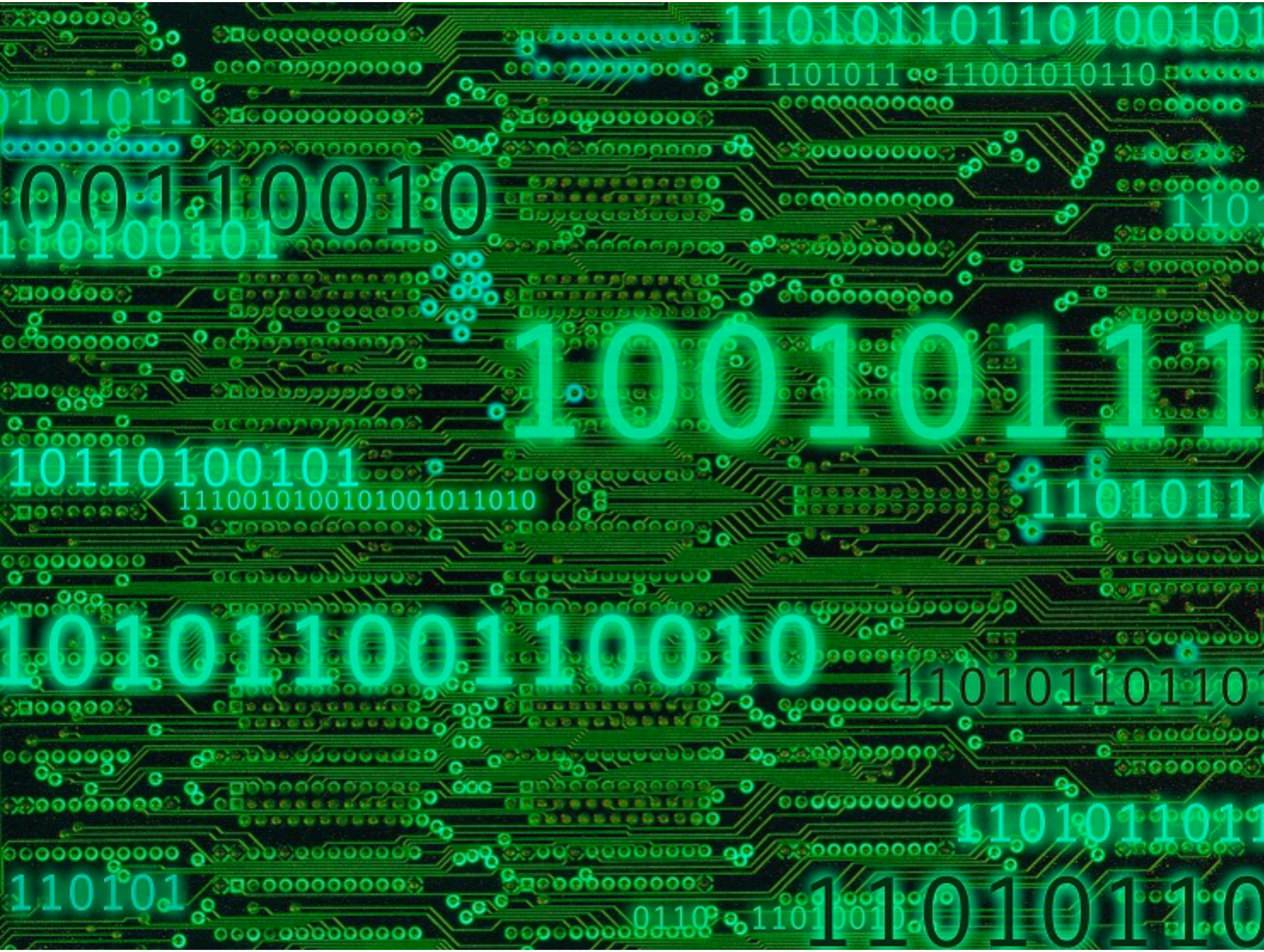


Binary Numbers

0110 (base 2)

$0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1$

6 (base 10)

Binary Numbers

$2^1 2^0$

00	0
01	1
10	2
11	3

max value = $2^2 - 1$

2-bit binary number

Binary Numbers

000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

max value = $2^3 - 1$

3-bit binary number

Boolean Logic (variables)

1 = True

0 = False

Boolean Logic (truth tables)

a	**b**	**a** *and* **b**
1	1	1
1	0	0
0	1	0
0	0	0

a *and* **b**

Boolean Logic (truth tables)

a	**b**	**a** *or* **b**
1	1	1
1	0	1
0	1	1
0	0	0

a *or* **b**

Boolean Logic (truth tables)

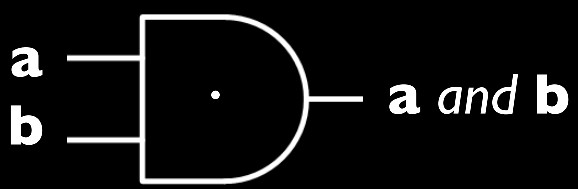
a	*not* **a**
1	0
0	1

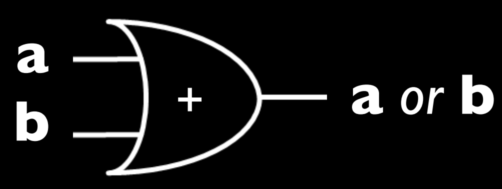
not **a**

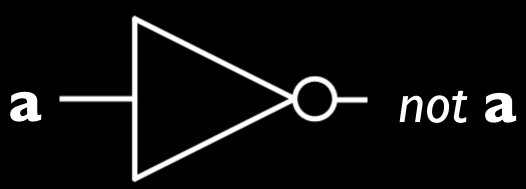
a	**b**	**a** *and* **b**
1	1	1
1	0	0
0	1	0
0	0	0

a	**b**	**a** *or* **b**
1	1	1
1	0	1
0	1	1
0	0	0

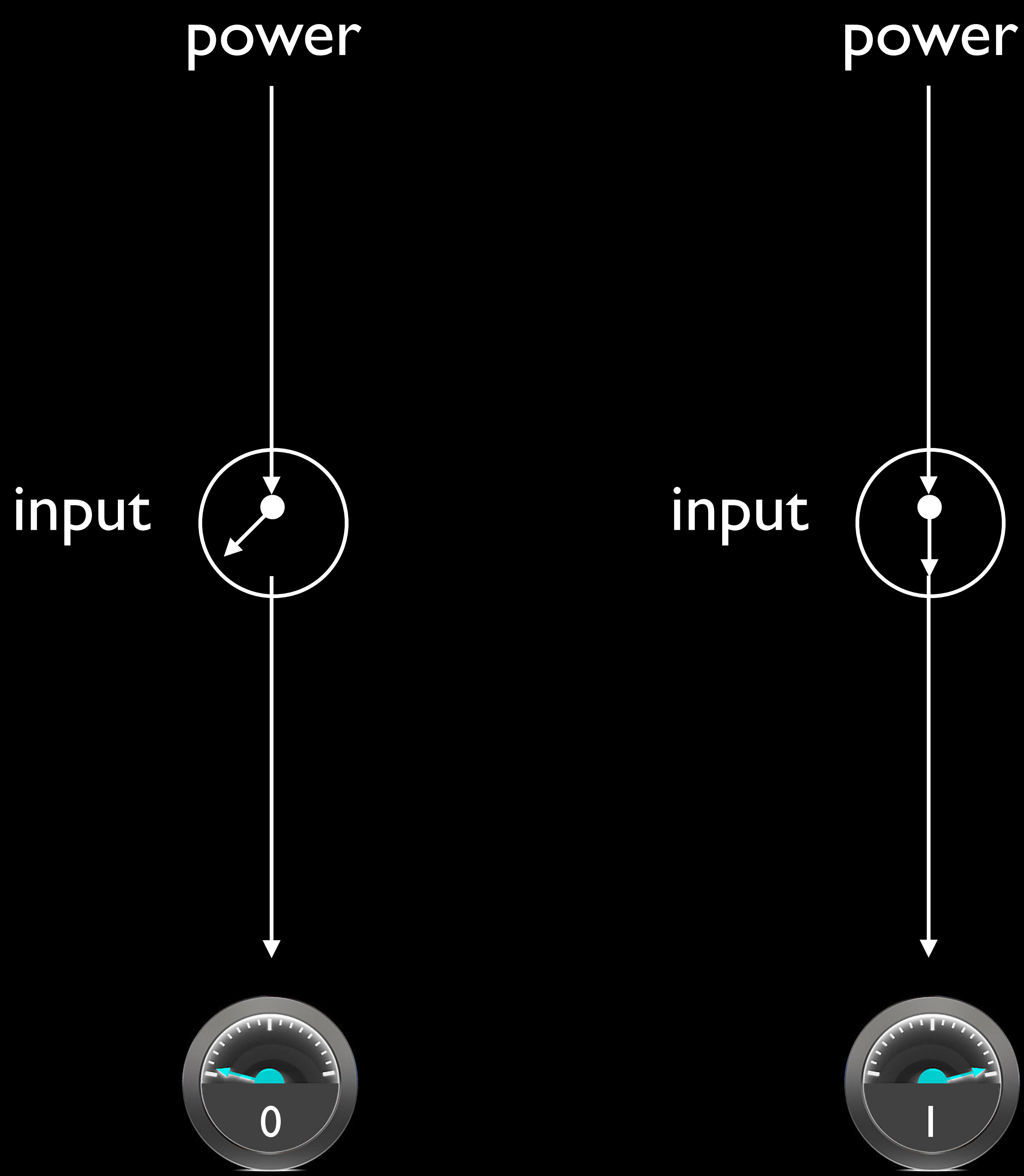
a	*not* **a**
1	0
0	1



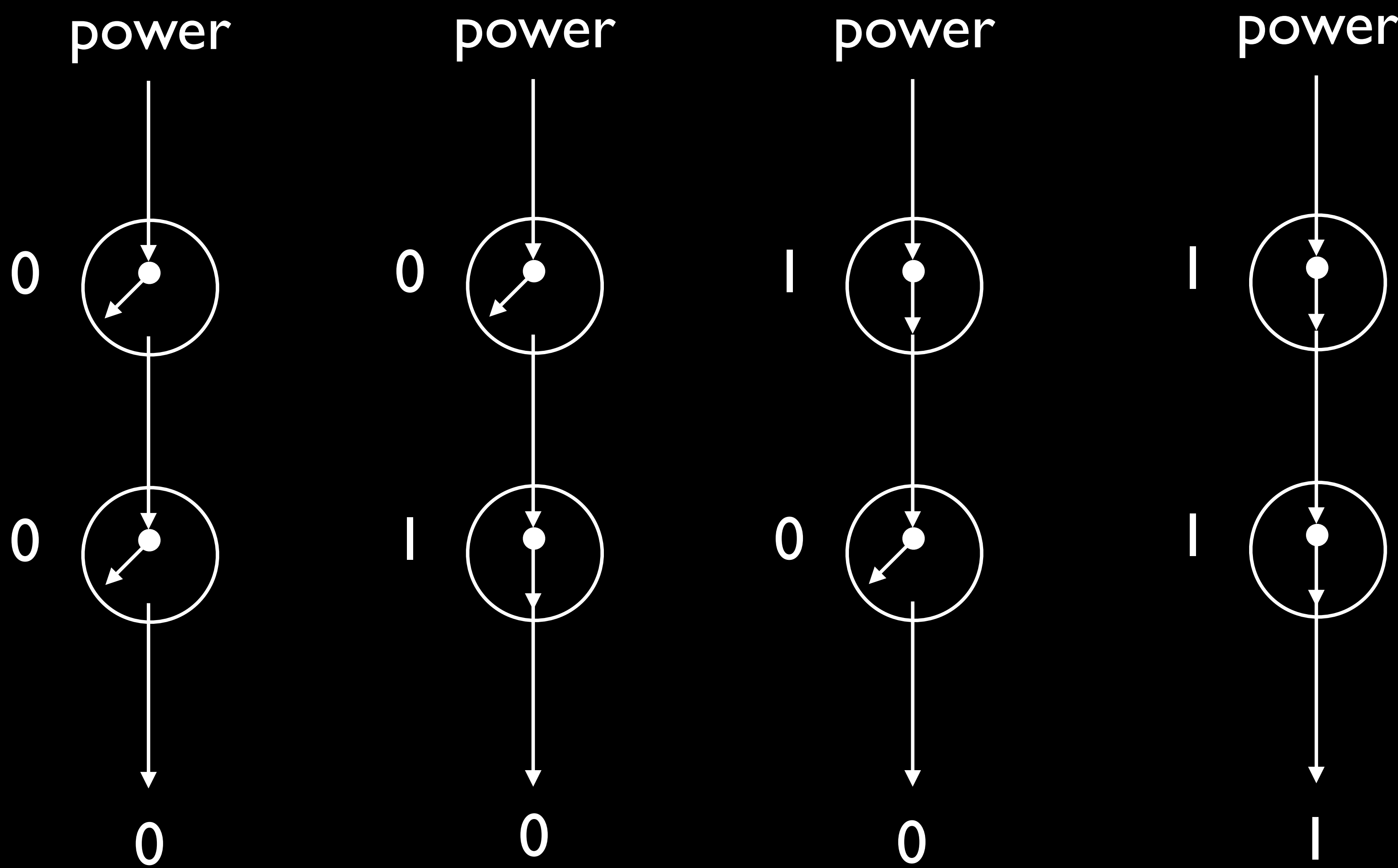




Building Gates (transistors)

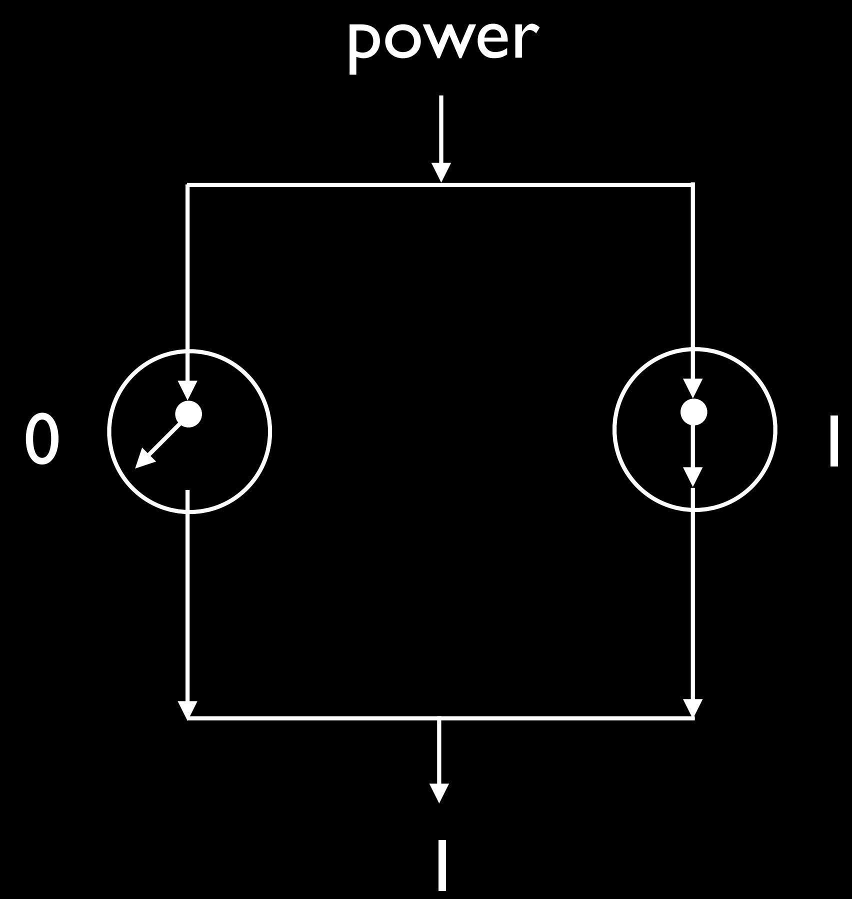


Building Gates (transistors)



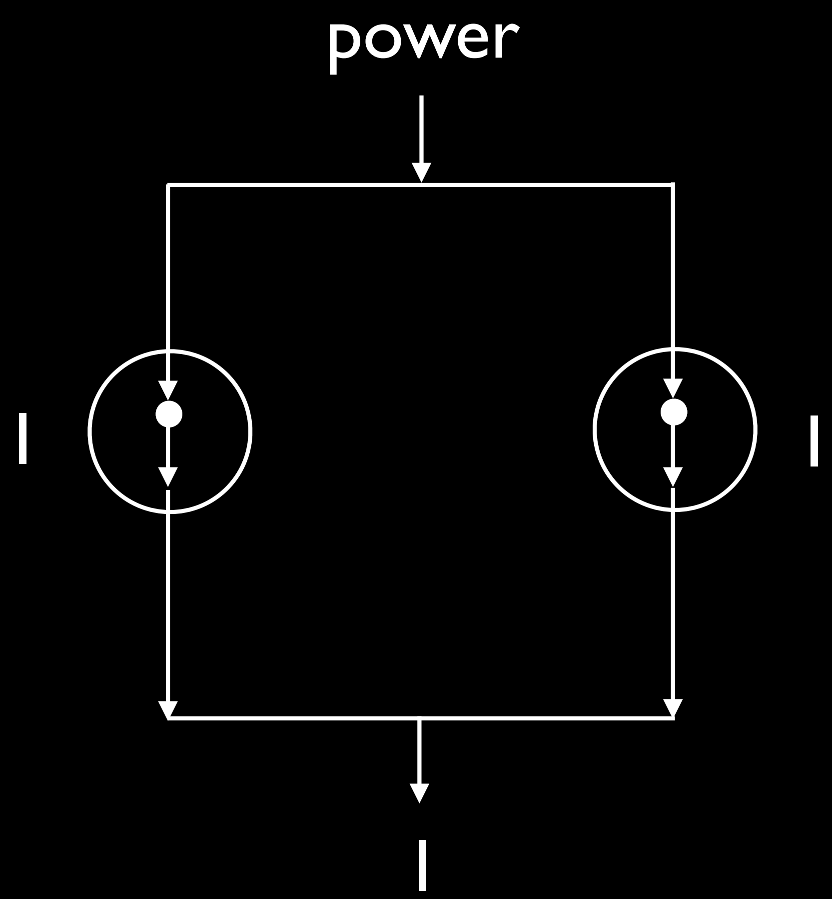
AND gate

Building Gates (transistors)



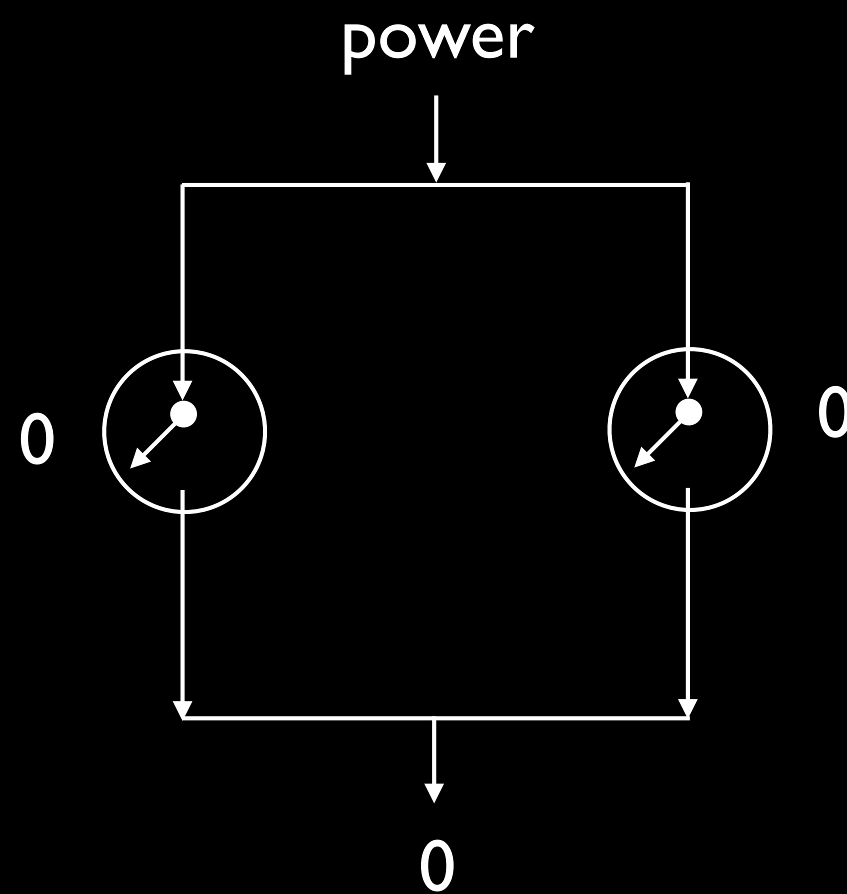
OR gate

Building Gates (transistors)



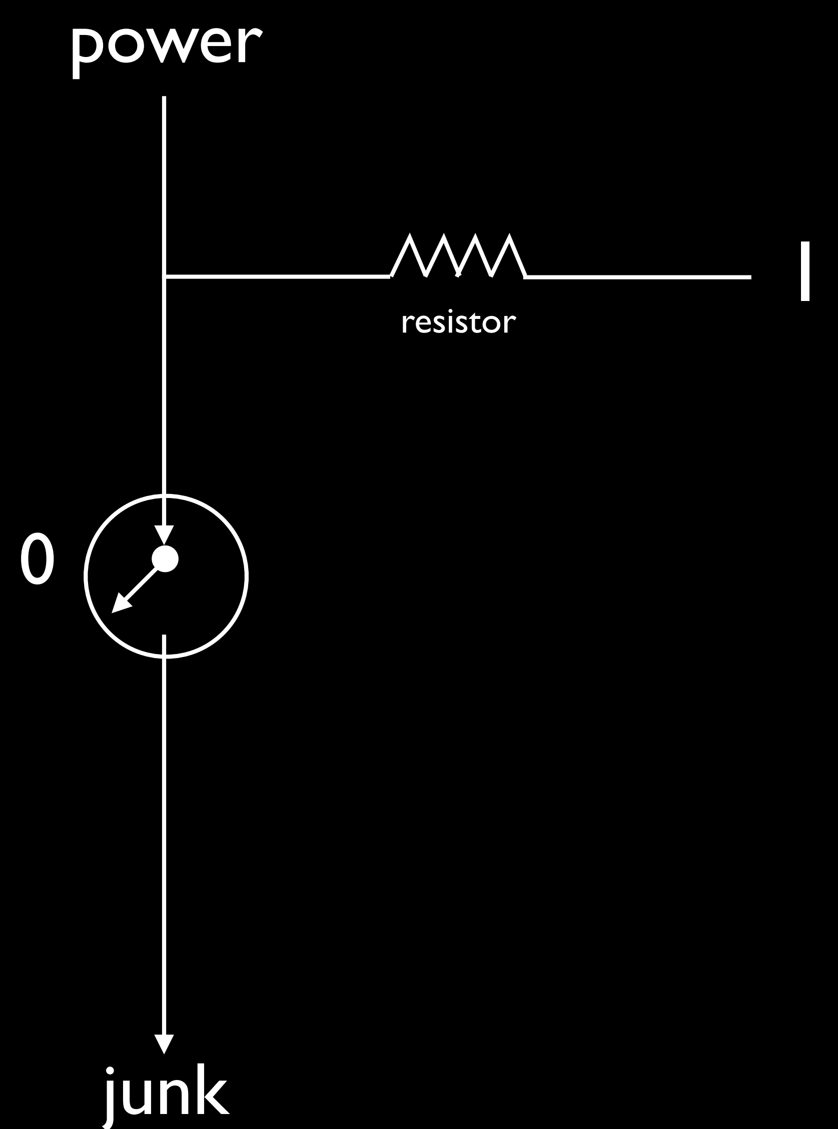
OR gate

Building Gates (transistors)



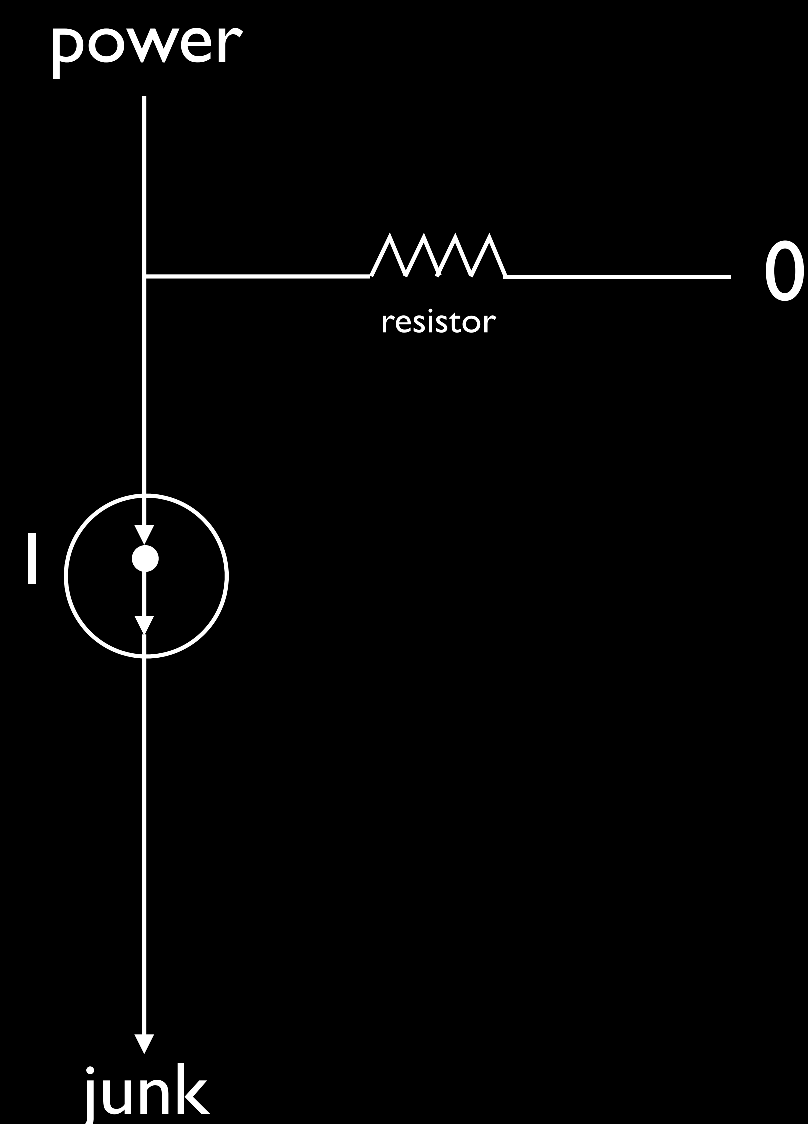
OR gate

Building Gates (transistors)



NOT gate

Building Gates (transistors)



NOT gate

Circuits

A circuit is a collection of logical gates that transforms a set of binary inputs into a set of binary outputs.

1-Bit Compare for Equality (CE)

If two bits *a*, *b* are equal then
    return 1
else
    return 0

1-Bit Compare for Equality (CE)

input		output
a	b	c
0	0	
0	1	
1	0	
1	1	

1-Bit Compare for Equality (CE)

input		output
a	b	c
0	0	1
0	1	0
1	0	0
1	1	1

1-Bit Compare for Equality (CE)

input		output	
a	b	c	sub-expression
0	0	1	
0	1	0	
1	0	0	
1	1	1	

1-Bit Compare for Equality (CE)

a	b	c	sub-expression
0	0	1	$a' \cdot b'$
0	1	0	
1	0	0	
1	1	1	$a \cdot b$

input output

1-Bit Compare for Equality (CE)

input		output	
a	b	c	sub-expression
0	0	1	$a' \cdot b'$
0	1	0	
1	0	0	
1	1	1	$a \cdot b$

$$c = (a' \cdot b') + (a \cdot b)$$

1-Bit Compare for Equality (CE)

input

a

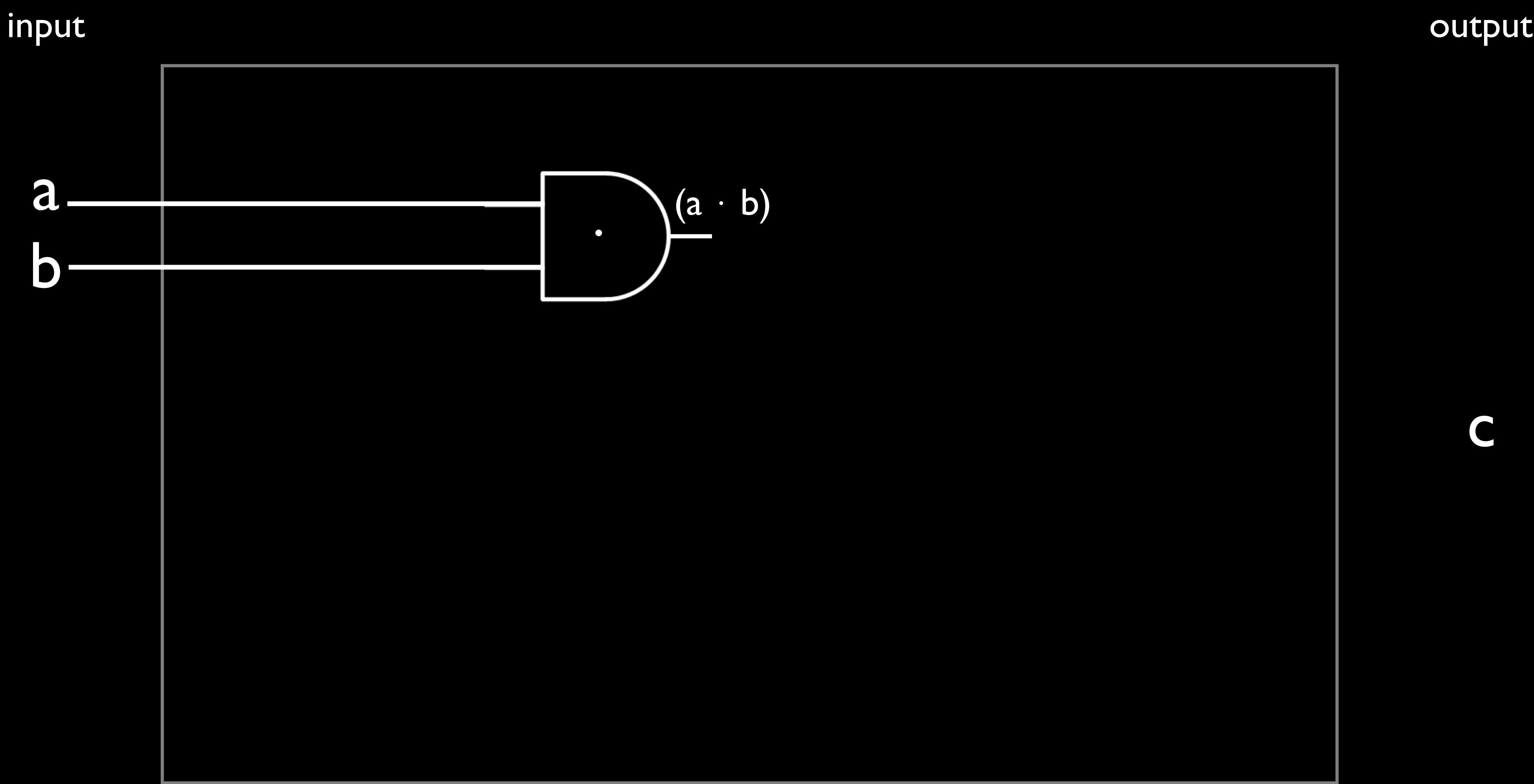
b

output

c

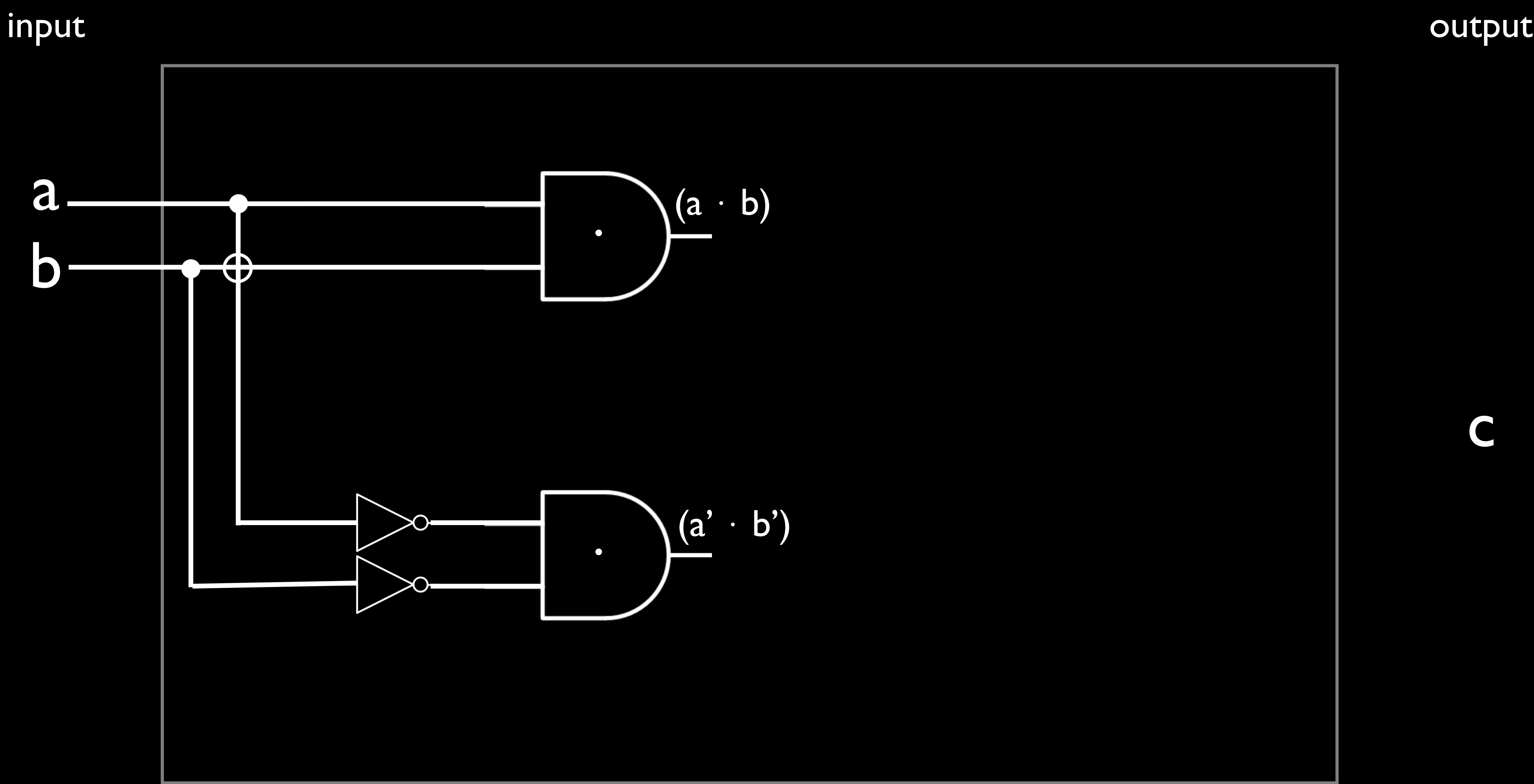
$$c = (a' \cdot b') + (a \cdot b)$$

1-Bit Compare for Equality (CE)



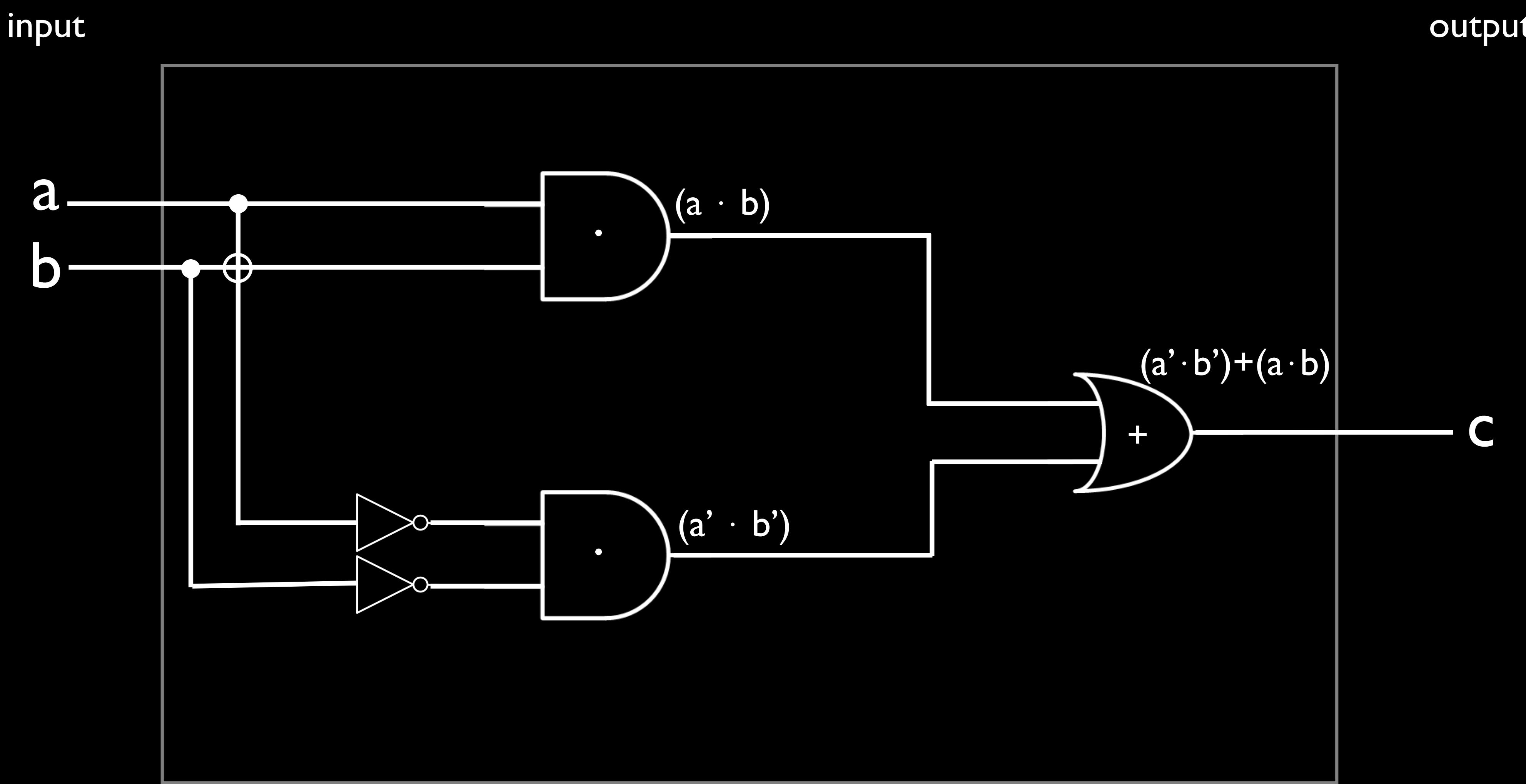
$$c = (a' \cdot b') + (a \cdot b)$$

1-Bit Compare for Equality (CE)



$$c = (a' \cdot b') + (a \cdot b)$$

1-Bit Compare for Equality (CE)



$$c = (a' \cdot b') + (a \cdot b)$$

Designing Circuits

step 1. build truth-table for all possible input/output values

step 2. build sub-expressions with *and/not* for each output column

step 3. combine, two at a time, sub-expressions with an *or*

step 4. draw circuit diagram

1-Bit Adder

build a circuit that adds two 1-bit numbers

1-Bit Adder

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = ?

1-Bit Adder

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 10$ *need to carry*

1-Bit Adder

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

input: two digits *a*, *b* and a carry *c*

1-Bit Adder

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

output: sum *d* and carry e

1-Bit Adder

5	101	2^2+2^0
1	001	2^0
---	---	
6		

1-Bit Adder

```
               1
  5          101
  1          001
 ---         ---
  6            0
```

1-Bit Adder

```
          1
  5     101
  1     001
---     ---
  6      10
```

1-Bit Adder

```
       1
5     101
1     001
---   ---
6     110
```

2^2+2^1

1-Bit Adder

a	b	c	d	e
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

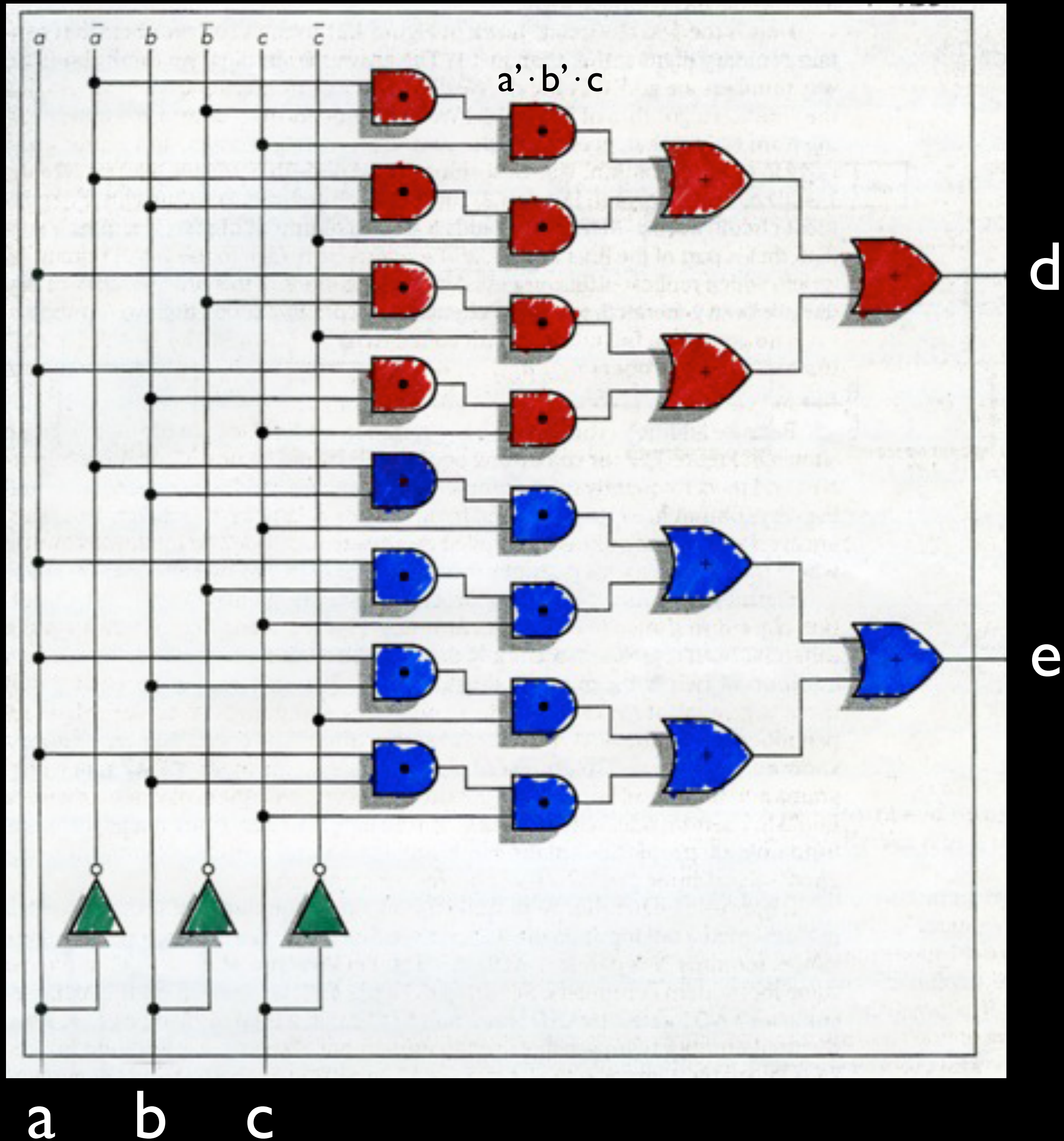
input: digits *a*, *b* and carry *c*
output: sum *d* and carry e

1-Bit Adder

a	b	c	d	e	sub-expressions (d)	sub-expressions (e)
0	0	0	0	0		
0	0	1	1	0	$a' \cdot b' \cdot c$	
0	1	0	1	0	$a' \cdot b \cdot c'$	
0	1	1	0	1		$a' \cdot b \cdot c$
1	0	0	1	0	$a \cdot b' \cdot c'$	
1	0	1	0	1		$a \cdot b' \cdot c$
1	1	0	0	1		$a \cdot b \cdot c'$
1	1	1	1	1	$a \cdot b \cdot c$	$a \cdot b \cdot c$

$$e = (a' \cdot b \cdot c) + (a \cdot b' \cdot c) + (a \cdot b \cdot c') + (a \cdot b \cdot c)$$

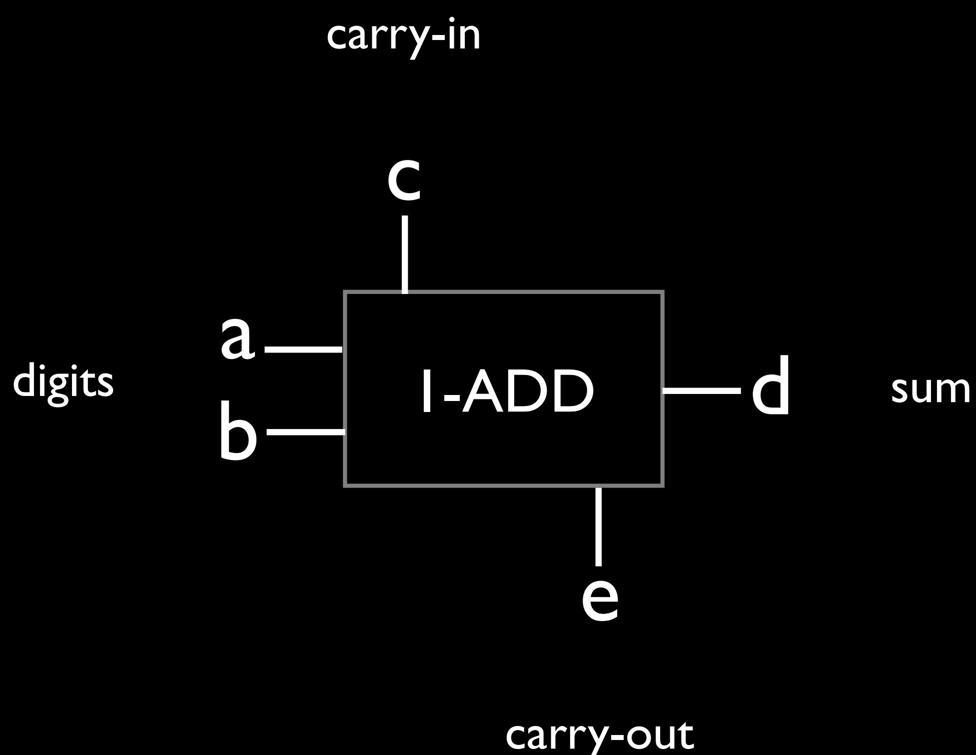
1-Bit Adder



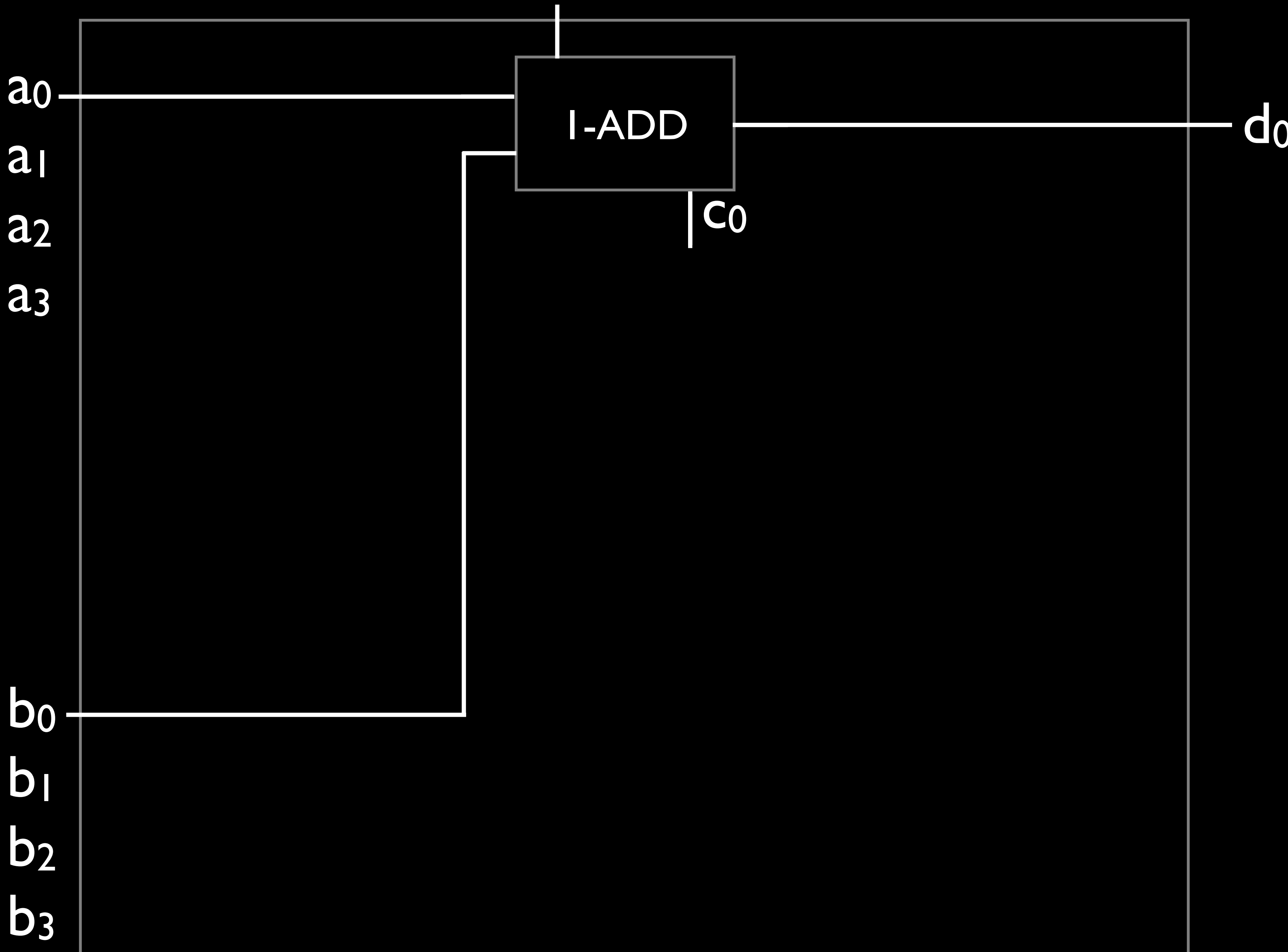
4-Bit Adder

build a circuit that adds two 4-bit numbers

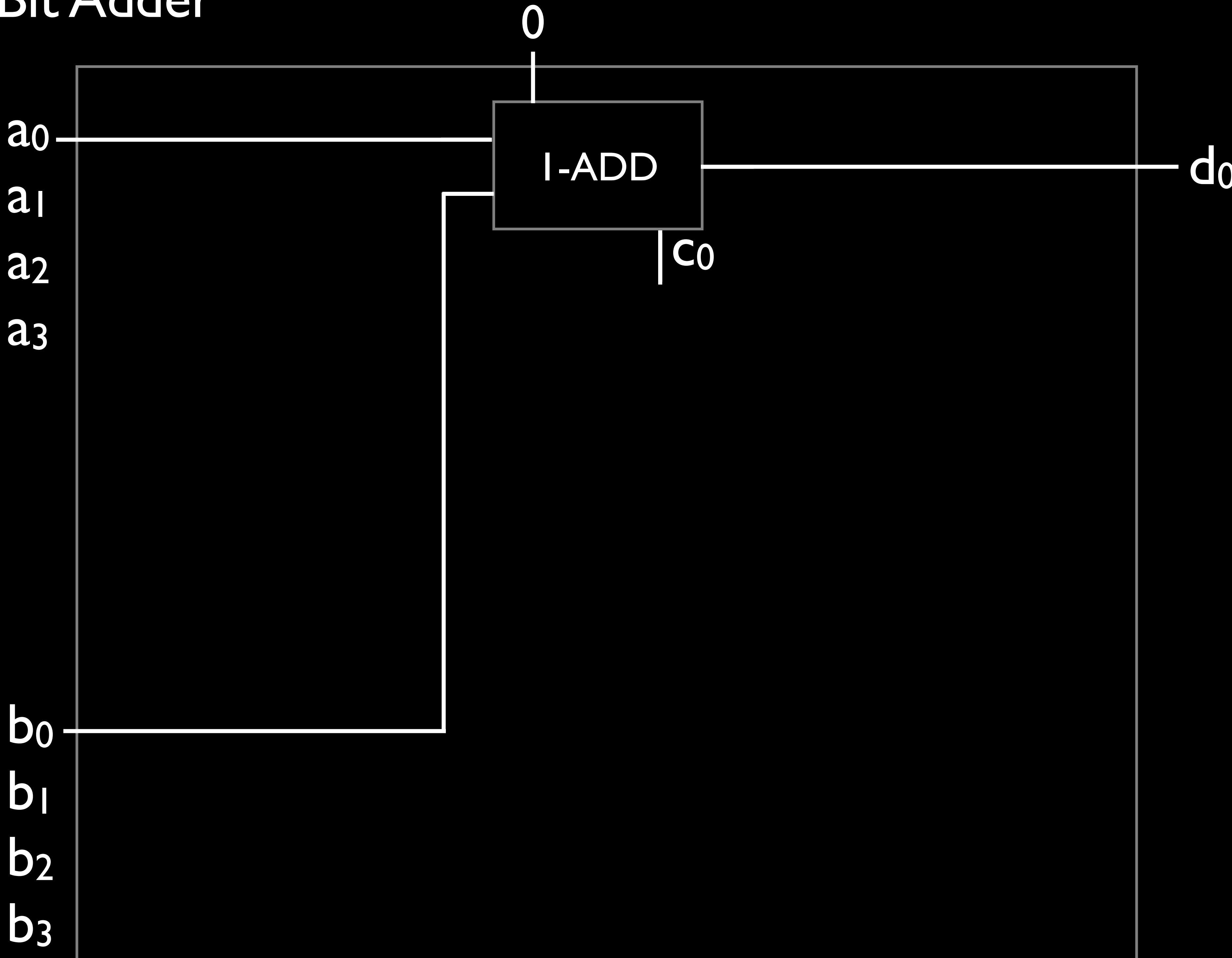
4-Bit Adder



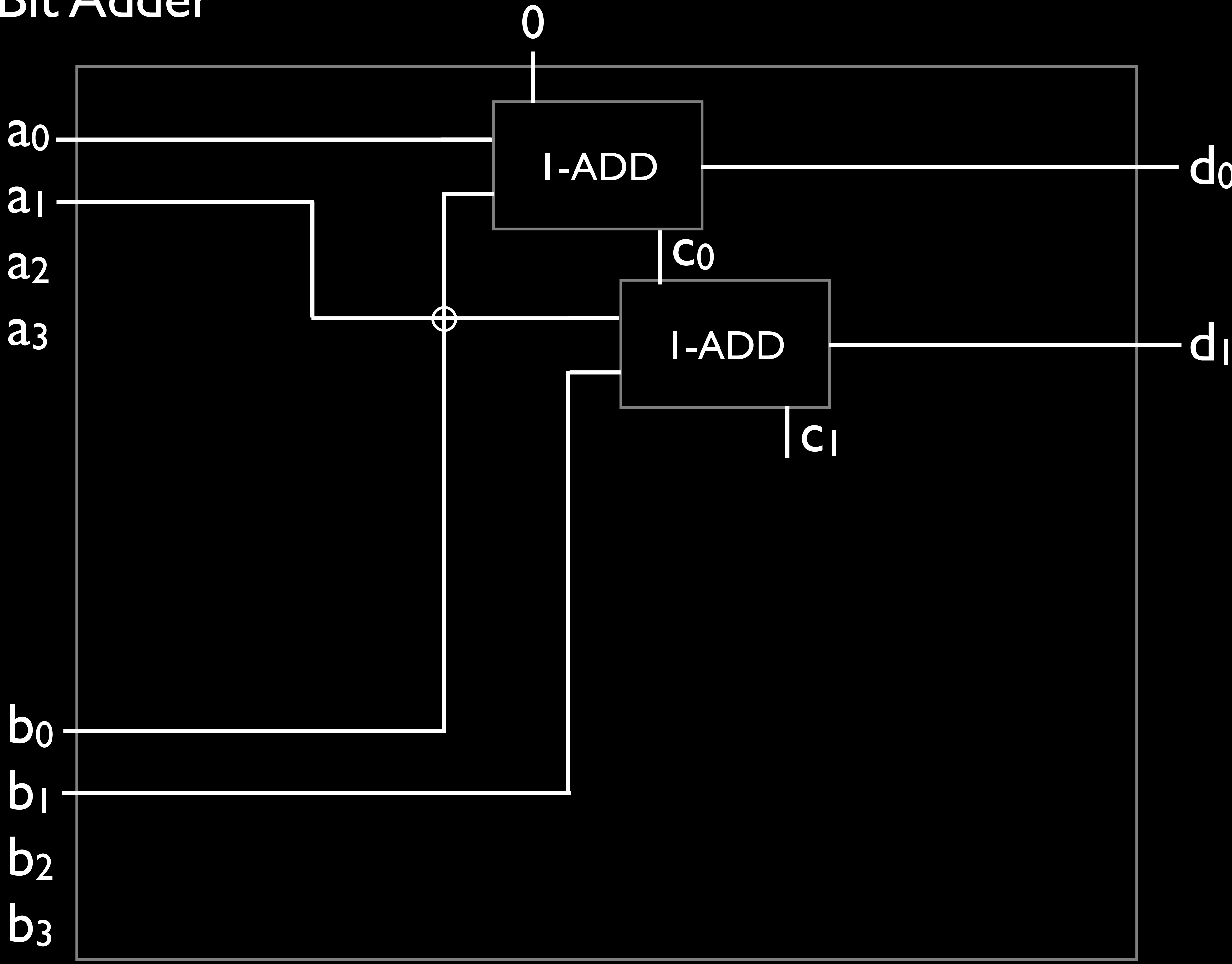
4-Bit Adder



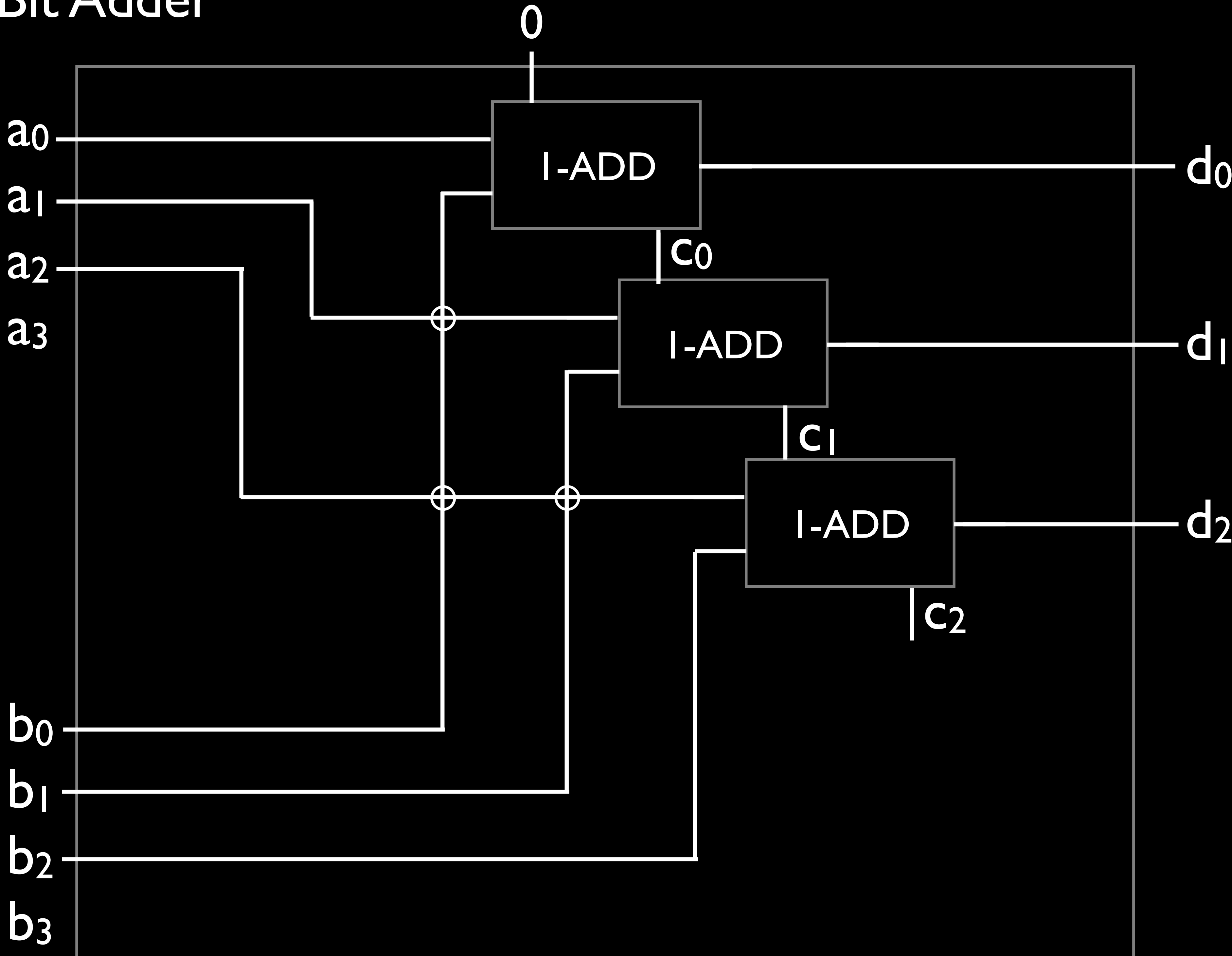
4-Bit Adder



4-Bit Adder



4-Bit Adder



4-Bit Adder