Digital Logic Design: a rigorous approach

Chapter 5: Binary Representation

Guy Even    Moti Medina
School of Electrical Engineering Tel-Aviv Univ.

November 7, 2017

Book Homepage:
http://www.eng.tau.ac.il/~guy/Even-Medina
Division and Modulo

**Definition**

Given $a, b \in \mathbb{Z}$ define:

\[
(a \div b) \triangleq \max\{q \in \mathbb{Z} \mid q \cdot b \leq a\}
\]

\[
\text{mod}(a, b) \triangleq a - b \cdot (a \div b).
\]

**Claim**

\[
\text{mod}(a, b) \in \{0, 1 \ldots, b - 1\}.
\]

- $(a \div b)$ is called the **quotient** and mod$(a, b)$ is called the **remainder**.
- if mod$(a, b) = 0$, then $a$ is a multiple of $b$ ($a$ is **divisible** by $b$).
- $(a \div b) = \left\lfloor \frac{a}{b} \right\rfloor$.
- $(a \mod b), \text{mod}(a, b), a(\mod b)$ denote the same thing.
Examples

1. $3 \mod 5 = 3$ and $5 \mod 3 = 2$.
2. $999 \mod 10 = 9$ and $123 \mod 10 = 3$.
3. $a \mod 2$ equals 1 if $a$ is odd, and 0 if $a$ is even.
4. $a \mod b \geq 0$.
5. $a \mod b \leq b - 1$. 
Lemma

For every $z \in \mathbb{Z}$,

$$x \mod b = (x + z \cdot b) \mod b$$

Lemma

$$((x \mod b) + (y \mod b)) \mod b = (x + y) \mod b$$
Definition

A binary string is a finite sequence of bits.

Ways to denote strings:

1. sequence \( \{A_i\}_{i=0}^{n-1} \),
2. vector \( A[0 : n - 1] \), or
3. \( \vec{A} \) if the indexes are known.

We often use \( A[i] \) to denote \( A_i \).
$A[0 : 3] = 1100$ means $A_0 = 1$, $A_1 = 1$, $A_2 = 0$, $A_3 = 0$.

The notation $A[0 : 5]$ is zero based, i.e., the first bit in $\vec{A}$ is $A[0]$. Therefore, the third bit of $\vec{A}$ is $A[2]$ (which equals 0).
A basic operation that is applied to strings is called concatenation. Given two strings $A[0 : n - 1]$ and $B[0 : m - 1]$, the concatenated string is a string $C[0 : n + m - 1]$ defined by

$$C[i] \triangleq \begin{cases} 
A[i] & \text{if } 0 \leq i < n, \\
B[i - n] & \text{if } n \leq i \leq n + m - 1.
\end{cases}$$

We denote the operation of concatenating string by $\circ$, e.g.,

$$\bar{C} \equiv \bar{A} \circ \bar{B}.$$
Examples of concatenation of strings. Let $A[0:2] = 111$, $B[0:1] = 01$, $C[0:1] = 10$, then:

\[ \vec{A} \circ \vec{B} = 111 \circ 01 = 11101 , \]
\[ \vec{A} \circ \vec{C} = 111 \circ 10 = 11110 , \]
\[ \vec{B} \circ \vec{C} = 01 \circ 10 = 0110 , \]
\[ \vec{B} \circ \vec{B} = 01 \circ 01 = 0101 . \]
Let $i \leq j$. Both $A[i : j]$ and $A[j : i]$ denote the same sequence $\{A_k\}_{k=i}^j$. However, when we write $A[i : j]$ as a string, the leftmost bit is $A[i]$ and the rightmost bit is $A[j]$. On the other hand, when we write $A[j : i]$ as a string, the leftmost bit is $A[j]$ and the rightmost bit is $A[i]$.

**Example**

The **least significant** bit of the string $A[i : j]$ is the bit $A[k]$, where $k \triangleq \min\{i, j\}$. The **most significant** bit of the string $A[i : j]$ is the bit $A[\ell]$, where $\ell \triangleq \max\{i, j\}$.

The abbreviations LSB and MSB are used to abbreviate the least significant bit and the most significant bit, respectively.
The least significant bit (LSB) of $A[0 : 3] = 1100$ is $A[0] = 1$. The most significant bit (MSB) of $\vec{A}$ is $A[3] = 0$.


The least significant and most significant bits are determined by the indexes. In our convention, it is not the case that the LSB is always the leftmost bit. Namely, if $i \leq j$, then LSB in $A[i : j]$ is the leftmost bit, whereas in $A[j : i]$, the leftmost bit is the MSB.
We are now ready to define the binary number represented by a string $A[n - 1 : 0]$.

**Definition**

The natural number, $a$, represented in binary representation by the binary string $A[n - 1 : 0]$ is defined by

$$a \triangleq \sum_{i=0}^{n-1} A[i] \cdot 2^i.$$ 

In binary representation, each bit has a **weight** associated with it. The weight of the bit $A[i]$ is $2^i$. 
Consider a binary string $A[n − 1 : 0]$. We introduce the following notation:

$$\langle A[n − 1 : 0]\rangle \triangleq \sum_{i=0}^{n−1} A[i] \cdot 2^i.$$ 

To simplify notation, we often denote strings by capital letters (e.g., $A$, $B$, $S$) and we denote the number represented by a string by a lowercase letter (e.g., $a$, $b$, and $s$).
Consider the strings: $A[2 : 0] \triangleq 000$, $B[3 : 0] \triangleq 0001$, and $C[3 : 0] \triangleq 1000$. The natural numbers represented by the binary strings $A$, $B$ and $C$ are as follows.

\[
= 0 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 = 0,
\]

\[
= 1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 = 1,
\]

\[
\langle C[3 : 0] \rangle = C[0] \cdot 2^0 + C[1] \cdot 2^1 + C[2] \cdot 2^2 + C[3] \cdot 2^3 \\
= 0 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 = 8.
\]
Consider a binary string $A[n - 1 : 0]$. Extending $\bar{A}$ by leading zeros means concatenating zeros in indexes higher than $n - 1$. Namely,

1. extending the length of $A[n - 1 : 0]$ to $A[m - 1 : 0]$, for $m > n$, and
2. defining $A[i] = 0$, for every $i \in [m - 1 : n]$.

**Example**

$A[2 : 0] = 111$
$B[1 : 0] = 00$
The following lemma states that extending a binary string by leading zeros does not change the number it represents in binary representation.

**Lemma**

Let $m > n$. If $A[m - 1 : n]$ is all zeros, then

$$\langle A[m - 1 : 0] \rangle = \langle A[n - 1 : 0] \rangle.$$ 

**Example**

Consider $C[6 : 0] = 0001100$ and $D[3 : 0] = 1100$. Note that $\langle C \rangle = \langle D \rangle = 12$. Since the leading zeros do not affect the value represented by a string, a natural number has infinitely many binary representations.
The following lemma bounds the value of a number represented by a $k$-bit binary string.

**Lemma**

Let $A[k − 1 : 0]$ denote a $k$-bit binary string. Then, $0 \leq \langle A[k − 1 : 0] \rangle \leq 2^k − 1$.

What is the largest number representable by the following number of bits: (i) 8 bits, (ii) 10 bits, (iii) 16 bits, (iv) 32 bits, and (v) 64 bits?
Fix $k$ the number of bits (i.e., length of binary string).

Goals:

1. show how to compute a binary representation of a natural number using $k$ bits.
2. prove that every natural number in $[0, 2^k - 1]$ has a unique binary representation that uses $k$ bits.
Algorithm \( BR(x, k) \) for computing a binary representation is specified as follows:

**Inputs:** \( x \in \mathbb{N} \) and \( k \in \mathbb{N}^+ \), where \( x \) is a natural number for which a binary representation is sought, and \( k \) is the length of the binary string that the algorithm should output.

**Output:** The algorithm outputs “fail” or a \( k \)-bit binary string \( A[k - 1 : 0] \).

**Functionality:** The relation between the inputs and the output is as follows:

1. If \( 0 \leq x < 2^k \), then the algorithm outputs a \( k \)-bit string \( A[k - 1 : 0] \) that satisfies \( x = \langle A[k - 1 : 0] \rangle \).
2. If \( x \geq 2^k \), then the algorithm outputs “fail”.

Algorithm 1 $BR(x, k)$ - An algorithm for computing a binary representation of a natural number $a$ using $k$ bits.

1. **Base Cases:**
   1. If $x \geq 2^k$ then return (fail).
   2. If $k = 1$ then return $(x)$.

2. **Reduction Rule:**
   1. If $x \geq 2^{k-1}$ then return $(1 \circ BR(x - 2^{k-1}, k - 1))$.
   2. If $x \leq 2^{k-1} - 1$ then return $(0 \circ BR(x, k - 1))$.

example: execution of $BR(2, 1)$ and $BR(7, 3)$

**Theorem**

If $x \in \mathbb{N}$, $k \in \mathbb{N}^+$, and $x < 2^k$, then algorithm $BR(x, k)$ returns a $k$-bit binary string $A[k-1:0]$ such that $\langle A[k-1:0] \rangle = x$. 
How many bits do we need to represent $x$?

<table>
<thead>
<tr>
<th>Corollary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every positive integer $x$ has a binary representation by a $k$-bit binary string if $k &gt; \log_2(x)$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Proof.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$BR(x, k)$ succeeds if $x &lt; 2^k$. Take a log:</td>
</tr>
<tr>
<td>$\log_2(x) &lt; k$.</td>
</tr>
</tbody>
</table>
Theorem (unique binary representation)

The binary representation function

\[ \langle \rangle_k : \{0, 1\}^k \to \{0, \ldots, 2^k - 1\} \]

defined by

\[ \langle A[k - 1 : 0] \rangle_k \triangleq \sum_{i=0}^{k-1} A[i] \cdot 2^i \]

is a bijection (i.e., one-to-one and onto).

Proof.

1. \( \langle \rangle_k \) is onto because \( \langle BR(x, k) \rangle_k = x \).
2. \( |\text{Domain}| = |\text{Range}| \) implies that \( \langle \rangle_k \) is one-to-one.

\[ \square \]
We claim that when a natural number is multiplied by two, its binary representation is “shifted left” while a single zero bit is padded from the right. That property is summarized in the following lemma.

**Lemma**

Let \( a \in \mathbb{N} \). Let \( A[k-1:0] \) be a \( k \)-bit string such that \( a = \langle A[k-1:0] \rangle \). Let \( B[k:0] \triangleq A[k-1:0] \circlearrowleft 0 \), then

\[
2 \cdot a = \langle B[k:0] \rangle.
\]

**Example**

\[
\langle 1000 \rangle = 2 \cdot \langle 100 \rangle = 2^2 \cdot \langle 10 \rangle = 2^3 \cdot \langle 1 \rangle = 8.
\]