CMP 334 (2/25/19)

HW 6 (unsigned binary subtraction)

Signed binary arithmetic (introduction)

TOY assembly language

- Other ALU ops: **`sub`**, **`and`**, **`nor`**

Combinational circuit design process

- 1 bit full adder circuits

Building block circuits:

- Inverters
- Decoders
- Multiplexers

HW 6: Unsigned Binary Subtraction

For each of the <**X**, **Y**> pairs below:

a) Convert **X** and **Y** → binary
b) Compute **Ÿ**, the 2's complement of **Y**
c) Compute 8-bit **diff** using 2's complement addition
d) Convert **diff** → hexadecimal
e) Indicate whether 8-bit subtraction produces a **carry**
f) Convert **X**, **Y**, **Ÿ**, **diff** → decimal (check your work)

Where <**X**, **Y**> =

1) <0x**4F**, 0x**6D**>
2) <0x**C8**, 0x**2B**>
3) <0x**A3**, 0x**95**>
4) <0x**B4**, 0x**E1**>

HW 6 #1

$$X = 0x\mathbf{4F}, \qquad Y = 0x\mathbf{6D}$$

$$X = 0b\mathbf{01001111}, \qquad Y = 0b\mathbf{01101101}$$

$$\overline{Y} = 10010010$$

$$\dddot{Y} = \overline{Y}+1 = \mathbf{10010011}$$

$$X \overset{\cdots}{-} Y = \begin{array}{r} \mathbf{01001111} \\ \underline{\mathbf{10010011}} \end{array}$$

carry

$$\begin{array}{l} 0\,0\,0\,1\,1\,1\,1\,1 \\ \mathbf{011100010} \end{array} \rightarrow 0x\mathbf{E2} \quad (+\ 0x\mathbf{0})$$

$$X = \mathbf{4}\cdot 16 + \mathbf{15} = \mathbf{79}$$

$$Y = \mathbf{6}\cdot 16 + \mathbf{13} = \mathbf{109}$$

$$\dddot{Y} = \mathbf{9}\cdot 16 + \mathbf{3} = \mathbf{147}$$

$$X \overset{\cdots}{-} Y = \mathbf{14}\cdot 16 + \mathbf{2} = \mathbf{226}$$

$$X - Y = \mathbf{-30} = \mathbf{226} - \mathbf{256}$$

HW 6 #2

$$
\begin{aligned}
X &= \mathtt{0xC8}, & Y &= \mathtt{0x2B} \\
X &= \mathtt{0b11001000}, & Y &= \mathtt{0b00101011} \\
& & \overline{Y} &= \phantom{\mathtt{0b}}11010100 \\
\dddot{Y} &= & \overline{Y}+1 &= \phantom{\mathtt{0b}}11010101
\end{aligned}
$$

$$
\begin{array}{rl}
X \mathbin{\dddot{-}} Y = & 11001000 \\
 & \underline{11010101} \\
 & \scriptstyle 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0 \\
\text{carry} & 110011101 \rightarrow \mathtt{0x9D}\ (+\ \mathtt{0x100})
\end{array}
$$

$$
\begin{aligned}
X &= 12 \cdot 16 + 8 = 200 \\
Y &= 2 \cdot 16 + 11 = 43 \\
\dddot{Y} &= 13 \cdot 16 + 5 = 213 \\
X \mathbin{\dddot{-}} Y &= 9 \cdot 16 + 13 = 157 \\
X - Y &= 157 \qquad = 157
\end{aligned}
$$

HW 6 #3

$$X = 0x\mathrm{A3}, \qquad Y = 0x95$$

$$X = 0b10100011, \quad Y = 0b10010101$$

$$\overline{Y} = 01101010$$

$$\dddot{Y} = \overline{Y}+1 = 01101011$$

$$X \overset{\cdots}{-} Y = \begin{array}{r} 10100011 \\ \underline{01101011} \end{array}$$

carry

$$\begin{array}{r} 1\;1\;1\;0\;0\;0\;1\;1 \\ 100001110 \end{array} \rightarrow 0x\mathrm{0E} \; (+\; 0x100)$$

$$\begin{aligned}
X &= 10 \cdot 16 + 3 = 163 \\
Y &= 9 \cdot 16 + 5 = 149 \\
\dddot{Y} &= 6 \cdot 16 + 10 = 106 \\
X \overset{\cdots}{-} Y &= 0 \cdot 16 + 14 = 14 \\
X - Y &= 14 \qquad\qquad = 14
\end{aligned}$$

HW 6 #4

$$X = \mathtt{0xB4}, \qquad Y = \mathtt{0xE1}$$

$$X = \mathtt{0b10110100}, \qquad Y = \mathtt{0b11100001}$$

$$\overline{Y} = 00011110$$

$$\dddot{Y} = \overline{Y}+1 = 00011111$$

$$X \overset{\cdots}{-} Y = \begin{array}{r} 10110100 \\ \underline{00011111} \end{array}$$

carry

0 0 1 1 1 1 0 0

$$011010011 \rightarrow \mathtt{0xD3} \ (+\ \mathtt{0x0})$$

$$X = 11 \cdot 16 + 4 = 180$$

$$Y = 14 \cdot 16 + 1 = 225$$

$$\dddot{Y} = 1 \cdot 16 + 15 = 31$$

$$X \overset{\cdots}{-} Y = 13 \cdot 16 + 3 = 211$$

$$X - Y = -45 \qquad = 211 - 256$$

Unsigned **n**-bit Subtraction ($0 == 2^n$)

A ≥ B

$$A \overset{\dots}{-} B \equiv \mathit{A - B} \text{ (natural numbers)}$$

A < B *A – B undefined on natural numbers*

$$\overset{\dots}{B} \equiv 2^n - B = (2^n - 1 - B) + 1 = \overline{B} + 1$$

$$B \overset{\dots}{-} B = B \overset{\dots}{+} \overset{\dots}{B} = 0 \qquad (0 == 2^n)$$

$$(A \overset{\dots}{-} B) \overset{\dots}{+} C = (A \overset{\dots}{+} C) \overset{\dots}{-} B$$

$$\boxed{A \overset{\dots}{-} B \equiv A \overset{\dots}{+} (\overline{B}+1)}$$

n-bit Binary Numbers

Unsigned: $b_{n-1}b_{n-2} \ldots b_1\ b_0$ $\quad$ ($b_i = 0$ or $b_i = 1$)

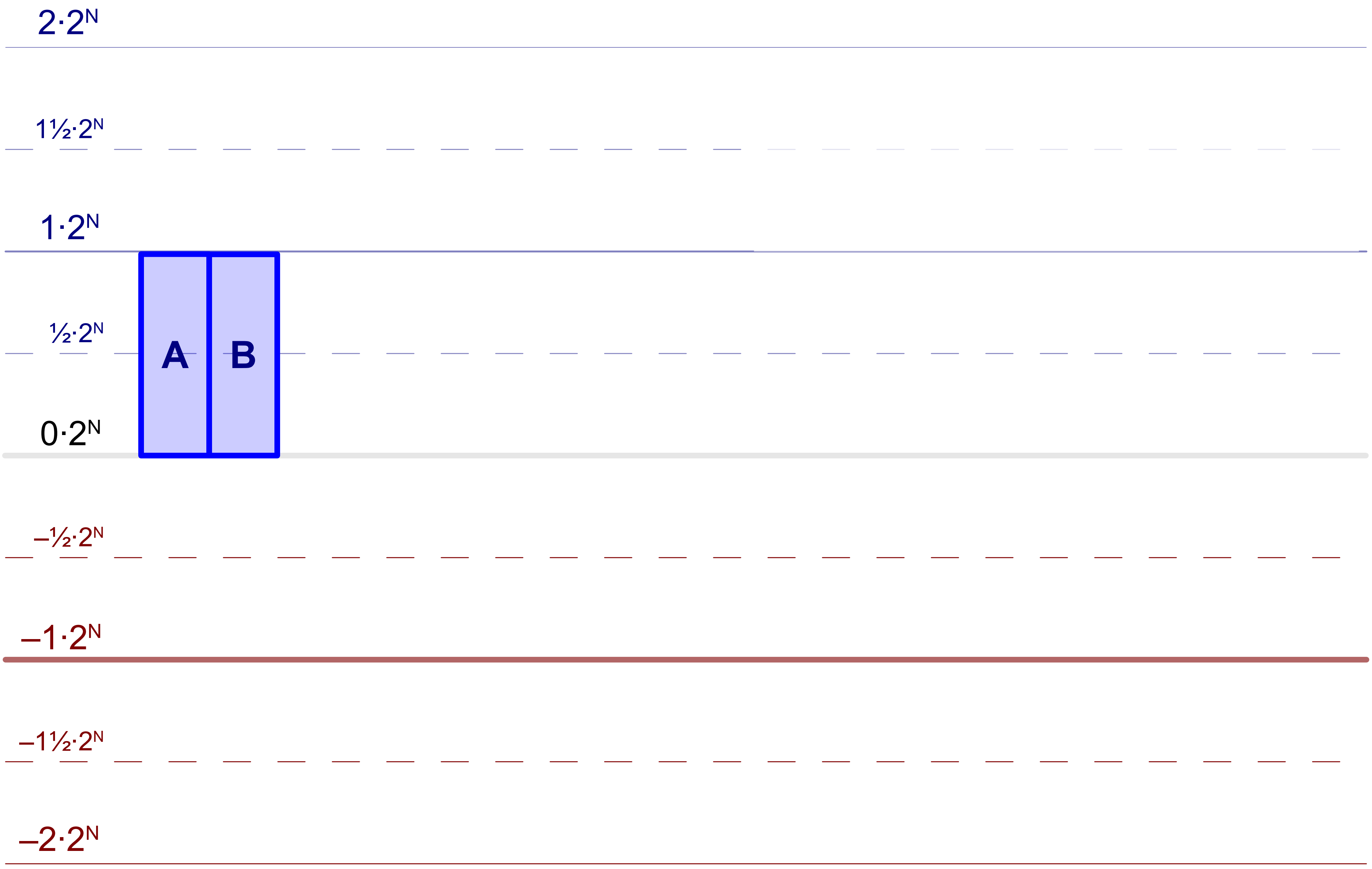
value: $b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_1 2^1 + b_0 2^0$

range: $[0\ ..\ 2^n\text{–}1]$

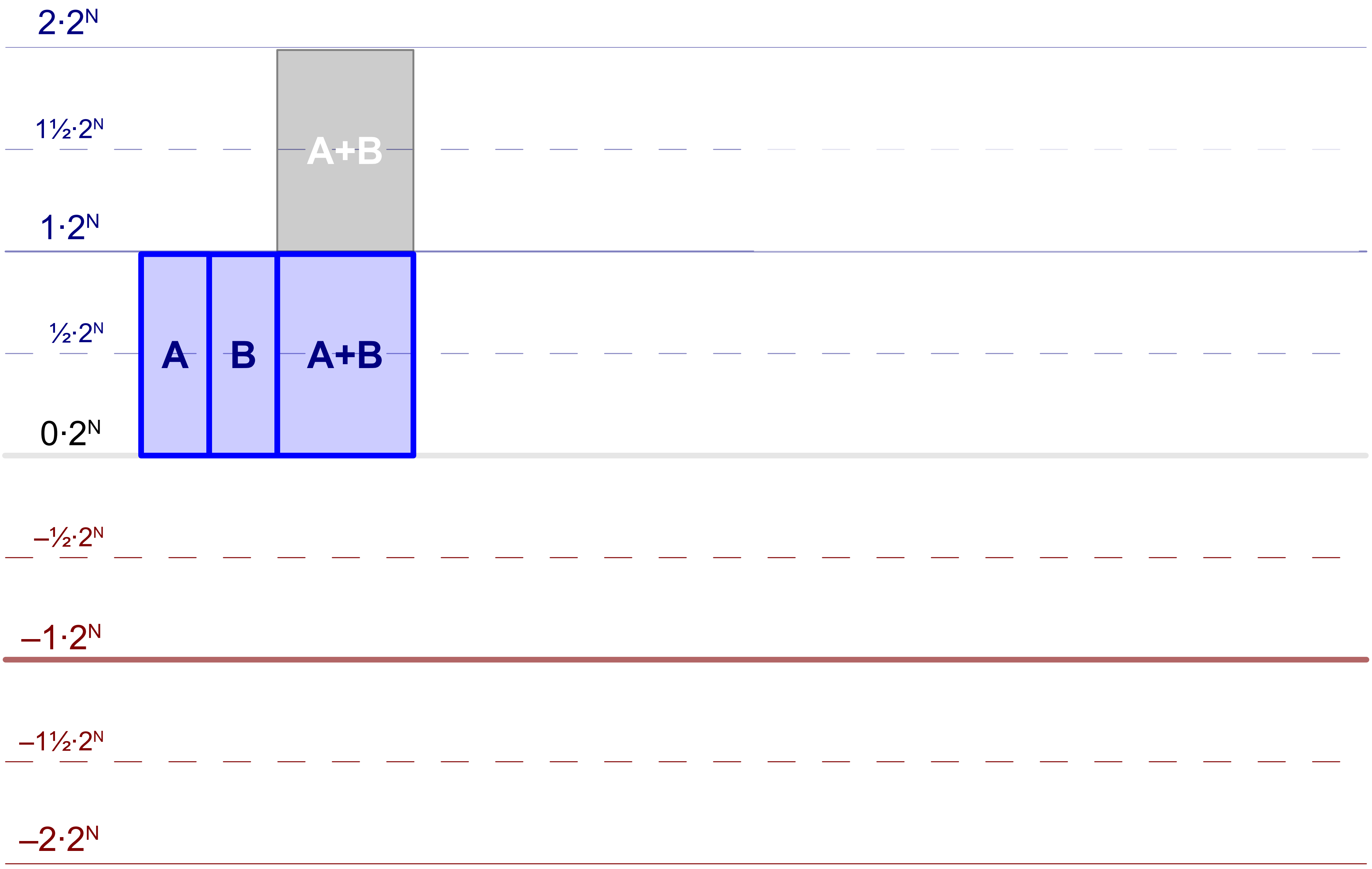
n-bit sum: $\mathbf{A} \overset{\cdots}{+} \mathbf{B} + \mathbf{c}\cdot\mathbf{2}^n = \mathbf{A} + \mathbf{B}$

n-bit diff: $\mathbf{A} \overset{\cdots}{-} \mathbf{B} \equiv \mathbf{A} \overset{\cdots}{+} (\mathbf{2}^n - \mathbf{B}) = \mathbf{A} \overset{\cdots}{+} \overline{\mathbf{B}}+1 = \mathbf{A}\text{–}\mathbf{B} + \mathbf{2}^n\text{–}\mathbf{c}\cdot\mathbf{2}^n$

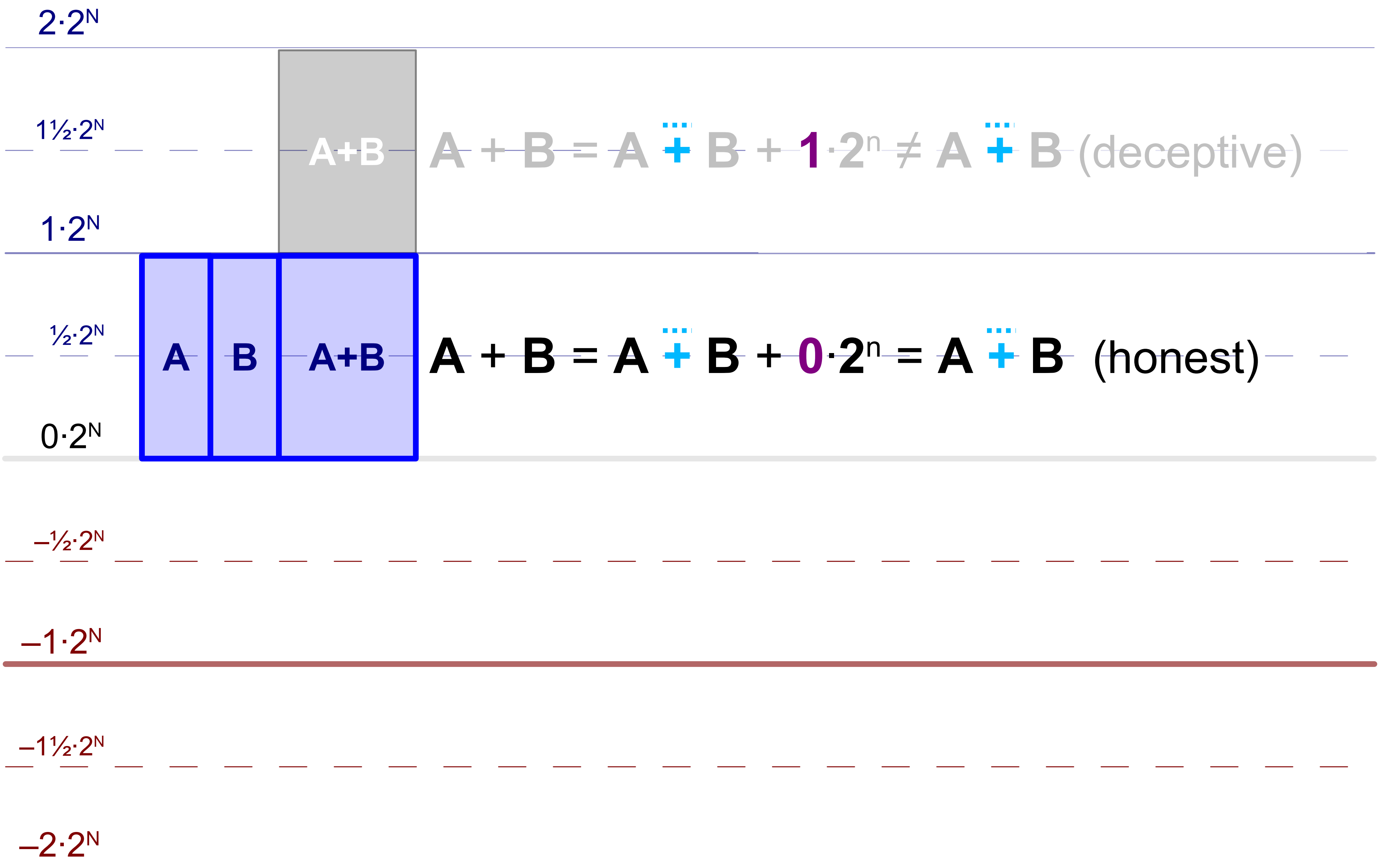
Unsigned Integers



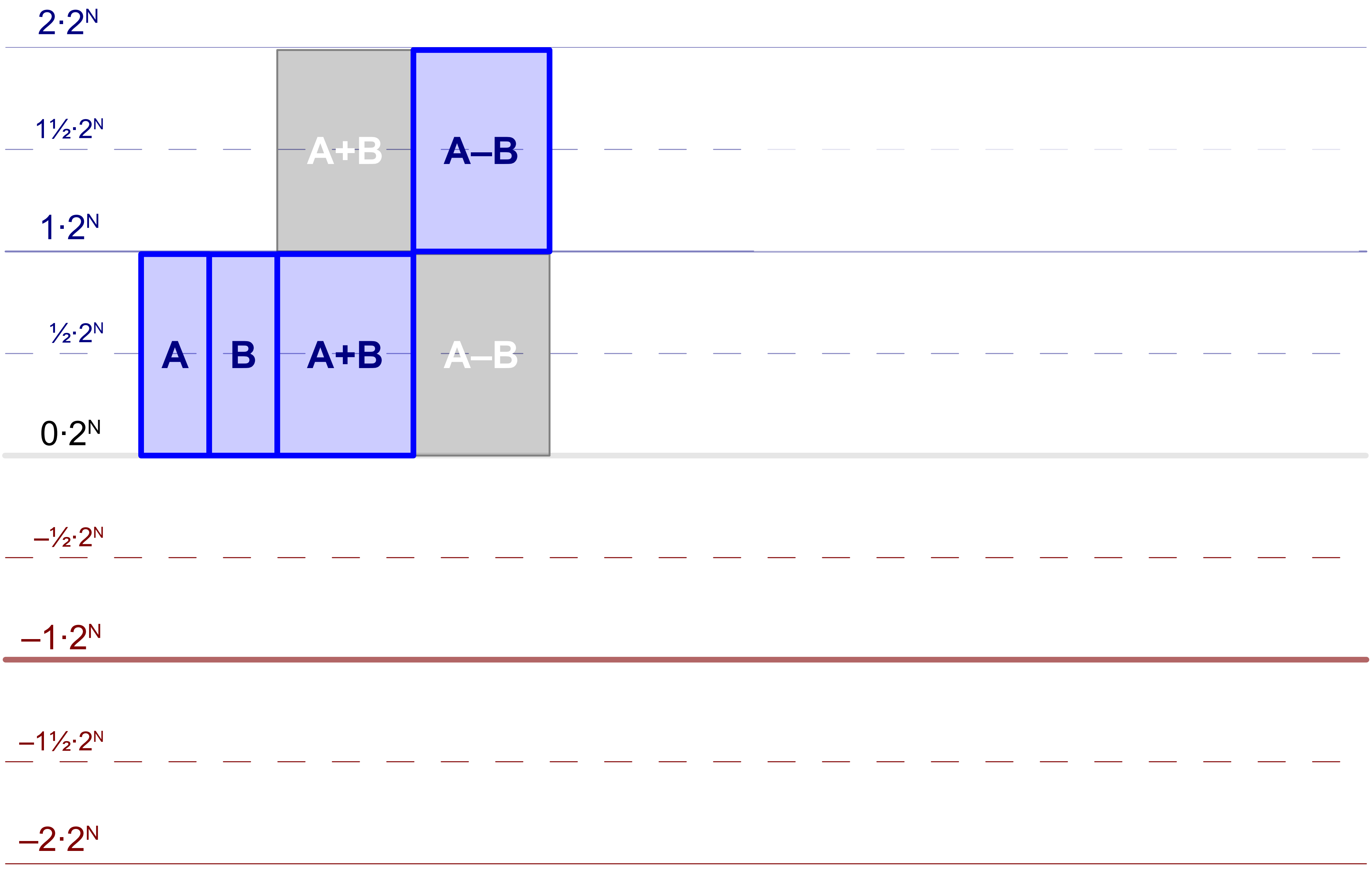
Unsigned Sum



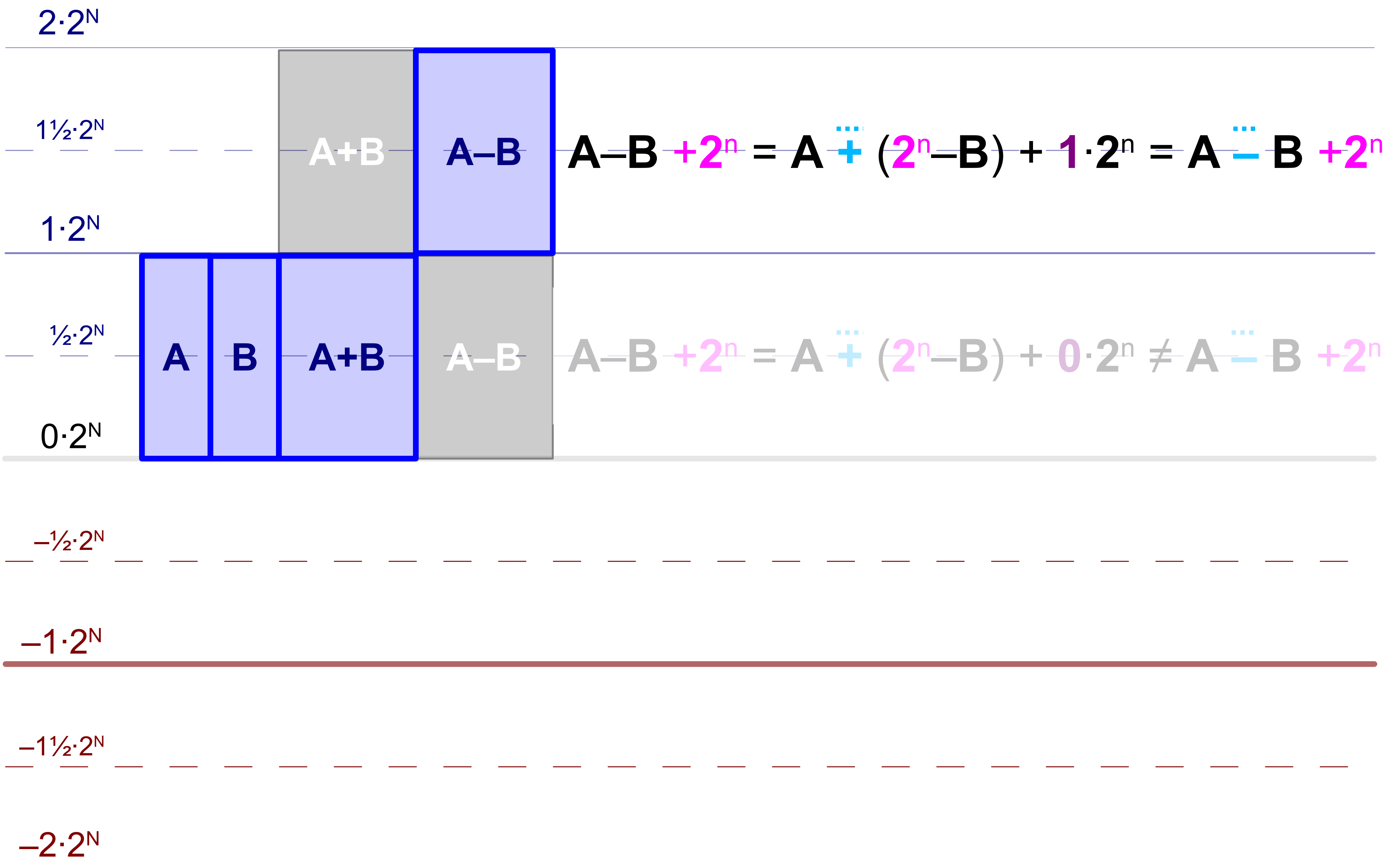
Unsigned Sum



Unsigned Sum and Difference



Unsigned Sum and Difference



n-bit Binary Numbers

Unsigned: $b_{n-1}b_{n-2} \ldots b_1\, b_0$ $\quad$ $(b_i = 0 \text{ or } b_i = 1)$

value: $b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_1 2^1 + b_0 2^0$

range: $[0 \,..\, 2^n\!-\!1]$

n-bit sum: $\mathbf{A} \overset{\cdots}{+} \mathbf{B} + \mathbf{c}\cdot\mathbf{2}^n = \mathbf{A} + \mathbf{B}$

n-bit diff: $\mathbf{A} \overset{\cdots}{-} \mathbf{B} \equiv \mathbf{A} \overset{\cdots}{+} (\mathbf{2}^n - \mathbf{B}) = \mathbf{A} \overset{\cdots}{+} \overline{\mathbf{B}}+1 = \mathbf{A}-\mathbf{B} + \mathbf{2}^n - \mathbf{c}\cdot\mathbf{2}^n$

Signed: $b_{n-1}b_{n-2} \ldots b_1\, b_0$

value: $-b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_1 2^1 + b_0 2^0$

range: $[-2^{n-1} \,..\, 2^{n-1}\!-\!1]$

n-bit Binary Numbers

Unsigned: $b_{n-1}b_{n-2} \dots b_1\, b_0$ $(b_i = 0 \text{ or } b_i = 1)$

value: $b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \dots + b_1 2^1 + b_0 2^0$

range: $[0 \,..\, 2^n - 1]$

n-bit sum: $\mathbf{A} \dot{+} \mathbf{B} + \mathbf{c} \cdot \mathbf{2}^n = \mathbf{A} + \mathbf{B}$

n-bit diff: $\mathbf{A} \dot{-} \mathbf{B} \equiv \mathbf{A} \dot{+} (\mathbf{2}^n - \mathbf{B}) = \mathbf{A} \dot{+} \overline{\mathbf{B}} + 1 = \mathbf{A} - \mathbf{B} + \mathbf{2}^n - \mathbf{c} \cdot \mathbf{2}^n$

Signed: $b_{n-1}b_{n-2} \dots b_1\, b_0$

value: $-b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \dots + b_1 2^1 + b_0 2^0$

range: $[-2^{n-1} \,..\, 2^{n-1} - 1]$

n-bit sum: $\mathbf{A} \dot{+} \mathbf{B} = \mathbf{A} + \mathbf{B}$ **iff $\mathbf{v}$=0**

Whole Numbers (binary)

...001000	+8
...000111	+7
...000110	+6
...000101	+5
...000100	+4
...000011	+3
...000010	+2
...000001	+1
...000000	+0
...111111	−1
...111110	−2
...111101	−3
...111100	−4
...111011	−5
...111010	−6
...111001	−7
...111000	−8

unbounded integers

... –2, –1, 0, 1, 2, ...

Closed under addition

Closed under subtraction

“Sign”

Negative integers => leading 1's

Positive integers => leading 0's

0 is an honorary positive integer

0 + 0 = 0	3 + -3 = 0	6 + -6 = 0
1 + -1 = 0	4 + -4 = 0	7 + -7 = 0
2 + -2 = 0	5 + -5 = 0	8 + -8 = 0

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
.000001	+1
.000000	+0
.111111	−1
.111110	−2
.111101	−3
.111100	−4
.111011	−5
.111010	−6
.111001	−7
.111000	−8

THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
.000001	+1
.000000	+0
.111111	−1
.111110	−2
.111101	−3
.111100	−4
.111011	−5
.111010	−6
.111001	−7
.111000	−8

THEROEM:

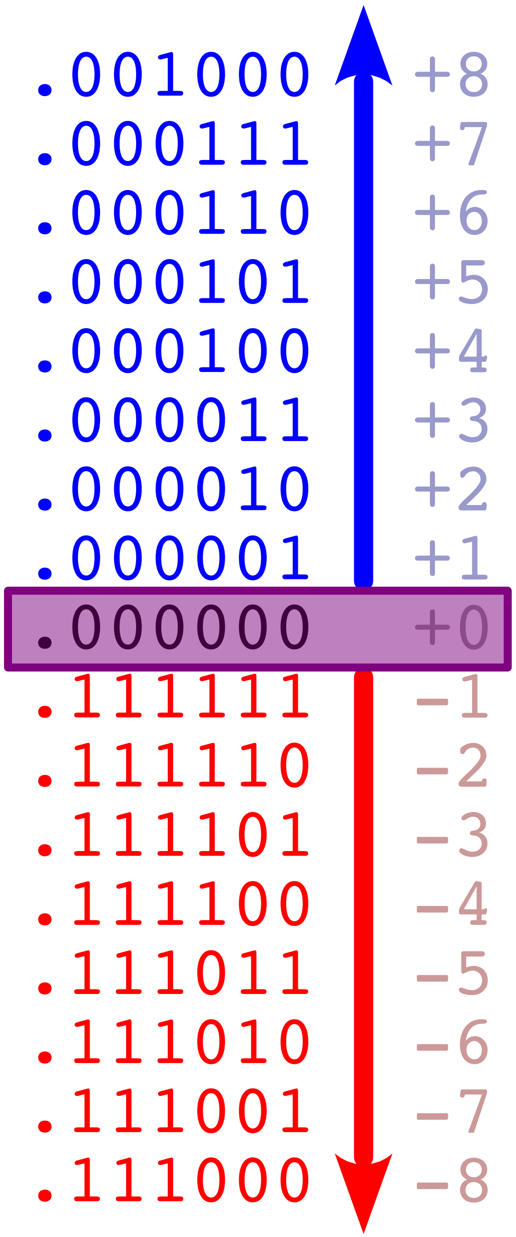
If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

Additive Inverse



THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

– .0000

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
.000001	+1
$\boxed{.000000 \quad +0}$	
.111111	−1
.111110	−2
.111101	−3
.111100	−4
.111011	−5
.111010	−6
.111001	−7
.111000	−8

THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$-\ .0000 = \overline{.0000} + 1$$

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
.000001	+1
.000000	+0
.111111	−1
.111110	−2
.111101	−3
.111100	−4
.111011	−5
.111010	−6
.111001	−7
.111000	−8

THEROEM:

If **X** and **Y** are signed integers then

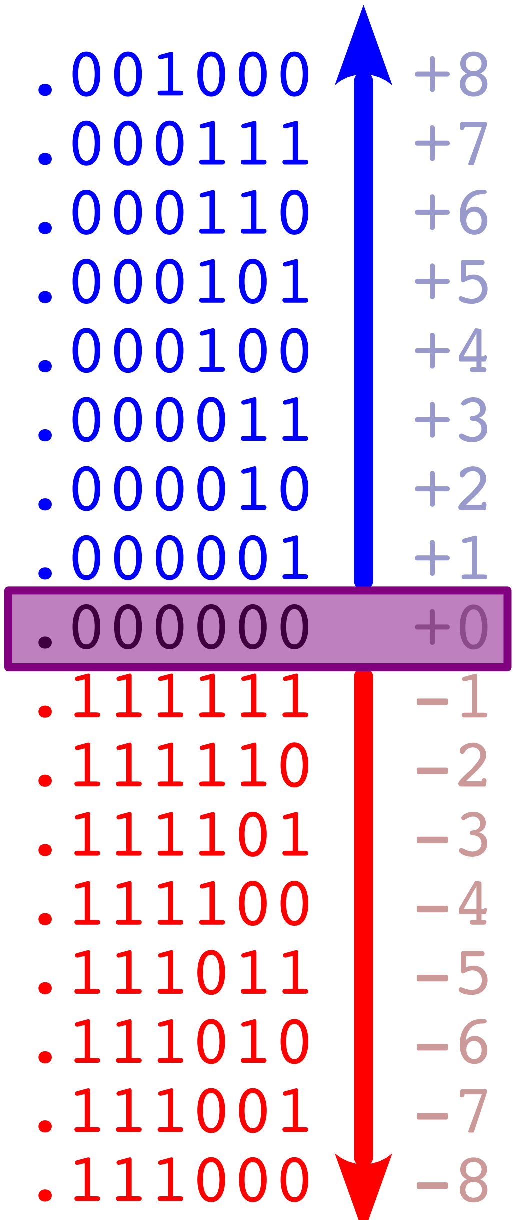
$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$-\ .0000 = \overline{.0000} + 1 = .1111 + 1$$

Additive Inverse



THEROEM:

If **X** and **Y** are signed integers then

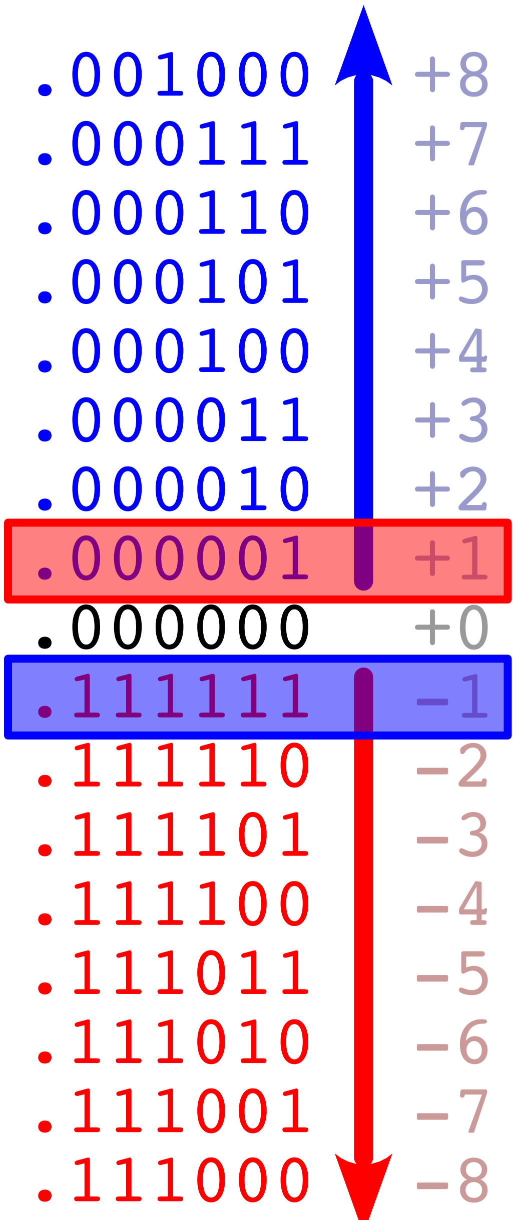
$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$-\ .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

Additive Inverse



THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

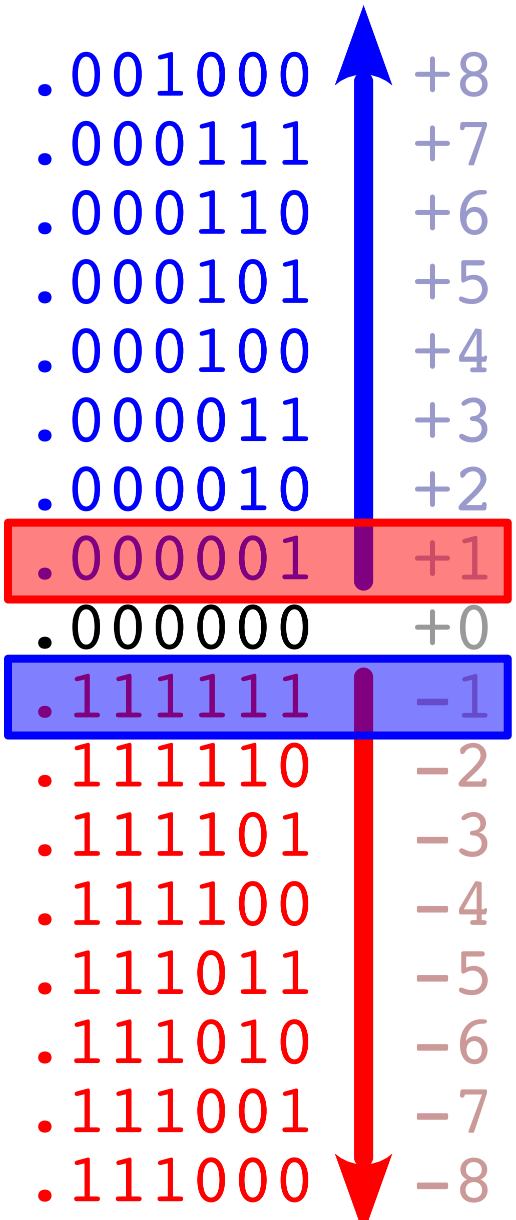
Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$- .0001$

$- .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$

Additive Inverse



THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

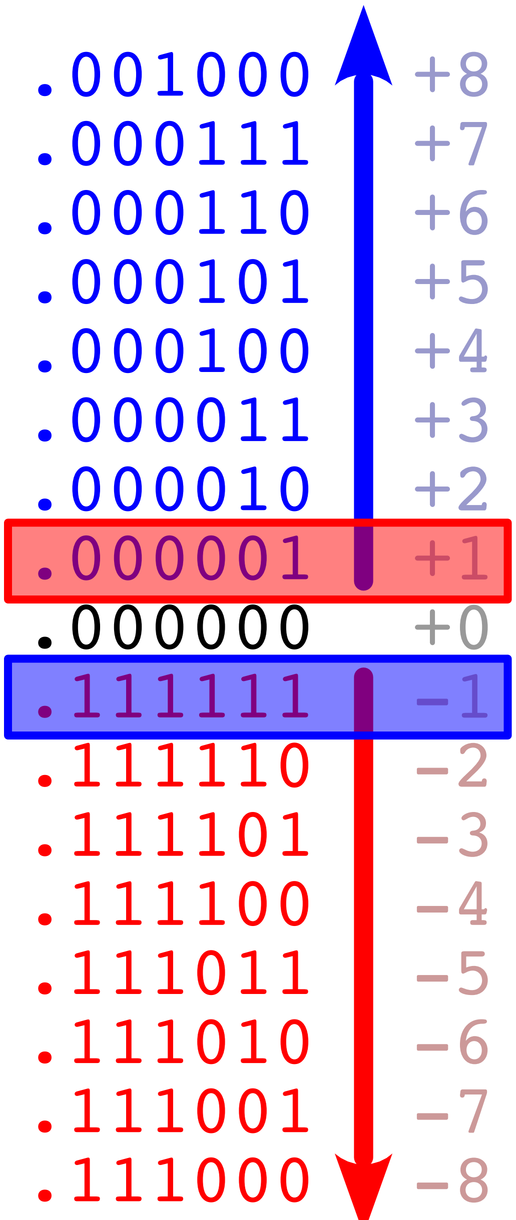
Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$-\ .0001 = \overline{.0001} + 1$$

$$-\ .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

Additive Inverse



THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$-\ .0001 = \overline{.0001} + 1 = .1110 + 1$$

$$-\ .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
.000001	+1
.000000	+0
.111111	−1
.111110	−2
.111101	−3
.111100	−4
.111011	−5
.111010	−6
.111001	−7
.111000	−8

THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

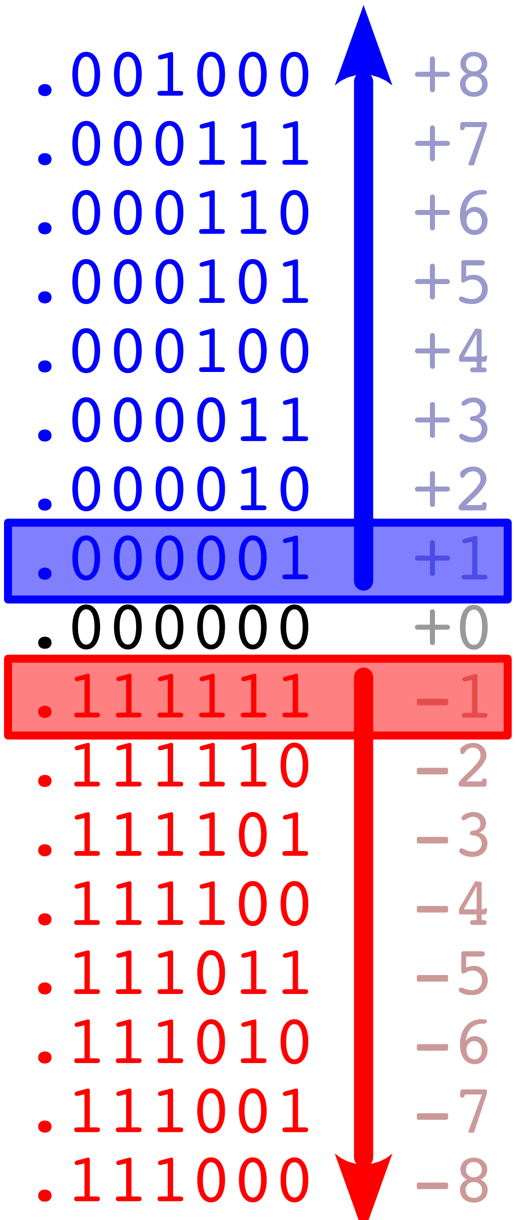
Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$-\ .0001 = \overline{.0001} + 1 = .1110 + 1 = .1111$$

$$-\ .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

Additive Inverse



THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$-\ .0001 = \overline{.0001} + 1 = .1110 + 1 = .1111$

$-\ .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$

$-\ .1111$

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
.000001	+1
.000000	+0
.111111	−1
.111110	−2
.111101	−3
.111100	−4
.111011	−5
.111010	−6
.111001	−7
.111000	−8

THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$-\ .0001 = \overline{.0001} + 1 = .1110 + 1 = .1111$$

$$-\ .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

$$-\ .1111 = \overline{.1111} + 1$$

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
$\boxed{.000001}$	$\boxed{+1}$
.000000	+0
$\boxed{.111111}$	$\boxed{-1}$
.111110	-2
.111101	-3
.111100	-4
.111011	-5
.111010	-6
.111001	-7
.111000	-8

THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$- .0001 = \overline{.0001} + 1 = .1110 + 1 = .1111$$

$$- .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

$$- .1111 = \overline{.1111} + 1 = .0000 + 1$$

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
.000001	+1
.000000	+0
.111111	−1
.111110	−2
.111101	−3
.111100	−4
.111011	−5
.111010	−6
.111001	−7
.111000	−8

THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$-\ .0001 = \overline{.0001} + 1 = .1110 + 1 = .1111$$

$$-\ .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

$$-\ .1111 = \overline{.1111} + 1 = .0000 + 1 = .0001$$

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
.000001	+1
.000000	+0
.111111	−1
.111110	−2
.111101	−3
.111100	−4
.111011	−5
.111010	−6
.111001	−7
.111000	−8

THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

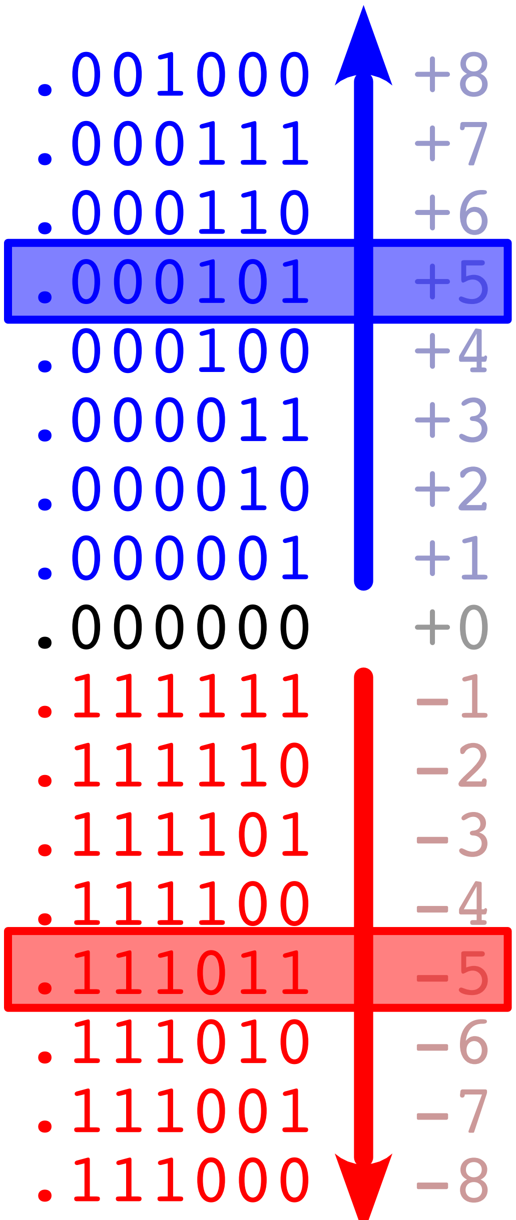
$$-\,.0001 = \overline{.0001} + 1 = .1110 + 1 = .1111$$

$$-\,.0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

$$-\,.1111 = \overline{.1111} + 1 = .0000 + 1 = .0001$$

$$-\,.1011$$

Additive Inverse



THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

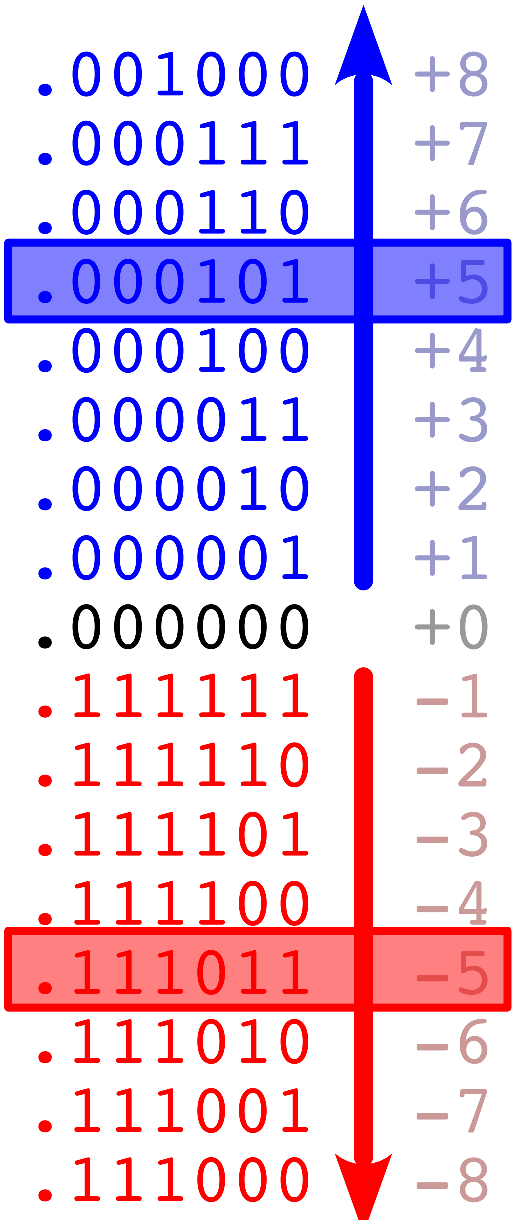
$$-\ .0001 = \overline{.0001} + 1 = .1110 + 1 = .1111$$

$$-\ .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

$$-\ .1111 = \overline{.1111} + 1 = .0000 + 1 = .0001$$

$$-\ .1011 = \overline{.1011} + 1$$

Additive Inverse



THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$-\,.0001 = \overline{.0001} + 1 = .1110 + 1 = .1111$$

$$-\,.0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

$$-\,.1111 = \overline{.1111} + 1 = .0000 + 1 = .0001$$

$$-\,.1011 = \overline{.1011} + 1 = .0100 + 1$$

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
.000001	+1
.000000	+0
.111111	−1
.111110	−2
.111101	−3
.111100	−4
.111011	−5
.111010	−6
.111001	−7
.111000	−8

THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$-\,.0001 = \overline{.0001} + 1 = .1110 + 1 = .1111$$

$$-\,.0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

$$-\,.1111 = \overline{.1111} + 1 = .0000 + 1 = .0001$$

$$-\,.1011 = \overline{.1011} + 1 = .0100 + 1 = .0101$$

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
$\boxed{.000101}$	$\boxed{+5}$
.000100	+4
.000011	+3
.000010	+2
.000001	+1
.000000	+0
.111111	−1
.111110	−2
.111101	−3
.111100	−4
$\boxed{.111011}$	$\boxed{-5}$
.111010	−6
.111001	−7
.111000	−8

THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$-\ .0101$

$-\ .0001 = \overline{.0001} + 1 = .1110 + 1 = .1111$

$-\ .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$

$-\ .1111 = \overline{.1111} + 1 = .0000 + 1 = .0001$

$-\ .1011 = \overline{.1011} + 1 = .0100 + 1 = .0101$

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
.000001	+1
.000000	+0
.111111	−1
.111110	−2
.111101	−3
.111100	−4
.111011	−5
.111010	−6
.111001	−7
.111000	−8

THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$-\ .0101 = \overline{.0101} + 1$$

$$-\ .0001 = \overline{.0001} + 1 = .1110 + 1 = .1111$$

$$-\ .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

$$-\ .1111 = \overline{.1111} + 1 = .0000 + 1 = .0001$$

$$-\ .1011 = \overline{.1011} + 1 = .0100 + 1 = .0101$$

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
.000001	+1
.000000	+0
.111111	−1
.111110	−2
.111101	−3
.111100	−4
.111011	−5
.111010	−6
.111001	−7
.111000	−8

THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$-\ .0101 = \overline{.0101} + 1 = .1010 + 1$$

$$-\ .0001 = \overline{.0001} + 1 = .1110 + 1 = .1111$$

$$-\ .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

$$-\ .1111 = \overline{.1111} + 1 = .0000 + 1 = .0001$$

$$-\ .1011 = \overline{.1011} + 1 = .0100 + 1 = .0101$$

Additive Inverse

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
.000001	+1
.000000	+0
.111111	−1
.111110	−2
.111101	−3
.111100	−4
.111011	−5
.111010	−6
.111001	−7
.111000	−8

THEROEM:

If **X** and **Y** are signed integers then

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

Corollary:

$$\mathbf{-Y = \overline{Y} + 1}$$

$$- .0101 = \overline{.0101} + 1 = .1010 + 1 = .1011$$

$$- .0001 = \overline{.0001} + 1 = .1110 + 1 = .1111$$

$$- .0000 = \overline{.0000} + 1 = .1111 + 1 = .0000$$

$$- .1111 = \overline{.1111} + 1 = .0000 + 1 = .0001$$

$$- .1011 = \overline{.1011} + 1 = .0100 + 1 = .0101$$

n-bit Binary Numbers

Unsigned: $b_{n-1}b_{n-2} \ldots b_1\, b_0$ $\quad (b_i = 0 \text{ or } b_i = 1)$

value: $b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_1 2^1 + b_0 2^0$

range: $[0\ ..\ 2^n - 1]$

n-bit sum: $\mathbf{A} \overset{\cdots}{+} \mathbf{B} + \mathbf{c} \cdot \mathbf{2}^n = \mathbf{A} + \mathbf{B}$

n-bit diff: $\mathbf{A} \overset{\cdots}{-} \mathbf{B} \equiv \mathbf{A} \overset{\cdots}{+} (\mathbf{2}^n - \mathbf{B}) = \mathbf{A} \overset{\cdots}{+} \overline{\mathbf{B}} + 1 = \mathbf{A} - \mathbf{B} + \mathbf{2}^n - \mathbf{c} \cdot \mathbf{2}^n$

Signed: $b_{n-1}b_{n-2} \ldots b_1\, b_0$

value: $-b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_1 2^1 + b_0 2^0$

range: $[-2^{n-1}\ ..\ 2^{n-1} - 1]$

n-bit sum: $\mathbf{A} \overset{\cdots}{+} \mathbf{B} = \mathbf{A} + \mathbf{B}$ $\quad$ **iff $\mathbf{v}$=0**

n-bit diff: $\mathbf{A} \overset{\cdots}{-} \mathbf{B} \equiv \mathbf{A} \overset{\cdots}{+} (\mathbf{2}^n - \mathbf{B}) = \mathbf{A} \overset{\cdots}{+} \overline{\mathbf{B}} + 1 = \mathbf{A} - \mathbf{B}$ $\quad$ **iff $\mathbf{v}$=0**

N–Bit Integers

(N = 8)

Unsigned Integers

Signed Integers

. . . .	. . . .
. . . 0100000000	$= 2^N$
. . . 0011111111	$= 2^N - 1$
. . . .	. . . .
. . . 0010000000	$= 2^{N-1}$
. . . 0001111111	$= 2^{N-1} - 1$
. . . .	. . . .
. . . 0000000101	
. . . 0000000100	$= 4 = 2^2$
. . . 0000000011	$= 3 = 2^2 - 1$
. . . 0000000010	$= 2 = 2^1$
. . . 0000000001	$= 1 = 2^0 = 2^1 - 1$
. . . 0000000000	$= 0 = 2^0 - 1$
. . . 1111111111	$= -1 = -2^0$
. . . 1111111110	$= -2 = -2^1$
. . . 1111111101	$= -3 = -2^1 - 1$
. . . 1111111100	$= -4 = -2^2$
. . . 1111111011	$= -5 = -2^2 - 1$
. . . .	. . . .
. . . 1110000000	$= -2^{N-1}$
. . . 1101111111	$= -2^{N-1} - 1$
. . . .	. . . .
. . . 1100000000	$= -2^N$
. . . 1011111111	$= -2^N - 1$
. . . .	. . . .

3-Bit Unsigned Integers (Binary)

00001111
00001110
00001101
00001100
00001011
00001010
00001001
00001000
00000111
00000110
00000101
00000100
00000011
00000010
00000001
00000000

8 supported values: 0 .. 7

Not closed under addition

carry => ***incorrect*** result: 8 .. 14

Not closed under subtraction

7 undefined (negative) differences

Want: $A \overset{\cdots}{-} B \overset{\cdots}{+} C = A \overset{\cdots}{+} C \overset{\cdots}{-} B$

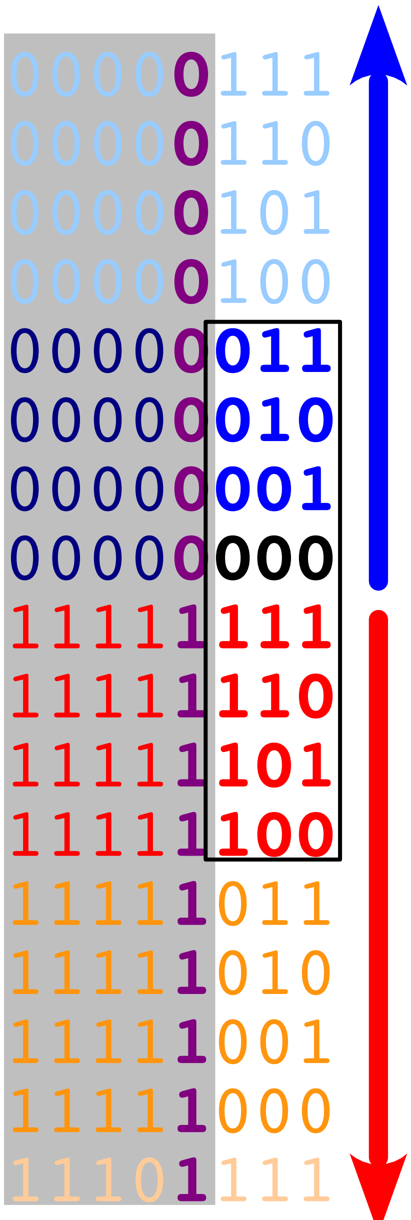
$A - B \equiv A \overset{\cdots}{+} 2^3 - B$

$\overset{\cdots}{B} \equiv 2^3 - B$ **($2^3 - B = \overline{B} + 1$)**

$A \geq B \Rightarrow A - B + 2^3 > 2^3$ (**carry**)

carry => ***correct*** result: 0 .. 7

3-Bit Signed Binary Integers



8 supported values: -4 .. 3

Sign: high bit: **0** positive, **1** negative

Not closed under addition

incorrect results: –8 .. –5, 4 .. 6

Not closed under subtraction

incorrect results: –7 .. –5, 4 .. 7

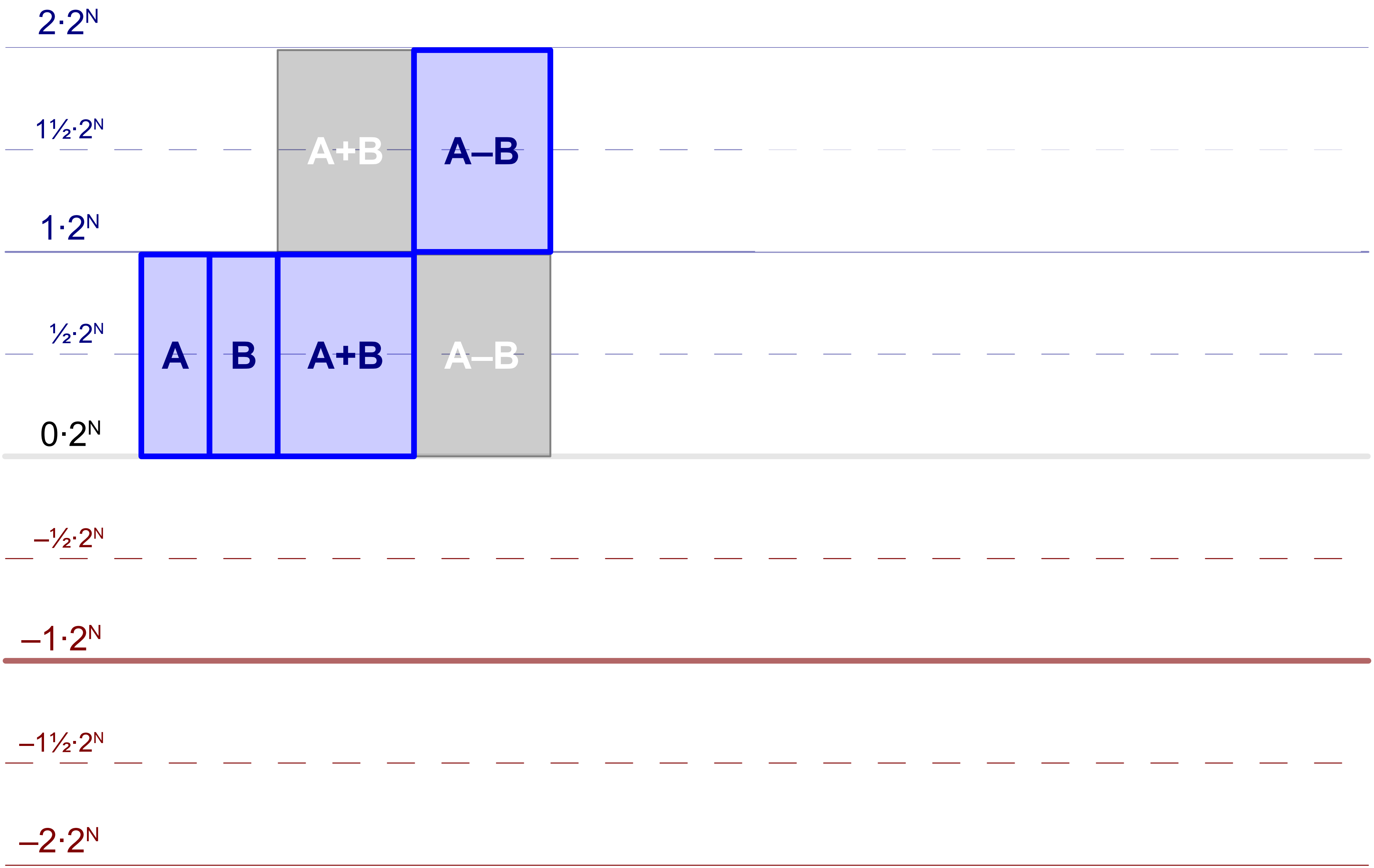
Correct: $X \overset{\cdots}{+} Y = Z$, $X \overset{\cdots}{+} Y = Z$

$X \overset{\cdots}{+} Y$, $X \overset{\cdots}{+} Y$ always correct. Why?

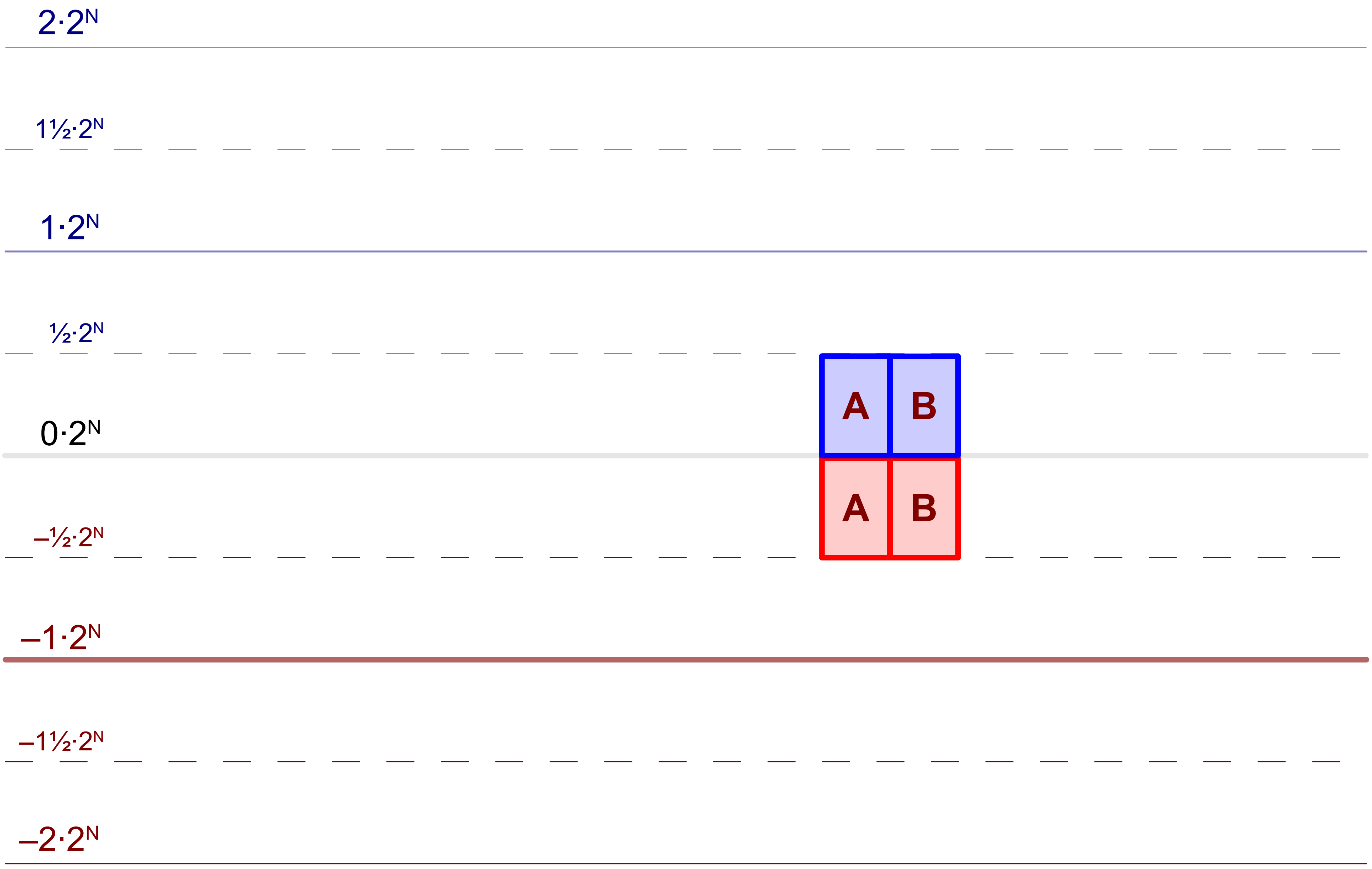
Incorrect: $X \overset{\cdots}{+} Y = Z$, $X \overset{\cdots}{+} Y = Z$

$X \overset{\cdots}{-} Y = Z$, $X \overset{\cdots}{-} Y = Z$

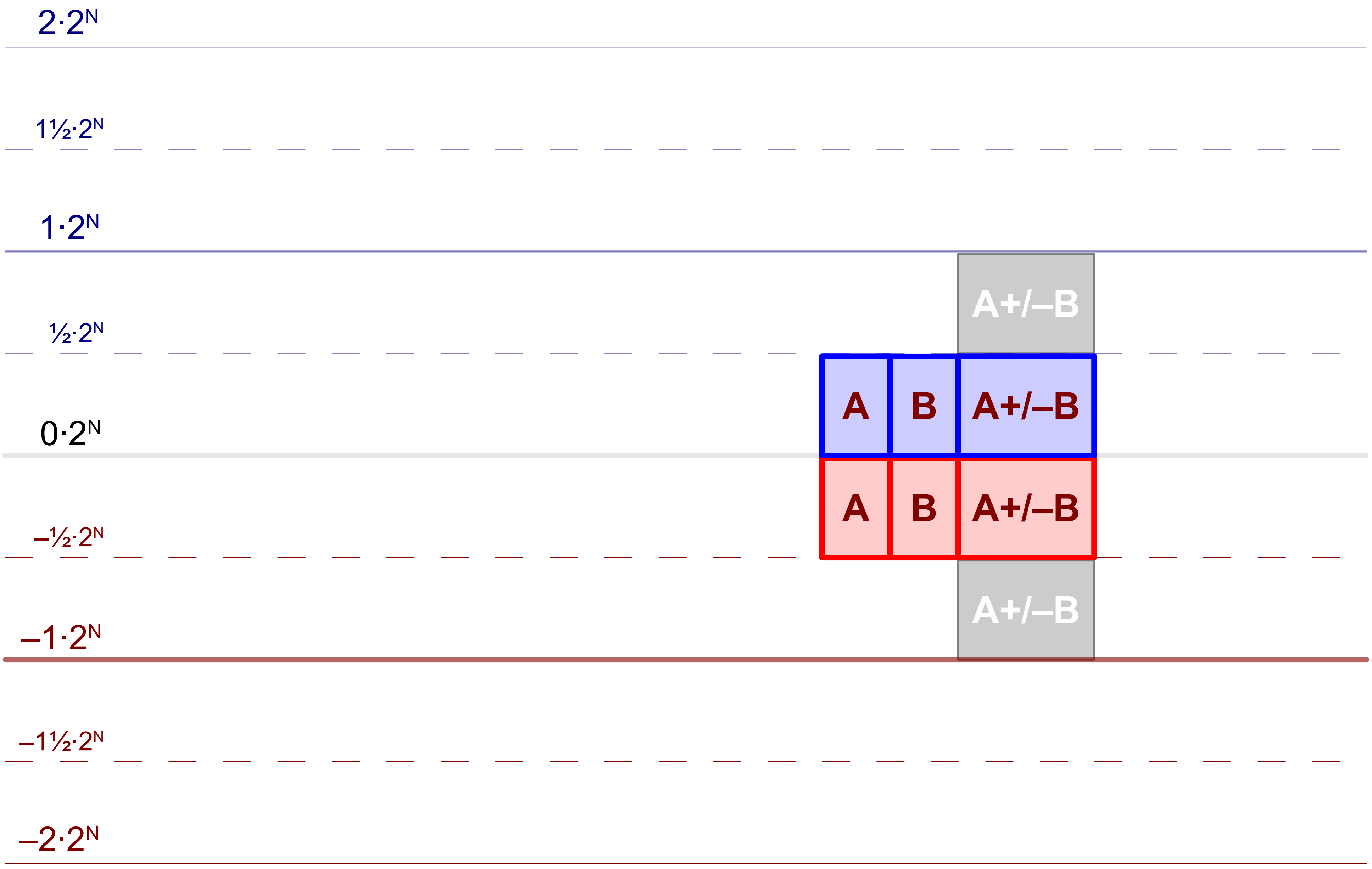
Unsigned Sum and Difference



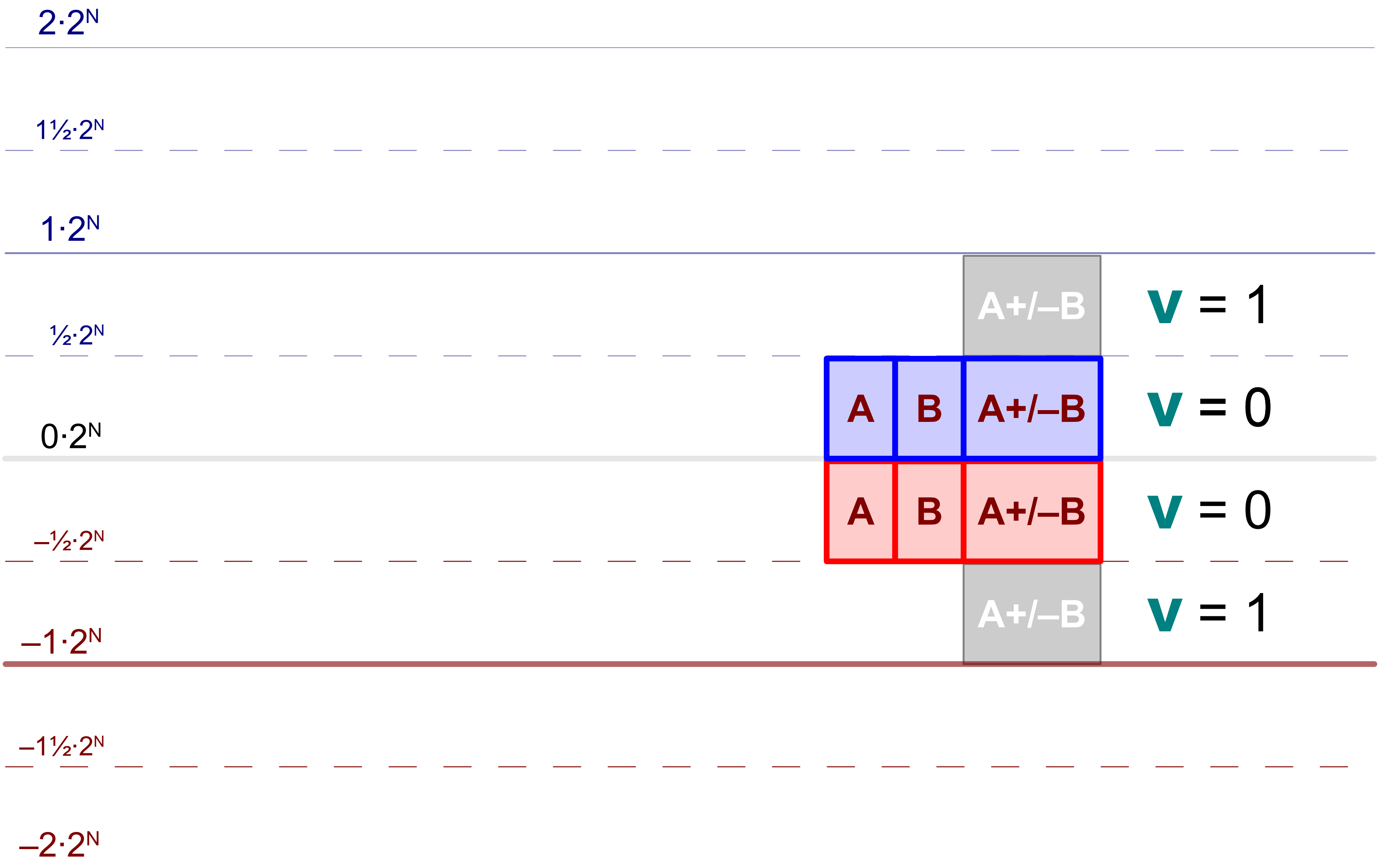
Signed Integers



Signed Integer Sum or Difference



Signed Integer Sum or Difference



Unsigned Two's Complement

.010000	16
.001111	15
.001110	14
.001101	13
.001100	12
.001011	11
.001010	10
.001001	09
.001000	08
.000111	07
.000110	06
.000101	05
.000100	04
.000011	03
.000010	02
.000001	01
.000000	00

Additive Inverse Theroem:

If **X** and **Y** are whole numbers

$$\mathbf{X - Y = X + \overline{Y} + 1}$$

$$\mathbf{-Y \equiv \overline{Y} + 1}$$

Unsigned <u>N-bit integers</u>: $\underline{A}$ and $\underline{B}$

$$\underline{B} + \underline{\overline{B}} + 1 = 2^N - 1 \;\; + 1 = 2^N = \underline{0}$$

$$\underline{A - B} \equiv \underline{A + \overline{B} + 1}$$

$$= 2^N + \mathbf{A} - \mathbf{B} \quad \text{if } 0 \le \mathbf{A} - \mathbf{B} < 2^N \; \mathbf{(A \ge B)}$$

Carry => **correct** subtraction

Signed Two's Complement

.001000	+8
.000111	+7
.000110	+6
.000101	+5
.000100	+4
.000011	+3
.000010	+2
.000001	+1
.000000	+0
.111111	−1
.111110	−2
.111101	−3
.111100	−4
.111011	−5
.111010	−6
.111001	−7
.111000	−8

Additive Inverse Theroem:

If **X** and **Y** are whole numbers

$$\mathbf{X - Y = X + \overline{Y}+1}$$

$$\mathbf{-Y \equiv \overline{Y}+1}$$

Signed <u>N-bit integers</u>: <u>C</u> and <u>D</u>

$$\mathbf{D} + \overline{\mathbf{D}} + 1 = -1 + 1 = 0$$

$$\underline{C - D} = \underline{C + \overline{D} + 1}$$

$$= \mathbf{C - D} \quad \text{if } -2^{N-1} \leq \mathbf{C - D} < 2^{N-1}$$

oVerflow => **incorrect** subtraction

Honesty Criteria

The n-bit result **r** of a binary operation on n-bit values **a** and **b** is ***honest*** (***deceptive***) if it is *the same as* (*different from*) the whole number result of the same operation on the same values.

(n-bit) unsigned addition is *honest* iff (**c** = 0)

Carry flag is not set

(n-bit) unsigned subtraction is *honest* iff (**c** = 1)

Carry flag is set

(n-bit) signed addition is *honest* iff (**v** = 0)

a and **b** have different signs or **a**, **b**, and **r** have same sign

(n-bit) signed subtraction is *honest* iff (**v** = 0)

a and **b** have same sign or **a** and **r** have same sign

HW 9: Signed Binary Arithmetic

For each of the <**X**, **Y**> pairs in the table below:

a) Convert **X** and **Y** → binary

b) Compute $X \dot{+} Y$ (the 8-bit **sum**)

c) Compute $\ddot{Y}$ (the 2's complement of **Y**)

d) Compute $X \dot{-} Y \equiv X \dot{+} \ddot{Y}$ (the 8-bit **difference**)

e) Convert $X \dot{+} Y$, $\ddot{Y}$, and $X \dot{-} Y$ → hexadecimal

f) Indicate condition flag (**z**, **n**, **c**, **v**) values for $X \dot{+} Y$, $X \dot{-} Y$

g) Indicate the signs of **X**, **Y**, $X \dot{+} Y$, $\ddot{Y}$, and $X \dot{-} Y$

h) Is $X \dot{+} Y$ honest? is $X \dot{-} Y$ honest?

Where <**X**, **Y**> =

1) <0x**4F**, 0x**6D**>
2) <0x**B3**, 0x**17**>
3) <0x**A3**, 0x**95**>
4) <0x**6E**, 0x**3A**>

Signed Arithmetic Example

X	Y	X + Y	~Y	X – Y
`0x8C`	`0x6F`			

Signed Arithmetic Example

X	Y	X + Y	~Y	X – Y
0x8C	**0x6F**			
10001100	**01101111**			

Signed Arithmetic Example

X	Y	X + Y	~Y	X – Y
0x8C	**0x6F**			
10001100	01101111	10001100 01101111 011111011		

Signed Arithmetic Example

X	Y	X + Y	~Y	X – Y
0x8C	**0x6F**			
10001100	01101111	10001100	$\overline{01101111}$	
		01101111	10010000	
		011111011	00000001	
			10010001	

Signed Arithmetic Example

X	Y	X + Y	~Y	X – Y
0x8C	**0x6F**			
10001100	01101111	10001100	$\overline{01101111}$	10001100
		<u>01101111</u>	10010000	<u>10010001</u>
		011111011	<u>00000001</u>	100011101
			10010001	

Signed Arithmetic Example

X	Y	X + Y	~Y	X – Y
0x8C	0x6F			
10001100	01101111	10001100	01101111	10001100
		01101111	10010000	10010001
		011111011	00000001	100011101
			10010001	

Signed Arithmetic Example

X	Y	X + Y	~Y	X – Y
0x8C	0x6F			
10001100	01101111	10001100 01101111 011111011	$\overline{01101111}$ 10010000 00000001 10010001	10001100 10010001 100011101
		0xFB	0x91	0x1D

Signed Arithmetic Example

X	Y	X + Y	~Y	X – Y
0x8C	0x6F			
10001100	01101111	10001100 01101111 011111011	01101111 10010000 00000001 10010001	10001100 10010001 100011101
		0xFB	0x91	0x1D
		no oVerflow		oVerflow

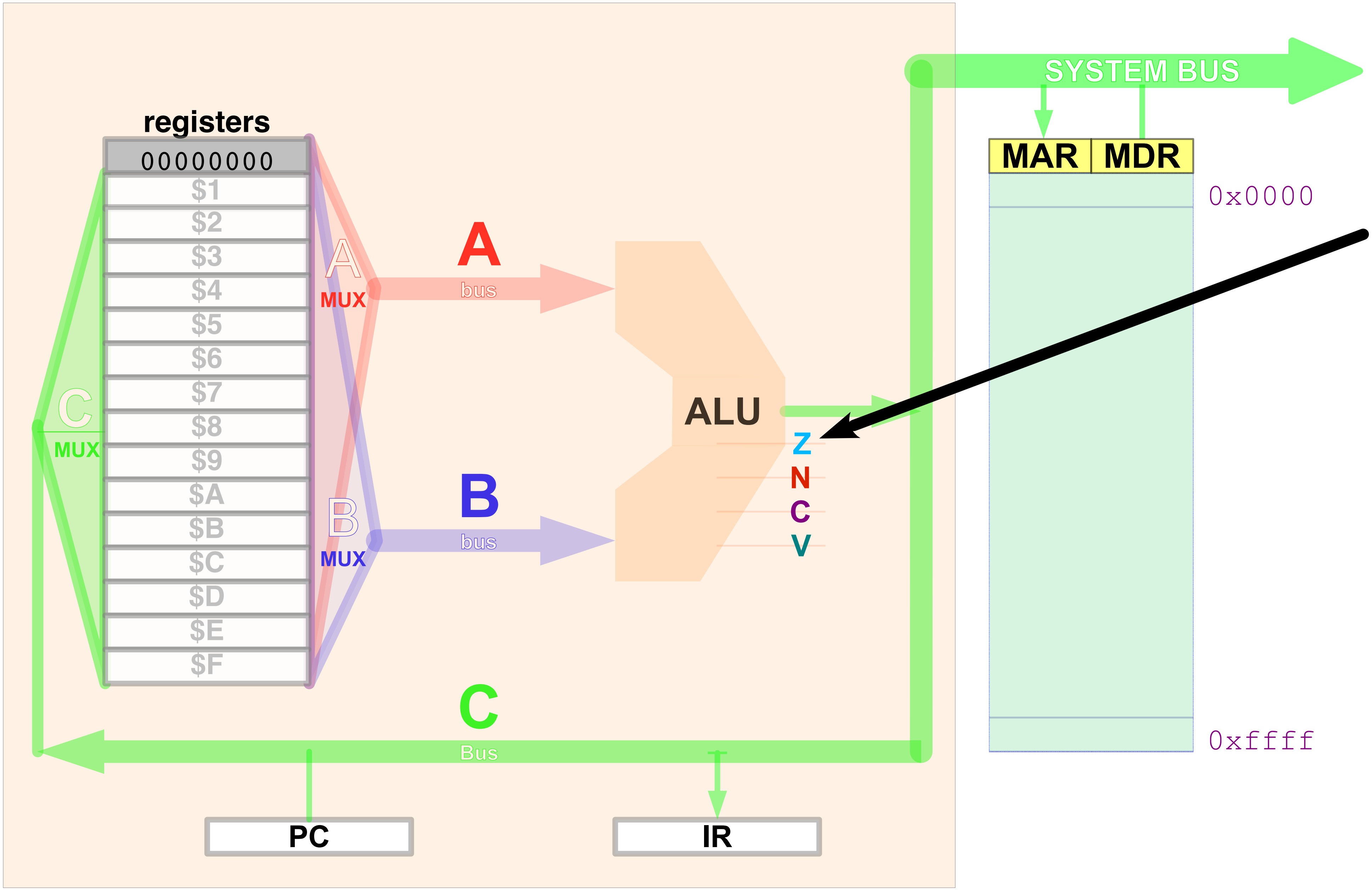
Signed Arithmetic Example

X	Y	X + Y	~Y	X – Y
0x8C	**0x6F**			
10001100	01101111	10001100 01101111 011111011	$\overline{01101111}$ 10010000 00000001 10010001	10001100 10010001 100011101
		0xFB	**0x91**	**0x1D**
		$\bar{z}n\bar{c}\bar{v}$		$\bar{z}\bar{n}cv$

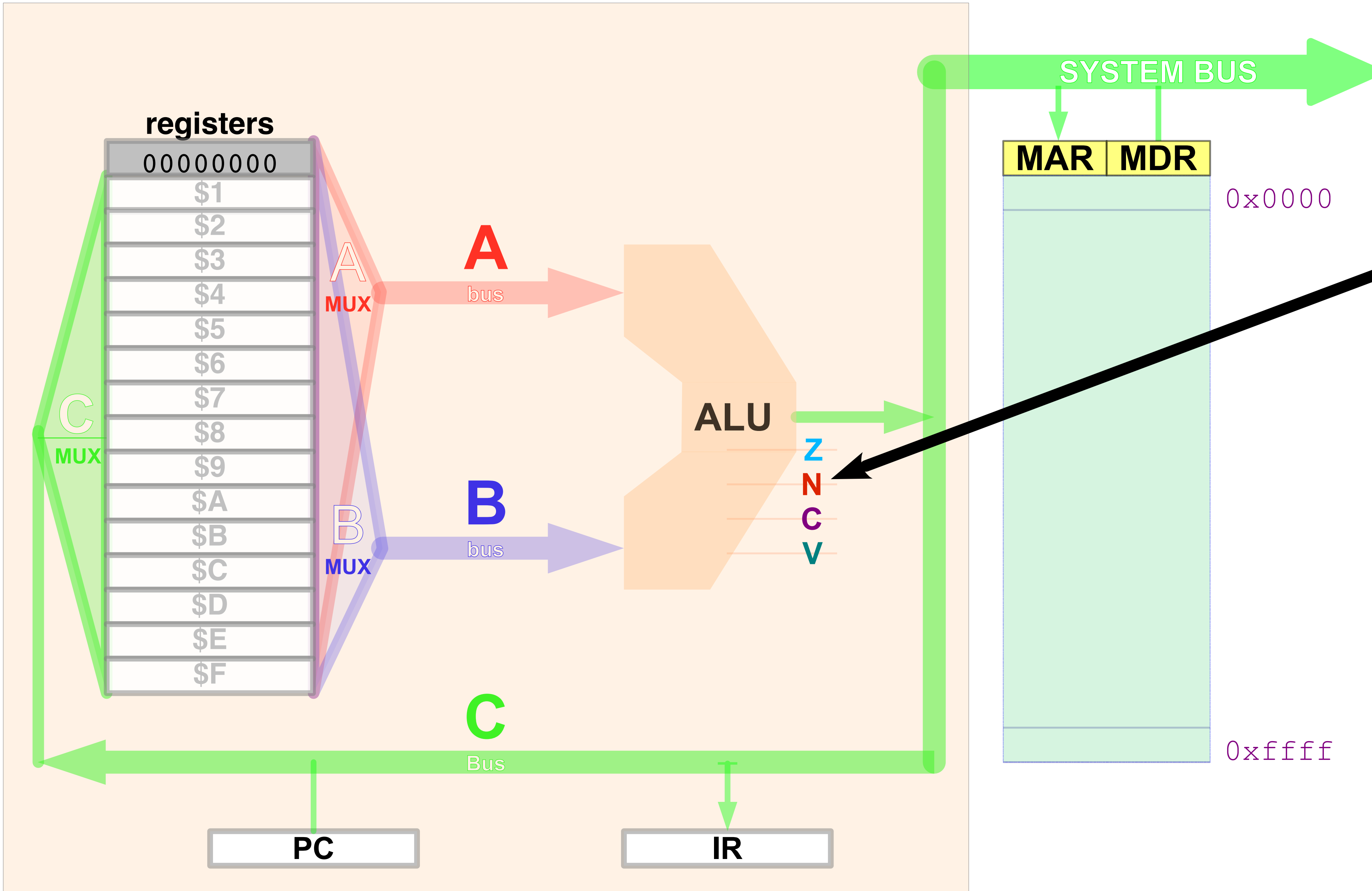
Signed Arithmetic Example

X	Y	X + Y	~Y	X – Y
0x8C	0x6F			
10001100	01101111	10001100	$\overline{01101111}$	10001100
		01101111	10010000	10010001
		011111011	00000001	100011101
			10010001	
		0xFB	0x91	0x1D
		$\bar{z}n\bar{c}\bar{v}$		$\overline{zn}cv$
		honest		deceptive

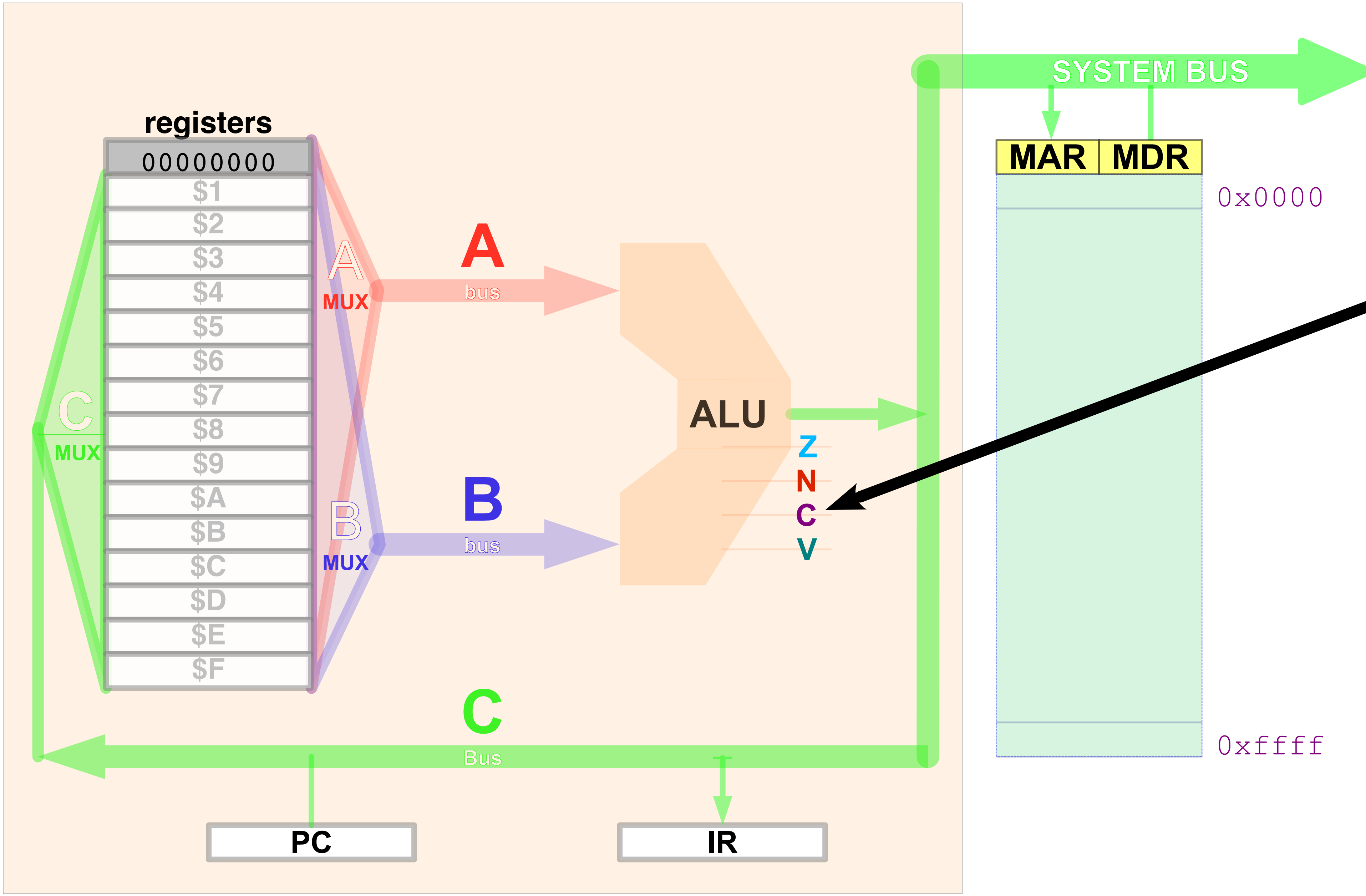
z — **Zero** condition flag



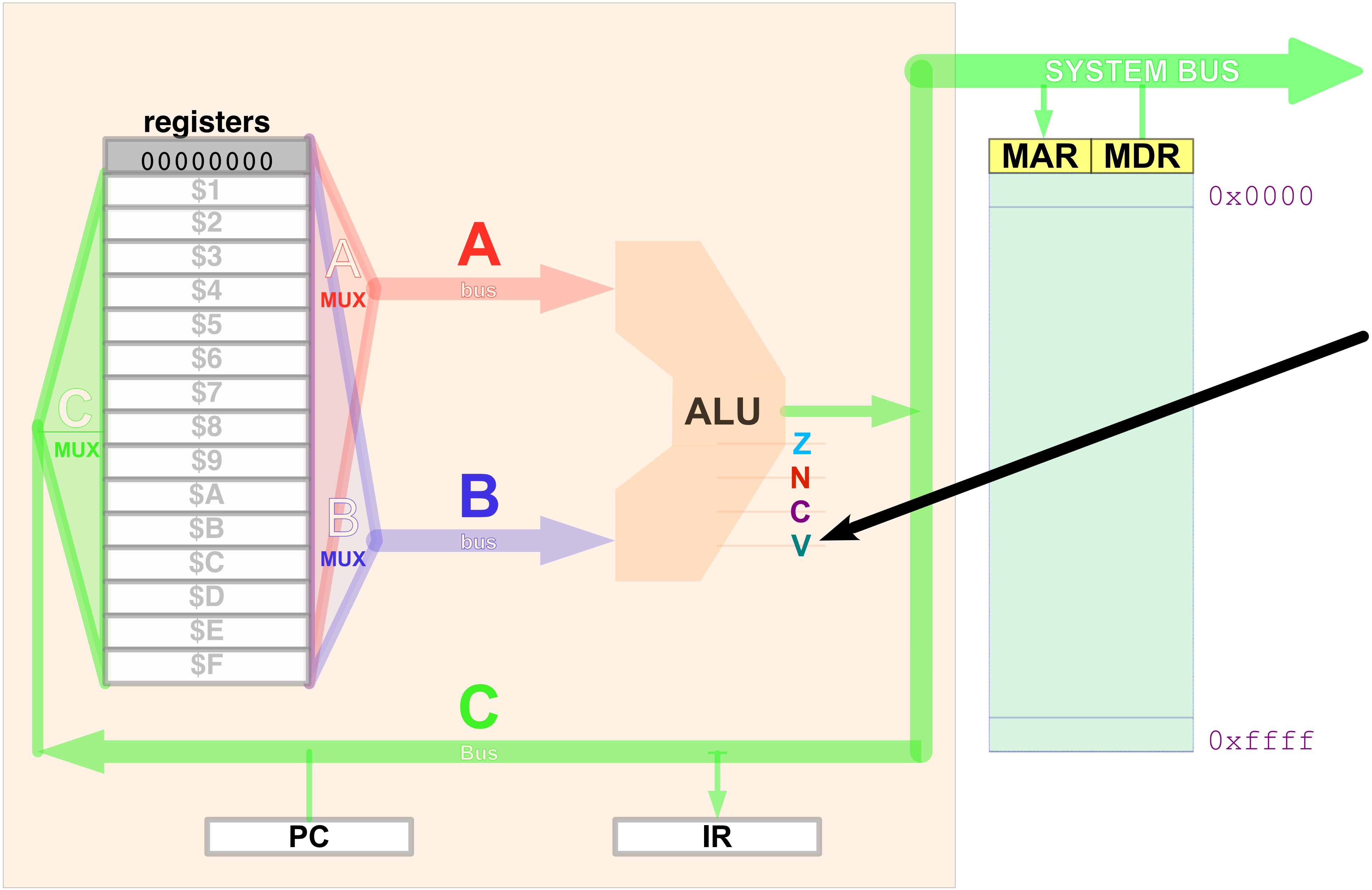
n — **Negative** condition flag



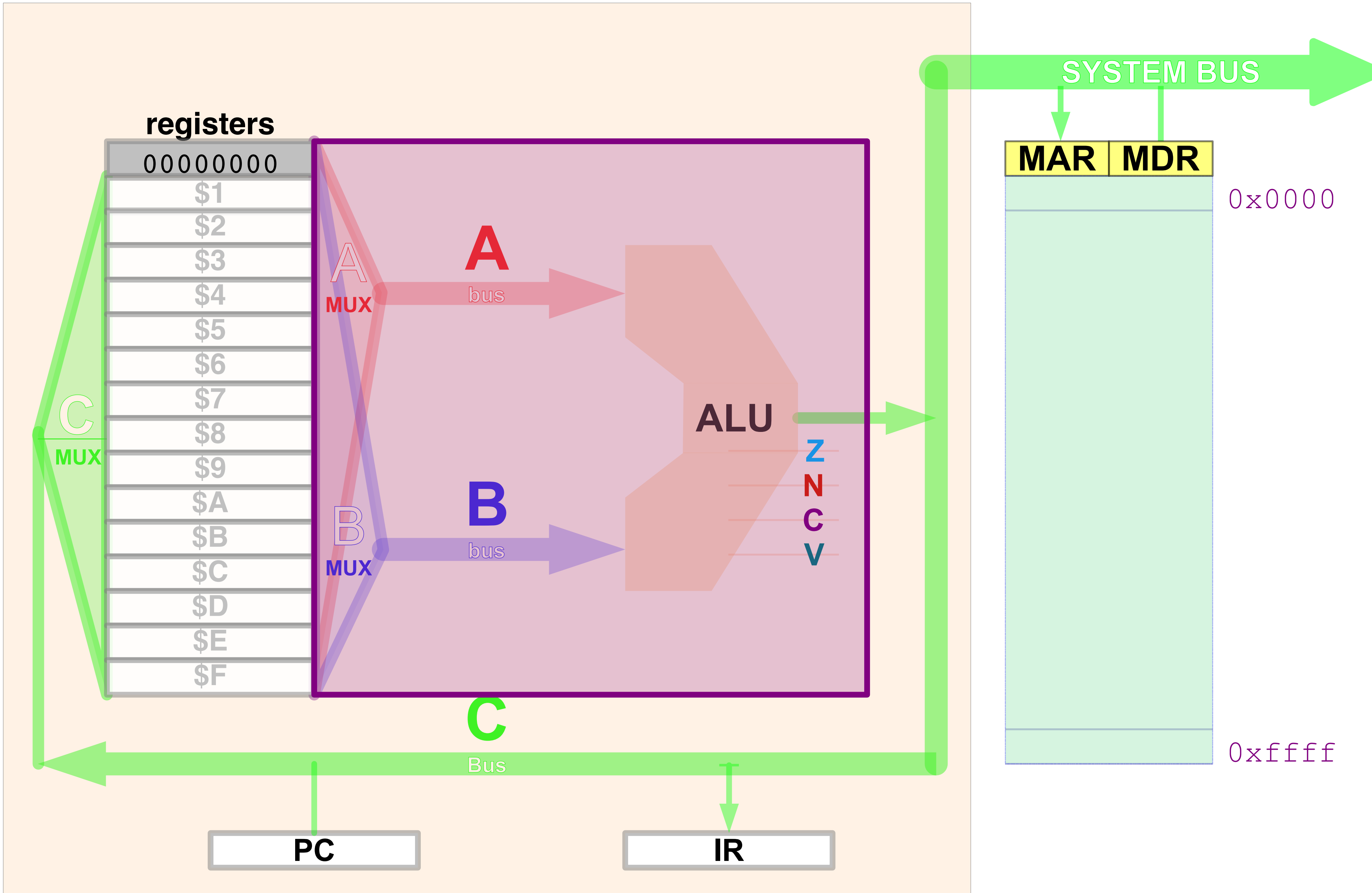
c — **Carry** condition flag



v — oVerflow condition flag



Core Combinational Circuit



HW 7: **W = X + Y + Z**

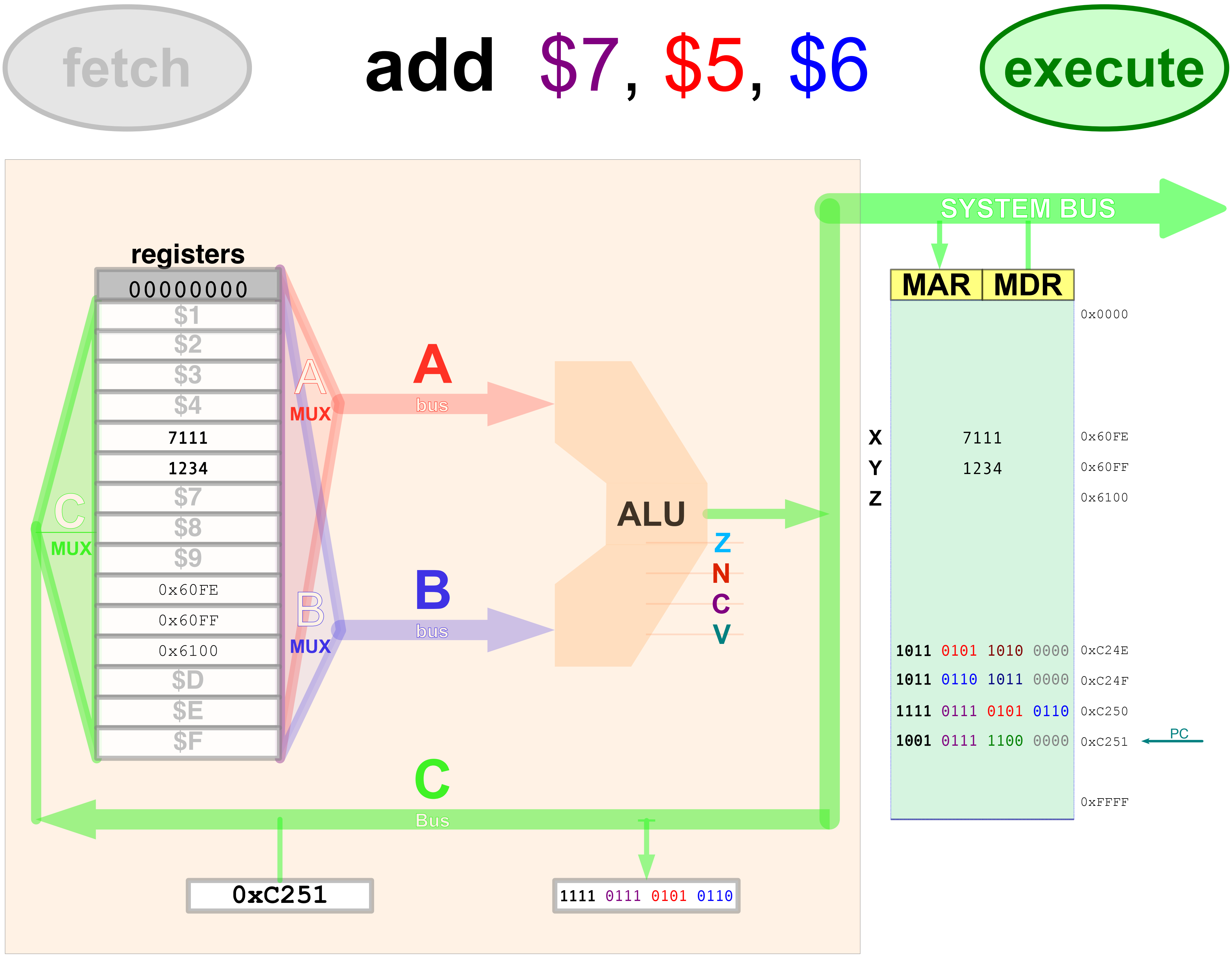
<u>Assignment</u>: Write a **TOY** assembly language program to add the values of 3 variables, **`X`**, **`Y`**, and **`Z`**, in memory and store the sum in a fourth, **`W`**.

<u>Details</u>: The variables occupy consecutive words of memory starting with **`W`**. The address of **`W`** is in register **`$3`**. *Do not change* the values in registers **`$0`** through **`$3`**. You can use registers **`$4`** through **`$F`** as you please.

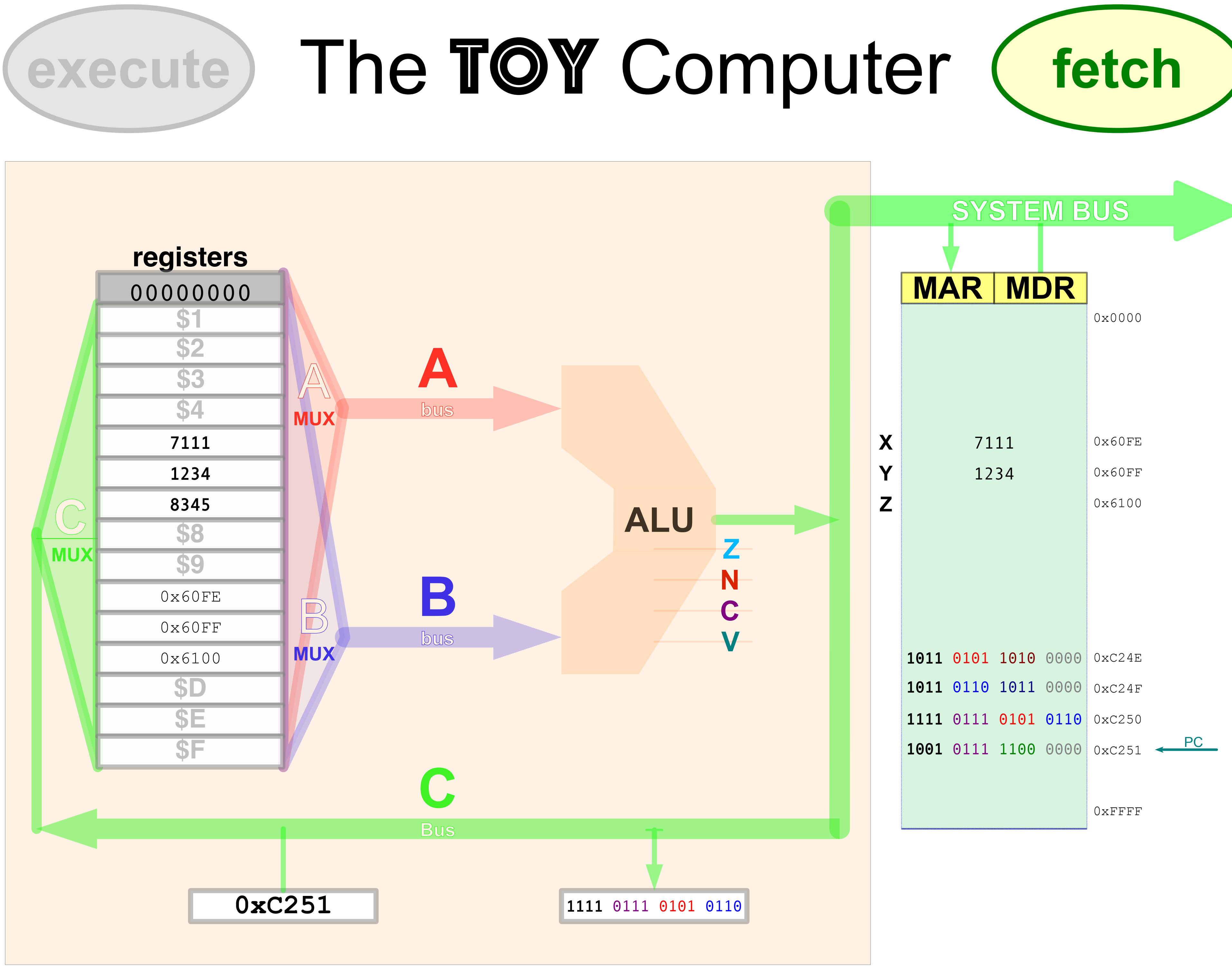
Your program should consist entirely of addition (**`add`**), load (**`l`**), and store (**`st`**) instructions.

add	$7, $5,$6	means	$7	←	[$5] + [$6]
l	$4, $C, 8	means	$4	←	[MEM[[$C]+8]]
st	$4, $C, 8	means	[$4]	→	MEM[[$C]+8]

add $7, $5, $6



fetch
execute
registers
00000000
$1
$2
$3
$4
7111
1234
$7
$8
$9
0x60FE
0x60FF
0x6100
$D
$E
$F
A
MUX
B
MUX
C
MUX
A
bus
B
bus
ALU
Z
N
C
V
C
Bus
0xC251
1111 0111 0101 0110
SYSTEM BUS
MAR
MDR
0x0000
X
7111
0x60FE
Y
1234
0x60FF
Z
0x6100
1011 0101 1010 0000
0xC24E
1011 0110 1011 0000
0xC24F
1111 0111 0101 0110
0xC250
1001 0111 1100 0000
0xC251
PC
0xFFFF



execute
The TOY Computer
fetch
SYSTEM BUS
registers
00000000
$1
$2
$3
$4
7111
1234
8345
$8
$9
0x60FE
0x60FF
0x6100
$D
$E
$F
A
MUX
B
MUX
C
MUX
A
bus
B
bus
ALU
Z
N
C
V
C
Bus
0xC251
1111 0111 0101 0110
MAR
MDR
0x0000
X
7111
0x60FE
Y
1234
0x60FF
Z
0x6100
1011 0101 1010 0000
0xC24E
1011 0110 1011 0000
0xC24F
1111 0111 0101 0110
0xC250
1001 0111 1100 0000
0xC251
PC
0xFFFF

TOY Instruction Set Architecture

Reference Card

Immediate values

s8	— 8-bit signed immediate
u4	— 4-bit unsigned immediate
cc	— 4-bit condition code

Register File

16 16-bit "registers"

15 real registers: `$1 … $F`

1 pseudo-register: `$0` `[$0] = 0`

Main Memory

65536 16-bit words

`M[n]` – n^{th} memory address

`^M[n]` – content of `M[n]`

Instructions[0]

add	`$T ← [$A]+[$B]`[1]	
and	`$T ← [$A]&[$B]`[1]	
bc	`PC ← [PC] + s8`	*iff* cc
bcl	`$L ← [PC], PC ← [$A]`[1]	*iff* cc
l	`$T ← ^M[[$A]+u4]`[1]	
lwr	`$T ← ^M[[$A]+u4]`[1]	*set* rsvn
lih	$\$T_{15..8} \leftarrow$ `s8`[1]	
lis	`$T ← s8`[1] `(sign extended)`	
nor	$\$T \leftarrow \overline{[\$A] \mid [\$B]}$ [1]	
sl	`$T ← [$A] << u4`[1]	
srs	`$T ← [$A] >> u4`[1]	
sru	`$T ← [$A] >>>u4`[1]	
st	`M[[$A]+u4] ← [$S]`	
stc	`M[[$A]+u4] ← [$S]`[4]	*iff* rsvn
sub	`$T ← [$A]-[$B]`[1,2]	
sys	`system call`	

Arithmetic / Logical

1111	**add**	$T	$A	$B
1110	**sub**	$T	$A	$B
1101	**and**	$T	$A	$B
1100	**nor**	$T	$A	$B

Load /Store

1011	**l**	$T	$A	u4
1010	**lwr**	$T	$A	u4
1001	**st**	$S	$A	u4
1000	**stc**	$S	$A	u4

Shift / Branch & Link

0111	**sru**	$T	$A	u4
0110	**srs**	$T	$A	u4
0101	**bcl**	cc	$A	$L
0100	**sl**	$T	$A	u4

Immediate

0011	**lih**	$T	s8
0010	**lis**	$T	s8
0001	**bc**	cc	s8
0000	**sys**	$X	s8

Condition Codes

1111	$\overline{znv} + nv$	SGT
1110	$n\overline{v} + \overline{n}v$	SLT
1101	n	NEG
1100	v	OVF
1011	$\overline{z}c$	UGT
1010	$\overline{c}$	ULT
1001	$\overline{z}$	NE
1000	0	NOP
0111	$z + n\overline{v} + \overline{n}v$	SLE
0110	$nv + \overline{nv}$	SGE
0101	$\overline{n}$	POS
0100	$\overline{v}$	NVF
0011	$z + \overline{c}$	ULE
0010	c	UGE
0001	z	EQ
0000	1	ALL

Notes

[0] `PC ← PC+1` ***before*** instruction execution

[1] $0 ***not*** changed ([$0] = 0 always)

[2] Determines flags: **z, n, c, v**

[4] Determines flag: **v**

TOY ALU Instructions

Addition

`add` `$9, $6, $5` **1111** 1001 0110 0101

$\$9 = [\$6] + [\$5]$

Subtraction

`sub` `$9, $6, $5` **1110** 1001 0110 0101

$\$9 = [\$6] - [\$5] = [\$6] + \overline{[\$5]} + 1$

And

`and` `$9, $6, $5` **1101** 1001 0110 0101

$\$9 = [\$6] \ \& \ [\$5]$

Not–Or [De Morgan's law: ~(A | B) = (~A) & (~B)]

`nor` `$9, $6, $5` **1100** 1001 0110 0101

$\$9 = \overline{[\$6]} \ \& \ \overline{[\$5]}$

(1-bit) Full Adder Circuit Design

Combinational circuit

- Output determined by input

Design process

1. Specify semantics
   - Black Box: *input* and *output* (informal semantics)
   - Truth Table (formal semantics)
2. Truth table → Boolean formula
3. Minimize boolean formula (optional)
   - Boolean algebra
   - Karnaugh maps
4. Boolean formula → combinational circuit

(1-bit) Full Adder Circuit Design

Combinational circuit

- Output determined by input

Design process

1. Specify semantics
   - Black Box: *input* and *output* (informal semantics)
   - Truth Table (formal semantics)
2. Truth table → Boolean formula
3. Minimize boolean formula (optional)
   - Boolean algebra
   - Karnaugh maps
4. Boolean formula → combinational circuit

(1-bit) Full Adder Circuit Design

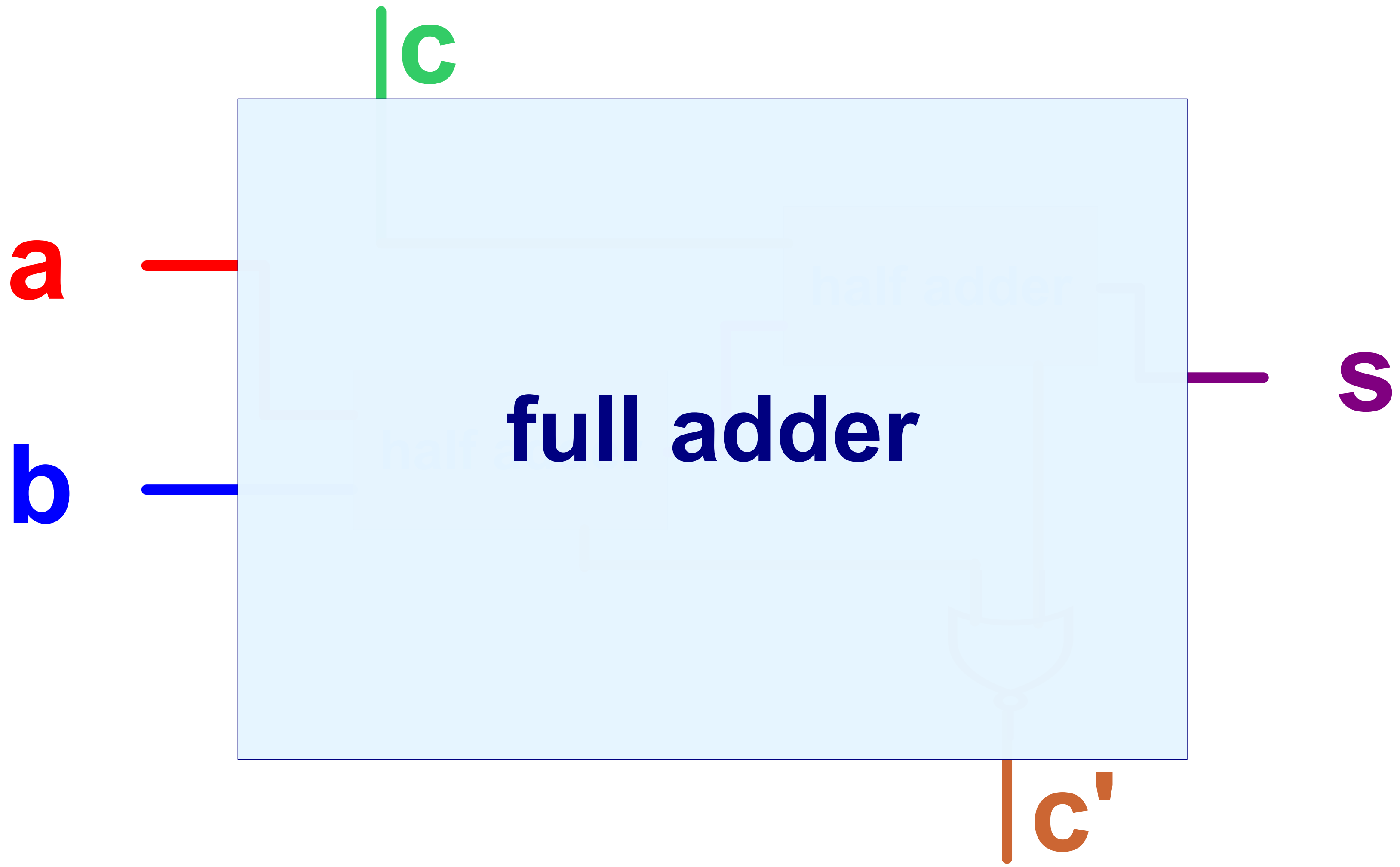
Combinational circuit

Output determined by input

Design process

1. Specify semantics
   - Black Box: *input* and *output* (informal semantics)
   - Truth Table (formal semantics)
2. Truth table → Boolean formula
3. Minimize boolean formula (optional)
   - Boolean algebra
   - Karnaugh maps
4. Boolean formula → combinational circuit

Full Adder Black Box



Sum 3 1-bit inputs to give a 2-bit output

(1-bit) Full Adder Circuit Design

Combinational circuit

Output determined by input

Design process

1. Specify semantics

   Black Box: *input* and *output* (informal semantics)

   Truth Table (formal semantics)

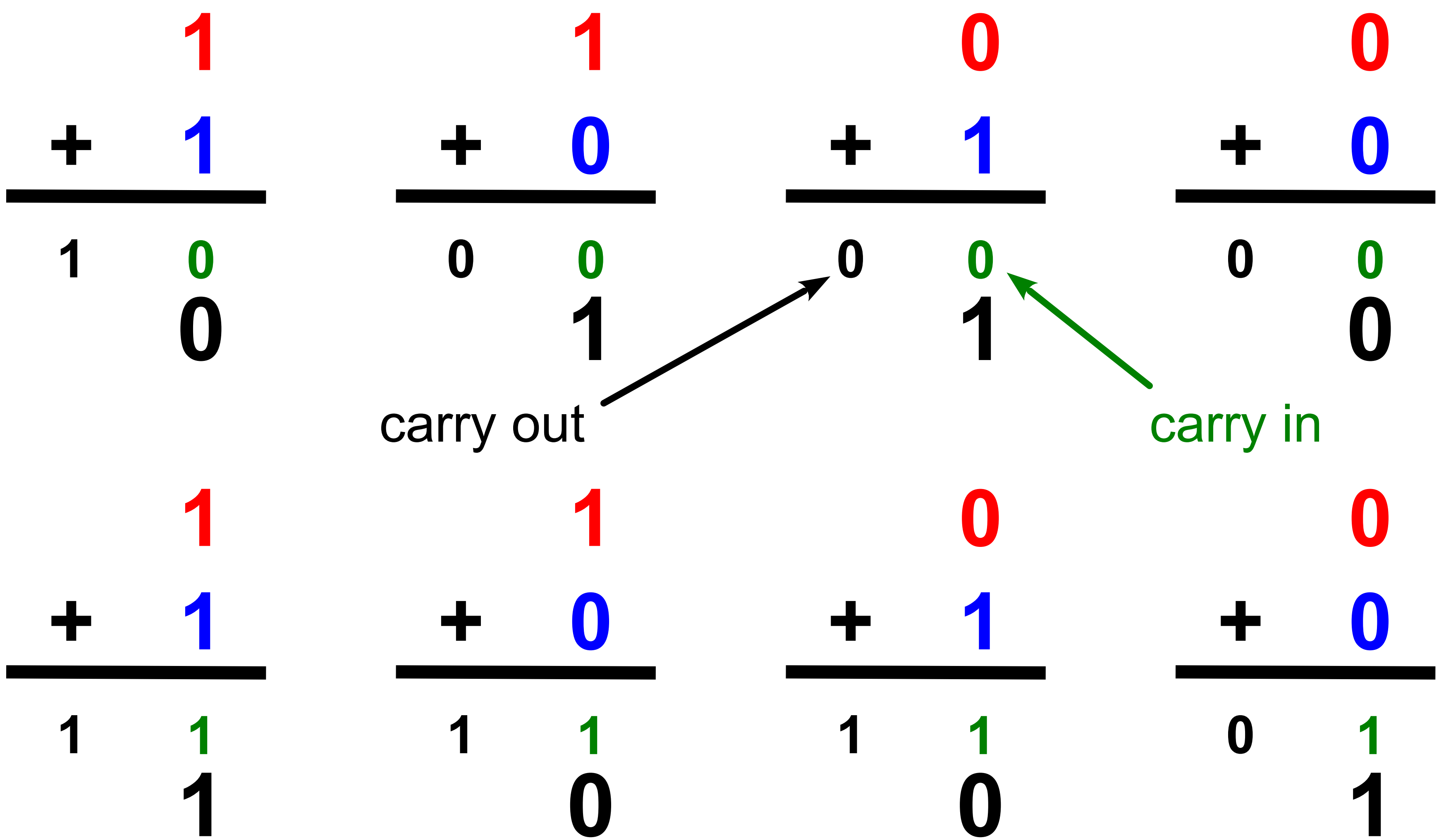
2. Truth table → Boolean formula
3. Minimize boolean formula (optional)

   Boolean algebra

   Karnaugh maps

4. Boolean formula → combinational circuit

Full Adder Truth Table



Full Adder Truth Table

#	a b c	c'	s
0	0 0 0	0	0
1	0 0 1	0	1
2	0 1 0	0	1
3	0 1 1	1	0
4	1 0 0	0	1
5	1 0 1	1	0
6	1 1 0	1	0
7	1 1 1	1	1

(1-bit) Full Adder Circuit Design

Combinational circuit

Output determined by input

Design process

1. Specify semantics

    Black Box: *input* and *output* (informal semantics)

    Truth Table (formal semantics)

2. Truth table → Boolean formula

3. Minimize boolean formula (optional)

    Boolean algebra

    Karnaugh maps

4. Boolean formula → combinational circuit

Full Adder Truth Table

#	a b c	c'	s
0	0 0 0	0	0
1	0 0 1	0	1
2	0 1 0	0	1
3	0 1 1	1	0
4	1 0 0	0	1
5	1 0 1	1	0
6	1 1 0	1	0
7	1 1 1	1	1

Full Adder Boolean Formulas

#	a b c	c'	c'	s	s
0	0 0 0	0		0	
1	0 0 1	0		1	$\overline{a}\,\overline{b}c$
2	0 1 0	0		1	$\overline{a}b\overline{c}$
3	0 1 1	1	$\overline{a}bc$	0	
4	1 0 0	0		1	$a\overline{b}\,\overline{c}$
5	1 0 1	1	$a\overline{b}c$	0	
6	1 1 0	1	$ab\overline{c}$	0	
7	1 1 1	1	abc	1	abc

Full Adder Boolean Formulas

#	a b c	c'	c'	s	s
0	0 0 0	0		0	
1	0 0 1	0		1	$\overline{a}\,\overline{b}c$
2	0 1 0	0		1	$\overline{a}b\overline{c}$
3	0 1 1	1	$\overline{a}bc$	0	
4	1 0 0	0		1	$a\overline{b}\,\overline{c}$
5	1 0 1	1	$a\overline{b}c$	0	
6	1 1 0	1	$ab\overline{c}$	0	
7	1 1 1	1	abc	1	abc

$$s = \overline{a}\,\overline{b}c + \overline{a}b\overline{c} + a\overline{b}\,\overline{c} + abc$$

$$c' = \overline{a}bc + a\overline{b}c + ab\overline{c} + abc$$

(1-bit) Full Adder Circuit Design

Combinational circuit

- Output determined by input

Design process

1. Specify semantics
   - Black Box: *input* and *output* (informal semantics)
   - Truth Table (formal semantics)
2. Truth table → Boolean formula
3. Minimize boolean formula (optional)
   - Boolean algebra
   - Karnaugh maps
4. Boolean formula → combinational circuit

Full Adder Boolean Formulas

#	a	b	c	c'	s
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

$$s = \overline{a}\,\overline{b}c + \overline{a}b\overline{c} + a\overline{b}\,\overline{c} + abc$$

$$c' = \overline{a}bc + a\overline{b}c + ab\overline{c} + abc$$

Full Adder Boolean Formulas

#	a b c	c'	s
0	0 0 0	0	0
1	0 0 1	0	1
2	0 1 0	0	1
3	0 1 1	1	0
4	1 0 0	0	1
5	1 0 1	1	0
6	1 1 0	1	0
7	1 1 1	1	1

$$s = \overline{a}\,\overline{b}c + \overline{a}b\overline{c} + a\overline{b}\,\overline{c} + abc$$

$$c' = \overline{a}bc + a\overline{b}c + ab\overline{c} + abc$$

$$= \overline{a}bc + a\overline{b}c + ab\overline{c} + abc + abc + abc$$

Full Adder Boolean Formulas

#	a	b	c	c'	s
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

$$s = \overline{a}\,\overline{b}c + \overline{a}b\overline{c} + a\overline{b}\,\overline{c} + abc$$

$$c' = \overline{a}bc + a\overline{b}c + ab\overline{c} + abc$$

$$= \boxed{\begin{matrix}\overline{a}bc\\ abc\end{matrix}} + \boxed{\begin{matrix}a\overline{b}c\\ abc\end{matrix}} + \boxed{\begin{matrix}ab\overline{c}\\ abc\end{matrix}}$$

Full Adder Boolean Formulas

#	a b c	c'	s
0	0 0 0	0	0
1	0 0 1	0	1
2	0 1 0	0	1
3	0 1 1	1	0
4	1 0 0	0	1
5	1 0 1	1	0
6	1 1 0	1	0
7	1 1 1	1	1

$$s = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$$

$$c' = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc$$

$$= \bar{a}bc + abc + a\bar{b}c + abc + ab\bar{c} + abc$$

$$= bc + ac + ab$$

Full Adder Boolean Formulas

#	a	b	c	c'	s
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

$$s = \overline{a}\,\overline{bc} + \overline{a}b\overline{c} + a\overline{bc} + abc$$

$$c' = bc + ac + ab$$

(1-bit) Full Adder Circuit Design

Combinational circuit

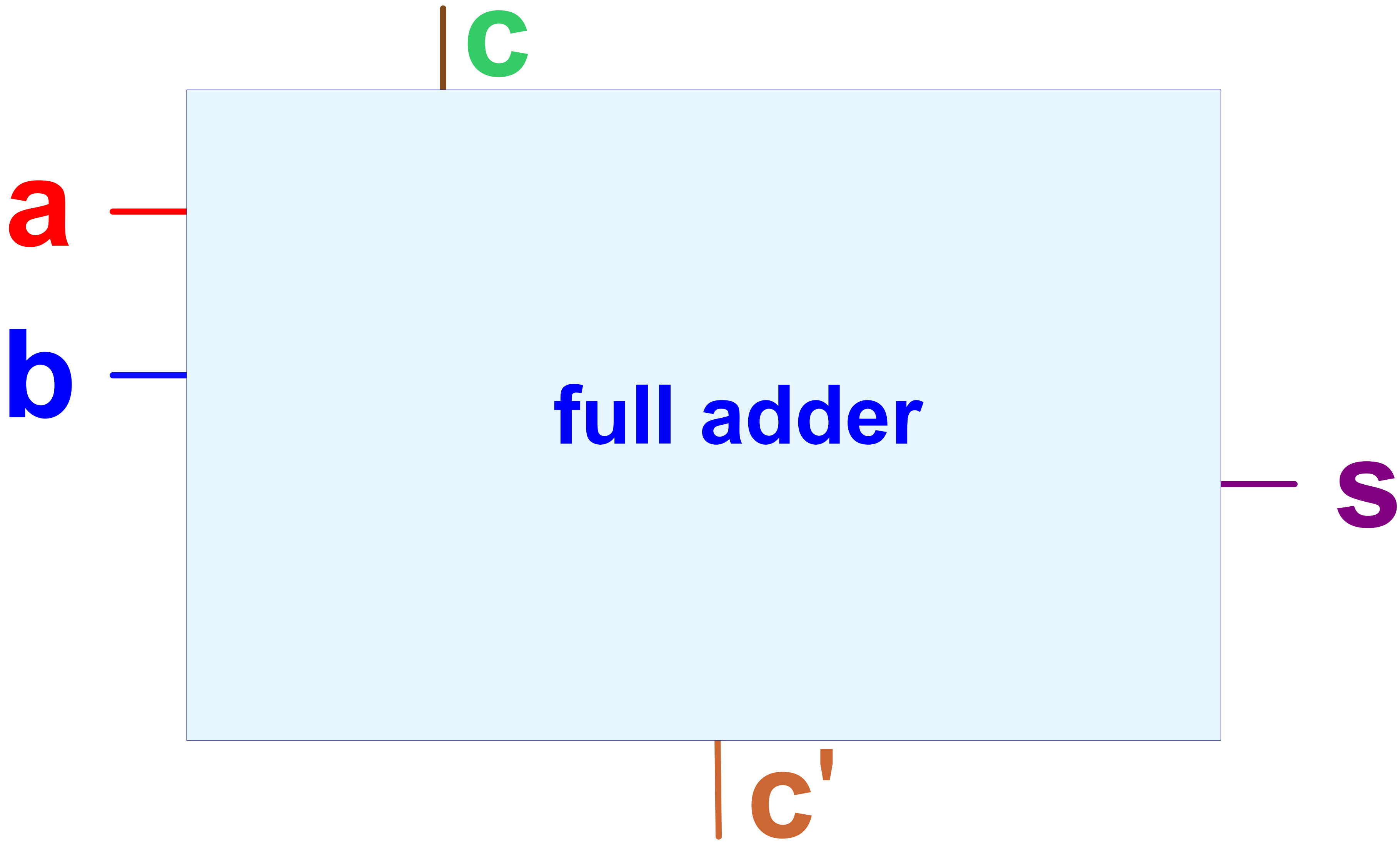
- Output determined by input

Design process

1. Specify semantics
   - Black Box: *input* and *output* (informal semantics)
   - Truth Table (formal semantics)
2. Truth table → Boolean formula
3. Minimize boolean formula (optional)
   - Boolean algebra
   - Karnaugh maps
4. Boolean formula → combinational circuit

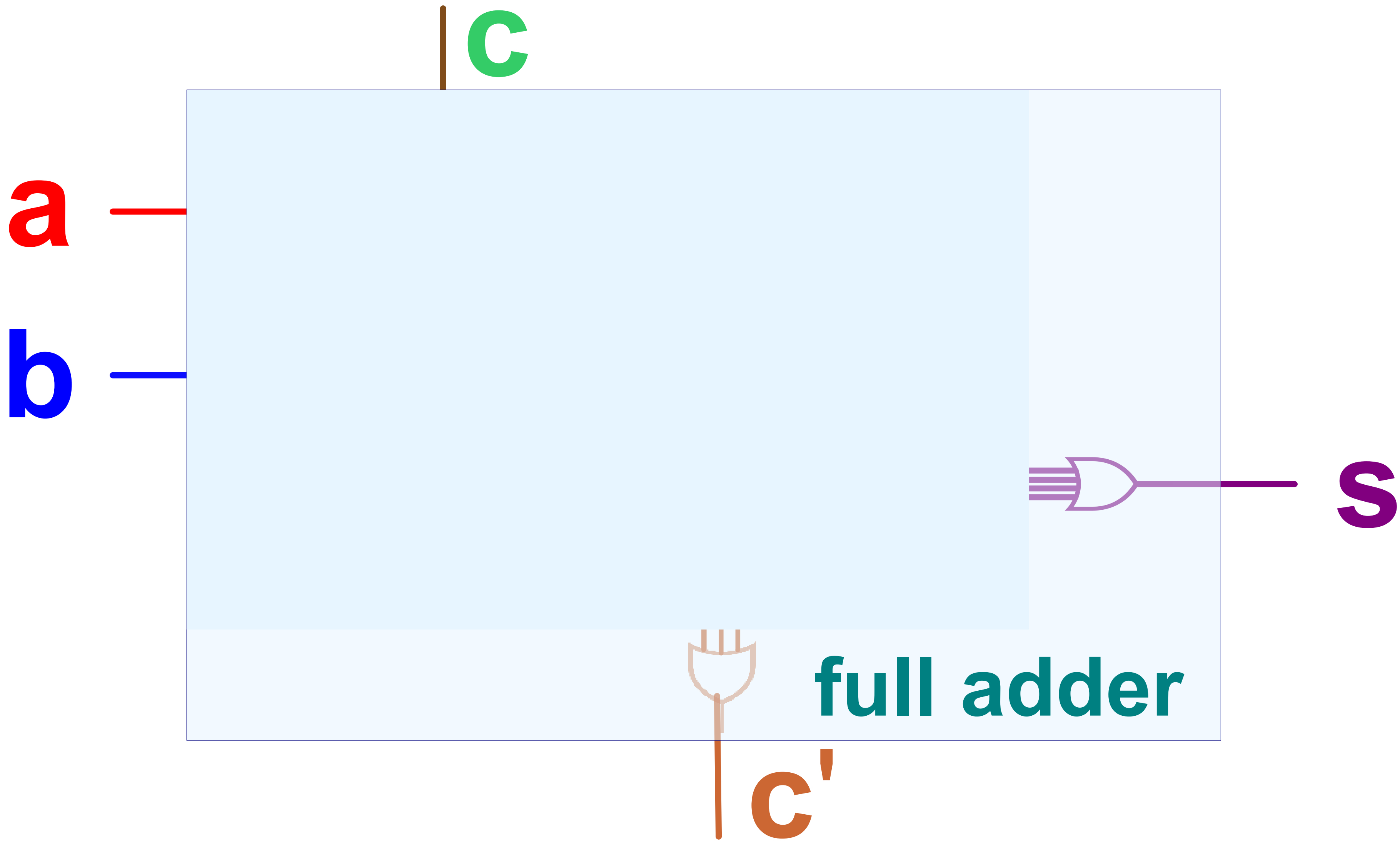
$$s = abc + \overline{ab}c + \overline{a}b\overline{c} + a\overline{bc}$$

$$c' = ab + ac + bc$$



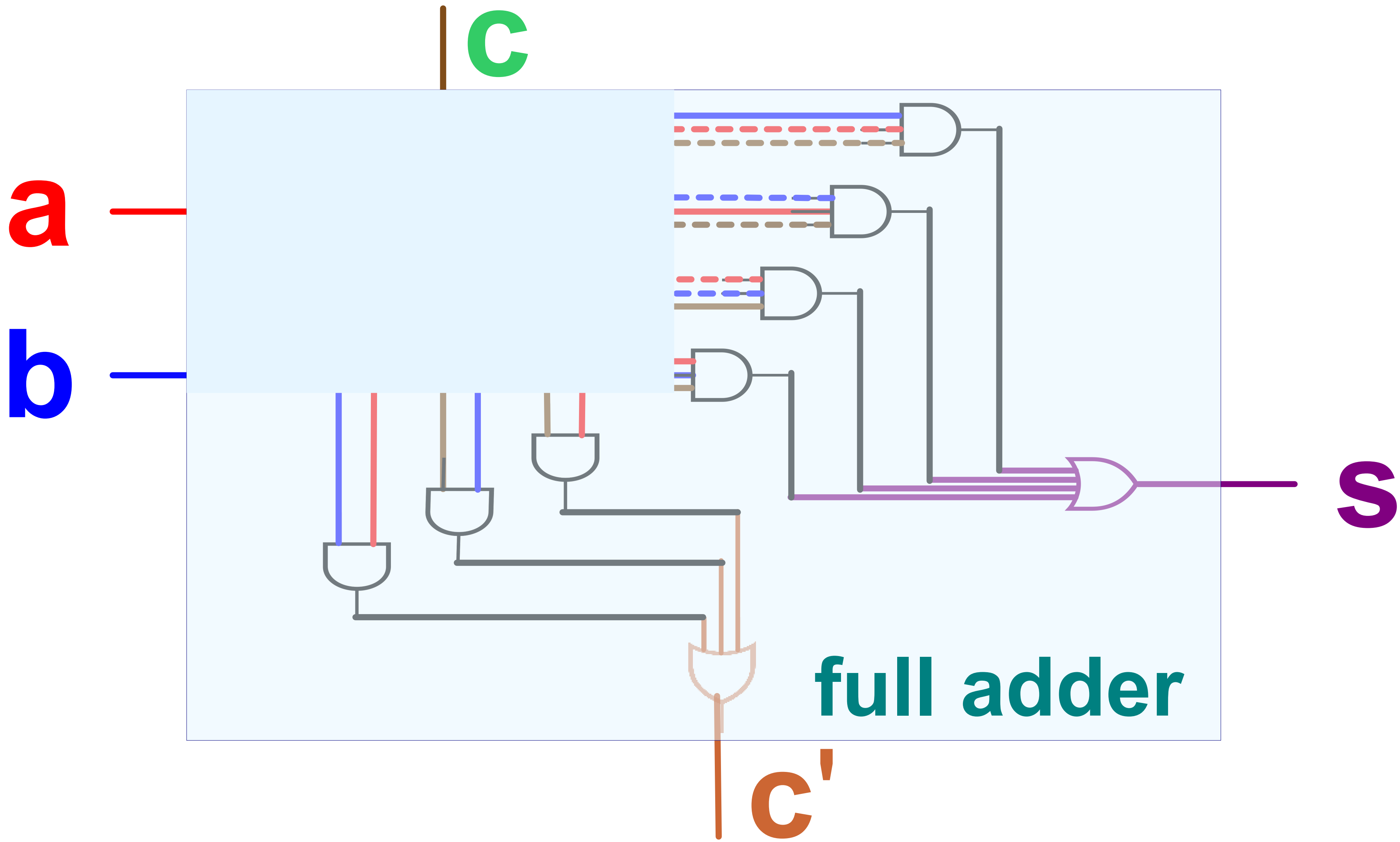
$$s = abc + \overline{ab}c + \overline{a}b\overline{c} + a\overline{bc}$$

$$c' = ab + ac + bc$$



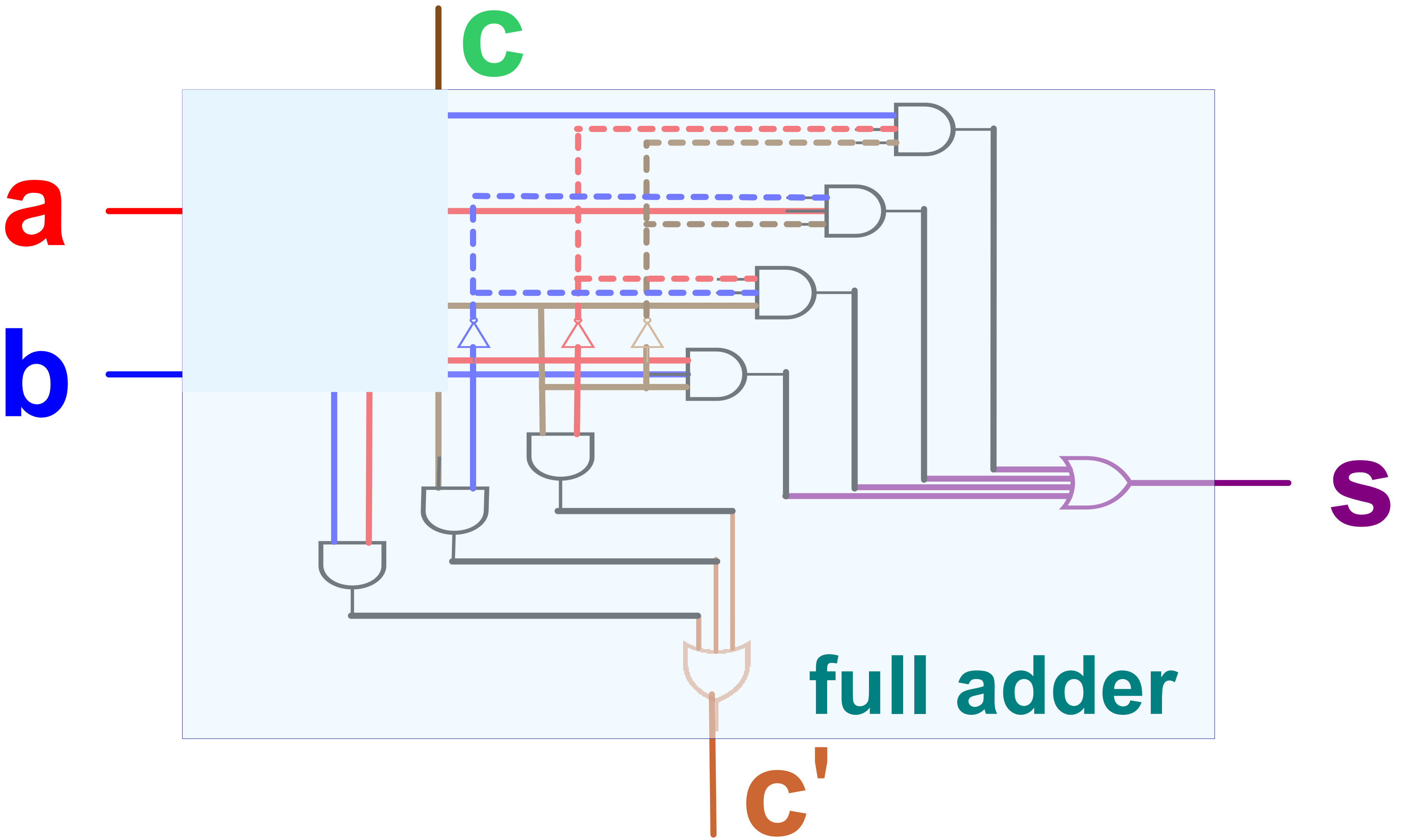
$$s = abc + \overline{ab}c + \overline{a}b\overline{c} + a\overline{bc}$$

$$c' = ab + ac + bc$$



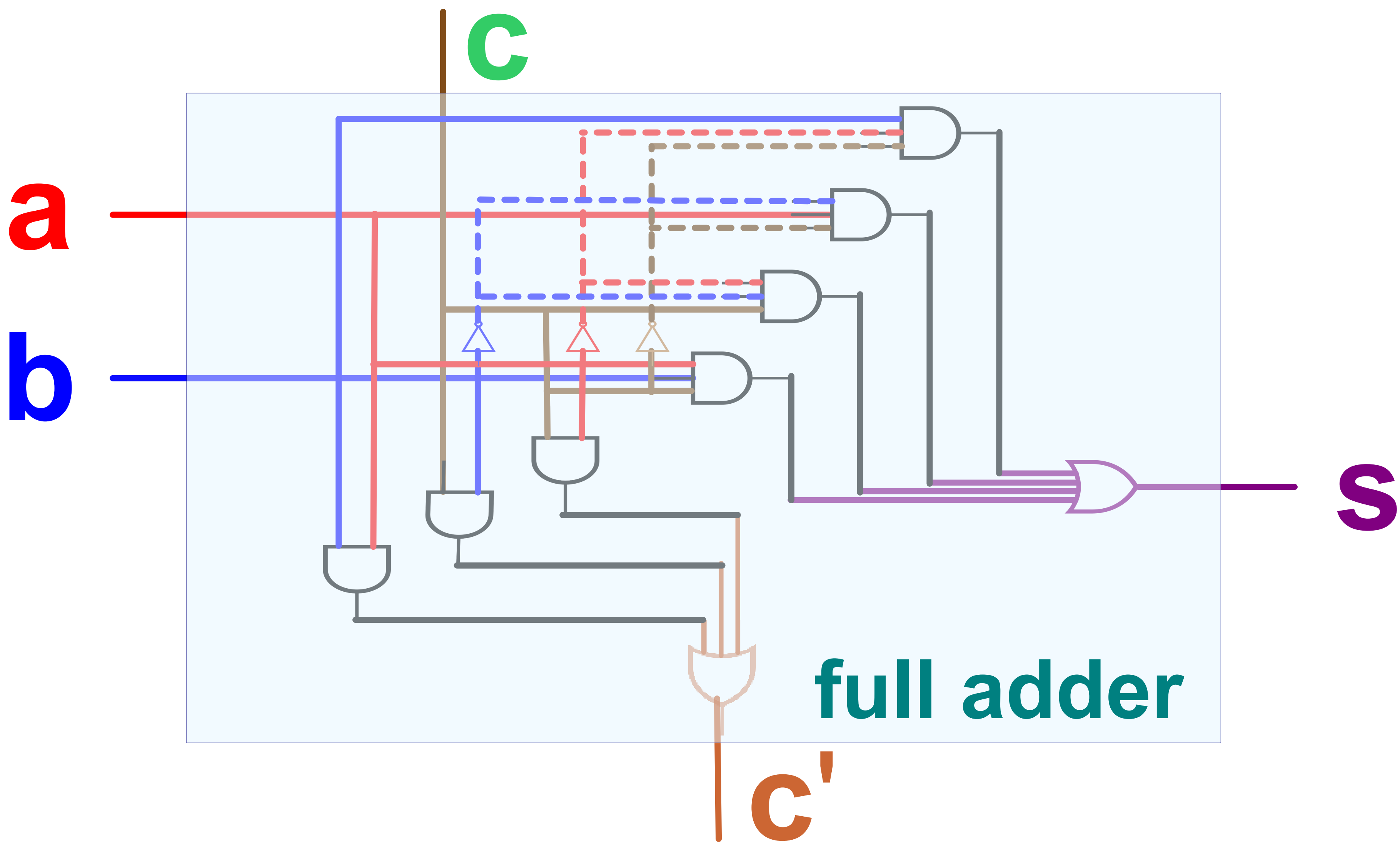
$$s = abc + \overline{ab}c + \overline{a}b\overline{c} + a\overline{bc}$$

$$c' = ab + ac + bc$$



$$s = abc + \overline{ab}c + \overline{a}b\overline{c} + a\overline{bc}$$

$$c' = ab + ac + bc$$



Inverters, Decoders, Multiplexer

Inverter: *select* data input or its negation

1 data input

1 selector input

1 output

Decoder: *select* unique output to be 1 (true)

N selector inputs

2^N outputs

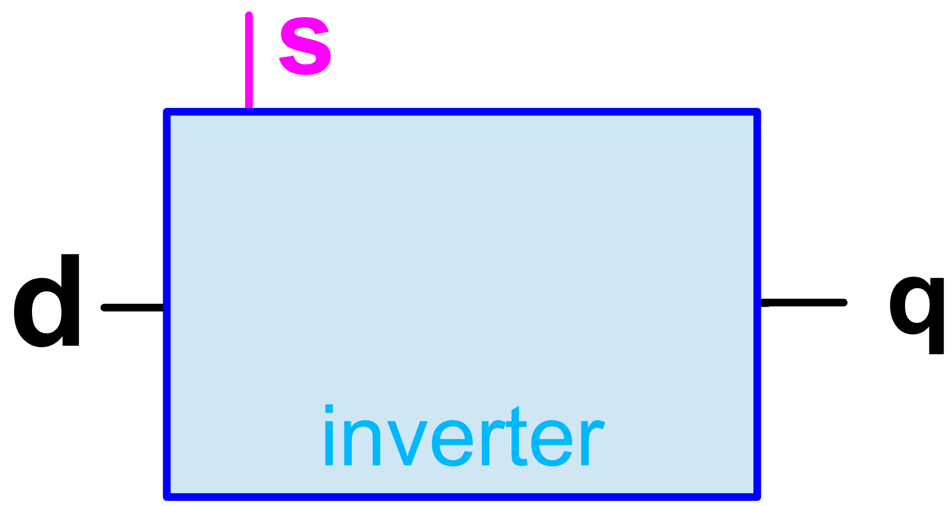
Multiplexer: *select* unique data input to be output

2^N data inputs

N selector inputs

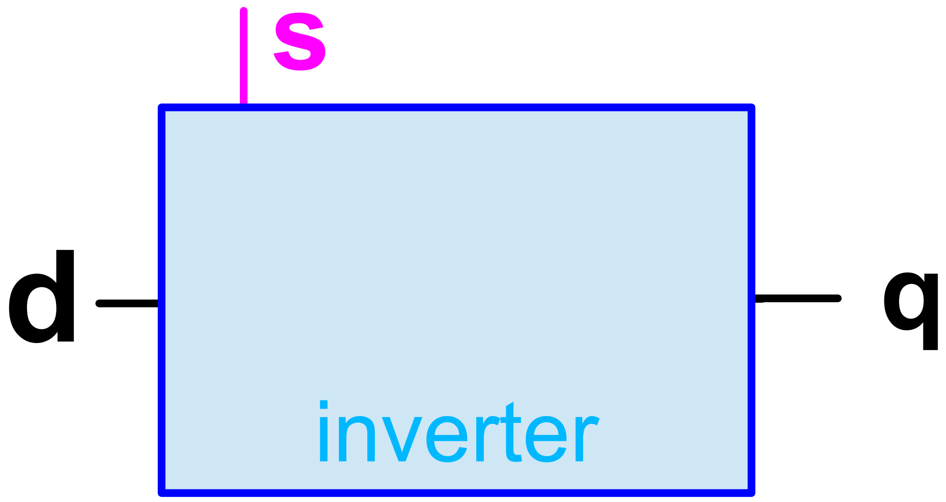
1 output

Inverter Black Box



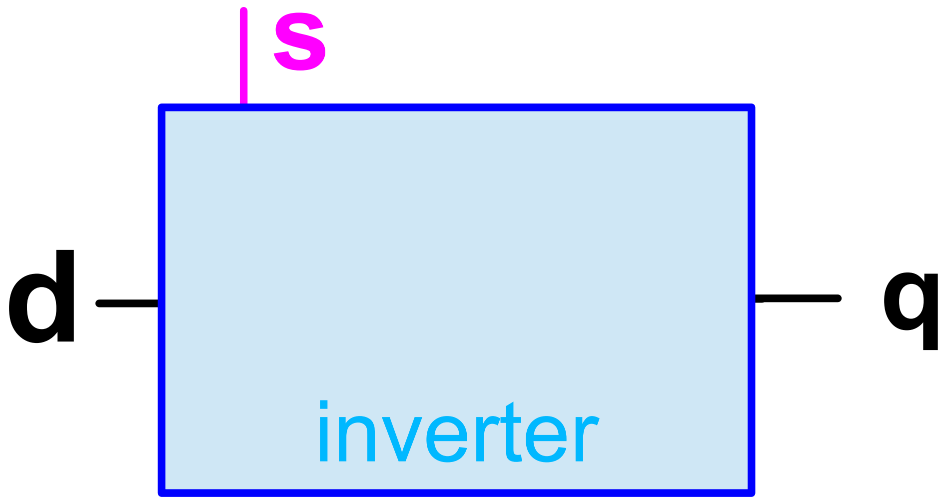
Inverter Truth Table

s	d	q
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**



Inverter Formula

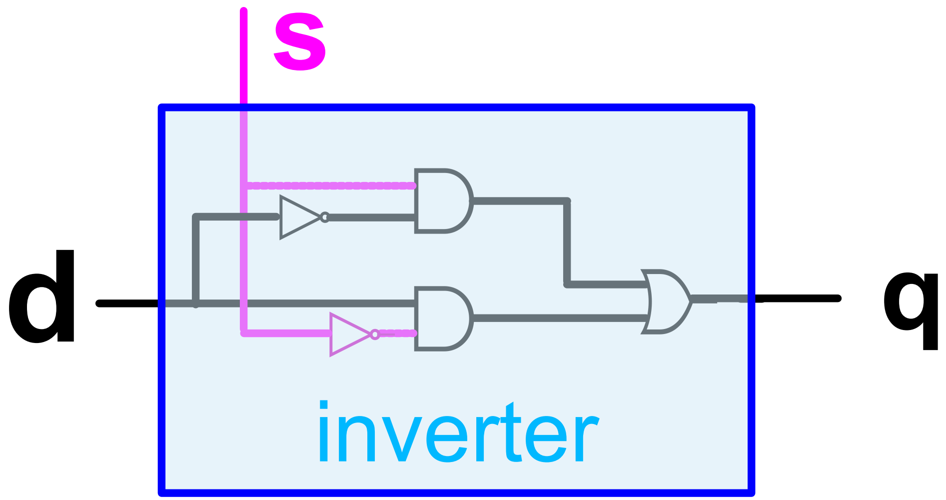
s	d	q
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**



$$q = \bar{s}d + s\bar{d}$$

Inverter Circuit

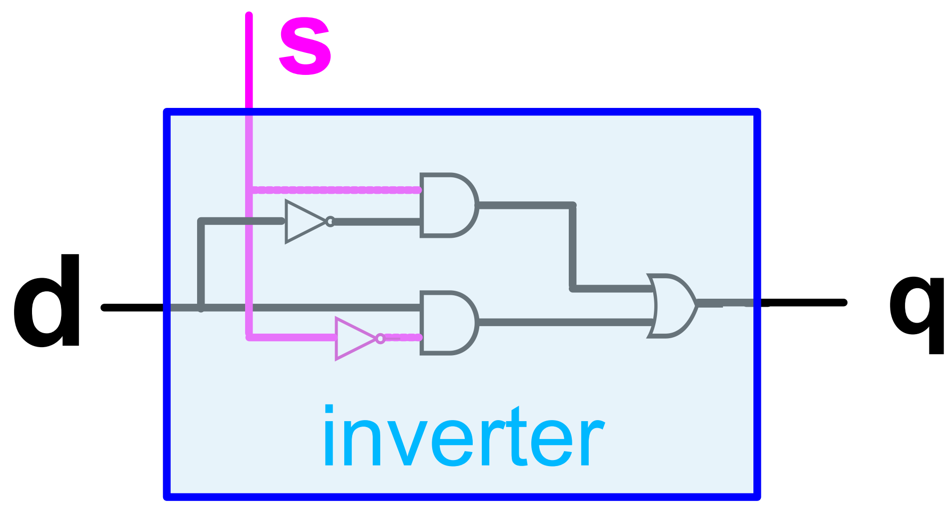
s	d	q
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**



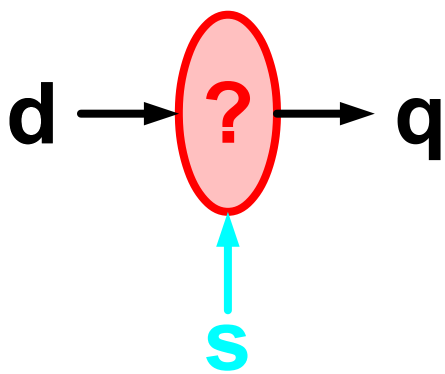
$$q = \overline{s}d + s\overline{d}$$

Inverter Component Icon

s	d	q
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**



$$q = \overline{s}d + s\overline{d}$$



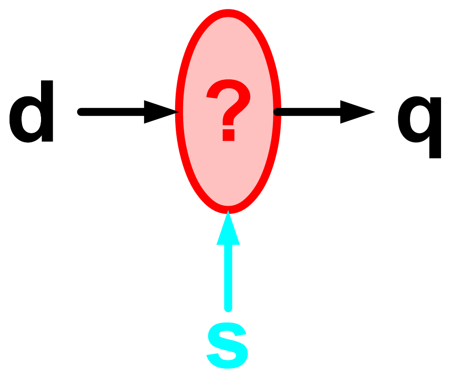
Inverter Summary

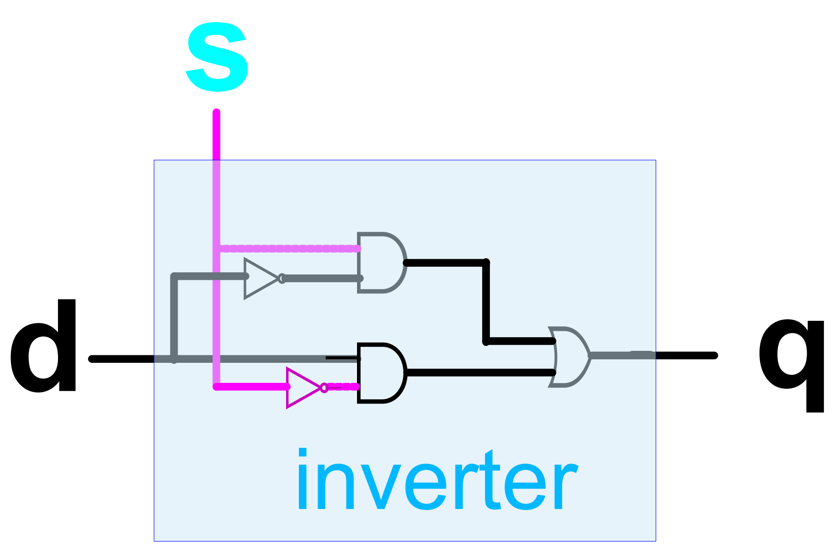
Inverter: *select* data input or its negation

1 data input

1 selector input

1 output





N-Bit Decoder

Each different combination of N input bits uniquely specifies one of 2^N outputs. An output is **1** if and only if the corresponding input combination is active (true). For any input, exactly one output is **1**.

N-Bit Decoder Truth Table:

- N input (selector) columns
- 2^N output columns
- 2^N rows

Exactly one 1 in each output column

Exactly one 1 in each output row

1-Bit Decoder Black Box

1-Bit Decoder Truth Table

s	w_0	w_1
0	1	0
1	0	1

1-Bit Decoder Formulas

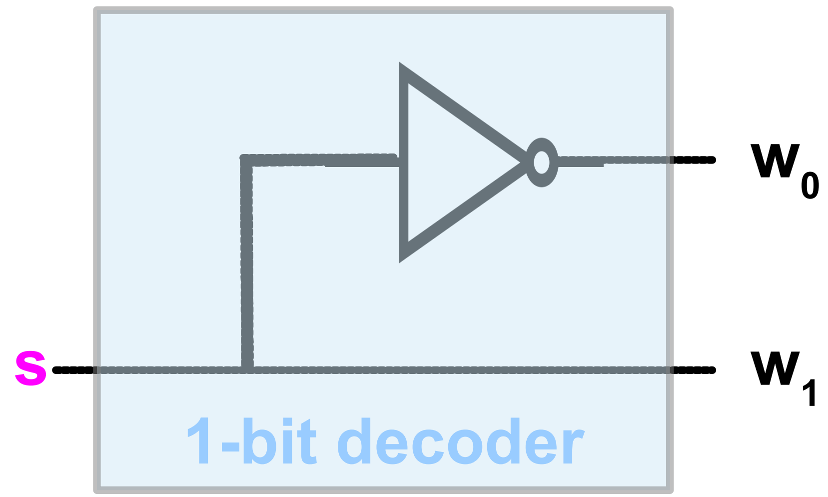
s	w_0	w_1
0	1	0
1	0	1

$$w_0 = \overline{s}$$

$$w_1 = s$$

1-Bit Decoder Circuit

s	w_0	w_1
0	1	0
1	0	1

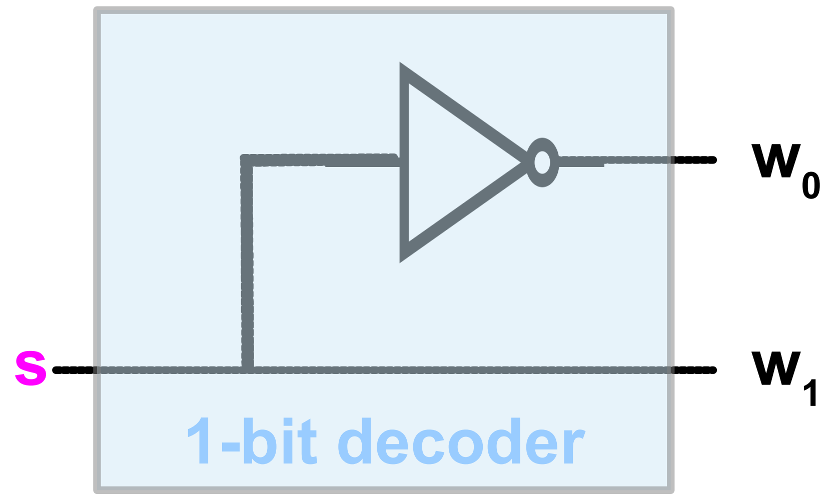


$$w_0 = \overline{s}$$

$$w_1 = s$$

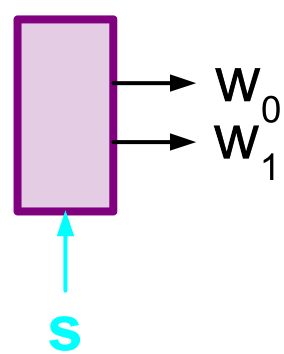
1-Bit Decoder Component Icon

s	w_0	w_1
0	1	0
1	0	1



$$w_0 = \overline{s}$$

$$w_1 = s$$



2-Bit Decoder

2-bit Decoder Truth Table

#	s_1	s_0	w_{00}	w_{01}	w_{10}	w_{11}
0	0	0	**1**	0	0	0
1	0	1	0	**1**	0	0
2	1	0	0	0	**1**	0
3	1	1	0	0	0	**1**

2-bit Decoder Formulas

#	s_1	s_0	w_{00}	w_{01}	w_{10}	w_{11}
0	0	0	**1**	0	0	0
1	0	1	0	**1**	0	0
2	1	0	0	0	**1**	0
3	1	1	0	0	0	**1**

$$w_{11} = s_1 s_0$$

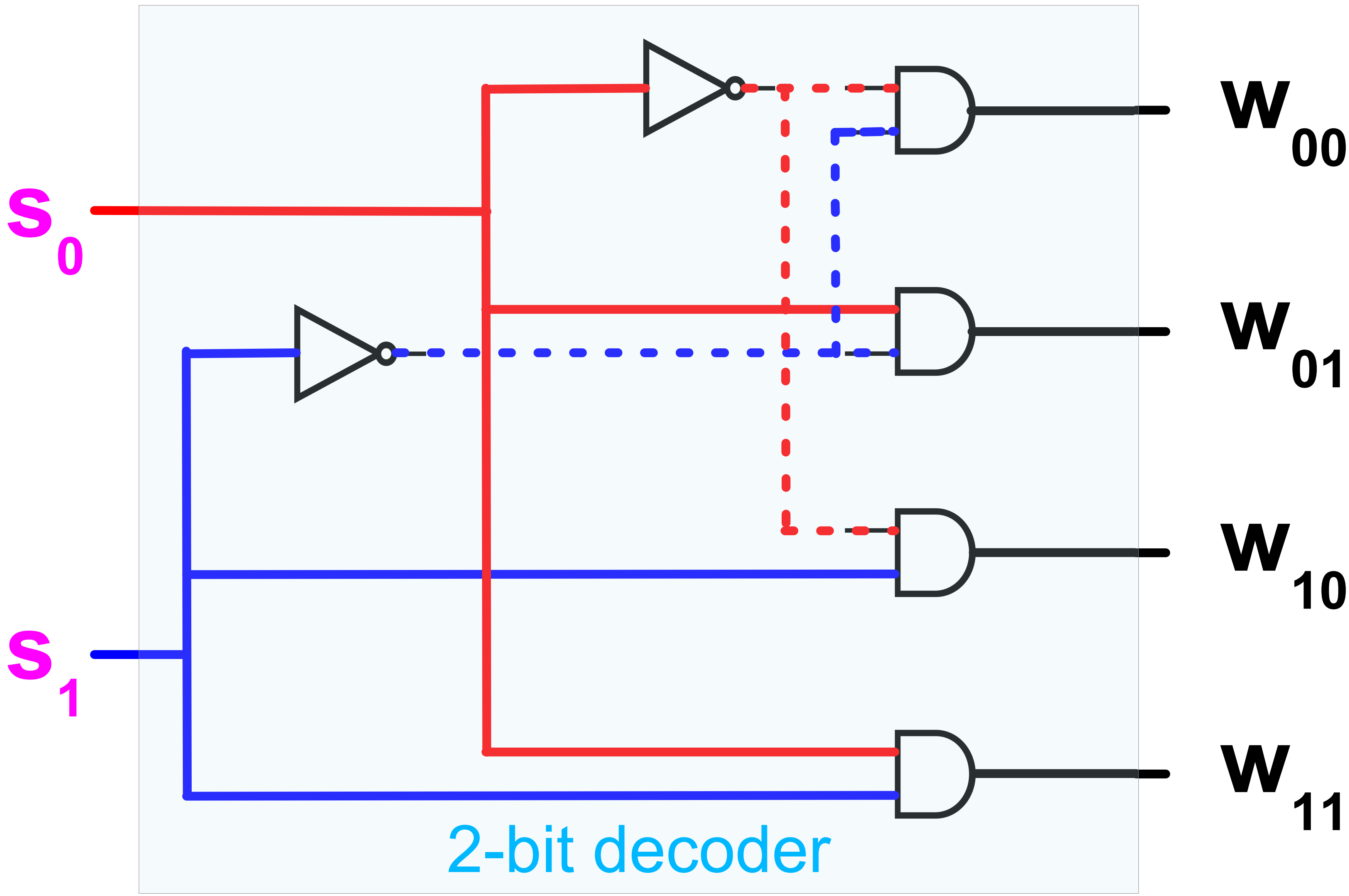
$$w_{10} = s_1 \overline{s_0}$$

$$w_{01} = \overline{s_1} s_0$$

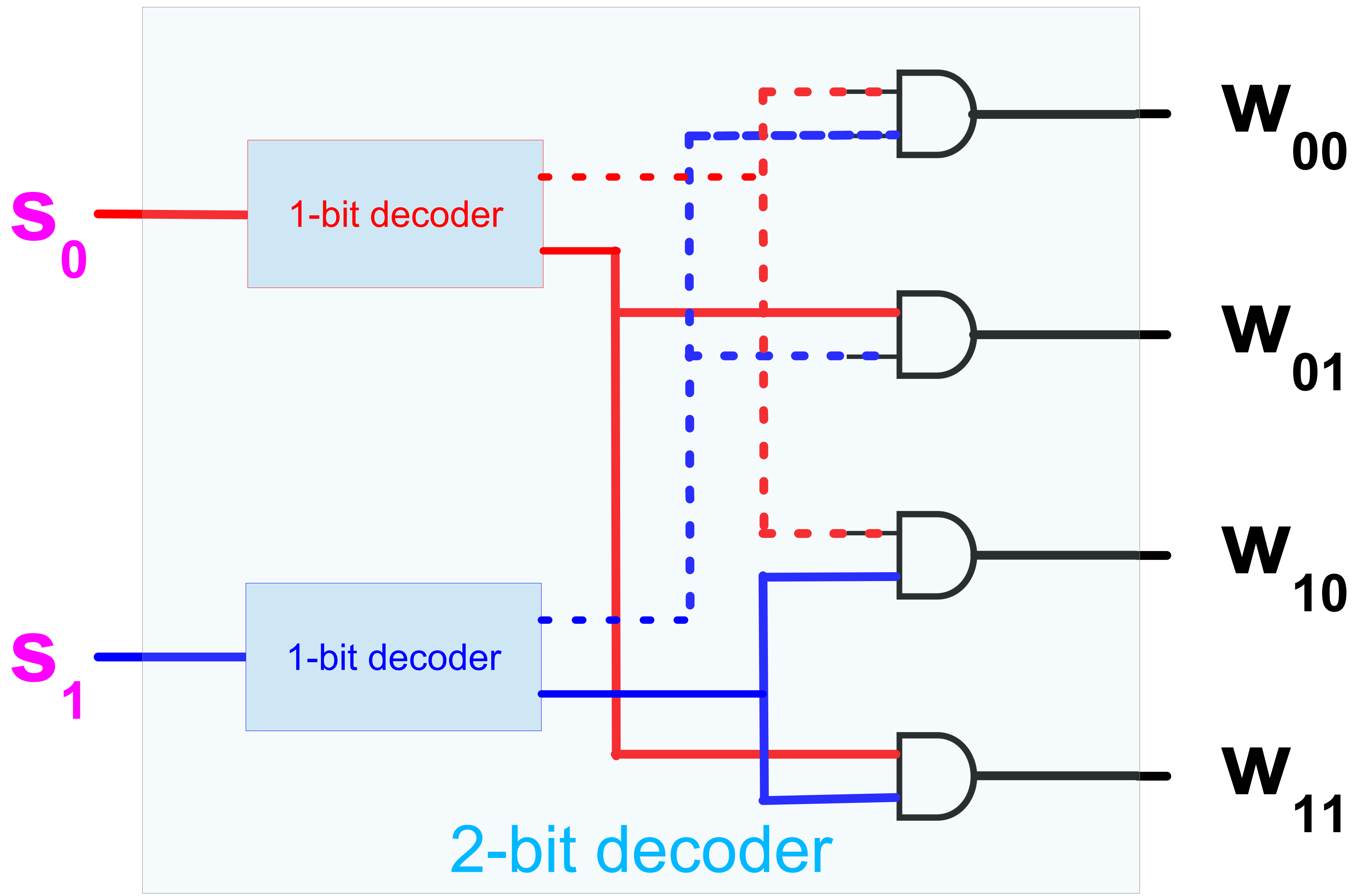
$$w_{00} = \overline{s_1}\,\overline{s_0}$$

2-Bit Decoder Circuit

2-Bit Decoder Circuit



2-Bit Decoder Circuit



3-Bit Decoder Black Box

3-bit Decoder Truth Table

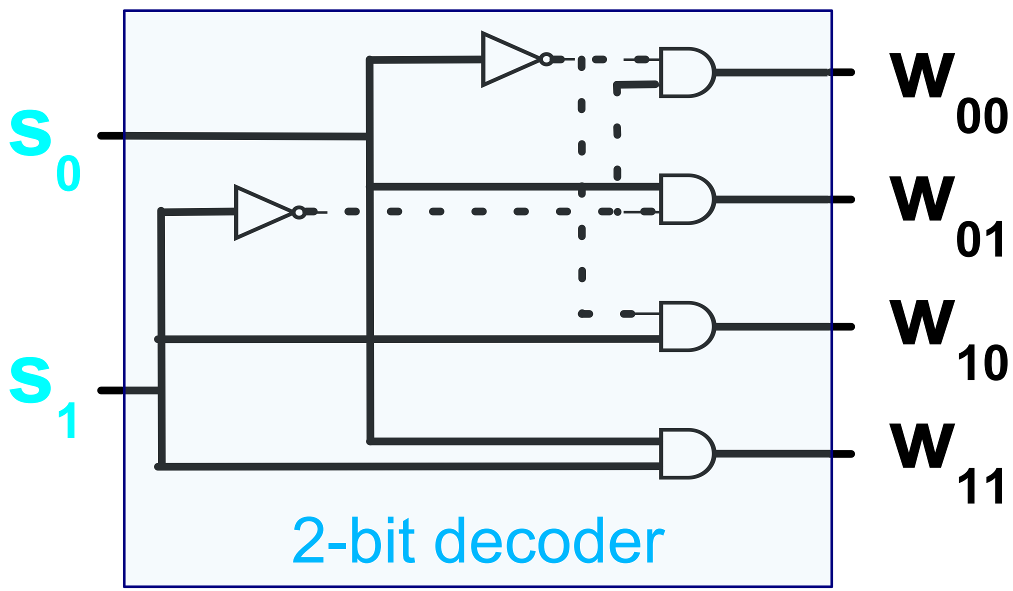
s_2	s_1	s_0	w_{000}	w_{010}	w_{010}	w_{011}	w_{100}	w_{111}	w_{110}	w_{111}
0	0	0	**1**	0	0	0	0	0	0	0
0	0	1	0	**1**	0	0	0	0	0	0
0	1	0	0	0	**1**	0	0	0	0	0
0	1	1	0	0	0	**1**	0	0	0	0
1	0	0	0	0	0	0	**1**	0	0	0
1	0	1	0	0	0	0	0	**1**	0	0
1	1	0	0	0	0	0	0	0	**1**	0
1	1	1	0	0	0	0	0	0	0	**1**

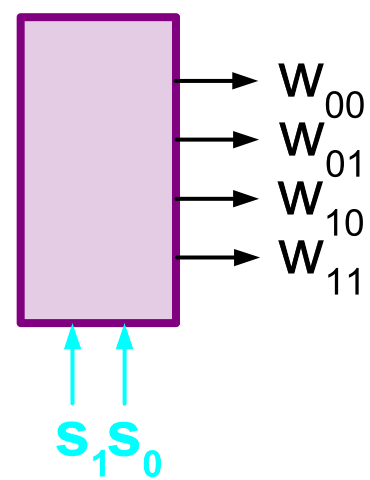
Decoder Summary

Decoder: *select* unique output to be 1 (true)

N selector inputs

2^N outputs





M-way Multiplexer ($M \equiv 2^N$)

N (lg M) *selector* inputs

- choose one of 2^N (M) data inputs to output

2^N-way multiplexer truth table:

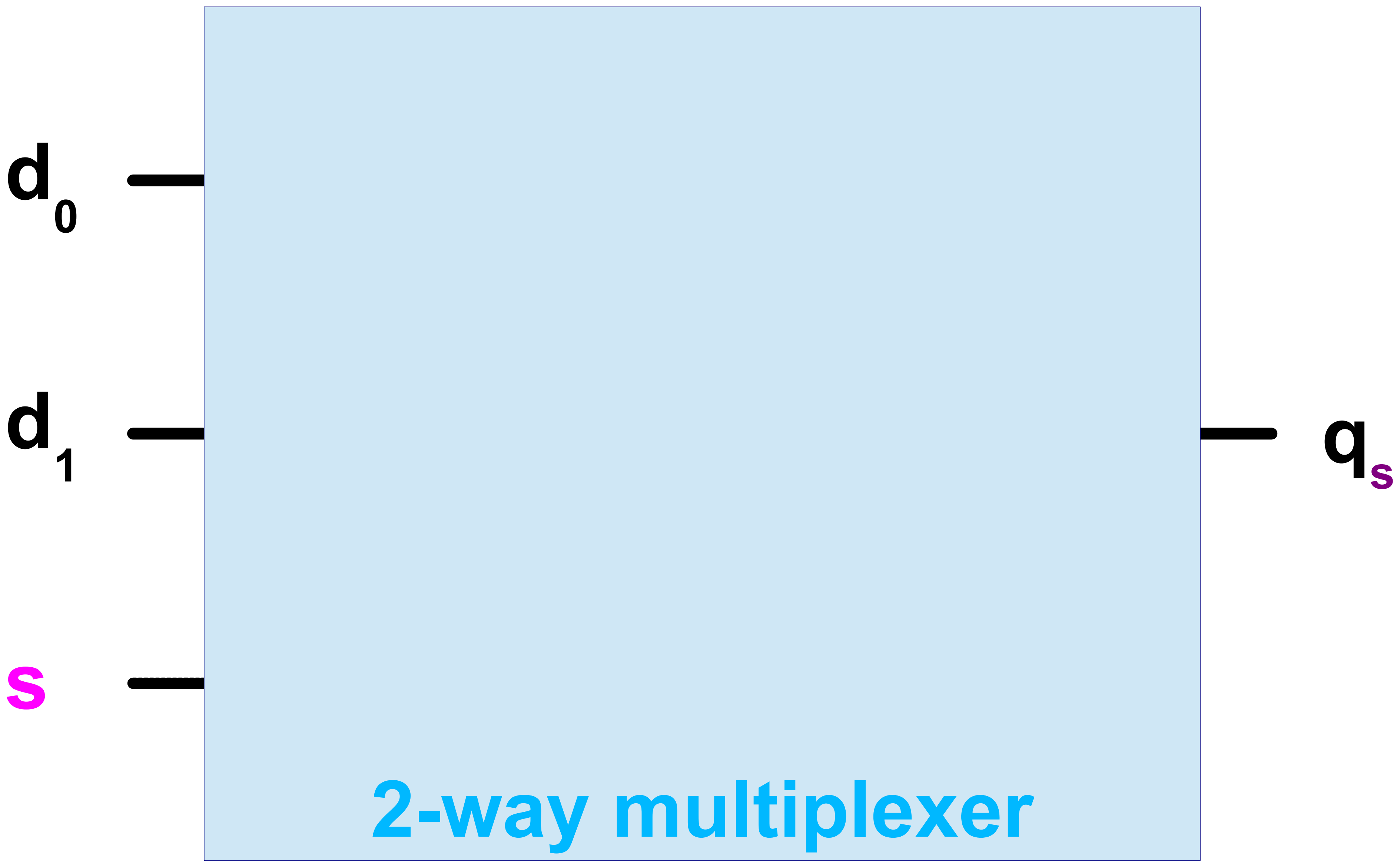
- 2^N data input columns ($\mathbf{d}_i$ $0 \le \mathbf{i} < 2^N$)
- N selector input columns ($\mathbf{s}_j$ $0 \le \mathbf{j} < N$, $0 \le \mathbf{s} < 2^N$)
- 1 output column ($\mathbf{x} = \mathbf{d}_s$)

2-Way **Multiplexer**

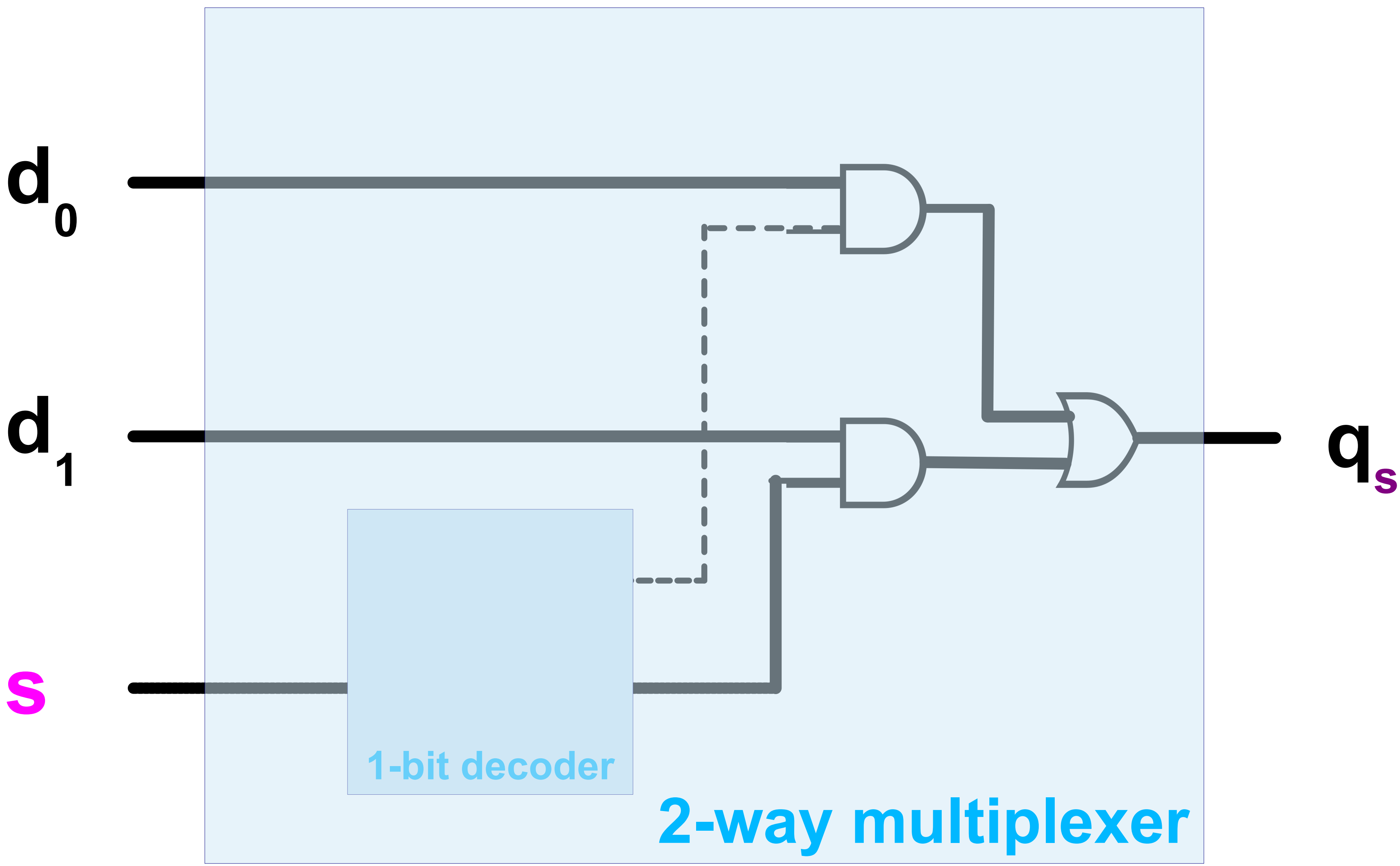
2-Way **Multiplexer**

#	s	d_1	d_0	q
0	**0**	0	**0**	**0**
1	**0**	0	**1**	**1**
2	**0**	1	**0**	**0**
3	**0**	1	**1**	**1**
4	**1**	**0**	0	**0**
5	**1**	**0**	1	**0**
6	**1**	**1**	0	**1**
7	**1**	**1**	1	**1**

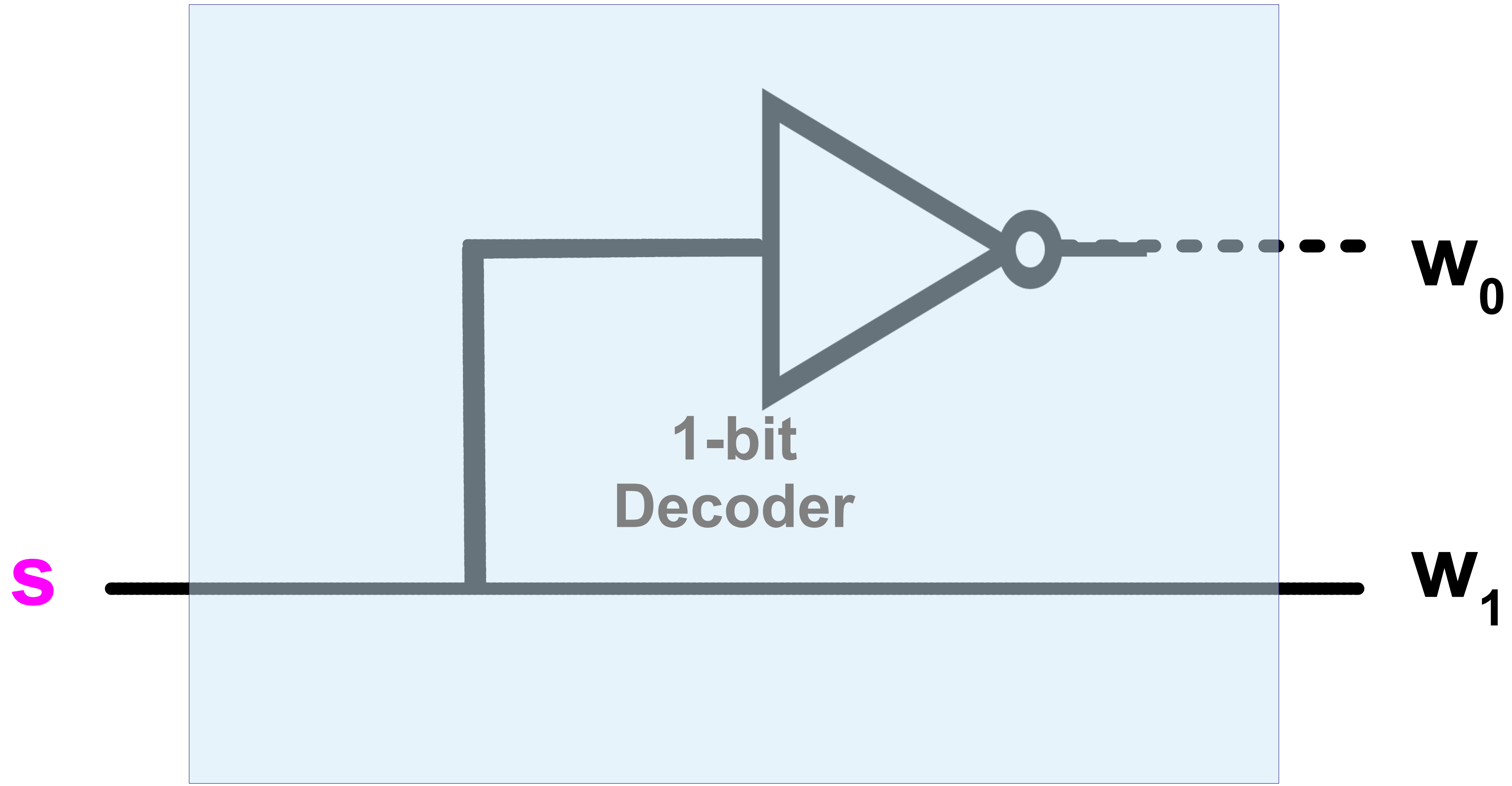
2-Way **Multiplexer**



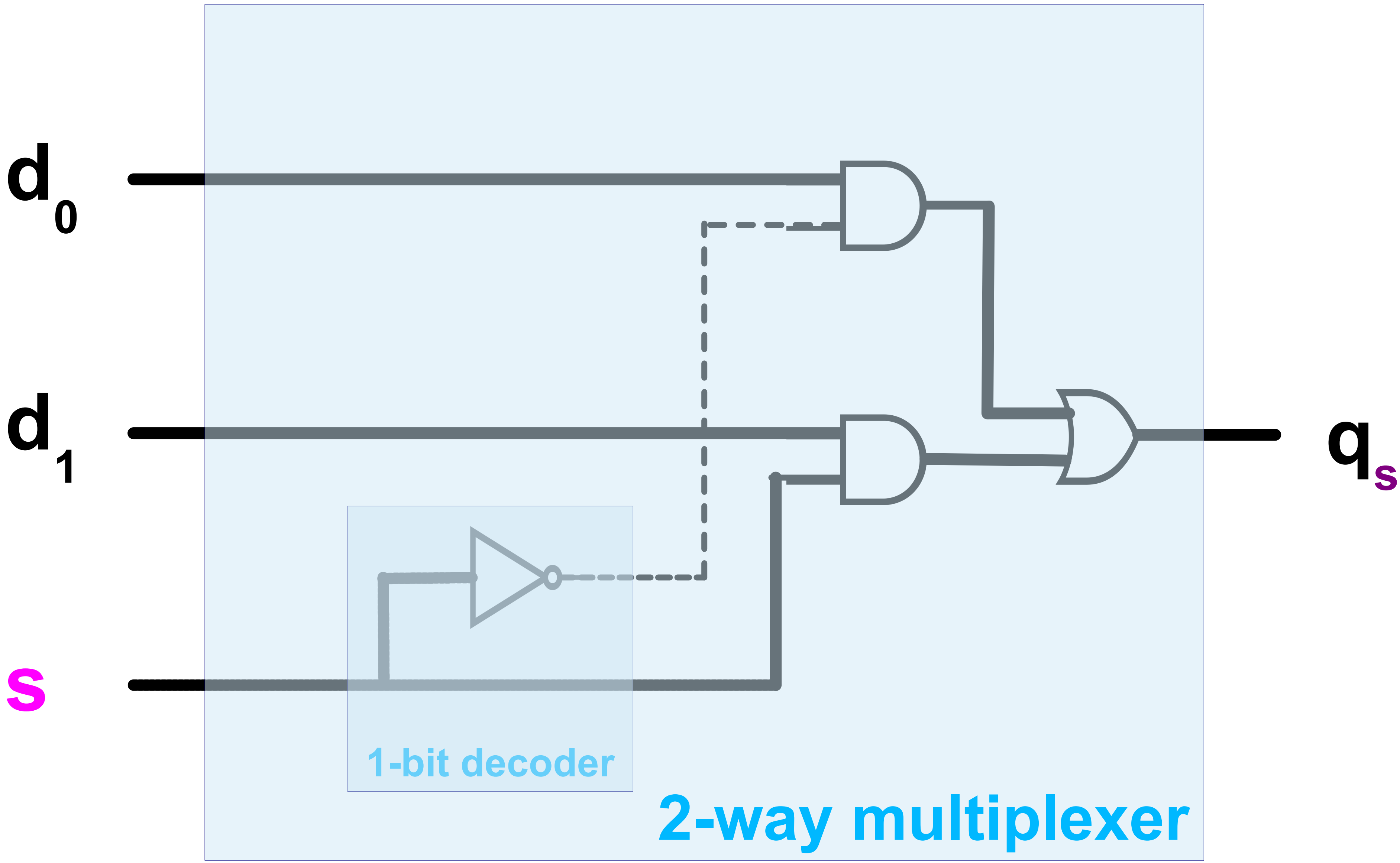
2-Way **Multiplexer**



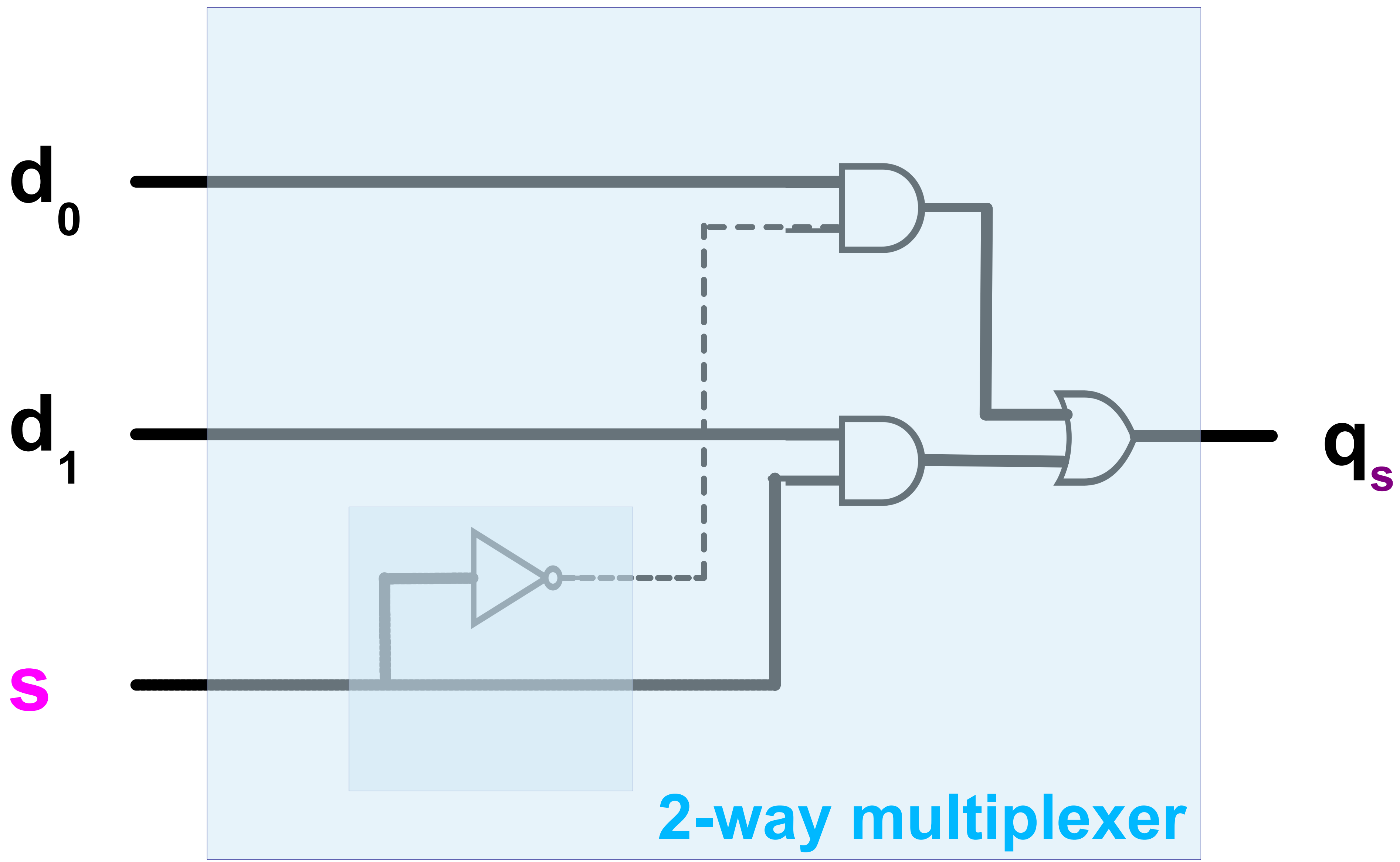
1-Bit Decoder



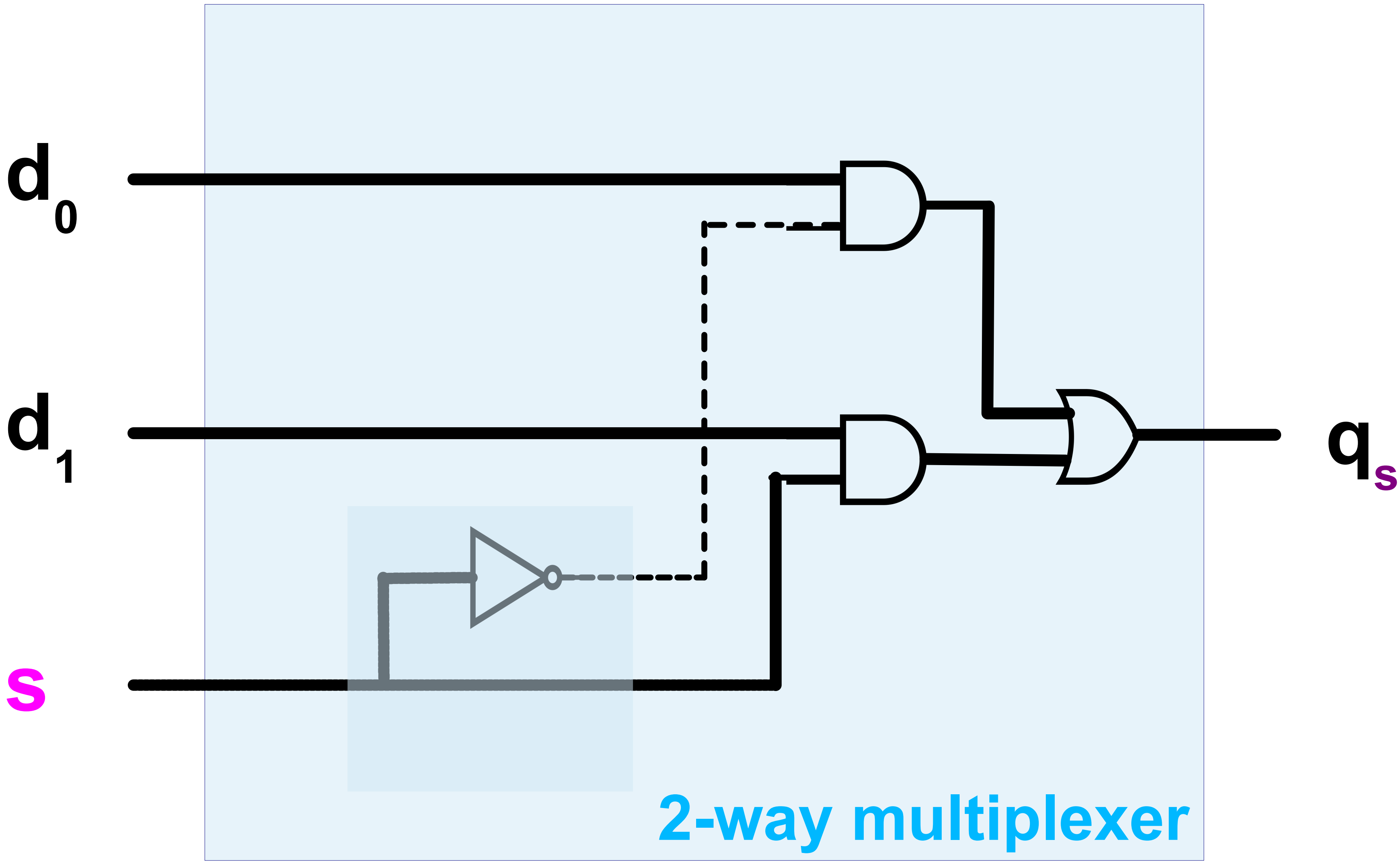
2-Way **Multiplexer**



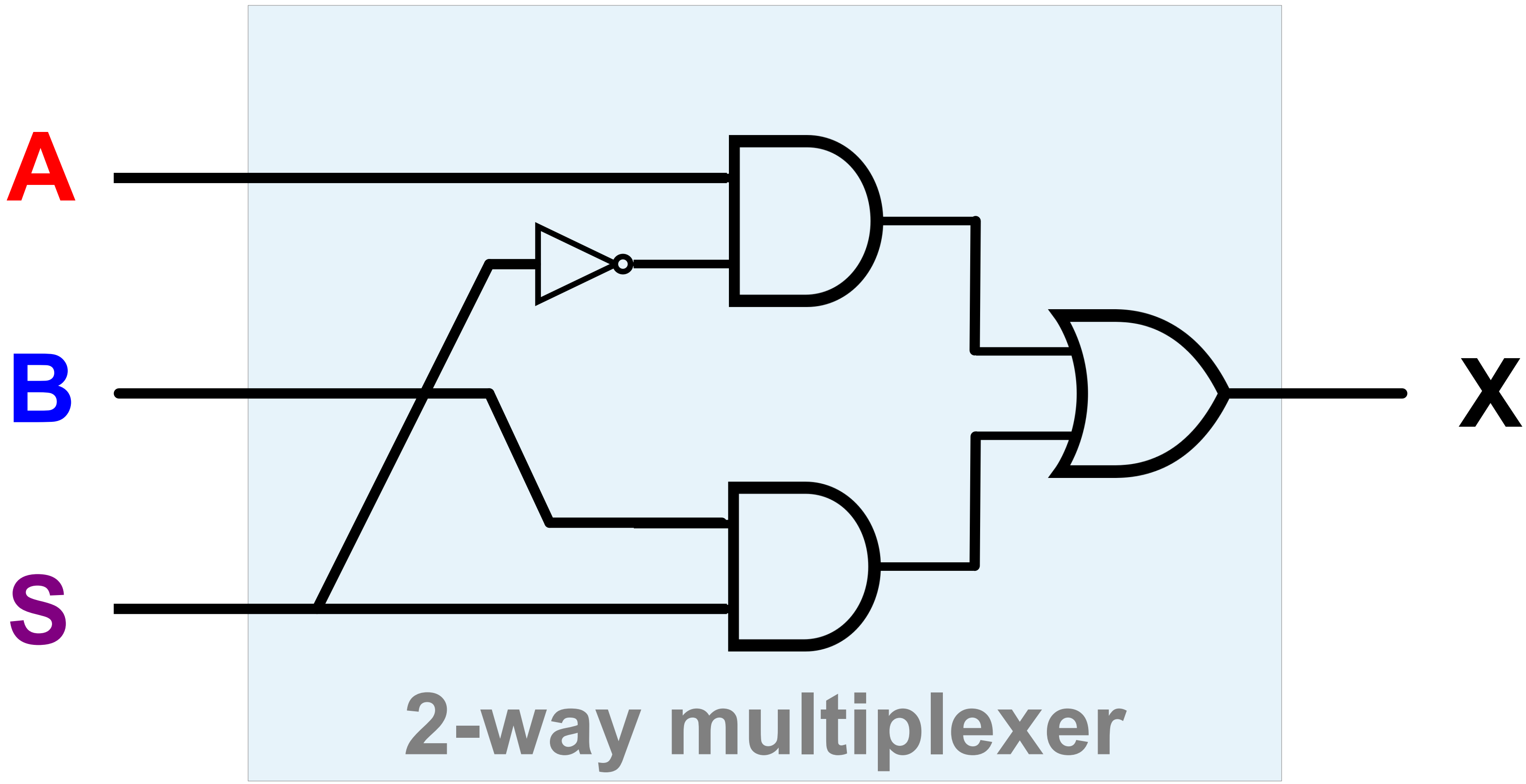
2-Way **Multiplexer**



2-Way **Multiplexer**



Two Way Multiplexer Circuit



$$X = \overline{S}A + SB$$

4-Way Multiplexer Black Box

4-Way Multiplexer

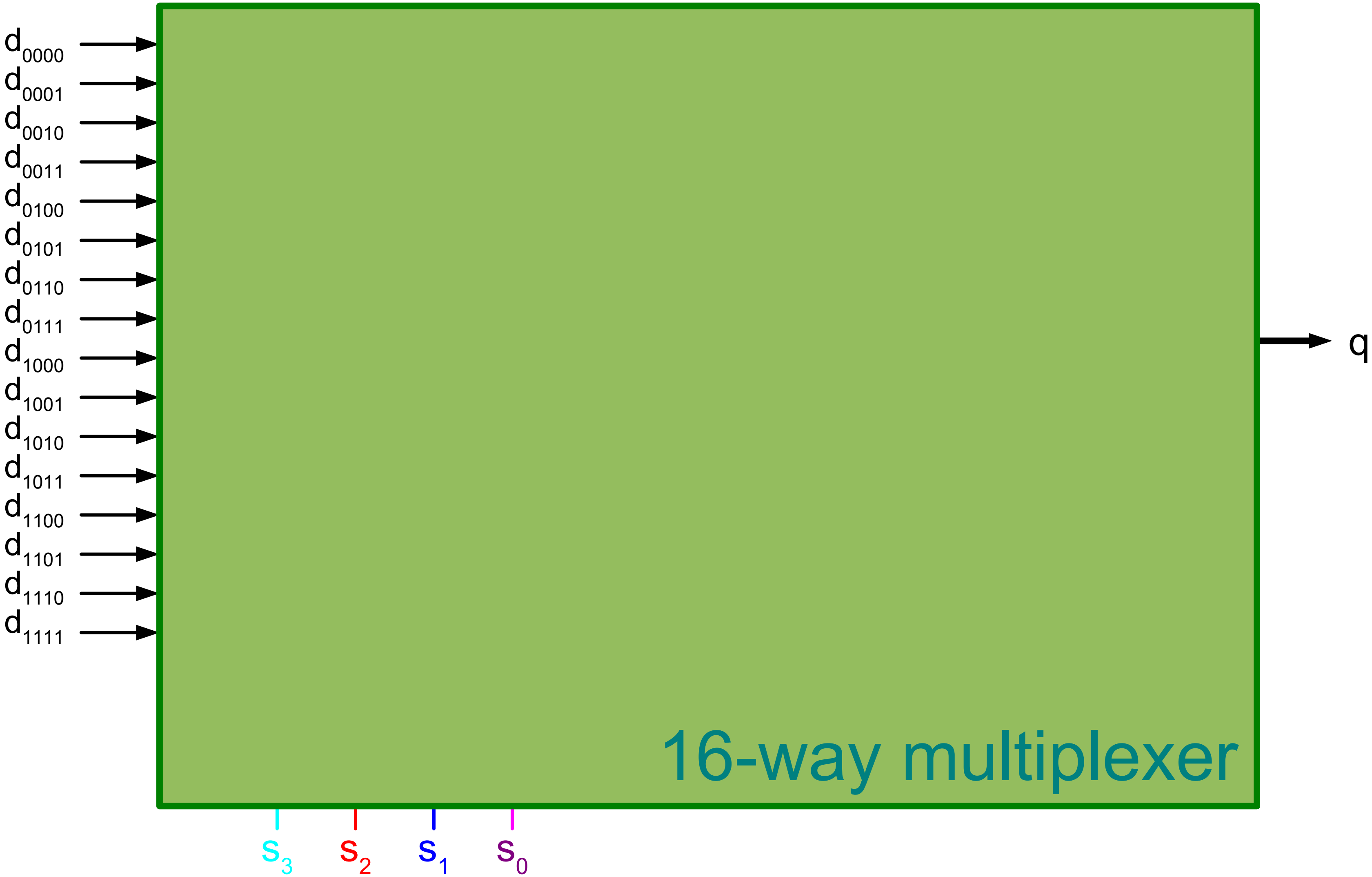
s_1	s_0	d_{11}	d_{10}	d_{01}	d_{00}	q
0	0	√	√	√	0	0
0	0	√	√	√	1	1
0	1	√	√	0	√	0
0	1	√	√	1	√	1
1	0	√	0	√	√	0
1	0	√	1	√	√	1
1	1	0	√	√	√	0
1	1	1	√	√	√	1

4-Way Multiplexer

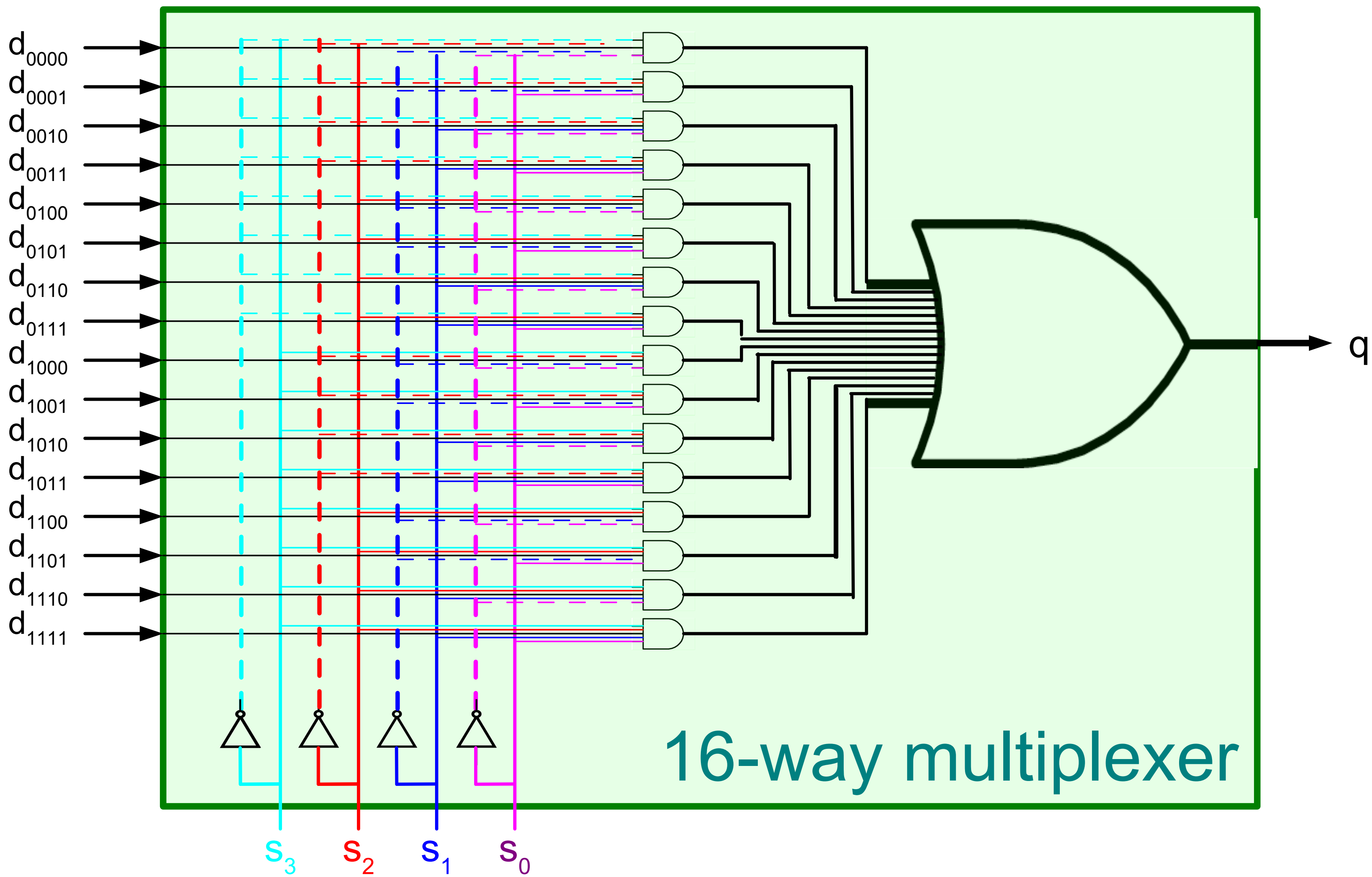
s_1	s_0	d_{11}	d_{10}	d_{01}	d_{00}	q	
0	0	√	√	√	0	0	
0	0	√	√	√	1	1	$\overline{s}_1\overline{s}_0 d_{00}$
0	1	√	√	0	√	0	
0	1	√	√	1	√	1	$+\ \overline{s}_1 s_0 d_{01}$
1	0	√	0	√	√	0	
1	0	√	1	√	√	1	$+\ s_1\overline{s}_0 d_{10}$
1	1	0	√	√	√	0	
1	1	1	√	√	√	1	$+\ s_1 s_0 d_{11}$

4-Way Multiplexer

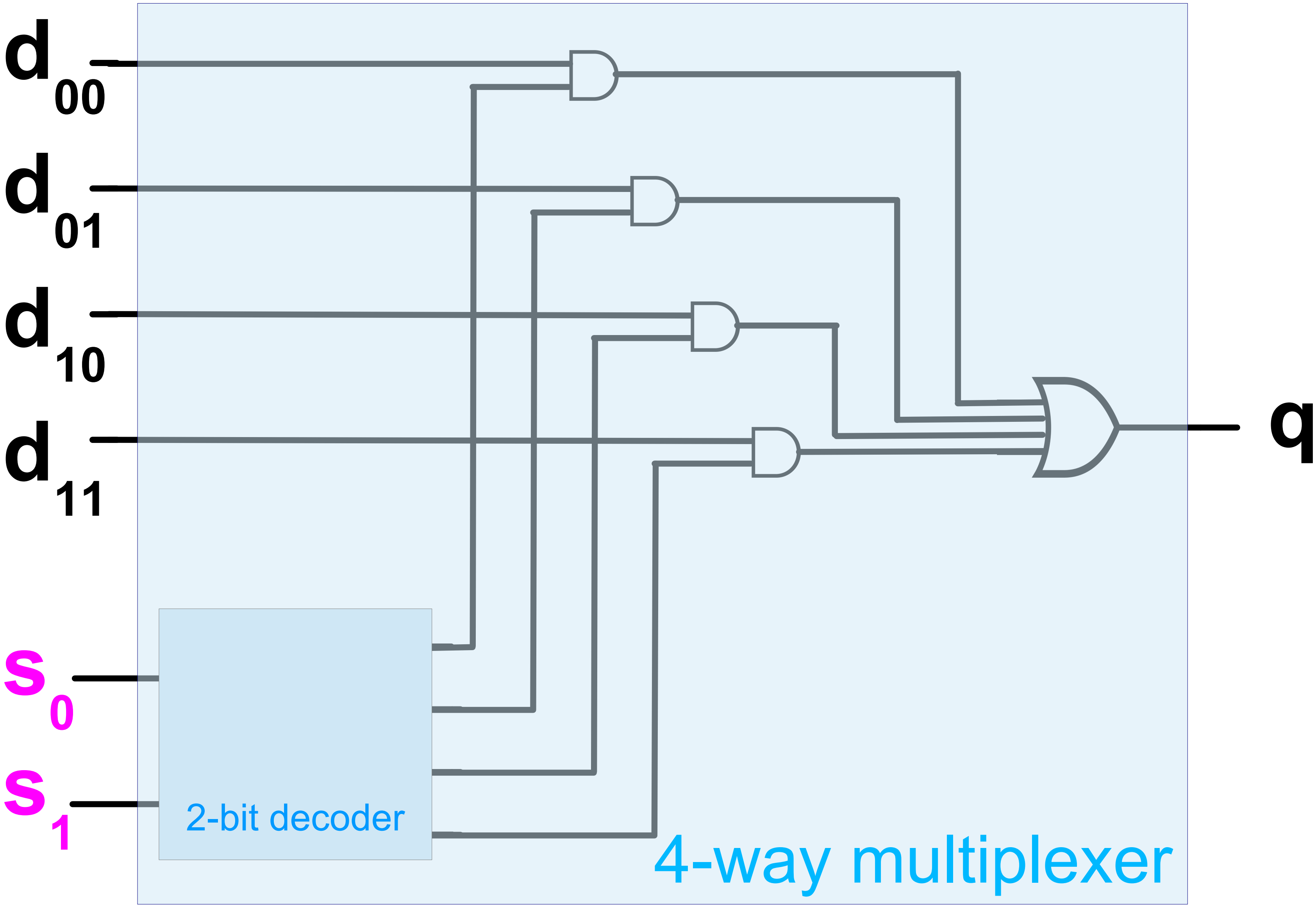
16-Way Multiplexer



16-Way Multiplexer



4-Way Multiplexer



2^N-Way Multiplexer Summary

N selector inputs specify one of 2^N data inputs to output