A PORTABLE HARDWARE DESIGN OF A FFT ALGORITHM

C. GONZÁLEZ-CONCEJERO, V. RODELLAR, A. ÁLVAREZ-MARQUINA, E. MARTÍNEZ DE ICAYA and P. GOMEZ-VILDA †

† Departamento de Arquitectura y Tecnología de Sistemas Informáticos.
Grupo de investigación en informática aplicada al procesamiento de señal e imagen.
Facultad de Informática - Universidad Politécnica de Madrid
Campus de Montegancedo s/n – Boadilla del Monte.
28630 (Madrid) – SPAIN
coral@junipera.datsi.fi.upm.es

Abstract— In this paper, we propose a portable hardware design that implements a Fast Fourier Transform oriented to its reusability as a core. The design has parameterized the number of samples and the number of the data’s bits. The module has been developed using a radix-2 decimation in time algorithm of n-point samples. Structural modelling is implemented using VHDL to describe, simulate, and perform the design. The resulting design is portable among different EDA tools and technology independent. The system has been synthesized with Quartus II from Altera and the performance results are presented.

Keywords— FFT, VHDL, reusability, portable, EDA tools, Altera.

I. INTRODUCTION

Nowadays semiconductor technology is able to create very complex devices that can enclose a complete system in a single chip (SoC). If the system is created from scratch, achieving the desired performance is costly and time consuming. To meet the tight time-to-market requirement, the electronic design uses pre-designed intellectual property (IP) cores as a common practice. These cores may be parametrizable and customizable to be synthesized in a large application specification. They are available to the designer from heterogeneous sources, design team, CAD tool libraries, CAD tool independent libraries, etc.

One of the areas that major demands of application specific circuits design is digital signal processing (DSP). Fast Fourier Transform is a computationally intensive DSP function, widely used in many applications.

Since the pioneering work by Cooley and Tukey (Cooley and Tukey, 1965), a lot of work has been done on the FFT algorithm such as the radix-2⁰⁰ algorithm and the split radix algorithm (Brigham, 1998; Wosngrad, 1976; Duhamel and Hallmann, 1984). Among them, the radix-2 and radix-4 algorithms have been used mostly for practical applications due to their simple structures.

Most of the implementations and benchmarking of FFT algorithms has been done using general purpose processors, DSP processors and dedicated FFT. In this paper, we present a radix-2 Fast Fourier Transform architecture design, under the point of view of its reusability to be embedded in different applications. The design has been modelled in VHDL according to the restrictions and recommendations for high level synthesis (Keating and Bricand, 2002). The resulting design is portable among different EDA tools and technology independent.

II. FAST FOURIER TRANSFORM

The N-point Discrete Fourier Transform (DFT) of a finite duration sequence x(n) is defined as follows.

\[ X(k) = \sum_{n=0}^{N-1} x(n)W^{nk} \quad k = 0,1,\ldots, N-1 \quad (1) \]

where \( W = e^{-j(2\pi/N)} \) is referred as the twiddle factor, N is the transform size and \( j = \sqrt{-1} \).

The FFT is an efficient algorithm to compute the DFT and its inverse (Cooley and Tukey, 1965; Brigham, 1998). It generally falls into two classes: Decimation In Time (DIT), and Decimation In Frequency (DIF). The DIT algorithm first rearranges the input elements in bit-reversed order and then builds the output transform. The DIF algorithm first transforms and then rearranges the output values. The basic idea of these algorithms is to break up an N–point DFT transform into successive smaller and smaller transform known as a butterfly (basic computational element). The smallest transform used is a 2-point DFT known as radix-2, it processes groups of 2 samples. The combination of two stages of radix-2 in one stage constitute radix-4 algorithms, it processes groups of 4 samples. There are also other decomposition schemes known as split-radix algorithms (Duhamel and Hallmann, 1984).

The calculations implied in the basic computational element (butterfly) for radix-2 in DIT algorithm will be introduced next. A and B are two complex numbers represented as:

\[ A = x + jX \quad (2) \]
\[ B = y + jY \quad (3) \]

where x and y are real parts and X and Y are imaginary of them. “A transform (A)” and “B transform (B)” are calculated as shown the next equations: