# SIMULASI DAN ANALISIS *ERROR* KOMPUTASI FFT WINOGRAD 16-TITIK MENGGUNAKAN XILINX ISE 10.1I

**Irma Yulia Basri**[1]
[1)]Universitas Negeri Padang
Email : irma_yulia_77@yahoo.com

**Abstract - Weakness data processing using analog processors are less efficient because if there is an error in the design of a system using an analog processor, the hardware of the system should be redesigned. Processing of analog signals using digital processor has several advantages such as efficient and easy to modify the system are made, without requiring hardware redesign as well as in the design of analog systems. Errors in the design of digital circuits in the processor only requires modification program, without having to change the hardware. Modifications can be done anywhere, without demanding we must be in the laboratory. Fast Fourier Transform (FFT) is a computational algorithm that is used for digital data processing such as in the field of image, music, and satellite. Winograd FFT 16 point multiplier, multiplier consists of 6 pieces in the form of fractions. Design of FFT multiplier in Xilinx ISE 10.1i could do with changing fractions menajdi integers and then converted into a number by logic 1 and 0. Changing fractions into an integer will give different computational results compared with the original computation using a multiplier of FFT. The shift results enormous computing will result in lost / loss information can be conveyed from the processed data. To that end, researchers tried to simulate and analyze the computational error rate of 16 point Winograd FFT processor using Xilinx ISE 10.1i. The results showed average percentage error computational simulations using FFT 16 point Xilinx Ise 10.1i compare with Matlab is 6.67% (first trial) and 4:48% (second trial). This error occurs because the processor does not allow the FPGA digital display numbers in a real number.**

**Key word: FFT, Xilinx 10.1i**

## I. Pendahuluan

Sinyal merupakan besaran fisis yang berubah ubah menurut waktu, ruang ataupun variabel bebas lainnya. Sinyal ada 2 jenis yaitu sinyal analog dan sinyal digital. Pengolahan sinyal analog membutuhkan prosessor analog, sedangkan untuk pengolahan sinyal digital membutuhkan prosessor digital. Besaran fisis yang ada disekitar kita umumnya sinyalnya merupakan sinyal analog. Kelemahan pengolahan data menggunakan prosessor analog adalah kurang efisien karena apabila terjadi kesalahan dalam perancangan sebuah sistem menggunakan prosessor analog, maka perangkat keras dari sistem tersebut harus dirancang ulang, begitupun untuk modifikasi dari sebuah sistem yang dirancang menggunakan prosessor analog sangatlah sukar dilakukan. Kelemahan yang ada pada sistem menggunakan prosessor analog diatasi dengan pengolahan data analog menggunakan prosessor digital.

Perkembangan teknologi digital semakin pesat. Pengolahan sinyal analog menggunakan prosesor digital memiliki beberapa keunggulan diantaranya adalah efisien dan mudah dalam memodifikasi sistem yang dibuat, tanpa mengharuskan merancang ulang perangkat keras seperti halnya dalam perancangan sistem analog. Kesalahan dalam perancangan rangkaian dalam prosessor digital hanya membutuhkan modifikasi program, tanpa harus merubah perangkat keras. Modifikasi tersebut bisa dilakukan dimana saja, tanpa menuntut kita mesti berada di laboratorium.

Teknologi jenis ini untuk wilayah Sumatera Barat masih belum memasyarakat. Prosessor digital yang baru berkembang di tengah tengah akademisi Universitas Negeri Padang, baru terbatas pada mikroprosessor dan mikrokontroler. Mikrokontroler, mampu mengaplikasikan program program yang bersifat spesifik. Perancangan sebuah sistem didalam IC

mikrokontroler hanya bisa dilakukan untuk satu rancangan saja, sementara prosessor digital seperti *Field Programmable Gate Array* (FPGA) mampu mengkonfigurasi dan mengimplementasikan berbagai jenis program aplikasi dan algoritma komputasi untuk sebuah chip.

Algoritma komputasi yang bisa di rancang dan di implementasikan pada prosessor digital FPGA salah satunya adalah *Fast Fourier Transform* (FFT). FFT sebagai algoritma komputasi cepat bisa digunakan untuk pengolahan citra digital, seperti biomedical, navigasi, telekomunikasi, pengolahan suara dan musik, serta pengolahan video dan gambar. *Discrete Fourier Transform* (DFT) memainkan peranan penting dalam banyak aplikasi pengolahan sinyal digital, termasuk pentapisan linier, analisis korelasi, dan analisis spektrum. FFT adalah suatu algoritma untuk menghitung DFT yang digunakan untuk menghitung spektrum frekuensi sinyal yang telah dicuplik. FFT 16 titik merupakan FFT hasil modifikasi dari FFT Winograd. Algoritma FFT Winograd mempunyai jumlah komputasi yang sedikit sehingga sangat menguntungkan bila diimplementasikan ke dalam perangkat keras digital, namun jika hanya diimplementasikan ke perangkat lunak akan membutuhkan waktu komputasi yang cukup lama, karena kompleksitas algoritmanya [1]. Penelitian ini mencoba merancang dan mengkonfigurasi prosessor FFT Winograd 16 titik melalui perangkat keras digital FPGA. Perancangan FFT Winograd 16 titik menggunakan menggunakan prosessor digital seperti pada FPGA secara teoritis sangat mempercepat waktu perancangan, mempermudah implementasi dan modifikasi rancangan.

Tujuan dari penelitian ini adalah untuk mensimulasikan rancangan prosessor FFT Winograd 16 titik menggunakan Xilinx ISE 10.1i dan menganalisis kesalahan yang dihasilkan antara komputasi FFT Winograd 16 titik menggunakan Xilinx Ise 10.1i .

## II. Metode Penelitian

Metode penelitian yang digunakan dalam penelitian ini adalah rancang bangun, " prosessor FFT Winograd 16 titik" dengan menggunakan *Field Programmable Gate Array* (FPGA) Xilinx Spartan 3E dengan melakukan simulasi memanfaatkan Xilinx Ise 10.1i dan Matlab. Penelitian ini mengasilkan sebuah perangkat lunak prosessor FFT Winograd 16 Titik, dan dapat diimplementasikan menggunakan FPGA Xilinx Spartan 3E untuk pengolahan citra digital.

### Rancangan Penelitian

Perancangan FFT Winograd 16 titik agar dapat diimplementasikan pada FPGA dirancang dengan menggunakan perangkat lunak ISE 10.1i, keluaran dari Xilinx corp, merupakan perusahaan terbesar, yang memasok FPGA keseluruh negara di dunia. Xilinx ISE 10.1i merupakan perangkat lunak yang digunakan untuk simulasi dan perancangan prosessor digital. Bahasa pemograman yang digunakan pada Xilinxx ISE 10.1i adalah bahasa VHDL (*Very High Digital Language*). Sistem digital tidak mengenal angka pecahan, karena digital hanya terdiri dari bilangan 0 dan 1, begitupun dengan bahasa VHDL.

Pengali FFT Winograd 16 titik, terdiri dari 17 buah pengali dalam bentuk pecahan. Perancangan pengali FFT ke dalam Xilinx ISE 10.1i bisa dilakukan dengan mengubah bilangan pecahan menjadi bilangan bulat dan kemudian diubah menjadi bilangan dengan logic 1 dan 0. Pengubahan bilangan pecahan kedalam bilangan integer akan memberikan hasil komputasi yang berbeda jika dibandingkan dengan komputasi yang asli menggunakan pengali dari FFT. Pergeseran hasil komputasi yang sangat besar akan berakibat hilang/berkurangnya informasi yang bisa disampaikan dari data yang diolah.
Prosedur penelitian yang dilakukan adalah sebagai berikut:

1. Perancangan pengali FFT Winograd 16 Titik menggunakan algoritma Booth.

Pengali untuk FFT 16 titik terdiri dari 17 pengali yang disimbolkan dengan $B_0\ s.d\ B_{17}$.

$$\varphi = \frac{2\pi}{16}$$

$B_0 = B_1 = B_2 = B_3 = B_4 = 1$
$B_5 = B_6 = \cos(2\varphi)$
$B_7 = \cos 3\varphi$

**Tabel 1.** Bit Code Algoritma Booth untuk Pengali $B_5$

| $Y_{2i+1}$ | $Y_{2i}$ | $Y_{2i-1}$ | Recoded Digit | Bit Code | Operation dengan j | Keterangan |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 000 | 0j | Tambahkan 0 pada partial product |
| 0 | 1 | 0 | +1 | 010 | +1j | Tambahkan j pada partial product |
| 0 | 1 | 0 | +1 | 010 | +1j | Tambahkan j pada partial product |
| 1 | 1 | 0 | -1 | 110 | -1j | Tambahakan two' s komplement j pada partial product |
| 1 | 0 | 1 | -1 | 101 | -1j | Tambahkan two' s komplement j pada partial product |
| 0 | 0 | 1 | +1 | 001 | +1j | Tambahkan j pada partial product |

$B_8 = \cos\varphi + \cos 3\varphi$
$B_9 = -\cos\varphi + \cos 3\varphi$
$B_{10} = B_{11} = B_{12} = 1$
$B_{13} = B_{14} = -\sin 2\varphi$
$B_{15} = -\sin 3\varphi$
$B_{16} = -\sin\varphi + \sin 3\varphi$
$B_{17} = -\sin\varphi - \sin 3\varphi$

Tahapan yang dilakukan untuk mengubah pengali ke dalam algoritma booth adalah sebagai berikut:

a) Tentukan nilai pengali dalam bentuk bilangan desimal
b) Nilai pengali dikalikan dengan $2^{10}$ atau 1024 dikarenakan angka pengali merupakan bilangan pecahan (real) sementara untuk prosessor digital hanya mengenal bilangan integer (bilangan bulat)
c) Konversi hasil perkalian pengali dengan 1024 ke bilangan biner. Jika pengali merupakan bilangan negatif maka pengali harus dikonversi ke bilangan digital two's komplemen.

Hasil dari bilangan digital ditambahkan 0 untuk LSB, kemudian dicuplik tiga-tiga dari LSB untuk menentukan bit code algoritma booth nya. Contohnya untuk pengali $B_5 = B_6 = \cos(2\varphi)$

$$B5 = Cos\ \ 2Q\ \ =$$
$$Cos\ \ 2\ \ \frac{phi}{8}\ \ = \cos 45 = 0.707$$

Q = 2.phi/16 = phi/8
B5 * 210 = 724.07773439(10) =
B5 = B5*1024 = 724(10) = 2D4(16) = 0010 1101 0100(2)

Bit code pengali B5:
$0010\ 1101\ 0100_{(2)}$ + 0 = $0010\ 1101\ 01000_{(2)}$ →<u>001</u> <u>101</u> <u>110</u> <u>010</u> <u>010</u> <u>000</u>$_{(2)}$

d) Setelah bit code diperoleh maka dikonsultasikan dengan tabel algoritma booth Sehingga diperoleh hasilnya operasi partial product yang akan dilakukan adalah sebagai berikut:
e) Pembuatan tabel partial product berdasarkan bit code algoritma booth seperti pada tabel 1.

2. Lakukan ujicoba untuk memastikan perkalian sebuah bilangan dengan menggunakan tabel 1 hasilnya sama/mendekati dengan perkalian biasa. Jika sudah hasil sudah cocok, baru dilanjutkan ke langkah no 3.
3. Perancangan algoritma FFT Winograd 16 Titik menggunakan bahasa VHDL.

```
----------------------------------------------------
------------------------------
-- Create Date:    09:24:10 10/12/2012
-- Design Name: Irma Yulia Basri
-- Module Name:    m1 - Behavioral
-- Project Name: Pengali B5
----------------------------------------------------
------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity m1 is
   Port ( j : in  STD_LOGIC_VECTOR (11 downto 0);
          m_1 : out  STD_LOGIC_VECTOR (11 downto 0));
end m1;

architecture Behavioral of m1 is
signal s_1,j_inv,p,c_m: STD_LOGIC_VECTOR(11 DOWNTO 0);
signal c_1: STD_LOGIC_VECTOR(12 DOWNTO 0);
signal c_2: STD_LOGIC_VECTOR(14 DOWNTO 1);
signal s_3,c_3: STD_LOGIC_VECTOR(13 DOWNTO 0);
signal s_2: STD_LOGIC_VECTOR(13 DOWNTO 1);
signal c_4: STD_LOGIC;
signal g: STD_LOGIC_VECTOR(10 DOWNTO 0);

begin

j_inv <= not(j);

s_1(7 downto 0) <= j(11 downto 4) xor j(9 downto 2) xor j_inv(7 downto 0);
s_1(8) <= j(11) xor j(10) xor j_inv(8);
s_1(9) <= j(11) xor j(11) xor j_inv(9);
s_1(10) <= j(11) xor j(11) xor j_inv(10);
s_1(11) <= j(11) xor '1';

c_1(0) <= (j(3) and j(1)) or ((j(2) and j(0)) and (j(3) xor j(1)));
c_1(8 downto 1) <= (j(11 downto 4) and j(9 downto 2)) or (j_inv(7 downto 0) and (j(11 downto 4) xor j(9 downto 2)));
c_1(9) <= (j(11) and j(10)) or (j_inv(8) and (j(11) xor j(10)));
c_1(10) <= (j(11) and j(11)) or (j_inv(9) and (j(11) xor j(11)));
c_1(11) <= (j(11) and j(11)) or (j_inv(10) and (j(11) xor j(11)));
c_1(12) <= j(11);

s_2(1) <= s_1(1) xor c_1(1);
s_2(11 downto 2) <= s_1(11 downto 2) xor c_1(11 downto 2) xor j_inv(9 downto 0);
s_2(12) <= s_1(11) xor c_1(12) xor j_inv(10);
s_2(13) <= s_1(11) xor c_1(12) xor j_inv(11);

c_2(1) <=( s_1(0) and c_1(0)) or (s_1(0) xor c_1(0));
c_2(2) <= s_1(1) and c_1(1);
c_2(12 downto 3) <= (s_1(11 downto 2) and c_1(11 downto 2)) or (j_inv(9 downto 0) and (s_1(11 downto 2) xor c_1(11 downto 2)));
c_2(13) <= (s_1(11) and c_1(12)) or (j_inv(10) and (s_1(11) xor c_1(12)));
c_2(14) <= (s_1(11) and c_1(12)) or (j_inv(11) and (s_1(11) xor c_1(12)));

s_3(0) <= s_2(2) xor c_2(1) xor '1';
s_3(1) <= s_2(3) xor c_2(1);
s_3(11 downto 2) <= s_2(13 downto 4) xor c_2(13 downto 4) xor j(9 downto 0);
s_3(12) <= s_2(13) xor c_2(14) xor j(10);
s_3(13) <= s_2(13) xor c_2(14) xor j(11);

c_3(0) <= s_2(1) and c_2(1);
c_3(1) <=(s_2(2) and c_2(2)) or (s_2(2) xor c_2(2));
c_3(2) <= s_2(3) and c_2(3);
c_3(12 downto 3) <= (s_2(13 downto 4) and c_2(13 downto 4)) or (j(9 downto 0) and (s_2(13 downto 4) xor  c_2(13 downto 4)));
c_3(13) <= (s_2(13) and c_2(14)) or (j(10) and  (s_2(13) xor  c_2(14)));

c_4 <= (s_3(1) and c_3(1)) or ((s_3(1) xor c_3(1)) and (s_3(0) and c_3(0)));
p(11 downto 0) <= s_3(13 downto 2) xor c_3(13 downto 2);
g(10 downto 0) <= s_3(12 downto 2) and c_3(12 downto 2);
c_m(0) <= c_4;

process(p,g,c_m,c_4)
begin
c_m(0) <= c_4;
          c_m(1) <= g(0) or (c_m(0) and
```

```
p(0));
        inst: for i in 1 to 10 loop
        c_m(i + 1) <= g(i) or (p(i) and
c_m(i));
        end loop;
end process;
m_1(11 downto 0) <= p(11 downto 0) xor
c_m(11 downto 0);

end Behavioral;
```

Semua pengali dan program induk (top modul) dirancang dengan menggunakan konsep seperti diatas.

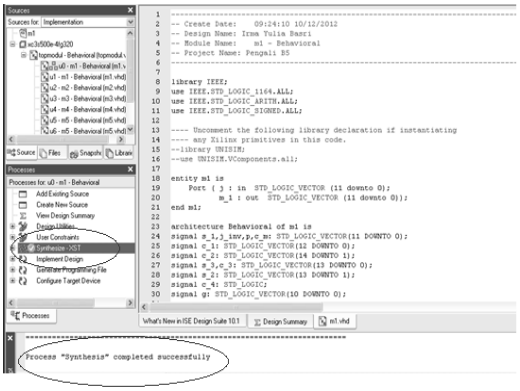
```
----------------------------------------------------
------------------------------
-- Create Date:    09:24:10 10/12/2012
-- Design Name: Irma Yulia Basri
-- Module Name: Top Modul - Behavioral
-- Project Name: Pengali B8  FFT
----------------------------------------------------
------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;

---- Uncomment the following library
declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity topmodul is
   Port                                (
v0,v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12
,v13,v14,v15: in  STD_LOGIC_VECTOR
(11 downto 0);

v_real0,v_real1,v_real2,v_real3,v_real4,v_
real5,v_real6,v_real7:              out
STD_LOGIC_VECTOR (11 downto 0);

v_real8,v_real9,v_real10,v_real11,v_real12
,v_real13,v_real14,v_real15:        out
STD_LOGIC_VECTOR (11 downto 0);

v_imag0,v_imag1,v_imag2,v_imag3,v_ima
g4,v_imag5,v_imag6,v_imag7:         out
STD_LOGIC_VECTOR (11 downto 0);

v_imag8,v_imag9,v_imag10,v_imag11,v_i
mag12,v_imag13,v_imag14,v_imag15: out
STD_LOGIC_VECTOR (11 downto 0));
end topmodul;

architecture Behavioral of topmodul is
signal
t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t1
4,t15,t16,t17,t18,t19,t20,t21:
STD_LOGIC_VECTOR(11 DOWNTO 0);
signal
a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a
13,a14,a15,a16,a17:
STD_LOGIC_VECTOR(11 DOWNTO 0);
signal
c0,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c
13,c14,c15,c16,c17:
STD_LOGIC_VECTOR(11 DOWNTO 0);
signal
S0,S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S
12,S13,S14,S15,S16,S17,S18,S19:
STD_LOGIC_VECTOR(11 DOWNTO 0);

component m1 is
   Port ( j : in  STD_LOGIC_VECTOR (11
downto 0);
        m_1 : out   STD_LOGIC_VECTOR
(11 downto 0));
end component;

component m2 is
   Port ( k : in   STD_LOGIC_VECTOR
(11 downto 0);
        m_2 : out   STD_LOGIC_VECTOR
(11 downto 0));
end component;

component m3 is
   Port ( n : in   STD_LOGIC_VECTOR
(11 downto 0);
        m_3 : out   STD_LOGIC_VECTOR
(11 downto 0));
end component;

component m4 is
   Port ( n : in   STD_LOGIC_VECTOR
(11 downto 0);
        m_4 : out   STD_LOGIC_VECTOR
(11 downto 0));
end component;

component m5 is
   Port ( n : in   STD_LOGIC_VECTOR
(11 downto 0);
        m_5 : out   STD_LOGIC_VECTOR
(11 downto 0));
end component;

component m6 is
   Port (  n : in   STD_LOGIC_VECTOR
(11 downto 0);
```

```
        m_6 : out   STD_LOGIC_VECTOR
(11 downto 0));
end component;

component m7 is
   Port ( n : in   STD_LOGIC_VECTOR
(11 downto 0);
        m_7 : out   STD_LOGIC_VECTOR
(11 downto 0));
end component;

component m8 is
   Port ( n : in   STD_LOGIC_VECTOR
(11 downto 0);
        m_8 : out   STD_LOGIC_VECTOR
(11 downto 0));
end component;


begin

-- step 1 Menentukan nilai t0 s.d t13
t0 <= v0 + v8;
t1 <= v4 + v12;
t2 <= v2 + v10;
t3 <= v2 - v10;
t4 <= v6 + v14;
t5 <= v6 - v14;
t6 <= v1 + v9;
t7 <= v1 - v9;
t8 <= v3 + v11;
t9 <= v3 - v11;
t10 <= v5 + v13;
t11 <= v5 - v13;
t12 <= v7 + v15;
t13 <= v7 - v15;

-- step 2 Menentukan nilai t14 s.d t21
t14 <= t0 + t1;
t15 <= t2 + t4;
t16 <= t14 + t15;
t17 <= t6 + t10;
t18 <= t6 - t10;
t19 <= t8 + t12;
t20 <= t8 - t12;
t21 <= t17 + t19;

-- step 3 Mennetukan nilai a
a0 <= t16 + t21;
a1 <= t16 - t21;
a2 <= t14 - t15;
a3 <= t0 - t1;
a4 <= v0 - v8;
a5 <= t18 - t20;
a6 <= t3 - t5;

a8 <= t7 - t13;
a9 <= t11 - t9;
a10 <= t19 - t17;
a11 <= t4 - t2;
a12 <= v12 - v4;
a13 <= t18 + t20;
a14 <= t3 + t5;
a16 <= t7 + t13;
a17 <= t11 + t9;

-- step 4
a7 <= a8 + a9;
a15 <= a16 + a17;

-- step 5
c0 <= a0;
c1 <= a1;
c2 <= a2;
c3 <= a3;
c4 <= a4;


----- perkalian antara a5 dengan B5 (m1)
satu: m1
        port map (a5,c5);


----- perkalian antara a6 dengan B5 (m1)
satu1: m1
        port map (a6,c6);


----- perkalian antara a7 dengan B7 (m2)
dua: m2
        port map (a7,c7);


----- perkalian antara a8 dengan B8 (m3)
tiga: m3
        port map (a8,c8);


----- perkalian antara a9 dengan B9 (m4)
empat: m4
        port map (a9,c9);


c10 <= a10;
c11 <= a11;
c12 <= a12;

----- perkalian antara a13 dengan B13 (m5)
lima: m5
        port map (a13,c13);


----- perkalian antara a14 dengan B14 (m5)
enam: m5
        port map (a14,c14);
```

```
----- perkalian antara a15 dengan B15 (m6)
tujuh: m6
        port map (a15,c15);


----- perkalian antara a16 dengan B16 (m7)
delapan: m7
        port map (a16,c16);


----- perkalian antara a16 dengan B16 (m8)
sembilan: m8
        port map (a17,c17);



-- step 6
S0 <= c3 + c5;

S1 <= c3 - c5;

S2 <= c11 + c13;
S3 <= c13 - c11;
S4 <= c4 + c6;
S5 <= c4 - c6;
S6 <= c8 - c7;
S7 <= c9 - c7;
S12 <= c12 + c14;
S13 <= c12 - c14;
S14 <= c15 + c16;
S15 <= c15 - c17;


-- step 7
S8 <= S4 + S6;
S9 <= S4 - S6;
S10 <= S5 + S7;
S11 <= S5 - S7;
S16 <= S12 + S14;
S17 <= S12 - S14;
S18 <= S13 + S15;
S19 <= S13 - S15;

-- step 8
v_real0 <= c0;
--v_imag0 <= '0';

v_real1 <= S8;
v_imag1 <= S16;

v_real2 <= S0;
v_imag2 <= S2;

v_real3 <= S11;
v_imag3 <= -(S19);

v_real4 <= c2;
v_imag4 <= c10;

v_real5 <= S10;
v_imag5 <= S18;

v_real6 <= S1;
v_imag6 <= S3;

v_real7 <= S9;
v_imag7 <= -(S17);

v_real8 <= c1;


v_real9 <= S9;
v_imag9 <= S17;

v_real10 <= S1;
v_imag10 <= -(S3);

v_real11 <= S10;
v_imag11 <= -(S18);

v_real12 <= c2;
v_imag12 <= -(c10);

v_real13 <= S11;
v_imag13 <= S19;

v_real14 <= S0;
v_imag14 <= -(S2);

v_real15 <= S8;
v_imag15 <= -(S16);

end Behavioral;
```

4. Perancangan algoritma FFT Winograd 16 Titik menggunakan menggunakan Matlab.
5. Perancangan simulasi algoritma FFT Winograd 16 Titik menggunakan Xilinx ISE 10.1i

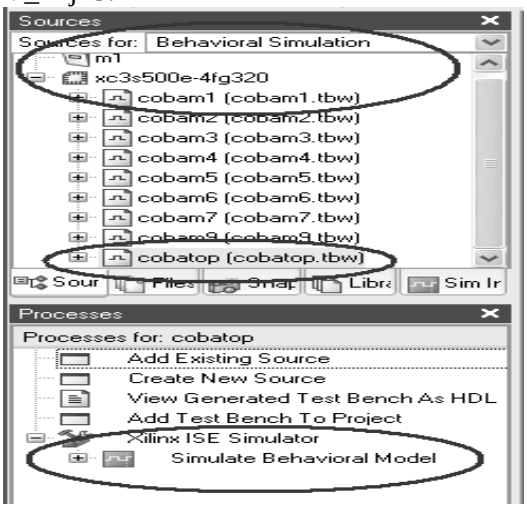
Sebuah program VHDL dikatakan sukses dioperasikan dengan Xilinx Ise 10.1i bila sintesis telah berhasil dan output saat disimulasikan menampilkan angka yang mendekati seperti yang telah dianalisis secara manual maupun menggunakan matlab.



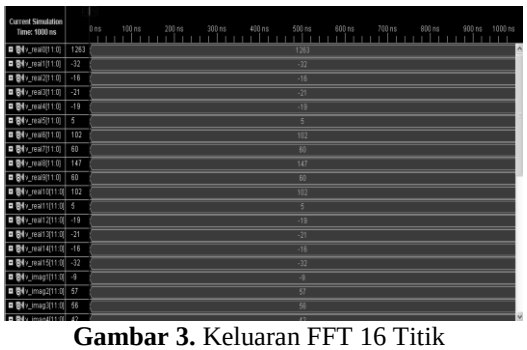
**Gambar 1.** Sistesis Pengali $B_5$ Xilinx Ise 10.1i

6. Membandingkan komputasi algoritma FFT Winograd 16 Titik yang dirancang menggunakan FPGA/ Xilix Ise 10.1i dengan komputasi yang dilakukan oleh Matlab. Hasil data keluaran dari FFT ditampilkan dengan menggunakan fasilitas *Behavioral Simulation*

Saat ditampilkan simulation behavior model, maka tampilan keluaran untuk top level FFT 16 titik seperti gambar 3. Pada gambar keluaran antara bilangan real dengan imajiner dipisah. Untuk bilangan real V0 s.d V15 disimbolkan dengan V_real0 s.d V_real15, dan untuk tampilan imajiner disimbolkan dengan V_imj0 s.d V_imj15.




**Gambar 2.** Behavioral simulation



**Gambar 3.** Keluaran FFT 16 Titik

7. Menganalisis tingkat kesalahan peracangan algoritma FFT Winograd 16 Titik menggunakan Xilinx Ise 10.1i. Hasil simulasi komputasi prosessor FFT Winograd 16 Titik dibandingkan hasilnya dengan FFT Winograd 16 Titik menggunakan komputasi Matlab. Data yang dianalisis dalam penelitian ini dilakukan dua kali percobaan dengan data masukan yang berbeda. Data masukan merupakan cuplikan 16 titik dari sinyal periodik yang diambil secara acak, untuk data percobaan pertama adalah:

$x_{[n]}$ = [98, 20, 94, 79, 80, 84, 96, 68, 95, 85, 77, 78, 70, 79, 95, 75]

Data input untuk percobaan kedua adalah:

$x_{[n]}$ = [98, 20, 94, 79, 80, 84, 96, 68, 95, 85, 77, 78, 70, 79, 95, 75]

Prosentase kesalahan (*error*) dapat ditentukan dengan persamaan:

$$\%\ Kesalahan = \frac{Selisih\ Perhitungan\ Komputasi\ menggunakan\ Matlab\ dengan\ FPGA}{Nilai\ Kompuatsi\ Matlab}\ x\ 100\%$$

## III. Hasil dan Pembahasan

Setelah dilakukan prosedur penelitian seperti yang dipaparkan pada Bab IV, maka data hasil penelitian yang diperoleh untuk percobaan pertama seperti terlihat pada tabel 2.

Pengolahan frekwensi sinyal cuplik menggunakan prosessor FFT Winograd 16 Titik yang dirancang menggunakan Xilinx Ise 10.1i, mengasilkan rata-rata kesalahan untuk percobaan pertama adalah 6.67%. Selisih nilai yang terjadi antara komputasi menggunakan Xilinx Ise 10.1i dengan Matlab, karena melalui Xilix 10.1i yang diolah prosessor digital sedangkan prosessor digital tidak menfasilitasi untuk memunculkan keluaran dalam bentuk bilangan berkoma (real) sehingga terjadi beberapa proses pembulatan bilangan, yang pastinya akan mengurangi informasi yang dihasilkan dari FFT Winograd 16 titik.

Percobaan kedua mengasilkan rata-rata kesalahan 6.67%, artinya akan terjadi kehilangan informasi sebesar 6.67% terhadap data yang diolah menggunakan FFT Winograd yang dirancang menggunakan Xilinx Ise 10.1i. Enam belas keluaran dari FFT Winograd 16 Titik, untuk titik keluaran $X_0$, $X_4$, $X_{15}$, $X_{12}$ mengasilkan prosentase kesalahan sebesar 0%. Hal ini dikarenakan $X_0$, $X_4$, $X_{15}$, $X_{12}$ , komputasi yang dihasilkan dalam bentuk bilangan integer baik yang diolah menggunakan Xilinx Ise 10.1i maupun dengan Matlab, sehingga tidak terjadi proses pembulatan nilai sinyal cuplik. Pembulatan pertama kali terjadi pada saat perancangan pengali, dan operasi masing-masing tahapan sampai mencapai keluaran FFT. Akan tetapi meskipun terjadi perbedaan hasil komputasi tersebut, hal ini tidak terlalu berpengaruh untuk aplikasi FFT Winograd 16 titik menggunakan FPGA Xilinx. Sampai saat ini justru teknologi FPGA dan simulasi menggunakan Xilinx Ise 10.1i tetap berkembang, seperti dibidang pengolahan citra digital maupun untuk komputasi.

**Tabel 2.** Data Percobaan Pertama

| No | Input | Output | | Error |
|---|---|---|---|---|
| | | **FFT 16 Titik ISE 10.1i** | **FFT 16 Titik Matlab** | |
| 1 | $x_0$=98 | 1273 | 1273 | 0% |
| 2 | $x_1$=20 | -41-5i | -40.8020 - 4.7180i | 0.49% |
| 3 | $x_2$=94 | -9+50i | -7.9117 +51.1127i | 13.78% |
| 4 | $x_3$=79 | -25+65i | -26.8138 +66.0875i | 6.75 |
| 5 | $x_4$=80 | -19+32i | -19.0000 +32.0000i | 0 |
| 6 | $x_5$=84 | 9+73i | 10.1864 +71.5433i | 11.59% |
| 7 | $x_6$=96 | 95+10i | 93.9117 +11.1127i | 1.16% |
| 8 | $x_7$=68 | 69+43i | 69.4295 +44.7379i | 0.06% |
| 9 | $x_8$=95 | 137 | 137 | 0 |
| 10 | $x_9$=85 | 69-43i | 69.4295 -44.7379i | 0.06% |
| 11 | $x_{10}$=77 | 95-10i | 93.9117 -11.1127i | 1.16% |
| 12 | $x_{11}$=78 | 9-73i | 10.1864 -71.5433i | 11.59% |
| 13 | $x_{12}$=70 | -19-32i | -19.0000 -32.0000i | 0 |
| 14 | $x_{13}$=79 | -25-65i | -26.8138 -66.0875i | 6.75% |
| 15 | $x_{14}$=95 | -9-50i | -7.9117 -51.1127i | 13.78% |
| 16 | $x_{15}$=75 | -41+5i | -40.8020 + 4.7180i | 0.49% |
| Rata-rata prosentase kesalahan | | | | 6.76% |

**Tabel 3.** Data Percobaan Kedua

| No | Input | Output | | Error |
|---|---|---|---|---|
| | | **FFT 16 Titik ISE 10.1i** | **FFT 16 Titik Matlab** | |
| 1 | $x_O$=18 | 243 | 243 | 0 |
| 2 | $x_1$=10 | -5+6i | -4.9234 + 5.8751i | 1.86% |
| 3 | $x_2$=14 | 4+3i | 4.5147 + 2.8284i | 11.08% |
| 4 | $x_3$=19 | -2+1i | -2.7763 + 1.7312i | 2.77% |
| 5 | $x_4$=10 | -9+12i | -9.0000 +12.0000i | 0 |
| 6 | $x_5$=14 | 14-1i | 14.4332 - 1.0972i | 2.97% |
| 7 | $x_6$=16 | 22+2i | 21.4853 + 2.8284i | 2.23% |
| 8 | $x_7$=18 | 5+3i | 5.2665 + 3.0467i | 4.94% |
| 9 | $x_8$=15 | -13 | -13 | 0 |
| 10 | $x_9$=15 | 5+3i | 5.2665 - 3.0467i | 4.94% |
| 11 | $x_{10}$=17 | 22-2i | 21.4853 - 2.8284i | 2.23% |
| 12 | $x_{11}$=18 | 14+1i | 14.4332 + 1.0972i | 2.97% |
| 13 | $x_{12}$=10 | -9-12i | -9.0000 -12.0000i | 0 |
| 14 | $x_{13}$=19 | -2+1i | -2.7763 - 1.7312i | 11.08% |
| 15 | $x_{14}$=15 | 4-3i | 4.5147 - 2.8284i | 1.86% |
| 16 | $x_{15}$=15 | -5+6i | -4.9234 - 5.8751i | 0.49% |
| Rata-rata prosentase kesalahan | | | | 4.48% |

## IV. Kesimpulan

Rata-rata prosentase kesalahan komputasi prosessor FFT Winograd 16 Titik menggunakan Xilinx Ise 10.1i untuk percobaan pertama adalah 6.67% , artinya informasi yang hilang dari hasil pengolahan FFT Winograd 16 Titik menggunakan Xilinx ISE 10.1i sebesar 6.67%.

Rata-rata prosentase kesalahan komputasi prosessor FFT Winograd 16 Titik menggunakan Xilinx Ise 10.1i untuk percobaan kedua adalah 4.48% , artinya informasi yang hilang dari hasil pengolahan FFT Winograd 16 Titik menggunakan Xilinx ISE 10.1i sebesar 4.48%.

Kesalahan terjadi karena prosessor FFT Winograd 16 Titik yang dirancang merupakan prosessor digital, dan perangkat ini tidak bisa menampilkan bilangan real.

## Daftar Pustaka

[1] Sutopo, B., (1994). *Unjuk Kerja Algoritma Algoritma FFT untuk Sembarang Cacah Sampel.* Jogyakarta: Teknik Elektro UGM (Seminar Nasional TE UGM)

[2] Agostini, L.V., Silva, I.S., dan Bampi, S., (2007) *"Multiplierless and fully pipelined JPEG compression soft IP targeting FPGAs",* Microprocessors and Microsystems 31 (2007) 487–497, www.elsevier.com/locate/micpro

[3] Blahut, R.E., (1985). *Fast Algoritms for Digital Signal Processing.* California*:* Addison – Wesley Publising Co, Reading: Massachusetts

[4] Sutopo, B., (2000), *" Implementasi FFT pada FPGA dengan Algoritma Winograd Kecil"* Quality In Research Seminar (QIR-2000), Fakultas Teknik, Universitas Indonesia.

[5] Xilinx., (2006). *User Guide Xilinx Spartan 3E*. Xilinx Co.