# A Low-Power, High-Performance, 1024-Point FFT Processor

Bevan M. Baas, *Student Member, IEEE*

***Abstract*—This paper presents an energy-efficient, single-chip, 1024-point fast Fourier transform (FFT) processor. The 460 000-transistor design has been fabricated in a standard 0.7 $\mu$m ($L_{\mathrm{poly}} = 0.6$ $\mu$m) CMOS process and is fully functional on first-pass silicon. At a supply voltage of 1.1 V, it calculates a 1024-point complex FFT in 330 $\mu$s while consuming 9.5 mW, resulting in an adjusted energy efficiency more than 16 times greater than the previously most efficient known FFT processor. At 3.3 V, it operates at 173 MHz—which is a clock rate 2.6 times greater than the previously fastest rate.**

***Index Terms*—Cache memories, chip, CMOS digital integrated circuits, CMOS integrated circuits, digital signal processors (DSP's), discrete Fourier transforms (DFT's), energy efficient, FFT, Fourier transforms, low power, memory architecture.**

## I. Introduction

THE FAST Fourier transform (FFT) is one of the most widely used digital signal processing (DSP) algorithms. While advances in semiconductor processing technology have enabled the performance and integration of FFT processors to increase steadily, these advances have also, unfortunately, led to an increase in power consumption. This has resulted in a situation where the number of potential FFT applications that are limited by power—not performance (e.g., portable applications)—is significant and growing.

For many CMOS circuits, energy dissipation is proportional to the supply voltage squared [1]. Consequently, substantial efficiency can be gained by aggressively reducing the supply voltage [2]. Unfortunately, a lower supply voltage reduces circuit performance. The processor presented here operates with a low supply voltage $V_{\mathrm{dd}}$, which approaches the value of the transistor thresholds $V_t$, to dramatically increase the overall energy efficiency. To regain some lost performance, the processor utilizes a high-performance algorithm and architecture that are shown to perform better than previous designs.

## II. FFT Processor Memory-System Architectures

As with most DSP algorithms, FFT's make frequent accesses to data in memory. FFT's are calculated in $\mathcal{O}(\log_r N)$ *stages*, where $N$ is the length of the transform and $r$ is the radix of the FFT decomposition. Each stage requires the reading and writing of all $N$ data words.

Manuscript received August 7, 1998; revised October 27, 1998. This work was supported by a National Science Foundation fellowship, by the National Aeronautics and Space Administration under GSRP fellowship NGT-70340, and by MOSIS.

The author is with StarLab, Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA.

Publisher Item Identifier S 0018-9200(99)01638-8.

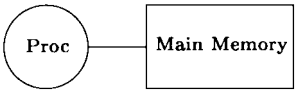



Fig. 1. Single-memory architecture block diagram.

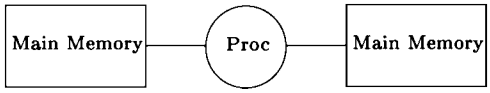



Fig. 2. Dual-memory architecture block diagram.

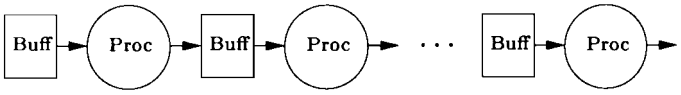



Fig. 3. Pipeline architecture block diagram.

### A. Previous Architectures

*1) Single Memory:* The simplest memory-system architecture is the *single-memory* architecture, as shown in Fig. 1, in which a memory of at least $N$ words is connected to a processor by a bidirectional data bus. In general, data are read from and written back to the memory once for each of the $\log_r N$ stages of the FFT.

*2) Dual Memory:* The *dual-memory* architecture places two memories of size $N$ on separate buses connected to a processor, as Fig. 2 shows. Data begin in one memory and "ping-pong" from memory to memory $\log_r N$ times until the transform has been calculated. The Honeywell DASP processor [3] and the Sharp LH9124 processor [4] use the dual-memory architecture.

*3) Pipeline:* For processors using a *pipeline* architecture, a series of smaller memories replace the $N$-word memory(ies). Either physically or logically, there are $\log_r N$ stages. Fig. 3 shows how processors and buffer memories are interleaved, as well as the flow of data through the pipeline structure. Typically, an $\mathcal{O}(r)$-word memory is on one end of the pipeline, and memory sizes increase by $r$ through subsequent stages, with the final memory of size $\mathcal{O}(N)$. The Logic Corp. (LSI) L64280 FFT processor [5], the FFT processor designed by He and Torkelson [6], and the FFT processor by Bidet *et al.* [7] use pipeline architectures.

*4) Array:* Processors using an *array* architecture are composed of a number of independent processing elements with local buffers, interconnected through some type of network, as depicted in Fig. 4. The Cobra FFT processor [8] uses an array architecture and is composed of multiple chips that each

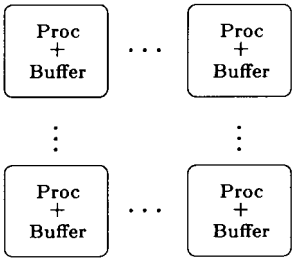



Fig. 4. Array architecture block diagram.

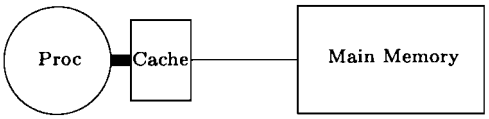



Fig. 5. Cached-memory block diagram.

contain one processor and one local buffer. The FFT processor by O'Brien *et al.* [9] uses an array-style architecture with four datapaths and four memory banks on a single chip.

### B. Cached-Memory Architecture

The proposed *cached-memory* architecture is similar to the single-memory architecture except that a small cache memory resides between the processor and main memory. Fig. 5 shows the tightly coupled processor-cache pair and the $N$-word main memory.

It is well known that data caches increase the effective bandwidth to a memory—but only if the memory access pattern exhibits sufficient locality [10]. Although nearly all FFT algorithms have very poor locality, [11] describes an algorithm that offers good locality over large portions of the computation. Section III gives an overview of the algorithm. In this algorithm, the global communication inherent in the FFT is concentrated into a few (typically one or two) intermediate steps and is easily accomplished through appropriate addressing when filling and flushing the cache. Because the FFT algorithm is deterministic, cache tags are unnecessary, and correct cache operation is achieved through a fixed, predetermined, cache address mapping. Since the flow of data is data independent, data may also be prefetched from main memory before they are needed.

The cached-memory architecture offers two key advantages over other approaches; in particular, it provides:

- increased speed—since smaller memories are faster than larger ones;
- increased energy efficiency—since smaller memories require lower energy per access and typically can be located nearer to the datapath.

When using a cached-memory architecture, the potential gains in speed and energy efficiency are larger with longer length transforms.

However, the algorithm also presents two primary disadvantages, including:

- the addition of new functional units (caches);
- increased controller complexity.

## III. The Cached-FFT Algorithm

### A. Key Features

A distinguishing characteristic of the proposed *cached-FFT* algorithm is that it isolates main memory from the high-speed portion of the processor. In a software implementation, the algorithm allows data to be accessed from a faster level of the memory hierarchy (e.g., register file) rather than a slower level (e.g., data cache or DRAM).

This section presents an overview of the cached-FFT algorithm; [12] provides a detailed description and development.

### B. Definitions

An *epoch* is defined as the portion of the cached-FFT algorithm where all $N$ data words are loaded into a cache, processed, and written back to main memory *once.* Although the number of epochs $E$ can equal one, that case is degenerate, and normally $E \geq 2$.

Given values for $N$ and $E$, the cache size $C$ is found by

$$C = \sqrt[E]{N}. \tag{1}$$

For values of $N$ and $E$ where $\sqrt[E]{N}$ is not an integer, the implementation is *unbalanced,* meaning the number of passes through data in the cache is not the same in every epoch. These cases are addressed in [12].

By storing frequently used data, the data cache reduces traffic to main memory several fold. With a cache, data are read and written to main memory only $E$ times. Therefore, the reduction in memory traffic is

$$\text{Memory traffic reduction multiple} = \log_r(N)/E. \tag{2}$$

This reduction in memory traffic enables more processors to work from a unified main memory and/or the use of a slower, lower power main memory. In all cases, power dissipated accessing data decreases since data words are stored in a smaller memory that is nearer to the datapath.

### C. Examples with $N = 64$

Fig. 6 is a representative dataflow diagram that shows the six stages of a standard 64-point radix-2 FFT. Memory locations are labeled along the vertical axis with the inputs to the FFT on the left and the outputs on the right. The global communication of the FFT is evident—each of the $N$ outputs depends on each of the $N$ inputs.

Fig. 7 is the flowgraph of a 64-point, radix-2, cached-FFT. Butterflies are drawn with heavier lines, and transactions between main memory and the cache—which involve no computation—are drawn with lighter weight lines. This example has two epochs ($E = 2$). From (1), the cache size is then $\sqrt[E]{N} = \sqrt[2]{64} = 8$ words. The diagram highlights a group of butterflies that are calculated together from the cache. Data flow in a consistent pattern within all eight-word groups throughout the entire transform.

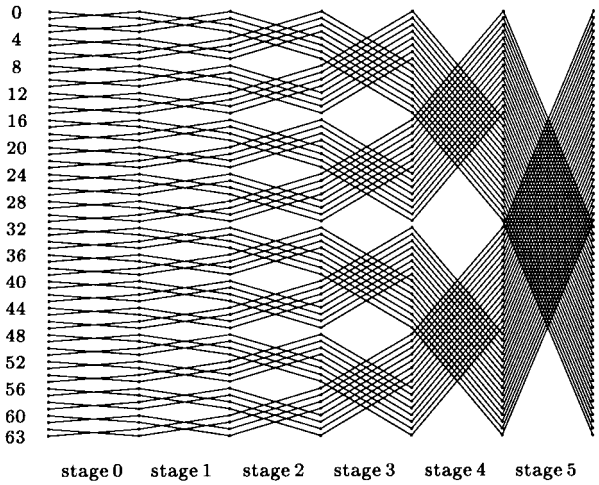



Fig. 6. Dataflow diagram for a standard FFT.

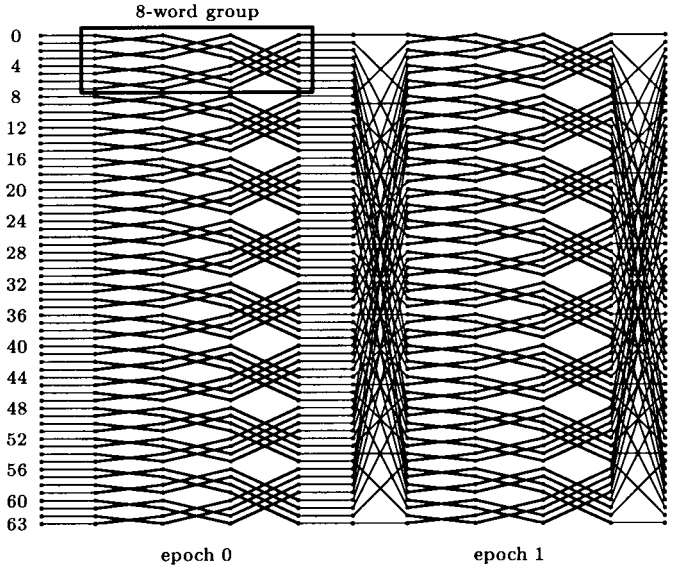



Fig. 7. Cached-FFT dataflow diagram.

### D. Previous Similar Algorithms

As early as 1966, authors proposed FFT algorithms that make use of hierarchical memory structures to increase performance [13]–[18]. Previously proposed algorithms can be viewed as a single two-dimensional decomposition of the discrete Fourier transform. This is essentially the same as a two-epoch case of the more general cached-FFT, providing less insight into how a balanced epoch structure is achieved or how the memory accesses, cache accesses, and processor design can be optimized.

## IV. The Spiffee Processor

A 1024-point single-chip FFT processor named *Spiffee* was designed and fabricated. The full-custom design contains 460 000 transistors and was fabricated in a standard single-poly, triple-metal CMOS process using 0.7 $\mu$m design rules with $L_{\text{poly}} = 0.6$ $\mu$m. PMOS thresholds are $-930$ mV, and NMOS thresholds are 680 mV. The processor occupies 5.985 $\times$ 8.204 mm$^2$ and is fully functional on first-pass silicon.

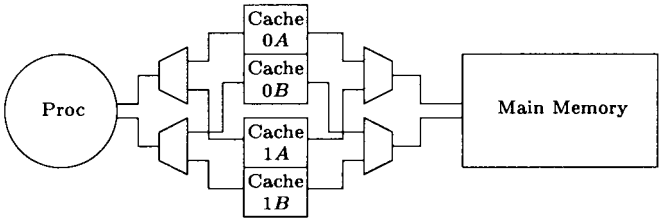



Fig. 8. Block diagram of cached-memory architecture with two cache sets of two banks each.

### A. FFT Algorithm

While higher radix, prime factor, and other FFT algorithms have been shown to require fewer operations than radix-2 algorithms [19], Spiffee was nevertheless designed with a simple radix-2 decomposition. A radix-2 algorithm was chosen because in a very-large-scale-integration implementation, the regularity and simplicity of an algorithm are very important factors in determining the clock speed, design time, debugging effort, and other key parameters.

The two main types of radix-2 FFT algorithms are the decimation-in-time (DIT) and decimation-in-frequency (DIF) varieties [20]. Because the DIT form is slightly more regular, it was chosen for Spiffee. The DIT approach calculates two butterfly outputs, $X = A + BW$ and $Y = A - BW$, from two butterfly inputs, $A$ and $B$, and a complex coefficient $W$. In general, all variables are complex.

Spiffee uses a two-epoch cached-FFT algorithm. With $N = 1024$ and from (1), the cache size $C$ equals $\sqrt[E]{N} = \sqrt[2]{1024} = 32$ words. Although the architecture easily supports multiple processors, the chip presented here contains a single processor/cache pair and a single set of main memory. From (2), the data cache reduces traffic to main memory by a factor of $\log_2(1024)/2 = 5$.

### B. Memory System

Performance of the memory system shown in Fig. 5 can be enhanced by adding a second cache set (zero and one) and partitioning the cache into two banks ($A$ and $B$), as shown in Fig. 8. In this configuration, the processor operates out of one cache set while the other set is being flushed and filled from memory. If the cache flush time plus fill time is less than the time required to process data in the cache (which is easy to accomplish), then the processor will not have to wait for data. The second cache set increases processor utilization, and therefore overall performance, at the expense of some additional area and complexity. The double-banked arrangement increases throughput as it allows an increased number of cache accesses per cycle.

### C. Processor Overview

The processor's datapath calculates one complex radix-2 DIT butterfly per cycle. It operates on complex fixed-point data and has internal datapath widths varying from 20 to 24 bits.

The pipeline has nine stages as shown in Fig. 9. In the first pipeline stage, $A$ and $B$ are read from the caches and $W$

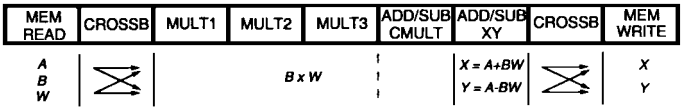



Fig. 9. Pipeline diagram.

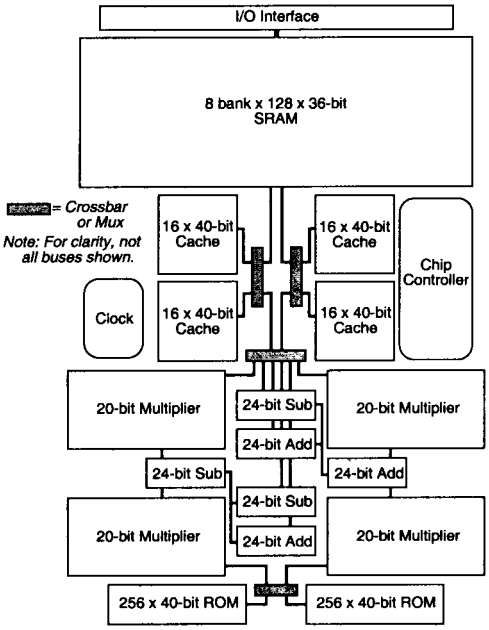



Fig. 10. Chip block diagram.

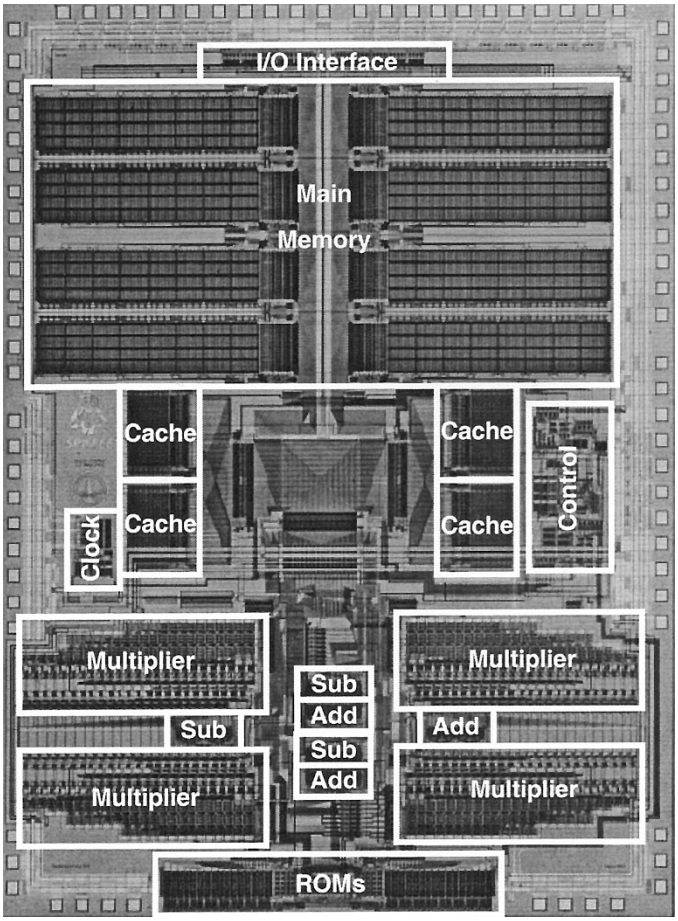



Fig. 11. Die microphotograph.

is read from a ROM. In stage two, the operands are routed through two $2 \times 2$ crossbars to the correct functional units. Four $B_{\{\mathrm{real,\,imag}\}} \times W_{\{\mathrm{real,\,imag}\}}$ multiplications of the real and imaginary components of $B$ and $W$ are calculated in stages three through five. Stage six completes the complex multiplication, stage seven performs the remaining additions or subtractions to calculate $X$ and $Y$, and pipeline stages eight and nine complete the routing and write-back of the results. The deep pipeline causes a read-after-write data hazard to occur once every 80 cycles and is handled by stalling the pipeline for one cycle.

Fig. 10 is a block diagram of the chip, and Fig. 11 is the corresponding die microphotograph.

### *D. Functional Units and Circuits*

The Spiffee processor contains a wide variety of functional units including SRAM, multiported SRAM, ROM, multiplier, and adder/subtracter structures. In addition, the chip also contains crossbar, control, clock generation, and test circuits. Static circuits are used almost exclusively and were optimized for low-power and robust low-voltage operation. The processor is functional at arbitrarily low clock frequencies. This section contains a brief overview of each functional unit type, followed by a discussion of the clocking scheme.

Although first implemented in a standard CMOS process, Spiffee was also designed to operate at very low supply voltages ($V_{\mathrm{dd}} \ll 1$ V) using transistors with very low thresholds [21]. At very low supply voltages, the control of transistor thresholds is critical to circuit performance and is accomplished through the biasing of transistor bodies. Spiffee was designed with n-well and p-substrate nodes not connected to $V_{\mathrm{dd}}$ or $Gnd$ but instead routed to pads to allow the biasing of transistor bodies and thereby adjust threshold voltages [22].

Another area of concern under low-$V_{\mathrm{dd}}$, low-$V_t$ conditions is the control of current leakage into high-fanin nodes. A common high-fanin circuit is the bitline of a memory. The proposed solution to the bitline-leakage problem is the use of a *hierarchical bitline* architecture [11]. Fig. 12 is a schematic of the main memory's SRAM. In this scheme, a column of cells is partitioned into segments by cutting the bitlines at uniform spacings. These short bitlines are called *local bitlines* (*lbl* and *lbl_* in the figure). A second set of bitlines called *global bitlines* (*gbl* and *gbl_* in the figure) are run over the entire column and connected to the local bitlines through connecting transistors. Word accesses are performed by connecting only the one correct local bitline to the global bitline in addition to activating the correct wordline as in a standard approach. Although not as fast as other designs, the sense amplifiers can operate in the presence of a large amount of noise, and they have the forgiving characteristic of being able to correct themselves if they begin latching the wrong value. Because the cached-memory architecture significantly reduces main memory traffic, the speed of the memory is not critical. The main memory is made up of eight 128-word by 36-bit SRAM arrays using six-transistor cells.

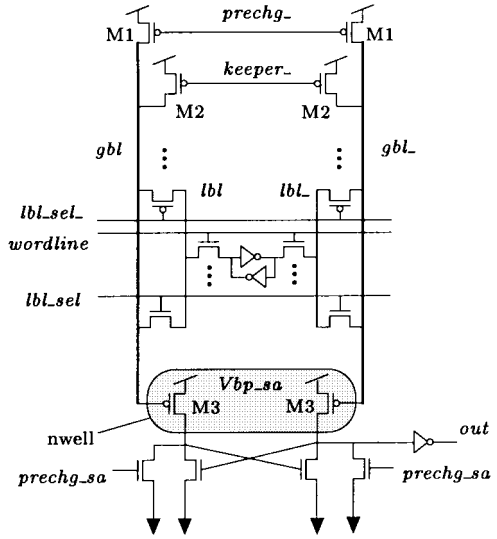



Fig. 12. Schematic of a hierarchical-bitline SRAM.

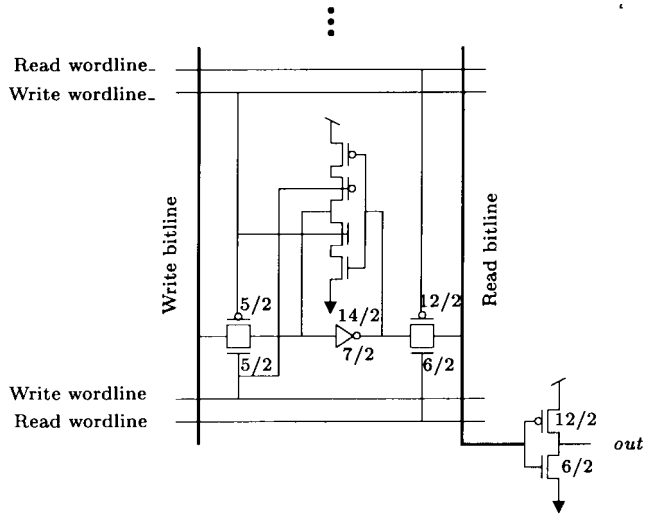



Fig. 13. Simplified schematic of a dual-ported memory array cell.

The calculation of an $N$-point radix-2 FFT requires $N/2$ complex $W_N$ coefficients. The processor presented here stores all $1024/2 = 512$ coefficients in an on-chip ROM. ROM's experience the same bitline leakage problems as SRAM's. To maintain functionality under low-$V_{\rm dd}$, low-$V_t$ conditions, the ROM also uses the hierarchical-bitline technique. It is partitioned into two 256-word by 40-bit arrays. Each array has 16 segments, or local bitlines, corresponding to a maximum of 16 cells per local bitline.

Each of the two cache sets is organized as two banks of 16-word by 40-bit dual-ported SRAM arrays using ten-transistor cells. The memory arrays perform one read and one write per cycle using separate single-ended read and write bitlines. Fig. 13 is a schematic of an SRAM cell with some transistor dimensions indicated as *width*($\lambda$)/*length*($\lambda$) with $\lambda = 0.35\ \mu$m.

Four signed, pipelined, array multipliers produce 24-bit products from 20-bit operands. They employ booth-2 encoding and use (4, 2) and (3, 2) adders to reduce partial products. Since the fixed-point data format requires the eventual truncation of the butterfly's outputs when writing the result back to the cache, it is unnecessary to calculate all 40 bits of the multipliers' outputs. To reduce power dissipation and area, 27% of the partial product bits are not calculated. A bias was added back into the partial product array to set the mean of the truncation error to zero.

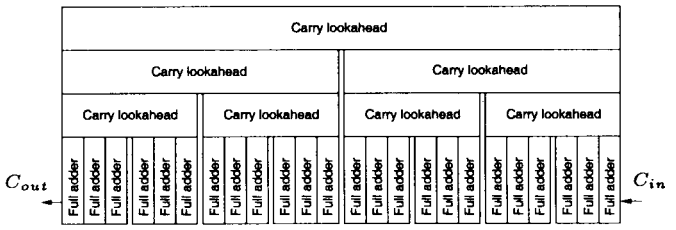



Fig. 14. Twenty-four-bit adder block diagram.

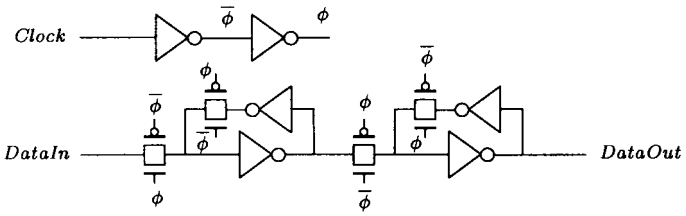



Fig. 15. Schematic for flip-flop with local clock buffer.

Six single-cycle 24-bit adders and subtracters propagate carries using a hybrid of carry-lookahead and ripple techniques. A hybrid approach was chosen because a standard carry-lookahead implementation is faster than what was needed, and larger. Fig. 14 is a block diagram of the adder. At the lowest level, carries ripple across three full-adder blocks. Above three-bit blocks, carry-lookahead circuits accelerate carry propagation through three levels.

The clocking scheme uses one single-phase global clock for the entire processor. The global clock is routed in a single chip-wide network with no local clock buffers. The architecture and functional units achieve a high fraction of utilization—meaning functional units are busy nearly every cycle. High utilization eliminates the need for clock gating. The advantages of a single nonbuffered clock are that it is much simpler to design and has very low clock skew.

Due to their robust operation, static edge-triggered flip-flops are used instead of latches. The schematic of a flip-flop is shown in Fig. 15. SPICE simulations with low-$V_t$ transistors at low-$V_{\rm dd}$ values show that gating the feedback inverters greatly improves the voltage range over which the flip-flops can operate. The use of full transmission gates instead of NMOS pass gates similarly improves operating range. The generation of $\phi$ and $\overline{\phi}$ locally in each flip-flop only slightly increased the size of each cell but reduced the loading on the global clock network by a factor of four. Transistor sizing and the layout of the flip-flop cells were carefully designed and simulated to avoid signal races over wide variations in operating conditions.

## V. Results and Analysis

This section presents and compares measured performance of Spiffee in three ways: first, at a low-power operating point;

TABLE I
MEASURED $V_t$ VALUES

| NMOS: $V_{bs}$ PMOS: $V_{sb}$ (Volts) | NMOS $V_t$ (Volts) | PMOS $V_t$ (Volts) |
|---|---|---|
| −2.0 V | 0.96 | −1.14 |
| 0.0 V | 0.68 | −0.93 |
| +0.5 V | 0.48 | −0.82 |

second, at a high-speed operating point; and third, with silicon area versus *Energy* × *Time* and execution time.

### A. Low-Power Operation

Through biasing of the n-wells and the substrate, it is possible to adjust the thresholds of transistors on the chip. Table I details the 480 and 320 mV $V_t$ shift range that was measured for the NMOS and PMOS, respectively.

The chip was measured functional with $V_{\rm dd}$ slightly below 1.0 V. Because the PMOS thresholds (−930 mV) are so much larger than the NMOS thresholds (680 mV), they are the primary performance limiter at low $V_{\rm dd}$ operating points. At $V_{\rm dd} = 1.1$ V, the processor runs at 16 MHz and 9.5 mW with the n-wells forward biased +0.5 V—which is a 60% speed improvement over the 10 MHz operation without bias. With that bias, 11 $\mu$A of current flows from the n-wells while the chip is active. Latchup was never observed.

Table II contains a summary of relevant characteristics of ten commercial and academic FFT processors calculating 1024-point complex FFT's. *CMOS technology* is the minimum feature size of the CMOS process in which the chip was fabricated. *Datapath width* is the width, in number of bits, of the multipliers for the scalar datapaths. *Number of chips* values with "+" indicate that additional memory chips for data and/or coefficients are required. *Normalized area* is the silicon area normalized to a 0.5 $\mu$m technology with the following relationship:

$$\textit{Normalized Area} = \frac{\textit{Area}}{(\textit{Technology}/0.5\ \mu\text{m})^2}. \tag{3}$$

The final column, *FFT's per energy*, compares the number of 1024-point complex FFT's calculated per unit of energy using (4). It attempts to factor out technology and the datapath word width $DPath$ by making use of the observation that roughly 1/3 of the energy consumption of the 20-bit Spiffee processor scales as $DPath^2$ (e.g., multipliers) and approximately 2/3 scales linearly with $Dpath$

$$\textit{FFT's/Energy} = \frac{\textit{Tech} \times \left(\frac{2}{3}\frac{DPath}{20} + \frac{1}{3}\left(\frac{DPath}{20}\right)^2\right)}{\textit{Power} \times \textit{Exec Time} \times 10^{-6}}. \tag{4}$$

Fig. 16 compares Spiffee's adjusted energy efficiency with other processors. At $V_{\rm dd} = 1.1$ V, Spiffee is 16 times more energy efficient than the previously most efficient known processor.

### B. High-Performance Operation

At $V_{\rm dd} = 3.3$ V, the processor is fully functional at 173 MHz—calculating a 1024-point complex FFT in 30 $\mu$s while consuming 845 mW.

While clock speed is not the only factor, it is certainly an important factor in determining the performance and area efficiency of a design. Fig. 17 compares clock speeds of this cached-FFT design running at $V_{\rm dd} = 3.3$ V with other FFT processors versus their CMOS technologies. Spiffee operates with a clock frequency that is 2.6 times greater than the next fastest processor. Though stressing the device beyond its specifications, the processor is functional at 201 MHz with $V_{\rm dd} = 4.0$ V.

Despite having a favorable maximum clock rate, the chip's circuits are not optimized for high-speed operation—in fact, nearly all transistors in logic circuits are near minimum size. The processor owes its high speed primarily to its algorithm and architecture, which enable the implementation of a deep and well-balanced pipeline.

### C. Other Comparisons

A popular metric that incorporates measures of both energy efficiency and performance is *Energy* × *Time*, or $E \times T$. Using values from Table II, we define it as

$$\textit{Energy} \times \textit{Time} = \frac{\textit{Exec Time}}{\textit{FFT's/Energy}}. \tag{5}$$

Since the quantity *FFT's/Energy* is compensated, to first order, for different *Technology* and *Datapath* values, the $E \times T$ product is also compensated. Fig. 18 compares the $E \times T$ values for various FFT processors versus their silicon areas, normalized to 0.5 $\mu$m. The dashed line highlights a constant $\textit{Area}' \times E \times T$ contour. The most comprehensive metric we consider is the product $\textit{Area}' \times E \times T$. The Spiffee processor running at a supply voltage of 2.5 V has an $\textit{Area}' \times E \times T$ product that is 17 times lower than the processor with the previously lowest value. Although the exact minimum $E \times T$ value for Spiffee was not measured, its magnitude is expected to be fairly constant for supply voltages in the vicinity of $3V_t$ [24]. Except for the Cobra processor, which uses 44 times as much area, the Spiffee processor has an $E \times T$ product less than one-third of the $E \times T$ values of previous processors.

The cost of a device is a strong function of its silicon area. So processors with high performance and small area are the most cost efficient. Fig. 19 shows the first-order-normalized FFT calculation time *(Exec Time/Technology)* versus normalized silicon area for several FFT processors. The dashed line indicates a constant $\textit{Area}' \times \textit{Time}'$ contour. The processor presented here compares favorably with other processors despite additional cache memories, circuits designed for low-voltage operation and $V_t$ tunability, and a less-than-optimum floorplan. However, some processors have more than a single set of main memory, and adding a second set to Spiffee would increase its area by approximately 30%.

TABLE II
COMPARISON OF PROCESSORS CALCULATING 1024-POINT COMPLEX FFT's

| Processor | CMOS Tech ($\mu$m) | Datapath width (bits) | 1024-pt Exec Time ($\mu$sec) | Power (mW) | Clock Freq (MHz) | Num of chips | Norm Area (mm$^2$) | FFTs per Energy |
|---|---|---|---|---|---|---|---|---|
| LSI, L64280 [5] | 1.5 | 20 | 26 | 20,000 | 40 | 20 | 233 | 2.9 |
| Plessey, 16510A [9] | 1.4 | 16 | 98 | 3,000 | 40 | 1 | 22 | 3.6 |
| Honeywell, DASP [3] | 1.2 | 16 | 102 | $\sim$ 5,250 | – | 2+ | – | 1.7 |
| Y. Zhu, U of Calgary | 1.2 | 16 | 155 | – | 33 | – | – | – |
| Dassault Electronique | 1.0 | 12 | 10.2 | 15,000 | 25 | 6 | 240 | 3.4 |
| Tex Mem Sys, TM-66 | 0.8 | 32 | 65 | 7,000 | 50 | 2+ | – | 3.4 |
| Cobra, Col. State [8] | 0.75 | 23 | 9.5 | 7,700 | 40 | 16+ | 1104+ | 12.4 |
| Sicom, SNC960A | 0.6 | 16 | 20 | 2,500 | 65 | 1 | – | 9.0 |
| CNET, E. Bidet [7] | 0.5 | 10 | 51 | 300 | 20 | 1 | 100 | 13.6 |
| M. Wosnitza, ETH [23] | 0.5 | 32 | 80 | 6000 | 66 | 1 | 167 | 2.4 |
| Spiffee, $V_{dd} = 3.3$V | 0.7/0.6 | 20 | 30 | 845 | 173 | 1 | 25 | 27.6 |
| Spiffee, $V_{dd} = 2.5$V | 0.7/0.6 | 20 | 41 | 363 | 128 | 1 | 25 | 47.0 |
| Spiffee, $V_{dd} = 1.1$V | 0.7/0.6 | 20 | 330 | 9.5 | 16 | 1 | 25 | **223** |

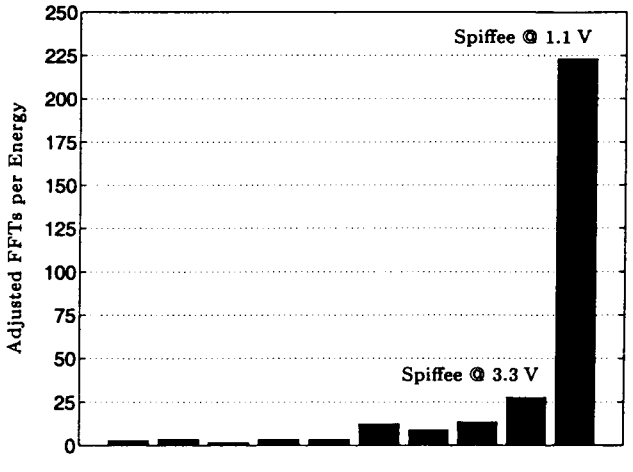



Fig. 16. The adjusted energy efficiency [FFT's/Energy, see (4)] of various FFT processors.

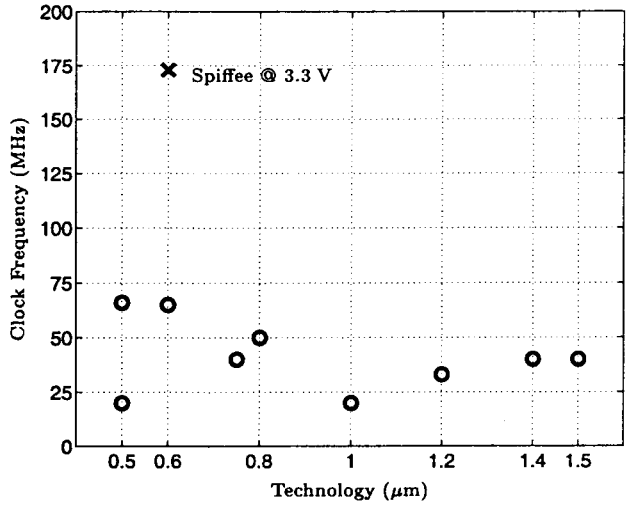



Fig. 17. CMOS technology versus clock frequency for various FFT processors.

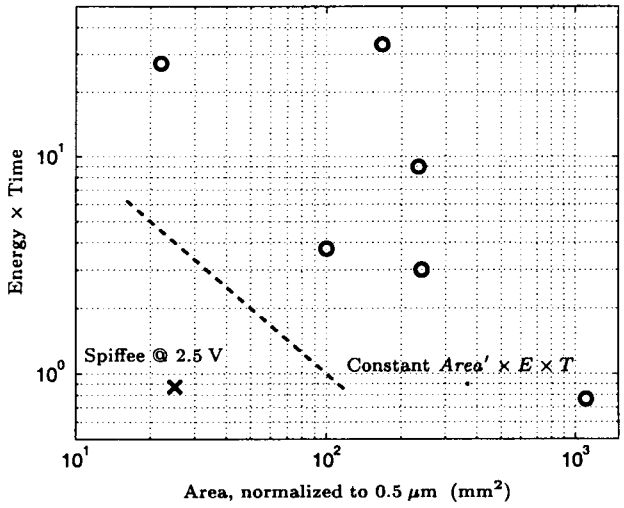



Fig. 18. Silicon area versus Energy × Time [see (5)] for several FFT processors.

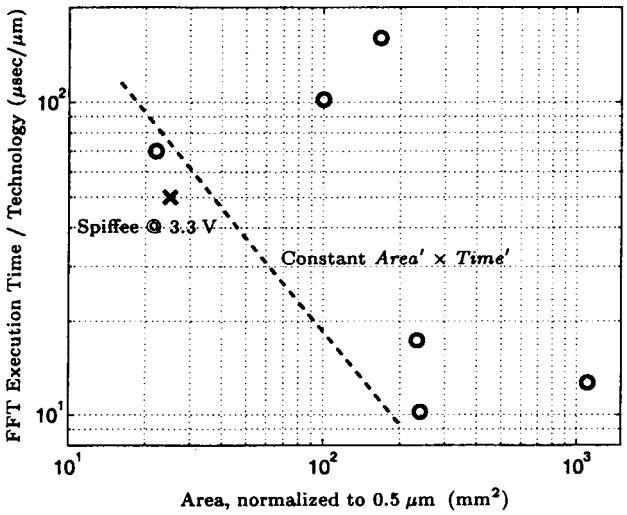



Fig. 19. Silicon area versus FFT execution time for several FFT processors.

## VI. CONCLUSION

An FFT architecture and algorithm, which are optimized for a processor with a memory system containing a cache, were introduced. A single-chip FFT processor that uses a cached main memory and realizes increased energy efficiency (through reduced communication energy) and increased performance (through a deep and well-balanced pipeline) was

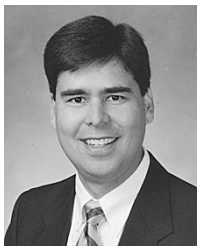
also presented. With PMOS thresholds of −930 mV, the processor is fully functional over a supply range of 1.0–4.0 V. At a supply voltage of 1.1 V, the device is 16 times more energy efficient than the previously most efficient known FFT processor; at a supply voltage of 3.3 V, it is functional with a clock rate more than 2.6 times greater than the previously fastest.

## ACKNOWLEDGMENT

The author gratefully acknowledges valuable guidance and mentoring from J. Burr, M. Matsui, and G. L. Tyler.

## REFERENCES

[1] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design.* Reading, MA: Addison-Wesley, 1985.
[2] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, pp. 473–483, Apr. 1992.
[3] S. Magar, S. Shen, G. Luikuo, M. Fleming, and R. Aguilar, "An application specific DSP chip set for 100 MHz data rates," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, Apr. 1988, vol. 4, pp. 1989–1992.
[4] "LH9124 digital signal processor user's guide," Sharp, Camas, WA, 1992.
[5] P. A. Ruetz and M. M. Cai, "A real time FFT chip set: Architectural issues," in *Proc. Int. Conf. Pattern Recognition*, June 1990, vol. 2, pp. 385–388.
[6] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 1998, pp. 131–134.
[7] E. Bidet, D. Castelain, C. Joanblanq, and P. Senn, "A fast single-chip implementation of 8192 complex point FFT," *IEEE J. Solid-State Circuits*, vol. 30, pp. 300–305, Mar. 1995.
[8] G. Sunada, J. Jin, M. Berzins, and T. Chen, "Cobra: An [*sic*] 1.2 million transistor expandable column FFT chip," in *Proc. IEEE Int. Conf. Computer Design*, Oct. 1994, pp. 546–550.
[9] J. O'Brien, J. Mather, and B. Holland, "A 200 MIPS single-chip 1k FFT processor," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 1989, vol. 36, pp. 166–167, 327.
[10] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 2nd ed. San Francisco, CA: Morgan Kaufmann, 1996.
[11] B. M. Baas, "An energy-efficient single-chip FFT processor," in *Proc. Symp. on VLSI Circuits*, June 1996, pp. 164–165.
[12] ——, "An approach to low-power, high-performance fast Fourier transform processor design," Ph.D. dissertation, Stanford University, Stanford, CA, to be published.
[13] W. M. Gentleman and G. Sande, "Fast Fourier transforms—For fun and profit," in *Proc. AFIPS Conf.*, Nov. 1966, vol. 29, pp. 563–578.
[14] R. C. Singleton, "A method for computing the fast Fourier transform with auxiliary memory and limited high-speed storage," *IEEE Trans. Audio Electroacoust.*, vol. AU-15, pp. 91–98, June 1967.
[15] N. M. Brenner, "Fast Fourier transform of externally stored data," in *IEEE Trans. Audio Electroacoust.*, vol. AU-17, pp. 128–132, June 1969.
[16] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1975.
[17] D. Gannon and W. Jalby, "The influence of memory hierarchy on algorithm organization: Programming FFT's on a vector multiprocessor," in *The Characteristics of Parallel Algorithms*, L. Jamieson, D. Gannon, and R. Douglass, Eds. Cambridge, MA: MIT Press, 1987, ch. 11, pp. 277–301.
[18] D. H. Bailey, "FFTs in external or hierarchical memory," *J. Supercomput.*, vol. 4, no. 1, pp. 23–35, Mar. 1990.
[19] C. S. Burrus and T. W. Parks, *DFT/FFT and Convolution Algorithms.* New York: Wiley, 1985.
[20] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1989.
[21] J. B. Burr and J. Shott, "A 200mV self-testing encoder/decoder using Stanford ultra-low-power CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 1994, vol. 37, pp. 84–85, 316.
[22] J. B. Burr and A. M. Peterson, "Ultra low power CMOS technology," in *Proc. NASA VLSI Design Symp.*, 1991, pp. 4.2.1–4.2.13.
[23] M. Wosnitza, M. Cavadini, M. Thaler, and G. Tröster, "A high precision 1024-point FFT processor for 2D convolution," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 1998, vol. 41, pp. 118–119, 424.
[24] M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-power digital design," in *Proc. IEEE Symp. Low Power Electronics*, Oct. 1994, vol. 1, pp. 8–11.



**Bevan M. Baas** (S'94) received the B.S. degree in electronic engineering from California State Polytechnic University, San Luis Obispo, in 1987 and the M.S. degree in electrical engineering from Stanford University, Stanford, CA, in 1990, where he currently is pursuing the Ph.D. degree in the Department of Electrical Engineering.

From 1987 to 1989, he was with Hewlett-Packard, Cupertino, CA, where he participated in the development of the processor for a high-end minicomputer. His current research interests are in the areas of high-performance and energy-efficient digital signal-processing algorithms, architectures, and circuits.

Mr. Baas was an NSF Fellow from 1990 to 1993 and a NASA GSRP Fellow from 1993 to 1996.