LOW-POWER DISTRIBUTED ARITHMETIC ARCHITECTURES USING NON-UNIFORM MEMORY PARTITIONING*

Sumant Ramprasad[a], Naresh R. Shanbhag[b], and Ibrahim N. Hajj[b]
[a]Dept. of Computer Science, [b]Dept. of Electrical and Computer Engineering,
Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign,
Urbana IL USA 61801.
{ramprasa, shanbhag, hajj}@uivlsi.csl.uiuc.edu

ABSTRACT

In this paper, we present a low-power Distributed Arithmetic (DA) architecture. In a DA architecture, a memory is employed to store linear combinations of coefficients. The probability distribution of addresses to the memory is usually not uniform because of temporal correlation in the input. We present a rule governing this probability distribution and use it to partition the memory such that the most frequently accessed locations are stored in the smallest memory. Power dissipation is reduced because accesses to smaller memories dissipate less power. Experimental results with an 8-tap filter with 8 bits of data precision result in a 32% power reduction in the memory. A 28% power reduction was obtained by just detecting accesses to the two most frequently accessed locations (0x00 and 0xFF), which is a strong argument for using the techniques proposed in this paper.

1. INTRODUCTION

A common operation in digital signal processing (DSP) is computing the inner product of two vectors. Distributed arithmetic (DA) [8] has been used to compute the inner product because of the efficiency of DA architectures. In a DA architecture, the multiplier is eliminated by employing a memory to store linear combinations of the coefficients. Figure 1 shows one possible DA-based implementation of a 4-tap FIR filter. The memory addresses are formed by grouping bits in the same bit position from successive input samples. The input is shifted in one bit at a time into the register containing $x(n)$. The output is available once every B clocks (where B is the input precision) from the accumulator. The size of the memory for a k-tap filter is 2^k words. It is possible to reduce the memory size to 2^{k-1} words by employing extra logic [8]. In this paper, we will concentrate on the architecture in Figure 1, though all our techniques are applicable to the architecture with memory of size 2^{k-1} words.

In a DA-based filter, power is dissipated in the shift register, memory, adder, shifter, and the accumulator. Since the memory size increases exponentially with the number of taps and the power dissipation in a memory increases with its size, a considerable amount of the total power dissipation in the filter occurs in the memory. This is particularly true as the number of taps increases. In this paper,

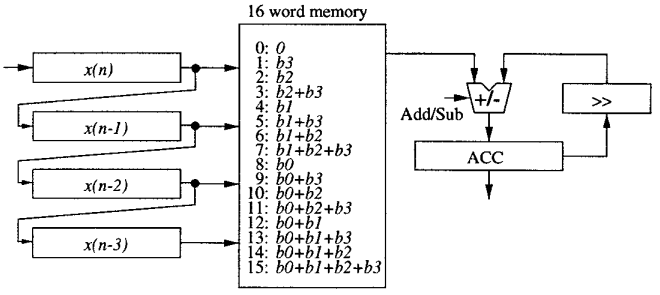

Figure 1. DA-based implementation of a 4-tap FIR filter

we present an architecture to reduce the power dissipation in the memory. The proposed architecture exploits the fact that the input to a filter is typically correlated, due to which the probability distribution of memory addresses is not uniform. We present a rule governing this distribution and use it to partition the memory so that the most frequently accessed locations are stored in a small memory and use a larger memory to store the remaining data. Power dissipation is reduced because accesses to the smaller memory dissipate less power. Experimental results indicate a reduction in power dissipation in the memory of up to 32% for an 8-tap filter with 8 bits of data precision.

The basic idea in our approach is similar to that of caches in general purpose microprocessors, where a small, fast, and lower-power memory is used to store the most frequently accessed data [2]. A similar idea, termed precomputation [1], has also been employed in logic synthesis, where a computation is divided into two parts, with the most frequent computations being done in the first part, which consumes less power. In [3], the non-uniform probability distribution of variable-length codes is employed to partition the table used in variable-length decoding so that the most frequently accessed data are stored in a small memory. To our knowledge, memory partitioning has not been applied to DA architectures, though there are other approaches to reducing power dissipation specifically in such architectures. For instance, in [5], the nega-binary code is employed to reduce the transitions in the shift register, whereas in [7], the shift register is eliminated by moving a pointer to the data instead of moving the data itself.

[1]THIS WORK WAS SUPPORTED BY DARPA CONTRACT DABT63-97-C-0025, NSF CAREER AWARD MIP-9623737, AND NSF AWARD MIP 97-10235.

0-7803-5471-0/99/$10.00©1999 IEEE

The rest of this paper is organized as follows. In section II, we present our low-power DA architecture and in section III we provide experimental results for the reduction in power dissipation.

2. LOW-POWER DA ARCHITECTURE

In this section, we present our low-power DA architecture. Since the architecture exploits the skewed probability distribution of memory addresses, we first describe this distribution followed by the low power architecture.

2.1. Probability distribution of memory addresses

In a DA based filter, the memory addresses are formed by grouping bits in the same bit position from successive input samples (see Figure 1). The transition activity of bits in typical audio, video, and communications channel data is shown in Figure 2, from which, we see that the less significant bits (by less significant bits, we mean bits close to and including the least significant bit) in a typical input sample have a transition activity close to 0.5 [4, 6]. Hence,

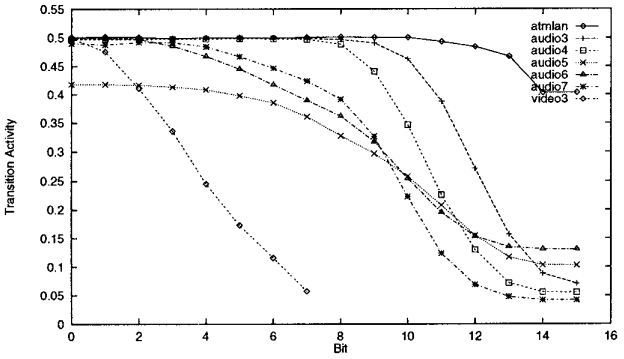

Figure 2. Transition activity versus bit position

addresses formed by grouping bits in the less significant bit positions are uniformly distributed. From Figure 2, we also see that the more significant bits (by more significant bits, we mean the bits close to and including the most significant bit) in a typical input sample have a transition activity significantly less than 0.5. Hence, addresses generated from a more significant bit position have the characteristic that those which have fewer transitions within that address are more probable. For instance, assuming 8-bit addresses, 0x00 (00000000) and 0xFF (11111111) are the most frequent since there are no transitions within those addresses. The addresses 0x55 (01010101) and 0xAA (10101010) are the least frequent since they have the maximum number of seven transitions. Figure 3 shows the probability of occurrence of addresses in an 8 tap filter. The inputs are 8-bit video data and 16-bit audio data. The addresses are arranged in order of increasing number of transitions on the x-axis. We present the following rule which we have found to hold for all input data we have examined and which we will use in our low-power architecture.

Rule 1: The probability of occurrence of an address in a DA based filter with correlated input decreases as the number of transitions within that address increases.

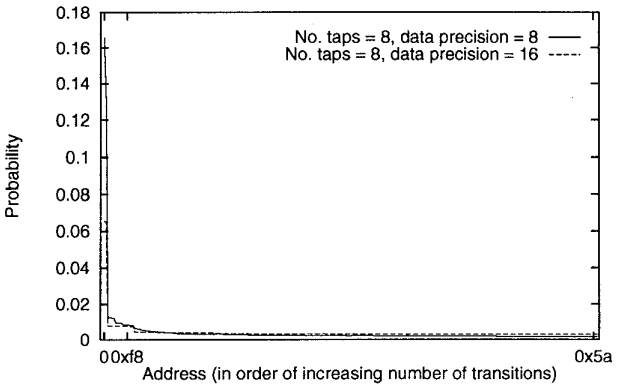

Figure 3. Variation in probability with transitions in an address

2.2. Low Power DA architecture

In this subsection, we present our low-power DA architecture. We place the most frequently accessed locations (which are also addresses with fewer transitions) in a small memory since accessing a small memory dissipates less power. Let M be the original memory containing all possible addresses. We partition M into l memories M_1, M_2, ..., M_l with M_1 containing the most frequently accessed locations, M_2 the next most frequently accessed locations, and so on. Let S_i be the size of memory M_i, E_{Hi} be the energy dissipated when there is a hit in M_i, $E_{overhead}$ be the energy dissipated while decoding which memory to access, and Pr_i to be the probability of a hit in M_i. The energy dissipated during a memory access is given by,

$$Energy = \sum_{i=1}^{l} Pr_i E_{Hi} + E_{overhead} \quad (1)$$

Figure 4 shows the proposed low-power DA architecture with two memories (i.e., $l = 2$). The memory select logic

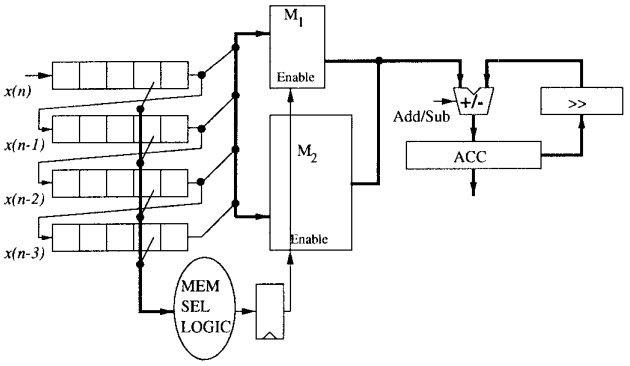

Figure 4. Multiple memory architecture

in Figure 4 decides which memory will be accessed. In Figure 4, this action is performed in the previous clock cycle in order that it not increase the cycle time. To make the memory select logic efficient, we use Rule 1 by placing all addresses with j or fewer transitions in the smaller memory (where j is determined experimentally). The memory select

logic outputs 0 or 1 depending on whether the number of transitions within the address is greater than, less than or equal to j, respectively. The memory select logic for an 8-bit address is shown in Figure 5, where a row of exclusive-or gates convert transitions within an address to 1's, followed by a tree of 1-bit full adders to count the number of 1's, which is then input to a comparator to compare with j. The output of the comparator is used to select the memory.

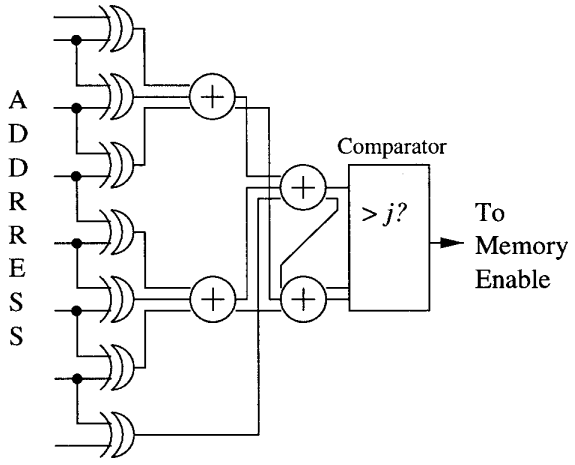

Figure 5. Memory select logic

Figure 6 shows the variation of Pr_1 with S_1 for filters with 4, 8, and 12 taps. The inputs are 8-bit video data and 16-bit audio data. For an 8-tap filter processing 8-bit video data, 10% of the addresses account for over half the accesses. The variation of hit probability Pr_1 with memory size S_1 for the architecture containing memory of size 2^{k-1} words is similar to Figure 6.

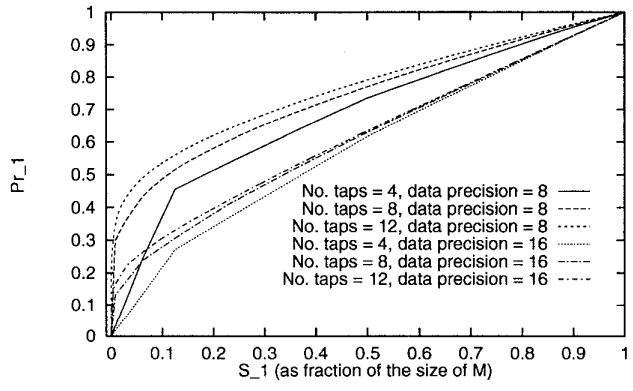

Figure 6. Pr_1 **vs.** S_1

2.2.1. Memory bypass architecture

If S_1 is small (< 8), then it is more efficient to provide the elements of M_1 directly to a multiplexer rather than employing a separate memory. This is shown in Figure 7, where we have logic to detect the accesses to M_1 and provide the data at those addresses directly. From Rule 1, we see that the logic to detect occurrence in M_1 basically detects the addresses with the fewest transitions (for instance, 00000000, 11111111, 11111100, and so on). It is possible to combine the memory bypass architecture with the multiple memory architecture. In the specific example of Figure 7, M_1 consists of the locations 0000 and 1111.

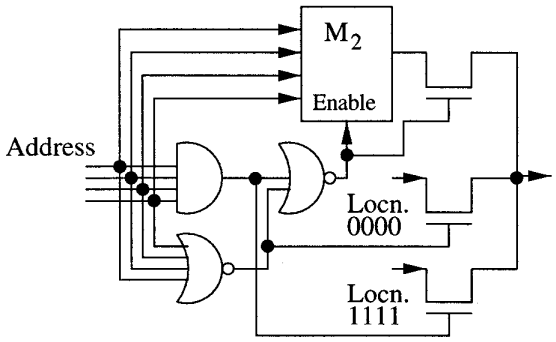

Figure 7. Memory bypass architecture to reduce memory accesses

3. EXPERIMENTAL RESULTS

In this section, we present experimental results on the reduction in power dissipation in the memory. We do not include the rest of the filter (i.e., shift registers, adder, shifter, and accumulator) since our techniques reduce the power dissipation in the memory and that in the rest of the filter is unchanged.

We assumed the memory is implemented as a ROM and estimated its energy dissipation per access as,

$$Energy = C_T V_{dd}^2, \quad (2)$$

where V_{dd} is the supply voltage and C_T is the average capacitance switched during a single access. We used the following formula from [4] to estimate C_T,

$$C_T = C_0 + C_1 N_I 2^{N_I} + C_2 P_O N_O 2^{N_I} + C_3 P_O N_O + C_4 N_O,$$

where,

- C_0, C_1, C_2, C_3, and C_4 are empirically determined capacitive coefficients that depend on the exact circuitry and technology used by the ROM (the values of the coefficients used in this paper are shown in Table 1),
- N_I is the number of address bits (From Figure 1, this is equal to the number of taps in the filter),
- P_O is the probability of a 1 in the data stored in the ROM (assumed to be 0.5 in our calculations), and
- N_O is the number of bits in the output word. Since the memory stores linear combinations of coefficients, N_O, for our purposes, is the sum of the coefficient bit-width (assumed to be 8) and $\log_2 N_I$.

We used the memory bypass architecture if S_1 was less than 9 and the multiple memory architecture for larger values of S_1. An 8-tap filter was used in the experiment, due to

Table 1. Capacitive coefficients for a ROM

Coefficient	Value (pF)
C_0	3.693000
C_1	0.002103
C_2	0.003092
C_3	0.302080
C_4	0.343750

which the size of the memory in the original filter was 256 words. The basic unit in the additional logic to detect membership in M_1 in Figure 7 is an 8-input NOR gate, which was estimated, using SPICE, to have an average switched capacitance of 0.15pF. The average switched capacitance in the memory select logic in Figure 4 was estimated, also using SPICE, to be 0.64pF for an 8-bit address. In Figure 8, we plot the reduction in power dissipation over an unpartitioned memory. In order to determine how much of the reduction is due to partitioning and how much due to the skewed address distribution, we also plot the reduction in power dissipation over a partitioned memory assuming uniform distribution of addresses. From Figure 8, we see that for 8-bit video data, the maximum reduction of 32% is obtained by the memory bypass architecture when S_1 is 8 and most of this reduction is due to the skewed address distribution. From Figure 8 we also see that, for 8-bit data, we can obtain a 28% reduction in power in the memory just by detecting the 2 most common addresses, which are 0x00 and 0xFF. This is a powerful argument for using the techniques presented in this paper. From Figure 8, we see that for 16 bit audio data, the maximum reduction of 18% is obtained by a multiple memory architecture when the sizes of M_1 and M_2 are both equal to 128 and most of this reduction is due to the partitioning of memory.

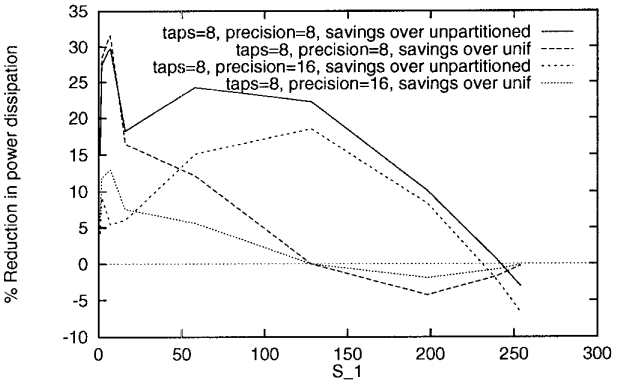

Figure 8. Power savings vs. S_1

4. CONCLUSION

In this paper, we have presented a low-power filter using distributed arithmetic, in which, we exploit the skewed distribution of addresses to the memory by partitioning it so that the most frequently accessed locations are stored in a small memory. Power dissipation is reduced because accesses to a small memory dissipate less power. Simulation results with 8-bit input data and an 8-tap filter indicate a reduction in power dissipation in the memory of up to 32%.

REFERENCES

[1] M. Alidina, J. Monterio, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Precomputation-based sequential logic optimization for low-power," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 426–436, December 1994.

[2] N. Bellas, I. N. Hajj, C. D. Polychronopoulos, and G. Stamoulis, "Architectural and Compiler Support for Energy Reduction in the Memory Hierarchy of High-Performance Microprocessors," *International Symposium on Low Power Electronics and Design*, pp. 70–75, Monterey CA, August 10–12 1998.

[3] S. Cho, T. Xanthopoulos, and A. P. Chandrakasan, "An Ultra Low Power Variable Length Decoder for MPEG-2 Exploiting Codeword Distribution," *IEEE Custom Integrated Circuits Conference*, pp. 177–180, Santa Clara CA, May 11–14 1998.

[4] P. E. Landman and J. M. Rabaey, "Activity-sensitive architectural power analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 6, pp. 571–587, June 1996.

[5] M. Mehendale, A. Sinha, and S. D. Sherlekar, "Low Power Realization of FIR Filters Implemented Using Distributed Arithmetic," Asia and South Pacific Design Automation Conference, pp. 151–156, Yokohama Japan, February 10–13 1998.

[6] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, "Analytical Estimation of Signal Transition Activity from Word-Level Statistics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 7, pp. 718–733, July 1997.

[7] N. Tan, S. Eriksson, and L. Wanhammar, "A Power-Saving Technique for Bit-Serial DSP ASICs," *International Symposium on Circuits and Systems*, vol. 4, pp. 51–54, London England, May 30 – June 2 1994.

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