ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
DEMOCRITUS UNIVERSITY OF THRACE

Low Power Networks-on-Chip

PhD Thesis

Ioannis Seitanidis

Advisor: Assistant Prof. Giorgos Dimitrakopoulos

Department of Electrical and Computer Engineering

Democritus University of Thrace, Xanthi, Greece

Abstract

The importance of System-on-Chip (SoC) interconnect technology is growing with each generation of new SoC devices. Networks-on-chip (NoC) provided the needed disruptive interconnect technology to scale SoC interconnects from simple buses to fully-fledged interconnection networks that behave like mini internets inside the chip. NoCs are now part of all SoCs and are being used in a variety of market segments, ranging from multimedia and telecom to automotive and medical devices.

There are several signs, however, that the current NoC architectures are faced with major challenges in satisfying stringent performance and power requirements. To this end, we propose three complementary techniques that allow for the design of low-power NoC architectures and the development of a methodology for automatically verifying the NoC's peak power consumption.

Two of the techniques tackle the reduction of the power consumed by NoC buffers. In the first approach, a low-cost virtual-channel-based shared buffer is proposed that drops the buffering requirements close to the absolute minimum, without sacrificing network performance. The proposed buffers can be used on the links, as a distributed elastic buffering architecture, or at the inputs and the outputs of NoC routers. The second approach employs multi-bit register composition to reduce the number of register cells of the design, thus significantly reducing the overall clock-tree complexity and power. Multi-bit register composition follows a balanced restructuring approach, where the reduction of the number of registers does not degrade timing slack, wire length or routing utilization.

The introduced low-cost buffers, after being enhanced with self-testability hardware components, have been employed in the design of a scalable distributed NoC architecture that employs virtual channels. The proposed architecture removes the burden of the tight placement of NoC components and allows them to be physically spread throughout the chip, irrespective of the network topology. This feature can be used either for reducing power consumption, or achieving higher clock rates.

The novel micro-architectures and the clock-tree complexity reduction methodology are complemented by a fully-automated methodology that produces appropriate traffic and data patterns that cause peak power consumption within the NoC. In this way, one can have a realistic estimate of a design's peak power consumption, which directly impacts other salient system attributes, such as performance, implementation costs, battery life, and reliability.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor, Giorgos Dimitrakopoulos, for the continuous support during my Ph.D, for his patience, motivation, and immense knowledge. His guidance helped me a lot during my PhD research. His advice on both research and career plans has been priceless. I am grateful to Giorgos for all the professional and research opportunities he offered me.

I would also like to thank the members of my advisory and defence committee, Ioannis Andreadis, Chrysovalantis Kavousianos, Konstantinos Margaritis and Giorgos Syrakoulis, for evaluating and reviewing my work, and for providing insightful comments that helped to improve this thesis. Especially, I would like to thank Chrysostomos Nicopoulos for our productive collaboration and his beneficial contribution in my published work. Also, my sincere thanks goes to Ioannis Karafyllidis for providing me with his helpful advice.

I thank my fellow labmate, Anastasios Psarras, for the stimulating discussions and for working together. Also, I need to thank my colleagues, Pavlos Mattheakis, Laurent Masse-Navete and David Chinnery, for working together at Mentor Graphics.

I am very grateful to the funding received through the Onassis Foundation and its Program of Scholarships for Hellenes.

Also, I need to thank all the people that we hung around during my studies and had a beer, Dimitris, Nikos, James, Aimilios, Giannis, Giannis, Thodoris, Thanasis, Kiki.

A big “thank you” to my sister, Martha, and my parents, Dimitrios and Despoina. I would not have made it without their support and encouragement. This thesis is dedicated to them.

Contents

Chapter 1

Introduction

Information and communication technology innovation has been enabled for more than three decades by advances in semiconductor technology and computer architecture. On the one hand, innovations within the semiconductor domain have repeatedly provided more transistors (Moore's Law) for roughly constant power and cost per chip [49]. On the other hand, computer architects took advantage of these extra available on-chip transistors and provided sophisticated techniques in order to transform the transistor budget into performance benefits.

Nowadays, integration capacity continues with scaling, though with limited performance and power benefit (Moore's gap). Multicore processor chips have been recruited to cover this gap [54]. The additional chip area offered by each new technology generation is filled with more cores that run at nearly constant frequency, thus transforming complex single-core architectures to Multi- or Many-Processor System-on-Chip (MPSoC) architectures. This design paradigm is further enhanced by on-chip cores of different capabilities (aka, heterogeneous multi-cores) in an effort to maximize energy efficiency that avoids dark silicon [32], and maintains high computation power.

In future MPSoCs, the energy expended for data movement across the various chip components (GPUs, CPUs, on-chip memories, DSP accelerators, and IO devices) will have a critical effect on achievable performance and energy dissipation [23]. Every nano-joule of energy used to move data up and down the memory hierarchy, as well as synchroniz-

ing the execution across and between cores takes away from the available (limited) chip power budget, thus limiting the energy available for the actual computations. Therefore, even if compute architectures evolve above expectations and new programming models allow the exploitation of the massive hardware parallelism, it is expected that the utilization of the system (number of processing cores that can be active at a given time) will be prohibited by the interconnection medium.

Network-on-Chip (NoC) provided the needed, disruptive interconnect technology that helped mitigate the interconnect and communication challenges of nanoscale SoC designs [88, 59, 84]. The seminal idea of letting the communication of multicore chips resemble little Internets, where a NoC transfers data between cores in packetized format, promoted system composability and scalability significantly.

1.1 The Importance of Networks-on-Chip

The NoC connects hundreds of disparate IP blocks, each with hundreds of interface signals, and dozens of transaction protocol attributes [46]. It does it in a way that each IP need not know the protocol details of any other, and does it while physically distributed across the chip, as seen in Fig. 1.1. NoC technology is very important for modern chip designs for the following reasons:

It is critical to CPU and GPU/accelerator performance in SoCs: The NoC is the connection between processors and coherence controllers, lastlevel system caches, and DRAM memories as shown in Fig. 1.1(a). Increases in CPU or GPU performance are only useful with a corresponding increase in interconnect bandwidth.

Provides the bandwidth and meets latency requirements: Different IPs have different quality of service (QoS) requirements. CPUs, cameras, and displays are latency critical, while others such DMA engines and video codecs are bandwidth hungry. The interconnect handles IPs of different frequencies, sizes, and protocols and serves as the place to implement QoS controls to avoid performance degradation.

Mixes interface protocols and avoids deadlocks: The heterogeneous compute IPs are prebuilt using mostly standard or custom interfaces

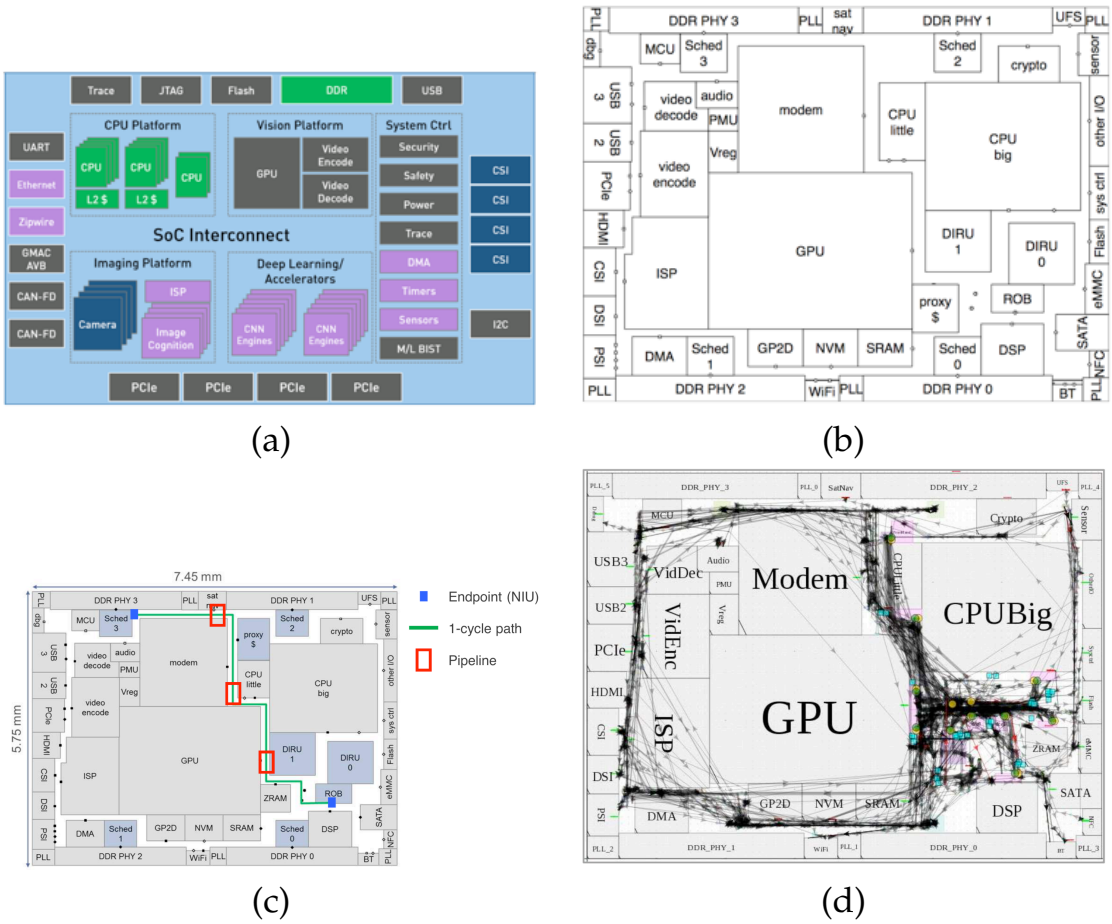


Figure 1.1: The components of a modern SoC and an example physical floorplan where the IP blocks span multiple mm and the NoC should connect all the highlighted dots together (Figures taken from [58]). (a) The components of a modern SoC, (b) the SoC floorplan, (c) NoC links need to be pipelined to cover long wires, and (d) routing congestion increases with NoC's complexity.

that drive what protocols must be supported . When the NoC comes into the picture, it must support all of those different protocols with minimal overhead and a fair degree of configurability. The NoC takes by design all necessary precautions to avoid any deadlock conditions that are possible especially when mixing several protocol semantics.

The last IP configured in a SoC: SoCs are assembled out of internally-developed and commercial IP blocks, and one of the important differentiators is how that IP is assembled together [118]. While the majority of the IP blocks are proven, established, and not very configurable, it is the interconnect IP that is revised many times during the course of a project.

Enables network virtualization: A NoC interconnect may contain multiple physical networks. Over each physical network, there may exist multiple virtual networks, wherein different message types are transmitted over different virtual networks. Virtual channels separate traffic in time instead of space thus simplifying interconnect and reducing retiming requirements, while keeping different traffic classes flowing.

Has the longest physical connections between cells: The NoC connects all the IPs on a chip, even ones that are physically far apart (see Fig. 1.1(b)). As a result, interconnect logic signals fan out over very long distances [58]. Interconnect RC delay (R: Resistance, C: Capacitance) not only has by far the lion's share of the total delay, but its variation across the metal stack has reached over one order of magnitude between the lowest and the highest metal layers, while the resistance contribution of vias increases dramatically [75]. The NoC, which utilizes a mixture of metal layers for inter- and intra-node connections in a modern SoC, feels the largest pressure. This makes it necessary to pipeline the NoC and constrain the placement of pipeline stages within the floorplan (see Fig. 1.1(c)).

Has a big impact on wire routing congestion: The NoC has a particularly high wire-to-gate ratio. This causes parts of the physical layout of the chip to have high wire congestion, which may possibly render the design un-routable, i.e., the wire metal layers do not suffice to route the nets of the design. Careful design of the NoC's topology minimizes congestion within the bandwidth requirements of the design (see Fig. 1.1(d)).

Spans across many chiplets: 2.5D (interposer-based stacks of chiplets) and 3D integration technologies have been pursued as a potential solution to help integrate more functions within confined available dimensions and/or reduce wire dimensions [71]. Such approaches are expected to remove a significant energy overhead related to off-chip communication. However, they will effectively increase the available transistors per chip and the number of endpoints that the NoC should connect.

Critical component for ensuring functional safety: The proliferation of SoC technology into automotive, industrial and medical markets re-

quires the addition of functional safety features currently not present in the available SoCs [47]. Design-for-Functional-Safety accepts the fact that even correctly designed systems may possibly fail and organizes the appropriate reactions needed (e.g., how to return the system to a safe state when detecting a hazardous condition or system malfunction). As a reaction to the increased need for functional safety and keeping the added hardware overhead under control, the amount of extra hardware added to enable functional safety follows the level of integrity needed at each hardware module within the SoC [116]. The NoC that connects all the system's component should receive as much attention as the most critical component that it connects, since it will be the NoC that would deliver the alarm and reaction messages in the case of an emergency, irrespective of the state of the system at the time of failure.

1.2 The Need for Low-Power NoCs

The ideal energy-delay of a NoC traversal should be that of just the on-chip links from the source to the destination, as shown in Fig. 1.2(a). However, this would require each core to have a direct connection with all other cores which is not scalable.

Packet-switched NoCs share links by placing routers at the intersections to manage buffering, arbitration for the crossbar switch, and routing of the flits in order to avoid collisions while the link is occupied, at the cost of increased delay and power, as shown in Fig. 1.2(b). Additional clock endpoints, i.e., the registers of the router's buffers and state, are necessarily added that increase the complexity of the clock tree that needs to drive the distributed placed NoC components.

But a NoC has advantages that can overcome this extra logic and the accompanying area and power overhead. First, bandwidth is enhanced by sharing network channels across all clients. In contrast, dedicated wiring often leaves many wiring resources idle at times of peak load. Second, transferring packet data requires fewer wires than a traditional bus. For example, control information is encoded in the packets rather than on separate wires [55]. In addition, data paths can be narrower because they can be sized for sustainable bandwidth rather than the max-

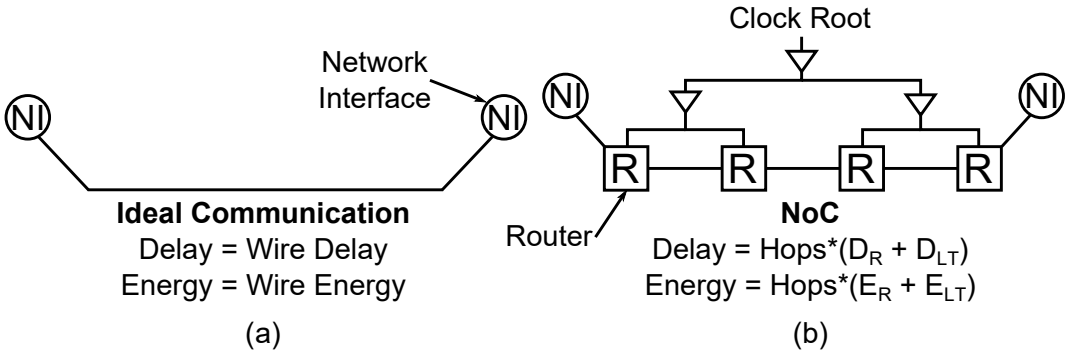


Figure 1.2: Ideal vs real energy for on-chip data transfer. On one hand networks enable sharing of resources, which reduces the overall power consumption relative to dedicated wiring, but, on the other hand, considerable power overhead still remains that needs to be removed. D_R and D_{LT} represent the delay in cycles of router and link traversal, while E_R and E_{LT} are the associated energy costs.

imum required on a given cycle, as in a crossbar. Narrower data paths require narrower FIFOs, which consume less logic and die area [24].

Also, using a NoC to replace top-level wiring has advantages of structure, performance, and modularity. A network structures the top-level wires simplifying their layout and giving them well-controlled electrical parameters. These well controlled electrical parameters in turn enable the use of high-performance circuits that result in significantly lower power dissipation and higher bandwidth that is possible with conventional circuits [25].

Finally, due to the locality of traffic, the NoC guarantees that only the wires that should receive traffic would be activated and thus allowing for energy-proportional communication, where power consumption is proportional to the amount of traffic it is moving [40].

Even if the NoC is a more scalable and power efficient solution relative to dedicated global wiring it still has a lot of room of improvement. At the moment a NoC constitutes around 10-15% of SoC's power consumption mostly due to buffering (sequential logic and clocking) and link traversal [29]. As the number of cores increases, it is going to become a significant challenge to sustain the current per-core bandwidth given the super-linear increase in network power consumption. The projected interconnect power by scaling up existing network designs far exceeds a practical power budget. A low power interconnect design is imperative to the realization of a 1000 core processor.

1.3 Thesis Contribution

In this work, we adopt a power-driven approach towards scaling on-chip interconnects to meet the bandwidth and latency targets necessary in high-performance processors. Our contributions are threefold:

We focus both on reducing the power overhead of buffering and and flow control by proposing new simplified virtual-channel buffers and simplified distributed NoC architectures. We extend elastic buffer (EB) architectures to support multiple virtual channels (VCs), and we derive ElastiStore, a novel lightweight EB architecture that minimizes buffering requirements without sacrificing performance [126]. ElastiStore uses just one register per VC and a shared buffer sized large enough to merely cover the round-trip time that appears either on the NoC links or due to the internal pipeline of the NoC routers. The integration of the proposed EB scheme in the NoC router enables the design of efficient architectures, which offer the same performance as baseline VC-based routers, albeit at a significantly lower cost.

ElastiStore is envisioned as an archetypical primitive for future, extremely low-cost NoC router implementations, where the performance and functionality enhancements provided by VCs cannot be sacrificed.

Building on top of ElastiStore, we present ElastiNoC, a novel distributed VC-based router architecture that enjoys all the benefits offered by VCs and leads to efficient silicon-aware implementations [128]. The proposed architecture utilizes ElastiStore buffering and allows for modular pipelined organizations that increase the clock frequency. Moreover, it offers maximum freedom in terms of physical placement, by allowing the NoC components to be physically spread throughout the chip, irrespective of the network topology. The combined effect of all supported features enables significant delay reductions under equal performance, when compared to state-of-the-art VC-based NoC implementations, which can be equally translated to significant power reductions under the same performance.

ElastiNoC offers a shift in the design philosophy of VC-based NoC routers from centralized and monolithic structures to modular and distributed components that can be stitched together to form a larger entity,

while still providing full VC support and extensive flexibility during physical placement.

Moreover, the careful addition of self-test structures allows ElastiNoC to enjoy fully distributed Built-In Self Testability (BIST), where testing unfolds in phases and reaches high fault coverage with small test application time. Self testing imbues ElastiNoC with a valuable (albeit often ignored) asset, which greatly enhances its viability to much larger future designs.

The distributed nature of every NoC design imposes additional constraints to the global clock tree of the SoC. Clock pins of NoC buffers are distributed throughout the layout thus increasing the number of clock tree branches, the clock wirelength as well as the clock buffers that need to drive those pins. To reduce the complexity of the clock tree the registers implementing the NoC's buffers can be replaced by multi-bit registers (MBRs) thus effectively reducing significantly the number of clock endpoints. The proposed MBR composition follows a balanced restructuring approach that is applied after global or detailed placement [122, 123]. Its goal is to minimize the total number of registers, and simplify subsequent clock tree synthesis, while taking care that any potential degradations in timing slack, wire length, or routing congestion do not offset the power benefits of a lighter clock tree. The proposed methodology, although initially tackled the clock-tree power of NoC's buffers, can find wider applicability by identifying nearby compatible registers that can be merged without degrading timing, and without reducing the "useful clock skew" potential.

Finally, early estimation of the peak power consumption of a system under development is crucial in assessing the design's reliability and thermal profile, and for benchmarking various architectural options and chip-level power management features. In this thesis, we present a high-level systematic methodology for generating the appropriate traffic patterns that trigger the peak power consumption in a Network-on-Chip (NoC), irrespective of the latter's structural and functional properties. The generation of peak-power traffic is performed by solving a novel optimization problem based on Integer Linear Programming (ILP) [125] (or by identifying hamiltonian cycles on channel dependeny graphs [124]), which models the traffic that can realistically flow in the network,

thus avoiding any fake and pessimistic scenarios. This formulation can handle arbitrary network configurations and routing algorithms, including heterogeneous network topologies with multiple link widths and voltage/clock domains. The proposed technique maximizes both the network utilization and the data switching activity, thereby causing, on average, 4× higher power consumption than synthetic traffic patterns with random behavior. Most importantly, the proposed method reveals the realistic ceiling of the NoC's peak power consumption, by reporting significantly lower peak power (3× less), as compared to fake worst-case scenarios that can never, in fact, occur during the NoC's normal operation.

1.4 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2 discusses the basic architecture and microarchitecture properties of Networks-on-Chip covering partially the properties of their physical implementation focusing on their power consumption. Additionally, low-power techniques are discussed and their applicability on NoCs is described, accompanied by a complete review of NoC-related low-power techniques covering among others dynamic voltage and frequency scaling, clock/power gating and link encoding for minimum switching activity, as well as NoC topology and routing reconfiguration and NoC dynamic clock throttling.

Chapter 3 presents the basic architecture of the ElastiStore and the integration into NoC routers. Initially, the operation of elastic flow control for the cases of single and multiple VCs is analyzed focusing on the relation between the chosen buffering architecture and the achieved throughput. Next the core architecture of ElastiStore buffers is described in detail together with its relationship to arbitrary link round-trip times. The integration of ElastiStore buffers within NoC routers is presented later on together the experimental results that demonstrate, by the integration of ElastiStore in both single-cycle and pipelined NoC routers, the same performance as baseline VC-based routers, albeit at a significantly lower area cost.

Chapter 4 introduces ElastiNoC architecture which enables the modular construction of pipelined routers. At first, we develop a simple intuitive analytical model that connects the network latency with the routers' oper- ating clock frequency and their internal pipeline organization. The goal is to construct a model that enables the designer to derive a first-order approximation to an optimal configuration, given certain parametrical constraints. Motivated by the outcome of the derived model, we produce new designs for NoCs that yields highly-scalable NoC implementations. The augmentation of ElastiNoC with self-testability capabilities is also presented, which enables NoC routers to conduct testing sessions in a modular manner over multiple phases, that achieve high fault coverage (in excess of 99%). The experimental results presented based on both network simulations and standard-cell-based hardware synthesis implementations validate the efficacy and efficiency of ElastiNoC.

Chapter 5 discusses the multi-bit register (MBR) composition techniques, to reduce the complexity of the clock tree. First, we discuss the goals of a successful MBR composition and present briefly the overall MBR composition flow. Then, we present the details of MBR decomposition and optimization. In the following, the compatibility criteria that determine which registers can be composed into MBRs are introduced, together with the novel ILP formulation that helps in minimizing the number of registers. The placement and mapping of the assigned MBRs to specific cells of the library is also discussed. The chapter concludes with the introduction of the post-MBR composition optimization steps and the presentation of a complete set of experimental results covering industrial designs.

Chapter 6 discusses the importance of the estimation of the peak power consumption of a system and presents a novel methodology for generating traffic patterns that trigger NoCs' peak power. Our presentation begins with the intuition that we developed for selecting the format of the traffic patterns needed for triggering the peak power consumption of the NoC in a controllable manner. Then, we introduce the ILP formulation that generates the appropriate traffic pattern for each NoC configuration. Our presentation is completed with the experimental results that prove the superiority of the proposed methodology.

Finally, a summary of our work including also its future research aspects is covered in Chapter 7.

Chapter 2

Background and Related Work

The number of components on a chip is rapidly growing due to increasing levels of integration, system complexity and shrinking transistor geometry. In both System-on-Chip (SoC) and Chip Multi-Processors (CMP) systems, the on-chip interconnect plays a vital role in providing high-performance communication between the various components. Due to scalability limitations of traditional buses and crossbar based interconnects, Network-on-Chip (NoC) has emerged as a paradigm to interconnect a large number of components on the chip. NoC is a global shared communication infrastructure made up of several routing nodes interconnected with each other using point-to-point physical links.

2.1 Overall NoC organization

The general structure of the interconnect system is shown in Figure 2.1. Each core, acting as a communication master, is presenting read and write transactions (or cache coherence requests) to the interconnect that gives the illusion of a slave to each master IP. The master interface of the cores is attached to the network interfaces (NIs) of the interconnect that translate the interface protocol to an internal packetized flow-control protocol that allows network transactions to travel in the network and get delivered to their destination as network packets. At the other end, the interconnect is attached to one or many ports of the hosting system that presents itself as one or many slave interfaces. The interconnect should play the role of the master for this connection. This is achieved

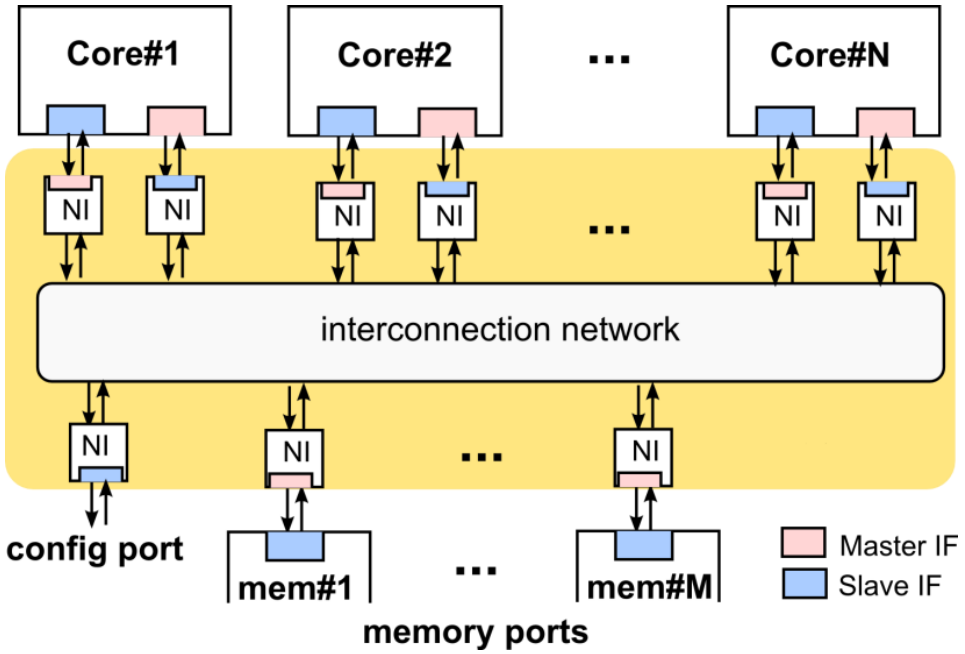


Figure 2.1: The general structure of the interconnect architecture.

via the reverse network interfaces that translate the intra-network protocol of packet transfers to master read and write transactions.

The NI is responsible for both sending packets to the network as well as receiving packets from the network and after the appropriate manipulation to present it to the connected IP core according to the semantics of the interface.

2.1.1 Links and Packets

Packets containing more than one word (aka flit – flow-control digit) are serialized and pass the link in multiple clock cycles. The first flit, called the header of the packet, denotes the beginning of the packet and contains the identification and addressing information needed by the packet, including the address of its source and its destination. The last flit of the packet is called the tail flit and all intermediate flits are called the body flits. Each packet should travel on each link of the network as a unified entity since only the header of the packet carries all necessary information about the packet's source and destination.

Figure 2.2 depicts the wires needed in a network-on-chip channel that supports many-flit packets. Besides data wires and necessary flow con-

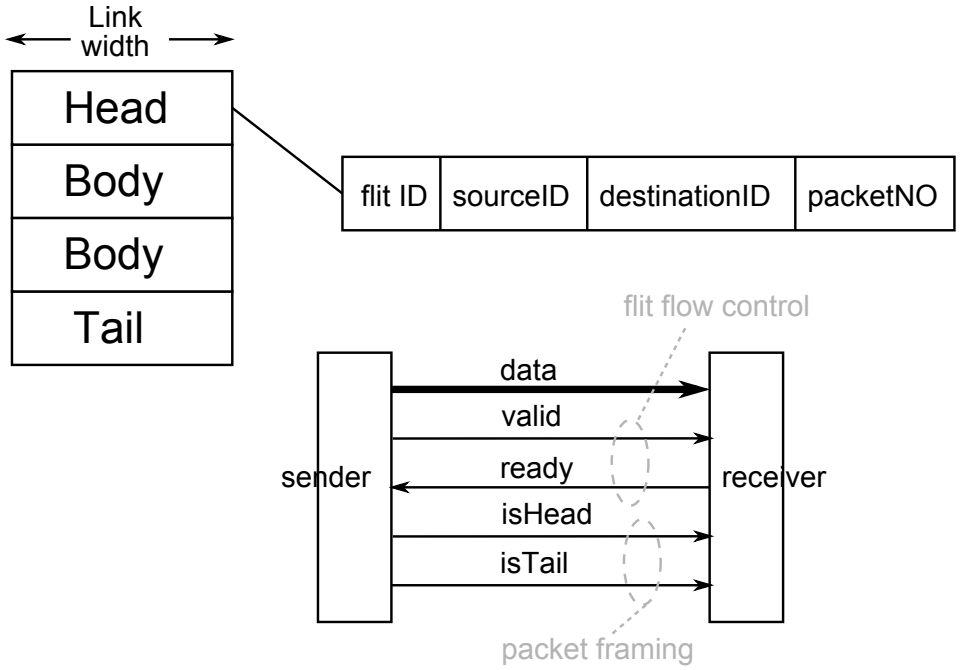


Figure 2.2: The additional signals added in the links to distinguish the type of each arriving flit.

trol signals (ready/valid is used in this example) two additional wires, e.g., isHead and isTail are needed that encode the type of the flit that traverses the channel per cycle. isHead and isTail signals are mutually exclusive and cannot be asserted simultaneously. When they are both inactive and valid=1, it means that the channel holds a body flit.

2.1.2 Router Functionality

The interconnection network which is the core of the NoC should transfer the packets produced at the NIs to their destination. The structure of the network and the possible paths between any pair of source and destination is determined by the network topology. The router is the hardware module placed at the crossroads of network topologies and should forward to the correct output all traffic that arrives at its inputs (see Figure 2.3(a)). Each input/output port of the router that is connected to the networks links should be independently flow controlled providing lossless operation and high communication throughput.

The router should support in parallel all input-output permutations, as depicted in Figure 2.3(b). When only one input requests a specific

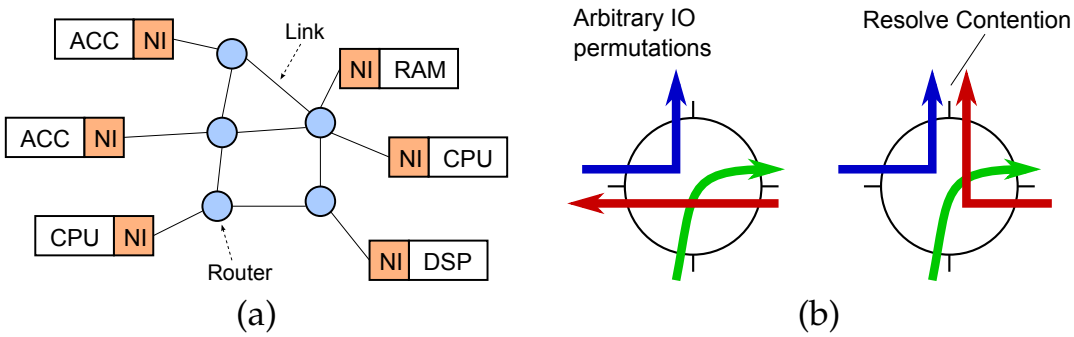


Figure 2.3: A network-on-chip consisting of routers and links that reach the systems modules via the network interfaces.

output, the router should connect the corresponding input with the designated output. When two or more inputs compete for gaining access to the same output in the same cycle the router is responsible for resolving the contention. This means that only one input will gain access to the output port. The flits of the input that lost stay it the input buffer of the current router and retry in the next cycle. Alternatively, the flits of the lost input can be misrouted to the first available output and move to another node of the network, hoping that they will reach from there their destination. Misrouting actually does not resolve contention, but spreads it in space (in the network), while the baseline approach spreads contention in time by allocating one output to one input in each clock cycle.

2.1.3 Topologies

In a network, the topology is the arrangement of nodes and channels. It determines the interconnection of nodes and can usually be modeled as a graph. The structure of the network topology directly determines the maximum communication throughput among the connected nodes, the complexity of the routers that implement the topology and the latency that each packet experiences when travelling in the network. In most cases, packet latency and router complexity are inversely proportional. In other words, the smaller the hop count allowed by the topology and the associated routing algorithm, the more number of input and output ports are needed by the routers of the network; hop count is the number of routers that a packet must pass in order to travel from source to destination.

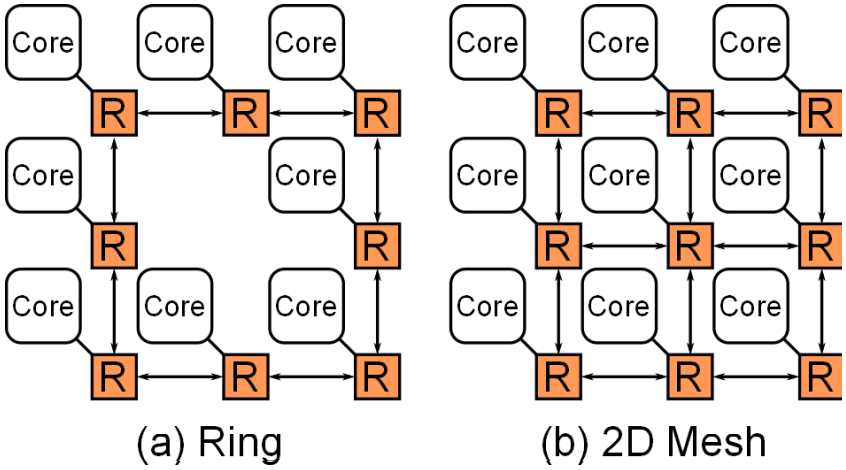


Figure 2.4: Two different topologies: (a) a 8-node ring and (b) a 9-node 2D Mesh.

Figure 2.4 shows two different topologies used to connect homogeneous tiled cores: (a) a ring and (b) a 2D Mesh. The NoC follows the physical layout of the existing IP cores of the chip and it should fit in the available space. In case of homogeneous CMPs, designers can choose between regular and homogeneous topologies, like rings and meshes.

Topology affects the physical characteristics of the links. Longer wires need repeaters to achieve the timing constraints increasing the power consumption. Moreover, the link may have to be pipelined if the constraints are not met, which lead to increased round trip times and more buffering in the routers.

2.1.4 Routing

Once a packet is injected in the network a mechanism is required that will inform the packet which output to follow at each intermediate router, in order to get closer to its destination. This mechanism is implemented by the routing computation logic at each router. Routing computation implements the routing algorithm which is a network wide operation that manages the paths that the packets should follow when travelling in the network. Consequently, each router should respect the properties of the routing algorithm and forward the incoming packets to the appropriate output following the path decided by the routing algorithm.

The routing algorithm, if not designed appropriately, may lead to a

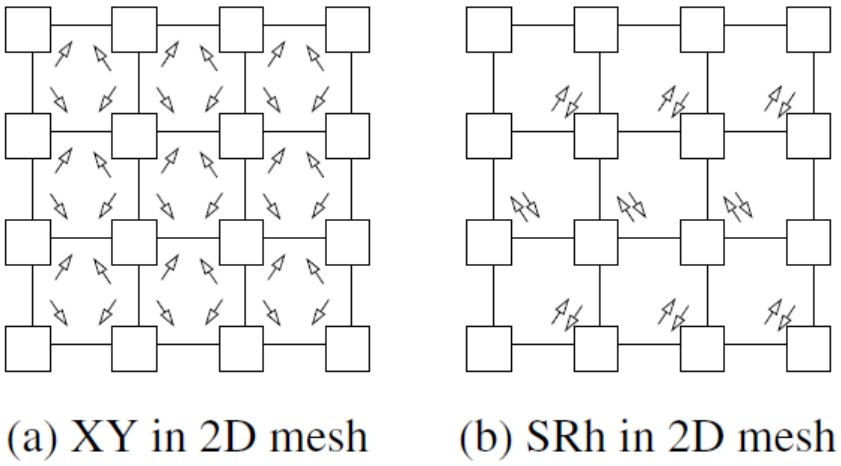


Figure 2.5: (a) A 2D mesh network with XY routing algorithm and (b) A 2D mesh network with SRh routing algorithm.

deadlock condition. A deadlock is formed when a cyclic dependency chain is formed among packets requesting an occupied resource of the network, thus no packet can make forward progress any more.

Deadlock-free routing algorithms guarantee that no cyclic dependencies are possible within the network, by limiting the paths a packet can follow to reach its destination. This path limiting process is actually a network-level planning on the allowed turns that a packet can take when traversing the networks links and routers. Figure 2.5 shows examples of deadlock-free routes, where the arrows on the network constitute carefully-designed turn prohibits. For example, a packet routed under the deadlock-free XY routing algorithm in a 2D mesh is only allowed to turn once during its route, from direction X to Y.

Alternatively, routing deadlocks can be avoided by the employment of virtual channels. Virtual channels correspond to parallel per input buffers that the routing algorithm uses to avoid any cycle formation: the acquisition of each virtual channel by a packet when moving in the network is done in a fixed and predefined order that avoids completely cyclic dependencies and allows more freedom in routing packets inside the network. In fact, in a graphic interpretation, virtual channels actually transform every cyclic connection to a screw-like connection where multiple crossings of the same channel are only allowed on a different virtual channel. Although VCs offer the maximum possible flexibility

the design of virtual-channel-based routers incurs a significant cost relative to simpler wormhole routers. This cost can be possibly amortized when large number of virtual channels should be supported.

2.2 Link-Level Flow Control

The flow of flits on the links of the network is determined by the credit-based flow control policy that allows for safe and lossless operation with the minimum buffering requirements. The sender tries to send its valid flits to the link as long as the buffer placed at the receivers side has available slots to host the incoming word. The buffer at the receiver should behave as a FIFO queue even if it has at least on position.

Credit-based flow control gives to the sender all the necessary knowledge to start, stop, and resume the transmission. In credit-based flow control, the sender explicitly keeps track of the available buffer slots of the receiver. The number of available slots is called credits and they are stored at the sender side in a credit counter. When the number of credits is larger than zero then the sender is allowed to send a new word consuming one available credit. At each new transmission the credit counter is decremented by one reflecting that one less buffer slot at the receive side is now available. When one flit is consumed at the receive side, leaving the input buffer of the receiver, the sender is notified via the credit update signal to increase the available credit count. No word can be dropped or lost since each word reaches the receiver after having first consumed the credit associated with a free buffer position.

An example of the operation of the credit-based flow control is shown in Figure 2.6. At the beginning the available credits of the sender are reset to 3 meaning that the sender can utilize at most 3 slots of the receivers buffer. When the number of available credits is larger than 0 the sender sends out a new word. Whenever the sink of the receiver consumes one new word, the receiver sends a credit update that reaches the sender one cycle later and it increases the credit counter. The credit updates, although arrive with cycle delay, they are immediately consumed in the same cycle. This immediate credit reuse is clearly shown in the clock cycles where the available credits are denoted as 0*. In those clock cycles, the credit counter that was originally equal to 0 stays at 0, since, it is

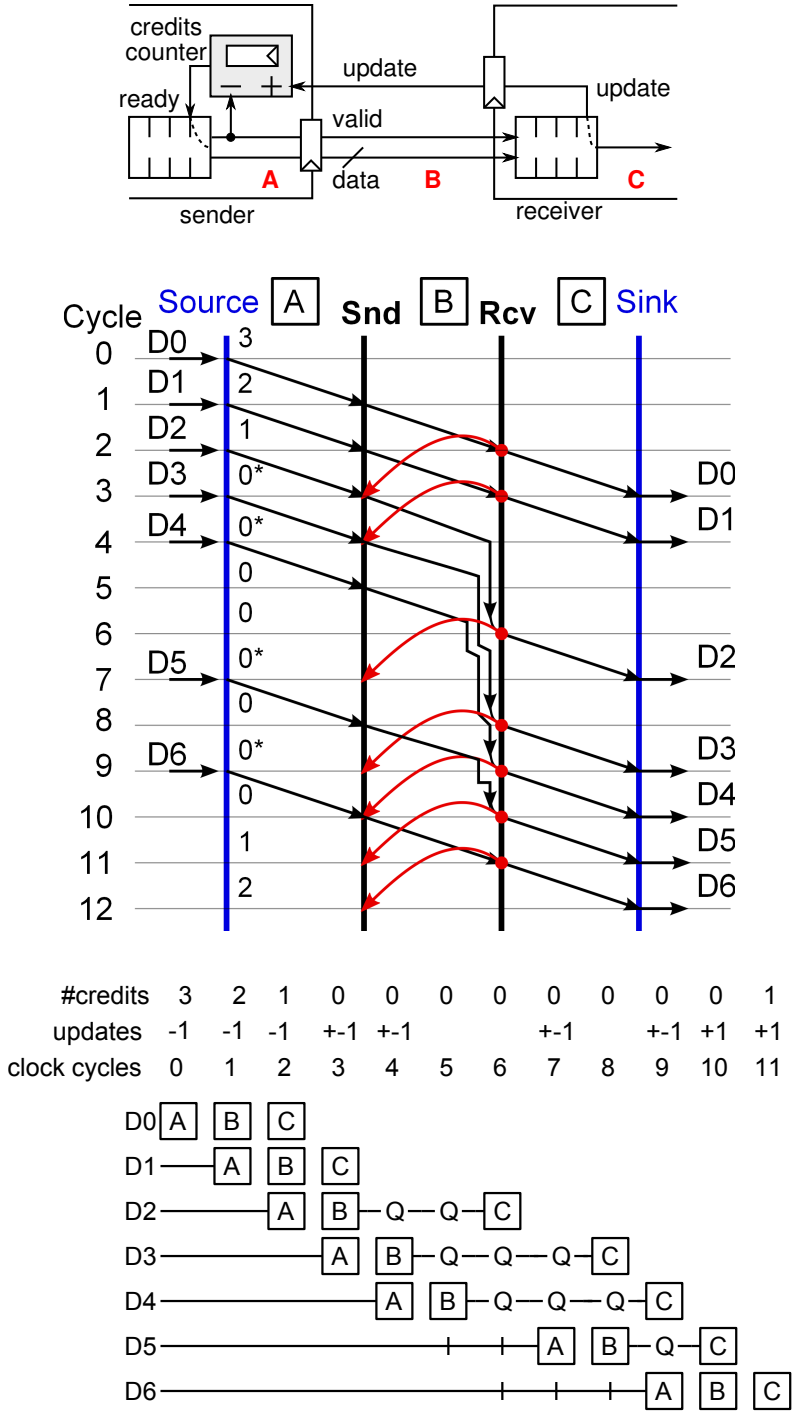


Figure 2.6: An example of data transfers on a link between a sender and a receiver governed by credit-based flow control. The figure includes the organization of the sender and receiver pair and the flow of information in time and space.

simultaneously incremented due to credit update and decremented due to the transmission of a new word. When the words are not drained at

the sink they are buffered at the receiver. No word can be dropped or lost since each word reaches the receiver after having first consumed the credit associated with a free buffer position.

The throughput of transmission on each link of the network is closely related to the number of credits used by the sender and the number of clock cycles that pass from the time a credit update leaves the sender until the first new word that consumes this returned credit reaches the input buffer of the receiver.

In the general case that (a) the credit update signal reaches the credit counter at the sender after L_b cycles from its generation and (b) the data in the forward direction reach the receiver L_f cycles after they are produced at the sender, the minimum number of buffers needed at the receiver under credit-based flow control to guarantee lossless operation and 100% throughput is $L_f + L_b$. Note that credit consumption is done once the data forwards the data to the link without any additional delay, i.e., the latency of the valid signal that consumes the credit is always zero.

2.3 Router Microarchitecture

The NoC router needs to performs a series of per-input and per-output tasks before being able to send incoming traffic (or that stored in the input buffers) at the output ports. Each packet should first complete routing computation (RC) (using its head flit) that informs the packet which output to follow at each router, in order to get closer to its destination. The result of RC is written in the outPort register (one per input) and used in the same cycle by the head flit of the packet. Then, each packet should fight for gaining access to the output via switch allocation/arbitration (SA) and move to the appropriate output via the multiplexers of the crossbar (Switch Traversal ST). Eventually, the packet will reach the next router, after leaving the output buffer and crossing the link (Link Traversal LT). We assume that the input/output links of the router are independently flow controlled, following the credit-based flow control.

A block diagram of NoC router is shown in Figure 2.7. The output buffers of the router can be either simple pipeline registers or nor-

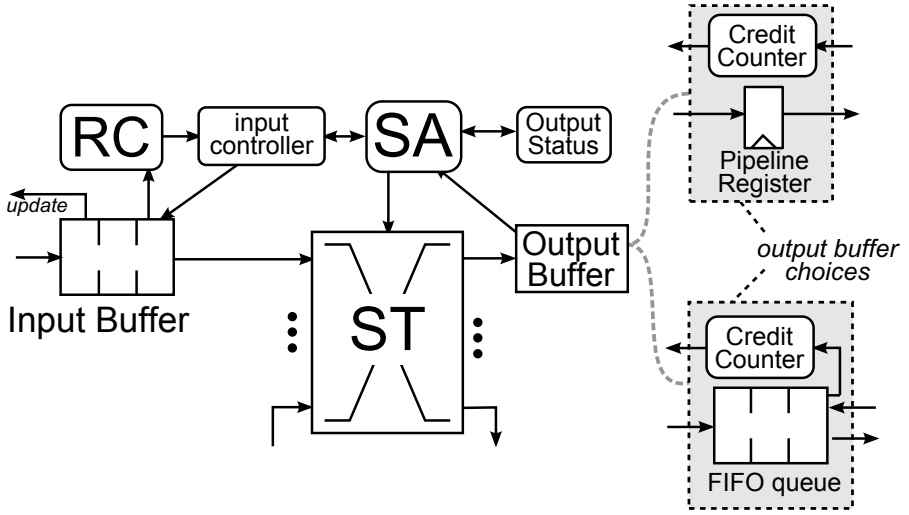


Figure 2.7: An abstract organization of the NoC router.

mal flow-controlled buffers (FIFO queues). In the first case, the credit counter refers to the available buffer slots of the buffer at the input of the next router, while in the second case the credit counter mirrors the available buffers of the local output buffer. In this design, we adopt the first design option and assume that the output of the router consists of a simple pipeline register for the signals in the forward (valid, data) and in the backward direction (credit update).

After the calculation of its destination output port through RC, a packet must generate a proper request to SA, according to the current states of the input. Each input wants to send a packet that contains one head flit, a number of body flits and a tail flit that declares the end of the packet. Since each flit should travel on the shared link as an atomic entity the link should be allocated to the packet as a whole: The head flit will arbitrate with the head flits of the other inputs and once it wins it will lock the access to the output. This lock will be released only by the tail flit of the packet.

Each input before transferring an active request to the arbiter needs also to guarantee that there is at least one free buffer slot at the sink by checking the credit counter at the selected output.

The grants of the output arbiters of SA play a triple role:

- They drive the select signals of the output multiplexer that will switch to the output the flit of the selected input.

- They are used per-input to set the winning input. In the next cycles, the body and the tail flits do not need to qualify their requests again.
- They drive the pop signals of the winning input buffers causing a dequeue operation to the corresponding input buffer. The inputs that did not win will see not see a pop and thus they will keep their data in their buffer. At the same time, the input buffer that dequeues a flit informs the previous router that a buffer slot is emptied, using the credit update mechanism.

At the output side, if a flit has won in per-output arbitration (SA) and is about to be stored at the output buffer, it must consume a credit (Credit Consume CC) that is, decrease the credit counters value to reflect the current free slot availability of the output buffer (placed at the input of the next router). If the granted flit was a head or a tail flit, the output is also locked for the whole duration of the packet (State Update SU).

Updating the output state and consuming the necessary credits is normally triggered once a flit traverses the output multiplexer, and is about to be written to the output pipeline register. Although this might seem a safe and reasonable choice it introduces some non-negligible delay overhead. The problem can be completely eliminated by making an important observation: both CC and SU can be executed without the need of knowing specifically which input allocates the output port or consumes an output credit. Simply knowing that some input wins in arbitration or sends a flit forward, suffices. Therefore, since the SA result is not required, those operations can occur in parallel to SA. For SU, this translates to checking whether any request from a head or a tail flit exists, while CC decrements the output credit counter if the corresponding output receives at least one request.

An example of the operation of the described NoC router can be seen in Figure 2.8. The execution diagram refers to the behavior of a single input that receives a consecutive traffic of incoming packets consisting of 3 flits (one head, one body and one tail flit). In parallel, the rest inputs follow a similar execution assuming that their requests and data move to a different output. A certain output can host the packet (on a flit-by-flit basis) of only one input at a time. In cycle 0, a head flit is written at the

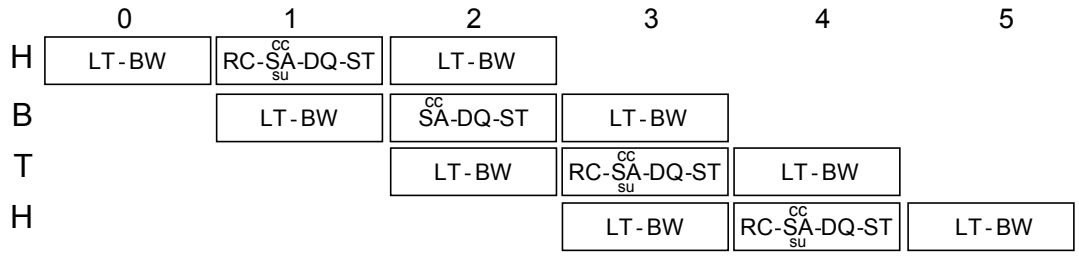


Figure 2.8: The tasks that need to be performed for the flits of a packet arriving at an input of the router.

input buffer (Buffer Write BW), after crossing the link (Link Traversal LT). The flit immediately appears at the frontmost position of the input buffer in cycle 1, and is able to execute all necessary operations within a single cycle: (a) the flits destination field feeds the RC and the request generation logic; (b) supposing that the output is available, the flit performs SA, while in parallel it consumes a credit (CC) and updates (SU) the output state; finally, (c) the grant produced by SA is used to dequeue (DQ) the flit from the input buffer, in order to traverse the crossbar (ST). As the head flit moves forward to the output pipeline register, a body flit is written at the input buffer. The output buffer has enough credits available, thus allowing the newly arrived body flit to generate a request to SA. Being the only active request (the requests of all other inputs for the selected output are nullified), the body flit is granted to move forward, after consuming a credit. In the same cycle, the head flit is moving to the next router. In cycle 3, the tail flit follows the same procedure, performing SU as well, in order to release the allocated port, while the next packets head flit arrives. In cycle 4, all previously allocated resources are already free and the following packet is able to generate a request and participate in arbitration, whatever its destined output port might be. Observing the rate of incoming and outgoing flits of this input, one would notice that a flit only requires a single cycle to exit the router, and no extra cycles are added in between packets. The only conditions under which a flit may be stalled is (a) if all the output buffers slots are full, or (b) a head flit loses in arbitration (in this case the output port is still utilized, but by a different input).

The support of virtual channels increases the complexity of the router. The input buffers are separated to independent queues (one per VC) and incoming packets are placed to the corresponding buffer depending on

the VC they belong to. After each packet independently need to find its way towards the requested output after allocating a virtual channel at the next router and time slot of the output port. A block diagram of a VC-based router with the processing flow of the incoming flits is shown in Figure 2.9.

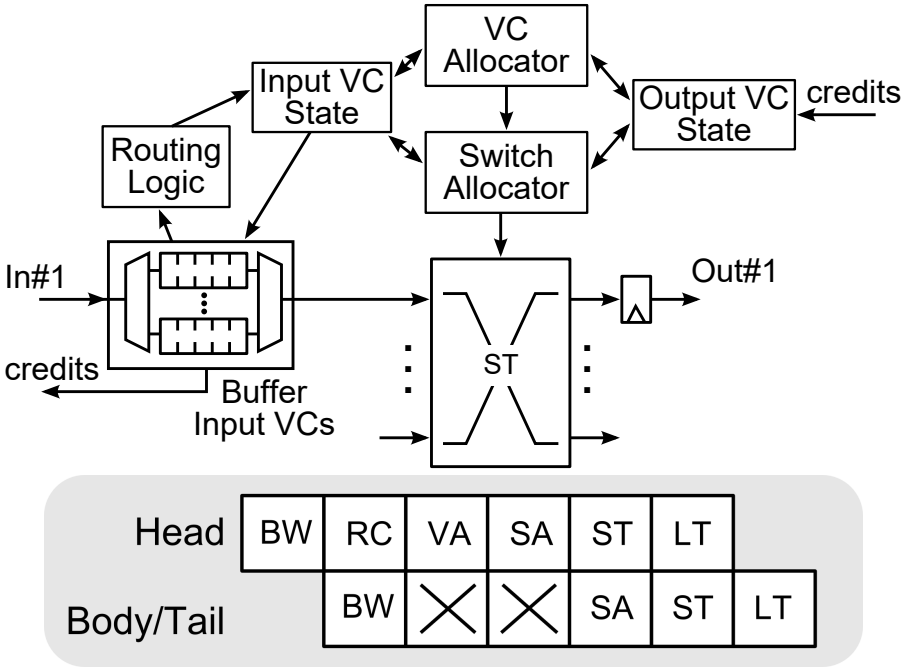


Figure 2.9: A block diagram of a NoC router which supports VCs. Also, the processing steps for the incoming flits at the router are presented.

New flits arrived at the inputs are stored into the input VC buffers of the router(BW). In the following clock cycles, they must find the correct output port and forward to the next router. A head flit, the first flit of new packet, must find its output destination port via a routing computation (RC) unit. Also, it must choose a VC at the input of the next router, an output VC. A VC allocator (VA) is used to match input VCs to output VCs. Based on the routing algorithm, there are different implementation of the VA. There are routing algorithms which allow the change of a VC while a packet traverses from a router to another (for example XY routing), while other may place specific restrictions on the use of VCs. The VC allocator must enforce all these rules during the VC allocation. When a flit has its own VC, it can arbitrate to use their output port. This step of arbitration is called Switch Allocation (SA) and

it is organized in two arbitration steps, a local step (SA1) and a global step (SA2). When a flit wins and has an output port, it will move to the crossbar (Switch Traversal ST) and then it will traverse the link (Link Traversal LT) to move to the next router.

2.4 Low-Power NoC Design

Reducing power consumption is a key design criterion for modern circuit designs, to extend battery lifetime, reduce packaging and cooling costs, and permit higher device performance. Modern semiconductor technologies that include FinFET transistors with increased input capacitances, significant wire RC parasitics, and 3D integrated structures for increased functionality on chip, make low power design a hard-to-achieve design goal. For maximum efficiency, even if low-power design starts at the architectural level, the low-power design techniques continue through the physical design implementation flow that creates, optimizes, and verifies the physical layout so that it meets the power budget along with timing, performance, and area goals.

Even until very recently, when product performance or power consumption suffered, designers conveniently migrated to the next generation manufacturing process to make chips with smaller, faster and more efficient transistors. Such approaches are longer viable in the post-Moore's law era where efficiency in power under strict performance requirements is the ultimate goal for any design.

Devices consume most of their power when they are operating. There are two major sources for the active power consumption: switching (dynamic) power and short-circuit power. Devices consume power also when they remain idle mostly due to transistor leakage currents. Normally, the active power consumption, especially switching power, is dominant one even if in areas of the chip with high inactivity leakage power can become a top concern.

NoC power consumption contributes around 10-15% of total chip power consumption. The power in a NoC is roughly spent in driving the long wires of the NoC, for buffering in-flight data, including also the required clock tree power for this operation, and for switching them from source

to destination. Several research efforts have tried to tackle in the past the design of low-power NoCs including architectural and circuit-level techniques and covering both dynamic and static power components. In the following paragraphs we present a detailed survey of most promising low-power alternatives targeting the design of scalable NoCs.

2.4.1 Power-aware Microarchitectures

Early works on NoC design explored the power efficiency of various NoC topologies and identified the power cost of NoC buffering, wire traversal and switching. In [2] detailed area and energy models for on-chip interconnection networks were developed and the tradeoffs in the design of efficient networks for tiled chip multiprocessors were identified for the first time. Using these detailed models, various aspects of the network architecture including topology, channel width, routing strategy, and buffer size are explored and their impact on performance, area and energy efficiency is quantified. The main first promising result identifies the use of multiple on-chip networks as a means to efficiently increase bandwidth and path diversity substantially without adversely affecting area or energy efficiency. Additionally, concentrated NoC topologies have been shown as a viable alternative for large-scale NoC designs.

The improvement of NoC router microarchitecture has been the focus of many research efforts with the goal to increase the energy efficiency of the designs without compromising performance (latency and throughout). The relation between performance and power consumption of various NoC and router alternatives have been initially compared and quantified in [4]. Speculative NoC routers that reduce the pipeline depth achieve the best performance.

Adopting the pipeline depth for optimal power-performance NoC behaviour has been explored in [86] and [100]. The router can adjust its pipeline depth (i.e., communication latency) and supply voltage level in response to the applied workload. Unlike dynamic voltage and frequency scaling (DVFS) routers, the operating frequency is the same for all routers throughout the CMP, which means that there is no need to synchronize neighboring routers working at different frequencies. The

router can select between two supply voltage levels and pipeline modes.

In [112], a token flow control, which allows bypassing of flit buffering in the routers, is proposed targeting to reduce the power consumption in both the control path (buffers) and the datapath (crossbar and links).

Datapath switching power has been tackled also in [141]. In this paper, a new power aware optimizations in the circuit and the micro-architecture of the routers is proposed, focusing on the design of a write-through buffer in the inputs of a router, which removes buffer read energy when bypassing can be done. Also, they used segmented crossbars, a simplified application of segmented busses.

Similarly, in [26] the router operation is splitted into two independent modules and each module handles traffic in one dimension (X- or Y-dimension). This allows simpler and smaller units for the arbiters and the crossbars, which are more power efficient.

In the same context, in [69] the cost of on-chip networks is reduced by partitioning the routers crossbar, and prioritizing packets in flight to simplify arbitration, and reducing the amount of buffers. A dimension-sliced router microarchitecture was used to minimize complexity, while intermediate buffers to decouple the x-dimension and the y-dimension of the dimension-sliced router are introduced. The amount of buffers where reduced to the absolute minimum thus negatively affecting performance. However, the use of prioritized allocation, favoring packets in flight, minimizes the loss of throughput.

Reducing the power contribution of buffers can be done by relying on bufferless flow control. Bufferless NoCs eliminate in-network buffers, such that flits contending for the same ports are either deflected towards other ports, or dropped awaiting retransmission by the upper layer protocol.

In [96] a novel buffer-less routing was used to eliminate the need for buffers for routing or flow-control in the network. In [42] a new bufferless router micro-architecture was proposed which supports a dropping mechanism to deal with routing conflicts. These techniques can reduce power consumption if the number of misroutes or dropped packets is small. Otherwise, the power consumption of the links is increased.

Also, they are fundamentally throughput limited as the congestion is increased quickly.

The work of [30] combines SMART flow control and bypass switching [14] with bufferless operation. SMART eliminates the latency dependence on hop count by using clock-less/asynchronous repeaters in routers' crossbar and associated flow to bypass multiple routers in the same cycle. Every router prioritizes requests using a fixed priority scheme based on the flit's distance from the source.

One additional effective way to reduce NoC power consumption is to reduce the amount of data sent over the network. To that extent, recent work has focused on compression at the cache and network levels [27, 62], as an effective power-reduction technique. Focusing on traffic reduction, the work in [67] seeks to reduce the amount of data transmitted through identification of useless words by transmitting only the flits that contain words predicted useful using a novel spatial locality predictor. To further lower NoC energy additional microarchitectural mechanisms are proposed that inhibit datapath switching activity for unused words in individual flits.

Reduced switching activity can be achieved via traditional clock gating of NoC components thus saving both datapath and clock tree power. In [97], clock gating optimisations are applied at two levels: (a) automated clock gating within router and (b) router Level Clock Gating exploiting the opportunities to isolates routers clock completely. Similarly, the work in [109] explored how the physical design flow reacts in NoCs buffer clock gating. Switching off unused components as enabled by clock gating can be enhanced at the network level where network links are turned off and switched back on depending on network utilization in a distributed fashion [133].

2.4.2 Wire Engineering

While the downscaling of device sizes led to continuous improvement in the properties of transistors, it caused significant degradation in properties of the metal wires that are used as system interconnects. Wires have become limiters of speed, power dissipation, and reliability because of their growing resistance and capacitance in scaled fabrication processes.

Due to nonuniform scaling of wire thickness and wire width, net-to-net cross- capacitance between adjacent wires constitutes the largest part of total interconnect capacitance. Line-to-line cross-capacitances within the same metal layer are important determinants of speed and power so that mutual effects between parallel adjacent wires must be considered during the physical design of the circuit layout.

Consequently, the spacing distance between wires on the chip has become a highly important resource, which deserves careful allocation and optimization [94]. The problem cannot be tackled alone but it should be combined wire sizing and bus repeater insertion for long links.

The optimization of wire placement in the physical layout reduces the capacitance seen by the drivers of the wires. The coupling capacitance shared of the total capacitance can be effectively reduced by appropriate encoding of the data transmitted on the wires. Encoding modifies the switching patterns of a group of wires in such a manner such that crosstalk coupling effect is reduced. A complete survey of bus encoding techniques can be found in [107].

Redundant bus coding has been used as an effective technique for trading off energy against reliability. Voltage on wires is reduced in order to save power. At the same time this effect decreases the reliability of wire transfers (noise can lead to transmission errors). The use of coding increases the hardware overhead but leads to safer transmissions at lower voltages (errors are corrected at the receiver) thus possibly increasing energy efficiency. The work of [6] explores this design tradeoff and evaluates the impact of two error recovery schemes when used for bus energy efficiency: correction at the receiver stage versus retransmission of corrupted data.

Low-swing signaling is now one well-known low-power design technique in both on- chip and off-chip interface circuits. In on-chip domain, the low-swing signaling will be considered as a natural choice for future on-chip communication fabrics since the wire performance benefit from CMOS process scaling does not keep up with the gate performance benefit. The low-swing technique is based on the dependence of dynamic energy on swing voltage. Reducing the voltage swing across data path leads to reduced charging and discharging of capaci-

tance on the wires thereby making on-chip communication fabrics more energy-efficient and faster. Particularly, under the circumstance where it is hardly possible to reduce length of wires and their fanout by using advanced processes or architectures, the low-swing signaling is the best design technique toward getting energy efficient on-chip networks [106, 152].

Another common technique is the charge sharing based low-swing drivers. The main idea is to limit power supply, but they need particular data patterns for reliable operation. The work in [45] uses a data-dependent logic swing bus, while in [74], a charge recycling technique is introduced. The half of the charge stored in the load capacitances is reused in every signal transition. The need to reduce supply voltage is removed by the use of cut-off drivers as proposed in [38].

2.4.3 Dynamic Voltage Frequency Scaling

Dynamic voltage and frequency scaling (DVFS) is a technique that takes advantage of the quadratic relationship between supply voltage and circuit power consumption to improve overall energy usage, making it a candidate for low-power VLSI design. There are tradeoffs to be made. Because circuits operating at a lower voltage will generally run more slowly, frequency of operation often needs to be scaled as the voltage reduces.

In order to apply dynamic voltage scaling, designers need to analyze how the system is managed dynamically and then find good control policies, which could be predictive, adaptive or a combination of both. The IC needs to be divided into domains that can be supplied with different clock signals and voltage levels. Control software and hardware are used to ensure each domain gets the right combination of voltage and clock signals. As with voltage islands, level shifters and synchronizers are needed to ensure that logic levels are adjusted correctly as the signals cross from one domain to another.

The work of [129] motivated the use of dynamic voltage scaling for links, where the frequency and voltage of links are dynamically adjusted to minimize power consumption. A history-based DVS policy that uses past network utilization to predict future traffic and tune link frequency

(and voltage) dynamically was proposed to minimize network power consumption while maintaining high performance.

Recent DVFS techniques for on-chip networks focuses on using some static network parameters like average queue utilization or average latency of memory requests replies, etc. to decide the new voltage-frequency (V-F) states of the routers. Typically, there is a DVFS controller, which is responsible for the following tasks: target a suitable network parameter and keep track of its values, based on previous states and target values compute a feedback and, finally, update V-F state.

For example, in [13] and [17] the main metric to tune the voltage and the frequency is the throughput of the network. In [13], a rate-based policy scales down voltage and frequency of the network to the minimum value that allows to sustain the injection rate without reaching saturation. Also, a target delay is set and a proportional-integral control loop is implemented. It measures the average delay and makes sure that the target is not exceeded

In [17] they proposed a central controller which tries to minimize the energy dissipation of both NoC and last-level caches (LLC) without degrade the performance of the whole chip in terms of total application time and the throughput. This work focuses on a realistic scenario where the entire NoC and LLC belong to a single V/F domain. As such, the interfacing overhead can be largely prevented and there is a coherent policy covering the whole of these shared resources. The throughput driven controller can use a dynamic reference point, while a new metric that bridges the gap between the NoC/LLC V/F level and the chip energy performance trade-off was introduced.

Other works, like in [8] and [92], target on the buffer utilization of the routers. The main goal of this idea is to determine the necessary operating frequencies such that the NoC queues reach and remain at their target reference values for bursty workloads. The runtime performance of application workloads can be used for the Voltage-Frequency assignment.

The work of [43] observes coherence messages to decide the state of the voltage and the frequency. Coherence messages usually follow more regular and predictable patterns than aggregate bandwidth in the net-

work, so they can be used to predict more accurately upcoming NoC bandwidth requirements. The traffic predictions are then used to infer aggregate bandwidth demands in the network that enable informed DVFS decisions. Similarly, in [Malleable NoC: Dark Silicon Inspired Adaptable] the L1 and L2 cache misses are taken into account in order to determine the values of voltage and frequency per router.

[148] presents a voting-based approach where the threads utilizing the NoC seek to influence the DVFS decisions independently by voting for a preferred V/F level that best suits their own performance requirements based on their runtime profiled message generation rate and data sharing characteristics. Then, the vote is carried in the packet header and spread to the routers on the packet route. A region DVFS controller takes the final DVFS decision democratically according the majority of collected votes from all active threads. To achieve scalable V/F adjustment, each region works independently, and the voting-based V/F tuning forms a distributed decision making process.

In [149] the per-router voltage/frequency tuning is done using the memory-access density information. Also, a priority-based switch allocation is performed to speed up critical packets and avoid starvation using the memory-criticality information.

2.4.4 Power Gating

Although the equation for switching power in CMOS circuits $P = kCfV^2$ suggests that reducing voltage as far as possible will minimize energy pointing to operation close to or below the threshold voltage the effect of leakage needs to be taken into account. In nanometer processes, the effects of leakage are severe and can easily exceed the benefits of cutting switching power through voltage and frequency reductions resulting in higher energy overall even though the instantaneous power demand will generally be lower.

The effects of leakage have made DVFS less attractive as a power-management and energy-reduction strategy in leading-edge processes than for older technologies that can offer better leakage control. A common alternative strategy is to run the circuit at full speed but use power gating to cut the supply when it has completed.

In mane applications, there are certain blocks of the chip which do not operate during different working modes like sleep or stand-by modes and only a part of the device is required to function. In these case, designers should be able to power off non-functional blocks so that they can reduce the power consumption of the unused blocks. Thus, the goal of power gating is to minimize leakage power by temporarily cutting power off to selective blocks that are not required in that mode.

At first, in [132] proposed a power gating mechanism, which checks the interconnect link utilization. Based on that information, it later decides to power gate components of the NoC such as ports of routers and links in response to bursts in network traffic. Although it shows promising results, it has some drawbacks. First, this approach reduces the performance remarkably in cases with big wake-up latency or unpredictable traffic. Also, it misses opportunities for power gating idle router which are attached to sleep cores.

The work of [16], implements a power-gating bypass mechanism that decouples the nodes ability for sending, receiving and forwarding packets from the powered-on/off status of the associated router. To ensure this, they provide a decoupling bypass path that connect the inject and the eject links for a bypass channel to the router. This idea reduces the number of states transition and increases the number of cycles a router that can remain idle. Also, it removes any disconnection problems between routers and it hides the wake-up latency from the critical path when a router moves from sleep state to normal working state. However, a bypass ring is not scalable to big networks.

PowerPunch [15] introduced a performance-aware, non-blocking power-gating mechanism which turn on routers from switch-off state along the path of packet in advance thereby preventing packet from suffering router wakeup latency or detour latency. The main idea is to send power control signal before the packet to "punch through" any blocked routers in power-gated mode.

In [120], the main idea is to power gate a some of routers associated with sleeping cores by proactively aggregating traffic to the active routers such that it conserves power, does not impact function, and minimizes the impact on performance. At the same time, the proposed method

makes sure that the connectivity between the cores of the network is maintained, and limits the average interconnect latency impact of packet detouring around inactive routers. For that reason, a centralized manager is responsible for configuring, monitoring and re-configuring the NoC.

Catnap [28] proposed a multiple-network design with synergistic power-gating policies. The main idea is that a multiple-network design is more amenable to power gating, as its subnetworks (subnets) can be power gated without compromising the connectivity of the network. However, it is only applicable to CMPs with high-bandwidth requirements

Finally, in [104], a power-aware routing and topology reconfiguration of the NoC is proposed. The goal is to minimize detours while selected components in routers are power-gated. The routing reconfiguration uses a distributed mechanism and guarantees that deadlock-free routes are available at all times. At runtime, its power-gating decisions should follow the applications communication patterns. This feedback-based mechanism is slow and reconfiguration takes place only on per epoch basis. Power-gating components inside the router in a fine grained fashion requires additional circuitry.

Chapter 3

ElastiStore: Low-Cost Virtual-Channel Buffers

The NoC needs to be both scalable, in terms of network functionality and performance, and flexible, in terms of physical implementation. This requirement motivates us to unify a VC-based architecture, which favors NoC scalability, with Elastic Buffering (EB), which eases physical implementation and promises area and power reduction.

Owing to its elastic operation – based on simple ready/valid handshakes – elastic buffering is a primitive and simplified form of NoC buffering, which can be easily integrated in a plug-and-play manner at the inputs and outputs of the routers (or inside them) [89, 12, 116], as well as on the network links to act as a buffered repeater [20]. Elastic buffering assumes only one form of handshake on each network channel. The handshake cannot distinguish between different flows, thus making the elastic buffering operation serial in nature. This feature prevents the interleaving of packets and the isolation of traffic flows, while it complicates deadlock prevention. Due to this limitation, direct support for VCs is abandoned and replaced by multiple physical networks, or implemented via complex and non-scalable hybrid EB/VC buffering architectures [90, 39, 72]. However, the latter techniques remove the basic property of the elastic buffers to act as stitching elements that can be placed seamlessly anywhere in the NoC.

In this work, we aim to address the aforementioned deficiencies of EB-based designs, by generalizing the operation of elastic buffering to

support multiple VCs, while at the same time retaining all scalability and composability attributes. The proposed architecture, which we call *ElastiStore*, minimizes the number of buffers per channel close to the absolute minimum of one buffer slot per VC, without sacrificing performance and without introducing any dependencies between VCs, thus ensuring deadlock-free operation. The operation of ElastiStore is generalized to support arbitrary round-trip latencies.

The elastic operation and minimum buffering are maintained, while the extra buffering required due to the increased round-trip latency is absorbed via a low-cost shared (across VCs) buffer structure inside ElastiStore.

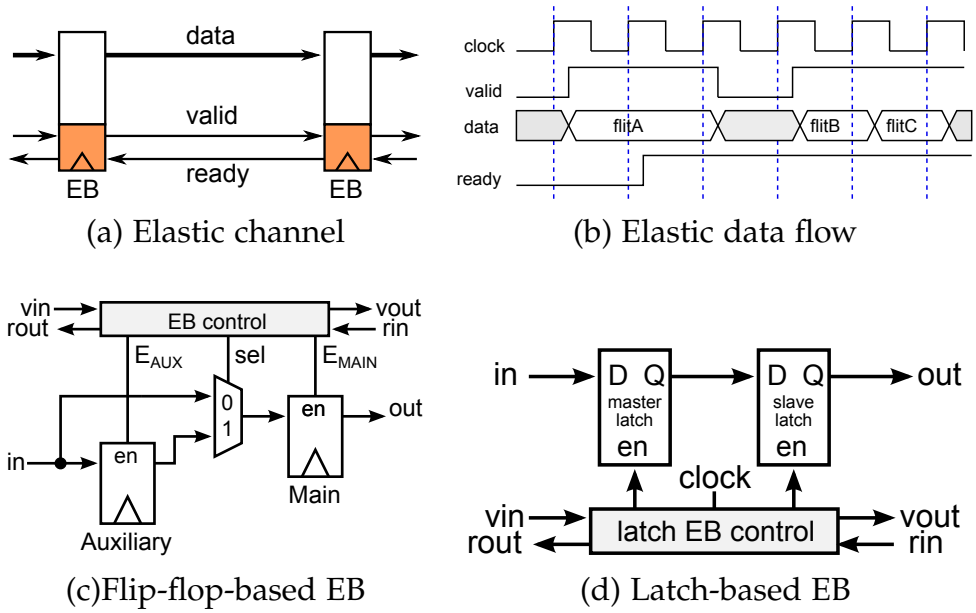
The scalability of the proposed scheme is demonstrated by the integration of ElastiStore in both single-cycle and pipelined NoC routers that offer the same performance as baseline VC-based routers, albeit at a significantly lower area cost. The experimental results – based on both synthetic traffic patterns and real application workloads running in a full-system simulation framework – validate the effectiveness of the proposed architecture. Additionally, hardware implementation results corroborate our claims pertaining to ElastiStore's efficiency. Overall, our evaluation demonstrates that ElastiStore enables the design of extremely low-cost and highly scalable VC-based NoC architectures that provide equal networking performance as much more expensive (in terms of area/power) state-of-the-art VC-based implementations. ElastiStore is envisioned as an archetypical primitive for future, extremely low-cost NoC router implementations, where the performance and functionality enhancements provided by VCs cannot be sacrificed.

3.1 Elastic Channels and Buffers

A single-lane elastic channel carries – in parallel to the data wires – two extra control wires (valid and ready), which are required to implement the elastic protocol, as shown in Figure 3.1(a). The EBs implement the elastic protocol by replacing any simple data connection with an elastic channel. When an EB can accept an input, it asserts its ready signal upstream; when it has an available output, it asserts the valid signal downstream. When two adjacent EBs both see that the valid and

ready signals are both true, they independently know the transfer has occurred, without negotiation or acknowledgement. An example of this handshake is shown in Figure 3.1(b).

When the output of a chain of EBs stalls, the stall can only propagate back one stage per cycle. To handle this, all EBs can hold two words, one for the stalled output, and one caught when necessary from the previous stage. Such an implementation is shown in Figure 3.1(c). By controlling the clock phases accordingly, as shown in [22], the 2-slot EB can also be designed using 2 latches in series, instead of two flip-flops, similar to Figure 3.1(d). Following the same methodology, any EB architecture derived for edge-triggered flip-flops can also be implemented with latches.



(a) Elastic channel (b) Elastic data flow

(c)Flip-flop-based EB (d) Latch-based EB

Figure 3.1: The fundamentals of the elastic buffering protocol. The protocol requires two control wires (valid and ready), which typically run in parallel to the data wires. The data and control wires together comprise a single-lane elastic channel. Any EB architecture derived for edge-triggered flip-flops can also be implemented with latches.

The same handshake signals can be used for deriving an inelastic flow-control policy. When elasticity is removed and the end of a pipeline of flow-controlled registers is stalled, data movement stops concurrently for all stages of the pipeline; data flow is simply frozen, and, inevitably, some pipeline stages remain empty. On the contrary, an elastic flow-control policy allows all empty stages to be filled with incoming data.

In NoCs, the flits of the packet need to flow as close as possible to their final destination before being stalled for any reason. Therefore, the implementation of any flow-control policy in NoCs should be inherently elastic.

3.2 VC Flow Control and Buffering

Baseline elastic flow control is serial in nature (FIFO-like). Thus, it is not possible to support different isolated flows analogous to a multilane street, or even to allow the interleaving of flits from different lanes on the same elastic channel. This can only be supported by employing individual handshaking interfaces for each supported VC, so that the various VC traffic flows are inherently logically separated and easily guided to their respective parallel buffer slots.

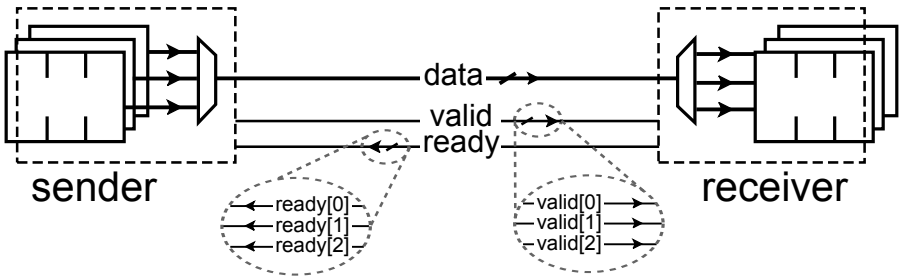


Figure 3.2: Virtual channels require the addition of separate buffers for each VC at the receivers side and at the same time call for enhancements to the flow control signaling to accommodate the multiple and independent flows travelling in each VC.

To divide a physical channel into V virtual channels, the input queue at the receiver needs to be separated into as many independent queues as the number of virtual channels. These virtual channels maintain control information that is computed only once per packet. To support the multiple independent queues link-level flow control is also augmented and includes separate information per virtual channel.

Ready/valid handshake on each network channel cannot distinguish between different flows. This feature prevents the interleaving of packets and the isolation of traffic flows, while it complicates deadlock prevention. A channel that supports VCs consists of a set of data wires that transfer one flit per clock cycle, and as many pairs of control wires

valid(i)/ready(i) as the number of VCs. Fig. 3.2 shows an example of a 3-VC elastic channel. Although multiple VCs may be active at the sender, flits from only one VC can be sent per clock cycle; only one valid(i) signal is asserted per cycle. The selection of the flit that will traverse the link requires some form or arbitration that will select one VC from those that hold valid flits. At the same time, the receiver may be ready to accept flits that can potentially belong to any VC. Therefore, there is no limitation on how many ready(j) signals can be asserted per cycle.

The basic property of VC-based flow control is the interleaving of flits of different packets. In each cycle, a flit from a different VC can be selected and appear on the link. The flit once it arrives at the receiver is placed on the appropriate VC buffer. Since the buffering resources of each VC at the receivers side are completely separated, interleaving of flits of different packets does not create any problems, assuming that the VC-based flow control mechanism does not involve any dependencies across VC, e.g., if the buffer of a certain VC is full to stop the transmission of flits from another VC.

In the simplest form of single-cycle links the valid and the backpressure information needs one cycle to propagate in the forward and in the backward direction. Therefore, as each VC needs 2 slots to enable lossless operation and 100% throughput.

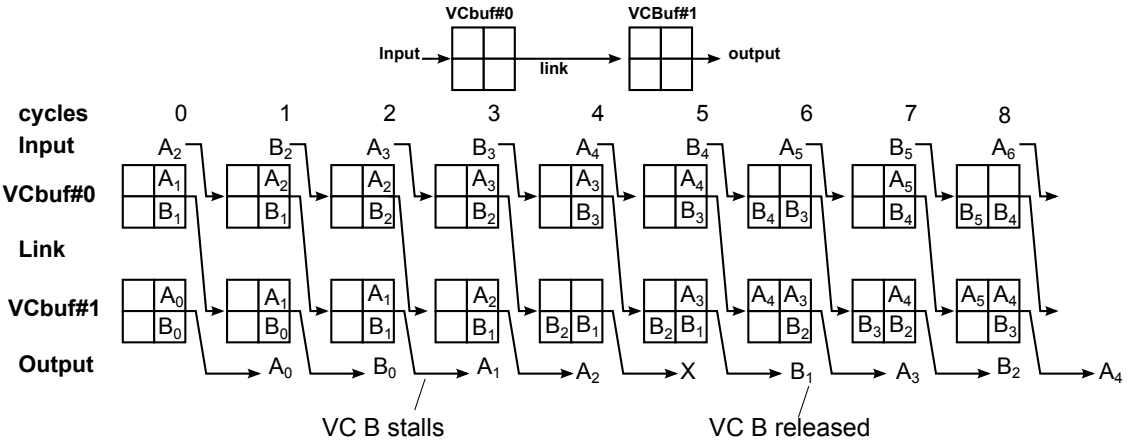


Figure 3.3: Flit flow on a channel between two primitive VC buffers that employ a 2-slot EB for each VC.

Fig. 3.3 depicts a running example of a VC-based pipeline using a 2-slot

queue per VC. The two active VCs each receive a throughput of 50% and each VC uses only one buffer out of the two available per VC. The second buffer is only used when a VC stalls. This uniform utilization of the channel among different VCs leads to high buffer underutilization. The buffer underutilization gets worse when the number of VCs increase. In the case of V active VCs, although the physical channel will be fully utilized, each VC will receive a throughput of 1/V and use only one of the two available buffer slots since it is accessed once every V cycles. Only under extreme congestion will one see the majority of the second buffers of each VC occupied. However, even under this condition, a single active VC is allowed to stop and resume transmission at a full rate independently from the rest VCs. This feature is indeed useful in the case of traffic originating only from a single VC, where any extra cycles spent per link will severely increase the overall latency of the packet. However, in the case of multiple active VCs, whereby each one receives only a portion of the overall throughput (1/M in the case of M active VCs), allocating more than 1 buffer slot per VC is an overkill.

3.3 VC Buffering on Pipelined Links

When the delay of the link exceeds the preferred clock cycle, one needs to segment the link into smaller parts by inserting an appropriate number of pipeline stages. In the case of single-lane channels, the role of the pipeline stages is covered by EBs, which isolate the timing paths (all output signals – data, valid, and ready – are first registered before being propagated in the forward or in the backward direction), while still maintaining link-level flow control.

In the case of VC-based elastic channels, we can rely on simple registers for pipelining the data and the ready/valid handshakes signals on the link, as shown in Figure 3.4. In this case, the flits cannot stop in the middle of the link, since the pipeline registers do not employ any flow control. Many words may be in-flight, since it takes L_f cycles for the signals to propagate in the forward direction and L_b cycles in the backward direction. Therefore, the buffers at the receiver need to be sized appropriately to guarantee lossless and full throughput operation, i.e., more buffers per VC are needed than the 2 slots per VC allocated in the

case of a single-cycle channel.

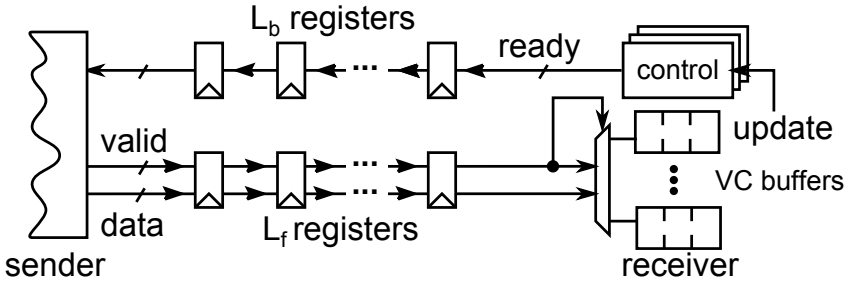


Figure 3.4: Abstract model of a pipelined link with multiple VCs and independent ready/-valid handshake signals per VC.

First of all, assume that only one VC, i.e., the ith one, is active for a period of time and the remaining VCs do not send or receive any data. When the buffer of the ith VC is empty, it asserts the ready(i) signal. The sender will observe that ready(i) is asserted after L_b cycles and immediately starts to send new data to that VC. The first flit will arrive at the receiver after $L_f + L_b$ cycles. This is the first time that the receiver can react by possibly de-asserting the ready(i) signal. If this is done, i.e., ready(i)=0, then under the worst-case assumption, the receiver should be able to accept the $L_f - 1$ flits that are already on the link, plus the L_b flits that may arrive in the next cycles (the sender will be notified to stop with a delay of L_b cycles). Thus, when the ith VC stalls, it should have at least $L_f + L_b$ empty buffers to ensure lossless operation. Actually, the minimum number of buffers for the ith VC reduces to $L_f + L_b - 1$, if we assume that the sender stops transmission in the same cycle it observes that ready(i)=0.

Thus, a channel with V VCs and a round-trip time of $L_f + L_b$ needs at least $V(L_f + L_b - 1)$ slots. When many VCs are active on the channel, their flits would be interleaved and the probability that all $L_f + L_b - 1$ flits belong to the same VC is small. However, the worst-case condition calls for providing as much buffer space to each VC as needed to prevent the dropping of any flit, independent of the traffic conditions on the remaning VCs.

Unfortunately, giving the minimum number of buffers to each VC has some throughput limitations. Assume that the ith VC has occupied all its buffer slots at the receiver and starts draining the stored flits down-

stream at a rate of one flit per cycle. After $L_f + L_b - 1$ cycles, the buffer will be empty (no more flits to drain) and the ready(i) signal will be asserted, causing the fist new flit to arrive $L_f + L_b - 1$ cycles later (the ready(i) signal is asserted in the same cycle that the last flit is drained). Therefore, in a time frame of $2(L_f + L_b - 1)$ cycles, the receiver was able to drain only $L_f + L_b - 1$ flits, which translates to 50% throughput. Thus, a single active VC can enjoy 100% throughput when it has $2(L_f + L_b - 1)$ buffers and is ready when the number of empty slots is at least $L_f + L_b - 1$. The baseline VC-based EB employed in single-cycle links ($L_f = L_b = 1$) is a sub-case of the general pipelined link and achieves 100% throughput of lossless operation using 2 buffers per VC.

The scenario of using simple EBs on the link instead of pipeline registers in the case of VC flow control will work, but only after introducing dependencies across VCs, since the flow control information per VC needs to be serialized under a common ready/valid handshake; if one VC stops being ready, all the words on the link should stop, irrespective of the VC they belong to, as done in [72]. Such dependencies ruin the isolation and deadlock-freedom properties of the VCs and require ad-hoc modifications to the flow control mechanism, even if sufficient private buffer space is allocated per VC.

3.4 The ElastiStore Buffer Architecture

ElastiStore represents the simplest form of a VC-based buffer structure that can be used either as a distributed buffering alternative, or at the inputs of VC-based routers, as will be shown in Section 3.5. When the flow-control signals between two ElastiStore units take more cycles to propagate in the forward or in the backward direction ElastiStore uses shared buffer positions to absorb the in-flight flits.

As in the case of single-cycle links, in pipelined links with L_f and L_b of forward and backward latencies, respectively, we minimize buffering by employing sharing and by exploiting the fact that only a single VC (dynamically selected) can enjoy 100% throughput when it is the only active VC in the system.

Instead of having $2(L_f + L_b - 1)$ buffer slots for each VC, we dedicate

$L_f + L_b - 1$ slots per VC needed for safe operation (called the main buffers) and $L_f + L_b - 1$ more, which can be dynamically shared by all VCs (called the shared buffer). Any VC is ready, as long as there are $L_f + L_b - 1$ empty slots either in its main register alone, or accounting for the free space in the shared buffer as well. Therefore, a single active VC can enjoy 100% throughput, while, in the case where the shared buffer is full, every active VC cannot get more than 50% of throughput (it can receive/send $L_f + L_b - 1$ flits at most every $2(L_f + L_b - 1)$ cycles); when many VCs are active, the throughput per VC is always much lower than 50%.

Under high utilization, the channel is already shared by many VCs, and achieving high-throughput per independent VC does not give much benefit, unless it is the only active VC. Therefore, our optimal goal is to offer full throughput to a single-active VC by using just one register per VC of private buffering, as well as $L_f + L_b - 1$ more positions shared by all VCs. If we try to achieve this goal with the current flow-control semantics, dependencies across VCs may appear that can possibly lead to a deadlock. Assume, for example, that the ith VC uses its main buffer (1 register) and at least one slot from the shared buffer, leaving less than $L_f + L_b - 1$ free slots in the shared buffer. Then, every other VC must de-assert its ready signal, even if its main register is empty, since the available free slots for each VC are less than $L_f + L_b - 1$, which are needed to guarantee safe operation per VC. Under this scenario, the traffic on one VC is allowed to block the traffic on another VC, which removes the needed isolation property across VCs.

3.4.1 Flow Control

To achieve our goal of minimum buffering, we should change the flow control mechanism and allow the sender to explicitly store the condition of each downstream VC, as done by credit-based flow control. The sender keeps a free slot counter, or else called a credit counter, for each downstream VC, and a counter for the shared buffer that counts the available shared buffer slots. A VC is eligible to send a new flit when there is at least one free position (either at the main register or the shared buffer). Once the flit is sent from the ith VC, it decrements the credit counter of the ith VC. If the credit counter of the ith VC was already

equal to or smaller than zero, this means that the flit consumed a free slot of the shared buffer and the counter of the shared buffer is also decremented.

Since the state of each VC is kept at the sender, the receiver only needs to send backwards a credit-update signal, including a VC ID, which indexes the VC that has one more available credit for the next cycle. On a credit update that refers to the jth VC, the corresponding credit counter is increased. If the credit counter is still smaller than zero, this means that this update refers to the shared buffer. Thus the credit counter of the shared buffer is also increased. Please note that even if there is a separate credit counter for the shared buffer the forward valid signals and the credit updates refer only to the VCs of the channel and no separate flow control information is needed for the shared buffer.

In this case, safe operation is guaranteed even if there is only 1 empty slot per VC (main register), but with very limited throughput due to the increased round-trip time; no flit can be in flight if it has not consumed a credit beforehand. A single active VC can utilize both its main register and all the positions of the shared buffer and achieve 100% throughput, by effectively allowing this VC to use $L_f + L_b$ buffers in total.With credits, once a credit update is sent backwards for a VC it means that a new flit will arrive for this VC after $L_f + L_b - 1$ cycles. Therefore, offering to a single VC $L_f + L_b - 1$ buffers, means that at the time the last flit is drained from the VC the first new flit will arrive thus leaving no gaps in the transmission and offering full throughput. Note again that *any VC is still eligible to accept a new flit in its private (main) buffer*, irrespective of the state of the shared buffer, thus *completely avoiding deadlock conditions*.

By adopting credit-based flow control, the ElastiStore minimizes the private buffer space per VC, thus achieving its main goal of minimizing the buffer space to just one register per VC, and some extra shared space to fully cover the round-trip time for one VC.

The use of credits, or ready(i)/valid(i) handshake signals, for the flow control of different VCs are similar, in the sense that they both count either implicitly, or explicitly, the available number of buffer slots for each VC. For the ready(i)/valid(i) interfaces, this counting is done at the receiver, and the sender is notified via the delayed ready signals. On

the contrary, in credit-based flow control, buffer availability is checked locally at the sender, without any notification delay. This difference in the notification delay causes the two flow-control policies to behave differently, in terms of minimum buffering requirements for achieving full throughput operation.

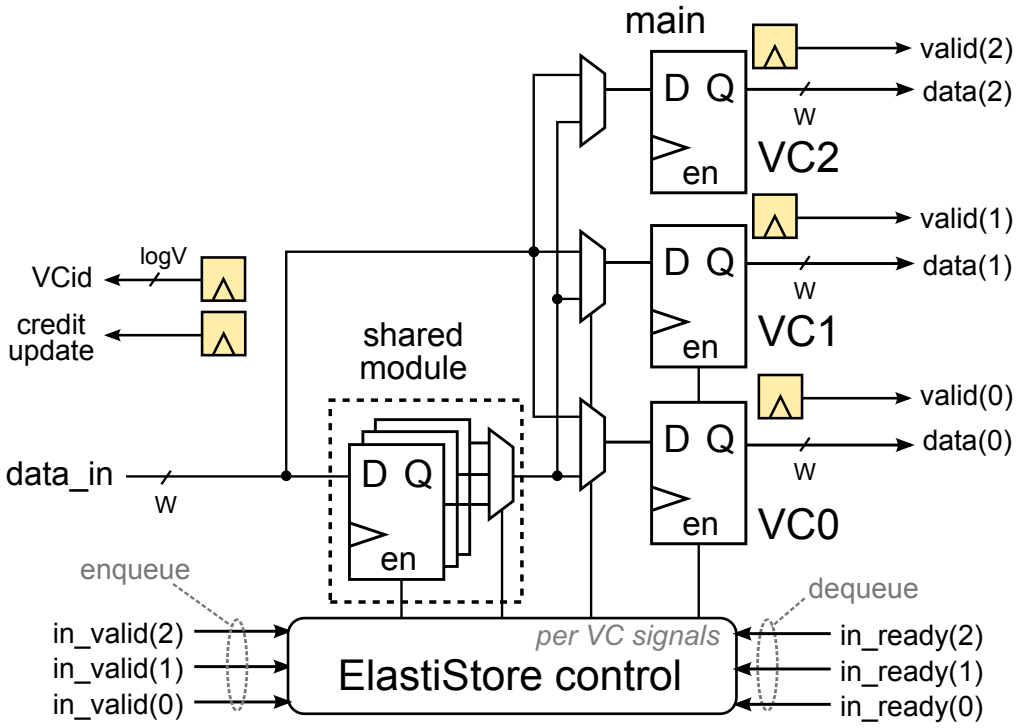


Figure 3.5: The organization of ElastiStore. The shared buffer consists of as many buffer slots as required to cover the round-trip time of the flow-control signals

3.4.2 Hardware Implementation

The implementation of the generalized ElastiStore (which is able to support arbitrary round-trip times) is based on the **three basic operational principles/rules** that characterize all ElastiStore designs and differentiate them from other state-of-the-art buffer implementations:

1. Each VC has only 1 slot of private buffer space implemented via a main register per VC, while the remaining buffer space is shared and used together with the one main register per VC to cover the round-trip time of the channel that ElastiStore is connected to (this translates to 1 shared buffer for single-cycle links and $L_f + L_b - 1$ buffer slots for a link with L_f forward and L_b backward laten-

cies, respectively). This configuration offers the minimum possible buffering, with the constraint that at most one VC can receive full throughput. This is not a restrictive choice, since a VC will get full throughput only when it is the only active VC. In all other cases, each VC will receive an equal share of the available throughput.

2. Any allocation decision regarding which VC should dequeue a flit from the ElastiStore structure is taken based only on the status of the main registers. In this way, request generation can begin without any overhead associated with checking the head-of-line flits of many VC queues hosted in a shared buffer space.

3. When a main register sends a flit downstream and gets empty, it is refilled in the same cycle, either with a flit possibly present in the shared buffer, or directly from the input, assuming the new flit belongs to the same VC. This design principle completes the previous one and avoids idle cycles in the data flow. With this type of refill, ElastiStore actually mimics the simple elastic buffer of Fig. 3.1(c), which refills (in the same cycle) the main register using the data of the auxiliary register, or the data of the input, when the main register gets empty, in order to offer full throughput of data transfer.

The introduced design principles and the use of negative credits – that simplify the process of credit identification, i.e., which credits belong to the main registers, and which ones belong to the shared buffer space – offer an overall simplified buffering architecture.

The datapath that implements ElastiStore is illustrated in Figure 3.5. The multiplexers at the input of the main registers allow new data to come directly from the ElastiStore's input, or the shared buffer. If the main registers cannot accommodate an incoming flit, a shared slot is allocated, where the flit is stored. As soon as the main register becomes available again, the flit is retrieved from the shared buffer and moves to the corresponding main register.

Every time a VC dequeues a flit from its main register, it should check the shared buffer for another flit that belongs to the same VC. Figure 3.6 demonstrates the interaction between the shared buffer and the main VC registers through a simple example. In cycle 0, VC A owns 2 shared

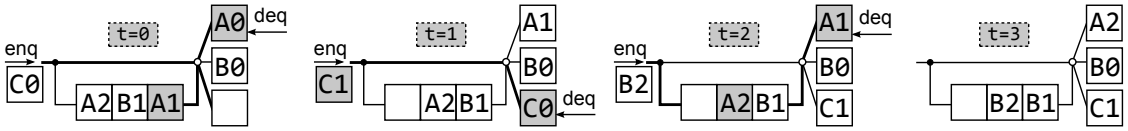


Figure 3.6: An example of the interaction between the shared buffer and the main VC registers in the generalized ElastiStore architecture. Every time a VC dequeues a flit from its main register, it should check the shared buffer for another flit that belongs to the same VC.

slots and dequeues a flit from its main register. The empty main register should be refilled in the same cycle in order to avoid any bubbles in the flit flow control. Therefore, VC A initiates a search on the shared contents to find the flits that match VC A and locate the oldest (the one that came first). The refill of the main register of VC A is completed in cycle 1. Then, in cycle 2, the same procedure is followed, effectively loading the main register of VC A with a new flit. The main register of a VC does not necessarily get new data from the shared buffer, but it can be loaded directly from the input, as done for VC C in cycle 1.

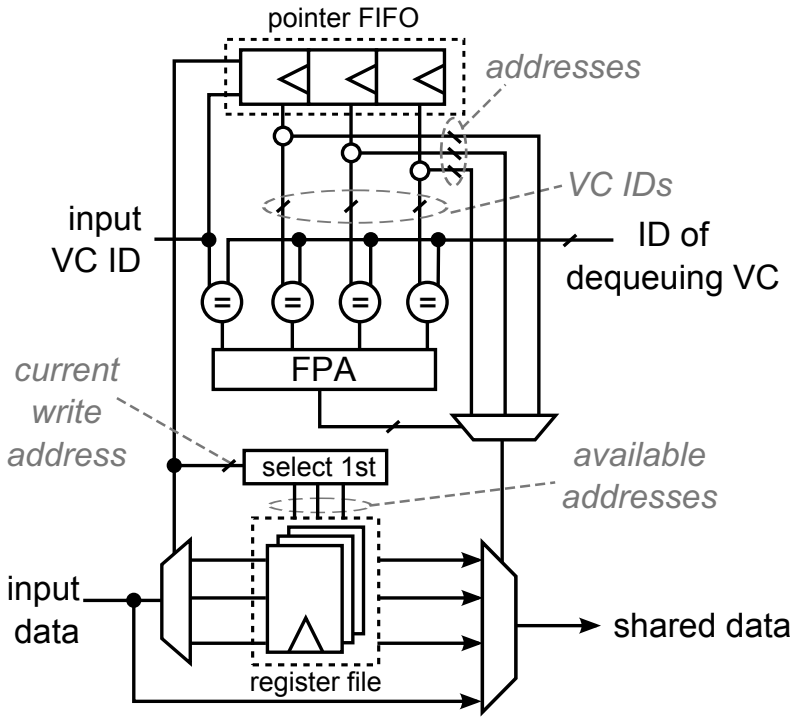


Figure 3.7: The block diagram of the *shared buffer* of the ElastiStore architecture. The shared buffer consists of two main storage modules. The first one (top part) is a shift-register-based FIFO that stores (in each slot) the VC ID of a flit and a pointer. The second storage module (bottom part) is the register file that stores the actual flit contents.

The shared buffer within ElastiStore should preserve a FIFO order, not in terms of the whole buffer, but separately for the flits of each VC. In the example of Figure 3.6, the flits' order of arrival is preserved by shifting them forward every time a slot in the shared buffer is emptied. However, this is simply an abstract representation of what is really happening: flits are not physically shifted, but, instead, their pointers change. In other words, the FIFO order is maintained through the pointer logic. The block diagram of the shared buffer of the ElastiStore architecture is depicted in Figure 3.7. It consists of two main storage modules. The first one (top part of Figure 3.7) is a shift-register-based FIFO that stores (in each slot) the VC ID of a flit and a pointer. The second storage module (bottom part of Figure 3.7) is the register file that stores the actual flit contents. When a flit is pushed to the shared buffer, the actual contents are written in the "register file", while the write-address and the VC ID of the incoming flit are pushed into the first empty slot of the shift register (the "pointer FIFO" in Figure 3.7).

The proposed shared buffer does not participate in router allocation, as imposed by the second design rule and, thus, it is redundant to be able to "see" the head-of-line flits of all hosted VC queues (this option would need at least one read pointer for each active VC queue). On a read (dequeue operation), the shared buffer should be able to read out the first appearance of the dequeuing VC ID, in order to refill its main register. This is done by comparing the IDs stored in the "pointer FIFO," or the input, with the ID of the dequeuing VC, and selecting – using a Fixed-Priority Arbiter (FPA) – the first one that matches. Then, the pointer stored in the "pointer FIFO" is used to retrieve the actual flit from the "register file," while the information of the dequeued flit is removed from the "pointer FIFO," thereby causing all subsequent pointers to shift forward. At the same time, the address of the dequeued flit is marked as available in the "register file" and can be reallocated to any VC.

3.5 Integration of ElastiStore in NoC Routers

ElastiStore can be integrated at the inputs and at the outputs of a router, as illustrated in Figure 3.8. When ElastiStores are used in both inputs

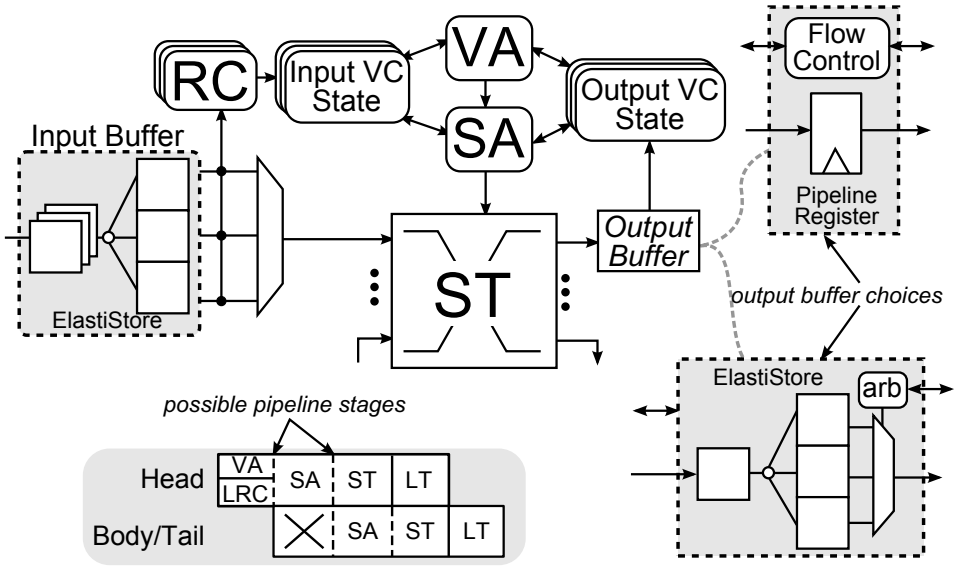


Figure 3.8: The integration of ElastiStore in NoC routers. ElastiStore modules can be integrated at the inputs and at the outputs of a router. In general, ElastiStores can be placed seamlessly and in a plug-and-play manner everywhere within the NoC.

and outputs, the output VCs refer to the buffers of the output ElastiStore and not to the VC buffers at the input of the next router. The same also holds for flow control information that reflects the buffer status of the output ElastiStore of the same router. A flit is ready to move to the output ElastiStore when the corresponding VC of the output ElastiStore is ready. The placement of an output ElastiStore actually breaks the flow-control cycle (which determines the round-trip time) between two neighboring routers in two parts: (1) the intra-router part, which involves the forward and backward latencies inside the router, and (2) the inter-router part, which only involves the flow-control latencies of the link. The flow control on the links does not allow packets to change VC, and its operation only needs an arbiter and a multiplexer for selecting a flit to send to the next router.

In single-cycle routers, where one cycle is spent inside the router and one on the link, the intra-router and inter-router round-trip times are equal to two cycles. This configuration enables the use of ElastiStores at the inputs and the outputs, which have only one register per VC, plus one shared slot among all VCs. In pipelined configurations, however, the intra-router round-trip time increases to more than two cycles, following the inevitable increase of the packet forward latency. In this case,

and assuming again that ElastiStores are placed both at the inputs and at the outputs, the input ElastiStore should have one register per VC plus an m-slot shared buffer (see Figure 3.5). When a VC is selected by SA, its main register is dequeued and a refill possibly occurs from the shared buffer. The refill data can be prepared beforehand, just after SA1, thus overlapping the search in the shared buffer with SA2, while actual dequeue (pointer movement) happens only if SA2 is also successful for the same VC. Since the round-trip time across the link remains the same, the output ElastiStore can still be the simplest one.

Baseline routers utilize a simple pipeline register – instead of the output ElastiStores – that isolates the inter-router timing paths from link traversal. Therefore, the round-trip times expand inevitably between the inputs of two neighboring routers, which are the only flow-controlled buffering points. In single-cycle baseline routers, this translates to 3 buffers per VC to cover the round-trip time, while a pipelined router with two stages increments the credit round-trip latency by one more cycle, thus needing a minimum of 4 buffers per VC. This analysis assumes that credit updates across routers need at least one cycle to propagate. This extra cycle can be removed if flow-control information is transferred across routers via direct combinational paths. However, this actually limits the benefits of pipelining, and the increased link delay directly affects the speed of the router.

In every case, the proposed designs save considerable amount of buffers, which directly translate to significant area savings without any network performance loss, as it will be shown in the next section. When using ElastiStore only at the inputs, one may reach the absolute minimum of VC buffering of one register per VC (needed also for deadlock freedom), plus any additional buffer slots needed for covering the round-trip time and offering the privilege to a single active VC to achieve 100% throughput. The use of ElastiStores at the output of the routers "steals" some time from link traversal, due to the arbitration and multiplexing operation. However, the extra delay added to the delay of the link will affect the NoC clock frequency only in the case of very long wires. In such cases, simple pipeline registers can still be used at the outputs, as shown in Figure 3.8.

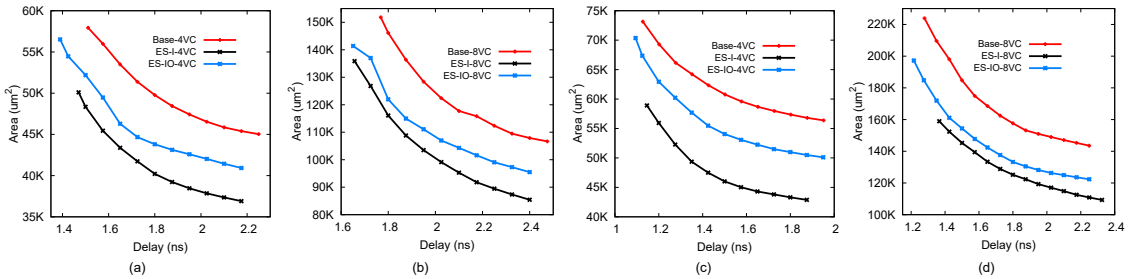


Figure 3.9: Hardware implementation results of various designs with (a)-(b) single-stage, and (c)-(d) two-stage router designs, using an industrial low-power 45 nm standard-cell library.

3.6 Evaluation

In this section, we compare ElastiStore-based routers with conventional VC-based routers, both in terms of hardware complexity and network performance, which includes synthetic traffic patterns (Section 3.6.2), as well as real application workloads (Section 3.6.3).

3.6.1 Hardware Implementation

The routers under comparison (using lookahead RC) were synthesized to an industrial low-power 45 nm standard-cell library under worst-case conditions (0.8 V, 125 °C), and placed-and-routed using the Cadence digital implementation flow. The generic router models have been configured to 5 input-output ports, as needed by a 2D mesh network, and to 4 and 8 VCs per port, while the flit width was set to 64 bits. The area/delay curves, shown in Figure 3.9, were obtained for all designs after constraining appropriately the logic-synthesis and back-end tools and assuming that each output is loaded with a wire of 2 mm.

The routers under comparison cover baseline implementations with one or two pipeline stages, as well as ElastiStore-based routers that include buffers both at the inputs and the outputs (ES-IO) and only at the inputs (ES-I). Baseline VC routers can be built with shallow, or deep buffers per VC. It is critical, however, for each VC to contain as many buffers as needed to cover the credit round-trip latency. For all the single-cycle routers, we employed the combined allocation approach presented in [82], which offers the same network performance as traditional allocation organizations, but with significantly better achievable clock fre-

quency. On the contrary, the two-stage pipelined routers follow the normal allocation strategy, where SA begins only after VA has been completed for an arriving packet.

In all cases, the ElastiStore-based routers offer significant area savings, up to 18% and 24% for 4 and 8 VCs respectively, without any delay overhead. This behavior is the result of the reduced number of buffer slots required by ElastiStore and the overall simplicity of its control logic (less than 10% of the total ElastiStore area). The latter is a consequence of the three newly introduced design principles/rules, and the simplified credit-handling protocol. The ES-I configuration, as expected, is the most area-efficient solution. The ES-IO setup, which completely isolates the inter-router flow control mechanism from the intra-router one, achieves even faster designs, since the readiness of each VC is directly provided by the ready/valid handshake signals, while still saving area relative to the baseline design. Note that the delay numbers reported correspond to operation at 0.8 V. At this low voltage, the clock frequency of even ultra-fast 3-stage commercial routers is below or marginally pass 1GHz [119, 50].

Table 3.1: The energy per cycle (in pJ) required for baseline and ElastiStore-based routers having 4 and 8-VCs, and operating in single-cycle or 2-stage pipelined configurations.

Buffers	4 VCs		8 VCs	
	1-stage	2-stage	1-stage	2-stage
ES-I	67	69	108	122
Base	87	91	153	167

The hardware complexity analysis is completed by reporting the energy behavior of the routers under comparison. Energy (or area) comparisons are meaningful when the compared circuits are optimized for the same delay target. Therefore, based on the delay profile reported in Figure 3.9 for single-cycle and 2-stage pipelined solutions, we select the designs that correspond to a delay of 1.5 ns and 1.8 ns for the case of 4 and 8 VCs, respectively. The energy consumed for each case is reported in Table 3.1. The energy analysis is reported after taking into account all layout parasitics, while the switching activity has been computed using delay-accurate simulation of the derived logic-level netlists. The evaluated routers are all driven by the same arriving packet sequence, which

mimics uniform random traffic of 1-flit and 5-flit packets at an injection rate of 0.2 flits/cycle. The traffic characteristics determine the header contents of each packet, while the data contents – i.e., the payload – of each packet (mostly for body and tail flits) is produced using a uniform random number generator. In all cases, the energy required to drive the output links is also included.

ElastiStore-based routers require the least energy per cycle, due to the significant energy cost reduction of the buffers, which are responsible – together with the network links – for the majority of the energy required in each data transfer. The energy reductions due to ElastiStore surpass its area savings and lead to 23% and 28% more energy-efficient routers for 4 and 8 VCs, respectively.

3.6.2 Network Performance

Network-performance comparisons were performed using a cycle-accurate SystemC network simulator that models all micro-architectural components of a NoC router, assuming an 8×8 2D mesh network with XY dimension-ordered routing. The evaluation involves two synthetic traffic patterns: Uniform Random (UR) and Bit-Complement (BC) traffic. Other permutation traffic patterns follow very similar trends to BC traffic. The injected traffic consists of two types of packets to mimic realistic system scenarios: 1-flit short packets (just like request packets in a CMP), and longer 5-flit packets (just like response packets carrying a cache line). For the latency-throughput analysis, we assume a bimodal distribution of packets with 50% of the packets being short, 1-flit packets, and the rest being long, 5-flit packets, in accordance to recent studies [83].

Even with the lower amount of buffering – which translates directly to area/power savings – the ElastiStore-based routers achieve similar network performance when compared to single and two-cycle VC-based routers. Figures 3.10(a)-(b) depict the load-latency curves of all single-cycle routers under comparison using 4 VCs and 8 VCs under UR traffic. In all cases, the performance of the routers is virtually indistinguishable, both at low and at high loads, while the ES-IO configuration achieves slightly less delay at high loads, when compared to ES-I. The

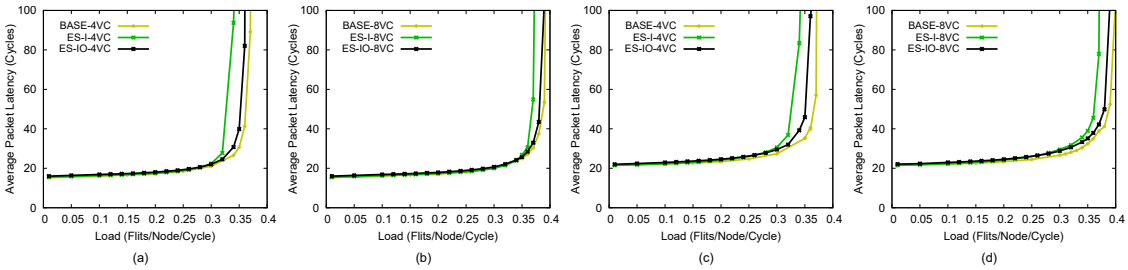


Figure 3.10: Latency vs. load curves for single-stage and two-stage baseline and ElastiStore-based pipelined routers with 4 VCs and 8 VCs under Uniform Random (UR) traffic. Configurations with 4 and 8 VCs per port are evaluated.

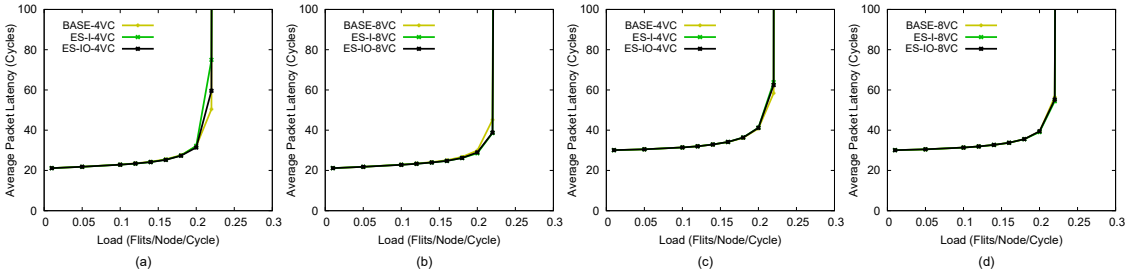


Figure 3.11: Latency vs. load curves for single-stage and two-stage baseline and ElastiStore-based pipelined routers with 4 VCs and 8 VCs under Bit-Complement (BC) traffic. Configurations with 4 and 8 VCs per port are evaluated.

same conclusion is drawn by the results shown in Figures 3.10(c) and (d), for the case of two-stage routers. Due to its directed nature, BC traffic eliminates the small differences in the performance of baseline and ElastiStore-based designs at high loads, as depicted in Figure 3.11. Therefore, *the savings of ElastiStore are offered to the NoC designer for free, without trading off performance*.

3.6.3 Full-System Simulation Results

Experimental Setup

To assess the impact of ElastiStore on overall system performance, we simulate a 64-core tiled CMP system running real application workloads on a commodity operating system. The execution-driven, full-system simulation framework employs Wind River's Simics [57] – which handles the functional simulation tasks – extended with the Wisconsin Multifacet GEMS simulator [85]. The latter provides a detailed timing model of the memory hierarchy and it includes the GARNET [1] cycle-accurate NoC simulator.

Table 3.2: System parameters for the execution-driven, full-system simulations.

Processor	64 in-order UltraSparcIII+ cores in a tiled CMP architecture
OS	Solaris 10
L1 caches	Private, separate 32 KB I & D, 4-way set associative, 2-cycle latency, 64 B cache-line
L2 cache	Shared NUCA LLC, 4-way set associative, 16 MB total (64 cores×256 KB slice/core), 10-cycle latency, 64 B cache-line
Coherence	MOESI directory-based cache coherence protocol
Main memory	4 GB, 300-cycle latency
Network	8×8 2D Mesh, 4-stage router pipeline (+1 cycle link delay), XY Routing, 3 VCs per input port
VC size	Baseline: 6 flits per VC ElastiStore: 1-flit register per VC, and a 5-flit shared

Table 3.2 shows the full-system simulation parameters. Each CMP "tile" consists of an in-order UltraSparc III+ processor core with private and separate 32 KB L1 I and D caches. The CMP has a total of 16 MB shared L2 cache (each tile has a 256 KB L2 slice; i.e., 64×256 KB = 16 MB total), and 4 GB of off-chip main memory (DRAM). The system uses the MOESI directory-based cache coherence protocol. The NoC is an 8×8 2D mesh (i.e., one router per CMP tile) employing a dimension-ordered XY routing. Each router is implemented as a conventional 4-stage pipelined router (RC, VA, SA, ST) with one cycle inter-router link delay. As previously mentioned, cache coherence protocols require isolation between the various message classes to avoid protocol-level deadlocks. Specifically, the MOESI protocol requires at least three virtual networks to prevent protocol-level deadlocks. Consequently, in our simulations, each router input port has 3 VCs, each handling a specific message class of the coherence protocol.

Two different router architectures were considered. The *baseline* router design uses 3 VCs per input port, with each VC buffer having a 6-flit depth. This setup represents a "traditional" NoC input port architecture, where the buffer space is statically allocated to each VC. On the contrary, the proposed ElastiStore architecture uses only one single-flit register per VC, plus a 5-flit buffer shared among all 3 VCs, which aims to provide a direct comparison with the baseline setup, since each VC

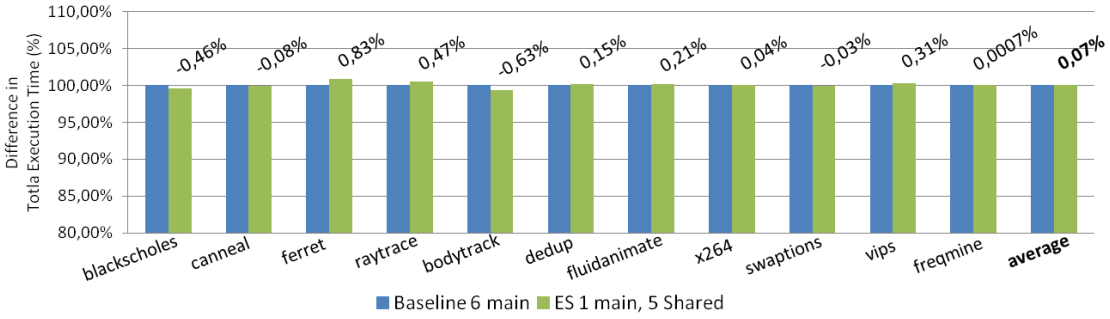


Figure 3.12: The total execution times of the various PARSEC benchmark applications, normalized to the baseline router. The baseline router setup uses 3 VCs per input port, with each VC buffer storing 6 flits. The ElastiStore setup uses one single-flit register per VC, plus one multi-flit buffer of size 5, which is shared among all 3 VCs.

can hold a maximum of 6 flits (1 in the main VC register + all 5 flits in the shared buffer). The six-flit buffers are necessary to cover the 6-cycle round-trip time of a 4-stage pipelined router (+1 inter-router link stage).

Both the baseline and ElastiStore-based router architectures were implemented within GARNET. The GARNET NoC simulator cycle-accurately models the packet-switched routers, their pipelines, virtual-channel buffers, allocators/arbiters, crossbars, and all inter-router links.

The executed applications are part of the PARSEC benchmark suite [7], which contains multi-threaded workloads from various emerging applications. All benchmarks were executed with 64 threads (one thread per processing core). The execution times reported are those of the "Regions Of Interest (ROI)", as identified in the PARSEC benchmarks. The ROI of each benchmark starts right after the initialization of the input data and ends when the computation is complete.

Results with PARSEC Applications

We ran the PARSEC benchmarks [7] using the setup described in the previous sub-section to evaluate the two different NoC configurations. Figure 3.12 shows the total execution times of the various applications, normalized to the baseline router.

The important insight that can be extracted from Figure 3.12 is that a lightweight ElastiStore design with one single-flit register per VC and a 5-flit shared buffer can yield near-identical performance as a baseline

design with a 6-flit buffer per input VC. In fact, the performances are indistinguishable. Both router architectures can provide a maximum space of 6 flits per VC, but the ElastiStore setup shares this maximum depth among all VCs (through the 5-flit shared buffer). This sharing results in much more efficient resource utilization, with no impact on performance. The reason why such a dramatic decrease in buffer space is *not* accompanied by a decrease in overall system performance is due to the very low average NoC traffic injection rates observed when running real single- and multi-threaded applications in CMPs [44]. Hence, the baseline router architecture is, in fact, significantly over-provisioned for the needs of real application workloads, such as the PARSEC benchmarks. The fact that the ElastiStore architecture provides the same performance as the baseline with only 44% of the buffer space (a total of 8 flit slots per input port versus 18 in the baseline) results in substantial savings.

3.6.4 Virtual Channels vs. Multiple Physical Networks

The separation of resources offered by VCs can also be achieved by multiple physical networks (built with wormhole routers), where each physical network serves a certain VC – or, more accurately, a virtual network, since moving from one VC to another one is impossible in the case of multiple networks, due to the physical separation of the networks. Low-cost wormhole routers can be built with simple EBs at the inputs and the outpus of the router, as proposed in [89], using the 2-slot EBs of Figure 3.1(c), or 3.1(d).

Comparing VC-based architectures with multiple physical VC-less networks is a multi-dimensional problem, which has been the focus of other independent research efforts, such as [151]. However, we repeat part of this study using ElastiStore-based VC routers.

In our comparisons, we consider four cases of an 8×8 2D mesh network, which offers 8-way separation of resources, assuming a channel width of 64 bits. The first examined case involves a network built with 8-VC ElastiStore-based routers, where ElastiStores are used only at the inputs of the router (ESI-8VC-64). The second and the third case involve 8 physical networks of EB-based wormhole networks. The second case uses 64-bit channels per network, thus having a total of 64×8 bits

per channel (EB-WHx8-64), while the third case assumes equal-bisection bandwidth with the VC-based networks and uses 64/8 bits per channel (EB-WHx8-13). However, since we would like to keep the packet's header in one flit, we need at least 13 bits per network channel (2 bits for the flit's ID, 6 bits for the network addressing, and 5 bits for encoding the output port request, as needed by the lookahead routing computation employed by all routers under comparison). The last case involves a hybrid of both worlds. It consists of 2 physical networks of 4-VC ElastiStore-based routers, which – under equal bisection bandwidth – operate on 32-bit channels (ESI-4VC-x2-32).

While the first two cases can send and receive directly the 1-flit and 5-flit packets used in the previous experimental setup, the third and the fourth cases impose a significant serialization latency, since the number of flits per packet should be increased by 5× and 2×, in order to fit into the 13-bit and 32-bit channels, respectively. Please note that EB-WHx8-13 gets 1.6× more bisection bandwidth that ElastiStore-based architectures due to the 13-bit channels.

For a fair comparison, we assume a static VC allocation policy for the VC-based routers. In this case, packets are not allowed to change VC in-flight and are forced to keep the VC given to them at the source (i.e., similar to what happens when a packet enters a physical network of VC-less routers). This feature simplifies significantly the VA logic of VC-based NoC routers [37], with a slight reduction in the overall network throughput, and can be used with ElastiStore, too.

Firstly, we compare the four examined cases in terms of network performance, using the same configuration of Section 3.6.2 for UR traffic. The results are shown in Figure 3.13(a). EB-based wormhole routers are small and fast. Thus, in the load-latency curves of Figure 3.13(a), we assume that the routers can switch incoming flits in one cycle. On the contrary, VC-based routers operate in a 3-stage pipelined configuration to achieve the same clock frequency. As expected, the EB-WHx8-64 configuration has the best performance, both in terms of latency and throughput. The EB-WHx8-13 setup is the worst. ElastiStore-based VC-based architectures enjoy high throughput of operation, despite having 8× less bisection bandwidth than EB-WHx8-64, and, as it will be shown later, they achieve this goal with significantly less cost. The only draw-

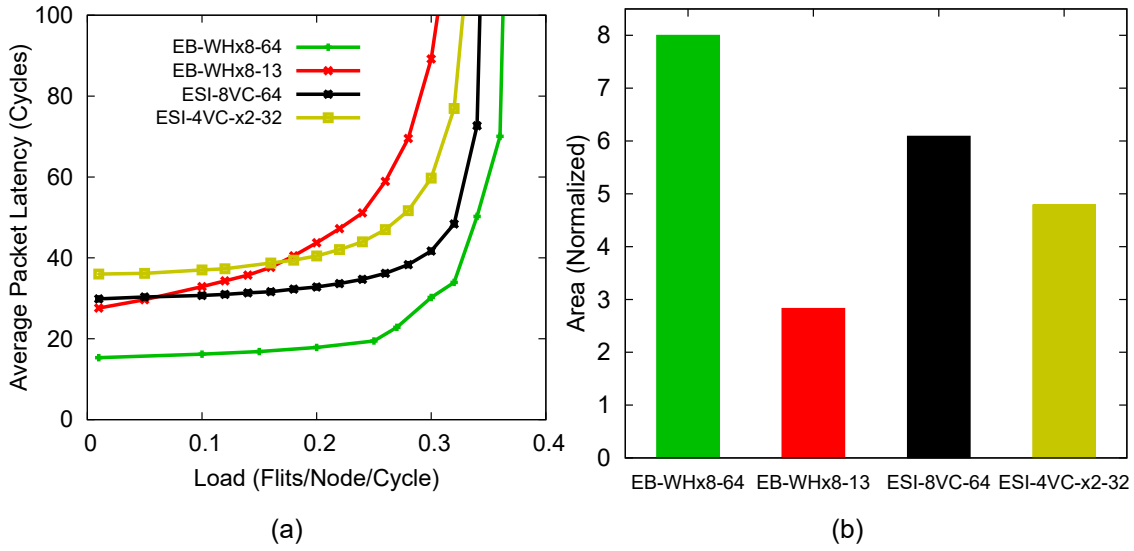


Figure 3.13: (a) The load-latency curves, and (b) the normalized area cost of all examined configurations for comparing *virtual* (using ElastiStore buffers), or *physical* separation of resources. The area results are normalized to the area of a single EB-based wormhole network.

back of the VC-based architectures is the high zero-load latency, due to the increased number of pipeline stages required to achieve high clock frequencies.

The cost of the implementation of each case (in terms of area) is shown in Figure 3.13(b), where – for the ElastiStore-based routers – 3-stage-pipelined alternatives are examined with static VC allocation, which have almost equal delay to that of simpler, single-cycle EB-based wormhole routers. ElastiStore with 8 VCs allows the design of NoCs offering 8-way separated resources, with less area cost than 8 physical EB-based WH networks. The 24% area savings are the result of the efficient buffer sharing mechanism of ElastiStore, derived using the three design principles presented in Section 3.4.2 and the newly introduced credit-based flow control that employs negative credits. As expected, the 8 physical networks of EB-based WH routers operating under equal bisection bandwidth have lower area cost, but, as shown in Figure 3.13(a), they also have the worst performance, both in terms of latency and saturation throughput. The hybrid solutions that employ multiple physical networks of ElastiStore-based routers, each one supporting a smaller number of VCs keep the network cost low with acceptable performance.

The final outcome of this analysis is that ElastiStore significantly re-

duced the cost of VC-based routers, without sacrificing the throughput of the network, thereby allowing the design of NoCs with a high-degree of resource separation with lower cost than multiple physical networks. The remaining challenge for VC-based architectures, which is orthogonal to ElastiStore, is the simplification of their allocation logic to decrease the number of pipeline stages required for achieving high clock frequency. This would also lower the zero-load latency of each flit.

3.7 Related Work

In this section, we focus on the two main thrusts of prior research that are most relevant to the proposed ElistiStore design: (a) elastic channels and buffering in NoCs, and (b) shared-buffering schemes in NoC routers.

Kodi et al. [72] explored the integration of elastic storage elements into the links of NoCs, in conjunction with traditional dynamically-assigned VC input buffers. The introduced iDEAL router employs a hybrid flow control, where multiple VCs are multiplexed on a single-lane link with a ready/valid handshake. This hybrid flow control obligates the first VC that is full to stop the remaining VCs – with possibly incoming flits on the link – thus creating dependencies across VCs that may lead to higher-level protocol deadlocks if not handled appropriately, irrespective of the available buffer space.

The notion of elastic channels in NoCs was further developed by Michelogiannakis et al. [89, 90], which reused the latch-based elastic buffers presented in [22] for minimizing the buffering cost in NoC routers. To alleviate the problems caused by the serializing nature of elastic links, which does not provide any isolation of traffic flows and prevents the interleaving of packets – the authors resort to multiple physical channels, in order to create multiple sub-networks, instead of relying on a hybrid flow control mechanism.

A modified version of elastic operation, which also enables traffic-flow separation without multiple physical resources, has been employed in [39] and [41]. In [39], elasticity is only preserved across the multi-drop busses of the MECS topology, and, when it comes to the inputs of the routers,

traditional VC-based buffers are used. On the contrary, in [41], elasticity is preserved on single-lane channels, while separation of resources is enabled by packet-level bubble flow control.

In contrast, ElastiStore uses lightweight EB primitives, and it can support any number of VCs, without the need to have multiple physical sub-networks, or to rely on any hybrid flow control mechanism. Moreover, ElastiStore can be used either as a distributed buffer primitive, or at the inputs and output of routers. The semantics of the VC flow control mechanism are preserved, implemented either with independent ready/valid handshakes per VC or using credits. The presence of a shared buffer in ElastiStore is instrumental in optimizing the use of the available buffer space and covering the arbitrary round-trip times imposed by router pipelining. In essence, the proposed mechanism achieves the same objective as the significantly more expensive per-input shared-buffering techniques [136, 135, 66, 103, 102, 105]. Other buffering architectures that extend sharing across multiple inputs [138, 101, 76] require multi-porting and even though they provide significant performance in terms of throughput, they suffer in terms of router delay (or they increase the required pipeline stages), and they do not scale well to higher numbers of VCs per input port.

As opposed to the extremely lightweight shared-buffering structure of ElastiStore,the aforementioned shared-buffering architectures rely on fairly complex logic to keep track of the location of flits within the unified buffer, which needs significant modifications to handle variable packet sizes. Specifically, in shared-buffer schemes, designers predominantly use linked lists [136, 135, 66], table-based approaches [103] that may possibly need multi-ported memory accesses in their pointer-tracking logic, or self-compacting buffers [102, 105] to coordinate traffic flow through the buffers. Each VC maintains its own set of pointers to identify where its flits are located in the buffer. For example, in the case of ViChaR [103], two control logic modules are needed in each input port (the Arriving/Departing Flit Pointers Logic and the Slot Availability Tracker) for the correct operation of the shared buffer, regardless of the size of the buffer. While the cost of this logic is amortized in routers with large buffer space, the overhead becomes significant in routers with minimal buffering.

Furthermore, state-of-the-art buffering mechanisms assume that all flits of the VCs that reside in the shared buffer space should be visible in the allocation logic, and the throughput per VC is a result of the VC utilization and the status of the buffer space (available credits) in each cycle. This behavior is avoided in ElastiStore, which allows only the main registers of each VC (just 1 per VC) and not the shared buffer to participate in allocation, and the throughput received per VC follows a more strict distribution, allowing only one VC to receive full throughput when it is the only active one (this is the only case that full throughput matters). The shared buffer is isolated from the router's operation; it just refills the main registers of the VC, when needed. In this way, every dependency across VCs is eliminated – which also enables deadlock-free operation – while variable packet sizes are supported for free, as in any baseline VC buffer with no sharing. The shared buffer of the generalized ElastiStore partially follows a self-compacting approach, similar in concept to [102, 105], although much simpler to implement. Flits are written at the end of a FIFO irrespective of the VC they belong to, and no data movement is needed to find an empty place for an incoming flit.

3.8 Conclusions

The NoC router's buffer architecture is a critical design aspect that affects both network-wide performance and implementation characteristics. In this work, we efficiently merge elastic operation and buffering with virtual-channel flow control. The derived buffer architecture, called ElastiStore, can take many forms, based on application demands. ElastiStore can be used as the simplest form of VC buffering, which uses only 1 register per VC, plus one more dynamically shared register that enables a single active VC to achieve full throughput. Additionally, when NoC routers follow a pipelined organization, ElastiStore can be adapted to its most generic form, which utilizes a larger shared buffer to cover the increased round-trip time arising from the pipelined operation. The new design principles governing the design of ElastiStore-based routers enable the design of low-cost routers with significant area savings and no delay penalty, as compared to current state-of-the art VC-based routers. More importantly, the resulting ElastiStore-based routers offer the same network performance as the aforementioned VC-based

implementations, as verified using extensive simulations with both synthetic traffic and real application workloads.

Chapter 4

Distributed VC-based Network-on-Chip Architecture

Traditional VC-based NoC architectures focus mostly on microarchitectural improvements to the router's internal organization and pipeline structure [108], [98]. Prior research has explored salient router attributes, such as appropriate allocation policies [91], as well as the optimization of the associated VC buffering structures [138, 5, 41], concentrating mostly on buffer sharing and related flow control strategies.

In this chapter, we revisit first the pipelined configurations of baseline routers with the goal of identifying – via a simple intuitive analytical model – the amount of pipelining needed to achieve optimal network latency under arbitrary topologies, packet sizes, and routing algorithms. Our analysis aims to shed more light on previous design trends that targeted primarily the reduction of the intra-router pipeline stages. We clearly show that router pipelining (and its associated clock frequency benefit) will *always be beneficial*, even for simple NoC designs, when applied with care, so as to avoid over-pipelining and its associated diminishing returns.

Motivated by this analysis, we introduce ElastiNoC a *distributed* VC-based router architecture, which enables fine-grained pipelining and provides maximum flexibility in terms of NoC physical placement. The proposed structures are also enhanced with novel *self-testability* features and a scalable testing mechanism that achieves high fault coverage with

small test application time.

While the concepts of distributed router design and fine-grained network pipelining have been explored in the past, the focus has been on applying the said attributes to designs that do not support VCs [116, 3, 115, 113, 48]. Supporting VCs in that context needs multiple parallel networks of such distributed routers. Obviously, multiple networks do not constitute the most resource-efficient solution, due to inevitable resource duplication.

Hence, the need for an architecture that efficiently combines all these benefits with support for VCs is imperative. To the best of our knowledge, the design proposed in this thesis is the first distributed VC-based router implementation that supports this form of modularity. The combined effect of all introduced features enables the design of highly scalable VC-based NoC architectures, which offer high operating frequencies and provide equal (or even better) networking performance, as compared to state-of-the-art VC-based implementations.

4.1 Modeling Low-latency On-Chip Networks

In this section, we develop a simple intuitive analytical model that connects the network latency with the routers' operating clock frequency and their internal pipeline organization. The goal is to construct a model that enables the designer to derive a first-order approximation to an optimal configuration, given certain parametrical constraints. The presented model, although based on several simplifying assumptions, provides valuable intuition on when router pipelining is needed, and which pipeline depth makes sense to implement.

First, assume that the NoC's topology and size are fixed, and the possible use of concentration has already been decided. Moreover, assume that the link bit-widths have also been decided. Such decisions fix the radix of the routers and their port sizes, which are critical factors in determining the overall delay. Still, even for fixed-radix routers, their delay can vary significantly, depending on the microarchitecture of the routers (e.g., support for VCs, allocation organization etc.) and other implementation constraints.

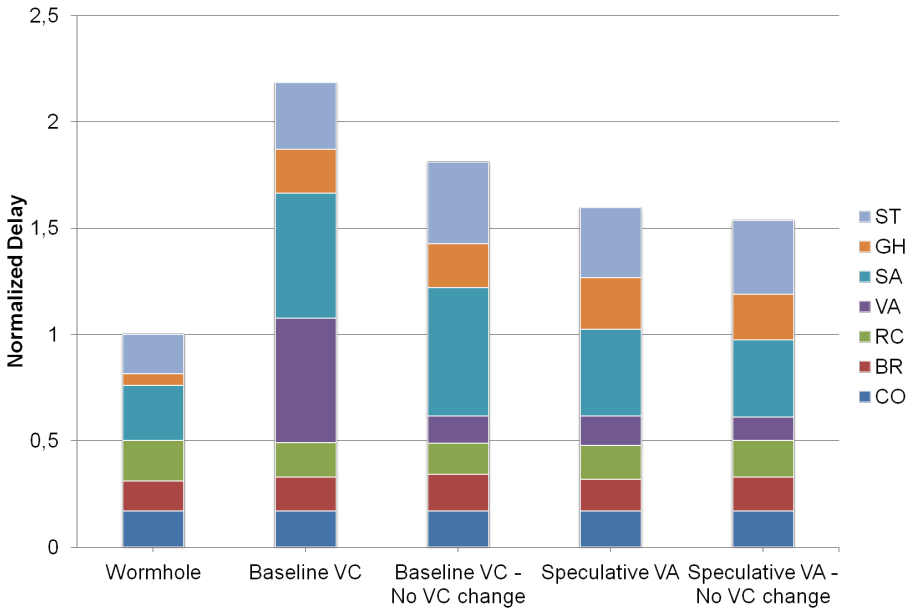


Figure 4.1: The delay of representative single-cycle 5-port NoC routers with 64-bit wide ports and 4 VCs per input in the case of VC-based routers. The results are normalized to the delay of a wormhole-based router (i.e., no VC support). "No VC change" means that packets do not change VC; their VC is decided upon injection and remains the same until they reach their destination.

Figure 4.1 shows the normalized minimum delay of several single-cycle routers with 5 input ports and 64-bit wide channels when synthesized in 45nm technology. The comparison includes (a) a simple wormhole router, (b) a VC-based router with baseline VC allocation, whereby packets can change VC in-flight, (c) a VC-based router with baseline VC allocation, whereby packets are not allowed to change VC, (d) a VC-based router with speculative VC allocation, whereby packets can change VC, and (e) a VC-based router with speculative VC allocation, whereby packets cannot change VC. All delays are normalized to the delay of the wormhole router, which does not support VCs. In all cases, the routing computation is performed in series with the remaining tasks.The VC-based routers have 4 VCs per input port with 4 buffers per VC. The wormhole router has 4 buffer slots per input port.

The delay of each single-cycle router is the sum of several sub-tasks, such as Buffer Read (BR), Routing Computation (RC), VC Allocation (VA), Switch Arbitration (SA), Handling of returning Grants (GH), and Switch Traversal (ST), which also includes the delay of credit updates and VC state re-allocation (in the case of a tail flit leaving an output port). Note

that in speculative routers that do VA and SA in parallel, the critical path passes through the SA unit. Even though the evaluated routers have completely different behavior in terms of throughput-versus-load performance, they represent almost all design options available for the design of monolithic NoC routers. In any case, the minimum clock cycle that a single-cycle router can operate at is $T_{CYC} \geq D + c$, where D represents the worst-case delay of the router's internal paths, and c is the clocking overhead (sum of the register clock-to-Q delay and the setup time; depicted as CO in Figure 4.1).

Router pipelining is expected to reduce the clock period. However, every pipelining decision stops across the borders of the traditional basic blocks within each router, e.g., VC allocation, switch arbitration, and switch traversal. The fact that such blocks do not have an evenly balanced delay profile – as shown in Figure 4.1 – makes pipelining even harder, since the achieved clock frequency is limited by the delay of the critical path. For the optimal case, we can assume that it is possible to break the router's critical path into k equal-delay stages. Then, the router's clock period can drop to

$$T_{CYC} \geq \frac{D}{k} + c \tag{4.1}$$

By increasing the pipeline stages of a router, its clock frequency can increase, thereby leading to faster implementations. At the same time, however, each flit spends more cycles inside each router, before moving to the next one. Therefore, the depth of the pipeline cannot be decided in isolation; the decision should also take into account other network parameters, such as the number of hops each packet needs to make between source and destination pairs, and the average packet size, assuming that a mix of packets of different sizes may traverse the network.

The hop count is determined by many factors, such as the network topology and size, the employed routing algorithm, and the statistics of the traffic patterns. To keep our model simple, we incorporate the contributions of all these factors within one variable, i.e., the average hop count H, which averages the contribution of each feature. Thus, the zero-load latency (in cycles) of a packet is equal to

$$T = H\,(k+1) + P - 1 \tag{4.2}$$

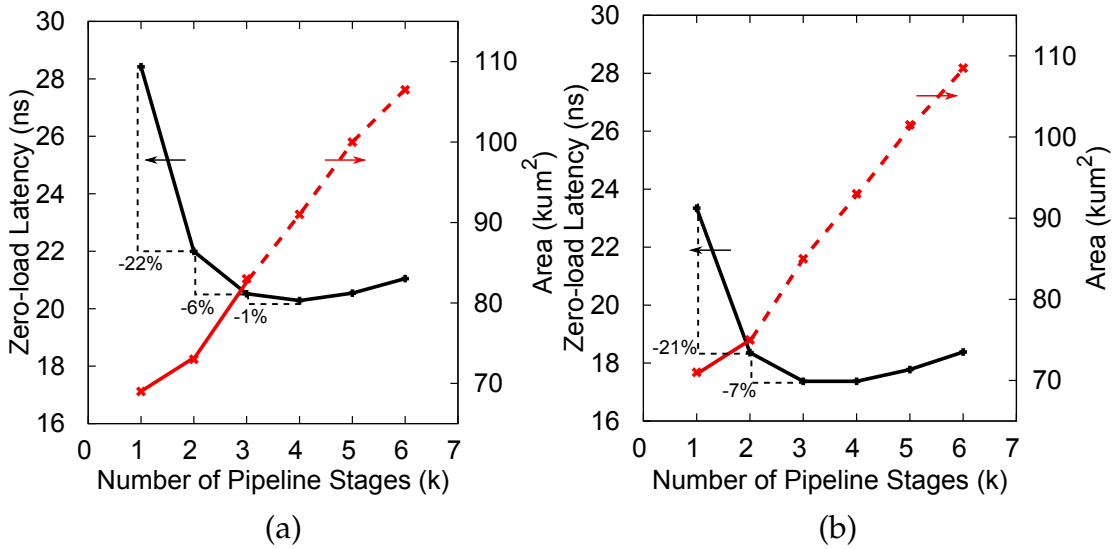


Figure 4.2: The average zero-load packet latency (in absolute time) computed directly from eq. (3) and the associated router area overhead, as the number of pipeline stages are varied. Results are shown for (a) a baseline, and (b) a speculative VC-based router assuming 4 VCs per input port.

Each flit spends k cycles in each k-pipelined router and 1 cycle to cross the link between two routers. Variable $P = L/W$ represents the serialization latency of a packet with a size of L bits traveling over W-bit wide physical links[1].

Since each k-stage pipelined router works with a clock period of T_{CYC}[2], the zero-load latency of each packet in *absolute* time is the product of the latency in cycles and the minimum clock period of a pipelined router:

$$T_{ABS}(k) = T \times T_{CYC} = (H(k+1) + P - 1)\left(\frac{D}{k} + c\right) \tag{4.3}$$

It should be noted here that even if the use of the packet's zero-load latency alone is not sufficient to fully capture a NoC design's behavior, the resulting configurations will still hold for the majority of possible network loading conditions that are not close to the saturation throughput.

[1]The minus one removes the contribution of the head flit, which is included in the first term of the equation.

[2]For simplicity, we assume that all NoC components (e.g., routers and links) belong to a single clock domain.

To explore the interesting interplay between packet latency and pipeline depth, we fix the average hop count to $H = 6.25$, which roughly corresponds to deterministic XY routing in an 8×8 2D mesh, assuming uniform random traffic and an average packet size of $P = 3$ (50% 1-flit request packets and 50% 5-flit reply packets). For this configuration, the zero-load latencies T_{ABS} (as a function of k) of (a) a baseline and (b) a speculative single-cycle VC-based router that allows for in-flight VC changes, are shown in Figure 4.2. When k moves from 1 to 2 ($k = 1$ corresponds to the un-pipelined single-cycle solution), the latency savings are significant and are above 20%. The addition of more pipeline stages reduces packet latency, but with diminishing returns. For example, moving from 3 to 4 pipeline stages offers less than 1% savings in packet latency, without justifying the additional cost in control logic and buffering resources. In a VC-based router, the number of buffers should be equal to the minimum required to cover the flow-control round-trip latency; else, throughput is severely compromised. Pipelining increases the round-trip delay, which, in turn, increases the mimimum buffering requirements of the entire router. Therefore, any pipeline decision should also take into account the buffering cost that this option incurs. Figure 4.2 depicts the area cost required for each pipelined alternative. Straight lines are actual measurements after synthesis while dashed lines correspond to calculations that add the area of extra buffering. Inspecting packet latency and buffering cost *together* leads to the conclusion that pipelining is indeed a useful design choice that *ends its useful contribution at around 3 pipeline stages*. Above that point, the investment in extra area due to more pipeline stages is not compensated by the (diminishing) reductions in packet latency.

In addition to the low-radix scenario examined above, we also experimented with *high-radix* routers (e.g., those found in a flattened butterfly topology [68]) to explore optimality in networks with lower hop counts, but higher router latencies (due to the complexities associated with high-radix designs). Our evaluation results – omitted for brevity – indicate that optimal pipelining in those scenarios is achieved with 4 or 5 stages, depending on the various salient parameters.

Using the simple analytical model leads to two interesting conclusions. The first one is that the decision of pipelining the router cannot be made

solely based on its delay, but the process should also take into account the environment in which the router will operate. The second one (and perhaps non-intuitive) is that the designer should not only opt for microarchitectural optimizations that decrease the router's delay by parallelizing its tasks (e.g., with speculation), but, instead, should *embrace a combined approach that utilizes optimal pipelining*. This realization serves as the *primary motivation* and *fundamental driver* of the work presented in this thesis.

Unfortunately, in state-of-the-art monolithic router structures, pipelining decisions stop across the boundaries of the traditional basic blocks, which have been widely viewed as "atomic" (i.e., indivisible). Furthermore, the delay of these blocks is not evenly balanced. Therefore, even if 3-stage pipelines (or 4- and 5-stage pipelines in high-radix environments) are still possible with this coarse separation, the achievable clock frequency would be sub-optimal, since the speed of the router would be limited by the worst-case delay of the most delay-critical block. Additionally, most existing router designs are inherently centralized in terms of their physical layout. This is attributed to certain monolithic components within each router; the crossbar switch, the allocators, and the buffering structures significantly limit the possible flexibility in the physical placement of the overall router design. Consequently, current architectures are not spatially scalable, i.e., they cannot be efficiently distributed in space. This limitation may also have adverse effects on the router's delay.

These limitations of traditional VC-based router architectures are addressed by the ElastiNoC architecture proposed in this work. The new design philosophy: (a) enables modular pipeline implementations, (b) yields high operating frequencies, and (c) allows for efficient spatially distributed hardware implementations. The latter characteristic provides the floor-planning and placement tools with more freedom in generating optimal layout configurations.

4.2 ElastiNoC: Modular VC-based Architecture

Any network topology, from single-stage crossbars to arbitrary cubes, meshes, or butterfly-based structures can be implemented by decom-

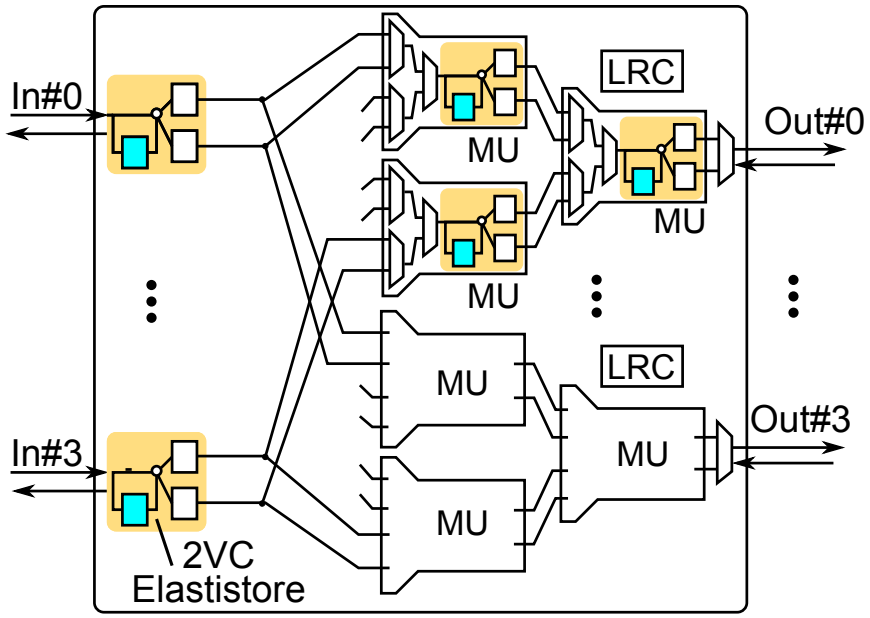


Figure 4.3: The modular construction of an example ElastiNoC 4×4 VC-based router using the proposed MU primitive that supports 2 VCs.

posing the switch operation to primitive merge and split functions. We design, for the first time, novel merge primitives (and the associated simplified split structures) that support VCs and offer the same degree of flexibility – in terms of network performance and functionality – as monolithic VC-based routers, but with higher-operating speed, and distributed physical placement capabilities.

4.2.1 Modular Router Construction

The fundamental primitive of ElastiNoC, called the Merge Unit (MU), consists of two inputs and one output. Its goal is to switch and buffer locally the flits of two inputs that belong to different VCs. Buffering is done via ElastiStore units [127], which follow an elastic protocol and are able to simultaneously store the data of many VCs using the minimum amount of buffering. Each ElastiStore module comprises one single-flit register per VC, plus one other single-flit register that can be dynamically allocated to the first stalled VC.

By using MUs and splitting the data arriving at each input port, one can design an arbitrary VC-based router. An example is shown in Figure 4.3, which depicts an ElastiNoC router with 4 inputs and 4 outputs. Upon

arrival at the input of the router, each packet has already computed its destined output port via Look-ahead Routing Computation (LRC). Subsequently – depending on buffer availability, output VC availability, and the allocation steps involved in each MU – the flits of the packet are forwarded to the MU of the appropriate output. Integration of MU and ElastiStore primitives is straightforward, since they all operate under the same ready(i)/valid(i) handshake protocol. All router paths from input to output see a pipeline of MUs of $\log_2 N$ stages. Moving to the next router involves one extra cycle on the link that is just a one-to-one connection between two ElastiStores. The flow control on the links does not allow packets to change VC and its operation needs only an arbiter and a multiplexer for selecting a flit to send to the next router.

The fact that all input-to-output paths experience $\log_2 N$ stages of MUs is extremely important. This attribute aligns ElastiNoC with the optimal pipelining conclusions extracted in Section 4.1 for both low- and high-radix routers. For low-radix routers (with 5-8 input ports), optimal pipelining calls for 2-3 stages, while the 4-5 pipeline stages required for high-radix routers (with more than 12 input ports) are also in agreement with the logarithmic number of stages of the proposed architecture. Thus, ElastiNoC allows for sufficiently fine-grained modularity, which can yield optimally pipelined designs over a wide spectrum of router radices.

Due to the distributed nature of ElastiNoC, the split connections can be customized to reflect the turns allowed by the routing algorithm. For example, in a 5-port router for a 2D mesh employing XY dimensioned-ordered routing, splitting from the Y+ input to the X+ output is not necessary since this turn is prohibited. Several other deterministic and partially-adaptive routing algorithms can be defined via turn prohibits as shown in [34]. When such customizations are applied, significant area savings are expected, due to the removal of both buffering and logic resources. On the contrary, such optimizations do not make sense in traditional VC-based routers, since only parts of the crossbar and switch allocation logic are reduced, while leaving input buffering, that is responsible for the majority of the router's area, unaffected.

This modular router construction enables packet flow to be pipelined in a fine-grained manner, implementing all necessary steps of buffering,

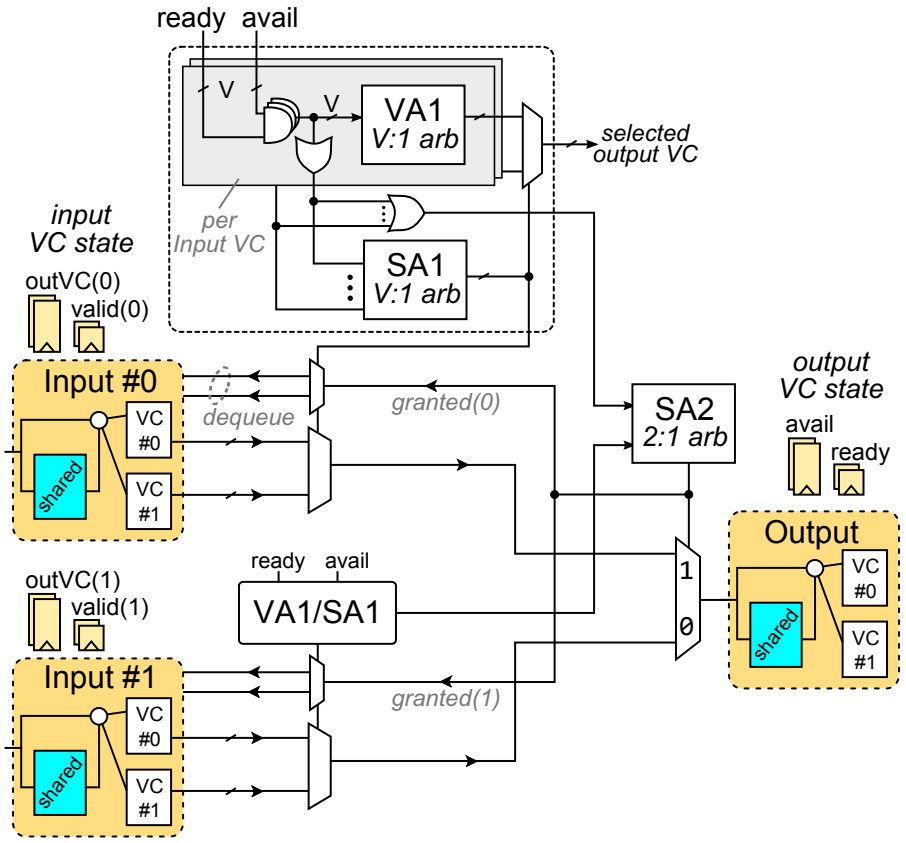


Figure 4.4: The fundamental ElastiNoC primitive, the Merge Unit (MU). The diagram depicts the per-input and per-output multiplexers together with the combined allocation logic (SA1, SA2) that runs in parallel to VA1.

VC and port allocation, and multiplexing in a distributed way inside each MU, or across MUs. Also, the placement of MUs does not need to follow the floor-plan of the chosen NoC topology. Instead, MUs can be freely placed in space, provided that they are appropriately connected.

4.2.2 The Merge Unit (MU)

Each MU is responsible for switching one output between 2 inputs assuming the existence of per-input and per-output VCs, as shown in Figure 4.4. Since switching is achieved by connecting several MUs in series (as illustrated in Figure 4.3), the buffers presented at the input of Figure 4.4 are actually the output buffers of the previous MUs.

Packets arriving at the two inputs of each MU must compete for a single output. Since the output can carry flits that belong to different VCs, each packet has to first allocate a VC at the output of the MU (known as an

"output VC"), before leaving the input VC. Allowing packets to change VC in-flight, within each MU, is possible when the routing algorithm does not impose any VC restrictions (e.g., XY routing does not even require the presence of VCs). However, if the routing algorithm and/or the upper-layer protocol (e.g., cache coherence) place specific restrictions on the use of VCs, arbitrary VC changes are prohibited, because they may lead to deadlocks. Any restrictions are enforced by the allocator of the MU.

Our goal is to make the MU as fast as possible without sacrificing throughput. Therefore, we follow a combined allocation approach [82], customized and optimized to the characteristics of our design by allowing packets to change VC in flight at the granularity of a single MU. Each input VC holds two state variables showing (a) if the VC has valid data, and (b) if it has been allocated to an output VC. Each output VC also holds two state variables: (a) variable "available" indicates whether it is currently allocated ("locked") by an input VC, and (b) variable "ready" indicates if there exists free buffer space, which, in our case, is received by the output ElastiStore's ready signals.

When a head flit arrives at an input VC it simultaneously tries to get matched to an output VC, and also to gain access to the output port of the MU. Both actions should be successful for the head flit to reach the output of the MU. Before issuing any request to the allocation logic, the head flit checks if there is at least one available and ready output VC (readiness corresponds to buffer availability). If this is true, the head flit issues a request to SA1 that promotes one flit from each input. Next, the two input ports (i.e., the SA1 winner of each port) arbitrate in global SA (SA2) to advance to the output port via the data multiplexers driven by the grant signals of the SA1 and SA2 round-robin arbiters.

In parallel to SA1 and SA2, the head flit has to select one available output VC. This is done independently per input VC using one V:1 round robin arbiter (VA1), where V denotes the number of supported VCs. Thus, when a head flit wins SA2, it is allocated to the output VC selected in parallel by VA, and it updates its per-input state variable. On the contrary, if a head flit loses in SA2, it will not refresh its VC state and retry in the next cycle, repeating the whole process. The parallel operation of VA1 and SA1-SA2 does not involve any speculation, since

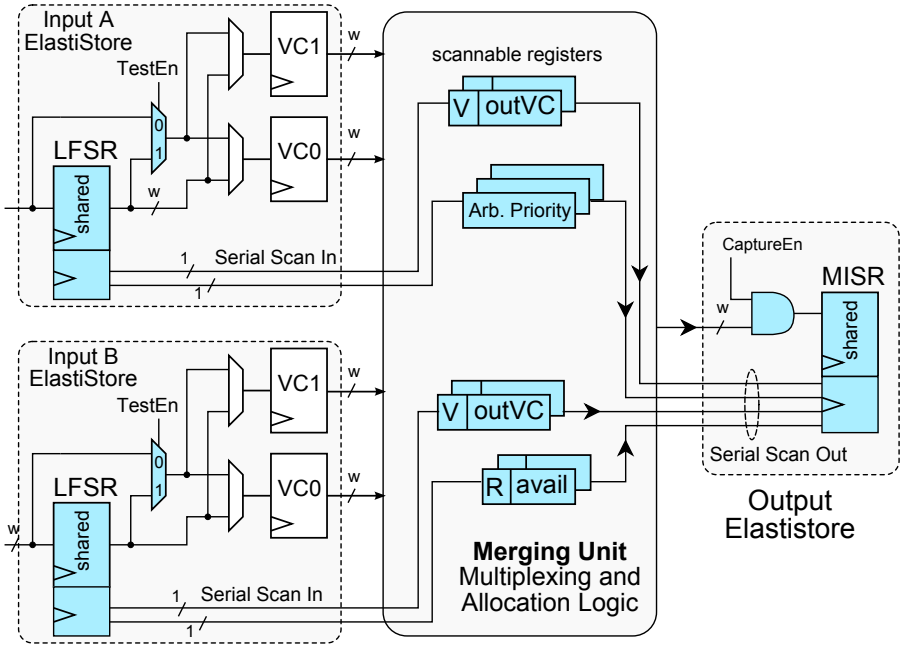


Figure 4.5: BIST enhancements of a merge unit by incorporating the logic of LSFR/MISR within that of the shared buffer at each ElastiStore.

SA1 requests are considered valid only when there is at least one available and ready output VC. The stored input VC state is inherited by the packet's following body and tail flits, which use it (a) to generate an SA1 request (after checking with the output VC's readiness), and (b) to reach the same output VC, after winning in SA2, as welll. The tail it is also responsible for releasing both the per-input and the per-output state variables, allowing the output VC to be allocated to another packet.

ElastiStore allocates the minimum of just one register (i.e., holding a single flit) per VC, plus one additional register that is shared dynamically between VCs, and enables full throughput of elastic operation using one ready/valid handshake signal per VC. The static allocation of a single buffer to each VC guarantees forward progress for all VCs and avoids possible protocol-level deadlocks.

4.3 ElastiNoC Self Testability

As technology continues to scale and chips continue to grow, system reliability and scalable Built-In Self-Test (BIST) architectures are gaining significant importance. NoC testing has evolved over the recent

years providing topology-agnostic and modular self-testing methodologies [65, 134]. The distributed structure of ElastiNoC does not match well with traditional core-level BIST architectures [87]. Therefore, we targeted the design of a new distributed BIST architecture that (a) reuses as much as possible the hardware of ElastiNoC, (b) achieves high fault coverage and fault localization at the MU level (detects which MU contains a faulty node) and (c) completes NoC testing within a small number of test cycles. The last feature is critical when the NoC that reaches all IP cores of the system is used as a test access mechanism for those cores. In such cases, the sooner the NoC is tested, the sooner the testing of the rest of the system can begin.

The self-testability features of ElastiNoC are applied at the MU level. Our target is to test the two input ElastiStores of each MU, along with the associated multiplexing and allocation logic, and capture the responses in the shared buffer of the output ElastiStore. To achieve this, the shared buffers of the input ElastiStores should function as Test Pattern Generators (TPGs) during testing, and specifically as Linear Feedback Shift Registers (LFSRs), so as to provide pseudorandom patterns to the tested circuit. Furthermore, the shared buffer of the output ElastiStore of the MU should act as a Multiple Input Signature Register (MISR), in order to compact the responses. This organization, shown in Figure 4.5, allows us to reuse the flip-flops of the shared register of each Elastistore and transform it into a Built-In Logic Block Observer (BILBO) with small hardware overhead (a BILBO register combines the operation of an LFSR, an MISR, and a shift register). Testing of a router's MUs that belong to the same switching level constitutes an independent test phase. In the next test phase, where the previous and the next MU levels are tested, the functionality of the shared buffers as LFSRs/MISRs is inverted, since the output ElastiStores of the current level are the inputs of the next, whereas the inputs of the current level are the outputs of the previous one. A test phase can be applied simultaneously to all NoC routers.

Allowing the shared buffers of the input and output Elastistores to act as TPGs and response compactors respectively, requires some additional test isolation logic that is enabled only during testing. The Elastistores under test (input ElastiStores) are isolated using 2-to-1 mul-

tiplexers that multiplex their data/control inputs with the outputs of the shared buffers that act as LFSRs, as depicted in Figure 4.5. During testing (TestEnable=1) the bypass multiplexers of Elastistores get the same value, and thus the outputs of the LFSRs propagate in the MU irrespective of the value on the select lines of the bypass multiplexers.

Additionally, every other testing logic added should pay off in terms of fault coverage. Data registers are easily testable since they are directly accessible. The combinational logic of the MUs can be easily tested as well. Testing gets complicated for the input/output VC state registers and the priority state of each round-robin arbiter that can be accessed and observed only implicitly. Our preliminary sequential ATPG and fault-simulation experiments indicate that long test sequences with top-off deterministic patterns cannot achieve anything more than just moderate fault coverage. For that reason, we have chosen to adopt a partial-scan approach, where the internal state registers shown in Figure 4.5 are put in scan chains (the tested circuit, as a whole, remains sequential). This choice allowed for very high fault coverage with very short, strictly pseudorandom, test sequences, without incurring significant overhead, since the aforementioned scannable flip-flops are only a small portion of the total flip-flops involved in a MU (the majority are data registers).

During each test phase, multiple MUs are independently tested in parallel. For example, the 4×4 router depicted in Figure 4.3 would have been tested in three phases. The first phase utilizes the shared buffers of the input Elastistores as TPGs and the shared buffers at the outputs of the first-level of MUs as response compactors. Before testing starts, all flip-flops of the tested circuits are reset. Due to the partial scan chain architecture, the generation of a new pseudo-random test vector requires a certain number of cycles (equal to the scan-chain length), so as to be shifted in the scan chains. Then, 1 clock cycle is needed to put the MU in normal mode and allow the circuit responses to be captured in the scan chains. In the same clock cycle, normal MU outputs are fed and compacted in the output MISR. Finally, the response captured in the scan chains is shifted-out and compacted to the MISR as well. This last operation is overlapped with the scan shift-in of the next test vector. Thus, the total number of clock cycles for generating, applying and compact-

ing a test vector is equal to *scan-chain length* + 1 (the "+1" term is for the capture cycle).

At the end of each test phase, a signature for all responses is stored in each MISR. To verify the results of the test session, this signature needs to be compared with the golden fault-free signature (computed off-line) of the applied test vectors and produce a final error bit. This comparison can be made serially, bit-by-bit, with a locally stored golden signature.

In the following two test phases, the intermediate ElastiStore shared buffers change role from MISR to LFSR and test the last MUs and their associated LRC logic, using exactly the same test sequence (the responses are captured at the output ElastiStores). The last test phase tests the output links that are connected to the inputs of the next routers via the two ElastiStores present at their endpoints. Since these three test phases can be applied simultaneously to all NoC routers, testing can finish in a few thousand cycles, as shown in Section 4.4. Gathering the error signals of each MU can be either done at the router level via a small local test controller, or they could be sent via 1-bit pipelined links to a centralized test controller. Depending on the number of pipeline stages per router, and based on the fact that the even-numbered stages can be tested independently from the odd-numbered ones, testing of ElastiNoC requires a constant number of 2 or 3 test phases overall, which is independent of the size of the network.

4.4 Experimental Results

In this section, we compare ElastiNoC with conventional VC-based architectures, both in terms of network performance and hardware complexity. We also report the fault coverage achieved by the proposed distributed BIST architecture and the required test application time, and we quantify the area/delay overhead of the self-testability features.

4.4.1 Hardware Complexity

The proposed ElastiNoC routers (using lookahead RC) were mapped (synthesized) to an industrial low-power 45 nm standard-cell library under worst-case conditions (0.8 V, 125 °C), using the Cadence digital

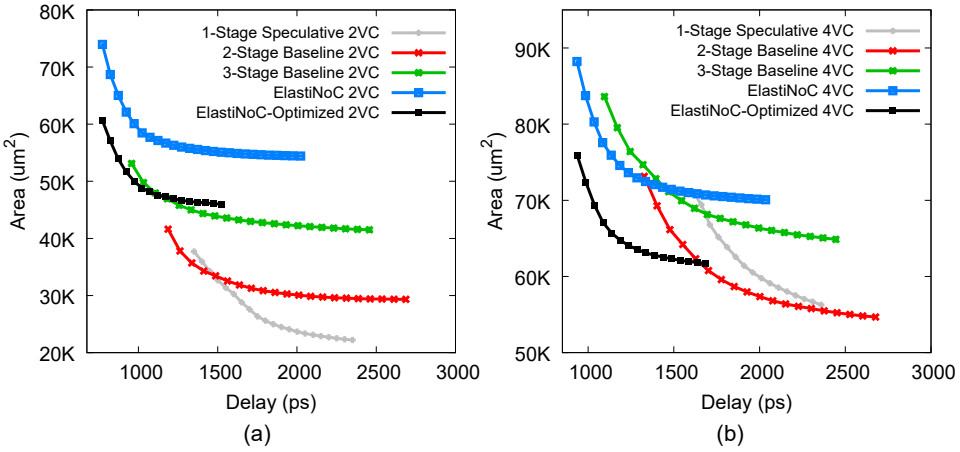


Figure 4.6: Hardware implementation results of various router designs with (a) 2 VCs, and (b) 4 VCs per port.

implementation flow. The generic router models have been configured to 5 input-output ports, as needed by a 2D mesh network, and to 2 and 4 VCs per port, while the flit width was set to 64 bits. Arbitration in all routers follow the fast arbiter design of [31]. The area/delay curves obtained for all designs - after constraining the logic-synthesis and back-end tools, and the extraction of physical layout information (each output is loaded with a wire of 2 mm) - are shown in Figure 4.6.

The routers under comparison for 2 and 4 VCs per port include an ElastiNoC design with 2 MUs in series per router, a speculative 1-cycle design that corresponds to the fastest monolithic design, as well as 2-stage and 3-stage pipelined baseline router implementations. In both cases (2 and 4 VCs), the proposed ElastiNoC design achieves the highest delay savings of 20% and 15% for 2 and 4 VCs, respectively, as compared to the fastest 3-stage pipelined baseline router. Please keep in mind that the delay numbers reported correspond to 0.8V operation, which increases significantly the delay of the circuits. For example, a close inspection of the clock frequency of ultra-fast 3-stage commercial routers optimized at the transistor level [119], [140], which offers additional benefits versus standard-cell-based design, reveals that their frequency marginally passes 1 GHz when operated at 0.8V.

For all cases regarding state-of-the-art routers, we assumed the minimum buffering requirement needed to cover the round-trip time imposed by their internal pipeline organization. The round-trip latency

of a single-cycle router is three cycles, translating to three buffers per VC, since each flit spends one cycle inside the router, one additional cycle on the link in the forward direction, while the back-pressure signals (such as credit updates) need one cycle on the link to return. Thus, the pipelined routers with two and three stages increase the round-trip latency by one and two, respectively – unless, direct combinational flow-control update paths are employed across routers, which limit the benefits of pipelining. As a result, the 2- and 3-stage pipelined routers need a minimum of four and five buffers per VC.

The amount of ElastiNoC buffering is between the buffers required for a 2- and a 3-stage router. While ElastiNoC requires larger area than monolithic routers for the case of 2 VCs, this trend changes in the case of 4 VCs. In this case, under equal delay, the proposed routers and especially the one that is optimized to the routing logic, depicted as "ElastiNoC-Optimized" save significant amount of area when compared to the 2- and the 3-stage routers, since it allows for both buffer and logic removal. The power consumption of the routers under comparison follows a similar trend.

Finally, we measured the hardware performance of ElastiNoC-Optimized and assuming that the packets entering the network are not allowed to change VC as done in [37]. This simplification saves more than 3% and 10% of the delay of ElastiNoC for 2 and 4 VCs per port, respectively, and lowers its area footprint by 12% and 15%. The expected drawbacks of such optimization are: (a) a reduction in throughput by increasing head-of-line blocking per static VC, and (b) complications in implementing adaptive routing.

4.4.2 Fault Coverage and Test Application Time

The synthesized MU netlists for 2 and 4 VCs were utilized for obtaining the self-testability results. The Hope sequential fault-simulator [77] was employed to compute the fault coverage (FC), while the LFSR TPG and the MISR compaction operations were simulated using a custom tool. The results of the proposed MU BIST approach are shown in Table 4.1.

Note that the exhibited FC has been calculated over all testable stuck-at faults of a circuit. A small amount of the total faults of each exam-

Table 4.1: Test coverage and test application results for a MU.

VCs	Scan FFs	Scan chains	Stuck-at FC	Test patterns	Test cycles	Aliasing
2	24	6	99.93%	302	1510	0%
4	78	13	100%	1642	11494	0%

ined MU (approximately 1-1.5%) have been reported as untestable by the Strategate sequential ATPG tool [52], and, as a result, they are not included in FC calculation. No deterministic test patterns have been used though for obtaining the reported results. The only purpose of ATPG was to determine the untestable faults. As can be seen, the proposed BIST approach achieves very high FC (complete in the case of 4 VCs) with quite short test sequences. The total NoC test time is network-size independent and equal to 3 (test phases) $\times$ 1510 = 4530 clock cycles for 2 VCs, and 3 $\times$ 11494 = 34482 cycles for 4 VCs. In these figures, a few extra cycles should be added for signatures comparison and error signal gathering. Our experiments showed that there is no FC penalty when modifying the scan chains volume; more and shorter scan chains can be used, when possible, for reducing test application time. Also, as expected with such wide MISRs (64 bits + the volume of ready/valid output signals), no aliasing was observed between the golden and the faulty circuits signatures.

After including the needed testability structures described in Section 4.3 in ElastiNoC and resynthezing the resulting designs, we end up with 22% increase, on average, in the total area for a 5$\times$5 router. The impact on the delay is a slight 7% increase, as compared to an ElastiNoC without any self-testability features. This area/delay overhead should be treated as an investment that pays off its purpose by offering fine-grained testability, and fault isolation at the MU level. Its cost can be amortized by increasing the flit width and the number of VCs. This happens since the extra cost involves mostly the logic of the shared buffer at each ElastiStore, which is constant irrespective of the number of VCs.

4.4.3 Network Performance

Network-performance comparisons were performed using a cycle-accurate SystemC network simulator that models all micro-architectural com-

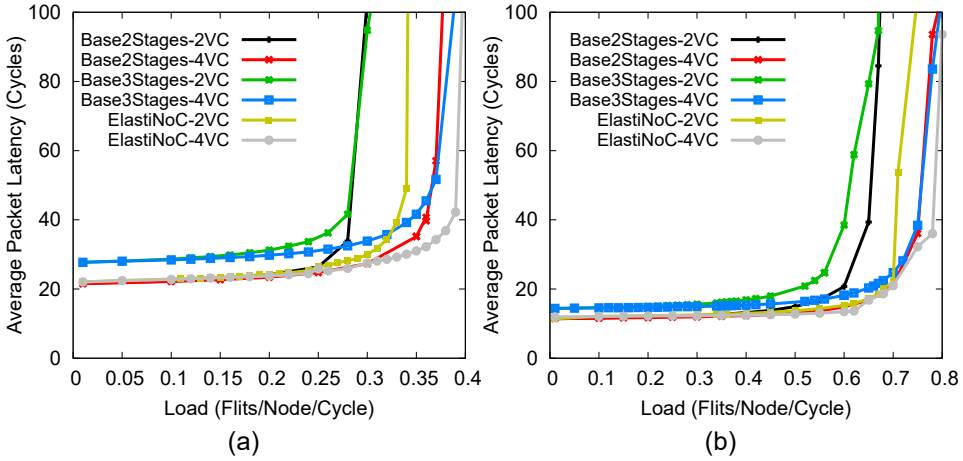


Figure 4.7: Latency vs. load curves for (a) uniform random traffic and (b) non-uniform localized traffic. Network traffic from real applications is estimated to lie in-between these two synthetic traffic patterns.

ponents of a NoC router, assuming an 8×8 2D mesh network with XY dimension-ordered routing. The evaluation involves two synthetic traffic patterns: Uniform Random (UR) and non-uniform Localized Traffic (LT). We estimate that network traffic from real applications would lie in-between these two synthetic traffic patterns. For LT traffic, we assume that 75% of the overall traffic is local (i.e., the destination is one hop away from the source), while the remaining 25% of the overall traffic is uniform-randomly distributed to the non-local nodes. We experimented with other distributions as well, but they all showed similar results. The injected traffic consists of two types of packets to mimic realistic system scenarios: 1-flit short packets (just like request packets in a CMP), and longer 5-flit packets (just like response packets carrying a cache line). For the latency-throughput analysis, we assume a bimodal distribution of packets with 50% of the packets being short, 1-flit packets, and the rest being long, 5-flit packets, in accordance to recent studies [83].

Figure 4.7 shows the latency-throughput curves as functions of the node injection rate, for the two aforementioned synthetic traffic patterns, and the same router configurations (in terms of numbers of supported VCs and their pipeline structure) used in the hardware complexity analysis. In all cases, the performance of the ElastiNoC routers is indistinguishable from the equivalent baseline routers, both at low and at high loads, while in some cases the performance of ElastiNoC is, in fact, better. The latency of the 3-stage pipelined router is higher, since it costs more cycles

to traverse each router of the network. For the 2-stage pipelined solutions that include ElastiNoC and the baseline router, keep in mind that even if the reported latency in cycles is equal, in reality it corresponds to different clock frequencies; ElastiNoC is at least 15% faster.

Multiple parallel physical elastic-buffer-based networks of simpler wormhole routers [89] (with each network mapped to one VC) would enjoy slightly higher clock frequencies, due to the complete removal of any VC allocation step. However, when compared with ElastiNoC routers under *equal network bisection bandwidth*, multiple networks would suffer in performance, as verified by our experiments, because of the high serialization latency imposed by the narrower channels in each physical network.

4.5 Conclusions

Virtual channels within NoC routers are quickly becoming a necessary ingredient of modern NoCs, and are viewed as instrumental in enhancing performance and offering several network- and system-wide services. In this thesis, we introduce the ElastiNoC architecture as the first NoC design that offers: (a) distributed implementation for VCs, including buffering, allocation, and necessary switching; (b) modular pipelined organization; (c) same (or even better) network performance, as compared to baseline monolithic VC-based architectures; and (d) a scalable self-testing mechanism that enables fine-grained fault localization (at the MU level) with small test application time.

Chapter 5

Multi-Bit Register Composition

5.1 Introduction

The NoC IP is physically distributed across all parts of the chip. This distributed placement of a certain IP of the chip creates additional burden to the clock tree that needs to drive its sequential elements. This phenomenon is especially critical when considering NoC buffers with very wide links. Multi-bit register (MBR) composition can reduce the complexity of the clock tree by reducing the number of clock sinks, thus shortening the clock tree's wire length, which decreases the wire capacitance. By sharing clock circuitry within the cell, MBRs also present a smaller pin capacitance load on the clock tree, compared to separate single-bit registers. Not only does this reduce the clock switching power at the leaf-level of the tree, but the reduced clock load allows a smaller clock tree to be used, with fewer and smaller clock buffers, further reducing the clock power. An example of the result of MBR composition is shown in Figure 5.1, where the registers of the original design are merged to fewer cells.

MBR composition must carefully select which registers to merge, to maintain the correct function and scan-connectivity. It needs to avoid degrading timing slack, wire-length, or routing congestion, while reducing clock power.

The proposed balanced restructuring approach targets MBR composition, after global or detailed placement, with the goal to (a) minimize

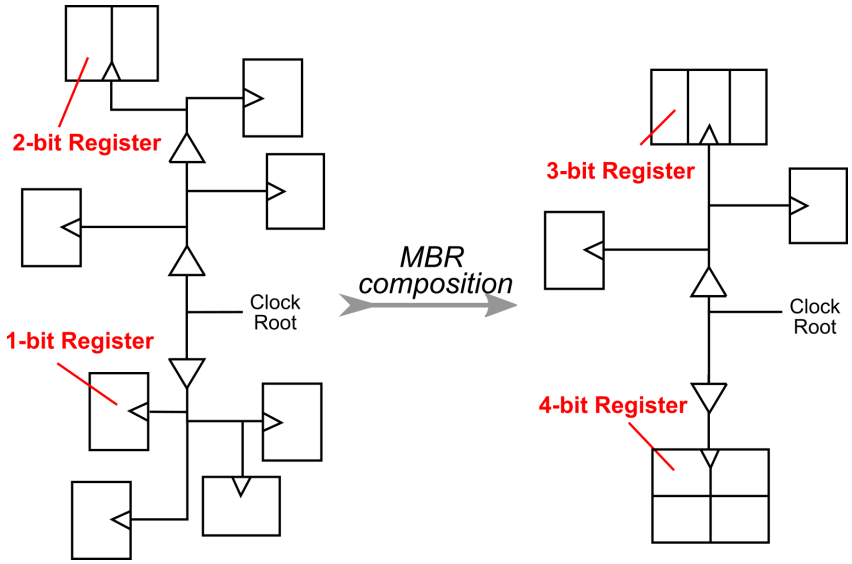


Figure 5.1: MBR composition reduces register count and simplifies clock tree synthesis by grouping registers to larger multi-bit cells.

the total number of registers in a design, (b) reduce clock power, and (c) simplify subsequent clock-tree synthesis (CTS). The proposed methodology equally applies to circuits that initially have only single-bit registers, or that are rich in MBRs identified earlier in the design flow.

MBR composition can be performed early in the flow, i.e., during logic synthesis [51, 147, 150] for register power reduction. Although early allocation of MBRs offers significant savings, it misses critical placement and timing information that affect the final result. For this reason, the majority of the work in MBR composition is focused on identifying MBRs after global or detailed placement.

In those cases, compatible registers are identified and grouped in MBRs with the goal of minimizing any wire length increase, timing degradation, and/or routing congestion [64, 60, 142, 131, 80, 139]. In most cases, the initial designs have only single-bit registers and do not consider any function or library limitations, which are standard restrictions in industrial designs. The composed MBRs are either limited to small sizes of 2 or 4, bits or they move to excessive sizes of up to 16 or 64 bits. In reality, quite a few of the registers may have no logically equivalent multi-bit version, or they may have been specified as fixed or size-only by the designer, and thus cannot be composed to MBRs. The main difference across the various approaches is the clustering or grouping

algorithm employed (clique partitioning, analytical or k-means clustering, force-driven bonding), and the selection of the placement window within which to search for compatible registers.

MBR composition has been also applied during placement, taking into account the effect of clock tree latency [79]. The late application of MBR composition narrows the design space to identify candidate MBRs. Each new choice requires incremental legalization and clock tree rebuilds, which results in long runtimes and can cause timing hotspots with the disturbance of the clock sink points.

Basic methods for MBR composition have been enriched with other features, such as the optimization of clock gating logic [81], data-driven clock gating [144], and crosstalk avoidance [53]. Recently, MBR composition has been extended to satisfy multi-mode multi-corner timing constraints, where the compatibility of registers is differentiated per mode of operation [78].

Even if registers are not replaced by MBRs their physical clustering can simplify the clock tree and reduce the buffering needed in the clock tree. In these cases, register banks are created in the layout after clustering nearby registers [95, 130, 145], with the goal being to create balanced clusters and minimize register displacement from its original position to the new position in the register bank.

In this work, MBR composition follows strict rules for identifying compatible registers that can be merged to MBRs. Candidate registers should be compatible in terms of functionality, timing, placement, and scan connectivity. Also, the registers replaced by a MBR should exhibit similar input/output slacks, thus enabling the application of the same useful clock skew after CTS.

To increase the possibility of identifying compatible registers and avoiding any timing incompatibilities, selected MBRs of the original circuit are decomposed and optimized, to facilitate higher quality MBR generation later in the flow. MBR composition uses a new weighted integer linear programming (ILP) formulation that offers significant reduction in the total number of registers with reasonable runtime. The weights assigned to each MBR candidate correspond to new simplified physical constraints that facilitate MBR detailed placement.

During MBR allocation, we allow incomplete MBRs, where some D/Q pin pairs are left tied-off/disconnected. This reduces register count, while later in the flow some of the unconnected pins are connected using an extra recovery step. After MBR composition, timing-driven MBR downsizing allows us to save additional clock pin capacitance. This further reduces the clock trees power consumption.

5.2 Overall Flow and Goals

MBR composition forms MBRs by grouping either single-bit flip flops or latches, or already existing MBRs composed during logic synthesis. The goal is to create larger MBRs, reducing the register count and simplifying the clock tree.

5.2.1 Goals of the MBR Composition Flow

When two or more registers are selected for merging, they are removed from the netlist and their nets are reconnected to the new MBR. The placement of the new MBR determines the wire length of the reconnected nets, and if not chosen appropriately, may cause timing violations. The candidate registers should have sufficient positive D/Q pin slack to allow them to reconnect to the newly formed MBR without introducing or increasing timing violations.

Any pre-existing MBR, or any newly formed one, should include pins that have similar input D-pin slacks and similar output Q-pin slacks. If the pins of one bit of a MBR have positive D/negative Q slack, and the pins of another bit exhibit negative D/positive Q slack, then those pins contradict possible useful clock skew assignment to the MBR. For example, considering the setup constraints, the pin with negative D slack favors a later clock arrival time, while the pin with negative Q slack prefers an earlier arrival [33].

Such cases of timing slack incompatibility should be avoided, either by disallowing candidate registers with incompatible timing profiles to be merged, or by decomposing existing MBRs with such characteristics to smaller MBRs. The pins assigned to each decomposed register (single- or multi-bit) can then be grouped according to their timing slack profile.

Additionally, routing congestion that may arise after MBR composition should not be overlooked. The generation of large MBRs brings many wires in the same region, thus possibly creating routing congestion problems in very dense placements. Given this, the availability of space and wiring resources should not be left as an afterthought, but should be included during MBR selection, mapping, and placement.

5.2.2 The Flow for MBR Composition

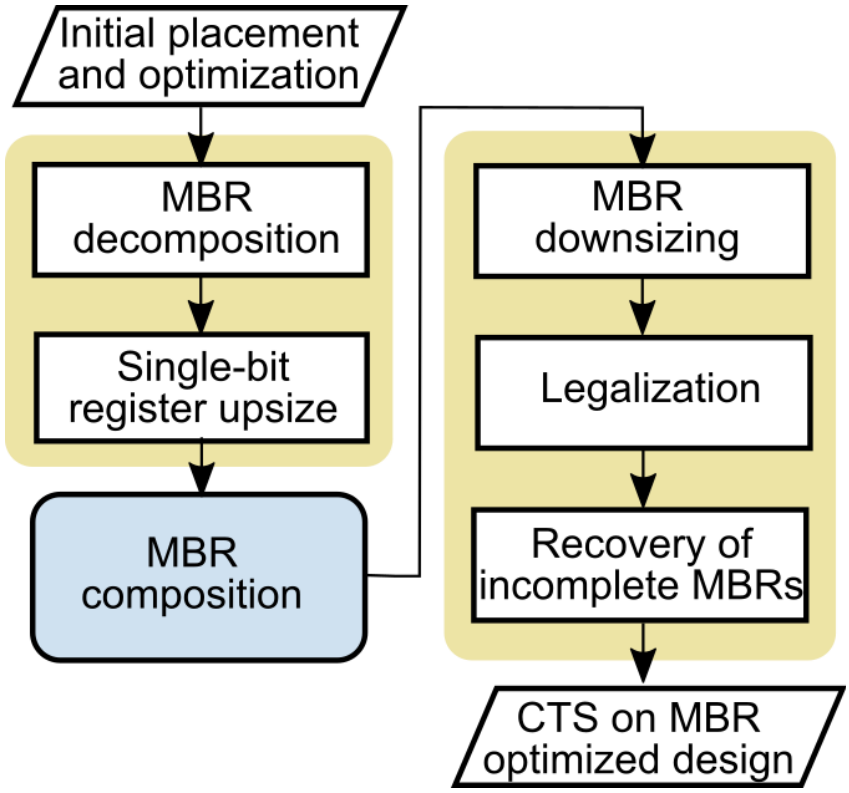


Figure 5.2: The proposed MBR composition flow follows a balanced restructuring approach that reduces the complexity of clock tree synthesis (CTS) without degrading timing, wirelength and routing congestion.

The proposed flow for MBR composition is depicted in Figure 5.2. After the initial placement and optimization, MBRs that have bits with positive D/negative Q slack, and other bits with negative D/positive Q slack, are decomposed to smaller MBRs or single-bit registers. Each of the resulting registers should contain bits with the same timing slack profile. In our example, all single-bit registers are upsized, thus increasing the probability of producing more efficient MBR mappings later.

The resulting circuit is then passed to the core of MBR composition. Compatible registers are identified, merged to new MBRs, and appropriately placed, ensuring that the impact on datapath timing, wire length, and routing congestion does not offset benefits of a lighter clock tree.

The composition flow permits the generation of incomplete MBRs, where some D/Q pin pairs are left tied-off/disconnected. Incomplete MBRs tackle the MBR bit-width granularity limitations in typical standard cell libraries, and help reduce register count. We ensure that the merging to incomplete MBRs does not negatively affect the area or leakage power. Although incomplete MBRs are used during MBR composition, they nearly all disappear after a final recovery step at the end of the flow.

Once MBR composition finishes, the MBRs are passed through a sequence of post-processing optimization steps that improve the overall result and simplify the clock tree synthesis that follows. The first step involves MBR downsizing, with the goal of reducing MBR area and clock pin capacitance without degrading timing. The circuit is then legalized to fix any placement violations produced during MBR composition, and redistribute the white space produced by the registers replaced by a MBR. On the legalized circuit, we perform one final optimization step that tries to use as many as possible of the incomplete MBRs pins by redistributing and reconnecting available nets from nearby registers.

5.3 MBR Decomposition and Optimization

Every MBR of the design that contains pins with different timing profiles is decomposed to registers of smaller bit-width. (Namely, for each MBR where a bits input-D/output-Q pins have positive D/negative Q slack, and another bits pins exhibit negative D/positive Q slack.) After decomposition, each one of the new registers can be either a single-bit register or an MBR, and includes pins with exactly the same timing profile.

During decomposition, we try to minimize the number of decomposed registers. For example, assume the case of an 8-bit MBR shown in Figure 5.3, where 5 pins exhibit positive D/positive Q slack, two pins negative D/positive Q slack and one pin positive D/negative Q slack. Ac-

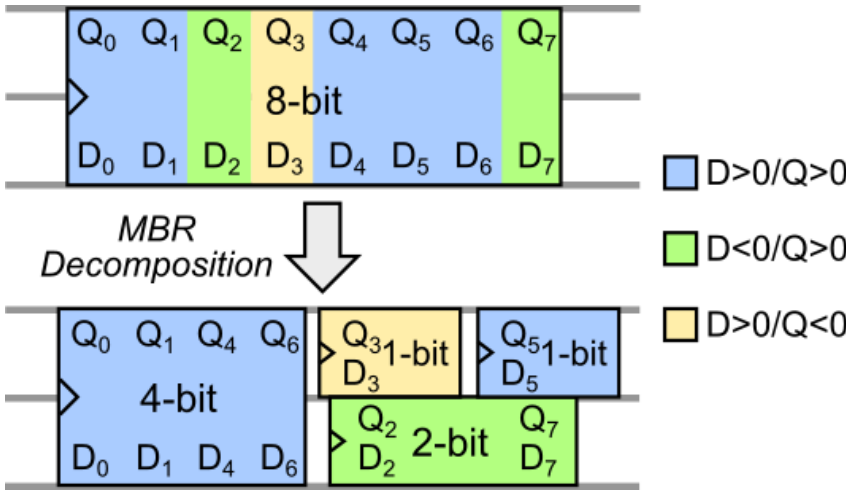


Figure 5.3: MBR decomposition of an 8-bit MBR with timing incompatible pins. Decomposition leads to four new registers either MBRs or single-bit registers that are placed in the position of the original MBR. In this example, the 8- and 4-bit MBRs are assumed to be two-row cells.

cording to Fig. 5.3, the 8-bit MBR that is implemented as a two-row cell, is decomposed to (a) one 4-bit MBR and a single-bit register for the five bits with positive D/positive Q slack, (b) a 2-bit MBR for the two bits with negative D/positive Q slack, and (c) one additional single-bit register for the bit with positive D/negative Q slack.

Inside each group, e.g., the group of five pins with positive D/positive Q slack, the separation to MBRs is done according to the available MBRs in the standard cell library and the Q slack of each pin. The pins are sorted according to their Q slack and then they are assigned in this order to the largest available MBR of the library. In this way, the pins with large values of Q slack are separated from the ones with less slack. Note that incomplete MBRs are not allowed in this step.

The new derived cells are placed temporarily at the position of the original MBR, while any useful clock skew properties applied by the designer on the original MBRs are transferred as is to each one of the decomposed cells.

After decomposition, the derived cells can be merged with other compatible registers producing more favorable MBR mappings. The total register count of the design is initially increased by MBR decomposition, but the final number of registers is significantly reduced relative to the original design.

After decomposition, we upsize to maximum size all the single-bit registers with negative Q slack. This improves their output-Q pin timing, which increases the probability of merging with other nearby registers during MBR composition, as detailed in Section 5.4. This upsizing only minimally affects the timing slack on the input D pin of the register, as verified by the experimental results.

On the other hand, upsizing the single-bit registers increases their clock pin capacitance. This overhead will later disappear, as those registers will likely be replaced by larger MBRs with less total clock pin capacitance than that of the original registers.

MBRs are not upsized at this stage, irrespective of their timing. As verified experimentally, the extra timing benefit that we would earn by upsizing MBRs at this stage does not pay off in reducing the total register count, or the total clock pin capacitance at the end of the flow.

5.4 MBR Composition

Even if a group of registers has an equivalent MBR in the library to replace them, they cannot be arbitrarily merged to new and larger MBRs. A group of registers can be merged to a larger MBR only if the registers are compatible in terms of functionality, scan chain organization, placement, timing profile, and drive strength.

Once the compatible registers are determined, an ILP-based optimization is formulated that selects which registers should be merged to MBRs. At this step, incomplete MBRs are considered as valid MBR candidates. The weight assigned to each MBR candidate corresponds to new simplified physical constraints that facilitate MBR placement legalization. Once the MBR candidates have been selected, they are mapped to specific library cells and placed after taking into account the position of the replaced cells and the wire-length.

5.4.1 Compatibility Checks

Registers can be merged to a new MBR only if there is one in the library with equivalent functionality. For example, a register with a reset pin can be replaced only if an MBR with a reset pin is in the library.

Similarly, scan flip-flops can be replaced only if scan-enabled multi-bit versions are available. Quite a few registers may have no logically equivalent multi-bit version, or they may have been specified as fixed or size-only by the designer, and thus cannot be composed to MBRs.

Registers are *functionally compatible* when they share exactly the same control pins, including clock and clock gating conditions. Many papers erroneously assume that any registers in the netlist are functionally compatible, maximizing the opportunities for MBR composition, but this is far from true for real industrial designs.

Scan compatibility dictates which registers are compatible, based on the scan chain definitions. Registers must be in the same scan partition, i.e., allowed on the same chain. MBRs may either have a single scan-in and scan-out pin, or multiple independent scan in/out pins. (The scan enable pin is still shared). In the first case, if the scan pins belong to the same scan partition then moving scan pins across different scan chains is allowed, and no additional constraints are imposed because of the scan chain definitions. However, for registers that belong to ordered scan chain sections, they may only be composed to a single MBR with an internal scan chain that preserves the same scan order within the MBR. In the second case, where MBR cells with separate scan pins per D/Q pair are used for composition, no restrictions are imposed as several scan chains with different constraints can cross the same MBR providing they have a common scan enable signal.

In the following steps, registers are checked for *placement compatibility*. For each register, a *timing-feasible placement region* is identified by transforming the positive timing slack of the input D and output Q pins to an equivalent distance that it can move without causing a timing violation.

Each register input (output) slack value defines a diamond. At the center of the diamond is the fanin (fanout) gate and its half diagonal is the equivalent distance. An example of the timing feasible region of a register is shown in Figure 5.4(a). We used Elmore delay for the timing slack to equivalent distance calculation similar to [78, 18].

Registers are compatible with respect to placement if their timing feasible regions overlap, as shown in Figure 5.4(b). The placement compatibility is checked on a global or detail placed design to give a realistic

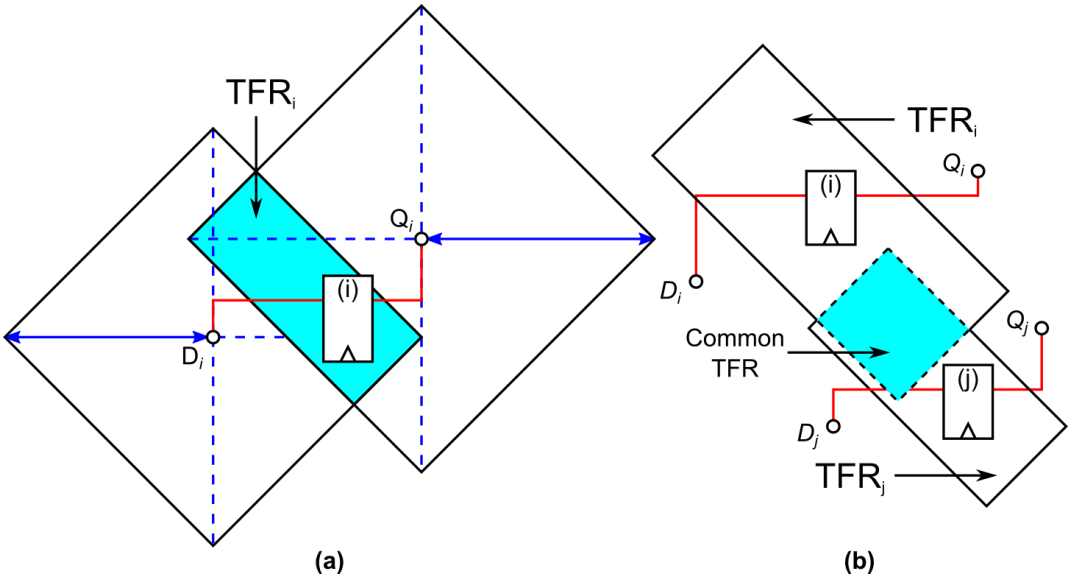


Figure 5.4: (a) An example of the formulation of timing feasible region (TFR) of a single bit register. (b) Two registers that their TFRs overlap and thus can be considered placement compatible.

sense of the relative placement of the registers under consideration for merging.

If the timing slack is negative, the feasible region is limited to the intersection of the bounding boxes of the violating pins with the feasible regions of the rest of the D and Q pins of the same cell. Even if a negative slack does not permit the cell to move, it is not left out of placement compatibility checking, as it has a timing feasible region that matches its footprint, to which other registers with positive slack can possibly move.

Next, *timing compatibility* is checked, to avoid merging cells that have positive D/negative Q slack with cells that exhibit negative D/positive Q slack. At this point, due to MBR decomposition, there is no MBR with such contradictory timing slacks. With this additional check, we ensure that we wont create new MBRs with this undesirable characteristic.

Even if the D/Q slack signs of two cells are the same, timing compatibility is preserved only if the magnitude of the observed slacks is similar. We should not merge registers with a large difference in timing criticality, because it increases power when a timing critical signal forces the MBR to be upsized, unnecessarily for the other signals. We must avoid

very different clock useful skew values, as only one can be realized for a given MBR, and the difference degrades useful skew opportunities for other timing paths to/from the MBR.

Finally, the last check is *drive-strength compatibility*. Two or more registers are considered compatible if their drive resistance differs by less than 3%. For drive resistance we refer to a linear model approximation of the registers delay as drive resistance multiplied by load capacitance, with some additional fixed "intrinsic" delay in the register. A cell with low drive resistance can drive more capacitance with less delay. In practice, we use accurate CCS standard cell library timing models.

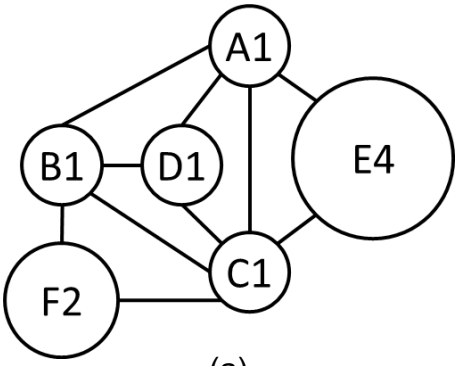
We need to avoid merging a high drive-strength cell with a low drive strength cell. If this happens, then the derived MBR should be of high drive strength in order not to degrade timing (implicitly the low-drive strength register is upsized when merged in the new MBR). However, this would increase significantly the MBRs area and power. By characterizing two registers with different drive strengths as incompatible we avoid such inefficient outcome

The only case that two registers are considered compatible, even if their drive strengths differ, is when a high-drive strength register is not timing critical (it has a lot of output-Q pin slack). In this case, when multiple registers are merged to a new MBR, the MBR can use the lower drive strength of the registers it came from. The high drive strength registers that participate in the new MBR are implicitly downsized, thus relinquishing some of their available slack.

5.4.2 MBR Candidate Enumeration and Incomplete MBRs

The compatible registers of the design are represented by the compatibility graph G. The graph nodes are the registers, whether single bit or pre-existing MBRs, and the edges of G reflect the compatibility between them, as shown in Figure 5.5(a).

A MBR can only be formed from registers that are all compatible with each other. Therefore, the registers that can be merged to a new MBR form a clique in G. For instance, the 4-node clique $\{A, B, C, D\}$ and the 3-node clique $\{B, C, F\}$ can each be mapped to a 4-bit MBR. By enumer-



(a)

Bit width	Possible groups for MBRs
2-bit	AB AD AC BC BD CD
3-bit	BF CF ABD BCD ABC ADC
4-bit	ABCD BCF
8-bit	AE (5-bit incomplete) AEC (6-bit incomplete)

(b)

Figure 5.5: (a) A compatibility graph of six registers, comprising ten bits in total. The compatible registers (nodes of the graph) are connected with an edge. Each register has a name and a size: A1 is a single-bit register; F2 is a 2-bit register; and E4 is a 4-bit MBR inserted during logic synthesis. (b) The possible groups for MBRs after clique enumeration.

ating all the cliques of G, we determine the set of candidate MBRs to consider during MBR composition.

During clique enumeration, a clique is considered valid if the number of register bits matches the size of at least one MBR in the cell library. For example, for a cell library that consists of 1, 2, 3, 4, and 8-bit MBRs, the 3-node clique $\{A, C, E\}$ that involves 6 register bits is invalid, since a 6-bit MBR is not available in the library. The clique $\{A, C, E\}$ is valid if it is allowed to map to an 8-bit MBR, which would be incomplete as only 6 out of the 8 D/Q bits are connected.

The table in Figure 5.5(b) lists all cliques for the compatibility graph of Figure 5.5(a), and the different bit widths of MBR cells that can be used for their mapping. Cliques $\{A, E\}$ and $\{A, E, C\}$ need 5 or 6-bit MBRs that are not available, but they can be mapped to an 8-bit MBR leaving some pins unconnected.

Incomplete MBRs may seem a waste of area and leakage power, but it can be advantageous as MBRs share the register control pins and associated logic. For example, replacing 7 single-bit registers, with 7 reset and 7 clock connections, with an 8-bit MBR that uses one reset and one clock wire saves 12 wire segments, even if one D/Q pin pair out of 8 is disconnected. However, MBRs with internal scan may not be suitable for this at the least, the first bit scan-input pin and the last bit scan-output pin must be connected to a scan chain.

Allowing incomplete MBR cells gives additional freedom to the MBR composition to minimize the number of registers. To keep the area and leakage overhead under control, we only consider an incomplete MBR as a valid candidate for MBR composition, when the area of the incomplete MBR does not exceed the area of the replaced registers multiplied by a selected overhead-allowance factor. Even if incomplete MBRs are used at this stage, the majority of them are fully utilized at the end of the flow, by reconnecting nets of nearby registers to the empty pins of the incomplete MBRs.

To enumerate all cliques of G, we first enumerate all maximal cliques of G using the Bron-Kerbosch algorithm [11]. For each maximal clique, we enumerate all the valid sub-cliques for the permitted bit widths per the MBR library cells using a dynamic programming approach.

The runtime complexity of maximal clique enumeration is $O(3^{\frac{n}{3}})$. This is not computationally tractable for large graphs.

Hence, G is partitioned to a set of connected components which are further decomposed to a set of sub-graphs using k-means clustering. The partitioning is driven by the register clock pin positions to maximize the clock tree power reduction achieved by MBR composition. Each sub-graph cannot exceed 30 nodes. Trying smaller bounds resulted in significantly more registers (less composition), especially with a bound of fewer than 20 nodes. Increasing the bound above 30 did not help, as the slight improvement cost too much additional runtime.

5.4.3 ILP Formulation

Clique enumeration defines the set $M = \{M_0, M_1, ..., M_k\}$ of valid MBR candidates. A register of the original design may participate in various MBR candidates. This attribute is declared via binary variables $a_{ij} \in \{0,1\}$, where $a_{ij} = 1$ if cell j participates to MBR candidate M_i, otherwise $a_{ij} = 0$. To identify which candidate MBRs are selected from among the MBR candidates, we add a binary variable $x_i \in \{0,1\}$; $x_i = 1$ when MBR M_i is assigned to replace the constituent compatible registers, else $x_i = 0$. When the register j is grouped in MBR M_i, and the corresponding MBR is selected, then both x_i and a_{ij} should equal one. The total number of registers is minimized by solving the following in-

teger linear program:

$$\textbf{minimize} \quad \sum_{i=1}^{|M|} w_i x_i$$

$$\textbf{subject to } \forall \text{ register } j : \sum_{i=1}^{|M|} a_{ij} x_i = 1, \qquad a_{ij}, x_i \in \{0, 1\}$$

The constraint added for each register j guarantees that each register will be part of only one MBR. The cost function of the ILP does not treat all MBR candidates equally. Each candidate MBR M_i is associated with a weight w_i; *the smaller the weight w_i, the more favorable* the choice of M_i.

5.4.4 Weights to Limit Wire-length and Congestion

MBRs, due to their multiple input and output pins, lead to wire concentration, increasing the possibility of local routing overutilization. To avoid this, we aim to spread the routing demand to nearby regions by penalizing (with appropriate weights) the composition of new large MBRs very close to other already formed MBRs. In this way, we implicitly handle the possible increase in routing congestion after MBR composition. Considering routing congestion explicitly in the ILP would require the addition of a routing utilization model or more constraints. However, our experimental results in Section 5.6 show that the weighting heuristic chosen to handle the MBR-specific routing demand is adequate for achieving our goal without increasing the initial routing congestion.

The weight assigned to each candidate MBR is based on the relative placement of the compatible registers that will be merged to this MBR. *The most favourable MBR candidate*, which receives the lowest weight, *is one that avoids any other closely-placed compatible registers* that do not participate in the clique and could belong in another composed MBR. This limits the probability that the nets of the two new MBRs cross each other, keeping routing utilization under control.

For each MBR candidate, we define a polygon formed by the corners of the participating registers. We compute the convex hull formed by the outer corners of those registers.

Figure 5.6 illustrates the test polygon that corresponds to the 4-node clique $\{A, B, C, D\}$ or the 3-node clique $\{A, B, C\}$, which produce respectively a 4-bit and a 3-bit MBR candidate. All registers of the $\{A, B, C, D\}$ clique are part of the test polygon and no other compatible register lies in the same region, so this choice is the most favorable and receives the minimum weight. The 4-bit MBR candidate has a clear area to be placed physically separate from any other MBR. The empty space, which will be available after removing the participating compatible cells, roughly defines the room to place the MBR. Even though this white space is not contiguous as required to place the MBR, placement legalization is simplified because no other register will be placed in the same area. It also reduces the displacement of non-register cells that exist in the same area registers are larger and often have higher placement priority, so smaller movement of fewer registers helps minimize the placement disturbance.

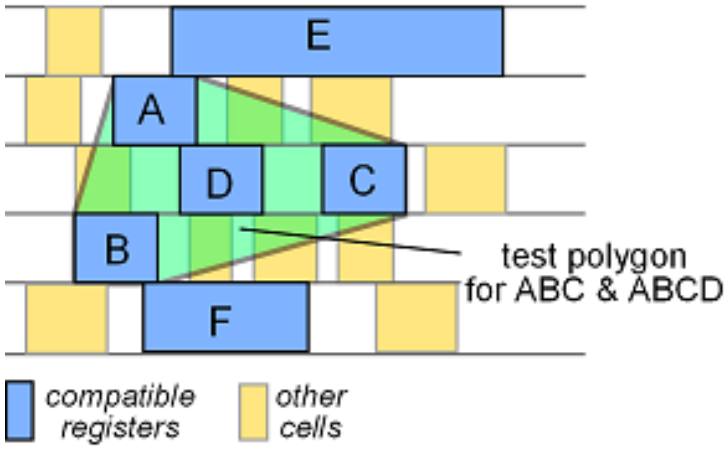


Figure 5.6: This shows the initial placement of registers in the Figure 5.5 compatibility graph. The size of each register corresponds roughly to its bit width (number of D and number of Q pins). To improve routability after mapping to MBRs, we check inside the surrounding polygon of the clique for the presence of other register. The fewer intervening registers, the more favorable the candidate MBR.

For the composition of a 3-bit MBR from $\{A, B, C\}$, we observe in Figure 5.6 that the polygon defined by the corners of A, B, and C includes register D. The composition of this 3-bit MBR is less favorable since register D may end up merging with another MBR that will be closely placed with the 3-bit MBR and increase locally the utilization of the routing resources.

By weighting appropriately each candidate MBR M_i, we promote the

composition of large MBRs, when the region defined by the constituent compatible registers is clean of other registers. When there are many intervening registers, we promote the selection of smaller, but clean, MBRs. This is achieved with a heuristic weight w_i for each candidate MBR M_i as follows:

$$w_i = \begin{cases} \frac{1}{b_i}, & n_i = 0 \\ b_i 2^{n_i}, & 0 < n_i < b_i \\ \infty, & n_i \geq b_i \end{cases}$$

b_i is the number of bits of the registers that will be merged to MBR M_i, and n_i is the number of other registers that block the convex polygon defined by the outermost corners of the registers replaced by M_i. To favour merging of registers, a weight of 1 is assigned to existing registers.

A register is a blocking register for M_i if its center is inside the corresponding test polygon and it is not a constituent register of M_i. For the example shown in Fig. 5.5, clique $\{A, B, D\}$ has $\{b_i, n_i\} = \{3, 0\} \Rightarrow w_i = 1/3$ since it is not blocked by any other register in Fig. 5.6, whereas, the clique $\{A, B, C\}$ has $\{b_i, n_i\} = \{3, 1\} \Rightarrow w_i = 6$ as the center of D is inside the polygon defined by the outmost corners of $\{A, B, C\}$.

When the test polygon for each candidate M_i is free of any other registers, the weight promotes the selection of larger MBRs. For instance, the weight of a clean 8-bit MBR is $\frac{1}{8}$, which is smaller than the weight of two clean 4-bit MBRs, i.e., $\frac{1}{4} + \frac{1}{4}$, needed to cover the same number of bits.

When there are obstacle registers the selection of large MBRs is penalized relative to the selection of more smaller MBRs. Large MBRs reduce the register count but can create routability problems when placed close to other MBRs and their large area can significantly increase the placement difficulty. Assume for example the case of an 8-bit MBR candidate that has one obstacle register, i.e., $\{b_i, n_i\} = \{8, 1\}$. In this case, the weight of this candidate would be $w_i = 16$. The equivalent choice with two smaller 4-bit MBRs would be to have one clean MBR with $\{b_i, n_i\} = \{4, 0\} \Rightarrow w_i = 1/4$, and another 4-bit MBR that includes the intervening register, $\{b_j, n_j\} = \{4, 1\} \Rightarrow w_j = 8$. The total cost of the second option would be equal to 8.25 that would make the ILP select

the two 4-bit MBRs relative to the one 8-bit MBR. It is more likely that the two 4-bit MBRs can be placed with reduced competition for routing resources with the intervening register. Large MBRs may reduce the register count but can create routability problems when placed close to other MBRs, and, their large area can increase the placement difficulty [19]. In future work, we plan to explore the possibility that instead of penalizing the MBR candidates with obstacle registers through an increased weight, to completely remove them from the candidate list.

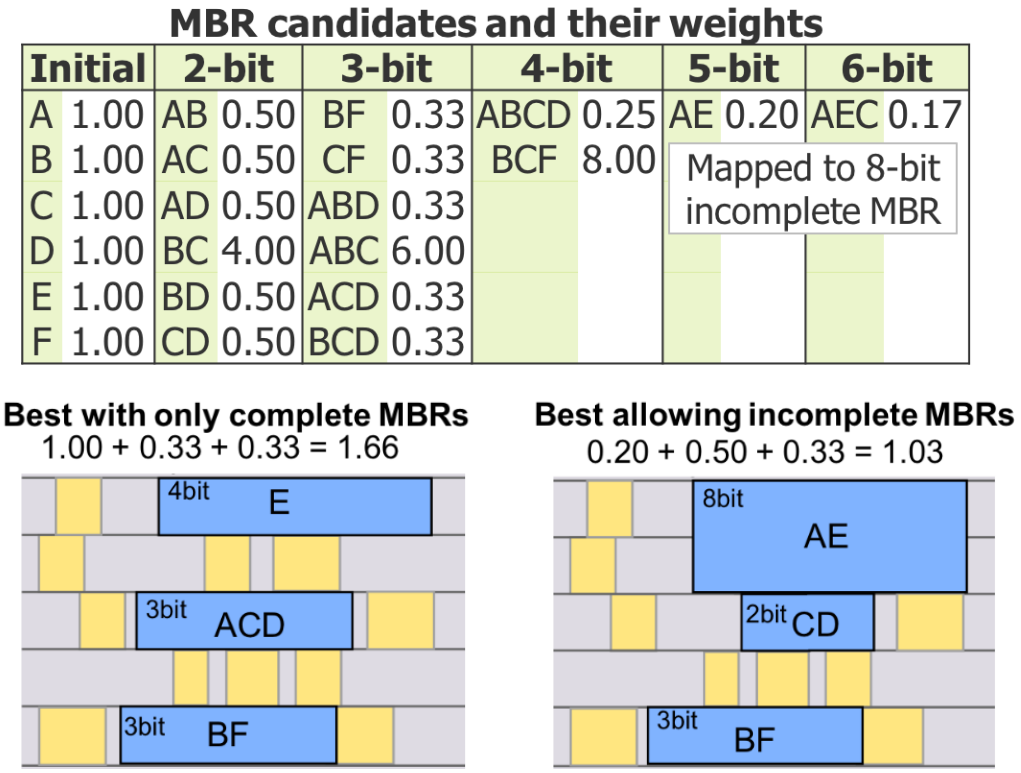


Figure 5.7: The weights of the candidate MBRs and the selected solution. The 5-bit and 6-bit MBRs can be mapped only to an 8-bit incomplete MBR.

Figure 5.7 summarizes the weights for the MBR candidates of the compatibility graph of Figure 5.5, with placement per Figure 5.6. When no incomplete MBRs are allowed, cliques $\{B, F\}$ and $\{A, C, D\}$ are mapped to 3-bit MBRs, while cell E is kept separate. This solution reduces the initial six registers to three. When incomplete MBRs are allowed, the same final register count is achieved with a different final outcome. Both choices in Fig 5.7 minimize the ILP cost function, and allow the three final MBRs to be placed in distinct regions without intersecting each other.

Before placement legalization, the composed MBRs may overlap with other non-register cells in the region, but the weights assigned to each candidate reduce the chance of overlapping with neighbouring MBRs. This example highlights the option of incomplete MBRs. In practice, the incomplete register AE will be rejected, as its area is significantly larger than the area of the registers it replaces.

Candidate MBRs involve both the initial registers of the design and the ones derived after MBR decomposition. As described in Section 5.3, the MBRs resulting after decomposition are placed at the position of the decomposed MBR. This initial placement decision does not limit the creation of large MBRs, as verified experimentally, since the test polygon for every candidate clique/MBR covers only compatible registers. The decomposed MBRs lead to some compatible registers and some incompatible ones (this is the reason why we decomposed in the first place). So the incompatible ones do not block the formation of larger cliques.

5.4.5 MBR Mapping

The ILP selects candidate MBRs that minimize the total number of registers and are less intertwined in the layout. For each MBR, the ILP selects just its bit width and the functional class of cells to which it belongs. Two further steps are needed for MBR assignment: MBR mapping and placement.

We must map the assigned MBR to a specific library cell. From the functional compatibility checks performed earlier, we know there is a compatible MBR in the library. From the available MBRs, we should select the one that best fits the timing and the drive strength profiles of the registers that it replaces.

The drive strength of the selected MBR should match the maximum drive strength of the registers that will be replaced by the MBR. This avoids degrading the timing of the design, but may incur an area and power overhead. However, since the registers to be merged are already drive strength compatible, the area overhead is avoided.

Registers with a high drive strength but a large timing slack (checked during drive-strength compatibility) dont determine the drive strength

of the new MBR. In this manner, those registers are implicitly downsized and their slack is reduced.

Any extra area paid depends on the difference of the drive strength between the composed registers versus how many control pins are shared by the MBR. To minimize clock power, we select the MBR with the lowest pin capacitance from the MBR library cells that closely match the drive strength of the registers to be replaced by the MBR. Due to the large variety of MBR cells in modern libraries, if the drive strength and the clock pin capacitance are not appropriately selected, they may cancel the benefits of MBR composition by creating significant timing problems or diminishing the clock tree power reduction.

MBR mapping also ensures that the scan chain definitions encoded as scan compatibility constraints are preserved with the lowest possible cost. MBRs with multiple scan in/out pins may seem attractive as their area and power are lower than their counterparts with internal scan. In reality, MBRs with multiple scan in/out pins incur the extra routing resource cost of the external scan chain connectivity. For this reason, MBR library cells with external scan chains are avoided during MBR selectionthey are typically selected only when there is no other alternative, or for mapping registers that are non- consecutive and belong to an ordered scan section.

5.4.6 MBR Connection and Placement

After mapping to the assigned MBR, we need to determine a location for the new cell and connect the input and output nets of the MBR to the many available pins.

We first assign nets to pins of the MBR. We topologically sort the replaced cells by their horizontal position, and connect the pins of the leftmost cell to the leftmost bit of the MBR. If these are registers on an ordered scan chain, the order of pin assignment is dictated by the scan order.

The new MBR is placed in the position that minimizes the length of the wires connected to its D and Q pins. To identify the best location, we use a linear programming (LP) approach. For each D/Q pin of the replaced registers, we identify their fan-in and fan-out pins to which

the MBR will connect, respecting the connectivity of the original registers. For all the identified pins, we create a bounding box and reference each pin's coordinate relative to the MBR's lower left corner plus some offset (dxi, dyi) for the pins location on the cell. The lower left (x, y) coordinates of the MBR are the variables to be determined by the LP. For the bounding box that corresponds to the input or output connections of each pin i, we use the half-perimeter wire-length to estimate the wire-length of the new wires. For each bounding box, the approximate wire-length wli is

$$\begin{aligned} wl_i = &(\max(x_h, x + dx_i) - \min(x_l, x + dx_i)) + \\ &(\max(y_h, y + dy_i) - \min(y_l, y + dy_i)) , \end{aligned}$$

where (x_h, x_l, y_h, y_l) are the coordinates of the box boundaries for each pin, and (x, y) are the coordinates of the MBR's corner. We use a linear program to minimize the wire-length of the D/Q pins of the MBR as follows:

$$\begin{aligned} &\textbf{minimize} \quad \sum_{i=1}^{|M|} wl_i \\ &\textbf{subject to } (x, y) \in \text{MBR's timing feasible region} \end{aligned}$$

The max and min functions in the objective are removed by the use of extra helper variables. For example, $\max(x_h, x + dx_i)$ is transformed to two inequality constraints $x_h \leq z$ and $x + dx_i \leq z$, while the opposite inequality is used for the min function.

Every new MBR replaces the set of merged registers, and its placement reuses the space freed by them. The drive strength compatibility check ensures that the area of the replaced cells is enough to contain the area of the larger composed MBR. However, the reorganization of this freed space for placing the new MBR causes the rest of the pre-placed gates in the same region to move slightly. As we verified experimentally, the legalization step that follows MBR placement, along with the incremental timing-driven optimization performed by default after legalization, manages to handle the movement of the rest of the gates without any true disturbance to timing, while preserving the desired wire length reduction.

5.5 Post MBR Composition Steps

After MBR composition, there are two further optimizations. As shown in Figure 5.2, these are MBR downsizing and the recovery of incomplete MBRs, with placement legalization in between. Downsizing reduces the clock pin capacitance and the area of the MBRs by sizing them down to the point that does not degrade timing. Recovery of incomplete MBRs better utilizes the unused pins of incomplete MBRs by locally merging compatible MBRs with incomplete MBRs.

5.5.1 MBR Downsizing

During MBR mapping, we selected for each new MBR the library cell that best matched the drive strength of the replaced registers, without performing any additional optimization that would trade-off timing slack with MBR area and clock pin capacitance. However, based on the timing profile of each register, significant clock pin capacitance and area can be saved by downsizing the non-timing critical MBR cells.

Algorithm 1 MBR downsizing

```
1: foreach register ∈ MBRs do
2:     prevTNS ← TNS; prevWNS ← WNS;
3:     while register is downsizable do
4:         Downsize register;
5:         Run Incremental STA;
6:         if prevTNS ≤ TNS && prevWNS ≤ WNS then
7:             prevTNS ← TNS; prevWNS ← WNS;
8:         else
9:             Undo downsize; Break;
10:        end if
11:    end while
12: end for
```

Downsizing the MBRs uses a brute-force approach, avoiding violating either the Total Negative Slack (TNS) or the Worst Negative Slack (WNS) of the design, shown in Algorithm 1.

For each MBR, we find the set of equivalent library cells. The cells are sorted in descending order based on their drive strength. We test the

cells with lower drive strength than the examined MBR. The MBR is down-sizable if there is at least one more cell with size smaller than the examined MBR that has not been tested yet. After every downsize, the TNS and the WNS generated by this change must not be worse than the TNS and WNS using the previous gate size. We stop searching when the MBR resize degrades TNS and WNS, in which case we keep the size from the previous round.

5.5.2 Recovery of the Unused Pins of Incomplete MBRs

After the placement has been legalized, we identify local compatible registers that can be merged with the incomplete MBRs to better utilize their disconnected pins. For example, if an 8-bit incomplete MBR with one empty bit is next to a compatible single-bit register, we can remove the single-bit register and connect its nets to the pins of the empty bit of the 8-bit MBR. This reduces both the number of incomplete MBRs and the total number of registers.

Algorithm 2 Recovery of incomplete MBRs

```
1: foreach register ∈ IncompleteMBRs do
2:     pins ← NumberOfUnusedPins(register);
3:     nearbyRegs ← Registers inside fixed size window;
4:     foreach s ∈ nearbyRegs do
5:         if !isCompatible(s, register) || NumberOfPins(s) > pins then
6:             Remove s from nearbyRegs;
7:         end if
8:     end for
9:     Sort nearbyRegs based on number of pins;
10:    foreach s ∈ nearbyRegs do
11:        if NumberOfPins(s) ≤ pins then
12:            Connect Nets(s) to the unused pins of register;
13:            pins ← pins  NumberOfPins(s);
14:            Remove s from the design;
15:            if pins == 0 then Break;
16:        end if
17:    end for
18: end for
```

The steps in the recovery of incomplete MBRs are shown in Algorithm 2.

At first, we find all the MBRs that have unconnected pins. For each incomplete MBR, we find all the compatible registers that are placed inside a small window around it. We consider only registers with bit width less than the empty pins of the incomplete MBR. In this way, we dont need to break down existing MBRs to reuse their pins; we can just reconnect the nets of the smaller nearby registers.

All candidate registers are sorted in descending order based on their number of pins. If two registers have the same number of pins, the register that is closer to the MBR is ordered first. Next, while there are empty pins in the examined MBR, we check if we can completely remove a nearby compatible register and connect its nets to the unconnected pins of the MBR. If all pins of the MBR are connected, we stop the procedure and move to the next incomplete MBR. Note that incomplete MBRs may still remain after checking all the nearby registers.

5.6 Experimental Results

The MBR composition methodology has been tested on six industrial benchmarks that are rich in MBRs after logic synthesis. Designs D1-D5 correspond to implementations at 28nm, while D6 is implemented in a 16nm process technology. The industrial designs used represent real use cases that were actually taped out. In all cases, the designs were optimized both in terms of physical layout density and timing. On average, the designs achieve an 80% layout density; above that, the designs are unroutable.

Our methodology aims to reduce the register count and clock tree capacitance, with only marginal disturbance to timing, wire length, and routing congestion. Before presenting the overall results for all designs, we would like to focus on two distinct cases that highlight how the proposed MBR composition flow performs under different scenarios. Two clock nets were selected in a placed design and CTS was performed on them, with and without the application of the proposed MBR composition flow. MBR composition was applied only on the registers driven by those clock nets.

The first case (Case A) is a clock net that drives 3571 registers, which

are mostly single-bit registers. This example shows the efficiency of the MBR composition step itself. Multiple compatible registers are composed to larger MBRs, allowing the use of incomplete MBRs.

The second case (Case B) is a clock net that has a similar fanout, consisting of 2795 registers, but a larger portion are existing MBRs formed in logic synthesis. This example highlights the power of MBR decomposition and optimization, combined with MBR re-composition, and sizing.

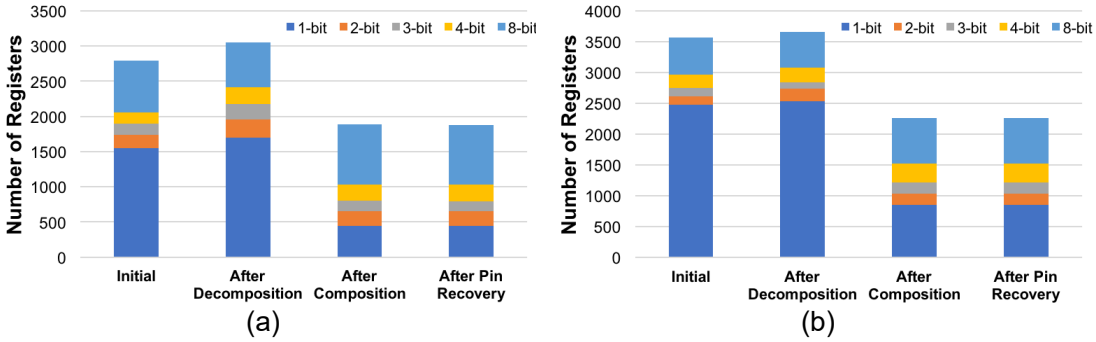


Figure 5.8: The initial distribution of register sizes, and after each major step of the proposed flow for the clock net of (a) Case A and (b) Case B.

The initial configuration of Case A is shown in the initial bar of Figure 5.8(a). The 3571 registers belong to several categories. The library used supports 1, 2, 3, 4 and 8-bit registers. After decomposition the register count increases to 3588 due to decomposition of the MBRs with timing incompatible pins. The MBR composition flow greatly reduces the total number of registers. The register count is reduced by 37% relative to the number of registers of the initial design. Single-bit registers are reduced to almost a third of the initial ones, and the number of 4 and 8-bit registers is increased accordingly.

Incomplete MBRs are allowed during MBR composition, provided that each incomplete MBR does not impose more than 5% area overhead relative to the area of the registers that it replaced. After MBR composition, there are 13 incomplete MBRs, all 8-bit MBRs. Following unused pin recovery, the final number of incomplete registers reduces to 9, reducing the total number of registers to 2229.

Note that the efficiency of the proposed MBR composition flow is actually higher if we take into account that 1192 of the initial 3571 registers

(almost a third) were skipped during the composition flow. They either do not match functionally to any MBR library cell, or there is no larger MBR in the library with the same functionality. From the 3571 initial registers, only 2159 of them were composable, which were reduced to 1037 registers (2229 final registers, minus the 1192 skipped ones), corresponding to more than 50% register reduction.

Similar results are derived for Case B, as shown in Figure. 5.8(b). With more MBRs initially, MBR decomposition touched more registers, and the 2795 initial registers increased to 2814 after decomposition. Despite this increase, MBR composition and unused pin recovery significantly reduced the total number of registers. After the application of the flow, the final number of registers is 1862 (with 21 incomplete MBRs), corresponding to 33% register savings. Again in this case, not all registers were composable. Case B included 962 non-composable registers. Subtracting these from the register count, the number of registers saved by the proposed method was 50%.

Table 5.1: The properties of two example clock nets of the initial design and their properties after the application of the proposed MBR composition.

Properties	Case A		Case B	
	Gnitial	Proposed	Initial	Proposed
Max Leaf Levels	10	9	9	9
Rise/Fall Avg. Latency (ps)	742	748	718	622
Rise/Fall Avg. Clock Skew (ps)	106	103	100	102
Clock Wirelength (mm)	32.0	31.1	34.2	31.5
Buffer Count	287	229	242	224
Total Capacitance (pF)	8.9	7.8	9.1	8.1

The register reductions achieved directly translate to power, buffer, and wire length savings in the clock tree network. Table 5.1 summarizes the results achieved for Cases A and B after applying CTS on the initial clock nets and on those restructured by the proposed MBR composition flow.

Clock tree capacitance, wire length, and buffer count are all reduced in both cases examined. In Case A, the savings are higher due to the larger

number of single-bit registers that allowed the MBR composition to explore more efficient MBR allocation choices. For example, clock tree capacitance and buffer count were reduced by 12% and 20%, respectively. In Case B, the composable MBRs were fewer and already had a larger bit width, limiting the available improvements for the proposed method. However, in addition to the 7% reduction in buffer count, measurable savings are still observed in clock tree capacitance and wire length (11% and 7.8%, respectively). The clock skew is also improved, by reducing the number of levels in the clock tree from 10 to 9 in Case A, and by reducing the clock wire length in both cases.

The clock leaf number and the clock pin capacitance reduction typically translate to lower clock latency. However, there are cases where the clock latency does not exhibit the expected reduction, due to the MBR placement algorithm that minimizes the total, and not specifically the clock tree wire length. As a result, MBRs might move away from their clock driver. In the case where the clock leaf defining the clock latency moves even further, a slight latency increase is observed. The same applies to clock skew, which is typically reduced; there are cases where slight increases can occur, due to the non CTS-friendly placement.

The MBR composition flow shown in Figure 5.2 achieves similar improvements when applied across all the clock nets of the industrial designs. Also, we did not apply any additional useful skew offsets in our flow for these results.

Table 5.2 summarizes the register counts of the initial and the MBR-restructured designs. Even in designs with a large initial portion of MBRs, many registers were replaced by larger MBRs. The average savings in the total number of registers achieved by the proposed method is almost 30%. The largest savings are observed in design D2, which had the largest initial number of single-bit registers, while the smallest savings are observed in design D4, where the majority of the registers were already large 8-bit MBRs. The D6 number of single-bit registers actually increased, due to the initial MBR decomposition, but this permits more register composition with compatible slacks, resulting in an overall register count reduction of 20%.

The reported savings correspond to a significant improvement after tak-

Table 5.2: The initial distribution of register sizes and after the proposed flow.

Initial	D1	D2	D3	D4	D5	D6
1-bit	21070	33918	19561	15478	19561	34938
2-bit	1116	0	1453	3513	1453	4771
3-bit	995	0	1726	1619	1726	2463
4-bit	1196	3483	2409	4180	2409	48441
8-bit	5445	0	9370	25148	9370	386
#Registers	29822	37401	34519	49938	34519	90999

Proposed	D1	D2	D3	D4	D5	D6
1-bit	7739	12187	5202	9622	6446	36105
2-bit	1576	1297	1340	2928	2445	3984
3-bit	1226	1000	1393	1967	2213	3225
4-bit	1842	2844	2220	4781	3785	9393
8-bit	6402	1892	10761	25466	9390	19608
Incomplete	247	562	806	165	619	78
#Registers	19032	19782	21722	44929	24898	72393
Savings	36%	47%	37%	10%	28%	20%

ing into account how constrained the application of MBR composition is in real industrial designs that are rich in MBRs after logic synthesis. For example, in all examined cases, the compatibility graphs were extremely fragmented, due to the strict compatibility constraints. Each compatibility graph contained multiple independent subgraphs with sizes fewer than 10 nodes (recall that a node can correspond to an already formed MBFF during logic synthesis). These small subgraphs were more than 85% of the total subgraphs and covered around 60% of the total register bits of the designs. MBR candidate enumeration and the ILP-based MBR assignment combine to get the most out of these small subgraphs. In larger graphs (less than 5% of the subgraphs consist of more than 30 nodes), the proposed approach achieves equally good results, but with increased runtime.

Several other design characteristics of the initial and restructured designs are shown in Table 5.3. By reducing the number of registers, MBR composition also reduces the complexity of the clock tree. Clock tree

Table 5.3: Industrial designs characteristics before (Init) and after the proposed MBR composition flow (New).

Design		Total Area (μm^2)	Total #Cells	Total Wire-length (m)	WNS (ps)	TNS (ns)	Clock skew (ps)	Clock Wire-length (mm)	#Clock Buffers	Clock Cap (pF)	Over flow Edges
D1	Init	598015	498434	14.19	47	1.82	176	669	2218	91	3
	New	596903	487646	14.01	42	0.48	194	613	1708	81	0
	Save	0.19%	2.16%	1.27%	10.64%	73.46%	-10.51%	8.37%	22.99%	10.99%	100%
D2	Init	952021	853912	16.67	719	865.33	196	453	3296	106	4
	New	943771	836690	16.57	616	869.18	181	416	2911	90	1
	Save	0.87%	2.02%	0.60%	14.33%	-0.45%	7.63%	8.17%	11.68%	15.09%	75%
D3	Init	1118762	666540	23.12	585	44.73	221	631	2825	132	382
	New	1118056	655151	23.76	665	69.75	189	603	2338	120	192
	Save	0.06%	1.71%	-2.77%	-13.87%	-55.93%	14.45%	4.44%	17.24%	9.09%	49.74%
D4	Init	2801693	1995640	65.04	2562	9018.63	264	1230	6459	272	46
	New	2800580	1984050	63.59	2548	9490.52	282	1219	6332	272	54
	Save	0.04%	0.58%	2.23%	0.55%	-5.23%	-6.82%	0.89%	1.97%	0.00%	-21.74%
D5	Init	1158447	693565	24.79	505	85.75	335	436	1875	92	354
	New	1152705	683582	24.80	435	42.20	233	417	1671	83	199
	Save	0.50%	1.44%	-0.04%	13.86%	50.79%	30.30%	4.36%	10.88%	9.78%	43.79%
D6	Init	653013	1188878	28.43	127	118.12	126	478	2762	192	0
	New	649263	1165887	28.49	109	77.51	126	437	2654	172	0
	Save	0.57%	1.93%	-0.21%	14.17%	34.38%	0.00%	8.58%	3.91%	10.42%	-

capacitance is reduced by 9.22% and buffer count is reduced by 11.45%, resulting in a similar reduction in clock power.

Clock and total wire length of the design is also reduced, due both to fewer registers and the wire-length-minimization-driven MBR placement. Note that in designs rich in MBRs, the clock wire length is a smaller percentage of the total wire length. The reduction of registers also led to a 0.37% and 1.64% average reduction in the total area of the designs and the total number of cells, respectively.

Although we perform significant circuit restructuring, on average we dont increase the timing violations, as highlighted by the worst negative slack (WNS) and the total negative slack (TNS) of the presented benchmarks. On average, we reduce both WNS and TNS by 6.64% and 16.17% respectively. The only large discrepancy appears in design D3, where MBR composition leads to an increase in timing violations.

Design D3 includes several highly dense regions that also include timing-critical (start) endpoints that span across many paths of the design. When composing new MBRs in such dense regions, the rest of the cells in the region are slightly moved, even if they dont participate in the

newly-formed MBRs. How far the rest of the cells move from their original positions depends on the internals of the detailed placement engine, and how it prioritizes each cells type (sequential or combinational), its area, or its timing criticality. In the case of D3, the slight displacement of timing-critical cells that affect many timing endpoints (mostly single-bit register cells that didnt participate in MBR composition) causes a cumulative effect that increases the TNS of the design.

There is no timing degradation if we avoid MBR composition in regions with excessive cell density. However, this approach leads to fewer composed MBRs, and to more registers in total at the end. The results given in this thesis do not take such precautions, and every region (independent of its utilization) participates in MBR composition. This fully unconstrained approach decreases the total number of registers to 21722 (37% savings - initially there were 34519 registers) with a 50% TNS overhead relative to the initial design. When we skip the registers that belong to regions with more than 95% cell density, then the total number of registers becomes 22467 (34% savings relative to the initial design) while setup timing (WNS and TNS) is not degraded. Further decreasing the density cut-off threshold, as tested with additional experiments for 90% and 85% densities, increases register count without a noticeable difference in TNS.

A similar trend is observed regarding the final clock skew. The majority of the simplified clock trees produced after MBR composition demonstrate improved behavior with respect to clock skew, except in D1 and D4, where clock skew is marginally worse, due to an unfavorable placement of the MBR cells.

In this work, we do not aim at reducing the overall routing congestion of the design. Rather, our goal is to not degrade the global routing congestion profile of the initial design after applying MBR composition to reduce the number of registers and simplify CTS. This goal is achieved, as shown by the results reported in the last column of Table 5.3. The difference in overflow edges [121], without and with our MBR composition methodology, is marginal due to the placement-aware weight selection for candidate MBRs in the ILP formulation.

In addition to the reduction of the number of overflow routing edges, as

depicted in the rightmost column of Table III, the restructuring of the designs does not globally alter the routing congestion profile of their critical edges. The routing edges with high utilization (demand/capacity ratio) receive less utilization after MBR composition and detailed placement, while routing edges with low utilization are additionally loaded.

This behavior is highlighted in Figure 5.9(a) for all routing edges of D1. Each bar of Fig 5.9(a) corresponds to the edges of the initial design that exhibit similar utilization, i.e., all edges that exhibit up to 0.1 utilization are covered in the leftmost bar of Figure 5.9(a). For each of the initial utilization groups, Figure 5.9(a) records the average percentage of change (positive or negative) to their utilization after MBR composition. Following the presented data, it is evident that the routing edges that initially exhibited low utilization have increased their utilization, and the ones with high utilization are less stressed after MBR composition. The same conclusion can be derived by the data presented in Figs 5.9(b) and 5.9(c) for designs D2 and D3, respectively, while the rest of the designs exhibit similar behavior.

To isolate the routing congestion near the regions that include a MBR, we identified the routing edges that pass on top of or next to multi-bit registers in both the initial designs and the designs derived after applying the proposed MBR composition flow. For each edge, we recorded the routing utilization in the initial and final designs. The histogram of utilization for those routing edges is depicted in Fig 5.10 for designs D1, D2 and D3. In all cases, the number of routing edges that are close to MBRs increases (due to the additional MBRs composed by the proposed flow), but their utilization remains under control, following the same trend depicted for all the routing edges of the design in Figure 5.9.

Finally, Figure 5.11 highlights the runtime of each part of the MBR composition flow shown in Figure 5.2. The flow was executed on an Intel Haswell server operating at 2.7 GHz with 512 GB of RAM. In all cases, MBR composition corresponds roughly to half of the runtime of clock tree synthesis (CTS), with the most expensive parts being the mapping of the selected MBRs and their legalization. The solution of the ILP, although consuming a non-negligible part of the overall runtime, is not the bottleneck for the application of the proposed MBR composition. On the contrary, the proposed enumerative ILP- driven methodology appro-

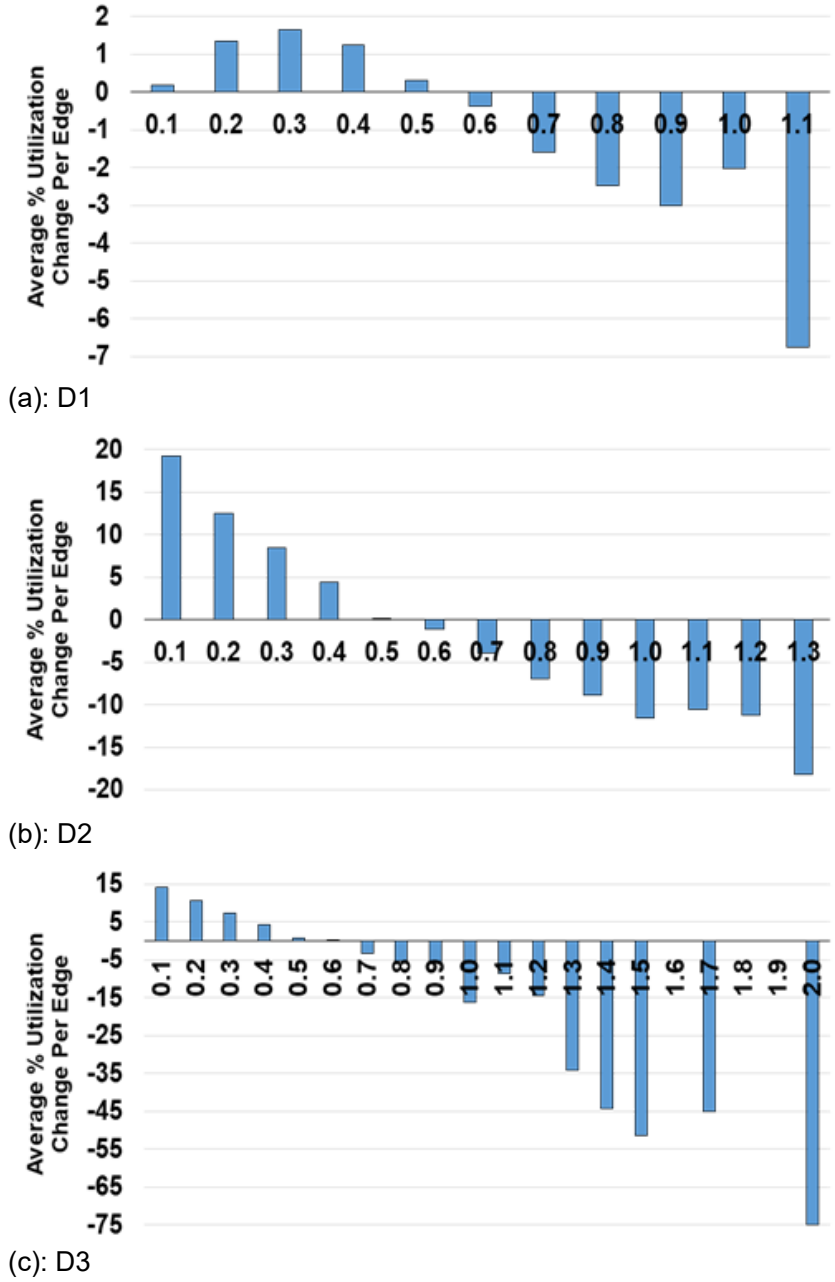
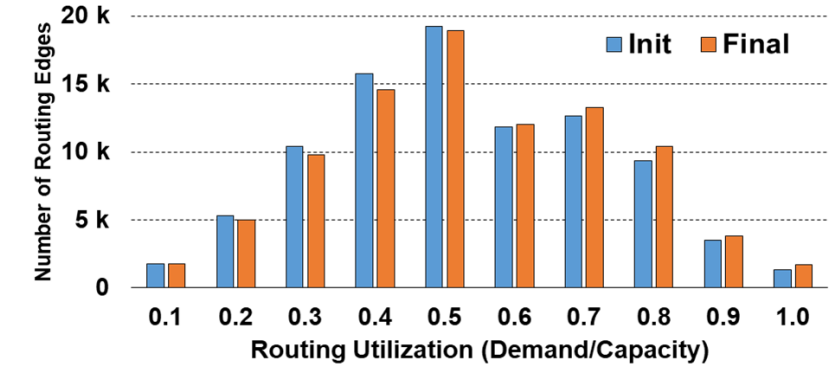
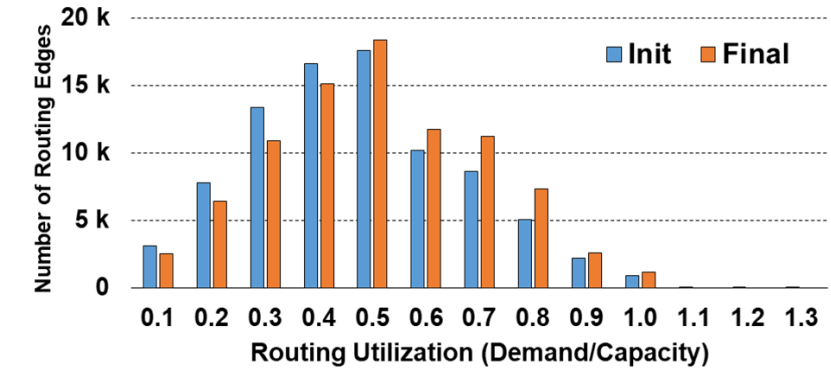


Figure 5.9: The average percentage of change in the utilization per edge for each group of edges that initially exhibited similar utilization (globally). After MBR composition, initial low-utilization edges receive more load, thus increasing on average their utilization, while edges with high utilization on average receive less load.

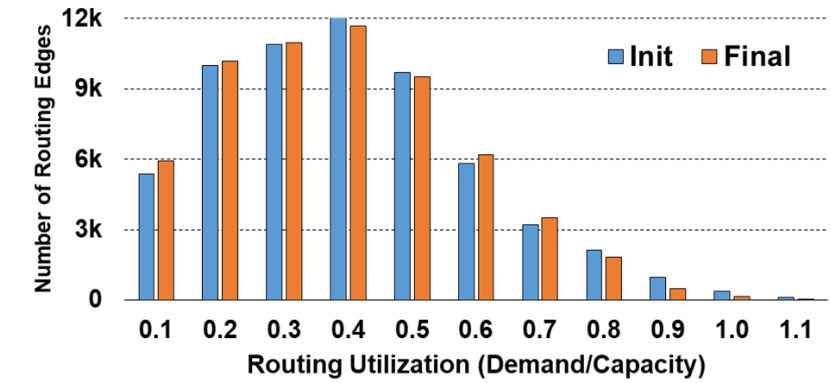
priately handles both the many small compatibility subgraphs and the small number of larger subgraphs, while significantly reducing the total number of registers.



(a): D1



(b): D2



(c): D3

Figure 5.10: The histogram of the utilization of all routing edges that are close to MBRs in the initial designs and final designs after applying the proposed MBR composition flow.

5.7 Conclusions

Applying MBR composition on industrial benchmarks requires a balanced restructuring approach. In addition to the reduction in the number of registers and clock tree capacitance, it should also keep the potential degradation in slack, wire length, and routing congestion under

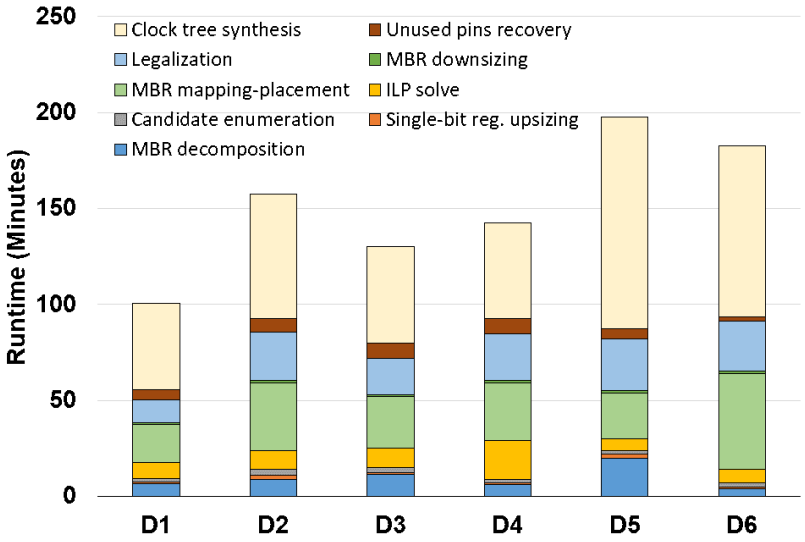


Figure 5.11: The runtime of all the intermediate steps involved in the proposed MBR composition methodology including also the runtime of CTS.

control.

In this work, we present a complete MBR composition flow that explores almost every aspect involved in the use of MBRs during physical design. MBR decomposition is introduced to partially alleviate the timing incompatibilities derived after the placement of the original netlist. Registers are then merged to form larger MBRs employing an ILP-based optimization that uses new and realistic rules that determine register compatibility. The ILP has a weighted selection of the best MBR candidates to facilitate their legalization and reduce contention for local routing resources. We permit incomplete MBRs to achieve additional MBR composition, but ensure that this is not detrimental for area or leakage.

The combined effect of these steps significantly reduces register count and, together with the timing-driven sizing of the MBRs, effectively reduces clock pin capacitance. The benefits are shown across six industrial benchmarks, demonstrating the effectiveness in producing a lighter clock tree without degrading timing or increasing routing congestion.

Chapter 6

Peak-Power Traffic for Networks-on-Chip

As technology scales, it enables digital system designs with billions of transistors integrated on a single chip. Besides the abundance of resources, which has been the driving force behind the multicore archetype, power constraints can rule key (micro-)architectural decisions. Keep in mind that excessive power dissipation increases packaging/cooling costs, reduces battery life in mobile devices, and adversely affects hardware reliability, primarily due to elevated temperatures [117]. The increasingly stringent requirement to adhere to a given power budget has rendered power consumption a first-class design constraint [9]. Hence, it is imperative for system architects to understand and accurately quantify their design's power usage from the early stages of the design process [70].

One particular salient attribute is of paramount importance: the peak (i.e., worst-case) power consumption [61, 73]. Both the system's maximum performance and implementation costs (power delivery, packaging, and cooling) are directly impacted by this worst-case power consumption. A pessimistic peak power estimate will unnecessarily curtail performance, while an optimistic estimate could potentially lead to reliability problems.

Designers should be able to identify accurately the worst-case power consumption. With the increased size of chips, it is an extremely chal-

lenging problem, as the worst-case power usage of a system is not simply the sum of the maximum power of each component, due to under-utilization and contention for shared resources. Hence, the peak power consumption must be estimated using a stimulus that is *realistic* and resides within the functionally feasible workload space of the system under evaluation. Typically, designers rely on hand-crafted, custom-made so called *power viruses* (also referred to as "stressmarks," or "powermarks") to estimate a system's peak power consumption [35, 111, 36]. However, the task of manually constructing a program for a specific architecture is very cumbersome and error-prone. Most importantly, the generated virus is not guaranteed to yield the maximum possible power usage. Thus, automatic approaches to the generation of effective power viruses are highly desirable.

Power viruses have been explored within the realm of CPUs, main memory, and off-chip I/O. One notable absentee is the on-chip network; there is currently no methodology to identify the peak power consumption in the system's Network-on-Chip (NoC) backbone. Given the NoC's functional and performance criticality, it is obvious that peak-power analysis cannot ignore such an elemental actor.

Merely summing the maximum possible link power consumption and the maximum possible router consumption would result in a *fake* worst-case estimate that would be excessively pessimistic. Salient functional characteristics of the NoC – such as the employed routing algorithm and the network topology – prohibit the simultaneous full utilization of every NoC component at the gate level.

The main goal of this work is to present a high-level systematic methodology to generate realistic traffic patterns that cause peak power consumption within the NoC. High network utilization alone is not enough to generate peak power, because, if the data payloads happen to be "favorable" (i.e., yielding low switching activity), the NoC power consumption may be quite low. Moreover, high network utilization is often confused with highly congested networks. Congestion may severely affect certain NoC regions, but may leave other regions under-utilized. Hence, formulating the generic problem of peak power consumption within the NoC involves the intricate interaction of all aforementioned nuances, which is one of the key contributions of this work.

The proposed framework can generate a peak-power "traffic virus" for any NoC configuration. State-of-the-art NoC architectures follow regular, or irregular, topologies with homogeneous, or heterogeneous, link widths, while their constituent components (routers, links, and network interfaces) may belong to arbitrary clock and voltage domains. Consequently, the various components contribute differently to the overall power consumption of the NoC. The proposed approach handles all the constraints that arise from the heterogeneous nature of modern NoCs and generates the peak-power traffic patterns using a novel formulation based on Integer Linear Programming (ILP), which executes in a reasonable time, even for very large networks.

Even if the proposed methodology constitutes a high-level approach, and it does not formally guarantee peak power consumption at the *gate level*, it still tackles efficiently the problem of verifying – at design time – the NoC's peak power consumption. Extensive and detailed experimental evaluations using various NoC configurations validate the effectiveness of the proposed methodology. The automatically generated traffic patterns are demonstrated to cause an average of 4× higher power consumption than randomly selected traffic patterns, or patterns that result in high saturation throughput. Conversely, the proposed methodology reveals that the realistic peak power of the NoC is 3× lower than the peak power reported by pessimistic scenarios that assume additional switching activity outside the functional workload space of the NoC. In this way, the true bounds of the NoC's peak power consumption are highlighted, thus preventing unnecessary over-design steps that tackle un-realistic worst-case scenarios.

6.1 Peak-power Traffic Characteristics and Problem Formulation

Peak power consumption is achieved when the system is operating at its maximum potential. Thus, the optimization goal when developing power viruses is to maximize system activity and increase dynamic power consumption. The dynamic power consumption of a NoC jointly depends on (1) the clock frequency and the supply voltage of its components, (2) the network component utilization that governs also the

activation of any clock-gating logic, and (3) the data switching activity caused by the traffic flowing inside the NoC every cycle. A network component that carries data – e.g., the links, the buffers, the crossbar, etc. – is considered utilized, as long as it performs a useful operation in each cycle, while the amount of power consumed is directly proportional to the switching activity caused by the bits of the traversing flits. On the contrary, the dynamic power of the control portion of the NoC does not depend on the bit-level profile of the data, but only on the per-cycle activity of the arbitration and flow control logic. In state-of-the-art NoCs employing wide links (128 bits, or more), the dynamic power of the control logic is minimal compared to the dynamic power of the datapath components [2, 63]. Therefore, in this work, we aim to trigger the maximum possible power consumption in the *datapath* components of the NoC, which are responsible for the majority of the total power consumption. This goal is achieved by maximizing both the datapath components' utilization, i.e, being active in every clock cycle, and the experienced data switching activity.

6.1.1 The Interplay of Contention and Data Switching Activity

The power consumption of every NoC component and, especially, the datapath components – considering both utilization and data switching activity – is directly related to the effect of contention and multiplexing. Whenever at least two flows compete for the same resource, e.g., a NoC link through a router's output port, they will possibly gain access to the shared resource in a time-multiplexed manner, depending on the employed arbitration policy. In this case, there is no way to predict the data switching profile seen at the output of the shared resource, since the output data stream is the result of multiplexing-in-time of two (or more) flows that are unrelated in terms of their data properties.

An example of the unpredictability of the output data stream is shown in Fig. 6.1, assuming two 4-bit-wide data flows. Multiplexing the two flows "corrupts" the cycle-by-cycle bit-level profile of the data traveling through the multiplexer. Although the two incoming streams exhibit considerable switching activity when viewed independently of

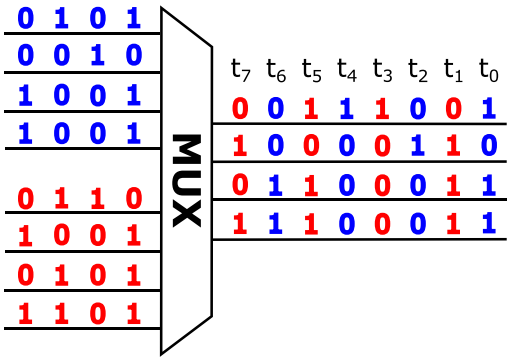


Figure 6.1: The process of multiplexing different data flows "corrupts" the data switching profile of each individual incoming data stream, and can possibly lead to very low dynamic power consumption.

each other, the arbitrated traffic that passes to the output of the multiplexer exhibits very few bits switching in every clock cycle.

Our goal is to control the data switching activity triggered inside the network through the data injected at the sources of the network, without any other intervention to the NoC's operation. To achieve this goal, we need to remove the unpredictability caused by intermediate contention/multiplexing points. This is enabled by injecting into the network totally *contention-free* traffic patterns. A traffic pattern is totally contention-free when the flows injected by the sources of the network never contend for the same resource in any part of the NoC. In this way, the data switching activity experienced by all the intermediate NoC components on the path from source to destination would exactly match the data switching activity of the injected traffic. This is facilitated by the fact that the unpredictability caused by multiplexing of unrelated data streams is completely avoided.

At first glance, it may seem contradictory that we employ contention-free traffic patterns, since contention-free flows are not often observed in real-world environments, and our goal from the outset was to generate a realistic stimulus. However, what we mean by a "realistic stimulus" is one that is *functionally feasible* (i.e., permitted to occur), not one that is necessarily frequently encountered. In fact, the goal of any power virus is to generate the *worst possible* load, not a typically encountered load.

It should be noted that, in the presence of traffic contention, the data switching activity may still be (possibly) predicted, and controlled di-

rectly from the sources of the NoC, but only if the details of the arbitration policy and the flow control rules (including any virtual channel allocation policy and the operational details of possibly pipelined routers) are considered. However, since we target a *high-level approach* that is *agnostic* to the NoC's micro-architectural details – in order to be easily applicable to any NoC configuration – we rely on conflict-free operation that allows us to fully control the data switching activity and, thus, maximize the observed power consumption.

6.1.2 Permutation Traffic and Network Utilization

To identify contention-free traffic patterns, we start by removing contention at the endpoints of the network, i.e., the traffic injected by each source is directed to a different destination. Therefore, out of all possible traffic patterns, we need to identify those permutation traffic patterns where (a) traffic is exchanged between unique source-destination pairs, (b) no flow contention is caused inside the network, and (c) network utilization is maximized.

An example of such a permutation traffic pattern is shown in Fig. 6.2(a) for a 3×3 2D homogeneous mesh NoC, where all links have the same width and operate under the same clock frequency, while the paths of the injected flows are determined by the XY routing algorithm. The terminal nodes are depicted as circles in the figure. The numbers beneath each terminal node indicate the source/destination pair of the flow originating from that terminal node, e.g., the "1 $\rightarrow$ 8" designation below node 1 indicates that the traffic generated at this node goes to node 8. Each source/destination pair is unique as needed by permutation traffic and contention is avoided completely thus allowing (a) for full NoC component utilization (injection and ejection throughput can be 100%), while, (b) at the same time, the sources can control, from outside the network, the data switching activity inside the NoC.

Not all permutation traffic patterns are appropriate for avoiding contention inside the network while still triggering the maximum network utilization. For example, the permutation traffic shown in Fig. 6.2(b) leaves 4 links idle (links 6 $\rightarrow$ 3, 3 $\rightarrow$ 6, 5 $\rightarrow$ 8, and 8 $\rightarrow$ 5), even if it completely avoids contention in the network and enables full injec-

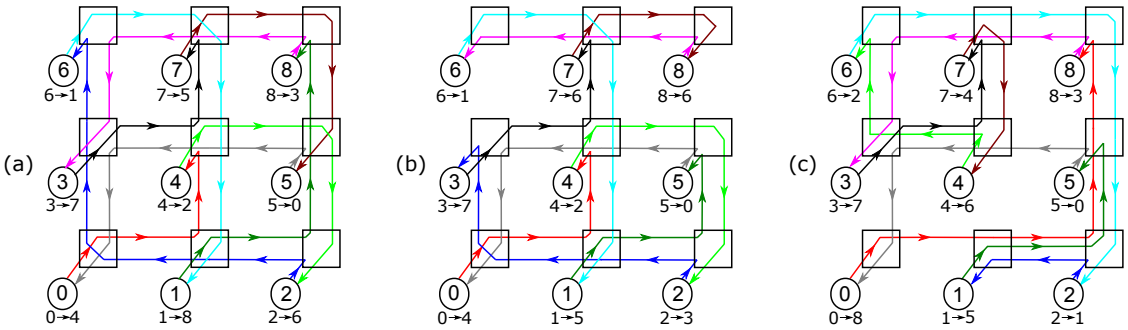


Figure 6.2: Three different permutation traffic patterns in a 3×3 2D mesh: (a) A pattern that causes maximum NoC component utilization, which yields peak power consumption (i.e., the desired pattern); (b) A pattern that leaves certain links idle, even though it achieves full injection/ejection throughput and avoids contention; (c) A pattern that merely congests certain network channels, while leaving parts of the NoC unutilized.

tion/ejection throughput transmissions.

Even worse, the permutation traffic shown in Fig. 6.2(c) not only leaves some links idle, e.g., $1 \rightarrow 0$ and $4 \rightarrow 5$, it also preserves in-network contention. Besides causing unpredictability in the data switching activity, in-network contention also limits the utilization of certain links, i.e., they do not receive a flit every cycle. For instance, the $0 \rightarrow 1$ link is only used with 50% throughput, because the red flow going from node 0 to node 8 encounters contention in the downstream node 1, and is forced to share the $1 \rightarrow 2$ link with the dark green flow going from node 1 to node 5. Since the red flow only uses 50% of the bandwidth of the $1 \rightarrow 2$ link, the throughput of the $0 \rightarrow 1$ link also falls to 50%, i.e., it remains idle for 50% of the time. Once a link is un(under)-utilized, the router components connected to the link's endpoints (multiplexers, pipeline registers, flow control logic, and input buffers) also remain idle, or under-utilized.

The permutation traffic shown in Fig. 6.2(c) is also a very good example of why peak-power traffic selection should not be confused with traffic patterns that cause network congestion and worst-case throughput. This permutation traffic is produced using the method in [137] with the goal being to maximize the load on certain channels, in order to stress the network and highlight its worst-case throughput. However, even though the NoC is congested, many portions of it remain under-/un-utilized, and exhibit lower power consumption.

In regular and homogeneous NoC topologies, peak power is triggered

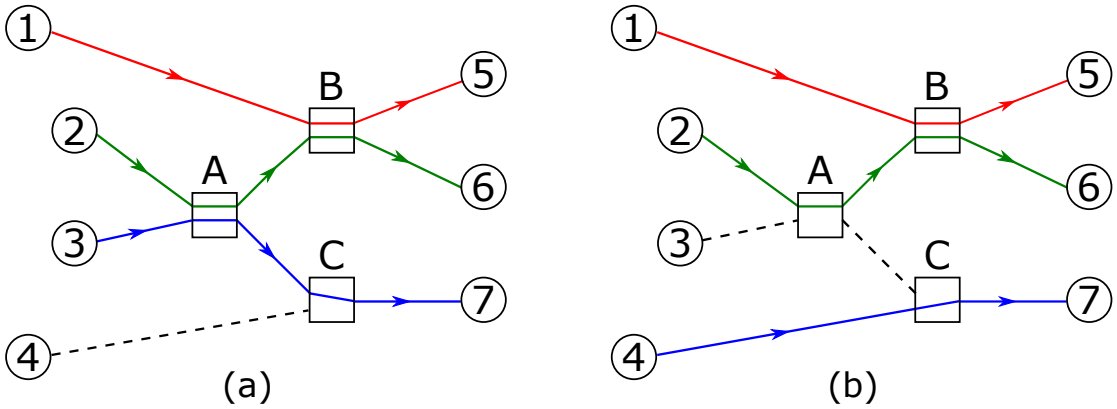


Figure 6.3: An example of a heterogeneous NoC with an irregular topology and one asymmetric router. Two traffic patterns are shown: (a) A contention-free traffic pattern that achieves the maximum possible utilization; (b) The traffic pattern that actually achieves peak power consumption, despite leaving two short links idle.

.

by appropriately-selected permutation traffic patterns, similar to the one shown in Fig. 6.2(a), which yield full utilization of all network links at the maximum throughput. Their contention-free operation allows the switching activity of each path to be controlled directly from the sources of the network in an end-to-end manner. Even if some NoC components are not identical – in terms of their micro-architecture – they are still fully utilized, irrespective of their differentiated design parameters. For example, all links receive full (100%) utilization, irrespective of their length. Both short and long wires are equally utilized; this attribute translates into longer wires consuming more power than shorter wires. However, identifying the appropriate permutation traffic by selecting randomly one from the set of all possible permutation traffic patters in not feasible. As shown in [124], only a few patterns (0.5% for a 3×3 mesh) fulfill the peak-power traffic requirements of allowing contention free operation and utilize all the links of the NoC. In larger networks the percentage of the favourable traffic patterns drops drastically, as well as the opportunity to find one randomly,e.g., on a 8×8 mesh $1.98 \cdot 10^{87}$ different permutation traffic patterns can occur.

6.1.3 The Case of Heterogeneous NoCs

In heterogeneous NoCs, many architectural and physical parameters can vary depending on the application scenario. For instance, the NoC may

consist of links with different bandwidths (i.e., the combined effect of link width and clock frequency), or parts of the NoC may belong to different voltage and frequency domains. In such cases, uniform full utilization of all NoC components may either be infeasible, or it may even lead to lower peak-power consumption. In such cases, leaving some links idle may increase the peak power consumption observed, by increasing the utilization of the parts of the NoC with the maximum contribution to the overall power consumption.

In the example shown in Fig. 6.3, the NoC has an irregular topology that employs uniform link widths, clock frequency, and voltage, but exhibits several other asymmetries in both the routers and the lengths of the links. The depicted contention-free traffic pattern in Fig. 6.3(a) achieves the maximum utilization and, inevitably, leaves one link idle (i.e., the one that connects source 4 with router *C*). Note that it is impossible to concurrently utilize all input links of router *C*, since the number of output ports (one) is smaller than the number of input ports (two), which means that the router's single output port cannot serve both of its inputs in the same cycle. However, in terms of peak power consumption, the most favorable traffic pattern for this topology is the one used in Fig. 6.3(b). In this case, the links are prioritized according to their power contribution; the ones with the larger power dissipation (i.e., the longer ones) are utilized, while other links with smaller power dissipation are left idle.

Leaving some links idle may also be required when trying to increase the power consumption of NoCs that are split across different voltage/frequency domains, as the one shown in Fig. 6.4[1]. In Fig. 6.4(a), the NoC is flooded by a contention-free permutation traffic pattern that utilizes all the NoC links. If the NoC operated under a single voltage/frequency domain, this traffic pattern would have been the most appropriate for triggering the peak power consumption. However, in this case, there are paths, such as $0 \rightarrow 5$, which involve links of different bandwidths (the links have the same bit-width, but they operate at different clock frequencies). This bandwidth asymmetry throttles the fast links (oper-

[1]In this example, we assume that voltage and clock-domain interfacing occur on the receiver side of each link. Thus, each link belongs to the voltage/frequency domain of its driver.

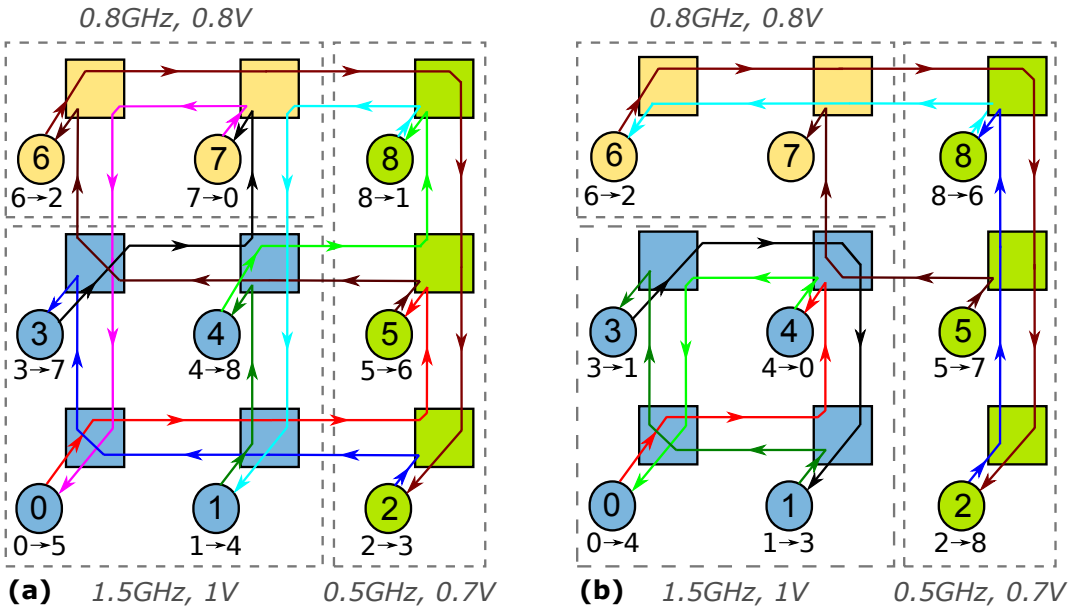


Figure 6.4: An example of a NoC that is split across different voltage/frequency domains. Two traffic patterns are shown: (a) A contention-free permutation traffic pattern that utilizes all the NoC links; (b) A traffic pattern that achieves higher peak power consumption by allowing the high-frequency links to operate at their maximum throughput (even if some other links remain idle).

ating at 1.5 GHz), and their effective throughput is determined by the slowest link of the path (i.e., the ones that belong to the 500 MHz domain). The injection throughput of source 0 that is clocked at 1.5GHz drops inevitably to 1/3 flits/cycle_at_1.5_GHz, due to the backpressure generated by the links that operate at 500 MHz. Therefore, the links of the path $0 \rightarrow 5$ that belong to the fast clock domain are under-utilized and experience only one third of their peak power.

On the contrary, if the same NoC is driven by the permutation traffic shown in Fig. 6.4(b), the peak power observed would be 1.3× higher than the peak power triggered by the pattern shown in Fig. 6.4(a). In this case, even if some links remain idle, the overall power consumption increases, since the most power-hungry links (the ones operating at 1.5 GHz) are allowed to operate at their maximum throughput of 1 flit/cycle_at_1.5_GHz, by avoiding mixing links with different bandwidths on the same path.

Overall, triggering the peak power consumption of a homogeneous, or a heterogeneous, NoC with a parameterized configuration *by only injecting traffic from the sources of the network* is a multi-parameter problem

with a huge design space.The approach presented in the next section can identify legal – with respect to the routing algorithm – and contention-free permutation traffic that tries to maximize – as much as possible – the power consumption of the NoC, by increasing the NoC components' utilization and their data switching activity in a controllable manner, after taking into account the NoC's structural and physical-implementation properties. By eliminating contention, the employed high-level approach allows us to control the utilization and the data switching activity experienced by the datapath components of the NoC. In this way, even though we cannot formally guarantee the generation of the maximum possible power consumption within the NoC at the gate level, we are able to significantly increase the peak power consumption of the NoC, as will be shown in the experimental results.

Other, non-permutation traffic patterns, whereby each source can send traffic to multiple destinations can be alternatively used for triggering high power consumption within the NoC. However, such patterns either lead to contention-free traffic that under-utilizes some links of the network (e.g., nearest-neighbor traffic in a 2D mesh, where each node sends all of its traffic to its immediate neighbors), or they achieve full utilization of all the NoC components while allowing contention during network traversal. As previously mentioned, contention makes the triggered data switching activity unpredictable.

6.2 Generation of Peak-Power Traffic

Deriving contention-free traffic patterns that maximize the NoC's power consumption is performed via a novel ILP-based formulation. The goal of the power maximization problem is to find a permutation traffic pattern – i.e., each source node sends all of its traffic to a unique destination different from the other sources – that does not cause any contention inside the network, and prioritizes the usage of the most power-consuming paths.

In its most generic form, the network connects N source nodes and M sink nodes. In reality, the number of source and sink nodes is equal, $N = M$, since each network terminal can both inject and receive data to/from the network.

The source and sink nodes that are allowed to communicate are declared via binary variables $c_{ij} \in \{0,1\}$. If $c_{ij} = 1$, then source i is eligible to send data to sink j.

The communication between a pair of source and sink nodes is performed using the links and the routers of the network. The path (i.e., the set of links and router input-output connections) that will be used for the pair's communication is solely determined by the routing algorithm. In this work, we target only deterministic routing algorithms, as employed in the majority of industrial NoC implementations [10, 110]. With deterministic routing, each source i can communicate with each sink node j via a single path, declared as P_{ij}. Therefore, for each one of the N sources, there are M candidate paths to be selected. For each path P_{ij}, we define a binary variable $x_{ij} \in \{0,1\}$ that declares if the path will be selected in the final permutation traffic pattern, or not. If $x_{ij} = 1$, then source i sends all of its traffic to sink j.

Permutation traffic imposes that each source i can send its traffic to at most one sink j, and each sink j can accept traffic from at most one source i. To satisfy these constraints, we need $\sum_i^N c_{ij}x_{ij} = 1$ and $\sum_j^M c_{ij}x_{ij} = 1$, where c_{ij} declares if endpoints i and j are allowed to communicate. As shown in Section 6.1, in order to maximize the power consumption of the NoC, it may be advantageous (in some cases) if the permutation traffic is not complete, i.e., some sources do not send any traffic, or some sinks do not receive any traffic. Therefore, to allow for this behavior, the constraints should be relaxed as follows: $\sum_i^N c_{ij}x_{ij} \leq 1$ and $\sum_j^M c_{ij}x_{ij} \leq 1$.

The selected permutation traffic should not cause any contention inside the network. This is guaranteed, if each link in the network is utilized for servicing the traffic of only one path P_{ij}. If more than one paths are serviced by the same link, it means that at least two sources send traffic through the same link, thus possibly causing contention and removing the required predictability of the data switching activity (see Section 6.1). If contention is avoided, then the switching activity of all the links and all the router input/output ports along a path P_{ij} can be directly controlled by the data injected from source i.

For each link k, we enumerate the set of paths that use the correspond-

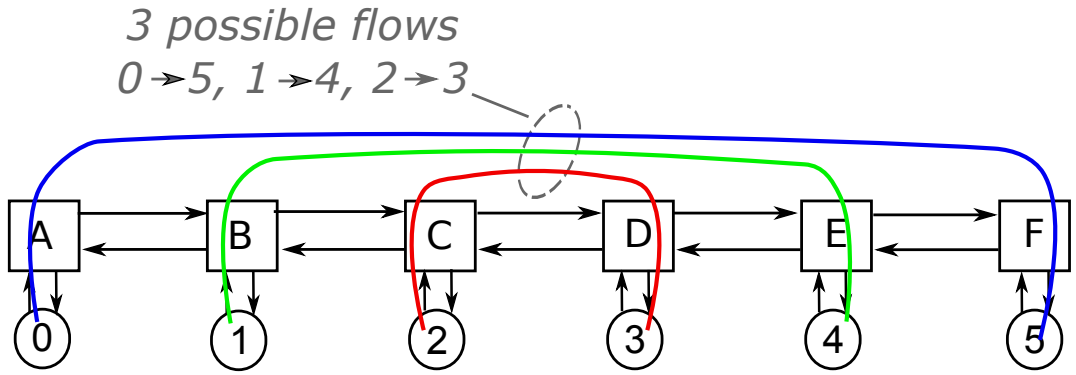


Figure 6.5: Choosing traffic-flow paths within the NoC. Link $C \rightarrow D$ may be used by 3 different flows, but, eventually, only one should be assigned to use it, or none of them (i.e., it may remain idle).

ing link, following the rules of the selected routing algorithm. In the example shown in Fig. 6.5, the link that connects routers C and D is used in the set of paths $\{P_{05}, P_{14}, P_{23}\}$. This link should be assigned to only one of the three paths that can possibly use it, or to none of them. This constraint is satisfied when $x_{05} + x_{14} + x_{23} \leq 1$. A link is allowed to stay idle, if this gives more freedom to the overall power maximization problem. Transforming this inequality to an equality would constrain the ILP to use every link for serving some communication path, which is only desirable in the case of *fully homogeneous* NoC designs. The relationship between the links of the network and the paths that can use them is summarized in a 3D binary matrix with elements $t_{ijk} \in \{0,1\}$. Element $t_{ijk} = 1$ when link k is used by path P_{ij}. This assignment holds for both the in-network links and the incoming/outgoing links that connect the source and sink terminals to the routers of the network.

The ILP used for identifying the appropriate traffic pattern that maximizes the power consumption of the NoC is formulated as follows:

$$\textbf{maximize} \quad \sum_{i=1}^{N}\sum_{j=1}^{M} w_{ij} c_{ij} x_{ij}$$

$$\textbf{subject to} \quad \forall \text{ link } k: \quad \sum_{i=1}^{N}\sum_{j=1}^{M} t_{ijk} c_{ij} x_{ij} \leq 1$$

$$x_{ij} \in \{0,1\}, \quad i = 1,\ldots,N \quad j = 1,\ldots,M$$

For the solution of the ILP, i.e., the identification of the optimal values

of x_{ij}, the binary variables c_{ij}, t_{ijk} are constants declaring whether source i can send traffic to sink j, and whether link k is used by path P_{ij}, respectively. The power cost of selecting the path P_{ij} is equal to w_{ij}.

For fully *homogeneous* NoCs, as the ones shown in Fig. 6.2, which merely need a permutation traffic pattern that is conflict-free and utilizes all the links of the NoC, we just need to set $w_{ij} = c_{ij} = 1$. The variables t_{ijk} are not simplified, and are set according to the properties of the routing algorithm.

In *heterogeneous* networks, the power cost w_{ij} of each path P_{ij} is affected by several parameters, since the routers and the links along the path between source i and sink j may belong to different voltage domains, may operate at different clock frequencies, and may have different bit-widths.

The approach in [124] identified the traffic pattern that utilized all the links of the NoC by finding a Hamiltonian path on the enhanced channel dependency graph. Although this approach is effective in homogeneous NoCs, it cannot be applied to heterogeneous topologies, since full link utilization may either be infeasible, or it may even lead to lower peak-power consumption. On the contrary, the ILP-based optimization covers both cases effectively.

6.2.1 The Power Cost of Each Path

The dynamic power consumption of a path P_{ij} is the sum of the dynamic power experienced in each segment of the path, which includes moving flits from the input buffer of one router to the input buffer of the next router. This data movement across each path segment consumes power inside the router, on the link that connects the two neighboring routers, and within the input buffer of the next (downstream) router.

Router traversal accounts for the power consumed for a buffer read and for traversing the crossbar, while also including the power expended in the control logic (input-request generation logic, arbitration, and flow control logic). The actual power cost can vary significantly based on the router's configuration, i.e., the number of input/output ports, the flit width, the number of Virtual Channels (VC), the buffer depth of each VC, and the number of pipeline stages.

In every configuration, the power spent inside the router is heavily data-dependent. Even if a new flit is sent in every clock cycle, if these flits happen to have almost the same bit-level profile, the actual power consumption remains very low, due to minimal switching activity. Similarly, the dynamic power of the link depends on the data switching activity of the transferred bits.

In either case, we cannot have a clear estimate of the power cost of each path segment, unless we have determined the data switching activity seen by the corresponding NoC components. The data switching activity is determined by two factors. The first one is the throughput of data transfers λ_{ij}, i.e., how often a new data word (flit) enters and leaves path P_{ij} of the NoC, and the second one is the bit-level profile of each transmitted word. Once both factors are known, the assumed power cost would be accurate enough, since it would reflect both the power of the exact NoC configuration (link widths, number of VCs, etc.), and the power consumed due to the selected data profile.

In general, the power cost w_{ij} of each path P_{ij} is equal to

$$w_{ij} = \lambda_{ij} \times \sum_{s}^{\#\text{segments}} \text{Power\_of\_segment}(s).$$

The calculation of the effective injection throughput for each path λ_{ij} is described in Section 6.2.2. The methodology of generating the bit-level data patterns that guarantee high switching activity (and are used for the determination of the Power_of_segment) is presented in Section 6.2.3.

6.2.2 Effective Throughput of Each Path

Even if the paths derived from the ILP will be conflict-free, we may not be able to inject new flits at the maximum throughput of 1 flit/cycle in certain paths. This phenomenon appears only in *heterogeneous* NoCs that consist of paths with links of different bandwidth. In homogeneous NoCs, full-throughput data transfers on all links is *always* achieved.

In the example shown in Fig. 6.6, all the links operate at the same clock frequency (1 GHz), but one link is narrower. Thus, all links can transfer 64-bit flits, except one link that can transfer 32-bit flits. In this case, the narrow link will throttle the flow of data due to (de)serialization under

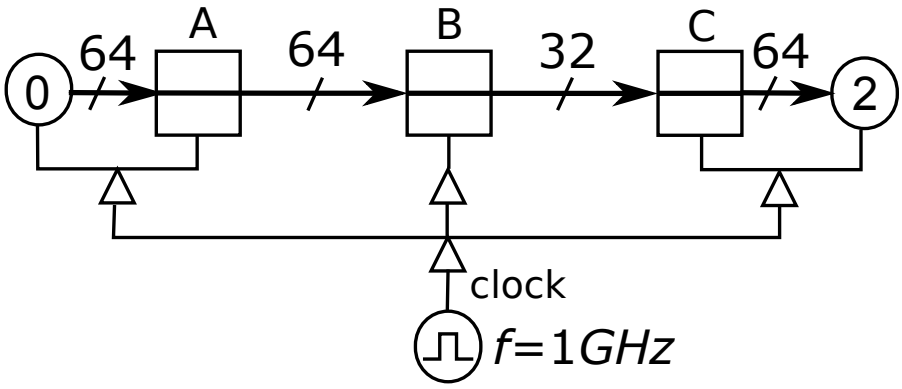


Figure 6.6: In this example, all links operate at the same clock frequency (1 GHz), but one link is half as wide as the other links (32 vs. 64 bits). Consequently, the narrow link will throttle the flow of data due to serialization, thereby limiting the maximum utilization and switching activity observed on the wide links.

the same clock frequency. The effective throughput seen at the injection source, which consists of wide links, would be half of the maximum possible. Hence, the effective switching activity experienced on the wide links would be half of the maximum that could be experienced if all the links of the $0 \rightarrow 2$ path had equal bandwidth.

If we denote the bandwidth of each link k as $BW_k = f_k \times W_k$, where f_k is the clock frequency of the driver of the link and W_k is the link's bit-width, the effective injection throughput of a path P_{ij} that consists of multiple links in series (separated by routers) is equal to

$$\lambda_{ij} = \frac{\min BW_k}{BW_i}, \quad \forall \, \text{link} \, k \in P_{ij}.$$

6.2.3 Maximizing The Data Switching Activity

The final step for the determination of the power cost w_{ij} of each path P_{ij} is to estimate the power that is dissipated in each router and link of the NoC (i.e., Power_of_segment). To do so, we rely on real NoC implementations using state-of-the-art EDA tools driven by data patterns that maximize switching activity.

Identifying accurately the appropriate (worst-case) data patterns can be done only using specific gate-level techniques [146, 99]. However, such techniques can be applied only in certain sub-modules of the NoC and cannot be extended to the network-path level. Therefore, to tackle the

problem of identifying the worst-case data patterns, we need to consider only the microarchitecture-level aspects of the NoC and focus on the microarchitectural features that are found in the majority of NoC designs.

For the links, a repetitive data pattern that switches between $0101\ldots01 \rightarrow 1010\ldots10$ is enough to trigger worst-case power consumption. Each bit experiences a change in every cycle, either $0 \rightarrow 1$ or $1 \rightarrow 0$, which switches the corresponding capacitance of the wire to ground. Further, this data pattern ensures that neighboring wires always switch in the opposite direction, thereby causing the worst-case power consumption, due to the link's coupling capacitance [143].

However, for the VC buffers and the internal logic of the router, we cannot be sure of the exact switching activity caused by this 2-data vector pattern. Assume, for example, the case of VC buffers that are built using register-based (i.e., flip-flop-based) FIFO queues, or using SRAM blocks. In either case, power is consumed every time a new flit is written to, or read from, the VC buffers. On each write, a new flit is written to only one VC. Inside the queue of each VC, the flit is written into the register that corresponds to the address pointed to by the tail pointer of the enabled VC queue. Therefore, on each write, only the bits of one register can change value. The rest are not enabled, or remain clock-gated, as normally done in industrial NoC implementations. Therefore, to maximize power, we need to guarantee that (a) the new value written to the register is different from the one already stored, and (b) the two values (the old and the new one) differ by as many bits as possible. This can only occur if we know beforehand the specific slot of the VC queue into which the incoming flit will be stored.

When a repetitive data pattern of D words (flits) is placed – one word after the other – in a buffer with B slots, then we can guarantee that any incoming word will be written (stored) into a register that already stores a different value, as long as the greatest common divisor of D and B is equal to one. When B is odd, the 2-vector data pattern that also maximizes the power on the links is the proper choice. When B is even, we can select a repetitive data pattern of $B+1$ words. The $B+1$ words can safely include $B/2$ repetitions of the 2-vector data pattern $0101\ldots01 \rightarrow 1010\ldots10 \rightarrow 0101\ldots01 \rightarrow 1010\ldots10$, plus an all-zero

vector. Depending on the NoC configuration, the repetitive set of data patterns can also extend across different packets, as long as the flits of the packets flow consecutively in the NoC.

In terms of power, the traffic injected can stay within the same VC from source to destination, as long as one flit is written and read per cycle, and the data values written and read have the maximum bit-wise difference. Distributing traffic across VCs for each non-conflicting flow produced by the proposed method is possible, but it needlessly complicates the derivation of the appropriate data switching patterns that cause the maximum switching activity, without any true impact on the triggered power consumption.

Even though the non-conflicting nature of the traffic patterns can maximize the switching activity in the datapath of the NoC (links, buffers, crossbar), the arbitration part is kept operating on the same requests and grants in each cycle. This causes minimum switching activity in this portion of the NoC. However, this is not a problem in modern NoCs with wide datapaths of 64 bits or more, since the power of the arbitration logic is low relative to the datapath portion [63]. This argument is also verified by the experimental results in Section 6.3, when using random traffic. The latter maximizes the switching activity in the arbitration modules, due to the random nature of the input requests. Even when compared with this traffic scenario, the proposed technique achieves significantly higher power consumption by only appropriately targeting the switching activity in the part of the NoC that carries data.

6.2.4 Overall Flow and Examples

The steps involved in the application of the proposed methodology is summarized in the flow depicted in Fig. 6.7. Driven by (a) the NoC topology that describes the NoC's structure, the voltage/frequency domains, and the link widths, (b) the routing algorithm, and (c) the set of nodes that are allowed to communicate, we form the network paths across all pairs of source and sink nodes, and then calculate the power cost of each path. For the calculation of the power cost of each path, we use the power pre-computed for each NoC component. Once the weight for every pair of communicating nodes is known, the ILP is formulated

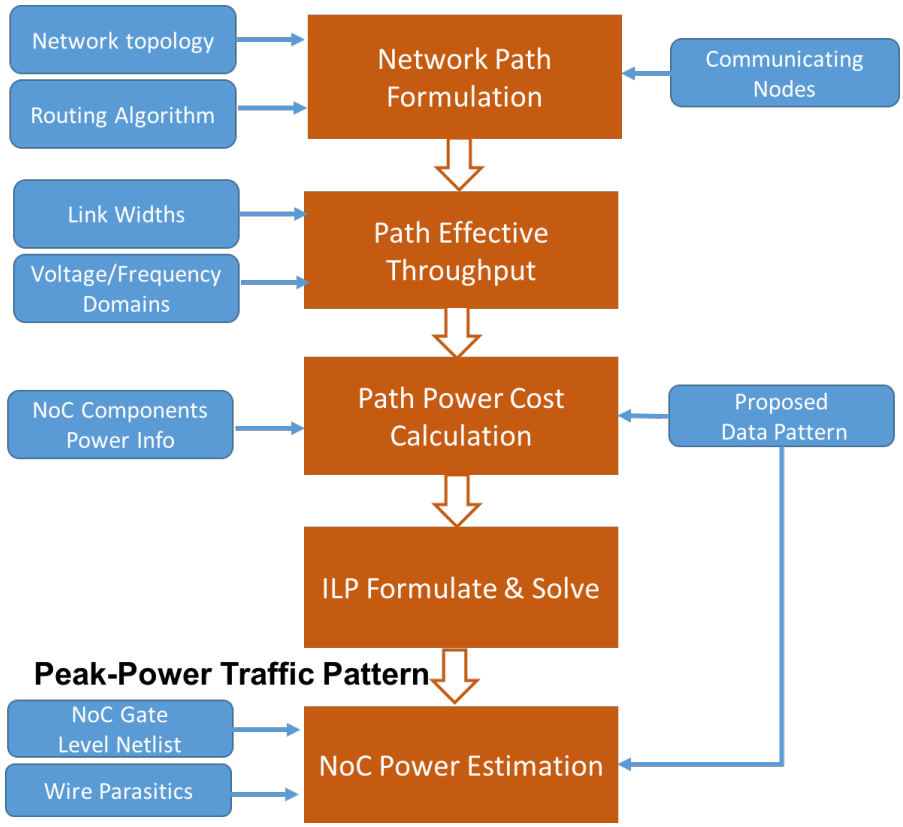


Figure 6.7: The overall flow of the proposed methodology, which can be used to derive peak-power traffic patterns.

and solved using the Gurobi solver [56]. The derived traffic patterns and the proposed data patterns then drive timing-accurate gate-level logic simulations to provide the actual switching activity that is subsequently used to calculate the power consumption of the NoC.

Fig. 6.8(a) depicts the ILP-derived non-conflicting permutation traffic that causes 100% link utilization in an asymmetric 2D mesh network (the asymmetry is the result of a faulty router which is decommissioned and not shown in the figure). In this case, the turning restrictions of the routing algorithm [114] (depicted as small arrows at certain turn-points within the network) guarantee connectivity and deadlock freedom. Equivalently, the peak power traffic for a tree that applies the up/down routing algorithm is highlighted in Fig. 6.8(b).

Fig. 6.9 illustrates a peak-power traffic scenario for a hierarchical ring. Ring and tori topologies employ VCs to ensure freedom from possible routing deadlocks. To model the specific use of VCs for deadlock-free

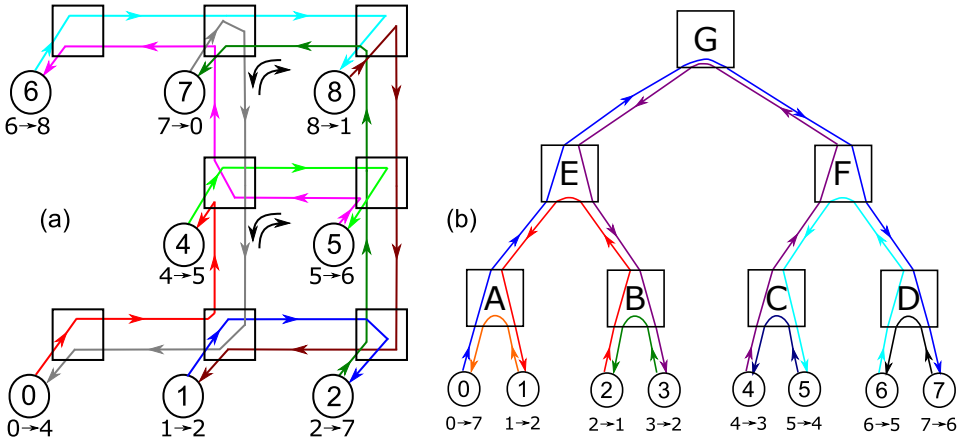


Figure 6.8: The peak-power traffic patterns generated by the proposed methodology for (a) an irregular network, such as an asymmetric 2D mesh, and for (b) a tree topology.

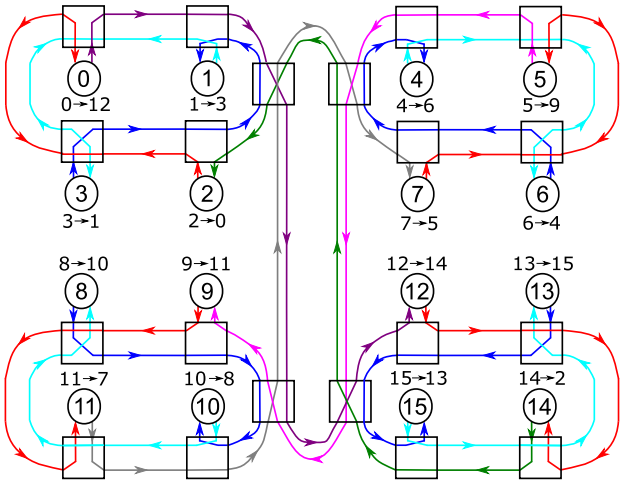


Figure 6.9: Peak-power traffic patterns derived by the proposed methodology for a hierarchical ring, which employs virtual channels for deadlock freedom.

routing, we include each link multiple times in the ILP; as many times as the number of supported VCs per network link. Eventually, after the solution of the ILP, only one VC will be used per physical channel of the network to carry the generated peak-power traffic. The derived traffic flows are allowed to change VC in-flight, as long as this is dictated by the routing algorithm; in any other case, each traffic flow remains within the same VC. In this way, the ILP exercises all possible turns at the VC level, but, ultimately, it selects only one VC-to-VC connection. The proposed optimization does not impose any specific rule for acquiring a VC, other than the rules imposed by the routing algorithm.

The proposed methodology can also be applied – without any changes –

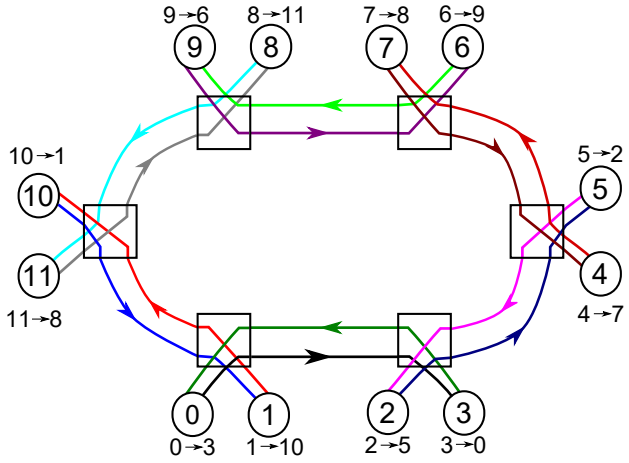


Figure 6.10: Peak-power traffic patterns derived by the proposed methodology for a ring that employs concentration, whereby two terminal nodes are connected per router.

to the case of NoC topologies that employ *concentration*, i.e., where multiple terminal nodes are connected per router. An example peak-power traffic scenario derived by the proposed approach on a concentrated ring is shown in Fig. 6.10. Depending on the NoC configuration, the derived peak-power traffic patterns may involve both local and global traffic. Local traffic involves the exchange of traffic across terminal nodes connected to the same router, while global traffic involves traffic exchanged across terminal nodes that belong to different routers. Both traffic cases enable the maximization of the power consumption across the NoC's datapath, since they utilize – as much as possible – both the routers' internal datapath and the NoC links.

6.2.5 The Complexity of The ILP

The complexity of the ILP is determined by the number of variables x_{ij}, i.e., $N \times M$, and the number of constraints, which is equal to the number of links in the NoC. For practical NoC cases of up to 1024 nodes, the ILP can easily be solved within a reasonable execution time. The peak-power traffic identification problem, including both path enumeration and weight calculation, as well as the solution of the ILP (assuming that the power of routers and links is computed beforehand), can complete its execution in just a *few minutes* for well-known NoC topologies of *hundreds of nodes*. The run-times required to derive peak-power permutation traffic patterns for various topologies and various network sizes

are shown in Table 6.1. The run-times correspond to an implementation that runs on a Linux computer with a 2.3 GHz Intel Core i7-4712HQ Processor and 16 GB of RAM.

Table 6.1: Run-times of the proposed technique: Time needed to derive peak-power permutation traffic patterns.

#Nodes	Ring	Hierarchical Ring	2D Mesh	3D mesh	2D Torus	Hetero 2D Mesh
16	0m 1s	0m 9s	0m 2s	0m 1s	0m 1s	0m 4s
64	0m 5s	0m 6s	0m 4s	0m 3s	0m 2s	0m 8s
256	0m 17s	0m 12s	0m 11s	0m 10s	0m 8s	0m 21s
1024	10m 32s	8m 27s	21m 06s	21m 31s	21m 47s	24m 43s

The investigated topologies correspond to well-known homogeneous networks and a heterogeneous 2D mesh that is split into 3 voltage/frequency domains, similar to Fig. 6.4(a). The size of each domain grows proportionally to the overall network size. From the runtimes reported in Table 6.1, deriving the peak-power traffic patterns for the heterogeneous 2D mesh (right-most column) requires more time than the homogeneous cases. The extra time is spent for weight calculations and for solving the ILP. Nevertheless, the reported runtimes are manageable even for very large NoCs.

6.3 Experimental Evaluation

The goal of the proposed method is to trigger the peak-power consumption of a NoC by injecting appropriately selected traffic patterns that maximize network component utilization and data switching activity. The proposed traffic patterns do not formally guarantee the maximization of the power consumption of the NoC at the *gate level*. However, they cause significantly higher peak power than random traffic, and, since they do not rely on the gate-level details of the NoC components, they can be successfully applied to multiple NoC configurations. In any case, it should be stressed that the proposed traffic patterns aim at the on-purpose maximization of the power consumption. Therefore, they represent – by construction – a *corner case* of the NoC's operation, which is expected to occur rarely under normal system operation.

The experiments evaluate 64-node NoCs following 2D mesh and hier-

archical ring topologies. Other tested topologies show similar trends. Both homogeneous and heterogeneous variants of the aforementioned topologies are evaluated. In order to contain the number of possible configurations, we assume a tile-based chip floor-plan similar to the Scorpio chip [29]. Scorpio was built at 45 nm technology (which matches the technology library we used in our implementations), using a tile size of approximately 2×2 mm. Based on the chosen NoC topology, the NoC routers can have a variable number of input and output ports. For every configuration, we assume that the NoC supports 4 VCs per input port, with 5 buffer-slots/VC, and the NoC routers employ the 3-stage pipelined organization of Scorpio routers [29].

All NoC components used in the evaluation were implemented in SystemVerilog, mapped to a commercial low-power 45 nm 0.8 V standard-cell library, and placed-and-routed using the Cadence digital implementation flow. Depending on the NoC topology, a different placement-and-routing round was conducted.

Power was measured after performing timing-accurate simulations, using the proposed data patterns and including all back-annotated layout parasitics. Power measurements were performed twice: once for characterizing the power cost of each NoC component, as needed for the computation of the weights of the ILP, and, secondly, for deriving the final power of the NoC when it operates on the selected traffic pattern produced by the ILP.

6.3.1 Homogeneous NoCs

In the first set of experiments we evaluated the proposed methodology on homogeneous NoCs operating at 1GHz. In this case, the inter-router NoC links carry 64 bits of data, plus some extra flow control information. The header flit, which also includes network-addressing information, carries fewer actual data bits.

In the first set of experiments, we compare the proposed method against random synthetic traffic patterns, under various data switching and network-injection scenarios. The instantaneous power consumed by a NoC when the incoming traffic causes contention across flows with unrelated data (this occurs in almost all cases under normal operation) can

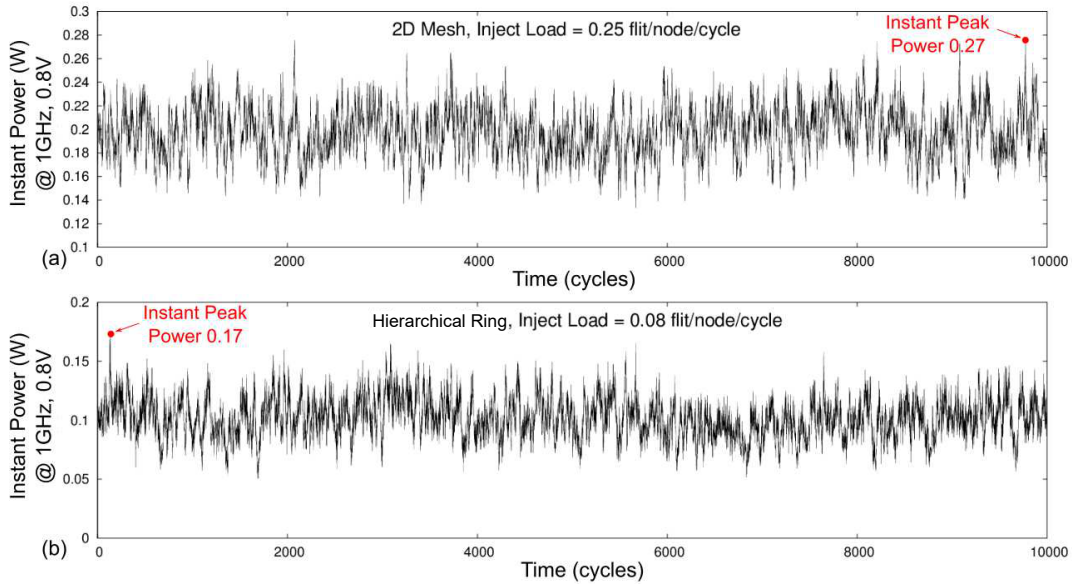


Figure 6.11: A 10K-cycle snapshot of the instantaneous power consumption of a homogeneous 64-node 2D mesh (top) and a hierarchical ring (bottom), after the network reaches steady-state operation, using uniform-random traffic and data.

vary significantly over time, depending on the switching activity in various parts of the network in each cycle. This behaviour is highlighted in Fig. 6.11, for an 8×8 2D mesh and a 64-node two-level hierarchical ring that consists of 8-node local rings connected via an 8-node global ring.

Both networks receive uniform-random traffic at a different rate (close to their saturation throughput), as reported in Fig. 6.11. The two NoCs have equal link width, i.e., 64 bits plus flow-control bits, and both operate at 1 GHz. Therefore, the bisection bandwidth of the 2D mesh is larger than the bisection bandwidth of the hierarchical ring. The injected packets are 5-flit long and carry random data in their payload portion. In this experiment, the bit of each flit when entering the network has equal probability of being 0 or 1, independent of the rest of the bits of the same flit, or the previous flits.

The peak power consumption achieved by random traffic is merely the peak instantaneous power observed during the simulation's time frame. There is no guarantee that a large power value can be triggered during simulation, due to the unpredictability in switching activity, and the lower NoC utilization caused by contention among different flows. Additionally, the observed peak power consumption simply represents an *instantaneous* peak. This cannot be sustained over a longer period

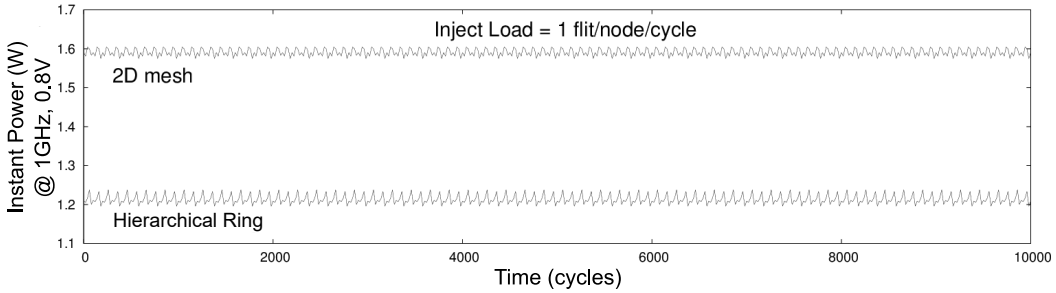


Figure 6.12: A 10K-cycle snapshot of the instantaneous power consumption of a 64-node 2D mesh and a hierarchical ring (under steady-state network operation), using traffic/data derived by the proposed methodology, with full injection throughput.

of time, which would be required to observe possible temperature increases and identify thermal hot-spots in the system.

On the contrary, the proposed method does not have such limitations. In Fig. 6.12, we report the instantaneous power consumed by the proposed approach under 100% injection load for each case (2D mesh and hierarchical ring). The results are measured by injecting 5-flit packets in the NoC, following the ILP-derived permutation traffic pattern, and carrying data payloads with the 2-vector data patterns described in Section 6.2.3. Evidently, the proposed methodology keeps power consumption constantly and consistently very high. The minimal variance in the power consumption is due to the switching profile of the header and flow-control bits, which are not controlled by our ILP-based approach.

Next, we compare the peak power consumption of the proposed method and random traffic scenarios (*uniform-random* and *bit-complement* traffic), under the same injection load. For each injection load, the maximum instantaneous power consumption value observed (over 500,000 cycles of simulation) was recorded. The results are depicted in Fig. 6.13, for the same 2D mesh and hierarchical ring topologies. The peak power consumption of random traffic (blue curves) follows the throughput behavior of the network itself, and, after saturation (when the utilization of NoC components reaches its limit), the peak power consumption observed is rather constant. On the contrary, the proposed approach can increase the power consumption to its true maximum value, due to its non-conflicting traffic. The data switching activity is directly controllable by the input sources, and it covers all the intermediate router ports

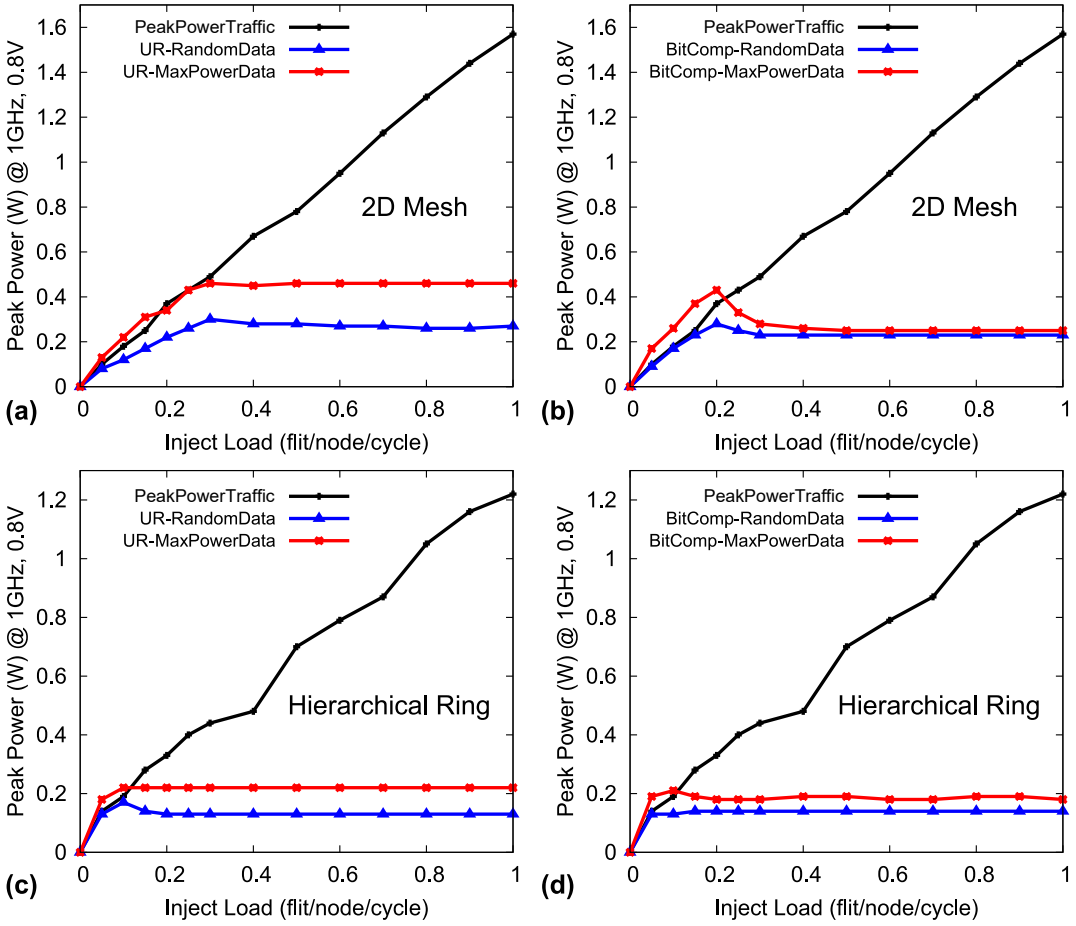


Figure 6.13: Peak power consumption vs. injection load in *homogeneous* NoC topologies. The power consumption triggered by the proposed methodology, as compared to the power consumed when using uniform-random and bit-complement traffic patterns.

and network links that are utilized by the injected flow. When compared against uniform-random traffic (Figs. 6.13(a) and (c)), the proposed technique triggers maximum power consumption, which can be more than 6× higher than the one achieved under uniform-random traffic with random data (i.e., blue curves).

This is also true when the NoC is driven by uniform-random traffic that allows contention in the network, but the injected *data patterns* are the same as the ones used in the proposed case (by following the guidelines described in Section 6.2.3). This scenario is also depicted in Figs. 6.13(a) and (c) with the red curves. The ILP-driven approach still consumes significantly more power (4× higher), since it simultaneously takes into account both the network utilization and the data switching activity.

Similar conclusions are derived when the power consumption of the NoC is triggered using other permutation traffic patterns, such as bit-complement traffic. In this case (Figs. 6.13(b) and (d)), the peak power consumption of the proposed method is 4× larger than the *largest* power observed under the bit-complement traffic patterns (red curves).

It should be noted that, at low injection rates, the red curves in Fig. 6.13 (i.e., random traffic scenarios using the data patterns proposed in this work) are – in some cases – slightly higher than the black curves. However, this is an artifact of the unpredictability in the data switching activity caused by traffic contention. This unpredictability causes large variance in the recorded peak power consumption, which makes the maximum value reported in Fig. 6.13 very hard to repeat in a systematic manner.

Table 6.2: Peak power of the proposed method vs. the power of a fake scenario, which assumes that every circuit node switches in every cycle.

@1 GHz, 0.8 V	Fake (pessimistic)	Proposed
2D Mesh	4.87 W	1.6 W
Hier. Ring	3.76 W	1.23 W

Finally, we compare the power triggered by the proposed peak-power traffic, versus the peak-power consumption that corresponds to the fake scenario of every circuit node switching in every cycle. In both cases depicted in Table 6.2, the peak power triggered by the proposed approach (measured at 100% injection rate) is lower than the one derived using the fake (un-realistic) approach. The difference between these two maximum power values depends on topology characteristics, and the power expended on the links vs. the power expended within the routers. In any case, the significant conclusion out of this comparison is that fake peak-power scenarios overestimate the true maximum power profile of the NoC and unnecessarily increase the overall system power budget. With the proposed optimization, worst-case power analysis is brought closer to what is attainable by a corner-case but realistic traffic pattern.

6.3.2 Heterogeneous Topologies

In addition to homogeneous topologies, the proposed methodology can also handle *heterogeneous* topologies. To test the effectiveness of the ILP-driven peak-power generation methodology when dealing with heterogeneous NoCs, we experimented with the two heterogeneous topologies shown in Fig. 6.14. The first one involves three voltage/frequency domains assuming homogeneous links of 64 bits, while the second one assumes the same voltage and clock frequency of 1 GHz@0.8 V throughout the NoC, but includes two different link widths, 64 and 128 bits, following a topology similar to the one presented in [93].

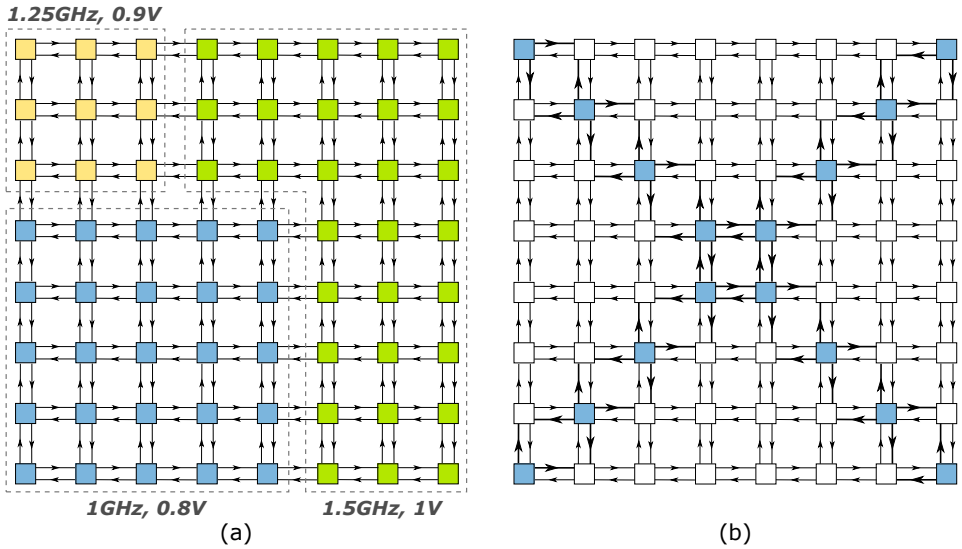


Figure 6.14: Two *heterogeneous* NoC topologies are evaluated: (a) one that includes multiple voltage/frequency domains, and (b) one that employs heterogeneous link widths and routers.

Similar to the homogeneous case, we initially compare the peak-power consumption of the proposed method and a uniform-random traffic scenario, under the same injection load. The results are depicted in Fig. 6.15. For each injection load, the maximum instantaneous power consumption value observed (over 500,000 cycles of simulation) was recorded. Note that the injection load reported in the x-axis of the diagrams of Fig. 6.15 refers to the injection load of the sources with the lower bandwidth. In Fig. 6.15(a), the injection load refers to the sources with the lower clock frequency, while, in Fig. 6.15(b), it refers to the sources with the narrower link width.

The ILP-based optimization guarantees the generation of non-conflicting permutation traffic and increases – at the same time – the effective injection bandwidth between each source and destination pair. This enables the application of the worst-case traffic pattern, driven by appropriately selected data, at the maximum possible rate. This property increases the maximum power observed, as compared to the power observed when driving the NoC with a uniform-random traffic pattern. Even if the random traffic uses the proposed worst-case data patterns, it still produces 2× lower power consumption than the proposed traffic. The conflicting nature of uniform-random traffic inevitably saturates the NoC, thus limiting the maximum utilization and, effectively, the power consumption that can be observed within the NoC.

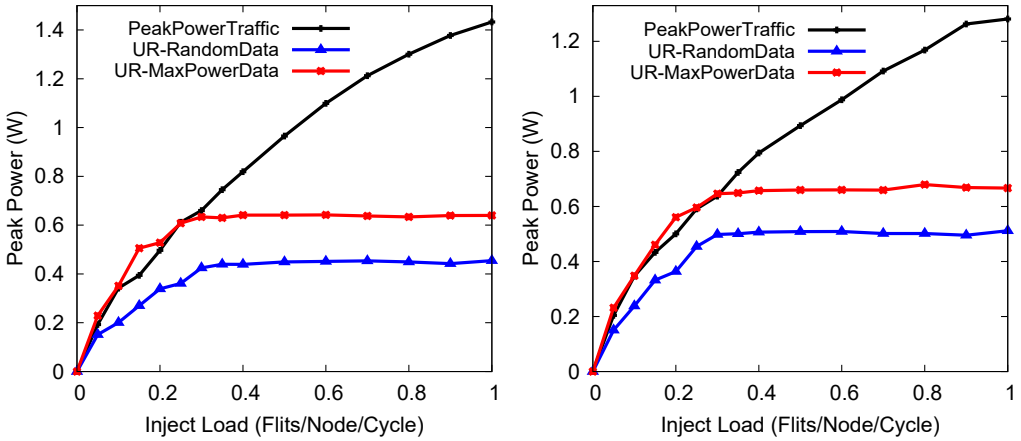


Figure 6.15: Peak power consumption vs. injection load in *heterogeneous* NoC topologies. The power consumption triggered by the proposed method, as compared to the power consumed when using uniform-random traffic patterns in the two examined heterogeneous topologies.

In the last set of experiments, we further highlight the need for non-conflicting traffic that would guarantee the control of the data switching activity in all NoC components, while still offering maximum NoC component utilization. We relax some of the constraints of the ILP to produce one additional traffic pattern that fully utilizes all NoC components, but it *allows contention during packet network traversal*. In this scenario, each NoC source is allowed to send traffic to multiple destinations, and each destination can receive traffic from multiple sources. The ILP selects the appropriate percentage of traffic injected for each

source-destination pair that guarantees maximum link utilization, but without guaranteeing contention-free network traversal. The derived traffic resembles the traffic that is produced by a maximum-flow-like algorithm [21], with the restriction that the traffic that floods the network links should use paths that are allowed by the routing algorithm.

Traffic is injected at the maximum possible rate for 500,000 cycles, using the the data patterns proposed in Section 6.2.3. The four highest recorded power values when using these traffic patterns – that allow for network *contention* – are included in the diagrams shown in Fig. 6.16, next to the peak-power consumption derived by the proposed method, assuming full injection throughput. The largest power measurements recorded for the generated traffic were always lower than the power triggered by the proposed methodology: 32% lower in the case of NoC topologies with multiple voltage domains (Fig. 6.16(a)), and 36% lower in NoCs with heterogeneous link widths (Fig. 6.16(b)). This result is a direct consequence of the contention that appears in the NoC, which, inevitably, (a) leaves some links unutilized for some cycles during the NoC's operation, and (b) destroys any predictability in the data switching activity.

The appropriate permutation traffic patterns that yield extremely high link utilization constitute an extremely small subset of the entire set of possible permutation traffic patterns. Therefore, randomly deriving an effective permutation traffic pattern, without relying on the proposed ILP formulation, should not be considered a viable/safe option.

To test this argument, we randomly generated 100,000 permutation traffic patterns (self-traffic was not allowed), and, for each one, we measured the peak-power consumption after injecting traffic at the maximum possible rate for 500,000 cycles. The four highest peak-power measurements recorded were also included in the diagrams shown in Fig. 6.16. The highest power measurements we got for the randomly generated traffic were always significantly lower than the power triggered by the proposed method.

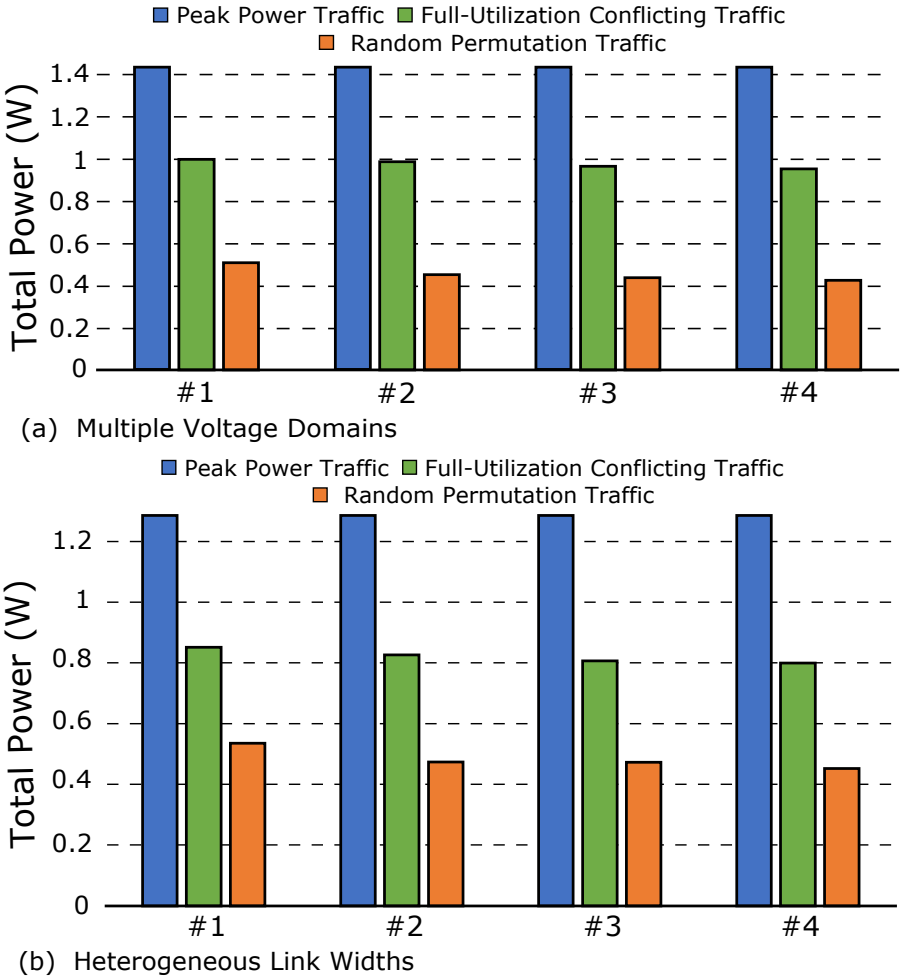


Figure 6.16: The peak-power consumption observed using (1) the proposed methodology, (2) the four best (in terms of peak-power consumption) traffic patterns that allow for *contention* within the NoC, and (3) the four best (in terms of peak-power consumption) permutation traffic patterns among 100K randomly generated patterns.

6.4 Conclusions

As chips become increasingly more dense and complex, power consumption becomes a primary design constraint. It is imperative for designers to realistically estimate a design's peak power consumption, which directly impacts other salient system attributes, such as performance, implementation costs, battery life, and reliability. This work introduces a fully-automated high-level methodology to generate appropriate traffic and data patterns that cause peak power consumption within the NoC. The peak power consumption triggered by the proposed method is, on average, 4× higher (up to 8× higher) than what is

observed after simulating random traffic and data patterns.

The introduced ILP-based optimization enables the power maximization of the majority of NoC components, irrespective of their differentiated design parameters. Heterogeneous and homogeneous NoCs are handled in a unified manner, allowing for the generation of appropriate traffic patterns – even for large topologies – within reasonable execution time.

Chapter 7

Conclusions

7.1 Summary

As the size of CMPs and SoCs continues to grow, and with the increasing use of mobile devices, finding an efficient solution to address the on-chip communication constraints becomes a major challenge. NoCs have been established as the dominant communication medium, due to their modular approach, their physical scalability, and ease of integration. NoCs must provide high communication throughput and low-latency transfers, as well as QoS guarantees. These features must be provided in efficient hardware implementations that satisfy tight timing, area, and power constraints. This thesis tries to address these challenges for power-efficient NoC design and implementations.

The first presented solution optimizes the buffer architecture of the NoC router. The proposed architecture, called ElastiStore, can efficiently merge elastic operation and buffering with VC flow control. It is a novel lightweight EB architecture that minimizes buffering requirements without sacrificing performance. ElastiStore can be adopted at the inputs and the outputs of NoC routers, covering the requirements of round-trip time arising from the pipelined organization of the router micro-architecture. ElastiStore-based routers enable the design of low-cost routers with significant area and power savings without sacrificing any network performance, as verified using simulations with both synthetic traffic and real application workloads.

With the use of the new buffer architecture, the second solution presented in this thesis employs a novel distributed VC-based router architecture. The proposed ElastiNoC design allows for modular pipelined organizations that increase the clock frequency with the utilization of an efficient buffering strategy. Additionally, it offers maximum freedom in terms of physical placement, by allowing the NoC components to be physically spread throughout the chip, irrespective of the network topology. Moreover, ElastiNoC has a fully distributed Built-In Self Testability (BIST) functionality that reaches high fault coverage with small test application time.

In the third research part of the thesis, we aim to reduce the clock-tree power consumption with Multi-bit Register (MBR) composition. MBR composition reduces the complexity of the clock tree by reducing the number of clock sinks, thus shortening the clock tree's wire length, which decreases the wire capacitance. We present a complete MBR composition flow that explores almost every aspect involved in the use of MBRs during physical design. We introduce an MBR decomposition step to remove some of the timing incompatibilities derived after the placement of the original netlist. Then, registers are merged to form larger MBRs as a result of an ILP-based optimization that uses new and realistic rules that determine register compatibility. We permit the use of incomplete MBRs to achieve additional MBR composition. The combined effect of these steps gives significant reduction in register count, and, together with the timing-driven sizing of the MBRs, it effectively reduces clock-pin capacitance. The benefits are shown across industrial benchmarks.

The final contribution of this thesis is a high-level systematic methodology for generating the appropriate traffic patterns that trigger the peak power consumption in a NoC. This work tries to identify a realistic estimation of a design's peak power consumption, as it directly impacts other salient system attributes, such as performance, implementation costs, battery life, and reliability. It introduces a fully-automated methodology to produce appropriate traffic and data patterns that cause peak power consumption within the NoC. This novel methodology uses an ILP-based mechanism to enable the power maximization of the majority of NoC components, irrespective of their differentiated design pa-

rameters. It can handle both heterogeneous and homogeneous NoCs and allows the generation of suitable traffic patterns within reasonable execution time.

7.2 Future Work

The proposed solutions in this thesis cover multiple aspects of NoC design, offering power-efficient approaches without sacrificing network performance. Although the presented solutions represent a mature set of optimized designs, there are still many opportunities left for future improvements.

As far as ElastiStore buffers are concerned, the main planned activity involves the addition of support to dynamically change the number of active VCs, and redistributing – in a cost-efficient manner – the available private buffering resources. At the moment, each VC statically receives at least one private buffer slot, even if it is totally inactive. These private buffers do not constitute a significant hardware overhead. However, our future plans are to let them be reused by others VCs in a programmable manner. This feature will certainly increase VC buffer utilization, but on the other hand, it will remove the simplicity of the proposed architecture and its low-cost nature. Our goal is to tackle efficiently this inherent cost-benefit tradeoff.

In parallel, regarding the ElastiNoC architecture, our goal is to test it under real industrial chip floorplans, where the distributed nature of the design will provide the most benefits. Even if the obtained experimental results are very promising, more realistic testcases will be valuable in understanding ElastiNoC's full potential. Also, in the same context, we plan to test ElastiNoC using higher-radix merge units. At the present moment, the VC-based merged units switch two inputs to one output, offering a balanced pipelined implementation. Higher-radix merge units will reduce the latency in terms of number hops, but will decrease the clock frequency of the design, increase the NoC's link wire-length, and, inevitably, also increase the routing congestion (more and longer wires need to be brought to the same place in the layout). Exploring this part of the design space will add more value to the proposed architecture.

MBR composition is an effective way to reduce clock-tree complexity. However, the overall benefits depend on the number and the distribution of the buffers in the NoC. The larger the number of buffers, the larger the expected savings in the clock and the larger the overall complexity of the NoC. Reducing the number of buffers and simplifying their design is a constant demand for NoCs that is also tackled in this thesis. Reducing the amount of buffering decreases the NoC's area and datapath power, but negatively affects performance (albeit slightly, if done correctly). On the other hand, MBR composition can save a lot from the clock-tree power, since it merges the buffers in to fewer but larger register cells. Therefore, MBR composition renders the need for buffer cost reduction less demanding, which, consequently, leaves room for lower NoC network performance degradation. Identifying the optimal balance point between the two approaches is the planned target of our future work on this topic.

Finally, the automatic generation methodology of peak-power traffic patterns focuses on maximizing the power consumption of the core of the NoC, i.e., the NoC routers and links. The derived traffic patterns can be applied as a corner-case scenario during hardware simulation/verification. Our future work will focus on extending this methodology to the generation of equivalent read/write transaction-level traffic, which can be reproduced at the software level. This effort involves the inclusion of AMBA AXI compatible network interfaces and taking into account their injection/ejection limitations and the support of out-of-order transactions.

Bibliography

[1] Niket Agarwal, Tushar Krishna, Li-Shiuan Peh, and Niraj K. Jha. GARNET: A detailed on-chip network model inside a full-system simulator. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*, pages 33–42, April 2009.

[2] James Balfour and William J. Dally. Design tradeoffs for tiled cmp on-chip networks. In *Proceedings of the 20th Annual International Conference on Supercomputing (ICS)*, pages 187–198, 2006.

[3] A.O. Balkan, Gang Qu, and U. Vishkin. Mesh-of-trees and alternative interconnection networks for single-chip parallelism. *IEEE Transactions on VLSI Systems*, 17(10):1419–1432, Oct 2009.

[4] A. Banerjee, P. T. Wolkotte, R. D. Mullins, S. W. Moore, and G. Smit. An energy and performance exploration of network-on-chip architectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 17(3):319–329, Mar 2009.

[5] Daniel U Becker, Nan Jiang, George Michelogiannakis, and William J Dally. Adaptive backpressure: Efficient buffer management for on-chip networks. In *Intern. Conf. on Computer Design*, pages 419–426, 2012.

[6] Davide Bertozzi, Luca Benini, and Giovanni De Micheli. Error control schemes for on-chip communication links: The energy-

reliability tradeoff. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(6):818–831, Nov 2006.

[7] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The PARSEC benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, pages 72–81, October 2008.

[8] Paul Bogdan, Radu Marculescu, and Siddharth Jain. Dynamic power management for multidomain system-on-chip platforms: An optimal control approach. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 18(4):46:1–46:20, Oct. 2013.

[9] S. Borkar and A. Chien. The future of microprocessors. *Commun. ACM*, 54(5):67–77, May 2011.

[10] Philippe Boucard and Luc Montperrus. Message switching system. US Patent 7639704, Arteris, 2009.

[11] Coen Bron and Joep Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, Sept 1973.

[12] Luca P. Carloni and Alberto L. Sangiovanni-Vincentelli. Coping with latency in soc design. *IEEE Micro*, 22(5):24–35, Sept 2002.

[13] Mario R. Casu and Paolo Giaccone. Rate-based vs delay-based control for dvfs in noc. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1096–1101, 2015.

[14] C. O. Chen, S. Park, T. Krishna, S. Subramanian, A. P. Chandrakasan, and L. Peh. Smart: A single-cycle reconfigurable noc for soc applications. In *Proceedings of the 2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 338–343, March 2013.

[15] L. Chen, D. Zhu, M. Pedram, and T. M. Pinkston. Power punch: Towards non-blocking power-gating of noc routers. In *Proceedings*

of the 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), pages 378–389, Feb 2015.

[16] Lizhong Chen and Timothy M. Pinkston. Nord: Node-router decoupling for effective power-gating of on-chip routers. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 270–281, 2012.

[17] Xi Chen, Zheng Xu, Hyungjun Kim, Paul V. Gratz, Jiang Hu, Michael Kishinevsky, Umit Ogras, and Raid Ayoub. Dynamic voltage and frequency scaling for shared resources in multicore processor designs. In *Proceedings of the 50th Annual Design Automation Conference (DAC)*, pages 114:1–114:7, 2013.

[18] Zhi-Wei Chen and Jin-Tai Yan. Routability-constrained multi-bit flip-flop construction for clock power reduction. *Integration, the VLSI Journal*, 46(3):290–300, June 2013.

[19] Wing-Kai Chow, Chak-Wa Pui, and Evangeline F. Y. Young. Legalization algorithm for multiple-row height standard cell design. In *Proceedings of the 53rd Annual Design Automation Conference (DAC)*, pages 83:1–83:6, 2016.

[20] Nicola Concer, Michele Petracca, and Luca P. Carloni. Distributed flit-buffer flow control for networks-on-chip. In *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, pages 215–220, 2008.

[21] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[22] J. Cortadella, M. Kishinevsky, and B. Grundmann. Synthesis of Synchronous Elastic Architectures. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 657–662, July 2006.

[23] W. J. Dally, J. Balfour, D. Black-Shaffer, J. Chen, R. C. Harting, V. Parikh, J. Park, and D. Sheffield. Efficient embedded computing. *Computer*, 41(7):27–32, July 2008.

[24] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference (DAC)*, pages 684–689, June 2001.

[25] William J. Dally, Chris Malachowsky, and Stephen W. Keckler. 21st century digital design tools. In *Proceedings of the 50th Annual Design Automation Conference (DAC)*, pages 94:1–94:6, 2013.

[26] C. R. Das, M. S. Yousif, V. Narayanan, Dongkook Park, C. Nicopoulos, Jongman Kim, C. R. Das, M. S. Yousif, V. Narayanan, Dongkook Park, C. Nicopoulos, and Jongman Kim. A gracefully degrading and energy-efficient modular router architecture for on-chip networks. In *Proceedings of the 33rd International Symposium on Computer Architecture (ISCA'06)*, pages 4–15, June 2006.

[27] R. Das, A. K. Mishra, C. Nicopoulos, D. Park, V. Narayanan, R. Iyer, M. S. Yousif, and C. R. Das. Performance and power optimization through data compression in network-on-chip architectures. In *Proceedings of the 2008 IEEE 14th International Symposium on High Performance Computer Architecture*, pages 215–225, Feb 2008.

[28] Reetuparna Das, Satish Narayanasamy, Sudhir K. Satpathy, and Ronald G. Dreslinski. Catnap: Energy proportional multiple network-on-chip. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA)*, pages 320–331, 2013.

[29] B. K. Daya, C. O. Chen, S. Subramanian, W. Kwon, S. Park, T. Krishna, J. Holt, A. P. Chandrakasan, and L. Peh. Scorpio: A 36-core research chip demonstrating snoopy coherence on a scalable mesh noc with in-network ordering. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pages 25–36, June 2014.

[30] Bhavya K. Daya, Li-Shiuan Peh, and Anantha P. Chandrakasan. Quest for high-performance bufferless nocs with single-cycle express paths and self-learning throttling. In *Proceedings of the 53rd*

Annual Design Automation Conference (DAC), pages 36:1–36:6, Aug 2016.

[31] Giorgos Dimitrakopoulos, Nikos Chrysos, and Costas Galanopoulos. Fast arbiters for on-chip network switches. In *IEEE International Conference on Computer Design (ICCD)*, pages 664–670, 2008.

[32] Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA)*, pages 365–376, 2011.

[33] John P. Fishburn. Clock skew optimization. *IEEE Transactions on Computers*, 39(7):945–951, July 1990.

[34] J. Flich, A. Mejia, P. Lopez, and J. Duato. Region-based routing: An efficient routing mechanism to tackle unreliable hardware in networks on chip. In *Proceedings of the 2007 ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, 2007.

[35] K. Ganesan, J. Jo, W. Bircher, D. Kaseridis, Z. Yu, and L. John. System-level max power (sympo): A systematic approach for escalating system-level power consumption using synthetic benchmarks. In *Int. Conf. on Parallel Architectures and Compilation Techniques (PACT)*, 2010.

[36] Karthik Ganesan and Lizy K. John. Maximum multicore power (mampo): An automatic multithreaded synthetic power virus generation framework for multicore systems. In *ACM Intern. Conf. for High Performance Computing, Networking, Storage and Analysis (SC)*, 2011.

[37] F. Gilabert, M. E. Gomez, S. Medardoni, and D. Bertozzi. Improved utilization of noc channel bandwidth by switch replication for cost-effective multi-processor systems-on-chip. In *Proceedings of the 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, pages 165–172, May 2010.

[38] R. Golshan and B. Haroun. A novel reduced swing cmos bus interface circuit for high speed low power vlsi systems. In *Proceedings*

of IEEE International Symposium on Circuits and Systems - ISCAS '94, volume 4, pages 351–354, May 1994.

[39] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu. A QoS-Enabled On-Die Interconnect Fabric for Kilo-Node Chips. *IEEE Micro*, 32(3), May 2012.

[40] Linley Gwennap. Low-power design using noc technology. The Linley Group, May 2015.

[41] Syed Minhaj Hassan and Sudhakar Yalamanchili. Centralized buffer router: A low latency, low power router for high radix nocs. In *IEEE/ACM International Symposium on Network on Chip*, April 2013.

[42] M. Hayenga, N. Enright Jerger, and M. Lipasti. Scarab: A single cycle adaptive routing and bufferless network. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 244–254, Dec 2009.

[43] Robert Hesse and Natalie Enright Jerger. Improving dvfs in nocs with coherence prediction. In *Proceedings of the 2015 Ninth International Symposium on Networks-on-Chip (NOCS)*, pages 24:1–24:8, 2015.

[44] Robert Hesse, Jeff Nicholls, and Natalie D. Enright Jerger. Fine-grained bandwidth adaptivity in networks-on-chip using bidirectional channels. In *6th IEEE/ACM Intern. Symp. on Networks on Chip*, pages 132–141, 2012.

[45] M. Hiraki, H. Kojima, H. Misawa, T. Akazawa, and Y. Hatano. Data-dependent logic swing internal bus architecture for ultralow-power lsi's. *IEEE Journal of Solid-State Circuits*, 30(4):397–402, April 1995.

[46] Byungchul Hong, Brian Huang, and Jonah Probell. The rubber jigsaw puzzle floorplanning for network-on-chip. In *Proceedings of the 2018 Synopsys Users Group (SNUG)*, SNUG '15, 2015.

[47] A. Hopkins. The functional safety imperative in automotive design. Whitepaper ARM, Sept 2016.

[48] Michael N. Horak, Steven M. Nowick, Matthew Carlberg, and Uzi Vishkin. A low-overhead asynchronous interconnection network for gals chip multiprocessors. In *Proc. of Symp. on Networks-on-Chip*, pages 43–50, 2010.

[49] M. Horowitz, E. Alon, D. Patil, S. Naffziger, Rajesh Kumar, and K. Bernstein. Scaling, power, and the future of cmos. In *IEEE InternationalElectron Devices Meeting, 2005. IEDM Technical Digest.*, pages 7–15, Dec 2005.

[50] Yatin Hoskote, Sriram Vangal, Arvind Singh, Nitin Borkar, and Shekhar Borkar. A 5-GHz mesh interconnect for a teraflops processor. *IEEE Micro*, 27(5):51–61, Sept 2007.

[51] Wenting Hou, Dick Liu, and Pei-Hsin Ho. Automatic register banking for low-power clock trees. In *Proceedings of the 2009 10th International Symposium on Quality of Electronic Design*, pages 647–652, 2009.

[52] Michael S. Hsiao, Elizabeth M. Rudnick, and Janak H. Patel. Dynamic state traversal for sequential circuit test generation. *ACM Trans. Des. Autom. Electron. Syst.*, 5(3):548–565, July 2000.

[53] C. Hsu, Y. Chang, and M. P. Lin. Crosstalk-aware power optimization with multi-bit flip-flops. In *Proceedings of the 17th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 431–436, Jan 2012.

[54] W. Huang, K. Rajamani, M. R. Stan, and K. Skadron. Scaling with design constraints: Predicting the future of big chips. *IEEE Micro*, 31(4):16–29, July 2011.

[55] Arteris Inc. A comparison of network-on-chip and busses. `https://www.design-reuse.com/articles/10496/a-comparison-of-network-on-chip-and-busses.html`.

[56] Gurobi Optimization Inc. Gurobi optimizer reference manual. https://www.gurobi.com/documentation/7.0/refman/index.html.

[57] Wind River Inc. Simics, product overview.

[58] Charles Janac. Physical interconnect aware network optimizer: Interconnect physical optimization. In *Proceedings of the 2018 on International Symposium on Physical Design (ISPD)*, March 2018.

[59] Natalie Enright Jerger, Tushar Krishna, and Li-Shiuan Peh. *On-Chip Networks: Second Edition*. Morgan & Claypool Publishers, 2nd edition, 2017.

[60] I. H. Jiang, C. Chang, and Y. Yang. Integra: Fast multibit flip-flop clustering for clock power saving. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(2):192–204, Feb 2012.

[61] Y. Jin, E. J. Kim, and K. H. Yum. Peak power control for a QoS capable on-chip network. In *International Conference on Parallel Processing (ICPP)*, pages 585–592, 2005.

[62] Yuho Jin, K. H. Yum, and E. J. Kim. Adaptive data compression for high-performance low-power on-chip networks. In *Proceedings of the 2008 41st IEEE/ACM International Symposium on Microarchitecture*, pages 354–363, Nov 2008.

[63] Andrew B. Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 423–428, 2009.

[64] Andrew B. Kahng, Jiajia Li, and Lutong Wang. Improved flop tray-based design implementation for power reduction. In *Proceedings of the 35th International Conference on Computer-Aided Design (ICCAD)*, pages 20:1–20:8, 2016.

[65] M.R. Kakoee, V. Bertacco, and L. Benini. A distributed and topology-agnostic approach for on-line noc testing. In *Proceed-*

ings of the ACM/IEEE International Symposium on Networks-on-Chip (NOCS, pages 113–120, May 2011.

[66] G. Kim, J. Kim, and S. Yoo. Flexibuffer: reducing leakage power in on-chip network routers. In *Proceedings of the 48th Design Automation Conference, DAC*, pages 936–941, 2011.

[67] Hyungjun Kim, Pritha Ghoshal, Boris Grot, Paul V. Gratz, and Daniel A. Jiménez. Reducing network-on-chip energy consumption through spatial locality speculation. In *Proceedings of the 5th ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, pages 233–240, 2011.

[68] J. Kim, J. Balfour, and W. J. Dally. Flattened butterfly topology for on-chip networks. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 172–182, 2007.

[69] John Kim. Low-cost router microarchitecture for on-chip networks. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 255–266, Dec. 2009.

[70] N. S. Kim, T. Austin, T. Mudge, and D. Grunwald. *Challenges for Architectural Level Power Modeling*. Springer, 2002.

[71] Johann Knechtel and Jens Lienig. Physical design automation for 3d chip stacks: Challenges and solutions. In *Proceedings of the 2016 on International Symposium on Physical Design (ISPD)*, pages 3–10, 2016.

[72] A. K. Kodi, A. Sarathy, and A. Louri. ideal: Inter-router dual-function energy and area-efficient links for network-on-chip (noc) architectures. In *Proc. of Intern Symp. on Comp. Architecture*, pages 241–250, 2008.

[73] V. Kontorinis, A. Shayan, D. Tullsen, and R. Kumar. Reducing peak power with a table-driven adaptive processor core. In *Intern. Symp. on Microarchitecture*, pages 189–200, 2009.

[74] E. D. Kyriakis-Bitzaros and S. S. Nikolaidis. Design of low power cmos drivers based on charge recycling. In *Proceedings of 1997 IEEE International Symposium on Circuits and Systems. Circuits and Systems in the Information Age ISCAS '97*, volume 3, pages 1924–1927, June 1997.

[75] Mark Lapedus. Interconnect challenges rising. Semiconductor engineering, June 2016.

[76] K. Latif, A.-M. Rahmani, Liang Guang, T. Seceleanu, and H. Tenhunen. Pvs-noc: Partial virtual channel sharing noc architecture. In *Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 470–477, Feb 2011.

[77] H. K. Lee and D. S. Ha. Hope: An efficient parallel fault simulator for synchronous sequential circuits. In *Proc. of the ACM/IEEE Design Automation Conference (DAC)*, pages 336–340, 1992.

[78] T. Lee, D. Z. Pan, and J. Yang. Clock network optimization with multibit flip-flop generation considering multicorner multimode timing constraint. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):245–256, Jan 2018.

[79] M. P. Lin, C. Hsu, and Y. Chen. Clock-tree aware multibit flip-flop generation during placement for power optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(2):280–292, Feb 2015.

[80] Sean Shih-Ying Liu, Wan-Ting Lo, Chieh-Jui Lee, and Hung-Ming Chen. Agglomerative-based flip-flop merging and relocation for signal wirelength and clock tree optimization. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 18(3):40:1–40:20, July 2013.

[81] Shih-Chuan Lo, Chih-Cheng Hsu, and Mark Po-Hung Lin. Power optimization for clock network with clock gate cloning and flip-flop merging. In *Proceedings of the 2014 on International Symposium on Physical Design (ISPD)*, pages 77–84, 2014.

[82] Ye Lu, Changlin Chen, John V. McCanny, and Sakir Sezer. Design of interlock-free combined allocators for networks-on-chip. In *EEE 25th International SOC Conference (SoCC)*, pages 358–363, 2012.

[83] S. Ma, N. Enright Jerger, and Z. Wang. Whole Packet Forwarding: Efficient Design of Fully Adaptive Routing Algorithms for Networks-on-Chip. In *Proc. of the Intern. Symp. on High Performance Computer Architecture*, pages 467–478, Feb. 2012.

[84] Sheng Ma, Libo Huang, Mingche Lai, Wei Shi, and Zhiying Wang. *Networks-on-Chip: From Implementations to Programming Paradigms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2014.

[85] Milo M. K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *ACM SIGARCH Computer Architecture News*, 33(4):92–99, Nov 2005.

[86] H. Matsutani, Y. Hirata, M. Koibuchi, K. Usami, H. Nakamura, and H. Amano. A multi-vdd dynamic variable-pipeline on-chip router for cmps. In *Proceedings of the 17th Asia and South Pacific Design Automation Conference (ASPDAC)*, pages 407–412, Jan 2012.

[87] Mentor Graphics. Logic BIST Applications and Usage Whitepaper. In *Silicon Test and Yield Analysis*, 2010.

[88] G. De Micheli, C. Seiculescu, S. Murali, L. Benini, F. Angiolini, and A. Pullini. Networks on chips: From research to products. In *Proceedings of the 47th Annual Design Automation Conference (DAC)*, 2010.

[89] G. Michelogiannakis, J. Balfour, and W. J. Dally. Elastic buffer flow control for on-chip networks. In *IEEE Int. Symp. on High Performance Computer Architecture*, 2009.

[90] G. Michelogiannakis and W.J. Dally. Elastic buffer flow control for on-chip networks. *IEEE Trans. on Computers*, 62(2), Feb. 2013.

[91] G. Michelogiannakis, N.Jiang, D.Becker, and W.J.Dally. Packet chaining: Efficient single-cycle allocation for on-chip networks. In *Proc. IEEE/ACM In. Symp. on Microarchitecture (MICRO)*, pages 83–94, 2011.

[92] Asit K. Mishra, Reetuparna Das, Soumya Eachempati, Ravi Iyer, N. Vijaykrishnan, and Chita R. Das. A case for dynamic frequency tuning in on-chip networks. In *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 292–303, 2009.

[93] Asit K. Mishra, N. Vijaykrishnan, and Chita R. Das. A case for heterogeneous on-chip interconnects for cmps. In *Proc. of the intern. symp. on Computer architecture*, ISCA '11, pages 389–400, 2011.

[94] Konstantin Moiseev, Avinoam Kolodny, and Shmuel Wimer. *Multi-Net Optimization of VLSI Interconnect*. Springer Publishing Company, Inc., 2014.

[95] H. Moon and T. Kim. Design and allocation of loosely coupled multi-bit flip-flops for power reduction in post-placement optimization. In *Proceedings of the 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 268–273, Jan 2016.

[96] Thomas Moscibroda and Onur Mutlu. A case for bufferless routing in on-chip networks. In *Proceedings of the 36th International Symposium on Computer Architectur*. IEEE, June 2009.

[97] Robert D. Mullins. Minimising dynamic power consumption in on-chip networks. In *Proceedings of the 2006 International Symposium on System-on-Chip*, pages 1–4, Nov 2006.

[98] Robert D. Mullins, Andrew F. West, and Simon W. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the International Symposium on Computer Architecture*, pages 188–197, June 2004.

[99] K. Najeeb, V. Vardhan, R. Konda, S. Kumar, S. Hari, V. Kamakoti, and V. M. Vedula. Power virus generation using behavioral mod-

els of circuits. In *Proc. of the 25th IEEE VLSI Test Symposium (VTS'07)*, pages 35–42, May 2007.

[100] T. Nakamura, H. Matsutani, M. Koibuchi, K. Usami, and H. Amano. Fine-grained power control using a multi-voltage variable pipeline router. In *Proceedings of the 2012 IEEE 6th International Symposium on Embedded Multicore SoCs*, pages 59–66, Sept 2012.

[101] M. H. Neishaburi and Zeljko Zilic. Reliability aware noc router architecture using input channel buffer sharing. In *Proceedings of the 19th ACM Great Lakes Symposium on VLSI*, pages 511–516, 2009.

[102] Nan Ni, Marius Pirvu, and Laxmi N. Bhuyan. Circular buffered switch design with wormhole routing and virtual channels. In *ICCD*, pages 466–473, 1998.

[103] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das. Vichar: A dynamic virtual channel regulator for network-on-chip routers. In *Proceedings of the 2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, pages 333–346, Dec 2006.

[104] Ritesh Parikh, Reetuparna Das, and Valeria Bertacco. Power-aware nocs through routing and topology reconfiguration. In *Proceedings of the 51st Annual Design Automation Conference (DAC)*, pages 162:1–162:6, 2014.

[105] J. Park, B. W. O'Krafka, S. Vassiliadis, and J. Delgado-Frias. Design and evaluation of a damq multiprocessor network with self-compacting buffers. In *Proceedings of the 1994 ACM/IEEE Conference on Supercomputing*, Supercomputing '94, pages 713–722. IEEE Computer Society Press, 1994.

[106] Sunghyun Park. *Low-Swing Signaling for Energy Efficient on-chip Networks*. PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2011.

[107] Sudeep Pasricha and Nikil Dutt. *On-Chip Communication Architectures: System on Chip Interconnect*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[108] Li-Shiuan Peh and William J. Dally. A delay model and speculative architecture for pipelined routers. In *Proceedings of the 7th IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 255–266, Jan 2001.

[109] M. Petracca and L. Carloni. The benefits of using clock gating in the design of networks-on-chip. *Columbia University Computer Science Technical Reports*, 2011.

[110] J. Philip, S. Kumar, E. Norige, M. Hassan, and S. Mitra. Automatic construction of deadlock free interconnects. US Patent 9244880, Netspeed Systems, 2016.

[111] S. Polfliet, F. Ryckbosch, and L. Eeckhout. Automated full-system power characterization. *IEEE Micro*, pages 46–59, May 2011.

[112] J. Postman, T. Krishna, C. Edmonds, L. Peh, and P. Chiang. Swift: A low-power network-on-chip implementing the token flow control router architecture with swing-reduced interconnects. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(8):1432–1446, Aug 2013.

[113] A. Rahimi, I. Loi, M.R. Kakoee, and L. Benini. A fully-synthesizable single-cycle interconnection network for shared-l1 processor clusters. In *DATE*, pages 1–6, March 2011.

[114] P. Ren, X. Ren, S. Sane, M. A. Kinsy, and N. Zheng. A deadlock-free and connectivity-guaranteed methodology for achieving fault-tolerance in on-chip networks. *IEEE Transactions on Computers*, 65(2):353–366, Feb 2016.

[115] A. Roca, J. Flich, and G. Dimitrakopoulos. Desa: Distributed elastic switch architecture for efficient networks-on-fpgas. In *Field Programmable Logic and Applications (FPL)*, pages 394–399, Aug 2012.

[116] Antoni Roca, Carles Hernndez, Jose Flich, Federico Silla, and Jose Duato. Silicon-aware distributed switch architecture for on-chip networks. *Journal of Systems Architecture*, 59(7):505 – 515, 2013.

[117] Tajana Simunic Rosing, Kresimir Mihic, and Giovanni De Micheli. Power and reliability management of socs. *IEEE Trans. VLSI*, 15(4), April 2007.

[118] Xavier Van Ruymbeke. Benefits of network on chip fabrics for late stage design changes, adaptive qos and floorplan selection. ChipEX, April 2014.

[119] P. Salihundam, S. Jain, T. Jacob, S. Kumar, V. Erraguntla, Y. Hoskote, S. Vangal, G. Ruhl, P. Kundu, and N. Borkar. A 2Tb/s 6x4 Mesh Network with DVFS and 2.3Tb/s/W router in 45nm CMOS. In *Proceedings of the 2010 Symposium on VLSI Circuits*, pages 79–80, June 2010.

[120] Ahmad Samih, Ren Wang, Anil Krishna, Christian Maciocco, Charlie Tai, and Yan Solihin. Energy-efficient interconnect via router parking. In *Proceedings of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, HPCA '13, pages 508–519, 2013.

[121] Prashant Saxena. *Routing Congestion in VLSI Circuits: Estimation and Optimization*. Springer Publishing Company, Inc., 2007.

[122] I. Seitanidis, G. Dimitrakopoulos, P. Mattheakis, L. Masse-Navette, and D. Chinnery. Timing driven incremental multi-bit register composition using a placement-aware ilp formulation. In *Proceedings of the 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2017.

[123] I. Seitanidis, G. Dimitrakopoulos, P. Mattheakis, L. Masse-Navette, and D. Chinnery. Timing-driven and placement-aware multi-bit register composition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, to appear, 2018.

[124] I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos. Powermax: an automated methodology for generating peak-power traffic in networks-on-chip. In *Proceedings of the 2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pages 1–8, Aug 2016.

[125] I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos. Automatic generation of peak-power traffic for networks-on-chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, to appear, 2018.

[126] I. Seitanidis, A. Psarras, K. Chrysanthou, C. Nicopoulos, and G. Dimitrakopoulos. Elastistore: Flexible elastic buffering for virtual-channel-based networks on chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(12):3015–3028, Dec. 2015.

[127] I. Seitanidis, A. Psarras, G. Dimitrakopoulos, and C. Nicopoulos. Elastistore: An elastic buffer architecture for network-on-chip routers. In *Proceedings of the Conference on Design, Automation and Test in Europe*, Mar 2014.

[128] I Seitanidis, A Psarras, E Kalligeros, C Nicopoulos, and G Dimitrakopoulos. ElastiNoC: A self-testable distributed vc-based network-on-chip architecture. In *Proceedings of the 2014 Eighth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pages 135–142, Sept 2014.

[129] Li Shang, Li-Shiuan Peh, and Niraj K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *Proceedings of the 9th International Symposium on High-Performance Computer Architecture (HPCA)*, HPCA '03, pages 91–102, Feb 2003.

[130] Rupesh S. Shelar. An efficent clustering algorithm for low power clock tree synthesis. In *Proceedings of the 2007 International Symposium on Physical Design (ISPD)*, pages 181–188, 2007.

[131] Y. Shyu, J. Lin, C. Huang, C. Lin, Y. Lin, and S. Chang. Effective and efficient approach for power reduction by using multi-bit flip-flops. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(4):624–635, April 2013.

[132] V. Soteriou and L. Peh. Exploring the design space of self-regulating power-aware on/off interconnection networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(3):393–408, March 2007.

[133] V. Soteriou and Li-Shiuan Peh. Dynamic power management for power optimization of interconnection networks using on/off links. In *Proceedings of the 11th Symposium on High Performance Interconnects, 2003.*, pages 15–20, Aug 2003.

[134] A. Strano, C. Gomez, D. Ludovici, M. Favalli, M. E. Gomez, and D. Bertozzi. Exploiting network-on-chip structural redundancy for a cooperative and scalable built-in self-test architecture. In *Proceedings of the 2011 Design, Automation Test in Europe (DATE)*, pages 661–666, March 2011.

[135] W. Su, J. S. Shen, and P. A. Hsiung. Network-on-Chip Router Design with Buffer-Stealing. In *ASP-Design Automation Conference*, 2011.

[136] Y. Tamir and G. L. Frazier. High-performance multiqueue buffers for VLSI communication switches . In *Proc. of the 15th Annual International Symposion on Computer Architecture (ISCA)*, pages 343–354, 1988.

[137] B. Towles and W. Dally. Worst-case traffic for oblivious routing functions. In *ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–8, 2002.

[138] A. T. Tran and B. M. Baas. RoShaQ: High-performance on-chip router with shared queues. In *IEEE International Conf. on Computer Design*, pages 232–238, Oct. 2011.

[139] Chang-Cheng Tsai, Yiyu Shi, Guojie Luo, and Iris Hui-Ru Jiang. Ff-bond: Multi-bit flip-flop bonding at placement. In *Proceedings of the 2013 ACM International Symposium on Physical Design (ISPD)*, pages 147–153, 2013.

[140] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS. *IEEE Journal of Solid-State Circuits*, 43:6–20, Jan 2008.

[141] H. Wang and L.S. Pehand S. Malik. Power-driven design of router microarchitectures in on-chip networks. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 105–116, 2003.

[142] S. Wang, Y. Liang, T. Kuo, and W. Mak. Power-driven flip-flop merging and relocation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(2):180–191, Feb 2012.

[143] Neil Weste and David Harris. *CMOS VLSI Design a Circuits and Systems Perspective*. Addison Wesley (3rd Edition), 2010.

[144] Shmuel Wimer and Israel Koren. Design flow for flip-flop grouping in data-driven clock gating. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(4):771–778, April 2014.

[145] Gang Wu, Yue Xu, Dean Wu, Manoj Ragupathy, Yu-yen Mo, and Chris Chu. Flip-flop clustering by weighted k-means algorithm. In *Proceedings of the 53rd Annual Design Automation Conference (DAC)*, pages 82:1–82:6, 2016.

[146] Q. Wu, Q. Qiu, and M. Pedram. Estimation of peak power dissipation in vlsi circuits using the limiting distributions of extreme order statistics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(8):942–956, Aug 2001.

[147] Kretchmer Y. Using multi-bit register inference to save area and power: The good the bad and the ugly. *EE Times Asia*, 2001.

[148] Y. Yao and Z. Lu. Dvfs for nocs in cmps: A thread voting approach. In *Proceedings of the 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 309–320, March 2016.

[149] Y. Yao and Z. Lu. Memory-access aware dvfs for network-on-chip in cmps. In *Proceedings of the 2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1433–1436, March 2016.

[150] Dongyoun Yi and Taewhan Kim. Allocation of multi-bit flip-flops in logic synthesis for power optimization. In *Proceedings of the 35th*

International Conference on Computer-Aided Design (ICCAD), pages 33:1–33:6, 2016.

[151] Y.J. Yoon, N. Concer, M. Petracca, and L. P. Carloni. Virtual channels and multiple physical networks: Two alternatives to improve noc performance. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(12):1906–1919, Dec 2013.

[152] H. Zhang, V. George, and J. M. Rabaey. Low-swing on-chip signaling techniques: Effectiveness and robustness. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(3):264–272, June 2000.