Prosiding
ANNUAL RESEARCH SEMINAR 2016
6 Desember 2016, Vol 2 No. 1

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id

Studi Pergeseran Paradigma Komputasi Tepat-Waktu

Sukemi
Jurusan Sistem Komputer
Universitas Sriwijaya
Indralaya Ogan Ilir, Indonesia
Sukemi@unsri.ac.id

Abstrak—**Sebuah prosessor yang dibangun dengan umpan/masukan waktu yang variabel atau disebut dengan *time driven* processor diharapkan mampu bekerja optimal pada batasan waktu (*deadline*) yang ditetapkan walaupun hal ini belum berlaku bagi arsitektur komputer secara umum. Pendekatan awal akan dilakukan dengan memilah atau bahkan memotong bagian dari *task* yang bersifat mandatori dan optional dari sebuah penjadwal (*scheduling*). Hasil dari pendekatan ini adalah sebuah prosessor yang memiliki pemisah/pemangkas prioritas dari sebuah *task*. Pendekatan kedua, merancang sebuah prosessor yang memiliki kemampuan sebagai penghitung dengan tingkat ketelitian yang beragam (*variable precision computing*). Pendekatan terakhir dilakukan dengan 'menambah' kepastian presisi berupa interval aritmetika yang mampu memotong data/*task*. Potongan 'tersebut' berupa batasan atas dan batasan bawah dari area (*bounds*). Ke-tiga pendekatan ini dapat dibangun secara *orthogonal* menjadi sebuah prototipe prosessor yang memiliki lebar bit yang bervariasi (4, 8, 16, 32 dan 64 bit) dengan menambahkan sebuah selektor sebagai pemilah lebar bit yang optimal untuk menghasilkan optimalitas waktu komputasi. Hasil akhir dari representasi unit aritmetika dengan simulasi Altera Quartus II menunjukkan bahwa sub unit aritmetika Adder 4, 8, 16, 32 dan 64 bit orthogonal dengan selektor dan kolektor memberikan optimalisasi waktu proses eksekusi komputasi sebesar 4,53% jika dibanding tidak menggunakan selektor dan kolektor.**

***Katakunci : Variabel bitspace,* kendali waktu, interval bound.**

I. PENDAHULUAN

Salah satu faktor kunci dalam sebuah sistem yang dikonsep untuk meningkatkan kehandalan terhadap perubahan yang cepat (misalnya, perubahan kondisi lingkungan) adalah dihasilkannya jawaban yang benar pada saat yang tepat (*timing correctness*). Konsep ini dikenal dengan nama sistem tepat-waktu (*real time system*). Ketepatan dan keterlambatan waktu keluar jawaban merupakan ragam karakter yang muncul dalam sebuah hasil proses komputasi. Ketepatan waktu keluar jawaban ditentukan oleh suatu rentang waktu-tanggap (*response time*) yang berbeda untuk setiap sistem, tergantung pada tuntutan lingkungan tempat sistem bekerja yang bervariasi dari rentang waktu yang sangat singkat hingga rentang waktu yang lama. Keterlambatan waktu suatu sistem (atau subsistem) untuk menghasilkan jawaban yang benar dan tepat waktu, berpotensi memunculkan resiko yang mengancam keberhasilan sistem lainnya [1]. Sebagai contoh, pada sistem pengambilan keputusan berbasis komputer, kegagalan mengeluarkan hasil operasi komputasi (misal, nilai numerik dari suatu representasi data) secara tepat waktu dapat menyebabkan terlambatnya proses pengambilan keputusan atas hasil akhir komputasi tersebut.

Beragam pendekatan yang telah diujicobakan untuk memenuhi tenggat-waktu atas sistem perhitungan tepat-waktu. Keragaman ini dapat dibagi ke dalam dua kelompok yakni kelompok perangkat lunak, misalnya tingkat sistem operasi yang fokus mengembangkan sistem penjadwalan *Rate Monotonic*, Lehoeczky et al., (1989), *Non Preemptive*, (Jeffay et al., 1991; Parks, Lee, 1995), *Earliest Deadline First* (EDF), Stankovic et al., (1998), *Increase Reward Increase Service* (IRIS), Dey et al., (1993), *Imprecise Computation*, Liu et al., (1991) dan di tingkat kelompok perangkat keras yang berupaya melalui percepatan *clock* prosessor atau melalui pembesaran granulasi operasi dengan ukuran *k*-bit, *Real Time Arithmetics Unit*, Mora-Mora et al., (2006).

Kedua kelompok pendekatan perangkat lunak dan perangkat keras *belum* memberikan solusi akhir sistem tepat-waktu, walaupun telah dikembangkan teknik penjadwalan baru. Selain itu, disebabkan juga karena sebagian besar hasil publikasi tentang kelompok-kelompok ini sulit diterapkan secara praktis akibat masih terbatasnya riset bidang lain yang dapat saling menunjang implementasinya.

Selain itu, percepatan *clock* prosessor yang bertujuan untuk membangun sistem perhitungan yang cepat dilakukan dengan usaha meminimalkan nilai rata-rata tenggat-waktu sejumlah *task* dalam suatu proses [2]. Hal ini berbeda dengan tujuan pembangunan sistem tepat-waktu yang berusaha memenuhi tenggat-waktu setiap *task*. Namun, hal yang paling penting dari suatu sistem tepat-waktu adalah *predictability*, bukan kecepatan dengan memiliki makna yang sangat relatif. *Predictability* diartikan sebagai kepastian perilaku fungsional

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id

dan temporal dari setiap *task*, yang harus se-deterministik mungkin sehingga cukup mampu untuk memenuhi spesifikasi sistem. Setiap perhitungan yang cepat akan sangat membantu dalam usaha pemenuhan tenggat-waktu yang pendek, tapi perhitungan cepat saja tidak dapat menjamin *predictability* suatu sistem tepat-waktu.

Kenyataannya, sistem tepat-waktu bekerja dengan benar hanya apabila tidak terjadi kegagalan pewaktuan. Namun, kegagalan pewaktuan akan terjadi dalam jumlah yang bervariasi saat sistem mengalami kondisi *overload* (akibat perubahan kondisi lingkungan) sehingga sistem harus mengatur ulang *task-task* yang harus dikerjakan untuk menanggapi perubahan tersebut. Padahal dalam sistem tepat-waktu kondisi *overload* tidak dapat dihindari.

Derajat/tingkat akurasi turut ditentukan oleh keterbatasan waktu yang disediakan untuk tercapainya sistem tepat-waktu. Oleh karena itu, diperlukan batasan atau *range* hasil perhitungan yang dapat memberikan jawaban untuk menentukan pengambilan keputusan. Penentuan akurasi suatu hasil perhitungan aritmetika secara umum tidak dapat dilakukan sebelum eksekusi perhitungan itu selesai, akibatnya kegagalan perhitungan menjadi rentan jika waktu yang tersedia tidak mencukupi. Jika digabungkan antara akurasi dan pewaktuan maka sangat menentukan tingkat presisi suatu operasi, misalnya operasi aritmetika.

Selain beberapa hal (pewaktuan, akurasi dan presisi) yang diperlukan pemilihan waktu, akurasi dan presisi menjadi hal yang menarik untuk ditetapkan agar sistem 'masih' memenuhi ke tepat-waktu-annya. Hingga saat ini, belum ada mekanisme yang dapat memerankan fungsi pemilih tersebut yang dapat diwakilkan oleh perangkat lunak maupun keras yang dapat bertugas untuk itu. Setiap pemilihan tentu memerlukan seleksi dengan parameter waktu, akurasi dan presisi agar sistem tepat-waktu lebih terjamin ketercapaiannya [1].

II. SPESIFIKASI SISTEM

Kandungan aritmetika adder pada model yang dirancang akan ditempatkan di setiap *variabel bitspace* yang akan dibangun secara bertingkat pada gambar 1 [3].

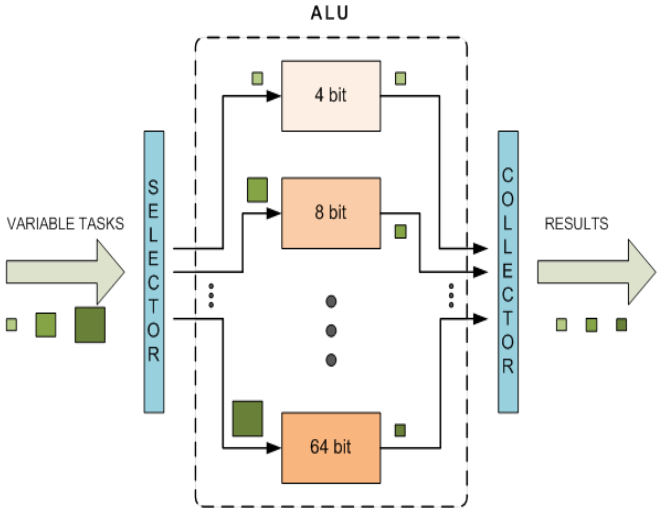
Semua fungsi aritmetika adder dibangun pada sebuah simulator perangkat lunak Altera Quartus II dalam pilihan granulasi (4 bit, 8 bit, 16 bit, 32 bit dan 64 bit) dan masing-masing akan dihubungkan dengan sebuah pemilah yang diletakkan dibagian muka arsitektur aritmetika. Pemilah ini sekaligus berfungsi sebagai penseleksi *bitspace* (istilah satuan granulasi) untuk lebih menjamin ketercapaian tepat-waktu sebagai paradigma baru dalam sebuah proses komputasi khususnya komputasi aritmetika.

Variabel bitspace 4, 8 hingga 64 bit rencananya dibangun pada LUT yang tersedia didalam FPGA dengan fungsi yang berbeda-beda untuk melaksanakan tugasnya masing-masing. *Variabel bitspace* adder yang dirancang ini juga memiliki selector [1] muka yang bertugas menseleksi lebar *task* untuk dikirim pada *variabel bitspace* adder yang ada. Hal ini dilakukan dalam upaya mencari tingkat optimalitas yang tinggi untuk waktu komputasi disetiap unit aritmetika yang akan dibangun secara bertingkat.

III. PROSES PEMILIHAN

Komponen selektor *task* berfungsi sebagai pengatur penggunaan sumber daya, dalam hal ini presisi *variabel bitspace* adder 4 bit, 8 bit, 16 bit, 32 bit dan 64 bit sesuai dengan jenis dan ukuran *task* yang masuk. Jadi sedikit berbeda dari arbitrase bus arbiter yang hanya menetapkan sumber daya pada sedikit lebar yang sama, misalnya 32 bit atau 64 bit saja.

Jadi selain mengatur jenis selektor *task* juga memperhatikan lebar bit *task* masuk sebelum diproses di *variabel bitspace* adder. Dengan menggunakan komponen ini,

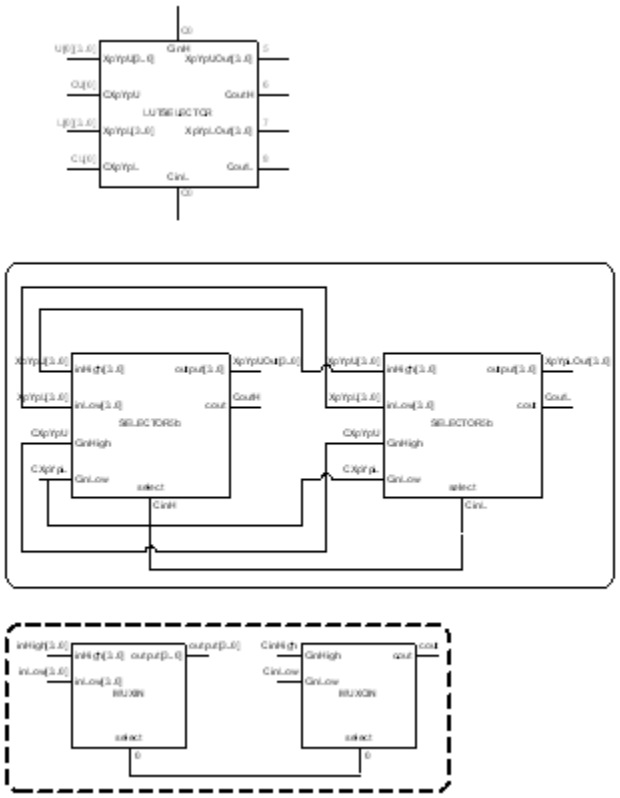


Gambar 1. *Variabel bitspace* adder [3].

waktu yang diharapkan dari pelaksanaan *task* atau serangkaian *task* diprediksi lebih cepat secara tepat-waktu.

Selain selektor task yang memiliki fungsi penseleksi *variabel bitspace*, juga dirancang selektor akurasi yang dibangun didalam fungsi aritmetika adder guna memperoleh nilai akurasi meskipun proses komputasi belum selesai secara keseluruhan. Rangkaian selektor akurasi diberikan pada gambar 2.

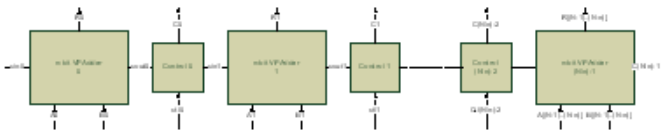
ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id



Gambar 2. Selektor akurasi pada *variabel bitspace* adder

Untuk menguji kemampuan selektor akurasi yang *terbenam* pada *variabel bitspace* adder pada gambar 2 perlu dilakukan uji mandiri bagi fungsi selektor akurasi itu sendiri. Uji mandiri selektor akurasi dilakukan dengan cara merancang blok diagram rangkaian seperti pada gambar 3.

Selektor akurasi ini dirancang agar berfungsi digunakan sinyal kontrol *ctrl* untuk menentukan ukuran presisi yang diinginkan dari sebuah operasi matematis. Sinyal kontrol ini dipergunakan oleh rangkaian kontrol yang berada diantara komponen *variabel bitspace* adder seperti yang diperlihatkan pada gambar 3[4].



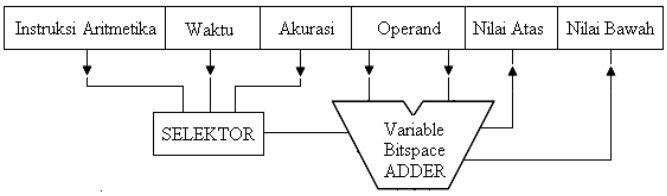
Gambar 3. Diagram blok *variabel bitspace* adder [4]

Penjabaran lebih lanjut dari rancangan yang dilakukan dengan batasan bahwa N merupakan lebar data instruksi, misalkan 8 bit, 16 bit, 32 bit, 64 bit atau 128 bit. Sedangkan *n* adalah ukuran blok granulasi dari *variabel bitspace* adder yang dipergunakan untuk memproses instruksi matematis tersebut. Jika lebar operand adalah 64 bit dan ukuran blok granulasi adalah 8 bit maka akan ada 6 blok kontrol (*control*) dengan sinyal *control*nya masing-masing dan 8 blok granulasi *variabel bitspace* adder. Blok *control* akan mengontrol ukuran presisi yang diinginkan setiap Blok *control* akan mengontrol ukuran presisi yang diinginkan dari setiap operasi berdasarkan nilai 0 atau 1 pada sinyal *control* nya. Jika bernilai 0 maka antar granulasi *variabel bitspace* adder terhubung sehingga presisi menjadi 2*n*-bit. Sebaliknya jika bernilai 1 maka antar granulasi *variabel bitspace* adder terputus sehingga presisi hanya sebesar *n*-bit. Jadi jika sinyal *control* bernilai 0 semua maka presisi sebesar N-bit dan sebaliknya presisi sebesar *n*-bit. Untuk uji mandiri ini diterapkan sinyal *ctrl* bernilai 1.

IV. PRINSIP PERANCANGAN

4.1. Arsitektur

Perancangan *variabel bitspace* adder *bertingkat* disusun dari LUT FPGA yang berjumlah 16 blok dengan lebar bit granulasi $k = 4$ dengan asumsi bahwa lebar operand 64 bit. Masing-masing LUT mengandung 2 buah *full adder* yang akan melakukan fungsi penjumlahan dengan hasil batas atas dan batas bawah, seperti yang diperlihatkan pada gambar 4.



Gambar 4. Blok komputasi Aritmetika Adder *Variabel Bitspace*

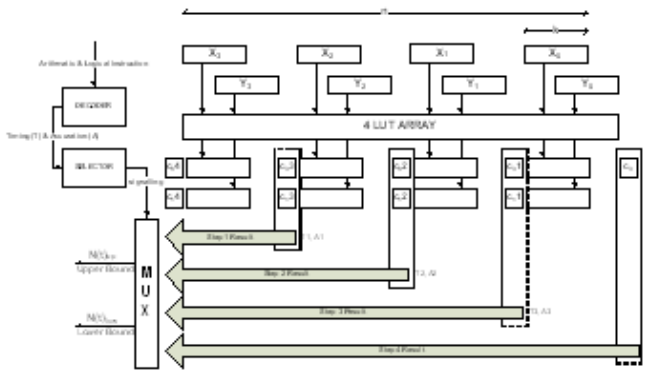
Perbedaan perhitungan komputasi terutama pada upaya yang dilakukan guna mencari optimalitas waktu yakni dengan memasang selektor yang berfungsi sebagai penseleksi *task* dengan tanpa penempatan selektor.

Unit aritmetika *variabel bitspace* adder yang dirancang dengan kriteria sebagai berikut:

1. Dapat memilih *bitspace* sesuai lebar bit *task* yang akan melakukan proses komputasi;
2. Dapat memberikan kepastian waktu jawaban dihasilkan dengan akurasi yang dapat ditentukan sendiri optimalitasnya;
3. Mampu memberikan nilai jawaban pada dua besaran berupa nilai atas dan nilai bawah;
4. Mampu menghasilkan jawaban dengan akurasi yang tinggi sejak awal proses eksekusi.

Gambar 5 menunjukkan pola komputasi aritmetika *variabel bitspace* adder yang dirancang dengan lebar bit 16 bit dan $k = 4$.

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id



Gambar 5. Pola komputasi 16 bit *variabel bitspace* adder

Blok aritmetika 16 bit *variabel bitspace* adder pada gambar 5 yang mengandung rangkaian decoder, selektor, *k*-operator, multiplexer dan pola komputasi presisi variabel. Masing-masing rangkaian ini berfungsi untuk merealisasi tujuan pencapaian optimalitas waktu pada presisi variabel jika akurasi minimum telah tercapai.

V. PENGUJIAN USULAN ARSITEKTUR

Tahapan pengujian dan analisis hasil pengujian dilakukan untuk setiap arsitektur *variabel bitspace* adder yang telah dirancang baik secara *stand alone* maupun *orthogonal*.

Hasil pengujian unit aritmetika *variabel bitspace* adder 32 bit merupakan hasil integrasi konsep dari beberapa metode yang diterapkan dengan melalui simulasi tools Altera Quartus II Cyclone. Dari hasil simulasi unit ini akan dicari nilai akurasi yang optimal pada waktu yang dikehendaki (*time driven*).

Rekapitulasi hasil pengujian ketercapaian waktu untuk mendapatkan nilai akurasi pada *variabel bitspace* adder 4, 8, 16, 32 dan 64 bit diberikan pada tabel 1.

Tabel 1. Rekapitulasi hasil waktu ketercapaian akurasi [1].

Task (bit)	*Variabel bitspace* adder (bit)			
	64	32	16	8
64	80ns	x	x	x
32	80ns	60ns	x	x
16	80ns	60ns	27,5ns	x
8	80ns	60ns	27,5ns	7,5ns

Hasil pengujian unit aritmetika *variabel bitspace* adder yang dibangun secara *orthogonal* menghasilkan hasil pengujian di tabel 2.

Tabel 2. Rekapitulasi hasil waktu ketercapaian akurasi *orthogonal* multi-*clock* [3,4].

n / N	*8*	*16*	*32*	*64*
4	2,72 ns	4,56 ns	8,24 ns	30,70 ns
8	2,48 ns	5,03 ns	9,15 ns	34,02 ns
16	x	5,21 ns	9,68 ns	35,96 ns
32	x	x	9,79 ns	37,02 ns
64	x	x	x	38,34 ns

Merujuk pada tabel 2, terbukti apabila lebar *task* 8 bit dijalankan pada *variabel bitspace* 32 bit akan mengkonsumsi waktu yang lebih lama, yakni sebesar 9,15 ns dibandingkan jika dijalankan pada *variabel bitspace* yang sesuai (8 bit) ternyata hanya mengkonsumsi waktu eksekusi 2,48 ns untuk menghasilkan akurasi awal.

Hasil pengujian unit aritmetika *variabel bitspace* adder yang dibangun secara *orthogonal* namun tidak mengandung selektor diawal *variabel bitspace* dengan hasil pengujian seperti yang ditunjukkan pada tabel 3:

Tabel 3. Rekapitulasi hasil waktu ketercapaian akurasi *orthogonal* tanpa selektor dan kolektor.

n / N	*8*	*16*	*32*	*64*
4	2,45ns	4,12ns	7,44ns	27,71ns
8	2,24ns	4,54ns	8,26ns	30,70ns
16	null	4,70ns	8,74ns	32,45ns
32	null	null	8,84ns	33,41ns
64	null	null	null	34,60ns

Berdasarkan tabel 3, terbukti apabila lebar *task* 8 bit dijalankan pada *variabel bitspace* 32 bit akan mengkonsumsi waktu yang lebih lama (8,26 ns) dibandingkan jika dijalankan pada *variabel bitspace* yang sesuai (8 bit) ternyata hanya mengkonsumsi waktu eksekusi 2,24 ns untuk menghasilkan akurasi awal dengan optimalitas waktu sebesar 6,02 ns.

Untuk hasil pengujian unit aritmetika *variabel bitspace* adder yang dibangun secara *orthogonal* dengan ditambahkannya rangkaian selektor diawal *variabel bitspace* dengan hasil pengujian seperti yang ditunjukkan pada tabel 4.

Tabel 4. Rekapitulasi hasil waktu ketercapaian akurasi *orthogonal* dengan selektor dan kolektor.

n / N	*8*	*16*	*32*	*64*
4	2,39 ns	3,96 ns	7,11 ns	26,33 ns
8	2,18 ns	4,36 ns	7,89 ns	29,17 ns
16	null	4,52 ns	8,34 ns	30,83 ns
32	null	null	8,44 ns	31,73 ns
64	null	null	null	32,86 ns

Merujuk tabel 4 terbukti apabila lebar *task* 8 bit dijalankan pada *variabel bitspace* 32 bit akan mengkonsumsi waktu yang lebih lama sebesar 7,89 ns dibandingkan jika dijalankan pada

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id

variabel bitspace yang sesuai (8 bit) ternyata hanya mengkonsumsi waktu eksekusi 2,18 ns untuk menghasilkan akurasi awal dengan optimalitas waktu sebesar 5,71ns.

Memperhatikan hasil pengujian *variabel bitspace* adder 4, 8, 16, 32 bit yang dibangun secara *orthogonal* dengan selektor dan kolektor memberikan optimalisasi waktu proses eksekusi komputasi sebesar 4,53% dibanding apabila tidak menempatkan selektor dan kolektor.

Secara keseluruhan jika diperhatikan dan dibandingkan antara tabel 1 dengan tabel 2 dan tabel 3 serta tabel 4, memperlihatkan bahwa unit aritmetika *variabel bitspace* adder yang dibangun secara *stand alone* membutuhkan waktu yang lebih lama untuk mendapatkan akurasi awal, jika dibandingkan dengan yang dibangun secara *orthogonal*. Hal ini membuktikan bahwa konsumsi waktu komputasi dapat ditekan untuk menghasilkan nilai akurasi awal. Jadi, dengan demikian perubahan paradigma komputasi sedikit demi sedikit mulai terjawab, yakni akurasi dapat diukur meskipun waktu komputasi/eksekusi belum selesai.

Begitu juga penempatan selektor dan kolektor diawal arsitektur *variabel bitspace* adder dapat mengoptimalkan waktu eksekusi sebesar 5 % dan juga membuktikan bahwa penempatan *variabel bitspace* adder yang variabel mampu juga mengoptimalkan waktu eksekusi sebesar 72,37%.

Berdasarkan hasil perhitungan menunjukkan bahwa *variabel bitspace* adder yang dibangun telah menghasilkan optimalitas waktu untuk mencapai akurasi awal namun telah sangat baik. Hal inilah yang mempertegas arah perubahan paradigma terhadap perhitungan aritmetika yang mengarah kepada aritmetika tepat-waktu.

VI. KESIMPULAN

Ternyata, penempatan selektor *task* diawal arsitektur *variabel bitspace* adder lebih menjamin optimalitas waktu sehingga prediktibilitas akurasi dapat ditentukan sebelum selesainya proses komputasi selesai. Optimalitas waktu dapat dihasilkan sebesar 5% lebih optimal jika menggunakan selektor dan kolektor.

VII. PENELITIAN KEDEPAN

Perubahan paradigma ini telah signifikan terjawab dan diharapkan lebih signifikan lagi, jika arsitektur *variabel bitspace* adder akan diuji pada data input yang berasal dari lingkungan sebenarnya (*weather, medical analysis*, data *mining*)

REFERENSI

[1] Sukemi, Ratna AAP., Sudibyo H. (2012): Priority based computation: A Study of paradigm shift on real time computation" *Compuatational Intelegence and Cybernetics (Cyberneticscom), 2012 IEEE International Conference on Digital Object Identifier*, 12-14 Juli 2012, Bali, Indonesia. Hal. 129-132.

[2] Stankovic, J.A., Spuri, M. dan Ramamritham, K. (1998): Deadline cheduling for Real-Time Systems : EDF and Related Algorithms. Kluwer Academic Publishers (Kluwer International Series in Engineering and Computer Science).

[3] Sukemi, Ratna A.A.P, Sudibyo H, 2014, Variable Bitspace of Variable Precision Processor, *Journal of Advanced in Information Technology*, accepted 24 Februari 2014: (not published yet).

[4] Sukemi, Ratna A.A.P, Sudibyo H, (2013): Variable Bitspace Architecture with Variable Precision Processor. *The 2013 Annual International Conference on Advances Technology in Telecomunication, Broadcasting, and Satellite (TelSaTech 2013)*, 2-3 Agustus 2013, Jakarta, Indonesia. Adv. Sci. Lett. 20, Hal. 531-533 (2014).

[5] Lehoczky, J., Sha, L. dan Ding, Y. (1989): The Rate Monotonic Scheduling Algorithm: Exact Characterization And. Average Case Behavior. Dalam: Proceedings of Real Time Systems Symposium. pp. 166-171.

[6] Parks, T.M. dan Lee, E.A. (1995): Non-Preemptive Real-Time Scheduling of Dataflow Systems. Dalam: IEEE International Conference on Acoustics, Speech, and Signal Processing.

[7] Dey, J.K., Kurose, James, F. dan Towsley, D. (1993): Efficient on-line processor scheduling for a class of IRIS (increasing reward withincreasing service) real-time tasks. Di: Proceedings of the 1993 ACM SIGMETRICS conference on Measurement and modeling of computer systems. pp. 217-228.

[8] Liu, J.W.S, Shih, W.K., dan Lin, K.J. (1991): Imprecise Computations. Dalam: Proceedings of the IEEE. 82 (1).

[9] Mora-Mora, H., Mora-Pascual, J. dan Garcia-Chamizo, J.M. (2006): Real-Time Arithmetic Unit. Dalam: Real-Time Systems. 34 , pp. 53-79.

[10] Kerlooza, Y,K., Gondokaryono, Y,S., and Mulyana, A. (2010): MSB-First Interval-Bounded Variable-Precision Real Time Arithmatic Unit. Submitted to ICT Journal, ITB, Juli 2010.

[11] Jeffay, K., Stanat, D.F., Martel dan C.U. (1991): On Non-Preemptive Scheduling of Periodic and Sporadic Tasks. Dalam: Proceedings of the 12th Real-Time Systems Symposium.

[12] wikipedia.org (2014), "time driven programming", in: http://en.wikipedia.org/wiki/Time-driven_programming.

[13] Jensen ED., Locke CD., Tokuda H.,(1985), "A-time-driven scheduling model for Real Time Operating Systems", in: IEEE CH2220-2/b5/0000/0112$01.00.

[14] Nielsen, asger Munk; Kornerup,Peter (1995): MSB-First Digital serial arithmatic",.In : Journal of Universal Computer science. 1(7).

[15] David J. Kinnement. (2007): Synchronization and Arbitration in Digital System., John Wiley., ISBN 978-0470-51082-7.

[16] Zadeh, Lotfi A (1994) : Fuzzy logic, Neural Network, and softcomputing".In: communication of the ACM.

[17] Schulte, M.J. (1996): A Variable-Precision, Interval Arithmetic Processor. PhD thesis University of Texas.

[18] Hormigo, J., Villalba, J. dan Schulte, M.J. (2000): A Hardware Algorithm for Variable-Precision Logarithm. Dalam: Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors.

[19] Tinder, R.F. (2000): Engineering Digital Design. 2. ed. San Diego, USA: Academic Press. — ISBN: 0-12-691295-5.