A Fast, Source-synchronous Ring-based Network-on-Chip Design

Ayan Mandal*, Sunil P. Khatri†, Rabi N. Mahapatra*
* Department of CSE, Texas A&M University, College Station TX 77843
† Department of ECE, Texas A&M University, College Station TX 77843
{ayan1731, sunilkhatri}@tamu.edu, rabi@cse.tamu.edu

***Abstract*— Most network-on-chip (NoC) architectures are based on a mesh-based interconnection structure. In this paper, we present a new NoC architecture, which relies on source synchronous data transfer over a ring. The source synchronous ring data is clocked by a resonant clock, which operates significantly faster than individual processors that are served by the ring. This allows us to significantly improve the cross section bandwidth and the latency of the NoC. We have validated the design using a 22nm predictive process. Compared to the state-of-the-art mesh based NoC, our scheme achieves a $4.5\times$ better bandwidth, $7.4\times$ better contention free latency with 11% lower area and 35% lower power.**

I. Introduction

Multi-core processors require an efficient and scalable NoC infrastructure to handle the inter-processor on-chip communication needs. There has been significant research in NoC topologies [1], [2], [3], global wire management [4], and power optimization [5]. In terms of topology, a 2D mesh interconnection network has received greatest attention by NoC designers due to its simpler implementation, high bandwidth and overall scalability. However the large diameter of the mesh has a negative effect on communication latency. Other popular topologies include Ring [6], Fat Tree, 2D Flattened Butterfly [7], Octagon [8] and Torus [9].

As the size of a chip multi-processor (CMP) grows, it becomes increasingly difficult to distribute a synchronous clock over the entire chip. This has led to the use of asynchronous communication between regions on the die, where each region is clocked in a synchronous manner. This is referred to as a Globally Asynchronous Locally Synchronous (GALS) [10] communication approach.

Recently, resonant clocking techniques [11], [12], [13], [14] have been demonstrated, resulting in ultra high-speed, low power, stable on-chip clock generators. In this paper, we utilize such a clock to develop a high-speed, source-synchronous NoC architecture, as outlined next.

In this paper, the CMP performs communication using a GALS paradigm. Each Processing Element (PE) operates in a synchronous manner, and is assumed to operate at 2 GHz. The NoC is comprised of a series of (horizontally and vertically arranged) flattened rings. A *Junction Station* (JS) is placed wherever these rings intersect, allowing data to switch between the horizontal ring and the vertical ring at the junction. Each ring operates significantly faster (about $9\times$ faster, from our HSPICE [15] simulations) than the PEs. The data on the ring is transmitted in a source synchronous manner with reference to a fast resonant clock. Each PE is connected to a ring by means of a *Add-Drop Station* (ADS), which allows it to remove/inject data from/into the ring. Each ADS contains two asynchronous FIFOs. One FIFO is written from the ring NoC, and read by the PE. The other FIFO is written by the PE and read by the ring NoC driver logic.

The block diagram of a single (flattened) ring of our ring based NoC is shown in Figure 1. Note that since this figure depicts a single ring, JS' are not shown. The ring carries three fields of information – d bit *data*, k bit *address*, and 1 *valid* bit. If there are P PEs in the CMP, then $k = log_2 P$. A high speed resonant clock signal *Rclk* runs parallel to the ring signals mentioned above, as shown in Figure 1. The data, address and valid signals are source synchronous with the *Rclk* signal.

The key contributions of this paper are:

- To the best of the authors' knowledge, this paper is the first to demonstrate a fast ring based source-synchronous NoC architecture using a resonant clock. We utilize a plurality of horizontal and vertical rings of this kind.
- In order to validate the idea, we have conducted circuit simulation using 22nm PTM process. The ADS's, JS's and FIFOs were simulated in HSPICE, to validate correct operation and routing performance at 18 GHz.
- Our HSPICE results show that the ring based NoC, when compared with the state-of-the-art mesh based NoC, delivers a $4.5\times$ better bandwidth, $7.4\times$ better contention free latency with 11% lower area and 35% lower power.
- Since each source-synchronous ring is mesochronous with other rings, and asynchronous with the PEs, precise synchronization of the rings with PLLs is not required, making the technique very practical to implement.

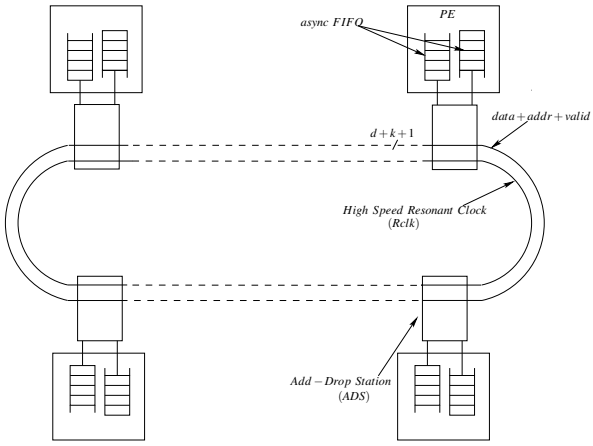


Fig. 1. Single Ring based NoC Architecture

The rest of the paper is organized as follows. Section II describes previous approaches in this area. Section III presents our approach, while Section IV describes the results of experiments which we performed to validate our approach. In Section V, we draw conclusions and discuss avenues for future work.

II. Previous Work

Any NoC architecture should have a combination of the following desirable features: 1) scalability and modularity 2) low interconnect latency 3) minimal power 4) high link data-rates. There have been several NoC topologies that have been developed to satisfy these requirements.

The simplest and most ubiquitous NoC topology is the 2D mesh. Dally and Towles [16] first proposed a 2D mesh as a NoC architecture. The topology consists of a 2D mesh of wires, with switches at the intersections of horizontal and vertical wires. Every switch has five ports, one connected to the local resource (PE) and the others connected to the closest neighboring switches. The torus architecture was proposed in [9], with switches at the edges connected to the switches at the opposite edge through wrap-around channels. The long end-around connections can yield excessive delays which can

978-3-9810801-8-6/DATE12/©2012 EDAA

be avoided by folding the torus [17]. The inherent disadvantage of the mesh or torus topologies is their large communication radius, resulting in large amounts of interconnect and large numbers of arbiters at the N-S-E-W crossings. These in turn leads to a large power consumption. Karim et. al. [8] proposed the Octagon architecture. The Octagon network consists of a basic *octagon unit* having eight nodes and 12 bidirectional links. It has a simpler implementation compared to the 2D mesh, with a higher throughput. Unlike the crossbar, the Octagon's implementation complexity increases linearly with the number of nodes. A 2D Flattened Butterfly was proposed in [7]. The 2D flattened butterfly is derived by flattening the routers in each row of a conventional butterfly topology and hence provides the connectivity of a mesh with additional links. The fat tree [18] connects routers in a tree manner, with sources and destinations at the leaves. The major advantage of the fat tree is the large amount of bandwidth available, with the downside of the requirement for large-radix routers toward the top of the tree. In [19], the authors propose an asynchronous 2D mesh NoC infrastructure. Compared to a synchronous mesh, their realization increases the area utilization by more than 3×, with a 30-50% gain in speed and a 5× reduction in power and a 6.9× energy improvement. The Ring [6] topology implements concentric connected rings (similar to ring road in city), which helps to reduce the risk of congestion in the central parts of the network. The approach is motivated by the smooth flow of traffic in ring roads. Their approach is fundamentally different than ours in its topology, as well as in the fact that their simulations are conducted purely at the architectural level.

In [20], the authors have implemented a power-efficient mesochronous NoC with no area and latency overhead. Although our NoC design is also mesochronous, we operate at a significantly higher speed. Additionally [20] use a traditional mesh topology in contrast with our source-synchronous ring.

In all the above implementations, design decisions are made based on the fact that the interconnection network runs at the same or lower frequency as the PEs. In contrast, our focus is a NoC architecture which runs significantly faster (9× in our simulation) than the PEs. This allows more architectural flexibility compared to existing NoC solutions. For example, the significantly higher bisection bandwidth allows the designer to implement our NoC architecture with narrower links (yielding a lower area and power for the same bisection bandwidth). The significantly lower latencies allow our approach to scale more elegantly for larger CMPs. In this paper, our focus is not on architectural aspects of the ring-based NoC. As a result we have not simulated any real workloads in this paper, but evaluate performance in terms of total available contention free bandwidth. Instead, we devote our attention to the circuit aspects, showing the validity of the approach by means of thorough circuit simulations.

In 2010, Sanchez et. al. compared various network topologies of interconnection networks in terms of latency, throughput, and energy dissipation [21]. The authors report that for a 64-core CMP, the total area utilization is lowest in case of the mesh topology. The flattened butterfly was shown to consume the largest area (by a factor of $\sim 3\times$). In terms of power consumption, the flattened butterfly (the topology with the largest occupied area) consumes only slightly more power than the mesh due to the higher leakage of the extra links. On the other hand, the fat tree consumes the most power because of the large number of high-radix router hops and link stages that a flit traverses, on average. Based on these observations, it was concluded that the 2D mesh is best NoC topology overall. As a consequence, the results of our paper are compared with the 2D mesh results shown in [21]. Other mesh based NoC studies are reported in [22], [23].

In this paper we present a fast, low-latency source-synchronous ring-based NoC architecture. Inter-processor communication is achieved with minimal latency with the use of extremely fast, overlapping source-synchronous data rings, which traverse the CMP in both the horizontal and vertical directions. Since our approach is GALS, the interconnection network operates on a different clock domain than the PEs. The router complexity as well as the link lengths determines the frequency of operation of the network. We have simulated the routers and links in HSPICE [15], with link parasitics extracted from Raphael [24]. A 4×4 section of the NoC is simulated, to validate routing functionality, as well as to provide accurate delay, area and power numbers.

Resonant oscillators are a promising technique to generate a high-frequency on-chip clock signal with low power. Recently, a *traveling wave* resonant oscillator circuit (referred to by the authors as a *rotary clock*) was described and implemented [13], [14]. The scheme utilizes a sufficiently long wiring ring, with cross-coupled inverter pairs spread uniformly along the ring, and a mobius connection at one location of the ring. The capacitive and inductive parasitics of the ring result in a high frequency oscillatory network. The key drawback of the rotary clock is that the phase of the generated clock varies along the ring making traditional synchronous clock based design extremely difficult. In response to this, a *standing wave* resonant oscillator (SWO) circuit was proposed [11]. In this approach, a long wiring ring is used, and oscillations are sustained in this resonant ring by just using a single inverter pair. A mobius connection at the end of the ring ensures uniform phase at all the points along the ring, averting the main drawback of the rotary clock approach. In [25] a resonant SWO based PLL was implemented, with an inductance control based coarse frequency adjustment mechanism. Fine frequency adjustment is achieved by controlling the body bias of the PMOS transistor of the inverter pair. In [12] the authors present a tiled SWO based resonant grid for high frequency clock distribution. In the NoC scheme of this paper, we use the ideas proposed in the scheme of [12] to distribute the high frequency clock along with each ring of the source-synchronous ring-based NoC, while keeping the ring clocks synchronized.

III. Approach

A. Overview

A single ring of our ring-based NoC is shown in Figure 1. Each ring is flattened, and consists of k bits of address, d bits of data and 1 valid bit. These wires are driven source-synchronously, along with a high speed resonant standing-wave clock [11], [12], [13], [14], [12]. The clock operates at 18 GHz, and the transmission rate of the address, data and valid bits is also 18 GHz. The PEs of the CMP connect to the ring at discrete locations through an Add-Drop Station (ADS) which contains two FIFOs as shown in Figure 1. PEs are assumed to operate at 2 GHz. Note that the ring-based NoC operates at a significantly faster clock speed than the cores (or PEs). Because of the extreme high speed of the ring, the latency and transfer characteristics of our ring-based NoC are extremely good.

Figure 2 illustrates a section of the CMP, with 2 horizontal rings and 3 vertical rings. The vertical and horizontal rings are shaded with diagonal patterns, to distinguish them. Regions where the vertical and horizontal rings overlap are shaded using a pattern which consists of the overlay of the horizontal and vertical patterns. Each ADS (marked with a × symbol) services a single PE. Also, a Junction Station (JS) (marked with a ∘ symbol) is located at each location where the horizontal and vertical rings intersect. Since there are a total of 20 ADS's in Figure 2, therefore 20 PEs are serviced in the NoC fragment shown in this figure. Each PE is shown with a dotted outline.

Our source-synchronous rings are unidirectional. Without loss of generality, we assume that each ring, transmits data in a counter-clockwise manner. The distance between two adjacent ADS's or JS's in the ring is fixed.

In the remaining subsections, we discuss our processor modeling assumptions, followed by a discussion of the key components of our design (resonant clock, asynchronous FIFO design, and ADS and JS design,).

All plots, tables and figures in this paper are generated for a 22nm PTM [26] fabrication process. In our experiments, we first validated that the resonant SWO operates at 18 GHz.

A ring based NoC is not fault tolerant. This can be fixed by using bidirectional rings.

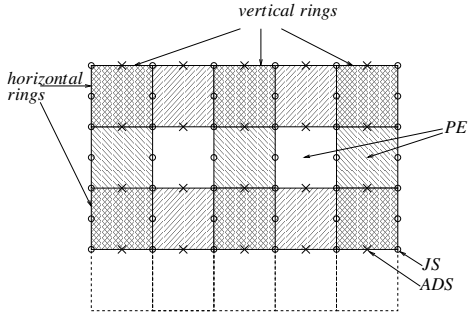


Fig. 2. Ring based NoC architecture

	Core Area (mm^2)	Core TDP (mW)
Niagra2	1.418	0.483
Atom	1.607	0.220

TABLE I
AREA AND POWER PROJECTIONS FOR THE CORES BASED ON THE SUN NIAGARA2 AND INTEL ATOM DESIGNS [21]

B. Processor Modeling Assumptions

The most important determinant of the speed of our source-synchronous ring-based NoC is the link length. A long link will force the ring-based NoC to operate slower. This section discusses our methodology to determine the length of each link. Based on Figure 2, we notice that each side of the PE corresponds to two links. This is because JS's and ADS's alternate along each horizontal section of any ring. Therefore, assuming a square PE, the length of each link is half the side of the PE.

In [21], the authors studied architectural implications of interconnect design for CMPs with up to 128 cores. The authors approximate the core area and power by scaling down two existing core designs: the Sun Niagara 2 [27] and the Intel Atom [28] for a 32nm process. Since our process is a 22nm process, we further scale their numbers. Table I shows the area and power for a 22nm implementation of the Niagara 2 and the Atom processors. Power numbers in this table are scaled from those of [21] by multiplying their numbers by the ratio of the square of the saturation I_{ds} at 22nm and 32nm respectively, from 22nm and 32nm PTM [26] processes.

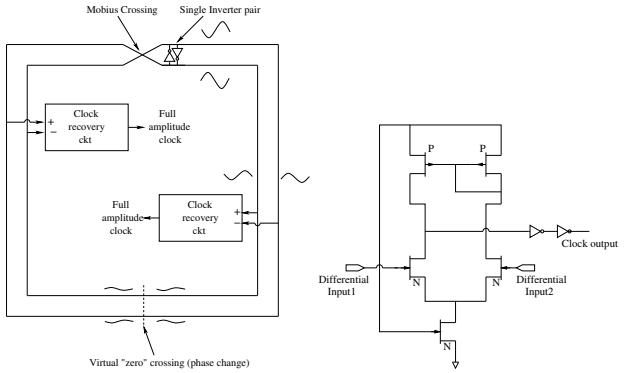
Just like [21], we take our PE area to be the average of the two areas shown in Figure 2. Assuming a square die, this means that each link is 615 μm long.

C. Resonant Clocking

The information transmitted in each ring is driven source-synchronously with a ring-based standing wave resonant oscillator (SWO). We implemented a variant of the SWO circuit as proposed in [11] . The authors utilize a long wiring ring and sustain oscillations in the resonant ring by just using a single inverter pair, as shown in Figure 3 a). By making a mobius connection at the end of the ring, it is ensured that the clock signal at any point on the ring has the same phase. By using differential amplifiers (Figure 3 b)), full rail clock signals are extracted at the locations desired. The total length of the ring is a half-wavelength ($\lambda/2$), where λ is the length over which a 360° change in phase is accomplished.

In our work, we have utilized the SWO concept of [11], and modified it so that the clock ring spans across several *PE*s. In our case, we have placed inverters at $\lambda/2$ locations in a ring whose total length is $p \cdot \lambda/2$ (where p is odd). This yields the same oscillations as for a $\lambda/2$ ring. In this way, we can implement rings of arbitrary length (in odd multiples of $\lambda/2$) while still sustaining the same high frequency of oscillation.

For the example in Figure 2, for the horizontal rings, p is 25 (since the ring length is 3 chip perimeters and p must be odd), and $\lambda/2$ is 615 μm (the link length). For the vertical rings, p is 17. Note that inverter pair locations are made to coincide with the ADS and JS locations, thereby providing the resonant clock at these locations. Note that since



(a) Standing-wave Resonant Clock (b) Clock Recovery Circuit

Fig. 3. Standing Wave Resonant Clocking Concept [11]

p is odd, and each ring requires $p-1$ clock connections to ADS/JS sites, and the signal from the last inverter pair is not utilized.

Note that it in order to ensure that the resonant structure bootstraps in a standing wave configuration, the signals at each inverter pair are initialized using a global bootstrap signal [12].

A final advantage of using several source-synchronous data rings is that since they have asynchronous FIFOs to transfer data into, out of, and across rings (through ADS's and JS's), the high-frequency clocks of the rings need not be synchronous. This makes the design of the ring clocks significantly simpler.

We do not include the ring clock power in the power numbers reported in our experiments. The reason for this is that the approach we compare our ring-based source-synchronous NoC with (the 2D mesh of [21]) does not include clocking power in their estimates. Further, resonant clocks are known to consume extremely low power [11], [12], [13], [14], since they are LC oscillators, consuming power only due to the losses in the wiring and inverter pair structures.

D. Asynchronous FIFO

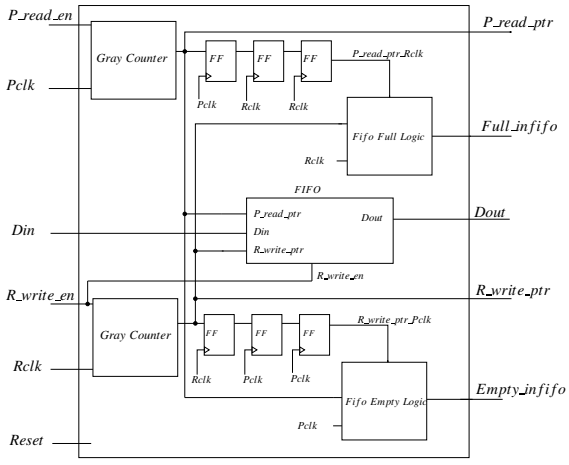


Fig. 4. Asynchronous FIFO (*Infifo*) Block Diagram

Asynchronous FIFOs are used to safely pass data from one clock domain to another clock domain. Our ADS implementation utilizes two asynchronous FIFOs (an *Infifo* and an *Outfifo*). We implemented the design of these FIFOs as suggested in [29]. In the following discussion, signals prefixed with a *R_* refer to ring signals, while signals with a *P_* prefix refer to PE signals. We describe the *Infifo* in this section. The *Outfifo* is identical, except that *R_* signals are replaced with *P_* signals and vice versa.

1) Block Diagram of Infifo*:* The block diagram of the *Infifo* is shown in Figure 4. The asynchronous FIFO read (*P_Read_en*) and write (*R_Write_en*) enable signals are clocked by independent read (*Pclk*) and write (*Rclk*) clocks. *Din* contains the input data to be

written, *Dout* contains the output data. The *Reset* signal is used to initialize the FIFO. Two status flags denote whether the FIFO is empty (*Empty_infifo*) of full (*Full_infifo*). Additionally, the read pointer *P_read_ptr* and the write pointer *R_write_ptr* are outputs of the FIFO. We next discuss how the Write/Read operations are done, followed by how the FIFO full/empty conditions are detected.

2) Infifo *Write and Read:* The write pointer (R_write_ptr) always points to the next word to be written and the read pointer (P_read_ptr) always points to the current FIFO word to be read. Figure 5 illustrates the core of the FIFO (the block for the FIFO core is also shown in Figure 4). Writing is done by converting R_write_ptr to a one-hot signal, and ANDing it with R_write_en. A single FIFO entry (the one that is selected) is therefore written. All other unselected entries are re-written into the FIFO.

For a read operation, the P_read_ptr appropriately selects a FIFO entry, which is driven out on the *Dout* output of the FIFO on *Pclk* unless the FIFO is empty.

3) Infifo *Full and Empty Detection:* A FIFO full condition occurs when the write pointer catches up to the synchronized and sampled read pointer ($Read\_ptr\_Wclk$).

Before checking the empty condition, the R_write_ptr is synchronized to the *Pclk* domain. The resulting value is compared with the P_read_ptr. Similarly, before checking the full condition, the P_read_ptr is synchronized to the *Rclk* domain. The resulting value is compared with the R_write_ptr. Synchronization is performed using a 2 flip-flop synchronizer [30] as shown in Figure 4. In order to achieve a higher MTBF, we can use a three or more flip-flop synchronizer. Another way of implementing the synchronizer is outlined in [31].

For a FIFO depth of n, we need $log(n)+1$ bits for the read and write pointers. One extra bit is required to distinguish between full and empty condition. When the write pointer increments past the maximum FIFO address, it will increment the unused MSB while setting the rest of the bits back to zero. The same is done with the read pointer. If the MSBs of the two pointers are different, it means that the write pointer has wrapped around one more time that the read pointer. If the MSBs of the two pointers are the same, it means that both pointers have wrapped the same number of times. This allows us to implement both the empty and the full condition and requires only XOR and XNOR gate for the logic.

Assuming there are n FIFO entries, the empty and full conditions are computed as:

$Empty\_infifo = R\_write\_ptr\_Pclk[3:0] == P\_read\_ptr[3:0]$ and $Full\_infifo = [!R\_write\_ptr[3], R\_write\_ptr[2:0]] == P\_read\_ptr\_Rclk[3:0]$

Gray code counters are used to keep track of the read and write pointers, as shown in Figure 4. Gray codes only allow one bit to change for each clock transition, eliminating the problem associated with trying to synchronize multiple changing signals on the same clock edge. The XNOR gate for the gray counters were implemented using pass-gates.

Dynamic flip-flops were used as these had to operate at a very high frequency. For the FIFOs of PE_i, *Rclk* and *Pclk* were taken to be 18 GHz and 2 GHz respectively

We have verified 100% correct functional as well as at-speed operation of the FIFOs.

E. Description of Add-Drop Station

Each PE communicates with one ring in the ring-based NoC. This communication is achieved using an Add-Drop Station (ADS). An ADS consists of two FIFOs, and the related logic to insert and remove items from the FIFO. Figure 6 shows a section of the data ring in detail. This figure shows three ADS's and PE's (indexed $i-1$, i and $i+1$). The i^{th} ADS and PE are expanded in the center of the figure. Each PE is connected to the bus ring by means of a ADS station. Each ADS station can perform one of three operations – it can *add* data into the ring, or *drop* (extract) data from the ring, or simply *repeat* the data and pass it along. Because of the extreme high rate of operation of the ring-based NoC, the *valid* and *address* signals are driven one cycle earlier than the corresponding *data* signals. This is because it would increase the cycle time if address matching and data manipulations were done in the same *Rclk* cycle. For this reason, signals which belong to the previous clock cycle are referred to with a "*prev*" subscript in the rest of this paper.

The ADS communicates with the *PE* through an inbound asynchronous FIFO (*Infifo*), and with the ring through an outbound (*Outfifo*) asynchronous FIFO. We next discuss the three operations of the ADS, using Figure 6 as a guide.

Drop Operation: This operation requires a write operation in the *Infifo*. From the right part of Figure 6, we note that the link data is captured at each cycle of *Rclk* into a *snoop* register. If R_write_en is true, then, on the rising edge of *Rclk*, data is entered into the *Infifo*, into the location pointed at by $R\_write\_pointer$.

$R\_write\_en = (valid_{prev}) \cdot (addrmatch_{prev} \cdot \overline{Full\_Infifo_{prev}})$

In other words, data is dropped if the packet is valid (first term above) and if the address matches, and the input FIFO is not full (second term). Note that each of the signals used in this computation are from the last cycle (since *valid* and *addr* are transmitted one cycle before *data*).

If there is data in the *Infifo* (i.e. *empty_infifo* is false), then data is captured into the register *fifodata_in* synchronous with *Pclk*, for consumption by the PE, as shown in the bottom right of the figure.

Add Operation: This operation requires a read operation from the *Outfifo*. From the left portion of Figure 6, we see that the FIFO output (which is the data stored in the entry pointed to by $R\_read\_pointer$) is captured in the *fifodata_out* register whenever the *Outfifo* is not empty. If *sel2* is low, then this data is driven out over the link.

$sel2 = (valid_{prev}) \cdot ((addrmatch_{prev} \cdot \overline{Full\_Infifo_{prev}}))$

We only drive out data from the *Outfifo* when *sel*2 is 0, or in other words, if the packet is invalid (first term above) or if the address matched and the *Infifo* is full (second term).

Repeat Operation: Based on Figure 6, we note that data is repeated if *sel*2 is high (i.e. the packet is valid (first term above) and if the address does not match or if the *Infifo* is full (second term).

Note that the bus ring operates at 18 GHz, therefore these operations must be performed extremely fast. Also, data is transmitted one cycle later than the control signals. If this were not the case, once the control signals were deciphered, the delay in communicating the decision to extract or drop data would add to the delay of the ADS station.

Address information is forwarded iff the packet satisfies $(valid) \cdot \overline{((addrmatch \cdot \overline{Full\_Infifo}))}$. In other words, address information is forwarded if the packet is valid, and does not match the ADS address, or if the *Infifo* is full. Otherwise, address information is sent out from the *Outfifo* on to the ring.

Valid information is forwarded when the disjunction of $(valid) \cdot \overline{((addrmatch \cdot \overline{Full\_Infifo}))}$ and $\overline{Empty\_outfifo}$ is true.

F. Description of Junction Station

A Junction Station (JS) operates in a same manner as an ADS other than the fact that it does not have any PE attached to it. A JS consists of 2 asynchronous FIFOs. One FIFO is responsible for collecting data from the horizontal ring and transferring it to the vertical ring. The other FIFO is responsible for collecting data from the vertical ring and transferring it to the horizontal ring. Both clocks in both these FIFOs operate at 18 GHz (but are mesochronous). Instead of asynchronous FIFOs, mesochronous FIFOs may be used as well [31].

IV. Experimental Results

We implemented our design in the 22 nm PTM [26] technology, with $VDD = 0.8V$. All simulations were conducted in HSPICE [15]. RLC parasitics for all the wires were extracted using Raphael [24].

A. Circuit Validation

For at-speed validation purposes, we simulated a 4×4 tile where each PE tile was assumed to be $1.229mm \times 1.229mm$. Hence the length of each link was taken to be $0.615mm$. Since the JS were $1.229mm$ apart in the vertical ring, we had to a put a repeater between two

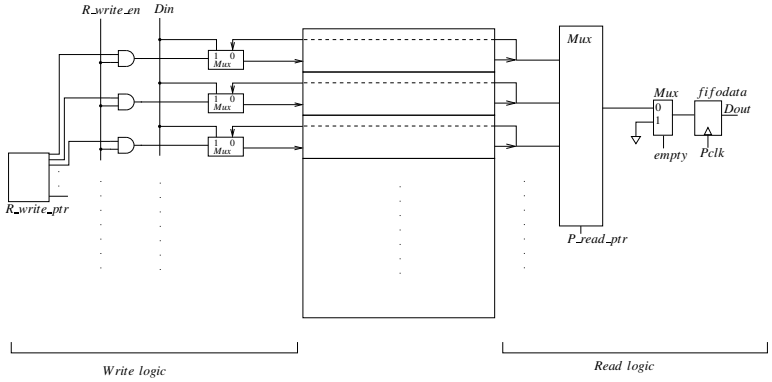


Fig. 5. FIFO Core Circuitry

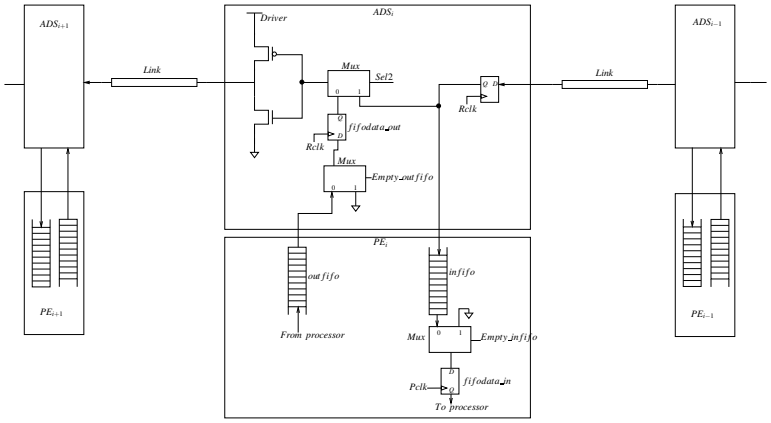


Fig. 6. Add Drop Station

JS's in the vertical ring. The address field (k) has 4 bits (to address 16 PEs) and the data width (d) was taken to be $9B$, $18B$ and $36B$. There are 2 horizontal and 2 vertical rings each of length $12.29mm$. Our simulation successfully routed a packet which traversed several ADS's and JS's before reaching its destination. We have calculated the power assuming that 26% of the links are active. The benchmark used in [21] for power consumption had a link activity of 26%, hence this choice. The areas reported are standard cell areas. The ring clock ($Rclk$) was assumed to be 18 GHz while the PE clock ($Pclk$) was 2 GHz.

The wire dimensions for the links between two stations are shown in Figure 7. Wires are implemented in Metal 9. In both the ADS and the JS, a driver of size $30\times$ of a minimum sized inverter was used to drive a single bit link with a stage ratio of $3\times$. The delay of the driver was $13.86ps$, while the delay of the mux was $16.08ps$. As the length of the link being driven is very small, the wire delay was found to be negligible. The clock-to-Q delay and the setup time adds to sum of the delay of the driver and the mux to determine the clock frequency. We found the entire 4×4 NoC operates with 100% functional correctness at 18 GHz. PVT or On-Chip variation (IR drop, local hot spots etc) can reduce the operating frequency of our ring-based NoC. Currently we address this issue by adding a nominal guard-band, which results in a operating speed of 18 GHz.

We have verified the correct operation of the *Outfifo* and *Infifo* for *Rclk* of 18 GHz and *Pclk* of 2 GHz, with varying phase randomly between the *Rclk* and the *Pclk*. In addition, we also verified the correct operation of the FIFOs in the JS where both the clocks were 18 GHz.

The ring which distributes the 18 GHz clock at the ADS and JS was assumed to be laid out on Metal 8, with wires of width $15\mu m$, spacing $1\mu m$ and height $0.9\mu m$. For a ring of length $1.229mm$ between two stations and an inverter of size $W_p = 40\mu m$ and $W_p/W_n = 2$, the oscillation frequency of 18 GHz was obtained. In order to achieve a ring of length $12.29mm$, we have implemented a $11\cdot\lambda/2$ ring. The power consumed by a single $\lambda/2$ ring was $2.67mW$.

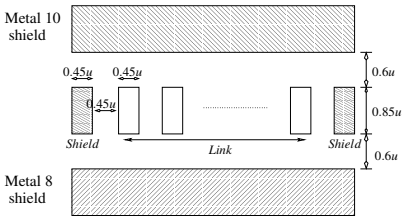


Fig. 7. Link Cross Section

B. Projection

1) Power and Area: In order to compare our results with a 16×16 mesh topology in [21], we have scaled the area and power appropriately to account for the additional ADS's, JS's and k values. The area and power comparison is given in Table II. For a flit width of 36B, we achieve a 35% gain in total power and a 11% gain in total area. The reason why our approach yields a better area is that we utilize unidirectional rings, which reduces the number of ports per intersection of horizontal and vertical links.

2) Latency and Bandwidth: In [32], the authors have presented detailed area and energy models for on-chip interconnection networks. Table III compares the network performance reported in [32], with our approach, for a 16×16 NoC.

We report the average contention-free latency (T_0) incurred by a flit from source s to destination d which includes: 1) the average hop count (H) from s to d, 2) router traversal latency (t_r) and 3) the average channel traversal latency (T_c), which is the latency induced by repeaters in long links.

$T_0(s,d) = H(s,d) * t_r + T_c(s,d)$

B_C corresponds to bisection channel count and B_B corresponds to bisection bandwidth (in bits), as defined in [32]. The flit width is taken to be 144 bits for both the topologies. $\frac{B_B}{T_{clk}^{NoC}}$ defines the aggregate bandwidth (in bits/sec) available. Note that for our scheme, T_c is zero

Flit Width	Mesh Power (W)			Mesh Area(mm^2)		Ring Power(W)			Ring Area(mm^2)	
	Router	Link	Total	Logic	Wire	Router	Link	Total	Logic	Wire
d=9B	0.70	5.81	6.51	3.400	8.976	1.57	2.11	3.68	2.246	10.07
d=18B	1.41	7.74	9.15	6.616	17.952	1.97	3.55	5.52	4.218	19.02
d=36B	2.46	11.97	14.43	14.936	35.904	2.87	6.43	9.30	8.162	36.92

TABLE II
POWER AND AREA COMPARISON WITH MESH TOPOLOGY IN [21]

Topology	H	$t_r(cycles)$	B_C	$B_B(bits)$	$\frac{B_B}{T_{clk}^{NoC}}(Gbits/sec)$	$T_c(cycles)$	$T_0(cycles)$
Mesh(16×16)	12.25	2	32	4608	9.216	10.6	35.10
Fast Ring(16×16)	24.25	1/6	16	2304	41.472	0.68	4.72
Mesh(n×n)	$\frac{3n+1}{4}$	2	2n	288n	576n	$n\times 0.6625$	$2.1625\times n+0.5$
Fast Ring(n×n)	$\frac{6n+1}{4}$	1/6	n	144n	2592n	$\frac{3n+1}{72}$	$\frac{21n+4}{72}$

TABLE III
COMPARISON OF NETWORK CONFIGURATIONS WITH MESH IN [32]

by design for horizontal rings and 1 NoC clock cycle for vertical rings. The JS which acts as a router consumes only 1 cycle if data stays in the same ring, else it takes 3 cycles if it has to switch rings. Since the horizontal and vertical rings communicate through asynchronous FIFOs, we incur a 2 cycle penalty for synchronization. The ADS always consumes only 1 cycle to repeat or drop the data. Hence the average router latency (t_r) is 1.5 cycles since it will encounter the ADS (with 1 cycle latency) and a JS (with expected penalty of 2 cycles) equally often.

In [32], the clock frequency for the NoC was assumed to be same (2 GHz) as the *PE*s. Our source synchronous ring based NoC runs 9× faster than the *PE*s. We report the result in Table III in terms of PE clock cycles. For a 16×16 topology, we improve the average contention free latency by 7.4×. This is achieved because our NoC clock is 9× faster than the *PE* clock. For the same reason, available aggregate bandwidth is 4.5× compared to a regular mesh. Note that our rings are unidirectional and hence we have half the number of links between two *PEs* compared to a bidirectional mesh as reported in [32].

V. CONCLUSIONS

Traditionally, network-on-chip (NoC) architectures are based on a mesh interconnection structures. In this paper, we present a ring based NoC architecture which is based on a source synchronous data transfer model over a ring. The source synchronous ring is clocked by a resonant clock which operates significantly faster than individual processors that are served by the ring. This allows us to significantly reduce the area devoted to the NoC logic and wiring. We have validated the design using a 22nm predictive process. Results indicate that our approach achieves 4.5× better bandwidth, 7.4× better contention free latency with lower area and power than a 2D mesh.

REFERENCES

[1] L. Bononi, N. Concer, M. Grammatikakis, M. Coppola, and R. Locatelli, "NoC Topologies Exploration based on Mapping and Simulation Models," in *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*, pp. 543 –546, 2007.

[2] M. Kim, J. Davis, M. Oskin, and T. Austin, "Polymorphic On-Chip Networks," in *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, pp. 101 –112, june 2008.

[3] S. Tota, M. R. Casu, and L. Macchiarulo, "Implementation analysis of NoC: a MPSoC trace-driven approach," in *Proceedings of the 16th ACM Great Lakes symposium on VLSI*, GLSVLSI '06, pp. 204–209, ACM, 2006.

[4] R. Balasubramonian, N. Muralimanohar, K. Ramani, and V. Venkatachalapathy, "Microarchitectural Wire Management for Performance and Power in Partitioned Architectures," in *Proceedings of the 11th International Symposium on High-Performance Computer Architecture*, (Washington, DC, USA), pp. 28–39, IEEE Computer Society, 2005.

[5] L. P. H Wang and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," in *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pp. 105 – 116, dec. 2003.

[6] H. Samuelsson and S. Kumar, "Ring Road NoC architecture," in *Norchip*, pp. 16–19, 2004.

[7] J. Kim, J. Balfour, and W. Dally, "Flattened butterfly topology for on-chip networks," *Computer Architecture Letters*, vol. 6, pp. 37 –40, feb. 2007.

[8] F. Karim, A. Nguyen, and S. Dey, "An interconnect architecture for networking systems on chips," *Micro, IEEE*, vol. 22, pp. 36 – 45, sep/oct 2002.

[9] J. Duato, S. Yalamanchili, and N. Lionel, *Interconnection Networks: An Engineering Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[10] A. Hemani, T. Meincke, S. Kumar, A. Postula, T. Olsson, P. Nilsson, J. Oberg, P. Ellervee, and D. Lundqvist, "Lowering power consumption in clock by using globally asynchronous locally synchronous design style," in *Design Automation Conference, 1999. Proceedings. 36th*, pp. 873 –878, 1999.

[11] V. H. Cordero and S. P. Khatri, "Clock distribution scheme using coplanar transmission lines," in *Proceedings of the conference on Design, automation and test in Europe*, DATE '08, (New York, NY, USA), pp. 985–990, ACM, 2008.

[12] A. Mandal, V. Karkala, S. Khatri, and R. Mahapatra, "Interconnected Tile Standing Wave Resonant Oscillator Based Clock Distribution Circuits," in *VLSI Design (VLSI Design), 2011 24th International Conference on*, pp. 82 –87, jan. 2011.

[13] J. Wood, T. Edwards, and S. Lipa, "Rotary traveling-wave oscillator arrays: a new clock technology," *IEEE Journal of Solid-State Circuits*, vol. 36, pp. 1654–1665, Nov 2001.

[14] J. Wood, T. Edwards, and C. Ziesler, "A 3.5GHz Rotary-Traveling-Wave-Oscillator Clocked Dynamic Logic Family in 0.25 μm CMOS," *Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers. IEEE International*, pp. 1550–1557, Feb. 2006.

[15] I. Meta-Software, "HSPICE user's manual," Campbell, CA.

[16] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Design Automation Conference, 2001. Proceedings*, pp. 684 – 689, 2001.

[17] W. J. Dally and C. L. Seitz, "The torus routing chip," *Distributed Computing*, vol. 1, pp. 187–196, 1986. 10.1007/BF01660031.

[18] C. E. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," *IEEE Trans. Comput.*, vol. 34, pp. 892–901, October 1985.

[19] Y. Thonnart, P. Vivet, and F. Clermidy, "A fully-asynchronous low-power framework for GALS NoC integration," in *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, (3001 Leuven, Belgium, Belgium), pp. 33–38, European Design and Automation Association, 2010.

[20] D. Ludovici, A. Strano, G. N. Gaydadjiev, and D. Bertozzi, "Mesochronous NoC technology for power-efficient GALS MPSoCs," in *Proceedings of the Fifth International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip*, INA-OCMC '11, (New York, NY, USA), pp. 27–30, ACM, 2011.

[21] D. Sanchez, G. Michelogiannakis, and C. Kozyrakis, "An analysis of on-chip interconnection networks for large-scale chip multiprocessors," *ACM Trans. Archit. Code Optim.*, vol. 7, pp. 4:1–4:28, May 2010.

[22] T. Bjerregaard, "The MANGO clockless network-on-chip: Concepts and implementation," 2005. Supervised by Assoc. Prof. Jens Sparsø, IMM.

[23] A. T. Tran, D. N. Truong, and B. Baas, "A Reconfigurable Source-Synchronous On-Chip Network for GALS Many-Core Platforms," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 29, pp. 897 –910, June 2010.

[24] "Raphael Interconnect Analysis Tool: User's Guide."

[25] V. Karkala, K. Bollapalli, R. Garg, and S. Khatri, "A PLL design based on a standing wave resonant oscillator," in *Computer Design, 2009. ICCD 2009. IEEE International Conference on*, pp. 511 –516, oct. 2009.

[26] "PTM website." `http://www.eas.asu.edu/~ptm`.

[27] U. Nawathe, "Design and implementation of Sun's Niagara2 processor," *Technical Report, Sun Microsystems*, 2007.

[28] G. Gerosa, S. Curtis, M. D'Addeo, B. Jiang, B. Kuttanna, F. Merchant, B. Patel, M. Taufique, and H. Samarchi, "A Sub-2W Low Power IA Processor for Mobile Internet Devices in 45 nm High-k Metal Gate CMOS," *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 1, pp. 73 –82, 2009.

[29] E. C. Cummings and P. Alfke, "Simulation and Synthesis Techniques for Asynchronous FIFO Design with Asynchronous Pointer Comparisons," *Technical Report, Sunburst Design*, 2002.

[30] W. J. Dally and J. W. Poulton, *Digital Systems Engineering*. Cambridge University Press, 1998.

[31] T. Chelcea and S. Nowick, "A low-latency FIFO for mixed-clock systems," in *VLSI, 2000. Proceedings. IEEE Computer Society Workshop on*, pp. 119 –126, 2000.

[32] J. D. Balfour and W. J. Dally, "Design tradeoffs for tiled CMP on-chip networks," in *International Conference on Supercomputing*, pp. 187–198, 2006.