Polaris: A System-Level Roadmap for On-Chip Interconnection Networks

Vassos Soteriou†, Noel Eisley†, Hangsheng Wang‡, Bin Li†, Li-Shiuan Peh†
†Department of Electrical Engineering
Princeton University
Princeton, NJ 08544
{soteriou,eisley,binl,peh}@princeton.edu

‡Freescale Semiconductor
7700 W Parmer LN
Austin, TX 78729
hangshen@freescale.com

***Abstract*— Technology trends are driving parallel on-chip architectures in the form of multi-processor systems-on-a-chip (MPSoCs) and chip multi-processors (CMPs). In these systems the increasing on-chip communication demand among the computation elements necessitates the use of scalable, high-bandwidth network-on-chip (NoC) fabrics. As transistor feature sizes are further miniaturized leading to rapidly increasing amounts of on-chip resources, more complicated and powerful NoC architectures become feasible that can support more sophisticated and demanding applications. Given the myriad emerging software-hardware combinations, for cost-effectiveness, a system designer critically needs to prune this widening NoC design space to identify the architecture(s) that best balance(s) cost/performance, before the actual design process begins. This prompted us to develop Polaris[1], a system-level roadmap for on-chip interconnection networks that guides designers towards the most suitable network design(s) tailored to their performance needs and power/silicon area constraints with respect to a range of applications that will run over this network(s). Polaris explores the plethora of NoC designs based on projections of network traffic, architectures, and process characteristics. While the Polaris roadmapping toolchain is extensible so new traffic, network designs, and processes can be added, the current version of the roadmap already incorporates 7,872 NoC design points. Polaris is rapid and iterates over all these NoC architectures within a tractable run time of 125 hours on a typical desktop machine, while maintaining high relative and absolute accuracies when validated against detailed NoC synthesis results.**

I. Introduction

The International Technology Roadmap for Semiconductors (ITRS) [25] projects that it will soon be feasible to design multi-billion transistor chips. Increasing design complexity and diminishing returns from uniprocessor optimizations have led to the emergence of multi-core architectures in the form of multi-processor systems-on-a-chip (MPSoCs) and chip multi-processors (CMPs). The growing number of on-chip resources in combination with the diverse range of current and future applications that these parallel systems are to run are sparking a widening design space of multi-core architecture implementations. Architects and designers are therefore faced with the tough challenge of selecting the most suitable parallel system design that will best balance their application requirements in terms of the anticipated cost-benefits.

While ITRS has served as an indispensable guide for circuit designers and researchers, providing well-calibrated projections into future device, interconnect, and technology parameters such as wire spacing, capacitance, and voltages, no system-level living roadmap (SLLR) [8] exists to guide system designers in their navigation of the vast multi-core design space. Similar to the goals of ITRS in projecting potential low-level device and component parameters, a SLLR projects potential high-level architecture parameters. As Figure 1-A shows, a 3-phase SLLR projects system-level parameters: (1) the application workload that a wide range of multi-core on-chip systems are targeted for; (2) the multi-core system architecture components such as processor types, cache sizes and organizations, network topologies, number of processing cores, etc.; and (3) the associated cost/performance characteristics such as latency, power consumption and silicon area of each system design. These metrics projected by a SLLR can guide a designer in the selection of a subset of parallel systems that best meet cost-benefit expectations before the detailed system design process begins.

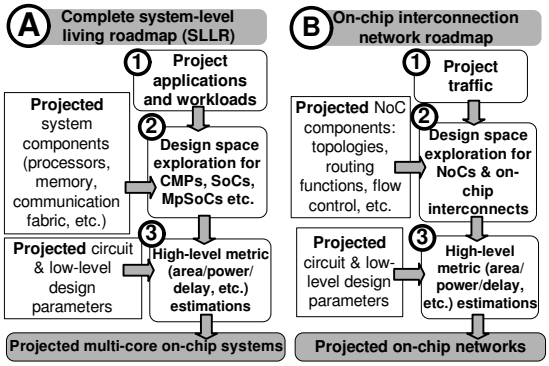

Fig. 1. Major phases in (A) complete multi-core system-level roadmapping versus (B) roadmapping for the communication subsystem.

In this paper, we take the first step towards such a SLLR, focusing on a key subcomponent of a multi-core on-chip system: its communication subsystem, the on-chip interconnection network or network-on-chip[2] (NoC). With chips increasingly composed of multiple processing elements, the impact of the design of the communication subsystem on overall chip performance, silicon area, and power, is becoming increasingly crucial [6], [4]. As the demand for bandwidth increases, on-chip networks are becoming the de facto interconnect fabric in multi-core systems-on-a-chip (SoCs), MPSoCs and CMPs, leading to a corresponding explosion in the multi-core on-chip design space.

This motivates us to propose Polaris, a system-level roadmap for on-chip networks that guides designers towards a subset of most suitable candidates for on-chip network designs while considering the complex tradeoffs between applications, architectures, and technologies. As shown in Figure 1-B, similar to a SLLR for the entire multi-core on-chip system, a NoC roadmapping toolchain consists of three equivalent phases: (1) the projection of workloads, in this case the network traffic among the cores of the various multi-core architectures; (2) the projection of several NoC components and subsequently the exploration of the vast design space of NoCs; and (3) the projection of circuit and technology parameters. This 3-phase system-level NoC projection toolchain derives high-level estimates of cost/performance tradeoffs of each NoC design.

Each of the three phases of the NoC SLLR encounters sig-

[1]Polaris is a star in our galaxy that has been used as a guiding light by sailors for centuries.

[2]NoCs and on-chip interconnection networks are used interchangeably in this paper. Tiles, nodes and routers are also used equivalently.

nificant challenges. First, network traffic projections are faced with the plethora of traffic patterns arising from the myriad current and potential architectures as well as the diverse range of applications and their network mappings [21], [24], [29]. It is therefore important to categorize traffic using a small range of spatio-temporal parameters that comprehensively capture the behavior of traffic. This enables a designer to realistically capture various traffic behaviors by using a representative set of projected applications that may not yet be exercised with real benchmark applications, an especially important issue in a relatively immature field such as NoCs. Additionally, a relatively small number of spatio-temporal variables enables designers to intuitively understand a wide spectrum of current and emerging traffic patterns. Polaris tackles this with a traffic model based on three statistical variables that can capture diverse traffic patterns, allowing designers to pick representative traffic categories in driving NoC roadmapping. Second, the design space exploration toolchain has to be rapid. With a huge design space to be explored, cycle-level simulations or circuit-level investigations will not be tractable. A rapid, yet relatively accurate toolchain is necessary. Currently, Polaris encompasses a design space of 21 different network architectures, each with 24 different network micro-architecture configurations (the binary fat trees examined in this paper possess 12 different network micro-architecture configurations) at 2 process technologies ($90nm$ and $50nm$), giving rise to a total of 984 distinct network designs. Coupled to 8 distinct traffic categories, Polaris currently explores 7,872 design points, each representing a unique traffic-architecture combination. Polaris iterates through these in a manageable 125 hours on a typical desktop machine[3]. Finally, technology scaling alters the complex tradeoffs between alternative designs at each technology node, so the high-level area/power/delay projections have to judiciously project suitable circuit and technology parameters while maintaining relative accuracy. Our validation results in Section IV show Polaris's high relative and absolute accuracies versus detailed NoC synthesis using Verilog.

Next, Section II compares NoC roadmapping with prior related work in SoC synthesis in order to show the differing design goals of the two toolchains. Following, Section III presents Polaris's roadmapping toolchain, composed of several tools heavily modified and integrated for system-level roadmapping. Section IV validates the toolchain against low-level synthesis tools while Section V presents Polaris's projections and insights on current state-of-the-art and future NoC designs. Finally, Section VI concludes the paper.

II. Comparing NoC Roadmapping and NoC Synthesis

We compare NoC roadmapping with prior related research in NoC synthesis. Figure 2-A sketches a typical NoC synthesis toolchain (NoCSyn), where the first two phases are based on the SUNMAP [15] tool and the third phase is based on the ×pipesCompiler [12] tool, while Figure 2-B sketches a NoC roadmapping toolchain (NoCRoad) based on Polaris. Similar NoCSyn flows have also been exhibited in a number of other synthesis toolchains, such as in the work by Pestana et al. [20] that uses XML-based simulation to find NoC cost/peformance instances, in the work by Horn et al. [10] that provides a mechanism for cost optimal construction of irregular networks based on three design criteria, Metropolis [9], an environment that uses compositional modeling to assemble SoC components so that their composition satisfies a given set of design constraints and properties, and in the work by Ögras and Marculescu [16] that presents a methodology for synthesizing customized communication architectures that are tailored to the communication requirements of a target application.

NoCRoad provides power/area/delay estimates for each tested NoC design to form a *large set* of projections. A designer can then prune this entire set of projections and pick a subset of designs that best suits the designer's projected design constraints from the tables of results, *before* the designer invests in detailed evaluation of these designs. NoCRoad can also be used as a complementary pre-processing tool for NoCSyn. In other words, if NoCRoad is used in conjunction with current process technology libraries, NoCRoad can point to a subset of NoC designs that can then be implemented in detail using a NoCSyn engine, thus leading to efficient exploration and implementation of NoCs. NoC roadmapping does not replace NoC synthesis. As it will be explained in more detail, NoCRoad's scope is about NoC *projection* and *exploration* while the scope of NoCSyn engine is about NoC *selection* and *generation* [15]. Since NoCRoad and NoCSyn have different goals the two toolchains cannot be contrasted to highlight each one's performance, rather, they can be compared to expose their individual goals in the design of NoCs. In detail, we refer to Figure 2 to show the differences of NoC synthesis (Figure 2-A) and roadmapping with NoCRoad (Figure 2-B) phase-by-phase.

Phase 1 of NoCSyn performs application traffic analysis and mapping. A *specific* application that the SoC is to run (e.g. MPEG video) is evaluated using simulations or abstracted using static analysis to capture the amount of inter-core communication. Design constraints such as area/power/latency/throughput are fed into the toolchain together with a user-chosen routing function, a limited set of topologies and NoC architecture parameters known a priori, along with power/area/delay libraries at current process technologies. Early power/area/delay and traffic bandwidth estimates are then calculated for each application-mapped NoC topology. In comparison, Phase 1 of NoCRoad performs application traffic projections. Based on statistical analysis, NoCRoad enables the representation of a wide range of applications using a small number of parameters that accurately capture the spatio-temporal characteristics of traffic. Armed with these spatio-temporal parameters, NoCRoad then synthesizes and thereafter categorizes traffic into a *set* of representative traffic groups. This enables a designer to capture various traffic behaviors by choosing a representative set of *projected* traffic that may not yet correspond to actual executing application traffic traces.

Phase 2 of NoCSyn is about topology *selection*. In detail, NoCSyn draws several known process technology and specific architecture parameters to evaluate the various topology choices in terms of the user-defined area/power/delay expectations. To achieve this, NoCSyn uses lower-level tools to accurately determine whether the *constraints* from Phase 1 have been met. Based on these evaluations and analysis, NoCSyn then *selects* a single NoC design out of the set of application-mapped topologies from Phase 1. In comparison, NoCRoad's Phase 2 performs high-level NoC design-space exploration. NoCRoad uses a high-level framework to rapidly *explore* a huge NoC design space that can consist of 10s of thousands of NoC design points. Using the various traffic categories from Phase 1, along with a wide range of future projected micro-architecture configurations, routing functions, flow control protocols, and wire and pipeline delay models, it rapidly analyzes each NoC's level of utilization. This high-level analysis enables NoCRoad to uncover the points of contention and their effects in a wide NoC design space through *analysis* rather than by running simulations. Here, absolute accuracy as in the case of NoCSyn is not necessary, but relative accuracy is important. No design constraints are input and examined in this phase as the toolchain *explores* rather than *selects* NoC design alternatives.

The third and final Phase of NoCSyn carries out synthesis, simulation, and verification. Here NoCSyn uses several back-end tools to instantiate (by drawing in several macro or mid-level NoC components) and then simulate the chosen NoC description, simultaneously verifying that the user-defined

[3]Each design point runs for half a million cycles of simulation on a $2.4GHz$ Pentium 4-based single processor machine with 1.25GB of RAM.

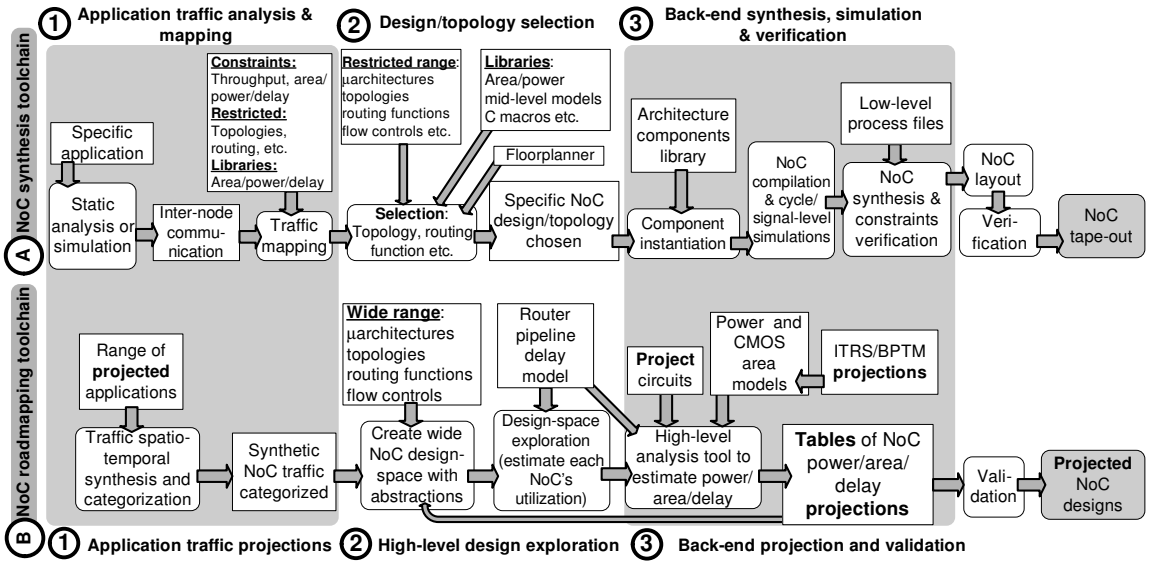

Fig. 2. (A) a typical 3-phase NoC synthesis engine toolchain and (B) Polaris's 3-phase roadmapping toolchain.

design constraints and goals have been met. Finally, the NoC is synthesized into gate, transistor or macro levels by employing lower-level circuit libraries. The detailed NoC low-level silicon layout is then drawn, verified, and sent for tape-out. In comparison, NoCRoad's third phase is concerned with NoC projections and validation. Here, NoCRoad draws in low-level technology, interconnect, and device projections from ITRS [25] and BPTM [2]. Using these low-level parameters within a high-level framework, NoCRoad then *projects* the NoC topology floorplan, router pipeline structures and the circuitry of all explored NoC designs, and then provides estimations of various metrics of interest: area/power/delay and various other combinations of results.

Research in system-level roadmapping is in its infancy. There have been several works [13], [1] that proposed high-level models of key system metrics such as reliability and variability. An interesting future direction is to explore factoring these models into the third phase of NoCRoad so that variability-aware metrics can also be projected. The closest work to NoCRoad is that by Wang et al. [34] which proposed an analytical design-space exploration tool for projecting NoC topologies at future technologies. However, it only explores topologies and a single metric: network power.

III. Polaris Toolchain

Polaris builds upon the interfacing of three tools where each tool corresponds to each of the three phases depicted in Figure 2-B. These three tools are (1) Trident [28], a synthetic traffic constructor/analyzer, (2) LUNA [7], an analytical framework that estimates network resource utilization for a wide spectrum of architectures to calculate network delay and power, and (3) Orion [32] a library of power, area and router pipeline delay models that is used to estimate power and CMOS area for each network configuration at every process technology node. Each such phase corresponds to the equivalent phase of the detailed roadmapping toolchain presented in Section II.

In the first phase an architect chooses an NoC traffic category based upon a set of spatio-temporal parameters that describe the form of that class of traffic. These three statistical components model the *hop count*, *burstiness*, and *packet injection distribution* of on-chip traffic. Given these statistical parameters Trident then automatically generates synthetic NoC traces that are fed into Phase 2 of Polaris. In Phase 2 LUNA consumes these traces and analyzes the utilizations at each router node and link of all NoC architectures under consideration. This vast NoC design space is built from permutations of various topologies, architecture parameters such as buffer sizes in terms of flits[4], flit width, flow control protocols, and routing functions. In Phase 3, these NoC utilization profiles are then fed into Orion to estimate power consumption and network delay at each process technology. Power/delay/area metrics are then output for each NoC architecture-traffic design point combination to form a huge table of results.

These three tools were interfaced to each other and significantly modified from their original implementations so as to allow Polaris to model various NoC architectures. Our contributions to the extensions of these tools are as follows:

(1) Trident was modified to accommodate traffic categorization or grouping. This allows traffic to be classified into various representative groups depending upon the spatio-temporal characteristics of traffic. This enables a designer to capture various traffic behaviors by using a representative set of projected applications that may not yet be exercised with real benchmark applications. This is especially important in a relatively immature field like NoCs, and for projecting into future technologies.

(2) LUNA, a tool originally proposed for NoC power analysis, was extended to include delay analysis and interfaced with Orion so that Orion's projected router pipeline delay models are factored in at each node and consequently in LUNA's delay analysis. LUNA originally modeled 2-dimensional mesh networks with static routing. LUNA was heavily modified to model 12 types of topologies such as binary fat trees, rings and hierarchical and express cube variants (see Figure 3). LUNA was also extended to incorporate adaptive routing as well, virtual cut-through and wormhole flow controls. LUNA now resolves and estimates delay using additional parameters that were added to Orion (see below).

(3) Orion was modified to operate in a statistical mode where all activity in a router is abstracted as one single router load. This enables Orion to operate much faster than in its original cycle-by-cycle mode. Area models were also added to Orion enabling it to estimate the silicon area required by the various components of a NoC topology. Orion now incorporates a delay model which depends on the pipeline lengths of routers used in the various NoC topologies and wire lengths (see Table I), and delay caused due to congestion, where the latter is estimated by LUNA via traffic analysis. Power models at current state-of-the-art $90nm$ and future $50nm$ process technologies were also incorporated.

[4]Flit stands for "flow control unit," a fixed-size segment of a packet.

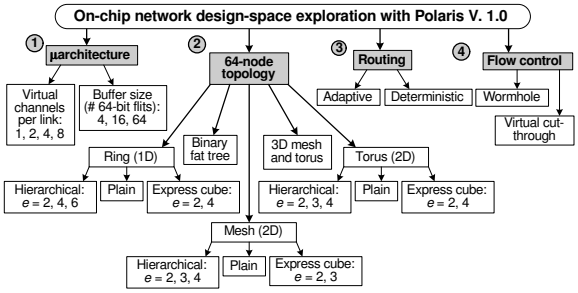

Fig. 3. NoC design-space currently explored by Polaris. "e" denotes the express cube or hierarchical link interval.

We next elaborate on the details of each component in Polaris.

A. 3-Tuple Statistical Traffic Model

Trident [28] is a tool developed to both (1) model statistically and (2) construct traffic based on three parameters, referred to as a *3-tuple*. The extraction of these three parameters is based on empirical analysis carried out on a representative set of real traces from three existing NoC architectures: the Raw CMP [29], TRIPS CMP [24], and a 16-tile conventional directory-based cache-coherent shared-memory CMP[5]. These three parameters allow Trident to capture the spatio-temporal characteristics of NoC traffic comprehensively and accurately and with less than 5% error when validated against the above actual NoC application traces. The 3-tuple comprises:

(1) Traffic burstiness. It models how often packet bursts are injected into network routers and how large these bursts are. It was found in the work by Soteriou et al. [28] that on-chip traffic exhibits self-similarity or "scale-invariant burstiness" [18], where traffic burst patterns repeat themselves over time. Trident models burstiness using the Hurst parameter, $0.5 < H \leq 1$, which defines the level of self-similarity. The closer H is to 1 the higher the level of burstiness.

(2) Traffic injection distribution. It models how packet injection streams are distributed among the various nodes in an NoC. Packet injections were found in the work of Soteriou et al. [28] to possess a Gaussian injection distribution when the aggregated injection peaks are graphed. Using a normalized standard deviation, σ_{Gauss}, to decouple the size of the network to the actual packet injection distributions, hot-spot or evened-out traffic can be modeled where a relatively larger σ_{Gauss} models a more evened-out injected traffic pattern.

(3) Traffic hop distance. With a single statistical parameter it models how long packets travel from source to destination, capturing long and short-distance traffic with respect to the topology's size and type, and consequently in relation to the maximum hop distance of a NoC.

The three statistical parameters are unitless and orthogonal to each other allowing Trident to generate a spectrum of synthetic traffic with various statistical characteristics. For instance, hot-spot traffic which is highly bursty and travels long distances (high hop count) can be generated given the three parameters as inputs by a designer. These synthetic traces contain timestamps and source-destination router coordinates that are fed into LUNA in Polaris as Section III-B describes. The reason for using synthetic traffic traces generated with Trident instead of using traffic traces obtained from parallelized benchmarks is that the use of our statistical traffic model allows us to extensively cover all traffic spatio-temporal forms, including those that are not precisely captured by real traces. This allows NoC designers to conjecture and predict the traffic for future NoC applications and designs before detailed traces are available.

Though Trident in its stand-alone mode can generate traffic of theoretically infinite spatio-temporal permutations, for the purposes of roadmapping traffic has to be categorized. This categorization enables a designer to more readily pick representative traffic sets, while covering a wide spectrum of traffic spatio-temporal characteristics without explicitly having to run parallelized benchmarks. Trident in Polaris was thus modified to accommodate this requirement. Traffic categorization in Polaris is derived empirically. First, Polaris calculates the statistical 3-tuples of 30 representative traffic traces obtained from simulations of benchmarks running on the 3 diverse architectures, from two message-passing CMP architectures [24], [29] and the cache-coherent CMP described earlier. As already described, these 3-tuples capture the burstiness, hop distance and injection distributions of the above traces. Second, these 3-tuples are plotted on a 3-dimensional space, with each axis representing a unique component of the 3-tuple. Polaris then observes for possible clusterings of real traffic 3-tuples in the three-dimensional space and based on these groupings of statistical parameters various categories of 3-tuples are formed. In this paper we consider 8 such categories; a user can change this number, although here we choose 8 so that the presentation of results in Section V becomes doable. These spatio-temporal traffic groupings are based on permutations of moderate and highly bursty traffic, global (long hop distance) and local traffic, and hot-spot and evened-out injected traffic. Details of the resulting categories from the spatio-temporal traffic clusterings follow.

Based on the empirical measurements of the 3-tuples of the 30 real application traces, short distance traffic is classified as the traffic where only 20% of the total injected traffic traverses distances greater than four hops; for long distance traffic this value rises to 40%. Polaris classifies highly bursty traffic with an $H = 0.9$ and moderately bursty with $H = 0.65$. Traffic injections are categorized as hot-spot and evened-out traffic. Hot spots are modeled such that 10% of the nodes (~ 6 for 64-node NoCs considered in the experiments of Section V) receive 64% of the total injected traffic ($\pm\sigma_{Gauss}$, or standard deviations around the Gaussian mean μ_{Gauss}), and evened-out traffic such that 20% of the nodes receive the same portion of 64% of the traffic. Since traffic is classified into two bins per statistical parameter of the 3-tuple, here we define 8 such traffic categories. Note that Trident in Polaris provides the flexibility to change how traffic is categorized/classified based on empirical measurements carried out on other sets of traffic and on a designer's intuition.

B. Network Utilization Analysis and Network Delay Estimation

LUNA's analytical framework captures both (1) *temporally* the amount of traffic across the entire duration of traffic flow and (2) the traffic *spatially* distributed across all nodes in the network, to estimate the levels of network resource utilization and consequently the level of network activity. Details of LUNA's framework can be found at its release web site [7] and are omitted here for brevity. LUNA was chosen in constructing Polaris for two reasons: to meet the challenge of implementing a fast roadmapping toolchain, with LUNA shown to be up to 360X faster and second to sustain high relative accuracy within 5.9% of error when compared against cycle-accurate simulators.

LUNA was heavily modified to accommodate the various network topologies, routing functions and flow-control protocols, and to model network delay and estimate power. Figure 3 shows the network configurations built into LUNA.

Network topologies and flow control. LUNA abstracts a network topology as a graph where network routers are mapped to the set of nodes and links to the set of edges. LUNA

[5]The cache-coherent CMP comprises 16 tiles each with an in-order processor core, 32KB write-through L1 data and instruction 4-way set-associative cache, and 4MB shared L2 16-way set-associative data cache with write-back. All caches consist of 32-byte blocks. Cache coherence is maintained through a directory-based MSI protocol. The cache-coherent CMP is simulated on the Liberty Simulation Environment [30] with traces obtained from the 16-node memory network interconnected in a 4×4 mesh array.

TABLE I
WIRE DELAYS IN TERMS OF CYCLES GIVEN THE WIRE LENGTH IN TERMS OF HOPS AND PIPELINE DELAYS BASED ON VARIOUS PHYSICAL CHANNEL (LINK) PER ROUTER (p) AND VIRTUAL CHANNEL (v) PER LINK (p,v) COMBINATIONS FOR PACKETS CONSISTING OF 64-BIT FLITS.

Hop length	1	2	3	4	5	6	7
90*nm* delay (cycles)	1	1	1	1	1	1	2
50*nm* delay (cycles)	1	2	2	3	3	4	4

(p,v)	Pipe length		(p,v)	Pipe length		(p,v)	Pipe length	
	90*nm*	50*nm*		90*nm*	50*nm*		90*nm*	50*nm*
(1+2, 1)	4	5	(1+4, 4)	5	7	(1+8, 1)	5	6
(1+2, 2)	5	6	(1+4, 8)	6	7	(1+8, 2)	5	7
(1+2, 4)	5	6	(1+6, 1)	5	5	(1+8, 4)	6	8
(1+2, 8)	6	7	(1+6, 2)	5	6	(1+8, 8)	6	8
(1+4, 1)	4	5	(1+6, 4)	5	7			
(1+4, 2)	5	6	(1+6, 8)	6	7			

is also extended to store information of each link's length and bandwidth, as well as the number of router pipeline stages and links at a router so as to accurately measure delay and power consumption when Orion's power-delay-area models are coupled with LUNA (see Section III-C).

Delay modeling. LUNA is extended to calculate the latency-throughput of traffic (in the form of a trace generated by Trident) given a network topology by summing (1) the zero-load delay and (2) the delay due to contention. The former is calculated by considering the number of pipeline stages in a router and the packet size in terms of flits, given by the expression $Network_{latency}$ = *(#pipeline stages*×*average traffic hop count)*+$(flits\ per\ packet - 1)$. This delay expression is applicable to both flow controls incorporated into LUNA: (1) wormhole and (2) virtual-cut through. LUNA also resolves contention-related delays based on the link bandwidth, number of buffers and number of virtual channels per link. It adds these additional delays to the above base delay to determine the final average network delay of a traffic-network combination. The router pipeline delay in terms of pipeline stages is derived from the delay model of Peh and Dally [19] and from ITRS frequency and pipeline stage delays in terms of fan-out-of-4 (FO4) inverter delay projections. Pipeline delays in terms of clock cycles are shown in Table I. These projections are fed into both LUNA and Orion to measure the effects on network latency and power consumption respectively, with more details about the delay model described in Section III-C.

Routing. LUNA supports both deterministic and adaptive routing functions, with both functions routing minimally (i.e. these routing functions route within a minimum rectangle). Deterministic dimension-ordered routing is modeled, with priority given to express channels if they lie along a packet's routing path. LUNA is extended to incorporate adaptive routing, by evenly distributing the packet injection rates among minimal paths between the source and destination nodes.

C. Power, Area and Delay Modeling

Orion [32] comprises architecture-level power models, targeted to various process technologies, of major building blocks in an interconnection network: buffers, crossbars, arbiters and channels [12]. For each building block, Orion models the static and dynamic power of several typical circuit implementations, and takes as input a range of parameters including clock frequencies, voltages, circuit structures, link lengths, etc. In this paper, we project these parameters based on our projected topology floorplan and with underlying device and interconnect projections from ITRS-2002 and 2004 [25], unless otherwise stated. We also extended Orion to model area in addition to power, and incorporated the pipeline delay model of Peh and Dally [19] into Orion to capture the effects on area and power estimates. We project the more preferable circuit structures at current state-of-the-art $90nm$ and future $50nm$ technologies before inputting them into Orion. For instance, a matrix crossbar circuit was found to be more energy efficient than a multiplexer-based crossbar at these technologies. Likewise, SRAM buffers instead of shift-registers are assumed as the former is more energy-efficient [33].

Interfacing with LUNA. Orion is used in tandem with LUNA to estimate CMOS area and power consumption for each NoC configuration. Instead of using Orion's original activity-accurate mode [32], where Orion is integrated into a cycle-accurate simulator to measure cycle-accurate switching activity in a router to calculate power consumption, here Orion was modified to work in a statistical mode.

In the statistical mode, all activity in a router is abstracted as one single router load: the average number of flits arriving at/leaving from every input/output port per cycle. From this number, Orion automatically derives the switching activity at every FIFO buffer, crossbar and arbiter within the router, assuming a 50% bit-level switching probability. This mode is much faster than the activity-accurate mode, and it is also fairly accurate if no switching activity oriented power optimization is in use. It must be noted that this mode does not distinguish between the various components present in a router as the various ports trigger the same block-level activity, and this mode already assumes equal bit-level probability for all blocks.

LUNA outputs link utilization information for all the channels of the network which translates to the traffic load of the upstream output port and downstream input port in terms of number of flits per cycle. By examining the traffic load of all input and output ports in a router, the average router load is calculated and fed into Orion to calculate the power consumption of the router.

Background on Orion's power models. To model dynamic power, Orion needs to estimate both switching capacitance and switching activity. Orion derives geometry estimation, *e.g.*, input line length of the crossbar, memory cell width of the FIFO buffer, etc., from both the circuit implementation type and the technology, and relies on a library of base technology parameters, *e.g.*, capacitance per unit length, minimal wire spacing, etc., and technology scaling factors to calculate the switching capacitance. Most technology parameters and scaling factors are taken from CACTI [27], ITRS [25] and from the work of Ho et al. [11]. The interested reader is urged to consult the Orion release site [17] for actual technology parameter values used in this paper. Orion models subthreshold leakage as the only type of static power. Leakage current values are estimated through SPICE simulations for various gate types at every technology, and scaled by the actual gate transistor size, either by a technology parameter or derived during geometry estimation [5].

Router pipeline delay. We incorporated the router pipeline delay model of Peh and Dally [19] into LUNA (see Section III-B) and Orion to derive the effects of router pipeline delay on overall network latency and power respectively. This model considers the number of physical ports p (links) and virtual channels v per link in a router to determine the number of router pipeline stages. To model router clock frequency trends, we adopt the concept that clock speeds cannot exceed a value which corresponds to a fixed number of gate delays, typically defined as the delay of an inverter loaded by 4 identical inverters, or fan-out-of-4 (FO4) inverter delay. Current state-of-the-art circuits at $90nm$ run at an aggressive $3GHz$ with a typical 14 FO4s between pipeline stages at $1.2V$. ITRS projects that it will be possible to clock future $50nm$ circuits at $10GHz$ with 10 FO4s between pipeline stages, again at $1.2V$; ITRS projects that it will not be possible to reduce this number as power issues will become extremely critical due to the frequency increase. Using these projections and the F04-based router pipeline model of Peh and Dally [19], we calculate the number of pipe stages for each router shown in Table I.

Topology floorplan, wire lengths and delays. As Figure 4 shows[6], we project the floorplan of the various topologies, based on prior literature, in order to account for wire delays as

[6]The indirect binary fat tree topology floorplan is based on Leiserson's model [14].

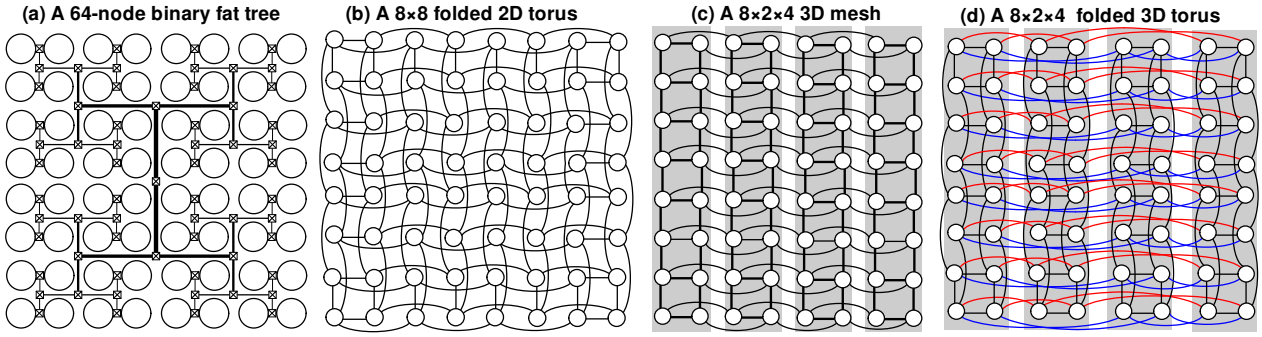

Fig. 4. Actual topology floorplans of 64-node (a) indirect binary fat tree (b) 2D folded torus (c) 3D mesh and (d) 3D folded torus.

technologies scale. Estimated wire delays are shown in Table I. In more complicated network layouts such as 3-dimensional and hierarchical topologies, we account for folding in tori, an H-layout for fat trees as well as the availability of upper metal layers for express links that can be utilized to enable feasibility of such topologies. We assume a square geometry so a network is always arranged as a $N \times N$ grid. Single hop distance is calculated as $\frac{\sqrt{chip\ area}}{N}$. Chip area of $310mm^2$ is assumed, which is the ITRS predicted chip size at production for high performance systems through 2018. Using ITRS interconnect RC delay projections, we calculate the link delays in cycles for 64-node networks as Table I shows.

Geometry and area estimation. Orion's power modeling is based on the determination of the geometry of building blocks in a router. This can thus be readily extended for estimating area. For instance, the area of a buffer is determined by the product of its width and its height, approximated by the length of the wordline and the bitline respectively (approximated, as the input buffer needs no decoder, and the areas of the pre-charging circuitry and an optional output driver are relatively negligible). The bitline and wordline lengths are measured based on the memory cell size and the number of memory cells (rows, columns) which depends on technology parameters inherited from Wattch [3], and the number of cells is derived from the architecture parameters of the buffer, which are the flit occupancy and flit size in bits. The interested reader is urged to consult the Wattch release web site [35] for actual technology parameter values used in this paper. Similarly, the area of the crossbar is calculated based on the lengths of the input and output lines.

Wire capacitance and scaling. Wire capacitance scaling factors, which affect power consumption, are obtained from the work of Ho et al. [11]; this work also provides scaling factors for metal wires with different spacing, which takes into account coupling capacitance. Orion follows the assumption of 5λ as the minimal pitch size ($\lambda = \frac{1}{2}$ gate length) as described by Ho et al. [11] and in the CACTI model [27]. The actual wire spacing is decided by additional factors, e.g. the wire spacing of channels is determined by the node size as $\frac{node_{edge\ size}}{2 \cdot flit_{size\ in\ bits}}$ (2 channels) then by comparing the actual wire spacing with the pitch size; Orion calculates the actual wire capacitance per unit length. For transistor capacitance scaling factors we use the "Ideal NMOS device gate capacitance" from the ITRS-2004 table of projections and the ITRS-2002 "high-performance NMOS device τ" given as $\tau = C_{gate} \times \frac{Vdd}{Idd_{NMOS}}$ that can be used to derive C_{gate}. Wire spacing of channels is dependent upon the technology parameters and affects link capacitance due to the effect of coupling capacitance.

IV. Polaris Validation

To evaluate the correctness of Polaris's high-level modeling we validate it against a detailed NoC synthesis toolchain. Analytically, we compare the entire 3-phase toolchains of Polaris vs. NoC synthesis as shown in Figure 2 for a number of CMP traffic traces and architectures. These architectures espouse various NoC topologies with a wide range of buffer sizes and virtual channels per link. For the NoC synthesis, detailed Verilog models for NoC routers were built for 5 and 9-ported pipelined routers, where the number of pipeline stages for each router was determined using the model of Peh and Dally [19]. We used ModelSim SE 5.6 to verify the correctness of the Verilog models and Synopsys Design Vision to synthesize the two routers with TSMC $90nm$ standard cell libraries. Using this synthesis tool we obtained detailed delay and power numbers for routers. Power and router pipeline delays along with TRIPS [24] traces are fed into PoPNet [23], a cycle-accurate network simulator. Detailed flit activity is tracked, and combined with Synopsys Design Vision's power estimates to derive the final NoC power consumption and average network delay.

For Polaris's roadmapping, we use Trident to extract the 3-tuple spatio-temporal statistics of the same TRIPS CMP traces and then regenerate artificial traces based on their statistical 3-tuples (see Section III-A), while maintaining similar throughput rates as those of the TRIPS traces. These traces are then fed into Polaris's toolchain, where power and delay estimates are obtained as described in Sections III-B and III-C. This procedure enables us to compare the two toolchains, synthesis versus roadmapping (Polaris), across all three phases. We note that due to the lack of standard cell libraries in both the NoC synthesis toolchain and Polaris, we modeled buffers with basic flip-flops and the crossbar structure as a mux-tree instead of incorporating SRAM cells and the matrix crossbar that Polaris originally uses. Also, as we lack back-end libraries, the crossbar was modeled with zero wire load.

It was found that, while Polaris's toolchain was two orders of magnitude faster than the synthesis engine, the two toolchains ranked the various 2D mesh and torus topologies and their hierarchical and express cube variants in the same order in terms of power and delay. This validates the correctness of the relative ranking of results in Polaris's roadmapping toolchain. In absolute terms, Polaris's estimated average network latency falls within 2.5 cycles as compared to the cycle-accurate PoPNet. Polaris's network power estimates have a maximum error of 4.2% and 2.5% on average. Polaris's validation at $90nm$ leads to high confidence with its toolchain flow. Polaris will be continuously validated once new industry libraries at future process technologies become available. Overall, since it is based on ITRS projections and it is highly extensible, as these projections change, Polaris will incorporate them. In short, we believe that as Polaris demonstrates high level of correctness at current technologies, future projections using the same toolchain will also yield good accuracy.

V. Experimental Results

Our experimental results consider 64-node topologies to fairly compare all the topologies currently implemented in Polaris, i.e. a reasonable common factor in terms of node count for all network types considered in this study is 64. Figure 3

TABLE II

ROADMAP SHOWING MINIMUM METRICS OR MOST SUITABLE TOPOLOGIES FOR 64-NODE NETWORKS AT $90nm$ AND $50nm$ TECHNOLOGIES. THE 4-TUPLE UNDER EACH TOPOLOGY DENOTES THE ROUTING TYPE WHERE A = ADAPTIVE AND D = DETERMINISTIC, THE FLOW CONTROL TYPE WHERE W = WORMHOLE AND V = VIRTUAL CUT-THOUGH , THE BUFFER SIZE PER VIRTUAL CHANNEL (VC) IN TERMS OF 64-BIT FLITS AND THE VC COUNT PER PHYSICAL CHANNEL. EC DENOTES EXPRESS CUBE TOPOLOGIES, WHERE "e" DENOTES THE LINK INTERVAL.

Application Characteristics			Latency (Cycles)		Network Power (W)		Energy/Flit (10^{-11}J)		Energy-Delay/Flit (10^{-19}Js)		Power/Area (10^{-8}Wμm^{-2})	
Burstiness	Injection	Hop distance	90nm	50nm	90nm	50nm	90nm	50nm	90nm	50nm	90nm	50nm
Moderate	Hot spot	Short	3D Torus 28.29 A,V,64,4	3D Torus 34.94 A,V,64,2	Ring EC, e=4 1.96 D,W,4,1	Ring EC, e=4 3.31 D,W,4,1	Ring EC, e=4 4.23 D,W,4,1	Ring EC, e=4 1.94 D,W,4,1	3D Torus 6.65 A,V,4,2	3D Torus 1.48 A,V,4,2	3D Torus 2.97 A,V,64,8	3D Torus 21.57 A,V,64,8
Moderate	Evened- out	Short	3D Torus 29.53 A,V,64,4	3D Torus 34.25 A,V,64,2	Ring EC, e=4 1.91 A,W,4,1	Ring EC, e=4 3.28 D,W,4,1	Ring EC, e=4 4.12 D,W,4,1	Ring EC, e=4 1.90 D,W,4,1	3D Torus 6.84 A,V,4,2	3D Torus 1.46 A,V,4,2	3D Torus 2.97 A,V,64,8	3D Torus 21.41 A,V,64,8
High	Hot spot	Short	3D Torus 30.56 A,V,64,4	3D Torus 36.87 A,V,64,2	Ring EC, e=4 1.79 D,W,4,1	Ring EC, e=4 3.08 D,W,4,1	Ring EC, e=4 3.71 D,W,4,1	Ring EC, e=4 1.91 D,W,4,1	3D Torus 6.84 A,V,4,2	3D Torus 1.51 A,V,4,2	3D Torus 2.96 A,V,64,8	3D Torus 21.60 A,V,64,8
High	Evened- out	Short	3D Torus 29.58 A,V,64,4	3D Torus 35.87 A,V,64,2	Ring EC, e=4 1.82 D,W,4,1	Ring EC, e=4 3.12 D,W,4,1	Ring EC, e=4 3.77 D,W,4,1	Ring EC, e=4 1.94 D,W,4,1	3D Torus 6.63 A,V,4,2	3D Torus 1.47 A,V,4,2	3D Torus 2.96 A,V,64,8	3D Torus 21.60 A,V,64,8
Moderate	Hot spot	Long	3D Torus 29.85 A,V,64,4	3D Torus 36.46 A,V,64,2	Ring EC, e=4 2.14 D,W,4,1	Ring EC, e=4 3.58 D,W,4,1	Ring EC, e=4 4.62 D,W,4,1	Ring EC, e=4 2.32 D,W,4,1	3D Torus 7.05 A,V,4,2	3D Torus 1.57 A,V,4,2	3D Torus 3.05 A,V,64,8	3D Torus 22.02 A,V,64,8
Moderate	Evened- out	Long	3D Torus 30.09 A,V,64,4	3D Torus 36.78 A,V,64,2	Ring EC, e=4 2.12 D,W,4,1	Ring EC, e=4 3.54 D,W,4,1	Ring EC, e=4 4.57 D,W,4,1	Ring EC, e=4 2.29 D,W,4,1	3D Torus 7.14 A,V,4,2	3D Torus 1.59 A,V,4,2	3D Torus 3.06 A,V,64,8	3D Torus 22.07 A,V,64,8
High	Hot spot	Long	3D Torus 31.52 A,V,64,4	3D Torus 38.34 A,V,64,2	Ring EC, e=4 2.22 D,W,4,1	Ring EC, e=4 3.69 D,W,4,1	Ring EC, e=4 4.61 D,W,4,1	Ring EC, e=4 2.29 D,W,4,1	3D Torus 7.22 A,V,4,2	3D Torus 1.59 A,V,4,2	3D Torus 3.07 A,V,64,8	3D Torus 22.11 D,V,64,8
High	Evened- out	Long	3D Torus 31.52 A,V,64,4	3D Torus 38.35 A,V,64,2	Ring EC, e=4 2.19 A,W,4,1	Ring EC, e=4 3.64 D,W,4,1	Ring EC, e=4 4.53 D,W,4,1	Ring EC, e=4 2.26 D,W,4,1	3D Torus 7.25 A,V,4,2	3D Torus 1.60 A,V,4,2	3D Torus 3.06 A,V,64,8	3D Torus 22.07 A,V,64,8

shows all topologies considered. We assume packets are each composed of five 64-bit flits. It is noted that the number of nodes and flit sizes are both configurable inputs in Polaris.

According to traffic classification model described in Section III-A we categorize network traffic into 8 classes of spatio-temporal parameters; these classes are shown in the 3 leftmost columns of Table II. The network traffic throughput requirements of each trace is set at a point where a link's bandwidth is utilized at a maximum of 60% for all network topologies tested here. The different networks exhibit different geometries due to their topologies; for fairness, we mapped the hot-spots in such as way so as the hop distance between them was kept roughly equivalent in all tested topologies.

Table II shows the best outcomes for each traffic category out of all the $7,872$ NoC configurations tested in Polaris for this paper. Five categories of results are displayed for both $90nm$ and $50nm$ technologies at the ITRS projected frequencies of $3GHz$ and $10GHz$ respectively, both at $1.2V$. These results are an indicative fragment of Polaris's complete set of roadmap tables that encompass all 12 topologies currently considered; the interested reader is urged to consult the Polaris release web site [22] where 7,872 designs metrics are presented in a similar and much longer table of results.

A number of insights can be drawn from Table II:

Topologies and micro-architectures. The best latency and energy-delay product per flit (EDPPF) performing topology for both $90nm$ and $50nm$ technologies are 3D tori. This can be explained from the fact that 3D tori topologies have a higher link density and a smaller mean hop distance than all other topologies considered here. Though 2D hierarchical topologies contain 8 links per router versus 6 for 3D tori, the greater number of links (ports) resulted in longer pipeline stages (see Section III-C and Table I) which increase their latency as packets have to traverse longer router pipelines. For minimum latency, Polaris indicates that 3D tori will perform best when 4 virtual channels (VCs) per link are used under $90nm$. This number falls to 2 for future $50nm$ topologies. This conclusion can be explained as follows. As the number of virtual channels per physical link and the frequencies increase as lithographic technologies enable further miniaturization at $50nm$, so are the pipeline stages in a router; so at $50nm$ the increase in traffic congestion caused by a fewer number of VCs per link is counterbalanced from the use of smaller pipelines, i.e. designers are better off in using a smaller number of VCs to make shorter pipelines at $50nm$. For EDPPF, Polaris further points out that 3D tori with just 2 VCs per link for both $90nm$ and $50nm$ technologies will outperform 3D tori with a greater number of VCs per link. Again, additional VCs in a router introduce extra pipeline stages with extra buffers needed, giving rise to more energy consumption and zero-load pipeline delays. For network power and energy per flit (EPF) express cube rings with express interval $e = 4$ dominate in both process technologies tested, due to simpler 2 or 4-ported routers. Under the metric of power per area (PPA, the NoC's power consumption divided by the NoC's CMOS area), an indicator of power consumption density and an important metric useful for thermal modeling [26], 3D tori outperform all other topologies across all application categories. This is due to the fact that with higher link densities in 3D tori coupled with a greater number of buffers and VCs per link (64 and 8 respectively), power is amortized over a relatively greater CMOS area to produce a smaller PPA.

Effect of process technologies. Table II shows that though latency will increase in terms of cycle counts when we move to $50nm$ technologies as wires will take multiple cycles to transfer data, latency in ns (absolute time) will improve at $50nm$ due to faster clocking from $3GHz$ to $10GHz$. This gives rise to an increase in the power consumption and a one order of magnitude increase in PPA. On the contrary, with faster clocking at $50nm$, EDPPF will decrease as flits will have shorter in-transit network durations.

Effect of application classes. Table II shows that the various topologies tested in this study will not perform equally well across all 8 application classes as they affect all measured metrics presented at varying levels. The worst performing traffic categories, with a marginal difference between them, under all metrics except for the metric of EPF, are highly bursty, long distance traffic. Longer distance traffic causes higher power consumption and produces more delay and EDPPF than the shorter distance traffic counterparts. For instance, long distance, highly bursty traffic with hot spots produces a higher latency, network power, EDPPF, and PPA, when compared to the results of short distance, highly bursty, hot sport traffic. As NoC architects need to design networks from the application's perspective, the results here indicate the importance of traffic categorization and future NoC roadmapping, projection and exploration.

Further insights. For further roadmapping details we refer to Table III. It shows the best output metrics for each topology tested in this study at $50nm$ for short hop distance, moderately bursty, evened-out injected traffic class of applications. The various NoC topologies are configured at 16-flit buffers with

TABLE III
ROADMAP DETAILING ESTIMATES OF THE BEST MICROARCHITECTURE FOR VARIOUS TOPOLOGIES AT $50nm$ FOR SHORT HOP DISTANCE, MODERATELY BURSTY EVENED-OUT INJECTED TRAFFIC CLASS OF APPLICATIONS. THE IDEAL ARCHITECTURE AT 16-FLIT BUFFERS PER VC AND 4 VCS PER LINK FOR EACH METRIC IS HIGHLIGHTED.

Topology	Latency (Cycles)	Network Power (W)	Area (μm^2)	Energy/Flit (10^{-11}J)	Energy-Delay /Flit (10^{-19}Js)
Binary Fat Tree (Indirect)	114.29	68.58	2.60E+07	44.39	50.74
Plain Ring	151.12	5.45	**4.80E+05**	3.53	5.32
Ring Hierarchical *e*=2	62.63	5.72	8.50E+05	3.70	2.32
Ring Express Cube *e*=4	95.92	**4.98**	6.60E+05	**3.22**	3.09
3D Torus	**38.10**	7.58	1.00E+07	4.90	**1.87**
3D Mesh	44.10	8.41	1.00E+07	5.45	2.40
2D Torus Plain	69.06	7.39	6.80E+06	4.78	3.30
2D Torus Hierarchical, *e*=2	53.04	12.90	1.40E+07	8.35	4.43
2D Torus Express Cube, *e*=4	72.42	7.73	7.30E+06	5.01	3.63
2D Mesh Plain	73.34	8.05	6.80E+06	5.21	3.82
2D Mesh Hierarchical, *e*=2	54.95	13.82	1.40E+07	8.95	4.92
2D Mesh Express Cube, *e*=3	69.52	8.63	7.80E+06	5.59	3.88

4 VCs per link. This class of applications models the behavior of streaming applications such as MPEG video [31] which possess similar statistical spatio-temporal characteristics. Though 3D tori outperform all other topologies in terms of latency and EDPPF, express cube rings with interval $e = 4$ consume 52% less power and 52% less EPF than 3D tori. Hierarchical rings with $e = 2$ also consume 32.5% less power than 3D tori. Using Polaris, a designer may opt to use hierarchical or express cube rings instead of 3D tori if the system the designer intends to build is not highly latency-sensitive; these ring variants offer better energy tradeoffs and 2 orders of magnitude smaller CMOS area, important design consideration factors that can be beneficial for portable devices.

Another interesting insight from Table III is that 64-node indirect binary fat trees (IBFT) may be unsuitable for NoCs. IBFT consume an order of magnitude more power, consume the most area and perform second worst in terms of delay. These are due to the many VCs and the high physical port width which increases by a factor of 2 at every tree level. The high latency can be explained from the fact that congestion is created at the lower levels of the tree due to the presence of links with smaller bandwidths and the longer pipeline delays in routers at the higher levels of the tree (the number of router pipeline stages are related to (p,v) though routers at higher levels can be segmented into smaller parts to enable design flexibility). In fact IBFT were found to perform similarly worse as compared to all other tested topologies under all 8 traffic categories. As a result, Polaris projects that IBFT may be unsuitable for current and future NoCs.

VI. CONCLUSIONS AND FUTURE WORK

With this paper we introduce Polaris, a system-level living roadmap (SLLR) for NoCs that acts as a first step towards a complete SLLR for multi-core on-chip systems. Polaris's 3-phase toolchain rapidly scans over thousands of NoC configuration-application points at various process technologies helping designers explore the design space of NoCs and identify the most suitable architecture(s) that best balance(s) NoC cost-benefits, before investing in actual detailed NoC designs. This paper provides insights on current $90nm$ and future $50nm$ technologies. As Polaris is flexible and expandable, this first version can be readily extended to accommodate more NoC designs and metrics. Possible future directions include the extension of Polaris to roadmap heterogeneous systems, and the addition of system-level reliability and variability predicting models. Future improvements in accuracy can be accomplished with the inclusion of traffic models that capture reactive or dynamic effects caused by data dependencies among the various routing messages and the addition of intuitive traffic categorization models that can help designers choose traffic that better emulates the spatio-temporal characteristics of future applications. In summary, we see Polaris effectively filling the absence of a NoC SLLR, serving as a guide for system-level predictions of future on-chip networks.

ACKNOWLEDGMENTS

We thank Sharad Malik of Princeton, Andrew Kahng of UC San Diego, and Dennis Sylvester of the University of Michigan for early discussions on the motivation and goals of a system-level roadmap for NoCs which sparked off Polaris. This work was supported in part by the MARCO Gigascale Systems Research Center as well as the Intel Corporation.

REFERENCES

[1] K. Agarwal et al. *Variational delay metrics for interconnect timing analysis*. In Proc. of the Design Automation Conference, pp. 381–384, June 2004.
[2] Berkeley Predictive Technology Model (BPTM), 2006. *Device Group, Univ. California at Berkeley. http://www-device. eecs.berkeley.edu/∼ptm*
[3] D. Brooks et al. *Wattch: A framework for architectural-level power analysis and optimizations*. In Proc. of the International Symposium on Computer Architecture, pp. 83–94, June 2000.
[4] L. Benini and G. D. Micheli. *Powering networks on chip*. In Proc. of the International Symposium on System Synthesis, pp. 33–38, Oct. 2001.
[5] X. Chen and L.-S. Peh. *Leakage power modeling and optimization in interconnection networks*. In Proc. of the International Symposium on Low Power Electronics and Design, pp. 90–95 , Aug. 2003.
[6] W. J. Dally and B. Towles. *Route packets, not wires: On-chip interconnection networks*. In Proc. of the Design Automation Conference, pp. 684–689, June 2001.
[7] N. Eisley and L.-S. Peh. *High-level power analysis for on-chip networks*. In Proc. of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems, pp. 104–115, Sept. 2004. *LUNA: http://www.princeton.edu/∼eisley/LUNA.html*
[8] Gigascale Systems Research Center (GSRC), 2006. System-Level Living Roadmap Thrust. *http://www.gigascale.org/roadmap/*.
[9] G. Göossler and A. Sangiovanni-Vincentelli. *Compositional modeling in Metropolis*. In Proc. of the International Conference on Embedded Software, pp. 93–107, Oct. 2002.
[10] G. Horn et al. *A criterion for cost optimal construction of irregular networks*. In Proc. of the International Parallel and Distributed Processing Symposium, pp. 197a–206a, April 2003.
[11] R. Ho et al. *The future of wires*. In Proc. of the IEEE, pp. 490–504, April 2001.
[12] A. Jalabert et al. *×pipesCompiler: A tool for instantiating application specific networks on chip*. In Proc. of the Design, Automation and Test in Europe Conference and Exhibition, Vol. II, pp. 884–889, Feb. 2004.
[13] A. B. Kahng et al. *Non-tree routing for reliability and yield improvement*. In Proc. of the International Conference on Computer-Aided Design, pp. 260–266, Nov. 2002.
[14] C. E. Leiserson. *Fat-trees: Universal networks for hardware-efficient supercomputing*. IEEE Transactions on Computers, Vol. 34, No. 10, pp. 892-901, 1985.
[15] S. Murali and G. D. Micheli. *SUNMAP: A tool for automatic topology selection and generation for NoCs*. In Proc. of the Design Automation Conference, pp. 914–919, June 2004.
[16] U. Y. Ogras and R. Marculescu. *Energy and performance-driven NoC communication architecture synthesis using a decomposition approach*. In Proc. of the Design Automation and Test in Europe Conference, Vol. I, pp. 352–357, March 2005.
[17] Orion release web site, 2006. *http://www.princeton.edu/ ∼peh/orion.html*
[18] K. Park and W. Willinger. *Self-similar network traffic and performance evaluation*. ISBN: 0471319740. John Wiley and Sons, New York, NY, 2000.
[19] L.-S. Peh and W. J. Dally. *A delay model and speculative architecture for pipelined routers*. In Proc. of the International Symposium on High-Performance Computer Architecture, pp. 255–266, Jan. 2001.
[20] S. G. Pestana et al. *Cost-performance trade-offs in networks on chip: A simulation-based approach*. In Proc. of the Design, Automation and Test Conference in Europe, Vol. II, pp. 764–769, Feb. 2004.
[21] D. Pham et al. *The design and implementation of a first-generation Cell processor*. In Proc. of the International Conference on Solid State Circuits Conference, pp. 184–185, March 2005.
[22] Polaris V. 1.0 complete NoC roadmapping tables, 2006. *http:// www.princeton.edu/∼peh/polaris/*
[23] PoPNet release web site, 2006. *http://www.princeton.edu/∼lshang/popnet.html*
[24] K. Sankaralingam et al. *Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture*. In Proc. of the International Symposium on Computer Architecture, pp. 422–433, June 2003.
[25] Semiconductor Industry Association, 2006. International Technology Roadmap for Semiconductors. *http://public.itrs.net/Files/2001ITRS/Home.htm*
[26] L. Shang et al. *Thermal modeling, characterization and management of on-chip networks*. In Proc. of the International Symposium on Microarchitecture, pp. 67–78, Dec. 2004.
[27] P. Shivakumar and N. P. Jouppi. *CACTI 3.0: An integrated cache timing, power and area model*. Western Research Lab (WRL) Research Report 2001/2.
[28] V. Soteriou et al. *A statistical traffic model for on-chip interconnection networks*. In Proc. of the 2006 International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, to appear.
[29] M. B. Taylor et al. *Evaluation of the Raw microprocessor: An exposed-wire-delay architecture for ILP and streams*. In Proc. of the International Symposium on Computer Architecture, pp. 2–13, June 2004.
[30] M. Vachharajani et al. *Microarchitectural exploration with Liberty*. In Proc. of the International Symposium on Microarchitecture, pp. 271–282, Nov. 2002.
[31] G. V. Varatkar and R. Marculescu. *On-chip traffic modeling and synthesis for MPEG-2 video applications*. In IEEE Transactions on Very Large Scale Integration Systems, Vol. 12, No. 1, 2004.
[32] H.-S. Wang et al. *Orion: A power-performance simulator for interconnection networks*. In Proc. of the International Symposium on Microarchitecture, pp. 294–305, Nov. 2002.
[33] H.-S. Wang et al. *Power-driven design of router microarchitectures in on-chip networks*. In Proc. of the International Symposium on Microarchitecture, pp. 105–116, Nov. 2003.
[34] H.-S. Wang et al. *A technology-aware and energy-oriented topology for on-chip networks*. In Proc. of the Design, Automation and Test in Europe Conference, Vol. I, pp. 238–243, March 2005.
[35] Wattch release site, 2006. *http://www.eecs.harvard.edu/ ∼dbrooks/wattch-form.html*.