MODULE 2: Data Representation

Lecture 2.3
Binary Arithmetic

Prepared By:
- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering
Virginia Tech

Lecture 2.3 Objectives

- Perform addition (and subtraction) for two's complement numbers
- Determine when an arithmetic overflow occurs
- Describe simple high-level hardware components to realize a two's complement adder

Two’s Complement Addition

- Two’s complement addition requires the simple bit-wise addition of the corresponding bits
  - Need to consider the “carry-out” from the bit-wise addition
  - Ignore the final carry-out
- Subtraction is implemented as addition in the following manner
  - $a - b = a + (-b)$
  - *b* is first negated and then added to *a*

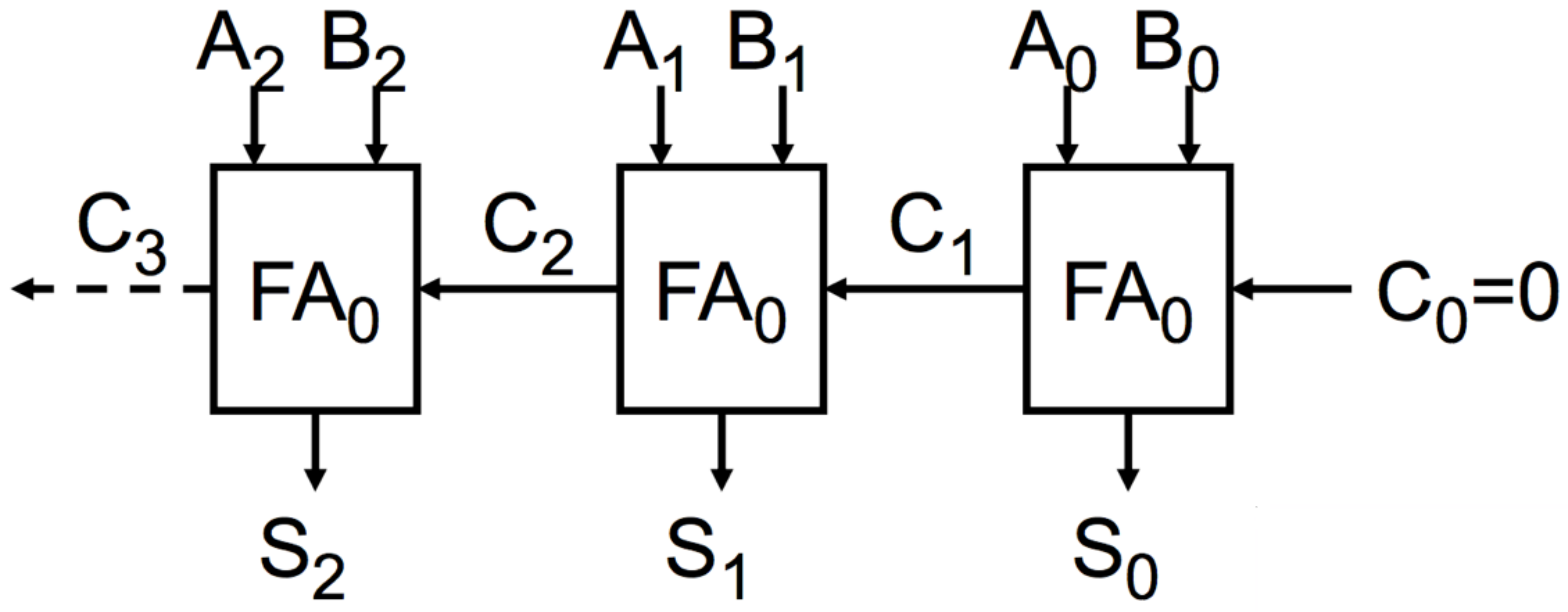
Addition Examples

```
           11100001   ← carry-out
  (+33)     00100001
+(-27)     +11100101
-------    ---------
  (+ 6)     00000110  ← sum
```

```
           00000111   ← carry-out
  (+33)     00100001
+(+ 7)     +00000111
-------    ---------
  (+40)     00101000  ← sum
```

Adder Hardware

- Simple adders are designed to mimic the scheme we've used for adding values
- A Full Adder (FA) is used to calculate the sum (S) and carry-out (C) for each column's bits (A and B)
- Need *n* Full Adders to add two *n*-bit numbers



CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Using two's complement addition (and assume 8 bits), add (+27) and (+9)

CHECK POINT

Answer:

```
 1      00011011   ←——   carry out
+27  =   00011011
+ 9  =   00001001
____     ________
+36  =   00100100
```

If you have any difficulties, please review the lecture video before continuing.

Overflow Example

- Consider the following example

```
              01111100    <- carry-out
( +60)        00111100
+( +70)      +01000110
-------      ---------
(+130)        10000010    <- sum
```

  - The sum for the binary version is -126, which is clearly incorrect
- Overflow has occurred
  - Both operands are positive, yet the result is negative

Overflow (1)

- Overflow results when insufficient range is available to represent the result
- Overflow cannot occur if the two operands have opposite signs since the magnitude of the sum must be smaller than the largest magnitude of the two operands
- Overflow can occur if the two operands have the same sign since the magnitude of the sum must be larger than than the magnitude of either operand
    - Indicated by a change in sign from the operands

Overflow (2)

Two methods for detecting overflow (they are equivalent, and you can use either one)

1. Overflow occurs if and only if the sign of both operands is the same and the sign of the sum is different from the sign of both operands

2. Overflow occurs if and only if the carry into the sign bit (the most significant bit) is different from the carry out of the sign bit

CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- State the two methods for detecting overflow

If you have any difficulties, please review the lecture video before continuing.

Summary

- Two’s complement addition requires just simple bit-wise addition of the operands
- Relatively easy to implement using Full Adders
- Overflow can occur when adding two positive numbers or two negative numbers

MODULE 2: Data Representation

Lecture 2.3
Binary Arithmetic

Prepared By:
- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering
Virginia Tech