UNIVERSITY OF MASSACHUSETTS
Dept. of Electrical & Computer Engineering

Digital Computer Arithmetic
ECE 666

Part 2
Unconventional Number Systems

Israel Koren

Spring 2004

Unconventional Fixed-Radix Number Systems

- Number system commonly used in arithmetic units - binary system with two's complement representation of negative numbers
- Other number systems have proven to be useful for certain applications -
- Negative radix number system
- Signed-digit number system
- Sign-logarithm number system
- Residue number system

Negative-Radix Number Systems

- The radix r of a fixed-radix system - usually a positive integer
- r can be negative - $r=-\beta$ (β - a positive integer)
- Digit set - $x_i = 0,1,...,\beta-1$
- Value of n-tuple $(x_{n-1},x_{n-2},...,x_0)$ -

$$X = \sum_{i=0}^{n-1} x_i \, (-\beta)^i.$$

- The weight w_i is -

$$w_i = \begin{cases} \beta^i & \text{if } i \text{ is even} \\ -\beta^i & \text{if } i \text{ is odd.} \end{cases}$$

Example - Negative-Decimal System

- Negative-radix number system with $\beta=10$ - nega-decimal system
- Three-digit nega-decimal numbers -
  * $192_{-10} = 100-90+2=12$; $012_{-10}=-10+2=-8$
  * Largest positive value - $909_{-10}=909_{10}$
  * Smallest value - $090_{-10}=-90_{10}$
  * Asymmetric range - $-90 \leq X \leq 909$
  * Approximately 10 times more positives than negatives
- This is always true for odd values of n - opposite for even n
- Example - the range for $n=4$ is $-9090 \leq X \leq 909$

Negative-Radix Number Systems - Properties

- No need for a separate sign digit
- No need for a special method to represent negative numbers
- Sign of number is determined by the first nonzero digit
- No distinction between positive and negative number representations - arithmetic operations are indifferent to the sign of the numbers
- Algorithms for the basic arithmetic operations in the negative-radix number system are slightly more complex then their counterparts for the conventional number systems

Example - Negative-Binary System

- Negative-radix numbers of length $n=4$, $\beta=2$ - nega-binary system
- Range - $-10_{10}=1010_{-2} \leq X \leq 0101_{-2}=+5_{10}$
- When adding nega-binary numbers - carry bits can be either positive or negative
- Example:

-8	+4	-2	+1	
0	0	1	0	-2
0	0	1	1	-1
1	1	0	1	-3

- Nega-binary - proposed for signal processing applications
- Algorithms exist for all arithmetic operations
- Did not gain popularity: Main reason - not better than the two's complement system

A General Class of Fixed-Radix Number Systems

- The negative-radix, and many other fixed-radix number systems are members of a broad class of non-redundant number systems
- An n-digit number system - characterized by a positive radix β, digit set $0,1,...,\beta-1$, a vector Λ of length n - $\Lambda=(\lambda_{n-1},\lambda_{n-2},...,\lambda_0)$; $\lambda_i=-1,1$
- System identified by triplet $< n , \beta , \Lambda >$
- The value X of an n-tuple $(x_{n-1}, x_{n-2},...,x_0)$ is

$$X = \sum_{i=0}^{n-1} \lambda_i x_i \beta^i.$$

- The multiplying factor Λ allows a different selection between β^i and $-\beta^i$ for every digit position i

Special Cases of General Class

- For any given radix - 2^n different number systems corresponding to the different values of Λ
- Positive-radix number system -
  $\lambda_i = +1$ for every i
- Negative-radix number system -
  $\lambda_i = (-1)^i$ for every i
- Radix-complement number system with x_{n-1} as a "true" sign digit ($x_{n-1} = 0,1$) -
  $\Lambda = (-1,1,1,...,1)$

General Class <n,β,Λ> - Largest Representable Integer

- $P=(p_{n-1},p_{n-2},...,p_0)$

$$p_i = \begin{cases} \beta - 1 & \text{if } \lambda_i = +1 \\ 0 & \text{otherwise} \end{cases} = \frac{1}{2}(\lambda_i + 1)(\beta - 1);$$

$$\begin{aligned} P &= \sum_{i=0}^{n-1} \frac{1}{2}(\lambda_i + 1)(\beta - 1)\beta^i = \frac{1}{2}\left[\sum_{i=0}^{n-1} \lambda_i(\beta - 1)\beta^i + \sum_{i=0}^{n-1}(\beta - 1)\beta^i\right] \\ &= \frac{1}{2}\left[Q + (\beta^n - 1)\right] \end{aligned}$$

- **Q - value of n-tuple $(\beta-1,\beta-1,...,\beta-1)$**

General Class <n,β,Λ> - Smallest Representable Integer

- $N=(y_{n-1},y_{n-2},\ldots,y_0)$

$$y_i = \begin{cases} \beta-1 & \text{if } \lambda_\mathrm{i} = -1 \\ 0 & \text{otherwise} \end{cases} \quad ; \qquad i=0,1,\cdots,n-1$$

$$N = \sum_{i=0}^{n-1} \frac{1}{2}(\lambda_i - 1)(\beta-1)\beta^i = \frac{1}{2}\left[Q - (\beta^n - 1)\right]$$

- Range $N \leq X \leq P$ is asymmetric (in general)
- Number of integers in range - $P-N+1 = \beta^n$
- Measure of asymmetry - difference between the absolute values of the largest and smallest numbers - $P-|N| = P+N = Q$

Range of System - Examples

- Negative-radix system with $\Lambda=(\ldots,-1,+1,-1,+1)$ - asymmetric range
- For an even n - β times more negative numbers than positive
- To have more positive numbers - $\Lambda=(\ldots,+1,-1,+1,-1)$
- Two of the binary systems are nearly symmetric
- Two's complement - $\langle\, n,\beta=2,\Lambda=(-1,1,1,\ldots,1)\,\rangle$

  $P+N=Q=-ulp$

- $\langle\, n,\beta=2,\Lambda=(+1,-1,-1,\ldots,-1)\,\rangle$

  $P+N=Q=+ulp$

Additive Inverse and Subtraction in <n,β,Λ>

- Complement of a digit x_i - $\bar{x}_i = (\beta-1) - x_i$
- Complement $\bar{X}$ of a number X -

$$\overline{X} = \sum_{i=0}^{n-1} \bar{x}_i \lambda_i \beta^i = \sum_{i=0}^{n-1} \lambda_i (\beta - 1)\beta^i - \sum_{i=0}^{n-1} \lambda_i x_i \beta^i = Q - X$$

- $-X = \bar{X} - Q = \bar{X} + (-Q)$
- Additive inverse of X - adding the additive inverse of Q to the complement $\bar{X}$
- Subtraction - $X - Y = X + \bar{Y} + (-Q)$
  * Two add operations with carry propagation - time consuming
- Alternative - $X - Y = \overline{\bar{X} + Y}$
  * Two digit-complement operations, and only one addition with carry-propagation

Example

- Two's complement system - $Q=-ulp$ and $-X=\overline{X}+ulp$
- Nega-binary system
  - * $Q = (\ldots,1,1,1,1)_{-2} = -(\ldots,0,1,0,1)_{-2}$
  - * $-Q = (\ldots,0,1,0,1)_{-2}$
- Example - additive inverse of $(01011)_{-2}=(-9)_{10}$

$$
\begin{array}{cccccccc}
 & & 16 & -8 & +4 & -2 & +1 & \\
\overline{X} & & 1 & 0 & 1 & 0 & 0 & \\
-Q & + & 1 & 0 & 1 & 0 & 1 & \\
\hline
 & & 1 & 1 & 0 & 0 & 1 & = 9_{10}
\end{array}
$$

- Addition - following rules of nega-binary addition
- Result can be verified by adding the original number to its additive inverse (nega-binary addition)

  $(01011)_{-2} + (11001)_{-2} = (00000)_{-2}$

Signed-Digit Number Systems

- So far - digit set $\{0,...,r-1\}$
- In the signed-digit (SD) number system, digit set is $\{\overline{r-1},\overline{r-2},\dots,\bar{1},0,1,...,r-1\}$ $(\bar{i} = -i)$
- No separate sign digit
- Example:
  * $r=10,\ n=2$; digits - $\{\bar{9},\bar{8},...,\bar{1},0,1,...,8,9\}$
  * Range - $\bar{9}\bar{9} \leq X \leq 99$ - 199 numbers
  * 2 digits, 19 possibilities each - 361 representations - redundancy
  * $01=1\bar{9}=1$; $0\bar{2}=\bar{1}8=-2$
  * Representation of 0 (or 10) is unique
  * Out of 361 representations, 361-199=162 are redundant - 81% redundancy
  * Each number in range has at most two representations

Reducing Redundancy in Signed-Digit Number Systems

- Redundancy can be beneficial - but more bits needed
- Reducing redundancy - digit set restricted to

$$x_i \in \{\,\bar{a}, \overline{a-1}, \ldots, \bar{1}, 0, 1, \ldots, a\} \qquad \text{with} \qquad \left\lceil \frac{r-1}{2} \right\rceil \leq a \leq r-1$$

  * $\lceil x \rceil$ - smallest integer larger than or equal to x

- To represent a number in a radix r system - at least r different digits are needed
- $\bar{a} \leq x_i \leq a$ - $2a+1$ digits
- $2a+1 \geq r$ and $\left\lceil \frac{r-1}{2} \right\rceil \leq a$

Example - SD Number System

- $r=10$ - range of a is $5 \le a \le 9$
- If $a=6$, $n=2$ - 133 numbers in range $\bar{6}\bar{6} \le X \le 66$
- 13 values for each digit - total of $13^2 = 169$ representations - 27% redundancy
- 1 has only one representation - 01 - $1\bar{9}$ is not valid
- 4 has two representations - 04 and $1\bar{6}$

Eliminating Carry Propagation Chains

- Calculating $(x_{n-1},...,x_0) \pm (y_{n-1},...,y_0)=(s_{n-1},...,s_0)$
- Breaking the carry chains - an algorithm in which sum digit s_i depends only on the four operand digits x_i, y_i, x_{i-1}, y_{i-1}
- Addition time independent of length of operands
- An algorithm that achieves this independence:
- Step 1: Compute interim sum u_i and carry digit c_i
- $u_i = x_i + y_i - r\, c_i$
  where

$$c_i = \begin{cases} 1 & \text{if } \ (x_i + y_i) \geq a \\ \bar{1} & \text{if } \ (x_i + y_i) \leq \bar{a} \\ 0 & \text{if } \ |x_i + y_i| < a \end{cases}$$

- Step 2: Calculate the final sum $s_i = u_i + c_{i-1}$

Example - r=10, a=6

- $x_i = \bar{6}, \ldots, 0, 1, \ldots, 6$
- Step 1 - $u_i = (x_i + y_i) - 10\, c_i$
- Example - 3645+1456

$$c_i = \begin{cases} 1 & \text{if } (x_i + y_i) \geq 6 \\ \bar{1} & \text{if } (x_i + y_i) \leq \bar{6} \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{array}{r} 3645 \\ +\,1456 \\ \hline 5101 \end{array}$$

$$\begin{array}{rrrrrl} & 3 & 6 & 4 & 5 & x \\ + & 1 & 4 & 5 & 6 & y \\ \hline 0 & 1 & 1 & 1 & & c \\ & 4 & 0 & \bar{1} & 1 & u \\ \hline & 5 & 1 & 0 & 1 & s \end{array}$$

- In conventional decimal number system - carry propagates from least to most significant digit
- Here - no carry propagation chain
- Carry bits shifted to left to simplify execution of second step

Converting Representations

- This addition algorithm can be used for converting a decimal number to SD form by considering each digit as the sum x_i+y_i above
- Example - converting decimal 6849 to SD

$$\begin{array}{c|cccccc}
x_i+y_i & & 6 & 8 & 4 & 9 \\
\hline
c & 1 & 1 & 0 & 1 & \\
u & & \bar{4} & \bar{2} & 4 & \bar{1} \\
\hline
s & 1 & \bar{3} & \bar{2} & 5 & \bar{1}
\end{array}$$

- Converting SD to decimal - subtracting digits with negative weight from digits with positive weight
- Example - converting $1\bar{3}\bar{2}5\bar{1}$ to decimal

$$\begin{array}{r}
10050 \\
-03201 \\
\hline
6849
\end{array}$$

Proof of the Two-Step Algorithm

- To guarantee no new carry - $|s_i| \le a$
- Since $|c_{i-1}| \le 1$, $|u_i|$ must be $\le a-1$ for all x_i, y_i
- Example - largest x_i+y_i is $2a \Rightarrow c_i=1$, $u_i=2a-r$ since $a \le r-1$, $u_i=2a-r \le a-1$
- Example - smallest x_i+y_i for which $c_i=1$ is $a \Rightarrow u_i=a-r<0$ or $|u_i|=r-a$; to get $|u_i| \le a-1$, $2a \ge r+1$
- Selected digit set must satisfy $\left\lceil \frac{r+1}{2} \right\rceil \le a \le r-1$
- Exercise - show that for all values of x_i+y_i, $|u_i| \le a-1$ if $a \ge \left\lceil \frac{r+1}{2} \right\rceil$
- Example - for SD decimal numbers, $a \ge 6$ guarantees no new carries in previous algorithm

Addition of Binary SD Numbers

- Only one possible digit set - $\{1,0,\bar{1}\}$ - $a=1$
- Interim sum and carry in addition algorithm -

$$u_i = (x_i + y_i) - 2c_i \qquad c_i = \begin{cases} 1 & \text{if } (x_i+y_i) \ge 1 \\ \bar{1} & \text{if } (x_i+y_i) \le \bar{1} \\ 0 & \text{if } (x_i+y_i) = 0. \end{cases}$$

- Summary of rules -

$x_i y_i$	00	01	$0\bar{1}$	11	$\bar{1}\bar{1}$	$1\bar{1}$
c_i	0	1	$\bar{1}$	1	$\bar{1}$	0
u_i	0	$\bar{1}$	1	0	0	0

- Addition is commutative - $10, \bar{1}0, \bar{1}1$ not included
- In the binary case - $a \ge \left\lceil \frac{r+1}{2} \right\rceil = 2$ cannot be satisfied
- No guarantee that a new carry will not be generated in the second step of the algorithm

Addition - Carry Generation

- If operands do not have $\bar{1}$ - new carries not generated
- Example -
  - In conventional representation - a carry propagates from least to most significant position
  - Here - no carry propagation chain exists

$$
\begin{array}{ccccccc}
 & & 1 & 1 & \cdots & 1 & 1 \\
+ & & 0 & 0 & \cdots & 0 & 1 \\
\hline
 & 1 & 1 & 1 & \cdots & 1 & & c_i \\
 & & \bar{1} & \bar{1} & \cdots & \bar{1} & 0 & u_i \\
\hline
 & 1 & 0 & 0 & \cdots & 0 & 0 & s_i
\end{array}
$$

- If operands have $\bar{1}$ - new carries may be generated
- Example -
  - If $x_{i-1}y_{i-1} = 01$ - $c_{i-1} = 1$ and if $x_i y_i = 0\bar{1}$ - $u_i = 1$ $s_i = u_i + c_{i-1} = 1+1$ - a new carry is generated
  - Stars indicate positions where new carries are generated and must be allowed to propagate

$$
\begin{array}{cccccccc}
 & & 0 & \bar{1} & 1 & \bar{1} & 1 & 1 \\
+ & & 1 & 0 & 0 & \bar{1} & 0 & 1 \\
\hline
 & 1 & \bar{1} & 1 & \bar{1} & 1 & 1 & & c_i \\
 & & \bar{1} & 1 & \bar{1} & 0 & \bar{1} & 0 & u_i \\
\hline
 & & * & * & * & 1 & 0 & 0 & s_i
\end{array}
$$

Addition - Avoiding Carry Generation

- $c_{i-1}=u_i=1$ when $x_i y_i = 0\bar{1}$ and $x_{i-1}y_{i-1} = 11$ or 01
- Selecting $c_i=0$ - $u_i = \bar{1}$
- However, for $x_{i-1}y_{i-1}=\bar{1}\bar{1} \Rightarrow c_{i-1}=\bar{1}$ and we must still select $c_i=\bar{1}$, $u_i=1$
- Similarly, when $x_i y_i=01$ and $x_{i-1}y_{i-1}= \bar{1}\bar{1}$ or $0\bar{1}$, instead of $u_i=\bar{1} \Rightarrow$ select $c_i=0$ & $u_i=1$
- u_i and c_i can be determined by examining the two bits to the right $x_{i-1}y_{i-1}$
- u_i and c_i can still be calculated in parallel for all bit positions

$x_i y_i$	00	01	$0\bar{1}$	11	$\bar{1}\bar{1}$	$1\bar{1}$
c_i	0	1	$\bar{1}$	1	$\bar{1}$	0
u_i	0	$\bar{1}$	1	0	0	0

Addition of Binary SD Numbers - Modified Rules

$x_i y_i$	00	01	01	$0\bar{1}$	$0\bar{1}$	11	$\bar{1}\bar{1}$
$x_{i-1}\,y_{i-1}$	–	neither is $\bar{1}$	at least one is $\bar{1}$	neither is $\bar{1}$	at least one is $\bar{1}$	–	–
c_i	0	1	0	0	$\bar{1}$	1	$\bar{1}$
u_i	0	$\bar{1}$	1	$\bar{1}$	1	0	0

- Example

-	1	$\bar{1}$	0	$\bar{1}$	1		7
	0	$\bar{1}$	1	0	0		-4
0	$\bar{1}$	0	0	1			c
	1	0	1	$\bar{1}$	$\bar{1}$		u
	0	0	1	0	$\bar{1}$		3

- **Direct summation of the two operands results in 001$\bar{1}$1, equivalent to 00101, all representing 3_{10}**

Minimal Representations of Binary SD Numbers

- Representation with a minimal number of nonzero digits
- Important for fast multiplication and division algorithms
- Nonzero digits - add/subtract operations, zero digits - shift-only operations
- Example - Representations of X=7:
- $100\bar{1}$ is the minimal representation
- The canonical recoding algorithm generates minimal SD representations of given binary numbers

	8	4	2	1
	0	1	1	1
	1	$\bar{1}$	1	1
	1	0	$\bar{1}$	1
	1	0	0	$\bar{1}$
1	$\bar{1}$	$\bar{1}$	1	1
		$\vdots$		

Encoding of SD Binary Numbers

- 4x3x2=24 ways to encode 3 values of a binary signed bit x using 2 bits, x^h and x^l (high and low)
- Only nine are distinct encodings under permutation and logical negation - two have been used in practice

x	Encoding 1 $x^h\ x^l$	Encoding 2 $x^h\ x^l$
0	0 0	0 0
1	0 1	0 1
$\bar{1}$	1 0	1 1

- Encoding #2 - a two's complement representation of the signed digit x
- Encoding #1 is sometimes preferable
  * Satisfies $x = x^l - x^h$ - 11 has a valid value of 0
  * Simplifies conversion from SD to two's complement - subtracting $x^h_{n-1}, x^h_{n-2}, ..., x^h_0$ from $x^l_{n-1}, x^l_{n-2}, ..., x^l_0$ using two's complement arithmetic
  * This requires a complete binary adder
  * A simpler conversion algorithm exists

Conversion Algorithm - Simpler Circuit

* Binary signed digits examined one at a time, right to left
* All occurrences of $\bar{1}$ are removed and the negative sign is forwarded to the most significant bit, the only bit with a negative weight in the two's complement representation
* The rightmost $\bar{1}$ is replaced by 1 and the negative sign is forwarded to the left, replacing 0's by 1's until a 1 is reached, which ``consumes" the negative sign and is replaced by 0
* If a 1 is not reached - the 0 in the most significant position is turned into a 1, becoming the negative sign bit of the two's complement representation
* If a second $\bar{1}$ is encountered before a 1 is, it is replaced by a 0 and the forwarding of the negative sign continues
* The negative sign is forwarded with the aid of a "borrow" bit which equals 1 as long as a $\bar{1}$ is being forwarded, and equals 0 otherwise

Conversion Algorithm Rules

- y_i - i-th digit of the SD number
- z_i - i-th bit of the two's complement representation
- c_i - previous borrow
- c_{i+1} - next borrow

y_i	c_i	z_i	c_{i+1}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0
$\bar{1}$	0	1	1
$\bar{1}$	1	0	1

- For the least significant digit we assume $c_0=0$
- $y_i - c_i = z_i - 2c_{i+1}$

Conversion Algorithm - Cont.

- **Example** - -10_{10} - converting SD to two's complement

y_i		0	$\bar{1}$	0	$\bar{1}$	0
c_i	1	1	1	1	0	0
z_i		1	0	1	1	0

- Range of representable numbers in SD method - almost double that of the two's complement method
- n-digit SD number must be converted to an ($n+1$)-bit two's complement representation

y_i		0	1	0	1	0	$\bar{1}$
c_i	0	0	0	0	1	1	0
z_i		0	1	0	0	1	1

- Without the extra bit position - the number 19 would be converted to -13