Technische Universität Dresden

Advanced Connection Allocation Techniques in Circuit Switching Network on Chip

Yong Chen
Born in Anhui, China, 10 June 1989

von der Fakultät Elektrotechnik und Informationstechnik
der Technischen Universität Dresden

zur Erlangung des akademischen Grades

Doktoringenieur

(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Prof. Dr.-Ing. Christian G. Mayr
Gutachter:	Prof. Dr.-Ing. Gerhard P. Fettweis
	Prof. Dr.-Ing. Holger Blume
	Prof. Dr. Ing. Ralf Lehnert

Tag der Einreichung: 29.06.2017
Tag der Verteidigung: 08.09.2017

Abstract

With the advancement of semiconductor technology, the System on Chip (SoC) is becoming more and more complex, so the on-chip communication has become a bottleneck of SoC Design. Since the traditional bus system is inefficient and not scalable, the Network-On-Chip (NoC) has emerged as the promising communication mechanism for complex SoCs. As some systems have specific performance requirements, such as a minimum throughput (for real-time streaming data) or bounded latency (for interrupts, process synchronization, etc), communication with Guaranteed Service (GS) support becomes crucial for predictable SoC architectures. Circuit Switching (CS) is a popular approach to support GS, which firstly has to allocate an exclusively connection (circuit) between the source and destination nodes, and then the data packets are delivered over this connection. However, it is inefficient and inflexible because the resource is occupied by single connection during its whole lifetime, which can block other communications. Hence, two extensions of CS have been proposed to share resources: i) Time-Division Multiplexing (TDM), in which the available link capacity is split into multiple time slots to be shared by different flows in TDM scheme; and ii) Space-Division-Multiplexing (SDM), in which only a subset (sub-channel) of the link wires is exclusively allocated to a specific connection, while the remaining wires of the link can be used by other flows.

The connection allocation is critical for CS, since the data delivery can start only after the associated connection is allocated. In this thesis, we propose a dedicated hardware connection allocator to solve the dynamic connection allocation problem for CS NoCs, which has to i) allocate a contention-free path between source-destination pairs and ii) allocate appropriate portions of link bandwidth (appropriate number of time slots and subsets) along the path. The dedicated connection allocator, called NoCManager, solves the connection allocation problem by employing a trellis-search based shortest path algorithm. The trellis search can explore all possible paths between source node and destination. Moreover, it shall find the requested path in a fixed low latency and can guarantee the path optimality in terms of path length if the path is available.

In this thesis, two different trellis graphs, *Forward-Backtrack trellis* and *Register-Exchange trellis* are proposed. The Forward-Backtrack trellis completes the path search in two steps: forward search and backtracking. Firstly, the forward search begins at source node that traverses the network to find the free path. When destination node is reached, the backtrack starts from destination to select the survivor path and collect the associated path parameters. However, Register-Exchange trellis saves the entire survivor path sequences during forward search. Consequently, the backtracking step can be omitted, and thus the allocation time is halved compared to forward-backtrack approaches. Moreover, each trellis graph consists of three categories, *unfolded structure*, *folded structure* and *bidirectional structure*. The unfolded structure can provide high allocation speed while folded structure is more efficient from a hardware point of view. The bidirectional structure starts the search at two sides, source node and destination node simultaneously, so the allocation speed is 2 times faster than previous unidirectional search. Furthermore, in order to address the scalability issue of previous centralized systems, the partitioned architecture (i.e. spatial partitioning technique) is proposed to divide the large system into multiple smaller differentiated logical partitions served by local NoCManagers. This partitioning technique keeps the request load of the manager and manager-node communication overhead moderate. Inside each partition, the path search problem is solved by a local manager with trellis-search algorithm. To establish a path that crosses partitions, the managers communicate with each other in distributed manner to converge the global path.

In order to further enhance the path diversity and resource utilization, we adopt the combined TDM and SDM technique. In combined TDM-SDM approach, each SDM sub-channel is split into multiple time slots so that can be shared by multiple flows. Hence, the number of sub-channels can be kept moderate to reduce router complexity, while still providing higher path diversity than TDM scheme. In order to investigate and optimize TDM-SDM partitioning strategy, we studied the influence of different TDM-SDM link partitioning strategies on success rate and path length that allowed us to find the optimal solution. The dedicated connection allocator using the trellis-search algorithm is employed for TDM, SDM and TDM-SDM CS.

In the end, we present the router architecture that combines the circuit-switching network (for GS communication) and packet-switching network (for best-effort communication).

Acknowledgment

This thesis comprises of the results generated during my work at the Vodafone Chair Mobile Communications Systems at Technische Universität Dresden between 2013 and 2017.

First of all I would like to thank my supervisor Prof. Gerhard Fettweis for giving me the opportunity to join his group and inspiring me with his invaluable guidance. I cannot be any less grateful to have Prof. Blume as my co-supervisor whose guidance and mentoring had made this journey possible. I also thank my group leader Dr. Emil Matus. Emil is a very good person. He teaches me how to do research, and helps me go through the difficult times. Additionally, I am grateful to my colleagues, Sadia Moriam, Seungseok Nam and Mohammed Radi, for reviewing the thesis.

I would like to thank all colleagues from the chair. I would like to thank Friedrich, Sebastian, Wen, Stefan, Robert, Mattis, Amanda, Song, Zou and Zhang for all the joys and fun shared with me.

Finally, I am grateful to my parents, and my wife Aihong, for their faith in me. Your support, encouragement and unwavering love made me able to finish the thesis.

Yong Chen
Dresden, Germany, June 2017

Contents

List of Symbols

$\mathbf{S}$	Slot table size
$\mathbf{T}$	the total link capacity
$\mathbf{t}$	a specific time slot
$\mathbf{R}$	required bandwidth
$\mathbf{H}$	route time over single hop
$\mathbf{m}$	the path length from source to destination
$\mathbf{B}$	branch metric
$\mathbf{a}$	the available slots of the branch
$\mathbf{r}$	the number of requested slots
$\mathbf{w}$	the weight of the branch
$\mathbf{P}$	path metric
$\mathbf{S_j}$	the set of states that have transitions to state j
$\mathbf{l}$	the distance between source and destination
$\mathbf{d}$	the number of allowed detours
$\mathbf{M}$	the number of routers of the whole network
$\mathbf{N}$	the side length of the network, hence the network is $N \cdot N$
T_{eff}	the effective time
E	the error rate
$\mathbf{C}$	the number of sub-channels

List of Abbreviations

ATM	Asynchronous Transfer Mode
Ans	Answer signal
Ack	Acknowledgment
AT	Area.Time product
AT/S	Area.Time/Success Rate measure
BE	Best Effort
bk	background traffic
CS	Circuit Switching
Des	destination
DSS	Detect-Select-Shift
DS	Detect-Select
EoP	End-Of-Packet
FB TESSA	Forward-Backtrack TrElliS-Search based Allocation
GS	Guaranteed Services
HAGAR	HArdware Graph ARray
HPU	Header Parsing Unit
IP	Intellectual Property
N	Node
NoC	Network-on-Chip
NoCM	NoCManager
NI	Network Interface
PE	Processing Element
RE TESSA	Register-Exchange TrElliS-Search based Allocation
R	Router
SDM	Space-Division-Multiplexing
SoC	System on Chip
Src	source
TDM	Time-Division Multiplexing
TESSA	TrElliS-Search based Allocation
2D	2 dimension

List of Figures

List of Tables

Chapter 1

Introduction

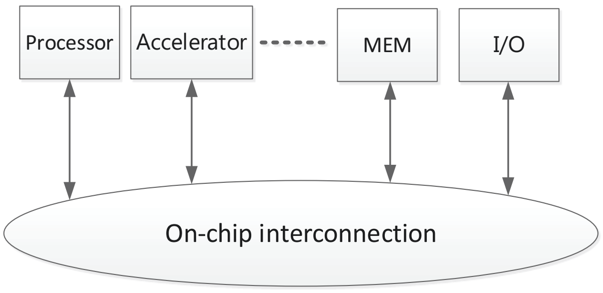
Due to the advances of semiconductor technology, the transistor size is decreasing while the die size is increasing. As a result, nowadays, a single chip can contain billions of transistors, so more devices and Intellectual Property (IP) cores can be integrated on a single chip. Consequently, the on-chip interconnection (communication) problem, i.e. how to connect the on-chip modules, is becoming a critical issue for complex System on Chip (SoC) design [WL03].

1.1 Overview of on-chip interconnection solutions

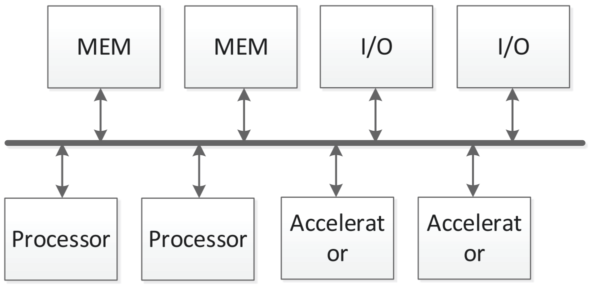
The on-chip interconnection is used to connect the modules of the on-chip system, to connect e.g. processors, memories, and peripherals. A typical interconnection diagram is illustrated in Fig. 1.1a.

The **bus** is a widely used interconnect, which is simple and cost effective with hardware complexity O(n) and performance scalable typically for a small number of nodes. All the modules on a chip are connected to a shared bus, while only one master module can access it at a time, as depicted in Fig. 1.1b. The data sent from a module is broadcasted to all the other modules. This induces a contention among competing modules resolved through arbitration, to determine who gains access to the bus at a time. The problem of the bus-based system is the lack of scalability. When the number of the connected modules increases, the average bandwidth per module decreases while the contention rate increases. Besides this, as the wire length and capacitance increases, the wiring delay increases, which reduces the operating frequency.

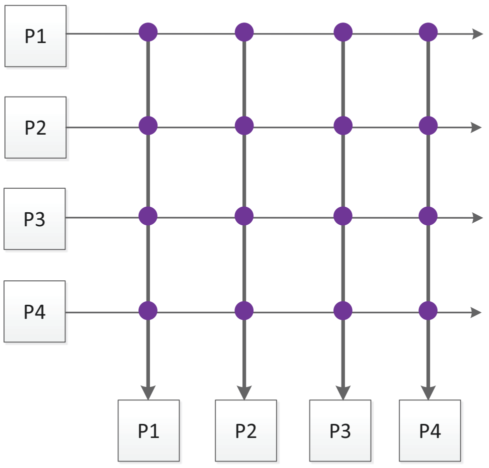
In order to enable simultaneous communications among multiple modules, the **crossbar switch** is proposed, as depicted in Fig. 1.1c. In crossbar, every module is connected to every other modules. The communication bandwidth is enhanced since multiple communications can be supported simultaneously as long as they occur between different modules. Though crossbar can provide low latency and high throughput, it is expensive.



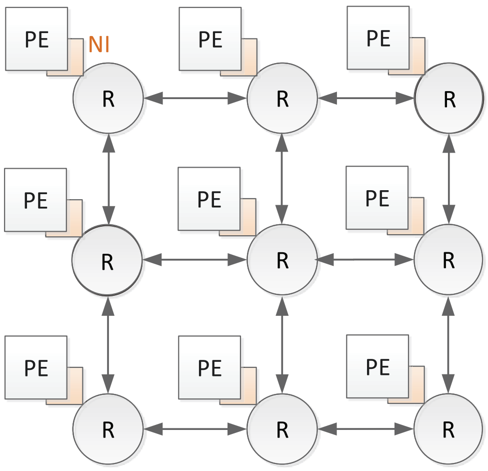
(a) On-chip interconnection diagram



(b) Bus



(c) Crossbar



(d) Mesh NoC

Figure 1.1: Interconnection solutions.

The hardware complexity of crossbar scales with O(n^2), where n is the number of the connected modules. Hence, it is good for small number of modules, but too expensive for large system.

In order to address the aforementioned problems, the advanced interconnection solution, **Network-On-Chips (NoC)** is proposed [DT04, BDM02, GHKM11]. The NoC borrows

Table 1.1: The interconnections advantages and disadvantages

	Bus	Crossbar	NoC
scaling	10	10- 100	10+
wire cost	low	high	average
logic/buff cost	low	high	average
throughput	low	highest	high
energy efficiency	poor	average	good

ideas from large-scale communication networks, e.g. Internet or Asynchronous Transfer Mode (ATM) network, to create a scalable on-chip communication network. The Processing Element (PE), or module, is connected to a router via Network Interface (NI). Data transfer in point-to-point communication is controlled and relayed by routers. Hence, the long wires between a source node and a destination are avoided, and the data packet may take multiple hops through the network to reach the destination. The popular NoC topology, a mesh, is depicted in Fig. 1.1d. Since each router just needs to connect the adjacent routers, the wire length is short. Concurrent communications is possible as long as they are delivered over different links. Since a mesh has a regular structure and equal-length links, it is easy to layout. Moreover, it has high path diversity so that there are many alternate paths possible from one node to another. Therefore, the NoC can provide large communication bandwidth, low wiring complexity and good scalability. The advantages and disadvantages of the aforementioned interconnections is shown in table 1.1 [Ste12].

1.2 Network on Chip

There is some terminology for NoC, which is:

1. **Router** connects fixed number of input links to fixed number of output links, which forwards the incoming packet to the specific output according to predefined routing policy.

2. **Network Interface (NI)** connects the PE to network and decouples communication.

3. **Link** is a bundle of wires that carries communication/data signals between two adjacent routers.

4. **Channel** is a single logical connection between two adjacent routers.

5. **Network node** is a logical abstraction of a router within a network.

6. **Traffic flow** is a sequence of packets from a source node to a destination.

7. **Message** is the transfer entity from the network's clients to the network.
8. **Packet**: A single message can be split into one or more packets, which represents transport element of network.
9. **Flit** i.e. FLow control unIT, is the elementary data unit of the flow control mechanism. Each packet is split into one or more fixed-size flits.
10. **Phit** is the physical digit, which is the data transmitted per link per cycle.

The basic characteristics of NoC are:

1. **Topology**: The network topology is the pattern in which network nodes are connected via links, i.e. the arrangement of these nodes and links. A good topology should meet the bandwidth and throughput requirements of the system at a minimum cost. The path diversity of a topology also determines the performance under adversarial traffic and fault tolerance. The most popular network topology is the mesh, which has regular structure and equal-length links, so it is easy to layout, as shown in Fig. 1.1d.
2. **Routing** is the process of selecting a path for traffic from source node to destination node. Once a topology is selected, the routing algorithm determines the system throughput. A good routing algorithm should be deadlock-free, can balance traffic load, and typically keep the path lengths as short as possible.
3. **Flow control** dictates which messages get access to particular network resources over time. It manages the allocation of channel bandwidth and buffer capacity to packets along the path from source to destination. A good flow control can minimize packets delivery latency, and can avoid resource conflicts and buffer overflow.
4. **Switching model**: Generally speaking, the NoC can be grouped into two categories: **packet switching** and **circuit switching** networks. In packet switching networks, the communication resources, e.g. link and buffers, are allocated at each node for each packet. When a packet reaches a node, the resources are allocated. When the packet leaves, the resources are released. In contrast, circuit switching networks allocate exclusively channels to form a circuit (connection) from source node to destination, and then the data packets are delivered over this connection, as depicted in Fig. 1.2. The comparison between Circuit Switching and Packet Switching networks is listed in table 1.2.
5. **Service class**: The traffic can be divided into two broad categories: **guaranteed service classes** and **best efforts (BE) classes** . The guaranteed service classes provide some minimum level of performance, such as a guaranteed loss rate, throughput, latency, and jitter. In contrast, the network provides no hard guarantees for best-effort classes.

Table 1.2: Comparison between Circuit Switching vs. Packet Switching networks

	Circuit Switching	Packet Switching
Orientation	Connection oriented	Connectionless
Connection Established	Yes	No
Resource Allocation	Resources are allocated before data transfer.	Not required.
Communication Reliability	High	Unreliable
Bandwidth	Fixed	Dynamic
Delay overheads	Connection Allocation delay.	Packet transmission delay.
Data transmission delay	Low	Unpredictable
Flexibility	Inflexible	Flexible

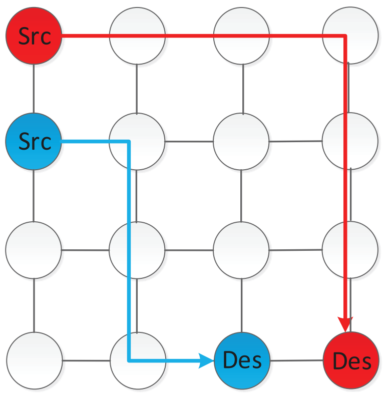


Figure 1.2: Circuit Switching network. Each flow has its own dedicated point-to-point connection to transfer data. Src: source node, Des: destination node.

1.3 Guaranteed Service in NoCs

In modern complex SoCs, many applications have specific requirements of the performance, such as a minimum throughput (for real-time streaming data), bounded latency (for interrupts, process synchronization, etc). Therefore, providing Guaranteed Services (GSs), e.g. to guarantee bounded latency, minimum bandwidth and low (or no) data loss, is crucial for predictable SoC architectures[SMG14]. In general, there are two popular approaches to provide GS in NoCs: i) the connection-less approach employing priority scheduling in packet switching networks and ii) the connection-oriented method based on Circuit Switching (CS) network. Priority scheduling assigns higher priority to the latency-sensitive flows, as in e.g. QNoC[BCGK04], artNoC[SLB07] and Quota-setting NoC[CTSM13]. This approach however suffers from reduced determinism and predictability due to the contention of multiple flows on shared resources. In contrast, the CS allocates exclusive channels to form a circuit (connection) to the particular flow, so it can provide hard GS, e.g. used in MANGO[BS05]. After the connection is established, the requested

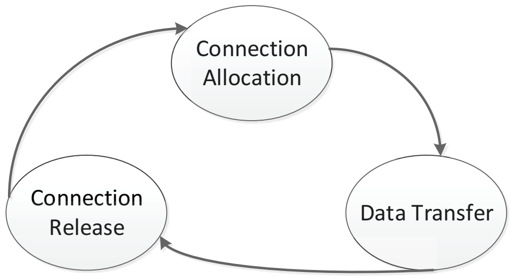


Figure 1.3: General operating procedure of Circuit Switching.

bandwidth (link capacity) and bounded/constant end-to-end latency are guaranteed. In order to achieve hard GS, in this thesis, we focus on CS approach only.

1.3.1 Circuit Switching NoCs

The general operating procedure of CS comprises three phases: connection allocation (setup), data transfer and connection release, as depicted in Fig. 1.3. In CS, when a source node wants to send data to a destination, a connection between the source node and destination should be allocated first. Thereafter, the data packets are delivered over this connection. When the data transfer is finished, the connection will be released.

In CS, since the resource is exclusively occupied during the entire lifetime of a connection, it may lead to inefficiencies and in-flexibilities for the system due to the blocking of other traffic flows. Hence, there are two extensions to share the resource (links) among multiple flows: 1) Time-Division Multiplexing (TDM) CS and 2) Space-Division-Multiplexing (SDM) CS.

In TDM CS, the link capacity is split into multiple time slots, and the link is allocated exclusively to a flow only in specific time slots, while the other time slots can be used by other flows, used in e.g. Nostrum[MNTJ04], AEthereal[GDR05, GH10], parallel probe NoC[LJL14b].

In SDM CS[LJL15, EJ13, LMV$^+$08, RRRM08], the link wires are physically split into subsets (sub-channels), and only subset of the link wires is exclusively allocated to a given connection, while the remaining wires of the link can be used by other flows.

1.4 Scope and Outline of this Work

CS is frequently adopted for providing GSs in NoCs. In order to share the resource (links), two extensions have been proposed: TDM CS and SDM CS. Since the data can be transferred only after the connection is allocated, the connection allocation is critical to CS

NoCs. In this thesis, we focus on the connection allocation problem, which has to i) allocate a contention-free path between source-destination pairs and ii) allocate appropriate portions of link bandwidth (appropriate number of time slots and subsets) along the path.

The dissertation is structured as follows.

- In Chapter 2, we first explain what is the problem of connection allocation in CS NoCs, and then present the overview of related work. At the end, we present the scheme of our allocation approach which is a dedicated allocator (i.e. NoCManager) based centralized allocation technique.
- Chapter 3 starts by describing the problem of the TDM CS connection allocation, and then presents the architecture of the dedicated allocation unit, NoC-Manager, which employs the dynamic programming to solve the connection allocation problem as a trellis path search algorithm, which can solve the shortest path search problem efficiently. The trellis graph consists of two categories: Forward-Backtrack trellis[CMF16a, CMF16b] and Register-Exchange trellis[CMF17d]. In Forward-Backtrack trellis, the path search is divided into two steps: forward search to try to reach the destination, and backtrack from destination to select the survivor path. However, in Register-Exchange trellis, we only need to do forward search and the backtrack step can be omitted. As soon as the forward search is finished, we can get the survivor path from the destination node immediately. And thereby compared to Forward-Backtrack approach, the search time is halved. Moreover, in both approaches, the trellis search can be further grouped into three categories, *unfolded structure*[CMF16a], *folded structure*[CMF16b] and *bidirectional structure*. The unfolded structure constructs the trellis graph as multiple stages to represent multi-hop traversal through the network, but the folded structure only implements one stage and reuses this stage to do path search. The folded structure is more efficient for area but the cost for clock cycles is increased to complete single allocation, as one cycle for traversing single stage. The bidirectional structure starts the search at two sides, source node and target node simultaneously, so compared to the traditional search that starts only at the source node, the path search time is halved. The synthesis results of area, Area.Time product and energy consumption per allocation of different trellis structures are presented and compared. The allocation time and allocation success rate of trellis search are compared to previous centralized and distributed allocation approaches. Finally, in order to address the scalability issue, the partitioning structure[CMF17a] that divides the large system into multiple partitions with multiple local managers is proposed.
- Chapter 4 starts by giving introduction of the SDM CS. In SDM CS, since there is no time slot scheduling constraint, and any free sub-channel at the next hop can be allocated, it can provide higher path diversity than TDM CS. However, the area cost of SDM switch scales as quadratic with the number of the sub-channels.

Moreover, the number of sub-channels is limited by the number of wires, so it cannot be increased arbitrarily. Hence, we present the combined TDM and SDM CS technique[CMF17c], in which each sub-channel is further split into time slots. The trellis path search based NoCManager is employed for the connection allocation of combined TDM and SDM CS. We also studied the influence of different link partitioning strategies with fixed link wires, i.e. the effect of splitting the link into different number of time slots and sub-channels, on allocation success rate and average path length.

- In Chapter 5, we present the router architecture[CMF17b] that combines the circuit-switching network (for GS communication) and packet-switching network (for BE communication).
- Finally, we draw conclusions and provide some future research directions in Chapter 6.

Chapter 2

Connection Allocation in CS NoCs

This section first explains what is the problem of the connection allocation in CS NoCs, and then presents the overview of known methods and approaches (related work). Finally, we introduce the idea of our allocation approach that bases on a centralized dedicated allocator (i.e. NoCManager).

2.1 Connection allocation problem

Some real-time applications have strict time requirements that the data delivery across the NoC must be in time, so it would be preferred to reserve dedicated transfer resources for the communication i.e. perform connection allocation. In CS, the connection allocation has to i) allocate a contention-free path from source node to destination node and ii) allocate appropriate portions of link bandwidth (appropriate number of time slots in TDM CS and subsets in SDM CS) along the path. Assume the total link capacity is T, and the link is split into p portions (number of slots/channels). If an application requires R bandwidth, then $R \div (\frac{T}{p}) = \left\lceil \frac{R \cdot p}{T} \right\rceil$ slots/channels will be assigned to it. If the path length from source to destination is m hops, and it takes time of H to route over single hop, then the packet delivery latency through NoC will be $m \cdot H$.

The connection allocation is critical to circuit switching since the data transfer relies on the allocated connections. If the connection allocation fails, the data transfer cannot be started at all. If we allocate a short path, we can reduce delivery latency, energy consumption and resource utilization.

In order to minimize the packet delivery latency and resource cost, usually the goal is to select the shortest path out among all contention-free paths from source to destination. This path search problem has exponential complexity with the path length, and the exact complexity function depends on the network topology, more particularly, on the path diversity and length parameters [Ste12]. For a mesh network, if a detour is allowed, at each hop there are up to 4 directions to go, as shown in Fig. 2.1. Assume l is the distance

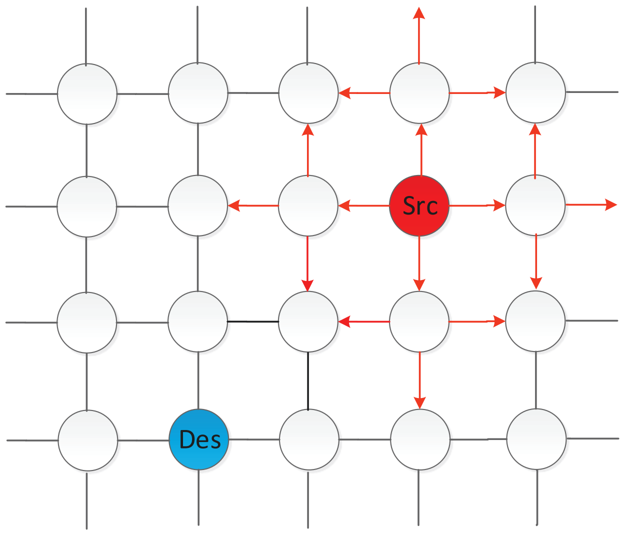


Figure 2.1: Src tries to find a contention-free path to the Des. As the detour is allowed, during the path search, at each node there are up to 4 directions to go. Src: source node, Des: destination node.

(i.e. the hop count) between source and destination, d is the number of allowed detours, then the rough complexity function would be $O(4^{l+d})$, where 4 is the path diversity and $(l + d)$ is the length components.

2.2 Related work

The allocation techniques can be grouped into two categories: i) static (design-time) allocation and ii) dynamic (run-time) allocation.

The static allocation [LJ08, SBSK12, KS14, SKS13, SG11a, MGK14] is done at the design (compile) time of the system, and usually complicated allocation algorithms are adopted. The communication patterns and connection requirements are assumed already known at design time, and the allocation cannot be changed according to the dynamic applications' requirements during run time. Moreover, since the static allocation cannot use the knowledge of real-time network communication traffics, the resource utilization is usually sub-optimal. Consequently, they are not well suitable for dynamic systems.

In the dynamic connection allocation techniques, the connections are allocated at run-time according to the real-time applications' requirements based on real-time network states. It can be further divided into two categories: i) centralized [SNG12, MMB07, MBD$^+$05, WF08, WF11, SG11b, PMM15, HCG07, HG07] and ii) distributed allocation [LJL14b, GDR05, LJL12, LL12a, LL12b, LL11, Hei14].

2.2.1 Distributed allocation techniques

In a distributed allocation, typically the source node sends a setup signal for searching a path that traverses through the NoC to try to reach the destination node. The search signal can be delivered over a dedicated setup network or normal NoC based on some routing algorithms. The resource is reserved by the search signal hop by hop. When the search signal reaches the destination, it means the search succeeds and an acknowledgment signal is sent back to the source node. When the source node receives the acknowledgment, the connection is established successfully, and the data transfer starts. The distributed allocation has good scalability, but the problem of distributed allocation is the lack of the global knowledge, e.g. if there are several concurrent requests, the corresponding searches might block each other, especially under heavy traffic load and high connection request rate. When the failure of the allocation occurs, an additional mechanism is needed to tear down the failed partial setup path. Moreover, since the setup search operates at relatively low clock frequency of the network (compared to the high speed central manager), the setup latency is increased. Furthermore, usually the distributed approaches are constrained to search minimal path, which limits the path diversity.

In [LL12a, LL12b, LL11], when a node needs a connection to another node in the network, it sends a best-effort setup packet, which is routed to the destination based on XY deterministic routing algorithm and reserves a channel in each crossed router along the path. When the setup packet reaches its destination, an ACK packet is generated. Upon reception of the ACK, the source then starts transferring data. The problem of this approach is because of the unpredictable contention of best-effort packets, the setup latency is not guaranteed. Moreover, the XY deterministic routing can only search one fixed path between source node and destination node without exploring other possible paths, which seriously limits the success probability of the connection setup.

In parallel probe search [LJL14b, LJL12, Sha15], the source node sends a setup packet for searching path that traverses through the NoC along all minimal paths to try to reach target node. It is a flood-based algorithm which eliminates redundant incoming paths. The probe search in [LJL12] is proposed for basic CS without link sharing. The probe search in [LJL14b] is an extended version for TDM CS with a double time-wheel technique used to make backtrack efficient and to guarantee the setup delay. In this approach, several trials for success might be needed due to the fact that this method investigates single slot at a time. For instance, if the link is split into S time slots, it has to search S times in the worst case to find the path or to determine the path is not available. Furthermore, this approach enables single-slot allocation, but the problem of multi-slot allocation was not addressed in recent work. Moreover, since the search packets have to flood through the NoC to route via relatively complex routers, it would cost more energy than dedicated centralized systems.

Virtual-channel based distributed allocation is proposed in [Hei14], which provides different levels of throughput and latency guarantees for point-to-point connections. The weighted round-robin scheduling is used for arbitration. Different Service Levels for communication are assigned for different applications based on their requirements. End-to-end connections from one node to another are reserved as a chain of virtual channels. This method can provide high throughput, but since each flow requires exclusive virtual channels, the cost of the area is high. Furthermore, the low setup latency is not guaranteed.

2.2.2 Centralized allocation techniques

In a centralized system, a central manager is responsible for connection allocation. Since the central manager has the global knowledge of the system, it could achieve global optimal results. The centralized system typically is based on software solution. The authors in [SNG12] e.g. utilize Microblaze processor while an ARM processor is employed in [SBSK12, MMB07]. Software solutions provide excellent flexibility, however, they might suffer from relatively long allocation time. For instance, single path exhaustive path-search in [SNG12] tries to add links to the current path if the link provides sufficient slots and is closer to destination. If all links of current node fail, it rolls back to the previous node and tries to search another direction. Due to sequential investigation of a single link at a time, and allocation of all required slots on a single path, thousands of processor cycles are required for single allocation.

In order to increase the allocation speed, HArdware Graph ARray (HAGAR) approaches [WF08, WF11] proposed a dedicated hardware connection allocator, which can speedup the allocation by two orders of magnitude against software methods. In HAGAR, the connection allocation problem is solved as a shortest path problem in a graph representation of the NoC. However, HAGAR is employed for basic CS, and does not support link sharing techniques such as TDM and SDM. In paper [PMM15], a centralized hardware unit that uses breadth-first path searching algorithm was proposed with excellent performance. But it is restricted to search of minimal paths, which only considers links that make the distance to the destination shorter, so it cannot detour when there is no available minimal path. Moreover, in both HAGAR and breadth-first search approaches, the path search is divided into two steps:

- Forward search i.e. firstly, the forward search begins at source node that traverses the network to find the path.
- Backtracking i.e. secondly, when destination node is reached, the backtrack starts from destination to collect the associated survivor path parameters.

Though the centralized system has the advantages of global knowledge and high performance, as the network grows and the allocation request rate at the central unit increases, the central unit might be the bottleneck due to the drawbacks of centralism in computation and communication.

The motivation of this work is to address this problem by a dedicated allocator for connection allocation employing novel trellis-search based Allocation algorithm for TDM and SDM CS NoCs. The Register-Exchange technique is adopted to merge the forward search and backtrack into single step to enhance the allocation speed, and the partitioning structure is proposed to enhance the scalability of centralized system.

2.3 Trellis Search based Allocation approach

In this thesis, we propose a dynamic allocation method using a dedicated centralized hardware unit called 'NoCManager', which solves the problem of connection allocation by employing a trellis-search based shortest path algorithm. We call this as TrElliS Search based Allocation (TESSA) approach. The aforementioned shortest path search problem has exponential complexity with the length of the paths. However, the path search problem can be efficiently solved by the dynamic programming optimization approach that transforms the complex problem into a sequence of simpler problems and solved stage by stage, with linear computation complexity [BHM77, Lou95]. Moreover, the dynamic programming optimized path search problem can be efficiently solved by trellis search approach [LKFF12]. The trellis search can explore all possible paths between two given nodes within a guaranteed low latency, and can ensure the found path is the contention-free shortest path. The details of trellis search for TDM CS is presented in chapter 3 and the trellis search for SDM CS is presented in chapter 4.

Chapter 3

Centralized Connection Allocation for TDM CS NoCs

This section presents the trellis path search algorithm for TDM CS connection allocation. In this thesis, we proposed two different approaches, *Forward-Backtrack (FB) trellis*[CMF16a, CMF16b] and *Register-Exchange (RE) trellis*[CMF17d]. The Register-Exchange technique saves the entire path information during the forward search, and thus compared to forward-backtrack approaches where a backward phase is required to build the path after the forward search, here the allocation time is reduced by half. Moreover, in both approaches, the trellis graph consists of three categories, *unfolded structure*[CMF16a], *folded structure*[CMF16b] and *bidirectional structure* [CMMF]. The bidirectional unfolded structure can provide high allocation speed while folded structure is more efficient in terms of hardware, which will be suitable for different scenarios depending on different requirements. Furthermore, the single-layer approach is proposed, which only needs to implement one layer of the trellis graph, and all slots can be searched simultaneously in the single layer. Compared to previous approaches in which multiple layers of the trellis have to be implemented, the consumption of hardware resource is reduced dramatically. Finally, the partitioning structure[CMF17a] is proposed to address the scalability issue. The different categories of TESSA structures are shown in Fig. 3.1.

3.1 Introduction of TDM CS

In TDM CS NoCs [YZSZ14], the link capacity is split into multiple time slots to be shared by multiple flows. The allocation information is stored in a slot allocation table of particular router with one table for each shared resource (a link). The allocation tables are synchronized such that a flow with slot t allocated at a specific router, gets slot $(t + 1) \bmod S$ at the next hop at neighbor router, where S is the number of slots in the slot table [GEEK11]. Fig. 3.2 illustrates TDM routing with a router network and

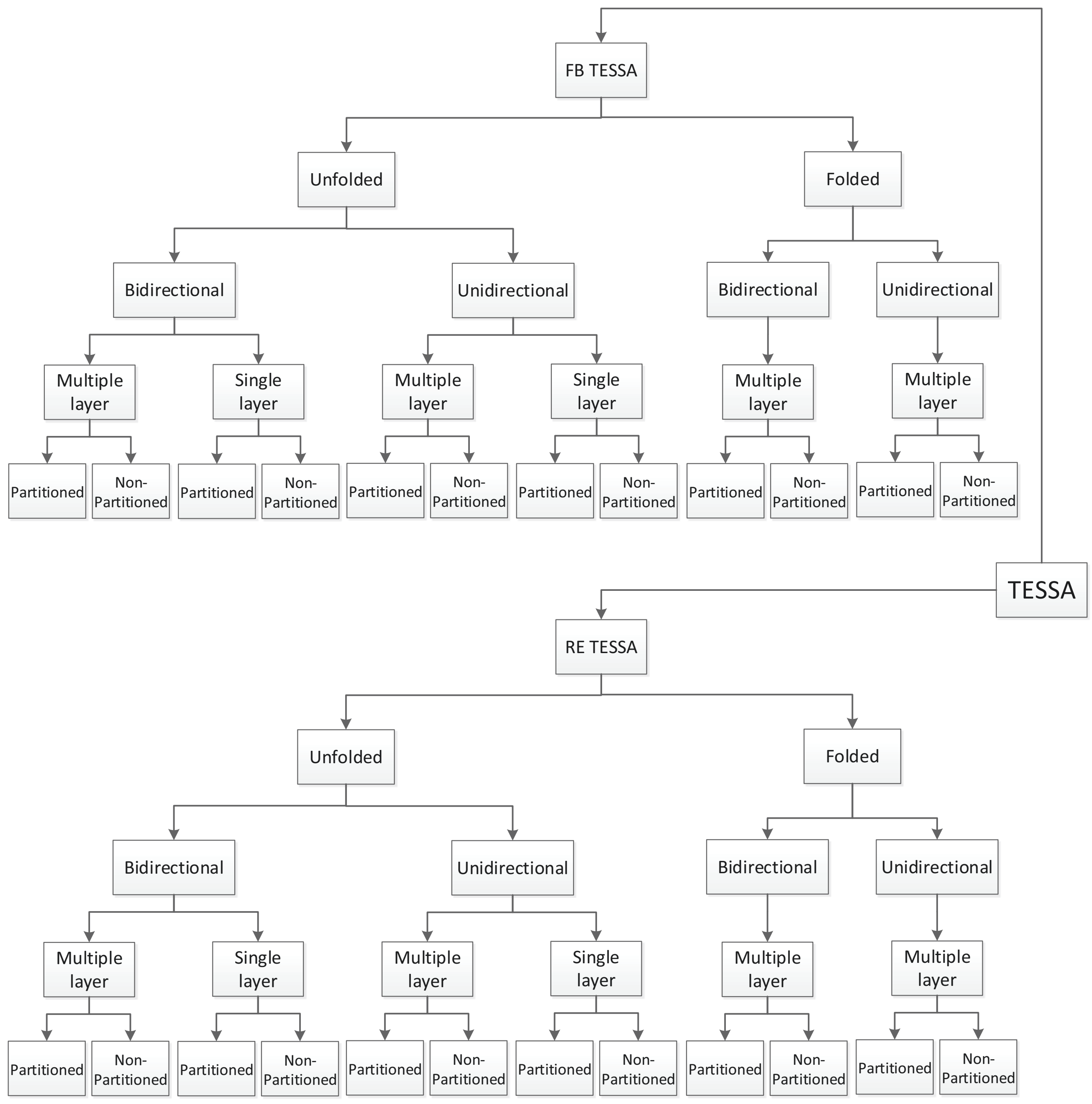


Figure 3.1: The classification tree of TESSA structures.

its corresponding slot tables. In this case, each link bandwidth is split into four time slots, and thus can be shared simultaneously by at most four different flows. The network contains four routers, R0, R1, R2, and R3. The three arrows labeled a, b and c represent flows. Router R0 switches flow a from input port i_3 to output port O_1 at time slot 0, as slot table 0 indicates. Similarly, R1 switches flow a from input i_3 to output O_2 at slot 1, as slot table 1 indicates. At slot 2, R2 switches flow a from i_0 to O_2. Hence, flow a travels along path $R0 \rightarrow R1 \rightarrow R2$ with slot sequence $\{0, 1, 2\}$. The TDM CS routing is contention-free as there is at most one input port to each output port at single time slot.

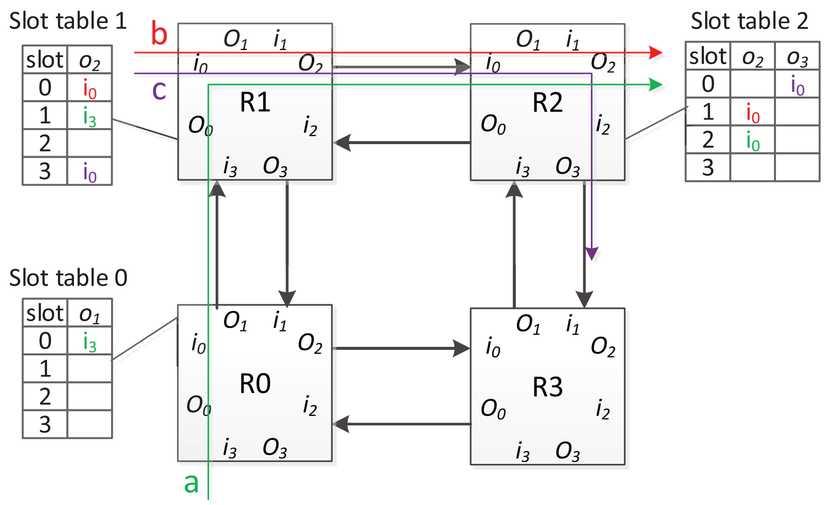


Figure 3.2: Contention-free TDM CS routing.

In TDM CS NoCs, the latency is guaranteed by allocating the contention-free shortest path, and the bandwidth is guaranteed by the number of slots allocated to the traffic flow. For example, if a traffic flow requires half of the link bandwidth, if the size of the slot table is four, two slots will be assigned to that flow.

3.1.1 Connection allocation in TDM CS

In connection-oriented TDM-CS communication, the connection allocation has to find the contention-free path from source node to destination and allocate slots along the path. An example of connection allocation is shown in Fig. 3.3. The source node sends out search flits to try to reach destination node. After each hop, the available slots on the path may become less. At some nodes, there may be no available slots at all, and thus those nodes will discard the search flit, as shown in Fig. 3.3 at node 2. When the destination is reached, the backtrack starts from destination to select the path and time slots. Consequently, the path from source to destination is selected as $Src \rightarrow N0 \rightarrow N1 \rightarrow Des$, and the slot sequence along the path is $\{1, 2, 3\}$.

3.2 System Model

The system model of a dedicated allocator (i.e. NoCManager) based NoC architecture is illustrated in Fig. 3.4. The NoCManager (NoCM) attempts to allocate the appropriate connections when it receives connection requests. In order to reduce communication cost, the NoCM is connected to the center node of the NoC.

There are three possible schemes for the communication between NoCM and NoC nodes, i.e. over a dedicated configuration network, connected via dedicated wires, or via the

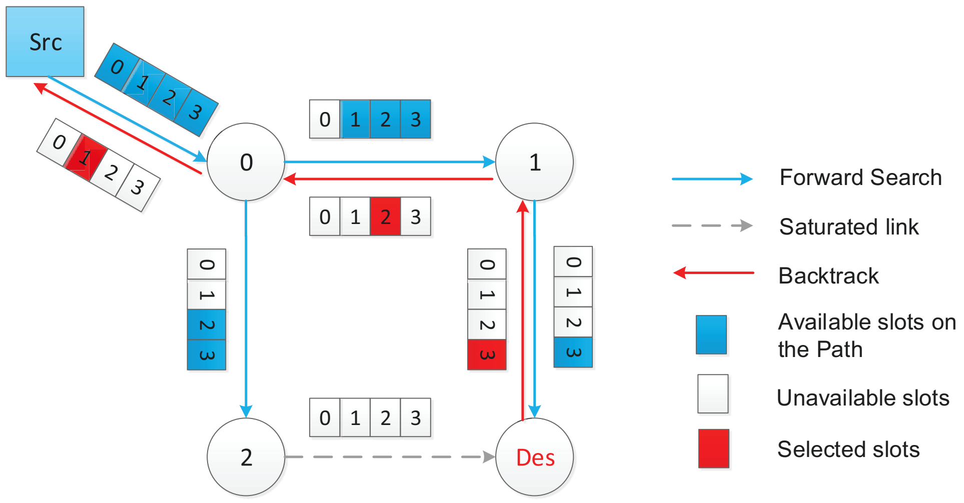
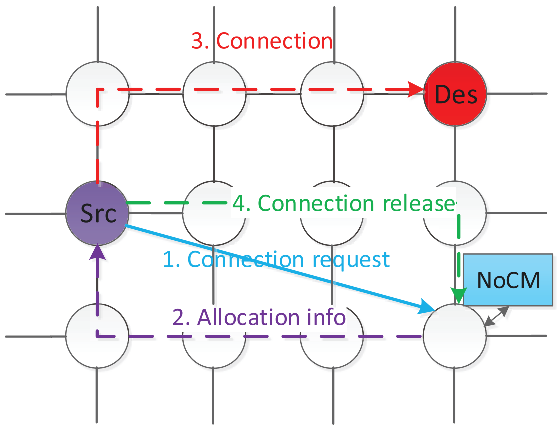


Figure 3.3: A connection allocation example for TDM CS. Src: source node, Des: destination node.

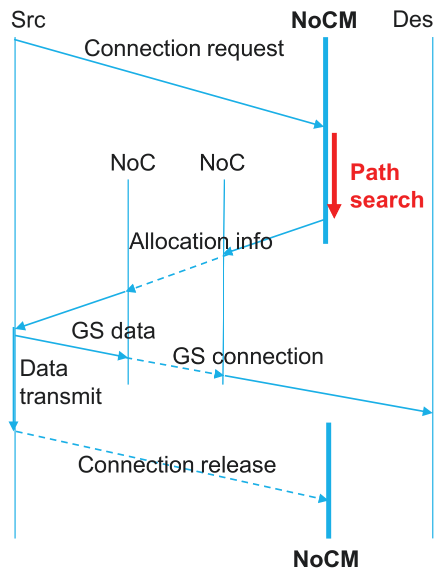
existing NoC. In [SMG14, PMM15, MTSA10, JPL08], the communication between the NoC and the manager is via a dedicated configuration network. The dedicated network can provide high speed, but is costly in terms of hardware resources. In HAGAR[WF08, WF11], the central manager communicates with the NoC via dedicated wires. This can provide high speed, but it may pose wiring difficulty for large network in chip design. In Lusala's work[LL11, LL12a, LL12b] and Wolkotte's work [WSRS05], the configuration packets are delivered over the NoC as best-effort packets. The cost of hardware is relatively low, but the problem is that some resources have already been reserved by other GS flows, so the configuration packet has to try node by node to get the free path to reach the target node. In addition to the unpredictable contention of best-effort packets, the connection configuration latency is not guaranteed. In order to achieve high allocation speed, in this thesis the source node sends the connection request to NoCM via dedicated wires. In a mesh network, for each source node, $\log_2 M$ bits wires are needed to indicate which node is the destination, where M is the number of nodes in the NoC. Due to the *partitioning structure* idea that divides the large system into multiple smaller logic partitions with multiple local managers (explained in section 3.7), each NoCM only manages and connects a limited number of nodes in its local region, so the dedicated NoCM-node wires will not be much overhead. The allocation information from NoCM to source is delivered over the NoC as GS packet, with the associated path found by NoCM as GS path.

In NoCM, the connection allocation comprises three steps: receive connection request from source node, find the required contention-free path and send back the allocation information to source node. The complete procedure for connection allocation is as follows:

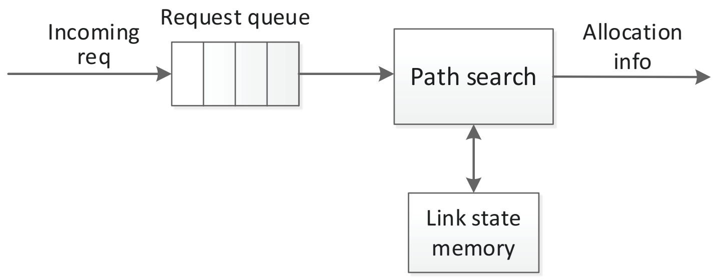
1. The source node sends the connection request to NoCM. These requests are buffered in a request queue in NoCM.



(a) System Model of the NoC



(b) Connection request processing procedure diagram



(c) Block diagram of the request processing procedure in NoCM. ‘Link state memory’ stores the current state of links.

Figure 3.4: Proposed System model of the NoCManager based NoC platform. Src: source node, Des: destination node.

2. NoCM processes the requests within the request queue, searching the best contention-free path between the source node and destination.

3. NoCM sends the allocation information (i.e. the information of the found out best path) to source node in the case of success or retries later if it fails. A failed request is retried only when the retried time does not exceed the deadline and there are no unprocessed requests waiting, otherwise it is discarded.

4. After receiving the allocation information, the source node starts to transmit data along the allocated connection. The implementation details are presented in chapter V.

5. After the data transfer is finished, the source node deletes the allocated connection, and also informs the NoCM to free the corresponding allocated resources. The release information to NoCM is sent as BE packet.

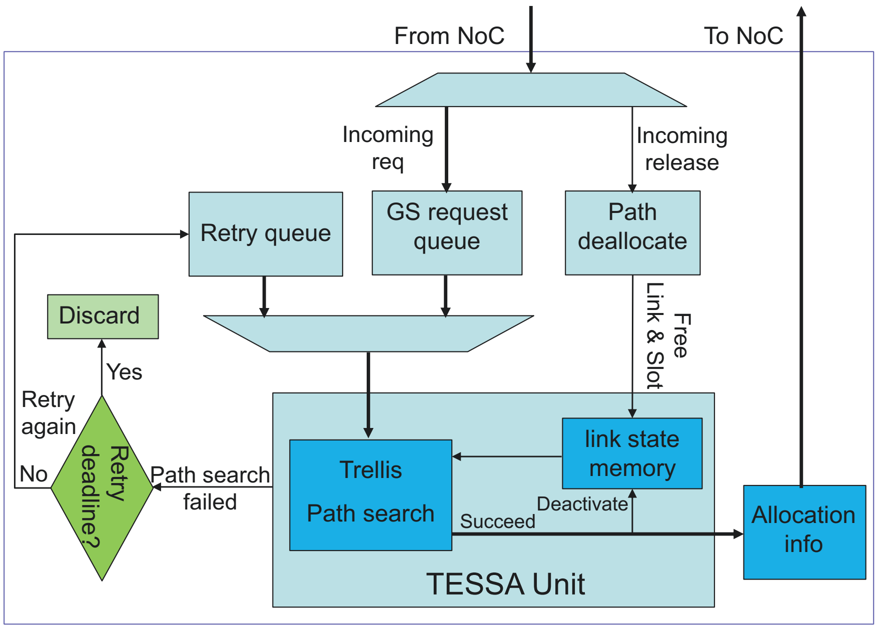


Figure 3.5: Block diagram of the NoCManager

Where the allocation information contains the connection information to indicate the specific time slots that the data is inserted into the network at source node, and the information for each hop to indicate which output port to go (3 bits for five directions in 2-D mesh network, east, west, north, south and local).

Step 2 is the main contribution of this thesis, and is explained in details in the following sections.

3.3 Connection Allocator Architecture

The NoCManager solves the problem of connection allocation as a shortest path problem in a trellis graph description of the NoC, in order to find the shortest contention-free path and allocate slots between two given nodes. A block diagram of the NoCM is shown in Fig. 3.5 and comprises trellis path search module and link state memory. NoCM collects and processes the incoming connection requests within the request queue. The resulting allocation parameters are sent through the NoC to respective source node.

3.3.1 Formalizing The Trellis Graph Structure

The aforementioned shortest path search problem can be efficiently solved by the dynamic programming approach that transforms the complex problem into a sequence of simpler problems and solving them step by step. Moreover, the path search algorithm comprising successive NoC traversal from source to destination node can be represented by trellis graph, which is similar to the popular Viterbi algorithm. The transformation of the NoC architecture to associated path search trellis graph is illustrated in Fig. 3.6. The trellis graph is a time indexed version of the NoC graph.

Note, though in this thesis we restrict the trellis graph to 2D-mesh topology at analysis, it can be applied to any other topologies even in mesochronous or asynchronous networks. The mesochronous or asynchronous TDM NoCs can use the synchronization token for handshaking as in aelite [HSG09].

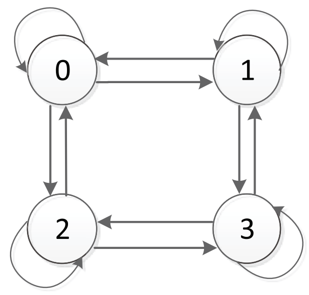
3.3.1.1 General Model

There are five important characteristics of the trellis graph, which are:

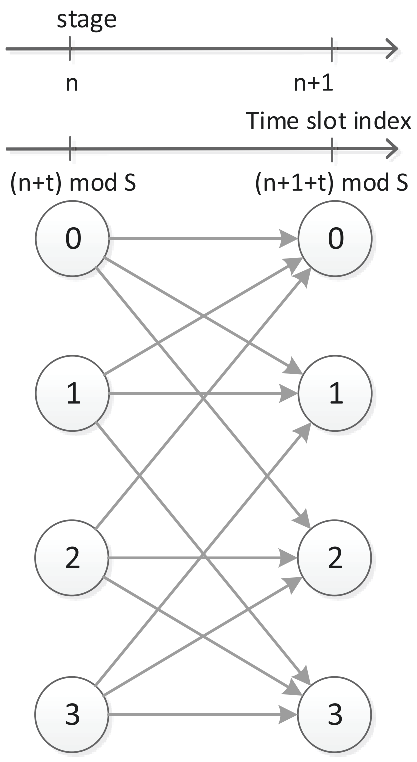
1. **Stages**: The network traversal can be mapped to trellis graph which is structured into multiple stages (Fig. 3.7a), while stage n is represented by set of the network nodes reached after n hops starting from the initial source node. Trellis graph branches (edges) are associated with node-to-node hops representing available NoC links. The path search problem is solved sequentially one stage at a time. The stage (i.e. the column) of the trellis graph is a collection of whole nodes of the network, which is defined to represent the traversal through the network of single hop. We refer to the "decision stages", meaning the number of stages which have to be traversed to make decision, not counting the first stage, since the first stage does not require any decision making. By default, the number of the "decision stages" is $2N - 2$ for an $N \cdot N$ mesh network, which equals the longest minimal path (i.e. the longest possible path of all minimal paths) in the network. The "decision stages" number can be increased to allow detours. For example, if the search is allowed to take a detour of d hops, then the trellis is constructed as $d + 2N - 2$ decision stages. Each decision stage is a representation of the network. [1] The stages have time implications to represent different time slots associated with different hops. As in Fig. 3.6b, assume at the initial stage (stage 0) the associated time slot is t, then at stage n, the associated time slot will be $(n + t) \bmod S$. If we assume the initial slot is slot 0, then the trellis graph can be simplified as Fig. 3.6c. The decision variable in any node is to select an *incoming branch* as *survivor path*. At any stage, we only need to know which node we are in to be able to make subsequent decisions. The subsequent decisions do not depend upon how we arrived at the particular node.

2. **States**: The node in the trellis graph, called state, summarizes the knowledge in order to make the current decisions. At each stage, the decision in a particular state is determined simply by choosing one and only one of the active *incoming branches* as *survivor path*, i.e. the previous node that this branch connected to is remembered as predecessor.

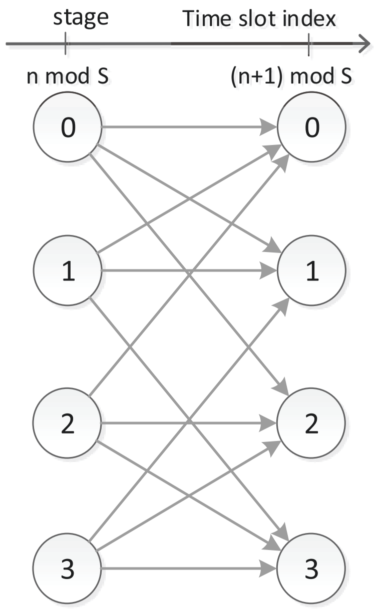
[1] In the following sections, the decision stage is called as stage for short.



(a) Example network graph. A node can reach itself (curve arrow).

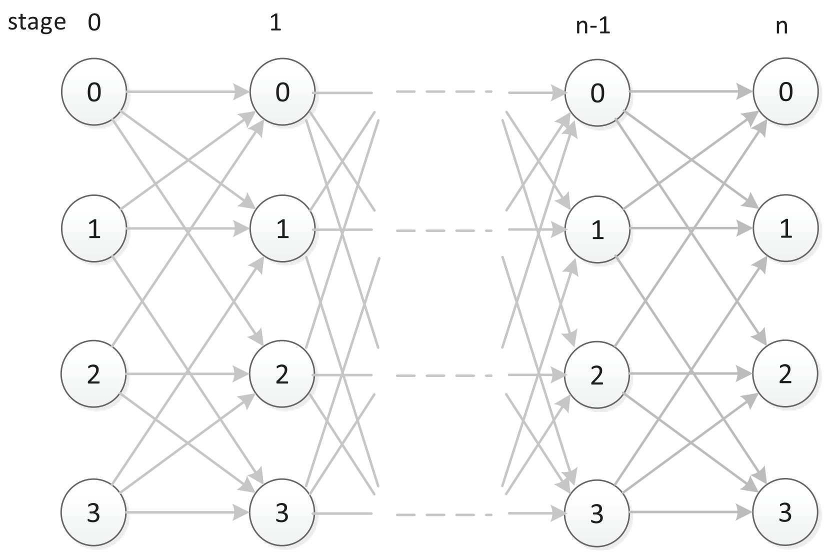


(b) A trellis graph represents the network graph. Assume the initial time slot at the initial stage is slot t.

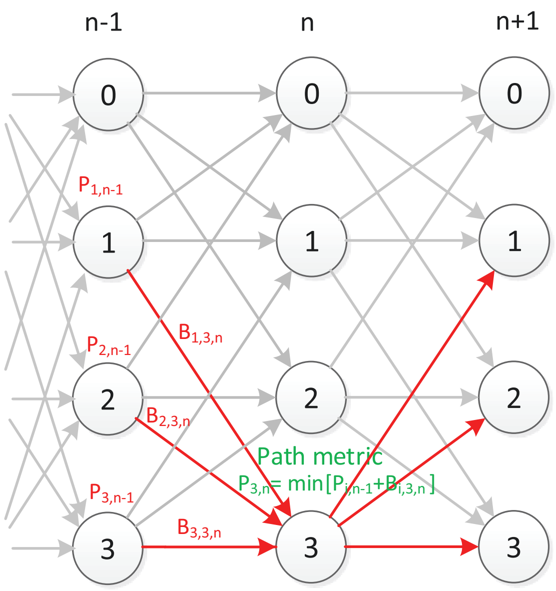


(c) Simplified trellis graph. Assume the initial time slot at the initial stage is slot 0.

Figure 3.6: Network graph represented by trellis graph



(a) Multiple stages of a trellis graph of the example network.



(b) Branch metric and path metric at node 3

Figure 3.7: Multiple stages trellis graph

3. **State transitions**: The forward move from one state at a stage to another allowable state at the next stage in one unit of time (a time slot), is called a state transition which, in the trellis graph, is represented by a directed edge (or branch) connecting the two states. The state transitions act as the links between two respective routers corresponding to a forward step in the network.

4. **Branch metric**: The branch metric (B) is a measure of the transition that reflects the value (importance) of the branch. It is a function of several variables, such as the available slots of the branch (a), the number of requested slots (r), and the weight (w) that reflects the priority of the branch, etc. The information of available slots and

the requested slots can be used to balance network load. The fewer available slots the branch can provide and the more slots the connection requested, the larger the branch metric will be, which indicates more inferior the branch is. When the number of requested slots exceed the available slots that branch can provide, the branch metric becomes infinity, which indicates that branch cannot satisfy the request and will be discarded. The function of the branch metric is as follows:

$$B = \begin{cases} f(r,a,w), & r \leqslant a \\ +\infty, & r>a \end{cases}$$

An example of branch metric function might be:

$$B = \begin{cases} \frac{1}{a}\text{r}\cdot w, & r \leqslant a \\ +\infty, & r>a \end{cases}$$

5. **Path metric**: The path metric is the minimal accumulated branch metric over the shortest path from the initial state to the current state. For each state, the incoming branch that produces minimal path metric is selected as *survivor path.* The branch metric for the transition from state i to state j at stage n is defined as:

$$B_{i,j,n}$$

Define $P_{j,n}$ as the path metric for state j at stage n, and S_j is the set of states that have transitions to state j, then:

$$P_{j,n} = \min_{i \in S_j} [P_{i,n-1} + B_{i,j,n}]$$

According to trellis graph example in Fig. 3.7b,

$$P_{3,n} = min\{P_{1,n-1} + B_{1,3,n}, P_{3,n-1} + B_{3,3,n}, P_{2,n-1} + B_{2,3,n}\}$$

If the branch from state 2 produces the minimal path metric, the path metric of state 3 at stage n will be

$$P_{3,n} = P_{2,n-1} + B_{2,3,n}$$

The goal of the shortest path problem is to find the path between source and destination node with the minimal path metric. At the last stage, the survivor path with the minimal path metric is the desired path.

3.3.1.2 Path Search Model Simplification

The previous section presents the general formalized model of trellis graph. However, the branch metric can be simplified. In this work, a simplified model has been employed that the branch metric can only have two possible values, either 1 or infinity, as follows:

$$B = \begin{cases} 1, & r \leqslant a \\ +\infty, & r > a \end{cases}$$

Therefore, the accumulation operation of branch metric in path metric can be omitted. As long as the branch metric is infinity, that branch will be discarded; otherwise, if the branch metric is 1, that branch can be selected. The simplified path metric becomes:

$$P_{j,n} = \min_{i \in S_j} \left[B_{i,j,n} \right]$$

In our system, every initial slot at the initial stage has such a representation of trellis graph, i.e. if the slot table size is S, there are S representations (layer) of trellis graph in parallel, as in Fig. 3.8. So every slot from the initial stage has its own trellis graph and can search its own path in parallel, as in Fig. 3.9. Consequently, during a search, at each stage we only need to know the branch state at a specific slot. Since the branch at each slot only has two possible status, either free or unavailable (i.e. already allocated), the branch metric of each slot t can be simplified as: [2]

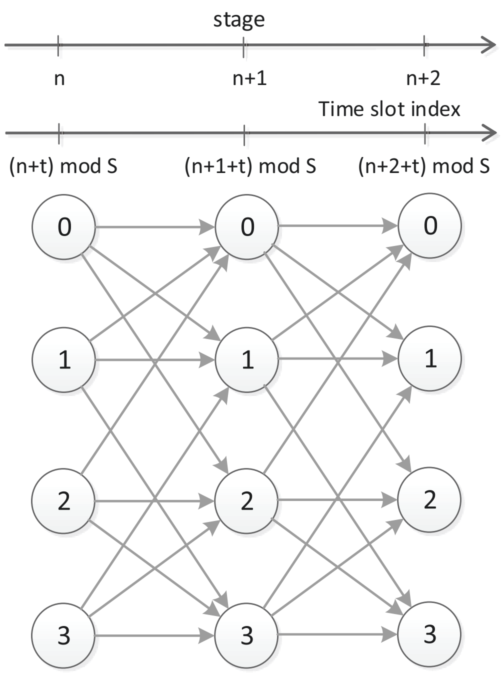
$$B_t = \begin{cases} 1, & \textit{branch is free} \\ +\infty, & \textit{branch is unavailable} \end{cases}$$

If a flow requires a large bandwidth, i.e. multiple slots, since in the proposed approach each slot searches its own path in parallel, the allocation of multiple slots can be done simultaneously and can set up multiple paths, which can split the bandwidth over multiple paths, as shown in Fig. 3.10. Compared to these approaches [SNG12] that allocate the whole bandwidth along single path, our multi-path allocation can increase the success rate significantly [SG11b]. It is worthwhile to mention that the state transition structure reflects the specific NoC topology. However, the dimension of trellis graph depends only on the number of NoC routers, and thus is topology invariant.

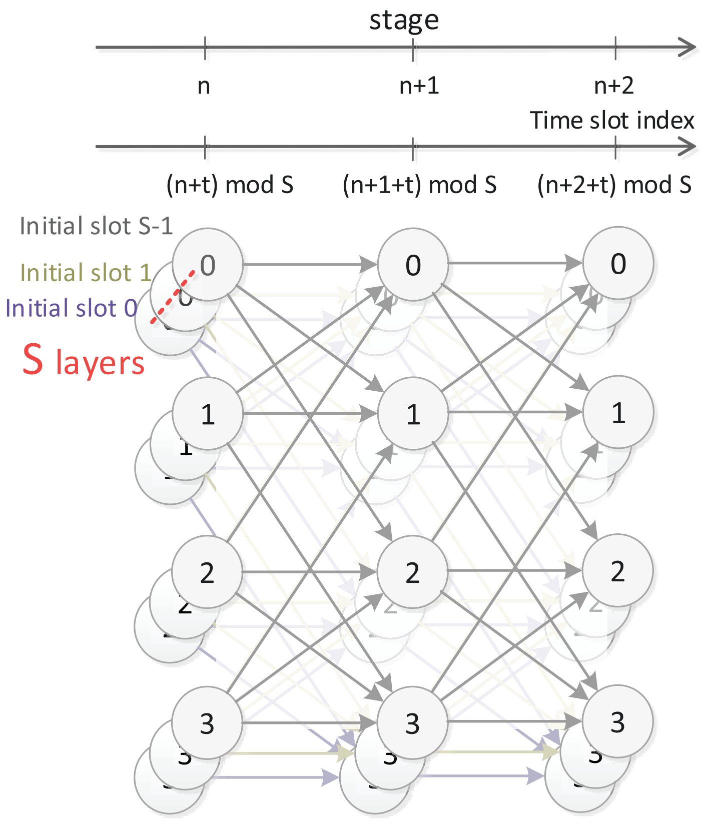
3.3.2 Forward-Backtrack Trellis Path Search

In this section, three different trellis structures are presented and investigated, called unfolded trellis graph, folded trellis graph and bidirectional trellis graph. The trellis graph

[2] In hardware implementation, the value '1' of the branch state register indicates the branch is free, and the value '0' of the branch state register indicates the branch is unavailable.



(a) One layer of the trellis graph. Assume the initial slot at the initial stage is t.



(b) Multiple layers of the trellis graph. Each initial slot at the initial stage has its own layer.

Figure 3.8: Each slot at the initial stage has its own layer of trellis.

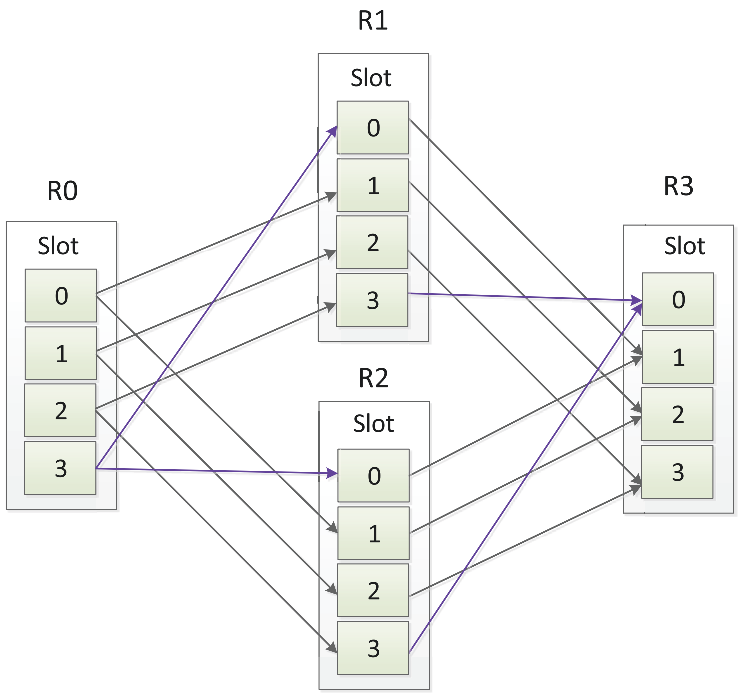


Figure 3.9: Each slot searches its own path in parallel.

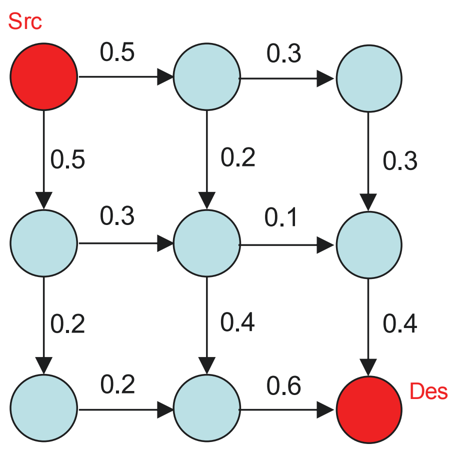


Figure 3.10: Communication bandwidth split over multiple paths. Src: source node, Des: destination node.

is constructed as multi-layer, but each layer is identical to each other, so in this section we take one layer as example to show the trellis path search.

The flow chart of the path search procedure is illustrated in Fig. 3.11. In general, the shortest path search in Forward-Backtrack trellis graph comprises two steps:

1. **Forward search** i.e. traverse the NoC from source node to find the best free path to the destination.

   (a) Search through the trellis, and calculate branch metric to determine whether

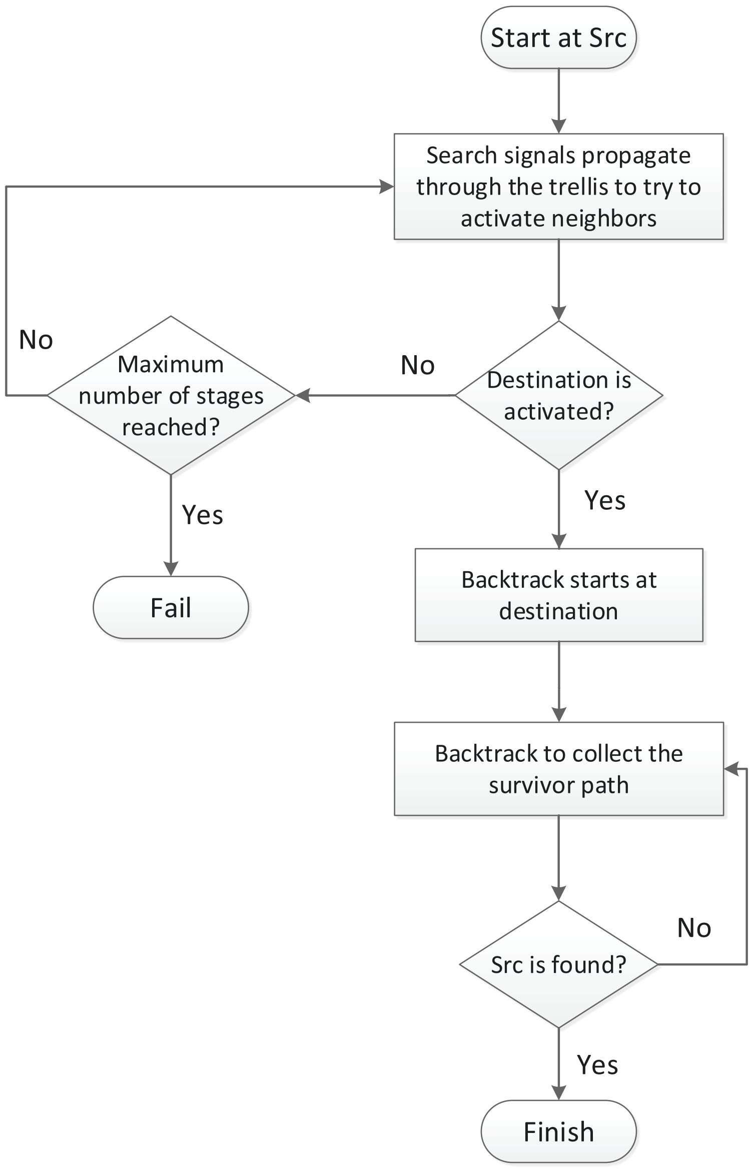

Start at Src
Search signals propagate through the trellis to try to activate neighbors
Destination is activated?
No
Maximum number of stages reached?
No
Yes
Fail
Yes
Backtrack starts at destination
Backtrack to collect the survivor path
Src is found?
No
Yes
Finish

Figure 3.11: The flow chart of the path search in trellis.

the current node can be activated, i.e. both the incoming branch and its associated link state register are available;

(b) If the current node can be activated, save one of the active incoming branches as survivor path;

(c) Continue the search to try to reach the destination node until reaching the last stage.

2. **Backtracking** i.e. sort out the saved survivor path from destination to backtrack the shortest path and collect the associated path and slot allocation parameters.

   (a) Read stored predecessor from current node;

(b) Output this “predecessor”;

(c) Use the predecessor to read out its previous state in the trellis;

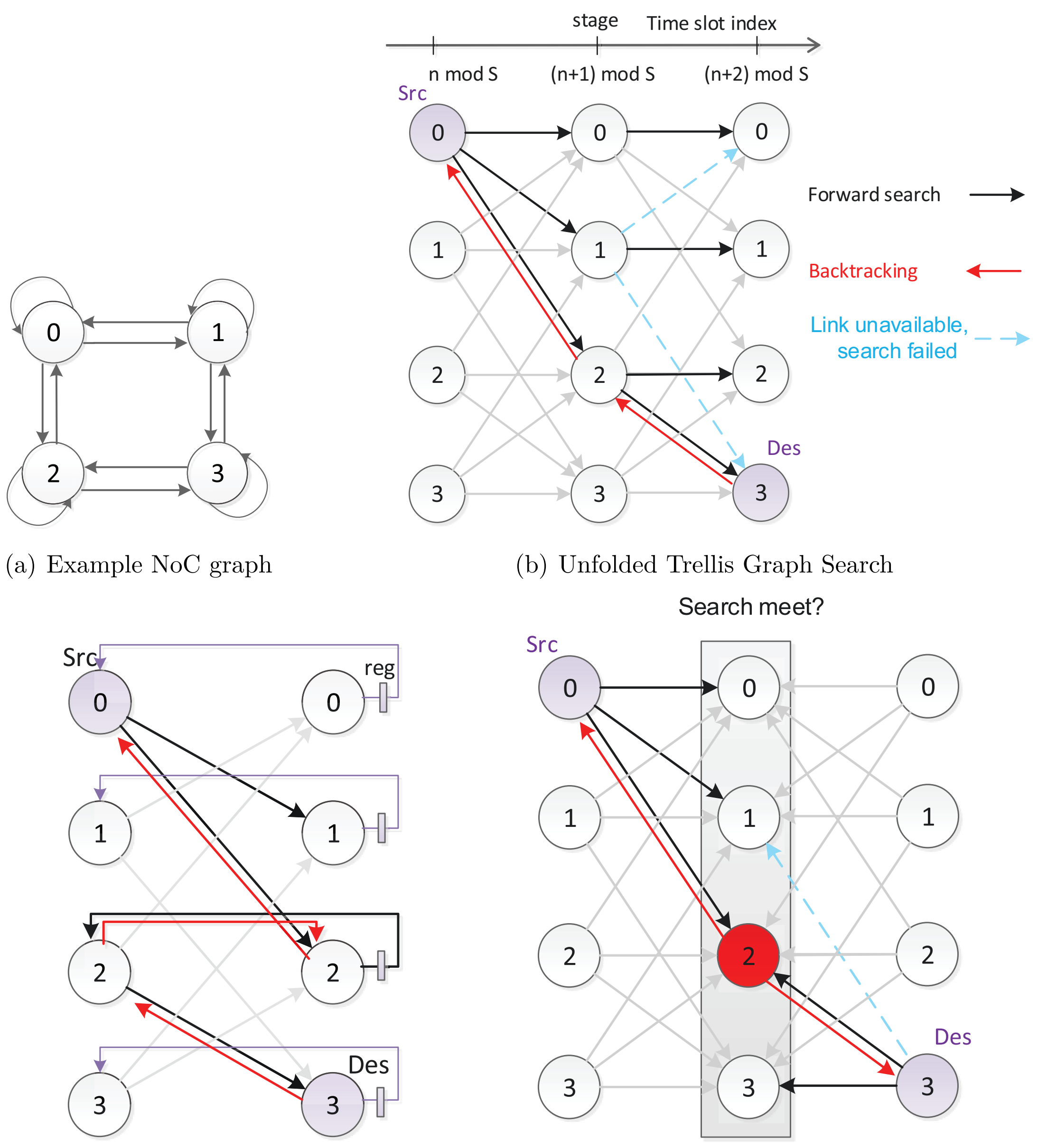
(d) Repeat until find the source node.

3.3.2.1 Unfolded Trellis Search

Since the NoC topology and size are already known at design time, the respective trellis graph can be constructed as e.g. for 2x2 NoC illustrated in Fig. 3.12a the associated trellis graph is constructed in Fig. 3.12b. Assume node 0 is the source and node 3 is destination. The search signals start from source at the first stage, traversing through the trellis to try to activate its connected neighbors at next stage. Source activates its connected neighbors node 1 and node 2 at the second stage, and node 1 and node 2 try to activate their connected neighbors at the third stage. Assume the branch $N1 \rightarrow N3$ at second stage is already occupied, so node 1 cannot activate node 3. During the forward search, if a node is activated by several nodes at the same stage, one and only one (any one) is remembered as its predecessor. If the node was already activated in the previous stage, it will store itself as the predecessor. Hence, it can ensure the selected path is the shortest between the source and target nodes. So suppose if the destination, node 3, is activated by both node 2 and node 1, but only one node will be remembered as its predecessor. When the destination is active, backtracking is started from destination to backtrack predecessors in order to collect the path information. Node 3 backtracks its predecessor node 2 at second stage, and node 2 backtracks node 0 at first stage. Now the path from source to destination is obtained as $N0 \rightarrow N2 \rightarrow N3$. Assume the beginning slot at source is t, then we can obtain the slot sequence along the path as $\{t$, $(t{+}1) \bmod S$, $(t+2) \bmod S\}$, where S is the slot table size.

3.3.2.2 Folded Trellis Search

The proposed unfolded trellis path search algorithm exhibits regular structure and, hence, it can be efficiently mapped on to a folded architecture. In such case, the hardware resources are reused by all partitions of folded algorithm. The folded architecture requires additional output registers in order to hold the values of intermediate results to be used as input values in the next iteration. The folded path search algorithm of example in Fig. 3.12b is illustrated in Fig. 3.12c, in which only one decision stage has to be implemented (does not count the first stage). There is a register for each node to store which predecessor activates it, and it only stores the predecessor that activates it first. When a node is active, at next cycle it will forward search signal to its first stage node, and does the search propagation again. Note now the search costs multiple cycles, i.e. one cycle per iteration. The search can be stopped in two cases: i) either the target node has been activated with sufficient bandwidth or ii) after certain number of iterations (by default $2N-2$ iterations)



(a) Example NoC graph

(b) Unfolded Trellis Graph Search

(c) Folded Trellis Graph Search

(d) Bidirectional Trellis Graph Search

Figure 3.12: a)2x2 2D-mesh example NoC; b)schematic structure of the unfolded trellis Search for the example NoC; c)schematic structure of the folded trellis Search; d)schematic structure of the bidirectional trellis Search. Src: source node, Des: destination node.

(or we can stop the search when there are no new nodes being activated during the search any more). Hence, the livelock is avoided. In the routing algorithm that offers several alternative paths to the destination, packets may arrive out of order at the destination. In the traditional routing algorithms that offer several alternative paths to the destination, like in [SG11b], a complicated in-order path selection mechanism is required to ensure packets arrive at the destination in order. This problem can be solved by only choosing paths that ensure in-order delivery. Assume there are two paths beginning at source node at slot $t1$ and $t2$ ($t1 < t2$), with path length $l1$ and $l2$ respectively. The paths are selected only if $t1 + l1 < t2 + l2$, which ensures in-order delivery. However, here, we can have a simple solution. We can only select the paths that reached the destination at the last cycle, which ensures the selected paths have the same length and thus the in-order delivery is guaranteed, so the complicated mechanism of in-order path selection can be omitted. The search signals start from source and activate node 2. At the next cycle, node 2 travels back to the node 2 at first stage and continues to activate node 3. The backtrack starts from destination node 3 and sorts out nodes in sequence $N3 \rightarrow N2 \rightarrow N0$. Therefore, the path from source to destination is acquired as $N0 \rightarrow N2 \rightarrow N3$.

3.3.2.3 Bidirectional Trellis Search

The path search presented in previous sections is started at one side, i.e. from source node to target node. Actually, the search can be started at two sides, source and target nodes simultaneously, and checked at the middle stage to see whether search signals from two sides meet to determine whether the search was successful or not. The bidirectional path search algorithm of example in Fig. 3.12b is illustrated in Fig. 3.12d. The search from source activates node 2, and the search from destination also activates node 2. In the middle stage, the search signals from source and destination meet at node 2, which means the search succeeds. The backtrack starts from node 2 to source and destination simultaneously. The path from source to destination is obtained as $N0 \rightarrow N2 \rightarrow N3$. In bidirectional search, the critical path is halved while the area stays almost the same.

3.3.3 Forward-Backtrack Trellis Path Search Implementation

This section presents the implementation details of unfolded trellis, bidirectional trellis and folded trellis.

The Detect-Select-Shift (DSS) Unit is the core module that implements the function of state in the trellis graph, which evaluates the propagated search signals from previous stages and generates bit-vector flags representing slot availability on specific links. The knowledge about the actual link allocation state is stored in the 'Link State' register. When a link is allocated at a specific slot, its corresponding state register is set to '0' excluding it thereby from future search. Correspondingly, state register is set to '1' when the link is released.

The details for an example DSS of node 3 in the example NoC with slot table size of 2 are shown in Fig. 3.13. Since the implementation for each slot is the same, we take one slot as example. The working flow for each slot in DSS unit is as follows:

1. **Detect the available slot**: The search signal from $N1 \rightarrow N3$ and its corresponding 'Link State' register are connected to an AND gate. If the search signal as well as

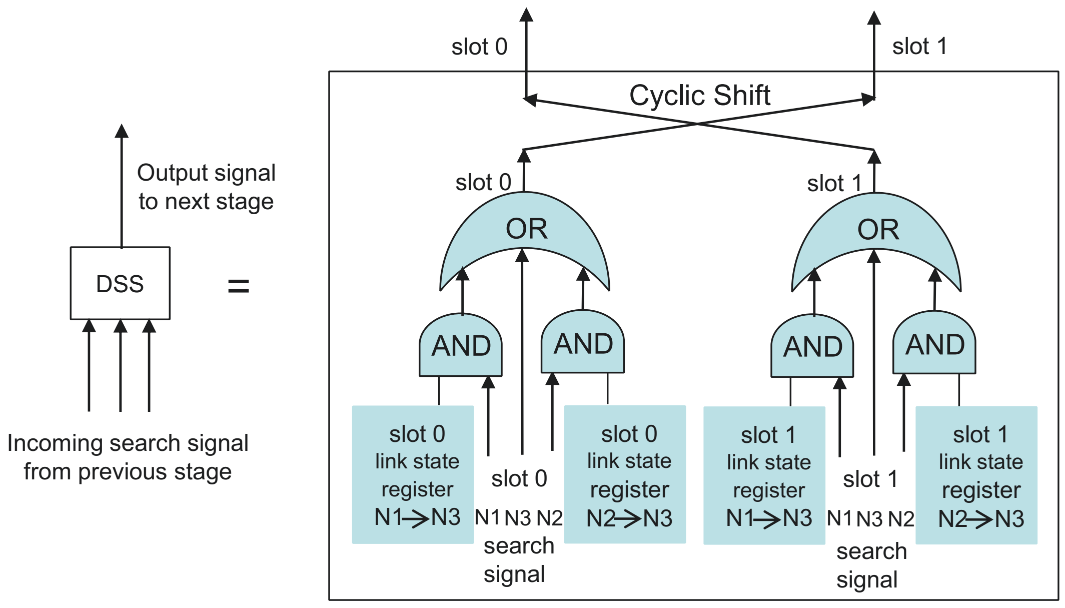

Output signal to next stage
DSS
=
Incoming search signal from previous stage
slot 0
slot 1
Cyclic Shift
slot 0
slot 1
OR
OR
AND
AND
AND
AND
slot 0 link state register N1->N3
slot 0 N1 N3 N2 search signal
slot 0 link state register N2->N3
slot 1 link state register N1->N3
slot 1 N1 N3 N2 search signal
slot 1 link state register N2->N3

Figure 3.13: Implementation details of an example DSS unit of node 3

the corresponding link are both valid, the current node can be activated by this search.

2. **Select active incoming branch as survivor path**: The detect signals from the neighbor (predecessor) nodes (i.e. the output of AND gates) that are at the same slot are connected to an OR gate. If the node can be activated (i.e. the output of OR gate is ‘1’), one of the active incoming branches is saved in register as survivor path.

3. **Cyclically shift slot**: Cyclically shifts slots to synchronize with next hop. The search signal at slot t after shifting comes to slot $(t+1) \bmod S$, where S is the slot table size. The cyclic shift is realized by wire connection.

We can see (in Fig. 3.13) the critical path of DSS unit is only an AND gate and an OR gate, which is quite simple. For all the three different trellis structures, unfolded trellis, bidirectional trellis and folded trellis, the implementation structure of DSS unit is the same.

3.3.3.1 Unfolded Trellis Implementation

The implementation schematic of the unfolded trellis is shown in Fig. 3.14. The path search is completed in two cycles. At the beginning search begins at the source node by setting its all slots to logic ‘1’ (i.e. be valid), then the search signals propagate forward along the edges to the connected neighbors via DSS Unit that checks the slot’s availability. The still valid signals continue to propagate in this way until the end of the trellis is reached where

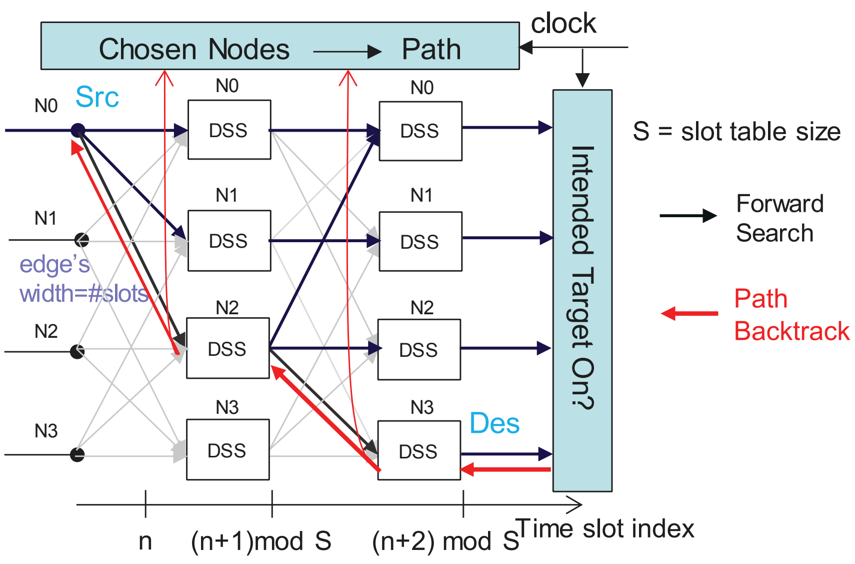


Figure 3.14: Implementation schematic of the unfolded trellis for the example NoC

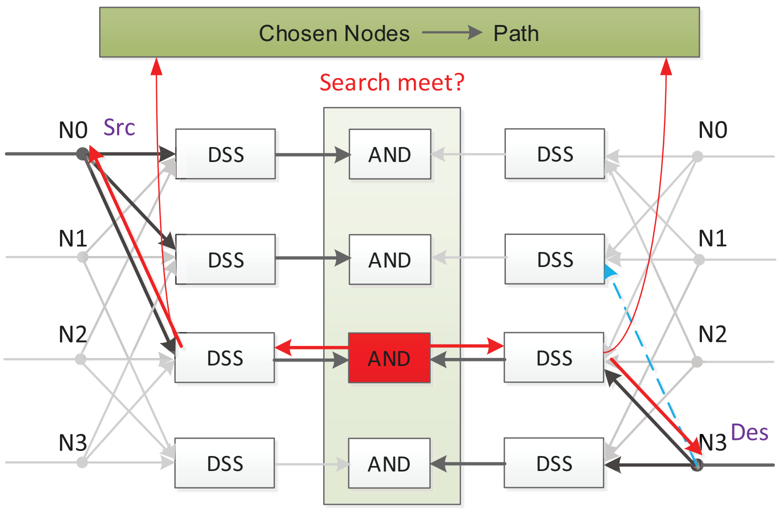


Figure 3.15: Implementation schematic of the bidirectional trellis for the example NoC

a register stores which of the nodes could be reached through the NoC. The bandwidth (i.e. the number of available slots) is only checked at the last stage.

At the next cycle, the backtrack is started with each selected slot backtracking its own path if the intended target node was active. The path is selected by reading the stored predecessors starting at the intended target. Each selected node at each stage updates that it was selected into the 'chosen node' register, where eventually the complete path from source to target node can be found. The selected slots' corresponding 'Link State' register will be set to '0'.

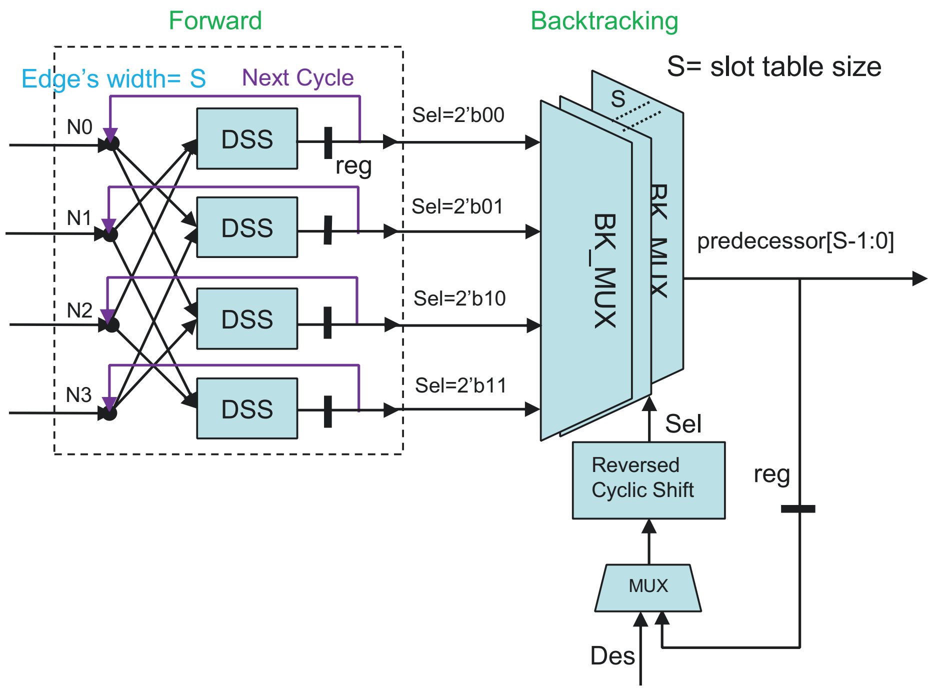


Figure 3.16: Implementation schematic of the folded trellis for the example NoC

3.3.3.2 Bidirectional Trellis Implementation

The implementation schematic of the bidirectional trellis is shown in Fig. 3.15. The search is started at source (at initial stage) and destination (at last stage) simultaneously, propagated until reaching the middle stage. At the middle stage, the corresponding search signals from two sides are connected to an AND gate to check whether searches meet. At the next cycle, if two searches meet, backtrack starts from the middle stage by reading out the stored predecessor hop by hop. Each selected predecessor at each stage are saved into the 'chosen node' register, where eventually the complete path from source to target node can be found.

3.3.3.3 Folded Trellis Implementation

The search begins at the source node by setting its all slots to logic '1', then the search signals propagate forward along the edges to the connected neighbors via DSS Unit. At the next cycle, the still valid signals travel back to the first stage and try to activate its connected neighbors. It continues to propagate in this way until reaching the target node or exceeding the limited number of search cycles.

If the destination node is activated with sufficient slots, the backtracking is started with each selected slot backtracking its own path simultaneously. The path is selected by reading the stored predecessors starting at the destination. The current node that is read out at last cycle will request its predecessor via multiplexer BK_MUX (in Fig. 3.16). Hence, the predecessors are sorted out in this manner until source node is obtained.

It should be noted that the data flit can wait in the node for several cycles until there is available path, to increase the success rate, which is different from the consecutive allocation in [LJL14b, SNG12]. For example, if a node is reached by search signal at slot 0, but the connected downstream neighbor is not available at slot 1 but available at slot 2, the search signal can wait in the current node for one time slot and then reaches the connected neighbor at slot 2.

3.4 Performance Evaluation of Forward-Backtrack trellis

The synthesis and simulation results of our NoCManagers are presented in this section.

3.4.1 Synthesis Results

The NoCManager is designed in synthesizable VerilogHDL and can be generated out of an XML description for different NoC sizes. Using Synopsys Design Compiler, the NoCM was synthesized with TSMC 65 nm technology, for different mesh networks of size from 4x4 to 10x10. For folded TESSA, the critical paths were constrained to 1 nanosecond. For unfolded TESSA and unfolded bidirectional TESSA, the critical path constraints were gradually increased based on the increased NoC size. For example, in unfolded bidirectional TESSA, the critical path is constrained to 1.11 ns for 6x6 mesh while increased to 2 ns for 10x10 mesh.

Since in TESSA all stages are identical, we can only implement one decision stage as folded architecture, which can reduce hardware resource remarkably but requires more cycles to do path search. Hence, we can combine the folded and unfolded structures to fold several stages instead of one stage to provide a suitable tradeoff between area and performance. Therefore, we implemented a TESSA approach that folds half stages, called half-folded TESSA, i.e. in $N \cdot N$ mesh, we implemented $N - 1$ stages, and the forward search can be finished in two cycles at most.

The synthesis results of the four different structures are presented and compared in this section. The performance of the four different structures are compared in terms of area, average $Area \cdot Time$ (AT) complexity per allocation and average energy consumption per allocation. For different structures, the allocation time per allocation is different. In $N \cdot N$ mesh, the average allocation time per allocation of different structures[3] is shown as follows:

[3] The clock frequency of different structures is different.

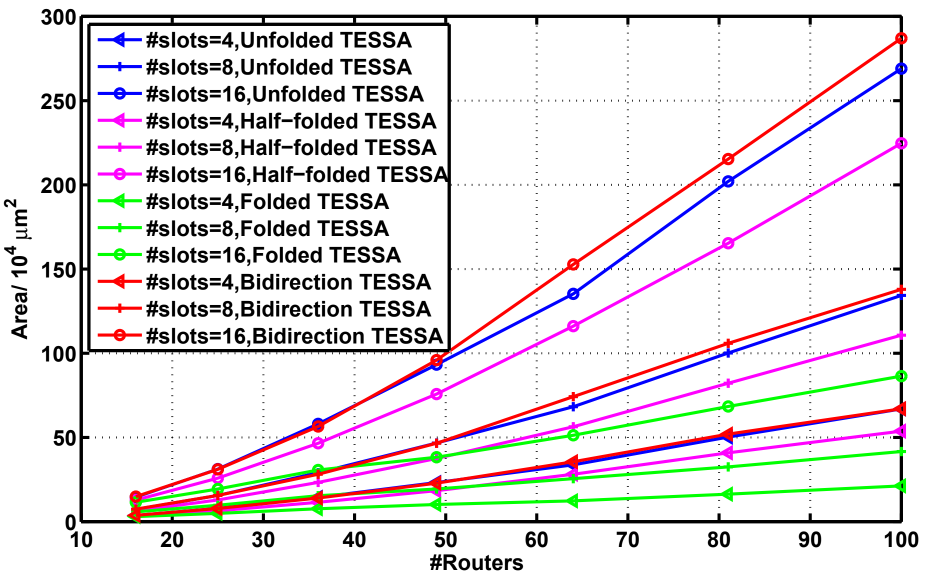


Figure 3.17: Area of different TESSA in different size NoC with different slot table size

- In unfolded TESSA, two cycles;
- In bidirectional TESSA, two cycles;
- In folded TESSA, $2 \cdot (N-1)$ cycles, because average path length is $N-1$;
- In half-folded TESSA, 3 cycles. Because the search can be finished in the first iteration or in the second iteration, i.e. the allocation time can be two cycles or four cycles, so in average three cycles.

From the area synthesis results shown in Fig. 3.17, we can see the area of unfolded TESSA grows with $O(S\cdot\ M\cdot\ \sqrt{\boldsymbol{M}})$ in 2D-mesh (M= #routers, S=slot table size), where $\sqrt{M}$ is related to the number of trellis stages (2$\cdot$ ($\sqrt{\boldsymbol{M}}$ -1)). The area of folded TESSA grows with $O(S\cdot\ M)$ in 2D-mesh. As folded TESSA reuses hardware, its area cost is the least. From the AT complexity results shown in Fig. 3.18, we can see the bidirectional TESSA presents the best results considering area and time product, which is due to the halved search time, while the AT complexity of unfolded TESSA is the worst. The energy consumption per allocation is shown in Fig. 3.19. Still, the bidirectional TESSA presents the best energy efficiency. Since in folded TESSA, additional registers are used to store the intermediate results, its energy efficiency is the worst. Compared to a microcontroller based software solutions [SNG12, MMB07, MBD$^+$05], in which energy consumption is roughly up to tens of nanojoules (hundreds of cycles) per allocation in 4x4 mesh network, our efficient hardware solutions that only cost tens of PicoJoules per allocation would be 1000X more energy efficient.

The comparison of different TESSA structures is shown in table 3.1. In conclusion, the folded structure is the most area efficient, while the bidirectional structure is the best in

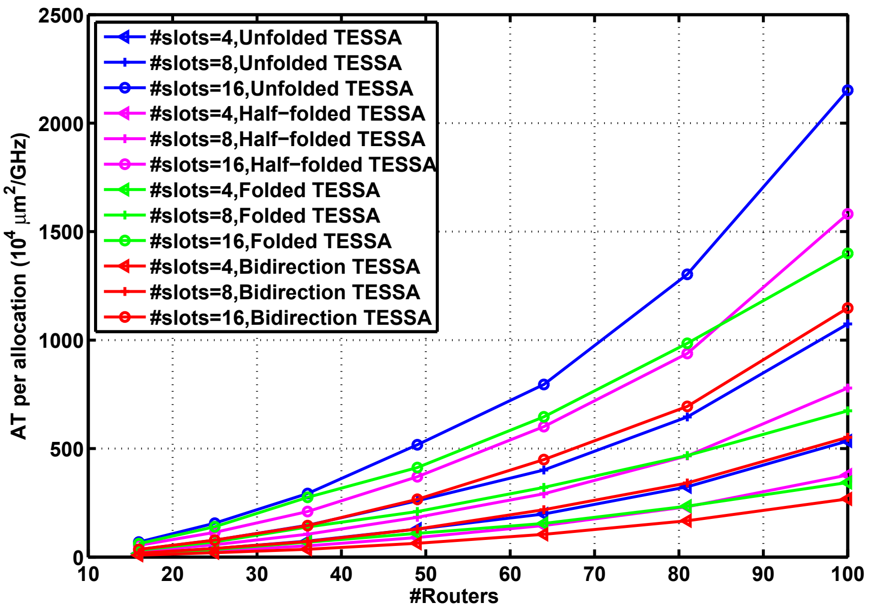


Figure 3.18: Average AT complexity per allocation of different TESSA in different size NoC with different slot table size

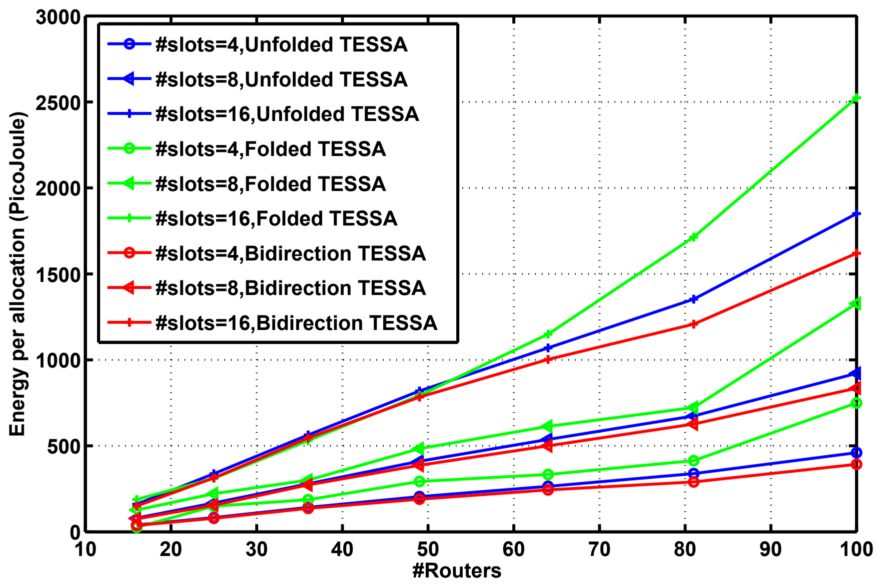


Figure 3.19: Average Energy consumption per allocation of different TESSA in different size NoC with different slot table size

terms of AT complexity and energy efficiency. Another advantage of bidirectional structure is its high allocation speed. Hence, in dynamic systems with high connection request rate, bidirectional structure will be more preferable. In different scenarios, different appropriate structures can be selected depending on specific system requirements.

Table 3.1: The comparison of different TESSA structures.

	Unfolded	Bidirectional unfolded	Folded	Half-folded
latency per allocation	low	lowest	high	average
complexity	O(S· M· $\sqrt{M}$)	O(S· M· $\sqrt{M}$)	O(S· M)	O(S· M· $\sqrt{M}$)
area cost	high	high	low	average
AT product	high	low	average	average
energy efficiency	good	good	poor	good

3.4.2 Simulation Results

In our TESSA, if a flow requires multiple slots, these multiple slots can be allocated along multiple paths, which can split the bandwidth over multiple paths to increase the success rate. In this section, in order to evaluate the influence of multi-path allocation of TESSA on success rate, we also realized a single path solution that employs the trellis graph algorithm for path search but allocates all required slots (bandwidth) on a single path. This solution is referred to as single path trellis. The trellis structure of single path trellis is the same as TESSA except the required bandwidth is allocated on a single path instead of multipath. The allocation speed and success rate of TESSA NoCManagers are compared to previous centralized and distributed allocation techniques for different NoC sizes with different slot table sizes under uniform random traffic. The source node sends the connection request to NoCM over dedicated wires, and the allocation information from NoCM to source is delivered via NoC as GS packet. We also evaluate the influence of splitting the link into different time slots, and allowing detours of different hops. These results are explained in the following sections.

For evaluation several performance metrics are used:

- *success rate* denotes the ratio of successful requests that established paths with sufficient bandwidth to the total requests.
- *background traffic* refers to the certain percentage of slots which are already randomly marked as occupied to exclude these slots from path search, same as in [SNG12]. In our experiments for each router, equal number of slots are occupied, but which slots are occupied is randomly selected.
- *allocation time* denotes the number of clock cycles that the algorithms need to find out a solution or to determine that the allocation is not possible.
- *total allocation time* denotes the number of clock cycles that the algorithms need to find the solution, in addition to the time to send the allocation information to source node.

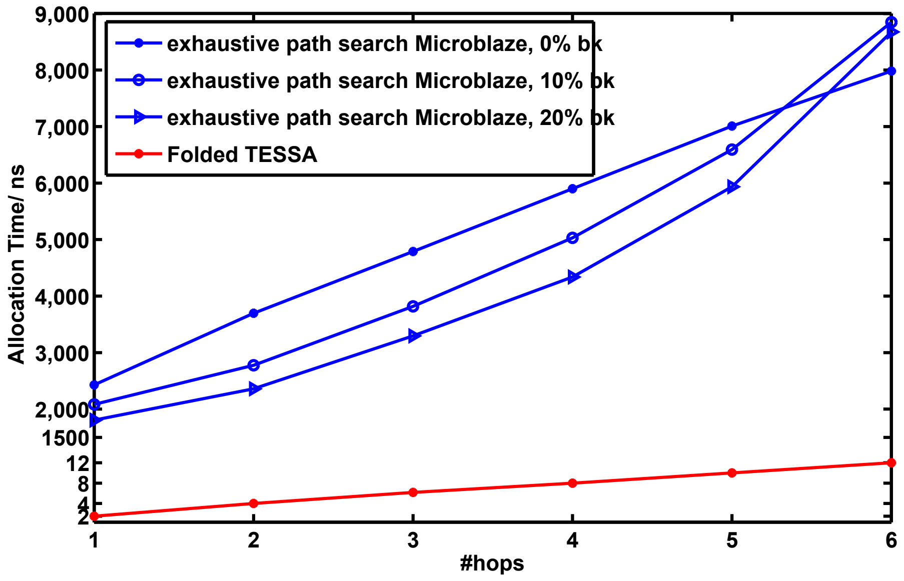


Figure 3.20: Allocation speed compared to Microblaze software-based approach[SNG12] with different background in 4x4 NoC with slot table size of 16.

- *GS offered load* refers to the required data transfer per connection multiplied by the connection request rate per master. Suppose the connection request rate per master is 1/2000 per cycle, and each connection can deliver 200 flits of data after setup, then the offered load is 200/2000=0.1 flits/cycle. It is a measure of the traffic each master offered compared to its maximum bandwidth.

3.4.2.1 Comparison with centralized exhaustive path-search

We compare folded TESSA against the single path exhaustive path-search that runs on Microblaze processor (@288 MHz) [SNG12] in this section.

Comparison of allocation speed We compare the allocation speed against the exhaustive path-search with 0%, 10% and 20% random background traffic (bk) in 4x4 mesh network. At each node, the exhaustive path-search algorithm has to try different directions one by one until gets the free path to next hop. Under heavy background traffic, it has to try more directions to get the free path. Hence, with different background traffic, the allocation time of exhaustive path-search is different. However, since the TESSA approach searches all directions simultaneously, all the possible paths are searched concurrently, and thus the allocation time of TESSA depends on the number of hops but is independent of background traffic.

From the Fig. 3.20, we can see the allocation time of the exhaustive path-search increases linearly with the length of the paths without background traffic (0% bk), which increases exponentially with background traffic (10% and 20% bk), while the allocation time of

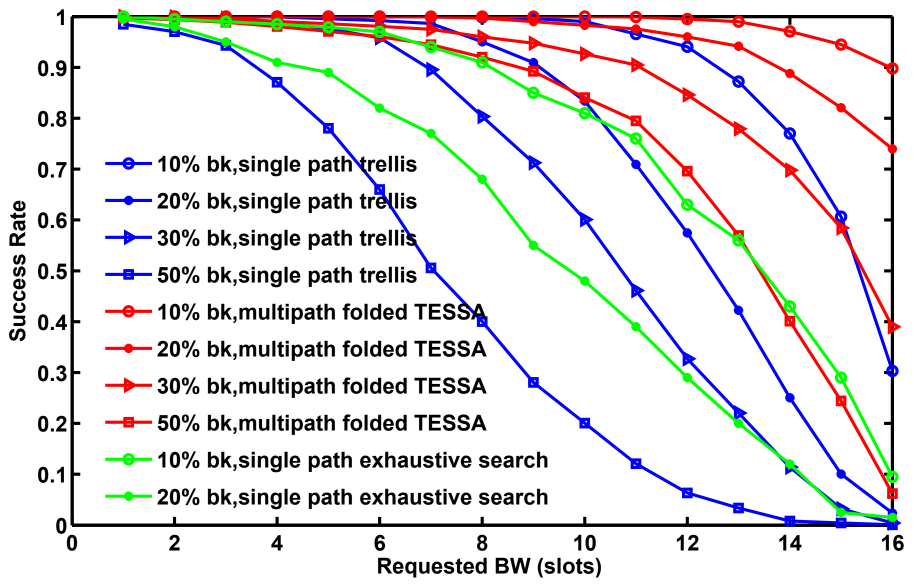


Figure 3.21: Success Rate compared to single path solutions in 4x4 NoC with different background traffic with Slot Table Size of 16.

TESSA always increases linearly with the length of the paths. The speed of TESSA is about hundreds to thousand times faster than exhaustive path-search. For 6 hops, 12 cycles (12 ns @ 1GHz) are needed for TESSA, i.e. 6 cycles needed for forward search and 6 cycles for backtracking, while 8848 ns are needed for exhaustive path-search [4] with 10% background traffic, which is 737 times faster. In the figure, it shows the exhaustive path-search with 20% background traffic has shorter allocation time than that with 10% background traffic. The reason is the higher background traffic induces lower success rate, and thus the algorithm will determine early that no route is possible.

Comparison of Success Rate The requests sent to NoCM are generated in this way: request to provide an allocation for every feasible source-destination pair combination with certain percentage of background traffic. We produce 1000 samples at each background traffic percentage. We do the simulation for 4x4 meshes with requested slots from 1 to 16 under background traffic from 10% to 50%.

The multipath folded TESSA, single path trellis, and single path software based exhaustive path-search [SNG12] are compared in Fig. 3.21 and Fig. 3.22. The searching algorithms of single path trellis and single path software approach are similar that the required bandwidth is allocated over single path, so in the scenarios where the software method's results are not provided, we can imagine it is similar to single path trellis's. The success rate of single path trellis is higher than the software method, which is due to the reason that it can detour when there is no minimal path, but in [SNG12] it only searches the minimal path.

[4] The hops in [SNG12] is adapted to the distance from router to router.

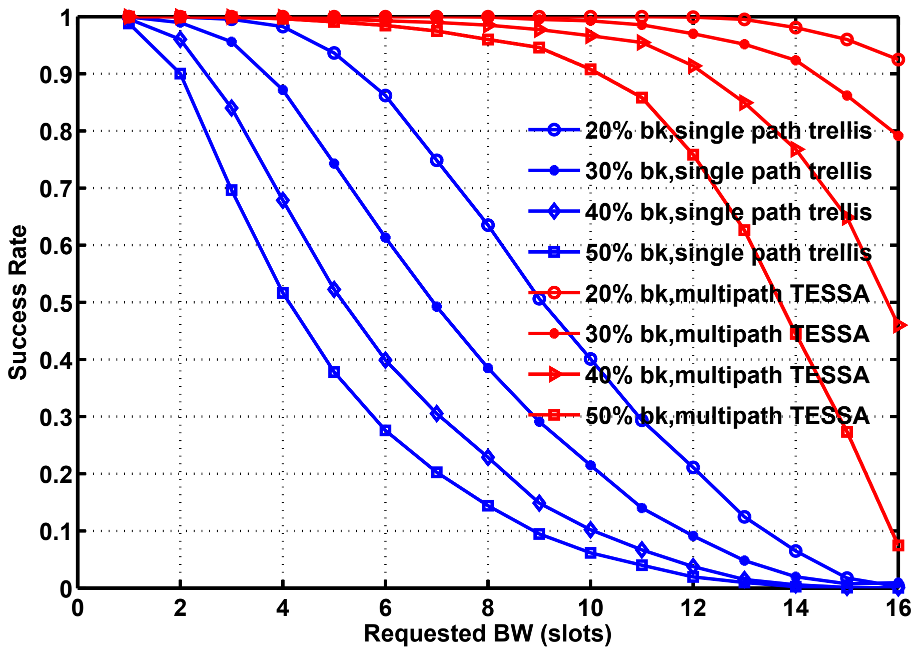


Figure 3.22: Success Rate compared to single path approach in 8x8 NoC with different background traffic with Slot Table Size of 16.

From Fig. 3.21 and Fig. 3.22, we can see the success rate decreases when requested bandwidth or background traffic increase, as expected. The success rate of folded TESSA can be several to hundred times higher than single path trellis, and it is even higher than single path exhaustive path-search. Under heavy background traffic with high requested bandwidth, the TESSA is far superior to the two single path solutions. For example, in 4x4 mesh with 16 requested slots, under 20% background traffic, the success rate of folded TESSA is 32X higher than single path trellis and 49X higher than exhaustive path-search, which increases to 103X higher than single path trellis under 50% background traffic. In 8x8 network with 16 requested slots, under 50% background, the success rate of TESSA is 0.074 while in single path trellis is only 0.0002, which is about 371 times higher.

3.4.2.2 Comparison with distributed parallel probe search

Bidirectional unfolded TESSA is compared to the state of the art distributed parallel probe search [LJL14b] in this section. In distributed parallel probe search, the source node sends a setup flit for searching path that traverses through the NoC along all minimal paths to try to reach target node. It is a flood-based algorithm which eliminates redundant incoming paths. Each point in the plot in the figures is obtained from simulation of 1 million cycles. The master issues a connection request to the NoCM in a uniform random traffic meeting the requirements of the GS offered load. The first and last 100,000 simulation cycles were not considered in order to prevent transient effects. The connection lifetime, i.e. the number of flits that each connection delivers, is set as 100 flits, 200 flits and 500 flits. During simulation, half of the nodes are assumed as masters

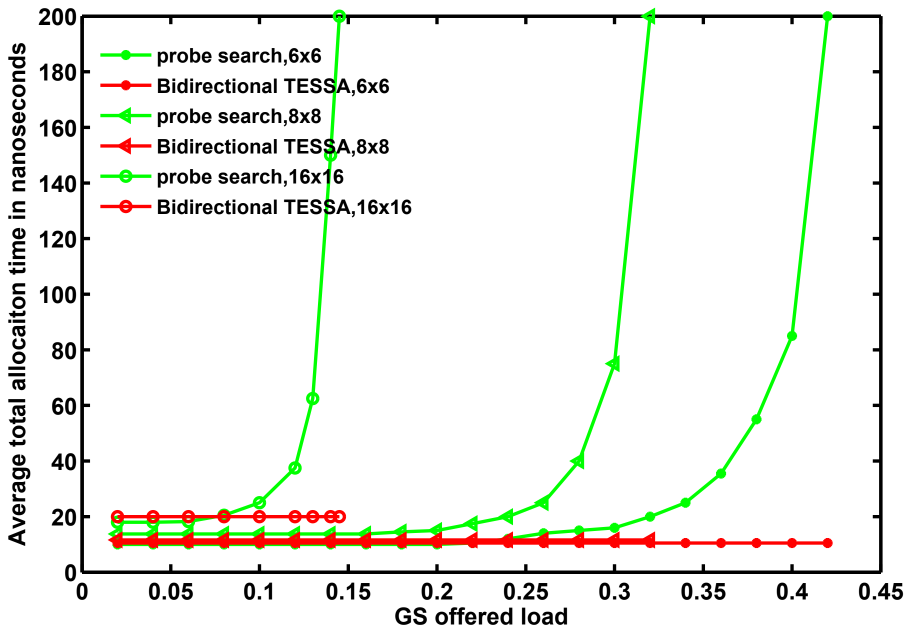


Figure 3.23: Allocation speed of bidirectional TESSA compared to probe search in different networks with different GS offered load with Slot Table Size of 16.

and half of the nodes are assumed as slaves. The master nodes are uniformly randomly distributed in the system. The source-destination pairs are uniform randomly selected that each source-destination pair has equal probability to be chosen.

Comparison of allocation speed In distributed parallel probe search, multiple trials might be needed before the eventual success of search due to the investigation of single slot at a time. On the contrary, in bidirectional TESSA, all slots are being searched in parallel, which completes the search in two clock cycles independent of the number of slots. Though our design needs additional time to send the allocation information to GS source, the path from NoCM to source node is found in two cycles by NoCM as a GS path. If the allocation of GS path from NoCM to source node fails, the allocation information will be sent to source as best-effort packets. Because in this simulation setting the GS offered load is not high (the offered load is lower than 0.145 in 16x16 mesh and the offered load is lower than 0.415 in 6x6 mesh), the corresponding allocation success rate is higher than 0.999, so the influence of the allocation failure of GS path from NoCM to source could be negligible.

For example, in 8x8 mesh with slot table size of 16 at GS offered load 0.3, the average total allocation time for single slot in probe search is 150 time slots ($\mathbf{0.5 \cdot 150 = 75\boldsymbol{ns}}$) [5]. However, in TESSA only 2 clock cycles each are needed for finding the requested GS

[5] time slot is the routing time per router, in [LJL14b] a slot is 0.5 ns. We can assume the time slots in both systems are the same.

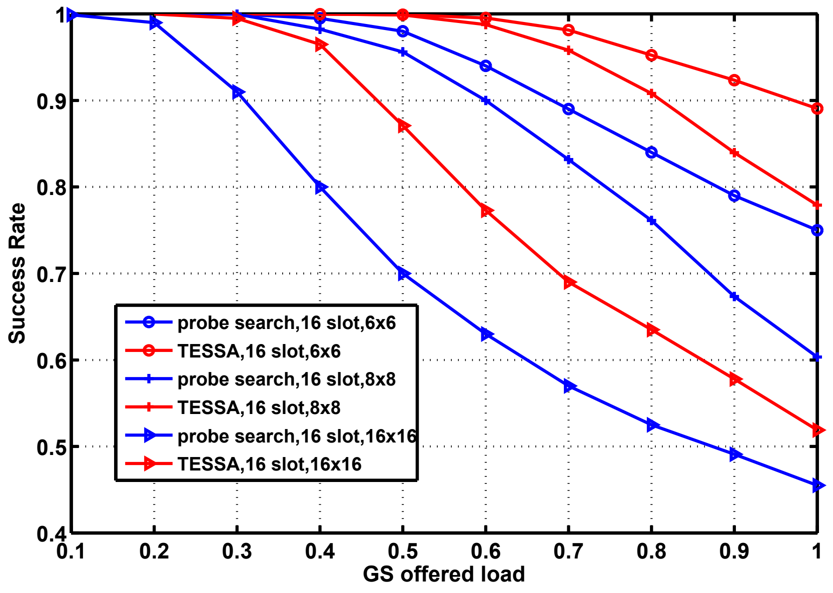


Figure 3.24: Success Rate of Bidirectional TESSA compared to probe search in different networks. Each connection delivers 200 flits.

path and for finding the path from NoCM to source. As NoCM is connected to the center node of the NoC, on average 4 time slots are needed by the NoCM to send the allocation information to source. With a slot table size of 16, there is on average 8 time slots waiting time at the NoCM to get its turn in the TDM scheme. So in total on average 12 time slots ($= \mathbf{4+8}$) in addition to 4 cycles (5.6ns @critical path 1.4ns) are needed, which is 11.6 ns ($= \boldsymbol{12 \cdot 0.5 + 5.6ns}$), which is 546% faster than distributed parallel probe search. In general, in $\boldsymbol{N \cdot N}$ mesh with slot table size of $\boldsymbol{S}$, the average allocation time is $\boldsymbol{4\ cycles + (\frac{N+S}{2}) time\ slots}$. When n slots are requested, the allocation time for probe search might be increased by n times, but the allocation time for TESSA will be the same as it is independent of the number of requested slots. Hence, when more slots are requested, our solution will present even better results.

Fig. 3.23 shows the average total allocation time when single slot is requested for different network sizes. When the GS offered load is low, the allocation speed of TESSA is similar to distributed parallel probe. However, when the offered load increases, the allocation speed of TESSA becomes much faster than distributed parallel probe. Compared to distributed parallel probe search, our approach can provide up to 710% higher speed in 6x6 mesh (@offered load 0.4), up to 647% higher speed in 8x8 mesh (@offered load 0.3), and up to 650% higher in 16x16 mesh (@offered load 0.14). In distributed parallel probe, after the saturation point of the network (offered load 0.14 in 16x16 mesh and offered load 0.41 in 6x6 mesh), the allocation time will increase dramatically. And consequently our approach will be far superior to distributed parallel probe.

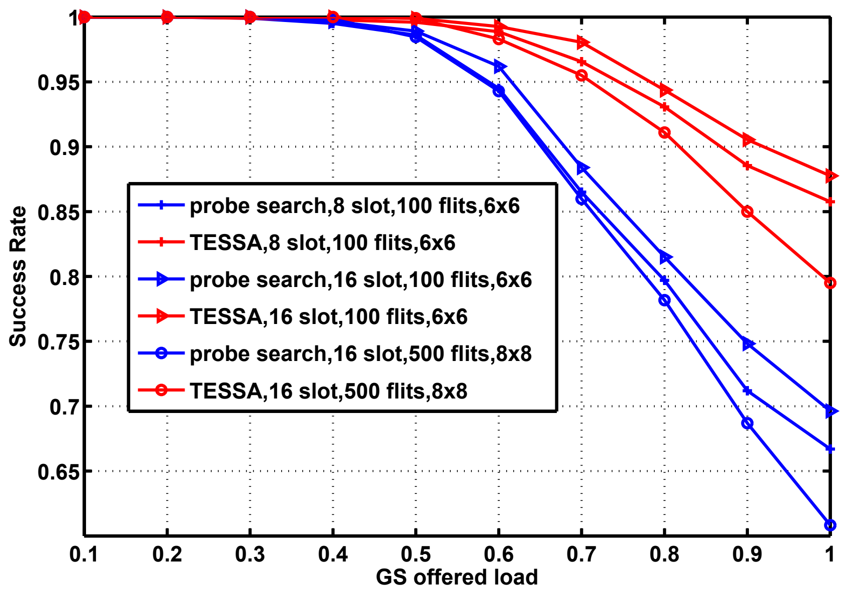


Figure 3.25: Success Rate compared to probe search in 6x6 and 8x8 mesh networks with 8 or 16 slot table size. Each connection delivers 100 or 500 flits.

Comparison of Success Rate We re-implement the distributed parallel probe search according to Liu's work [LJL14b] for comparison, with retry deadline as 200 cycles. In distributed parallel probe, each node is attached with a buffer to store the incoming requests, and the buffer size is equal to the slot table size. The source node keeps retrying a request until it succeeds or the deadline is exceeded, or the buffer is full.

In distributed parallel probe search, in which when several connections are requested simultaneously, the concurrent searches might block each other. It only searches the minimal path, and cannot make detours as in TESSA. Retry before deadline policy is employed, which can stop the search as failure before all slots are investigated even though there might be available paths. As in its simulation setting, the deadline is 200 cycles, and $\boldsymbol{2l + S + 6}$ cycles are needed for investigating single slot ($\boldsymbol{l}$ is the distance between source and destination, $\boldsymbol{S}$ is the slot table size). Now assume the $\boldsymbol{l}$ is 10, $\boldsymbol{S}$ is 16, so only four slots can be investigated before deadline, while in TESSA all 16 slots are investigated simultaneously. According to these factors, our system would have much higher success rate than distributed parallel probe search.

Fig. 3.24 and Fig. 3.25 shows the comparison results of success rate. From the simulation results we can see that the success rate of our method is higher than probe search's. E.g. in 6x6 NoC at offered load between 0.6 and 1.0 with slot table size of 16, our solution offers up to 26% higher success rate (@ connection lifetime of 100 flits). And our solution offers up to 29% and 24% higher success rate in 8x8 and 16x16 NoC, respectively. From Fig. 3.25 we can see with more slots (16 slots against 8 slots), the success rate becomes higher, which is due to the increased path diversity.

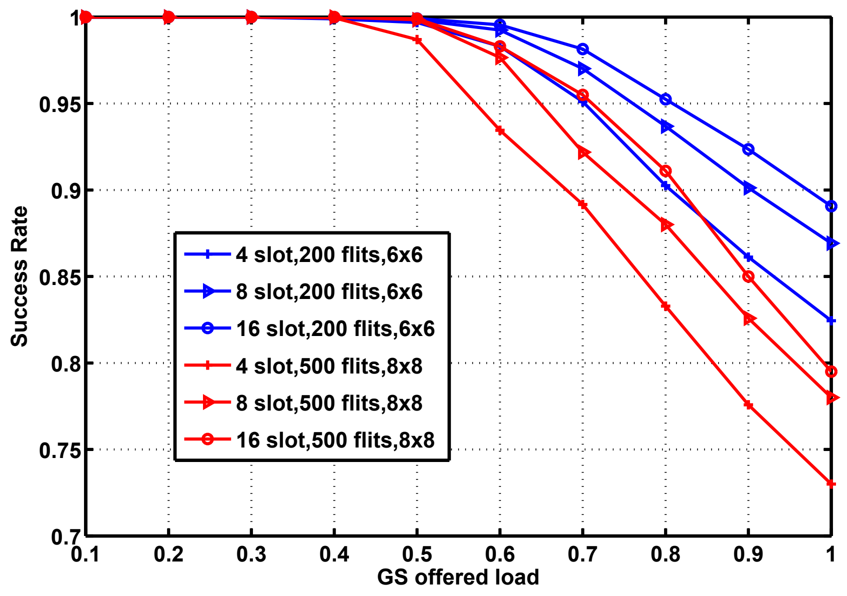


Figure 3.26: Success Rate influence of split link into different slots in 6x6 and 8x8 networks. Each connection delivers 200 or 500 flits.

3.4.2.3 Influence of splitting the link into different time slots on success rate

In this section, we split the link into different time slots, e.g. 4 slots, 8 slots and 16 slots, for 6x6 and 8x8 mesh networks to see the influence on success rate. In the simulation setting, each connection delivers 200 flits before release in 6x6 mesh and delivers 500 flits before release in 8x8 mesh. As the results in Fig. 3.26 show, when the link is split into more time slots, it can provide higher success rate. For 6x6 mesh, when the link is split into 16 slots, it can provide up to 9% higher success rate than 4-slot split, and up to 3% higher than 8-slot split. For 8x8 mesh, when the link is split into 16 slots, it can provide up to 10% higher success rate than 4-slot split, and up to 3.5% higher than 8-slot split. The reason for this is when the link is split into more time slots, it can provide higher path diversity, which can contribute to the higher success rate. For example, if the link is split into $\boldsymbol{S}$ time slots, it can provide up to $\boldsymbol{S}$ different options to route over this link, and can support up to $\boldsymbol{S}$ flows simultaneously.

3.4.2.4 Influence of allowing different hops of detours on success rate

In this section, we evaluate the influence of allowing different hops of detours on success rate for 6x6 and 8x8 mesh networks. In the simulation setting, the unfolded trellis is constructed as 10, 12, 14 and 16 stages for 6x6 mesh to allow 0, 2, 4 and 6 more hops detours than the default trellis (by default, unfolded trellis is constructed as $\boldsymbol{2N-2}$ for $\boldsymbol{N \cdot N}$ mesh network, so 10 stages for 6x6 mesh), and it is constructed as 14, 17 and 20 stages for 8x8 mesh to allow 0, 3 and 6 more hops detours than the default trellis.

As the simulation results in Fig. 3.27 and Fig. 3.28 show, when more hops of detours

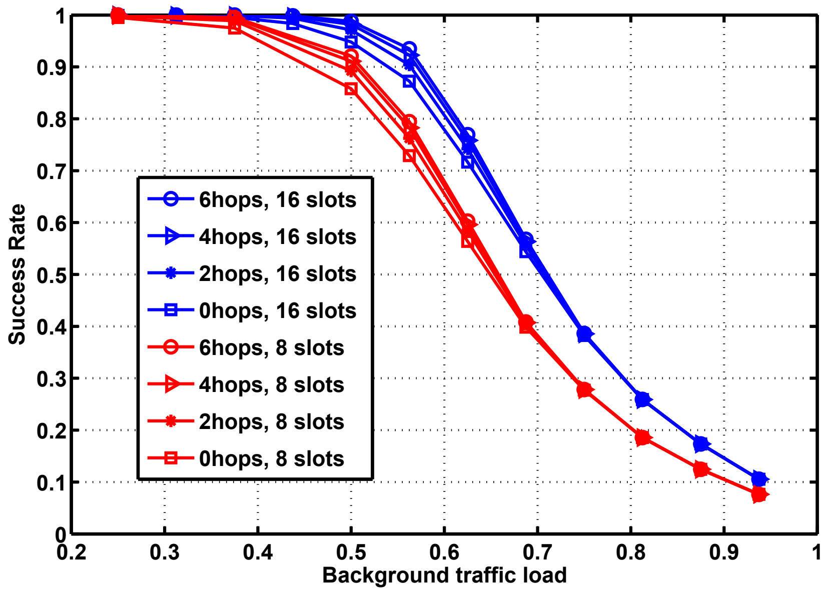


Figure 3.27: Success rate under different hops of allowed detours in 6x6 network with 8 or 16 slot table size.

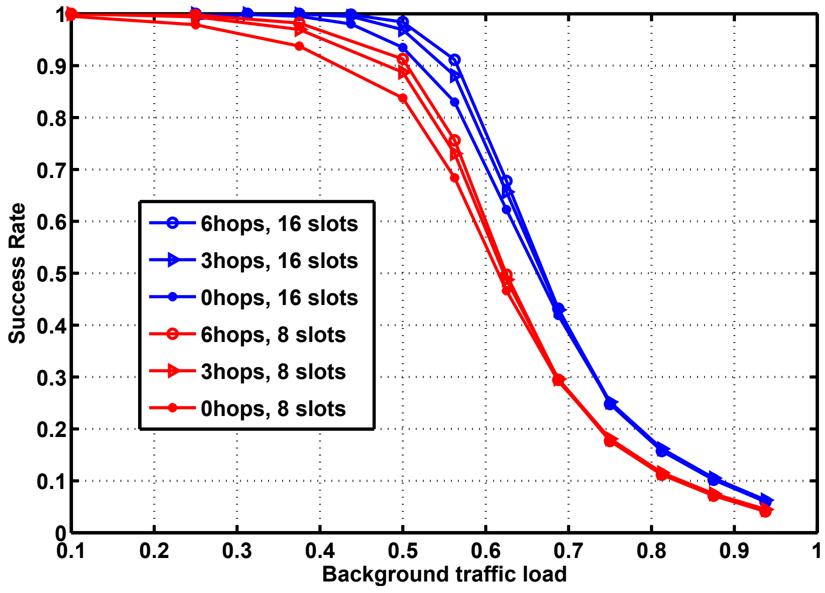


Figure 3.28: Success rate under different hops of allowed detours in 8x8 network with 8 or 16 slot table size.

allowed, it can provide higher success rate. For 6x6 mesh, with 8 time slots, the 16 stages trellis (6 more hops detour allowed) can provide up to 4% higher success rate than 12 stages trellis (2 more hops detour allowed), and up to 10% higher success rate than 10 stages trellis (0 more hops detour allowed); with 16 time slots, the 16 stages trellis (6 more hops detour allowed) can provide up to 4% higher success rate than 12 stages trellis, and up to 7.5% higher success rate than 10 stages trellis. For 8x8 mesh, with 8 time slots, 20 stages trellis (6 more hops detour allowed) can provide up to 10% higher success rate

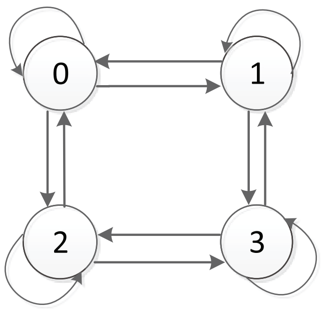


Figure 3.29: 2x2 2D-mesh example NoC.

than 14 stages trellis (0 more hops detour allowed); with 16 time slots, 20 stages trellis can also provide up to 10% higher success rate than 14 stages trellis.

3.5 Register-Exchange Trellis Search

In previous forward-backtrack trellis search, the path search is divided into two steps: forward search and backtrack. In this section, we present the register-exchange trellis that saves the entire survivor path during the forward search, so that when the destination node is reached during the forward search, the entire survivor path can be read out from the destination node directly [Kam07, Fet95]. Consequently, the backtrack is omitted and the search time is halved compared to previous forward-backtrack approaches, which can also contribute to the allocation success rate.

3.5.1 Register-Exchange Trellis Path Search Algorithm

In Register-Exchange (RE) trellis search, each state forwards the entire survivor path (i.e. the path from the initial state to the current state) to neighbors at the next stage. The entire information sequences of the survivor paths are then continuously updated during path search. As long as the target state is reached, the entire survivor path sequence can be read directly from the target state. The survivor path is the shortest contention-free path between initial state and target state, which is to be found. The RE trellis can also adopt the unfolded structure, folded structure and bidirectional structure. In this section, we take the unfolded RE trellis and folded RE trellis as example.

3.5.1.1 Unfolded Register-Exchange Trellis Path Search

The unfolded trellis graph of example network in Fig. 3.29 is illustrated in Fig. 3.30. The search begins at the source node at the initial stage, traversing through the trellis to try to activate its all connected neighbors. The neighbors can be activated only if the associated incoming branch is available. The new active node will continue the propagation search to

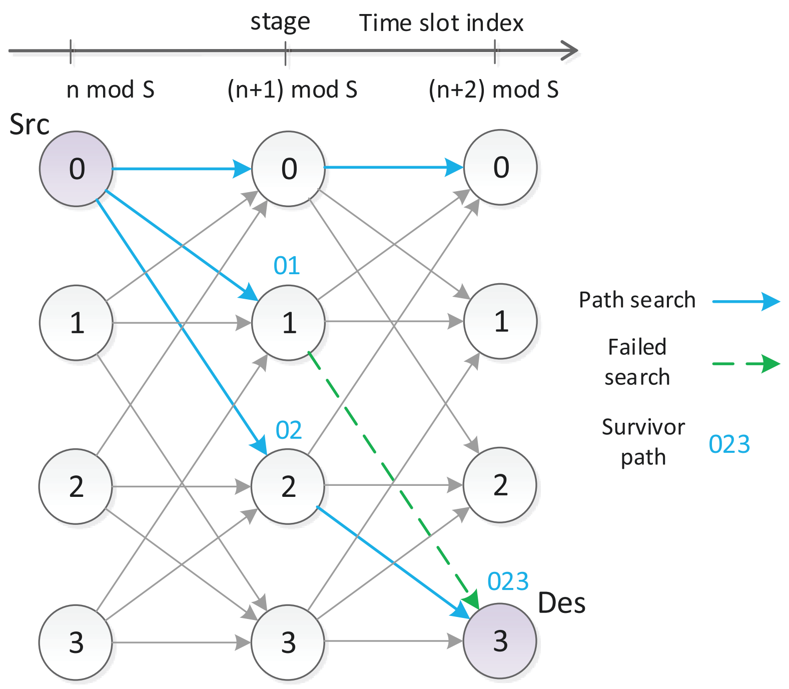


Figure 3.30: Unfolded RE trellis path search. The survivor path is read directly from destination node without backtrack.

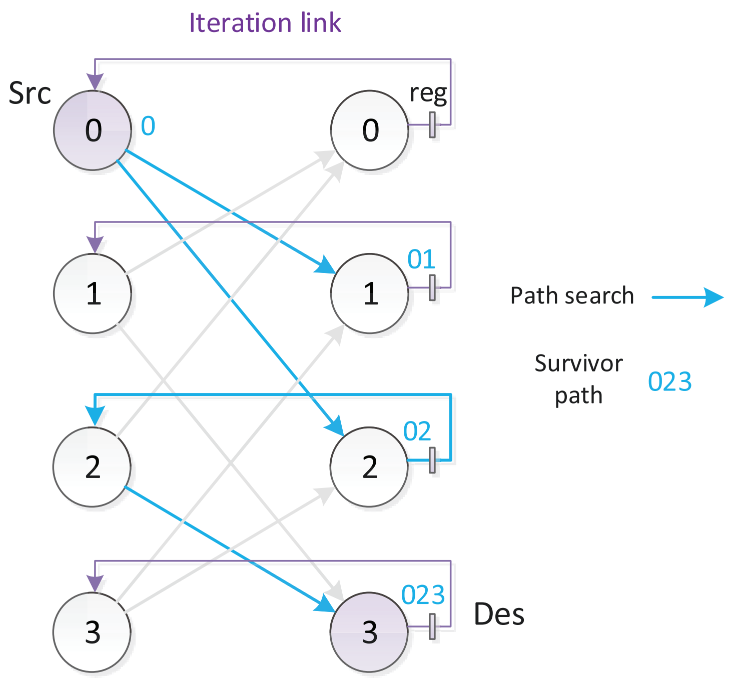


Figure 3.31: The folded RE trellis search graph of the example NoC.

activate its neighbors, and forwards the associated survivor path sequence to its neighbors. The search will be stopped when the last stage is reached.

In Fig. 3.30, assume node 0 is source node and node 3 is destination node. The search starts from source and activates its neighbors (node 2 and node 1), and the activated

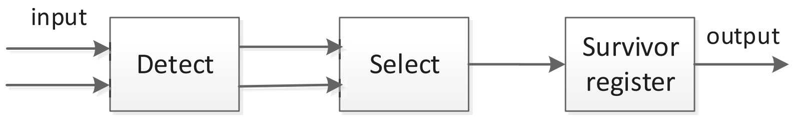


Figure 3.32: Block diagram of single state.

neighbors update their survivor path sequence (node 2 as $\{\mathbf{02}\}$ and node 1 as $\{\mathbf{01}\}$). The active neighbors continue the propagation, e.g. node 2 activates node 3, and then node 3 updates its survivor path sequence as $\{\mathbf{023}\}$. Simultaneously, the node 1 also tries to reach node 3. Assume the branch $\boldsymbol{N1 \rightarrow N3}$ is already allocated by previous allocation, then node 3 cannot be activated by node 1 at this time. When target node (node 3) is reached, the survivor path is read directly from target node as $\boldsymbol{N0 \rightarrow N2 \rightarrow N3}$. Assume the beginning slot at source is t, then we can obtain the slot sequence along the path as $\{\boldsymbol{t}, \boldsymbol{(t+1) \bmod S}, \boldsymbol{(t+2) \bmod S}\}$.

3.5.1.2 Folded Register-Exchange Trellis Path Search

Proposed unfolded trellis graph can be efficiently mapped on the folded structure. To enable iterative traversals to represent multi-hop search, there is 'iteration link' from second stage to first stage. A register is assigned to each state that stores the survivor path. At the target state, as long as it is reached, the corresponding survivor path is the one stored in the survivor path register. Each iteration consumes a clock cycle. The folded structure can reduce the resource cost significantly, while consuming longer search time.

The folded path search algorithm of example network in Fig. 3.29 is illustrated in Fig. 3.31. The search begins at the source node of the initial stage, traversing through the trellis to try to activate its connected neighbors. When a node is active, its associated register will remember the entire survivor path sequence. At the next cycle the active node will travel back to its first stage, and does the propagation search again. The search will be stopped in two cases: i) either the target node has been reached or ii) after certain iterations (by default, $\boldsymbol{2N-2}$ iterations for NxN mesh NoC). Hence, the livelock is avoided. As shown in Fig. 3.31, the search signals started from source and activated node 2, then node 2 updated its survivor path register as $\{\mathbf{02}\}$. At the next cycle, node 2 continued to activate node 3, and node 3 updated its survivor path register as $\{\mathbf{023}\}$. When node 3 is reached, the path is read from its survivor register as $\boldsymbol{N0 \rightarrow N2 \rightarrow N3}$.

3.5.2 Trellis Path Search Implementation

The implementation block diagram of single state of a RE trellis can be divided into three basic units, as shown in Fig. 3.32. The input data is used in the detect unit to detect which incoming branches are active. These are then fed to the select unit which selects one of the incoming branches as *survivor path*, and accumulates the survivor path. Thereafter,

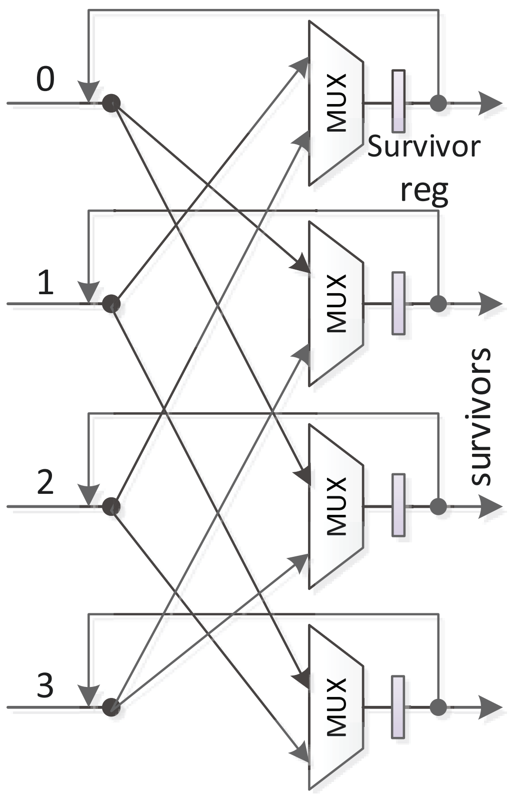


Figure 3.33: Implementation schematic of the folded RE trellis

the survivor register saves the accumulated survivor path, and outputs it to neighbors at next stage.

The implementation schematic structure of Fig. 3.31 for the example 2x2 mesh NoC is shown in Fig. 3.33. The survivor path scales with the number of stages and nodes, and associated storage is necessary. Each state has a register (of size $\boldsymbol{(2N-2)\cdot\log_2 N^2}$ bits for NxN mesh) to store the whole survivor path. There is also a 'branch state' register for each branch, which stores the actual branch allocation state. When a branch is allocated at a specific slot, its corresponding state register is set to '0' excluding it thereby from future search. Correspondingly, state register is set to '1' when the branch is released. For each state, one and only one of the available incoming branches is selected as survivor path and updated in the survivor register. The detection of the available incoming branch is as: if the search signal from source node reaches this branch as well as ('AND' operation) its corresponding 'branch state' register is valid, this incoming branch is available. The time slot shift (slot shifts from $\boldsymbol{t}$ to $\boldsymbol{(t+1)\ \mathbf{mod}\ S}$ after each stage) is realized by wire connection. When the target node is reached, the entire path sequence is read directly from its associated survivor register.

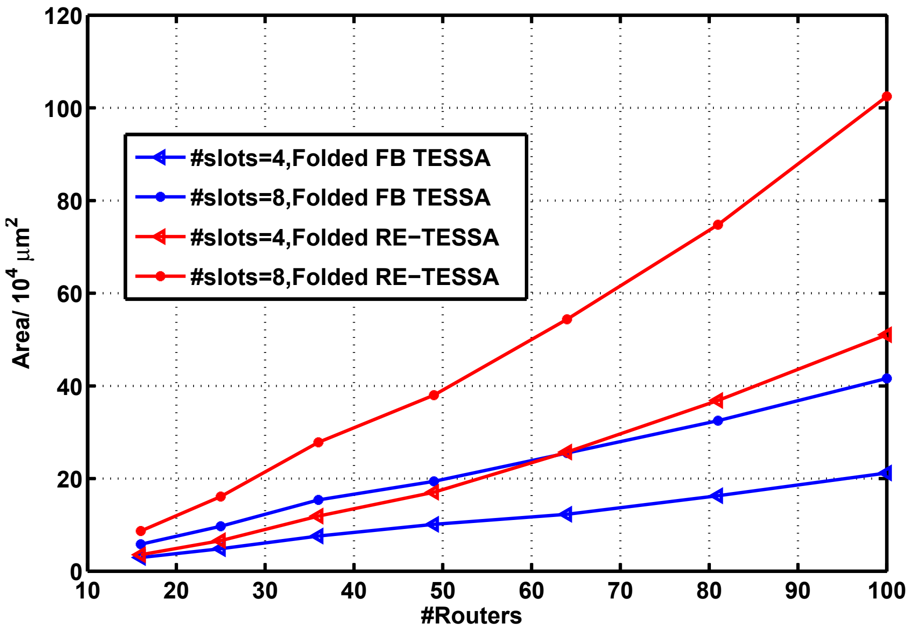


Figure 3.34: Area of folded FB and folded RE NoCManagers in different size NoCs with different slot table sizes.

3.5.3 Performance Evaluation

In this section, we present the synthesis and simulation results of folded RE trellis against the folded FB trellis and against distributed parallel probe search [LJL14b].

3.5.3.1 Synthesis Results

The NoCManager is available in synthesizable VerilogHDL and can be generated out of an XML description for different NoC sizes. Using Synopsys Design Compiler, the NoCM was synthesized with TSMC 65 nm technology, for different mesh networks of size 4x4 to 9x9. Both the FB and RE folded TESSA NoCM were synthesized with 1 GHz clock frequency constraints. [6]

From the area consumption shown in Fig. 3.34, we can see the area of RE TESSA grows with $O(S\cdot M)$ in 2D-mesh (M= #routers, S= #slots). Since the RE-TESSA has to use a larger register file (of size $\boldsymbol{N\cdot(2N-2)\cdot\log_2 N^2}$ bits) , its area is about twice as the folded FB TESSA. The average AT complexity in Fig. 3.35 shows, when the NoC is small (smaller than 7x7 mesh), the AT complexity of RE and FB TESSA is similar. However, when the network size grows, the AT complexity of FB TESSA becomes better than RE approach. The average energy consumption per allocation in Fig. 3.36 shows, RE TESSA consumes less energy than FB TESSA in small NoC (smaller than 8x8 NoC), while consuming more energy in large NoC. The reason for this is, the RE TESSA takes less search time as

[6] Since in RE TESSA there is more data to be forwarded to next stage, for large NoC, the clock frequency of RE TESSA could be lower than FB TESSA.

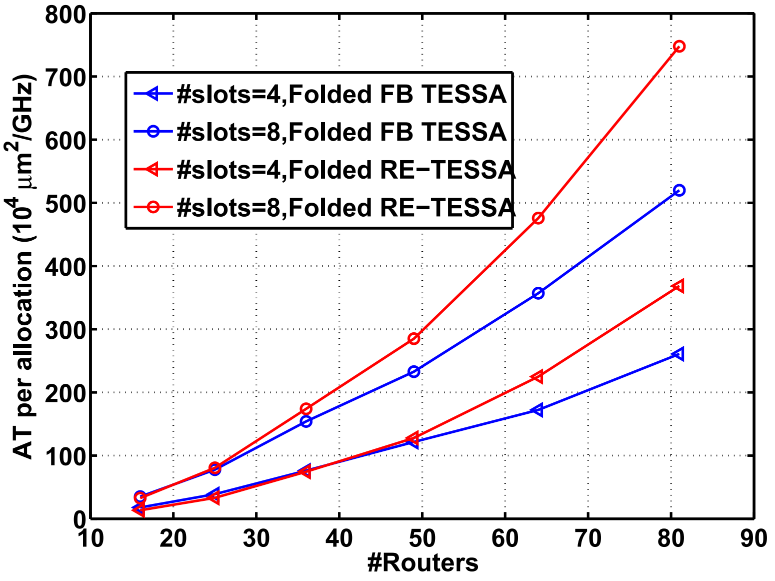


Figure 3.35: Average AT complexity per allocation of RE and FB NoCManagers in different size NoCs with different slot table sizes.

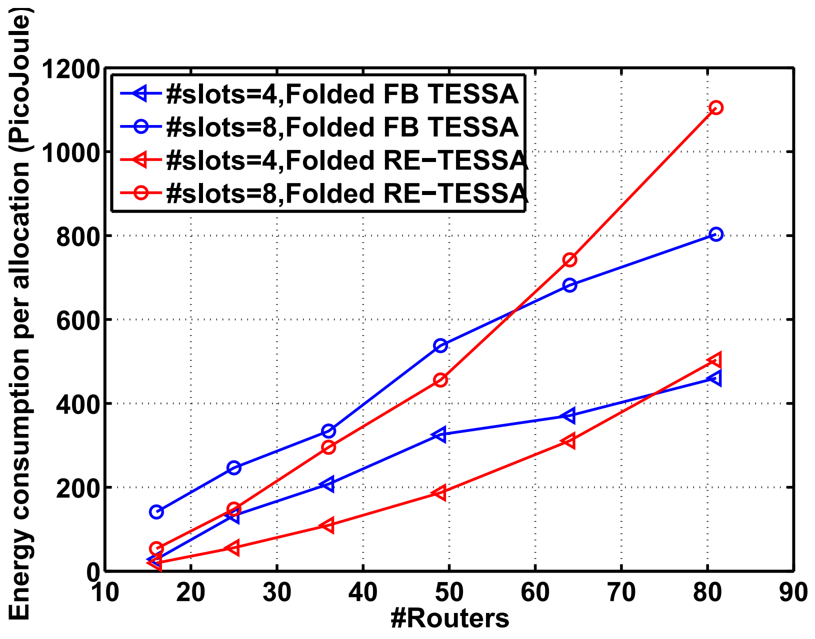


Figure 3.36: Average Energy consumption per allocation of RE and FB NoCManagers in different size NoCs with different slot table sizes.

there is no backtrack, so it could offer better energy efficiency than FB TESSA. But when the NoC size increases, the survivor registers in RE TESSA increase dramatically, so the energy consumption in RE TESSA increases faster than in FB TESSA. After certain point, the energy consumption in RE TESSA could be higher than FB TESSA. However, due to

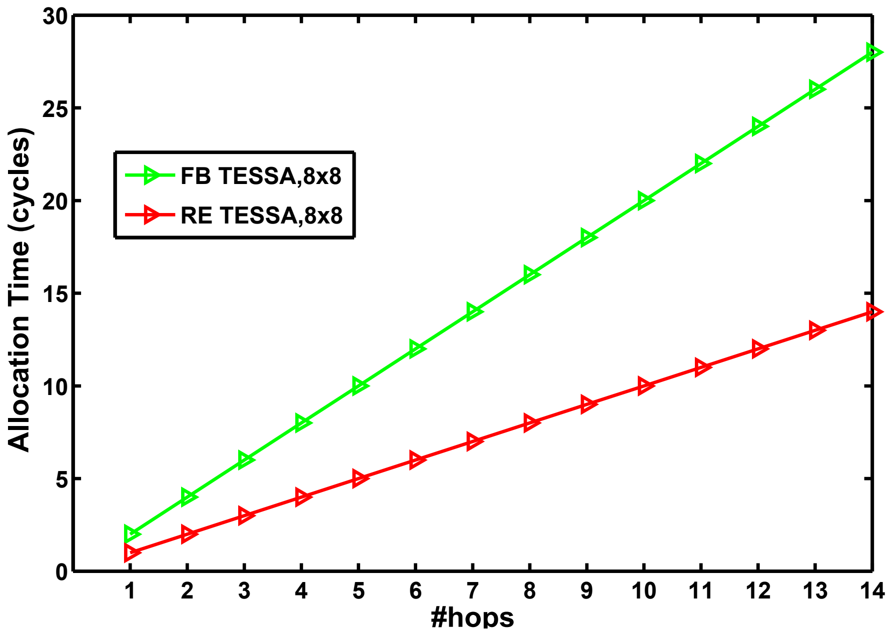


Figure 3.37: Allocation speed comparison between RE TESSA and FB TESSA.

the *partitioning architecture* idea that divides the large system into multiple small logic partitions with multiple managers (explained in section 3.7), each NoCM only manages limited number of nodes in its local region, and thus the energy consumption of RE TESSA could be better than FB approach (if each partition size is smaller than 8x8).

3.5.3.2 Simulation Results

The allocation speed and success rate of RE TESSA NoCManagers are compared to the FB TESSA and distributed parallel probe search [LJL14b] under uniform random traffic. The request queue sizes of FB and RE NoCMs are both 64 flits deep, and larger queue size did not show significant performance improvements in our continuous simulations. We also re-implement a distributed parallel probe connection setup approach according to Liu's work [LJL14b] for comparison, with a retry deadline of 300 clock cycles attached to each request. Each node is attached with a buffer to store the incoming requests, with the buffer size equal to the slot table size.

The performance metrics, *success rate*, *allocation time* and *GS offered load* are explained in previous section.

Any data point that is shown in the figures comes from simulation of 1 million cycles. The NoC issues a GS connection request to the NoCM in a uniform random traffic meeting the requirements of the GS offered load. The first and last 100,000 simulation cycles were not considered in order to prevent transient effects. The connection lifetime, i.e. the number of data flits transmitted over established connection, is set as 100 flits, 200 flits, 300 flits and 500 flits.

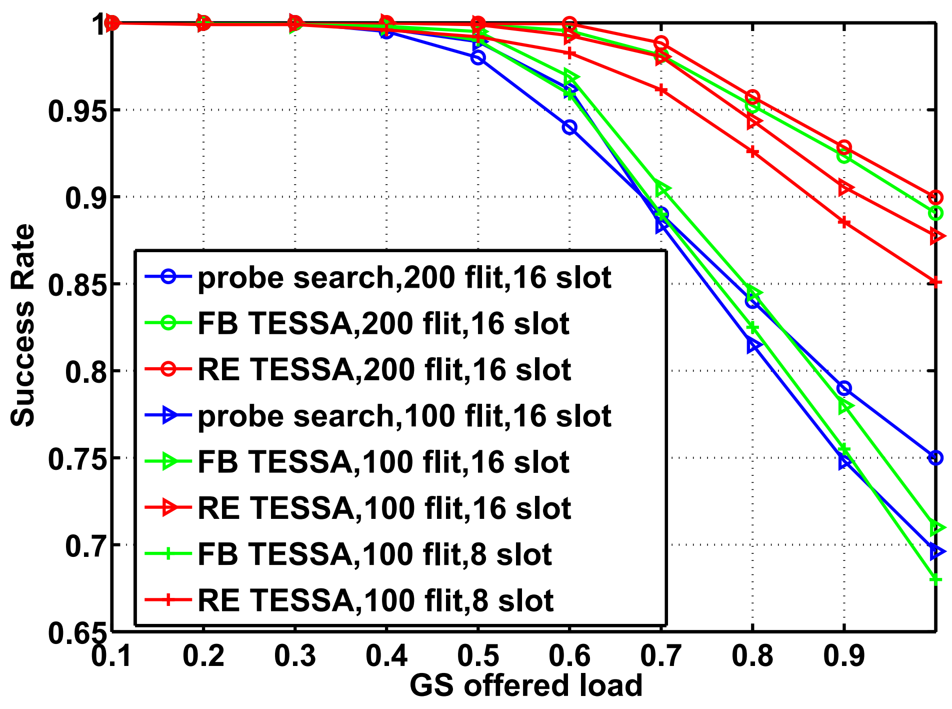


Figure 3.38: Success Rate compared to FB TESSA and probe search in 6x6 network with Slot Table Size of 16 and 8. Each connection delivers 100 or 200 flits.

Comparison of allocation speed Since RE TESSA omitted the backtrack step while which is necessary in FB approach, its allocation speed is twice as in FB TESSA. The RE TESSA needs one cycle to traverse single hop through the network, while two cycles are required in FB approach, i.e. one cycle for forward search and one cycle for backtrack. Since the RE TESSA has to forward the entire survivor path during the forward search, it may have longer critical path than FB approach. However, until 8x8 mesh, the critical path of RE TESSA is constrained to 1 ns, which is as the same as in FB approach. Due to the *partitioning architecture*, each NoCM only manages limited number of nodes in its local region, and thus the critical path of RE TESSA would not increase too much.

As shown in Fig. 3.37, to traverse $\boldsymbol{n}$ hops, for RE TESSA, only $\boldsymbol{n}$ cycles are needed in NoCM to find the path, but $\mathbf{2} \cdot \boldsymbol{n}$ cycles are necessary in FB TESSA to find the path. Hence, the allocation speed of RE TESSA is doubled over FB approach.

Comparison of Success Rate In distributed parallel probe search, in which when several connections are requested simultaneously, the concurrent search flits might block each other. Moreover, it only searches the minimal path, cannot detour as in TESSA. For

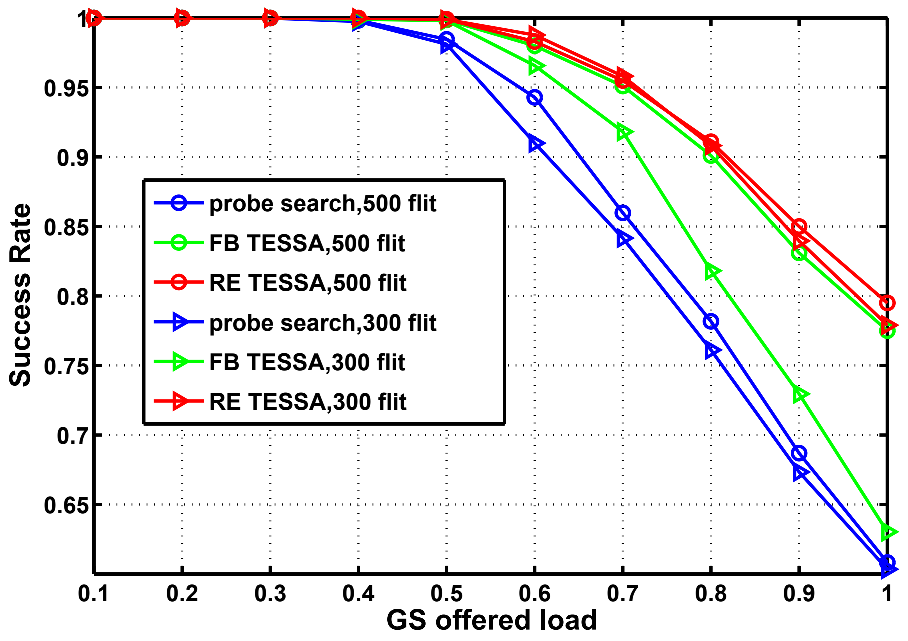


Figure 3.39: Success Rate compared to FB TESSA and probe search in 8x8 network with Slot Table Size of 16. Each connection delivers 300 or 500 flits.

example, in Fig. 3.38 in 6x6 NoC, RE TESSA can offer up to 25% higher success rate than distributed parallel probe search. Compared to FB TESSA, when each connection delivers 200 flits, the success rate of FB and RE TESSA are similar; while when each connection delivers 100 flits, the RE approach can achieve up to 22% higher success rate. The reason for this is there are more new requests to NoCM when connection delivers 100 flits than connection delivers 200 flits for the same offered load. Therefore, many requests arriving at the low speed FB NoCM must be rejected immediately because of a full request queue although there might be a free path for this request. In Fig. 3.39, in 8x8 NoC, RE TESSA can offer up to 33% higher success rate than distributed parallel probe search. When each connection delivers 300 flits, RE approach can provide up to 22% higher success rate than FB TESSA.

Comparison of Area.Time/Success Rate In previous section, we have shown the AT complexity. However, due to the success rate that some failed allocations have to be dropped, the effective allocation time per allocation is increased according to $\boldsymbol{T_{eff}} = \frac{\boldsymbol{T}}{\boldsymbol{S}}$, where $\boldsymbol{S}$ is success rate. Therefore, the effective $\boldsymbol{AT_{eff}} = \boldsymbol{A}\frac{\boldsymbol{T}}{\boldsymbol{S}}$. This can be further decomposed into true cost and overhead:

$$\boldsymbol{AT_{eff}} = \boldsymbol{AT}(1 + \frac{\boldsymbol{E}}{\boldsymbol{S}}) = \boldsymbol{AT} + \boldsymbol{AT}\frac{\boldsymbol{E}}{\boldsymbol{S}}$$

Note:

$$\frac{\boldsymbol{1}}{\boldsymbol{S}} = \frac{\boldsymbol{S}+\boldsymbol{E}}{\boldsymbol{S}} = 1 + \frac{\boldsymbol{E}}{\boldsymbol{S}}$$

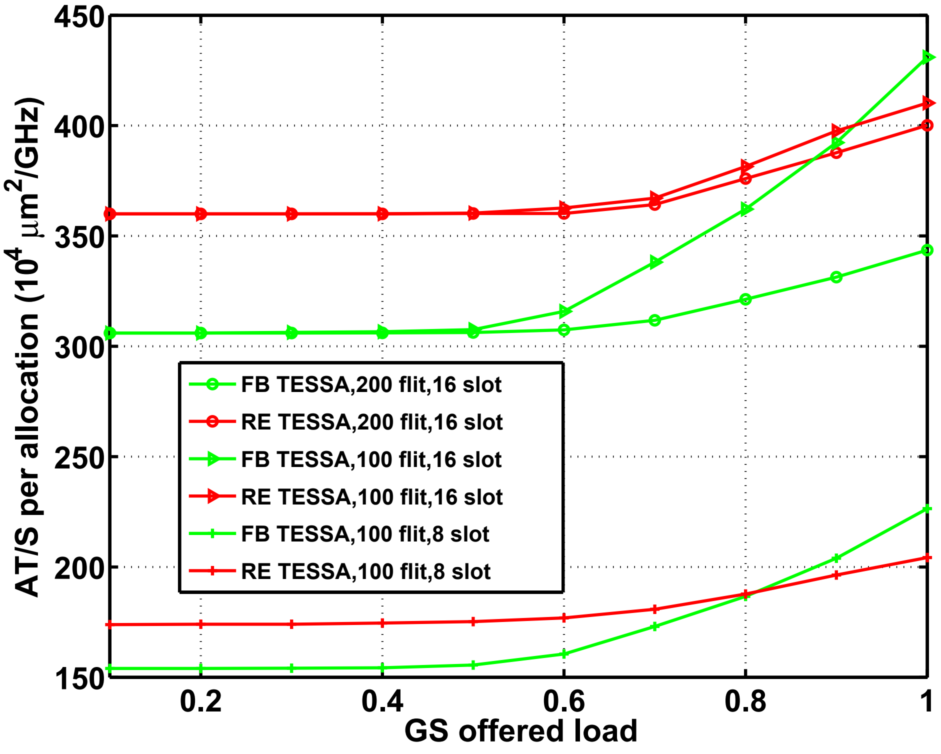


Figure 3.40: Average Area.Time/Success Rate (per allocation) in 6x6 network with Slot Table Size of 16 and 8. Each connection delivers 100 or 200 flits.

Where $\boldsymbol{E}$ is the error rate. Hence, the overhead scales with $\frac{\boldsymbol{E}}{\boldsymbol{S}}$ and this ratio is specific for each algorithm and implementation.

In this section, we show the measurement of AT/S (Area.Time/Success Rate) for FB and RE TESSA in 6x6 mesh, as shown in Fig. 3.40. The results show when the offered load becomes higher, since the corresponding success rate becomes lower, the value of AT/S becomes higher. When each connection delivers 200 flits, the AT/S of FB TESSA is better than RE TESSA. When each connection delivers 100 flits, if the offered load is higher than 0.9, the AT/S of RE TESSA would be better than FB TESSA.

3.6 Single Layer Trellis

In previous sections, every slot at the initial stage has its own layer of the trellis graph, i.e. if the size of the slot table is S, there are S layers of trellis graph in parallel, as in Fig. 3.42. We name this as multi-layer approach. In this section, we present the single-layer approach, in which only one layer of the trellis graph need to be implemented. Moreover, all slots at

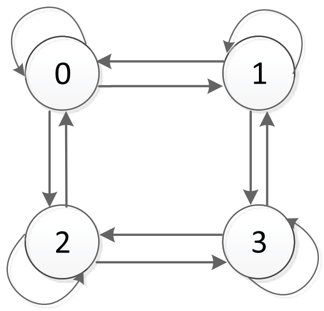


Figure 3.41: 2x2 example NoC.

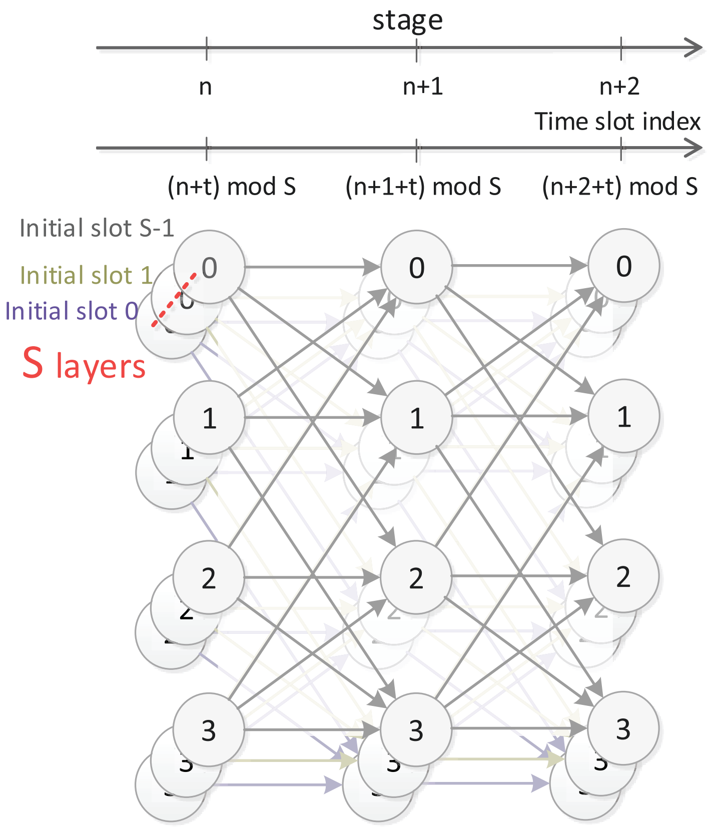


Figure 3.42: Multiple-layer trellis of the example NoC. Each slot at the initial stage has its own layer.

the initial node (source node) can share the single layer, and can search simultaneously in the single layer. Compared to previous approaches that multiple layers of trellis graph has to be implemented, the hardware consumption is reduced dramatically.

3.6.1 Single-layer Trellis Path Search Algorithm

The advantage of the multi-layer approach is that every slot from the source node can search its own path in parallel, so it can allocate several slots to a single flow simultaneously. However, in some scenarios maybe most flows only need one portion of the link

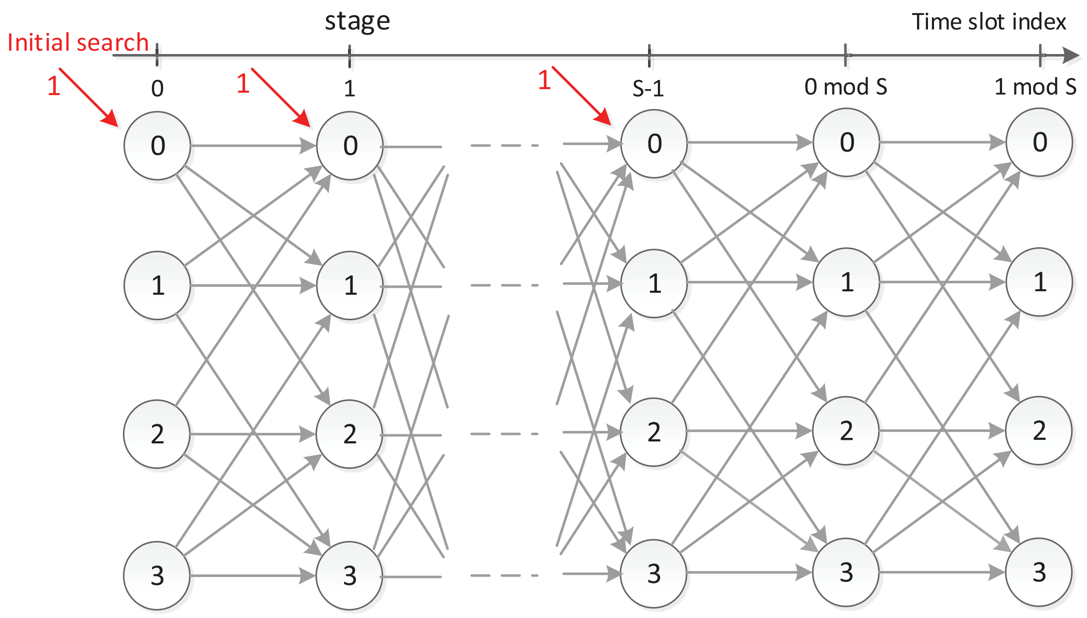


Figure 3.43: General schematic structure of the single-layer approach. The slot table size is S, so there are S-1 additional stages. The first S stages are associated with S time slots, which can launch S initial searches simultaneously at the source node.

bandwidth, i.e. one time slot, so the allocation of several slots in parallel to single flow may become unnecessary. Consequently, we do not always need to implement S layers of trellis graph, but sometimes we can only implement single layer of trellis to be shared by all slots. We name this as single-layer approach.

The single-layer approach is shown in Fig. 3.43. There is only one layer of trellis graph that is shared by all slots. Now, the trellis is constructed as $\boldsymbol{S-1+2N-2}$ stages for NxN mesh NoC, i.e. $\boldsymbol{S-1}$ stages longer than the default unfolded trellis. The first S stages of the trellis are used to launch S initial searches at source node, that each stage is associated with a time slot and can launch a search. Hence, all slots at the source node can search simultaneously in the single layer.

An example of single-layer is shown in Fig. 3.44 for the example 2x2 mesh network. Node 0 is the source and node 3 is the destination. The size of the slot table is 2, so the trellis has one additional stage than the default unfolded trellis. At the first stage, the slot 0 from the source starts the search and activates node 2. At the second stage, the slot 1 from the source also sends out a search and activates node 1. The searches are forward propagated until reaching the last stage. At the last stage, the destination are activated by both node 1 and node 2, and the node 1 is selected by the destination as the predecessor. At the next cycle, the backtrack starts to select the survivor path as $\boldsymbol{N0 \rightarrow N1 \rightarrow N3}$, and the corresponding slot sequence along the path is acquired as $\{\mathbf{1}, \mathbf{0}, \mathbf{1}\}$.

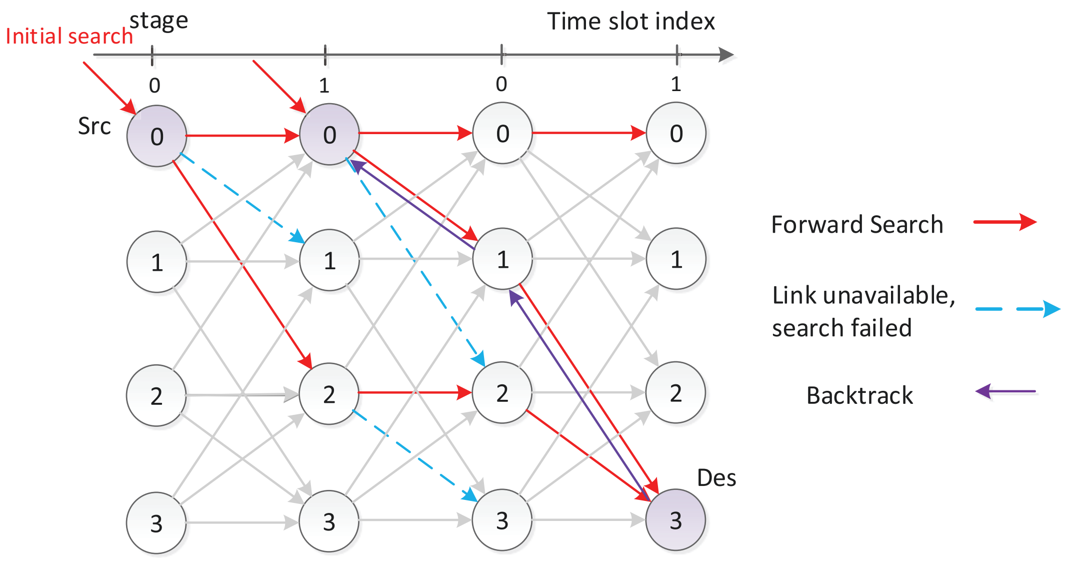


Figure 3.44: Single-layer path search example.

3.6.2 Single-layer Trellis Path Search Implementation

The implementation schematic of single state of single-layer is shown in Fig. 3.45. Each state is implemented as Detect-Select (DS) Unit, which evaluates the incoming search signals to determine whether this state can be activated, and forwards the active search to the next stage. The actual branch state (link allocation state at a specific slot) is stored in the 'Branch State' register. When a branch is allocated, its associated state register is set to '0' to exclude it from later allocation. Correspondingly, state register is reset to '1' to allow allocation when the branch is released.

The work-flow of DS unit for each state is the same, which can be divided into two steps:

1. Detect the active incoming branches: the branch becomes active only when the branch state register as well as ('AND' operation) the corresponding incoming search are both valid. A node can be active as long as any incoming branch is active ('OR' operation).

2. Select one active branch as survivor path: one and only one of the active incoming branches is selected as 'survivor path', saved in the 'Survivor Path Register'. At the same time, the active search is forwarded to the next stage.

3.6.3 Synthesis Results

Using Synopsys Design Compiler, the NoCM was synthesized with TSMC 65 nm technology, for different mesh networks of size 4x4 to 10x10. From the area consumption shown

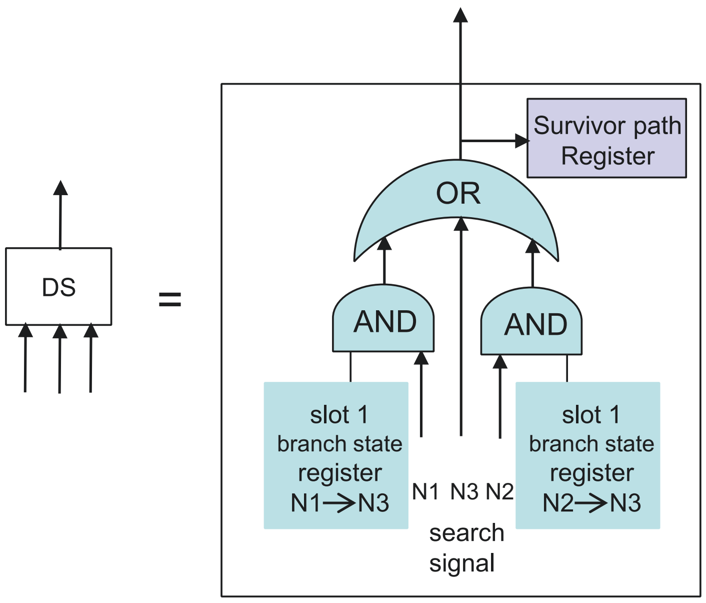

DS
=
Survivor path Register
OR
AND
AND
slot 1 branch state register N1→N3
slot 1 branch state register N2→N3
N1 N3 N2
search signal

Figure 3.45: The implementation schematic of node 3 at stage 1, so the associated time slot is slot 1.

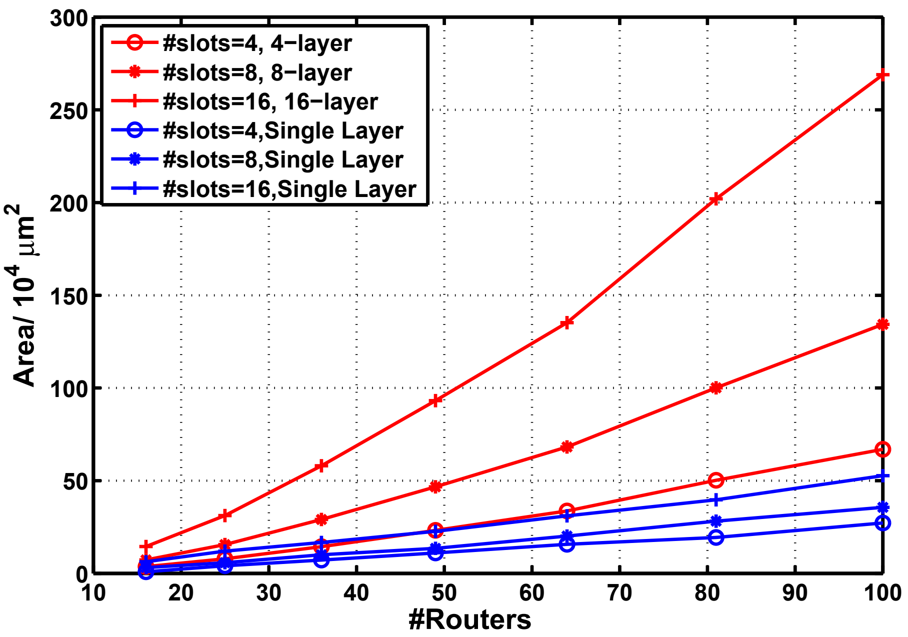


Figure 3.46: Area of unfolded single layer and multi-layer NoCManagers in different size NoCs with different slot table sizes.

in Fig. 3.46, we can see the area of single-layer NoCM grows with *O(N+S)* in 2D-mesh (N= #routers, S= #slots). Because it only implements single layer rather than S layers as in multi-layer approach, its hardware resource consumption is dramatically less than the multi-layer approach. In NxN mesh with slot table size of S, for multi-layer approach, each layer has $\boldsymbol{2N-2}$ stages and there are S layers, so the trellis stage number in total is $\boldsymbol{(2N-2)\cdot S}$; for single-layer, the trellis stage number is $\boldsymbol{S-1+2N-2=S+2N-3}$.

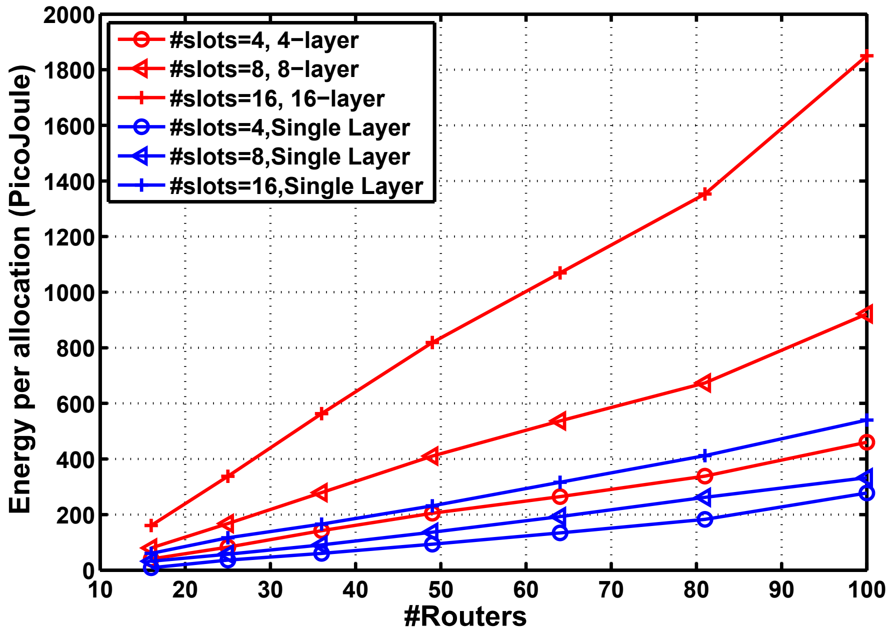


Figure 3.47: Average Energy consumption per allocation of unfolded single layer and multi-layer NoCManagers in different size NoCs with different slot table sizes.

Hence, with large slot table size and large NoC size, the hardware efficiency of single-layer will be remarkably better than multi-layer approach.

As Fig. 3.46 shows, for 10x10 mesh, compared to multi-layer approach, with 4 slots, the single-layer NoCM can reduce the area by a factor of 2.5; with 8 slots, it can reduce the area cost by a factor of 3.8; and with 16 slots, it becomes a factor of 5.1. The plot of average energy consumption per allocation in Fig. 3.47 shows, single-layer consumes much less energy than multi-layer approach. For 10x10 mesh, compared to multi-layer approach, with 4 slots, the single-layer NoCM can reduce the energy consumption by a factor of 1.7, which becomes a factor of 2.8 with 8 slots, and becomes a factor of 3.4 with 16 slots.

3.7 Partitioned Trellis Architecture

The centralized methods for connection allocation in circuit-switched NoCs may pose serious performance and scalability issues in large-scale networks due to the

1. limited path search speed,
2. increasing allocation request rate at central unit,
3. and the increasing communication cost between the central unit and NoC nodes.

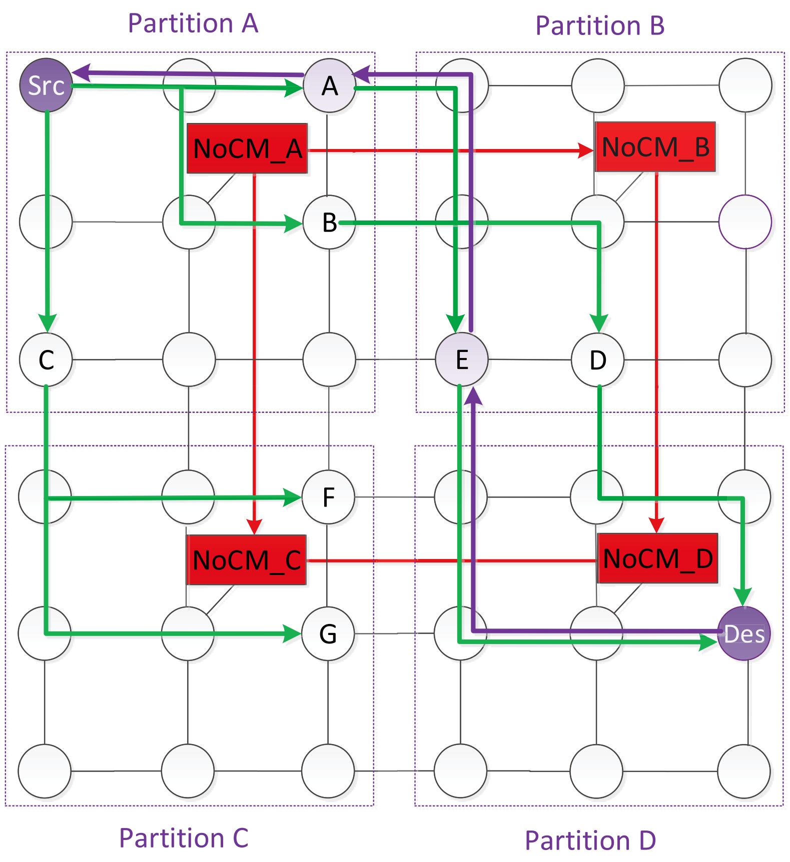


Figure 3.48: The original NoC is divided into 4 partitions with 4 dedicated NoCMs. The green arrow: forward search, purple arrow: backtrack, and the red arrow: communication among NoCMs. The border nodes A and E are backtracked as survivor path. The cross-partition search is along $\boldsymbol{NoCM\_A \rightarrow (NoCM\_B, NoCM\_C) \rightarrow NoCM\_D}$. The path search inside each partition is done as forward-backtrack trellis search, while cross-partition search among NoCMs is as probe search.

We tackle this problem by proposing the partitioned architecture that divides the original system into multiple partitions, and each partition has its own local manager. Each local manager only stores the information of nodes in its region, and is responsible for searching path in its local region. These local managers can work simultaneously, and they need to communicate with each other only when the connection requests cross partitions, i.e. the source node and destination are not in the same partition. Since the managers work simultaneously, the computation capacity is increased. As the NoC nodes only communicate with their local managers, the communication overhead is mitigated. In this section, we employ the folded forward-backtrack multi-layer approach to search path inside partitions.

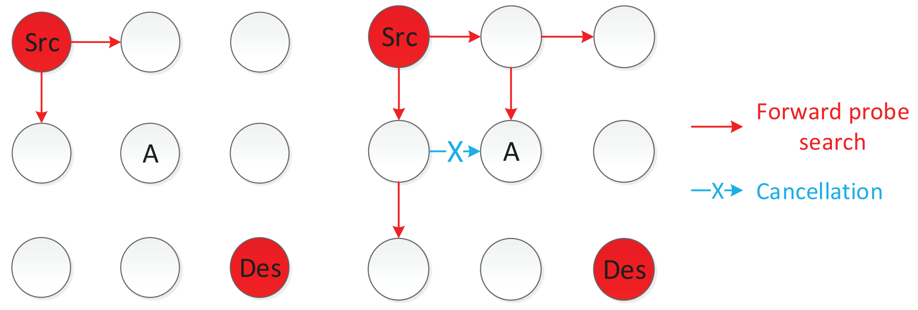
3.7.1 Partitioned TESSA search algorithm

Due to the global knowledge of the system, the previous centralized approaches for connection allocation provide good performance for small and moderate NoCs. However, as the size of the NoC grows, the more allocation requests are received by the central unit per cycle, and the higher is the communication cost between the central unit and NoC nodes. Consequently, the central unit can be a serious bottleneck especially in hotspot traffic. In this section, we proposed a scalable mechanism to address the dynamic connection allocation problem in large systems. The partitioned architecture (i.e. spatial partitioning technique) is used to overcome the scalability problem in traditional centralized systems. NoC is divided into small non-overlapped logical partitions served by local NoCMs. This partitioning technique keeps the request load of the manager and manager-node communication overhead moderate. Inside each partition, the path search problem is solved by a local manager with trellis-search algorithm. To establish a path that crosses partitions, the managers communicate with each other in distributed manner to converge the global path. Hence, good scalability and high performance can be achieved at the same time.

The NoCMs are connected with each other in 2D-mesh topology via dedicated links. The dedicated link width is $\boldsymbol{2 \cdot \log_2 M + \log_2(N \cdot N) + N \cdot S + 2}$ bits, where M is the number of partitions (for destination and source NoCM), $\boldsymbol{N \cdot N}$ is partition size (for the destination node and border nodes) and S is slot table size, and 2 bits for control signals. The link width can be reduced, but then more cycles are needed to send single message. The dedicated links among NoCMs are longer than normal links in the NoC, but authors in [WPG10] claim that it is possible to route these links on high metal layer with reduced RC-delay, and new techniques like high-speed serialized, LVDS on-chip signaling will be available to allow long on-chip link without frequency degradation. These NoCMs can work in parallel, and they need to exchange information only when requested connections cross partitions.

If the source and destination nodes of the request are both at the same partition, this request will be handled by the local manager only. Otherwise, as illustrated in Fig. 3.48, firstly, the local NoCM (NoCM_A) starts the path search in its trellis graph from source node to reach the border nodes (node A,B and C), and then forwards the search message to its neighbor NoCMs (NoCM_B and NoCM_C), continually until reaches the destination (NoCM_D). When destination node is reached, backtrack starts from destination to select the survivor path. We can see multiple nodes on the border can be activated and set as start nodes in the next trellis search, which is due to the feature of trellis search that can support multiple start nodes and multiple destination nodes in one search without additional cost.

The header of the search message among NoCMs contains the address of destination node, source and destination NoCM. For the payload, in forward search it contains the activated border nodes, and in backtrack it contains the selected border node.



(a) In each node a probe may double.

(b) When two probes meet, one is canceled.

Figure 3.49: The probe search among NoCMs. Each node represents a NoCM.

Table 3.2: The usage of control signals

Signals	Usage
00	Idle
01	Search comes in
10	Nack (Path search failed)
11	Ack (Path established)

The basic search idea among NoCMs is similar to parallel probe search [LJL14b], as shown in Fig. 3.49. The source NoCM sends the searches to its all productive (i.e those that lead closer to the destination) neighbor NoCMs. The reached neighbor performs trellis search in its partition, and then forwards the message to its reached productive neighboring NoCMs. Consequently, the search is forwarded to the destination along all possible minimal paths. If two searches meet at one NoCM, then one of them will be canceled based on RoundRobin arbitration. If the downstream NoCM is not available, then that probe search will be canceled immediately without waiting. All channels setup only by the canceled probe are released hop by hop.

3.7.2 NoCM architecture

The block diagram of the NoCM is shown in Fig. 3.50 and comprises trellis graph and control modules (routing module and control logic). The details are explained in the following sections.

3.7.2.1 Control signals

There are 2 bits control wires for 4 control signals for probe search among NoCMs, as listed in table 3.2.

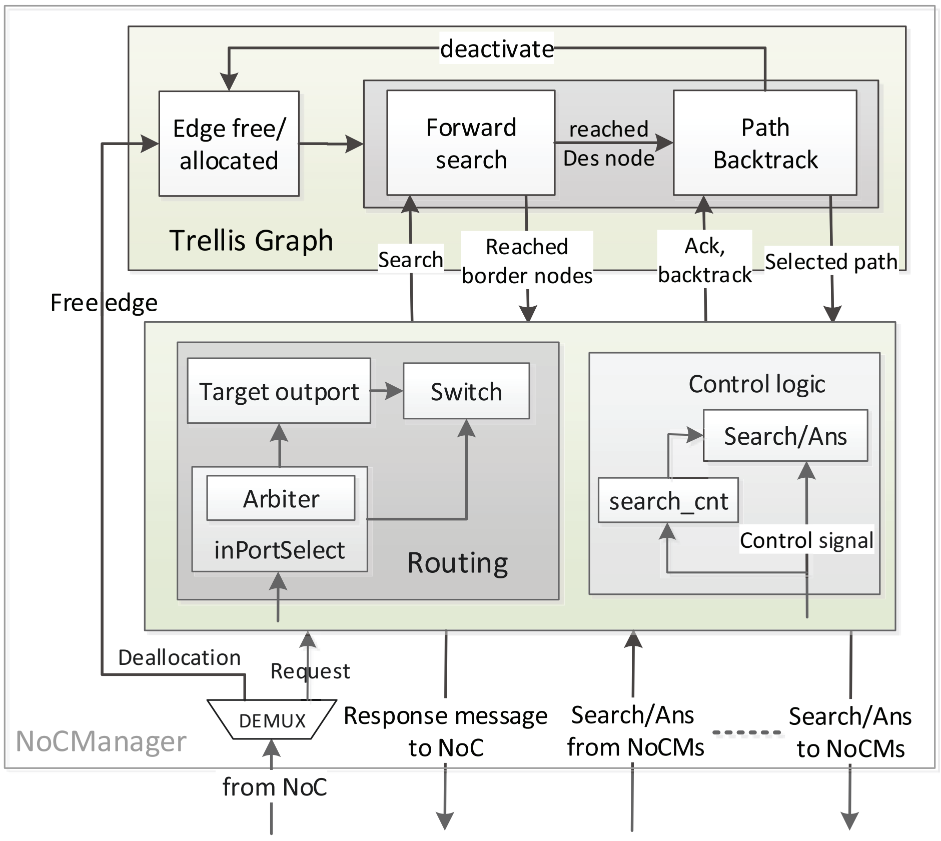


Figure 3.50: Block diagram of the NoCManager

The 'Search comes in' indicates a new probe search comes in, and Nack/Ack are backward answer signals (Ans). The initial 'Search comes in' is generated by source NoCM and the initial Ack is generated by destination NoCM. The detailed control signals usage during probe search is illustrated in Fig. 3.51. For each NoCM, if it accepts a search, it will become busy and reject any later search until becomes free again. Since each search may have two productive output directions and may send out two searches, a counter (search_cnt) is used to record the number of sent out searches. Hence, the value of the counter will be decreased when it receives an Ans signal from the downstream NoCM. The NoCM sends Ack to its upstream NoCM as long as a Ack signal from downstream NoCM is sent back, while sending out Nack only when it receives all Ans signals (search_cnt=0) and they are all Nack. The busy NoCM will become free again in two cases: i) either receives the Ack or ii) receives all Ans signals (search_cnt=0). The search procedure is detailed in Fig. 52.

3.7.2.2 Control trellis path search in each partition

If the NoCM is reached by a probe search, it will start the path search in its trellis graph to search the path inside its region. The forward trellis search will stop in three cases: either reached the border of non-busy productive neighboring NoCMs, or reached the destination node (when the destination node is in this partition), or after certain cycles

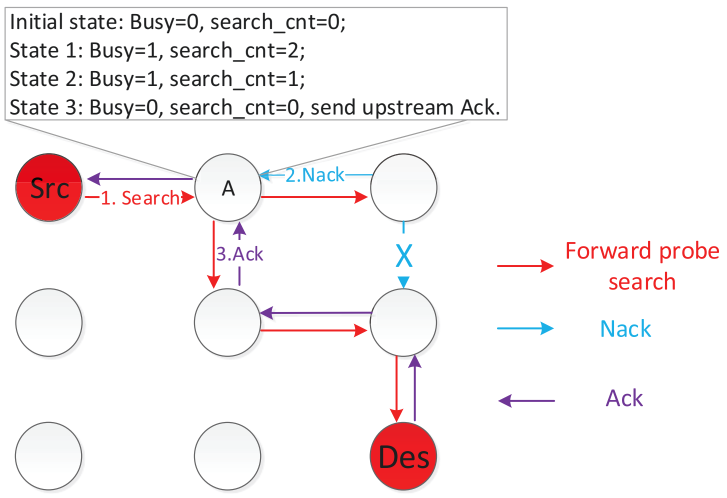

Initial state: Busy=0, search_cnt=0;
State 1: Busy=1, search_cnt=2;
State 2: Busy=1, search_cnt=1;
State 3: Busy=0, search_cnt=0, send upstream Ack.
Src
A
1. Search
2.Nack
3.Ack
X
Des
Forward probe search
Nack
Ack

Figure 3.51: At the beginning, NoCM A is free. At state 1, probe search comes, NoCM A becomes busy. At state 2, Nack comes. At state 3, Ack comes.

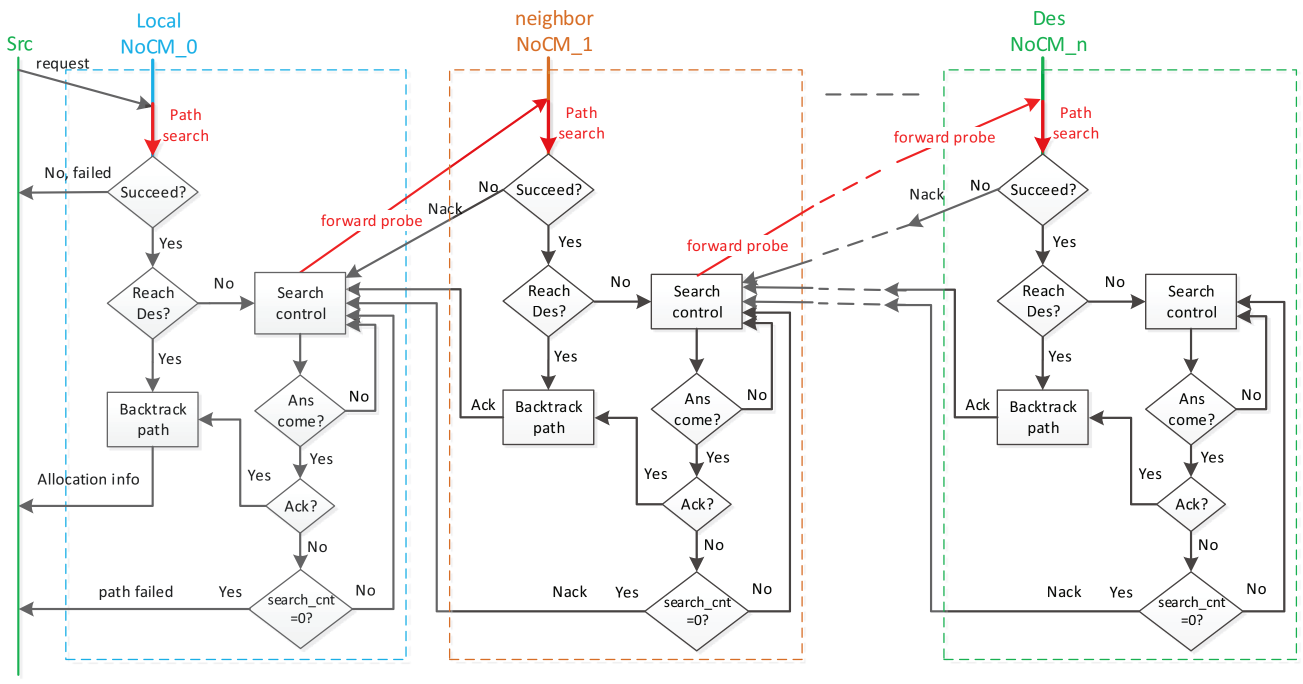


Figure 3.52: The search procedure of partitioned TESSA.

(i.e. $2N - 2$ cycles for NxN mesh partition). The path backtrack in trellis will start in two cases: i) either an Ack is sent back, or ii) the destination node is reached in this trellis graph. In forward probe search, message of the reached border nodes will be sent to the corresponding downstream NoCMs; and in backtrack, only that of the backtracked border node needs to be sent to the upstream NoCM. The connection inside each region is set up by its local NoCM.

3.7.2.3 Ensuring that the destination is activated only once by one request

One request can send out several searches along different paths to reach the destination, and different paths may take different time. In order to avoid allocating several overlapped paths, it should guarantee the destination only be activated by the first search, and will not be activated by the later searches of the same request any more. This is guaranteed by request ID. Each destination NoCM has a table that stores the ID of last request from each source NoCM. The searches belonging to the same request are assigned the same ID. So when the destination NoCM receives a search, the ID of the received search is compared to the ID of last search from that source NoCM. If these two IDs are the same, it means they belong to the same request and the newly received one will be rejected. Otherwise, the new search will be accepted and the ID table will be updated. Hence, for each request, it ensures there is at most one Ack that is sent back.

Since the forward probe search is along minimal paths and does not wait, the livelock and deadlock are avoided.

3.7.3 Performance Evaluation

In this section, we present the synthesis and simulation results of partitioned trellis against the non-partitioned trellis and distributed parallel probe search [LJL14b], and also provide an intuitive idea that how to partition the system.

3.7.3.1 Synthesis Results

The NoCManager is available in synthesizable VerilogHDL and can be generated out of an XML description for different NoC sizes. Using Synopsys Design Compiler, the NoCM was synthesized with TSMC 65 nm technology. Both the non-partitioned and partitioned NoCMs were synthesized with 0.5 GHz clock frequency constraints. For partitioned TESSA, in 18x18 mesh, it is divided into 4 partitions (four 9x9 mesh partitions) or 9 partitions (nine 6x6 mesh partitions); in 16x16 and 20x20 meshes, it is divided into 4 partitions or 16 partitions.

The area consumption is illustrated in Fig. 3.53. It might be surprising that the total area of partitioned TESSA is less than non-partitioned TESSA. The reason is that in non-partitioned TESSA, single trellis graph contains the whole network, while in partitioned TESSA each trellis only contains the local region nodes. Hence, the non-partitioned trellis is much larger than that of partitioned architecture, which induces more effort for wiring and more flip-flops to distinguish different nodes. On the other side, in partitioned TESSA, the more partitions we have, the more control logic is required, so using more partitions may bring the increase of logic area. Therefore, the partitioned TESSA with 4 partitions costs the least area. The 4-partition TESSA can provide up to 21% lower area than

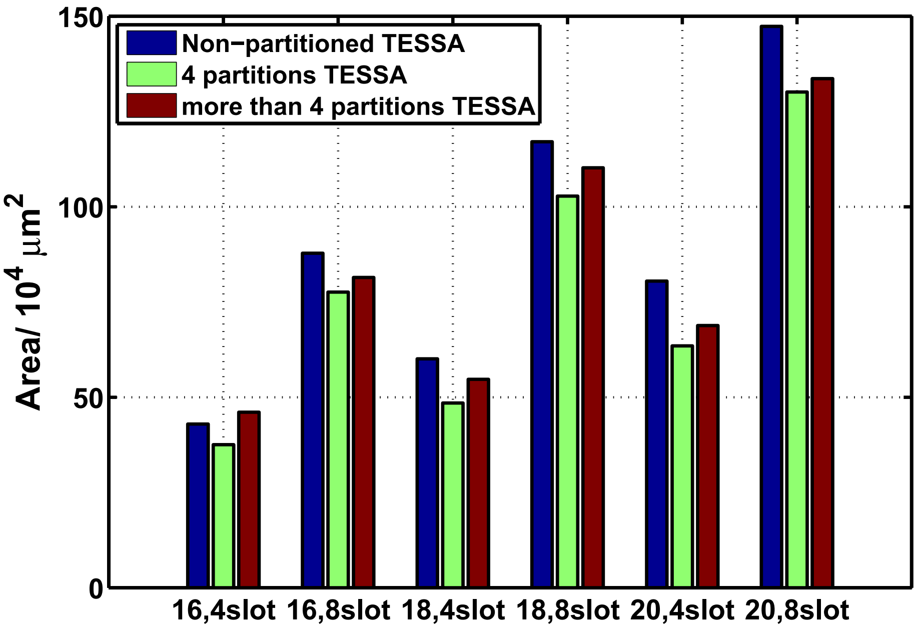


Figure 3.53: Total area of non-partitioned and partitioned NoCMs in different size NoC with different slot table size. The x-axis label '16, 4slot' indicates 16x16 mesh with 4 slots.

non-partitioned (@20x20 mesh with 4 slots). The area of TESSA grows with *O(S· M)* in 2D-mesh (M= #routers, S=slot table size).

3.7.3.2 Simulation Results

The allocation speed and success rate of partitioned NoCMs are compared to the state of the art centralized and distributed allocation techniques under uniform random traffic. We re-implemented a distributed parallel probe connection setup approach according to Liu's work [LJL14b] for comparison, with a retry deadline attached to each request. The request queue sizes of non-partitioned NoCM, single 9-partition NoCM and single 4-partition NoCM are 64, 7 and 16, respectively. The evaluated performance metrics, *success rate*, *total allocation time* and *GS offered load* have been explained in previous section. The results are explained in the following sections.

Any data point shown in the figures comes from simulation of 1 million cycles. The NoC issues a connection request to the NoCM in a uniform random traffic depending on the requirements of the GS offered load. During simulation, half of the nodes are assumed as masters and half of the nodes are assumed as slaves. The connection lifetime, i.e. the number of flits that each connection delivers, is set as 2000 flits, 3000 flits and 4000 flits. The retry deadline of parallel probe search [LJL14b] is set to 3000 cycles.

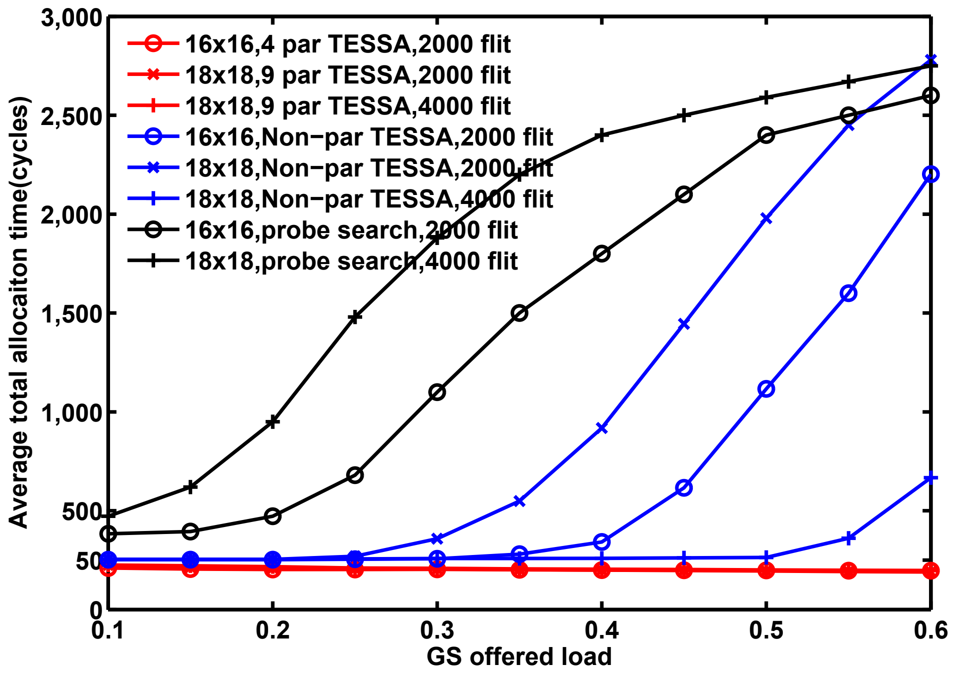


Figure 3.54: Comparison of allocation speed of partitioned and non-partitioned TESSA and probe search in different network with Slot Table Size of 8.

Comparison of allocation speed In parallel probe search, multiple trials might be needed before the eventual success of search due to the investigation of single slot at a time. On the contrary, in TESSA, all slots are being searched simultaneously. As shown in Fig. 3.54, the partitioned TESSA provides much higher allocation speed. Compared to parallel probe search, it offers 7X (@offered load 0.1 in 16x16 mesh) to 71X (@offered load 0.6 in 18x18 mesh) higher allocation speed. Compared to non-partitioned TESSA with offered load from 0.25 to 0.6, it offers 48% (@offered load 0.25 in 16x16 mesh with 2000 flits) to 72X (@offered load 0.6 in 18x18 mesh with 2000 flits) higher speed. We can see in non-partitioned TESSA with low request rate, e.g. lower offered load (from 0.1 to 0.2) or longer connection lifetime (4000 flits), the allocation time is reduced significantly. The reason for this is in high request rate the non-partitioned NoCM is too busy, so the incoming requests have to wait longer in the queue to get their turn. However, in partitioned TESSA, since there are multiple NoCMs working in parallel, the requests are usually processed in time without waiting, thereby the allocation speed is much higher. In partitioned TESSA, with different connection lifetime (2000 flits or 4000 flits in 18x18mesh), the allocation time is similar.

Comparison of Success Rate The simulation results of success rate is illustrated in Fig. 3.55 and Fig. 3.56. For partitioned TESSA, the systems that are divided into 4 partitions achieve the best success rate in comparison to those with 9 or 16 partitions. When the GS offered load is high, the partitioned TESSA provides much higher success

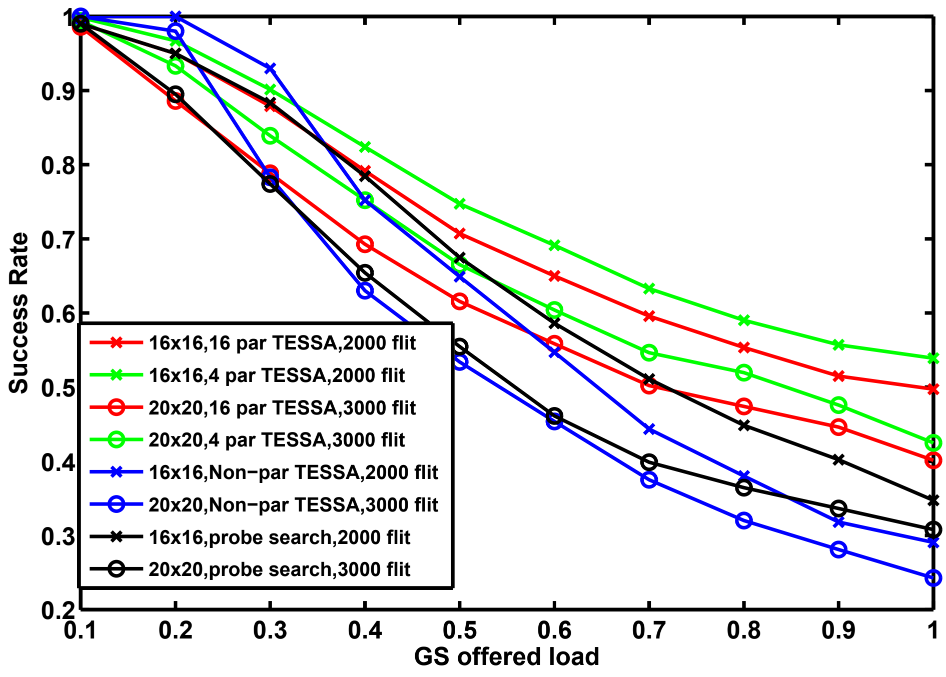


Figure 3.55: Comparison of Success Rate of partitioned and non-partitioned TESSA and probe search in 16x16 and 20x20 networks with Slot Table Size of 8.

rate than non-partitioned TESSA and parallel probe. In parallel probe search, since single slot is investigated at a time and the retry before deadline policy is employed, the search can be stopped as failure when the deadline is reached, though the remaining slots may provide an available path. Moreover, it only searches the minimal path, cannot detour as in TESSA. Therefore, compared to parallel probe search, the 4 partitions TESSA can offer up to 55% higher success rate in 16x16 mesh and up to 33% higher in 18x18 mesh. In non-partitioned TESSA with high offered load, many incoming requests must be rejected as failure immediately because of the full request queue even though there might be a free path for these requests. Hence, compared to non-partitioned TESSA, the 4-partition TESSA can provide up to 85% higher success rate in 16x16 mesh, up to 84% higher success rate in 18x18 mesh, and up to 75% higher in 20x20 mesh. However, when the offered load is very low, e.g. lower than 0.3 in 16x16 and 18x18 mesh, the single NoCM can handle all the requests in time. In partitioned TESSA, since the search among NoCMs is along minimal path, its path diversity is less than the non-partitioned TESSA, thereby now the success rate of non-partitioned TESSA is the best.

3.7.3.3 Suggestion on how to partition the system

From the simulation results we can see, the partitioned system does not necessarily always provide better performance than non-partitioned system. If the request injection rate is not heavy, the performance of partitioned system may even be worse than non-partitioned

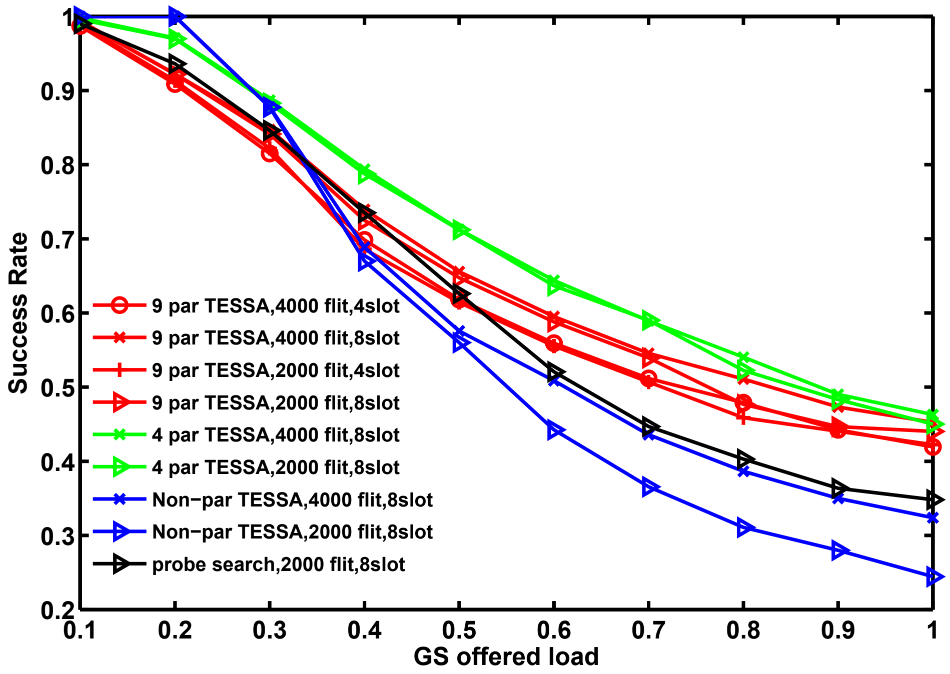


Figure 3.56: Comparison of Success Rate of partitioned and non-partitioned TESSA and probe search in 18x18 network with Slot Table Size of 4 or 8.

system. The request injection rate indicates the number of requests generated by each master per cycle. Then how to partition the system? In this section, we provide an intuitive suggestion to this question.

The reason to partition the system is given in a large NoC with heavy request injection rate, when there are too many requests to the NoCM in a short time that the NoCM cannot process the requests in time, and thus some requests may even need to be discarded. Therefore, when the injection rate of connection request exceeds the NoCM's processing capacity, the system should be partitioned. We take folded FB TESSA as an example. Assume half of the nodes in the NoC are masters and half of the nodes are slaves. For $\boldsymbol{N \cdot N}$ mesh network, since the average path length is $\boldsymbol{N-1}$ hops, so the average processing time for single request is $\boldsymbol{2 \cdot (N-1)}$ cycles. Hence, the corresponding boundary of request injection rate that is under the NoCM's processing capacity is $\boldsymbol{1 \div ((2 \cdot (N-1)) \cdot (N \cdot N \cdot \frac{1}{2})) = \frac{1}{N^2(N-1)}}$ request per cycle per master. If the request injection rate is higher than this boundary, the manager cannot process the incoming requests in time, and thus it is better to partition the system. The boundary of request injection rate that ensures the NoCM can process the requests in time in different NoC sizes is shown in Fig. 3.57. As long as the average request injection rate is above this curve, we are going to partition the system, and the partition size is chosen as to make the local NoCM in each partition can process the incoming requests in time.

We should note this is just a preliminary idea, and we do not consider the communication cost among NoCMs. In reality, if the system is partitioned into too many partitions, the

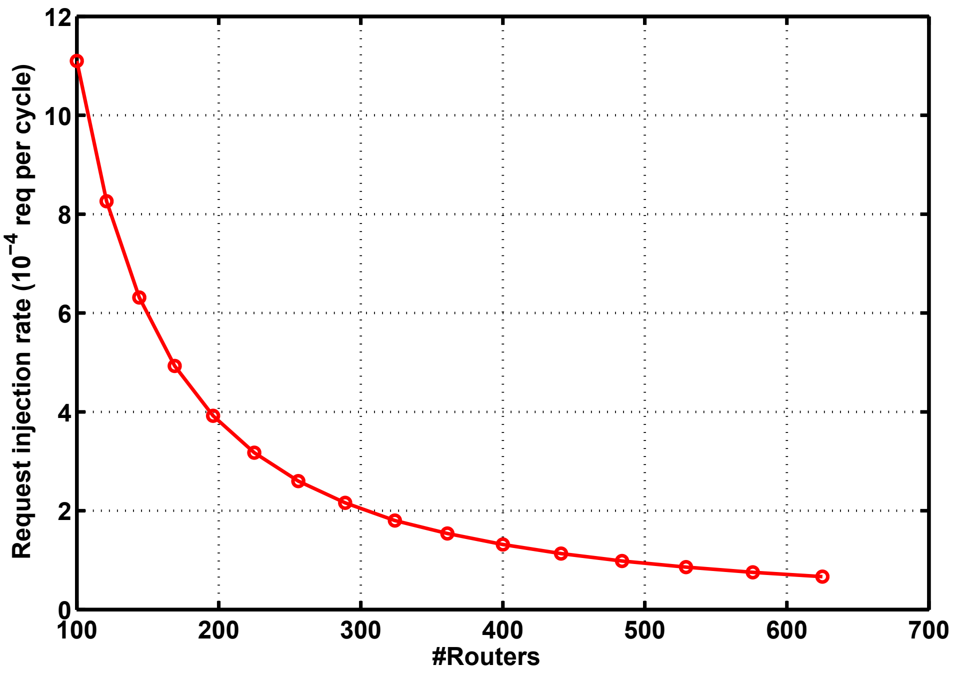


Figure 3.57: The boundary of request injection rate under the NoCM's capacity in different NoC sizes.

communication cost among NoCMs will become dominant, and thus the system performance may decrease. Hence, as long as the local managers already can handle the requests in time, more partitions does not necessarily provide better performance any more.

3.8 Summary

This chapter introduces a dedicated connection allocator, NoCM, which employs the trellis path search algorithm for the connection allocation of TDM CS. The results are summarized below,

- We proposed the TESSA algorithm, which can explore all possible paths between source-destination node pairs within a guaranteed latency.
- We proposed the FB TESSA, which comprises two steps: forward search and path backtrack. Three different TESSA structures, unfolded structure, folded structure and bidirectional structure are presented to be chosen for different scenarios.
- In order to save the path search time, the RE TESSA is proposed, which merges the forward search and path backtrack into single step.
- The single-layer TESSA is proposed that only needs to implement single layer of trellis. Hence, compared to previous multi-layer approach, the resource consumption is reduced dramatically.

- In order to address the scalability problem of centralized system, the partitioned structure is proposed, which divides the system into multiple partitions with multiple local NoCMs. Since each NoCM only manages and communicates with its local NoC nodes, the request load and communication overhead is reduced. As the NoCMs can work simultaneously, the computation capacity is enhanced.

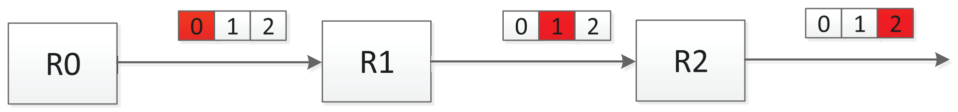
Chapter 4

Centralized Connection Allocation for Combined TDM-SDM CS NoCs

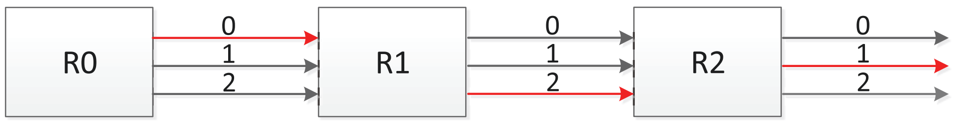
In this chapter, we present the trellis path search for the connection allocation of combined TDM and SDM CS[CMF17c]. TDM can share the resource by splitting the link bandwidth into time slots. But there is a constraint on the time slot scheduling that after each hop the reserved slot along the path should be increased by 1, which can limit the probability of successful connection allocation. To mitigate this, SDM is proposed, which physically splits the link wires into sub-channels, and any free sub-channel at the next hop along the path can be reserved. However, the area cost of SDM switch scales quadratically with the number of sub-channels, which limits the scalability. Moreover, the number of sub-channels the link can be split into is limited by the bits of the link wires. In order to address the problem of i) path diversity and ii) scalability, the combined TDM and SDM CS was proposed, in which the sub-channel is further split into time slots, and thus can increase the path diversity as well as share a sub-channel among multiple connections to improve the resource utilization. Hence, the poor resource usage inherent to CS is mitigated. In this chapter, we propose a dedicated connection allocator for combined TDM-SDM CS NoCs based on trellis-search algorithm, which can explore all possible paths between source-destination node pairs within a guaranteed latency. All the trellis structures presented in chapter 3 can be applied to combined TDM and SDM CS. Finally, we studied the influence of different TDM-SDM link partitioning strategies on success rate and path length that allowed us to find the optimal solution.

4.1 Introduction of SDM CS

Due to the advanced semiconductor technology, wires have become an abundant resource in NoC. However, using all the wires between two routers as single wide communication channel becomes inefficient and inadequate in many situations. Hence, we can adopt



(a) In this TDM NoC, each link is split into 3 time slots. Along the path, if slot 0 is reserved in R0, slot 1 must be reserved at the next hop.



(b) In this SDM NoC, each link is split into 3 sub-channels. Along the path, if sub-channel 0 is reserved in R0, any free sub-channel can be reserved at the next hop.

Figure 4.1: Connection allocation in TDM CS and SDM CS.

the SDM technology to physically split the wires into several sub-channels to offer more flexibility[LJL15, EJ13, LMV+08, MSAA09, YKH10, LJL14a]. In TDM CS, there is tight scheduling constraint on time slots reservation that, if a slot t is reserved in a router and then slot $(t + 1) \bmod S$ must be reserved at next hop along the path, where S is the slot table size, which limits the probability of path establishment in the network. However, unlike TDM, in SDM, no matter which sub-channel is reserved in a router, any free sub-channel at the next hop can be reserved along the path, which can provide more path diversity, as depicted in Fig. 4.1.

4.1.1 Combined TDM and SDM CS

In SDM CS, as the number of sub-channels increase, the cost of the crossbar switch of a router increases quadratically. If n is the number of sub-channels of a link, the cost of the crossbar of a router scales with $O(n^2)$. The quadratic complexity can be reduced by using a multiple-stage switch, but then the switch delay will be increased. Moreover, the number of sub-channels is limited by the number of wires, so it cannot be increased arbitrarily. As NoCs scale up in size, or as traffic flows increase, the number of required sub-channels increases, which may induce unacceptable router complexity or be just impossible because of insufficient wires.

In TDM CS, if we have more flows and need a finer granularity for bandwidth allocation, a larger slot table is required. Though increasing the slot table size does not increase the area much, however, there is a direct relation between table size and maximum delay. If the link is split into n slots, a packet has to wait n cycles for its next slot to appear [GEEK11]. Moreover, the tight scheduling constraint on time slots reservation restricts the path diversity.

Hence, the combined TDM and SDM CS is proposed. Combining TDM and SDM can offer more flexibility since optimization can be made either in time or in space. As depicted

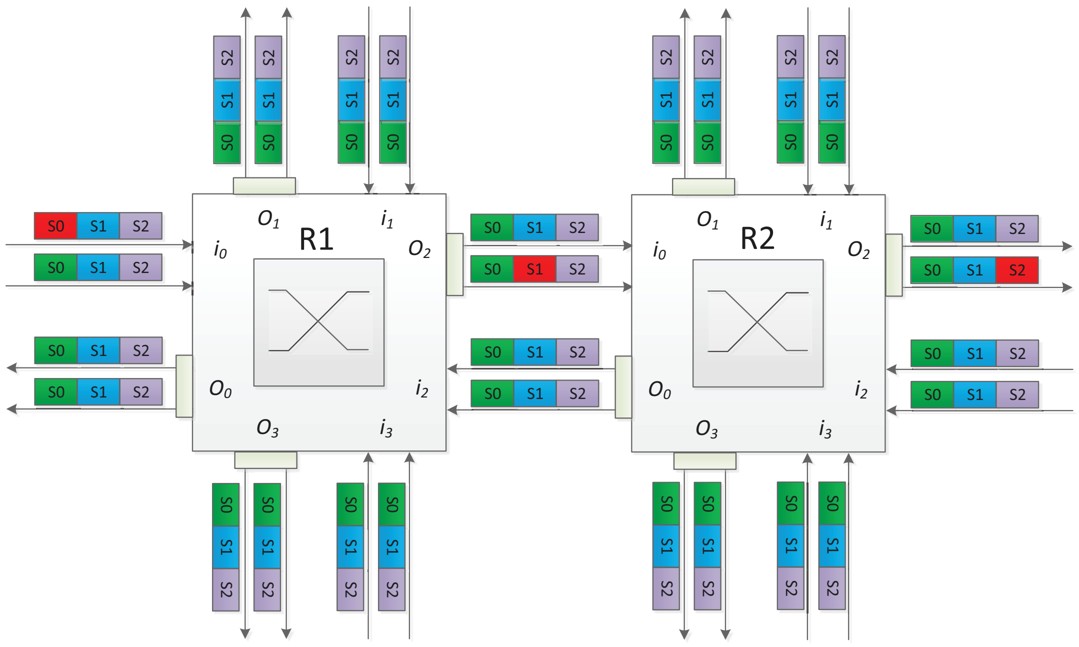


Figure 4.2: Combined TDM-SDM routers with each link split into 2 sub-channels and 3 time slots. Along the path, if time slot 1 is reserved at a sub-channel in router R1, slot 2 at any of the two sub-channels can be reserved in R2.

in Fig. 4.2, each sub-channel is split into multiple time slots so that it can be shared by multiple connections, and thus the resource utilization of sub-channel is improved. Consequently, the number of sub-channels and time slots can be kept moderate and thus reducing the router complexity. Actually, the TDM CS and SDM CS are special cases of combined TDM-SDM CS. In combined TDM-SDM CS, when the link is split into only 1 sub-channel, it will be TDM CS; and if the link is split into only 1 time slot, it will be SDM CS.

4.1.2 Connection allocation in combined TDM and SDM CS

The connection allocation in combined TDM and SDM CS has to i) allocate a contention-free path from source node to destination node and ii) allocate the time slots and sub-channels along the path. As depicted in Fig. 4.2, the path is allocated as $\boldsymbol{R1} \rightarrow \boldsymbol{R2}$, and the corresponding slot sequence and sub-channels along the path are $\{(\boldsymbol{slot}\ 0\ \boldsymbol{subchannel}\ 0), (\boldsymbol{slot}\ 1\ \boldsymbol{subchannel}\ 1), (\boldsymbol{slot}\ 2\ \boldsymbol{subchannel}\ 1)\}$.

4.2 System Model

The system model of dedicated allocator (i.e. NoCManager) based NoC architecture is illustrated in Fig. 4.3, which is similar to the system model presented in chapter III.

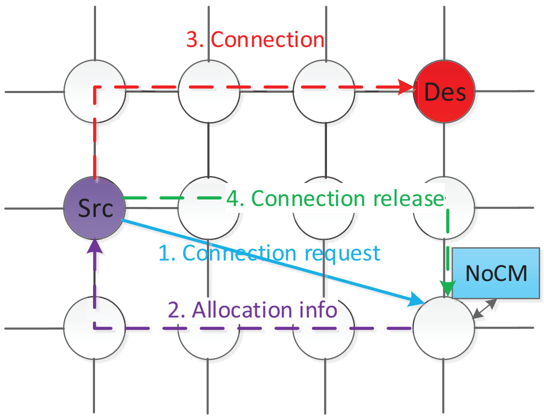


Figure 4.3: System Model of the NoCManager based NoC platform. Src: source node, Des: destination node.

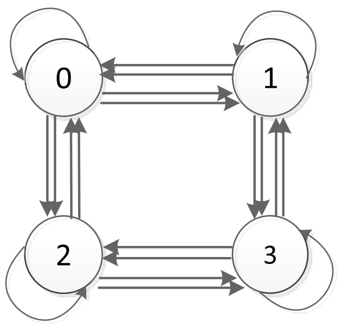


Figure 4.4: Example NoC graph. Each link has two sub-channels. A node can reach itself (curve arrow).

NoCManager (NoCM) attempts to allocate the appropriate connections when it receives connection requests.

When a node needs a connection, it sends the connection reservation request to the NoCM. The NoCMs tries to allocate the connection when it receives the request. When the allocation succeeds, NoCM sends the resulting allocation information to the corresponding source node. Then the source node starts to transmit data along the allocated connection based on the allocation information. When the data transfer is finished, the source node deletes the established connection, and also informs the NoCM to free the corresponding allocated sub-channels and time slots.

4.3 Connection Allocator Architecture

In this section, we present a dedicated hardware connection allocator (NoCM) to allocate connection for TDM-SDM NoCs, which employes the trellis path search algorithm that can guarantee the setup latency and explore all possible paths between two given nodes.

4.3.1 Formalizing The Trellis Graph Structure

The aforementioned path search problem can be efficiently solved by trellis search approach (TESSA). The example NoC in Fig. 4.4 can be represented by the trellis graph in Fig. 4.5. There are three important characteristics of the trellis graph, which are:

1. **Stages**: The stage (i.e. the column) of the trellis graph is a recursion of the network, as in Fig. 4.5. The trellis graph can be constructed as multi-stage (by default, $\boldsymbol{2N-2}$ stages for NxN mesh NoC, which equals the longest minimal path in the network) to represent the multi-hop traversals through the network.
2. **States**: The node in the trellis graph is called state, which represents a router in the NoC.
3. **State transitions**: Each state transition corresponds to a forward step in the trellis, which is represented by a directed edge (or branch) connecting the two states. A branch represents a link at a specific time slot.

In this section, we adopt the multi-layer approach that every slot at the initial stage has its own layer of trellis graph, so each slot from the source node can search its own path in parallel.

4.3.2 Trellis Path Search Algorithm

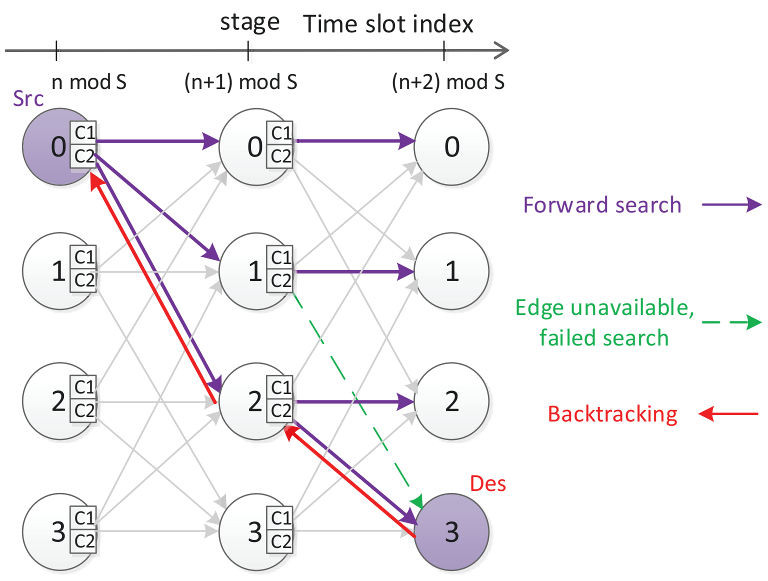
In this section, the forward-backtrack trellis is adopted, which completes the path search in two steps:

- **Forward search** i.e. traverse the NoC to search the best free path out of all possible survivor paths.
- **Backtracking** i.e. when the forward search succeeds, the backtrack starts to collect the survivor path, i.e. the contention-free path which is to be found.

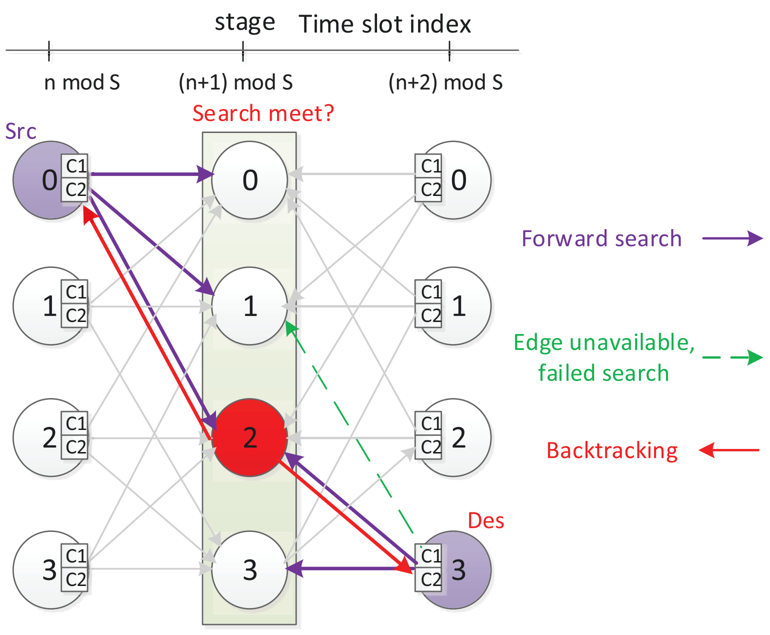
All the trellis search approaches presented in chapter 3 can be applied to combined TDM-SDM CS, and we take unidirectional FB trellis search and bidirectional FB trellis search as example in this section.

4.3.2.1 Unidirectional Trellis Path Search

The unidirectional trellis graph of example network in Fig. 4.4 is illustrated in Fig. 4.5a. The source node at the initial stage sends out the search signal to try to activate its connected neighbors, and the activated nodes continue to forward the search until reaching



(a) Unidirectional trellis search graph



(b) Bidirectional trellis search graph

Figure 4.5: The example NoC can be represented by trellis. Assume each link has 2 sub-channels (C1 and C2). The green dotted arrow indicates the search of this edge failed because it is not available at the moment (already occupied). Src: source node, Des: destination node.

the last stage. During the search, if any sub-channel of a branch is free, the search signal can propagate along this branch. At the last stage, if the destination node is active, the backtrack starts to collect the survivor path and the associated sub-channels and time slots.

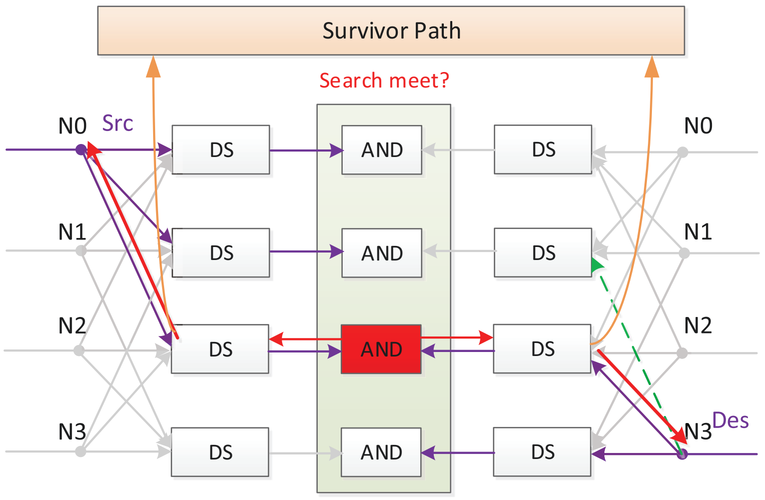


Figure 4.6: Implementation schematic of the bidirectional trellis graph

4.3.2.2 Bidirectional Trellis Path Search

In order to reduce the search time, we proposed the bidirectional search, which launches the search at two sides, source node and destination node simultaneously. At the middle stage of the trellis, it checks whether the searches from two sides meet at a node. If this is true, the backtrack starts from the middle stage to select the survivor path. Otherwise, it fails. Hence, the search time is reduced to half. The bidirectional trellis graph of example network in Fig. 4.4 is illustrated in Fig. 4.5b. The search begins at the source (at the initial stage) and destination (at the last stage) simultaneously, traversing through the trellis to try to activate the connected neighbors. Source and destination both activate node 2 at middle stage. Hence, the search succeeds. The backtrack starts from node 2 at middle stage to source and destination simultaneously. The survivor path is selected as $\boldsymbol{N0} \rightarrow \boldsymbol{N2} \rightarrow \boldsymbol{N3}$.

4.3.3 Trellis Path Search Implementation

The implementation schematic of a bidirectional trellis graph is shown in Fig. 4.6. Each state is implemented as Detect-Select (DS) Unit, as detailed in Fig. 4.7, which evaluates the incoming search signals to determine whether this state can be activated. If yes, it forwards the active search to next stage. The actual branch state (link allocation state at a specific slot) is stored in the 'Branch State' register, with each element representing a specific sub-channel. When a sub-channel is allocated at a specific slot, its associated state register is set to '0' to exclude it from later allocation. Correspondingly, state register is reset to '1' to allow allocation when the sub-channel is released.

The work-flow of DS unit for each slot is the same, which can be divided into two steps:

1. Detect the active incoming branches: the state registers of all sub-channels of a

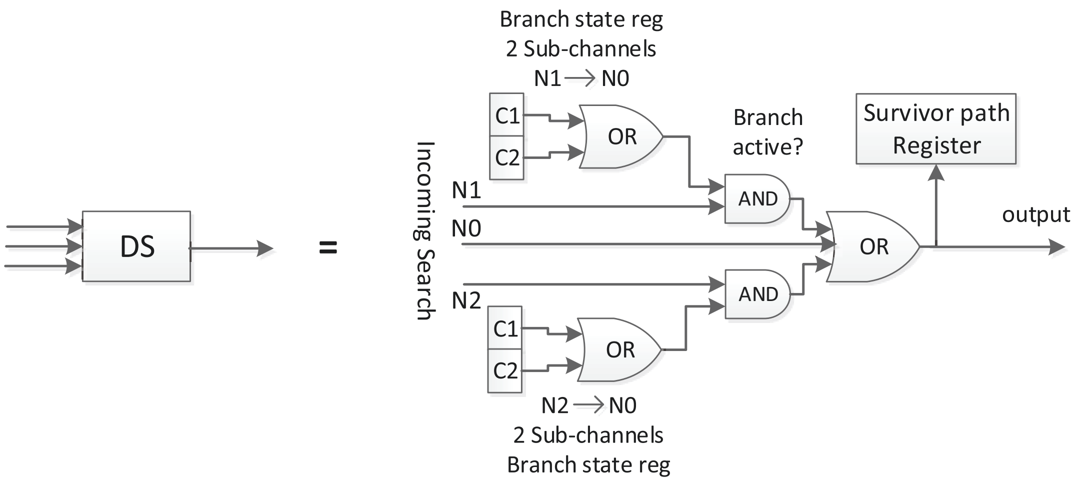


Figure 4.7: Implementation details of a DS unit with 2 sub-channels for single slot of node 0

branch are connected to an OR gate, so that as long as any sub-channel is valid (free), this branch is valid. The branch becomes active only when the branch as well as ('AND' operation) the corresponding incoming search are both valid. A node can be active as long as any incoming branch is active ('OR' operation).

2. Select one active branch as survivor path: one and only one of the active incoming branches is selected as 'survivor path', saved in the 'survivor path Register'. At the same time, the active search is forwarded to next stage. Note, if a node is already active, in the future, it will select itself as the predecessor.

The path search is completed in two cycles. First, the search is started at source (at initial stage) and destination (at last stage) simultaneously, propagated until reaching the middle stage. At the middle stage, the corresponding search signals from two sides are connected to an AND gate to check whether searches meet. At the next cycle, if two searches meet, backtrack starts from the middle stage by reading out the stored predecessor hop by hop. Each selected predecessor and one of its valid associated sub-channels at each stage are saved into the 'Survivor Path' register, where eventually the complete path from source to target node can be found.

4.4 Performance Evaluation

The synthesis and simulation results are presented in this section.

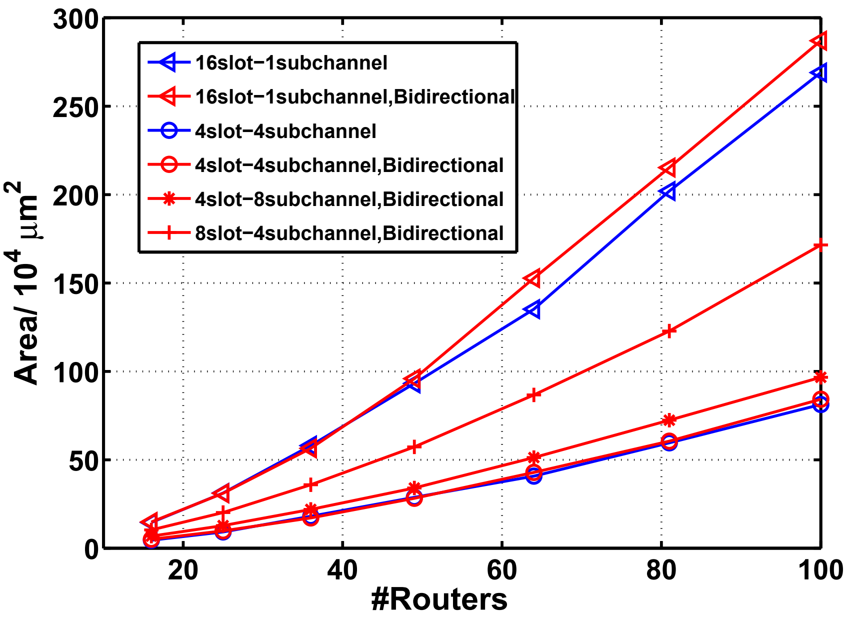


Figure 4.8: Area of (unidirectional) TESSA and Bidirectional TESSA NoC-Managers with different link partitioning for different NoC sizes.

4.4.1 Synthesis Results

The NoCManager is available in synthesizable VerilogHDL and can be generated out of an XML description for different NoC sizes. Using Synopsys Design Compiler, the NoCM was synthesized with TSMC 65 nm technology, for different mesh networks of size 4x4 to 10x10. For both (unidirectional) TESSA and Bidirectional TESSA, the critical path constraints were gradually increased according to the increased NoC size. For instance, in Bidirectional TESSA, the critical path is constrained to 1.11 ns for 6x6 mesh which is then increased to 2 ns for 10x10 mesh. The link is split into different partitions (different time slots and different sub-channels), e.g. 16slot-1subchannel, 4slot-4subchannel, 4slot-8subchannel, etc.

As shown in Fig. 4.8, the area of TESSA and bidirectional TESSA NoCManagers grows with $O(S\cdot\ M\cdot\ C\cdot\ \sqrt{M})$ in 2D-mesh (M= #routers, S= #slots, C= #sub-channels), where $\sqrt{M}$ is related to the number of trellis stages $(2\cdot\ (\sqrt{M}\ -1))$. The area grows rapidly with slot table size, while quite slowly with the number of sub-channels. The area of bidirectional TESSA is almost the same as the (unidirectional) TESSA. As shown in Fig. 4.9, the average AT complexity of bidirectional TESSA is almost 2X better than the (unidirectional) TESSA. As shown in Fig. 4.10, the average energy consumption per allocation grows with $O(S\cdot\ M\cdot\ \sqrt{M})$. The influence of the number of sub-channels on energy is almost negligible. In conclusion, the bidirectional TESSA can halve the critical path so that it can halve the path search time compared to (unidirectional) trellis search algorithm, while consuming almost the same area and energy.

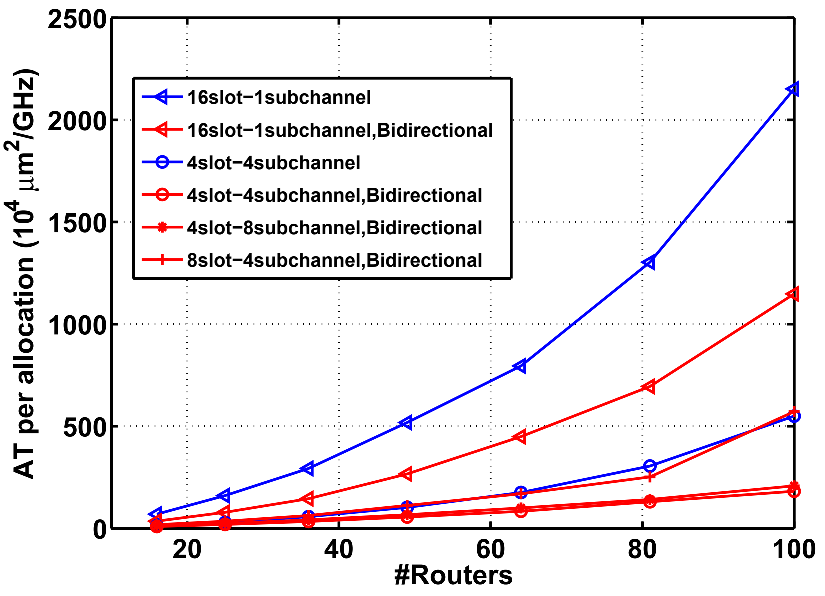


Figure 4.9: Average AT complexity per allocation of two different NoCManagers with different link partitioning for different NoC sizes.

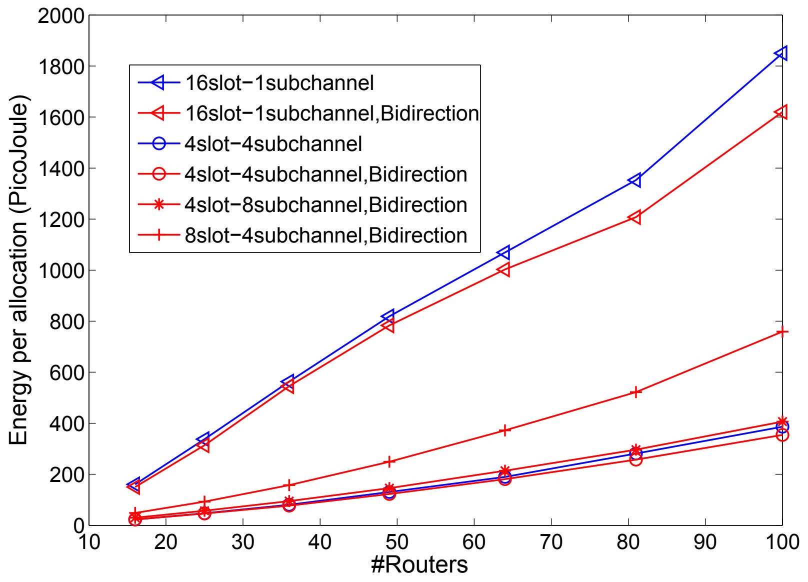


Figure 4.10: Average energy consumption per allocation of two different NoCManagers with different link partitioning for different NoC sizes.

4.4.2 Simulation Results

We evaluate the influence of different link partitioning strategies on success rate and average flit delivery latency. These results are explained in this section.

For evaluation several performance metrics are used:

- *success rate* denotes the ratio of successful requests that established paths to the total requests.
- *GS offered load* refers to the ratio of the GS traffic each master offered to its maximum capacity. Suppose the link width is 256 bits, a new request generated by each master after every 2000 cycles, and each connection can deliver 128,000 bits data after setup, then the offered load is $\mathbf{128000 \div 256 \div 2000 = 0.25}$.
- *background traffic load* refers to the ratio of links capacities (time slots and subchannels) of each router which are already randomly occupied to be excluded from path search to the total links capacities of each router.

In this section, the simulations are done with the same 256 bits wires per link in 6x6 mesh network.

4.4.2.1 Influence of different link partitioning on success rate

The NoC issues a connection request to the NoCM in a uniform random traffic meeting the requirements of the GS offered load. During simulation, half of the nodes are assumed as master and half nodes are assumed as slave. The source-destination pairs are uniform randomly selected that each source-destination pair has equal probability to be chosen. Any data point that is shown in the figures comes from simulation of 1 million cycles. The link width is set as 256 bits, so each subchannel is 128 bits and 16 bits in 2-subchannel and 16-subchannel partitioning, respectively.[7] After a connection is established, 128,000 bits of data are delivered before the connection is released.

When the link is split into fixed number of partitions ($\boldsymbol{\#time\ slots \cdot \#subchannels}$), splitting the link into greater number of subchannels can provide more path diversity, which can contribute to higher success rate. As depicted in Fig. 4.11, the 2slot-8subchannel partitioning can offer up to 4% higher success rate than 16slot-1subchannel partitioning (@ GS offered load 0.1). However, after certain number of sub-channels, more sub-channels do not necessarily contribute to higher success rate. The success rate of 1slot-16subchannel is even a little lower than 2slot-8subchannel. The reason is that the network reaches saturation point. It cannot allocate more successful connections simply because it has already reached the limit of NoC capacity. When the link is split into 16 partitions, it can only support at most 16 concurrent connections no matter how it is split. Since 1slot-16subchannel partitioning offers more path diversity, it can successfully allocate some long paths, which occupy a lot of link capacities, and thus the network reaches the saturation early. Alternatively, if we increase the partitions of the link, it can provide more path

[7] We assume only payload data is delivered without header overhead, because the packet header could be omitted in CS.

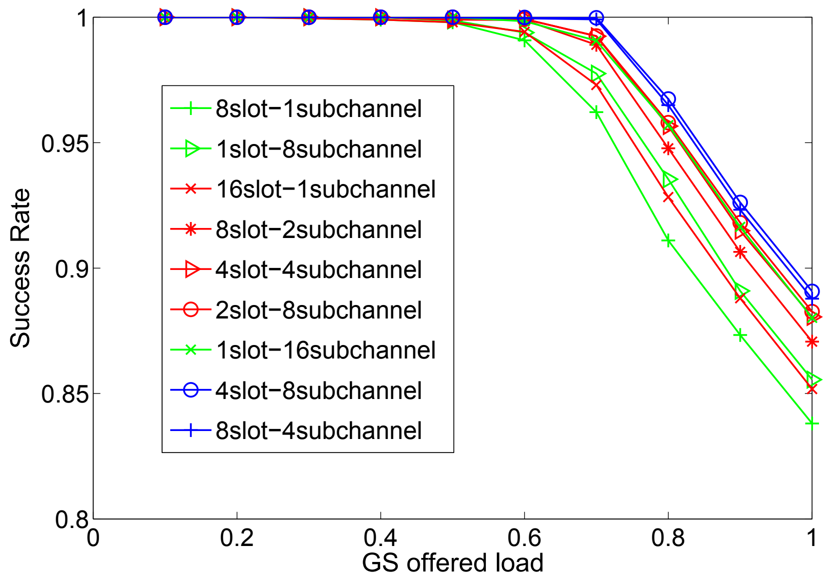


Figure 4.11: Influence of different link partitioning on success rate in 6x6 mesh.

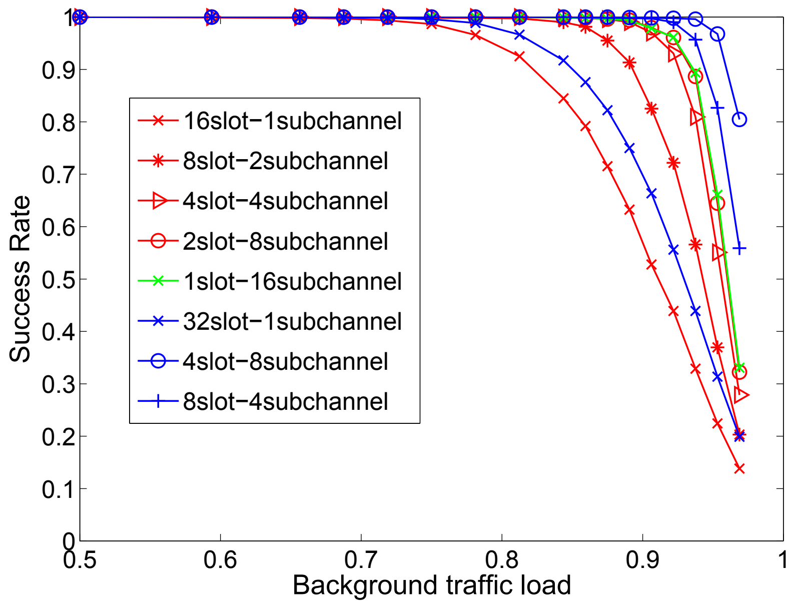


Figure 4.12: Influence of different link partitioning on success rate under background traffic.

diversity as well as more NoC capacity. Compared to 1slot-16subchannel partitioning, both 4slot-8subchannel and 8slot-4subchannel can offer higher success rate. Since more sub-channels leads to higher router complexity, it could be a better choice to split the sub-channel into time slots instead of increasing the number of sub-channels (e.g. 2slot-8subchannel against 1slot-16subchannel), which can provide good success rate as well as relatively low router complexity.

4.4.2.2 Evaluation of different link partitioning under certain background traffic

In this section, we run the simulation under pre-defined background traffic to show that increasing the sub-channels can increase the path diversity significantly. Moreover, the number of trellis stage is set as 16 (by default is 10 stages for 6x6 mesh) to allow the exploration of non-minimal paths. We first generate an amount of random background traffic that uses some of the link capacities. Then we measure the path length and success rate of path allocations between all pairs of two nodes. We produce 300 samples at each background traffic load.

Influence of different link partitioning on success rate From the simulation results of success rate in Fig. 4.12, we can see the 1slot-16subchannel partitioning can offer up to 3X higher success rate than 16slot-1subchannel partitioning (@ background traffic load 0.95). But the success rate of 2slot-8subchannel is almost the same as 1slot-16subchannel. Alternatively, we can split each sub-channel into more time slots (increase the link partitions) to achieve higher success rate. The 4slot-8subchannel can provide up to 2.4X higher success rate against 2slot-8subchannel and 1slot-16subchannel.

Influence of different link partitioning on delivery latency and path length Some messages, e.g. control message, whose message size might be small, but have strict requirement for the delivery latency. So in this section, we present the comparison results of delivery latency, i.e. latency for delivering single flit. For each flit, the delivery latency comprises two parts for TDM-SDM CS: 1) waiting time in TDM scheme to get its time slot to be delivered, which equals the slot table size; and 2) the routing time to traverse the NoC, which equals the path length. For example, assume a connection in 8slot- 2subchannel CS has path length of 7 hops, then the delivery latency per flit in total is 15 cycles ($= \mathbf{8} + \mathbf{7}$).

The average delivery latency per flit and the average path length per connection is shown in Fig. 4.13 and Fig. 4.14. As we can see, when the background traffic load is heavy (more than 0.94), the higher the background traffic load is, the shorter the path is. However, when the background traffic is low (lower than 0.9), the lower the background traffic is, the shorter the path is. The reason is, when the background traffic load is very high, the source node can only reach the nearby nodes. By reducing the background traffic, the source node can reach more and more nodes, and thereby the path becomes longer. However, after certain point, the source node can reach all nodes in the network but may have to do a lot of detours. So if we keep reducing the background traffic, the number of detours becomes smaller, and thereby the path becomes shorter.

When the link is split into more sub-channels, or the sub-channel is split into more slots, the path diversity is enhanced, which may reduce detours and thereby shorten path length.

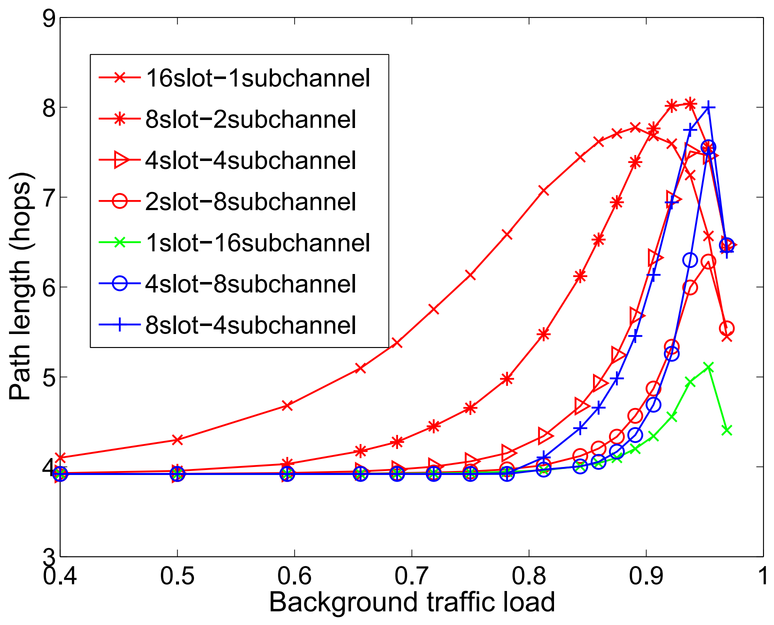


Figure 4.13: Influence of different link partitioning on average path length under background traffic.

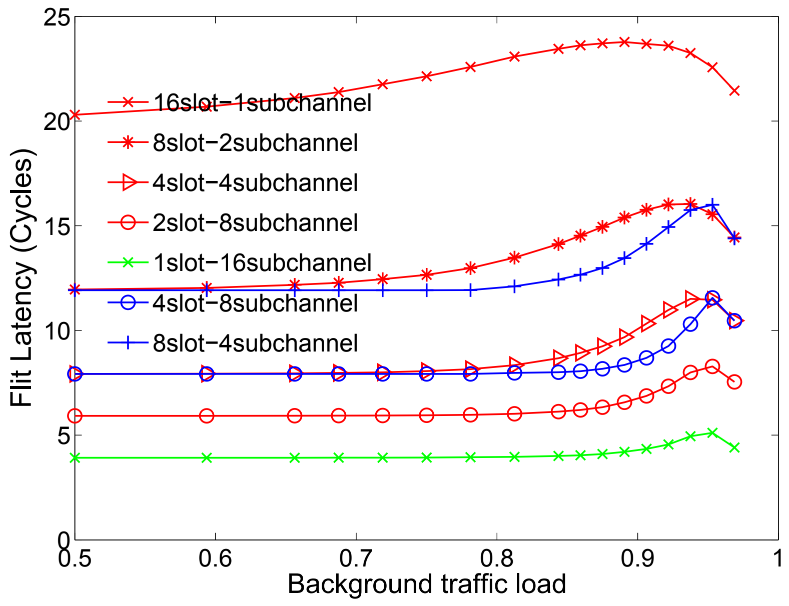


Figure 4.14: Influence of different link partitioning on average delivery latency under background traffic.

As Fig. 4.13 shows, the path in the 1slot-16subchannel partitioning is the shortest, which can be reduced by half against 16slot-1subchannel partitioning (@background traffic load 0.87). When the background traffic is low (lower than 0.93), the path length of 8slot-4subchannel partitioning is shorter than 4slot-4subchannel. If the link is split into more time slots, a message has to wait longer time for its slot to appear, which can induce longer delivery latency per flit. As Fig. 4.14 shows, the delivery latency in the 1slot-

16subchannel partitioning can be reduced by a factor of 6 against 16slot-1subchannel partitioning (@background traffic load 0.87).

4.5 Summary

This chapter presents a dedicated connection allocator, NoCM, which employs the trellis path search algorithm for the connection allocation of combined TDM-SDM CS. The results are summarized below,

- In order to mitigate the poor resource utilization problem of CS, the combined TDM-SDM CS is proposed, which can increase the path diversity and improve sharing of sub-channel among multiple connections.
- For connection allocation of combined TDM-SDM CS, we proposed the TESSA based dedicated allocator.
- The unidirectional and bidirectional TESSA are presented and compared in this chapter. Since the bidirectional approach does the search at two sides, source node (at the initial stage) and destination node (at the last stage) simultaneously, the search time is halved compared to unidirectional approach, while the area and energy consumption is almost the same.
- Finally, in order to investigate and optimize TDM-SDM partitioning strategy, we studied the influence of different TDM-SDM link partitioning strategies on success rate and path length.

Chapter 5

Router Design

This chapter presents the router architecture that combines the circuit-switching network and packet-switching network in order to efficiently and separately handle the GS and BE traffics. The GS message is transmitted over the circuit-switching network along the pre-reserved connection. On the other hand, the packet-switching network can utilize the unreserved resource to transmit BE message, which can increase the resource utilization. Furthermore, when the allocation of the requested connection fails, the corresponding GS message can be transmitted over the packet-switching network, which provides an additional solution to the unsuccessful GS traffic. Part of the results presented here have been previously published in [CMF17b].

5.1 System Model

The system model of a dedicated allocator (i.e. NoCManager) based NoC architecture is illustrated in Fig. 5.1. NoCManager (NoCM) attempts to allocate the appropriate connections when it receives connection requests.

Typically, there are two kinds of routing mechanisms for GS communication, source routing and distributed routing. In source routing, all routing decisions (path information) for a packet are made entirely in the source terminal by table lookup of a precomputed route. The path information is embedded in the packet header by the source node for each hop to indicate which output port to go (3 bits for five directions in 2-D mesh network, east, west, north, south and local). In contrast, in distributed routing, the routing is performed by storing the routing information distributed in the routing nodes along the path rather than in the source terminals. So the routing decision is made at each hop, and each node needs to hold only the routing decision of current node rather than the entire path. The advantage of source routing is that, as long as the source node has received the allocation information from NoCM, then the data transmission can start. However, in distributed routing, the source node has to set up the connection that has to configure the routers

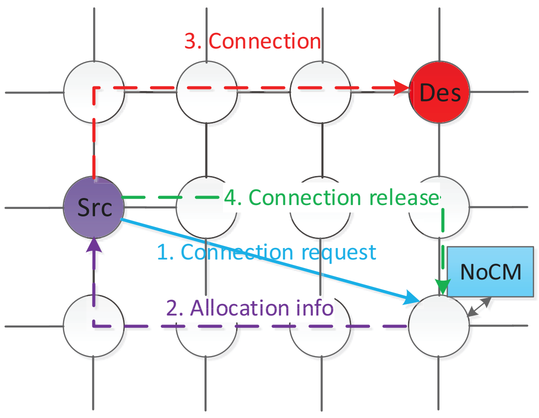


Figure 5.1: System Model of the NoCManager based NoC platform.

along the path hop by hop before starting data transmission. Hence, in order to save connection setup time, the source routing is employed in our system for GS communication. In the source routing, the packets are inserted into the network only at specific time slots, which is regulated by the source terminal, same as in Aelite[HG10].

In our system, when a source node needs a connection, it sends the connection reservation request to the NoCM. The NoCM tries to allocate the connection when it receives the request. When the allocation succeeds, NoCM sends the resulting allocation information to the corresponding source node. After receiving the allocation information, source node starts to transmit GS data along the connection. When the data transfer is finished, the source node deletes the allocation information, and sends a BE packet to inform NoCM that the connection is released. Since the NoC does not drop any packet, this BE packet will reach the NoCM eventually.

There are three possible schemes for the communication between NoCM and NoC nodes, i.e. over a dedicated configuration network, connected via dedicated wires, or use the existing NoC. In order to achieve the high allocation speed, in this thesis, the connection request from source node is sent to NoCM via dedicated wires, while the allocation information from NoCM to source node is delivered over the existing NoC as GS packet. If the allocation of the GS path from NoCM to source node fails, the allocation information will be sent as BE packet. In mesh network, the source node needs $\mathbf{log_2} \boldsymbol{M}$ bits wires to deliver the connection request to indicate which node is the destination, where M is the number of nodes in the NoC. Due to the *partitioning architecture* idea that divides the large system into multiple smaller logic partitions with multiple local managers (explained in section 3.7), each NoCM only manages and connects limited number of nodes in its local region, so the dedicated NoCM-node wires will not be much overhead.

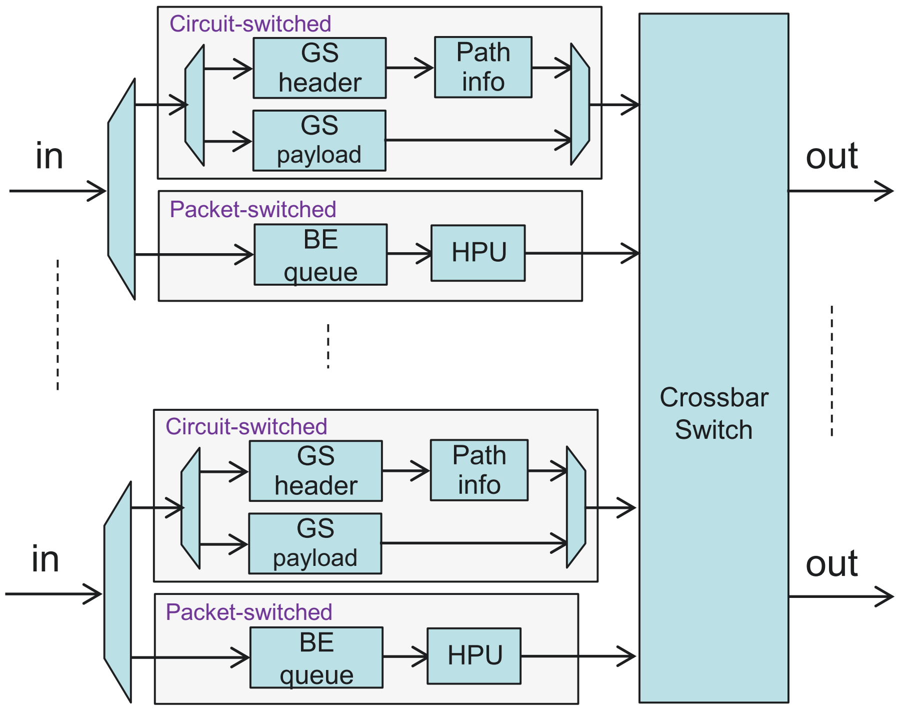


Figure 5.2: The proposed router architecture.

5.2 Proposed Router Architecture

5.2.1 Router architecture

The proposed router architecture consists of two modules as illustrated in Fig. 5.2 : a circuit-switching part for transferring GS packet and a packet-switching part for transferring BE packet. The circuit switching and packet switching share the switch and links. In our system, the packet is divided into small flits. In the flit header, there is a bit to indicate whether this flit is BE or GS. The GS flits are prioritized over BE flits. The BE flit is forwarded only if the output port is not reserved by GS at the moment. Each GS flit has two phits.

The router has five bidirectional ports. Four bidirectional ports are used to connect the four adjacent routers, and the fifth port is used to connect the local module. The routing is distributed such that up to five packets can be simultaneously routed when they request different output ports. Since the BE flit might be stalled in the router if the desired output port is not available, there is a BE queue to buffer the BE data. However, since the GS flit is delivered along pre-scheduled contention-free path, it will never be stalled and there is no need for buffering. The router has 2-stage pipeline, corresponding to a flit size of two phits, and hence a time slot has two cycles.

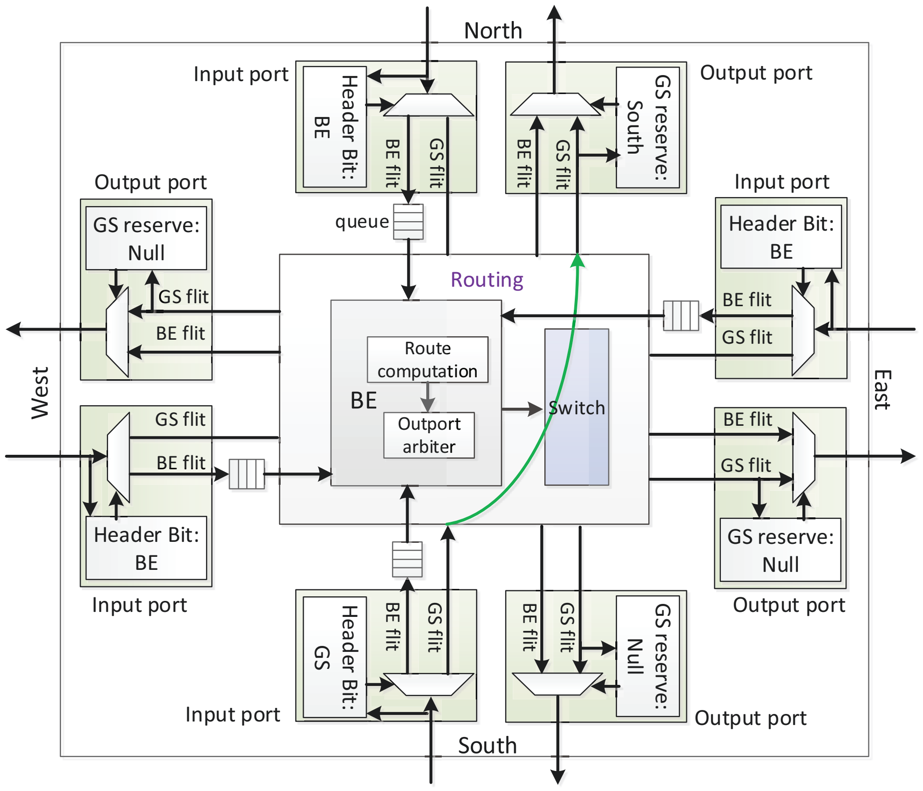


Figure 5.3: Block diagram of the proposed combined BE-GS router with 4 ports. As an example a GS connection (green arrow) from port south to north is established and the GS flits from south to north are directly forwarded.

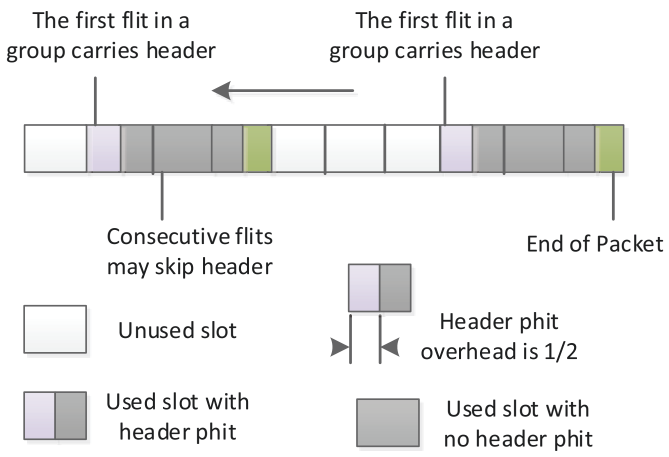

The first flit in a group carries header
The first flit in a group carries header
Consecutive flits may skip header
End of Packet
Unused slot
Header phit overhead is 1/2
Used slot with header phit
Used slot with no header phit

Figure 5.4: GS flits.

5.2.2 Packet-switching Part

The packet-switching part consists of input queues and Header Parsing Units (HPUs), as shown in Fig. 5.2. The input queues store the incoming BE packets. The HPUs do the routing computation to determine the output port to which the fetched packet should be forwarded, depending on the target address in the packet header. The fetched packet is

Table 5.1: The control signals of GS phit

Signals	Usage
00	Idle
01	Header
10	Payload
11	Tail, payload with EoP

then stored in a register until it receives a signal from the switch which indicates that the corresponding output port is available. BE packets will be forwarded only if the desired output port is free. Different routing algorithms are available (deterministic or adaptive) and can be used in a specific NoC realization. However, since in our system the packet switching network only delivers the non-critical BE packets, it is kept as simple as possible. Hence, in our system, the simple deterministic x-y routing is adopted for BE packet, with a stall/go protocol to realize the flow control, similar as in [WF11, MF16]. Round robin arbitration is used in case multiple BE packets want to get access to the same output port in the same clock cycle.

5.2.3 Circuit-switching Part

GS bases on the reservation of a connection between two given nodes, which is scheduled by the NoCM, and has no routing restriction. In the routers along the connection the corresponding input and output ports are reserved for the GS packet. There is no arbiter for the GS packets because the contention is avoided by the schedule of NoCM. The router has no notion of TDM scheme and blindly forwards the data on the input ports to the correct output ports. As an example in Fig. 5.3 where a combined BE-GS router can be found. In this example the router has 4 ports (north, east, south and west). This router is part of a reserved GS connection. The connection crosses the router from south to north. The connection reservation is done by setting the reserve registers in the routers' ports explicitly indicating whether the corresponding output port is a part of a connection and which input port should be connected to it (here set the reserve register of north outport as South inport). Due to the reservation, a GS flit will always be forwarded as soon as it arrives at a router independent of possible congestion which only affects BE traffic.

Each GS flit has two phits, which are grouped into two types: i) header or ii) payload (payload or payload with End-of-Packet). There are 2-bit control signals to indicate different types for each phit, as listed in table 5.1. The control signals indicate the validity of the flit, and is used in order to release or maintain reserved connection. The value '00' indicates the phit is not valid. '01' indicates this is a header phit that contains the path information to the destination node. '10' indicates a payload phit, and the transfer of the GS traffic is ongoing. When the value is '11', it is payload but contains End-of-Packet (EoP) to release the connection.

Table 5.2: Resource Consumption of the Proposed Router with 65 nm technology

Data width (bits)	Area (μm^2)	Power (mw)	Critical path (ns)
16	10875	2.2	0.8
32	18085	3.2	0.8
48	24784	4.2	0.9
64	32801	4.8	0.9
80	40326	6.2	1

In a flit, the first phit is header, which sets up the connection (i.e. configures the corresponding outport GS reserve registers) during routing. The next phit (payload) carries the payload data, and follows the path of the first phit blindly. Since the path is the same for all flits belonging to a specific connection, if there are multiple consecutive flits belonging to the same connection, only the first flit contains the header, and all the other flits only contain the payload data, as shown in Fig. 5.4. The connection (selected output port) remains the same until the coming of the last phit (tail) that contains the EoP flag to release the connection. Within the header phit, the path field holds a sequence of target output ports, encoding the path through the network. Along each hop, the router looks at the lowest bits (3 bits) of the path and then shifts those bits away.

5.3 Synthesis Results

Using Synopsys Design Compiler, the proposed router architecture was synthesized with TSMC 65 nm technology. The size of BE queue is set as 4 words. Since the source routing is employed, the router has no notion of TDM scheme, and thus splitting the link into different time slots does not affect the router resource consumption. In this section, we show the synthesis results with different link data bits, as in table 5.2. The number of data bits does not count the first 3 bits, i.e. 1 bit for indicating BE flit or GS flit and 2 bits for control signals. For example, if the data width is 16 bits, the total link width is 19 bits. The synthesis results show the router area increases linearly with the data width, and every 16 more bits increases the area by about 7500 μm^2.

5.4 Summary

This chapter presents a router architecture for GS and BE traffics. The results are summarized below,

- The router consists of two parts: a circuit-switched part for transferring GS packet and a packet-switched part for transferring BE packet. The packet switched network will utilize the unreserved resource to increase resource utilization, and can provide additional solution to the GS packet that failed in connection allocation.

- In this work, the source node sends the connection request to the central manager via dedicated wires, and the manager sends back the response message as GS packet. Due to the partitioned architecture that each manager only manages its local nodes, the dedicated wires would not be too much overhead. Since the dedicated configuration network that is widely used in previous works is avoided, the hardware cost is reduced, while the setup latency is still guaranteed.

- Finally, the source routing is adopted in our design to save connection setup time, and also to make router design simple since the router has no idea of the complicated TDM scheme but just blindly forwards the GS data. Moreover, it makes it possible to enable ordinary router to support TDM CS without too much modification.

Chapter 6

Conclusions and Future Work

This work was mainly concentrated on the connection allocation for circuit switching networks. In this thesis, we proposed a high performance dedicated centralized connection allocator, NoCManager, to address the dynamic connection allocation problem. The NoC-Manager solves the connection allocation problem based on trellis-search algorithm, which can explore all possible paths between source-destination node pairs within a guaranteed latency. We summarize the main contributions of the dissertation as follows.

1. The path search problem is solved step by step as dynamic programming to reduce computation complexity, as well as to ensure path optimality (shortest path).

2. We search all slots in all directions simultaneously along multi-path, which can complete the path search in a guaranteed latency as well as enhance the allocation success probability.

3. The hardware architecture of NoCM is efficient, and the critical path of each stage is only an OR gate and an AND gate.

4. Different trellis structures, unfolded trellis, folded trellis and bidirectional trellis are presented. The unfolded trellis can achieve high allocation speed while the folded trellis is more efficient in terms of area. The bidirectional trellis can double the allocation speed compared to unidirectional trellis. In different scenarios, different trellis structures can be adopted according to specific system requirements.

5. The Register-Exchange technique is adopted that omits the backtrack step. Hence, compared to forward-backtrack approaches where a backward phase is required to build the path after the forward search, the allocation time is halved.

6. In order to reduce the resource consumption, we proposed the single-layer TESSA, which is much more area and energy efficient than previous multi-layer approach.

7. In order to address the scalability problem of centralized systems, we proposed the partitioned architecture, which divides the original system into multiple smaller partitions served by multiple local managers. Since the managers can work simultaneously, the computation capacity is enhanced. As the NoC nodes only communicate with their local managers, the communication overhead is mitigated.

8. In general, circuit switching NoCs suffer from limited path diversity and resource utilization problem. In order to mitigate this problem, the TDM, SDM and combined TDM and SDM resource sharing mechanisms are proposed to extend the circuit switching. In this thesis, we proposed connection allocation approaches for all these extended circuit switching mechanisms. Moreover, in order to investigate and optimize TDM-SDM partitioning strategy, we studied the influence of different link partitioning strategies with the same total wire resources for TDM-SDM CS on success rate and path length.

Future Research Directions

How to partition the system: The partitioned architecture can enhance the scalability of centralized systems by dividing the original system into multiple smaller partitions with multiple local managers. However, how to partition the system to achieve the best performance, how many partitions do we have to divide for the system? We provide a preliminary suggestion for this problem in this thesis, but we will study deeply in the future to find the best partitioning strategies.

Important request served first: For the NoCM, the incoming connection requests are stored in a FIFO, and the request that comes first is served first. However, some requests may be latency sensitive, and they need to be served first. So in the future, we will assign the requests different priority, and the requests with high priority will be served first.

Weight the link to balance the traffic load: In the network, some links are more critical. So we can assign different weight to different links, and the critical links have higher weight. During the search, the weight along the path is accumulated. At each state, the incoming path with lowest accumulated weight is selected as survivor path. For instance, we can assign the link that provides more free time slots low weight, and assign the link with less free time slots high weight, and thus we can balance the traffic load.

Efficient communication mechanism between routers and NoCMs: The communication between routers and NoCMs is important in centralized system, and sometimes the communication time may even be longer than the path search time. In this thesis, the routers send the requests to NoCM via dedicated wires, and the NoCM sends back the response message to routers as GS packets. The dedicated wire is not scalable (though the partitioned architecture can mitigate the scalability issue), and the GS packet requires to allocate the associated GS path. We will try to find a more efficient communication mechanism between routers and NoCMs to minimize the resource consumption and communication time in the future.

Hierarchy NoC: In this thesis, we apply the trellis path search algorithm for the flat 2-D mesh. However, according to [WPG10, PNPR07], the clustered and hierarchical NoC sometimes may provide better performance by shortening the distance between two modules and adding more bandwidth. In clustered and hierarchical NoC, each cluster can be a 2D-mesh NoC on its own, and the clusters itself are connected with each other in 2D-mesh topology. Hence, in large NoC, we can adopt the hierarchy NoC. In this case, each cluster can be managed by one manager, and managers are connected with each other in 2D-mesh topology, which may mitigate the scalability issue.

Folded bidirectional structure: In this thesis, the bidirectional TESSA is implemented as unfolded structure. However, it can also be implemented as folded structure. In this case, each of the two sides is implemented as folded structure. In a cycle, if the searches from two sides reach the middle stage, but do not meet at any node, at the next cycle, the search from source node will go back to the initial stage and the search from destination node will go to the last stage, and continue the search. The search will be stopped until the searches from two sides meet at the middle stage or exceed limited cycles. The half-folded or quarter-folded structures can also be applied to bidirectional structure.

Release some reserved connections for the important request: Sometimes the connection allocation for the important request may fail because there is no available resource. In this case, we may release some low priority connections for the successful allocation of the important request.

Joint optimization for BE and GS: In this thesis, we mainly focus on GS traffic. In the future, we would pay more attention to do joint optimization for BE and GS.

Bibliography

[BCGK04] Evgeny Bolotin, Israel Cidon, Ran Ginosar, and Avinoam Kolodny. Qnoc: Qos architecture and design process for network on chip. *Journal of systems architecture*, 50(2):105–128, 2004.

[BDM02] Luca Benini and Giovanni De Micheli. Networks on chip: a new paradigm for systems on chip design. In *Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings*, pages 418–419. IEEE, 2002.

[BHM77] Stephen Bradley, Arnoldo Hax, and Thomas Magnanti. Applied mathematical programming. 1977.

[BS05] Tobias Bjerregaard and Jens Sparso. A router architecture for connection-oriented service guarantees in the mango clockless network-on-chip. In *Design, Automation and Test in Europe*, pages 1226–1231. IEEE, 2005.

[CMF16a] Yong Chen, Emil Matus, and Gerhard P Fettweis. Centralized parallel multi-path multi-slot allocation approach for tdm nocs. In *Electrical and Computer Engineering (CCECE'16), 2016 IEEE Canadian Conference on*, pages 1–5. IEEE, 2016.

[CMF16b] Yong Chen, Emil Matus, and Gerhard P Fettweis. Trellis-search based dynamic multi-path connection allocation for tdm-nocs. In *Great Lakes Symposium on VLSI (GLSVLSI'16), 2016 International*, pages 323–328. ACM, 2016.

[CMF17a] Yong Chen, Emil Matus, and Gerhard P Fettweis. Combined centralized and distributed connection allocation in large tdm circuit switching nocs. In *Great Lakes Symposium on VLSI(GLSVLSI'17), 2017 International.* ACM, 2017.

[CMF17b] Yong Chen, Emil Matus, and Gerhard P Fettweis. Combined packet and tdm circuit switching nocs with novel connection configuration mechanism. In *Circuits and Systems (ISCAS'17), 2017 IEEE International Symposium on.* IEEE, 2017.

[CMF17c] Yong Chen, Emil Matus, and Gerhard P Fettweis. Combined tdm and sdm circuit switching nocs with dedicated connection allocator. In *IEEE Annual Symposium on VLSI(ISVLSI'17)*. IEEE, 2017.

[CMF17d] Yong Chen, Emil Matus, and Gerhard P Fettweis. Register-exchange based connection allocator for circuit switching nocs. In *Parallel, Distributed, and Network-Based Processing (PDP'17), 2017 25th Euromicro International Conference on*. IEEE, 2017.

[CMMF] Yong Chen, Emil Matus, Sadia Moriam, and Gerhard P Fettweis. High performance dynamic resource allocation for guaranteed service in network-on-chips. *IEEE Transactions on Emerging Topics in Computing, submitted.*

[CTSM13] Kazem Cheshmi, Jelena Trajkovic, Mohammadreza Soltaniyeh, and Siamak Mohammadi. Quota setting router architecture for quality of service in gals noc. In *2013 International Symposium on Rapid System Prototyping (RSP)*, pages 44–50. IEEE, 2013.

[DT04] William James Dally and Brian Patrick Towles. *Principles and practices of interconnection networks*. Elsevier, 2004.

[EJ13] Ahsen Ejaz and Axel Jantsch. Costs and benefits of flexibility in spatial division circuit switched networks-on-chip. In *Proceedings of the Sixth International Workshop on Network on Chip Architectures*, pages 41–46. ACM, 2013.

[Fet95] Gerhard Fettweis. Algebraic survivor memory management design for viterbi detectors. *IEEE Transactions on communications*, 43(9):2458–2463, 1995.

[GDR05] Kees Goossens, John Dielissen, and Andrei Radulescu. Æthereal network on chip: concepts, architectures, and implementations. *IEEE Design & Test of Computers*, 22(5):414–421, 2005.

[GEEK11] Fayez Gebali, Haytham Elmiligi, and Mohamed Watheq El-Kharashi. *Networks-on-chips: theory and practice*. CRC press, 2011.

[GH10] Kees Goossens and Andreas Hansson. The aethereal network on chip after ten years: Goals, evolution, lessons, and future. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 306–311. IEEE, 2010.

[GHKM11] Boris Grot, Joel Hestness, Stephen W Keckler, and Onur Mutlu. Kilo-noc: a heterogeneous network-on-chip architecture for scalability and service guarantees. In *ACM SIGARCH Computer Architecture News*, volume 39, pages 401–412. ACM, 2011.

[HCG07] Andreas Hansson, Martijn Coenen, and Kees Goossens. Channel trees: reducing latency by sharing time slots in time-multiplexed networks on chip. In *Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, pages 149–154. ACM, 2007.

[Hei14] Jan Heisswolf. *A Scalable and Adaptive Network on Chip for Many-Core Architectures*. PhD thesis, Karlsruhe, Karlsruher Institut für Technologie (KIT), Diss., 2014, 2014.

[HG07] Andreas Hansson and Kees Goossens. Trade-offs in the configuration of a network on chip for multiple use-cases. In *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, pages 233–242. IEEE, 2007.

[HG10] Andreas Hansson and Kees Goossens. *On-chip interconnect with aelite: composable and predictable systems*. Springer Science & Business Media, 2010.

[HSG09] Andreas Hansson, Mahesh Subburaman, and Kees Goossens. aelite: A flit-synchronous network on chip with composable and predictable services. In *Proceedings of the conference on design, automation and test in Europe*, pages 250–255. European Design and Automation Association, 2009.

[JPL08] Natalie D Enright Jerger, Li-Shiuan Peh, and Mikko H Lipasti. Circuit-switched coherence. In *Proceedings of the second ACM/IEEE international symposium on networks-on-chip*, pages 193–202. IEEE Computer Society, 2008.

[Kam07] Matthias Kamuf. Trellis decoding: From algorithm to flexible architectures. *Series of licentiate and doctoral theses*, 2007.

[KS14] Evangelia Kasapaki and Jens Sparsø. Argo: A time-elastic time-division-multiplexed noc using asynchronous routers. In *Asynchronous Circuits and Systems (ASYNC), 2014 20th IEEE International Symposium on*, pages 45–52. IEEE, 2014.

[LJ08] Zhonghai Lu and Axel Jantsch. Tdm virtual-circuit configuration for network-on-chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(8):1021–1034, 2008.

[LJL12] Shaoteng Liu, Axel Jantsch, and Zhonghai Lu. Parallel probing: Dynamic and constant time setup procedure in circuit switching noc. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1289–1294. EDA Consortium, 2012.

[LJL14a] Shaoteng Liu, Axel Jantsch, and Zhonghai Lu. A fair and maximal allocator for single-cycle on-chip homogeneous resource allocation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(10):2230–2234, 2014.

[LJL14b] Shaoteng Liu, Axel Jantsch, and Zhonghai Lu. Parallel probe based dynamic connection setup in tdm nocs. In *Proceedings of the conference on Design, Automation & Test in Europe*, page 239. European Design and Automation Association, 2014.

[LJL15] Shaoteng Liu, Axel Jantsch, and Zhonghai Lu. Multics: Circuit switched noc with multiple sub-networks and sub-channels. *Journal of Systems Architecture*, 61(9):423–434, 2015.

[LKFF12] Shu Lin, Tadao Kasami, Toru Fujiwara, and Marc Fossorier. *Trellises and trellis-based decoding algorithms for linear block codes*, volume 443. Springer Science & Business Media, 2012.

[LL11] Angelo Kuti Lusala and Jean-Didier Legat. Combining sdm-based circuit switching with packet switching in a noc for real-time applications. In *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pages 2505–2508. IEEE, 2011.

[LL12a] Angelo Kuti Lusala and Jean-Didier Legat. Combining sdm-based circuit switching with packet switching in a router for on-chip networks. *International Journal of Reconfigurable Computing*, 2012, 2012.

[LL12b] Angelo Kuti Lusala and Jean-Didier Legat. A sdm-tdm-based circuit-switched router for on-chip networks. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 5(3):15, 2012.

[LMV$^+$08] Anthony Leroy, Dragomir Milojevic, Diederik Verkest, Frédéric Robert, and Francky Catthoor. Concepts and implementation of spatial division multiplexing for guaranteed throughput in networks-on-chip. *IEEE Transactions on Computers*, 57(9):1182–1195, 2008.

[Lou95] H-L Lou. Implementing the viterbi algorithm. *IEEE Signal processing magazine*, 12(5):42–52, 1995.

[MBD$^+$05] Théodore Marescaux, B Bricke, P Debacker, Vincent Nollet, and Henk Corporaal. Dynamic time-slot allocation for qos enabled networks on chip. In *3rd Workshop on Embedded Systems for Real-Time Multimedia, 2005.*, pages 47–52. IEEE, 2005.

[MF16] Sadia Moriam and Gerhard P Fettweis. Fault tolerant deadlock-free adaptive routing algorithms for hexagonal networks-on-chip. In *Digital System Design (DSD), 2016 Euromicro Conference on*, pages 131–137. IEEE, 2016.

[MGK14] Usman Mazhar Mirza, Flavius Gruian, and Krzysztof Kuchcinski. Mapping streaming applications on multiprocessors with time-division-multiplexed network-on-chip. *Computers & Electrical Engineering*, 40(8):276–291, 2014.

[MMB07] Orlando Moreira, Jacob Jan-David Mol, and Marco Bekooij. Online resource management in a multiprocessor with a network-on-chip. In *Proceedings of the 2007 ACM symposium on Applied computing*, pages 1557–1564. ACM, 2007.

[MNTJ04] Mikael Millberg, Erland Nilsson, Rikard Thid, and Axel Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 2, pages 890–895. IEEE, 2004.

[MSAA09] Mehdi Modarressi, Hamid Sarbazi-Azad, and Mohammad Arjomand. A hybrid packet-circuit switched on-chip network based on sdm. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 566–569. European Design and Automation Association, 2009.

[MTSA10] Mehdi Modarressi, Arash Tavakkol, and Hamid Sarbazi-Azad. Virtual point-to-point connections for nocs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(6):855–868, 2010.

[PMM15] Farhad Pakdaman, Abbas Mazloumi, and Mehdi Modarressi. Integrated circuit-packet switching noc with efficient circuit setup mechanism. *The Journal of Supercomputing*, 71(8):2787–2807, 2015.

[PNPR07] Christoph Puttmann, Jorg-Christian Niemann, Mario Porrmann, and Ulrich Ruckert. Giganoc-a hierarchical network-on-chip for scalable chip-multiprocessors. In *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*, pages 495–502. IEEE, 2007.

[RRRM08] Crispin Gomez Requena, Maria Engracia Gomez Requena, Pedro Lopez Rodriguez, and Jose Duato Marin. Exploiting wiring resources on interconnection network: increasing path diversity. In *Parallel, Distributed and Network-Based Processing, 2008. PDP 2008. 16th Euromicro Conference on*, pages 20–29. IEEE, 2008.

[SBSK12] Martin Schoeberl, Florian Brandner, Jens Sparsø, and Evangelia Kasapaki. A statically scheduled time-division-multiplexed network-on-chip for real-time systems. In *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pages 152–160. IEEE, 2012.

[SG11a] Radu Stefan and Kees Goossens. An improved algorithm for slot selection in the æthereal network-on-chip. In *Proceedings of the Fifth International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip*, pages 7–10. ACM, 2011.

[SG11b] Radu Stefan and Kees Goossens. A tdm slot allocation flow based on multipath routing in nocs. *Microprocessors and Microsystems*, 35(2):130–138, 2011.

[Sha15] Liu Shaoteng. *New circuit switching techniques in on-chip networks.* PhD thesis, KTH Royal Institute of Technology, 2015.

[SKS13] Jens Sparsø, Evangelia Kasapaki, and Martin Schoeberl. An area-efficient network interface for a tdm-based network-on-chip. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1044–1047. EDA Consortium, 2013.

[SLB07] Christian Schuck, Stefan Lamparth, and Jurgen Becker. artnoc-a novel multifunctional router architecture for organic computing. In *2007 International Conference on Field Programmable Logic and Applications*, pages 371–376. IEEE, 2007.

[SMG14] Radu Andrei Stefan, Anca Molnos, and Kees Goossens. daelite: A tdm noc supporting qos, multicast, and fast connection set-up. *IEEE Transactions on Computers*, 63(3):583–594, 2014.

[SNG12] Radu Stefan, Ashkan Beyranvand Nejad, and Kees Goossens. Online allocation for contention-free-routing nocs. In *Proceedings of the 2012 Interconnection Network Architecture: On-Chip, Multi-Chip Workshop*, pages 13–16. ACM, 2012.

[Ste12] RA Stefan. *Resource allocation in time-division-multiplexed networks on chip.* TU Delft, Delft University of Technology, 2012.

[WF08] Markus Winter and Gerhard P Fettweis. A network-on-chip channel allocator for run-time task scheduling in multi-processor system-on-chips. In *Digital System Design Architectures, Methods and Tools, 2008. DSD'08. 11th EUROMICRO Conference on*, pages 133–140. IEEE, 2008.

[WF11] Markus Winter and Gerhard P Fettweis. Guaranteed service virtual channel allocation in nocs for run-time task scheduling. In *2011 Design, Automation & Test in Europe*, pages 1–6. IEEE, 2011.

[WL03] Daniel Wiklund and Dake Liu. Socbus: Switched network on chip for hard real time embedded systems. In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pages 8–pp. IEEE, 2003.

[WPG10] Markus Winter, Steffen Prusseit, and P Fettweis Gerhard. Hierarchical routing architectures in clustered 2d-mesh networks-on-chip. In *SoC Design Conference (ISOCC), 2010 International*, pages 388–391. IEEE, 2010.

[WSRS05] Pascal T Wolkotte, Gerard JM Smit, Gerard K Rauwerda, and Lodewijk T Smit. An energy-efficient reconfigurable circuit-switched network-on-chip. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 8–pp. IEEE, 2005.

[YKH10] Zhiyao Joseph Yang, Akash Kumar, and Yajun Ha. An area-efficient dynamically reconfigurable spatial division multiplexing network-on-chip with static throughput guarantee. In *Field-Programmable Technology (FPT), 2010 International Conference on*, pages 389–392. IEEE, 2010.

[YZSZ14] Jieming Yin, Pingqiang Zhou, Sachin S Sapatnekar, and Antonia Zhai. Energy-efficient time-division multiplexed hybrid-switched noc for heterogeneous multicore systems. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pages 293–303. IEEE, 2014.