Circuit Logic for Addition

DSC 315: Computer Organization & Operating Systems

Dr. Laltu Sardar

School of Data Science,
Indian Institute of Science Education and Research Thiruvananthapuram (IISER TVM)

9th January, 2026

Biased (Excess) Representations

Biased (Excess) Representation

- Values are stored with a fixed bias added to the true integer value
- Actual value is obtained by subtracting the bias from the stored value
- General form:

$$\text{Stored Value} = \text{Actual Value} + \text{Bias}$$

- Example using Excess-127:
  - Stored value 130 represents the actual value 3
- Use cases:
  - Exponent field in IEEE 754 floating-point representation
  - Simplifies comparison operations between signed values
  - Avoids explicit sign bit handling in hardware

Biased (Excess) Representation: Example with Binary

```
Code      Value
================
00000000  -127  <--- smallest negative value with 8 bits (-2^7)
00000001  -126
.....
01111000    -7
01111001    -6
01111010    -5
01111011    -4
01111100    -3
01111101    -2
01111110    -1
01111111     0
10000000     1
10000001     2
10000010     3
10000011     4
10000100     5
10000101     6
10000110     7
10000110     8
10000111     9
10001000    10
.....
11111111   128  <--- largest positive value with 8 bits (2^7-1)
```

- minimal negative value is represented by all-zeros

Addition for Unsigned numbers

Unsigned 4-bit Integers

- A 4-bit unsigned integer uses 4 bits to represent a number
- Possible values range from:

$$0 \text{ to } 15$$

- Binary representation:

$$0000_2 = 0, \quad 0001_2 = 1, \quad \ldots, \quad 1111_2 = 15$$

Rule for Addition

- Addition is performed bit by bit from right to left
- Each bit addition, say x_i, y_i, produces:
    - A sum bit s_i
    - A carry bit c_i to the next higher bit
- Only the lowest 4 bits of the result are kept, last carry bit is removed
- Its up to the software developer, whether it wants the carry bit, before next usage of addition circuit

Binary Addition Example (No Overflow)

$$
\begin{array}{ccccccl}
 & 0 & 1 & 1 & 1 & & <--c_i \\
 & & 0 & 1 & 0 & 1 & <--x_i \\
+ & & 0 & 0 & 1 & 1 & <--y_i \\
\hline
 & & 1 & 0 & 0 & 0 & <--s_i
\end{array}
$$

- $0101_2 = 5$
- $0011_2 = 3$
- Result: $1000_2 = 8$
- No overflow occurs

Binary Addition Example (Overflow)

$$\begin{array}{ccccc} & 1 & 1 & 0 & 1 \\ + & 0 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 \end{array}$$

- $1101_2 = 13$
- $0111_2 = 7$
- Binary sum is $10100_2 = 20$
- Only lower 4 bits are kept: $0100_2 = 4$
- Overflow occurs and the carry is discarded

Overflow and Underflow in Integer Arithmetic

Overflow

- Occurs when the result of an integer operation exceeds the **maximum representable value**
- Example: adding two large positive integers
- In fixed width integers, overflow may cause **wrap around**

Underflow

- Occurs when the result of an integer operation goes below the **minimum representable value**
- Example: subtracting a large integer from a smaller one
- In fixed width integers, underflow may also cause **wrap around**

Note

- Behavior depends on whether integers are **signed or unsigned**

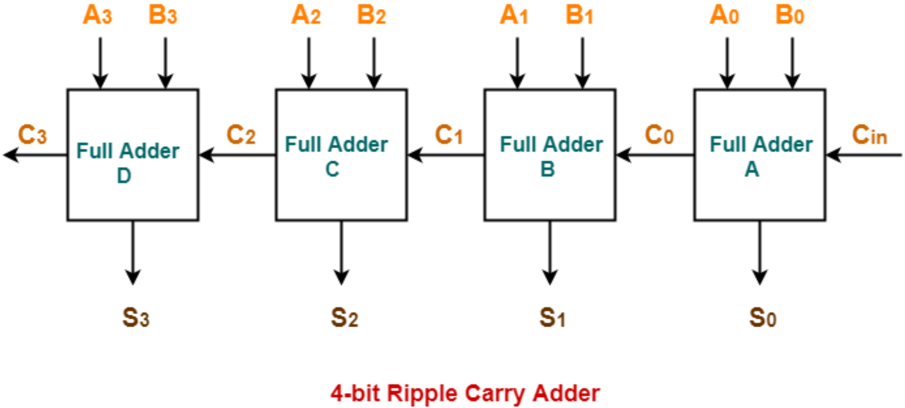
Overflow Rule

- Overflow occurs when the sum exceeds 15
- Carry out of the most significant bit is ignored
- Result wraps around modulo 16

$$\text{Result} = (a + b) \bmod 16$$

Homework: Find rule for 8-bit unsigned numbers How to detect it?

Circuit for Unsigned Binary number addition



Addition Rule in 2's Complement

- Numbers are represented in 2's complement form
- Addition is performed exactly like unsigned binary addition
- No special handling is required for the sign bit during addition
- The result is interpreted as a signed value
- The most significant bit represents the sign:
  - 0 for non-negative numbers
  - 1 for negative numbers

Overflow Detection in 2's Complement

- Overflow occurs only when adding two numbers with the same sign
- Adding two positive numbers gives a negative result
- Adding two negative numbers gives a positive result
- Overflow can be detected by comparing carries:

$$\text{Overflow} = c_n \oplus c_{n-1}$$

- Where:
  - c_{n-1} is the carry into the sign bit
  - c_n is the carry out of the sign bit
  - Why? `Homework: Check correctness`

Dr. Laltu Sardar, Assistant Professor, IISER Thiruvananthapuram
laltu.sardar@iisertvm.ac.in, laltu.sardar.crypto@gmail.com

Course webpage: https://laltu-sardar.github.io/courses/corgos_2026.html.