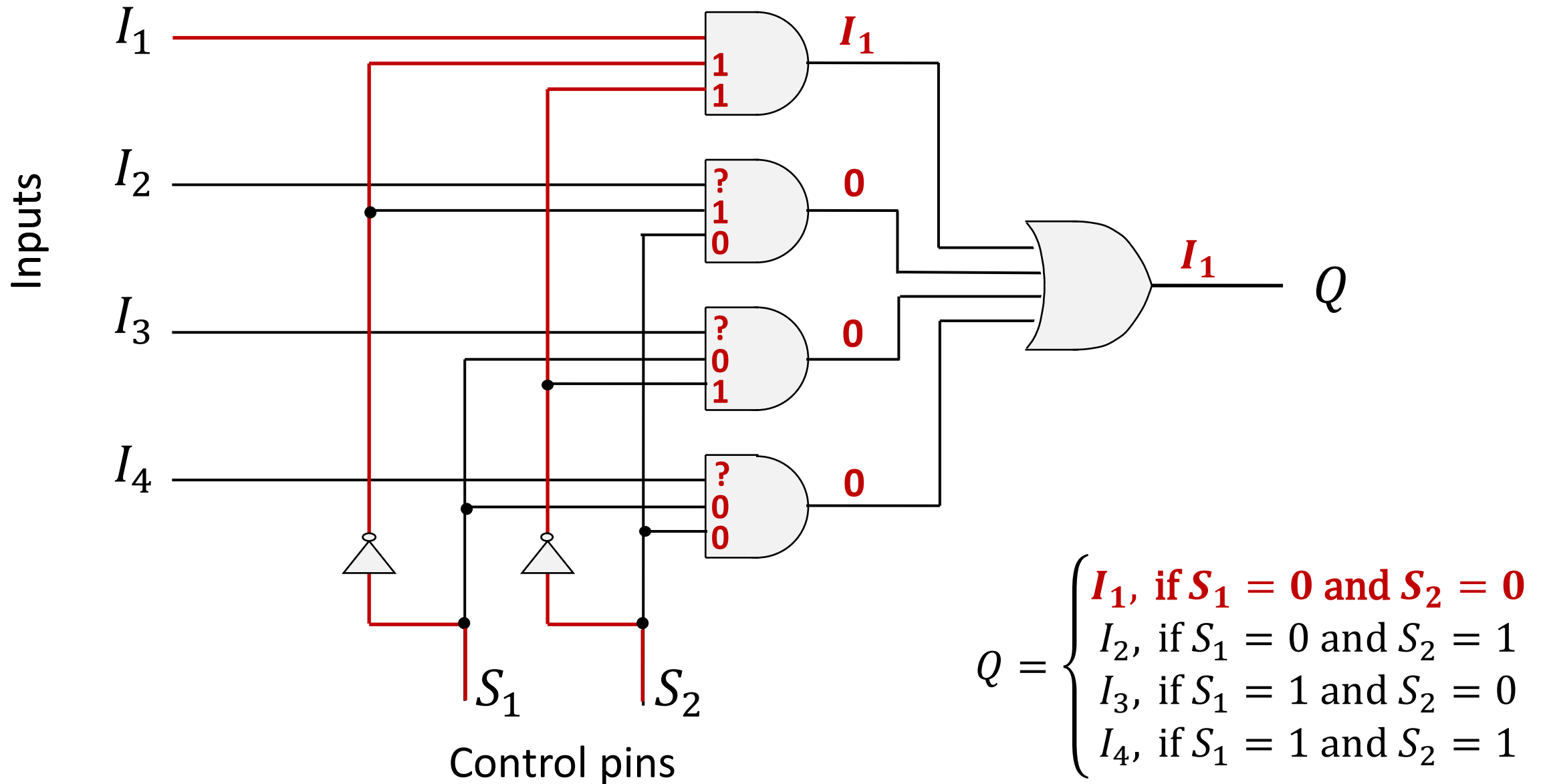
Based on S_1 and S_2 signals,
output Q is set to one of $I_1, \dots, I_4$ input pins



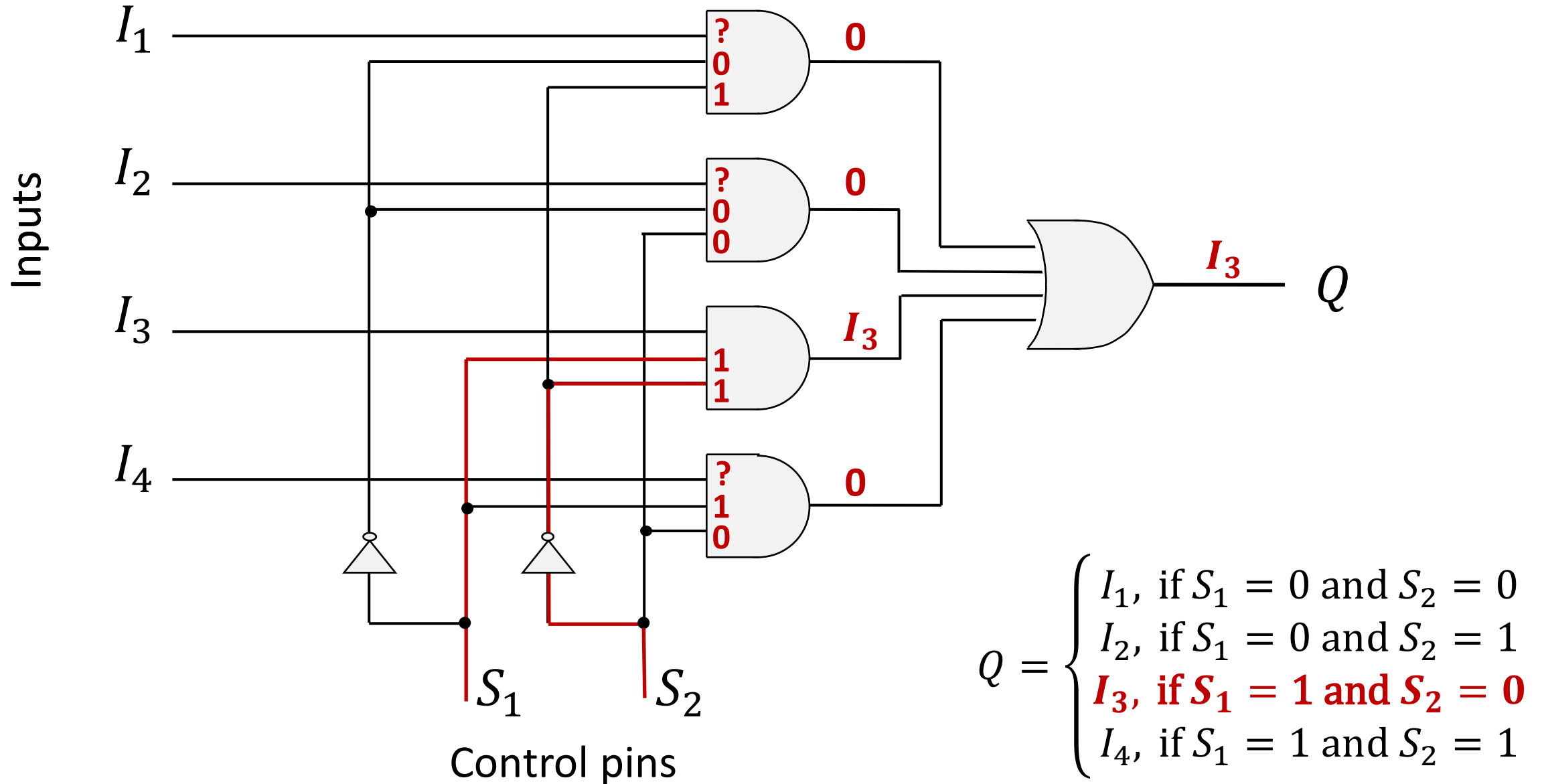
4-to-1 Multiplexer Implementation



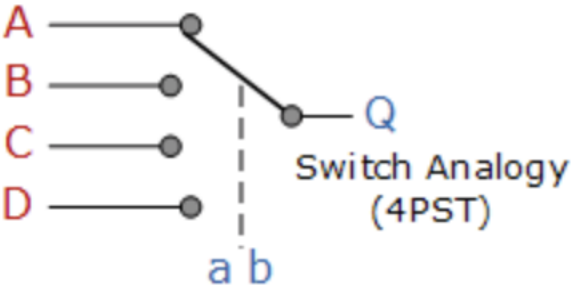
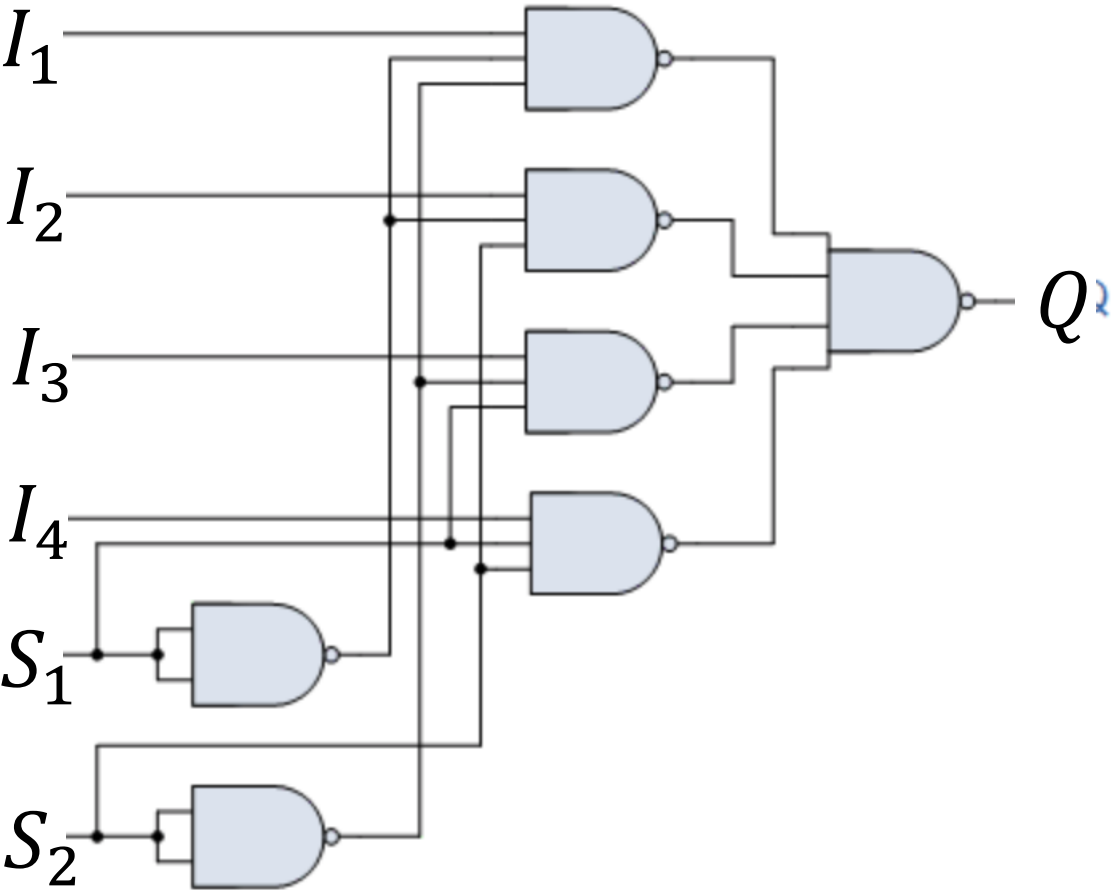
4-to-1 Multiplexer Implementation



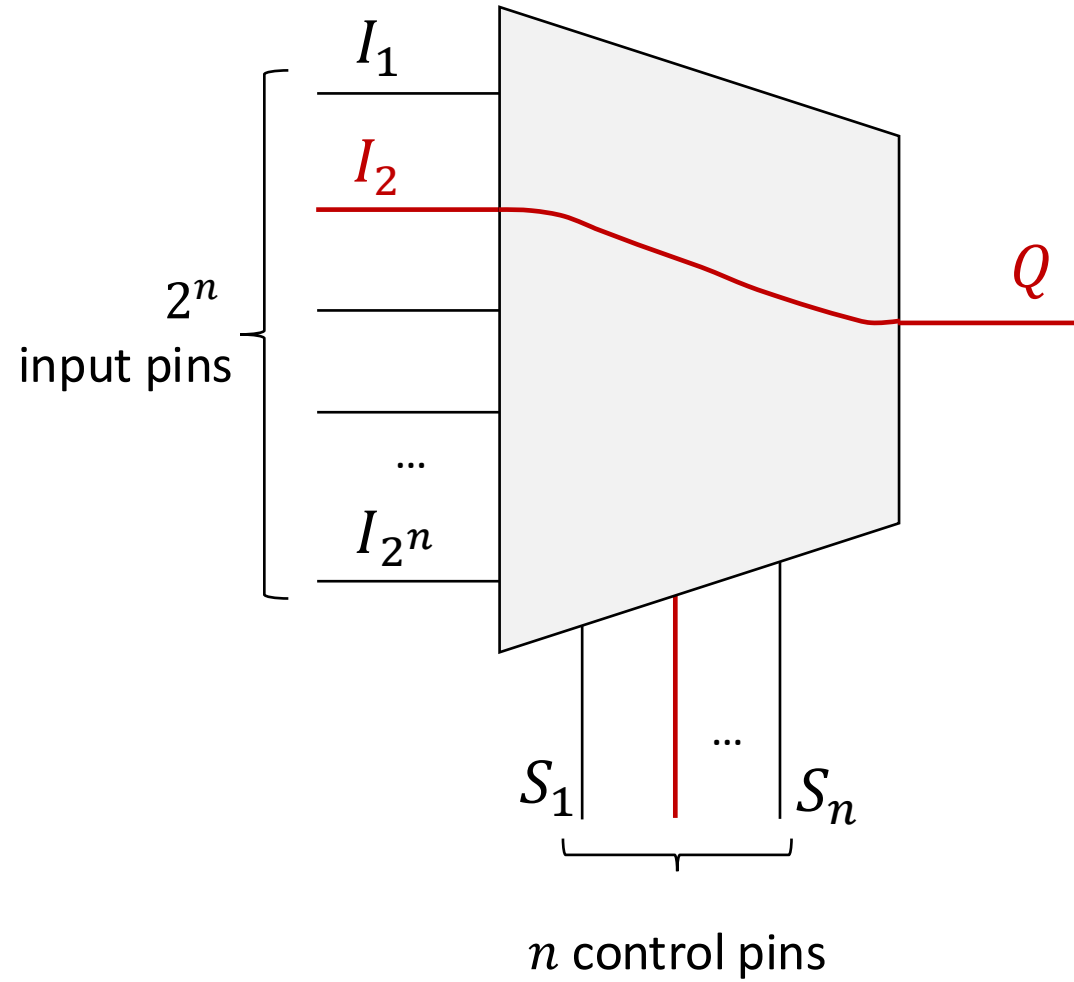
4-to-1 Multiplexer Implementation



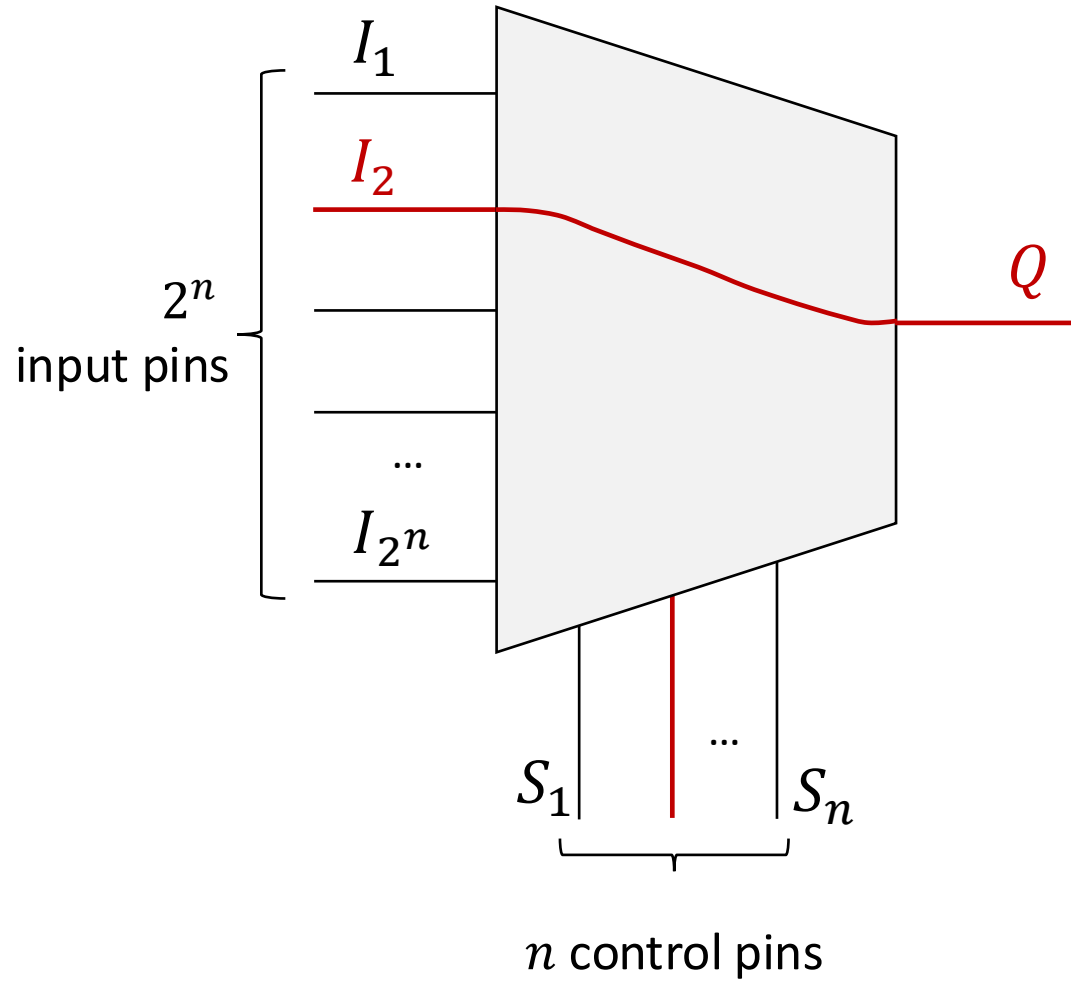
4-to-1 Multiplexer: NAND-based Implementation



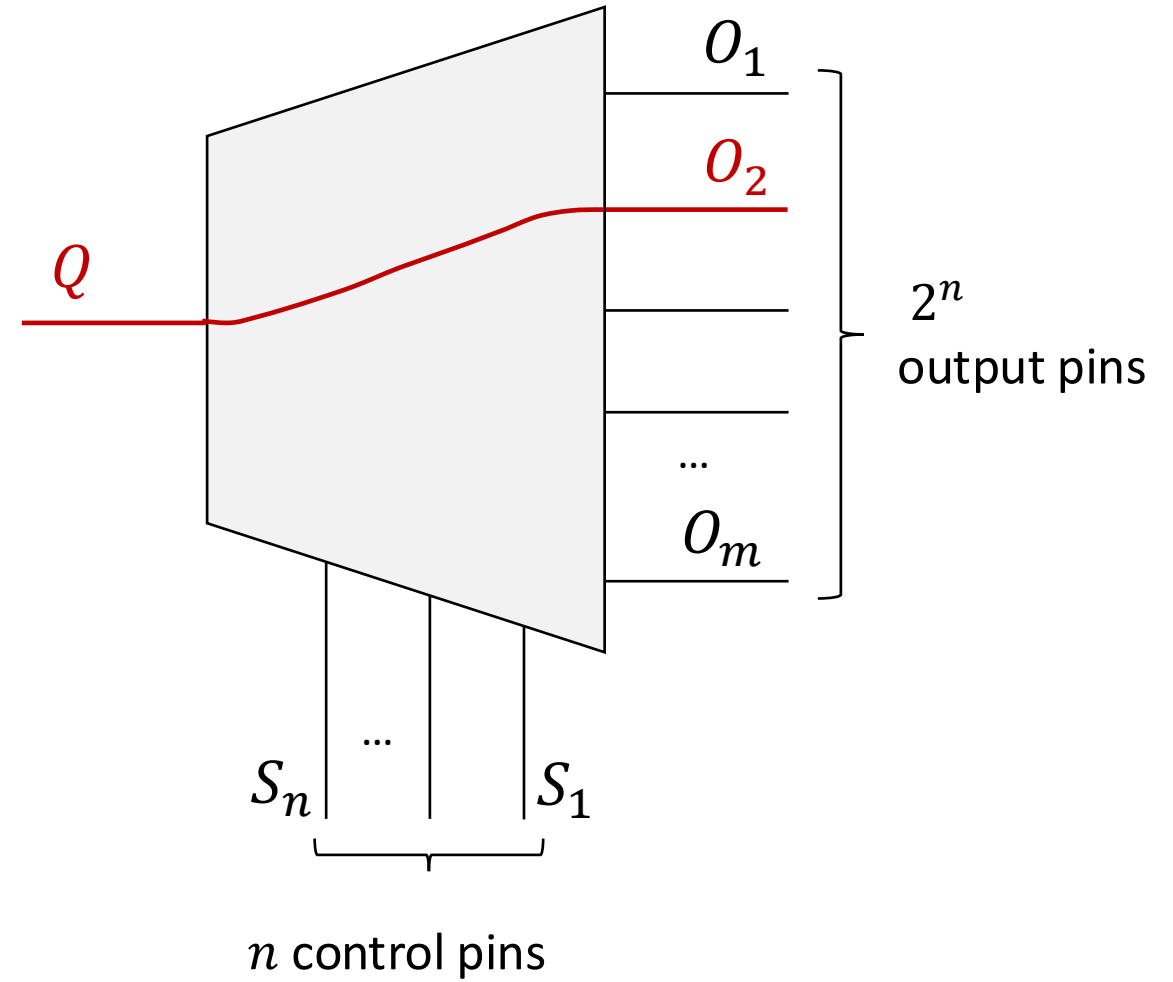
Multiplexer



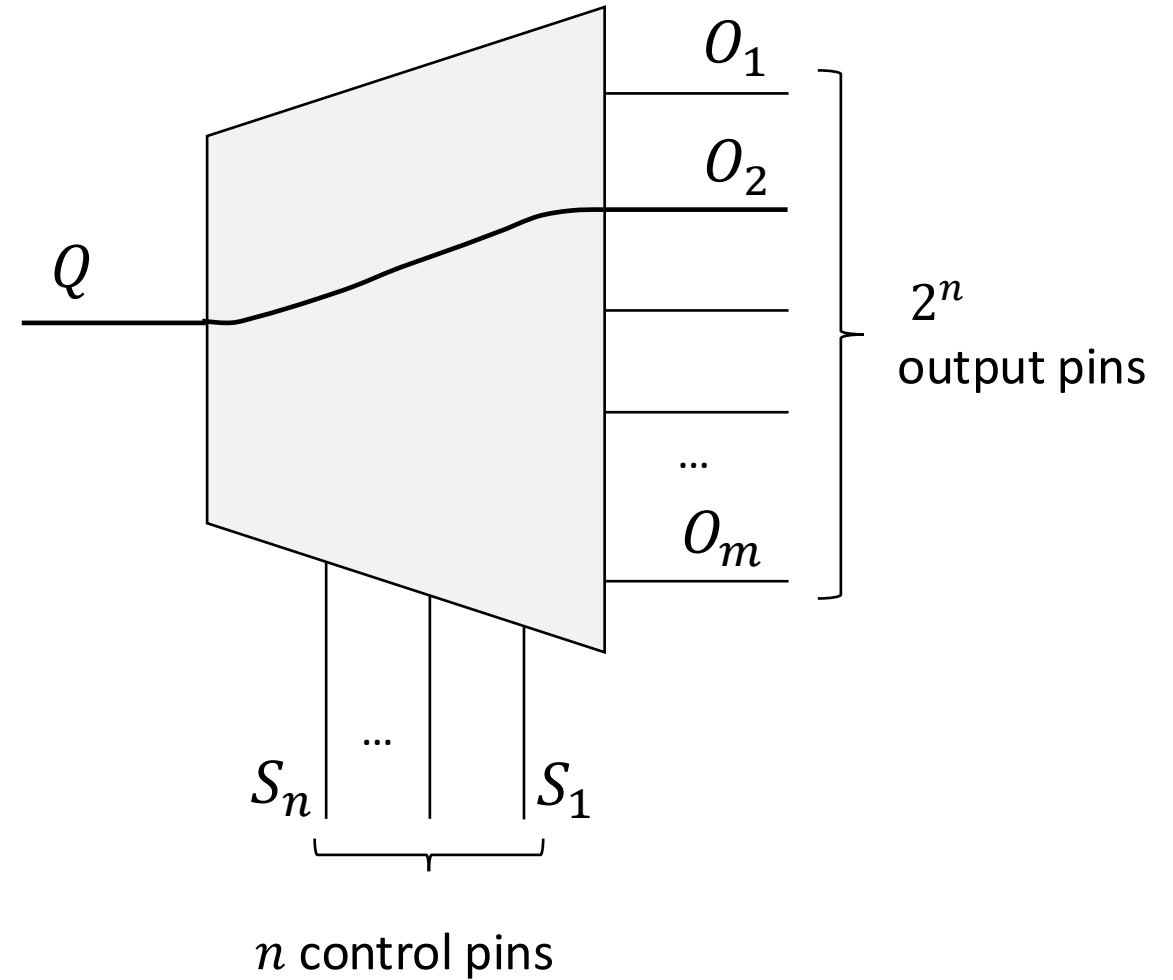
Multiplexer



Demultiplexer

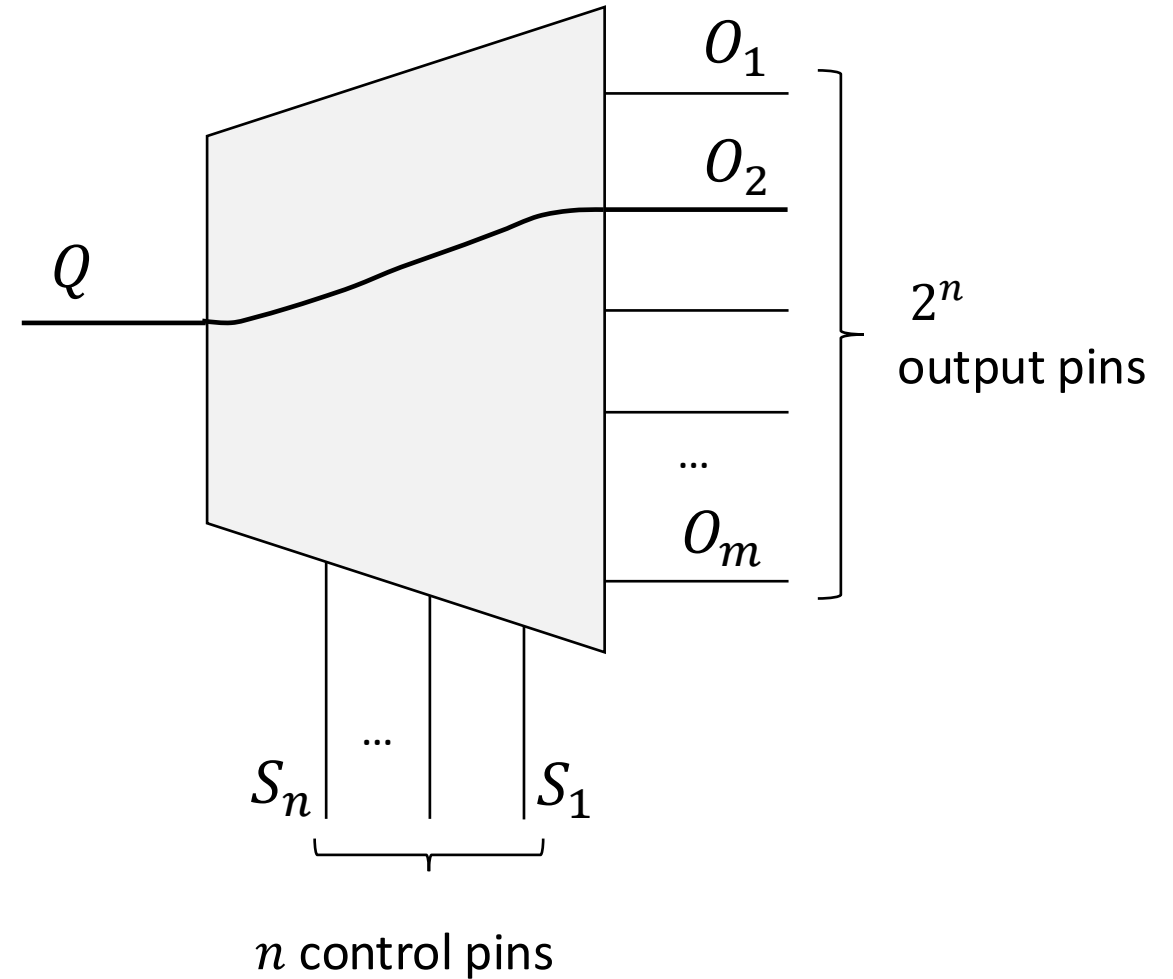


Demultiplexer



**Based on control pin signals,
demultiplexer sends its input to one of its outputs**

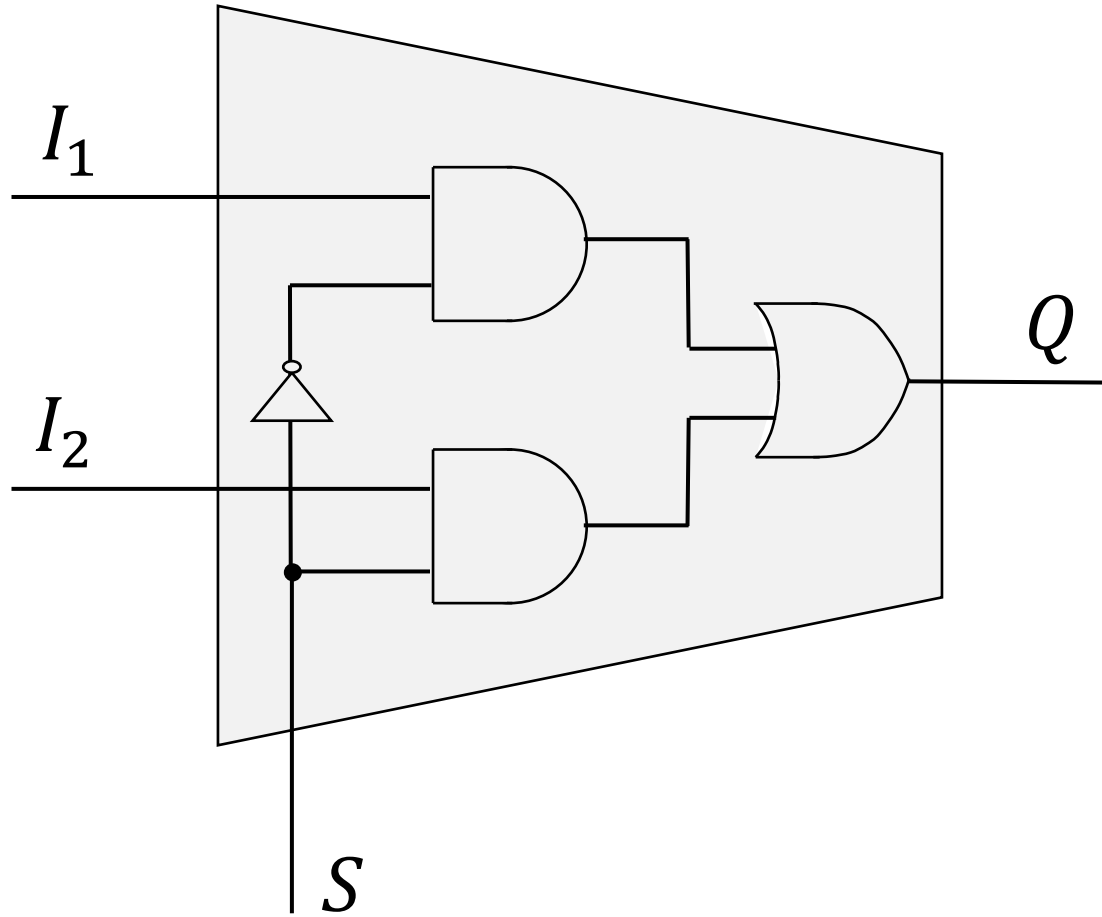
Demultiplexer



Based on control pin signals,
demultiplexor sends its input to one of its outputs

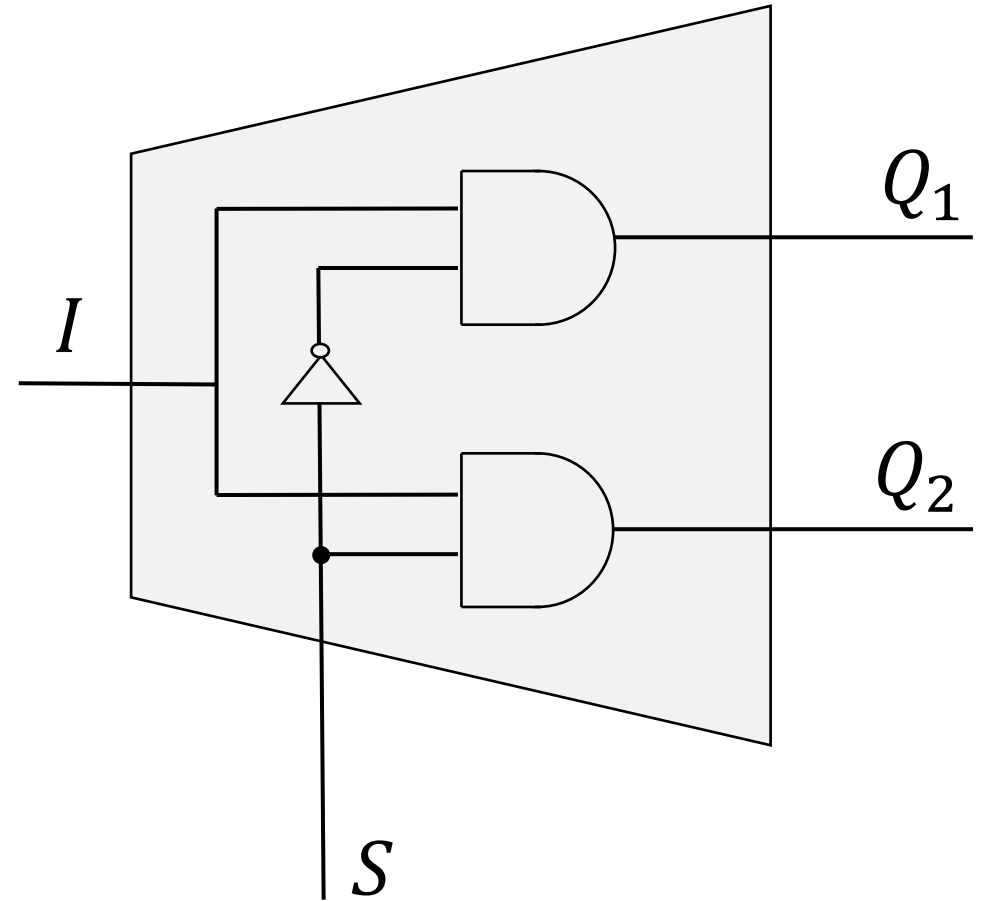
**All other outputs are set to some
default value (e.g. "0")**

2-to-1 Multiplexer



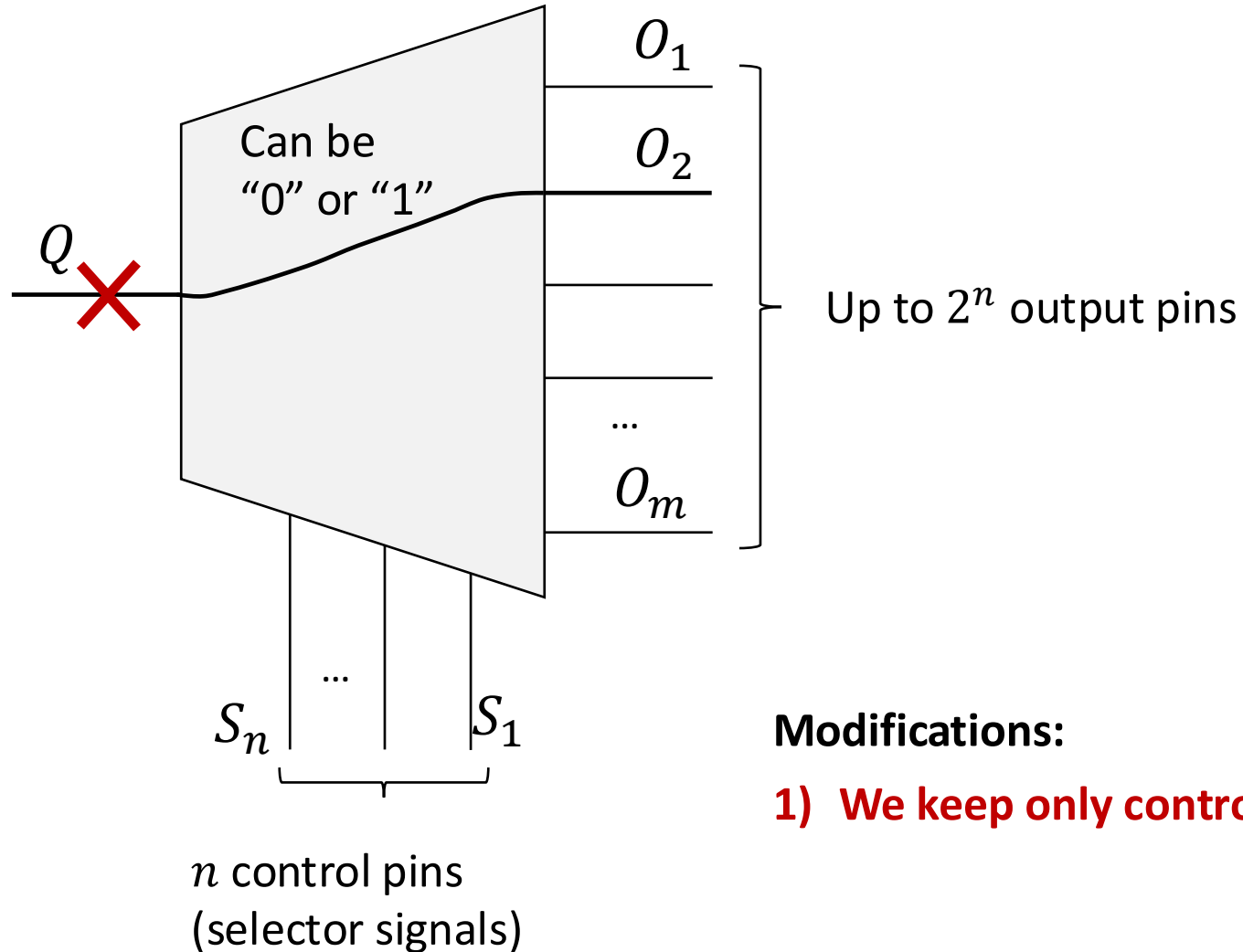
$$Q = \begin{cases} I_1, & \text{if } S = 0 \\ I_2, & \text{if } S = 1 \end{cases}$$

1-to-2 Demultiplexer



$$\begin{aligned} Q_1 &= I \text{ and } Q_2 = 0, & \text{if } S = 0 \\ Q_1 &= 0 \text{ and } Q_2 = I, & \text{if } S = 1 \end{aligned}$$

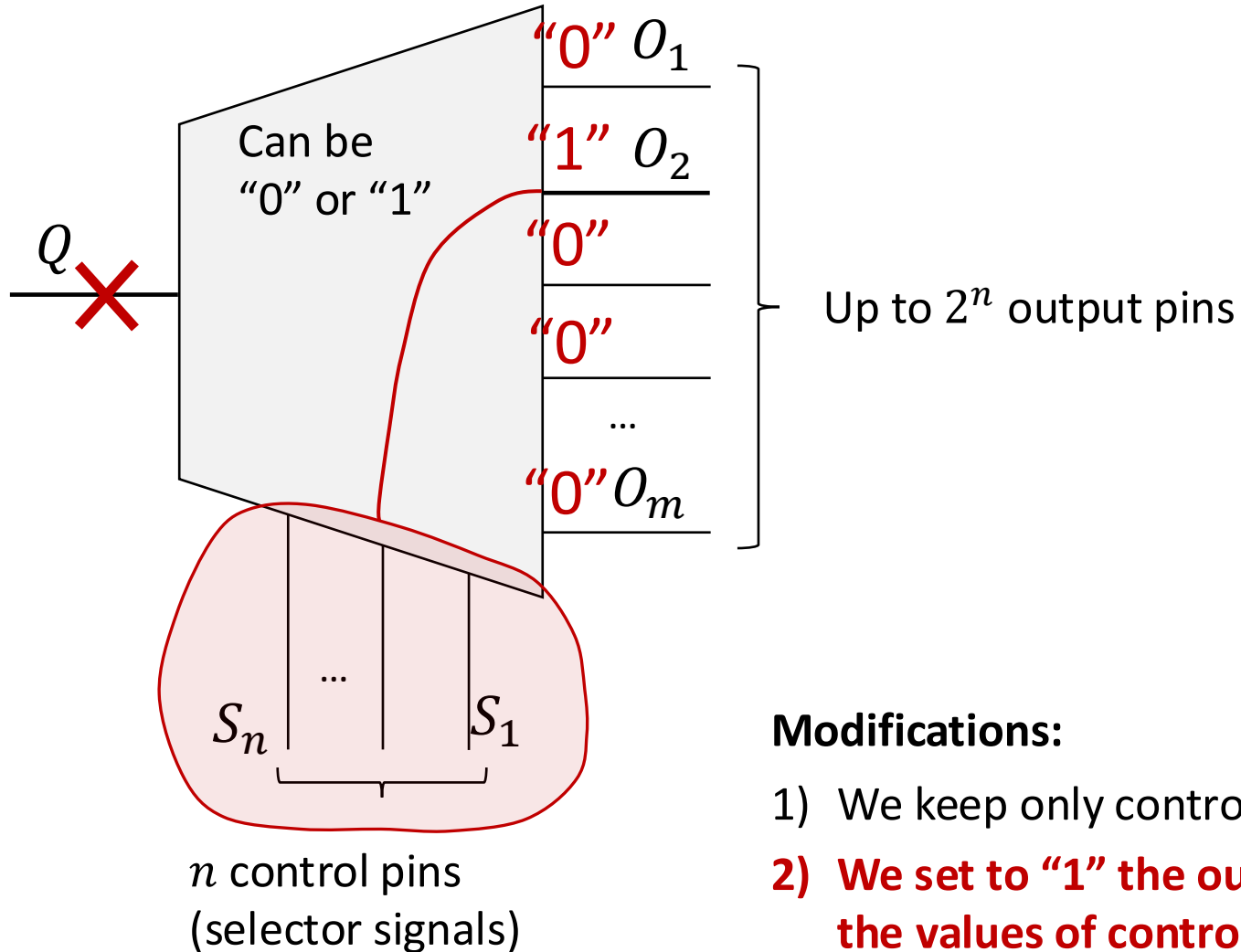
Demultiplexer: Transformation



Modifications:

- 1) We keep only control pins, and remove input pin Q ;

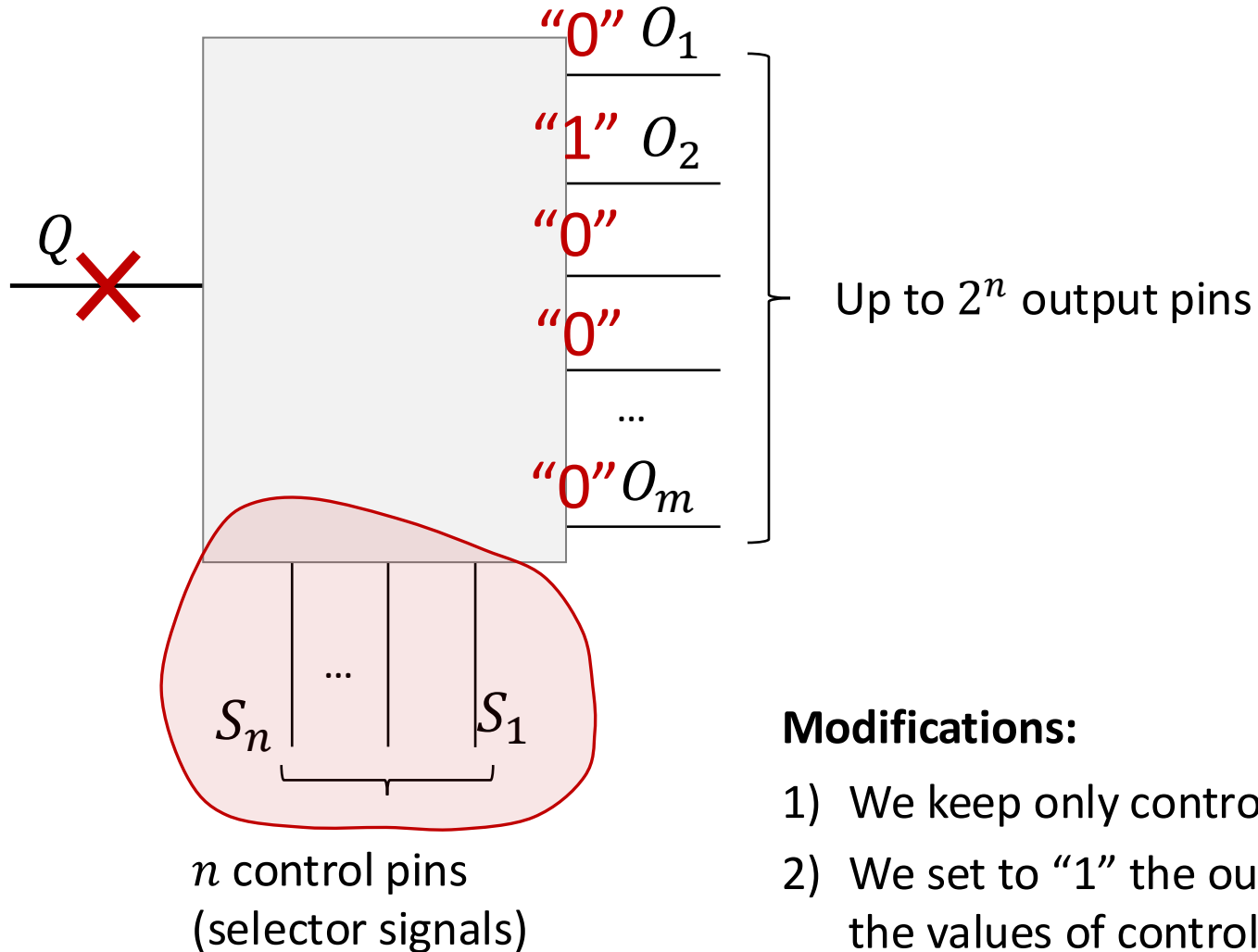
Demultiplexer: Transformation



Modifications:

- 1) We keep only control pins, and remove input pin Q ;
- 2) We set to "1" the output pin, which corresponds to the values of control pins;

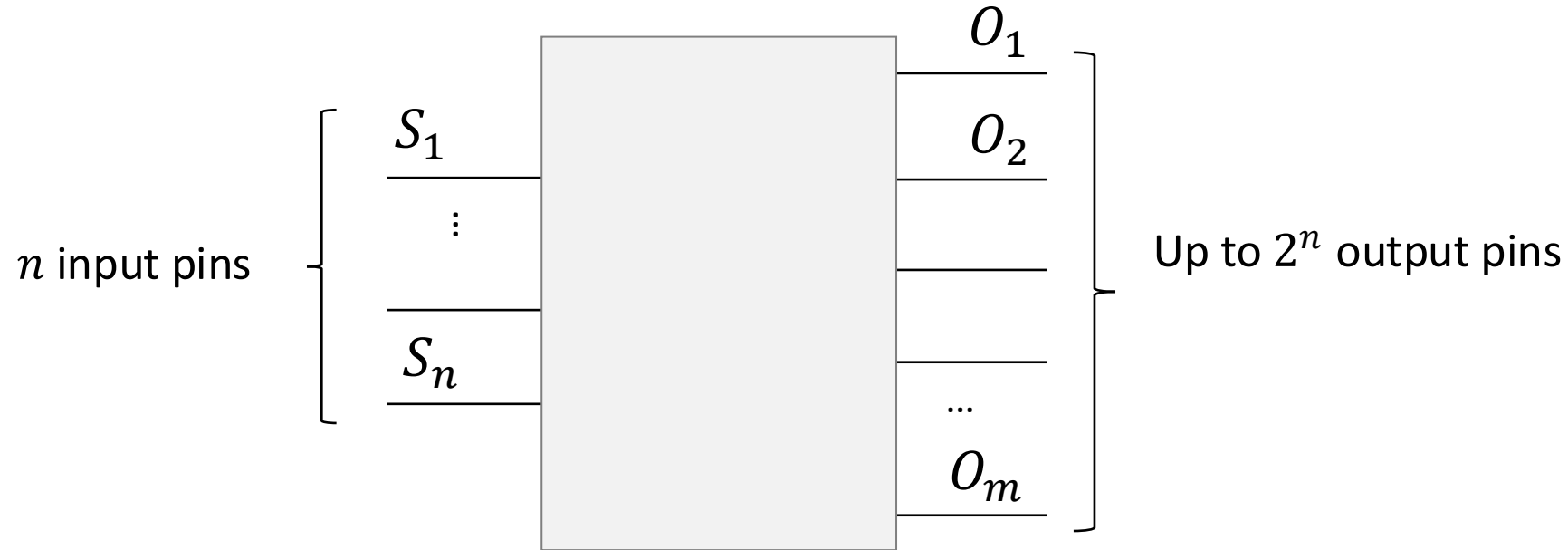
Demultiplexer: Transformation



Modifications:

- 1) We keep only control pins, and remove input pin Q ;
- 2) We set to "1" the output pin, which corresponds to the values of control pins;
- 3) **We change the shape to a rectangle**

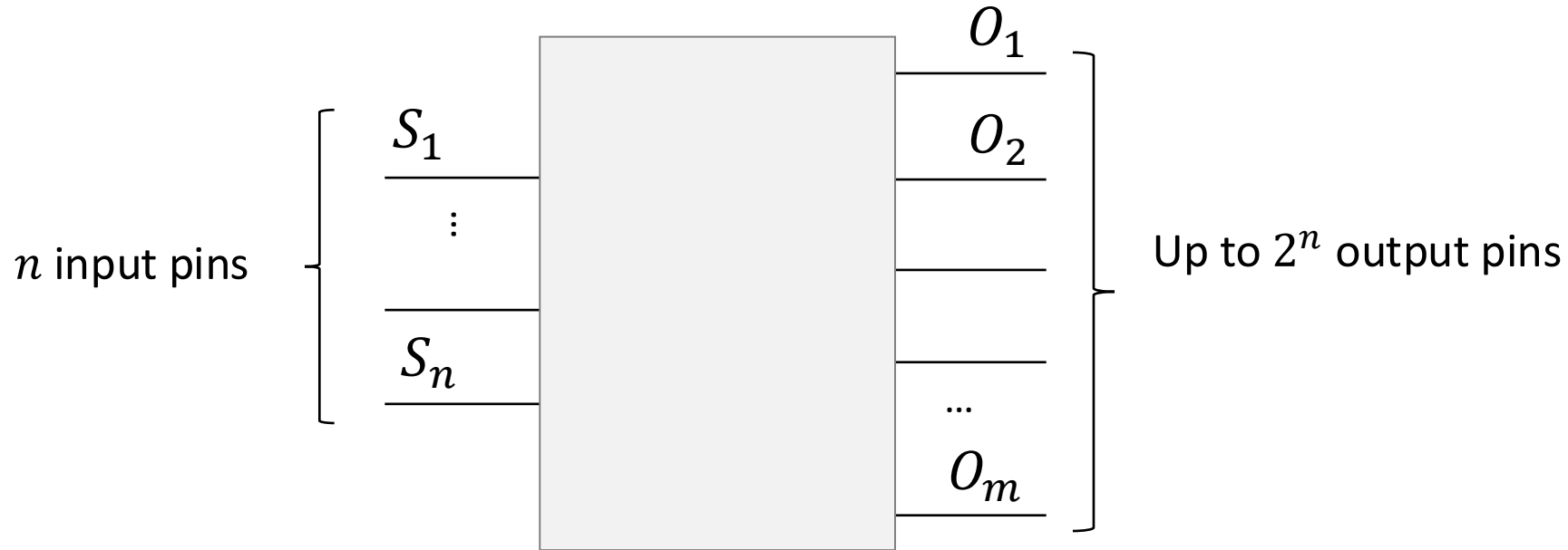
Decoder



Modifications:

- 1) We keep only control pins, and remove input pin Q;
- 2) We set to "1" the output pin, which corresponds to the values of control pins;
- 3) We change the shape to a rectangle

Decoder



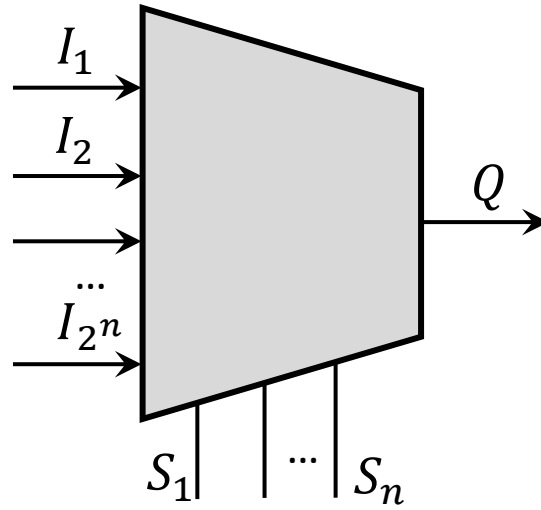
Decoder function:

- All output pins are set to “0”, except
- The only one pin is set to “1”, which corresponds to a given combination of inputs

Hardware Building Blocks

Multiplexer

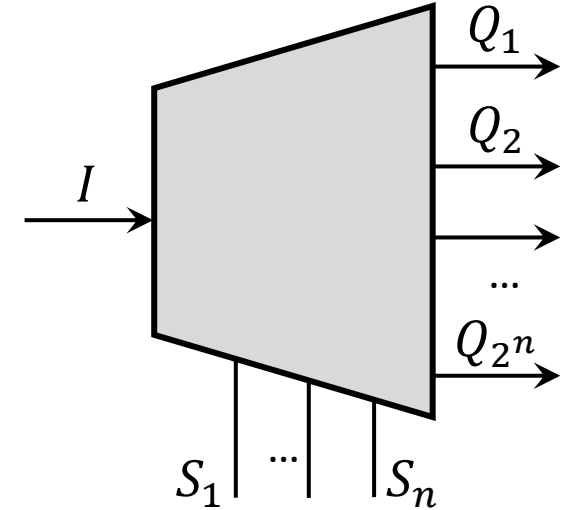
Output Q is set to one of inputs $I_1, \dots, I_{2^n}$, based on selectors $S_1, \dots, S_n$



Demultiplexer

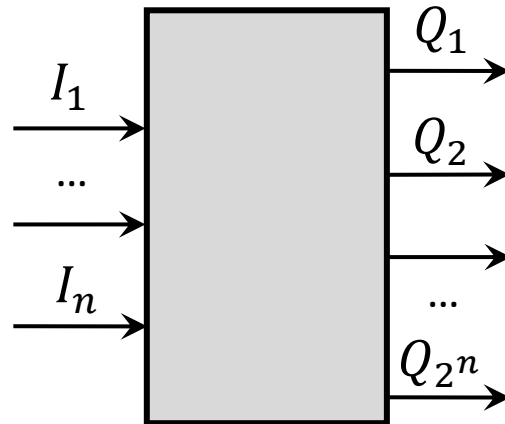
“Flipped” Multiplexer:

Some output Q_x is set to input I , while others Q_i with $i \neq x$ are set to “0”s



Decoder

Based on inputs $I_1, \dots, I_{2^n}$, some output Q_x is set to “1”, while others Q_i with $i \neq x$ are set to “0”s

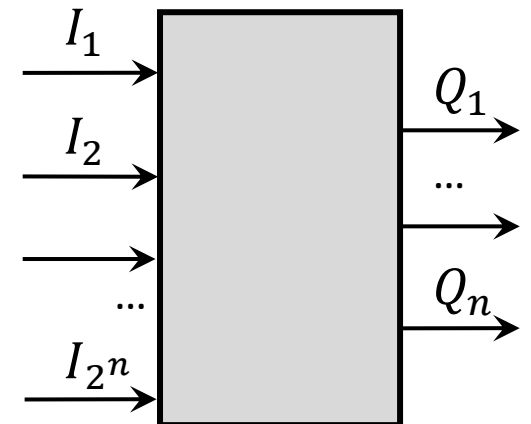


Encoder

“Flipped” Decoder:

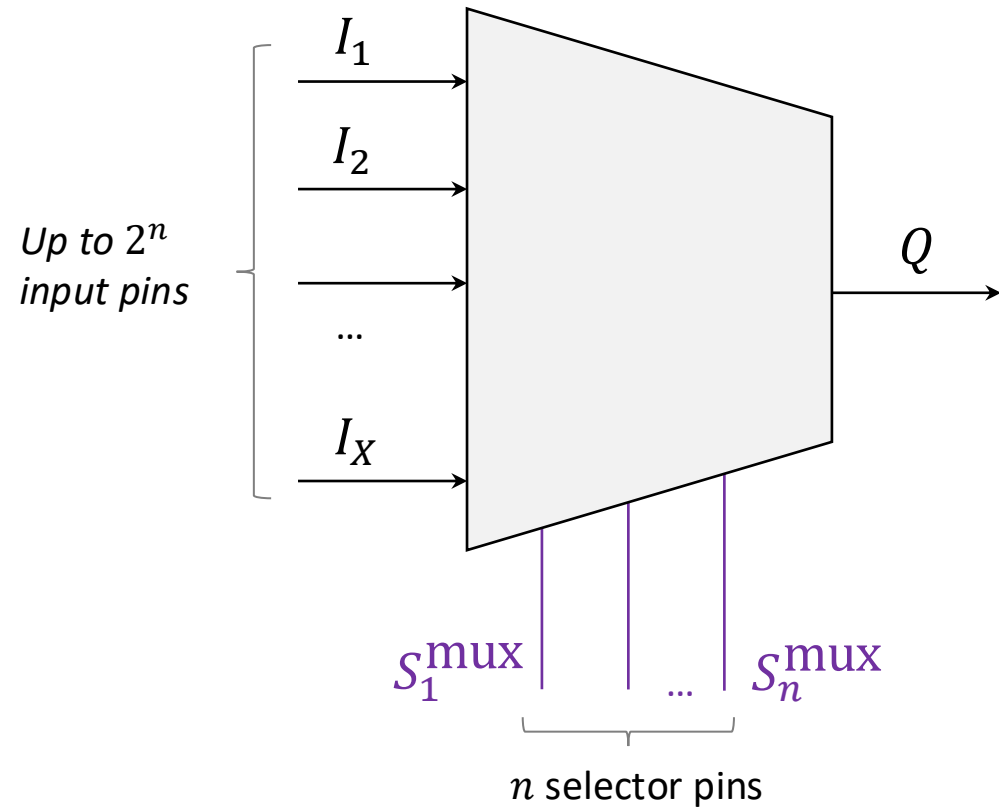
Only one input in $I_1, \dots, I_{2^n}$ is allowed to be “1”;

For given inputs, there is a pre-defined set of outputs $Q_1, \dots, Q_n$



X-to-1 Multiplexer

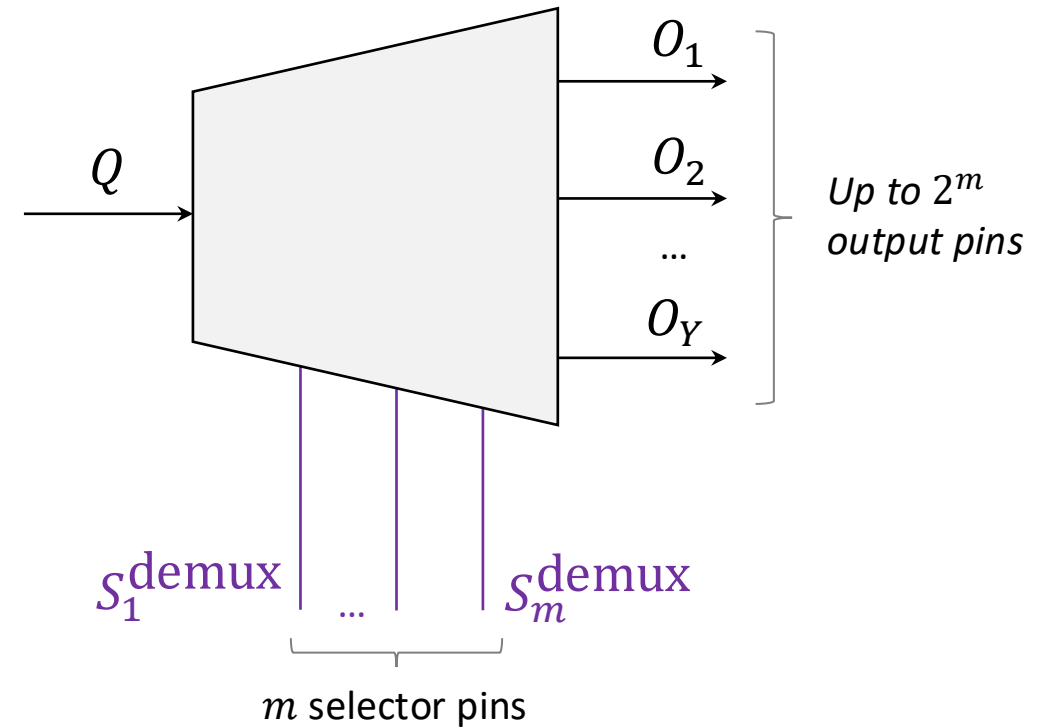
Based on the values of its n selector (control) pins, a multiplexer sets the value of its output Q to the value of one of its $X \leq 2^n$ inputs



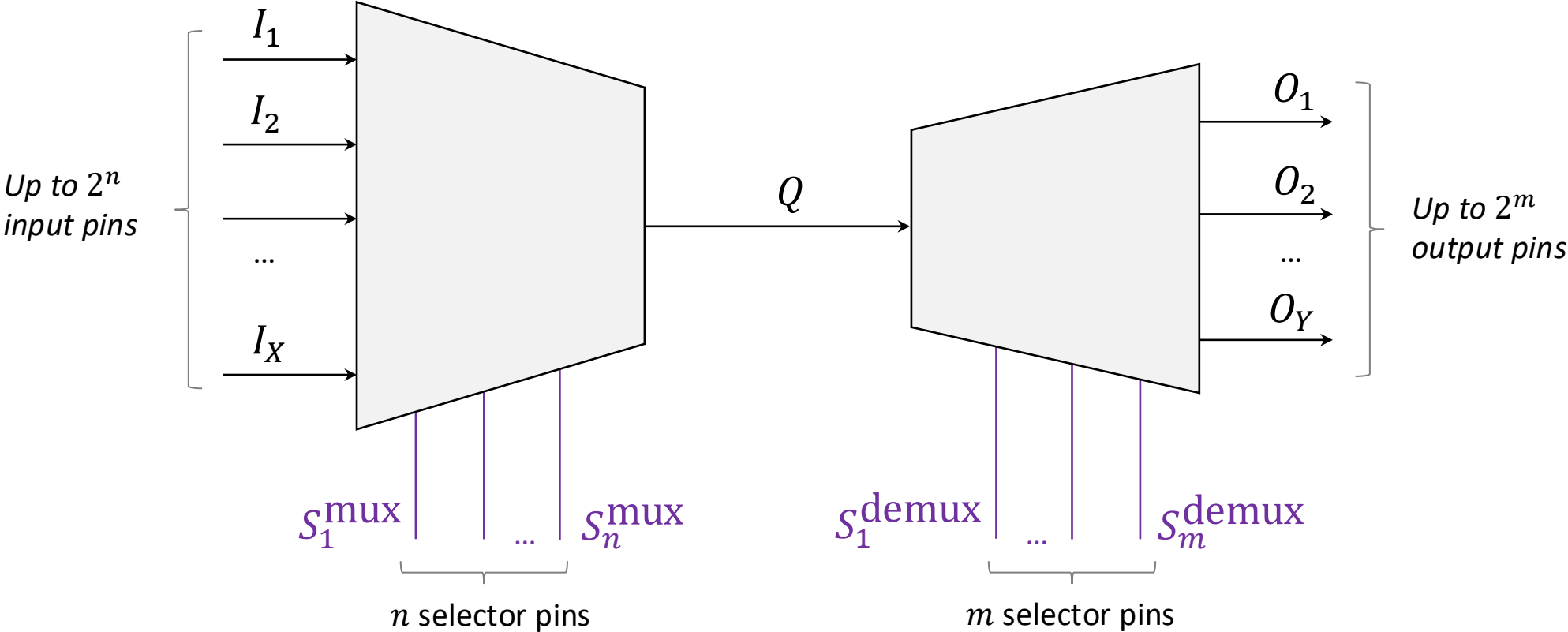
Question: How to construct an X-to-Y Multiplexer?

1-to-Y Demultiplexer

Based on the values of its m selector pins, a demultiplexer outputs its input value Q at one of its $Y \leq 2^m$ output pins, while all other output pins are set to 0s

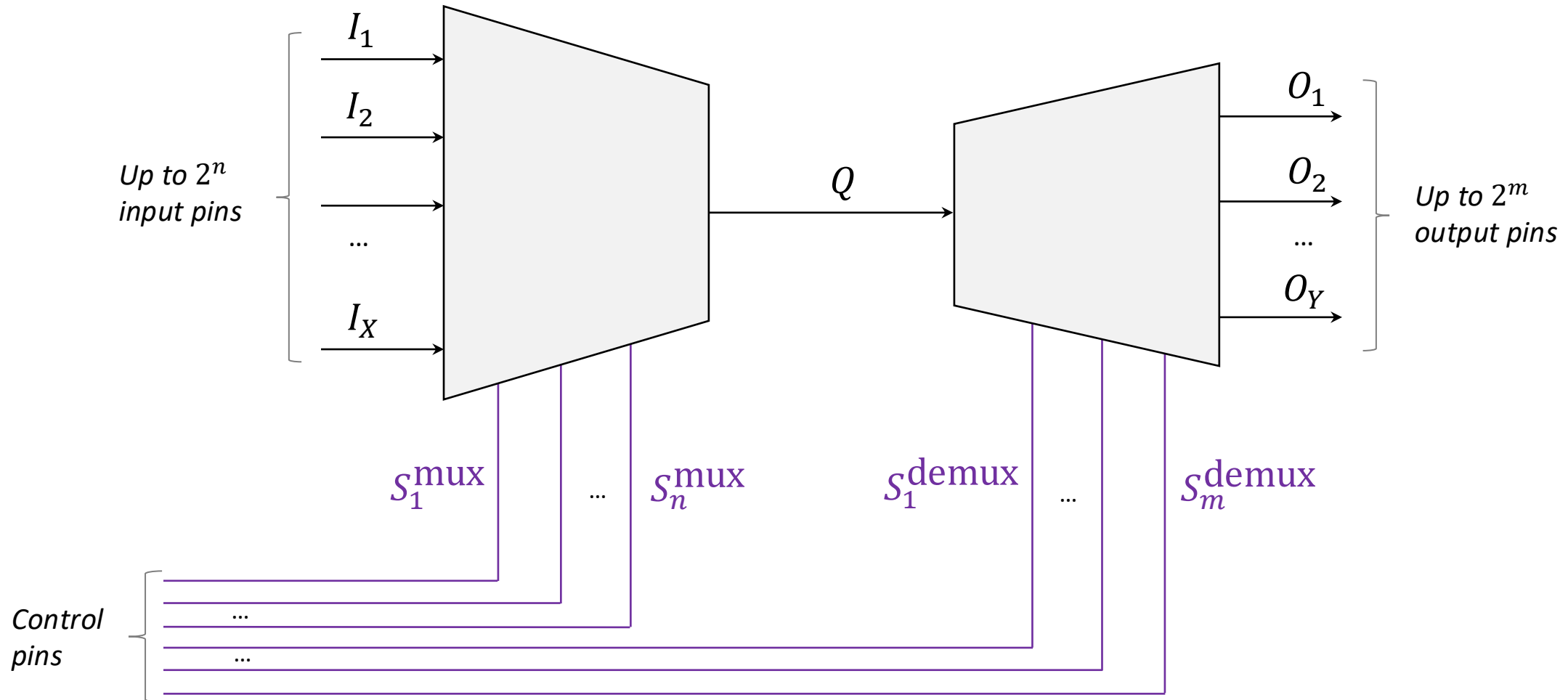


Multiplexer-Demultiplexer Composition



X-to-Y Multiplexer:

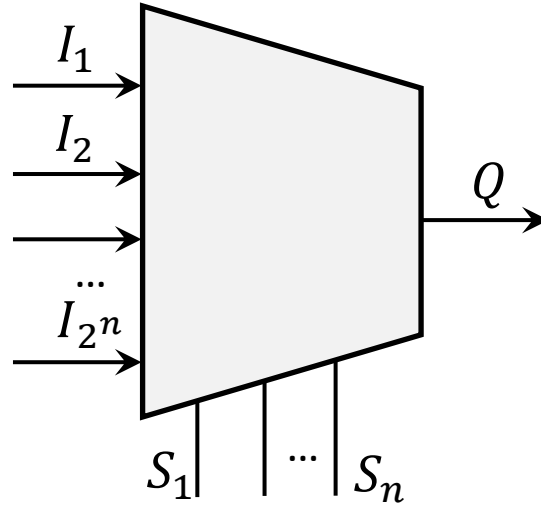
Chooses one of its X input pins, and outputs its value at one of its Y output pins, while all other output pins are reset to 0s (usually)



Hardware Building Blocks

Multiplexer

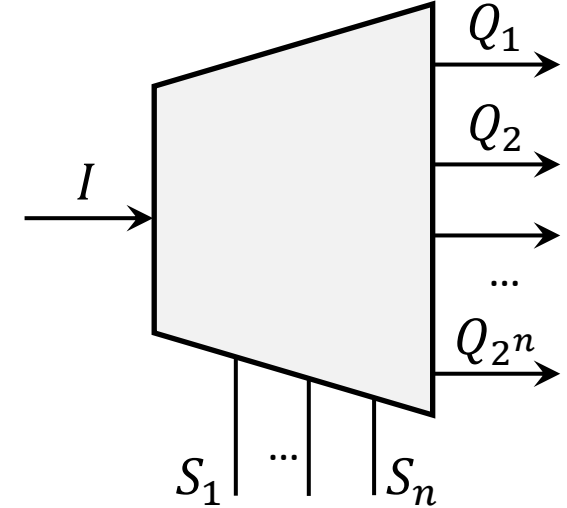
Output Q is set to one of inputs $I_1, \dots, I_{2^n}$, based on selectors $S_1, \dots, S_n$



Demultiplexer

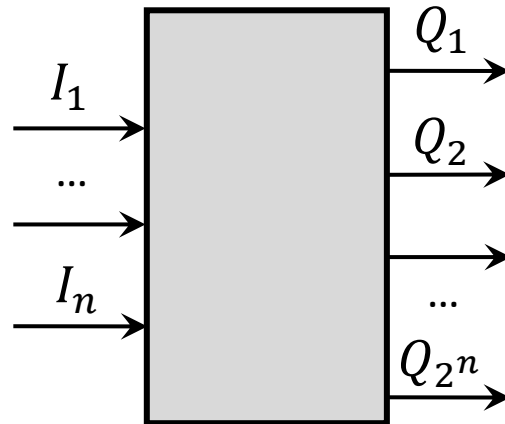
“Flipped” Multiplexer:

Some output Q_x is set to input I , while others Q_i with $i \neq x$ are set to “0”s



Decoder

Based on inputs $I_1, \dots, I_{2^n}$, some output Q_x is set to “1”, while others Q_i with $i \neq x$ are set to “0”s

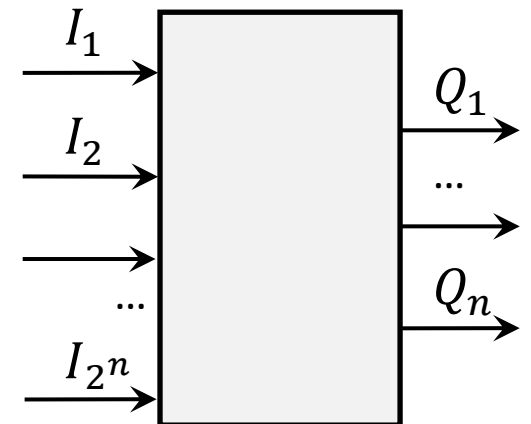


Encoder

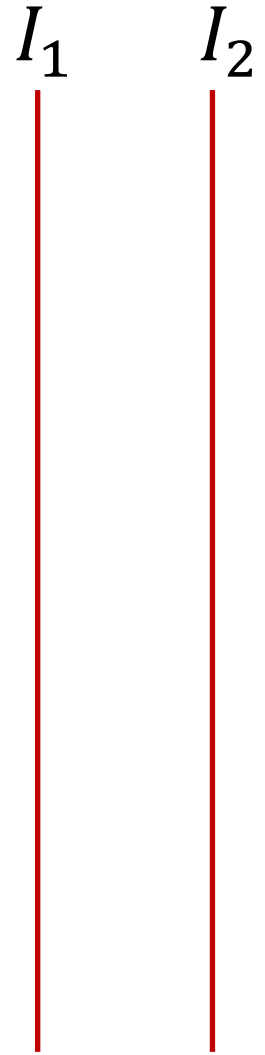
“Flipped” Decoder:

Only one input in $I_1, \dots, I_{2^n}$ is allowed to be “1”;

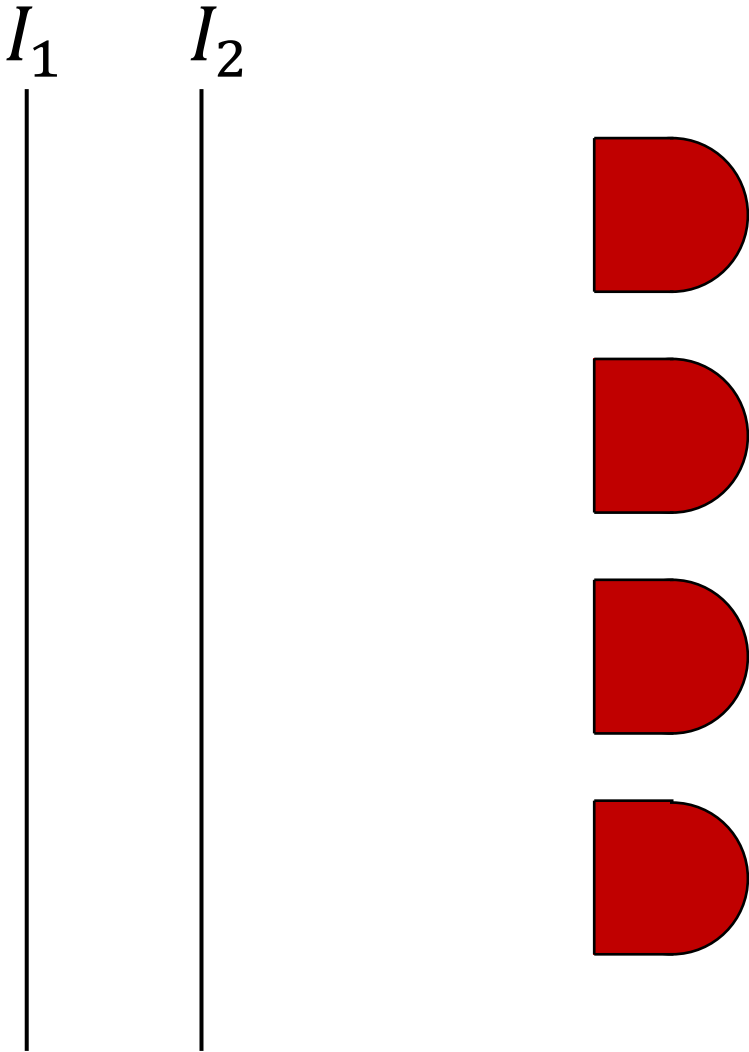
For given inputs, there is a pre-defined set of outputs $Q_1, \dots, Q_n$



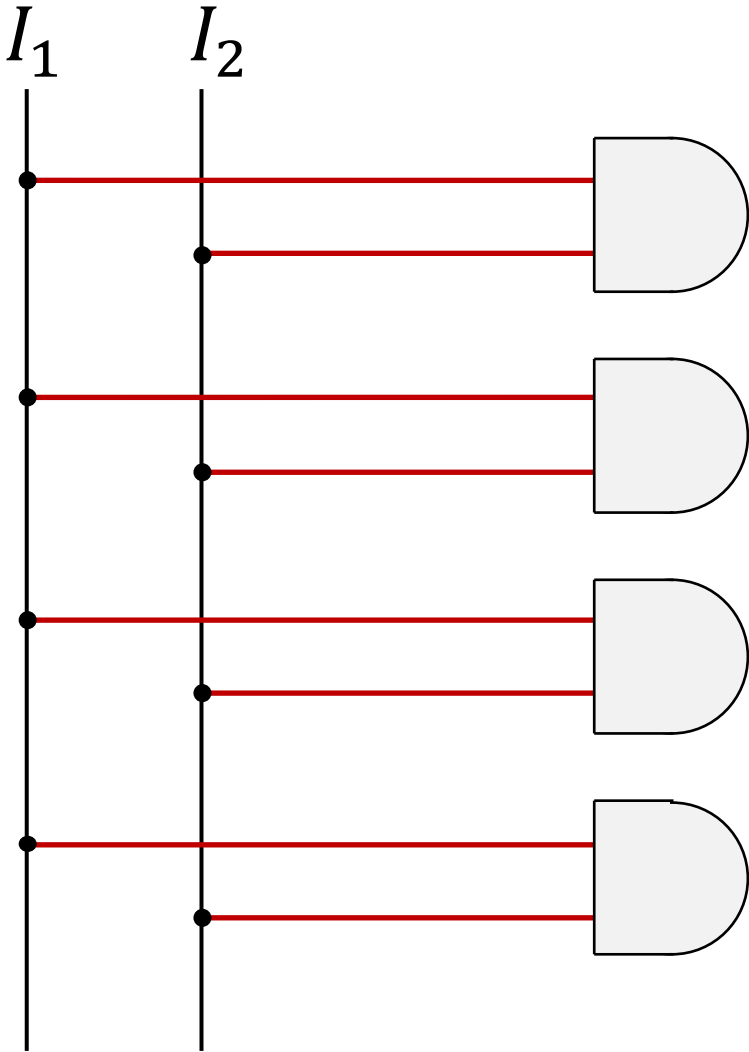
2-to-4 Decoder Implementation



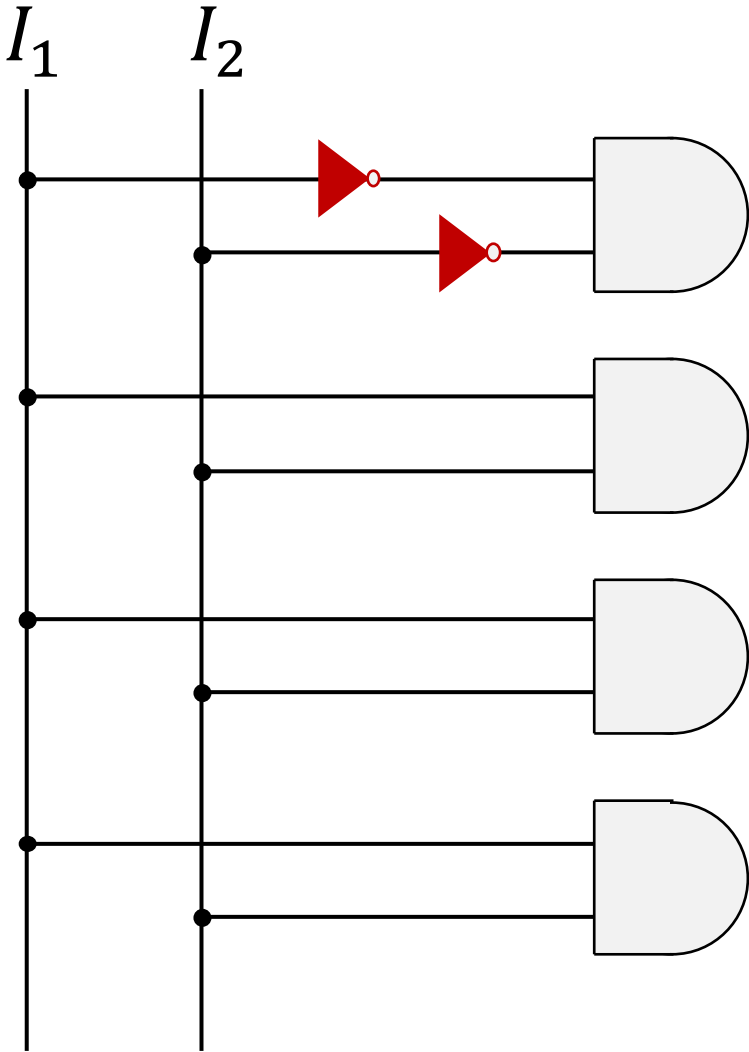
2-to-4 Decoder Implementation



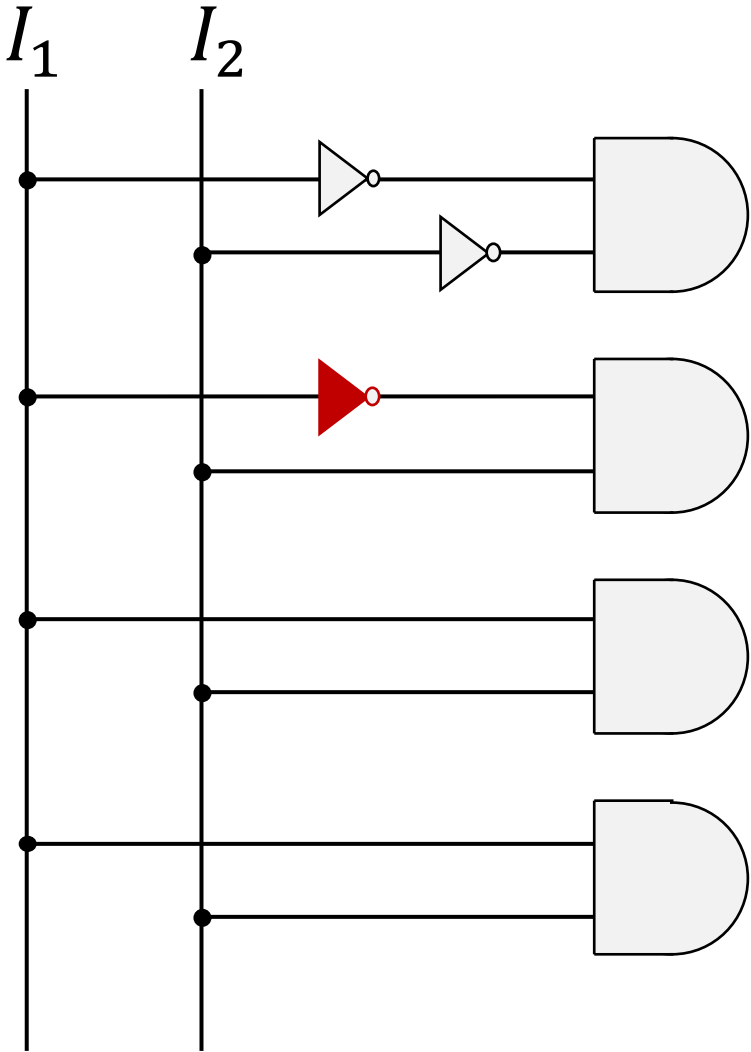
2-to-4 Decoder Implementation



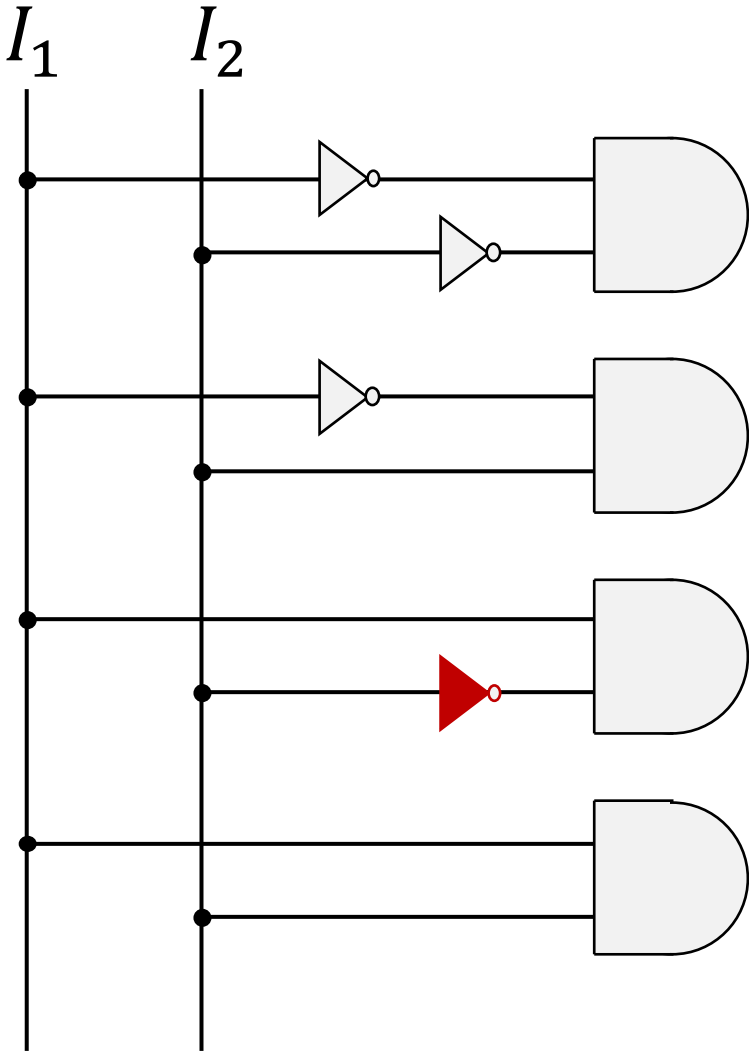
2-to-4 Decoder Implementation



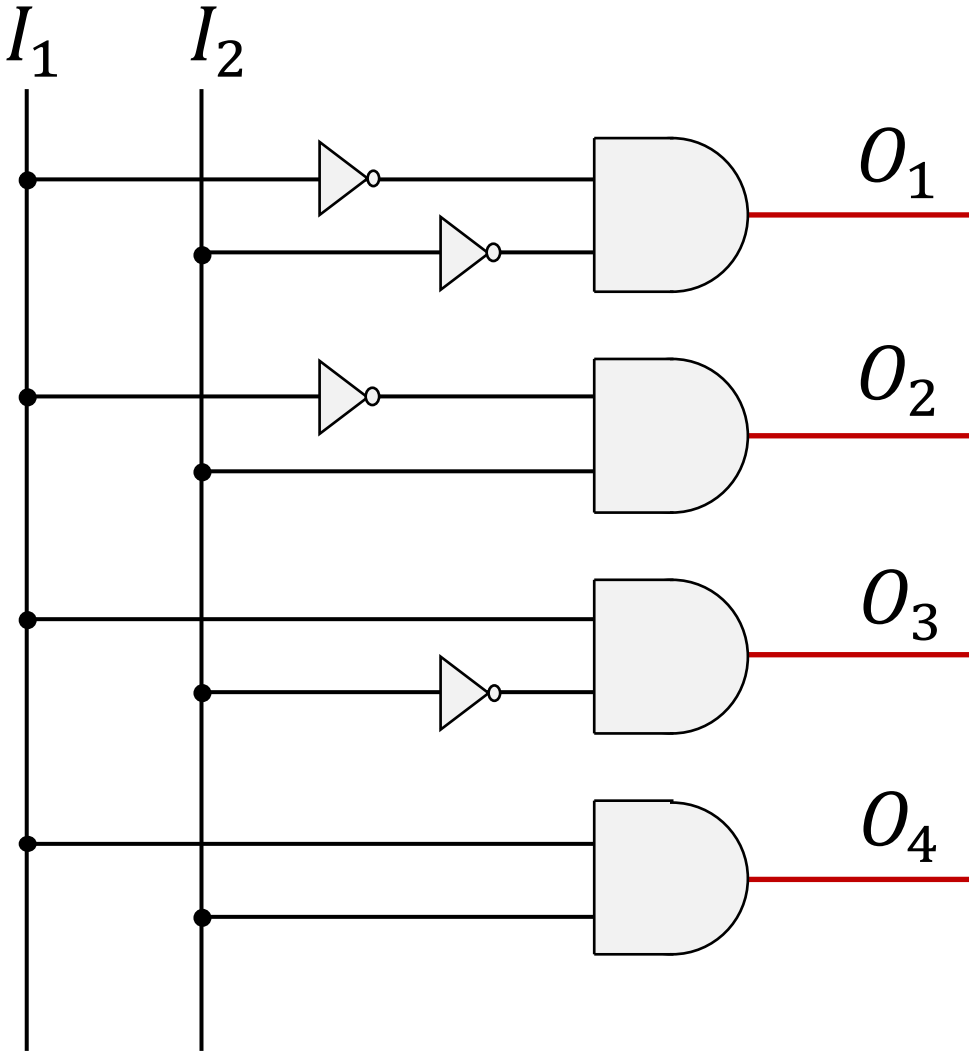
2-to-4 Decoder Implementation



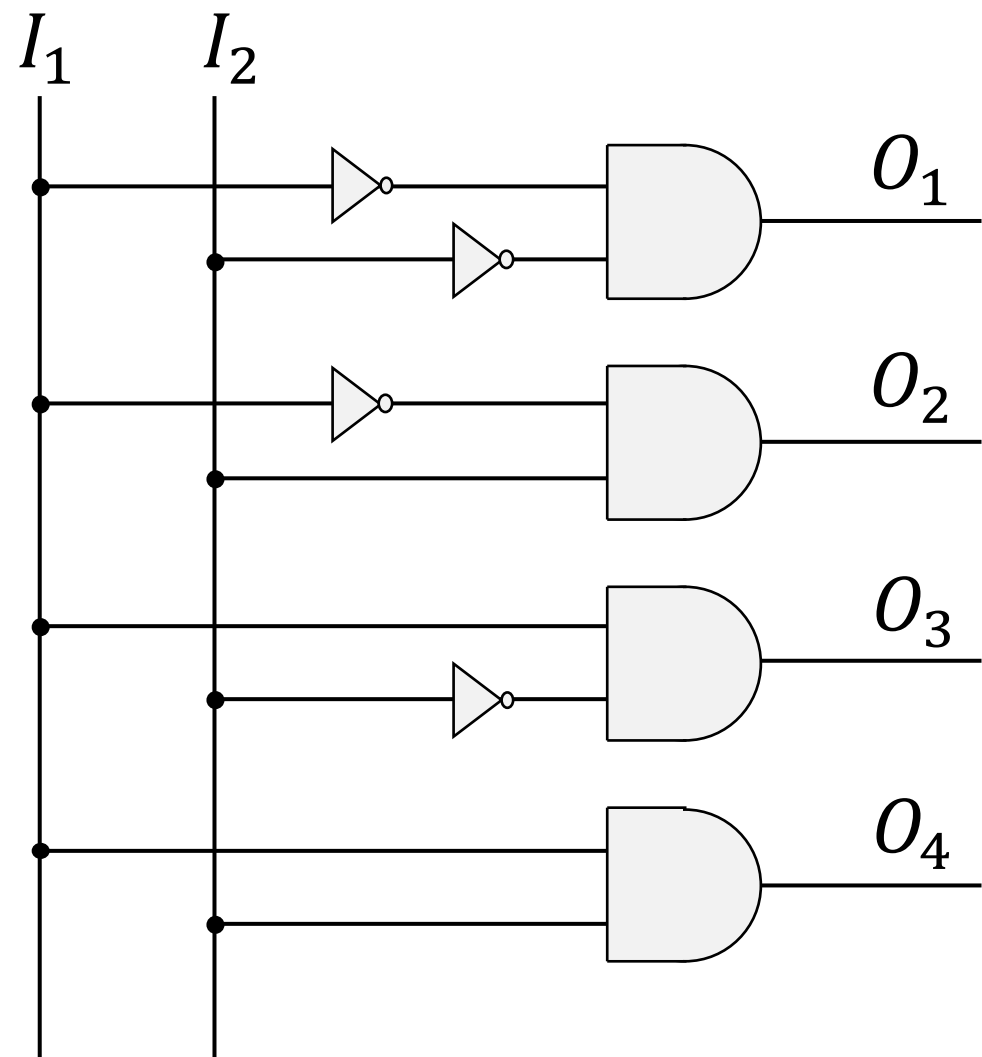
2-to-4 Decoder Implementation



2-to-4 Decoder Implementation

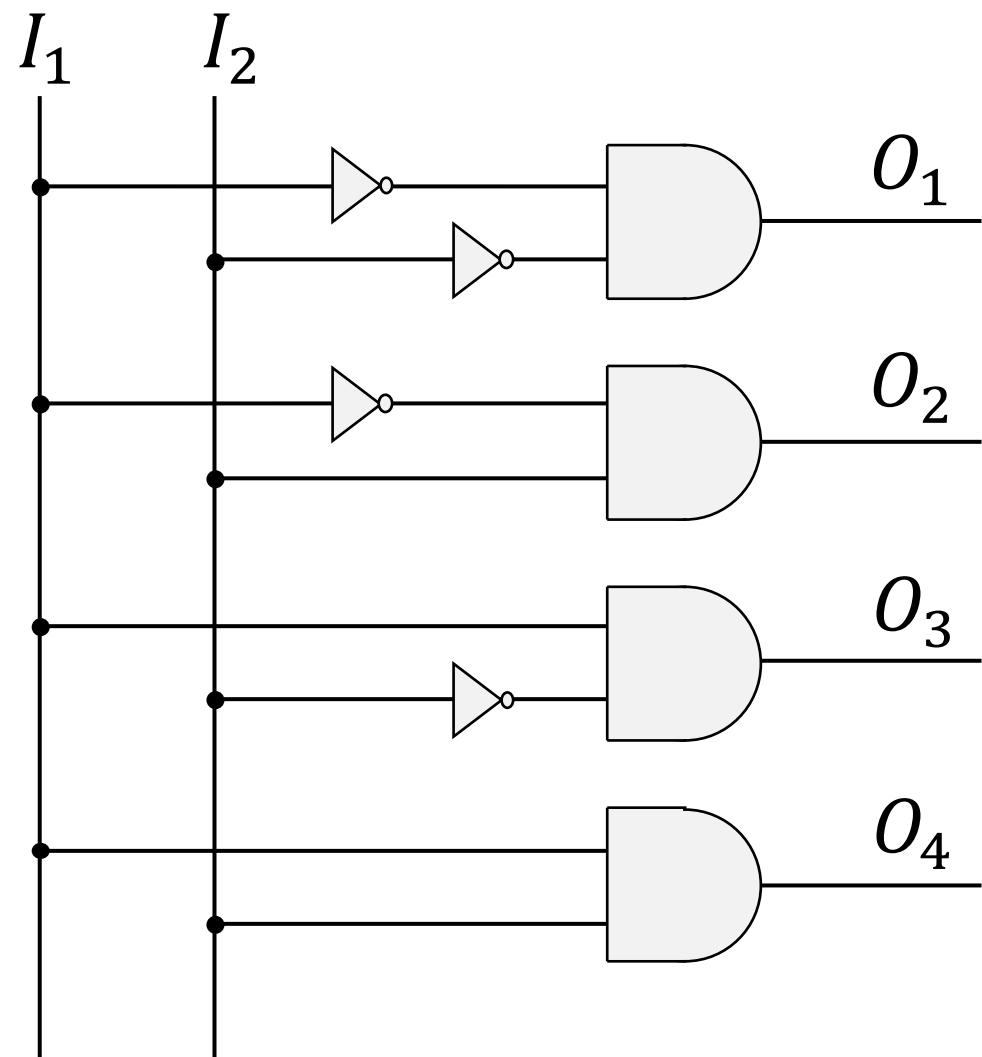


2-to-4 Decoder Implementation



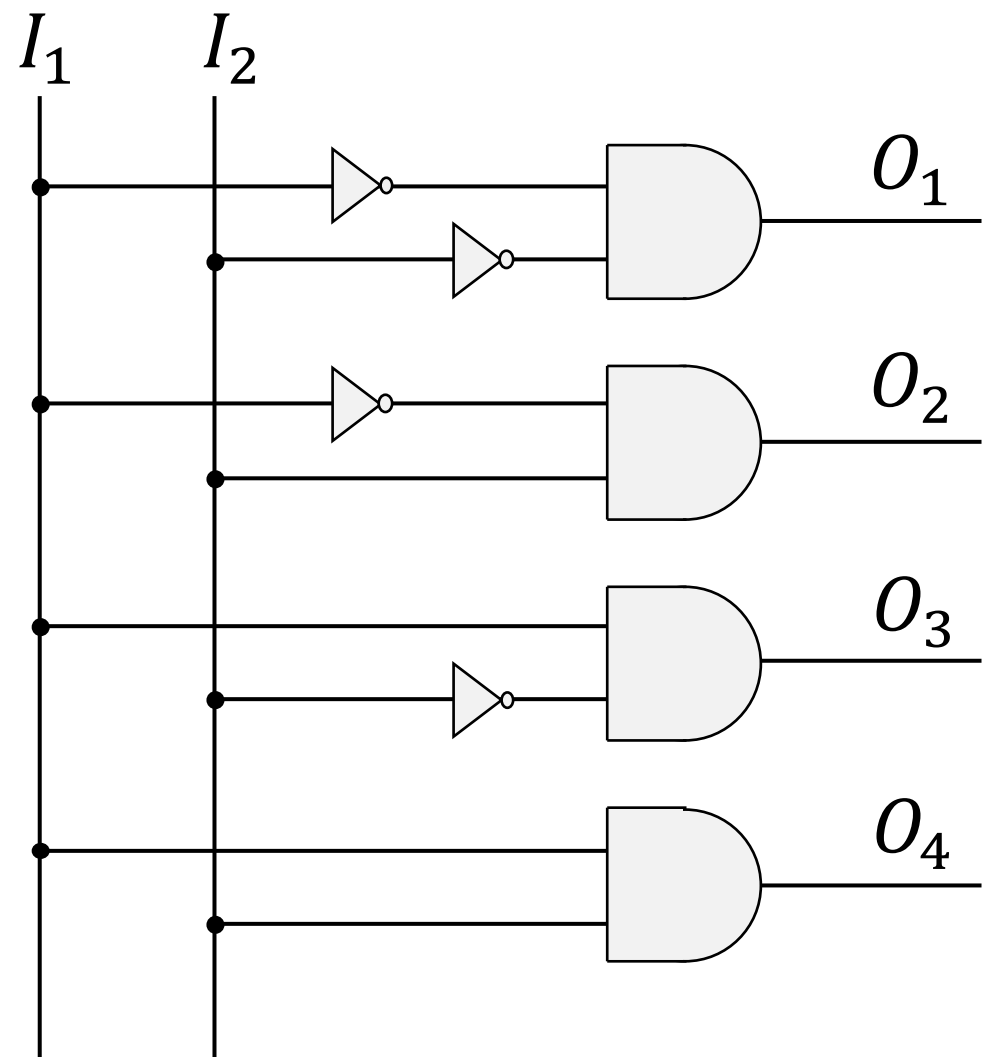
I_1	I_2	O_1	O_2	O_3	O_4
0	0				

2-to-4 Decoder Implementation



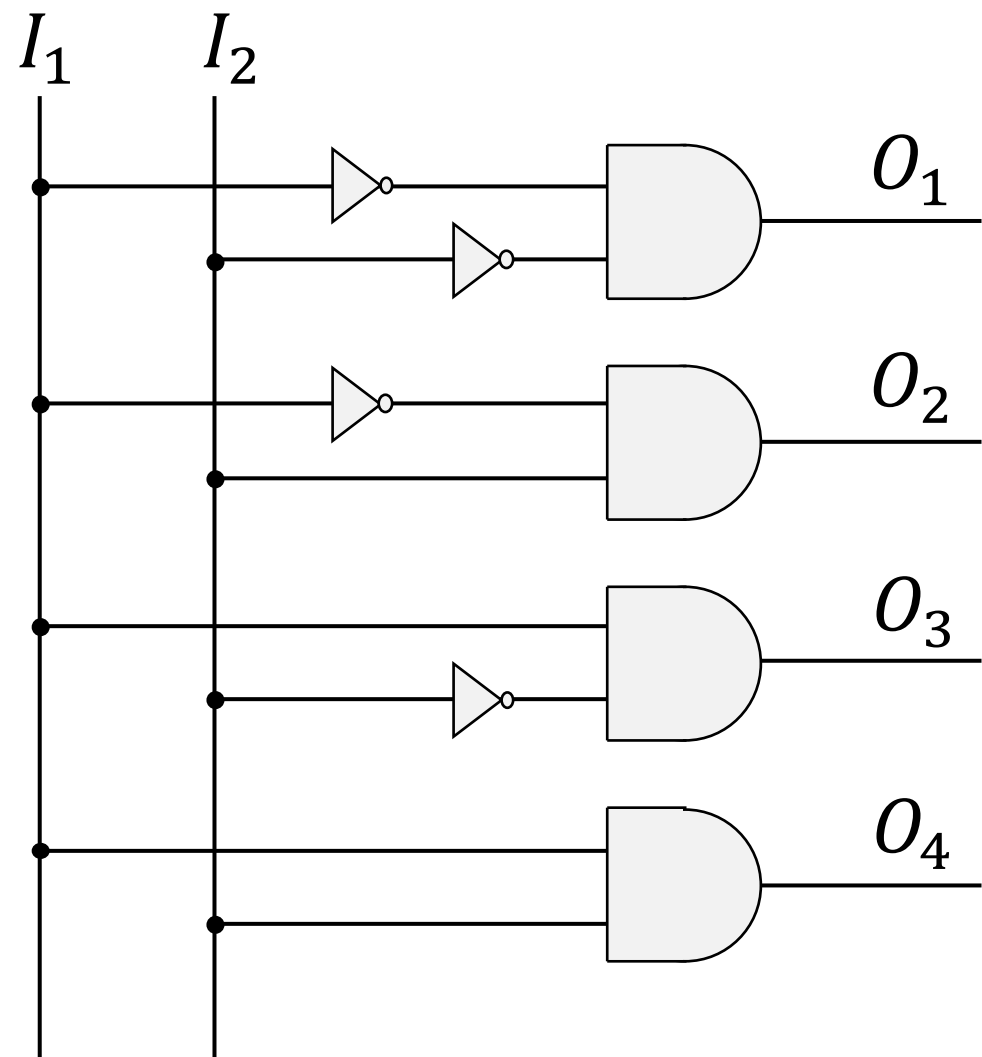
I_1	I_2	O_1	O_2	O_3	O_4
0	0	1			

2-to-4 Decoder Implementation



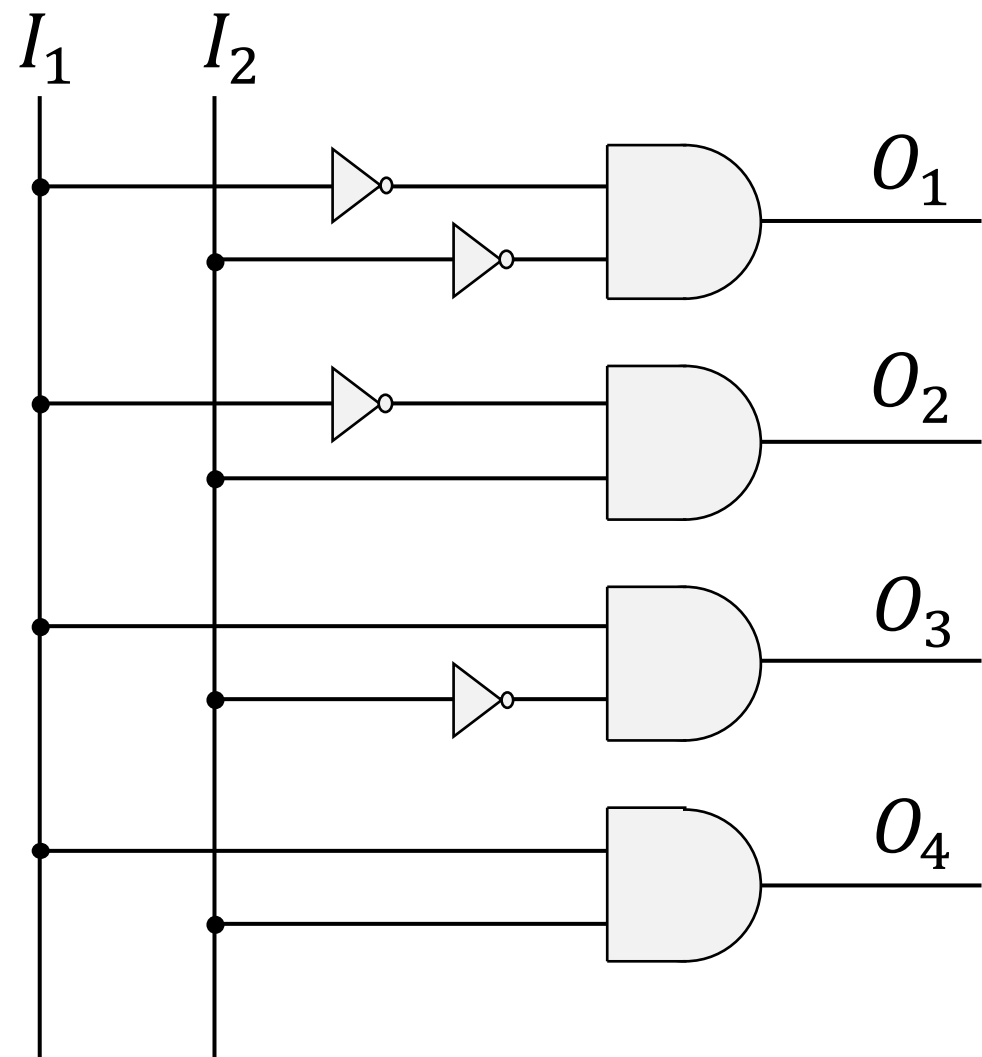
I_1	I_2	O_1	O_2	O_3	O_4
0	0	1	0	0	0

2-to-4 Decoder Implementation



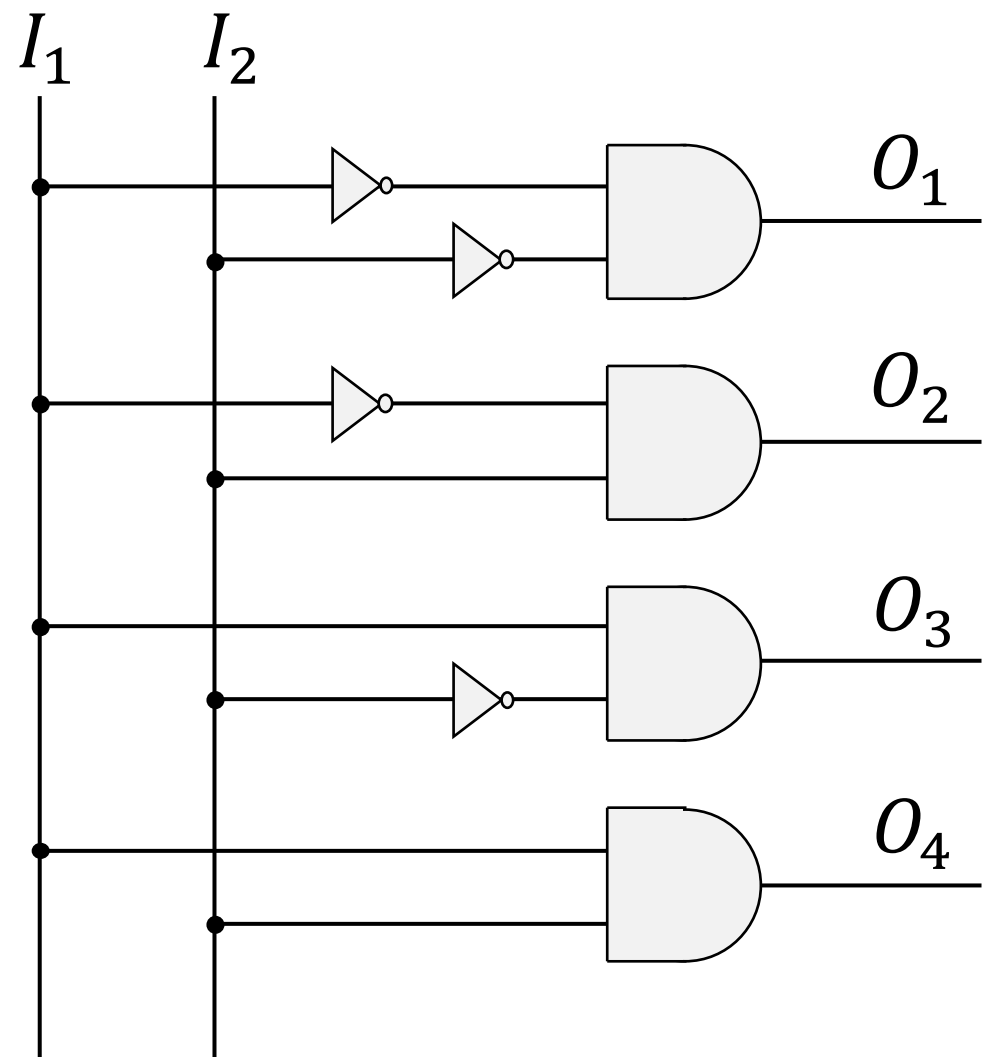
I_1	I_2	O_1	O_2	O_3	O_4
0	0	1	0	0	0
0	1		1		

2-to-4 Decoder Implementation



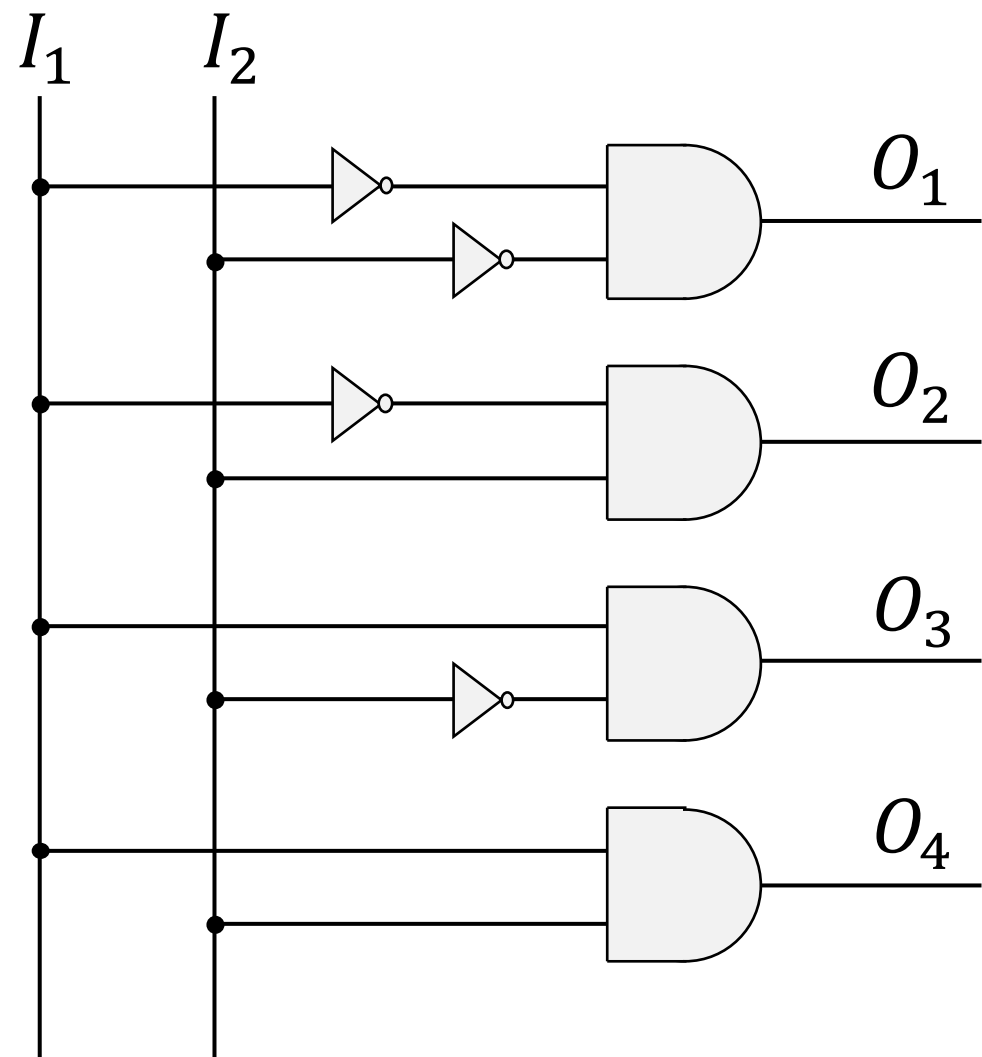
I_1	I_2	O_1	O_2	O_3	O_4
0	0	1	0	0	0
0	1	0	1	0	0

2-to-4 Decoder Implementation



I_1	I_2	O_1	O_2	O_3	O_4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0

2-to-4 Decoder Implementation

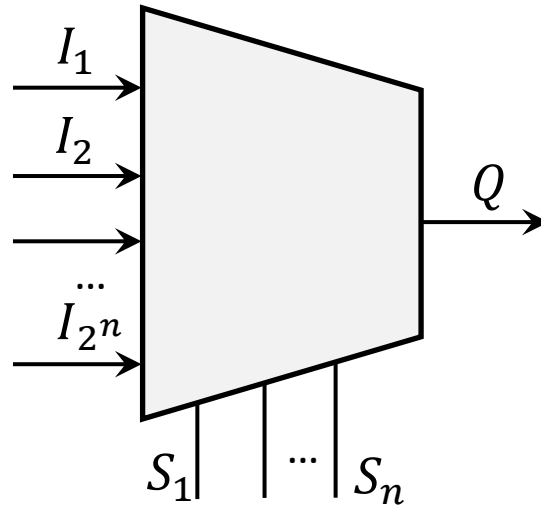


I_1	I_2	O_1	O_2	O_3	O_4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Hardware Building Blocks

Multiplexer

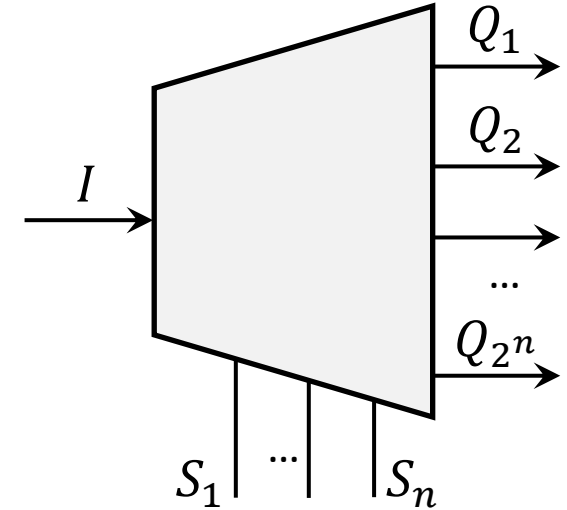
Output Q is set to one of inputs $I_1, \dots, I_{2^n}$, based on selectors $S_1, \dots, S_n$



Demultiplexer

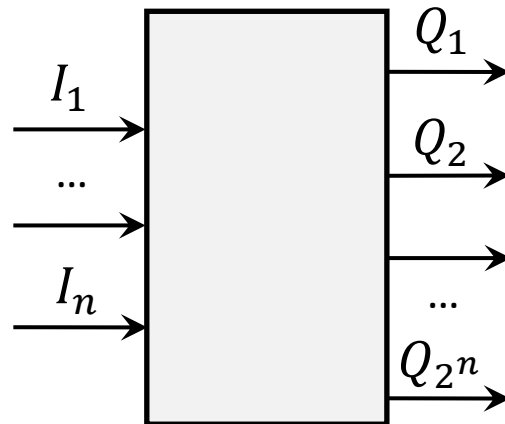
“Flipped” Multiplexer:

Some output Q_x is set to input I , while others Q_i with $i \neq x$ are set to “0”s



Decoder

Based on inputs $I_1, \dots, I_{2^n}$, some output Q_x is set to “1”, while others Q_i with $i \neq x$ are set to “0”s

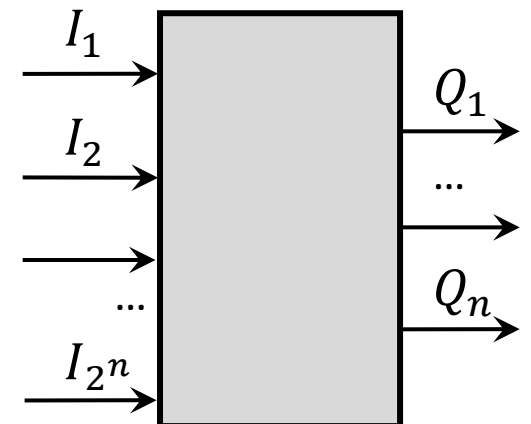


Encoder

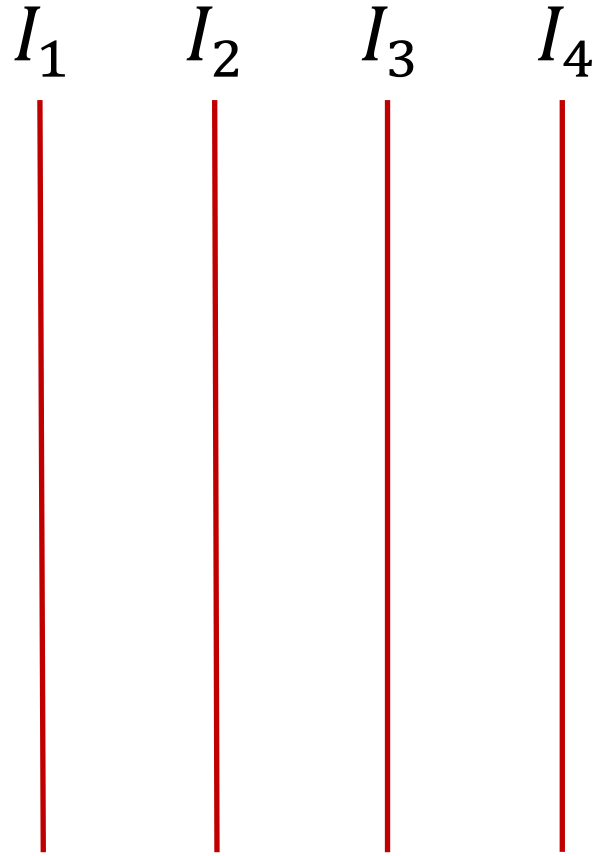
“Flipped” Decoder:

Only one input in $I_1, \dots, I_{2^n}$ is allowed to be “1”;

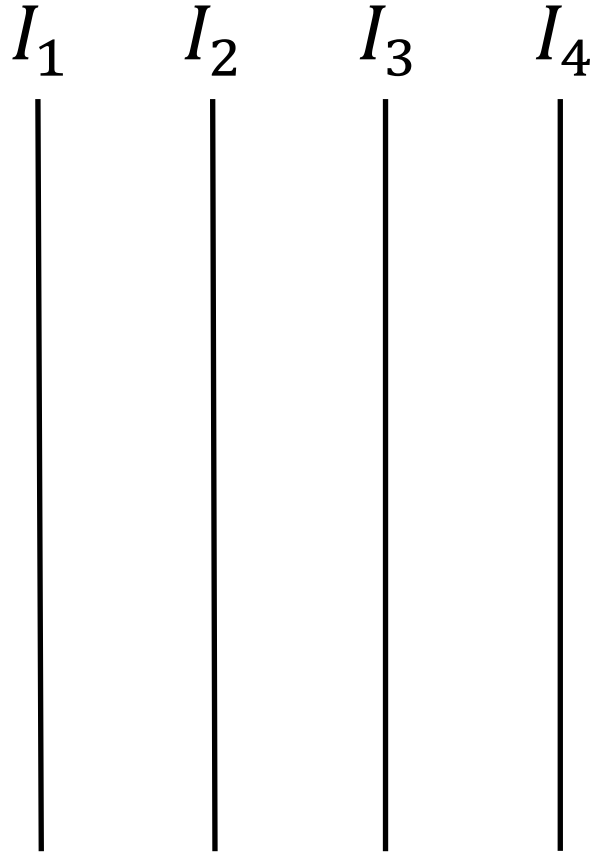
For given inputs, there is a pre-defined set of outputs $Q_1, \dots, Q_n$



4-to-2 Encoder Implementation



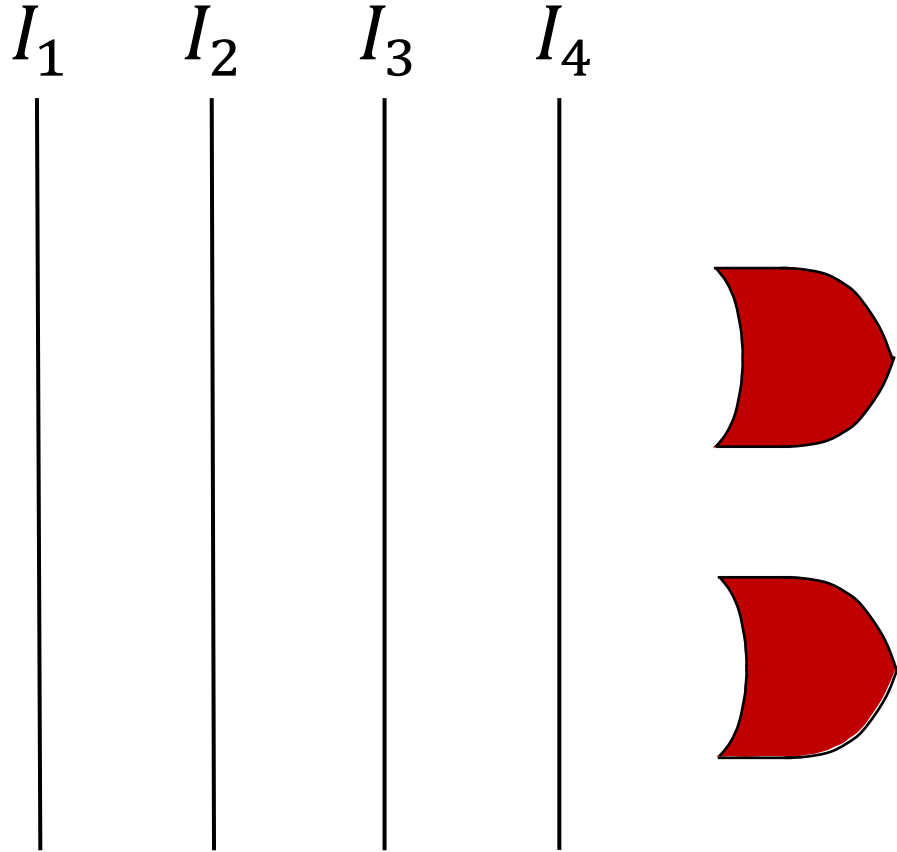
4-to-2 Encoder Implementation



Assumption by definition:

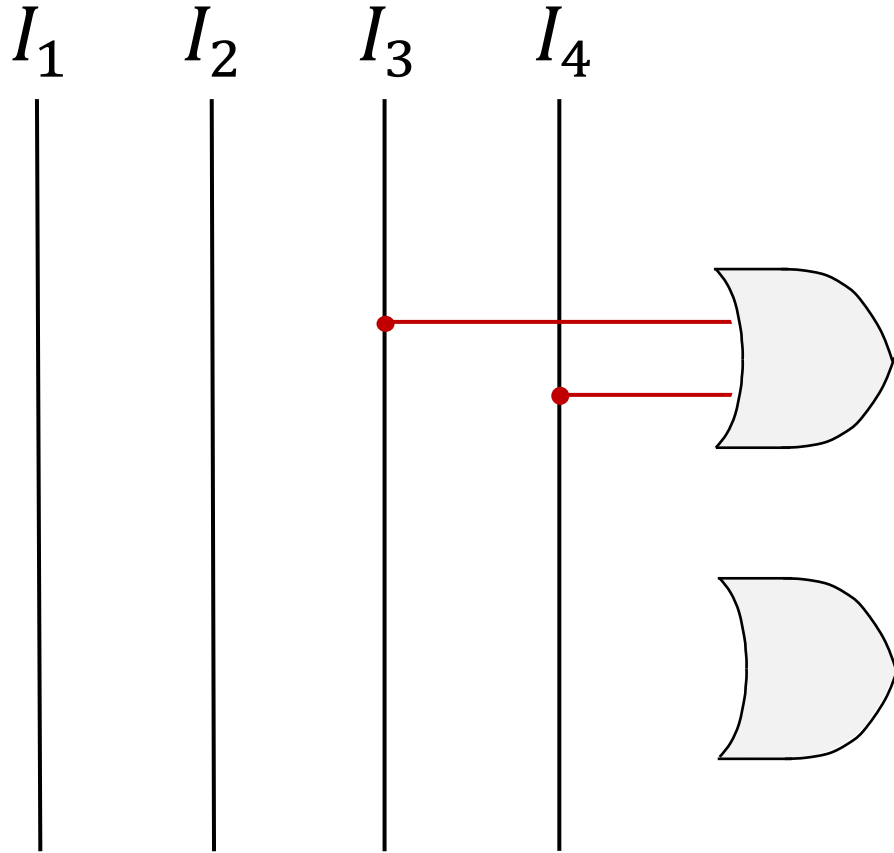
Exactly one input is set to “1”, while all others are “0”s

4-to-2 Encoder Implementation



Assumption by definition:
Exactly one input is set to “1”, while all others are “0”s

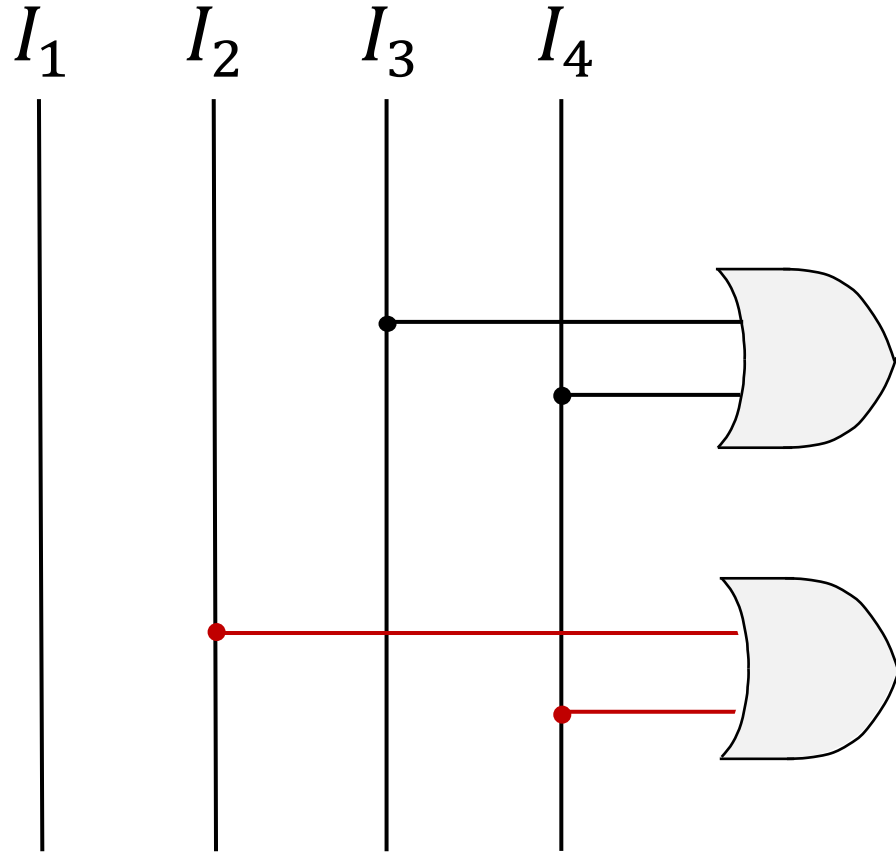
4-to-2 Encoder Implementation



Assumption by definition:

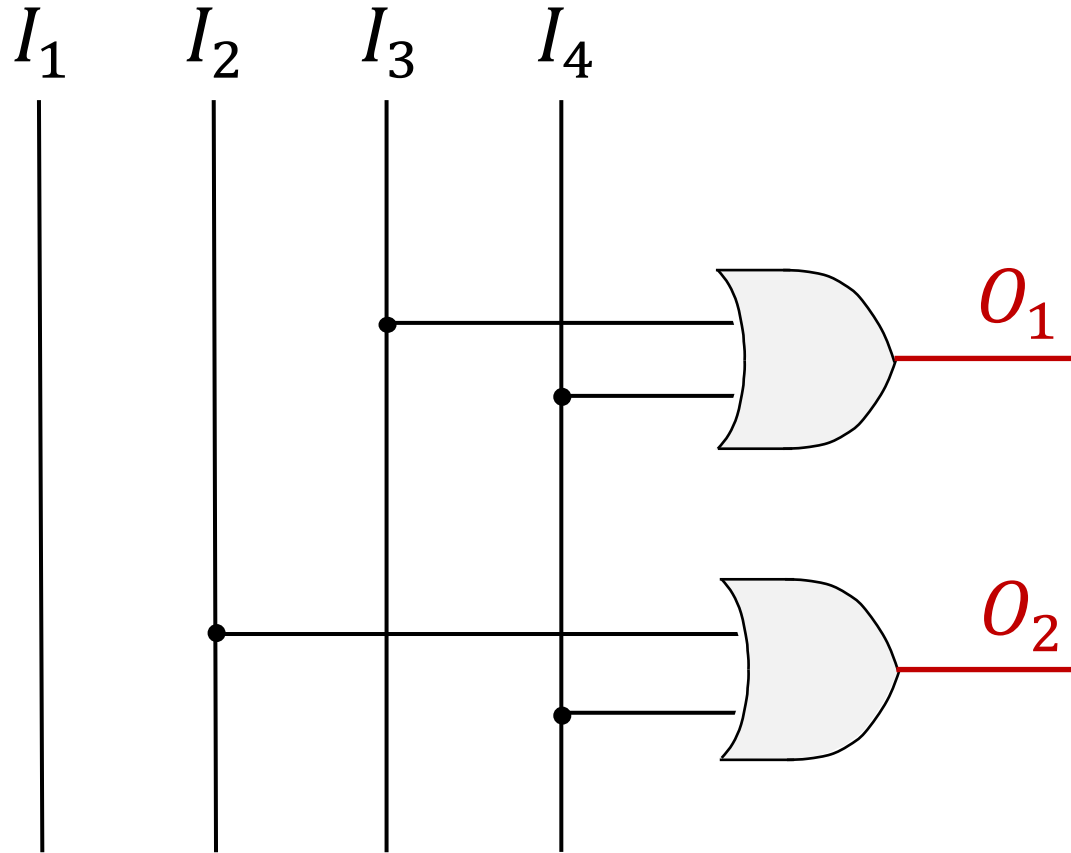
Exactly one input is set to “1”, while all others are “0”s

4-to-2 Encoder Implementation



Assumption by definition:
Exactly one input is set to “1”, while all others are “0”s

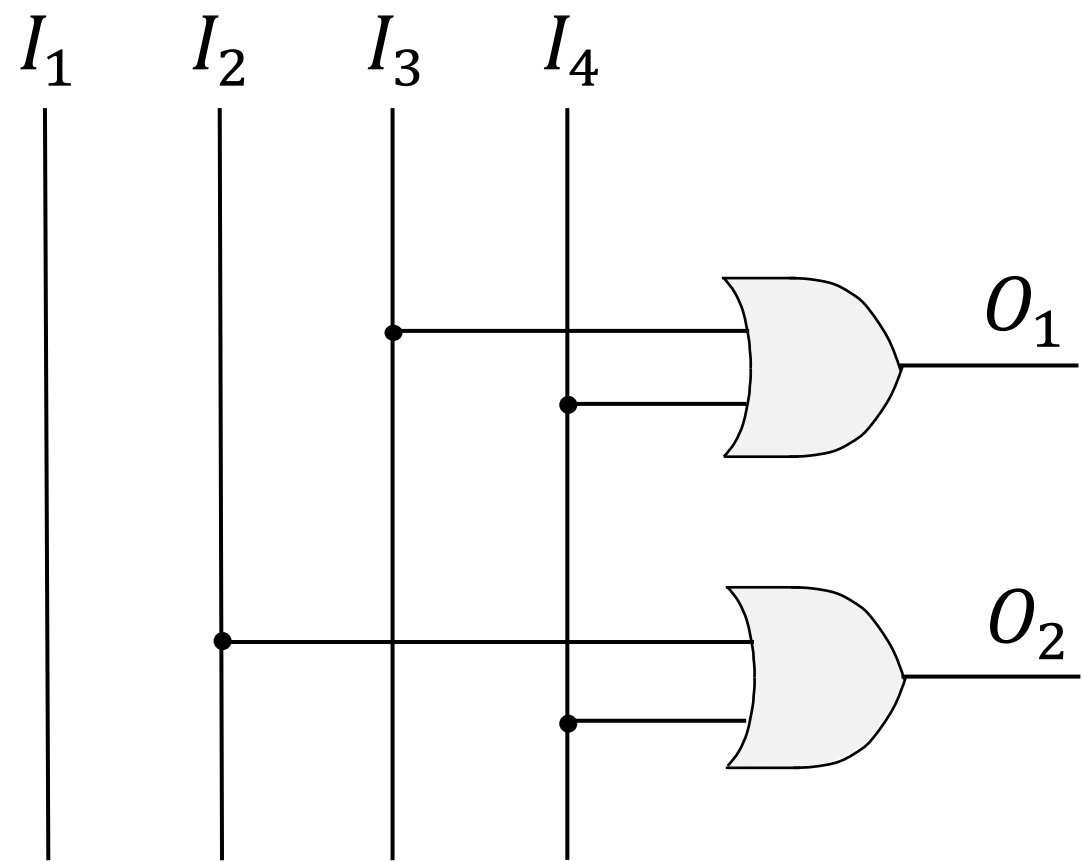
4-to-2 Encoder Implementation



Assumption by definition:

Exactly one input is set to "1", while all others are "0"s

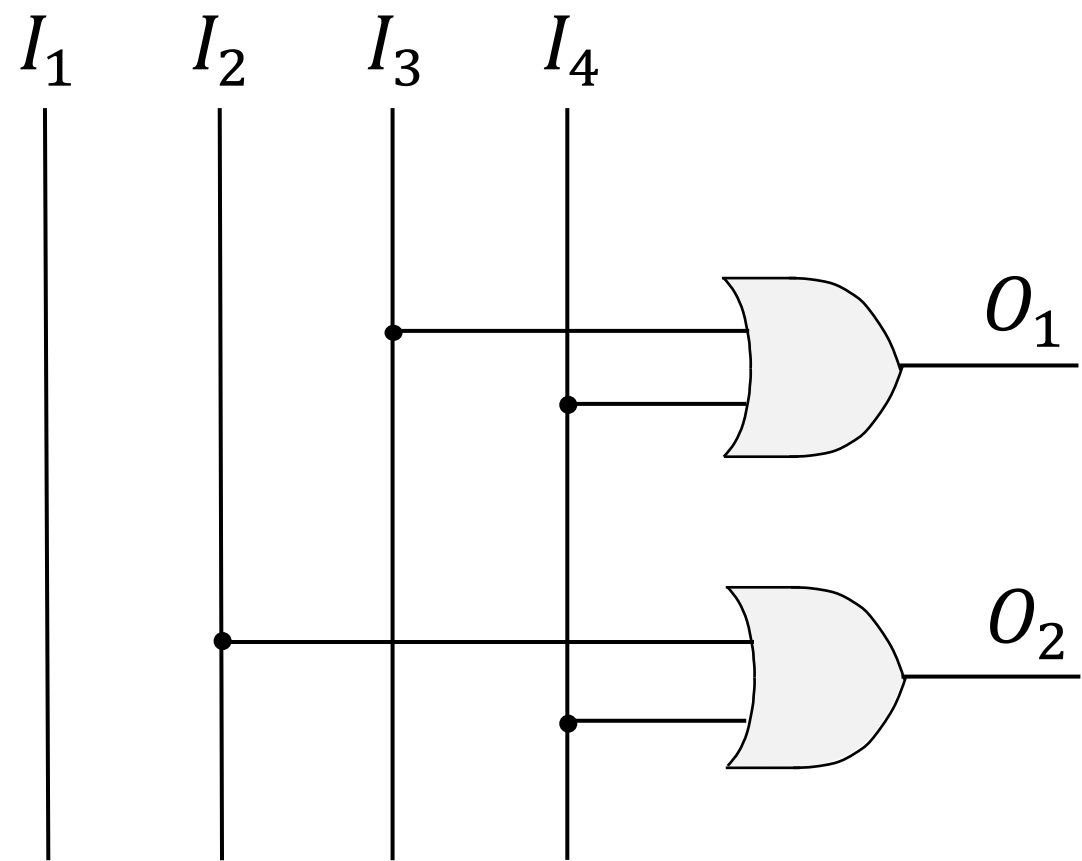
4-to-2 Encoder Implementation



I_1	I_2	I_3	I_4	O_1	O_2
1	0	0	0	0	0

Assumption by definition:
Exactly one input is set to “1”, while all others are “0”s

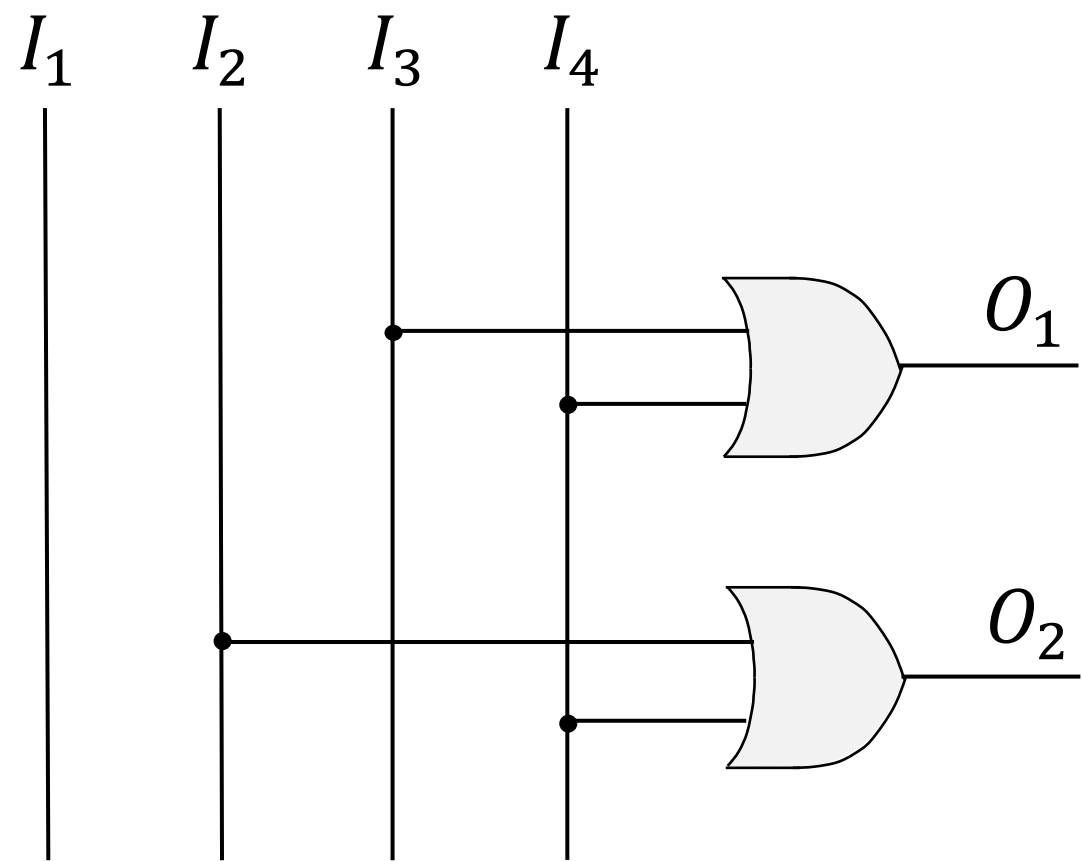
4-to-2 Encoder Implementation



I_1	I_2	I_3	I_4	O_1	O_2
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Assumption by definition:
Exactly one input is set to “1”, while all others are “0”s

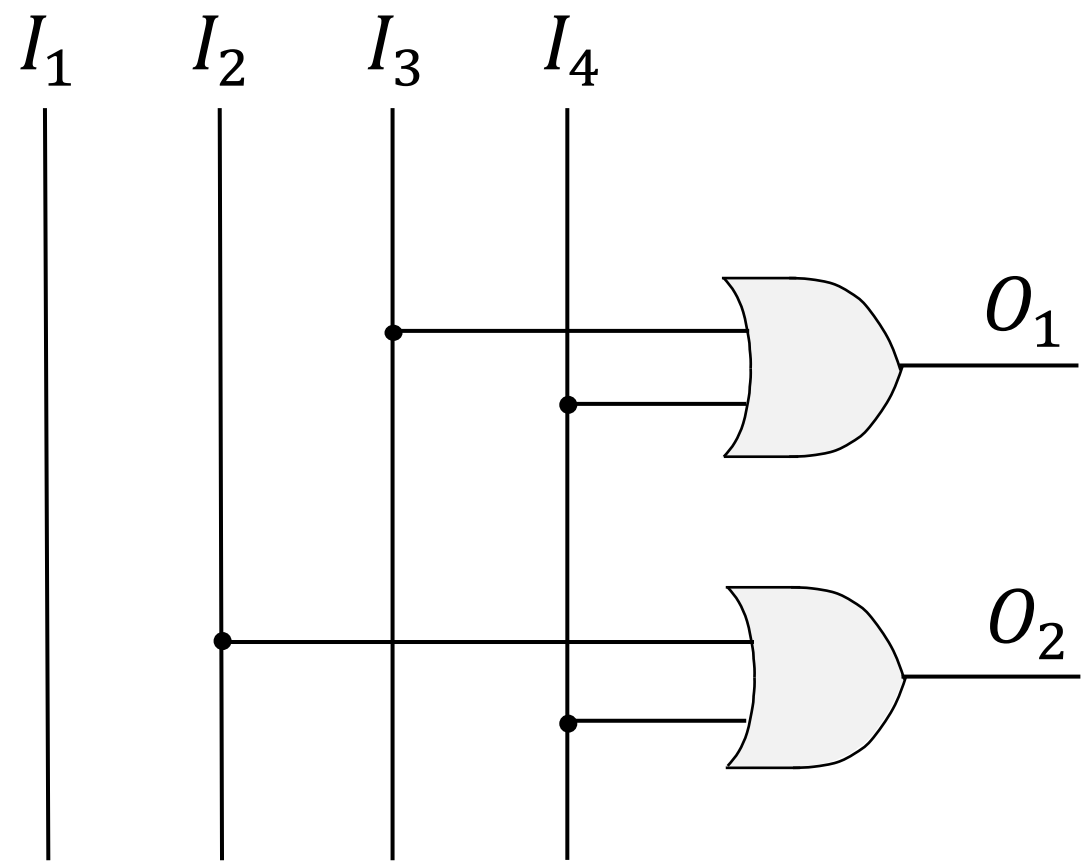
4-to-2 Encoder Implementation



I_1	I_2	I_3	I_4	O_1	O_2
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Assumption by definition:
Exactly one input is set to “1”, while all others are “0”s

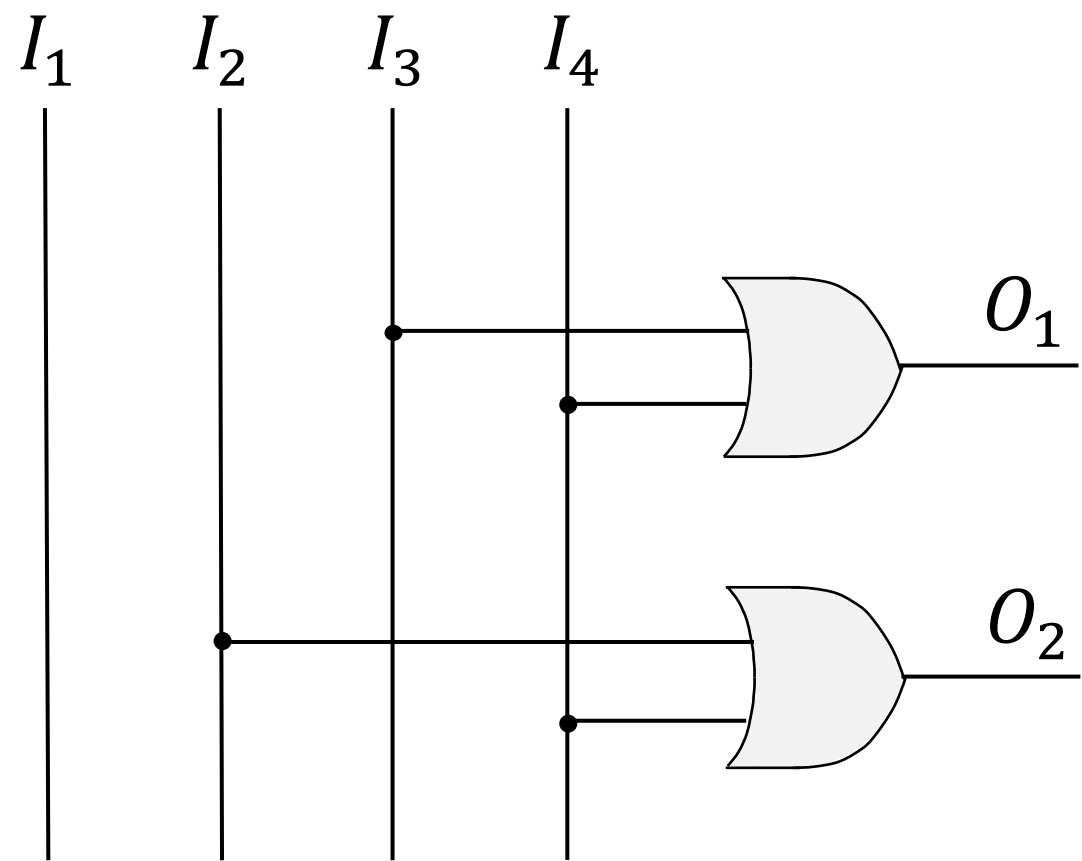
4-to-2 Encoder Implementation



I_1	I_2	I_3	I_4	O_1	O_2
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

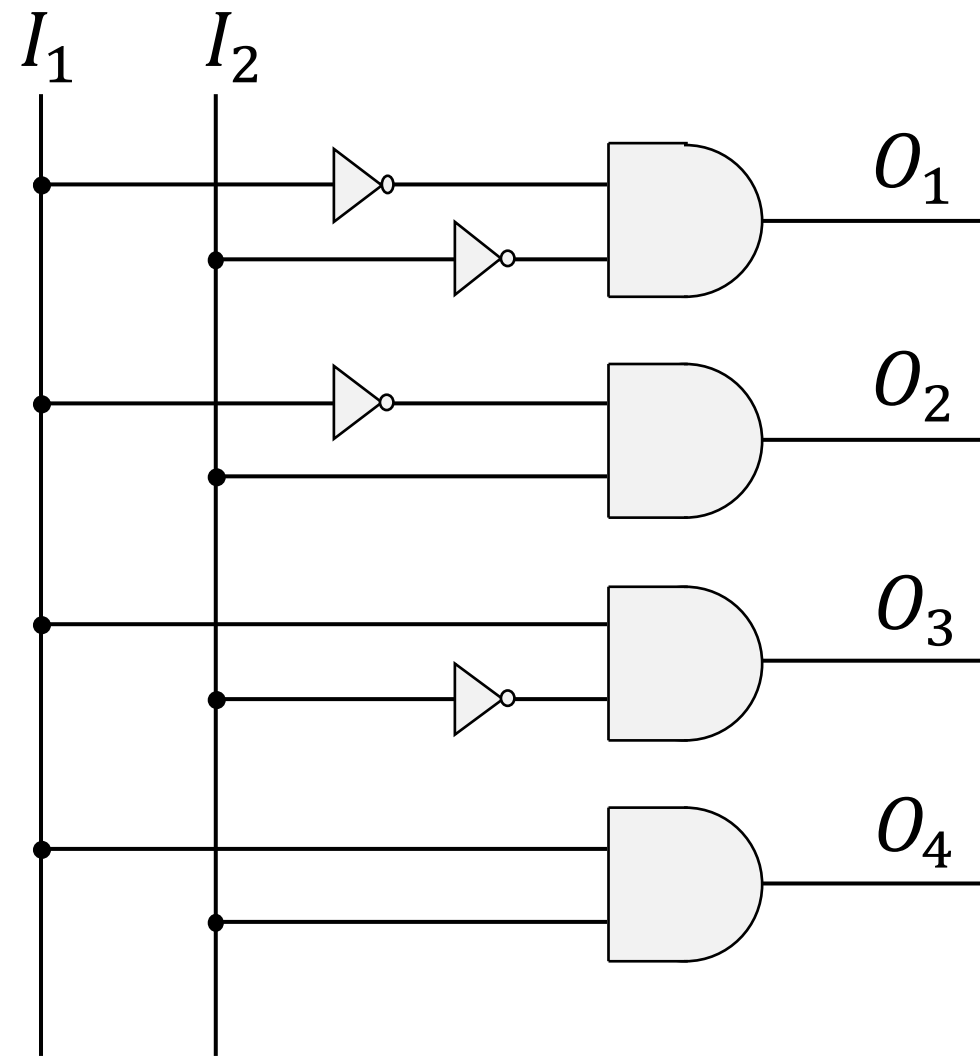
Assumption by definition:
Exactly one input is set to “1”, while all others are “0”s

4-to-2 Encoder



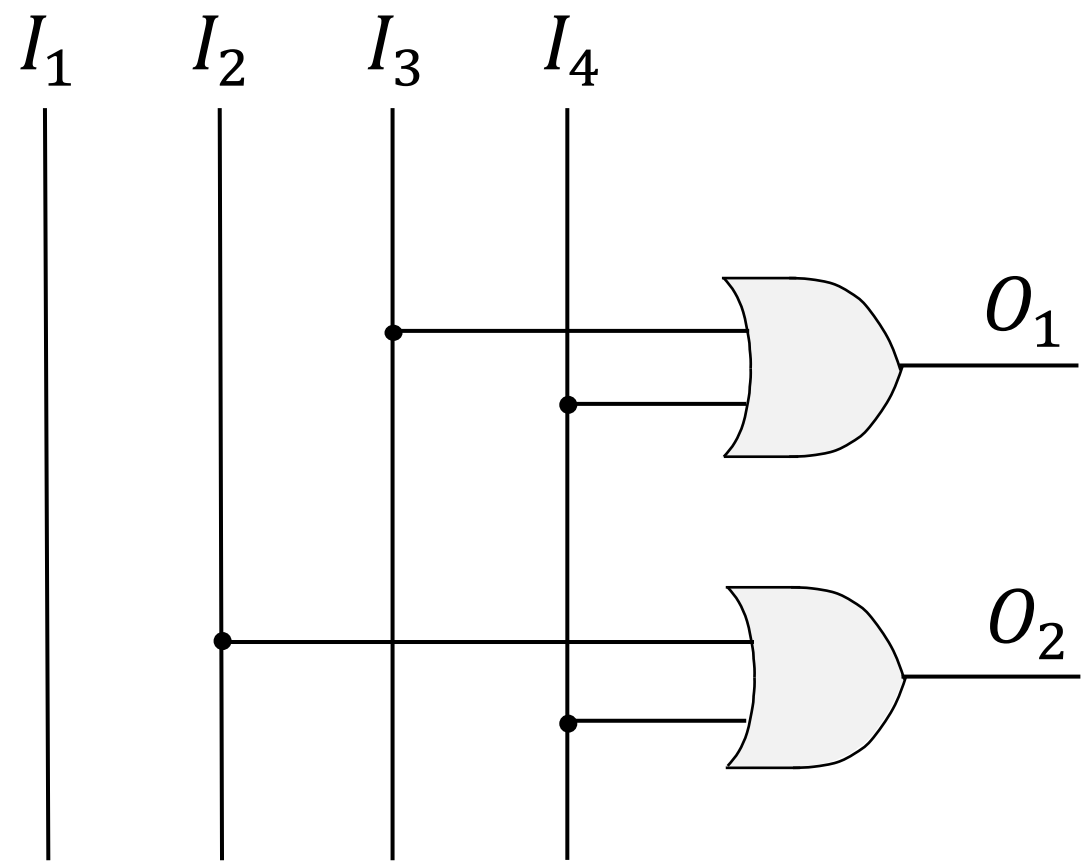
One **input** is set to “1”, while all others are “0”s

2-to-4 Decoder

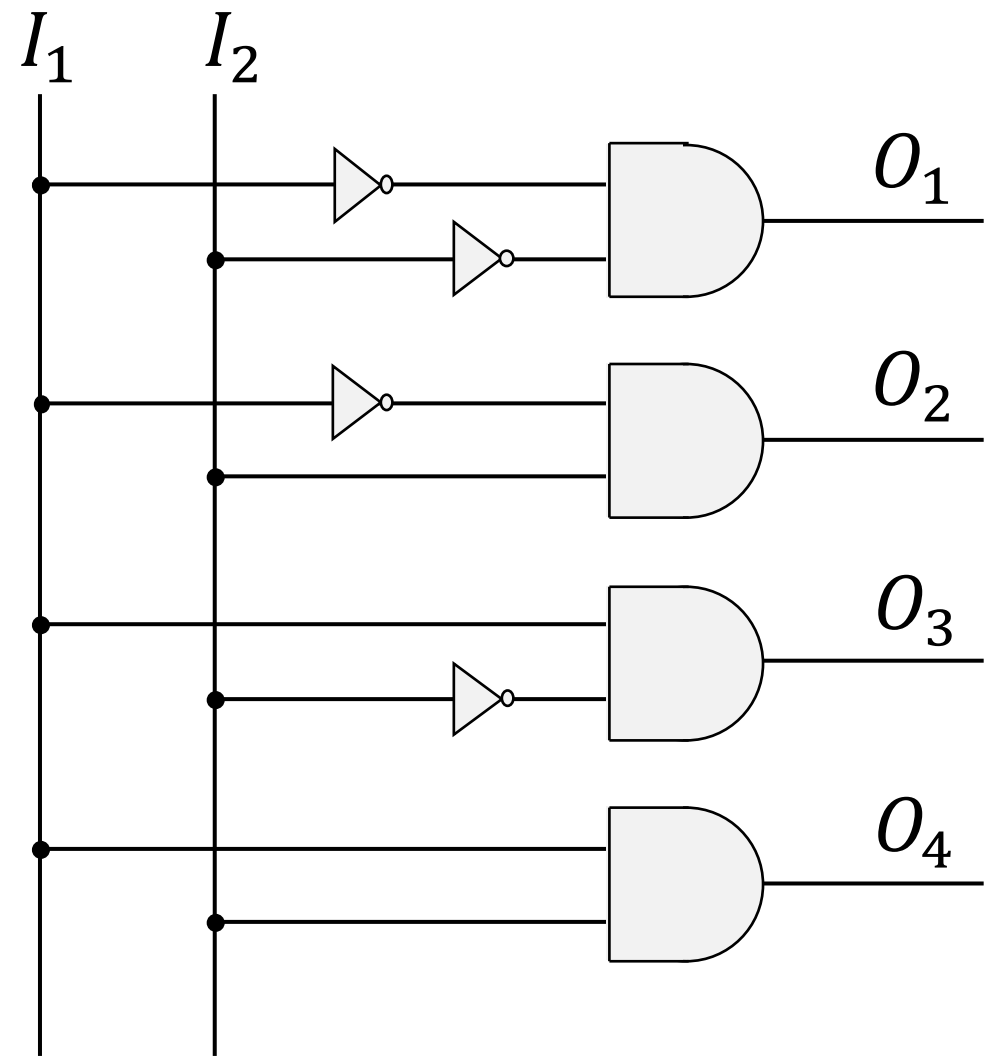


One **output** is set to “1”, while all others are “0”s

4-to-2 Encoder



2-to-4 Decoder

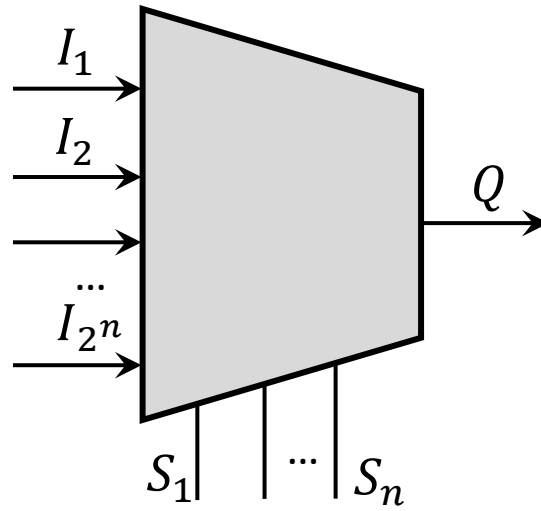


Note: These encoder and decoder circuits have symmetric truth tables.
In practice, the requirement of symmetric truth tables might be omitted

Hardware Building Blocks

Multiplexer

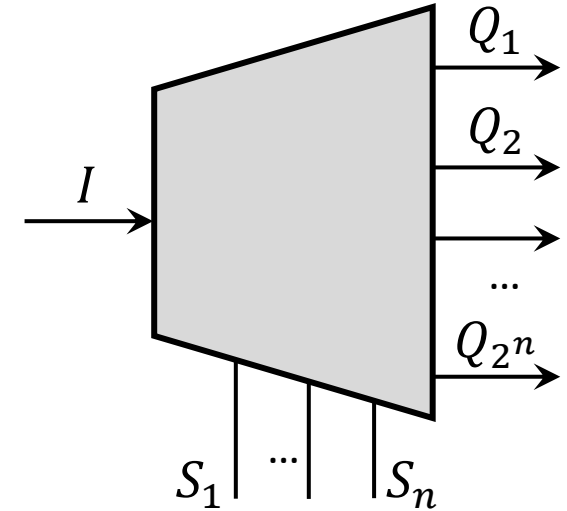
Output Q is set to one of inputs $I_1, \dots, I_{2^n}$, based on selectors $S_1, \dots, S_n$



Demultiplexer

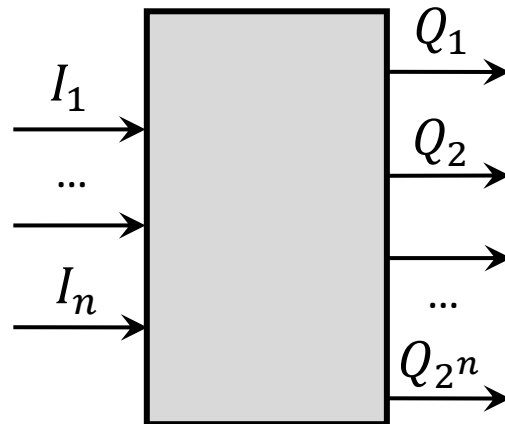
“Flipped” Multiplexer:

Some output Q_x is set to input I , while others Q_i with $i \neq x$ are set to “0”s



Decoder

Based on inputs $I_1, \dots, I_{2^n}$, some output Q_x is set to “1”, while others Q_i with $i \neq x$ are set to “0”s

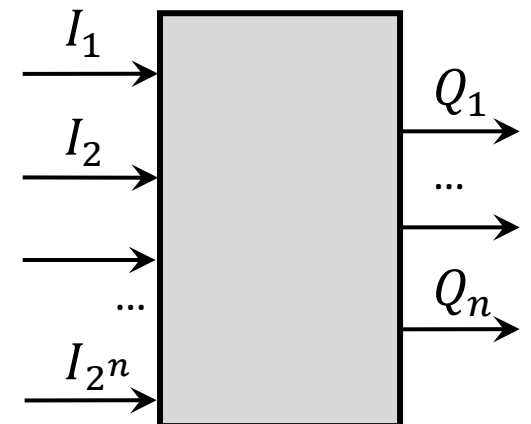


Encoder

“Flipped” Decoder:

Only one input in $I_1, \dots, I_{2^n}$ is allowed to be “1”;

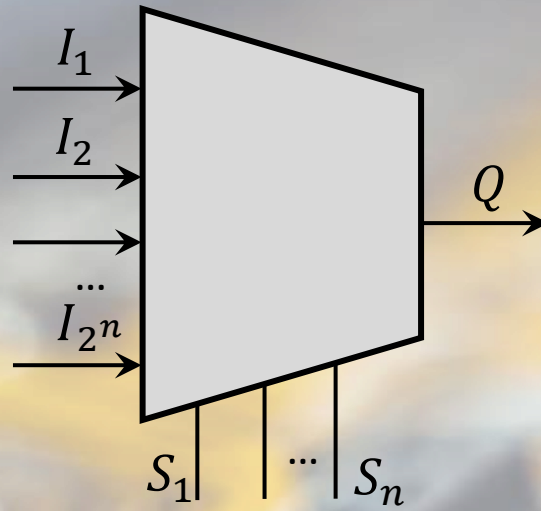
For given inputs, there is a pre-defined set of outputs $Q_1, \dots, Q_n$



Hardware Building Blocks

Multiplexer

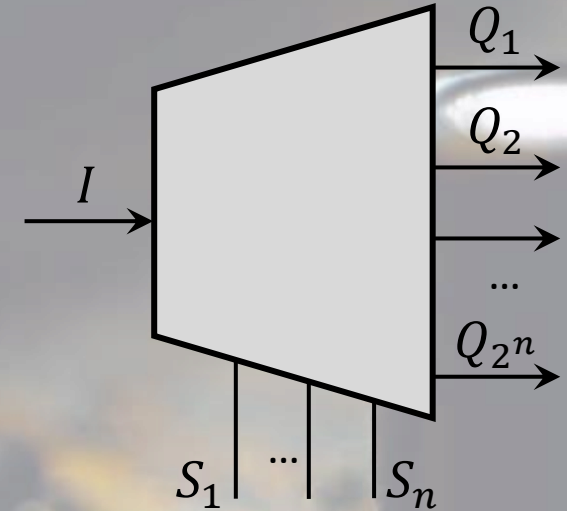
Output Q is set to one of inputs $I_1, \dots, I_{2^n}$, based on selectors $S_1, \dots, S_n$



Demultiplexer

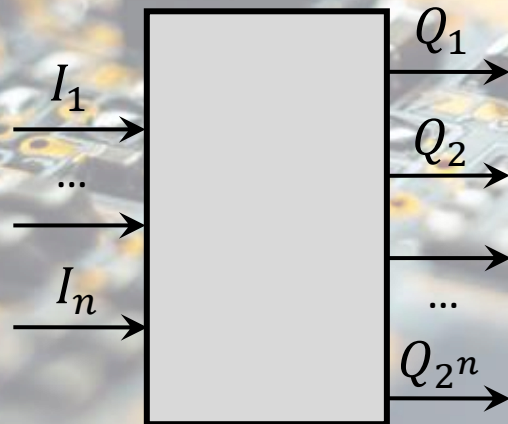
“Flipped” Multiplexer:

Some output Q_x is set to input I , while others Q_i with $i \neq x$ are set to “0”s



Decoder

Based on inputs $I_1, \dots, I_{2^n}$, some output Q_x is set to “1”, while others Q_i with $i \neq x$ are set to “0”s



Encoder

“Flipped” Decoder:

Only one input in $I_1, \dots, I_{2^n}$ is allowed to be “1”;

For given inputs, there is a pre-defined set of outputs $Q_1, \dots, Q_n$

