ANALISIS PERBANDINGAN METODE PERKALIAN ARRAY DAN BOOTH

Hendra Setiawan[1*], **Fahmi Nugraha**[1]
[1] Program Studi Teknik Elektro, Fakultas Teknologi Industri, Universitas Islam Indonesia
Jl. Kaliurang km.14.5, Yogyakarta 55582
[*]Email: hendra.setiawan@uii.ac.id

Abstrak

Di era komputasi digital sekarang ini, optimasi sistem komputasi yang terdiri dari berbagai operasi matematika merupakan salah satu topik bahasan yang penting. Diantara operasi matematika, perkalian merupakan operasi dasar yang memiliki komplesitas, area, dan konsumsi daya paling tinggi dibandingkan operasi lainnya. Pada makalah ini, dibandingkan dua jenis teknik implementasi perkalian yang sampai saat ini masih terus dikembangkan, yaitu perkalian array dan perkalian Booth. Perbandingkan keduanya meliputi frekuensi clock maksimum yang dapat diterapkan, komplesitas, dan kecepatan proses komputasi. Perbandingan dilakukan dengan acuan perkalian signed binary dengan lebar data 4 bit dan 8 bit. Hasil analisa dan implementasi pada FPGA menunjukkan hasil bahwa perkalian array mempunyai keunggulan pada kecepatan proses komputasi, sedangkan perkalian Booth mempunyai kelebihan dalam hal komplesitas rangkaian.

***Kata kunci**: arsitektur perkalian, FPGA, perkalian array, perkalian Booth*

1. PENDAHULUAN

Perkalian merupakan salah satu komponen penting dalam berbagai sistem digital seperti filter, mikroprosesor, dan digital signal processors (DSP). Kompleksitas suatu sistem digital pada umumnya ditentukan oleh seberapa banyak perkalian yang terlibat di dalamnya. Kecepatan proses komputasi juga sangat dipengaruhi oleh proses perkalian yang ada dikarenakan proses perkalian pada umumnya merupakan bagian yang memerlukan waktu paling lama (Ghosh,2007). Dibandingkan dengan penjumlahan dan pengurangan, perkalian memerlukan sumber daya dan area yang paling besar. Pada kenyataannya, 8.72% dari keseluruhan tahapan dari suatu unit proses adalah perkalian (Laxman dkk., 2012). Sehingga pemilihan arsitektur perkalian yang tepat menjadi salah satu kunci utama dalam optimasi rangkaian digital.

Unjuk kerja suatu proses perkalian ditentukan oleh kecepatan proses (*speed*) dan ukuran (*area*). Kecepatan proses sangat dipengaruhi oleh seberapa panjang tahapan perhitungan yang dilakukan, sedangkan ukuran sangat dipengaruhi oleh seberapa banyak komponen yang terlibat di dalamnya. *Speed* dan *area* pada umumnya saling mempengaruhi, misalkan ketika *speed* ditingkatkan dengan teknik *pipeline*, maka akan berdampak pada *area* yang semakin besar. Selain kedua hal tersebut, unjuk kerja suatu proses perkalian juga dapat dilihat dari *latency* yang dihasilkan. Semakin besar nilai *latency* maka semakin besar tunda hasil perkalian diperoleh.

Kompleksitas suatu perkalian sangat dipengaruhi ukuran bit bilangan pengali dan yang dikalikan. Untuk *n*-bit pengali dan *m*-bit yang dikalikan, memerlukan *n*+*m* bit sistem perkalian. Hal tersebut berlaku untuk sistem *unsigned* bit. Operasi perkalian yang melibatkan nilai positif dan negatif dengan sistem 2's *complement* untuk panjang bit masing-masing *n* dan *m*, diperlukan (*n*+*m*)-1 bit sistem perkalian. Proses perkalian ini akan semakin kompleks ketika bilangan yang dioperasikan adalah bilangan pecahan.

Sampai saat ini terdapat berbagai macam teknik perkalian dengan keunggulan dan kelemahannya masing-masing. Pembahasan di makalah ini akan dibatasi pada teknik array multiplier, algoritme Booth, dan teknik perkalian dengan memory(RAM)-based. Perbandingan dilakukan pada aspek kecepatan dan ukuran komponen yang digunakan. Komponen yang digunakan melingkupi penjumlahan, register, dan multiplexer/selektor.

Pembahasan di bab-bab berikutnya dirinci sebagai berikut. Pada bab II dipaparkan tentang beberapa teknik perkalian yang ada. Perbandingan masing-masing teknik perkalian tersebut dan diskusinya disajikan di bab III. Akhir dari makalah ini ditutup dengan kesimpulan di bab IV.

2. TEKNIK PERKALIAN

2.1. Perkalian *Array*

Teknik perkalian *array* merupakan salah satu teknik perkalian yang muncul terlebih dahulu. Konsep yang diusung sama dengan konsep perkalian konvensional. Untuk perkalian X dan Y yang masing-masing terdiri dari 4 bit, maka diperlukan tahapan sebagaimana ditunjukkan di gambar 1.

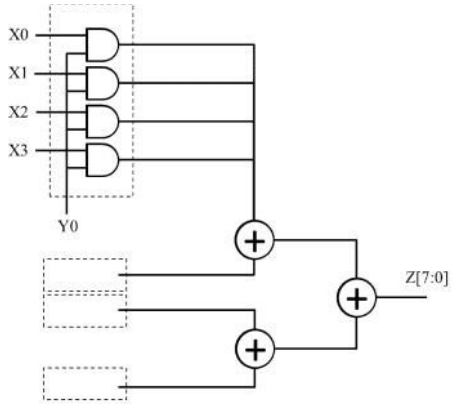
			X3	X2	X1	X0	
			Y3	Y2	Y1	Y0	×
			Y0X3	Y0X2	Y0X1	Y0X0	
		Y1X3	Y1X2	Y1X1	Y1X0		
	Y2X3	Y2X2	Y2X1	Y2X0			
Y3X3	Y3X2	Y3X1	Y3X0				+
Z6	Z5	Z4	Z3	Z2	Z1	Z0	

Gambar 1. Proses perkalian *array* 4 bit

Menurut Actel, 1996, implementasi dari proses di atas adalah dengan gerbang logika AND dan beberapa penjumlahan seperti ditunjukkan pada gambar 2. Sehingga untuk perkalian 4 bit diperlukan 16 gerbang AND dua input dan 3 buah penjumlah 8 bit. Namun tidak semua penjumlah memerlukan lebar 8 bit, sehingga kompleksitas dapat dikurangi.

Keuntungan dari perkalian array adalah memiliki struktur yang teratur, sehingga mudah untuk ditata berakibat pada ukuran yang kecil. Keuntungan kedua dari teknik perkalian ini adalah kemudahan desain untuk dilakukan *pipeline*. Setelah proses pipeline, kecepatan frekuensi cuplik yang diterapkan pada rangkaian ini dapat meningkat cukup fantastis.

Keterbatasan utama dari sistem perkalian ini adalah penambahan kompleksitas yang diakibatkan oleh penambahan ukuran bit baik di sisi pengali maupun yang dikalikan. Selain itu implementasi *pipeline* untuk meningkatkan kecepatan frekuensi cuplik mengakibatkan penambahan ukuran yang cukup berarti. Jalur terpanjang (*critical path*) pada umumnya terjadi di proses penjumlahan yang berada di ujung proses perkalian.

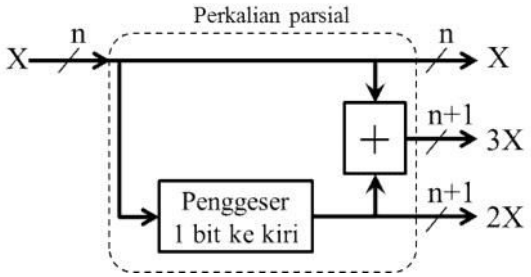


Gambar 2. Implementasi perkalian *array* 4 bit

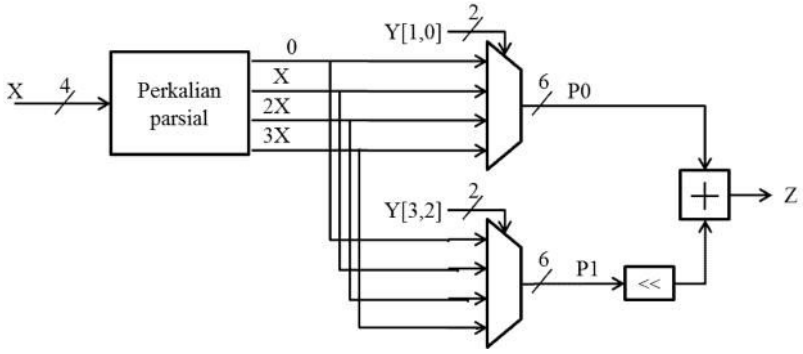
Sistem perkalian array selanjutnya dapat dimodifikasi untuk mengurangi proses penjumlahan yang diperlukan. Modifikasi yang dilakukan adalah dengan melakukan perkalian langsung 2 bit [3]. Misalkan X = 1011 dikalikan dengan Y = 0110, maka proses perkalian tersebut terdiri dari dua buah perkalian parsial yaitu P1 = 1011×01 dan P0 = 1011×10. Perkalian parsial pertama memberikan hasil P1 = 1011 dan proses perkalian parsial kedua menghasilkan P0 = 10110. Kemudian keduanya ditambahkan dengan posisi P1 bergeser 2 digit ke kiri, sehingga diperoleh 101100+10110 = 0100 0010.

Perkalian parsial dapat diimplementasikan dengan sebuah penjumlah dan sebuah penggeser (*shifter*) sebagaimana ditunjukkan pada gambar 3. Keluaran dari perkalian parsial ini kemudian

masuk ke selektor 4 ke 1 dengan logika selektor ditentukan oleh dua bit dari Y. Secara keseluruhan sistem perkalian dengan algoritma ini dinyatakan pada gambar 4.



Gambar 3. Implementasi perkalian parsial pada perkalian *array*



Gambar 4. Sistem perkalian array dengan modifikasi

2.2. Perkalian *Booth*

Pada tahun 1951, Andrew Donald Booth mengembangkan algoritma yang dikenal sebagai algoritma Booth (Booth, 1951). Perkalian Booth berlaku baik untuk format *signed* maupun *unsigned* dengan bentuk 2's *complement*. Konsep dasarnya dengan memanfaatkan Booth algoritma *encoding* untuk mengurangi jumlah perkalian parsial. Berikut ini langkah-langkah perkalian dengan algoritme Booth.

Contoh perkalian $2_{10} \times (-4)_{10}$ atau 0010×1100, maka langkah pertama yang dilakukan pada algoritme Booth adalah menentukan mana yang akan dijadikan pengalinya. Kriterianya adalah yang perubahan bit berurutannya paling sedikit. Pada 0010 terjadi dua kali perubahan dari LSB yaitu dari '0' ke '1' dan dari '1' ke '0'. Sedangkan pada 1100 hanya terdapat satu perubahan bit yaitu '0' ke '1', sehingga 1100 menjadi pengali (X), dan 0010 menjadi yang dikalikan (Y).

Langkah berikutnya cari nilai –Y dengan konsep 2s complement atau sama dengan 1110. Kemudian buat tabel yang terdiri dari nilai U, V, X dan X-1. U dan V awalnya berisi nol dan di akhir proses nanti merupakan hasil perkalian. X-1 berisi bit LSB sebelum isi bit di X digeser. Operasi yang dilakukan pada U dan V tergantung dari kombinasi LSB di kolom X dan nilai X-1. Kombinasi-kombinasi tersebut ditunjukkan di tabel 1.

Tabel 1. Kombinasi koding operasi

LSB X	X-1	Operasi
0	0	Geser ke kanan
0	1	$U_n = U_{n-1} + Y$ Kemudian di geser ke kanan
1	0	$U_n = U_{n-1} - Y$ Kemudian digeser ke kanan
1	1	Geser ke kanan

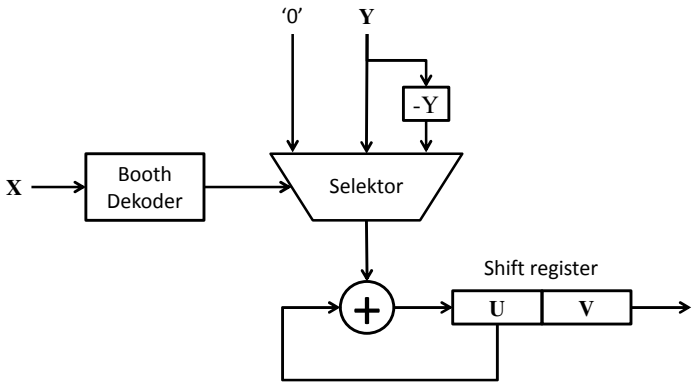
Untuk 4 bit pengali, maka akan terdapat empat step sampai diperoleh hasil perkalian. Keempat step ini ditunjukkan dalam tabel 2 di bawah ini. Pada awalnya nilai X dimasukkan ke kolom X. Nilai U, V dan X-1 di-set pada nilai '0'. Perhatikan kombinasi antara LSB kolom X dan X-1. Karena kombinasi yang terjadi adalah '0'-'0', maka step berikutnya adalah pergeseran ke kanan saja. Perhatikan kembali nilai LSB dari X dan X-1. Kombinasi yang terjadi masih '0'-'0', sehingga proses yang terjadi hanya pergeseran saja. Hal yang sama juga terjadi di step ke-2.

Tabel 2. Proses perkalian Booth

Step	U	V	X	X-1
0	0000	0000	1100	0
1	0000	0000	0110	0
2	0000	0000	0011	0
3a	1110	0000	0011	0
3b	1111	0000	1001	1
4	1111	1000	1100	1

Pada step ke-3, karena di step sebelumnya kombinasi LSB dari X dan X-1 adalah '1'-'0' maka berdasarkan tabel 1, proses yang dilakukan adalah menambahkan U dengan –Y kemudian digeser ke kanan seperti ditunjukkan oleh step 3 di tabel 2. Karena kombinasi di step ke-3 adalah '1'-'1' maka pada step ke-4, proses yang dilakukan hanya pergeseran ke kanan. Sehingga hasil akhir perkalian tersebut adalah "11111000".

Teknik perkalian Booth dapat diimplementasikan dengan melibatkan dekoder Booth, selektor, sebuah penjumlah dan register geser sebagaimana ditunjukkan pada gambar 5. Booth dekoder dibuat berdasarkan tabel 1 yang umumnya dapat disusun dari beberapa gerbang logika dasar. Demikian juga dengan selektor dan 2's komplemen (-Y) yang diimplementasikan dari beberapa gerbang logika dasar. Sehingga kompleksitas terbesar ada di fungsi penjumlahan.



Gambar 5. Implementasi sistem perkalian Booth

Masalah utama dalam algoritme Booth adalah urutan proses yang cukup lama. Semakin panjang bit yang dikalikan maka semakin lama proses perkalian yang dilakukan. Untuk itu beberapa modifikasi dilakukan untuk mempercepat proses komputasi pada algoritma ini. Salah satu modifikasi yang dilakukan adalah dengan pemrosesan langsung 2 bit sekaligus yang tentu saja menggunakan aturan yang berbeda dengan aturan di tabel 1. Dengan demikian untuk perkalian 4 bit hanya diperlukan dua kali step atau lebih cepat dua kali dari sebelumnya. Modifikasi lainnya adalah dengan melakukan proses secara paralel untuk mempercepat siklus perhitungan. Beberapa hasil modifikasi pada algoritma Booth telah dilakukan oleh Elguibaly, 2000, Jagadeech, dan Chary, 2012, serta Yeh dan Jen, 2000.

3. HASIL DAN PEMBAHASAN

Pada bagian ini akan disajikan analisis perbandingan kedua teknik perkalian tersebut. Parameter yang dibandingkan terdiri dari frekuensi maksimum, beban komputasi dan latency.

Perbandingan parameter dibuat dengan acuan implementasi perkalian 4 bit dan 8 bit di perangkat *Field Programmable Gate Array* (FPGA).

3.1. Frekuensi Maksimum

Frekuensi maksimum merupakan salah satu variabel yang menentukan seberapa cepat komputasi itu dapat dijalankan. Ia ditentukan oleh jalur (*path*) terpanjang pada proses komputasi yang dilakukan. Pada perkalian *array* 4 bit seperti ditunjukkan gambar 2, jalur terpanjang terdiri dari sebuah gerbang logika dan dua buah penjumlah. Sedangkan pada perkalian Booth (gambar 5), jalur terpanjang dari proses dekoder sampai penjumlahan yang terdiri dari beberapa gerbang dasar dan sebuah penjumlahan. Pada kasus ini nampaknya keduanya memiliki kecepatan yang sama, karena sebuah penjumlahan 4 bit dapat direalisasikan dalam beberapa gerbang logika dasar, namun jalur pada penjumlahan 4 bit lebih panjang daripada proses dekoder dan selektor pada perkalian Booth. Lebih lanjut pada kasus perkalian 8 bit, nampak perbedaan yang cukup berarti, dimana pada perkalian *array*, jalur terpanjang terdiri dari beberapa gerbang dasar dan tiga buah penjumlahan. Sedangkan jalur terpanjang pada perkalian Booth masih tetap sama. Sehingga pada perkalian Booth dimungkinkan penerapan frekuensi *clock* yang lebih tinggi daripada perkalian *array*.

Frekuensi maksimum berbanding terbalik dengan *delay* maksimum. Implementasi pada FPGA Virtex-II Pro XC2VP30 (Xilinx, 2003), diperoleh *delay* maksimum untuk perkalian *array* 4 bit dan 8 bit adalah 17,15ns dan 19,33ns, sehingga frekuensi *clock* masimumnya adalah 58,3MHz dan 51,7MHz. Sedangkan teknik perkalian Booth untuk 4 bit dan 8 bit berturut-turut mempunyai *delay* maksimum 10,44ns dan 12,33ns, sehingga frekuensi maksimumnya adalah 95,8MHz dan 81,1MHz. Dari data tersebut diketahui bahwa pada perkalian *array* penurunan frekuensi maksimum yang diakibatkan oleh penambahan lebar bit lebih besar daripada perkalian Booth. Sehingga untuk parameter frekuensi *clock*, perkalian dengan algoritma Booth memiliki keunggulan daripada *array*. Namun ketika teknik *pipeline* diimplementasikan pada perkalian *array* maka akan diperoleh *delay* maksimum yang lebih kecil daripada teknik perkalian Booth.

3.2. Beban Komputasi

Beban komputasi dinilai dari seberapa banyak komponen yang terlibat di dalam komputasi tersebut. Pada perkalian *array* 4 bit, komponen yang terlibat adalah beberapa gerbang logika dasar dan tiga buah penjumlahan. Sedangkan perkalian Booth 4 bit terdiri dari beberapa gerbang logika dasar dan sebuah penjumlahan. Nampaklah bahwa komplesitas perkalian *array* di atas perkalian Booth. Komplesitas perkalian array akan menjadi sekitar dua kali lipat ketika lebar bit menjadi dua kali lipat. Sedangkan pada perkalian Booth, komplesitas relatif sama yaitu hanya beberapa gerbang logika dasar dan sebuah penjumlahan.

Hasil implementasi pada FPGA menunjukkan bahwa untuk implementasi perkalian 4 bit dengan teknik *array* diperlukan 30 *Look Up Tables* (LUTs), sedangkan dengan teknik Booth diperlukan 25 LUTs. Untuk perkalian 8 bit, teknik *array* memerlukan 72 LUTs, sedangkan Booth memerlukan 34 LUTs. Sehingga dari segi beban komputasi teknik perkalian Booth mempunyai beban komputasi yang lebih ringan daripada teknik *array*.

3.3. *Latency*

Latency dihitung berdasarkan banyaknya siklus *clock* yang diperlukan dari data masuk sampai dengan hasil komplet diperoleh. Pada perkalian *array*, hasil perkalian diperoleh bersamaan dengan data masuk. Jika disisipkan register di akhir komputasi maka *latency* perkalian *array* adalah satu *clock*. *Latency* pada perkalian dengan metode Booth sangat dipengaruhi oleh lebar bit angka yang dikalikan. Untuk perkalian 4 bit, perkalian Booth memerlukan empat kali siklus *clock* untuk mendapatkan hasil yang benar. Demikian juga untuk perkalian 8 bit, maka diperlukan 8 siklus *clock* untuk mendapatkan hasil yang benar. Sehingga dilihat dari sisi *latency*, perkalian *array* lebih unggul daripada perkalian Booth karena memiliki *latency* yang lebih kecil.

Kecepatan proses komputasi sangat berhubungan dengan frekuensi *clock* dan *latency*. Pada bahasan frekuensi maksimum untuk perkalian 4 bit diperoleh frekuensi clock maksimum yang dapat diterapkan pada komputasi *array* 4 bit adalah 58,3MHz. Sedangkan frekuensi maksimum yang dapat digunakan pada perkalian Booth 4 bit adalah 95,8MHz. Dengan menggunakan frekuensi tersebut dan nilai *latency* 1 clock untuk perkalian *array* dan 4 clock untuk perkalian Booth maka kecepatan komputasi perkalian pada *array* dan Booth masing-masing adalah 17,15ns dan 41.76ns. Sehingga dari segi kecepatan komputasi, perkalian *array* lebih unggul daripada metode Booth.

Secara keseluruhan perbandingan unjuk kerja kedua teknik perkalian disajikan dalam Tabel 3 berikut ini.

Tabel 3. Perbandingan keseluruhan

Teknik perkalian	Frekuensi maksimum	Beban komputasi	Latency	Kecepatan proses
Array	Lebih rendah	Lebih ringan	Panjang	Cepat
Booth	Lebih tinggi	Lebih berat	Singkat	Lebih lambat

4. KESIMPULAN

Pada makalah ini telah dibahas mengenai dua teknik implementasi perkalian, yaitu teknik *array* dan Booth. Teknik implementasi perkalian array memerlukan beberapa gerbang logika dasar dan beberapa buah perjumlahan yang komplesitasnya tergantung dari lebar bit. Sedangkan implementasi perkalian terdiri dari dekoder, selektor, register geser dan sebuah penjumlahan. Sehingga dari segi komplesitas perkalian Booth memerlukan sumber daya komputasi yang lebih rendah daripada perkalian array.

Tanpa tambahan modifikasi apapun pada kedua teknik tersebut, implementasi pada perangkat FPGA Virtex-II Pro XC2VP30, diperoleh hasil bahwa delay maksimum perkalian array lebih besar daripada perkalian Booth. Namun latency perkalian Booth jauh lebih besar daripada perkalian array. Sehingga dari hasil perhitungan diperoleh bahwa kecepatan proses perkalian array lebih cepat daripada perkalian Booth.

DAFTAR PUSTAKA

Actel Corporation, (1996), *Implementing Multipliers with Actel FPGAs*, Application Note AC108.

Booth, A.D., (1951), *A signed binary multiplication Technique*, Quart. Journ. and Applied Math., vol.IV Pt.2, pp. 236–240.

Elguibaly, F., (2000), *A fast parallel multiplier–accumulator using the modified Booth algorithm*, IEEE Trans. Circuits Syst., vol. 27, no. 9, pp. 902–908.

Ghosh, M., (2007), Design and Implementation of Different Multipliers Using VHDL, *Bachelor Thesis*, Dept. of Elect. and Comm. Eng., National Inst. of Tech. Rourkelain.

Jagadeech, S., Chary, S.V., (2012), *Design of Parallel Multiplier–Accumulator Based on Radix-4 Modified Booth Algorithm with SPST*, International Journal Of Engineering Research And Applications (IJERA), vol. 2, no. 5, pp.425–431.

Laxman, S., Prabhu, R.D., Mahesh, S.S., Manjula, B.M., and Sharma, C., (2012), *FPGA Implementation of Different Multiplier Architectures*, Int. Journal of Emerging Technology and Advanced Engineering, vol.2, no.6, pp. 292–295.

Yeh, W.C., and Jen, C.W., (2000), *High-speed Booth encoded parallel multiplier design*, IEEE Trans. Comput., vol. 49, no. 7, pp. 692–701.

Xilinx, (2003), *Virtex-II Pro™ Platform FPGAs: Complete Data Sheet*, Advance Product Specification.