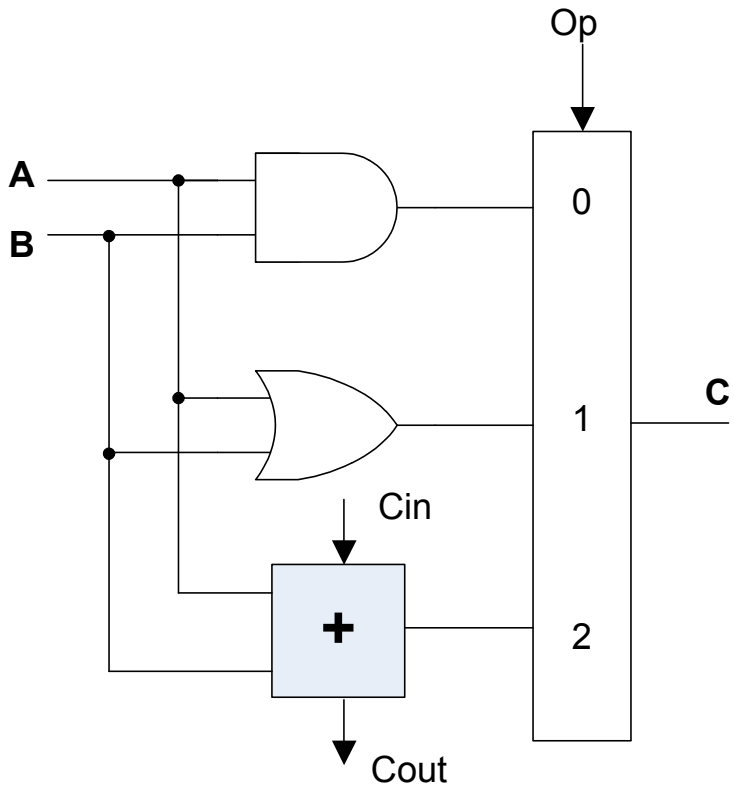
ARITHMETIC & LOGICAL UNIT (ALU)

Arsitektur Komputer

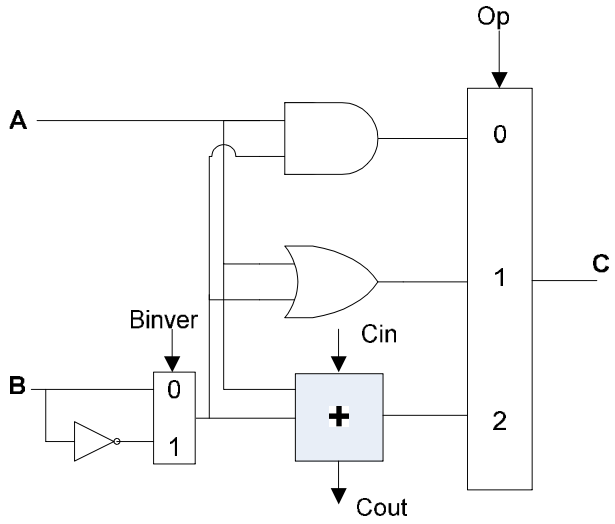
PENDAHULUAN

- Empat metoda komputasi dasar yang dilakukan oleh ALU komputer : penjumlahan, pengurangan, perkalian, dan pembagian.
- Rangkaian ALU dasar terdiri atas gerbang OR, AND, dan rangkaian full adder 1 bit.
- Rangkaian full adder 1 bit pada rangkaian ALU dasar pada awalnya hanya melakukan penjumlahan unsigned number.
- Pengembangan lebih lanjut pada rangkaian ALU dasar mampu melakukan operasi pengurangan.

RANGKAIAN ALU DASAR KOMPUTER



Tanpa Fungsi Pengurangan



Dengan Fungsi Pengurangan

Operasi Aritmatika Dasar

- Addition / Penjumlahan
- Complements
- Subtraction / Pengurangan

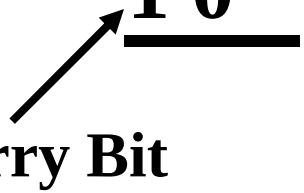
Penjumlahan Biner

(a)
$$\begin{array}{r} 0 \\ +\ 0 \\ \hline 0 \end{array}$$

(b)
$$\begin{array}{r} 0 \\ +\ 1 \\ \hline 1 \end{array}$$

(c)
$$\begin{array}{r} 1 \\ +\ 0 \\ \hline 1 \end{array}$$

(d)
$$\begin{array}{r} 1 \\ +\ 1 \\ \hline 1\ 0 \end{array}$$



Contoh Penjumlahan Biner dengan operand lebih dari 1 bit

(a)

```
  1011
+ 1100
------
 10111
------
```

(b)

```
  1010
+  100
------
  1110
------
```

(c)

```
  1011
+  101
------
 10000
------
```

(d)

```
   101
+ 1001
------
  1110
------
```

(e)

```
 10011001
+   101100
----------
 11000101
----------
```

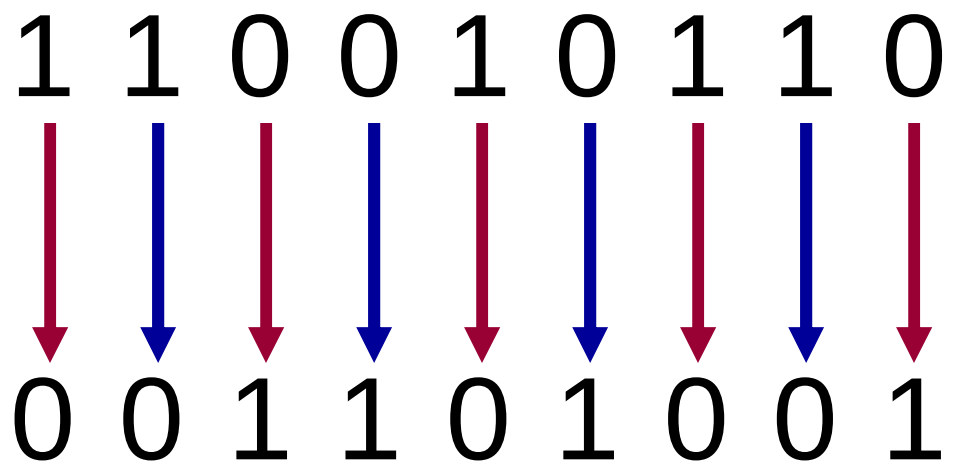
Binary Complement

Operasi (1s Complement)

1 → 0

0 → 1

Example



Two’s Complement

Nilai Two’s complement bilangan biner diperoleh dengan menambahkan nilai ‘1’ pada hasil One’s Complement.

1001110

0110001 ⟸ One’s Complement

\+ 1

0110010 ⟸ Two’s Complement

Pengurangan Biner

Pengurangan Biner diimplementasikan dengan menjumlahkan Two's complement bilangan yang akan dikurangkan.

Example

```
  1101
 -1001
```

```
  1101
 +0111
 -----
 10100
```

Two's complement of 1001

Carry yang dihasilkan dapat diabaikan. Sehingga, hasilnya adalah 0100.

RANGKAIAN PERKALIAN

- Dua Buah bilangan biner dapat dikalikan dengan metoda yang sama dengan metoda perkalian pada bilangan desimal.
- Sebagai pengantar akan ditunjukkan operasi perkalian konvensional dengan bilangan tak bertanda (unsigned number).
- Sebagai contoh akan ditunjukkan operasi perkalian untuk operand Multiplicand (M) = 1110 dan Multiplier (Q) = 1011.

Gambaran Proses Perkalian 4 Bit

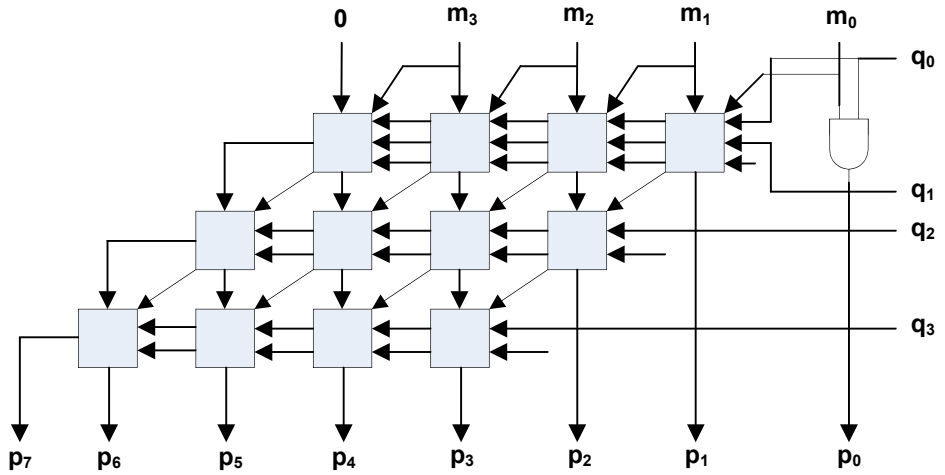
Multiplicand (M)	1110	(14)
Multiplier (Q)	x 1011	(11)
	1110	
	1110	
	0000	
	+ 1110	
Product (P)	10011010	(154)

Konsep dasar perkalian konvensional

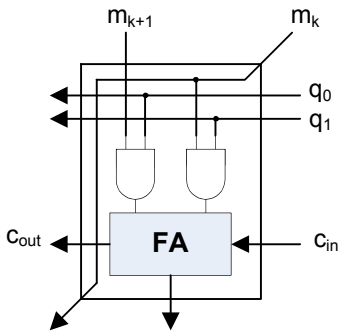
Multiplicand (M)	1110	(14)
Multiplier (Q)	x 1011	(11)
Partial Product 0	1110	
	+ 1110	
Partial Product 1	10101	
	+ 0000	
Partial Product 2	01010	
	+ 1110	
Product (P)	10011010	(154)

Perkalian Konvensional Implementasi HW

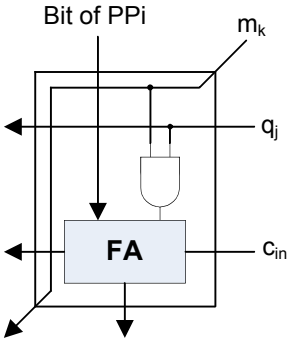
ARRAY MULTIPLIER UNTUK BILANGAN UNSIGNED



Struktur Rangkaian

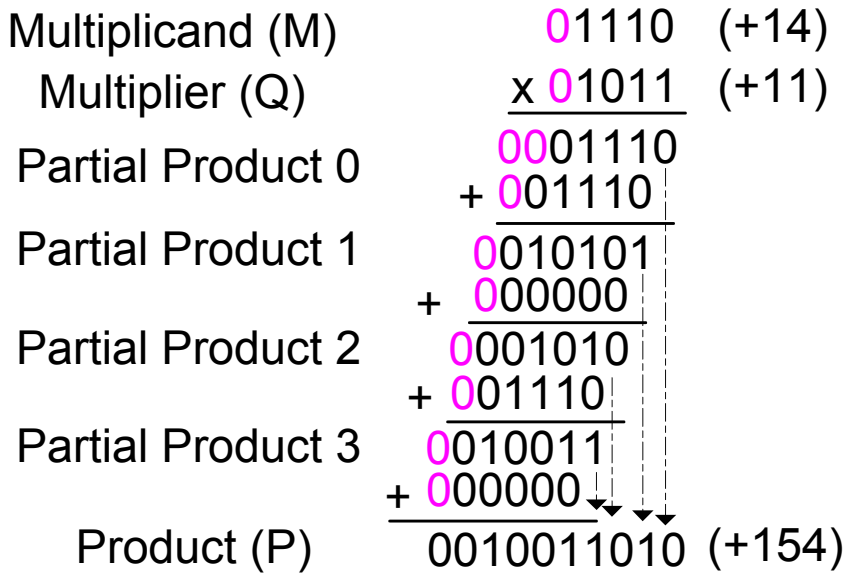


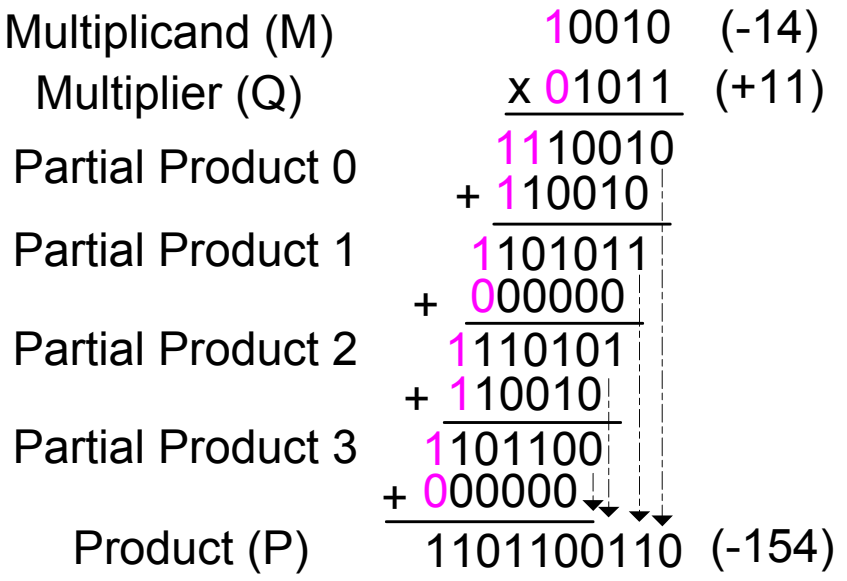
Blok Pada Baris Pertama



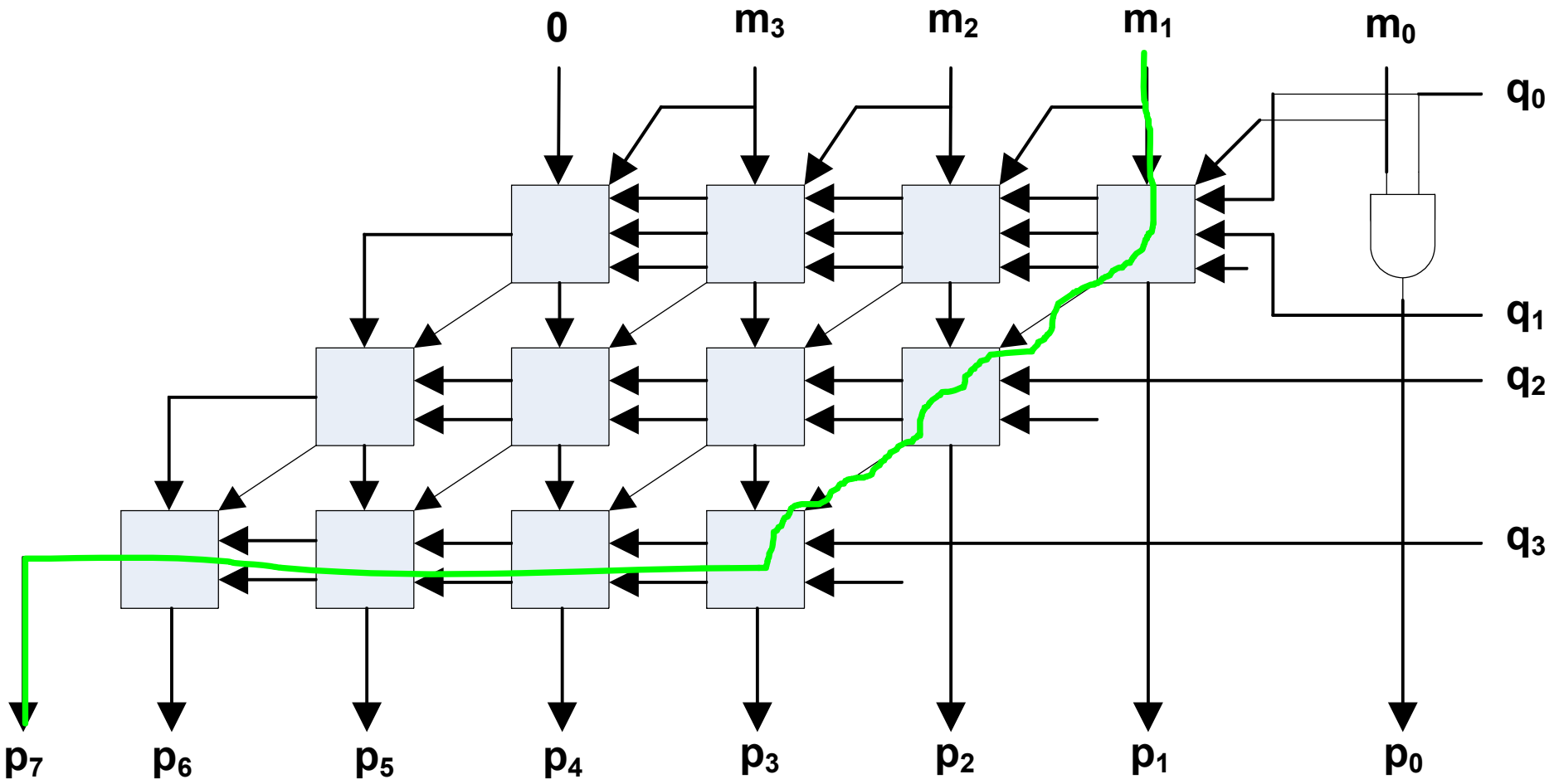
Blok Pada Baris Kedua dan Ketiga

Perkalian Bilangan Bertanda





Critical Delay Path Pada Array Multiplier



Masalah & Pemecahan Pada Array Multiplier

- Critical Delay Path nya besar
- Untuk meningkatkan performansi multiplier digunakan konsep pipelining
- Pipelining mampu mengurangi waktu siklus tetapi tidak mengurangi waktu total proses perkallian.
- Salah satu algoritma untuk mempercepat perkalian ini adalah Booth encoding algorithm.

Booth Encoding Algorithm

- Merupakan salah satu algoritma untuk meningkatkan kecepatan proses perkalian
- Algoritma ini menggunakan ide dasar bahwa proses adder-subtractor secara kecepatan dan tingkat kesederhanaan rangkaian hampir sama dengan adder sederhana.
- Bentuk umum algoritma ini berhubungan dengan 3 bit pengali pada satu waktu yang membentuk proses perkalian dua tingkat.

Algoritma Booth

Jika dinyatakan representasi 2's complement multiplier y :

$$y = -s^n y_n + 2^{n-1} y_{n-1} + 2^{n-2} y_{n-2} + \ldots$$

Dengan ide dasar : $2^a = 2^{a+1} - 2^a$

Dua item awal persamaan pertama dapat dinyatakan sebagai :

$$2^n(y_{n-1} - y_n) + 2^{n-1}(y_{n-2} - y_{n-1})$$

Setiap bentuk merupakan satu tahapan pada algoritma perkalian dasar.

Tabel Recoding Bits

y_i	y_{i-1}	y_{i-2}	increment
0	0	0	0
0	0	1	x
0	1	0	x
0	1	1	2x
1	0	0	-2x
1	0	1	-x
1	1	0	-x
1	1	1	0

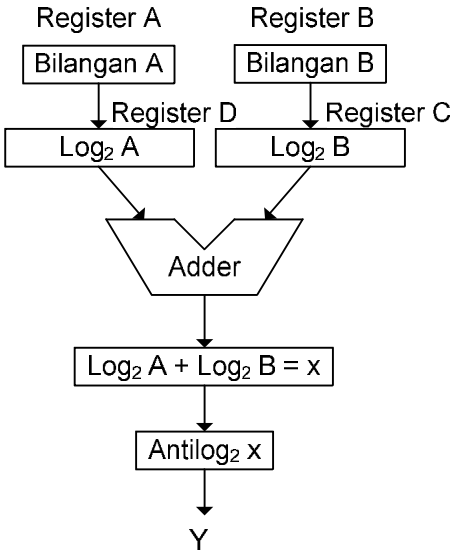
Algoritma Perkalian Multioperand Dengan Fungsi Logaritmik dan MSB First BIT Adder

- Salah satu algoritma untuk mengatasi masalah waktu proses dalam multiplikasi.
- Merupakan algoritma perkalian paralel yang menggabungkan Logarithmic Multiplier dan Multioperand MSB first adder.
- Pada algoritma Logaritmik, perkalian dilakukan dengan menjumlahkan operand satu sama lain.
- Penjumlahan multioperand dengan metode MSB First Adder adalah suatu konsep metoda penjumlahan sejumlah bilangan dengan dimulai dari bit MSB nya terlebih dahulu.

Algoritma dan Model Arsitektur Perkalian Logaritmik 2 Operand

- Ide dasar perkalian dengan metode logaritmik dilakukan dalam bentuk penjumlahan sesuai dengan persamaan sebagai berikut :
  - $\text{Log}(A \times B) = \text{Log } A + \text{Log } B$
  - $\text{Log}_2(A \times B) = \text{Log}_2 A + \text{Log}_2 B$
  - $A \times B = \text{Antilog}_2 (\text{Log}_2 A + \text{Log}_2 B)$
- Yang perlu diperhatikan dalam operasi perkalian logaritmik ini adalah error yang dapat muncul pada saat konversi ke bentuk logaritmik dan antilogaritmik.

Perkalian Logaritmik 2 Operand



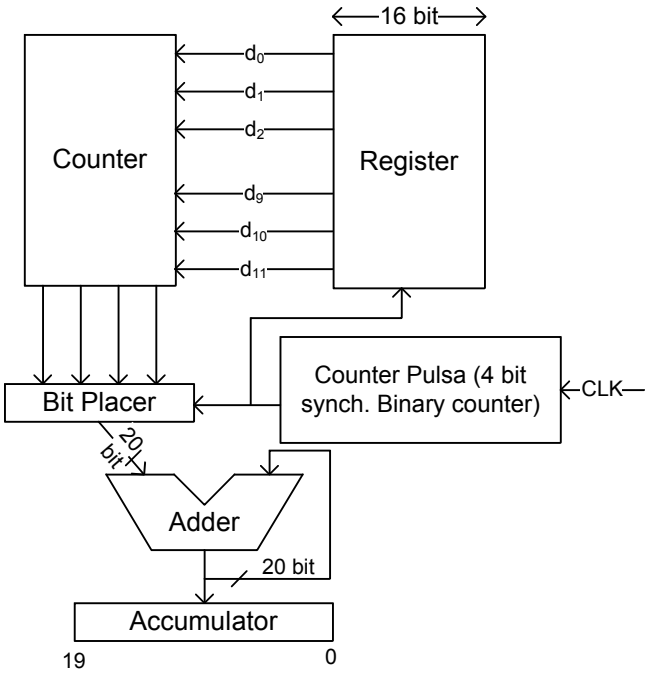
Algoritma Perkalian Logaritmik antara 2 buah bilangan

- Berdasarkan persamaan di atas, langkah yang harus ditempuh adalah sebagai berikut :
- Ambil 2 buah bilangan biner, masukkan kedua bilangan ke dalam register A dan B.
- Konversikan kedua bilangan tersebut dalam nilai logaritma basis 2 dan masukkan ke dalam register C dan D.
- Lakukan penjumlahan isi register C dan D, simpan hasilnya pada Accumulator.
- Konversikan hasil penjumlahan tersebut dengan menggunakan antilog2 dan simpan hasilnya pada suatu register.

Algoritma dan Model Arsitektur Penjumlahan Multioperand Dengan MSB First Bit Process

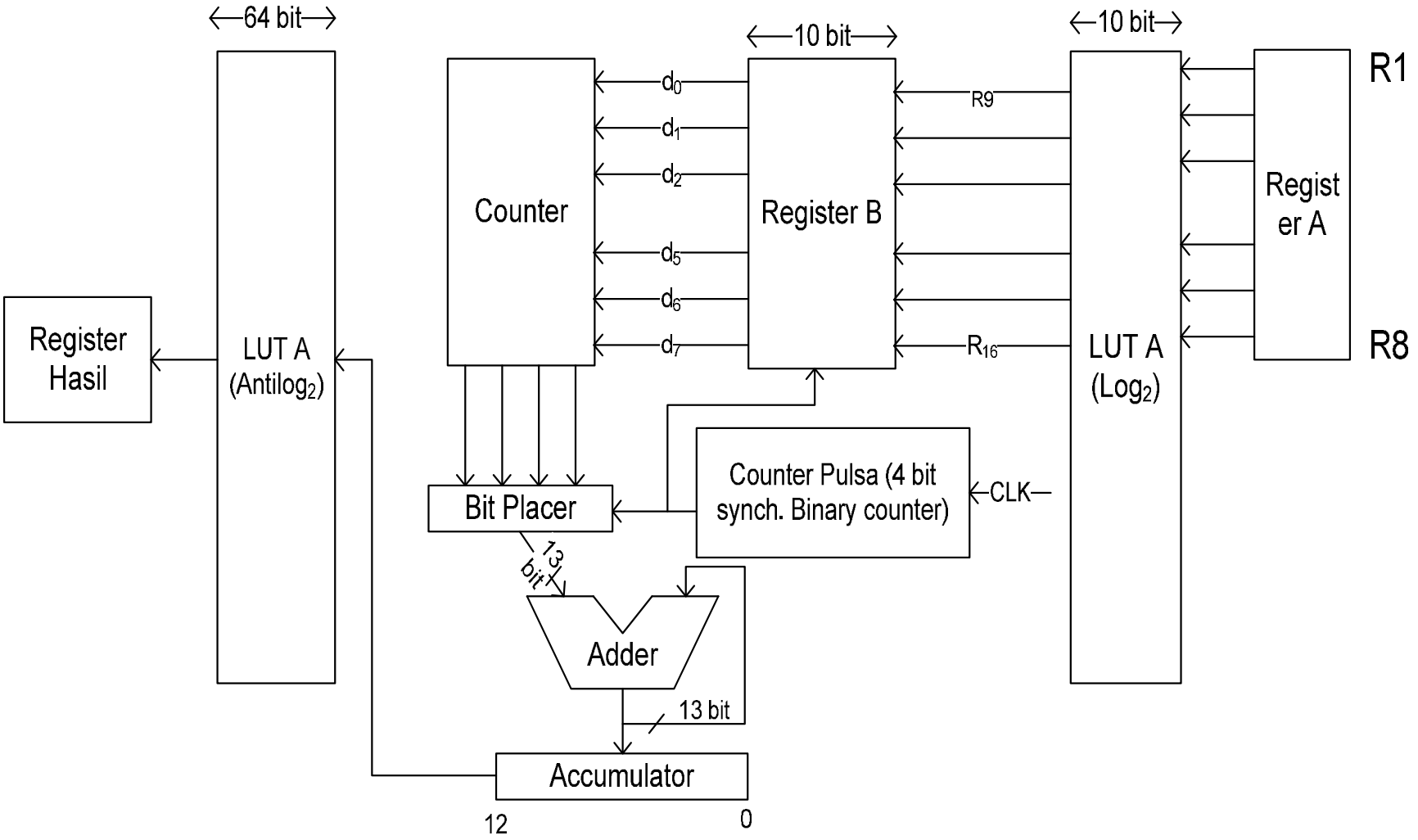
- Diaplikasikan untuk sistem waktu nyata.
- Perbedaan dengan algoritma penjumlahan konvensional terletak pada urutan penjumlahan yang dilakukan.
- Pada algoritma ini bit yang pertama kali dijumlahkan adalah bit MSB → MSB-1 → LSB. (Tenggat waktu yang ditetapkan dapat dipenuhi).
- Dengan algoritma ini, sebelum penjumlahan sampai bit LSB, hasil yang tersimpan pada accumulator telah dapat digunakan.

Arsitektur Penjumlahan Multioperand MSB First Bit



Tahapan Algoritma yang dilakukan

- Masukkan semua operand n bit ke dalam N register.
- Untuk N operand dengan n bit data, lakukan langkah-langkah berikut :
  - Jumlahkan semua MSB dari setiap operand dan letakkan hasilnya pada accumulator.
  - Jumlahkan semua MSB-1 dan jumlahkan hasilnya dengan yang tersimpan pada accumulator lalu simpan hasilnya kembali pada accumulator.
  - Lakukan langkah kedua tersebut sampai bit LSB dari setiap operand selesai dijumlahkan.



Model Arsitektur Perkalian Multioperand fungsi Logaritmik

Algoritma Perkalian Logaritmik Multioperand

- Secara konsep akan melakukan perkalian dengan banyak operand dengan cara menjumlahkan nilai logaritmik setiap operand.
- Konsep dasar secara matematis :
- $Log_2(AxBx…xN) = Log_2A + Log_2B + … + Log_2N$
- Jadi secara umum CPU hanya melakukan proses penjumlahan untuk sejumlah operand. Namun untuk mempercepat hasil penjumlahan, digunakan algoritma penjumlahan dengan dimulai dari MSB → LSB.
- Untuk mendapatkan hasil logaritma basis 2 dari tiap operand, dan mengembalikannya ke bentuk asal, digunakan look up table yang digabungkan dengan konsep segmentasi.

Lanjutan Algoritma

- Arsitektur sistem ini dibatasi untuk operand 8 bit dan jumlah operand maksimal yang terlibat dalam operasi perkalian sebanyak 8 operand juga.
- Operasi maksimal yang dapat dilakukan adalah 255^8.
- Berarti bit data maksimum yang dihasilkan dari perkalian 8 operand 8 bit dengan look up table adalah 13 bit.

Algoritma Perkalian 8 operand 8 bit adalah :

- Cocokkan isi register 1 s.d 8 dengan LUT nilai logaritmik basis 2.
- Ambil data dari LUT dan masukkan ke dalam register 9 s.d 16.
- Lakukan penjumlahan multioperand dengan dimulai dari MSB.
- Hasil penjumlahan yang tersimpan pada accumulator dicocokkan dengan LUT antilog basis 2 untuk mendapatkan nilai sebenarnya.

Tabel 1 : Proses perkalian manual

Langkah iterasi	Operand A	Operand B	Hasil Accumulator
1 : 8	00000001	00000010	00000010
2 : 8	00000010	00000011	00000110
3 : 8	00000110	00000100	00011000
4 : 8	00011000	00000101	01111000
5 : 8	01111000	00000010	11110000
6 : 8	11110000	00000001	11110000
7 : 8	11110000	00000001	11110000

Tabel 2 :

Clock		Operand (A dan B)	Accumulator
1	A :	1000000000000	1000000000000
	B :	0000000000000	
2	A :	0100000000000	1100000000000
	B :	1000000000000	
3	A :	0010000000000	1110000000000
	B :	1100000000000	
4	A :	0001000000000	1111000000000
	B :	1110000000000	
5	A :	0000100000000	1111100000000
	B :	1111000000000	
6	A :	0000010000000	1111110000000
	B :	1111100000000	
7	A :	0000001000000	1111111000000
	B :	1111110000000	
8	A :	0000000100000	1111111100000
	B :	1111111000000	
9	A :	0000000010000	1111111110000
	B :	1111111100000	
10	A :	0000000001000	1111111111000
	B :	1111111110000	

KESIMPULAN

- Perkalian dengan menggunakan algoritma perkalian dengan logaritmik lebih cepat dan efisien, karena hanya membutuhkan proses penjumlahan.
- Faktor error merupakan ekses yang muncul saat terjadi proses konversi nilai logaritmik dan antilogaritmik yang dilakukan.