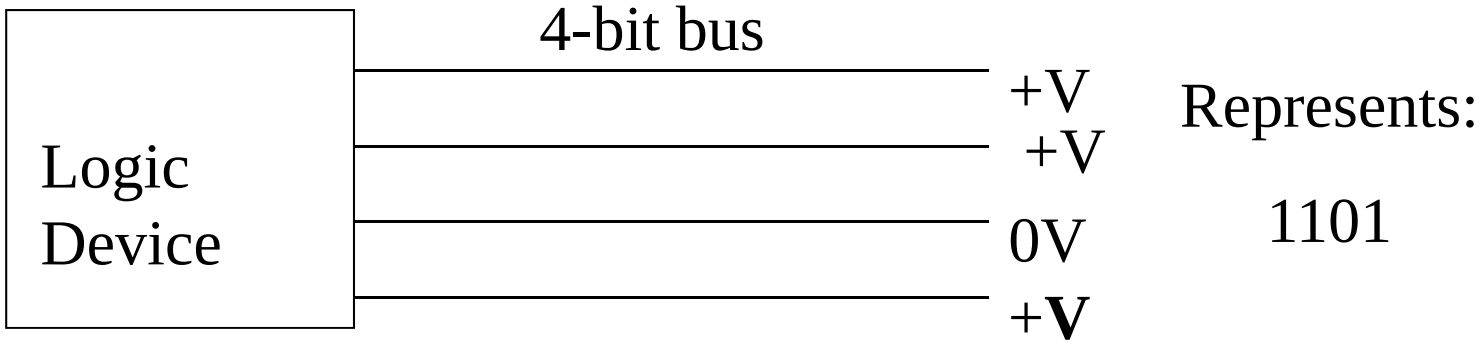
Digital System Blocks

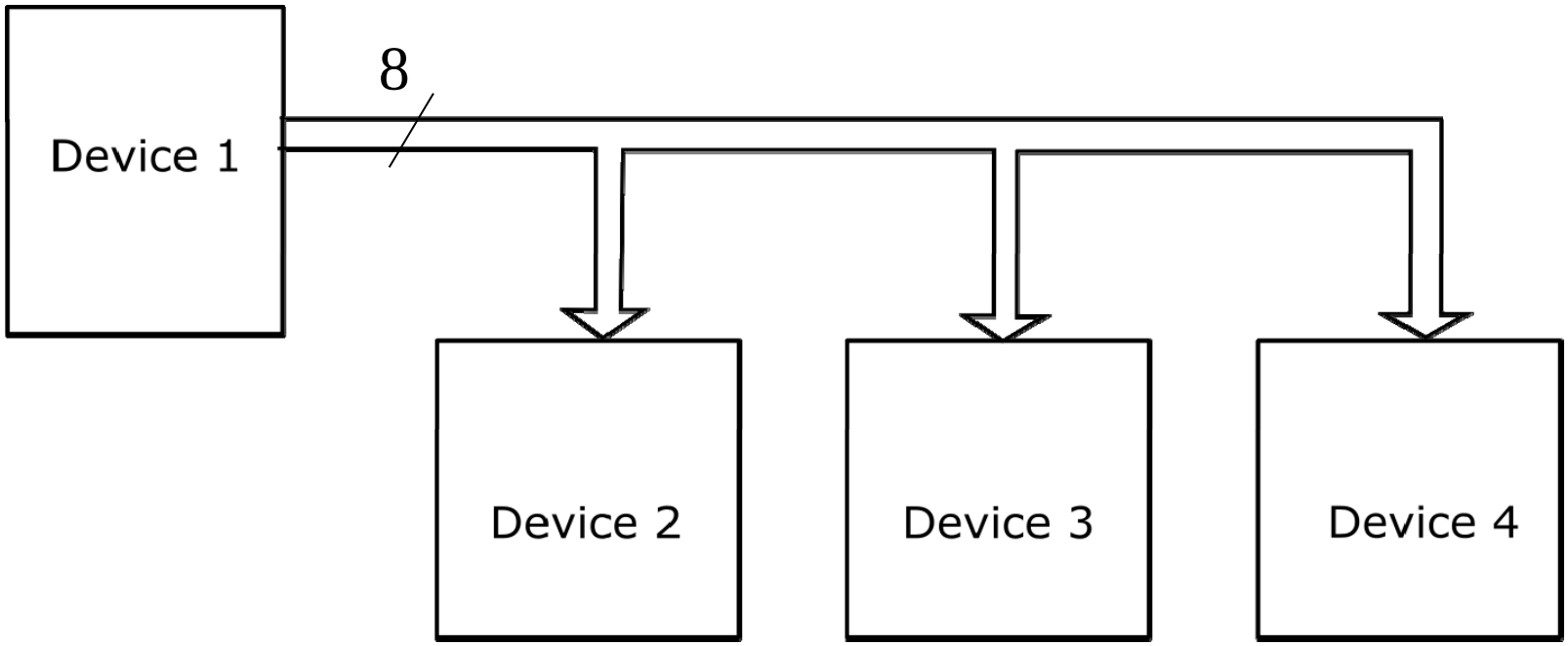
Buses

- Set of two or more electrical conductors representing a binary value



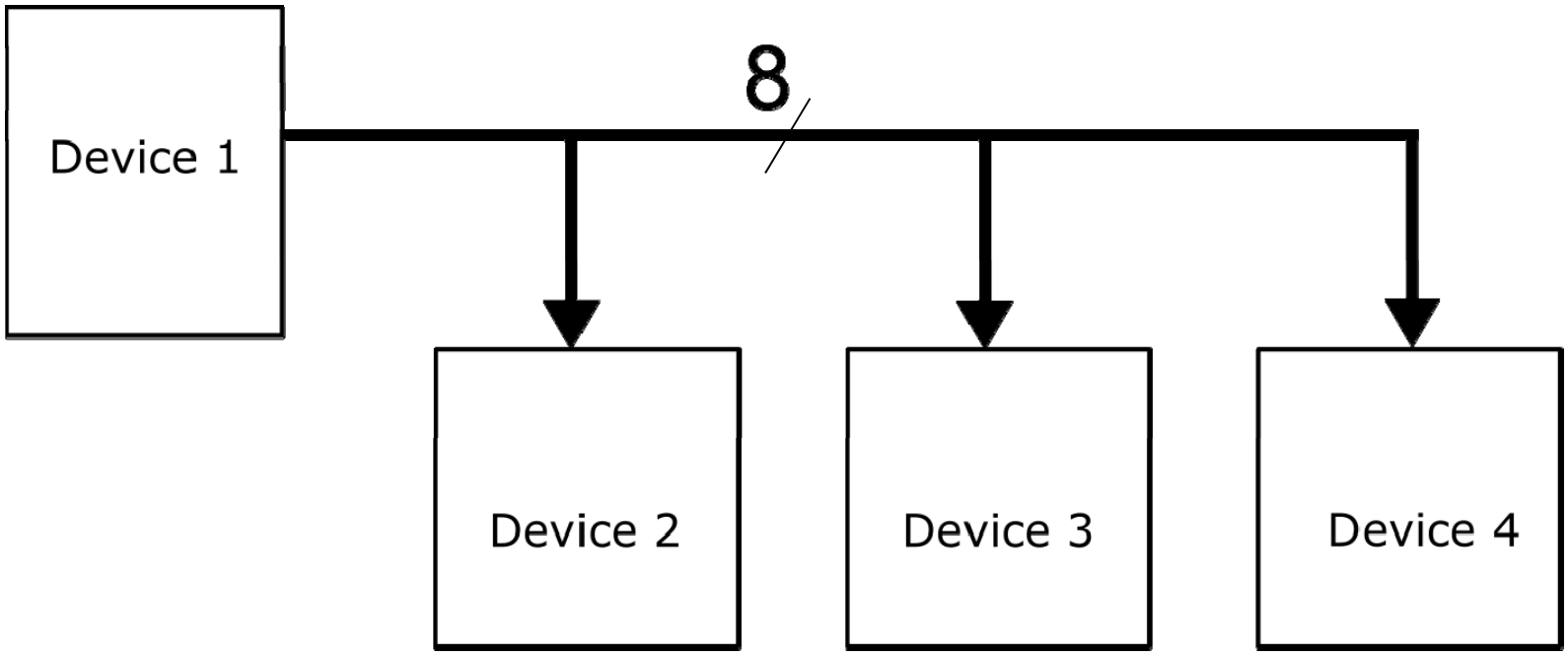
Buses

- Often more than just a one-to-one connection



Buses

- Often more than just a one-to-one connection

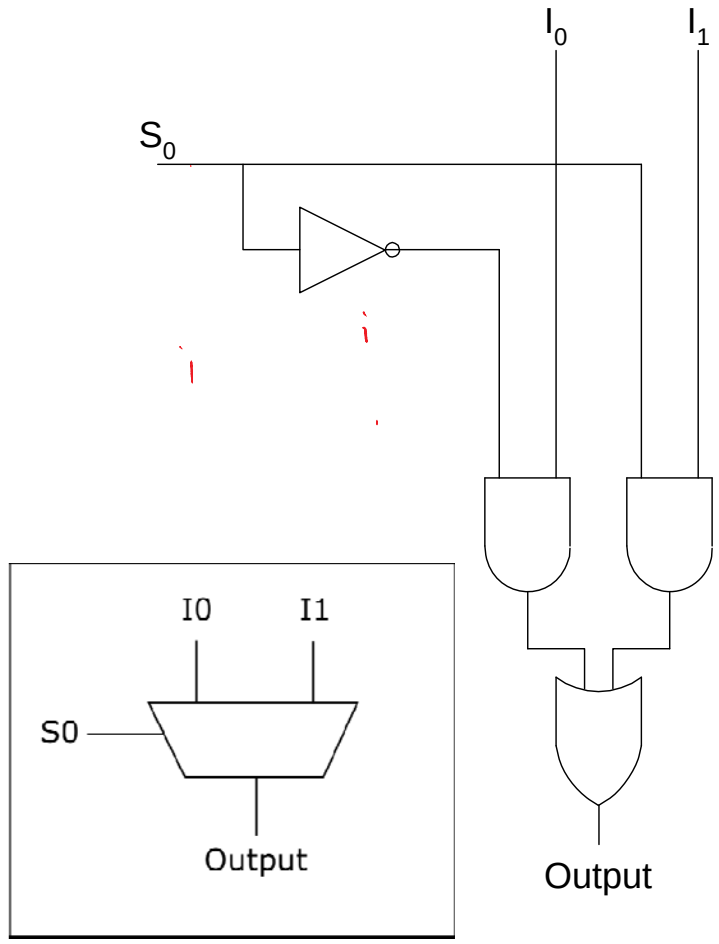


Selectors and Decoders

Selectors or *Multiplexers*

- Often labelled MUX
- 2-Input Selector:

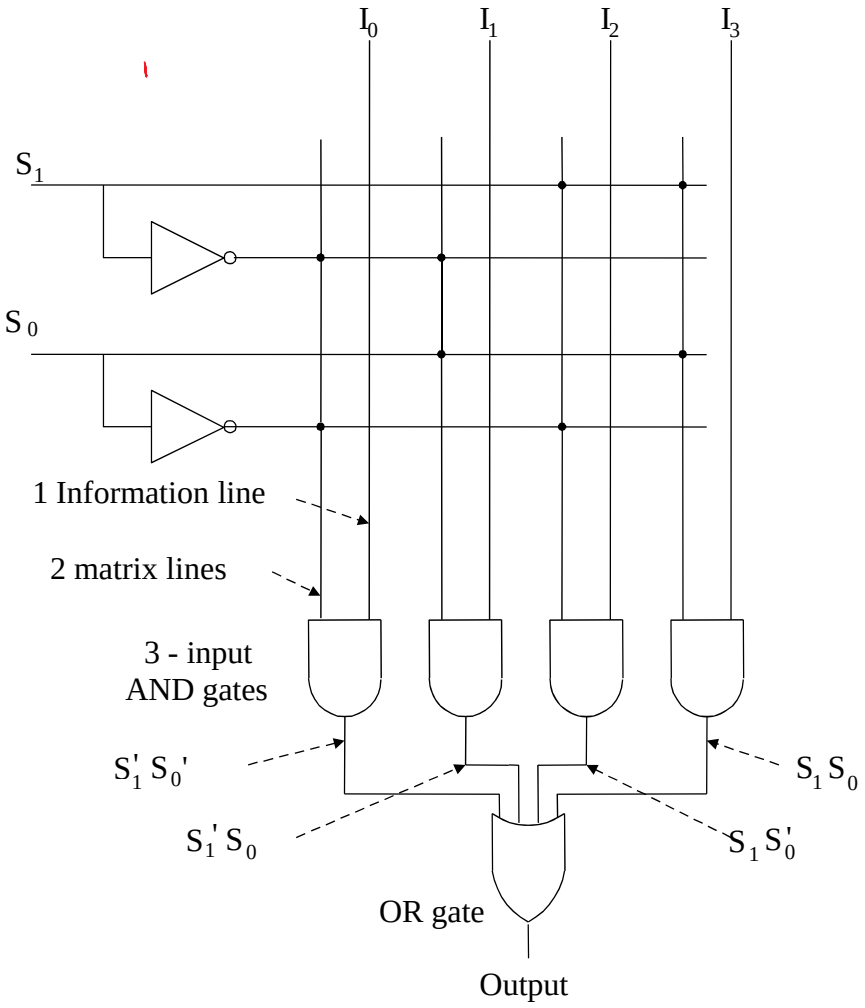
S_0	Output
0	I_0
1	I_1



Selectors or *Multiplexers*

4-input Selector

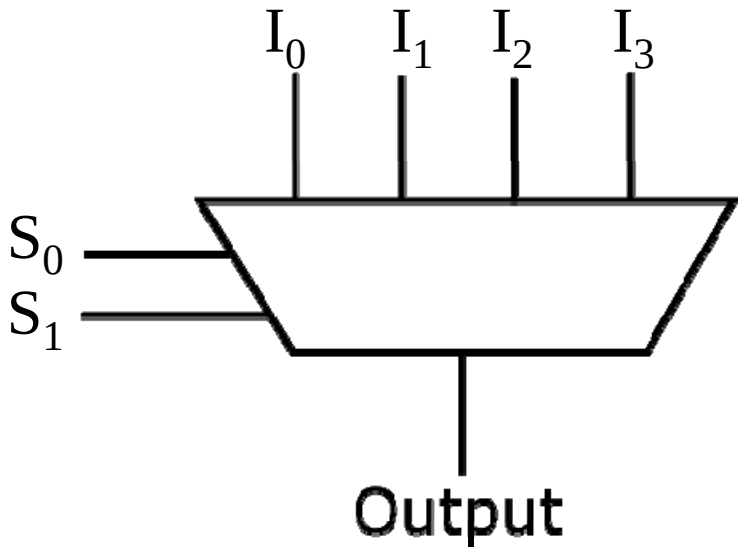
S_1	S_0	Output
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



Multiplexers or Selectors

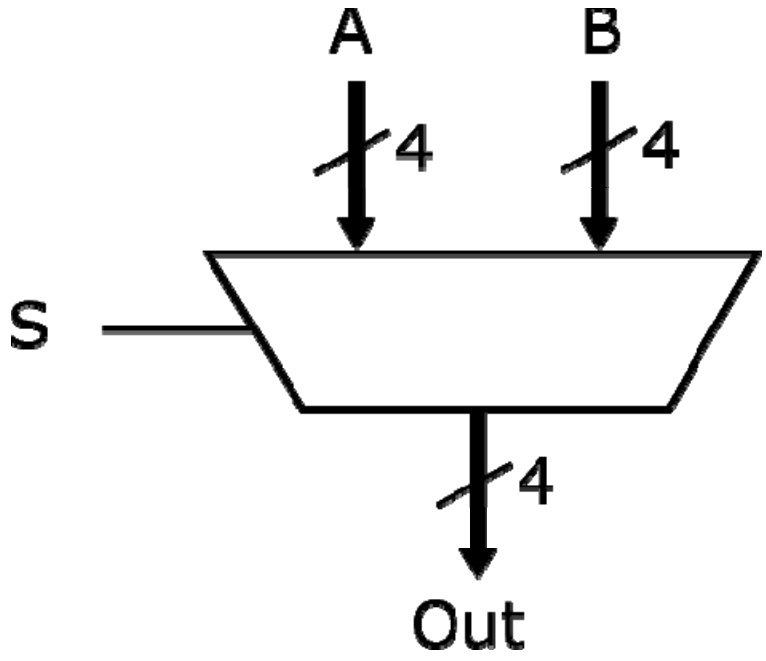
4-input Selector

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



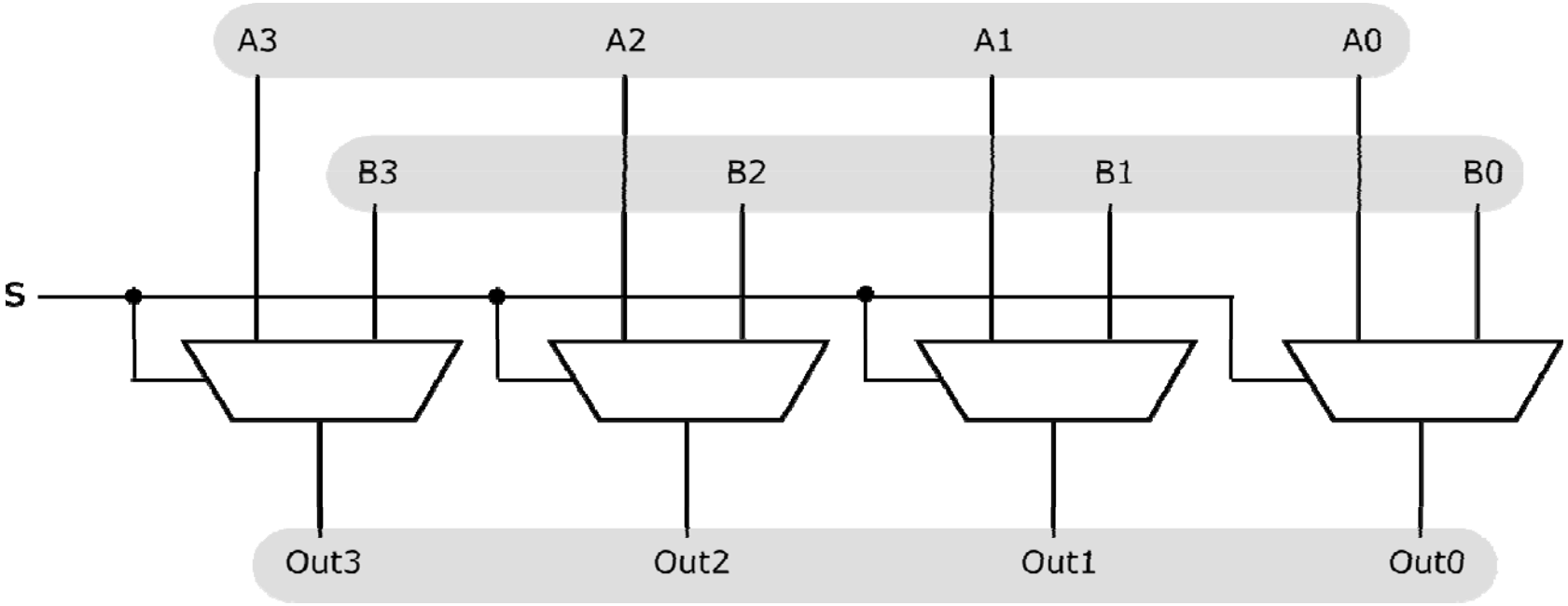
Multiplexers or Selectors

2-Input 4-bit MUX

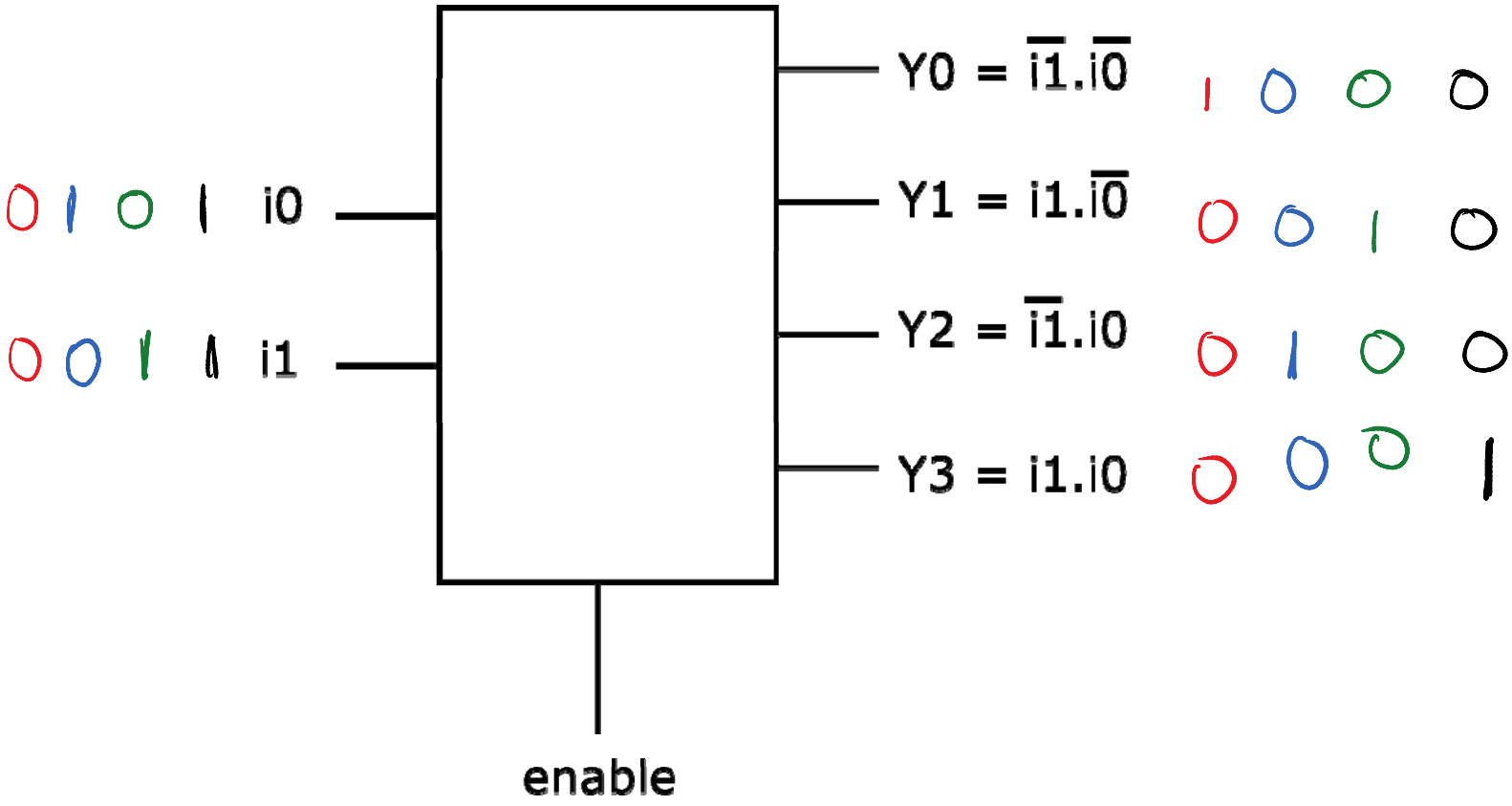


Multiplexers or Selectors

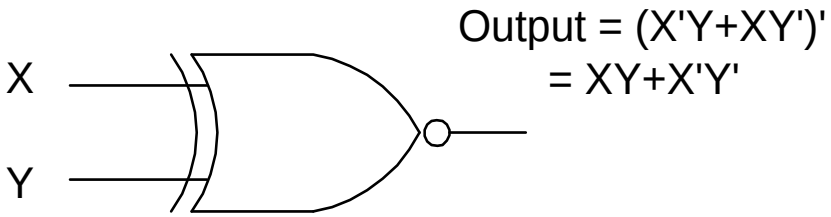
2-Input 4-bit MUX – Internal Implementation



Decoder or *Demultiplexer*



Magnitude Comparator (XNOR)



X	Y	Output
0	0	1
0	1	0
1	0	0
1	1	1

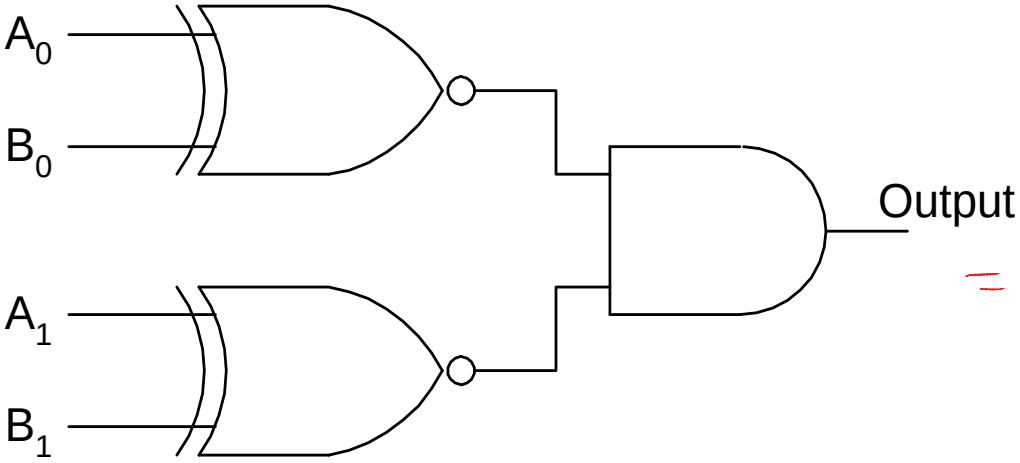
Basic comparator:

- output when X = Y
- complement gives X ≠Y

Other types give:

- X > Y
- X < Y
- complements give:
  - X ≤ Y
  - X ≥ Y

Multiple-bit magnitude comparator



For more than 1-bit comparisons, the XNORs are ANDed together

Exercise: Show that output of above 2-bit comparator is given by:

$$(A_0B_0 + A_0'B_0') \cdot (A_1B_1 + A_1'B_1')$$

Arithmetic Units

Adders - the half-adder

Half-adder:

- accepts two binary digit inputs (X & Y)
- produces Sum (S) & Carry (C) outputs

Arithmetically:

$$\begin{array}{cc} & X \\ & Y \quad + \\ \hline C & S \end{array}$$

		Y = 0	Y = 1
X	0	0	1
	1	1	0

Sum

$$S = X\overline{Y} + \overline{X}Y = X \oplus Y$$

Truth table:

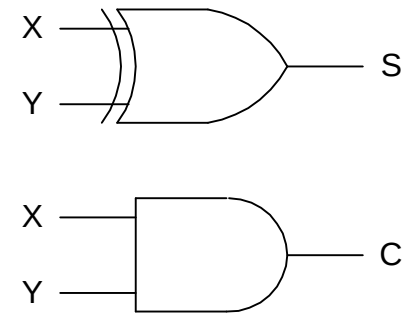
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

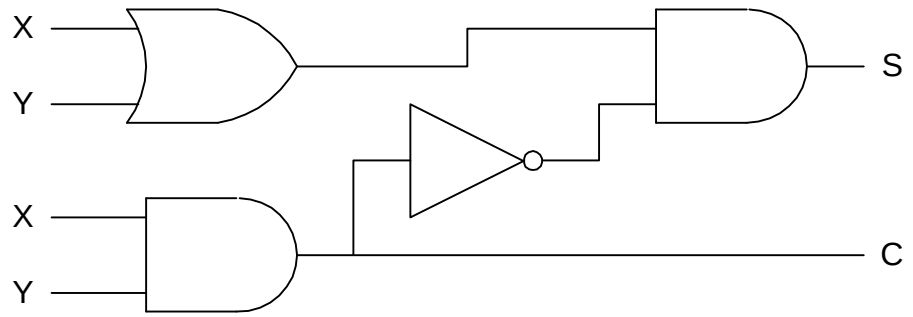
		Y = 0	Y = 1
X	0	0	0
	1	0	1

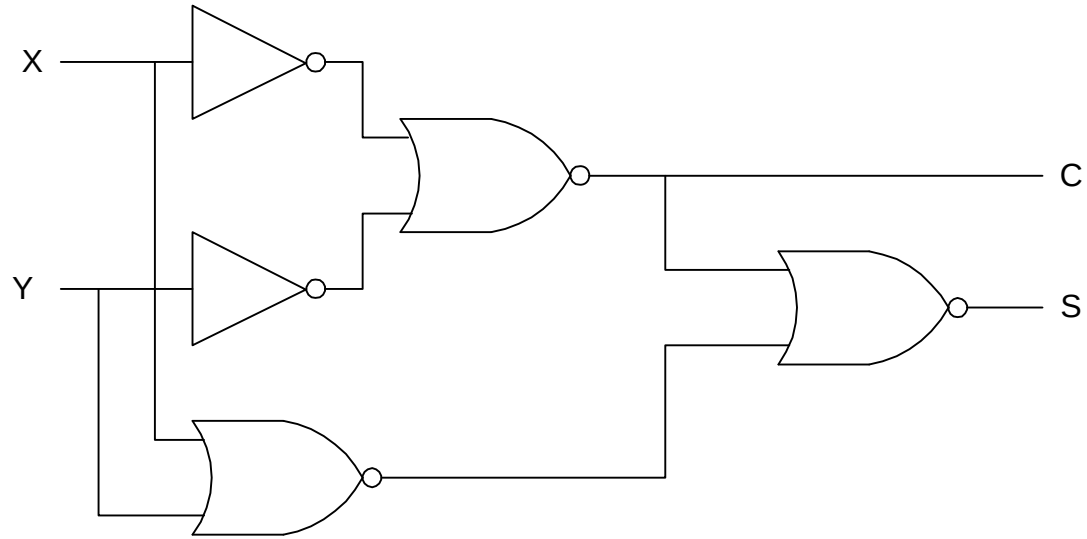
Carry

$$C = XY$$

Examples of half-adder implementations







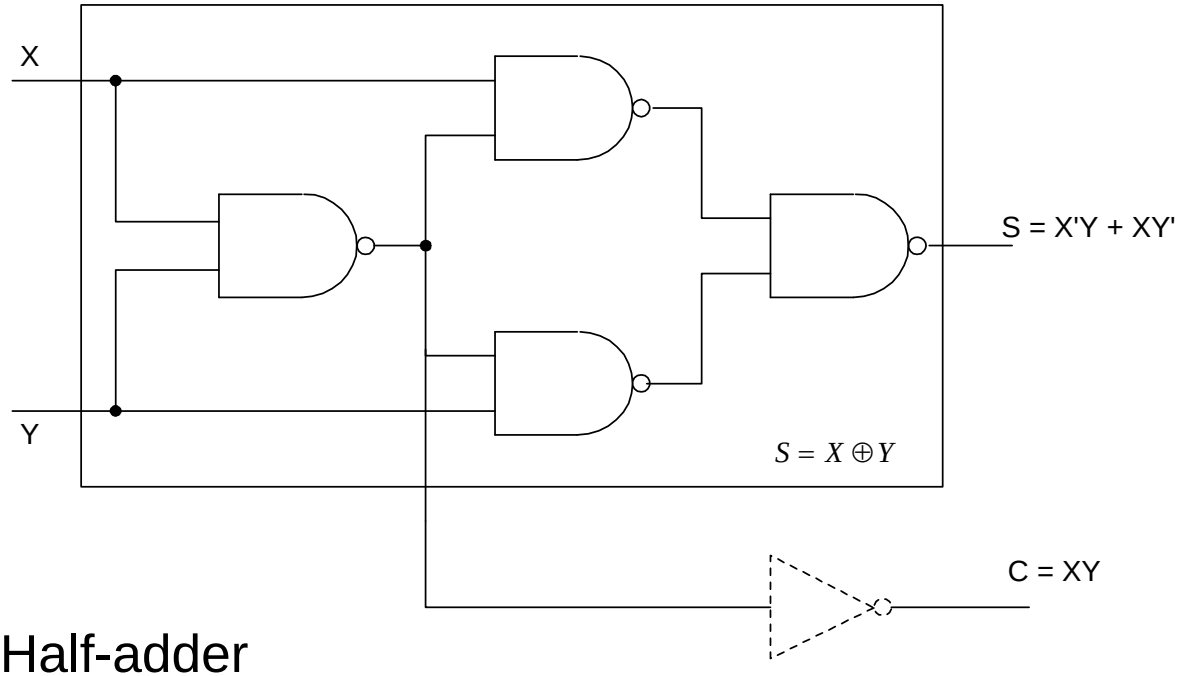
Exercise:

Show that for all circuits:

$$S = X\overline{Y} + \overline{X}Y$$

$$C = X \cdot Y$$

Half-adder NAND gate implementation



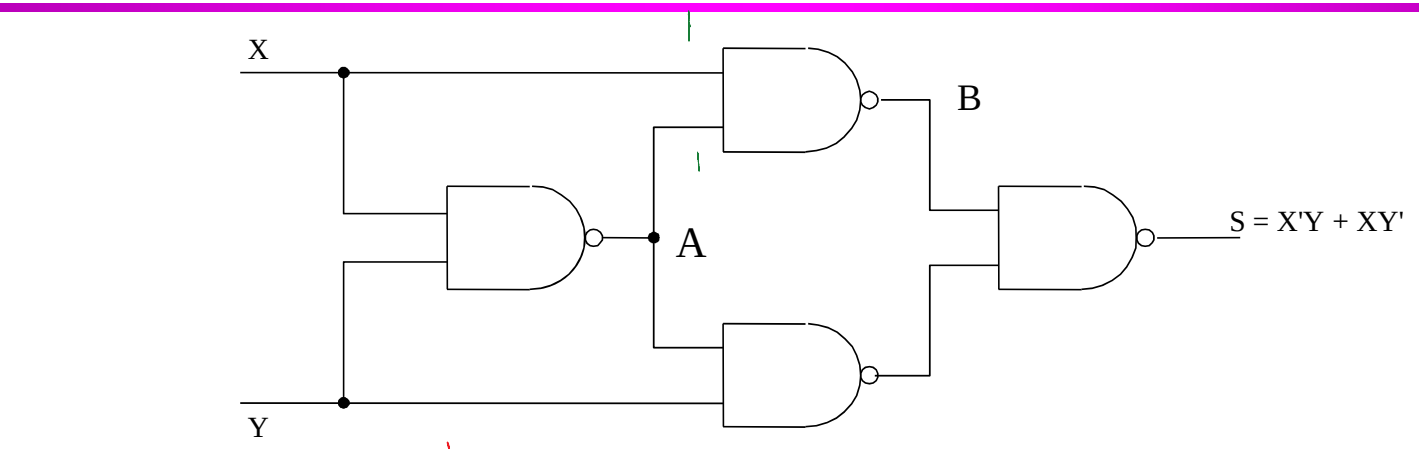
Half-adder

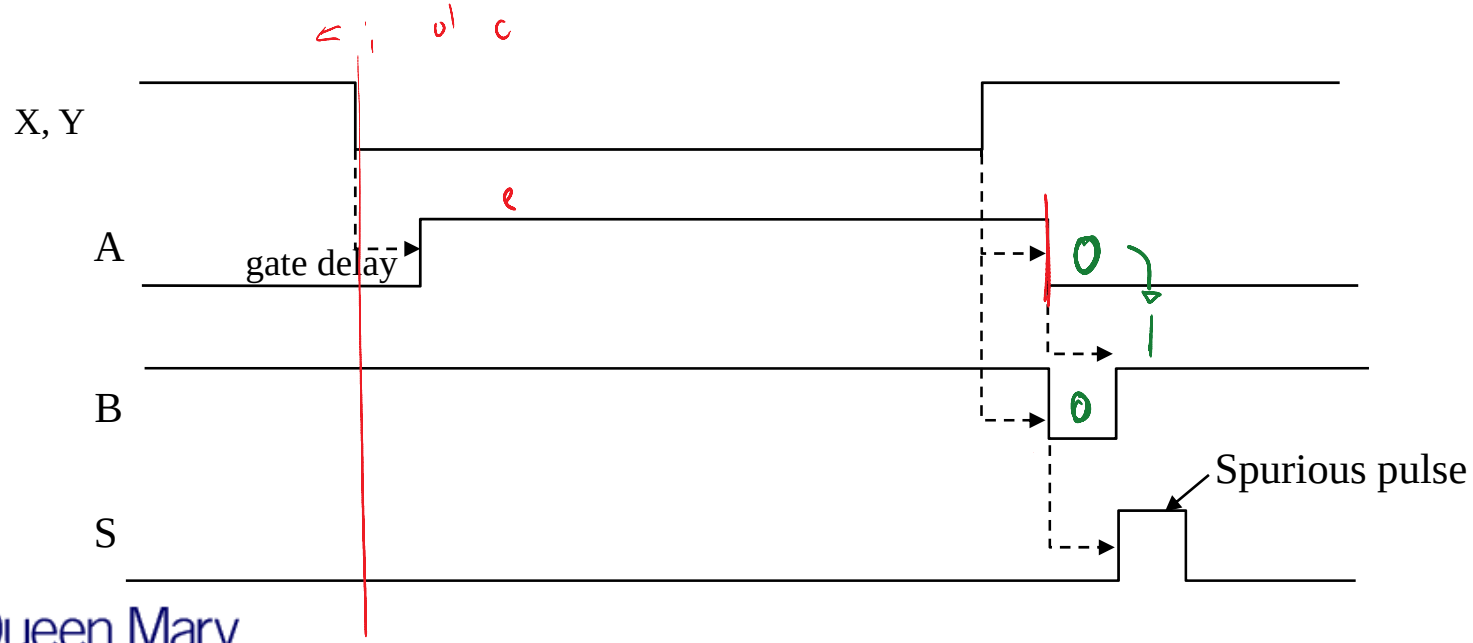
- Arithmetic X + Y
- Max gate delay 3 units

  Sum has 3 units delay

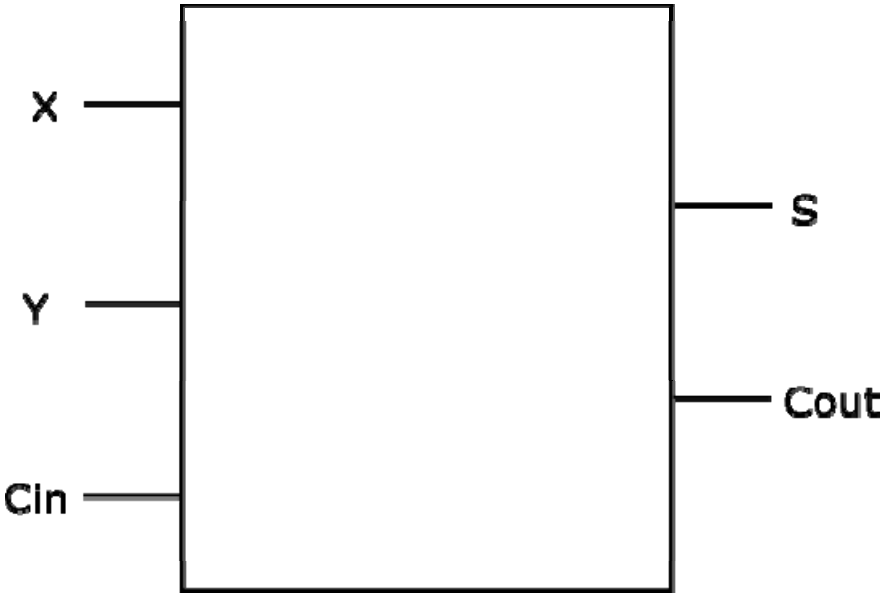
  Carry has 2 units delay (*Carry'* has 1 unit delay)

Effect of Delay





Full Adder



$$s = X \oplus Y \oplus c\_in$$

$$c\_out = (X.Y) + (X.c\_in) + (Y.c\_in)$$

Full Adder

C_{in}	X	Y	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Cout

Cin \ XY	00	01	11	10
0	0	0	1	0
1	0	1	1	1

S

Cin \ XY	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Full Adder

Cout = Cin.X + Cin.Y + X.Y

Cout

Cin \ XY	00	01	11	10
0	0	0	1	0
1	0	1	1	1

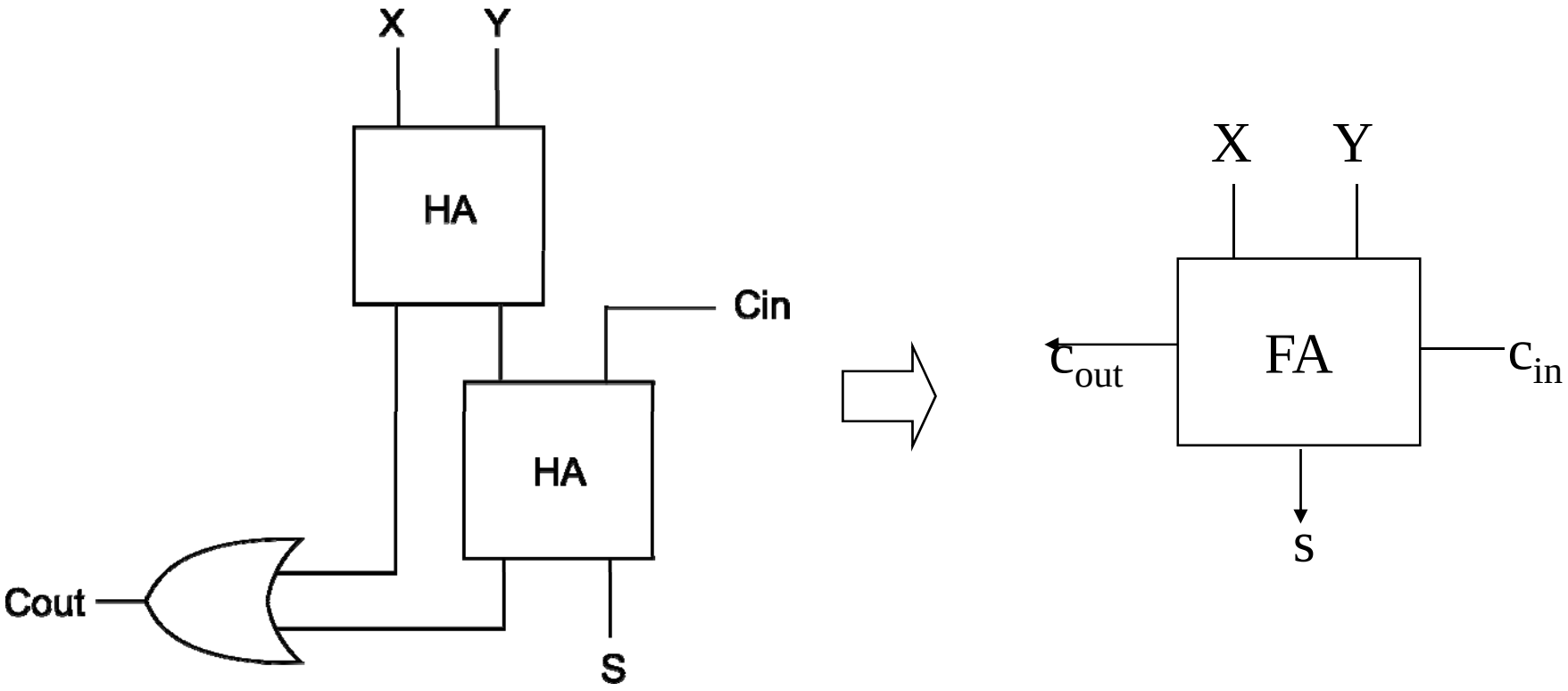
S = Cin'.X'.Y +
Cin'.X.Y' +
Cin.Y'.X' +
Cin.X.Y

S

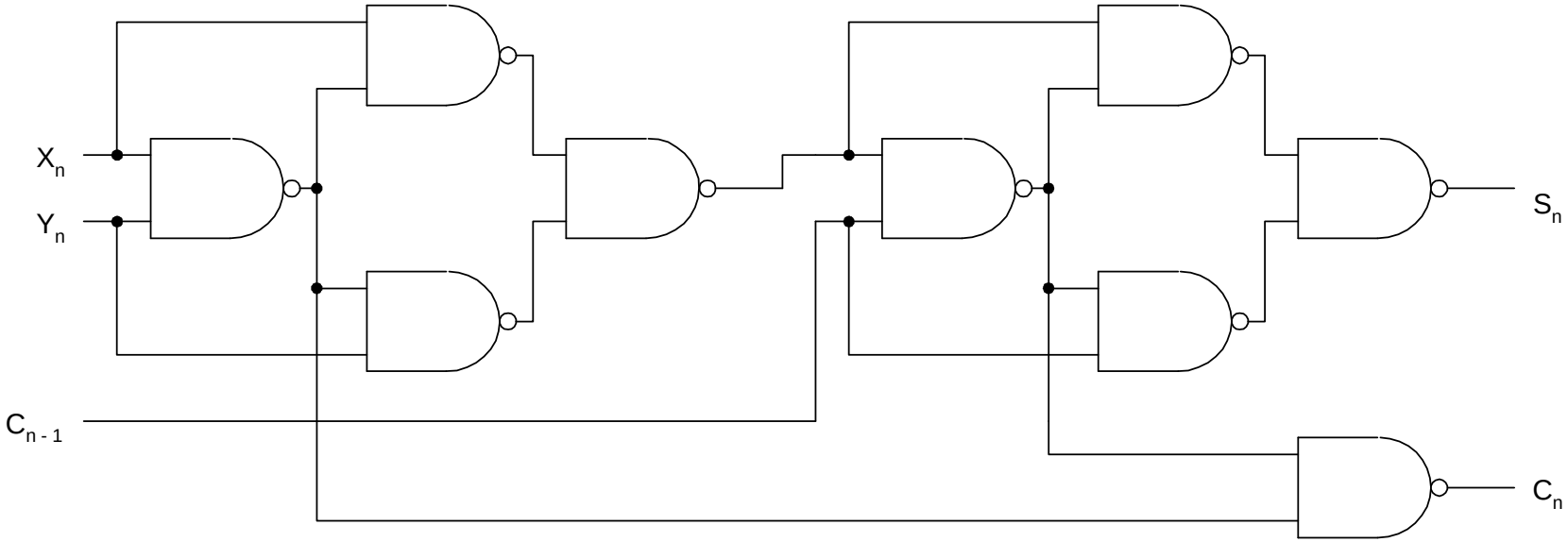
Cin \ XY	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Full Adder gate implementations

Can be implemented with 2 half adders and an OR gate:

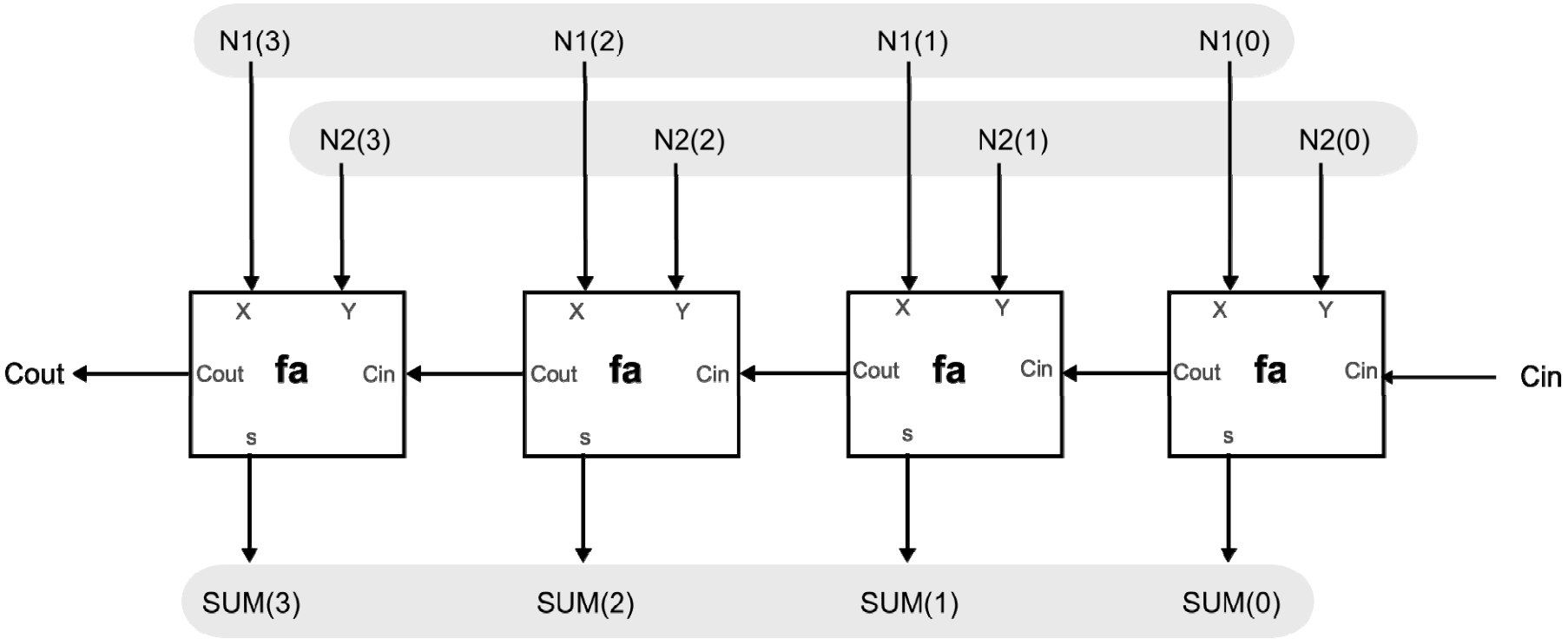


Full Adder: NAND-based Ex-OR blocks

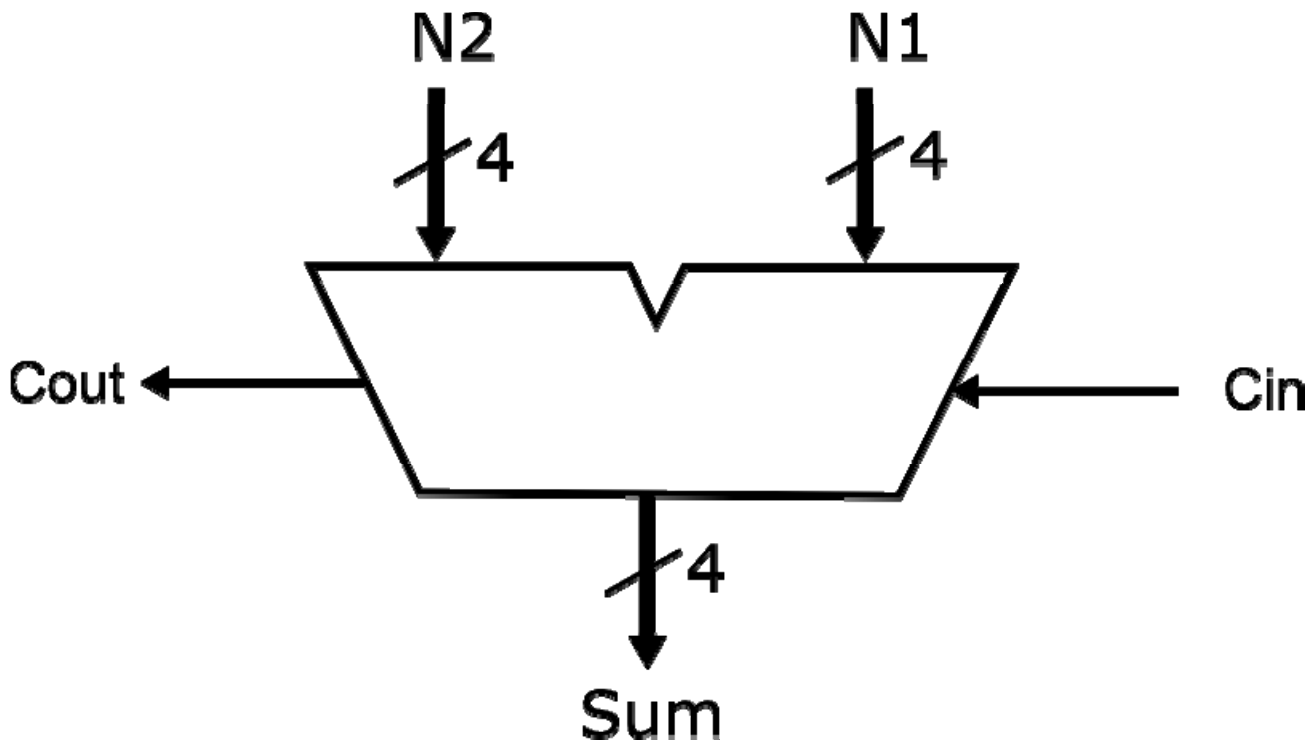


Consists essentially of two half-adders (XORs with Carry outputs)

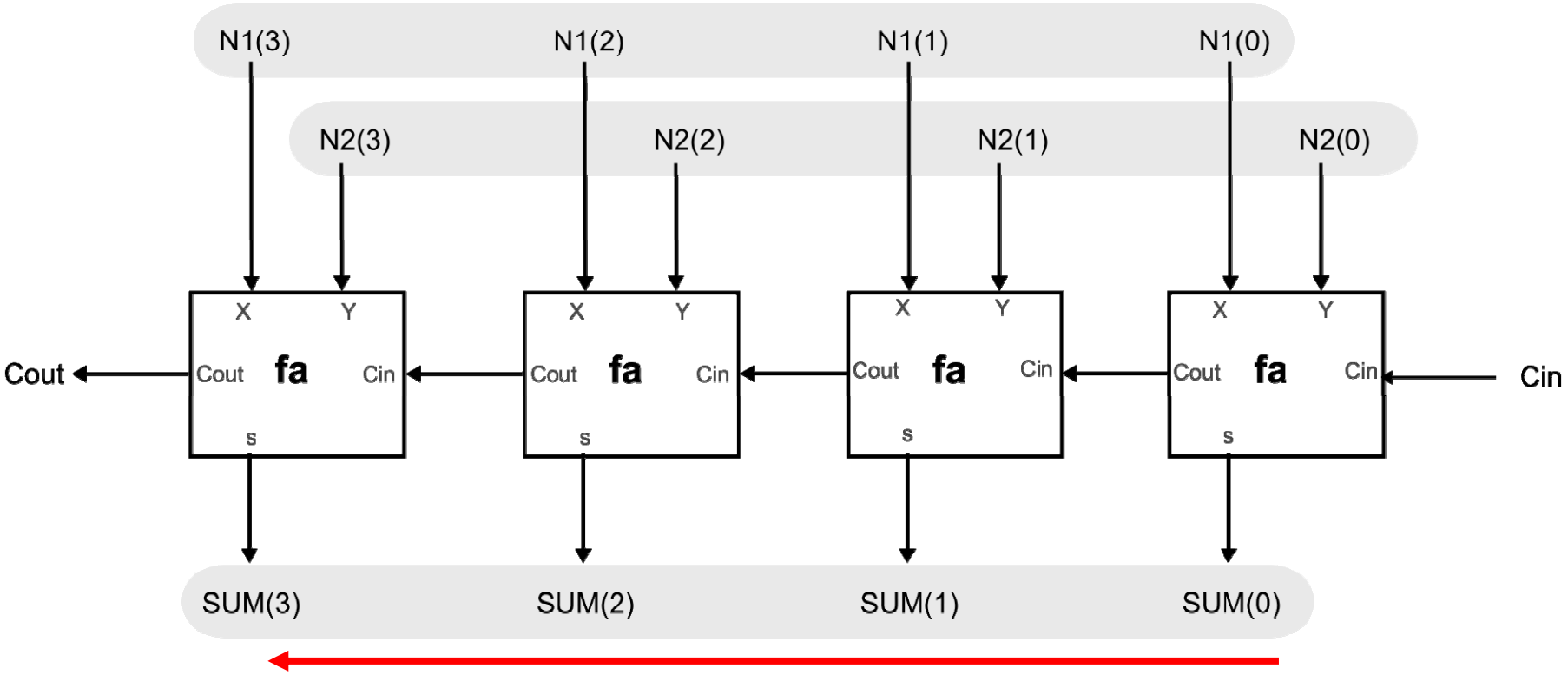
4-bit Parallel Adder



4-bit Parallel Adder

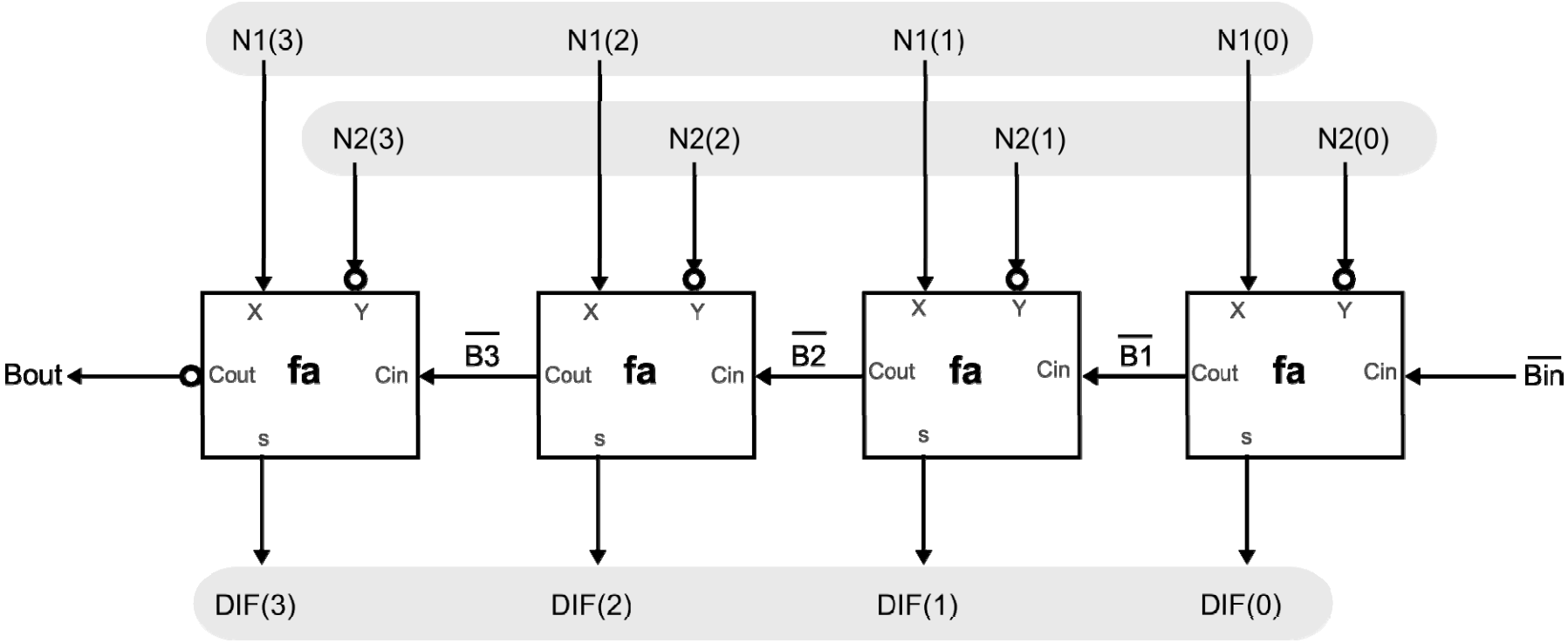


Ripple Delay



Delay caused by carry outputs at each stage

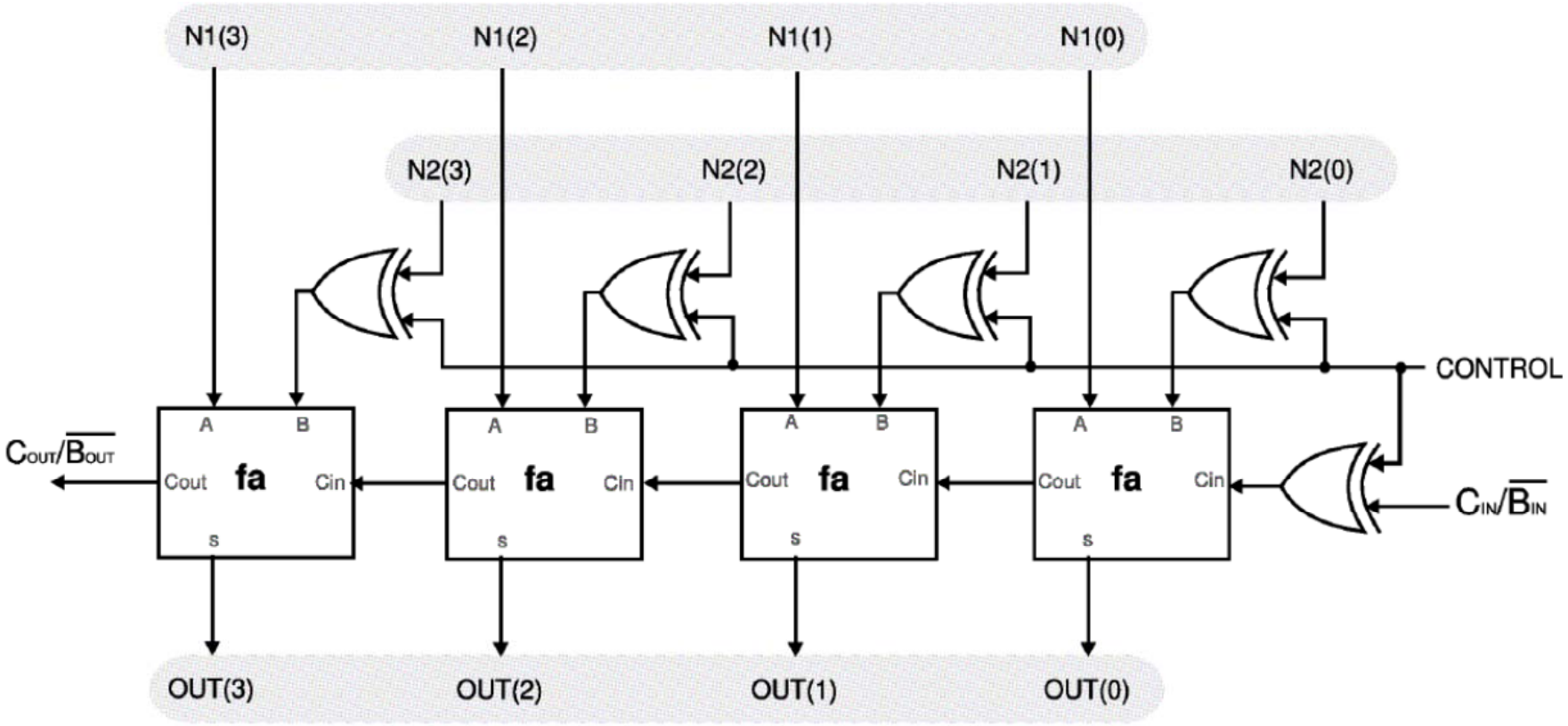
4-bit Ripple Subtractor using Adders



Use two's complement: A-B= A+(B+1)

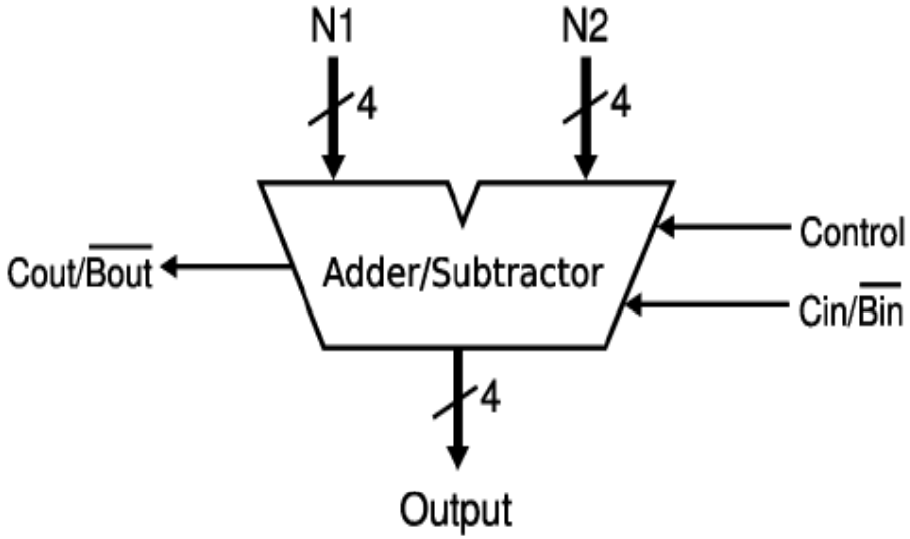
Queen Mary University of London

4-bit Adder/Subtractor



Queen Mary University of London

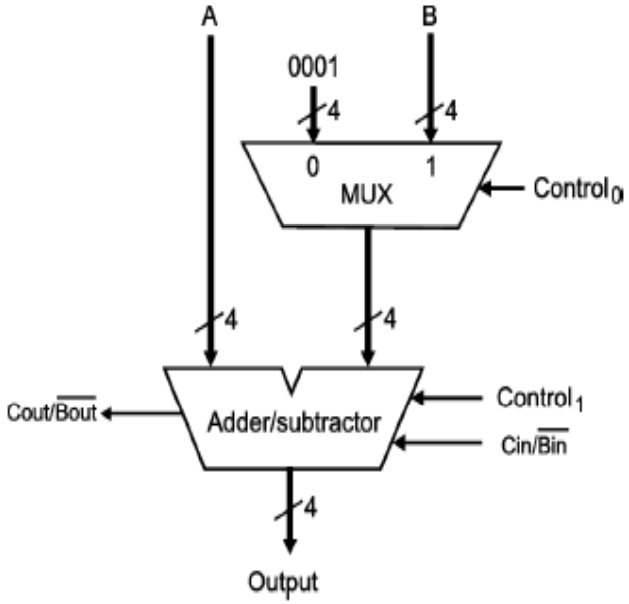
4-bit Adder/Subtractor



Control	Output Function
0	Add (N1 + N2)
1	Subtract (N1 – N2)

An Arithmetic Unit

Control 1	Control 0	Arithmetic Function
0	0	A + 1
0	1	A + B
1	0	A - 1
1	1	A - B



An Arithmetic Unit

Can add and subtract
You can also make it increment and decrement by 1

Control 1	Control 0	Arithmetic Function
0	0	A + 1
0	1	A + B
1	0	A - 1
1	1	A - B