EE 224: INTRODUCTION TO DIGITAL CIRCUITS & COMPUTER DESIGN

Lecture 1: Introduction

3/29/2010

Avinash Kodi, kodi@ohio.edu

Agenda

2

- Go over the syllabus
- Introduction to Digital Systems

Why Digital Systems?

3

- Obvious reason
  - Implementation basis for all modern computing, communication and control devices (C^3)
    - Building larger things from smaller components
    - Faster, cheaper, more efficient than analog
- More reasons
  - Inherent parallelism in hardware
  - In addition to serial software design

Binary Representation

4

- Binary Information digiTS (BITS) – 0 or 1
- *n* bits specify M = 2^n different values
  - Or M values specified by $n = \log_2 M$ bits

The “Digital-Binary” World

5

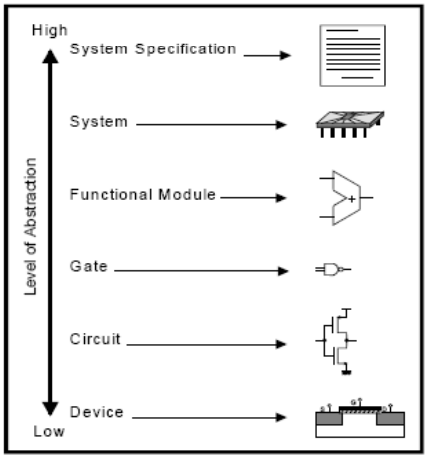
- Easy to process binary data
  - Digital logic where voltage < 1.5 V is a “0” and > 3. 5 V is a “1”
- Easy to store binary data
  - Fast disk and CD-ROM storage devices, cheap and small
  - Memory stick
- Easy to transmit and protect
  - Data protection well developed (parity)
- Easy to describe mathematically
  - Discrete mathematics (boolean algebra)

Design of Computers – Hierarchical Design Hardware

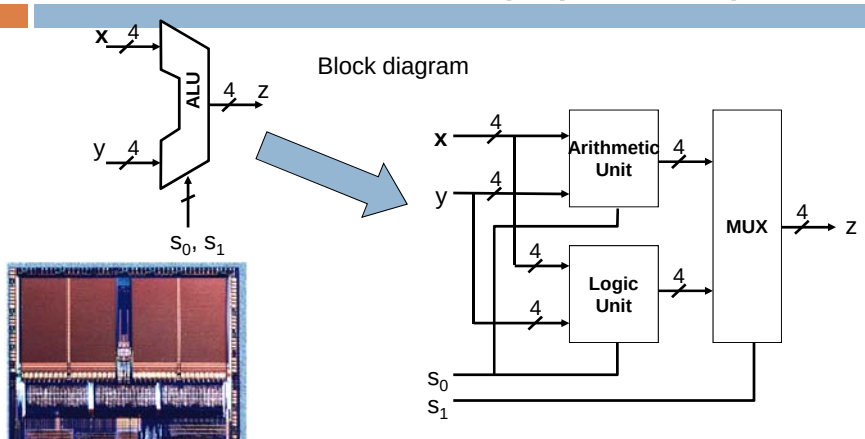
- Computer-based system design
  - Design in which a computer is interfaced with and/or embedded in a larger system with a specific function,
- System design
  - Design in which components such as the CPU. Memory, and I/O chips are interfaced to build a computer,
- Gate or Logic level
  - Design of higher-level components with chips containing logical gates
- Device level
  - Design of logic gates using transistors

Abstraction Levels

7

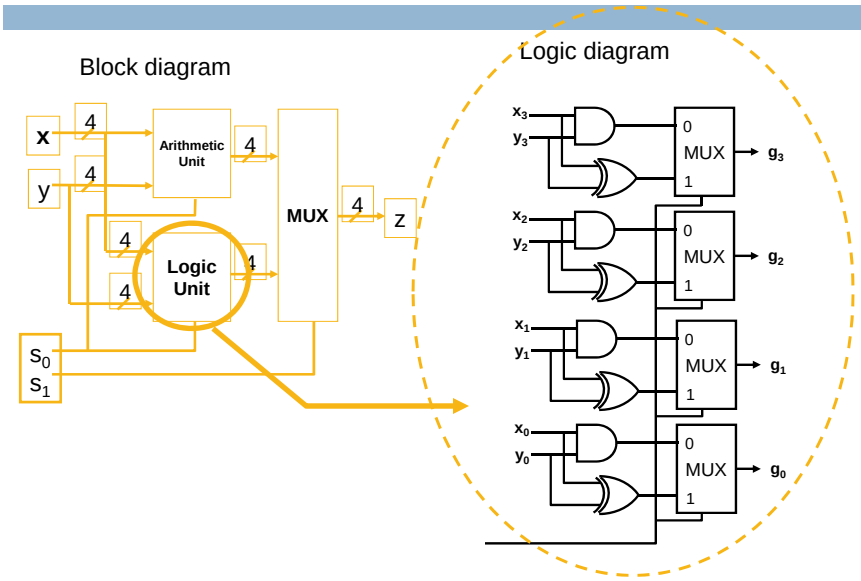


Hardware: Levels of Detail - 1 (top-down)

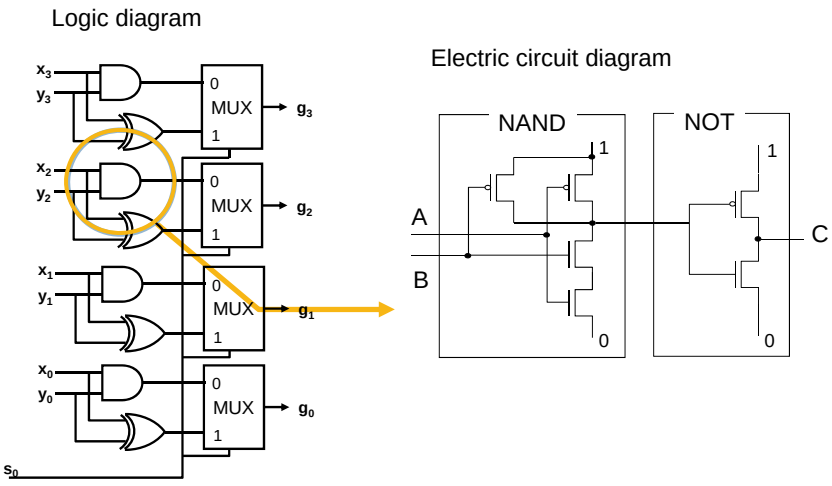


Magnified processor topography

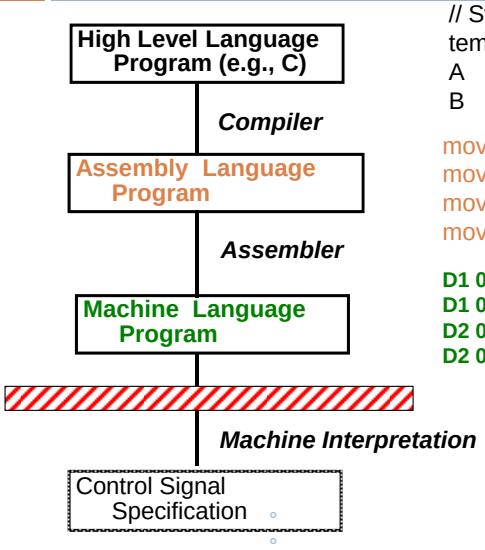
Hardware: Levels of Detail - 2



Hardware: Levels of Detail – 3 & 4



Design of Computers – Hierarchical Design Software (top-down)




```
// Swap 'A' and 'B'
temp    = A;
A       = B;
B       = temp;
```

```
mov     AX,A
mov     BX,B
mov     A,BX
mov     B,AX
```

```
D1 00 00
D1 01 01
D2 00 01
D2 01 00
```

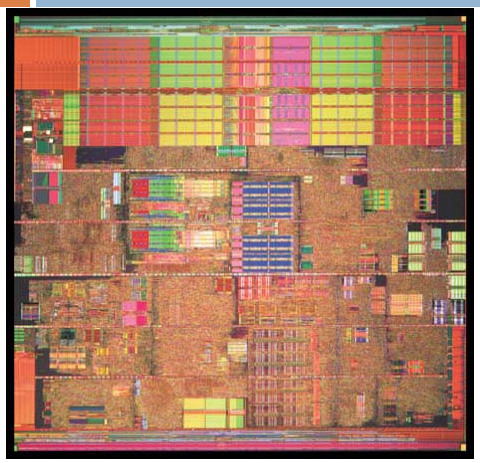
or

```
1101 0001 0000 0000 0000 0000
1101 0001 0000 0001 0000 0001
1101 0010 0000 0000 0000 0001
1101 0010 0000 0001 0000 0000
```

History of Computers

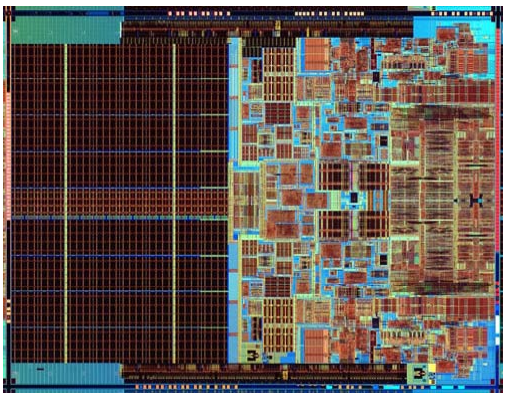
- Vacuum Tubes → Transistors → VLSI
- ENIAC
  - Eckert and Mauchly
  - 1946
  - 18,000 vacuum tubes, 1,800 instructions/sec, 3,000 ft^3
- Intel Microprocessor
  - Introduced in 1970
  - 2,250 transistors, 12 mm^2, 108 KHz
- Processor speed doubles every 18 months (Moore's Law)
- Memory speed doubles every 4-5 years

Pentium - 4



- 55,000,000 Transistors
- 146 mm^2
- 3 Ghz
- Introduced in 2000

Intel Core 2 Duo E6600



- 65nm technology node
- 291 Million transistors
- 143 mm^2

Number System - Definition

15

- A set of values to represent quantity
- We apply numbers everyday and knowing how numbers work will give us insight into how computer manipulates and stores numbers
  - Example: Roman Numerals (**1 = I, 5 = V, 10 = X, 50 = L, 100 = C, 500 = D, 1000 = M**)
  - http://mathforum.org/alejandre/numerals.html
  - http://www.sf.airnet.ne.jp/~ts/language/number.html

History

16

- The number system in most common use today is the *Arabic* system. It was first developed in India and was used as early as the 3rd century BC.
  - The introduction of the symbol 0, used to indicate the positional value of digits was very important. We thus became familiar with the concept of groups of units, tens of units, hundreds of units, and so on.
- In number systems, it is often helpful to think of **recurring sets**, where a set of values is repeated over and over again.
- We always write the digit with the largest value on the left of the number

Base Values

17

- The base (R=radix) value of a number system is the number of different values the set has before repeating itself
  - Binary = 2 (0, 1)
  - Octal = 8 (0 - 7)
  - Decimal = 10 (0 – 9)
  - Hexadecimal = 16 (0 – 9, A – F)
- A number at any base can be expanded in a power series (positional notation)

$$N = (a_4 a_3 a_2 a_1 a_0 . a_{-1} a_{-2} a_{-3})_R =$$
$$= a_4 \times R^4 + a_3 \times R^3 + a_2 \times R^2 + a_1 \times R^1 + a_0 \times R^0$$
$$+ a_{-1} \times R^{-1} + a_{-2} \times R^{-2} + a_{-3} \times R^{-3}$$
$$0 \le a_i \le R-1$$

Decimal Number System (Base 10)

18

- Uses 10 different symbols to represent values

9 8 7 6 5 4 3 2 1 0

the greatest value (digit and column) ↑ ↑ the least value (digit and column)

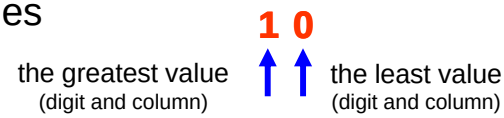
- $964_{10} = 9 \times 10^2 + 6 \times 10^1 + 4 \times 10^0$

$$N = (a_4 a_3 a_2 a_1 a_0 . a_{-1} a_{-2} a_{-3})_R =$$
$$= a_4 \times R^4 + a_3 \times R^3 + a_2 \times R^2 + a_1 \times R^1 + a_0 \times R^0 + \ldots$$
$$+ a_{-1} \times R^{-1} + a_{-2} \times R^{-2} + a_{-3} \times R^{-3}$$
$$0 \le a_i \le R-1$$

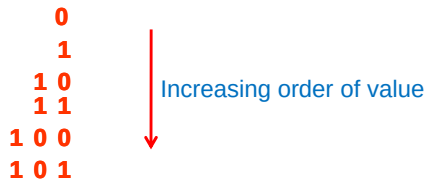
Decimal Number System (Base 2)

19

- Uses only **2** different symbols to represent values



- Columns are used identical to the decimal system, leftmost is used to represent the greatest value



Octal Number System (Base 8)

20

- Uses 8 different symbols to represent values

Hexadecimal Number System (Base 16)

21

- Uses 16 different symbols to represent values

0 1 2 3 4 5 6 7 8 9 A B C D E F

the least value ↑ ↑ the greatest value

- A = 10, B = 11, C = 12, D = 13, E = 14, F = 15

Binary to Decimal

22

- Convert 1110_2 to Decimal

- $1110_2 = 1\times2^3+1\times2^2+1\times2^1+0\times2^0$

  $= 8 + 4 + 2 + 0$

  $= 14_{10}$

$$N = (a_4a_3a_2a_1a_0.a_{-1}a_{-2}a_{-3})_R =$$
$$= a_4\times R^4 + a_3\times R^3 + a_2\times R^2 + a_1\times R^1 + a_0\times R^0 + \ldots$$
$$+ a_{-1}\times R^{-1} + a_{-2}\times R^{-2} + a_{-3}\times R^{-3}$$

$$0 \le a_i \le R-1$$

What about fraction conversion?

23

- Example:
  - $0.32_{10} = 3\times10^{-1} + 2\times10^{-2}$
  - $0.101_2 = 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$
    $= 0.5 + 0 + 0.125$
    $= 0.625$

$$N = (a_4 a_3 a_2 a_1 a_0 . a_{-1} a_{-2} a_{-3})_R =$$
$$= a_4 \times R^4 + a_3 \times R^3 + a_2 \times R^2 + a_1 \times R^1 + a_0 \times R^0 + \ldots$$
$$+ a_{-1} \times R^{-1} + a_{-2} \times R^{-2} + a_{-3} \times R^{-3}$$
$$0 \le a_i \le R-1$$

Decimal to Binary

24

- Divide the number by 2, then divide what's left by 2, and so on until there is nothing left (zero)
- 54_{10} =
  54/2 = 27, Remainder = 0
  27/2 = 13, Remainder = 1
  13/2 = 6, Remainder = 1
  6/2 = 3, Remainder = 0
  3/2 = 1, Remainder = 1
  1/2 = 0, Remainder = 1

  1 1 0 1 1 0

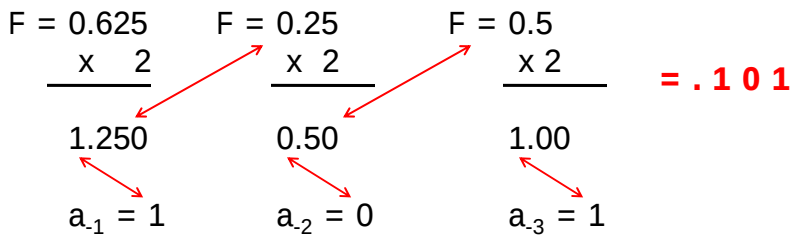
What about decimal fractions into binary?

25

- 0.625_{10} =

$$F = (.a_{-1}a_{-2}a_{-3})_R$$
$$F \times R = a_{-1} + a_{-2}R^{-1} + a_{-3}R^{-2} = a_{-1}.F_1$$
$$F_1 \times R = a_{-2} + a_{-3}R^{-1} = a_{-2}.F_2$$

Conversion of a decimal fraction to base R can be done using successive multiplications by R.



Class Problem

26

- Convert 147.3_8 into decimal
- Convert $A2F_{16}$ into decimal
- Convert 254_{10} into binary
- Convert 0.7_{10} into binary

Inter-Base Conversion

27

- Conversion between bases other than decimal is generally easier if we use decimal as an intermediate base
  - Example: Convert $(231.3)_4$ to base 7
- For Binary – Hexadecimal, each hexadecimal digit represents 4 binary bits, split the binary numbers into groups of 4 bits, starting from right
  - **Example: Convert $10101010010101101 10.011_2$**

101	0101	0010	1011	0110	.	011	
0101	0101	0010	1011	0110	.	0110	
5	5	2	11	6	.	6	$= (552B6.6)_{16}$

BIN-HEX-DEC Conversion Table

28

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	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

Binary Numbers & Coding

29

- Flexibility of representation
  - In principle, can assign any binary combination (called a code word) to any data as long as it is uniquely encoded
- Information Type
  - Numeric
    - Must represent the range of data needed
    - Simple, straightforward computation for common arithmetic operations permitted
    - Relation to binary numbers
  - Non-Numeric
    - Greater flexibility since no arithmetic operations
    - Not tied to binary numbers

Binary Codes for Decimal Digits

30

- There are over 8,000 ways in which you can choose 10 elements from 16 possible 4 bit binary numbers.

Decimal	8,4,2,1	Excess 3	8,4,-2,-1	Gray
0	0000	0011	0000	0000
1	0001	0100	0111	0100
2	0010	0101	0110	0101
3	0011	0110	0101	0111
4	0100	0111	0100	0110
5	0101	1000	1011	0010
6	0110	1001	1010	0011
7	0111	1010	1001	0001
8	1000	1011	1000	1001
9	1001	1100	1111	1000

Binary Coded Decimal (BCD)

31

- BCD is the 8,4,2,1 code
  - Referring to bit weights used
- Simplest, intuitive code for binary
- Uses powers of 2, but only encodes the first ten values from 0 to 9
  - Eg: 1001 (9) = 1000 (8) + 0001 (1)
  - There are “invalid” code words like 1100 or 1011

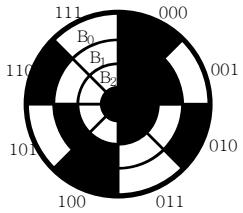
Decimal	8,4,2,1
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Gray Code

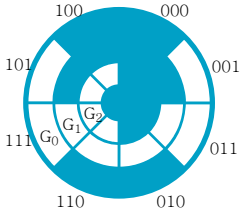
32

- Unweighted and non-arithmetic, arranged so that every transition from one value to the next involves only one bit change.

Decimal	Gray
0	0000
1	0100
2	0101
3	0111
4	0110
5	0010
6	0011
7	0001
8	1001
9	1000



(a) Binary Code for Positions 0 through 7



(b) Gray Code for Positions 0 through 7

Arithmetic: Binary, Octal, Hexadecimal, Binary Codes

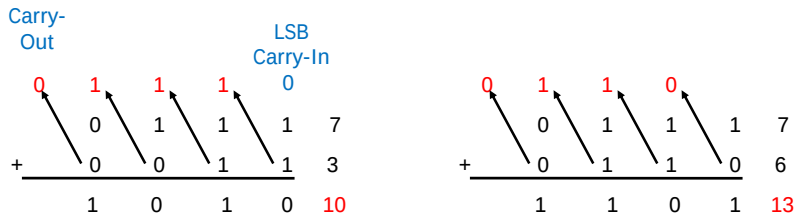
33

- Addition
- Subtraction
- Multiplication
- Division

Unsigned Binary Addition

34

- Just like in high school (carry 1s)

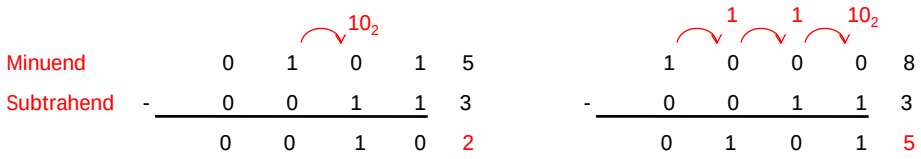


- 0 is the default Carry-In to the Least Significant Bit (LSB)

Binary Subtraction

35

- Just like in high school (borrow 1s)



Binary Multiplication

36

- Binary Multiplication is simple
  - 1 x 0 = 0; 0 x 0 = 0; 0 x 1 = 0; 1 x 1 = 1

Multiplicand	1011
Multiplier	x 101
Partial Products	1011 0000x 1011xx
Products	110111

Binary Division

37

- Arithmetic Inversion of Multiplication
- Example: Divide 11011.10 with 101

```
      00101.10
101 | 11011.10
    - 101
      ---
      111
    -  101
      ----
       101
     -  101
       ----
          0
```

Octal Multiplication

38

- Multiply 156_8 with 42_8 showing all intermediate steps

$$\begin{array}{r} 156_8 \\ \times \quad 42_8 \\ \hline 334_8 \\ + \quad 670_8 \\ \hline 7234_8 \end{array}$$

$2_8 \times 6_8 = 12_{10} = 14_8$
$1 + 2_8 \times 5_8 = 11_{10} = 13_8$
$1 + 2_8 \times 1_8 = 3_{10} = 3_8$

$4_8 \times 6_8 = 24_{10} = 30_8$
$3 + 4_8 \times 5_8 = 23_{10} = 27_8$
$2 + 4_8 \times 1_8 = 6_{10} = 6_8$

Hexadecimal Subtraction

39

- Subtract $(A96B)_{16} – (9F2C)_{16}$

Minuend		A 9 6 B_{16}
Subtrahend	-	9 F 2 C_{16}
Difference		0 A 3 F_{16}

The partial differences were found

a) B-C =1B-C=(16+11)-12=27-12=15=F

B-C produces a negative result, so a borrow is generated

b) 5 - 2 = 3

c) (9 + 16) – F = A

d) 9 – 9 = 0

BCD Arithmetic

40

- Given a BCD code, we use binary arithmetic to add the digits:

8	1000	Eight
+5	+0101	Plus 5
13	1101	is 13 (> 9)

- Note that the result is MORE THAN 9, so needs two digits!
- To correct the BCD digit, ADD 6,

8	1000	Eight
+5	+0101	Plus 5
13	1101	is 13 (> 9)
	+0110	so add 6
carry = 1	0011	leaving 3 + carry
	0001 | 0011	Final answer (two digits)

More BCD Arithmetic

41

- Add 448_{BCD} to 489_{BCD} showing carries and digit corrections

Carry 110

```
  448
+ 489
  937
```

BCD Carry		1	1	0
		0100	0100	1000
	+	0100	1000	1001
Binary sum		1001	1101	10001
Add 6	+		0110	0110
BCD Sum		1001	1 0011	1 0111
BCD Result		9	3	7