NoC Frequency Scaling with Flexible-Pipeline Routers

Pingqiang Zhou*, Jieming Yin†, Antonia Zhai † and Sachin S. Sapatnekar*
* Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, Minnesota 55455
Email: {pingqiang, sachin}@umn.edu
† Department of Computer Science and Engineering, University of Minnesota, Minneapolis, Minnesota 55455
Email: {jyin, zhai}@cs.umn.edu

***Abstract*—Voltage and frequency scaling (VFS) for NoC can potentially reduce energy consumption, but the associated increase in latency and degradation in throughput limits its deployment. We propose flexible-pipeline routers that reconfigure pipeline stages upon VFS, so that latency through such routers remains constant. With minimal hardware overhead, the deployment of such routers allows us to reduce network frequency and save network energy, without significant performance degradation. Furthermore, we demonstrate the use of simple performance metrics to determine the optimal operation frequency, considering the energy/performance impact on *all* aspects of the system - the cores, the caches and the interconnection network.**

***Index Terms*—Chip Multiprocessors, Interconnects, NoC, Pipelined Router, Voltage Scaling, Frequency Scaling.**

I. Introduction

Advances in semiconductor technology have led to continuous increases in device density and larger system sizes. Concomitant with exponentially reducing device dimensions, designers face new challenges in maximizing computation while remaining with a stringent power envelope. Over the last decade, chip multiprocessors (CMPs) have emerged as a potential solution to address some of these problems by integrating multiple smaller and more energy efficient cores in order to replace a single, larger core. These cores must communicate through an efficient on-chip interconnection network (NoC), and NoC design is vital to both performance and power.

If incorrectly designed and/or poorly utilized, NoCs can become a major performance bottleneck and a significant source of power consumption for CMP systems [1]–[3]. As CMP-based systems become the main powerhouse for computation, they must serve diverse computing needs; and thus the on-die NoCs must be designed for a variety of traffic patterns. The integration of heterogeneous cores [3], [4] onto a single die further aggravates this situation, since cores with different computation capability have different performance goals. By identifying the performance requirements of each core, it might be possible to reduce the energy consumption of the NoC, while achieving the same overall performance.

State-of-the-art NoC designs often use packet-switched routers to support high bandwidth traffic. Under this model, it often takes multiple hops for messages to reach their destinations, and the energy/delay associated with packets traversing through routers is the dominating factor. There have been several proposals for reducing the performance penalty, such as router bypassing [5]–[7] and enhancing router pipeline design [8]–[10]. There also exists a large body of work on reducing router energy consumption, which corresponds to a large portion of NoC energy [1], [2].

A critical design parameter that directly affects both performance and power of NoC is the network frequency. Techniques such as VFS [11]–[14] have been widely investigated to allow the network to operate at a lower frequency to reduce energy consumption, when possible. Of which [11]–[13] specifically focus on link latency or power. Mishra *et al.* [14] use VFS on the NoC routers to reduce the network power consumption. However, reducing NoC frequency increases latency and reduces network throughput, which in turn, degrades overall system performance. An unfortunate side-effect of performance degradation is that the processors must be kept active for a longer duration, causing them to consume more leakage power: this factor has become increasingly dominant in nanometer-scale technologies. As a result, in prior work, frequency is only moderately scaled, by up to 20%, with the network and the cores operating asynchronously at different frequencies. To fully realize the potential of VFS, it is desirable to be able to scale down the network frequency significantly (our work examines changes of $2\times$-$4\times$) up to the point where the performance penalty remains small. Moreover, since our frequency scaling uses integer multiples of the clock frequency, we can continue to operate within a fully synchronous paradigm.

Our performance analysis indicates that different applications and cores are sensitive to different NoC performance metrics. For some applications, a reduction in NoC throughput has relatively little impact on performance, but increasing NoC latency can cause significant performance degradation. For these types of workloads, we propose to reconfigure the router pipeline when scaling down the network frequency. With this technique, the impact on the average time to traverse the NoC for a single message is minimal, while the peak throughput of the NoC degrades. Thus, for workloads that have a low or moderate throughput requirements, but are sensitive to the NoC latencies, this technique allows us to reduce NoC energy consumption, without significant performance degradation. We refer to the proposed routers as *flexible-pipeline* routers, since the router pipeline stages are adaptively configured based on the workload.

To demonstrate the effectiveness of the proposed technique, we build a flexible-pipeline router based on a classic 4-stage baseline pipeline. In general, when NoC utilization is low, NoC frequency is scaled down. Meanwhile, the router pipeline is reconfigured by combining some pipeline stages, while using the same latch boundaries. We use time-borrowing techniques to ensure that this delay-unbalanced reconfigured pipeline runs at the optimal clock period. We evaluate the proposed router in a chip multiprocessor connected by a mesh NoC. In this work, we evaluate the performance of static flexible-pipelines, where the pipeline structure is unchanged during operation. Dynamic flexible-pipelines, which dynamically change the pipeline structure over runtime, may be used instead to achieve further benefits but are beyond the scope of our work at this time. Our results demonstrate that for a large class of applications that are only sensitive to network latency, NoC frequency can be dramatically scaled down without significant performance degradation. This leads to improvement in NoC energy efficiency.

Any solution that uses pipelined routers for energy savings must satisfy two requirements: (1) The hardware overhead must be low: The overhead for our flexible-pipeline router is minimal, and it only requires a set of multiplexers to bypass registers and alter clock skews. Our analysis in Section II-B3 shows that the cost of the additional hardware is less than 3% compared to the classic 4-stage baseline router. (2) The savings must be demonstrable at the system level: We evaluate the impact of flexible pipeline routers in the context of an entire multicore system, including the cores, the caches and the interconnection network, and present the system-wide performance and energy-delay^2 (ED^2) product. We find that static energy consumption corresponds to a significant portion of energy consumption. When lowering router frequency, system performance can degrade, and the weight of static energy increases in total energy consumption, leading to the conclusion that VFS is unable to consistently improve system ED^2.

978-1-61284-660-6/11/$26.00 © 2011 IEEE

Related work by Hirata *et al.* [15] presented the idea of variable-pipeline (VP) routers, which aims to achieve similar functionality. However, when the VP router is evaluated using the above two criteria: (1) its hardware overhead is substantial (13%), and (2) it only simulated VP routers with network components under a simplified zero-load latency model and did not provide system-level conclusions.

In the context of improving NoC energy efficiency, this paper makes the following contributions:

- We propose a flexible-pipeline router, where the number of pipeline stages can be dynamically reconfigured. We reduce the number of pipeline stages when scaling down the NoC frequency. The proposed router enables us to scale down the network frequency without increasing router latency.
- We deploy the flexible-pipeline routers in a homogeneous CMP system connected by a mesh network, and use realistic workload to evaluate their performance impact when the network frequency is scaled down to reduce network energy consumption.
- We find that VFS may improve or degrade system ED^2. Therefore, we propose to use simple performance metrics to determine whether VFS should be applied.

The rest of the paper is organized as follows: In Section II, we discuss our pipeline reconfiguration approach for the flexible pipeline. The experimental platform and workload used in this study are described in Section III. We present our experimental results in Section IV followed by concluding remarks in Section V.

II. Flexible router pipeline design

We use a classic four-stage-pipelined virtual-channel router as an example to show how our strategy works. However, our approach can definitely work on other enhanced router designs so long as they have multiple pipeline stages. In this section, we first provide an architectural overview of the classic router. We then introduce our procedure for optimizing the pipeline design to maximize the benefit obtained while speed-tuning the NoC.

A. Baseline Router Architecture

Figure 1 illustrates the microarchitecture of a classic fixed four-stage-pipelined virtual-channel (VC) router with p input/output ports, as used in Garnet [16]. The major components that constitute a router are the input buffers, the route computation logic, the VC allocator, the switch allocator, and the crossbar switch.

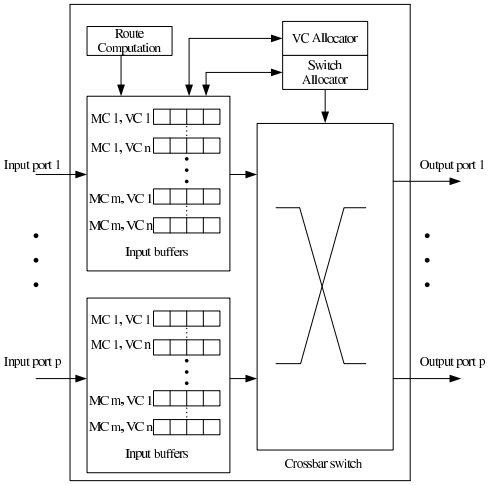


Fig. 1. Classic four-stage virtual-channel router

Figure 2 shows the corresponding fixed four-stage router pipeline. Since on-chip designs must adhere to tight budgets and low router footprints, flit-level buffering and credit-based VC flow control are applied for every router. The router supports multiple message classes (MCs), and VCs from all MCs are multiplexed across the input port. Every VC has its own private flit buffer and its size can be specified at runtime. The routing is table-based and deterministic.

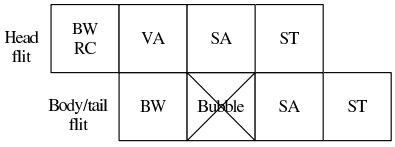


Fig. 2. Router pipeline

When a head flit arrives at an input port, it is first decoded and buffered in the buffer write (BW) pipeline stage, according to its input VC ID. In the same cycle, a request is sent to the route computation unit (RC) simultaneously, and the output port for this packet is calculated based on the destination information presented in each head flit. The header then arbitrates for a free VC of its output port in the VC allocation (VA) stage. Upon successful allocation of an output VC, it proceeds to the switch allocation (SA) stage where it arbitrates for the switch input and output ports. On winning the switch, the flit moves to the switch traversal (ST) stage, where it traverses the crossbar and is placed on the output link connected to the next node. The body and tail flits follow a similar pipeline except that they do not go through RC and VA stages, instead inheriting the VC allocated by the head flit. The tail flit deallocates the VC reserved by the packet when it leaves the router.

B. Flexible-pipeline Router

Although the fine-granularity router pipeline design introduced in Section II-A can run at high frequency, and therefore support high throughput, it may degrade the system performance significantly when the network is slowed down to save power consumption. This is because such a change causes the network latency to increase, which has undesirable effects as discussed in Section I. Therefore, we propose a flexible-pipeline reconfiguration approach to adapt to a change in the network speed, based on accurate delay models for the components in the baseline router introduced in Section II-A.

1) Delay Models of Router Components: We model the delay of each router component shown in Figure 1 by the technology-independent parametric equations presented in [8].

For each component i, we have two delay estimates: *latency* (t_i) and *overhead* (h_i). Defined precisely, the *latency* is the time from when inputs are presented to the component to when the outputs needed by the next component are stable. The *overhead* refers to the setup delay expended by additional circuitry required before the next set of inputs can be presented to the component. As we introduce our timing model, let

- τ be the delay of an inverter with identical input capacitance (at 65nm technology, SPICE simulations show that the value of τ is 7.8ps@1.2V),
- p be the number of input/output ports,
- c be the number of message classes,
- v be the number of VCs per message class, and
- w be the flit size in bits.

The technology-independent parametric delay equations for each router component are listed in Table I [8].

2) Flexible Router Pipeline Reconfiguration: Based on the delay models shown in Table I, we can further determine the optimal number of pipeline stages, for various router sizes and various operating frequencies and supply voltages for the NoC.

We first introduce how the stage delay, T, is calculated for a pipeline stage that includes a few sequential components. Let a and b be the first and last components in the pipeline stage. Given the t_i and h_i of

TABLE I
PARAMETERIZED DELAY EQUATIONS (IN τ) FOR BASELINE ROUTER

Component	Parametric delay equation
BW + RC(BR)	$t_{BR} = 100,\ h_{BR} = 0$
VA	$t_{VA} = 16\frac{1}{2}\log_4 pv + 16\frac{1}{2}\log_4 v + 20\frac{5}{6},\ h_{VA} = 9$
SA	$t_{SA} = 11\frac{1}{2}\log_4 p + 23\log_4 cv + 20\frac{5}{6},\ h_{SA} = 9$
ST	$t_{ST} = 9\log_8(w\lfloor\frac{p}{2}\rfloor) + 6\lceil\log_2 p\rceil + 6,\ h_{ST} = 0$

each component i on the critical path, we have

$$T = \sum_{i=a}^{b} t_i + h_b \tag{1}$$

For example, if we combine the stages SA and ST in the baseline router into a single stage, then the delay for combined stage would be $T = t_{SA} + t_{ST} + h_{ST} = 11\frac{1}{2}\log_4 p + 23\log_4 cv + 9\log_8(w\lfloor\frac{p}{2}\rfloor) + 6\lceil\log_2 p\rceil + 26\frac{5}{6}$ in units of τ.

TABLE II
DELAY VALUES (IN UNITS OF τ) OF EACH ROUTER COMPONENT

Router	BW+RC(BR)		VA		SA		ST	
	t_{BR}	h_{BR}	t_{VA}	h_{VA}	t_{SA}	h_{SA}	t_{ST}	h_{XB}
5-port	100	0	56.5	9	68.7	9	45.0	0
6-port	100	0	58.7	9	70.2	9	46.8	0

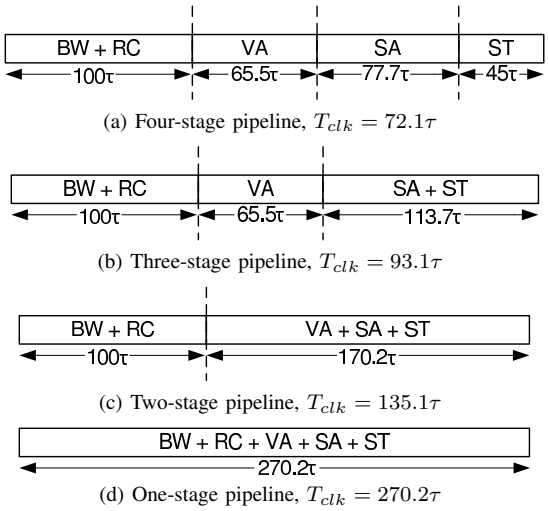
Table II lists the t_i and h_i numbers of each component for the cases of a 5-port router and a 6-port router. From this data we can clearly see that the pipeline stages are imbalanced, and conventional clocking would set the period to correspond to the pipeline stage with the longest delay. Instead, in our work, we apply the time-borrowing techniques to boost the pipeline frequency using clock skew optimization [17], where slower stages in the pipeline borrow time from faster stages, such that the linear router pipeline can operate at the average cycle time of all the pipeline stages. Let T_i be the delay for pipeline stage i and let n be the total number of stages. Then the clock time for the n-stage router after time-borrowing is

$$T_{clk} = \frac{\sum_{i=1}^{n} T_i}{n} \tag{2}$$

Furthermore, this four-stage linear router can be reconfigured as either a three-stage, or a two-stage, or even a one-stage pipeline. For example, a three-stage pipeline can be obtained by combined any of the two successive components such as 1) BR and VA, or 2) VA and SA, or 3) SA and ST. Of these, we choose the one with the minimum optimized clock period T that maximizes the router frequency that the pipeline can support: note that since the h_i values for various stages are different, the optimal frequency varies with our choice. The best choices are found to be: 1) for the three-stage pipeline, we combine SA and ST into one single stage and 2) for the two-stage pipeline, we combine VA, SA and ST into one single stage.

Figure 3 shows our final pipeline reconfiguration results for a 5-port router, and Table III summarize the results for two different router sizes after applying time-borrowing techniques. For each pipeline design case, we list the optimal clock time T_{clk} in units of τ, and the maximum frequency F that the pipeline can support in the GHz range. To achieve energy saving, we also apply voltage scaling as we scale down the frequency and the following V_{dd} values are used: 1.2V for four-stage pipeline, 1.1V for three-stage pipeline, 1.0V for two-stage pipeline and 0.8V for one-stage pipeline. Considering that the circuit delay is a function of the supply voltage V_{dd}, we scale τ accordingly based on Alpha-power law [18] for the above voltage settings.

Given the maximum frequency numbers for each pipeline design, it is straightforward to select the optimal pipeline design at a given network speed. The basic principle behind this idea is that at the same network speed, a shorter pipeline with fewer stages leads to lower router latency, and therefore, better network performance. For example, for the 5-port router, if its frequency is set to be 1.5GHz, then only the four-stage pipeline with a peak frequency of 1.78GHz can be fast enough. However, if we reduce the router frequency to 1.0GHz, then both four-stage pipeline and three-stage pipeline can meet the frequency demand, but we will choose three-stage pipeline to maximize the router performance. Table IV presents the results of optimal pipeline stage number $N = 1, 2, 3, 4$ for different routers with different router to processor clock ratio S (explained in Section III-A).



(a) Four-stage pipeline, $T_{clk} = 72.1\tau$

(b) Three-stage pipeline, $T_{clk} = 93.1\tau$

(c) Two-stage pipeline, $T_{clk} = 135.1\tau$

(d) One-stage pipeline, $T_{clk} = 270.2\tau$

Fig. 3. Optimal pipeline reconfiguration for a 5-port router, time borrowing technique is applied to boost the pipeline frequency.

TABLE III
OPTIMAL CLOCK PERIODS/FREQUENCIES FOR VARIOUS PIPELINE CONFIGURATIONS

Router	4-stage		3-stage		2-stage		1-stage	
	T_{clk}	Max. F	T_{clk}	Max. F	T_{clk}	Max. F	T_{clk}	Max. F
5-port	72.05	1.78	93.07	1.32	135.10	0.87	270.2	0.39
6-port	73.43	1.75	94.90	1.30	137.85	0.85	275.7	0.38

TABLE IV
THE OPTIMAL NUMBER, N, OF PIPELINE STAGES WITH DIFFERENT PROCESSOR TO ROUTER CLOCK RATIO S; THE PROCESSOR FREQUENCY IS 1.5 GHZ.

Router	S = 1	S = 2	S = 3	S = 4	S = 5
5-port	4	2	2	1	1
6-port	4	2	2	1	1

3) Architecture Support for Flexible Pipeline Reconfiguration: The hardware support required for out flexible-pipeline router is minimal, and only needs a set of multiplexers that bypass registers and alter clock skews. In Figure 4 we only show the multiplexers used to bypass the registers after each pipeline stage. Given a particular pipeline configuration, we set the stage selection signals to combine the successive stages correspondingly. The delay of the 2:1 multiplexers used in the flexible-pipeline router is just several τs, therefore their impact on the pipeline clock period is negligible for all the pipeline configurations. For the area cost, after analyzing the gate count of each component in the flexible-pipeline router, we find that the router area is dominated by the input buffers and VC allocators, and the multiplexers only account for less than 2% hardware overhead in terms of the router area.

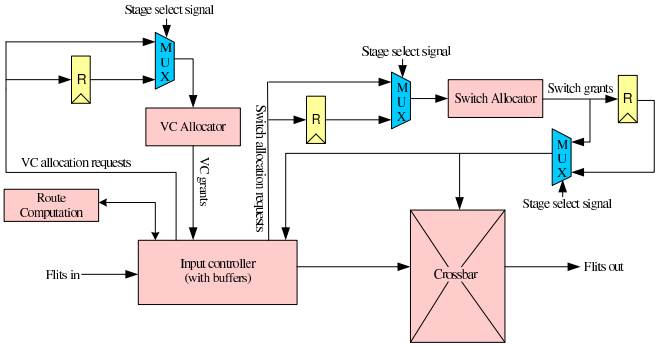


Fig. 4. Router architecture for flexible pipeline reconfiguration

III. Experimental Platform

A. CMP System Simulator

We use Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) [19] as our simulation engine. GEMS is a Simics-based [20] full system simulator using timing-first simulation approach. This infrastrcture provides detailed performance and power models for the processor pipeline, the memory hierchy, as well as the NoC. The NoC simulator provides the flexibility for customizing the interconnection by providing support for various network topologies, including user-specified ones. The NoC power model is provided by Orion 2.0 [21], which reports the total switch and link power. The NoC power breaks down into three segments: dynamic, static (leakage) and clock power.

In this work, we simulate an 8-core CMP system with 65nm technology nodes, where the cores/L2 cache banks and memory controllers are connected using a mesh network. Architecture parameters can be found in Table V. The interconnection network uses a deterministic shortest path routing algorithm. For the processor power, we employ Wattch [22], an architectural level power modeling tool. We have updated the technology-specific parameters in Wattch based on the ORION 2.0 technology file. Simulation results show that the power of floating function units (FUs) dominates the total FU power. Therefore, we use power gating and clock gating techniques to disable the inactive floating FUs in the CMP system.

TABLE V
Baseline Simulation Configuration

Processor Core	1.5GHz, one-way in-order 3 integer FUs, 6 floating FUs
Private L1 Cache	Split private I/D caches, each 2KB, 2-way set associative, 64B block size, 1-cycle access latency
Shared L2 Cache	4M banked, shared distributed, 512KB (per core) 4-way set associative, 64B block size, 8-cycle access latency
Memory	4GB DRAM, 200 cycle access latency, four memory controllers (one in each corner node)
Router	5 Input/output ports, 4-stage Pipeline, 5 message classes, 2 VCs per class, 64-bit flits, 4-flit buffer depth, 1 flit per control packet, 9 flits per data packet, wormhole routing

In our baseline, the processors and NoC operate at the same frequency. Upon frequency scaling, we introduce a slow down parameter S, which is the router-to-processor clock ratio. For example, $S = 2$ implies that one network clock cycle is equivalent to two processor cycles. In our experiment, we experiment with three integer values, S = 1, 2, and 4. We found that, the link delays are small enough that for any of the S values used here, a flit can traverse a link within one network cycle.

B. Workloads

We choose applications from the SPEC OMP2001 [23], NU-Mine [24] and PARSEC [25] benchmark suites as our workload input. In total we evaluate 6 correctly-compiled benchmarks written in C/C++. An overview of these applications are shown in Table VI.

TABLE VI
Benchmark Descriptions

ammp	Computational Chemistry
art	Neural network simulation; adaptive resonance theory
blackscholes	Computational finance
equake	Finite element simulation; earthquake modeling
fkmeans	Fuzzy-logic based data partitioning
kmeans	Mean based data partitioning

In accordance with common practice in the architecture community when working with such benchmarks, in order to reduce simulation time, for each benchmark we first fast-forward to the beginning of the region of interest (the parallel section representative of the whole application) in Simics without loading Ruby; after that, we load Ruby and simulate one billion instructions.

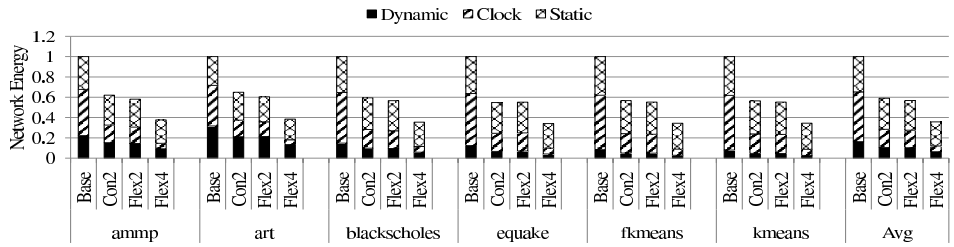
IV. Experimental Results

In this section, we evaluate a CMP, as described in Section III-A, with the proposed flexible-pipeline routers, and compare its performance and energy consumption with a system that uses fixed-pipeline routers. Figure 5 shows the performance and energy results, with routers scaled with different slow down factors ($S = 2, 4$). For the reasons specified in Section I, we limit the choice of S to integer values. The results are normalized to the case where the NoC is unscaled i.e., S = 1.

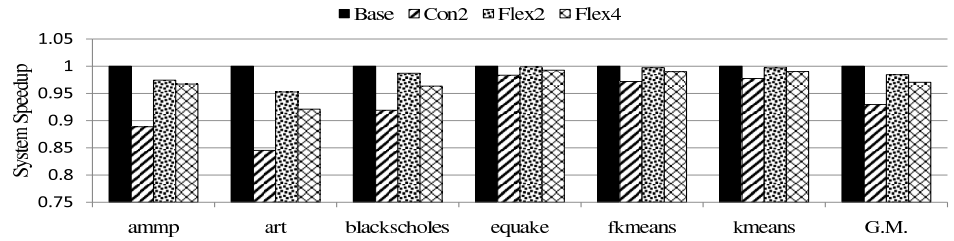
In general, reducing the network frequency can reduce energy consumption (Figure 5(a)), but it can also increase network latency and degrade network throughput. Such impacts directly translate into system performance degradation, as shown in Figure 5(b). The first set of bars in Figure 5(b) corresponds to the unscaled NoC (bars *Base*), the next two sets of bars correspond to CMP systems with fixed-pipeline routers (bars *Con2*) and flexible-pipeline routers (bars *Flex2*) when the network frequency is scaled by 50% and voltage is scaled from 1.2V to 1.0V.

For fixed-pipeline routers (bars *Con2*), on average, energy consumption is reduced by 41%, but system performance is degraded by 7%. The reduction in energy consumption is mainly due to reduction in clock energy and dynamic energy. In particularly, clock energy reduction due to frequency scaling is significant. This is because clock energy is proportional to the product of V_{dd}^2 and number of clock transitions. For example, when the network voltage is reduced from 1.2V to 1.0V and the network frequency is slowed down by a factor of 2, it corresponds to a 65% reduction of clock energy. Dynamic energy is proportional to V_{dd}^2 and the amount of work performed by the NoC. Since the data path that each packet traverses is the same and the number of packets is similar before and after scaling; the amount of work performed by the NoC is not significantly changed due to scaling. As a result, dynamic energy of the network decreases quadratically as network voltage goes down. Static energy of the network reduces linearly with network voltage but increases linearly with total execution time. In our experiment, changes in network static energy are small compared to clock and dynamic energy.

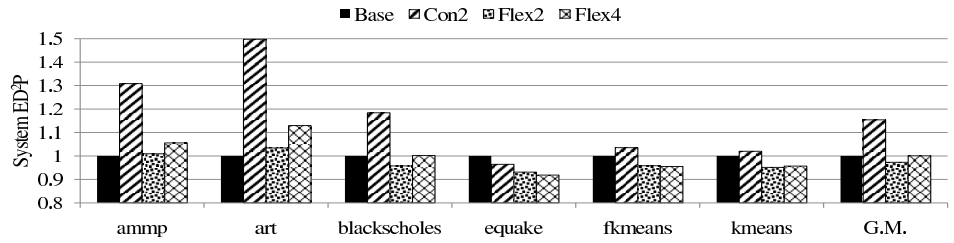
When the flexible-pipeline router is deployed, with the same voltage/frequency scaling, the average system performance degradation is only 1.6%, and the average network energy consumption is reduced by 43% (bars *Flex2*). This corresponds to a 2% additional reduction from the fixed pipeline case. This small reduction in energy is a side effect of performance improvement: by completing the application faster, we



(a) Network energy. *Dynamic* corresponds to the total dynamic energy consumed by both routers and network links; segment *Clock* corresponds to dynamic energy for distributing the clock; *Static* corresponds to the total static (leakage) energy consumed.



(b) System performance



(c) ED^2 of the system

Fig. 5. Comparison of fixed-pipeline and flexible-pipeline routers. *Base* corresponds to no scaling and using fixed-pipeline routers. *Con2* corresponds to network frequency scaled down by a factor of two, and using fixed-pipeline routers. *Flex2* and *Flex4* corresponds to network using flexible-pipeline router and frequency scaled down by a factor of two and four, respectively. All results are normalized to *Base*.

are able to reduce static energy. Overall, NoCs with flexible-pipeline routers are more energy efficient.

TABLE VII
CACHE MISS RATES FOR EVALUATED WORKLOADS

Workload	L1 data cache (misses/K instructions)	L2 cache (misses/K instructions)
ammp	13.7	4.4
art	40.8	18.1
blackscholes	8.1	0.9
equake	2.8	2.6
fkmeans	1.9	1.7
kmeans	2.4	1.9

The performance impact of network voltage and frequency scaling varies with the workload. Table VII lists the cache miss rates for all the workloads. Some applications, such as *ammp* and *art* suffer from a large number of L1 cache misses, and thus are very sensitive to L2 cache response time through the network. For these applications, network frequency scaling can degrade performance in two ways. On the one hand, frequency scaling leads to increases in network latency and L2 cache response times, which in turn increases the number of cycles per instruction (CPI). On the other hand, network frequency scaling decreases the throughput capability of the network, and thus causes more contentions. Although flexible-pipeline routers are able to effectively avoid increasing network latency in the absence of network contention, they see some level of throughput degradation due to contention. Thus, applications with high throughput requirements are likely to suffer performance degradation with frequency scaling. For other applications, when frequency is scaled down, deploying flexible-pipeline routers allows us to reduce energy consumption without suffering appreciable performance degradation. Dynamically determining the optimal operation frequency of the NoC is desirable [14], but is a topic for future work.

If the network contention is low, flexible-pipeline routers make it possible to scale down network frequency without increasing the latency. However, this technique cannot prevent degradation in the network throughput, and thus frequency scaling can still lead to performance degradation. Bars 1, 3 and 4 of Figures 5(a) and 5(b) show the performance and network energy consumption as a result of voltage and frequency scaling when the network is scaled by factors of two and four, respectively, again using the no scaling case as a reference. It is clear that scaling reduces both the performance and the energy. When network is scaled by a factor of four, all workloads suffer noticeable performance degradation. However, the energy savings in going from a scaling factor of two to four shows diminishing returns as compared to the case where we go from scaling factor one to two. This is primarily related to the fact that applications take longer to complete, and the NoC hardware is activated for a longer duration, during which static (leakage) energy is expended. This increase in static energy offsets the gains made in reducing the dynamic network energy.

Up to this point, we have only considered the energy consumption of the network. However, whether VFS is beneficial must be considered in the context of the entire system, including the cores, the caches and

the interconnection network. Figure 5(c) shows the ED^2 of the entire system. In comparison to flexible-pipeline routers, system with fixed-pipeline routers has higher ED^2 due to significant system performance degradation. We also find that for applications with a high cache miss rate, such as *ammp* and *art*, frequency scaling degrades ED^2 even with flexible-pipeline routers. This is because network throughput degrades with frequency scaling, even when flexible-pipeline router is used; and the above applications have high throughput requirement. As a result, these applications suffer from performance and ED^2 degradation upon frequency scaling. Furthermore, extended execution time also leads to more static energy consumption, which in turn further degrades ED^2.

The decision of scaling down network frequency should be specific to each application. In our case, the four out of six benchmarks benefit from frequency scaling. For some applications, such as *blackscholes*, aggressive frequency scaling adversely degrades ED^2. This is because for *blackscholes*, when the scaling factor is increased from 2 to 4, the decrease of system energy (0.4%) cannot offset the increase of CPI (2.4%). As a result, in some cases it is undesirable to aggressively scale down network frequency. It is worth pointing out that cache miss rate is only one metric for determining optimal router frequency. Other performance metrics, such as router utilization, can also serve as good indicators.

V. Conclusion

This paper proposes flexible-pipeline routers that are capable of re-balancing the pipeline stages upon voltage and frequency scaling, while operating the cores at the original frequency. The hardware complexity for supporting pipeline rebalancing is minimal. We compare 8-core mesh-based CMP systems with fixed- and flexible-pipeline routers running realistic workloads, respectively. When frequency is scaled down, energy reduction is dramatic for both systems, while the performance degradation can be low to medium. We compute the system ED^2 and show that for some benchmarks this value is noticeably improved with frequency scaling, while for others, it degrades. The improvement is application-specific and we see a close relationship with the cache miss count. However, for systems with fixed-pipeline routers, both throughput and latency degrades; while for systems with flexible-pipeline routers, latency remains unchanged as throughput degrades. Thus, as long as the network throughput is low, frequency-scaling on systems with flexible-pipeline routers are able to achieve energy saving with minimal performance degradation; and thus are more energy-efficient.

The proposed routers are able to improve energy-efficiency of the system by exploiting the fact that certain workloads are latency-sensitive, but are not throughput-intensive. The proposed technique can work in tandem with (and are largely orthogonal to) other techniques that are intended to reduce router power, such as router bypassing [5]–[7], router pipeline bypassing [10], speculative switch arbitration [8], as well as various congestion management strategies [14]. For routers that support bypassing and speculative arbitration, it is still possible to re-balance the pipeline stages, but we must take care to only re-balance the non-bypassed pipeline stages. Existing congestion control mechanisms can work in NoC with flexible-pipeline routers, although network bottlenecks may shift as a result of frequency scaling.

In this paper, we only demonstrate the benefit of flexible-pipeline routers through static frequency scaling. Being able to dynamically adjust router frequency and balance router pipeline allows the system to adapt to dynamic behaviors of the programs. Such techniques will be explored in our future work.

Acknowledgment

This work is supported in part by grants from National Science Foundation under CNS-0834599, CSR-0834599, CPS-0931931, and CCF-0903427, a contract from Semiconductor Research Cooperation under SRC-2008-TJ-1819. We would like to thank all anonymous reviewers for their constructive comments that help improve the quality of this paper.

References

[1] H. Wang, L.-S. Peh, and S. Malik, "Power-driven Design of Router Microarchitectures in On-chip Networks," in *Proc. Int. Symp. Microarchitecture*, 2003, pp. 105–116.

[2] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar, "An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS," *J. Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, 2008.

[3] R. Kumar, D. M. Tullsen, N. P. Jouppi, and P. Ranganathan, "Heterogeneous Chip Multiprocessors," *Computer*, vol. 38, pp. 32–38, 2005.

[4] F. Bower, D. Sorin, and L. Cox, "The Impact of Dynamically Heterogeneous Multicore Processors on Thread Scheduling," *IEEE Micro*, vol. 28, no. 3, pp. 17–25, May. 2008.

[5] U. Y. Ogras and R. Marculescu, "It's a small world after all: NoC Performance Optimization via Long-range Link Insertion," *IEEE Trans. VLSI Systems*, vol. 14, no. 7, pp. 693–706, 2006.

[6] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express Virtual Channels: Towards the Ideal Interconnection Fabric," in *Proc. Int. Symp. Computer Architecture*, 2007, pp. 150–161.

[7] M. F. Chang, J. Cong, A. Kaplan, M. Naik, G. Reinman, E. Socher, and S.-W. Tam, "CMP Network-on-chip Overlaid with Multi-band RF-interconnect," in *Proc. Int. Symp. High-Performance Computer Architecture*, 2008, pp. 191–202.

[8] L.-S. Peh and W. J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," in *Proc. Int. Symp. High-Performance Computer Architecture*, 2001, pp. 255–266.

[9] R. Mullins, A. West, and S. Moore, "The Design and Implementation of a Low-Latency On-Chip Network," in *Proc. Asia & South Pacific Design Automation Conf.*, 2006, pp. 164–169.

[10] A. Kumar, P. Kundu, A. P. Singh, L.-S. Peh, and N. K. Jha, "A 4.6Tbits/s 3.6GHz Single-cycle NoC Router with a Novel Switch Allocator," in *Proc. Int. Conf. Computer Design*, 2007, pp. 63–70.

[11] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks," in *Proc. Int. Symp. High-Performance Computer Architecture*, 2003, pp. 91–102.

[12] E. J. Kim, K. H. Yum, G. M. Link, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, M. Yousif, and C. R. Das, "Energy Optimization Techniques in Cluster Interconnects," in *Proc. Int. Symp. Low Power Electronics & Design*, 2003, pp. 459–464.

[13] S. E. Lee and N. Bagherzadeh, "A Variable Frequency Link for a Power-aware Network-on-chip (NoC)," *Integration, the VLSI Journal*, vol. 42, no. 4, pp. 479–485, 2009.

[14] A. K. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. R. Das, "A Case for Dynamic Frequency Tuning in On-chip Networks," in *Proc. Int. Symp. Microarchitecture*, 2009, pp. 292–303.

[15] Y. Hirata, H. Matsutani, M. Koibuchi, and H. Amano, "A Variable-pipeline On-chip Router Optimized to Traffic Pattern," in *Int. Workshop on Network on Chip Architectures*, 2010, pp. 57–62.

[16] N. Agarwal, L.-S. Peh, and N. Jha, "Garnet: A Detailed Interconnection Network Model inside a Full-system Simulation Framework," Princeton University, Tech. Rep. CE-P08-001, 2008. [Online]. Available: http://www.princeton.edu/~niketa/garnet

[17] S. Sapatnekar, *Timing*. Boston, MA: Kluwer Academic Publishers, 2004.

[18] T. Sakurai and A. R. Newton, "Alpha-power Law MOSFET Model and Its Applications to CMOS Inverter Delay and Other Formulas," *J. Solid-State Circuits*, vol. 25, no. 2, pp. 584–594, 1990.

[19] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset," *SIGARCH Computer Architecture News*, vol. 33, pp. 92–99, 2005.

[20] P. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A Full System Simulation Platform," *Computer*, vol. 35, no. 2, pp. 50–58, Feb 2002.

[21] A. Kahng, K. Samadi, B. Li, and L.-S. Peh, "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration," in *Proc. Design, Automation & Test in Europe Conf.*, 2009, pp. 423–428.

[22] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," in *Proc. Int. Symp. Computer Architecture*, 2000, pp. 83–94.

[23] "SPEC OMP2001," Available at http://www.spec.org/omp/.

[24] R. Narayanan, B. Ozisikyilmaz, J. Zambreno, J. Pisharath, G. Memik, and A. Choudhary, "MineBench: A Benchmark Suite for Data Mining Workloads," in *IEEE Int. Symp. on Workload Characterization*, 2006, pp. 182–188.

[25] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *Proc. Int. Conf. Parallel Architectures and Compilation Techniques*, 2008, pp. 72–81.