XPoint Cache: Scaling Existing Bus-Based Coherence Protocols for 2D and 3D Many-Core Systems

Ronald G. Dreslinski, Thomas Manville, Korey Sewell, Reetuparna Das, Nathaniel Pinckney, Sudhir Satpathy, David Blaauw, Dennis Sylvester, Trevor Mudge

Department of Electrical Engineering and Computer Science
The University of Michigan, Ann Arbor – Ann Arbor, MI, 48109

ABSTRACT

With multi-core processors now mainstream, the shift to many-core processors poses a new set of design challenges. In particular, the scalability of coherence protocols remains a significant challenge. While complex Network-on-Chip interconnect fabrics have been proposed and in some cases implemented, most of industry has slowly evolved existing coherence solutions to meet the needs of a growing number of cores. Industries' slow adoption of Network-on-Chip designs is in large part due to the significant effort needed to design and verify the system. However, simply scaling bus-based coherence is not straightforward either because of increased contention and latency on the bus for large core counts.

This paper proposes a new architecture, *XPoint*, which does not need to modify existing bus-based snooping coherence protocols to scale to 64 core systems. *XPoint* employs interleaved cache structures with detailed floorplaning and system analysis to reduce contention at high core counts. Results show that the *XPoint* system achieves, on average, a 28× and 35× speedup over a single core design on the Splash2 benchmarks for a 32 and 64 core system respectively (a 1.6× improvement over a 64 core conventional bus). *XPoint* is also evaluated as a 3D stacked system to reduce further bus latency. Results show a 29× and 45× speedup for 32 and 64 core systems respectively (a 2.1× improvement over a 64 core conventional bus and within 8% of the speedup of a 64 core system with an ideal interconnect). Measurements also show that the *XPoint* system decreases bus contention of a 64 core system to only 13% higher than that of an 8-core design (a 29× improvement over a 64 core conventional bus).

Categories and Subject Descriptors

B.0 [**Hardware**]: General; C.0 [**Computer Systems Organization**]: General

Keywords

3D Integration, Bus Design, Cache Coherence, Interconnect

PACT'12, September 19–23, 2012, Minneapolis, Minnesota, USA.

1. INTRODUCTION

Shared memory multi-core processors have become mainstream. As core counts in these systems continue to rise, it becomes important to support scalable cache coherence protocols. While complex systems have been proposed for directory based Network-on-Chip (NoC) interconnects, they have only been adopted in a limited number of commercial systems [3]. Meanwhile, many manufacturers continue to run on evolved versions of older snooping based designs, such as the rings in the IBM Power 4/5 series [41] and the Intel Sandybridge [36]. Other systems have taken small steps away from snooping protocols by adopting crossbars to interconnect the chip such as Sun's Niagara 2 [26] and IBM's BlueGene/Q [24].

The reluctance of industry to move to NoC designs is in part driven by the difficulty of system verification. There is a significant infrastructure [31] around existing coherence solutions, which makes validating evolutionary coherence solutions much less costly in both engineering effort and time to market. Thus, an interconnect that scales to many-core systems without modifying the underlying coherence protocol is desirable.

In this paper, we propose a new interconnect architecture, *XPoint*, which employs address-interleaved cache architectures to scale unmodified snooping coherence protocol designs to many-core systems as large as 64 cores. Address interleaving of the memory system, coupled with the aid of point-to-point interconnects and optimized layout designs, helps to reduce the contention for shared resources in the system while leaving the coherence protocol *unmodified*. Detailed floorplaning and analysis is done in a 32nm technology which shows that the design can easily scale to 64 cores in a conventional 2D layout—*XPoint 2D*. A 32 core *XPoint 2D* system only increases the bus contention by 9% on average when compared to an 8 core design, and has 10× less bus contention than a 32 core system with a conventional design. A 32 and 64 core *XPoint 2D* system achieves, on average, a 28× and 35× speedup over a single core, which is a 1.6× improvement over a 64 core conventional design.

While the *XPoint 2D* system itself scales to 64 cores, at the higher core counts (32-64) there is still room for improvement. The bus delay of the system at this point is much larger than the 8 core system, and leads to increased bus contention and longer L1 miss latencies. We show that the *XPoint* design extends naturally to 3-dimensions (3D). While 3D integration is itself a speculative technology, recent designs [17, 29] have supported the feasibility of such systems with fabricated test chips. The work by Fick *et al*. [17] demonstrates a 7-layer, 3D-stacked system with 128 ARM cores on 4 layers of CMOS logic and 256*MB* of DRAM on the remaining 3 layers. With optimized floorplans the *XPoint 3D* system can reduce the bus delay of a 64 core *XPoint 2D* system to that of

a 16 core system. This reduction in bus delay shows that an *XPoint 3D* system with 64 cores only increases bus contention by 13% on average over an 8 core system and has 29× less contention than a 64 core conventional design. The *XPoint 3D* also improves the average speedup of the system to 29× and 45× for a 32 and 64 core system, which is a 2.1× improvement over a 64 core conventional design and is within 8% of the speedup of an ideal interconnect.

An important concern for 3D stacked systems is cooling. Thermal analysis shows it is possible to stack 4 layers of *XPoint 2D* systems in a single chip that can be cooled by conventional means.

The main contributions of this paper are:

- An architecture that uses snooping buses with *unmodified* coherence protocols coupled with address-interleaved caches and point-to-point links to reduce shared bus contention in on-die many-core chips
- Detailed floorplaning and spice analysis of the buses and other key components in 32nm
- An extension of the design in a proven 3D technology, including thermal analysis, that reduces the latency of shared buses
- Detailed analysis of the performance of this design from 8 to 64 cores, showing the system readily scales to 64 cores in both 2D and 3D implementations

The rest of the paper is organized as follows: In Section 2 we will motivate the advantage of simply extending existing snooping protocols and illustrates the key barriers. Section 3 will present the *XPoint Cache* architecture. Experimental methodology is presented in Section 4 and the results in Section 5. A brief summary of related work is presented in Section 6. Finally we provide concluding remarks in Section 7.

2. MOTIVATION

Broadcast-based snooping protocols are a common approach to building multicore processors [8]. In the past, there has been considerable effort to retain and scale snooping protocols by adapting them for split transaction buses [18], hierarchical buses [40], and address broadcast trees [11] that provide a "logical bus" ordering. Existing products, like the IBM Power4/5 and the Intel Sandybridge, retain and scale snooping coherence protocols for ring interconnects [36, 41].

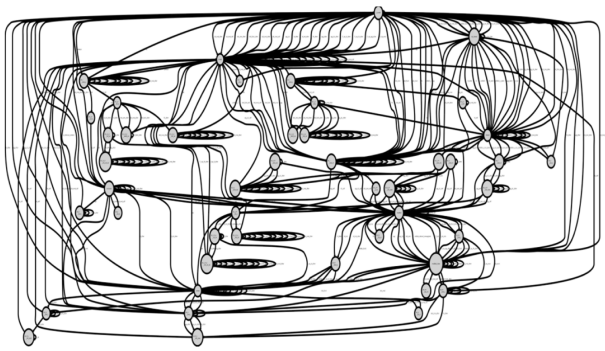


Figure 1: State transition diagram for the MESTI coherence protocol [32] including spit-transactions and race conditions (Containing 47-states). Industry protocols similar to this have an extensive infrastructure to verify state interaction.

While more advanced approaches have been discussed in literature, industry designers have been more comfortable evolving their existing snooping-based system over the years. One of the reasons such effort has been devoted towards continuously scaling and supporting snooping protocols is the prohibitive design and verification complexity of any coherence protocol [31]. While it is possible to formally verify relatively simple state machines at the abstract level [14, 16, 19, 38, 39], verifying the concrete system with detailed interactions is difficult and requires a mixture of pre-silicon [49] and post silicon validation [9, 13, 35, 43]. Although significant efforts have been devoted to the verification of protocols, coherence bugs in commercial processors often go undetected [25]. In fact, the need for simpler verification has even led architects to propose coherence protocols designed explicitly for verification [50].

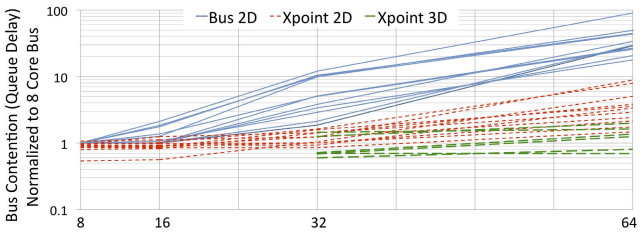


Figure 2: Bus Utilization for 8-64 Cores on the SPLASH2 Benchmarks. Each line represents one benchmark. The conventional bus based system sees a 50-100× increase in contention, while the *XPoint* designs limit this to much lower amounts.

Some designs, such as the AMD Opteron [27] and IBM BlueGene/Q [24], have migrated to a directory based approach for modest core counts. These designs are built on top of either point-to-point interconnects or crossbar based designs. Directory protocols are more complex than snooping, but because the interconnects in these systems are predictable and contain no intermediate buffering within the interconnect itself, the increased verification effort is bounded. Meanwhile, in the research community even more complex directory systems have been proposed. These systems typically employ a Network-on-Chip (NoC) approach to address the scalability of many-core systems, however very few commercial chips have adopted this approach [3].

Although NoC systems scale towards large core counts, transitioning to such a design represents a more radical change than just evolving an existing snooping system with well defined design flow and verification infrastructure. The MESTI [32] protocol for snooping shown in Figure 1 contains 47 states, while the directory protocol used in the *gem5* simulator [6] for an NoC system requires 65 states for the L2, 15 states for the L1, 19 states for the directory, and 3 states for the DMA engine. The product of all the potential states that need to be verified in such systems becomes daunting, particularly when many of these controllers are instantiated multiple times in the system. Compounding this problem is the internal buffering and routing algorithms employed in the NoC which create unpredictable, non-uniform latencies. This, in turn, increases the number of transient states and makes post-silicon verification more difficult as routing algorithms and buffering solutions also need to be verified.

The goal of this work is to extend the snooping bus based coherence protocol to many core systems without modifying the underlying implementation of snooping protocols. However, the increasing bandwidth pressure and latency on the bus as the core count increases provide two significant hurdles to scalability. The proposed *XPoint* design challenges the conventional wisdom that bus-based approaches cannot scale to many-core systems [12]. While

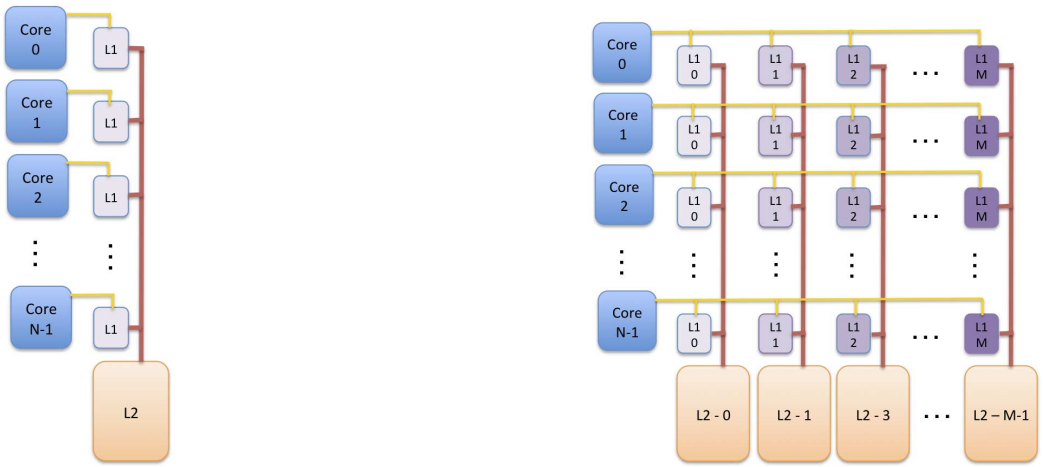


Figure 3: High level view of (a) a conventional bus based architecture, and (b) The *XPoint* architecture. Caches in a vertical column are all assigned to the same address range. No snooping is required between vertical columns. The vertical buses represent independent unmodified snooping busses. The horizontal connections are made with fast point-to-point links.

the *XPoint* design does not scale indefinitely, detailed results show that in a 32nm technology it can easily scale to at least 32 cores in a conventional 2D layout and to at leaset 64 cores when 3D integration techniques are employed.

To better understand the barriers associated with scaling snooping coherence on a bus based interconnect Figure 2 shows the bus contention rate for the SPLASH2 benchmarks at various core counts. The methodology and configuration parameters are presented later in Section 4. The bus contention rate, approximated by the average L1 queueing delay, is normalized to an 8 core system. Each line in the diagram represents one SPLASH2 benchmark. As the core count is increased, the bus contention of traditional designs (solid lines) grows dramatically. At a core count of 64, an 8× increase in the number of cores, the benchmarks experience 20-100× increases in bus contention. To extend unmodified snooping coherence a two pronged approach will be necessary that attacks the contention on the bus as well as the bus latency. The *XPoint 2D* system (short dashed lines) employs techniques to reduce contention, as a result 32 cores only increase the contention on the bus by 1.6× in the worst case. As the *XPoint 2D* system is scaled further to 64 cores, the increased delay on the bus pushes contention higher to 8.9× that of the baseline 8 core system. The *XPoint 3D* system (long dashed lines) tackles the delay scalability of the bus using 3D integration. At 64 cores the bus contention for *XPoint 3D* is, on average, only 13% higher than that of an 8-core system, even though it supports 8× the number of cores.

3. CROSSPOINT CACHE ARCHITECTURE

In the following subsections we will present architectures which overcome the two major hurdles of scaling bus based snooping protocols. First we tackle the bus contention with an architecture called *XPoint 2D*. *XPoint 2D* uses multiple cache line interleaved buses supporting unmodified coherence protocols. Results in Section 5 will show this system scales readily to 64 cores. The *XPoint* architecture naturally extends to 3D allowing further scaling by overcoming the increasing bus latency—*XPoint 3D*. The section concludes with thermal analysis of *XPoint 3D* and the applicability of the *XPoint* to other bus-based systems.

3.1 XPoint 2D - Overcoming Bus Contention

To understand the *XPoint 2D* architecture a brief review of a traditional bus based system follows. Figure 3(a) shows a high level view of a traditional bus based system. Each core is connected to a private L1, each of the L1's are connected to a shared snooping bus, which in turn is connected to the L2. As the number of cores grows, the contention on the shared bus becomes a bottleneck in the system. The *XPoint 2D* architecture addresses this bottleneck while leaving the coherence protocol unmodified.

Our baseline builds on memory interleaving techniques developed in the 1980's to address interconnect congestion in board level multi-processors [46]. However, integration levels and pin constraints limited their practicality. We show that with new architectural techniques these designs are now practical for single chip implementations, where point-to-point connections, higher cache associativites, and a greater number of wires are available. In addition we explore the implications of such designs on coherence mechanisms.

The *XPoint 2D* system exploits address-interleaving to reduce bandwidth pressure on the shared bus fabric. Figure 3(b) shows the logical layout of our proposed architecture. To scale the coherence protocol to n cores, each core's last level of private cache (L1 in this diagram) is split into m equal slices. The core can access all the m slices via direct point-to-point channels. By using point-to-point connections, the speed of the interconnect can be much faster than a traditional bus which requires bi-directional repeaters and arbitration units. The shared L2 cache is also split into m equal slices. A bus connects a vertical column of private cache slices (n L1 slices) to a shared L2 cache slice, all of which map to the same addresses. This isolates the coherence traffic separately on each vertical bus from the other vertical buses in the system. The result is a multi-bus system with simple, unmodified coherence that reduces contention and increases bandwidth.

3.1.1 Architectural Impacts

The floorplan of the *XPoint 2D* system is shown in Figure 4. The introduction of point-to-point links from the core to the L1 increases the access latency. By keeping the connections point-to-

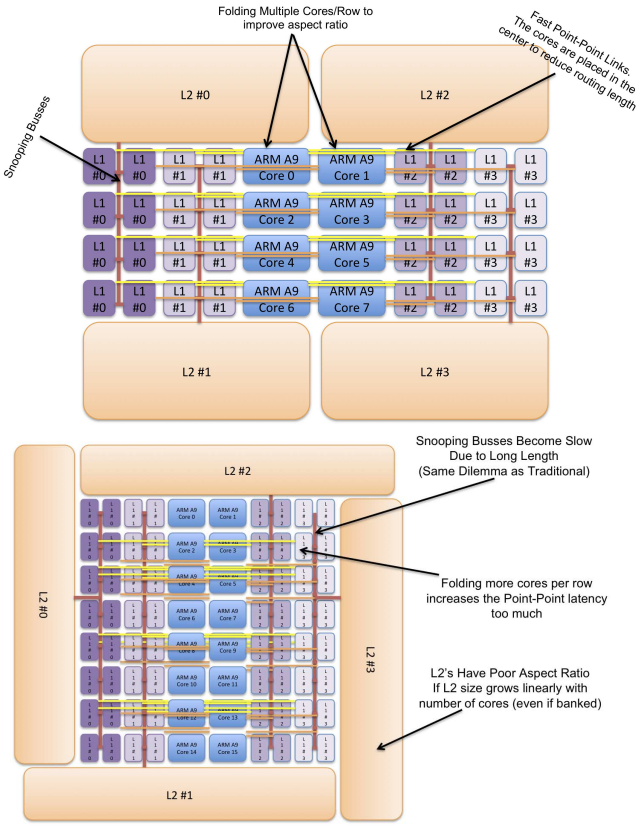


Figure 4: Floorplan view of a *XPoint 2D* system for 8 and 16 cores.

point, rather than bus based, the system can communicate to/from the L1 slices in one additional cycle. To help hide this new latency a 4 cache line, fully-associative L0 buffer is inserted in the core. The L0 and L1 are accessed in parallel, so hits in the L0 take a single cycle, and hits in the L1 take two cycles. Coherence of the L0 is maintained by keeping a copy of the tags in the L1 slices. When snoops occur on addresses contained in the L0 that must be invalidated, the L1 can signal the L0 via a point-to-point invalidation wire. We will show with results, that the increase in L1 access time is offset by the increased bandwidth and reduced contention of the *XPoint 2D* system in Section 5.

Splitting the cache into slices has the potential to produce different cache miss/hit behavior. For the *XPoint* architecture we propose smaller associativities for slices. The lowest bits of the set index are used to determine address mappings to the slices. In addition, because the accesses are split across the buses, prefetch mechanisms should be located at the core/L0 to observe the entire pattern.

3.1.2 Coherence and Consistency

In Figure 3(b) each vertical bus is an *unmodified* snooping based interconnect, and because each vertical bus is independent of the other vertical buses (different address ranges), no snooping is needed for caches in the horizontal direction. By splitting the bus into m slices, the total bandwidth to memory is m-times larger, and the contention on the bus is $1/m^{th}$ that of a traditional design.

Although the coherence mechanism remains unmodified, there may still be consistency issues. Coherence is an ordering of requests to the same block, so splitting the bus into slices means there is no ordering constraints between vertical buses. Consistency on the other hand is an ordering of requests to different addresses, and therefore requires some guarantees across vertical slices. However, for x86 systems which support a form of processor consistency, the consistency can be maintained by the store buffer in the core itself, insuring that all stores are drained from the processor in program order. More constrained consistency models will require the processor to properly order requests to each of the slices.

3.1.3 Interleaving Mechanism

As mentioned earlier, by splitting the bus resource there is the potential for uneven utilization of the vertical slices. For uniform-random traffic the bus utilization is even. However, if the system is split into four slices, a program with an access pattern where the stride is a harmonic of 4 will access only a subset of slices creating an uneven utilization.

To address these concerns a more complex mapping scheme can be used to determine the slice index of the system. In the results section we present a system with simple address interleaving on low order bits and show that for all but 3 benchmarks utilization is not significantly impacted. For the 3 benchmarks impacted by the interleaving, they still outperform the baseline bus system, however the optimal solution is a system with only 2 or 4 slices, instead of 8. We leave more detailed analysis of other interleaving hash functions as future work.

3.1.4 Remaining Scaling Limitations

While the *XPoint 2D* system addresses the increased contention on the shared bus, it does not address the increased latency of the bus in the system. Figure 4(b) shows the *XPoint 2D* architecture scaled to 32 cores. This system still has several limits on ultimate scalability. First, the bus length grows linearly with the number of cores, the same obstacle faced by a traditional system. Second the aspect ratios of the L2 cache become large, meaning that the cache banks are spread out which increases latency and power consumption.

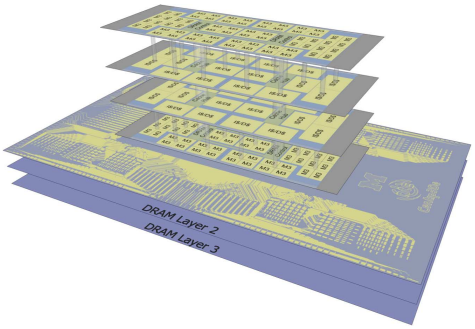
3.2 XPoint 3D - Overcoming Bus Latency

The *XPoint* architecture naturally extends to 3D allowing further scaling by overcoming the increasing bus latency of the *XPoint 2D* system. The following subsections will discuss the 3D technology used for our study and present the *XPoint 3D* architecture.

3.2.1 3D Integration Technology

There are many techniques for 3D integration. Simple techniques such as face-to-face bonding allow simple stacks of 2 chips while more complicated techniques that use wafer thinning allow for larger stack systems. For our analysis we use figures from a 3D integration technology by Tezzaron [23]. Their technique is a via-first, back-end-of-line integration technology and has been demonstrated in several test chips [17,29]. In the system by Fick *et al*. [17] the Tezzaron technology is used to stack four layers of CMOS logic on top of three layers of DRAM. Figure 5 (left) shows a diagram of the system implemented by Fick *et al*. Figure 5 (right) shows a cross section [29] of the through-silicon-via (TSV) technology from Tezzaron. The layers are thinned to less than 12 microns, and the TSV's themselves are less than 6 microns thick. The size of the TSV's are 1.4 square microns, and can be placed with a density of 62,000 TSV's per square mm.

The resistance ($<.35\Omega$) and capacitance ($2fF$) of these TSV's are extremely small compared to other 3D integration technologies. They allow for extremely fast and short connections between lay-



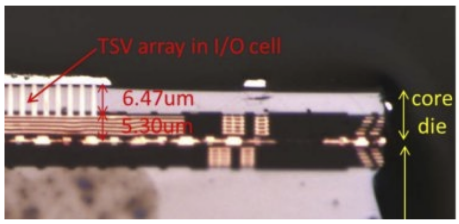


Figure 5: Top level view of the Centip3De 7-layer 3D system [17] built on Tezzaron 3D stacking technology and a cross section of the same process on the 3D-MAPS system [29]. Note the TSV's are only 6.47 microns deep and the wafer is thinned to less than 12 microns which is important for reducing thermal resistance and RC delays.

ers. In fact, in a 4 layer stack the length of a TSV running the whole distance of the stack is $<$50 microns. This allows the *XPoint 3D* system to create buses that run in the 3rd dimension that add minimal latency to the 2D bus.

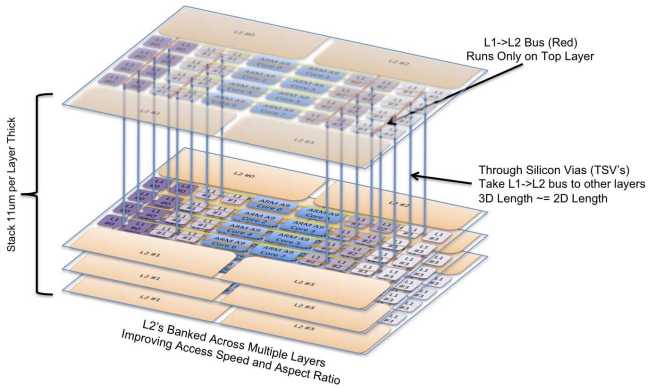


Figure 6: Diagram of the *XPoint 3D* design.

3.2.2 XPoint 3D Architecture

Figure 6 shows the proposed *XPoint 3D* system. The coherent buses are run only along the top layer, and connections to L2's at lower levels are made with TSV's which are 11 microns long per layer. This pitchfork layout creates a 3D bus that is approximately the same length as the bus on the top layer. This means that the effective length of the bus is halved when the system contains 2-layers, and quartered when spread across 4-layers. To a first order the latency of the bus grows linearly with $cores/layers$ instead of linearly with $cores$ as it does in a 2D system. In addition the L2 cache is banked across multiple layers to improve the aspect ratio of the cache itself, thereby improving access times and power consumption of the L2.

3.2.3 Thermal Constraints

As with any 3D chip design, thermal constraints can be a matter of concern. To verify the system operates in a thermal region that can be cooled by conventional solutions, we perform analysis of the system with the HotSpot 5.1 [42] simulator. The thermal characteristics of the Tezzaron process were modeled in HotSpot and peak power draw numbers were used for the core. Power numbers

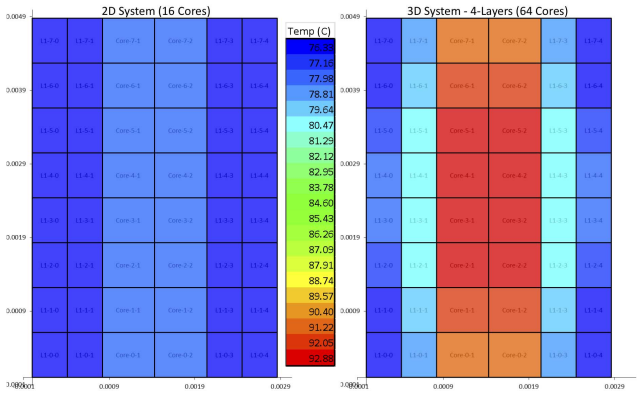


Figure 7: HotSpot simulation of the *XPoint* system on 1 layer,16 core system and across a 4 layer, 64 core system. The peak temperature of the chip is 92 degrees Celsius on 4-layers.

for the Cortex-A9 were based on published data [2] and scaled to 32nm. Figure 7 shows the simulated system for a single layer and a 4-layer stack. The low power design of the Cortex-A9 processor helps to make stacking feasible. The peak temperature of the 4-layer system reaches 92 degrees centigrade, well within conventional cooling solutions. Had the thermal profile been more of an issue, designs where cores are placed on the left side of the L1's on even numbered layers and the right side of the L1's on odd numbered layers could be used to reduce the thermal profile at the expense of more power to drive the longer point-to-point links. The HotSpot analysis did not consider the thermal dissipating characteristics of the TSV's, which would have further reduced the peak temperature. More details on the component models and methodology can be found in Section 4.

3.3 Other Opportunities

This architectural design was done with a system containing private L1 caches and shared L2 caches. Many commercial systems implement private L2 caches and shared L3 caches. The coherence protocol for these systems is different, however the techniques of interleaving the system will also work in such designs without modification of the coherence protocol. In such systems the L2 caches would be interleaved the interleaving would be done on the

L2 caches and the L1 could remain coupled with the processor in either interleaved slices, or as a single L1.

4. EXPERIMENTAL METHODOLOGY

4.1 Component Models

The following sections will describe the models used for simulation. A summary of the components area and speed can be found in Table 1, details on how each value was arrived at are found in the corresponding sub-section.

Table 1: Component Areas and Speeds in 32nm

	Area/Length	Speed/Latency
ARM Cortex-A9	$0.38mm^2$	$1.25GHz$
L1 (64kB I&D) per core	$0.61mm^2$	1 cycle
L2 (256 kB) per core	$1.33mm^2$	8 cycles
Coherent Bus	$0.31mm/core$	160 ps/mm

Table 2: 2D Floorplan Sizes and Bus Speeds

Num. of Cores	Die Size	Bus Length	Bus Speed	Total L2 Size
8	$19mm^2$	$2.48mm$	$>1.25\ GHz$	$2MB$
16	$38mm^2$	$4.96mm$	$1.25\ GHz$	$4MB$
32	$78mm^2$	$10.1mm$	$\sim 630\ MHz$	$8MB$
64	$156mm^2$	$20.1mm$	$\sim 315\ MHz$	$16MB$

Cores: The system is based on ARM Cortex-A9 cores. Area and power estimates are drawn from ARM data sheets [2] and scaled to 32nm. The core area in 32nm is $0.38mm^2$, the speed is $1.25GHz$, and the power is $0.5W$.

Caches: Cache sizes and speeds are obtained from a comercial memory compiler. The total L1 cache size per core is $64kB$ for both I- and D-Cache. The cache area is $0.6mm^2$. The L1 access time is 1 cycle for the bus based system and 2 cycles in the *XPoint* cache system due to the point-to-point link latency. To hide this latency in the *XPoint* cache system a 4 cache line, fully-associative buffer is added as a L0 next to the core. The L0 and L1 are accessed in parallel. The associativity of the entire L1 is 4-way. When the system is interleaved, the associativity of each slice is divided. When the system is split into 8-banks, the associativity is 1-way per 8 bank caches. The L2 cache size is $256kB$ per core, 16-way associative. The die area of the L2 is $1.33mm^2$ per core, and the access latency is 8 cycles. The L2 size and associativity are divided among interleaved slices in the same manner as the L1.

Busses: Bus delays and power estimations were obtained using a 13-layer metallization stack from an industrial 32nm process. In this metallization stack there are five *1X*, three *2X*, two *4X*, two *8X*, and one *16X* metal layers. The *1X* metal layers and one of the *2X* metal layers are reserved for local routing (within the core/-cache). The *8X* and *16X* metal layers are reserved for power and clock routing. That leaves two *2X* and two *4X* layers for global routing. Bus delays were calculated using wire models from the design kit and calculated using SPICE including wire parasitics, repeaters, and worst case cross-coupling capacitance of neighboring wires and metal layers. Repeater placement is done using optimally interleaved spacing as described by Ghoneima and Ismail [20].

The L1 uni-directional point-to-point links can achieve a speed of $54ps/mm$ in double spaced *4X* metal, permitting a delay of less than one clock cycle to/from the L1 banks. The L2 snooping bus requires more complicated bi-directional repeaters and achieves a maximum speed of $160ps/mm$ in double spaced *4X* metal. Assuming the L1 caches in the traditional bus based system are mirrored across the snooping bus to reduce wire length, the bus length is $0.31mm$ per core (about half the length of a core due to mirroring). Table 2 shows the achieved speeds of the bus for each core count. For the 8 and 16 core system, the bus can operate at the speed of the cores ($1.25GHz$), for larger systems the bus is slower.

Table 3: *gem5* Simulation Parameters

Component	Bus Based System	*XPoint System*
Processor	ARM Cortex-A9, 1.25 GHz, 2-Wide, 56 Physical Registers	
Cache Block Size	64 Bytes	
L0 Cache	None	4-entry, fully associative, 1-cycle
L1 Cache	64kB Split I and D Caches	
	4-way Associative, 1-Cycle	Div. by number of Slices Associativity 4-way/#Slices, 2 -cycle
L2 Cache	256 kB per Core, 16-way, 8-cycle	
Coherent Bus	312-1250 MHz, 64Bytes	
Main Memory	2GB, 50 Cycle Latency	

4.2 Simulator & Benchmarks

We evaluate the *Bus 2D* (conventional bus), *XPoint 2D*, and *XPoint 3D* architectures using the *gem5* full-system simulator [6]. The *gem5* simulator was extended to accommodate the *XPoint* interleaved architecture. No modifications to the coherence protocol were necessary. The *gem5* simulator accurately tracks data throughout the memory system. Because the data is accurately passed through the system, any bugs in the coherence protocol would manifest themselves as errors while simulating. Table 3 details the simulation parameters for the studies. To account for non-determinism in threaded workloads, we randomly perturb memory access latencies and run multiple simulations to arrive at stable runtimes following the approach described by Alameldeen *et al*. [1]. A comparison to an ideal interconnect is presented in Table 4, which simulates an interconnect with infinite bandwidth and zero-contention. Results show the *XPoint* system achieves speedups within 8% of ideal, obviating the need for a comparison to an NoC based system.

We use benchmarks from the SPLASH2 [48] suite to test the systems. The SPLASH2 benchmarks are of particular interest for the study of on-chip interconnects as they have many sharing and data migration patterns. This is illustrated in the work of Barrow-Williams *et al*. [4].

5. RESULTS

In the following sub-sections we will present the results. First the *XPoint 2D* system will be analyzed against a traditional bus based system, showing the decreased bus contention and improved scalability of the system across various numbers of *XPoint* slices. Next results will be presented showing how the improvement of the bus latency impacts the *XPoint 3D* system to take scalability even further. Then an analysis of the speedups of the best configurations for all 2D and 3D systems will be presented.

5.1 Reducing Bus Contention - XPoint 2D

Analysis of the *XPoint 2D* system is made based on three key metrics: Runtime/Speedup, Bus Utilization, and Bus Contention. The bus contention is plotted in Figure 2, and a description of the results is presented at the end of Section 2. Figure 8 presents the other key metrics. The plots in Figure 8 show, on a log-log plot, the normalized runtime (solid lines) and bus utilization (dotted lines).

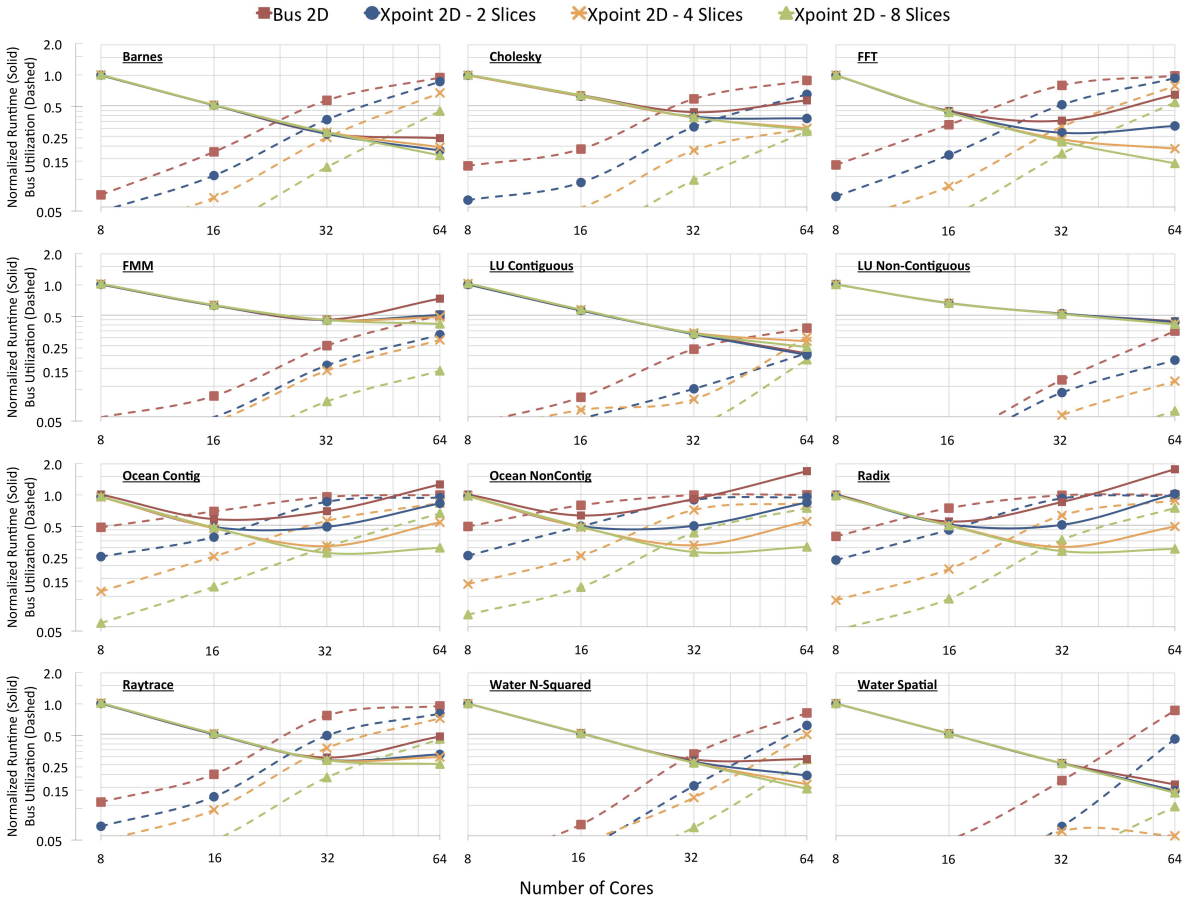


Figure 8: Runtime (solid lines) and Bus Utilization (dotted lines) plotted across core counts. A straight line for runtime represents ideal scaling of the benchmark. As the core count is increased the contention/utilization of the bus is increased, this leads to non-ideal speedup. For the *Bus 2D* system many benchmarks do not scale well beyond 16 cores. The *XPoint 2D* system improves the scalability to higher core counts.

The ideal scaling of these benchmarks would be a straight line (for runtime) where the slope is determined by the efficiency of the parallel scaling in the application itself (workload imbalance and/or ratios of serial to parallel code). Bus utilization is a metric from 0 to 1, describing how often the bus is utilized. The bus utilization numbers saturate for the *Bus 2D* systems for most benchmarks around 32 cores. This saturation correlates with the dramatic increase in bus contention plotted in Figure 2. As the bus utilization and contention increase, the overall runtimes of the benchmarks increase. Eventually the increase in contention outweighs the performance gains of more cores, yielding an optimal number of cores for the system. For the *Ocean Contig*, *Ocean NonContig*, and *Radix* benchmarks the optimal point for a *Bus 2D* system is 16 cores. Beyond this point the runtime increases, even with the addition of more cores. Some benchmarks scale slightly better to 32 cores. There are only two benchmarks that scale to 64 cores on a conventional system. The first is *Water Spatial* where the working set fits in the L1 cache. For the second, *LU Non-Contig*, the runtime line is straight but with a small slope due to workload imbalance. Giving this workload 8× the number of cores only results in a 2× speedup.

The *XPoint 2D* architecture targets benchmarks with scalability limitations due to contention. The plot shows the effects of different degrees of slicing. For the benchmarks whose *Bus 2D* line is not straight, the *XPoint* cache reduces bus contention and lowers the runtime at high core counts—making the line straighter. For many benchmarks, like *Water N-Squared* the *XPoint 2D* system brings the line back to linear at 64 cores. However, for some benchmarks, like *Ocean Non-Contig*, the *XPoint 2D* system is only effective up to 32 cores. This limit occurs because the bus latency is increasing as the core count is increased, reducing the total bandwidth of the system effectively creating more contention.

5.1.1 Cache Slice Fairness

As mentioned in Section 3.1.3, by splitting the cache, unfairness can occur in the allocation of cache lines to slices. This will result in higher L1 miss rates, as less of the cache is being utilized. This effect is particularly evident in strided accesses. To quantify the impact of the cache slicing mechanism we plot a fairness metric in Figure 10. The metric is derived by dividing the number of accesses to the most accessed bank by the number of accesses to the least accessed bank. A fairness of 1 indicates that the cache lines are divided evenly amongst the slices. For all but 3 benchmarks the fairness is good, around 1.2. However, for *Barnes*, *FMM*, and *LU* there is less balance. This corresponds to designs that favor less slicing for optimal runtime (Figure 8). Overall, there are some benchmarks that show sensitivity to the slicing allocation, however

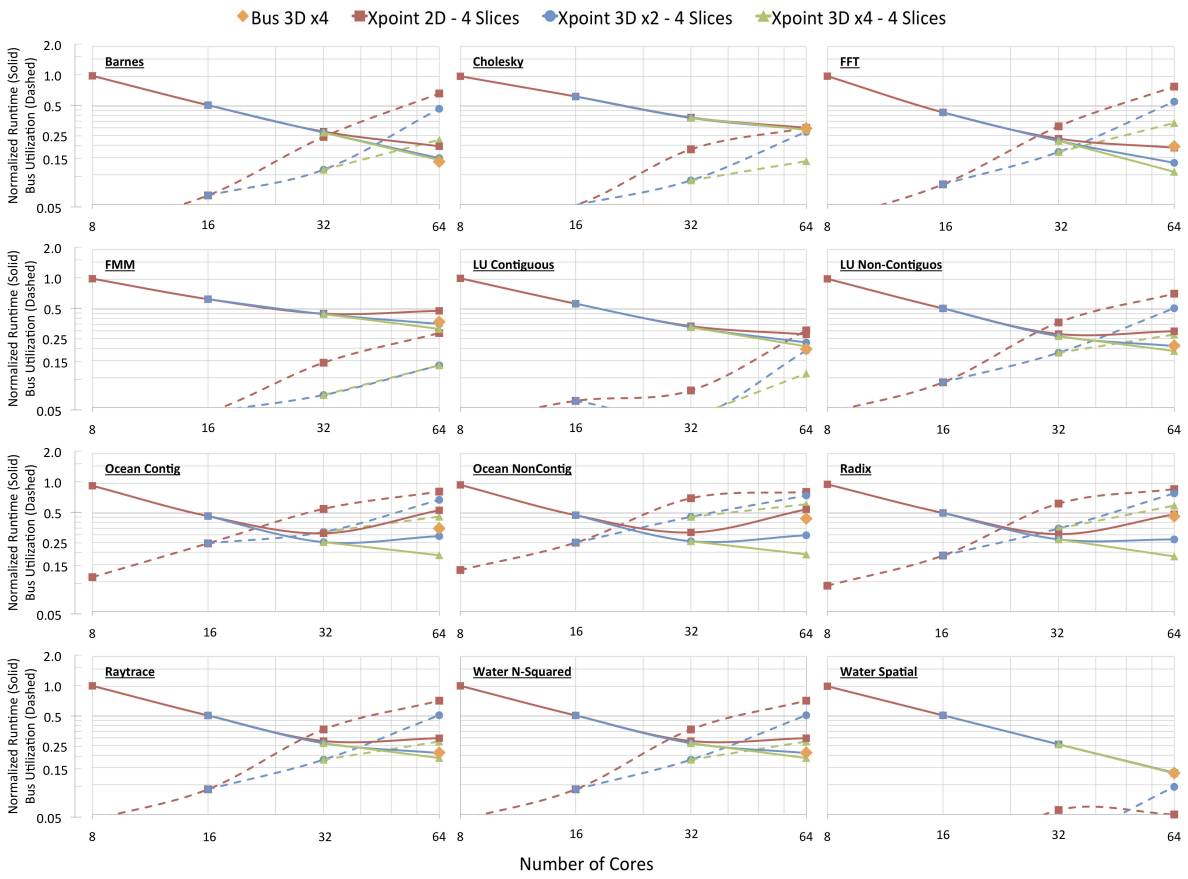


Figure 9: Runtime (solid lines) and Bus Utilization (dotted lines) plotted across core counts. A straight line for runtime represents ideal scaling of the benchmark. All configurations use a cache slicing of 4. *XPoint 3D* improves the scaling compared to the *XPoint 2D*.

the system still shows improvement. We leave exploring more complex hashing functions to balance interleaving for future work.

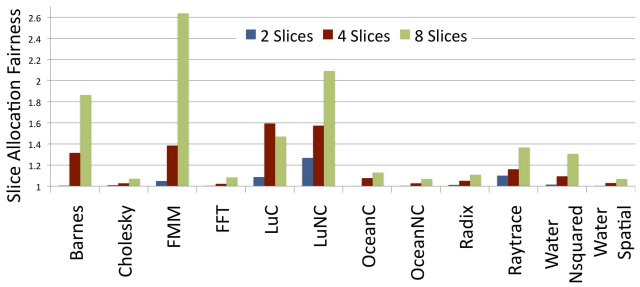


Figure 10: Fairness of slice allocation. Calculated as the total number of accesses to the most accessed cache slice divided by the total number of accesses to the least accessed cache slice. A completely even distribution has a fairness of 1.

5.2 Improving Bus Latency - XPoint 3D

While the results show that the *XPoint2D* system scaled well to 32 cores, there were several benchmarks that still had scaling limitations resulting from increased bus delay. By extending the system in 3D, the bus latencies are decreased and further scaling is possible for these benchmarks. Figure 9 shows runtime and bus utilization on the same axes as before, but for *XPoint 2D* and *XPoint 3D* designs with 4-slices. For the *XPoint 2D* benchmarks that did not scale well—*FMM*, *Ocean Contig*, *Ocean Non-Contig*, *Radix*, and *Water Spatial*—the use of *XPoint 3D* helps reduce the runtime at 64 cores, making the trend closer to linear. On average, the *XPoint 3D* improves performance by 28% over the *XPoint 2D* system at 64 cores.

5.3 Best Configurations

The best performing configurations for each benchmark on each type of system is listed in Table 4. For systems with uneven slice utilization the optimal number of slices is less than 8, which correspond to the same benchmarks shown in Section 5.1.1. The average speedup achieved by the *XPoint 2D* system is 35× , the *XPoint 3D* system further improves that number to 45×. On average the *XPoint 2D* system performs 1.6× better than the bus, it also outperforms the conventional bus in 3D for all but 3 benchmarks. Overall the *XPoint 3D* system improves performance by 2.1× compared to the 2D conventional bus. The *XPoint 3D* system achieves speedups within 8% of an ideal interconnect, showing that both the *XPoint 2D* and *XPoint 3D* take the scalability of bus based systems farther than commonly thought, making them ideal architectures for many-core systems of 32-64 cores.

Table 4: A Breakdown of the best performing parameters on each system for each benchmark. Speedup is presented over a uniprocessor system with 2MB L2. Ideal Interconnect allows infinite requests on the bus simultaneously, with a single cycle latency. Super linear speedups occur for FFT due to the increased L2 size available in the 64 core system. The *XPoint 3D* design is within 8% of the speedup of an ideal interconnect.

	Bus 2D		*XPoint 2D*			Bus 3D			*XPoint 3D*				Ideal Interconnect	
	Cores	Speedup	Cores	Slices	Speedup	Cores	Layers	Speedup	Cores	Slices	Layers	Speedup	Cores	Speedup
Barnes	64	33×	64	8	49×	64	4	58×	64	2	4	**60×**	64	64×
Cholesky	64	18×	64	8	28×	64	4	27×	64	2	4	**29×**	64	30×
FFT	32	22×	64	8	59×	64	4	41×	64	8	4	**83×**	64	86×
FMM	32	17×	64	8	19×	64	4	21×	64	8	4	**25×**	64	39×
Lu Contig	64	38×	64	2	39×	64	4	41×	64	2	4	**41×**	64	42×
Lu NonContig	64	18×	64	8	19×	64	4	20×	64	4	4	**21×**	64	22×
Ocean Contig	16	13×	32	8	30×	64	4	23×	64	8	4	**54×**	64	60×
Ocean NonContig	16	13×	32	8	29×	64	4	18×	64	8	4	**53×**	64	63×
Radix	16	15×	32	8	28×	64	4	17×	64	8	4	**48×**	64	50×
Raytrace	32	27×	64	8	32×	64	4	38×	64	8	4	**43×**	64	52×
Water NSquared	32	28×	64	8	55×	64	4	51×	64	8	4	**59×**	64	59×
Water Spatial	64	50×	64	4	60×	64	4	61×	64	8	4	**62×**	64	62×
Geometric Mean		22×			35×			31×				**45×**		49×

Overall, the results have shown that the *XPoint* architecture scales to 64 core systems with an unmodified bus based snooping protocol. Given the infrastructure already in place for verifying these types of systems, *XPoint 2D* presents a desirable approach to interconnect at 32-64 cores in 32nm. As 3D technology becomes part of the commercial mainstream the *XPoint 3D* systems will further scale bus based designs to higher core counts.

6. RELATED WORK

Our baseline builds on memory interleaving techniques developed in the 1980's to address interconnect congestion in board level multi-processors [46]. However, integration levels and pin constraints limited their practicality. We show that with new architectural techniques these designs are now practical for single chip implementations, where higher cache associativites and a greater number of wires are available. We further differentiate our design from theirs with the use of on-die point-to-point connections from the core to the L1's to improve latency, the addition of L0 buffers to hide increased L1 access time, and 3D integration to reduce bus latency. Many other researchers have looked at multiple bus systems in both interleaved and non-interleaved architectures [5, 10, 15, 44]. However, none of their work explicitly shows how interleaved busses can be used to scale *unmodified* coherence protocols. Recent work by Udipi *et al*. [45] also seeks to scale conventional snooping coherence to many-core systems with the design of a segmented bus and an associated filtering mechanism. Only in passing do they mention interleaved buses and they leave the evaluation of such designs as future work. In addition the *XPoint* system shows how 3D integration can be used to improve the scalability of the bus, an approach not considered in any of these previous works.

There have also been many papers on the topic of 3D integration [7, 28, 34, 47]. While these papers target increased system performance, scaling coherence protocols were not their primary purpose. Other papers on 3D interconnects [30, 33, 37] have extended the research of NoC designs to 3D chips. These papers seek to reduce hop counts in NoC systems by leveraging the 3^{rd} dimension to implement higher radix NoC topologies, such as hyper-cubes and flattened butterflies. The *XPoint 3D* architecture shows that both 2D and 3D stacked systems with conventional snooping coherence can be extended to systems of 64 cores, without the need for complex designs that require extensive verification effort.

Finally there has also been some related work on interleaving main memory [21] and caches [22] for VLIW systems, however none of these designs addresses the scalability of cache coherence protocols or 3D integration.

7. CONCLUSION

Design, verification, and validation of coherence protocols in comercial chips is difficult and expensive. Within industry there is a large amount of infrastructure built around validating current interconnect solutions. While distributed directory architectures represent one approach to scalability, their design is more complex and less familiar compared to current bus-based snooping mechanisms and therefore present a significant hurdle to current validation techniques. On the other hand, simply scaling traditional bus solutions leads to increased contention and latency on the bus.

To address the scalability of existing solutions while leaving the coherence protocol unmodified, this paper proposed the *XPoint* architecture. The *XPoint* system showed that bus-based snooping coherence can scale to 64 core systems by leveraging architectural and layout enhancements designed to combat the degraded speed of long busses and increased contention at large core counts. Overall the *XPoint* system showed a 29× and 45× speedup for 32 and 64 core systems respectively(a 2.1× improvement over a 64 core conventional bus and within 8% of a 64 core system with an ideal interconnect). Measurements also showed that the *XPoint* system decreased the bus contention of a 64 core system to only 13% higher than that of an 8-core design.

8. REFERENCES

[1] A. Alameldeen and D. Wood. Variability in architectural simulations of multi-threaded workloads. In *The Ninth International Symposium on High-Performance Computer Architecture*, Feb 2003.

[2] ARM Ltd. http://www.arm.com/products/processors/cortex-a/cortex-a9.php, 2011.

[3] M. Baron. Tilera's Cores Communicate Better: Mesh Networks and Distributed Memory Reduce Contention Among Cores. *Microprocessor Report*, 2007.

[4] N. Barrow-Williams, C. Fensch, and S. Moore. A communication characterisation of SPLASH-2 and parsec. In *IISWC*, 2009.

[5] J. Bertoni, J.-L. Baer, and W.-H. Wang. Scaling shared-bus multi-processors with multiple buses and shared caches: a

performance study. *Microprocess. Microsyst.*, 16:339–350, September 1992.

[6] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. The M5 simulator: Modeling networked systems. *IEEE Micro*, 26(4):52–60, 2006.

[7] B. Black, M. Annavaram, N. Brekelbaum, J. Devale, L. Jiang, G. H. Loh, D. Mccauley, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb. Die stacking (3D) microarchitecture. In *In Proceedings of MICRO-39*, 2006.

[8] G. Blake, R. Dreslinski, and T. Mudge. A survey of multicore processors. In *Signal Processing Magazine*. IEEE, 2009.

[9] J. F. Cantin, M. H. Lipasti, and J. E. Smith. Dynamic verification of cache coherence protocols, 2001.

[10] M. J. Carlton. Multiple-bus, scalable, shared-memory multiprocessors, 1995.

[11] A. Charlesworth. Starfire: Extending the SMP envelope. In *IEEE Micro*, 1998.

[12] D. Culler, J. Singh, and A. Gupta. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, 1998. The Morgan Kaufmann Series in Computer Architecture and Design.

[13] A. DeOrio, A. Bauserman, and V. Bertacco. Post-silicon verification for cache coherence. In *Computer Design, 2008. ICCD 2008. IEEE International Conference on*, pages 348 –355, oct. 2008.

[14] D. Dill, A. Drexler, A. Hu, and C. Yang. Protocol verification as a hardware design aid. In *Computer Design: VLSI in Computers and Processors, 1992. ICCD '92. Proceedings., IEEE 1992 International Conference on*, pages 522 –525, oct 1992.

[15] M. Dubois. Throughput analysis of cache-based multiprocessors with multiple buses. *Computers, IEEE Transactions on*, 37(1):58 –70, jan 1988.

[16] E. A. Emerson and V. Kahlon. Exact and efficient verification of parameterized cache coherence protocols. In *Correct Hardware Design and Verification Methods (CHARME'03), LNCS 2860*, pages 247–262. Springer, 2003.

[17] D. Fick, R. Dreslinski, B. Giridhar, G. Kim, S. Seo, M. Fojtik, S. Satpathy, Y. Lee, D. Kim, N. Liu, M. Wieckowski, G. Chen, T. Mudge, D. Sylvester, and D. Blaauw. Centip3De: A 3930 dmips/w configurable near-threshold 3D stacked system with 64 ARM Cortex-M3 cores. *To appear in IEEE International Solid-State Circuits Conference, San Francisco, CA, 2012*, 2012.

[18] M. Galles and E. Williams. Performance optimizations, implementation, and verification of the sgi challenge multiprocessor. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, 1994.

[19] S. M. German. Formal design of cache memory protocols in ibm. *Form. Methods Syst. Des.*, 22:133–141, March 2003.

[20] M. Ghoneima and Y. Ismail. Optimum positioning of interleaved repeaters in bidirectional buses. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(3):461 – 469, march 2005.

[21] S. Ghosh, J. Ghosh, and S. Ray. Architecture of configurable k-way c-access interleaved memory. In *Process Automation, Control and Computing (PACC)*, pages 1 –5, july 2011.

[22] E. Gibert, J. Sanchez, and A. Gonzalez. An interleaved cache clustered VLIW processor. *Architecture*, pages 210–219, 2002.

[23] S. Gupta, M. Hibert, S. Hong, and R. Patti. Techniques for producing 3D ICs with high-density interconnect. *White Paper*, 2004.

[24] R. Haring. The IBM Blue Gene/Q Compute chip+SIMD floating-point unit. In *HotChips 23: A Symposium on High-Performance Chips*, 2011.

[25] Intel. Intel core2 extreme processor x6800 and intel core2 duo desktop processor e6000 and e4000 sequence, 2008.

[26] T. Johnson and U. Nawathe. An 8-core, 64-thread, 64-bit power efficient sparc soc (niagara2). In *International symposium on Physical design*, ISPD '07, pages 2–2, New York, NY, USA, 2007. ACM.

[27] C. N. Keltcher, K. J. McGrath, A. Ahmed, and P. Conway. The amd opteron processor for multiprocessor servers. *IEEE Micro*, 23:66–76, March 2003.

[28] T. Kgil, S. D'Souza, A. G. Saidi, N. L. Binkert, R. G. Dreslinski, T. N. Mudge, S. K. Reinhardt, and K. Flautner. PicoServer: using 3D stacking technology to enable a compact energy efficient chip multiprocessor. In *ASPLOS*, pages 117–128, 2006.

[29] D. H. Kim, K. Athikulwongse, M. B. Healy, M. M. Hossain, M. Jung, I. Khorosh, G. Kumar, Y.-J. Lee, D. L. Lewis, T.-W. Lin, C. Liu, S. Panth, M. Pathak, M. Ren, G. Shen, T. Song, D. H. Woo, X. Zhao, J. Kim, H. Choi, G. H. Loh, H.-H. S. Lee, and S. K. Lim. 3D-MAPS: 3D massively parallel processor with stacked memory. *To appear in IEEE International Solid-State Circuits Conference, San Francisco, CA, 2012*, 2012.

[30] J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, V. Narayanan, M. S. Yousif, and C. R. Das. A novel dimensionally-decomposed router for on-chip communication in 3D architectures. *SIGARCH Comput. Archit. News*, 35:138–149, June 2007.

[31] J. Leary, M. Talupur, and M. R. Tuttle. Protocol verification using flows: An industrial experience. In *IEEE Formal Methods in Computer-Aided Design*, 2009.

[32] K. M. Lepak and M. H. Lipasti. Temporally silent stores. In *In Proceedings of PACT-2000*, pages 30–41, 2002.

[33] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, and M. Kandemir. Design and management of 3D chip multiprocessors using network-in-memory. In *Computer Architecture, 2006. ISCA '06. 33rd International Symposium on*, pages 130 –141, 0-0 2006.

[34] G. H. Loh, Y. Xie, and B. Black. Processor design in 3D die-stacking technologies. *Micro, IEEE*, 27(3):31 –48, may-june 2007.

[35] A. Meixner. Error detection via online checking of cache coherence with token coherence signatures. In *In Proceedings of HPCA-13*, pages 145–156, 2007.

[36] R. Merritt. Inside Intel's Sandy Bridge architecture.

[37] V. Pavlidis and E. Friedman. 3-D topologies for networks-on-chip. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 15(10):1081 –1090, oct. 2007.

[38] A. Pnueli, S. Ruah, and L. Zuck. Automatic deductive verification with invisible invariants, 2001.

[39] F. Pong and M. Dubois. Verification techniques for cache coherence protocols. *ACM Comput. Surv.*, 29:82–126, March 1997.

[40] J. Schanin. The design and development of a very high speed

system bus - the encore multimax nanobus. In *ACM Fall Joint Computer Conference*, 1986.

[41] B. Sinharoy, R. Kalla, J. Tendler, R. Eickemeyer, and J. Joyner. Power5 system microarchitecture. In *IBM Journal of Research and Development*, 2005.

[42] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*, pages 2 – 13, june 2003.

[43] D. Sorin, M. Hill, and D. Wood. Dynamic verification of end-to-end multiprocessor invariants. In *Dependable Systems and Networks*, pages 281 – 290, june 2003.

[44] S. Thakkar, M. Dubois, A. Laundrie, and G. Sohi. Scalable shared-memory multiprocessor architectures. *Computer*, 23(6):71 –74, jun 1990.

[45] A. Udipi, N. Muralimanohar, and R. Balasubramonian. Towards scalable, energy-efficient, bus-based on-chip networks. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1 –12, jan. 2010.

[46] D. Winsor and T. N. Mudge. Crosspoint cache architectures. In *In International Symposium on Computer Architecture*, pages 266–269, 1987.

[47] D. H. Woo, N. H. Seong, D. Lewis, and H.-H. Lee. An optimized 3D-stacked memory architecture by exploiting excessive, high-density TSV bandwidth. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1 –12, jan. 2010.

[48] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The splash-2 programs: characterization and methodological considerations. In *ISCA '95: Proceedings of the 22nd annual international symposium on Computer architecture*, pages 24–36, New York, NY, USA, 1995. ACM.

[49] D. Wood, G. Gibson, and R. Katz. Verifying a multiprocessor cache controller using random test generation. *Design Test of Computers, IEEE*, 7(4):13 –25, aug 1990.

[50] M. Zhang, A. R. Lebeck, and D. J. Sorin. Fractal coherence: Scalably verifiable cache coherence. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '43, pages 471–482, Washington, DC, USA, 2010. IEEE Computer Society.