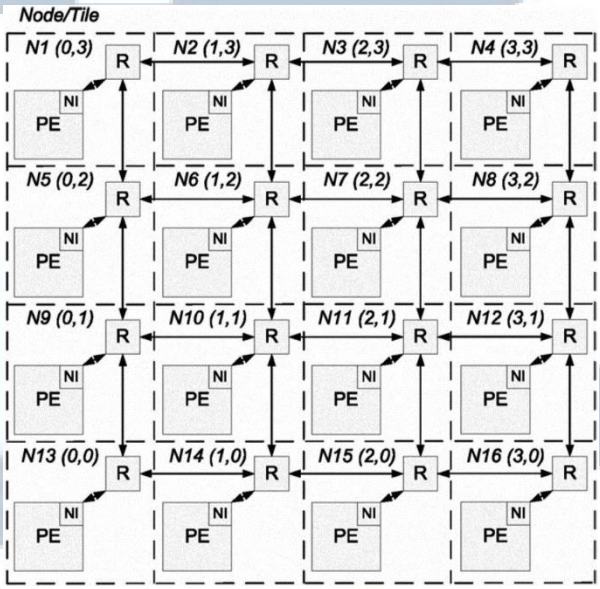
BAB II

LANDASAN TEORI

2.1 Konsep Network-on-Chip

Pada System-on-Chip yang berbasis NoC, komponen-komponen chip dibagi menjadi beberapa *Processing Block* (PB) [8]. Sebuah PB terdiri dari satu atau beberapa komponen yang membentuk sebuah subsistem. Semua PB saling terhubung dengan menggunakan router. Sebuah router dapat terhubung dengan satu atau beberapa router lainnya berdasarkan topologi dari jaringan. Gambar 2.1 merupakan contoh sebuah NoC dengan topologi Mesh 4x4.



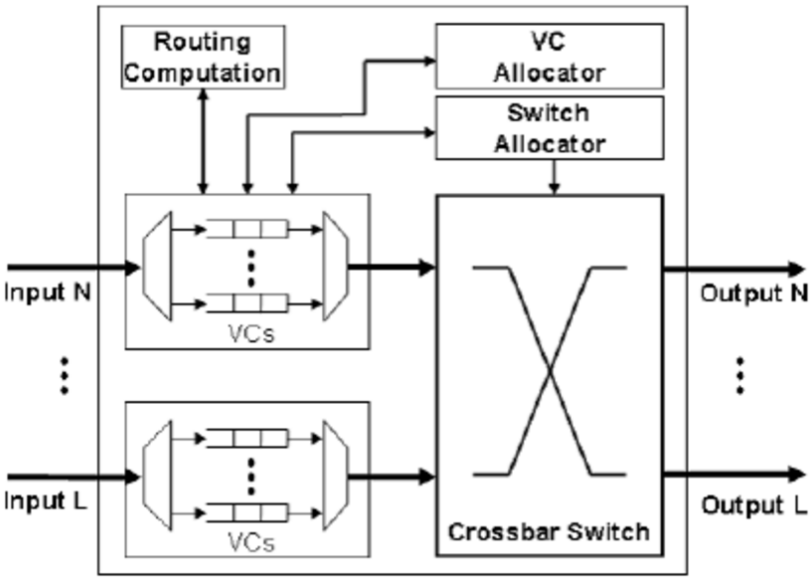
Gambar 2.1 NoC Topologi Mesh 4x4[13]

Semua router dalam jaringan memiliki sebuah *address* yang digunakan sebagai *unique ID*. Data dan informasi tambahan seperti *destination address* dan tipe data dibungkus dalam sebuah flit. Informasi tambahan tersebut terdapat pada header dan digunakan oleh router untuk menentukan apa yang harus dilakukan pada flit

tersebut. Setiap PB akan mengirim semua flit ke *destination address* melalui jaringan. Algoritma routing pada setiap router akan menentukan jalur selanjutnya yang akan dilalui oleh flit untuk mencapai tujuan. Desain dari NoC dapat dibagi menjadi beberapa aspek seperti topologi, routing, *flow control*, dan mikroarsitektur router. Topologi menentukan gambaran fisik dan koneksi antar node pada jaringan. Rancangan topologi memiliki peran penting dalam menentukan algoritma routing dan pemetaan aplikasi [3]. Rancangan topologi berawal dari topologi jaringan yang umum digunakan, seperti tree, ring, star, atau mesh. Optimasi dan fitur tambahan kemudian diterapkan untuk mengembangkan topologi umum ini.

Algoritma routing menentukan jalur yang akan dilalui sebuah paket untuk mencapai tujuan. Kemampuan sebuah algoritma routing untuk mengatur *traffic* memiliki dampak langsung pada *throughput* dan performa jaringan. *Flow control* menentukan bagaimana *resource* dialokasikan untuk paket saat paket bergerak dalam jaringan.

Mikroarsitektur router memiliki beberapa komponen, yaitu input buffer, router state, routing logic, allocator, dan crossbar. Mikroarsitektur router mempengaruhi performa, biaya, dan efisiensi sebuah router, dimana umumnya peningkatan sebuah parameter akan mengurangi parameter lainnya. Gambar 2.2 merupakan contoh mikroarsitektur dari sebuah router.



Gambar 2.2 Mikroarsitektur Router[14]

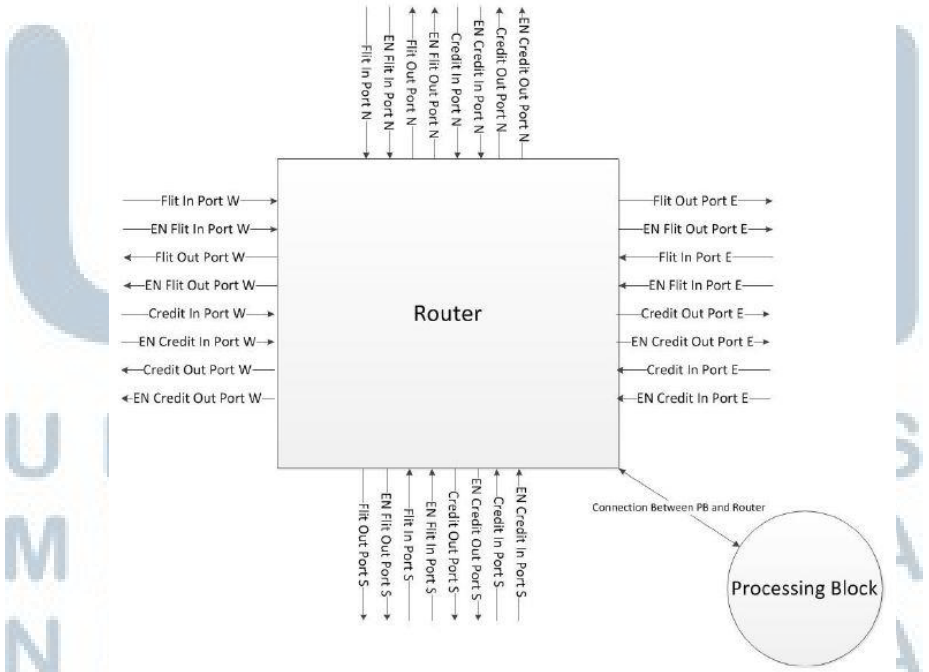
Pemodelan dan optimasi aplikasi merupakan tantangan lain dalam sebuah MPSoC berbasis NoC [3]. Setiap aplikasi memiliki *communication traffic pattern* yang berbeda, bahkan untuk aplikasi yang sama, pemetaan aplikasi yang berbeda akan berdampak pada *traffic* jaringan secara signifikan. Pemahaman pada *traffic pattern* dan *system requirement* sangat membantu dalam membuat topologi jaringan yang optimal, juga memiliki dampak yang besar pada biaya, daya, dan performa.

Pemodelan dan pemetaan aplikasi yang tepat dapat membantu untuk menentukan rancangan paradigma komunikasi [9]. NoC biasanya dirancang untuk aplikasi spesifik. Tidak ada NoC yang memberikan performa optimal pada banyak aplikasi sekaligus [10].

2.2 Router UTAR NoC

UTAR NoC terdiri atas sejumlah router UTAR NoC, dimana jumlahnya berdasarkan pada besar dari jaringannya [4]. Router UTAR NoC memiliki 5 buah port, yaitu North, East, South, West, dan PB. Pada Gambar 2.3, setiap port pada router memiliki 8 buah pin, yaitu:

1. flit_in, untuk flit yang datang dari router tetangga atau PB.
2. EN_flit_in, untuk sinyal enable dari flit_in.
3. flit_out, untuk flit yang keluar dari router ke router tetangga.
4. EN_flit_out, untuk sinyal enable dari flit_out.
5. credit_in, untuk credit yang datang dari router tetangga atau PB.
6. EN_credit_in, untuk sinyal enable dari credit_in.
7. credit_out, untuk credit yang keluar dari router ke router tetangga.
8. EN_credit_out, untuk sinyal enable dari credit_out.



Gambar 2.3 Router UTAR NoC [4]

Flit_in dan flit_out memiliki panjang 75 bit. Credit_in dan credit_out memiliki panjang 2 bit. Seluruh sinyal enable memiliki panjang 1 bit. Flit memiliki sebuah *field* standar, yaitu:

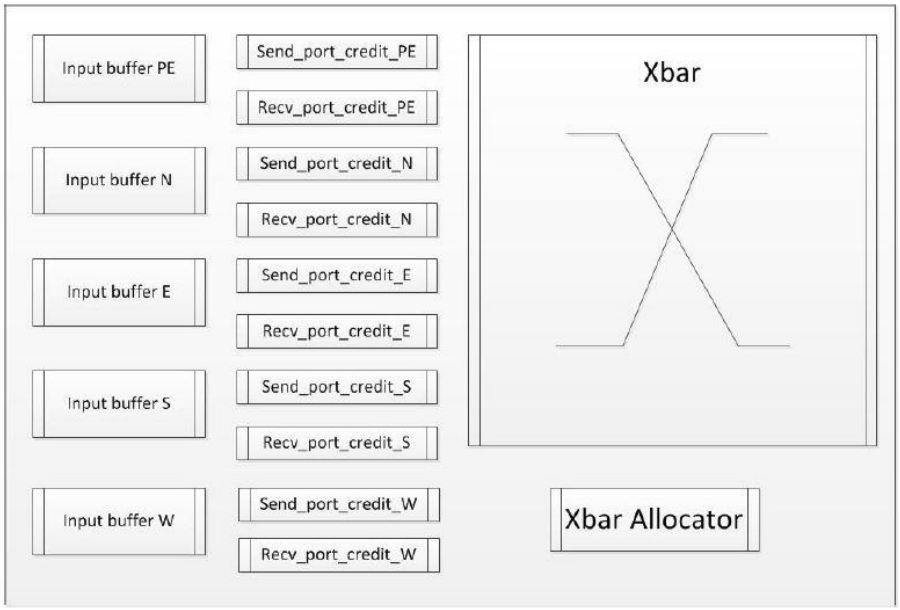
1. valid_bit, menentukan validitas sebuah flit.
2. is_tail, menentukan apakah sebuah flit adalah tail flit atau bukan. Paket yang besar dapat dipecah menjadi beberapa flit dan tail flit adalah flit terakhir dari grup.
3. Destination, berisi informasi address tujuan dari flit.
4. Virtual channel, menunjukkan VC mana yang akan digunakan untuk mengirim flit.
5. Source, berisi informasi address dari pengirim flit.
6. Data.

Gambar 2.4 Flit field dari UTAR NoC[4]

Sinyal yang menghubungkan antar router dapat dikelompokkan menjadi dua port, yaitu Send port dan Receive port. Send port digunakan untuk mengirim flit dan menerima credit, sedangkan Receive port digunakan untuk menerima flit dan mengirim credit.

Router UTAR NoC memiliki 5 input port dan output port sesuai dengan 4 arah port router tetangga dan PB lokal. Komponen utama yang mendasari router adalah input buffer, route computation logic, virtual channel allocator, switch allocator, credit mechanism, dan crossbar switch. Input buffer berperan untuk menampung flit yang masuk pada input port. Route computation logic berperan untuk

menentukan jalur mana yang harus dilalui flit dari source sampai destination. Allocator berperan untuk menentukan flit mana yang diberi akses untuk melewati crossbar. Credit mechanism berperan sebagai flow control. Crossbar switch berperan untuk memindahkan flit dari input port ke output port. Gambar 2.5 merupakan mikroarsitektur dari router UTAR NoC.



Gambar 2.5 Mikroarsitektur Router UTAR NoC[4]

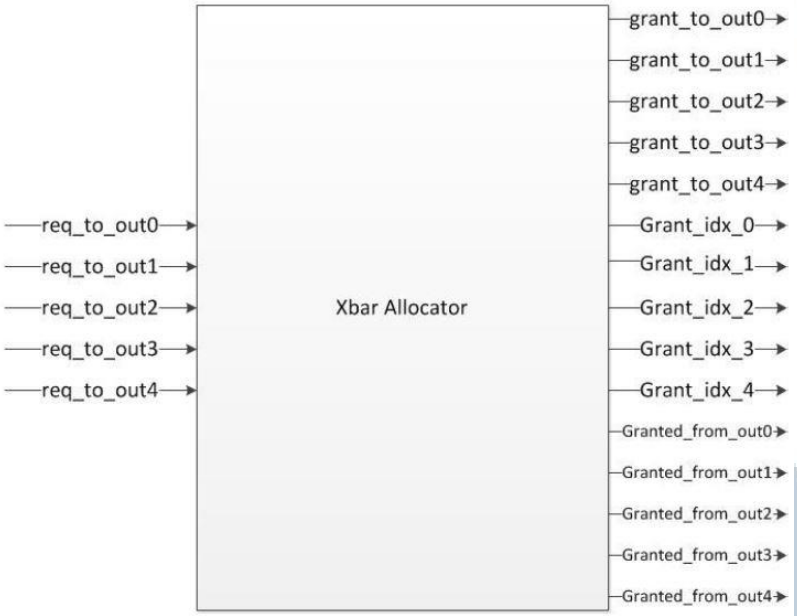
2.3 Arbiter UTAR NoC

Arbiter adalah sebuah *logic* yang mengatur beberapa agent untuk mengakses sebuah *shared resources.* Proses koordinasi ini dinamakan arbitrasi. Pada router, *shared resources* yang dimaksud adalah Virtual Channel (VC) dan *crossbar switch ports*.

Arbiter pada router UTAR NoC menggunakan algoritma round robin, dimana setiap agent diberi prioritas yang sama untuk mengakses *shared resources* [4].

Algoritma ini juga memastikan bahwa setiap *request* dari agent pasti akan diberi *grant*. Algoritma round robin mudah diimplementasi sekaligus memberikan *strong fairness* pada router.

Crossbar allocator UTAR NoC terdiri dari 5 buah arbiter untuk 5 output crossbar. Sinyal *request* pada crossbar allocator berasal dari *input buffer*, dan *grant* dari crossbar allocator akan diberikan kembali ke *input buffer* yang diberi *grant*. Diagram blok dari crossbar allocator dapat dilihat pada Gambar 2.6.



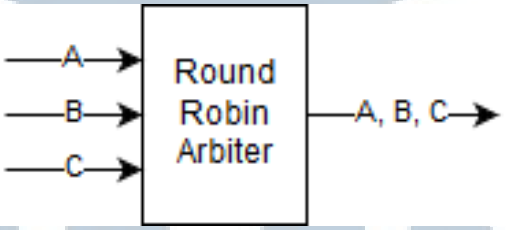
Gambar 2.6 Diagram Blok Crossbar Allocator [4]

2.4 Perbandingan Distance-based Priority dengan Round Robin

Algoritma *distance-based priority* merupakan algoritma yang menyerupai algoritma *age-based priority*, hanya saja algoritma ini menggunakan jumlah hop sebagai parameter untuk menghitung umurnya. Algoritma *distance-based* lebih sederhana karena nilai dari jumlah hop dapat diperoleh dari *field* yang sudah ada di

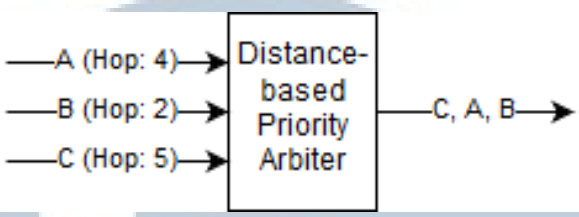
dalam sebuah paket, yaitu *source address*. Berbeda dengan algoritma age-based yang harus menambahkan *field* "age" untuk mencatat umur dari paket. Tentunya, penambahan *field* "age" ini akan menambah ukuran dari paket sekaligus menyebabkan implementasi algoritma ini menjadi lebih kompleks [6].

UTAR NoC menggunakan algoritma round robin untuk proses arbitrasinya. Round robin memberikan local fairness, dimana setiap request paket pada sebuah node akan dilayani secara bergiliran, tidak peduli dari mana source paket tersebut. Namun, hal ini dapat meningkatkan delay network secara keseluruhan, karena paket yang letaknya jauh harus tetap menunggu gilirannya pada tiap node yang dilewatinya [6]. Diagram alir dapat dilihat pada Gambar 3.6 di Bab III. Gambar 2.7 menunjukkan bagaimana algoritma round robin memberikan akses.



Gambar 2.7 Algoritma Round Robin

Algoritma *distance-based* menggunakan jumlah hop sebagai prioritas dari paket dalam menentukan akses. Jumlah hop didapat dengan menghitung selisih *current address* dan *source address*. Paket dengan jumlah hop paling besar akan diberikan akses terlebih dahulu. Diagram alir dari algoritma *distance-based* akan dijelaskan pada Bab III. Gambar 2.8 menunjukkan bagaimana algoritma *distance-based* memberikan akses.

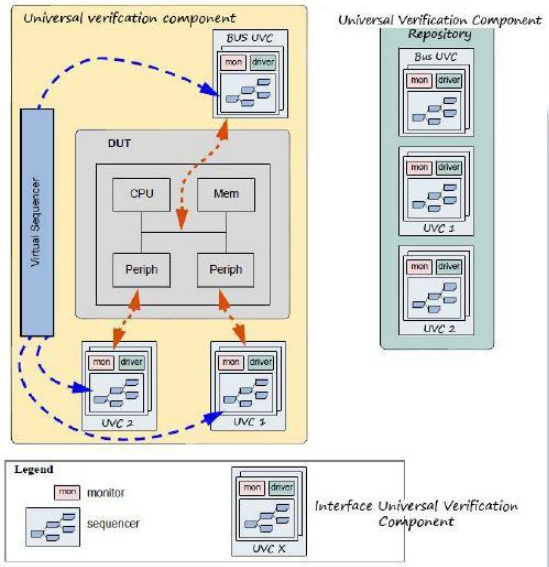


Gambar 2.8 Algoritma Distance-based

2.5 Universal Verification Methodology (UVM)

Universal Verification Methodology (UVM) adalah sebuah metodologi standar untuk melakukan verifikasi desain *integrated circuit* [11]. UVM sebagian besar berasal dari OVM (Open Verification Methodology). UVM dibuat berdasarkan SystemC dan System Verilog Object-Oriented Programming (OOP). Komponen UVM menggunakan *transaction level modelling* (TLM) untuk komunikasi antar objek. Sebuah testbench UVM terbuat dari *verification environment* yang dapat digunakan kembali yang dinamakan Universal Verification Components (UVC). UVC adalah *verification environment* yang terenkapsulasi, siap dipakai, dan dapat dikonfigurasi untuk *interface protocol*, *design submodule*, atau keseluruhan sistem.

Setiap UVC memiliki arsitektur yang konsisten dan terdiri atas sebuah kumpulan elemen yang lengkap untuk simulasi, pengecekan, dan mengumpulkan informasi pada protokol atau desain tertentu. UVC diterapkan pada *device under test* (DUT) untuk memverifikasi implementasi dari sebuah protokol atau arsitektur desain. Gambar 2.9 merupakan contoh sebuah *verification environment* dengan 3 interface UVC.



Gambar 2.9 Contoh Verification Environment [11]

2.6 MCSL NoC Traffic Pattern Suite

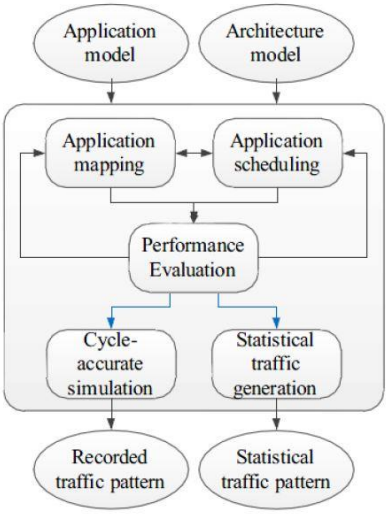
NoC Traffic Pattern adalah sebuah *tool* untuk melakukan *benchmark* pada NoC. NoC Traffic Pattern terdiri dari random traffic pattern dan realistic traffic pattern [12]. Random traffic pattern menggunakan distribusi probabilitas untuk mengacak komunikasi traffic, seperti *packet destination* dan *transmission interval*. Untuk mengatur parameter pada random traffic diperlukan pengetahuan yang cukup memadai tentang realistic traffic pattern yang sesuai dan secara umum sulit untuk mencerminkan karakteristik realistic traffic. Di sisi lain, realistic traffic pattern didasarkan pada aplikasi nyata yang dapat memberi hasil performa dan konsumsi daya yang lebih akurat, dan dapat memberi informasi yang lebih komprehensif untuk meningkatkan arsitektur NoC.

MCSL NoC traffic pattern suite adalah sebuah realistic traffic pattern, dimana di dalamnya terdapat 8 aplikasi *real-world* dan mencakup arsitektur NoC yang populer. MCSL NoC traffic pattern suite merekam perilaku komunikasi NoC dan

keterkaitan di antaranya. Setiap traffic pattern pada MCSL terdiri dari 2 jenis, *recorded traffic pattern* (RTP) dan *statistical traffic pattern* (STP). RTP menyediakan rekaman komunikasi secara detail untuk studi NoC yang komprehensif. STP lebih difokuskan untuk mempercepat eksplorasi NoC dengan mengurangi akurasinya.

Metodologi penghasilan traffic menggunakan *formal computational models* untuk merekam komunikasi dan kebutuhan komputasi dari aplikasi. Model ini mengoptimasi kebutuhan memori aplikasi, pemetaan, dan penjadwalan untuk memaksimalkan keseluruhan performa sistem dan utilisasi sebelum diubah menjadi traffic pattern melalui *cycle-level simulations*.

Proses penghasilan dimulai dengan *application model* dan *architecture model*. Traffic pattern dihasilkan melalui 5 tahap, yaitu pemetaan aplikasi, penjadwalan aplikasi, evaluasi performa pada hasil hasil pemetaaan dan penjadwalan aplikasi, *cycle-accurate simulations*, dan *statistical traffic generations*. Gambar 2.10 menunjukkan metodologi dari penghasilan traffic.



Gambar 2.10 Metodologi Penghasilan Traffic [15]