IEEE TRANSACTIONS ON
PARALLEL AND
DISTRIBUTED SYSTEMS

IEEE Xplore® Digital Library
http://ieeexplore.ieee.org/Xplore

MACS: A Highly Customizable Low-latency Communication Architecture

Rohit Kumar and Ann Gordon-Ross {*kumar, ann*}*@chrec.org*
NSF Center for High-Performance Reconfigurable Computing
University of Florida, Gainesville, FL 32611

Abstract—Networks-on-chips (NoCs) are an increasingly popular communication infrastructure in single chip VLSI design for enhancing parallelism and system scalability. Processing elements (PEs) connect to a communication topology via NoC switches, which are responsible for runtime establishment and management of inter-PE communication channels. Since NoC switch design directly affects overall system performance and exploited communication parallelism, much previous work focuses on efficient NoC switch design. In this paper we present MACS—a highly parametric NoC switch architecture that provides reduced data transfer latency, increased designer flexibility, and scalability as compared to previous architectures by combining and enhancing several NoC design strategies. MACS enhances inter-PE communication using a circuit switching technique with minimal adaptive routing and a simple and fair path resolution algorithm to maximize bandwidth utilization. We evaluate area and performance of an FPGA implementation of MACS, and, compared to previous work, MACS offers a 2x to 7x decrease in average channel setup latency, a 1.7x to 2x reduction in area requirements, similar average packet latency, up to a 6x increase in the network saturation point, and up to a 1.4x increase in bandwidth utilization. Additionally, we illustrate MACS's low average channel setup latency using 6 network traffic patterns and 8 parallel JPEG decompression core trace simulations.

Index Terms—Network-on-chip, minimal adaptive, distributed round-robin, FPGA

1 INTRODUCTION

SYSTEM-ON-CHIPS (SoCs) are composed of multiple processing elements (PEs) that are typically assembled in an application domain-specific architecture with an inter-PE communication network to enable the PEs to carry out application tasks. The PEs and the communication network exploit inherent inter-task parallelism, which typically increases as the number of PEs increases, for efficient system performance. However, the key to maximizing leveraged parallelism in modern and future SoCs is an efficient and scalable inter-PE network and communication methodology. To provide efficient and scalable SoC inter-PE communication Dally et al. [9] and Benini et al. [2] introduced a *network-on-chip* (NoC) design paradigm. Benini et al. [2] showed that NoCs provided enhanced performance, bandwidth, and scalability as compared to dedicated wires or buses [6] [19] [29] [11].

An NoC is composed of a set of switches and a *point-to-point* interconnect to form a network *topology* (e.g., mesh/torus, tree/fat-tree, and ring). Switches connect to neighboring switches and one or more PEs using *ports*. PEs coordinate task execution over these ports using control and data messages that are either broken into packets or streamed over a *communication channel*.

Traditional switches are categorized based on the switch's structural and transmission parameters. Structural design parameters include arbitration policy (e.g., round robin, priority, bus-based, etc.), buffering strategy (e.g., buffered, bufferless, etc.), and routing algorithm (e.g., dimension order, minimal adaptive, etc.). Transmission design parameters include channel allocation (e.g., physical, virtual, etc.) and switching mode (e.g. packet switching, circuit switching, etc.). An NoC's topology and the switch's structural and transmission design parameters determine the NoC's usability and application performance [7]. Since exploring the combination of all of these design parameters is impractical, we restrict our analysis and evaluation to a mesh topology, but also consider structural and transmission design parameters, and compare our circuit switched NoC with packet switching and mixed switching NoCs.

Packet switching(e.g., [26], DBAR [22], [23], [18], MANGO [4], HERMES [25], Xpipes [1], and SoCIN [36]) is the most commonly used NoC switching mode [3] because packet switching typically offers higher aggregate bandwidth and more efficiently utilizes resources by dynamically distributing simultaneous communication channels through different switches. However, packet switching disadvantages include: packet buffering and processing requirements, which increase the NoC's size and complexity; the channel latency's (time between the source PE's packet transmission and the destination PE's packet reception) dependence on the network load; a potentially low network saturation point (i.e., the point after which increasing the network load does not increase the throughput but increases channel latency exponentially); and the potential for blocked switches in heavily loaded networks [7].

Circuit switching (e.g., [20], [8], [7], PNoC [14], Æthereal [13], SoCBUS [34], and Octagon [16]) provides exclusive and static data communication channels be-

Digital Object Identifier no. 10.1109/TPDS.2015.2390631

1045-9219 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TPDS.2015.2390631

tween communicating PEs, thereby providing guaranteed throughput and channel latency bounds for individual packets. Additionally, since packets in a circuit switching network are pipelined along the channel (i.e., each router acts as a pipeline stage), circuit switching requires only a single register for packet buffering (as opposed to packet switching's FIFO buffer). Circuit switching is most efficient for network traffic with long packets, high transmission rates, and/or throughput and latency requirements. However, circuit switching disadvantages include channel bandwidth under utilization for lower data transmission rates and circuit setup latency, which depends on the channel's path's current network traffic load. Additionally, most circuit switching NoCs have a fixed architecture and thus provide limited flexibility for customizing the NoC architecture to a particular application's network traffic load, bandwidth, and throughput requirements (aside from complete architectural redesign).

To leverage the advantages of both circuit and packet switching, several recent NoCs combined circuit and packet switching into a mixed switching NoC [31], [21], [23], [24]. [31] and [24] show a 37% and 45% reduction in the average packet latency with a 12% and 10% increase in the area overhead as compared to a conventional packet switching 2D mesh-based NoC, respectively. [23] shows a 19% reduction in average packet latency as compared to a conventional packet switching NoC. Although [31], [21], [23], and [24] do not provide bandwidth utilization results, the high network saturation point in [31], [23], and [24] suggest high bandwidth utilization.

To increase bandwidth utilization and reduce area overhead, more than one PE can be connected to each NoC switch, which reduces the number of switches required and reduces the per-PE area overhead. However, increasing the number of PEs per switch increases the number of communication requests, which increases the number of communication paths required. Typical NoC switches establish a communication path using deterministic-XY routing, which may not take advantage of all available paths in a mesh topology. Deterministic-XY routing algorithms overuse the same routes, which leads to inefficient communication resource utilization and may fail to establish a communication path between PEs even if a valid communication path does exist (i.e., the valid path cannot be discovered using deterministic-XY routing). To increase the number of available communication paths and maximize the bandwidth utilization, an adaptive routing algorithm with low path setup latency and a fair path resolution mechanism can be implemented.

In this work, we addressed these limitations with MACS—a **M**inimal **A**daptive routing **C**ircuit **S**witching based switch for a two-dimensional mesh topology NoC. Since circuit switching NoCs are best suited for streaming data traffic and streaming data typically exist between producer-consumer PE pairs, MACS leverages *PE clustering*. MACS's PE clustering connects two PEs to each switch, which enables fast circuit establishment and data transfers between producer-consumer pairs that are placed in the same cluster because the data can be transferred directly between these PEs without traversing the network topology. Additionally, PE clustering reduces the total number of switches (which in turn reduces the area requirements), increases bandwidth utilization, and increases system design flexibility, enabling designers to strategically place producer-consumer PE pairs on the same switch. MACS provides efficient communication between arbitrary PE pairs using three techniques: establishing communication channels through the switches with the maximum number of available path choices, which maximizes *bandwidth utilization*, using minimal adaptive shortest path routing with fair path resolution; reducing communication channel setup latency with lightweight, distributed round-robin arbitration; and providing high communication operation frequency with an efficient circuit-switched routing decision state machine.

To increase design flexibility and system customization, we implemented MACS as a highly parametric VHDL model with numerous tunable architectural parameters. We developed FPGA implementations of MACS for varying architectural parameters and evaluated the impact of varying these architectural parameters on area and operating frequency. Additionally, we analyzed the average channel setup latencies using several synthetic network traffic patterns (uniform random, bit-complement, tornado, transpose, and nearest neighbor) for a 3x3 MACS-NoC (3x3 mesh topology NoC of MACS switches). To further analyze average channel setup latency, we also evaluate a trace-based simulation of eight JPEG decompression cores [32] operating in parallel on an 8x8 MACS-NoC. Finally, we compare an 8x8 MACS-NoC's average packet latency, network saturation point, and average bandwidth utilization to previous works.

The remainder of this paper is organized as follows. In Section 2, we discuss previous work in circuit switching NoC architectures. Section 3 presents the MACS switch architecture and switch operations. Section 4 discusses MACS's communication channel routing algorithm. In Section 5, we present the MACS FPGA implementation and simulation results and comparison with prior work. Finally, Section 6 concludes our work and discusses future work.

2 Related Work

Efficient switch architecture design motivated much early NoC research and in order to compare these switch architectures, [11] [37] [34] [33] provided comparisons based on different switching techniques and topologies. Wiklund et al. [33] evaluated different topologies and proved that the mesh topology was the most appropriate for on-chip networks.

Wiklund et al. [33] and Zheng et al. [37] both provided strong arguments for the advantages of circuit

U.S. Government work not protected by U.S. copyright.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TPDS.2015.2390631

switched NoCs over packet-switched NoCs. Wiklund et al. [34] proposed a mesh topology NoC (SoCBUS) using distributed arbitration and a new switching technique known as Packet Connected Circuits (PCCs). In a PCC, packets traverse the network, locking circuits by leveraging round-robin arbitration to select one of the two paths during the packet's progression. Due to this round-robin path selection methodology, SoCBUS explored only one shortest path (even though many existed). In addition, due to selection of only one shortest path, SoCBUS provided a fewer number of simultaneous communication channels.

To increase the number of simultaneous channels, Zheng et al. [37] proposed a Time Division Multiplexed (TDM) scheme for communication channels on a 2D mesh NoC. Due to time multiplexing of channels over the same links in TDM, the NoC offered a higher number of simultaneous channels than SoCBUS. However, one potential drawback of TDM was out-of-order data arrival, but the authors provided a mechanism to ensure in-order data arrival. Moreover, due to the use of centralized routing in [Zheng2004], [Zheng2004] could suffer from communication bottlenecks.

In order to alleviate communication bottlenecks, Ahmadinia et al. [5] proposed RMBoC (a circuit-switched NoC) based on the Reconfigurable Multiple Bus (RMB) [12]. RMBoC was implemented as both a 1D-array and a 2D-mesh topology. In the 2D RMBoC, each switch contained per-row and per-column network controllers to deterministically route channel establishment requests. RMBoC had several drawbacks including deterministic routing that did not maximize bandwidth utilization, large area requirements, and low operating frequencies due to a complex routing algorithm and the network controllers.

In order to improve architectural customization, Hilton et al. [14] presented the Programmable Network-on-Chip (PNoC), a highly flexible circuit switched NoC for field programmable gate array (FPGA) systems. PNoC's tunable parameters included the number of switch ports and data and address bus widths. In addition, PNoC connected multiple PEs to each switch but allowed only one PE to communicate at a time. Whereas this technique appeared to be inflexible, the technique was suitable for processor-farm systems. Furthermore, PNoC could not efficiently utilize communication resource unless the operating system updated the routing table during runtime.

To improve the flexibility of PNoC while keeping the architectural flexibility and to improve real-time data processing, Carara et al. [7] proposed a 4x4 NoC (which we refer to as CararaNoC henceforth) that employed mixed switching techniques with fixed sized packets. In order to provide high throughput to support real-time data processing, CararaNoC implemented two parallel interfaces (i.e., lanes) in all switch ports and routing resource multiplexing with the help of multi-phased data transfer sessions. Due to implementation of multi-phased data transfer sessions, session buffers, and cell-based data transfer, CararaNoC provided better resource utilization and run-time adaptation of NoC channel bandwidth with application bandwidth requirements. Additionally, CararaNoC provided two lanes in each direction to increase the communication bandwidth and reduce the switch's area requirements. However, high latency values in simulation (only a few PEs out of a total of 16 PEs were active in simulation) suggested that CararaNoC suffered from path contention due to deterministic-XY routing. Furthermore, CararaNoC was implemented as fixed 4x4 NoC architecture with no customizable parameters.

In order to increase architectural customizability and to reduce area overhead of earlier architectures we introduced SCORES [15], a 1D circuit switched architecture with stream-based data transfer. SCORES used a 1D-array topology and was implemented as a highly parametric VHDL model with numerous tunable architectural parameters. Tunable architectural parameters included number of port lanes and data width. Compared to previous work, SCORES had a low area architecture that reduced communication establishment bottlenecks, increased architectural specialization, and provided high communication operating frequencies. However, due to the one dimensionality of SCORES, SCORES's main drawback was a lower number of available paths between PEs, and thus lower bandwidth, as compared to previous works.

To increase NoC bandwidth, Lin et al. [20] proposed express communication paths in a mesh-based circuit switched NoC (referred to as LinNoC henceforth). LinNoC implemented additional connections between switches three hops apart, which served as express, low latency communication paths. LinNoC showed up to a 19.43% reduction in packet latency, but suffered from a low network saturation point.

To increase the network saturation point and thus maximize bandwidth utilization, Lusala et al. [21] proposed a switch architecture (which we refer to as LusalaNoC henceforth) that combined spatial division multiplexing(SDM) and time division multiplexing(TDM). SDM and TDM allow the router to share channels among multiple connections thereby increasing the probability of establishing paths through the network. However, due to LusalaNoC's deterministic-XY routing, LusalaNoC did not explore all paths between two switches. Additionally, due to SDM and TDM, LusalaNoC suffered from high packet latency and high area overhead.

To reduce packet latency and improve bandwidth, Qian et al. [26] (referred to as BiNoC* henceforth) combined a bi-directional channel NoC [17], BiNoC, with express channels, regional hub routers, and regional traffic congestion information. Regional hub routers created short-cut paths by bypassing fixed number of intermediate switches and thus reduced packet latency. Additionally, to reduce network congestion's effect on latency,

U.S. Government work not protected by U.S. copyright.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TPDS.2015.2390631

each switch contained traffic congestion information of fixed number of hops. Due to bi-directional channels, express channels, traffic congestion information, and regional hub routers, BiNoC* achieved up to 80% latency improvement with 19% area overhead as compared to a typical NoC.

To reduce area overhead, recent work by Teimouri et al. [31] (referred to as TeimouriNoC henceforth) divided a [10]-based n-bit network into two n/2-bit sub-networks. One of the n/2-bit sub-networks directed the packets according to the traditional packet switching scheme and the other sub-network was used to build partial communication paths to shorten the packet's path to the packet's destination. [31] showed a 37% reduction in latency, but a 12% increase in area overhead as compared to a conventional packet switching NoC [10]. Since [31] used deterministic-XY routing, [31] did not utilize all available paths and thus did not maximize bandwidth utilization.

In this work, MACS addresses SCORE's, LinNoC's, LusalaNoC's, and TeimouriNoC's bandwidth and bandwidth utilization limitations and introduces novel enhancements, such as exploration of additional paths and fair path resolution. MACS significantly enhances SCORES into a switch appropriate for a 2D mesh topology NoC, which removes communication channel bottlenecks using a minimal adaptive routing algorithm to ensure alternate path exploration in orthogonal directions. A lightweight cost metric (local to each switch) evaluates and selects the best available path to maximize number of possible paths through a switch and distributed round-robin arbitration reduces channel setup latency. We implemented MACS as a highly parametric VHDL model to provide numerous architectural customizations (e.g., number of lanes in each direction, number of lanes on local PE ports, data width and network dimension). In addition to a parametric design, the MACS's RTL model is a highly modular design (e.g., the routing algorithm is implemented as an entity independent of other components such as arbiter or path resolution module) that provides low modification-verification cycle time and high communication operating frequencies. Finally, we present extensive simulation studies of 3x3-MACS (NoC of MACS switches) for several synthetic network traffic patterns and trace-based simulation studies of 8x8-MACS for JPEG decoders.

3 MACS Architecture

Figure 1 (left) depicts 3x3-MACS's high-level architectural layout. X and Y coordinates identify individual switch addresses based on horizontal and vertical positions, respectively. Each switch's two PEs are addressed relative to their connected switch. MACS has four total *switch ports* with one port connected to each neighboring switch (left, right, up, and down) and two *local ports* connected to the two PEs (Figure 1 (right)). A *port identification number* (*PID*) uniquely identifies each port. Each port contains multiple input and output communication *lanes* to support multiple simultaneous communication channels between different PE pairs (denoted by the K tunable architectural parameter in Figure 1). Furthermore, each lane contains input and output data signals (denoted by the W tunable architectural parameter in Figure 1) to provide data transfer bandwidth on a communication channel. MACS provides guaranteed throughput using a circuit-switched communication methodology with distributed arbitration. In this section, we provide an overview of switch operation followed by switch architectural details.

3.1 Switch Operation

Switch operations include communication channel establishment for inter-PE data transfers (transactions), waiting for transaction completion, and releasing *communication channel resources* (e.g., logic elements, registers, etc.). Channel establishment connects an input lane to an output lane and is the process of allocating channel resources for routing incoming channel establishment requests and data on this input-output lane connection, thus establishing a dedicated communication channel. After channel resource allocation, the switch waits until transaction completion before releasing these resources.

Each port contains *control logic* for channel establishment. The control logic consists of two types of *control logic blocks*: an External Signal Forwarder (*ExSIF*) and an Internal Signal Forwarder (*InSIF*) (collectively referred to as *signal forwarders*). Signal forwarders are responsible for controlling all communication operations such as request servicing, channel establishment negotiations, and channel release.

3.2 Switch Architecture

Figure 1 (right) depicts MACS's detailed switch architecture including all switch/local ports and signals associated with each port's input and output lanes. Switch ports and local ports are similar in that they consist of a set of lanes. A *lane identification number (LID)* uniquely identifies each port's lane. Lanes can be further classified as input and output lanes that carry request/data signals into and out of a switch, respectively. Each input lane consists of several control signals (*req_in*, *gnt_out*, *dny_out*, and *ful_out*) and W input data signals (*data_in*). Similarly, each output lane consists of several control signals (*req_out*, *gnt_in*, *dny_in*, and *ful_in*) and W output data signals (*data_out*). Control signals negotiate channel establishment while data signals provide inter-PE communication bandwidth (increasing W increases MACS's communication bandwidth).

MACS's ports are highly parametric, providing fine-grained per-direction communication bandwidth specialization. Each switch/local port can be specialized with different numbers of input or output lanes. K denotes a port's lane count, and to distinguish between distinct switch/local ports, K is appended with the *PID*.

U.S. Government work not protected by U.S. copyright.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TPDS.2015.2390631

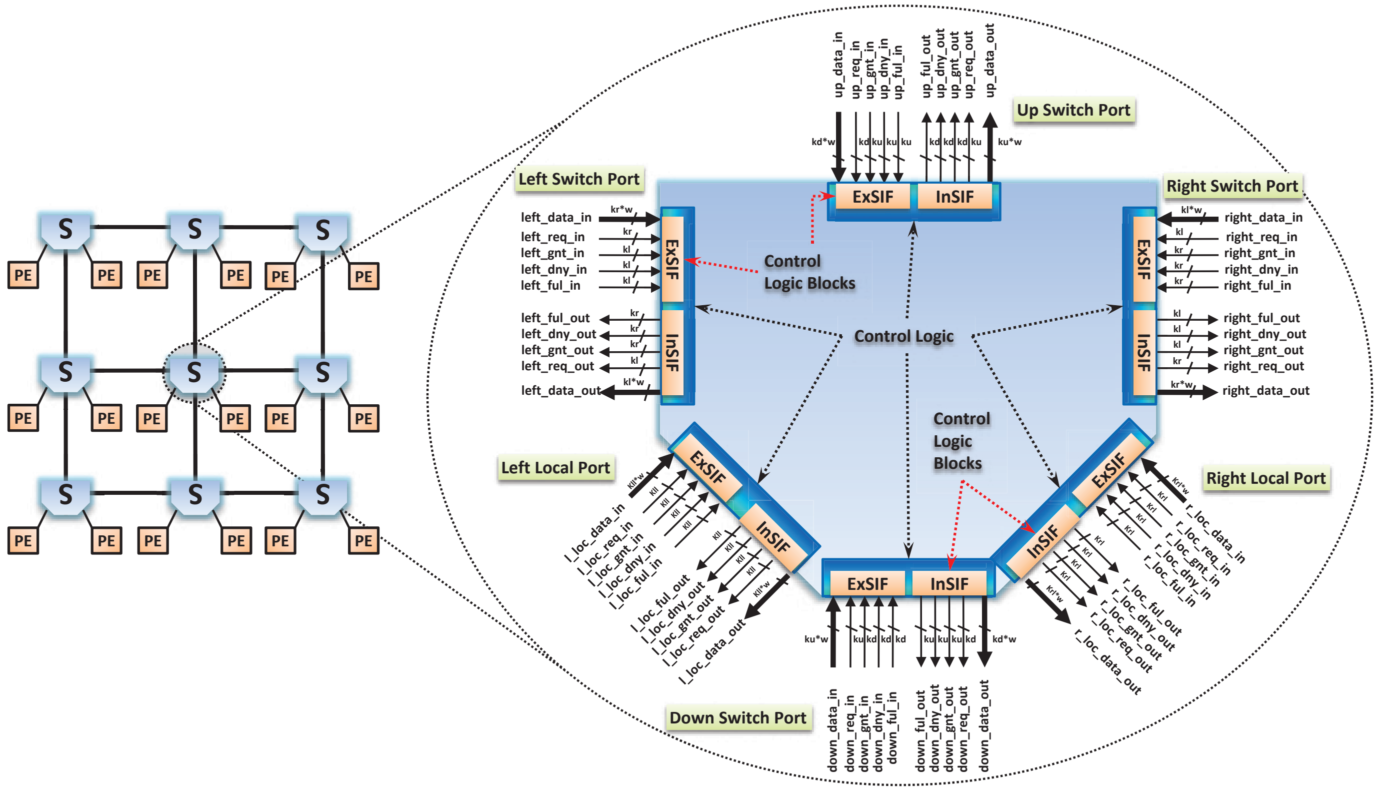


Figure 1: 3x3-MACS's architectural layout (left) and detailed MACS switch's architecture (right). The switch architecture shows all switch ports, local ports, and control logic blocks (ExSIF and InSIF) with associated port input/output signals. Each port shows all port signals. Architectural parameters include number of lanes per port (K) and number of data signals per lane (W).

Thus, a switch has six total K parameters for communication bandwidth specialization (the number of simultaneous *different* per-port inter-PE communication channels, and thus the bandwidth, monotonically increases with K).

In order to control communication channels and ensure correct communication, switches record channel routing information in *status registers*. Channel routing information specifies lane availability and input-output port lane connections. Section 3.2.1 discusses signal forwarders and status registers. Signal forwarders populate status registers during channel establishment to subsequently maintain fixed communication channels. Section 3.2.2 discusses the distributed round-robin arbitration implemented in the *InSIF*s to arbitrate incoming requests to output lanes.

3.2.1 Signal Forwarders and Status Registers

Figure 2 depicts the internal connections between two arbitrary switch port lanes (connections with remaining switch port lanes are labeled but not shown) and the status register placement. The signal forwarders, *ExSIF* and *InSIF*, establish internal connections between input and output lanes, respectively. For readability, Figure 2 has been subsetted to show components and signals for a sample input-output lane connection. In the actual architecture, every lane contains all shaded components.

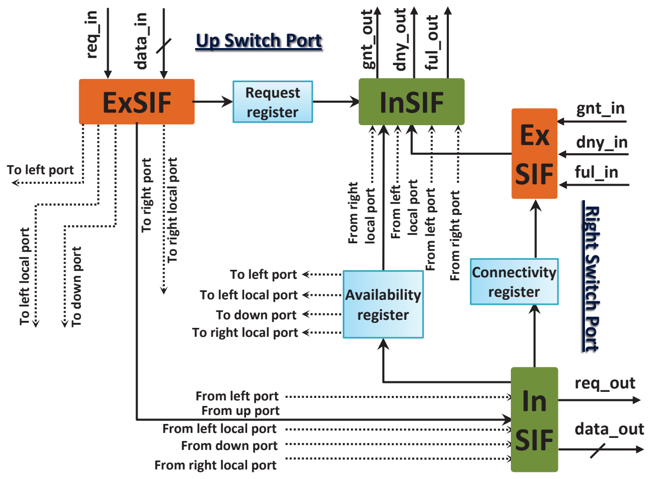


Figure 2: A subset of internal connections between the signal forwarders (*ExSIF* and *InSIF*) and status table placement of two arbitrary lanes (up and right) (additional connections with remaining switch ports are labeled). The figure has been subsetted for readability. In the actual architecture, every lane contains *all* of the shaded components.

ExSIF and *InSIF* use the *request register* to forward, maintain, and release incoming requests from the neighboring switches and the request's grant/deny to neighboring switches, respectively. *InSIF* polls internal requests forwarded by *ExSIF* in a distributed round robin

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TPDS.2015.2390631

fashion (Section 3.2.2 and assigns an output to a neighboring switch based on the communication channel routing algorithm (Section 4) and the *availability register*. The *availability register* contains information about the number of output lanes not serving an output to a neighboring switch. *InSIF* and *ExSIF* use the *connectivity register* to forward, maintain, and release internal requests forwarded by *ExSIF* and the request's grant/deny coming from the neighboring switch, respectively.

MACS's routing algorithm evaluates multiple routing paths and allows each port to store the port's unavailable communication resources (i.e., number of the port's output lanes that are reserved for established channels) in the *availability register*. As the number of unavailable communication resources decreases, the likelihood of a new channel establishment assigned to this port increases and thereby maximizes bandwidth utilization.

3.2.2 Distributed Round-Robin Arbitration

As shown in Figure 2, an *InSIF* receives incoming requests from all *ExSIFs*, except *InSIF*'s own port's *ExSIF*. Selecting and maintaining a set of incoming requests for a set of outputs requires a complex round-robin arbiter, which could increase area and channel setup latency of the switch and reduce the switch's maximum attainable operating frequency. A typical round-robin arbiter (referred to as a typical arbiter henceforth) identifies an incoming request from a set of incoming requests in a round-robin fashion, identifies an available output lane(s) in a round-robin fashion, and connects the incoming request to the output lane. Identification and connection of multiple incoming requests to multiple output requests simultaneously greatly increases the typical arbiter's complexity. In order to reduce the arbiter's complexity, the *InSIF* uses distributed round-robin arbitration with cyclic distribution of incoming requests over output lanes. This distribution causes disjoint sets of incoming requests to map to different output lanes. Additionally, if the K architectural parameter for all ports are equal, cyclic distribution ensures one-to-one correspondence between the *LID* of incoming requests' input lane and the *LID* of the output lane (e.g., all requests arriving on *LID=n* are always mapped to the output lane with *LID=n*). One-to-one correspondence of *LIDs* provides improved controllability of the NoC due to constraint on routing traceability.

In order to realize arbitration of cyclically distributed incoming requests, the *InSIF* implements a per-output lane distributed round robin arbiter (referred to as distributed arbiter henceforth) to choose a single request from multiple incoming requests for connection to an output lane. Figure 3a shows the distributed arbiter for an output lane. Each distributed arbiter consists of two components: a multiplexer and a counter. To select an incoming request, the counter's output is attached to the MUX's *select* lines and to control the counter, the MUX's output is attached to the counter's active low *enable* signal. To exemplify the distributed round robin arbitration process, we consider the initial state where the MUX's inputs/output and counter's output are de-asserted. As the MUX's output is de-asserted in the initial state, the counter is enabled and starts selecting one incoming request per clock cycle (shown as *clk* in Figure 3a). As soon as the counter selects an asserted incoming request, the MUX's output is also asserted which, in turn, disables the counter. Disabling the counter holds the current counter value and MUX's *select* lines persistent thereby keeping a persistent connection between the incoming request and the output lane. De-assertion of the currently selected incoming request de-asserts the MUX's output and enables the counter for selection of subsequent incoming requests.

To decrease the channel setup latency and to reduce the area and frequency overhead of the distributed arbiters, the distributed arbiters' MUX's size and the counter threshold depend on the ratio of the total number of incoming requests and the total number of output lanes (e.g., for a distributed arbiter in the right switch port's *InSIF*, the counter threshold is $ceil((Kll + Krl + Kd + Kr + Ku)/Kr)))$. Limiting the counter size limits the service latency (the number of clock cycles before selection of an asserted incoming request signal by the MUX) experienced by an incoming request. As service latency accrues over the channel path, which in turn increases the channel setup latency, the distributed arbiters ensure lower channel setup latency as compared to typical arbiters. Finally, due to a small (only two components) and simple design, distributed arbiters do not adversely affect the switch's area requirements or maximum attainable frequency.

3.2.3 Distributed Versus Typical Round-Robin

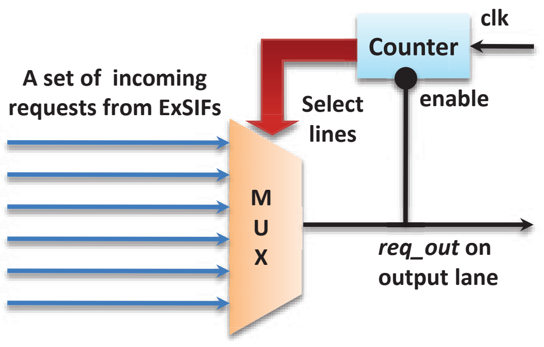
In order to estimate a distributed arbiter's advantages over a typical arbiter, we compared and evaluated these arbiters using FPGA implementation and simulation.

Fig. 3b compares the distributed and typical arbiters for a varying number of input and output lanes based on the efficiency (i.e., percentage of incoming requests that are granted an output) difference. The efficiency difference was calculated by subtracting the distributed arbiter's efficiency from the typical arbiter's efficiency. Since the minimum number of input or output lanes on a MACS port is 1, and 1 output lane receives requests from at least 5 input lanes, the graphs vary the total number of arbiter inputs from 5 to 25 and the total number of arbiter outputs from 1 to 5.

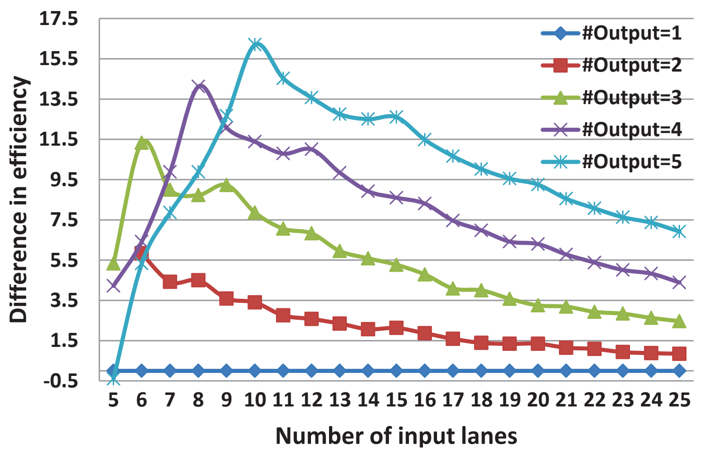
To evaluate the arbiters' efficiency, we created a C simulation for both arbitrations. The data transfer and idle (no request or data transfer) periods are simulated by generating/maintaining and destroying the input requests at uniformly distributed random intervals for each input. Figure 3b shows that the distributed arbiter's efficiency increases (i.e., arbiters' efficiency difference decreases) as the number of input lanes increases or number of output lanes decreases. Since the distributed arbiter's efficiency is calculated from smaller typical

U.S. Government work not protected by U.S. copyright.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TPDS.2015.2390631



(a)



(b)

Figure 3: (a) Distributed round robin arbiter architecture for one output lane. Selection of an asserted incoming request asserts the output lane signal, disables the counter, and retains a persistent input-output connection. De-assertion of a selected input lane will de-assert the output lane and enable the counter for further selection. (b) The percentage efficiency difference, based on results gathered from C-based simulation, between the typical and distributed round robin arbiter. The percentage efficiency difference was calculated for a varying number of input (x-axis) and output lanes (#Output) in the typical and distributed round robin arbiters

arbiters' efficiencies, the distributed arbiter's efficiency change is not monotonic (particularly when the *number_of_inputs* are nearly equal to an integer multiple of *number_of_outputs*) and causes local maximas in the efficiency difference between the typical and distributed arbiters. Although the maximum efficiency difference is 16% for five output lanes, a typical NoC switch implements at most two output lanes and for a MACS switch implementation, the efficiency difference between the typical and distributed arbiters is negligible. To compare area and frequency overhead of the distributed and typical arbiters, we implemented these arbiters in VHDL and compare the arbiters' FPGA implementations' area and frequency results. The results reveal that a distributed arbiter requires 1x-12.6x less area and provides a 1x-5.4x increase in maximum operating frequency as compared to a typical arbiter. Since the distributed arbiter is implemented with a smaller one-output typical arbiter, the distributed arbiter and the typical arbiter show the same area, frequency, and efficiency when *number_of_output* is equal to 1.

4 Communication Channel Routing

Establishing an arbitrary inter-PE communication channel on a 2D mesh is a straightforward process, however, selecting the best communication channel given all potential routes is challenging. For example, minimal adaptive routing chooses the shortest path between two points in a 2D mesh, thus defining the best route as simply the shortest path. However, between two points in a 2D mesh, there are $(\Delta X + \Delta Y)!/((\Delta X)! * (\Delta Y)!)$ equal length shortest path routes where ΔX and ΔY are the differences between the X and Y coordinates of the two points, respectively. Routing algorithms that arbitrarily and/or deterministically choose a particular path can lead to communication bottlenecks and communication resource starvation (all port's lanes are occupied). Our implementation of minimal adaptive routing and the distributed arbiter to resolve the best routing path are unique and minimalistic, and thus afford low area overhead and small latency. We, define the best routing path as both a shortest path (with available communication lanes) and one that best distributes communication channels to avoid communication bottlenecks and communication resource starvation. MACS achieves these routing goals using a minimal adaptive routing state machine, as discussed in Section 4.1.

4.1 Routing State Diagram

Our communication channel routing algorithm is based on minimal adaptive routing to establish, maintain, use (transfer data), and release inter-PE communication channels. These processes correspond to four channel phases: the request service phase, the grant/deny phase, the data transfer phase, and the resource release phase. Each switch can maintain multiple channels, each of which may be in any one of these phases (regardless of the other channel's phases).

In the request service phase, a switch identifies destinations for all incoming requests, forwards the incoming requests to neighboring switches towards the destination, and waits for grants/denies. In the grant/deny phase, a switch port receives all grants/denies, resolves between contending grants, if any, based on the number of busy channels and forwards the grants/denies. A communication channel is established at the end of the grant/deny phase if the grant/deny phase received at least one grant from the neighboring switch(es). In the

U.S. Government work not protected by U.S. copyright.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TPDS.2015.2390631

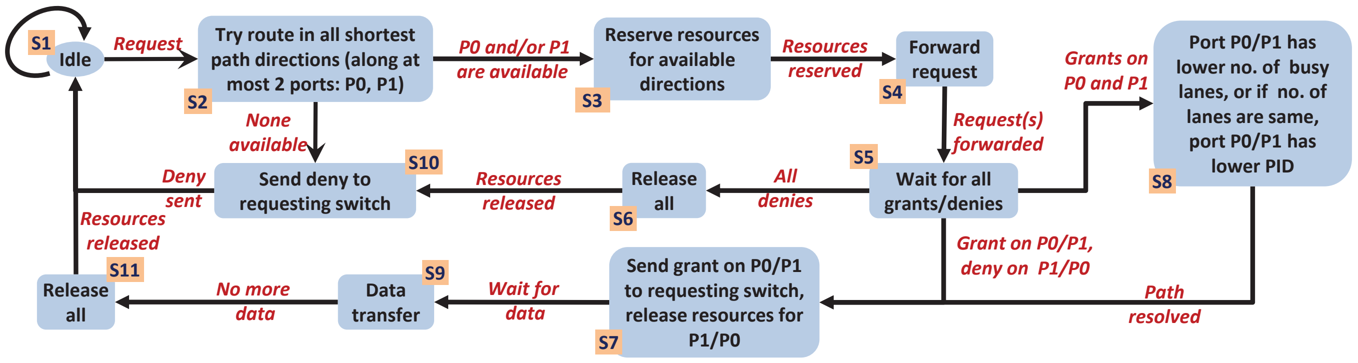

S1 Idle
Request
S2 Try route in all shortest path directions (along at most 2 ports: P0, P1)
P0 and/or P1 are available
S3 Reserve resources for available directions
Resources reserved
S4 Forward request
Request(s) forwarded
S5 Wait for all grants/denies
Grants on P0 and P1
S8 Port P0/P1 has lower no. of busy lanes, or if no. of lanes are same, port P0/P1 has lower PID
Path resolved
All denies
S6 Release all
Resources released
S10 Send deny to requesting switch
None available
Deny sent
Resources released
S11 Release all
Grant on P0/P1, deny on P1/P0
S7 Send grant on P0/P1 to requesting switch, release resources for P1/P0
Wait for data
S9 Data transfer
No more data

Figure 4: State diagram for a switch's channel phase actions and transitions.

data transfer phase, the switch maintains the channel based on several status registers, populated by the request service phase and the grant/deny phase, and transfers data through the communication channel. The end of data transfer triggers the resource release phase. In the resource release phase, the communication channel is destroyed and all resource, such as the status registers related to the communication channel, are reset. Figure 4 depicts a state diagram of channel phase actions and transitions.

The switch begins operation in the idle state (*S1*). If the switch receives a channel establishment request and associated channel establishment information on the *req_in* and *data_in* signals, the switch transitions to *S2* and begins the request service phase. In *S2*, the switch compares its address with the destination switch address and determines the potential ports (*'P0'* and/or *'P1'* in *S2*) in which to forward the incoming request (according to the minimal adaptive algorithm). If the current and destination switch addresses do not match, the requested PE is not connected to the current switch and the current switch must forward the request to neighboring switches. If the current and destination switch addresses do match, the request is forwarded to the appropriate local PE port. If there are available output port lanes, the switch transitions to *S3* to establish input-output lane connections and complete the request service phase. In *S3*, the switch adds the appropriate entries to the status tables (Sections 3.2.1). After adding these entries, the switch transitions to *S4* and forwards the request to the available lane on port(s) *'P0'* and/or *'P1'*. On the other hand, if there are no available output port lanes, the routing fails and the switch transitions to *S10*, sends a deny response to the requesting switch, and transitions back to the idle state (*S1*).

After request service phase completion (i.e., after request forwarding in *S4*), the switch enters the grant/deny phase (*S5*) and waits for the grant and/or deny signal responses (*gnt_in* or *dny_in*, respectively) on the output lane of port(s) *'P0'* and/or *'P1'*. If only one port (direction*'P0'*) was selected in the request service phase and a grant is received, the switch transitions to *S7* and forwards the *gnt_in*, *dny_in*, and *ful_in* signals to the corresponding *gnt_out*, *dny_out*, and *ful_out* signals of the requesting port's input lane (the port's input lane in which the request generated from). If two ports (both directions) were selected in the request service phase, there are three possible state transition situations. In the first situation, the switch receives denies from both ports and transitions to *S6* to release all channel resources associated with both ports. The switch transitions to *S10*, sends a deny to the requesting port's output lane and transitions back to the idle state (*S1*). In the second situation, the switch receives one grant (e.g., associated with *'P0'*) and one deny (e.g., associated with *'P1'*). The switch transitions to *S7*, forwards *gnt_in*, *dny_in*,and *ful_in* from *'P0'* to the requesting port's input lane, and releases the resources associated with *'P1'* (note that the case is similar when *'P0'* denies and *'P1'* accepts.) In the third scenario, the switch receives grants from both ports and transitions to *S8* for grant resolution. Grant resolution selects the best channel to establish by evaluating the port responses and associated route costs to determine which response to forward to the requesting port's input lane. Route cost is defined as the number of lanes already assigned to existing communication channels. If both ports route costs are different, the switch selects the lowest route cost port as the best port. If both ports route costs are equal, the switch selects the port with the lowest *PID* as the best port. Due to this lowest route cost-based port resolution and best port selection, MACS distributes new communication channels towards less congested portions of the network, thereby achieving communication load balancing. After the best port is selected (arbitrarily denoted as *'P0'* in *S8* and *S7*), the switch transitions to *S7*, forwards *gnt_in*, *dny_in*, and *ful_in* of port *'P0'* to the requesting port's input lane, and releases the resources associated with port *'P1'*. This algorithm is deadlock free because in all situations, the switch forwards/sends either a grant or a deny to the requesting input port, which prohibits infinite channel locking. The total number of cycles required for releasing all of the resources is linear with $(\Delta X + \Delta Y)$.

The request service phase propagates successively down all shortest paths from the source switch to the destination switch simultaneously. The grant/deny

U.S. Government work not protected by U.S. copyright.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TPDS.2015.2390631

phase propagates successively backward on all of these paths. Even though multiple request service phase paths may reach the destination switch, only one path will propagate the grant signal all the way back to the source switch, allocating channel resources on this backward propagation.

At each switch, if sufficient channel resources exist and that switch is allocated to the routing path (i.e., lies on the best routing path), grant/deny phase completion (*S7*) and channel establishment occur simultaneously and pipelined data transfers (*S9*) can begin along the channel. Data transfer pauses/resumes when the '*full*' signal (associated with the destination PE's temporary inability to accept more data) is asserted/de-asserted by the destination switch (denoted by the *S9* to *S11* and *S11* to *S9* transitions, respectively). Since MACS uses a circuit switching methodology, in-order data arrival is guaranteed. The channel remains established until the switch enters the resource release phase (i.e., there is no more data to transfer) to free all associated channel resources. The switch transitions to *S12* and releases channel resources by removing the corresponding status table entries. The switch enters the resource release phase (*S12* and *S6*) under two situations. The first situation occurs when a PE requests a data transaction termination (*S12*). In the second situation, the resource release phase occurs simultaneously with the grant/deny service phase when a communication channel establishment request is denied (*S6*) (there are insufficient channel resources or the switch does not lie on the best routing path). Communication channel establishment denials can be either internal (a switch generates an internal deny due to grant/deny resolution) or external (a switch receives a denial from a neighbouring switch).

5 Experiments and Results

Selecting the best MACS configuration (combination of MACS's architectural parameter values) for an application is crucial to achieve system-specific area and performance goals. To evaluate the area utilization and maximum operating frequency, we implemented MACS as a highly parametric VHDL model. In order to facilitate system designers in selecting the most appropriate MACS configuration to meet application requirements, we evaluated MACS by varying the number of lanes per port (*K* architectural parameters) and the lane's data width (*W* architectural parameter). Given the prohibitively large design exploration space for all possible configuration combinations, we fixed the number of switch port lanes with respect to each other (*Kl* = *Kr* = *Kd* = *Ku*) and number of local port lanes with respect to each other (*Kll* = *Krl*) and evaluated the area utilization and maximum operating frequency for each combination of *Kl*, *Kr*, *Kd*, *Ku*, and *Kll* and *Krl*.

In addition to area utilization and maximum operating frequency evaluation, an accurate estimate of the channel setup latencies are required to evaluate how the MACS configuration affects application performance. To evaluate and compare the channel setup latencies, we simulated several standard network traffic patterns on the MACS-NoC, a 3x3 mesh topology NoC of MACS switches, using the Xilinx 12.2 ISE simulator. Additionally, to validate accuracy of estimates given by these traffic pattern for larger NoCs, we performed a trace-based simulation of eight JPEG [32] decompression cores on an 8x8 MACS-NoC.

To give further insight into MACS-NoC's performance, we presented a comparative study of average packet latency and average bandwidth utilization for an 8x8 MACS-NoC with respect to recent prior work on packet switching and mixed switching NoCs. We leveraged Riviera-Pro for Linux [27] to perform the comparative study's simulation. To provide a fair comparison, we performed simulations on the same two synthetic traffic patterns, uniform random and transpose, used in recent prior work. Results in Section 5.2.2 show that MACS-NoC provides similar average packet latency, but has a much higher network saturation point, and, either equal or better bandwidth utilization due to MACS-NoC's PE clustering and minimal adaptive routing as compared to prior work.

5.1 Area, Frequency, and Power Analysis

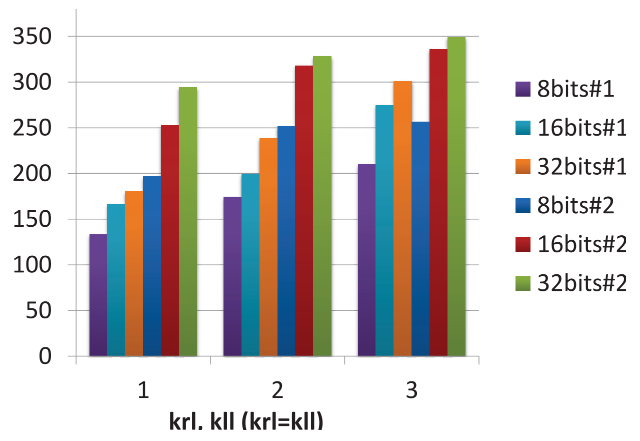
Figure 5a depicts MACS's switch area utilization in slices per PE (total switch area is twice these values since each switch has two PEs) for the Xilinx Virtex-6 (V6) XC6VLX760 FPGA for data widths *W*=8, 16, and 32 bits. The x-axis in each graph varies the *Krl* and *Kll* architectural parameters from 1 to 3 lanes per local port. Figure 5a shows area utilization results for both *Kr*=*Kl*=*Ku*=*Kd*=1 and *Kr*=*Kl*=*Ku*=*Kd*=2. For an example, the area usage for *Kl*=*Kr*=*Kd*=*Ku*=*Kll*=*Krl*=1 and *W*=32 bits is 180 slices per PE, which equates to only 0.15% of the available slices on the V6. In general, results reveal a linear increase in the area utilization for each doubling of the data width.

Figure 5b depicts the maximum achievable clock frequency for the same parameter values as Figure 5a. The results show that MACS can achieve high operating frequencies ranging from 157 MHz to 255 MHz which provide bandwidths ranging from 1.2 Gbps to 5.9 Gbps. Figure 5b reveals an abrupt increase in clock frequency when increasing *W* from 8 bits to 16/32 bits, which we attribute to clock frequency estimation difficulties resulting from too many MACS I/O ports for the V6 test device. Therefore, we discard clock frequencies with *W*=16,32 from MACS's bandwidth calculations.

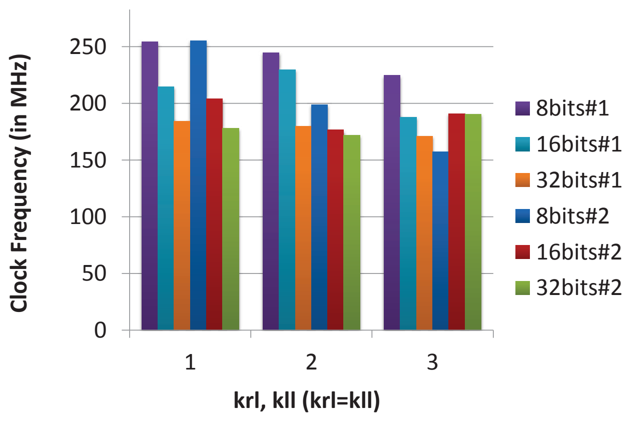
We compared MACS to the recent works in circuit switching NoCs that provided area and clock frequency values for FPGA implementation, CararaNoC [7] and LusalaNoC [21], in terms of area overhead in FPGA slices, LUTs, and flip-flop requirements. To make the comparison as fair as possible, we use the same Xilinx ISE version 12.2 with default synthesis and implementa-

U.S. Government work not protected by U.S. copyright.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TPDS.2015.2390631



(a)



(b)

Figure 5: (a) Area utilizations in number of slices per PE (b) Maximum achievable clock frequencies. Both, (a)and (b), show results for data widths W = 8, 16, and 32 bits with a varying number of lanes per switch and local port. The x-axis in each graph varies the *Krl* and *Kll* parameters from 1 to 3 lanes per local port. Area utilization and maximum achievable frequency is also obtained for varying the *Kr, Kl, Ku* and *Kd* parameters are from 1 to 2 lanes per switch port (shown as *W*bits#1 and *W*bits#2).

Table 1: Comparison of per-PE switch area overhead.

NoC:Device	Slices	LUTs	FFs	Frequency	Usage
CararaNoC(VC) [7]: Virtex2	861	1722	455	Not published	6.20%
CararaNoC(RC) [7]: Virtex2	758	1515	398	Not published	5.50%
MACS(per-PE): Virtex2	433	785	274	106.7 MHz	3.10%
LusalaNoC [21]: Stratix3	Absent	1927	1918	179.0 MHz	1.4%
MACS(per-PE): Virtex5	396	780	630	148.1 MHz	0.84%

tion parameters, and chose similar architectural parameters as reported for the CararaNoC—1 lane per port, a data-width of 8 bits, and the same device (Virtex-2, XC2VP30). CararaNoC provides two variations in NoC switches: switches with virtual channels and switches with repeated channels. The architectural parameters for LusalaNoC are 1 lane per port, a data-width of 32 bits (equivalent to 3 sub-channels in LusalaNoC), and a Xilinx Virtex-5 V5LX330 device, which is comparable to the Altera Startix-III EP3SL340F device used for LusalaNoC [30]. Since MACS connects two PEs per switch, a MACS-NoC requires half as many switches as CararaNoC or LusalaNoC for the same number of PEs and thus we compared the both NoCs' area requirements with MACS's per-PE area requirement. Table 1 compares MACS with CararaNoCs (virtual and repeated channel variations are suffixed with VC and RC, respectively) and LusalaNoCs reported area results. Compared to both variations of CararaNoC and LusalaNoC, MACS provides a 2x and 1.7x reduction in area overhead, respectively.

To evaluate MACS's per-component power consumption, we leveraged Xilinx Xpower [35] to perform MACS's switch's per-component percentage power consumption analysis for *Kl*=*Kr*=*Kd*=*Ku*=*Kll*=*Krl*=1 and *W*=32 bits. The results show that the *ExSIF*s consume 52.54% of the total power, which is due to the large de-multiplexers that route incoming requests and data signals to all ports identified by minimal adaptive routing algorithm. The *InSIF*s consume only 5.75% of the total power due to our distributed round robin arbiter and simple path contention resolution algorithm. The Toplevel includes the interconnections and steering logic that connect the *ExSIF*s and the *InSIF*s and consumes 41.71% of the total power due to routing wires and logic.

5.2 MACS Simulation Results

5.2.1 Channel Setup Latency

To evaluate channel setup latency estimates, we simulated several traffic patterns on MACS-NoC with Riviera-Pro on a Linux platform [27]. MACS-NoC contains 9 MACS switches (18 PEs) arranged in a 3x3 mesh topology with *Kl*=*Kr*=*Kd*=*Ku*=*Kll*=*Krl*=1 and *W*=32 bits. To transfer data from/to PEs to/from MACS-NoC, we implemented FIFO-based PE interfaces with a simple request-grant (*RG*) protocol. In order to handle channel setup request denials, PE interfaces include a counter-based request retrial mechanism (counter period was proportional to MACS-NoC's diameter). Each PE initiates a data transmission to the destination-PE (dictated by the network traffic pattern under consideration) on Poisson distributed random intervals (the Poisson distribution mean was proportional to MACS-NoC's diameter).

We simulated five standard network traffic patterns: uniform-random, nearest-neighbor, bit-complement, transpose, and tornado [11]. With the exception of tornado, and nearest-neighbor, all other traffic patterns contain multi-turn routes. Since all of these five traffic patterns are well suited for throughput calculation in

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TPDS.2015.2390631

Table 2: Average channel setup latencies for MACS-NoC.

Traffic pattern	Average number of grant cycles	Average number of deny cycles
Uniform-random	18.8	3.7
Nearest neighbor	7	0
Bit-complement	22	2.9
Transpose	17.8	3
Tornado	16	4

packet-switched NoCs, we did not use these traffic patterns for MACS-NoC throughput calculation. We leveraged these traffic patterns only to calculate average channel setup latency.

We averaged the channel setup latency over 64 simulations for each network traffic pattern. All simulations showed a low deny latency (number of clock cycles required for a deny) because of the quick release of unsuccessful channel establishment requests afforded by the minimal adaptive routing mechanism. Low deny latencies decrease the network congestion generated by network flooding since, at each hop, minimal adaptive routing generates at most two requests. Thus in some cases, the number of requests in the network may increase exponentially [11]. The grant latency (number of clock cycles required for a channel request grant) and the deny latency results for the nearest-neighbor traffic pattern simulation supports PE clusterig (i.e., producer-consumer PE pair placement on the same switch).

Table 2 shows the average grant and deny latencies calculated across all PEs in MACS-NoC for all traffic patterns. Due to the efficient mapping of the nearest neighbor traffic pattern's source-destination layout to the MACS PE clustering, the nearest neighbor traffic shows the lowest grant and deny latency. Results also reveal that traffic patterns with multi-turn routes (uniform-random, bit-complement and transpose) require slightly more cycles for a grant as compared to traffic patterns with one-turn routes (tornado, nearest neighbor). Additionally, since literature does not provide the average transfer latency for the LusalaNoC, we compared the average transfer latency for 1,280 flits for CararaNoC and MACS-NoC. CararaNoC's latency ranged from 4,700 to 9,100 clock cycles and MACS-NoC's latency ranged from 1287 clock cycles, on nearest neighbor traffic pattern, to 2,465 clock cycles on bit compliment traffic pattern. MACS-NoC provides 2x to 7x latency reduction as compared to CararaNoC. The variation in latency range is attributed to the variation in the source-destination layout of nearest-neighbors and Bit-complement traffic pattern. Since nearest-neighbor traffic leverages MACS's PE clustering, nearest-neighbors show lowest data transfer latency.

5.2.2 Average Latency and Bandwidth Utilization

Since circuit switching networks provide statically-scheduled communication paths, direct network performance comparison with dynamically-scheduled com-

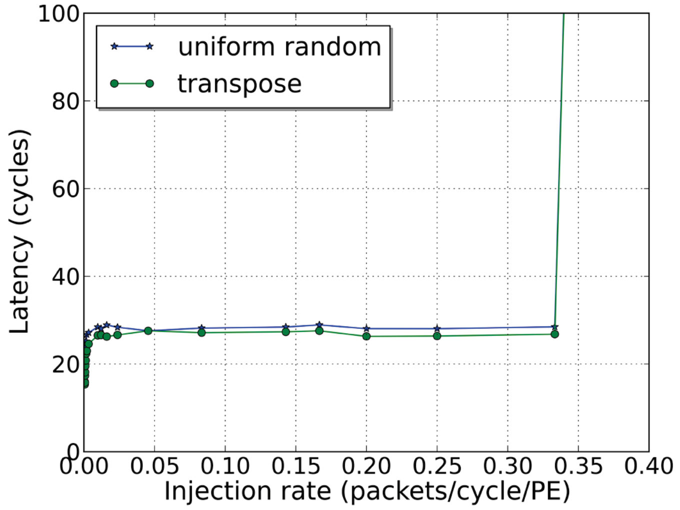


Figure 6: Average packet latency in number of clock cycles with respect to injection rate for uniform random and transpose traffic patterns for MACS-NoC. The injection rate represents the number of data packets sent by each PE per clock cycle.

munication path networks, such as packet and mixed switching networks, is not straightforward. Additionally, since data in a circuit switching network is streamed, and thus the packet size is not fixed, it is difficult to compare the average packet latency with packet and mixed switching networks.

Therefore, to provide as fair of a comparison as possible, our simulations use a packet size of P_m, which is a product of the compared NoC's packet size P_c and the simulated network traffic pattern's average hop count H for an 8x8 mesh network (i.e., $P_m = P_c * H$). We averaged the transfer latency over 128 iterations and normalize this average with P_m to calculate the normalized latency L_t. The average packet latency L_p is then calculated as a product of the normalized latency and the compared NoC's packet size (i.e., $L_p = L_t * P_c$).

We simulated an 8x8 MACS-NoC with Riviera-Pro [27] on a Linux platform for uniform random and transpose traffic patterns for different injection rates, which represents the number of data packets sent by each PE per clock cycle and compared with recent results presented in [26] and [22]. Figure 6 depicts our simulation results for MACS-NoC and reveals that MACS-NoC has similar average packet latency, approximately 30 clock cycles, for both network traffic patterns, and shows up to an approximate 6x higher network saturation points compared to [26] and [22](see appendix for visual comparison). This higher network saturation point shows that MACS-NoC can sustain higher data injection rates as compared to [26] and [22]. We attribute MACS-NoC's high network saturation point to MACS-NoC's ability to explore additional paths using minimal adaptive routing.

We calculated MACS-NoC's average bandwidth utilization for a network traffic pattern as the ratio of the total bandwidth used by MACS-NoC for that network traf-

U.S. Government work not protected by U.S. copyright.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TPDS.2015.2390631

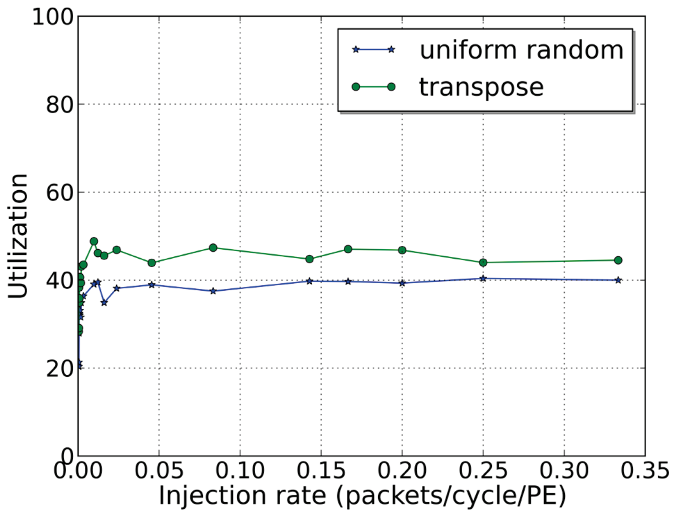


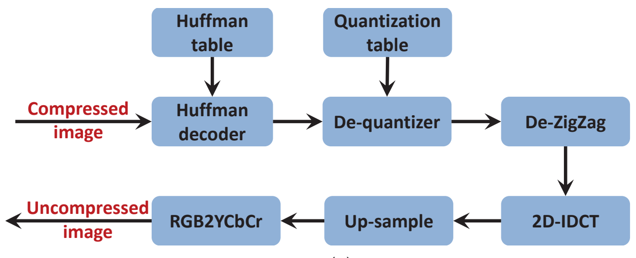
Figure 7: Average bandwidth utilization percentage with respect to injection rate for uniform and transpose traffic patterns for MACS-NoC. The injection rate represents the number of data packets sent by each PE per clock cycle.

fic pattern as compared to MACS-NoC's total available bandwidth, and compare these results with the recent results presented by Lan et al. in [18]. Figure 7 shows the average bandwidth utilization with respect to varying injection rates and depicts that MACS-NoC provides better peak bandwidth utilization, 50% in MACS-NoC as compared to 35% in [18], for transpose traffic pattern and slightly lower peak bandwidth utilization, 40% in MACS-NoC as compared to 50% in [18], for uniform random traffic pattern. Additionally, Figure 7 reveals that minimal adaptive routing alleviates the increase in network congestion introduced by MACS's PE clustering due to the exploration of additional paths and, thus, maintains high bandwidth utilization.

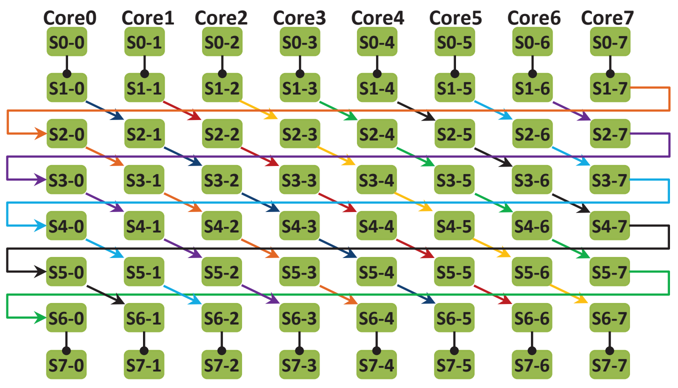
5.2.3 Trace-based Simulation

In order to demonstrate MACS's performance for real time data processing, we used a JPEG decompression core for our trace-based simulation experiments. JPEG is a stream-based data processing application, which has a defined number of data processing blocks, and thus easily maps to a MACS-NoC for simulation. Figure 8a shows JPEG decompression algorithm's block diagram, which includes the Huffman decoder, dequantizer, de-zigzag, two-dimensional inverse discreet cosine transform (2D-IDCT), up-sample, and color space converter (RGBtoYCbCr) block. Compressed image data is streamed into first block, the Huffman decoder, and decompressed image data is streamed out of last block, the RGB2YCbCr.

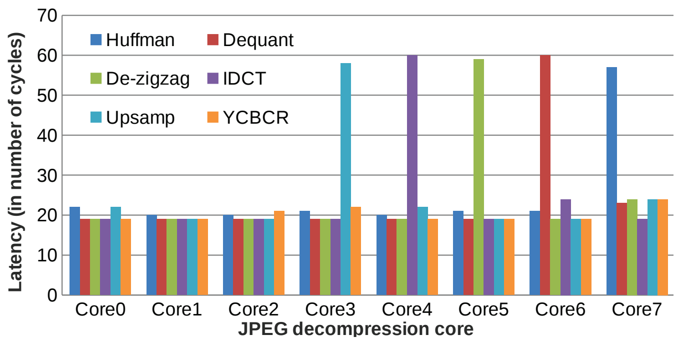
We collected simulation traces (i.e., the core's per-block image data and time stamp at the block's input and output) from an in-house JPEG decompression core implementation. We mapped and simulated parallel 8 JPEG decompression cores to an 8x8 MACS-NoC, as shown in Figure 8b. *Sn-m* denotes the (m, n) coordinates



(a)



(b)



(c)

Figure 8: (a) The JPEG decompression core's block diagram. (b) Mapping of the 8 JPEG decompression cores on an 8x8 MACS-NoC. *Sn-m* represents the MACS switches located at coordinate (m, n) and *CoreN* denotes the column where the *N*th JPEG decompression core's first block is mapped. (c) shows the per-block per-core average channel setup latencies (deny latencies are all zeroes, only grant latencies are shown in the graph) for 8 JPEG decompression cores.

for each switch. Each MACS switch connected one JPEG decompression core block to the left local PE. *CoreN* denotes the N^{th} JPEG decompression core and that core's first block is mapped as switch *S1-N*'s left local PE. Arrows with triangular heads connect the sequence of switches used by the N^{th} core starting at switch *S1-N*. The core's subsequent blocks are mapped to switches at equal increments of one row and one column. For example, if *Core0*'s first block is mapped to *S1-0*, *Core0*'s second block is mapped to *S2-1*, *Core0*'s third block is mapped to *S3-2* and so on until seventh column is reached and the mapping wraps back to column zero. Rows zero and seven are used for control purposes (depicted as arrows with circular heads) and do not

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TPDS.2015.2390631

contributed to channel latency or data transfer latency calculations. These mappings ensure a uniform and fair mapping to each decompression core in terms of minimum number of switches between a core's two blocks. Mapping applications on an NoC is a vast research area in itself and thus, is out of the scope of this paper [28].

To begin the simulation and calculate latencies, switch *S0-N* sends a data transfer start request to *CoreN*'s first block (i.e., switch *S1-N*'s left local PE) at Poisson distributed random intervals (across switch *S0-0* to *S0-7*). For example, if *S0-0* initiates a transfer request at t then *S0-1* initiates a transfer request at $t+\delta t$ where δt is a Poisson distributed random interval. For each core, a block initiates a communication channel establishment request over MACS-NoC to the next block. If the requesting block receives a deny, the requesting block uses the same retrial mechanism as in Section 5.2.1. If the requesting core receives a grant, the requesting block transmits the image data based on the image data's time-stamp to the next block. To verify transmission data integrity, we compare the received image data and time-stamp with the receiving block's input trace file. We performed 13 experiments by varying the Poisson distribution mean from 10 to 40 in intervals of 5 and 40 to 100 in intervals of 10 and calculated average channel setup latencies and average data transfer latencies over these experiments.

Figure 8c shows the per-block per-core average channel setup latencies. Since all blocks receive grants for every request, the deny latencies are zero and are not reported. Additionally, since PEs connected at the last column of each row in MACS-NoC send data to PEs connected to switches at first column in next row, these PEs show high average grant latency. The total average grant latency averaged over all PEs and all cores is ~ 23.7 cycles, which is very close to the average grant latency of the bit-complement traffic pattern simulation (22 cycles) (Section 5.2.1). We note that this MACS-NoC traffic simulation is sufficient to estimate MACS-NoC's scalable network performance because the average channel setup latency is proportional to the average number of switches along the channels irrespective of MACS-NoC dimensions.

6 Conclusion

In this paper, we introduced MACS, a highly parametric switch for a 2D mesh topology NoC. MACS uses a minimal adaptive routing algorithm with multiple path evaluation and fair path resolution to maximize bandwidth utilization. Our minimal adaptive routing and contention avoidance algorithms afford low average deny latency, which decreases channel setup latency and increases the possibility of a successful channel establishment and data transfer. To the best of our knowledge, the MACS switch is the first NoC switch to use minimal adaptive routing to explore all shortest paths and leverage route cost evaluation to maximize bandwidth utilization. Results show that minimal adaptive routing results in a 6x increase in the network saturation point, up to a 1.4x increase in the bandwidth utilization, and similar average packet latency as compared to prior work. Additionally, to reduce area overhead and increase application specialization, MACS connects two processing elements (PEs) to each switch. Whereas the system designer must strategically place producer-consumer PE pairs on common switches in order to exploit this increased performance benefit, producer-consumer PE pair placement is not required and only enhances MACSs specialization abilities. Results show that MACS offers low average channel setup latencies, a 2x to 7x decrease in channel setup latency and a 1.7x to 2x reduction in area requirements as compared to recent circuit-switched NoCs.

References

[1] L. Benini and D. Bertozzi, "Xpipes: A network-on-chip architecture for gigascale systems-on-chip," *IEEE Circuits System Magazine*, vol. 4, no. 2, pp. 18–31, 2004.
[2] L. Benini and G. D. Micheli, "Networks on chips: A new soC paradigm," *IEEE Comput.*, vol. 35, no. 1, pp. 70–78, Jan. 2002.
[3] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of Network-on-chip," *ACM Computing Surveys*, vol. 38, no. 1, Jun. 2006.
[4] T. Bjerregaard and J. Sparso, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, Washington, DC, USA, 2005, pp. 1226–1231.
[5] C. Bobda and A. Ahmadinia, "Dynamic interconnection of reconfigurable modules on reconfigurable devices," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 443–451, May 2005.
[6] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *J. Syst. Architecture: EUROMICRO J.*, vol. 50, pp. 105–128, Feb. 2004.
[7] E. Carara, N. Calazans, and F. Moraes, "A New Router Architecture for High-Performance Intrachip Networks," *Journal Integrated Circuits and Systems*, vol. 3, no. 1, pp. 23–31, 2008.
[8] N. Chin-Ee and N. Soin, "Qualitative and quantitative evaluation of a proposed circuit switched network-on-chip," in *Semiconductor Electronics (ICSE), 2010 IEEE International Conference on*. Ieee, Jun. 2010, pp. 108–113.
[9] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Design Automation Conference, 2001. Proceedings*. ACM, 2001, pp. 684–689.
[10] ——, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.
[11] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks: An engineering approach*. Morgan Kaufmann, 2003.
[12] H. ElGindy, H. Schroder, A. Spray, A. K. Somani, and H. Schmeck, "RMB-a reconfigurable multiple bus network." IEEE Comput. Soc. Press, 1996, pp. 108–117.
[13] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip:concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, May 2005.
[14] C. Hilton and B. Nelson, "PNoC: a flexible circuit-switched noC for FPGA-based systems," *IEE Proceedings - Computers and Digital Techniques*, vol. 153, no. 3, p. 181, 2006.
[15] A. Jara-Berrocal and A. Gordon-Ross, "SCORES: A scalable and parametric streams-based communication architecture for modular reconfigurable systems," in *Design, Automation and Test in Europe Conference*, 2009, pp. 268–273.
[16] F. Karim, A. Nguyen, and S. Dey, "An interconnect architecture for networking systems on chips," *IEEE Micro*, vol. 22, no. 5, pp. 36–45, Sep. 2002.
[17] Y. Lan, H. Lin, S. Lo, Y. H. Hu, and S. Chen, "A bidirectional noc (binoc) architecture with dynamic self-reconfigurable channel," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 30, no. 3, pp. 427–440, March 2011.

U.S. Government work not protected by U.S. copyright.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TPDS.2015.2390631

[18] Y. Lan, S. Lo, Y. Lin, Y. Hu, and S. Chen, "BiNoC: A bidirectional NoC architecture with dynamic self-reconfigurable channel," *2009 3rd ACM/IEEE International Symposium Networks-on-Chip*, pp. 266–275, May 2009.
[19] J. Liang, A. Laffely, S. Srinivasan, and R. Tessier, "An architecture and compiler for scalable on-chip communication," *IEEE Trans. VLSI Syst.*, vol. 12, no. 7, pp. 711–726, 2004.
[20] J. Lin and X. Lin, "Express Circuit Switching: Improving the Performance of Bufferless Networks-on-Chip," in *2010 First International Conference on Network Computing*. IEEE, Nov. 2010, pp. 162–166.
[21] A. K. Lusala and J.-D. Legat, "A sdm-tdm based circuit-switched router for on-chip networks," in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2011 6th International Workshop on*, june 2011, pp. 1–8.
[22] S. Ma, N. E. Jerger, and Z. Wang, "DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip," *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, 2011.
[23] M. Modarressi, "Virtual point-to-point connections for NoCs," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 29, no. 6, pp. 855–868, 2010.
[24] M. Modarressi, H. Sarbazi-azad, and M. Arjomand, "A Hybrid Packet-Circuit Switched On-Chip Network Based on SDM," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2009, pp. 566–569.
[25] F. G. Moraes, N. Calazans, A. Mello, L. Mller, and L. Ost, "HERMES: An infrastructure for low area overhead packet-switching networks on chip," *Integration. The VLSI Journal*, vol. 38, no. 1, pp. 69–93, 2004.
[26] Z. Qian, P. Bogdan, and G. Wei, "A traffic-aware adaptive routing algorithm on a highly reconfigurable network-on-chip architecture," *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, pp. 161–170, 2012.
[27] Riviera Pro, "Aldec inc," 2100 Logic Drive San Jose, CA 95124-3400.
[28] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *J. Syst. Archit.*, vol. 59, no. 1, pp. 60–76, Jan. 2013.
[29] E. Salminen, T. Kangas, V. Lahtinen, J. Riihimaki, K. Kuusilinna, and T. Hamalainen, "Benchmarking mesh and hierarchical bus networks in system-on-chip context," *Journal of Systems Architecture*, vol. 53, no. 8, pp. 477–488, Aug. 2007.
[30] Stratix III FPGAs vs. Xilinx Virtex-5 Devices: Architecture and Performance Comparison, "Altera inc," 101 Innovation Drive San Jose, CA 95134.
[31] N. Teimouri, M. Modarressi, and H. Sarbazi-Azad, "Power and Performance Efficient Partial Circuits in Packet-Switched Networks-on-Chip," in *2013 21st Euromicro Int. Conf. Parallel, Distrib. Network-Based Process.* Ieee, Feb. 2013, pp. 509–513.
[32] G. Wallace, "The jpeg still picture compression standard," *Consumer Electronics, IEEE Transactions on*, vol. 38, no. 1, pp. xviii – xxxiv, feb 1992.
[33] D. Wiklund and D. Liu, "Design of a system-on-chip switched network and its design support," vol. 2. IEEE, 2002, pp. 1279–1283.
[34] ——, "SoCBUS: switched network on chip for hard real time embedded systems." IEEE Computer Society, 2003, p. 8.
[35] Xpower, "Xilinx inc. usa," 2260 Corporate Circle Henderson, NV 89074 USA.
[36] C. A. Zeferino and A. A. Susin, "SoCIN: a parametric and scalable network-on-chip." IEEE Computer Society, 2003, pp. 169–174.
[37] L. R. Zheng and H. Tenhunen, "A circuit-switched network architecture for network-on-chip," in *Proceedings of the IEEE International SOC Conference, 2004*, 2004, pp. 55–58.

R. Kumar received the B.Tech degree in electrical and telecommunication engineering from Indian Institute of Technology, Varanasi, India in 2006 and the MS degree in electrical and computer engineering from the University of Florida, in 2010. He is currently working towards the PhD degree at University of Florida. His research interests involve on-chip communication networks, FPGA partial reconfiguration, and hardware software co-design.

A. Gordon-Ross (M'00) received her B.S and Ph.D. degrees in Computer Science and Engineering from the University of California, Riverside (USA) in 2000 and 2007, respectively.

She is currently an Associate Professor of Electrical and Computer Engineering at the University of Florida (USA) and is a member of the NSF Center for High Performance Reconfigurable Computing (CHREC) at the University of Florida. She received Best Paper awards at the Great Lakes Symposium on VLSI (GLSVLSI) in 2010 and the IARIA International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM) in 2010. Her research interests include embedded systems, computer architecture, low-power design, reconfigurable computing, dynamic optimizations, hardware design, real-time systems, and multi-core platforms.

Dr. Gordon-Ross is the faculty advisor for the Women in Electrical and Computer Engineering (WECE) and the Phi Sigma Rho National Society for Women in Engineering and Engineering Technology and is an active member of the Women in Engineering ProActive Network (WEPAN). She received her CAREER award from the National Science Foundation in 2010.

U.S. Government work not protected by U.S. copyright.