EE292: Fundamentals of ECE

Fall 2012
TTh 10:00-11:15 SEB 1242

Lecture 21
121113

http://www.ee.unlv.edu/~b1morris/ee292/
Outline

• Chapter 7 - Logic Circuits
• Binary Number Representation
• Binary Arithmetic
• Combinatorial Logic
Logic Circuits

- Analog signal – signal of continuous “time” variable with a continuous range of outputs
  - The signal has an infinite range of values at any time
  - E.g. a speech signal
- Digital signal – a signal with discrete “time” variable and only a few restricted amplitude values

![Diagram of analog and digital signals](image)
Digital Signals

• Computers are examples of digital circuits
  ▫ They operate on digital signals

• Binary signals are the most common type of signal
  ▫ The output of a binary signal takes only two possible values
  ▫ The two output values are often given “logical values” of a 1 or 0

• Often digital signals often come from physical analog processes
  ▫ The analog signal is converted into a digital form for processing in a computer
Digital Noise Advantage

- Digital signals are robust to noise
  - The exact signal value is not required
  - Rely on “logic” values
- Today it is possible to manufacture large numbers of digital logic circuits on integrated circuits because of this simplification

![Digital Signal Diagrams](image-url)
Positive Logic

- **Logical 1**
  - The higher amplitude value in a binary system
    - E.g. 5 volts
  - Also known as “high”, “true”, or “on”

- **Logical 0**
  - The lower amplitude in a binary system
    - E.g. 0 volts
  - Also known as “low”, “false”, or “off”

- **Logic variables** – signals in logic systems that switch between high and low
  - Will be denoted by uppercase letters (E.g. $A, B, C$)
Logic Ranges and Noise Margins

- Logic circuits are designed to have a range of input voltages map to a logical “high” or “low”
  - $V_{IL}$ - largest input value for logic 0 at input
  - $V_{IH}$ - smallest input value for logic 1 at input
  - $V_{OL}$ - largest output value for logic 0 at input
  - $V_{OH}$ - smallest output value for logic 1 at input
- Input and output have different logical ranges due to noise
  - The difference is known as the noise margin
Digital Words

• Bit – a single binary digit
  ▫ Smallest amount of information that can be represented in a digital system
  ▫ Represents a yes/no for a digital variable
  ▫ E.g. \( R = 0 \), represents not raining while \( R = 1 \), represents raining

• In order to represent more complex information, bits can be combined into digital words
  ▫ A byte is 8 bits and a nibble is 4 bits (used often in computers, e.g. a byte to represent each key on a keyboard)

• Example \( RWS \)
  ▫ \( R \) for rain, \( W \) for wind, \( S \) for sunny
  ▫ \( RWS = 110 \) indicates it is raining, with winds, and cloudy (e.g. not sunny)
Representation of Numerical Data

- Digital words allow representation of more complex values by concatenating digital variables
  - Only binary yes/no results were allowed
  - $RWS$ allowed $2^3$ different combinations of weather conditions
- Need a way to represent the wide range of values encountered in the physical world
  - Must be able to convert real numbers into a binary form for computation in a digital fashion
Decimal Representation of Numbers

• Consider a decimal number (base 10)
  ▫ This is what we as humans are familiar with

• Example $743.2_{10}$
  ▫ This is interpreted as

  $7 \times 10^2 + 4 \times 10^1 + 3 \times 10^0 + 2 \times 10^{-1}$

  ▫ Each digit is a multiplier by $10^d$
    • $d$ is the digit location
    • Positive to the left of decimal point and negative to the right
Binary Representation of Numbers

• Use the same technique as for decimal but instead use base 2 numbers

• Example 1101.1₂
  1 × 2³ + 1 × 2² + 0 × 2¹ + +1 × 2⁰ + 1 × 2⁻¹
  ▫ 1 × 2³ = 8
  ▫ 1 × 2² = 4
  ▫ 1 × 2⁰ = 1
  ▫ 1 × 2⁻¹ = 0.5
  1101.1₂ = 13.5

• Notice the subscript is used to indicate what the base to use for the number interpretation
Numerical Binary Words

- Enumerate all combinations of values for binary word
  - An $N$ bit word can represent $2^N$ different numbers

- Let $N = 4$, then there are $2^4 = 16$ different values that can be represented

- The leading zeros are presented in binary form because the digital circuits typically operate on fixed size words

<table>
<thead>
<tr>
<th>Binary</th>
<th>0000</th>
<th>0001</th>
<th>0010</th>
<th>0011</th>
<th>0100</th>
<th>0101</th>
<th>0110</th>
<th>0111</th>
<th>1000</th>
<th>1001</th>
<th>1010</th>
<th>1011</th>
<th>1100</th>
<th>1101</th>
<th>1110</th>
<th>1111</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>
Positional Notation for Numbers

- **Base B number → B symbols per digit**
  - Base 10 (Decimal): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - Base 2 (binary) 0, 1

- **Number representation**
  - $d_{31}d_{30}...d_2d_1d_0$ is 32 digit number
  - Value = $d_{31}B^{31} + d_{30}B^{30} + ... + d_1B^1 + d_0B^0$

- **Examples**
  - (Decimal): 90
    - $= 9 \times 10^1 + 0 \times 10^0$
  - (Binary): 1011010
    - $= 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
    - $= 64 + 16 + 8 + 2$
    - $= 90$
  - 7 binary digits needed for 2 digit decimal number
Hexadecimal Number: Base 16

- More human readable than binary
- Base with easy conversion to binary
  - Any multiple of 2 base could work (e.g. octal)
- Hexadecimal digits

- 1 hex digit represents 16 decimal values or 4 binary digits
  - Will use 0x to indicate hex digit
Hex/Binary Conversion

- Convert between 4-bits and a hex digit using the conversion table above

**Examples**

- 1010 1100 0101 (binary)
  - = 0xAC5
- 10111 (binary)
  - = 0001 0111 (binary)
  - = 0x17
- 0x3F9
  - = 0011 1111 1001 (binary)
  - = 11 1111 1001 (binary)
Signed Numbers

- N bits represents $2^N$ values
- Unsigned integers
  - Range $[0, 2^{32}-1]$
- How can negative values be indicated?
  - Use a sign-bit
  - Boolean indicator bit (flag)
Sign and Magnitude

- 16-bit numbers
  - +1 (decimal) = 0000 0000 0000 0001 = 0x0001
  - -1 (decimal) = 1000 0000 0000 0001 = 0x8001

- Problems
  - Two zeros
    - 0x0000
    - 0x8000
  - Complicated arithmetic
    - Special steps needed to handle when signs are same or different (must check sign bit)
Ones Complement

- Complement the bits of a number
  - $+1$ (decimal) = 0000 0000 0000 0001 = 0x0001
  - $-1$ (decimal) = 1111 1111 1111 1110 = 0xFFFE

- Positive number have leading zeros
- Negative number have leading ones
- Arithmetic not too difficult
- Still have two zeros
Two’s Complement

- Subtract large number from a smaller one
  - Borrow from leading zeros
  - Result has leading ones
- Unbalanced representation
  - Leading zeros for positive
    - $2^{N-1}$ non-negatives
  - Leading ones for negative number
    - $2^{N-1}$ negative number
  - One zero representation
- First bit is sign-bit (must indicate width)
  - Value = $d_{31} \times -2^{31} + d_{30} \times 2^{30} + ... + d_1 \times 2^1 + d_0 \times 2^0$
    - Negative value for sign bit

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>... 0011</td>
<td>3</td>
</tr>
<tr>
<td>... 0100</td>
<td>4</td>
</tr>
<tr>
<td>... 1111</td>
<td>-1</td>
</tr>
</tbody>
</table>
Two’s Complement Negation

- Shortcut = invert bits and add 1
  - Number + complement = 0xF..F = -1
    - \( x + \bar{x} = -1 \)
    - \( \bar{x} + 1 = -x \)

- Example
  \[
  \begin{array}{c|c}
    x & 1111 1110 \\
    \bar{x} & 0000 0001 \\
    \bar{x} + 1 & 0000 0010 \\
  \end{array}
  \]
Two’s Complement Sign Extension

• Machine’s have fixed width (e.g. 32-bits)
  ▫ Real numbers have infinite width (invisible extension)
    • Positive has infinite 0’s
    • Negative has infinite 1’s
• Replicate sign bit (msb) of smaller container to fill new bits in larger container
• Example

  1111 1111 1111 1111
  1111 1111 1111 1110

  1111 1111 1111 1110
Overflow

- Fixed bit width limits number representation
- Occurs if result of arithmetic operation cannot be represented by hardware bits
- Example
  - 8-bit: 127 + 127

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0111 1111</td>
<td>127</td>
</tr>
<tr>
<td>0111 1111</td>
<td>127</td>
</tr>
<tr>
<td>1111 1110</td>
<td>-2 (254)</td>
</tr>
</tbody>
</table>