SIMULASI ALGORITMA REROUTING DAN PROSEDUR CONTENTION CONTROLLER PADA SISTEM PENYAMBUNGAN ATM

Makalah Seminar Tugas Akhir disusun oleh :
Yudhitya Sorrenti NIM. L2F000652
Mahasiswa Jurusan Teknik Elektro Universitas Diponegoro Semarang
Email :yudhityasorrenti@yahoo.com

Abstrak- *Sebuah sistem penyambungan ATM dirancang untuk beroperasi pada kecepatan tinggi dan dapat melayani trafik yang beragam untuk permintaan layanan yang berkualitas. Teknologi yang dapat membuktikan kinerja dari suatu penyambungan ATM salah satunya adalah Algoritma Rerouting yang memiliki prosedur Contention Controller sebagai prosedur pengoperasiannya.*

Algoritma Rerouting merupakan suatu algoritma yang mampu mengatasi perutingan yang pada tingkat tertentu terjadi konflik, sehingga dapat mengembalikan jalannya ruting sesuai dengan alamat tujuan. Prosedur Contention Controller merupakan suatu pengendali yang terletak di elemen penyambungan digunakan untuk menentukan rute, dapat dilanjutkan atau tidak. Prosedur ini terdapat pada Algoritma Rerouting dengan menggunakan jaringan Banyan yang memiliki jalur tunggal pada rangka sambungan di penyambungan ATM.

Tugas Akhir ini dibuat untuk mensimulasikan Algoritma Rerouting baik itu tanpa konflik maupun dengan konflik di berbagai tingkat. Prosedur Contention Controller bertujuan untuk memberikan gambaran dengan rinci kapan terjadi konflik pada elemen penyambungan dan bertugas untuk menyeleksi perutingan sel-sel ATM berdasarkan nilai RNS-nya. Prosedur ini harus mengacu pada Algoritma Rerouting dan menghasilkan keluaran yang sesuai dengan algoritma tersebut.

Kata kunci:Algoritma Rerouting, Contention Controller, penyambungan ATM, jaringan Banyan.

I. PENDAHULUAN

1.1 Latar Belakang

Di dalam arsitektur sistem penyambungan ATM terdiri dari beberapa bagian diantaranya rangka sambungan (*switch fabric*). Rangka sambungan juga beraneka ragam, salah satunya adalah jaringan Banyan, yaitu jaringan interkoneksi tingkat jamak. Jaringan ini memiliki kontrol *routing* berdasarkan kontrol bitnya, dikenal sebagai *rerouting*. Sedangkan prosedur *contention controller* sendiri merupakan bagian dari perutingan apabila terjadi bloking yaitu terjadi lebih dari satu sel ATM yang dirutingkan.

Algoritma *rerouting* bekerja pada suatu algoritma interkoneksi tingkat jamak tertentu. Pada algoritma *rerouting* ini setiap sel yang datang akan dirutingkan melalui tingkat – tingkat yang ada pada jaringan sesuai dengan tanda jalurnya. Setiap kali terjadi konflik pada tingkat ke-*t* maka proses akan diulang pada tingkat ke *t*+1.

Prosedur *contention controller* merupakan suatu teknik pengontrolan sel yang terdapat pada pada elemen penyambungan. Setiap sel yang datang memasuki elemen penyambungan akan dijalankan sesuai dengan tujuan yang terdapat pada tanda jalur. Apabila terdapat lebih dari satu sel ATM yang dirutingkan, maka sel-sel ini akan mengantri di penyangga (b*uffer*) untuk menunggu giliran berdasarkan skala prioritas. Melalui mekanisme ini, sel-sel tersebut akan diputuskan diterima atau ditolak.

1.1 Tujuan

Tujuan pembuatan tugas akhir ini adalah mensimulasikan Algoritma *Rerouting* dan Prosedur *Contention Controller* yang terdapat pada jaringan Banyan sebagai salah satu rangka sambungan pada sistem penyambungan ATM.

1.2 Pembatasan Masalah

1. Sistem penyambungan menggunakan rangka sambungan dengan model jaringan Banyan dengan ukuran $N = 8$ dan dibuat menjadi 6 tingkat.
2. Konflik yang disimulasikan hanya pada tingkat kedua saja.
3. Pada Simulasi Algoritma *Rerouting* hanya mensimulasikan perjalanan *routing* dari asal hingga tujuan, tidak membahas pentransmisian sel.
4. Simulasi Prosedur *Contention Controler* hanya mengambil satu tingkat pada jaringan Banyan.

II. TINJAUAN UMUM MENGENAI ATM, JARINGAN PENYAMBUNGAN ATM DAN ALGORITMA *REROUTING*

2.1 Asynchronous Transfer Mode (ATM)

Asynchronous Transfer Mode (ATM), yang kadang-kadang ditunjukkan sebagai sel *relay*, merupakan titik kulminasi dari keseluruhan perkembangan yang terjadi dalam penyambungan sirkuit dan penyambungan paket selama 25 tahun terakhir.

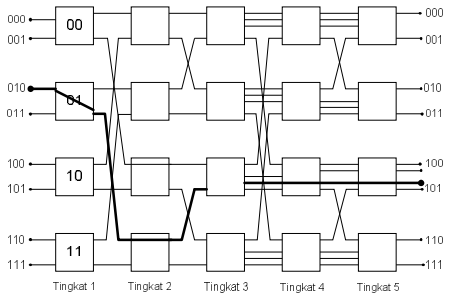
ATM dapat dipandang sebagai suatu evolusi *frame relay*. Perbedaan yang sangat jelas antara *frame relay* dan ATM adalah bahwa *frame relay* menggunakan paket panjang-bervariasi yang disebut bingkai, sedangkan ATM menggunakan paket panjang-tetap yang disebut dengan sel. Sebagaimana *frame relay*, ATM menyediakan sedikit *overhead* untuk mengkontrol kesalahan, tergantung pada reabilitas yang melekat pada sistem transmisi serta pada lapisan logika yang lebih tinggi di akhir sistem untuk menangkap dan memperbaiki kesalahan. Dengan menggunakan paket panjang-tetap, proses *overhead* dapat dikurangi bahkan lebih jauh untuk ATM yang dibandingkan dengan *frame relay*. Hasilnya adalah ATM dirancang sedemikian rupa agar mampu bekerja

dalam kisaran 10 sampai 100 Mbps serta untuk kisaran Gbps.

Model transfer *asynchronous* memakai sel-sel

2.2 Jaringan Banyan

Jaringan Banyan memiliki properti ruting. Meskipun ruter global, elemen penyambungan secara individu memungkinkan merutingkan sel secara tepat. Strategi ruting untuk setiap elemen penyambungan pada tingkat ke-*t* terlihat bit ke-k yang menunjukkan alamat tujuan dari sel dan mengirimkan sel tersebut ke bagian keluaran 'atas' jika bit tersebut adalah 0, dan menuju ke 'bawah ' bila keluaran bit adalah 1. Pada gambar 4 terdapat 8x8 jaringan Banyan, dan mencontohkan sebuah self-ruting. Sel tiba di masukan 010 dengan tujuan 101. Pada tingkat pertama dirutingkan ke bagian keluaran bawah karena bit pertama alamat tujuan adalah 1. Pada tingkat kedua, sel dirutingkan ke bagian keluaran atas karena bit kedua adalah 0, dan tingkat terakhir adalah ke bagian keluaran bawah karena bit terakhir dari alamat tujuan adalah 1. Maka didapatkan arah alamat yang otomatis memiliki tujuan yang diinginkan, tanpa memperdulikan dimana bagian masukan dimulai.



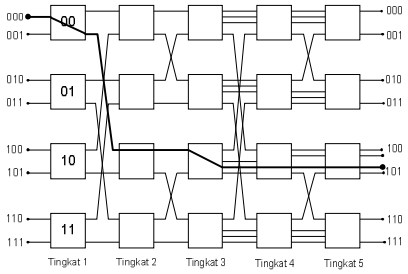
Gambar 1 Self-Ruting Penyambungan Jaringan Banyan 8x8

2.3 Algoritma *Rerouting* dan Prosedur *Contention Controller*

2.1.1 Algoritma *Rerouting*

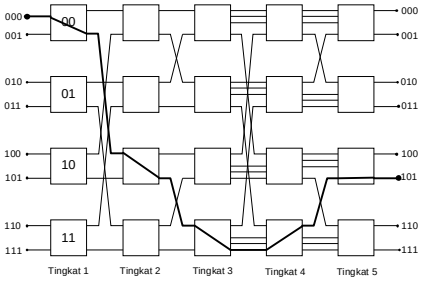
Berikut ini merupakan contoh yang merupakan bagian dari penjelasan *routing* sel dari garis masukan 000 ke garis masukan 101. Langkah–langkah *routing*-nya adalah sebagai berikut :

1. Karena jaringan yang digunakan adalah jaringan Banyan dengan ukuran (*N*) = 8, maka RNS = log_2 8 = 3.
2. Alamat bit tujuan (δ) dihitung dengan rumus :
   $\delta = d_j$ dimana
   $$j = (n-1) - (k-1) \bmod (n-1) \qquad (2\text{-}1)$$
   Contoh : alamat tujuan (*Fd*) = 101 = $d_2\, d_1\, d_0$
3. Untuk k = 1 (tingkat satu) dengan RNS 3,
   Sehingga $j = (3-1) - (1-1) \bmod (3-1) = 2$
   diketahui d_2 = 1, jadi bit tujuan (δ) - nya adalah 1
   Sehingga elemen penyambungan (EP) = $[00]_1$ mengurangi RNS menjadi 2. Dan melewatkan sel ATM menuju EP $[10]_2$ melalui *outlet* $[001]_1{}^0$.
4. Untuk tingkat k = 2 (tingkat kedua) dengan RNS = 2, j = 1, δ = d_1 diketahui d_1=0. Jadi bit tujuan–nya adalah 0. Sehingga EP$[10]_2$ mengurangi RNS menjadi 1 dan melewatkan sel ATM menuju EP$[10]_3$ melalui alamat tujuan $[100]_2{}^0$.
5. Untuk k = 3 (tingkat ketiga) dengan RNS = 1, karena RNS = 1, maka EP$[10]_3$ mengurangi RNS menjadi 0 dan merutekan sel ATM menuju tujuan akhir melalui alamat tujuan $[101]_3{}^1$ (*bypass link*). Gambar contoh *routing* pada pada Gambar 2.



Gambar 2 Algoritma *Rerouting* sel dari 000 menuju 101

Apabila terjadi konflik di tingkat ke-k+1, maka jalur *routing* akan diulang pada tingkat ke-k dengan cara RNS-nya kembali ke sebelumnya (tidak berkurang satu).



Gambar 3 Algoritma *Rerouting* dari 000 ke 101 dan terjadi konflik pada tingkat ke – 2

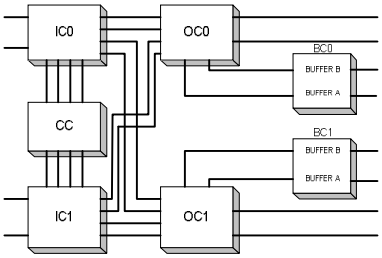
Pada gambar 3 diperlihatkan apabila terjadi konflik di tingkat ke-2. Dengan algoritma *Rerouting*, maka dapat diatasi sebagai berikut. Pada tingkat ke-1 nilai RNS bernilai 3, dan mengendalikan bit ke-3 pada alamat tujuan [101], yaitu 1 (mulai dari LSB paling kiri). Pada tingkat ke-2, RNS berkurang menjadi 2, mengendalikan bit ke-2 yaitu 0, namun pada tingkat ini terjadi konflik. Sehingga bit tidak sesuai tujuan. Maka pada tingkat ke-3, seharusnya RNS menjadi 1, RNS kembali berulang menjadi 2. Selanjutnya, *routing* berlangsung hingga tiba di alamat tujuan.

2.1.2 Prosedur *Contention Controller*

Prosedur *Contention Controller* merupakan komponen pengendali dalam elemen penyambungan pada algoritma *Rerouting*. Dapat dikatakan pula, prosedur *contention controller* merupakan gambaran secara terperinci mengenai apa yang terjadi pada suatu titik elemen penyambungan.

Pada penamaannya, prosedur *Contention Controller* diambil dari salah satu komponen pengendalinya. Pada prosedur ini, yang mewakili satu titik elemen penyambungan, akan dimulai pada masing- masing pengendali masukan (*input controller*) Pada gambar 4 merupakan blok diagram prinsip keja

dari prosedur *Contention Controller* pada suatu tingkat tertentu.

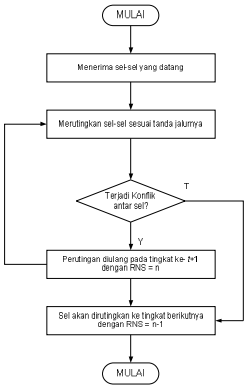


Gambar 4 Blok Diagram Prosedur *Contention Controller*

III. PERANCANGAN ALGORITMA *REROUTING* DAN PROSEDUR *CONTENTION CONTROLLER*

3.1 Algoritma *Rerouting*

3.1.1 Diagram Alir



Gambar 5 Diagram Alir Algoritma *rerouting*

3.1.2 Penentuan Kondisi

3.1.2.1 Alamat

Alamat disini terbagi menjadi dua, yaitu Asal dan Tujuan. Masing-masing terdiri dari bit-bit yang memiliki nilai RNS = 3, sehingga $N = 2^3 = 8$. Nilai N merupakan suatu variasi dari masukan dan keluaran pada suatu jaringan Banyan.

3.1.3.2 Keadaan

1. Tanpa Konflik
2. Konflik Tingkat ke-1
3. Konflik Tingkat ke-2
4. Konflik Tingkat ke-3

3.1.3 Perancangan Program

3.1.3.1 Rute

Pada *code editor* dituliskan beberapa kode program berupa prosedur, diantaranya betujuan untuk mendaftarkan komponen Rute, medefinisikan Rute, menentukan posisi Rute, menentukan besar nilai maksimum Rute, menentukan warna Rute, menentukan jalannya Rute, menampilkan Rute, dan menentukan bila Rute satu melewati Rute berikutnya.

3.1.3.2 Switching

Di elemen penyambungan akan terjadi :

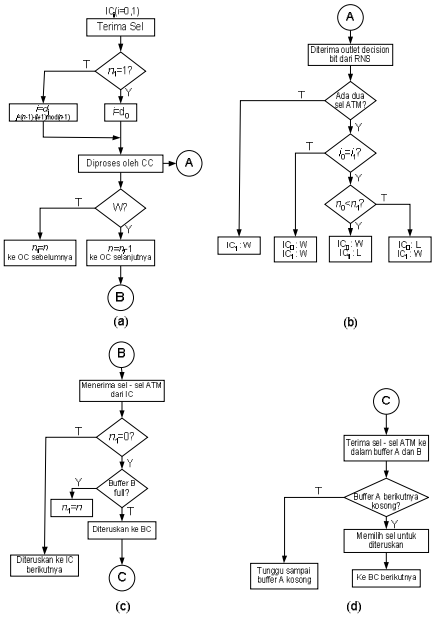
1. Pendeteksian jumlah alamt bit, harus berjumlah 3 bit.
2. Pendeteksian nilai RNS, jika belum mencapai 0 maka akan berkurang setelah melewati elemen ini.
3. Bit tujuan merupakan nilai RNS pada elemen tersebut.

3.1.3.3 URerouting

Pada *unit* ini berisi program utama yang akan mendefinisikan jalannya simulasi. Mulai dengan menentukan kondisi yaitu alamat Asal dan Tujuan. Selanjutnya menentukan pilihan Keadaan, dan menjalankan simulasi.

3.2 Prosedur *Contention Controller*

3.2.1 Diagram Alir



Gambar 6 Diagram Alir *Contention Controller*
(a) IC (b) CC (c) OC (d) BC

3.2.2 Kondisi Inisialisasi

Sebelum simulasi diljalankan, terlebih dahulu akan dilakukan suatu inisialisasi yang bertujuan diantaranya :

1. Menentukan pilihan masukan yang hendak diisi, IC0, IC1 ataupun keduanya.
2. Mengisi alamat bit tujuan dan RNS pada IC0 maupun IC1.
3. Menentukan pilihan untuk mengisi *Buffer* A atau *Buffer* B, boleh tidak diisi.
4. Mengisi alamat bit tujuan pada *Buffer* A atau *Buffer* B.

3.2.2 Perancangan Program

3.2.2.1 UContention

Pada *unit* ini akan berisi program utama yang terdiri dari memberi masukan inisialisasi, menjalankan simulasi, menata ulang komponen, dan keluar dari simulasi.

3.2.2.2 UContentionController

UContentionController merupakan *unit* yang berisikan kode program untuk mengontrol masing-masing cara kerja dari blok-blok *controller* yang menyusun operasi prosedur ini secara keseluruhan. Kode program dituliskan berdasarkan diagram alir pada gambar .

3.2.2.3 UInit

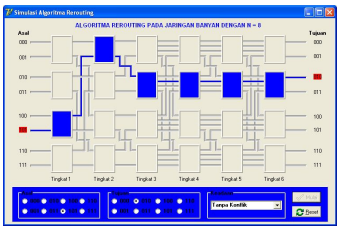
Pada *unit* ini berisi kode program proses Inisialisasi. Pada Inisialisasi, akan dilakukan pengecekan pengisian baik itu bit tujuan maupun nilai RNS. Alamat bit tujuan harus memiliki tiga bit yang terdiri dua bilangan biner yaitu 0 dan 1. Sedangkan nilai RNS hanya berkisar antara 1, 2, atau 3, tidak boleh selain angka tersebut.

IV. PENGUJIAN PROGRAM DAN ANALISA

Pada Bab ini akan dijelaskan implementasi dari program yang telah dirancang. Berikut ini merupakan hasil pengujian program untuk beberapa kondisi yang telah ditentukan.

4.1 Simulasi Algoritma *Rerouting*

4.1.1 *Routing* Tanpa Konflik



Gambar 7 Tampilan dari [101] menuju [010] dengan Algoritma *Rerouting* Tanpa Konflik

Pada gambar 7 ditampilkan *routing* dari 'Asal' [101] dan 'Tujuan' [010]. Setelah itu, akan dicek dengan menggunakan Algoritma *Rerouting*, sudah memenuhi atau belum.

Dari proses *routing* yang telah dicek dengan menggunakan Algoritma *Rerouting* tadi, tampilan yang dihasilkan akan sesuai dengan gambar 7.

Untuk berbagai variasi 'Asal' dan 'Tujuan', diperoleh tampilan yang sesuai bila dicek dengan menggunakan Algoritma *Rerouting*.

4.1.2 *Routing* dengan Konflik

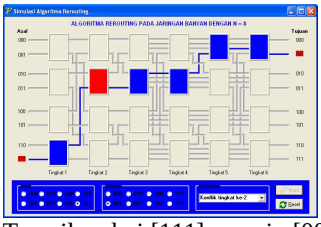
4.1.2.1 Konflik di Tingkat ke-1

Konflik tidak dapat terjadi di tingkat ke-1. Berikut ini penjelasan yang mendasari hal tersebut.

Apabila konflik terjadi di tingkat ke-1, maka pada tingkat ke-2 akan diulang nilai RNS-nya, yaitu tetap menjadi 3. Dengan mengasumsikan konflik sebagai lebih dari satu sel ATM yang datang, maka pada elemen penyambungan di tingkat itu akan dicek nilai RNS-nya. Sedangkan konflik di tingkat ke-1, belum terjadi pengurangan RNS (dalam pengertian tidak memiliki nilai sisa pada tingkat tersebut). Hal ini yang menyebabkan konflik tidak mungkin terjadi pada tingkat ke-1, karena tidak sesuai dengan Algoritma *Rerouting*.

4.1.2.2 Konflik di Tingkat ke-2

Untuk berbagai variasi 'Asal' dan 'Tujuan', diperoleh *routing* yang tepat bila dicek dengan menggunakan Algoritma *Rerouting*. Sehingga dapat disimpulkan *routing* Keadaan Konflik tingkat ke-2 ini sesuai dengan Algoritma *Rerouting*, apabila menggunakan jaringan Banyan dengan $N = 8$.



Gambar 8 Tampilan dari [111] menuju [001] dengan Algoritma *Rerouting* bila terjadi Konflik Tingkat ke-2

4.1.2.3 Konflik di Tingkat ke-3

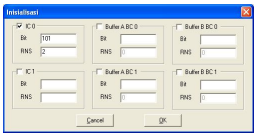
Sebagian besar *routing* dengan keadaan konflik di tingkat ke-3, menghasilkan alamat tujuan yang tidak tepat. Hal ini dikarenakan *routing link* dan *bypass link* yang menyusun jaringan Banyan ini mengendalikan bit sesuai keluaran *link-link* di tiap-tiap elemen penyambungan, pada saat dicek dengan Algoritma *Rerouting*. Dan hanya dapat diatasi dengan merubah susunan jaringan Banyan dengan nilai *N* yang lebih besar. Oleh karena itu Simulasi Algoritma *Rerouting* ini tidak dapat berlaku pada konflik di tingkat ke-3.

4.2 Simulasi Prosedur *Contention Controller*

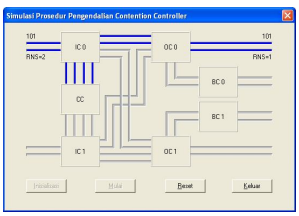
4.2.1 Simulasi dengan Berbagai Macam Masukan di Lembar Inisialisasi

4.2.1.1 Inisialisasi Hanya pada IC0

Pada IC0 akan diisi bit tujuan [101] dengan RNS = 2. Seperti pada gambar 9. Hasil simulasi dapat dilihat pada gambar 10, rute diberi warna biru menandakan bahwa berasal dari IC0. Selanjutnya akan menuju OC0, karena pada bit tujuan akan menuju ke 0, RNS akan berkurang menjadi 1.



Gambar 9 Tampilan Lembar Kerja Inisialisasi dengan Hanya Mengisi IC0



Gambar 10 Tampilan Simulasi *Contention Controller* dengan tujuan [101] dan RNS = 2 di IC0

Hasil pengisian hanya pada masukan IC0 tidak akan menimbulkan *Loss* dimasing-masing masukan, sehingga tidak akan menimbulkan konflik.

4.2.1.2 Inisialisasi Hanya pada IC1

Pada IC1 akan diisi bit [001] dengan RNS 3. Maka rute akan berwarna merah hingga keluaran. Karena RNS = 3, maka bit tujuan akan menuju OC0, dan RNS berkurang menjadi 2.

Hasil pengisian hanya pada masukan IC1 tidak akan menimbulkan *Loss* dimasing-masing masukan, sehingga tidak akan menimbulkan konflik.

4.2.1.3 Inisialisasi Hanya pada IC0 dan IC1

Hasil pengisisan pada kedua masukan ini akan menghasilkan *Loss* pada kondisi bila bit tujuan yang dihasilkan adalah sama.

4.2.1.4 Inisialisasi pada IC0, IC1 dan *Buffer* B

Buffer B merupakan penyangga pada elemen penyambungan apabila nilai RNS keluaran seteleh diproses dengan Algoritma *Rerouting* berkurang menjadi 0. Pada inisialisasi dengan penambahan sel di ***Buffer*** B, sel-ATM pada masing-masing IC, nilai RNS-nya tidak boleh 1 karena sel akan dinyatakan konflik dan mengalami *loss*. Karena *Buffer* B diisi penuh oleh sel ATM, sehingga sel yang nilainya sudah 1 dan berkurang menjadi 0 tidak dapat menuju *Buffer* B.

Namun bila RNS pada masing-masing IC tidak bernilai 1, maka tampilan simulasi akan berwarna putih yang keluar dari BC0 dan BC1 menandakan pada *Buffer* B yang tadi terisi akan dilanjutkan menuju *Buffer* A di IC pada tingkat berikutnya.

4.2.1.5 Inisialisasi pada IC0, IC1 dan *Buffer* A

Pada inisialisasi ini akan diisi *Buffer* A dalam keadaan penuh, IC0 dan IC1 juga diisi masing- masing dengan [001] RNS = 2 dan [111] RNS = 3. Maka keluaran BC0 dan BC1 akan berwarna hitam, yang menandakan *Buffer* A yang terisi tadi dilanjutkan menuju *Buffer* A di IC pada tingkat berikutnya.

Cara kerja *Buffer* A berbeda dengan *Buffer* B, karena pengisian *Buffer* A tidak mempengaruhi *Contention Controller*. Apabila nilai RNS pada masukan berkurang menjadi 0, maka akan dilanjutkan ke *Buffer* B. Sehingga tidak mengakibatkan *loss* pada masing-masing masukan.

V. PENUTUP

5.1 Kesimpulan

Setelah merancang dan mensimulasikan Algoritma *Rerouting* dan Prosedur *Contention Controller*, maka diperoleh hasil-hasil yang dapat disimpulkan sebagai berikut :

1. Simulasi Algoritma *Rerouting* jaringan Banyan dengan N =8 merupakan *routing* pada sistem penyambungan ATM menggunakan jaringan Banyan dengan $N = 2^n$. Sehingga menghasilkan nilai n = 3, yaitu nilai RNS yang berfungsi mengendalikan bit pada suatu alamat tujuan.
2. Konflik yang terjadi pada simulasi Algoritma *Rerouting* hanya terjadi pada tingkat ke-2 saja, karena memenuhi algoritma dan mencapai alamat tujuan dengan tepat.
3. Konflik pada tingkat ke-1 tidak dapat terjadi, karena nilai RNS di tingkat ke-1 merupakan asumsi nilai RNS awal dan belum terjadi pengurangan nilai RNS (dalam pengertian tidak memilki nilai sisa pada tingkat ke-1) sebagai parameter perutingan.
4. Konflik pada tingkat ke-3 tidak dapat terjadi pada jaringan Banyan dengan N = 8, karena sebagian besar alamat tujuan yang dihasilkan tidak sesuai dengan penggunaan Algoritma *Rerouting*.
5. Konflik di tingkat 4, 5, dan 6 tidak dapat terjadi karena nilai RNS sudah berkurang menjadi 0 sehingga Algoritma *Rerouting* tidak dapat digunakan.
6. Pada simulasi Prosedur *Contention Controller* konflik hanya dapat terjadi bila salah satu masukan IC mengalami *Loss*, Algoritma *Rerouting* dilanjutkan ditingkat berikutnya.
7. Pengisian *Buffer* B dapat menyebabkan konflik, bila salah satu masukan memiliki RNS 1.
8. Pengisian *Buffer*A tidak menimbulkan gangguan *Loss* pada proses di *Contention Controller*, sehingga tidak akan menyebabkan konflik.

5.2 Saran

1. Membuat simulasi Algoritma *Rerouting* menggunakan jaringan Banyan dengan nilai N > 8, sehingga konflik yang terjadi dapat divariasikan pada tingkat lebih besar dari tingkat ke-2.
2. Membuat simulasi *Contention Controller* secara bertingkat, sehingga dapat dianalisa dengan menggunakan grafik.

BIOGRAFI

Yudhitya Sorrenti

(L2F 000 652)

Mahasiswi Jurusan Teknik Elektro, Fakultas Teknik Universitas Diponegoro Semarang, dengan pilihan konsentrasi Elektronika Telekomunikasi

Menyetujui/Mengesahkan:

Pembimbing I	Pembimbing II
Wahyul Amien S. ST, MT	**Imam Santoso, ST, MT**
NIP. 132 137 934	**NIP. 132 162 546**