ElastiNoC: A Self-Testable Distributed VC-based Network-on-Chip Architecture

I. Seitanidis*, A. Psarras*, E. Kalligeros†, C. Nicopoulos‡ and G. Dimitrakopoulos*
*Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece
†Information & Communication Systems Engineering, University of the Aegean, Samos, Greece
‡Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus

Abstract—**Network-on-Chip (NoC) design tries to keep a balance between network performance and physical implementation flexibility. The adoption of Virtual Channels (VC) holds promise for scalable NoC design. VCs allow for traffic separation and isolation, enable deadlock avoidance and improve network performance. In this paper, 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. The proposed architecture utilizes an efficient buffering strategy 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. 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.**

I. Introduction

The design of scalable Network-on-Chip (NoC) architectures calls for new implementations that achieve high throughput and low-latency operation, without exceeding the stringent area-energy constraints of modern Systems-on-Chip (SoC) [1].

In terms of network functionality, Virtual Channels (VCs) – which allow a physical channel to be used in a time-multiplexed manner by different traffic flows, provided that each flow owns a separate buffer space – are an already proven solution [2]. Architectures supporting the use of VCs (1) enable traffic separation and isolation by assigning different traffic classes to different VCs, (2) improve performance, and (3) reduce on-chip physical routing congestion, by trading off physical channel width with a number of VCs, thereby creating a more layout-flexible SoC architecture.

VCs are also instrumental for the correct operation of higher-level mechanisms. For instance, protocol-level restrictions in Chip Multi-Processors (CMP) employing directory-based cache coherence necessitate the use of VCs. Coherence protocols require isolation between the various message classes to avoid protocol-level deadlocks. For example, the MOESI directory-based cache coherence protocol requires at least three virtual networks to prevent protocol-level deadlocks. A virtual network comprises of one VC (or a group of VCs) that handles a specific message class of the protocol [3].

Traditional VC-based NoC architectures focus mostly on microarchitectural improvements to the router's internal organization and pipeline structure [4], [5]. Prior research has explored salient router attributes, such as appropriate allocation policies [6], as well as the optimization of the associated VC buffering structures [7], [8], [9], concentrating mostly on buffer sharing and related flow control strategies. A complete overview of routers' microarchitecture can be found in [10].

In this paper, 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 a novel *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 [11], [12], [13], [14], [15]. 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 paper is the first distributed VC-based router implementation that supports this form of modularity. In summary, the main contributions of this work are:

- The introduction of a new architectural paradigm for VC-based NoC routers, called *ElastiNoC*, which enables the modular construction of pipelined routers. The resulting design, presented in Section III, yields highly-scalable NoC implementations.
- 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.
- The augmentation of ElastiNoC with self-testability ca-

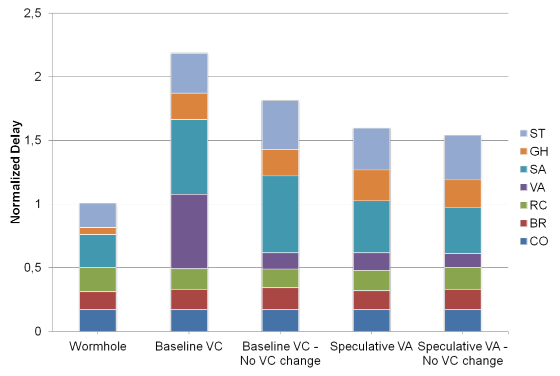


Fig. 1. The delay of representative single-cycle 5-port NoC routers with 64-bit wide ports and 4 VCs per input in the case of VC-based routers. The results are normalized to the delay of a wormhole-based router (i.e., no VC support). "No VC change" means that packets do not change VC; their VC is decided upon injection and remains the same until they reach their destination.

pabilities, as presented in Section IV. The routers are able to conduct testing sessions in a modular manner over multiple phases, that achieve high fault coverage (in excess of 99%). Self testing imbues ElastiNoC with a valuable (albeit often ignored) asset, which greatly enhances its viability to much larger future designs.

The experimental results presented in Section V – based on both network simulations and standard-cell-based hardware synthesis implementations – validate the efficacy and efficiency of ElastiNoC. 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.

II. 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 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 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 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

$$T_{CYC} \geq \frac{D}{k} + c \tag{1}$$

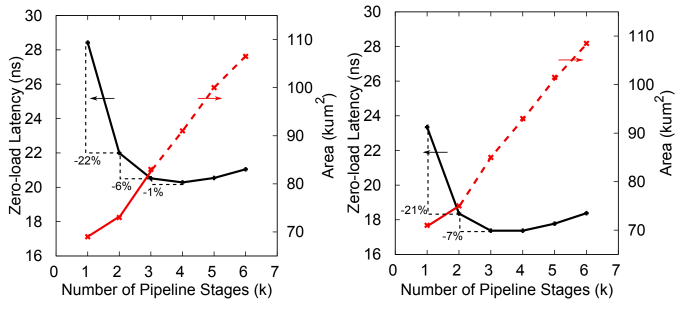
By increasing the pipeline stages of a router, its clock frequency can increase, thereby leading to faster implementations. At the same time, however, each flit spends more cycles inside each router, before moving to the next one. Therefore, the depth of the pipeline cannot be decided in isolation; the 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

$$T = H\,(k + 1) + P - 1 \tag{2}$$

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].

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



(a) (b)

Fig. 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.

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:

$$T_{ABS}(k) = T \times T_{CYC} = (H(k+1) + P - 1)\left(\frac{D}{k} + c\right) \quad (3)$$

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.

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 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 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.

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

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 [16]) 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 paper.

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 floorplanning and placement tools with more freedom in generating optimal layout configurations.

III. ElastiNoC: Modular VC-based Architecture

Any network topology, from single-stage crossbars to arbitrary cubes, meshes, or butterfly-based structures can be implemented by decomposing the switch operation to primitive merge and split functions. In this paper, 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.

A. Modular router construction

The fundamental primitive of ElastiNoC, called the Merge Unit (MU), consists of two inputs and one output. Its goal is

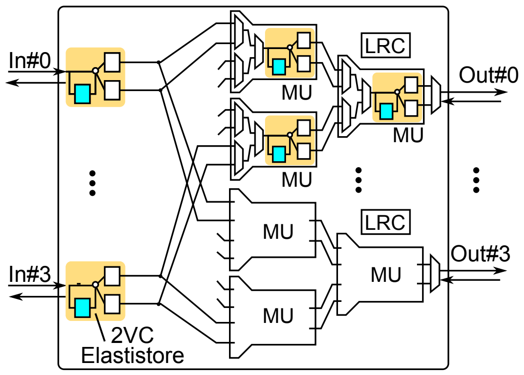


Fig. 3. The modular construction of an example ElastiNoC 4×4 VC-based router using the proposed MU primitive that supports 2 VCs.

to switch and buffer locally the flits of two inputs that belong to different VCs. Buffering is done via ElastiStore units [17], 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 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 II 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 [18]. When such customization is utilized, significant area savings are expected, due to the removal of both buffering

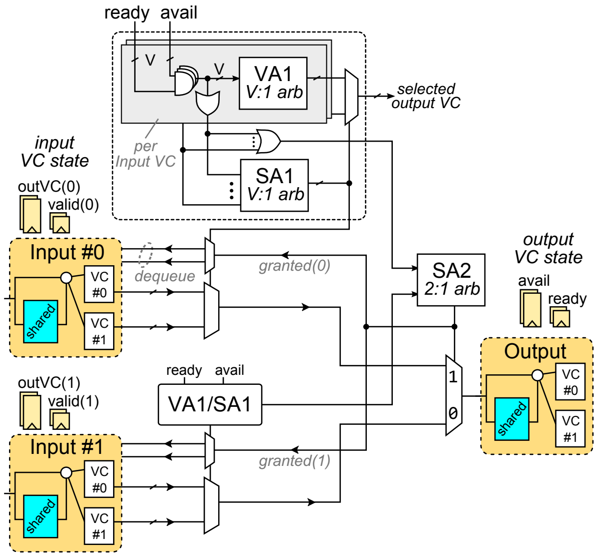


Fig. 4. The fundamental ElastiNoC primitive, the Merge Unit (MU). The diagram depicts the per-input and per-output multiplexers together with the combined allocation logic (SA1, SA2) that runs in parallel to VA1.

and logic resources. On the contrary, when such optimizations are performed in traditional VC-based routers, only parts of the crossbar and switch allocation logic are reduced, while the VC allocation logic and buffering, which are responsible fot the majority of the router's area, are not affected.

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

B. 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. Since switching is achieved by connecting several MUs in series (as illustrated in Figure 3), the buffers presented at the input of Figure 4 are actually the output buffers of the previous MUs.

1) Allocation and Switching Logic: 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 [19], 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 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 well. 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.

2) Buffering: 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. Since each ElastiStore acts both as an input VC buffer for downstream connections and as an output VC buffer for upstrem connections, it keeps all the necessary variables per VC, as shown in Figure 5.

When an input VC is ready to accept a new flit it asserts

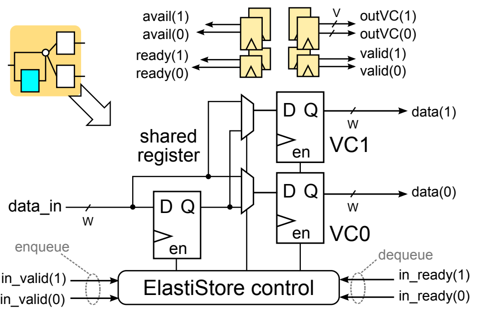


Fig. 5. The ElastiStore-based buffer architecture. ElastiStore consists of merely one single-flit register per VC, plus one additional single-flit register that is shared dynamically between VCs.

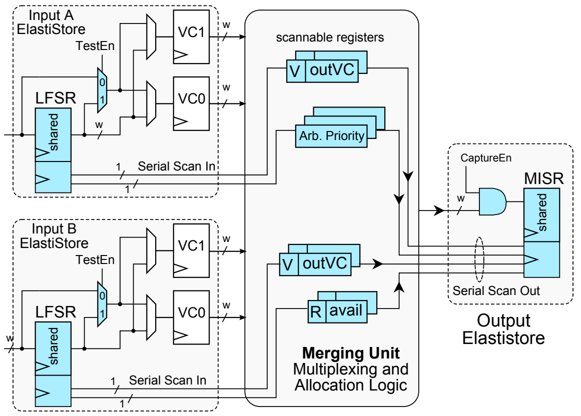


Fig. 6. BIST enhancements of a merge unit by incorporating the logic of LSFR/MISR within that of the shared buffer at each ElastiStore.

the ready(i) signal. An in_valid(i) signal means that valid data for the ith VC has arrived at the ElastiStore buffer. A transfer occurs for VC#i, when both signals are asserted. Flits arriving at each ElastiStore are stored in the main register of each VC. The shared register is used only to host any in-flight arriving flit for a stalled VC. This may happen, since back-pressure signals are first registered, before being sent upstream, to guarantee maximum scalability in terms of delay. Filling the main register with data does not render the corresponding VC unavailable, since there is still extra space at the shared register. A VC stops being ready when its main and the shared buffer are full. When the allocation logic of MU dequeues a flit from an input VC, the main register of this VC is refilled either with new arriving data from the input, or with data possibly stored in the shared buffer. This automatic data movement from the shared to main buffer avoids any bubbles in the flow of flits across the MUs and achieves maximum throughput.

IV. 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 [20], [21]. The distributed structure of ElastiNoC does not match well with traditional core-level BIST architectures [22]. 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 6, 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 multiplexers that multiplex their data/control inputs with the outputs of the shared buffers that act as LFSRs, as depicted in Figure 6. During testing (TestEnable=1) the original bypass multiplexers of Elastistores get the same value, which guarantees that 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 6 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). To drive the scan chains and compact the captured responses, we augmented the shared buffer with some additional flip-flops, as shown in Figure 6.

During each test phase, multiple MUs are independently tested in parallel. For example, the 4×4 router depicted in Figure 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 compacting 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 V. 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.

V. 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.

A. Hardware complexity

The proposed ElastiNoC routers (using lookahead RC) were implemented in VHDL and 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 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 [23]. 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 7.

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

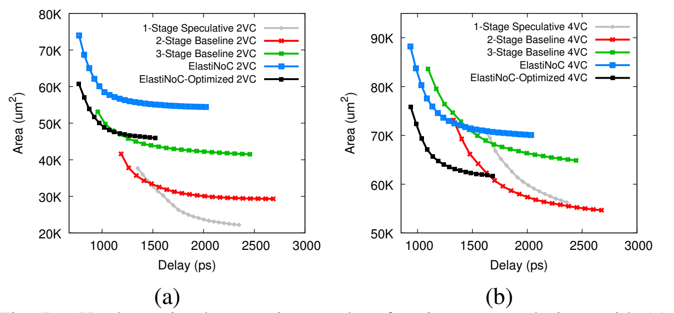


Fig. 7. Hardware implementation results of various router designs with (a) 2 VCs, and (b) 4 VCs per port.

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 [24], [25], 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 area-delay performance of ElastiNoC-Optimized and assuming that the packets entering the network are not allowed to change VC as done in [26]. 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.

B. 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 [27] 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 I.

TABLE I
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%

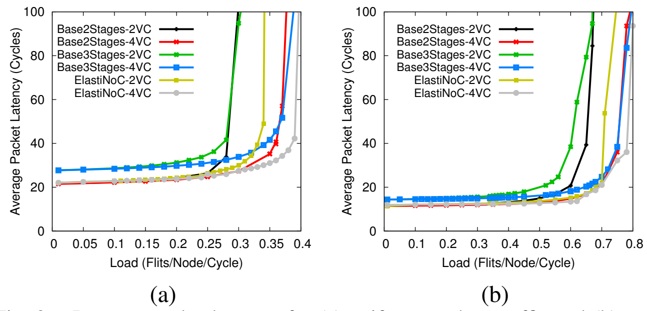


Fig. 8. Latency vs. load curves for (a) uniform random traffic and (b) non-uniform localized traffic. Network traffic from real applications is estimated to lie in-between these two synthetic traffic patterns.

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 examined MU (approximately 1-1.5%) have been reported as untestable by the Strategate sequential ATPG tool [28], 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 IV 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.

C. 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$\times$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 [29].

Figure 8 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 [30] (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 (omitted due to space limitations), because of the high serialization latency imposed by the narrower channels in each physical network.

VI. Conclusions

Complex MPSoCs built for embedded systems, as well as modern CMPs, have adopted NoC technology for their internal connectivity. This approach was originally utilized to tackle the physical integration and verification complexity of large systems. Future NoCs go beyond requirements related to physical implementation and actively participate in delivering high performance, quality of service, and dynamic adaptivity at a minimum area overhead. 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 paper, 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.

References

[1] J. Handy, "NoC interconnect improves SoC economics," *Objective analysis - Semiconductor market research*, 2011.

[2] W. J. Dally, "Virtual-Channel Flow Control," in *Proc. of the 17th Annual Intern. Symp. on Computer Architecture (ISCA)*, May 1990, pp. 60–68.

[3] M. M. K. Martin and et al., "Multifacet's general execution-driven multiprocessor simulator (gems) toolset," *SIGARCH Comput. Archit. News*, vol. 33, no. 4, pp. 92–99, Nov. 2005.

[4] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *HPCA*, 2001.

[5] R. D. Mullins, A. F. West, and S. W. Moore, "Low-latency virtual-channel routers for on-chip networks," in *ISCA*, 2004, pp. 188–197.

[6] 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)*, 2011, pp. 83–94.

[7] A. T. Tran and B. M. Baas, "RoShaQ: High-performance on-chip router with shared queues," in *IEEE ICCD*, 2011, pp. 232–238.

[8] D. U. Becker, "Adaptive backpressure: Efficient buffer management for on-chip networks," in *IEEE ICCD*, 2012.

[9] S. M. Hassan and S. Yalamanchili, "Centralized buffer router: A low latency, low power router for high radix nocs," in *IEEE/ACM Intern. Symp. on Network on Chip*, April 2013.

[10] G. Dimitrakopoulos, A. Psarras, and I. Seitanidis, *Microarchitecture of Network-on-Chip Routers: A designer's perspective*. Spinger, 2014.

[11] A. Roca, C. Hernndez, J. Flich, F. Silla, and J. Duato, "Silicon-aware distributed switch architecture for on-chip networks," *Journal of Systems Architecture*, vol. 59, no. 7, pp. 505 – 515, 2013.

[12] A. Balkan, G. Qu, and U. Vishkin, "Mesh-of-trees and alternative interconnection networks for single-chip parallelism," *IEEE Transactions on VLSI Systems*, vol. 17, no. 10, pp. 1419–1432, Oct 2009.

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

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

[15] M. N. Horak, S. M. Nowick, M. Carlberg, and U. Vishkin, "A low-overhead asynchronous interconnection network for gals chip multiprocessors," in *Proc. of Symp. on Networks-on-Chip*, 2010, pp. 43–50.

[16] J. Kim, J. Balfour, and W. J. Dally, "Flattened butterfly topology for on-chip networks," in *Proc. of In. Symp. on Microarchitecture*, 2007.

[17] I. Seitanidis, A. Psarras, G. Dimitrakopoulos, and C. Nicopoulos, "Elastistore: An elastic buffer architecture for network-on-chip routers," in *Proc. of Design Automation and Test in Europe (DATE)*, Mar. 2014.

[18] 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 *Intern. Symp. on Networks on Chip (NOCS)*, 2007.

[19] Y. Lu, C. Chen, J. V. McCanny, and S. Sezer, "Design of interlock-free combined allocators for networks-on-chip," in *EEE 25th International SOC Conference (SoCC)*, 2012, pp. 358–363.

[20] M. Kakoee, V. Bertacco, and L. Benini, "A distributed and topology-agnostic approach for on-line noc testing," in *IEEE/ACM Intern. Symp. on Networks on Chip (NoCS)*, May 2011, pp. 113–120.

[21] A. Strano and et al., "Exploiting network-on-chip structural redundancy for a cooperative and scalable built-in self-test architecture," in *DATE*, 2011, pp. 661–666.

[22] Mentor Graphics, "Logic BIST Applications and Usage Whitepaper," in *Silicon Test and Yield Analysis*, 2010.

[23] G. Dimitrakopoulos, N. Chrysos, and C. Galanopoulos, "Fast arbiters for on-chip network switches," in *IEEE ICCD*, 2008, pp. 664–670.

[24] P. Salihundam and et al., "A 2Tb/s 6x4 Mesh Network with DVFS and 2.3Tb/s/W router in 45nm CMOS," in *Symp. VLSI Circuits*, 2010.

[25] S. R. Vangal and et al., "An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS," *IEEE JSSC*, vol. 43, pp. 6–20, Jan. 2008.

[26] F. Gilabert and et al., "Improved utilization of noc channel bandwidth by switch replication for cost-effective multi-processor systems-on-chip," in *NOCS*, 2010, pp. 165–172.

[27] H. K. Lee and D. S. Ha, "Hope: An efficient parallel fault simulator for synchronous sequential circuits," in *DAC*, 1992, pp. 336–340.

[28] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Dynamic state traversal for sequential circuit test generation," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 5, no. 3, pp. 548–565, Jul. 2000. [Online]. Available: https://filebox.ece.vt.edu/∼mhsiao/strat_dnload/

[29] S. Ma, N. Enright Jerger, and Z. Wang, "Whole Packet Forwarding: Efficient Design of Fully Adaptive Routing Algorithms for Networks-on-Chip," in *HPCA*, Feb. 2012, pp. 467–478.

[30] G. Michelogiannakis, J. Balfour, and W. J. Dally, "Elastic buffer flow control for on-chip networks," in *HPCA*, 2009.