Jurnal Technoscientia, ISTA Yogyakarta, Edisi Agustus 2009.Vol. 2, No. 1 Agustus 2009, pp. 50-60 ISSN: 1979-8415

DEKOMPOSISI TUGAS-TUGAS *SOFTWARE-DEFINED RADIO* (SDR)

Eko Marpanaji[1,2]

[1]Jurusan Teknik Elektronika, Fakultas Teknik, Universitas Negeri Yogyakarta
[2]Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung,

ABTRACT

This paper addresses decomposition of the tasks of Software-Defined Radio (SDR) computation by taking Gaussian Minimum Shift Keying (GMSK) as a case study. The aim of this research was to produce a task graph of the SDR that was needed in scheduling of distributed computing with task parallelism using genetics algorithm. Based on the tasks decomposition and the evaluation of execution time of each task, we obtained a task graph of SDR computation. We conclude that there are 11 (eleven) tasks for the modulator dan 15 (fifteen) for the demodulator.

Keywords: *executiion time, gaussian minimum shift keying, height function, modulation, task graph, task-parallelism, software-defined radio.*

INTISARI

Makalah ini membahas dekomposisi tugas-tugas komputasi *Software-Defined Radio* (SDR) dengan mengambil studi kasus skema modulasi *Gaussian Minimum Shift Keying* (GMSK). Tujuan penelitian ini adalah menghasilkan sebuah grafik tugas (*task graph*) komputasi SDR yang diperlukan untuk proses penjadwalan komputasi terdistribusi dengan metoda paralelisme tugas (*task-parallelism*) menggunakan algoritma genetika. Berdasarkan hasil dekomposisi tugas dan pengujian waktu eksekusi masing-masing tugas, maka diperoleh sebuah grafik tugas komputasi SDR menggunakan skema modulasi GMSK yaitu 11 (sebelas) tugas untuk modulator dan 15 (lima belas) tugas untuk demodulator.

Kata Kunci: fungsi ketinggian, *gaussian minimum shift keying*, grafik tugas, modulasi, paralelisme tugas, *software-defined radio*, waktu eksekusi.

PENDAHULUAN

Kendala utama dalam mewujudkan sistem *Software-Defined Radio* atau disingkat SDR adalah dalam hal kebutuhan komputasi. Zheng (2007) menggunakan skema penjadwalan *weighted-selective* dan metoda paralelisme data (*data-parallelism*) dalam melakukan komputasi paralel sistem SDR dalam sebuah jaringan cluster PC.

Kelemahan sistem tersebut adalah: metoda paralelisme data menyebabkan semua *node* komputasi akan melakukan tugas yang sama untuk paket data yang berbeda. Sedangkan dan skema penjadwalan *weighted-selective* akan selalu memprioritaskan *node* yang memiliki kemampuan komputasi paling tinggi. Hal ini mengakibatkan adanya kemungkinan beberapa *node* tidak melakukan komputasi sama sekali karena nilai prioritasnya paling rendah. Dengan demikian, beban kerja komputasi menjadi tidak merata.

Makalah ini akan menjelaskan dekomposisi tugas-tugas komputasi dalam SDR untuk menghasilkan sebuah grafik tugas, dan merupakan bagian dari penelitian tentang skema penjadwalan komputasi terdistribusi menggunakan al-

[1] Karangmalang, Yogyakarta, 55281. Telp. (0274) 586168, eko@uny.ac.id,
[2] Jl. Ganesha 10 Bandung 40132, Tel. (022)-4254034, ekoaji332@students.itb.ac.id

goritma genetika dengan metoda paralelisme tugas atau *task-parallelism*.

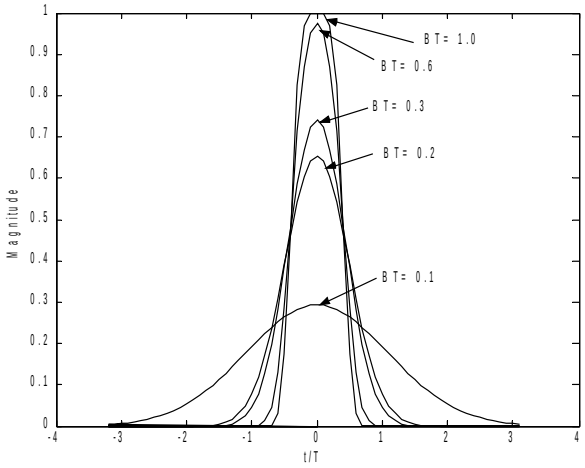
Tujuan penelitian tersebut adalah mengembangkan komputasi terdistribusi sebagai perbaikan terhadap kelemahan skema penjadwalan *weighted-selective* dan metoda paralelisme data yang digunakan oleh Zheng. Dengan menerapkan metoda paralelisme tugas, maka tiap-tiap tugas dapat dibagikan kepada seluruh *node* komputasi. Optimasi penjadwalan dilakukan dengan menggunakan algoritma genetika sehingga diperoleh sistem penjadwalan yang optimal dan pembagian beban kerja lebih merata.

Metoda yang digunakan dalam mengatasi permasalahan tentang penjadwalan komputasi terdistribusi menggunakan algoritma genetika dengan metoda paralelisme tugas diawali dengan melakukan identifikasi tugas-tugas komputasi SDR untuk berbagai jenis modulasi, melakukan dekomposisi tugas-tugas tersebut sehingga diperoleh sebuah grafik tugas komputasi SDR lengkap dengan parameter setiap *node* dalam grafik tersebut. Parameter tiap *node* dapat diperoleh dengan menghitung nilai ketinggian masing-masing tugas dan menguji waktu eksekusi masing-masing tugas.

Makalah ini akan menjelaskan bagaimana mewujudkan sebuah grafik tugas komputasi SDR dengan mengambil studi kasus modulasi *Gaussian Minimum Shift Keying* (GMSK). Waktu komunikasi antar *node* komputasi diabaikan.

Modulasi GMSK sangat populer dan digunakan dalam komunikasi seluler GSM. Hubungan antara *bandwidth* pada awal modulasi (B) dengan perioda bit (T) merupakan *bandwidth* dari sistem modulasi GMSK. Pada sistem seluler GSM nilai BT = 0.3 dengan laju bit saluran sebesar 270.8 kbps. Respons *filter* Gaussian ditentukan dengan persamaan 1 (Haykin, 2004):

$$g(t) = \frac{1}{2}\left[\mathbf{erfc}\left(\pi \sqrt{\frac{2}{\log 2}}\, BT_b \left(\frac{t}{T_b} - \frac{1}{2} \right) \right) - \mathbf{erfc}\left(\pi \sqrt{\frac{2}{\log 2}}\, BT_b \left(\frac{t}{T_b} - \frac{1}{2} \right) \right) \right] \quad (1)$$



Gambar 1. Respons *filter* Gaussian berdasarkan variasi nilai BT

Gambar 1 menunjukkan hasil simulasi respons *filter* Gaussian dengan menggunakan Matlab dengan variasi nilai BT (Marpanaji, 2007).

Untuk keperluan penjadwalan komputasi terdistribusi, maka tugas-tugas digambarkan dengan sebuah model grafik tugas (*task graph model*) atau

[1] Karangmalang, Yogyakarta, 55281. Telp. (0274) 586168, eko@uny.ac.id,
[2] Jl. Ganesha 10 Bandung 40132, Tel. (022)-4254034, ekoaji332@students.itb.ac.id

directed acyclic graph (DAG). Grafik tugas $TG = (V, E)$ adalah DAG yang memiliki *node-node* V (sekumpulan *m* tugas atau $V = \{T_1, T_2, ..., T_m\}$, dan sekumpulan garis berarah (*directed edge*) E yang menghubungkan *node* satu ke *node* lainnya. Garis berarah dilambangkan dengan $E = \{e_{ij}\}$ dimana e_{ij} adalah garis yang menghubungkan dua buah tugas yaitu tugas T_i menuju tugas T_j dan menunjukkan sebuah urutan atau hubungan preseden (*precedence relation*) diantara tugas tersebut dan biasanya dinotasikan dengan $>>$. Sebagai contoh, garis berarah $e_{ij} \in E$ dari T_i menuju T_j dapat dinotasikan $T_i >> T_j$, dan memiliki arti bahwa tugas T_i harus diselesaikan terlebih dahulu sebelum tugas T_j mulai dikerjakan (Hou, 1994; Lee, 2003; Rahmani, 2008). Selain itu, T_i adalah pendahulu (*predecesor*) atau leluhur (*ancestor*) dari T_j, atau biasa dutulis $T_i = PRED(T_j)$. T_j adalah pengikut (*successor*) atau anak (*child*) dari T_i, atau dapat ditulis $T_j = SUCC(T_i)$.

Untuk keperluan penjadwalan beberapa tugas, kita harus tahu tugas mana saja yang harus didahulukan dan tugas mana saja yang dapat dikerjakan kemudian, serta tugas-tugas mana saja yang dapat dikerjakan secara bersamaan (*concurrent*). Ada tidaknya hubungan preseden antar *node* dalam grafik tugas akan menentukan apakah *node* tersebut dapat dieksekusi secara seku-ensial atau paralel. Dua buah *node* da-pat dieksekusi secara bersamaan jika tidak ada hubungan preseden antara dua buah *node* tersebut.

Berdasarkan uraian tersebut, maka perlu dilakukan penentuan prioritas masing-masing tugas (*prioritizing of tasks*) dengan menghitung nilai ketinggian tiap-tiap tugas (*height of a task*) menggunakan fungsi ketinggian (*height function*). Nilai ketinggian tersebut menunjukkan hubungan preseden antar tugas. Jika T_i adalah leluhur T_j, maka T_i harus dieksekusi sebelum T_j, dan nilai ketinggiannya ditulis:

$$height(T_i) < height(T_j).$$

Dengan demikian, semakin kecil nilai ketinggian sebuah tugas, maka tugas tersebut akan semakin didahulukan, sebaliknya semakin besar nilai ketinggian sebuah tugas maka tugas tersebut akan semakin dieksekusi lebih akhir. Nilai ketinggian dapat dihitung menggunakan persamaan 4.

$$height(T_i) = \begin{cases} 0, & jika\ PRED(T_i) = \phi \\ 1 + \max\limits_{T_j \in PRED(T_i)} height(T_j), & untuk\ lainnya \end{cases} \quad (4)$$

Hou (1994) dan Tsujimura (1997) dalam menghitung fungsi ketinggian melakukan penambahan persamaan 5 untuk lebih mengoptimalkan penjadwalan, yaitu:

$$height'(T_j) = rand \in [\max\{height(T_i)\} + 1, \min\{height'(T_k)\} - 1] \quad (5)$$

untuk seluruh:

$T_i \in PRED(T_j)$ dan $T_k \in SUCC(T_j)$

Sedangkan Rahmani (2008) juga membuat modifikasi dalam menghitung nilai ketinggian sebuah tugas, yaitu:

$$height'(T_j) = rand \in X, \quad (6)$$

[1] Karangmalang, Yogyakarta, 55281. Telp. (0274) 586168, eko@uny.ac.id,
[2] Jl. Ganesha 10 Bandung 40132, Tel. (022)-4254034, ekoaji332@students.itb.ac.id

dimana:

$$\min height(T_k) - 1 \le X \le \max height(T_i) + 1$$

untuk, $\forall T_i \in PRED(T_j), T_k \in SUCC(T_j)$

Untuk keperluan penjadwalan, parameter tiap *node* yang digunakan adalah bobot tiap *node* (waktu eksekusi) dan nilai ketinggiannya yang ditulis disamping tiap-tiap *node* dengan format $(t_i, height(T_i))$ Sedangkan waktu komunikasi untuk sementara diabaikan dengan asumsi untuk komputasi paralel multiprosesor, perbedaan waktu komunikasi relatif kecil dan relatif sama.

PEMBAHASAN

Dalam mengembangkan komputasi paralel dengan metoda paralelisma tugas, langkah awal yang harus dilakukan adalah membagi sebuah proses yang besar menjadi satuan-satuan tugas yang lebih kecil, sehingga dapat dilakukan proses komputasi terdistribusi. Namun demikian, kadang-kadang sebuah proses tidak dapat dibagi menjadi beberapa satuan tugas yang dapat diproses secara serempak semuanya. Dalam hal ini, perlu dilakukan dekomposisi tugas yang bertujuan untuk menentukan bagian proses yang dapat dieksekusi dalam bentuk paralel dan bagian proses yang harus dieksekusi dalam bentuk sekuensial. Beberapa tugas dapat dikerjakan secara paralel (*concurrent*) apabila tidak ada keterkaitan data antar tugas-tugas tersebut.

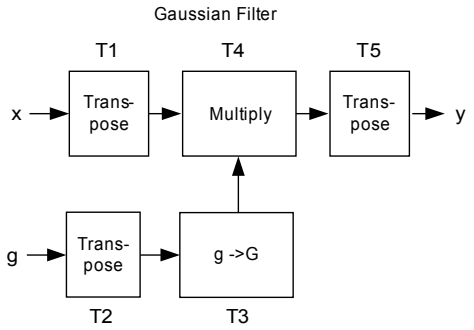
Dekomposisi tugas-tugas SDR dalam penelitian ini berdasarkan blok diagram modulasi GMSK baik dari sisi pemancar maupun penerima.

Peran algoritma sangat menentukan dalam melakukan dekomposisi tugas. Berdasarkan hasil eksperimen yang telah dilakukan, maka dapat dinyatakan bahwa: kesulitan umum dalam menghasilkan tugas-tugas yang dapat diproses secara bersamaan adalah memecah algoritma komputasi yang berbentuk iterasi (*looping*) menjadi beberapa iterasi lain yang tidak memiliki keterkaitan data. Jika sebuah iterasi dapat dipecah menjadi beberapa iterasi lain tanpa keterkaitan data, maka iterasi-iterasi tersebut merupakan satuan-satuan tugas yang dapat diproses secara bersamaan.

Berikut ini diberikan contoh dalam melakukan dekomposisi tugas dari blok fungsi *filter* Gaussian pada modulasi GMSK. Fungsi *filter* Gaussian adalah sebuah proses konvolusi antara vektor baris deretan bit $\boldsymbol{x}$ dengan vektor baris respons *filter* Gaussian $\boldsymbol{g}$. Algoritma konvolusi dapat menggunakan perkalian bentuk matrik (Orfanidis, 1996).

Algoritma 1. Konvolusi:

1. Buat vektor baris $\boldsymbol{x}$ menjadi vektor kolom $\boldsymbol{x}'$ (transpose(x)).
2. Buat vektor baris $\boldsymbol{g}$ menjadi vektor kolom $\boldsymbol{g}'$ (transpose(g)).
3. Buat matrik $\boldsymbol{G}$ dari vektor kolom $\boldsymbol{g}'$.
4. Kalikan matrik $\boldsymbol{G}$ dengan vektor kolom $\boldsymbol{x}$ dan tampung hasilnya dalam vektor kolom $\boldsymbol{y}$.
5. Buat vektor kolom $\boldsymbol{y}$ menjadi vektor baris $\boldsymbol{y}'$.



Gambar 2. Dekomposisi fungsi *filter* Gaussian

Gambar 2 menunjukkan bahwa fungsi *filter* Gaussian memiliki 5 buah komponen tugas, yaitu: (1) tugas T1 melakukan transpose vektor baris $\boldsymbol{x}$ ($1 \times M$ elemen) menjadi vektor kolom $\boldsymbol{x}'$ ($M \times 1$) elemen; (2) tugas T2 melakukan transpose vektor baris $\boldsymbol{g}$ ($1 \times L$) menjadi vektor kolom $\boldsymbol{g}'$ ()$L \times 1$; (3) tugas T3 membentuk matrik $\boldsymbol{G}$ yang memiliki elemen $(L + M - 1) \times M$ dimana nilai tiap-

[1] Karangmalang, Yogyakarta, 55281. Telp. (0274) 586168, eko@uny.ac.id,
[2] Jl. Ganesha 10 Bandung 40132, Tel. (022)-4254034, ekoaji332@students.itb.ac.id

tiap elemen di salin dari vektor kolom $\boldsymbol{g}'$ dengan mengunakan Algoritma 2 dan hasilnya ditunjukkan pada Gambar 3; (4) tugas T4 melakukan proses perkalian matrik $\boldsymbol{G}$ dengan vektor kolom $\boldsymbol{x}'$ sehingga menghasilkan vektor kolom $\boldsymbol{y}$ yang berukuran $(L+M-1)\times 1$; dan (5) tugas T5 melakukan transpose vektor kolom $\boldsymbol{y}$ dengan $(L+M-1)\times 1$ elemen menjadi vektor baris $\boldsymbol{y}'$ yang berukuran $1\times(L+M-1)$.

Berdasarkan Gambar 2, nampak bahwa fungsi T1 dan T2 tidak memiliki keterkaitan data sehingga dapat diproses secara bersamaan. Sedangkan T3 merupakan proses setelah T2; proses T4 merupakan proses setelah T1 dan T3, dan terakhir proses T5 merupakan proses setelah proses T1, T2, T3, dan T4. Dengan demikian proses T3, T4, dan T5 tidak bisa dilakukan secara paralel tetapi harus dilakukan secara sekuensial.

Sedangkan algoritma tugas T3 dapat dijelaskan sebagai berikut:

Algoritma 2. Menyalin matrik $\boldsymbol{g}'$ ke dalam matrik $\boldsymbol{G}$:

1. Inisialisasi ukuran matrik $\boldsymbol{G}$ adalah $(L+M-1)\times M$.
2. *Loop*: untuk *i* = 0 sampai dengan M – 1 lakukan langkah 3.
3. Salin vektor kolom $\boldsymbol{g}'$ kedalam matrik ke $\boldsymbol{G}$ dengan posisi baris ke-*i* dan kolom ke-*i.*

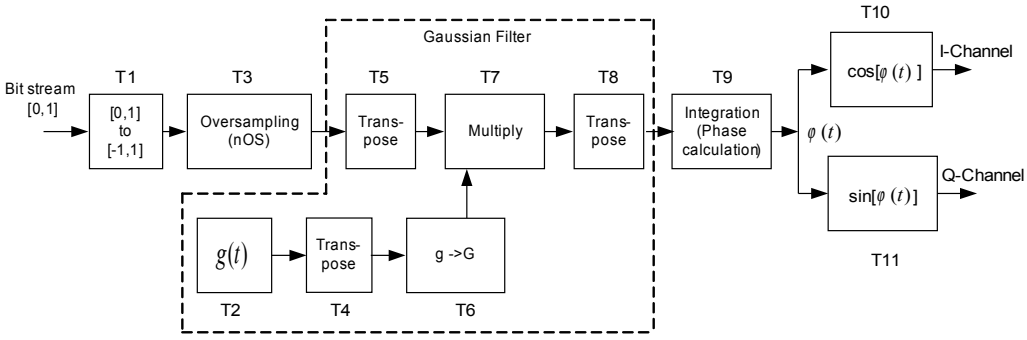
Gambar 3 menunjukkan perkalian matrik berdasarkan Algoritma 1 dan Algoritma 2 tersebut di atas.

Hasil dekomposisi tugas secara lengkap pada bagian modulator GMSK ditunjukkan pada Gambar 4.

$$\boldsymbol{y} = \boldsymbol{G}\times\boldsymbol{x}'$$

$$= \begin{bmatrix} g_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g_1 & g_0 & 0 & 0 & 0 & 0 & 0 \\ g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 \\ 0 & g_2 & g_1 & g_0 & 0 & 0 & 0 \\ 0 & 0 & g_2 & g_1 & g_0 & 0 & 0 \\ 0 & 0 & 0 & g_2 & g_1 & g_0 & 0 \\ 0 & 0 & 0 & 0 & g_2 & g_1 & g_0 \\ 0 & 0 & 0 & 0 & 0 & g_2 & g_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & g_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix}$$

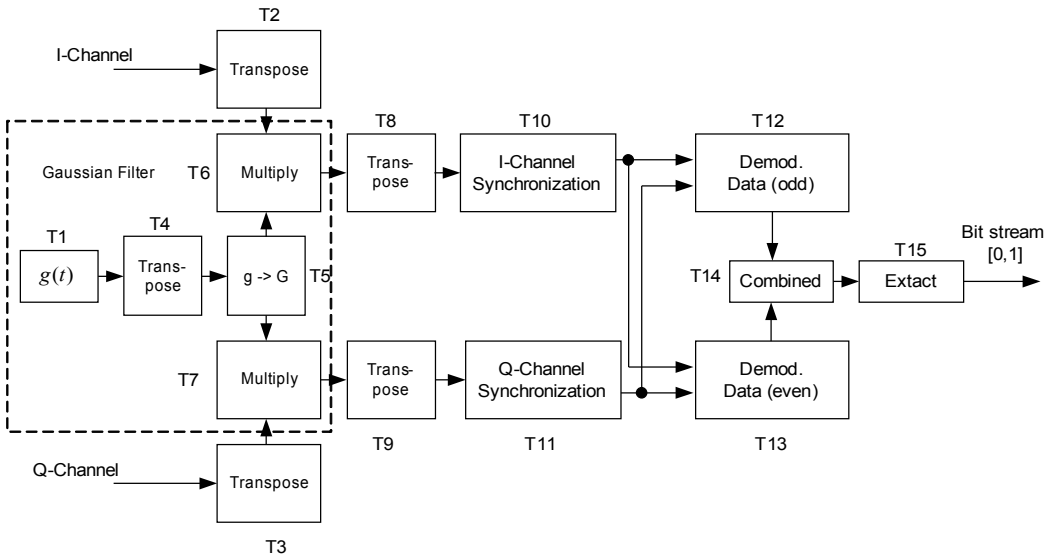
Gambar 3. Perkalian matrik $\boldsymbol{G}$ dengan vektor kolom $\boldsymbol{x}'$



Gambar 4. Dekomposisi tugas pada modulator GMSK

[1] Karangmalang, Yogyakarta, 55281. Telp. (0274) 586168, eko@uny.ac.id,
[2] Jl. Ganesha 10 Bandung 40132, Tel. (022)-4254034, ekoaji332@students.itb.ac.id

Berdasarkan hasil dekomposisi tugas tersebut, maka modulator GMSK memiliki 11 (sebelas tugas), yaitu: (1) tugas T1 mengubah deretan bit non polar menjadi deretan bit bipolar; (2) tugas T2 menghasilkan *impulse response filter* Gaussian; (3) tugas T3 melakukan *oversampling*; (4) tugas T4 melakukan transpose vektor baris g; (5) tugas T5 melakukan transpose vektor baris x; (6) tugas T6 membuat matrik G; (7) tugas T7 melakukan perkalian matrik G dengan vektor kolom x' sehingga menghasilkan vektor kolom y; (8) tugas T8 melakukan transpose vektor kolom y; (9) tugas T9 melakukan integrasi untuk menghitung sudut fase θ dari vektor baris y'; (10) tugas T10 menghitung nilai cosinus dari sudut fase sehingga menghasilkan vektor baris I untuk sinyal I-*Channel*; dan (11) tugas T11 menghitung nilai sinus dari sudut fase sehingga menghasilkan vektor baris Q untuk sinyal Q-*Channel*.



Gambar 5. Dekomposisi tugas pada demodulator GMSK

Gambar 5 menununjukkan hasil dekomposisi tugas secara lengkap bagian demodulator GMSK. Berdasarkan hasil dekomposisi tugas tersebut, maka demodulator GMSK memiliki 15 (lima belas) tugas, yaitu: (1) tugas T1 menghasilkan *impulse response filter* Gaussian; (2) tugas T2 melakukan transpose vektor baris I; (3) tugas T3 melakukan transpose vektor baris Q; (4) tugas T4 melakukan transpose vektor baris g; (5) tugas T5 membuat matrik G; (6) tugas T6 melakukan perkalian matrik G dengan vektor kolom I' sehingga menghasilkan vektor kolom i; (7) tugas T7 perkalian matrik G dengan vektor kolom Q' sehingga menghasilkan vektor kolom q; (8) tugas T8 melakukan transpose vektor kolom i; (9) tugas T9 melakukan transpose vektor kolom q; (10) tugas T10 melakukan sinkronisasi vektor baris i'; (11) tugas T11 melakukan sinkronisasi vektor baris q'; (12) tugas T12 melakukan demodulasi data ganjil dari vektor kolom i' dan q'; (13) tugas T13 melakukan demodulasi data genap dari vektor kolom i' dan q'; (14) tugas T14 melakukan penggabungan data ganjil dan genap; dan (15) tugas T15 melakukan ekstraksi bit menjadi deretan bit non polar.

[1] Karangmalang, Yogyakarta, 55281. Telp. (0274) 586168, eko@uny.ac.id,
[2] Jl. Ganesha 10 Bandung 40132, Tel. (022)-4254034, ekoaji332@students.itb.ac.id

Setiap *node* dalam grafik tugas memiliki dua parameter yang sangat diperlukan dalam proses penjadwalan, yaitu nilai ketinggian dan waktu eksekusi. Nilai ketinggian dapat dihitung setelah susunan *node* dalam grafik tugas bisa terbentuk berdasarkan hubungan antar *node* (*presedence relations*), sedangkan waktu eksekusi diperoleh dengan mengeksekusi fungsi tersebut. Penelitian ini menggunakan bahasa pemrograman JAVA, sehingga waktu eksekusi diperoleh dengan cara mengimplementasikan algoritma dalam bentuk *method-method* dari kelas yang didefinisikan untuk modulator dan demodulator GMSK, dan menjalankan atau mengeksekusi setiap *method* tersebut.

Spesifikasi komputer yang digunakan dalam menghitung waktu eksekusi masing-masing tugas untuk modulator dan demodulator GMSK ditunjukkan pada Tabel 1.

Tabel 1. Spesifikasi Komputer

No	Komponen	Spesifikasi
1.	Processor	Intel Pentium 1.73 GHz
2.	RAM	DDR2 794 MHz, 1.272 MByte
3.	Sistem Operasi	Window XP Home Edition
4.	Bahasa Pemrograman	Java

Waktu eksekusi absolut setiap tugas pada modulator GMSK disajikan pada Tabel 2 dalam satuan nano detik (10^{-9} detik). Pengujian waktu eksekusi dilakukan untuk beberapa ukuran data, yaitu 8 bit, 32 bit, 64 bit, 128 bit, 256 bit, 512 bit, dan 1024 bit.

Berdasarkan hasil eksperimen untuk memperoleh waktu eksekusi tiap-tiap tugas modulator GMSK, nampak bahwa waktu eksekusi dari T3 (menghasilkan respons *filter* Gaussian) untuk data 8 bit memiliki waktu eksekusi paling tinggi. Hal ini disebabkan fungsi tersebut melibatkan *method* yang berfungsi untuk menghitung nilai erf (*error function*) seperti yang ditunjukkan pada persamaan 1. Dua fungsi lain yang juga sangat besar waktu eksekusinya adalah T6 dan T7.

Grafik hubungan antara variasi ukuran data dalam bit terhadap waktu eksekusi ditunjukkan pada Gambar 6 dan Gambar 7. Gambar 6 menunjukkan grafik hubungan antara variasi ukuran data dalam satuan bit terhadap waktu eksekusi masing-masing tugas komputasi pada modulator GMSK dari Gambar 4. Waktu eksekusi yang ditampilkan dalam gambar ini adalah: (1) tugas T1 (2) tugas T3; (3) tugas T4; (4) tugas T5; (5) tugas T8; (6) tugas T9; (7) tugas T10; dan (8) Tugas T11.

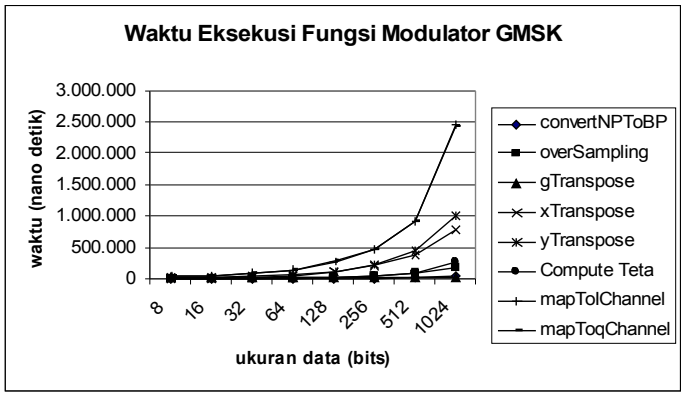
Tabel 2. Waktu Eksekusi Absolut Tugas-Tugas Modulator GMSK (dalam nano detik)

Tugas	Fungsi	Jumlah Data (bit)							
		8	16	32	64	128	256	512	1024
T1	convertNPToBP	407	713	1.338	2.603	5.153	10.594	20.594	40.470
T2	overSampling	1.397	2.306	4.138	9.231	19.547	40.344	94.470	188.530
T3	respFiltGauss	4.718.000	4.688.000	4.688.000	4.720.000	4.720.000	4.688.000	4.684.000	4.658.000
T4	gTranspose	18.600	15.800	15.200	18.600	15.800	15.600	15.600	16.000
T5	xTranspose	6.260	6.240	25.000	49.700	102.180	210.620	390.640	783.120
T6	copyMatrik	374.000	374.000	1.594.000	3.376.000	7.280.000	16.064.000	33.750.000	79.560.000
T7	kalikanMatrix	160.000	220.000	814.000	2.468.000	7.782.000	27.032.000	101.844.000	436.686.000
T8	yTranspose	28.000	21.800	40.600	65.600	115.400	219.000	437.400	1.018.400
T9	Compute Teta	4.688	5.658	7.940	12.468	20.624	45.340	95.620	270.920
T10	mapTolChannel	38.120	51.520	79.040	134.700	267.800	478.100	926.920	2.463.120
T11	mapToqChannel	39.360	52.840	79.060	134.680	286.600	474.700	920.620	2.427.520

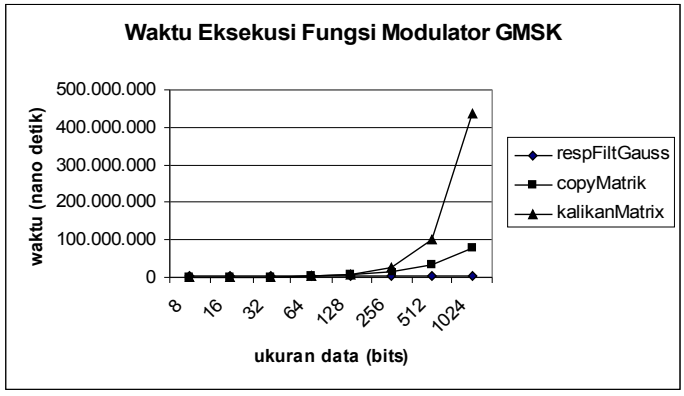
[1] Karangmalang, Yogyakarta, 55281. Telp. (0274) 586168, eko@uny.ac.id,
[2] Jl. Ganesha 10 Bandung 40132, Tel. (022)-4254034, ekoaji332@students.itb.ac.id

Berdasarkan gambar tersebut maka pertambahan jumlah data secara eksponensial juga akan mengakibatkan peningkatan secara eksponensial terhadap waktu eksekusi dari tugas-tugas T1, T3, T5, T8, T9, T10, dan T11. Dengan kata lain, peningkatan waktu eksekusi sebanding dengan peningkatan jumlah data yang akan diproses. Sedangkan waktu eksekusi dari tugas T4 relatif tetap dalam arti waktu eksekusi tidak dipengaruhi oleh ukuran data yang akan diproses oleh modulator GMSK.

Waktu eksekusi paling tinggi adalah tugas T10 dan T11, yaitu untuk data 1024 bit waktu eksekusi T10 adalah 2.463.120 nano detik dan T11 adalah 2.427.520 nano detik. Kedua tugas ini memiliki waktu eksekusi relatif sama karena tugas yang dikerjakan adalah sama, yaitu T10 menghitung nilai cosinus dan T11 menghitung nilai sinus. Urutan waktu eksekusi dari yang paling besar ke waktu ekekusi yang paling kecil berikutnya dari tugas-tugas tersebut setelah T10 dan T11 untuk ukuran data 1024 adalah T8 dengan waktu eksekusi 1.018.400 nano detik, T5 dengan waktu eksekusi 783.120 nano detik, T9 dengan waktu eksekusi 270.920 nano detik, T2 dengan waktu eksekusi 188.530 nano detik.



Gambar 6. Grafik hubungan antara ukuran data (bit) terhadap waktu eksekusi



Gambar 7 Grafik hubungan antara ukuran data (bit) terhadap waktu eksekusi

Gambar 7 menunjukkan grafik hubungan antara variasi ukuran data dalam satuan bit terhadap waktu eksekusi masing-masing tugas komputasi pada modulator GMSK dari Gambar 4 selain tugas-tugas yang telah ditampilkan pada

[1] Karangmalang, Yogyakarta, 55281. Telp. (0274) 586168, eko@uny.ac.id,
[2] Jl. Ganesha 10 Bandung 40132, Tel. (022)-4254034, ekoaji332@students.itb.ac.id

Gambar 6. Tugas-tugas yang ditunjukkan pada grafik ini adalah: (1) tugas T3; (2) tugas T6; dan (3) tugas T7.

Berdasarkan gambar tersebut maka pertambahan jumlah data secara eksponensial juga akan mengakibatkan peningkatan secara eksponensial terhadap waktu eksekusi dari tugas-tugas T6 dan T7. Dengan kata lain, peningkatan waktu eksekusi sebanding dengan peningkatan jumlah data yang akan diproses. Sedangkan waktu eksekusi dari tugas T3 relatif tetap dalam arti waktu eksekusi tidak dipengaruhi oleh ukuran data yang akan diproses oleh modulator GMSK.

Waktu eksekusi paling besar dari kelompok tugas tersebut di atas adalah tugas T7, yaitu untuk data 1024 bit waktu eksekusi T7 adalah sebesar 436.686.000 nano detik. Urutan berikutnya tugas T6 dengan waktu eksekusi untuk data 1024 bit adalah 79.560.000 nano detik. Sedangkan tugas T3 waktu eksekusi tidak terpengaruh oleh besarnya ukuran data yang akan diproses oleh modulator GMSK yaitu sebesar 4.658.000 nano detik.

Tabel 3 menunjukkan waktu eksekusi absolut dari tugas demodulator GMSK dalam satuan nano detik (10^{-9} detik). Pengujian waktu eksekusi dilakukan untuk beberapa ukuran data, yaitu 8 bit, 32 bit, 64 bit, 128 bit, 256 bit, 512 bit, dan 1024 bit.

Pada bagian demodulator, nampak bahwa waktu eksekusi T1 (menghasilkan respons *filter* Gaussian) memiliki sifat yang sama seperti pada modulator. Sedangkan untuk data 1024 bit, waktu eksekusi yang tinggi adalah T5, T6 dan T7. Untuk demodulator, tugas mengalikan matrik ada dua, yaitu untuk sinyal I-*Channel* dan sinyal Q-*Channel.*

Grafik hubungan antara variasi ukuran data dalam bit terhadap waktu eksekusi ditunjukkan pada Gambar 8 dan Gambar 9. Gambar 8 menunjukkan grafik hubungan antara variasi ukuran data dalam satuan bit terhadap waktu eksekusi masing-masing tugas komputasi pada demodulator GMSK seperti yang ditunjukkan pada Gambar 5. Waktu eksekusi yang digambarkan pada grafik ini adalah 4 (empat) tugas dari demodulator GMSK yang memiliki waktu eksekusi terbesar. Tugas-tugas tersebut adalah: (1) tugas T1; (2) tugas T5; (3) tugas T6; dan (4) tugas T7.

Tabel 4. Waktu Eksekusi Absolut Tugas-Tugas Demodulator GMSK (nano detik)

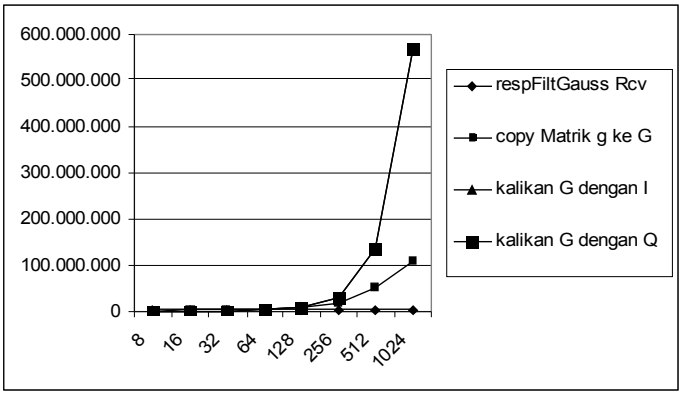
Tugas	Fungsi	Jumlah Data (bit)							
		8	16	32	64	128	256	512	1024
T1	respFiltGauss Rcv	3.502.000	3.280.000	3.250.000	3.030.000	3.032.000	2.934.000	2.970.000	2.906.000
T2	gTranspose	19.000	21.800	21.800	24.800	25.000	21.800	31.200	22.000
T3	iCh Transpose	18.600	37.600	37.600	62.600	118.600	218.800	565.400	1.062.400
T4	qCh Transpose	28.000	37.600	37.600	65.400	115.800	218.800	518.600	1.018.600
T5	copy Matrik g ke G	1.440.000	1.906.000	1.906.000	4.536.000	8.372.000	17.218.000	52.218.000	107.718.000
T6	kalikan G dengan I	656.000	938.000	938.000	3.592.000	9.876.000	31.188.000	137.440.000	568.500.000
T7	kalikan G dengan Q	656.000	938.000	938.000	3.592.000	9.876.000	31.188.000	137.440.000	568.500.000
T8	I Transpose	47.000	47.000	47.000	84.200	128.200	234.400	643.400	1.106.000
T9	Q Transpose	40.400	43.800	43.800	91.000	128.000	234.600	575.000	1.100.200
T10	Sinkronisasi I-Channel	1.540	1.560	1.900	1.900	2.500	5.940	17.500	33.780
T11	Sinkronisasi Q-Channel	1.580	1.580	2.480	2.800	3.420	6.240	12.500	29.680
T12	Demodulasi Data Odd	1.560	3.740	5.320	10.900	20.660	41.260	82.820	167.500
T13	Demodulasi Data Even	2.160	3.120	5.920	10.980	21.540	42.500	82.840	165.300

[1] Karangmalang, Yogyakarta, 55281. Telp. (0274) 586168, eko@uny.ac.id,
[2] Jl. Ganesha 10 Bandung 40132, Tel. (022)-4254034, ekoaji332@students.itb.ac.id

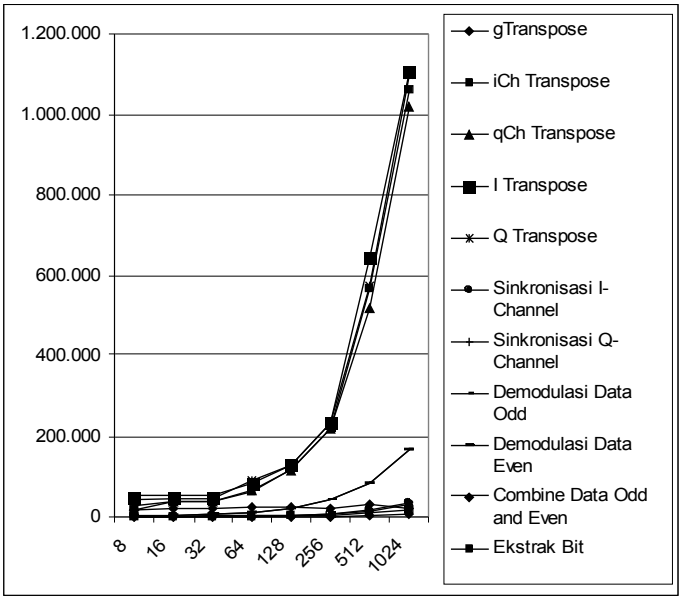
T14	Combine Data Odd and Even	57	100	188	356	703	1.368	2.700	5.350
T15	Ekstrak Bit	522	747	859	1.435	2.175	4.685	9.050	17.597

Berdasarkan Gambar 8 tersebut maka pertambahan jumlah data secara eksponensial juga akan mengakibatkan peningkatan waktu eksekusi secara eksponensial untuk tugas-tugas T5, T6 dan T7. Dengan kata lain, peningkatan waktu eksekusi sebanding dengan peningkatan jumlah data yang akan diproses. Waktu eksekusi tugas T1 relatif tetap dalam arti waktu eksekusi tidak dipengaruhi oleh ukuran data yang akan diproses oleh demodulator GMSK.

Waktu eksekusi paling besar adalah tugas T6 dan T7, yaitu untuk data 1024 bit waktu eksekusi T6 adalah sebesar 568.500.000 nano detik dan T7 adalah 568.500.000 nano detik. Urutan berikutnya adalah tugas T5 yaitu 107.718.000 nano detik, dan terakhir adalah T1 dengan waktu eksekusi 2.906.000 nano detik. Namun demikian, waktu eksekusi tugas T1 tidak terpengaruh oleh besarnya ukuran data yang akan diproses oleh demodulator GMSK.



Gambar 8. Grafik hubungan antara ukuran data (bit) terhadap waktu eksekusi



Gambar 9. Grafik hubungan antara ukuran data (bit) terhadap waktu eksekusi

[1] Karangmalang, Yogyakarta, 55281. Telp. (0274) 586168, eko@uny.ac.id,
[2] Jl. Ganesha 10 Bandung 40132, Tel. (022)-4254034, ekoaji332@students.itb.ac.id

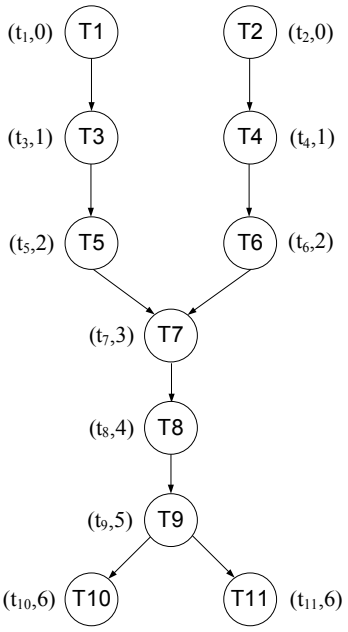
Gambar 9 menunjukkan grafik hubungan antara variasi ukuran data dalam satuan bit terhadap waktu eksekusi masing-masing tugas komputasi pada demodulator GMSK seperti yang ditunjukkan pada Gambar 5, selain yang telah ditampilkan pada Gambar 8. Waktu eksekusi yang digambarkan pada grafik ini adalah 11 (sebelas) tugas demodulator GMSK selain yang telah ditampilan pada Gam-bar 8. Tugas-tugas tersebut adalah: (1) tugas T2; (2) tugas T3; (3) tugas T4; (4) tugas T8; (5) tugas T9; (6) tugas T10; (7) tugas T11; (8) tugas T12; (9) tugas T13; (10) tugas T14; dan (11) tugas T15.

Berdasarkan Gambar 9 tersebut maka pertambahan jumlah data secara eksponensial juga akan meningkatkan waktu eksekusi tugas-tugas T3, T4, T8, T9, T12, T13 secara eksponensial. Dengan kata lain, peningkatan waktu eksekusi sebanding dengan peningkatan jumlah data yang akan diproses. Enam tugas ini memiliki waktu eksekusi yang terbesar dibanding tugas-tugas lain yang digambarkan dalam grafik ini. Sedangkan waktu eksekusi dari tugas T10, T11, T14, T15 memiliki waktu eksekusi yang lebih rendah serta memiliki sifat akan berubah secara eksponensil jika ukuran data berubah secara eksponensial. Khusus untuk tugas T2 relatif tetap dalam arti waktu eksekusi tidak dipengaruhi oleh ukuran data yang akan diproses oleh demodulator GMSK.

Waktu eksekusi paling besar dari kelompok tugas tersebut adalah tugas T1, yaitu 2.906.000 nano detik. Empat tugas berikutnya yang memiliki waktu eksekusi terbesar di bawah waktu eksekusi T1 adalah waktu eksekusi tugas T3, T4, T8, dan T9. Masing-masing waktu eksekusi T3, T4, T8, dan T9 untuk ukuran data 1024 bit secara berturut-turut adalah 1.062.400 nano detik, 1.018.600 nano detik, 1.106.000 nano detik, dan 1.100.200 nano detik. Sedangkan waktu eksekusi T12 dan T13 merupakan kelompok berikutnya yaitu untuk ukuran data 1024 bit memiliki waktu eksekusi 167.300 nano detik dan 165.300 nano detik. Terakhir, tugas yang memiliki waktu eksekusi paling rendah adalah tugas T10, T11, T14, dan T15 yaitu untuk ukuran data 1024 bit secara berturut-turut memiliki waktu eksekusi sebesar 33.780 nano detik, 29.650 nano detik, 5.350 nano detik, dan 17.597 nano detik.

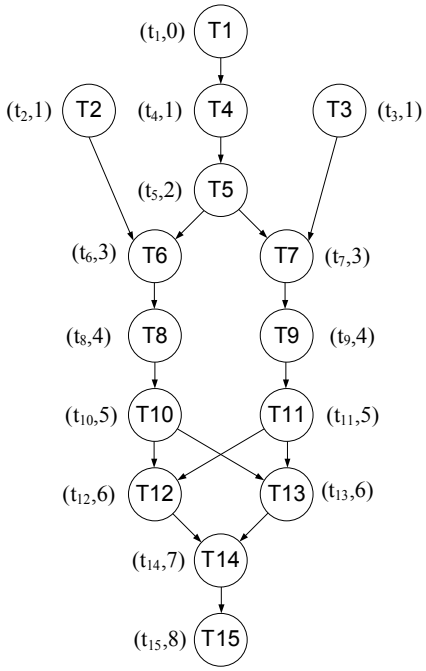
Berdasarkan uraian tentang dekomposisi tugas serta pengujian untuk memperoleh waktu eksekusi, maka grafik tugas modulator GMSK ditunjukkan pada Gambar 10 dan grafik tugas demodulator GMSK ditunjukkan pada Gambar 1 .

Grafik tugas modulator GMSK tersebut terdiri dari 11 (sebelas) *node* atau tugas seperti ditunjukkan pada Gambar 10. Masing-masing nilai ketinggian untuk tiap-tiap *node* tugas sudah dicantumkan, sedangkan parameter waktu eksekusi untuk menyatakan bobot komputasi dapat diperoleh dari Tabel 2 berdasarkan ukuran data yang diinginkan.



Gambar 10. Grafik tugas modulator GMSK

[1] Karangmalang, Yogyakarta, 55281. Telp. (0274) 586168, eko@uny.ac.id,
[2] Jl. Ganesha 10 Bandung 40132, Tel. (022)-4254034, ekoaji332@students.itb.ac.id



Gambar 11. Grafik tugas demodulator GMSK

Sedangkan Grafik tugas untuk demodulator GMSK ditunjukkan pada Gambar 11, yaitu terdiri dari 15 (lima belas) *node* atau tugas, dan parameter tentang nilai ketinggian masing-masing *node* tugas telah dicantumkan sedangkan pa-rameter bobot komputasi dapat diperoleh dari Tabel 3 untuk waktu eksekusi ab-solut berdasarkan ukuran data yang diinginkan.

KESIMPULAN

Penelitian ini telah menghasilkan grafik tugas untuk modulator dan demodulator GMSK untuk keperluan penjadwalan komputasi terdistribusi dengan metoda paralelisme tugas, dimana modulator terdiri dari 11 (sebelas) tugas dan demodulator terdiri dari 15 (lima belas) tugas.

Waktu eksekusi paling besar adalah fungsi menghitung respons *filter* Gaussian, yaitu T3 pada modulator dan T1 pada demodulator. Waktu eksekusi yang tergolong besar berikutnya adalah fungsi menyalin matrik (T6 pada modulator dan T5 pada demodulator), dan fungsi mengalikan matrik (T7 pada modulator, T6 dan T7 pada modulator.

DAFTAR PUSTAKA

Abdeyazdan, M., dan Rahmani, A. M. (2008) : *Multiprocessor Task Scheduling Using a New Prioritizing Genetic Algorithm based on number of Task Children*, Distributed and Parallel Systems, Springer US, pp. 105 – 114. [On-line], Available at http://www. Springerlink.com/ content/ v57t546 13657845l/fulltext.pdf.

Haykin, S. (2004) : *Communication System*, 4th ed. New York, John Wiley & Sons, Inc, pp. 396 – 402.

Hou, E. S. H., Ansari, N., dan Ren, H. (1994) : A Genetic Algorithm for Multiprocessor Scheduling, *Journal of IEEE Trans. on Parallel and Distributed Systems*, **Vol. 5, No. 2**, pp. 113 –120.

Lee, Y. H., dan Chen, C. (2003) : A Modified Genetic Algorithm For Task Scheduling In Multiprocessor Systems, [On-line], Available at http://parallel.iis. sinica.edu.tw/cthpc2003/papers/CTHPC2003-18.pdf.

Marpanaji, E., dkk. (2007) : Pengukuran Unjuk Kerja Modulasi GMSK pada Software-Defined Radio Platform, *Jurnal Telkomnika*, **Vol. 5, No. 2 Agustus 2007**, 73 – 84.

Orfanidis, S. J. (1996) : *Intoduction to Signal Processing*, New Jersey, USA, Prentice Hall, 132 – 134.

Rahmani, A. M., dan Vahedi, M. A. (2008) : A Novel Task Scheduling In Multiprocessor Systems With Genetic Algorithm By Using Elitism Stepping Method, *Journal of Computer Science*, **Vol. 7 No. 2 June 2008**. [On-line]. Available at http:// www.dcc.ufla.br/infocomp/artigos/ v7.2/art08.pdf.

Tsujimura, Y., dan Gen M. (1997) : *Genetic Algorithms for Solving Mul-tiprocessor Scheduling*

[1] Karangmalang, Yogyakarta, 55281. Telp. (0274) 586168, eko@uny.ac.id,
[2] Jl. Ganesha 10 Bandung 40132, Tel. (022)-4254034, ekoaji332@students.itb.ac.id

Problems, Simulated Evolution and learning, Heidelberg, Springer Berlin, pp. 106 – 115. [On-line], Available at http:// www.springerlink. com/content/w461447527q15086/fulltext.pdf.

Zheng, K., Li G., dan Huang L. (2007) : A weighted Scheduling Scheme in an Open Software Radio Environment, Communication, Computer and Signal Processing, PacRim 2007, *Proceedings of IEEE Pacific Rim Conference, Canada, August, 2007*, pp. 561 – 564.

[1] Karangmalang, Yogyakarta, 55281. Telp. (0274) 586168, eko@uny.ac.id,
[2] Jl. Ganesha 10 Bandung 40132, Tel. (022)-4254034, ekoaji332@students.itb.ac.id