SIGN DETECTION IN THE SYMMETRIC RESIDUE NUMBER SYSTEM

Saroj Kaushik and R.K. Arora

Computer Centre, Indian Institute of Technology, Delhi
New Delhi 110016, India

ABSTRACT

This paper is concerned with the algebraic sign detection of a number in the Symmetric Residue Number System. A new approach has been suggested which completely avoids the time consuming process of the Symmetric Mixed Radix Conversion (SMRC). An algorithm based on the above approach implementable in parallel for sign detection is also presented. The hardware representation of the above algorithm is shown. The time and hardware complexity required for the process have also been computed.

INTRODUCTION

Several kinds of representation for residue numbers have been proposed, each of which has merits and demerits. The Symmetric Residue Number System (SRNS) has many advantages over the Residue Number System (RNS) viz., it is easy in the SRNS to find the additive inverse of a residue digit, it gives an effective solution for sign sensing and magnitude determination etc. Many solutions to the problem of sign detection in the RNS have been suggested[1-4], but in the SRNS, there is no specific method for detecting the sign of a number except the SMRC process. Certainly it computes the sign but is relatively time consuming because of the cascaded process used. Besides, it furnishes more information (all symmetric mixed radix co-efficients). The purpose of this paper is to present a new approach for algebraic sign determination in the SRNS, which completely avoids the time consuming process of the SMRC.

SYMMETRIC RESIDUE ARITHMETIC FUNDAMENTALS

Before considering the problem of sign detection, the basic definitions and operations in the SRNS are introduced to make this paper self-contained.

Definition 1 : The symmetric residue of X modulo m is the least remainder in absolute value when an integer X is divided by another positive integer m. Here m is called a modulus. A commonly used form of this condition is

$$X = \left[\frac{X}{m}\right]^{*} \cdot m + /X/_{m} ,$$

where $\left\lfloor -\frac{m}{2}\right\rfloor < /X/_{m} \leq \left\lfloor\frac{m}{2}\right\rfloor^{**}$.

$/X/_{m}$ is the notation for symmetric residue of X with respect to m. When m is odd, these residues are symmetric with respect to the origin, but when m is even, perfect symmetry is lost. In this case, the quantity $\frac{X}{m}$ is again the closest integer to $\frac{X}{m}$, except that if X is of the form $\frac{nm}{2}$, where n is odd, then the quantity $\frac{X}{m}$ is the closest integer to $\frac{X-1}{m}$.

Definition 2 : The ordered n-tuple $(/X/_{m_1}, /X/_{m_2}, \ldots, /X/_{m_n})$ of symmetric residues of X with respect to an ordered set $(m_1, m_2, \ldots, m_n)$ of n moduli form the symmetric residue representation of X.

Definition 3 : The SRNS is defined as a number system where the integers in the interval $\left[\left\lfloor -\frac{M}{2}\right\rfloor + 1, \left\lfloor\frac{M}{2}\right\rfloor\right]$ (upper square brackets indicate a closed interval), where $M = \prod_{i=1}^{n} m_i$, are represented by their symmetric residue representation corresponding to the moduli $m_1, m_2, \ldots, m_n$. If these moduli are chosen to be pairwise relatively prime, then any integer in the interval $\left[\left\lfloor -\frac{M}{2}\right\rfloor + 1, \left\lfloor\frac{M}{2}\right\rfloor\right]$ is uniquely

* $[I]$ denotes the closest integer to I.

** $\lfloor I \rfloor$ denotes the floor of I i.e., the largest integer $\leq$ I.

represented by its symmetric residue representation and it is denoted by

$$X \longleftrightarrow (/X/_{m_1}, /X/_{m_2}, \ldots, /X/_{m_n}).$$

The addition, subtraction and multiplication of two numbers X and Y in the SRNS is defined as follows as long as the operands and the result are falling in the interval $[\lfloor -\frac{M}{2} \rfloor + 1, \lfloor \frac{M}{2} \rfloor]$.

$$X*Y \longleftrightarrow (//X/_{m_1}*/Y/_{m_1}/_{m_1}, //X/_{m_2}*/Y/_{m_2}/_{m_2}, \ldots, //X/_{m_n}*/Y/_{m_n}/_{m_n})$$

where the symbol * represents addition, subtraction or multiplication of two numbers.

Symmetric Mixed Conversion Process: It is used to convert a residue code of a number to its symmetric mixed-radix representation. A number X may be expressed in symmetric mixed radix form as

$$X = r_n \prod_{i=1}^{n-1} m_i + \ldots + r_3 \cdot m_1 \cdot m_2 + r_2 \cdot m_1 + r_1, \qquad (1)$$

where r_i and determined sequentially from equation (1).

GENERAL APPROACH FOR SIGN DETECTION IN THE SRNS

We prove the following theorem based on which an algorithm has been suggested. An implementation of this algorithm has also been proposed which is parallel in nature. For the sake of simplicity, we write $/X/_{m_i} = x_i$, $i=1,2, \ldots, n$. Let $\hat{m}_i = \frac{M}{m_i}$.

Theorem

For any modulus m_i, $1 \leq i \leq n$, the sign of a number X in the interval $[\lfloor -\frac{M}{2} \rfloor + 1, \lfloor \frac{M}{2} \rfloor]$ is established by a proposition P in the following way:

Case (a) : If $\hat{m}_i$ is odd, then X is non-negative iff

$$P = \bigvee_{j=1}^{\lfloor \frac{\hat{m}_i}{2} \rfloor} \{x_k = /j \cdot m_i + x_i/_{m_k}\}$$
$$\vee \{x_i = /x_i/_{m_k} \wedge x_i \geq 0\}$$

is true for $k = 1,2,\ldots, i-1, i+1,\ldots, n$.

Case (b) : If $\hat{m}$ is even, then X is non-negative iff

$$P = \bigvee_{j=1}^{\lfloor \frac{\hat{m}_i}{2} \rfloor -1} \{x_k = /j \cdot m_i + x_i/_{m_k}\}$$
$$\vee \{x_k = /\lfloor \tfrac{\hat{m}_i}{2} \rfloor \cdot m_i + x_i/_{m_k} \wedge x_i \leq 0\}$$
$$\vee \{x_k = /x_i/_{m_k} \wedge x_i \geq 0\} \text{ is true}$$

for $k = 1,2,\ldots,i-1, i+1,\ldots, n$,

where $\vee$, $\wedge$ are logical 'OR' and 'AND' operators.

Proof: Proof is given in an appendix.

Remark 1 : The number of iterations can be reduced by choosing m_i to be the largest modulus.

ALGORITHM FOR SIGN DETECTION OF A RESIDUE NUMBER

The following algorithm, based on the above theorem is used to detect the sign of a residue number. Choose m_i to be the largest modulus and let $P = \frac{\hat{m}_i}{2}$.

Algorithm

Input : The number $X \leftrightarrow (x_1, x_2, \ldots, x_n)$.

Output: The sign of X.

Procedure

Step 1 : Start the processes Q_1 and Q_2 simultaneously:

Q_1 and Q_2 : Obtain the symmetric residue representation of x_i & m_i w.r.t. $m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_n$ as

$$x_i \leftrightarrow (x_i^1, x_i^2, \ldots, x_i^{i-1}, x_i^{i+1}, \ldots, x_i^n),$$

$$m_i \leftrightarrow (m_i^1, m_i^2, \ldots, m_i^{i-1}, m_i^{i+1}, \ldots, m_i^n),$$

where

$x_i^j = /x_i/_{m_j}$ and $m_i^j = /m_i/_{m_j}$, $j=1,2,\ldots, i-1, i+1, \ldots n$.

Step 2 : Start the processes $R_0, R_1, R_2, \ldots, R_p, R'_p$.
Comment - Description of R_j, $j = 0,1,\ldots,p$ and R'_p is given at the end of this procedure.

Step 3 : Find $S' = (\alpha^0 \cdot \alpha^1 \cdots \alpha^{p-1})$.

Step 4 : If $\hat{m}_i$ is even, then go to the next step else form $S=S'\cdot\alpha^p$ and go to step 6.

Step 5 : Form $S = S'\cdot\alpha_1^p$.

Step 6 : If S = 0 then X is non-negative else it is negative Stop.

<u>Description of the process R_0</u> :

Start the processes $R_0^1, R_0^2, \ldots, R_0^{i-1}, R_0^{i+1}, \ldots, R_0^n$

where R_0^k : Mod m_k comparator, which compares x_k and x_i^k. If compared, then set $\beta_0^k = 1$, otherwise $\beta_0^k = 0$, for k=1,2, ...,i-1, i+1, ..., n.

If $x_i \geq 0$, then set a = 1 else a = 0.
Compute a logical variable α^0 as

$$\alpha^0 = \overline{(\beta_0^1 \cdot \beta_0^2 \cdots \beta_0^{i-1} \cdot \beta_0^{i+1} \cdots \beta_0^n \cdot a)}$$

<u>Description of the process R_j</u>, j=1,2, ..., p.

Let $j.m_i \leftrightarrow (/j.m_i^1/_{m_1}, \ldots, /j.m_i^{i-1}/_{m_{i-1}}, /j.m_i^{i+1}/_{m_{i+1}}, \ldots, /j.m_i^n/_{m_n})$,

then $Z^j = x_i + j.m_i \leftrightarrow (z_1^j, z_2^j, \ldots, z_{i-1}^j, z_{i+1}^j, \ldots, z_n^j)$,

where $z_k^j = /x_i^k + /j.m_i^k/_{m_k}/_{m_k}$, k=1,2,...,n.

Start the processes $R_j^1, R_j^2, \ldots, R_j^{i-1}, R_j^{i+1}, \ldots, R_j^n$,

where R_j^k : Mod m_k comparator which compres x_k and z_k^j. If compared, then set $\beta_j^k = 1$, otherwise $\beta_j^k = 0$, for k=1,2, ..., i-1, i^j+1, ..., n.

Compute a logical variable α^j as

$$\alpha^j = \overline{(\beta_j^1 \cdot \beta_j^2 \cdots \beta_j^{i-1} \cdot \beta_j^{i+1} \cdots \beta_j^n)}.$$

<u>Description of the process R'_p</u>

If $x_i \leq 0$, set b = 1 else b = 0.

Compute a logical variable α_1^p as

$$\alpha_1^p = \overline{(\beta_p^1 \cdot \beta_p^2 \cdots \beta_p^{i-1} \cdot \beta_p^{i+1} \cdots \beta_p^n \cdot b)}$$

where β_p^k, k = 1,2,...,i-1,...,n are obtained in the same manner as obtained in the case of the process R_p above.

<u>Example</u>

Consider the SRNS of 4 pairwise relatively prime moduli 7,5,3,2. Let $X \leftrightarrow (1, 1, 1, 0)$ be the number whose sign is to be detected. Choose any modulus, say m_1 = 7 and p = 15.

Step 1 : Start the processes Q_1 and Q_2

Q_1: $x_i = x_1 \leftrightarrow (1, 1, 1)$ with respect to the moduli 5,3,2

Q_2: $m_i = m_1 \leftrightarrow (2, 1, 1)$ with respect to the moduli 5,3,2

Step 2 : Start the processes $R_0, R_1, \ldots, R_{15}, R'_{15}$.

<u>Process R_0</u>: Start the process R_0^2, R_0^3, R_0^4.

R_0^2: Mod m_2 comparator which compares x_2 and $/x_1/_{m_2}$ i.e., compare -1 and -1, in mod 5 comparator. Since they compare, set $\beta_0^2 = 1$.

Similarly the processes R_0^3 and R_0^4 yield, $\beta_0^3 = 1$ and $\beta_0^4 = 0$.

Now $x_i = x_1 \geq 0$, set a = 1.
Compute a logical variable α^0 as

$$\alpha^0 = \overline{(\beta_0^2 \cdot \beta_0^3 \cdot \beta_0^4 \cdot a)} = 1.$$

Process R_1 : $z^1 = x_1 + m_1 \longleftrightarrow (z_2^1, z_3^1, z_4^1)$

$= (2, -1, 0)$

Start the processes R_1^2, R_1^3, R_1^4.

R_1^2 : Compare x_2 and z_2^1 in mod 5 comparator. They do not compare, so set $\beta_1^2 = 0$.

The processes R_1^3 and R_1^4 give,

$$\beta_1^3 = 0 \text{ and } \beta_1^4 = 1.$$

Compute $\alpha^1 = (\beta_1^2 \cdot \beta_1^3 \cdot \beta_1^4) = 1$

Similarly the processes R_2, R_3, ..., R_{15} are executed simultaneously to yield

$\alpha^2 = \alpha^3 = \ldots = \alpha^{14} = 1$ and $\alpha^{15} = 0$

Process R_{15}' : Since $x_1 \geq 0$, set $b = 0$. Compute a logical variable α_1^{15} as

$$\alpha_1^{15} = (\beta_{15}^1 \cdot \beta_{15}^2 \cdots \beta_{15}^{14} \cdot b) = 1.$$

Step 3 : Find $S' = (\alpha^0 \cdot \alpha^1 \ldots \alpha^{14}) = 1$.

Step 4 : Since $\hat{m}_1$ is even, form

$$S = S' \cdot \alpha_1^{15} = 1.$$

Step 5 : Since $S = 1$, the number X is negative. Stop.

HARDWARE REPRESENTATION

A hardware representation of this algorithm, to detect the sign of $X \longleftrightarrow (x_1, x_2, x_3)$, is shown in Fig. 1, for the case when the SRNS consists of the moduli 5,3,2. The symmetric residue corresponding to any modulus will be represented in binary representation. It requires one extra bit for sign viz., symmetric residues corresponding to the moduli 5,3,2 are represented in 4,3 and 2 bits respectively.

In Fig. 1 circled '+' denotes the modular addition. The comparators have output 1 only for identical inputs. We use semiconductor gates in our design. The '•' denotes 'wired OR' function.

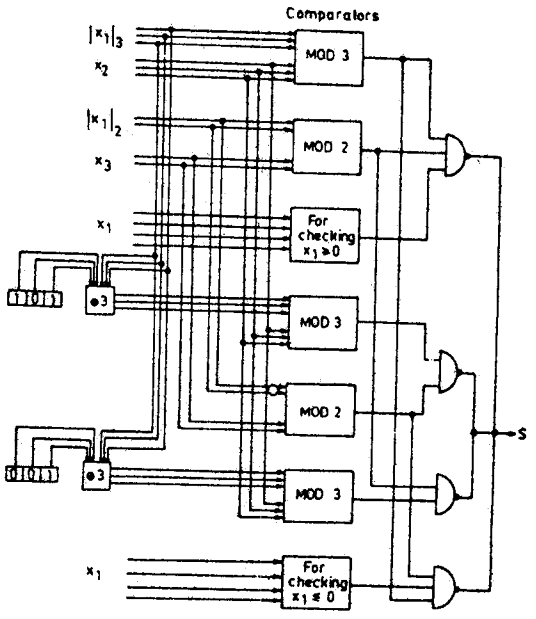

Fig. 1 Hardware representation of the scheme for sign detection

Remarks 2 : The proposed scheme for sign detection can be generalized to one in which the selected modulus m_i is replaced by a selected subset of the set $(m_1, m_2, \ldots, m_n)$ of the moduli, $\hat{m}_i$ is then the product of the remaining moduli.

Remark 3 : In the above implementation, we have considered the case when $\hat{m}_i$ is even. But if $\hat{m}_i$ is odd then in the proposed implementation, disconnect the output of the last comparator.

TIME AND HARDWARE COMPLEXITY REQUIRED FOR THE PROCESS OF SIGN DETECTION

In this section, we estimate the time required to detect the sign of a number in its symmetric residue representation and the hardware complexity.

Time Required to Compute the Sign

Let T_S = time required to detect the sign of a residue number,

t_c = time required to convert x_i to $(/x_i/_{m_1}, /x_i/_{m_2}, \ldots, /x_i/_{m_{i-1}}, /x_i/_{m_{i+1}}, \ldots, /x_i/_{m_n})$,

t_a = time required for one residue addition,

t^* = time required by comparator to compare,

$/x_i/_{m_k}$ with x_k, $k = 1,2,\ldots,i-1, i+1, \ldots, n$,

t_s = time required to compute S with the help of α^j's.

Then

$T_S \approx t_c + t_a + t^* + \Delta + t_s$, where Δ is a single gate delay.

Assume that combinational logic is used to convert x_i to $(/x_i/_{m_1}, \ldots, /x_i/_{m_{i-1}}, /x_i/_{m_{i+1}}, \ldots, /x_i/_{m_n})$ is used, then

$T_S \approx 8\Delta + t_s$, where $t_a = 3\Delta$

Hardware Complexity

Let L_k = the number of gates required to convert x_i to $/x_i/_{m_k}$,

A_k = the number of gates required for mod m_k adder,

C_k = the number of gates required for mod m_k comparator,

D_k = the number of gates required for comparator, comparing whether $x_i \geq 0$ or $x_i \leq 0$, for $k = 1,2,\ldots,i-1,i+1,\ldots,n$.

For the implementation suggested here, the number of mod m_k adders used are $(m_k - 1)$, the number of mod m_k comparators used are m_k,

for $k = 1,2,\ldots,i-1,i+1,\ldots,n$.

The number of NAND gates used are $\left\lfloor \frac{\hat{m}_i}{2} \right\rfloor + 1$. It is assumed that the NAND gate has fan-in of n. Here, the sign function S is directly obtained by performing 'Wired OR' function. But, if the value of $\left\lfloor \frac{\hat{m}_i}{2} \right\rfloor$ is very large, then we have to introduce the logical gates. Let T* be the number of gates required in the last part of our design to get S, then the total complexity estimated (in terms of gates) is

$$\sum_{\substack{k=1 \\ k \neq i}}^{n} \{ L_k + A_k (m_k - 1) + \hat{C}_k m_k \} + 2 D_k + \left\lfloor \frac{\hat{m}_i}{2} \right\rfloor + T^* .$$

Here we see that the hardware complexity increases with the increase in the range of the system. So, this approach is suitable for MSI/LSI realization. But this scheme saves execution time at the cost of hardware complexity.

CONCLUSION

The scheme for sign detection of a number in the SRNS presented in this paper completely does away with the time consuming process of the SMRC. The hardware implementation of the algorithm based on the proposed approach is also suggested. As we have shown that the hardware complexity increases with the increase in the range of the system, it is suitable only for MSI/LSI realization.

REFERENCES

1 Banerji, D.K. and Brzozowski, J.A., Sign Detection in Residue Number Systems, IEEE Trans. on Computers, vol. C-18, pp. 313-320, April 1969.

2 Banerji, D.K., Residue Arithmetic in Computer Design, Ph.D. Dissertation, Dept of Applied Analysis and Computer Science, University of Waterloo, Ontario, Canada, March 1971.

3 Banerji, D.K., On Combinational Logic for Sign Detection in Residue Number Systems, Third IEEE Symposium on Computer Arithmetic, Southern Methodes University, Dallas, Texas, pp. 113-116, November 1975.

4 Kaushik, S., On the Arithmetic Operations and Error Correction in Residue Code, Ph.D. Thesis, Indian Institute of Technology, New Delhi, March 1980.

5 Szabo, N.S., Sign Detection in Non-Redundant Residue Systems, IRE Trans. on Electronic Computers, vol. EC-11, pp. 494-500, August 1962.

6 Szabo, N.S. and Tanaka, R.I., Residue Arithmetic and its Applications to Computer Technology, New York: McGraw-Hill, 1967.

Appendix

Proof of the theorem

$X \in [\lfloor -\frac{M}{2} \rfloor + 1, \lfloor \frac{M}{2} \rfloor]$, can be written as

$X = j\, m_i + /X/_{m_i}$, for some j,

$$\left\lfloor -\frac{\hat{m}_i}{2} \right\rfloor < j \leq \left\lfloor \frac{\hat{m}_i}{2} \right\rfloor,$$

or $X = j\, m_i + x_i$, (2)

$\Rightarrow /X/_{m_k} = /j\, m_i + x_i/_{m_k}$

for $k = 1,2,\ldots,i-1,i+1,\ldots,n$,

or $x_k = /j\, m_i + x_i/_{m_k}$.

It is obvious that X is non-negative in the range $[0, \lfloor \frac{M}{2} \rfloor]$. There are two cases.

<u>Case (a)</u> : If $\hat{m}_i$ is odd, then for non-negative X, j is bounded between 1 and $\lfloor \frac{\hat{m}_i}{2} \rfloor$, since for j = 1

$$0 < X = m_i + x_i < \lfloor \tfrac{M}{2} \rfloor$$

and for $j = \lfloor \frac{\hat{m}_i}{2} \rfloor$

$$0 < X = \lfloor \tfrac{\hat{m}_i}{2} \rfloor .\, m_i + x_i \leq \lfloor \tfrac{\hat{m}_i}{2} . m_i \rfloor = \lfloor \tfrac{M}{2} \rfloor .$$

If j = 0 then equation (2) becomes

$$X = x_i .$$

For non-negative X,

$$X \geq 0 = x_i \geq 0 .$$

In conclusion, we say that X is non-negative iff

$$\bigvee_{j=1}^{\lfloor \frac{\hat{m}_i}{2} \rfloor} \{x_k = /j m_i + x_i/_{m_k}\} \vee \{ x_k = /x_i/_{m_k} \wedge x_i \geq 0\}$$ is true.

<u>Case (b)</u> : If $\hat{m}_i$ is even, then $\lfloor \frac{\hat{m}_i}{2} \rfloor = \frac{\hat{m}_i}{2}$

For $j = \lfloor \frac{\hat{m}_i}{2} \rfloor$, we get

$$X = \lfloor \tfrac{\hat{m}_i}{2} \rfloor .\, m_i + x_i,$$ by using (2),

Or

$$X = \frac{\hat{m}_i . m_i}{2} + x_i = \frac{M}{2} + x_i = \lfloor \tfrac{M}{2} \rfloor + x_i$$

since M is even.

For non-negative X, $X \leq \lfloor \frac{M}{2} \rfloor$,

therefore from above equation $x_i \leq 0$.

For $j = 0$, $x_i \geq 0$ (shown in case (a)).

Hence X is non-negative iff

$$\bigvee_{j=1}^{\lfloor \frac{\hat{m}_i}{2} \rfloor - 1} \{ x_k = /j.m_i + x_i/_{m_k}\} \vee \{x_k = /\lfloor \tfrac{\hat{m}_i}{2} \rfloor m_i + x_i/_{m_k} \wedge x_i \leq 0\} \vee \{x_k = /x_i/_{m_k} \wedge x_i \geq 0\}$$ is true.

$\longleftrightarrow$