# GANA: A Novel Low Cost Conflict Free NoC Architecture

EITAN ZAHAVI, ISRAEL CIDON AND AVINOAM KOLODNY
Electrical Engineering Department, the Technion – Israel Institute of Technology

Similar to off-chip networks, current NoC architectures are based on the store and forward of uncoordinated end-to-end packet transmissions through autonomous buffered routers. However, the monolithic nature and the small physical dimensions of on chip networks open up the opportunity for much more tightly controlled architectures. We present GANA, a new Global Arbiter NoC Architecture. In GANA, the transmission of end-to-end data is timed by a global arbiter in a way that avoids any queuing in the network. The arbitration takes into account the complete transfer of the end-to-end packets through the entire network path, avoiding any intermediate queuing and hop-by-hop packet arbitration. Consequently, buffers and arbiters are no longer required in the routers, resulting in smaller area and low power consumption. It is demonstrated through detailed design and full synthesis that the additional area of the central arbiter and the control path are negligible in comparison to the provided area saving. For example, an 8x8 GANA consumes only 16% of the area of an equivalent autonomous NoC while providing a better end-to-end throughput. The end-to-end performance of GANA at high network loads is typically much better than in a distributed-control NOC, because resource contention and queuing in the network are avoided. This comes at the cost of a few percentage increase in latency at light loads due to the additional arbitration phase. GANA architecture combines the inherent benefits of a network (parallelism and spatial reuse of links) with the inherent benefits of high integration (global view of the system state, central control, and synchronization). The scalability of GANA is evaluated analytically, showing that it can be superior to fully-distributed networks in systems up to a size of about 100 modules manufactured in 45nm technology, which can be used today as well as in the foreseeable future.

Categories and Subject Descriptors: C.2.1 [Network Architecture and Design]: Network on Chip – *Central Arbitration*
General Terms: Design, Network on Chip, Arbitration
Additional Key Words and Phrases: Arbitration Fairness,

## 1. INTRODUCTION

Network on Chip (NoC) architectures have evolved in the last decade as a key technology for System on Chip (SoC) and Chip Multi Processor (CMP) designs [1]. The prevalent NoC architecture is an embedding of a 2D interconnection mesh network within the device. In such a NoC, each vertex of the mesh consists of an autonomous router which stores, routes, arbitrates and performs time-multiplexing of packets received at an input port and forwarded to an appropriate output port. To support intermediate packet arbitration, routing and virtual channel (VC) logic at full wire speed, routers typically need 2 to 4 pipe stages and 3 or more buffers per VC.

Authors' addresses: Eitan Zahavi, Israel Cidon and Avinoam Kolodny, Department of Electrical Engineering, The Technion – Israel institute of technology, Haifa 32000, Israel. Permission to make digital/hard copy of part of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date of appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.
© 2001 ACM 1073-0516/01/0300-0034 $5.00

Several router enhancements to reduce the router latency can be added [16][6], albeit they

increase the area and power of the NoC.

We argue that this common architecture of NoCs has been borrowed from wide-area macro-networks, where link distances and latencies are large, while the cost of additional out-of-band communication is prohibitive. Indeed, modern on-chip communication requires the inherent capability of networks, namely sharing and spatial reuse of wires, in order to provide the necessary throughput and scalability.

However, on-chip distances are small, additional wires are cheap, and synchronization is quite possible in the foreseeable future, up to fairly large system sizes of around 256 modules. Therefore, in NoCs it is possible to combine the benefits of data-parallelism (as in macro-networks) with the benefits of centralized control (conceptually as in bus-based systems).

This paper presents a novel NoC architecture that performs NoC input packet arbitration and time scheduling in a global fashion. The input arbitration by a central arbiter takes into account all flit movements in a fully synchronized and precisely predictable way. As a consequence of these accurate timing calculations, the central arbitration can guarantee that all flits will traverse the NoC without any conflict at any intermediate router, experiencing a full path cut-through latency. We term this architecture GANA (Global Arbiter NoC Architecture). Through a detailed design and synthesis we show that overall, taking into account the arbitration overheads, GANA saves a considerable amount of area and power due to the elimination of router's buffers, VCs and arbiters. These area and power savings are much larger than the corresponding area and power added for the sake of the central arbitration system, including both its communication and computation parts. It is demonstrated that an 8x8 GANA saves 84% of the area and 24% of the power compared to an equivalent traditional NoC (at 25% load) while providing better end-to-end throughputs. Under light traffic loads, the latency of GANA is slightly higher in comparison with the traditional NoC because of the extra request/grant phase associated with the central control unit. However, for medium to high traffic loads, GANA provides a superior performance, since its control mechanism guarantees a conflict free network traversal eliminating any queuing delays in the routers.

Our work follows and extends the area saving ideas of Æthereal Guaranteed Service, [4][5] which relies on availability of the (static) SoC traffic matrix to pre-calculate periodic arbitration slots which in turn reduces buffering costs. We extend these ideas to handle online adaptive scheduling to support the dynamically-varying traffic patterns of a CMP. Clearly, GANA can also be employed in a SoC environment, especially when traffic patterns cannot be predicted ahead of time (e.g. MPSoC). The GANA idea also relates to previous work on optical burst reservation [2] which uses out-of-band control

lines and circuits to dynamically allocate data links to specific flows in a buffer-less system. We extend that work by introducing centralized control and applying these ideas to NoC. Previous work on buffer-less NoC, relying on ‘‘deflection’’ routing to avoid the need for message buffers, is limited to low network loads [15][14]. Our approach is able to remove both the buffers and arbiters from the NoC and support high network throughputs.

Unlike most previous NoC architectures, our tightly controlled architecture has no comparable in off-chip networks. Therefore, the inevitable question is 'What makes this architecture possible and scalable in this environment?' The answer lies in the core capability of on-chip technology that was not fully exploited for packet based NoC architectures: Off-chip interconnection networks of different scales cannot synchronize the arbitration across multiple nodes due to the variable and large latencies imposed by the long wires and the distributed clocking. These variable latency links should be used both for the data packet transmissions and for communicating the arbitration information gathering and commands. In contrast, the monolithic nature of NoCs provides low latencies and enables synchronous or mesochronous on-chip clock. The relatively small wire delays over the N x N mesh links (as compared to macro networks) enables a global arbiter to receive requests from each network interface within about N cycles, preferably using an out-of-band and highly predictable dedicated lines (similar to the approach taken by [12]).

GANA connectivity is an overlay of two layers described in Figure 1. The network elements that carry data packets are shown in Figure 1(a). Although they are laid out similarly to the common NoC, buffering is provided only at the Network Interfaces (NI), and no packet arbiters are placed in the Network Routers (NR), which are actually equivalent to simple latched crossbar switches. Figure 1(b) shows the Global Arbitration Unit (GAU) located in the middle of the chip. The GAU is connected by Transmission Request (TR) and grant lines to every NI. It is important to note that massive amounts of data can be transmitted through the distributed NoC structure of Figure 1(a), exploiting the parallel operation of all links to provide the required throughput. In contrast to this, the wires connected to the GAU in Figure 1(b) carry only a few control bits. These wires leading to the central unit may be long, but repeaters and flip-flops are inserted along their path, such that the velocity of signal propagation along these wires is similar to the propagation in NoC links.

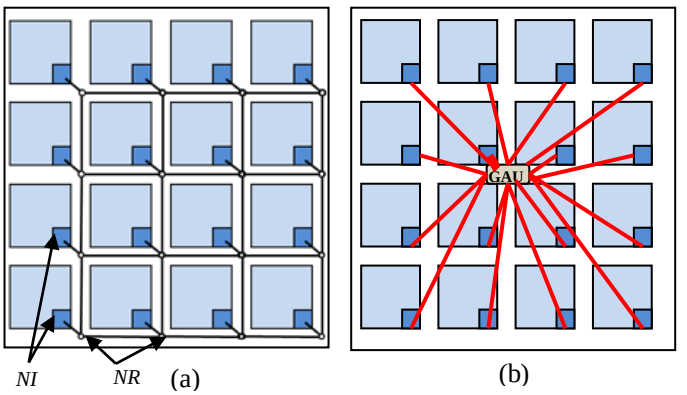



Fig. 1. A 4 x 4 Global arbiter NoC Architecture (GANA).
a) Data plain containing NoC Interfaces (NI) and very simple Network Routers (NR);
b) Control plain connections of the Interfaces to the GAU

The rest of the paper is structured as follows: section 2 describes GANA principles; Section 3 describes GANA's detailed micro-architecture; Section 4 evaluates GANA in terms of area, power and performance; Section 5 provides analysis of GANA latency and discusses scaling. Section 6 provides our conclusions.

## 2. THE GLOBAL ARBITER NOC ARCHITECTURE

### 2.1 Operation Example

We first provide an example for the basic operation of GANA. Our example assumes a 4 x 4 Mesh topology using static, dimension-order XY routing as depicted in Figure 2. Let further assume that two network interfaces are set to send data: NI 0 needs to send a packet of 4 flits to NI 11 and NI 2 needs to send a packet of two flits to NI 15. Prior to t=0, the GAU inspects the requests and concludes that while both flows use links C, D and E, both NIs can start transmission at t=0 without any future contention. First, the GAU checks the previous allocations of the links A, B, C, D and E. Assuming that these links were not previously allocated to any flow in any future cycle, the earliest time the requests can be granted is t=0 since the second (and last) flit of the flow from NI 2 will leave link C at t=1 while the first flit of the flow from NI 0 will reach link C at t=2.

Once NI 0 and NI 2 receive the grant signal from the GAU they immediately place the first flit of their packets on the module output, and continuously place the next flits every cycle until their packets transmission is complete. The flits traverse the NoC toward their destination without experiencing any port collision, making a one hop progress of every single cycle. Assuming NI 2 needs to send a second packet to NI 15, it can place its request at t=1. The GAU now inspects the link availability: the first cycle link C is

available at t=6 (after the tail of the first 4 flits packet of NI 0 is passed). So the GAU grants the new request at t=6.

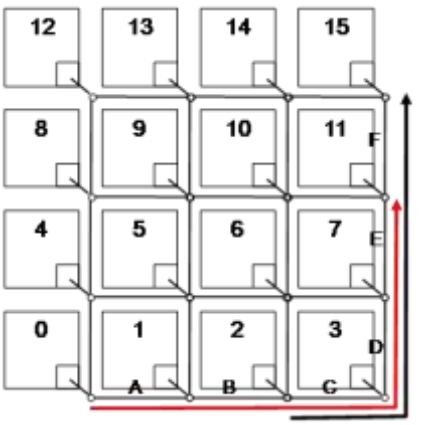



Fig. 2.Flows from core 0 to 11 and 2 to 15 using common links C, D and E but GAU is able to avoid contention by controlling the exact time each packet is injected into the network

## 2.2 Building Blocks Requirements

There are different ways to implement a generic global arbitration in a NoC [20]. The architecture presented in this paper is based on several design choices. The most important choice that leads to most of the area saving is to implement an accurate link-by-link conflict free scheduling. Here, the global arbiter plans which link in the network will deliver traffic at each and every cycle. This provides a full congestion avoidance that enables the removal of all intermediate buffers from the network. This decision leads to several behavioral requirements from the GANA building blocks:

Network interfaces must precisely follow the GAU instructions. They must send flits exactly at the granted time, and maintain a continuous flow of the flits of the packet. These requirements translate into a simple NI design and simple transmission gating logic. On the receive side, the NI is required to be able to always consume packets at the full rate of its input link. Accurate scheduling requires the GAU to plan the usage of the NoC links such that no collision of two flits on the same link ever occurs. It also requires the GAU to assure fairness so that network interfaces do not starve. As packet arbitration is taken off the routers, the requirements from routers are minimized to route the received flits to the correct output port. Since the GAU assures that no link contention is possible, no buffering is required within the routers. Figure 3 depicts the structure of today's common NoC router. The blocks that are not needed in a GANA router are crossed out. The remaining blocks are the crossbar switch and the routing logic which controls its data path.

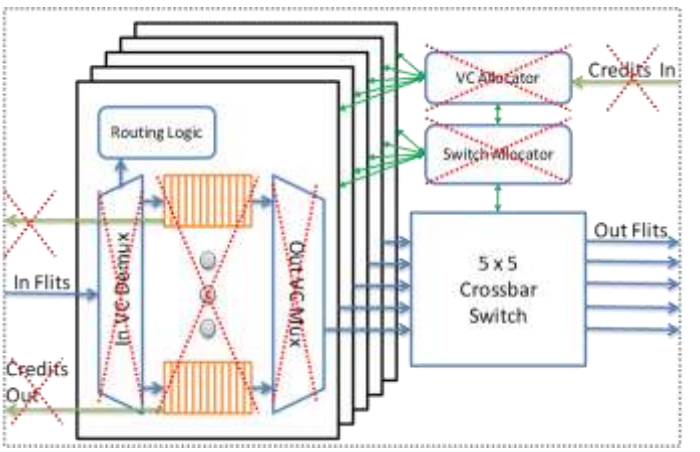


Fig. 3. A common mesh router. Blocks not needed for the GANA mesh router are crossed out. The only remaining blocks are the crossbar switch and the router logic.

For the sake of simplicity, some common architectural choices are made in this paper. We assume the classical wormhole approach of providing the destination address only in the header flit (e.g., GAU is required to schedule all the packet's flits to be transmitted continuously). Static XY routing was selected, while each NI can have a single outstanding request.

## 3. GLOBAL ARBITER UNIT MICRO-ARCHITECTURTE

### 3.1 GAU State Registers

Transmission requests arrive from each NI to the GAU and are stored in $\mathbf{N}^2$ request registers within the GAU **–** one for each NI.

The task of the GAU is to manage the planned usage of each (unidirectional) link in the **N x N** mesh over a future time window of **F** clock cycles. The future allocation of links is tracked in usage registers, one for each link. Since each bidirectional link in the NoC has 2 links and each NI connects to a router via 2 (directed) links, the total number of usage registers (and unidirectional links) in the network is $6\mathbf{N}^2$-$4\mathbf{N}$. Usage registers are bit-vectors of a size of **F** bits. The $\mathbf{j}^{th}$ bit of a usage register indicates that the link is scheduled to pass traffic at clock cycle **j** (relative to the current time t=0).

The arbitration result for each NI is stored in grant registers of a size of **F** bits. The $\mathbf{j}^{th}$ bit of a grant register indicates that the corresponding NI should send a flit at (exactly) cycle **j**.

In every clock cycle all the usage and grant registers are shifted to the right, removing the already used information. Request registers are loaded when the previous request is

granted or if they are empty. So a network interface may send a new request whenever a grant is received. Two requests may be sent from one network interface before the first grant.

### 3.2 Computing the Grant Window

To service a packet transmission request, the GAU initiates a greedy algorithm to find the earliest contiguous sequence of time slots – named ‘‘grant window’’ which allows non-stop transmission of the whole packet along the full route from source to destination.

The non-stop transmission is required in order to avoid the need for buffering inside the network. This means that when a flit is sent by a NI, all the links along its way to the destination should be free when the flit arrives. The example in Figure 4 shows a packet path traversing through three links. The usage registers of these links are drawn one below the other in the order of the path traversal.

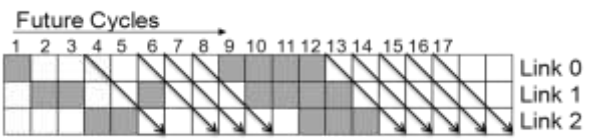



Fig. 4. Possible future cycles for Non-Stop flit transmission along the path from source to destination. Previously allocated cycles are shown darkened on the usage registers. Arrows show valid cycles.

The gray coloring indicates that the link is already allocated in this particular future cycle. In the example of Figure 4 the first link is pre-allocated at t=1,9,10,11,12, so new flits cannot be allocated in these future cycles. A flit injected at the first non-allocated cycle of link 0, at t=2, reaches link 1 at t=3 and finds it is already allocated. Similarly, a flit sent at t=3 reaches link 2 at t=5 and finds it pre-allocated too.

The possible non-stop sending times are marked by ‘‘free’’ arrows. The GAU calculates them using a sequence of bitwise-OR over the path’s usage registers, where each register is shifted to the right by its distance from the source. We term the intermediate result of this OR along the path ‘‘aggregated usage’’. This computation stage is termed the ‘‘propagation phase’’, as the information represented by the diagonal arrows is propagated on the path from source to destination.

The propagation phase provides a set of possible ‘‘free’’ cycles in which a flit can be sent from the source and reach the destination without being buffered. However, for a wormhole routing based NoC, it is required that the entire packet of $\boldsymbol{L}$ flits is sent on $\boldsymbol{L}$ contiguous cycles. Therefore, a logic circuit called ‘‘find-first-range’’ receives the possible set of cycles and searches for the first $\boldsymbol{L}$ consecutive free cycles. This selection is done in the second phase of computation, named the FFR phase. For the example of Figure 4,

assuming the packet length is 3 flits, the first free range starts at cycle 6. The resultant ‘‘grant window’’ will consist of cycles 6, 7 and 8.

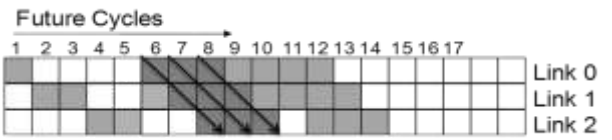



Fig. 5.The usage registers after being updated with the ‘‘Grant Window’’ matching a three flits packet. To support ‘‘Wormhole’’ routing this window is the first contiguous three free transmission cycles

Once the ‘‘grant window’’ is assigned in the find-first-range phase, the usage registers of the links on the path should be updated by marking the appropriate bits as allocated. The updated content of the usage registers for our example is shown in Figure 5. The process of updating the usage registers is performed in the ‘‘allocation’’ phase. Note, how the grant window is shifted by one clock every hop on the path from the source to destination.

The result of the ‘‘grant window’’ computation is written into the grant register associated with the requestor NI (bitwise-OR’ed with previous set bits). Once granted, the request stored in the request register may be replaced with a new one. The stage of the algorithm, in which the grant and request registers are updated, is termed the load phase.

If the find-first-range circuit cannot find a valid grant window, no allocation is performed on the path back to the source (no update is performed to the usage registers). The grant register is not loaded and the request is kept pending in the request register. As the usage registers are shifted to the right every clock cycle, the same request may be granted after few cycles.

The four phases described above, which are propagate, find-first-range, allocation and load, form a complete scheduling cycle. Since the allocation phase writes the same ‘‘usage’’ structures being read during the propagate phase, the GAU micro architecture described in this section (and the rest of the paper) does not allow a pipelined implementation of the scheduling cycles.

As different network interfaces have different distances from the GAU, the request and grant lines take different number of cycles to propagate data to and from the GAU. It is crucial for the GAU to consider these different latencies, such that exact transmission timing on the NoC data lines can be maintained. For that sake, the GAU may equalize the request latency by adding flip-flops, or calculate the appropriate future cycle for every request according to distance. Similarly, the grant register should be shifted to accommodate the grant latency of the particular NI it connects to.

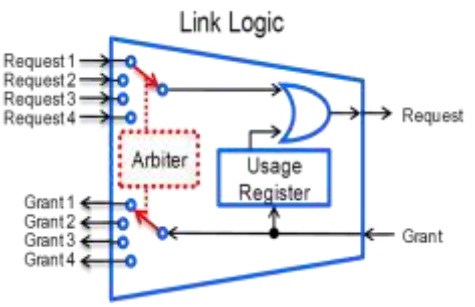
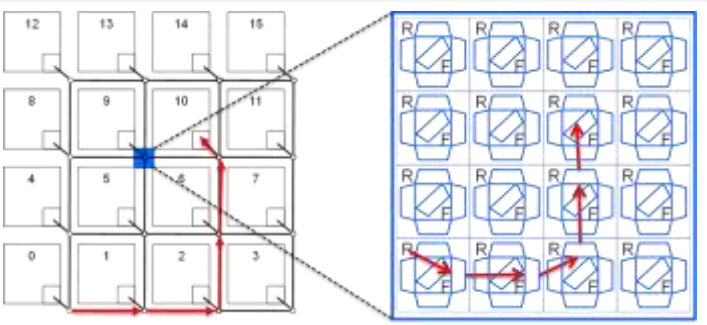



Fig. 6. GAU Hardware per Link:

Requests that are routed through the link are arbitrated. The winner request aggregated usage propagates forward after being OR'ed with the link usage register. The computed ''Grant Window'' updates the usage register and propagates backwards through the arbitrated input.

The hardware involved in the ''grant window'' calculation, which is replicated for each link, is depicted in Figure 6. Each output link is handled by link logic which includes the usage register and it's bitwise-OR with the previous links aggregated usage. The four input ports Request1 to Request4, represent the multiple input links that may feed the output link. Grant window bits, provided on the Grant input, are loaded into the usage register and passed to the appropriate input.




(a) (b)

Fig. 7. (a) A path from NI-0 to NI-10 in the NoC (b) The Global Arbitration Unit (GAU) is a small block located near the center of the die, built as an array of identical circuits. Each of the circuits corresponds to a router in the NoC.

The GAU, a small block located near the center of the die, is an array of identical circuits (See Figure 7(b)). Each of the circuits corresponds to a router in the NoC. Each of such circuits includes link logic for all the ports of the router, a single request/grant logic for the local core, and FFR logic. For demonstration purposes we illustrate a path in the NoC for a particular source destination pair in Figure 7 (a), and show the associated link-logic elements representing the path within the GAU.

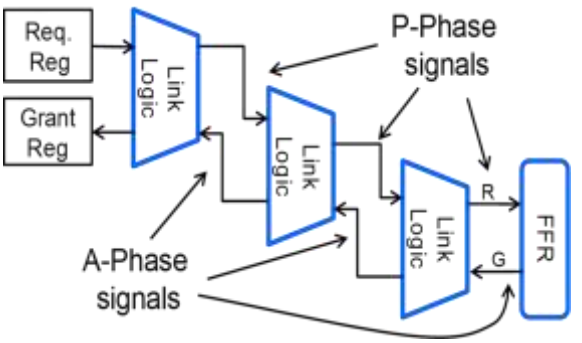



Fig. 8. The hardware involved in a single request.

In the Propagation phase, previous allocation information is aggregated and passed forward through a series of links. In the FFR phase: Find-First-Range logic, at the destination, computes the ‘‘grant Window’’. In the Allocation phase: If a valid window exists, it is propagated back through the links and finally loaded into the grant register.

The complete path from the source to the destination is shown in Figure 8. It starts from the request register and traverses the link logic circuits towards the destination. The aggregated usage is processed by the FFR and the resulting ‘‘grant window’’ (if any) is sent back over the path, updating the allocated data into each link usage register. Finally, when the source is reached, the grant register is loaded with the ‘‘grant window’’.

### 3.3 Arbitrating Several Requests

So far we have showed how the GAU calculates the ‘‘grant window’’ for a single transmission request. This subsection addresses the case in which several requests are concurrently issued by different NIs.

If all of the different requests traverse disjoint, non-intersecting paths, the mechanism described above can handle all of them in parallel. In such a case no arbitration is required. However, if several requested paths intersect, their requests are arbitrated. In each scheduling cycle the GAU selects only one of the intersecting requests to propagate through each link logic, as represented by the switches and arbiter in Figure 6. Since arbitration is introduced, there is a need to guarantee starvation-free scheduling for all NIs. To that end the GAU uses the concept of ‘‘age’’. Each request register is extended to also hold the request age. The age is initialized to 0 when a request is loaded into to the GAU and increases every cycle. The arbiter corresponding with the link logic selects the oldest request available on its inputs. The propagated data during the propagation phase includes the aggregated usage, destination, packet length and request age. This scheme prevents starvation, since from all intersecting requests at least the oldest one is scheduled (note that younger requests on non-intersecting paths are scheduled

concurrently). Those requests that lose the arbitration are kept in the request register and their age increases by one, which improves their chance to win in the next GAU cycle. NI's never post more than 2 requests. Once they do post 2 requests, they wait until the grant line is asserted to indicate that the first one is granted. Introducing age based arbitration on every hop of the request propagation is greedy and might be sub-optimal. Improving this algorithm, for example, by introducing a second arbitration iteration, is considered for future work. The information about the winning request (and the input it arrived on) is used in the allocation phase to select the appropriate input to be updated with the grant window.

## 4. EVALUATION

In this section, we evaluate GANA performance in terms of area, power, latency (including buffering time in the NI) and throughput. We compare GANA to a ''baseline NoC'' which is an XY routed mesh with wormhole switching using 2 virtual channels (VC) and 4 flit-buffers per VC. The cost comparison is based on a detailed GAU model implemented in Verilog synthesized and tested for correctness. For the performance evaluation, a simulator for both the baseline NoC and GANA was built under the framework of OMNET++ simulation environment [19]. The simulated baseline-NoC and GANA use a 500MHz clock and 32bits flit (resulting in 2GB/s link data rate). The evaluated cases are the following: a standard set of fixed communication patterns, a case showing fairness advantages of GANA and a case showing GANA ability to avoid blocking due to VC shortages when congestion is introduced by hot-modules.

### 4.1 Cost Comparison

Two implementations, a baseline NoC and GANA, are compared. The NoC topology is an 8x8 mesh with inter-router distances of 2000um, a flit size of 32 bits, running at 750MHz on TSMC 45nm process. The baseline NoC was evaluated by running using ORION2 [7] with 2 VCs per link and 4 flits buffer per VC. ORION2 is a program that reports the area and power of NoC elements utilizing macro models adjusted to best-fit detailed NoC implementations, and can be configured for specific NoC architectures and specific process technologies. Although the NoC research community keeps improving NoC implementations providing area savings of up to 37% [9], most of these publications do not provide accurate area or process information to enable a detailed comparison. Therefore, it is convenient to use the ORION models as a reference baseline. The GAU implementation supports F=64 cycles look-ahead and a maximal packet length of M=31 flits.

In addition to the GAU, the cost of GANA is composed of the following elements:

NR: as in the baseline NoC there are $\mathbf{N}^2$ such routers. However, their per-port area only includes the area taken by the XY routing logic and single flit storage implemented using flip-flops.

NoC Routing: 32 wires (a flit width) in each direction. GANA does not require flow-control wires or logic. So the area and power are limited to the data wires and buffers. The total area is proportional to $\mathbf{N}^2$ as in a baseline NoC. In this comparison we ignored the GANA saving of backpressure lines.

Transmission and Grant requests: there are $\mathbf{N}^2$ lines connecting all NIs to the GAU. Each transmission request line carries the destination address of $A = 2\lceil \log_2(N) \rceil$ bits and message length of $B = \lceil log_2(M) \rceil$ bits when **M** is the maximal packet length in flits. The grant line carries a single bit back to the NI that times the transmission of the next flit. The NI to GAU connection width is **A**+**B**+1 (which is 15 bits for meshes with $\mathbf{N} \leq 15$). The request and grant lines bridge distances as far as **N** routers away from the GAU require buffering and are sampled by latches or flip-flops at most **N** times along their way. Assuming a methodology of Over-the-Cell routing (OTC), which is commonly used in processor design, the quoted routing area is of the repeater cells and flip-flops used along the wires.

Table 1 shows the area and power of a baseline NoC in comparison with GANA implementation. The different contributions for the cost in area and power are listed with their occurrence multiplier. The bottom line shows that for **N**=8, GANA takes only 16% (i.e., 84% saving) of the baseline NoC area and consumes 38% and 24% less power for the 25% and 75% load cases. Note that although the Req/Gnt lines travel the same per-hop distance as the data lines, their power is lower due to their lower activity. Even under100% load their activity will be 1/avg-packet-length which is assumed to be 4-flits. For the data lines only 75% of the flip-flop power is assumed to be proportional to the offered load the rest is leakage power.

Table I. Area and power comparison of baseline to GANA

| 8x8 Mesh NoC | Area [um^2] | | Power @ 25% load [W] | | Power @ 75% load [W] | |
|---|---|---|---|---|---|---|
| Parameter | Base Line | GANA | Base Line | GANA | Base Line | GANA |
| Router | 76,811 | 873 | **3.16E-02** | **2.02E-04** | **4.49E-02** | **3.74E-04** |
| Num Routers | 64 | 64 | 64.000 | 64.000 | 64.000 | 64.000 |
| Total Routers | 4,915,872 | 55,848 | **2.020** | **0.013** | **2.875** | **0.024** |
| Link | 621 | 621 | 0.013 | 0.013 | 0.027 | 0.027 |
| Num Links | 224 | 224 | 224.000 | 224.000 | 224.000 | 224.000 |
| Total Links | 139,059 | 139,059 | 2.901 | 2.901 | 6.118 | 6.118 |
| GAU | | 580,788 | | **0.040** | | **0.080** |
| Single Req/Gnt | | 615 | | 0.002 | | 0.010 |
| Num Reqs | | 64 | | 64.000 | | 64.000 |
| Total Req/Gnt | | 39,328 | | 0.097 | | 0.615 |
| Total | **5,054,931** | **815,024** | **4.92** | **3.05** | **8.99** | **6.84** |

Another indication for the cost of GANA compared to baseline NoC is to count the flip-flops required by each architecture. For GANA each router contains an output buffer per direction: $4F(N^2-N)$. GAU holds utilization register of the size of the look-ahead window per link: $4L(N^2-N)$ and FFR and Request and Grant registers per NI: $(2L+A+B+1)N^2$. The flip-flops used to store the arbitration results are negligible. In total GANA requires:

$$FF_{GANA} = (4F+6L+A+B+1)N^2 - 4N(L+F)$$

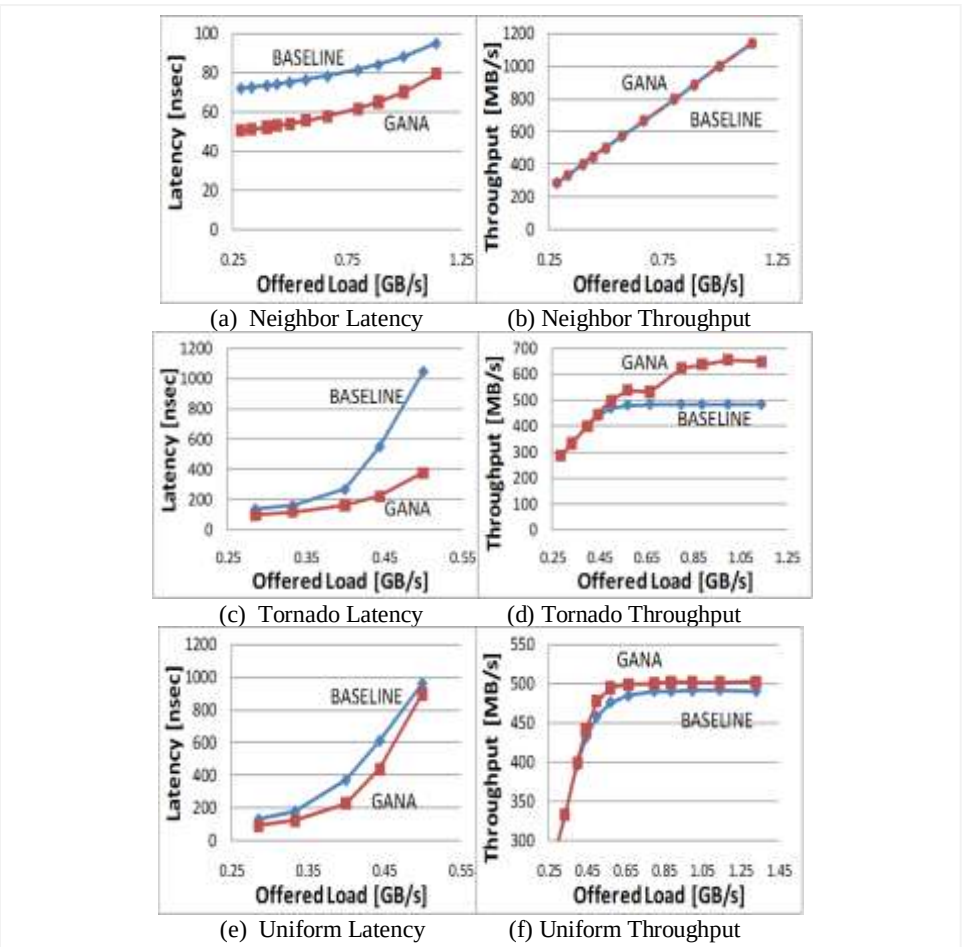
The baseline NoC has all its flip-flops within the routers. It must use output flip-flops like the GANA routers: $4F(N^2-N)$ but it also stores D flits on each input port for each VC (we denote number of VCs by V). So buffering uses a total of $5DFV(N^2-N)$. We assume the number of flip-flops of the arbitration logic are negligible and obtain the total number of baseline router flip-flops:

$$FF_{BL} = F(4+5DV)(N^2-N)$$

For our comparison we used N=8, F=32, L=64, A+B+1=15, D=4, V=2 and obtain: $FF_{GANA} = 30656$ and $FF_{BL} = 78848$.

### 4.2 Synthetic Traffic Patterns Benchmark

To obtain a performance benchmark of the baseline NoC versus GANA we follow the same approach used by [15][18][8] and apply standard communication patterns over an 8x8 size NoC: Tornado, Neighbor, Uniform-Random, Bit-complement, Bit-transpose, Bit-rotate and Bit-shuffle. For the Neighbor, Tornado and Uniform-Random patterns the average latency and average throughput versus the offered load are presented in Figure 9. The Neighbor pattern which avoids link contention shows no difference between the baseline and GANA. It also shows that both can support full wire speeds without contention. For the Tornado communication pattern which creates many hot spots, GANA and the baseline NoC have roughly the same latency but GANA provides around 30% more throughput for loads above 0.85GB/s (42.5% utilization). In contrast to the previous fixed permutations, the Uniform-Random destinations change randomly. Figure 9(e) and (f) show the GANA and the baseline NoC latency and throughput versus offered load for traffic that changes the destination every 16 packets.



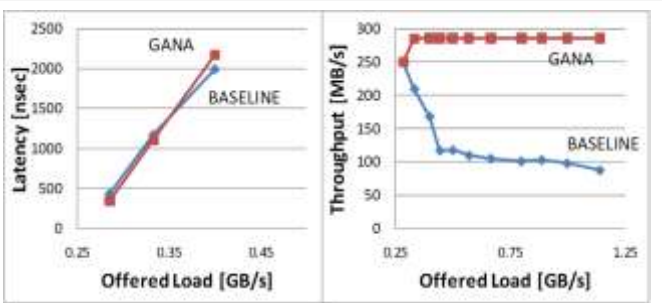

(a) Neighbor Latency (b) Neighbor Throughput

(c) Tornado Latency (d) Tornado Throughput

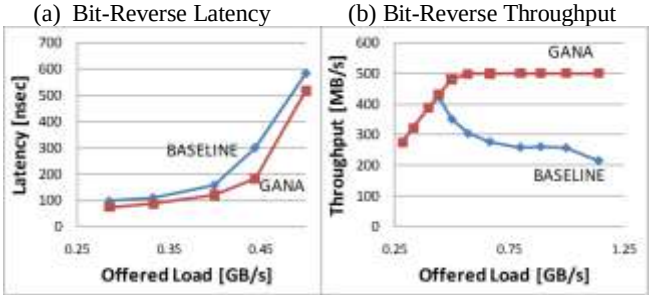
(e) Uniform Latency (f) Uniform Throughput

Fig. 9. Synthetic Traffic Patterns Average Latency and Throughput versus Offered-Load

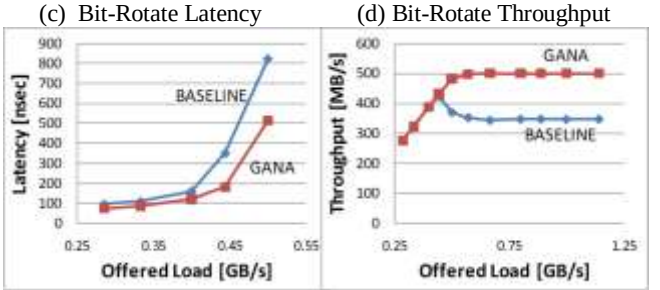
For the Bit-complement, Bit-transpose, Bit-rotate and Bit-shuffle communication patterns, Figure 10 shows the average latency and minimal output bandwidth from all the NIs. As can be seen, the GANA latency is better or equal to the baseline NoC latency. The minimal throughput of the baseline NoC is much lower than that of GANA. In two of the cases, the baseline NoC throughput is less than half of the GANA throughput. All the plots show that as the offered load increases from 0, both architectures have the same linear increase in throughput. However, once some network links reach their maximal capacity, the baseline NoC tends to favor certain flows while others are discriminated. For permutation traffic, this means that the baseline NoC minimal bandwidth out of the network is lowered, as shown in the graphs.




(a) Bit-Reverse Latency (b) Bit-Reverse Throughput



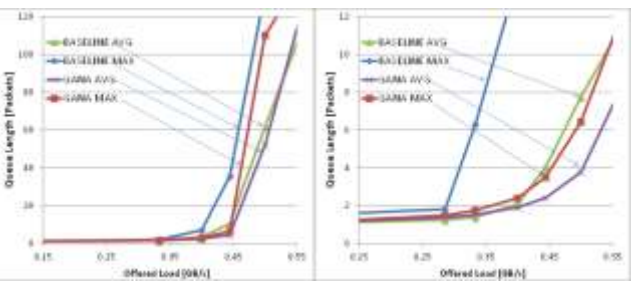

(c) Bit-Rotate Latency (d) Bit-Rotate Throughput




(e) Bit-Shuffle Latency (f) Bit- Shuffle Throughput

Fig. 10 Synthetic Traffic Patterns Average Latency and MIN Throughput versus Offered-Load

As GANA requires the NI's to hold packets until they are granted. Another aspect worth examining is the amount of NI buffering required by GANA compared to the baseline NoC. To that end we track the number of queued packets in each NI. This number is sampled whenever a new packet is generated. The average and maximal packet queue depth for the uniform and tornado traffic patterns are presented in Figure 11. It can be observed that GANA requires less NI buffers than the baseline NoC. For the uniform traffic in Figure 11(a) GANA is close to the baseline NoC mainly since the random destination evens out the network load. For the tornado traffic pattern in Figure 11(b) network bottlenecks cause unfair throughput in some flows of the baseline NoC. This in-turn cause backpressure on some NI's that increase the number of packets queued




(a) (b)

Fig. 11. NI Output Queue Depth in packets (a) for uniform and (b) for tornado traffic. The AVG and MAX lines show queue length or maximal queue length through the simulation time averaged over all NI's and 10 different simulations.

Another aspect to observe is the distribution of packet latency. Figure 12 presents the packet latency (from generation to ejection) for uniform traffic at 5% load. GANA provides a narrow distribution with peak around 30-40 cycles while the baseline NoC provides a wider distribution with peaks from 50 to 100 cycles. This may be attributed to GANA cut-through latency versus the baseline NoC where different sources have to pass a different number of queues and the latency distribution exhibit multiple peaks.

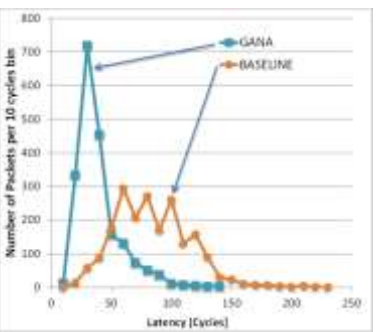



Fig. 12. Distribution of end to end packet latencies from all sources into a single sink for GANA and baseline NoC, under uniform destination traffic at a light network load.

### 4.3 Arbitration Fairness

The GAU age based arbitration improves fairness as reported by [10]. To demonstrate the fairness provided by GANA, a case of 4 flows that pass through a single link is presented in Figure 13 (a). Arbitration at the baseline NoC, which is based on round robin among the input ports of the router, yields an unfair throughput in what is known as the ‘‘parking lot’’ problem [3]. In this case, the three flows from NIs 0, 1 and 2 share the same input port of NR 3 and compete against the single flow from NI 3. Accordingly, the link bandwidth of 2GB/s is split between the flow from NI 3 which receives 1GB/s and the three flows which receive together the other half. Similarly, at NR 2, the flow from NI 2 receives 500MB/s and the other two flows together receive the other half. At NR 1, each of these flows receives 250MB/s. GANA that arbitrates based on request age, provides an equal bandwidth share to each flow. Figure 13 (b) shows the simulated throughput for the baseline NoC and GANA. The results match exactly the expected behavior.

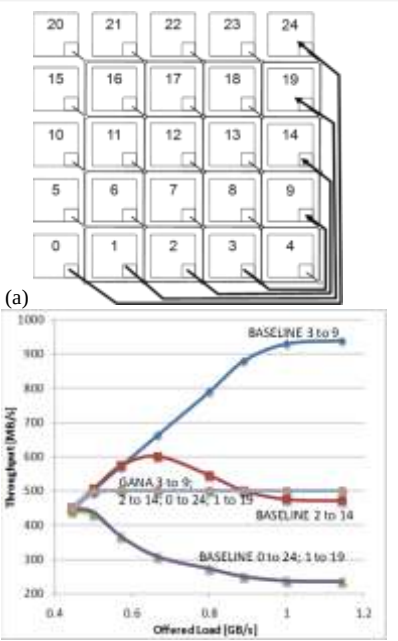



Fig. 13 A Case of 4 Flows passing through one link showing the ‘‘Parking Lot Effect’’. The Baseline NoC discriminates long flows while GANA provides equal share to all flows.

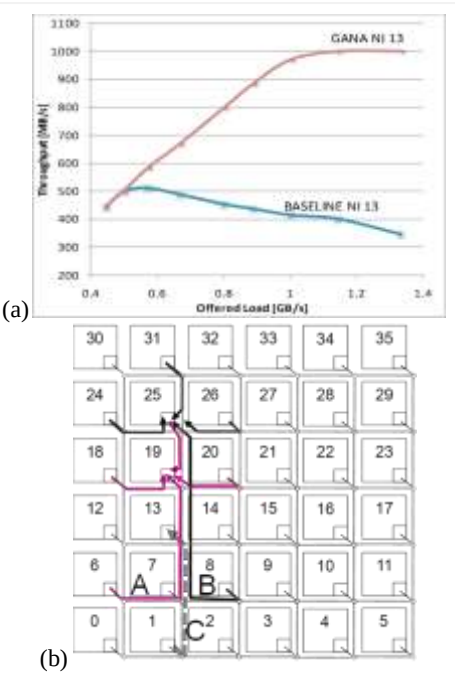



Fig. 14 A Case of VC Congestion Spreading On a XY routed 6x6 mesh: NI 19 receives traffic from NIs 6, 18, 20 and 25; NI 25 receives traffic from NIs: 8, 24, 26 and 31. So hot-modules 19 and 25 are 1:4 congested. The flow ***C*** (NI 1 to 13) is sharing a link with flows ***A*** (6 to 19) and ***B*** (8 to 25). On a baseline NoC this causes VC starvation for the non-congested flow ***C***. On GANA ***C*** reaches the expected ½ throughput (while ***A*** and ***B*** get ¼ each).

### 4.4 Handling Congestion

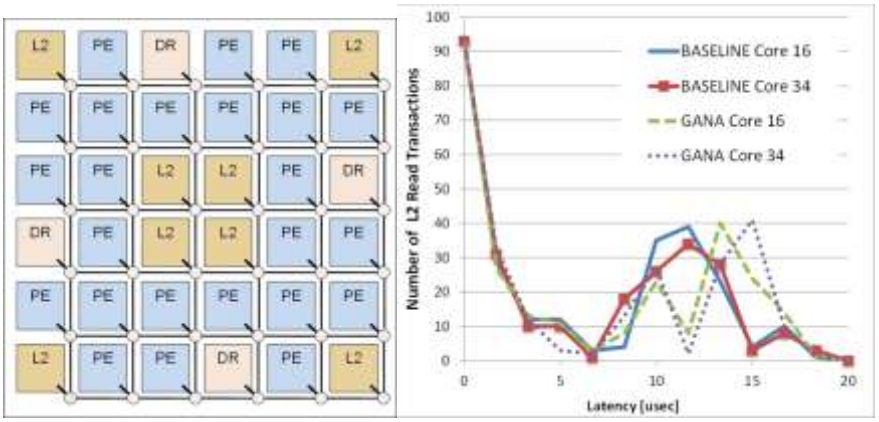
Hot-Modules are those receiving traffic from multiple sources, exceeding their input capacity. CMPs, where the NoC is heterogeneous, are likely to contain hot-modules. For example, if few DRAM controllers are shared by several L2 caches. When the congestion rooted at the input of these modules spreads toward the sources, then VCs that were allocated to congested flows are not released, resulting in the stalling of other non-congested flows [11][17].

The example traffic presented in Figure 14(a) has two hot modules: NI 19 and NI 25 (each driven by 4 flows). A victim flow named "**C**" (from NI 1 to NI 13) is sharing one link with other flows "**A**" (NI 6 to NI 19) and "**B**" (NI 8 to NI 25) driving these two hot spots. Since our NoC has 2 VCs, the resulting effect is that the victim flow "**C**" is stalled due to a lack of VCs at the +Y output of NR 7. The resulting throughput at NI 13 for GANA and the baseline NoC are shown in Figure 14(b). As expected for the baseline NoC the worst throughput is over NI 13 with ~300MB/s at an offered load of 1400MHz which is far less than the 1/3rd of the shared link (NR 7 +Y). It is closer to 1/8th of the link bandwidth. This is expected due to the sharing of a VC with traffic destined to a hot spot with a degree of 4. GANA, however, is able to provide the saturation ¼ of link bandwidth to flows "**A**" and "**B**" and the rest ½ to flow "**C**".

### 4.5 CMP Application Simulation

The simulation models presented above were extended to provide functionality of a CMP with three types of cores: Processing Element (PE) with L1 cache, L2 cache and DRAM interface. These modules were tiled to form a 6x6 cores CMP presented in Figure 15(a). L2 access traces of the Blackscholes benchmark of the PARSEC suite were applied to this model. Two parallel instances of the program were run – each on 12 PE cores (on the upper and lower halves of the CMP) assuming 64KB L1 and 4MB L2 caches, both with 64 bytes cache lines and associativity of 64 lines per set. All NoC packets are of 16 flits of 4 bytes length. The simulation results show runtimes of 2.7msec and 2.6msec for the GANA and the baseline NoCs respectively. The end-to-end packet latency histograms for the last cores to complete (PE 16 and 34) are presented in Figure 15(b). The histograms show that GANA and the baseline NoC have similar latency for the short L2 transactions that do not require DRAM access. However, GANA is slower by 23% then the baseline NoC in transactions that require DRAM to L2 page replacement. The fairness provided by GANA actually hurts these long transactions since every single packet out of the 64 packets making the cache page is considered "new" and thus the complete page replacement is slower than the baseline.

The optimization of GANA for CMP traffic and the simulation of the complete PARSEC benchmark as well as various placements of the L2 and DRAM cores are considered for future work.




(a) (b)

Fig. 15. (a) CMP Model containing 24 Processing Element Cores with L1, 8 L2 caches and 4 DRAM interfaces and the L2 transaction completion histograms for a baseline and GANA based NoCs. (b) L2 Read Latency histograms.

## 5. GLOBAL ARBITER NOC ARCHITECTURE SCALABILITY

GANA provides significant area, power, throughput and fairness benefits as shown in the previous section, however being based on a central scheduler that is known to be unable to scale to infinite number of cores. There are two scalability limiters: The first one is the logic complexity of the central arbiter which affects the number of cycles required for a single arbitration. The number of cycles required for arbitration limits the NoC throughput for short packets. The second limiter is the required wire density around the central unit. To quantify these limitations, we have conducted a detailed implementation of GANA using synthesis at TSMC 45nm technology for GANA with N in the range 4 to 10 (16 to 100 cores). From these experiments we conclude that GANA easily scales in current technology up to ~100 cores. With next generation technology nodes, it is expected that wire density will increase, enabling the support of several hundred cores. From latency perspective we show that GANA is scalable compared to mesh based NoC. The rest of this section describes in details the above physical implementation scalability aspects, provides analysis of the throughput and latency and show why the latency of GANA scales equally to the mesh NoC.

### 5.1 Physical Implementation Scalability

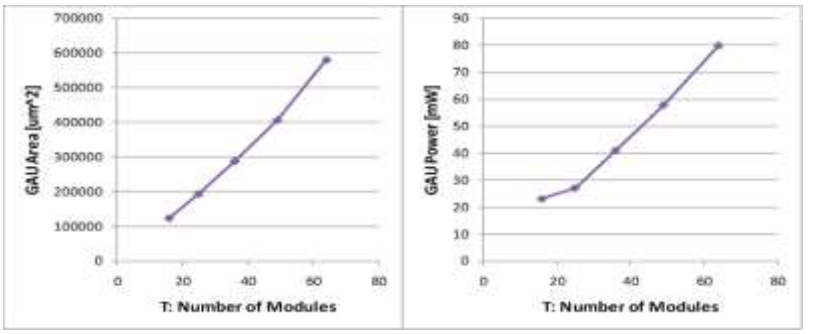
The proposed GAU was implemented in Verilog and synthesized using Synopsys Design-Compiler Topographical using 45nm TSMC technology without using Low-VT cells (which is a common tradeoff of power versus speed for this process. If required, low-VT cells can be used to speed-up the design significantly). To meet a clock cycle of 750MHz the different phases are performed in different clock cycles. The propagation phase involves paths that traverse 2N hops. Each hop involves multiplexers and routing comparators, N 2-input age sorters and N 4-input age sorters. To overcome the inherent delay of such a large combinatorial circuit, several clock cycles are allocated for the propagation phase (which results in the dependency of scheduling cycle time on N). The FFR for 64bit schedule-ahead and maximal packet length of 31 flits meets the timing for 1GHz in a single cycle. The total number of cycles S required for a single arbitration is 4 cycles for N=4, growing with N by a factor of $^{N}/_{2}$ cycles. These measurements provide an empirical result for 45nm technology:

$$S = N/2 \qquad (1)$$

With the expected minor speedup in the move to 28nm technology we can safely use (1) for larger N values.

For comparison, several NoC router microarchitecture papers discuss the critical path. In most of them, the critical path is within the virtual channel and switch allocator stages of the router. However, unlike GANA, multi-cycle arbitration is not a practical solution in such routers because any added arbitration latency must be multiplied by the number of hops.

Figure 16(a) shows the area of the synthesized GAU in [um2] and (b) the power in [mW] for **N** ranging from 4 to 8 (and **T** ranging from 16 to 64). It can be seen that the area and power are close to linear with the number of modules. Introduction of the 28nm technology is expected to scale area by 0.5 which should allow doubling the logic fit within the GAU without area increase.




(a) (b)

Fig. 16 Area and power of synthesized GAU for 750MHz TSMC 45nm for N=4,5,6,7,8 with Propagate stage of 2,2,3,3,4 cycles respectively

Routing porosity at the GAU interface is another potential scalability challenge to consider. It was shown that the total number of transmission request lines is $O(\mathbf{N}^2\log(\mathbf{N}))$, although hyper quadratic, for our implementation of NoC with $\mathbf{N} \leq 10$ on TSMC 45nm there is no issue to route 100 15bit busses into an area of 900,000um2, as it leads to a sustainable pin density of 0.32um on 5 metals. With future technologies providing dimensions scalability of 0.7, in accordance to Moore's law, the number of modules GANA can handle will grow. Future opportunities in sharing request lines or building hierarchical GAU's may also be needed if scalability to thousands of modules will be required.

The implementation provided in 45nm and the expected 28nm properties, serve as an indication that GANA is expected to scale to several hundred nodes in the future technology.

Another aspect of the tightly synchronized architecture is the need to use a synchronous clock. In large VLSI devices, there is actually no need to fully synchronize the clocks and a Globally Asynchronous Locally Synchronous (GALS) methodology based on mesochronous clocking is used. For such structures, clock phase can be predictable and well controlled via the usage of adaptive clock phase synchronizers [13]. Therefore, synchronization poses the same scalability issues to GANA and the common NoC.

### 5.2 Throughput and Latency Analysis

The simulation results for various traffic patterns demonstrate GANA's performance under complex traffic patterns. In this section, we provide closed-form analytical expressions under some simple traffic assumptions (such as zero load and average latency to a random destination). The purpose of such modeling is to gain insight into scalability

issues. The following notation and assumptions are used in the analysis: The topology is an N x N mesh with XY routing. Flits progress through the baseline network with L cycles latency per hop. The GAU takes S cycles for each arbitration loop. It can schedule as many as N2 requests in every loop as long as they do not intersect. It is also possible to pre-schedule up to F cycles in advance. The latency of the transmission request for reaching the GAU and for the grant to get back to the NI is at most N cycles.

Since the GAU arbitration takes **S** NoC cycles, it can maintain full wire speed only for packets with at least **S** flits. As the GAU is able to schedule all the outstanding requests in parallel as long as they do not intersect at a link, the same rule can maintain the bandwidth for all flows under the no contention condition.

Under very low utilization, the average path latency can be expressed by the average path length. The average path length is proportional to the average distance between all possible pairs:

$$\frac{1}{(N^4-N^2)}\sum_{r1=1}^{N}\sum_{c1=1}^{N}\sum_{r2=1}^{N}\sum_{c2=1}^{N}|r1-r2|+|c1-c2| \qquad (2)$$

That evaluates to:

$$\frac{2N}{3} \qquad (3)$$

The latency per hop of common NoCs **L** is in the range of 2 to 4. So the resulting average path latency in cycles is:

$$\alpha N \mid \frac{4}{3} \leq \alpha \leq \frac{8}{3} \qquad (4)$$

For GANA, the latency is the time it takes the transmission request to get to the GAU, plus the GAU scheduling latency, plus the time it takes the grant to reach back to the source, plus a single cycle per hop. The average distance of NI from the GAU is **N**/2-1. Consequently, the average GANA latency (assuming single clock cycle per hop) is:

$$S + 2\left(\frac{N}{2}-1\right)+\frac{2N}{3} = S-2+\frac{5}{3}N \qquad (5)$$

For the maximal path, the number of hops is 2N-2. For the common NoC of 2 to 4 cycle latency per hop this means latency of:

$$4N-4 < Common\ NoC\ maximal\ path\ latency < 8N-8 \qquad (6)$$

For GANA the same number hops will consume less latency on the data path (2N-2 cycles) but require the request to propagate to the GAU (N-1 cycles), S number of cycles for arbitration and then the response will need to propagate back to the sender (N-1 cycles):

$$GANA\ maximal\ path\ latency = 4N-4+S \qquad (7)$$

Figure 17 shows the plots of the average path and the maximal path zero-load latencies comparing the fast (2 cycles/hop) and slow (4 cycles/hop) baseline NoC with

GANA computation latency as provided by (1). It can be seen that both GANA and the baseline NoC show a linear dependency of the latency on **N** (square root of the number of modules). For the average path, GANA performs a little slower than the fastest baseline NoC. For the longest path GANA is 7.5% slower than the fastest NoC. For both path lengths, a NoC that does not incorporate advanced opportunistic arbitration methods [16][6] will have a higher latency than GANA.

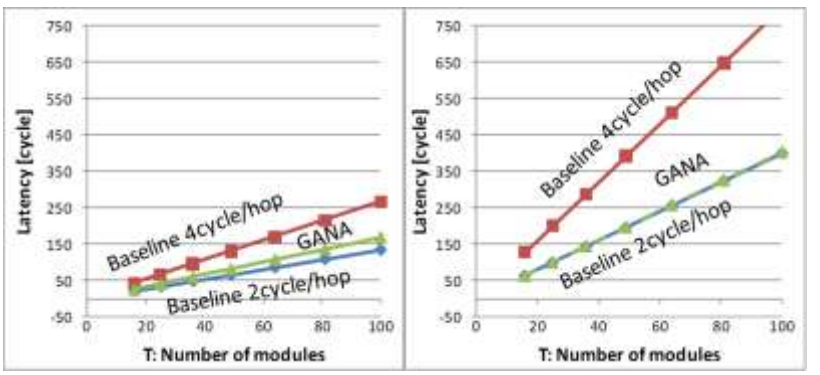



Fig. 17 Comparing GANA and NoC zero load latency for NoCs with 2 to 4 cycles latency per hop. a) Average Path Length; b) Maximal Path Length

## 5.3 Performance Scalability

In the previous sections we showed that GANA was able to keep the full speed line rate for packets that are longer than **S –** the number of NoC cycles for a single GAU scheduling computation cycle. From (1), **S** is proportional to **N**/2. For example, a full line rate can be maintained for a 10 x 10 NoC for packets of at least 5 flits. Under empty network conditions, the maximal throughput for packets of length $l$ < **S** is $l/S$ of the link bandwidth. In terms of scalability with the total number of modules **T**= $\mathbf{N}^2$, the minimal packet length to reach full wire speed **S** is proportional to

$$\sqrt{T}/2. \qquad (8)$$

Latency scalability under empty network condition can be deduced from the equations presented in the previous section. From (1), (4) and (5) the ratio of the GANA average path latency to 4-cycles/hop NoC latency is:

$$\frac{N/2-2+\frac{5}{3}N}{8/3N} = \frac{13}{16} - \frac{3}{4N} \qquad (9)$$

As N increases, the empty network average latency of GANA is close to 3/4$^{th}$ of the NoC. Similarly, the longest path empty network latency of GANA as compared to the NoC is obtained from (1), (6) and (7):

$$\frac{4N-4+N/2}{8N-8} = \frac{N+8(N-1)}{16(N-1)} = \frac{1}{2} + \frac{1}{16}\frac{N}{N-1} \qquad (10)$$

Consequently, the ratio of the GANA longest path latency to that of a 4 cycles per hop NoC is close to ½ as N is increased. For the fastest NoC with 2 cycles per hop GANA's latency is higher than the NoC latency.

## 6. CONCLUSION

A new Global Arbitration NoC Architecture has been shown to be viable for 45nm technology with 10x10 NoCs and for further scale with the technology to several hundred cores. GANA, which relies on the synchronous clocking supported by the monolithic nature of NoCs, provides a lower cost solution as compared with a standard NoC. This new architecture completely removes buffers and arbiters from the network routers, saving a considerable amount of area and power. For an 8x8, GANA, 84% of the area and 24% the power are saved compared with a standard buffered NoC. We demonstrated additional benefits of the global arbitration, such as a better fairness and the ability to avoid blocking due to VC starvation imposed by hot-modules. Our evaluation shows that GANA is superior to a baseline NoC in terms of throughput and equal in latency for all tested communication patterns, including a standard set of traffic permutations and random flows.

## ACKNOWLEDGMENTS

## REFERENCES


[1] BENINI L. AND MICHELI G.D. 2006, Networks on chips: technology and tools, Academic Press.

[2] CHEN Y., QIAO C., AND YU X. 2004, Optical Burst Switching (OBS): A New Area in Optical Networking Research.

[3] DALLY W.J. AND TOWLES B. 2004, Principles and practices of interconnection networks, Morgan Kaufmann.

[4] GOOSSENS K., DIELISSEN J., AND RADULESCU A. 2005, AEthereal Network on Chip:Concepts, Architectures, and Implementations, IEEE Design and Test of Computers, vol. 22, pp. 414-421.

[5] HANSSON A., SUBBURAMAN M. AND GOOSSENS K. 2009, AElite: A Flit-Synchronous Network on Chip with Composable and Predictable Services.

[6] IZU C., BEIVIDE R., JESSHOPE C. AND ARRUABARRENA A. 1993, Experimental evaluation of Mad Postman bidimensional routing networks, Microprocessing and Microprogramming, vol. 38, pp. 33-41.

[7] KAHNG A., LI B., PEH L. AND SAMADI K. 2009, ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration, Design, Automation & Test in Europe Conference & Exhibition, DATE '09., pp. 423-428.

[8] KIM J., PARK D., THEOCHARIDES T., VIJAYKRISHNAN N. AND DAS C.R. 2005, A Low Latency Router Supporting Adaptivity for On-Chip Interconnects," DAC '05: Proceeding of the 42nd annual conference on Design Automation, pp. 559-564.

[9] KIM J., 2009, Low Cost Router Microarchitecture for On-Chip Networks, MICRO'09

[10] LEE M. M., KIM J., ABTS D., MARTY M., and LEE J. W., Approximating age-based arbitration in on-chip networks, in PACT '10: Proceedings of the 19th international conference on Parallel architectures and compilation techniques, New York, NY, USA, 2010.

[11] MANEVICH R., CIDON I., KOLODNY A. AND WALTER I. 2010, Centralized Adaptive Routing for NoCs, IEEE Computer Architecture Letters.

[12] MANEVICH R., WALTER I., CIDON I. AND KOLODNY A. 2009, Best of both worlds: A bus enhanced NoC (BENoC), 3rd ACM/IEEE International Symposium on Networks-on-Chip, La Jolla,

CA, USA, pp. 173-182.
[13] MANGANO D., LOCATELLI R., SCANDURRA A., PISTRITTO C., COPPOLA M., FANUCCI L., VITULLO F. AND ZANDRI D. 2006, Skew Insensitive Physical Links for Network on Chip, 1st International Conference on Nano-Networks and Workshops, Lausanne, Switzerland, pp. 1-5.
[14] MICHELOGIANNAKIS G., SANCHEZ D., DALLY W.J. AND KOZYRAKIS C. 2010, Evaluating Bufferless Flow Control for On-chip Networks, Networks-on-Chip, International Symposium on, Los Alamitos, CA, USA: IEEE Computer Society, pp. 9-16.
[15] MOSCIBRODA T. AND MUTLU O. 2009, A Case for Bufferless Routing in On-Chip Networks, ISCA'09.
[16] MULLINS R., WEST A. AND MOORE S. 2004, Low-Latency Virtual-Channel Routers for On-Chip Networks, ACM SIGARCH Computer Architecture News, vol. 32, p. 188.
[17] NYCHIS G., FALLIN C., MOSCIBRODA T. AND MUTLU O. 2010, Next Generation On-Chip Networks: What Kind of Congestion Control Do We Need?, 9th ACM Workshop on Hot Topics in Networks, Montterey CA.
[18] SOTERIOU V., RAMANUJAM R.S., LIN B. AND PEH L. 2009, A High-Throughput Distributed Shared-Buffer NoC Router, IEEE Computer Architecture Letters, vol. 8, pp. 21-24.
[19] VARGA A., OMNET++ URL reference: http://www.omnetpp.org/
[20] WALTER I., CIDON I., GINOSAR R. AND KOLODNY A. 2007, Access Regulation to Hot-Modules in Wormhole NoCs, First International Symposium on Networks-on-Chip (NOCS'07), Princeton, NJ, USA, pp. 137-148.