# A Low Power, High Performance Approach for Time-Frequency/Time-Scale Computations

Bruce W. Suter

Air Force Research Laboratory/IFGC
525 Brooks Rd
Rome NY 13441

Kenneth S. Stevens

Intel
Strategic CAD Labs
Portland, OR 97124

## ABSTRACT

This paper presents an application of formal mathematics to create a high performance, low power architecture for time-frequency and time-scale computations implemented in asynchronous circuit technology that achieves significant power reductions and performance enhancements over more traditional approaches. Utilizing a combination of concepts from multirate signal processing and asynchronous circuit design, a case study is presented dealing with a new architecture for the fast Fourier transform, an algorithm that requires globally shared results. Then, the generalized distributive law is presented as an important paradigm for advanced asynchronous hardware design.

**Keywords:** time-frequency computations, time-scale computations, asynchronous circuit design, multirate signal processing, generalized distributive law, fast Fourier transform

## 1. INTRODUCTION

In spite of the relentless reduction of feature sizes in CMOS technology, the structure of design and its figures of merit have been evolving slowly. The increase in transistor count and the reduction of feature size in each process generation is increasing the importance of power, skew, increasing process variations, and the increased capacitance of non-local communication.

The ability to view a die as a unified circuit controlled by a single frequency becomes less viable as design size increases. We feel that future architectures will be modular where each section contains its own frequency domain, and where they they communicate via point-to-point unidirectional communication links.

We decided to focus on new design approaches in order to investigate bringing formal mathematics to bear on new design realities. A fast Fourier transform (FFT) architecture, which is efficient in terms of performance and power, has been created as a case study. This FFT work is a precursor to investigating more general high performance, low power architectures. Towards this end, Aji and McEliece[1] have recently defined a unified theory of fast algorithms that they call the *Generalized Distributive Law*. This class of algorithms, which includes the FFT, are characterized by their application of the distributive law to perform factorizations. Since this approach permits a unified strategy for considering both numerical and nonnumerical algorithms, further study is needed for its application.

## 2. CASE STUDY - FAST FOURIER TRANSFORM

The following section describes a successful case study that was undertaken to utilize concepts from multirate signal processing and asynchronous circuit design to achieve a low power, high performance design. The fast Fourier transform was chosen since it is an algorithm that requires globally shared results.

Our mathematical approach is hierarchically formed and expressed in terms of the $W_N = exp(-j\frac{2\pi}{N})$ notation as shown in Equation 1. The derivation can be found in.[2]

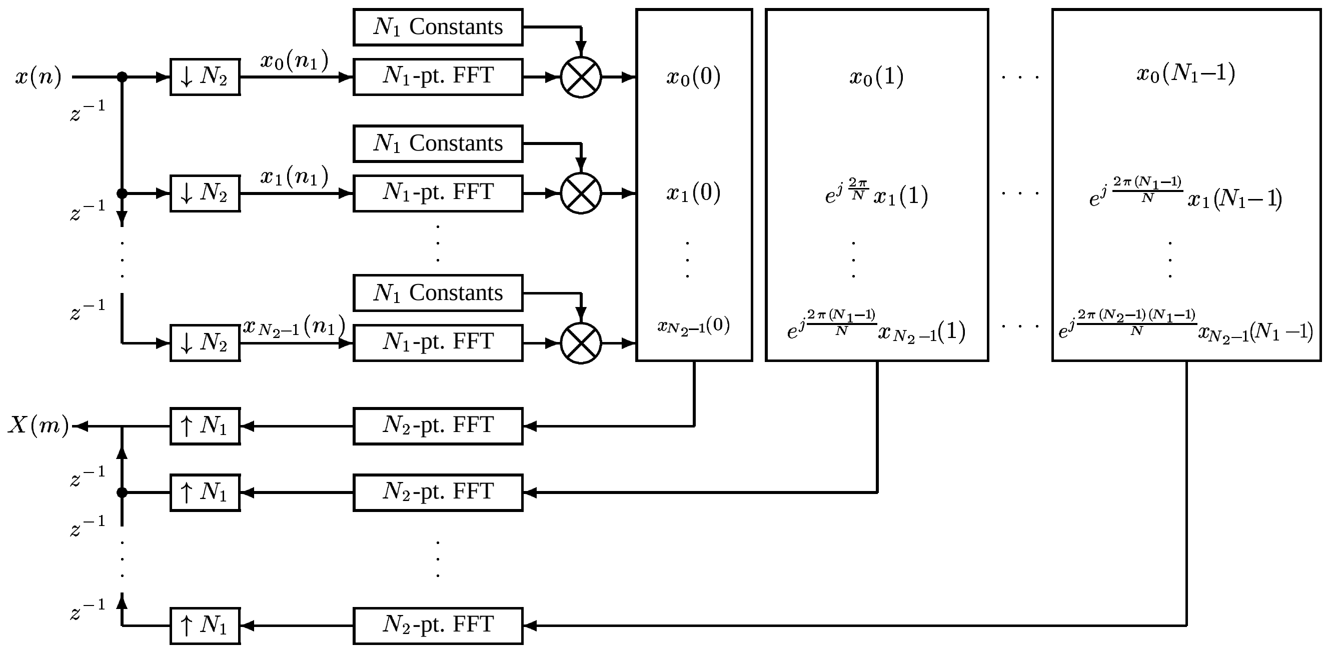

**Figure 1.** Low Power FFT Architecture

$$X_{m_1}(m_2) = \sum_{n_2=0}^{N_2-1}\left[W_N^{m_1 n_2}\sum_{n_1=0}^{N_1-1} x_{n_2}(n_1)W_{N_1}^{m_1 n_1}\right]W_{N_2}^{m_2 n_2} \tag{1}$$

This notation represents $N_2$ FFTs using $N_1$ values as the inner summation, which are scaled and then used to produce $N_1$ FFTs of $N_2$ values. The total operation achieves the desired FFT of size $N$.

Historically, equations for FFT systems similar to our approach have been developed for two applications. In the mid 1960's the problem of computing the FFT of a vector that was too large to fit in main memory was addressed. An approach similar to that presented here was created to limit the storage requirements in these primitive systems.[3] A second similar approach was achieved in the 1980's for multiprocessor applications of the FFT algorithm. The underlying architectures created from these equations are vastly different than that achieved here.[4]

The goals of this case study were to attempt to take a common application area and investigate novel formal architectural approaches to architect low power and high performance with additional constraints on what we project future designs will require. We therefore emphasized in our formulation pipelining, increasing localization, hierarchy, and establishing multiple frequency domains where we attempt to push the critical path into concurrent lower frequency domains to support high performance.

The multiplicative complexity of our approach is the same as the conventional Cooley-Tukey FFT formulation, which is $O(NlogN)$. But, our approach permits localized computations, as opposed to globally computing butterflies. This in turn suggests a low power silicon implementation, which is shown in Figure 1.

The multirate formulation of this algorithm has resulted in an implementation parallelized in a pipelined fashion. Each "row" in the architecture contains point-to-point unidirectional data pipelines. The entire design is implemented using asynchronous finite state machines for control.

The frequency of each horizontal track in the architecture operates at $\frac{1}{N_2}$ the frequency of the initial sample rate due to decimation – an average cycle time of 160ns. The asynchronous design methodology allows the rate division to occur locally with much of the circuit idle consuming only leakage current when the operation is complete.

The down arrow blocks of Figure 1 are decimators.[5] The output of the M-fold decimator for a sampled signal $X(n)$ is given by $y(n) = x(Mn)$. This is effectively a demux operation where each output is selected in order.

| Chip | Power per transform |
|---|---|
| DSP-24 (DSP Arch.) | 143$\mu$J / transform |
| SPIFFEE-1 (Stanford) | 50$\mu$J / transform |
| space FFT | 97$\mu$J / transform |
| earth FFT | 18$\mu$J / transform |

**Table 1.** FFT Power Comparison

| Chip | Throughput |
|---|---|
| DSP-24 (DSP Arch.) | 48k transforms/sec |
| SPIFFEE-1 (Stanford) | 33k transforms/sec |
| Our design | 25,000k transforms/sec |

**Table 2.** FFT Performance Comparison

Each of the $N_1$ and $N_2$ blocks represent another FFT operation which can be a hierarchical instantiation of the structure in the figure where the values of $N_1 \times N_2$ equals $N_1$ or $N_2$ at the higher level in the hierarchy.

The product blocks multiply a stream of results coming from the $N_1$ point FFT units by a set of constant values. Both constants and results are complex numbers, requiring four multiplications and two additions per sample. The constants are calculated by $W_N^{m_1 n_2}$, where $m_1 = 0, \ldots, N_1 - 1$ and $n_2 = 0, \ldots, N_2 - 1$.

The large pipeline switch maps results from the product block to the $N_2$ FFT units. The $N_2$ FFT units take a transform of time displaced Fourier transform samples. Each $N_1$-point FFT provides one data sample to each of the $N_2$-point FFT units, the first row providing the first sample.

A stream of data $x_0(m_2), \ldots, x_{N-1}(m_2)$ is output by the $N_2$ FFT units to an array of expanders.[5] The output frequency of the expanders increases $N_1$ fold, with each expander cell providing a single data sample.

This case study illustrates potential benefits of a mathematical approach to design for low power and implemented using self-timed methodologies. We have designed and submitted to MOSIS a circuit containing the FFT-4 logic using a radiation

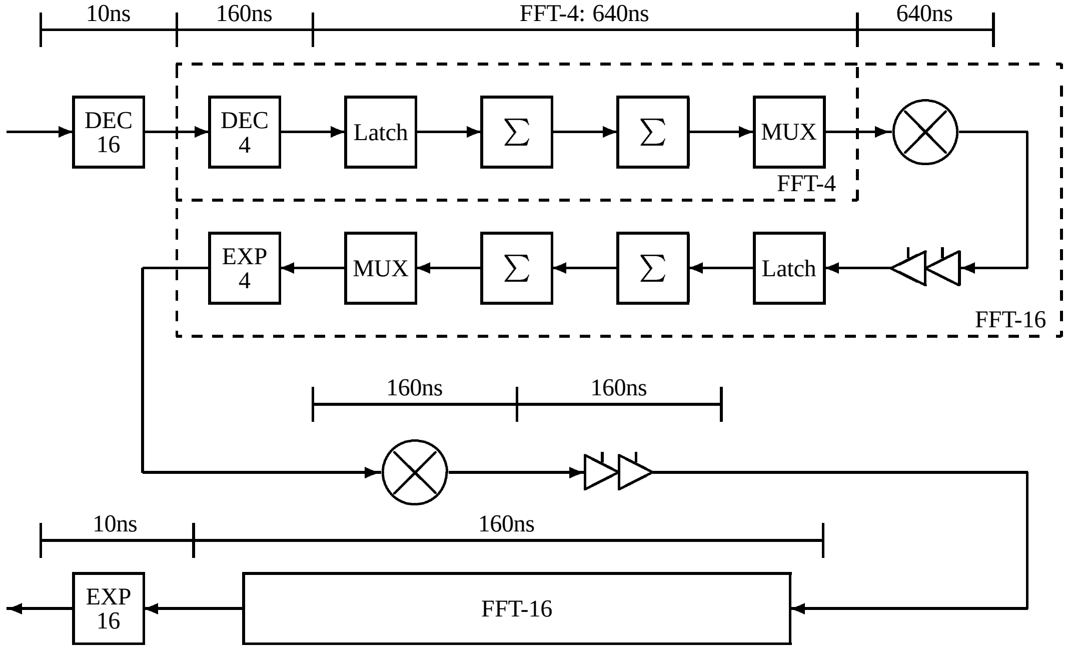

**Figure 2.** Data Path For Low Power FFT Architecture

| Local Domain | Local Kernel |
|---|---|
| $\{x_1\}$ | $\alpha_1(x_1)$ |
| $\{x_2\}$ | $\alpha_2(x_2)$ |
| $\{x_1, x_2, x_3\}$ | $\alpha_3(x_1, x_2, x_3)$ |
| $\{x_2, x_4\}$ | $\alpha_4(x_2, x_4)$ |
| $\{x_3, x_5\}$ | $\alpha_5(x_3, x_5)$ |

**Table 3.** Set Up for GDL Example

tolerant cell library in 0.8$\mu$m CMOS. The power consumption of the fabricated FFT-4 and completed implementation of the FFT-16 has been used to estimate the overall power efficiency of a 1024-point FFT. These results are shown in comparison with other FFT designs in Table 1. In addition to these reductions in power consumption, we achieved a remarkably high sustained throughput as can be observed in Table 2. All designs are measured using a 3.3V voltage source.

Figure 2 projects out a single "row" of the architecture presented in Figure 1. This shows the data path from input sample to output in a 256-point FFT ($N_1 = N_2 = 16$). Note that the data path switches between frequency domains of 100MHz, 6.26MHz, and 1.5625MHz. The higher frequency domains - such as input decimation and output expansion - are zones of reduced concurrency whereas the more concurrent operations function at decreased frequency. The ultra low frequency of the bulk of the circuit permits compact low energy circuit implementations, such as ripple-carry adders rather than look-ahead or array type adders, for the summation and product blocks.

## 3. GENERALIZED DISTRIBUTIVE LAW (GDL)

We believe that the GDL is an important new paradigm for advanced asynchronous hardware design. The following section provides some insight into this powerful new methodology.

The GDL is a general technique that includes as special cases: FFT, matrix multiplication, Kalman filtering, Viterbi's algorithm, Pearl's belief propagation algorithm, turbo coding, and many more. To put this in the proper frame of reference, first recall the distributive theorem, which we all learned in grade school. It states that given three variables $a, b, c$ then $ab + ac = a(b + c)$. From a computational point of view the distributive law reduced the number of computations from three to two. Thus, the distributive law can be viewed as a fast algorithm. The GDL is a generalization of the distributive law in that it is based on the idea of factoring a global kernel into the product of local kernels. Moreover, it can be shown that all *fast* algorithms are based on the GDL.

The *general problem* here is to calculate tables of one or more *local marginalizations* of the global kernel, *i.e.*

$$\beta_1(x_1, x_4) = \sum_{x_2, x_3} \beta(x_1, x_2, x_3, x_4). \tag{2}$$

Recall that a tree is a connected graph with no cycles and that in a junction tree all nodes involving a given variable must be in communication with one another. This leads us to the following result of Shafer and Shenoy[6]: If the local domain is organized as a junction tree, then there exists a fast algorithm based on decentralized message passing on the junction tree. In other words, a junction tree is a communications network in which an edge from vertex $v_i$ to vertex $v_j$ is a communications channel that filters out by marginalization dependence on all variables but those common to $v_i$ and $v_j$. Although Shafer and Shenoy[6] were the first to describe this junction tree paradigm, Aji and McEliece[1] were the first to realize the broad applicability of this formulation to a wide variety of algorithms. To illustrate these ideas, consider the example, which is defined in terms of its local domains and associated local kernels in Table 3. Then, a possible junction tree for this computation is given in Figure 3.

To see the relevance of this discussion to current hardware practices, we can reflect on some comments made by Bill Brownhill, principal Alpha designer, at the recent VLSI Circuit Symposium. He stated that the clock distribution on the Alpha now takes 33-40% of the chip power. So, they have decided to get rid of the global clock on their next generation Alpha chip. Instead, they will use "multiple clock zones"; a local high speed clock in each of them; a globally asynchronous, locally synchronous architecture; and a mixture of adaptive and passive synchronizers for inter-zone communication.

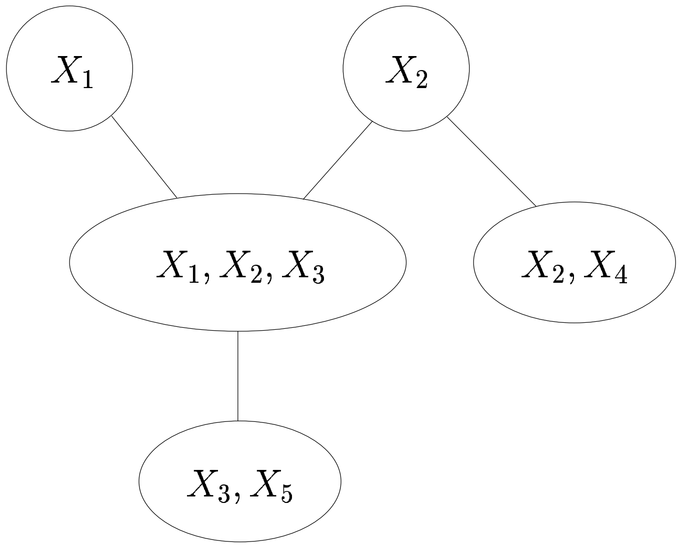

**Figure 3.** Junction Tree for GDL Example

## 4. CONCLUSIONS

CMOS process technology should continue to scale to feature sizes that will support highly concurrent architectures. Elegant mathematical design approaches, as presented in this paper, can be designed for such technology to implement ultra high performance low power systems. We feel that a combination of greater locality and parallelism permits various frequency domains which are exploited to increase throughput and decrease energy per operation. Asynchronous design methodologies support modular design, low power synchronization, and scale well to projected future technologies. The robustness of the unclocked synchronization directly supports the functionally equivalent factorizations of multirate and GDL optimizations. As we look to the future, we are currently exploring the use of the GDL as a paradigm for advanced hardware design.

## REFERENCES

1. S. M. Aji and R. J. McEliece, “The Generalized Distributive Law,” tech. rep., California Institute of Technology, 1998.
2. B. W. Suter and K. S. Stevens, “Low Power, High Performance FFT Design,” in *Proceedings of IMACS World Congress on Scientific Computation, Modeling, and Applied Mathematics*, pp. 99–104, 1997.
3. W. Gentleman and G. Sande, “Fast Fourier Transforms for Fun and Profit,” in *AFIPS Conference Proceedings*, vol. 29, pp. 563–578, 1966.
4. D. Bailey, “FFTs in External or Hierarchical Memory,” *Journal of Supercomputing* **4**, pp. 23–35, 1990.
5. B. W. Suter, *Multirate and Wavelet Signal Processing*, Academic Press, 1997.
6. G. R. Shafer and P. P. Shenoy, “Probability Propagation,” *Ann. Math. Art. Intel.* **2**, pp. 327–352, 1990.