BAB III METODOLOGI DAN PERANCANGANG SISTEM

3.1 Metodologi Penelitian

Beberapa metode penelitian yang digunakan untuk melakukan penelitian ini antara lain:

- Studi Literatur

  Dalam studi literatur dilakukan pembelajaran terhadap konsep-konsep *routing protocol* yang telah digunakan pada arsitektur komunikasi NoC yang memiliki topologi *mesh*. Beberapa metode dan algoritma yang digunakan untuk membuat sebuah *routing protocol* juga dipelajari, seperti *Temperature-aware* [10], dan *Balanced Distribution* [11]. Bagaimana sistem NoC bekerja serta, konsep-konsep pembuatan arsitektur komunikasi UTAR NOC, bagaimana struktur TOP, *Test Bench,* hingga *DUT (Device Under Test)* pada arsitektur UTAR NOC.

- Perancangan Algoritma

  Dalam tahapan ini, dilakukan perancangan terhadap *DUT* dengan menggunakan aplikasi modelSIM sebagai *debugger* dan simulator.Perancangan *DUT* berupa *Trial and error* pada beberapa kasus yang akan diberikan kepada *routing protocol* baru yang akan diajukan. Perancangan bagaimana *routing protocol* memilih *path*, akan dijelaskan pada *flowchart* yang telah dibuat.

- Perancangan Sistem

  Pada perancangan sistem, *environment* yang ada pada arsitektur UTAR NoC diubah sesuai dengan parameter yang dibutuhkan *routing protocol*. File TOP, serta *test bench* yang ada pada arsitektur UTAR NoC ditambahkan beberapa parameter dan juga menambahkan beberapa bit pada *flit* yang akan dikirim pada setiap.

- Pengujian dan Implementasi Sistem

  Tahapan pengujian dan implementasi sistem dimulai dengan mengimplementasikan *routing prortocol* baru yang telah dibuat pada penelitian ini kedalam UTAR NoC yang telah dimodifikasi sesuai dengan *environment* yang dibubutuhkan oleh *routing protocol*. Pengujian akan dilakukan dengan menggunakan beberapa *test cases* dasar hingga dengan *traffic pattern* kompleks. Hasil pengujian berupa berapa *clock time* yang dibutuhkan untuk menyelesaikan *test cases*, berapa waktu rata-rata yang dibutuhkan setiap *processing block* untuk mengirimkan paket ke *processing block* lainnya, berapa waktu maksimum serta waktu minimum dalam pengiriman paket pada setiap *processing block*, *throughput* pada jaringan, dan *latency* rata-rata pada jaringan.

- Analisis

  Tahapan analisis merupakan tahapan dimana dilakukan pengambilan hasil performa *latency*, dan *throughput* pada setiap hasil dari *test cases* yang telah dijalankan. Hasil ini akan dibandingkan dengan hasil dari performa arsitektur komunikasi UTAR NoC pada setiap *test case*..

3.2 Perancangan Algoritma

Pada penilitian ini, *algoritma routing* pada arsitektur komunikasi UTAR NoC akan diperbarui dengan menambahkan TTL sebagai bahan pertimbangan. *Routing* algoritma ini akan mencarikan rute tercepat untuk sampai pada tujuan. *Routing* algoritma berbasis TTL ini akan memberikan batas minimum *bandwidth* pada setiap *interface router,* dan menambahkan beberapa bit TTL pada setiap *header packet*, parameter yang berisikan tentang informasi apakah ada *router* tetangga pada setiap sisi *router* yang berfungsi untuk mencegah *packet* dibuang ke *interface* kosong, serta parameter baru yang digunakan sebagai penanda darimana paket berasal.

Cara kerja TTL pada *algoritma routing* berbasis TTL tidak sama dengan TTL pada umumnya. Walau TTL pada *algoritma routing* berbasis TTL berfungsi sama pada TTL umumnya yaitu membatasi jumlah hop *packet*, namun *packet* tidak akan di *drop* seperti pada umumnya. Pada *algoritma routing* berbasis TTL *packet* akan mendapat perlakuan khusus dari algoritma. *Packet* yang membawa nilai TTL sama dengan nilai maximum TTL yang telah ditentukan, tidak akan diberi rute alternatif seperti *packet* lain, *packet* hanya akan diberi rute tercepat untuk mencapai *router* tujuan.

Cara kerja *algoritma routing* berbasis TTL awalnya sama dengan, *xy adaptive routing*, yang membedakan adalah *algoritma routing* berbasis TTL akan memberikan rute alternatif tambahan apabila rute awal yang diberikan memiliki nilai *bandwidth* sama dengan nilai minimum *bandwidth*. Namun sebelum rute dikirim ke rute alternatif, bit TTL pada *packet* juga akan dibaca untuk mencegah terjadinya *livelock*, karena apabila suatu *packet* terus menerus diberi rute alternatif,

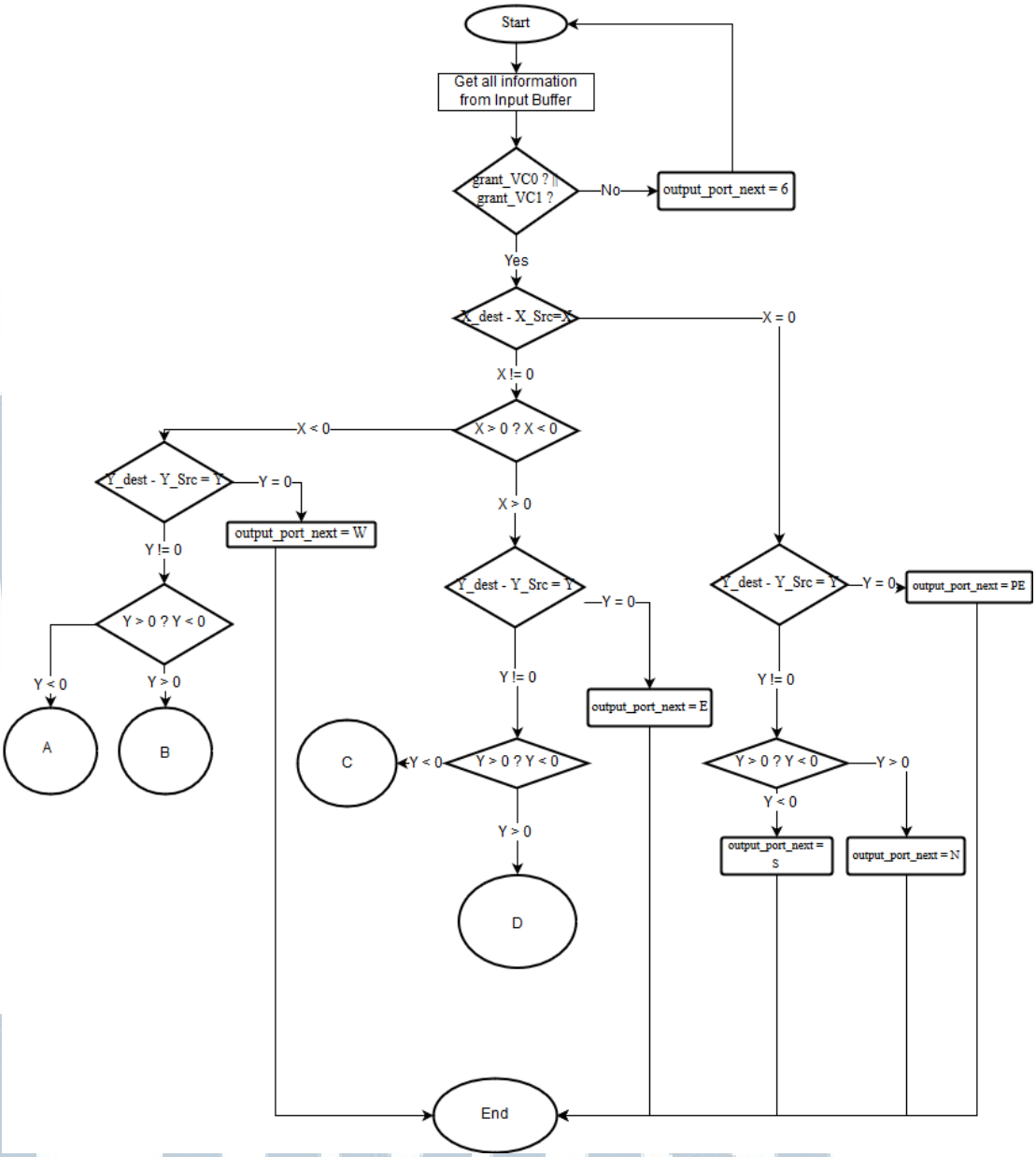
packet tersebut tidak akan pernah sampai. *Packet* yang memiliki nilai TTL sama dengan nilai maximum TTL hanya akan mendapatkan rute terdekat ke tujuan *router*.

Rute alternatif diberikan apabila memang *router* memiliki tetangga pada rute yang diberikan. Hal ini dipastikan menggunakan parameter yang telah disediakan diawal. Rute alternatif yang diberikan juga telah memastikan bahwa rute yang diberikan bukan rute darimana *packet* berasal.

Perancangan algoritma dimulai dengan menggambarkan *flow chart* yang menggambarkan alur bagaimana pemilihan rute dilakukan.

3.2.1 Flow Chart

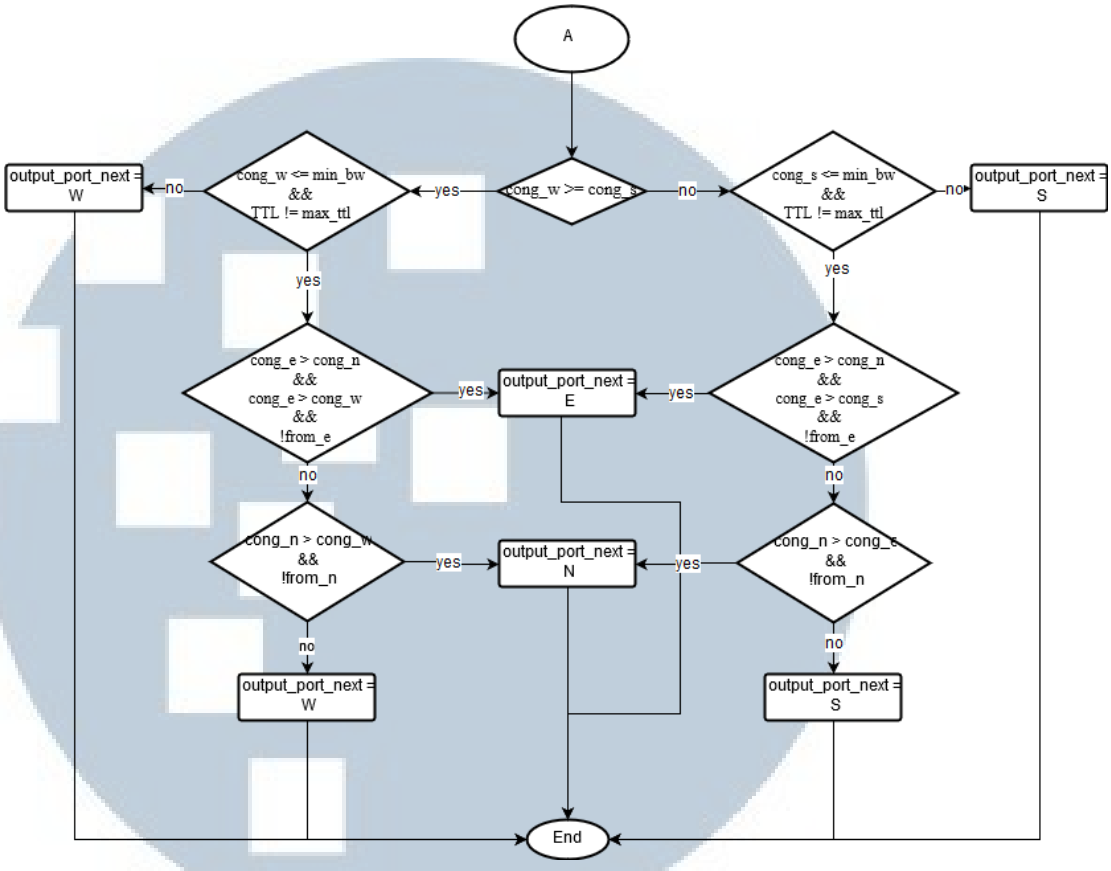
Flow chart diagram menggambarkan alur dari bagaimana algoritma *routing protocol* bekerja. Pada Gambar 3.1 menggambarkan bagaimana alur algoritma *routing protocol* menentukan *output_port_next* sesuai dengan pertimbangan yang ada. *Routing protocol* berjalan ketika didapatnya *grant_vc0* maupun *grant_vc1*. Apabila belum ada *grant* yang diberikan ke *Virtual Channel* 0 maupun 1, nilai dari *output_port_next* akan diubah menjadi 6 yang berarti data tersebut akan disimpan dalam *router* hingga adanya *grant*. Ketika telah diterimanya *grant*, algoritma akan memeriksa dimana *router* destinasi berada. Algoritma memeriksanya dengan cara mengurangi koordinat dari destinasi dengan koordinat *router*.



Gambar 3.1 Flow Chart MAIN

Pada Gambar 3.1 dijelaskan bahwa, ketika *input_buffer* sampai ke *router* dan memanggil *routing protocol*, *routing protocol* akan menerima semua informasi yang dibawa oleh *input_buffer* dari *testbench*, *routing protocol* akan menentukan kemana *packet* yang dibawa berdasarkan informasi yang diberikan *input_buffer*. Informasi yang dibawa *input_buffer* yakni seperti koordinat tujuan dan sumber *packet*, darimana *packet* tersebut berasal, TTL, dan data dari *packet* itu sendiri.

Ketika koordinat dari x dan y destinasi dikurangkan dengan koordinat dari x dan y sumber sama dengan nol, nilai *output_port_next* akan diubah menjadi PE, yang berarti *packet* sudah sampai di tujuan. Namun apabila hasil dari koordinat y destinasi dikurangkan dengan koordinat y *router* lebih besar dari nol, nilai *output_port_next* akan diubah menjadi N yang berarti *packet* akan diteruskan melalui *port* ke utara. Apabila hasil pengurangan lebih kecil dari nol, nilai *output_port_next* akan diubah menjadi S yang berarti *packet* akan diteruskan melalui *port* selatan. Namun apabila hasil dari pengurangan koordinat x destinasi dengan koordinat x *router* lebih besar dari nol dan hasil dari pengurangan koordinat y destinasi dan koordinat y lokasi adalah nol, nilai dari *output_port_next* adalah E yang berarti *packet* akan diteruskan melalui *port* timur. Begitupun sebaliknya ketika hasil dari pengurangan koordinat x destinasi dengan koordinat x *router* lebih kecil dari nol, nilai dari *output_port_next* adalah W yang berarti *packet* akan diteruskan melalui *port* barat. Penjelasan ketika hasil pengurangan dari koordinat x destinasi dengan koordinat x *router* tidak sama dengan nol, dan begitupun dengan hasil pengurangan dari koordinat y destinasi dengan koordinat y *router* akan dijelaskan pada Gambar 3.2,Gambar 3.3, Gambar 3.4, Gambar 3.5.



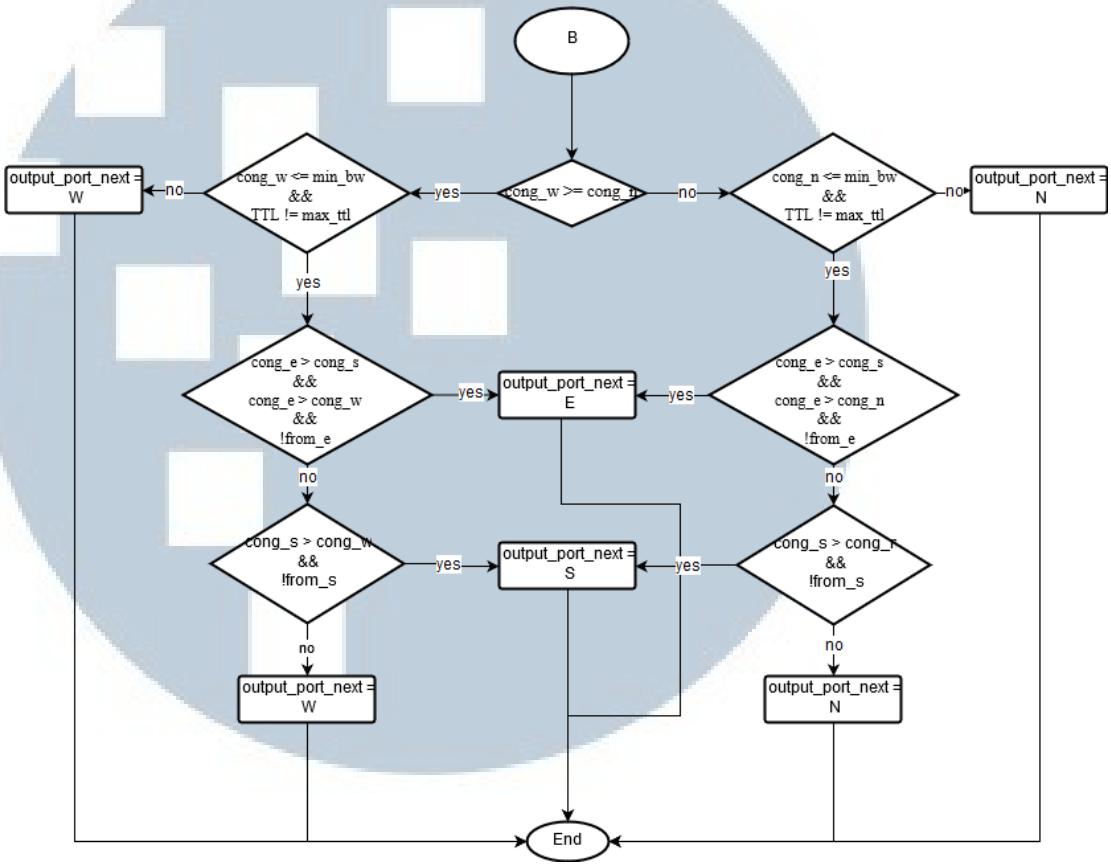
Gambar 3.2 Flow Chart A

Pada Gambar 3.2, menunjukkan alur algoritma ketika hasil pengurangan koordinat x destinasi dengan koordinat x *router* lebih kecil dari nol, begitupun hasil pengurangan koordinat y destinasi dengan koordinat y *router* lebih kecil dari nol. Algoritma akan memeriksa *nilai congestion* yang ada pada barat apakah lebih besar dari *nilai congestion* yang ada pada *port* selatan. Apabila *nilai congestion* pada *port* barat lebih besar algoritma akan memeriksa apakah *nilai congestion* pada *port* barat lebih besar dari minimal *bandwidth* yang telah ditentukan. Jika benar maka nilai dari *output_port_next* akan diubah menjadi W yang berarti *packet* akan dikirim melalui *port* barat. Apabila nilai *congestion* pada *port* barat memiliki *bandwidth* dibawah batas minimal *bandwidth* yang ditentukan, dan TTL yang dibawa pada *packet* belum mencapai nilai maximum, algoritma akan mengecek apakah nilai

congestion pada *port* timur lebih besar dari nilai *congestion* yang ada pada *port* utara dan barat, dan algoritma juga memastikan kalau *packet* tidak datang dari *port* timur, jika semua kondisi telah dipenuhi, nilai *output_port_next* akan diubah menjadi E, yang berarti *packet* akan dikirim melalui *port* timur. Namun apabila kondisi ada yang tidak dipenuhi algoritma akan mengecek nilai *congestion* pada *port* utara apakah nilai *congestion* pada *port* utara lebih besar dari nilai *congestion port* barat dan *packet* tidak datang dari *port* utara. Apabila kondisi terpenuhi, *packet* akan dikirimkan melalui *port* utara. Namun apabila ada satu kondisi yang tidak terpenuhi, *packet* akan dipaksa dikirim melalui *port* barat.

Ketika nilai *congestion* yang ada pada barat lebih kecil dari nilai *congestion* yang ada pada *port* selatan, algoritma akan memeriksa apakah nilai *congestion* pada *port* selatan lebih besar dari minimal *bandwidth* yang telah ditentukan. Jika benar maka nilai dari *output_port_next* akan diubah menjadi S yang berarti *packet* akan dikirim melalui *port* selatan. Apabila nilai *congestion* pada *port* selatan memiliki *bandwidth* dibawah batas minimal *bandwidth* yang ditentukan, dan TTL yang dibawa pada *packet* belum mencapai nilai maximum, algoritma akan mengecek apakah nilai *congestion* pada *port* timur lebih besar dari nilai *congestion* yang ada pada *port* utara dan selatan, dan algoritma juga memastikan kalau *packet* tidak datang dari *port* timur, jika semua kondisi telah dipenuhi, nilai *output_port_next* akan diubah menjadi E, yang berarti *packet* akan dikirim melalui *port* timur. Namun apabila kondisi ada yang tidak dipenuhi algoritma akan mengecek nilai *congestion* pada *port* utara apakah nilai *congestion* pada *port* utara lebih besar dari nilai *congestion port* selatan dan *packet* tidak datang dari *port* utara. Apabila kondisi

terpenuhi, *packet* akan dikirimkan melalui *port* utara. Namun apabila ada satu kondisi yang tidak terpenuhi, *packet* akan dipaksa dikirim melalui *port* selatan.



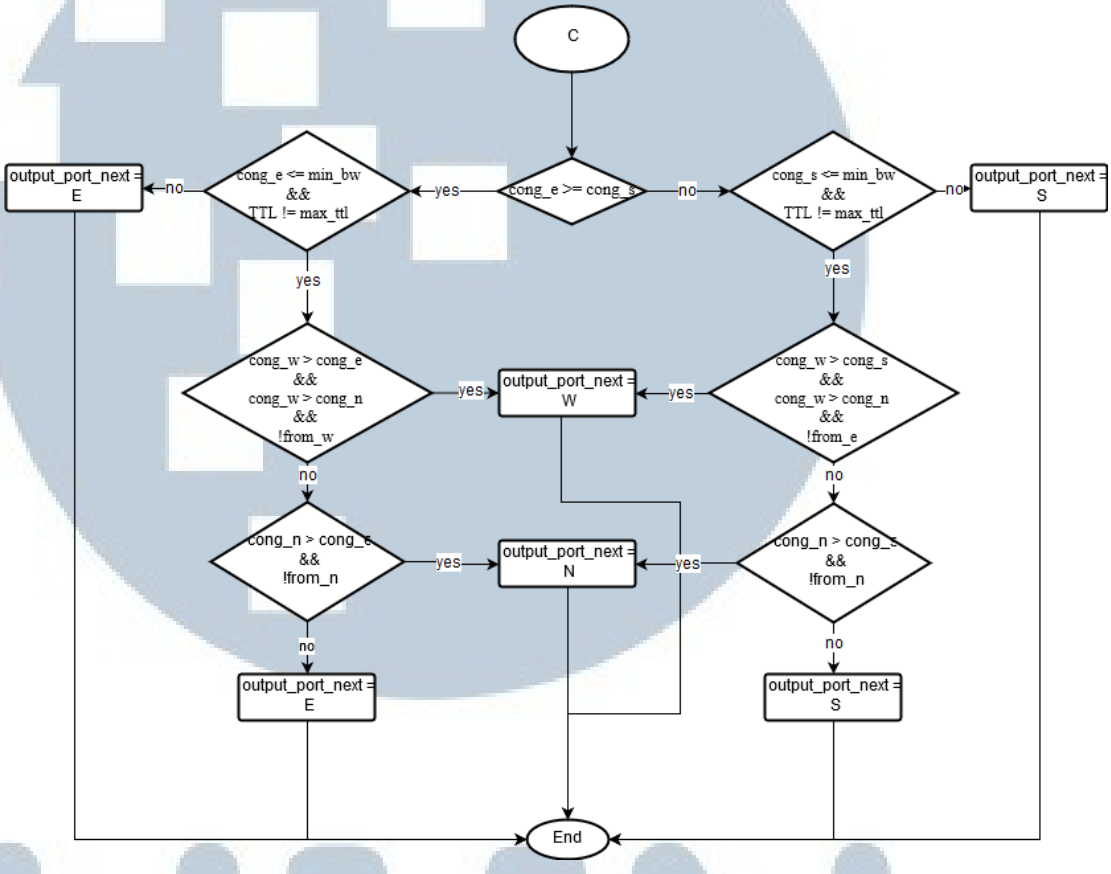
Gambar 3.3 Flow Chart B

Pada Gambar 3.3, menunjukkan alur algoritma ketika hasil pengurangan koordinat x destinasi dengan koordinat x *router* lebih kecil dari nol, dan hasil pengurangan koordinat y destinasi dengan koordinat y *router* lebih besar dari nol. Algoritma akan memeriksa nilai *congestion* yang ada pada barat apakah lebih besar dari nilai *congestion* yang ada pada *port* utara. Apabila nilai *congestion* pada *port* barat lebih besar algoritma akan memeriksa apakah nilai *congestion* pada *port* barat lebih besar dari minimal *bandwidth* yang telah ditentukan. Jika benar maka nilai dari *output_port_next* akan diubah menjadi W yang berarti *packet* akan dikirim melalui *port* barat. Apabila nilai *congestion* pada *port* barat memiliki *bandwidth*

dibawah batas minimal *bandwidth* yang ditentukan, dan TTL yang dibawa pada *packet* belum mencapai nilai maximum, algoritma akan mengecek apakah nilai *congestion* pada *port* timur lebih besar dari nilai *congestion* yang ada pada *port* selatan dan barat, dan algoritma juga memastikan kalau *packet* tidak datang dari *port* timur, jika semua kondisi telah dipenuhi, nilai *output_port_next* akan diubah menjadi E, yang berarti *packet* akan dikirim melalui *port* timur. Namun apabila kondisi ada yang tidak dipenuhi algoritma akan mengecek nilai *congestion* pada *port* selatan apakah nilai *congestion* pada *port* selatan lebih besar dari nilai *congestion port* barat dan *packet* tidak datang dari *port* selatan. Apabila kondisi terpenuhi, *packet* akan dikirimkan melalui *port* selatan. Namun apabila ada satu kondisi yang tidak terpenuhi, *packet* akan dipaksa dikirim melalui *port* barat.

Ketika nilai *congestion* yang ada pada barat lebih kecil dari nilai *congestion* yang ada pada *port* utara, algoritma akan memeriksa apakah nilai *congestion* pada *port* utara lebih besar dari minimal *bandwidth* yang telah ditentukan. Jika benar maka nilai dari *output_port_next* akan diubah menjadi N yang berarti *packet* akan dikirim melalui *port* utara. Apabila nilai *congestion* pada *port* utara memiliki *bandwidth* dibawah batas minimal *bandwidth* yang ditentukan, dan TTL yang dibawa pada *packet* belum mencapai nilai maximum, algoritma akan mengecek apakah nilai *congestion* pada *port* timur lebih besar dari nilai *congestion* yang ada pada *port* utara dan selatan, dan algoritma juga memastikan kalau *packet* tidak datang dari *port* timur, jika semua kondisi telah dipenuhi, nilai *output_port_next* akan diubah menjadi E, yang berarti *packet* akan dikirim melalui *port* timur. Namun apabila kondisi ada yang tidak dipenuhi algoritma akan mengecek nilai *congestion* pada *port* selatan apakah nilai *congestion* pada *port* selatan lebih besar dari nilai

congestion port utara dan *packet* tidak datang dari *port* selatan. Apabila kondisi terpenuhi, *packet* akan dikirimkan melalui *port* selatan. Namun apabila ada satu kondisi yang tidak terpenuhi, *packet* akan dipaksa dikirim melalui *port* utara.



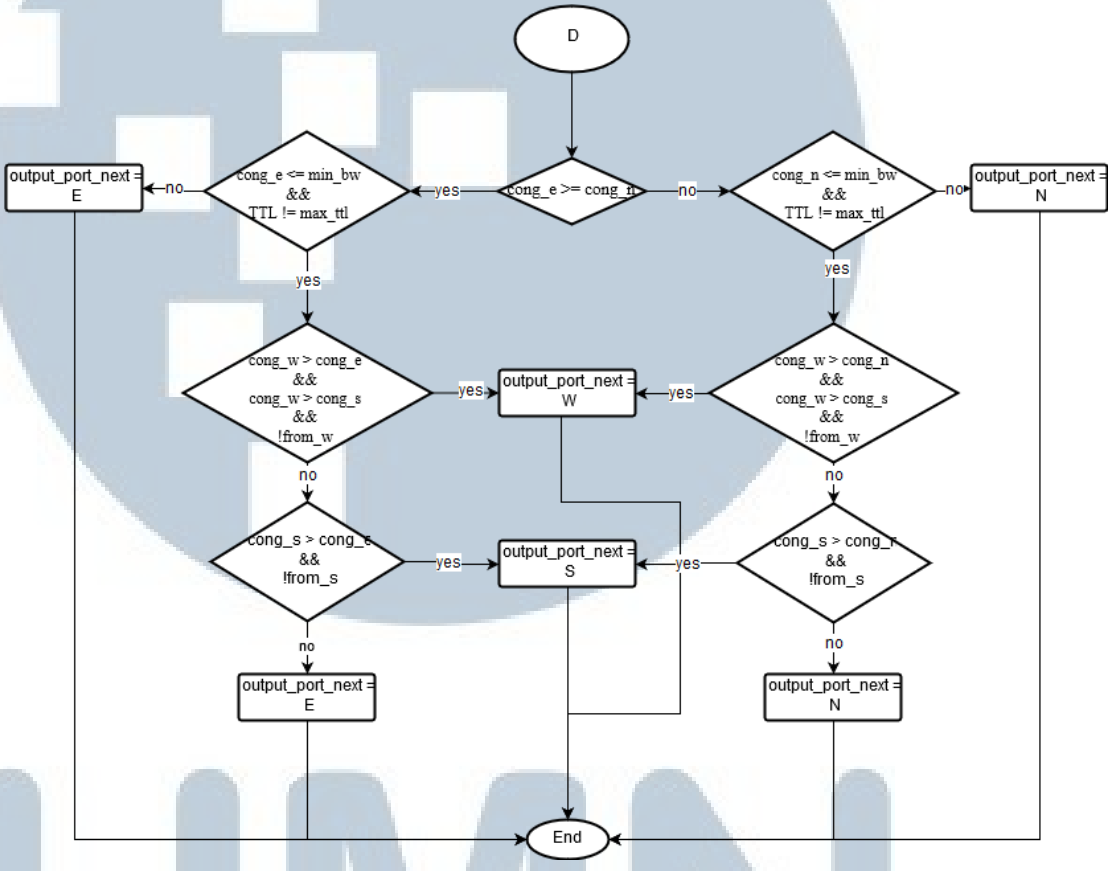
Gambar 3.4 Flow Chart C

Pada Gambar 3.4, menunjukkan alur algoritma ketika hasil pengurangan koordinat x destinasi dengan koordinat x *router* lebih besar dari nol, dan hasil pengurangan koordinat y destinasi dengan koordinat y *router* lebih kecil dari nol. Algoritma akan memeriksa nilai *congestion* yang ada pada timur apakah lebih besar dari nilai *congestion* yang ada pada *port* selatan. Apabila nilai *congestion* pada *port* timur lebih besar algoritma akan memeriksa apakah nilai *congestion* pada *port* timur lebih besar dari minimal *bandwidth* yang telah ditentukan. Jika benar maka nilai dari *output_port_next* akan diubah menjadi E yang berarti *packet* akan dikirim

melalui *port* timur. Apabila nilai *congestion* pada *port* timur memiliki *bandwidth* dibawah batas minimal *bandwidth* yang ditentukan, dan TTL yang dibawa pada *packet* belum mencapai nilai maximum, algoritma akan mengecek apakah nilai *congestion* pada *port* barat lebih besar dari nilai *congestion* yang ada pada *port* timur dan utara, dan algoritma juga memastikan kalau *packet* tidak datang dari *port* barat, jika semua kondisi telah dipenuhi, nilai *output_port_next* akan diubah menjadi W, yang berarti *packet* akan dikirim melalui *port* barat. Namun apabila kondisi ada yang tidak dipenuhi algoritma akan mengecek nilai *congestion* pada *port* utara apakah nilai *congestion* pada *port* utara lebih besar dari nilai *congestion* *port* timur dan *packet* tidak datang dari *port* utara. Apabila kondisi terpenuhi, *packet* akan dikirimkan melalui *port* utara. Namun apabila ada satu kondisi yang tidak terpenuhi, *packet* akan dipaksa dikirim melalui *port* timur.

Ketika nilai *congestion* yang ada pada timur lebih kecil dari nilai *congestion* yang ada pada *port* selatan, algoritma akan memeriksa apakah nilai *congestion* pada *port* selatan lebih besar dari minimal *bandwidth* yang telah ditentukan. Jika benar maka nilai dari *output_port_next* akan diubah menjadi S yang berarti *packet* akan dikirim melalui *port* selatan. Apabila nilai *congestion* pada *port* selatan memiliki *bandwidth* dibawah batas minimal *bandwidth* yang ditentukan, dan TTL yang dibawa pada *packet* belum mencapai nilai maximum, algoritma akan mengecek apakah nilai *congestion* pada *port* barat lebih besar dari nilai *congestion* yang ada pada *port* utara dan selatan, dan algoritma juga memastikan kalau *packet* tidak datang dari *port* barat, jika semua kondisi telah dipenuhi, nilai *output_port_next* akan diubah menjadi W, yang berarti *packet* akan dikirim melalui *port* barat. Namun apabila kondisi ada yang tidak dipenuhi algoritma akan mengecek nilai

congestion pada *port* utara apakah nilai *congestion* pada *port* utara lebih besar dari nilai *congestion port* selatan dan *packet* tidak datang dari *port* utara. Apabila kondisi terpenuhi, *packet* akan dikirimkan melalui *port* utara. Namun apabila ada satu kondisi yang tidak terpenuhi, *packet* akan dipaksa dikirim melalui *port* selatan.



Gambar 3.5 Flow Chart D

Pada Gambar 3.5, menunjukkan alur algoritma ketika hasil pengurangan koordinat x destinasi dengan koordinat x *router* lebih besar dari nol, begitupun hasil pengurangan koordinat y destinasi dengan koordinat y *router* lebih besar dari nol. Algoritma akan memeriksa nilai *congestion* yang ada pada timur apakah lebih besar dari nilai *congestion* yang ada pada *port* utara. Apabila nilai *congestion* pada *port* timur lebih besar algoritma akan memeriksa apakah nilai *congestion* pada *port* timur lebih besar dari minimal *bandwidth* yang telah ditentukan. Jika benar maka

nilai dari *output_port_next* akan diubah menjadi E yang berarti *packet* akan dikirim melalui *port* timur. Apabila nilai *congestion* pada *port* timur memiliki *bandwidth* dibawah batas minimal *bandwidth* yang ditentukan, dan TTL yang dibawa pada *packet* belum mencapai nilai maximum, algoritma akan mengecek apakah nilai *congestion* pada *port* barat lebih besar dari nilai *congestion* yang ada pada *port* timur dan selatan, dan algoritma juga memastikan kalau *packet* tidak datang dari *port* barat, jika semua kondisi telah dipenuhi, nilai *output_port_next* akan diubah menjadi W, yang berarti *packet* akan dikirim melalui *port* barat. Namun apabila kondisi ada yang tidak dipenuhi algoritma akan mengecek nilai *congestion* pada *port* selatan apakah nilai *congestion* pada *port* selatan lebih besar dari nilai *congestion port* timur dan *packet* tidak datang dari *port* selatan. Apabila kondisi terpenuhi, *packet* akan dikirimkan melalui *port* selatan. Namun apabila ada satu kondisi yang tidak terpenuhi, *packet* akan dipaksa dikirim melalui *port* timur.

Ketika nilai *congestion* yang ada pada timur lebih kecil dari nilai *congestion* yang ada pada *port* utara, algoritma akan memeriksa apakah nilai *congestion* pada *port* utara lebih besar dari minimal *bandwidth* yang telah ditentukan. Jika benar maka nilai dari *output_port_next* akan diubah menjadi N yang berarti *packet* akan dikirim melalui *port* utara. Apabila nilai *congestion* pada *port* utara memiliki *bandwidth* dibawah batas minimal *bandwidth* yang ditentukan, dan TTL yang dibawa pada *packet* belum mencapai nilai maximum, algoritma akan mengecek apakah nilai *congestion* pada *port* barat lebih besar dari nilai *congestion* yang ada pada *port* utara dan selatan, dan algoritma juga memastikan kalau *packet* tidak datang dari *port* barat, jika semua kondisi telah dipenuhi, nilai *output_port_next* akan diubah menjadi W, yang berarti *packet* akan dikirim melalui *port* barat.

Namun apabila kondisi ada yang tidak dipenuhi algoritma akan mengecek nilai *congestion* pada *port* selatan apakah nilai *congestion* pada *port* selatan lebih besar dari nilai *congestion port* utara dan *packet* tidak datang dari *port* selatan. Apabila kondisi terpenuhi, *packet* akan dikirimkan melalui *port* selatan. Namun apabila ada satu kondisi yang tidak terpenuhi, *packet* akan dipaksa dikirim melalui *port* utara.

3.3 Perancangan Sistem

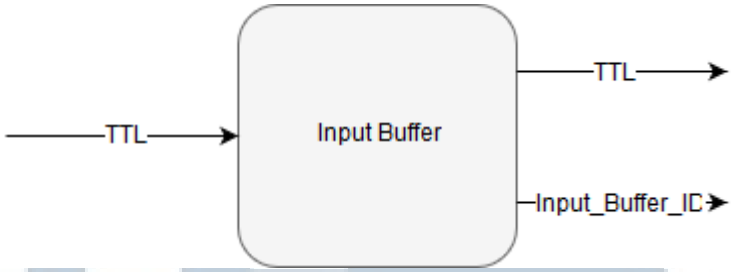
Pada penelitian ini, akan dimodifikasi beberapa *environment* dari UTAR NoC sesuai dengan kebutuhan dari algoritma *routing protocol* baru. Beberapa penambahan pada *environment* UTAR NoC antara lain adalah :

- Penambahan TTL pada setiap header *packet*
- Parameter baru pada *inputbuffer* untuk menandakan darimana *packet* berasal

3.3.1 Perancangan Register-Transfer Level

Pada Register-Transfer Level (RTL) UTAR, ditambahkan beberapa parameter untuk dilempar kedalam *algoritma routing*. Parameter ini akan digunakan sebagai aspek pemilihan rute oleh *algoritma routing* berbasis TTL. RTL yang ditambahkan parameternya ialah, *router, network,* serta *input buffer*.

Pada *router* ditambahkan beberapa parameter seperti indeks untuk mencari dimana letak bit TTL pada header *packet*, dan juga penambahan pemilihan untuk *input buffer* agar dapat dipilih. Parameter ini digunakan agar nilai dari TTL dapat dan juga *routing protocol* baru dapat dipilih untuk dijalankan pada simulasi. Pada *input buffer* juga dimasukan beberapa fungsi untuk menambahkan bit TTL pada *packet* dan memasukan kembali ke *packet*.



Gambar 3.6 Modified Input Buffer Block Diagram

Pada Gambar 3.6 diperlihatkan bahwa pada *input_buffer* yang telah dimodifikasi akan menerima input tambahan yaitu TTL. TTL tersebut akan nilainya ditambahkan dengan satu, dan apabila telah mencapai MAX_TTL, nilai pada TTL tidak akan ditambahkan dan hanya akan diteruskan ke *router*.

Pada Gambar 3.6 diperlihatkan juga bahwa ada *output input_buffer_ID*, yang menandakan darimana *packet* berasal. Pada file *input buffer* dimasukan parameter lokal yang menandakan darimana *packet* tersebut datang. Yang nantinya akan menjadi penilaian pada *routing protocol*. *Packet* yang datang dari suatu *input buffer* dari suatu port tidak boleh keluar ke port yang sama lagi. Pada Tabel 3-1 dijelaskan parameter lokal yang akan digunakan pada *input buffer*.

Tabel 3.1 Tabel Input Buffer

Input_Buffer_ID	Keterangan
0	PE
1	N
2	E
3	S
4	W

3.3.2 Perancangan UVC Compenent

Pada beberapa komponen UVC perlu ditambahkan untuk menambahkan bit pada *packet* yang akan dikirim, dan menyampaikan pada komponen monitor untuk bahwa ada bit tambahan. Semua komponen ini akan dijadikan satu menjadi satu

package, yang nanti akan digunakan oleh *testbench* dalam pembentukan *file test*, dan juga *source* packet pada simulasi.

. *Flits* dasar dari *router* yang digunakan dalam penelitian ini dapat dilihat pada Tabel 3-2 "*valid_bit*" digunakan sebagai penanda validasi dari *flit*. Nilai bit ini akan bernilai satu ketika mengirim *flit* baru. "is_tail" digunakan sebagai penanda *flit* tersebut adalah *flit* akhir atau bukan *flit* akhir. *Packet* besar dapat dipotong menjadi beberapa bagian *flit*, dan *flit* terakhir adalah *flit* yang membawa bit "is_tail" bernilai satu. Destinasi memberikan informasi dari alamat destinasi flit. Lebar dari bit ini tergantung dari ukuran jaringan. *Virtual channel* menandakan VC mana yang akan digunakan untuk mengirim *flit*. TTL memberikan informasi untuk *routing protocol* untuk meninggalkam Bit Source memberikan informasi dari alamat sumber flit. Bit terakhir adalah bit data. TTL awal ditetapkan 0 dan maximum dari nilai bit TTL field ini adalah 7. Maximum dari nilai bit TTL ditentukan sesuai dengan maximum hop yang mungkin dilalui oleh XY algoritma *routing* pada topologi *mesh* berukuran 4x4.

Tabel 3.2 Modified flit

valid_bit	is_tail	destination	virtual_channel	Time to Live	source	data
1-bit	1-bit	8-bit	1-bit	3-bit	8-bit	56-bit