ELSEVIER

Information Processing Letters 69 (1999) 249–252

Information Processing Letters

A parallel approach to direct analog-to-residue conversion

D. Radhakrishnan *, A.P. Preethy

Division of Computer Engineering, School of Applied Science, Nanyang Technological University, Singapore 639798, Singapore

Received 27 January 1998; received in revised form 22 October 1998
Communicated by D. Gries

Abstract

A novel design of a direct analog-to-residue converter is presented in this paper. The design makes use of two successive approximation analog-to-digital (A/D) converters, a few modulo adders and a small look-up table. One of the digital-to-analog converters is modified to generate outputs which are weighted by a constant factor, and one of the comparators is replaced by a difference amplifier. The look-up table needed is a very small percentage of the entire chip area and is shown to be only 840 bytes for a 36-bit residue number system converter.

Keywords: Parallel processing; RNS; Analog-to-residue converter; Modulo adder

1. Introduction

Residue number systems (RNS) are becoming very popular in many computation intensive applications like DSP [4]. 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 $x_i = X \bmod m_i$, 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 to residue using a binary-to-residue converter. The residue results produced at the end of the processing are finally reconverted back to binary. But these conversions from binary to residue and from residue to binary are highly involved operations for a general moduli set. Many researchers are therefore exploring new ways of tackling this problem—the one most sought after being the direct conversion of an analog signal to the residue form, an analog-to-residue (A/R) converter.

Recently a scheme for direct conversion from analog-to-residue form using a flash converter was proposed [2]. A flash A/D converter with n-bit resolution consists of $2^n - 1$ comparators and 2^n resistors which produce a thermometer code that is finally converted into binary using a decoder [1,3].

The flash A/R converter proposed in [2] uses a PLA, a few latches, a code converter, a set of buffers and XOR gates in addition to the comparators and resistor elements. The size of the PLA in this case is

$$(L + m_r - 1) \times \sum_{i=1}^{r} \lceil \log_2(m_i - 1) \rceil$$

* Corresponding author. Email: asdrkrishnan@ntu.edu.sg.

0020-0190/99/$ – see front matter
PII: S0020-0190(99)00012-5

bits, where

$$L = \left\lceil \log_2 \left(\frac{M}{m_r} - 1 \right) \right\rceil.$$

It is impractical to construct such converters for large values of n due the exponential growth of the PLA size, the number of comparators and resistors. Hence this cannot be used for converter sizes of more than 8–10 bits.

For almost all real world problems we need much higher resolutions than 8 bits. One promising approach will be to use a successive approximation type of converter. A block diagram of such a converter is shown in Fig. 1. A sample of the analog input X appears at one input of the comparator. The digital word being generated at the output register is converted to analog voltage using a D/A converter and compared with the analog input sample X. Starting with all zeroes, a first 'trial' digital value is generated by changing the MSB of the output register to 1. The D/A converter converts this to an analog value V_R, which is then compared with X. If $X < V_R$, the assumed bit of 1 is rejected and replaced by 0. Otherwise, the 1 is retained. Focus is next directed to the second most significant bit of the trial value and is continued in a similar fashion till all bit positions in the output register are compared. The completion of conversion is triggered by a change in the state of the comparator. At this time the conversion stops and the output register contains the digital word. The total time (worst case) needed for conversion is $n + 1$ clock cycles, as the trial bit has to traverse all n-bit positions of the output register. A successive approximation A/D converter is reasonably fast for many applications. Hence this is used as the basic element in our A/R converter.

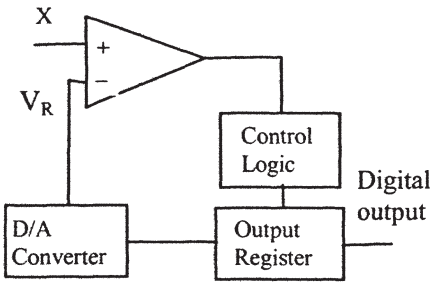

Fig. 1. A successive approximation A/D converter.

2. A successive approximation analog-to-residue (A/R) converter

A block diagram of the proposed A/R converter is shown in Fig. 2. This is obtained by cascading two separate successive approximation A/D converters. The sampled analog input voltage X is applied to the input of the first A/D converter. This stage is modified by replacing the comparator with a difference amplifier (Amplifier 1) and adding a weighing factor (the largest modulus m_r) to the output of the D/A converter. Hence the difference amplifier 1 produces an output voltage equivalent to $X - i * m_r$ during each successive step of the iteration, where i represents the value stored in Register 1. The size of this register is chosen to have k bits with

$$k = \left\lceil \log_2 \left(\frac{M}{m_r} - 1 \right) \right\rceil.$$

Once all the k-bit positions of Register 1 are identified, the difference amplifier output will become $X - R * m_r$, where R represents the value stored in Register 1. It can be easily verified that this voltage $X - R * m_r < m_r$. The first stage of conversion stops at this time.

The output of the difference amplifier 1 now represents the voltage equivalent of $x_r = X \bmod m_r$. This residue value x_r can be converted to a digital output using any one of the known A/D conversion techniques. In our design, a successive approximation A/D converter is used for this second stage also.

The second stage converter is exactly similar to the one shown in Fig. 1. The output register size for this converter is $l = \lceil \log_2 m_r \rceil$ bits. The output of this register represents the residue x_r corresponding to the input X with respect to the modulus m_r. It may be

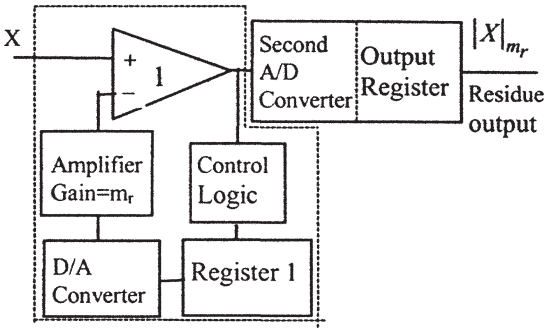

Fig. 2. A successive approximation A/R converter.

noted that the second stage conversion is to be delayed till the completion of the conversion in the first stage. Furthermore, the extra amplifier stage with gain m_r shown in Fig. 2 can be easily incorporated into the design of the D/A converter stage.

The thrust of the whole design depends on the generation of the remaining $r-1$ residues. This must be done at the cost of minimum extra hardware and minimum delay. Our proposed architecture is based on the observation that the contents of Register 1 (first stage A/D converter) together with the output x_r (second stage A/D converter) uniquely identifies the original input X, since $X = R * m_r + x_r$. Hence this will also identify uniquely each one of the remaining residues $x_1, x_2, \ldots, x_{r-1}$. One solution is therefore to use a PLA to generate the above residues, with Register 1 and x_r as inputs to the PLA. But the size of this PLA becomes prohibitively large even for very small word length RNS converters. Hence a completely different design strategy by partitioning Register 1 into a number of equal length partitions is used in this paper.

To efficiently use our technique, we assume that the moduli are all of equal word length with l bits to represent each. First let us consider the generation of residue x_j corresponding to the modulus m_j. A residue generator for x_j is shown in Fig. 3. Each partition group of Register 1 addresses a ROM that is programmed to produce its residue with respect to m_j. The programming of the ROMs is done as follows:

For an analog input sample equivalent to an integer X, let $X_1, X_2, \ldots, X_n$ represent the decimal equivalents corresponding to the l-bit segments of Register 1 as shown in Fig. 3. Then X is given by:

$$X = x_r + \left[X_1 + X_2 2^l + \cdots + X_n 2^{(n-1)l}\right] m_r, \quad \text{and}$$
$$|X|_{m_j} = \left|x_r + |X_1 m_r|_{m_j} + |X_2 2^l m_r|_{m_j} + \cdots + |X_n 2^{(n-1)l} m_r|_{m_j}\right|_{m_j}. \tag{1}$$

The ROMs corresponding to each term in Eq. (1) are programmed to provide the residue outputs when they are addressed by X_i. These residues are generated simultaneously and are added using modulo adders to get the final residue. Modulo correction for x_r is not needed since it is directly fed to a modulo adder. A total of $r-1$ similar stages are needed to generate all the remaining residues $x_1, x_2, \ldots, x_{r-1}$. Hence a parallel processing approach is used for the generation of these residues.

An example to illustrate the above residue generation scheme is given below.

Example 1. A 36-bit RNS converter can be implemented using eight 5-bit moduli 17, 19, 23, 25, 27, 29, 31 and 32, with $m_r = 32$. The length of the registers for the first and second stage converters are $k = 31$ and $l = 5$, respectively. Register 1 is divided into 7 partitions, six of 5 bits each and the last partition with a single bit. An analog input sample X equivalent to an integer 40 can be expressed as: $1 * 32 + 8$ in modulus 32. Hence the contents of Register 1 is $R = 1$ and the Output Register value $x_r = 8$. For a modulus $m_j = 23$, the right most adder, modulo 23, in Fig. 3 receives 2 inputs, one is '8' from the Output Register ($x_r = 8$), and the other '$|1 * 32|_{23} = 9$' from the ROM corresponding to the rightmost partition of Register 1. Hence the output from the modulo adder is $|9 + 8|_{23} = 17$ which agrees with the value $|40|_{23} = 17$.

The conversion time for generating residue x_r equals the sum of the conversion times in both the A/D converter stages. This time is given by $k + l + 2$ clock cycles, where k and l are the sizes of the registers in the first and second stage A/D converters, respectively. A conventional successive approximation A/D converter of the same word length uses more or less the same number of clock pulses for conversion of an analog signal to binary. The maximum number

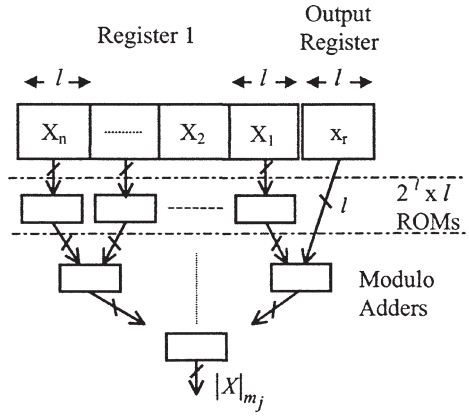

Fig. 3. Residue generation logic for x_j.

of extra clock pulses needed for the A/R converter will be 3 in the worst case.

The generation of the remaining residues needs additional delay. This extra delay is due to the small ROM look-up tables and the modulo adder stages. If Register 1 is divided into 'p' partitions of l bits each, then the number of modulo addition stages is $\lceil \log_2(p+1) \rceil$. But as can be seen from Fig. 3, a large portion of these delays can be made to overlap with the conversion time of the second stage. Hence all the residues are generated almost simultaneously.

The hardware requirements for the residue generation logic for a 36-bit A/R converter is given in Example 2.

Example 2. Consider the same moduli set used in Example 1. The generation of the first residue $|X|_{32}$, for any integer X within the range of 36 bits, needs a total of 38 clock pulses. Register 1 is divided into 7 partitions, six of 5 bits each and one last partition with a single bit as in Example 1. Each 5-bit partition uses a ROM of size 32×5 to generate the residues, thereby requiring a total of 192×5 per modulus. This amounts to a total of 840 bytes of storage for the entire system. A simple logic circuit can be used with the MSB bit to generate its equivalent residue value. The converter also need seven 5-bit modulo adders per modulus, thereby requiring a total of 49 for the entire set.

3. Conclusion

The design of a direct A/R converter using two stages of successive approximation A/D converters is presented in this paper. The modifications needed in the A/D converters are very minimal, one of the comparators being replaced by a difference amplifier, and one D/A converter output being modified by adding an extra weighing factor. For generating all the residues, extra memory storage and a few modulo adders are used. For a 36-bit converter this amounts to only 840 bytes of storage and forty nine 5-bit modulo adders.

The overall conversion speed of the proposed A/R converter is found to be very close to that of a similar A/D converter since all the residues are generated in parallel. This eliminates the extra delay due to the binary-to-residue converter stage used in conventional approaches. Furthermore, the conversion speed of the second stage A/D converter can be increased by using a flash A/D converter. This is possible since the resolution of this stage is usually less than 8 bits (5 bits for a 36-bit converter) even for a very large sized converter.

The ROM size can be reduced further by using multiple access techniques. By this method the ROM size for a 36-bit converter can be reduced to a mere 140 bytes of storage. But this in turn increases the overall delay for residue generation. This delay could be offset by using pipeline stages in the ROM-adder set up.

Acknowledgments

This work was supported in part by the NTU Research Grant under AcRF RG-22/95. The authors wish to thank the anonymous reviewers for their objective comments which have significantly enhanced the quality of this paper.

References

[1] D.W. Cline, Noise, speed, and power tradeoffs in pipelined analog-to-digital converters, Ph.D. Thesis, University of California, Berkeley, CA, 1995.

[2] S. Mandyam, T. Stouraitis, Efficient analog-to-residue conversion schemes, in: Proc. IEEE International Symposium on Circuits and Systems, New Orleans, LA, May 1990, pp. 2885–2888.

[3] G.F. Miner, D.J. Comer, Physical Data Acquisition for Digital Processing, Prentice-Hall, Englewood Cliffs, NJ, 1992.

[4] M.A. Soderstrand, W.K. Jenkins, G.A. Jullien, F.J. Taylor, Residue Number System Arithmetic: Modern Applications in Digital Signal Processing, IEEE Press, New York, 1986.