0x01 - Digital Logic and Arithmetic

ENGR 3410: Computer Architecture

Jon Tse

Fall 2020

Housekeeping

Feedback Side Channel - Google Docs

Lecture Pace Live Feedback

HW 1 and Lab 1A live!

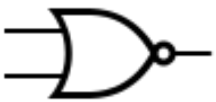
Lab 1 Individual/Group Submission

Review - Gates

NOT

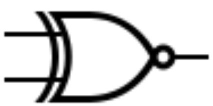
NAND

AND



NOR

OR

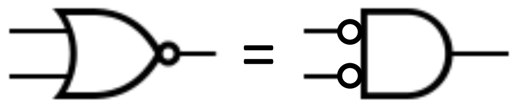


XNOR

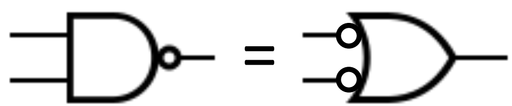
XOR

De Morgan's Law

$$\overline{X+Y} = \overline{X}\;\overline{Y}$$



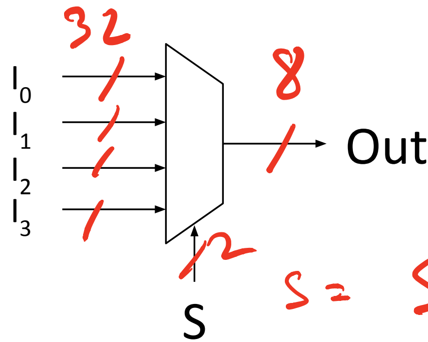
$$\overline{X\;Y} = \overline{X}+\overline{Y}$$



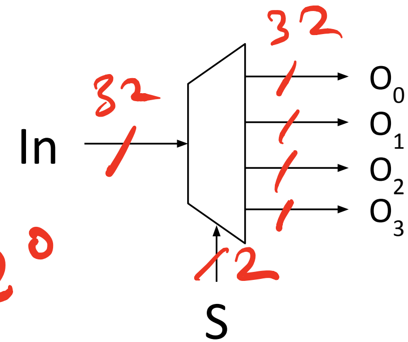
Review - Choice, Width

$w(S) = \log_2(\#\text{ inputs})$

Multiplexer (MUX)



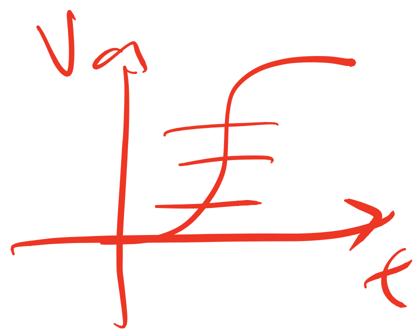
Demultiplexer (DEMUX)



$S = S_1 2^1 + S_0 2^0$

```
if (S==0)
    out = I0
elseif (S==1)
    out = I1
```

S_1	S_0	S
0	0	0
0	1	1
1	0	2
1	1	3



-plex from Latin: “plicare” or to fold, interweave

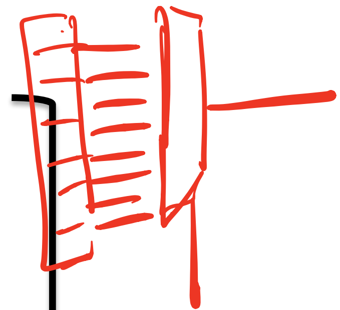
Today

- Wrap up logic gates, start to use in context
- Binary Math
- Intro to Arithmetic Circuits

Using Choice for Computation

S0	S1	S2	Out
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Select
$\log_2$(Depth)



Depth

S

$f(S) = Out_S$

$Out[S]$

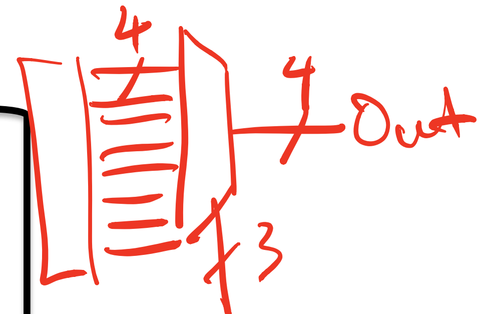
Look Up Tables

- A form of **R**ead **O**nly **M**emory
- Defined by Width and Depth
  - Width = How many bits per word
  - Depth = How many words available
- Constructed as Muxes with constant inputs
  - Use Select to *look up* the *table* entry

Look Up Table as a Truth Table

A0	A1	A2	Out0	Out1	Out2	Out3
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Stored Information



Depth 8

Select
$\log_2(\text{Depth})$

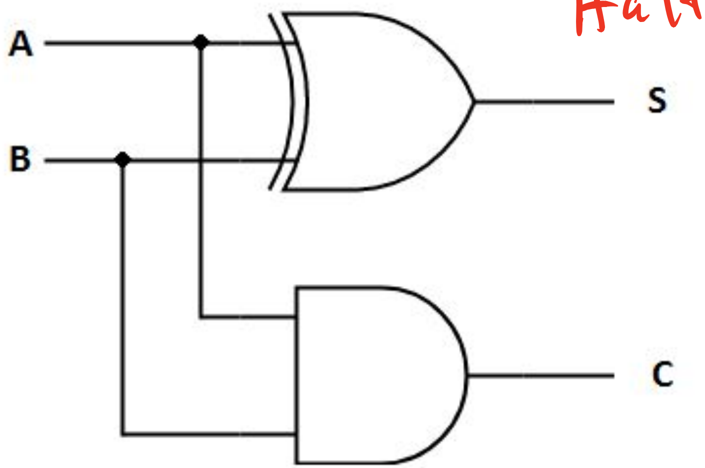
Width

nibble Out[8]

Out [8]

Combinational Logic

Half Adder



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$\begin{array}{r} 1 \\ +\ 1 \\ \hline 10 \end{array} \qquad \begin{array}{r} B \\ +\ A \\ \hline \end{array} \qquad \begin{array}{r} 0 \\ +\ 0 \\ \hline 0 \end{array}$$

C S

Math in Binary

Basically just like in grade school…

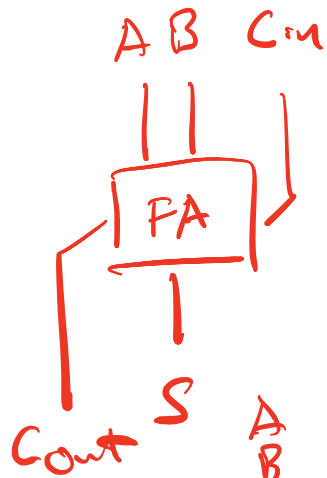
$$
\begin{array}{rcccc}
 & 0 & 0 & 1 & 0 \\
+ & 0 & 1 & 0 & 0 \\
\hline
 & 0 & 1 & 1 & 0
\end{array}
\qquad\qquad
\begin{array}{rccccc}
 & 1 & 1 & & & \\
 & & 1 & 1 & 1 & 0 \\
+ & & 1 & 1 & 0 & 0 \\
\hline
 & 1 & 1 & 0 & 1 & 0
\end{array}
$$

$$
\begin{array}{rcccc}
 & 0 & 0 & 1 & 0 \\
+ & 1 & 1 & 0 & 0 \\
\hline
 & 1 & 1 & 1 & 0
\end{array}
\qquad\qquad
\begin{array}{rccccc}
 & & 1 & & & \\
 & & 1 & 1 & 1 & 0 \\
+ & & 0 & 1 & 0 & 0 \\
\hline
 & 1 & 0 & 0 & 1 & 0
\end{array}
$$

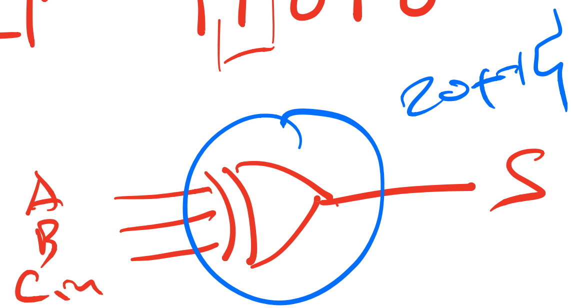
But available digits are SUPER important!

Full Adder

Input Operands	A, B
Carry In	C_{in}
Sum	S
Carry Out	C_{out}



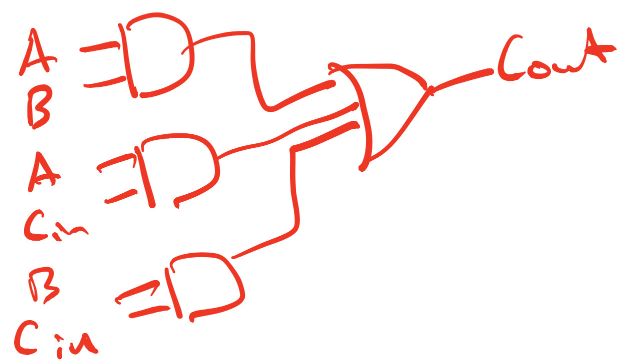

```
 1 1
   1 1 1 0
+  1 1 0 0
----------
 1 1 0 1 0
```



$$S = A \oplus B \oplus C_{in}$$

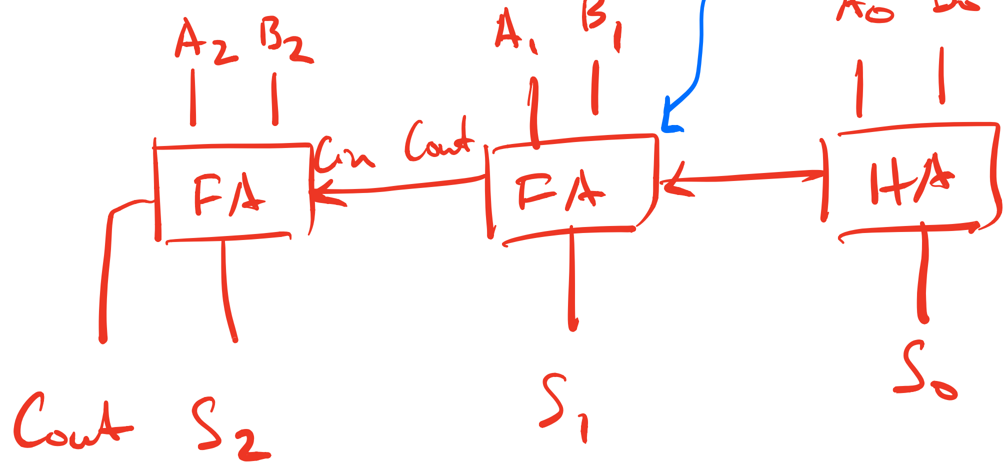
AND

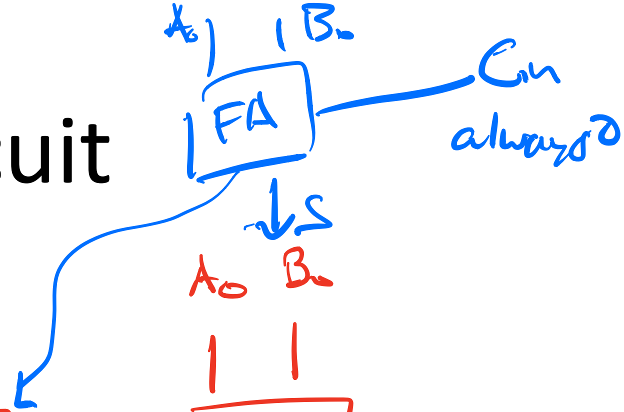
$$C_{out} = \underbrace{AB + AC_{in} + BC_{in}}_{\text{OR}}$$



Adder Circuit

A, B 3 bits





$$\begin{array}{r} {\scriptstyle 1} \\ 43 \\ +\,37 \\ \hline 80 \end{array}$$

no Cin

⇓

HA

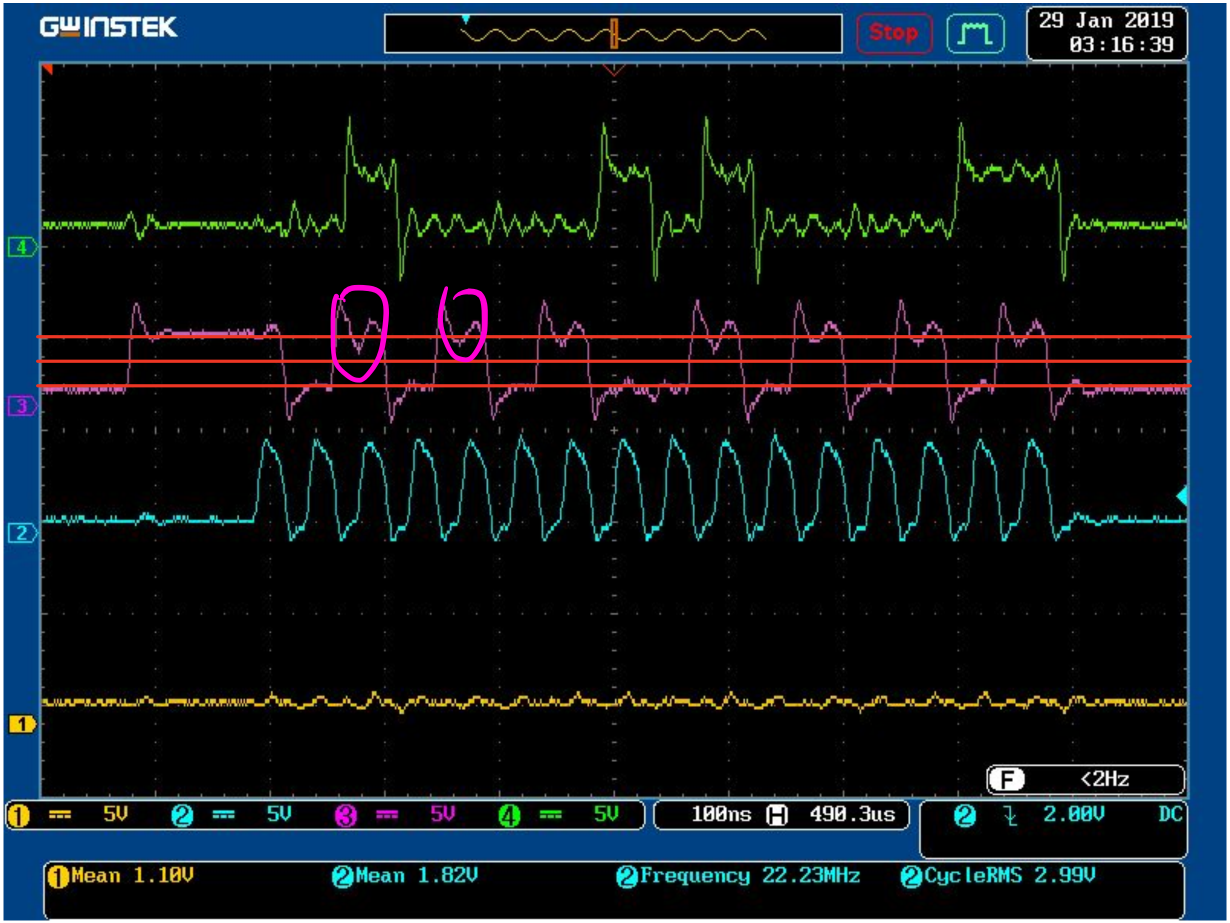
Number Systems

- Decimal is most common
- Base 2 – Computers
- Base 8 – Native Americans
- Base 12 – Nigeria, various mathematicians
- Base 20 – Mayan
- Base 24 – Kaugel (Papa New Guinea)
- Base 60 – Babylonians

Non Binary in Computers

- Base 3: Setun ternary computer (1958 in Moscow)
- Negabinary (Base -2): Invented in 1885, used in 50s in Poland
- Base 10: Financial computations (Binary Coded Decimal)
- Base 36: Storing case insensitive alphanumerics
- Base64: MIME (Multipurpose Internet Mail Extensions)
  - Encodes arbitrary information in printable characters

Why Binary?



https://e2e.ti.com/support/microcontrollers/c2000/f/171/t/767263?TMS320F28069M-SPI-is-inserting-an-extra-bit-in-front-of-read-transfer

Common Computer Bases

- Hexadecimal (Base 16)
  - D = 0123456789ABCDEF
  - 0xDEAD
  - hBEEF
  - $BA11_{16}$
- Octal (Base 8)
  - D = 01234567
  - 01234
  - o1234
  - 1234_8
- Decimal (Base 10)
  - D = 0123456789
  - 1234
  - d1234
  - 1234_{10}
- Binary (Base 2)
  - D = 01
  - 0b1010
  - b1010
  - 1010_2

base

digit

$\sum_{i=0}^{n-1} b^i d_i$

$s_1 2^1 + s_0 2^0$

Converting to Decimal

$b'1010 = 1\cdot 2^3+0\cdot 2^2+1\cdot 2^1+0\cdot 2^0 = 10$

$01010 = 1\cdot 8^3+0\cdot 8^2+1\cdot 8^1+0\cdot 8^0 = 520$

$1010 = 1\cdot 10^3+0\cdot 10^2+1\cdot 10^1+0\cdot 10^0 = 1010$

$0x1010 = 1\cdot 16^3+0\cdot 16^2+1\cdot 16^1+0\cdot 16^0 = 4112$

see next slide for examples

Radix / Base Conversion to Decimal

$$\sum_{i=0}^{n-1} b^i d_i = \text{decimal value}$$

digit

base

Hex → decimal

0x 1 2 3 A $b = 16$

d_3 d_2 d_1 d_0

$$16^3 \cdot 1 + 16^2 \cdot 2 + 16^1 \cdot 3 + 16^0 \cdot A$$

$$4096 \cdot 1 + 256 \cdot 2 + 16 \cdot 3 + 1 \cdot 10$$

$$4096 + 512 + 48 + 10$$

A = 10
B = 11
C = etc

0x123A = 4666 in decimal

Binary → decimal

b'0110

b' ⇒ indicates it's a binary #

b the base is 2 $b = 2$

$$\sum_{i=0}^{3} b^i d_i = 2^3 \cdot 0 + 2^2 \cdot 1 + 2^1 \cdot 1 + 2^0 \cdot 0$$

$$= 0 + 4 + 2 + 0$$

$$= 6$$

Radix Conversion

right to left

- General Conversion to base N
  - Digit = (Input) % (N)
  - Input = (Input) / (N)
  - Repeat until Input is exhausted (zero) writing digits 'right to left'.

```
def binary(input_num, n):
    bin = []
    while (input_num > 0):
        d = input_num % n
        i = input_num / n
        bin.insert(0,d)
        input_num = i
    return bin

print binary(4, 2)
```

- Or, left to right.
  - Digit = (Input) / (N^ position)
  - Input = (input) % (N ^ position)

warning: requires you to know # of digits of output beforehand!

100 → hex

100 / 16 = 6

100 % 16 = 4

"6.25"

Radix Conversion

decimal. 100 → hex

- Convert d100 to hex
  - 1^{st} Digit = 100 % 16 = 4
  - Input = 100/16 = 6
  - 2^{nd} Digit = 6%16=6
  - Input = 6/16 = 0
    - DONE

```
def binary(input_num, n):
    bin = []
    while (input_num > 0):
        d = input_num % n
        i = input_num / n
        bin.insert(0,d)
        input_num = i
    return bin

print binary(4, 2)
```

- d100 = 0x64

Mixed Radix Conversion -- Time

H m S

8506 seconds = 02:21:46

remainder

8506 / 3600 = 02 hours

(8506 % 3600) / 60 = 21 minutes

(8506 % 3600) % 60 = 46 seconds

decimal → hex

100 → 0x?

right → left method

100 % 16 = 4 , 1st digit is 4

100/16 = 6 .

6 % 16 = 6 , 2nd digit is 6

100 = 0x64

$16^1 \cdot 6 + 16^0 \cdot 4 = 100$ check our work ✓

decimal → binary

100 → b'?

100 % 2 = 0 — 0th digit

100/2 = 50

50 % 2 = 0 — 1st digit

50/2 = 25

25 % 2 = 1 — 2nd digit

25/2 = 12

12 % 2 = 0 — 3rd digit

12/2 = 6

6 % 2 = 0 — 4th digit

6/2 = 3

3 % 2 = 1 — 5th digit

3/2 = 1

1 % 2 = 1 — 6th digit

100 = b'110 0100

check work

$2^6 + 2^5 + 2^2$

64 + 32 + 4 = 100 ✓

Easy Radix Conversion

- When converting between bases that are powers of each other, the division operation becomes trivial.
  - 1 Hex digit = 4 binary digits (bits)
  - 3 Binary digits = 1 Octal digit (octets)
  - 1 R=r^n digit = n R=r digits

hex 4
oct 3
bin 1

Easy Radix Conversion Example

`b100111111011101011011`

Easy Radix Conversion Example

b10011111101110101011011

binary

b100 111 111 011 101 011 011

octal

o 4 7 7 3 5 3 3

$8 = 2^3$

To Hex

b 1 0011 1111 0111 0101 1011

hex

h 1 3 F 7 5 B

$16 = 2^4$

Negative Numbers

How can we represent negative numbers in a chosen radix in a way that computers can use?

r = radix
b = base
equivalent (basically)

Sign Magnitude

sign magnitude

-42

(r-1)'s Complement

9's complement

0 1 2 3 4 5 6 7 8 9

9 8 7 6 5 4 3 2 1 0

r's Complement

4
6 → 10

Negative Numbers

MSB LSB

n = 3			Sign Magnitude	1's Complement	2's Complement
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	2	2	2
0	1	1	3	3	3
1	0	0	-0	-3	-4
1	0	1	-1	-2	-3
1	1	0	-2	-1	-2
1	1	1	-3	-0	-1

$-x = (2^n - 1) - x$ $\qquad$ $-x = 2^n - x$

Complement $\qquad$ Complement

Which to Use?

Ignore carry out

Let's try: 1-1+1 = 1

Sign Magnitude	1's Complement	2's Complement
0 0 1 (+1)	0 0 1 (+1)	0 0 1 (+1)
+ 1 0 1 (-1)	+ 1 1 0 (-1)	+ 1 1 1 (-1)
1 1 0	1 1 1	1 0 0 0
+ 0 0 1 (+1)	+ 0 0 1 (+1)	+ 0 0 1 (+1)
1 1 1 (-3)	1 0 0 0 (0)	0 0 1 (1)

Converting to 2's Complement

R's complement

(R-1)'s complement +1

$$-x = 2^n - x$$

$$c(x) = 2^n - x$$

$$2^n = x+c(x)$$

step 1: input is 2 = b'010

step 2: invert ⇒ b'101

step 3: add 1

$$\begin{array}{r} 010 \\ +\ 110 \\ \hline 1000 \end{array}$$

$$\begin{array}{r} 101 \\ +\ 1 \\ \hline 110 \end{array} \quad (-2)$$

1's complement

010 = 2 (1's)

101 = -2 (1's)

$$\begin{array}{r} 010 \\ +\ 101 \\ \hline 111 \end{array} \quad (7)$$

$2^3 = 8$

$$\begin{array}{r} 111 \\ +\ 1 \\ \hline 1000 \end{array}$$

Addition in 2's Complement

2^3 2^2 2^1 2^0

```
  0 0 1 0 (+2)
+ 0 1 0 0 (+4)
  0 1 1 0 (+6)
```

```
  1 1 1 0 (-2)
+ 1 1 0 0 (-4)
(1) 1 0 1 0 (-6)
```

1010
0101
0110

```
  0 0 1 0 (+2)
+ 1 1 0 0 (-4)
  1 1 1 0 (-2)
```

1110 $\xrightarrow{\text{inv}}$ 0001

add 1

```
  0001
+    1
    10 (2)
```

```
  1 1 1 0 (-2)
+ 0 1 0 0 (+4)
(1) 0 0 1 0 (2)
```

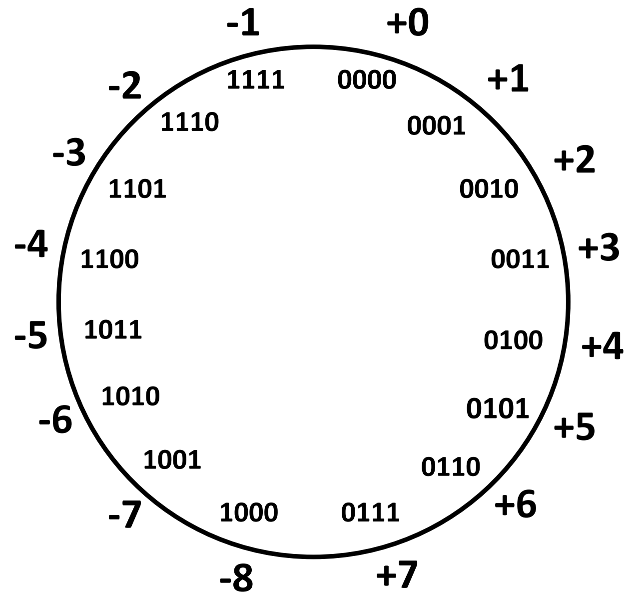
Subtraction in Two's Complement

- $A - B = A + (-B) = A + \overline{B} + 1$

- 0010 - 0110

- 1011 - 1001

- 1011 - 0001



Subtraction in Two's Complement

- $A - B = A + (-B) = \boxed{A + \overline{B} + 1}$

2 - 6

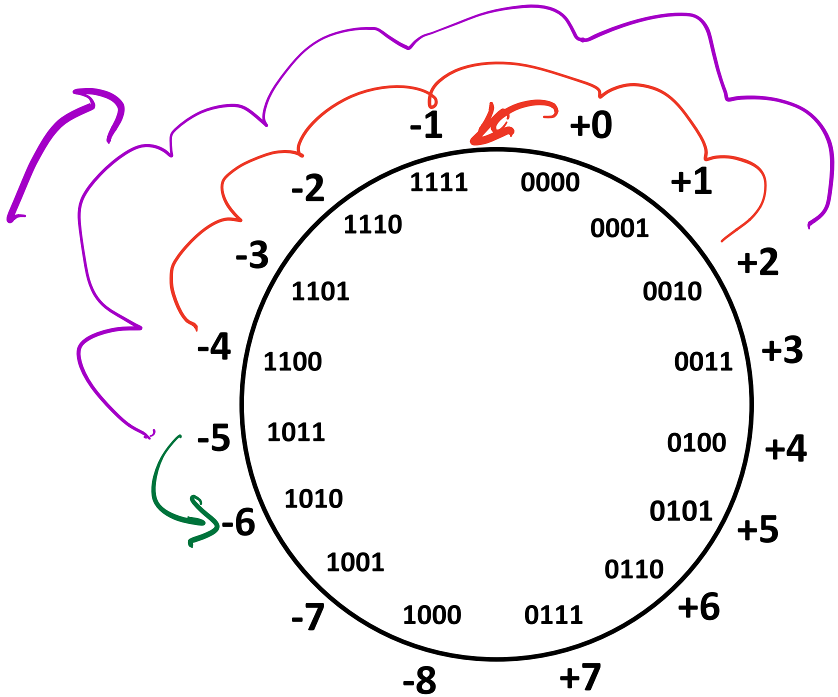
- 0010 - 0110
  0010 + 1001+1
  1100

-5 - -7

- 1011 - 1001
  1011 + 0110+1
  (1)0010

-5 - 1

- 1011 – 0001
  1011 + 1110+1
  (1)1010



Sign Extension in R’s Complement

- Positive numbers
  - Left pad with 0
- Negative Numbers
  - Left Pad with (R-1)s
- 2’s Complement
  - Left Duplicate the most significant (sign) bit.

Overflows in Two's Complement

Add two positive numbers to get a negative number

or two negative numbers to get a positive number