Low Power Realization of FIR Filters Implemented Using Distributed Arithmetic

Mahesh Mehendale
Texas Instruments (India) Ltd
Golf View Homes, Murugeshpalya
Bangalore, 560017, India
Tel: 091-80-5269451
Fax: 091-80-5269456
e-mail: mhm@india.ti.com

Amit Sinha
Dept. of Electrical Engineering
IIT Delhi
New Delhi, 110016, India
Tel: 091-11-6861799
Fax: 091-11-6861799
e-mail: prsingh@ee.iitd.ernet.in

S. D. Sherlekar
Dept. of Comp. Sc. & Engg
IIT Bombay, Powai
Mumbai, 400076, India
Tel: 091-80-5281461
Fax: 091-80-5284396
e-mail: sds@sasi.com

Abstract— We present a technique for low power realization of Finite Impulse Response (FIR) filters implemented using Distributed Arithmetic. In most applications, the distribution profile of input data values is known. The proposed technique uses a data encoding which can be tuned to the specific distribution profile so as to reduce toggles in the shift register chain. We present a generic Nega-Binary coding approach and show how a specific Nega-Binary scheme can be derived to achieve maximum power reduction. We also show how the binary to Nega-binary conversion can be performed bit-serially with minimal area (and hence power dissipation) overhead. The paper finally presents a shift-free implementation which uses memory array to store data values. We present a technique based on Gray coded addressing to reduce the power dissipation in such implementations.

I. Introduction

Finite Impulse Response (FIR) filters are one of the most common components of digital signal processing systems. FIR filtering involves a convolution of input data samples with the unit impulse response of the filter [1]. The output Y[n] of an N-tap FIR filter is a weighted sum of the preceding N inputs.

$Y[n] = \sum_{i=0}^{N-1} A[i] \cdot X[n-i]$

High speed filtering generally involves a dedicated hard-wired implementation of the filter. Distributed Arithmetic (DA) provides a multiplier-less implementation of FIR filters, with the flexibility that the filter coefficients are programmable [2]. It uses a bit serial computation that forms an inner product of a pair of vectors in a single step by storing all possible intermediate computations in a look-up table (LUT). Fig. 1 illustrates a typical DA based implementation of a 4-tap FIR filter.

The rightmost bits in the shift registers constitute the address for the LUT. Data is shifted every clock cycle and the LUT outputs are shifted and accumulated. This is done N times where N is the precision of the input data

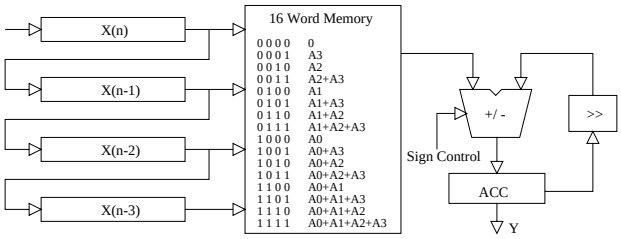

Fig. 1. DA based Implementation of a 4-tap FIR Filter

(and hence the length of the shift registers). At the end of every N clock cycles, the output is tapped at Y. For a 2's complement representation, the Sign Control is always positive except for the MSB i.e for the N^{th} clock cycle.

Substantial power consumption occurs as a result of toggles occurring in the shift registers every clock cycle. In most applications the distribution of data values is known. In this chapter we propose a data coding scheme which for a given distribution profile of data values, results in lesser number of toggles in the shift registers. The main constraint is that we do not want a scheme which results in toggle reduction with excessive hardware overhead (implying power dissipation overhead as well). The second important constraint is that the coding scheme should be programmable so that the same hardware can be used for different distribution profiles of data values. The Nega-Binary scheme that we propose satisfies these two constraints and can be directly incorporated into the structure of the DA based FIR shown in Fig. 1. To the best of our knowledge, such a Nega-Binary coding scheme to reduce toggling has not been exploited before.

Instead of shifting the data every cycle and accessing it from the same bit location, the data can be kept fixed and the pointer moved every cycle to access data from the desired bit location. The shift-register chain can then be replaced by a memory array with an appropriate column decoder. For such a scheme, we propose a Gray coded addressing to minimize the power dissipated in the column

decoder. Further, since there are several possible Gray codes for any given number of bits, we can choose a code which also minimizes the number of toggles in the data itself. We present results to demonstrate the power savings possible using these schemes.

The rest of the paper is organized as follows. Section II deals with Nega-Binary coding of data values and illustrates some typical data value distributions that are frequently encountered. Section III deals with Gray coded sequencing in shift-free implementations. In section IV we present the results of applying the above mentioned schemes to different types of data distributions. Finally, section V concludes this chapter with key observations.

II. Toggle reduction in DA based FIR Filters using Coding

Any coding scheme that seeks to reduce toggling must meet the following criteria:
1. It should result in minimum hardware overhead.
2. It should represent the entire range of values of the source data being coded.

The generic Nega-Binary scheme that we propose meets the above two requirements. The scheme that we propose has the added flexibility of choosing one of the several possible Nega-Binary schemes that meet the above criteria and also results in maximum toggle reduction.

A. Nega-Binary Coding

Nega-Binary numbers [5] are a more generic case of a 2's complement representation. Consider an N bit 2's complement number. Only the most significant bit (MSB) has a negative weight while all others have a positive weight. An N-bit Nega-Binary number is a weighted sum of $\pm 2^i$. As a special case consider a weighted $(-2)^i$ series where nb_i denotes the ith bit in the Nega-Binary representation of the number.

$Number = \sum_{i=0}^{N-1} nb_i \cdot (-2)^i$

In the above case, powers of 2 alternate in signs. While the 2's complement representation has the range of [-2^{N-1} , $2^{N-1}-1$], this Nega-Binary scheme has the range of [$-(4^{\lfloor N/2 \rfloor}-1)/3$, $(4^{\lceil N/2 \rceil}-1)/3$]. It can be noted that in general the Nega-Binary scheme results in a different range of numbers than the 2's complement representation. Thus there can be a number that can have an N bit 2's complement representation but cannot have an N bit Nega-binary representation. We address this issue in section II.B. We now present a simple example that demonstrates how the Nega-binary scheme can result in reduced number of toggles. Consider the 2's complement number 01010101_B. Using a Nega-Binary scheme with alternating positive and negative signs (weights $-(-2)^i$), the corresponding representation will be 11111111_{NB}. Clearly while the first case has maximum possible toggles the second one has minimum toggles. If instead the number was 10101010_B, this Nega-binary scheme would result in a representation with same number of toggles as the 2's complement. However, a different Nega-Binary scheme (weights $(-2)^i$) will have a representation 11111110_{NB} with just 1 toggle. Thus it can be noted that different Nega-Binary schemes have different 'regions' in their entire range which have fewer toggles and hence depending on the data distribution we have the flexibility of choosing that scheme which minimizes toggling without altering the basic structure of the DA based FIR.

In existing literature [5], the term Nega-Binary is used specifically for binary representations with radix -2. In this chapter, we have extended the definition of the term to encompass all possible representations obtained by using $\pm 2^i$ as the weight for the i^{th} bit. Thus for an N-bit precision there exist 2^N different Nega-Binary schemes.

B. 2's Complement vs Nega-Binary Representation

Since the range of values for the two representations are different we need to increase the bit precision for the Nega-Binary scheme to N+1 in which case there are 2^N+1 cases where the 2's complement range is a subset of the Nega-Binary range. With N+1 bits of precision, when all sign bits are negative, the corresponding Nega-Binary range is [$-2^{N+1}+1$, 0] and likewise when all the sign bits are positive, the range is [0 , $2^{N+1}-1$]. All intermediate sign combinations have a range lying between [$-2^{N+1}+1$, $2^{N+1}-1$] and each combination represents 2^{N+1} consecutive numbers. The N-bit 2's complement range being [-2^{N-1} , $2^{N-1}-1$] overlaps and completely lies within the N+1 bit Nega-Binary range for exactly 2^N+1 different Nega-Binary representations out of the possible 2^{N+1} total cases.

The advantage of using such a scheme is that we can choose a Nega-Binary representation that minimizes the number of toggles in the data values while covering the entire range spanned by its 2's complement counterpart. In most applications, the data distribution profile is known. This fact can be exploited to choose a Nega-Binary scheme, out of the ones which overlap with the 2's complement representation, such that it minimizes the total weighted toggles i.e. the product of the number of toggles in a data value and the corresponding probability of its occurrence.

Fig. 2 illustrates the distribution profile of a typical audio data extracted from an audio file. The non-uniform nature of the distribution is at once apparent. A Nega-Binary scheme which has the minimum number of toggles in data values with very high probability of occurrence will substantially reduce power consumption. Further, each of the 2^N+1 overlap cases have different 'regions' of minimum toggle over the range, which implies that there exists a Nega-Binary representation which minimizes total weighted toggles corresponding to a data distribution peaking at a different 'region' in the range. While the relative data distribution of a typical audio data is similar to that shown in Fig. 2, its mean can shift depending on

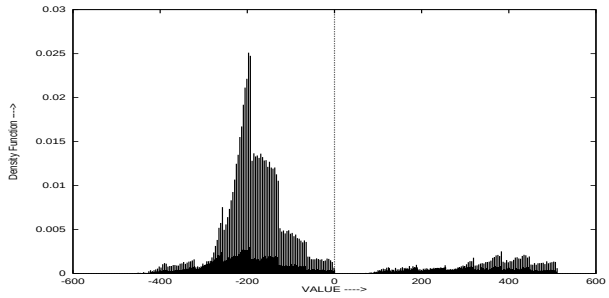

Fig. 2. Typical audio data distribution for 25000 samples extracted from an audio file

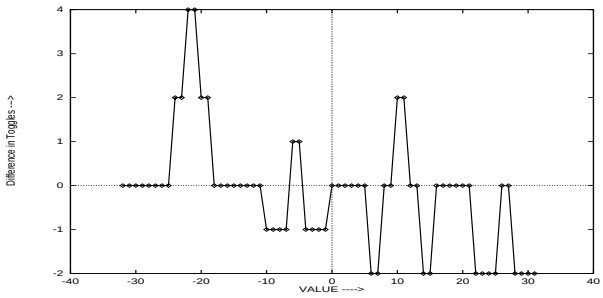

Fig. 3. Difference in toggles for N=6, 2's complement and Nega-Binary scheme : - + + - + - +

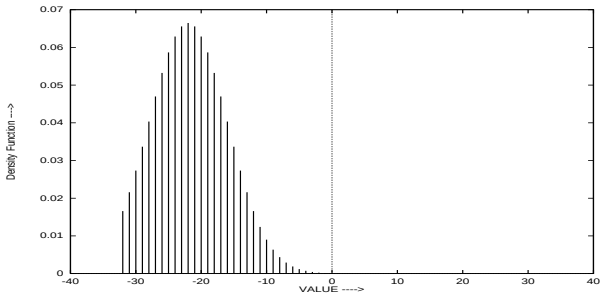

Fig. 4. Gaussian distributed data with N=6, mean=-22, SD=6

factors such as volume control. The flexibility of selecting a coding scheme depending on the 'mean' values is hence very critical for such applications. We show in section II.C that the binary to Nega-Binary conversion can be made programmable so that the desired Nega-Binary representation can be selected (even at run-time) by simply programming a register.

It can be noted that the toggle reduction using the Nega-Binary coding comes at the cost of an extra bit of precision. The amount of saving hence reduces as the distribution becomes more and more uniform. This is to be expected, as any exhaustive N-bit code (i.e one that comprises of all possible combinations of 1s and 0s) will necessarily have the same total number of toggles (summed over all its representations) as any other similar code. Therefore, as the data distribution becomes more and more uniform i.e. all possible values tend to occur with equal probability, toggle reduction decreases.

Fig. 3 illustrates the difference in number of toggles for a 6-bit, 2's complement representation and a 7-bit, Nega-Binary representation for each data value. Fig. 4 shows a profile for 6-bit Gaussian distributed data. As can be seen the Nega-Binary scheme of Fig. 3 can be used effectively for a distribution like the one shown in Fig. 4, resulting in 34.6% toggle reduction. Fig. 3 depicts one out of a total of 65 possibilities. Each of these peaks (i.e. the corresponding Nega-Binary scheme has fewer toggles compared to the 2's complement case) differently, and so there will exist a scheme that will perform better than the 2's complement scheme, for a given distribution.

C. Incorporating a Nega-Binary Scheme into the DA based FIR

Conversion of a 2's complement number to a Nega-Binary representation can be done bit-serially. We now present a pseudo-code to do the same; b_i's represent the 2's complement bits, nb_i's are the Nega-Binary bits, $sign_i$'s are the signs for the particular Nega-Binary representation chosen.

$c_0 = 0$;
for(i = 0; i < N; i++) {
    $nb_i = b_i$ .XOR. c_i;
    if($sign_i$ == '+') $c_{i+1} = b_i$ .AND. c_i;
    else $c_{i+1} = b_i$ .OR. c_i; /* $sign_i$ == '-' */
}

The above algorithm can be directly implemented in hardware resulting in a small area overhead. Data values can be bit serially converted from a 2's complement representation to a Nega-Binary representation and loaded into the shift registers. The sign bits can be directly coupled to the Sign Control of the adder shown in Fig. 1. Fig. 5 illustrates the complete Nega-Binary DA based FIR architecture. The 'sign' register is a programmable register which holds the sign combination for the chosen Nega-Binary scheme. The bit serial Nega-Binary computation logic requires just 5 gates and has a typical power dissipation equivalent to 2 flip-flops, which is negligible compared to the number of flip-flops in the shift register chain.

It is important to note that a simple difference of weighted toggle sums obtained for the 2's complement and the Nega-Binary representation does not give the actual toggle reduction occurring in the concatenated registers. Since the Nega-Binary registers have N+1 bits of precision, each data value contributes its toggles (N+1)/N times more than the corresponding 2's complement value. Therefore, the Nega-Binary weighted toggle sum needs to be multiplied by a factor equal to (N+1)/N.

Hence, we have the following estimate for the power saving.

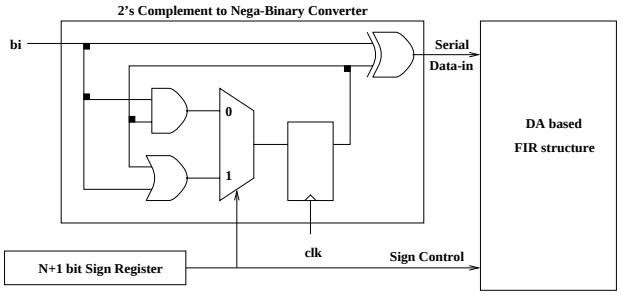

Fig. 5. DA based FIR architecture incorporating the Nega-Binary scheme

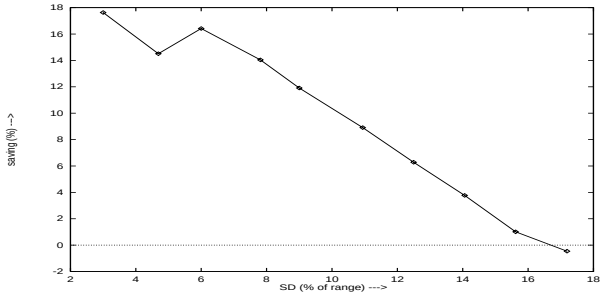

Fig. 6. Saving vs SD plot for N=8, Gaussian distributed data with mean=max/2

$$saving = \frac{\sum_i p(i) \cdot togs(i) - (\frac{N+1}{N} \cdot \sum_i p(i) \cdot negatogs(i))}{\sum_i p(i) \cdot togs(i)}$$

where p(i) is the probability of occurrence of a data with value i, N is the 2's complement bit-precision used, togs(i) and negatogs(i) are the number of toggles in the representation of i for the 2's complement case and the Nega-Binary case respectively.

The above saving computation does not account for 'inter-data' toggles that result from two data values being placed adjacent to each other in the shift register sequence. It may be observed that for a T-tap filter with N-bit precision registers an architecture similar to Fig.1 would imply a virtual shift register (obtained through concatenating all the individual registers) of length TxN.

We performed actual shift simulations sample by sample for different data profiles and different number of samples to find out the Nega-Binary scheme that results in maximum saving. These simulations showed that in all cases, the Nega-Binary scheme that resulted in the best saving was the same as the scheme that resulted in maximum estimate of power saving. This can be attributed to the observation (based on our simulations) that the contribution due to inter-data toggle is almost identical across various Nega-Binary schemes. We hence can use the power saving estimate to arrive at the optimum Nega-Binary scheme. There are two advantages of choosing a Nega-Binary scheme this way. One, it does not require actual sample by sample data, only an overall distribution profile will suffice. Two, the run times for computing the best Nega-Binary scheme are orders of magnitude smaller.

D. A Few Observations

• We observed that for a given type of distribution (e.g. Gaussian, bimodal etc.) there was a fixed trend in the best Nega-Binary representation for different precisions (i.e. N values). In fact, from a knowledge of the best Nega-Binary representation for lower values on N, the scheme for higher values could be inductively obtained. Table I shows the best Nega-Binary schemes for 5 to 10 bit precision data having a non-zero mean Gaussian distribution. The trend in the best Nega-Binary scheme across bit-precision is at once apparent.

TABLE I
BEST NEGA-BINARY SCHEMES FOR GAUSSIAN DATA DISTRIBUTION
(MEAN = MAX/2; SD = 0.17 MAX)

N PRECISION	BEST NEGA-BINARY SCHEME (N+1 BIT PRECISION)	SAVING
5	+ - - - + +	25.41%
6	+ - - - + + +	17.87%
7	+ - - - + + + +	13.73%
8	+ - - - + + + + +	11.16%
9	+ - - - + + + + + +	9.42%
10	+ - - - + + + + + + +	8.15%

• As we pointed out before, the Nega-Binary scheme performs well only with peaked distributions. For a symmetrical, uniform distribution the 2's complement scheme is better. This is apparent since the Nega-Binary scheme is implemented with N+1 bits of precision to take care of the entire range. It is only in a few regions that the toggle count is lesser compared to its 2's complement counterpart. A uniform distribution nullifies this effect. Fig. 6 shows a plot of saving versus the Standard Deviation (SD) expressed as a percentage of the entire span (256 in this case), for an N=8, Gaussian distributed data with mean=max/2.

E. An Additional Power Saving in the Nega-Binary Architecture

In sections II.A to II.D we have demonstrated toggle reduction in the shift registers and as a result toggle reduction in the address lines driving the LUT. Power savings by employing a Nega- Binary architecture is not restricted to the shift register only. Our simulations reveal around 20% additional toggle reduction in the LUT outputs. Such a reduction apart from saving power in the adder also results in substantial power savings in the LUT itself. Table III shows the number of repeated consecutive addresses (RCAs) to the LUT for the 2's complement and the Nega-Binary case. It is easy to observe that the number of repeated consecutive addresses in the shift register outputs gives the number of times no toggles occurs

in the LUT outputs (since the same contents are being read). This toggle reduction is, therefore, independent of the filter coefficients.

TABLE II
TOGGLE REDUCTION IN LUT (FOR 10,000 SAMPLES; GAUSSIAN DISTRIBUTED DATA)

TAPS	N	NEGA-BINARY SCHEME	TOGGLE REDUCTION (% OF 2'S COMP.)
8	4	+ - - + +	25.32%
	6	+ - - + - - -	18.90%
	8	+ - - + - - - - -	12.08%
4	4	+ - - + +	26.75%
	6	+ - - + - - -	17.93%
	8	+ - - + - - - - -	13.14%

A few comments need to be made about these numbers.

• 2's complement RCAs were obtained by counting the number of cases (out of a possible of 10000xN times the LUT is addressed) where two consecutive addresses were identical. A similar computation was performed for the best Nega-Binary scheme (the total number of cases in this case is obviously 10000x(N+1)).

• Toggle reduction was computed by finding the difference between the number of times at least one toggle occurred at the LUT output for the two schemes.

• For all the three different precisions a Gaussian distribution with mean = max/2 and an SD = 0.2 max was used.

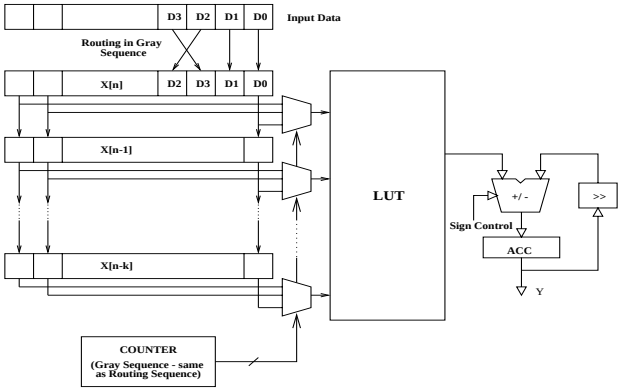

Fig. 7. Shiftless implementation with fixed Gray Sequencing

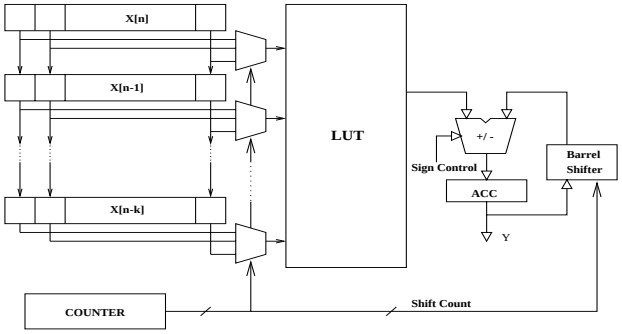

Fig. 8. Shiftless implementation with any Sequencing

III. TOGGLE REDUCTION IN MEMORY BASED IMPLEMENTATIONS BY GRAY SEQUENCING AND SEQUENCE RE-ORDERING

Techniques have been proposed[7,8] to eliminate shifting in registers to reduce power by storing data as memory bits and using a column decoder to access bits at the desired location. In other words, instead of shifting the data, the pointer is moved. While such a technique reduces power dissipation due to data shifting, it results in additional power dissipation in the column decoder. We now propose the following techniques for reducing power in such shift-less DA implementation.

1. Using a Gray sequence in the counter (column decoder) for selecting subsequent bits - this would reduce the toggling in the counter outputs which drive the muxes to the theoretical minimum.

2. Using the flexibility of having several Gray schemes to choose a data distribution dependent scheme which minimizes toggles in the mux outputs.

Gray coded addressing has been shown to result in upto 50% reduction in the address bus power dissipation [6]. Fig. 7 illustrates a DA based FIR with a fixed Gray sequencing scheme. This results in theoretically minimum possible toggles occurring in the counter output. As can be seen such an implementation requires no additional hardware in the basic DA structure.

An N-bit Gray code can be obtained in N! ways (in a Gray code any two columns can be swapped and we still have a Gray code). This freedom can be exploited to obtain a data specific Gray code which minimizes the toggle count as successive bits are selected within the register. This gives us dual power saving : one, in the counter output lines themselves and two, in the multiplexer output which drives the LUT (i.e the LUT address bus). There is an additional overhead of course. Since we are not scanning the register sequentially, a simple shift and accumulate cannot be used. Instead, we require a barrel shifter, as shown in Fig. 8, to shift and accumulate as per the counter sequence. As with the best Nega-Binary scheme, we can choose the Gray code which minimizes the weighted toggle sum using a saving formula very similar to the one used in section II.C. However, no extra bit of precision is required.

TABLE III
TOGGLE REDUCTION FOR DIFFERENT GRAY SEQUENCES

NO.	GRAY SEQUENCE USED	% SAVING
1.	0 1 3 2 6 7 5 4	9.17%
2.	0 2 3 1 5 7 6 4	3.54%
3.	0 1 5 4 6 7 3 2	6.08%
4.	0 2 6 4 5 7 3 1	4.14%
5.	0 4 5 1 3 7 6 2	9.51%
6.	0 4 6 2 3 7 5 1	2.22%

Table III shows toggle reduction for different 3-bit Gray sequences of an N=8 Gaussian distributed data with mean = -max/2 and SD = 0.16 max. As can be seen, the best case toggle reduction is 9.51%. The toggle reduction varies depending on the type of distribution. One important observation is that we get savings for symmetric Gaussian as well as symmetric bimodal data, where the Nega-Binary performance deteriorates substantially.

IV. Results

Tables I, II and III highlight the effectiveness of the proposed techniques in reducing power dissipation in the DA based implementation of FIR filter. We now present some more results on power savings obtained for different number of bits of data precision and different distribution profiles of data values. Table IV shows the percentage reduction in the number of toggles for two different Gaussian distributions. TR1 is the weighted toggle reduction as computed using the saving formula; TR2 is the percentage toggle reduction obtained by using 25000 actual samples (i.e. it accounts for the 'inter-data' toggles) in a 8-tap filter. The predictable trend in the best case Nega-Binary scheme for different precisions is at once apparent.

TABLE IV
Toggle Reduction as a percentage of 2's complement case for two different Gaussian distributions

	Best Nega-Binary Scheme	N	TR1 (%)	TR2 (%)
1	+ - - + +	4	49.75 %	41.96 %
	+ - - + + +	5	35.63 %	28.95 %
	+ - - + + + -	6	28.34 %	24.04 %
	+ - - + + + + - -	8	20.88 %	16.94 %
	+ - - + + + + - - - -	10	16.59 %	12.80 %
	+ - - + + + + - - - - - -	12	13.52 %	7.17 %
2	- + + - -	4	42.41 %	32.51 %
	- + + - - +	5	34.07 %	26.74 %
	- + + - - + +	6	27.71 %	22.06 %
	- + + - - + + + +	8	19.85 %	16.12 %
	- + + - - + + + + + +	10	15.39 %	12.63 %
	- + + - - + + + + + + + +	12	12.55 %	8.68 %

V. Conclusion

In this paper we have presented a technique based on data encoding for low power realization of FIR filters implemented using Distributed Arithmetic. For many applications, the distribution profile of the input data values is known. We have shown how this knowledge can be used to arrive at a Nega-Binary coding scheme which minimizes the number of toggles that occur in the shift registers. We have presented results to show that using this technique, the power dissipation in the shift registers can be reduced by upto 40% for different data distribution profiles, different number of filter taps and different number of bits of data precision. We have further demonstrated upto 25% toggle reduction in the LUT outputs, independent of its contents. The choice of a Nega-Binary scheme is effective for two main reasons. Firstly, with one extra bit of precision, we can get $2^N + 1$ different Nega-Binary schemes that cover the entire range of the 2's complement representation while having different regions of minimum toggle. Hence, for distributions peaking at different regions, there always will be a Nega-Binary scheme that will out-perform the 2's complement coding. Secondly, the conversion from binary to Nega-Binary representation can be achieved with minimal hardware overhead. We have presented a hardware efficient bit-serial implementation of binary to Nega-Binary conversion, which results in a minimal power dissipation overhead. The implementation also achieves the important constraint of programmability. The same design can be used for different Nega-Binary scheme by programming the 'sign' register appropriately.

For shift-free memory based implementations, we have shown that Gray sequencing can be used to reduce power consumption by upto 50% in the column decoder. We have presented a bit-reordering technique which with negligible area overhead, enables sequential access of register bits using graycoded column decoding. We have demonstrated that a further reduction (upto 9.5toggles in the LUT address bus is possible by choosing the best Gray code (out of the N! different combinations available for an N-bit Gray code).

References

[1] A. V. Oppenheim and R. W. Schafer, *Discrete Time Signal Processing*, Prentice Hall, 1989

[2] S. A. White, "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review", IEEE ASSP Magazine, July 1989, pp. 4-19

[3] Texas Instruments, TCS4000ULV 0.35μm CMOS Standard Cell, Macro Library Summary, Application Specific Integrated Circuits 1996

[4] A. P. Chandrakasan and R. W. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits", Pro- ceedings of the IEEE, Vol. 83, No. 4, April 1995

[5] S. Zohar, "A VLSI Implementation of a Correlator/Digital Filter Based on Distributed Arithmetic", IEEE Trans- actions on Acoustics, Speech and Signal Processing, Vol. 37, No. 1, January 1989, pp. 156-160

[6] M. Mehendale, S. D. Sherlekar and G. Venkatesh, "Techniques for Low Power Realization of FIR Filters", ASP- DAC 95

[7] N. Tan, S. Eriksson and L. Wanhammar, "A Power-Saving Technique for Bit-Serial DSP ASICs", ISCAS 94, Vol. IV, pp. 51-54

[8] B. New, "A distributed arithmetic approach to designing scalable DSP chips", EDN, August 17, 1995