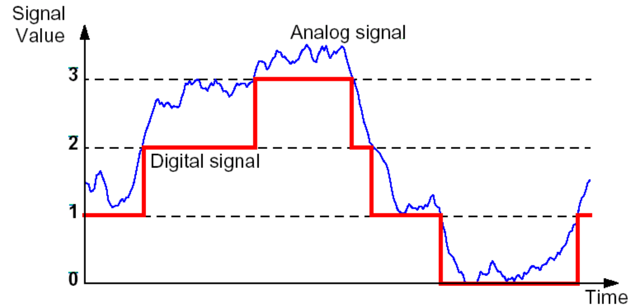
Digital Logic Design

Chapter 1
Numbers System

Recap: Analog Signal vs. Digital



Numbers

- Each number system is associated with a base or radix
  - The decimal number system is said to be of base or radix 10
- A number in *base r* contains r digits 0,1,2,...,r-1
  - Decimal (Base 10): 0,1,2,3,4,5,6,7,8,9
- Numbers are usually expressed in positional notation

$$A_{n-1}A_{n-2}\ldots A_1A_0.A_{-1}A_{-2}\ldots A_{-m+1}A_{-m}$$

- A number is expressed as a power series in *r* with the general form

$$A_{n-1}r^{n-1}+A_{n-2}r^{n-2}+\ldots+A_1r^1+A_0r^0+A_{-1}r^{-1}+A_{-2}r^{-2}$$
$$+\ldots+A_{-m+1}r^{-m+1}A_{-m}r^{-m}$$

Numbers

$$A_{n-1}A_{n-2}\ldots A_1A_0.A_{-1}A_{-2}\ldots A_{-m+1}A_{-m}$$

- The . is called the radix point
- A_{n-1} : most significant digit (msd)
- A_{-m} : least significant digit (lsd)

$$(724.5)_{10}=724.5=7\times10^2+2\times10^1+4\times10^0+5\times10^{-1}$$
$$1620.375=1\times10^3+6\times10^2+2\times10^1+3\times10^{-1}+7\times10^{-2}+5\times10^{-3}$$
$$(312.4)_5=3\times5^2+1\times5^1+2\times5^0+4\times5^{-1}=(82.8)_{10}$$

In addition to decimal, three number systems are important: Binary, Octal, and Hexadecimal

Decimal Number Representation

- Example: 90134 (base-10, used by Homo Sapien)
  - = 90000 + 0 + 100 + 30 + 4
  - = $9*10^4 + 0*10^3 + 1*10^2 + 3*10^1 + 4*10^0$
- How did we get it?

```
10 | 90134
10 | 9013  ........→ 4
10 | 901   ........→ 3
10 | 90    ........→ 1
     9     ........→ 0
```


Generic Number Representation

- 90134
  - = $9*10^4 + 0*10^3 + 1*10^2 + 3*10^1 + 4*10^0$
- $A_4\ A_3\ A_2\ A_1\ A_0$ for base-10 (or radix-10)
  - = $A_4*10^4 + A_3*10^3 + A_2*10^2 + A_1*10^1 + A_0*10^0$
  - (A is coefficient; b is base)
- Generalize for a given number **N** w/ base-b

  $\mathbf{N} = A_{n-1}\ A_{n-2} \ldots A_1\ A_0$

  $\mathbf{N} = A_{n-1}*b^{n-1} + A_{n-2}*b^{n-2} + \ldots + A_2*b^2 + A_0*b^0$

  **Note that $A < b$

Decimal Review

- Numbers consist of a bunch of digits, each with a weight:

1	6	2	.	3	7	5	Digits
100	10	1		1/10	1/100	1/1000	Weights

- The weights are powers of the base, which is 10

1	6	2	.	3	7	5	Digits
10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}	Weights

- To find the decimal value of a number, multiply each digit by its weight and sum the products

$(1 \times 10^2) + (6 \times 10^1) + (2 \times 10^0) + (3 \times 10^{-1}) + (7 \times 10^{-2}) + (5 \times 10^{-3}) = 162.375$

Counting numbers with base-b

0	10	20		90	100
1	11	21		91	101
2	12	22		92	102
3	13	23		93	103
4	14	24	.....	94	104
5	15	25		95	105
6	16	26		96	106
7	17	27		97	107
8	18	28		98	108
9	19	29		99	109

Base-10

0	10	20		70	100
1	11	21		71	101
2	12	22		72	102
3	13	23		73	103
4	14	24	.....	74	104
5	15	25		75	105
6	16	26		76	106
7	17	27		77	107

How about Base-8

How about base-2

0 10 100 1000
1 11 101 1001
110 1010
111 1011
1100
1101
1110
1111

How about base-2

0 10 100 1000
1 11 101 1001
110 1010
111 1011
1100
1101
1110
1111

How about base-2

0 = 0 **10 = 2** **100 = 4** **1000 = 8**
1 = 1 **11 = 3** **101 = 5** **1001 = 9**
110 = 6 **1010 = 10**
111 = 7 **1011 = 11**
1100 = 12
1101 = 13
1110 = 14
1111 = 15

Binary = Decimal

Information Representation: Binary Numbers

- Radix = 2; Digits A_i can only take one of two values (0 or 1)
  - It is customary to refer to *b*inary dig*it*s as *bit*s

	2	3	4	5	⋯	8	⋯	10	11	12	⋯	16
	0001	001	01	01		01		01	01	01		1
	0010	002	02	02		02		02	02	02		2
	0011	010	03	03		03		03	03	03		3
	0100	011	10	04		04		04	04	04		4
	0101	012	11	10		05		05	05	05		5
	0110	020	12	11		06		06	06	06		6
$(N)_b$	0111	021	13	12		07		07	07	07		7
	1000	022	20	13		10		08	08	08		8
	1001	100	21	14		11		09	09	09		9
	1010	101	22	20		12		10	0A	0A		A
	1011	102	23	21		13		11	10	0B		B
	1100	110	30	22		14		12	11	10		C
	1101	111	31	23		15		13	12	11		D
	1110	112	32	24		16		14	13	12		E
	1111	120	33	30		17		15	14	13		F

Number Examples with Different Bases

- Decimal (base-10)
  - $(982)_{10}$
- Binary (base-2)
  - $(01111010110)_2$
- Octal (base-8)
  - $(1726)_8$
- Hexadecimal (base-16)
  - $(3d6)_{16}$
- Others examples:
  - base-9 = $(1321)_9$
  - base-11 = $(813)_{11}$
  - base-17 = $(36d)_{17}$

Convert between different bases

- Convert a number base-x to base-y, e.g. $(0100111)_2$ to $(?)_6$
  - First, convert from base-x to base-10 if $x \neq 10$
  - Then convert from base-10 to base-y

$$0100111 = 0*2^6 + 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = 39$$

6 | 39
6 | 6 → 3
1 → 0

$\therefore (0100111)_2 = (103)_6$

Derive Numbers in Base-2

- Decimal (base-10)
  - $(25)_{10}$
- Binary (base-2)
  - $(11001)_2$
- Exercise

2 | 25
2 | 12 → 1
2 | 6 → 0
2 | 3 → 0
1 → 1

Converting Binary to Decimal

- For example, here is 1101.01 in binary:

1	1	0	1	.	0	1	Bits
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	Weights (in base 10)

$$2^{10}: K(kilo);\quad 2^{20}: M(mega);\quad 2^{30}: G(giga)$$

- The decimal value is:

$$(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) =$$

$$8 + 4 + 0 + 1 + 0 + 0.25 = 13.25$$

Converting Decimal to Binary

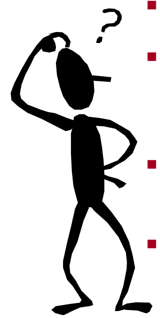
- To convert a decimal integer into binary, keep dividing by 2 until the quotient is 0. Collect the remainders in *reverse* order
- To convert a fraction, keep multiplying the fractional part by 2 until it becomes 0. Collect the integer parts in forward order
- Example: 162.375:

162 / 2 = 81 rem 0
81 / 2 = 40 rem 1
40 / 2 = 20 rem 0
20 / 2 = 10 rem 0
10 / 2 = 5 rem 0
5 / 2 = 2 rem 1
2 / 2 = 1 rem 0
1 / 2 = 0 rem 1

0.375 x 2 = 0.750
0.750 x 2 = 1.500
0.500 x 2 = 1.000

- So, $(162.375)_{10} = (10100010.011)_2$

Why does this work?



- This works for converting from decimal to *any* base
- Why? Think about converting 162.375 from decimal to decimal

162 / 10 = 16 rem 2
16 / 10 = 1 rem 6
1 / 10 = 0 rem 1

- Each division strips off the rightmost digit (the remainder). The quotient represents the remaining digits in the number
- Similarly, to convert fractions, each multiplication strips off the leftmost digit (the integer part). The fraction represents the remaining digits

0.375 x 10 = 3.750
0.750 x 10 = 7.500
0.500 x 10 = 5.000

Octal and Hexadecimal Numbers

- The octal number system: Base-8
  - Eight digits: 0,1,2,3,4,5,6,7

$$(127.4)_8 = 1\times 8^2 + 2\times 8^1 + 7\times 8^0 + 4\times 8^{-1} = (87.5)_{10}$$

- We use Base-16 (or Hex) a lot in computer world
  - Sixteen digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
  - Ex: A 32-bit address can be written as
    - `0xfe8a7d20` (`0x` is an abbreviation of Hex)
    - Or in binary form `1111_1110_1000_1010_0111_1101_0010_0000`

$$(B65F)_{16} = 11\times 16^3 + 6\times 16^2 + 5\times 16^1 + 15\times 16^{-0} = (46687)_{10}$$

- For our purposes, base-8 and base-16 are most useful as a "shorthand" notation for binary numbers

Numbers with Different Bases

Decimal	Binary	Octal	Hex
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

You can convert between base-10 base-8 and base-16 using techniques like the ones we just showed for converting between decimal and binary

Binary and Octal Conversions

- Converting from octal to binary: Replace each octal digit with its equivalent 3-bit binary sequence

$(673.12)_8$ = 6 7 3 . 1 2
= 110 111 011 . 001 010
= $(110111011.001010)_2$

- Converting from binary to octal: Make groups of 3 bits,starting from binary point. Add 0s to ends of number if needed. Convert each bit group to its corresponding octal digit.

10110100.001011_2 = 010 110 100 . 001 011_2
= 2 6 4 . 1 3_8

Octal	Binary
0	000
1	001
2	010
3	011

Octal	Binary
4	100
5	101
6	110
7	111

Binary and Hex Conversions

- Converting from hex to binary: Replace each hex digit with its equivalent 4-bit binary sequence

261.35_{16} = 2 6 1 . 3 5_{16}
= 0010 0110 0001 . 0011 0101_2

- Converting from binary to hex: Make groups of 4 bits, starting from the binary point. Add 0s to the ends of the number if needed. Convert each bit group to its corresponding hex digit

10110100.001011_2 = 1011 0100 . 0010 1100_2
= B 4 . 2 C_{16}

Hex	Binary
0	0000
1	0001
2	0010
3	0011

Hex	Binary
4	0100
5	0101
6	0110
7	0111

Hex	Binary
8	1000
9	1001
A	1010
B	1011

Hex	Binary
C	1100
D	1101
E	1110
F	1111

Information Representation (cont.)

- Various Codes used in Computer Industry
  - Number-only representation
    - BCD (4 bits per decimal number)
  - Alpha-numeric representation
    - ASCII (7 bits)
  - Unicode (16 bit)

Numeric Codes

Decimal digit	8 b_3	4 b_2	2 b_1	1 b_0	8	4	-2	-1	2	4	2	1	Excess-3			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	1	1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	1	0	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	0	1	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	1	1	1	0	1	1	1	0	0	0
6	0	1	1	0	1	0	1	0	1	1	0	0	1	0	0	1
7	0	1	1	1	1	0	0	1	1	1	0	1	1	0	1	0
8	1	0	0	0	1	0	0	0	1	1	1	0	1	0	1	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0

Negative Number Representation

- Options
  - Sign-magnitude
  - One's Complement
  - Two's Complement (we use this in this course)

Sign-magnitude

- Use the most significant bit (MSB) to indicate the sign
  - 0: positive, 1: negative
- Problem
  - Representing zeros?
  - Do not work in computation

+0	000
+1	001
+2	010
+3	011
−3	111
−2	110
−1	101
−0	100

One's Complement

- Complement (flip) each bit in a binary number
- Problem
  - Representing zeros?
  - Do not always work in computation
    - Ex: 111 + 001 = 000 → Incorrect !

+0	000
+1	001
+2	010
+3	011
−3	100
−2	101
−1	110
0	111

Two's Complement

- Complement (flip) each bit in a binary number and add 1, with overflow ignored
- Work in computation perfectly

3 011
↓ One's complement
100
↓ Add 1
-3 101

-3 101
↓ One's complement
010
↓ Add 1
3 011

Two's Complement

- Complement (flip) each bit in a binary number and adding 1, with overflow ignored
- Work in computation perfectly
- We will use it in this course !

0	000
+1	001
−1	111
+2	010
−2	110
+3	011
−3	101
??	100

100

One's complement

011

Add 1

100

The same 100 represents both 4 and -4 which is no good

Two's Complement

- Complement (flip) each bit in a binary number and adding 1, with overflow ignored
- Work in computation perfectly
- We will use it in this course !

0	000
+1	001
−1	111
+2	010
−2	110
+3	011
−3	101
−4	100

100

One's complement

011

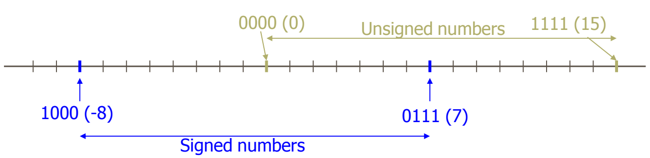
Add 1

100

MSB = 1 for negative Number, thus 100 represents -4

Range of Numbers

- An N-bit number
  - Unsigned: $0 \;..\; (2^N - 1)$
  - Signed: $-2^{N-1} \;..\; (2^{N-1} - 1)$
- Example: 4-bit



Binary Arithmetic

Sum	Carry	Difference	Borrow
$0+0=0$	0	$0-0=0$	0
$0+1=1$	0	$0-1=1$	1
$1+0=1$	0	$1-0=1$	0
$1+1=0$	1	$1-1=0$	0

Base 2

Carries: *10011 11*

1001.011 = $(9.375)_{10}$

1101.101 = $(13.625)_{10}$

10111.000 = $(23)_{10}$ = Sum

Borrow: *1 1*

	01011
Minuend	101000
Subtrahend	−011001
Difference	001111

Binary Arithmetic: Multiplication

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

```
   111.10
     10.1
 --------
    11110
   00000x
  11110xx
 --------
 10010110
```

$$(7.5)_{10} = (111.10)_2 \quad Q3.2$$
$$(2.5)_{10} = (10.1)_2 \quad Q2.1$$
$$(18.75)_{10} = (10010.110)_2 \quad Q5.3$$

Binary Computation

010001 (17=16+1)
001011 (11=8+2+1)

011100 (28=16+8+4)

Unsigned arithmetic
010001 (17=16+1)
101011 (43=32+8+2+1)

111100 (60=32+16+8+4)

Signed arithmetic (w/ 2's complement)
010001 (17=16+1)
101011 (-21: 2's complement=010101=21)

111100 (2's complement=000100=4, i.e. -4)

Binary Computation

The carry is discarded

Unsigned arithmetic
101111 (47)
011111 (31)

001110 (78?? Due to overflow, note that 78 cannot be represented by a 6-bit unsigned number)

The carry is discarded

Signed arithmetic (w/ 2's complement)
101111 (-17 since 2's complement=010001)
011111 (31)

001110 (14)