Penerapan *Floating Point Unit* Sebagai *Co-processor* yang Diimplementasikan dengan Menggunakan FPGA

Kanzi Mahfi[1], Mochammad Rif'an, ST., MT.[2], Waru Djuriatno, S.T., M.T.[3]

***Abstract*— Arithmetic circuit is an important part of the digital system. With the tremendous advances in VLSI, more complex circuits, which today is unthinkable to be easily realized at this time. In this thesis work arithmetic unit based on the IEEE 754 standard for single-precision floating-point numbers has been implemented on Spartan 3E FPGA. Arithmetic units that have implemented 32-bit processing unit that is able to perform arithmetic operations such as addition, subtraction, multiplication and division operations and be able to handle the specific calculations that can't be done by standard operating model. FPGA synthesis for arithmetic unit is done using Xilinx ISE 11.1. The result is the arithmetic unit capable of working calculates two single-precision floating point numbers with the IEEE 754 standard time required for 233.689ns.**

***Index Terms*— Floating Point, FPGA, IEEE 754 Standard, Arithmetic Unit, Arithmetic Operation**

***Abstrak*–- Sirkuit aritmatika merupakan bagian penting dari sistem digital. Dengan kemajuan luar biasa dalam VLSI, banyak sirkuit yang kompleks, yang kemarin tak terpikirkan menjadi mudah terealisasi saat ini.Dalam skripsi ini unit aritmatika bekerja berdasarkan standar IEEE 754 untuk bilangan *floating point* presisi tunggal telah diimplementasikan pada FPGA Spartan 3E. Unit aritmatika yang diimplementasikan memiliki unit pengolahan 32 bit yang mampu untuk melakukan operasi aritmatika seperti, penjumlahan, pengurangan, perkalian dan pembagian serta mampu menangani operasi perhitungan khusus yang tidak bisa dikerjakan oleh model operasi standar. Sintesis FPGA untuk unit aritmatika dilakukan dengan menggunakan Xilinx ISE 11.1.Hasilnya unit aritmatika ini mampu bekerja menghitung dua buah bilangan *floating point* presisi tunggal standar IEEE 754 dengan waktu yang dibutuhkan sebesar 233.689ns.**

***Kata Kunci*— Floating Point, FPGA, Standar IEEE 754, Unit Aritmatika, Operasi Aritmatika.**

I. PENDAHULUAN

FPGA telah menetapkan diri sebagai alat berharga dalam pelaksanaan sistem kerja tinggi, yang menggabungkan keunggulan *reprogrammability processor* dengan kecepatan dan keunggulan pemrosesan paralel pada suatu perangkat keras.Namun masalah yang sering dijumpai dalam sistem yang berbasis FPGA adalah pemrosesan aritmatika bilangan ilmiah yang biasanya direpresentasikan dalam bentuk bilangan *floating point*. Hal ini memang disebabkan pemrosesan bilangan floating point bukan termasuk komponen standar FPGA.Padahal banyak sistem yang membutuhkan perhitungan bilangan ilmiah. Misalnya pada pemrosesan sinyal digital dan pengolahan gambar. Oleh karena itu diperlukan implementasi sistem yang dapat menghitung bilangan *floating point* dalam FPGA [1].

Metode standar internasional untuk mewakili angka pada komputer telah ditetapkan oleh standar IEEE 754 untuk memenuhi tujuan-tujuan dasar [2] [3].

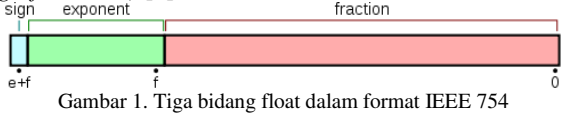
Floating Point Unit (FPU) sebagai unit yang bekerja khusus untuk memproses perhitungan bilangan *floating point* menjadi bagian penting dari banyak komputasi ilmiah, FPU ini dapat berguna dalam pelaksanaan sistem FPGA yang kompleks yang tidak hanya memanfaatkan *reprogrammability* dan paralelisme FPGA tetapi juga yang membutuhkan suatu unit aritmatika yang dapat menghitung bilangan ilmiah. Karena FPU ini akan diimplementasikan pada FPGA maka sebagian pengolahan *floating point* diterapkan pemrosesan secara paralel dan penggunaan blok-blok logika khusus agar *floating point unit* yang dirancang dapat melakukan perhitungan dengan kecepatan tinggi.

II. TINJAUAN PUSTAKA

A. *Floating Point Standar IEEE 754*

Bentuk standar *floating point* IEEE 754 awalnya ditentukan pada tahun 1985, kemudian mengalami revisi pada tahun 2008.Standar IEEE 754 dijadikan standar aritmatika bilangan *floating point* yang didukung oleh seluruh CPU [2].

Kode biner bilangan *floating point* dalam format IEEE 754 terdiri dari 3 bidang, yaitu: bit tanda, eksponen, dan pecahan tanpa bit paling signifikan (*most significant bit*) [2].



Gambar 1. Tiga bidang float dalam format IEEE 754

[1] *Mahasiswa, Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, Desa Gunungrejo RT.17 RW.05 Kec. Singosari Kab. Malang 65163 INDONESIA (telp: 083848583834; e-mail: zifi@mail.com)*

[2, 3] *Dosen, Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, Jalan MT. Haryono 167, Malang 65145 INDONESIA.*

Eksponen tidak disimpan dalam bentuk komplemen dua, tetapi dalam bentuk format bias (*offset biner*).

$E = e - bias$ .....(1)

e adalah nilai tak bertanda (*unsigned*) yang dikonversikan langsung dari eksponen, dan $bias = 2^{n-1} - 1$ dimana n adalah banyaknya bit eksponensial.

MSB (*most significant bit*) dalam signifikan tidak disimpan tetapi dapat ditentukan dari nilai eksponen.Jika $0 < eksponen < 2^{E-1}$, maka MSB dari signifikan adalah 1, dan nilai ini dinamakan ternormalisasi. Jika eksponen 0 dan signifikan tidak 0 maka MSB dari signifikan adalah 0 maka nilai ini dikatakan ter-denormalisasi [2].

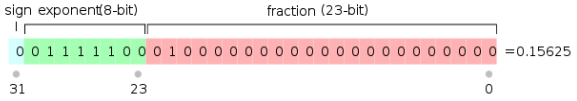
Dan tiga kasus khusus lainnya yang muncul adalah [1]:

1) Jika eksponen adalah 0 dan mantisa (*fraction)* adalah 0, maka nilainya adalah ± 0 (tergantung dari bit tanda).
2) Jika eksponen= 2^{e-1} dan mantisa adalah 0, maka nilainya adalah $\pm\infty$ (tergantung dari bit tanda).
3) Jika eksponen= 2^{e-1} dan mantisa tidak sama dengan 0, maka nilai yang diwakilkan adalah bukan angka (*Not a Number*).

B. *Floating Point Presisi Tunggal*

Dalam sistem bilangan *floating point* standar IEEE 754, format bilangan dibagi menjadi bilangan *floating point* biner dan desimal.Format bilangan *floating point* biner dibagi menjadi empat yaitu, biner 16 (*half precission*), biner 32 (*single precission*), biner 64 (*double precission*), dan biner 128 (*quadruple precission*). Bilangan *floating point* biner 32 atau bilangan biner *floating point* presisi tunggal disimpan dalam 32 bit, yang terdiri dari 1 bit *sign*, 8 bit eksponen, dan 23 bit mantissa [2].

Mantissa pada format *floating point* standar IEEE 754 presisi tunggal memiliki bit tersembunyi yang bernilai 1 kecuali jika semua bit eksponent bernilai nol. Bit tersembunyi ini menjadi MSB pada nilai mantisa. Dengan 23 bit mantissa yang muncul pada format memori, maka total presisi untuk mantisa adalah 24 bit atau sekitar 7 digit desimal ($7 \log_{10} 2 \approx 7.224$) [1].



Gambar2. Susunan bit bilangan *floating point* presisi tunggal

Nilai riil yang diasumsikan oleh data yang diberikan bilangan *floating point* presisi tunggal dengan memperhitungkan eksponen bias dan bit tersembunyi mantissa adalah:

$V = (-1)^{sign}(1 + \sum_{i=1}^{23}[b_{-i}]\, 2^{-i}) \times 2^{(e-127)}$.....(2)

Dengan:

- $sign$ adalah bit tanda
- e adalah eksponen sebesar 8 bit
- b adalah mantissa sebesar 23 bit

C. *Aritmatika Bilangan Floating Point*

Perhitungan dua bilangan A dan bilangan B dengan hasil bilangan R yang masing-masing terdiri dari bit tanda (S_A, S_B, dan S_R), eksponen (E_A, E_B, dan E_R), dan pecahan (M_A, M_B, dan M_R) [2] [3].

Langkah-langkah utama dalam perhitungan R jumlah atau selisih dua bilangan *floating point* A dan B adalah sebagai berikut:

1) Menghitung nilai absolut dari selisih kedua eksponen, yaitu $|E_A - E_B|$, dan mengatur E_R eksponen hasil dari eksponen *operand* yang nilainya lebih besar.
2) Lakukan pergeseran signifikan dengan eksponen yang lebih kecil ke kanan sebanyak $|E_A - E_B|$.
3) Tambahkan atau kurangkan M_A dan M_B sesuai dengan operasi yang efektif berdasarkan operasi yang dinginkan dan kedua tanda negatif *operand* ($S_A$dan S_B), dan dapatkan mantisa hasil perhitungan M_R sekaligus tanda negatif hasil perhitungan S_R.
4) Normalisasi M_R dan sesuaikan nilai E_R-nya, dan bulatkan nilai M_R yang mungkin memerlukan pengkajian ulang terhadap nilai E_R.

Sedangkan langkah-langkah utama dalam perhitungan R hasil perkalian dari perkalian dua bilangan A dan B adalah sebagai berikut:

1) Menghitung eksponen dari hasil perhitungan $E_R = E_A + E_B - E_{BIAS}$.
2) Kalikan M_A dan M_B untuk mendapatkan hasil dari mantisa M_R.
3) Normalisasi M_R dan sesuaian E_R. Bulatkan nilai M_R yang mungkin memerlukan pengkajian ulang terhadap nilai E_R.

Dan langkah-langkah penting dalam perhitungan R hasil dari bilangan A dibagi bilangan B adalah sebagai berikut:

1) Menghitung ekponen dari hasil perhitungan $E_R = E_A - E_B + E_{BIAS}$.
2) Bagi nilai M_A dengan M_B untuk mendapatkan hasil dari mantisa M_R.
3) Normalisasi M_R dan sesuaian E_R. Bulatkan nilai M_R yang mungkin memerlukan pengkajian ulang tehadap nilai E_R.

Dalam sistem aritmatika *floating point* terdapat unit penangan khusus untuk masukan bernilai tertentu. Unit penangan khusus ini biasa disebut dengan *exception handler* [2].

IEEE standar mendefinisikan 5 jenis pengecualian yang harus diisyaratkan melalui *flag status* ketika ditemui operasi yang membutuhkan penanganan khusus. Antara lain: [2] [3]

1) *Invalid Operation*

Beberapa operasi aritmatika adalah tidak valid, misalnya adalah 0 dibagi 0. Hasil perhitungan ini akan menghasilkan keluaran berupa *Not a Number*.

2) *Division By Zero*

Dalam matematika, pembagian ini disebut pembagian dengan nol ketika nilai pembagi adalah 0. Pembagian ini dapat dinyatakan sebagai $a\ /\ 0$, dimana a adalah bilangan terbagi (dividen).

3) *Inexact*

Inexact adalah jenis pengecualian yang mengisyaratkan ketika hasil operasi aritmatika tidak tepat karena keterbatasan nilai eksponen atau karena rentang presisi.

4) Overflow

Pengecualian *overflow* mengisyaratkan ketika hasil perhitungan melebihi nilai maksimum yang dapat diwakili oleh format *floating point* standar IEEE 754.

5) Underflow

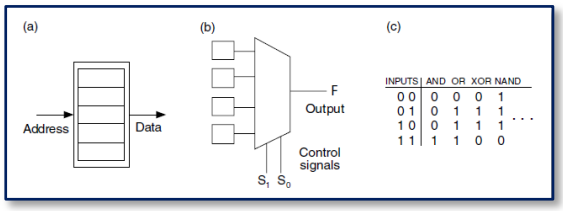
Pengecualian *underflow* mengisyaratkan ketika hasil dari perhitungan lebih kecil dari nilai minimum yang dapat diwakili oleh format *floating point* standar IEEE 754.

D. Field-Programmable Gate Array (FPGA)

Field-Programmable Gate Array (FPGA) dapat digunakan untuk mengimplementasikan hampir semua desain hardware. Salah satu penggunaan umum dari FPGA adalah prototipe dari perangkat keras yang pada akhirnya akan dilaksanakan kemudian ke sebuah ASIC (*Application Spesific Integrated Circuit*). Namun demikian, FPGA telah semakin banyak digunakan sebagai platform produk akhir. Penggunaannya tergantung, untuk proyek tertentu, pada bobot relatif kinerja yang diinginkan, pengembangan, dan biaya produksi [3].

Konsep dasar arsitektur FPGA terdiri dari *array* dua dimensi dari blok logika dan flip-flop tersusun sedemikian rupa agar pengguna dapat mengonfigurasi fungsi masing-masing blok logika, masukan dan keluaran, dan interkoneksi antar blok logika [4] [5] [6].

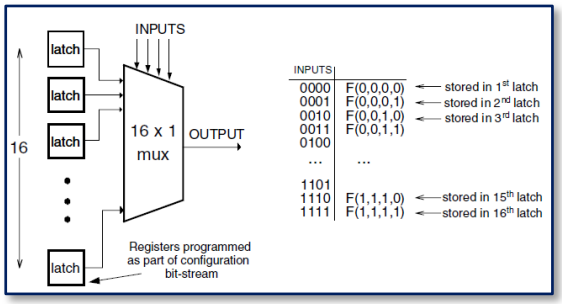
Ada beberapa jenis metode pemrograman pada FPGA, *Look-up table* adalah alternatif lain untuk mengimplementasikan fungsi logika kedalam FPGA. Blok logika yang melaksanakan fungsi logika adalah *look-up table* (LUT), diimplementasikan sebagai memori, atau multiplekser dan memori [4] [5] [6].



Gambar 3.*Look-up table* diimplementasikan sebagai (a) Memori. (b) Multiplekser dan Memori. (c) Contoh konten memori untuk beberapa fungsi logika. [4]

Masukan kontrol multiplekser adalam masukan LUT, keluarannya adalah gerbang logika seperti pada umumnya.Sebuah LUT dapat diimplementasikan sembarang fungsi sebesar n-bit [4] [5] [6].

Sebuah n-LUT adalah implementasi langsung dari table kebenaran fungsi. Setiap *latch* menahan nilai fungsi sesuai dengan suatu kombinasi masukan. Sebuah contoh dari 4-LUT ditujukkan pada Gambar 2.6.

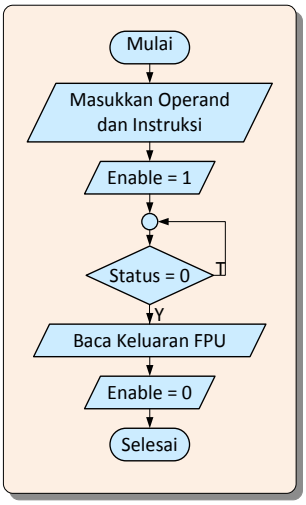


Gambar 4. Implementasi 4-LUT dan isi tabel kebenarannya [4]

III. PERANCANGAN FPU

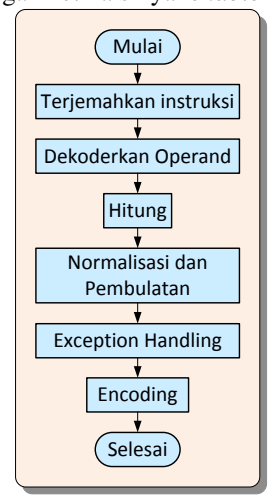
A. Sistem FPU

Secara umum diagam alir pengerjaaan FPU ditunjukkan pada Gambar 5.



Gambar 5. Diagram alir pengerjaan FPU

Dan di dalam FPU, FPU baru akan melakukan proses perhitungan ketika sinyal *enable* tepi naik.

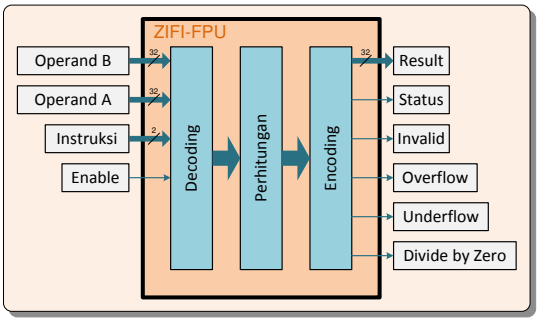


Gambar 6 Diagram alir proses perhitungan pada FPU

B. Pembangunan Hardware FPU

Masing-masing unit *hardware* manipulasi bit bersifat sangat khusus, ia hanya dapat melakukan tugas yang telah ditentukan dalam perancangan, tidak dapat melakukan tugas lain dan bahkan *bus* data-nya benar-benar disesuaikan dengan *bus* data yang akan diproses. Hal ini dilakukan guna mempermudah dalam proses perancangan dan untuk menghemat *slice resources* dalam FPGA yang digunakan.

Dimana FPU yang akan dirancang dapat melakukan empat macam operasi aritmatika: Penjumlahan, pengurangan, perkalian dan pembagian.



Gambar 7. Blok Diagram FPU yang akan dirancang.

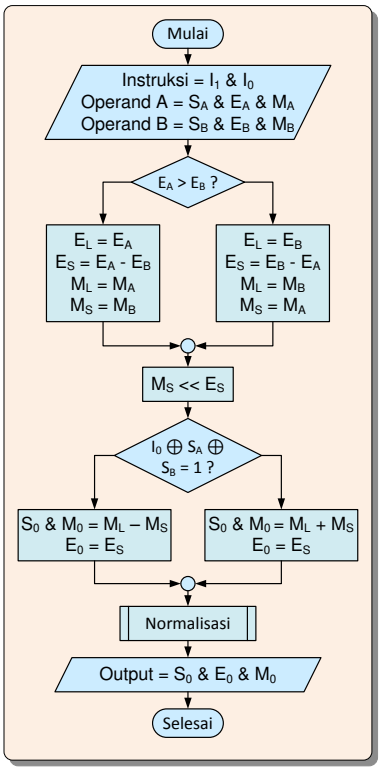
Di dalam FPU tahapan proses perhitungan bilangan *floating point* dapat dibedakan menjadi tiga tahapan, antara lain:

1) Pendekoderan, *operand* masukan diubah kedalam format terpecah antara sign, eksponen, mantissa dan *status flag*. Hal ini agar untuk membuat mudah dan efisien proses di dalam FPU.
2) Penghitungan, terdiri dari perhitungan aritmatika yang terjadi dan proses normalisasi. Penanganan perhitungan khusus juga termasuk dalam tahapan ini.
3) Encoding, Pengemasan hasil keluaran perhitungan atau hasil *exception handling* kedalam format bilangan *floating point* presisi tunggal standar IEEE 754.

C. *Algoritma FPU*

1) *Algoritma Penjumlahan dan Pengurangan*

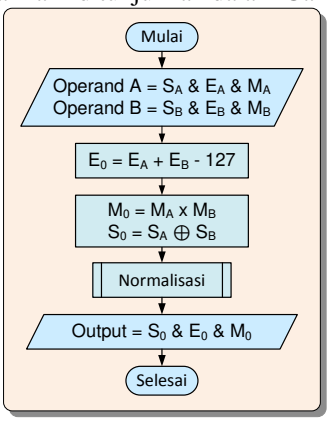
Algoritma yang digunakan untuk menjumlah dan mengurangi dua buah bilangan *floating point* bila dijabarkan dalam bentuk diagram alir ditunjukkan pada Gambar 8.



Gambar 8. Diagram alir algoritma penjumlahan dan pengurangan

2) *Algoritma Perkalian*

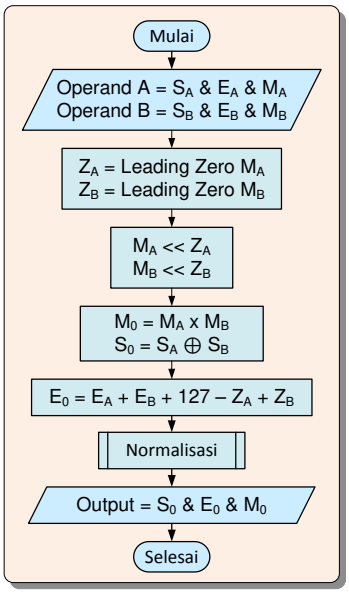
Algoritma yang digunakan untuk mengalikan dua buah bilangan *floating point* bila dijabarkan dalam bentuk diagram alir ditunjukkan dalam Gambar 9.



Gambar 9.Diagram alir untuk algoritma perhitungan perkalian.

3) *Algoritma Pembagian*

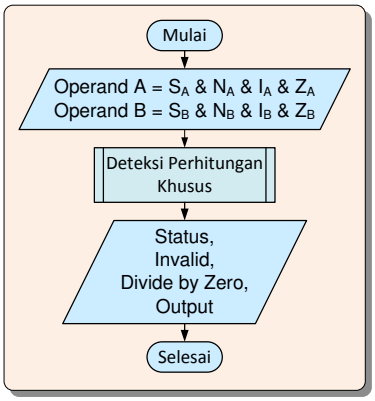
Algoritma yang digunakan untuk melakukan pembagian dua buah bilangan *floating point* bila dijabarkan dalam bentuk diagram alir ditunjukkan dalam Gambar 10.



Gambar 10. Diagram alir untuk algoritma perhitungan pembagian

4) *Algoritma Exception Handler*

Secara singkat dan sederhana, penjabaran algoritma *exception handler* dalam bentuk diagram alir ditunjukkan dalam Gambar 11.



Gambar 11.Diagram alir algoritma *exception handler*.

Perhitungan-perhitungan khusus yang dideteksi oleh algoritma *exception handler* antara lain:

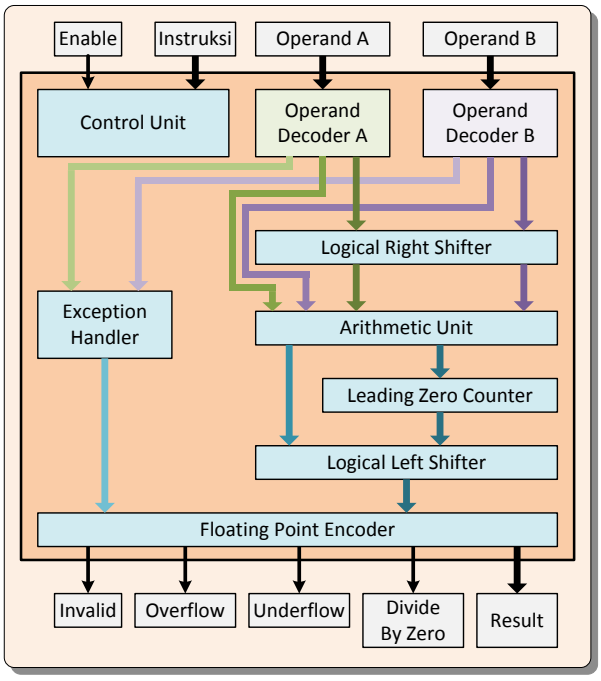
1) Salah satu masukan adalah *Not a Number*, maka keluaran perhitungan adalah *Not a Number*.
2) Nol dikali dengan *Infinity*, Maka keluaran perhitungan adalah *Not a Number.*
3) Nol dibagi dengan Nol, maka keluaran perhitungan adalah *Not a Number*.
4) Bukan Nol dibagi dengan Nol, maka keluarannya adalah *Infinity* dan keluaran *Sign* akan bernilai 1jika *sign* salah satu atau kedua masukan bernilai 1.
5) *Infinity* dibagi dengan *Infinity*, maka keluaran perhitungan adalah *Not a Number.*
6) Bukan *Infinity* dibagi dengan *Infinity*, maka keluaran perhitungan adalah Nol dan keluaran *Sign* akan bernilai 1 jika S*ign* salah satu atau kedua masukan bernilai 1.
7) Positif *Infinity* ditambah negatif *Infinity*, maka keluaran perhitungan adalah *Not A Number*.
8) Positif *Infinity* dikurangi positif *Infinity*, atau negatif *Infinity* dikurangi negatif *Infinity*, maka keluaran perhitungan adalah *Not A Number*.
9) Positif *Infinity* ditambah positif *Infinity* atau positif *Infinity* dikurangi negatif *Infinity*, maka keluaran perhitungan adalah positif *Infinity*.
10) Negatif *Infinity* dikurangi positif *Infinity* atau negatif *Infinity* ditambah negatif *Infinity*, maka keluaran hasil perhitungan adalah negatif *Infinity*.

D. Perancangan Komponen FPU

Berdasarkan fungsi-fungsi algoritma yang telah ditetapkan. Maka sistem FPU yang dirancang tersusun dari blok-blok komponen penyusun FPU antara lain:

1) Unit *OperandDecoder*
2) *Arithmetic Unit*
3) Unit *Leading Zero Counter*
4) Unit *Logical Right Shifter*
5) Unit *Logical Left Shifter*
6) Unit *Exception Handler*
7) Unit *Floating Point Encoder*
8) *Control Unit*

Susunan komponen-komponen FPU secara keseluruhan ditunjukkan dalam Gambar 12.



Gambar 12. Blok diagram FPU beserta komponen-komponen penyusun FPU.

IV. PENGUJIAN DAN ANALISIS

Pengujian dilakukan dengan menggunakan peralatan pendukung antara lain:

1) FPGA Xilinx 3E-500 FG320.
2) ZIFI-FPU Simulator sebagai unit input-output tambahan.
3) Komputer dengan dukungan USB 2.0 dengan daya 500 mA.
4) *Software* Xilinx ISE Project Navigator 11.1 untuk pemrogram.
5) *Software* Digilent Adept untuk antarmuka.

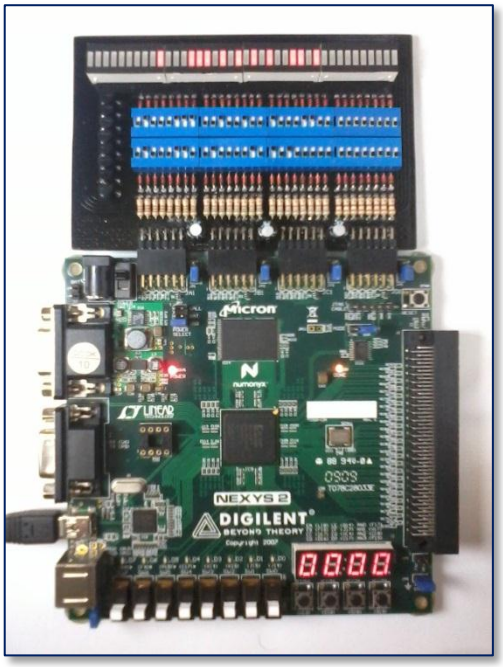
Data hasil pengujian yang telah dilakukan ditampilkan pada Tabel 1.

TABEL 1
DATA HASIL PENGUJIAN

No	Instr	Op-A	Op-B	Result	Flag
1.	00	00000000	0000000	00000000	0000
2.	00	47248C00	4642A000	47553400	0000
3.	00	863A1788	000211A4	863A1745	0000
4.	00	7FEDCBA9	10630055	FFFFFFFF	1000
5.	01	4E32D05E	4B3EBC20	4E2FD56D	0000
6.	01	7F800000	7F800000	FFFFFFFF	1000
7.	01	40A00000	C0E00000	41400000	0000
8.	01	3F800000	36EAE18A	3F7FFF8A	0000
9.	10	41800000	41A80000	43A80000	0000
10.	10	44F34000	3A102DE0	3F88FF97	0000
11.	10	000021F1	0A000028	00000000	0001
12.	10	7F800000	00000000	FFFFFFFF	1000
13.	11	40400000	40A00000	3F199999	0000
14.	11	42F80000	0198724D	3F800001	0010
15.	11	4D6E6B28	42200000	4ABEBC20	0000
16.	11	C4FBA000	00000000	FF800000	0100

Keterangan:

- Instr menunjukkan masukan instruksi yang disajikan dalam bentuk biner.
- Flag terdiri dari keluaran *Invalid, Divide by zero, Infinity*, dan *Underflow*.



Gambar 13. Hasil pengujian ke-2

Pada pengujian kelimabelas, FPU menghitung bilangan `4D6E6B28h/42200000h` atau setara dengan perhitungan 250000000/40 dan hasil dari FPU adalah `4ABEBC20h` atau setara dengan nilai 6250000.
Dalam pengujian yang dilakukan, FPU telah bekerja dengan benar dengan memberikan keluaran yang diharapkan dan tidak memberikan hasil *error*.

V. PENUTUP

A. *KESIMPULAN*

Berdasarkan pengujian dan analisis yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut:

1) Sistem FPU yang dapat diimplementasikan dengan menggunakan FPGA adalah FPU yang dapat mengoperasikan dua masukan *operand* dalam bentuk bilangan *floating point* presisi tunggal standar IEEE 754. Operasi perhitungan dapat dipilih dengan memberi masukan tertentu.
2) Sistem FPU dapat melakukan empat perhitungan dasar dari dua buah bilangan *floating point* presisi tunggal standar IEEE 754. Perhitungan yang dapat dilakukan yaitu penjumlahan, pengurangan, perkalian dan pembagian. Sistem FPU juga dapat menangani perhitungan-perhitungan khusus yang tidak bisa diselesaikan oleh hardware perhitungan standar.
3) Sistem FPU dapat diaplikasikan sebagai *co-processor* dengan cara menggabungkan sistem FPU dan sistem lain.
4) Sistem FPU menggunakan blok-blok yang telah dirancang khusus untuk membangun FPU, karena itu masing-masing blok hanya bekerja satu kali setiap proses FPU dan dapat bekerja secara paralel antar satu sama lain.
5) Dari proses *synthesize* yang dilakukan, besar delay total adalah 233.689ns, artinya FPU dapat bekerja pada frekuensi maksimal 4279191Hz.

B. *SARAN*

Saran yang dapat diberikan dari penelitian perancangan sistem FPU ini antara lain:

1) Penambahan proses aritmatika yang dapat dikerjakan oleh FPU, misalnya akar kuadrat, Logaritma, dll.
2) Penambahan converter bilangan integer ke bilangan *floating point* dan sebaliknya.

Perbaikan sistem aritmatika agar kecepatan perhitungan FPU dapat lebih tinggi.

REFERENSI

[1] Muller, Jean-Michel dkk. 2010. *Handbook of Floating-Point Arithmetic*. Boston: Birkhäuser
[2] Muller, Jean-Michel dkk. 2010. *Handbook of Floating-Point Arithmetic*. Boston: Birkhäuser.
[3] Overton, Michael L. 2001. *Numerical Computing with IEEE Floating Point Arithmetic*. New York: SIAM.
[4] Deschamps, Jean-Pierre dkk. 2006. *Synthesis of Arithmetic Circuit: FPGA, ASIC, and Embedded Systems*. New Jersey: John Wiley & Sons, Inc.
[5] Kilts, Steve. 2007. *Advanced FPGA Design Architecture, Implementation, and Optimization*. New York: John Wiley & Sons, Inc.
[6] Navabi, Zainalabedin. 2005. *Digital Design and Implementation with Field Programmable Devices*. New York: Kluwer Academic Publishers.
[7] Overton, Michael L. 2001. *Numerical Computing with IEEE Floating Point Arithmetic*. New York: SIAM.
[8] Perry, Douglas L. 2002. *VHDL: Programming by Example*. New York: McGraw-Hill.
[9] Sass, Ronald dkk. 2010. *Embedded Systems Design with Platform FPGAs: Principles and Practices*. Amsterdam: Morgan Kaufmann Publishers.
[10] Xilinx, Inc. 2009. *Constraints Guide*. California: Xilinx, Inc.
[11] _____. 2009. *ISE In-Depth Tutorial*. California: Xilinx, Inc.
[12] _____. 2009. *Spartan-3E Libraries Guide for HDL Designs*. California: Xilinx, Inc.
[13] _____. 2009. *XST User Guide*. California: Xilinx, Inc.