International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 10, No. 2, July 2021, pp. 77~89
ISSN: 2089-4864, DOI: 10.11591/ijres.v10.i2.pp77-89

Cost-efficient reconfigurable geometrical bus interconnection system for many-core platforms

Tirumale Ramesh, Khalid Abed
Department of Electrical & Computer Engineering and Computer Science, Jackson State University, USA

Article Info

Article history:

Received Oct 20, 2020
Revised Oct 29, 2020
Accepted Feb 15, 2021

Keywords:

Artificial intelligence
Cost per bandwidth
Edge computing
Geometrical bus connections
Many-core embedded
On-chip interconnection
Reconfigurable

ABSTRACT

System-on-chip (SoC) embedded computing platforms can support a wide range of next generation embedded artificial intelligence and other computationally intensive applications. These platforms require cost effective interconnection network. Network-on-chip has been widely used today for on-chip interconnection. However, it is still considered expensive for large system sizes. As full bus-based interconnection has high number of bus connections, reduced bus connections might offer considerable implementation economies with relatively small design cost for field programmable gate arrays (FPGAs) based embedded platforms. In this paper, we propose a cost efficient generalized reconfigurable bus-based interconnection for many-core system with reduced number of bus connections. We generalize the system with b number of interconnect buses in which $b = min\{n, m\}/k$ where n is the number of processor cores, m is the number of memory-modules and k is the general bus reduction factor. We present four geometrical interconnect configurations and provide their characterization in terms of memory bandwidth, cost per bandwidth and bus fault tolerance for various system sizes. Our results show that these configurations provide reduced cost per bandwidth and can achieve higher system throughput with bus cache.

Corresponding Author:

Khalid Abed
Department of Electrical and Computer Engineering and Computer Science
Jackson State University
1400 John R Lynch Street (JSU Box 17098), Jackson, MS. 39217, United States
Email: khalid.h.abed@jsums.edu

1. INTRODUCTION

Multi-core processor designs are now seeing big shift [1] towards many-core computing [2]. Many-core processor systems are now trending as a platform for massive parallel computing [3] and are also had been used as a co-processor for a multi-core system [4]. In recent years, we see an emergence of extreme computing [5] for big data. Some architectural models for many-core processor systems [6, 7] and performance improvement techniques [8] for multi-core have been developed. Many-core computing is gaining interests for artificial intelligence (AI) based defense applications [9]. Edge computing [10] is now considered as a relatively a new paradigm where the computational resources are placed at the edge of the network. Use of multi-core is gaining interests for edge computing [11] as data transfer between cores is power demanding and require a very complex connection infrastructure. With increased very large-scale integration (VLSI) density, heterogeneous many-core based chip multiprocessors (CMP) [12] for big data are on the rise. For these applications running on many-core platforms, we require cost efficient on-chip interconnection.

Most of the work today for on-chip interconnection is network-on-chip (NoC) [13-15]. They generally offer more degree of freedom than bus-based systems. NoC has high overhead due to packetization and multi-hop routing, which affects physical area, power and latency leading to increased cost and creates challenges for achieving high performance. Current NoC researches continue to address these challenges. In recent years, some novel NoC architectures [16-21] were proposed. In [16], buffer less circuit switched NoC was addressed for both cost and energy efficiency. A hybrid NoC was proposed in [17]. Reliability of NoC router was focused in [18]. A field programmable gate array (FPGA) based solution targeting area optimization for improved network performance using several techniques was proposed recently in [19]. There was some recent work on routerless NoC [20] using smart on-chip wiring resource management for reducing cost. In [21], an efficient crossbar switch implementation for NoC was proposed. Heterogeneous computing [22] exists today in the form of functionally diverse computational devices, memory systems and includes heterogeneity in interconnection. Machine learning is gaining interests in processor affinity characterization for functionally diverse heterogenous devices [23]. There is some recent work [24] on optimization of multi-core systems using machine learning. Smaller size many-core processors with NoC on FPGAs are built [25]. There is some recent hybrid NoC work [26] that utilized bus-based interconnection.

Achieving energy efficient on-chip interconnection at reduced area cost for the scale of hundreds of many-cores is a major challenge. Bus-based interconnection generally considered simpler in design, energy efficient and fault tolerant than NoC; but lacks scalability for large system size. Some VLSI experts in the past argued that metal/wiring are cheap and plentiful and we are no longer pin-limited thus favoring bus-based on-chip interconnection [27]. For multi-core processor systems, performance may not be maximized by even using highest bandwidth on-chip interconnects topology as it consumes power and area resources [28]. As further argued in [28], increasing the number of multi-cores places higher interconnect bandwidth demand which decreases the available silicon real estate for large size multi-cores. Thus, we believe that by taking advantage of current VLSI density, silicon real estate can be still be managed for small size many-cores combined with reduced cost interconnection. This motivated us to look into simpler reduced cost bus-based on-chip interconnection solution for many-cores that can achieve same bandwidth and offer good bus fault tolerance for moderate number of cores (16-128). Our main contributions of the paper are as follows:

- Propose a generalized reconfigurable many-core system with cost effective geometrical bus on-chip interconnection configurations extended from our earlier work [29].
- Present the bus arbitration algorithm for these configurations and provide a comprehensive system characterization in terms of memory bandwidth, cost per bandwidth, bus fault tolerance and system throughput with bus cache added on each bus line.
- Present detailed results and discuss these results.
- Estimate cost of these configurations in comparison to an example circuit switched router.
- Provide conclusion and present some insight into future research.

2. SYSTEM ARCHITECTURE AND CONIGURATIONS

2.1. Generalized reconfigurable many-core system platform

From the earlier work on multiple bus system [30], it has been observed that by using number of buses equal to one-half of the number of memory-modules or processors, we can achieve a memory bandwidth within 25 % of the crossbar bandwidth. For complete bus connections, all memory-modules connected to all buses. Increase in buses incurs high number of bus connections and cost. Earliest work on reduced bus connection schemes [31] showed general theorems and properties for the scheme. With reduction in the number of buses and bus connections, we propose a reconfigurable many-core platform with *n* cores, *m* memory-modules, *b* buses and *k* bus reduction factor. Figure 1 shows the system architecture that includes bus cache at each bus line and reconfigurable control.

$$b = \frac{\min\{n,m\}}{k} \quad (1)$$

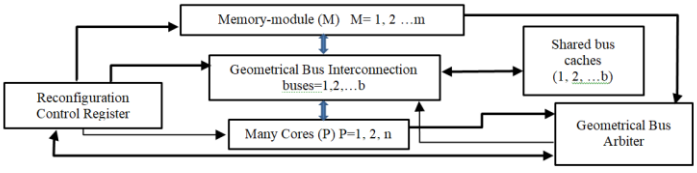

Figure 1. Reconfigurable many-core system platform

2.2. Geometrical bus interconnection configurations

Our earlier work in [29], we complimented [31] and presented a generalized system architecture and characterization. In this paper we supplement expands our earlier work [29] and propose four distinct geometrical bus configurations: i). Group Rhombic 2 (GR2), ii). Group Rhombic 4 (GR4), iii). Hierarchical Rhombic (HR) and iv). Quadrant Rhombic (QR). We provide a comprehensive system characterization for these configurations. Although in generality, we stated a bus reduction factor k in (1), we used $k = 2$ throughout this paper. We considered rhombic as a geometrical pattern base to define these four configurations as rhombic was considered most cost-effective topology [29]. However, in general, any of the other geometrical pattern presented in [29] could be used as a base. In Figure 2, Figure 3 and Figure 4, all connections marked as "x" refers to the buses numbered on its left.

- GR2: Memory-modules and buses divided into two groups connected in rhombic pattern. Figure 2 shows the configuration. All processor cores connected to all buses.
- GR4: Memory-modules and buses divided into four groups connected in rhombic pattern. Figure 3 shows the configuration. All processor cores connected to all buses.
- HR: Memory-modules and cores connected in hierarchical bus system [32] with modifications. Cores are connected in two groups in level 1 and memory-modules connected in rhombic pattern in level 2. Processor core and memory buses are interconnected. Figure 4 shows both memory-module and processor core connections.
- QR**:** Memory-modules and buses divided into four quadrants with each quadrant connected in rhombic pattern. Figure 5 shows the memory-module connections

Group and quadrant rhombic configurations are tightly coupled as both core and memory bus connections are on the same set of buses. Hierarchical rhombic is a loosely coupled configuration since they have separate core and memory bus connections. It was shown in [31] that for any rhombic connections, the lower bound for number of memory-modules connected to each bus is equal to $(m - b + 1)$ to assign b buses to b memory-modules. In Figures 2 through 5 below shows these configurations. M1-M16 is memory-modules and P1-P8 is processor cores. We explicitly shown processor core connections only for hierarchical configuration.

BUS	M1	M2	M3	M4	BUS	M5	M6	M7	M8
1	x	x	x		3	x	x	x	
2		x	x	x	4		x	x	x

Figure 2. Group rhombic 2(GR2) for n=m=8, b=4

BUS	M1	M2	M3	M4	BUS	M5	M6	M7	M8
1	x	x	x		3	x	x	x	
2		x	x	x	4		x	x	x

BUS	M9	M10	M11	M12	BUS	M13	M14	M15	M16
5	x	x	x		7	x	x	x	
6		x	x	x	8		x	x	x

Figure 3. Group rhombic 4(GR4) for n=m=16, b=8

BUS	M1	M2	M3	M4	M5	M6	M7	M8
1	x	x	x	x	x			
2		x	x	x	x	x		
3			x	x	x	x	x	
4				x	x	x	x	x

BUS	P1-P4	BUS	P5-P8
1	x	3	x
2	x	4	x

Figure 4. Hierarchical rhombic (HR) for n=m=8, b=4

BUS	M1	M2	M3	M4	M5	M6	M7	M8
1	x	x	x		x	x	x	
2		x	x	x		x	x	x
3	x	x	x		x	x	x	
4		x	x	x		x	x	x

Figure 5. Quadrant rhombic (QR) for n=m=8, b=4

Table 1 summarizes the total cost of interconnections. The first term is the processor core connection cost, the second term is the memory connection cost and the third term is the core to memory bus inter-level connection cost for hierarchical rhombic. As observed from Table 1, group rhombic has reduced number of bus connections compared to regular rhombic [29]. Quadrant rhombic has slightly higher number of bus connections than rhombic but is inherently fault tolerant to critical bus faults (explained in section 3.6). All four configurations have reduced number of bus connections compared to complete bus connections. Figure 6 shows the average cost savings. The cost savings ranges from 24 % to 42 % across all system sizes.

Table 1. Cost of interconnections (no of connections)

Connection Topologies	**Total Cost**	System Size			
		16	32	64	128
Complete bus connections	$b[n+m]$	256	1024	4096	16384
Regular rhombic[29]	$n.b+b[m-b+1]$	200	784	3104	12352
Group Rhombic 2	$n.b+b\left[\frac{m}{2}-\left(\frac{b}{2}-1\right)\right]$	168	656	2592	10304
Group Rhombic 4	$n.b+b\left[\frac{m}{4}-\left(\frac{b}{4}-1\right)\right]$	152	592	2336	9280
Hierarchical Rhombic	$n.\frac{b}{2}+b[m-b+1]+b.b$	200	784	3104	12352
Hierarchical Rhombic	$n.b+2b\left[\frac{m}{2}-\left(\frac{b}{2}-1\right)\right]$	208	800	3136	12416

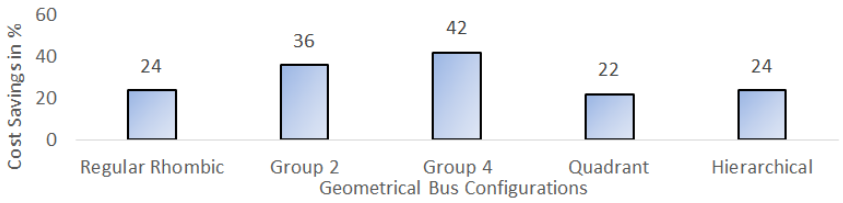

Figure 6. Average cost savings

2.3. System reconfiguration

Reconfiguration Control Register (RCR) shown in Figure 7 facilitates reconfiguration of the system for plurality of processor cores, memory-modules and buses connected to the system and have the following functions:

- Switches SP, SM and SB: Connects cores, memory-modules and buses to the system respectively.
- Registers SPR, SMR, and SBR: Controls SM, SP, and SB respectively. For example, if SP(i)=1, then a core i is connected to the system. Similarly, SM(j) $=1$ connects a memory j and SB(k) $=1$ connects a bus k to the system.
- Registers SCPR and SCMR: Controls the reconfiguration of the interconnection to connect a core or memory to a specific bus. For example, memory "1" connected to bus "1" for SCM $(1,1)=1$.
- On a single bus fault, the interconnection is reconfigured and updates the SCPR, SCMR and SBR registers. Reconfiguration is also performed for group rhombic to add connections.

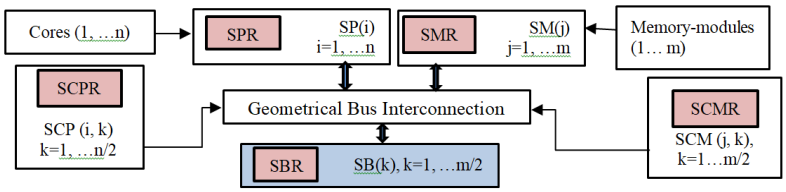

Figure 7. System reconfiguration controls register (RCR)

Table 2 gives an illustrated RCR for bus "2" for GR2 and QR.

Table 2. System reconfiguration illustration for n=m=8

Configuration	Bus *k*	SBR(k)	SCPR (i, k)	SCMR (j, k)
Group Rhombic 2	2	[1111]	[11111111]	[0111 0000]
Quadrant Rhombic	2	[1111]	[11111111]	[0111 0111]

3. GEOMETRICAL BUS INTERCONNECTION VALIDATION AND ARBITRATION

When random bus is assigned for geometrical bus configurations, some memory requests may not complete in the current memory cycle due to no bus connection to the memory; as a result, it requires specific bus arbitration. For assigning b distinct buses to b memory-modules, we require that the number of memory-modules connected to each bus for rhombic connection is equal to $(m - b + 1)$ [31]. This is the lower bound condition. We validate this theorem for our proposed group and quadrant configurations using the corollaries in the following section.

3.1. Geometrical bus interconnection configuration validation

Let $r(i)$ be the number of successful distinct bus connections in each group i for $i = 1,..g$ where $g = 2$ *or* 4 or in each quadrant *i* for $i = 1, ...4$.

- *Corollary* 1: For group rhombic configurations, if a bus is connected to less than $m - b + 1$ memory-modules, then we cannot assign *b* distinct buses; in this case, additional connections needed.
  Proof: For b requests, group rhombic satisfies the lower bound condition locally (within group). However, as not all memory-modules connections exist in each group, $\sum_{i=1}^{g} r(i) < b$. Hence, the lower bound condition for the overall interconnection will not be satisfied and require at most b additional connections.
- *Corollary* 2: Let A = {1, 2, …$m/2$} and B= {($\frac{m}{2}$)+1, …m}. For b memory requests, if $\frac{b}{2}$ memory requests $\in A$ and $\frac{b}{2}$ memory requests $\in B$, then b distinct buses can be assigned for GR2. This can be extended to GR4 with sets A, B, C, D of memory request each mapped to each group.
  Proof: In group rhombic, with g groups , b/g rhombic connections exist giving $\sum_{i=1}^{g} r(i) = \text{g. b/g} = b$. In this case, the lower bound is satisfied in each group and b distinct buses can be assigned.
- *Corollary* 3: For quadrant rhombic, if every bus is connected to $m - b + 1$, then there exists a bus arbitration algorithm that can assign b distinct buses for *b* memory-modules.
  Proof: For any *b* memory requests, each quadrant is connected in rhombic and satisfies the lower bound [31] within each quadrant. In this case, $r(i)$ takes combination giving $\sum_{i=1}^{4} r(i) = b$. Hence, b distinct buses can be assigned.
- *Corollary* 4*:* Let A = {1, 2, …$m/2$} and B = {$m/2$+1, …m}. For b memory requests, if $\frac{b}{2}$ memory requests $\in A$ and $\frac{b}{2}$ requests $\in B$, then b distinct buses can be assigned in quadrant rhombic.
  Proof: For quadratic rhombic (see Figure 5), the lower bound condition is applied to $(m/2 - b/2 + 1)$ memory-modules connected to each quadrant. These connections are contributed from one quadrant each from left half quadrant ($b/2$ connections) and right half quadrants ($b/2$ connections). In total, we can assign b distinct buses to b memory-modules.

3.2. Geometrical bus arbitration algorithm for hierarchical and quadratic rhombic

Let A be the set of b memory requests sorted in ascending order and M is the memory-module in A. Let $bg = bg[1, ... b]$ be the status of the bus assigned with $bg[bus] = 0$ and $bus = 0$ initially. Let $count$ be the number of successful memory-bus connection with $count = 0$ intially. Let $u[bus, M]$ be the connection

from bus to M. The algorithm searches for a bus *in order* for every M in set A and grants the bus if it is connected to M. If the bus $> b$, then the next bus is updated to "1" and the search continues.
For hierarchical rhombic, we arbitrate core bus as:

Core bus← $\left[\frac{P}{2}\right]$, example core bus=1 if core=1

The memory bus arbitration given as:

for each M *in* A **do**:
until $u[bus, M] = 1$ *and* $bg[bus] = 0$
$count = count + 1$
$bus = bus + 1$
$bg[bus]$← 1
if $bus > b$ then $bus = bus - b$

3.3. Geometrical bus arbitration algorithm for group rhombic

Let A be the set of b requests and M is the memory-module in set A and $bus = 0$ initially. Let $count$ be the number of successful memory-to-bus connections with $count = 0$ initially. Let u be the connectivity defined as before. The algorithm searches a bus for every M in set A and grants the bus if is connected to M. If the bus $> b$, then the next bus is updated to bus "1" and the search continues.
The memory bus arbitration given as:

for each M *in* A **do**
if M≤ b:
if $u[bus, M] = 1$ and $bg[bus]$=0
$count = count + 1; bg[bus]$]← 1
$bus = bus + 1$
if M> b:
if bus>b
bus=bus-b
if $u[bus, M] = 1$ and $bg[bus]$ =0
$count = count + 1; bg[bus]$]← 1
$bus = bus + 1$

3.4. Geometrical bus arbitration simulation

We conducted extensive geometrical bus arbitration simulation for all system sizes and configurations to verify the algorithm. We show the illustrated assigned buses to memory (shaded) in Figures 8 through 9.

- *Simulation* 1: Quadratic Rhombic: Memory Request= $\{M3, M4, M5, M8\}$.
- *Simulation* 2*:* Group Rhombic 2: Non-Favorable Request = $\{M1, M5, M6, M7\}$
- *Simulation* 3*:* Group Rhombic 4: Favorable Request= {M2, M3, M5, M8}

Figure 8 shows the bus assignment, Figure 9 shows the bus assignment (color shaded) for simulation 2 indicating that added connection R on reconfiguration is required to grant b buses to memory request, Simulation 3 is shown in Figure 9 (color bolded) for favorable memory requests requiring no additional connections and b buses can be assigned.

BUS	M1	M2	M3	M4	M5	M6	M7	M8
1	x	x	x		x	x	x	
2		x	x	x		x	x	x
3	x	x	x		x	x	x	
4						x	x	x

Figure 8. Bus assignment for quadratic rhombic for n=m=8

BUS	M1	M2	M3	M4	M5	M6	M7	M8
1	x	x	x					
2		x	x	x	R			
3					x	x	x	
4						x	x	x

Figure 9. Bus aassignment illustration for quadratic rhombic for n=m=8

3.5. Memory bandwidth

From the multiple-bus bandwidth analysis [30], with random bus arbitration, the memory bandwidth BW_{crb} with reduced number of buses ($b = m/2$) and complete bus connections is given by:

$$BW_{crb} = BW_{cb} - BW_r \quad (2)$$

The first term in (2) is the crossbar bandwidth, the second term BW_r is the reduction in bandwidth with due to reduce number of buses. As the number of bus connections reduced for geometrical bus configurations, the bandwidth in (2) is further reduced by:

$$1/m \sum_{i=1}^{b}(m - m_i) \quad (3)$$

Where m_i is the number of memory-modules connected to bus i and $(m - m_i)$ is the number of memory-modules not connected to bus *i*. As equal number of memory-modules are connected to each bus in a rhombic topology [29], the second term in equation (4) gives the average number of memory-modules not connected to any bus.

$$BW_{grb} = BW_{crb} - b(m - m_i)/m \quad (4)$$

where BW_{grb} is the memory bandwidth using geometrical bus configuration with random bus arbitration.

We analytically derive the reduction in bandwidth in (4) using lower bound condition $(m - b + 1)$ [31].

- *Corollary* 5: When a bus is arbitrated randomly for hierarchical and quadrant geometrical bus configurations, the reduction in bandwidth is equal to $(b - 1)/2$ for $k1$ number of memory-modules not connected to a bus.
  Proof: For rhombic based connections, we require $k1 \leq m - (m - b + 1) = k1 \leq (b - 1)$. This sets a threshold for $k1$ for reducton in bandwidth to $b - 1$. The average number of memory-modules not connected to a bus is then equal to $(b - 1)/m$. The average number of memory-modules not connected to any bus is equal to $b(b - 1)/m$ which reduces to $(b - 1)/2$ for $m = 2b$.

The reduction $(b - 1)/2$ is also equal to second term in (4). In general, memory bandwidth with random bus assignment for g number of groups is given by:

$$BW_{rb} = b - (m - \frac{m}{g} + \frac{b}{g} - 1)/2 \quad (5)$$

we see that the reduction in bandwidth increases with g. For hierarchical and quadrant rhombic, we can apply g=1 as a special case for equation (5) which yields the reduction of $(b - 1)/2$ as stated in corollary 5.

- *Corollary* 6: For hierarchical and quadrant configurations, with geometrical bus arbitration, the reduction in bandwidth in (5) is nullified.
  Proof: For $k2$ memory-modules connected to each bus; it requires $k2 \geq (m - b + 1)$ for assigning b distinct buses to b memory-modules. For m=2b gives $k2$=b+1. As $k2 > b$-1, the reduction in (5) is nullified giving effective bandwidth of b.

However, for group rhombic, there still exists some small reduction in bandwidth from b as $k2 < b - 1$. But, if there is a favorable memory request pattern, then each group locally satisfies the lower bound condition [31] yielding $k2 > b - 1$ thus assigning b distinct buses to b memory-modules. For complete bus connections, we can deduce $k2 = 2b$. As $k2 > b$-1, it nullifies the reduction in bandwidth validating that the bandwidth for multiple bus system [30] with complete bus connections is given by (2).

3.6. Bus load and bus fault tolerance

Bus load is the number of memory-modules connected to each bus. As we increase the bus load, the capacitive loading on the bus increases; a point is eventually reached when speed up with multiple processors gets saturated. Memory load is the number of buses connected to a memory-module and dictates the bus fault tolerance. If a bus $i \in b$ fails, then it forces the lower bound to $(m - b + 2)$ [31]. We look at two specific bus fault conditions for hierarchical and quadrant connections:

Critical Buses: Buses with a memory load of "1". Bus "1" and "b "are critical buses. If critical buses fail, then the memory-module "1" or "*m"* is completely disconnected. The remedy is to provide an additional connection R to critical buses. Figure 10 shows the bus assignment (color shaded) for a critical bus fault with a memory request {M1, M5, M6, M7} for quadrant rhombic. In this case, memory-module "1" is not

completely disconnected. Thus, we see that quadrant rhombic is inherently fault tolerant to critical buses and ensures assignment of $b-1$ buses requiring no additional connections.

Non-Critical Buses: Buses with a memory load >1 are non-critical buses. When there is a fault on a non-critical bus, the memory bandwidth is degraded to $(b-1)$; but no memory-module is disconnected and always a bus connection exists for all memory-modules.

Group rhombic is less bus fault tolerant as the number of critical buses nb_c increase with the number of g groups. We note that $nb_c = 2g$. Non-critical bus faults can disconnect $2(m/g - b/g + 1)$ memory-modules. However, the added connections can sustain these faults. Even with the increase in cost for added connections, the group interconnection is still cost effective.

BUS	M1	M2	M3	M4	M5	M6	M7	M8
1	Faulty Critical Bus							
2		x	x	x		x	x	x
3	x	x	x		x	x	x	
4		x	x	x		x	x	x

Figure 10. Quadrant rhombic with bus fault

4. PERFORMANCE CHARACTERIZATION

In this section, we present the results of our system characterization in terms of cost per bandwidth, cost per degraded bandwidth and system throughput with bus cache.

4.1. Cost per memory bandwidth

Table 3 shows the memory bandwidth of geometrical bus interconnection where, col 1 represents bandwidth with random bus arbitration from eq. (4). For group rhombic, col 2 represents bandwidth with no added connections and col 3 represents bandwidth with added connections (both shaded) by using geometrical bus arbitration algorithm in sections 3.3. We determined the average bandwidth from over 100 iterations of random memory request. We obtained 30 to 50 % reduction in memory bandwidth with group rhombic when algorithm 3.3 is applied without any added connections. However, as we added bus connections, the bandwidth increases to $(b-1)$. For HR and QR, col 2 represents bandwidth from algorithm 3.2 giving the same bandwidth as with complete bus connections (row 1).

Table 3. Memory bandwidth

Interconnection	**System Size**											
	16			**32**			**64**			**128**		
Multiple-Bus [30]		7.9			15.9			31.9			63.9	
Group Rhombic 2	2.4	**5.6**	**7.4**	4.4	**13.4**	**15.2**	8.5	**28.5**	**30.7**	16.5	**58.8**	**61.7**
Group Rhombic 4	1.5	**4**	**7**	2.4	**11**	**15**	4.5	**26**	**31**	8.5	**57**	**64**
Hierarchical	4.4	7.9		8.5	15.9		16.5	31.9		32.5	63.9	
Quadrant	4.9	7.9		8.9	15.9		17	31.9		33	63.9	

Table 4 shows the cost per bandwidth for all system sizes. For group rhombic, col 1 and col 2 represents the result with no added connections and added connections (both shown shaded) and col 3 represents cost per bandwidth for favorable memory request. Figure 11 shows the average cost per bandwidth across all system sizes. The reduction in average cost per bandwidth varies from $1.3x$ to $1.8x$ compared to complete bus connections.

Table 4. Cost per bandwidth

Interconnection	System Size											
	16			**32**			**64**			**128**		
Complete Bus Connections	32			64			128			256		
Group Rhombic 2	**29.8**	**24.8**	21	**48.8**	**44.8**	41	**91.2**	**85.8**	81	**174.7**	**168.5**	161
Group Rhombic 4	**38**	**26.3**	19	**53.8**	**45.7**	37	**89.8**	**82.1**	73	**162**	**151**	145
Hierarchical/Quadrant	25		25	49			97			193		

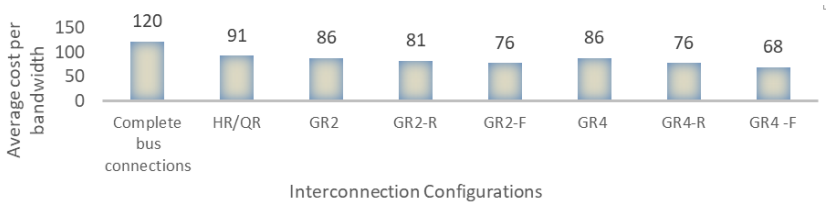

Figure 11. Average cost per bandwidth

4.2. Cost per degraded bandwidth

We ran extensive simulation for determining the degraded bandwidth on a single bus fault. Table 5 shows the average cost per bandwidth (col 1) and cost per degraded bandwidth (col 2).

Table 5. Cost per bandwidth comparisons

Interconnection	System Size							
	16		**32**		**64**		**128**	
Group Rhombic 2-R	24.8	28.2	44.8	48.3	85.8	88.3	168.5	170.2
Group Rhombic 4-R	26.3	36.2	45.7	48.1	82.1	82.1	151	153.4
Hierarchical/Quadrant	25	28.6	49	52.3	97	100.1	193	196.1

Figure 12 shows the average increase in cost per bandwidth (from col 1 to col 2 in Table 5) across all system sizes. The percentage average increase in cost per bandwidth due to bandwidth degradation varies from 3.6 to 4.8.

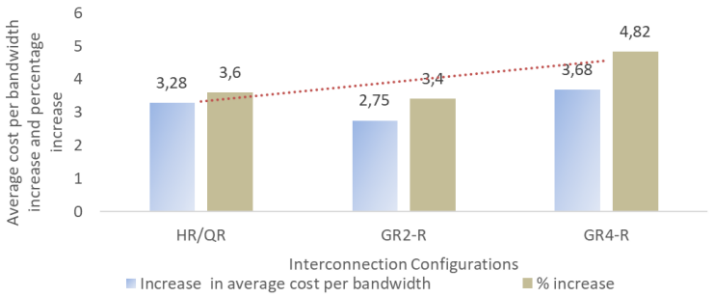

Figure 12. Average cost per bandwidth comparisons

4.3. Effective system throughput with bus cache

As the number of processor cores on a chip multiprocessor increased, there is always a challenge to provide adequate interconnection bandwidth. Use of multi-level cache can increase the system throughput. However, use of large number of fast on-chip private core caches increases the system cost. A much slower shared bus cache placed on every bus line can optimize the overall system cost and reduce average memory access time leading to lower clocks per instruction (CPI). The hit ratio of bus cache can be given as:

$$hbc = (BW_{cb} - BW_{gb}) / BW_{cb} \quad (6)$$

Where BW_{cb} is the crossbar bandwidth [30] and BW_{gb} is the bandwidth using geometrical bus (Table 3). Table 6 shows the effective system throughput with bus cache. In Table 6, col 1 correspond to throughput using hbc from (6) and col 2 represents throughput with hbc increased by 15 % from col 1. As observed in Table 6, we see an increase in effective throughput of the system when bus cache hit ratio is increased by 15 %. Figure 13 shows the average throughput for each system size. We observe that the average throughput is higher for hierarchical and quadrant rhombic compared to group rhombic. However, as the system size increases, the difference in throughput from hierarcchical and quadrant rhombic to group rhombic reduces; suggesting an advantage for using bus cache for group rhombic with increase in system size.

Table 6. System throughput with bus cache

Interconnection	System Size							
	16		**32**		**64**		**128**	
	BW_{cb}= 10.4		BW_{cb}= 20.4		BW_{cb}= 40.6		BW_{cb}= 80.2	
Group Rhombic 2-R	9.2	9.5	18.4	18.9	37.2	38.1	74.9	76.2
Group Rhombic 4-R	8.6	8.6	18.2	18.6	36.4	38.6	76.4	80.2
Hierarchical/Quadrant	9.8	10.1	19.5	19.9	38.8	39.8	77.6	78.9

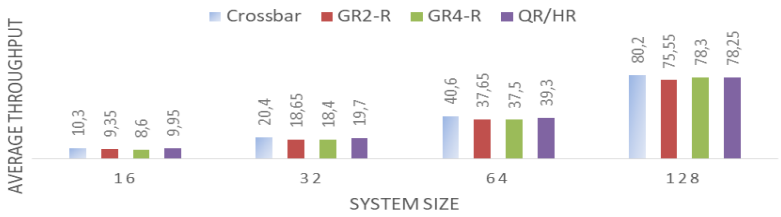

Figure 13. Average system throughput with bus cache

4.4. Estimated cost comparisons to NoC CLOS network

We compared generically the geometrical bus configurations cost with a circuit switched NoC [16] based on CLOS network [33] using the same system size. In the CLOS based NoC [16], the network was organized as 3 lanes (input, middle and output) consisting of $k\ x\ k$ crossbar switches. For example, a 20 x 20 circuit switched router has five 4 x 4 switches in the input lane, four 5 x 5 switches in the middle lane and five 4 x 4 switches in the output lane. We assumed a relative cost increase by $4x$ for each switch input/output in CLOS NoC compared to the geometrical bus connection switch due to network layers and buffers. We used switch input unit cost increase for regular crossbar by $2x$ compared to geometrical bus connection switch due to larger switch matrix at each cross point. Table 7 and Figure 14 shows the estimated cost comparisons. We noticed an average reduction by $2x$ in estimated cost of the geometric bus interconnection compared to CLOS based circuit switched NoC across all configurations.

Table 7. Cost comparisons with circuit switched NoC [16]

Size	HR	GR2	GR4	QR	NoC	NoC Configuration
16	200	168	152	208	768	(4, 4 x 4,), (4, 4 x 4), (4, 4 x 4)
32	784	656	592	800	2048	(8, 4 x 4), (4, 8 x 8), (8, 4 x 4)
64	3104	2592	2336	3136	6144	(16, 4 x 4), (4, 16 x 16), (16, 4 x 4)
128	12352	10304	9280	12416	20480	(32, 4 x 4), (4, 32 x 32), (32, 4 x 4)

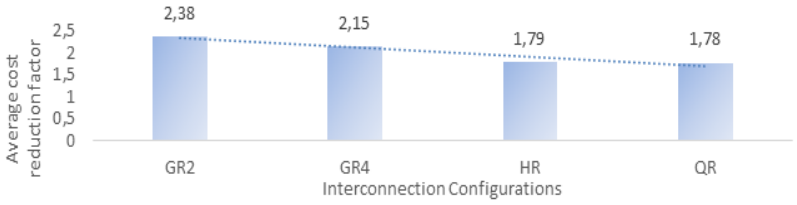

Figure 14. Estimated average cost reduction factor with circuit switch router NoC

4.4. Summary of results

In summary, our results are as follows:

- Group rhombic offer the best average cost savings (36 % to 42 %). For non-favourable memory requests additional connections are required to achieve same memory bandwidth of b. Even by adding at most b number of connections, group rhombic still offers the best average cost savings making it a good choice for higher system sizes.
- We achieved reduction of 1.5x cost per memory bandwidth with group rhombic across all system sizes.
- Quadratic and hierarchical rhombic achieves the same memory bandwidth without requiring any additional connections and we achieved reduction of 1.3x cost per memory bandwidth.

- A single critical bus fault disconnects memory-module that is only connected to the critical bus. A connection reconfiguration is needed for fault tolerance. For all configurations with a single bus fault, the average cost per degraded bandwidth is within 5 % of the cost per bandwidth.
- Quadrant rhombic configuration is inherently fault tolerant for critical bus failure.
- The average effective system throughput with bus cache is within 10 % of crossbar bandwidth.
- System throughput is higher for hierarchical and quadrant rhombic compared to group rhombic with added connections. However, the throughput difference of hierarchical/quadrant rhombic from group rhombic reduces as system size increases.
- The average cost reduction for geometrical bus interconnection to CLOS based NoC is 2x and varies from 1.8x to 2.4x.
- The added connections in group rhombic also allow for good fault tolerance for critical and non-critical bus faults.
- Optimum selection for configuration is to use inherently fault tolerant quadrant rhombic for small system sizes (less than 32 cores) and use group rhombic for larger system size (greater than 32 cores) to take cost reduction advantage for higher system size.

5. CONCLUSION AND FUTURE RESEARCH

For moderate number of small sized processor core from 16 to 128 (many-cores), few cost-effective geometrical bus interconnection configurations with reduced number of buses and reduced number of bus connections may offer as a cost-effective solution for on-chip interconnection. These configurations may provide an overall system cost advantage for many-core platforms. Today's VLSI density advantage plays a major role for the implementation economies of the interconnection. We achieved low cost per bandwidth and good bus fault tolerance with bandwidth degraded by less than 5 % across all geometrical bus configurations. By placing a small bus cache on each bus line, we achieved an increase in the overall system throughput. From our results, we conclude that quadrant rhombic is the best option for lower system size ($\leq$ 32 processor cores) and group rhombic is a better option for larger system sizes (>32 processor cores).

For further research, our first plan is to extend the work to present a comprehensive interconnection system simulation combining on-chip multi-level cache that includes bus cache. Secondly, we will investigate hybrid approaches by combining geometrical bus interconnection with circuit switched routers for better scalability. Thirdly, we anticipate to use machine learning to capture parallel application affinity for selection of geometrical bus interconnect configurations and use any locality of memory references in the program that may benefit better performance from group connections. Finally, we plan to prototype the system on one or more FPGAs and assess the overall energy efficiency.

ACKNOWLEDGEMENTS

This work was supported in part by Army Research Office HBCU/MSI contract number W911NF-13-1-0133 entitled: "Exploring high performance heterogeneous computing via hardware/software co-design.

REFERENCES

[1] Ann Steffora Mutschler, "Big shift in multi-core design," *Semiconductor Engineering*, 2019. [Online] Available: https://semiengineering.com/big-shift-in-multi-core-design.
[2] S. Le Beux, P. V. Gratz, I. O'Connor, "Guest editorial: Emerging technologies and architectures for many core computing part 1: Hardware techniques," in *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 2, pp. 97-98, 2018,
[3] Süleyman Savas, Zain Ul-Abdin, Tomas Nordström, "A framework to generate domain-specific many core architectures from dataflow programs," *Microprocessor and Micro Systems*, Elsevier Publication, vol. 72, p. 102908, 2020.
[4] Chenggang Lai, Xuan Shi, Miaoqing Huang, "Efficient utilization of multi-core processors and many-core co-processors on supercomputer beacon for scalable geocomputation and geo-simulation over big earth data," *Journal on Big Earth Data*, pp. 65-85, 2018.
[5] Mark Asch, Terry Moore, Rosa M. Badia, "Big data and extreme-scale computing: Pathways to convergence-toward a shaping strategy for a future software and data ecosystem for scientific inquiry," *International Journal of High Performance Computing Applications*, vol. 32, no. 4, pp. 435-479, 2018.
[6] A. Rafiev, M. A. N. Al-Hayanni, F. Xia, R. Shafik, A. Romanovsky, A. Yakovlev, "Speedup and power scaling models for heterogeneous many-core systems," in *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 3, pp. 436-449, 2018.

[7] I. Loi, A. Capotondi, D. Rossi, A. Marongiu, L. Benini, "The quest for energy-efficient I$ design in ultra-low-power clustered many-cores," in *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 2, pp. 99-112, 2018.
[8] Ola Surakhi, Mohammad Alkhanafseh, Sami Sarhan, "A survey on parallel multicore computing: Performance and improvement," *Advances in Science Technology and Engineering Systems Journal*, vol. 3, no.3, pp. 152-160, 2018.
[9] John Keller, "Wanted: Cyber-hardened high-performance embedded computing, artificial intelligence (AI), machine learning," *Military & Aerespace Electronic*, June 2020. [Online]. Available: https://www.militaryaerospace.com/computers/article/14176895/cyber-embedded-computing-artificial-intelligence-ai.
[10] W. Shi, G. Pallis, Z. Xu, "Edge Computing [Scanning the Issue]," in *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1474-1481, 2019
[11] Maurizio Capra, Riccardo Peloso, Guido Masera, Massimo Ruo Roch, Maurizio Martina, "Edge Computing: A survey on the hardware requirements in the internet of things world," *Future Internet*, vol. 11, no. 4, pp. 1-25, 2019.
[12] P. Hijma, C. J. H. Jacobs, R. V. Van Nieuwpoort, H. E. Bal, "Cashmere: heterogeneous many-core computing," *2015 IEEE International Parallel and Distributed Processing Symposium*, pp. 135-145, 2015.
[13] S. J. Hollis, C. Jackson, P. Bogdan, R. Marculescu, "Exploiting emergence in on-chip interconnects," in *IEEE Transactions on Computers*, vol. 63, no. 3, pp. 570-582, 2014.
[14] G. Sievers *et al.*, "Evaluation of interconnect fabrics for an embedded MPSoC in 28 nm FD-SOI," *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1925-1928, 2015.
[15] Daniel Sanchez, George Michelogiannakis, Christos Kozyrakis, "An analysis of on-chip interconnection networks for large-scale chip multiprocessors," *ACM Transactions on Architecture and Code Optimization*, vol. 7, no. 1, pp. 1-28, 2010.
[16] A. Naik, T. K. Ramesh, "Efficient Network on Chip (NoC) using heterogeneous circuit switched routers," *2016 International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA)*, pp. 1-6, 2016.
[17] A. K. Lusala, J. Legat, "A hybrid router combining SDM-based circuit swictching with packet switching for on-chip networks," *2010 International Conference on Reconfigurable Computing and FPGAs*, 2010, pp. 340-345.
[18] Douglas R. Melo, Cesar A. Zeferino, Luigi Dilillo, Eduardo A. Bezerra, "Maximizing the inner resilience of a network-on-chip through router controllers design," *Sensors Journal*, vol. 19, no. 24, pp. 1-23, 2019.
[19] Khyam Parane, Prabhu Prasad, Basavaraj Talawar, "LBNoC: Design of low-latency router architecture with lookahead bypass for network-on-chip using FPGA," *ACM Transactions on Design Automation on Electronic Systems*, article NL. 9, 2020.
[20] F. Alazemi, A. AziziMazreah, B. Bose, L. Chen, "Routerless network-on-chip," *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 492-503, 2018.
[21] Vivek Tiwari, Kavita Khare, "Efficient configurable crossbar switch design for NoC," *International Journal of Scientific and Technology Research*, vol 8, issue. 11, pp. 959-964, 2019.
[22] Mohamed Zahran, "Heterogenous computing: Hardware and software perspectives," New York, NY: Association for Computing Machinery, 2019.
[23] E. Alareqi, T. Ramesh, K. Abed, "Functional heterogeneous processor affinity characterization to big data: towards machine learning approach," *2017 International Conference on Computational Science and Computational Intelligence CSCI*, 2017, pp. 1432-1436.
[24] C. Ababei, M. G. Moghaddam, "A survey of prediction and classification techniques in multicore processor systems," in *IEEE Transactions on Parallel and Distributed*, 1184-1200, 2019.
[25] Alizera Monemi, Jia Wei Tang, Maurizio Palesi, Muhammad N. Marsono, "ProNOC: A low latency network-on-chip based many-core system-on-chip prototyping platform," *Microprocessors and Microsystems*, vol. 54, pp. 60-74, 2017.
[26] Rakesh Pandey, Aryabartta Sahu, "Performance and area trade-off of 3D-stacked DRAM bases chip multiprocessor with hybrid interconnect," *IEEE Transactions on Emerging Topics in Computing*, 2019.
[27] Arslan Munir; Ann Gordon-Ross; Sanjay Ranka, "Parallelized benchmark-driven performance evaluation of symmetric multiprocessors and tiled multicore architectures for parallel embedded systems," in *Modeling and Optimization of Parallel and Distributed Embedded Systems*, IEEE, pp.269-286, 2016.
[28] R. Kumar, V. Zyuban, D. M. Tullsen, "Interconnections in multi-core architectures: understanding mechanisms, overheads and scaling," *32nd International Symposium on Computer Architecture (ISCA'05)*, pp. 408-419, 2005.
[29] Tirumale Ramesh, S. Ganesan, "A reconfigurable reduced-bus multiprocessor interconnection network," *Fifth Distributed Memory Computing Conference*, 1990, pp. 719-724.
[30] T. N. Mudge, J.P. Hayes, G.D. Buzzard, "Analysis of multiple-bus interconnection networks," *Journal of Parallel and Distributed Computing*, vol. 3, no. 3, pp. 328-343, 1986.
[31] Lang. T, Valero. M, Fiol M. A, "Reduction of connections for multibus organization," *IEEE Transactions on Computers*, vol. c-32, no. 8, pp.707-716, 1983.
[32] Donald Winsor, Trevor Mudge, "Analysis of bus hierarchies for multiprocessors," *ACM SIGARCH Computer Architecture News*, vol. 16, no. 2,1988.
[33] Zahra Sadat Ghandrizp, Esmaeil Zeinali, "A new routing algorithm for a three-stage CLOS interconnection network," *International Journal of Computer Science Issues*, vol. 8, no. 5, pp. 309-313, 2008.

BIOGRAPHIES OF AUTHORS

Tirumale Ramesh received his BE degree in electrical engineering from Bangalore University, India in 1975, an MSEE in VLSI area from Mississippi State University in 1983 and the PhD degree in computer engineering from Oakland University, Michigan in 1993. He is currently supporting Jackson State University as an academic research consultant where he previously served as a Senior Research Associate. Previously he served as a tenured professor of computer engineering in academia. He was a corporate tech fellow at Boeing and a senior engineer at IBM. Ramesh has numerous US and foreign patents and published widely. As a senior member of IEEE, he has served in leadership roles for IEEE conferences and IEEE computer society and received several professional awards. He currently serves as an adjunct professor at George Washington University in DC. His current research interests include heterogeneous computing, reconfigurable computing and interconnection, artificial intelligence (AI)/machine learning.

Khalid Abed is a Tenured Professor in the Department of Electrical & Computer Engineering and Computer Science at Jackson State University (JSU). His research interests include high performance heterogeneous/reconfigurable computing (HPRC/HPHC), Edge Computing, Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL). He received his B.S., M.S., and Ph.D. in Electrical Engineering from Wright State University in 1995, 1996, 2000, respectively. He has published extensively in IEEE journals and conferences and is a technical reviewer for several IEEE journals and conferences. Dr. Abed is a Senior Member of IEEE, IEEE Computer Society. He co-authored several patent submissions in the HPHC/HPRC, areas. He has received funding from sources including the NSF, the DoD, and the Army Research Office. He has received about $3M in grants for HPHC/HPRC education and research.