Optimizing the NoC Slack Through Voltage and Frequency Scaling in Hard Real-Time Embedded Systems

Jia Zhan, *Student Member, IEEE,* Nikolay Stoimenov, *Member, IEEE,* Jin Ouyang, Lothar Thiele, *Member, IEEE,* Vijaykrishnan Narayanan, *Fellow, IEEE,* and Yuan Xie, *Senior Member, IEEE*

***Abstract*—Hard real-time embedded systems impose a strict latency requirement on interconnection subsystems. In the case of network-on-chip (NoC), this means each packet of a traffic stream has to be delivered within a time interval. In addition, with the increasing complexity of NoC, it consumes a significant portion of total chip power, which boosts the power footprint of such chips. In this paper, we propose a methodology to minimize the energy consumption of NoC without violating the prespecified latency deadlines of real-time applications. First, we develop a formal approach based on network calculus to obtain the worst-case delay bound of all packets, from which we derive a safe estimate of the number of cycles that a packet can be further delayed in the network without violating its deadline—the worst-case slack. With this information, we then develop an optimization algorithm that trades the slacks for lower NoC energy. Our algorithm recognizes the distribution of slacks for different traffic streams, and assigns different voltages and frequencies to different routers to achieve NoC energy-efficiency, while meeting the deadlines for all packets. Furthermore, we design a feedback-control strategy to enable dynamic frequency and voltage scaling on the network routers in conjunction with the energy optimization algorithm. It can flexibly improve the energy-efficiency of the overall network in response to sporadic traffic patterns at runtime.**

***Index Terms*—Dynamic voltage and frequency scaling (DVFS), network calculus, network-on-chip (NoC), slack, worst-case delay analysis.**

I. Introduction

CONTEMPORARY embedded systems and SoCs feature an increasing number of processing elements (PE) and other components, a sign that interconnection will play a more vital role in these chips. Network-on-chip (NoC) is a promising design paradigm for future many-core chips as found by many previous researches [1], [7]. However, the fundamental challenge of using NoCs in many-core embedded systems is that these systems often have very limited resources and stringent processing latency requirements, which places very different constraints than general-purpose processors on NoC design. There are two major differences between embedded systems and general-purpose processors.

1) General-purpose processors are often designed to achieve a high aggregate throughput, and therefore the NoCs for them are allocated sufficient resources to sustain the peak performance. In contrast, embedded systems are designed to provide just enough performance to accommodate specific tasks. Thrift is a virtue in designing NoC for those systems, in order for power and area reduction.
2) General-purpose processors care about the overall progress of all tasks running on all cores. In contrast, embedded systems often provide certain guarantees for individual tasks' progress. In the so-called hard real-time embedded systems, to provide certain quality-of-service (QoS), each task has an associated maximum allowed communication delay. Reflected on NoC, each network packet needs to be delivered to the destination before a deadline; otherwise the corresponding task may not be able to deliver the required QoS, and even causes catastrophic outcomes.

One way to address the conflicting requirements of energy and latency is to leverage the inherent heterogeneity in NoC traffics, and use voltage frequency scaling (VFS) to improve the energy-efficiency of NoC. A lot of previous work [23], [34], [42] have adopted dynamic voltage and frequency scaling (DVFS) to reduce the energy consumption of NoC while still providing high throughput. Heterogeneity can also be utilized to improve the efficiency of NoC. Das *et al.* [9] were the first to propose the idea of network slack, which refers to the number of cycles that a packet can be delayed in the network without affecting execution time. In their work, packets with smaller slacks (those more likely to impact execution time) are prioritized. This slack-based approach improves the throughput of all running tasks. However, the above researches are still focused on designing NoC for general-purpose processors, and aimed at improving the overall throughput. For example, the estimated slack proposed by Das *et al.* [9] does not consider precise deadlines on individual packets and only serves as a hint in assigning priorities

Manuscript received January 12, 2014; revised June 11, 2014; accepted July 8, 2014. Date of current version October 16, 2014. This paper was recommended by Associate Editor N. Wong.

J. Zhan and V. Narayanan are with the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16803 USA (e-mail: juz145@cse.psu.edu; vijay@cse.psu.edu).

Y. Xie is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA (e-mail: yuanxie@ece.ucsb.edu).

N. Stoimenov and L. Thiele are with the Computer Engineering and Networks Laboratory, ETH Zurich, Zurich 8092, Switzerland (e-mail: stoimenov@tik.ee.ethz.ch; thiele@tik.ee.ethz.ch).

J. Ouyang is with NVIDIA, Santa Clara, CA 95050 USA (e-mail: jouyang@nvidia.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2014.2347921

to packets. There is no guarantee that a packet will arrive in time before it is needed. Therefore, these approaches cannot be applied to NoC in embedded systems where violating deadlines could be disastrous.

Unlike previous work, we focus on improving NoC energy-efficiency in hard real-time embedded systems by leveraging the heterogeneity of network routers to serve NoC traffics. We propose a design methodology that provides just enough power to NoC in order to meet the latency requirements (deadlines) of all traffic streams. Inspired by Das *et al.*'s work [9], we first calculate the worst-case slacks for packets of different streams. Then an energy optimization algorithm is proposed that leverages the slacks to allocate differentiated resources (energy) to different portions of the network. Different from their work, the slack calculation must be precise and conservative in order to guarantee the timing-correctness of real-time systems. To solve this problem, we adopt network calculus [3], [20] to predict the worst-case latency of different packets, from which the worst-case slacks can be obtained. Leveraging these slacks, we can progressively reduce the voltages and frequencies of individual routers in the network to reduce energy consumption while still meeting all deadlines. To this end, our framework is able to compute the optimal voltage and frequency configurations given a fixed traffic pattern. However, during runtime, there may be changes of network traffic that would possibly cause deadlines misses. These changes include phase transitions in a single application or arrival/departure of new/old applications. Therefore, we design a feedback-control system to monitor the traffic changes and dynamically adapt the voltage/frequency levels of individual routers for the best power efficiency at all time. To summarize, the contributions of this paper are as follows.

1) We develop a formal method based on network calculus to obtain the worst-case slacks of packets in the NoC for hard real-time embedded systems. We improve over previous work [30] by taking virtual channels and heterogeneous router frequencies into consideration.
2) We propose an effective algorithm that trades slacks for energy-efficiency of NoC, and thus minimizes the total communication energy while still maintaining timing-correctness. This algorithm adjusts energy and performance by assigning the optimum voltage and frequency configurations to individual routers in the network.
3) We propose a feedback-control strategy to monitor real-time performance of network flows and conduct DVFS on individual routers to provide the optimal network power efficiency during runtime.

The remainder of this paper is organized as follows. Section II reviews the related work. Section III describes how to derive network slacks through worst-case delay analysis. Leveraging this analysis approach, Section IV proposes several energy optimization algorithms that trade slacks for energy savings. Section V proposes dynamical voltage and frequency scaling based on the proposed static optimization schemes. Experiments are conducted in Section VI. Finally, Section VII concludes the paper.

II. Related Work

A. Voltage and Frequency Scaling

Network-on-Chip consumes a significant portion of total chip power for multicore systems. Some recent research prototypes [15], [22], [38] have validated the percentage to be 10%–36%. Therefore, NoC power reduction becomes a critical issue in designing multicore systems, especially in the dark silicon age [11], [14], [32], [37], [39] when power budget is the primary design constraint. One way to tackle the power concern is through DVFS. Several studies have proposed DVFS for general purpose systems [18], [33] and real-time embedded systems [40], [41]. However, these results focus on either processor or cache power optimization. Recently, DVFS on NoCs are proposed to further reduce the interconnect power dissipation. Some prior work proposes similar techniques on NoCs [5], [16], [23], [34], [42] by scaling the voltage/frequency of individual routers, links, or the whole network. In the mean time, there are some prior studies [24], [26], [27] on static voltage and frequency assignment for NoC components through voltage island partitioning and optimization. However, these results are still focused on the general-purpose domain, whereas in hard real-time systems the requirement for NoC DVFS becomes more challenging.

B. Worst-Case Analysis of NoC

There are some prior works [21], [28] on providing QoS guarantees in NoC design for general purpose systems. In hard real-time systems, each network packet in a certain flow needs to be delivered to the destination within a fixed deadline. Such deadlines cannot be violated, otherwise the QoS would be severely degraded. Therefore, in order to safely conduct DVFS on NoC for such systems, an accurate prediction of the worst-case packet delay should be provided. In wormhole networks, this becomes more complicated due to the existence of credit-based flow control as well as the network contention for shared resources. Network calculus [3], [20] is a theory of deterministic queuing systems for communication networks. Some recent work [29], [31] has validated the effectiveness of using network calculus for worst-case delay prediction on NoC-based systems, although the state-of-the-art virtual-channel based NoC is not modeled. In this paper, we adopt network calculus in modeling VC-based NoC systems and integrate it with our energy optimization techniques.

III. Worst-Case Delay Analysis

A. Router Architecture

Most of the state-of-the-art NoC researches assume a baseline wormhole router that achieves high energy-efficiency [7]. To provide guaranteed services in NoC, researchers extended the baseline router architecture to either preallocate switching time slots for critical packets [12], [13], or preserve a virtual channel for each traffic stream [2], [35]. Preallocating switching time slots eliminates run-time contentions altogether, while preserving virtual channels only prevents head-of-line blocking and still needs proper arbitration schemes for performance guarantee.

TABLE I
SYMBOLS USED FOR MODELING AND ANALYSIS

Symbols	Description
α	arrival curve
β	service curve
β^{R_i}	overall service curve of router R_i
$\beta^{R_i'}$	ideal service curve of router R_i without back-pressure
$A[s,t)$	the number of packets that arrive during $[s,t)$
$C[s,t]$	the number of packets that can be processed during $[s,t]$
d_{worst}	worst-case packet delay
s_i	length of a slot assigned to flow i in the scheduling model
B	VC buffer size
η	router frequency scaling factor
D_i	deadline constraint for flow i
$\otimes$	min-plus convolution e.g. $a \otimes b = \inf_{0\le s\le t}\{a(s)+b(t-s)\}$
$\wedge$	minimum e.g. $a \wedge b = min\{a,b\}$
δ_T	burst delay function $\delta_T = +\infty$ if $t > T$, else 0
$\gamma_{r,b}$	affine arrival curve $\gamma_{r,b}(t) = rt + b$ if $t > 0$, else 0
$\beta_{\lambda,T}$	rate-latency function $\lambda[t-T]^+ = \lambda(t-T)$ if $t > T$, else 0
$\overline{f}$	sub-additive closure $\overline{f} = \delta_0 \wedge f \wedge (f \otimes f) \wedge (f \otimes f \otimes f) \wedge \ldots$

In this paper, we assume a baseline wormhole router architecture with five router stages: 1) BW: buffer write; 2) RC: routing computation; 3) VA: virtual channel allocation; 4) SA: switch allocation; and 5) ST: switch traversal. Recently NoC router architectures with fewer stages were also proposed. However, changing the number of router stages affects only the initial latency in our analysis (refer to details below), and therefore our approach can be applied to router architectures with fewer pipeline stages. In addition, we assume that each traffic stream uses a dedicated virtual channel throughout the network, which is in line with the designs proposed by [2] and [35]. We do not opt for preallocating time-slots for each packet [12], [13], because this approach eliminates the flexibility of scaling voltage and frequency to reduce energy consumption.

Consequently, there are two types of pipeline stalls to be addressed for this kind of router.

1) With credit-based flow control, packet will stall when there are no available credits which indicates the downstream VC that they are writing into is full.
2) Packet stall may happen in SA stage due to switch contention, when packets from different input ports or different VCs at the same input port may not only compete for the same switch input port but also the same output port.

In this section, the detailed analytic models for the two types of router pipeline stalls are presented, and the worst-case packet delay bound is derived from these models. Table I summarizes symbols used in our modeling and analysis.

B. Principles of Network Calculus

Network calculus [20] is a theory of deterministic queuing systems for communication networks. In particular, this approach is based on three important concepts.

1) Arrival Curve: If $A[s,t)$ denotes the number of packets (here, we define a packet as a fixed-length basic unit in network traffics; variable-length packets can be viewed as a sequence of fixed-length packets) that arrive in the time interval $[s,t)$, then we say the flow A is constrained by an arrival curve α if and only if for all $s < t$

$$A[s,t) \le \alpha(t-s). \tag{1}$$

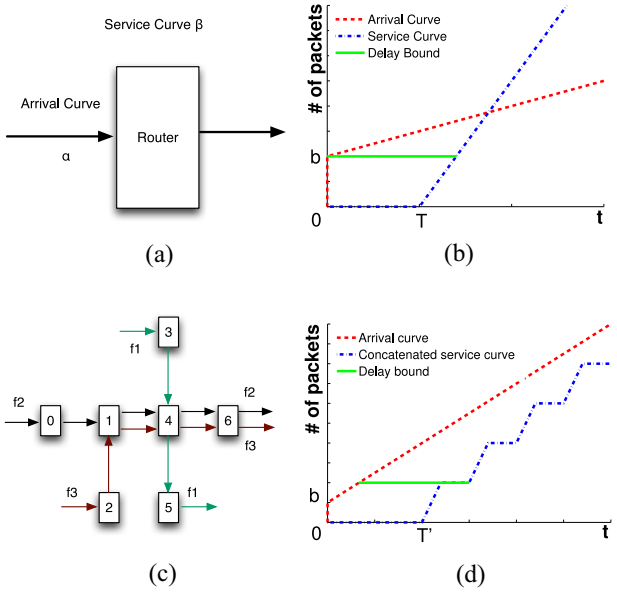


Fig. 1. Delay bound from network calculus. (a) Single router. (b) Delay bound for a router. (c) Multiple flows. (d) Delay bound for a flow.

2) Service Curve: If $C[s,t)$ denotes the number of packets that can be processed by a router or a whole network over the time interval $[s,t)$, and C is bounded by a service curve β if and only if for all $s < t$

$$C[s,t) \ge \beta(t-s). \tag{2}$$

3) Delay Bound: Assume a packet stream, constrained by an arrival curve α, traverse a system that offers a service curve β. Then the worst-case packet delay d_{worst} can be bounded as

$$d_{\text{worst}} \le \sup_{t\ge 0}\{\inf\{\tau \ge 0 : \alpha(t) \le \beta(t+\tau)\}\} \tag{3}$$

where inf{} and sup{} stand for infimum and supremum, respectively. An example is shown in Fig. 1(a) and (b) for a single router, where we show an affine arrival curve $\alpha_{r,b}$, defined by: $\alpha_{r,b}(t) = rt + b$ for $t > 0$, and $\alpha_{r,b} = 0$ otherwise, and a rate-latency service curve $\beta_{\lambda,T}$, defined by: $\beta_{\lambda,T}(t) = \lambda[t-T]^+ = \lambda(t-T)$ for $t > T$, and $\beta_{\lambda,T} = 0$ otherwise. The arrival curve $\alpha_{r,b}$ implies that the source can send at most b packets at once, but no more than r packets/cycle in the long run, while the service curve $\beta_{\lambda,T}$ implies a pipeline delay T for a packet to traverse a router and an average service rate of λ packets/cycle. As shown in Fig. 1(b), the worst-case delay bound d is the maximum horizontal distance between arrival curve and service curve.

When extended to multiple interconnected components, as shown in Fig. 1(c), the end-to-end packet delay becomes more unpredictable. Fig. 1(d) is the worst-case delay analysis for one flow f_2. As we can see, arrival curve α remains as the given injection pattern, while service curve is now the concatenation of all the routers that f_2 traverses from source to destination, which can be calculated through server concatenation [20]. For instance, the concatenation of two routers with service curves β^{R_1} and β^{R_2} is

$$\beta^{R\{1,2\}} = \beta^{R_1} \otimes \beta^{R_2} = \inf_{0\le s\le t}\{\beta^{R_1}(s) + \beta^{R_2}(t-s)\}. \tag{4}$$

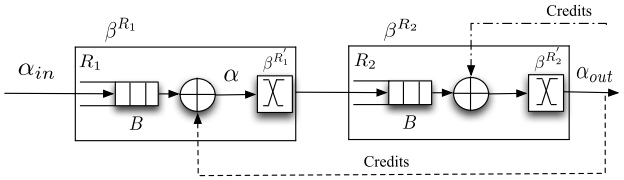


Fig. 2. Analysis of credit-based flow control.

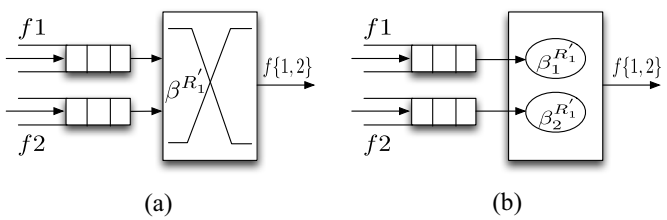


Fig. 3. Analysis of switch contention. (a) Two flows competing for the same. (b) Partial service curve for each flow switch input.

So far we only consider the case where routers with infinite buffers provide service to a single flow. In reality, the router is designed with a finite buffer size that exerts back-pressure, and many flows may share routers in NoC experiencing reduced service quality. Both factors introduce additional stalls (flow-control stall and switch-contention stall). In the rest of this section, we consider the additional stalls resulted from back-pressure and resource sharing in the worst-case delay analysis.

C. Flow-Control Stall

With credit-based flow control [8], the upstream router keeps a count of the number of free buffers in each virtual channel downstream. No packets will be forwarded if their intended buffers are full, until the downstream buffer forwards a packet and sends a credit back to the upstream router. Here we adapt Chang's work [4] to derive the worst-case latency bound under the back-pressure of credit-based flow control. For simplicity, we consider two adjacent routers R_1 and R_2 in our demonstration, and the results can be easily applied to the case when more routers are involved. Fig. 2 shows a graphical view of the two-router case.

Let α_{in} be the generic input process to router R_1 while α be the effective input process to the internal crossbar of the router, which is the outcome of both α_{in} and back-pressure. α_{out} is the output process of router R_2. Suppose the overall service curve β^{R_2} of R_2 seen by R_1 is known, and the ideal service curve (without back-pressure) of R_1 is $\beta^{R_1'}$ (provided by the crossbar as defined in Section III-B), then according to [4], the overall service curve β^{R_1} of R_1 considering back-pressure is given as

$$\beta^{R_1} = \beta^{R_1'} \otimes \overline{(I_B \otimes (\beta^{R_1'} \otimes \beta^{R_2}))} \tag{5}$$

where B is the buffer size, and I_B is defined as $I_B(t) = \infty$ for $t > 0$ and $I_B(0) = B$. The horizontal bar is the operation for sub-additive closure. In this way, we can derive the service curve of each router and then recursively concatenate them based on (4) from destination to source to get the concatenated service curve for all routers along a flow's path.

D. Switch-Contention Stall

Packet stall can happen at switch allocation stage, when all front packets in different virtual channels compete for the same crossbar input or output port. Here we model a generic switch arbiter that allocates time slots to different input ports according to their priorities. In Fig. 3(a), we show an example where two flows arrive at a router and compete for the same output link to illustrate service curves experienced by each flow.

The length of individual slot $s_i (i = 1, 2)$ assigned for flow f_i is proportional to the relevant priorities (arrival rates) of incoming flows and should only take values that are multiples of a cycle length, and the corresponding total cycle length is $s = \sum_i s_i$. Assume the full ideal service curve of the router is $\beta^{R_1'}$, then the partial ideal service curve for f_i, as shown in Fig. 3(b), is proportional to the slot distribution

$$\beta_i^{R_1'} = \frac{s_i}{s} \beta^{R_1'} \otimes \delta_{s-s_i} \tag{6}$$

where δ_{s-s_i} is the delay bound when stream f_i just missed its slots in the worst case and has to wait for the next round.

As for stream f_i, (5) can be rewritten as

$$\beta_i^{R_1} = \beta_i^{R_1'} \otimes \overline{\left(I_B \otimes \left(\beta_i^{R_1'} \otimes \beta_i^{R_2}\right)\right)}. \tag{7}$$

Then by plugging (6) into (7) we can derive the allocated service curve for a specific stream at each router it traversed, and the concatenated service curve for the entire path according to (4). Finally, the worst-case delay bound can be obtained by applying the principles of network calculus, essentially by (3). In this paper, we assume a round-robin arbiter for every router in the network, which implies that the priorities of each flow are proportional to their arrival rates to a specific router. While not considered here, such results can be obtained in a similar way for other scheduling policies like fixed priority (FP), rate monotonic (RM), and earliest deadline first (EDF).

IV. Slack Optimization for Saving Energy

Applying the methodology in the previous section, we are able to bound the worst-case packet delay for individual application streams, hence obtaining worst-case slack—the time interval between the delay bound and its prespecified transmission deadline, which indicates the number of cycles that a packet can be further delayed in the network. Based on this idea, we propose to optimize network energy-efficiency under deadline constraints through voltage and frequency scaling.

A. Frequency and Voltage Scaling

The existence of packet slack implies that we can still achieve the required performance while lowering the operating frequency of some routers instead of making them run at a homogeneous high speed. The supply voltage, in the meantime, can be reduced together with the frequency to reduce energy.

Voltage-frequency islands (VFI) [19] have been adopted for achieving fine-grain system-level power management. A fine granularity partitioning could assume that each module in the design belongs to a different island [25] for best flexibility, or find the optimum partitioning via island merging [27] for energy savings. Here we are doing static voltage-frequency

assignment and model these routers to be able to run at its own voltage and frequency. For completeness, we also explore the energy gain of operating all routers at homogeneous voltage and frequency.

In order for the network routers to operate at different frequencies, they should communicate in a globally asynchronous locally synchronous (GALS) mechanism. To address synchronization latency, we adopt the fast synchronizer proposed by Dally and Tell [6] which adds only half cycle of synchronization delay that can be well absorbed in the buffer write stage.

Note that if we assign a lower frequency to a router, its service curve is affected accordingly. Specifically, the packet service rate will decrease and the time it takes to traverse a router will increase. For example, for a router R_k with rate-latency service curve $\beta^{R_k}_{\lambda,T}(t) = \lambda[t-T]^+$ as discussed in Section III, when its operating frequency is scaled by a factor η, the service rate will scale accordingly, whereas the initial delay remains the same. Its new service curve is modeled as

$$\beta^{R_k}_{\eta\lambda,T/\eta}(t) = \eta\lambda[t - T/\eta]^+. \tag{8}$$

Then the worst-case packet delay should be updated to check if there is remaining slack time for further optimization. Our energy optimization algorithms are described below in detail.

B. Straightforward Solutions

There are two straightforward ways to trade slack for energy savings. One is to simply scale down all the routers simultaneously by the same factor, which will keep them running at homogeneous frequency and voltage. The other is through exhaustive search to find out the optimum assignment. However, the former approach is not flexible enough for adjustment and results in performance bottleneck in the network, because the degree to which the network speed can be reduced will be limited by the packet flow with minimum slack. The latter way enumerates all combinations of voltage/frequency pair for different routers. Therefore, it is time-consuming and not scalable for a large network with many possible voltage-frequency levels.

C. Interference-Aware List-Based Algorithm

As illustrated above, it is not flexible enough to treat routers homogeneously while conducting voltage and frequency scaling. The best power-efficient methods should explore the internal heterogeneity of the network. One way to learn about the heterogeneity for such flow-based network is through analyzing the network interference of individual router.

Noticing that even though any routers in the network can be a candidate for frequency and voltage scaling, their impacts on end-to-end packet latency vary. Because routers with more contentions are likely to have greater impact on end-to-end delay, we propose to conservatively start scaling with the ones that have less interfering streams, expecting they would cause less extra delay with the same amount of energy reduction, and therefore safely leave more space for slack optimization. Specifically, we propose to sort individual active routers in the ascending order of contentions, and store the results into a priority list. Algorithm 1 describes our interference-aware priority assignment mechanism in detail. It firstly groups the routers into different clusters, where every element in a cluster has the same number (m) of flows passing through. These clusters are then sorted in ascending order of the number of m. Within each cluster, the routers are further categorized into different lists by analyzing the actual number of contention flows (n, $(0 \le n \le m)$), and the lists are also sorted in ascending order of n. Note that interfering flows refers to those that are actually contending for a switch port in a router, and thus may be a subset of the total number of streams that are flowing through. At last, within each list, routers are sorted in ascending order of their distance to the destination if they belong to the same flow, otherwise they have the same priority and can be randomly picked for scaling.

Algorithm 1: Interference-Aware Search Algorithm for Priority Assignment

```
Result: Priority list for router frequency and voltage scaling
for every active router R_i do
    if R_i has m streams passing by then
        put R_i in Cluster C_m;
    end
end
Sort Cluster C_m(m = 1, 2, ...) in ascending order of m;
for every Cluster C_m(m = 1, 2, ...) do
    for every router R_i do
        if R_i has n (0 ≤ n ≤ m) streams interfering then
            put R_i in List L_n;
        end
    end
    Sort List L_n(n = 1, 2, ...) in ascending order of n;
    for every router in list L_n do
        if R_i and R_j belongs to the same stream then
            sort the routers in L_n in ascending order of
            their distance to destination;
        else
            Priorities: P_{R_i} = P_{R_j};
        end
    end
end
```

TABLE II
PRIORITY LIST FOR FIG. 1(C)

C_1	C_2	C_3
$(R_5 \rightarrow R_2 \rightarrow R_0) \rightarrow R_3 \rightarrow$	$R_1 \rightarrow R_6 \rightarrow$	R_4

We greedily ramp down the frequency of the router at the head of this list until no more energy reduction is possible without violating deadlines. Then we pop the head of the list and start to work on the next router.

Taking Fig. 1(c) as an example, we can obtain the priority list as show in Table II. Note order of R_5, R_2, and R_0 are randomly selected.

With this result, we scale down network routers according to the priority list and redo worst-case delay analysis presented in Section II to check if there is any deadline miss. Hence we can continuously choose routers for frequency and voltage scaling

as long as there is remaining slack time, leading to an energy-efficient NoC design for hard real-time embedded systems.

D. Energy-Aware Heuristic Search Algorithm

The previous list-based algorithm proposes to conduct voltage and frequency scaling based on the network traffic flows, but it lacks an accurate model to capture the network contention. Additionally, the list-based algorithm would ramp down the voltage/frequency level of routers through one iteration of the list, i.e., reducing the voltage/frequency of a router to the maximum degree before turning into the next router, whereas a more fine-grained and effective approach is to select among all routers during each step of scaling.

Therefore, we propose an energy-aware heuristic search (EHS) algorithm, which can find an efficient solution leveraging network heterogeneity and avoiding exhaustive search at the same time. Specifically, we abstract a NoC energy model and integrate it into our worst-case delay analysis framework to automatically generate the frequency-voltage assignments.

1) Energy Models: The set of nodes in the network is denoted by $T = \{0, 1 \ldots N-1\}$. The supply voltage-frequency pairs of each node $i \in T$ are given by (V_i, f_i). Then the sum of dynamic and static energy consumption associated with node i is

$$E(V_i, f_i) = E_d(V_i, f_i) + E_s(V_i, f_i). \tag{9}$$

The dynamic energy part can be calculated through

$$E_d(V_i, f_i) = M_i * E_p(V_i, f_i) \tag{10}$$

where M_i is the total number of packets that traverse node i during execution, and $E_p(V_i, f_i)$ is the energy consumption when a packet traverses node i

$$E_p(V_i, f_i) = E_{\text{buffer}} + E_{\text{switch}} + E_{\text{link}} \tag{11}$$

where E_{buffer}, E_{switch}, and E_{link} represent the energy dissipated at input buffers, switch and link and are found experimentally using DSENT [36].

The static energy $E_s(V_i, f_i)$ part is defined as

$$E_s(V_i, f_i) = P_s(V_i, f_i) * t \tag{12}$$

where t is the system execution time, while $P_s(V_i, f_i)$ is static power for node i and can be obtained as

$$P_s(V_i, f_i) = I^i_{\text{static}} * V_i \tag{13}$$

where I^i_{static} is the leakage current for node i and can also be extracted from DSENT [36].

Thus, combining (9), (10), and (12), the total NoC energy consumption for an application can be expressed as

$$E = \sum_{i=0}^{N-1} (M_i * E_p(V_i, f_i) + P_s(V_i, f_i) * t). \tag{14}$$

2) Algorithm Description: If node i is scaled from the current voltage-frequency level (V_i^k, f_i^k) to the next lower level (V_i^{k+1}, f_i^{k+1}), then energy reduction can be expressed as

$$\Delta E_i = \sum_{i=1}^{N} (M_i * (E_p(V_i^k, f_i^k) - E_p(V_i^{k+1}, f_i^{k+1})) + P_s(V_i^k, f_i^k) * t^k - P_s(V_i^{k+1}, f_i^{k+1}) * t^{k+1}). \tag{15}$$

Algorithm 2: Energy-Aware Heuristic Search Algorithm

```
Result: Frequency-voltage assignment for all nodes
Initialize: Flag = 0;
while Flag == 0 do
    Flag = 1;
    for i ← 0 to N − 1 do
        if f_i is at its lowest level then
            continue;
        else
            /* worst-case delay analysis
               */
            Calculate d_{s_j}(∀s_j ∈ S), Δd_i and ΔE_i if scaling
            down f_i by one level, and insert them into a
            list L as one element;
        end
    end
    Sort L in ascending order of Δd_i/ΔE_i, associated
    with the original index i;
    for each entry in the list L do
        if d_{s_j} < D_{s_j}(∀s_j ∈ S) then
            Scale down f_i by one level; Flag = 0; break;
        end
    end
end
```

Under deterministic routing, the only undetermined variable is the system execution time t^k. Assume the set of application streams is denoted by $S = \{s_1 \ldots s_m\}$. The number of packets injected by s_j is M_{S_j} with average injection rate r_{s_j}. Then t^k can be approximated from the slowest stream

$$t = \max_{s_j \in S} \left\{ M_{s_j} / r_{s_j} \right\}. \tag{16}$$

This is because the end-to-end packet delay is negligible compared to t, especially when the packet number is large.

At the same time, the service curve β^{R_i} is modified based on (8) if scaling i and the new stream delay d_{s_j} is calculated via the worst-case delay analysis in Section II. The accumulated slack cost after scaling is represented as

$$\Delta d_i = \sum_{s_j \in S} \Delta d_{s_j} \tag{17}$$

where Δd_{s_j} is the reduced slack for stream s_j.

Our heuristic search algorithm uses $\Delta d_i / \Delta E_i$ as a measure of the slack cost and the energy gain if we adjust the voltage-frequency level of a router i. The pseudo-code below outlines this algorithm. Specifically, the algorithm iterates through all routers in the network. At each iteration, it generates a list containing the slack costs and the energy gains for all routers in the network, picks the router i with lowest $\Delta d_i / \Delta E_i$ without causing deadline violations, and steps down the router's voltage and frequency. Finally, the algorithm terminates when there is no router in the network of which the voltage and the frequency can be further reduced without causing timing violations.

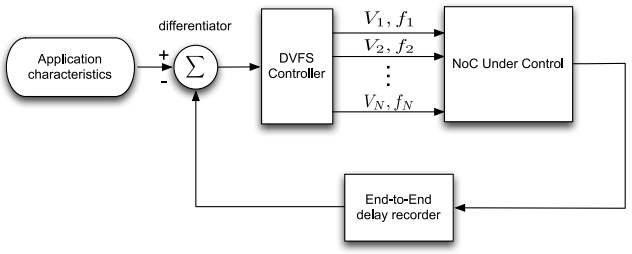


Fig. 4. Feedback control system for dynamic voltage and frequency scaling of network routers.

TABLE III
SIMULATION PARAMETERS

Baseline Network Configuration			
Topology	2D mesh	Phit width	128bits
Size	4×4=16	Frequency	2GHz
VC #	3	Voltage	1.5v
Buffer depth	4 flits	Routing	dimension-order
Video Application Configuration			
MJPEG		PiP (HR)	PiP (LR)
Frame	352×240	Frame	704×576
Period	90,000	PE service	226.57
Throughput	307.2KB	Macroblock #	1584
JPEG size	8Kb	Rate: 25 frames/s	Rate: 12.5 frames/s
Deadline Constraint (cycle)			
$D_1 = 50$		$D_2 = 95$	$D_3 = 50$

For an N-node NoC with k voltage-frequency levels for each node, the algorithm complexity of our EHS algorithm is $(k-1) * N^2 logN$, compared to k^N for exhaustive search.

V. DYNAMIC VOLTAGE AND FREQUENCY SCALING

The optimization approaches presented in the previous section depend on nominal traffic parameters known at design time. The proposed framework can statically assign frequencies and voltages to individual routers based on fixed patterns of network flows. However, during runtime, there may be changes of network traffic that would possibly cause deadline misses. On the one hand, a single network flow may experience different phases of transmission with different arrival rates, apart from traffic bursts that have already been accounted in our model. On the other hand, applications may come and go during runtime, which indicates the number of flows and the mapping of flows will change as well. In these scenarios, it is inappropriate to fix the network configuration at design time, which may either hurt the network power efficiency or cause packet deadline misses. Therefore, in this section, we design a feedback control system such that the voltages and frequencies of individual routers can be dynamically adjusted according to the traffic characteristics.

Fig. 4 depicts the feedback-control system. Specifically, the input application characteristics include the arrival rates, bursts, and prespecified deadlines of all the network flows. The DVFS controller takes into account these input parameters and computes the optimum voltage and frequency assignments using the methods in Section III. As a result, the NoC serves the network flows under such configurations. In the mean time, the end-to-end packet delays are checked and the longest delay for each flow during a period is recorded for comparing with application deadlines. Subsequently, the

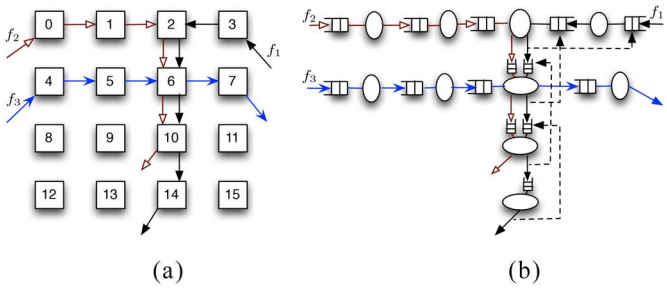


Fig. 5. Case study: Three video streams. (a) Mapping of application streams. (b) Resource sharing analysis.

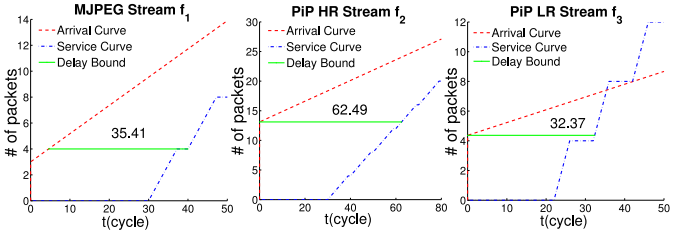


Fig. 6. Delay bounds for three flows.

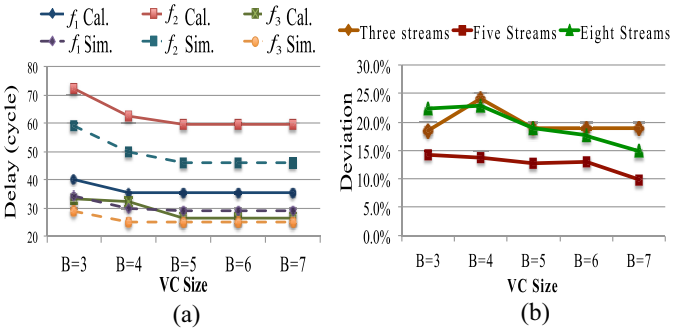


Fig. 7. Calculated (*Cal.*) versus simulated (*Sim.*) delay bound. (a) Delay comparison for three streams. (b) Average difference for three, five, and eight streams.

DVFS controller reconfigures the NoC routers once it senses changes of network traffic from the differentiator. Also, partitions of different application phases should be carefully conducted to avoid frequent frequency/voltage reconfiguration.

Calculating the optimum voltage and frequency assignment, and transmitting control and feedback information over the network for reconfiguration requires a nonnegligible amount of time. Therefore, the DVFS controller needs to be activated long enough (denoted as T) to perform these operations. After this time interval, the worst-case end-to-end packet delays will be reset in the recorder. This allows the recorder to capture the new worst-case delays of individual flows in order to check if there is any new traffic change.

The outcomes generated from the differentiator are twofold. If negative slack is detected, it indicates the network is operated too slow to prevent deadline misses. This may result from the increase of arrival rates of some existing flows or the generation of new flows. If positive slack is detected, the network is working safely, guaranteeing all packet flows can reach their destinations within the deadlines. However, there may be chance that the network is over-designed, due to the decrease of arrival rates or even departure of some flows.

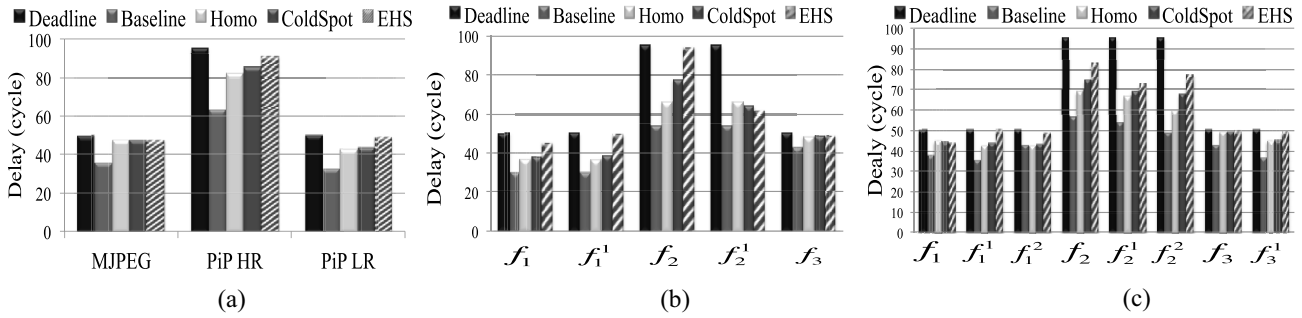


Fig. 8. Worst-case delay bound after frequency scaling for three, five, and eight streams. (a) Delay bounds for three streams. (b) Delay bounds for five streams. (c) Delay bounds for eight streams.

Therefore, in both cases, the DVFS controller will be triggered to calculated the optimum frequency and voltages assignments for individual routers. Note that, although we use a conservative model based on network calculus to predict worst-case packet latencies, deadlines misses are inevitable in case of significant abrupt traffic fluctuations.

VI. Experiments

A. Experimental Setup

We implemented a cycle-accurate network simulator based on the booksim 2.0 simulator [17], with dynamic and leakage power numbers extracted from DSENT [36].

We also analyze the timing behavior of some video applications and characterize the arrival curves for packet streams. Table III shows the simulator and benchmark configurations.

1) Motion-JPEG (MJPEG) Decoder: The MJPEG decoder is a video codec in which each video frame is compressed as a JPEG image. The video of 352×240 pixels is spilt into JPEG image size of 8 Kb. The maximum throughput is 307.2 KB per invocation with a period of 90 000 cycles.

2) Picture-in-Picture (PiP): We use two sets of video clips: regular clips with moderate to high motion content and clips displaying still images. These two sets characterize the two streams high-resolution (HR) and low-resolution (LR). Incoming streams have the same frame resolution of 704×576 pixels but will be down-scaled for LR, and each frame consists of 1584 macroblocks. Frames are read at a constant rate of 25 frames/s for HR and 12.5 frames/s for LR. The service offered by a PE is 226.57 macroblocks/ms.

A JPEG image or a macroblock is treated as a packet, and we derive arrival curves for the three packet streams based on their throughput and frame rates

$$\text{MJPEG stream } f_1\text{: } \alpha_1(t) = 0.218t + 3.0 \tag{18a}$$

$$\text{PiP HR stream } f_2\text{: } \alpha_2(t) = 0.175t + 13.109 \tag{18b}$$

$$\text{PiP LR stream } f_3\text{: } \alpha_3(t) = 0.086t + 4.37. \tag{18c}$$

And the baseline service curve is shown in (19) for a generic wormhole NoC router that can process one packet per cycle with a total pipeline length of five cycles

$$\text{Baseline router: } \beta(t) = [1.0 \times t - 5]^+. \tag{19}$$

As a case study, we consider that the three application streams are mapped in a 4×4 mesh network shown in Fig. 5(a) with deterministic routing. Fig. 5 shows the resource sharing, including feedback loops in stream f_1 as an example. A detailed network configuration is shown in Table III.

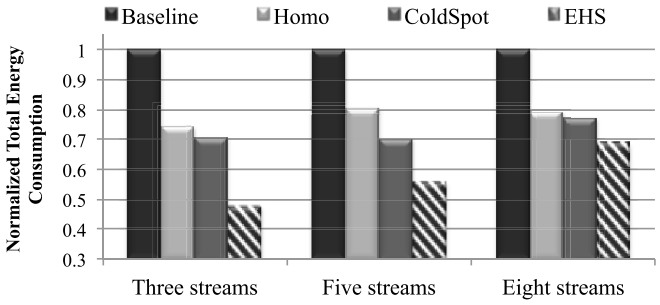


Fig. 9. Comparisons of energy consumption under different optimization schemes.

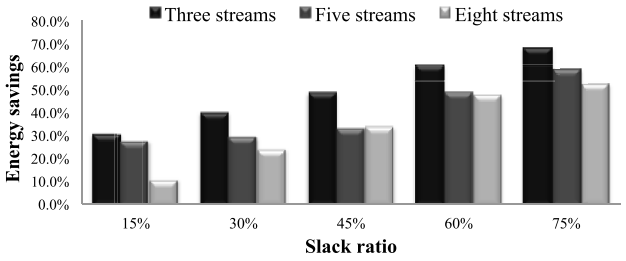


Fig. 10. Energy savings when slack varies.

B. Experimental Results

We use the methodology in Section II to analyze the worst-case latency in our case study. Results are shown in Fig. 6.

At the same time, we run simulation to get the maximum packet latency for individual streams. A comparison between the calculated worst-case delay bound (solid lines) and the simulated maximum packet latency (dashed lines) when varying virtual channel buffer size B is shown in Fig. 7(a). We can see that the calculated delay bounds are fairly tight.

Apart from the case study with three applications streams, we duplicate each of the three sample streams to generate more streams and map them on the NoC platform to form different traffic scenarios. The whole process is conducted randomly. Here we consider running another five streams (two MJPEG, two PiP HR, one PiP LR) and eight streams (three MJPEG,

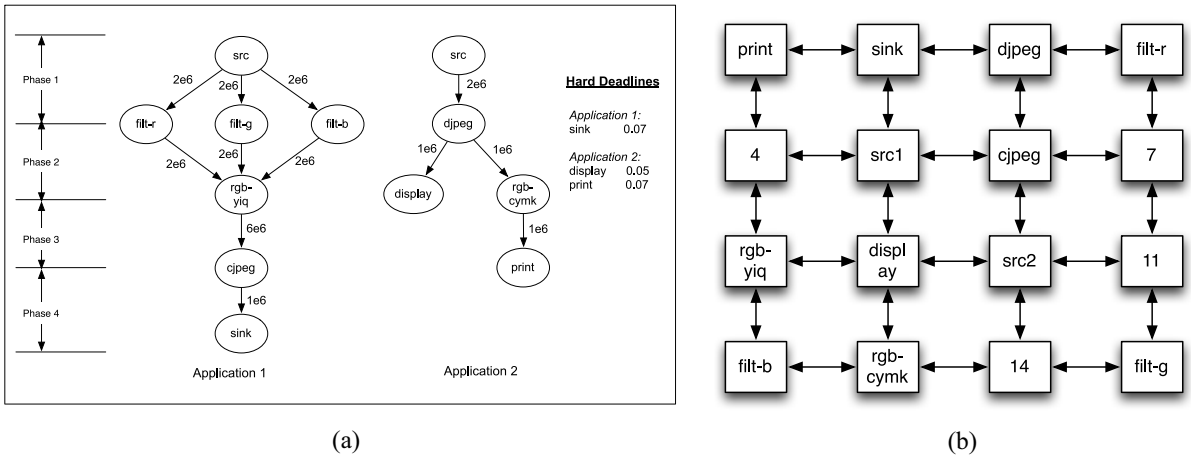


Fig. 11. Application task graphs and mapping. (a) Application task graphs for consumer benchmark. (b) Mapping results of the consumer benchmark.

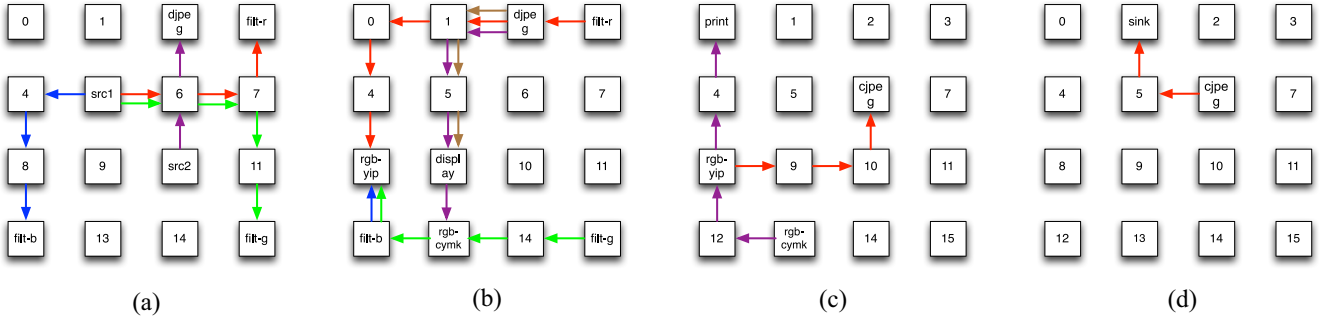


Fig. 12. Runtime execution phases of the applications. (a) Phase 1. (b) Phase 2. (c) Phase 3. (d) Phase 4.

three PiP HR, two PiP LR). Similarly, we do worst-case delay analysis to derive the delay bound and run simulation to get the maximum packet latency. Due to space limitation, we only show the differences between calculated delay bounds and simulated results as the ratio with respect to simulated results, when buffer size varies from 3 to 7, as shown in Fig. 7(b).

We find that the average difference is 17.2% while the ratio is generally decreasing as VC size increases. This is because with a small VC size, the effect of back-pressure is more salient and the results from our analytical model is more pessimistic. With VC size increased, the effect of back-pressure is smoothed out, resulting in a tighter delay estimate that converges with the simulation results.

Furthermore, we perform the proposed EHS algorithm to reduce the energy of routers, assuming that three discrete frequencies are available (1.0, 1.5, and 2.0 GHz), and the minimum required voltage is 0.8, 1.2, and 1.5 volts in 45nm CMOS, respectively. As a comparison, we also evaluate homogeneous scaling (Homo) in which case the frequency-voltage level of the routers in the whole network is scaled together. In addition, we compare EHS with another simple algorithm proposed in Section IV-C. It starts scaling routers with the ones that have less interfering streams. We call it *ColdSpot* here. Results of the worst-case delay bounds are shown in Fig. 8, where we refer to the *j*th duplicate of the *i*th sample stream as f_i^j. In addition, the normalized total network energy consumption are shown in Fig. 9.

As we can see from Fig. 8(a)–(c), there is no deadline miss using any of these scaling mechanisms. We define slack utilization as the amount of slack scavenged by the algorithm to save energy, divided by the amount of initial slack under baseline configuration. Our proposed EHS algorithm has effectively exploited individual stream slack, making the completion time (delay) close to the deadline with a slack utilization of 80.7% on average. In contrast, Homo only has a slack utilization of 53.9% on average, and *ColdSpot* has a slack utilization of 60.8%.

As for energy optimization, shown by Fig. 9, our proposed EHS mechanism significantly outperforms Homo and *ColdSpot*. On average, EHS achieves 42.7% energy reduction, while Homo saves 22.0% and *ColdSpot* saves 27.2%. These results confirm the effectiveness of our energy optimization algorithm. EHS can efficiently utilize the available slack of individual application streams for energy optimization, while for Homo case, it cannot exploit slack in fine granularity and therefore leads to relatively poor energy savings. *ColdSpot* achieves better energy saving but fails to capture the interference of flows accurately.

In addition, for sensitivity evaluation, we apply the proposed EHS algorithm with different slack ratios available. Slack ratio is the amount of slack over baseline worst-case latency. As shown in Fig. 10, Our EHS algorithm works well under different slack ratios, saving energy by 23.1%, 31.1%, 38.4%, 52.0%, and 59.7%, respectively.

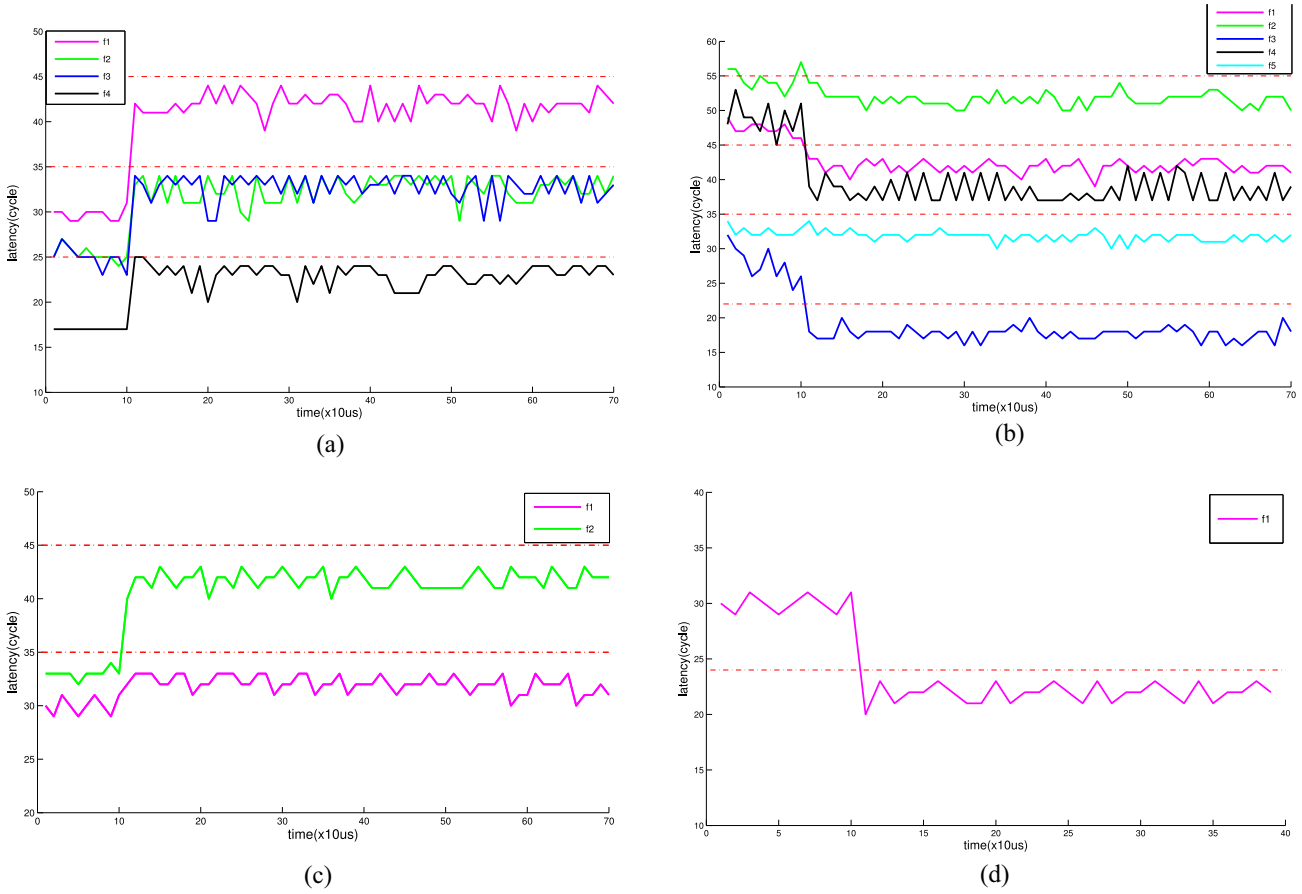


Fig. 13. Packet delays at different phases. (a) Phase 1. (b) Phase 2. (c) Phase 3. (d) Phase 4.

C. Evaluation on DVFS

In this section, we illustrate the operation of DVFS with our feedback-control strategy. Specifically, we take two applications from the consumer benchmark in the E3S benchmark suite [10] and analyze the effect of DVFS on their delay patterns. Fig. 11(a) shows the task graphs for these two applications, where each circle represents a task and the arc between two circles represents the communication. Each arc is labeled with the total quantity of communication. The total hard deadlines associated with each leaf of the application task graphs are also displayed.

For evaluation purposes, we partition the application graphs into different phases and distribute the total deadline into these phases in proportional to their communication volumes. For example, in Application 2, src→djpeg is Phase 1 with a deadline of 0.033, djpeg→display and djpeg→rgb-cymk are in Phase 2 with a deadline of 0.017 for each, and rgb-cmk→print belongs to Phase 3 with a deadline of 0.02. Note that the deadlines shown here are the final deadlines for transmitting the designated amount of messages as labeled in the communication task graph. We calculate the worst-case average arrival rates based on the deadlines and communication volumes associated with each arc. For example, the worst-case arrival rate for src→djpeg in Application 2 is 0.06 flit/cycle. The individual packet deadline of a flow are obtained by multiplying the best-case delay (based on the application mapping) by a constant factor. We run these two applications concurrently and assume tasks are randomly mapped onto the network, as shown in Fig. 11(b). In this case, not only a single application will change phases during execution, different applications will influence each other at runtime. For clarity, the application flows for each phase are presented in Fig. 12.

We conservatively set the control interval to be T = 100 μs [27] in our feedback controller. This duration is sufficient enough to change the operating voltage/frequency and exchange feedback and control signals over the network. We apply our DVFS scheme on these two applications and report the worst-case packet delays at every 10μs. Results are shown in Fig. 13 where the dashed horizontal lines mark the deadlines. Initially at the beginning of Phase 1, the network routers are operated at their highest voltages and frequencies. Then after an interval T, the system reconfigures and operates individual routers at the optimum frequency and voltage levels, while still guarantees no deadline violations. Subsequently at the beginning of Phase 2, the network is still operating at the old frequency/voltage levels as in Phase 1 even through new flows are generated. As a result, some flows fail to meet their deadline constraints. In response, the system reacts to such traffic change and ramps down the frequencies/voltages of some network routers to the new optimum configurations after an interval of T. Therefore, by tracking the slacks of each flow, our feedback-control system can dynamically adjust

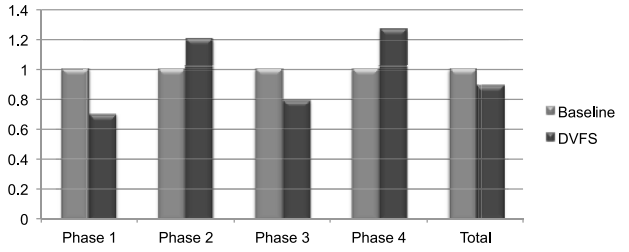


Fig. 14. Energy consumption at different phases.

the frequency/voltage levels of individual routers to achieve the optimum energy efficiency. It ramps down the frequencies/voltages of some routers when there is enough slack, as shown in Phases 1 & 3. Correspondingly, it raises the frequencies/voltages of some routers as long as deadline misses happen, as shown in Phases 2 & 4.

Moreover, we collect the overall network energy consumption under our DVFS scheme. The baseline design does not employ the DVFS scheme, i.e., the network configurations remain the same even though network traffic changes after entering the new phase. As we can see from Fig. 14, our DVFS scheme saves 30.1% and 20.9% of energy by cutting down the slacks, respectively in Phases 1 & 3. However, it incurs 19.8% and 26.3% of energy increase in Phases 2 & 4. As explained with Fig. 13, this is because the network must increase the voltage/frequency levels to avoid deadline misses. Nevertheless, the overall energy consumption of four phases is still reduced by 12.2%.

VII. Conclusion

In this paper, a formal analysis based on network calculus is adopted to obtain the worst-case slacks of packets in the NoC for hard real-time embedded systems, and used to trade slacks for energy savings by applying different voltages and frequencies to individual routers. Experimental results show that our worst-case delay analysis can derive an upper bound for packet latency, and our energy-aware heuristic search algorithm can effectively find the frequency-voltage assignment that can reduce network energy significantly under variable slack ratios. Additionally, our feedback-control strategy successfully adjusts the voltage/frequency levels of individual routers dynamically for the optimal energy-efficiency during runtime.

References

[1] L. Benini and G. D. Micheli, "Networks on chips: A new SoC paradigm," *IEEE Comput.*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[2] T. Bjerregaard and J. Sparso, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proc. Des. Autom. Test Eur.*, 2005, pp. 1226–1231.

[3] S. Chakraborty, S. Kunzli, and L. Thiele, "A general framework for analysing system properties in platform-based embedded system designs," in *Proc. Des. Autom. Test Eur.*, Munich, Germany, 2003, pp. 190–195.

[4] C.-S. Chang, *Performance Guarantees in Communication Networks*. Berlin, Germany: Springer, 2000.

[5] X. Chen *et al.*, "Dynamic voltage and frequency scaling for shared resources in multicore processor designs," in *Proc. 50th ACM/EDAC/IEEE Des. Autom. Conf.*, Austin, TX, USA, May/Jun. 2013, p. 114.

[6] W. Dally and S. Tell, "The even/odd synchronizer: A fast, all-digital, periodic synchronizer," in *Proc. IEEE Symp. Asynchronous Circuits Syst. (ASYNC)*, Grenoble, France, 2010, pp. 75–84.

[7] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. Des. Autom. Conf.*, 2001, pp. 684–689.

[8] W. J. Dally and B. Towles, Eds., *Principles and Pracitices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann, 2003, pp. 245–247.

[9] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Aergia: Exploiting packet latency slack in on-chip networks," in *Proc. Int. Symp. Comput. Archit.*, 2010, pp. 106–116.

[10] R. Dick. (2008). "*Embedded System Synthesis Benchmarks Suite (E3S)*," [Online]. Available: http://ziyang.eecs.umich.edu/~dickrp/e3s/

[11] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proc. Int. Symp. Comput. Archit.*, 2011, pp. 365–376.

[12] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Des. Test Comput.*, vol. 22, no. 5, pp. 414–421, Sep./Oct. 2005.

[13] K. Goossens and A. Hansson, "The Æthereal network on chip after ten years: Goals, evolution, lessons, and future," in *Proc. Des. Autom. Conf.*, 2010, pp. 306–311.

[14] N. Goulding *et al.*, "Greendroid: A mobile application processor for a future of dark silicon," *Hot Chips*, vol. 22, Aug. 2010.

[15] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sep./Oct. 2007.

[16] Y.-C. Huang, K.-K. Chou, and C.-T. King, "Application-driven end-to-end traffic predictions for low power NoC design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 2, pp. 229–238, Feb. 2013.

[17] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles, and W. J. Dally, *BookSim 2.0 User's Guide*. Palo Alto, CA, USA: Standford Univ. Press, 2010.

[18] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *Proc. Int. Symp. High-Perform. Comput. Arch.*, Salt Lake City, UT, USA, 2008, pp. 123–134.

[19] D. Lackey *et al.*, "Managing power and performance for system-on-chip designs using voltage islands," in *Proc. Int. Conf. Comput.-Aided Des.*, 2002, pp. 195–202.

[20] J.-Y. Le Boudec and P. Thiran, Network Calculus: A Theory of Deterministic Queuing Systems for the Internet (Lecture Notes in Computer Science), vol. 2050. Berlin, Germany: Springer, 2001.

[21] J. W. Lee, M. C. Ng, and K. Asanovic, "Globally-synchronized frames for guaranteed quality-of-service in on-chip networks," in *Proc. Int. Symp. Comput. Arch.*, Beijing, China, 2008, pp. 89–100.

[22] T. G. Mattson *et al.*, "The 48-core SCC processor: The programmer's view," in *Proc. Int. Conf. High Perform. Comput. Netw. Stor. Anal.*, New Orleans, LA, USA, Nov. 2010, pp. 1–11.

[23] A. K. Mishra *et al.*, "A case for dynamic frequency tuning in on-chip networks," in *Proc. Int. Symp. Microarchit.*, New York, NY, USA, 2009, pp. 292–303.

[24] K. Niyogi and D. Marculescu, "Speed and voltage selection for GALS systems based on voltage/frequency islands," in *Proc. Asia South Pac. Des. Autom. Conf.*, 2005, pp. 292–297.

[25] K. Niyogi and D. Marculescu, "Speed and voltage selection for GALS systems based on voltage/frequency islands," in *Proc. Asia South Pac. Des. Autom. Conf.*, 2005, pp. 292–297.

[26] U. Y. Ogras, R. Marculescu, P. Choudhary, and D. Marculescu, "Voltage-frequency island partitioning for GALS-based networks-on-chip," in *Proc. Des. Autom. Conf.*, 2007, pp. 110–115.

[27] U. Y. Ogras, R. Marculescu, D. Marculescu, and E. G. Jung, "Design and management of voltage-frequency island partitioned networks-on-chip," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 3, pp. 330–341, Mar. 2009.

[28] J. Ouyang and Y. Xie, "LOFT: A high performance network-on-chip providing quality-of-service support," in *Proc. Int. Symp. Microarchit.*, Atlanta, GA, USA, 2010, pp. 409–420.

[29] Y. Qian, Z. Lu, and W. Dou, "Analysis of communication delay bounds for network on chips," in *Proc. Asia South Pac. Des. Autom. Conf.*, Yokohama, Japan, 2009, pp. 7–12.

[30] Y. Qian, Z. Lu, and W. Dou, "Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip," in *Proc. 3rd ACM/IEEE Int. Symp. Netw. Chip (NOCS)*, San Diego, CA, USA, 2009, pp. 44–53.

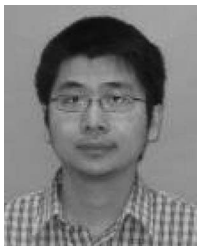
[31] Y. Qian, Z. Lu, and W. Dou, "Analysis of worst-case delay bounds for on-chip packet-switching networks," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 29, no. 5, pp. 802–815, May 2010.
[32] A. Raghavan *et al.*, "Computational sprinting," in *Proc. Int. Symp. High-Perform. Comput. Archit.*, 2012, pp. 1–12.
[33] G. Semeraro *et al.*, "Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," in *Proc. Int. Symp. High-Perform. Comput. Archit.*, 2002, pp. 29–40.
[34] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," in *Proc. 9th Int. Symp. High-Perform. Comput. Archit.*, 2003, pp. 91–102.
[35] A. Sharifi, H. Zhao, and M. Kandemir, "Feedback control for providing QoS in NoC based multicores," in *Proc. Des. Autom. Test Eur.*, Dresden, Germany, 2010, pp. 1384–1389.
[36] C. Sun *et al.*, "DSENT—A tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *Proc. 6th IEEE/ACM Int. Symp. Netw. Chip (NoCS)*, 2012, pp. 201–210.
[37] M. B. Taylor, "Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse," in *Proc. 49th ACM/EDAC/IEEE Des. Autom. Conf.*, San Francisco, CA, USA, 2012, pp. 1131–1136.
[38] M. B. Taylor *et al.*, "The Raw microprocessor: A computational fabric for software circuits and general-purpose programs," *IEEE Micro*, vol. 22, no. 2, pp. 25–35, Mar./Apr. 2002.
[39] G. Venkatesh *et al.*, "Conservation cores: Reducing the energy of mature computations," in *Proc. 15th Edition ASPLOS Archit. Support Program. Lang. Oper. Syst.*, vol. 38. 2010, pp. 205–218.
[40] W. Wang and P. Mishra, "PreDVS: Preemptive dynamic voltage scaling for real-time systems using approximation scheme," in *Proc. 47th ACM/IEEE Des. Autom. Conf. (DAC)*, Anaheim, CA, USA, Jun. 2010, pp. 705–710.
[41] W. Wang, P. Mishra, and S. Ranka, "Dynamic cache reconfiguration and partitioning for energy optimization in real-time multi-core systems," in *Proc. 48th ACM/EDAC/IEEE Des. Autom. Conf.*, New York, NY, USA, 2011, pp. 948–953.
[42] P. Zhou, J. Yin, A. Zhai, and S. S. Sapatnekar, "NoC frequency scaling with flexible-pipeline routers," in *Proc. Int. Symp. Low Power Electron. Des. (ISLPED)*, Fukuoka, Japan, 2011, pp. 403–408.

Jia Zhan (S'12) received the B.S. degree from the Harbin Institute of Technology, Harbin, China, and joined the Pennsylvania State University, University Park, PA, USA, in 2011, where he is currently pursuing the Ph.D. degree from the Department of Computer Science and Engineering.

His current research interests include a broad range of computer architecture, with an emphasis on network-on-chip for many-core processors and real-time embedded systems.

Nikolay Stoimenov (M'12) received the bachelor's and the Honours bachelor's degrees in computer science from the University of Adelaide, Adelaide, SA, Australia, in 2004 and 2005, respectively, and the Ph.D. degree in computer engineering from the ETH Zurich, Zurich, Switzerland, in 2011.

He is currently a Senior Scientist with ETH Zurich. His current research interests include models and methods for design of embedded real-time systems.



Jin Ouyang received the B.S. degree in microelectronics from Peking University, Beijing, China, in 2007, and the Ph.D. degree in computer engineering from the Pennsylvania State University, University Park, PA, USA, in 2012.

He is currently a Senior Engineer with NVIDIA Corporation, Santa Clara, CA, USA. His current research interests include computer architectures and VLSI systems and circuits.

Lothar Thiele (M'86) received the Diplom-ingenieur and the Dr.-Ing. degrees in electrical engineering from the Technical University of Munich, Munich, Germany, in 1981 and 1985, respectively.

He joined the Information Systems Laboratory at Stanford University, Stanford, CA, USA, in 1987. In 1988, he was the Chair of Microelectronics at the Faculty of Engineering, University of Saarland, Saarbrucken, Germany. He joined ETH Zurich, Zurich, Switzerland, as a Full Professor of Computer Engineering, in 1994. His current research interests include models, methods, and software tools for the design of embedded systems, embedded software, and bioinspired optimization techniques.

Mr. Thiele was the recipient of the Dissertation Award of the Technical University of Munich, in 1986, the Outstanding Young Author Award of the IEEE Circuits and Systems Society, in 1987, the Browder J. Thompson Memorial Award of the IEEE, in 1988, and the IBM Faculty Partnership Award in 2000–2001. In 2004, he joined the German Academy of Sciences Leopoldina. He was also the recipient of the Honorary Blaise Pascal Chair of University Leiden, The Netherlands, in 2005. He is an Associate Editor of the IEEE TRANSACTION ON INDUSTRIAL INFORMATICS, the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, *Journal of Real-Time Systems*, *Journal of Signal Processing Systems*, *Journal of Systems Architecture*, and *INTEGRATION, the VLSI Journal*. Since 2009, he has been a member of the Foundation Board of Hasler Foundation, Switzerland. Since 2010, he has been a member of the Academia Europaea. He joined the National Research Council of the Swiss National Science Foundation, in 2013.

Vijaykrishnan Narayanan (F'11) received the bachelor's degree from the University of Madras, Chennai, India, in 1993, and the Ph.D. degree from the University of South Florida, Tampa, FL, USA, in 1998, both in computer science and engineering.

He is a Professor of Computer Science and Engineering and Electrical Engineering with Pennsylvania State University, University Park, PA, USA. His current research interests include embedded systems, computer architecture, and power-efficient system design.

Yuan Xie (SM'07) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1997, the M.S. and the Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, USA, in 1999 and 2002, respectively.

He is currently a Professor with the Department of Electrical and Computer Engineering, University of California at Santa Barbara (UCSB), Santa Barbara, CA, USA. Before joining UCSB, he was a Professor with Pennsylvania State University, University Park, PA, USA, since 2003. He was also with IBM and AMD, from 2002 to 2003 and from 2012 to 2013, respectively. His current research interests include VLSI design, computer architecture, embedded systems design, and electronics design automation.