

# CMP 334: *Seventh Class*

## Performance

- HW 5 solution

- Averages and weighted averages (review)

- Amdahl's law

## Ripple-carry adder circuits

- Binary addition

- Half-adder circuits

- Full-adder circuits

## Subtraction, negative numbers, signed arithmetic

$$A - B = A + -B$$

For next class: HW 6; read A.3-4, 2.1-5, 3.1-2

# HW 5: Performance Problems

- 1) Computer A has a 5 GHz clock and executes program P in 30 seconds with an average CPI (cycles per instruction) of 3.0. How many instructions does it execute for program P?
- 2) Computer B has a 2 GHz clock. It executes P with the same number of instructions as computer A with an average CPI 1.0. How long does it take to execute P?
- 3) Compare the performance of computer B and computer A on program P. Which is faster? by how much?
- 4) A new compiler for computer A compiles program P so that it executes only half as many instructions. Unfortunately, the CPI for computer A on these instructions is 4.0. How long does it take to execute the newly compiled program ?
- 5) Compare the performance of computer B (with the old compiler) to computer A (with the new compiler) on program P. Which is faster? by how much?

# Performance Equations

Performance – inverse of execution time

$$\text{performance: } P_x \equiv \frac{1}{T_x} \quad \text{relative performance: } \frac{P_x}{P_y} = \frac{T_y}{T_x}$$

## CPU time equation

$$T_{CPU}(\text{execution}) = \frac{\# \text{ instructions}}{\text{execution}} \cdot \frac{\# \text{ cycles}}{\text{instruction}} \cdot \frac{\# \text{ seconds}}{\text{cycle}}$$

## Amdahl's law

$$T_{new} = \frac{\text{fraction affected} \cdot T_{old}}{\text{improvement}} + \text{fraction not affected} \cdot T_{old}$$

# Processor Performance Equations

$$T_X = \# \text{ instructions}_X \cdot \text{CPI}_X \cdot \text{cycleTime}_X$$

$$T_X = \frac{\# \text{ instructions}_X \cdot \text{CPI}_X}{\text{clockRate}_X}$$

$$\frac{P_X}{P_Y} = \frac{T_Y}{T_X} = \frac{\# \text{ instructions}_Y \cdot \text{CPI}_Y \cdot \text{cycleTime}_Y}{\# \text{ instructions}_X \cdot \text{CPI}_X \cdot \text{cycleTime}_X}$$

$$\frac{P_X}{P_Y} = \frac{T_Y}{T_X} = \frac{\# \text{ instructions}_Y \cdot \text{CPI}_Y \cdot \text{clockRate}_X}{\# \text{ instructions}_X \cdot \text{CPI}_X \cdot \text{clockRate}_Y}$$

# HW 5.1 Instruction Count

Computer **A** has a **5 GHz clock** and executes program P in **30 seconds** with an average **CPI** (cycles per instruction) of **3.0**. How many **instructions** does it execute for program P?

$$T_A = \frac{\# \text{ instructions}_A \cdot \text{CPI}_A}{\text{clockRate}_A}$$

$$30 \text{ seconds} = \frac{\# \text{ instructions}_A \cdot 3.0 \text{ cycles / instruction}}{5 \text{ GHz}}$$

$$\begin{aligned} \# \text{ instructions}_A &= \frac{30 \text{ seconds} \cdot 5 \cdot 10^9 \text{ cycles / second}}{3.0 \text{ cycles / instruction}} \\ &= 50 \cdot 10^9 \text{ instructions} \end{aligned}$$

# HW 5.2 Execution Time

Computer **B** has a **2 GHz** clock. It executes P with **the same number** of instructions as computer **A** ( **$50 \cdot 10^9$  instructions**) with an average **CPI 1.0**. How long does **it** take to execute P?

$$\begin{aligned} T_B &= \frac{\# \text{ instructions}_B \cdot \text{CPI}_B}{\text{clockRate}_B} \\ &= \frac{50 \cdot 10^9 \text{ instructions} \cdot 1.0 \text{ cycles / instruction}}{2 \text{ GHz}} \\ &= \frac{50 \cdot 10^9 \text{ instructions} \cdot 1 \text{ cycles / instruction}}{2 \cdot 10^9 \text{ cycles / second}} \\ &= \mathbf{25 \text{ seconds}} \end{aligned}$$

# HW 5.3 Relative Performance

Compare the performance of computer **B** and computer **A** on program P.  
Which is faster? by how much?

$$\frac{P_B}{P_A} = \frac{T_A}{T_B} = \frac{30 \text{ seconds}}{25 \text{ seconds}} = 1.2$$

**B** is 1.2 times faster than **A**.

# HW 5.4 Execution Time

A **new compiler** for computer **A** compiles program P so that it executes only **half as many instructions**. Unfortunately, the **CPI** for computer **A** on *these* instructions is **4.0**. How long does it take to execute the newly compiled program ?

$$\begin{aligned} T_{A'} &= \frac{\# \text{ instructions}_{A'} \cdot \text{CPI}_{A'}}{\text{clockRate}_{A'}} \\ &= \frac{\frac{1}{2} \cdot \# \text{ instructions}_A \cdot 4.0 \cdot \text{cycles} / \text{instruction}}{\text{clockRate}_A} \\ &= \frac{0.5 \cdot 50 \cdot 10^9 \text{ instructions} \cdot 4.0 \cdot \text{cycles} / \text{instruction}}{5 \cdot 10^9 \text{ cycles} / \text{second}} \\ &= \frac{100}{5} \text{ seconds} = 20 \text{ seconds} \end{aligned}$$



# HW 5.5 Relative Performance

Compare the performance of computer **B** (with the old compiler) to computer **A'** (**A** with the new compiler) on program P. Which is faster? by how much?

$$\frac{P_{A'}}{P_B} = \frac{T_B}{T_{A'}} = \frac{25 \text{ seconds}}{20 \text{ seconds}} = 1.25$$

**A'** is 1.25 times faster than **B**.

$$\frac{P_{A'}}{P_A} = \frac{T_A}{T_{A'}} = \frac{30 \text{ seconds}}{20 \text{ seconds}} = 1.5$$

**A'** is 1.5 times faster than **A**.

# Averages and Weighted Averages

Given values:  $\{v_1, v_2, \dots, v_N\}$  & weights:  $\{w_1, w_2, \dots, w_N\}$

$$\textit{average: } \bar{v} \equiv \frac{\sum_{i=1}^N v_i}{N}$$

$$\textit{total weight: } W \equiv \sum_{i=1}^N w_i \quad \textit{normalized weight: } q_i \equiv \frac{w_i}{W} \quad \left( \sum_{i=1}^N q_i = 1 \right)$$

$$\textit{weighted average: } \frac{\sum_{i=1}^N w_i v_i}{\sum_{i=1}^N w_i} = \frac{\sum_{i=1}^N w_i v_i}{W} = \sum_{i=1}^N \frac{w_i}{W} v_i = \sum_{i=1}^N q_i v_i$$

# Typical Instruction Statistics

Instruction types, frequencies, and execution times

50% ALU instructions                      5 CPI

30% Memory instructions

20% Load                                  8 CPI

10% Store                                  6 CPI

20% Branch instructions                  10 CPI

0.5% Special instructions

# Average Cycles Per Instruction

(Weighted) average CPI

$$\begin{aligned} &= q_{\text{ALU}} T_{\text{ALU}} + q_{\text{Load}} T_{\text{Load}} + q_{\text{Store}} T_{\text{Store}} + q_{\text{Branch}} T_{\text{Branch}} \\ &= 0.5 \cdot 5 + 0.2 \cdot 8 + 0.1 \cdot 6 + 0.2 \cdot 10 \\ &= 2.5 + 1.6 + 0.6 + 2.0 \\ &= 6.7 \text{ cycles} \quad \text{approximation: } 20 / 6.7 \approx 3 \end{aligned}$$

Execution time fraction by instruction type

ALU	2.5 / 6.7	~ 37.5%
Load	1.6 / 6.7	~ 24.0%
Store	0.6 / 6.7	~ 9.0%
Branch	2.0 / 6.7	~ 30.0%

# CPU Time Equation

$$T_{CPU}(\text{execution}) = \frac{\# \text{ instructions}}{\text{execution}} \cdot \frac{\# \text{ cycles}}{\text{instruction}} \cdot \frac{\# \text{ seconds}}{\text{cycle}}$$

If  $T_{CPU}(\text{execution}) \approx 20$  seconds,  $\text{cycle}_{\text{time}} = 10^{-9}$  seconds

$$20 \text{ seconds} \approx \# \text{ instructions} \cdot 6.7 \cdot 10^{-9} \text{ seconds}$$

$$\# \text{ instructions} \approx \frac{20}{6.7 \cdot 10^{-9}} \approx 3 \cdot 10^9$$

$$\text{instruction}_{\text{time}} = \frac{\# \text{ seconds}}{\text{instruction}} = \frac{\# \text{ cycles}}{\text{instruction}} \cdot \frac{\# \text{ seconds}}{\text{cycle}}$$

# Performance Equations

Performance – inverse of execution time

$$\text{performance: } P_x \equiv \frac{1}{T_x} \quad \text{relative performance: } \frac{P_x}{P_y} = \frac{T_y}{T_x}$$

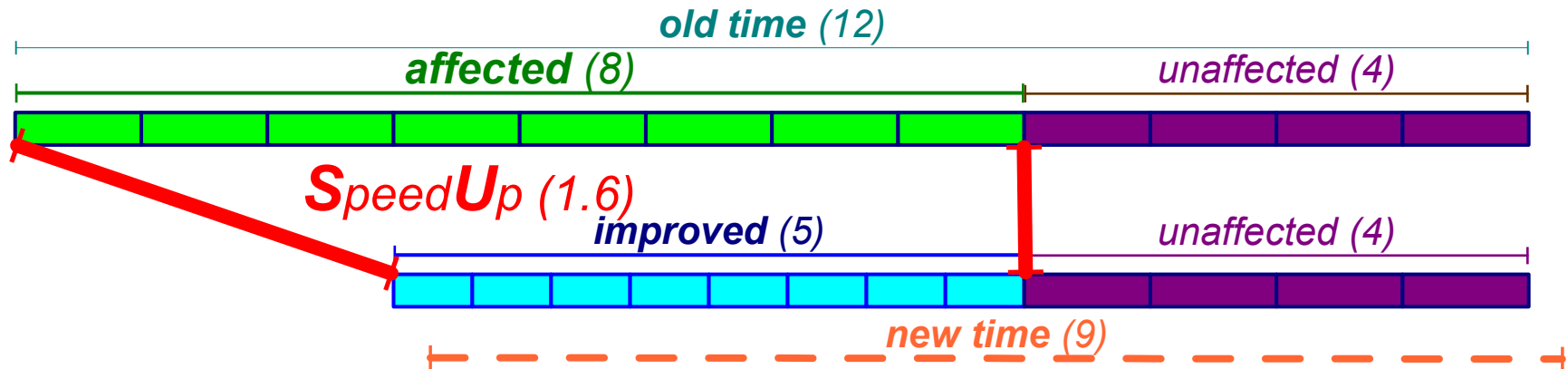
CPU time equation

$$T_{CPU}(\text{execution}) = \frac{\# \text{ instructions}}{\text{execution}} \cdot \frac{\# \text{ cycles}}{\text{instruction}} \cdot \frac{\# \text{ seconds}}{\text{cycle}}$$

## Amdahl's law

$$T_{new} = \frac{\text{fraction affected} \cdot T_{old}}{\text{improvement}} + \text{fraction not affected} \cdot T_{old}$$

# Amdahl's Law



$$T_{\text{old}} = \text{affected} + \text{unaffected}$$



$$T_{\text{new}} = \text{improved} + \text{unaffected}$$



$$\text{SpeedUp} = \text{affected} / \text{improved}$$



$$\text{Overall SpeedUp} = P_{\text{new}} / P_{\text{old}} = T_{\text{old}} / T_{\text{new}}$$



$$(\text{fraction affected}) \mathbf{F}_a = \text{affected} / T_{\text{old}}$$



$$(\text{fraction unaffected}) \mathbf{\bar{F}}_a = \text{unaffected} / T_{\text{old}}$$



# Improving *Race Car* Performance

	miles	miles/hours
cruising	900	90
other	100	50

Race time =  $900/90 + 100/50 = 12$  hours

Change #1: 1.11 x improvement in cruising speed

Change #2: 2.00 x improvement in other speed



# Change # 1

$$\mathbf{T}_{\text{old}} = 12 \text{ hours}$$

$$\mathbf{fa} = 0.90 \quad (\text{fraction affected} = \text{affected/total} = \frac{900 \text{ miles}}{1000 \text{ miles}})$$

$$\overline{\mathbf{fa}} = 0.10 \quad (\text{fraction unaffected} = \text{unaffected/total} = \frac{100 \text{ miles}}{1000 \text{ miles}})$$

$$\mathbf{su} = 1.11 \quad (\text{speedup for affected})$$

$$\begin{aligned} \mathbf{T}_{\text{new}} &= \frac{\mathbf{fa} \cdot \mathbf{T}_{\text{old}}}{\mathbf{su}} + \overline{\mathbf{fa}} \cdot \mathbf{T}_{\text{old}} \\ &= \frac{0.9 \cdot 12}{1.11} + 0.1 \cdot 12 \approx 9.73 + 1.2 = \mathbf{10.93} \text{ hours} \end{aligned}$$

## Change # 2

$$T_{\text{old}} = 12 \text{ hours}$$

$$\mathbf{fa} = 0.10 \quad (\text{fraction affected} = \text{affected/total} = \frac{100 \text{ miles}}{1000 \text{ miles}})$$

$$\overline{\mathbf{fa}} = 0.90 \quad (\text{fraction unaffected} = \text{unaffected/total} = \frac{900 \text{ miles}}{1000 \text{ miles}})$$

$$\mathbf{su} = 2.00 \quad (\text{speedup for affected})$$

$$\begin{aligned} T_{\text{new}} &= \frac{\mathbf{fa} \cdot T_{\text{old}}}{\mathbf{su}} + \overline{\mathbf{fa}} \cdot T_{\text{old}} \\ &= \frac{0.1 \cdot 12}{2} + 0.9 \cdot 12 \approx 0.6 + 10.8 = \mathbf{11.4 \text{ hours}} \end{aligned}$$

# ~~Change # 1~~ **wrong!**

$$T_{\text{old}} = 12 \text{ hours}$$

$$fa = 0.90 \quad (\text{fraction affected} = \text{affected/total} = \frac{900 \text{ miles}}{1000 \text{ miles}})$$

$$\overline{fa} = 0.10 \quad (\text{fraction unaffected} = \text{unaffected/total} = \frac{100 \text{ miles}}{1000 \text{ miles}})$$

$$su = 1.11 \quad (\text{speedup for affected})$$

$$T_{\text{new}} = \frac{fa \cdot T_{\text{old}}}{su} + \overline{fa} \cdot T_{\text{old}}$$

$$= \frac{0.9 \cdot 12}{1.11} + 0.1 \cdot 12 \approx 9.73 + 1.2 = 10.93 \text{ hours}$$

# ~~Change # 2~~ wrong!

$$T_{\text{old}} = 12 \text{ hours}$$

$$fa = 0.10 \quad (\text{fraction affected} = \text{affected/total} = \frac{100 \text{ miles}}{1000 \text{ miles}})$$

$$\overline{fa} = 0.90 \quad (\text{fraction unaffected} = \text{unaffected/total} = \frac{900 \text{ miles}}{1000 \text{ miles}})$$

$$su = 2.00 \quad (\text{speedup for affected})$$

$$T_{\text{new}} = \frac{fa \cdot T_{\text{old}}}{su} + \overline{fa} \cdot T_{\text{old}}$$

$$= \frac{0.1 \cdot 12}{2} + 0.9 \cdot 12 \approx 0.6 + 10.8 = 11.4 \text{ hours}$$

# Change # 1

correct

$$T_{\text{old}} = 12 \text{ hours}$$

$$\mathbf{fa} = 0.833 \quad (\text{fraction affected} = \text{affected/total} = \frac{10 \text{ hours}}{12 \text{ hours}} = \frac{5}{6})$$

$$\overline{\mathbf{fa}} = 0.167 \quad (\text{fraction unaffected} = \text{unaffected/total} = \frac{2 \text{ hours}}{12 \text{ hours}} = \frac{1}{6})$$

$$\mathbf{su} = 1.11 \quad (\text{speedup for affected})$$

$$T_{\text{new}} = \frac{\mathbf{fa} \cdot T_{\text{old}}}{\mathbf{su}} + \overline{\mathbf{fa}} \cdot T_{\text{old}}$$

$$\approx \frac{0.833 \cdot 12}{1.11} + 0.167 \cdot 12 \approx 9 + 2 = 11 \text{ hours}$$

# Change # 2

## correct

$$T_{\text{old}} = 12 \text{ hours}$$

$$\begin{aligned} \mathbf{fa} &= 0.167 \quad (\text{fraction affected} = \text{affected/total} = \frac{2 \text{ hours}}{12 \text{ hours}} = \frac{1}{6}) \\ \overline{\mathbf{fa}} &= 0.833 \quad (\text{fraction unaffected} = \text{unaffected/total} = \frac{10 \text{ hours}}{12 \text{ hours}} = \frac{5}{6}) \end{aligned}$$

$$\mathbf{su} = 2.000 \quad (\text{speedup for affected})$$

$$T_{\text{new}} = \frac{\mathbf{fa} \cdot T_{\text{old}}}{\mathbf{su}} + \overline{\mathbf{fa}} \cdot T_{\text{old}}$$

$$\approx \frac{0.167 \cdot 12}{2} + 0.833 \cdot 12 \approx 1. + 10 = 11 \text{ hours}$$

# Average Cycles Per Instruction

(Weighted) average CPI

$$\begin{aligned} &= q_{\text{ALU}} T_{\text{ALU}} + q_{\text{Load}} T_{\text{Load}} + q_{\text{Store}} T_{\text{Store}} + q_{\text{Branch}} T_{\text{Branch}} \\ &= 0.5 \cdot 5 + 0.2 \cdot 8 + 0.1 \cdot 6 + 0.2 \cdot 10 \\ &= 2.5 + 1.6 + 0.6 + 2.0 \\ &= 6.7 \text{ cycles} \quad \text{approximation: } 20 / 6.7 \approx 3 \end{aligned}$$

Execution time fraction by instruction type

ALU	2.5 / 6.7	~ 37.5%
Load	1.6 / 6.7	~ 24.0%
Store	0.6 / 6.7	~ 9.0%
Branch	2.0 / 6.7	~ 30.0%

# CPU Time Equation

$$T_{CPU}(\text{execution}) = \frac{\# \text{ instructions}}{\text{execution}} \cdot \frac{\# \text{ cycles}}{\text{instruction}} \cdot \frac{\# \text{ seconds}}{\text{cycle}}$$

If  $T_{CPU}(\text{execution}) \approx 20$  seconds,  $\text{cycle}_{\text{time}} = 10^{-9}$  seconds

$$20 \text{ seconds} \approx \# \text{ instructions} \cdot 6.7 \cdot 10^{-9} \text{ seconds}$$

$$\# \text{ instructions} \approx \frac{20}{6.7 \cdot 10^{-9}} \approx 3 \cdot 10^9$$

$$\text{instruction}_{\text{time}} = \frac{\# \text{ seconds}}{\text{instruction}} = \frac{\# \text{ cycles}}{\text{instruction}} \cdot \frac{\# \text{ seconds}}{\text{cycle}}$$



# Amdahl's Law 1

$$T_{new} = \frac{\text{fraction affected} \cdot T_{old}}{\text{improvement}} + \text{fraction not affected} \cdot T_{old}$$

Improvement  $X$

reduces ALU instruction CPI from 5 to 4

$$T_X = \frac{\text{fraction affected} \cdot 20 \text{ sec}}{\text{improvement}} + \text{fraction not affected} \cdot 20 \text{ sec}$$

# Amdahl's Law 1 (wrong!)

$$T_{new} = \frac{\text{fraction affected} \cdot T_{old}}{\text{improvement}} + \text{fraction not affected} \cdot T_{old}$$

Improvement  $X$

reduces ALU instruction CPI from 5 to 4

$$T_X = \frac{\text{fraction affected} \cdot 20 \text{ sec}}{\text{improvement}} + \text{fraction not affected} \cdot 20 \text{ sec}$$

$$= \frac{0.5 \cdot 20}{\frac{5}{4}} + 0.5 \cdot 20 \text{ sec} = 8 + 10 \text{ sec} = 18 \text{ sec}$$

# Amdahl's Law 1

$$T_{new} = \frac{\text{fraction affected} \cdot T_{old}}{\text{improvement}} + \text{fraction not affected} \cdot T_{old}$$

Improvement  $X$

reduces ALU instruction CPI from 5 to 4

$$T_X = \frac{\text{fraction affected} \cdot 20 \text{ sec}}{\text{improvement}} + \text{fraction not affected} \cdot 20 \text{ sec}$$

$$= \left( \frac{\frac{2.5}{6.7} 20}{\frac{5}{4}} + \frac{4.2}{6.7} 20 \right) \text{ sec} \approx \left( \frac{7.5}{1.25} + 12.6 \right) \text{ sec} = 18.6 \text{ sec}$$

# Amdahl's Law 2

$$T_{new} = \frac{\text{fraction affected} \cdot T_{old}}{\text{improvement}} + \text{fraction not affected} \cdot T_{old}$$

Improvement  $Y$

reduces Load instruction CPI from 8 to 4

$$T_Y = \frac{\text{fraction affected} \cdot 20 \text{ sec}}{\text{improvement}} + \text{fraction not affected} \cdot 20 \text{ sec}$$

$$= \left( \frac{\frac{1.6}{6.7} 20}{\frac{8}{4}} + \frac{5.1}{6.7} 20 \right) \text{sec} \approx \left( \frac{4.8}{2} + 15.3 \right) \text{sec} = 17.7 \text{ sec}$$

# Amdahl's Law 3

$$T_{new} = \frac{\text{fraction affected} \cdot T_{old}}{\text{improvement}} + \text{fraction not affected} \cdot T_{old}$$

Improvement  $Z$

reduces Store instruction CPI from 6 to 2

$$T_Z = \frac{\text{fraction affected} \cdot 20 \text{ sec}}{\text{improvement}} + \text{fraction not affected} \cdot 20 \text{ sec}$$

$$= \left( \frac{\frac{0.6}{6.7} 20}{\frac{6}{2}} + \frac{6.1}{6.7} 20 \right) \text{sec} \approx \left( \frac{1.8}{3} + 18.3 \right) \text{sec} = 18.9 \text{ sec}$$

# Amdahl's Law 4

$$T_{new} = \frac{\text{fraction affected} \cdot T_{old}}{\text{improvement}} + \text{fraction not affected} \cdot T_{old}$$

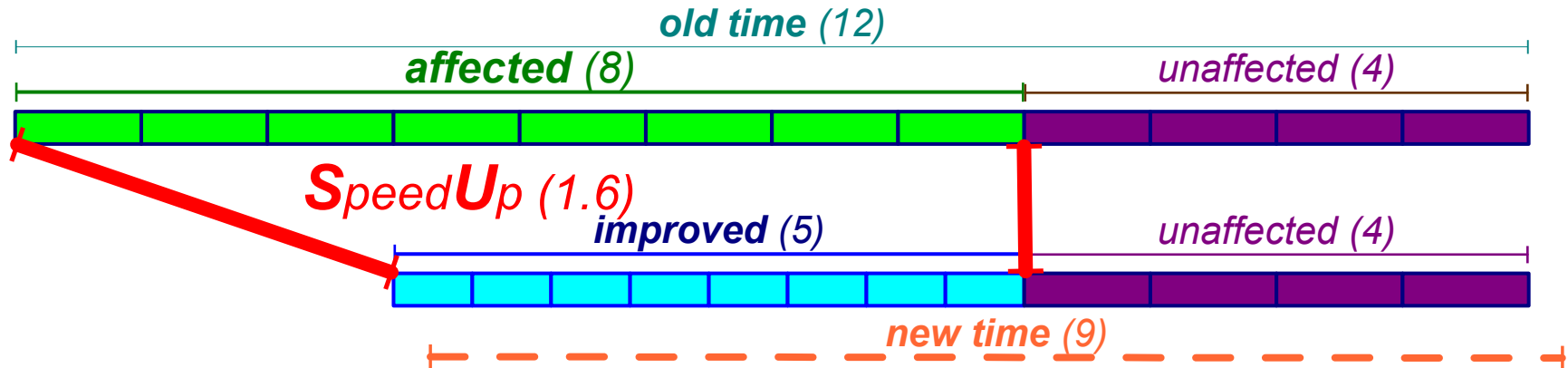
Improvement  $W$

reduces Branch instruction CPI from 10 to 5

$$T_w = \frac{\text{fraction affected} \cdot 20 \text{ sec}}{\text{improvement}} + \text{fraction not affected} \cdot 20 \text{ sec}$$

$$= \left( \frac{\frac{2.0}{6.7} 20}{\frac{10}{5}} + \frac{4.7}{6.7} 20 \right) \text{sec} \approx \left( \frac{6}{2} + 14.1 \right) \text{sec} = 17.1 \text{ sec}$$

# Amdahl's Law



$$T_{\text{old}} = \text{affected} + \text{unaffected}$$



$$T_{\text{new}} = \text{improved} + \text{unaffected}$$



$$\text{SpeedUp} = \text{affected} / \text{improved}$$



$$\text{Overall SpeedUp} = P_{\text{new}} / P_{\text{old}} = T_{\text{old}} / T_{\text{new}}$$



$$(\text{fraction affected}) \mathbf{F}_a = \text{affected} / T_{\text{old}}$$



$$(\text{fraction unaffected}) \mathbf{\bar{F}}_a = \text{unaffected} / T_{\text{old}}$$



# Amdahl's Law Overall SpeedUp

$$\text{performance: } P_x \equiv \frac{1}{T_x} \quad \text{relative performance: } \frac{P_x}{P_y} = \frac{T_y}{T_x}$$

$$\frac{P_X}{P_{old}} = \frac{T_{old}}{T_X} = \frac{20}{18.6} \approx 1.075$$

$$\frac{P_Y}{P_{old}} = \frac{T_{old}}{T_Y} = \frac{20}{17.7} \approx 1.130$$

$$\frac{P_Z}{P_{old}} = \frac{T_{old}}{T_Z} = \frac{20}{18.9} \approx 1.058$$

$$\frac{P_W}{P_{old}} = \frac{T_{old}}{T_W} = \frac{20}{17.1} \approx 1.170$$



# Amdahl's Law Overall SpeedUp

$$\begin{aligned}\frac{P_{\text{new}}}{P_{\text{old}}} &= \frac{T_{\text{old}}}{T_{\text{new}}} \\ &= \frac{T_{\text{old}}}{\frac{\text{fa} \cdot T_{\text{old}}}{\text{su}} + \overline{\text{fa}} \cdot T_{\text{old}}} \\ &= \boxed{\frac{1}{\frac{\text{fa}}{\text{su}} + \overline{\text{fa}}}}\end{aligned}$$

# Human Addition (Binary)

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

# Binary Addition

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

# Binary Addition

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

# Binary Addition

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

# Binary Addition

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

# Binary Addition

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

# Binary Addition

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



# Binary Addition

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

# Binary Addition

$$\begin{array}{r} \phantom{+} 1101110 \\ + 1000010 \\ \hline 1001110 \\ 1011000 \end{array}$$

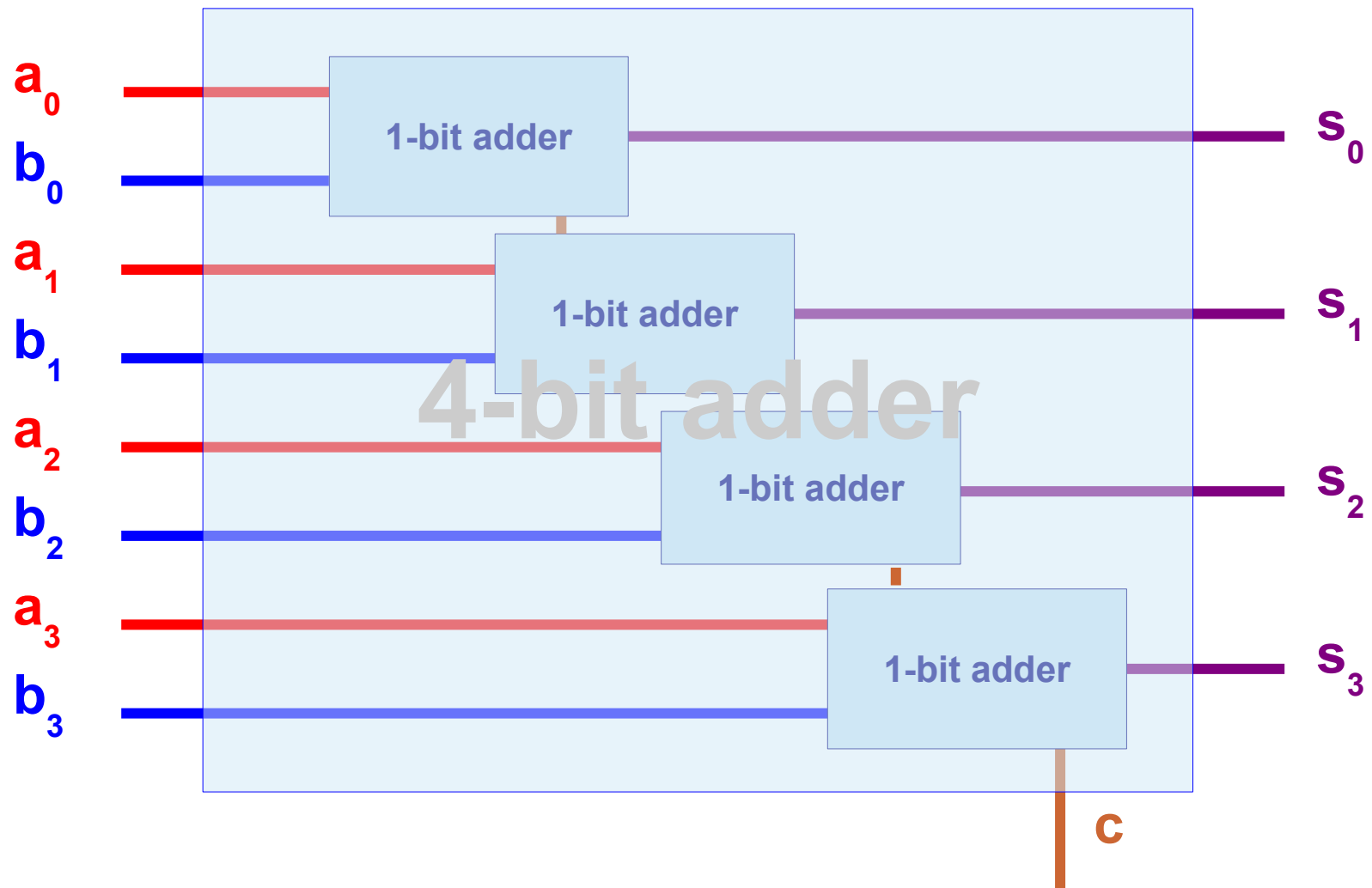
# 64-Bit Computer Addition



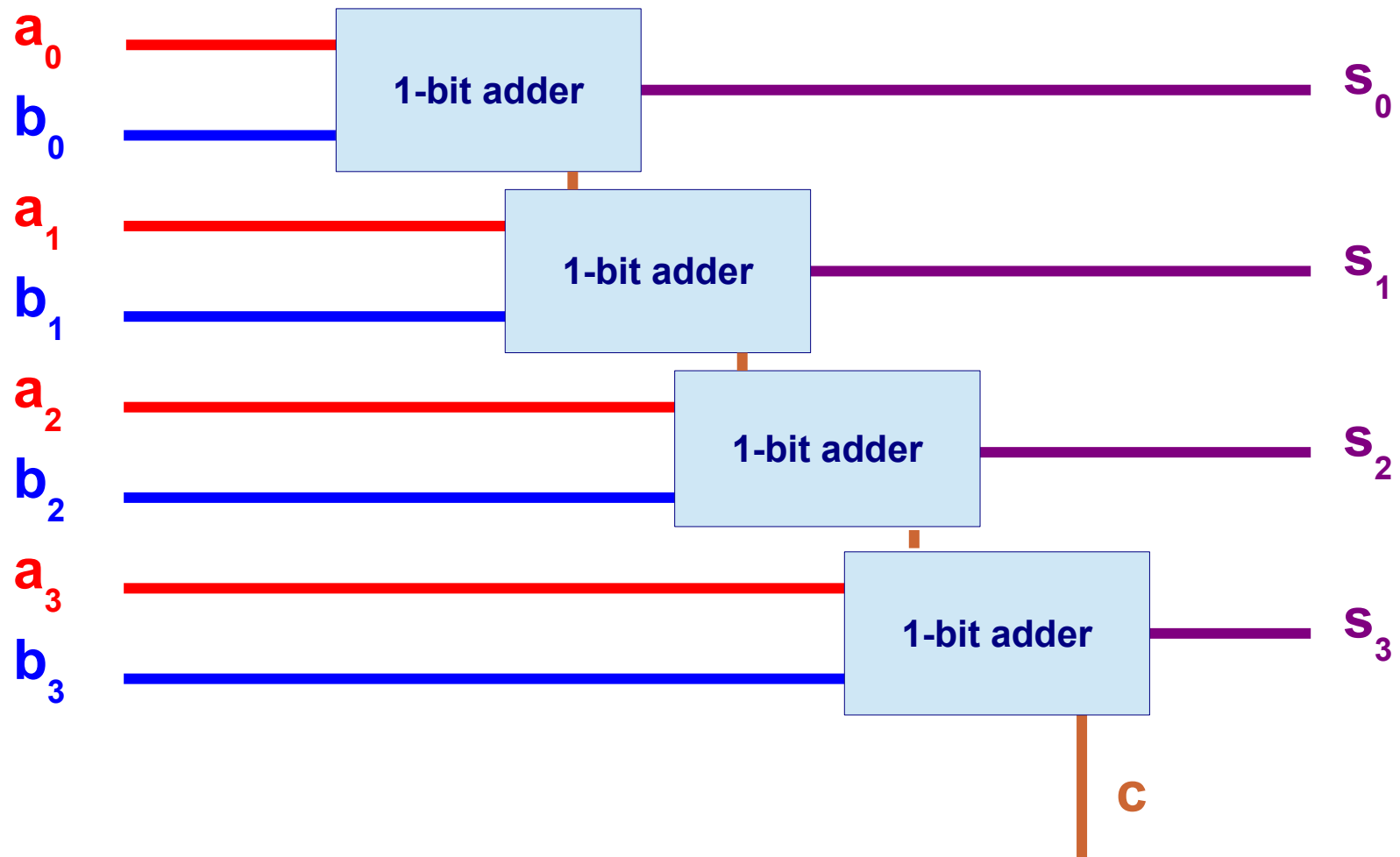
# 4-Bit Computer Addition



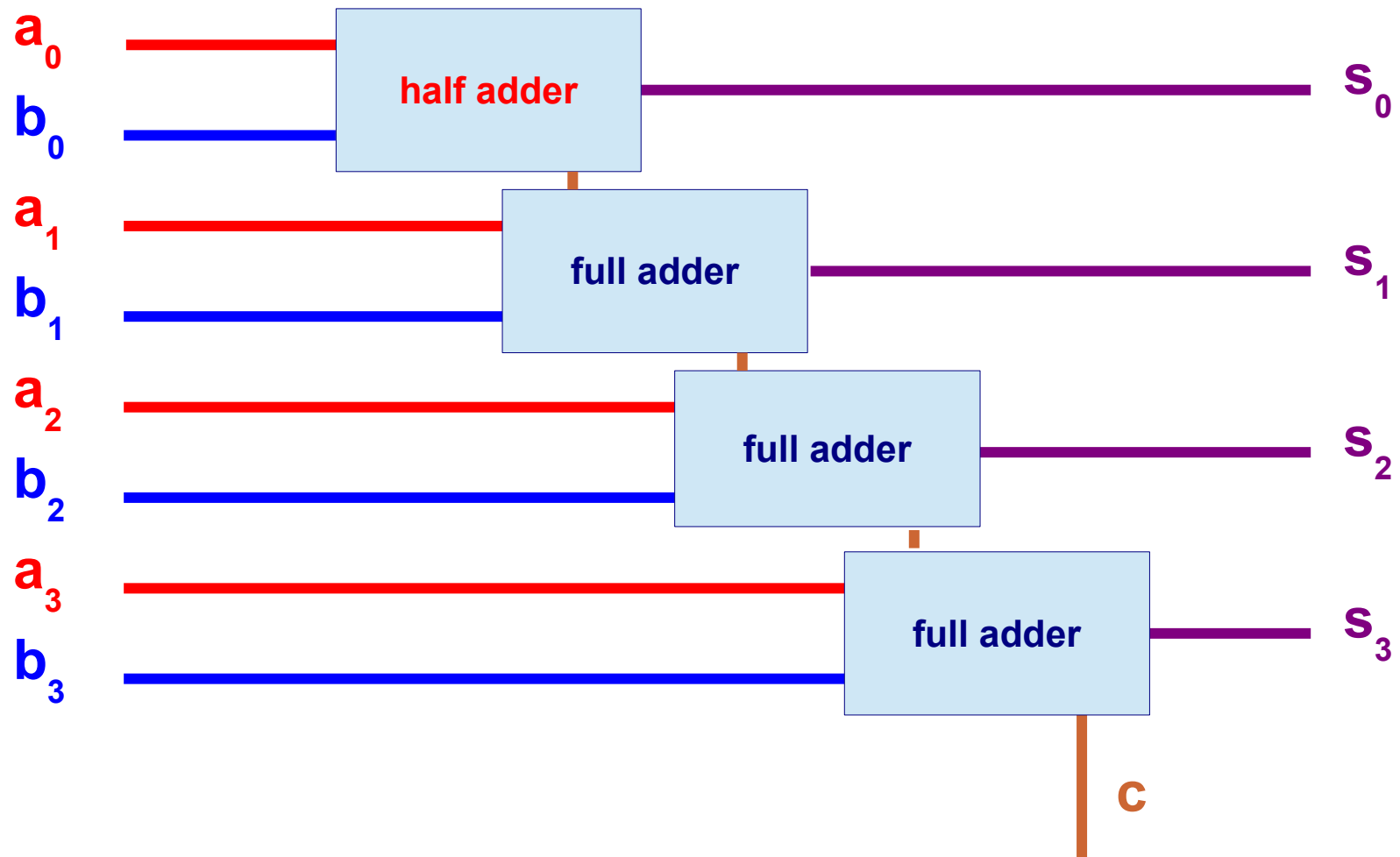
# 4-Bit Ripple Carry Adder



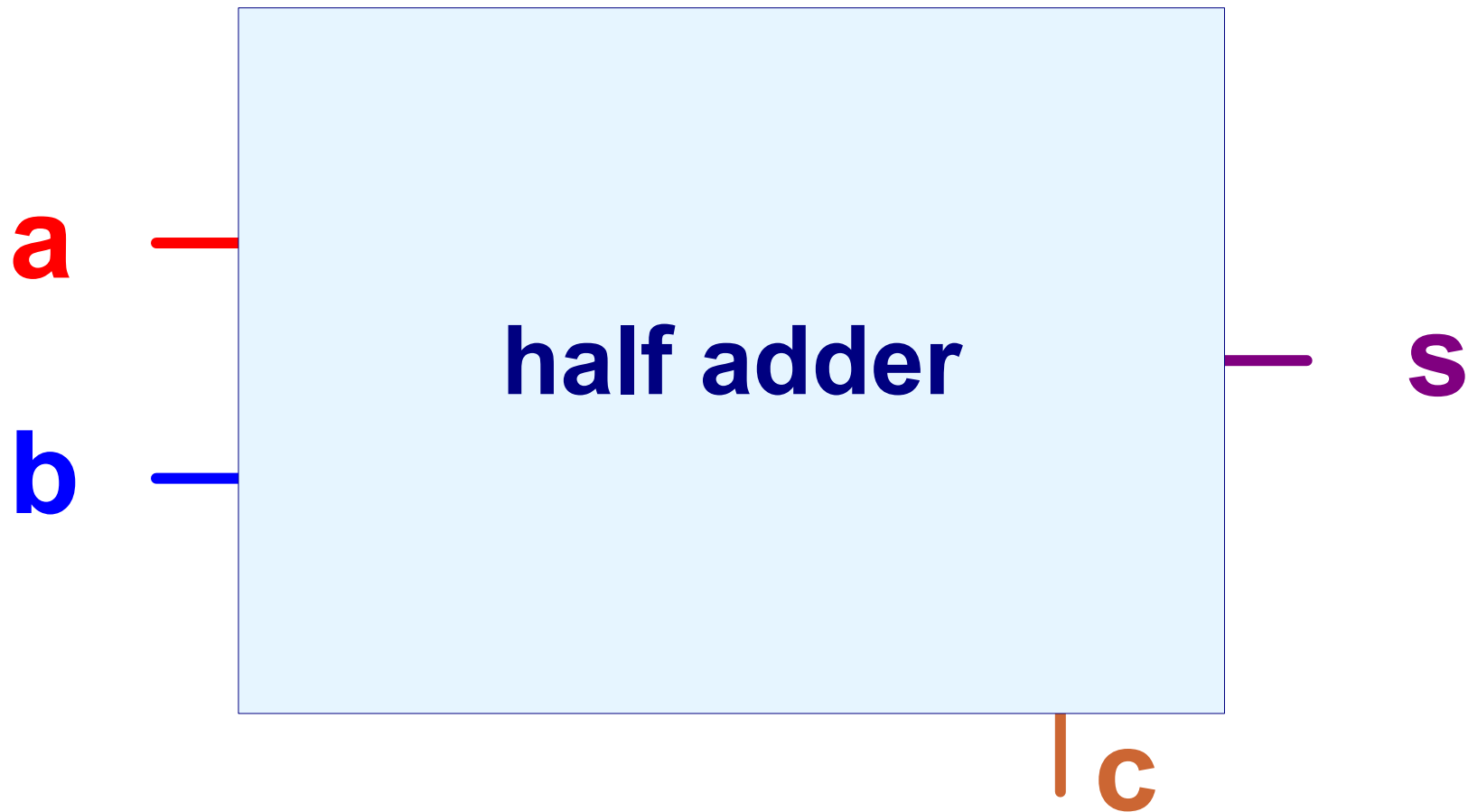
# 4-Bit Ripple Carry Adder



# 4-Bit Ripple Carry Adder



# 1-Bit Computer Addition (take 1)



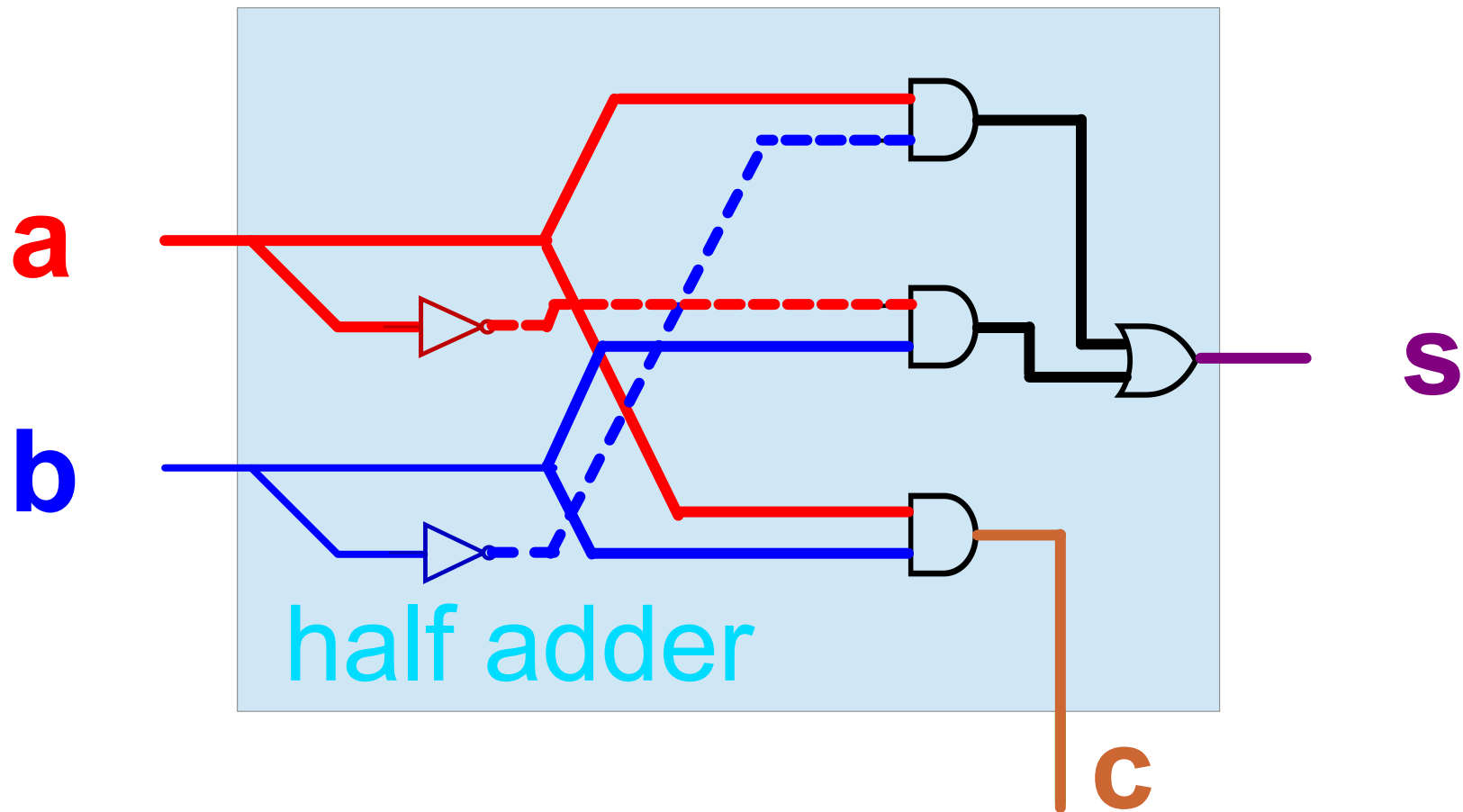


# Half Adder Truth Table

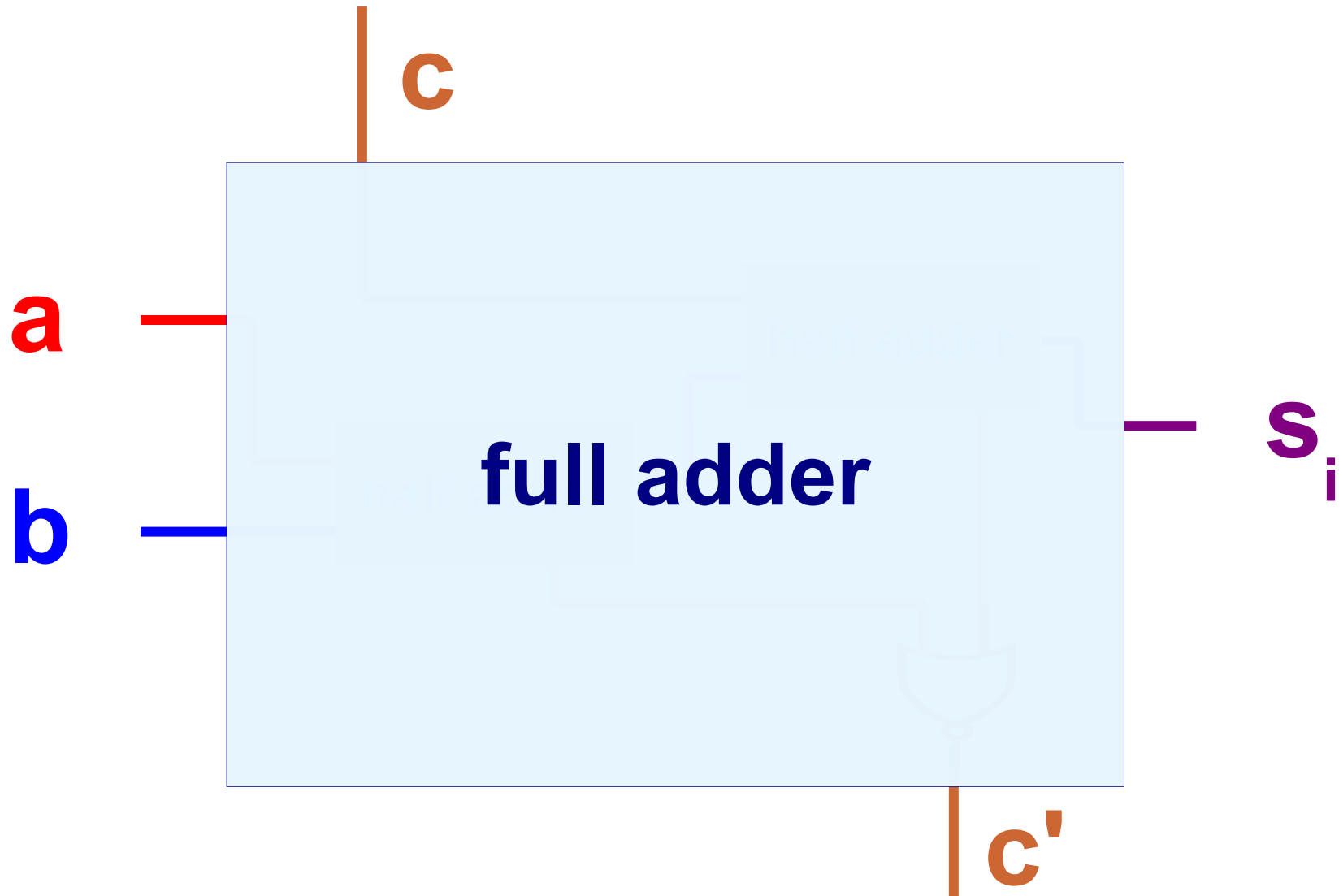
a	b	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

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

# *Half* Adder Circuit



# 1-Bit Computer Addition (take 2)



# Full Adder

#	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

#	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

#	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

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

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

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

# Full Adder

#	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

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

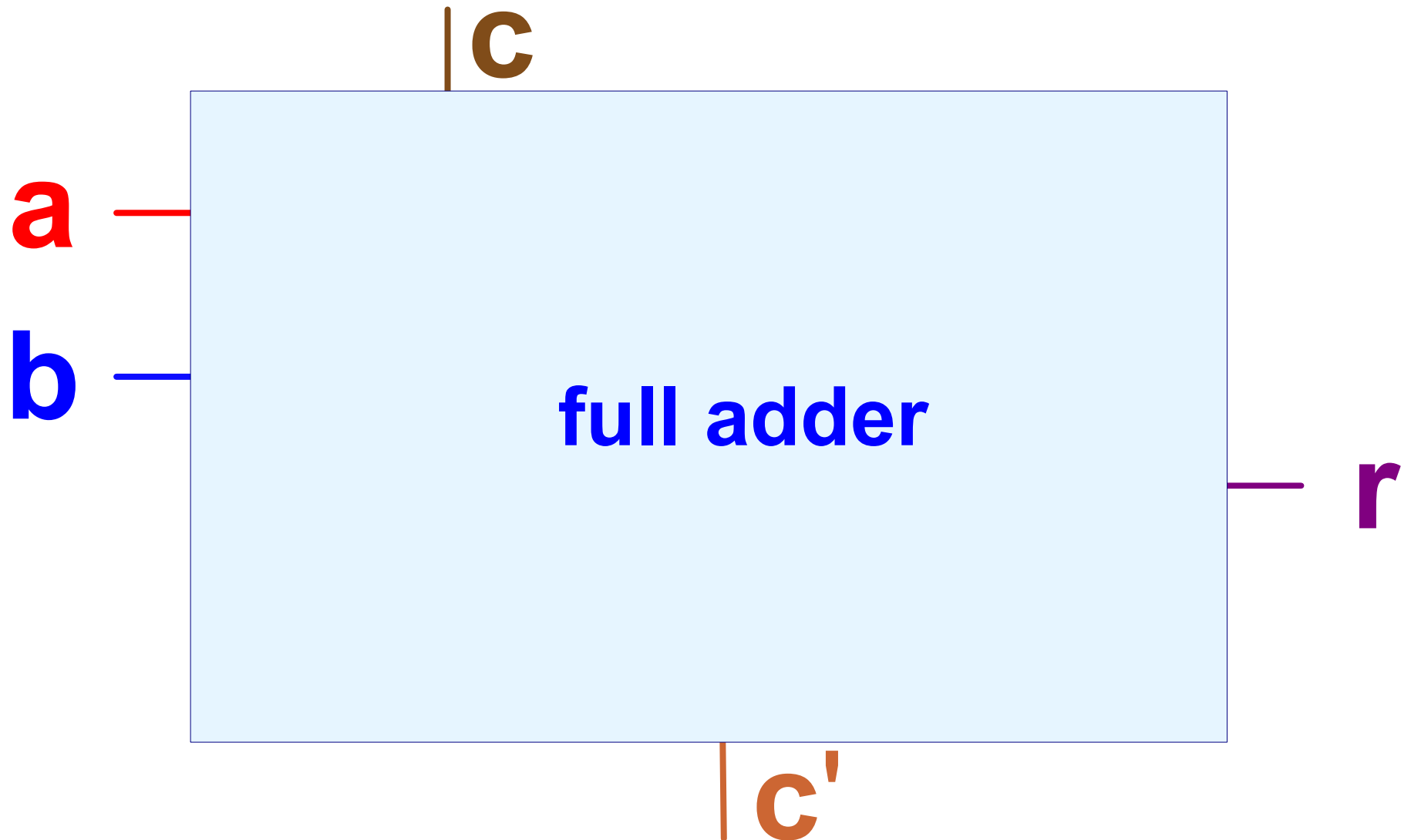
$$c' = \overline{a}bc + a\overline{b}c + ab\overline{c} + abc$$

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

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

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

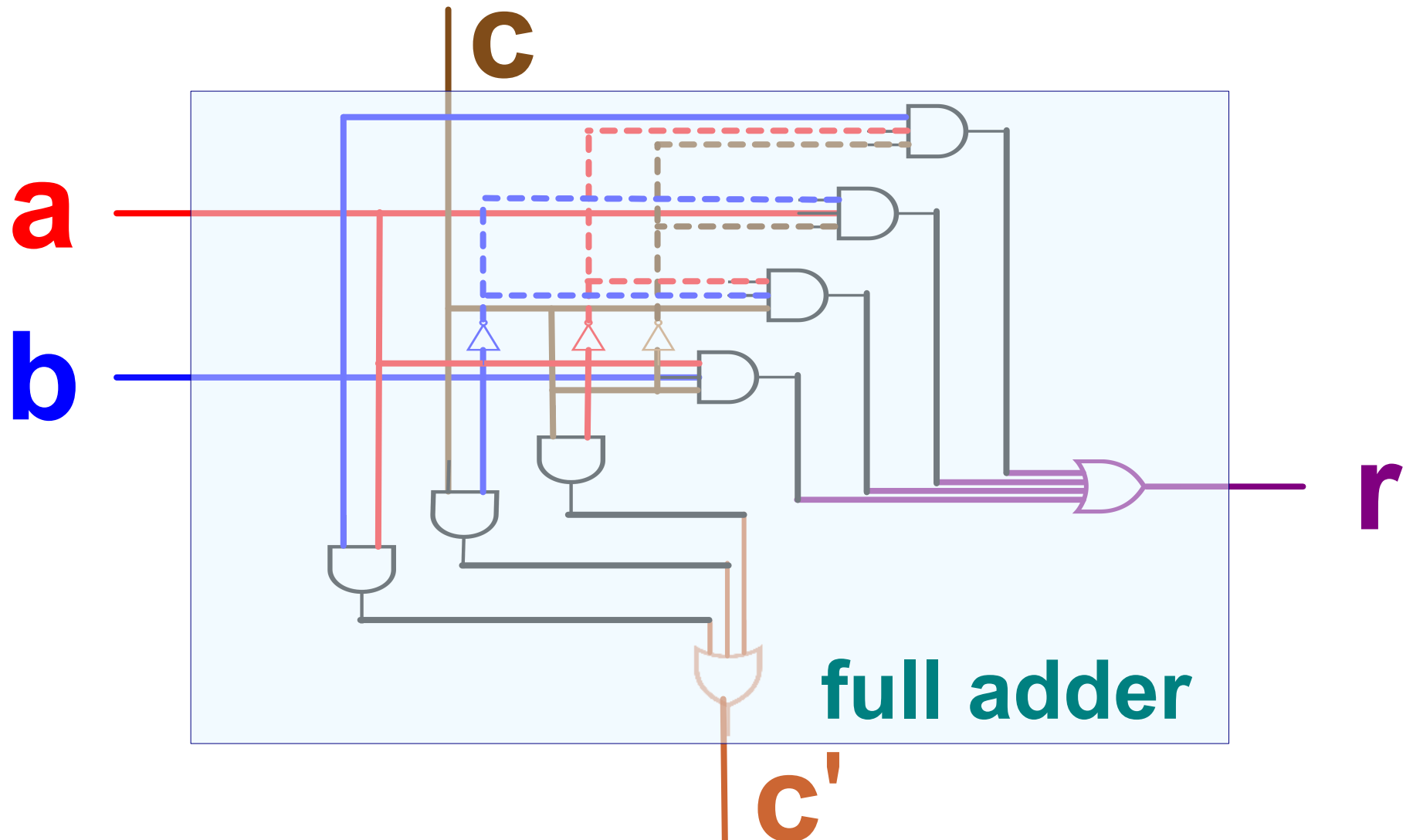
$$c' = ab + ac + bc$$



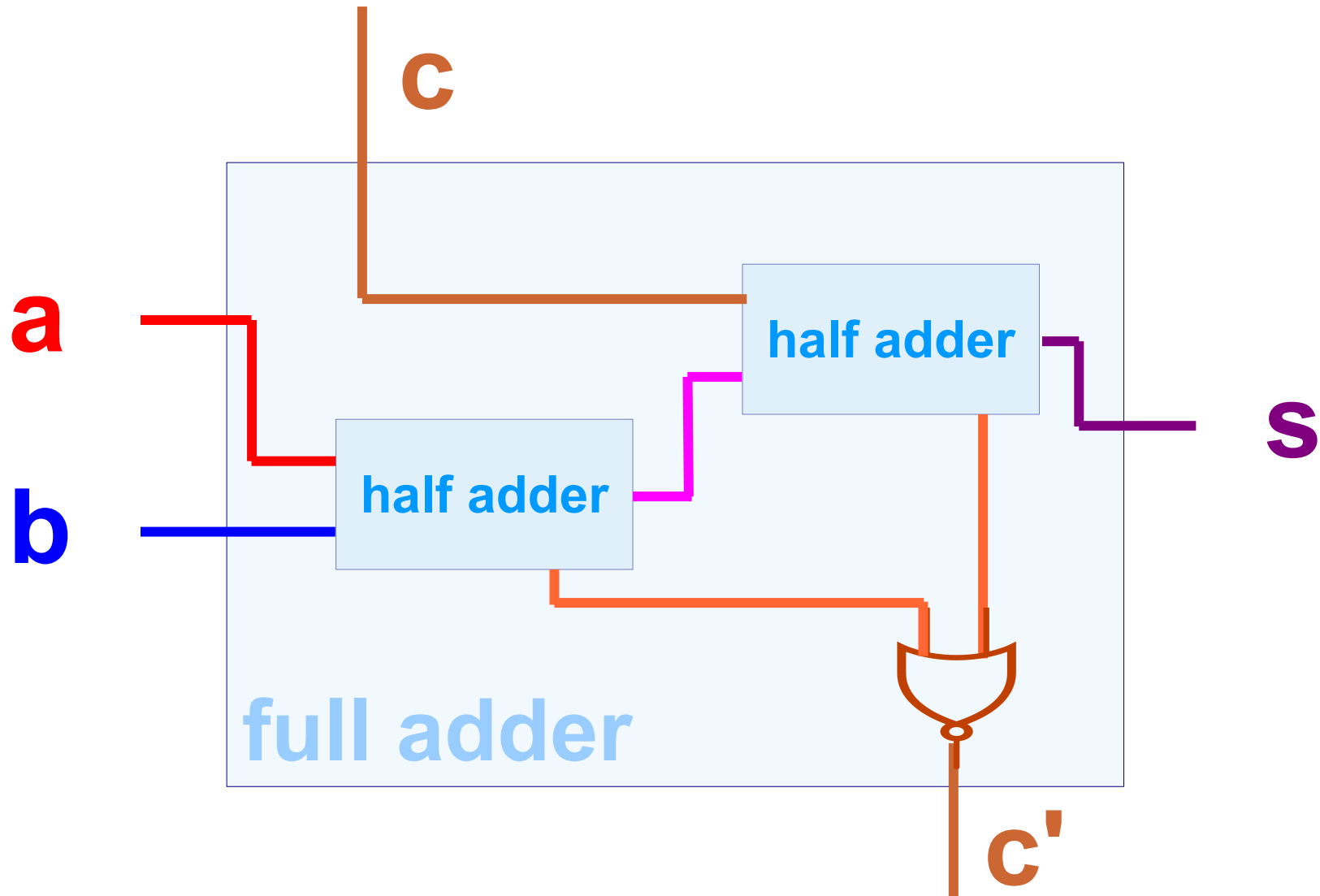


$$s = abc + \overline{a}bc + \overline{a}\overline{b}\overline{c} + a\overline{b}c$$

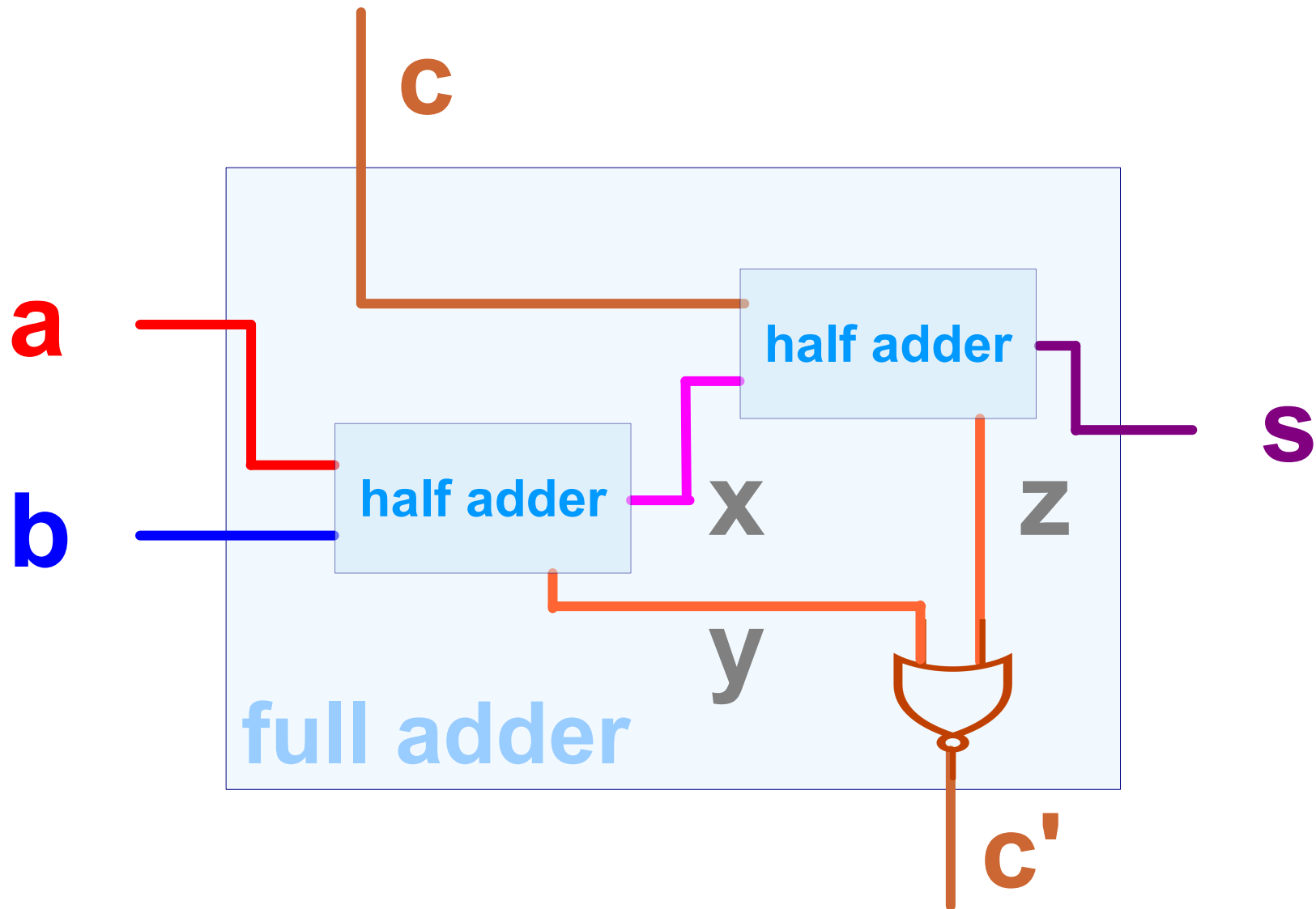
$$c' = ab + ac + bc$$



# *Full* Adder ??

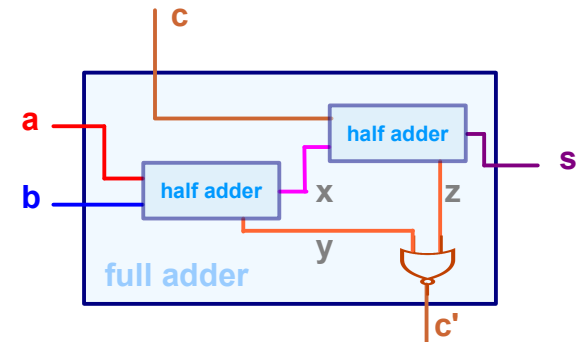


# *Full* Adder ??



# Full Adder Implementation ??

#	a	b	c	x	y	z	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



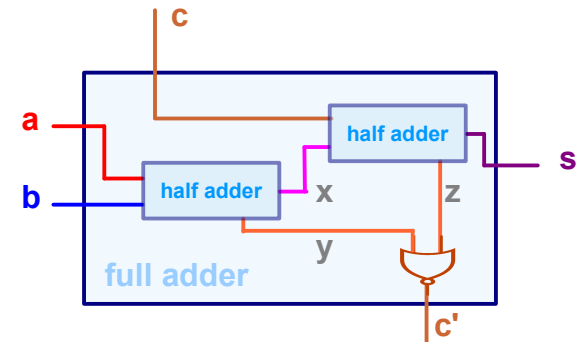
u	v	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$c = uv$$

$$s = \overline{u}v + u\overline{v}$$

# Full Adder Implementation!

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

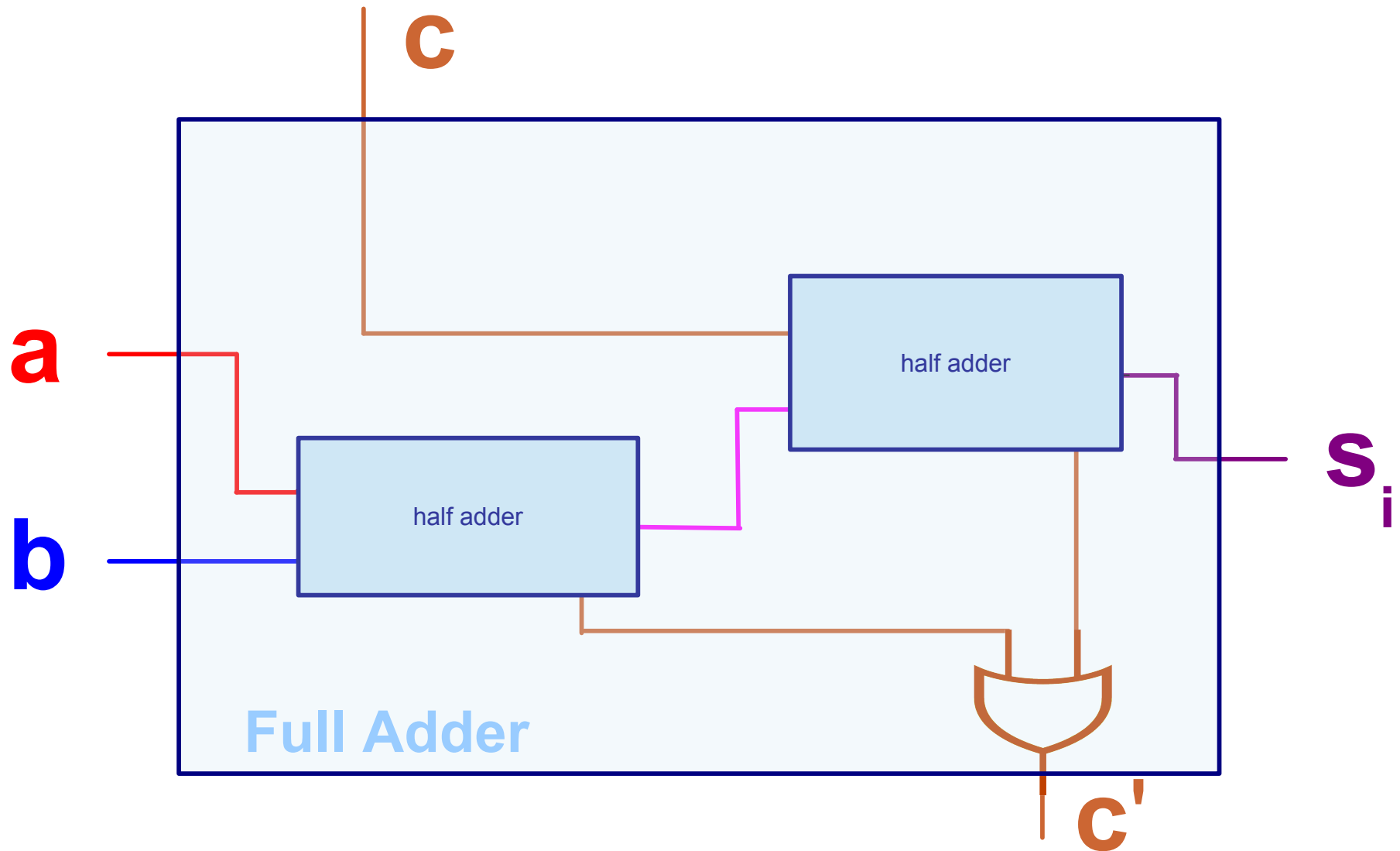


u	v	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

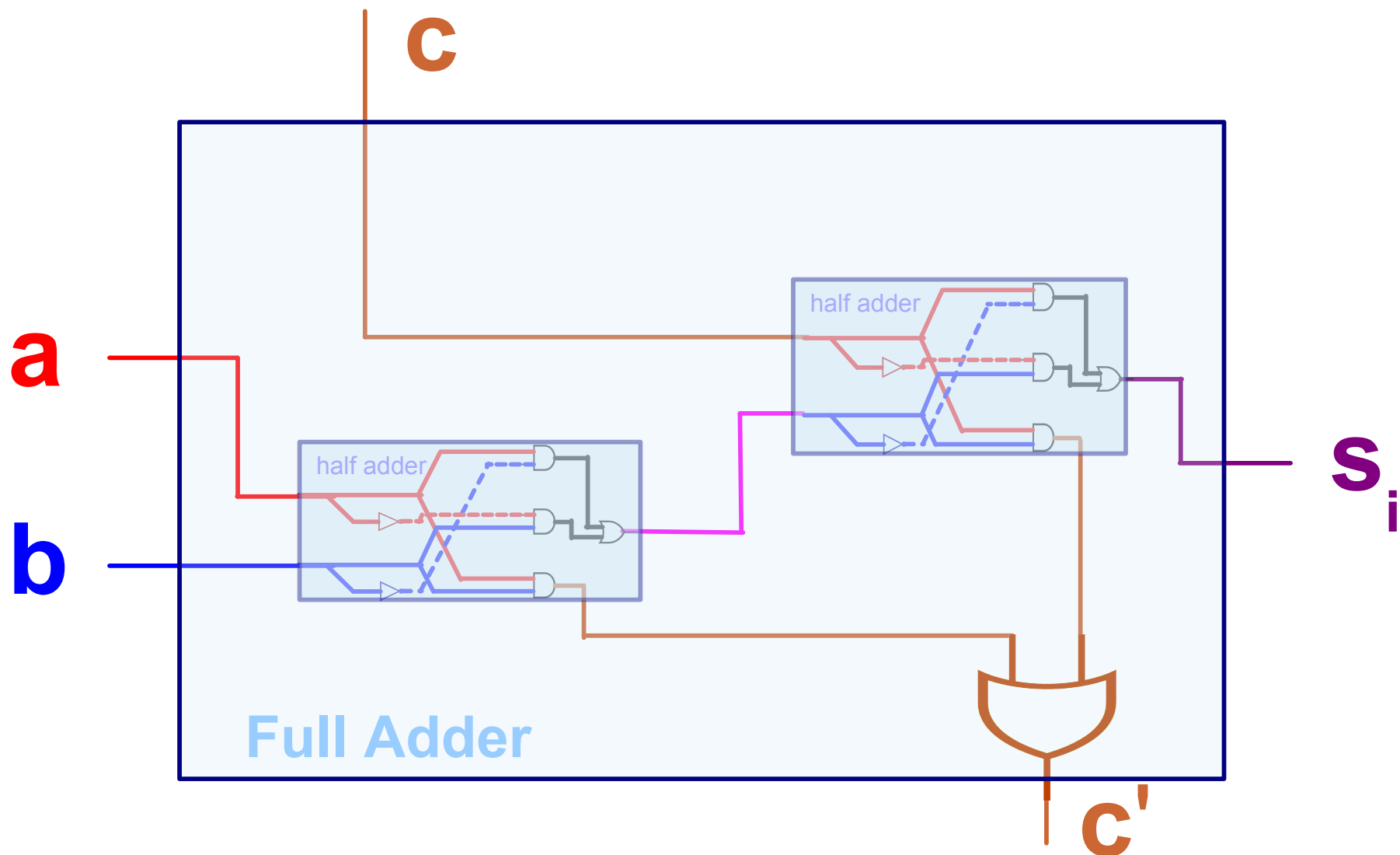
$$c = uv$$

$$s = \overline{u}v + u\overline{v}$$

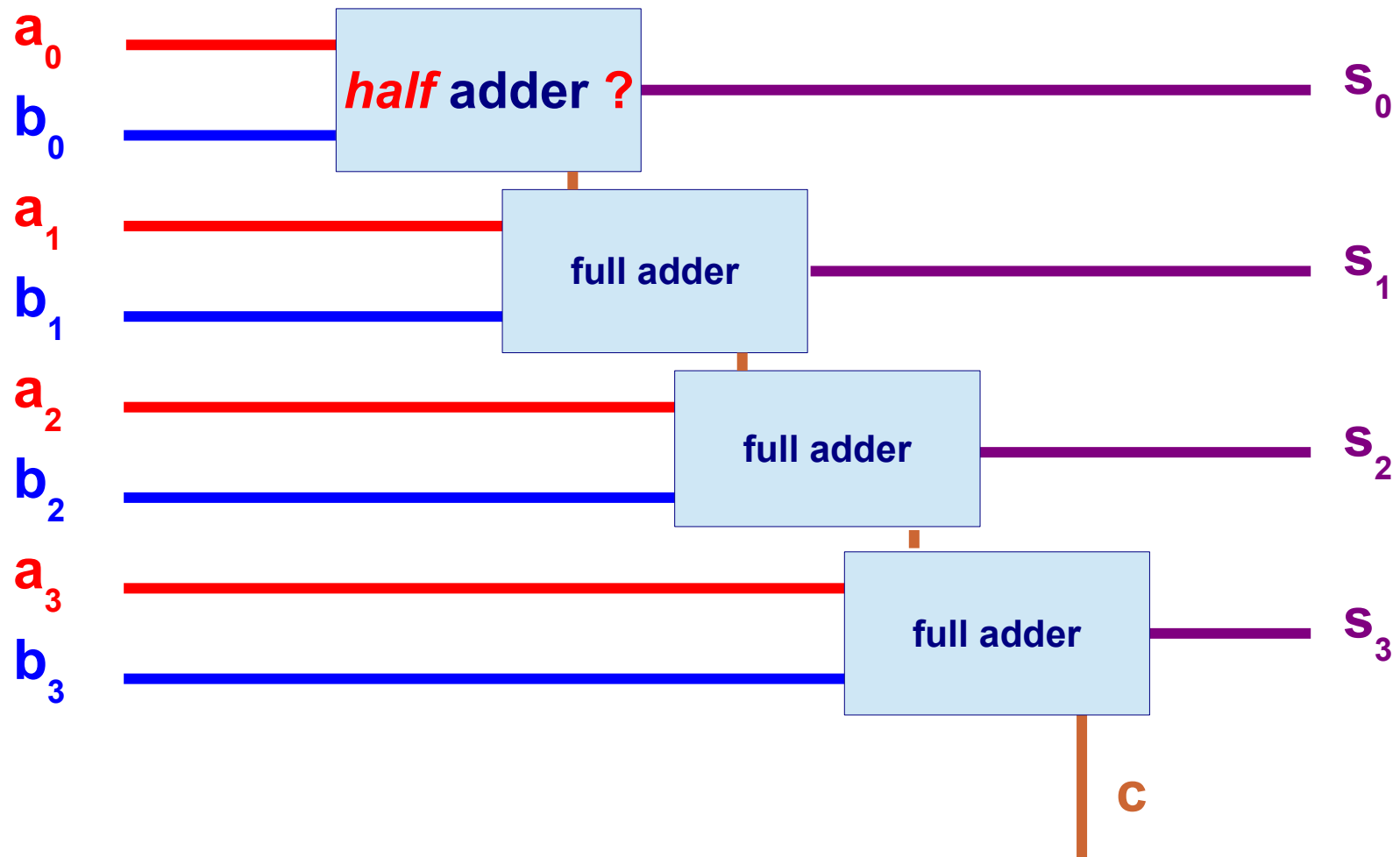
# *A Full Adder*



# *A Full Adder*



# Initial adder — half or full?





# *Full* Initial adder?

## Con

Superfluous wires and gates

## Pro

General simplicity

Avoid special cases where ever practical

Simplifies addition of big integers

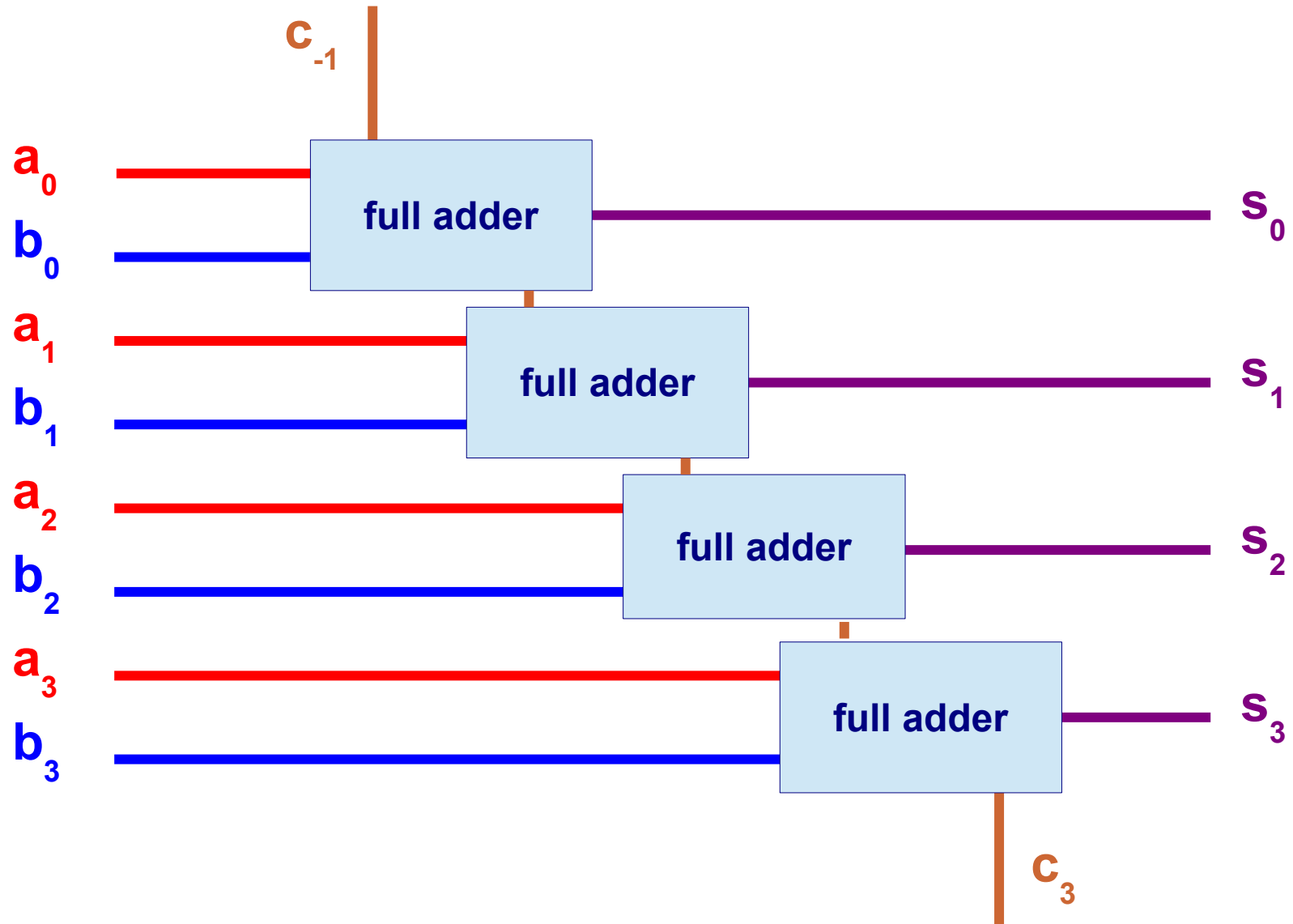
Big Integer:  $N = x_k \cdot J^k + \dots + x_2 \cdot J^2 + x_1 \cdot J^1 + x_0 \cdot J^0$

Where  $J \equiv 2^{32}$

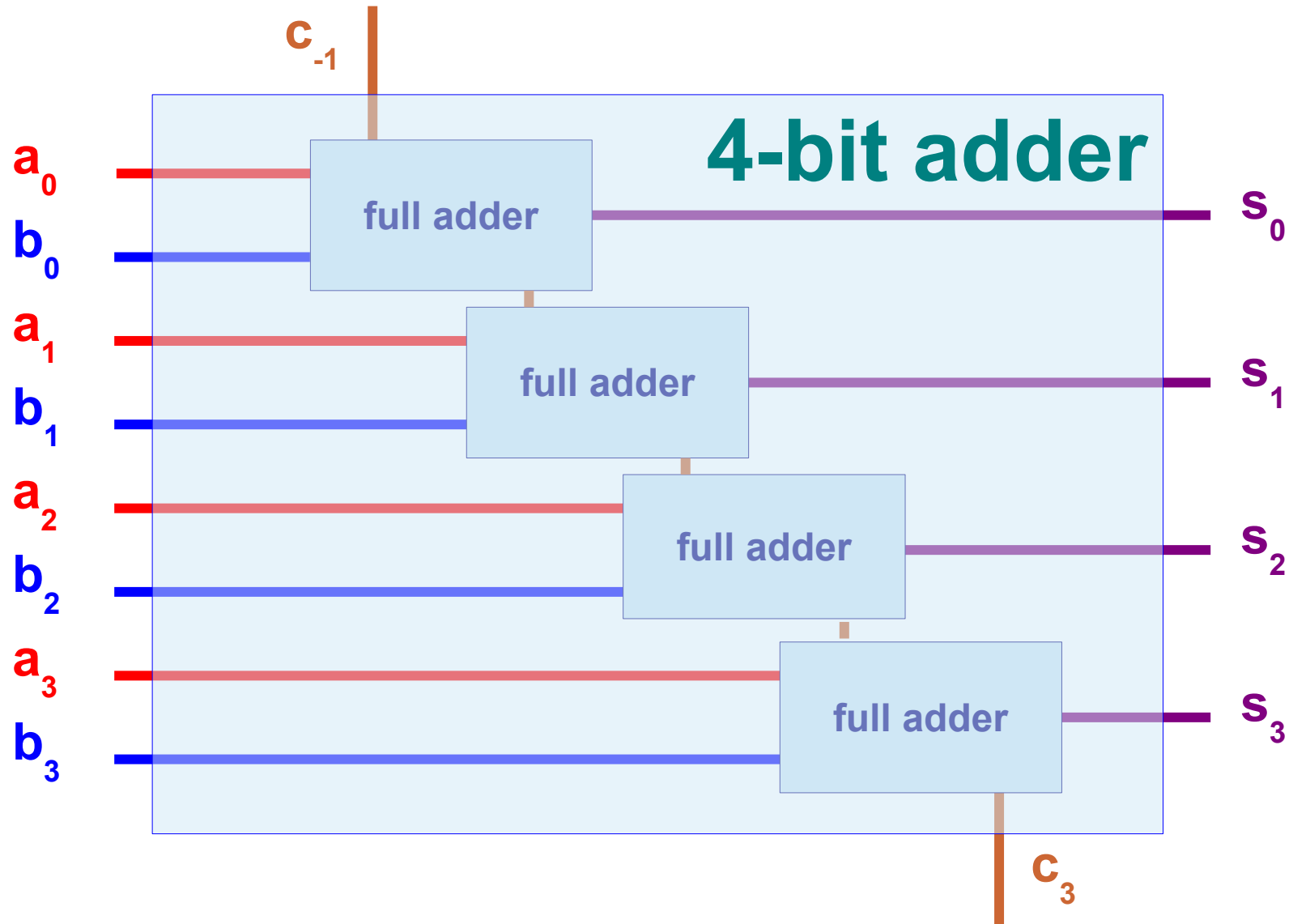
Like base 10 – but with sixteen billion billion fingers

**Simplifies subtraction**

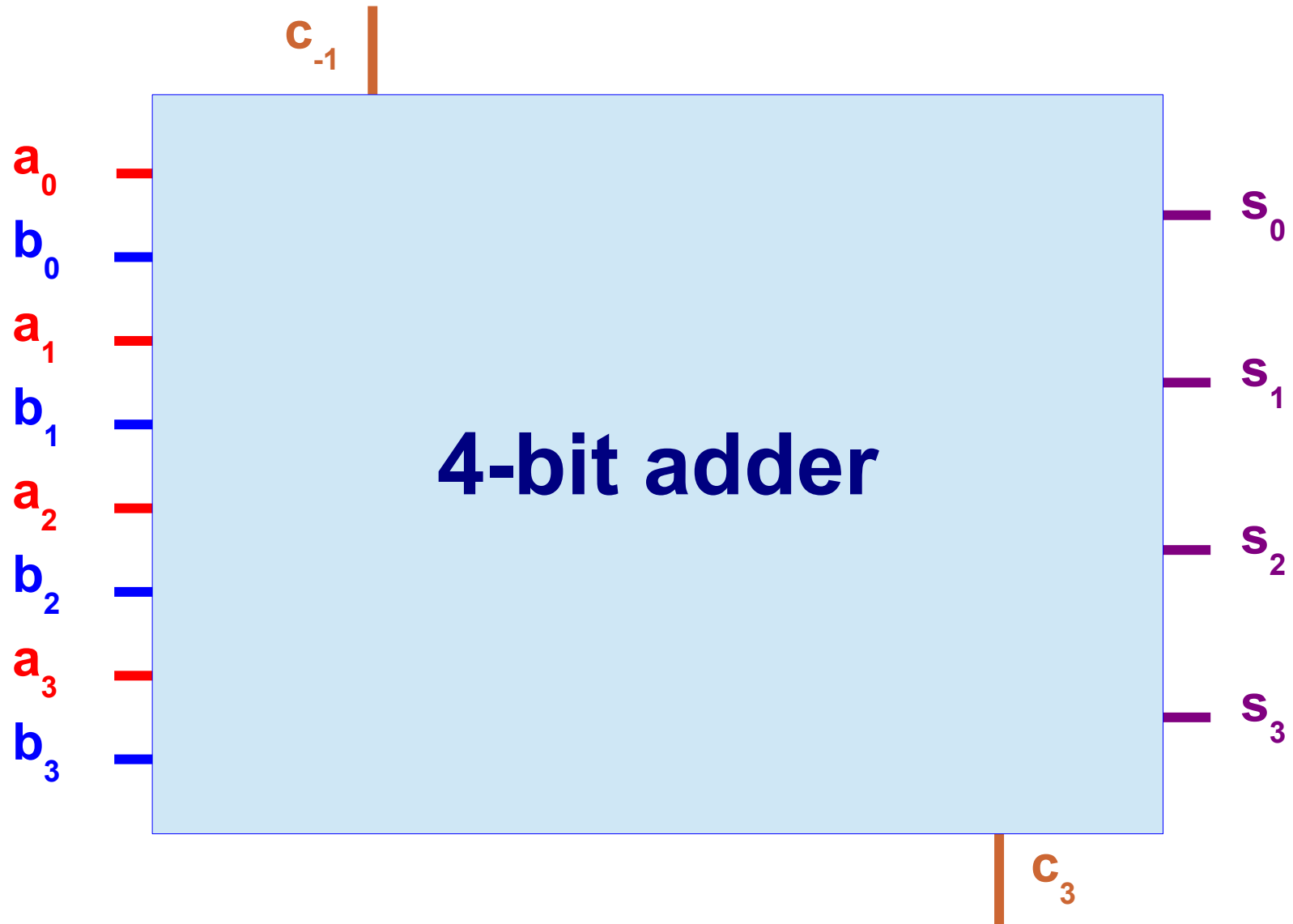
# 4-Bit Ripple Carry Adder



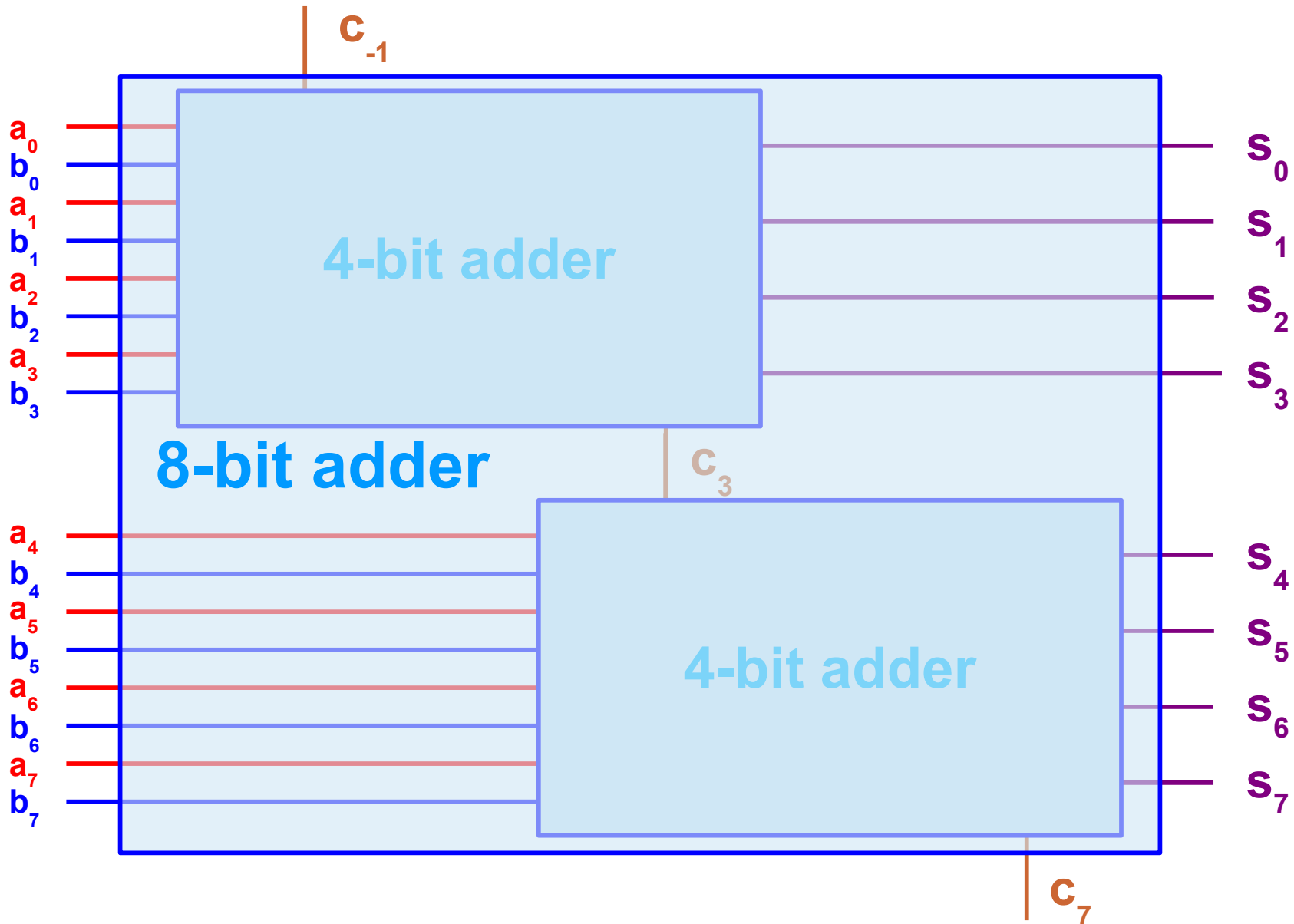
# 4-Bit Ripple Carry Adder



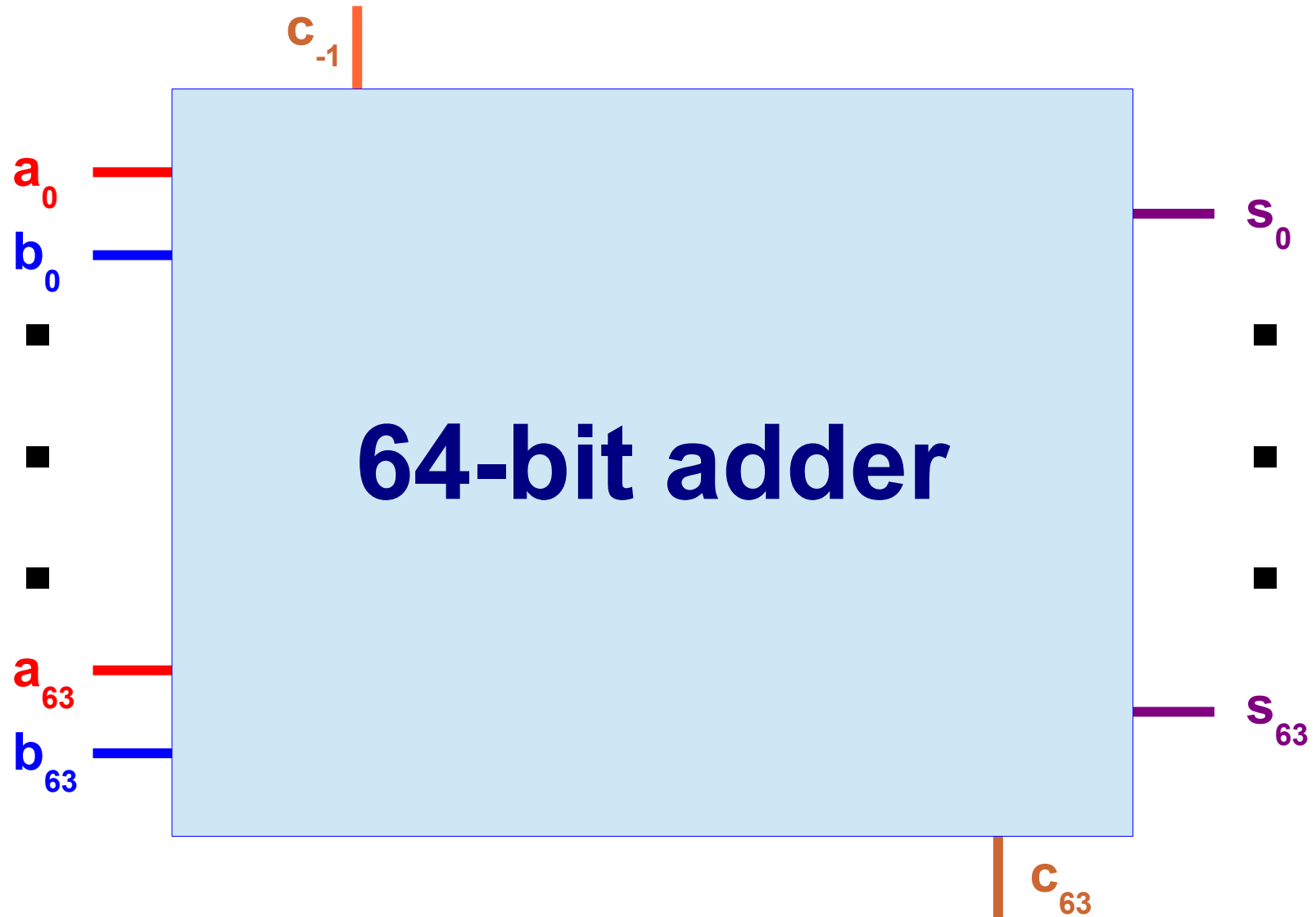
# 4-Bit Ripple Carry Adder



# 8-Bit Ripple Carry Adder



# 64-Bit Ripple Carry Adder



# Fixed Width Binary Addition

		1	1	1	0
+		0	0	1	0
<hr/>					

# Fixed Width Binary Addition

A diagram illustrating fixed-width binary addition. A horizontal line is crossed by a vertical line, creating a grid. To the left of the vertical line is a plus sign '+'. To the right of the vertical line, there are four columns of digits. The first row contains the digits 1, 1, 1, 0 in red. The second row contains the digits 0, 0, 1, 0 in blue. The third row contains a single digit 0 in green, positioned under the third column. The fourth row contains a single digit 0 in purple, positioned under the fourth column.

	1	1	1	0
+	0	0	1	0
			0	
				0



# Fixed Width Binary Addition

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

# Fixed Width Binary Addition

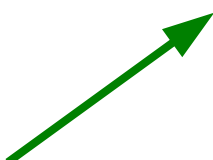
		1	1	1	0
		0	0	1	0
+		1	1	0	
			0	0	0

# Fixed Width Binary Addition

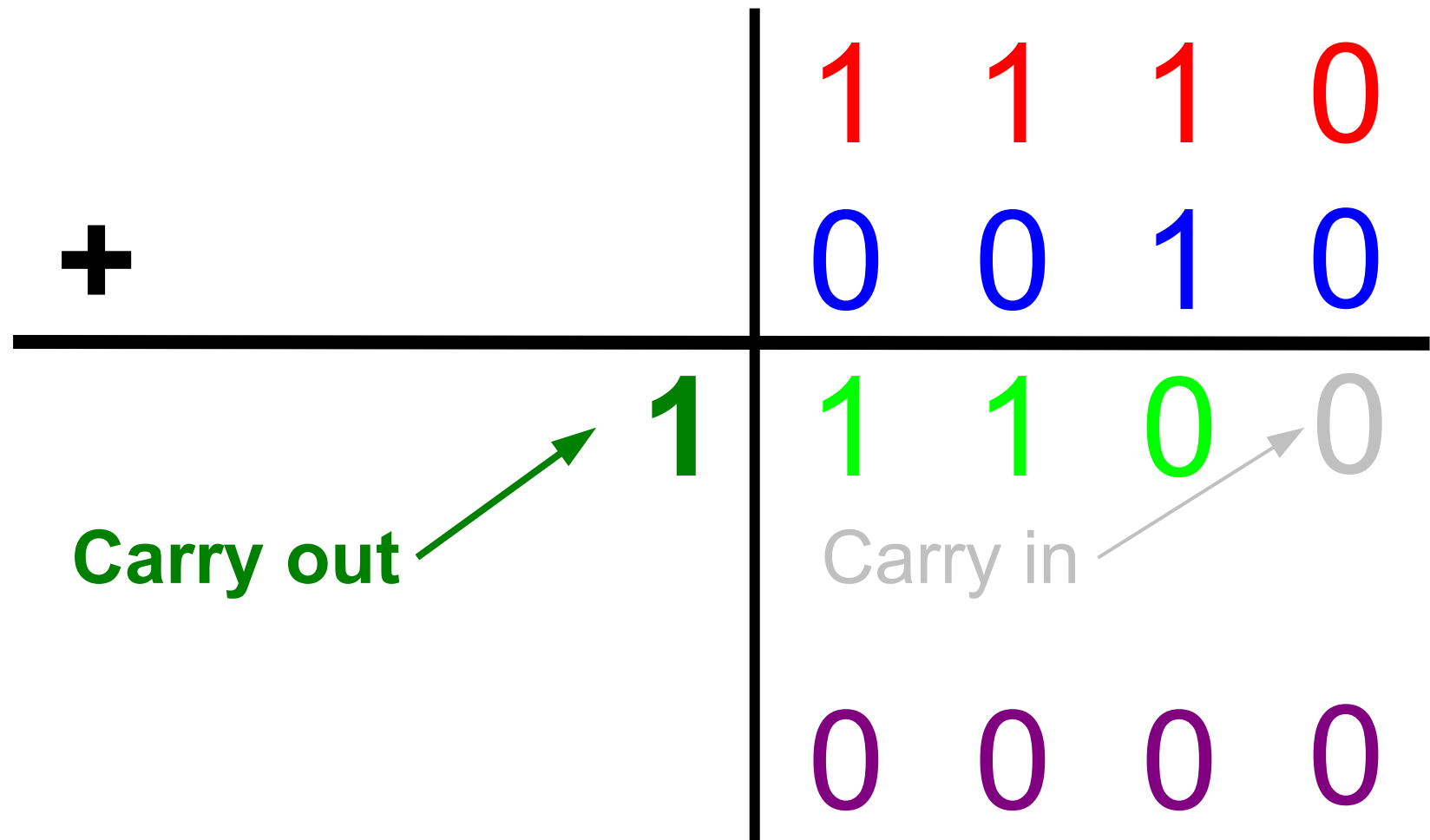
		1	1	1	0
		0	0	1	0
+					
	1	1	1	0	
		0	0	0	0

# Fixed Width Binary Addition

		1	1	1	0
		0	0	1	0
+		<hr/>			
	1	1	1	0	
		0	0	0	0

Carry out 

# Fixed Width Binary Addition



# Non-Negative Numbers Subtraction

$$\mathbf{A \geq B}$$

$$\mathbf{A \sim B \equiv A - B} \text{ (ordinary arithmetic)}$$

$$\mathbf{A < B}$$

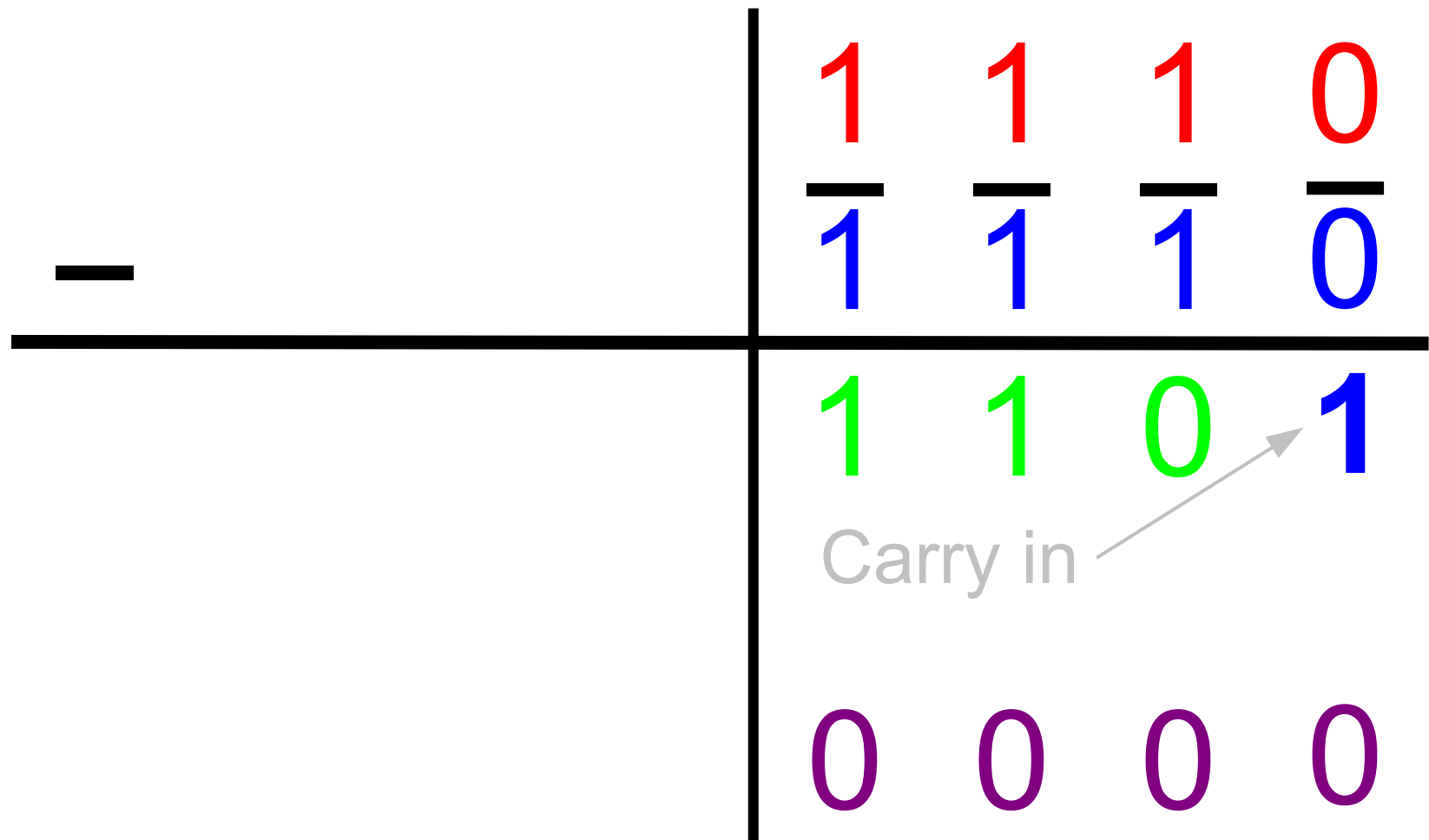
$$1) \mathbf{B \sim B = 0}$$

$$2) \mathbf{(A \sim B) + C = (A + C) \sim B}$$

$$\mathbf{\sim B \equiv 0 \sim B = 2^n - B = \overline{B} + 1} \quad \sim \mathbf{B} \text{ pseudoinverse of } \mathbf{B}$$

$$\mathbf{A \sim B \equiv A + \overline{B} + 1}$$

# Fixed Width Binary Addition



# Negative Number Representation

## Alternatives

### 1. Sign-magnitude

How would it help

### 2. Bias

Complicates arithmetic

### 3. 1's complement

Too many zeros

### 4. 2's complement



# Negative Number Representation

Unsigned numbers:  $0 \dots 2^N - 1$

Signed number alternatives

1. Sign-magnitude:  $-2^{N-1}-1 \dots 2^{N-1}-1$

How would this help?

2. Bias:  $-\text{bias} \dots 2^N-1 - \text{bias}$

Complicates arithmetic

$$(a-\text{bias} + b-\text{bias}) = (a + b)-\text{bias}-\text{bias}$$

3. 1's complement:  $-2^{N-1}-1 \dots 2^{N-1}-1$

+0 and -0

4. 2's complement:  $-2^{N-1} \dots 2^{N-1}-1$

# Negative Number Representation

#	binary	sign magnitude	bias (8)	1's complement	2's complement
0	0000	+ 0	-8	0	0
1	0001	+ 1	-7	1	1
2	0010	+ 2	-6	2	2
3	0011	+ 3	-5	3	3
4	0100	+ 4	-4	4	4
5	0101	+ 5	-3	5	5
6	0110	+ 6	-2	6	6
7	0111	+ 7	-1	7	7
8	1000	- 0	0	-7	-8
9	1001	- 1	1	-6	-7
A	1010	- 2	2	-5	-6
B	1011	- 3	3	-4	-5
C	1100	- 4	4	-3	-4
D	1101	- 5	5	-2	-3
E	1110	- 6	6	-1	-2
F	1111	- 7	7	-0	-1

# 10's Complement Arithmetic

The 9's complement,  $\widetilde{d}$ , of a decimal digit  $d$  is  $9 - d$

The 9's complement,  $\widetilde{X}$ , of a 4-digit  $X$  is  $9999 - X$

The 10's complement,  $\overline{X}$ , of  $X$  is  $\widetilde{X} + 1$

$$\overline{X} = \widetilde{X} + 1 = 9999 - X + 1 = 10000 - X$$

**convention:**  $X$  is positive and  $Y$  is negative iff

$$0 \leq X < 5000 \leq Y < 10000$$

$$-Y \equiv \overline{Y} \text{ and}$$

$$X - Y = X + \overline{Y} \text{ unless}$$

**Overflow** —  $\text{sign}(X) = \text{sign}(Y) \neq \text{sign}(X + Y)$

—  $\text{sign}(X) \neq \text{sign}(Y) = \text{sign}(X - Y)$

# 10's Complement Example

$$\overline{1000} = 8999$$

$$\overline{1000} = 8999 + 1 = 9000$$

$$3000 + \overline{1000} = 12000 \approx 2000 = 3000 - 1000$$

$$-200 \approx 10000 - 200 = 9799 + 1 = \overline{200} + 1 = \overline{200}$$

$$-300 \approx 10000 - 300 = 9699 + 1 = \overline{300} + 1 = \overline{300}$$

$$100 + \overline{300} = 9800 = \overline{200} \approx 100 - 300$$

## ***overflow***

$$4000 + 2000 = 6000 = 5999 + 1 = \overline{4001} + 1 \neq -4001$$

# 2's Complement Arithmetic

The 1's complement,  $\bar{b}$ , of a binary bit  $b$  is  $1 - b$

The 1's complement,  $\bar{X}$ , of a 4-bit  $X$  is  $1111 - X$

The 2's complement,  $\bar{\bar{X}}$ , of  $X$  is  $\bar{X} + 1$

$$\bar{\bar{X}} = \bar{X} + 1 = 1111 - X + 1 = 10000 - X$$

**convention:**  $X$  is positive and  $Y$  is negative iff

$$0 \leq X < 2^{N-1} \leq Y < 2^N$$

$-Y \equiv \bar{\bar{Y}}$  and

$$X - Y = X + \bar{\bar{Y}} \text{ unless}$$

**Overflow** —  $\text{sign}(X) = \text{sign}(Y) \neq \text{sign}(X + Y)$

—  $\text{sign}(X) \neq \text{sign}(Y) \neq \text{sign}(X - Y)$

# N-Bit Integers (N = 8)

Unsigned Integers

...	0100000000
...	0011111111
...	0010000000
...	0001111111
...	0000000101
...	0000000100
...	0000000011
...	0000000010
...	0000000001
...	0000000000
...	1111111111
...	1111111110
...	1111111101
...	1111111100
...	1111111011
...	1110000000
...	1101111111
...	1100000000
...	1011111111

$$= 2^N$$

$$= 2^N - 1$$

$$= 2^{N-1}$$

$$= 2^{N-1} - 1$$

$$= 4 = 2^2$$

$$= 3 = 2^2 - 1$$

$$= 2 = 2^1$$

$$= 1 = 2^0 = 2^1 - 1$$

$$= 0 = 2^0 - 1$$

$$= -1 = -2^0$$

$$= -2 = -2^1 = 2^0 - 1$$

$$= -3 = -2^1 - 1$$

$$= -4 = -2^2$$

$$= -5 = -2^2 - 1$$

$$= -2^{N-1}$$

$$= -2^{N-1} - 1$$

$$= -2^N$$

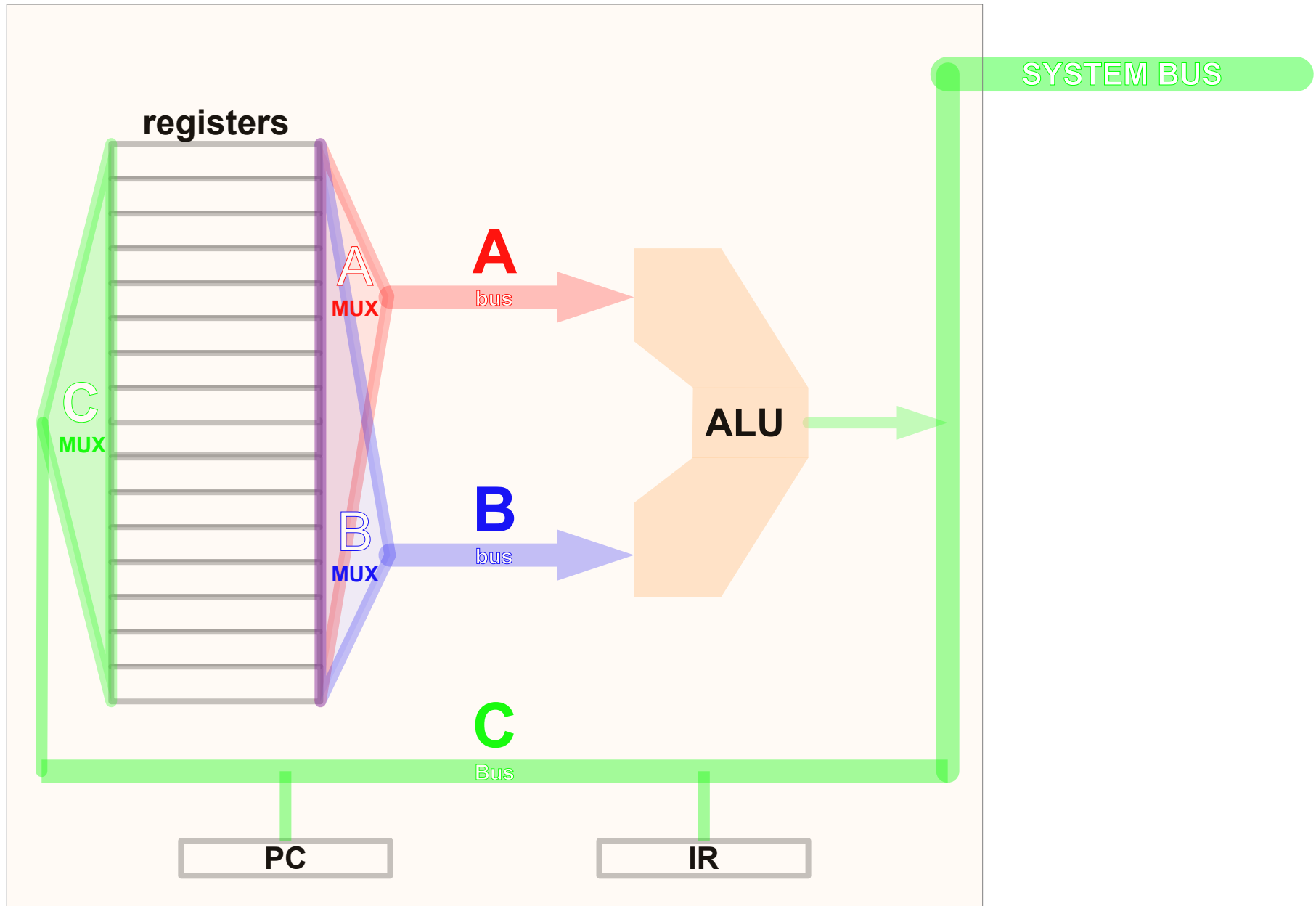
$$= -2^N - 1$$

Signed Integers

# Negative Number Representation

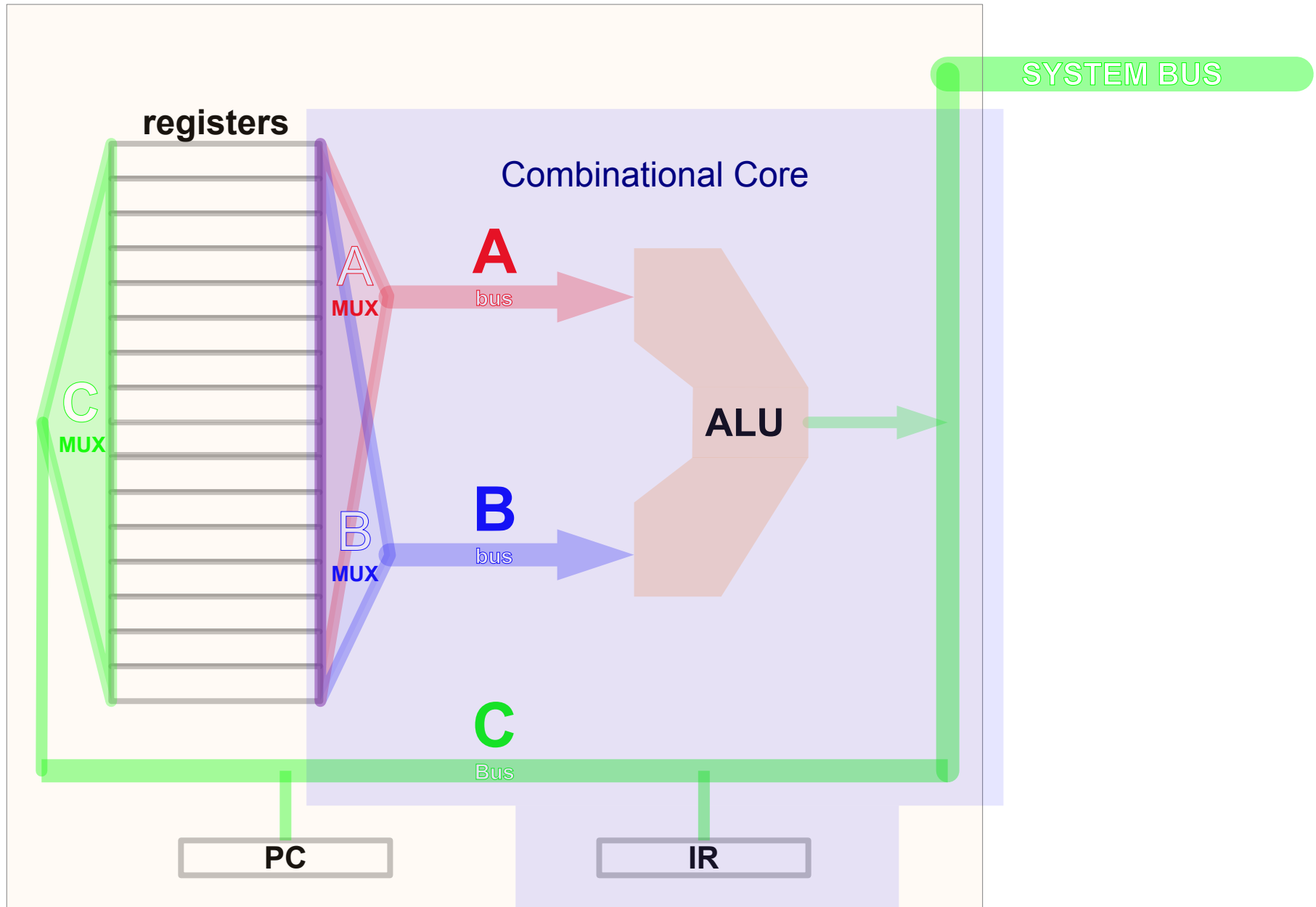
#	binary	sign magnitude	bias (-7)	1's complement	2's complement
0	0000	+ 0	-7	0	0
1	0001	+ 1	-6	1	1
2	0010	+ 2	-5	2	2
3	0011	+ 3	-4	3	3
4	0100	+ 4	-3	4	4
5	0101	+ 5	- 2	5	5
6	0110	+ 6	- 1	6	6
7	0111	+ 7	- 0	7	7
8	1000	- 0	1	-7	-8
9	1001	- 1	2	-6	-7
A	1010	- 2	3	-5	-6
B	1011	- 3	4	-4	-5
C	1100	- 4	5	-3	-4
D	1101	- 5	6	-2	-3
E	1110	- 6	7	-1	-2
F	1111	- 7	8	-0	-1

# Basic Processor Model





# Basic Processor Model



# Arithmetic / Logical Unit

## Building Blocks

AND, OR, and NOT gates

***Inverters, Decoders, Multiplexers***

Inputs (*operands*): **A** and **B** buses

Output (*result*): **C** bus

## Logical

Bitwise:  $\overline{\mathbf{A}}$ ,  $\mathbf{A} \& \mathbf{B}$ ,  $\mathbf{A} | \mathbf{B}$ ,  $\mathbf{A} \wedge \mathbf{B}$ ,  $\mathbf{A} \uparrow \mathbf{B}$ ,  $\mathbf{A} \downarrow \mathbf{B}$ , ...

## Arithmetic

$\mathbf{A} + \mathbf{B}$ ,  $\mathbf{A} - \mathbf{B}$ ,  $\mathbf{A} \cdot \mathbf{B}$ ,  $\mathbf{A} \text{ div } \mathbf{B}$ ,  $\mathbf{A} \text{ mod } \mathbf{B}$

## Comparison

$\mathbf{A} < \mathbf{B}$ ,  $\mathbf{A} = \mathbf{B}$ ,  $\mathbf{A} \geq \mathbf{B}$ , etc. and  $\mathbf{X} < 0$ ,  $\mathbf{X} = 0$ ,  $\mathbf{X} \geq 0$ , etc.

# TINY Arithmetic / Logical Unit

## Building Blocks

AND, OR, and NOT gates

***Inverters, Decoders, Multiplexers***

Inputs (*operands*): **A** and **B** buses

Output (*result*): **C** bus

## Logical

Bitwise:  $\overline{\mathbf{A}}$ ,  $\mathbf{A} \& \mathbf{B}$ ,  $\mathbf{A} \mid \mathbf{B}$ ,  $\mathbf{A} \wedge \mathbf{B}$ ,  $\mathbf{A} \uparrow \mathbf{B}$ ,  $\mathbf{A} \downarrow \mathbf{B}$ , ...

## Arithmetic

$\mathbf{A} + \mathbf{B}$ ,  $\mathbf{A} - \mathbf{B}$ ,  $\mathbf{A} \cdot \mathbf{B}$ ,  $\mathbf{A} \text{ div } \mathbf{B}$ ,  $\mathbf{A} \text{ mod } \mathbf{B}$

## Comparison

$\mathbf{A} < \mathbf{B}$ ,  $\mathbf{A} = \mathbf{B}$ ,  $\mathbf{A} \geq \mathbf{B}$ , etc. and  $\mathbf{X} < 0$ ,  $\mathbf{X} = 0$ ,  $\mathbf{X} \geq 0$ , etc.

# Muxes, Buses, and ALU

ALU inputs (*operands*): **A** and **B** buses

ALU output (*result*): **C** bus

Logical

Bitwise:  $\overline{\mathbf{A}}$ ,  $\mathbf{A} \& \mathbf{B}$ ,  $\mathbf{A} | \mathbf{B}$ ,  $\mathbf{A} \wedge \mathbf{B}$ ,  $\mathbf{A} \uparrow \mathbf{B}$ ,  $\mathbf{A} \downarrow \mathbf{B}$ , ...

Arithmetic

$\mathbf{A} + \mathbf{B}$ ,  $\mathbf{A} - \mathbf{B}$ ,  $\mathbf{A} \cdot \mathbf{B}$ ,  $\mathbf{A} \text{ div } \mathbf{B}$ ,  $\mathbf{A} \text{ mod } \mathbf{B}$

Comparison

$\mathbf{A} < \mathbf{B}$ ,  $\mathbf{A} = \mathbf{B}$ ,  $\mathbf{A} \geq \mathbf{B}$ , etc. and  $\mathbf{X} < 0$ ,  $\mathbf{X} = 0$ ,  $\mathbf{X} \geq 0$ , etc.

Combinational building blocks

AND, OR, and NOT gates

***Inverters, Decoders, Multiplexers***

# 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

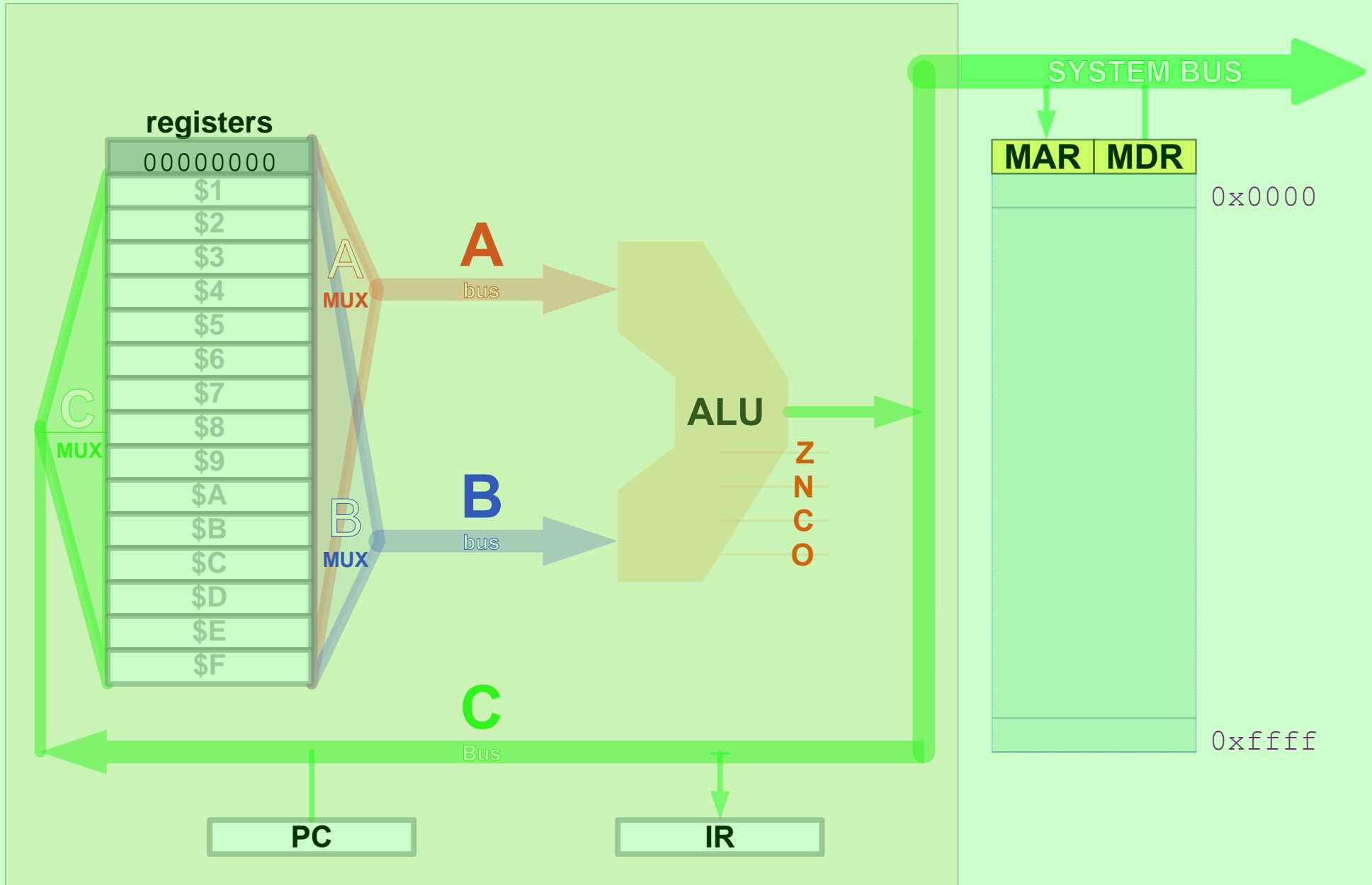
Multiplexer: select unique data input to be output

- $2^N$  data inputs

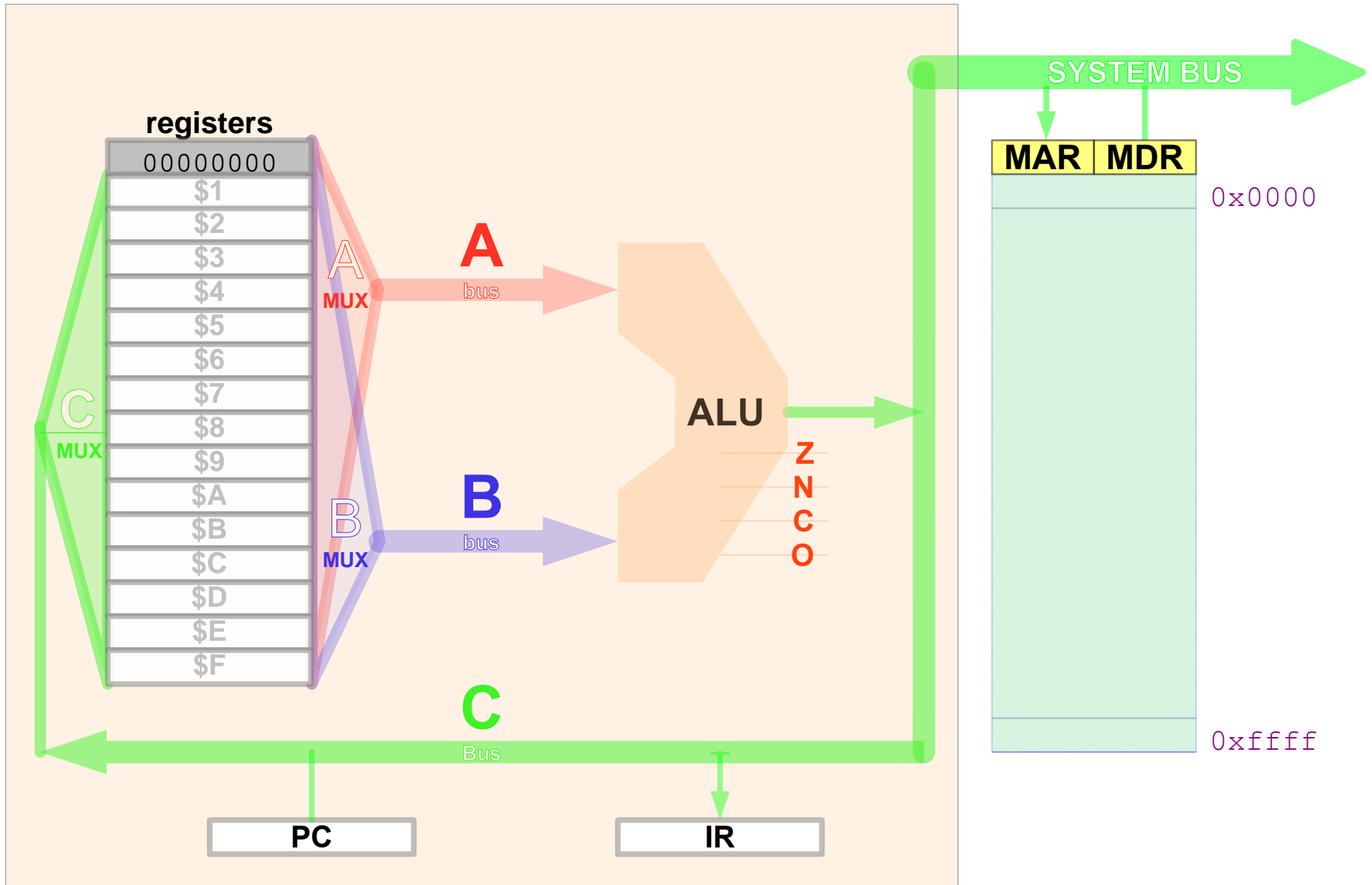
- N selector inputs

- 1 output

# The TINY Computer



# The TINY Computer



## Main Memory 65536 16-bit words

$M[n]$  –  $n^{\text{th}}$  memory address

$\wedge M[n]$  – content of  $M[n]$

## Register File 16 16-bit “registers”

15 real registers: \$1 ... \$F

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

## Immediate values

In — n-bit signed int

Un — n-bit unsigned int

CC — 4-bit condition code

## Instructions<sup>0</sup>

ADD  $rT \leftarrow [rA] + [rB]$  <sup>1,2</sup>

AND  $rT \leftarrow [rA] \& [rB]$  <sup>1,3</sup>

BRC  $PC \leftarrow [rA] + U4 + 1$  *cc* CC

BRU  $rL \leftarrow PC, PC \leftarrow [rA] + [rB]$  <sup>1</sup>

LDI  $rT \leftarrow \wedge M[[rA] + U4 + 1]$  <sup>1</sup>

LDX  $rT \leftarrow \wedge M[[rA] + [rB]]$  <sup>1</sup>

LIH  $rT_{15..8} \leftarrow I8$  <sup>1</sup>

NOR  $rT \leftarrow \overline{[rA] \mid [rB]}$  <sup>1,3</sup>

SLL  $rT \leftarrow [rA] \ll I4$  <sup>1,3</sup>

SRS  $rT \leftarrow [rA] \gg I4$  <sup>1,3</sup>

SRU  $rT \leftarrow [rA] \ggg I4$  <sup>1,3</sup>

STI  $M[[rA] + U4 + 1] \leftarrow [rS]$

STX  $M[[rA] + [rB]] \leftarrow [rS]$

SUB  $rT \leftarrow [rA] - [rB]$  <sup>1, 2</sup>

SYS system call <sup>4</sup>

## Arithmetic / Logical

0100	ADD	rT	rA	rB
0101	SUB	rT	rA	rB
0110	AND	rT	rA	rB
0111	NOR	rT	rA	rB

## Shift / Load Immediate

1000	LIH	rT	I8	
1001	SLL	rT	rA	U4
1010	SRS	rT	rA	U4
1011	SRU	rT	rA	U4

## Load/Store

0111	LDI	rT	rA	U4
0110	LDX	rT	rA	rB
0101	STI	rS	rA	U4
0100	STX	rS	rA	RB

## Branch/Special

0011	BRC	C	rA	U4
0010	BCU	rL	rA	rB
0001	reserved			
0000	SYS	U12		

## Condition Codes

0000	true	TT
0001	false	FF
0010	A = B signed	EQ
0011	A < B signed	NE
0100	A < B signed	LT
0101	A < B signed	GE
0110	A < B signed	LE
0111	A > B signed	GT
1000	true	
1001	false	
1010	A = B unsigned	
1011	A < B unsigned	
1100	A < B unsigned	LTU
1101	A < B unsigned	GEU
1110	A < B unsigned	LEU
1111	A > B unsigned	GTU

## Notes

<sup>0</sup> PC  $\leftarrow$  PC+1 *before* instruction execution

<sup>1</sup> \$0 *not* changed

<sup>2</sup> Determines flags: **z**, **n**, **c**, **o**

<sup>3</sup> Determines flags: **z**, **n**,

<sup>4</sup> No op *cc* U12 = 0