Unsigned Signed

01 → 001

0 = +

1 = −

1's Compliment

Signed → 1010

0101

2's Compliment

(only convert if negative)

Signed → 1010 → 1011

0101 +1

0100 → 1011 → 1100

+1

Reversing: Copy 0 ant 1st 1 bit

Complimet everything else

1's

If both operands have the same sign:

- Add the magnitudes of the operands
- Use the sign of the operands

If the operands have different signs:

- Subtract the smaller magnitude from the larger magnitude
- Use the sign of the number with the larger magnitude.

This requires more complicated logic to compare and subtract.

As we will see with the other 2 representations, there is an easier way when you are dealing with binary numbers.

Table 5.1 Interpretation of four-bit signed integers.

$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	−0	−7	−8
1001	−1	−6	−7
1010	−2	−5	−6
1011	−3	−4	−5
1100	−4	−3	−4
1101	−5	−2	−3
1110	−6	−1	−2
1111	−7	−0	−1

1's

```
  (+5)      0101          (−5)      1010
+ (+2)    + 0010        + (+2)    + 0010
------    ------        ------    ------
  (+7)      0111          (−3)      1100

  (+5)      0101          (−5)      1010
+ (−2)    + 1101        + (−2)    + 1101
------    ------        ------    ------
  (+3)     10010          (−7)     10111
               1                       1
          ------                  ------
            0011                    1000
```

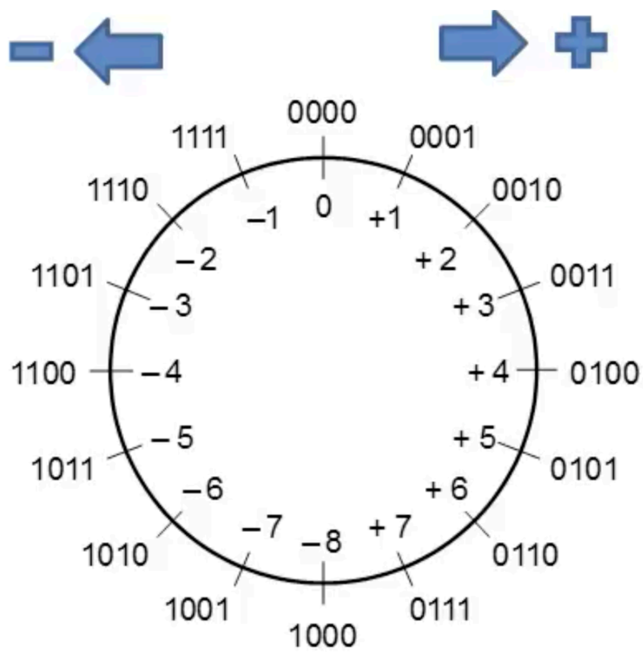
2's

```
  (+5)      0101          (−5)      1011
+ (+2)    + 0010        + (+2)    + 0010
------    ------        ------    ------
  (+7)      0111          (−3)      1101

  (+5)      0101          (−5)      1011
+ (−2)    + 1110        + (−2)    + 1110
------    ------        ------    ------
  (+3)     10011          (−7)     11001
           ↑                       ↑
           ignore                  ignore
```

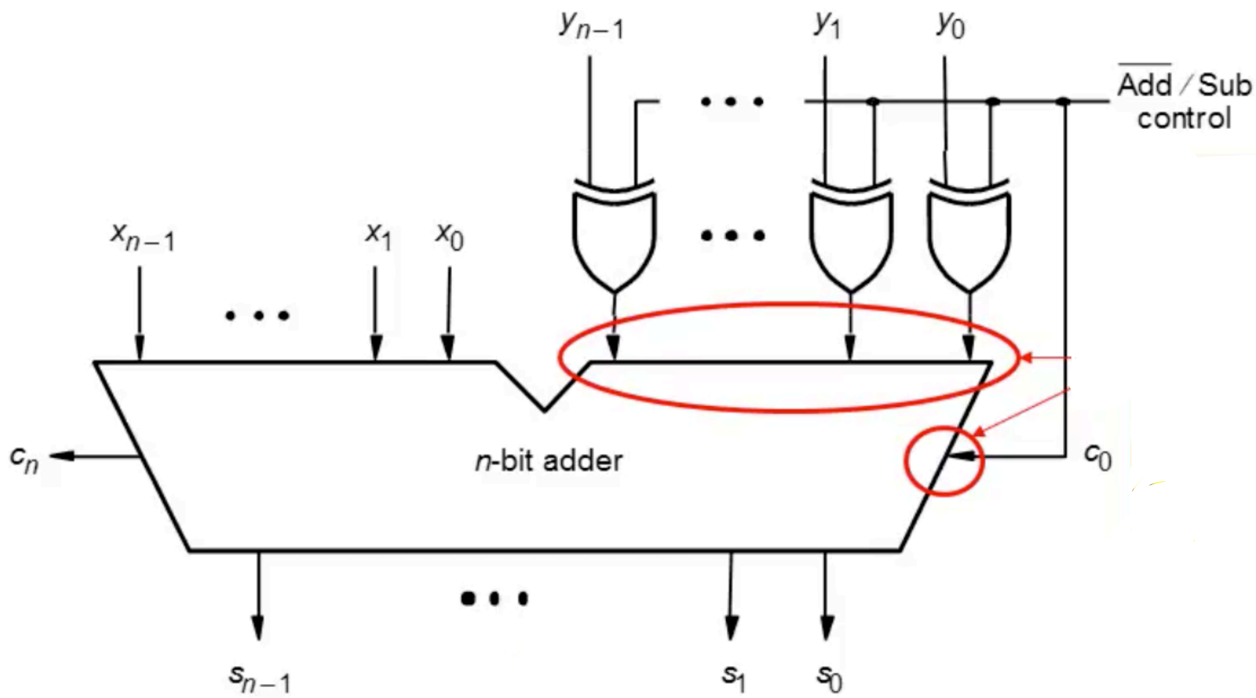
2's subtraction

```
  (+5)      0101             0101
− (+2)    − 0010    ⟹     + 1110
------    ------           ------
  (+3)                      10011
                            ↑
                            ignore

  (−5)      1011             1011
− (+2)    − 0010    ⟹     + 1110
------    ------           ------
  (−7)                      11001
                            ↑
                            ignore

  (+5)      0101             0101
− (−2)    − 1110    ⟹     + 0010
------    ------           ------
  (+7)                       0111

  (−5)      1011             1011
− (−2)    − 1110    ⟹     + 0010
------    ------           ------
  (−3)                       1101
```

take 2's

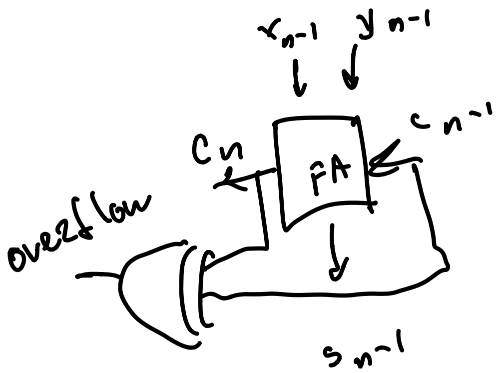


!Add / Sub - (bit) 0 - perform addition, 1 - subtraction



Result within: -2^{n-1} and $2^{n-1} - 1$

Overflow when $c_{n-1} \oplus c_n$



Critical-path delay - for longest path of gates

$2n$ - delay for n-bit adder

$2n+1$ - for subtractor

\+ overflow detector

$2n+2$ -