# A 190GFLOPS/W DSP for Energy-Efficient Sparse-BLAS in Embedded IoT

Richard Dorrance, Dejan Marković
University of California, Los Angeles, CA

## Abstract

A DSP for sparse-BLAS is realized in 40nm CMOS. Featuring an efficient data stream reordering scheme and an intelligent, CSC-aware memory controller, the DSP achieves a peak energy efficiency of 190 GFLOPS/W at 0.6V, 160MHz, and a peak performance of 4.12 GFLOPS at 1V, 515MHz showing more than 6,600×, 2,700×, 1,100×, and 450× higher energy efficiency than state-of-the-art CPU, GPU, DSP, and FPGA hardware designs, respectively.

## Introduction

With the explosive growth of the Internet of Things (IoT), mobile and embedded systems are being asked to support more and more computationally intensive applications such as augmented reality, neural networks, 3D gaming, portable medical imaging, and mobile health monitoring—all of which rely on the manipulation of very sparse data sets [1,2]. The energy efficiency of the sparse basic linear algebra subroutines (sparse-BLAS) has always significantly lagged behind that of their dense counterparts by 2 to 3 orders of magnitude. In existing architectures, sparse matrix-vector and matrix-matrix multiplications (SpMxV and SpMxM) are almost exclusively calculated as series of dot products using the compressed sparse row (CSR) sparse matrix data format. However, due to low computational complexity and the irregular memory accesses of CSR, frequent cache misses and data hazards force the processor to stall for dozens to hundreds of cycles [2] (Fig. 1), further exacerbating the 100× energy gap between computation and memory accesses [3]. In order to improve the energy-efficiency of sparse-BLAS for embedded IoT applications, this paper presents a dedicated sparse linear algebra DSP, based on the compressed sparse column (CSC) data format, in a 40nm CMOS technology.

## System Architecture

We designed a scalable sparse-BLAS architecture (Fig. 2) with CSC to roughly halve the number of required memory accesses and eliminate the data hazards present in CSR. It consists of a sparse-BLAS controller (with an integrated memory controller), 4 processing elements (PE), and a 512Kb cache memory. Each PE uses CSC to calculate SpMxV ($y = Ax$) as a series of column-wise vector additions of $A$ weighted by each element of $x$. Due to the format of the memory references in CSC, the resulting column-major operations allow each element of $x$ to be fetched sequentially and reused (as opposed to multiple random accesses and no reuse for CSR). For example, a sparse matrix with ~10 nonzero elements per column results in a 40% reduction in the memory bandwidth using CSC versus CSR due to reuse. Additionally, every CSR partial product results in a data hazard that stalls the processor for 3-8 clock cycles (fused multiply-add latency) due to the recursive data dependencies of the row-based approach. In principle, CSC avoids these data hazards by accumulating partial products from different rows each clock cycle, but it does not eliminate them. CSC sparse matrices on average see one data hazard for every 15-50 nonzero elements [1,2].

To eliminate the remaining data hazards, a stall-free "Shuffler" has been designed to manage the flow of data in each PE. Fig. 3 shows a block diagram and layout of the PE. In addition to the "Shuffler," each PE occupies 0.055mm$^2$, contains a single FPU (32b FP adder and multiplier), and a 16Kb dual-port SRAM (DP-SRAM) to compute and update partial products in the same clock cycle. When enabled, the "Shuffler" first fills a FIFO-like buffer (depth of 4) for each element of the data matrix ($A_{ij}$), vector ($x_j$), row address ($i$), and *Valid* signal used to calculate a partial product. The "Shuffler" then monitors the last 4 addresses of $i$ issued to the FPU (the latency of our 32b FP adder) for potential data hazards in the buffer. When the first item in the buffer causes a data hazard, the shuffler substitutes the first available, hazard-free, partial product. If no hazard-free, partial product exists in the buffer, the "Shuffle FSM" stalls the PE until the data hazard is resolved. However, a buffer depth equal to the adder latency guarantees zero data hazards. Using this strategy, data can be continuously streamed into each PE with a small startup overhead equal to the combined latency of the adder, multiplier, and the "Shuffler" buffer. When the "Shuffler" is disabled, the PE must stall for 4 clock cycles to resolve a data hazard.

To provide a scalable, high-speed data interface between the 2GB of DDR2 memory on our FPGA system and each of the 4 PEs (chosen to optimize energy per bandwidth, Fig. 2), an on-chip, 512Kb, DP-SRAM is used as a memory cache. The resulting memory hierarchy is top heavy compared to CPUs, GPUs, and DSPs (Fig. 1). Since each PE can only store a finite number of elements, the PE contains a partial working copy of the vector being computed (the 16Kb DP-SRAM in each PE is chosen to optimize energy per partial product computation, Fig. 2). Blocking is performed along the rows of $A$ by the on-chip, sparse-BLAS controller (i.e. each PE is assigned up to 512 rows of $A$ for computation). The final vector is then assembled by concatenating the output of each PE during memory write-back (requiring no additional latency).

## Measurement and Comparisons

SpMxV was performed on 10 unstructured matrices [1,2] using our DSP and the results are compared to existing CPU, GPU, and DSP architectures [2,4-6]. Detailed in Fig. 1, FP resource utilization hovers around 1-2% for CPUs, 0.2-0.5% for GPUs, and 18-20% for DSPs. Our DSP achieves a maximum utilization of 99.92%, with an average of 95.29% (Fig. 4). To measure its effectiveness, the "Shuffler" was disabled and the tests were repeated. The average utilization dropped to 75.82% due to the PEs having to stall for 4 clock cycles once every ~20 nonzero elements due to data hazards.

The power consumption and operating frequency of the sparse-BLAS DSP were measured vs. supply voltage (Fig. 4). The minimum energy point (MEP) is found to be at 0.6V, which corresponds to 6.73mW at 160MHz. At this point, the sparse-BLAS DSP achieves a peak throughput of 1.28 GFLOPS for an energy-efficiency of 190 GFLOPS/W. The maximum operating frequency of the sparse-BLAS kernel is 515MHz at $V_{DD}$=1V, achieving 3.2× higher throughput than the MEP at the cost of 2.9× lower energy efficiency.

Our sparse-BLAS kernel is compared to several CPUs and GPUs, as well as prior FPGA and DSP chip implementations for SpMxV (Fig. 5) [2, 4-6]. Overall, it achieves a 2× higher throughput, while averaging 6,600× better energy efficiency (~100× from reduction in processing time and memory access, 3× from voltage and 20× from frequency scaling), than a CPU running architecture-specific software. Similarly, it achieves 2× higher throughput than the prior DSP chips, with a 1,100× higher energy efficiency. Both the GPU and FPGA SpMxV implementations average ~4× higher throughput, while using ~20× more memory bandwidth than our sparse-BLAS kernel. However, our DSP has 2,700× and 450× higher energy efficiency than the GPU and FPGA designs, respectively. Consuming less than 10mW, the 0.927mm$^2$ sparse-BLAS kernel in 40nm CMOS (Fig. 6) can provide an energy savings of 2 to 3 orders of magnitude, enabling a variety of IoT applications.

## References

[1] N. Bell, et al., SC’09, pp. 1-11, Nov. 2009.
[2] R. Dorrance, et al., FPGA’14, pp. 161-170, Feb. 2014.
[3] M. Horowitz, ISSCC’14, pp. 10-14, Feb. 2014.
[4] J. Fowers, et al., FCCM’14, pp. 36-43, May 2014.
[5] Y. Gao, et al., ASAP’13, pp. 168-174, Jun. 2013.
[6] Y. Gao, et al., HPEC’14, pp. 1-6, Sep. 2014.

| Memory Hierarchy | | CPU | GPU | DSP | This Work |
|---|---|---|---|---|---|
| Register | Structure [bits] | 8 to 64 | 64 to 384 | 32 to 64 | 512 |
| | Size [words] | 8 to 32 | 32 | 32 to 64 | 32 |
| | Latency [cycles] | 1 | 0 | 1 | 1 |
| Cache | Structure | L1/L2/L3 | L1/L2 | L1/L2 | L1 |
| | Size | 32KB/256KB/8MB | 64KB/256KB | 32KB/512KB | 64KB |
| | Latency [cycles] | 6/12/35 | 8/120 | 9/28 | 7 |
| External | Structure | DDR SDRAM | GDDR SDRAM | DDR SDRAM | DRAM |
| | Size [GB] | 4 to 64 | 1 to 12 | 1 to 4 | 2 |
| | Latency [cycles] | 50 to 200 | 200 to 800 | 700 to 1,100 | 80 to 160 |

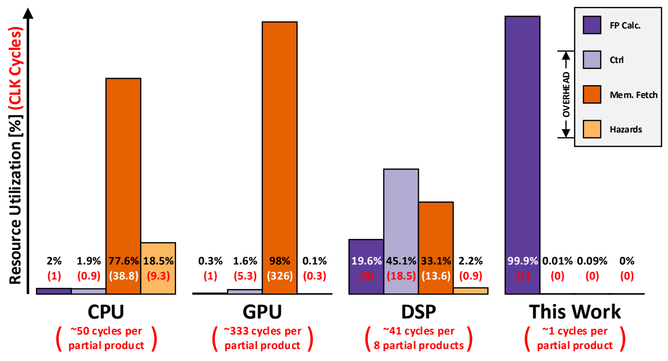


Fig. 1 Resource utilization and memory hierarchy for performing sparse-BLAS.

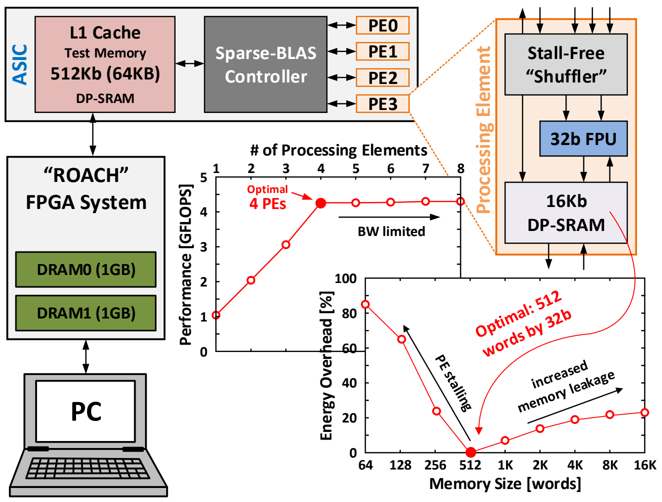


Fig. 2 System architecture and optimization.

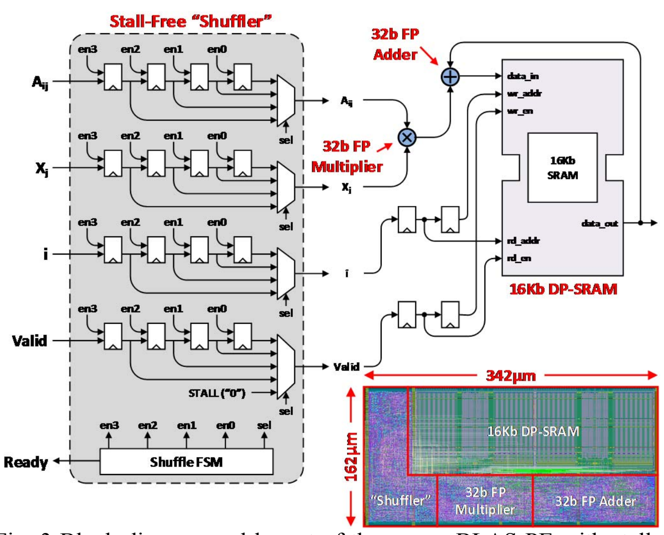


Fig. 3 Block diagram and layout of the sparse-BLAS PE with stall-free, data stream reordering.

| N×N Matrix | Dense | Protein | FEM/Spheres | FEM/Cantilever | Wind Tunnel | FEM/Harbor | QCD | FEM/Ship | Economics | FEM/Accelerator |
|---|---|---|---|---|---|---|---|---|---|---|
| N | 2,000 | 36,417 | 83,334 | 62,451 | 217,918 | 46,835 | 49,152 | 140,874 | 206,500 | 121,192 |
| Nonzeros | 4,000,000 | 4,344,765 | 6,010,480 | 4,007,383 | 11,524,432 | 2,374,001 | 1,916,928 | 3,568,176 | 1,273,387 | 2,624,331 |
| Density [%] | 100 | 0.32761 | 0.08655 | 0.10275 | 0.02427 | 0.10823 | 0.07935 | 0.01798 | 0.00299 | 0.01787 |
| Perf. @ 1V [GFLOPS] | 4.12 | 4.07 | 4.03 | 4.03 | 4.01 | 4.01 | 3.95 | 3.87 | 3.39 | 3.77 |

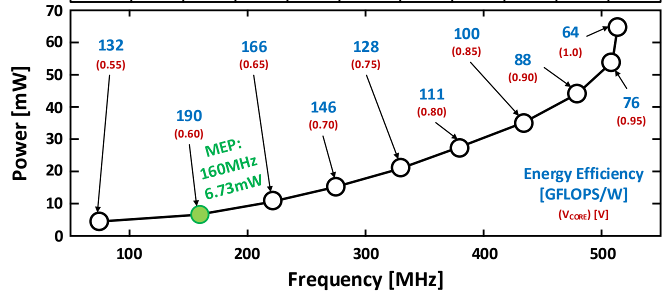
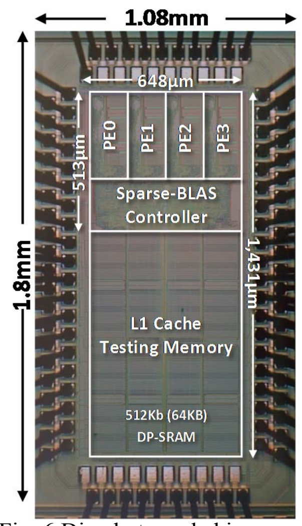


Fig. 4 Measurement results.

| Design | CPU [5] i5-650 | CPU [2] i7-2600 | GPU [4] GTX 580 | GPU [2] GTX Titan | DSP [5] C6678 | DSP [6] 66AK2H12 | FPGA [4] Stratix V D5 | FPGA [2] Virtex-5 SX95T | This Work |
|---|---|---|---|---|---|---|---|---|---|
| Tech. [nm] | 32 | 32 | 40 | 28 | 45 | 28 | 45 | 65 | 40 |
| # of Cores | 2/4* | 4/8* | 512 | 2,668 | 8 | 8 | 96 | 64 | 4 |
| Core Area [mm$^2$] | 195 | 216 | 520 | 561 | < 576† | < 1600† | ~120‡ | ~150‡ | 0.927 |
| Freq. [MHz] | 3,200 | 3,400 | 772 | 837 | 1,250 | 1,200 | 150 | 150 | ≤ 515 |
| Core Perf. [GFLOPS] | 1.89 | 2.01 | 13.45 | 14.86 | 1.67 | 2.63 | 9.99 | 17.64 | 4.12 |
| Core Power [W] | 55.6 | 77.2 | 231.9 | 163.0 | 9.77 | 14.9 | 45.0 | 5.1 | < 0.065 |
| Energy Efficiency [GFLOPS/W] | 0.034 (5,597×) | 0.026 (7,319×) | 0.058 (3,281×) | 0.091 (2,091×) | 0.171 (1,113×) | 0.176 (1,081×) | 0.222 (857×) | 3.46 (55×) | 190 (1×) |

* physical/virtual cores † based on package size ‡ estimated area based on resource usage

Fig. 5 Comparison with prior CPU, GPU, DSP, and FPGA sparse-BLAS implementations.



| | |
|---|---|
| Technology | 40nm 1P10M CMOS FO4 16.3ps (TT) |
| Core $V_{DD}$ | 0.55 to 1V |
| I/O $V_{DD}$ | 1.8V |
| Frequency | from 75 to 515 MHz |
| Power | 64.3 mW (1.0V) 6.73mW (0.6V) |
| Energy Efficiency | ≤ 190 GFLOP/s/W |
| Core Size | 1,431µm × 648µm |
| Transistor Count | 6.98 million |

Fig. 6 Die photo and chip summary.