LUT optimization of Fast Fourier Transform

EE14B095

January 17, 2019

Abstract

Fast Fourier Transform (FFT) remains of a great importance due to its substantial role in the field of signal processing and imagery. In this report, multiple designs of 8 point FFT algorithm is proposed. The developed architecture was implemented using an FPGA . Though, the material resources of the FPGA are limited, particularly the integrated DSP blocks, different approaches are used during the Verilog description with the aim to reduce the necessary number of LUTs. The experimental validation was done using ISIM simulation tool, where the numerical synthesis and the post and route described in Verilog was realized using ISE Design Suite 14.7. The FFT modules of all the implementations were tested using a python script with various corner input test vectors.

1 Cooley–Tukey algorithm

1.1 Overview

Cooley–Tukey algorithm re-expresses the discrete Fourier transform (DFT) of an arbitrary composite size N = $N_1 N_2$ in terms of N_1 smaller DFTs of sizes N_2, recursively, to reduce the computation time to **O**(N log N) for highly composite N.

1.2 Calculation

This is a direct implementation of Cooley-Turkey Algorithm without any modifications , and for a 8 point FFT , it involves 2 multiplications for a single output calculation. The equations involving the calculation of FFT in shown below.

$t_1 = D(0) + D(4); m_3 = D(0) - D(4);$
$t_2 = D(6) + D(2); m_6 = j * (D(6) - D(2));$
$t_3 = D(1) + D(5); t_4 = D(1) - D(5);$
$t_5 = D(3) + D(7); t_6 = D(3) - D(7);$
$t_8 = t_5 + t_3; m_5 = j * (t_5 - t_3);$
$t_7 = t_1 + t_2; m_2 = t_1 - t_2;$

$m_0 = t_7 + t_8; m_1 = t_7 - t_8;$
$m_4 = sin(\pi/4) * (t_4 - t_6); m_7 = -j * sin(\pi/4) * (t_4 + t_6);$
$s_1 = m_3 + m_4; s2 = m_3 - m_4;$
$s_3 = m_6 + m_7; s4 = m_6 - m_7;$
$DO(0) = m_0; DO(4) = m_1;$
$DO(1) = s_1 + s_3; DO(7) = s_1 - s_3;$
$DO(2) = m_2 + m_5; DO(6) = m_2 - m_5;$
$DO(5) = s_2 + s_4; DO(3) = s_2 - s_4;$

where D and DO are input and output arrays of the complex data t_1,...,t_8, m_1,..., m_7, s_1,...,s_4 are the intermediate complex results. As we see the algorithm contains only 2 multiplications to the untrivial coefficient $\sin(\pi/4) = 0.7071$, and 26 * 2 real additions and subtractions. The multiplication to a coefficient j means the negation the imaginary part and swapping real and imaginary parts.

1.3 Implementation

The below code implements the Cooley–Tukey algorithm . inp1,...inp8 are the 16 bit signed floating point inputs of Q format and out1-real , out1-imag, ...,out8-real,out8-imag are the real and imaginary outputs in 16 bit Q format of the FFT8 module.clk , rst are the clock , reset inputs respectively , and output-stb is the output strobe , which is enabled once the output is calculated.

```
1  `timescale 1ns / 1ps
2
3  module fft8(
4          input signed [15:0] inp1,
5          input signed [15:0] inp2,
6          input signed [15:0] inp3,
7          input signed [15:0] inp4,
8          input signed [15:0] inp5,
9          input signed [15:0] inp6,
10         input signed [15:0] inp7,
11         input signed [15:0] inp8,
12         input clk,
13         input rst,
14         output signed [15:0] out1_real,
15         output signed [15:0] out1_imag,
16         output signed [15:0] out2_real,
17         output signed [15:0] out2_imag,
18         output signed [15:0] out3_real,
19         output signed [15:0] out3_imag,
20         output signed [15:0] out4_real,
21         output signed [15:0] out4_imag,
22         output signed [15:0] out5_real,
23         output signed [15:0] out5_imag,
```

```
24          output signed [15:0] out6_real,
25          output signed [15:0] out6_imag,
26          output signed [15:0] out7_real,
27          output signed [15:0] out7_imag,
28          output signed [15:0] out8_real,
29          output signed [15:0] out8_imag,
30          output out_stb
31  );
32
33          localparam signed sin_45 = 16'b00000000_10110101;
34          localparam signed sin_315 = 16'b11111111_01001011;
35
36          reg signed [31:0] t1_46,t2_46;
37          reg signed [15:0] t1,t2,t3,t4,t5,t6,t7,t8,m0,m1,m2,m3,m4,m5_imag,m6_imag,m7_i
38          reg output_stb;
39
40          initial
41                  begin
42                          output_stb = 1'b0;
43                  end
44
45          always @( posedge clk )
46                  begin
47                          if (rst == 1'b1)
48                                  begin
49                                          output_stb = 1'b0;
50                                  end
51                          else
52                                  begin
53                                          t1 = inp1 + inp5;
54                                          t2 = inp7 + inp3;
55                                          t3 = inp2 + inp6;
56                                          t5 = inp4 + inp8;
57                                          m3 = inp1 - inp5;
58                                          m6_imag = inp7 - inp3;
59                                          t4 = inp2 - inp6;
60                                          t6 = inp4 - inp8;
61                                          t8 = t5 + t3;
62                                          t7 = t1 + t2;
63                                          m0 = t7 + t8;
64                                          t1_46 = sin_45 * ( t4 - t6);
65                                          m4 = t1_46 [23:8];
66                                          m5_imag = t5 - t3;
67                                          m2 = t1 - t2;
68                                          m1 = t7 - t8;
69                                          t2_46 = sin_315 * ( t4 + t6);
70                                          m7_imag = t2_46 [23:8];
71                                          s1 = m3 + m4;
72                                          s2 = m3 - m4;
73                                          s3_imag = m6_imag + m7_imag;
```

```
74                                          s4_imag = m6_imag - m7_imag;
75                                          output_stb = 1'b1;
76                                  end
77                  end
78          assign out1_real = m0;
79          assign out1_imag = 16'b0000000000000000;
80          assign out2_real = s1;
81          assign out2_imag = s3_imag;
82          assign out3_real = m2;
83          assign out3_imag = m5_imag;
84          assign out4_real = s2;
85          assign out4_imag = ~s4_imag + 1'b1;
86          assign out5_real = m1;
87          assign out5_imag = 16'b0000000000000000;
88          assign out6_real = s2;
89          assign out6_imag = s4_imag;
90          assign out7_real = m2;
91          assign out7_imag = ~m5_imag + 1'b1;
92          assign out8_real = s1;
93          assign out8_imag = ~s3_imag + 1'b1;
94          assign out_stb = output_stb;
95  endmodule
```