Design and Implementation of Pipelined Floating Point Fast Fourier Transform Processor

Veera Kamat Dhakankar
Department of Electronics & Telecommunications Engineering
Goa College of Engineering, Farmagudi, Goa, India

Prof. Sonia Kuvelkar
Department of Electronics & Telecommunications Engineering
Goa College of Engineering, Farmagudi, Goa, India

Abstract

There are several methodologies and techniques that already offer hardware and software solutions for computing Fast Fourier Transform (FFT), which have advantages for specific applications. These solutions are developed for running in several platforms, such as GPU, DSP, FPGA and ASIC and they are usually described in C/C++ language or HDL. Implementation intended for reconfigurable logic is usually described in HDL, such as VHDL or Verilog. Different FFT algorithms have been proposed to exploit certain signal properties to improve the trade-off between computation time and hardware requirements. For years the common practice for implementing FFTs has been to run them on a digital signal processor (DSP). But as FPGAs have evolved and have begun to accommodate function specific computing cores, FPGAs are beginning to displace DSP as the optimum FFT solution. In this project the implementation of FFT radix 2 algorithm design of floating point FFT in FPGA is discussed. The VHDL Design was tested using Modelsim measure its performance in term of speed and scalability and also the processor is synthesized on an FPGA to measure the performance parameters such as maximum operating frequency, area and power consumption. Synthesis tool is Xilinx ISE Design Suit 10.1 and FPGA is Spartan III.

Keywords: FFT processor, Pipelined processor

I. INTRODUCTION

FFT is efficient algorithm to calculate the DFT. FFT algorithms use the symmetry and periodicity properties of the DFT recursively divide the calculation. Radix is the size of FFT decomposition. Twiddle factors are the coefficients used to combine results from a previous stage to form inputs to the next stage. The number of steps depends on the number of inputs into the algorithm. In 1965 when Cooley and Tukey re-developed the first well known version in a Mathematical Computing paper [1].

Algorithm Computation of DFT/FFT of a time domain digital signal x(n) results in converting it into a frequency domain signal. Analysis and processing of a discrete signal in frequency domain is more efficient than its analysis in time domain. The FFT algorithm was first developed and presented by Cooley and Tukey in [1]. It was developed in order to reduce number of complex multiplications additions in DFT. An N-point DFT is given by eq.(1),

\[ X(k) = \sum_{n=0}^{N-1} x(n) e^{-j(\frac{2\pi nk}{N})} \]

where \( k = 0,1,2..N-1 \) (1)

According to eq.(1) DFT computations require \( N^2 \)-N complex additions and N2 complex multiplications. An N point FFT equation is given by eq(2).

\[ X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n) e^{-j(\frac{2\pi nk}{N})} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)e^{-j(\frac{2\pi nk}{N})} \]

where \( W_N^k = e^{-j2\pi k/N}, k = 0,1,2..(N-1) \) (2)

According to eq.(2) the number of multiplications and additions are reduced to (N/2)log2(N) and (N)log2(N) respectively. Preferring FFT over DFT for hardware implementation means increased speed, reduced power consumption, reduced area and reduced cost.

In DIT algorithm, inputs are in bit reversed order and the outputs are in natural order. Whereas in DIF algorithm, the inputs are in natural order and the outputs are in bit reversed order. The DIT algorithm provides better signal-to-noise ratio when compared to DIF algorithm for a finite word length. Based on size of FFT decimations, the algorithm can be radix-2, radix-4, radix-8 or split-radix type. In radix-2 algorithm FFT size is a power of two, radix-4 FFT size is a power four while radix-8 FFT size is power of eight. And split-radix type involves mix of any of the specified radix combinations. In radix-2 DIT FFT algorithm can be depicted as a butterfly diagram as shown in figure 1. Considering N-point FFT, there are log2N no of stages and each stage requires N/2 butterfly operations. The butterfly operation is pictorially described as shown in figure 2.

Butterfly operation can be given in equation as: \( X = P + WQ \) and \( Y = P - WQ \) Where, P, Q : Complex inputs and W : Complex input Twiddle Factor X , Y : Complex output values.

All rights reserved by www.ijirst.org 512
II. LITERATURE REVIEW

FFT processors can be divided into three main classes, Simple FFT has simple structure, but requires large memory and power consumption is high. Pipelined FFTs utilize concurrent processing of different stages to achieve high throughput. In fully parallel FFTs the implementation is hardware intensive and is not practical for large FFTs. Used for real time operations. Different types of architectures proposed to design FFT processors. Each architecture has its own limitations and benefits and are proposed by keeping in mind its potential application where it is going to be used. Single Memory Architecture is the simplest FFT system, in which the FFT processor reads from a memory and writes back to the same location. The memory is N times the data size capacity and the entire data is written once for every Log2N stages of the FFT. This kind of design has the minimum requirement of hardware resulting in high computation time. Some FFT processors use two memories, one for input and the other for output. Thus while data processing is carried out using one memory, the processed data fed into the other memory. In ping pong memory design, the first stage data is read from the memory one and after computation the data is written in memory two. At the next stage the data is read from memory two and written into memory one.

Ref [1] : J. Cooley and Tukey presented the FFT algorithm in their paper in 1965, which gives the basic derivation. In particular research is focused on FFT algorithm and finds efficient hardware solutions based on Specific wireless standards [6], Execution speed [9], Area [7], Power efficiency, Hardware complexity [10], Flexibility and precision. Ref [6], [7], [9] : all presented FFT processor with two butterfly units and a two port block RAM i.e. using pipelined architecture Ref [9] : proposed a reconfigurable FFT with a butterfly unit having three multipliers and carry look ahead adder to increase the speed but had to compensate with increase in area consumption,[10] used a generalized conflict free memory addressing scheme for continuous addressing for parallel memory based architecture but for limited number of inputs. Majority of the work focus on a specific standard in order to meet various standards a scalable FFT is a need of an Hour. To design balanced a scalable FFT processor which can be scaled according to the application and meets the requirement of multiple wireless standards. An attempt is made to find reasonable balance between low area, low cost, high speed flexibility and scalability.

The IEEE Standard Format for Floating Point Data : IEEE -754 floating point standard is used. The target device selected is Spartan III FPGA used in VDO PCI board. Spartan III PCI Board (MXS3FA-PCI-FG900).

III. DESIGN OF FFT PIPELINED PROCESSOR

By considering all the architectures proposed by many people, we have proposed a novel architecture which combines the advantages of dual memory architecture and pipelined architecture. The data flow of the pipelined dual memory The data flow of the pipelined dual memory architecture is shown in figure 3. The block diagram of one stage of FFT processor is shown in figure 4.

There are three main processes controlling this design.
- Loading Twiddle Coefficients into the internal memory of FPGA.
- Loading Data in internal memory.
- Computation of FFT.

On the initial reset of the system the Twiddle coefficients need to be stored in the Twiddle Ram and simultaneously, the incoming data samples are stored in data memory M0. The FFT Processor begins computation simultaneously by accessing the
data from M1 which is initialized to Zeros. Once all the data has been stored in M0. FFT processor must have accessed all the location of M1. Now it shifts to M0 and accesses M0. At this time all odd stages begin computation with Memory M0 while all even stages begin computation with memory M1. It takes 10 clock cycles after reset for the first processed data to be available at the output of the first stage. The corresponding address of the data is also available at the output of the stage. Also stage done signal is asserted which is used by the next stage to synchronize the storing of input data which is nothing but the output from the previous stage.

Address Generator block generates the required addresses considering the butterfly structure for each stage for the memory to be accessed and also the address for the output sample where it is to be stored in the next stage’s memory. While the FFT computation is being carried out the new sets of data are being loaded in the alternate memories of the first stage and this process repeats till FFT of a set of data points are calculated.

The 1024 point FFT processor is shown in figure 5. The data samples are accepted in sequence and they are given to the Bit reversal module. The bit reversal module reverses the address and stores these samples in memory M0 and similarly in memory M1. These samples are then accessed by Stage1 in sequential manner. Once the first FFT output is ready at the FFT out the stage1 done signal is generated for synchronization with the next stage2. Similarly all stages generates corresponding Stage done signal to synchronize the full process.

Fig. 3: Dual Memory Pipelined Data

A. Design of Building Blocks of FFT:
The main component of the FFT is the Butterfly, which carries out floating point computations and the address generating unit. The detailed design of these units is described in subsequent sections.

B. Pipelined Floating Point Butterfly Unit:
The butterfly is the basic computational block with which FFTs are calculated. The butterfly is given by Equations (1) and (2). One butterfly operation requires four real multiplications and six real additions. The design of pipelined butterfly unit as shown in figure 6 has only two multipliers and two adder/subtractors as compared to four multiplier and six adders required for parallel operation. The Complex multiplication of two numbers (Re1+Img1)×(Re2+Img2) can be broken down into real and imaginary parts as follows,

Real Part = (Re1×Re2)-(Img1×Img2) and
Imaginary Part = (Re1×Img2)+(Img1×Re2)

Fig. 4: Single Stage of Pipelined FFT Butterfly design

The clockwise functioning of the pipelined complex butterfly can be explained as follows:-
1) Clock1: Twiddle real (real part of Twiddle factor) is selected by the multiplexer0 and Tw_img is selected by multiplexer1 and applied to the two Multipliers along with In2_real and In2_img respectively.

2) Clock2: Multipliers provide outputs for (In2_real×Tw_real) and (In2_img×Tw_img) and these outputs are stored in R1 and R2 registers. Also new set of values i.e. Tw_img and In2_real are given to multiplier1 and In2_img and Tw_real are given to multiplier2 through multiplexers.

3) Clock3: R1 and R2 are given to the subtractor since we need to subtract (In2_real×Tw_real) and (In2_img×Tw_img)

4) Clock4: output of the subtractor is stored in R3 also new set of inputs from R1 and R2 i.e. (In2_real×Tw_img) and (In2_img×Tw_real) are given to the adder.

5) Clock5: Result of the adder is stored in R4 and output of R3 is selected through multiplexer and given to the Adder to add with IN1 coming from delay pipe.

6) Clock6: IN2 real goes through a delay of 4 clock cycles and is given as input to the Adder along with output of R3.

7) Clock7: Result of this addition is stored in R5 also In2_img and output from R4 is applied again to the adder.

8) Clock8: Result of this addition is stored in R7. Also the other set of outputs are stored in R6 and R8 (the second set of inputs IN1 and IN2).

9) Clock9: The values in R5(Real) and R6(Img) are combined to form OUT1. Clock8: Result of this addition is stored in R7. Also the other set of outputs are stored in R6 and R8 (the second set of inputs IN1 and IN2).

10) Clock 10: The values in R7(Real) and R8(Img) are combined to form OUT2.

C. Pipelined Floating Point Multiplier Unit:
The multiplier has been designed as a three stage pipeline. The pipelined multiplier carries out one multiplication at every clock and has a latency of 3 clocks. The basic operation involves multiplying the mantissa of both the numbers and adding their exponent values. The Exception handler checks for the following exceptions, Infinity, NaN, Zero. Any number greater than 1.0×2^{128} is set as infinity and the overflow bit is set. Any multiplication by a NaN or infinity results in the output being set to infinity. Any multiplication by a zero results in a zero. And any number less than less than 1.0×2^{-127} results in the output being set to zero.

D. Pipelined Floating Point Adder Unit:
The adder designed is also a three stage pipeline and carries out one addition every clock with a latency of 3 clocks. The floating point addition operation can be explained in the following steps,

The exception handler checks for exceptions similar to those in floating point multiplier design.

E. Address Generator:
The address generator unit is one of the important blocks of FFT design. It has to be synchronized perfectly with the butterfly operation. For each butterfly operation, we have to read a pair of data inputs from the memory and after the required computation, the results have to be written back to the same memory locations. Each butterfly operation requires a particular twiddle factor for its computation, so we need to generate the addresses of input data as well as corresponding twiddle addresses. The address generating unit should be able to generate addresses for any FFT size in the radix 2 algorithm and generate different addresses for different stages. The address generator designed is for 1024 point FFT and generates all the required addresses of the Radix 2 algorithm. The IN2 address is generated first and IN1 address on the next clock. The Twiddle address is available for two clock periods.

F. Butterfly Operation:
For each butterfly operation we have to read a pair of data inputs from the memory and after the required computation, the results have to be written back to the same memory locations. Each butterfly operation requires a particular twiddle factor for its computation, so we need to generate the addresses of input data as well as corresponding twiddle addresses. The address generating unit should be able to generate addresses for any FFT size in the radix 2 algorithm and generate different addresses for different stages. The address generator designed is for 1024 point FFT and generates all the required addresses of the Radix 2
algorithm. The IN2 address is generated first and IN1 address on the next clock. The Twiddle address is available for the two clock periods.

![Single Stage Pipelined Butterfly][9]

Fig. 6: Single Stage Pipelined Butterfly[9]

IV. RESULT AND EVALUATION

A. Simulation:
The FFT processor was simulated before synthesizing it on an FPGA device. Simulation was carried out in order to verify the functionality of the processor and to validate the results.

B. Pre-Simulation:
Before simulating FFT processor necessary environment need to be set up. Simulation of FFT processor operation requires RAM and ROM inputs. RAM and ROM data are provided in text files as inputs to simulation. RAM stores data samples whereas ROM stores twiddle factors required for FFT computation.

For simulation, the tool used was ModelSim SE 5.7 from Mentor Graphics Inc. Load the design. After compilation, design file was created with the name of top level entity inside the work directory. Add ports and signals to wave window. To observe the state and values of various ports and signals, they were loaded onto the wave window.

Synthesis Power Analysis For synthesis Xilinx ISE Design Suit 10.1 is used.

C. Synthesis and Place and Route Timing Report:
The Single Precision (32 bit IEEE 754 format) FFT design was synthesized for various data sizes. The device selected was Xilinx Spartan III. The synthesis and timing report is given in Table.

<table>
<thead>
<tr>
<th>No. of Points</th>
<th>Min Period(ns)</th>
<th>Max Freq (MHz)</th>
<th>No of Slices</th>
<th>Slice FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>27</td>
<td>37.03</td>
<td>19930</td>
<td>19930</td>
</tr>
<tr>
<td>512</td>
<td>29</td>
<td>34.48</td>
<td>22452</td>
<td>22452</td>
</tr>
<tr>
<td>1024</td>
<td>29</td>
<td>34.48</td>
<td>25059</td>
<td>14458</td>
</tr>
</tbody>
</table>

V. CONCLUSION

A radix 1024 N-point novel FFT processor architecture based on a reasonable balance between performance, power, area parameters are proposed. The FFT processor was designed, implemented using VHDL, simulated using ModelSim and synthesized on Spartan III (FGPA device MXS3FA-PCI-FG900). The processor architecture can be adopted in number of devices where a reasonable balance between specific parameters is required.

REFERENCES


