A Single Chip Low Power Asynchronous Implementation of an FFT Algorithm for Space Applications*

B. W. Hunt K. S. Stevens† B. W. Suter D. S. Gelosh
Department of Electrical and Computer Engineering
Air Force Institute of Technology
WPAFB, OH 45433

Abstract

A fully asynchronous fixed point FFT processor is introduced for low power space applications. The architecture is based on an algorithm developed by Suter and Stevens specifically for a low power implementation. The novelty of this architecture lies in its high localization of components and pipelining with no need to share a global memory. High throughput is attained using large numbers of small, local components working in parallel. A derivation of the algorithm from the discrete Fourier transform is presented followed by a discussion of circuit design parameters specifically those relevant to space applications. A survey of this application specific architecture is included with a detailed look at the design of the complex-valued Booth multiplier to demonstrate the design methodology of this project. Finally, simulation results based on layout extractions are presented and an outline for future work is given.

1 Introduction

This paper discusses the background, design, and implementation methodology of an asynchronous fixed point FFT processor. The algorithm used to perform the FFT was specifically tailored for low power hardware realizations using asynchronous communication. This project has been motivated by several factors. First, a formal mathematical approach to develop low power high performance numerical applications is being investigated by Suter and Stevens [8]. An implementation of this circuit is being designed and will be fabricated to accurately evaluate and validate the power benefits of this formal approach. The effort to take an academic paper design to a complete integrated circuit is also motivated by the desire to understand interface issues more accurately as well as validate the design methodology that will be presented later. The need for low power signal processing in space applications is particularly vexing, and presented an excellent target for this design.

A brief mathematical derivation of the architecture based on wavelet theory is presented. Background information regarding micro-electronic radiation hardening and how it effects low power designs is discussed. A brief description of the overall architecture is presented followed by the design methodology using a simple component of the design as an example. The paper concludes with some simulation power and speed results as well as expected results from future test chips.

2 Background

2.1 The Suter/Stevens FFT Algorithm

The Discrete Fourier Transform, or DFT, is a common signal processing algorithm that supports a variety of uses. The FFT or Fast Fourier Transform reduces the complexity, in terms of multiplication and additions, of the DFT. The DFT is typically characterized as in Equation 1.

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

where

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

Briefly, a wavelet approach was applied by Suter and Stevens to create a hardware realization that parallelizes and localizes the computations of an FFT in such a way as to reduce the overall power consumption. We can represent N in Equation 1 as $N_1 N_2$. Then, by the division theorem for integers, let $m = m_2 N_1 + m_1$ and $n = n_1 N_2 + n_2$ where $m_1, n_1 = 0, 1, \cdots, N_1 - 1$ and $m_2, n_2 = 0, 1, \cdots, N_2 - 1$. If you are given a sequence, $x(n)$, its polyphase components [9] are defined as $x_k(M_n + k)$ where $k = 0, 1, \ldots, M-1$. Now, the original N point FFT problem can be divided into equivalence classes where $X_{m_1}(m_2) = X(m_2 N_1 + m_1)$ and

*This work is supported by a grant from the Space Technology Directorate of Phillips Laboratory, Kirtland Air Force Base, New Mexico

†Now with Intel Corp.

$x_{n_2}(n_1) = x(n_1 N_2 + n_2)$. Using this polyphase notation of the FFT enables the DFT from Equation 1 to be written as

$$X_{m_1}(m_2) = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} x_{n_2}(n_1) e^{S_1} \quad (3)$$

$$S_1 = -j\frac{2\pi(m_2 N_1 + m_1)(n_1 N_2 + n_2)}{N} \quad (4)$$

With the exponential numerator multiplied out, the result is

$$X_{m_1}(m_2) = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} x_{n_2}(n_1) e^{S_{2a}} e^{S_{2b}} e^{S_{2c}} e^{S_{2d}} \quad (5)$$

$$S_{2a} = -j\frac{2\pi m_2 N_1 n_1 N_2}{N} \quad (6)$$

$$S_{2b} = -j\frac{2\pi m_2 N_1 n_2}{N} \quad (7)$$

$$S_{2c} = -j\frac{2\pi m_1 n_1 N_2}{N} \quad (8)$$

$$S_{2d} = -j\frac{2\pi m_1 n_2}{N} \quad (9)$$

where the first exponential is equal to unity. Further simplification leads to

$$X_{m_1}(m_2) = \sum_{n_2=0}^{N_2-1} \left[e^{S_{3a}} \sum_{n_1=0}^{N_1-1} x_{n_2}(n_1) e^{S_{3b}} \right] e^{S_{3c}} \quad (10)$$

$$S_{3a} = -j\frac{2\pi m_1 n_2}{N} \quad (11)$$

$$S_{3b} = -j\frac{2\pi m_1 n_1}{N_1} \quad (12)$$

$$S_{3c} = -j\frac{2\pi m_2 n_2}{N_2} \quad (13)$$

or

$$X_{m_1}(m_2) = \sum_{n_2=0}^{N_2-1} \left[W_N^{m_1 n_2} \sum_{n_1=0}^{N_1-1} x_{n_2}(n_1) W_{N_1}^{m_1 n_1} \right] W_{N_2}^{m_2 n_2} \quad (14)$$

in W_N notation. The motivation behind manipulating the FFT into the form of Equation 14, and how this maps to a low power implementation will be more clear following a discussion of the general architecture in Section 3.1.

General Notes on the Suter/Stevens Algorithm
This algorithm can be implemented hierarchically, allowing the N_1 and N_2 blocks to be a smaller instantiation of the algorithm. For instance, if $N = 1024$ where $N_1 = 16$, and $N_2 = 64$, the N_1 and N_2 blocks can each be decomposed into $N_{11} = N_{12} = 4$ and $N_{21} = N_{22} = 8$. Four point FFTs can be implemented without multiplication, so a hierarchical decomposition which maps the leaf nodes to four-point FFTs is the most efficient realization. Our test chip design implements a simple 16-point FFT with $N_1 = N_2 = 4$ so that all the components of a larger FFT circuit are included. From this, it is easy to see that FFT point sizes that are an *even* power of the base FFT point size work best.

Although a synchronous implementation of this algorithm is realizable, the multirate algorithm based on concurrent execution of each decimated sequence maps very well to an asynchronous implementation. The shifts between time domains occurs naturally and with minimal energy expense.

2.2 Designing for Space Applications

In the harsh environment of space, there are many radiation hazards that must be eliminated or minimized for a circuit to perform reliably. Several types of radiation are harmful to CMOS circuits and have differing effects, including neutron radiation, ionizing radiation, and total-dose radiation. Each one of these forms of harmful radiation can cause a single event effect (SEE) which is defined as either a hard or soft error. Hard errors include latchup, burnout, gate rupture, frozen bits, noise in CCDs, and snapback. Soft errors include bit upsets in memories or registers or transient signals in logic circuits. These soft errors are commonly referred to as *single event upsets*, or SEU [1].

Ionizing radiation, for example, is caused mainly by gamma and x-rays as well as other minor sources and primarily affects the oxide layers of a CMOS circuit [2]. Upon irradiation, electron-hole pairs are generated and evenly distributed throughout the SiO_2 layer. Many of these pairs recombine within 100μsec, but some free electrons are swept out by the electric field in the gate insulator. The trapped holes that remain in the insulator cause a negative shift in the MOSFET threshold voltage. Over time, the holes slowly migrate toward the most negative potential within the SiO_2. If this most negative potential is the channel, the holes will tend to migrate toward the insulator-channel interface decreasing V_t from its pre-rad or initial value. If this most negative potential is the gate, then the holes migrate toward the insulator-gate interface decreasing V_t from the initial value. After a period of time, the holes are annealed out of the SiO_2, allowing V_t to shift back toward its initial value [3, 7].

The key parameter changes caused by ionizing radiation is the threshold voltage shift. For P-FETS, V_t is shifted negatively at all dose levels because the trapped holes in the oxide and the interface states work together. For N-FETS, V_t is shifted negatively at low dose levels

and positively at high dose levels [3].

A thin gate oxide allows fewer electron-hole pairs to be formed, mitigating the effects of ionizing radiation. A reentrant form of the N-FET also keeps the area of the field oxide at a minimum, further reducing the amount of oxide that can be "ionized".

The following section discusses how rad hardening effects low power CMOS designs.

2.3 Designing Low Power Space Applications

Solar panels and nuclear generators are the only way a satellite can acquire energy. Therefore both peak and standby power must be kept to a minimum.

A common method of reducing power consumption in integrated circuits is to lower the supply voltage, yielding a quadratic improvement in power at a linear cost in performance. However, scaling the voltage of a CMOS circuit allows it to become more susceptible to SEU because the noise margin between a logic high and a logic low is reduced. SEU possibilities are further acerbated due to the threshold shifts that occur under radiation. Therefore, voltage scaling must be used judiciously and in general has more restrictions than in earthbound electronics.

The power and complexity required to implement many CMOS functions can be reduced using circuit techniques such as dynamic, pre-charge, and pass-gate logic. Unfortunately these techniques are also to be avoided since the single event effects can prey very easily on these structures. Design is largely limited to static logic gates.

Fortunately standard CMOS processes can be used. A radiation tolerant[1] cell library developed jointly by Mission Research Corporation (MRC)and Air Force Research Laboratory[4] specifically for the HP 0.8 μm process via MOSIS has been used for our test chip. However, the radiation requirements result in devices much larger than would be used otherwise, resulting in additional power consumption. For example, the minimum size *inverter* width in this cell library is 50 λ for the N-FET and 90 λ for the P-FET! With the additional rad tolerant characteristics, the total size of the inverter cell is 42×119 λ.

Architecture becomes the primary means of reducing power in space applications, due to constraints to voltage scaling, circuit structure, and device size. This FFT architecture implemented using asynchronous circuits significantly reduces the power compared to other space worthy designs. The most significant contribution to low power in this architecture are twofold. First, the algorithm has been designed to maximize locality, point-to-point data pipelining, and hierarchy. The only shared structure in the design is the decimator inputs, expander outputs, and pipelined crossbar switch (all discussed in Section 3.1). Second, the frequency is greatly reduced by decimation allowing devices and drivers to be undersized. This can significantly reduce the capacitance of transmitting data signals. For example, assume a 100MHz sample rate of a 256-point FFT. This design requires additions at a low frequency rate of 320ns in the leaf FFT cells. The pipelined crossbar transmits one data word across each row and column every 160ns. This allows ample latency to size devices optimally. The asynchronous implementation technology allows the common frequency changes to be supported at minimal energy dissipation.

3 Architecture[2]

3.1 General Architecture

There are six major components required to implement the FFT of Equation 14. The block diagram of Figure 4 shows how each of the components fit into the computation. First of all, the data is decimated in time into N_2 sequences of length N_1. Then, the FFT of each sequence is computed (the interior summation), followed by the multiplication of the constants ($W_N^{m_1 n_2}$). After the complex multiply, the partial transformed data is interleaved, as required by the FFT, through a pipelined crossbar switch. This pipes a data stream to the N_2 point FFT blocks. Finally the data, which is decimated in frequency, goes to an expander to correctly sequence each fully transformed element in the output sequence as $X(m)$ and regenerate the input frequency.

3.2 Specific Architecture for This Project

For test chip, the minimum iteration necessary to demonstrate the functionality of the algorithm was desired. Therefore $N = 16$ was chosen with $N_1 = N_2 = 4$. This allowed the basic architecture to reuse the N_1 and N_2 blocks, drastically saving on area with only a minimum of control overhead.

Control operates in data-flow pipelines, using data bundling and four-phase handshaking protocol between each stage. Control of every major component is implemented using one-hot encoded state machines [6]. There are 16 unique burst-mode asynchronous finite state machines (AFSMs) used in the control structures. Some of the AFSMs are very simple like the ones used to control register locking which have 3 states with 2 inputs

[1] Rad tolerant implies the ability to withstand 100 krad(Si) and maintain whereas rad hard implies the ability to withstand 1 Mrad(Si)

[2] patent pending

and 2 outputs. Others are more complex like the multiplier controller which has 9 states with 8 inputs and 6 outputs.

3.3 Formal Specifications

Table 1 displays the CCS specifications for a 2:1 four-cycle decimator and expander controller. Larger decimators and expanders can be specified by increasing the number of interfaces DIFC or EIFC. While we have several implementations of these circuits for our test chip, we will leave the design and synthesis of these cells as a challenge for the synthesis tools in the asynchronous community.

Table 1: CCS Specifications for Decimator and Expander Cells

```
agent DIFC       =  tn.r.a.r.a.DIFC
agent DIN        =  req.t1.ack.req.ack.req.t2.ack.req.ack.DIN
agent DECIMATOR  =  (DIN | DIFC[t1/tn, r1/r, a1/a] |
                     DIFC[t2/tn, r2/r, a2/a]) \ { t1, t2 }
```

```
agent EIFC       =  r.tn.a.r.a.EIFC
agent EOUT       =  t1.req.ack.req.ack.t2.req.ack.req.ack.EOUT
agent EXPANDER   =  (EOUT | EIFC[t1/tn, r1/r, a1/a] |
                     EIFC[t2/tn, r2/r, a2/a] \ { t1, t2 }
```

4 Design Process

4.1 Design Methodology

Our design methodology uses VHDL as the central simulation and specification language. Designs are refined using VHDL, and when we are satisfied with the simulation results we then implement the blocks. Although there are many tools available today for asynchronous design, the necessary tools are not integrated into a tool flow so the simulation, synthesis, and implementation steps were disjoint.

Generic timing diagrams and petri-nets were devised to demonstrate the flow of control and data through the computation and to help understand interface timing and sequencing. Burst mode AFSM specifications were derived and synthesized by the 3D [10] and MEAT [5] tools. The synthesized equations were written in behavioral VHDL, analyzed and simulated with a test bench modeling the timing diagrams to validate the burst specifications. Once the VHDL was confirmed, the static logic equations were laid out using the MRC cell library. Some of the asynchronous designs were testable with IRSIM (v9.02 from the Berkeley tool suite) and others would only function using SPICE (Avanti Corporation HSPICE v95.1). Occasionally, it would be found that a design was too big to be practical or its function could not deal with the simulated real hazards of the circuit and the design would need to be repartitioned or redesigned.

4.2 An Example of Iterative Component Improvement

The design of the complex multiply block of Figure 4 will be used as a design example. This is the most time-critical component in the design, since a complex multiply must occur every 160ns to meet the intended 10ns sample rate. A radix-4 Booth multiplication algorithm was chosen.

The shift-and-add control was implemented with a sequence of nine one-hot cells (eight for shifting, and the ninth to indicate a "done" condition). On each successive pulse, a different combination of three bits are enabled onto the three decoding lines going to the Booth decoder. The final design of the multiplier is shown in Figure 1.

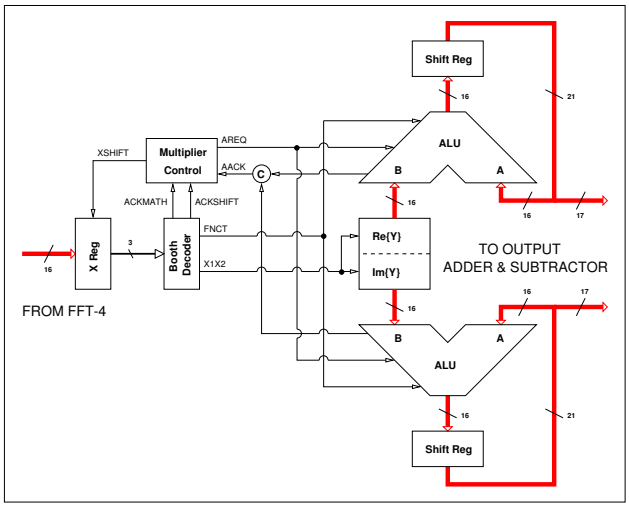

Figure 1: Dual Fixed Point Multiplier

Note that two multiplications are performed in parallel using a single control unit. As can be seen from Figure 4, the FFT-4 produces a pipelined stream of real and imaginary data words to the complex product block. This data is multiplied in the product block by a complex constant. The FFT-4 first produces the real data component followed by the imaginary data component. Since both the real and imaginary constants are always available (stored locally in a static register bank), time and area are conserved by decoding the Booth instructions from the FFT-4 input and producing two partial products simultaneously. Although all the control in the multiply unit can be shared, only the control for the ALUs is illustrated to simplify the figure. Since the Booth instruction will be the same for both multiplies, the AREQ signal is routed to both ALUs. A C-element is required to synchronize the AACK signals from each ALU because the ALU operating time is data depen-

dent and the real and imaginary constant additions will likely complete at different times. Each data path has its own ALU, shift register, and constant storage.

A complex multiply requires four integer multiplications, an integer addition, and an integer subtraction. We use the "multiplier^2" block with two adders and registers to complete a complex multiply. Table 2 gives the step by step procedure of the complex multiply operation corresponding to Figure 2 parts a to d.

	Operation	
1	Receive Re$\{X\}$ from FFT-4	Figure 2 (a)
2	Multiply Re$\{X\}$ by Re$\{Y\}$ and Im$\{Y\}$ creating the two partial products $XrYr$ and $j * XrYi$	Figure 2 (b)
3	Receive Im$\{X\}$ from FFT-4	Figure 2 (b)
4	Multiply Im$\{X\}$ by Re$\{Y\}$ and Im$\{Y\}$ creating the two partial products $j * XiYr$ and $XiYi$	Figure 2 (c)
5	Subtract $XiYi$ from $XrYr$ to produce Re$\{Z\}$ and add $j * XrYi$ and $j * XiYr$ to produce Im$\{Z\}$	Figure 2 (d)

Table 2: Complex Multiplication Operation Steps

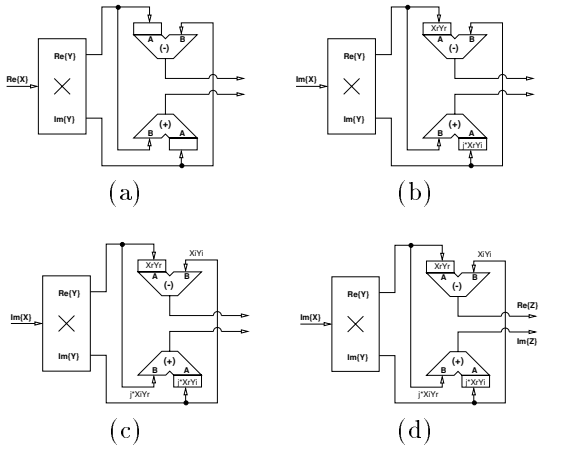

Figure 2: Complex Multiplier Data Path Operation

As shown in Figure 2, each output of the multiplier^2 block connects to one A and B input of an adder and subtracter. Each A input contains a latch to store the integer product of the first two multiplications as described in step 2 of Table 2. The second multiplication result can be passed directly to the adder/subtracter and used with the latched value to produce the complex valued result. Figure 2(a) shows the circuit after the arrival of Re$\{X\}$. The first two partial products have been computed and latched in Figure 2(b). Since the FFT-4 will likely produce its outputs faster than the multiplier can use them, Im$\{X\}$ will probably arrive early. Despite this, Im$\{X\}$ will not be used until after it is latched in step 3 of Table 2. The second multiplication pair has completed in Figure 2(c) and held statically on the data lines. The final step of Table 2 occurs when all four integer multiplication products are present, and the final complex products can be computed and latched into the crossbar switch, as shown in Figure 2(d).

5 Results

We are able to obtain some preliminary power and timing results based on SPICE simulations on extracted layouts. The MRC cell library is designed to be maximally radiation tolerant when Vdd is 5.0 Volts. However, 2.2V is customary for many of today's low power designs due to the benefits of voltage scaling. Our preliminary numbers use a middle-ground Vdd of 3.3 Volts. VHDL simulations at this voltage have been projected to the system timing chart of Figure 3. We have included results with Vdd of 5.0V and 2.2V along with the baseline of 3.3 Volts to examine the MRC operating range as well as to permit closer comparisons to other low-power FFT chips.

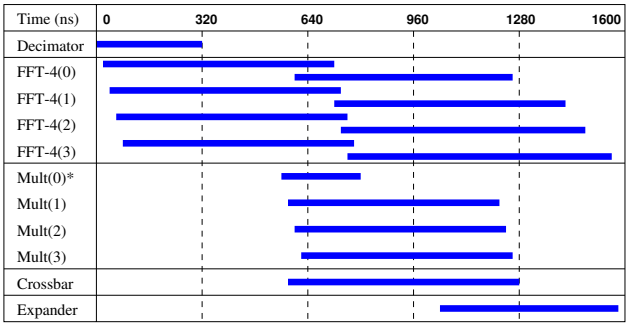

Figure 3: FFT-16 System Timing @ Vdd=3.3V

Notice how the duty cycle of all the major components overlap. The actual amount of overlap (pipelining) will vary depending on the data but this figure gives a good timing estimate. The asterisk by the Mult(0) line in Figure 3 is there to indicate that this is not an actual multiplier. The constants for the 0^{th} sequence are all equal to $1 + j0$ so no multiplication is required. In place of a multiplier, a holding register is used so a full 32-bit complex value is sent to the crossbar switch. Based on empirical SPICE data, we are able to extrapolate system and component timing to Vdd = 5.0 Volts and Vdd = 2.2 Volts. Table 3 shows the timing comparison between the three Vdd levels.

Table 3: Latency by Element (ns)

Vdd	Dec	FFT-4	Mult	C-bar	Exp	FFT-16
5.0	240	480	480	510	390	1200
3.3	320	640	640	680	520	1600
2.2	531	1062	1062	1129	863	2656

Using the actual SPICE power numbers for each component running at the frequencies in Table 3, we are

able to compile projected power consumption for the FFT-16. The component and system power consumption numbers are given in Table 4.

Table 4: Power Consumption by Element (mW)

Vdd	Dec	FFT-4	Mult	C-bar	Exp	FFT-16
5.0	5.8	182	350	114	74	1076
3.3	1.1	45	86	25	18	264
2.2	0.6	11	22	9	5	67

Then, using the scaleability properties of each segment, energy consumption for larger point sizes results can be determined as in Table 5.

Table 5: Extrapolation of Energy Consumption (μJ)

Vdd	FFT-16	FFT-32	FFT-128	FFT-256	FFT-1024
5.0 V	1.29	3.29	26.2	68.0	494.9
3.3 V	0.422	1.07	8.4	21.6	153.25
2.2 V	0.178	0.455	3.7	9.9	77.5

Since these numbers are very rough estimates, direct comparisons against current FFT chips are not that conclusive. These comparisons are still drawn to show that, despite using a power hungry cell library and disregarding many known power reduction techniques, similar power efficiency numbers can be achieved with architecture and asynchronous design techniques.

Table 6: Energy Efficiency Comparison

Chip	Vdd (V)	Power (mW)	Time (μs)	Energy/unit-transform (nJ)
This Work	5.0	2578	192	483
Plessey PDSP16510A	5.0	3000	98	287
This Work	3.3	599	256	149
Spiffee1	3.3	845	30	24.7
This Work	2.2	182	425	75
Spiffee1	2.5	339	42	13.9

Table 6 gives a rough comparison to the SPIFFEE project at Stanford University and a commercial FFT processor (The Plessey PDSP 16510A). It is fair to point out that the Plessey DSP chip uses a block-float data format instead of fixed point which accounts for some of the additional energy required. The figure-of-merit we are using, that of energy consumed per unit transform, compares the energy efficiency of the architecture in generating a result and is independent of the frequency. We must also point out that the sample frequency of our asynchronous design is considerable faster than that of any of these processors. The numbers for this project will remain fairly constant for larger point sizes due to the hierarchical nature of our FFT algorithm. However, as the point-size grows, additional hierarchical layers are required which will result in increased power consumption.

The 2.2 Volt Vdd entry of "This Work" in Table 6 probably could not be used in space because of the single event effects discussed in Section 2.2. It is presented here to show how the efficiency FOM scales between the different Vdd levels.

6 Conclusion

This paper shows a formal mathematical approach that has been directed towards architecting low power FFT circuits. The architecture relies on localization techniques, pipelining, and frequency shifts of decimation and expansion. Asynchronous implementation techniques are a particularly appealing low power implementation methodology for such an architecture due to the ability to shift between various frequencies at no additional cost in power or complexity. The asynchronous pipeline supports both rapid computation and minimal energy dissipation during idle periods.

It is prudent to say that the methodology we are using is effective for this project. Seldomly will the best design arise from the first specification. Typically a design will go from specification to implementation before many important design considerations become evident. Our methodology based on VHDL, as both a specification language and simulator, was very helpful in directing our top level design because handshake protocol inconsistencies are detected very easily due to the way VHDL performs value resolution. However, the VHDL is a bit unwieldy at times because it does not recognize dynamic values on a bus, as well as a few other minor problems. The hand layout was fairly easy because the MRC library cells fit together in a gate-array format. However, accurate circuit simulations become difficult as the design increases requiring modular approaches to circuit analysis.

The numbers attained from the extracted SPICE simulations show surprisingly low power for the device sizes required by the rad-tolerant library. We surmise that a significant effect is due to the synergy between asynchronous design and the architecture. Our preliminary results also give significant motivation for future design exploration and test chips, particularly for radiation tolerant applications. Should this project be extended to an implementation that is not radiation tolerant, we could expect an additional reduction power consumption beyond what is presented here, achieved through smaller transistor widths, pre-charge logic, and other circuit level optimizations.

Acknowledgments

We would like to thank several people for making this project possible. First of all, our sponsor, Charles Brothers who is the Chief of the Applied Technologies Branch of the Space Technology Directorate, Phillips Laboratory, Kirtland Air Force Base, New Mexico. Ad-

ditionally, we would like to thank further personnel at the Air Force Institute of Technology, namely David Gallagher, Jeff Butler, and Sam SanGregory. Each has contributed direct support on this project or provided the groundwork for it to begin.

References

[1] Joseph Azarewicz and Sheldon Jurist. Short course slides. *Introduction to Single Event Effects.* Physitron, Inc. Huntsville, AL, May 1995.

[2] Joseph Azarewicz and Sheldon Jurist. Short course slides. *Ionizing Radiation Effects in Electronics.* Physitron, Inc. Huntsville, AL, May 1995.

[3] Charles P. Brothers Jr. *Rapid and Accurate Timing Simulation of Radiation-Hardened Digital Microelectronics Using VHDL.* PhD thesis, Air Force Institute of Technology (AU), March 1994.

[4] Charles P. Brothers Jr., Joseph R. Chaves, David R. Alexander, and David G. Mavis. Radiation Hardening Techniques for Commercially Produced Microelectronics for Space Guidance and Control Applications. In *20th Annual American Astronautical Society Guidance and Control Conference*, February 1997.

[5] William S. Coates, Al L. Davis, and Kenneth S. Stevens. Automatic Synthesis of Fast Compact Self-Timed Control Circuits. In *IFIP Working Conference on Design Methodologies*, pages 193–208, April 1993.

[6] Lee A. Hollaar. Direct Implementation of Asynchronous Control Units. *IEEE Transactions on Computers Vol. C-31*, No. 12, pp. 1133-41, December 1982.

[7] George C, Messenger and Milton S. Ash. *The effects of Radiation on Electronic Systems.* New York, NY: Van Nostrand Reinhold, 1992.

[8] Bruce W. Suter and Kenneth S. Stevens. Low Power, High Performance FFT Design. In A. Sydow, editor, *Proceedings of IMACS World Congress on Scientific Computation, Modeling, and Applied Mathematics*, number 1, pages 99–104, June 1997.

[9] Bruce W. Suter. *Multirate and Wavelet Signal Processing.* San Diego, CA: Academic Press, 1997.

[10] Kenneth Yi Yun. *Synthesis of Asynchronous Controllers for Heterogeneous Systems.* PhD thesis, Stanford University, August 1994.

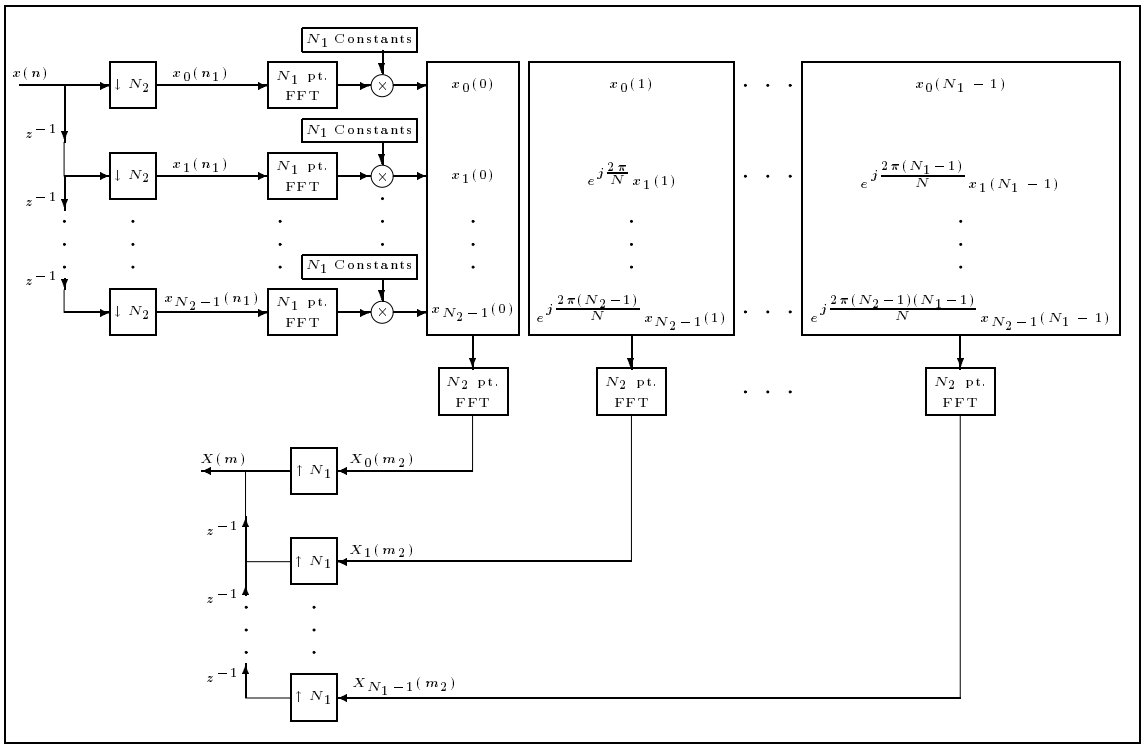

Figure 4: General Block Diagram of the Suter/Stevens FFT Algorithm