4. ALU

4.1. ALU (*Arithmetic and Logic Unit*)

Unit Aritmetika dan Logika merupakan bagian pengolah bilangan dari sebuah komputer. Di dalam operasi aritmetika ini sendiri terdiri dari berbagai macam operasi diantaranya adalah operasi penjumlahan, pengurangan, perkalian, dan pembagian. Mendesain ALU juga memiliki cara yang hampir sama dengan mendesain enkoder, dekoder, multiplexer, dan demultiplexer. Rangkaian utama yang digunakan untuk melakukan perhitungan ALU adalah Adder.

4.1.1. *Adder*

Rangkaian ALU (*Arithmetic and Logic Unit*) yang digunakan untuk menjumlahkan bilangan dinamakan dengan *Adder*. Karena *Adder* digunakan untuk memproses operasi aritmetika, maka *Adder* juga sering disebut rangkaian kombinasional aritmetika. ALU akan dijelaskan lebih detail pada bab 3. Ada 2 jenis *Adder* :

1. Rangkaian *Adder* yang hanya menjumlahkan **dua bit** disebut *Half Adder*.
2. Rangkaian *Adder* yang menjumlahkan **tiga bit** disebut *Full Adder*.
3. Rangkaian Adder yang menjumlahkan banyak bit disebut paralel Adder

4.1.1.1. *Half Adder*

Rangkaian *half adder* merupakan dasar penjumlahan bilangan biner yang masing-masing hanya terdiri dari satu bit, oleh karena itu dinamakan penjumlah tak lengkap.

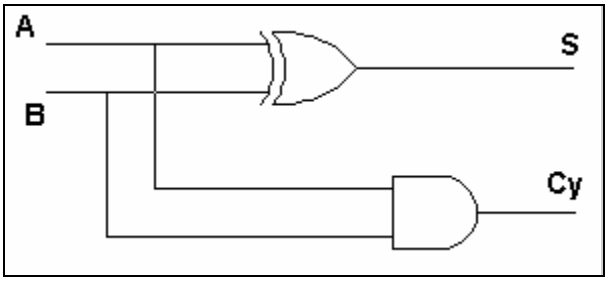
1. Jika A=0 dan B=0 dijumlahkan, hasilnya S (*Sum*) = 0.
2. Jika A=0 dan B=1 dijumlahkan, hasilnya S (*Sum*) = 1.
3. Jika A=1 dan B=1 dijumlahkan, hasilnya S (*Sum*) = 0. dengan nilai pindahan Cy(*Carry Out*) = 1.

Dengan demikian, *half adder* memiliki 2 masukan (A dan B) dan dua keluaran (S dan Cy).

A	B	S	Cy
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabel 4.1.

Dari tabel diatas, terlihat bahwa nilai logika dari *Sum* sama dengan nilai logika dari gerbang XOR, sedangkan nilai logika Cy sama dengan nilai dari gerbang logika AND. Dari tabel tersebut, dapat dibuat rangkaian *half adder* seperti pada gambar berikut:



Gambar 4.1.

4.1.1.2. *Full Adder*

Full adder mengolah penjumlahan untuk 3 bit bilangan atau lebih (bit tidak terbatas), oleh karena itu dinamakan rangkaian penjumlah lengkap. Perhatikan tabel kebenaran dari *Full adder* berikut :

A	B	C	S	Cy
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabel 4.2.

Dari tabel diatas dapat dibuat persamaan boolean sebagai berikut :

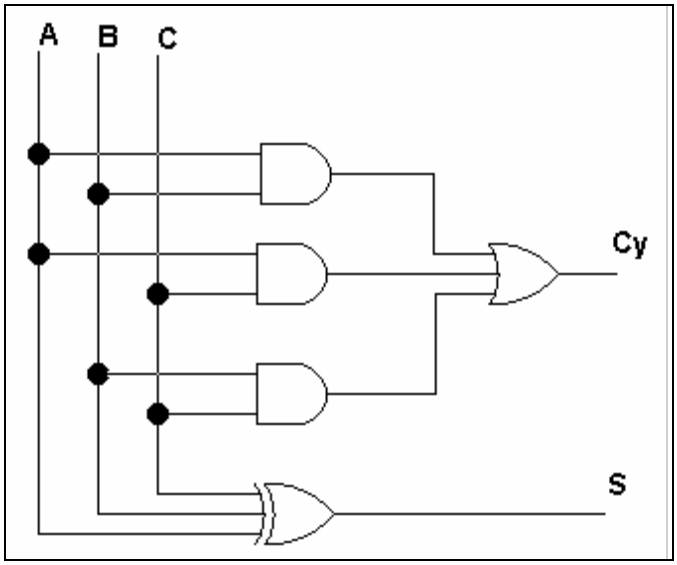
$$S = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + A\,B\,C$$

$$S = A \oplus B \oplus C$$

$$Cy = \overline{A}\,B\,C + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C$$

Dengan menggunakan peta *karnaugh*, Cy dapat diserhanakan menjadi :

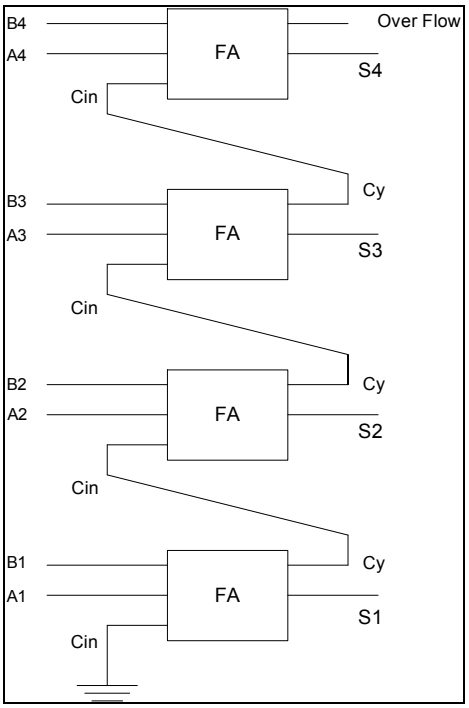
$Cy = AB + AC + BC$



Gambar 4.2.

4.1.1.3. *Parallel Adder*

Parallel Adder adalah rangkaian *Full Adder* yang disusun secara parallel dan berfungsi untuk menjumlah bilangan biner berapapun bitnya, tergantung jumlah *Full Adder* yang diparallelkan. Gambar berikut menunjukkan *Parallel Adder* yang terdiri dari 4 buah *Full Adder* yang tersusun parallel sehingga membentuk sebuah penjumlah 4 bit.



Gambar 4.3.

4.2.1. Penjumlahan

Komputer hanya dapat melakukan proses aritmetika menggunakan bilangan biner. Semua sistem bilangan harus diubah terlebih dahulu ke biner agar dapat diproses. Proses yang biasa dilakukan oleh komputer untuk menjumlahkan sistem bilangan desimal biasanya adalah menyandikan ke 8421BCD terlebih dahulu sebelum dijumlahkan. Sebelum mempelajari tentang penjumlahan pada 8421BCD, ada baiknya mengetahui cara menjumlahkan bilangan biner.

a. Penjumlahan Biner

Ada 4 kondisi yang terjadi pada penjumlahan biner yaitu apabila 0 + 0, 0 + 1, 1 + 0, dan 1 + 1. Jika yang terjadi adalah 1 + 1, kita tidak dapat menyatakan hasil jumlah dalam satu digit. Tetapi kita harus melakukan penyimpanan (*Carry Out*) kedalam kolom yang lebih tinggi. Ini berlaku untuk seluruh sistem bilangan. Sebagai contoh pada bilangan desimal 2 + 5 = 7 dengan *carry out* = 0, 9 + 9 = 8 dengan *carry out* = 1.

Contoh :

1. $10_2 + 10_2$

1	Carry out
10	
10 +	
100	

2. $0100_2 + 0111_2$

1	Carry
0100	
0111 +	
1011	

3. $11111_2 + 1111_2 + 11101_2 + 10111_2$

3 3 3 2 2	Carry out
1 1 1 1 1	
0 1 1 1 1	
1 1 1 0 1	
1 0 1 1 1 +	
1 1 0 0 0 1 0 $_2$	

b. Penjumlahan 8421BCD

Sandi 8421BCD hanya menggunakan bilangan biner untuk 0 sampai 9, karena yang disandikan hanya 1 digit angka desimal. Dalam penjumlahan yang perlu diperhatikan adalah jika hasilnya lebih dari 9 sehingga akan dihasilkan *auxillary carry* (*Carry* dari bilangan keempat LSB) maupun *carry* dari MSB.

Berikut adalah aturan penjumlahan sandi 8421BCD:

- Jika jumlah biner dan jumlah BCD sama, yaitu AC (*Auxillary Carry*) = 0 dan *Carry* = 0 maka tidak diperlukan aturan tambahan.

Contoh :

Bilangan 1	=	0	1	1	0		0	0	1	0	(BCD)	=	6	2	(desimal)	
Bilangan 2	=	0	0	1	0		0	1	0	1	(BCD)	=	2	5	(desimal)	+
Biner	=	1	0	0	0		0	1	1	1			(Cy=0; AC = 0)			
BCD	=	1	0	0	0		0	1	1	1		=	8	7	(desimal)	

- Jika jumlah biner tidak sama dengan jumlah desimal maka memerlukan pengaturan tambahan :

Jika *Auxillary Carry* (AC) = 0 atau AC = 1 dan *Carry* (Cy) = 0 dimana hasil penjumlahan binernya lebih dari 9 desimal, maka perlu ditambahkan 6 pada nible rendah tersebut, dan tambahkan 1 pada nible yang lebih tinggi.

Contoh :

			1			1	1								
Bilangan 1	=	0	0	1	1	0	1	1	1	(BCD)	=	3	7	(desimal)	
Bilangan 2	=	0	0	1	0	0	1	1	0	(BCD)	=	2	6	(desimal)	+
Biner	=	0	1	0	1	1	1	0	1						
+ 1 & + 6	=	0	0	0	1	0	1	1	0			(Cy=0; AC = 0)			
BCD	=	0	1	1	0	0	0	1	1		=	6	3	(desimal)	

4.2.2. Pengurangan

Pengurangan pada dasarnya merupakan penjumlahan, yaitu penjumlahan dengan bilangan negatif.

500 – 255 = 245 (Pengurangan)

500 + (-)255 = 245 (Penjumlahan)

Komputer hanya bekerja pada bilangan “0” dan “1” dan tidak mengenal bilangan negatif. Untuk menunjukkan bilangan negatif, komputer menggunakan tanda modulus (*Modulus Sign*). Pada penjumlahan desimal tanda modulus yang digunakan adalah “0” untuk bilangan positif dan “9” untuk bilangan negatif. Untuk bilangan negatif, pada operasi penjumlahannya, harus dikomplemen. Komplemen yang digunakan pada bilangan desimal adalah komplemen-10 dan komplemen-9.

Pengurangan Bilangan Desimal

- Komplemen-10

Pada komplemen-10, bilangan negatif dikurangkan 9, kemudian ditambahkan 1 pada bit terakhir. Pada penjumlahannya, bila ada carry, carry tersebut dapat dihilangkan. Tanda modulus ikut dijumlahkan.

Contoh :

Komplemen-10 dari -255.

- 2 5 5_{10} = (9) 7 4 5_{10} (angka 9 menunjukkan tanda modulusnya).

5	0	0			(0)	5	0	0	
2	5	5	-		(9)	7	4	5	+
2	4	5		1	(0)	2	4	5	
				Cy	Dihilangkan				

- Komplemen-9

Pada komplemen-9, bilangan negatif dikurangkan 9. Bila ada carry, maka carry ikut dijumlahkan pada hasil akhir.

Contoh :

Komplemen-9 dari -255.

- 2 5 5_{10} = (9) 7 4 4_{10} (angka 9 menunjukkan tanda modulusnya).

5	0	0			(0)	5	0	0	
2	5	5	-		(9)	7	4	4	+
2	4	5		1	(0)	2	4	4	
								1	+(Cy)
					(0)	2	4	5	
				Cy	Ditambahkan				

Bila hasil akhir bernilai negatif, maka nilainya harus dikomplemen lagi (Berlaku untuk komplemen-10 dan komplemen-9). Jika komplemen-10, maka hasil akhir setelah dikomplemen harus ditambah 1. Jika komplemen-10, hasil akhirnya merupakan hasil sebenarnya (tidak perlu ditambah 1).

Contoh :

Komplemen-10 dari -500.

$- 5\ 0\ 0_{10} = (9)\ 5\ 0\ 0_{10}$ (angka 9 menunjukkan tanda modulusnya).

	2	5	5		(0)	2	5	5	
	5	0	0	-	(9)	5	0	0	+
-	2	4	5		(9)	7	5	5	(9 menunjukkan negatif)
						2	4	4	+ 1
					(9)	2	4	5	

Komplemen-9 dari -500.

$- 5\ 0\ 0_{10} = (9)\ 4\ 9\ 9_{10}$ (angka 9 menunjukkan tanda modulusnya).

	2	5	5		(0)	2	5	5	
	5	0	0	-	(9)	4	9	9	+
-	2	4	5		(9)	7	5	4	(9 menunjukkan negatif)
						2	4	5	
					(9)	2	4	5	

Pengurangan Bilangan Biner

Pada penjumlahan biner, komplemen yang digunakan adalah komplemen-2 dan komplemen-1. Untuk mendapatkan komplemen bilangan biner, cukup dengan membalik angkanya saja. Jika "0" dibalik menjadi "1", dan jika "1" dibalik menjadi "0". Komplemen-2 mirip dengan komplemen-10 pada bilangan desimal (Carry dihilangkan), sedangkan komplemen-1 mirip dengan komplemen-9 (Carry ditambahkan pada hasil akhir).

- Komplemen-2

Contoh :

Pengurangan antara 9_{10} (1001_2) dengan 5_{10} (0101_2)

Komplemen-2 dari –5 (0101).

0 1 0 1 = (1) 1 0 1 1 (angka 1 menunjukkan tanda modulusnya).

9				(0)	1	0	0	1		
5	-			(1)	1	0	1	1	+	
4		1		(0)	0	1	0	0		
		Cy	Dihilangkan							

- Komplemen-1

Contoh :

Komplemen-1 dari –5 (0101).

0 1 0 1 = (1) 1 0 1 0 (angka 1 menunjukkan tanda modulusnya).

9				(0)	1	0	0	1		
5	-			(1)	1	0	1	0	+	
4		1		(0)	0	0	1	1		
								1		
				(0)	0	1	0	0		
		Cy	Ditambahkan							

Bila hasil akhir bernilai negatif, maka nilainya harus dikomplemen lagi (Berlaku untuk komplemen-2 dan komplemen-1). Jika komplemen-2, maka hasil akhir setelah dikomplemen harus ditambah 1. Jika komplemen-1, hasil akhirnya merupakan hasil sebenarnya (tidak perlu ditambah 1).

Contoh :

Pengurangan antara 5_{10} dengan 9_{10}

Komplemen-2 dari –9.

1 0 0 1 = (1) 0 1 1 1 (angka 1 menunjukkan tanda modulusnya).

	5		(0)	0	1	0	1		
	9	-	(1)	0	1	1	1	+	
-	4		(1)	1	1	0	0	(1 menunjukkan negatif)	
				0	0	1	1	+ 1	
			(1)	0	1	0	0		

Komplemen-1 dari –9.

1 0 0 1 = (1) 0 1 1 0 (angka 1 menunjukkan tanda modulusnya).

	5		(0)	0	1	0	1	
	9	-	(1)	0	1	1	0	+
-	4		(1)	1	0	1	1	(1 menunjukkan negatif)
				0	1	0	0	
			(1)	0	1	0	0	

4.2.3. Perkalian

Perkalian antara bilangan biner adalah perkalian yang paling mudah diantara sistem bilangan lainnya.

9							1	0	0	1			
10	x						1	0	1	0	x		
90							0	0	0	0			
					1	0	0	1					
				0	0	0	0						
			1	0	0	1					+		
			1	0	1	1	0	1	0				
			64	0	16	8	0	2	0	=	90		

Pada Teknik Komputer, perkalian dilakukan menggunakan register geser kanan (*Shift Right Register*). Perhatikan contoh berikut :

Register A untuk menyimpan data yang akan dikalikan (*Multiplicand*).

Register B untuk menyimpan data pengali (*Multiplier*).

Register P untuk menyimpan hasil perkalian.

9 X 10

Register A	Register B	Register P
1 0 0 1	1 0 1 0	0 0 0 0 0 0 0 0
	M	M = 0, Reg. P tidak diubah
		Geser Reg B & P 1 bit kekanan
1 0 0 1	1 0 1	0 0 0 0 0 0 0 0
	M	M = 1, Reg A ditambahkan pada P di MSBnya.
		1 0 0 1 0 0 0 0
		Geser Reg B & P 1 bit kekanan
1 0 0 1	1 0	0 1 0 0 1 0 0 0
	M	M = 0, Reg. P tidak diubah
		Geser Reg B & P 1 bit kekanan
1 0 0 1	1	0 0 1 0 0 1 0 0
	M	M = 1, Reg A ditambahkan pada P di MSBnya
		1 0 1 1 0 1 0 0
		Reg. P geser lagi
		0 1 0 1 1 0 1 0
		0 64 0 16 8 0 2 0 =90

4.2.4. Pembagian

Kebalikan dari perkalian, pembagian (Division) adalah suatu bentuk dari pengurangan yang dilakukan berulang-ulang. Dan proses ini juga dapat dilakukan pada rangkaian logika dengan cara pengurangan dan penggeseran ke kiri (menggunakan shift-left register). Berikut adalah aturan dari pembagian:

Kurangkan bilangan pembagi (Divisor) dari MSB bilangan yang akan dibagi (Dividend), lihat hasil pengurangan.

Bila hasilnya 1 atau positif :

Berarti hasil pembagian (Product) adalah 1. Setelah itu hasil pengurangan digeser kekiri satu bit, dan dimulai lagi pengurangan oleh bilangan pembagi (Divisor).

Bila hasilnya 0 atau negatif :

Berarti hasil pembagian (Product) adalah 0. Dalam hal ini sebelum digeser ke kiri harus ditambah dulu dengan bilangan pembagi (Divisor). Setelah digeser ke kiri satu bit, dimulai lagi proses pengurangan oleh bilangan pembagi. Pengurangan oleh bilangan pembagi dilakukan dengan penjumlahan komplemen-2. Bila dalam penjumlahan tersebut terdapat pindahan (Carry), maka carry tersebut diabaikan. Perhatikan contoh berikut :

$10_{10} : 4_{10} = 1010_2 : 100_4$

Pembagi	Yang Dibagi	Hasil Bagi	Keterangan
(0) 1 0 0	(0) 1 0 1 0		- Kurangkan bil. pembagi
	(1) 1 0 0		
	1 (0) 0 0 1 0	1	- Hasil positif, hasil bagi = 1
	(0) 0 1 0 0	1	- Digeser ke kiri satu bit
	(1) 1 0 0		- Kurangkan bil. pembagi
	(1) 1 1 0 0	1 0	- Hasil negatif, hasil bagi = 0
	(0) 1 0 0		
	1 (0) 0 1 0 0		
	(0) 1 0 0 0	1 0	- Digeser ke kiri satu bit
	(1) 1 0 0		- Kurangkan bil. pembagi
	1 (0) 0 0 0 0	1 0 1	- Hasil positif, hasil bagi = 1

Catatan : Karena ada hasil pengurangan yang negatif, maka digit yang dihasilkan setelah itu adalah digit pecahan, sehingga hasil yang benar $10,1_2$ atau $2,5_{10}$.