RECONFIGURABLE FPGA COMPUTATIONAL STRUCTURES FOR SIGNAL PROCESSING

Yvonne Avilés yvonne@ece.uprm.edu

Domingo Rodríguez domingo@ece.uprm.edu

Computational Signal Processing Group
Department of Electrical and Computer Engineering
University of Puerto Rico, Mayagüez Campus
Mayagüez, PR 00681-9042

ABSTRACT

This work concentrates on the implementation of Kronecker-based signal processing algorithms as hardware computational structures in FPGA technology; and presents a methodology to establish the correspondence. This methodology is also presented as an alternative technique for modular design of computational structures for signal processing algorithms as well as an efficient framework for rapid prototyping. Its application is described in terms of fast Fourier transform (FFT) algorithms; but basic principles can be abstracted and generalized.

1. INTRODUCTION

Field programmable gate arrays (FPGA) present an alternative technology to traditional DSP processors or custom circuitry, combining flexibility and performance of these conventional methods. FPGA architectures feature an embedded parallel structure of Configurable Logic Blocks (CLBs) performing as function generators, connected by an interconnect network and Input/Output Blocks (IOB) and can be reconfigured within the system. This organization provides an internal environment for rapid prototyping, well suited for implementation of scalable computational structures of signal processing algorithms.

In our conception of computational structure design, modularity and scalability play an important role. Signal processing algorithms, analyzed as stages of computation in a hardware transformation environment, provide a basis for modular design of hardware constructs. Kronecker product algebra is used as a tool in this transformation environment due to its properties, which lead to the analysis of signal processing algorithms as mathematical formulations of Kronecker products. These mathematical formulations are implemented as stages arranged as compositions of primitive structures known as functional primitives, and become building blocks for hardware computational structures. These formulations may also be expressed as distinct organizations of computational kernels, addressing then a range of computational structure implementations. By this means, implementing signal processing algorithms as FPGA computational structures centers on identifying a correspondence between the functional primitives in a mathematical composition of a given signal-processing algorithm and suitable hardware constructs in FPGA. This correspondence addresses reconfigurable properties of FPGA resulting in rapid prototyping operations

2. KRONECKER PRODUCTS AND COMPUTATIONAL STRUCTURES

Kronecker product algebra, a branch in finite multilinear algebra, is presented in this context as an organizational language for designing computational structures for signal processing algorithms implementation. Some important properties of this algebra allow rapid prototyping of computational structures based on selection and organization of the basic building blocks within an algorithm. By this means suitable implementations are based on performance assessment.

2.1 Definition: Kronecker product of two matrices

Let any two arbitrary matrices **A** and **B** be of order $N \times N$ and $M \times M$, respectively. The Kronecker product of the two matrices is a new matrix, denoted by $(A \otimes B)$, of order $NM \times NM$, given by:

$$C = A \otimes B = [a_{N,N}]B$$

$$A \otimes B = \begin{bmatrix} a_{0,0}B & a_{0,1}B & \cdots & a_{0,N-1}B \\ a_{1,0}B & a_{1,1}B & \cdots & a_{1,N-1}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{N-1,0}B & a_{N-1,1}B & \cdots & a_{N-1,N-1}B \end{bmatrix}$$

Matrices of this type are called *Kronecker matrices*. A $NM \times NM$ Kronecker matrix $A \otimes B$ can be decomposed as follows:

$$A \otimes B = (AI_M) \otimes (I_N B) = (A \otimes I_N)(I_M \otimes B)$$

where I_M is the M - dimensional identity matrix.

The factor $(A \otimes I_N)$ or *vector Kronecker factor* represents a vector operation. On the other hand, the factor $(I_M \otimes B)$ is a parallel operation. Accordingly, the computation of a Kronecker operation on an input vector **x** represented by $y=(A \otimes B)x$ can be executed in two stages: One performing a vector operation and another performing a parallel operation.

2.2 Kronecker–based DSP Algorithms

Applying the definition and properties of Kronecker products algebra to matrix representations of DSP algorithms generates a new representation for the algorithm. The novel aspect of this transformation underlies in the properties of Kronecker algebra, which serve as an organizational agent to change the structure of the Kronecker formulation by means of permutation matrices. These structures, identified as computational structures, allow new forms of implementation for signal processing algorithms as hardware constructs.

2.3.1 Kronecker decomposition of the FFT

The general Kronecker decomposition of the FFT algorithm using the Cooley-Tukey method for decimation in time, for a Fourier matrix F_N of order $N = R \cdot S$, was presented in [3] and is shown below.

$$F_N \mathbf{x} = (F_S \otimes I_R) T_{N,S} (I_S \otimes F_R) P_{N,S} \mathbf{x}$$

Using the referenced information as a fist step in describing our methodology, we present a Kronecker formulation of a four- point Cooley-Tukey FFT decimation in time algorithm.

$$F_4 \mathbf{x} = (F_2 \otimes I_2) T_{4,2} (I_2 \otimes F_2) P_{4,2} \mathbf{x}$$

Each factor in the above composition represents a computational kernel of the FFT computational structure, which will be directly associate to a hardware kernel in a hardware computational construct. The hardware construct implementation will then be a direct correspondence of the software construct.

3. FPGA COMPUTATIONAL STRUCTURES

As stated above, signal processing algorithms, efficiently modeled by means of Kronecker products formulations, may be directly implemented as hardware constructs, establishing a direct correspondence between hardware and software structures. Using the application example discussed earlier a computational structure for an N-point FFT may be represented as the structure in Figure 1.

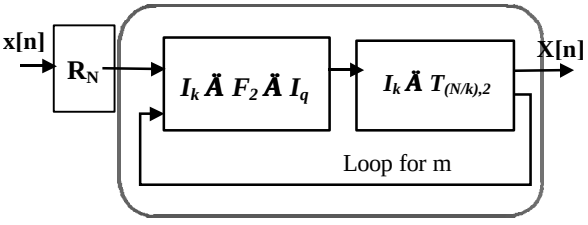

Figure 1: F_N Genral Computational Structure

The input x[n] passes through two main modules. The first module performs a bit reversal permutation on the elements of the input x[n]; the second module performs an internal loop of Kronecker products. The number of iterations or stages in the loop depends on the order of the FFT, where m = stages = $\log_2 N$ for a radix-2 FFT and $N = 2^m$. The implementation of the general structure for an 8-point FFT may be represented as the structure in Figure 2.

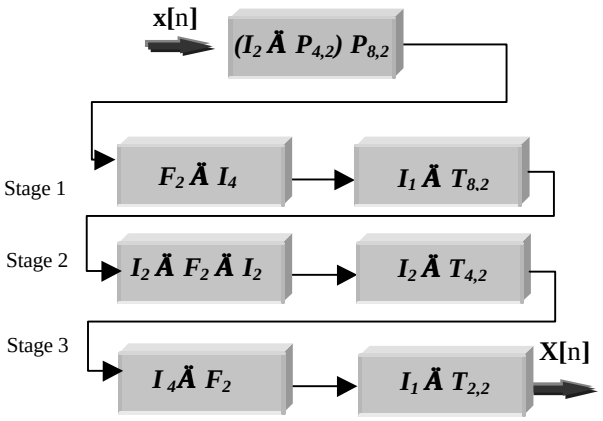

Figure 2: F8 Computational Structure

The block elements within the computational structure represent the functional primitives.

3.1 Design Tools

Hardware constructs of functional primitives are built using the VHDL hardware description language targeted to Xilinx FPGA XC4000 devices and the Foundation Series Software.

3.2 Design Methodology

The design methodology for the Kronecker-SPA environment is defined as a procedure of the *Hardware Computational Structure Generator Environment*, discussed below.

HCS Generator Environment

- ***Modeling and simulation of Kronecker-based DSP functional primitives through hardware description language.***

The VHDL entity declaration for one of the functional primitive modules is shown below.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity ikronp_ram is
  generic(BusWidth:Integer:=16;
          VecSize: Integer:= 16;
          M:Integer:=4; N:Integer:=4;
          Stride: Integer:=2);

  port (d: std_logic_vector (BusWidth-1 downto 0);
        load:              in STD_LOGIC;
        rst_load:          in STD_LOGIC;
        clk:               in STD_LOGIC;
        ce:                in STD_LOGIC;
        permuted_data: out std_logic_vector
                           (BusWidth-1 downto 0));

end ikronp_ram;
```

- ***Implementation of the basic DSP functional primitives as hardware constructs.***

  Functional primitives, as the two-point FFT(F(2), stride permutations, twiddle factor matrix operator, and other primitive functions are implemented as hardware constructs by means of hardware description language and Xilinx FPGA implementation tools. Verification of these implementations is obtained through the Xilinx Logical and Timing Simulation Tool.

- ***Functional Primitive Library (FPL)***

  The hardware constructs implemented in the previous stage are stored as a library of functional primitives.

- ***FPGA Implementation of Computational Structures using the Xilinx Implementation Tools***

Synthesis of the VHDL programs produce a *netlist* file as input to the Xilinx implementation environment, which is then transformed into a bit stream to be downloaded into an FPGA device.

The design layout may be viewed and edited through an implementation editor as in Figure 3.

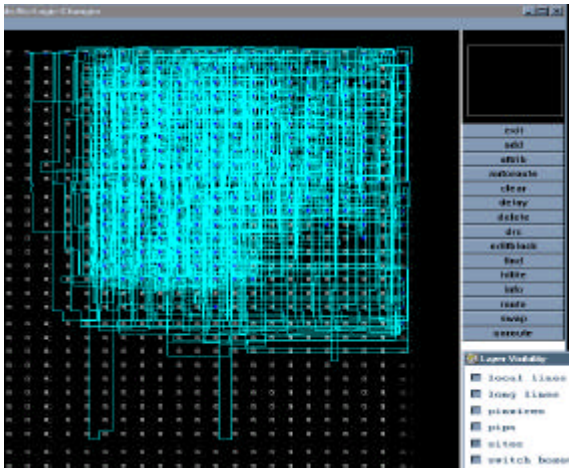

Figure 3: Design Implementation Layout (Xilinx)

- ***Performance Assessment***

Reports on area and speed values are indicators of performance and are a comparison tool between performance of the various structure implementations of an algorithm. Results are reported through simulation tools as shown on Figure 4.

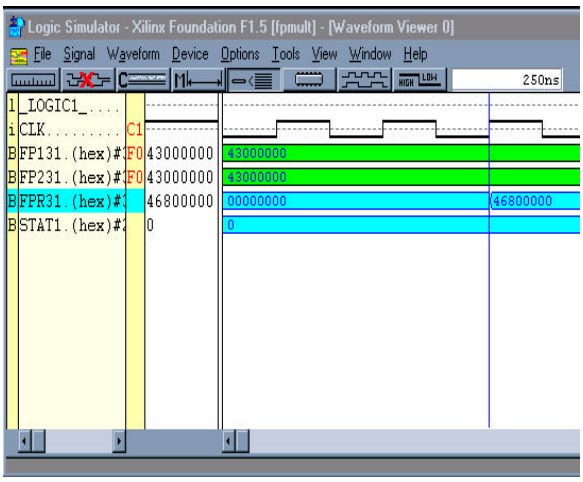

Figure 4: Simulation and Verification Tool

4. FUTURE WORK

The first stage of our environment is designed for FPGA computational structure implementation of the FFT. Our future work consists on updating the functional primitives library (FPL) for general signal processing algorithms, i.e. Cyclic Convolution, FIR filters, etc., and suit the environment for general methods for these algorithm implementations.

Our future interest also resides in designing a methodology for implementing hardware computational structures using VHDL recursive components. This requires changes in the structure of the standard VHDL hardware description language, which are actually being studied, to provide designers with various approaches for synthesis execution.

5. CONCLUSIONS

We have presented an environment with an identified methodology for the mapping of Kronecker–based signal processing algorithms to FPGA computational structures and have demonstrated that Kronecker products represent an alternative method for modular design of computational structures for signal processing algorithms. We have also shown that FPGAs reconfigurable properties and internal block structure provide an efficient framework for rapid prototyping of modular design.

6. REFERENCES

[1] D. L. Dai, S. K. S. Gupta, S. D. Kaushik, J. H. Lu, R. V. Singh, C.H. Huang, P. Sadayappan, R. W. Johnson “EXTENT: A Portable Programming Environment for Designing and Implementing High-Performance Block Recursive Algorithms”,

[2] J. R. Johnson, R. W. Johnson, D. Rodríguez, R. Tolimieri, “A Methodology for Designing, Modifying, and Implementing Fourier Transform Algorithms on Various Architectures”, IEEE *Transactions on Circuits, Systems, and Signal Processing*, Vol. 9, No. 4, 1990.

[3] D. Rodríguez, "Kronecker Product Algebra as a Tool for VLSI Implementation of the Discrete Fourier Transform", *Proceedings ICASSP 91*, Vol. 2, May 1991, pp.1025-1028.