ON RESIDUE NUMBER A/D AND D/A CONVERTERS

G. J. Lipovski
Department of Electrical Engineering
University of Florida
Gainesville, Florida

ABSTRACT

A very simple analog to digital converter and digital to analog converter is described for residue number digital processing. These simple devices make it feasible to replace analog components with comparitively inexpensive digital processors that use residue, or modulus, arithmetic capable of operating at very high speeds. Using off-the-shelf integrated circuits, addition, subtraction or multiplication of about 15 bits of accuracy can easily be done in as little as fifty nanoseconds. Any function using these operations (polynomial expansions, linear filters, fast fourier transforms) can be economically implemented in a pipeline or other structure to get very fast systems. Moreover, a stage in the pipeline can correct for nonlinearities in the A/D or D/A converters. The simple devices described herein make residue arithmetic digital processors extremely attractive for use in fast analog systems.

INTRODUCTION

A number of instrumentation and control systems require a considerable amount of processing. For example, digital filters and fast fourier transforms are used in signal processing and control. In computer controlled engines, complex polynomials are evaluated in order to determine the amount of fuel that is injected into the engine cylinder. Although the input and output for the process is usually an analog voltage, the signals are generally converted to digital form for processing by a minicomputer, or more recently, by a microcomputer. The computer is used because of its capability to add, subtract and multiply quickly and accurately. However, the minicomputers are too expensive, and the microcomputers are too slow for many of the kinds of problems mentioned above.

It has long been known that residue, or modulus, arithmetic is very fast and is economical to implement for addition, subtraction and multiplication. This should make them attractive for the problems mentioned above. The problem with this form of arithmetic is that division, comparison and decoding residue numbers are all quite difficult. Herein,it is shown that decoding is actually very easy, especially when the decoded output is analog (i.e., a simple D/A converter is shown.) A corresponding A/D converter is also shown. This simple technique makes residue number processors extremely attractive, at least for processors whose input and output are analog voltages.

This paper describes a coincidence of three very simple ideas from diverse areas of Computer Engineering that together contribute to a potentially important result. Even though the basic concepts in this paper can be explained on one short paragraph, it is likely that the reader will be unfamiliar with at least part of the necessary background. Therefore residue numbers, digital processing, and A/D converters will each be briefly covered.

RESIDUE NUMBERS

Herein, the characteristics of residue number addition, subtraction, multiplication, and incrementation are briefly discussed. To accurately describe these characteristics, a simple language will be used. Integers will be denoted as lower case symbols: c[1 to n] is an n element vector of integers. As in APL, +, -, x, *, ⍟, ⌈, | and , are / addition, subtraction, multiplication power, logarithm, maximum, modulus and concatenation. a|b is b modulo a. a * b is a to the power b, a⍟b is b log to the base a, and a ⌈ b is the maximum of a,b. For example, 2|5 is 1. These operations are extended to vectors. If a[1 to 3] is 2,3,5 and b[1 to 3] is 7,7,7 then a[1 to 3]|b[1 to 3] is 1, 1,2. Scalars are expanded to vectors when they are operated on with vectors. If c is 7 then a[1 to 3]|c is 1,1,2. Finally, / indicates an operator reduction over a vector. For instance, x/a[1 to 3] means a[1]x a[2]xa[3]. Or ⌈/a[1 to 3] is the maximum of a[1], a[2], a[3]. These simple APL constructs make the following explanation very simple.

A residue number is defined by a modulus vector m[1 to n] whose elements are relatively prime to each other. For reasons explained in the next section, a good modulus for about eight bits of accuracy is 15, 16, and for about fifteen bits, 11, 13, 15, 16. A number c is represented by the n element vector m[1 to n]|c. For example, the representation of 0 is 0,0; of 1 is 1,1; of 100 is 5,4 for the first modulus vector given above. Any number between 0 and (x/m[1 to n]) - 1 can be uniquely represented modulo m[1 to n]. Two modulus m[1 to n] numbers, a[1 to n] and b[1 to n], can be added using the formula m[1 to n]|(a[1 to n] + b[1 to n]). This means that for each i, the ith element of the sum is a[i] added to b[i] modulo m[i]. Note that a[1 to n] is simply incremented by the formula m[1 to n]|(a[1 to n] + 1). Similarly, the product of a[i] times b[i] is m[1 to n]|(a[1 to n] x b[1 to n]). This means that the ith element of the product is a[i] times b[i] modulo m[i]. Note that addition, incrementing,subtraction and multiplication are done element-by-element without shifting carries or borrows between elements.

DIGITAL PROCESSING HARDWARE

Residue numbers can be expressed by encoding each element in binary. Some notation to handle bit vectors and arrays is now introduced. Upper case symbols will denote binary variables. A[1 to n] is an n bit binary vector. B[1 to h; 1 to k] is an h word, k bit per word array (e.g. a memory). Finally, B[A[1 to p];] denotes the entire row of arry B which is addressed by binary vector (number) A[1 to p].

The modulus representation a[1 to n] can, in turn, be represented as a binary array A[1 to n; 1 to p] where p is ≥ 2 ⍟ (⌈/m[1 to n]). For example, as 100 is represented in modulus 15, 16 numbers by 5,4, it is represented now by the binary array:

$$\begin{bmatrix} 0101 \\ 0100 \end{bmatrix}.$$

Addition, subtraction, and multiplication can be done by table lookup for each element. The tables can be stored in read-only memories (ROM's) or more easily used programmable read-only memories (PROM's). Each element needs only one table (ROM or PROM) having a surprisingly small number k ≥ (m[i]*2) rows and k ≥ (2⊛m[i]) columns for storing the addition table, subtraction table, or else the multiplication table. Suppose ADDi [0 to h-1; 1 to k] is the addition table for the ith element of modulus in [1 to n] numbers, that was formed as follows: To add any two numbers a, b, their bit patterns A[1 to k] and B[1 to k] are concatenated to get the address A[1 to k], B[1 to k] of a row in ADDi wherein the bit representation of sum m[i] |(a + b) has already been stored. Then once these tables have been stored, the ith row of the representation of the sum of a and b is simply ADDi [A [i to k], B[1 to k];]. Multiplication and subtraction are similarly easily carried out by table lookup.

The significance of table lookup can be appreciated when the rapid drop in the price of PROM integrated circuits is seen. Fast, cheap PROM's have been developed because PROM's are the easiest way to design control modules in computers, and control modules have to be faster than anything they control. The "eight bit" modulus 15,16 adder requires two PROM's, ADD1[0 to 255; 1 to 4] and ADD2[0 to 255; 1 to 4] that have size 256 x 4 (a conventional binary adder can be used in place of a PROM if the modulus is a power of 2). Similarly, the corresponding multiplier uses two PROM's, MULT1 [0 to 255; 1 to 4], MULT2[0 to 255; 1 to 4]. A 256 x 4 signetics PROM (82 S 1291) is now available in small or large quantities for under $4.00. And it has a maximum access time of 50 nanoseconds. Faster, more expensive, (ECL) PROM's are also readily available with 15 nanosecond access times. Note that the adder or multiplier can be built for about $10.00 each and can add or multiply in about 50 nanoseconds. Similarly, the "fifteen bit" modulus 11,13,15,16 adder or multiplier can each be built for about $20.00 and can add or multiply in about 50 nanoseconds. As was suggested earlier, the 15,16 modulus or 11,13,15,16 modulus number forms are desireable because their addition and multiplication tables can be effectively put in fast, inexpensive, readily available PROM's.

The residue adder or multiplier can be connected to standard microcomputers as I/O devices. Two registers (A[1; 1 to 4], A[2; 1 to 4]) and (B[1; 1 to 4], B[2; 1 to 4]) can be loaded by the microcomputer, and then the sum ADD1[A[1;], B[1;];], ADD2[A[2;], B[2;];] can be read into the microprocessor using one input device location. Also the product MULT1[A[1;], B[1;];], MULT2[A[2;], B[2;];] can be read into the microprocessor using another input device location. Programs to load the input registers and read the outputs of the PROM's can evaluate any formula involving addition, multiplication and subtraction. These include polynomials +/a[1 to n] x (b * (n-1,...,0)), linear expressions +/a[1 to n] x b[1 to n] used in linear filters and graphical transformations, and fast-fourier transform operations.

The extreme simplicity and low cost of these adders and multipliers makes it attractive to implement hard-wired logic with them as building blocks. For example, the polynomial above can be evaluated by repetatively using the formula p' = a + (b x p). The output of the multiplier can directly feed the input of the adder, so that for the ith element, the output is ADD1[A[1;], MULT1[B[1;], P[1;];];], ADD2[A[2;], MULT2[B[2;], P[2;];];].

By holding the variable B in a latch, supplying coefficients A from a shift register, and updating P with the result of this operation, an eighth order polynomial can be evaluated in about one microsecond, at a cost of about $30.00.

One last observation that is critical to the D/A converter is that a residue number can be incremented using the formula m[1 to n]|(a[1 to n] + 1). This means the binary representation A[i;1 to p] of each element a[i] can be put into a conventional modulo m[i] binary counter. The whole number is incremented by merely clocking each counter at the same time.

It should be noted that table lookup is feasible using modulus arithmetic that uses small modulus elements. Ordinary binary addition or multiplication would require enormous tables. For example, eight bit multiplication would require a 64k by 8 PROM. It should also be noted that residue incrementing is easier to carry out in a counter than conventional binary number incrementing because the clock goes into all the (small) counters at the same time rather than rippling through one counter into the next counter. There are no substantial propagation delays to slow down the counter.

All of these concepts are well known or are easily observed. They invite the use of residue arithmetic in many applications.

However, there always was a problem converting residue numbers to some other form. The next section shows that it is easy to convert the residue number into a voltage, or vice versa.

A/D AND D/A CONVERTERS

The A/D converter converts an analog voltage into a digital binary number, and the D/A converter does the inverse operation. A/D conversion can be accomplished in one of two basic ways. Successive approximation is somewhat like binary division in that one bit of the output binary number at a time, most significant bit first, is generated. Actually, an analog voltage is generated like the partial remainder and its sign is used to determine the next converted bit, which is like the quotient. Since conversion time is dependent on the number of bits produced in the binary number, it is quite fast. Ramp converters, on the other hand, simultaneously generate a voltage ramp (saw tooth wave form) as a counter is incremented. The binary number in the counter is always equal in value to the voltage on the ramp. By means of a voltage comparator, when the input voltage equals the ramp voltage the counter value is loaded into a register as the output binary number. Various different schemes use the ramp technique, such as dual ramp converters found in digital voltmeters, and so on.

A D/A converter can be implemented by a resistor ladder network with some analog switches, or by means of a ramp and counter. While the former scheme is faster, the latter scheme is less costly and can share the logic of the ramp A/D converter. Basically, to output a voltage equal to a binary number, a binary comparator and a sample-and-hold are used. As the ramp voltage and count increase, when the count equals the binary number, the ramp voltage is sampled and held until the next sample.

A conventional binary A/D converter can be modified to provide the residue number by simultaneously accumulating the residue representations of the values of the bits generated for the binary number output. However, no fast corresponding D/A converter is known. Nevertheless, ramp type A/D and D/A can easily be adapted.

The whole point of this paper is this: Replace the binary counter by a simpler residue number counter in ramp type A/D and D/A converters and take advantage of the properties of residue arithmetic.

A "fifteen bit" A/D and D/A converter would consist of a ramp generator, four counters that are modulo 16, modulo 15, modulo 13 and modulo 11 counters, and a voltage comparator, a binary comparator, and a sample-and-hold circuit. See figure 1.

The A/D converter works thus: The ramp voltage rises from 0 to (x/m[1 to 4]) -1 millivolts as x/m[1 to 4] clock pulses cause the four counters to increment. When the input becomes less than the ramp, the counter

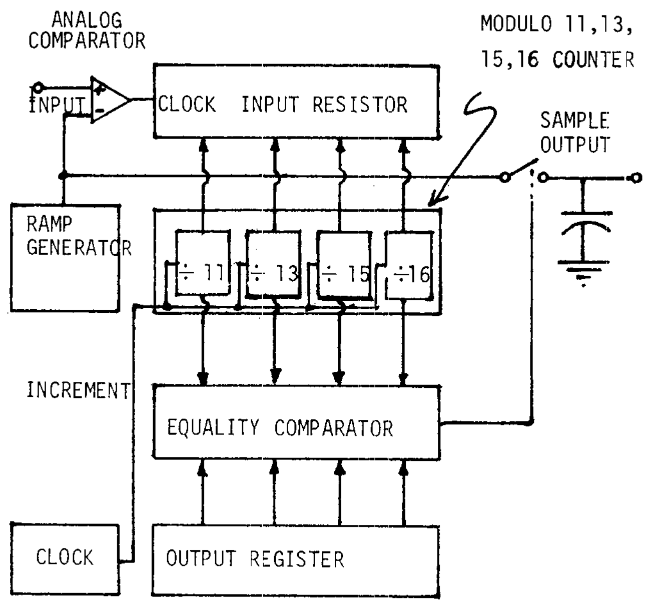

Figure 1. A/D and D/A Converter

values are loaded into the input register, where they are available as the residue number representation of the input voltage (in millivolts). To output a residue number, it is put into the output register. When it agrees, bit by bit, with the counter values, the sample-and-hold switch is closed to sample the ramp voltage.

It should be noted that the comparatively expensive ramp generator and counters can be shared between input and output systems. More input systems can be added simply by replicating the voltage comparator and input register. More output systems can be added by replicating the output register, equality comparator and sample-and-hold circuit.

Since 1GHz counters are available, "15 bit" converters can conceivably be built with sample times of about 30 microseconds, and "8 bit" converters can be built with sample times of about 250 nanoseconds. Although it is difficult to build ramp generators and especially sample-and-hold circuits at these frequencies, the power of residue arithmetic should be very useful in "linearizing" these components. It should be noted that the accuracy of most ramp type A/D and D/A converters depends strongly on the linearity of the ramp. Highly linear ramps are attainable using operational amplifiers to generate a constant current to charge a capacitor. Unfortunately, these operational amplifiers are subject to temperature and aging problems. However, a simple RC network can generate a very stable exponential ramp. The converted number x[1 to n; 1 to p] can be corrected by applying a polynomial expansion of the function e⊛(x[1 to n; 1 to p] - 1). Actually, any monotonic highly repeatable non-linear ramp can be used. Similarly, the output can be precorrected by a formula that takes into account the nonlinearity of the ramp and the inaccuracies of the sample-and-hold circuit. The actual correction formula can be attained by simply calibrating the circuit once it is built. Note that the hysteresis of the output capacitor can also be corrected by a linear filter formula. Thus, using residue arithmetic, it is possible to run ramp-type A/D and D/A converters well beyond their rated capabilities for speed or accuracy.

Finally, the use of pipelining can be effective where a high sampling rate is required. Suppose an FFTprocessor is to be designed. The A/D converter is the first stage, the polynomial evaluator to correct for nonlinearity is the second, the FFT processor (which itself may be pipelined) is the third, the polynomial evaluator/linear filter is the fourth, the D/A converter is the fifth stage. The sampling rate is limited by the maximum time for any of the five operations rather than the sum of these times. It is entirely reasonable to expect a FFT processor to keep up with the TV scan rate to produce real time speech plots, where the intensity shows the frequency components as a function of time.

CONCLUSIONS

Simple A/D and D/A converters have been described that invite the use of residue arithmetic in digital processors that interface with analog systems. These processors should find application in fast-fourier transform, and linear filter processors, as well as controls for fuel injection systems in automobiles. Due to the development of inexpensive fast programmable read-only memories, these processors should be both an order of magnitude faster and an order of magnitude cheaper than current systems.

REFERENCES

1) Szabo, N.S., and Tanaka, R.I., Residue Arithmetic and Its Applications to Computer Technology, McGraw Hill, 1967.

2) Flores, I., The Logic of Computer Arithmetic, Prentice Hall, 1963.