CryoWire: Wire-Driven Microarchitecture Designs for Cryogenic Computing

Dongmoon Min*
Electrical and Computer Engineering
Seoul National University
Seoul, Republic of Korea
dongmoon.min@snu.ac.kr

Yujin Chung*
Electrical and Computer Engineering
Seoul National University
Seoul, Republic of Korea
yujin.chung@snu.ac.kr

Ilkwon Byun
Electrical and Computer Engineering
Seoul National University
Seoul, Republic of Korea
ik.byun@snu.ac.kr

Junpyo Kim
Electrical and Computer Engineering
Seoul National University
Seoul, Republic of Korea
junpyo.kim@snu.ac.kr

Jangwoo Kim†
Electrical and Computer Engineering
Seoul National University
Seoul, Republic of Korea
jangwoo@snu.ac.kr

ABSTRACT

Cryogenic computing, which runs a computer device at an extremely low temperature, is promising thanks to its significant reduction of wire resistance as well as leakage current. Recent studies on cryogenic computing have focused on various architectural units including the main memory, cache, and CPU core running at 77K. However, little research has been conducted to fully exploit the fast cryogenic wires, even though the slow wires are becoming more serious performance bottleneck in modern processors. In this paper, we propose a CPU microarchitecture which extensively exploits the fast wires at 77K. For this goal, we first introduce our validated cryogenic-performance models for the CPU pipeline and network on chip (NoC), whose performance can be significantly limited by the slow wires. Next, based on the analysis with the models, we architect *CryoSP* and *CryoBus* as our pipeline and NoC designs to fully exploit the fast wires. Our evaluation shows that our cryogenic computer equipped with both microarchitectures achieves 3.82 times higher system-level performance compared to the conventional computer system thanks to the 96% higher clock frequency of CryoSP and five times lower NoC latency of CryoBus.

CCS CONCEPTS

• **Computer systems organization** → **Superscalar architectures**; **Pipeline computing**; **Multicore architectures**.

KEYWORDS

Cryogenic Computing, Pipelining, Network on Chip, Multicore Architectures, Superscalar Architectures, Chip Multi Processor

*Both authors contributed equally to this research.
†Corresponding author.

ASPLOS '22, February 28 – March 4, 2022, Lausanne, Switzerland

ACM ISBN 978-1-4503-9205-1/22/02…$15.00
https://doi.org/10.1145/3503222.3507749

ACM Reference Format:
Dongmoon Min, Yujin Chung, Ilkwon Byun, Junpyo Kim, and Jangwoo Kim. 2022. CryoWire: Wire-Driven Microarchitecture Designs for Cryogenic Computing. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '22), February 28 – March 4, 2022, Lausanne, Switzerland*. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3503222.3507749

1 INTRODUCTION

In modern high-performance server systems, slow wires have become severe performance bottlenecks. For example, it is now extremely challenging to increase the core's clock frequency and thus the single-thread performance as the wires connecting the microarchitectural units (i.e., intra-core wires) dominate the critical-path delay of the CPU pipeline. In addition, as the number of cores in a CPU continuously increases, its multi-thread performance can be limited by a network on chip (NoC) where the slow inter-core wires can significantly degrade the performance. As the technology scaling continues, the wire performance bottlenecks are getting more serious due to the increasing wire resistance.

Cryogenic computing, which is to run a computer device at extremely low temperatures (e.g., 77K), is a highly promising solution to resolve the wire performance bottlenecks thanks to the reduction of wire resistance. Our analysis confirms the significant wire speed-up (over three times) at 77K, compared to 300K. In addition, with the exponentially reduced leakage current, cryogenic computing can benefit from the fast wires while improving power efficiency. Therefore, in this work, we aim to apply cryogenic computing to CPU cores and NoCs to resolve the wire-performance bottlenecks.

However, there exist challenges to exploit the fast cryogenic wire. Cryogenic cooling alone cannot fully exploit the potential of fast wires. For instance, in the recent study [16], the performance gain of cryogenic processors was marginal (15%) compared to the wire speed-up at 77K (>300%). That is, architects should conduct an in-depth analysis on the CPU cores and NoCs at 77K, and explore a novel microarchitecture design to fully exploit the fast wires.

In this paper, we propose our CPU pipeline and NoC microarchitectures to exploit the fast cryogenic wires. To achieve the goal, we first introduce our cryogenic pipeline and NoC modeling frameworks. For our models, we adopt and modify the state-of-the-art

cryogenic modeling tool [16] and then validate their accuracy with real-world experiments and circuit-level simulations.

Next, based on the critical-path delay analysis using our pipeline model, we derive a 77K-optimal CPU pipeline (*CryoSP*) by utilizing the fast intra-core wires. First, in the 300K critical-path analysis, we observe that the modern CPU's frequency scaling is limited by the un-pipelinable backend stages for data-forwarding, whose critical-path delay is the highest among the stages. We also observe that the backend stages have a high wire-delay portion in their critical-path delay. Second, in the 77K critical-path analysis, we observe that the frontend stages become the frequency bottleneck while the backend-stage delay significantly decreases. In other words, we observe an opportunity to improve the CPU frequency with the frontend superpipelining at 77K. Finally, based on the observations, we propose CryoSP, our cryogenic superpipelined processor which has 96% higher clock frequency than 300K baseline processor.

Third, we architect a 77K-optimal NoC (*CryoBus*) to fully exploit the fast inter-core wires while providing two design guidelines. As the first guideline, we suggest to utilize the shared bus at 77K instead of the router-based NoCs. Our analysis shows that the router-based NoCs' performance just marginally increases while the shared bus's latency significantly decreases thanks to the huge wire speed-up at 77K. Next, for the second guideline, we emphasize the importance of minimizing the shared bus's latency to overcome the bus contention. Even though the shared bus's performance significantly increases at 77K, our analysis shows that the bus is still too slow to run the representative workloads (e.g., SPEC2006, SPEC2017) in a 64-core CPU system. Based on our design guidelines, we propose CryoBus, our fast cryogenic bus which is scalable to the 64-core CPU system. By applying the H-tree-shaped bus topology and dynamic link connection, CryoBus becomes five times faster than 300K Mesh while providing the comparable bandwidth.

For the performance evaluation, we compare our proposed CPU microarchitectures (i.e., CryoSP and CryoBus) with the state-of-the-art cryogenic core (i.e., CHP-core [16] with 77K Mesh). Compared to CHP-core, CryoSP improves the multi-thread performance by 16.1% thanks to the 28% higher clock frequency. Next, compared to 300K Mesh, CryoBus improves the multi-thread performance by 110%. Finally, our cryogenic system with both CryoSP and CryoBus achieves significant performance gain by 153% and 282% on average compared to the 77K and 300K baseline, respectively.

In summary, our work makes the following contributions:

- **Cryogenic pipeline and NoC performance modeling:** To our knowledge, this is the first work to model, validate, and optimize the pipeline and NoC architecture for 77K.
- **In-depth analysis and observation:** We thoroughly analyze the impact of wire speed-up on the CPU pipeline and the NoC design and provide valuable observations at 77K.
- **Wire-driven microarchitecture design:** Driven by our observations and guidelines, we propose the novel CPU pipeline and NoC microarchitectures (CryoSP and CryoBus) which fully exploit the fast cryogenic wires.
- **Significant performance improvements:** With CryoSP and CryoBus, our cryogenic system significantly improves the system-level performance by 153% and 282% compared to the 77K and 300K baseline systems, respectively.

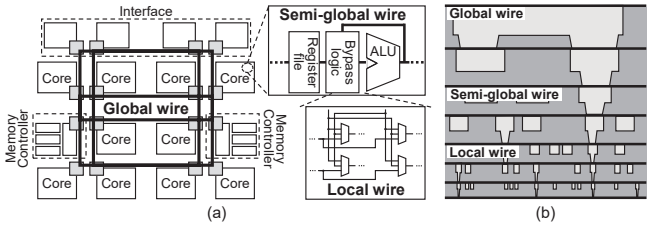

Figure 1: (a) Multi-core CPU architecture and (b) metal layers

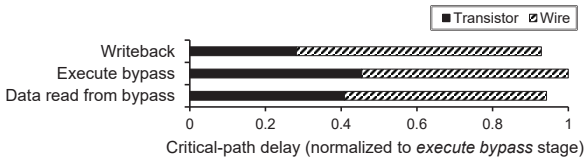

Figure 2: Critical-path delay breakdown of three pipeline stages (derived from our modeling in Section 4)

2 BACKGROUND AND MOTIVATION

2.1 Metal Wires in Server-Class Processors

Fig. 1 shows the overview of a modern server-class CPU architecture (i.e., Intel Cascade Lake). In the recent fabrication process, wires using different metal layers can be classified into three types: global wire, semi-global wire, and local wire. First, the global wire is the thickest wire with the lowest wire resistivity which is usually utilized in the network on chip (NoC). Second, the semi-global wire is the wire at middle-level metal layers which is used to connect microarchitectural units inside a CPU core (Semi-global wire in Fig. 1). Third, the local wire is the thinnest wire with highest resistivity which is used to connect adjacent logic gates and transistors inside a microarchitectural unit (Local wire in Fig. 1). In the rest of our paper, we refer to the global wire as inter-core wire while the semi-global wire and the local wire as intra-core wire.

2.2 Server Performance Bottleneck on Wires

In a modern high-performance server with multi-core CPUs, its single and multi-thread performance can be significantly limited due to the slow intra-core wires and inter-core wires, respectively.

Intra-core wire and single-thread performance. In the past, architects have mainly improved the single-thread performance with the clock frequency scaling. However, the frequency scaling has become difficult as the intra-core wires dominate the critical-path delays of CPU pipeline stages. Fig. 2 shows our model-driven critical-path delay of *writeback, execute bypass*, and *data read from bypass* stages whose delays are the longest among all the stages (Fig. 12). With 57.6% of the wire portion on average, the result clearly shows the intra-core wire limits the CPU frequency scaling.

Inter-core wire and multi-thread performance. Since early 2000s, architects have improved the processor's multi-thread performance by increasing the number of cores on chip (Chip Multiprocessing, CMP) [35, 47, 60, 61, 65]. For example, the number of cores for a single-die Intel Xeon processor exponentially increased from 2002 and reached 28 in 2019 [26]. However, as the on-chip

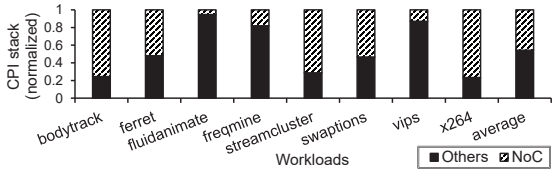

Figure 3: Normalized CPI stacks of PARSEC 2.1 workloads

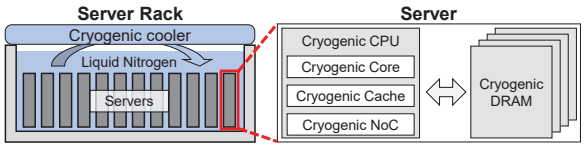

Figure 4: Cryogenic-cooled computer systems where the entire nodes are immersed in LN (liquid nitrogen; 77K)

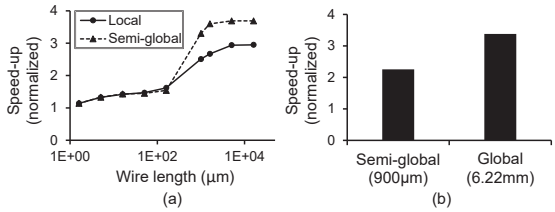

Figure 5: 77K wire speed-up (a) without and (b) with repeaters

communication overhead increases, the multi-thread performance of modern multi-core CPUs can be limited by the inter-core wire speed. Fig. 3 shows the normalized CPI stacks of PARSEC 2.1 workloads [11] obtained by our Gem5 simulations [12] for a 64-core CPU system. The figure clearly shows that the NoC performance significantly affects the modern application performance (45.6% on average, 76.6% in maximum). Note that the slow inter-core wire can greatly degrade the NoC performance by reducing the NoC frequency or increasing the cycles consumed for the packet transfer.

Therefore, architects are now in dire need of fast intra-core and inter-core wires to improve modern server performance.

2.3 Potential of Cryogenic Computing

Cryogenic computing, which runs a computer at extremely low temperatures, is highly promising to resolve the wire performance bottleneck of modern processors. Specifically, both the intra-core and inter-core wires' speed become significantly fast as the wire resistivity linearly decreases with the temperature [42]. In addition, with the exponentially reduced leakage current [56], cryogenic computing enables to lower the operating and threshold voltages (i.e., V_{dd}, V_{th}) of transistors. As a result, architects can exploit the fast wires while improving the power efficiency. In this work, we choose 77K as our target temperature because the 77K computing can cost-effectively utilize conventional CMOS technology (e.g., cheap liquid nitrogen (LN), moderate cooling-power cost) [15, 16, 27, 37, 43].

For example, Fig. 4 shows the 77K-cooled computer systems assumed in previous works [15, 16, 37, 43]. As the cryogenic system fully immerses the entire server rack into liquid nitrogen (LN), it reliably maintains the low temperature with significantly high heat-transfer speed (e.g., 2.64 times higher than 300K cooling in [16]). In addition, the cryogenic system also minimizes the one-time cost (i.e., LN cost) and overall cooling cost, as the cooling system recycles the LN with cryo-coolers (i.e., Stinger cooling systems [27]). Previous studies successfully showed the cost efficiency of the cryogenic computer systems where the 77K-optimized CPU [15, 16], cache [43], and memory devices [37] are integrated. In this work, we also assume the above cryogenic computer system and maximize the server performance with the faster wires.

To accurately estimate the potential of wires at 77K, we perform an in-depth analysis through Hspice simulation. In our analysis, we construct the local, semi-global, and global wire circuits and measure their speed-up at 77K compared to the 300K. For the wire resistivity and capacity, we use the measured values of Intel 45nm fabrication at 300K and 77K [44, 52]. Meanwhile, we use an industry-provided MOSFET model card to simulate the repeaters, which we insert into the semi-global and global wire circuits in a latency-optimizing manner. The industry model card is based on 2z-nm MOSFET samples and already validated with real measurements at the 77K-to-300K temperature range. From our analysis in Fig. 5, we confirm that cryogenic computing can significantly improve the speed of all types of wires. Fig. 5(a) shows that both long local and semi-global wires without repeaters achieve the speed-up of 2.95 and 3.69 in maximum, respectively. In addition, Fig. 5(b) shows that the average-length semi-global and global wire (900μm and 6.22mm [9]) with repeaters achieve the speed-up of 2.25 and 3.38.

2.4 Research Goal and Challenges

In this work, we aim to improve the performance of CPU core and NoC by utilizing the fast wires enabled at 77K. However, there are several challenges to exploit the fast cryogenic wires as follows.

In-depth analysis. Architects should exactly identify the relationship between the wire speed-up and the target hardware performance at 77K. Recent studies on cryogenic computing benefit from the fast wires and improve the hardware units' performance [15, 16, 37, 43]. However, their performance gains are much lower than the cryogenic wire's performance potential (e.g., only 15% higher CPU clock frequency in [16]). Unfortunately, without an in-depth analysis, it is difficult to reason about the gap between the wire's potential and the real performance gain as well as an opportunity for further performance improvement.

77K-optimal microarchitecture. Architects should actively explore the microarchitectural changes required to fully exploit the fast cryogenic wires. A conventional computer system may not be able to benefit from the wire speed-up due to the limitations of the current microarchitecture. For example, if an NoC's microarchitecture is designed to consume only one cycle for the packet transfer (e.g., existing router-based NoCs), its performance improvement can be significantly limited. That is, cryogenic wire's speed-up can be meaningless without microarchitectural innovations accompanied.

Therefore, in this work, we resolve the challenges and propose our CPU pipeline and NoC designs by exploiting the fast intra-core and inter-core wires. For this goal, we first introduce our cryogenic core and NoC performance-modeling methodology (Section 3). Next, with our model, we perform an in-depth analysis and

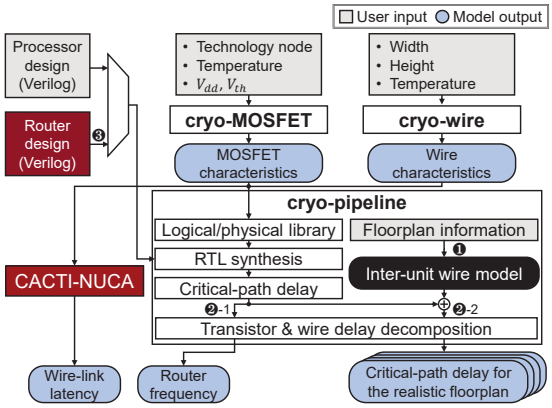

Figure 6: Overview of our cryogenic pipeline and NoC model

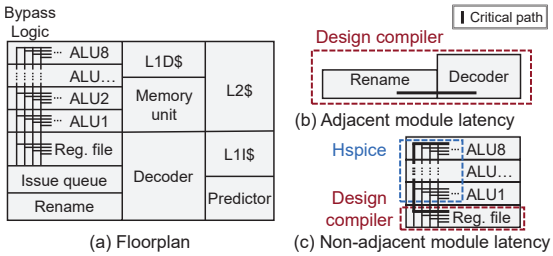

Figure 7: Inter-unit wire model with Intel Skylake floorplan

propose the novel microarchitectures for the pipeline (Section 4) and NoC running at 77K (Section 5).

3 MODELING AND VALIDATION

3.1 Modeling Methodology

Fig. 6 shows the overview of our cryogenic pipeline and NoC modeling methodology based on the CryoCore-Model (CC-Model) [16], the state-of-the-art and validated cryogenic processor modeling tool. In the following sections, we first introduce CC-Model and then show how we modify and adapt the model for our pipeline and NoC performance modeling.

3.1.1 CC-Model Overview. CC-Model consists of MOSFET (cryo-MOSFET), wire (cryo-wire), and processor model (cryo-pipeline). First, cryo-MOSFET is a validated cryogenic MOSFET model which takes a model card as an input, automatically adjusts the model card for the given V_{dd} and V_{th}, and derives the MOSFET characteristics (i.e., I_{on}, I_{leak}) at the target low temperature. Second, cryo-wire takes a metal-layer specification as its input and generates the geometry-aware wire resistivity at cryogenic temperatures. Third, for the given processor design, cryo-pipeline can predict its critical-path delay at low temperatures by taking the logical and physical libraries generated by cryo-MOSFET and cryo-wire, respectively. For the purpose, cryo-pipeline synthesizes a target Verilog design with Design Compiler Topographical Mode [58]. In addition, by using Design-Compiler options, cryo-pipeline can derive the critical-path delay while decomposing its transistor/wire delay portion.

Table 1: Area, width, and height of ALU and register file, and the forwarding-wire length in our processor model

	Area (μm^2)	Width (μm)	Height (μm)
ALU	25757	345	74
Register file	376820	345	1090
Forwarding wire	(8*ALU + Register file)		1686

3.1.2 CC-Model Extension: Inter-Unit Wire Model. As we emphasized in Section 2.4, an in-depth analysis for the critical-path delay is necessary. For such an in-depth analysis, the realistic floorplan of a target processor is crucial because it greatly affects the inter-unit wires' length and latency. However, as the floorplanning of Design Compiler Topographical Mode can be different from the commercial processor's floorplan, CC-Model-based analysis can be different from the real-world results. Therefore, we modify CC-Model to include more realistic inter-unit wire latency by adding the inter-unit wire model which can consider commercial processor's floorplan (marked as black in Fig. 6).

The modified CC-Model derives the stage-wise critical-path delay as follows. First, CC-Model takes the floorplan and processor design as its input (❶). We use Intel Skylake-based floorplan [67] (Fig. 7(a)) because it is the most detailed floorplan among the available designs. For the input processor design, we utilize Intel Skylake-like 8-issue out-of-order processor which follows Intel Skylake's microarchitecture specification with BOOM's pipeline [5].

Next, our model derives the critical-path delay of each pipeline stage in two different ways, depending on the relative locations of included microarchitectural units in the floorplan (❷). For the stage only including a path passing through adjacent units, we derive the critical-path delay with Design Compiler only (❷-1). For example, to derive the critical-path delay of *decode & rename* stage in Fig. 11, we compile the decoder and rename logic together and extract the delay between them by using Design Compiler (Fig. 7(b)).

On the other hand, for the stages with non-adjacent units (i.e., the stages including long-forwarding wires (*wakeup from writeback, writeback, data read from bypass, execute bypass* in Fig. 11)), we derive long inter-unit wire latency with Hspice simulation while separately deriving intra-unit latency with Design Compiler (❷-2). For example, for the critical-path delay of *writeback* stage in Fig. 11, we calculate the ALU-to-register-file wire latency with Hspice and register-file write latency with Design Compiler (Fig. 7(c)).

For Hspice simulation, we assume the semi-global wires for inter-unit wire modeling and follow the similar simulation methodology as that of semi-global wire analysis in Section 2.3. To derive the length of long inter-unit wires for Hspice simulation, we obtain the area of microarchitectural units from Design Compiler, and calculate the distance between units based on the assumed floorplan and the obtained area information. For example, for the length of forwarding wire traversing the ALUs and the register file in Fig. 7(a), we add all ALUs' and register file's height. We summarize area, width, and height of ALU and register file (derived from BOOM using Design Compiler with FreePDK 45nm library) and the length of forwarding wire in Table 1. Note that we follow the previous works [39, 48, 49] for the forwarding wire's floorplan (i.e., ALUs and register files share the same set of forwarding wires together).

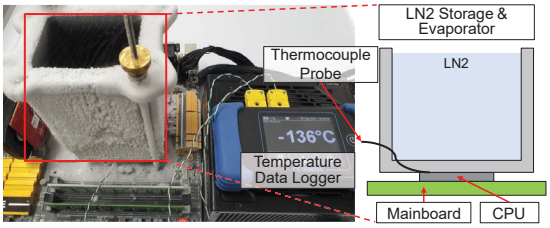

Figure 8: Experimental setup for model validation

Table 2: CPU and mainboard specification for the validation

CPU			Mainboard
Technology	Microarchitecture	Model name	
32nm	Sandy Bridge	i7-2700K	GA-Z77X-UD3H
22nm	Haswell	i7-4790K	GA-Z97X-UD5H
14nm	Skylake	i5-6600K	GA-Z170X-Gaming 7

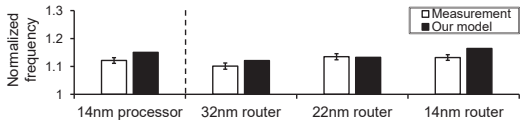

Figure 9: Pipeline and router model validation results

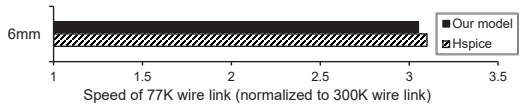

Figure 10: Wire-link model validation results

3.1.3 CC-Model Extension: NoC Model. We model NoC's router and wire-link running at 77K by using CC-Model and the modeling methodology in [37]. In Fig. 6, we marked the modifications for the NoC domain as red boxes. First, based on the given router design, router model reports the maximum clock frequency at low temperatures. For the router model, we utilize CC-Model [16] and take a router's Verilog design as input instead of a processor design (❸). Second, wire-link model generates a target temperature-optimal wire-link design and reports its latency. For the wire-link model, we use CACTI-NUCA [45], which takes the NUCA cache specifications from users (e.g., the number of banks), finds an optimal cache-bank layout, and reports the length and latency of the optimized wire-links as one of its outputs. We extend CACTI-NUCA's coverage to the cryogenic temperatures by adding an interface for accepting MOSFET and wire parameters from cryo-MOSFET and cryo-wire.

3.2 Model Validation

3.2.1 Methodology for Pipeline and Router Models. To validate our pipeline and router models, we individually measure the commercial CPU's core and uncore frequency speed-up at 135K. Fig. 8 shows our experimental setup to measure the core and router frequency at 135K. We construct a sample computer board using various commodity parts (e.g., mainboard, CPU, DIMMs) and the evaporator for LN cooling. Especially, we use the CPUs which not only utilize a router-based NoC but also can separately adjust the frequency of cores (i.e., pipeline, L1/L2 cache) and uncore components (i.e., NoC, L3 cache). With the setup in Fig. 8, we find the maximum frequency of each temperature by increasing the frequency until the booting process fails. We take the uncore frequency speed-up as the router speed-up by assuming that the wire-dominant uncore components (i.e., cache, wire link) become much faster than the routers at 135K.

For the models' input technology library, we use 45nm FreePDK [46], as it is the only option which is open-sourced, Synopsys-compatible, and editable without decryption. We use BOOM [5] and EVA [10] as the input pipeline and router designs.

As the Verilog source file of a commercial processor is unavailable and there is no frequency-unlocked commercial 45nm CPU with router-based NoC, we take an alternative approach to validate our model. We measure 32nm, 22nm, and 14nm Intel CPUs with the ring-bus NoC (Table 2) and project our model's result (45nm) by following the ITRS roadmap [68]. For example, if wire and transistor delay increase (or decrease) in 32nm, 22nm, and 14nm technology node in the roadmap, we scale our model's result with the ratio.

3.2.2 Methodology for Wire-Link Model. We validate our wire-link model by comparing its prediction with the results of Hspice simulations. In the validation, we target the wire-link length of our final network design (i.e., CryoBus), whose length is 6mm from our model (with the setup in Table 4). The Hspice simulation methodology is same as that of the global-wire analysis in Section 2.3.

3.2.3 Validation Result of Pipeline and Router Models. Fig. 9 shows the validation results of our pipeline and router models. We compare our pipeline model's prediction with the measured result of the 14nm processor, as we adopt the floorplan of 14nm Skylake. On the other hand, we verify our router model with the 32nm, 22nm, and 14nm processors, because they utilize the router-based NoCs. The error bars indicate the last succeeded and first failed frequency from the experiments, normalized to the 300K frequency. In Fig. 9, the router model reports a reasonably accurate frequency speed-up at 135K, with 2.8% of maximum error between the model and measurement in the 14nm router. The pipeline model's prediction (15.0%) is also close to the measurement (12.1%).

3.2.4 Validation Result of Wire-Link Model. Fig. 10 shows the validation result of the 6mm 77K wire link, where the 77K latency values are normalized to that of 300K wire link. In our model, the wire link becomes 3.05 times faster at 77K. The speed-up value matches the Hspice simulation results with 1.6% of error rate.

4 WIRE-DRIVEN PIPELINE DESIGN AT 77K

In this section, we derive a 77K-optimized pipeline by utilizing the fast intra-core wire with our modeling. First, we emphasize our observations in critical-path delays at 300K and 77K. Next, based on our observations, we derive *CryoSP*, our superpipelined cryogenic core design operating at a higher clock frequency.

4.1 Baseline Pipeline Microarchitecture

We utilize RISC-V BOOM's pipeline microarchitecture as the baseline, because BOOM is the most sophisticated out-of-order processors among the open-source processors [5]. Fig. 11 shows the

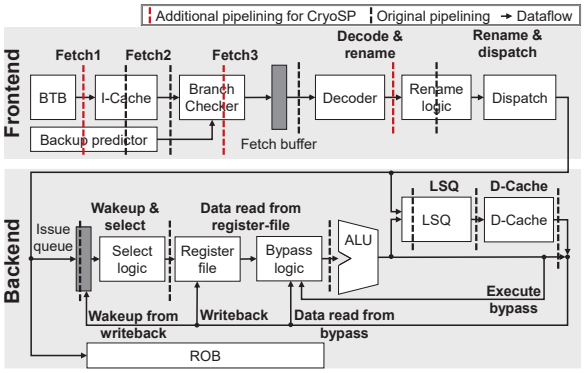

Figure 11: CPU pipeline microarchitecture of BOOM

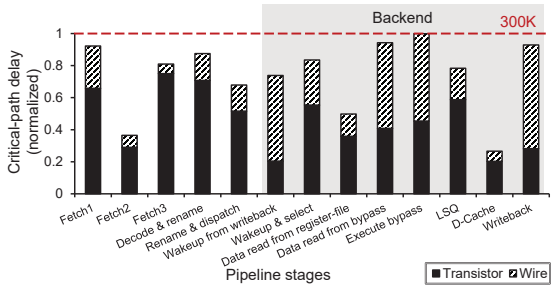

Figure 12: Critical-path delay of the baseline core at 300K

pipeline structure of BOOM. In Fig. 11, 'Original pipelining' indicates the pipeline stages, and bolded texts are the name of stages.

We select 13 representative stages where the upper stages indicate the frontend stages and the lower stages refer to the backend stages. First, in frontend stages, BOOM adopts the overriding-predictor structure [29, 30, 40], which consists of the fast 1-cycle branch predictor (inside BTB) and slow but accurate 2-cycle main predictor (i.e., Backup predictor; GShare, TAGE). If the fast predictor's result is different from the prediction of the accurate backup predictor, the branch checker recognizes the mismatch and rolls back (or overrides) the prediction of the fast predictor. *Fetch1, fetch2, and fetch3* indicate the branch prediction, instruction cache access, and branch checking stage, respectively. Note that the overriding structure is actively used in modern processors to hide the long branch-prediction latency. Next, for backend stages, BOOM adopts the read-after-issue design and has the separate issue queue for integer and floating point operation, respectively. We exclude *commit* stage as BOOM asynchronously conducts the commit [5].

For the processor design, we utilize the BOOM pipeline with Intel Skylake's specification (e.g., issue width, number of microarchitectural units) as summarized in Table 3 (300K Baseline). Note that the issue width of recent BOOM (i.e., 8-issue) is already comparable to Skylake. We utilize the FreePDK 45nm library for evaluation.

4.2 Critical-Path Delay Analysis at 300K

We first analyze critical-path delays of pipeline stages at 300K. Fig. 12 shows the stage-wise critical-path delays of 300K Baseline, which is normalized to the maximum critical-path delay at 300K. At 300K, we find out two major observations.

300K Observation #1. Most backend stages have huge wire-delay portion. In Fig. 12, we observe that the backend-pipeline stages have higher wire portion (45%) than the frontend stages (19%) on average. There are two major reasons for the higher wire portion. First, several backend stages include long data-forwarding wires to support the execution of dependent instructions. For example, *data read from bypass* and *execute bypass* stages have long data-forwarding wires to forward the execution results to data-bypass logic. Second, the backend stages including Contents Addressable Memory (CAM) (e.g., *wakeup & select, LSQ*) show high wire-latency portion because CAM should drive huge fanout [49, 63].

300K Observation #2. Backend stages with long-forwarding wires are the main bottlenecks for frequency scaling. We also observe that the backend stages with the long-forwarding wires are generally the bottleneck stages (as shown in Fig. 12) because most of them are difficult to pipeline due to the huge IPC overhead. If we pipeline the backend stages related to the dependent-instruction execution (i.e., *data read from bypass, execute bypass*), the dependent instructions cannot be executed in a back-to-back manner and thus significantly degrades IPC [13, 48, 49]. Therefore, the backend stages supporting the dependent instructions should operate in a single cycle for high performance. As a result, the un-pipelinable backend stages with the long-forwarding wires (e.g., *data read from bypass, execute bypass*) become the main bottlenecks for frequency scaling in modern processors. Note that for modern wide-issue processors, the slow forwarding-wire challenge is a well-known and general problem which has been raised by several studies [7, 21, 48, 49, 61].

4.3 Critical-Path Delay Analysis at 77K

We now analyze the critical-path delays of Baseline at 77K (Fig. 13). We find out two key observations in the critical-path delays at 77K.

77K Observation #1. Transistor-dominant frontend stages limit the frequency improvement at 77K. With the huge wire-latency reduction at 77K, the critical-path delays of backend data-forwarding stages (i.e., *data read from bypass, execute bypass*) are significantly reduced and thus we expect the high frequency gain. However, we observe that the maximum critical-path delay is only slightly reduced (19% in Fig. 13) due to the transistor-dominant frontend stages (*fetch1, fetch3, decode & rename*). Specifically, the frontend stages cannot fully utilize the advantage of the cryogenic temperature because they do not utilize long wires like data-forwarding stages, and the performance of transistors just marginally increases (8%) compared to that of the wires (2.81 times). As a result, the critical-path stage moves from the backend stages to the frontend stages and limits the frequency gain at 77K.

77K Observation #2. Frontend superpipelining can improve the performance at 77K thanks to the significantly reduced latency of backend-forwarding stages. Different from the 300K case, we observe that the huge wire-latency reduction at 77K makes an opportunity to improve the performance with frontend superpipelining. As we mentioned in 300K Observation #2, the processor's frequency cannot be increased through further pipelining because the un-pipelinable backend stages are the bottlenecks. However, as the backend stages actively utilize long semi-global

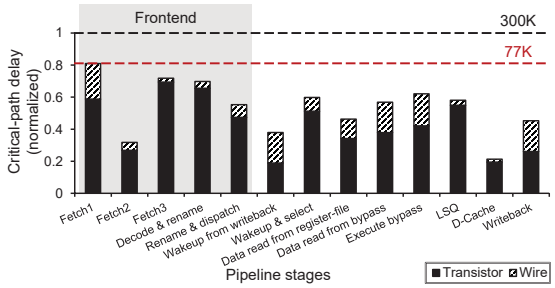

Figure 13: Critical-path delay of baseline core at 77K

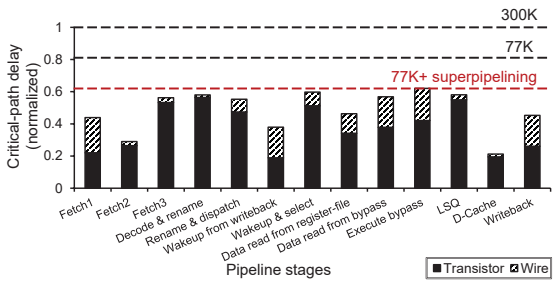

Figure 14: Critical-path delay after superpipelining at 77K

wires for data forwarding, the un-pipelinable stages' delays are greatly reduced at 77K. This is a very important phenomenon because it makes further pipelining meaningful if pipelinable stages are the new critical-path stages at 77K. In other words, with the 77K Observation #1, we can improve the cryogenic processor's performance through the frontend superpipelining.

Note that several frontend stages have potential for further pipelining because most of them are not related to the execution of dependent instructions. Specifically, previous works point out that renaming logic and cache access can be pipelined [48]. In addition, even though a BTB should take its prediction result within one cycle, branch predictors can be pipelined in general [29, 51, 57]. For example, in the case of overriding-predictor, we can pipeline the main predictor with the little IPC overhead as the fast 1-cycle predictor hides the multi-cycle prediction [30, 40].

4.4 Superpipelining Methodology at 77K

In the previous section, we find that further pipelining becomes effective at cryogenic temperatures and thus the frontend superpipelining can increase the core's frequency. Following our observation, we propose the methodology for frontend superpipelining at 77K. To clarify our methodology with an example, we use Baseline microarchitecture analyzed in the previous sections. The detailed methodology is as follows.

First, among the latency of the un-pipelinable backend stages (e.g., *data read from bypass, execute bypass*), we choose the longest latency as the superpipelining's target latency. In our scenario, the latency of *execute bypass* is the target because it is the longest among the backend latencies (as shown in Fig. 13). Next, we find frontend stages whose delays exceed the target latency, and pipeline those stages to make their delays lower than the target latency. As the latency of *fetch1, fetch3, decode & rename* exceed the *execute bypass* latency, we pipeline those stages (highlighted with 'Additional pipelining for CryoSP' in Fig. 11). Finally, by analyzing the performance overhead (i.e., IPC reduction) of the pipelining, we determine whether the frequency gain of superpipelining surpasses the IPC reduction or not. From the Gem5 simulation with PARSEC 2.1 workloads, we find that the three stages added with the superpipelining reduces IPC by only 4.2% (due to higher mis-prediction penalty) which is far less than the benefits of the higher frequency. Note that the IPC value in Table 3 is obtained under the same frequency (i.e., 4GHz) to clarify the impact of pipelining.

On top of Baseline, we implement the frontend superpipelining to *fetch1, fetch3, and decode & rename* stages as follows.

***Fetch1* stage.** We pipeline *fetch1* stage by inserting a flip-flop between the BTB and the instruction cache (I-cache) decoder. In addition, to match the branch-check timing, we also add one more stage to the main predictor (Backup predictor in Fig. 11) by dividing the hashing and decoding steps in GShare predictor.

***Fetch3* stage.** For *fetch3* stage, we divide the branch checker into two pipeline stages which correspond to the branch decoder and the address checker, respectively. The branch decoder checks whether the fetched instruction's type is the branch or not, and the address checker compares the addresses predicted by the fast predictor and the main predictor. For the pipelining, we insert a flip-flop between the branch decoder and the address checker while also pipelining input and output signals to match the timing.

***Decode & Rename* stage.** We pipeline *decode & rename* stage by separating the instruction decoder and rename logic's dependency checker. In our Baseline, the rename logic's dependency checker and mapping table access are divided into two different stages (i.e., *decode & rename, rename & distpatch*) and thus *decode & rename* stage consists of the decoder and the dependency checker. Therefore, we insert a flip-flop between the decoder and dependency checker to reduce the critical-path delay of *decode & rename* stage.

As a result, we superpipeline the five-stage frontend into the following eight stages: 1) BTB + fast branch prediction, 2) I-cache decoding, 3) I-cache access, 4) branch decoding, 5) address checking, 6) instruction decoding, 7) dependency checking, and 8) mapping table access + dispatch. Note that the frontend stages are not exhaustively pipelined in 300K Baseline (e.g., *fetch1, decode & rename*), which well matches our observation on 300K superpipelining (i.e., further frontend pipelining is meaningless at 300K).

Fig. 14 shows the critical-path delay after superpipelining at 77K. For the superpipelined stages, we show the latency of the longer stage for simplicity. Fig. 14 clearly shows that the maximum critical-path delay is reduced by 38.0% compared to 300K Baseline. As a result, our frontend superpipelining at 77K increases the frequency by 61% and 38% compared to 300K and 77K Baseline processors.

4.5 CryoSP: Superpipelined 77K Core Design

In this section, we derive the 77K-optimal superpipelined processor, *CryoSP*. CryoSP adopts the CryoCore specification with V_{dd} and V_{th} scaling, along with our frontend superpipelining [16]. CryoCore is the previously-proposed cryogenic-core microarchitecture, which reduces its issue width and number of microarchitectural units by half to reduce the power consumption. Note that the V_{dd} and V_{th}

Table 3: Pipeline specification of various cores

	300K Baseline	77K Super-pipeline	77K Super-pipeline + CryoCore	77K CryoSP	CHP-core
Frequency	4.0GHz	6.4GHz	6.4GHz	7.84GHz	6.1GHz
Core power	1	1.61	0.3575	0.093	0.093
Total power	1	17.15	3.73	1	1
Pipeline depth	14	17	17	17	14
Pipeline width	8	8	4	4	4
Load queue size	72	72	24	24	24
Store queue size	56	56	24	24	24
Issue queue size	97	97	72	72	72
Reorder buffer size	224	224	96	96	96
# physical integer registers	180	180	100	100	100
# physical float registers	168	168	96	96	96
IPC (@4GHz)	1	0.96	0.9	0.9	0.93
V_{dd}	1.25	1.25	1.25	0.64	0.75
V_{th}	0.47	0.47	0.47	0.25	0.25

scaling is only feasible at cryogenic temperatures, thanks to the nearly eliminated leakage current at 77K (whereas reducing V_{dd} and V_{th} exponentially increases the leakage current at 300K).

Table 3 shows the pipeline specifications for various cores following our optimization steps, which also includes the cryogenic high-performance core (CHP-core) [16]. CHP-core is derived from CryoCore by applying the V_{dd} and V_{th} scaling to maximize its frequency while maintaining the total power consumption lower than 300K Baseline. The total power (also summarized in Table 3) indicates the power consumption including the cryogenic cooling power cost. See Section 6.1 for the detailed cooling power modeling.

We start from the 300K processor which adopts Intel Skylake specification (300K Baseline). First, we apply our superpipelined architecture and increase the clock frequency by 61% at 77K (77K Superpipeline). Next, to reduce the power consumption, we reduce the issue width and size of microarchitectural units following CryoCore design (77K Superpipeline+CryoCore). This optimization reduces the power consumption by 77.8%. Finally, to further improve the frequency and power efficiency, we adopt the V_{dd} and V_{th} scaling with the same method applied to CHP-core. As a result, we design *CryoSP*, whose clock frequency is 95% higher than 300K Baseline while consuming the same power including the cooling cost (i.e., total power). CryoSP also operates at 28% higher clock frequency than the CryoCore-based voltage-optimized design (CHP-core) with the same total power consumption.

5 WIRE-DRIVEN NOC DESIGN AT 77K

In this section, we architect a 77K-optimized NoC design by exploiting the fast inter-core wire (i.e., global wire) with our NoC modeling. First, we provide guidelines for designing cryogenic-friendly NoCs by analyzing representative NoCs at 77K. Next, based on the guidelines, we propose *CryoBus*, our fast and scalable cryogenic bus architecture which successfully supports the 64-core CPU systems.

5.1 Guidelines for Designing Cryogenic NoC

In the following subsections, we draw the key directions for designing a 77K-optimized NoC in terms of performance. To achieve the goal, we analyze the representative NoC designs (i.e., Mesh [17], Flattened Butterfly (FB) [32], Concentrated Mesh (CMesh) [8], and Shared bus [36]) depicted in Fig. 15. For the Shared bus (Fig. 15(d)), we scale a conventional bidirectional bus because there has been no single shared-bus topology supporting 64-core CPU systems.

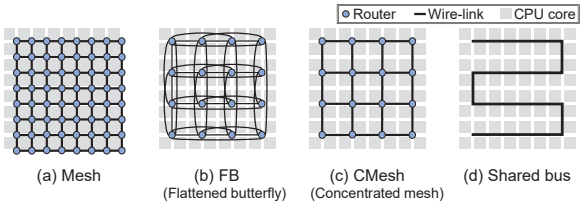

Figure 15: Representative NoC designs on a 64-core system

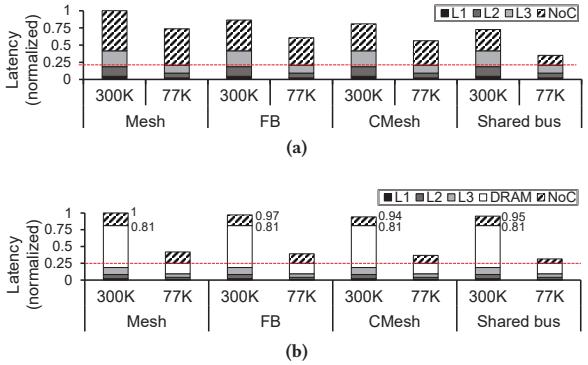

Figure 16: L3 cache (a) hit and (b) miss latency breakdown for various NoC designs running at 300K and 77K

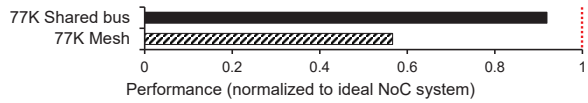

Figure 17: System-level performance of the 77K computer systems with Shared bus and Mesh for PARSEC workloads

We assume the 64-core CPU system with 64-MB shared L3 cache (1-MB L3 for each core) to analyze the scalability of various NoCs.

For the 300K NoCs, we conservatively assume the 1-cycle routers [34, 50] and the 4-hop/cycle wire links following the results from CACTI-NUCA [45] (i.e., 0.064ns for 2mm global-wire link at 4GHz in 45nm technology). Then, we derive the performance of 77K NoCs by adopting frequency-scaled routers and 12-hop/cycle wire links (as modeled in Section 3).

To thoroughly analyze the performance at 77K, we also include the cache and DRAM access latencies. For 300K cache and DRAM latency, we utilize CACTI-NUCA. For 77K memory latency, we utilize the existing 77K memory model, CryoRAM [37, 43].

Guideline #1. Utilizing the shared bus instead of the router-based NoCs. We first suggest to utilize the shared bus to maximize the performance at 77K. We analyze the performance of 77K-optimized memory with various 77K NoC designs, where all the caches and DRAMs are optimized for 77K [37, 43]. The detailed specifications are summarized in Table 4 (i.e., 77K memory).

Fig. 16 shows the L3 cache hit and miss latency breakdown for the various NoCs at 300K and 77K, where the values are normalized to those of 300K Mesh. We take the zero-load latency for each NoC latency. The red dotted lines in the figure indicate the latency

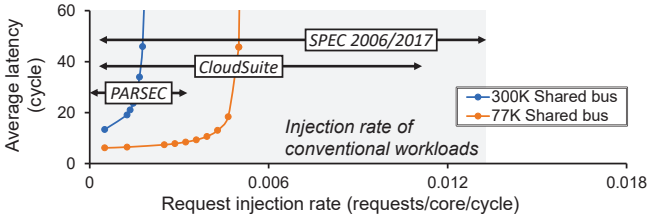

Figure 18: Load-latency results for 300K and 77K Shared bus

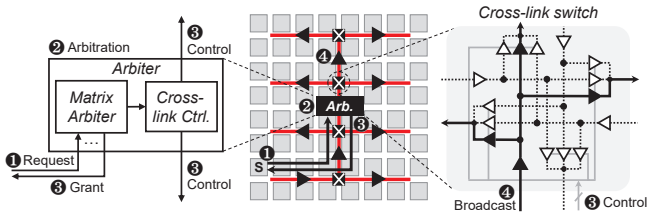

Figure 19: CryoBus overview and working mechanism

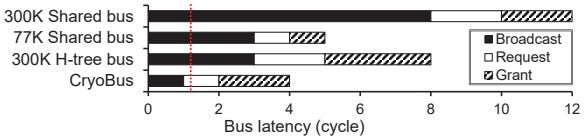

Figure 20: Latency breakdown for the various bus designs

assuming zero NoC latency at 77K. From the result, we first observe that the router-based NoCs (i.e., Mesh, FB, CMesh) become the performance bottleneck in the 77K-optimized memory systems. For example, if we adopt Mesh at 77K, NoC latency takes up to 71.7% and 40.4% of L3 hit and miss latency, respectively. This is because the router-based NoCs' latency barely decreases even at 77K, while the cache and DRAM access latencies are significantly reduced. Specifically, the speed-up of router-based NoCs is limited by the marginal frequency improvement at 77K (9.3% in our modeling result). Moreover, router-based NoCs benefit less from the wire-link speed-up as they typically have short links (e.g., maximum six hops in FB) which consume only one or two cycles even at 300K.

On the other hand, Shared bus can maximize the performance of the 77K-optimized memory system. At 300K, the L3 cache hit and miss latencies of Shared bus are comparable to those of the router-based NoCs. However, the latency of Shared bus greatly decreases at 77K because its latency is entirely composed of the wire-link latency. As a result, the L3 hit and miss latencies of 77K Shared bus nearly reach the zero-NoC-latency cases.

For further analysis, we evaluate the system-level performance of two 77K-optimized computer systems equipped with 77K Mesh and 77K Shared bus, respectively. Fig. 17 shows the average performance (i.e. the inverse of execution time) obtained from Gem5 simulations [12] for PARSEC workloads [11]. We normalize the values to that of the same system with an ideal NoC which has zero latency without contention and runs with snooping protocol. In Fig. 17, 77K Mesh suffers from the significant slowdown (43.3%) while 77K Shared bus achieves only 8.1% lower performance than the ideal scenario. Therefore, to fully exploit the faster wire at 77K, we should utilize Shared bus instead of the router-based NoCs.

Guideline #2. Maximizing the bus performance to overcome the contention problem. As the second guideline, we emphasize to further improve the performance of 77K Shared bus to overcome the contention problem. To draw the guideline, we use a cycle-accurate network simulator BookSim [28] to perform load-latency analysis for the Shared bus running at 300K and 77K. For our analysis, we also measure the per-core request injection rate (i.e., L2 MPKI) of various workloads including SPEC2006 [22], SPEC2017 [14], PARSEC [11], and CloudSuite [20] based on the Gem5 simulation and real-machine profiling. The range of the measured request injection rate for each workload is depicted in Fig. 18. Fig. 18 also shows the load-latency analysis of Shared bus running at 300K and 77K under the uniform random traffic. First, 300K Shared bus significantly suffers from the steeply increasing latency even under the very low request injection rate due to the bus contention. In our analysis, 300K Shared bus cannot run even the PARSEC workloads.

Even though 77K Shared bus significantly increases the performance, it still suffers from the severe contention problem. The fast 77K Shared bus can tolerate much higher request injection rate and thus can support the PARSEC workloads without performance degradation. However, 77K Shared bus is still too slow to run other representative workloads such as SPEC2006, SPEC2017, and CloudSuite in our target 64-core CPU system. Therefore, we should maximize the bus performance to successfully support the 64-core systems without the contention problem.

5.2 CryoBus: Fast and Scalable 77K Bus Design

In this section, we propose *CryoBus*, our fast cryogenic snooping-bus architecture which is scalable to the 64-core CPU system. The key idea is to enable a novel H-tree-shaped bus topology with the dynamic link connection scheme to minimize the broadcast latency.

5.2.1 CryoBus Overview. Fig. 19 shows the overview of CryoBus. The speed-up of CryoBus mainly originates from the H-tree-shaped bus topology. Specifically, in the 64-core system, the maximum distance between the cores is only 12 hops in CryoBus while 30 hops in the baseline Shared bus. However, the H-tree-shaped topology requires special architectural supports because we cannot build the topology with a simple bidirectional bus. Therefore, we propose a dynamic link connection scheme to control the signal transfer direction according to the location of the broadcasting core.

5.2.2 Dynamic Link Connection Mechanism. For the dynamic link connection, we implement the cross-link switches at the intersections of wire links and the cross-link controller in the arbiter, respectively. Fig. 19 shows the detailed working mechanism. When a CPU core tries to own the bus (source (S) in Fig. 19), it sends a request signal to the arbiter which locates at the center of a CPU die (❶ Request). Among the requesting cores, the arbiter selects one by using a matrix arbiter (❷ Arbitration). Based on the arbitration result, the matrix arbiter sends a grant signal to the chosen core (❸ Grant). At the same time, the cross-link controller sends a set of control signals to the cross-link switches to determine the wire-link connection according to the granted core's location (❸ Control). Then, the granted core broadcasts to all other cores through the

wire links which transfer the signal in the direction set by the cross-link switches (❹ Broadcast).

5.2.3 Performance and Power Consumption of CryoBus. We conduct performance and power analysis for CryoBus by assuming it operates with the 77K-optimized memory system. In this evaluation, we include the effect of the voltage optimization enabled at 77K. Specifically, as the LLC and NoC share the same voltage domain, we assume the same voltage level (i.e., V_{dd}, V_{th}) for the 77K caches and NoCs. As our model supports the voltage scaling, we include the effect of the voltage scaling in our performance and power results. The detailed setup are summarized in Table 4.

First, CryoBus successfully maximizes its performance and resolves the contention problem. Fig. 20 shows the latency breakdown for 300K Shared bus, 77K Shared bus, 300K H-tree-shaped bus, and CryoBus. The red dotted line indicates the target broadcast latency which can successfully support all the workloads in Fig. 18 without the contention problem. As the broadcasting latency mainly determines the bandwidth limit of the snooping bus, we highlight the target broadcast latency with the red line. CryoBus achieves the 1-cycle broadcast latency thanks to the fast wire link at 77K (12-hop/cycle) and the reduced broadcast distance (12 hops) of the H-tree bus topology. Even though CryoBus consumes an additional cycle for the control-signal generation (❸; added to Grant), it does not worsen the contention. Note that both the 77K cooling (77K Shared bus) and the bus topology optimization (300K H-tree bus) cannot achieve the target broadcast latency alone.

Next, CryoBus also achieves comparable bandwidth limit with the router-based NoC designs (e.g., CMesh, FB). Fig. 21 clarifies the scalability (i.e., tolerance against the contention) of CryoBus with the load-latency analysis for various NoCs (Fig. 15) running at 77K. To consider both realistic and conservative scenarios, we analyze the router-based NoCs for both 3-cycle router from industries [23, 25] and 1-cycle router from academia [34, 50], respectively (detailed router setup is in Table 4). As Fig. 21 shows, CryoBus becomes much more tolerable to the contention problem compared to 77K Shared bus thanks to our optimization. As a result, CryoBus not only covers all the representative workloads but also achieves comparable scalability with CMesh (3-cycle) and FB (3-cycle).

Finally, CryoBus consumes less power than other networks, even including its cooling power consumption. Fig. 22 shows the power analysis for 300K Mesh, 77K Mesh, 77K Shared bus and CryoBus. We also include the cooling cost (discussed in Section 6.1.2) in our power analysis. CryoBus consumes 57.2% less power than 300K Mesh because the 300K-dominant static power is almost eliminated at 77K and the dynamic power is greatly reduced thanks to the voltage optimization. In addition, CryoBus consumes 40.5% and 30.7% less power compared to 77K Mesh and 77K Shared bus, respectively. This is because CryoBus's dynamic link connection can minimize the number of activated links and avoid wasteful broadcasting when the destination of the packet is specified (i.e., data response).

6 EVALUATION

To evaluate CryoSP and CryoBus, we first introduce our evaluation methodology (Section 6.1) and then show their system-level performance (Section 6.2) and power consumption (Section 6.3).

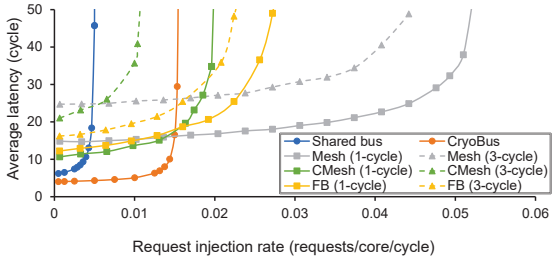

Figure 21: Load-latency analysis at uniform random pattern for various NoCs with voltage optimization applied at 77K

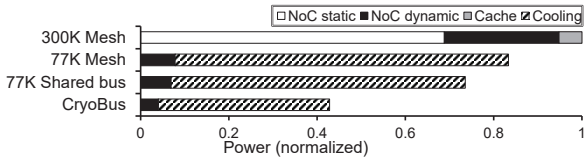

Figure 22: Power consumption of various NoCs with the voltage optimization applied at 77K

Table 4: Evaluation setup

Evaluation setup				
Design	Core type	# cores	NoC type	Memory type
Baseline (300K, Mesh)	300K Baseline (4GHz)	64	300K Mesh	300K Memory
CHP-core (77K, Mesh)	CHP-core (6.1GHz)	64	77K Mesh	77K Memory
CryoSP (77K, Mesh)	CryoSP (7.84GHz)	64	77K Mesh	77K Memory
CHP-core (77K, CryoBus)	CHP-core (6.1GHz)	64	CryoBus	77K Memory
CryoSP (77K, CryoBus)	CryoSP (7.84GHz)	64	CryoBus	77K Memory
NoC specification				
Design	Protocol	Frequency	Vdd/Vth	Router
300K Mesh	directory-based	4GHz	1.0V / 0.468V	4 VC
77K Mesh	directory-based	5.44GHz	0.55V / 0.225V	4 VC
CryoBus	snoop-based	4GHz	0.55V / 0.225V	N/A
Memory specification				
Design	L1 (private)	L2 (private)	L3 (shared)	DRAM random access latency
300K memory	32KB (4cyc @4GHz)	256KB (12cyc @4GHz)	1MB / core (20cyc @4GHz)	60.32ns
77K memory	32KB (2cyc @4GHz)	256KB (6cyc @4GHz)	1MB / core (10cyc @4GHz)	15.84ns

6.1 Evaluation Methodology

6.1.1 Performance Evaluation Methodology. We evaluate the performance of CryoSP and CryoBus with five combinations of core, NoC, and memory designs: (1) 300K Baseline core with 300K Mesh and 300K memory (*Baseline (300K, Mesh)*), (2) CHP-core with 77K Mesh and 77K memory (*CHP-core (77K, Mesh)*), (3) CryoSP with 77K Mesh and 77K memory (*CryoSP (77K, Mesh)*), (4) CHP-core with CryoBus and 77K memory (*CHP-core (77K, CryoBus)*), and (5) CryoSP with CryoBus and 77K memory (*CryoSP (77K, CryoBus)*). We summarize the setup in Table 4.

Core design. We use three core designs in evaluation: 300K baseline, previously-proposed cryogenic core (CHP-core), and CryoSP. We set 300K baseline core following 4GHz Intel Skylake i7-6700 specification. For CHP-core, we set its maximum frequency to 6.1GHz with 4-issue width following the previous work [16]. We set

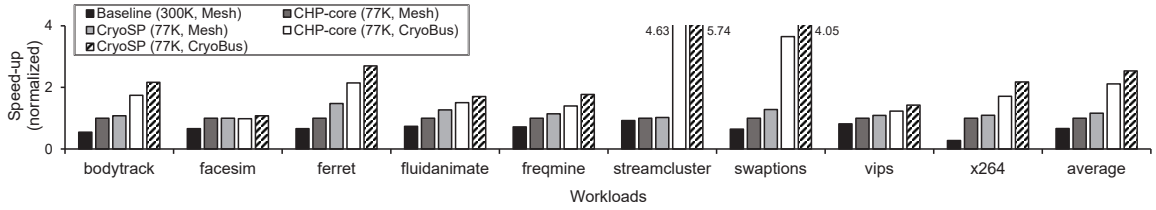

Figure 23: Multi-thread performance of PARSEC workloads for various system designs

maximum frequency of CryoSP to 7.84GHz with the same number of issue width and microarchitectural units of CHP-core.

NoC design. We use three types of NoC in our evaluation: 300K Mesh, 77K Mesh and CryoBus. For conservative evaluations, we assume that 300K Mesh and 77K Mesh utilize a state-of-the-art 1-cycle router instead of realistic 3-cycle router [23, 25, 34, 50]. Each router has 4 virtual channels (VC) per input, each with 3-flit buffers as in the previous work [33] and utilizes XY-routing for its routing method. Note that 300K and 77K Mesh use a directory-based cache coherency protocol while CryoBus uses a snoop-based protocol. The last-level cache slices in the mesh interconnect keep the directory states for the dedicated cache blocks.

Memory design. We use two memory hierarchies (i.e., cache and DRAM) in evaluation: 300K memory and 77K memory. For 300K memory, we utilize the cache specification of Intel i7-6700 and DRAM specification of DDR4-2400. For 77K memory, we utilize the cryogenic SRAM-based cache and previously proposed memory architecture (CLL-DRAM [37]). The 77K memory setup provides twice faster caches and 3.8 times faster DRAM compared to the 300K memory setup. By integrating 77K memory in our designs, we can fully evaluate the potential of the cryogenic computer, where all the 77K-optimal core, cache, DRAM and NoC are integrated.

6.1.2 Power Evaluation Methodology. We include both device power and cooling cost for the power consumption evaluation.

Device power model. To evaluate the power of various core and NoC designs running at 300K and 77K, we utilize McPAT [39] and Orion 2.0 [31], respectively. In addition, we integrate the tools with cryo-MOSFET [16] to derive the power consumption for various temperatures (i.e., 300K, 77K) and voltage levels (i.e., V_{dd}, V_{th}). For example, we derive the voltage level and leakage current at the target temperature from cryo-MOSFET, and feed them as inputs to McPAT and Orion to scale the dynamic and static power, respectively. We obtain the input access trace for McPAT and Orion from the Gem5 simulations [12] with PARSEC 2.1 workloads [11].

Cooling cost model. To accurately model the power efficiency of cryogenic computers, we include the cost for the cryogenic cooling. In our assumed LN-recycling cooling systems (i.e., Stinger systems shown in Fig. 4), the recurring cooling-power cost dominates the overall cooling cost [27, 41], as other one-time costs (i.e., cryo-cooler, LN cost) are much lower than the cooling power cost and even amortized every year. Therefore, we carefully model the cooling-power cost based on the real cryo-coolers' data as follows.

$$P_{cooling} = P_{dev} \cdot CO \quad (1)$$

$$P_{total} = P_{dev} + P_{cooling} = (1 + CO) \cdot P_{dev} = 10.65 \cdot P_{dev} \quad (2)$$

The cooling power ($P_{cooling}$) is calculated as Eq. (1), where P_{dev} is the power dissipated by the dev and the cooling overhead (CO) is the required power to remove unit heat (1W) from the cooling system [27]. We use 9.65 for the 77K cooling overhead taken from the real data of 77K cooling systems [27, 62]. Thus, we calculate the total power for the cryogenic computer (P_{total}) as Eq. (2).

6.2 System-Level Performance Evaluation

For the system-level performance evaluation, we perform Gem5 full-system simulations [12] running PARSEC 2.1 benchmarks [11] with Garnet [1]. Fig. 23 shows the performance of various system designs shown in Table 4. The performance is the inverse of the execution time and normalized to that of CHP-core (77K, Mesh). In summary, compared to state-of-the-art cryogenic computer system (CHP-core (77K, Mesh)), our proposed design with CryoSP and CryoBus (CryoSP (77K, CryoBus)) achieves 2.53x speed-up on average, up to 5.74x on *streamcluster*. Compared to the 300K computer system (Baseline (300K, Mesh)), our design achieves 3.82x speed-up on average, which shows the superiority of cryogenic computers.

First, with our core design, CryoSP (77K, Mesh) achieves 16.1% speed-up over the baseline cryogenic system (CHP-core (77K, Mesh)). Despite the penalties of superpipelining (i.e., higher branch misprediction), CryoSP improves the performance for every workload thanks to the 28.5% higher clock frequency. However, several workloads (i.e., *bodytrack*, *x264*) show marginal performance gain (7.7% and 9.2%, respectively) due to their memory-bounded nature.

Next, with our bus design, CHP-core (77K, CryoBus) achieves 2.1x speed-up on average, up to 4.63x in *streamcluster*. The significant speed-up of *streamcluster* can be explained by the number of barrier instructions. As *streamcluster* contains large number of barrier instructions [11], *streamcluster* highly benefits from the snoop-based protocol enabled by our CryoBus interconnect. Workloads with frequent cache and memory accesses (i.e., *bodytrack*, *ferret*, *swaptions*) also highly benefit from the reduced L3 cache and memory access latency with 74%, 114%, and 265% speed-up respectively. Also note that CryoBus successfully resolves the contention problem and boost the performance of all workloads.

Finally, CryoSP (77K, CryoBus) achieves the highest performance gain among all designs, with 2.53x speed-up on average, up to 5.74x in *streamcluster*. Note that CryoSP (77K, CryoBus) shows the synergistic effect by improving both the core and NoC performance. For several workloads (e.g., *bodytrack*, *streamcluster*, *x264*), the speed-up of CryoSP with CryoBus (CryoSP (77K, CryoBus)) is much higher than the sum of CryoSP (CryoSP (77K, Mesh)) and CryoBus only (CHP-core (77K, CryoBus)), which indicates the synergistic effect of

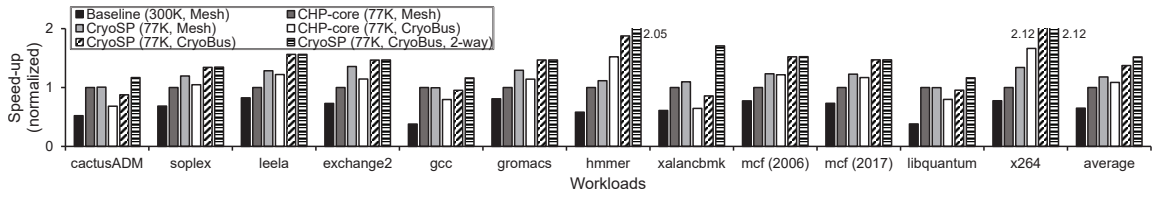

Figure 24: Single-thread performance of SPEC 2006 and 2017 workloads with an aggressive stride prefetcher

CryoSP and CryoBus. As a result, our cryogenic system equipped with CryoSP and CryoBus greatly improves the system performance by 3.82x and 2.53x on average, compared to baseline 300K and 77K systems, respectively.

6.3 Power Consumption Evaluation

Our power evaluation in Section 4.5 and Section 5.2.3 implies that the entire cryogenic-computer system consumes less power than the conventional systems, even after including the huge cooling power (which is 9.65x the device power). In the previous studies, the full-cryogenic computer system's power efficiency is successfully shown with the 77K-optimized CPU [15, 16], cache [43], and memory devices [37]. As we show the power efficiency of our modifications (i.e., CryoSP, CryoBus) on the top of them, we also achieve the entire system's power efficiency.

7 DISCUSSION

7.1 CryoBus with Memory-Intensive Scenarios

CryoBus is still effective even for the much more memory-intensive scenarios. Fig. 24 shows the Gem5 simulation result using SPEC2006 and SPEC2017 workloads. To emulate memory-intensive scenarios, we concurrently run 64 copies of SPEC workloads in the 64 core systems. To further increase the bus contention, we utilize the inefficient stride prefetcher to frequently access the shared bus. We make the prefetchers be activated even at the cache hits for the same reason. Even with the extreme scenarios, CryoBus (CryoSP (77K, CryoBus)) achieves 2.11 times and 37.2% higher speed-up than Baseline (300K, mesh) and CHP-core (77K, mesh), respectively.

In addition, even if the bus contention can become the performance bottleneck (as shown in *cactusADM, gcc, xalancbmk, libquantum* of Fig. 24), we can resolve the CryoBus's contention problem by using orthogonal bus bandwidth scaling works. For example, Fig. 24 also shows the performance of CryoBus when the existing address interleaving scheme is applied. Address interleaving bus is the concept of using multiple buses for different addresses [19, 64] by fully utilizing the abundant global wires inside chips [4]. Fig. 24 clearly shows that CryoBus with the 2-way address interleaving (CryoSP (77K, CryoBus, 2-way)) resolves the contention problem, and achieves the highest speed-up for every workload (with 2.34 times and 52% higher speed-up than Baseline (300K, mesh) and CHP-core (77K, mesh), respectively). As several previous works already showcased the snooping bus with from 2 to 8-way of address interleaving [19, 64], we believe that CryoBus also well operates without contention problems for the most workloads.

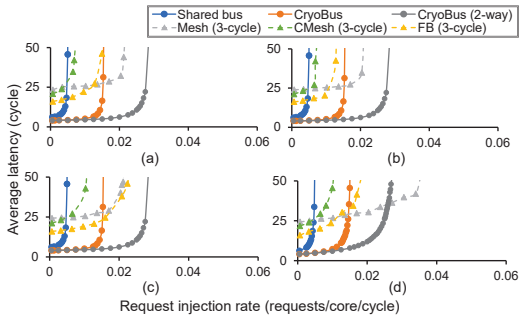

Figure 25: Load-latency analysis at various traffic patterns; (a) Transpose, (b) Hotspot, (c) Bit Reverse, and (d) Burst.

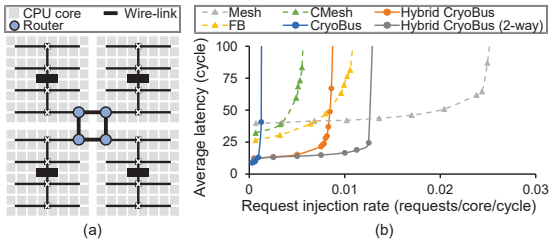

Figure 26: (a) Hybrid CryoBus architecture for 256-core CPU and (b) its load-latency result at uniform random pattern

7.2 Load Latency of Other Traffic Patterns

Regardless of the traffic pattern, CryoBus always achieves the lowest latency and comparable bandwidth with other NoCs. Fig. 25 shows the load-latency results for various NoCs under (a) Transpose, (b) Hotspot, (c) Bit Reverse, and (d) Burst traffic patterns. As the uniform random is the most favorable traffic for router-based NoCs (i.e., Mesh, CMesh, FB), their results are significantly degraded compared to the uniform-random results in Fig. 21. However, CryoBus shows similar results regardless of the traffic, and it even achieves better bandwidth than CMesh and FB at Transpose and Hotspot. We also confirm that CryoBus with 2-way interleaving (CryoBus (2-way)) attains comparable bandwidth with Mesh, while achieving six times lower zero-load latency.

7.3 Scaling CryoBus to 64+ Cores

To scale CryoBus beyond 64 cores, we can apply a directory-based hybrid NoC topology [18] along with CryoBus. Fig. 26(a) shows the 256-core directory-based hybrid CryoBus which consists of four

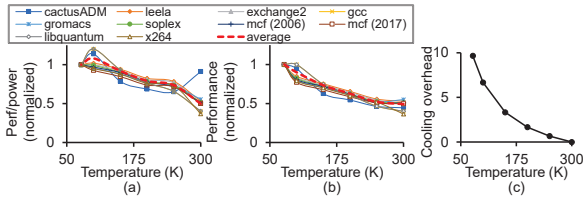

Figure 27: (a) Performance/power, (b) performance, and (c) cooling overhead for various temperatures. We assume 30% of Carnot for cooling overhead of various temperatures.

CryoBus clusters and a global mesh network. Fig. 26(b) is the load-latency analysis result of hybrid CryoBus and other router-based NoCs on a 256-core scale. Hybrid CryoBus achieves the lowest latency among all designs and it scales comparably with other router-based NoCs. With 2-way interleaving, its scalability can further increase. Although the hybrid CryoBus gives up the snooping protocol, it is still effective thanks to the significantly reduced latency with enough bandwidth covering major workloads.

7.4 Other Optimal Temperatures

We utilize 77K as our target temperature because the simple and powerful LN cooling can stably maintain 77K. However, there can be other sweet spot due to the trade-off between the transistor/wire speed-up and cooling cost. For example, Fig. 27 shows performance/power at various temperatures when running SPEC 2006 and 2017 workloads. To derive the results, we assume that the clock frequency and our target voltage level for each temperature are linearly scaled with the target temperature. For the system design, we utilize Baseline (300K, Mesh) for 300K design and CryoSP (77K, CryoBus) for other designs. Fig. 27 shows that 100K computing achieves higher performance/power on average than 77K computing. It is due to the exponentially increasing cooling overhead (CO) with the temperature reduction, while the server performance almost linearly changes with the temperature. As the exact sweet-spot temperatures highly depend on the workload, microarchitecture, target voltage level, and target metrics (e.g., TCO/performance), finding the optimal temperature will be the promising future work.

7.5 Wires in Smaller Technologies

The wire speed-up can be reduced in smaller nodes because the degree of wire-resistivity reduction shrinks in thinner wires [52]. However, we believe that our observations and microarchitectures are still useful for the following reasons. First, the wires that we mainly use in our research (i.e., semi-global wires for data forwarding, global wires for network interconnect) are thick enough to benefit from 77K even in the latest process. For example, the pitch of semi-global (e.g., M7 and M8) and global wires (e.g., M9 and M10) of Intel's latest process (i.e., Intel 10nm process or Intel 7 after rebranding) are 112nm and 160nm, respectively [6]. Second, we can draw our target wires thicker to prevent the reduction of cryogenic wire's benefits. As our target wires (i.e., data-forwarding wire, networking wire) occupy a small portion in a chip, we can maintain the cryogenic wire's speed-up with small additional cost.

8 RELATED WORK

Critical-path delay analysis at 77K. Palacharla et al. [48, 49] modeled the critical-path delays of major pipeline stages with varying issue width and technology. Li et al. [39] built power, area, and timing model of modern processor designs, following the modeling of the previous works [48, 49]. However, no previous works cover most of the pipeline stages from Verilog-level hardware. They also do not cover the various temperatures including 77K.

Large-scale bus. Das et al. [18] proposed a bus-router hybrid topology for 64-core CPU while giving up the powerful snooping protocol. Udipi et al. [64] proposed segmented bus, which divided the large bus network into small segment bus, to reduce the NoC power consumption. However, there is no previous work which successfully scales the single snooping bus in 64-core system.

77K cryogenic computing. Lee et al. [37] and Min et al. [43] and Byun et al. [15, 16] developed the 77K DRAM, cache, and pipeline modeling tool, and propose the DRAM, cache, and CPU core architecture at 77K. Tannu et al. [59] and Rambus [66] showed commodity DRAM can reliably operate at 77K. Lee et al. [38] found out the critical reliability issue in cryogenic DRAM running showed the solutions. Resch et al. [53] and Alam et al. [3] and Hou et al. [24] showed the cryogenic PIM architecture to resolve the memory wall. Sanuki et al. [55] and Aiba et al. [2] developed 3D flash prototype for cryogenic computing and showed the potential of higher endurance, capacity, and speed-up. Saligram et al. [54] showed the characteristics of processors for various cryogenic temperatures and voltages. However, no previous works proposed the novel pipeline and NoC architecture to exploit the fast cryogenic wires.

To the best of our knowledge, our work is the first study to model, analyze, and optimize the CPU pipeline and NoC architecture at 77K with significant performance improvement.

9 CONCLUSION

This paper proposed a CPU microarchitecture which extensively exploits the fast wires at 77K. For the purpose, we first conducted an in-depth analysis using our model, and derived key observations for developing the 77K-optimal pipeline and network on chip (NoC). Next, based on the observations, we architected *CryoSP* and *CryoBus* as our optimal pipeline and NoC designs, respectively. Our evaluation showed that our proposed CPU architecture can significantly improve the performance with little total power consumption.

ACKNOWLEDGMENTS

This work was supported in part by the Samsung Electronics, National Research Foundation of Korea (NRF) funded by the Korean Government under Grants NRF-2019R1A5A1027055, NRF-2021R1A2C3014131, NRF-2021M3F3A2A02037893, and NRF-2019-Global Ph.D. Fellowship Program; in part by the Institute of Information & Communications Technology Planning & Evaluation funded by the Korean Government under Grant 1711080972, 2021000853; in part by the Creative Pioneering Researchers Program through Seoul National University; and in part by the Automation and Systems Research Institute (ASRI) and Inter-university Semiconductor Research Center at Seoul National University.

REFERENCES

[1] Niket Agarwal, Tushar Krishna, Li-Shiuan Peh, and Niraj K Jha. 2009. GARNET: A detailed on-chip network model inside a full-system simulator. In *2009 IEEE international symposium on performance analysis of systems and software*. IEEE, 33–42. https://doi.org/10.1109/ISPASS.2009.4919636

[2] Yuta Aiba, Hitomi Tanaka, Takashi Maeda, Keiichi Sawa, Fumie Kikushima, Masayuki Miura, Toshio Fujisawa, Mie Matsuo, Hideto Horii, Hideko Mukaida, et al. 2021. Bringing in Cryogenics to Storage: Characteristics and Performance Improvement of 3D Flash Memory. In *2021 IEEE International Memory Workshop (IMW)*. IEEE, 1–4. https://doi.org/10.1109/IMW51353.2021.9439594

[3] Shamiul Alam, Md Mazharul Islam, Nazmul Amin, Md Shafayat Hossain, Akhilesh Jaiswal, and Ahmedullah Aziz. 2021. CryoCiM: Cryogenic Compute-in-Memory based on the Quantum Anomalous Hall Effect. *arXiv preprint arXiv:2112.00124* (2021).

[4] Fawaz Alazemi, Arash Azizimazreah, Bella Bose, and Lizhong Chen. 2018. Routerless network-on-chip. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 492–503. https://doi.org/10.1109/HPCA.2018.00049

[5] Krste Asanovic, David A Patterson, and Christopher Celio. 2015. *The berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized risc-v processor*. Technical Report. University of California at Berkeley Berkeley United States.

[6] Chris Auth, A Aliyarukunju, M Asoro, D Bergstrom, V Bhagwat, J Birdsall, N Bisnik, M Buehler, V Chikarmane, G Ding, et al. 2017. A 10nm high performance and low-power CMOS technology featuring 3 rd generation FinFET transistors, Self-Aligned Quad Patterning, contact over active gate and cobalt local interconnects. In *2017 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 29–1. https://doi.org/10.1109/IEDM.2017.8268472

[7] Rajeev Balasubramonian, Sandhya Dwarkadas, and David H Albonesi. 2003. Dynamically managing the communication-parallelism trade-off in future clustered processors. In *30th Annual International Symposium on Computer Architecture, 2003. Proceedings.* IEEE, 275–286. https://doi.org/10.1109/ISCA.2003.1207007

[8] James Balfour and William J Dally. 2006. Design tradeoffs for tiled CMP on-chip networks. In *ACM International conference on supercomputing 25th anniversary volume*. 390–401. https://doi.org/10.1145/2591635.2667187

[9] Kaustav Banerjee, Shukri J Souri, Pawan Kapur, and Krishna C Saraswat. 2001. 3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration. *Proc. IEEE* 89, 5 (2001), 602–633. https://doi.org/10.1109/5.929647

[10] Daniel U Becker. 2012. *Efficient microarchitecture for network-on-chip routers*. Stanford University.

[11] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. The PARSEC benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*. 72–81. https://doi.org/10.1145/1454115.1454128

[12] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. 2011. The gem5 simulator. *ACM SIGARCH computer architecture news* 39, 2 (2011), 1–7. https://doi.org/10.1145/2024716.2024718

[13] Eric Borch, Eric Tune, Srilatha Manne, and Joel Emer. 2002. Loose loops sink chips. In *Proceedings Eighth International Symposium on High Performance Computer Architecture*. IEEE, 299–310. https://doi.org/10.1109/HPCA.2002.995719

[14] James Bucek, Klaus-Dieter Lange, and Jóakim v. Kistowski. 2018. SPEC CPU2017: Next-generation compute benchmark. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*. 41–42. https://doi.org/10.1145/3185768.3185771

[15] Ilkwon Byun, Dongmoon Min, Gyuhyeon Lee, Seongmin Na, and Jangwoo Kim. 2021. A Next-Generation Cryogenic Processor Architecture. *IEEE Micro* 41, 3 (2021), 80–86. https://doi.org/10.1109/MM.2021.3070133

[16] Ilkwon Byun, Dongmoon Min, Gyu-hyeon Lee, Seongmin Na, and Jangwoo Kim. 2020. CryoCore: A Fast and Dense Processor Architecture for Cryogenic Computing. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 335–348. https://doi.org/10.1109/ISCA45697.2020.00037

[17] William J Dally and Brian Towles. 2001. Route packets, not wires: on-chip inteconnection networks. In *Proceedings of the 38th annual design automation conference*. 684–689. https://doi.org/10.1145/378239.379048

[18] Reetuparna Das, Soumya Eachempati, Asit K Mishra, Vijaykrishnan Narayanan, and Chita R Das. 2009. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs. In *2009 IEEE 15th International Symposium on High Performance Computer Architecture*. Ieee, 175–186. https://doi.org/10.1109/HPCA.2009.4798252

[19] Ronald G Dreslinski, Thomas Manville, Korey Sewell, Reetuparna Das, Nathaniel Pinckney, Sudhir Satpathy, David Blaauw, Dennis Sylvester, and Trevor Mudge. 2012. XPoint cache: Scaling existing bus-based coherence protocols for 2D and 3D many-core systems. In *2012 21st International Conference on Parallel Architectures and Compilation Techniques (PACT)*. IEEE, 75–85. https://doi.org/10.1145/2370816.2370829

[20] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. 2012. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. *Acm sigplan notices* 47, 4 (2012), 37–48. https://doi.org/10.1145/2248487.2150982

[21] Antonio Gonzalez, Fernando Latorre, and Grigorios Magklis. 2010. Processor microarchitecture: An implementation perspective. *Synthesis Lectures on Computer Architecture* 5, 1 (2010), 1–116. https://doi.org/10.2200/S00309ED1V01Y201011CAC012

[22] John L Henning. 2006. SPEC CPU2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News* 34, 4 (2006), 1–17.

[23] Yatin Hoskote, Sriram Vangal, Arvind Singh, Nitin Borkar, and Shekhar Borkar. 2007. A 5-GHz mesh interconnect for a teraflops processor. *IEEE micro* 27, 5 (2007), 51–61. https://doi.org/10.1109/MM.2007.4378783

[24] Yaoru Hou, We Ge, Yanan Guo, Lirida Naviner, You Wang, Bo Liu, Jun Yang, and Hao Cai. 2021. Cryogenic In-MRAM Computing. In *2021 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. IEEE, 1–6. https://doi.org/10.1109/NANOARCH53687.2021.9642238

[25] Jason Howard, Saurabh Dighe, Yatin Hoskote, Sriram Vangal, David Finan, Gregory Ruhl, David Jenkins, Howard Wilson, Nitin Borkar, Gerhard Schrom, et al. 2010. A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS. In *2010 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 108–109. https://doi.org/10.1109/ISSCC.2010.5434077

[26] Intel. 2021. Intel Xeon Platinum 9282 Processor. https://ark.intel.com/content/www/us/en/ark/products/194146/intel-xeon-platinum-9282-processor-77m-cache-2-60-ghz.html [Online Accessed, 12-August-2021].

[27] Yukikazu Iwasa. 2009. *Case studies in superconducting magnets: design and operational issues*. Springer Science & Business Media.

[28] Nan Jiang, Daniel U Becker, George Michelogiannakis, James Balfour, Brian Towles, David E Shaw, John Kim, and William J Dally. 2013. A detailed and flexible cycle-accurate network-on-chip simulator. In *2013 IEEE international symposium on performance analysis of systems and software (ISPASS)*. IEEE, 86–96. https://doi.org/10.1109/ISPASS.2013.6557149

[29] Daniel A Jiménez. 2003. Reconsidering complex branch predictors. In *The Ninth International Symposium on High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings.* IEEE, 43–52. https://doi.org/10.1109/HPCA.2003.1183523

[30] Daniel A Jiménez, Stephen W Keckler, and Calvin Lin. 2000. The impact of delay on the design of branch predictors. In *Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*. 67–76. https://doi.org/10.1109/MICRO.2000.898059

[31] Andrew B Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. 2009. ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration. In *2009 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 423–428. https://doi.org/10.1109/DATE.2009.5090700

[32] John Kim, James Balfour, and William Dally. 2007. Flattened butterfly topology for on-chip networks. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*. 172–182. https://doi.org/10.1109/MICRO.2007.29

[33] Tushar Krishna, Chia-Hsin Owen Chen, Woo Cheol Kwon, and Li-Shiuan Peh. 2013. Breaking the on-chip latency barrier using SMART. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 378–389. https://doi.org/10.1109/HPCA.2013.6522334

[34] Tushar Krishna, Jacob Postman, Christopher Edmonds, Li-Shiuan Peh, and Patrick Chiang. 2010. Swift: A swing-reduced interconnect for a token-based network-on-chip in 90nm cmos. In *2010 IEEE International Conference on Computer Design*. IEEE, 439–446. https://doi.org/10.1109/ICCD.2010.5647666

[35] Rakesh Kumar, Dean M Tullsen, Norman P Jouppi, and Parthasarathy Ranganathan. 2005. Heterogeneous chip multiprocessors. *Computer* 38, 11 (2005), 32–38. https://doi.org/10.1109/MC.2005.379

[36] Rakesh Kumar, Victor Zyuban, and Dean M Tullsen. 2005. Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling. In *32nd International Symposium on Computer Architecture (ISCA'05)*. IEEE, 408–419. https://doi.org/10.1109/ISCA.2005.34

[37] Gyu-hyeon Lee, Dongmoon Min, Ilkwon Byun, and Jangwoo Kim. 2019. Cryogenic Computer Architecture Modeling with Memory-Side Case Studies. In *Proceedings of the 46th International Symposium on Computer Architecture* (Phoenix, Arizona) *(ISCA '19)*. Association for Computing Machinery, New York, NY, USA, 774–787. https://doi.org/10.1145/3307650.3322219

[38] Gyu-Hyeon Lee, Seongmin Na, Ilkwon Byun, Dongmoon Min, and Jangwoo Kim. 2021. CryoGuard: A near refresh-free robust DRAM design for cryogenic computing. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 637–650. https://doi.org/10.1109/ISCA52012.2021.00056

[39] Sheng Li, Jung Ho Ahn, Richard D Strong, Jay B Brockman, Dean M Tullsen, and Norman P Jouppi. 2009. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 469–480. https://doi.org/10.1145/1669112.1669172

[40] Gabriel H Loh. 2006. Revisiting the performance impact of branch predictor latencies. In *2006 IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE, 59–69. https://doi.org/10.1109/ISPASS.2006.1620790

[41] William L Luyben. 2017. Estimating refrigeration costs at cryogenic temperatures. *Computers & Chemical Engineering* 103 (2017), 144–150. https://doi.org/10.1016/j.compchemeng.2017.03.013

[42] Richard Allen Matula. 1979. Electrical resistivity of copper, gold, palladium, and silver. *Journal of Physical and Chemical Reference Data* 8, 4 (1979), 1147–1298. https://doi.org/10.1063/1.555614

[43] Dongmoon Min, Ilkwon Byun, Gyu-Hyeon Lee, Seongmin Na, and Jangwoo Kim. 2020. CryoCache: A Fast, Large, and Cost-Effective Cache Architecture for Cryogenic Computing. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) *(ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 449–464. https://doi.org/10.1145/3373376.3378513

[44] Kaizad Mistry, C Allen, Chris Auth, Bruce Beattie, D Bergstrom, M Bost, M Brazier, M Buehler, A Cappellani, R Chau, et al. 2007. A 45nm logic technology with high-k+ metal gate transistors, strained silicon, 9 Cu interconnect layers, 193nm dry patterning, and 100% Pb-free packaging. In *2007 IEEE International Electron Devices Meeting*. IEEE, 247–250. https://doi.org/10.1109/IEDM.2007.4418914

[45] Naveen Muralimanohar, Rajeev Balasubramonian, and Norm Jouppi. 2007. Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*. 3–14. https://doi.org/10.1109/MICRO.2007.33

[46] Inc NanGate. 2008. NanGate FreePDK45 Open Cell Library.

[47] BA Nayfeh and K Olukotun. 1997. A single-chip multiprocessor. *Computer* 30, 9 (1997), 79–85. https://doi.org/10.1109/2.612253

[48] Subbarao Palacharla, Norman P Jouppi, and James E Smith. 1996. *Quantifying the complexity of superscalar processors*. Technical Report. University of Wisconsin-Madison Department of Computer Sciences.

[49] Subbarao Palacharla, Norman P Jouppi, and James E Smith. 1997. *Complexity-effective superscalar processors*. Vol. 25. ACM. https://doi.org/10.1145/264107.264201

[50] Sunghyun Park, Tushar Krishna, Chia-Hsin Chen, Bhavya Daya, Anantha Chandrakasan, and Li-Shiuan Peh. 2012. Approaching the theoretical limits of a mesh NoC with a 16-node chip prototype in 45nm SOI. In *Proceedings of the 49th Annual Design Automation Conference*. 398–405. https://doi.org/10.1145/2228360.2228431

[51] Chris H Perleberg and Alan Jay Smith. 1993. Branch target buffer design and optimization. *IEEE transactions on computers* 42, 4 (1993), 396–412. https://doi.org/10.1109/12.214687

[52] JJ Plombon, Ebrahim Andideh, Valery M Dubin, and Jose Maiz. 2006. Influence of phonon, geometry, impurity, and grain size on copper line resistivity. *Applied physics letters* 89, 11 (2006), 113124. https://doi.org/10.1063/1.2355435

[53] Salonik Resch, Husrev Cilasun, and Ulya Karpuzcu. 2021. Cryogenic PIM: Challenges & Opportunities. *IEEE Computer Architecture Letters* (2021). https://doi.org/10.1109/LCA.2021.3077536

[54] Rakshith Saligram, Divya Prasad, David Pietromonaco, Arijit Raychowdhury, and Brian Cline. 2021. A 64-Bit Arm CPU at Cryogenic temperatures: Design Technology Co-Optimization for Power and Performance. In *2021 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 1–2. https://doi.org/10.1109/CICC51472.2021.9431559

[55] Tomoya Sanuki, Yuta Aiba, Hitomi Tanaka, Takashi Maeda, Keiichi Sawa, Fumie Kikushima, and Masayuki Miura. 2021. Cryogenic Operation of 3D Flash Memory for Storage Performance Improvement and Bit Cost Scaling. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* (2021). https://doi.org/10.1109/JXCDC.2021.3123783

[56] Oleg Semenov, Arman Vassighi, and Manoj Sachdev. 2002. Impact of technology scaling on thermal behavior of leakage current in sub-quarter micron MOSFETs: perspective of low temperature current testing. *Microelectronics Journal* 33, 11 (2002), 985–994. https://doi.org/10.1016/S0026-2692(02)00071-X

[57] André Seznec and Antony Fraboulet. 2003. Effective ahead pipelining of instruction block address generation. In *30th Annual International Symposium on Computer Architecture, 2003. Proceedings.* IEEE, 241–252. https://doi.org/10.1109/ISCA.2003.1207004

[58] Synopsys. 2021. Synopsys DC Ultra. https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html [Online Accessed, 12-August-2021].

[59] Swamit S Tannu, Douglas M Carmean, and Moinuddin K Qureshi. 2017. Cryogenic-DRAM based memory system for scalable quantum computers: a feasibility study. In *Proceedings of the International Symposium on Memory Systems*. ACM, 189–195. https://doi.org/10.1145/3132402.3132436

[60] M Bedford Taylor, Walter Lee, Saman Amarasinghe, and Anant Agarwal. 2003. Scalar operand networks: On-chip interconnect for ILP in partitioned architectures. In *The Ninth International Symposium on High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings.* IEEE, 341–353. https://doi.org/10.1109/HPCA.2003.1183551

[61] Michael Bedford Taylor, Walter Lee, Jason Miller, David Wentzlaff, Ian Bratt, Ben Greenwald, Henry Hoffmann, Paul Johnson, Jason Kim, James Psota, et al. 2004. Evaluation of the Raw microprocessor: An exposed-wire-delay architecture for ILP and streams. *ACM SIGARCH Computer Architecture News* 32, 2 (2004), 2. https://doi.org/10.1145/1028176.1006733

[62] Hermanus JM ter Brake and GFM Wiegerinck. 2002. Low-power cryocooler survey. *Cryogenics* 42, 11 (2002), 705–718. https://doi.org/10.1016/S0011-2275(02)00143-1

[63] Shyamkumar Thoziyoor, Naveen Muralimanohar, Jung Ho Ahn, and Norman P Jouppi. 2008. *CACTI 5.1*. Technical Report. Technical Report HPL-2008-20, HP Labs.

[64] Aniruddha N Udipi, Naveen Muralimanohar, and Rajeev Balasubramonian. 2010. Towards scalable, energy-efficient, bus-based on-chip networks. In *HPCA-16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*. IEEE, 1–12. https://doi.org/10.1109/HPCA.2010.5416639

[65] Sriram Vangal, Jason Howard, Gregory Ruhl, Saurabh Dighe, Howard Wilson, James Tschanz, David Finan, Priya Iyer, Arvind Singh, Tiju Jacob, et al. 2007. An 80-tile 1.28 TFLOPS network-on-chip in 65nm CMOS. In *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*. IEEE, 98–589. https://doi.org/10.1109/ISSCC.2007.373606

[66] Fiona Wang, Thomas Vogelsang, Brent Haukness, and Stephen C Magee. 2018. DRAM retention at cryogenic temperatures. In *2018 IEEE International Memory Workshop (IMW)*. IEEE, 1–4. https://doi.org/10.1109/IMW.2018.8388826

[67] Wikichip. 2021. Intel Skylake Client core floorplan. https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(client).

[68] Linda Wilson. 2013. International technology roadmap for semiconductors (ITRS). *Semiconductor Industry Association* 1 (2013). https://doi.org/10.1007/978-3-642-23096-7_7