Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 537286, 12 pages
doi:10.1155/2012/537286

Research Article

A Buffer-Sizing Algorithm for Network-on-Chips with Multiple Voltage-Frequency Islands

Anish S. Kumar,[1] M. Pawan Kumar,[1] Srinivasan Murali,[2] V. Kamakoti,[1] Luca Benini,[3] and Giovanni De Micheli[4]

[1] *Indian Institute of Technology Madras, Chennai 600036, India*
[2] *iNoCs, 1007 Lausanne, Switzerland*
[3] *University of Bologna, 40138 Bologna, Italy*
[4] *EPFL, 1015 Lausanne, Switzerland*

Correspondence should be addressed to Anish S. Kumar, ask.anish@gmail.com

Received 17 July 2011; Accepted 1 November 2011

Academic Editor: An-Yeu Andy Wu

Buffers in on-chip networks constitute a significant proportion of the power consumption and area of the interconnect, and hence reducing them is an important problem. Application-specific designs have nonuniform network utilization, thereby requiring a buffer-sizing approach that tackles the nonuniformity. Also, congestion effects that occur during network operation need to be captured when sizing the buffers. Many NoCs are designed to operate in multiple voltage/frequency islands, with interisland communication taking place through frequency converters. To this end, we propose a two-phase algorithm to size the switch buffers in network-on-chips (NoCs) considering support for multiple-frequency islands. Our algorithm considers both the static and dynamic effects when sizing buffers. We analyze the impact of placing frequency converters (FCs) on a link, as well as pack and send units that effectively utilize network bandwidth. Experiments on many realistic system-on-Chip (SoC) benchmark show that our algorithm results in 42% reduction in amount of buffering when compared to a standard buffering approach.

1. Introduction

In modern SoC designs, power consumption is a critical design constraint as they are targeted as low-power devices. To achieve this, SoC designs employ power gating, where the cores are shutdown when they are unused. Instead of shutting down each core, certain techniques cluster cores into *voltage and frequency* (VF) *islands,* and when all the cores in an island are unused, the entire VI is shut down. The cores in a single VI have same operating voltage but can operate at different frequencies. Running cores at different frequencies is an effective method to trade off performance and power consumption.

Scalable on-chip networks, network-on-chips (NoCs), have evolved as the communication medium to connect the increasing number of cores and to handle the communication complexity [1–3]. With designs having multiple VF islands, the interconnect can reside in a separate island. By clustering the NoC into a single island, routing the VDD and ground lines across the chip becomes difficult. Instead, the NoC is spread across the entire chip with different components of the network operating at different voltage/frequency. If the core in an island is operating in a different frequency than the switch to which it is connected, the NI does the frequency conversion, and when a switch from one island is connected to another switch in a different island, *frequency converters* (FCs), such as the ones in [4, 5], are used to do the frequency conversion. Even if the two switches are operating at same frequencies, there might be clock skew for which synchronization is required.

In an NoC, a packet may be broken down into multiple flow control units called flits, and NoC architectures have the ability to buffer flits inside the network to handle contention among packets for the same resource link or switch port. The buffers at the source network interfaces (NIs) are used to queue up flits when the network-operating frequency is different from that of the cores or when there is congestion inside the network that reaches the source. NoCs also employ

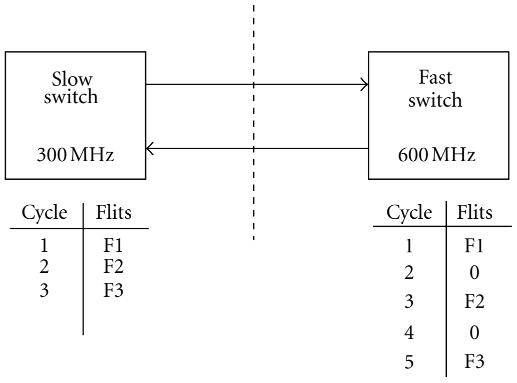


Figure 1: Bubbles generated moving from slow to fast clock.

some flow control strategy that ensures flits are sent from the switch (NI) to another switch (NI) only when there are enough buffers available to store them in the downstream component.

In many NoCs, a credit-based flow control mechanism is used to manage transfer of flits at full throughput. In this scheme the upstream router keeps a count of the number of free buffers in the downstream router. Each time a flit is communicated from an upstream router and is consumed by the downstream router, the credit counter is decremented. Once the downstream router forwards the flit and frees a buffer, a credit is sent to the upstream router and hence incrementing the credit count.

The network buffers account for a major part of the power and area overhead of the NoC in many architectures. For example, in [6], the buffers account for more than 50% of the dynamic power consumption of the switches. A major application domain for NoCs is in mobile and wireless devices, where having a low-power consumption is essential. Thus, reducing the buffering overhead of the NoC is an important problem.

As such NoCs are targeted for specific applications, the buffers and other network resources can be tuned to meet the application bandwidth and latency constraints. Several earlier works have dealt with application-specific customization of various NoC parameters, such as the topology, frequency of operation, and network paths for traffic flows [7–9]. In fact, several works have also addressed the customization of NoC buffers to meet application constraints [10, 11]. Many of the existing works utilize methods such as queuing theory and network calculus to account for dynamic queuing effects. While such methods could be used to compute the buffer sizes quickly, they have several limitations in practice. Most queuing theory-based works require the input traffic injection to follow certain probabilistic distributions, such as the Poisson arrival process. Other schemes require regulation of traffic from the cores, which may not be possible in many applications (details given in the next section).

Although these methods can be used for fast design space exploration, for example, during topology synthesis, final buffer allocation needs to consider simulation effects to accurately capture the congestion effects. In this paper, we present a simulation-based algorithm for sizing NoC buffers for application traffic patterns. We present a two-phase approach. In the first phase, we use mathematical models based on static bandwidth and latency constraints of the application traffic flows to minimize the buffers used in the different components based on utilization. In the second phase, we use an iterative simulation-based strategy, where the buffers are increased from the ideal minimal values in the different components, until the bandwidth and latency constraints of all the traffic flows are met during simulations. While in some application domains, such as in *chip multiprocessors* (CMPs), it is difficult to characterize the actual traffic patterns that occur during operation at design time, there are several application domains (such as mobile, wireless) where the traffic pattern is well behaved [12]. Our work targets such domains where the traffic patterns can be precharacterized at design time and a simulation-based mechanism can be effective.

With the communication subsystem running on different operating frequencies, the effective bandwidth and utilization on the links change. For example, when a switch operating at a slower clock frequency communicates to a switch at a higher operating frequency, bubbles may be introduced between flits. This will lead to overutilization of resources at the faster switch. As an example, consider a setup as illustrated in Figure 1. Here the destination is operating twice as fast as the source. Assume flits are forwarded at every cycle of the source switch. Since the destination is faster, the forwarded flits are consumed every other cycle (of the faster clock). This results in empty flits being generated in between the flits of a packet. The destination buffer is held by the packet till the tail flit leaves. And hence this leads to overutilization of the destination buffer, which otherwise would have been half this utilization. One way to effectively handle bubbles in the network is by employing *pack and send* (PS) units (discussed later in the paper).

Moreover, when switches of different frequencies communicate with each other, the number of buffers required varies depending where the *frequency converters* are placed. When the converters are placed to the slower clock, the link operates at the faster clock domain, thereby incurring smaller delay in transferring flits and credits. Thus, fewer buffers are required as the number of in-flight flits that need to be stored at the buffers is fewer. However, placing the converters near the slower clock leads to higher power consumption on the links as they are operating at a higher frequency. This effect also needs to be considered during sizing of buffers.

In this paper, we present a buffer-sizing algorithm for application-specific NoCs having multiple VF islands. We consider a complex mobile benchmark to validate the buffer-sizing algorithm presented. We also analyze the effect of placement of frequency converters on the buffer size. Our results show that there is 42% reduction in the buffer budgets for the switches, on an average. Based on the models from [6], this translates to around 35% reduction in the overall power consumption of the NoC switches. We also apply the approach on a variety of *system-on-chip* (SoC) benchmarks, which show a significant 38% reduction in buffer budget. Also, we study the impact of pack and send units on the buffer utilization. Results show that the PS units have a better utilization of network resources.

2. Related Work

A lot of work has gone into proposing techniques for scaling the voltage and frequencies of different IP cores on a chip. The authors of [13] propose techniques to identify optimal voltage and frequencies levels for a dynamic voltage and frequency scaling (DVFS) chip design. In [14] the authors propose methods of clustering cores into islands, and DVFS is applied for these islands. In our work, we assume such clustering of cores and NoC components as a part of the architecture specifications. In [15] the authors identify the theoretical bounds on the performance of DVFS based on the technology parameters.

With such partitioning of cores into VF islands becoming prevalent, globally asynchronous- and locally synchronous- (GALS-) based NoC designs have become the de facto interconnect paradigm. In [16] the authors propose an algorithm to synthesize an NoC topology that supports VF islands. This is one of the first approachs to design an NoC considering the support for shutting down of VF islands. The output of this algorithm can serve as input to our approach. In [17], the authors propose an reconfigurable NoC architecture to minimize latency and energy overhead under a DVFS technique. In [18], the authors propose asynchronous bypass channels to improve the performance of DVFS enabled NoC.

In this work we extend the proposed buffer-sizing algorithm to designs with VF islands to optimize NoC power and area while meeting the design requirements. Sizing buffers is critical for reducing the power and area footprint of an NoC.

In [10], the authors proposed an iterative algorithm to allocate more buffers for the input ports of bottleneck channels found using analytical techniques and also proposed a model to verify the allocation. The model assumes the Poisson arrival of packets. In [19], buffer sizing for wormhole-based routing is presented, also assuming the Poisson arrival of packets. The problem of minimizing the number of buffers by reducing the number of virtual channels has been addressed in [20] assuming that input traffic follows certain probabilistic distributions. In [21], a queuing theory-based model to size the number of virtual channels is proposed by performing a chromosome encoding of the problem and solving it using standard genetic algorithm, again assuming the Poisson arrival of packets. The authors of [22] proposed an analytical model to evaluate the performance of adaptively routed NoCs. This work again assumes the Poisson arrivals for the flows. In [23] the authors proposed a probabilistic model to find the average buffer utilization of a flow accounting for the presence of other flows across all ports. The authors of [24] used an approach to minimize buffer demand by regulating traffic through a delayed release mechanism and hence achieving the goal of appropriate buffer sizing. Unlike all these earlier works, we make no assumption on the burstiness of input traffic and the arrival pattern for packets.

In [25], the authors propose an algorithm to size the buffers, at the NIs, using TDMA and credit-based flow control. This work is complimentary to ours, as the authors target designing NI buffers to match the different rate of operation of cores and the network. A trace-driven approach to determine the number of virtual channels is presented in [26]. While the notion of simulation-driven design method is utilized in the work, the authors do not address the sizing of buffers. Our buffer-sizing methods are significantly different from methods for virtual channel reduction, as we need a much more fine grained control on buffer assignment. Towards this end, we present an iterative approach to buffer sizing that utilizes multiple simulation runs.

3. Design Approach

In this section, we give a detailed explanation of the design approach used for buffer sizing. The approach is presented in Figure 2. We use a two-phase method: static sizing, involving constraint solving, followed by a simulation-based approach.

We obtain two sets of inputs for the buffer-sizing approach: application and architecture specifications. The application specifications include the bandwidth and latency constraints for the different flows. The architecture specifications include the NoC topology designed for the application, routes for the traffic flows, the number of voltage/frequency islands, and flit width of the NoC.

We have the following assumptions about the architecture and application.

(i) For illustrative purposes, we consider input-queued switches for buffer sizing. In fact, the algorithm presented in generic and can be easily extended to output-queued (and hybrid) switches as well.

(ii) We define the term *number of buffers* used at a port to be the number of flits that the buffers can store at that port.

(iii) A wormhole, credit-based flow control is assumed in the NoC, which is widely used.

(iv) We do not explicitly consider the use of virtual channels. The algorithm, in fact, can be easily applied to architectures that support multiple virtual channels as well.

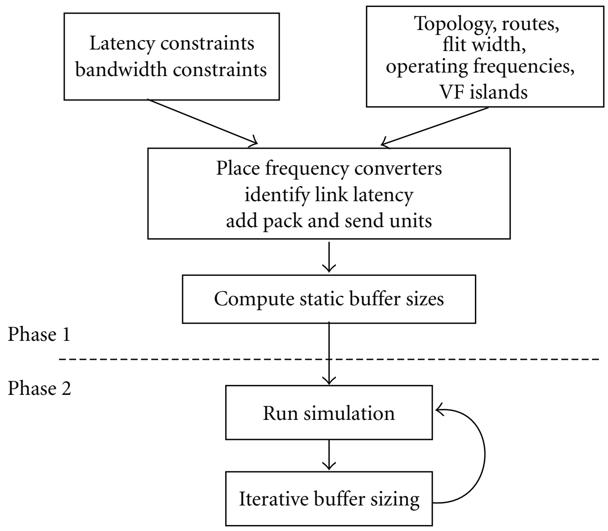


Figure 2: Buffer sizing design approach.

(v) We assume a uniform flit width across the whole network, which is again commonly observed in many NoCs.

(vi) We apply the buffer-sizing algorithm for a single application. In many designs, multiple applications can be run on the same device. The extension of the buffer-sizing method to support multiple application scenarios is similar to the extension of topology synthesis methods to consider multiple applications and has been thoroughly addressed by several researchers before [27]. Hence, we only show the core method applicable for a single application here.

The output of the buffer-sizing algorithm is the number of flit buffers at each input port of the different switches. We only perform the sizing of the switch buffers, and we refer the reader to earlier works on sizing NI buffers that consider the different rates of the cores and the network [25].

The algorithm phases are as follows.

Phase 1 (*Static Sizing*). To achieve full throughput and utilization on all the switches and links, the credit-based flow control mechanism requires a minimum number of buffers that depends on the number of cycles to traverse the links. In an application-specific topology, many parts of the network can have much less than 100% utilization. In this first phase, we formulate mathematical models relating the buffering at a port with the bandwidth utilization of the port and capturing latency constraints of traffic flows. We build a *linear program-* (LP-) based model to minimize the number of buffers at a port based on the utilization at the port and to respect the latency constraints of all flows across the different paths.

Phase 2 (*Simulation-Based Sizing*). In the second phase, we perform simulation of the NoC with the buffering values obtained from Phase 1. There are three important parameters of a traffic flow that significantly affect the congestion behavior: the bandwidth of the flow, the burstiness, and the number of flows overlapping at each link/switch port. While the static sizing mechanism considers the bandwidth of flows to compute utilization, the effects of burstiness and overlapping flows are considered during this second phase. We run simulations and utilize methods to iteratively increase buffers at ports until the bandwidth and latency constraints of all flows are met.

4. Buffer Sizing

4.1. Basic Architecture. In this section, we formulate the problem of buffer sizing.

We represent the communication constraints between the cores using a core graph.

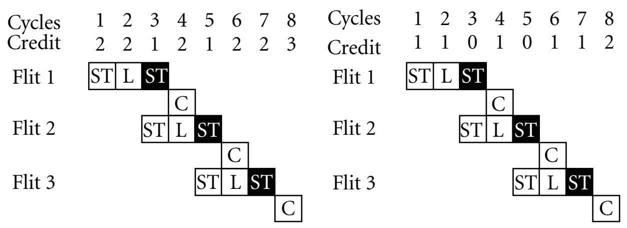
Definition 1. The core graph is a directed graph, $G(V, E)$ with vertex $v_i \in V$ representing the core and the directed edge $e_{i,j} \in E$ connecting vertices v_i and v_j, representing the communication link between the cores. The edge weight, $\mathrm{comm}_{i,j}$, denotes the communication bandwidth between v_i and v_j. The set F represents the set of flows between the cores.

An NoC graph denotes the NoC topology and the capacity of the links in the topology.

Definition 2. The NoC graph is a directed graph, $T(P, Q)$ with vertex $p_i \in P$ representing the switch and the directed edge $q_{i,j} \in Q$ connecting the vertices p_i and p_j representing the link connecting the switches. The edge weight, $\mathrm{bw}_{i,j}$, denotes the link bandwidth or capacity available across p_i and p_j.

Definition 3. Let the set of operating frequencies of various domains (in GHz) be denoted by the set D. Let freqency (i) be a mapping function that maps a chip component to the frequency at which the domain is operating at:

$$\text{frequency (i)} : \{V, P\} \longrightarrow D, \quad i \in V, P \tag{1}$$



(a) With 3 buffers

(b) With 2 Buffers

FIGURE 3: Timing diagram of two different buffer configurations. ST—switch traversal delay, L—link latency, C— credit latency.

When crossing VF islands, converters are required to do the frequency conversion. For this purpose, we use FC units that are basically dual-clocked FIFOs. The size of the FCs is uniform throughout the design. Most FCs incur a delay penalty for traversal, which is typically few cycles of the slow clock [28]. We denote this latency by FC_lat.

The latency of a link is the sum of the latency to traverse the FC and link traversal latency. The link traversal latency is defined by the frequency at which the link is operated. Let unit_len denote the distance in mm a signal can traverse in 1 *ns*. This can be determined based on the design's technology node. Then the latency of a link is given by

$$N_{i,j} = \text{FC\_lat} + \frac{1}{\text{freq}} \times \frac{\text{length}_{i,j}}{\text{unit\_len}} \tag{2}$$

$$\text{freq} = \begin{cases} \text{frequency } (i) & \text{FC near destination,} \\ \text{frequency } (j) & \text{FC near source,} \end{cases} \tag{3}$$

where freq denotes the operating frequency of the link and it depends on where the FC is placed on the link, and $\text{length}_{i,j}$ denotes the length of the link in mm.

The bandwidth or capacity of a link is given by the product of link width and frequency of operation of the link:

$$\text{bw}_{i,j} = \text{freq} \times \text{link\_width}_{i,j}. \tag{4}$$

Definition 4. The links traversed by a flow, f_k, $\forall k \in F$, connecting source s_k and destination d_k is represented by the set path_k.

The utilization $U_{i,j}$ of a link $q_{i,j}$ is the sum of the bandwidths of all the flows using the link divided by the capacity:

$$U_{i,j} = \frac{\sum_{l,m} \text{comm}_{l,m}}{\text{bw}_{i,j}}, \quad \forall l, m, k, \qquad \text{s.t. } q_{i,j} \in \text{Path}_k,\ s_k = v_l,\ d_k = v_m. \tag{5}$$

We assume a *pack and send* (PS) unit that can be used to better utilize the network resource. A PS unit is a 1-packet-long buffer that holds an entire packet before forwarding it to the downstream buffer. Employing PS units changes the above link utilization $U_{i,j}$.

The NoC architecture assumes a credit-based flow control mechanism. The following is a lemma for the number of buffers required in the downstream switch for credit-based flow control mechanism to support full throughput and utilization [29].

Lemma 5. *For a link with delay N cycles, in credit-based flow control, the number of flit buffers required at the downstream router in order to get 100% throughput is at least* $(2N + 1)$.

The intuitive reasoning for this buffering value is as follows. A flit takes N cycles to reach the next switch and, when it leaves the downstream buffer, the credit takes another N cycles to reach back and it takes one more cycle to process the credit. Thus, the overall time delay between sending a flit and processing the corresponding credit is $(2N + 1)$. During that time, under full utilization, the same number of flits could be sent from the sender switch which needs buffering at the downstream switch.

When the link utilization is less than 100%, the downstream router needs not have $(2N + 1)$ buffers and can be sized according to the utilization. The illustration in Figure 3 shows that buffers in the downstream router can be less than $(2N+1)$. In the example, two setups with downstream router having 3 and 2 buffers and link latency of 1 cycle are shown. The flow is assumed to have a 50% link utilization with the packet comprising of 1 flit. The packets are generated every other cycle, hence having utilization of 50%. In Figure 3(a), the timing diagram for a setup with 3 buffers and the credit counter value (available buffers at downstream router) at each cycle are shown. The same throughput (50%) can be achieved with 2 (lesser than $(2N + 1)$) buffers (Figure 3(b)). However, when the number of buffers is reduced from the ideal values, the packet latencies increase. For example, consider a 4-flit packet in the above scenario with 2 buffers. Since, the buffers are reduced from the ideal, the flits can be sent only every other cycle, and hence the packets have a latency of 7 cycles to be sent from the upstream to the downstream switch. Thus, when reducing the buffer size, we should also consider whether the latency constraints for the flows are met.

Table 1 summarizes the different parameters of the network.

5. Static Buffer Sizing

The latency of a link in the network is defined by the rate at which the link is clocked. Without loss of generality, let us assume $N_{i,j}$ to be the number of cycles needed to traverse the link $q_{i,j}$. Then the minimum buffering required at a port, based on the utilization at the port, is given by

$$\beta_{i,j} \geq \left(2N_{i,j} + 1\right) * U_{i,j}, \tag{6}$$

Table 1: Network parameters.

Parameter	Description
V	Set of IP cores
P	Set of NoC switches
D	Set of frequencies of VF islands
F	Set of flows in the network
FC_lat	Latency of frequency converter
$\text{length}_{i,j}$	length of link in mm
$N_{i,j}$	Link latency in cycles
$U_{i,j}$	Link utilization
$\beta_{i,j}$	Buffer size at link $q_{i,j}$
ps_k	Packet size of flow k
LC_k	Latency constraint of flow k

where $\beta_{i,j}$ represents the buffers statically assigned at the port connecting switches p_i & $p_j \in P$.

Let the latency constraint that needs to be met by a flow f_k, which is obtained as part of the input application specifications, be LC_k. The latency of a flow depends on the size of the buffers along its path and the size of packet (in flits). For the first flit of the packet, the latency is given by the length of the path (hops), and for the consequent flits it is determined by the buffering available at the downstream router. When buffering is less than the ideal value at an input port that is connected to the link $q_{i,j}$, the body (and tail) flits may need to wait at the upstream switch for credits to reach back. This delay for the flits at a link $q_{i,j}$ is given by

$$\frac{\left(2N_{i,j}+1\right)}{\beta_{i,j}} \times (\text{ps}_k - 1), \tag{7}$$

where ps_k denotes the number of flits in a packet.

A packet encounters this delay at that part of the path where the amount of buffering, when compared to the ideal value, is lowest.

Under zero load conditions, the latency constraint on a flow is met if the following constraint is satisfied:

$$\max_{\forall i,j,\ \text{s.t. } q_{i,j} \in \text{Path}_k} \left\{ \frac{\left(2N_{i,j}+1\right)}{\beta_{i,j}} \times (\text{ps}_k - 1) \right\} + H_k \leq \text{LC}_k, \tag{8}$$

where H_k denotes the hop count of the flow f_k. The first term on the left-hand side accounts for the maximum delay across the entire path due to the reduced buffering.

The problem of computing the minimum number of buffers required at the different ports to meet the bandwidth and latency constraints can be formulated as a *linear program* (LP) as follows:

$$\begin{gathered}
\min : \sum_{i=1}^{|P|}\sum_{j=1}^{|P|} \beta_{i,j}, \quad i \neq j, \\
\text{s.t.} \quad \beta_{i,j} \geq \left(2N_{i,j}+1\right) * U_{i,j}, \\
\max_{\forall i,j,\ \text{s.t. } q_{i,j} \in \text{Path}_k} \left\{ \frac{\left(2N_{i,j}+1\right)}{\beta_{i,j}} \times (\text{ps}_k - 1) \right\} + H_k \leq \text{LC}_k \\
\beta_{i,j} \leq \left(2N_{i,j}+1\right), \quad \beta_{i,j} \geq 0, \quad \forall i,j \in P.
\end{gathered} \tag{9}$$

The objective function to be minimized is the total buffering used in the switches. The bandwidth and latency constraints, obtained from (6) and (8), form the constraints of the LP. The formulation can be solved quickly and efficiently by any linear/convex program solver, such as the lp_solve [30]. Since the resulting buffer values by solving the LP can be fractional, we round up the value to the next integer. In fact, we could have formulated the above equations as an integer linear program (ILP), where we can force the buffer values to be integers. However, as solving the ILP formulation has exponential time complexity, it will be unfeasible to apply in practice. Hence, we use the heuristic of LP formulation with the rounding scheme.

6. Simulation-Based Buffer Sizing

After Phase 1, we perform simulation of the NoC using the computed buffer sizes and injecting packets to model the application communication patterns that are taken as inputs. The simulation-oriented approach is iterative, where buffers are added and simulations performed iteratively, until all the flows meet the bandwidth and latency requirements. To perform the sizing, we propose two strategies called as *uniform increment* and *flow-based increment*.

In the first strategy, buffers at all the ports are incremented iteratively by a small step. The buffer increment at a port depends on the burstiness of the flows and the number of flows contending for the same port. During simulations, the burst sizes of the flows at each port are tracked and the average burstiness of a flow is identified. The burstiness of flow f_k is denoted as B_k.

We use the following term to increment the buffers at a port connected to the link $q_{i,j}$ at each iteration:

$$\frac{\sum_{\forall f_k \text{s.t.} q_{i,j} \in \text{Path}_k} \alpha * B_k}{\max_{\forall f_k} B_k * |F|}, \tag{10}$$

where α is the parameter that is increased from 0 in small steps with each simulation iteration. Intuitively, the increment captures the burstiness of flows, scaled by the maximum burstiness of any flow of the application and the summation captures the number of contending flows, normalized to the total number of flows in the application. Thus, ports that have many contending flows, or flows with

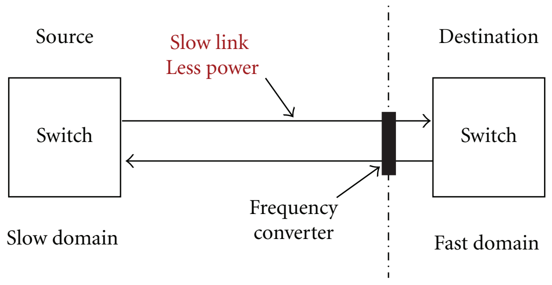


FIGURE 4: Placed closer to the fast clock domain.

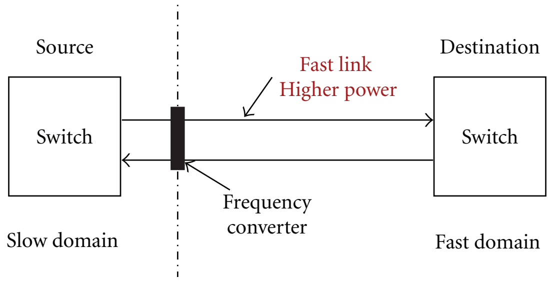


FIGURE 5: Placed closer to the slow clock domain.

large burst sizes get a larger increment at each iteration. The value of α is set experimentally. A very low value would result in lot more solutions being explored, while requiring more numbers of simulations.

In the second strategy, we track the flows that violate the bandwidth or latency constraints, and only the buffers along the path of such flows are incremented. This approach is an optimization of the previous strategy, giving finer control to access individual flows. For faster results, the *uniform increment* scheme can be used, while for better results, the *flow-based increment* scheme can be used. Thus, the two schemes allow a tradeoff between having fine control over the buffer allocation and minimizing the number of simulation runs required. In Section 9, the results of the two proposed strategies are discussed.

7. Placement of Frequency Converters

In this section the proposed buffer sizing scheme is extended to designs with multiple VF islands. We assume that the architecture characteristics include the VF island information.

FCs are used when crossing clock domains. Several kinds of FCs are proposed in the literature, but the most commonly used one is a dual clock FIFO (DCFIFO). A DCFIFO consists of a series of shift registers connected together that shift the input data from one domain to the output to the other domain. The input is clocked by one clock and output is clocked by the other, and the data is stored in the intermediate registers till the output is ready to consume it.

From the input architectural specification, the clock domains are identified and DCFIFOs are added along the links that cross the domains. Placement of the DCFIFOs is a critical design problem as they affect the power consumption and buffering required, hence, the performance of the interconnect.

7.1. Frequency Converters near Fast Domain.

In this setup, the frequency converters are placed close to the fast clock domain, as shown in Figure 4. By placing the converters near the fast clock, the link is clocked by the slower clock. From (2), it is evident that when the link operates at a lesser frequency, the latency increases. But, since the operating frequency is lesser, the power consumed by the link is lesser, as $P \propto f$. The increased latency of the link will demand the downstream router to have more buffering (according to Lemma 5).

7.2. Frequency Converters near Slow Domain.

In this setup, the FCs are placed closer to the slow clock domain, thereby clocking the link by the faster clock, as shown in Figure 5. This makes the link to operate at higher speed, and hence the latency is lesser (2), thereby reducing the buffering needed at the downstream router. But the higher operating frequency makes the link consumes more power.

This tradeoff between power consumed by the link and power consumed because of extra buffering can be explored to choose a specific design point that meets the system requirements. The effects of placement of FCs is analyzed, and the results are discussed in Section 9.3.

8. Handling Bubbles

In multiclock designs, inherently lot of empty flits are generated because of difference in the operating speed of the network components. Network flows traversing from a faster to slower frequency domain will incur higher latency, and enough buffering must be provided to meet the design constraints.

On the other hand, network flows traversing from a slower to a faster clock domain create *bubbles* in the network. Since destination is faster than the source, empty flits (bubbles) are generated in the network which underutilize the network resources. These bubbles must be reduced in order to better utilize the resources, and employing *pack and send* (PS) units can help in reducing them. Bubbles in a flow hold the buffers along the path for a long period unnecessarily, and hence other flows contending for the links are delayed. Waiting for the entire packet to arrive before the flow requests for the downstream buffer allows other flows contending for the link to proceed. Pack and send unit holds the flits temporarily till the entire packet is formed, and then it is forwarded to the downstream router. Hence a PS unit contains buffers to hold an entire packet. Section 9.4 discusses the results obtained when employing PS units.

9. Results

We consider a 26-core multimedia benchmark for a detailed study of the buffer-sizing algorithm. In Section 9.6, we show the application of the algorithm to a variety of benchmarks. The system includes ARM, DSPcores, memory banks, DMA engine, and several peripheral devices [31].

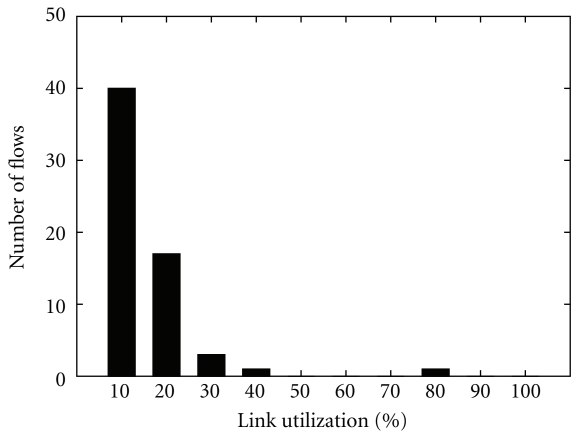


Figure 6: Histogram of link utilization.

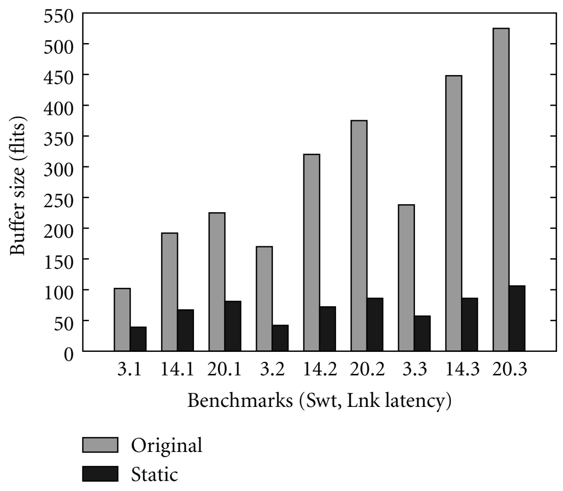


Figure 7: Comparison of buffering schemes.

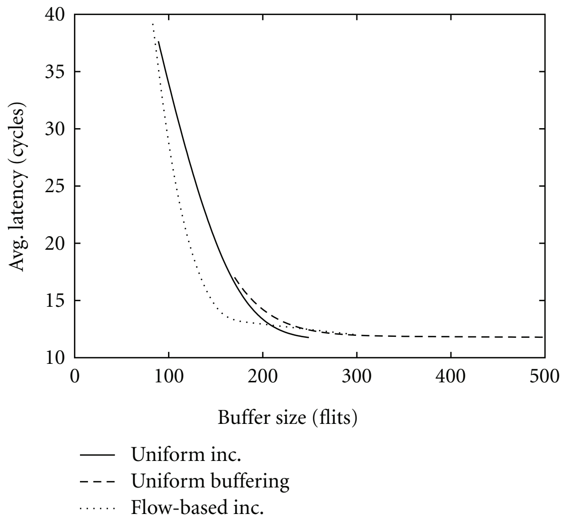


Figure 8: Latency versus buffering.

We consider benchmarks with custom topologies, but the algorithm is extendable to regular topologies too. Some of the benchmarks involve topologies with large number of switches (26 cores and 20 switches) which is similar to regular topologies. For the initial study to validate the buffer-sizing algorithm we consider no voltage and frequency domains for the network. The entire interconnect operates at a single frequency.

We use existing tools to synthesize NoC topologies with different number of switches [9]. The flit width of the NoC is set to 32 bits. For each topology synthesized, we perform simulations with very large (40 buffers at each port) buffer sizes and set the frequency of operation to a value where the application constraints are met during simulations. This ensures a working solution with very large buffering values, and the objective is to reduce it to much smaller values. We chose 3 different topologies with few to large number of switches (3, 14, and 20 switches) for the study. The number of cycles needed to traverse the links between the switches depend on the floor plan of the design, the technology library characteristics, and the operating frequencies of the components. For this study, we consider 3 different configurations for the link delay: 1, 2, and 3 cycles across all the links. This allows us to show the effect of link delays on the efficiency of the proposed methods. We denote the topology points by *the number of switches, link delay*.

9.1. Effects of Static Buffer Sizing. In the static sizing, we reduce the buffering at any port when the utilization at the port is less than 100%. We observed that in this and most other embedded benchmarks, the link utilization is very nonuniform, and only some bottleneck links have high utilization while others have much lower values. For example, in Figure 6, we show the utilization of the different links for a 20-switch benchmark. It can be seen that there are a lot of flows that have a very low utilization of less than 10%. Hence, utilization-based sizing can lead to a large reduction in buffering. Please note that, due to protocol overhead, we could not achieve 100% utilization on any link and needed to operate the network at a slightly higher frequency than the minimum needed to meet the bandwidth constraints.

We show the effect of static sizing in Figure 7. We compare the results with an original uniform buffering, where the minimum number of buffers for full utilization is used at all the ports. We can see that the proposed scheme results in large reduction in buffering requirements.

9.2. Effects of Simulation-Based Buffer Sizing. In this subsection, we compare the application of the two strategies, *uniform increment* and *flow-based increment*. For comparisons we also developed a standard *uniform buffering* strategy, where all ports have the same number of buffers. This is set to the minimum value at which the latency and bandwidth constraints of all the flows are met during simulations. We consider 3 different burst sizes for all the traffic flows: 4, 8, and 16 byte bursts.

Figure 8 shows the average latency across all the flows for a spectrum of buffer budgets for a benchmark with 3, 3 design and a burst size of 16 for all the flows. Depending on

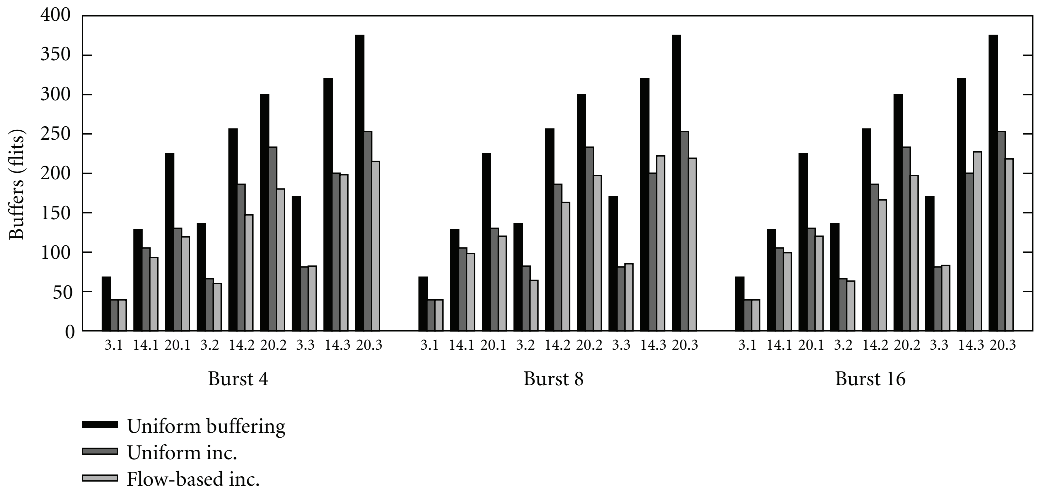


Figure 9: Comparison of different buffer-sizing strategies.

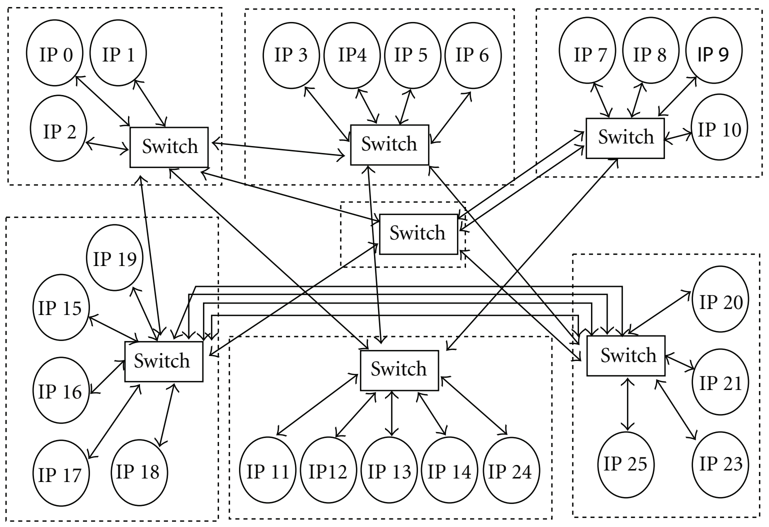


Figure 10: Example topology.

the tightness of the average latency constraint, the amount of buffering achieved by the different schemes varies. When loose latency constraint is used, the proposed strategies can provide large reduction in buffering requirements. For a very tight average latency constraint, all schemes perform similarly and the simulation-based sizing methods do not give any savings. Depending on the constraint, the algorithm can give the best buffer setting.

We set the average latency constraint to 50 cycles and perform buffer sizing. The buffer budget in the graphs (Figure 9) denotes the minimum buffer budget at which the simulation is stable and all constraints are met. We find that both *uniform increment* and *flow-based increment* strategies perform significantly better when compared to the standard *uniform buffering* strategy. The minimum buffer budget in the case of *uniform increment* is higher than *flow-based increment* in most cases, because the increment to the buffers is uniform on all the ports, and thus the addition of buffers is at a coarse level. Moreover, as expected, the savings are more pronounced when the traffic is more bursty and/or the link delays are larger. The results show that there is a 42% reduction in the buffer budgets for the switches, on an average. Based on the models from [6], this translates to around 35% reduction in the overall power consumption of the NoC switches.

9.3. Effect of Placement of FCs. We implemented the above buffer-sizing algorithm to a benchmark with multiple clock domains. The benchmark consisted of the entire topology clustered into different VF islands with operating frequencies ranging from 200 to 600 MHz. An example input topology is illustrated in Figure 10. One of the above proposed strategies, *uniform increment,* was used in this study. The main goal was to analyze the impact of placement of the FCs. Figure 11 shows the reduction in the buffer budgets compared to the standard uniform buffering scheme. Also the results show

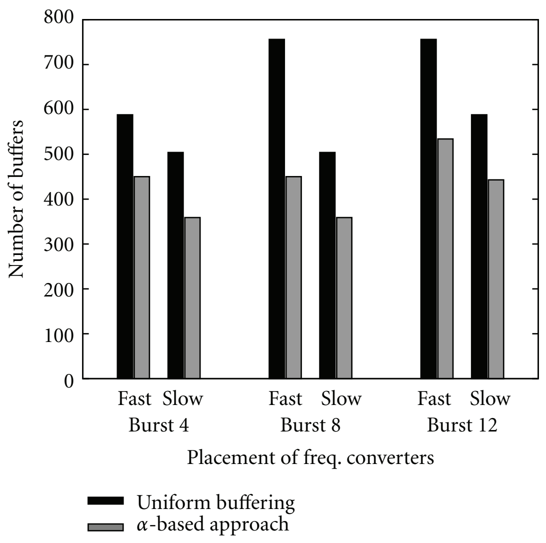


Figure 11: Placements of FCs.

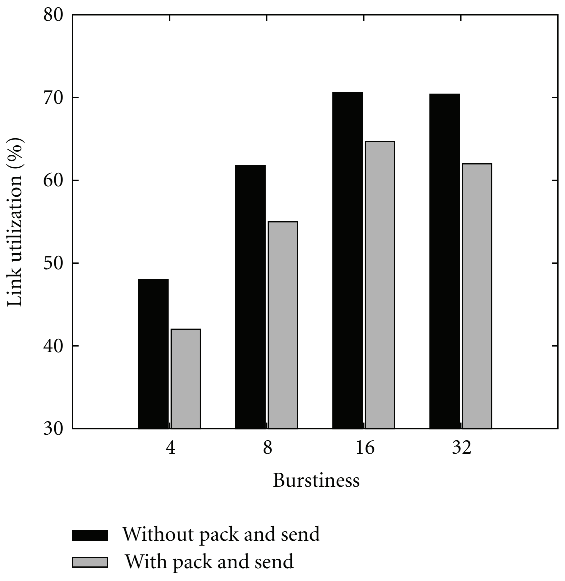


Figure 12: Pack and send.

that the buffering required when placing the FCs closer to the slow domain is lesser than the buffering required when placing it close to fast domain. This however affects the link power, as the links are clocked higher and hence the link power is more. With smaller transistor sizes, the link power is as significant as the logic power, and hence the power consumed by the links must also be taken into account.

9.4. Impact of Pack and Send. For studying the effect of pack and send unit, the benchmark with multiple clock domains was considered. Since the pack and send unit has an effect only across links where the utilization is high, we scaled the frequencies to a range of 500–900 MHz and the burstiness was increased. For the study the proposed buffer-sizing algorithm was used with PS units placed along links going from slow to fast clock domain. The effect on buffering with and without pack and send unit was analyzed. Though the PS units require extra buffers to hold a packet, results showed that the total buffering required (including the PS buffers) remains the same. However the link utilization reduces when using PS units. Since there is lesser contention among the flows while using PS units, there is a direct impact in terms of reduction in the buffering required at the switches. However, the total buffering required remains the same as the decrease in buffering at the switches is compensated by the extra buffers required in the PS units. Figure 12 shows the overall average link utilization with and without PS unit. This shows that the PS unit helps in better utilizing the resources, and this reduction in link utilization can be directly converted to power savings at lower technology nodes.

9.5. Run Time of the Methods. Since Phase 1 uses LP formulation and not ILP, the solution is tractable and is obtained quickly. Phase 1 of the algorithm finished in few seconds for all the benchmarks on a 2.66 GHz Linux Machine. Among the two strategies presented in Phase 2, there is a tradeoff between the running time of simulations and the granularity of controlling buffer sizes. Previous sections show that the fine control of buffer sizes for *flow based increment* approach helps in achieving a lower budget. But the running time to converge to a budget is more. Each simulation run can take from 20 minutes to few hours, depending on how long a trace needs to be simulated. The *uniform increment* approach took 20–60 simulation runs, while the *flow-based increment* approach took 50–100 runs. Thus, the design time required could be significantly higher for the *flow-based increment* strategy. The designer has the choice of the strategy and the step used for the α values and can make a tradeoff between buffer reduction and design time.

9.6. Experiments on Other Benchmarks. To show the generality of the method, we consider 4 other SoC benchmarks: *D_36_4, D_36_6, D_36_8, and D_35*. The first 3 benchmarks have 36 cores, with each communicating to 4, 6, and 8 other cores respectively. The last benchmark has 35 cores and models bottleneck traffic communication, such as memory controller traffic. On average, the proposed schemes result in 38% reduction in buffer sizes when compared to the standard uniform sizing schemes.

9.7. Comparison with Theoretical Models. To show that the proposed buffer-sizing methodology works for well-behaved traffic also, we compare our results with queuing theory-based model proposed in [10]. Figure 13 shows that the proposed model is able to achieve buffer budgets close to the theoretical limit proposed in the paper.

10. Conclusion

As buffers account for a large area, power overhead in NoCs, reducing the amount of buffering is an important problem. Buffer sizing is closely tied to dynamic congestion effects that can be observed only during simulations. Towards this end,

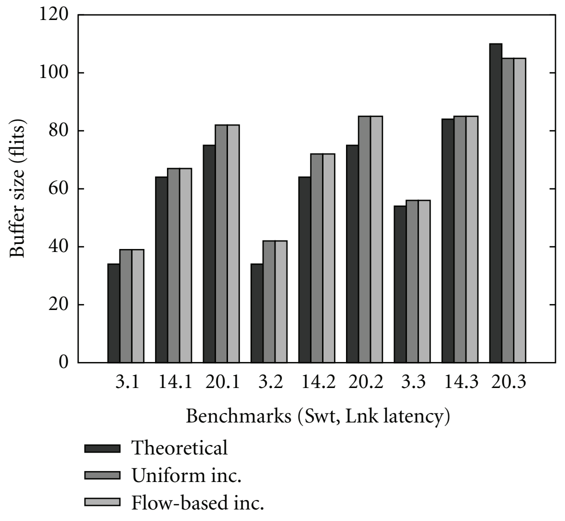


FIGURE 13: Comparison with theoretical models.

in this paper, we present a two-phase algorithm for buffer sizing. Our approach considers the nonuniformity in the utilization of the different parts of the network and uses a simulation-based iterative mechanism. Our results show a large reduction in buffering required (42%) when compared to standard buffering approach. The proposed buffer-sizing algorithm was extended to designs with multiple clock domains. Results showed a significant reduction in buffer budget. We also analyzed the effect of placement of FCs on the overall buffer budget. Also the use of PS units to handle bubbles in the network was studied, and results showed that employing them utilize resources better.

Future Work

In the proposed buffer-sizing approach, the second phase that involves simulations is the step that consumes a significant portion of the total run time. We can use intuitive approaches to reduce the simulation time for that step. But this is beyond the scope of this work and can be a part of an extension to this work.

Acknowledgments

The authors would like to acknowledge the ARTIST-DESIGN Network of Excellence. This work has also been supported by the project *NaNoC* (project label 248972) which is (partly) funded by the European Commission within the Research Programme FP7.

References

[1] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[2] G. D. Micheli and L. Benini, *Networks on Chips: Technology and Tools*, Morgan Kaufmann, 2006.

[3] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet switched interconnections," in *Proceedings of the Design, Automation and Test in Europe*, pp. 250–256, 2000.

[4] R. W. Apperson, Z. Yu, M. J. Meeuwsen, T. Mohsenin, and B. M. Baas, "A scalable dual-clock FIFO for data transfers between arbitrary and haltable clock domains," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 15, no. 10, pp. 1125–1134, 2007.

[5] A. Strano, D. Ludovici, and D. Bertozzi, "A library of dualclock fifos for cost-effective and flexible mpsoc design," in *Proceedings of the International Conference on Embedded Computer Systems (SAMOS '10)*, pp. 20–27, 2010.

[6] S. Murali, T. Theocharides, N. Vijaykrishnan, M. J. Irwin, L. Benini, and G. De Micheli, "Analysis of error recovery schemes for networks on chips," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 434–442, 2005.

[7] J. Hu and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," in *Proceedings of the Design, Automation and Test in Europe*, 2004.

[8] A. Hansson et al., "A unified approach to mapping and routing in a combined guaranteed service and best-effort Network-on-Chip architecture," in *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, 2005.

[9] S. Murali, P. Meloni, F. Angiolini et al., "Designing application-specific networks on chips with floorplan information," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD '06)*, pp. 355–362, November 2006.

[10] J. Hu and R. Marculescu, "Application-specific buffer space allocation for networks-on-chip router design," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers (ICCAD '04)*, pp. 354–361, November 2004.

[11] Y. Yin and S. Chen, "An application-specific buffer allocation algorithm for network-on-chip," in *Proceedings of the 8th IEEE International Conference on ASIC (ASICON '09)*, pp. 439–442, October 2009.

[12] W. H. Ho and T. M. Pinkston, "A methodology for designing efficient On-Chip interconnects on well-behaved communication patterns," in *Proceedings of the International Symposium on High Performance Computer Architecture*, 2003.

[13] J. Kong, J. Choi, L. Choi, and S. W. Chung, "Low-cost application-aware DVFS for multi-core architecture," in *Proceedings of the 3rd International Conference on Convergence and Hybrid Information Technology (ICCIT '08)*, pp. 106–111, November 2008.

[14] T. Kolpe, A. Zhai, and S. Sapatnekar, "Enabling improved power management in multicore processors through clustered dvfs," in *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE '11)*, pp. 1–6, March 2011.

[15] S. Garg, D. Marculescu, R. Marculescu, and U. Ogras, "Technology-driven limits on DVFS controllability of multiple voltage-frequency island designs: a system-level perspective," in *Proceedings of the 46th ACM/IEEE Design Automation Conference (DAC '09)*, pp. 818–821, July 2009.

[16] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli, "NoC topology synthesis for supporting shutdown of voltage islands in SoCs," in *Proceedings of the 46th ACM/IEEE Design Automation Conference (DAC '09)*, pp. 822–825, July 2009.

[17] L. Guang, E. Nigussie, and H. Tenhunen, "Run-time communication bypassing for energy-efficient, low-latency per-core dvfs on network-on-chip," in *Proceedings of the SOC Conference*, pp. 481–486, September 2010.

[18] T. Jain, P. Gratz, A. Sprintson, and G. Choi, "Asynchronous bypass channels: improving performance for multisynchronous nocs," in *Proceedings of the 4th ACM/IEEE International Symposium on Networks-on-Chip (NOCS '10)*, pp. 51–58, May 2010.

[19] W. Liwei, C. Yang, L. Xiaohui, and Z. Xiaohu, "Application specific buffer allocation for wormhole routing Networkson-Chip," *Network on Chip Architectures*, pp. 37–42, 2008.

[20] M. A. Al Faruque and J. Henkel, "Minimizing virtual channel buffer for routers in on-chip communication architectures," in *Proceedings of the Design, Automation and Test in Europe (DATE '08)*, pp. 1238–1243, March 2008.

[21] S. Yin, L. Liu, and S. Wei, "Optimizing buffer usage for networks-on-chip design," in *Proceedings of the International Conference on Communications, Circuits and Systems (ICCCAS '09)*, pp. 981–985, July 2009.

[22] N. Alzeidi, M. Ould-Khaoua, L. M. Mackenzie, and A. Khonsari, "Performance analysis of adaptively-routed wormhole-switched networks with finite buffers," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 38–43, June 2007.

[23] S. Foroutan, Y. Thonnart, R. Hersemeule, and A. Jerraya, "An analytical method for evaluating network-on-chip performance," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '10)*, pp. 1629–1632, March 2010.

[24] S. Manolache, P. Eles, and Z. Peng, "Buffer space optimisation with communication synthesis and traffic shaping for NoCs," in *Proceedings of the Design, Automation and Test in Europe (DATE '06)*, pp. 718–723, March 2006.

[25] M. Coenen, S. Murali, A. Ruadulescu, K. Goossens, and G. De Micheli, "A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control," in *Proceedings of the 4th International Conference on Hardware Software Codesign and System Synthesis*, pp. 130–135, October 2006.

[26] A. B. Kahng, B. Lin, K. Samadi, and R. S. Ramanujam, "Trace-driven optimization of networks-on-chip configurations," in *Proceedings of the 47th Design Automation Conference (DAC '10)*, pp. 437–442, June 2010.

[27] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. De Micheli, "A methodology for mapping multiple use-cases onto networks on chips," in *Proceedings of the Design, Automation and Test in Europe (DATE '06)*, pp. 118–123, March 2006.

[28] E. Beigne and P. Vivet, "Design of On-chip and Off-chip interfaces for a GALS NoC architecture," in *Proceedings of the International Symposium on Asynchronous Circuits and Systems (ASYNC '06)*, pp. 172–183, IEEE Computer Society, Washington, DC, USA, 2006.

[29] W. J. Dally and B. Towels, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, 2003.

[30] Lp solve, http://lpsolve.sourceforge.net/.

[31] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli, "SunFloor 3D: a tool for networks on chip topology synthesis for 3D systems on chips," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '09)*, pp. 9–14, April 2009.