

Computer Architecture
Lecture Week 4 (Part I)

Adder and Subtractor Circuits

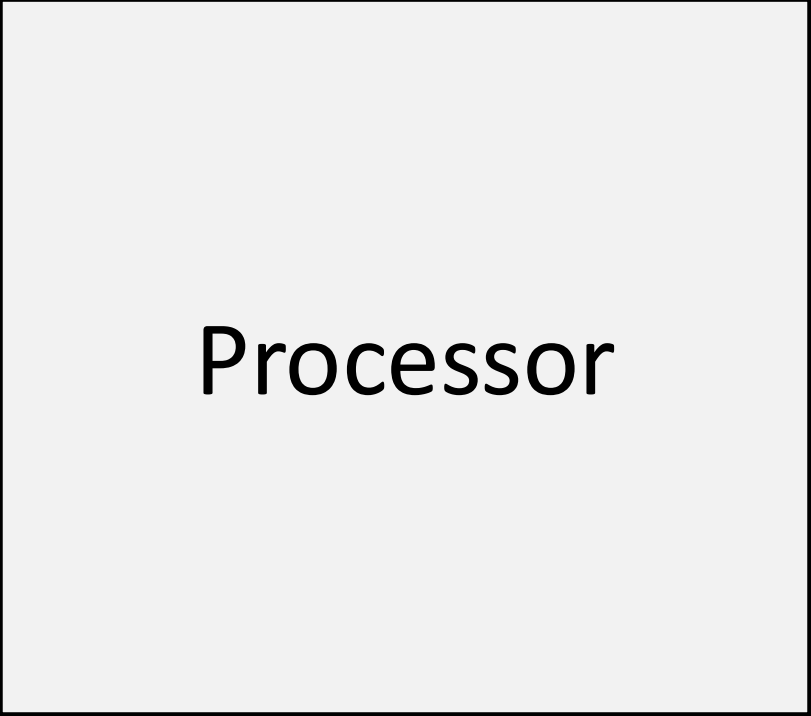
Artem Burmyakov

September 25, 2026



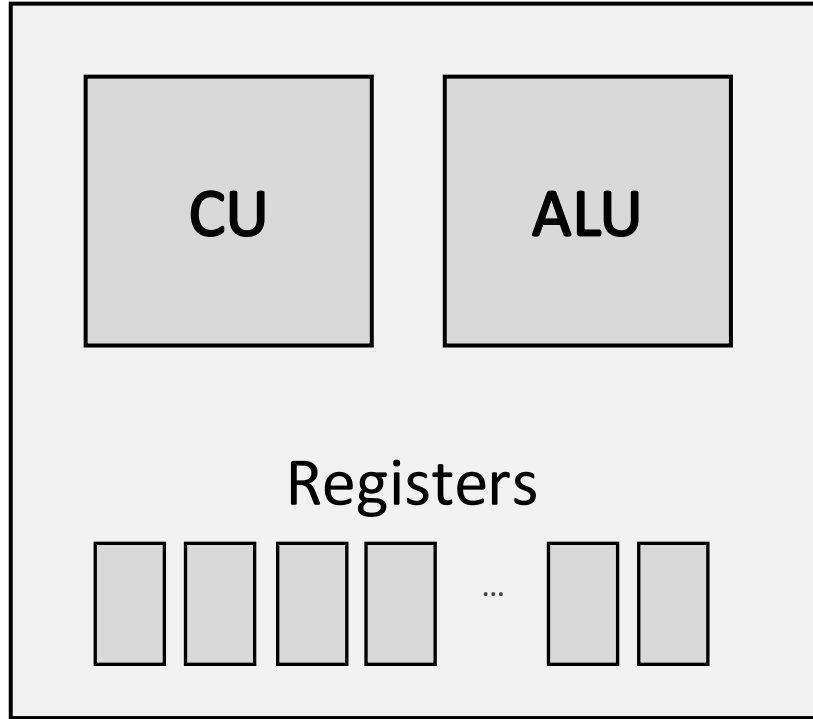
INNOPOLIS
UNIVERSITY

Processor: What are the key components?



Processor

Processor Components

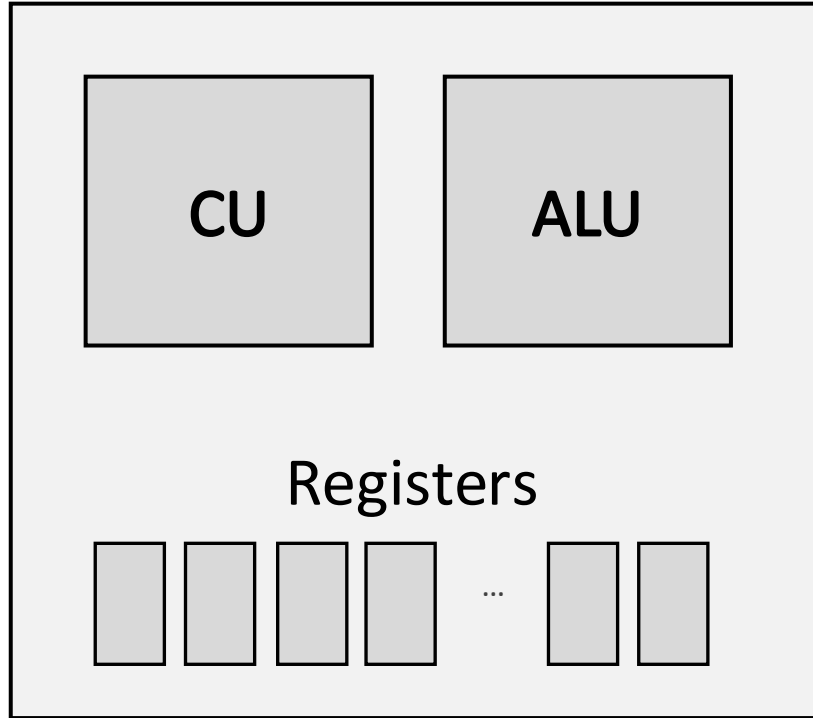


CU: Control Unit

ALU: Arithmetic-Logic Unit

Registers

Processor Components



CU: Control Unit

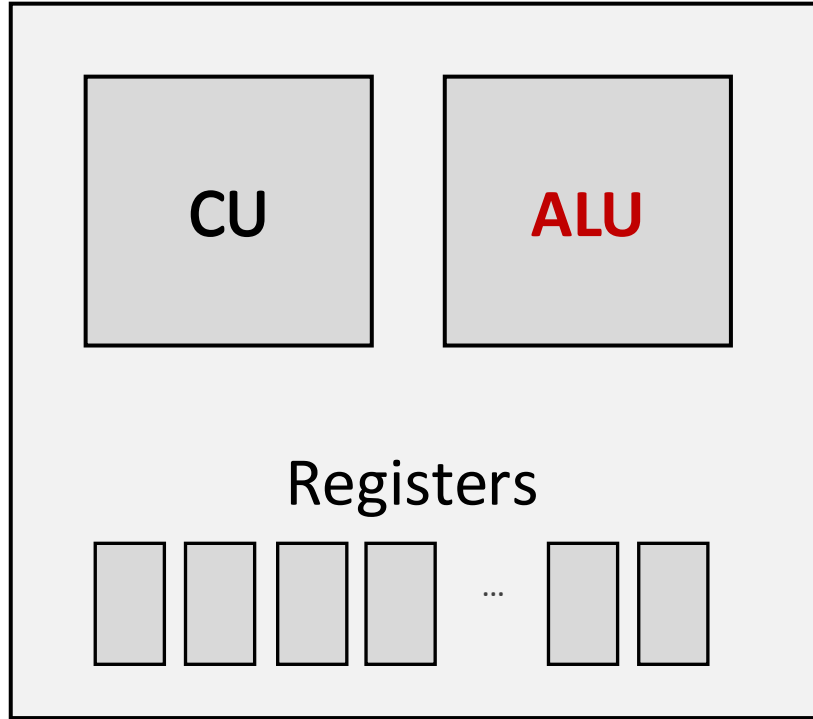
ALU: Arithmetic-Logic Unit

Registers

Question 1:

Which component is the “simplest”?

Processor Components



CU: Control Unit

ALU: Arithmetic-Logic Unit

Registers

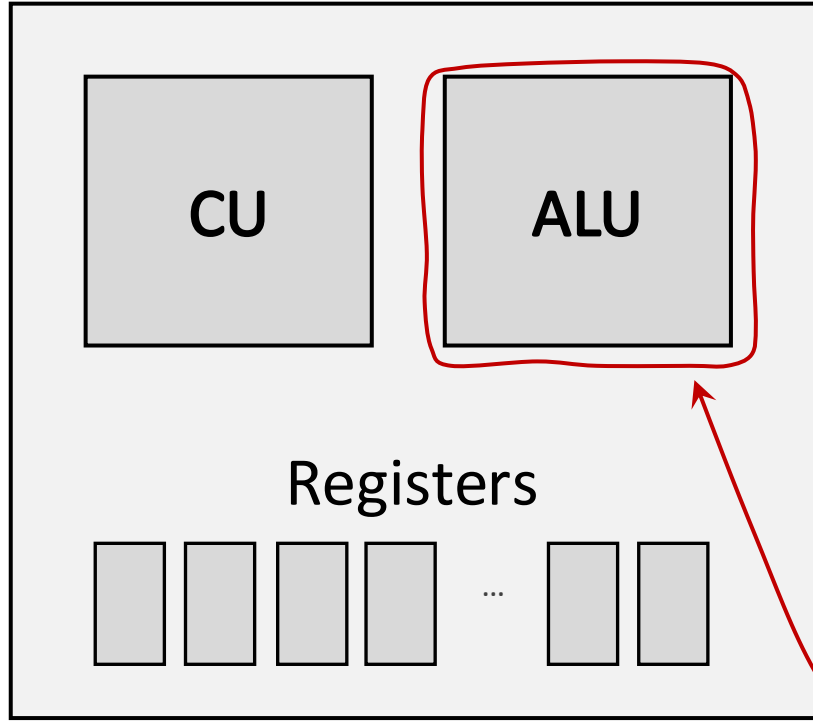
Question 1:

Which component is the “simplest”?

Question 2:

What is the purpose of ALU?

Processor Components



CU: Control Unit

ALU: Arithmetic-Logic Unit

Registers

Question 1:

Which component is the “simplest”?

Question 2:

What is the purpose of ALU?

**“Combinational logic circuit” for
arithmetical and logic operations**

ALU Instruction Example: **Binary Values Addition**

Binary Values Addition

$$\begin{array}{r} + \quad 0 \quad 0 \quad 1 \quad 1 \\ \quad 0 \quad 0 \quad 1 \quad 0 \\ \hline \end{array}$$

Binary Values Addition

Most Significant Bits (MSB)

Least Significant Bits (LSB)

$$\begin{array}{rcccc} & 0 & 0 & 1 & 1 \\ + & 0 & 0 & 1 & 0 \\ \hline \end{array}$$

Binary Values Addition

Most Significant Bits (MSB)

Least Significant Bits (LSB)

$$\begin{array}{r} + \quad \begin{array}{|c|} \hline 0 \\ \hline 0 \\ \hline \end{array} \quad 0 \quad 1 \quad \begin{array}{|c|} \hline 1 \\ \hline 0 \\ \hline \end{array} \\ \hline \end{array}$$



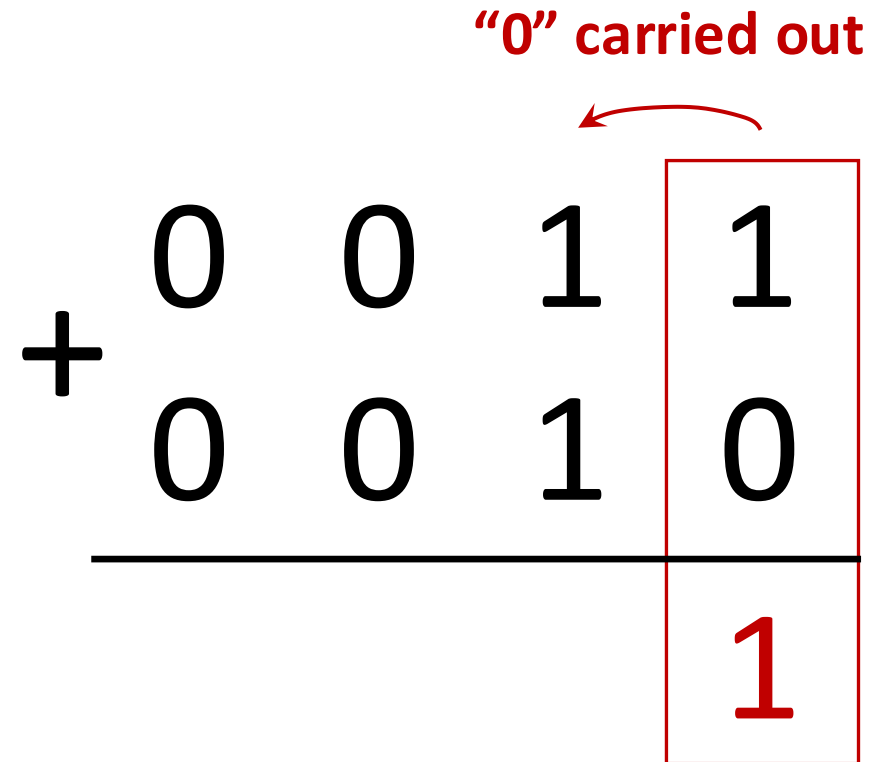
**Computation starts from the least significant bits;
Excess is carried-out**

Binary Values Addition

$$\begin{array}{r} + \quad 0 \quad 0 \quad 1 \quad 1 \\ \quad 0 \quad 0 \quad 1 \quad 0 \\ \hline \quad \quad \quad 1 \end{array}$$

Binary Values Addition

"0" carried out



	0	0	1	1
+	0	0	1	0
				1

Binary Values Addition

"1" carried out

	0	0	1	1
+	0	0	1	0
		0	1	

Binary Values Addition

"1" carried out

The diagram illustrates the addition of two 4-bit binary numbers: 0011 and 0010. The numbers are aligned vertically with a plus sign to the left. A horizontal line separates the operands from the result. The result is shown below the line as 101. A red box highlights the second and third columns of the addition. A red arrow points from the top of the third column (where 1+1=10) to the top of the second column, indicating the carry-out of 1. The result '1' in the second column is also red, while the '0' in the third column and the final '1' in the fourth column are black.

	0	0	1	1
+	0	0	1	0
		1	0	1

Binary Values Addition

"0" carried out

$$\begin{array}{r} + \begin{array}{cccc} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{array} \\ \hline \begin{array}{cccc} & 1 & 0 & 1 \end{array} \end{array}$$

Binary Values Addition

$$\begin{array}{r} + \begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{cccc} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 1 \end{array} \end{array}$$

Binary Values Addition

$$\begin{array}{r} 0011 \\ + 0010 \\ \hline 0101 \end{array}$$

The result:

ALU Instruction Example: Binary Values Addition

Question: How to implement this instruction in ALU?

$$\begin{array}{r} 0011 \\ + 0010 \\ \hline 0101 \end{array}$$

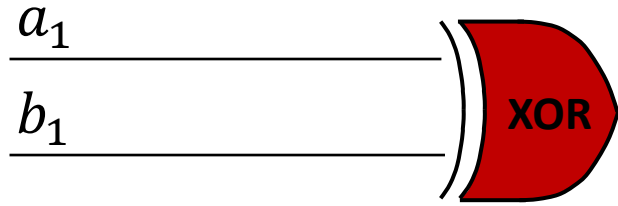
The result:

Adder Circuit Design

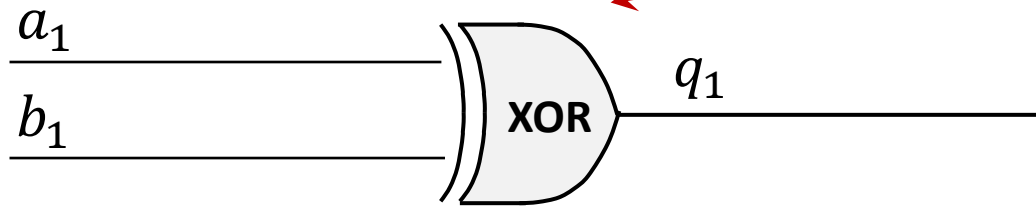
Adder Circuit Design

$$\begin{array}{r} a_1 \\ b_1 \end{array}$$

Adder Circuit Design



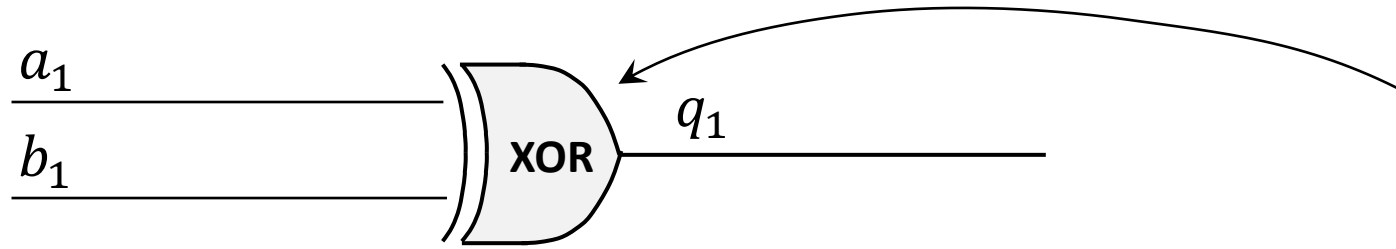
Adder Circuit Design



XOR Truth Table:

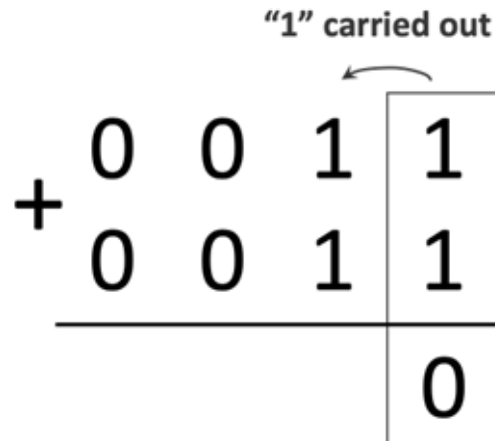
a	b	q
0	0	0
0	1	1
1	0	1
1	1	0

Adder Circuit Design

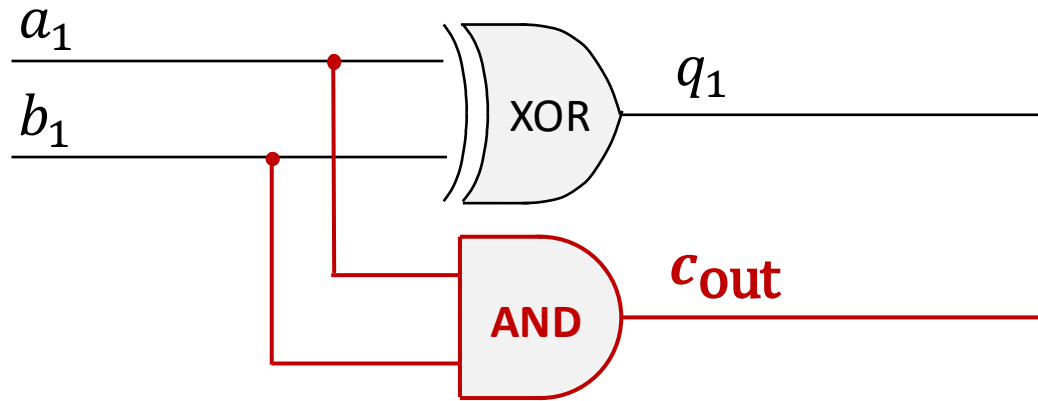


a	b	q
0	0	0
0	1	1
1	0	1
1	1	0

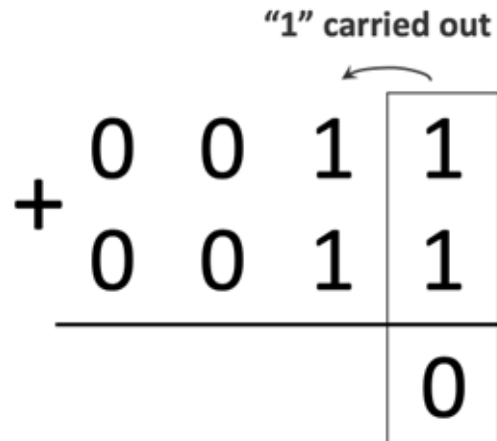
Carry-out case is not handled
(when both inputs are “1”s)



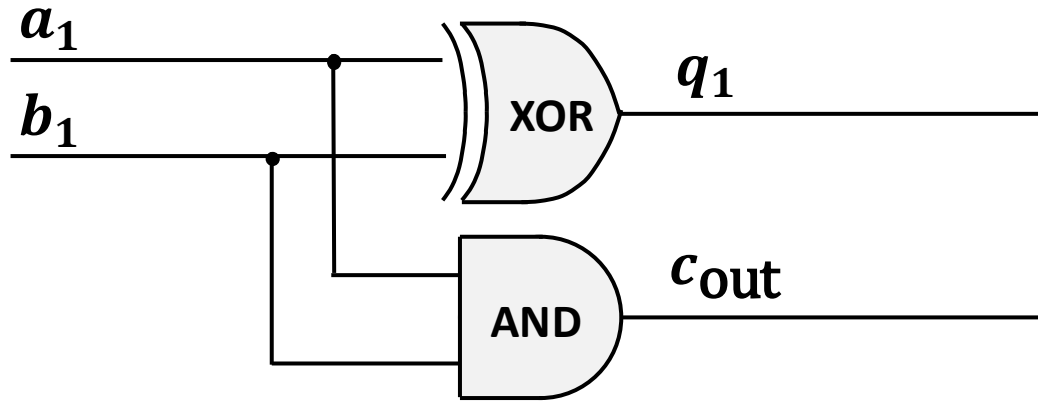
Adder Circuit Design



Carry-out case handling
(when both inputs are “1”s)



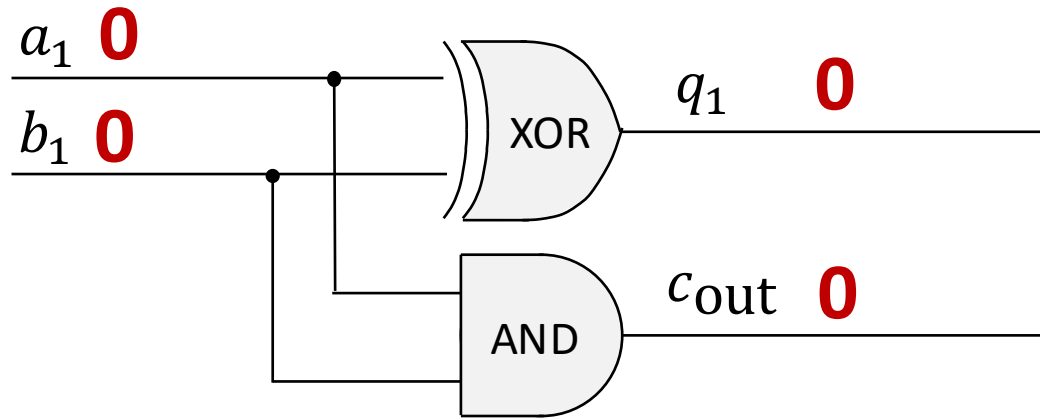
Half-Adder Circuit: Least Significant Bits (LSB) Summation



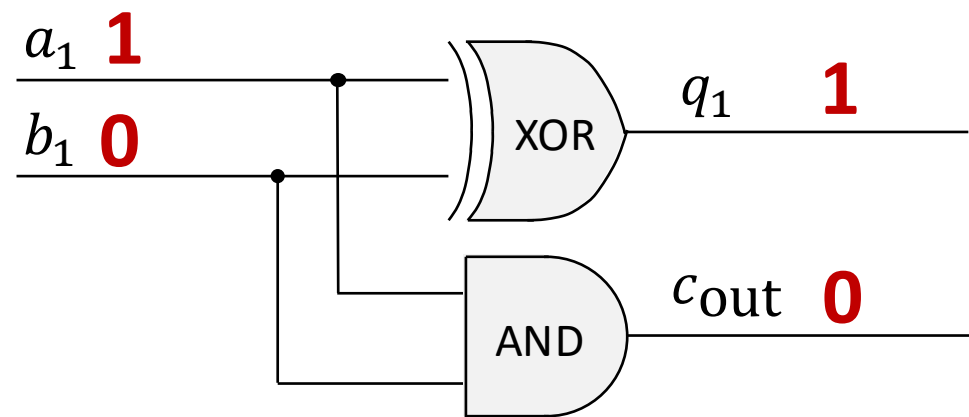
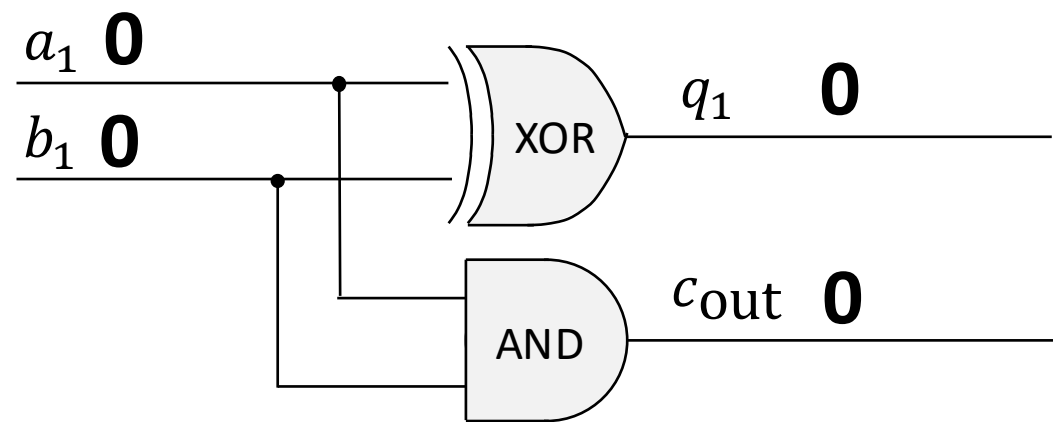
“1” carried out

	0	0	1	1
+	0	0	1	1
				0

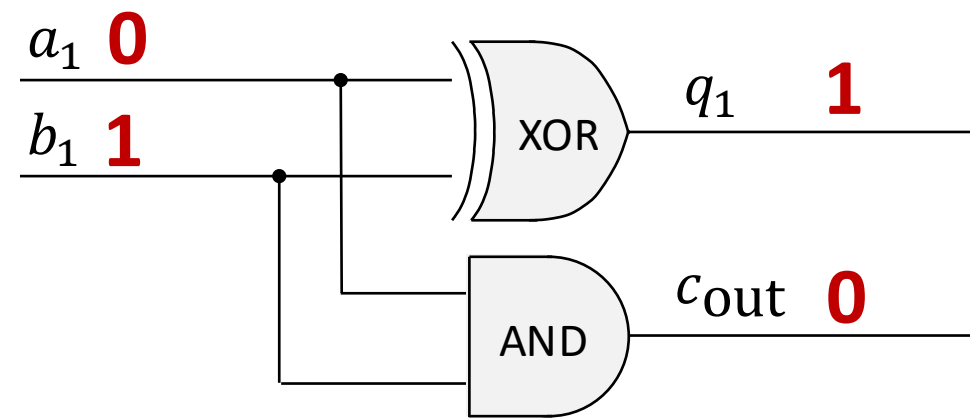
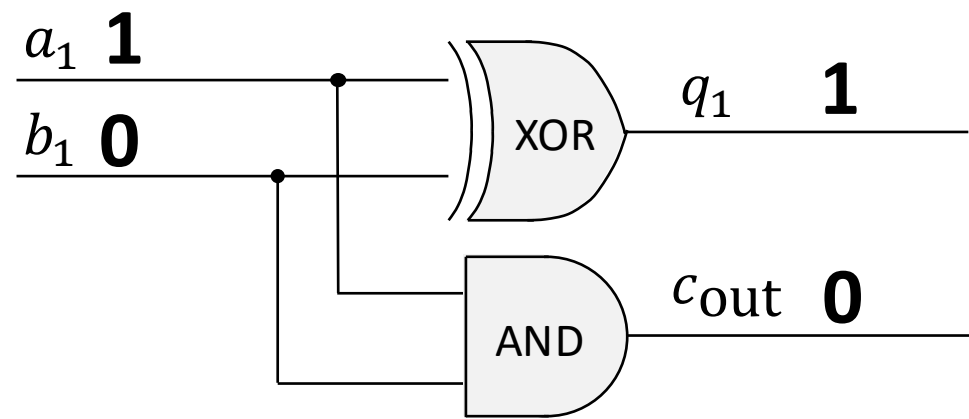
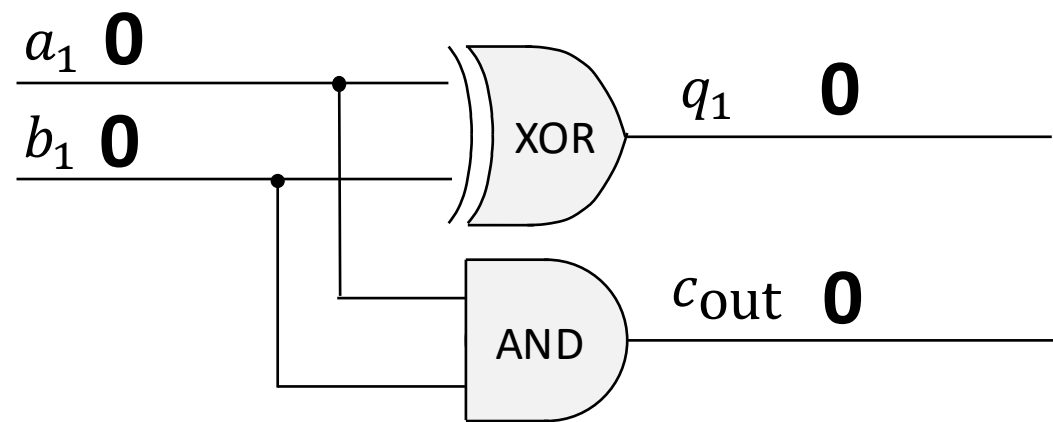
Half-Adder Circuit: Design Verification



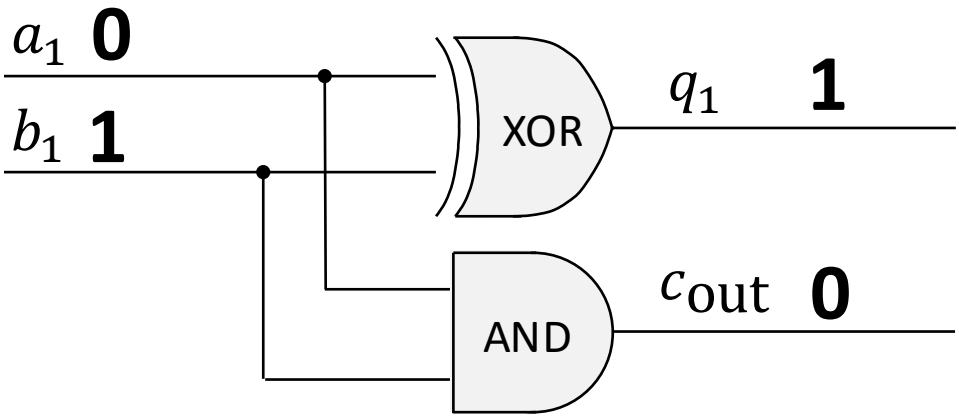
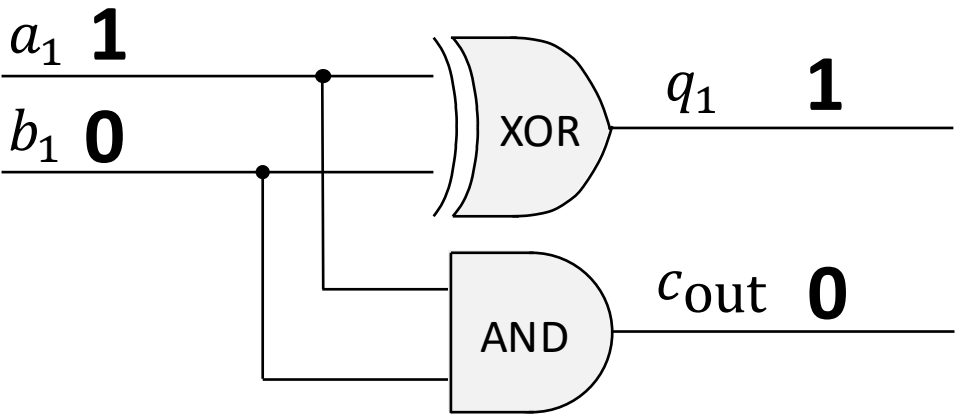
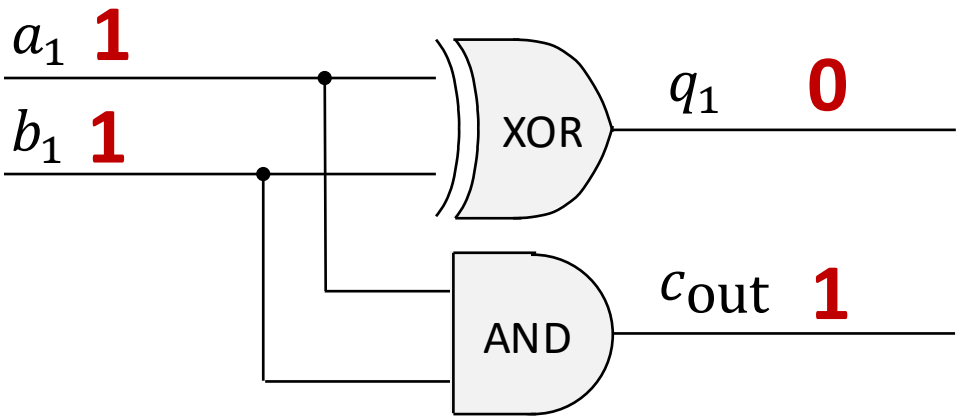
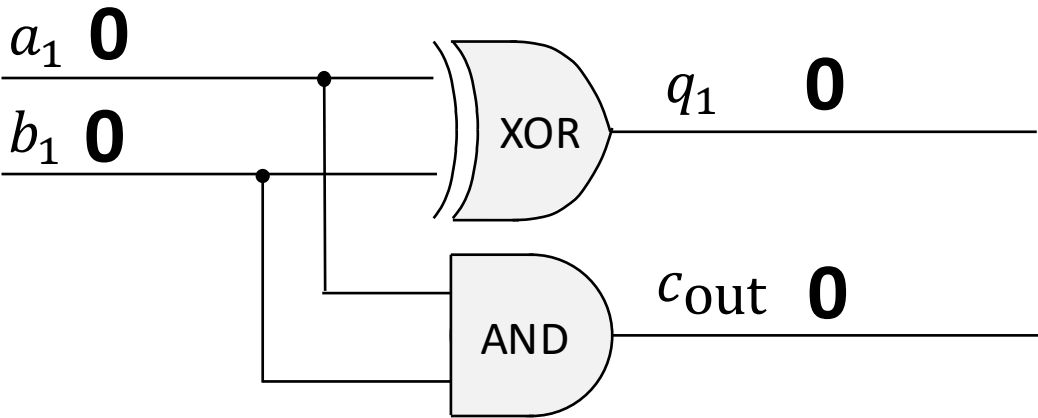
Half-Adder Circuit: Design Verification



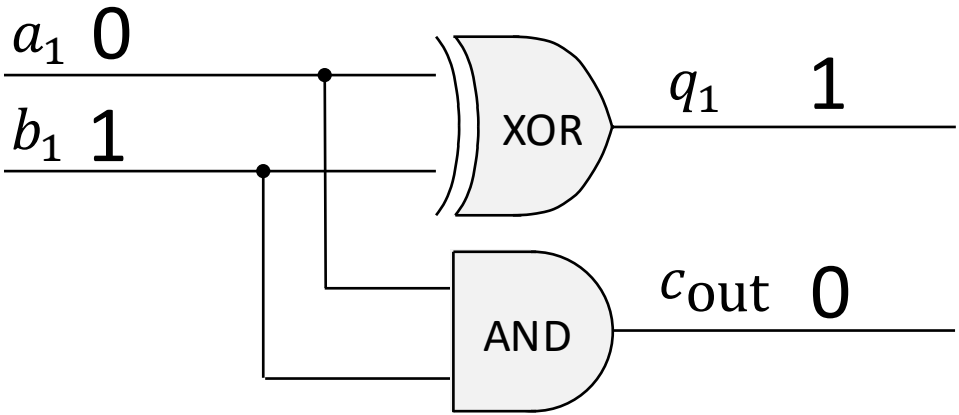
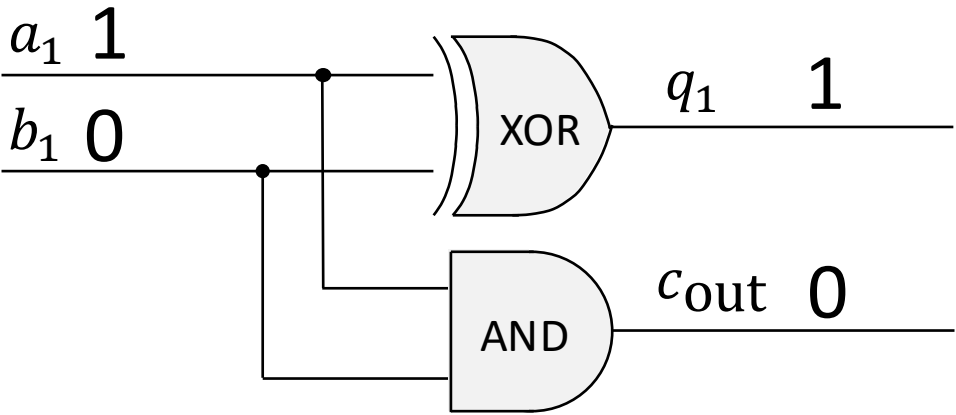
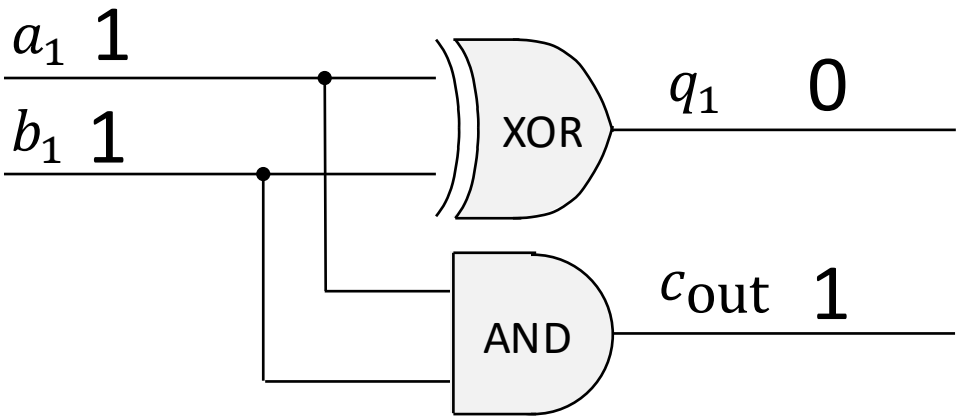
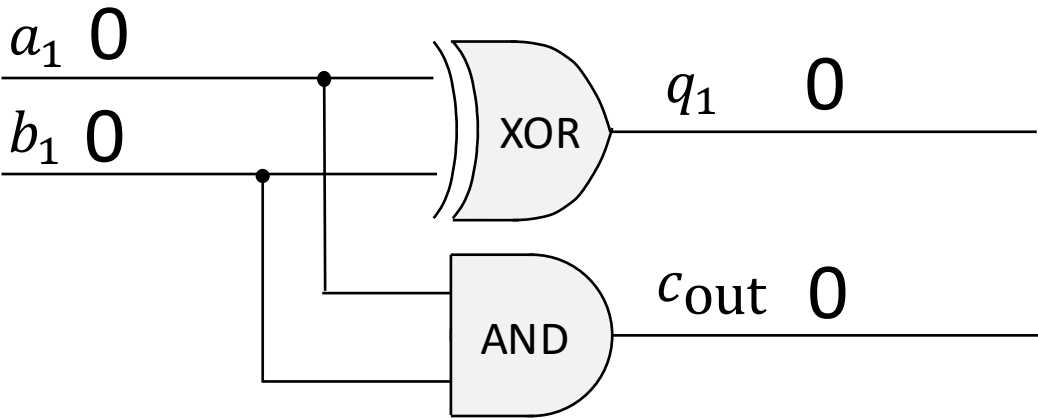
Half-Adder Circuit: Design Verification



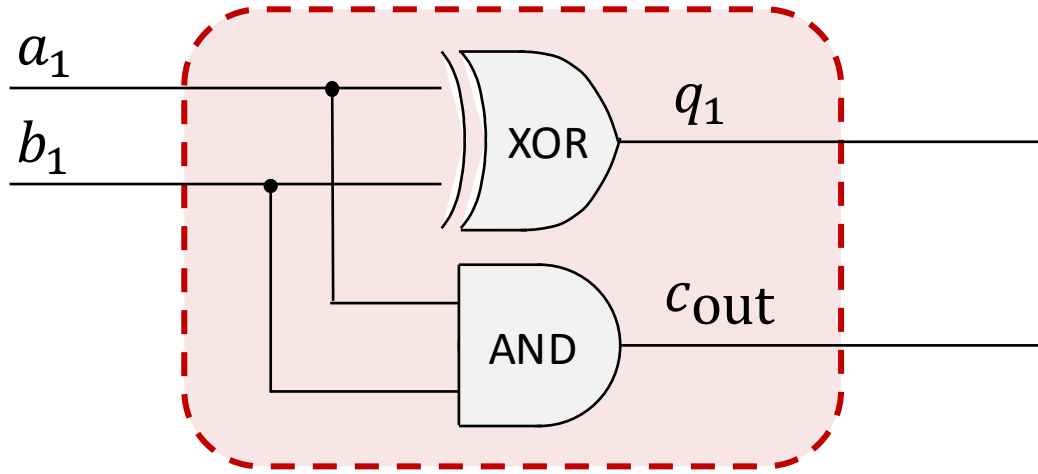
Half-Adder Circuit: Design Verification



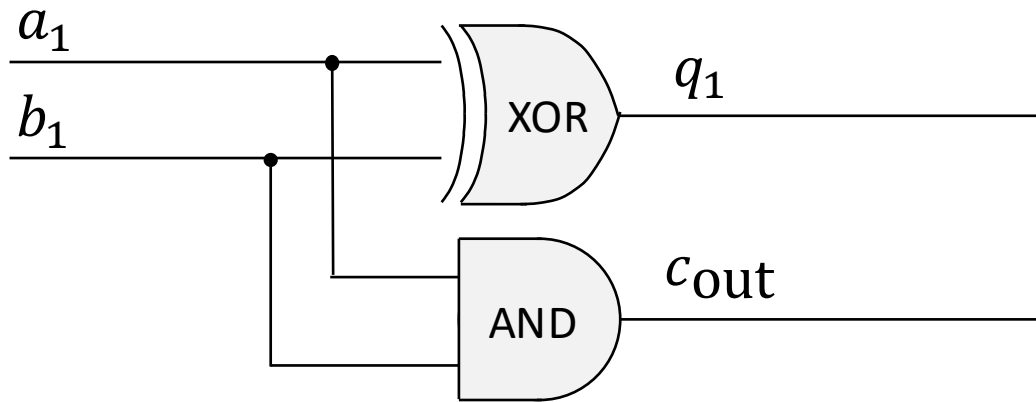
Half-Adder Circuit



Adder Circuit Design: Continuation



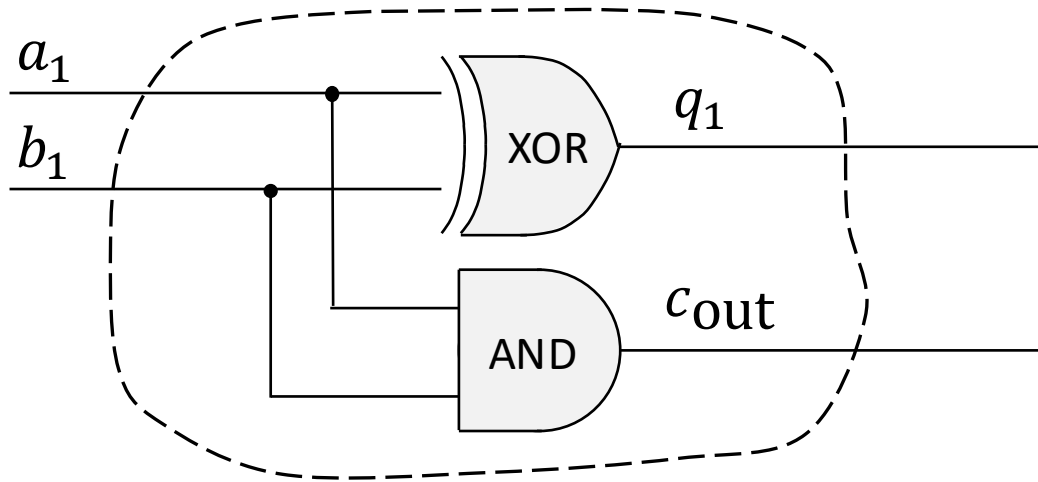
Adder Circuit Design



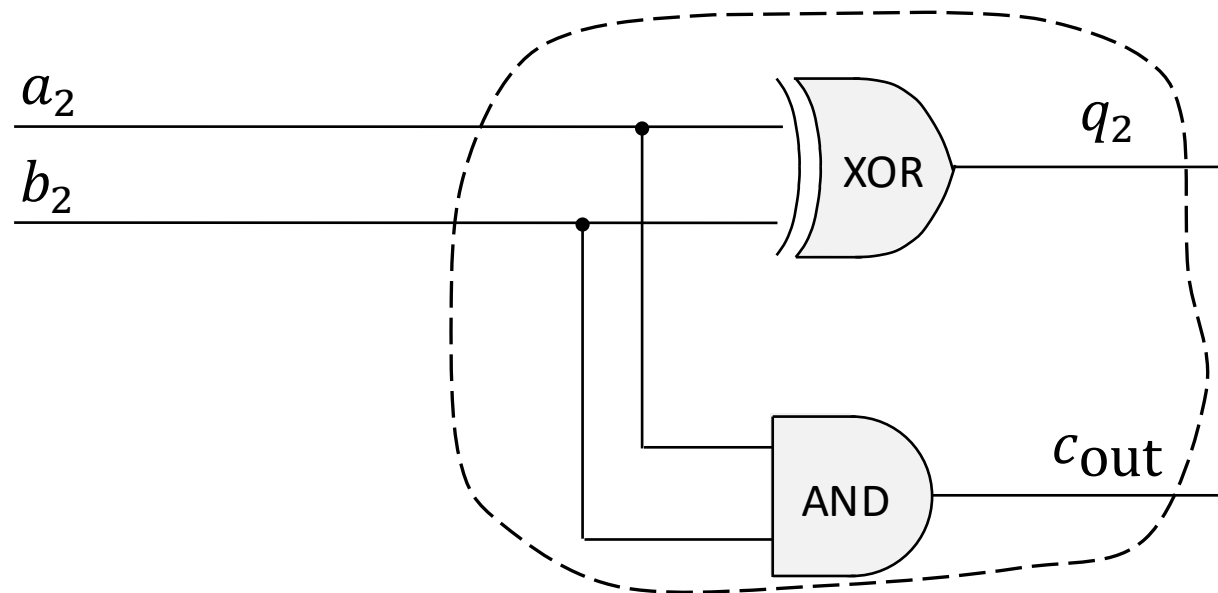
a_2

b_2

Adder Circuit Design

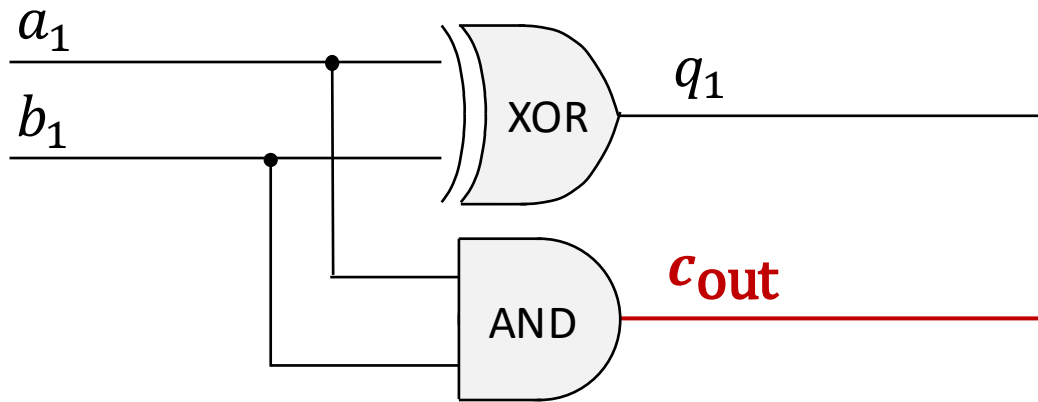


Identical



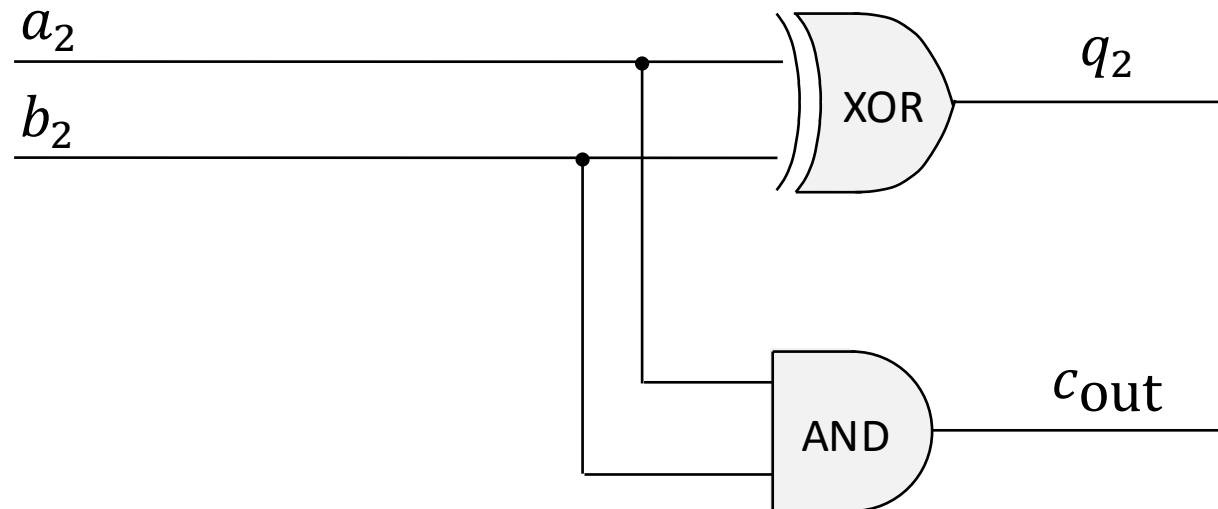
Adder Circuit Design:

Carry-in is to be handled



“1” carried out

	0	0	1	1
+	0	0	1	1
				0



Adder Circuit Design: Carry-in Handling

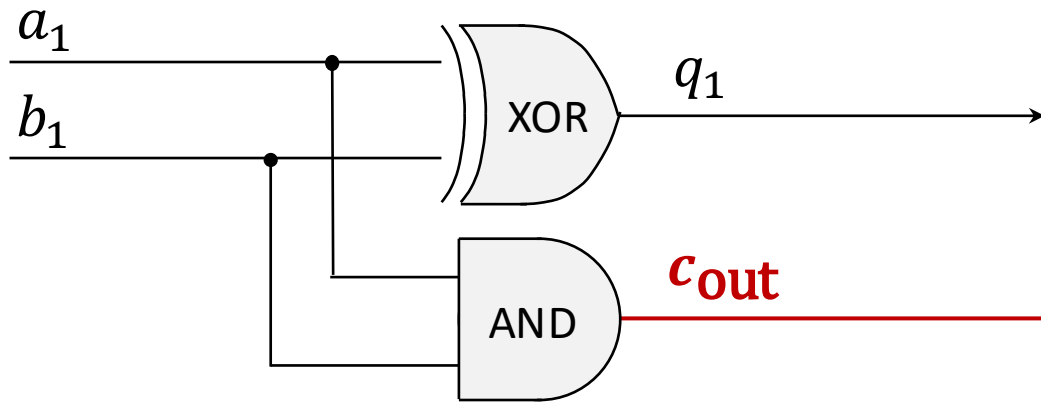
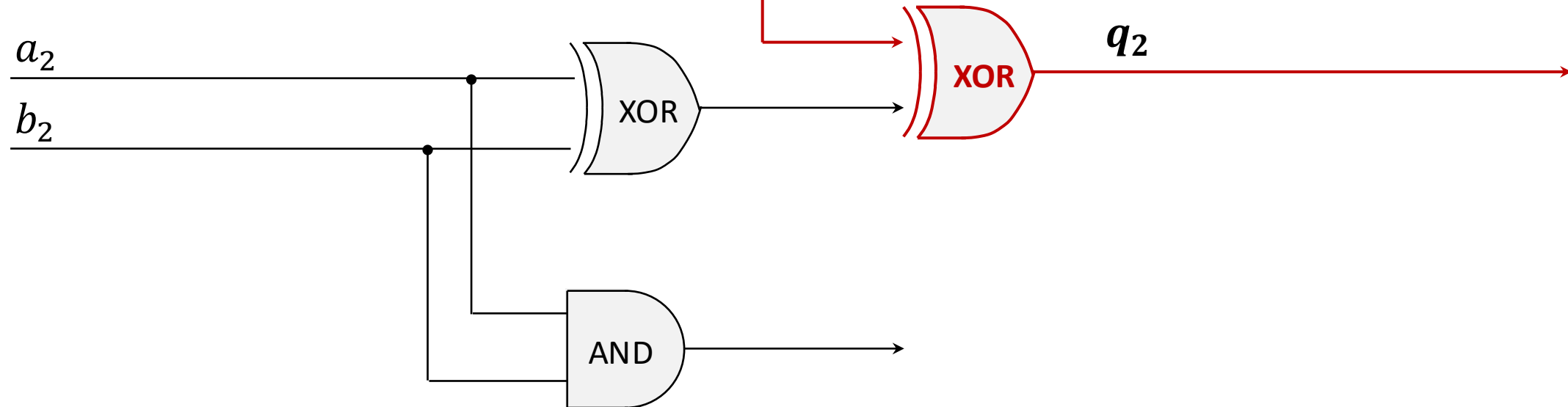
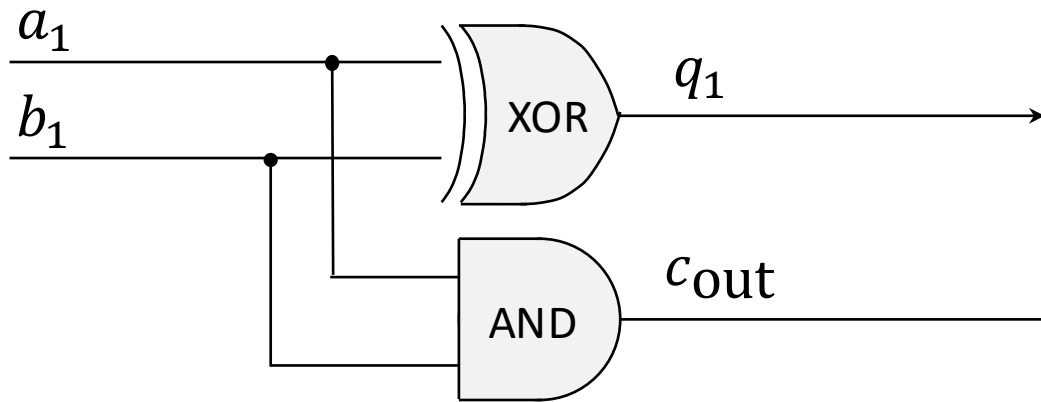


Diagram illustrating the carry-in handling for the second stage. The carry-in is a "1" carried out from the first stage, which is added to the second stage inputs a_2 and b_2 .

	0	0	1	1
+	0	0	1	1
				0

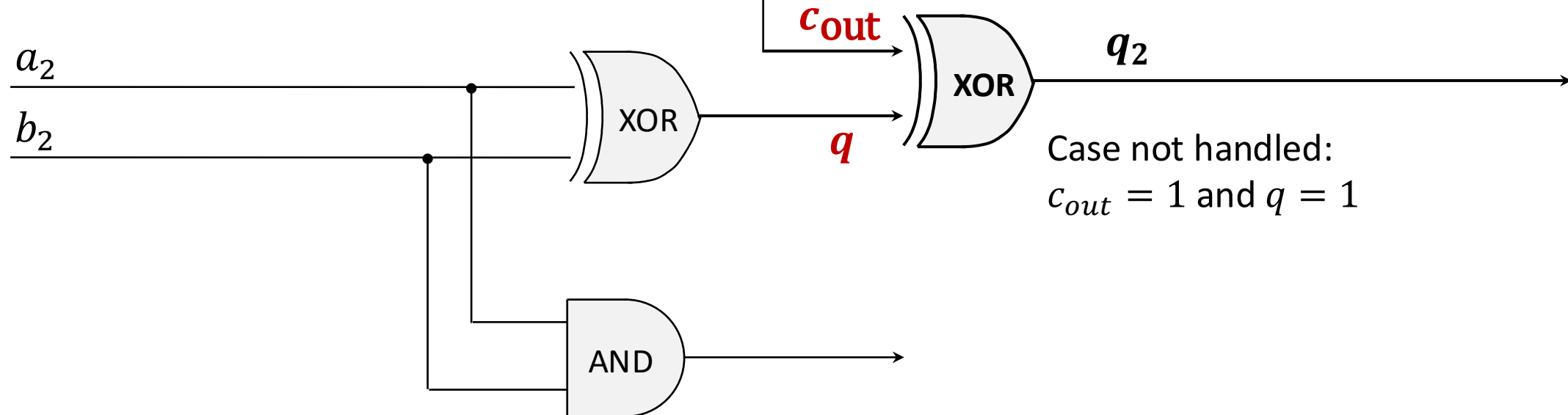


Adder Circuit Design: Carry-in Handling



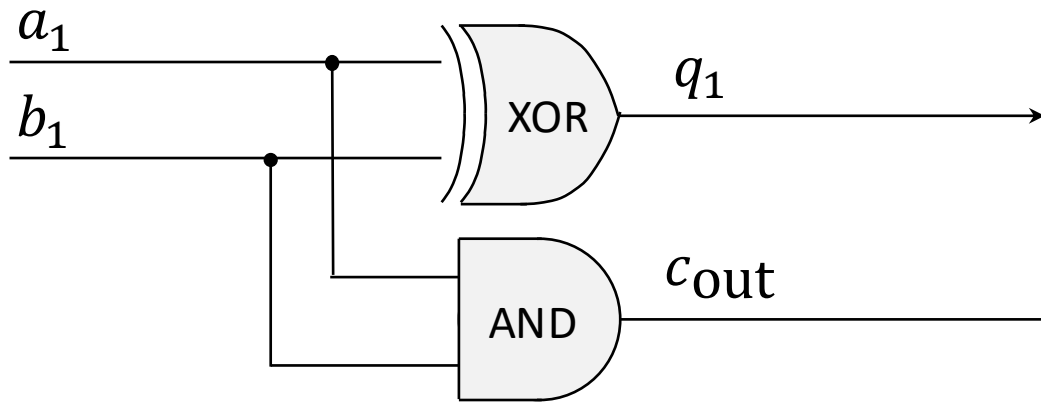
"1" carried out

	0	0	1	1
+	0	0	1	1
				0



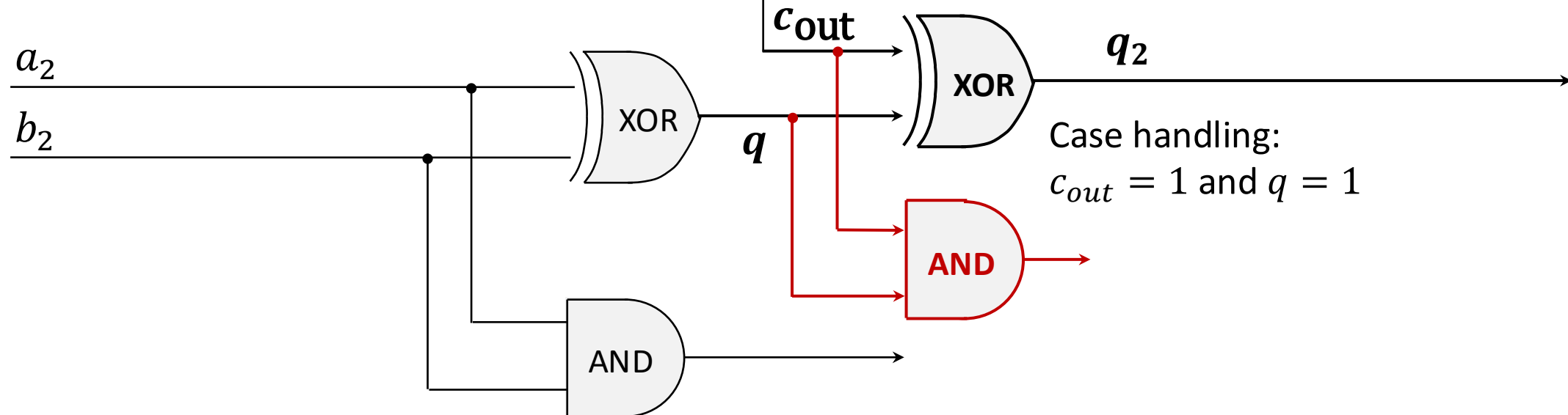
Case not handled:
 $c_{out} = 1$ and $q = 1$

Adder Circuit Design: Carry-in Handling

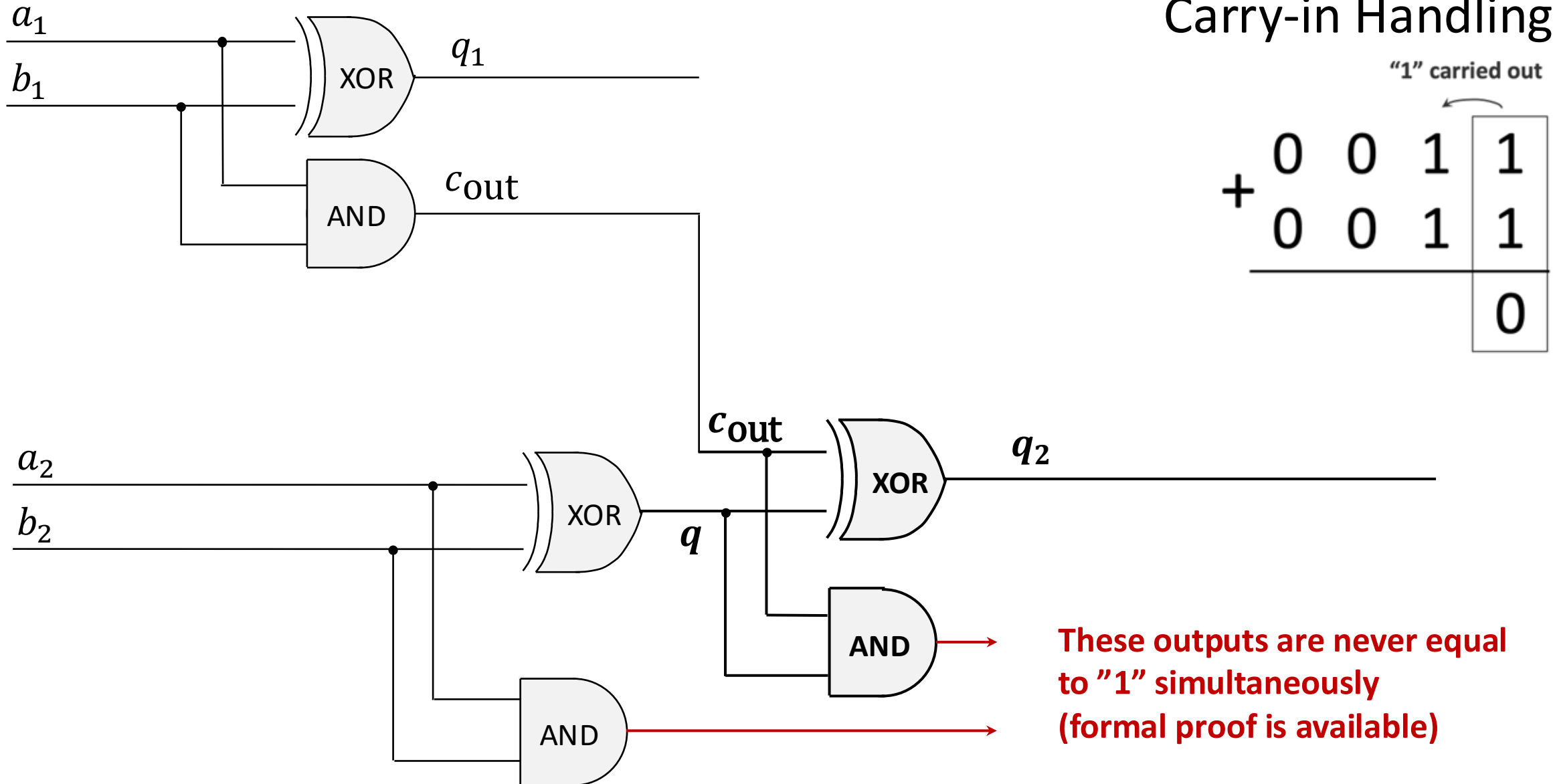


"1" carried out

	0	0	1	1
+	0	0	1	1
				0



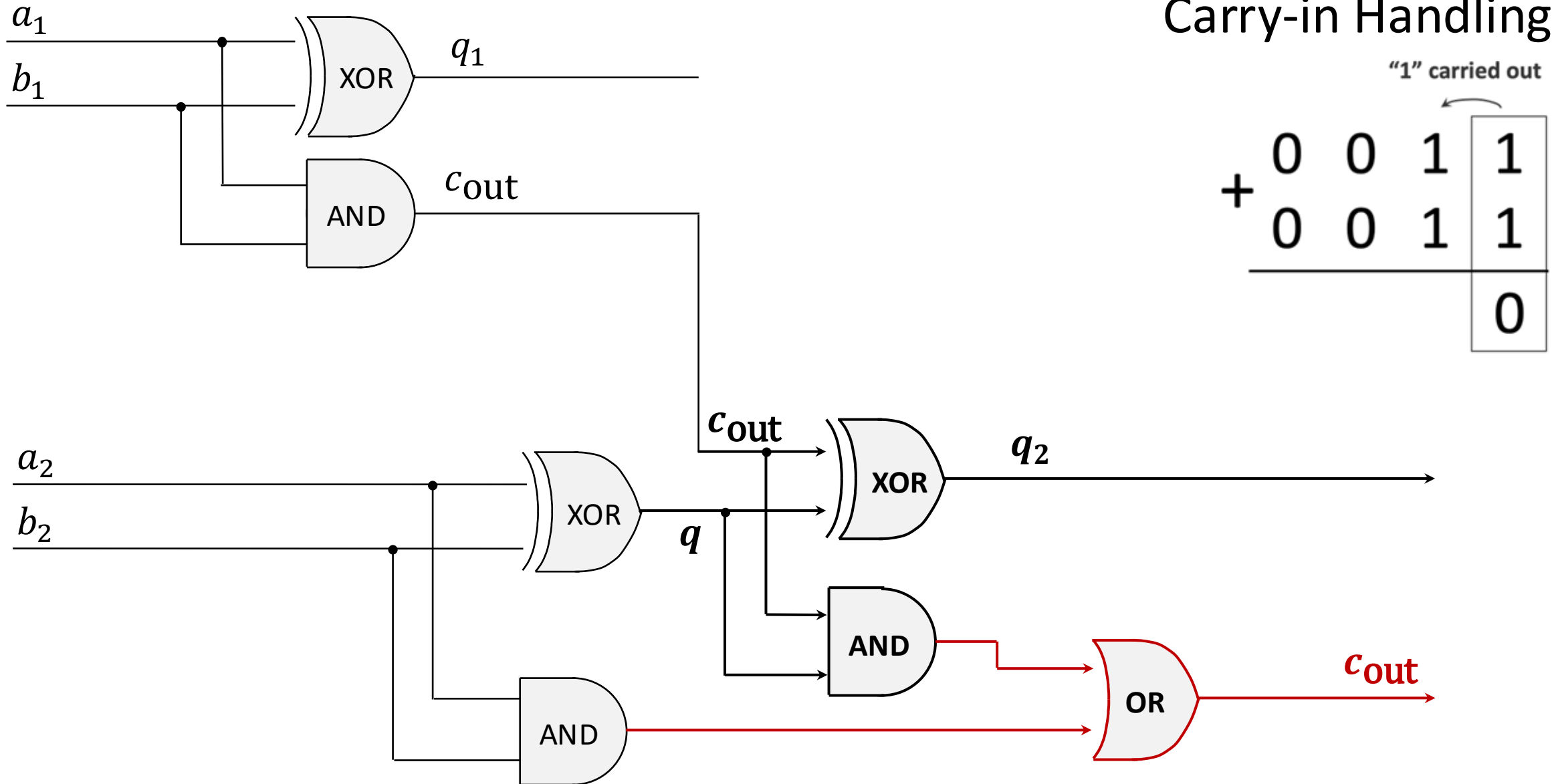
Adder Circuit Design: Carry-in Handling

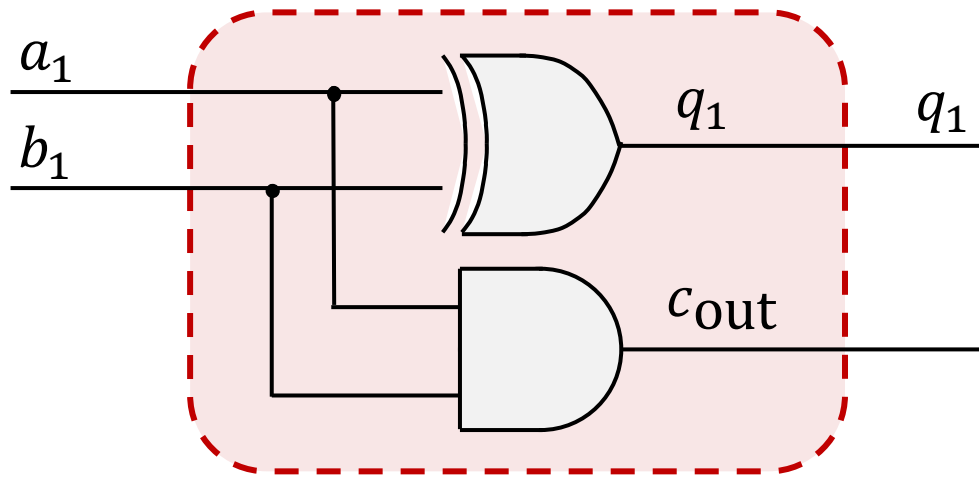


"1" carried out

+	0	0	1	1
	0	0	1	1
				0

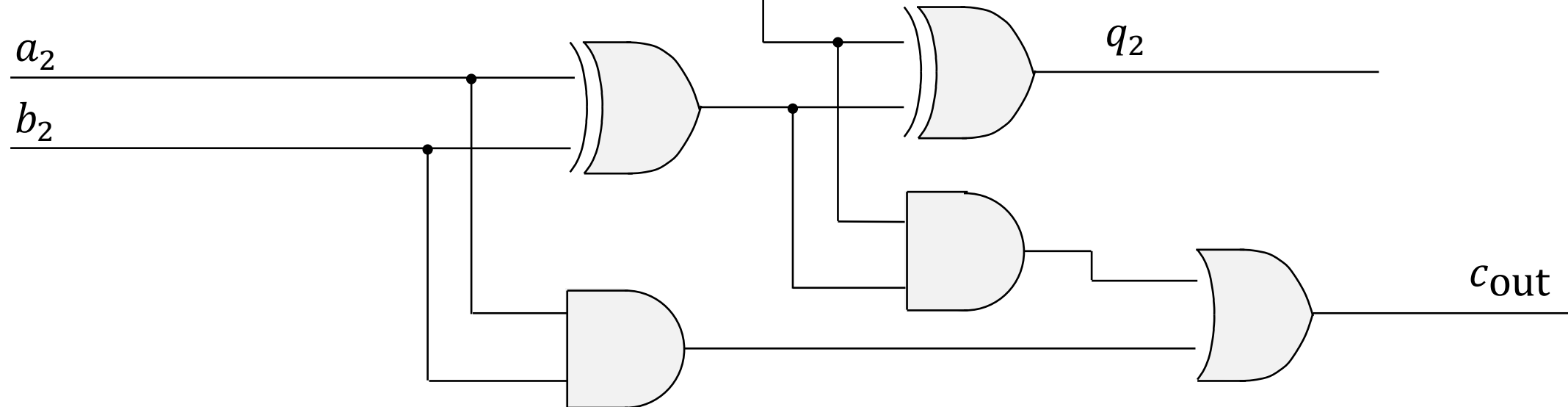
Adder Circuit Design: Carry-in Handling

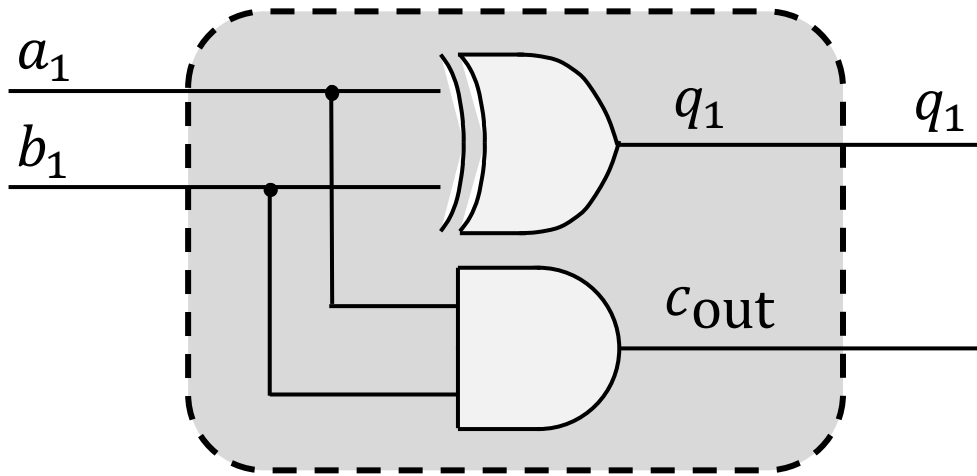




Half-Adder:

- No carry-in handling
- Least Significant Bits (LSB) summation only



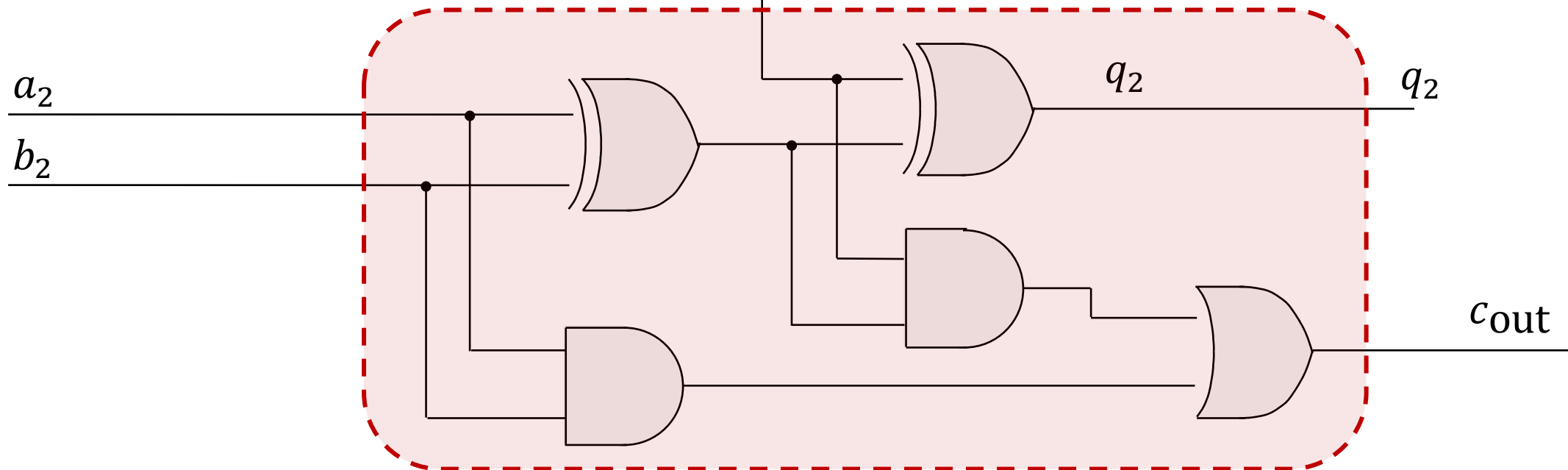


Half-Adder:

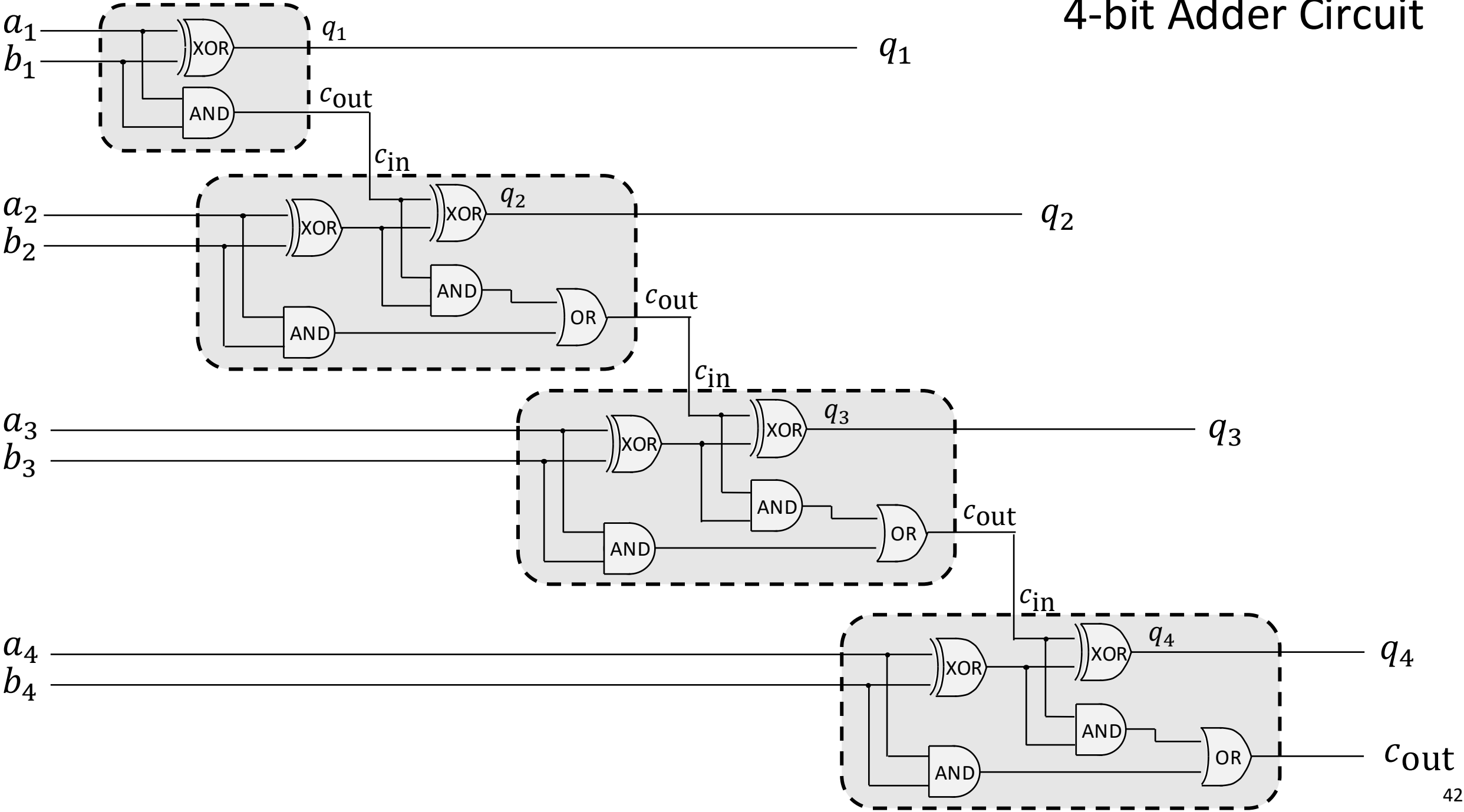
- No carry-in handling
- Least Significant Bits (LSB) summation only

Full-Adder:

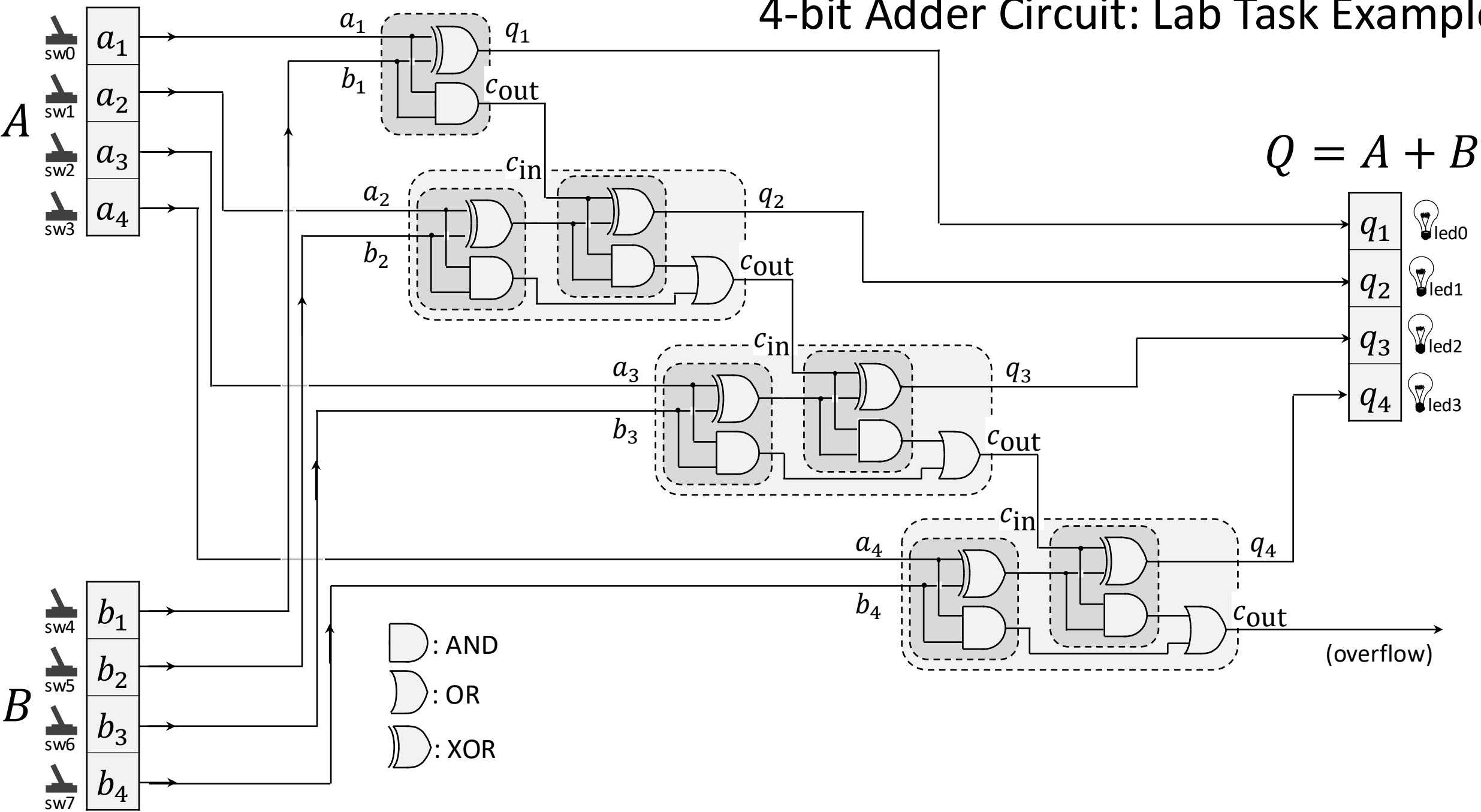
- Carry-in handling



4-bit Adder Circuit

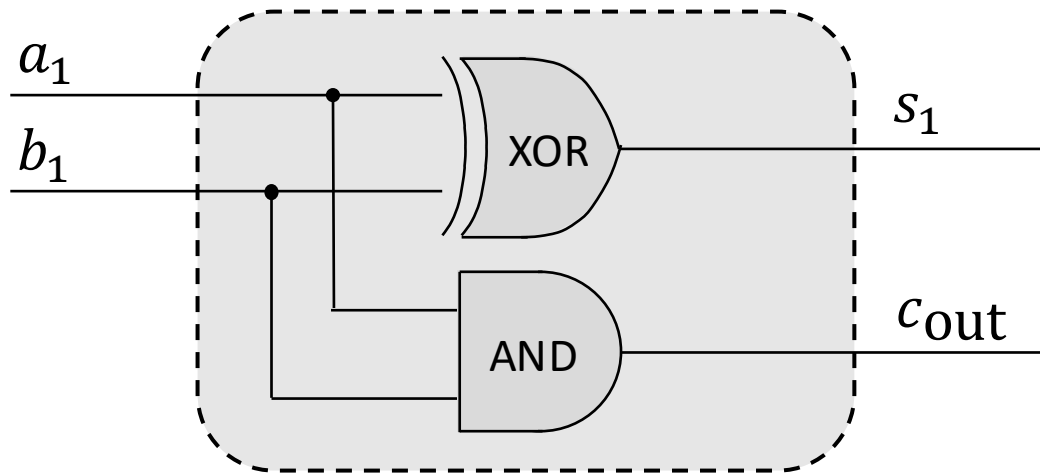


4-bit Adder Circuit: Lab Task Example



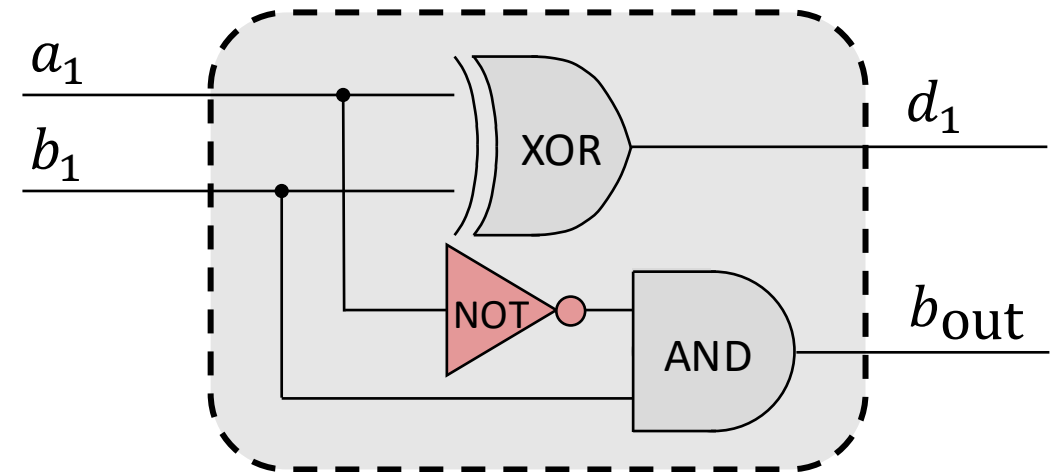
Half Adder:

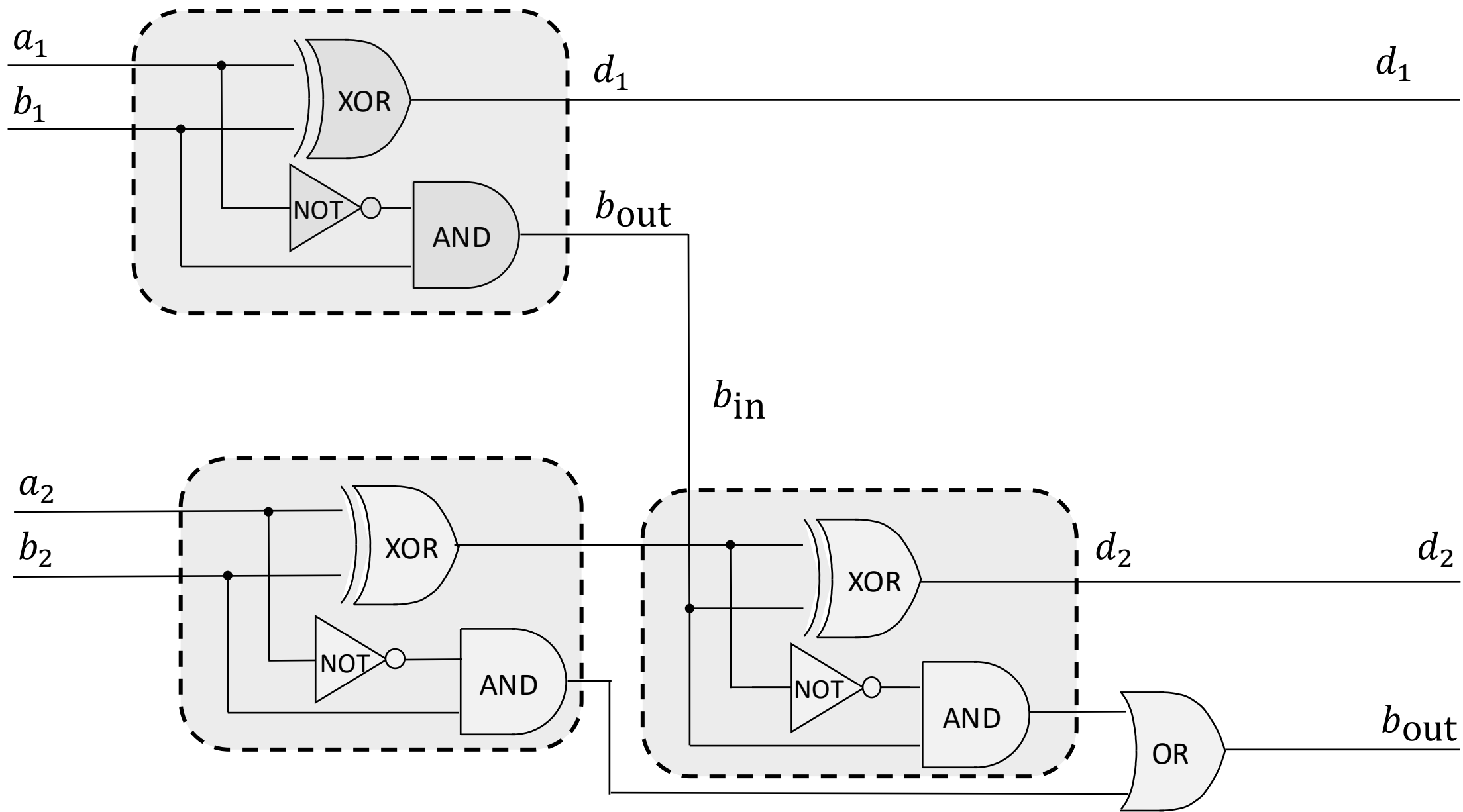
- No carry-in handling
- Summation of Least Significant Bits



Half Subtractor:

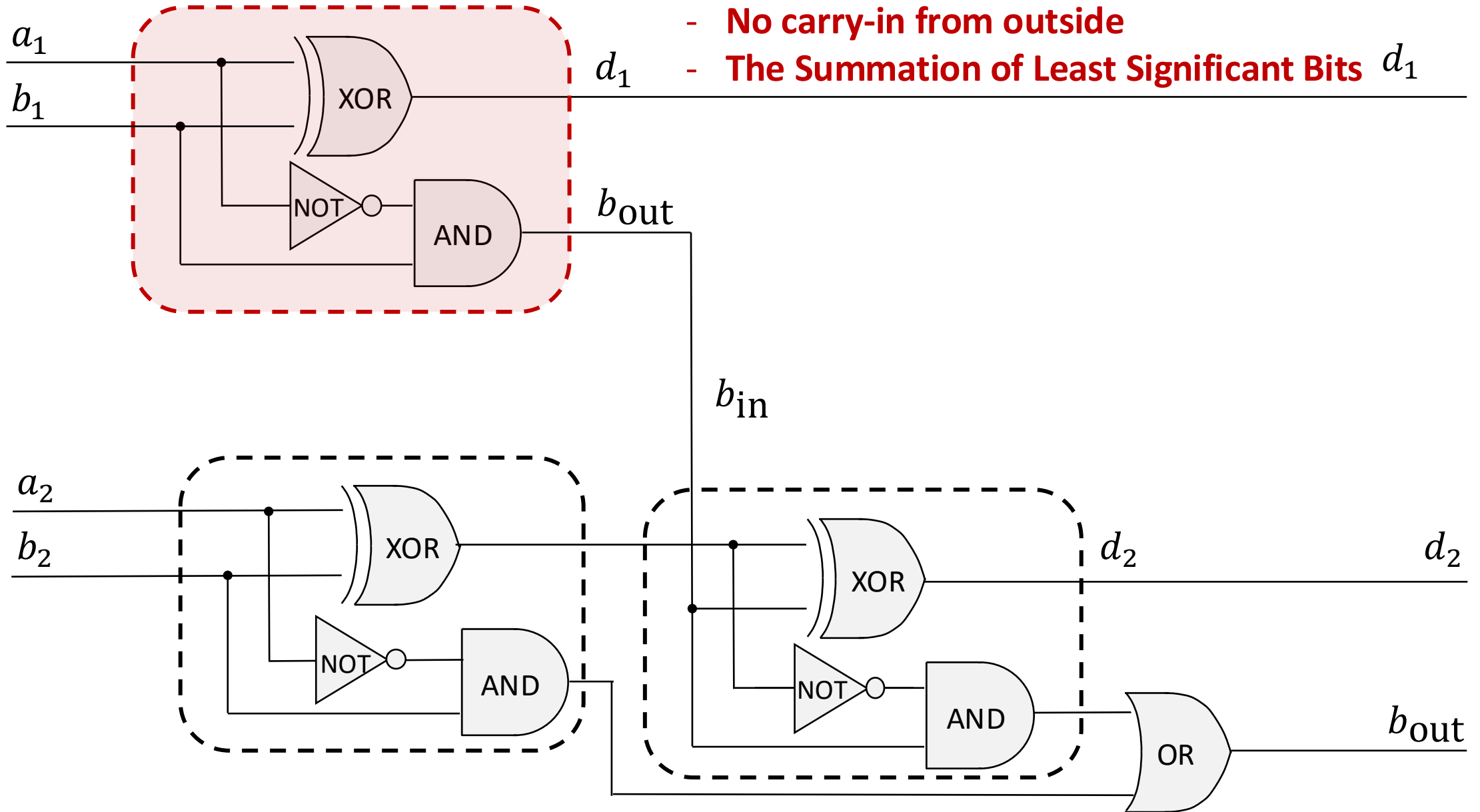
- No borrow-in handling
- Subtraction of Least Significant Bit





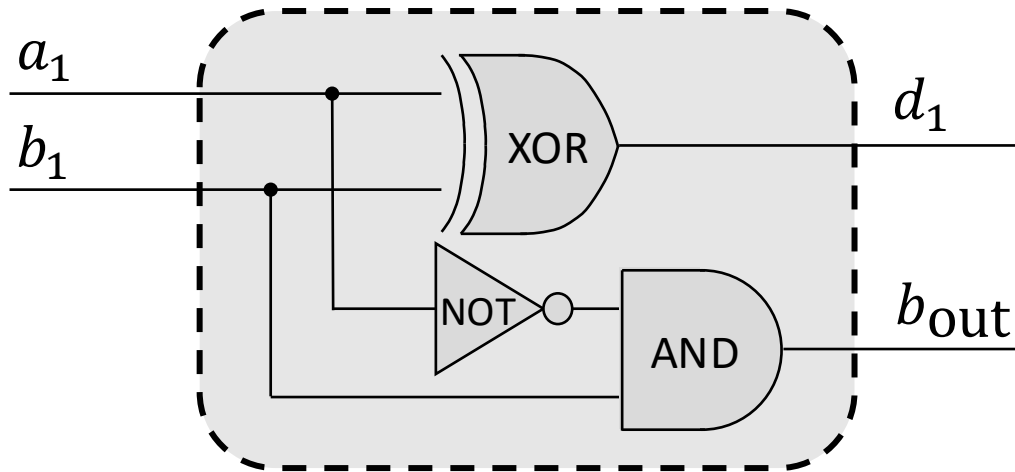
Half Subtractor Circuit:

- No carry-in from outside
- The Summation of Least Significant Bits d_1



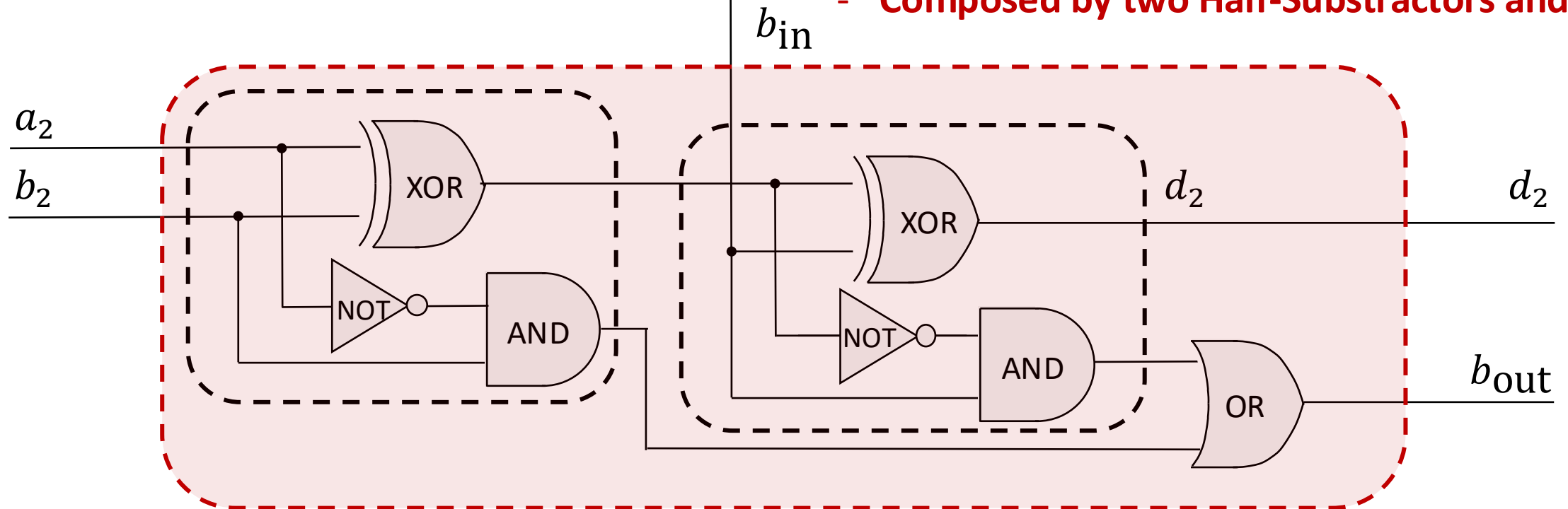
Half Subtractor Circuit:

- No carry-in from outside
- The Summation of Least Significant Bits d_1

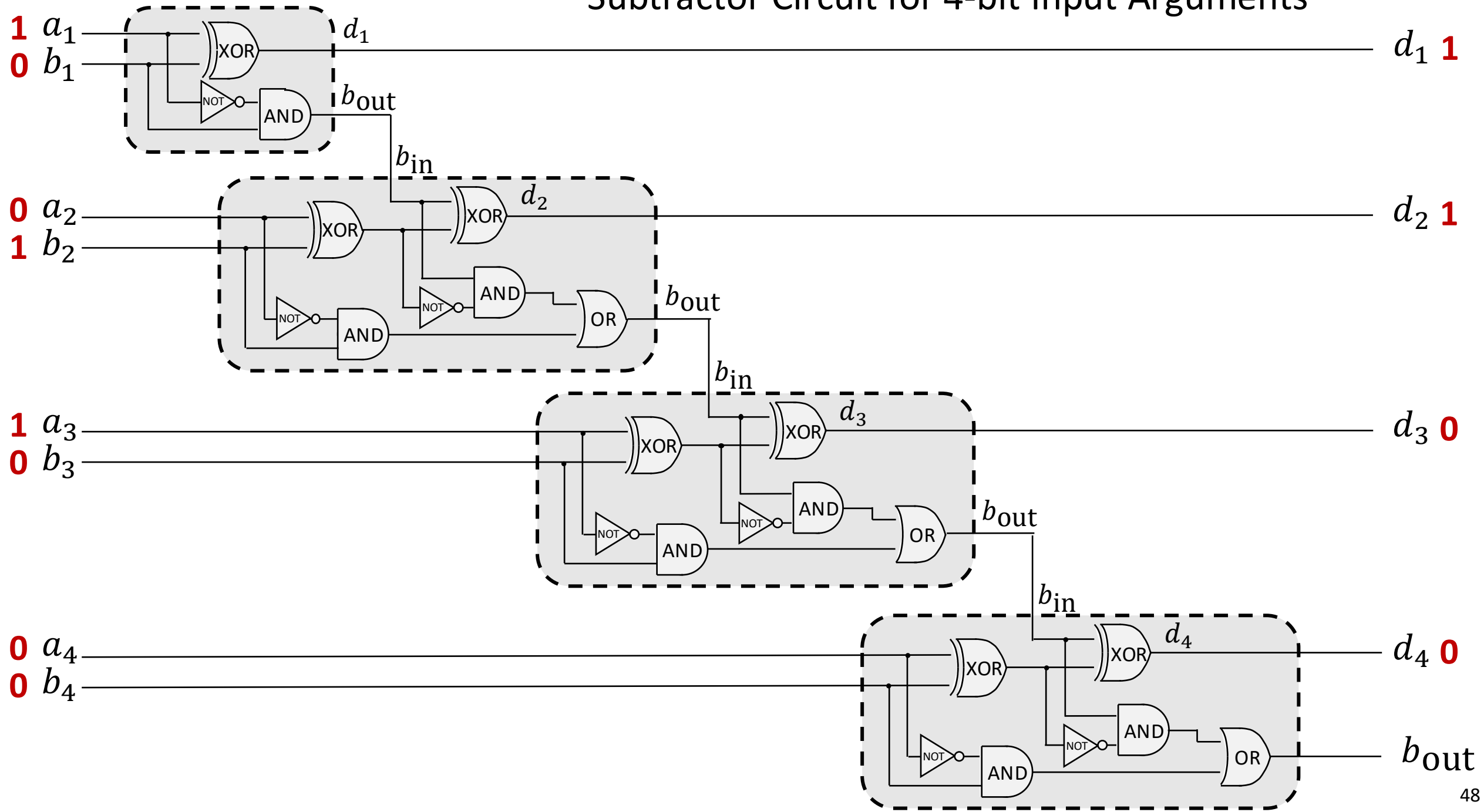


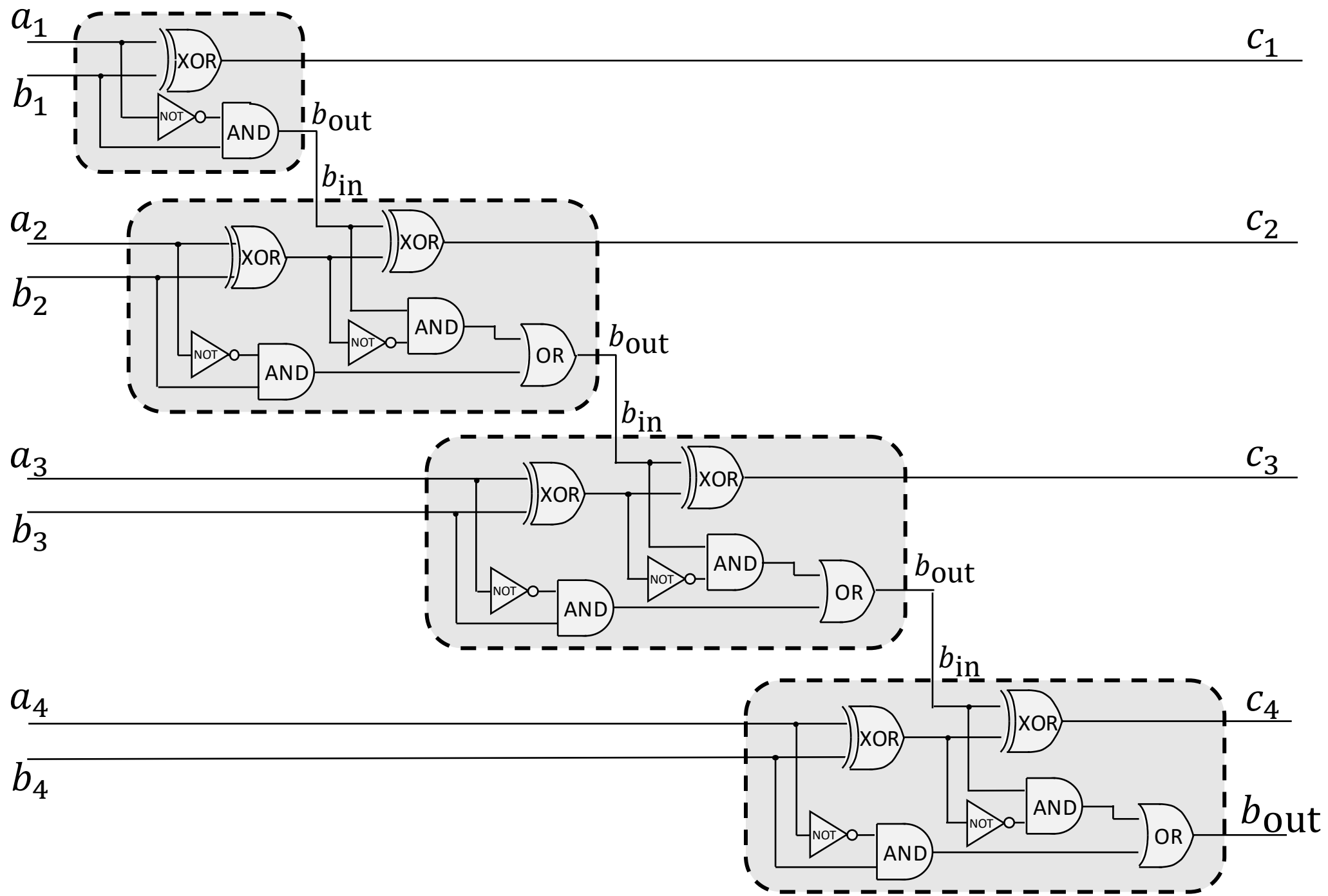
Full Subtractor Circuit:

- Takes borrow-in from outside
- Composed by two Half-Subtractors and OR



Subtractor Circuit for 4-bit Input Arguments





Another Example of Combinational Circuit:
Multiplier for 2-by-2 Bits Input Arguments

