

► **Concept:** consider

$$\begin{array}{ccc} \hat{x} & \mapsto & x \\ \hat{y} & \mapsto & y \end{array}$$

► **Concept:** consider

$$\begin{array}{lll} \hat{x} & \mapsto & x \\ \hat{y} & \mapsto & y \\ \hat{r} & \mapsto & r = x \times y \end{array}$$

► **Concept:** consider

$$\begin{array}{rcl}
 \hat{x} & \mapsto & x \\
 \hat{y} & \mapsto & y \\
 f(\hat{x}, \hat{y}) = \hat{r} & \mapsto & r = x \times y
 \end{array}$$

where f

1. has an action on $\hat{x}$ and $\hat{y}$ compatible with that of $\times$ on x and y :
 - accepts n -bit
 - **multiplier** $\hat{y}$ (that “does the multiplying”), and
 - **multiplicand** $\hat{x}$ (that “is multiplied”)
 - as input, and
 - produces an $(2 \cdot n)$ -bit **product** $\hat{r}$ as output,
2. is a Boolean function:

$$f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^{2 \cdot n}$$

- **Agenda:** produce a design(s) for f , which
 1. functions correctly, and
 2. satisfies pertinent quality metrics (e.g., is efficient in time and/or space).

Part 1: multiplication in theory (1)

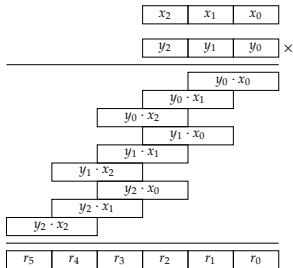
Example

$x =$	$623_{(10)} \mapsto$			6	2	3	
$y =$	$567_{(10)} \mapsto$			5	6	7	$\times$
$p_0 = 7 \cdot 3 \cdot 10^0 =$	$21_{(10)} \mapsto$				2	1	
$p_1 = 7 \cdot 2 \cdot 10^1 =$	$140_{(10)} \mapsto$				1	4	
$p_2 = 7 \cdot 6 \cdot 10^2 =$	$4200_{(10)} \mapsto$		4		2		
$p_3 = 6 \cdot 3 \cdot 10^1 =$	$180_{(10)} \mapsto$				1	8	
$p_4 = 6 \cdot 2 \cdot 10^2 =$	$1200_{(10)} \mapsto$				1	2	
$p_5 = 6 \cdot 6 \cdot 10^3 =$	$36000_{(10)} \mapsto$		3		6		
$p_6 = 5 \cdot 3 \cdot 10^2 =$	$1500_{(10)} \mapsto$				1	5	
$p_7 = 5 \cdot 2 \cdot 10^3 =$	$10000_{(10)} \mapsto$		1		0		
$p_8 = 5 \cdot 6 \cdot 10^4 =$	$300000_{(10)} \mapsto$	3	0				
$r =$	$353241_{(10)} \mapsto$	3	5	3	2	4	1

Part 1: multiplication in theory (2)

Example (operand scanning)

Consider an example where where $|x| = |y| = 3$:

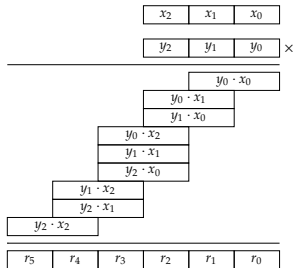


Notice that

1. an outer-loop steps through digits of y , say y_j ,
2. an inner-loop steps through digits of x , say x_i .

Example (product scanning)

Consider an example where where $|x| = |y| = 3$:



Notice that

1. an outer-loop steps through digits of r , say r_k ,
2. two inner-loops step through matching digits of y and x , say y_j and x_i .

Part 2: multiplication in practice: an algorithm (1)

Operand scanning

Algorithm (operand scanning)

Input: Two unsigned, base- b integers x and y

Output: An unsigned, base- b integer $r = x \cdot y$

```
1  $l_x \leftarrow |x|, l_y \leftarrow |y|, l_r \leftarrow l_x + l_y$ 
2  $r \leftarrow 0$ 
3 for  $j = 0$  upto  $l_y - 1$  step  $+1$  do
4    $c \leftarrow 0$ 
5   for  $i = 0$  upto  $l_x - 1$  step  $+1$  do
6      $u \cdot b + v = t \leftarrow y_j \cdot x_i + r_{j+i} + c$ 
7      $r_{j+i} \leftarrow v$ 
8      $c \leftarrow u$ 
9   end
10   $r_{j+l_x} \leftarrow c$ 
11 end
12 return  $r$ 
```

Part 2: multiplication in practice: an algorithm (2)

Operand scanning

Example (operand scanning)

Consider a case where $b = 10$, $x = 623_{(10)}$ and $y = 567_{(10)}$:

j	i	r	c	y_i	x_j	$t = y_i \cdot x_j + r_{i+j} + c$	r'	c'
		$\langle 0, 0, 0, 0, 0, 0 \rangle$						
0	0	$\langle 0, 0, 0, 0, 0, 0 \rangle$	0	7	3	21	$\langle 1, 0, 0, 0, 0, 0 \rangle$	2
0	1	$\langle 1, 0, 0, 0, 0, 0 \rangle$	2	7	2	16	$\langle 1, 6, 0, 0, 0, 0 \rangle$	1
0	2	$\langle 1, 6, 0, 0, 0, 0 \rangle$	1	7	6	43	$\langle 1, 6, 3, 0, 0, 0 \rangle$	4
0		$\langle 1, 6, 3, 0, 0, 0 \rangle$	4				$\langle 1, 6, 3, 4, 0, 0 \rangle$	
1	0	$\langle 1, 6, 3, 4, 0, 0 \rangle$	0	6	3	24	$\langle 1, 4, 3, 4, 0, 0 \rangle$	2
1	1	$\langle 1, 4, 3, 4, 0, 0 \rangle$	2	6	2	17	$\langle 1, 4, 7, 4, 0, 0 \rangle$	1
1	2	$\langle 1, 4, 7, 4, 0, 0 \rangle$	1	6	6	41	$\langle 1, 4, 7, 1, 0, 0 \rangle$	4
1		$\langle 1, 4, 7, 1, 0, 0 \rangle$	4				$\langle 1, 4, 7, 1, 4, 0 \rangle$	
2	0	$\langle 1, 4, 7, 1, 4, 0 \rangle$	0	5	3	22	$\langle 1, 4, 2, 1, 4, 0 \rangle$	2
2	1	$\langle 1, 4, 2, 1, 4, 0 \rangle$	2	5	2	13	$\langle 1, 4, 2, 3, 4, 0 \rangle$	1
2	2	$\langle 1, 4, 2, 3, 5, 0 \rangle$	1	5	6	35	$\langle 1, 4, 2, 3, 5, 0 \rangle$	3
2		$\langle 1, 4, 2, 3, 5, 0 \rangle$	3				$\langle 1, 4, 2, 3, 5, 3 \rangle$	3
		$\langle 1, 4, 2, 3, 5, 3 \rangle$						

Part 2: multiplication in practice: an algorithm (3)

Product scanning

Algorithm (product scanning)

Input: Two unsigned, base- b integers x and y

Output: An unsigned, base- b integer $r = x \cdot y$

```
1  $l_x \leftarrow |x|, l_y \leftarrow |y|, l_r \leftarrow l_x + l_y$ 
2  $r \leftarrow 0, c_0 \leftarrow 0, c_1 \leftarrow 0, c_2 \leftarrow 0$ 
3 for  $k = 0$  upto  $l_x + l_y - 1$  step  $+1$  do
4   for  $j = 0$  upto  $l_y - 1$  step  $+1$  do
5     for  $i = 0$  upto  $l_x - 1$  step  $+1$  do
6       if  $(j + i) = k$  then
7          $u \cdot b + v = t \leftarrow y_j \cdot x_i$ 
8          $c \cdot b + c_0 = t \leftarrow c_0 + v$ 
9          $c \cdot b + c_1 = t \leftarrow c_1 + u + c$ 
10         $c_2 \leftarrow c_2 + c$ 
11      end
12    end
13  end
14   $r_k \leftarrow c_0, c_0 \leftarrow c_1, c_1 \leftarrow c_2, c_2 \leftarrow 0$ 
15 end
16  $r_{l_x+l_y-1} \leftarrow c_0$ 
```

Part 2: multiplication in practice: an algorithm (4)

Product scanning

Example (product scanning)

Consider a case where $b = 10$, $x = 623_{(10)}$ and $y = 567_{(10)}$:

k	j	i	r	c_2	c_1	c_0	y_i	x_j	$t = y_i \cdot x_j$	r'	c'_2	c'_1	c'_0
			$\langle 0, 0, 0, 0, 0, 0 \rangle$	0	0	0				$\langle 0, 0, 0, 0, 0, 0 \rangle$	0		
0	0	0	$\langle 0, 0, 0, 0, 0, 0 \rangle$	0	0	0	7	3	21	$\langle 0, 0, 0, 0, 0, 0 \rangle$	0	2	1
0			$\langle 0, 0, 0, 0, 0, 0 \rangle$	0	2	1				$\langle 1, 0, 0, 0, 0, 0 \rangle$	0	0	2
1	0	1	$\langle 1, 0, 0, 0, 0, 0 \rangle$	0	0	2	7	2	14	$\langle 1, 0, 0, 0, 0, 0 \rangle$	0	1	6
1	1	0	$\langle 1, 0, 0, 0, 0, 0 \rangle$	0	1	6	6	3	18	$\langle 1, 0, 0, 0, 0, 0 \rangle$	0	3	4
1			$\langle 1, 0, 0, 0, 0, 0 \rangle$	0	3	4				$\langle 1, 4, 0, 0, 0, 0 \rangle$	0	0	3
2	0	2	$\langle 1, 4, 0, 0, 0, 0 \rangle$	0	0	3	7	6	42	$\langle 1, 4, 0, 0, 0, 0 \rangle$	0	4	5
2	1	1	$\langle 1, 4, 0, 0, 0, 0 \rangle$	0	4	5	6	2	12	$\langle 1, 4, 0, 0, 0, 0 \rangle$	0	5	7
2	2	0	$\langle 1, 4, 0, 0, 0, 0 \rangle$	0	4	7	5	3	15	$\langle 1, 4, 0, 0, 0, 0 \rangle$	0	7	2
2			$\langle 1, 4, 0, 0, 0, 0 \rangle$	0	7	2				$\langle 1, 4, 2, 0, 0, 0 \rangle$	0	0	7
3	1	2	$\langle 1, 4, 2, 0, 0, 0 \rangle$	0	0	7	6	6	36	$\langle 1, 4, 2, 0, 0, 0 \rangle$	0	4	3
3	2	1	$\langle 1, 4, 2, 0, 0, 0 \rangle$	0	4	3	5	2	10	$\langle 1, 4, 2, 0, 0, 0 \rangle$	0	5	3
3			$\langle 1, 4, 2, 0, 0, 0 \rangle$	0	5	3				$\langle 1, 4, 2, 3, 0, 0 \rangle$	0	0	5
4	2	2	$\langle 1, 4, 2, 3, 0, 0 \rangle$	0	0	5	5	6	30	$\langle 1, 4, 2, 3, 0, 0 \rangle$	0	3	5
4			$\langle 1, 4, 2, 3, 0, 0 \rangle$	0	3	5				$\langle 1, 4, 2, 3, 5, 0 \rangle$	0	0	3
			$\langle 1, 4, 2, 3, 5, 0 \rangle$	0	0	3				$\langle 1, 4, 2, 3, 5, 3 \rangle$	0	0	3
			$\langle 1, 4, 2, 3, 5, 3 \rangle$										

Part 2: multiplication in practice: an algorithm (5)

Repeated addition

► Idea:

- multiplication *means* repeated addition, i.e.,

$$y \times x = \underbrace{x + x + \cdots + x}_{y \text{ terms}}$$

so if $y = 14_{(10)}$ we have

$$y \times x = x + x + x + x + x + x + x + x + x + x + x + x + x + x.$$

- expressing y in base-2, we can rewrite this as

$$\begin{aligned} y \times x &= \left(\sum_{i=0}^{n-1} y_i \cdot 2^i \right) \times x \\ &= (y_0 \cdot 2^0 + y_1 \cdot 2^1 + \cdots + y_{n-1} \cdot 2^{n-1}) \times x \\ &= (y_0 \cdot 2^0 \cdot x) + (y_1 \cdot 2^1 \cdot x) + \cdots + (y_{n-1} \cdot 2^{n-1} \cdot x) \\ &= (y_0 \cdot x \cdot 2^0) + (y_1 \cdot x \cdot 2^1) + \cdots + (y_{n-1} \cdot x \cdot 2^{n-1}) \end{aligned}$$

Part 2: multiplication in practice: an algorithm (5)

Repeated addition

► Idea:

- given $y = 14_{(10)} = 1110_{(2)}$ we can see that

$$\begin{aligned}y \times x &= y_0 \cdot x \cdot 2^0 + y_1 \cdot x \cdot 2^1 + y_2 \cdot x \cdot 2^2 + y_3 \cdot x \cdot 2^3 \\&= 0 \cdot x \cdot 2^0 + 1 \cdot x \cdot 2^1 + 1 \cdot x \cdot 2^2 + 1 \cdot x \cdot 2^3 \\&= 0 \cdot x + 2 \cdot x + 4 \cdot x + 8 \cdot x \\&= 14 \cdot x\end{aligned}$$

- given $y = 14_{(10)} = 1110_{(2)}$ we can see that

$$\begin{aligned}y \times x &= y_0 \cdot x + 2 \cdot (y_1 \cdot x + 2 \cdot (y_2 \cdot x + 2 \cdot (y_3 \cdot x + 2 \cdot (0)))) \\&= 0 \cdot x + 2 \cdot (1 \cdot x + 2 \cdot (1 \cdot x + 2 \cdot (1 \cdot x + 2 \cdot (0)))) \\&= 0 \cdot x + 2 \cdot (1 \cdot x + 2 \cdot (1 \cdot x + 2 \cdot (1 \cdot x + 0))) \\&= 0 \cdot x + 2 \cdot (1 \cdot x + 2 \cdot (1 \cdot x + 2 \cdot (1 \cdot x))) \\&= 0 \cdot x + 2 \cdot (1 \cdot x + 2 \cdot (1 \cdot x + 2 \cdot x)) \\&= 0 \cdot x + 2 \cdot (1 \cdot x + 2 \cdot (3 \cdot x)) \\&= 0 \cdot x + 2 \cdot (1 \cdot x + 6 \cdot x) \\&= 0 \cdot x + 2 \cdot (7 \cdot x) \\&= 0 \cdot x + 14 \cdot x \\&= 14 \cdot x\end{aligned}$$

via application of **Horner's rule**.

Part 3: multiplication in practice: a circuit (1)

A combinatorial, bit-parallel design

► **Idea:** for $b = 2$ we now know

$$r = y \times x = \left(\sum_{i=0}^{n-1} y_i \cdot 2^i \right) \times x = \sum_{i=0}^{n-1} y_i \cdot x \cdot 2^i,$$

plus

► for any t ,

$$y_i \cdot t = \begin{cases} 0 & \text{if } y_i = 0 \\ t & \text{if } y_i = 1 \end{cases}$$

► for any t ,

$$t \cdot 2^i \equiv t \ll i,$$

so we can compute r via

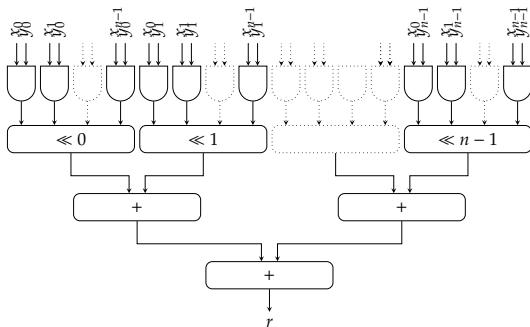
1. some AND gates to generate partial products (i.e., $y_i \cdot x$),
2. some left-shift components to scale the partial products correctly (i.e., $y_i \cdot x \cdot 2^i$), and
3. some adder components to sum the scaled partial products.

Part 3: multiplication in practice: a circuit (2)

A combinatorial, bit-parallel design

► Design:

Circuit



► Evaluation:

- ve: requires a larger data-path
- +ve: requires a smaller control-path (i.e., none at all)
- +ve: requires less steps (i.e., 1)
- ve: has a longer critical path (meaning each step is longer)

Part 3: multiplication in practice: a circuit (3)

An iterative, bit-serial design

- **Idea:** for $b = 2$ we now know

$$r = y \times x = \left(\sum_{i=0}^{n-1} y_i \cdot 2^i \right) \times x = \sum_{i=0}^{n-1} y_i \cdot x \cdot 2^i,$$

so we can compute r by evaluating the Horner expansion: we

- start with the general step

$$r' \leftarrow y_i \cdot x + 2 \cdot r,$$

- specialise it to read

$$r' \leftarrow \begin{cases} 2 \cdot r & \equiv (r \ll 1) & \text{if } y_i = 0 \\ x + 2 \cdot r & \equiv x + (r \ll 1) & \text{if } y_i = 1 \end{cases}$$

- using it to accumulate the result step-by-step.

Part 3: multiplication in practice: a circuit (3)

An iterative, bit-serial design

► Idea:

Algorithm

Input: Two unsigned, n -bit, base-2 integers x and y

Output: An unsigned, $2n$ -bit, base-2 integer
 $r = y \cdot x$

```
1  $r \leftarrow 0$ 
2 for  $i = n - 1$  downto 0 step -1 do
3    $r \leftarrow 2 \cdot r$ 
4   if  $y_i = 1$  then
5      $r \leftarrow r + x$ 
6   end
7 end
8 return  $r$ 
```

Example

Consider a case where $y = 14_{(10)} \mapsto 1110_{(2)}$:

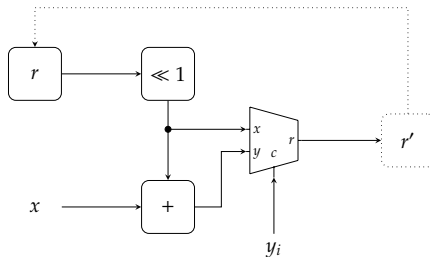
i	r	y_i	r'	
	0			
3	0	1	x	$r' \leftarrow 2 \cdot r + x$
2	x	1	$3 \cdot x$	$r' \leftarrow 2 \cdot r + x$
1	$3 \cdot x$	1	$7 \cdot x$	$r' \leftarrow 2 \cdot r + x$
0	$7 \cdot x$	0	$14 \cdot x$	$r' \leftarrow 2 \cdot r$
	$14 \cdot x$			

Part 3: multiplication in practice: a circuit (4)

An iterative, bit-serial design

► Design:

Circuit



► Evaluation:

- +ve: requires a smaller data-path
- ve: requires a larger control-path (i.e., an entire FSM),
- ve: requires more steps (i.e., n)
- +ve: has a shorter critical path (meaning each step is shorter)

► Take away points:

1. Computer arithmetic is a foundational (sub-)field of computer science:
 - it's a broad topic with a rich history,
 - there's usually a large design space of potential approaches,
 - they're often easy to understand at an intuitive, high level,
 - correctness and efficiency of resulting low-level solutions is vital and challenging.
2. The strategy we've employed is important and (fairly) general-purpose:
 - explore and understand an approach in theory,
 - translate, formalise, and generalise the approach into an algorithm,
 - translate the algorithm, e.g., into circuit,
 - refine (or select) the circuit to satisfy any design constraints.

Additional Reading

- ▶ *Wikipedia: Computer Arithmetic*. URL: https://en.wikipedia.org/wiki/Category:Computer_arithmetic.
- ▶ D. Page. “Chapter 7: Arithmetic and logic”. In: *A Practical Introduction to Computer Architecture*. 1st ed. Springer, 2009.
- ▶ B. Parhami. “Part 3: Multiplication”. In: *Computer Arithmetic: Algorithms and Hardware Designs*. 1st ed. Oxford University Press, 2000.
- ▶ W. Stallings. “Chapter 10: Computer arithmetic”. In: *Computer Organisation and Architecture*. 9th ed. Prentice Hall, 2013.
- ▶ A.S. Tanenbaum and T. Austin. “Section 3.2.2: Arithmetic circuits”. In: *Structured Computer Organisation*. 6th ed. Prentice Hall, 2012.

References

- [1] [Wikipedia: Computer Arithmetic](https://en.wikipedia.org/wiki/Category:Computer_arithmetic). URL: https://en.wikipedia.org/wiki/Category:Computer_arithmetic (see p. 19).
- [2] D. Page. “Chapter 7: Arithmetic and logic”. In: *A Practical Introduction to Computer Architecture*. 1st ed. Springer, 2009 (see p. 19).
- [3] B. Parhami. “Part 3: Multiplication”. In: *Computer Arithmetic: Algorithms and Hardware Designs*. 1st ed. Oxford University Press, 2000 (see p. 19).
- [4] W. Stallings. “Chapter 10: Computer arithmetic”. In: *Computer Organisation and Architecture*. 9th ed. Prentice Hall, 2013 (see p. 19).
- [5] A.S. Tanenbaum and T. Austin. “Section 3.2.2: Arithmetic circuits”. In: *Structured Computer Organisation*. 6th ed. Prentice Hall, 2012 (see p. 19).