ALGORITMA MULTIPLIKASI KECEPATAN TINGGI DENGAN MATEMATIKA *VEDIC*

HIGHSPEED MULTIPLICATION ALGORITHM WITH VEDIC MATHEMATIC

Putri Ghina Khusnul Fuadah[1], Iswahyudi Hidayat[2], Estananto[3]

[1,2,3]Prodi S1 Teknik Elektro, Fakultas Teknik Elektro, Universitas Telkom

[1]putrigkf@icloud.com [2]iswahyudihidayat@telkomuniversity.ac.id [3]estananto@telkomuniversity.ac.id

Abstrak

Prosesor berkecepatan tinggi pada sistem komputasi sangat bergantung pada proses operasi matematika, karena operasi matematika merupakan salah satu blok perangkat keras utama dari sebagian besar sistem DSP *(Digital Signal Processing)*. Multiplikasi merupakan operasi dasar matematika yang paling mendominasi waktu eksekusi pada prosesor jika dibandingkan operasi lainnya. Makalah ini menyajikan teknik multiplikasi berkecepatan tinggi 8x8 bit yang sangat berbeda dengan sistem multiplikasi konvensional, karena metode yang diusulkan berlandasan pada struktur *Vertical* dan *Crosswise* dari matematika *Vedic*. Matematika *Vedic* memfasilitasi beberapa solusi sampai batas tertentu. Sistem multiplikasi dirancang menggunakan matematika *Vedic*, dikodekan dalam bahasa VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) serta diimlementasikan menggunakan *board* FPGA Altera DE-1. Hasil analisa dan implementasi dibandingkan dengan metode multiplikasi konvensional dan metode Booth radix-4 untuk menunjukan peningkatan efisiensi yang signifikan dalam waktu *delay*. Waktu *delay* berkurang sebesar 1,231 *clock* untuk *slow model* dan 0,413 *clock* untuk *fast model* dibandingkan dengan waktu *delay* menggunakan metode konvensional.

Kata kunci: Matematika *Vedic*, VHDL, Multiplikasi, FPGA, *delay*, *Ripple Carry Adder*, Arsitektur.

Abstract

In computing systems, high-speed processors rely on mathematical operations. Mathematical operations are part of the main hardware block of most digital signal processing systems. Multiplication is basic mathematical operation that dominates the processor execution time when compared to other operations. This paper presents high-speed 8x8 bit multiplication technique that is very different from conventional multiplication systems, since the proposed method is based on Vertical and Crosswise structures of Vedic mathematics. Vedic Mathematics facilitates some solutions to some extent. The proposed multiplication system using Vedic math is encoded in VHDL (Very High Speed Integrated Circuit Hardware Description Language) and implemented using the Altera DE-1 FPGA board. The results of the analysis and implementation are compared with conventional multiplication method and Booth radix-4 method to show significant efficiency improvement in time delay. The delay time is reduced by 1,231 clock for slow model and 0,413 clock for fast model compared to the delay time if using conventional methods.

***Keywords**: Vedic Mathematic, VHDL, multipliers, FPGA, delay, Ripple Carry Adder, architecture.*

1. Pendahuluan

Seiring dengan perkembangan zaman, kebutuhan pemrosesan data berkecepatan tinggi telah meningkat sebagai akibat dari perkembangan pemrosesan sinyal dan aplikasi komputer. Untuk mengatasi masalah tersebut, optimasi terhadap sistem komputasi yang terdiri dari berbagai operasi matematika ini terus dilakukan. Salah satu operasi aritmatika penting dalam aplikasi tersebut adalah dengan melakukan perhitungan matematis skala besar dalam waktu yang sangat singkat. Dan diantara operasi matematika, multiplikasi merupakan operasi dasar yang memiliki kompleksitas dan area paling tinggi dibandingkan operasi lainnya. Karena dalam melakukan perhitungan matematika terutama multiplikasi komputer menghabiskan cukup banyak waktu pemrosesan, sehingga peningkatan kecepatan processor matematika untuk melakukan multiplikasi akan meningkatkan kecepatan keseluruhan komputer.

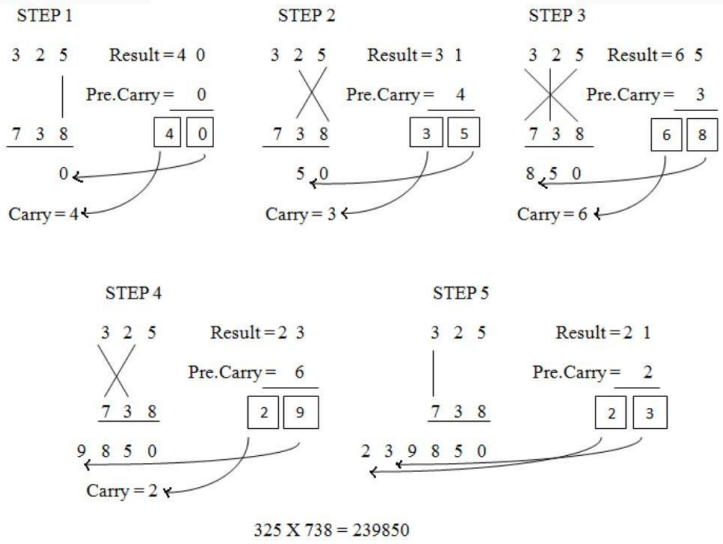
Multiplikasi merupakan salah satu fungsi dasar pada operasi aritmatika yang mempunyai peran utama dalam berbagai sistem digital. Operasi berbasis multiplikasi seperti *Multiply and Accumulate unit* (MAC), konvolusi, *Fast Fourier Transform* (FFT) dan filter banyak digunakan dalam berbagai macam aplikasi pemrosesan sinyal dan data. Kompleksitas suatu sistem digital pada umumnya ditentukan oleh seberapa banyak multiplikasi yang terlibat di dalamnya. Kecepatan proses komputasi juga sangat dipengaruhi oleh proses multiplikasi yang ada dikarenakan proses multiplikasi pada umumnya merupakan bagian yang memerlukan waktu paling lama (Ghosh, 2007). Dibandingkan dengan penjumlahan dan pengurangan, multiplikasi memerlukan sumber daya dan area yang paling besar. Pada kenyataannya, 8.72% dari keseluruhan tahapan dari suatu unit proses adalah multiplikasi (Laxman dkk., 2012) [1]. Sehingga arsitektur multiplikasi yang tepat merupakan hal esensial dalam kinerja optimasi rangkaian

digital. Dengan demikian, multiplikasi berperan penting dalam menentukan kinerja sistem karena mendominasi waktu eksekusi sistem dan pengoptimalan sistem multiplikasi dalam segi kecepatan dan daya.

Unjuk kerja suatu proses multiplikasi ditentukan oleh kecepatan proses komputasi dan ukuran area pemrosesan. Kecepatan proses merupakan suatu takaran untuk menilai efisiensi dari suatu sistem, karena kecepatan pemrosesan selalu menjadi kendala dalam sistem operasi multiplikasi, maka dalam kasus ini harus diberi perhatian khusus. Peningkatan kecepatan dapat dicapai dengan cara mengurangi jumlah langkah penyelesaian dalam proses perhitungan dengan menggunakan teknik multiplikasi yang mumpuni.

Sampai saat ini terdapat berbagai macam teknik multiplikasi dengan keunggulan dan kelemahannya masing-masing dan teknik multiplikasi matematika *Vedic* dapat digunakan secara efisien untuk mengatasi kasus ini.

Matematika *Vedic* direkonstruksi dari kitab suci India kuno (*Vedic*) oleh Swami Bharati Krishna Tirthaji Maharaja (1884-1960) dengan waktu delapan tahun penelitian. Matematika *Vedic* didasarkan pada 16 prinsip atau rumusan kata yang disebut sebagai Sutra yang dapat diterapkan pada berbagai cabang matematika seperti aljabar, trigonometri, geometri, dll. Metodenya bekerja dengan cara mengurangi perhitungan kompleks menjadi metode yang lebih sederhana karena berasal dari metode yang serupa dengan cara kerja pikiran manusia, sehingga membuatnya menjadi lebih mudah di pahami [2]. Penggunaan konsep matematika *Vedic* dalam algoritma komputasi suatu prosesor akan mengurangi kompleksitas eksekusi waktu, kecepatan, dan *delay*. Matematika *Vedic* didasarkan pada konsep baru di mana proses generasi suatu produk parsial dapat dilakukan bersamaan dengan proses penambahannya (paralel). Paralelisme dalam menghasilkan produk parsial dan penjumlahannya diperoleh dengan menggunakan *Urdhva-Tiryagbhyam* yang dijelaskan pada gambar 1.1 untuk multiplikasi 2 angka desimal (325 * 738). Seperti yang terlihat pada gambar 1.1 angka pada kedua sisi garis dikalikan dan ditambahkan dengan *carry* dari langkah sebelumnya, dan menghasilkan satu bit dari hasil dan *carry*. *Carry* ditambahkan pada langkah selanjutnya dan proses terus berlanjut. Karena sebagian produk dan jumlahnya dihitung secara paralel, maka proses multiplikasi tidak bergantung pada frekuensi prosesor. Dengan demikian, menggunakan Sutra *Urdhva-Tiryagbhyam* dalam multiplikasi biner menghasilkan jumlah langkah yang dibutuhkan untuk menghitung produk akhir akan berkurang dan oleh karena itu terjadi pengurangan waktu komputasi dan peningkatan kecepatan multiplikasi. Oleh karena itu, mengintegrasikan matematika *Vedic* untuk mendesain multiplikasi akan meningkatkan kecepatan operasi multiplikasi.



Gambar 1 Ilustrasi Sutra Urdhva-Tiryagbhyam pada multiplikasi [2].

Seperti halnya yang telah dilakukan oleh penelitian sebelumnya, telah dilakukan ujicoba penggunaan matematika *Vedic* dengan Sutra *Nikhilam* disertai *Carry Save Adder* dan pemrosesan paralel. Pada percobaannya, hasil implementasi matematika *Vedic* berhasil menghasilkan *delay* yang cukup kecil yaitu 4 ns *delay* untuk multiplikasi 16x16-bit.

Dalam tulisan ini, multiplikasi dengan menggunakan matematika *Vedic* 8x8 bit dengan Sutra *Urdhva-Tiryagbhyam* akan diimplementasikan pada *board* FPGA Altera DE-1 dan membandingkan arsitektur multiplikasi yang diusulkan dengan arsitektur sebelumnya dalam segi kendala kecepatan dan memori.

2. Dasar Teori

2.1. Matematika *Vedic*

Matematika *Vedic* adalah sebuah metode atau teknik matematika yang dikembangkan oleh Swami Bharati Krishna Tirtha (1884-1960) dari kitab Atharva *Vedic*. Matematika *Vedic* ini dinilai sangat menarik karena menyediakan beberapa algoritma efektif yang dapat diterapkan pada berbagai cabang teknik, seperti pemrosesan dan komputasi sinyal digital. Banyak metoda *Vedic* yang dianggap sebagai terobosan baru, sederhana, dan inovatif.

Matematika *Vedic* merupakan prinsip-prinsip yang terkandung dalam 16 Sutra (sastra) dan 13 sub bab Sutra dari kitab Atharwa *Vedic* yang kemudian dikembangkan oleh Swami Bharati Krishna Tirtha. Berikut adalah Sutra beserta maknanya:

1. *(Anurupye) Shunyamanyat* - Jika ada dalam rasio, yang lainnya nol.
2. *Chalana-Kalanabyham* - Perbedaan dan Kesamaan.
3. *Ekadhikina Purvena* - Lebih dari yang sebelumnya.
4. *Ekanyunena Purvena* - Dengan yang kurang dari yang sebelumnya.
5. *Gunakasamuchyah* - Faktor jumlahnya sama dengan jumlah faktornya.
6. *Gunitasamuchyah* - Produk dari jumlah tersebut sama dengan jumlah produk.
7. *Nikhilam Navatashcaramam Dashatah* - Semua dari 9 dan terakhir dari 10.
8. *Paraavartya Yojayet* - Transpose dan sesuaikan.
9. *Puranapuranabyham* - Dengan selesainya atau tidak selesai.
10. *Sankalana- vyavakalanabhyam* - Sebagai tambahan dan dengan pengurangan.
11. *Shesanyankena Charamena* - Sisanya dengan digit terakhir.
12. *Shunyam Saamyasamuccaye* - Bila jumlahnya sama dengan jumlah nol.
13. *Sopaantyadvayamantyam* - Yang terakhir dan dua kali yang kedua dari belakang.
14. *Urdhva-Tiryagbhyam* - Vertikal dan melintang.
15. *Vyashtisamanstih* - Bagian dan Utuh.
16. *Yaavadunam* - Apapun tingkat kekurangannya.

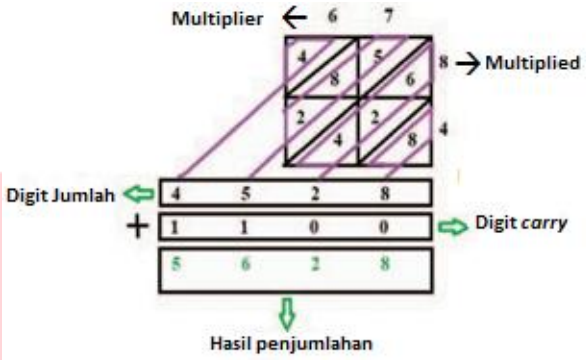
Berdasarkan konsep diatas untuk aplikasi pemrosesan sinyal digital, banyak peneliti telah mengusulkan arsitektur multiplikasi dan unit komputasi lainnya. Dari penelitian ini, mereka telah membuktikan bahwa algoritma komputasi aritmatika yang diajukan lebih efisien dalam segi kecepatan dan area di bandingkan dengan algoritma konvensional. Prabir Saha dan Arindam Banarjee telah mengusulkan sebuah desain algoritma multiplikasi dengan menggunakan konsep matematika *Vedic*. Mereka merancang dan menganalisis multiplikasi *Vedic* menggunakan Sutra *Nikhilam*. Dengan metode multiplikasi ini, dia menghilangkan langkah-langkah multiplikasi yang tidak diinginkan dengan angka nol dan berhasil menghasilkan *delay* yang cukup kecil yaitu 4 ns untuk multiplikasi 16x16 bit dibandingkan dengan algoritma konvensional. Dengan ini, dia mencapai multiplikasi berkecepatan tinggi [2].

Salah satu metode multiplikasi yang diajarkan dalam buku *Vedic mathematics method* diberi nama *Urdhva-Tiryagbhyam.* Sutra *Urdhva-Tiryagbhyam* akan di pakai sebagai landasan untuk membuat algoritma multiplikasi pada topik ini.

2.1.1 Sutra *Urdhva-Tiryagbhyam*

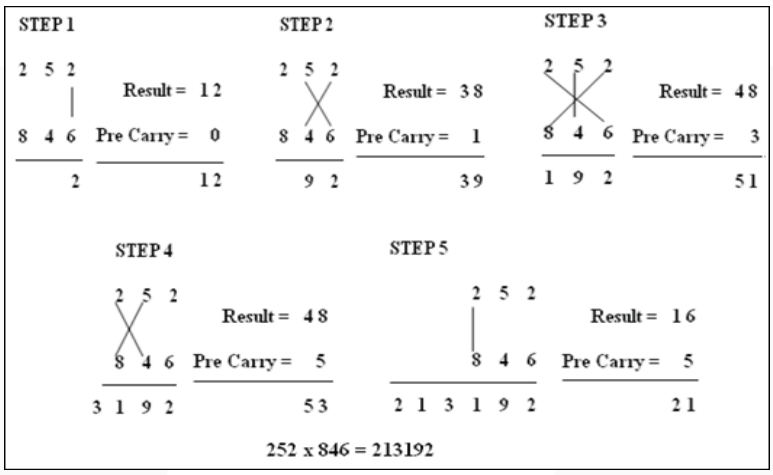
Metode multiplikasi yang dirancang didasarkan pada sebuah algoritma yang disebut dengan *Urdhva-Tiryakbhyam* dalam Istilah Sanskerta berarti ‘vertikal dan melintang’, *Urdhva-Tiryakbhyam* berasal dari salah satu Sutra didalam matematika *Vedic* India kuno. Sutra *Urdhva-Tiryakbhyam* adalah formula multiplikasi umum yang berlaku untuk semua kasus multiplikasi. Pada algoritma ini proses penjumlahan produk parsial dan proses produksi dari seluruh produk parsial dilakukan secara paralel, karenanya maka prosesor tidak bergantung pada frekuensi *clock* [6]. Keuntungannya di sini proses paralelisme dapat mengurangi kebutuhan prosesor untuk beroperasi pada frekuensi *clock* yang lebih tinggi, karena semakin tingginya frekuensi *clock* pada suatu prosesor maka semakin tinggi pula daya yang dibutuhkan prosesor yang berdampak pada disipasi daya. Dengan memakai metode multiplikasi *Vedic*, seluruh kerugian yang terkait dengan peningkatan disipasi daya dapat direduksi dikarenakan metode ini cukup cepat dalam pemrosesan, efisien dalam segi tata letak dan memiliki struktur yang cukup teratur. Keuntungan dari penggunaan multiplikasi *Vedic* yaitu dari jumlah bit, *delay* pada *gate* dan luas area *gate* yang lebih hemat dibandingkan dengan multiplikasi konvensional. Sehingga berdampak pada *speed*, *area* dan *power* dinilai lebih efisien [4].

Gambar 2 mengambarkan prosedur umum dari multiplikasi bilangan desimal 2×2. Skema multiplikasi dapat dipahami dengan mempertimbangkan prosedur multiplikasi dua bilangan desimal (67 * 84). Prosedur diawali dari angka yang akan dikalikan ditulis pada dua sisi persegi berturut-turut seperti yang ditunjukkan pada gambar 2.2. Masing-masing kotak kecil dipartisi menjadi dua bagian yang sama oleh garis melintang. Setiap digit *multiplier* kemudian dikalikan secara independen dengan setiap digit dari *multiplied* dan produk dari dua digit ditulis dalam kotak umum. Semua digit yang terletak di atas kotak ungu ditambahkan lalu menghasilkan jumlah dan *carry*. Hasil akhirnya diperoleh dengan penambahan digit jumlah dan digit *carry* sebelumnya. Untuk langkah pertama, *carry* diasumsikan menjadi nol.



Gambar 2 Implementasi multiplikasi dengan menggunakan Sutra Urdhva-Tiryagbhyam.

Ilustrasi berikutnya yaitu dengan mempertimbangkan multiplikasi dua bilangan desimal. Bilangan yang digunakan saat ini sebesar 3x3 digit dimisalkan dengan 252 * 846. Jika dihitung menggunakan dengan metode *Urdhva-Tiryagbhyam* dapat diperhatikan seperti yang ditunjukkan pada gambar 2.1. Prosedur diawali dari angka pada kedua sisi garis yaitu 2 dan 6, kedua angka tersebut dikalikan (6 * 2 = 12). Dari prosedur tersebut menghasilkan dua bit, yaitu satu bit untuk produk akhir (yaitu 2) dan bit lainnya sebagai *carry* (yaitu 1). Bit *carry* ini akan ditambahkan pada langkah selanjutnya, oleh sebab itu proses terus berlanjut. Jika angka yang dihasilkan dalam satu langkah lebih dari satu bit, maka semua kelebihannya disalurkan ke *carry* selanjutnya. Pada setiap *step* perkalian, *Least Significant Bit* (LSB) bertindak sebagai bit hasil akhir dan bit yang tersisa bertindak sebagai *carry* untuk langkah selanjutnya. Untuk langkah pertama, *carry* diasumsikan menjadi nol [3].



Gambar 3 Multiplikasi dua bilangan desimal 252 * 846

bilangan biner menggunakan algoritma Booth, ditemukan pada tahun 1950 oleh Andrew Donald Booth. Metode ini menggunakan notasi *two's complement* dan *right shift arithmetic* bilangan biner. Sistem yang menggunakan metode Booth umumnya memiliki tujuan untuk mencapai hasil yang lebih cepat daripada metode konvensional. Beberapa contoh sistem yang menggunakan metode ini adalah FIR, *microprocessor*, dan *digital signal* processor[1].

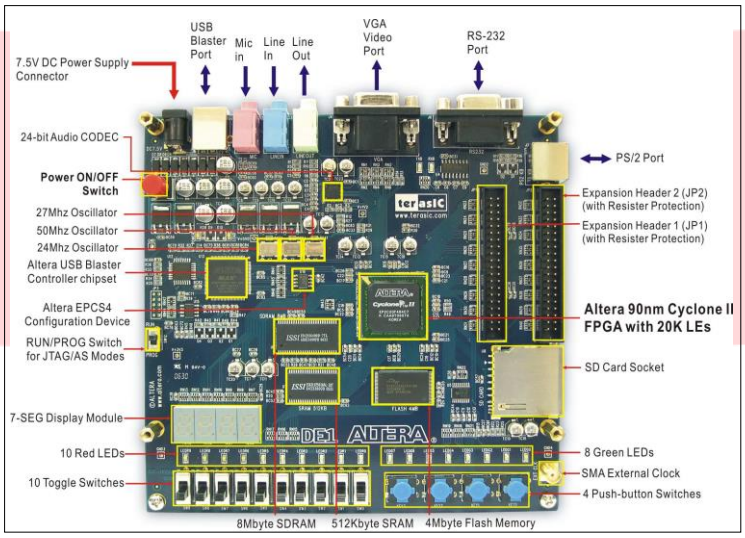
Algoritma perkalian Booth yang paling dasar disebut dengan Booth radix-2. Sedangkan penulis menggunakan metode Booth yang sudah dikembangkan berupa Booth radix-4. Booth radix-4 mengurangi setengah dari jumlah *partial product* yang digunakan pada Booth radix-2[1].

2.2 FPGA Altera DE-1

FPGA adalah sejenis IC yang isinya bisa diprogram. Yang dimaksud dengan "diprogram" adalah bukan diprogram seperti halnya mikrokontroler. Mikrokontroller diprogram dengan diisi perangkat lunak didalamnya. Sedangkan FPGA diprogram dengan mengatur konfigurasi struktur rangkaian yang ada didalamnya. Konfigurasi rangkaian yang ada didalam FPGA bisa diatur atau diubah dengan bahasa VHDL. Pengaturan ini bisa dilakukan berkali-kali.

Didalam FPGA secara umum terdapat sekumpulan blok yang berisi gerbang- gerbang logika dan blok interkoneksi yang belum disambung satu sama lain. Penyambungan ini yang dilakukan ketika pengubahan konfigurasi FPGA dilakukan. Umumnya arsitektur internal FPGA berbeda tergantung perusahaan pembuatnya. Blok-blok internal FPGA disebut Logic Cell didalam FPGA buatan Xilinx, dan Logic Element untuk FPGA buatan

Altera. Selain blok-blok tersebut, terdapat beberapa fitur tambahan seperti PLL (Phase-Locked Loops), DSP *slice*, *multiplier*, dll.



Gambar 4 Spesifikasi dan Layout Board Altera DE-1

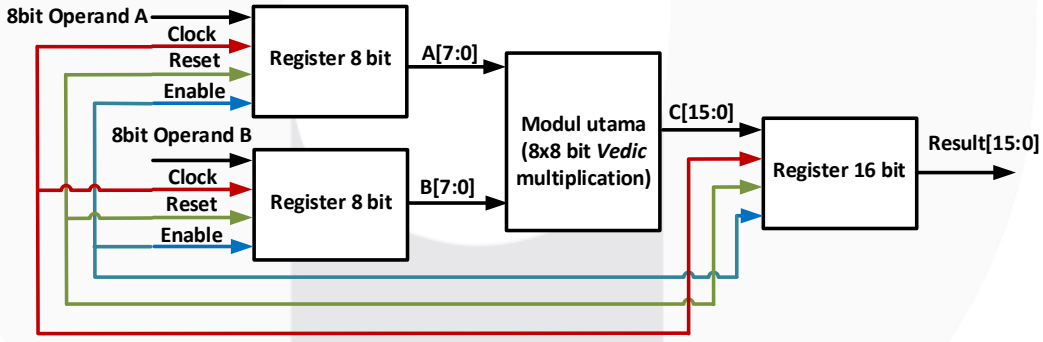
Selain dibuatnya FPGA, diproduksi juga papan pengembangan FPGA, yaitu sejenis papan rangkaian yang terdapat FPGA dan *peripheral* pendukung seperti Ethernet, HDMI, audio dll dengan tujuan membantu perancang dalam melakukan desain. Untuk FPGA buatan Altera, papan pengembangan biasanya dibuat oleh Terasic. Contoh papan buatan Terasic adalah DE1, yang dilengkapi dengan FPGA Altera Cyclone II yang memiliki 20000 buah LE (logic- element).

Altera DE-1 merupakan jenis FPGA buatan dari *Altera* yang umumnya sering digunakan untuk *development* dan *education*, sehingga memiliki spesifikasi yang cukup lengkap untuk penggunaan *board* dalam penelitian ini. Perhatikan gambar 2 untuk tampilan *layout* dan komponen dari *Altera* DE-1 seperti gambar 4.

3. Pembahasan

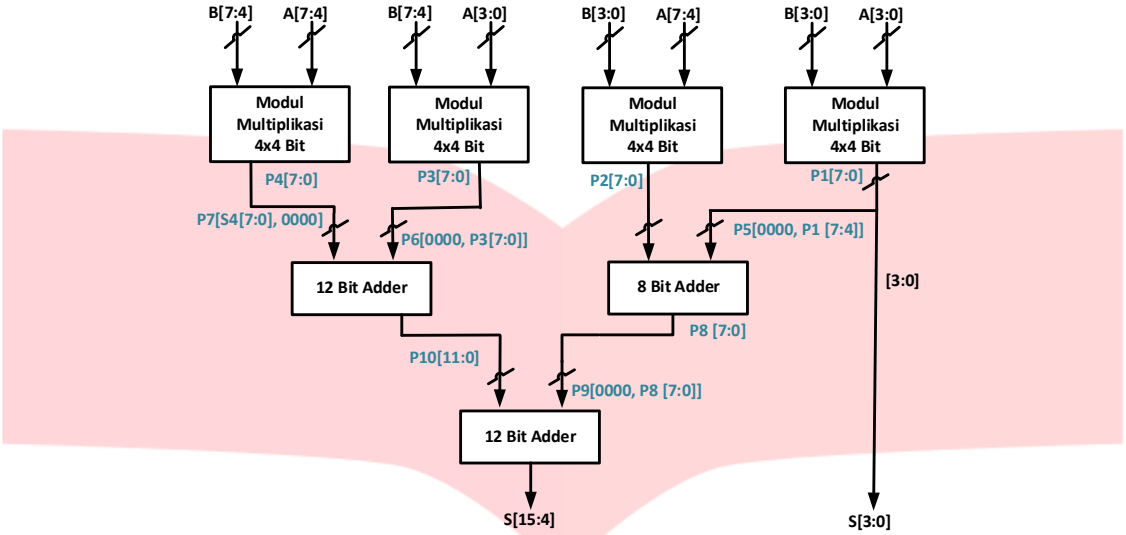
3.1 Desain Blok Sistem

Komposisi komponen pada perancangan modul multiplikasi ini terbagi menjadi 3 bagian yaitu register input, modul utama, dan register output. 2 register input berukuran 8 bit diletakan pada awal modul utama, berfungsi untuk menyimpan nilai input sebelum masuk ke dalam sistem multiplikasi. Dan digunakannya 1 register output berukuran 16 bit pada akhir modul utama yang berfungsi untuk menyimpan data output dari modul utama. Seperti yang dapat ditunjukan pada gambar 3.2.



Gambar 5 Blok Diagram keseluruhan system

Proses pengkodean modul utama menggunakan algoritma *Vedic* sutra *Urdhva-Tiryakbhyam*. Konsep dasar algoritma *Vedic* dijelaskan pada bagian landasan teori pada bab II. Pada bab ini, matematika *Vedic* diterjemahkan dalam bentuk arsitektur multiplikasi yang diusulkan, yang selanjutnya akan diimplementasikan dalam bahasa VHDL. Arsitektur multiplikasi 8x8 bit menggunakan algoritma *Vedic* dapat ditunjukan sebagai berikut:

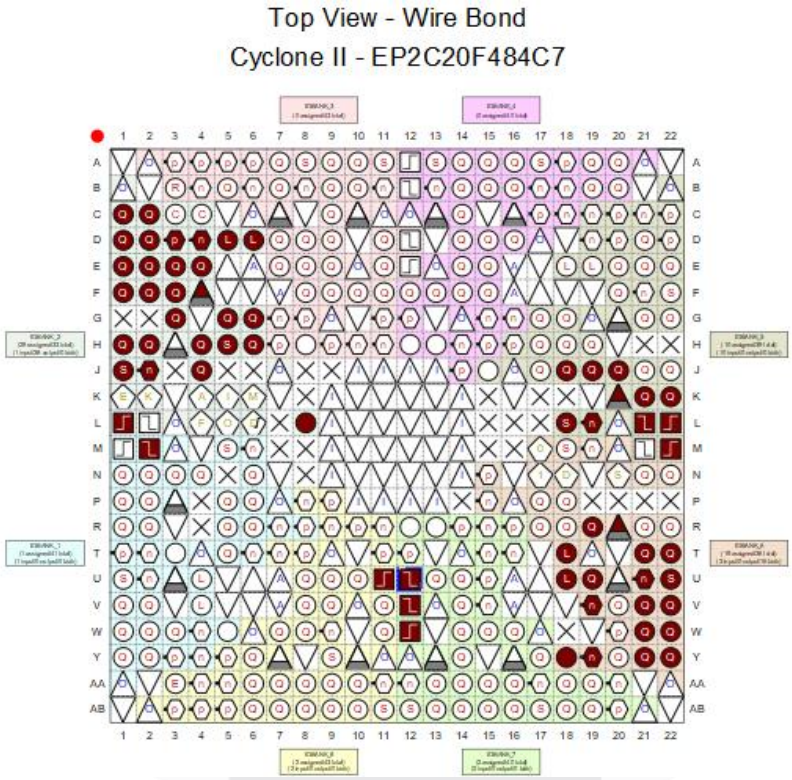


Gambar 6 Arsitektur multiplikasi *Vedic* 8x8 bit.

Modul ini diimplemetasi dengan menggunakan teknik *square*, yaitu dengan cara membagi *operand* menjadi 4 bagian. Sehingga sistem dapat dibangun menggunakan modul multiplikasi dengan bit yang lebih kecil. Tujuan dari penggunaan teknik *square* untuk meminimalisir penggunaan *logic gate* dan memudahkan pengembangan arsitektur multiplikasi dengan bit yang lebih tinggi.

3.2 Sintesis Pada Sistem Multiplikasi

Operasi multiplikasi yang telah dirancang akan implementasikan kedalam *board field-programmable gate array* (FPGA). Pada penelitian ini, *board* FPGA yang digunakan yaitu Altera DE-1, analisis dan sintesis dilakukan menggunakan Quartus II. program yang disintesis ke *board* FPGA harus melakukan pemetaan *Pin Planener* terlebih dahulu. Proses ini merupakan proses untuk mendeskripsikan penempatan *port* modul pada pin FPGA yang akan digunakan dalam *board* FPGA



Gambar 7 Penggunaan *Pin* pada *Board Altera DE*-1 sesuai dengan Sistem

. Untuk modul multiplikasi *Vedic* 8x8 bit digunakan 8 pin switch sebagai *port* input 8 bit *multiplier*, 8 pin GPIO (*General Port Input* Output) sebagai *port* input 8 bit *multiplicand*, 1 *port clock*, 2 *port* push button sebagai *port reset* dan *enable* dan 4 buah seven segmen sebagai *port* 16 bit *output*. Data *port* dan pin dapat dilihat pada tabel berikut.

3.4 Analisis dan Perbandingan Hasil Sintesis

Analisis hasil sintesis pada penelitian ini berfokus pada kecepatan sistem multiplikasi yang dapat dilihat pada waktu *delay* sistem dan juga berfokus pada penggunaan *resource board* FPGA. Data waktu *delay* sistem dan besar *resource* yang dipakai pada *board* FPGA dari sistem operasi multiplikasi *Vedic* 8x8 bit akan dibandingkan dengan multiplikasi konvensional (metode *Shift and Add*).

Implementasi sistem multiplikasi *Vedic* 8x8 bit pada *board* FPGA Altera DE-1 menghasilkan *report* implementasi yang terlihat pada tabel 1.

Tabel 1 Report perbandingan penggunaan *Resource*

Vedic				*Shift and Add*			
Total Logic Element		*Logic Usage by Number of LUT Input*		*Total Logic Element*		*Logic Usage by Number of LUT Input*	
Combinational with No Register	187	*4 Input Function*	151	*Combinational with No Register*	77	*4 Input Function*	59
Register Only	0	*3 Input Function*	22	*Register Only*	0	*3 Input Function*	27
Combinational with Register	32	*<=2 Input Function*	46	*Combinational with Register*	25	*<=2 Input Function*	16
Total	219/18.752 (1%)	*Register Only*	0	*Total*	102/18.752 (<1%)	*Register Only*	1

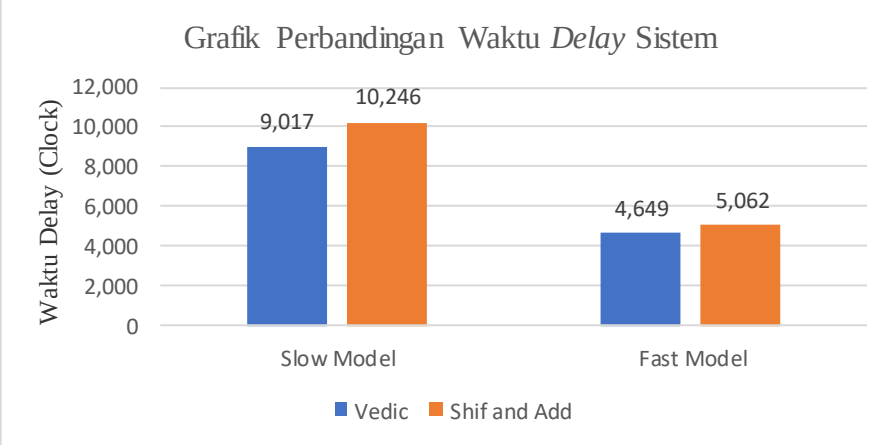
Berdasarkan hasil sintesis pada tabel 1, dapat diperhatikan bahwa jumlah area *resource* yang termakan pada sistem multiplikasi menggunakan metode *shift and add* lebih sedikit dibanding menggunakan algoritma *Vedic*. Jumlah *resource* yang dibutuhkan oleh *shift and add* adalah sebesar 102 dari 18.752 *element*, atau kurang dari 1% dari *total logic elemen* yang tersedia oleh *board*. Algoritma *Vedic* membutuhkan 219 dari 18.752 atau sekitar 1% dari *logic resource* yang tersedia pada *board* FPGA, hal ini disebabkan metode *shift and add* merupakan metode yang jauh lebih sederhana sehingga komponen penyusun modulnya hanya memerlukan sedikit ruang *chip* dibandingkan dengan metode *Vedic*. Sedangkan metode *Vedic multiplier* membutuhkan lebih banyak *resource* dikarenakan metode *Vedic* dibangun dari modul-modul perkalian yang lebih kecil, sehingga lebih banyak *logic element* yang di butuhkan jika dibandingkan dengan metode *shift and add*.

Perbandingan waktu *delay* mencakup pada *slow model* dan *fast model*. *Slow model* menunjukkan analisis ketika kecepatan memproses lambat, nilai tegangan yang diterima adalah minimum dan nilai *temperature* bernilai *maximum*. Hal sebaliknya terjadi pada analisis *fast model*, kecepatan memproses cepat, nilai tegangan bernilai *maximum*, dan *temperature minimum*.

Tabel 2 Hasil Sintesis berdasarkan Waktu *Delay*

Vedic		*Shift and add*	
Slow Model	*Fast Model*	*Slow Model*	*Fast Model*
9,017 clk	4,649 clk	10,248 clk	5,062 clk

Berdasarkan dari tabel 2, dapat diperoleh grafik perbandingannya sebagai berikut:



Grafik 1 Nilai Perbandingan Waktu *Delay*

Bila dibandingan pada grafik 4.1, waktu *delay* yang paling rendah ketika kondisi *slow model* dan *fast model* adalah menggunakan metode *Vedic* jika dibanding dengan metode *shift and add* yaitu sebesar 10,248 *clock* pada keadaan *slow* model dan 5,062 *clock* pada keadaan *fast model*. Metode *Vedic* lebih cepat dikarenakan menggunakan teknik multiplikasi paralel.

4. Kesimpulan

Berdasarkan hasil pengujian dan analisis yang dilakukan pada sistem multiplikasi 8x8 bit, maka dapat disimpulkan beberapa hal sebagai berikut:

1. Implementasi algoritma *Vedic* pada sistem operasi multiplikasi 8x8 bit berhasil dilakukan pada *board* FPGA Altera DE-1 menggunakan Bahasa deskripsi perangkat keras VHDL.
2. Dari hasil penelitian yang dilakukan oleh penulis, jumlah *resource* pada board yang digunakan oleh metode *Vedic* adalah 219 dari 18.752 element yang tersedia. Sedangkan untuk nilai *delay* pada board yang telah disimulasikan untuk *fast model* adalah sebesar 4,649 *clock* dan 9,107 *clock* untuk *slow model*.
3. Dibandingkan dengan multiplikasi konvensional yang implementasikan pada perangkat FPGA Altera DE-1, diperoleh hasil bahwa *delay* maksimum perkalian *Vedic* lebih kecil dari pada metode lainnya. Namun *resource* metode *Vedic* lebih besar daripada metode lainnya.
4. Jumlah *resource* yang digunakan pada metode perkalian bilangan biner tidak mempengaruhi nilai *delay* pada sistem.
5. Dari hasil perhitungan diperoleh bahwa kecepatan proses perkalian menggunakan metode *Vedic* lebih cepat dari pada perkalian metode konvensional.

Daftar Pustaka:

[1] Pedroni, Volnei A (2004), *Circuit Design with VHDL*, MIT Press Cambridge, Massachusetts, London, England.
[2] Pong P Chu (2008), *FPGA Prototyping By VHDL Examples*, John Wiley & Sons, Inc, Hoboken, New Jersey.
[3] *What are* FPGAs?". http://www.fpga4fun.com/fpgainfo1, 2009.
[4] "Field Programmble *Gate* Array", http://en.wikipedia.org/wiki/Field- programmable.*gate*.array, 2009.
[5] Anantha P. Chandrakasan, and Donald E. Troxel, "3D FPGA Design and CAD Flow, chapter 2", MTL/RLE Groups, 2006.
[6] Himanshu Thapliyal, and Hamid Arabnia, "A Time-Area- Power Efficient *Multiplier* and Square Architecture Based On Ancient Indian *Vedic* Mathematics", Proceedings of the International Conference on Embedded Sistems and Applications, 2004.
[7] Nanda A. Fakhri, "Perancangan dan Implementasi Algoritma FFT 64 Titik Menggunakan Menggunakan Multipath *Delay* Commutator pada FPGA", e-Proceeding of Engineering, 2016.
[8] "Altera DE-1 *Board*", https://www.altera.com/solutions/partners/partner-profile/terasic-inc-/*board*/altera-DE-1-*board*.html, 2010.
[9] Nafi, Ghazali Al. 2012. PERANCANGAN DAN IMPLEMENTASI ENCODER SFBC PADA LTE ARAH DOWNLINK BERBASIS FPGA. Institut Teknologi Telkom : Tidak diterbitkan.
[10] Naik, Suketu. 2015. Digital System. Weber State University, 2016.