AN FIR DIGITAL FILTER USING ONE-HOT CODED RESIDUE REPRESENTATION

Jimson Mathew and D. Radhakrishnan
School of Computer Engineering
Nanyang Technological University, Nanyang Ave., Singapore 639798
email: damu@ieee.org

ABSTRACT

Several Digital Signal Processing (DSP) structures based on Residue number systems (RNS) have been proposed in the technical literature. Most of them employ a look up table approach, that is highly space inefficient and slow. In this regard, we propose a memoryless high-speed error detecting FIR filter architecture using 1-out-of-n code for representing the residue numbers. With the use of 1-out-of-n code, error detection is achieved without any redundant moduli. The proposed design exhibits VLSI efficient layout, operand independent delay and low power consumption. We also propose a technique to achieve direct analog output from the system.

1 INTRODUCTION

Residue Number System (RNS) eliminates the speed-draining carry propagation in arithmetic computations, thus making it attractive in heavily computation intensive applications [1,2]. The parallel arithmetic nature of RNS offers a potential speed up in FIR filtering, discrete Fourier transforms, matrix multiplications, and similar other computations. Even though RNS provides very fast and reliable arithmetic due to its inherent parallel processing and error correction capabilities, its use is severely limited to a few applications due to the complexities involved in its conversion circuitry. An RNS is defined by a set of relatively prime integers \( m_1, m_2, \ldots, m_r \) called the moduli of the number system. Such a system provides unique representation of numbers from 0 to M-1, where \( M = \prod_{i=1}^{r} m_i \). Each integer \( X \) is represented by an \( r \) tuple \((x_1, x_2, \ldots, x_r)\), where each residue \( x_i = X \mod m_i \) is defined as the least remainder when \( X \) is divided by the moduli \( m_i \). In RNS, arithmetic operations on large integers are done by converting them into smaller residues and performing the operations independently and all in parallel, thereby speeding up the whole operation. In present systems an analog signal is first converted into binary and then a binary-to-residue converter is used to generate the residues. The residue results produced at the end of processing are finally converted back to binary. Two reverse conversion methods are generally used; one is based on Chinese Remainder Theorem (CRT) and is given by:

\[
X = \sum_{i=1}^{r} a_i x_i \bar{M}_i \mod M, \text{ where } \bar{M}_i = \frac{M}{m_i} \text{ and } a_i = \begin{bmatrix} 1 \\ \vdots \\ \frac{1}{M_i m_i} \end{bmatrix}.
\]

The second uses Mixed Radix Conversion (MRC) and is given by the following equation:

\[
X = A_r \prod_{i=r}^{1} m_i + \ldots + A_3 m_2 m_1 + A_2 m_1 + A_1,
\]

where \( A_i \in [0, m_i) \) and

\[
A_i = \begin{bmatrix} 1 \\ \vdots \\ \frac{1}{m_1 m_2 \ldots m_{i-1}} \end{bmatrix} (x_i - (A_{i-1} m_{i-2} \ldots m_1 + \ldots + A_1)) \mod m_i.
\]

In this paper an MRC based architecture is used for reverse conversion.

In a conventional RNS processor error detection is achieved by incorporating extra moduli called redundant moduli. This necessitates the use of additional modulo channels in the processor. In addition, these redundant moduli must be selected so that they are the largest ones in the moduli set. This increases the hardware complexity at every stage of the processor. To overcome this we use the property of totally self checking codes for error detection in our design. This is achieved by using 1-out-of-n codes for representing the residue numbers. The input values are therefore directly converted to 1-out-of-n codes which are then used to perform modulo arithmetic in each channel of the filter module.

2 FIR FILTER ARCHITECTURE

Finite Impulse Response (FIR) digital filters have attracted a great deal of interest because they are inherently stable structures which are much less sensitive to quantization errors than filters of the recursive type. The major disadvantage of an FIR design is that usually a large number of nonzero terms are required in the impulse response in order to adequately control the frequency response of the filter. The resulting large number of multiplications and additions which must be executed during the sample interval seriously limits the data rate of the filter. In this regard RNS based implementations are shown to have significant advantages. A number of FIR filters based on residue arithmetic are available in the literature [2] that are mostly look up table based implementations. On the otherhand, we propose a high speed memoryless RNS filter implementation using One Hot coded Residue (OHR) representation. A block diagram of such a filter is shown in Figure 1. It mainly consists of
the binary to OHR converter, the modulo filter modules and reverse converter to get the output in a useful form.

The general equation for an FIR filter is given as

\[ y(n) = \sum_{k=0}^{N} h(k) x(n-k) \]  

(2.1)

where \( x(n) \) and \( y(n) \) are the filter input and output respectively, and \( h(k) \) are the filter coefficients. To use RNS, first a moduli set must be selected which provides sufficient dynamic range for the filter. A single channel corresponding to modulus \( m_i \) of such a filter is shown in Figure 2. The filter coefficients are directly represented in residue form. In Figure 2, \( \oplus \) and \( \otimes \) represent modulo multiplication and modulo addition respectively. The basic building block used for these arithmetic operations is an OHR cell. Using one-hot coded representation of the residue digits, addition can be performed by cyclic shifts (rotations). Hence one of the operands is rotated by an amount equal to the other value. The rotation can be performed by barrel shifters. These circuits perform all possible rotations in parallel and pass the appropriate one to the output when required. The barrel shifters can be built using pass transistors or transmission gates.

This is achieved by using a set of prime moduli for the whole system. A useful property of any prime modulus \( p \) is that it has at least one primitive root. A primitive root is an integer \( \alpha \) of order \( p-1 \) under multiplication modulo \( p \). That is, it is an integer whose successive powers, taken modulo \( p \), generate all the nonzero integers modulo \( p \). For any integer \( x \), \( 0 < x < p \), \( x = \alpha^k \mod p \) for some some \( 0 \leq k \leq p-2 \). It is then said that \( x \) has index \( k \). A primitive root allows multiplication to be performed by adding indices modulo \( p-1 \), to the use of logarithms in the binary number system. As an example, let \( p=13 \). Then \( \alpha=2 \) is a primitive root because the quantities \( 2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9 \), and \( 2^{10} \) reduced to modulo 13 are equal to \( 1, 2, 4, 8, 3, 6, 12, 11, 9, 5, 10 \) and \( 7 \). The latter is the set of nonzero integers modulo 13. Furthermore, letting \( X = 5 = 2^2 \mod 13 \) and \( Y = 7 = 2^1 \mod 13 \), we have that \( XY = 35 = 9 \mod 13 = 2^8 \mod 13 \). But the index of the product can be determined by adding the operand indices modulo \( p-1 \), i.e., \( 9 + 11 = 8 \mod 12 \). If we restrict the moduli up to 5 bits, a dynamic range of 29 bits is possible using the moduli set \( \{11, 13, 17, 19, 23, 29, 31\} \).

Therefore, irrespective of the operation such as modulo multiplication or addition, the operational delay is a single gate delay/switching delay. Since each residue state in this architecture is represented by a separate active line, the coefficient ROM in the traditional filter design can be replaced by mere decoders which saves hardware and takes minimum delay. Moreover, all the modules with different bits/moduli can work synchronously because modulo multiply/add times are independent of their word length. Therefore, the moduli set selected need not be a balanced one. Here we assume that the dynamic range is selected in such a way that there won’t be any overflow upon performing the multiply accumulate operation.

### 3 BINARY-TO-OHR CONVERTER

In conventional RNS based designs, the inputs are first converted to residue form using binary-to-residue converters. A number of such converters are available in the literature [2-5]. A good majority of them makes use of ROM tables. On the contrary, our design uses a memoryless binary to OHR converter as shown in Figure 3. For convenience the input binary number is assumed to be \( q \) bits long. This is divided into \( l \) bit fields, where \( l = \lceil \log_2 m_i \rceil \). Each \( l \) bit field is decoded by a \( l:0 \), decoder. The decoding procedure is illustrated as follows: let us assume that \( q \) is an integer multiple of \( l \). Hence \( q = sl \), where \( s \) is an integer. The \( l \) bit fields are numbered from 0 to \( s-1 \), starting from LSB. The decimal equivalent of the number \( X \) stored in the \( q \) bit binary word is given by:

\[ X = \sum_{i=0}^{s-1} b_i 2^i = \sum_{i=0}^{s-1} x_i \times 2^i \text{.} \]

where the radix \( R = 2^i \) and \( x_i \) denotes the decimal value corresponding to the \( i \)th \( l \) bit field. The decoder for each \( x_i \) in addition to decoding the \( l \) bit pattern, also does a modulo conversion. The decoder for

---

**Fig. 1:** RNS Based Filter

**Fig. 2:** \( i^{th} \) Modulo Filter Module

Modular multiplication is based on index calculus approach \[9\], where multiplication is replaced by addition.
x₀ (Field 0) has mᵢ outputs (d₀, d₁, ..., dᵢ₋₁). The decoder for x₁ (Field 1) also has mᵢ outputs and are given by:

\[ d_j = 2^j \cdot x_1 \mod m_i \]

where \( j = 0, 1, ..., m_i - 1 \). It can easily be seen that the decoded output for Field 1 can now be obtained from a decoder which is identical to the one used for x₀ by permuting its outputs. Hence a total of s decoders are needed for each modulus. The decoder outputs are fed to the OHR cell. The function of the OHR cell is to perform a modulo addition with respect to mᵢ. A typical OHR cell is shown in Figure 4. It consists of an \((m_i \times m_i)\) array of NMOS transistors. For every input combination one of the transistors in the array is switched on to produce the output. Hence, OHR cell delay is only a single gate switching delay. The overall binary-to-OHR conversion time is only a few gate delays. In Figure 1, the block binary-to-OHR converter represents r independent channels each having a similar structure as shown in Figure 3.

4 OHR-TO-ANALOG FORM

The OHR outputs from the filter modules are first converted to mixed radix form using OHR cells and finally to analog form. The traditional mixed radix converter is based on look up tables [6]. In look up table implementation, the function of each table is to perform a subtraction and a multiplication by a constant, followed by a modulo correction with m. In general, if two inputs to a block are labeled as \( x_i \) and \( y_i \), then the output of the block is given by:

\[ z_i = (x_i - y_i) \mod m_i \]

where \( k \) is a constant. To do the same operation we use the OHR cell. Subtractor is implemented in the same way, like addition except that the subtrahend input bus is permuted to generate the additive inverse (modulo \( m_i \)) of its operand. Multiplication by a constant modulo \( m_i \) is done by wire transposition. The mixed radix coefficient in conventional form can be generated using an encoder. The encoder is nothing but a set of OR gates. A block diagram of the converter is shown in Figure 5. In an RNS with cardinality \( r \) the MRC conversion time is \((r-1)\) switching/gate delays and the hardware requirement is \( r(r-1)/2 \) OHR cells.

Now to incorporate error detection feature we use a totally self-checking (TSC) checker. A TSC checker is a kind of on-line checking circuit that is capable of detecting errors in the functional circuit as well as in the checker itself during normal operation [7, 8]. However, to use the checker the information has to be encoded. The conventional method is to define some inputs as valid inputs (called input codewords) and some outputs as valid outputs (called output codewords) so that the system can detect faults by observing whether the outputs are valid or not. The system and the code are designed such that under fault free situations, an input codeword to the system will produce a non-codeword.

In this architecture all the modules work on 1-out-of-n code and hence error checking can be postponed to the last stage. The error detection capability is built into the design due to its 1-out-of-n code nature, rather than by the addition of redundant moduli as in conventional redundant RNS architectures. The cells that output the MRC coefficients are checked for errors using a TSC checker. That is, for an \( r \) moduli system the \( r \) MRC coefficient channels are checked for errors using self checking checkers. This is done by first converting the 1-out-of-n code from each channel into a k-out-of-2k code, and subsequently to a 1-out-of-2 code. The conversion from 1-out-of-n code to k-out-of-2k code uses a set of simple OR gates. The k-out-of-2k code to 1-out-of-2 code conversion uses majority detection circuits. These 1-out-of-2 codes from the individual channels are finally combined to form a final 1-out-of-2 code, which in turn displays the error as shown in Figure 5.

The actual output of the system is generated in analog form from the weighted mixed radix coefficients by means of small digital-to-analog (D/A) converters and a summing amplifier. The multiplication by constants in MRC equation is done by adjusting the resistor values. The structure can
be modified without D/A converters. That is, the output of the MRC converter in 1-out-of-n code is directly summed up using an operational amplifier and the multiplication by a constant is done by adjusting the value of resistors. These outputs from r coefficients are finally summed up using an op-amp with unity gain. This is not shown in the figure.

CONCLUSION

An RNS FIR filter architecture with error detection capability is presented in this paper. The proposed architecture replaces all arithmetic modules in conventional designs by simple switching modules. The hardware overhead is comparable to that of the existing ones. The error detection feature is incorporated into the design without the use of any redundant moduli.

The overhead for error detection is about 10% of the overall hardware of the system. Apart from the error detection capability, the proposed architecture is totally memoryless, and exhibits regular layout and operand independent delay.

References


