Computer Architecture

Chapter 3-1

Arithmetic for computers

Addition & Subtraction

- **Just like in grade school (carry/borrow 1s)**

```
   0111          0111          0110
 + 0110        - 0110        - 0101
```

- **Two's complement operations easy**
  - subtraction using addition of negative numbers

```
   0111
 + 1010
```

- **Overflow (result too large for finite computer word):**
  - e.g., adding two n-bit numbers does not yield an n-bit number

```
   0111
 + 0001
 ------
   1000
```

note that overflow term is somewhat misleading, it does not mean a carry "overflowed"

Detecting Overflow

- **No overflow when adding a positive and a negative number**
- **No overflow when signs are the same for subtraction**
- **Overflow occurs when the value affects the sign:**
  - overflow when adding two positives yields a negative
  - or, adding two negatives gives a positive
  - or, subtract a negative from a positive and get a negative
  - or, subtract a positive from a negative and get a positive
- **Consider the operations A + B, and A – B**
  - Can overflow occur if B is 0 ?
  - Can overflow occur if A is 0 ?

Effects of Overflow

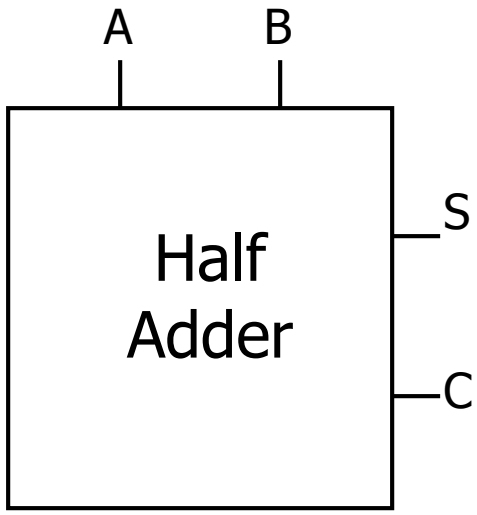
- **An exception (interrupt) occurs**
  - Save PC in exception program counter (EPC) register
    - `mfc0` (move from coprocessor reg) instruction can retrieve EPC value, to return after corrective action
  - Control jumps to predefined address for exception
- **Details based on software system / language**
  - example: flight control vs. homework assignment
- **Don't always want to detect overflow**
  - Overflow MIPS instructions: **`add, addi, sub`**
  - No overflow MIPS instructions: **`addu, addiu, subu`**

note: **`addiu`** *still sign-extends! (negative constant)*
note: **`sltu, sltiu`** *for unsigned comparisons*

Arithmetic for Multimedia

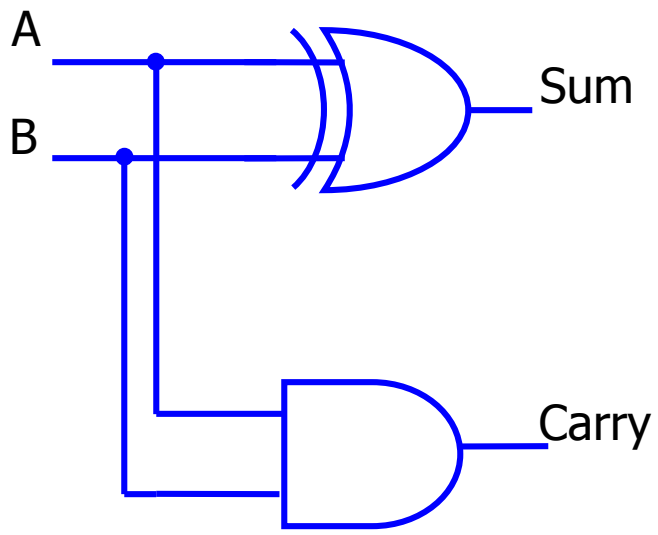
- **Graphics and media processing operates on vectors of 8-bit and 16-bit data**
  - Use 64-bit adder, with partitioned carry chain
    - Operate on 8×8-bit, 4×16-bit, or 2×32-bit vectors
  - SIMD (single-instruction, multiple-data)
- **Saturating operations**
  - On overflow, result is largest representable value
    - c.f. 2s-complement modulo arithmetic
  - E.g., clipping in audio, saturation in video

Half Adder (1-bit)



A	B	S(um)	C(arry)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half Adder (1-bit)

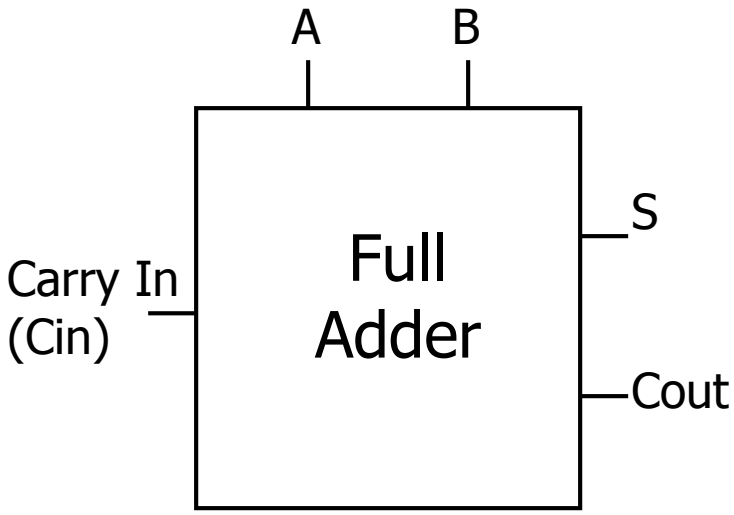


A	B	S(um)	C(arry)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = \overline{A}B + A\overline{B} = A \oplus B$$

$$C = AB$$

Full Adder



Cin	A	B	S(um)	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full Adder

Cin	A	B	S(um)	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Cin \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$S = Cin\,\overline{A}\,\overline{B} + \overline{Cin}\,\overline{A}B + CinAB + \overline{Cin}A\overline{B}$$
$$= Cin(\overline{A}\,\overline{B} + AB) + \overline{Cin}(\overline{A}B + A\overline{B})$$
$$= Cin(\overline{A \oplus B}) + \overline{Cin}(A \oplus B)$$
$$= Cin \oplus A \oplus B$$

Cin \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$Cout = CinB + CinA + AB$$

Or

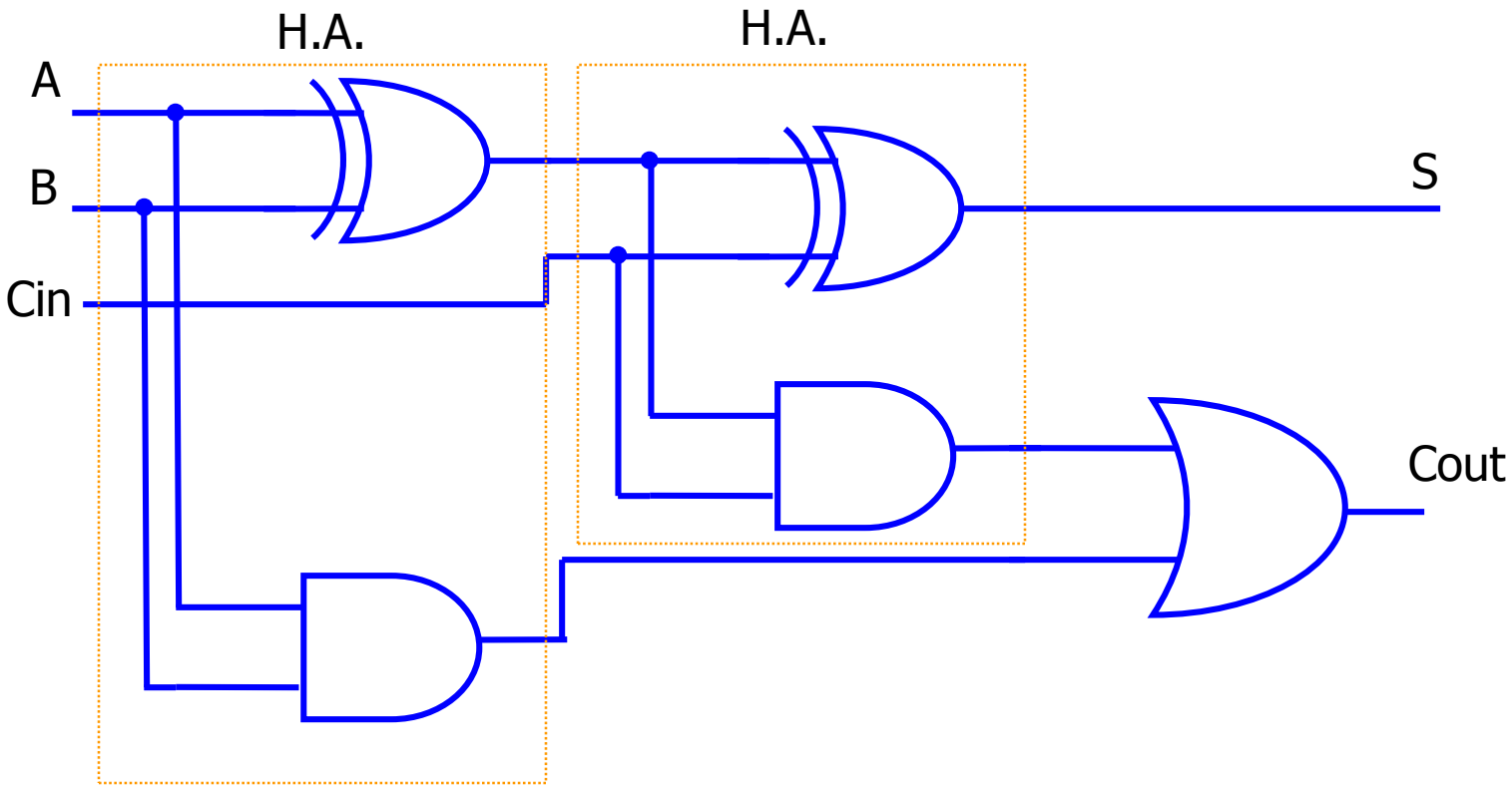
Cin \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$Cout = AB + Cin(\overline{A}B + A\overline{B}) = AB + Cin(A \oplus B)$$

Full Adder

$$S = Cin \oplus A \oplus B$$

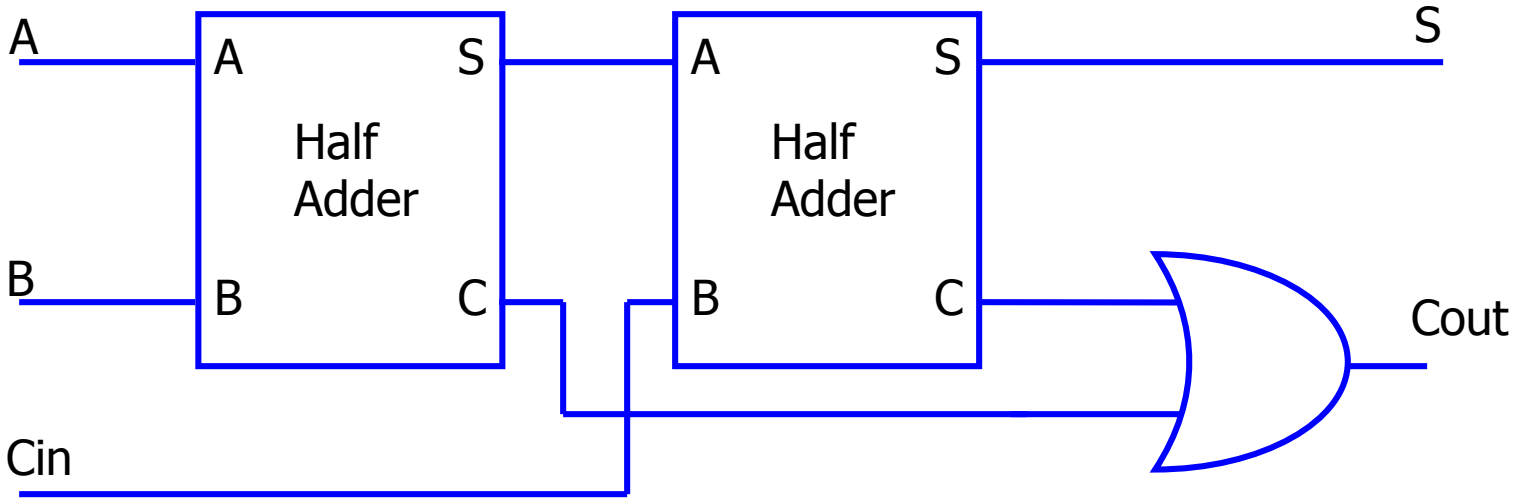
$$Cout = AB + Cin(A \oplus B)$$



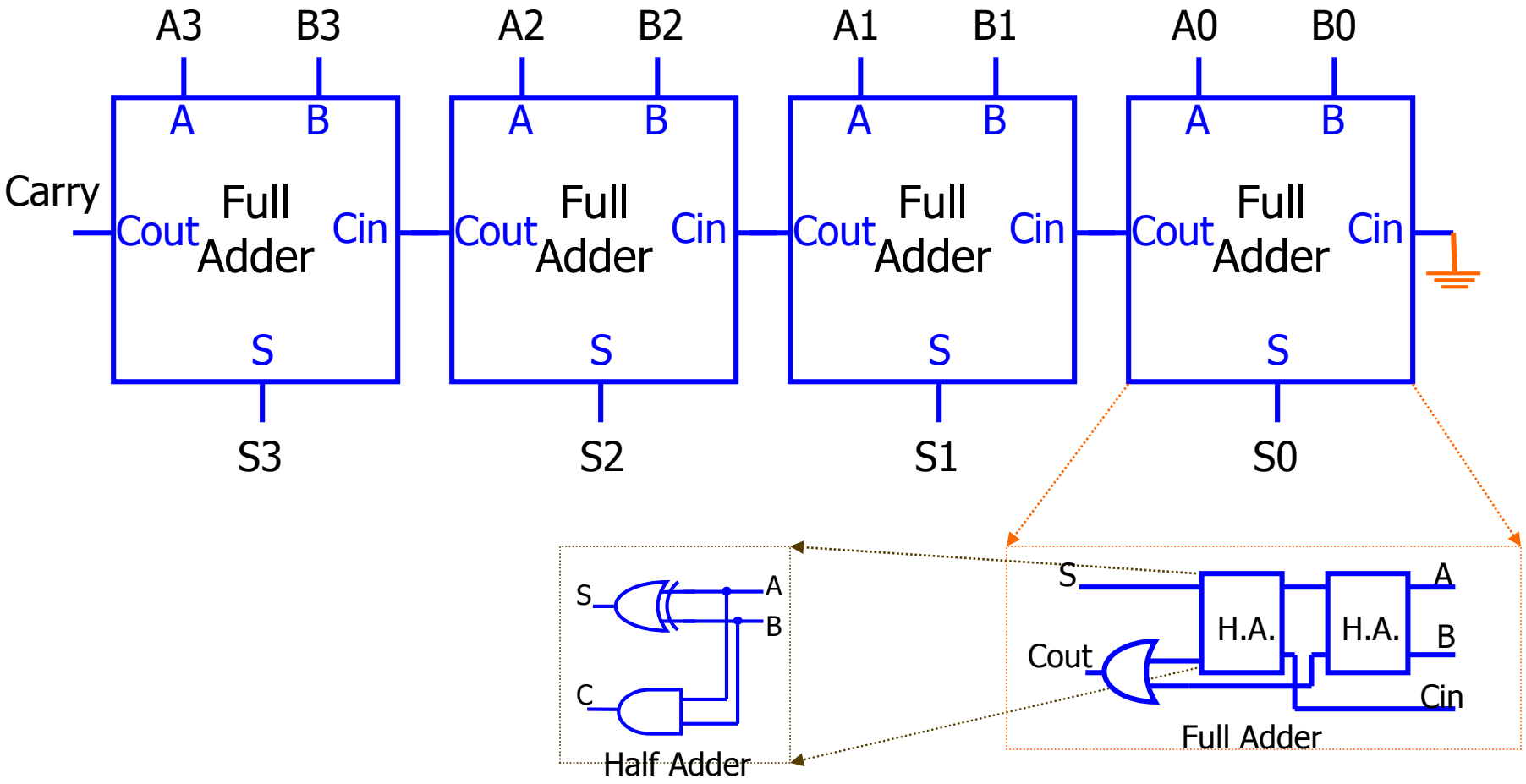
Full Adder

$$S = Cin \oplus A \oplus B$$

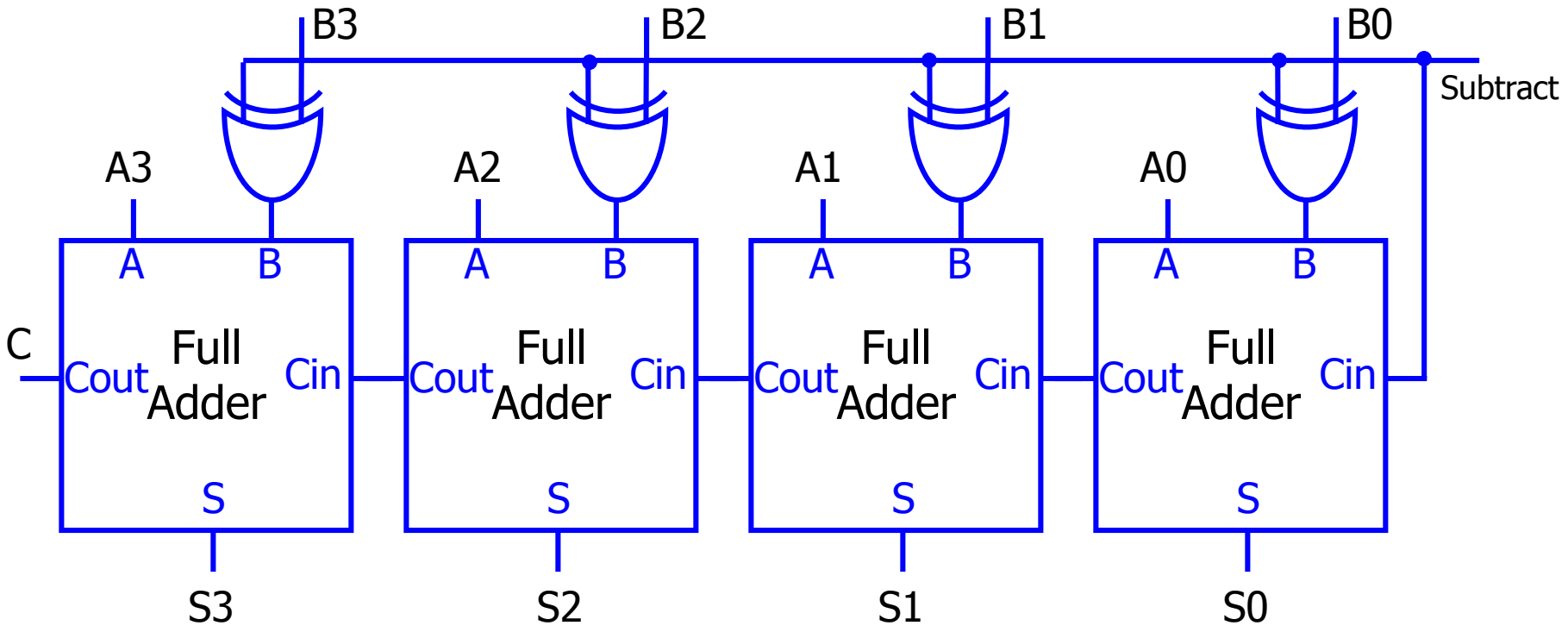
$$Cout = AB + Cin(A \oplus B)$$



4-bit Ripple Adder using Full Adder



Subtractor Design



- **A – B = A + (-B)**
  - Take 2's complement of B
  - Perform addition of A and 2's complement of B