Abstract
This paper presents preliminary results of a methodology for the mathematical formulation and hardware implementation of computing methods for the digital processing of finite discrete signals in a signal algebra setting. An algorithm development and implementation environment is described in this work as a central theme in studying DSP computing methods. This environment is an aggregate of the following items: a PC Workstation platform, MATLAB® tools, digital signal processing (DSP) microprocessor units, and field programmable gate array (FPGA) units. Special emphasis is given to the concepts of modularity and scalability during a hardware implementation. A main goal of this ongoing work is to establish formal links amongst the elements of the environment in order to assist in reducing the algorithm development and implementation time-line. The preliminary results presented here center on the formulation of a methodology for computing methods as an operator theoretic approach to the digital processing of signals and on the study of the computing hardware structure and overall architecture of floating point DSP microprocessor units for the implementation of complex fast Fourier transform (FFT) cores.

1. Introduction
This work deals with the formulation of computing methods for the action of operators on discrete, finite length signals. These operators are termed here finite dimensional signal algebra operators and they play a very important role in many scientific and engineering applications such as multimedia signal processing, digital communications, remote sensing imaging, time series and econometric modeling, channel error correcting codes, biomedical signal processing, and seismic signal processing.

We give emphasis to modular and scalable computing methods for Communications Signal Processing applications and their implementation using digital signal processing (DSP) computing units and field programmable gate array (FPGA) structures. These methods deal with the algorithmic treatment of finite duration signals in order to extract information important to a user or software/hardware agent. The modular and scalable approach to these computing methods implies that the functions and structures of the algorithmic treatment should adapt to changes in the scales of an associated target system and the size or dimensionality of the signals to be processed. Our algorithmic treatment concentrates on understanding fundamental principles involved in Communications Signal Processing to observe, quantify, represent, transform, qualify, and render information carrying signals in our sensory reality. We define here an algorithms as a “well-defined” procedure to solve a problem in a finite number of steps. A problem is anything which requires a solution. Communications Signal Processing is defined here as an area dealing with analysis, design, and implementation (using a rapid prototyping approach) of circuits, signals, and systems for the transmission and reception of communications signals. A communication signal is defined as a signal appearing in any of the stages of an arbitrary communications system. We concentrate on systems designed for the digital communications and computing of information. The diagram in Figure 1 depicts a basic communications system.

Figure 1: Basic Communication System.

2. Signal Algebra Operators on $l^2(Z_N)$
We define a finite dimensional signal algebra as any linear algebra where its elements can be represented as coordinate vectors in a finite dimensional Hilbert space, a linear vector space over the scalar body or field of complex numbers $\mathbb{C}$. 
The signals become points in the space spanned by a finite set of discrete signals. All the signal algebra operators used in this work are defined as actions on the space \( l^2(Z_N) \), \( Z_N = \{0, 1, 2, \ldots, N - 1\} \) isomorphic to \( \mathbb{C}^N \). We concentrate on the use of the cyclic convolution operation as a binary operation that turns the space \( l^2(Z_N) \) into an algebra. The cyclic convolution operation is also viewed as a signal algebra operator acting on elements in \( l^2(Z_N) \), conditioned on an arbitrary but fixed signal as a parameter. The space \( l^2(Z_N) \) will be used to represent two basic related ideas. On the one hand, \( l^2(Z_N) \) will be thought of as representing the space of all complex discrete signals \( x \in l^2(Z_N) \) with domain \( Z_N \). On the other hand, it will also be thought of as representing the space of all periodic complex sequences with fundamental period of length \( N \). This latter interpretation allows us to perform modulo \( N \) operations on the indexing set \( Z_N \), turning this set into a group of order \( N \). It also allows us to identify correspondences between \( Z_N \) and Cartesian product sets of the form \( Z_R \times Z_S = \{(m, n) : m \in Z_R, n \in Z_S\} \), for \( N = RS \). These correspondences will become very useful when studying group theoretic properties of input/output indexing sets during signal transform operations of the form \( x \xleftarrow{A_N} \hat{x} \in l^2(Z_N) \).

We will act on elements of \( l^2(Z_N) \) with special operators such as the ubiquitous discrete Fourier transform (DFT) \( F_N \subset l^2(Z_N) \times l^2(Z_N) \).

Other operators are the cyclic shift operator, the cyclic reflection operator, the character modulation operator, and the cyclic correlation operator. We intend to implement these operators on DSP computing units. We will use \( l^2(Z_N) \) to treat signals with computing methods presented as compositions of these type of operators. For example, Figure 2 below shows a signal being treated by a linear combination of powers of the cyclic shift operator, presented as a new linear operator called “low-pass” finite impulse response (FIR) operator. Observe the “high frequency” attenuation effects exhibited by the output signal without having to resort to spectral analysis. This composed operator was implemented in MATLAB. One objective of this work is to translate these type implementations developed in MATLAB into firmware implementations using DSP computing units with associated field programmable gate array units, in some cases. In our signal algebra approach, we use MATLAB (it stands for MATrix LABoratory) to develop algorithms expressed as compositions of matrices representing operators. These algorithms are then mapped to hardware computing units using a defined modular, scalable methodology.

The simplest FIR system or operator, apart from the trivial system, i.e., the system represented by the identity operator \( I_N \), is the system \( T_h = T_{\delta_{t_1}} \), which turns into the shift operator \( S_N \). This system is sometimes called the unit delay system because its digital electronics hardware implementation may be accomplished by using a single delay element, which acting on the unit sample signal \( \delta_{\{t_1\}} \) results in

\[
T_{\delta_{\{t_1\}}} = \sum_{j \in Z_N} \delta_{\{t_1\}}(j)S_{N}^j = S_{N}^1 = S_{N}.
\]

The matrix representing the shift operator \( S_N \) is obtained by allowing the vector representation (with respect to the standard basis set \( \{\delta_{\{k\}} : k \in Z_N\} \)) of the signal \( T_{\delta_{\{t_1\}}} \{\delta_{\{t_1\}}\}, k \in Z_N \), become the columns of the matrix. In this manner, we obtain the following general expression:

\[
S_N = \left[ T_{\delta_{\{0\}}} \{\delta_{\{0\}}\}, T_{\delta_{\{1\}}} \{\delta_{\{1\}}\}, \ldots, T_{\delta_{\{N-1\}}} \{\delta_{\{N-1\}}\} \right] = \begin{bmatrix}
\delta_{\{0\}}, \delta_{\{1\}}, \ldots, \delta_{\{N-1\}}, \delta_{\{0\}}
\end{bmatrix}, \quad (1)
\]

where we have separated by commas the columns of \( S_N \) for legibility. The matrix then becomes:

\[
S_N = \begin{bmatrix}
0 & 0 & \ldots & 0 & 1 \\
1 & 0 & \ldots & 0 & 0 \\
0 & 1 & \ldots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \ldots & 1 & 0
\end{bmatrix} \quad (2)
\]

An important property of the \( S_N \) operator matrix is that any cyclic FIR system \( T_h \) may be represented by a matrix \( H_N \) which can be written as a sum of powers of the matrix \( S_N \) pre-multiplied by a diagonal matrix \( D_{h_{\{j\}}} \) as

\[
H_N = \sum_{j \in Z_N} D_{h_{\{j\}}} S_N^j = \sum_{j \in Z_N} \left(h_{\{j\}} \otimes S_N^j\right).
\]
3. Research Results

In this section we describe the research results obtained so far using the algorithm development and implementation environment depicted in Figure 3 for the study of computing methods for signal algebra operators. Our results concentrate on the computation of the DFT operator through the implementation of complex, radix-2, FFT algorithms. We begin by providing a description of the harmonic properties of the DFT operator and proceed to provide benchmarks of FFT core measurements.

Since $F_N$ is an operator defined on a finite dimensional vector space, it has a matrix representation. We shall denote by $\{W_N\}_{k=0}^{N-1}$ the matrix for $F_N$ with respect to the basis set $\{\delta_k\}_{k=0}^{N-1}$. Using the character signal $\chi_k = F_N(\delta_k) = \sum_{l=0}^{N-1} W_N \delta_k$, we get, for a signal $f$, $f = \frac{1}{N} \sum_{k=0}^{N-1} \hat{f}[k] \chi_k$. Hence,

$$f = \frac{1}{N} \sum_{k=0}^{N-1} \hat{f}[k] \chi_k^* = \frac{1}{N} \sum_{k=0}^{N-1} \hat{f}[k] (F_N(\delta_k))^*$$

$$f = \frac{1}{N} \sum_{k=0}^{N-1} \hat{f}[k] \sum_{l=0}^{N-1} W_N \delta_l = \frac{1}{N} \sum_{k=0}^{N-1} \hat{f}[k] \sum_{l=0}^{N-1} (w_N)^l \delta_l$$

$$f = \sum_{k=0}^{N-1} \left\{ \frac{1}{N} \sum_{l=0}^{N-1} (w_N)^l \hat{f}[k] \right\} \delta_l$$

$$f = \sum_{k=0}^{N-1} f[k] i[k]$$

Thus, the expression within curly braces represents

$$f[k] = \frac{1}{N} \sum_{l=0}^{N-1} (w_N)^l \hat{f}[k] = \frac{1}{N} \sum_{l=0}^{N-1} \chi_k[l] \hat{f}[k]$$

$$\text{and } \hat{f}[k] = \sum_{l=0}^{N-1} \chi_k[l] f[l] = \sum_{l=0}^{N-1} (w_N^l) f[l].$$

The $N$-point sequence $x[k]$ can also be expressed as

$$x[k] = F_N(\delta_k) = \sum_{j=0}^{N-1} \left\{ F_N(\delta_j) \right\} \delta_j$$

Comparing this expression (29) with the expression given by (25), results in the following equality $w_N = \left\{ F_N(\delta_k) \right\}$. It is important to observe here that we have been using the same symbol to denote both, an $N$-point sequence, say $x[k]$, as well as its vector coordinate representation. For instance, the vector $X_k$ is the vector $X_k = [x[k], x[N-k], x[N-2k], \ldots, x[N-1]]^T$.

Figure 3: Algorithm Development and Implementation Environment.

3.1 Implementation of FFT Cores

We worked with the C6000 DSK board, that contains a TMS320C6711 Floating Point Processor with the following specifications: 150 MHz clock rate, 32 bits data bus, 6ns instruction cycle time, 4 Kbytes L1 data cache, 4KByte L1 program cache, 64KByte L2 data cache, four ALUs floating and fixed-point, two ALU’s fixed point, two multipliers floating and fixed-point. Texas Instruments provides routines written in assembly language for applications in digital signal processing, such as filters FIR, Complex FFT, convolution, and others associated with the digital computing of signals to extract information.

We implemented a program written in C language that integrates the routine "cfft2.asm" to compute complex
FFT radix-2, and the routine "bitrev.fasm" to execute the bit-reversal of an array of data, in order to determine the existing data flow between the external memory, L2 internal memory, and general purpose registers of the CPU using Composer Studio tools. Figure 4 above shows the organization of the external memory and commits for handling of variables and constants used by the routines.

Initially the C program is compiled, including files of variables that serve as arguments for each routine in assembly language. After linking, the program is loaded to the DSK board; the variables and constants are stored in the area of memory denominated "cinit" used during boot of the processor. At the start of the program the data are placed in the L2 internal memory of the processor before executing a routine, later each routine uses the data and the constants stored in L2 and leave stored the results in L2, each routine has indications for timing. This count of cycles initiates with the access of variables and constants stored in L2 and finishes when the results are completely available in L2. The arguments to each routine must be available in the internal memory of the processor so that number of cycles taken in computing of the data is in agreement with formulas associated to each routine. The memory area can be distributed by command linker file and allow that the arguments to each routine are available in IRAM area (L2 Internal Memory). At the end of routine the results are stored in L2 memory and additional cycles are required to place data in external memory (SDRAM) on DSK, therefore, before benchmarking the data must be available in L2 memory. Table 1 shows the results obtained in each measurement with respect to number of points of complex FFT radix-2.

<table>
<thead>
<tr>
<th>Length of FFT</th>
<th>Bit Rev.</th>
<th>Bit Rev. (Cycles)</th>
<th>Complex FFT (Radix2) (Cycles)</th>
<th>Complex FFT (μ sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>62</td>
<td>0.414</td>
<td>182</td>
<td>1.09</td>
</tr>
<tr>
<td>16</td>
<td>92</td>
<td>0.552</td>
<td>291</td>
<td>1.74</td>
</tr>
<tr>
<td>32</td>
<td>158</td>
<td>0.948</td>
<td>523</td>
<td>3.13</td>
</tr>
<tr>
<td>64</td>
<td>280</td>
<td>1.68</td>
<td>1019</td>
<td>6.11</td>
</tr>
<tr>
<td>128</td>
<td>524</td>
<td>3.14</td>
<td>2106</td>
<td>12.63</td>
</tr>
<tr>
<td>256</td>
<td>1012</td>
<td>6.07</td>
<td>4512</td>
<td>27.07</td>
</tr>
<tr>
<td>512</td>
<td>2020</td>
<td>12.12</td>
<td>11290</td>
<td>67.74</td>
</tr>
<tr>
<td>1024</td>
<td>4943</td>
<td>29.65</td>
<td>32762</td>
<td>196.5</td>
</tr>
</tbody>
</table>

It is important to emphasize that the bit reversal routines are clocked apart from other FFT computing stages.

**Conclusions**

In this work we presented some preliminary results of a methodology for the formulation of computing methods for digital signal processing, based on operator theoretic methods in a discrete signal algebra setting. Implementation of operators using an algorithm development and implementation environment was conducted using C language on DSP microprocessor units. Our C programs allow the integration of routines written in assembly that are available for the purpose of benchmarking. We also developed C-programs to manipulate the input data and the creation of data arrays before the compilation. We have started to identify new FFT variants, for non-power-of-two lengths, to adapt to the specific computing structure of the TI 6711 DSP unit.

Finally, we used the basic foundation tools provided by XILINX, Inc, in order to learn about basic concepts, and design methodologies using code written in VHDL. An elementary application, such as a simple 4-bit counter, was implemented including functional simulation, timing simulation, and synthesis before design implementation on FPGA structure. This was a preview step before mapping and implementation of signal algebra operators using a defined modular, scalable methodology. The internal structure of a given FPGA is shown in Figure 5.

![Figure 5: Basic representation of an FPGA.](image)

**References**
