APLIKASI JARINGAN SYARAF TIRUAN BERBASIS TRANSPUTER PADA SIMULASI SISTEM DRIVE-RIG

TUGAS AKHIR

Oleh :

A Z H A R

2922202000

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

APLIKASI JARINGAN SYARAF TIRUAN BERBASIS TRANSPUTER PADA SIMULASI SISTEM DRIVE-RIG

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik Elektro
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
S u r a b a y a

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
(Ir. ISKANDAR ZULKARNAIN)	(Ir. P U J I O N O)

SURABAYA

MARET 1996

ABSTRAK

Transputer merupakan mikroprosesor yang telah berisi prosesor, memory lokal dan serial link kecepatan tinggi untuk komunikasi antar transputer pada sistem komputasi paralel. Untuk mengeksploitasi aplikasi transputer pada sistem-sistem konkuren dapat didesain modul-modul proses secara konkuren melalui link.

Salah satu kendala yang mendasar dalam merancang sistem jaringan syaraf tiruan (JST) adalah sifat sekuensial dari komputer konvensional, yang bertentangan dengan sifat bawaan yang dimiliki oleh JST, yaitu pemrosesan secara paralel. Dalam tugas akhir ini dibahas *parallel processing* JST dengan aplikasi pada plant dinamis sebagai kontroler sistem *electric drive-rig*. Dengan mendistribusikan beberapa proses secara konkuren diharapkan akan menambah unjuk kerja sistem dalam tracking plant dinamis.

KATA PENGANTAR

Puji syukur kami panjatkan kehadirat Allah Yang Maha Esa yang telah melimpah Taufik dan Hidayah-Nya sehingga dapat menyusun dan menyelesaikan tugas akhir ini.

Tugas akhir ini disusun untuk memenuhi salah satu syarat untuk memperoleh gelar Sarjana Teknik Elektro pada bidang studi Elektronika, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya. Sesuai dengan judulnya yaitu "Aplikasi Jaringan Syaraf Tiruan Berbasis Transputer Pada Simulasi Sistem Drive-Rig," dalam tugas akhir ini akan dibahas implementasi JST pada transputer dengan proses-proses konkuren dengan aplikasi pada simulasi sistem drive-rig.

Pada kesempatan ini saya menyampaikan terimakasih untuk semua pihak yang membantu baik secara moril maupun materil selama penelitian sampai sesesainya buku tugas akrir, khususnya kepada :

1. Bapak Ir. Iskandar Zulkarnain, yang tanpa lelah membimbing, mengarahkan dan membantu penelitian tugas akhir ini.
2. Bapak Ir.Soetikno, kordinator Bidang Studi Elektronika, yang telah memberi izin penggunaan fasilitas selama penelitian sampai penyusunan buku laporan tugas akhir.

3. Bapak Ir. Pujiono, pembimbing dan dosen wali, yang selalu memberi bimbingan dan pengarahan baik selama kuliah dan dalam penelitian tugas akhir.
4. Ayahanda, Ibunda, adik Faisal dan adik Aty yang banyak memberi bantuan moril dan materil yang tidak ternilai, dorongan dan doa sehingga penelitian ini dapat selesai.
5. Adek (Hendrawaty), yang banyak memberi dorongan, doa dan semangat, terimakasih (*lsgb*).
6. Seluruh staf dosen bidang studi Elektronika yang banyak memberi ilmu selama kuliah di Institut Teknologi Sepuluh Nopember.
7. Rekan-rekan di Lab. Riset dan Pengembangan (B-203) dan Lab. Elektronika Pemrosesan Sinyal (B-205), terimakasih banyak atas solidaritas dan bantuan yang diberikan.

Tugas akhir ini masih jauh dari sempurna dan banyak kekurangan. Maka kritik dan saran sangat diharapkan guna penyempurnaan. Meskipun demikian semoga tugas akhir ini ada manfaatnya.

Surabaya, Februari 1996

Azhar

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Komputer dan program konvensional berjalan secara sekuensial, artinya sebuah instruksi harus deselesaikan terlebih dahulu sebelum instruksi yang lain dikerjakan. Bagian dari permasalahan dikerjakan secara serial, bahkan untuk bagian-bagian yang sebenarnya terjadi secara bersamaan. Pada saat program sekuensial digunakan untuk menggambarkan fenomena nyata, pengaruh interaksi waktu-nyata (*real-time*) hilang, dan hanya model yang disederhanakan yang dimungkinkan.

Pemrosesan dan pemrograman paralel masih dalam taraf awal perkembangan. Belum banyak *body of knowledge* yang dapat ditarik untuk langsung diterapkan. Apalagi untuk komputasi kelas PC. Tetapi dilain pihak, penelitian yang berhubungan dengan jaringan syaraf tiruan (JST) banyak dilakukan pada sistem komputasi kelas PC, sehingga bukan sesuatu yang mengherankan bila ditemukan kenyataan bahwa seseorang melatih (melakukan iterasi sampai error yang diperoleh memenuhi kriteria) JST-nya membutuhkan waktu yang cukup lama.

Berdasarkan hal inilah, dengan berbagai keterbatasan yang ada, akan mengimplementasikan JST secara parallel processing dengan transputer. Diharapkan, dengan mendistribusikan proses komputasi pada transputer semirip

mungkin dengan proses sebenarnya pada JST (dan tentunya dengan mempertimbangkan faktor efisiensi dan kemudahan implementasi). Aplikasi parallel processing dengan transputer untuk kontroler jaringan syaraf tiruan akan digunakan sebagai unit pengontrol sistem drive-rig.

Sistem drive-rig adalah sistem pemindahan material yang digerakkan oleh motor-motor listrik pada proses-proses di Industri. Penggunaan pengaturan yang konvensional mempunyai reaksi lambat dan tidak adaptif terhadap perubahan beban yang non linier dan plant yang bersifat dinamis. Banyak tuntutan dalam pengaturan drivr-rig secara real-time dan multivariable dalam mengatur kecepatan dan ketegangan belt dalam pemindahan material diperlukan suatu kontrol yang cepat.

1.2 Permasalahan

Permasalahan yang dihadapi dalam tugas akhir ini adalah bagaimana caranya untuk mengaplikasikan jaringan syaraf tiruan pada pengaturan multivariable, yang dapat memberikan respon real-time pada pengaturan drive-rig, terutama untuk mengatasi kemampuan suatu kontroller yang mempunyai respon yang cepat terhadap dead time suatu plant yang dinamis.

Solusi paralel ideal yang mungkin terbayang pertama sekali adalah sejumlah besar susunan prosesor yang terhubung lengkap, dan setiap prosesor tersebut mengemulasikan sebuah sel tunggal. Penyelesaian ini tidak praktis dan juga tidak perlu. Mempunyai sebuah mekanisme yang dapat dikoneksikan dengan mudah dan dapat mengemulsikan sejumlah sel, akan dapat melahirkan pembangun-sistem yang

memiliki nilai-rekayasa. Disini akan dikemukakan alasan bahwa transputer INMOS menyediakan blok pembangun (*building block*) untuk membangun mekanisme seperti itu.

1.3 Tujuan

Tujuan dari pembuatan tugas akhir ini diantaranya :

- ☐ Mempelajari Jaringan Syaraf Tiruan untuk aplikasi dalam sistem pengaturan multivariable.
- ☐ Mempelajari Sistem Transputer untuk implementasi paralel processing Sistem pengaturan berbasis Jaringan Syaraf Tiruan
- ☐ Merencanakan dan membuat kontroller jaringan syaraf tiruan dengan sistem parallel processing dengan transputer yang diaplikasikan padasistem drive-rig
- ☐ Melakukan pengujian sistem pada prototype yang dibuat

1.4 Metodologi

Dalam menyelesaikan dan penelitian tugas akhir ini perlu dilakukan beberapa langkah yaitu :

- ☐ Studi perpustakaan dan studi lapangan
- ☐ Perencanaan dan pembuatan sistem pengaturan untuk sistem pengaturan drive-rig secara hardware dan software.
- ☐ Melakukan pengujian sistem dan analisa kinerja

1.5 Sistematika

Buku tugas akhir ini dibagi menjadi enam bab dengan urutan pendahunuan, jaringan syaraf tiruan, sistem transputer, perancangan sistem pengaturan drive-rig, pengujian sistem dan analisa, serta kesimpulan dan saran.

Bab I adalah pendahuluan yang berisikan mengenai latar belakang, permasalahan, tujuan, sistematika, dan relevansi yang diharapkan dari tugas akhir ini.

Teori mengenai konsep jaringan syaraf tiruan, aplikasi jaringan syaraf tiruan dalam sistem pengaturan, dan algoritma pokok implementasi dari rumusan tersebut dibahas dalam bab II.

Bab III yang membahas transputer ini akan berisi sejarah singkat, gambaran umun, arsitektur internal, dan kemampuan dari transputer.

Bab IV Aplikasi jaringan syaraf tiruan secara paralel processing dengan menggunakan transputer untuk pengaturan sistem elektric drive-rig merupakan inti dari tugas akhir ini akan membahas tujuan perancangan, gambaran sistem drive-rig, implementasi pemrosesan paralel dengan Transputer, hardware pendukung untuk I/O data, dan implementasi JST pada transputer.

Pengujian sistem dan analisa akan dibahas dalam bab V, yang berisikan pengujian sistem secara modular, pengujian sistem secara keseluruhan, analisa kinerja, menampilkan grafik grafik hasil tanggapan plant saat belajar dan saat melakukan pengendalian.

Bab VI berisi kesimpulan penelitian dan tindak lanjut untuk pengembangan sistem dimasa yang akan datang.

1.6 Relevansi

Diharapkan tugas akhir ini dapat menjadi masukan dan referensi awal dalam penelitian parallel processing dan jaringan syaraf tiruan pada aplikasi sistem pengaturan multivariable dimasa yang akan datang, terutama di Laboratorium Riset dan Pengembangan Teknik Elektro Institut Teknologi Sepuluh Nopember Surabaya.

BAB II

JARINGAN SYARAF TIRUAN

Jaringan syaraf tiruan (JST, *Artificial Neural Network*), didefinisikan sebagai jaringan terhubung paralel yang biasanya terdiri dari elemen-elemen dan organisasi hirarkinya, yang diharapkan dapat berintegrasi dengan benda atau keadaan nyata sama seperti sitem syaraf biologis melakukannya. Jaringan syaraf tiruan juga dipandang sebagai suatu sistem yang terdiri dari elemen-elemen pengolah terdistribusi secara paralel dengan kemampuan untuk memperbaiki kinerjanya melalui proses belajar.

Telah diketahui bahwa waktu pemrosesan sistem *neurobiologis* adalah tujuh kali lebih lambat dari pada komputer elektronik; waktu tanggap sel-sel biologis berorde tipikal beberapa milidetik, tetapi sistem neurobiologis dapat melakukan pengenalan pola-pola kompleks yang terdiri dari sejumlah besar elemen hanya dalam beberapa ribu milidetik[1]. Aristektur sel biologis berupa interkoneksi paralel-masif menerangkan hal di atas, dan dari karakteristik fisiologis, sifat ini diterapkan pada sistem pemrosesan yang dinamakan jaringan syaraf tiruan (JST, *artificial neural network*).

Salah satu contoh aplikatif adalah pengenalan citra visual, pola komplek yang terdiri dari sejumlah besar elemem (piksel) yang secara individual sangat sedikit memperlihatkan ciri suatu pola, tetapi secara kolektif menggambarkan suatu

---

[1] Freeman, J.A., dan Skapura, D.M., ' Neural NetworkAlgorithms, Applications, and Programming Technique ', Addison-Wesley Publishing Company, Massachussetts, 1991

objek yang dapat dikenal (oleh manusia). Bila pola tersebut berubah dengan waktu, bagaimanapun juga, komputasi sekuensial tidak akan mampu untuk melakukan tugasnya.

2.1 Gambaran Umum

Intelligent control telah diterima dan mendapat perhatian yang cukup baik dalam bidang sistem pengaturan secara umum. Secara umum untuk meng-implementasi-kan sistem tersebut menimbulkan beban komputasional yang berat, dan hal ini tak dapat dipisahkan dan membutuhkan perlakuan secara konkuren. Diantara berbagai arsitektur tersedia, JST merupakan subjek yang cukup hangat dari banyak penelitian, pada hakekatnya, memberi kecakapan belajar-sendiri (*self-learning*) kecepatan perosesan yang ultra cepat oleh adanya sifat paralel-masif. Dapat dikatakan, JST mengandung jumlah elemen pemroses yang cukup banyak -- *neuron-neuron* -- yang dihubungkan oleh bobot-bobot yang dapat di perbaharui, atau dilatih, selama digunakan. Aplikasi JST dalam masalah pengaturan merupakan '*state of the art*' dalam penelitian.[2]

Penerapan JST dalam sistem pengaturan telah mengalami perkembangan yang cukup pesat, karena tuntutan dari kontroler untuk menyediakan sinyal input yang tepat pada *plant*, sehingga menghasilkan tanggapan seperti yang diharapkan. *Neurocontroller* pertama dikembangkan oleh Widrow pada tahun 1963, yang mana telah dibentuk untuk pengaturan *inverted pendulum*. JST mempunyai banyak

---

[2] Rogers, E., dan Yun Li, *'Parallel Processing In A Control Systems Environment'* Prentice Hall, London, 1993

memiliki keuntungan, dan cukup pantas diaplikasikan pada *intelligent control* :

1. JST untuk mendekati suatu sistem dengan belajar (*self learning ability*).
2. Kemampuan untuk memetakan data input menjadi data output secara non-linier (*non-linier mapping*).
3. Kemampuan untuk mengolah sinyal input secara paralel dan serentak (*masively parallel distributed processing*), dan waktu-nyata (*real-time*).

JST yang cukup populer dalam *neurokontrol* adalah *multilayer perceptron* (MLP) (Narendra dan Parathasarathy, 1990), *functional link networks* (FLN) (Pao,1989), *B-spline networks*, *assosiative memory network* seperti Albus CMAC (Miller, 1987), dan *radial basis function* (RBF) (Chen dan Billings, 1992).

Multilayer perceptron (MLP) dilatih menggunakan algoritma seperti *backwards error propagation* (BEP), jaringan ini merupakan jaringan syaraf multi-lapis umpan-maju (*multilayer feedforward neural network*). Arsitektur multilayer perceptron neural network dengan algoritma back-propagation telah digunakan pada proses multi-variabel, plant MIMO (*multi input - multi output*), seperti oleh Narendra dan Parathasarathy[3]. Pendekatan yang diberikan tidak lagi berdasarkan inversed control, tetapi berdasarkan neural adaptive control, yaitu fungsi pengatur dan estimatornya digantikan oleh jaringan syaraf tiruan.

2.2 Kemampuan Belajar JST

Idealnya sistem pemrosesan untuk pengenalan pola tidak kita program secara eksplisit, tetapi sistem ini dapat "belajar" dan menentukan sendiri hubungan

---

[3] Narendra, K. S., dan Parathasarathy, K., *'Identification and Control of Dynamical Systems Using Neural Networks'* IEEE Trans. Neural Networks, 1, 1990, hal 4-27

antara sekelompok pola masukan dengan pola keluaran yang kita berikan, dan menggunakan hubungan ini pada saat kita memberikan pola yang baru.

Belajar (*learning*) bagi JST merupakan proses untuk mengatur parameter bobot yang terbaik, dengan melatih (*training*) jaringan akan diperoleh untuk kerja yang dikehendaki. Selama proses belajar, pembobot secara konvergen menuju pada harga tertentu, sehingga menghasilkan pola output seperti yang diinginkan. Dua metoda belajar bagi JST yaitu :

1. *Suverfised Learning* (belajar dengan pengawasan)

   Algoritma ini membutuhkan training pair atau pasangan dari tiap vektor input dengan vektor target, yang menyatakan output yang diinginkan. Biasanya JST dilatih dengan training pair.

   Contohnya adalah *back propagation network*, jaringan ini mempelajari sejumlah pola-masukan dan pola-keluaran dengan siklus dua fase, yaitu propagasi dan adaptasi. Setelah pola masukan diberikan pada lapis pertama jaringan, pola ini dipropagasi sampai lapis keluaran. Pola keluaran ini dibandingkan dengan pola yang diinginkan, kemuadian error dihitung untuk tiap sel keluarannya.

   Error ini kemudian ditransmisikan kebelakang melalui lapis antara (*hidden layer*), yang sebelumnya memberi konstribusi langsung pada lapis keluaran. Proses ini berlanjut lapis demi lapis, sampai tiap sel pada tiap lapis menerima sinyal error yang menggambarkan kontribusi relatifnya terhadap errot total. Berdasarkan sinyal error yang diterima, bobot interkoneksi tiap

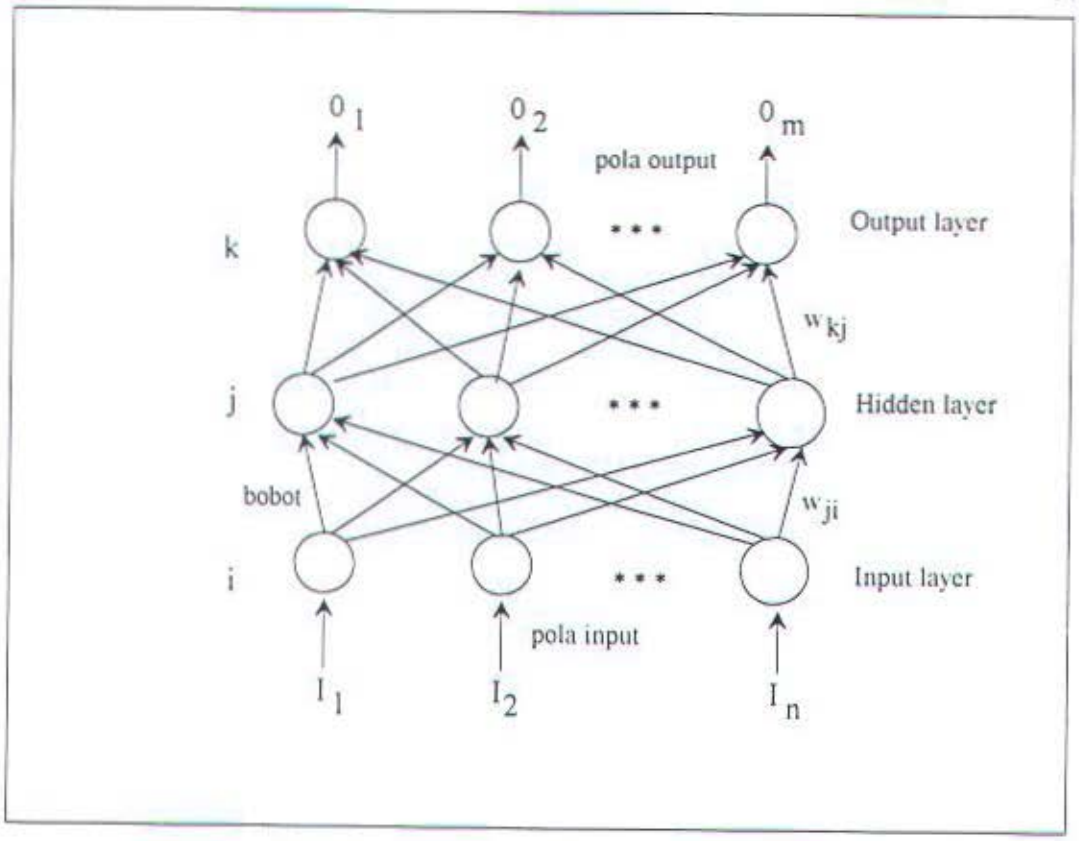
suatu sel tertentu diperbaharui, yang menyebabkan keseluruhan jaringan konvergen menuju keluaran yang diinginkan. Seluruh proses dilakukan berulang secara iteratif sampai diperoleh error yang dapat diterima.

2. *Unsuverfised Learning* (belajar tanpa pengawasan)

Metoda ini tidak memerlukan vektor target untuk outputnya, sehingga tidak ada perbandingan untuk menentukan vektor yang ideal. Kumpulan pola training hanya terdiri dari vektor input. Algoritma training hanya memodifikasi pembobot jaringan untuk menentukan vektor output yang konsisten, sehingga penerapan dua vektor training atau suatu vektor lain yang sejenis akan menghasilkan output yang sama. Dalam proses trining, jaringan mengklasifikasikan pola-pola input menjadi grup-grup yang sejenis. Penerapan satu vektor dari satu kelas tertentu pada inputnya akan menghasilkan pola output yang spesifik. Salah satu contoh metoda belajar ini adalah Kohonen.

2.3 Arsitektur Multilayer Perceptron

Arsitektur *multilayer perceptron* adalah jaringan *feedforward* (umpan-maju) multi-lapis tesusun dari lapis masukan, satu atau beberapa lapis hidden, dan lapis keluaran. Tiap sel pada setiap lapis saling terhubung secara lengkap dengan sel-sel pada lapis sebelum dan sesudahnya, tetapi tidak pada lapis yang sama. Dari gambar ini terlihat bahwa lapisan masukan menerima masukan dari sejumlah sensor atau reseptor yang tersusun paralel, dan data mengalir ke lapis berikutnya (yaitu lapis tersembunyi), dan akhirnya ke lapis keluaran.



Gambar 2.2 Arsitektur JST Multilayer

2.3.1 Algoritma *Backwards Error Propagation* (BEP)[4]

Sekumpulan vektor **P**, $\{(x_1,y_1), (x_2,y_2), \ldots, (x_p,y_p)\}$ merupakan contoh pemetaan fugnsional $y = F(x) : x \hat{I} R^N$, $y \hat{I} R^M$. Kita akan melatih JST sehingga dapat memperoleh keluaran pendekatan $o = y' = F'(x)$, metoda pelatihan ini dengan metoda yang umum dipakai.

Dari Gambar 2.2 suatu fektor masukan $x_p = (x_{p1}, x_{p2}, \ldots, x_{pN})^t$, diberikan pada input layer jaringan. Sel-sel masukan mendistribusikan nilai-nilai pada sel-sel lapis tersembunyi. Masukan *net* pada sel tersembunyi lapis ke-*j* adalah :

---

[4] Freeman, J.A., dan Skapura, D.M., ' Neural NetworkAlgorithms, Applications, and Programming Technique ', Addison-Wesley Publishing Company, Massachussetts, 1991

$$net^{h}_{pj} = \sum_{i=1}^{N} w^{h}_{ji} x_{pi} + \theta^{h}_{j} \quad (2.1)$$

dimana w^{h}_{ji} adalah bobot interkoneksi yang berasal dari sel masukan ke-*i*. Subskrip *h* berkaitan dengan nilai lapis tersembunyi. Diasumsikan bahwa aktifasi node ini sama dengan masukan *net*, sehingga keluaran node ini adalah :

$$i_{pj} = f^{h}_{j}(net^{h}_{pj}) \quad (2.2)$$

persamaan node-node keluaran adalah :

$$net^{o}_{pk} = \sum_{j=1}^{L} w^{o}_{kj} i_{pj} + \theta^{o}_{k} \quad (2.3)$$

$$o_{pk} = f^{o}_{k}(net^{h}_{pk}) \quad (2.4)$$

dimana superskrip *o* berkaitan dengan nilai lapis keluaran.

Prosedur untuk pelatihan jaringan adalah sebagai berikut :

1. Memberikan vektor masukan pada jaringan dan menghitung nilai (vektor) keluarannya.
2. Membandingkan keluaran-keluaran jaringan dengan keluaran yang diinginkan, yaitu menghitung error.
3. Menentukan arah (positif atau negatif) untuk mengubag setiap bobot agar dapat memperkecil error.
4. Menentukan besarnya perubahan setiap bobot.
5. Memberikan hasil koreksi pada bobot-bobot (memperbaiki bobot).
6. Mengulangi langkah 1 sampai 5 untuk semua vektor latihan sampai error dalam himpunan latihan berkurang sampai pada nilai yang diterima.

Aturan pengubahan bobot pada jaringan tanpa lapis tersembunyi dan sel keluarannya linier, disebut aturan *least mean square* (LMS) atau aturan delta :

$$w(t+1)_i = w(t)_i + 2\mu\, \varepsilon_k\, x_{ki} \qquad (2.5)$$

dimana m adalah konstanta positif, x_{ki} komponen ke-*i* dari vektor ke-*k*, dan $e_k = (d_k - y_k)$.

2.3.2 Perbaikan Bobot-bobot Lapis Keluaran

Dalam penurunan algoritma generalized learning rule (GRD), error vektor masukan ke-*k* adalah $e_k = (d_k - y_k)$ dimana d_k adalah keluaran yang diinginkan dan y_k adalah keluaran jaringan. Pada pembahasan ini akan dipakai notasi yang berbeda, karena menggunakan banyak sel untuk satu lapisnya. Pendefinisian error untuk sel keluaran adalah $\delta_{pk} = (y_{pk} - o_{pk})$, dimana subskrip p menyatakan vektor latihan ke-p, dan k keluaran sel ke-k.

Error yang diminimasi oleh algoritma GRD adalah jumlah kuadrat error untuk semua sel keluaran :

$$E_p = \tfrac{1}{2} \sum_{k=1}^{M} \delta_{pk}^{\,2} \qquad (2.6)$$

Faktor ½ dalam persamaan diatas dipakai untuk menyesuaikan proses perhitungan turunan nanti. Karena konstanta sembarang akan muncul pada hasil akhir, adanya faktor tersebut akan membuat turunan tetap berlaku.

Penetapan arah pengubahan bobot-bobot memerlukan perhitungan gradien negatif E_p, ∇E_p berkenaan dengan bobot-bobot w_{ij}. Kemudian, nilai-nilai

bobot disesuaikan sampai error keseluruhan berkurang. Untuk mempermudah persoalan, setiap komponen ∇Ep dihitung secara terpisah. Dari persamaan (2.6) dan definisi δ_{pk}, diperoleh :

$$E_p = \frac{1}{2} \sum_{k=1}^{M} (y_{pk} - o_{pk})^2 \tag{2.7}$$

dan

$$\frac{\partial E_p}{\partial w_{kj}^o} = -(y_{pk} - o_{pk}) \cdot \frac{\partial f_k^o}{\partial (net_{pk}^o)} \frac{\partial (net_{pk}^o)}{\partial w_{kj}^o} \tag{2.8}$$

Kita telah menggunakan persamaan (2.4) untuk nilai keluaran, o_{pk}, dan aturan rantai (*chain rule*) untuk turunan parsial. Untuk sementara, kita tidak akan mengevaluasi turunan f_k^o secara langsung, tetapi dituliskan secara sederhana sebagai $f_k^{o'}(net_{pk}^o)$. Faktor terakhir dalam persamaan (2.8) adalah :

$$\frac{\partial (net_{pk}^o)}{\partial w_{kj}^o} = -\left(\frac{\partial}{\partial w_{kj}^o} \sum_{j=1}^{J_k} w_{kj}^o i_{pj} + \theta_k^p\right) \tag{2.9}$$

Dengan menggunakan persamaan (2.8) dan (2.9), kita peroleh negatif gradien :

$$-\frac{\partial E_p}{\partial w_{kj}^o} = -(y_{pk} - o_{pk}) f_k^{o'}(net_{pk}^o) i_{pj} \tag{2.10}$$

Selama perubahan *magnitude* bobot dilakukan, perubahan diambil sebanding dengan negatif gradien. Kemudian bobot-bobot pada lapis keluaran diperbaiki dengan mengikuti persamaan :

$$w_{kj}^o(t+1) = w_{kj}^o(t) + \Delta_p w_{kj}^o(t) \tag{2.11}$$

dimana

$$\Delta_p w_{kj}^o = \eta (y_{pk} - o_{pk}) f_k^{o'}(net_{pk}^o) i_{pj} \tag{2.12}$$

Faktor η disebut konstanta belajar, umumnya nilai η diambil pada selang (0,1).

Kembali pada fungsi $f_k^{o'}$, pertama fungsi f_k^o harus diferensiabel. Persyaratan ini menghapuskan kemungkinan penggunaan suatu *threshold* satuan linier, karena fungsi keluaran untuk *threshold* satuan tidak diferensiabel pada nilai *threshold*-nya.

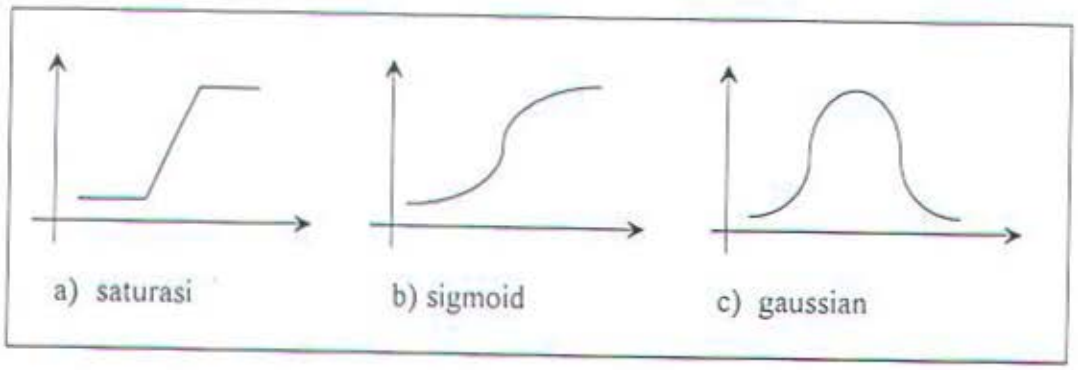
Terdapat dua fungsi keluaran yang dapat dipakai disini :

$$f_k^o(net_{jk}^o) = net_{jk}^o \tag{2.12a}$$

dan

$$f_k^o(net_{jk}^o) = \left(1 - e^{-net_{jk}^o}\right)^{-1} \tag{2.12b}$$

Fungsi pertama menyatakan fungsi keluaran linier satuan. Fungsi yang kedua disebut fungsi sigmoid, atau fungsi logistik. Pemilihan fungsi keluaran bergantung pada data keluaran, tetapi pada umumnya fungsi keluaran sigmoid lebih banyak dipakai, karena memiliki keluaran yang terbatas tetapi juga diferensiabel, sehingga cocok untuk kebanyakan kasus.



Gambar 2.2 Fungsi Aktivasi

Untuk kasus pertama

$$f_k^{o'}(net_{jk}^o) = 1 \tag{2.12c}$$

sedangkan untuk kasus kedua,

$$f_k^{o'}(net_{jk}^o) = f_k^o(1 - f_k^o) \qquad (2.12d)$$
$$= o_{pk}(1 - o_{pk})$$

Untuk kedua kasus diperoleh berturut-turut :

$$w_{kj}^o(t+1) = w_{kj}^o(t) + \eta(y_{pk} - o_{pk})\, i_{pj} \qquad (2.13)$$

dan

$$w_{kj}^o(t+1) = w_{kj}^o(t) + \eta(y_{pk} - o_{pk}) o_{pk}(o_{pk} - 1)\, i_{pj} \qquad (2.14)$$

Persamaan (2.13) untuk keluaran linier dan persamaan (2.14) untuk persamaan sigmoid.

Untuk meringkas persamaan perbaikan bobot, kita gunakan definisi :

$$\delta_{kj}^o = (y_{pk} - o_{pk}) f_k^{o'}\left(net_{pk}^o\right) \qquad (2.15)$$
$$= \delta_{pk}^o f_k^{o'}(net_{pk}^o)$$

Kemudian persamaan perbaikan bobot dapat dituliskan sebagai :

$$w_{kj}^o(t+1) = w_{kj}^o(t) + \eta\, \delta_{pk}^o\, i_{pj} \qquad (2.16)$$

Tanpa memperhatikan bentuk fungsional keluaran, f_k^o, error yang diminimasi pada proses ini adalah :

$$E_p = \sum_{p=1}^{P} E_p \qquad (2.17)$$

2.4 Perbaikan Bobot-bobot Lapis Tersembunyi

Perhitungan lapis tersembunyi menggunakan cara yang sama dengan yang dilakukan pada lapis keluaran. Masalah muncul pada penentuan pengukuran error

keluaran-keluaran lapis tersembunyi. Keluaran sel diketahui, tetapi tidak ada cara untuk mengetahui lebih lanjut mengenai bagaimana seharusnya keluaran sel-sel lapis tersembunyi ini. Secara intuitif, error total E_p harus sedemikian rupa sehingga dapat dihubungkan dengan nilai keluaran pada lapis tersembunyi. Pembenaran intuisi ini dapat dilakukan dengan menggunakan persamaan (2.7) :

$$E_p = \frac{1}{2}\sum_{k=1}^{M}(y_{pk} - o_{pk})^2$$

$$= \frac{1}{2}\sum_{k=1}^{M}(y_{pk} - f_k^o(net_{pk}^o))^2$$

$$= \frac{1}{2}\sum_{k=1}^{M}(y_{pk} - f_k^o(\sum_j w_{kj}^o\, i_{pj} + \theta_{pk}^o))^2$$

Kita ketahui bahwa i_{pj} tergantung bobot pada lapis tersembunyi melalui persamaan (2.2). Berdasarkan kenyataan ini, kita dapat menghitung gradien E_p berdasarkan bobot-bobot pada lapis tersembunyi.

$$\frac{\partial E_p}{\partial w_{ji}^h} = \frac{1}{2}\sum_k \frac{\partial}{\partial w_{ji}^h}(y_{pk} - o_{pk})^2$$

$$= -\sum_k (y_{pk} - o_{pk})\frac{\partial o_{pk}}{\partial(net_{pk}^o)}\frac{\partial(net_{pk}^o)}{\partial i_{pj}}\frac{\partial i_{pj}}{\partial(net_{pj}^o)}\frac{\partial((net_{pj}^h)}{\partial w_{ji}^h} \qquad (2.18)$$

Setiap faktor pada persamaan (2.18) dapat dihitung secara eksplisit dari persamaan sebelumnya. Hasilnya adalah :

$$\frac{\partial E_p}{\partial w_{ji}^h} = -\sum_k (y_{pk} - o_{pk}).f_k^{o'}(net_{pk}^o).w_{kj}^o.f_j^{h'}(net_{pk}^h).x_{pi} \qquad (2.19)$$

Perbaikan bobot-bobot lapis tersembunyi sebanding dengan negatif persamaan (2.20) :

$$\Delta_p w_{ji}^h = \eta f_j^{h'}(net_{pj}^h)\, x_{pi} \sum_k (y_{pk} - o_{pk}) f_k^{o'}(net_{pk}^o).w_{kj}^o \qquad (2.20)$$

dimana η adalah konstanta belajar.

Definisi δ^o_{pk} yang digunakan sebelumnya dapat digunakan di sini dan dituliskan sebagai :

$$\Delta_p w_{ji}^h = \eta f_j^{h'}(net_{pj}^h)\, x_{pi} \sum_k \delta_{pk}^o w_{kj}^o \qquad (2.21)$$

Perhatikan bahwa setiap perbaikan bobot pada lapis tersembunyi tergantung pada semua error lapis keluaran δ^o_{pk}. Hal inilah yang mengakibatkan munculnya nama propagasi balik (*backpropagation*). Error-error yang ditemukan pada lapis keluaran dipropagasikan ke belakang ke lapis tersembunyi untuk menentukan perubahan bobot yang sesuai untuk lapis tersebut. Mendefinisikan error lapis tersembunyi sebagai

$$\delta_{pj}^h = \eta f_j^{h'}(net_{pj}^h)\, x_{pi} \sum_k \delta_{pk}^o w_{kj}^o \qquad (2.22)$$

menyebabkan persamaan perbaikan bobot untuk lapis tersembunyi analog dengan perbaikan bobot pada lapis keluaran

$$w_{ij}^h(t+1) = w_{ji}^h + \eta\, \delta_{pj}^h x_i \qquad (2.23)$$

Inilah akhir lingkaran perhitungan GDR. Perhatikan, kedua persamaan (2.16) dan persamaan (2.23) memiliki bentuk yang sama dengan persamaan (2.5) pada aturan delta.

2.4 Sistem Pengaturan Berbasis Jaringan Syaraf Tiruan

Adanya kemampuan belajar yang dimiliki JST dapat dipakai untuk meng-identifikasi-kan karakteristik proses fisik (*plant*) dan beradaptasi terhadap perubahan lingkungan. Sifat pemetaan yang non-linier memungkinkan JST dapat digunakan pada proses (*plant*) yang non-linier dan dinamis. JST memiliki arsitektur yang paralel dan memiliki sifat kebal terhadap kegagalan (*fault tollerance*), ketika sebagian elemen pengolah rusak.

Intelligent controllers (pengatur cerdas) mampu menentukan struktur dan aksi suatu proses berdasarkan tanggapan dari pengamatan kelakuan input/output suatu *plant* dengan referensi yang minimal untuk memodelkan atau mengetahui persamaan karakteristik suatu *plant*. Kebanyakan dari suatu proses secara non-linier, dinamis yang berubah terhadap waktu, dan kompleks, maka diperlukan suatu pengatur cerdas. Salah satunya adalah *neural net based controllers* (*neurocontrollers*).

Ketika kita mempertimbangkan kegunaan JST sebagai suatu controller sistem dinamik, maka mesti dibandingkan dengan algorithms control konvensional, terutama dengan teori control adaptip. Karena maksud teori control adaptip adalah untuk menjalankan kecakapan atau kemampuan controller yang fleksibel dari penyesuaian parameter yang tidak diketahuil. Tujuan dari controller JST adalah sama yaitu melaksanakan suatu kemampuan controller yang fleksibel seperti controller adaptip.

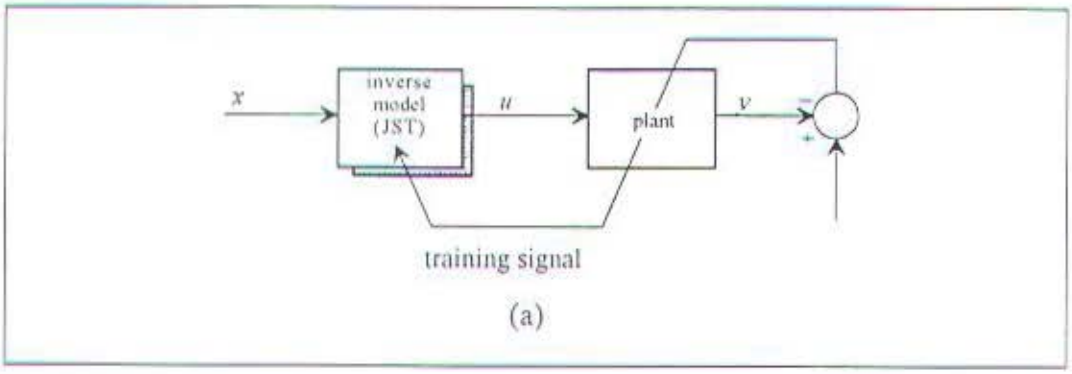
Kontroller JST diharapkan mempunyai beberapa kelebihan diantaranya struktur yang fleksibel untuk menyatakan sistem yang nonlinear dan kemampuan belajar untuk menyesuaikan dengan karakteristik plant yang dikontrol untuk mendapatkan suatu skematik control baru. Klasifikasi dari arsitektur controller JST dinamik dengan pendekatan pada *Neural Adaptive Controller.*

2.4.1 Identifikasi *Inverse Plant*

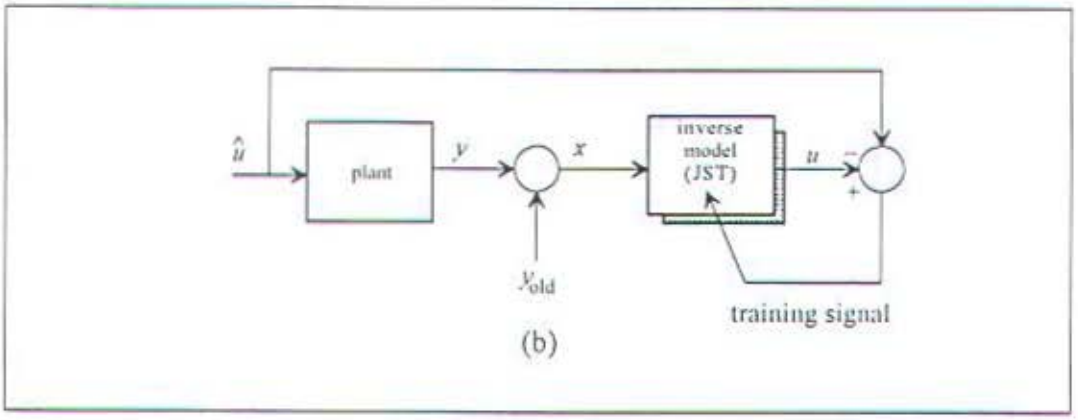
Suatu plant sistem dinamik menggunakan feedforward controller untuk memperbaiki kerugian (mengkompensasi) variasi parameter yang nonlinier. Gambar diatas memperlihatkan sebuah contoh dari JST direct controller. Dalam sistem ini JST menjalankan sebuah inverse dinamic dari plant. Keuntungan JST direct controller adalah untuk membuat sebahagian besar kegunaan dari kemampuan yang dimiliki JST. Walaupun sistem ini dapat menghilangkan kekuatan (force) pada permulaan control, karena hal itu tergantung dari matrik pembobot mula (initial weight matrix) dari JST.

Identifikasi sistem diartikan sebagai proses menentukan model sistem (*plant*) dari data input-output. Untuk mengetahui karakteristik suatu plant dan agar dapat beradaptasi dengan suatu proses, kontroller harus mengetahui parameter-parameter kontrol, berdasarkan data outputnya. Umumnya informasi ini tidak diperoleh, yang ada hanya error keluaran suatu plant. Dalam inverse control, JST dilatih sebagai inverse dynamic dari plant, digunakan sebagai *tracking* plant agar menghasilkan output yang diinginkan.

Suatu kontroller dikatakan memiliki bentuk inverse dari suatu plant, jika kontroller tersebut dapat memerankan dirinya sebagai sistem dengan fungsi alih yang merupakan kebalikan dari fungsi alih plant yang dikendalikan. Jika plant yang dikendalikan mempunyai fungsi alih y = f(x,t) maka kontroler harus mempunyai fungsi alih mendekati x = g(y,t). Dengan demikian, keluaran plant yang diinginkan mengikuti masukan yang diberikan kontroler[5].



Gambar 2.5 Model Inverse Plant
JST On-line controller



Gambar 2.6 Pemodelan Off-line Controller

[5] Brown, M., dan C.J. Harris, '*The B-Spline Neurocontroller*' in Parallel Processing In A Control Systems Environment, Prentice Hall Int. London, 1993

Cost function dari kedua metoda pelatihan yaitu persamaan (2.24) untuk pelatihan off-line, dan persamaan (2.25) untuk pelatihan on-line.

$$J_u = \frac{1}{2T} * \sum_{t=1}^{T} [\hat{u}(t) - u(t)]^2 \qquad (2.24)$$

$$J_y = \frac{1}{2T} * \sum_{t=1}^{T} [\hat{y}(t) - y(t)]^2 \qquad (2.25)$$

dimama u(t) adalah sinyal kontrol, y(t) adalah output plant, tanda ^ adalah harga yang diinginkan.

Untuk mendapatkan *inverse dynamic* dari plant, JST dapat dilatih berdasarkan perbandingan antara output yang didinginkan dengan output aktual dari plant, dan menggunakan perbedaan ini untuk menyesuaikan parameter jaringan (identifikasi), mengubah bobot-bobot interkoneksi antar sel-selnya, sehingga diperoleh hubungan input-output mendekati fungsi yang diinginkan.

2.4.2 Adaptasi Parameter

Untuk dapat menurunkan cost function (daerah error tipikal) JST akan dilatih untuk menyesuaikan parameter, fungsi adaptasi pembobot digunakan allgotithma gradien dengan persamaan

$$\Delta w_i(t-1) = -\delta_i \frac{\delta J(t)}{\delta w_i(t-1)} \qquad (2.26)$$

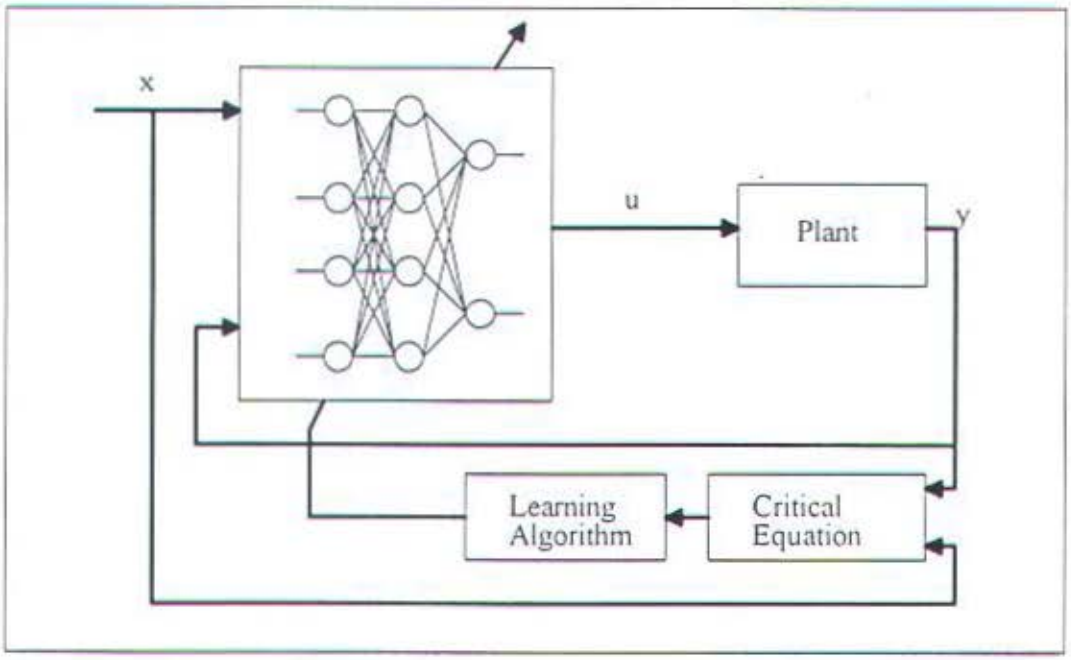
dimana Δw_i (t-1) adalah $w_i(t)$ - $w_i(t$-1), dan δ_i adalah laju belajar ke-i.

Pada pelatihan off-line perbedaan $e_u(t) = [\ddot{u}(t) - u(t)]$ jika suatu sinyal kontrol diberikan pada plant, kemudian diambil keadaan plant saat itu sebagai input vektor $x(t)$, Sehingga output kontroler untuk suatu input $x(t)$, dapat dibandingkan dengan sinyal kontrol yang diinginkan $\ddot{u}(t)$, error ini digunakan melatih kontroler menggunakan algoritma belajar *least-mean square* (LMS) seperti dibawah

$$\Delta w_i(t-1)=\delta_i \varepsilon_u(t)\, x_i(t) \qquad (2.27)$$

Metoda ini juga dapat digunakan pada pelatihan on-line, jika keadaan plant , karena input sebelumnya, adalah relatif. Maka output kontroller dibandingkan lagi dengan control sinyal aktual, yaitu sinyal control yang diinginkan.

Adaptasi on-line umumnya menggunakan gradien sebagai algoritma LMS sebagai dasar untuk memperbaiki error *cost function*, J_y dan J_u.



Gambar 2.27 Sistem Pengaturan Berbasis Jaringan Syaraf Tiruan

BAB III
TRANSPUTER

Transputer adalah suatu sistem mikrokomputer yang di dalamnya sudah tercakup prosesor, memori lokal, dan link komunikasi titik-ke-titik untuk menghubungkan satu transputer dengan transputer lainnya. Sebuah transputer dapat digunakan sebagai sistem prosesor tunggal atau jaringan transputer untuk membentuk sistem konkuren.

Transputer dirancang untuk menjadi prosesor serba guna. Nama transputer merupakan singkatan dari "*TRANSistor comPUTER*", yang menyiratkan bahwa piranti tersebut adalah sebuah komputer, dan memiliki kemampuan sebagai blok pembangunan (*building block*), seperti halnya transistor pada awal tahun 60-an. Transputer memiliki kecepatan proses yang baik (T414 10 MIPS, T805 30 MIPS, T9000 150 MIPS; bandingkan 68020 sekitar 2.5 MIPS), integrasi tingkat tinggi lebih fleksibel dengan adanya antarmuka memori eksternal bersifat dapat diprogram (*programmable*), dan *on-chip communications* yang memudahkan pengoperasian untuk sistem multiprosesor.

Transputer telah digunakan mulai dari *embedded system* (sistem yang ditambahkan pada sistem lain) sampai pada superkomputer. Bidang aplikasi transputer sangat luas, meliputi aplikasi ilmiah dan matematika, sistem multiprosesor kecepatan tinggi, pengolah grafik kinerja tinggi, superkomputer,

workstations, pengolah sinyal digital, basis data terdistribusi, simulasi sitem, telekomunikasi, robotika, sistem *fault tolerant*, pengolahan citra, pengenalan pola, dan kecerdasan buatan.

3.1 Sejarah Singkat Transputer

Pada awal tahun delapan puluhan harga memori cukup mahal sehingga pada umumnya prosesor hanya mampu mengalamati memori yang kecil (64 kbyte). Atas dasar ini pula, set instruksi prosesor dirancang agar hanya memakai memori yang kecil, yaitu dengan memiliki banyak set instruksi. Dengan pendekatan ini, untuk menjalankan suatu instruksi hanya diperlukan beberapa byte saja, bahkan untuk instruksi yang rumit. Arsitektur komputer semacam ini dekenal dengan *CISC (Complex Instruction Set Computer)*.

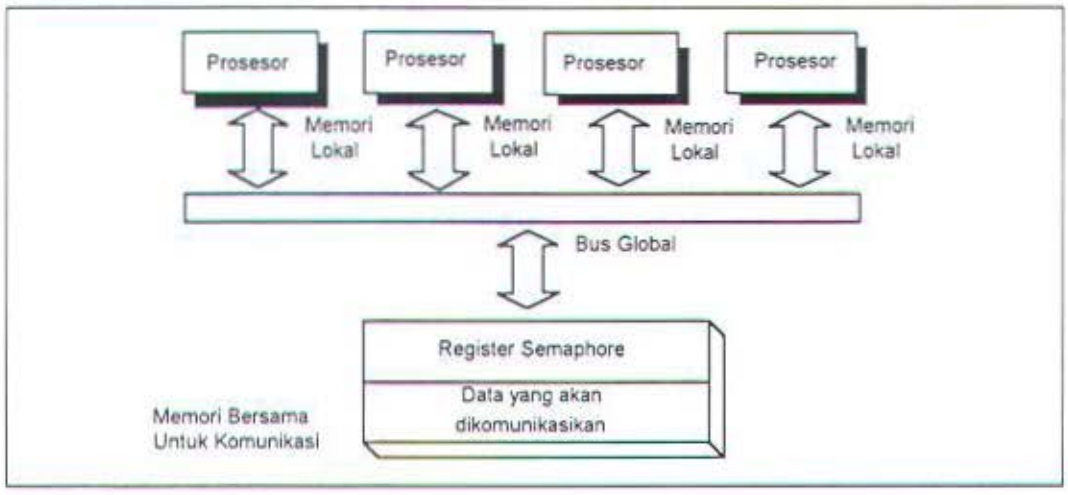
Pada saat berbagai pekerjaan dilakukan dengan menggunakan set instruksi ini (pada saat bahasa tingkat tinggi dikompilasi) ternyata sekitar 90 % dari waktu eksekusi digunakan hanya 10% dari set instruksi ini. Kenyataan ini melahirkan arsitektur *RISC (Reduced Instructions Set Computer)* yang menggunakan ide bahwa instruksi yang sering digunakan tersebut (yang 10%) dirancang agar beroperasi dengan 1 siklus, dan instruksi yang lebih rumit tetapi jarang digunakan dibuat dengan rutin perangkat lunak.

Rancangan dasar trasputer didasarkan pada konsep RISC ini, dengan modifikasi untuk meningkatkan kenerjanya, diantaranya penggunaan mikrokode pada instruksi tingkat tinggi (misal perkalian), tidak semata-mata menggunakan

loop dan compare, yang menghasilkan kode secepat RISC, tetapi mengurangi pemakaian register. Modifikasi dan peningkatan lainnya menghasilkan hal-hal sebagai berikut :

1. CPU tidak perlu mengakses bus pada tiap siklus (aslinya, hal ini hanya berlaku untuk CISC, dan tidak untuk RISC murni yang menggunakan konsep satu instruksi satu- siklus).
2. Penjadwalan proses (*process/task scheduling*) dapat dibuat dengan mikrokode, hal ini dapat menghasilkan penyaklaran (*switching*) yang efisien, berorde sepersekian mikrodetik bila tidak banyak register yang harus disimpan (*pop* dan *push*) isinya.
3. Komunikasi proses antar prosesor dilakukan dengan menyalin data dari satu prosesor ke prosesor yang lain.

Ada dua cara pemindahan data antar prosesor dalam sebuah sistem, yaitu pelaluan pesan (*mesage passing*) dan penggunaan-bersama memori (*shared memory*).



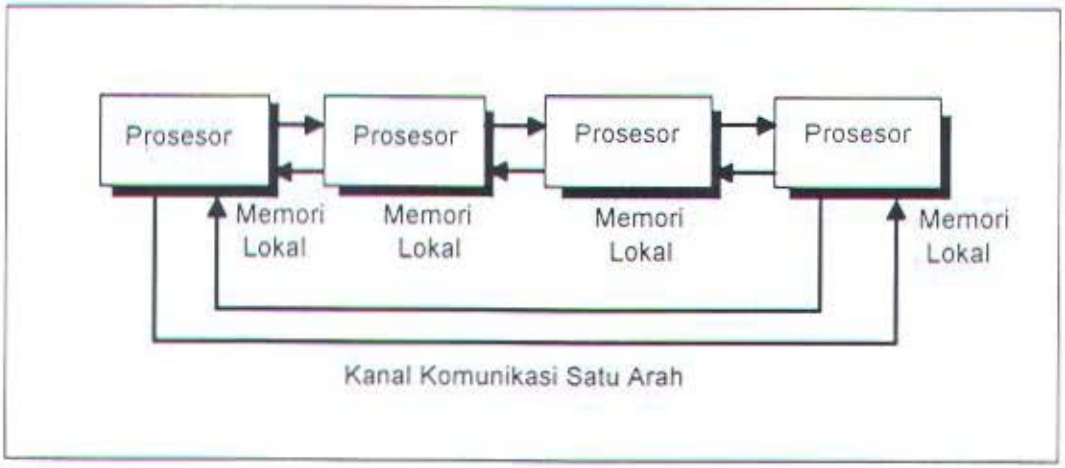
Gambar 3.2 Sistem Memori-bersama

Penggunaan bersama memori memerlukan instruksi bus khusus, misalnya suatu prosesor membaca suatu lokasi memori dan mengubahnya (siklus baca-tulis-ubah), dengan jaminan prosesor lain tidak dimungkinkan untuk menginterupsi, maka sebuah lokasi memori digunakan sebagai register *flag* atau *semaphore*. Metode komunikasi seperti ini memerlukan *poll* perangkat lunak, atau interupsi perangkat keras ke prosesor tujuan yang menandakan bahwa data sedang dikirimkan padanya. Kerugian lain metode ini adalah keterbatasan bandwidth bus. Makin banyak prosesor, makin besar *bandwidth* yang diperlukan, dan makin lama waktu yang diperlukan untuk suatu proses.

Pelaluan pesan biasanya dilakukan dengan menyalin satu blok memori pada sebuah prosesor untuk diberikan pada prosesor lain, melalui suatu antarmukan (*interface*). Meskipun sederhana, metode ini jarang digunakan karena proses menjadi lambat untuk jumlah data yang besar, dan biasanya implementasinya menggunakan interupsi pada prosesor, seperti *DMA (Direct Memory Acces)*, sehingga mengganggu kenerja prosesor tersebut.

Metode terakhir ini diterapkan dalam transputer, tetapi untuk meningkatkan efisiensi digunakan kontrol perangkat keras dan perangkat lunak sekaligus, inilah *link* transputer. *Link* ini berupa *link* serial berkecepatan tinggi (20 Mbps, *megabit per second*) dengan dua pengendali DMA, satu untuk masukan dan satu untuk keluaran. Link ini merupakan hubungan titik-ke-titik yang menjadi penghubung antar transputer, atau transputer dengan piranti lainnya. Tidak ada *arbitrasi* (pengaturan) yang diperlukan. Pengendali DMA dapat dirancang untuk

pemindahan data masukan, keluaran, atau kedua-duanya secara bersamaan, tanpa ketergantungan satu sama lain *(independent)*. Data ini dapat berasal dari atau untuk ke memori internal atau eksternal, tanpa campur tangan lebih lanjut dari CPU. Yang diperlukan hanyalah ketersediaan *bandwidth* memori. Penggunaan mikrokode, sehingga semua instruksi dilaksanakan hanya dengan satu siklus (berarti memerlukan akses memori tiap siklus), menghasilkan sistem pelaluan pesan yang efisien dan transparan untuk multiprosesing.



Gambar 3.3 Sistem Pelaluan Pesan[6]

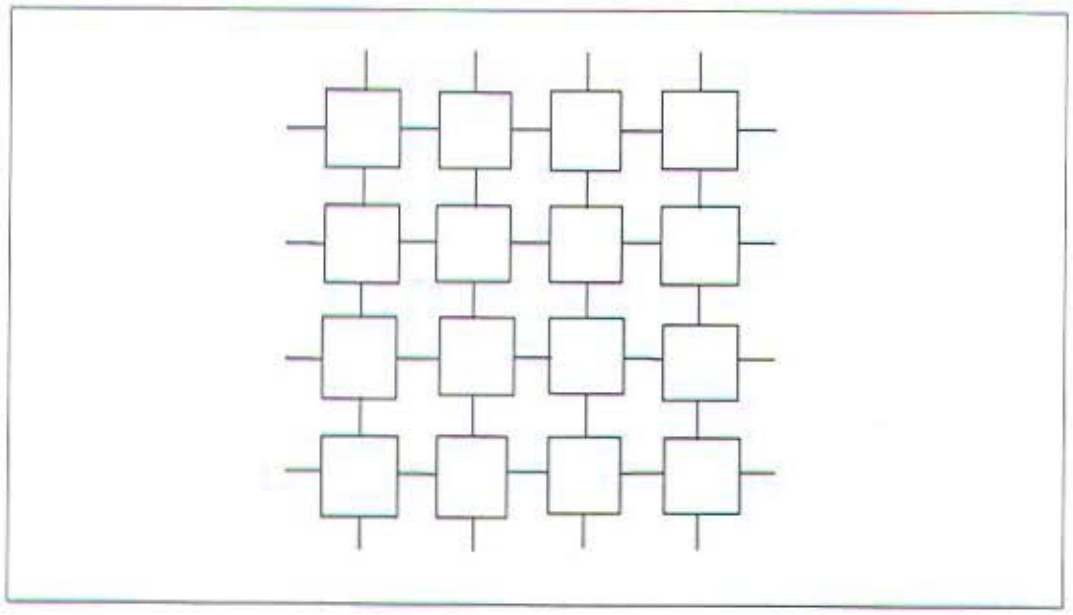
Kelemahan hubungan titik-ke-titik ini adalah terbatasnya interkoneksi. Bandwidth untuk metode komunikasi titik-ke-titik akan bertambah sejalan dengan bertambahnya jumlah prosesor, meskipun "diameter" bertambah. (Diameter dalam terminologi multiprosesor adalah jumlah prosesor antara yang harus dilalui oleh pesan). Di lain pihak, topologi hiperkubus memiliki diameter yang kecil, tetapi jumlah prosesor harus merupakan pangkat dari dua.

---

[6] Occam and Transputer A Workbook, Computer System Architects, Utah, 1990

3.2 Gambaran Umum Transputer

Definisi arsitektur transputer dapat ditinjau dari dua aspek, yaitu aspek logik dan aspek fisik. Aspek logik mendefinisikan bagaimana suatu sistem yang terdiri atas transputer-transputer yang saling berhubungan dirancang dan diprogram, sedangkan aspek fisik mendefinisikan bagaimana transputer sebagai komponen VLSI dihubungkan dan dikendalikan.



Gambar 3.4 Jaringan Transputer[7]

Sebuah transputer dapat digunakan pada sistem prosesor tunggal atau pada jaringan transputer untuk membentuk sistem paralel kinerja tinggi. Kinerja sistem yang dibangun dari susunan transputer tergantung dari jumlah transputer, kecepatan komunikasi antar transputer, dan kinerja tiap transputer itu sendiri.

---

[7] *Transputer Architerture and Overview*, Computer System Architects INMOS Limited, 1990

Transputer dikombinasikan sebagai blok-blok pembangun dalam jaringan multiprosesor untuk mencapai peningkatan kinerja yang hampir linear. Dengan kata lain, sebuah jaringan yang terdiri atas N transputer akan mengeksekusi kodenya hampir N kali lebih cepat daripada sistem transputer tunggal.

Transputer dapat diprogram dengan beberapa bahasa tingkat tinggi. Jika dibutuhkan sifar konkuren, tetapi tetap dengan menggunakan bahasa tingkat tinggi standard (yang telah ditambahkan fasilitas pemrograman paralel). Occam dapat dipergunakan sebagai alat untuk menghubungkan modul-modul yang ditulis dengan bahsa tersebut.

Untuk mendapatkan keuntungan yang sebesar-besarnya dari arsitektur transputer, seluruh sistem dapat diprogram dengan Occam. Dengan cara ini akan diperoleh semua keuntungan dan kemudahan bahasa tingkat tinggi, sementara kita dapat tetap menjaga efisiensi program secara maksimum, dan mempergunakan sifat khusus transputer.

Occam menyediakan kerangka kerja untuk merancang sistem konkuren menggunakan transputer dengan cara yang sama seperti halnya aljabar Boolean digunakan dalam merancang sistem elektronika dari gerbang logika. Tugas perancang sistem dipermudah karena dekatnya hubungan arsitektural antara Occam dan transputer. Sebuah program yang dijalankan pada transputer setara dengan sebuah proses Occam, jaringan kerja ransputer dinyatakan sebagai program Occam.

3.2.1 Perancangan Sistem

Arsitektur transputer dapat menyederhanakan perancangan sistem, yaitu dengan digunakannya proses sebagai blok pembangunan perangkat lunak dan perangkat keras.

Seluruh sistem dapat dirancang dan diprogram dalam Occam, mulai dari konfigurasi sistem sampai ke level rendah masukan/keluaran (I/O) dan interupsi waktu-nyata (*real time*).

3.2.1.1 Pemrograman

Blok pembangunan perangkat lunak disebut dengan **proses**. Suatu sistem dirancang dalam wujud sejumlah proses yang saling berhubungan. Proses dapat dipandang sebagai satuan perancangan yang independen. Sebuah proses berhubungan dengan proses lain melalui kanal titik-ke-titik. Rancangan internalnya tersembunyi dan benar-benar hanya dinyatakan dengan pesan-pesan yang dikirimkan dan diterima. Komunikasi antar proses telah disinkronisasi, sehingga tidak diperlukan mekanisme sinkronisasi yang terpisah.

Secara internal, setiap proses dapat dirancang sebagai sekumpulan proses yang berkomunikasi. Dengan demikian, perancangan sistem menjadi terstruktur secara hirarkis. Pada setiap level rancangan, perancang hanya perlu memperhatikan sejumlah proses kecil yang bisa diatur-atur. Occam didasari oleh konsep-konsep ini, dan memberikan definisi arsitektur transputer dari sudut pandang rancangan logik.

3.2.1.2 Perangkat Keras

Proses dapat diimplementasikan dalam perangkat keras. Suatu transputer yang melaksanakan program Occam adalah sebuah proses perangkat keras. Proses tersebut dapat dirancang dan dikompilasi secara independen. Detekankan lagi disini, struktur internalnya tersembunyi. Sebuah transputer berhubungan dan melakukan sinkronisasi dengan transputer lain melalui *link*-nya, yang mewujudkan kanal Occam.

Implementasi perangkat keras lain dari proses juga dimungkinkan. Contohnya, sebuah trasputer dengan set instruksi yang berbeda dapat digunakan untuk memberikan kompromi rasio kinerja yang lain. Atau sebuah proses dapat diperangkatkeraskan secara *hard-wired* kalau diinginkan peningkatan kinerja.

Kemampuan transputer menggunakan proses yang diperangkatkeraskan memberikan kerangka kerja untuk merancang sistem dengan kemampuan khusus, misalnya pengolah grafik. Fungsi yang diinginkan didefinisikan sebagai sebuah proses Occam, dan diimplementasikan dalam perangkat keras dan dengan standard antarmuka kanal Occam.

3.2.1.3 Komponen Yang Dapat Diprogram

Sebuah transputer dapat diprogram untuk melaksanakan fungsi khusus, dan selanjutnya dapat dianggap sebagai sebuah kotak hitam. Seperti telah dibahas pada bagian sebelumnya, beberapa proses dapat diperangkatkeraskan untuk meningkatkan kinerja.

Sebuah sistem yang diwujudkan dalam transputer tunggal dapat dibangun dari kombinasi proses perangkat lunak, transputer yang telah dipraprogram, dan proses perangkat keras. Sistem seperti itu dapat dipandang sebagai sebuah komponen untuk sistem yang lebih besar.

Transputer dirancang untuk mengeksploitasi potensi teknologi VLSI yang memungkinkan sejumlah piranti identik diproduksi. Untuk alasan ini, mengimplementasikan sistem konkuren dengan menggunakan sejumlah komponen yang dapat desesuaikan dengan program yang diperlukan merupakan pekerjaan yang cukup menjanjikan.

Arsitektur transputer telah dirancang sedemikian sehingga dapat bertindak sebagai komponen yang dapat diprogram yang dapat membentuk topologi yang diinginkan, dan hanya dibatasi oleh jumlah link setiap transputer. Arsitektur tersebut memberikan kebebasan ukuran sistem. Di lain pihak, struktur hirarkis yang dimiliki Occam telah menyederhanakan pekerjaan perancangan sistem dam pemrograman. Seluruh sistem dapat dirancang dan diprogram dengan Occam.

Semua ini bertujuan untuk meningkatkan kinerja setiap aplikasi yang dibuat, yaitu dengan mengeksploitasi sifat konkuren pada sejumlah besar komponen yang dapat diprogram.

3.2.1.4 Penjadwalan Proses Dimikrokodekan

Cara yang paling efektif untuk mengimplementasikan program sederhana dengan komputer yang dapat diprogram adalah dengan pemroses sekuensial. Oleh

karena itu, transputer memiliki pemroses konvensional yang dimikrokodekan. Ada sejumlah 32 instruksi yang digunakan untuk mengimplementasikan program sekuensial sederhana. Selain itu, sejumlah instruksi khusus yang menyediakan fasilitas seperti fungsi aritmetika dan penjadwal proses (*process scheduler*).

Sebuah proses yang dijalankan dalam sebuah transputer mungkin saja terdiri atas sejumlah proses konkuren yang harus didukung oleh transputer dalam model pemrograman Occam. Transputer, oleh karena itu, memiliki penjadwal yang dimikrokodekan untuk membagi bersama waktu prosesor di antara proses-proses konkuren tersebut. Penjadwal menyediakan dua level prioritas. Sembarang proses prioritas tinggi dapat menyela proses prioritas rendah dan segera dijalankan.

3.2.2 Arsitektur Sistem

3.2.2.1 Link Komunikasi Titik-ke-titik

Arsitektur transputer telah menyederhanakan perancangan sisten dengan digunakannya link komunikasi titik-ke-titik. Setiap anggota keluarga transputer mempunyai satu atau lebih link yang standard, yang masing-masing dapat dihubungkan dengan link dari beberapa komponen lain. Hal ini memungkinkan pembentukan jaringan transputer dengan berbagai ukuran dan topologi.

Link komunikasi titik-ke-titik mempunyai beberapa keuntungan dibandingkan dengan bus multiprosesor.

1. Tidak ada aturan mekanisme komunikasi, berapapun jumlah transputer pada sistem.

2. Tidak ada akibat beban kapasitif sejalan dengan penambahan transputer.
3. Bandwidth komunikasi tidak akan jenuh sejalan dengan penambahan ukuran sistem. Sebaliknya, semakin banyak transputer dalam sistem, bandwidth komunikasi lokal makin besar. Betapapun besarnya sistem, seluruh hubungan antar transputer dapat dibuat pendek dan lokal.

3.2.2.2 Memori Lokal

Setiap transputer mempunyai memori lokal. Keseluruhan *bandwidth* memori sebanding dengan jumlah transputer pada sistem, dan ini berkebalikan dengan pemakaian memori global yang besar, yang harus dibagi untuk setiap prosesor.

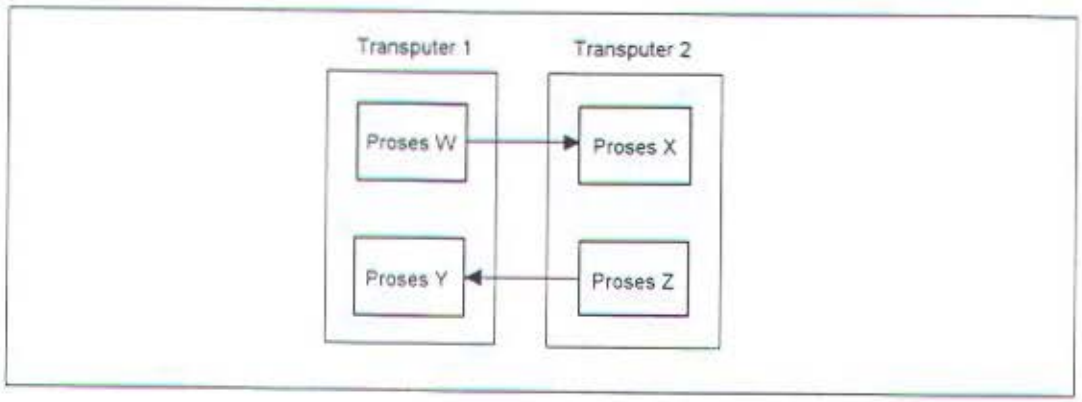
Karena antarmuka memori tidak dipakai bersamaan, dan terpisah dari antarmuka komunikasi, maka memori tersebut dapat dioptimisasi sendiru-sendiri pada setiap transputer untuk menghasilkan kapasitas besar dengan komponen eksternal yang minimum.

3.2.2.3 Komunikasi

Untuk menghasilkan komunikasi yang disinkronisasi, setiap pesan harus di *acknowledge*. Dengan demikian, sebuah *link* membutuhkan paling sedikit satu kawat sinyal untuk setiap arah.

Sebuah *link* antara dua transputer diwujudkan dengan menghubungkan *link* antarmuka pada satu transputer ke *link* antarmuka pada transputer lain dengan dua

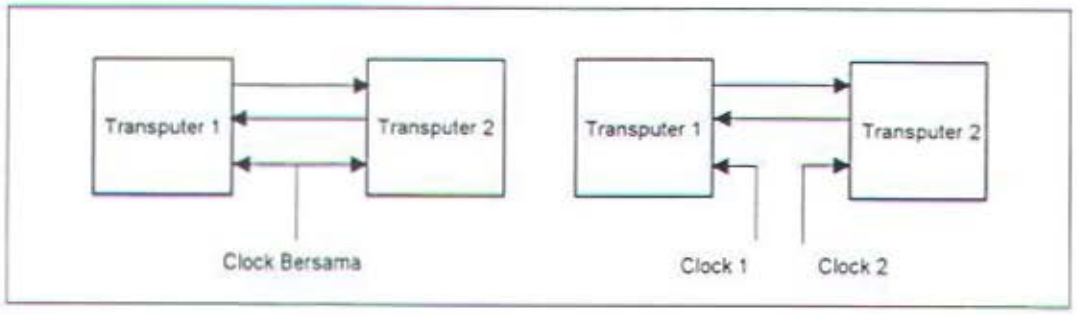
buah kawat sinyal satu arah, karena data dikirimkan secara serial. Kedua kabel sinyal tersebut dapat dipakai sebagai dua kanal Occam, satu untuk setiap arahnya. Cara ini hanya membutuhkan protokol sederhana. Setiap jalur sinyal membawa informasi data dan kendali.



Gambar 3.5 *Link* Komunikasi Titik-ke-titik

Protokol *link* menyediakan komunikasi tersinkronisasi pada Occam. Protokol ini juga mampu mengirimkan urutan byte dengan ukuran word yang berbeda.

Link dirancang untuk memudahkan rekayasa sistem transputer. Tata letak papan cetak untuk dua kawat sangat mudah dibuat dan hemat tempat. Semua transputer mendukung frekuensi komunikasi standard sebesar 10 megabit per detik (Mbps), untuk segala tingkat kinerja prosesor. Dengan demikian transputer dengan kinerja yang berbeda dapat dihubungkan langsung dan saling berkomunikasi.



Gambar 3.6 Model Komunikasi Menggunakan Link

Komunikasi *link* tidak peka terhadap fasa *clock*. Dengan demikian, komunikasi dapat berlangsung antara dua sistem dengan *clock* yang berbeda, selama frekuensi komunikasi yang digunakan sama. Keluarga transputer mempunyai *link* adaptor yang mampu menghubungkan *link* transputer dengan piranti selain transputer.

3.3 Arsitektur Fisik Transputer

3.3.1 *Link* Serial INMOS

Protokol *link* dan karakteristik kelistrikannya distandardisasi untuk seluruh transputer INMOS dan produk periferalnya. Setiap transputer mendukung frekuensi *link* komunikasi standard pada 10 Mbps. Beberapa peralatan juga mampu bekerja pada frekuensi yang lain. Penjagaan frekuensi komunikasi standard akan menyebabkan peralatan dengan kinerja dan jenis yang berbeda dapat saling berkomunikasi.

Setiap *link* terdiri atas dua kawat sinyal satu arah yang membawa bit kendali dan data sekaligus. Sinyal link kompatibel dengan sinyal TTL sehingga

dapat dengan mudah ditambahkan sebuah *buffer*. Keadaan diam (*quiescent*) sinyal link adalah "rendah", "0". Sinyal *link* masukan dan keluaran merupakan sinyal kompatibel TTL standard. *Link* transputer harus mempunyai spesifikasi frekuensi clock dan beban kapasitif yang cocok antara dua transputer yang dihubungkan dengan *link* agar dapat bekerja dengan benar.

3.3.2 Pelayanan Sistem (*System Services*)

3.3.2.1 Pemberian dan Pematian Catu

Tegangan masukan terhadap kaki GND dan VCC harus selalu dalam spesifikasi. Demikian juga pada saat kaki VCC naik 5 V, serta turun dari 5 V menuju 0 V.

Pelayanan sistem memiliki clock, catu daya, dan sinyal untuk inisialisasi. Spesipikasi mencakup juga waktu minimum VCC harus berbeda dalam tegangan yang diizinkan, waktu minimum clock masukan harus beroisilasi, dan waktu minimum sinyal RESET dalam keadaan "tinggi" sebelum menjadi "rendah". Spesifikasi ini menjamin bahwa clock dan rangkaian logika internal telah siap sebelum transputer mulai bekerja.

Prosesor dan *link* serial INMOS mulai bekerja setelah reset. Transputer melaksanakan sebuah program *bootstrap* yang bisa berada dalam ROM atau dilewatkan melalui salah satu link dari penyimpanan sekunder. Hal ini tergantung dari transputer yang dipakai. Program tersebut biasanya akan mengambil program yang lebih besar dari ROM atau dari periferal seperti disk.

Pada saat catu dimatikan (*power down*), seperti halnya pada saat catu dinyalakan (*power up*), tegangan kaki-kaki masukan dan keluaran terhadap VCC dan GND harus dalam spesifikasi.

Kesalahan dari perangkat lunak, seperti *overflow* aritmatik, pelanggaran batas *array*, atau pembagian dengan nol, akan menyebabkan *flag* kesalahan (*error*) di-set pada prosesor transputer. Flag ini langsung dihubungkan dengan kaki ERROR. *Flag* dan kaki tersebut dapat diabaikan, atau operasi transputer dihentikan. Penghentian transputer pada saat kesalahan menyebabkan kesalahan tersebut tidak akan menimbulkan kesalahan-kesalahan berikutnya. Pada saat kesalahan terjadi, status prosesor dan memorinya dapat detentukan.

3.3.2.2 Pendistribusian *Clock*

Setiap transputer bekerja pada *clock* masukan standard 5 MHz. *Clock* kecepatan tinggi diturunkan secara internal untuk menghindarkan kesulitan dalam pendistribusian *clock* frekuensi tinggi. Penerimaan data asinkron pada *link* berarti perbedaan *clock* antar chip tidak dipentingkan.

3.3.3 *Bootstrap* dari ROM atau dari *Link*

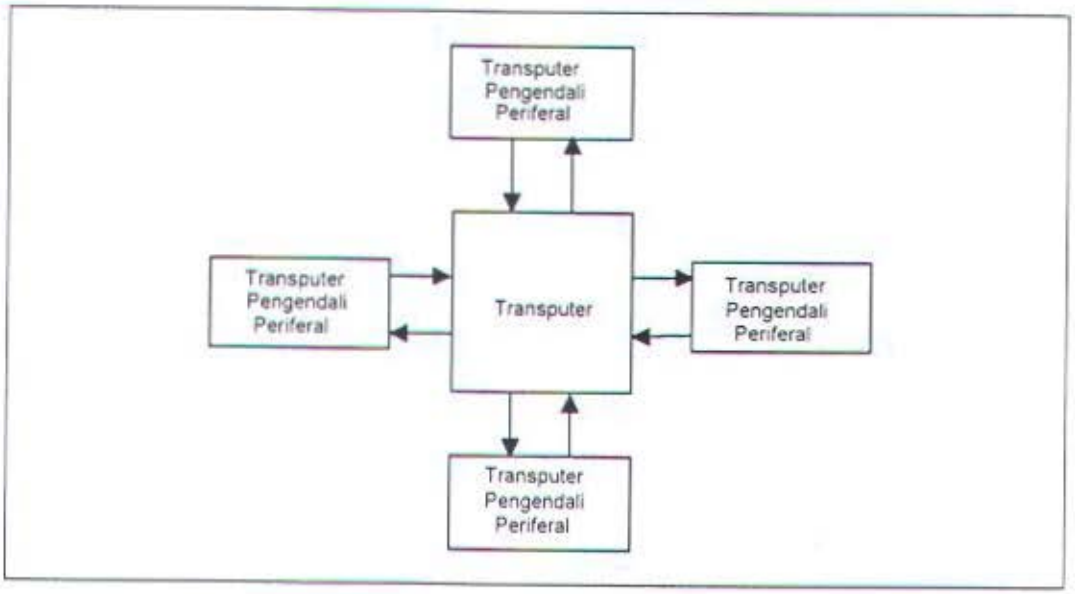
Program yang dijalankan setelah reset dapat berada dalam ROM pada daerah alamat transputer ataul melalui *link* serial INMOS transputer. Transputer mem-*bootstrap* dari ROM dengan memindahlan kontrol dua *byte* teratas memory yang berisi lompatan mundur ke ROM.

Jika proses *bootstrap* melalui *link*, byte pertama dari pesan adalah hitungan jumlah byte program yang akan diberikan. Program dimasukkan ke memory sesuai dengan variabel MemStart, dan kontrol diberikan ke alamat ini.

Pesan-pesan yang datang melalui link yang lain sesudahnya tidak akan diterima sampai transputer selesai melaksanakan proses dari link pertama ini. Pemuatan program pada jaringan transputer dikendalikan oleh sistem pengembangan transputer (*transputer development system*) yang menjamin bahwa pesan yang diterima pertama kali adalah program *bootstrap*.

3.3.4 Hubungan Periferal

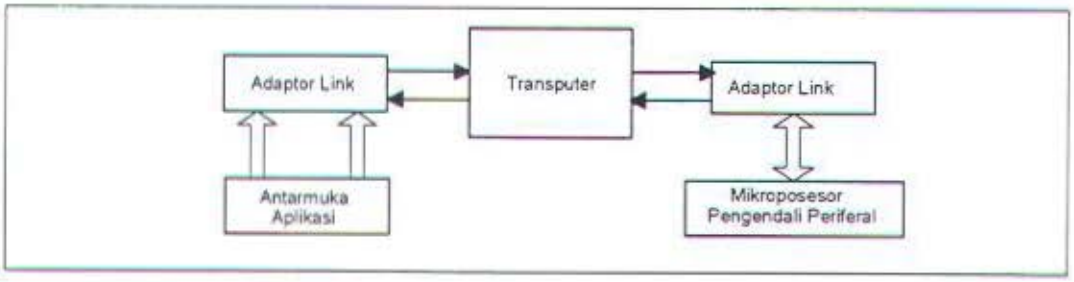
Aplikasi dengan transputer untuk berkomunikasi dengan peroferal atau transputer lain menggunakan *link* serial INMOS, dan transputer tertentu yang mempunyai antarmuka aplikasi khusus. Terdapat tiga metoda untuk berkomunikasi dengan menggunakan periferal.



Gambar 3.7 Transputer dengan Pengendali Periferal

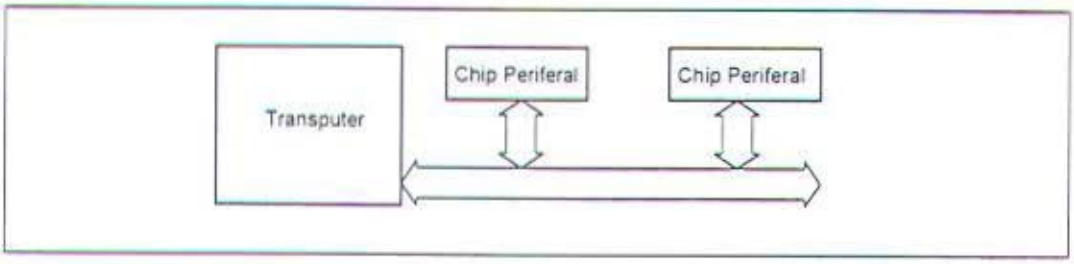
Pertama dengan menggunakan transputer pengendali periferal (seperti untuk grafik dan disk), transputer dihubungkan langsung dengan periferal (Gambar 3.7). *Interface* terhadap periferal diwujudkan dengan perangkat keras khusus di dalam transputer. Perangkat lunak aplikasi diwujudkan sebagai suatu proses Occam, untuk mengendalikan interface yang menghubungkan prosesor dengan perangkat keras tersebut.

Metoda kedua dengan menggunakan adaptor *link* (Gambar 3.8). Adaptor *link* dihubungkan dengan *link* transputer untuk pengendali perferal yang diwujudkan sebagai sebuah proses Occam.



Gambar 3.8 Transputer dengan Adaptor Link

Metoda ketiga adalah dengan memetakan periferal ke alamat memori transputer (Gambar 3.9). Dalam hal ini pengendali periferal harus menyediakan antarmuka kanal Occam. Periferal dikendalikan dengan akses memori yamg digolongkan sebagai hasil masukan dan keluaran jenis PORT.

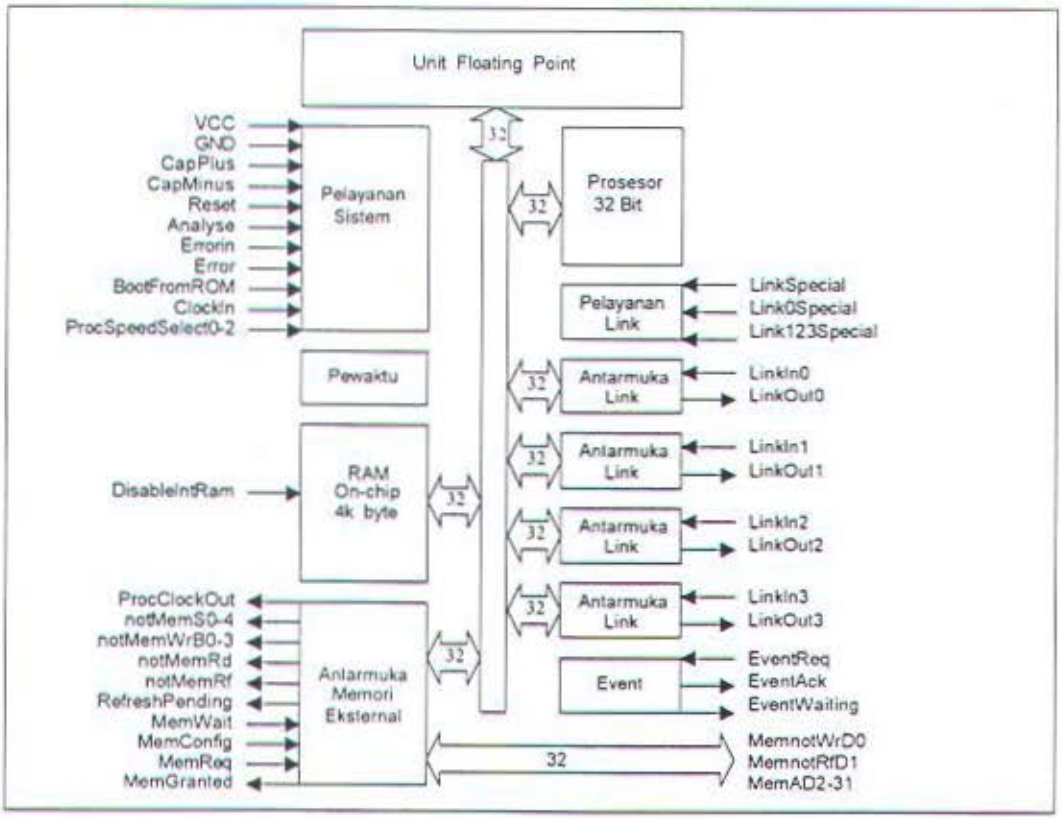


Gambar 3.9 Hubungan Periferal dengan Pemetaan Memori

Pada ketiga metoda tersebut, driver pengendali periferal menghubungkan bagian aplikasi lainnya melalui kanal-kanal Occam. Dengan demikian, sebuah perangkat periferal dapat disimulasikan dengan sebuah proses Occam. Cara ini memungkinkan pemeriksaan seluruh aspek sistem transputer sebelum mengkonstruksi perangkat keras.

3.4 Arsitektur Internal Transputer IMS T805

Secara internal transputer terdiri atas prosesor, memori dan sistem komunikasi, antarmuka memori eksternal yang dihubungkan dengan *bus* 32-bit, yang memungkinkan penambahan memori lokal.



Gambar 3.10 Block Diagram Transputer IMS T805[8]

[8] *Transputer*, Computer System Architechs, INMOS Limited, Utah, 1990

Transputer IMS T805 adalah sebuah mikrokomputer dengan arsitektur 32-bit, dengan sebuah unit *floating point* 64-bit, yang sesuai dengan standard IEEE 754. Transputer mempunyai 4 Kbyte *on-chip* RAM kecepatan tinggi, sebuah interface memori eksternal yang dapat dikonfigurasi sampai 4 Gbyte, dan empat INMOS serial *link* yang dapat diatur kecepatannya pada 5/10/20 Mbit/sec.

Set instruksi lebih efisien jika diimplementasikan dengan bahasa tingkat tinggi yang mendukung secara langsung pemrograman konkuren dengan proses Occam, baik dengan menggunakan satu transputer atau jaringan transputer. Pemanggilan prosedur, proses *switching* dan interupsi, umumnya membutuhkan waktu tidak lebih dari sub-mikrodetik.

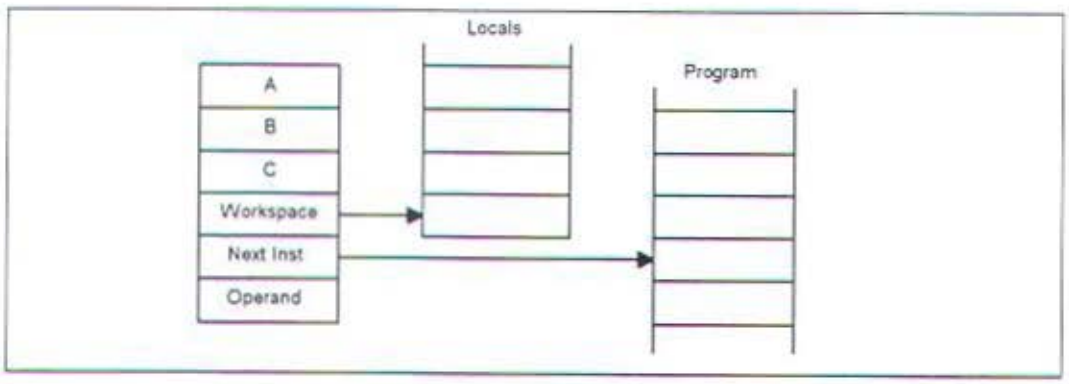
CPU transputer memiliki tiga register (A,B dan C) yang digunakan untuk variabel integer dan alamat aritmatik, yang membentuk *stack* perangkat keras. Memasukkan sebuah nilai kedalam stack akan mem-*push* B ke A, dan A ke B, sebelum mengisi A. Menyimpan nilai dari A akan mem-*pop* B ke A, C ke B. Demikian juga halnya dengan FPU memiliki tiga register *floating point* stack pengevaluasi, terdiri dari register AF, BF dan CF. Pada saat nilai dimasukkan atau disimpan dari stack, register AF, BF dan CF mem-*push* dan mem-*pop* seperti halnya terjadi pada register A, B dan C.

Alamat *floating point* dibentuk pada stack CPU, dan nilai ini dipindah-pindah antara lokasi alamat memori tertentu dengan stack FPU dibawah kendali CPU. Karena stack CPU hanya digunakan untuk menyimpan alamat nilai *floating point*, lebar word CPU independen terhadap lebar word FPU.

Penjadwalan transputer menyediakan dua level prioritas. Register stack CPU diduplikasi sehingga pada saat transputer *floating point* berpindah dari proses prioritas rendah ke prioritas tinggi, tidak ada status di FPU yang disimpan ke memori. Mekanisme ini menghasilkan tanggapan interupsi tiga mikrodetik pada kasus terburuk.

3.4.1 Pemrosesan Sekuensial

Perancangan prosesor transputer mengeksploitasi keberadaan memori cepat *on-chip* dengan memiliki hanya sejumlah kecil register. CPU hanya memiliki enam register yang digunakan dalam proses sekuensial. Jumlah kecil register, bersama-sama dengan kesederhanaan set instruksi memungkinkan prosesor untuk memiliki jalur data dan kendali logik yang relatif sederhana dan cepat.



Gambar 3.11 Register-Register Transputer

Keenam register itu sebagai berikut :

- Register A, B, dan C yang membentuk stack evaluasi.
- Register penunjuk daerah-kerja (*workspace pointer*), yang menunjuk pada sebuah lokasi penyimpanan dimana variabel lokal disimpan.

- Register penunjuk instruksi (instruktion pointer), yang menunjuk pada instruksi selanjutnya untuk dijalankan.
- Register operan (operand register), yang digunakan untuk pembentukan operan instruksi.

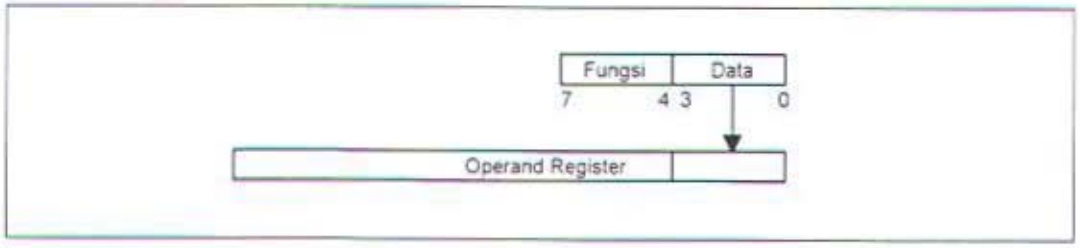
Pernyataan dievaluasi pada stack evaluasi, dan instruksi mengacu pada mekanisme stack. Sebagai contoh instruksi add akan menjumlahkan dua isi teratas stack dan menempatkan hasilnya pada puncak stack. Penggunaan stack menghilangkan keperluan instruksi untuk menyatakan lokasi operan-nya. Statistik yang diperoleh dari sejumlah besar program memperlihatkan bahwa tiga register memberikan keseimbangan yang efektif antara kekompakan kode dan kerumitan implementasi.

Tidak ada mekanisme perangkat keras yang disediakan untuk mendeteksi adanya lebih dari tiga nilai dimuatkan pada stack. Sangat mudah bagi kompilator untuk menjamin bahwa hal ini tidak akan pernah terjadi.

3.4.2 Instruksi

Set instruksi transputer telah dirancang secara sederhana dan efisien untuk kompilasi dalam bahasa tingkat tinggi. Transputer memiliki sejumlah kecil instruksi, semuanya dalam format yang sama, dipilih untuk memberikan representasi yang ringkas untuk operasi yang sering dipakai dalam program. set instruksi independen terhadap lebar *word* prosesor, memungkinkan mikrokode yang sama digunakan untuk transputer yang berbedalebar *word*-nya. Setiap

instruksi terdiri dari satu byte yang dibagi menjadi dua bagian 4-bit. Keempat MSB dari byte ini merupakan kode fungsi, dan empat LSB sisanya merupakan nilai data.



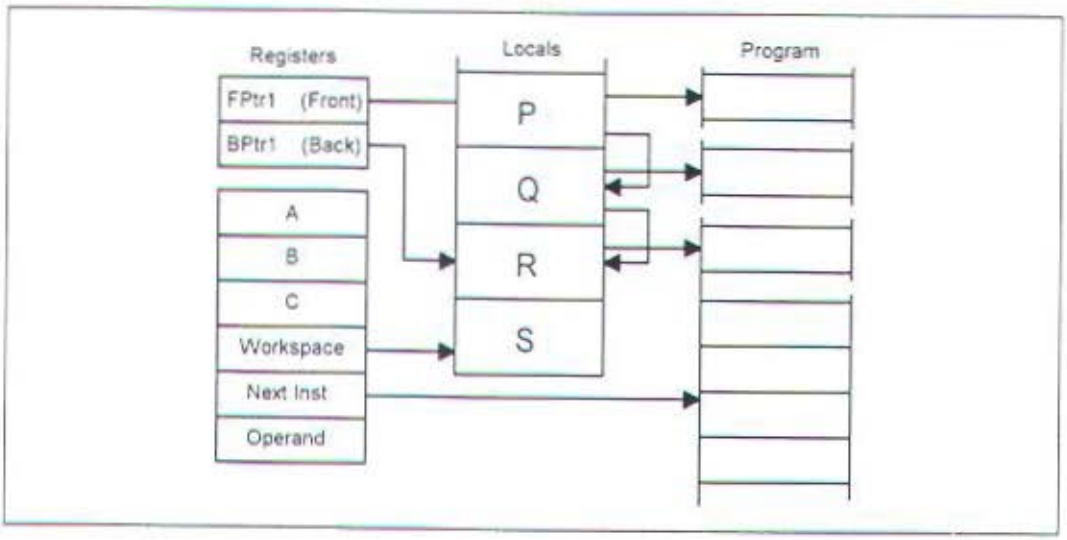
Gambar 3.12 Format Instruksi

3.4.3 Proses-Proses dan Konkurensi

Prosesor menyediakan dukungan yang efisien untuk konkurensi dan komunikasi model Occam. Prosesor memiliki penjadwalan yang dimikrokodekan sehingga memungkinkan sejumlah proses konkuren untuk dieksekusi bersamaan, menggunakan bersama waktu prosesor. Cara ini menghilangkan kebutuhan perangkat lunak *kernel*. Prosesor tidak perlu mendukung alokasi dinamis memori karena kompilator Occam mampu melaksanakan alokasi tempat untuk proses konkuren.

Proses konkuren mempunyai dua keadaan pada saat tertentu mungkin aktif atau tidak aktif. Penjadwalan beroperasi sedemikian agar dapat mengatur waktu tidak aktif dari proses pada saat tertentu tidak banyak. Proses aktif yang menunggu dikerjakan disimpan dalam daftar. Daftar terkait (*linked-list*) daerah-kerja proses ini menggunakan dua register, satu penunjuk pada proses pertama pada daftar, dan kedua menunjuk pada proses terakhir. Dalam Gambar

3.12, proses S sedang dikerjakan, proses P, Q dan R aktif dan menunggu dikerjakan.



Gambar 3.13 *Linked-list* Proses

Prosesor menyediakan operasi khusus yang mendukung model proses, di antaranya start process, dan end process. Instruksi start process akan menciptakan proses konkuren dengan menambahkan daerah kerja baru pada akhir daftar penjadwalan, maka proses konkuren yang baru akan dikerjakan bersama-sama dengan proses yang telah dikerjakan. Pengakiran konstruksi paralel yang benar, dijamin oleh instruksi end process, sampai semua komponen terakhir dikerjakan pencacah akan menjadi nol.

3.4.4 Komunikasi Antar Proses

Komunikasi antar proses dilaksanakan melalui kanal. Komunikasi Occam adalah titik-ke-titik, sinkron, dan tanpa penyangga. Hasilnya sebuah kanal tidak

memerlukan antrian proses, antrian pesan dan penyangga pesan. Sebuah kanal antara dua proses dalam transputer diimplementasi dengan *word* dalam memori, sedangkan kanal antar proses yang dikerjakan dalam transputer yang berbeda diwujudkan dengan *link* titik-ke-titik. Prosesor menyediakan sejumlah operasi pelaluan pesan, diantaranya yang paling penting adalah input message dan output message.

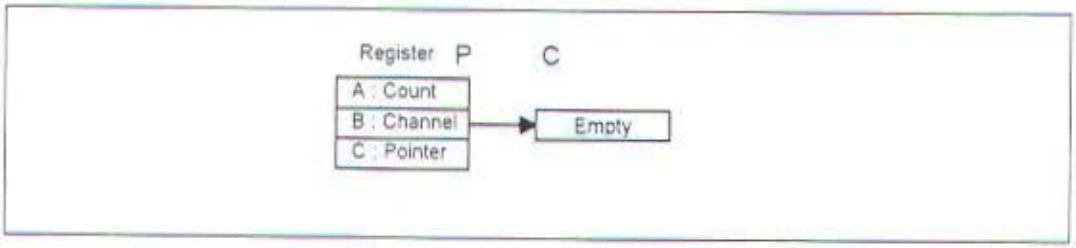
Kedua instruksi tersebut menggunakan alamat kanal untuk menentukan apakah tersebut kanal internal atau eksternal. Hal ini berarti bahwa urutan instruksi yang sama dapat digunakan untuk kanal-keras (*link*) dan kanal-lunak (memori), dan memungkinkan suatu proses untuk ditulis dan dikompilasi tanpa perlu diketahui kemana kanal tersebut dihubungkan.

Seperti halnya dalam model Occam, komunikasi terjadi pada saat kedua proses yang memasukkan dan mengeluarkan, proses pertama kali siap akan menunggu proses pasangannya siap. Proses-proses tersebut memuat stack evaluasi dengan penunjuk ke pesan, alamat sebuah kanal, dan mencacah jumlah byte yang akan dipindahkan. Kemudian menjalankan instruksi *input message* dan *output message*.

3.4.4.1 Kanal Komunikasi Internal

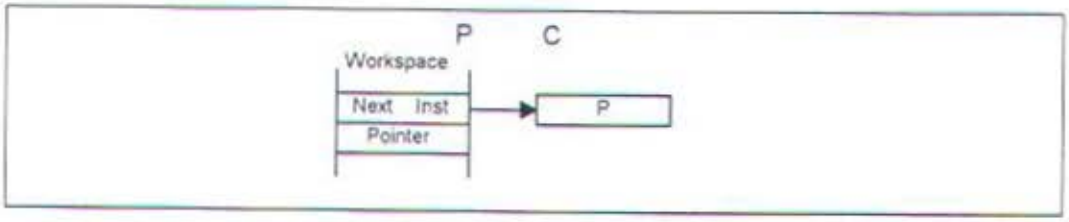
Setiap saat, kanal internal (sebuah *word* di memori) mungkin menyimpan identitas proses atau nilai khusus empty. Semua kanal diinisialisasi empty sebelum digunakan.

Pada saat sebuah pesan dilalukan melalui sebuah kanal, identitas proses pertama disimpan dalam kanal dan prosesor mulai mengerjakan proses selanjutnya dari daftar penjadwalan. Ketika proses kedua yang akan menggunakan kanal siap, pesan disalin, dan proses yang menunggu ditambahkan ke daftar penjadwalan (*scheduling list*) dan kanal diinisialisasi ke keadaan awal. Tidak jadi masalah apakah proses masukan atau keluaran.



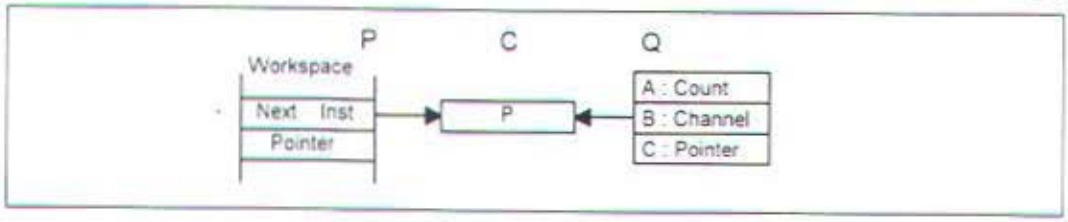
Gambar 3.14 Proses Konunikasi Tahap Pertama
Pengeluaran ke Kanal Empty

Pada Gambar 3.14 proses P akan mengerjakan instruksi pengeluaran pada sebuah kanal 'kosong' (empty) C. Stack evaluasi menyimpan penunjuk pesan, alamat kanal C, dan pencacah jumlah byte dalam pesan.



Gambar 3.15 Proses Komunikasi Tahap Kedua

Setelah mengerjakan instruksi pengeluaran, kanal C menyimpan alamat daerah-kerja P, dan alamat pesan yang akan dipindahkan disimpan dalam daerah kerja P. P dilepas-jadwalkan, dan proses mulai mengerjakan peoses selanjutnya dari daftar penjadwalan.



Gambar 3.16 Proses Komunikasi Tahap Ketiga

Kanal C dan proses P tetap dalam keadaan ini sampai proses kedua, Q mengerjakan instruksi pengeluaran pada kanal. Pesan disalin, proses yang menunggu P ditambahkan ke daftar penjadwalan, dan kanal C direset pada keadaan awal empty.

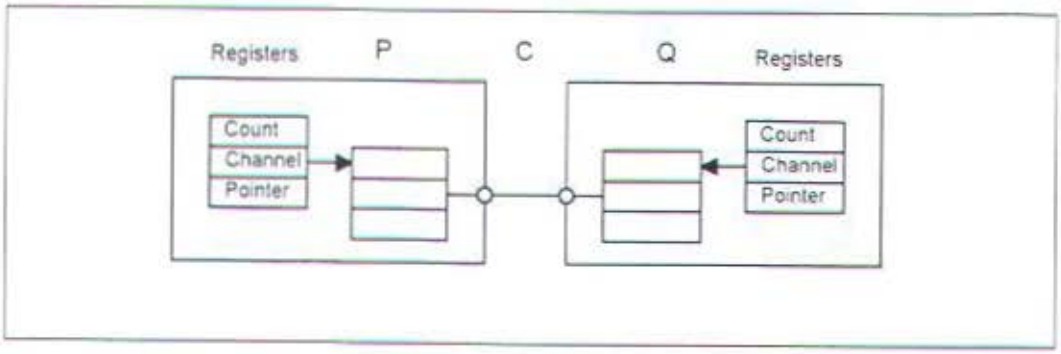
3.4.4.2 Kanal Komunikasi Eksternal

Ketika sebuah pesan dilewatkan melalui sebuah kanal eksternal, prosesor memberikan pekerjaan pemindahan pesan pada antarmuka link dan *deschedule* proses. Ketika pesan telah dipindahkan antarmuka link menyebabkan prosesor me-*reschedule* proses yang menunggu. Prosesor dapat melanjutkan pengekskusian proses yang lain sementara transfer pesan eksternal berlangsung.

Tiap antarmuka link memiliki tiga register yaitu :

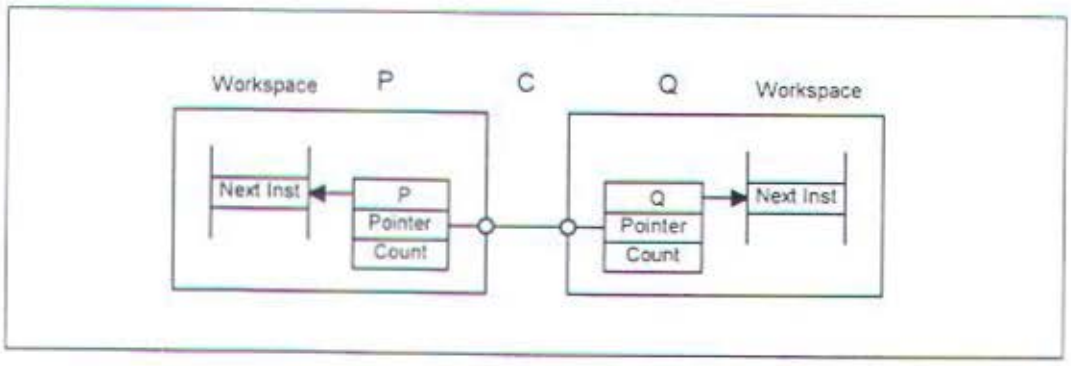
- ☐ penunjuk pada daerah-kerja proses
- ☐ sebuah penunjuk pada pesan
- ☐ dan sebuah pencacah byte pesan.

Gambar 3.17 menunjukkan Proses P dan Q dikerjakan oleh transputer yang berbeda yang berkomunikasi dengan kanal C. Implimentasi proses dengan link menghubungkan kedua transputer.



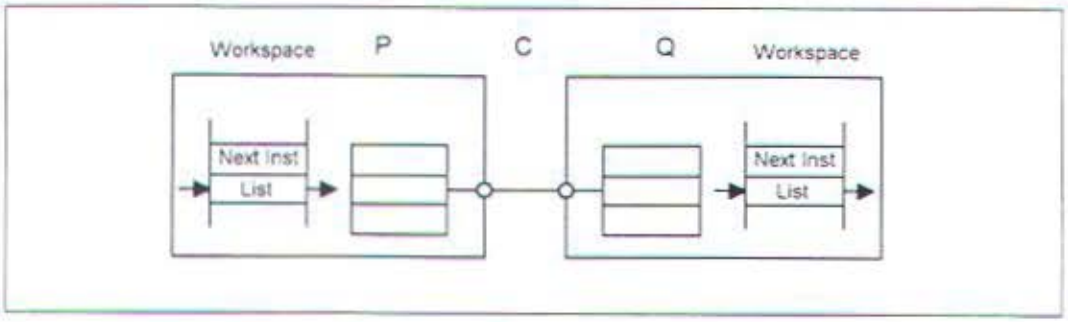
Gambar 3.17 Komunikasi Antar Transputer Tahap Pertama

Pada saat P mengerjakan instruksi pengeluaran, register-register antarmuka link dari transputer yang mengeksekusi P diinisialisasi, dan P di-*deschedule*. Demikian juga halnya, ketika Q mengerjakan instruksi pemasukan, register-register pada antarmuka link dari proses mengeksekusi Q diinisialisasi, dan Q dilepas-jadwalkan (Gambar 3.18).



Gambar 3.18 Komunikasi Antar Transputer Tahap Kedua

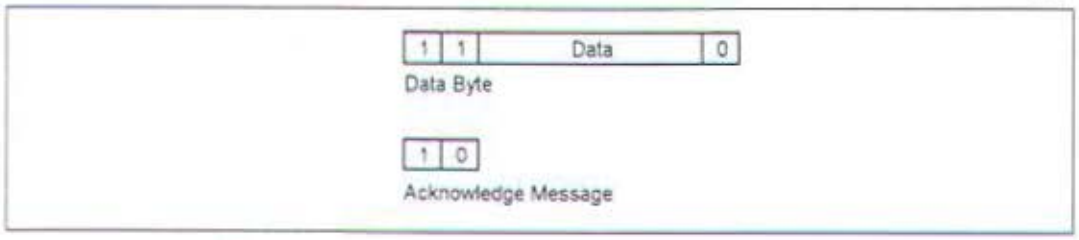
Pesan sekarang disalin ke *link*, setelah tiap daerah-kerja dari P dan Q dikembalikan pada daftar penjadwalan yang bersangkutan (Gambar 3.19). Protokol yang digunakan pada P dan Q menjamin bahwa tidak menjadi soal proses yang mana diantara P dan Q yang lebih dulu siap.



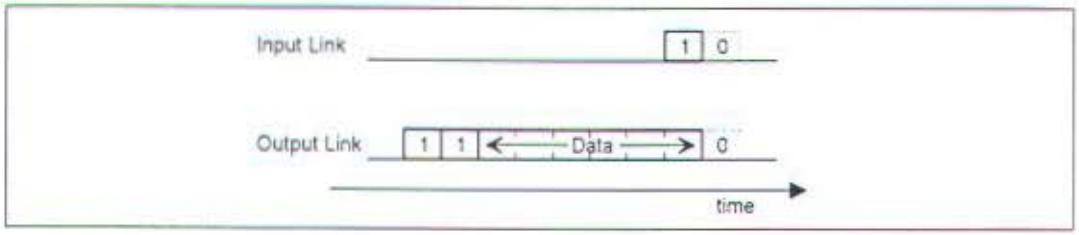
Gambar 3.19 Komunikasi Antar Transputer Tahap Ketiga

3.4.4.3 Link Komunikasi

Sebuah *link* antara pada transputer diwujudkan dengan menghubungkan antarmuka *link* pada satu transputer dengan antarmuka *link* transputer lain dengan menggunakan dua buah kawat sinyal satu arah, dimana data dipindahkan secara serial. Dua kawat tersebut mewakilkan dua kanal Occam, satu untuk setiap arahnya. Cara ini hanya memerlukan protokol yang sederhana untuk memultipleks informasi data dan kendali. Setiap pesan dikirimkan sebagai sebuah urutan komunikasi byte tunggal, sehingga hanya diperlukan *buffer* berukuran satu byte tunggal pada trasnputer penerima, untuk memastikan tidak adanya data yang hilang. Setiap byte dikirimkan dengan sebuah bit start "1", diikuti sebuah bit "1", delapan bit data itu sendiri, dan sebuah bit stop "0". Setelah mengirimkan sebuah bit data, pengirim menunggu sampai tanda terima (*acknowledgement*) deterima, yang berupa sebuah bit start "1" diikuti dengan bit "0". Tanda terima ini menandakan bahwa sebuah proses telah menerima byte tersebut, dan link penerima telah siap menerima byte berikutnya. Link pengirim hanya akan mengulangi proses pengiriman setelah tanda *acknowledge* untuk byte terakhir pesan diterima.



Gambar 3.20 Data *Link* dan Format *Acknowledge*



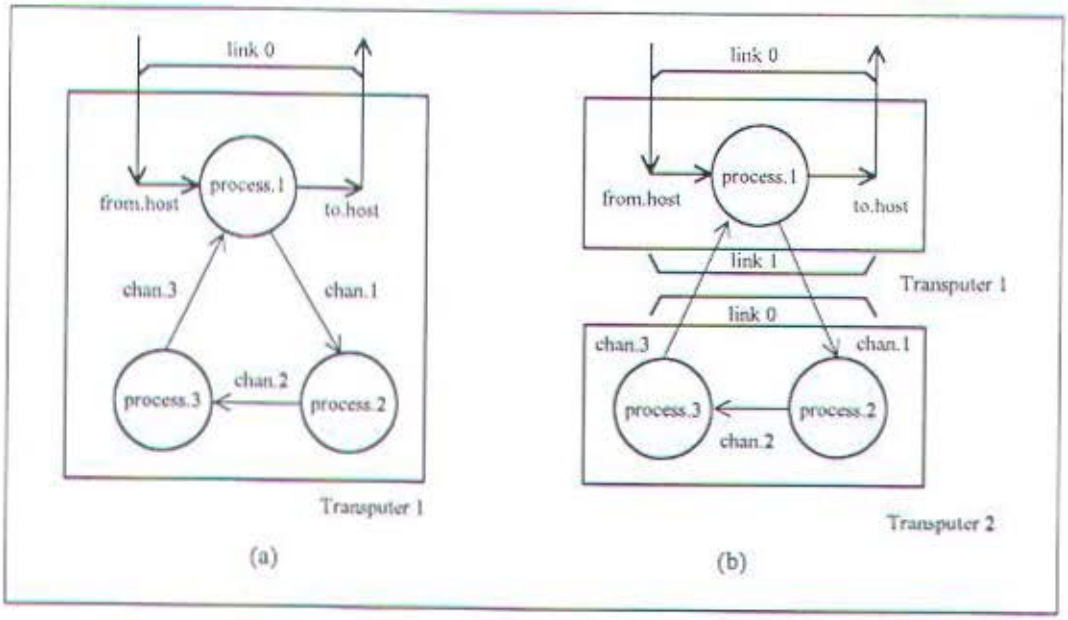
Gambar 3.21 Overlapped Link Acknowledge

Byte-byte data dan *acknowledge* dimultipleks pada setiap jalur sinyal. Sebuah *acknowledge* dapat dikirimkan sesegera setelah penerimaan sebuah byte data (jika terdapat tempat untuk mem-*buffer* yang lain). Dengan demikian transmisi dapat berlangsung secara kontinyu, tanpa penundaan antar byte data.

Protokol membolehkan sebuah tanda terima untuk dibangkitkan sesegera setelah penerima mengidentifikasi paket data. Dengan cara ini tanda terima dapat diterima oleh pengirim sebelum semua data paket selesai dikirimkan dan pengirim dapat mengirim paket data berikutnya dengan segera. Beberapa transputer tidak menerapkan tumpang tindih (*overlapping*) ini dan hanya mencapai kecepatan transmisi data 0,8 Mbyte/detik dengan menggunakan sebuah link untuk memindahkan data dalam satu arah. Gambar 3.21 memperlihatkan sinyal yang harus dibangkitkan pada kedua kawat *link* jika sebuah paket data ditumpang tindih dengan tanda terima.

3.5 Komunikasi Antar Proses

Pemrograman paralel pada Transputer (*concurrent process*), didukung melalui pelaluan pesan (*message-passing*) dan *inter-process* komunikasi. Sehingga hampir tidak ada perbedaan dalam pemrograman untuk Transputer tunggal atau jaringan Transputer. Transfer data antar proses dilakukan melalui *channel* . Perbedaannya hanya pada media kanal komunikasi (*channel*) apakah melalui kanal lunak (*memory*) untuk Transputer tunggal atau kanal keras (*link*) untuk jaringan Transputer.



Gambar 3.22 Tiga Proses Paralel (a) Pada Satu Transputer Dan (b) Pada Dua Transputer

Suatu modul proses konkuren yang telah dirancang untuk satu Transputer dapat dijalankan pada Transputer yang berbeda hanya dengan inisialisasi kanal komunikasi yang berbeda. Sedangkan kode programnya tidak perlu diubah sama sekali, seperti diperlihatkan pada Gambar 3.22.

BAB IV
IMPLEMENTASI JARINGAN SYARAF TIRUAN BERBASIS TRANSPUTER T805 PADA SIMULASI SISTEM DRIVE-RIG

Sistem komputer telah banyak digunakan sebagai pengendali cerdas, responsif dan waktu nyata (*real-time*), misalnya untuk pengaturan gerakan robot, sistem rem anti-slip, reaktor nuklir, pembimbing arah peluru, dan akhir-akhir ini dikembangkan *neurokomputer* yang mengimplimentasikan mekanisme jaringan syaraf tiruan sehingga menghasilkan mesin yang mampu belajar.

Penelitian ilmiah dan teknologi komputer hampir tidak dapat dipisahkan. Meskipun kinerja komputer sekarang sudah demikian menakjubkan, ilmu pengetahuan masih saja kehausan dan terus menuntut kecepatan yang lebih tinggi. Masalah-masalah ilmiah sudah sedemikian rumitnya sehingga superkomputer yang memiliki kemampuan 1,5 milyar hitungan per detik pun masih juga kewalahan.

Salah satu batu sandungan penggunaan arsitektur komputasi jaringan syaraf tiruan secara meluas dan besar-besaran adalah masalah kenerja, yang terutama berkaitan dengan kebutuhan merepresentasikan banyak interkoneksi antara satuan-satuan syaraf. Implementasi JST dalam komputer sekuensial telah dibatasi oleh ketidakmampuan untuk mengerjakan lebih dari sebuah tugas dalam satu waktu. Ringkasnya, JST beroperasi secara paralel baik dalam lingkungan biologis maupun dalam abstraksi, yaitu model matematikanya. Oleh karena itu, masalah

implementasi JST dalam perangakat keras dan perangkat lunak juga semestinya dilakukan dengan pendekatan paralel.

Alternatif lain yang lebih realistis adalah pemrosesan paralel. Sebagai alih-alih teknologi proses dalam meningkatkan kecepatan, permasalahan diselesaikan dengan menggunakan beberapa prosesor paralel yang dirangkai bersama dalam suatu sistem.

Komputer yang digunakan untuk jaringan syaraf tiruan (JST), pengendalian mesin jet, pendiagnosa medis dan penjadwalan di bandar udara semuanya menggambarkan dunia nyata. Untuk memodelkan secara teliti reaksi nuklir, percobaan ekstraksi obat-obatan, pengembang kantung udara pada kecelakaan kendaraan, simulasi pengaruh total dari sejumlah gaya, hanya akan benar-benar mendekati kelakuan sistem sebenarnya jika dilakukan dengan pendekatan paralel, sesuai dengan kelakuan alami sistem-sistem tersebut, yang memberikan hasil secara waktu-nyata.

Masalah-masalah seperti tersebut di atas merupakan masalah pemrosesan paralel yang semestinya dipecahkan secara paralel juga. Salah satu prosesor yang telah dirancang khusus untuk pemrosesan paralel adalah ***transputer***, dibuat oleh INMOS dan dilengkapi dengan bahasa pemrograman paralel alaminya, yaitu *Occam*.

Trasnputer merupakan mesin paralel baru yang dapat derealisasikan sistem komputasi kelas PC. Sistem yang dibangun dengan beberapa transputer meningkatkan kecepatan prosesnya dengan mendistribusikan bagian-bagian

permasalahan pada tiap transputer untuk diselesaikan secara konkuren, sebagaimana bagian-bagian ini terjadi dan memerlukan penyelesaian semestinya.

Berdasarkan hal inilah, dengan berbagai keterbatasan yang ada, penulis mencoba melakukan sesuatu yang baru, meskipun sangat sederhana, untuk mengimplementasikan JST secara paralel dengan transputer. Diharapkan, dengan mendistribusikan proses komputasi pada transputer semirip mungkin dengan proses sebenarnya pada JST (dan tentunya dengan mempertimbangkan faktor efisiensi dan kemudahan implementasi) kasus seperti disebutkan pada alinea di atas dapat diperbaiki.

Dasar pemikirannya adalah sebagai berikut :

- dua karakteristik khusus transputer , yaitu fasilitas perhitungan *full on-chip* dan *link* komunikasi titik-ke-titik.
- sifat alami pemrosesan paralel dengan transputer, yaitu peningkatan kinerja hampir linear, dan kemampuan mengelolah masukan dari banyak sumber yang tak-tentu waktu.
- arsitektur dasar transputer, yang memungkinkan penggabungan ko-prosesor tujuan-khusus (misalnya *ASIC*).

Paling tidak terdapat dua keuntungan penggunaan multiprosesor dalam menangani masalah komputasi seperti ini :

- kecepatan (kinerja), membagi-bagi pekerjaan diantara prosesor seringkali mempercepat waktu penyelesaian tugas secara radikal, yang merupakan keuntungan dari pemrosesan paralel.

Keuntungan kedua adalah dengan pendekatan paralel kita dapat memiliki sejumlah besar piranti masukan/keluaran (I/O) yang berkomunikasi secara paralel, asinkron dan tak-tentu waktu dengan jaringan prosesor, piranti ini dapat berupa ADC (*analog to digital converter*), mesin DSP (*digital signal processing*), dan piranti lainnya.

Keuntungan khusus penggunaan transputer untuk aplikasi jaringan syaraf tiruan diantaranya :

- ❑ pemrosesan paralel, dengan memori-prosesor yang independent, bebas bottle-neck
- ❑ link kecepatan tinggi untuk komunikasi antar prosesor
- ❑ modularitas diantara arsitektur modul transputer (TRAM, *transputer module*)
- ❑ kemampuan sensitivitas waktu-nyata dan multiple I/O melalui link
- ❑ kemudahan perluasan sistem dengan sedikit atau tanpa perubahan sama sekali pada kode program
- ❑ kemudahan pembangunan fault tolerance secara fisik

Pemrograman paralel tidak sesederhana pemrograman sekuensial konvensional, karena kita sebagai perancang sistem bertangguang jawab atas pendistribusian proses yang akan diparalelisasi. Oleh karena itu, pada tugas akhir ini jumlah sel dibuat tetap, tidak dapat deset secara interaktif pada saat program dijalankan seperti halnya dalam program sekuensial. Untuk mengubah jumlah ini, kita harus mengubah kode programnya, kemudian dikompilasi ulang.

Arsitektur Jaringan Syaraf Tiruan yang akan dibahas pada tugas akhir ini diperlihatkan pada BAB II. Dari gambar ini terlihat bahwa lapisan masukan menerima masukan dari sejumlah sensor atau reseptor yang tersusun paralel. Proses sebenarnya yang terjadi dalam setiap sel adalah sederhana dan bukan tipe perhitungan yang menguntungkan bila didistribusikan untuk tiap prosesor secara paralel. Bagaimanapun juga, jumlah koneksi yang besar antara sel-sel dalam dua lapis yang berturutan (2^n untuk koneksi terhubung penuh) menyiratkan bahwa penyelesaian paralel sangat imperatif untuk memperoleh kinerja yang memuaskan. Secara umum, sebagian besar perhitungan adalah berupa operasi jumlahan-perkalian dan fungsi aktifasi.

Arsitektur komputasional juga harus mudah dan efisien dalam melakukan komunikasi, (tidak menimbulkan overhead, apalagi bottleneck), sebab hasil perkalian vektor, misalnya error kali bobot, harus didistribusikan ke tiap sel dalam lapis yang sama, sebelumnya, dan sesudahnya. Transputer sangat cocok untuk tujuan ini, karena memiliki mekanisme manajemen proses paralel On-chip dalam *workspace* dan penggunaan antrian berganda untuk tipe proses yang berbeda.

Hal yang penting lainnya adalah pembagian bagian kerja tiap prosesor juga merupakan masalah yang cukup menentukan efisiensi dan kinerja sistem.

Dalam tugas akhir ini akan dibahas mengenai aplikasi jaringan syaraf tiruan dengan metoda *Multilayer Perceptron (MLP)* dengan metoda belajar back-error propagation, yang berfungsi sebagai *neurocontrol* untuk aplikasi pada sistem pengaturan Drive-Rig.

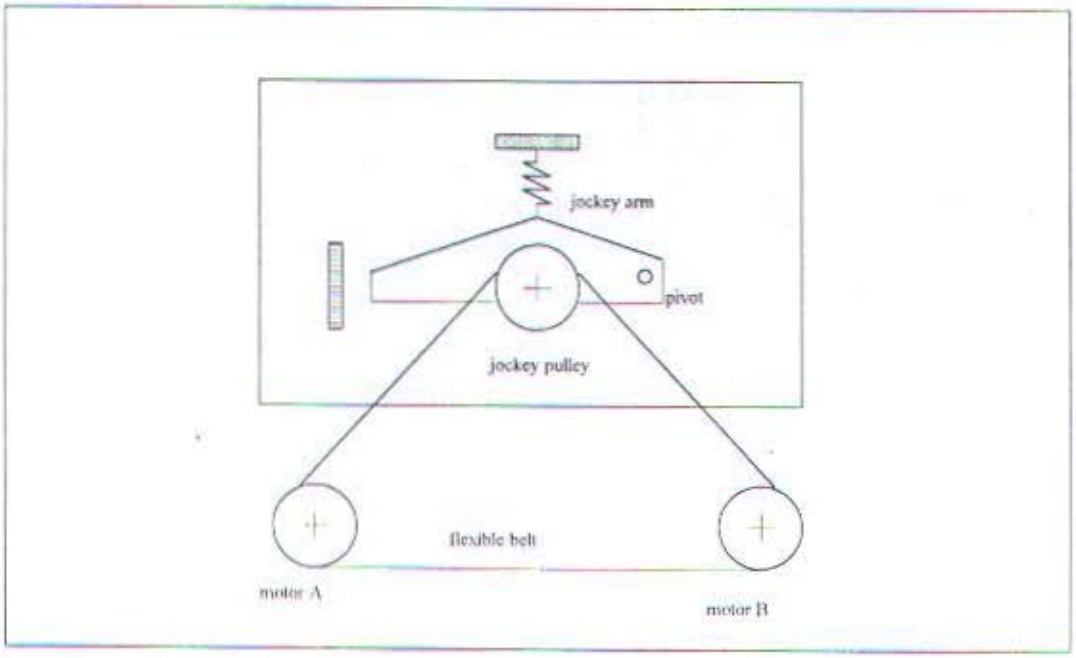
4.1 Tujuan Perancangan

Tujuan umum penelitian tugas akhir ini adalah merealisasikan sistem pengaturan multivariable berbasis jaringan syaraf tiruan pada aplikasi pengaturan electric drive-rig system, dengan metoda parallel processing menngunakan Transputer T805. Tujuan penelitian ini diharapkan memiliki nilai rekayasa pada ilmu pengetahuan, dilain pihak di laboratorium riset dan pengembangan sedang dengan intensifnya dilakukan aplikasi parallel processing dengan Transputer seperti jaringan syaraf tiruan, yang secara alaminya memerlukan penyelesaian paralel.

4.2 ELECTRIC DRIVE-RIG SYSTEMS

Sistem Drive-Rig dengan real-time coupled electric merupakan peralatan pemindahan material (*material handling and transport*) yang banyak digunakan di industri, sistem ini mempunyai multivariable kontrol karena permasalahan nya adalah mengontrol daya tegang dan kecepatan dalam material handling dan transport.

Peralatan coupled electric terdiri atas dua motor servo dc yang menjalankan *jockey-pulley* melalui suatu belt kontinyu. Pengukuran kecepatan belt diukur melalui suatu tacho-generator yang dipasang pada simpul jockey pulley, sedangkan gaya tegang dari belt dimonitor secara langsung melalui defleksi anguler dari lengan katrol dengan menggunakan servo-potensiometer. Parameter kecepatan belt dan gaya tegang dibutuhkan untuk perhitungan sinyal kontrol.



Gambar 4.1 Blok Diagram Skematis Electric
Drive-Rig System

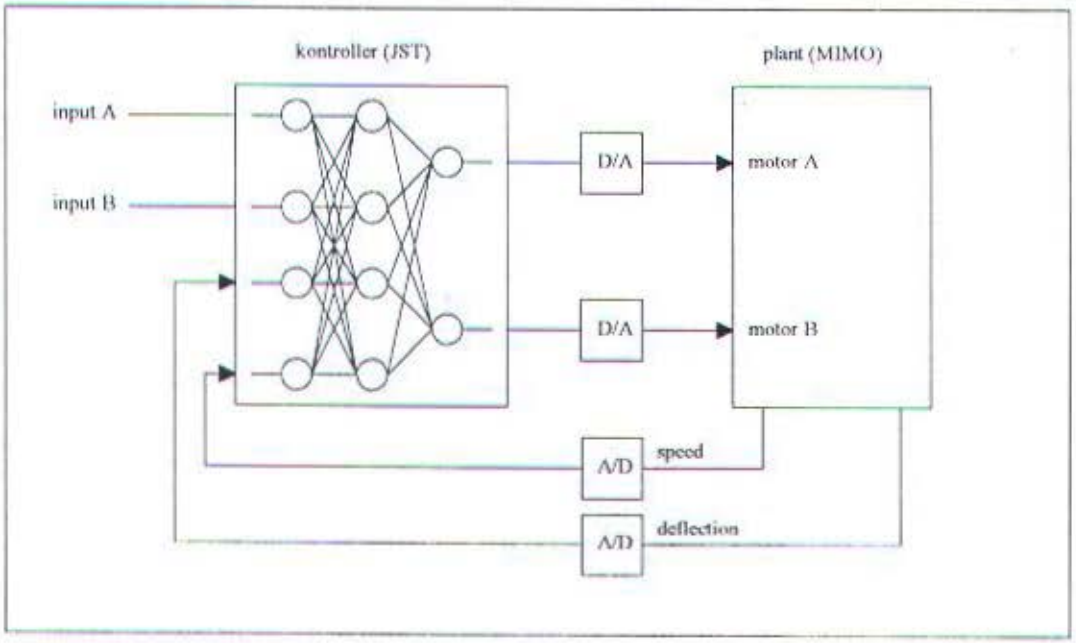
Sistem kontrol Drive-Rig ini digolongkan kedalam kontrol MIMO (multi-input multi-output), dimana inputnya adalah tegangan untuk motor A dan motor B, sedangkan output dari sistem tersebut adalah mengatur kecepatan belt dan gaya tegang dari belt tersebut

4.2.1 Model Sistem Pengaturan Drive-Rig

Sistem pengaturan untuk sistem dinamis harus dapat menjalankan algoritma kontrolnya lebih cepat dari waktu sampling, maka diperlukan parallel processing untuk mempercepat jalanya proses kontrol. Untuk menjalankan algorithma kontrol secara multitasking, prosedur pertama adalah mengambil contoh output dari plant melalui ADC kemudian menghitung sinyal kontrol dengan algorithma kontrol JST,

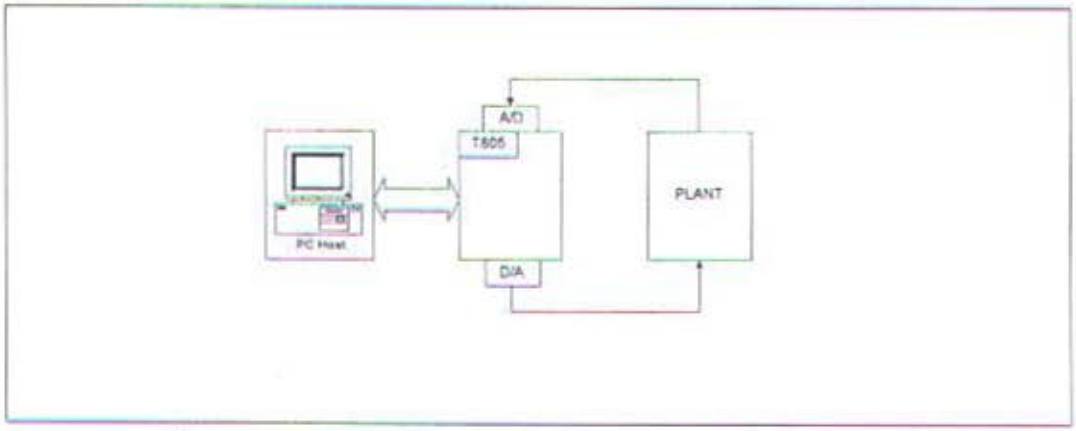
menulis status kontrol ke layar dan membaca dari keyboard tugas operator, dan akhirnya menuliskan sinyal kontrol proses melalui DAC, itu semua harus dilaksanakan dalam batas waktu sampling.

Penggunaan transputer untuk melakukan proses paralel untuk menjalankan algoritma kontrol, terdiri dari proses-proses yang berjalan secara paralel. Proses pertama menjalankan konverter A/D dan konverter D/A, kemudian proses kedua terdiri dari proses menghitung algorithma kontroller jaringan syaraf tiruan.

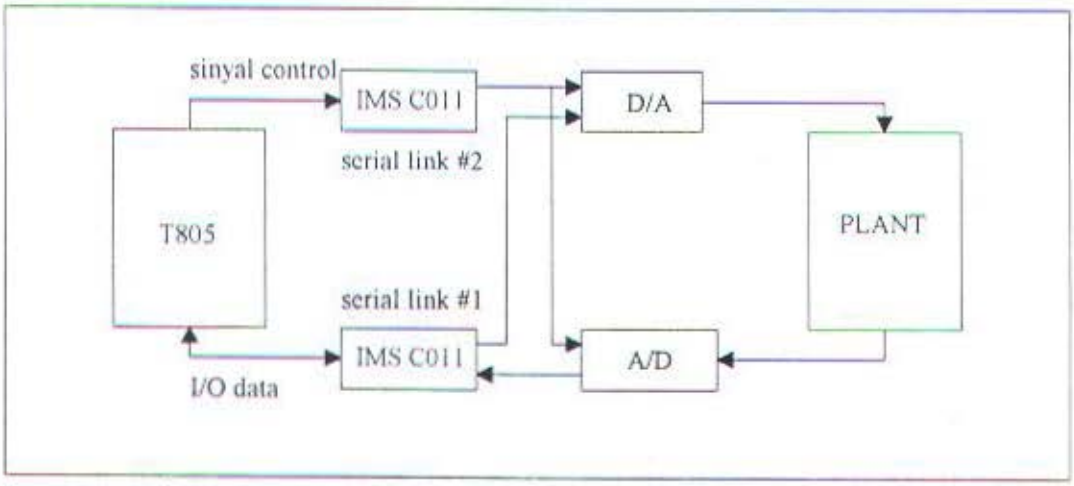


Gambar 4.2 Block Diagram Sistem Pengaturan Drive-Rig

Untuk mendukung proses konkuren tersebut diatas masih ditambah dengan proses keyboard handler dan screen handler untuk menampilkan hasil dan untuk berkomunikasi dengan host komputer dan untuk pembacaan dan penyimpanan file bobot.



Gambar 4.3 Arsitertur Sistem Pengaturan dengan Transputer



Gambar 4.4 I/0 Hand Shaking dengan Serial Link IMS C011

4.2.2 Implementasi Paralel JST

Implementasi JST pada Transputer tunggal atau jaringan Transputer, perlu suatu analisa terhadap interkoneksi sel-sel dalam JST, aliran informasi yang berguna untuk dapat menentukan metode paralelisasi dan komunikasi.

Sel-sel JST melaksanakan operasi perkalian vektor dan fungsi aktifasi yang memerlukan prosesor yang cepat. Walaupun demikian operasi tersebut bukanlah

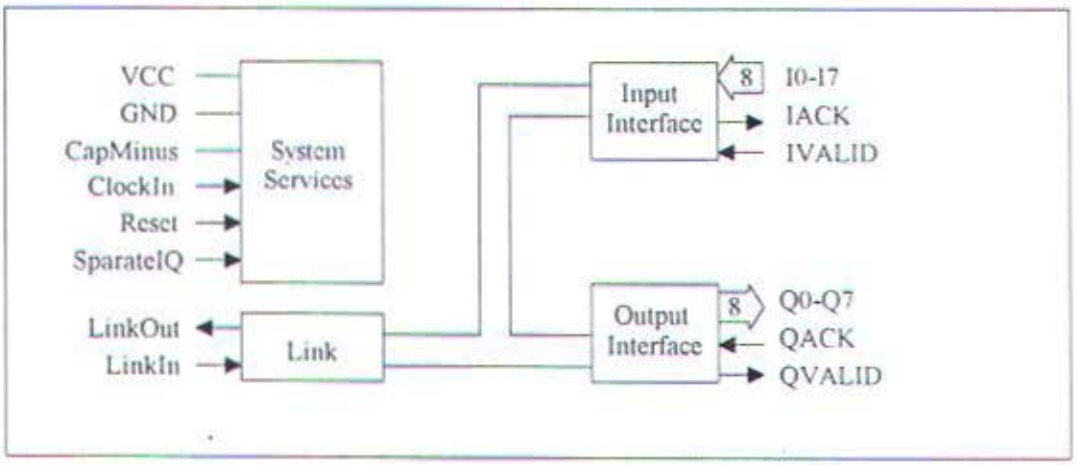
operasi yang rumit, dengan memetakan setiap sel pada satu prosesor bukanlah suatu solusi yang baik, karena dengan semakin banyak jumlah sel akan semakin banyak jumlah prosesor dan semakin rumit lalulintas komunikasi.

Disamping itu ada hal lain yang perlu diperhatikan yaitu operasi waktu sinkronisasi komunikasi lebih lama dibandingkan dengan waktu untuk melakukan perhitungan. Ditambah lagi perhitungan setiap sel pada lapisan yang berbeda dilakukan lapis demi lapis atau secara sekuensial. Pada fase feedforward perhitungan masukan suatu sel pada lapisan tertentu akan menunggu keluaran dari sel sebelumnya. Demikian juga pada fase backfoeward, perhitungan error pada layer tertentu, memerlukan perhitunganerror layer sesudahnya. Salah satu metoda paralelisasi yang dapat dipilih adalah metoda *pipelining* untuk layer yang sama.

4.3 SERIAL LINK ADAPTOR INMOS C011 MODE 1

Adaptor Link adalah sebuah chip untuk menghubungkan link Transputer dengan bus komponen periferal. Ini dapat dipandang sebagai paralel ke serial link konverter. Serial link adapter tersedia dalam dua type untuk keguanaan yang berbeda yaitu IMS C011 dengan kemasan 28-pin dan IMS C012 dengan 24-pin.

Komponen tersebut sangat simpel untuk interface input/output 8-bit seperti keyboard dan komponen I/O analog seperti ADC dan DAC. IMS C011 mode-1 mempunyai I/O 8-bit data paralel secara terpisah. Dalam mode ini adaptor link mengkonversi antara serial link dengan dua interface 8-bit dengan *handshake*, satu untuk input dan satu untuk output.



Gambar 4.5 Blok Diagram Adaptor Link Mode-1[8]

Salah satu keunggulan dari transputer adalah tidak diperlukannya banyak komponen pendukung untuk dapat berkomunikasi dan memori interface. **INMOS** telah membuat 2 model dari komponen pendukung, adaptor link digunakan untuk menghubungkan transputer dengan sistem orientasi-bus konvensional dan *swicth link*, yang dapat memungkinkan link konfigurasi elektronik dengan transputer.

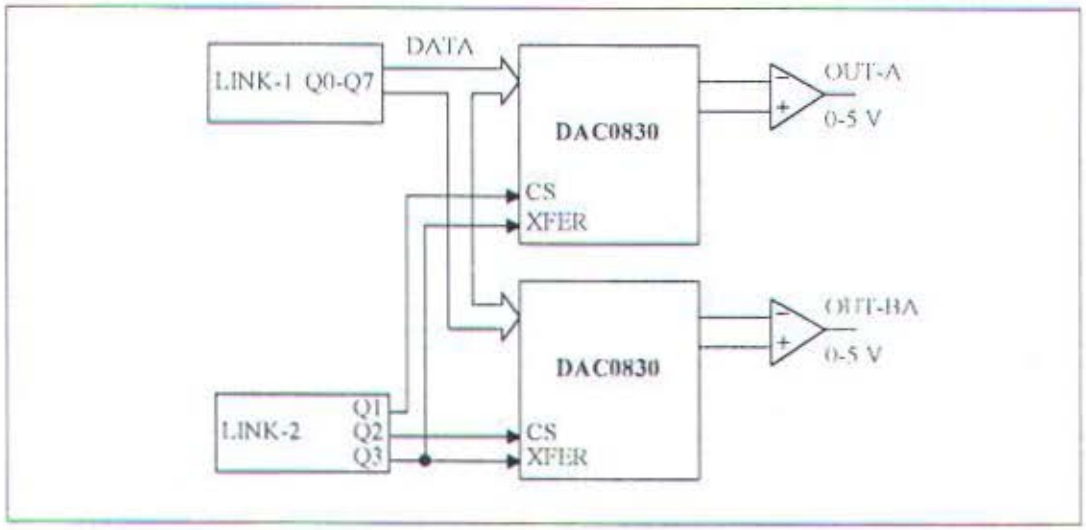
Dalam mode ini C011 dikonfigurasi sebagai peripheral interface dengan port input terpisah dari port output. Pin IValid dan IAck adalah pin handshake untuk memasukkan data dari peripheral ke transputer. Ketika data pada pin I0-I7 valid, IValid diberi sinyal '1' oleh komponen peripheral. Link kemudian akan mengirim data yang ada pada pin-pin I0-I7, dan ketika transfer data selesai maka *IAck* akan berlogika '1'. Ketika komponen peripheral mengirim sinyal '0' pada *IValid*, *IAck* akan menjadi '0' oleh C011. Sebuah byte data yang diterima oleh C011 akan diteruskan pada port output Q0-Q7, dan QValid akan berlogika '1' oleh C011.

8 _____,Transputer Serial Link Interfacing Kit IMS C011 Link Adaptor Engineering Data, Computer System Architects (CSA), USA, 1990, hal. 390

Setelah data dibaca oleh peripheral maka peripheral harus mengirim sinyal '1' pada pin 1*QAck*. C011 kemudian akan mengirim sebuah paker *acknowledgment* pada link untuk memberi sinyal bahwa transfer data sudah selesai. C011 akan mengeset *QValid* '0', dan peripheral dapat mengirim sinyal '0' pada QAck lagi.

4.4 Dua Kanal Konverter Digital ke Analog (DAC)

Rangkaian DAC mempunyai dua kanal output analog, DAC0830 telah dilengkapi dengan buffer dan latch internal. DAC tersebut mempunyai tipe 8-bit TTL/CMOS compatible. Dalam tugas akhir ini dibuat dua kanal konverter digital ke analog dengan output bervariasi dari 0 sampai 5 volt.



Gambar 4.6 Block Diagram Fungsional Dua Kanal DAC0830

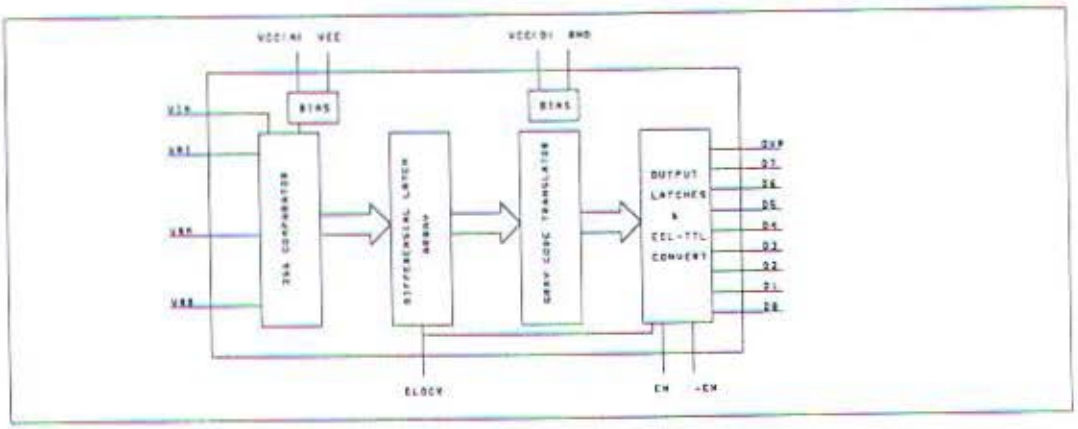
Analog Devices DAC0830 kompatibel dengan bus 8-bit mikroprosesor yang telah dilengkapi dengan buffer dan latch internal, mempunyai waktu konversi 1 μs, untuk interface dengan transputer digunakan serial link C011 mode-1.

Diperlukan dua langkah untuk menuliskan harga baru pada DAC, pertama CS(chip select) digunakan untuk menempatkan data byte pada internal latch, kemudian langkah kedua XFER diberikan high untuk menuliskan data byte pada latch internal ke register DAC output yang dipilih.

Untuk menuliskan data pada kedua DAC dimulai dengan memberikan sinyal kontrol melalui serial link2 (link kontrol), yaitu Q1 low untuk CS (chip select) DAC 1 atau Q2 untuk CS DAC 2, pada fase ini data yang ada pada serial link1(link I/O data) Q0-Q7 akan ditulis pada internal latch DAC, dan Q3 sinyal XFER diberikan untuk menetuskan (update) register output DAC.

4.5 Konverter Analog Ke Digital (ADC)

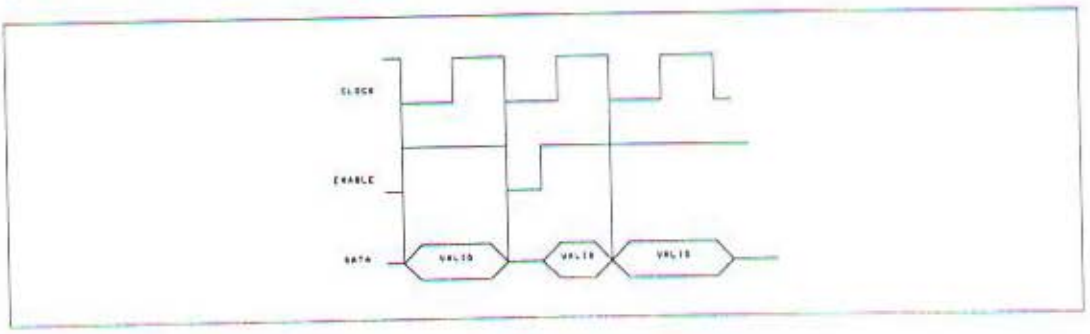
ADC digunakan untuk mengkonversi sinyal analog menjadi sinyal digital. Sinyal yang berupa data digital inilah yang dapat diproses oleh komputer. ADC yang digunakan disini adalah MC13019 dari Motorola. ADC tersebut merupakan flash ADC dengan spesifikasi sampling rate 25 MHz, resolusi 8 bit, konversi satu clock cycle, dengan output 3 state LS TTL.



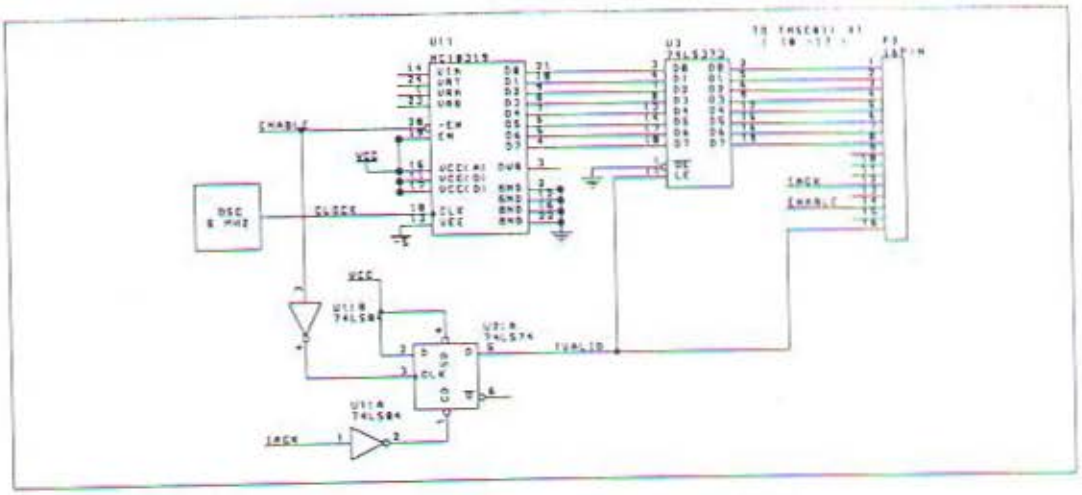
Gambar 4.7 Blok diagram MC10319

Dua buah In Out enable, pin 19 dan 20, kompatibel dengan TTL dan digunakan untuk merubah data output (D7-D0) dari kondisi aktif ke kondisi 3 state. Pada perencanaan pin 19 diberi input start untuk memulai pengukuran dan pin 20 diberi logika "0" sehingga output dalam keadaan aktif.

Input clock (pin 18) kompatibel dengan TTL, mempunyai range frekuensi 0 sampai 30 MHz tanpa batasan duty cycle. Pada perencanaan ini input clock ADC dihubungkan ke clock generator 24 MHz. ADC MC10319 mempunyai dua tegangan referensi, yaitu VRT untuk referensi positif dan VRB untuk referensi negatif. MC10319 dapat mengkonversi sinyal analog adalah 2 volt. Pada perencanaan ini dibuat range pengukuran dari -1 volt sampai 1 volt.



Gambar 4.8 Timing Diagram MC10319



Gambar 4.9 Diagram Fungsional ADC

4.6 Perencanaan Software

Tujuan tahap perancangan dengan mengimplementasikan algoritma kontroller Jaringan Syaraf Tiruan dapat dikelompokkan dalam dua bagian yaitu perencanaan software di komputer host dan pemrograman di transputer. Software sistem secara keseluruhan diharapkan dapat bekerja secara paralel dan mempunyai kinerja yang optimum.

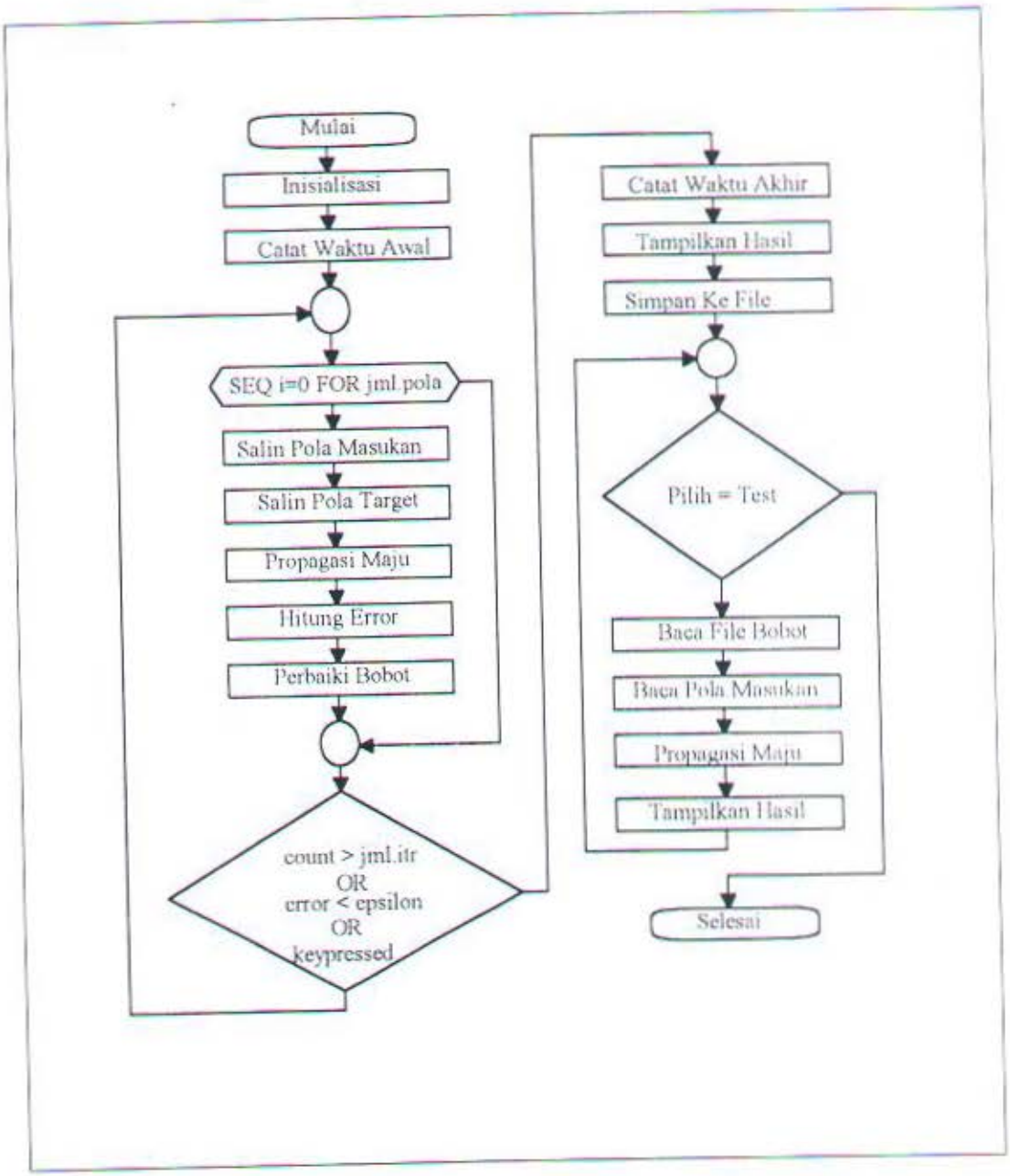
Tahap awal dalam perencanaan software yang dapat bekerja secara konkuren dengan baik adalah optimasi komunikasi terutama antara sofware yang dijalankan pada komputer *host* dan Transputer. Hal ini karena kedua pemrograman menggunakan *compiler* yang berbeda, maka hal yang paling penting yaitu kita harus memastikan sinkronisasi komunikasi data dan tidak terjadinya *deadlock*.

Software yang dijalankan pada *host* mempunyai tugas menangani operasi input/output sistem diantaranya operasi file, menampilkan hasil, membaca perintah dari keyboard, dan menangani driver komunikasi dengan transputer.

4.6.1 Algoritma dan Desain Proses

Program yang di-*download* pada Transputer merupakan program utama sistem pengaturan Jaringan Syaraf Tiruan, proses pembacaan ADC dan penulisan algorintma kontroller melalui DAC. Program tersebut dibagi menjadi dua proses utama yang berkomunikasi melalui *hard-channel*, yaitu proses kontroler Jaringan Syaraf Tiruan dan proses mengakses periferal (ADC dan DAC), sebagai tracking

kontroler. Program tersebut adalah proses-proses yang bekerja secara konkuren melalui kanal. Proses untuk menjalankan algoritma JST bekerja secara paralel dengan mengakses periferal, proses proses JST berkomunikasi melalui *link0* dengan *host* komputer, dan komunikasi dengan periferal melalui *link1* dan *link2*.



Gambar 4.8 Diagram Alir Implementasi JST

Algoritma yang digunakan dalam program ini diantaranya adalah membaca parameter JST diantaranya pola masukan, konstanta belajar, jumlah iterasi maksimum. Program terus melakukan iterasi sampai salah satu dari ketiga hal tercapai yaitu jumlah iterasi melebihi iterasi maksimum, error total kurang dari epsilon (error yang dapat diterima), dan ada sembarang tombol yang ditekan.

Setelah proses diatas kemudian program akan merekam data ke suatu file, kemudian dilanjutkan ke fase pemakaian. Sebagai catatan representasi data bobot, keluaran sel, data pola masukan, error dan perubahan bobot dinyatakan dengan variabel array bertipe float.

4.6.2 Parallel Processing Pada Transputer

Program pada transputer menggunakan C 89.1 dari Logical System yang telah dilengkapi fasilitas pemrograman paralel. Dalam pemrograman JST perhitungan antar sel-sel pada layer yang sama dilakukan secara paralel dengan metoda *pipelining* dan *bypassing*.

Pengalokasian dan inisialisasi *channel* dilaksankan dengan instruksi ***ChanAlloc()*** dan ***ChanReset()***. Khusus untuk empat Link transputer hardware dapat berfungsi sebagai *channel bidirectional pointer* dengan spesifikasi alamat hardware yang berada pada *library*, yaitu :

```
#define LINK0OUT ((channel *) 0x80000000)
#define LINK1OUT ((channel *) 0x80000004)
#define LINK2OUT ((channel *) 0x80000008)
#define LINK3OUT ((channel *) 0x8000000c)
```

Instruksi-instruksi untuk mengakses *channel* adalah sebagai berikut:

ChanOut(), instruksi pendefinisian *channel* untuk pengiriman keluar proses dengan tipe data bebas.

ChanIn(), instruksi pendefinisian *channel* untuk penerimaan ke dalam proses dengan tipe data bebas.

Salah satu hal yang sangat penting, apabila diinginkan suatu eksekusi proses yang cepat, dapat digunakan internal memory, maka eksekusi kode akan jauh lebih cepat. Penggunaannya dapat dilaksanakan dengan instruksi ***ProcCall()***.

Proses 1 adalah kontroler JST dioptimasikan dengan merancang operasi tiap sel pada layer yang sama harus bekerja secara paralel dengan menggunakan *soft-channel* (memory). Metoda paralelisasi yang digunakan adalah *pipelining* dan *bypassing*. Komunikasi dengan *host* melalui *link 0* untuk menerima bobot interkoneksi antar layer, kemudian melakukan perhitungan setiap sel pada layer yang sama secara konkuren, kemudian setiap kali iterasi mengirim data ke *host*.

Proses 2 yaitu membaca data ADC dan menuliskan sinyal kontrol dengan DAC dimana komunikasi dengan proses 1 yaitu melalui *link 1* untuk data input dan output, dan melalui link 2 untuk sinyal kontrol periferal.

BAB V
PENGUJIAN SISTEM DAN ANALISA

Pada bab ini akan dibahas pengujian dari sistem pengaturan yang direncanakan dan dibuat pada tugas akhir ini. Uji coba dilakukan untuk mengetahui unjuk kerja dan fungsional dari sistem yang dibuat. Pengujian meliputi pengujian tiap modul peralatan pengujian unjuk kerja sistem pengaturan drive-rig menggunakan kontroler berbasis jaringan syaraf tiruan secara parallel processing yang dijalankan pada Transputer T805.

5.1 UJICOBA FUNGSIONAL

Sebelum ujicoba dilakukan, pengkalibrasian MC 10319 sebagai unit digital, kehandalan perangkat lunak yang dibuat terutama sistem jaringan saraf tiruan dan penentuan hasil sistem jaringan saraf tiruan dalam tingkat belajar ataupun pengujian. Penguat transduser terlebih dahulu dikalibrasi. Uji coba dilakukan dengan cara memberi tegangan DC tertentu kepada ADC dan kemudian data hasil konversi ADC dibaca.

Kalaikan unjuk kerja modul ini adalah kemampuan untuk mengkonversi tegangan analog menjadi data digital yang dipengaruhi beberapa hal antara lain linieritas, offset null, tegangan referensi, dan kesalahan konversi. Tegangan referensi untuk ADC adalah +1 volt dan -1 volt.

Prosedur Program untuk pembacaan ADC yaitu, data dibaca melalui serial link 1 IMS C011 dan untuk memberikan sinyal kontrol digunakan link 2. Untuk sinyal enable (mulai konversi) dilakukan secara sofftware. Prosedur program untuk pembacaan adc yaitu seperti berikut ini.

Software yang dijalankan pada *host* mempunyai tugas menangani operasi input/output sistem diantaranya operasi file, menampilkan hasil, membaca perintah dari keyboard, dan menangani driver komunikasi dengan transputer.

Rutin proces untuk akses periferal (DAC dan ADC) yaitu :

```
#define to_link_data            LINK1OUT
#define from_link_data          LINK1IN
#define to_link_control         LINK2OUT
#define from_link_control       LINK2IN

void dac()
  {
      ChanOut (to_link_data, (char*)&data_dac, 1);
      ChanOut (to_link_control, (char*)&addr_dac, 1);
      ChanOut (to_link_control, (char*)&xfer, 1);
  }

void adc()
  {
      ChanOut (to_link_control, (char*)&adc, 1);
      ChanOut (to_link_control, (char*)&enable, 1);
      ChanIn (from_link_data, (char*)&data_adc, 1);
      return data_adc;
  }
```

Pengujian DAC0830 dengan 2 kanal analog yaitu tegangan keluar DAC dari 0 samapi 5 volt, tegangan ini diumpankan ke input driver motor untuk mendapatkan penguatan arus yang besar.

Pengujian sensor yaitu terdiri dari pengukuran output tacho-generator untuk mengetahui besarnya kecepatan belt yang diubah kedalam bentuk tegangan.

Dari data tacho-generator adalah 2000 rpm untuk tegangan 2 volt maka untuk kecepatan yang diinginkan adalah 50 rpm akan menghasilkan tegangan output 50 mV.

Output besarnya defleksi berdasarkan konversi dari servo-potensiometer adalah servo-potensiometer adalah 10 kΩ dengan jangkauan 360 derajad. Untuk 1 derajad defleksi servo-potensiometer maka akan mengakibatkan perbahan tahanan sebesar 27.778 Ω.

Rasio perobahan resistansi terhadap perubahan sudut digunakan untuk mengetahui efek ketengan belt. Untuk memdapatkan defleksi pada 1 derajad, servo-potensiometer didet pada posisi 0, kemudian deberi tegangan bipolar pada kedua ujung terminal potensiometer yaitu 12 volt, maka tegangan output defleksi 1 derajat adalah 33.33 mV.

5.2 UJI PERANGKAT LUNAK JST

Jaringan yang dibuat mempuyai spesifikasi dan parameter sebagai berikut mempunyaui tiga layer, layer input mempunyai 6 sel, layer hidden mempunyai 6 sel dan layer output mempunyai 2 output masing masing untuk mengatur putaran motor 1 dan motor 2.

Pengaruh konstanta belajar bersama sama dengan matrik pembobot mula sangat mempengaruhi pengujian struktural implementasi Jaringan Syaraf Tiruan untuk aplikasi plant dinamis kontroler electric drive-rig. Dalam ujicoba sistem konstanta belajar mendekati harga 1.0 akan memerlukan iterasi unruk mencapai

error toleransi semakin kecil. Biasanya pembobot mula suatu sistem JST dilakukan dengan bilangan random.

Dalam ujicoba sistem secara perangkat lunak adalah meliputi beberapa asper yaitu menguji komunikasi antar proses secara konkuten. Link 0 merupakan kanal komunikasi antara proses pada komputer host dengan program JST yang di down-load ke transputer (proses 1).

Pada proses 1 juga terjadi komunikasi dengan proses 2 untuk mendapatkan informasi plant (output yang sebenarnya), yaitu data ADC, dan setelah melakukan proses komputasi JST akan memberikan tracking kontrol melalui kanal ke proses 2 untuk menuliskan ke DAC.

Komunikasi antara proses 1 dengan proses 2 adalah melalui serial link adaptor IMS C011 mode-1. Untuk sinyal kontrol digunakan link 2 dan untuk I/O data digunakan link 1.

5.3 PENGUJIAN SISTEM

Dalam pengujian sistem secara keseluruhan, yang diuji adalah respon plant terhadap sinyal kontrol yang diberikan yaitu kecepatan putaran belt dan defleksi lengan katrol sistem Drive-Rig. Dengan melakukan training pada kontroller JST sehingga diperoleh kecepatan 50 rpm dan defleksi tidak melebihi 1 derajad.

Bobot interkoneksi yang diperoleh selama training untuk mendapatkan error yang minimal, dan respon kontroller yang cepat diperoleh selah melakukan iterasi lebih dari 300 kali, dan bobot ini disimpan sebagai matrik bobot jaringan.

BAB VI
PENUTUP

Dalam *parallel processing*, beberapa bagian pekerjaan (dalam konteks transputer dinamakan *proses*) dikerjakan secara konkuren. Proses-proses ini umumnya tidak independen sehingga diperlukan mekanisme sinkronisasi komunikasi yang tepat agar tidak terjadi *dead-lock*.

Transputer dapat membangun suatu sistem komputasi dengan pendekatan paralel, yaitu beberapa proses/komputasi dilakukan secara konkuren, dengan metode dari rancangan logic sampai menuju implementasi fisik yang dapat dimodularisasi, yaitu penentuan topologi jaringan.

Dalam mengimplementasikan jaringan syaraf tiruan pada sistem dinamik, dengan proses konkuren (dengan memetakan operasi sel neuron secara paralel), metode komunikasi yang sesuai, akan menghasihkan kinerja kontroller yang lebih baik karena dapat informasi plant dapat diperoleh mendekati sempurna.

Transputer dapat berkomunikasi antar Transputer atau untuk komunikasi dengan periferal melalui *link*, proses untuk melakukan *tracking* kontrol dan informasi keadaan plant, bekerja secara paralel dengan proses komputasi Jaringan Syaraf Tiruan.

Jaringan Syaraf Tiruan dapat digunakan pada kontroler plant dinamik, dan berubah terhadap waktu, karena mempunyai sifat adaptif terhadap kesalahan dan

dapat beradaptasi untuk menyesuaikan bobot-bobot interkoneksi antar sel sehingga didapat hasil yang diinginkan.

Tugas akhir ini merupakan titik awal yang berkenaan dengan masalah parallel processing, sehingga masih banyak aspek-aspek yang belum dapat diungkapkan, sehingga perlu penelitian yang berkelanjutan. Diantara masalah yang dihadapi diantaranya optimasi komunikasi antar proses terutama komunikasi dengan periferal melalui serial link IMS C011 mode-1.

Kurangnya literatur yang ada sehingga tidak dapat menggali potensial transputer yang ada diantaranya, kemampuan grafik kecepatan tinggi, dan rancangan arsitektur konkuren yang lebih baik.

Transputer merupakan sarana yang sangat potensial untuk mengaplikasikan kontrol pada proses plant dinamik, real-time dan berubah terhadap waktu, juga dengan mendistribusikan proses-proses pada beberapa transputer akan menghasilkan kinerja sistem yang lebih baik.

DAFTAR PUSTAKA

1. A.E. Gore, **Workbook-Transputer Education Kit**, Computer System Architects (CSA), USA, 1990
2. Cavalieri, S., ***On the Design of A Multiprocessor Architechture for Neural Network Simulation***, IEEE Proceedings on Circuit and Systems, 1991
3. Freeman, J.A., dan Skapura, D.M., ***Neural Networks Algorithms, Applications, and Programming Techniques***, Addison-Wesley Publishing Company, Massachussetts, 1991
4. Fujimoto, Y., ***Massively Paraller Architechture for Large Scale Neural Network Simulations***, IEEE Trans. on Neural Network, vol.3, no. 6, November 1992
5. Murre, Jacob M.J., ***Transputers and Neural Networks : An Analysis of Implementation Constrain and Performance,*** IEEE Trans. Neural Networks, Vol 4. No.2, pp.284-292, 1993
6. Narendra, K.S. dan Parathasarathy, K. ***Identification and Control of Dynamical Systems Using Neural Networks,*** IEEE Trans. Neural Networks, 1, pp.4-27, 1990
7. Psaltis, D., Sideris, A. and Yamamura, A., ***Neural controllers,*** IEEE First International Conference on Neural Networks, Vol. 4, pp.551-8, 1987
8. Rogers, Eric dan Yun Li, ***Parallel Processing In A Control Systems Environment***, Prentice Hall International (UK) Ltd, Cambridge, 1993
9. Saerens, M. dan Soquet, A., ***Neural Controller Based On Back Propagation Algorithm***, IEE Proceedings-F, vol. 138, no. 1, pp. 55-62, February 1991
10. Yuh, J., A, ***Multilayered Neural Net Controller Using Direct Learning Algorithm***, Computere Elect. Engng., vol. 19, no. 4, hal : 255-264, 1993
11. _____, ***C 89.1 Logical Systems***, Computer System Architechs, USA, 1990
12. _____, ***Transputer***, Computer System Architechs, Utah, USA, 1990
13. _____, ***Transputer Technical Notes***, INMOS Limited - Prntice Hall, New York, 1989
14. _____, **IMSC011 Link Adaptor - Engineering Data**, INMOS Limited, USA, 1988

RIWAYAT HIDUP

AZHAR dilahirkan di Sigli, Aceh pada tanggal 30 Agustus 1965. Putra pertama dari Bapak M. Ali Mahmud dan Ibu Halimah S.

Terdaftar sebagai mahasiswa Jurusan Teknik ELektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya, dengan nomor regristasi 2922202000.

Selama menjadi mahasiswa aktif sebagai asisten praktikum rangkaian Listrik, Elektronika dan Prantikum Elektronika Lanjutan II di Lingkungan Bidang Studi Elektronika.

Riwayat Pendidikan :

- Institut Teknologi Sepuluh Nopember, Jurusan Teknik Elektro, Bidang Studi Elektronika (1992-1996)
- Politeknik ITB Bandung, Jurusan Teknik Elektro (1985-1988)
- SMA Negeri Sigli (1980-1983)
- SMP Negeri Sigli (1977-1980)
- SD Negeri Batee, Kab. Pidie (1971-1977)

Riwayat Pekerjaan :

- Instruktur Politeknik Unsyiah Lhokseumawe Aceh-Utara, Jurusan Teknik Elektro (1989-1992)