

1 Organization and Introduction

- The Art of managing complexity
 - Abstraction: Hiding details when they are not important
 - Discipline: Intentionally restricting your design choices to that you can work more productively at higher abstraction levels
 - The three -Y's
 - * Hierarchy: A system is divided into modules of smaller complexity
 - * Modularity: Having well defined functions and interfaces
 - * Regularity: Encouraging uniformity, so modules can be easily re-used
- Bit: **B**inary **d**igit

2 Binary Numbers

- Powers of two:

| | | |
|------------|-------------|------------------|
| $2^0 = 1$ | $2^5 = 32$ | $2^{10} = 1024$ |
| $2^1 = 2$ | $2^6 = 64$ | $2^{11} = 2048$ |
| $2^2 = 4$ | $2^7 = 128$ | $2^{12} = 4096$ |
| $2^3 = 8$ | $2^8 = 256$ | $2^{13} = 8192$ |
| $2^4 = 16$ | $2^9 = 512$ | $2^{14} = 16384$ |

- Binary to decimal conversion

$$\begin{aligned}
 10011_2 &= 2^4 \times 1 + 2^3 \times 0 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 1 \\
 &= 16 \times 1 + 8 \times 0 + 4 \times 0 + 2 \times 1 + 1 \times 1 \\
 &= 16 + 0 + 0 + 2 + 1 = 19_{10}
 \end{aligned}$$

- Convert decimal to binary (roughly). Example with 47_{10} to binary

| | | | | |
|------------|-------------------|-----|---|-----------------|
| $2^6 = 64$ | is $64 \leq 47$? | no | 0 | do nothing |
| $2^5 = 32$ | is $32 \leq 47$? | yes | 1 | $47-32=15$ |
| $2^4 = 16$ | is $16 \leq 15$? | no | 0 | do nothing |
| $2^3 = 8$ | is $8 \leq 15$? | yes | 1 | $15-8=7$ |
| $2^2 = 4$ | is $4 \leq 7$? | yes | 1 | $7-4=3$ |
| $2^1 = 2$ | is $2 \leq 3$? | yes | 1 | $3-2=1$ |
| $2^0 = 1$ | is $1 \leq 1$? | yes | 1 | $1-1=0$; done! |

$\Rightarrow 47_{10}$ to binary is 0101111_2

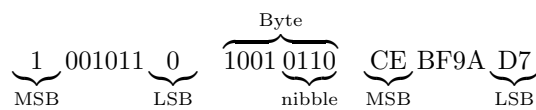
- Binary values and range

| | How many values? | Range? | Example: 3-digit number |
|---------------------------|------------------|-----------------|--|
| N -digit decimal number | 10^N | $[0, 10^N - 1]$ | $10^3 = 1000$ possible values, range: $[0, 999]$ |
| N -bit binary number | 2^N | $[0, 2^N - 1]$ | $2^3 = 8$ possible values, range: $[0, 7] = [000_2 \text{ to } 111_2]$ |

- Hexadecimal (Base-16) Numbers

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

- Bits, Bytes, Nibbles...



Where MSB=Most significant Bit and LSB=Least significant Bit

- Addition in base two works exactly the same as in base 10, using carries

- Overflow

- Digital systems operate on a fixed number of bits
- Addition overflows when the result is too big to fit in the available number of bits

- Signed Binary Numbers

- Sign/Magnitude Numbers

- * 1 sign bit, $N - 1$ magnitude bits
- * Sign bit is the most significant (left-most) bit
- * Example: 4-bit sign/mag repr. of ± 6 :
 - $+6 = \mathbf{0110}$
 - $-6 = \mathbf{1110}$
- * Range of an N -bit sign/magnitude number:

$$[-(2^{N-1} - 1), 2^{N-1} - 1]$$
- * Problems:
 - Addition doesn't work
 - Two representations of 0 (± 0): 1000 and 0000
 - Introduces complexity in the processor design

- One's Complement Numbers

- * A negative number is formed by reversing the bits of the positive number (MSB still indicates the sign of the integer)

| 2^7 | 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 | | One's Compl. | Unsigned |
|-------|-------|-------|-------|-------|-------|-------|-------|-----|--------------|----------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | = | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | = | 2 | 2 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = | 127 | 127 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = | -127 | 128 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | = | -126 | 129 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | = | -2 | 253 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | = | -1 | 254 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = | -0 | 255 |

* Range of n -bit number: $[-2^{n-1} - 1, 2^{n-1} - 1]$, 8 bits: $[-127, 127]$

* Addition: Done using binary addition with end-around carry. If there is a carry out of the MSB of the sum, this bit must be added to the LSB of the sum

– Two's Complement Numbers

* Don't have same problems as sign/magnitude numbers:

- addition works
- Single representation for 0

* Has advantages over one's complement:

- Has a single 0 representation
- Eliminates the end-around carry operation required in one's complement addition.

* A negative number is formed by reversing the bits of the positive number (MSB still indicates the sign of the integer) and adding 1:

| 2^7 | 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 | | Two's Compl. | Unsigned |
|-------|-------|-------|-------|-------|-------|-------|-------|-----|--------------|----------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | = | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | = | 2 | 2 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = | 127 | 127 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = | -128 | 128 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | = | -127 | 129 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | = | -3 | 253 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | = | -2 | 254 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = | -1 | 255 |

* Same as unsigned binary, but the most significant bit (MSB) has value of -2^{N-1}

- Most positive 4-bit number: 0111
- Most negative 4-bit number: 1000

* The most significant bit still indicates the sign (1=neg., 0=pos.)

* Range of an N -bit two's comp. number: $[-2^{N-1}, 2^{N-1} - 1]$, 8 bits: $[-128, 127]$

- Increasing bit width (assume from N to M , with $M > N$):

- Sign-extension
 - * Sign bit is copied into MSB
 - * Number value remains the same
 - * Give correct result for two's compl. numbers
 - * Example 1:
 - 4-bit representation of $3 = 0011$
 - 8-bit sign-extended value: **00000011**
 - * Example 2:
 - 4-bit representation of $-5 = 1011$
 - 8-bit sign-extended value: **11111011**
- Zero-extension
 - * Zeros are copied into MSB
 - * Value will change for negative numbers
 - * Example 1:
 - 4-bit value: $0011_2 = 3_{10}$
 - 8-bit zero-extended value: **00000011₂ = 3₁₀**
 - * Example 2:
 - 4-bit value: $1011_2 = -5_{10}$
 - 8-bit zero-extended value: **00001011₂ = 11₁₀**

3 Short Introduction to Electrical Engineering (EE Perspective)

- The goal of circuit design is to optimize:
 - Area: Net circuit area is proportional to the cost of the device
 - Speed/Throughput: We want circuits that work faster, or do more
 - Power/Energy
 - * Mobile devices need to work with a limited power supply
 - * High performance devices dissipate more than $100\text{W}/\text{cm}^2$
 - Design time
 - * Designers are expensive
 - * The competition will not wait for you
- (Frank's) Principles for engineering
 - Good engineers are lazy: They do not want to work unnecessarily, be creative
 - They know how to ask the question “why”? : take nothing for granted
 - Engineering is not a religion: Use what works best for you
 - Keep it simple and stupid: Engineers' job is to manage complexity