IEEE 2006 Custom Intergrated Circuits Conference (CICC)

An Energy Scalable Computational Array for Sensor Signal Processing

Liping Guo, Mackenzie Scott, Rajeevan Amirtharajah
Dept. of Electrical and Computer Engineering, University of California, Davis CA 95616

Abstract— **Harvesting energy from environmental sources can extend wireless sensor network node lifetime beyond the limits of battery technology. However, the output power from an energy harvester is highly variable. We propose a domain-specific computational array which maximizes sensor performance by matching system power consumption to the available scavenged energy through power scalable approximate signal processing. The array consists of distributed arithmetic (DA) based functional units coupled with a reconfigurable interconnect structure. Each unit implements several core linear and nonlinear signal processing functions in an area efficient manner which also minimizes leakage power. Several sensor DSP applications (FIR, IIR, and FFT) have been mapped onto the array. Post-layout simulations confirm that the proposed domain-specific computational array is energy efficient and energy scalable.**

I. Introduction

Wireless sensor networks have tremendous potential for enabling new environmental monitoring, public health, and security applications. However, the operating lifetime of a sensor node is often constrained by its battery. For large-scale and dense deployment of sensor networks, long operating lifetimes are necessary to reduce battery replacement costs. To prolong sensor node lifetime, two complementary solutions are available: improving energy efficiency through power-aware operation and scavenging energy from environmental sources. Both solutions require the underlying hardware to provide energy scalable computation.

The principal idea behind energy scalable computation is to maximize computational quality for a given energy constraint [1]. Implementing scalability requires a set of hardware configurations, each providing a different energy-quality tradeoff. Power-aware operation is achieved by selecting appropriate configurations based on either user requirements or the available power. This tunable energy-quality tradeoff is also essential for efficient utilization of scavenged energy because the available energy is typically small and highly variable. By altering the throughput and computation accuracy through configuration, the user can trade power consumption for output quality as the available energy varies [2].

For the proposed energy scalable computational array, we examine serial techniques to provide energy scalability with low overhead. In older VLSI technologies, bit serial techniques were used to reduce the area of arithmetic structures such as multipliers [3] by using flip-flops to decrease the combinational logic required for the computation. For fixed throughput, an N-bit serial implementation must be clocked at N times the desired frequency resulting in increased dynamic power. However, the reduced transistor count in the serial implementation decreases leakage, which contributes an increasing percentage of total power as CMOS technologies scale [4]. A comparison of post-layout simulated power between serial and parallel adders and multipliers demonstrates that, at low throughput, static power dominates and serial structures consume less total power than parallel structures [5].

In Section II, we explore energy scalable approximate signal processing using distributed arithmetic (DA). We describe an enhanced DA unit which supports a set of linear/nonlinear DSP functions in Section III. We evaluate the reconfigurable interconnect structure and a simple inter-DA communication scheme in Section IV. In Section V, we discuss DA-array application mapping and results. Our conclusions and future work are presented in Section VI.

II. Energy Scalable DSP Using DA

Sensor DSP applications are well suited for serial implementation because they typically require low throughputs. Distributed arithmetic computes inner products using look-up tables (LUTs) instead of multipliers [6], and is an elegant technique for implementing low power and energy scalable matched filters for sensor applications [7]. DA is inherently bit-serial, but operates on multiple bit streams simultaneously; thus it retains the power advantages of serial computation while reducing the needed frequency for a given throughput.

Consider the calculation of the inner product $y = \sum_{k=0}^{M-1} a_k x_k$ where $\mathbf{a}$ is an M-dimensional constant vector and $\mathbf{x}$ is an M-dimensional input vector. Using N-bit two's complement representation, the equation can be rearranged as:

$$y = -\sum_{k=0}^{M-1} a_k b_{k(N-1)} 2^{N-1} + \sum_{n=0}^{N-2} \left[\sum_{k=0}^{M-1} a_k b_{kn} \right] 2^n \quad (1)$$

Since each b_{kn} equals 0 or 1 only, the bracketed term in equation 1 has 2^M possible values, which maybe precomputed and stored in a 2^M-word memory. The bit serial input data addresses the memory, whose contents are accumulated to obtain the outer sum of Equation 1. Minor additional circuitry is necessary to handle the sign bit. For an N-bit x_k vector, the final result y is produced after N cycles.

A DA unit consists of a shift memory for input vector $\mathbf{x}$, an LUT for storing linear combinations of constant vector $\mathbf{a}$, and a shift and accumulate circuit (Fig. 1). The shift memory can be implemented with flip-flops and bypass muxes to vary the input data bitwidth and achieve energy scalability. Configuration bits can gate the flip-flop clocks to further reduce power. Layouts indicate that the 4 input DA unit has 15% less area than using four serial multiply-accumulators.

A large FIR filter cannot be implemented using a single table since the memory would grow exponentially. Instead, an

1-4244-0075-9/06/$20.00 ©2006 IEEE

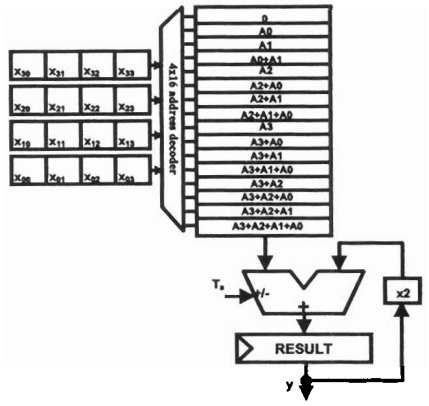


Fig. 1. A simple DA unit

Operation	Exe. Cycles	Algorithm
Vector dot product	16	distributed arithmetic
Serial multiply	16	shift + add
Division	2-55	subtract + shift
Sqrt	33	nonrestoring algorithm: add + subtract + shift
Log	18-33	lookup table + linear approx.
Add/Sub	1	parallel addition
Polynomial	64 + N*81	serial Horner's Rule expansion N = polynomial degree
Complex multiply (cmul)	64+8	shift + add + subtract + swap
Complex add/sub (cadd, csub)	4	add + subtract
Note: Exe. cycles are based on full-length inputs and I/O cycles are not included		

Fig. 2. Enhanced DA supported functions

array of DA units are used whose outputs are accumulated into the final filter result. This enables another power-performance tradeoff: varying the number of active DA units varies the filter length along with filter performance [8]. Decreasing the number of taps increases the passband ripple, reduces stop band attenuation, and increases transition region width [9]. Another important parameter is the bit width of the filter coefficient elements a_k. Recomputing the coefficients for a smaller bit width reduces LUT area in the DA architecture. This does not affect the transition region for longer coefficient widths, however the stop band attenuation will be worse as the width decreases. One can also truncate coefficient width by zeroing the lower order bits, but the coarser quantization introduces high frequency noise in the stop band. These effects illustrate the notion of approximate processing: static recomputation of filter taps and coefficient widths, dynamic truncation of these values, and scaling of input data bit width all trade precision for potential power reductions [9].

III. An Enhanced DA Unit

To enable a wider range of DSP applications, the basic DA structure (Fig. 1) is augmented to support a set of linear/nonlinear functions (Fig. 2) besides the dot product. These functions are implemented serially to achieve energy scalability with minimal hardware overhead. Four major modifications are added to the basic DA structure to support the new operations (see Fig. 3): (1) two extra data ports provide flexible data flow between DA units; (2) multiplexers are inserted for adder operand and accumulator load data selection; (3)

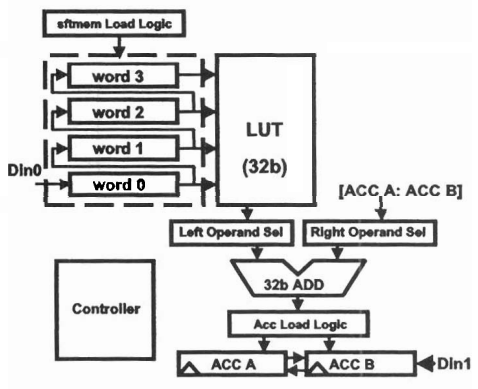


Fig. 3. Enhanced DA microarchitecture

a parallel load feature is added to the input shift register; and (4) the 32-bit accumulator is split to allow separate 16-bit manipulation within one execution cycle and power saving through clock gating when fewer bits are needed.

At configuration time, each DA unit is initialized via a 12-bit word which determines its function, data I/O method, throughput, and desired level of energy scalability. The execution of each function involves four phases: an idle phase, two data I/O phases, and a data processing phase. All phases except data processing are the same for all functions. During idle, the DA unit is in sleep mode and the input shift registers, counters, and accumulator are all clock-gated to reduce power consumption. Two I/O phases are responsible for correct dataflow between DAs. Because the core functions are data-dependent, the data processing of each DA usually requires a variable number of cycles to complete. This poses a challenge for data synchronization and demands an I/O method general enough to support different signal flowgraphs.

To significantly reduce DA power, the flip-flop-based input shift register can be replaced by a multiported register file. A single memory element is shown in Fig. 4, where a 6T SRAM cell can be written through two differential ports, XDI and YDI, and read through three single-ended ports XDO, XDOE, and YDO. The two x-direction read ports, XDO and XDOE, can form a single 32-bit operand or two 32-bit operands after sign extension. The bitlines for the X and Y ports route orthogonally to allow parallel loading along the X direction and shifting multiple parallel bit streams in the Y direction. The y-axis read port, YDO, addresses the LUT. Occupying about the same area as the flip-flop shift register, the 4x16 mutiport memory array is 3-10 times lower power for all interesting input bit widths as shown in Fig. 5. Besides decreasing power, the parallel load capability of the SRAM-based approach dramatically reduces the hardware overhead in function implementation. Since the cells in each row share the same input data line and bit shifting is directed by the *read* and *write* control lines, this structure provides bit-level energy scalability without extra mutiplexers.

IV. DA Array Communication

The island-style architecture is the most widely used interconnect model in today's commerically avaialble FPGAs. Our

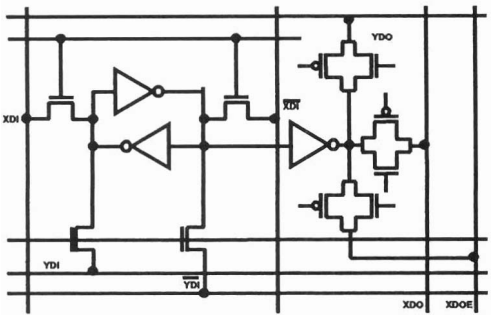


Fig. 4. SRAM-based shift memory cell

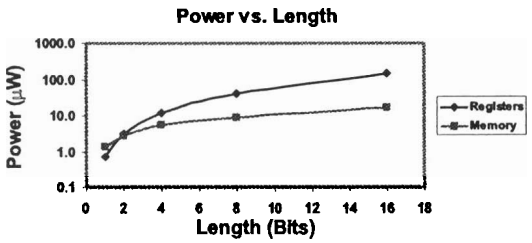
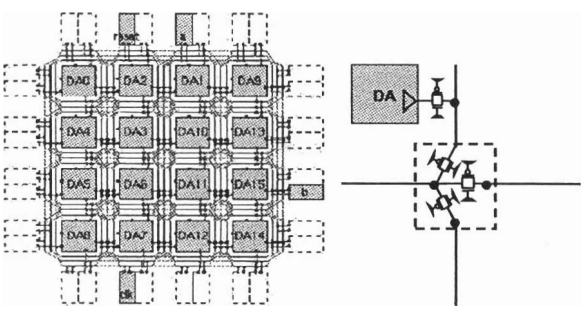


Fig. 5. SRAM sftmem vs. flip-flop reg. power comparision

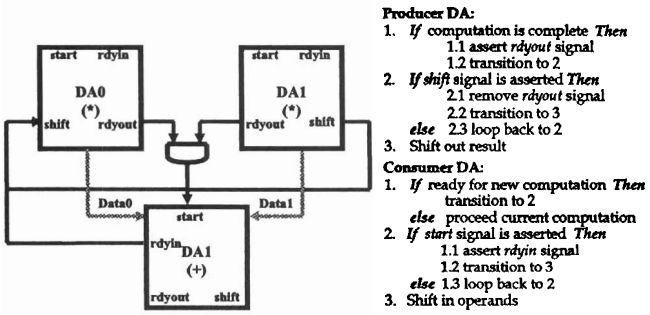
computational array is formed by embedding the enhanced DA units into an island-style interconnect (Fig. 6a). For general purpose FPGAs, interconnect consumes most of the energy while logic is responsible for only 5% of the total energy consumption [10]. However, this power breakdown does not hold in our case. Serial computation and a simple communication scheme result in few interconnect channels while the target applications' low thoughput requirements decrease transisions on the wires. Post-layout simulation of an FIR filter confirms that most of the power and area of the array are consumed by the functional unit, and not interconnect.

We explored several interconnect parameters including connection box flexibility, switch box topology, and segmentation schemes, using VPR [11]. Based on the evaluation, the *wilton* switch box topology and a segmentation scheme with 20% of tracks spanning one block and 80% of tracks spanning three blocks was adopted. We also evaluated the power-delay-area (PDA) performance of three types of routing switches: an NMOS pass gate, a tri-state buffer, and a full transmission gate, for a set of wire segments of various lengths. The results



(a) 4x4 array (b) switch box

Fig. 6. Reconfigurable DA array structure



(a) (b)

Fig. 7. Inter-DA communication scheme

show that pass gate switches give the best PDA performance and that transmission gate switches outperform the tri-state buffer switches for all types of tested segments. Although pass gate switches offer small area and fast speed, they can cause leakage in downstream buffers and logic gates due to their output voltage failing to rise to full swing. Other techniques such as gate boosting and level-restoring circuits are needed to eliminate leakage [12]. In comparison, transmission gate switches do not have such leakage problems. Traditionally, the transmission gate is not a popular choice for FPGA routing switches because of the area-delay penalty resulting from its large device area and capacitance [12]. In our case, however, implementation complexity and power are of more concern than the area-delay penalty since the computation is running at low speed and requires much less routing area than general-purpose FPGAs. Fig. 6b shows the *wilton* switch box with transmission gate switches. As mentioned in Section III, the variable number of execution cycles of the core functions impose challenges on data synchronization and suggests asynchronous inter-DA communication. We found that a simple handshaking communication scheme is a viable solution for most DSP dataflows. In Fig. 7(a), we use three DAs to illustrate the communication scheme. The top DAs are data producers and the bottom DA is a consumer. The handshaking process follows the pseudo-code described in Fig. 7(b).

V. Application Mapping and Results

DSP flowgraphs are realized by assigning an appropriate function for each DA unit and configuring the interconnect. For general-purpose FPGAs, mapping, placement, and routing are often done separately due to high complexity. In our case, the limited application domain and the regularity of the DSP flowgraphs permit simultaneous mapping, placement, and routing, which leads to solutions with the best overall performance. For example, for a mapped array, the workload of each DA is determined by its operation and the DSP flowgraph derived from the application. Knowing mapping implications at placement time, we can distribute the heavily-loaded DA units sparsely on the array to avoid localized hotspots induced by an unbalanced workload.

A flow graph of the decimate-in-frequency (DIF) FFT butterfly computation is shown in Fig. 8(a). The computation

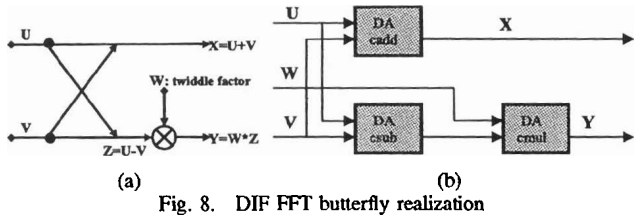


Fig. 8. DIF FFT butterfly realization

involves three functions, a complex addition, a complex subtraction, and a complex multiplication, which are all supported by our enhanced DA unit. A single butterfly can be realized by three DA units as shown in Fig. 8(b). A N-point radix-2 FFT requires $\log_2 N$ stages to compute with each stage consisting of $N/2$ butterflies. A 4-point FFT can be directly mapped onto a 4x4 array. In 0.25μm CMOS, a 4x4 array built using standard cells is about 3 mm×3 mm. Higher-point FFTs can be realized by interatively computing a subset of butterfies on the array. The partial results are stored in a small SRAM of an on-chip microcontroller, which is responsible for configuring the DA LUTs and controlling array computations. In the FFT, this involves reordering partial results to prepare inputs for the next iteration and accumulating the final results. The iterative mapping approach places no constraints on the FFT size and offers a tradeoff between array size and number of iterations. Energy scalability can be controlled at two levels: at bit-level by varying bit width of the twiddle factors and the inputs to generate results with different precision, or at iteration-level by adjusting the number of points (N) of the transform to produce a freqency spectrum with different resolutions.

A 32-tap FIR matched filter for biomedical event detection is also mapped on the 4x4 DA array. With each DA unit capable of computing a 4-tap filter, the 32-tap filter is divided into 8 4-tap filters and realized by configuring 8 DAs for the vector dot product and 7 DAs for addition. The post-layout simulation results in Fig. 9 show that power consumption is reduced as input bit width shrinks while the likelihood of correctly recognizing an event also decreases due to the increased quantization noise.

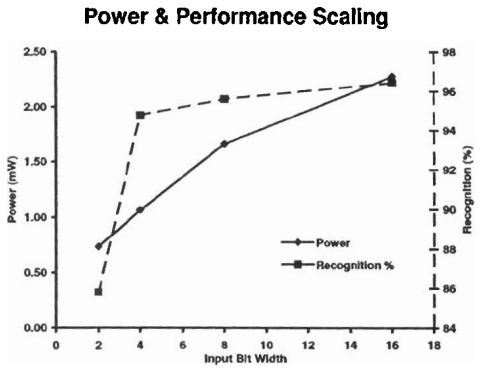


Fig. 9. Power scalable FIR matched filter results.

VI. Conclusion and Future Work

We have shown how DA, as a bit serial technique, provides an elegant solution for low leakage, low power, and energy scalable DSP. To accommodate a wider range of sensor DSP applications, we propose a domain-specific reconfigurable array using enhanced DAs as functional units and an island-style interconnect structure. The enhanced DA unit supports several linear/nonlinear DSP functions in a power efficient and power scalable manner. To significantly reduce power consumption, we designed an SRAM-based shift memory for data input which result in 3-10 times less power consumption than its flip-flop-based counterpart with the same area. We evaluated an inter-DA communication scheme and explored the interconnect design at both architecture and circuit levels. Several sensor DSP applications have been mapped onto the proposed reconfigurable array. The simulated results for an FIR matched filter demonstrate that, through power scalable design, a tunable power/quality tradeoff can be achieved.

Future work includes implementing and refining the enhanced DA unit microarchitecture and circuits, evaluating more sensor DSP applications, automating the mapping process, and examining other data communication schemes. An implementation of the computational unit partially using energy recovery circuits is currently being developed and a custom integrated circuit will ultimately be implemented to validate our architecture, communication, and circuit concepts.

Acknowledgments

This work is supported by the MARCO Interconnect Focus Center under subcontract no. B-12-M06-S12, the Xilinx University Program, and Xilinx Research Labs.

References

[1] A. Sinha, A. Wang, and A. P. Chandrakasan, "Energy scalable system design," *IEEE Transactions on VLSI Systems*, vol. 10, no. 2, pp. 135–45, April 2002.

[2] R. Amirtharajah, Jamie Collier, Jeff Siebert, and Bicky Zhou, "Dsps for energy harvesting sensors," *IEEE Pervasive Computing*, pp. 72–29, July-September 2005.

[3] N. H. E. West and D. Harris, *CMOS VLSI Design - A Circuits and Systems Persective, 3rd ed. Reading*, Massachusetts: Addison-Wesley Publishing Company, 2004.

[4] K. Roy, S. Mukhopadhyay, and H. Mahmood-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits," *Proc. of the IEEE*, vol. 91, no. 2, pp. 305–27, Feburary 2003.

[5] Bicky Bici Zhou, "Memory design for energy scalable reconfigurable logic," October 2004, MS Thesis, ECE Dept., UC Davis.

[6] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE ASSP Magazine*, pp. 4–19, July 1989.

[7] R. Amirtharajah and A. P. Chandrakasan, "A micropower programmable dsp using approximate signal processing based on distributed arithmetic," *IEEE J. Solid State Circuits*, vol. 39, no. 2, pp. 337–347, 2004.

[8] J. T. Ludwig, S. H. nawab, and A. P. Chandrakasan, "Low-power digital filtering using approximate processing," *IEEE J. Solid State Circuits*, vol. 31, no. 3, pp. 395–9, March 1996.

[9] Zulfiqar Ali Ansari, "Energy scalable distributed arithmetic on a field programmable gate array and a standard-cell core," June 2005, MS Thesis, ECE Dept., UC Davis.

[10] V. George, H. Zhang, and J. Rabaey, "The design of a low energy fpga," in *International Symposium on Low-Power Design (ISLPED)*, 1999, vol. ACM Press, pp. 188–93.

[11] V. Betz and J. Rose, "Vpr: A new packing, placement and routing tool for fpga research," in *Proceedings of FPL'97*, 1997, pp. 213–22.

[12] G. Lemieux and D. Lewis, *Design of Interconnection Networks for Programmable Logic*, Kluwer Academic Publisher, 2004.