# Computer Architecture

Daniel Page

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB. UK.
⟨csdsp@bristol.ac.uk⟩

October 14, 2024

Keep in mind there are *two* PDFs available (of which this is the latter):

1. a PDF of examinable material used as lecture slides, and
2. a PDF of non-examinable, extra material:
   - the associated notes page may be pre-populated with extra, written explaination of material covered in lecture(s), plus
   - anything with a "grey'ed out" header/footer represents extra material which is useful and/or interesting but out of scope (and hence not covered).

Notes:

Notes:

- Concept: consider

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

Notes:



COMS10015 lecture: week #2

- Concept: consider

$$\begin{array}{lcl} \hat{x} & \mapsto & x \\ \hat{y} & \mapsto & y \\ & & r = x + y \end{array}$$

Notes:

© Daniel Page ⟨cs@bristol.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

- Concept: consider

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

where $f$

1. has an action on $\hat{x}$ and $\hat{y}$ compatible with that of $+$ on $x$ and $y$:
   - accepts $n$-bit
     - **addend** $\hat{x}$, and
     - **addend** $\hat{y}$

     as input, and
   - produces an $(n+1)$-bit **sum** $\hat{r}$ as output,
2. is a Boolean function:

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

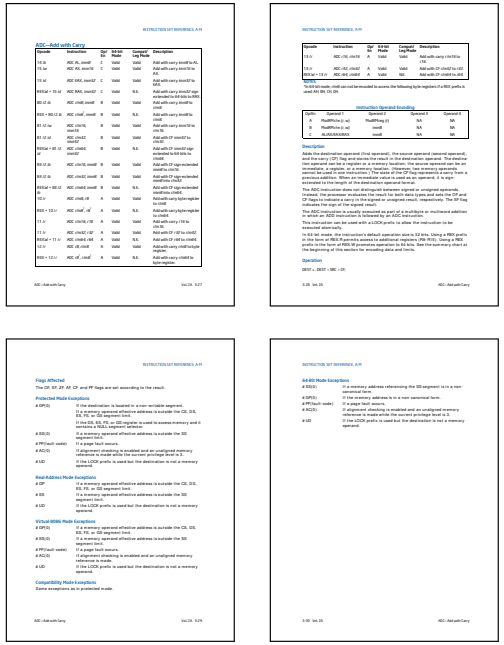
Notes:



COMS10015 lecture: week #2

- 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).

Notes:





Notes:

## Part 1: addition in theory (1)

- Concept:

**Example ($b = 10$)**

$$
\begin{array}{rcll}
x & = & 107_{(10)} & \mapsto \quad 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 1\ 4\ + \\
c & = & & \\
r & = & & \_\_\_\_\_
\end{array}
$$

**Example ($b = 2$)**

$$
\begin{array}{rcll}
x & = & 107_{(10)} & \mapsto \quad 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & \\
r & = & & \_\_\_\_\_
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.

Notes:

## Part 1: addition in theory (1)

- Concept:

**Example ($b = 10$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 1\ 4\ + \\
c & = & & \qquad\ 0 \\
r & = & & \hline
\end{array}
$$

**Example ($b = 2$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & \\
r & = & & \hline
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.

Notes:



## Part 1: addition in theory (1)

- Concept:

**Example ($b = 10$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 1\ 4\ + \\
c & = & & \qquad\ 1\ 0 \\
r & = & & \qquad\ \ 1 \\
\end{array}
$$

**Example ($b = 2$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & \\
r & = & & \hline
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.

Notes:

© Daniel Page ⟨cs@cs.bris.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

## Part 1: addition in theory (1)

- Concept:

**Example ($b = 10$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 1\ 4\ + \\
c & = & & \quad 0\ 1\ 0 \\
r & = & & \quad 2\ 1
\end{array}
$$

**Example ($b = 2$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & \\
r & = & &
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.

Notes:



## Part 1: addition in theory (1)

- Concept:

**Example ($b = 10$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 1\ 4\ + \\
c & = & & \quad 0\ 0\ 1\ 0 \\
r & = & 121_{(10)} & \mapsto \quad 1\ 2\ 1
\end{array}
$$

**Example ($b = 2$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & \\
r & = & &
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.

Notes:

© Daniel Page ⟨cs@cs.bris.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

## Part 1: addition in theory (1)

- Concept:

**Example ($b = 10$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 1\ 4\ + \\
c & = & & \quad 0\ 0\ 1\ 0 \\
r & = & 121_{(10)} & \mapsto \quad 1\ 2\ 1
\end{array}
$$

**Example ($b = 2$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & \quad\quad\quad\quad\quad\quad\quad\ 0 \\
r & = & &
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.

Notes:



## Part 1: addition in theory (1)

- Concept:

**Example ($b = 10$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 1\ 4\ + \\
c & = & & \quad 0\ 0\ 1\ 0 \\
r & = & 121_{(10)} & \mapsto \quad 1\ 2\ 1
\end{array}
$$

**Example ($b = 2$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & \quad\quad\quad\quad\quad\quad\quad\ 0\ 0 \\
r & = & &
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.

Notes:

© Daniel Page ⟨cs@cs.bris.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

## Part 1: addition in theory (1)

- Concept:

**Example ($b = 10$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 1\ 4\ + \\
c & = & & \quad 0\ 0\ 1\ 0 \\
r & = & 121_{(10)} & \mapsto \quad 1\ 2\ 1
\end{array}
$$

**Example ($b = 2$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & \quad\quad\quad\quad\quad\ 1\ 0\ 0 \\
r & = & & \quad\quad\quad\quad\quad\ 0\ 1
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.

Notes:



## Part 1: addition in theory (1)

- Concept:

**Example ($b = 10$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 1\ 4\ + \\
c & = & & \quad 0\ 0\ 1\ 0 \\
r & = & 121_{(10)} & \mapsto \quad 1\ 2\ 1
\end{array}
$$

**Example ($b = 2$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & \quad\quad\quad\quad\quad\ 1\ 1\ 0\ 0 \\
r & = & & \quad\quad\quad\quad\quad\ 0\ 0\ 1
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.

Notes:

© Daniel Page ⟨cs@phoo.cs.bris.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

## Part 1: addition in theory (1)

- Concept:

**Example ($b = 10$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 1\ 4\ + \\
c & = & & \quad 0\ 0\ 1\ 0 \\
r & = & 121_{(10)} & \mapsto \quad 1\ 2\ 1
\end{array}
$$

**Example ($b = 2$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & \quad\ \ 1\ 1\ 1\ 0\ 0 \\
r & = & & \quad\ \ 1\ 0\ 0\ 1
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.



Notes:

## Part 1: addition in theory (1)

- Concept:

**Example ($b = 10$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 1\ 4\ + \\
c & = & & \quad 0\ 0\ 1\ 0 \\
r & = & 121_{(10)} & \mapsto \quad 1\ 2\ 1
\end{array}
$$

**Example ($b = 2$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & \quad\ \ 0\ 1\ 1\ 1\ 0\ 0 \\
r & = & & \quad\ \ 1\ 1\ 0\ 0\ 1
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.

© Daniel Page ⟨cs@cs.bris.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

Notes:

## Part 1: addition in theory (1)

- Concept:

**Example ($b = 10$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 1\ 4\ + \\
c & = & & \overline{0\ 0\ 1\ 0} \\
r & = & 121_{(10)} & \mapsto \quad 1\ 2\ 1
\end{array}
$$

**Example ($b = 2$)**

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & \quad\ \overline{0\ 0\ 1\ 1\ 1\ 0\ 0} \\
r & = & & \quad\ 1\ 1\ 1\ 0\ 0\ 1
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.

Notes:



Part 1: addition in theory (1)

- Concept:

Example ($b = 10$)

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 1\ 4\ + \\
c & = & & \overline{0\ 0\ 1\ 0} \\
r & = & 121_{(10)} & \mapsto \quad 1\ 2\ 1
\end{array}
$$

Example ($b = 2$)

$$
\begin{array}{lcll}
x & = & 107_{(10)} & \mapsto \quad 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto \quad 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & \quad\ \overline{0\ 0\ 0\ 1\ 1\ 1\ 0\ 0} \\
r & = & & \quad\ 1\ 1\ 1\ 1\ 0\ 0\ 1
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.

Notes:

© Daniel Page ⟨cs@phoo.cs.bris.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

## Part 1: addition in theory (1)

- Concept:

**Example ($b = 10$)**

$$
\begin{array}{lcrcr}
x & = & 107_{(10)} & \mapsto & 1\ 0\ 7 \\
y & = & 14_{(10)} & \mapsto & 0\ 1\ 4\ + \\
c & = & & & 0\ 0\ 1\ 0 \\
r & = & 121_{(10)} & \mapsto & 1\ 2\ 1
\end{array}
$$

**Example ($b = 2$)**

$$
\begin{array}{lcrcr}
x & = & 107_{(10)} & \mapsto & 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
y & = & 14_{(10)} & \mapsto & 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ + \\
c & = & & & 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0 \\
r & = & 121_{(10)} & \mapsto & 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1
\end{array}
$$

$\therefore$

1. this process matches our understanding of manual, "school-book" addition, *and*
2. the same process applies, irrespective of $b$.

Notes:



## Part 2: addition in practice: an algorithm (1)

**Algorithm**

**Input:** Two unsigned, $n$-digit, base-$b$ integers $x$ and $y$, and a 1-digit carry-in $ci \in \{0, 1\}$
**Output:** An unsigned, $n$-digit, base-$b$ integer $r = x + y$, and a 1-digit carry-out $co \in \{0, 1\}$

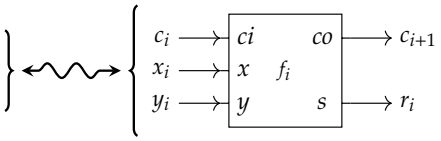
```
1 r ← 0, c_0 ← ci
2 for i = 0 upto n − 1 step +1 do
3 |   r_i ← (x_i + y_i + c_i) mod b
4 |   if (x_i + y_i + c_i) < b then c_{i+1} ← 0 else c_{i+1} ← 1
5 end
6 co ← c_n
7 return r, co
```

Notes:

## Part 3: addition in practice: a circuit (1)

- Idea:
  1. for $b = 2$, it's clear from the algorithm that

```
3  r_i ← (x_i + y_i + c_i) mod b
4  if (x_i + y_i + c_i) < b then c_{i+1} ← 0 else c_{i+1} ← 1
```




  2. the loop body is therefore analagous to a Boolean function

$$f_i : \{0,1\}^3 \to \{0,1\}^2$$

specified by the following truth table

| $ci$ | $x$ | $y$ | $co$ | $s$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

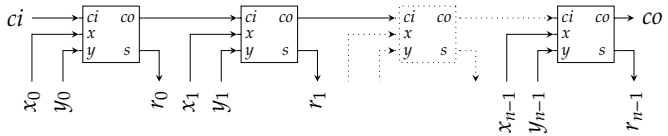


Notes:

## Part 3: addition in practice: a circuit (1)

- Idea:
  3. the loop bound is fixed, i.e., $n$ is some known constant, so we can unroll it to yield

**Circuit**




which, now read left-to-right, mirrors the algorithm:

- the $i$-th instance $f_i$ implements the $i$-th loop iteration,
- the connection, or **carry chain** between instances captures $c$,
- those instances are termed **full adder** cells,
- this combination of them is termed a **ripple-carry adder**.

© Daniel Page ⟨cs@bristol.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

Notes:

## Part 3: addition in practice: a circuit (2)

- Beware:
  - the magnitude of $r = x + y$ can exceed what we can represent via $\hat{r}$:

    $\hat{x}$ and $\hat{y}$ are *unsigned*, and there is a carry-out $\Rightarrow$ **carry** condition
    $\hat{x}$ and $\hat{y}$ are *signed*, and the sign of $\hat{r}$ is incorrect $\Rightarrow$ **overflow** condition

  - to cope, we typically
    1. detect the condition,
    2. potentially take some action (e.g., try to "fix" the result somehow),
    3. potentially signal the condition somehow (e.g., via a status register or some form of exception).

Notes:



## Part 3: addition in practice: a circuit (3)

**Example**

Consider use of an *unsigned* representation:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $x$ | = | $15_{(10)}$ | $\mapsto$ | | 1 | 1 | 1 | 1 | |
| $y$ | = | $1_{(10)}$ | $\mapsto$ | | 0 | 0 | 0 | 1 | + |
| $c$ | = | | | 1 | 1 | 1 | 1 | 0 | |
| $r$ | = | $0_{(10)}$ | $\mapsto$ | | 0 | 0 | 0 | 0 | |

Here, the carry-out indicates an error: the correct result $r = 16$ is too large for $n = 4$ bits.

- Note that
  1. detection:

     $c_n = co = 0 \Rightarrow$ no carry
     $c_n = co = 1 \Rightarrow$ carry

  2. action, e.g., **truncate** the result to $n$ bits.

Notes:

**Example**

Consider use of a *signed* representation:

$$\begin{array}{lcrcccccc} x & = & -1_{(10)} & \mapsto & 1 & 1 & 1 & 1 & \\ y & = & 1_{(10)} & \mapsto & 0 & 0 & 0 & 1 & + \\ c & = & & & 1 & 1 & 1 & 1 & 0 \\ r & = & 0_{(10)} & \mapsto & 0 & 0 & 0 & 0 & \end{array}$$

Irrespective of the carry-out, the signs of inputs and output make sense: there is no overflow, so $r = 0$ is correct.

**Example**

Consider use of a *signed* representation:

$$\begin{array}{lcrcccccc} x & = & 7_{(10)} & \mapsto & 0 & 1 & 1 & 1 & \\ y & = & 1_{(10)} & \mapsto & 0 & 0 & 0 & 1 & + \\ c & = & & & 0 & 1 & 1 & 1 & 0 \\ r & = & -8_{(10)} & \mapsto & 1 & 0 & 0 & 0 & \end{array}$$

Irrespective of the carry-out, the signs of inputs and output make no sense: there is an overflow, so $r = -8$ is incorrect.

- Note that
  1. detection:

$$\begin{array}{llll} x \text{ +ve} & y \text{ -ve} & & \Rightarrow \text{ no overflow} \\ x \text{ -ve} & y \text{ +ve} & & \Rightarrow \text{ no overflow} \\ x \text{ +ve} & y \text{ +ve} & r \text{ +ve} & \Rightarrow \text{ no overflow} \\ x \text{ +ve} & y \text{ +ve} & r \text{ -ve} & \Rightarrow \text{ overflow} \\ x \text{ -ve} & y \text{ -ve} & r \text{ +ve} & \Rightarrow \text{ overflow} \\ x \text{ -ve} & y \text{ -ve} & r \text{ -ve} & \Rightarrow \text{ no overflow} \end{array}$$

  2. action, e.g., **clamp** (or **saturate**) the result to the largest magnitude representable in $n$ bits.



Notes:

## Conclusions

- Take away points:
  1. Computer arithmetic is a broad, interesting (sub-)field:
     - 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.



Notes:

## 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 2: Addition/subtraction". 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.

Notes:



## References

[1] *Wikipedia: Computer Arithmetic*. URL: https://en.wikipedia.org/wiki/Category:Computer_arithmetic (see p. 57).

[2] D. Page. "Chapter 7: Arithmetic and logic". In: *A Practical Introduction to Computer Architecture*. 1st ed. Springer, 2009 (see p. 57).

[3] B. Parhami. "Part 2: Addition/subtraction". In: *Computer Arithmetic: Algorithms and Hardware Designs*. 1st ed. Oxford University Press, 2000 (see p. 57).

[4] W. Stallings. "Chapter 10: Computer arithmetic". In: *Computer Organisation and Architecture*. 9th ed. Prentice Hall, 2013 (see p. 57).

[5] A.S. Tanenbaum and T. Austin. "Section 3.2.2: Arithmetic circuits". In: *Structured Computer Organisation*. 6th ed. Prentice Hall, 2012 (see p. 57).

Notes: