# 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-$2^m$ algorithm and the split radix algorithm (Brigham, 1998; Winograd, 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.

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

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:

$$A = x + jX \qquad (2)$$
$$B = y + jY \qquad (3)$$

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:

$$A' = x' + jX' = A + B\,W_N^k \quad (4)$$
$$B' = y' + jY' = A - B\,W_N^k \quad (5)$$
$$W_N^k = \cos(2\pi k/N) - j\sin(2\pi k/N) \quad (6)$$

Taking into consideration (2), (3) and (6), the transforms may be written as:

$$A' = [(x + y\cos(2\pi k/N) + Y\sin(2\pi k/N)) + j(X + Y\cos(2\pi k/N) - y\sin(2\pi k/N))] \quad (7)$$
$$B' = [(x - y\cos(2\pi k/N) - Y\sin(2\pi k/N)) + j(X - Y\cos(2\pi k/N) + y\sin(2\pi k/N))] \quad (8)$$

where k depends of the number of stages and the number of samples.

## III. VHDL MODELING

The objective of this paper is to implement in an efficient manner the equations (7) and (8) having in mind the reusability of the resulting design as embedded core in a possible wide range of applications. Then the number of samples N and the number of bits to coding each sample must be considered as generic parameters in order to adapt the size to the specific application and to control the quantization errors when using fixed point arithmetic. Also the degradation performance should be considered with the increment of the number of samples N. All components of the hardware used in this paper have been modelled in VHDL according to the restrictions and recommendations for high level behavioural synthesis (Keating and Bricand, 2002).

### A. General Description

The FFT-N core interface structure is shown in Fig. 1 and its associated pin functionality is described in Table I. This core is able to compute the direct or inverse FFT.

The input data to the algorithm is a vector of N complex values with *b* bits for the real part and *b* bits for the imaginary part representing numbers in two's complement. In our work we have considered b = 16, as default value.

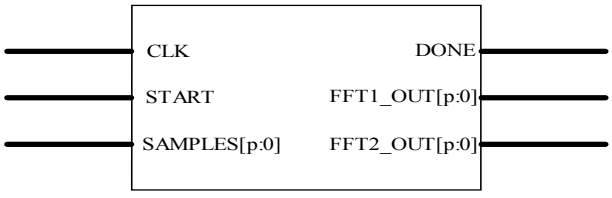


**Fig. 1.** FFT-N core interface.

**Table I.** I/O pins functionality description.

| *Signal Name* | *Dir* | *Description* |
|---|---|---|
| CLK | Input | Master clock (active rising edge) |
| START | Input | Start processing (active high). This signal must be synchronized with clk. |
| SAMPLES[p:0] | Input | Real component of the input data. The imaginary component is initially stored in RAM. |
| DONE | Output | FFT finished (active high). |
| FFT1_OUT[p:0] | Output | Real component of the FFT result. The values are in two's complement format. |
| FFT2_OUT[p:0] | Output | Imaginary component of the FFT result. The values are in two's complement format. |

The complete operation of the FFT processor is partitioned into three main processes. These are the DATA load, COMPUTE and RESULT unload. The processing cycle starts with the DATA load process, when sampled data is read in and stored in a RAM memory. During the COMPUTE process the FFT is computed on the stored data. And finally the RESULT unload process make the FFT results available at its output, ready to be used by another application. A block diagram of the general structure is shown in Fig. 2. It has four basic blocks:

An internal double port RAM memory to hold the values of the input samples, intermediate operations and results, a butterfly unit consisting of radix-2 butterflies and two ROM memories to generate the twiddle factors, the address unit to provide the synchronized addresses to extract the data from the RAM memory and twiddle factors, and finally a control unit.

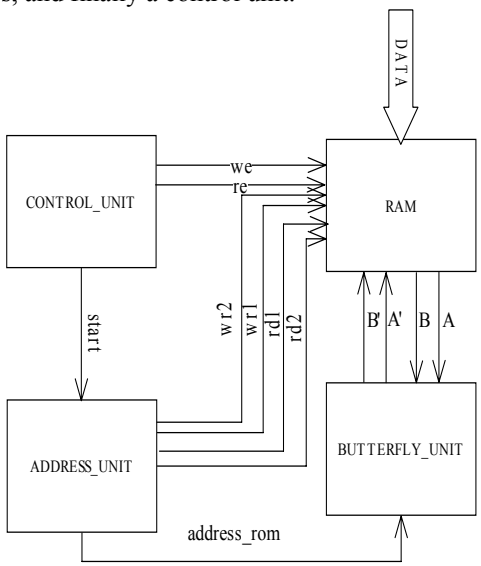


**Fig. 2.** Blocks general structure

### B. RAM

This auxiliary internal memory has different types of information as the computation of the algorithm progresses. Initially holds the complete input data vector. Later on the result of a butterfly process between two samples overwrites the input data positions. And finally during the OUTPUT process, bit reversed address is given to the RAM and it reorders the final results in it accordingly.

The real and imaginary part of a data share the same memory addresses, so in the default case the memory size will be N x 32 bits. The dual port capability has the advantage of doing available two data samples at the same time which is very convenient to calculate the radix-2 algorithm.

### C. Butterfly Unit

A butterfly unit block consisting of (N/2) butterflies with the basic structure shown in Figure 3. Each one of them containing: two (N/2) x 16-bits ROMs to store the sine and cosine of the twiddled factors, four 16 x 16 multipliers in two's complement, six 32-bits accumula-

tors and two special operators to adequate the data format. The arithmetic operations involved in this block are shown in Table II and are performed according to a pipeline data flow structure. First the cosine and sine from the ROMs and the data inputs are read (R). Then, the read elements must be multiplied (x). Next the four partial products are added or subtracted and truncated (±) to 16 bits default data format. The truncated real and imaginary parts are concatenated (&) to assembly the 32 bits data format. And finally these results are written to in the RAM (W). The scheduled operations to calculate a butterfly demands six instants of time.

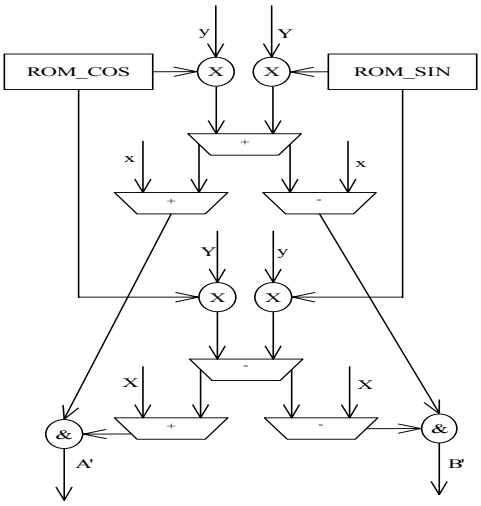


**Fig. 3.** Butterfly structure.

**Table II.** Operations scheduling in the butterfly block.

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| R (Read data) | cos | | | | | |
| | sin | | | | | |
| | A | | | | | |
| | B | | | | | |
| x (Partial products) | | m1=ycos | | | | |
| | | m2=Ysin | | | | |
| | | m3=ysin | | | | |
| | | m4=Ycos | | | | |
| ± (Terms grouping and truncation) | | | s1=m1+m2 | | | |
| | | | s2=m4–m3 | | | |
| | | | | s3=x+s1 | | |
| | | | | s4=x–s1 | | |
| | | | | s5=X+s2 | | |
| | | | | s6=X–s2 | | |
| & (Format Recovery) | | | | | A'=s3&s5 | |
| | | | | | B'=s4&s6 | |
| W (Write results in RAM) | | | | | | A' |
| | | | | | | B' |

### D. Address and control unit

The purpose of the address unit is to provide the RAM and the ROMs with the correct addresses to access to its contents. It also keeps track of which butterfly is being computed in each stage. For an N-point complex FFT there are s stages, where $s = \log_2 N$, each stage consists of (N/2) butterflies with the structure described before.

The correct address depends on the mode of operation (input, output or computation). In the input and output process the address goes from 0 to N-1. But in the FFT computation process it goes from 0 to (N/2) -1 and carries on this process s times, one per stage. As mentioned, the result of FFT computation is written back into the same location as it was read. However, there is a latency of one clock cycle. For example, if "B" is read from the RAM during cycle "3", "B transform" is written into the same location after one cycle, which is during cycle "4". So the read address is delayed in this cycle.

The block diagram of this unit is shown in Fig. 4. It includes the following sub-blocks: a base index, delay units and MUX's to manage RAM memory, a romadd_unit to generate the ROM memory addresses and an address_count block to manage the stages account. The address_count block provides the base_index and romadd_unit blocks with the stage in which FTT is being computed. During the input and output process the base index block goes from 0 to N-1 and this value is transfer to a MUX_add to generate reading address rd1, rd2 and to a delay unit to generate writing address wr1, wr2. During the FFT computation process the base index block goes from 0 to (N/2)-1 and carries on this process *"s"* times, one per stage. The romadd_unit blocks provides the ROM with the correct address, for collecting the sine and cosine coefficients, according to the stage in which FFT is found.

The control unit that supervises all the operations of the hardware is implemented as a finite state machine having twelve states.

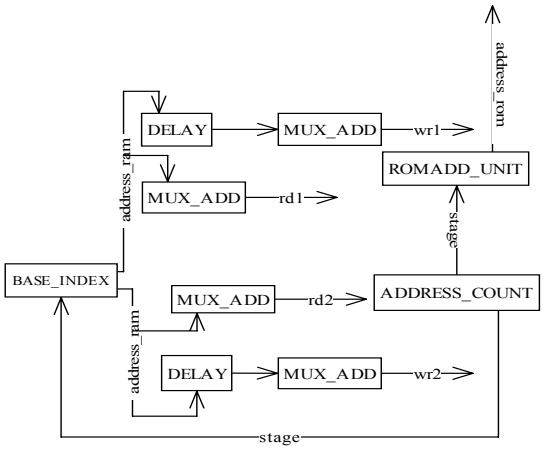


**Fig.4.** Address unit scheme

## IV. RESULTS

The VHDL code has been done following the recommendations of the methodology for reuse (Keating and Bricand, 2002). No any pre-designed component available from the libraries of any EDA vendor have been used, which makes possible the portability among different synthesis tools and the technology independence. The system has been simulated and synthesized with the EDA tools Quartus II from Altera for evaluation performance purposes. The selected device was the

EP2S60F1020C4 from Stratix II family. In Table III the max frequency of operation and the demanded resources in terms of Logic Elements and DSP are shown for the complete system synthesis and for each block of the general structure shown in Figure 2. The results presented are for N = 8, 64 and 256 samples. As is obvious, the RAM's size is proportionally increased according to the number of samples N. The sizes of Butterfly and Address unit blocks augment very slightly as the number of samples N increments. This characteristic is very much appropriated for reusability purposes. The control unit has the same size for any number of samples. The total conversion time of the 8-point FFT is 39 clock cycles. The clock cycles depend on the chip used; in this paper, the clock cycle of the chip is 93.86 MHz, given a 0.42 μs conversion time. The max frequency of conversion is obviously degrading as the number of samples increases but not drastically. This operation frequency could be notably improved when using library elements of target technology.

**Table III.** Synthesis results for the complete system and partial blocks.

| | LE | DSP | LE | DSP | LE | DSP |
|---|---|---|---|---|---|---|
| | 8-point | | 64-point | | 256-point | |
| RAM | 477 | 0 | 3651 | 0 | 14540 | 0 |
| Butterfly | 320 | 32 | 335 | 32 | 444 | 32 |
| Address unit | 92 | 0 | 263 | 0 | 873 | 0 |
| Control unit | 17 | 0 | 17 | 0 | 17 | 0 |
| Total | 906 | 32 | 4266 | 32 | 15874 | 32 |
| F.Max (MHz) | 93.86 MHz | | 90.88 MHz | | 86.61 MHz | |

## V. CONCLUSIONS

In this paper, we have presented a preliminary design of a portable and technological independent hardware design that implements a FFT oriented to its reusability as a core for DSP applications. Fast Fourier Transform module has been developed using radix-2 decimation in time algorithm of N-point samples. We have borne Hardware Description Languages restrictions for synthesis in mind. The design has parameterized the number of samples and the number of the data's bits. The values of the twiddle factors to be located in ROMs must be obtained off-line, and then the reusability of this structure will need some changes in its values. The performance of the design, excluding the size of the RAM memory, presents a slightly degradation as the number samples increments which is very adequate for reusability purposes.

## ACKNOWLEDGMENT

This research is being carried out under the following projects of Programa Nacional de las Tecnologias de la Información y de las Comunicaciones from the Ministry of Science and Technology of Spain: TIC2002-02273 and TIC2003-08756.

## REFERENCES

Cooley, J.W., and J.W. Tukey, "An algorithm for the machine calculation of the complex Fourier series", *Math. of Computation*, **19**, 297-301 (1965).

Brigham, E.O., *The Fast Fourier Transform and its Applications*. Prentice Hall (1998).

Winograd S., "On computing the discrete Fourier transform", *Proc. Nat. Acad. Sci*. U.S, **73**, 1005-1006, (1976).

Duhamel, P., and H. Hallmann, "Split Radix FFT algorithm", *IEEE Electronic Letters*, **20**, 14-16, (1984).

Keating, M. and P. Bricand, *Reuse Methodology Manual: For System-on-a-Chip Designs*. Third Edition. Kluwer Academic Publishers (2002).

**Received: April 14, 2006.**
**Accepted: September 8, 2006.**
**Recommended by Special Issue Editors Hilda Larrondo, Gustavo Sutter.**