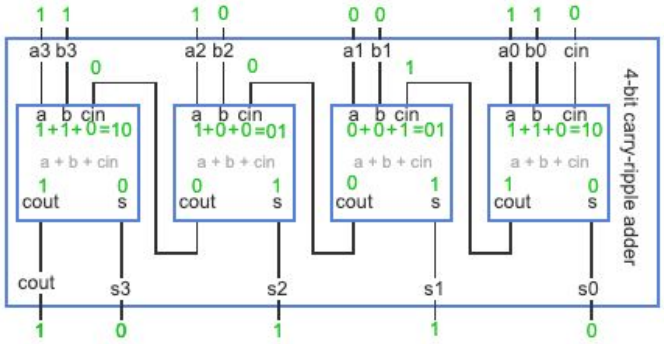
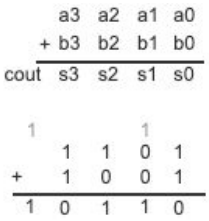
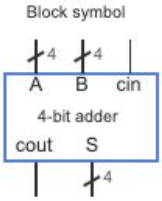
## 4.1 Adders

- Binary adders, 1+1 carries 1
- Computes A+B
    - A, B are N-bit numbers
    - Carry-ripple adder




-
    - N-bit carry-ripple adder adds two N-bit numbers
    - N is the size of each input
    - $C_{out}$ is the carry bit, 1 indicates a carry

A = 1111, B = 0001

- ○ cout = 0
  s3s2s1s0 = 0000
- ○ cout = 1
  s3s2s1s0 = 1111
- ◉ cout = 1
  s3s2s1s0 = 0000

-
    - **Full adder**
        - circuit that adds three bits and generates a sum and carry-out.
        - Half adder
            - circuit that adds two bits and generates a sum and carry-out bit
        - An N-bit carry-ripple adder is constructed with N full adders
- **Incrementer**
    - Adds 1 to a number (Adds 1 to input A)
    - Usually use half adder to implement incrementer but Full adder also works, but it requires larger circuits

## 4.2 Signed numbers in Binary

- Unsigned - positive only
- Signed - both positive and negative
- Use the left bit for the sign, 0 - positive, 1 - negative
- **Two's complement signed number representation**
  - For negative numbers, calculate its complement
    - Invert 1 and 0
    - Add 1 to the result
  - Add two numbers up, and ignore the carry bit
  - example

Complement: invert bits, add 1.

0011 (3) has complement: (0011)' + 1
1100 + 1
1101

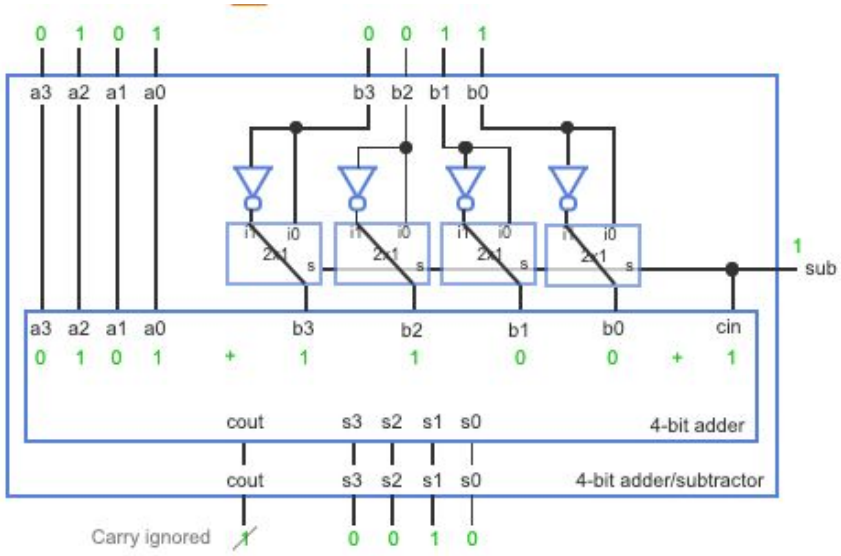
Replace with complement:

```
  0101  (5)          0101  (5)
- 0011  (3)        + 1101  (-3)
-------------      -------------
  0010  (2)        ✗0010  (2)   ignore carry
```

  -
  - More examples
    - 1011, negative because the left bit is 1
    - 1001, negative because the left bit is 1, in decimal
      - 0110 + 1 = 0111
      - Magnitude is 7, yields to -7 in decimal
    - -3 in eight-bit two's complement representation is
      - 00000011
      - 11111100 + 1
      - 11111101
  - Allows adder to deal with negative number additions
    - 3 + -4
    - 0011 + 1100
    - 1111 (two's complement representation)
    - 0000 + 1 = 0001, magnitude is 1, in decimal is -1
- **Overflow**
  - Adding two positives, or adding two negatives, may yield a value that can't be represented in the given number of bits
  - Example
    - 0001 + 1111 = 0000
    - Adding a negative number and a positive number cannot result in overflow
    - 0111 + 0110 = 1101 (-3)
    - 6 + 7 results overflow

## 4.3 Subtractors

- Computes A - B
- Using two’s complement representation for (-B)
- Because two's-complement representation performs subtraction by complementing and adding, a single adder circuit can perform either addition or subtraction, thus saving circuit size




sub = 0: addition

0011 (3) + 0100 (4)

0 111 (7)

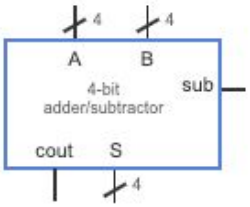
sub = 1: subtraction

0101 (5) - 0011 (3)

0101(5) + 1101 (-3)

0 010 (2)

Block symbol




-
-