A Scalable Micro Wireless Interconnect Structure for CMPs

Suk-Bok Lee[†], Sai-Wang Tam[‡], Ioannis Pefkianakis[†], Songwu Lu[†], M. Frank Chang[‡], Chuanxiong Guo[*], Glenn Reinman[†], Chunyi Peng[*], Mishali Naik[†], Lixia Zhang[†], Jason Cong[†]
[†]UCLA Computer Science [‡]UCLA Electrical Engineering [*]Microsoft Research Asia
{sblee,pefkian,slu,reinman,mishali,lixia,cong}@cs.ucla.edu,
{roccotam,mfchang}@ee.ucla.edu, {chguo,chunyip}@microsoft.com

ABSTRACT

This paper describes an unconventional way to apply wireless networking in emerging technologies. It makes the case for using a two-tier hybrid wireless/wired architecture to interconnect hundreds to thousands of cores in chip multiprocessors (CMPs), where current interconnect technologies face severe scaling limitations in excessive latency, long wiring, and complex layout. We propose a recursive wireless interconnect structure called the WCube that features a single transmit antenna and multiple receive antennas at each micro wireless router and offers scalable performance in terms of latency and connectivity. We show the feasibility to build miniature on-chip antennas, and simple transmitters and receivers that operate at $100 - 500$ GHz sub-terahertz frequency bands. We also devise new two-tier wormhole based routing algorithms that are deadlock free and ensure a minimum-latency route on a 1000-core on-chip interconnect network. Our simulations show that our protocol suite can reduce the observed latency by 20% to 45%, and consumes power that is comparable to or less than current 2-D wired mesh designs.

Categories and Subject Descriptors

C.1.2 [**Multiple Data Stream Architectures (Multiprocessors)**]: Interconnection architectures; C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Algorithms, Design

Keywords

Chip multiprocessors, On-chip wireless interconnection network

1. INTRODUCTION

In this paper, we explore a new area for the application of wireless networking technology. We devise a novel wireless network structure and associated protocols to interconnect hundreds to thousands of cores in a multi-core chip multiprocessor (CMP). Current generation interconnect solutions [23, 24, 32] face severe scaling limitations as the core population grows into the upper hundreds or even lower thousands, including excessive latency due to large hop counts, long wires, many intermediate repeaters, and complex wiring layout. We argue that wireless networking has tremendous promise as a scalable interconnect architecture for future generation CMPs.

Our research is motivated by several factors. CMP's have been widely acclaimed as the architecture of choice for future high-performance processors [1, 28]. Today's CMPs support tens to low hundreds of cores, including Intel's 80-core Terascale chip [32] and NVIDIA's 128-core Quadro GPU [26]. Both industrial and academic roadmaps project that we will see commodity CMP core counts in the upper hundreds or even thousands in the near future [4, 5, 25]. For example, Intel expects to have CMPs with high hundreds to low thousands of cores within ten years [5, 22]. Such a large amount of resources argues for new parallel programming paradigms to harness this processing power [19], and new OS efforts to manage these resources [10]. However, there still remains the question of how to scale the on-chip interconnect to provide low-latency and high-bandwidth communication between hundreds or even thousands of cores.

In this work, we set three specific goals in our design of a next generation, scalable on-chip interconnect structure for CMPs. First, the network architecture must scale to a large number of cores (e.g., accommodating potentially thousands of cores). This implies that it needs to ensure small hop counts and rich connectivity while reducing the use of long wires. Second, the structure and protocols must provide *low network latency* to minimize the time taken by inter-core data communications. Third, the operation must be *simple* enough to be implementable at the microarchitecture level.

In this paper, we make a case for a micro wireless interconnect architecture for CMPs with hundreds to thousands of cores. This wireless network-on-chip (NoC) architecture uses a two-tiered structure – a *wireless backbone* and *wired edges*, in contrast to today's wireless Internet architecture of a wired backbone and wireless edges. The wired two-dimensional mesh topology serves as the network edge and offers a local route for neighborhood inter-core message exchanges. The wireless structure provides wireless express pathways for long-haul, inter-core communication to ensure improved connectivity and reduced latency. Within this wireless interconnect, we also show the feasibility for novel, micro wireless routers. Each wireless router has a single transmitter and multiple receivers, both in very simple forms. These routers communicate wirelessly at tens of Gbps speed over the $100 - 500$ GHz, sub-Terahertz band via miniature on-chip antennas. External interference is negligible for CMPs at this high frequency band due to masking and the relatively few operational devices that are present.

MobiCom'09, September 20–25, 2009, Beijing, China.

Another benefit is that wireless signals attenuate too rapidly in the free space to create any interference for other devices. The transmit power we use is also too small to conflict with FCC regulations.

To realize the wireless on-chip interconnect architecture, we propose a new wireless interconnect structure called the *WCube* and devise routing and MAC protocols to leverage this architecture. WCube uses a recursively-defined structure to interconnect micro wireless routers (MWR) *wirelessly*, each of which is responsible for a local cluster of cores on the baseline mesh. It is a multi-level, two-dimensional (vertical and horizontal) structure, in that each MWR has wireless connections to different levels of WCubes along both dimensions via frequency division multiple access (FDMA) techniques. WCube scales exponentially with the MWR's logical connection degree, and its diameter is proportional to the WCube level. In practice, a WCube with even a small number of levels (e.g. 2), together with the baseline 2-D mesh, can support 1000s of cores with low network latency (e.g. at most 4 wireless hops for inter-core communication). With WCube in place, we also devise a new two-tier, wormhole-based routing algorithm that enables flit (i.e. a fragment in a packet) pipelining and ensures minimum-latency routes. The algorithm avoids packet-forwarding deadlocks, which occur due to cyclic dependences among multiple packet flows (note: these are different from routing loops). We also design a simple MAC that exploits multi-receiver capability and takes an FDMA based, cross-layer design approach. Overall, the WCube protocol operations such as routing decision logic and MAC unit are simple enough to be implemented at the microarchitecture level. Our simulation-based evaluations demonstrate that the latency can be reduced by 20% to 45% with comparable or even lower power consumption compared with the current 2-D mesh design.

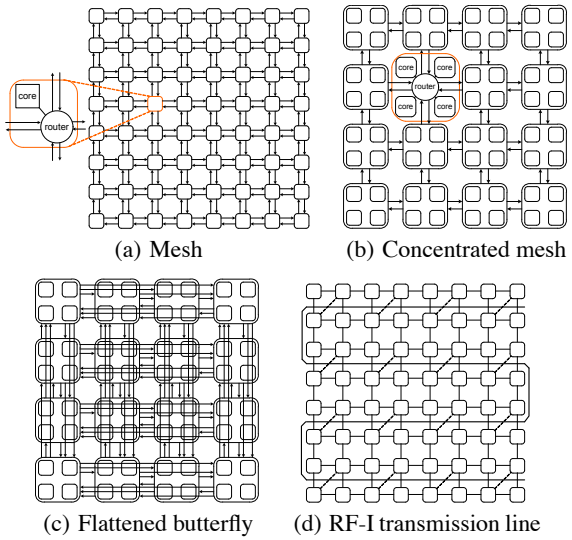
The rest of the paper is organized as follows. Section 2 introduces the existing interconnect solutions for CMPs, and Section 3 describes scaling limitations of current solutions to 1000s-core CMPs. Section 4 makes a case for on-chip wireless interconnect and demonstrates its feasibility. Section 5 presents WCube, a new wireless interconnect structure for 1000s-of-cores CMPs, and Section 6 describes routing and MAC protocols over WCube. Section 7 evaluates its performance. Section 8 discusses related issues, and Section 9 concludes the paper.

2. BACKGROUND

2.1 Chip Multi-Processors (CMPs)

In recent years, the performance increases possible with conventional superscalar single-core processors have encountered fundamental limits [1, 5], leading to an industry-wide turn towards chip multiprocessor (CMP) systems.

CMPs are becoming ubiquitous in all computing domains ranging from general purpose servers (e.g. DEC Piranha [6]) to the domain specific processors (e.g. NVIDIA Quadro FX 5600 [26]) – from 3G cellular base stations (e.g. picoChip PC205 [15]) to the latest game consoles (e.g. IBM/Microsoft Xenon Xbox360 [2] and Sony/Toshiba/IBM Cell PlayStation3 [23]). These CMPs today have dozens of tiled cores on a single chip. As the technology progresses, the core count is expected to grow to hundreds or even thousands in the near future [4, 5, 25]. Conventional wisdom is to double the number of cores on a chip with each silicon generation [4]. For example, the latest release (Fall 2008) of NVIDIA Tesla C1060 GPU has as many as 240 cores integrated in a chip [27]. The fact that such a high number of cores will be tightly integrated onto the same die presents a fundamental challenge for on-chip communication among cores, which is different from the challenges in previous multi-processor systems. A high performance interconnect for CMPs is essential to satisfying the data supply requirements of these cores, and plays a central role in overall system performance.



(a) Mesh (b) Concentrated mesh

(c) Flattened butterfly (d) RF-I transmission line

Figure 1: Current CMP topologies for a 64-core network.

2.2 Existing On-Chip Interconnect Solutions

The predominant approach to multi-core CMP interconnect is through various wired structures. Rings [23] and two-dimensional meshes [15, 32] are two common topologies, fitting well to a planar silicon die due to their low dimensionality. Two-dimensional meshes (see Figure 1(a)) have several key benefits, including short channel lengths and low router complexity. However, their network diameter is proportional to the perimeter of the mesh. Even a moderate-sized network will suffer from long network diameter, energy inefficiency, and high network latency. For example, a 10×10 mesh has a diameter of 18 hops.

A concentrated mesh (illustrated in Figure 1(b)) reduces the number of network nodes by sharing each network interface among multiple cores. A mesh with k-way concentration has the effective node count reduced by a factor of k, which leads to a smaller diameter and improved resource sharing.

More recent efforts propose to flatten a conventional butterfly topology into on-chip networks [24]. The resulting topology, called a flattened butterfly, together with the concentration technique, significantly improves the network diameter to only 2 hops. This is achieved by utilizing dedicated links to fully connect all the concentrated nodes in each dimension (see Figure 1(c)). However, in the flattened butterfly, the channel count in each dimension grows quadratically with the number of nodes, complicating the wiring layout. In addition, the wires connecting distant routers are necessarily much longer than those in the mesh. Long wires are generally undesirable since on-chip RC wires need a repeater every 1mm (or less) to propagate signals over long wires without degradation. The non-minimal channel lengths of the flattened butterfly adversely impact wire delay and energy, and complicate the routing and buffer reservation scheme [24]. Flattening other topologies widely used in multi-computer systems or data center networks, such as hypercubes, fat trees, or clos networks, is also not desirable for similar reasons.

An alternative approach [7, 8] proposes to use waveguide based transmission lines (called RF-I), where an electromagnetic (EM) wave is sent along the transmission line. This is in contrast to tradi-

tional voltage signaling over a conventional resistance-capacitance (RC) wire, whose entire length has to be charged and discharged to signify either a '1' or '0' in order to send information. This process of charging and discharging conventional RC lines can consume more time and energy than transmission lines for longer wires. Figure 1(d) shows the RF-I transmission line for a 64-core CMP. Transmission line-based interconnect offers several appealing properties over the traditional RC-wired interconnect. It can effectively reduce network diameter and improve power savings for mid-sized (in the range of tens to lower hundreds of cores) multi-core interconnect. Simulation evaluation also shows that RF-I can provide an average 22% reduction in packet latency on a 64-core CMP [7], compared with the conventional 2D mesh.

3. SCALING LIMITATIONS OF CURRENT SOLUTIONS TO 1000S-CORE CMPS

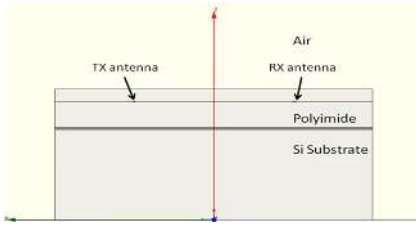
The current interconnect solutions all face severe scaling challenges when applied to CMPs with 100s to 1000s of cores. We now use a 1000-core example case to elaborate the scaling limitations of each solution. In this context, minimizing hop count turns out to be critical, as intermediate routers are the source of significant delay: typically five cycles per router. Moreover, long wires are undesirable since on-chip RC wires need a repeater every 1mm or less to retain signal quality over long wires. From both perspectives, existing solutions to 1000-core CMPs either suffer from latency problems (e.g., ring, mesh, concentrated mesh), or face structural limitations to accommodate 1000-core CMPs (e.g., flattened butterfly, RF-I transmission line).

The ring topology is cost effective, yet the least scalable technology. Its hop count grows linearly with the number of interconnected cores. Therefore, its diameter approaches 500 when the number of cores reaches 1000. The 2D mesh does not scale well, either. Its network diameter scales with the square root of the mesh size. It results in about a 60-hop diameter in the 1000-core case. Concentrated meshes are better, but their scaling property is still poor. Since physical limitations restrict the degree of concentration (as in Figure 1(b), typically $k = 4$), a larger network still has unacceptable network latency. A 1024-core mesh with 4-way concentration has a diameter of 30 hops.

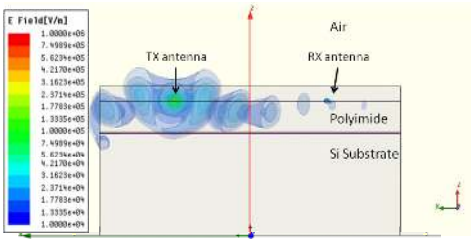
On the other hand, low-diameter topologies, such as flattened butterfly, have severe wiring problems caused by a large number of dedicated point-to-point links and long wires connecting distant routers. As a result, flattened butterfly is not implementable for thousands of cores because of long wires and the fact that there will be too many channels from each node. Note that long wires lead to long latency due to repeater insertion. Moreover, embedding conventional high-dimension topologies onto 2-D substrates is similarly prohibitive.

Finally, while the RF-I transmission line based interconnect holds great promise for the lower hundreds of cores, it also faces several challenges in the 1000-core setting. The RF-I transmission line (TR line) needs to span the entire chip area, and requires excessive branching points to connect to local cores. Moreover, the TR line is not as effective as antennas at very high frequencies. The cross-talk (or inter-channel interference) between adjacent TR lines may also pose problems for long TR lines.

In summary, a desirable CMP solution for 1000s-core CMPs should feature (i) *low dimensional baseline infrastructure* that is compatible with the planar VLSI layout constraint for on-chip networks, and (ii) *some form of express channels* to ensure small-hop and rich connectivity [20]. In addition, *it is desirable to avoid high wiring complexity and long wires.*



(a) Cross section of on-chip antenna model.



(b) On-chip antenna radiation intensity plot.

Figure 2: On-chip antenna simulation.

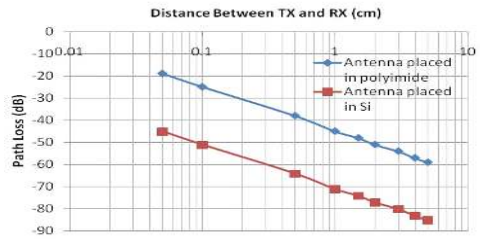
4. A CASE FOR WIRELESS INTERCONNECT

4.1 On-Chip Wireless Interconnect

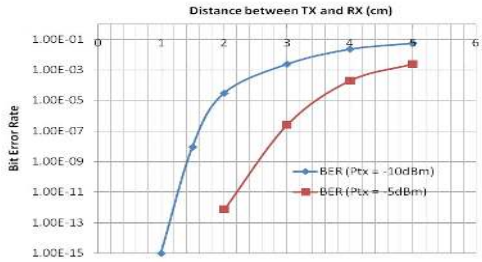
A key benefit from the scaling of CMOS is that the switching speed of transistors improve over successive technology generations. According to the International Road Map for Semiconductors (ITRS) [21], the unity current gain frequency f_T and the maximum available power gain f_{max} will be 600 GHz and 1 THz, respectively, in 16 nm CMOS technology. A new record of a 324 GHz terahertz (between 300 GHz to 3 THz) CMOS oscillator using a linear superposition technique [14] has been reported using the standard digital 90 nm CMOS process. Based on this technique, the output power level of the on-chip millimeter-wave generator has been predicted to be as high as -1.4 dBm in the 32 nm CMOS process, which is large enough for on-chip short distance communication. With the advance in CMOS mm-wave circuits, hundreds of GHz of bandwidth will be available in the near future. The question is: how can we utilize hundreds of GHz of bandwidth for future CMPs?

Size of on-chip components. Using Terahertz CMOS for on-chip wireless interconnect has several benefits. An on-chip antenna is always one of the most difficult components that can be integrated on-chip, since passive devices such as inductors consume the dominant portion of the transceiver area. For example, at 15 GHz the size of the on-chip antenna is as large as 2 mm [16], which is too large for an on-chip wireless network. The size of a passive device, as well as the wavelength, scales down with the operating frequency of a given circuit. As the CMOS technology improves, not only the size but also the cost of the antenna and required circuits will decrease dramatically. Consequently, the size of the on-chip antenna at 300 GHz can be as small as $100\mu m$, about a 20x reduction in area. At 32 nm CMOS technology, the area of each individual core in a 1000-core CMP can be as small as $700\mu m \times 700\mu m$, where a terahertz antenna can be easily placed at each individual core.

Our proposed on-chip antenna. Despite a reduction in antenna size by more than 20x, the current practice of integrating an antenna directly on silicon will cause a significant energy loss



(a) Channel loss improvement using our antenna.



(b) Bit-error rate with two different transmitter output power -10 dBm and -5 dBm via our proposed antenna.

Figure 3: On-chip wireless communication using our proposed on-chip antenna.

in a conductive silicon substrate. In order to minimize the substrate loss, we propose that an on-chip antenna should be placed in a polyimide layer (a few mils thick), which deposits on the top of the top silicon, such that most electro-magnetic energy can be confined within this low-loss dielectric layer. As shown in Figure 2(a), the cross section of the terahertz on-chip antenna is placed in the thick polyimide layer, which is easily manufactured by simple post-silicon fabrication processes [9]. In order to demonstrate the feasibility of a terahertz on-chip antenna, we have developed a detailed design prototype and used the industry-strength, full EM-wave simulator, Ansoft HFSS [3], to further understand the characteristics of the proposed on-chip antenna. As illustrated in Figure 2(b), the on-chip antenna radiation intensity plot shows that most of the energy is confined within the low-loss dielectric polyimide layer. Even though radiation attenuates significantly at the receive antenna (about -40 dB at 1 cm away from the transmitter), the receiver still has enough sensitivity to detect the signal. The key point is that *antenna radiation penetrating to the substrate must be minimized because performance of the on-chip antenna is highly sensitive to the substrate loss*.

On-chip wireless channel characteristics. Compared with early work which directly implements the antenna in the silicon, our proposed on-chip antenna placed in the low-loss dielectric polyimide layer improves the channel loss by $20 - 30$ dB on average. Figure 3(a) shows the channel loss improvement using our proposed antenna. From this model, we further develop a simple communication system to estimate the bit-error rate (BER) for a future on-chip wireless channel. Figure 3(b) plots BER with two different transmit output powers -10 dBm and -5 dBm, and shows several interesting characteristics of future on-chip wireless networks. Within the maximum communication distance of CMPs ($\sim$ 1.5 cm), the BER of the future on-chip wireless network is less than 10^{-9} which is low enough for typical wireless networks. This BER is however far higher than that of RC wires ($\sim 10^{-14}$), and we address this issue later in Section 6. Note that there is a direct tradeoff between

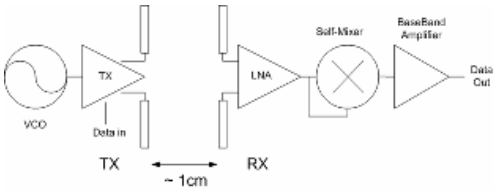


Figure 4: A simple asynchronous amplitude-shift-keying (ASK) system for on-chip wireless network.

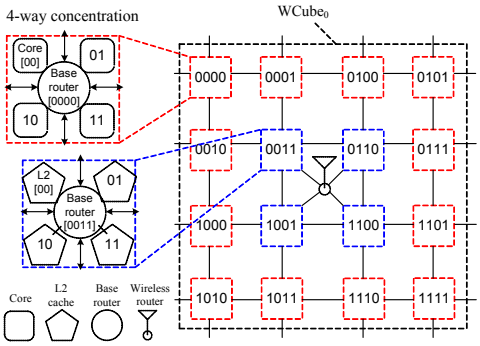
power consumption and BER. Depending on the size of the CMP, it is possible to consume more power to compensate for the BER. Nevertheless, the transmit power is too low to violate any FCC regulations. Another interesting aspect is that as the distance increases beyond 2 cm, the BER rapidly increases. This trend, together with the channel loss result, indicates that on-chip wireless signals attenuate too fast in free space to create interference with any off-chip devices. Moreover, no other system devices operate in such extremely high frequency bands. Thus, the terahertz on-chip wireless channel is interference free, and possibly reusable to adjacent on-chip networks.

On-chip wireless channel capacity. Because of such low signal loss over on-chip wireless channels and new techniques in generating terahertz signals on-chip [14, 31], the on-chip wireless network becomes feasible. In addition, it is possible to switch a CMOS transistor as fast as 500 GHz at 32 nm CMOS [21], thus allowing us to implement a large number of high frequency bands for the on-chip wireless network. Following a rule of thumb in RF design, the maximum available bandwidth is 10% of the carrier frequency. For example, with a carrier frequency of 300 GHz, the data rate of each channel can be as large as 30 Gbps. Using a 32 nm CMOS process, there will be total of 16 available channels, from 100 GHz to 500 GHz, for the on-chip wireless network, and each channel can transmit at 10 to 20 Gbps. In the 1000-core CMPs design, the total aggregate data rate can be as high as 320 Gbps with 16 TX's and 64 RX's.

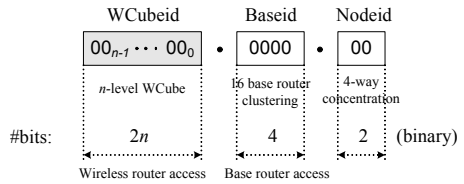
On-chip radio architecture. In addition to a large bandwidth capacity, the on-chip wireless network requires a simple architecture and low power transceiver design to satisfy the stringent requirements of future CMPs. A simple asynchronous amplitude-shift-keying (ASK) system suffices to satisfy these requirements [30]. Figure 4 shows such a simple on-chip radio architecture, which has one oscillator and one ASK modulator in a TX and one demodulator and simple baseband circuit in a RX. With such a simple transceiver architecture, the terahertz on-chip wireless network only uses 1% to 2% of the total CMP power consumption. It is a low-power, high-data-rate, and reconfigurable interconnect technology.

4.2 Two-tier Hybrid Wireless/Wired Architecture

In this section, we present a novel, two-tier, hybrid wireless/wired interconnect architecture for future 1000-core CMPs. Our proposed two-tier architecture benefits from the baseline 2-D concentrated mesh (CMesh) to provide a base network with very short wires, and exploits a wireless backbone to enhance connectivity, reducing the network latency. There are three types of nodes in the architecture: cores, L2 caches, and memory interfaces. They are all potential sources/destinations of packets. In addition, the network has two types of routers: base routers that make up the baseline CMesh, and wireless routers that have wireless interfaces to form a wireless backbone.



(a) A $WCube_0$: a cluster of 16 base routers (i.e. 64 nodes). The numbers inside nodes and base routers indicate nodeid and baseid respectively.



(b) Hierarchical addressing.

Figure 5: (a) Layout of a $WCube_0$. (b) Addressing example with wcubeid, baseid, and nodeid all zeros.

4.2.1 *Baseline Topology*

The baseline CMesh is basically a 2-D mesh employing k-way concentration (see Figure 1(b)). Since physical limitations restrict the degree of concentration [24], we set $k = 4$. Thus, we use a 4-way concentrated mesh as the baseline topology in this work. As in Figure 5(a), four nodes (all four cores, or all L2 caches) are attached to a single base router that also connects to adjacent routers. We apply concentration on cores and caches separately to exploit a storage spatial hierarchy, thus reducing overall network latency. This issue will be elaborated in the next section.

4.2.2 *Wireless Backbone*

Built on top of the baseline CMesh, the wireless backbone consists of wireless routers, each of which is responsible for a cluster of m base routers (i.e. $4m$ nodes due to 4-way concentration). The choice of m offers a design tradeoff; a larger m would lead to a smaller number of wireless routers in the network while increasing network latency because of high hop count between base routers within the same cluster (e.g., in the case that source and destination nodes are in the same cluster). The opposite would be true for a smaller m. In this work, we use a cluster of $m = 16$ base routers (i.e. 4×4 base routers) for two reasons: (a) the hop count within a cluster is at most 6 (via the baseline CMesh), which is reasonably small for the intra-cluster communication, and (b) in terms of packet latency, it is more beneficial to use the baseline wires within a size of 4×4 CMesh (rather than using wireless backbone) due to our wireless channel characteristics as described in Section 6. In addition, with our choice of $m = 16$, the hop count between a base router and a wireless router is at most three for long-distance packets that use the wireless backbone (see Figure 5(a)).

L2 caches are attached to the four base routers in the center of a cluster, while the surrounding 12 base routers are responsible for cores. In addition, each cluster has one memory interface that has access to off-chip DRAM, and one of the four centered base routers is in charge of one memory interface and three caches (instead of four caches). We choose such a layout to decrease the distance that the largest messages (i.e. DRAM responses, 128-byte L2 cache blocks from a main memory interface to a requesting L2 bank) must travel, and to reduce contention between these larger messages and traffic between cores and L2 caches – effectively, the L2 caches are surrounded by the processor cores. Data from the DRAM will never use a wireless backbone since every cluster has a memory interface.

Nodes	256	512	1024	2048	4096
Wireless routers	4	8	16	32	64
Structure	**Node degree , Network diameter**				
FullMesh	3 , 1	7 , 1	15 , 1	31 , 1	63 , 1
Ring	2 , 2	2 , 4	2 , 8	2 , 16	2 , 32
2D Mesh	4 , 2	4 , 3	4 , 6	4 , 9	4 , 14
FButterfly	4 , 2	7 , 2	12 , 2	18 , 2	28 , 2
FatTree	4 , 4	4 , 6	4 , 8	4 , 10	4 , 12
Hypercube	2 , 2	3 , 3	4 , 4	5 , 5	6 , 6

Table 1: Comparison of different structures in terms of the number of wireless routers.

The wireless routers are wirelessly interconnected using WCube, a new wireless interconnection structure which inherits some useful properties from a hypercube topology. With our target of 1000s of cores, our hybrid architecture has up to tens of wireless routers in the wireless backbone. As shown in Table 1, the hypercube performs the best among different structures in such moderate-sized networks (in terms of the number of wireless routers). A small node degree implies fewer links, which results in fewer receive antennas at each wireless router. A small network diameter reflects a small hop count between any two nodes, typically resulting in low network latency. As shown in the table, a hypercube balances both metrics, while other structures are unable to achieve both. In this work, we focus on a 1024-node on-chip network – so we have a total of 16 wireless routers in the network to form the wireless backbone. We will detail this further in Section 5. However, WCube is a highly scalable structure and could easily adapt to different numbers of core.

4.2.3 *Addressing*

In our hybrid architecture, each component is assigned an address of the triple-dotted form. The wireless routers are given addresses of the form *wcubeid.X.X*, where *wcubeid* denotes the WCube address assigned to the wireless router, and *X* denotes *don't care* bits for padding purpose. In fact, only *wcubeid* is an effective address part for identifying a wireless router. The base routers have addresses of the form *wcubeid.baseid.X*, where *baseid* denotes the position of the base router in the cluster. We use binary addressing so *baseid* has 4 bits (the first two bits indicate the position of the base router from the wireless router - one of four quadrants, starting from left to right, top to bottom). In our architecture, the router addresses allow such *don't care* bits, as the wireless routers and the base routers only deal with the *wcubeid* part and with the extra *baseid* part respectively, and they only relay packets. They do not generate or consume packets, i.e., they are neither the source nor the destination. The nodes (cores, caches, and memory interface) are given addresses of the form: *wcubeid.baseid.nodeid*, where *nodeid* denotes the position of the node from the attached the base router (starting from left to right, top to bottom). Figure 5(b) shows an example of a node address with *wcubeid*, *baseid*, and *nodeid* as all zeros.

The next two sections describe the complete architecture and its protocol design in detail.

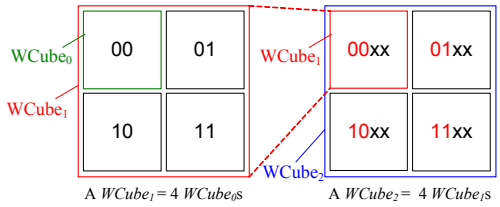


Figure 6: WCube physical structure and its addressing.

5. WCUBE: WIRELESS BACKBONE STRUCTURE

5.1 Physical Structure

The WCube is a multi-level, two-dimensional structure, which can be recursively described. A high-level WCube is composed of multiple low-level WCubes. We use $WCube_n$ ($n \geq 0$) to denote a level-n WCube. $WCube_0$ is the building block to construct larger WCubes. $WCube_0$ is represented by a single wireless router that is responsible for a cluster of $m = 16$ base routers of the baseline CMesh, thus comprising $4m$ nodes (64 nodes for $WCube_0$ in Figure 5(a)).

In the WCube structure, a level-i $WCube_i$ is constructed using four $WCube_{i-1}$s (i.e. two $WCube_{i-1}$s along the vertical and the horizontal dimensions each). Figure 6 illustrates the structure of a level-2 $WCube_2$. It consists of four $WCube_1$s, each of which also has four $WCube_0$s in it.

To identify a wireless router in the n-level WCube structure, each wireless router is assigned a WCube address (or wcubeid) of the $2n$-bit form $< a_{2n-1}a_{2n-2}\cdots a_1a_0 >$, where $a_i \in [0, 1]$ and $i \in [0, 2n-1]$. As shown in Figure 6, each WCube level contributes two bits to the WCube addressing space such that a_{2i+1} and a_{2i} are the respective vertical and horizontal coordinates of the level-i $WCube_i$ within the level-$(i+1)$ $WCube_{(i+1)}$ that this wireless router belongs to. For example, in the 2-level WCube (see Figure 6), WCube address $< 0011 >$ identifies the wireless router that is positioned at the top left $WCube_1$ in $WCube_2$ and the bottom right $WCube_0$ in that $WCube_1$.

5.2 WCube Construction

The WCube uses wireless routers each equipped with *one wireless transmitter and multiple receivers (i.e., multiple receive antennas)* to construct its recursively defined structure.

Parallel transmission. Leveraging an extremely wide spectrum available in a chip, every wireless router ν is assigned a single, different frequency band (i.e. orthogonal to one another) exclusively used for ν's transmission such that wireless routers can transmit data in parallel and do not interfere with one another. The basic idea to implement wireless connection among wireless routers is to tune each router's receivers to a certain set of frequency bands (via multiple receive antennas), thus offering a natural method to perform parallel multicast across the chip.

Receiver tuning. In the WCube, each wireless router ν's receivers are *statically* tuned to the frequency bands of the other wireless routers whose WCube addresses differ from ν in only one bit. We refer to those wireless routers as ν's logical neighbors. For example, wireless router $< 0000 >$'s four receivers RX1∼RX4 are statically tuned to the TX frequency bands of $< 0001 >$, $< 0010 >$, $< 0100 >$, and $< 1000 >$, respectively. The Hamming distance between two wireless routers ν and υ, denoted by Ham(ν,υ), is defined as the number of different bits in their WCube addresses. In the WCube structure, two wireless routers ν and υ are tuned to each other if and only if Ham(ν,υ) = 1.

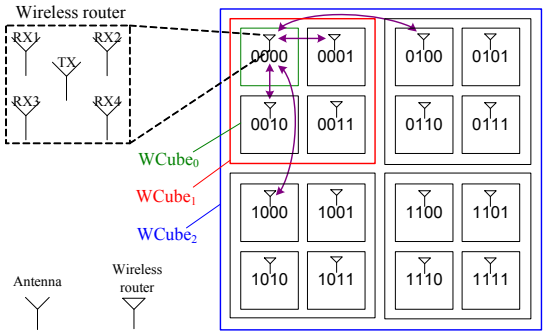


Figure 7: A 2-level WCube. Lines denote the logical connection of wireless router $< 0000 >$ with one TX and four RXs.

Multicast via multiple receivers. Using such static receiver tuning, every wireless router ν is able to listen on all its logical neighbors' transmissions via multiple receivers. It only accepts packet(s) whose intended next-hop wcubeid is of ν itself. Furthermore, every packet transmitted from each wireless router ν is multicast to all its logical neighbors, and since they tune the receivers to their own neighbors' frequency bands, that must include ν's frequency band as well. Only the intended next-hop neighbor accepts the packet while the others simply discard the overheard packet. Therefore, every wireless router has bidirectional wireless connections with its logical neighbors. It is worth noting that there is no dedicated wireless link between any two wireless routers in WCube. Physically, each wireless router has access to the logical neighbors via its own wireless channel.

Illustration. Figure 7 illustrates the logical connection of wireless router $< 0000 >$ in a 2-level $WCube_2$. Lines in the figure indicate bidirectional wireless links between $< 0000 >$ and its logical neighbors. Outgoing links from $< 0000 >$ are all a single wireless TX channel assigned to $< 0000 >$, and each incoming link from the neighbors are a different frequency channel that $< 0000 >$'s receivers (RX1∼RX4) are statically tuned to. Note that every wireless router has different set of logical neighbors, since no two wireless routers have the same WCube address.

5.3 Properties of WCube

Given its physical structure and wireless connectivity among wireless routers, WCube has following properties:

- An n-level WCube, incorporating the baseline k-way CMesh (with a cluster of m base routers associated with a single wireless router), can support up to $2^{2n} \times m \times k$ nodes. For practical considerations discussed in Section 4.2.2, we use the 4-way concentration ($k = 4$) and a cluster of $m = 16$ base routers so that an n-level $WCube_n$ can have at most 2^{2n+6} nodes. For example, when $n = 2$, a WCube can have as many as 1024 nodes.
- Each wireless router has $2n$ logical neighbors in an n-level WCube. For each dimension, any wireless router has only one neighbor whose wcubeid differs by one bit at each WCube level. WCube is n-level two-dimensional structure so that every wireless router has $2n$ neighbors. Thus, each wireless router has $2n$ receivers, each of which is tuned to one of $2n$ logical neighbors. For example, a 2-level WCube has four logical neighbors per wireless router, i.e each wireless router has four receive antennas.

- The hop count between any two wireless routers ν and υ is equal to $Ham(\nu, \upsilon)$, i.e. the number of different bits of their WCube addresses. This is due to the WCube construction procedure, and we only need to change $Ham(\nu, \upsilon)$ bits to get the complete routing path from ν to υ. The maximum hop count is therefore $2n$ for a $WCube_n$ network.

5.4 Comparison to Hypercube

WCube can be viewed as a binary hypercube in that each wireless router connects wirelessly to one logical neighbor in each WCube level along each dimension, and such "logical" links of WCube are in a sense similar to the "wired" links of the hypercube. However, the main difference is the use of multicast links by exploiting the broadcast nature of the wireless medium in the WCube structure, i.e. all logical neighbors can be accessed via a single wireless channel. The use of multicast channels completely removes the dedicated point-to-point links that are problematic for a large network, thus significantly reducing the number of physical channels in the network. For example in a WCube with N wireless routers, the total number of channels is N; whereas for an equivalently-sized hypercube the number of wires is $N \log_2 N$ (due to its dedicated links). Moreover, embedding a hypercube onto 2D substrates is prohibitive due to the resulting awkward network layout and long wire delays as explained in Section 2.2.

5.5 Partial WCube

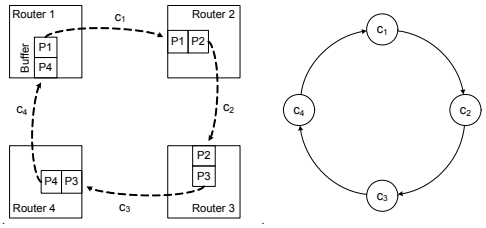
A nice property of WCube is that the structure can easily adapt to the various core counts by appropriate WCube addressing. For example, we can build a partial $WCube_2$ for 512 nodes by regulating their wcubeid form $< a_3 a_2 a_1 a_0 >$ to have $a_3 = 0$ and $a_2, a_1, a_0 \in [0, 1]$. Then, the resulting partial $WCube_2$ consists of only two full $WCube_1$s instead of four $WCube_1$s, while retaining the original WCube logical connection property imposed by the Hamming distance. Using this way of controlling the radix of wcubeid (together with varying the size of $WCube_0$ if needed), a partial WCube can manage different-sized CMPs.

6. ROUTING AND MAC SUPPORT OVER WCUBE

In this section, we present the WCube routing protocol for our two-tier interconnect architecture. Our two-tier routing is based on the idea of opportunistic use of WCube to improve network latency and connectivity of the baseline CMesh. Using two-tier routing, the baseline CMesh is analogous to city streets accommodating local traffic, and WCube is like a superhighway, connecting distant spots on the chip.

6.1 Issues

There are several issues when designing an on-chip wireless routing protocol, due to the unique characteristics of on-chip networks. First, packet latency is the most critical factor for the performance of on-chip interconnection networks, thus making a case to use latency-oriented routing, a wormhole-based delivery, which enables flit (i.e., a fragment in a packet) pipelining and facilitates minimum-latency routes. Second, such latency-oriented routing may incur packet-forwarding deadlocks, which never occur in the traditional packet store-and-forward approach. Deadlock in the interconnection network is different from the routing loop in that it is triggered by the cyclic channel dependency among multiple packet flows. Third, the impact of packet loss is much severe in the on-chip network, compared with the conventional wireless networks. The BER of the on-chip wireless channel still does not satisfy the stringent on-chip reliable communication requirements. Therefore, loss



(a) Deadlock situation (b) Dependency graph

Figure 8: Deadlock situation involving four channels.

management must be addressed. Fourth, it should support broadcast. Fifth, protocol operations such as routing decision logic and MAC have to be simple enough to be implemented at the microarchitecture level.

6.2 Wormhole Routing over 2-tier Structure

In order to minimize the communication latency, our architecture uses wormhole routing [13], an efficient switching technique widely used in parallel computers. In wormhole routing, a packet is divided into a sequence of fixed-sized units of data, called *flits*, where a flit is the smallest unit of flow control. Instead of storing a packet completely in a node before transmitting it to the next node, wormhole routing operates by advancing each flit of a packet directly from incoming to outgoing channels as soon as it arrives at a node (pipelining). Examining the header flit of a packet, a node selects the next channel on the route and begins forwarding followup flits down that channel. As the header flit advances along the specified route, the remaining flits of the packet follow in a pipelined fashion. Wormhole routing is attractive in that (i) it reduces the latency of packet delivery noticeably compared with conventional store-and-forward switching that waits for the whole packet before forwarding, and (ii) it requires only a small FIFO flit buffer at each node [12].

On the other hand, by its nature, it is subject to deadlock conditions. Because most flits contain no routing information, the flits in a packet must remain in contiguous channels of the network and cannot be interleaved with the flits of other packets. When the header flit of a packet is blocked, all of the flits of a message stop advancing and block the progress of any other packet requiring the channels they occupy. Figure 8(a) shows such deadlock example where the header flit of each packet is blocked by the tail flit in the different packet one another in a circular manner (e.g., at router 2, header flit of packet 1 is blocked by tail flit of packet 2, etc). This condition of circular dependency leads to a state of deadlock, where all of the packets involved in the deadlocked channels are blocked. Figure 8(b) shows a corresponding *channel dependency graph*, where the vertices are the channels and the edges are the pairs of channels connected by a routing algorithm. One possible way of assuring that a routing algorithm is deadlock-free is to verify that no cycles exist in the network's channel dependency graph [11]. Deadlock avoidance is one of the most critical issues in wormhole networks.

6.2.1 Deadlock-free Routing at Each Tier

We first address the deadlock avoidance issue at each tier independently: baseline CMesh routing (performed by base routers) and WCube routing (by wireless routers). We then extend the deadlock-free routing to our two-tier architecture.

In general, deadlock avoidance tries to prevent the formation of a cycle, which is a necessary condition of deadlock. The turn-

```
/* pkt is the packet received by wireless router ν
   ν.wcubeid = < a_{2n-1}a_{2n-2}···a_1a_0 >
   pkt.dst.wcubeid = < b_{2n-1}b_{2n-2}···b_1b_0 > */
WCubeRouting(pkt)
   if (pkt.nexthop == ν.wcubeid)
      if (pkt.dst.wcubeid == ν.wcubeid)
         out_port = FTableLookup(pkt.dst);
         forward pkt to out_port and return;
      else
         pkt.nexthop = < a_{2n-1}a_{2n-2}···a_1a_0 >;
         for (i = 2n − 1; i ≥ 0; i − −)
            if (a_i ≠ b_i) change a_i to b_i in pkt.nexthop;
               transmit pkt and return;
   else discard pkt;
```

Figure 9: Pseudocode for WCube deadlock-free routing.

model [18] completely avoids deadlock by making sure that the set of allowable turns made by packets in the network cannot form a cycle. A cycle in a mesh consists of several turns. As an example in 8(a), SE (from South input channel to East output channel), ES, NW, and EN turns are essential in a clockwise cycle. Our baseline 2D mesh routing uses the standard XY routing where the packets are routed along the X dimension first, then along the Y axis to their destination. Note that XY routing is made deadlock-free by restricting turns from the Y dimension to the X dimension.

For the WCube structure, we can ensure deadlock freedom by employing dimension-ordered routing [11], which assigns each channel a unique number and allocates packets to channels in strictly decreasing orders along the route. In an n-level WCube, a wireless router R_ν has $2n$ logical output channels (one for each neighbor), labeled $c_{0,\nu}, \cdots, c_{(2n-1),\nu}$. We have a total ordering of the channels in the structure according to their subscript: $c_{(2n-1),(2^{2n}-1)} > c_{(2n-1),(2^{2n}-2)} > \cdots > c_{0,1} > c_{0,0}$. Figure 9 shows our WCube routing algorithm. A packet arriving at wireless router R_ν destined for wireless router R_υ is routed on channel $c_{i,\nu}$ where i is the position of the most significant bit in which ν and υ differ. Since packets are routed in the order of decreasing channel subscript, there are no cycles in the channel dependency graph, and hence our WCube routing is deadlock free.

Upon receiving a packet, the wireless router first decides whether to forward or discard it by checking the *nexthop* field of the packet that contains the wcubeid of the intended next-hop wireless router so that other logical neighbors drop this packet immediately. Then, the wireless router compares the destination wcubeid of the packet and its own wcudeid, and decides the next hop by correcting the left-most unmatched bit. Then, the destination wireless router forwards the packet to the base router whose first two bits of baseid match those of the destination baseid of the packet by looking up the forwarding table. Recall that only four base routers are directly connected to a wireless router, and the first two bits of baseid indicate the position of the base router from the wireless router - one of four quadrants (see Figure 5(a)). Since at most three extra hops between the wireless router and the base router are needed at both the beginning and the end of the WCube route whose path length is at most $2n$ in a $WCube_n$, the maximum hop count between any two nodes is therefore $2n + 6$ in our two-tier architecture.

The number of entries of the forwarding table at each wireless router is $2n$, which is the number of logical neighbors a wireless router has, plus m entries for a cluster of m base routers ($m = 16$ in our case) that the wireless router is in charge of. Thus, every wireless router has a forwarding table with $2n + m$ valid entries. For example, the table has only 20 entries in a 2-level WCube network, accommodating as many as 1024 nodes. Each entry in the table has two fields: an *out_port* indicating the outgoing port number and *status_info* indicating the status of the neighbor. Note that all $2n$ entries for logical neighbors of the wireless router have the same *out_port* value, since all of the logical neighbors can be accessed via a single wireless channel.

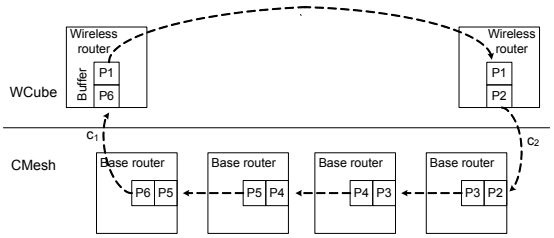


Figure 10: Deadlock example in 2-tier structure.

6.2.2 Deadlock-free in 2-tier Hybrid Architecture

While our baseline mesh routing and WCube routing are deadlock free by themselves, the resulting two-tier hybrid architecture is not actually deadlock free yet. It can still form a cycle, which spans over both mesh and WCube structure. Figure 10 illustrates such a deadlock example.

We address this issue by breaking the channel dependency between mesh and WCube (i.e., between base routers and wireless routers). Our hybrid architecture employs *virtual channels* (VCs) [12], where each physical channel is split into several VCs (rather than a single FIFO buffer).[1] In order to break deadlock in the hybrid architecture, we define two types of VCs per physical channel, $*$Up and $*$Down. Only the physical channels between base routers have both types of VCs, and other physical channels (e.g., between a wireless router and a base router) have only $*$Down channels. We note that the wireless channels (i.e., between wireless routers) do not use VCs in order to simplify the loss management mechanism, which will be described in Section 6.4. Packets, once generated, first move through the base routers via $*$Up channels. Channel switching from $*$Up to $*$Down is allowed only when the packet enters the WCube. Recall that the physical channels between base routers and wireless routers have only $*$Down channels. VC switching from $*$Down to $*$Up is prohibited in any case. With these constraints, the channel dependency graph for the hybrid architecture is acyclic.

THEOREM 1. *The combination of XY routing and WCube routing in the 2-tier hybrid architecture, along with $*$Up and $*$Down virtual channels, is deadlock free.*

The proof is given in the Appendix.

Using $*$Up and $*$Down VCs, we break the dependency on the channel from a base router to a wireless router (i.e., channel c_1 in the example of Figure 10), instead of breaking the channel from a wireless router to a base router. While breaking the dependency on either channel can achieve deadlock freedom, we choose the former approach where the VC switching is completely hidden from wireless routers that would otherwise be involved in VC switching.

6.3 Minimum-Latency Routing

We now have deadlock-free baseline mesh routing and WCube routing, and the next design concern is how to determine whether a packet should be forwarded towards a wireless router (to use the WCube) or delivered only by base routers (to use mesh routing).

[1]Each virtual channel has its own buffer to help in decoupling buffer resources from transmission resources, thus increasing channel utilization.

The latency of a packet through an interconnection network can be expressed as the sum of the header latency T_h, the serialization delay T_s, and medium delay T_m:

$$T = T_h + T_s + T_m = Hd_r + L/W + T_m$$

where H is the hop count, d_r is the router delay, L is the packet size, and W is the channel bandwidth. Minimizing latency requires establishing careful balance between T_h and T_s [24]. The conventional on-chip network has plentiful channel bandwidth (e.g., link bandwidth W_{mesh} = 16 bytes/cycle), thus significantly reducing T_s. However, 2D mesh networks fail to balance between T_h and T_s due to the high hop count. On the other hand, WCube has a very small hop count while the wireless channel bandwidth is not yet comparable to the wire link. According to our result in Section 4, wireless link bandwidth W_{wcube} is 1 byte/cycle at the current stage. Given that the average packet size L is 20 bytes[2], and using 5-cycle pipelined routers (i.e. d_r = 5 cycles), the latency of the packet using only the mesh T_{mesh} and the latency with WCube $T_{mesh+wcube}$ can each be calculated in terms of cycles: $T_{mesh} = 5H_{mesh} + 1.25$ and $T_{mesh+wcube} = 5H_{mesh+wcube} + 20$. Hence, we can minimize latency by opportunistically using WCube when $H_{mesh} - H_{mesh+wcube} \geq 4$; and in other cases, using mesh routing only.

Recall that we use a cluster of 4×4 base routers as $WCube_0$, and the hop count within a cluster is at most 6 (via the baseline mesh). The above criterion indicates that we do not benefit from using wireless shortcut within a cluster size of 4×4 CMesh (even if we have multiple wireless routers in a $WCube_0$) since we cannot satisfy the inequality $H_{mesh} - H_{mesh+wcube} \geq 4$ in any case.

A routing table in each base router is statically configured based on the above criterion, so that it has one entry per destination base router that needs mesh-routing only, and a single default entry for other destinations that need WCube routing. The *out_port* of the default entry connects either directly to the wireless router or to a neighbor base router closer to the wireless router. Note that we can further reduce the number of entries in the routing table by combining entries for 16 base routers with the same wcubeid into one entry. We have a single bit per packet indicating whether to use WCube. This bit is set only by a source base router (by looking up its routing table), and always reset by wireless routers. Once this bit is set, the packet must use WCube (i.e., at each intermediate base router, this packet comes under the default entry towards WCube even if there exists an entry for the destination of this packet). We again note that our baseline mesh routing uses XY routing to ensure deadlock freedom.

6.4 Loss Management

A single message loss can cause serious performance degradation in the on-chip network, since a message itself may have some dependency on the operation of a group of different nodes (e.g., cache coherence protocol, pipelining data flow, etc). The current RC wires have extremely low bit-error rate (BER) of approximately 10^{-14}. Within the maximum communication distance of future CMPs, 1.5cm, the BER of the on-chip wireless channel is less than 10^{-9} (see Figure 3(b)), which is far higher than that of RC wires. Hence, WCube must properly manage message loss.

We devise a novel and simple loss management solution in WCube. We use a zero-signaling-overhead scheme OAR based on overhearing on intermediate hops, and use an on-demand, checksum-based error-detection and retransmission scheme at the last hop. Overhearing-and-retransmission (OAR) detects and recovers packet losses without extra signaling overhead. It exploits the interconnection property of WCube. OAR does not require an explicit ACK. Rather, it utilizes the free overheard packets for loss detection. Note that every wireless router is able to listen on all the logical neighbors' transmissions via multiple receivers, and every logical link is bidirectional, i.e. any two neighbors are mutual neighbors in WCube. For example, wireless router $< 0000 >$ forwards a packet to $< 1000 >$, and when $< 1000 >$ relays this packet to any next hop, this packet is overheard by previous hop $< 0000 >$ due to the WCube interconnection property. Instead of simply discarding the overheard packet, the router checks whether the packet matches those in the corresponding retransmission FIFO buffer (RtFIFO). Each wireless router maintains an RtFIFO per its logical neighbor. When router i forwards a packet to j, it stores the packet into RtFIFO in charge of j unless j is the destination WCube router. When this packet is overheard by i and found in the RtFIFO, then all the remaining packets in the RtFIFO, if any, ahead of this packet are considered lost, and will be retransmitted to j and put back into the end of the RtFIFO. Note that the lost packet(s) is retransmitted to j only by previous-hop router i, since only the original sender i has the copy of the overheard packet in its RtFIFO. Then the overheard packet is removed from the RtFIFO. Since WCube channels do not use virtual channels, packets are always forwarded in sequence. Hence, OAR works correctly and guarantees the reliable packet delivery using simple buffering mechanism.

However, we still have not addressed the last-hop case, where the next hop is the destination WCube router and overhearing is impossible. OAR resolves this issue by inserting checksums (4-bit or 8-bit) into the packet, if the wireless router sees that the next hop is the destination. At the destination, the packet is verified by the checksum. Upon checksum mismatch, the destination node sends back a negative acknowledgment (NACK) to the sender, which subsequently retransmits the packet.

The loss management scheme in WCube has advantages over alternative solutions. Forward-error-correction (FEC) improves the wirelss channel reliability by detecting and correcting errors on the receivers but incurs fixed overhead over every packet. In fact, several error correcting codes have been proposed for wired NoC routers. However, the hardware implementation comes at a cost in both encoding and decoding logics.

6.5 Broadcast Support

We use a hierarchical approach to broadcast in WCube. We form a spanning-tree for each source wireless router, and a spanning-tree for each source base router inside each $WCube_0$ using the 2-D mesh. These spanning-trees are static in nature. For each $WCube_0$, each base router has a broadcast table with $m + 1$ entries. Each entry corresponds to a base router or the wireless router as the root of a spanning tree, and gives the next hops of the spanning tree. For each wireless router, it also contains a broadcast table in which each entry also corresponds to a wireless router as the root of a spanning tree. But the entry contains a field indicating whether the wireless router is a leaf node and a field describing the parent node of the wireless router. This is because wireless routers use broadcast channels to simultaneously connect to all neighbors, and we cannot ask a sender to only send to a subset of its neighbors. Instead, the receiver must decide whether it needs to accept a packet, based on whether or not the packet is from its parent node. The number of entries in the table is the number of wireless routers, which again is a small number.

The broadcast procedure then works as follows. When a source core broadcasts a packet, it sends the packet to its base router, the base router then forwards the packet to its next hops by looking up

[2]Request messages and data messages (between cores and L2 cache banks or between cores) are 8 bytes and 32 bytes, respectively.

its broadcast table. By doing so, the packet propagates along the spanning-tree in that $WCube_0$. When a wireless router receives a broadcast packet, if the packet is internally generated or from its parent node, it accepts the packet. Otherwise, the packet is discarded. If the wireless node is not a leaf node, it broadcasts the packets to all its logical neighbors.

6.6 Wireless MAC

A key design requirement for wireless MAC in WCube is operation simplicity, as it is implemented at the microarchitecture level. We cannot afford the powerful, yet sophisticated MAC mechanisms employed in today's conventional wireless networks, e.g., CSMA/CA, collision avoidance, and random backoff. While some prior work on on-chip UWB interconnect [34] uses a contention-based MAC, they assume to have separate wired controlling channels for arbitration among nodes.

Two distinctive features for wireless MAC in WCube are frequency-division multiple access (FDMA) and a cross-layer design for reliability management. WCube MAC uses a different frequency channel to deliver a packet at each wireless router. FDMA-based MAC effectively offers a dedicated link for each transmission. WCube MAC naturally supports multicast. In $WCube_n$, each router has one transmitter, as well as $2n$ receivers each of which tunes to a different frequency channel. Thus, each transmission is received by all $2n$ receivers, each at $2n$ logical neighbors in different levels. The reliability management for data transmissions in WCube MAC takes a cross-layer, on-demand approach. It receives information from the network-layer routing protocol at the router, regarding whether it is an intermediate hop or the last hop over WCube. It invokes overhearing, which incurs zero signaling overhead, as an intermediate-hop delivery. If it acts as the last hop, it will send a MAC-layer NACK when the router detects checksum errors in the packet at the network layer. In this way, wireless MAC in WCube reduces the signaling and communication overheads in reliability management.

7. EVALUATION

In this section, we evaluate the performance of our two-tier hybrid architecture in a 1000-node on-chip network. We use Garnet [17], a detailed on-chip interconnect network model simulation infrastructure that enables system-level performance and power modeling of network-level technique. Garnet models the detailed features of a state-of-the-art network such as 5-stage pipeline router design with wormhole switching, and it also includes the Orion power models [33] to report both dynamic and leakage power. Table 2 summarizes a list of the simulation parameters we used.

Parameter	Setting
Technology	32nm
Clock frequency	2 GHz
Number of cores on chip	1024
Number of processing cores	768
Number of L2 Caches	240 banks
Number of DRAM interfaces	16
Switching technique	Wormhole
Baseline topology	4-way concentrated mesh
Baseline mesh routing	XY routing
Baseline link bandwidth	16/8/4 bytes/cycle
Wireless link bandwidth	1 byte/cycle
Number of virtual channels	8 VCs
Number of wireless routers	16
Number of antennas in wireless router	1 TX / 4 RX antennas
Wireless backbone structure	2-level WCube
Size of $WCube_0$	16 base routers (64 nodes)

Table 2: Simulation parameters

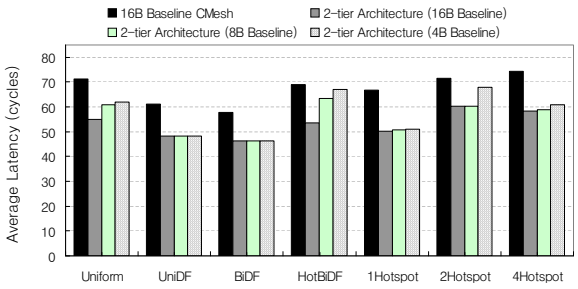


Figure 11: Latency reduction.

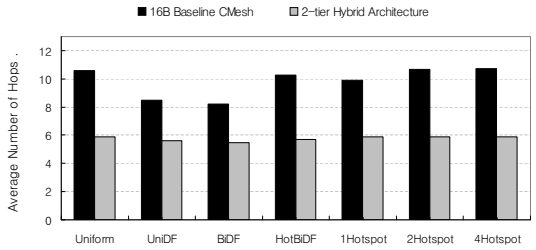


Figure 12: Hop count reduction.

In order to assess the interconnect demand of future applications, we construct synthetic traces to represent a variety of communication patterns for cooperative multithreaded applications. Each synthetic trace is executed on Garnet for 1 million network cycles. Our synthetic traces are based on the actual component placement in our 32×32 mesh (16×16 CMesh) design. We constructed seven total traces: *uniform*, *uniDF*, *biDF*, *hotbiDF*, *1Hotspot*, *2Hotspot*, and *4Hotspot* respectively. These traces are detailed in Table 3.

We use a 1024-node CMesh (16×16 base routers) as our baseline topology, and on top of that, we construct the WCube structure with 16 wireless routers. We choose the baseline CMesh that does not have the wireless backbone as our reference topology, since mesh networks are simple enough to be implemented on a 2-D silicon die, while other existing topologies would be impossible to build for a 1000-node on-chip network due to their structural scaling limitation. Our baseline topology uses 16B links between base routers.

Latency results. We first evaluate the network latency of our architecture. We measure the average packet latency in terms of cycles taken by packets to traverse the network from the source node to the destination node. Figure 11 presents the average packet latency for our seven probabilistic traces compared to the 16B baseline topology. In order to see the latency reduction by using the wireless backbone over the baseline mesh, we first focus on the performance of our 2-tier architecture built with the 16B baseline and a 2-level WCube. In the figure, we see a 20% reduction in latency on average, compared to the reference topology. Such latency reduction comes from the reduced hop count offered by WCube. Figure 12 shows the average hop count for packets traversing the network from the source node to the destination node. We see that, by opportunistically using the WCube, the packets are delivered via much shorter paths (nearly 40% decrease in hop count on average), compared to the mesh. However, the gain from hop count reduction is not directly translated to latency reduction, as seen from Figure 11, due to the fact that the bandwidth of wireless links is smaller than that of baseline links at the current CMOS technol-

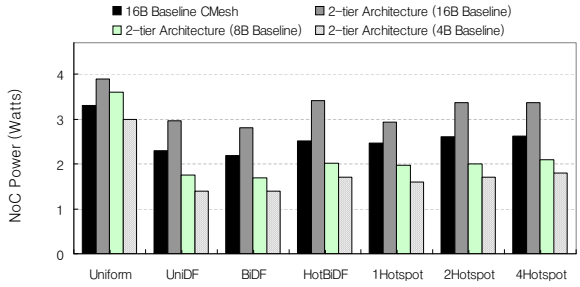


Figure 13: Power consumption.

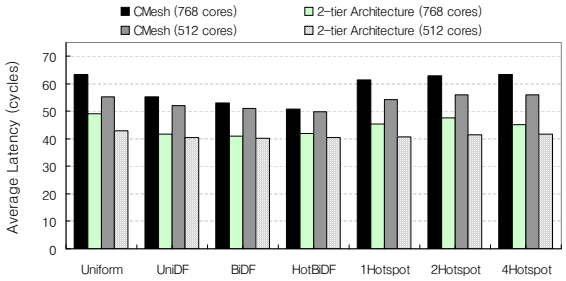


Figure 14: Latency comparison of different-sized CMPs.

ogy. According to ITRS [21], gain frequency f_T and power gain f_{max} will be 600GHz and 1 THz, respectively, in 16nm CMOS technology. As long as the carrier frequency increases at each new generation of technology, the number of available channels in the network can scale up, thus increasing the overall network aggregate data rate, and improving the on-chip wireless channel bandwidth up to 8 bytes/cycle. We have confirmed (results not shown) that with such a technology improvement, our 2-tier architecture can achieve a latency reduction of nearly 45% for our seven traces.

Power consumption results. In order to gauge the impact of wireless interconnect on future 1000-core CMPs, we perform power measurements using the Orion [33] power model to collect the data of router dynamic energy per flit and leakage power with various configurations. According to our result in Figure 3(b), wireless interconnect (with transmitter power -10dBm) energy consumes 4.5 pJ/bit. Using router, link and wireless interconnect power models, we present power-consumption as the average instantaneous power (in Watts) over the execution of an application. Figure 13 shows the results for power consumption. We see that our 2-tier architecture causes up to a 35% increase in NoC power, compared to the 16B baseline. This power increase is indeed the cost we need to pay for incorporating our wireless interconnect with the baseline structure.

Power efficiency enhancement. One approach to reducing the power consumption of the NoC with our approach is to simplify the underlying baseline topology of our 2-tier architecture [8]. If we use wireless interconnect to handle a large volume of our communication load, the underlying baseline topology can be simplified to enhance power efficiency. Figures 11 and 13 provide the results on latency and power when the link bandwidth of our baseline mesh is reduced from 16B to 8B and then to 4B (see the bars noted as 2-tier Architecture (8B Baseline) and (4B Baseline)). The results demonstrate the latency/power tradeoff, i.e., by reducing the baseline bandwidth, we can achieve a power savings at the cost of an up to 25% increase in latency. Nevertheless, a closer look at Figure 11 reveals that despite the baseline bandwidth reduction, *our 2-tier architecture still has lower latency than the 16B baseline mesh for all seven probabilistic traces, while it can save power consumption by up to 35%*. The results show that our 2-tier architecture built with the baseline mesh and WCube structure effectively reduces latency, while network energy consumption can be minimized by simplifying the baseline topology.

Uniform: A random traffic distribution - nodes are equally likely to communicate with all other nodes.
Dataflow: nodes clustered into groups which are functionally laid out in a dataflow-like fashion on our mesh. nodes are biased to communicate with nodes within their group and with nodes in groups that neighbor them on either one side (UniDF) or both sides (BiDF). This pattern would be seen in a data decomposition like medical imaging or a functional decomposition into a pipelined pattern (like an imaging pipeline or a cryptographic algorithm).
Hotspot: One or more nodes in the mesh are sending/receiving a disproportionate amount of traffic - a hotspot in the mesh. This can be exhibited by caches holding frequently used synchronization vars or a master/worker paradigm.
Hot Bidirectional Dataflow: The **Dataflow** pattern but with one group in the quadrant sending/receiving a disproportionate amount of traffic. This differs from **Hotspot** as communication is still biased in the dataflow pattern direction. This pattern could be seen in a pipelined parallel application where communication load is not evenly balanced across all parts of the pipeline.

Table 3: Trace patterns

Different-sized CMP results. In order to verify the adaptability of WCube, we also evaluate the network latency of our 2-tier architecture for different-sized CMPs. Figure 14 presents the latency comparison of our 2-tier architecture and 16B baseline reference topology in 512- and 768-core CMPs. As described in Section 5.5, a partial 2-level WCube is built on top of the corresponding baseline CMesh to support such core counts. For both core counts, our 2-tier architecture successfully reduces the latency on the order of 18-25% on average, compared to the reference topology across all traces. This result is consistent with the 1024-core CMP result, and we obtain similar results in other scenarios as well. The performance study shows the viability of our 2-tier hybrid architectural approach for future many-core CMPs.

8. DISCUSSION

In this section we discuss other issues relevant to our two-tier hybrid architecture.

Traffic patterns. We expect the overall traffic pattern for all applications running on many-core CMPs will be a mix of local and global traffic. A uniformly random traffic or all-to-all communication pattern is very unlikely in most applications. Summing up all applications running on CMPs, we expect that the dominant traffic seen will still be sent to local cores, and that traffic to more distant destinations will be a relatively small but non-negligible proportion of communication. WCube works best for this particular scenario.

Congestion awareness. Our two-tier architecture successfully reduces the network latency in general, but it is possible that wireless routers could become an NoC bottleneck if too many packets try to use the wireless backbone at once, causing buffers to fill up at wireless router entrances. While we do not explore the impact of the worst-case adversarial traffic on performance in this paper, some type of congestion control mechanism is necessary due to the fact that on-chip wireless channel bandwidth is not yet comparable to the wire link at the current technology.

Scalability issue. While WCube is a scalable structure, there are several factors that could impact the scalability of the structure.

One prominent concern is about the receive antenna spacing at each wireless router. In an n-level WCube, each wireless router has $2n$ receivers. As the core count grows, the number of receive antennas increases and thus, antenna spacing within each wireless router will reduce with n, which may cause inter-channel interference. However, we believe that there will be a practical upper bound on future CMP core count (e.g., high hundreds, low thousands of cores) [5, 22]. In practice, a WCube with a small level (e.g., n=3) can accommodate as many as several thousands of cores.

Alternative interconnect structure. One future direction to further enhance network performance is to build a wireless interconnect on top of the RF-I transmission lines. Since RF-I achieves latency reduction by offering shortcuts in mid-sized networks (in the range of tens to lower hundreds of cores) [8], it can bridge the gap between the baseline mesh and our wireless interconnect. With three levels of hierarchy, local messages would go through the baseline RC wires (e.g., within a $WCube_0$), mid-range messages would be forwarded via RF-I transmission lines that span over multiple $WCube_0$s, and only the long-range messages would be delivered using wireless interconnects. Our work in this paper serves as the basis for this future direction.

9. CONCLUSION

In this paper, we make a case for using a two-tier wireless and wired architecture to interconnect hundreds to thousands of cores on a system-on-chip. The wireless express way eliminates long wires and reduces latency for long-haul, many-hop, inter-core communication in a way that is not possible with today's wired interconnect technology. To this end, we propose a recursive, wireless interconnect structure called the WCube, which features a single transmit antenna and multiple receive antennas at each micro wireless router and offers scalable performance in terms of latency and connectivity. We further devise a new wormhole-based, two-tier routing algorithm that is deadlock free and ensures minimum-latency route. The early evaluation result of a $20 \sim 45\%$ latency reduction is also quite significant from the CMP interconnect perspective.

Designing wireless interconnect at the microarchitecture level also opens new research opportunities for the wireless networking community. The miniature antenna, together with simple transceiver circuits, enables us to build multiple transmitters, and/or multiple receivers at each micro wireless router. This enables the application and deployment of many cooperative wireless communication and networking techniques. The $100 - 500$ GHz, sub-terahertz frequency band does not require sophisticated transceiver design to achieve high data rates of 10s of Gbps. But operation simplicity is a key requirement for microarchitectures. All of these make cases for new wireless networking solutions efficiently operating and sharing in the frequency domain.

10. ACKNOWLEDGMENTS

We greatly appreciate the insightful comments by our shepherd, Dr. David Wetherall and the anonymous reviewers for their constructive feedback. This work was supported in part by SRC grant #1796, and the U.S. Army Research Laboratory and the U.K. Ministry of Defense under Agreement Number W911NF-06-3-0001.

11. REFERENCES

[1] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, and D. Burger. Clock rate versus IPC: the end of the road for conventional microarchitecture. *ISCA-27*, 2000.
[2] J. Andrews and N. Backer. Xbox360 system architecture. *Hot Chips*, 2005.
[3] Ansoft Corporation. High Frequency Structure Simulator (HFSS). http://www.ansoft.com/products/hf/hfss/
[4] K. Asanovic et al. The landscape of parallel computing research: a view from Berkeley. Technical Report, UCB/EECS-2006-183.
[5] S. Borkar. Thousand core chips: a technology perspective. *DAC*, 2007.
[6] L. A. Barroso et al. Piranha: a scalable architecture based on single-chip multiprocessing. *ISCA-27*, 2000.
[7] M.-C. F. Chang et al. CMP network-on-chip overlaid with multi-band RF-Interconnect. *HPCA*, 2008.
[8] M.-C. F. Chang et al. Power reduction of CMP communication networks via RF-Interconnects. *MICRO*, 2008.
[9] D. Choudhury, J. Foschaar, R. Bowen, M. Mokhtari. A 70+GHz BW package for multigigabit IC applications. *Microwave Symposium Digest*, June 2004.
[10] S. Boyd-Wickizer et al. Corey: an operating system for many cores. *OSDI*, 2006.
[11] W. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. on Computers*, 1987.
[12] W. Dally. Virtual-channel flow control. *IEEE Trans. on Parallel and Distributed Systems*, 1992.
[13] W. Dally. Wire efficient VLSI multiprocessor communication networks. *Proc. Stanford Conf. Advanced Research VLSI*, 1987.
[14] D. Huang et al. Terahertz CMOS frequency generator using linear superposition technique. *IEEE Journal of Solid State Circuits*, Dec 2008.
[15] A. Duller, G. Panesar, and D. Towner. Parallel Processing - the picoChip way!. *Communicating Process Architectures*, 2003.
[16] B. A. Floyd. Intra-chip wireless interconnect for clock distribution implemented with integrated antennas, receivers, and transmitters. *IEEE JSSC*, 2002.
[17] N. Agarwal et al. Garnet: A detailed interconnection network model inside a full-system simulation framework. TR CE-P08-001, Princeton University, 2007.
[18] C. J. Glass, L. M. Ni. The turn model for adaptive routing. *ISCA-19*, 1992.
[19] A. Ghuloum, Unwelcome advice from Intel. blogs.intel.com/research/2008/06/unwelcome_advice.php
[20] B. Grot and S. W. Keckler. Scalable on-chip interconnect topologies. *2nd Workshop on Chip Multiprocessor Memory Systems and Interconnects*, 2008.
[21] International technology roadmap for semiconductors, 2007 edition. http://www.itrs.net/Links/2007ITRS/2007_Chapter/ 2007_Wireless.pdf
[22] D. N. Jayasimha, B. Zafar, Y. Hoskote. On-chip interconnection networks: why they are different and how to compare them. Technical Report, Intel Corp, 2006
[23] J. Kahle, M. Day, H. Hofstee, C. Johns, T. Maeurer and D. Shippy. Introduction to the Cell multiprocessor. *IBM Journal of Research and Development*, 2005.
[24] J. Kim, J. Balfour, and W. Dally. Flattened butterfly topology for on-chip networks. *MICRO*, 2007.
[25] G. Koch. Intel's road to multi-core chip architecture. www.intel.com/cd/00/00/22/09/220997_220997.pdf
[26] NVIDIA Quadro FX 5600. http://www.nvidia.com/docs/IO/40049/quadro_fx_5600_datasheet.pdf
[27] NVIDIA Tesla C1060. http://www.nvidia.com/docs/IO/56483/ Tesla_C1060_boardSpec_v03.pdf
[28] K. Olukotun and L. Hammond. The future of microprocessors. *ACM QUEUE Magazine*, September 2005.
[29] K. Olukotun, L. Hammond, and J. Laudon. Chip multiprocessor architecture: techniques to improve throughput and latency. *Morgan & Claypool*, 2007.
[30] S. -W. Tam et al. A simultaneous tri-band on-chip RF-Interconnect for future network-on-chip. *VLSI Symposium*, 2009.
[31] E. Seok et al. A 410GHz CMOS push-push oscillator with an on-chip patch antenna. *ISSCC*, 2008.
[32] S. Vangal et al. An 80-tile 1.28 TFLOPS network-on-chip in 65nm CMOS. *IEEE ISSCC*, 2007.
[33] H. -S. Wang et al. Orion: a power-performance simulator for interconnection networks. *Int' Symposium on Microarchitecture*, 2002.
[34] D. Zhao and Y. Wang. SD-MAC: design and synthesis of a hardware-efficient collition-free QoS-aware MAC protocol for wireless network-on-chip. *IEEE Transactions on Computers*, Vol.57, No.9, September 2008.

APPENDIX

A. Proof of Theorem 1

PROOF. A routing algorithm is deadlock-free if the network channels can be enumerated such that the algorithm always routes the packets along the channels with strictly decreasing numbers. We label each channel in the network as follows: (1) for each base router R_i, label k output $*$Up channels $c^{*U}_{2,0,i}, \cdots, c^{*U}_{2,k-1,i}$, and label l output $*$Down channels $c^{*D}_{0,0,i}, \cdots, c^{*D}_{0,l-1,i}$; (2) for each wireless router R_i, label m output logical channels $c^{W}_{1,0,i}, \cdots, c^{W}_{1,m-1,i}$. Since our mesh routing and WCube routing use channel ordering, each of $c^{*U}_{2,j,i}$, $c^{*D}_{0,j,i}$, and $c^{W}_{1,j,i}$ $(\forall i, j)$ is independently deadlock free. Now, by our channel labeling, $c^{*U}_{2,j,i} > c^{W}_{1,j,i} > c^{*D}_{0,j,i}$, $\forall i, j$. Thus, our 2-tier routing is deadlock free. □