Hardware Implementation of Decimation in Time FFT

Akshata. U
PG Scholar, Department of ECE,
MVJ College of Engineering,
Bangalore, Karnataka, INDIA
Email: akshata.arikady@gmail.com

Gopika. D.K
PG Scholar, Department of ECE
MVJ College of Engineering
Bangalore, Karnataka, INDIA
Email: gopika.dk@gmail.com

Hameem Shanavas. I
Assistant Professor, Department of ECE
MVJ College of Engineering
Bangalore, Karnataka, INDIA
Email: hameemshan@gmail.com

Abstract

The Fast Fourier transform is an indispensable operation in many digital signal processing applications but yet is deemed computationally expensive when performed on a conventional general purpose processors. This paper explains the implementation of radix-22 single-path delay feedback pipelined DIT-FFT processor. This architecture has the same multiplicative complexity as radix-4 algorithm, but retains the simple butterfly structure of radix-2 algorithm. The implementation was made on a Field Programmable Gate Array (FPGA) because it can achieve higher computing speed than digital signal processors, and also can achieve cost effectively ASIC-like performance with lower development time, and risks.

Keywords: dit-fft, fpgas, radix-22 single-path.

I. INTRODUCTION

Fast Fourier Transform (FFT) has become almost ubiquitous and most important in high speed signal processing. Using this transform, signals can be moved to the frequency domain where filtering and correlation can be performed with fewer operations. It has been widely applied in the analysis and implementation of digital communication systems and television terrestrial broadcasting systems, such as the xDSL (de)modulator, phase correlation system, mobile receiver, as well as fault characterization and classification[1][2]. When considering the alternate implementations, the FFT/IFFT algorithm should be chosen to consider the execution speed, hardware complexity, and flexibility and precision. Nevertheless, for real time systems the execution speed is the main concern [3]. Several architectures have been proposed over the last 3 decades like: single memory architecture, dual-memory architecture, cached memory architecture, array architecture, and pipelined architecture [4]. Pipelined architectures characterized by real time, non-stopping processing and present smaller latency with low power consumption which makes them suitable for most application. The pipelined architectures can be classified into two types: single-path architectures and multi-path architectures. Several single-path architectures have been proposed: Radix-2 single path delay feedback, Radix-4 single-path delay feedback, Radix-2² single-path delay feedback, Radix-2⁴ single-path delay feedback, Split-Radix single-path delay feedback, and Radix-4 single-path delay commutator. The multi-path architectures: Radix-2 multi-path delay commutator, Radix-4 multi-path delay commutator, Split-Radix multi-path delay commutator, and Mixed-Radix multi-path delay commutator. The observation made on the listed architectures reveals that the delay feedback architecture is more efficient than the corresponding delay commutator in terms of memory utilization and Radix-2² has simpler butterfly and higher multiplier utilization [5]. This makes Radix-2² single path delay feedback an attractive architecture for implementation. Classical implementation of the FFT/IFFT algorithm, with digital signal processors (DSPs), requires a sequential algorithm. This slows down the execution time. On the other hand, the modern programmable circuits, like an FPGA, utilizes a tens of thousands of lists and triggers during operation, resulting of parallel processing system, putting the FPGA computing speed at a significant advantage over DSPs [6], [7], [8]. This paper presents the implementation of Radix-2² single path delay feedback pipelined DIT FFT/IFFT processor on an FPGA.

II. RADIX-2² FFT ALGORITHM

The Discrete Fourier Transform (DFT) X(k), k = 0, 1, …, N − 1 of a sequence x(n), n = 0, 1, …, N − 1 is defined as:

\[ X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \]  

(1)
Where \( N \) is the transform size, \( W_N^{nk} = e^{-j2\pi nk/N} = \) and \( j = \sqrt{-1} \). According to the decomposition method of [9] that done

\[
\begin{align*}
  n &= \left( \frac{N}{2} n_1 + \frac{N}{4} n_2 + n_3 \right)_N \\
  k &= (k_1 + 2k_2 + 4k_3)_N \\
  X(k_1 + 2k_2 + 4k_3) &= \sum_{n_3=0}^{N/4-1} \left[ H(k_1, k_2, n_3), W_N^{n_3(k_1+2k_2)} \right] W_N^{n_3k_3}_{N/4} \\
  H(k_1, k_2, n_3) &= \left[ x(n_3) + (-1)^{k_2} x \left( n_3 + \frac{N}{2} \right) \right] + (-j)^{(k_1+2k_2)} \left[ x \left( n_3 + \frac{N}{4} \right) \right. \\
  &\left. \left. + (-1)^{k_1} x \left( n_3 + \frac{3N}{4} \right) \right] \right)
\end{align*}
\] (2)

(3)

After this simplification, we have a set of four DFTs of length \( N/4 \). Each term in equation (2) and (3) represents a Radix-2 butterfly (BFI), and the whole equation also represents Radix-2 butterfly (BFII) with trivial multiplication by \(-j\). Figure 1 shows an example of \( N = 16\)-points Radix-2 decimation in Time (DIT)—the method used for the pipelined, streaming I/O architecture—FFT algorithm. It is to be noted that the inputs are in permuted (digit-reversed) order whereas the outputs are in normal order. The pentagons between BFI and BFII represent the trivial multiplication by \(-j\). After these two butterflies, full twiddle factor multipliers (TFM) are required to compute the multiplication by the twiddle factor \( W_N^{n_3(k_1+k_2)} \). Figure 2 shows the block diagram for Radix-2 \( 2^2 \) \( N \)-point FFT processor. The \( N \)-point FFT processor has \( \log_2(N) \)-stages with \( i \) is the stage number. A typical stage consists of BFI, BFII and TFM. A \( \log_2(N) \) counter is used to control the processor. The structure of the last stage is different according to the size of FFT; if \( N \) is power of 2, the last stage is composed of BFI only. But if \( N \) is power of 4, the last stage composed of BFI and BFII.

**Figure 1:** Flow Graph for Radix \( 2^2 \) DIT for \( N = 16 \) FFT Algorithm

**Figure 2:** Block Diagram for Radix-2 \( 2^2 \) \( N \)-points FFT Processor
III. FPGA IMPLEMENTATION OF RADIX-2^{2}

To implement in FPGA, the selection of target FPGA should consider the required resources for the pipelined architecture to be implemented (for N = 256): complex multipliers, complex adders/subtractors for BFI and BFII, registers and memory for delay feedback and pipelining, ROM for storing the twiddle factor, and the control unit. The complete DIT can be implemented in a Xilinx Spartan3 XC3S200 series FPGA device, because it can accommodate 200,000 gates, 18 × 18 hardware multiplier, 18k bit RAM, and maximum data rate of 622 Mbps.

3.1 ButterfFly I (BFI) Structure

The detailed structure of BFI is shown in Figure 3. The A input comes from the previous component, TFM. The B output fed to the next component, normally BFII. In first N/2i + 1 cycles, multiplexors direct the input data to the feedback registers until they are filled (position “0”). On next N/2i + 1 cycles, the multiplexors select the output of the adders/subtractors (position “1”), the butterfly computes a 2-point DFT with incoming data and the data stored in the feedback registers.

3.2 ButterfFly II (BFII) Structure

The detailed structure of BFII is shown in Figure 4. The B input comes from the previous component, BFI. The Z output fed to the next component, normally TFM. In first N/2i + 2 cycles, multiplexors direct the input data to the feedback registers until they are filled (position “0”). In next N/2i + 2 cycles, the multiplexors select the output of the adders/subtractors (position “1”), the butterfly computes a 2-point DFT with incoming data and the data stored in the feedback registers. The multiplication by −j involves real-imaginary swapping and sign inversion. The real-imaginary swapping is handled by the multiplexors MUXim, and the sign inversion is handled by switching the adding-subtracting operations by mean of MUXsg. When there is a need for multiplication by −j, all multiplexors switches to position “1”, the real-imaginary data are swapped and the adding-subtracting operations are switched.

3.3 Twiddle Factor Multiplier Structure

A six clock cycle fully-pipelined complex-multiplier has been implemented to multiply the twiddle factor by the output of BFII. According to Equation (4), the algorithm of multiplying the twiddle factor (c + js) by BFII output (Z_r + jZ_i) uses four multipliers, one adder, and one subtractor. The structure of TFM shown in Figure 5.

\[(Z_r + jZ_i)(c + js) = (Z_r c - Z_i s) + j(Z_r c - Z_i s)\]  (4)

Twiddle factor generator is a key component in IFFT/FFT computation. There exist many popular generation techniques for twiddle factor: COordinate Rotation DIgital Computer (CORDIC) algorithm, pipelined-CORDIC algorithm, polynomial-based approach, ROM-based scheme, and the recursive function generators. For small lengths such as 64 up to 512, ROM-based is a better choice[10]. The twiddle factors are generated using MATLAB according to equation (5), converted to fixed point, and then stored in ROM.

The twiddle factor at the ith-stage, with i = 0, 1, ..., (log_a N) – 2 is given by

\[v = \begin{cases} 
0, & 0 \leq x < a \\
2^{i+1}(x - a), & a \leq x < 2a \\
2^i(x - 2a), & 2a \leq x < 3a \\
3 \cdot 2^i(x - 3a), & 3a \leq x < 4a 
\end{cases}\]  (5)
IV. RESULTS

The FFT processor was described with hardware description language VHDL as fixed-point arithmetic has to be synthesized with XST tool in Xilinx ISE version 10.1 on Xilinx xc3s200 FPGA chip, the high-performance signal processing applications chip with advanced serial connectivity, and simulated using ModelSim Xilinx Edition (MXE III). The $X_n$ is input data (16-bit real and 16-bit imaginary), $X_k$ output data (16-bit real and 16-bit imaginary), synchronous reset; $rst$, clock, chip enable, start, busy, and finish.

V. CONCLUSION AND FUTURE WORK

The design and implementation of Radix$2^2$ single-path delay feedback pipelined DIT FFT processor on an FPGA has been presented. The description was made to implement in Xilinx ISE on Spartan 3 family and the functionality was verified by ModelSim. The outputs from the described architecture has to be validated against the standard FFT in Matlab. Future work includes the design to increase the number of points of FFT as well as IFFT by imposing simple modification.

REFERENCES


