ANALYSIS OF SYNTHESIS ISSUES ABOUT DESIGNING DSP DEVICES

Akash Verma¹, B.S. Rai²
M.Tech. (Pursuing), Dept. of Electronics Engineering, MMMEC, Gorakhpur, India¹
Head of Dept., Dept. of Electronics Engineering, MMMEC, Gorakhpur, India²

ABSTRACT: This paper discusses the issues related to the synthesizing the designs of DSP devices to FPGA. The high level codes used for synthesis input, are in VHDL. The central issues behind the designs are synthesizable or not are, used HDL libraries and data types. All the issues and solutions are illustrated using 32-Point Fast Fourier Transform. In the beginning, the IEEE fixed point package (fixed_pkg) is used for designing FFT-32 then whole logic is designed using single IEEE package (STD_LOGIC_1164) which is absolutely synthesizable to FPGA. For implementing DSP algorithms using ‘STD_LOGIC_1164’, ‘real type’ data structure is represented by array of bits, that is ‘bit_vector’. Algorithms for real type addition, subtraction and multiplication are developed using array of bits which will fulfill the function of complex and real arithmetic. DSP algorithms implemented through this design method are complete synthesizable and can be implemented with very high degree of precision.

Keywords: DSP, FFT, FPGA, Fixed_Pkg, Radix-2 algorithm, STD_LOGIC_1164, Synthesis, VHDL, Virtex-5.

I.INTRODUCTION

This paper proposes the issues and solutions of synthesis problems of DSP designs to FPGA [6]. DSP algorithms which are designed on VHDL do not guarantee that they are synthesizable [8] to FPGA. It may be possible that these HDL designs do compile and simulate properly on HDL simulators and compilers but still it is not sure that they are completely synthesizable to FPGA [5]. There are several issues which restricts these codes from synthesizing [7]. The FPGA vendors provide their software tools for synthesizing the HDL codes. For synthesizing the HDL design, the libraries and packages used must be supported by these tools [8]. Data type used for HDL designing of DSP algorithm is also an important factor for design synthesis [3]. In this paper, radix-2 based 32-point Fast Fourier Transform algorithm [4] is synthesized to Virtex-5 FPGA. VHDL design of FFT used for synthesis is developed by following two different ways:
1) Using IEEE fixed point package, ‘fixed_pkg’[2].
2) Using IEEE package, ‘STD_LOGIC_1164’ [5].

Data types of DSP algorithm is mostly of ‘signed real’ type [3]. VHDL provides various fixed point and floating point data types for representing real type data [1]. Since floating point data type is not efficient for synthesizing to FPGA so fixed point data type is used. For implementing DSP algorithm using only IEEE package, ‘STD_LOGIC_1164’, a method is developed for representing real data type by array of bits. All real arithmetic such as multiplication, addition and subtraction are implemented by specialized algorithms. These algorithms manipulate the bit_vector to implement the real arithmetic.
II. RADIX-2 ALGORITHM

It is one of the simplest Fast Fourier Transform algorithm in which a Butterfly structure is replicated to get higher order FFT.

![Butterfly structure](image1)

Fig. 1: Butterfly structure

In replicated butterfly structure we change the values of weighted coefficients ‘w’ as per the position of Butterfly. Radix-2 algorithm may be implemented by either decimation in time or decimation in frequency. In decimation in time algorithm we shuffle the order of input while in decimation in frequency algorithm we shuffle the output. Here decimation in time algorithm is used for implementing 32-Point FFT.

III. SIMULATION OF FFT DESIGN USING IEEE FIXED POINT PACKAGE, ‘FIXED_PKG’

IEEE accepted fixed point Package; ‘fixed_pkg’ in VHDL-2008 [1]. This package has powerful operators and functions that specially suits for designing DSP algorithms. 32-Point FFT, radix-2 algorithm is first designed on VHDL using this package then HDL design is compiled and simulated on ModelSim PE Student Edition 10.2a.

A. SIMULATION OF BUTTERFLY STRUCTURE

Here (ar, ai) and (br, bi) are two fixed point real inputs and (yr0 , yi0) and (yr1, yi1) are outputs of Butterfly structure. (wr, wi) is the weighted coefficient of Butterfly structure. Simulation result of Butterfly component is shown in the following figure.

![Simulation result of Butterfly](image2)

Fig. 2: Simulation result of Butterfly
B. SIMULATION OF 32-POINT FFT

In this 32-Point FFT design, above simulated Butterfly is used as component. Butterfly component is used in five stages and in each stage sixteen instances of Butterfly are used. Here \( x_r \) and \( x_i \) are real and imaginary parts of input \( x \). Similarly \( y_r, y_i \) and \( w_r, w_i \) are real and imaginary parts of \( y \) and \( w \) respectively. There are 32 complex inputs, 32 complex outputs and 17 different weighted coefficients. Simulation result of 32-Point FFT is shown in the following figure.

![Simulation result of 32-point FFT](image)

---

C. ISSUES ON SYNTHESIZING THIS DESIGN

Fixed point package, fixed_pkg is brought to VHDL-2008 by IEEE. It is said that the designs using this package will be synthesizable. All the data structures used in this package are fixed point.
This package contains powerful operators and functions which makes it very efficient in designing DSP algorithm. But, still this package is not absolute synthesizable because it is not fully supported by synthesizer tools.

IV. IMPLEMENTATION OF FFT USING IEEE PACKAGE, ‘STD_LOGIC_1164’ ALONE

For implementing the DSP algorithms using IEEE package, ‘STD_LOGIC_1164’ alone, a data type is required which may implement both the real and complex numbers. Its real data type is of floating point, so it will not be synthesizable. So a method is developed which represent the real and complex numbers by ‘bit_vector’.

A. REPRESENTATION OF REAL AND COMPLEX NUMBERS BY ‘BIT_VECTOR’

‘A’ is an array of bits of length ‘L’ in which M bits represents fraction part and N bits represents whole part.

\[ A = b_{(N+1)} \ldots b_2 b_1 b_0 \ldots b_{(M+1)} b_M \]

The numbers N and M are chosen depending on the precision and range of real number. Here important point is that the software and FPGA will treat this array of bits ‘A’ as simply a bit vector of length ‘M+N’.

B. ALGORITHMS FOR ARITHMETIC OPERATION ON THIS DATA STRUCTURE

Specialized algorithms are developed which will operate on above defined data structure and manipulate the array of bits in such a way that they will fulfill the functions of complex and real numbers. For the 32-Point FFT calculation, addition, subtraction and multiplication algorithms are used. Multiplier algorithm is explained in following context.

C. ALGORITHM FOR MULTIPLIER

This multiplier is designed for multiplication operation in FFT design. It takes two bit vectors as input and gives a bit vector as output with its length equal to multiplicand.

![Flow Chart for Multiplier Algorithm](image-url)
D. SYNTHESIS AND SIMULATION OF FFT USING IEEE PACKAGE, ‘STD_LOGIC_1164’

These results are compiled and simulated on Modelsim PE Student Edition 10.2a and synthesized on Xilinx ISE 10.1 design suite. For design simulation ‘Xilinx Virtex-5’ FPGA is used.

Table I: Target FPGA Properties

<table>
<thead>
<tr>
<th>Target FPGA Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family</td>
</tr>
<tr>
<td>Device</td>
</tr>
<tr>
<td>Package</td>
</tr>
<tr>
<td>Speed</td>
</tr>
</tbody>
</table>

1. SIMULATION OF BUTTERFLY STRUCTURE

Simulated result is shown in the following figure. As it was shown in figure-1, it has two complex inputs, two complex outputs and one complex weighted coefficient. Each input and output is of 23 bits length and coefficient is of 12 bits length. Least significant 10 bits are fraction bits.

Fig. 5: simulated result of Butterfly component

2. SYNTHESIS RESULTS OF BUTTERFLY STRUCTURE

The HDL design of Butterfly is synthesized with Xilinx ISE 10.1 design suite. RTL view of Butterfly structure is shown in the following figure. This RTL structure is automatically generated after synthesizing the design with Xilinx design suite. The next is synthesis design summary of Butterfly on Virtex-5 FPGA.

Fig. 6: RTL view of Butterfly component

Fig. 7: Synthesis design summary of Butterfly component
3. TIMING SUMMARY

Maximum combinational path delay of designed FFT-32 on FPGA is 95.814ns.

Table II: Time delay of Butterfly component

<table>
<thead>
<tr>
<th>Delay type</th>
<th>Delay (ns)</th>
<th>Delay (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic</td>
<td>15.522</td>
<td>16.0</td>
</tr>
<tr>
<td>Route</td>
<td>81.292</td>
<td>84.0</td>
</tr>
<tr>
<td>Total</td>
<td>95.814</td>
<td>100.0</td>
</tr>
</tbody>
</table>

4. SIMULATION OF 32-POINT FFT

In the 32-Point FFT design, above described Butterfly design is used as component. Total 80 instances of Butterfly are used in this design.
5. SYNTHESIS OF 32-POINT FFT

The proposed design of 32-FFT block is synthesized using the Xilinx ISE 10.1 design suite. Thus the RTL block obtained after synthesizing the design is shown below.

From the above synthesized Internal RTL architecture it clear that whole architecture is divided in five stages and each stage comprises sixteen instances of Butterfly, thus total eighty instances of Butterfly are visible in above shown RTL architecture.

The next is synthesis design summary of 32-Point FFT on Virtex-5 FPGA, it shows the features of the Virtex-5 FPGA used by Xilinx design suite for proposed work. Maximum combinational path delay: 466.732ns.
The HDL design of 32-Point FFT is implemented with IEEE fixed point package, ‘fixed_pkg’. This design works properly on Modelsim but is non-synthesizable. A new data structure is devised which is actually a bit vector that fulfills all the complex and real data structure needs. This data structure is completely defined by IEEE package, ‘STD_logic_1164’. For implementing DSP algorithms using this data structure some specialized arithmetic algorithms are designed. 32-Point FFT

VI. CONCLUSION

The HDL design of 32-Point FFT is implemented with IEEE fixed point package, ‘fixed_pkg’. This design works properly on Modelsim but is non-synthesizable. A new data structure is devised which is actually a bit vector that fulfills all the complex and real data structure needs. This data structure is completely defined by IEEE package, ‘STD_logic_1164’. For implementing DSP algorithms using this data structure some specialized arithmetic algorithms are designed. 32-Point FFT

For

Average Percentage Error-

*4404.59939 is outside the range (4095.99)

VI. CONCLUSION

The HDL design of 32-Point FFT is implemented with IEEE fixed point package, ‘fixed_pkg’. This design works properly on Modelsim but is non-synthesizable. A new data structure is devised which is actually a bit vector that fulfills all the complex and real data structure needs. This data structure is completely defined by IEEE package, ‘STD_logic_1164’. For implementing DSP algorithms using this data structure some specialized arithmetic algorithms are designed. 32-Point FFT
is again implemented with this data structure and this design is absolutely synthesizable. Simulated results of synthesizable 32-Point FFT design are compared with Matlab results, and we get average error of 0.1%. This shows that this method of designing and implementing DSP algorithms is very efficient and completely synthesizable.

REFERENCES