SENATIK 2020, Vol. VI, ISBN 978-602-52742-2-0 DOI: 10.28989/senatik.v6i0.423

Rancang bangun *Arithmetic Logic Unit* 8 bit pada Spartan 2 *field programmable gate array*

Denny Dermawan*, Maesa Agny Manggala Putra, Catur Budi Waluyo, Bambang Sudibya
Program Studi Teknik Elektro Sekolah Tinggi Teknologi Adisudjipto

Email Korespondensi: *dennydermawanstta@gmail.com

Abstract. *As we know, digital systems have been used in everyday life or in industry today because they are more useful than analog systems. Because it is important to develop digital systems, many new digital devices have been designed in complex ways. Some of these devices are called microprocessors, microcontrollers or microchips. It is very important to have very high speed performance in all devices. In this study, a tool was designed, namely, the Arichmetic Logic Unit or it can be called ALU. An important part of Field Programmable Gate Arrays or also known as FPGA, ALU generally has functions to perform arithmetic and logic calculations. Based on the results of the testing of the tools that have been carried out, it can be concluded that the design of the Arichmetic Logic Unit on the Spartan 2 FPGA. with the arithmetic operations performed are the ADDER (A + B) and SUBTRACTOR (A - B) operations, the logical operations performed are the OR (A OR B) and AND (A AND B) operations. It has been simulated and implemented with results that match the specifications of the design.*
Keywords: *Arichmetic Logic Unit* 8 bit, *Adder, Subtractor, AND, OR, Spartan 2 FPGA*.

1. Pendahuluan

Seperti yang kita ketahui, sistem digital telah digunakan dalam kehidupan sehari-hari atau dalam bidang industri saat ini karena lebih bermaanfaat dibandingkan dengan sistem analog. Karena penting mengembangkan sistem digital, banyak prangkat digital yang baru yang telah di desain secara kompleks. Beberapa perangkat yang disebut mikroprosesor, mikrokontroler atau *microchip*. Hal ini sangat penting untuk memiliki kinerja kecepatan yang sangat tinggi di semua perangkat.

Multiplexer adalah salah satu bagian yang paling penting dalam perangkat yang dapat mempengaruhi kinerja perangkat. Jadi, kecepatan tinggi dan sistem *Multiplexer* yang efisien adalah faktor penting bagi para perancang perangkat mikroprosesor, mikrokontroler dan lain-lain. Seperti yang kita ketahui, operasi

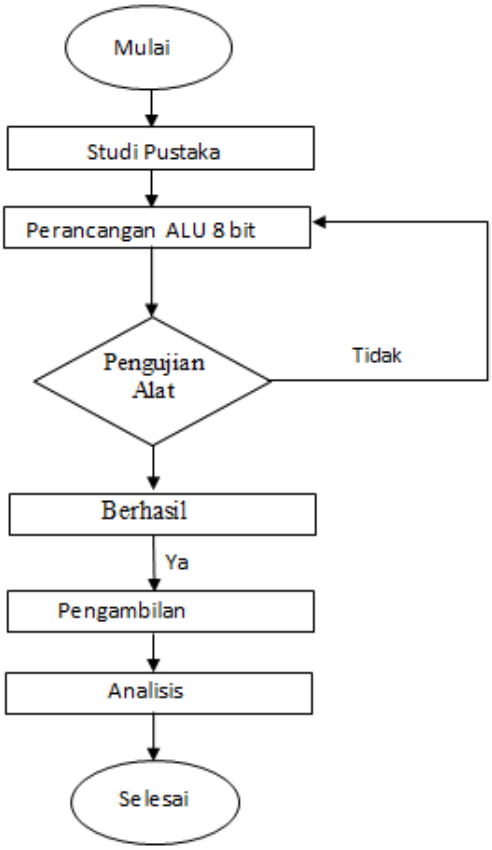
perkalian tidak sulit untuk dilakukan di angka desimal. Tapi, untuk melakukan operasi dalam bilangan biner (yang digunakan dalam sistem digital) adalah operasi yang sangat kompleks. Percepatan segnifikan dalam perhitungan waktu dapat dicapai menetapkan secara intensif proses intensif tugas proses perhitungan dengan menggunakan pernagkat keras dan memanfaatkan proses paralel dalam algoritma. *Field Programmable Gate Arrays* (FPGA) telah muncul sebagai platform pilihan untuk implmentasi *hardware* yang efisien.

FPGA adalah (*Field Programmable Gate Arrays)* sebuah *Intergrated Circuit* yang didesain untuk dapat dikonfigurasi oleh *user* atau *designer* setelah keluar dari produksi. Pengkonfigurasi FPGA pada umumnya adalah spesifik menggunakan deskripsi bahasa *hardware* atau HDL (*Hardware Descripsion Language*). *Field Programmable Gate Arrays* (FPGA) memungkinkan tingkat paralisme yang tinggi sehingga dapat meningkatkan sumber daya yang tertanam yang tersedia pada FPGA. FPGA mendapatkan manfaat dari kecepatan *hardware* dan fleksibelitas perangkat lunak. Tiga faktor utama yang mempunyai peran penting dalam desain FPGA adalah arsitektur FPGA yang digunakan, perangkat *Eelectronic Design Automation* (EDA) dan *desaign* teknik yang digunakan pada tingkat algoritma yang menggunakan *Hardware Description Language* (HDL).

Arithmetic Logic Unit (ALU) adalah inti dari implentasi dalam *field programmable gate arrays* (FPGA) yang mempunyai andil yang cukup besar dalam penentuan gerbang, *Arithmetic Logic Unit* (ALU) menampilkan operasi penjumlahan, pengurangan, perkalian integer, logika AND, OR, NOT, NOR, NAND, XOR, XNOR dan operasi boolean yang lain.
Oleh karena itu penelitian ini akan mendesain dan mensimulasikan ALU 8 bit pada Spartan 2 FPGA. Berdasarkan paparan latar belakang di atas, maka penulis akan meneliti mengenai Rancang Bangun *Arithmetic Logic Unit* pada Spartan 2 FPGA.

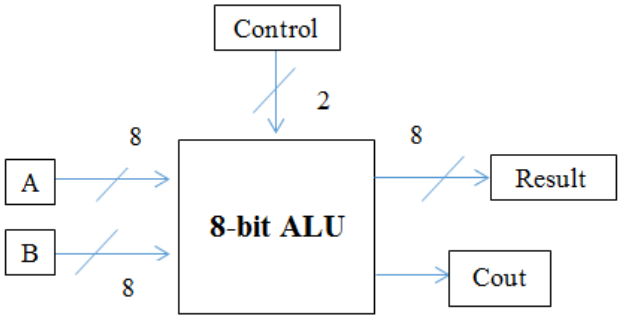
2. Metode penelitian

Diagram alir penelitian diperlihatkan pada Gambar 1 merupakan proses penelitian yang dilakukan dari mulainya penelitian hingga pengumpulan data dan penulisan pada laporan penelitian. Diagramm alir penelitian berfungsi untuk merinci dengan detail setiap kegiatan penelitian yang akan dilakukan, sehingga dengan adanya diagram alir ini diharapkan kegiatan penelitian dapat diketahui dengan rinci dan jelas.



Gambar 1. Diagram alir penelitian

Blok diagram dari penelitian ini ada pada Gambar 2. Data yang akan dianalisis didapat pada masing-masing blok diagram, untuk memahami dasar-dasar rangkaian dari gerbang logika AND, OR, NOT dan XOR yang sederhana. Blok diagram ini akan mengetahui cara membuat *Arithmetic Logic Unit* (ALU) dari awal, menggunakan gerbang logika sederhana dan komponen lainnya. ALU akan mengambil dua nilai dari 8 bit, dan 2 garis kontrol. Tergantung pada nilai garis kontrol, *Outputnya* adalah penambahan, pengurangan, *Bitwise* AND atau *Bitwise OR dari Input.*

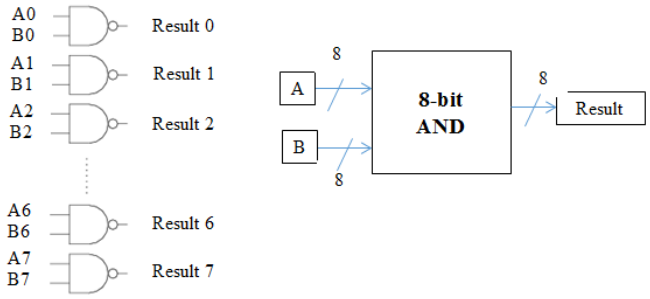


Gambar 2. Blok diagram ALU 8 bit

Pada Gambar 2 Blok diagram perancangan ALU 8 bit, dimana terdapat masukan A dan B dan hasil keluaran *Cout*. Beberapa data dan garis kontrol di tampilkan dengan garis miring dan angka 8, garis tersebut sebenarnya garis parallel, hasilnya adalah 8 bit.

2.1 AND 8 bit

Pada penelitian ini akan dirancang sekema *Bitwise* AND 8 bit. Dengan dua masukan karena setiap logika AND bitnya tersendiri, jadi perlu membuat 8 rangkaian gerbang AND secara paralell. Dapat dilihat pada Gambar 3.

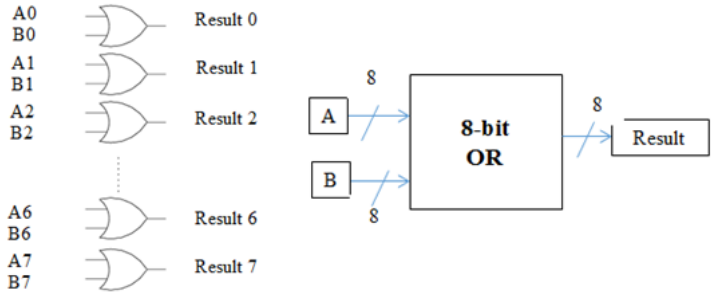


Gambar 3 Blok diagram *Bitwise* AND 8 bit dan rangkaian gerbang AND

Pada Gambar 3 terdapat 8 bit logika AND *Inputnya* adalah A0, B0 sampai A7, B7 dan *Outputnya* R0 sampai R7. Dan setiap logika AND adalah 1 bit. Maka di paralellkan menjadi 8 bit.

2.2 OR 8 bit

Untuk membuat sekema logika 8 bit OR yang mempunyai dua masukan, sama halnya dengan logika AND. Dapat dilihat pada Gambar 4.

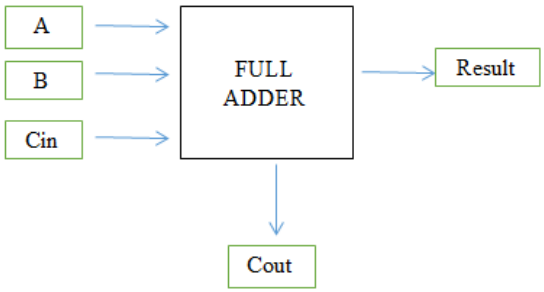


Gambar 4. Blok diagram OR 8 bit

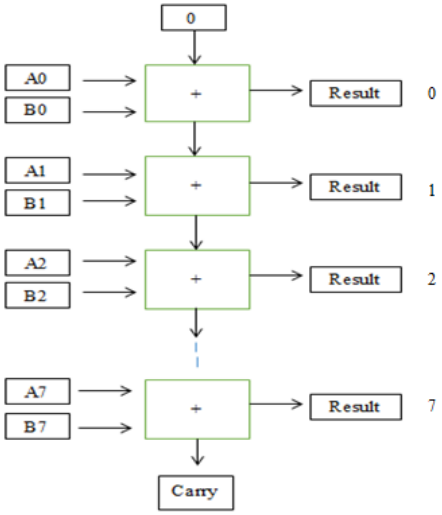
Pada Gambar 4 terdapat 8 bit logika OR *Input-nya* adalah A0, B0 sampai A7, B7 dan *Output-nya* R0 sampai R7. Dan setiap logika OR adalah 1 bit, Maka diparalelkan menjadi 8 bit.

2.3 *ADDER* 8 bit

Pada Gambar 5 harus menambahkan bit, dan akan menghasilkan *Carry* sebagai *Input* bersama dengan dua masukan A dan B. Akan menghasilkan *C-out* 1 bit. ADD 1 bit menerima *C-in* menghasilkan keluaran1 bit dan sebuah eksekusi disebut *Full Adder*. Dapat dilihat pada Gambar 5 *Full Adder*.



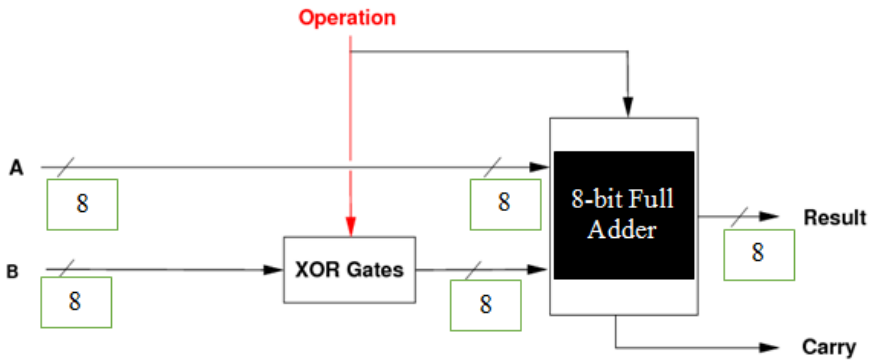
Gambar 5. Blok diagram *full adder*



Gambar 6. Blok diagram *ADDER* 8 bit

Pada Gambar 6 *full Adder* keluaran terakhir adalah 1, menunjukkan bahwa hasilnya terlalu besar untuk dimasukkan ke dalam 8 bit. Untuk menunda diagram penjumlah 8 bit dapat merubahnya dengan melakukan pengurangan.

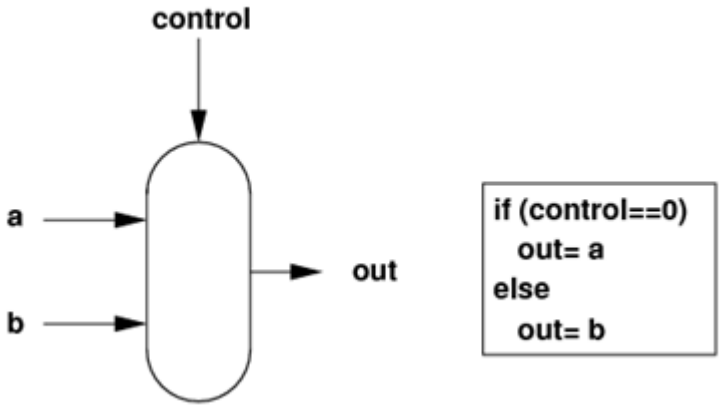
2.4 *ADDER* dan *SUBTRACTOR* 8 bit



Gambar 7. Blok diagram penjumlah dan pengurang

Gambar 7 adalah Blok diagram penjumlah dan pengurang 8 bit , jika bit operasi adalah 0, maka A + B. Jika bit operasi adalah 1, maka A + ~ B + 1. Untuk membedakan antara garis data (yang melewatkan data) dan garis kontrol (yang mengontrol tindakan komponen). Garis data juga dikenal sebagai data *Path*. Masukan dari data operasi pada Gambar 7 adalah garis kontrol, karena mengontrol tindakan yang sedang berjalan.

2.5 *Multiplexer*

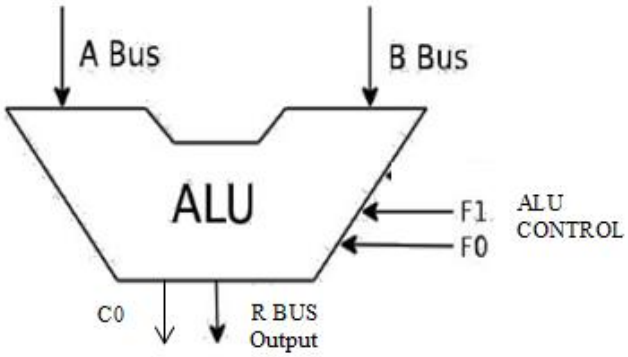


Gambar 8. Blok diagram *Multiplexer*

Multiplexer pada Gambar 8 adalah *Multiplexer* 2 arah dengan lebar 1 bit 2 *inpu*t, masing-masing lebar 1 bit. Jika menambahkan jalur kontrol lebih, dapat memilih lebih banyak *input* 2 jalur kontrol digunakan dalam *Multiplexer* 4 arah, 3 jalur kontrol dalam *Multiplexe*r 8 arah. Dan dengan menggunakan beberapa *Multiplexer* secara parallel, dapat membuat *Multiplexer* banyak bit.

2.6 *Arithmetic Logic Unit* 8 bit

ALU 8 bit memerlukan 2 buah *input* BUS 8 bit dan 1 buah *output* 8 bit. 3 buah masukan untuk *Control* ALU *Operation*, sedangkan untuk operasi ALU sendiri memerlukan 3 buah masukan *Control* untuk memilih operasi yang ada. Untuk blok diagram dari sebuah rancangan ALU 8 bit dapat dilihat pada Gambar 9.



Gambar 9. Blok diagram *Arithmetic Logic Unit* 8 bit

Berdasarkan Gambar 9 Blok diagram ALU 8 bit, mempunyai 2 masukan yang bernilai 8 bit yaitu masukan A dan B. Sedangkan untuk operasi *Control* ALU mempunyai masukan F0, F1 dan menjadi 3 masukan. *Control* memiliki fungsi untuk memilih 1 sampai 4 fungsi logika yang terdapat di dalam ALU. Dalam pemilihan 1 sampai 4 fungsi logika dapat menggunakan *Multiplexer*, dari masukan 4 menjadi 1 masukan. Masukan dari 4 fungsi pada ALU bisa di lihat pada Tabel 1.

Tabel 1. *Control* ALU

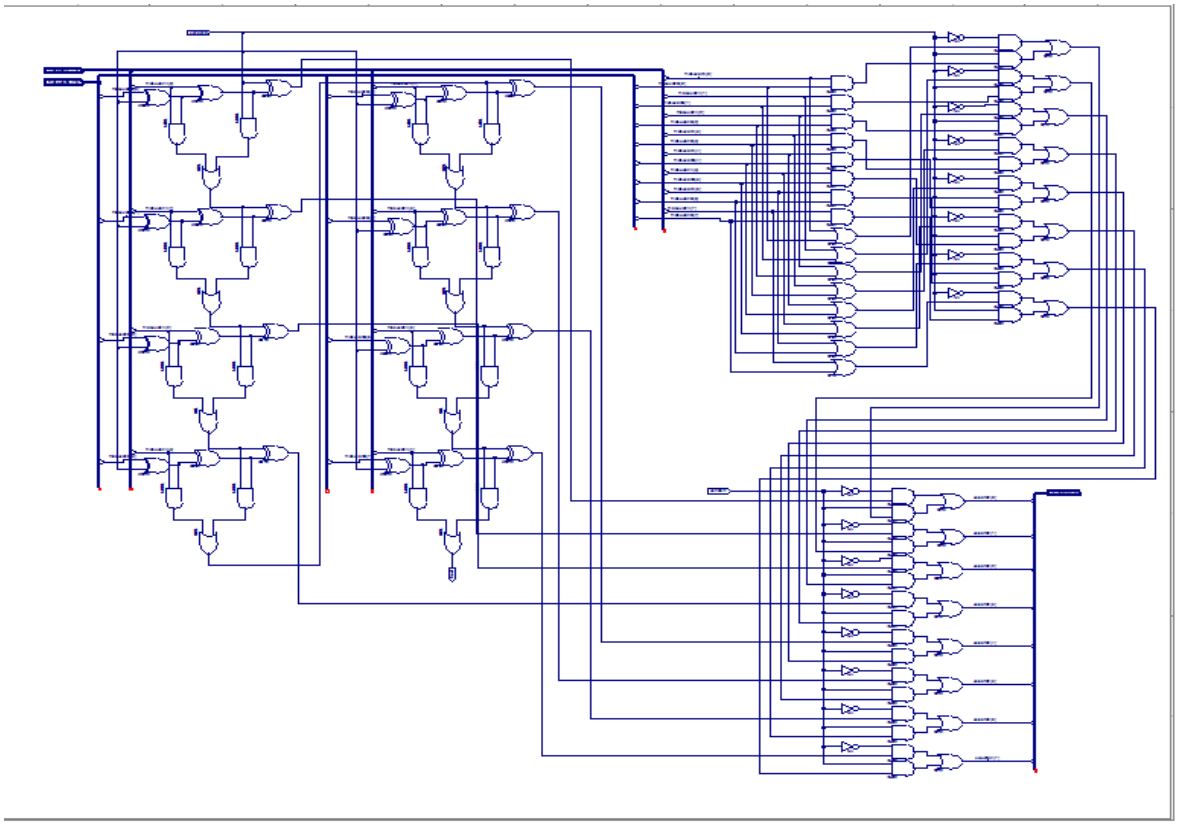
Input		Output
F1	F0	
0	0	**A+B**
0	1	**A-B**
1	0	**A OR B**
1	1	**A AND B**

Seperti yang dijelaskan pada Tabel 1 di atas ALU memerlukan masukan untuk operasi *Control* yang berupa 2 masukan F0 dan F1, masukan biner F0 + F1 akan menghasilkan operasi nilai awal A + B, masukan biner F0 - F1 akan menghasilkan operasi dari nilai awal A - B, sedangkan masukan biner bernilai OR akan menghasilkan opersi logika A OR B, dan masukan biner bernilai AND akan meghasilkan operasi logika A AND B.

3. Hasil dan Pembahasan

3.1 Hasil simulasi rangkaian ALU (*Arithmetic Logic Unit*) 8 bit

Pada bagian ini akan membahas tentang hasil simulasi ALU 8 bit. Untuk membuat rangkaian ALU 8 bit membutuhkan rangkaian rangkaian seperti *ADDER* dan *SUBTRACTOR*, *multiplexer*. Masukan dari sinyal A dan B yang masing - masing memiliki lebar 8 dengan nama sinyal A(7:0) yang terdiri dari A(0), A(1), A(2), A(3), A(4), A(5), A(6), A(7) dan sinyal B(7:0) yang terdiri dari B(0), B(1), B(2), B(3), B(4), B(5), B(6), B(7), sedangkan sinyal keluaran adalah sinyal Y(7:0) yang terdiri dari Y(0), Y(1), Y(2), Y(3), Y(4), Y(5), Y(6), Y(7). Rangkaian skematik ALU 8 bit diperlihatkan pada Gambar 10.



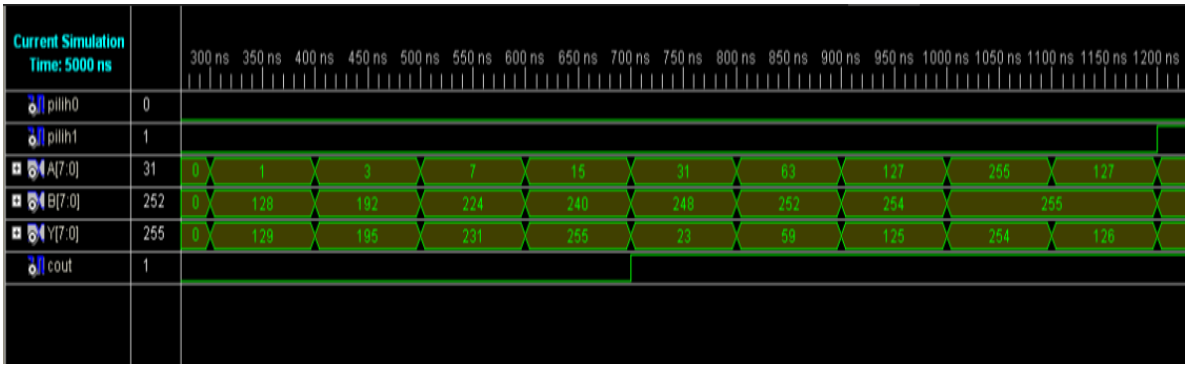
Gambar 10. Rangkaian skematik ALU 8 bit

ALU 8 bit pada perancangan ini memiliki 2 buah fungsi aritmatika yaitu ADD dan SUB dan 2 buah fungsi logika yaitu AND dan OR dengan memilih kombinasi pin kendali/kontrol F1 dan F0 seperti diperlihatkan pada Tabel 2.

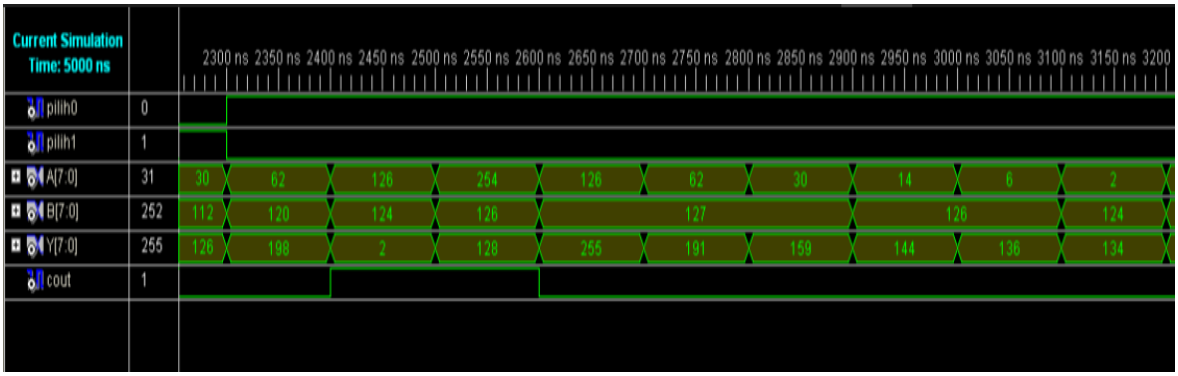
Tabel 2. Hasil Simulasi ALU 8 bit

No	Current simulation time(nS)	A(7:0) dec	B(7:0) dec	Y(7:0) dec	Fungsi	P1	P0	C out
1	300-400	1	128	129	A+B	0	0	0
2	400-500	3	192	195	A+B	0	0	0
3	500-600	7	224	231	A+B	0	0	0
4	2300-2400	62	120	198	A-B	1	1	0
5	2400-2500	126	124	2	A-B	1	1	1
6	2500-2600	254	126	128	A-B	1	1	1
7	1400-1500	0000 0111	1111 1000	1111 1111	A OR B	0	0	
8	1500-1600	0000 0111	1111 0000	1111 0111	A OR B	0	0	
9	1600-1700	0000 0011	1110 1111	1110 0011	A OR B	0	0	
10	3900-4000	1111 1111	1110 1100	0110 1100	A AND B	1	1	
11	4000-4100	1110 0001	1110 1000	1110 1000	A AND B	1	1	
12	4100-4200	1110 0001	1110 1000	1110 1010	A AND B	1	1	

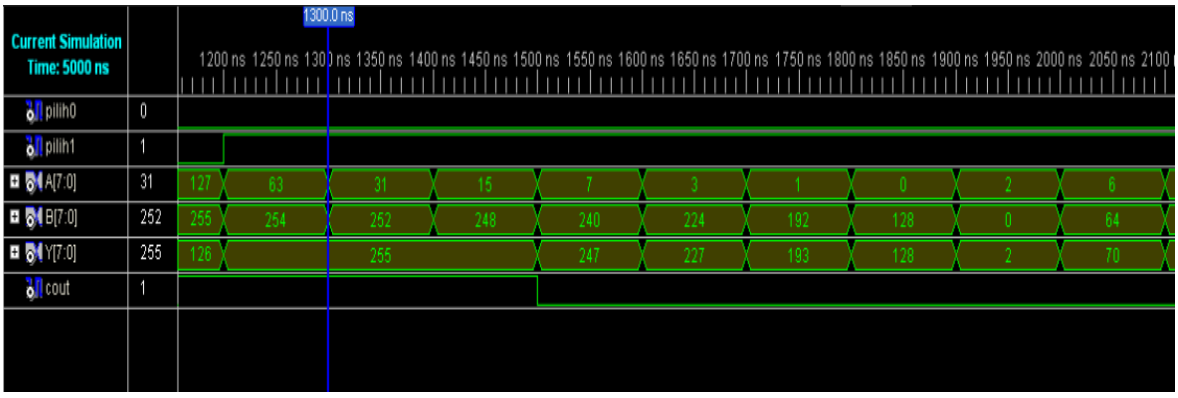
Hasil simulasi ALU 8 bit diperlihatkan pada Gambar 11, 12, 13 dan 14. Pada Gambar 11, 12, 13 dan 14 hasil simulasi ALU 8 bit sudah sesuai dengan tabel 2 Sinyal uji masukan untuk gerbang ALU 8 bit, sehingga dapat dikatakan bahwa rangkaian skematik ALU 8 bit, sudah teruji dan menunujukkan hasil yang benar. Hasil pada baris ketujuh sampai baris ke dua belas menggunakan bilangan logika OR dan AND.



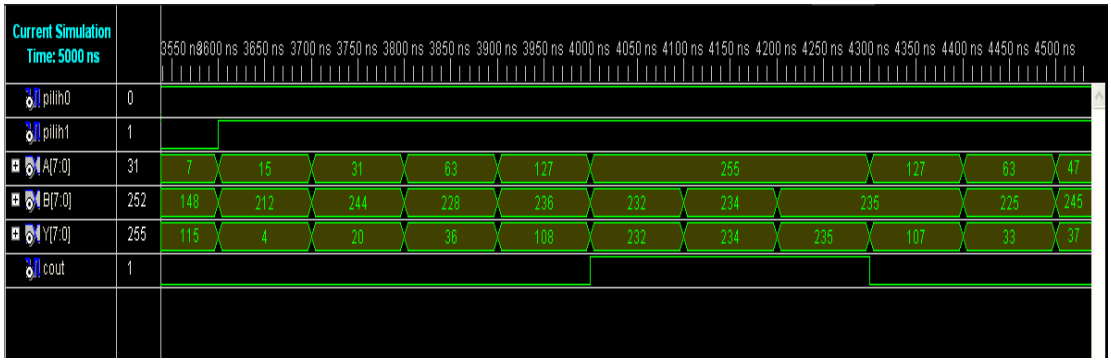
Gambar 11. Hasil simulasi sinyal uji ALU 8 bit A+B



Gambar 12. Hasil simulasi sinyal uji ALU 8 bit A-B



Gambar 13. Hasil simulasi sinyal uji ALU 8 bit A OR B



Gambar 14. Hasil simulasi sinyal uji ALU 8 bit A AND B

3.2 Hasil implementasi rangkaian ALU (*Arithmetic Logic Unit*) 8 bit

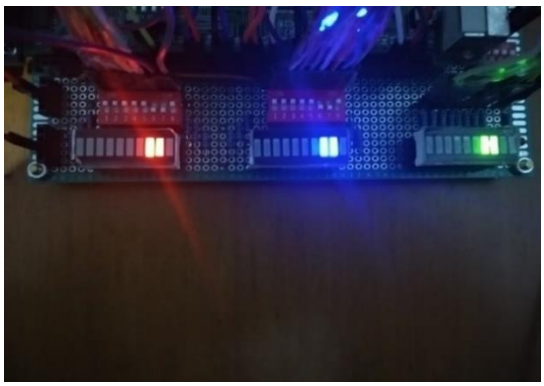
3.2.1 Hasil Rancangan ALU 8 bit *Mode ADDER* (A+B)

Hasil implementasi ALU 8 bit *Mode ADDER* (A+B) diperlihatkan pada Tabel 3. Sinyal masukan adalah A (7:0) dan B (7:0) sedangkan sinyal keluaran adalah Y (7:0) dan *C-out*. Hasil implementasi *Mode ADDER* (A+B) sudah diuji dan menunujukkan hasil yang benar.

Tabel 3. *Mode ADDER* (A+B)

No	A(7:0)		B(7:0)		Y(7:0)		Cout
	Biner	Dec	Biner	Dec	Biner	Dec	
1	0000 0110	6	0000 0110	6	0000 1100	12	0

Hasil implementasi *mode* ADD (A+B) dari pada tabel 3 diperlihatkan pada Gambar 15. Lampu indikator warna merah (kiri) adalah masukan A, lampu indikator warna biru (tengah) adalah masukan B, dan lampu indikator warna hijau (kanan) adalah keluaran Y.



Gambar 15. Hasil Implementasi tabel 4.7 Mode A+B baris pertama

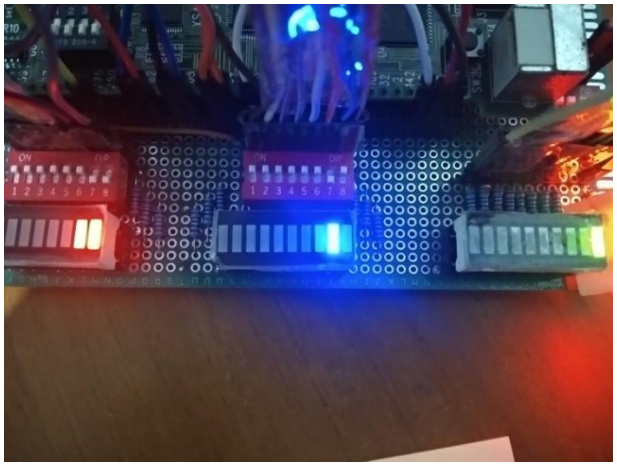
3.2.2 Hasil Rancangan ALU 8 bit *Mode SUBTRACTOR* (A-B)

Hasil implementasi ALU 8 bit *Mode SUBTRACTOR* (A-B) diperlihatkan pada Tabel 4. Sinyal masukan adalah A (7:0) dan B (7:0) sedangkan sinyal keluaran adalah Y (7:0) dan *C-out*. Hasil implementasi *mode SUBTRACTOR* (A-B) sudah diuji dan menunjukkan hasil yang benar.

Tabel 4. *Mode* A-B

No	A(7:0)		B(7:0)		Y(7:0)		Cout
	Biner	Dec	Biner	Dec	Biner	Dec	
1	0000 0011	3	0000 0010	2	0000 0001	1	1

Hasil implementasi *mode* SUB (A-B) dari Tabel 4 diperlihatkan pada Gambar 16. Lampu indikator warna merah (kiri) adalah masukan A, lampu indikator warna biru (tengah) adalah masukan B, dan lampu indikator warna hijau (kanan) adalah keluaran Y.



Gambar 16. Hasil implementasi tabel 4 *Mode* A-B baris pertama

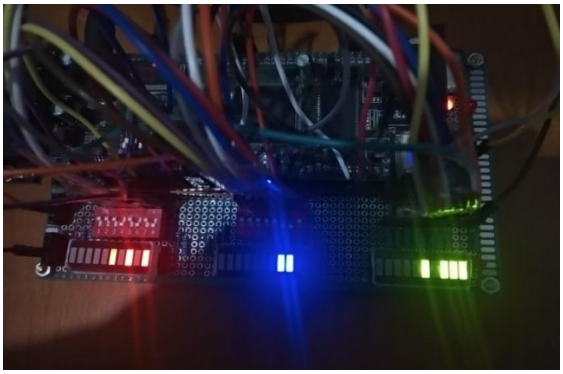
3.2.3 Hasil Rancangan ALU 8 bit *Mode* OR

Hasil simulasi ALU 8 bit *Mode* OR diperlihatkan pada Tabel 5. Sinyal masukan adalah A (7:0) dan B (7:0) sedangkan sinyal keluaran adalah Y (7:0) dan *C-out*. Hasil implementasi *mode* OR sudah diuji dan menunujukkan hasil yang benar.

Tabel 5. *Mode* A OR B

No	Masukan		Keluaran
	A(7:0)	B(7:0)	Y(7:0)
1	0001 0101	0000 0110	0001 0111

Tabel 5 menunjukkan bahwa hasil keluaran pada Y(7:0) merupakan hasil operasi OR dari masukan A(7:0) dan B(7:0), tidak ada kesalahan yg muncul sehingga dapat dikatakan hasil implementasi ALU pada *mode* OR sudah berhasil. Hasil implementasi *mode* OR (AOR B) dari Tabel 5 diperlihatkan pada Gambar 17. Lampu indikator warna merah (kiri) adalah masukan A, lampu indikator warna biru (tengah) adalah masukan B, dan lampu indikator warna hijau (kanan) adalah keluaran Y.



Gambar 17. Hasil implementasi tabel 5 *Mode* A OR B baris pertama

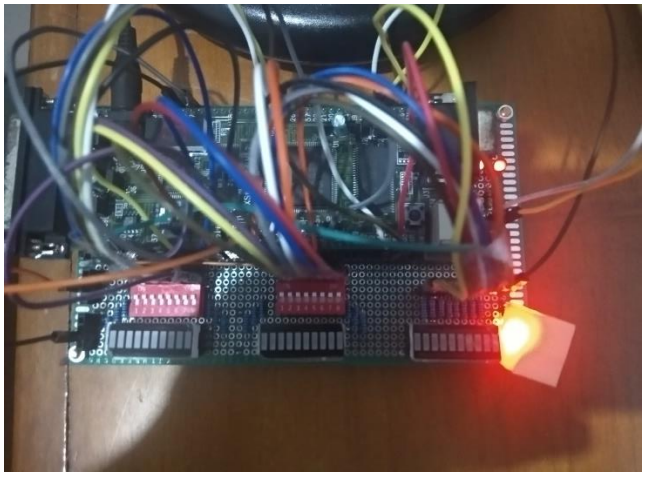
3.2.4 Hasil Rancangan ALU 8 bit *Mode* AND

Hasil simulasi ALU 8 bit *mode* AND diperlihatkan pada Tabel 6. Sinyal masukan adalah A (7:0) dan B (7:0) sedangkan sinyal keluaran adalah Y (7:0) dan *Cout*. Hasil implementasi *mode* AND sudah diuji dan menunujukkan hasil yang benar.

Tabel 6. *Mode* A AND B

No	Masukan		Keluaran
	A(7:0)	B(7:0)	Y(7:0)
1	0000 0000	0000 0000	0000 0000

Tabel 6 menunjukkan bahwa hasil keluaran pada Y(7:0) merupakan hasil operasi AND dari masukan A(7:0) dan B(7:0), tidak ada kesalahan yg muncul sehingga dapat dikatakan hasil implementasi ALU pada *mode* AND sudah berhasil. Hasil implementasi *mode* AND (A AND B) dari Tabel 6 diperlihatkan pada Gambar 18. Lampu indikator warna merah (kiri) adalah masukan A, lampu indikator warna biru (tengah) adalah masukan B, dan lampu indikator warna hijau (kanan) adalah keluaran Y.



Gambar 18. Hasil implementasi tabel 6 *Mode* A AND B baris pertama

4. Kesimpulan

Berdasarkan hasil perancangan *Arithmetic Logic Unit* 8 bit Spartan 2 FPGA, maka dihasilkan kesimpulan sebagai berikut adalah.

1. *Arithmetic Logic Unit* 8 bit Spartan 2 FPGA dengan menggunakan Xilinx ISE 10.1 telah dapat di implementasikan pada FPGA dan hasil keluaran telah sesuai dengan spesifikasi hasil rancangan.
2. Operasi Aritmatika yang dilakukan adalah operasi *ADDER* (A + B) dan *SUBTRACTOR* (A – B), operasi logika yang dilakukan adalah operasi OR (A OR B) dan AND (A AND B).
3. Pada operasi *ADDER* (A + B), apabila *Cout* aktif menunjukkan bahwa hasil operasi *ADDER* melebihi kapasitas ALU 8 bit (256_{10}).
4. Pada operasi *SUBTRACTOR* (A – B), hasil operasi adalah merupakan bilangan komplemen 2 apabila nilai yang dikurangi (A) lebih kecil dari nilai pengurang (B).

5. Daftar pustaka

[1] Rizki Jumadil Putra (2013) “Implementasi *Filter* Digital FIR (*Finite Impulse Response*) Pada *Field Programmable Gate Arrays* (FPGA)”.

[2] L.Hermanto (2012) “Implementasi Serial *Multiplexer* 8 Bit Ke Dalam IC FPGA Sebagai Pendukung Percepatan Operasi Perkalian Dalam Kompresi Citra”.

[3] Adi Setiawan (2015) “Pengali Pada SPARTAN - 2 FPGA Sebagai Pendukung Tapis Digital Pada *Radio Detection Finder* (RDF)”.

[4] Ashari, Mochamad. 2012. Sistem Konverter DC (Desain Rangkaian Elektronika Daya). Surabaya: ITSpress.

[5] Dermawan D. dkk, 2018, “Implementasi pengali pada Spartan 2 FPGA sebagai pendukung tapis digital pada Radio *Direction Finding* (RDF)”, Prosiding Nasional : Rekayasa Teknologi Industri dan Informasi (ReTII) ke-14; STTNAS Yogyakarta; ISSN 1907-5995; Hal 296 – 305; 4 November 2019.

[6] Lakadiwala U et al, 2016,” *Implementation of* ALU *on* FPGA”, *International Research Journal of Engineering and Technology* (IRJET)Vol 3. Issue 4.

[7] M. Morris Mano, “*Digital Design*” (3rd Edition), *Prentice Hall*.

[8] M. Morris Mano & C. Kime, “Logic and Computer Design Fundamentals” Prentice Hall.

[9] Kevin Skahill, “VHDL for Programmable Logic”, Addison Wesley & Sons.
[10] Stefan Sjoholm & L. Lindh, “VHDL for Designers” Prentice Hall.
[11] Swamynathan S.M, Banuvathi V., 2017,”Design and Analysis of FPGA based on 32 bit ALU using reversible gates”, IEEE International conference on Electrical, Instrumentation and Communication Engineering.
[12] Widodo Budihartato dan Sigit Firmansyah. 2005. “Elektronika Digital dan Mikroprosesor”. Yogyakarta: ANDI.
[13] Xilinx FPGA Ise Webpack 10.1 Tutorial.