MACS: A MINIMAL ADAPTIVE ROUTING CIRCUIT-SWITCHED ARCHITECTURE FOR SCALABLE AND PARAMETRIC NOCS

Rohit Kumar and Ann Gordon-Ross

NSF Center for High-Performance Reconfigurable Computing (CHREC)
Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL
{kumar, ann}@chrec.org

ABSTRACT

Networks-on-Chips (NoCs) are an emerging communication topology paradigm in single chip VLSI design, enhancing parallelism and system scalability. Processing units (PUs) connect to the communication topology via routers, which are responsible for runtime establishment and management of inter-PU communication channels. Router design directly affects overall system performance and exploited parallelism. In this paper, we present a highly parametric NoC architecture, MACS, providing increased system speed, designer flexibility, and scalability as compared to previous methods. In addition, MACS enhances inter-PU communication using a circuit-switching technique with dedicated, high frequency communication channels. Compared to previous work, MACS offers a 5x increase in operating frequency and a 2x reduction in area overhead.

1. INTRODUCTION AND MOTIVATION

In order for applications to harness new capabilities made possible by the transistor explosion provided by Moore's law, applications are typically decomposed into multiple parallel processing units (PUs). For applications to exploit this increased parallelism, Networks-on-Chips (NoCs) [3][4] provide a scalable, modular communication architecture to efficiently and effectively communicate shared data and control signals across inter-PU communication channels. In NoCs, routers or switches (for packet switching or circuit switching, respectively), connect PUs to a routing fabric (PUs may connect to more than one router) and the routing fabric connects routers in a single- or multi-dimensional topology.

Currently, NoC performance is primarily restricted by three limitations: connecting a single PU to each router, inefficient routing algorithms, and inefficient packet switching methodologies. Connecting a single PU to each router can result in increased area overhead (allowing PU's to share routers reduces the number of required routers), reduced operating frequency, and increased wire length. Inefficient routing algorithms (such as deterministic routing) can cause communication bottlenecks (all lanes dedicated to communication channels) and unbalanced communication load (some routers may have many communication channels while others have few or none). Finally, inefficient switching methodologies for packetized data transfers can increase router area and reduce the communication operating frequency due to complex decoding logic and extra logic, such as counters, to monitor the number of packets.

In this paper, we address these NoC limitations with MACS, a **M**inimal **A**daptive routing **C**ircuit-**S**witching based switch for a two-dimensional mesh topology. MACS connects two PUs to each switch, providing quick channel establishment (only one switch involved) and fast data transfers (data moves directly from one PU to the other without traversing the routing fabric) for critical PU pairs. This enhancement increases system design flexibility, enabling designers to strategically place critical PU pairs on the same router. MACS efficiently distributes communication load using a minimal adaptive shortest path routing algorithm with distributed arbitration. In addition, MACS uses a simple and efficient circuit-switched routing decision state machine, resulting in high communication operating frequency. Finally, to increase design flexibility and system customization, we implement MACS as a highly parametric VHDL model with numerous tunable architectural parameters such as number of communication lanes per port and data bit width. MACS offers a 5x increase in communication operating frequency and a 2x reduction in area compared to a previous packet-switched architecture [2] and a 2x increase in communication operating frequency with only a slight increase in area compared to a previous circuit-switched architecture [6].

2. RELATED WORK

Efficient router architecture design motivated much early NoC research and in order to compare these router architectures, [5][7][8][9] provided comparisons based on different switching techniques and topologies. Wiklund et al. [9] evaluated different topologies and proved that the mesh topology was the most appropriate for on-chip networks.

Liu et al. [7] and Wiklund et al. [8] both provided strong arguments for the advantages of circuit-switched NoCs over packet-switched NoCs. Liu et al. [7] proposed a Time Division Multiplexed (TDM) scheme for

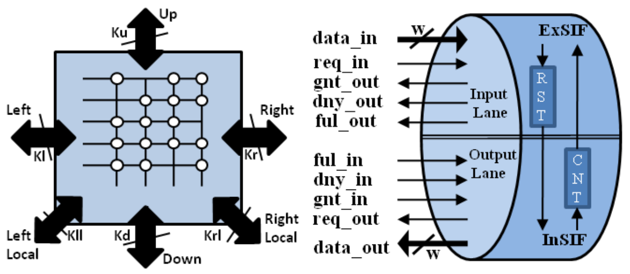


Fig 1. MACS's switch architectural layout (left) and port details (right) (all ports are similar with individually parameterized number, denoted by 'K's, of similar lanes). Lane details show all signals and routing control logic blocks (ExSIF/InSIF) associated with each input/output lane. ExSIF/InSIF coordinate using status tables (RST and CNT).

communication channels on a two-dimensional mesh NoC. 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, TDM could suffer from communication bottlenecks.

To alleviate these bottlenecks, Wiklund et al. [8] proposed a mesh topology NoC (SoCBUS) using distributed arbitration. SoCBUS explored only one shortest path (even though many existed) and did not consider communication load balancing. In addition, the authors suggested that SoCBUS was not suitable for networks with random traffic.

In order to provide better communication bandwidth for random traffic, Ahmadinia et al. [1] proposed RMBoC (a circuit-switched NoC). RMBoC had several drawbacks including deterministic routing that did not consider communication load balancing, large area requirements due to a separate network controller for each dimension, and low communication operating frequencies due to a complex routing algorithm.

Hilton et al. [6] presented the Programmable Network-on-Chip (PNoC), a highly flexible circuit switched NoC for FPGA systems. PNoC's tunable parameters included the number of router ports, data, address bus widths, and number of PUs attached to a router. All PUs connected to the same router formed a subnet and these PUs shared a common address. Therefore, during a connection establishment request to a particular subnet, only one PU could communicate at a time. Whereas this technique appeared to be inflexible, the technique was highly suitable for processor-farm systems. Furthermore, PNoC could not consider communication load balancing unless the operating system could update the routing table during run-time.

In order to address limitations of previous architectures, we introduce MACS, a two-dimensional mesh topology circuit-switched architecture with distributed arbitration. We implement MACS as a highly parametric VHDL model with numerous tunable architectural parameters. Compared to previous work, MACS is a low area architecture that reduces communication establishment bottlenecks using a minimal adaptive routing algorithm to ensure alternate path exploration in orthogonal directions, increases architectural specialization, and provides high communication operating frequencies.

3. MACS ARCHITECTURE

Fig 1 (left side) depicts MACS's switch architectural layout. X and Y coordinates identify individual switch addresses based on horizontal and vertical positions, respectively, in the two-dimensional topology. Each switch's two PUs are addressed relative to their connected switch. A MACS switch 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 PUs. 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 PU pairs (denoted by the '*K'* tunable architectural parameters in Fig 1). A *lane identification number* (LID) uniquely identifies each port's input and output lane. Each switch/local port can be specialized with different numbers of input or output lanes providing fine-grained per-direction communication bandwidth specialization. Furthermore, each input lane consists of several control signals (*req_in, gnt_out, dny_out,* and *ful_out*) and *W* data signals (*data_in*). Similarly, each output lane consists of several control signals (*req_out, gnt_in, dny_in,* and *ful_in*) and *W* data signals (*data_out*). Control signals negotiate channel establishment and data signals provide inter-PU communication bandwidth.

3.1 SWITCH OPERATION

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

Each port contains two types of routing control logic blocks (signal forwarders) for channel establishment: an External Signal Forwarder (ExSIF) and an Internal Signal Forwarder (InSIF). Signal forwarders are responsible for controlling all communication operations such as communication request servicing and channel establishment negotiations and record necessary channel routing information in *status tables.* Table 1 depicts the

Table 1. Status table details for each input lane (ExSIF maintains the RST) and output lane (InSIF maintains the CNT).

Request Service Table (RST)	
Field	Description
req_servd	Request has been serviced by all possible destination ports
C0Pavl	Output lane available on one of the possible destination port
C1Pavl	Output lane available on other possible destination port
C0PID	PID of port associated with C0Pavl
C1PID	ID of port associated with C1Pavl
C0LID	ID of the lane that has been reserved at port C0
C1LID	ID of the link that has been reserved at port C1
Connectivity Table (CNT)	
PID	Port on which channel establishment request has been received
LID	Input lane on which channel establishment request has been received

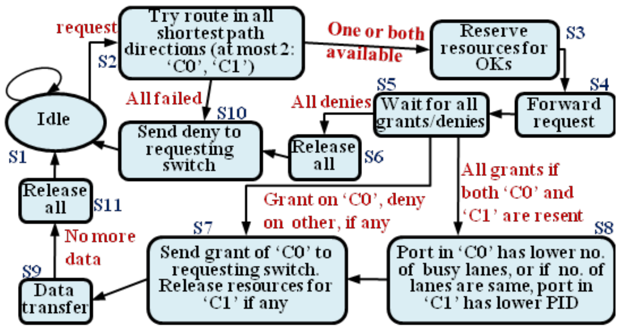
status table details for the request service table (RST) maintained by the ExSIF and the connectivity table (CNT) maintained by the InSIF. Channel routing information specifies lane availability and input-output port lane connections. The ExSIFs, InSIFs and the status tables play a key role in channel routing, as discussed in Section 4.

4. CHANNEL ROUTING ALGORITHM

Establishing an arbitrary inter-PU communication channel on a two-dimensional mesh is a straightforward process, but however, choosing the best communication channel given all potential routes is challenging. For example, minimal adaptive routing chooses the shortest path between two points in a two-dimensional mesh, thus defining the best route as simply the shortest path. However, between two points in a two-dimensional mesh, there exists $(\Delta X + \Delta Y)! / ((\Delta X)! + (\Delta Y)!)$ equal length shortest path routes where ΔX and ΔY are the differences between the X and Y coordinates of the two points, respectively. 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.

Our communication channel routing algorithm is based on minimal adaptive routing to establish, maintain, use (transfer data), and release inter-PU communication channels, which correspond to 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). Fig 2 depicts a state graph 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



request
Try route in all shortest path directions (at most 2: 'C0', 'C1')
S2
One or both available
Reserve resources for OKs
S3
Idle
S1
All failed
S10
All denies
S5
Wait for all grants/denies
Forward request
S4
Send deny to requesting switch
Release all
S6
Release all
S11
Grant on 'C0', deny on other, if any
All grants if both 'C0' and 'C1' are resent
No more data
S7
S8
S9
Data transfer
Send grant of 'C0' to requesting switch. Release resources for 'C1' if any
Port in 'C0' has lower no. of busy lanes, or if no. of lanes are same, port in 'C1' has lower PID

Fig 2. State diagram for a switch's channel phase actions and transitions.

and begins the request service phase. In S2, the switch compares its address with the destination switch address and determines (according to the minimal adaptive algorithm) the potential destination ports ('C0' and/or 'C1') in which to forward the incoming request. If the current and destination switch addresses do not match, the requested PU is not connected to the current switch and the current switch must forward the request to (a) neighboring switch(es). Otherwise, the request is forwarded to the appropriate local PU port. If there is an available output lane on 'C0' and/or 'C1', the switch transitions to S3 to establish input-output lane connections. In S3, the requesting port's ExSIF adds the appropriate entries to the RST (Table 1 (top)) and the destination port's InSIF adds the appropriate entries to the CNT (Table 1 (bottom)). After adding these entries, the switch transitions to S4 and forwards the request to the available lane on port(s) 'C0' and/or 'C1' and completes the request service phase. On the other hand, if there are no available output port lanes, routing at this switch fails and the switch transitions to S10, sends a deny response to the requesting switch, and transitions back to the idle state (S1) (the status tables are unchanged).

After successful completion of the request service phase, the switch enters the grant/deny phase (S5) and waits for grant and/or deny signal responses (*gnt_in* or *dny_in*, respectively) on the output lane of port(s) 'C0' and/or 'C1'. If only one port (direction 'C0') 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 and has already been stored in the CNT in S3). If both ports 'C0' and 'C1' 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 deny to the requesting port and transitions back to the idle state (S1). In the second situation, the switch receives one grant and one deny (associated with 'C0' and

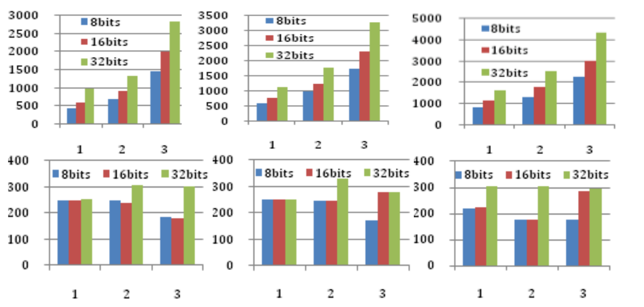


Fig 3. Area usage in number of slices per PU (top row) and maximum operating frequency (bottom row) for data widths W = 8, 16, and 32 bits for a varying number of lanes per switch and local port. The x-axis in each graph varies the Kl, Kr, Kd, and Ku parameters from 1 to 3 lanes per switch port. From left to right, the graphs vary the Kll and Krl parameters from 1 to 3 lanes per local port.

'C1', respectively, or vice versa). The switch transitions to S7, forwards *gnt_in*, *dny_in,* and *ful_in* from 'C0' to the requesting port, and releases resources associated with 'C1'. (Note that the case is similar when 'C0' denies and 'C1' 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. After the best port is selected (denoted as 'C0' in S8 and S7), the switch transitions to S7, forwards *gnt_in*, *dny_in,* and *ful_in* of port 'C0' to the requesting port's input lane using information in C0's output lane's CNT, and releases the resources associated with port 'C1' using information in the requesting port's input lane's RST. This algorithm is deadlock free because in all situations, the switch forwards/sends either a grant or 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 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 (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. 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, either due to data transfer completion or the denial of a channel establishment request, to free all associated channel resources. The switch transitions to S11 and releases channel resources by removing corresponding status table entries.

5. RESULTS

We implemented the MACS switch as a highly parametric VHDL soft core providing architectural parameters to specify the number of switch port lanes (*Kl* (left), *Kr* (right), *Kd* (down), *Ku* (up)), local port lanes (*Kll* (left local), *Krl* (right local)), and lane data width *W*. Given the prohibitively large design exploration space for all possible combinations of all architectural parameters, we fix 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*). Our design targeted the Xilinx Virtex-II Pro XC2VP30-7FF1152 device.

For each combination of *Kl*, *Kr*, *Kd*, *Ku* and *Kll*, *Krl,* we evaluated area usage and maximum operating frequency. We measured maximum operating frequency after place and route using the Xilinx static timing analysis tool, *trace*, with no clock constraint (*trce -v-u-a*). We used the Xilinx ISE simulator to simulate the design.

5.1 AREA USAGE AND TIMING ANALYSIS

Fig 3 (top row) depicts switch area usage per PU (total switch area is twice these values since each switch has two PUs). The x-axis in each graph varies the *Kl*, *Kr*, *Kd*, and *Ku* architectural parameters from 1 to 3 lanes per switch port. From left to right, the graphs vary the *Kll* and *Krl* parameters from 1 to 3 lanes per local port. In addition, each graph also depicts the area usage for data widths *W* = 8, 16, and 32 bits. For example, the area usage for *Kl* = *Kr* = *Kd* = *Ku* = *Kll* = *Krl* = 1 and *W* = 16 bits is 576 slices per PU, which equates to only 4.2% of available slices on our test device

Fig 3 (bottom row) depicts maximum operating frequency (for the same parameter values as discussed in the previous paragraph) and shows that MACS can achieve high operating frequencies ranging from 170 MHz to 308 MHz.

We evaluate MACS compared to two previous works, a packet-switched architecture [2] and a circuit-switched architecture (the PNoC) [6], in terms of area overhead (in FPGA slice and block-RAM (BRAM) requirements) and attainable operating frequency. We point out that direct comparison with previous work is difficult due to a large variation in tools, devices, and architectural layout. To provide as fair a comparison as possible, we choose similar architectural layouts i.e. topology (two-dimensional mesh), number of PUs (8), number of lanes per port (1), data width (16 bits), and the same device platform (Xilinx Virtex-II Pro). For MACS, since edge switch ports (those on the periphery of the mesh) do not connect to any neighboring switches, we tied these switch ports to ground, and thus allowed the synthesis tool to optimize edge switch ports and reduce area requirements.

Table 2 compares MACS with the previous NoC architectures. We obtained the area and frequency values directly from literature for the packet-switched architecture [2] and PNoC circuit-switched architecture [6]. Compared to a packet-switching architecture, MACS provides a 2x reduction in area requirements and a 5x improvement in operating frequency. When compared to PNoC, MACS imposes a slight area increase but provides a 2x improvement in operating frequency.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we introduce MACS, a highly parametric two-dimensional switch mesh topology NoC. MACS uses a minimal adaptive routing algorithm with multiple path evaluation for dynamic communication channel establishment and communication load balancing. 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 route cost evaluation for communication load balancing. In addition, to reduce area overhead and increase application specialization, MACS connects two processing units (PU) to each switch. Whereas the system designer must strategically place critical PU pairs on common switches in order to exploit this increased performance benefit, critical PU placement is not required (this placement only enhances MACS specialization abilities). Results show that MACS is highly scalable and achieves high operating frequencies even for systems with large data buses. Results show that MACS offers a 5x increase in communication operating frequency and a 2x reduction in area compared to a previous packet-switched architecture and a 2x increase in communication operating frequency with only a slight increase in area compared to a previous circuit-switched architecture.

Future work includes switch power analysis and reducing communication channel establishment latency by improving the routing algorithm's round-robin arbitration. In addition, we plan to explore bi-directional request and data buses to optimize communication, as well as protocol development. Detailed simulation is planned for visualization and analysis of communication load balancing. Finally, we will provide a design exploration script with MACS to assist designers in per application architectural specialization.

Table 2. Comparison of 8 PU two-dimensional mesh topology NoCs

Network Architecture	Slices	BRAMs	Frequency
Packet-Switching [2]	2400	8	50 MHz
PNoC [6]	1223	1	134 MHz
MACS	1478	0	251 MHz

7. ACKNOWLEDGEMENTS

This work was supported in part by the I/UCRC Program of the National Science Foundation under Grant No. EEC-0642422. We gratefully acknowledge tools provided by Xilinx.

8. REFERENCES

[1] Ahmadinia A, Bobda C., Ding J., Majer M. and Teich T. "A Practical approach for Circuit Routing on Dynamic reconfigurable Devices". International Workshop on Rapid System Prototyping, 2005, pages 84-90.

[2] Bartic, A., Mignolet, J.Y., Nollet, V., Marescaux, T., Verkest, D., Vernalde, S., and Lauwereins, R. "Highly scalable network on chip for reconfigurable systems".In Proceedings of International Symposium on System-on-Chip, 2003, pages 79–82.

[3] Benini L. and Micheli G. De, "Networks on Chips: a New a SOC Paradigm", IEEE Computer, 2002, pages70-78.

[4] Dally W. J. and Towles B. "Route Packets, Not Wires: On-Chip Interconnection Networks". In Proceedings of the 38th Design Automation Conference, 2001, pages 684-689

[5] Duato J., Yalamanchili S., and Ni L. M. "Interconnection networks: an engineering approach", 1997. ISBN 0-8186-7800-3.

[6] Hilton C. and Nelson B., "PNoC: a flexible circuit-switched NoC for FPGA-based systems". In Proceedings of Computers and Digital Techniques, 2006, pages 181-188.

[7] Liu, J., Zheng, L. R., and Tenhunen, H.: "A circuit-switched network architecture for network-on-chip". In Proceedings of International Symposium on System-on-Chip, 2004, pages 55–58.

[8] Wiklund D. and Liu D., "SoCBUS: Switched Network on Chip for Hard Real Time Embedded Systems". In Proceedings of International Parallel and Distributed Processing Symposium, 2003, pages. 78-85.

[9] Wiklund D. and Liu D. "Design of a system-on-chip switched network and its design support". In Proceedings of the International conference on communications, circuits and systems (ICCCAS),2002, pages 1279-1283