Low Power, High Performance FFT Design[1]

B. W. Suter, K. S. Stevens
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433-7765, USA
e-mail: b.suter@ieee.org

ABSTRACT

Architecture and circuit design are the two most effective means of reducing power in CMOS VLSI. This paper presents an application of formal mathematics to create a power efficient architecture of an FFT implemented in asynchronous circuit technology that achieves significant power reduction over other FFT architectures. Multirate signal processing concepts are applied to the FFT to localize communication and remove the need for globally shared results in the FFT computation. A novel architecture is produced from the polyphase components that is mapped to an asynchronous architecture. The asynchronous design continues the localization of communication and can be mapped to standard cell libraries such as radiation-hardened libraries for space electronics.

We present a methodology based on multirate signal processing techniques and asynchronous design style that supports significant reduction in power over conventional design practices.

INTRODUCTION

Low Power is becoming increasingly important in design both for mobile and desktop applications. Moreover, the importance of low power is rivaling performance as the primary figure of merit in integrated electronics.

The Semiconductor Industry Association, a consortium of United States semiconductor manufacturers, publishes their predictions for future integrated circuit technology. Their latest report predicts[1], based on the last 20 years, the future of circuit technology. An analysis of the trends reveals the alarming effect that power related issues will have on the industry. Voltage scaling in battery powered electronics is predicted to max out at 0.9 volts by 2004, and at the same voltage for desktop systems in 2010 as feature size decreases to 0.07 microns. Current will increase an order of magnitude inside a chip due to voltage scaling as well as power consumption doubling over that time frame. Current in high-end desktop systems is predicted to increase from 24 amperes today to 200 amperes!

Thus, reduction in supply voltage as a means of controlling power consumption is asymptotically limited by current and resistivity of wires at very low voltages and small feature sizes. Other mechanisms for controlling power will be increasingly required in future technologies.

We show that architecture and design style can be combined to make a significant an impact on design for low power. Moreover, this approach can be mapped to circuits today, as well as in future technologies.

---

[1]patent pending

Fast Fourier Transform

The FFT was chosen as an example for our low power architectural experiment due to its broad applicability and availability in commercial integrated circuits. The Fourier Transform is a principle analytical tool in many diverse fields, including linear systems, optics, probability theory, antennas, radar, and signal analysis. The complexity of computing a Discrete Fourier Transform (DFT) is $O(N^2)$, the square of the number of sample points examined.

Many well known algorithms have been applied based on the periodicity of the FFT to reduce the number of multiplications and additions required to compute the Fourier Transform. These algorithms are collectively called Fast Fourier Transforms (FFTs)[2]. FFT algorithms reduce the complexity in terms of complex multiplications to $O(N\log N)$.

Our approach is to mathematically design an FFT algorithm based on architectural metrics that will yield a high throughput, low power design. Merely attempting to create an algorithm that is computationally simplified creates architectural tradeoffs that reduce the utility of the approach - as can be seen in other FFT designs where tradeoffs between butterfly networks, busses, memory structure and hierarchy, and radix of the algorithm become the critical design issues limiting both performance and power.

This paper will first outline the metrics used to drive our architectural decisions, the mathematics behind our FFT derivation, then discuss the mapping of the algorithm to an architecture, and the architecture to an energy efficient implementation.

ARCHITECTING FOR LOW POWER

Our approach is to design a methodology that applies to standard logic design practices and processes of today and based on the SIA predictions of the future. Such techniques can therefore be adopted directly into designs.

Power consumption in an IC can be split into static and dynamic sources. Static power dissipation comes from two main sources - leakage and current based circuits. In todays processes, leakage current is low - and can be maintained at about the magnitude of the auto-discharge current of batteries. Although leakage issues will become more important in the future, we ignore design techniques to mitigate leakage in this paper and focus on achievements made in dynamic power dissipation.

Dynamic power dissipation in static and precharged CMOS is linearly dependent on the load and activity of a node, and quadratically dependent on the supply voltage. Although supply voltage scaling makes the largest impact to dynamic power dissipation, its effect on architectures comes from a performance perspective rather than from a power perspective. Since performance degrades linearly with voltage scaling, we must increase the performance of our design to match throughput under voltage scaling. Therefore, voltage scaling requires *parallelism*.

The parasitics of a design are mainly impacted by architecture and circuit design style. Local signals will have small parasitics when compared to more global signals such as memory busses and clocks. Fanout also has a cumulative effect on circuit parasitics. Point-to-point signals therefore have much smaller load than centralized caches and signals that synchronize multiple devices. Architecture and circuit design style are not

entirely orthogonal, and our architecture is based on asynchronous design. Our design style eschews global structures such as clocks, busses, and shared memory in preference to pipelined and local communication. The impact of reducing parasitics can be massive across designs, and may be the largest contribution to low power architectures using standard logic.

The final method of reducing power is to reduce the number of transitions required to complete a computation. Such transformations are directly supported by asynchronous design styles, which are monotonic, optimized for the common case, and have variable delay and complexity in pipeline stages. The monotonic nature of asynchronous logic will not permit hazards which have been reported to account for up to 40% of the power in some designs. Further, common-case optimization allows power to be optimized for common operations at the expense of de-optimizing uncommon operations, whereby the overall cost is minimized. The flexibility in pipeline design permits the design to be architected around data flows rather than clock frequencies - reducing the need to unnecessarily latch and shift data in the design. Further, the local handshaking implements "clock gating" at the transistor level at no extra cost.

Our goals of locality, flexibility in the pipeline stages, and circuit design for low power were applied to our FFT example. Multirate signal processing techniques were applied to a mathematical derivation of an FFT to produce an architecture free of shared memory with a greatly simplified communication network that has significantly fewer parasitics than other implementations.

MATHEMATICAL DERIVATION

Historically, similar equations for FFT systems have been derived 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. A derivation similar to that presented here was created to limit the storage requirements in these primitive systems[3]. A second similar derivation 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].

Consider the following equation for an FFT

$$X(m) = \sum_{n=0}^{N-1} x(n) W_N^{mn} \tag{1}$$

where

$$W_N = e^{-j\frac{2\pi}{N}} \tag{2}$$

We can now assume that N is not prime, that is $N = N_1 N_2$. By the division theorem for integers, we can let $m = m_2 N_1 + m_1$ and $n = n_1 N_2 + n_2$ where $m_1, n_1 = 0, 1, \ldots, N_1 - 1$ and $m_2, n_2 = 0, 1, \ldots, N_2 - 1$.

Given a sequence $x(n)$, then its polyphase components[5] are defined as $x_k(n) = x(M_n + k)$, $k = 0, \ldots, M - 1$. Now, we can break the FFT computation up using this polyphase

notation into interdependent "equivalent classes" of calculations by letting $X_{m_1}(m_2) = X(m_2 N_1 + m_1)$ and $x_{n_2}(n_1) = x(n_1 N_2 + n_2)$. Using this polyphase representation, we can represent our FFT as

$$X_{m_1}(m_2) = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} x_{n_2}(n_1) e^{-j\frac{2\pi(m_2 N_1 + m_1)(n_1 N_2 + n_2)}{N}} \tag{3}$$

or equivalently,

$$X_{m_1}(m_2) = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} x_{n_2}(n_1) e^{-j\frac{2\pi m_2 N_1 n_1 N_2}{N}} e^{-j\frac{2\pi m_2 N_1 n_2}{N}} e^{-j\frac{2\pi m_1 n_1 N_2}{N}} e^{-j\frac{2\pi m_1 n_2}{N}} \tag{4}$$

Since the first exponential term is equal to unity, we can simplify as:

$$X_{m_1}(m_2) = \sum_{n_2=0}^{N_2-1} \left[e^{-j\frac{2\pi m_1 n_2}{N}} \sum_{n_1=0}^{N_1-1} x_{n_2}(n_1) e^{-j\frac{2\pi m_1 n_1}{N_1}} \right] e^{-j\frac{2\pi m_2 n_2}{N_2}} \tag{5}$$

Using the W_N notation we can express our final form as

$$X_{m_1}(m_2) = \sum_{n_2=0}^{N_2-1} \left[W_N^{m_1 n_2} \sum_{n_1=0}^{N_1-1} x_{n_2}(n_1) W_{N_1}^{m_1 n_1} \right] W_{N_2}^{m_2 n_2} \tag{6}$$

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.

FFT IMPLEMENTATION

The goal of this work is to distribute the FFT computation on silicon in a much more efficient and localized manner than previous work. Given the above derivation, an efficient silicon implementation is still not trivial. One possible implementation is depicted in Figure 1. Each of the N_1 and N_2 blocks represent another FFT operation where each can also be broken up using the same structure as in the figure where the values of N_1 and N_2 will be different.

The down arrow blocks are called decimators[5]. For a sampled signal $X(n)$, the output of the M-fold decimator is given by $y(n) = x(Mn)$. The sampling of the N_2 decimators are arranged in a regular repeating fashion where the first sample is taken by the first decimator, the second by the second and so on. After the last decimator has sampled the data, the first decimator then takes its second sample. The second sample from each decimator is segregated from the first by being assigned to a different sequence of input data. The decimators must map the high frequency input signal to the low frequency of the FFT design. The decimation function can be designed using simple asynchronous control and steering logic where the data points are steered in order to the correct FFT blocks. A logarithmic decimator network is used as a tradeoff between complexity, power, delay, and shared structures. Our analysis shows that simple 1:2 or 1:4 decimators best optimize the requirements.

$x(n)$ $\downarrow N_2$ $x_0(n_1)$ N_1 Constants N_1 - pt. FFT $\otimes$

z^{-1}

$\downarrow N_2$ $x_1(n_1)$ N_1 Constants N_1 - pt. FFT $\otimes$

z^{-1}

z^{-1} $\downarrow N_2$ $x_{N_2\text{-}1}(n_1)$ N_1 Constants N_1 - pt. FFT $\otimes$

$x_0(0)$ $x_0(1)$ $\cdots$ $x_0(N_1\text{-}1)$

$x_1(0)$ $e^{j\frac{2\pi}{N}}x_1(1)$ $\cdots$ $e^{j\frac{2\pi(N_1\text{-}1)}{N}}x_1(N_1\text{-}1)$

$x_{N_2\text{-}1}(0)$ $e^{j\frac{2\pi(N_2\text{-}1)}{N}}x_{N_2\text{-}1}(1)$ $\cdots$ $e^{j\frac{2\pi(N_2\text{-}1)(N_1\text{-}1)}{N}}x_{N_2\text{-}1}(N_1\text{-}1)$

N_2 - pt. FFT N_2 - pt. FFT $\cdots$ N_2 - pt. FFT

$X(m)$ $\uparrow N_1$ $x_0(m_2)$

z^{-1}

$\uparrow N_1$ $x_1(m_2)$

z^{-1}
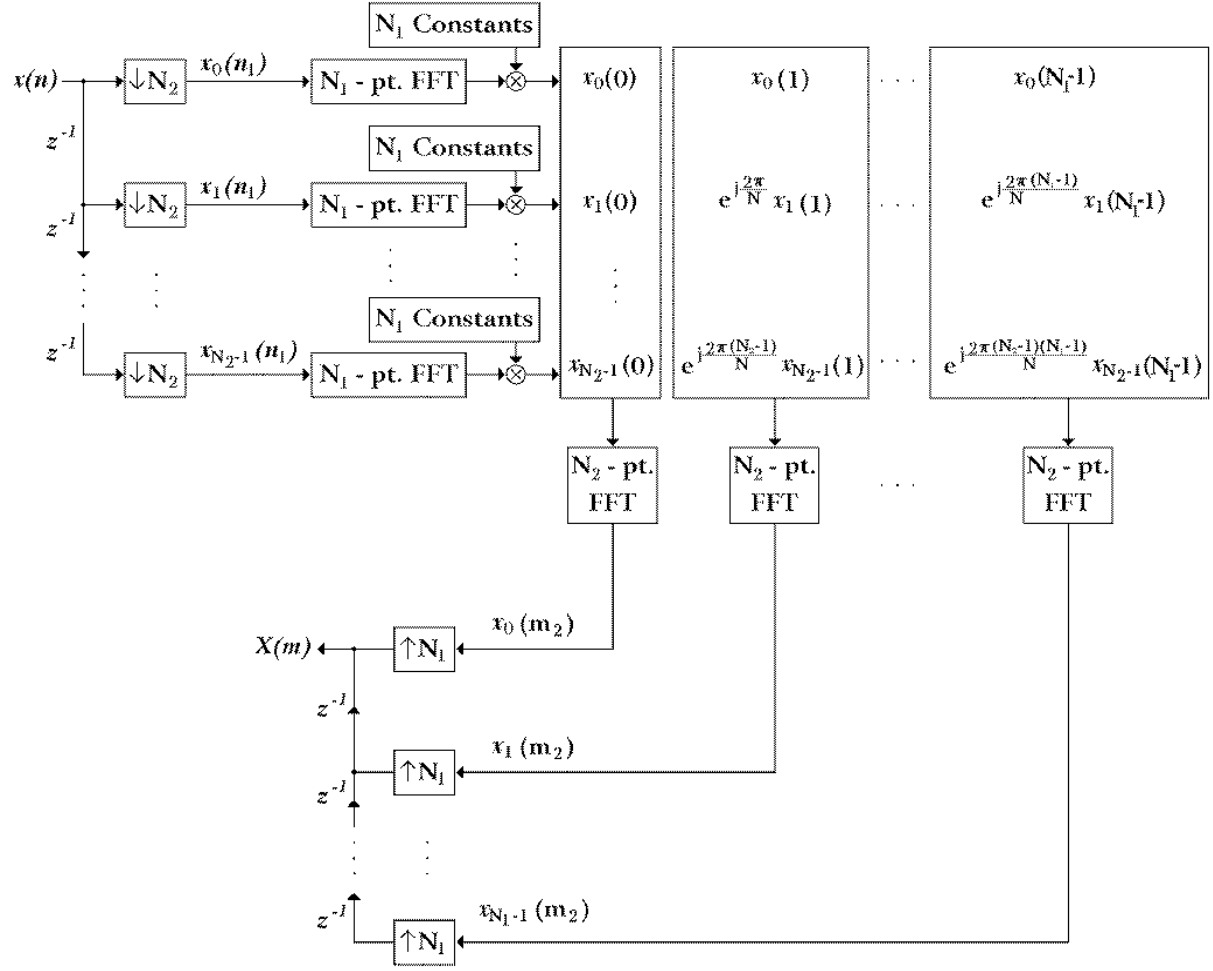
z^{-1} $\uparrow N_1$ $x_{N_1\text{-}1}(m_2)$

Figure 1: Low Power FFT Architecture

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. The asynchronous design methodology allows the rate division to occur locally with much of the circuit idle consuming only leakage current.

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 $e^{-j\frac{2\pi m_1 m_2}{N}}$ where $m_1 = 0, \ldots, N_1 - 1$ and $m_2 = 0, \ldots, N_2 - 1$. Therefore all constants in the top product block are unity, and the product logic is replaced with a wire in the asynchronous implementation. Every other product block contains at least one unity constant per set of N_1 constants. The asynchronous implementation of the product logic optimizes for power such that one, negation, and zero operations consume little power.

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 and outputs data streams $x_0(m_2), \ldots, x_{N-1}(m_2)$ to an array of expanders[5]. 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.

Finally, the up arrow blocks are called expanders[5]. For the signal $x_k(m_2)$, the output of the M-fold expander is given by

$$y(n) = \begin{cases} x_k\left(\frac{m_2}{M}\right) & , \quad \text{if } m_2 \text{ is a multiple of } M \\ 0 & , \quad \text{otherwise} \end{cases} \tag{7}$$

The expander design is akin to that of the decimator.

CONCLUSIONS

A new VLSI design methodology has been presented that illustrates the tradeoffs possible between power dissipation and chip area utilized. A more detailed discussion of engineering analysis and associated tradeoffs will be presented in [6].

REFERENCES

[1] *The National Technology Roadmap for Semiconductors*, Semiconductor Industry Association, 1997 revision of 1994 Report.

[2] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM(1992).

[3] W. Gentleman and G. Sande, *Fast Fourier Transforms for Fun and Profit*, Proc. AFIPS 29, p.563–578(1966).

[4] D. Bailey, *FFTs in External or Hierarchical Memory*, J. Supercomputing 4, p.23–35(1990).

[5] B. Suter, *Multirate and Wavelet Signal Processing*, Academic Press(1997).

[6] B. Suter and K. Stevens, *A New VLSI Design Paradigm*, in preparation.