1.1 Digital Systems and Switching Circuits

Analog vs. Digital Information Processing

Analog: physical quantities (signals) vary continuously over a range, continuously valued, most "real world" signals are analog

Digital: discrete-valued, signals take on one of a finite set of possible values

Ex 1: Representation (time)

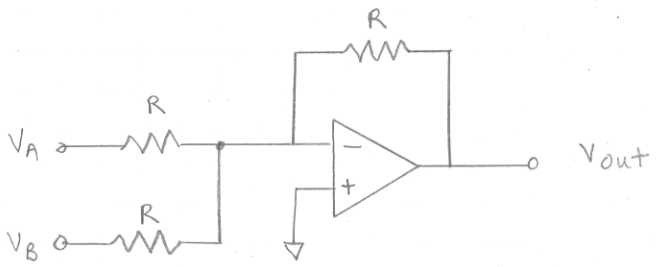
analog

vs. 5:05

digital

Ex 2: Computation (addition)

analog



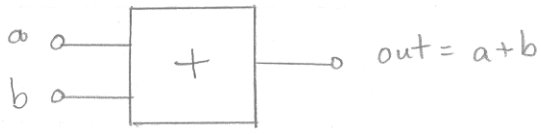
$V_{out} = -(V_A + V_B)$ ideal

$V_A = 4V \quad V_B = 2V$

$V_{out} = -6\,V \pm V_{error} \approx -6\,V$

V_{error} due to noise, nonideal circuits, etc.

digital



$a = 4 \quad b = 2$

out = 6 exactly!

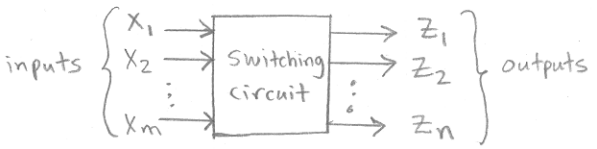
Trend is to replace analog with digital (music, video, telephony) because digital is arbitrarily accurate, stable (with respect to noise, temperature, manufacturing variations), and easier to design.

Digital System Design

Roth describes three parts: ① system design (180B, 170)
② logic design (180A)
③ circuit design (118, 116)

Logic Design: interconnecting gates and flip-flops to perform a specific function

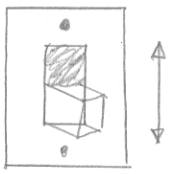
Switching Circuits



① Combinational Logic Circuits

Def: Outputs of a combinational block depend only on its present inputs.

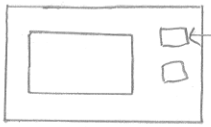
Ex: Light switch



② Sequential Logic Circuits

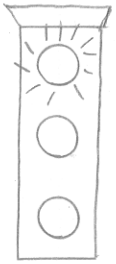
Def: Outputs of a sequential logic block depend on present inputs and past inputs, i.e. it has "memory"

Ex: Nearly any useful digital device typically is made up of combinational circuits and memory.



Start triggers internal timer or counter

Microwave Oven



Remembers previous color to go to next color.

Traffic Light

Ex: Digital processors like microprocessors, digital signal processors (DSPs), graphics processors usually have

1.) Memory – stores programs and data
2.) Datapath – process information (Arithmetic/Logic Unit (ALU), adder, multiplier)
3.) Controller – direct memory controller, disk drive controller
4.) Supporting circuitry

1.2 Number Systems and Conversion

Positional Notation: Each digit is multiplied by an appropriate power of the base number depending on its position in the number.

Decimal (base 10) digit values: $\{0, 1, 2, \ldots 9\}$

$\times 10^2$	$\times 10^1$	$\times 10^0$		$\times 10^{-1}$
1	2	3	.	4
$\times 100$	$\times 10$	$\times 1$		$\times 0.1$

$$123.4_{10} = 1 \times 100 + 2 \times 10 + 3 \times 1 + 4 \times 0.1$$

Binary (base 2) digit values: $\{0, 1\}$

$\times 2^2$	$\times 2^1$	$\times 2^0$		$\times 2^{-1}$
1	0	1	.	1
$\times 4$	$\times 2$	$\times 1$		$\times 0.5$

$$101.1_2 = 1 \times 4_{10} + 0 \times 2_{10} + 1 \times 1_{10} + 1 \times 0.5_{10} = 5.5_{10}$$

To convert to decimal, write the power series in terms of the decimal representation of the base and sum the series.

Octal (base 8) digit values: $\{0, 1, 2, \ldots, 7\}$

$\times 8^2$	$\times 8^1$	$\times 8^0$		$\times 8^{-1}$
1	2	3	.	4
$\times 64$	$\times 8$	$\times 1$		$\times 0.125$

$$123.4_8 = 1 \times 64_{10} + 2 \times 8_{10} + 3 \times 1_{10} + 4 \times 0.125_{10} = 83.5_{10}$$

Hexadecimal (base 16) digit values: $\{0, 1, \ldots, 9, A, B, C, D, E, F\}$
$A_{10}\ B_{11}\ C_{12}\ D_{13}\ E_{14}\ F_{15}$

$\times 16^2$	$\times 16^1$	$\times 16^0$		$\times 16^{-1}$
A	0	1	.	C
$\times 256$	$\times 16$	$\times 1$		$\times 1/16 =$

$$A01.C_{16} = A \times 256_{10} + 0 \times 16_{10} + 1 \times 1_{10} + C \times \frac{1}{16_{10}}$$
$$= 10_{10} \times 256_{10} + 0 \times 16_{10} + 1 \times 1_{10} + 12_{10} \times \frac{1}{16_{10}}$$
$$= 2561\,{}^{12}/_{16}{}_{10} = 2561.75_{10}$$

To convert from decimal to another base, use successive division by the new base for integers and successive multiplication by the new base for fractions.

For successive division, divide original number and each quotient by the new base until the quotient is 0. The remainders form the digits for the representation in the new base.

Ex: Convert 11_{10} to binary.

	Quotient	Remainder	
$2\overline{)11}$	5	1	← Least Significant Bit (furthest right)
$2\overline{)5}$	2	1	
$2\overline{)2}$	1	0	
$2\overline{)1}$	0	1	← Most Significant Bit (furthest left)

$$11_{10} = 1011_2$$

Alternative method: Start with largest power of 2 less than original number, subtract, and repeat with the difference until the difference is 0.

$$\underline{16}\ \ \underline{8}\ \ \underline{4}\ \ \underline{2}\ \cdot\ \underline{1}\ .\ \underline{1/2}\ \ \underline{1/4}$$
$$1\ \ 0\ \ 1\ \ 1$$

$11 - 8 = 3$
$3 - 2 = 1$
$1 - 1 = 0$

Check: $1 \times 8_{10} + 0 \times 4_{10} + 1 \times 2_{10} + 1 \times 1_{10} = 8_{10} + 2_{10} + 1_{10} = 11_{10}$ ✓

For successive multiplication, multiply original number and each fractional part by the new base until the fractional part is 0. The integer parts form the digits in the new base representation.

Ex: Convert 0.375_{10} to base 2.

0.375	0.750	0.500
$\times \quad 2$	$\times \quad 2$	$\times \quad 2$
$\boxed{0}.750$	$\boxed{1}.500$	$\boxed{1}.000$

$0 \qquad 1 \qquad 1 \quad \Rightarrow \quad 0.375_{10} = 0.011_2$

Ex: Convert 0.48046875_{10} to hexadecimal (base 16).

0.48046875	0.6875
$\times \quad 16$	$\times \quad 16$
$\boxed{7}.6875$	11.00

$11_{10} = \boxed{B}_{16} \quad \Rightarrow \quad 0.48046875_{10} = 0.7B_{16}$

Binary $\leftrightarrow$ Hexadecimal : 4 binary digits correspond to one hex digit.

Ex: $\underbrace{1010}\ \underbrace{0011} . \underbrace{0111}\ \underbrace{0010}_2$

$A \quad 3 . 7 \quad 2 \quad = A3.72_{16}$

Binary $\leftrightarrow$ Octal : 3 binary digits (bits) correspond to one octal digit.

Ex: $\underbrace{100}\ \underbrace{110} . \underbrace{011}$

$4 \quad 6 . 3 \quad = 46.3_8$

It is often easier to convert to binary first before converting to hexadecimal or octal!

1.3 Binary Arithmetic (Note: unsigned binary numbers)

Addition Same procedure as decimal addition.

Ex: 1 ← carry

$$\begin{array}{r} 1100_2 = 12_{10} \\ +\ 0110_2 = 6_{10} \\ \hline 10010_2 = 18_{10} \end{array}$$

Ex:

$$\begin{array}{r} {\scriptstyle 1\ 1}\ \ \\ 0110_2 = 6_{10} \\ 0011_2 = 3_{10} \\ +\ 1110_2 = 14_{10} \\ \hline 10111_2 = 23_{10} \end{array}$$

Subtraction ① Similar to decimal subtraction: borrow from more significant columns.

② Add the negative: $a - b = a + (-b)$

We'll concentrate on method ②, used much more commonly in hardware.

Multiplication Same procedure as base 10 multiplication.

Ex:

$$\begin{array}{r} 111_2 = 7_{10} \\ \times\ 101_2 = 5_{10} \\ \hline 111 \\ 000 \\ +\ 111 \\ \hline 100011_2 = 35_{10} \end{array}$$

"Partial products" are very simple in binary!

Division Similar to decimal division, but must subtract using subtraction methods above.

$$\begin{array}{r} 1_2 \\ 101_2 \overline{)\ 111_2} \\ 101_2 \\ \hline 10_2 \end{array}$$

Quotient is 1_2 with remainder 10_2.

1.4 Negative Numbers

① Sign and Magnitude Representation (similar to decimal)

Base 10: $+5$, -5 } magnitude (absolute value) = 5

Base 2: $+101 \Rightarrow 0101$, $-101 \Rightarrow 1101$ magnitude = $101_2 = 5_{10}$

"sign" bit, typically 0 = positive, 1 = negative

- Not very convenient representation for addition/subtraction hardware. ⊖
- Great for multiplication ⊕
- Two representations for zero: $+0 = 0000 \neq -0 = 1000$ ⊖

② Two's Complement (2's Complement)

Most common signed binary number representation.

- For an n-bit number N, 2's complement $N^* = -N = 2^n - N$

Ex: -3: $+3 = 0011$ $\quad -3 = 2^4 - 0011$
$= 10000 - 0011$
$= 1101$

- Same as the unsigned representation, except Most Significant Bit (MSB) corresponds to negative number.

only change ↓

$\times -2^3$	$\times 2^2$	$\times 2^1$	$\times 2^0$	.	$\times 2^{-1}$	$\times 2^{-2}$
1	0	1	1	.		
$\times -8$	$\times 4$	$\times 2$	$\times 1$		$\times 1/2$	$\times 1/4$

$1101_2 = 1 \times -8_{10} + 1 \times 4_{10} + 0 \times 2_{10} + 1 \times 1_{10} = -8 + 5 = -3$ ✓

- Easiest method to negate a number: "Flip (Invert) all bits and add 1"

Ex:

$+3 =$	0011	
	1100	flip bits
	1	add 1
$-3 =$	1101	
	0010	flip bits
	1	add 1
$+3 =$	0011	

- 2's complement features
  1.) MSB is sign bit, 0 = positive, 1 = negative
  2.) One representation for zero: all 0's, e.g. 0000
  3.) All 1's represents −1, e.g. 1111

  Ex: 111 = −4+2+1 = −1 11111 = −16+8+4+2+1 = −16+15 = −1

  4.) Extending (replicating) the sign bit (called sign extension) will never change a number and is often required/convenient:

  Ex: positive +2 = 010 = 0010 = 0000 0010

  negative −2 = 110 = 1110 = 1111 1110

  5.) Errors can result if operation overflows/underflows

③ One's Complement (1's Complement)
Good to know, but not often used.

- Negate by inverting all bits Ex: +3 = 011
  −3 = 100 flip bits
- Two representations for zero:
  +0 = 0000 ≠ −0 = 1111

1.5 Binary Codes

Format is a hybrid of base 10 (decimal) and base 2 (binary) that is simple for humans to read.

Numbers are made up of base 10 digits represented by base 2 digits.

Ex: $129_{10} \Rightarrow \underset{0001}{1}\ \underset{0010}{2}\ \underset{1001}{9} = 000100101001_{BCD}$

BCD ≡ Binary Coded Decimal

BTW, $129_{10} = 10000001_2$

Ex: $11001_2 = 25_{10} = 0010\ 0101_{BCD}$

Some old computers did math in BCD, today most use 2's complement binary and convert to BCD for input/output.