

Power-efficient Interconnection Networks: Dynamic Voltage Scaling with Links

Li Shang, Li-Shiuan Peh, and Niraj K. Jha

Department of Electrical Engineering, Princeton University, Princeton, NJ 08544
{lshang,peh,jha}@ee.princeton.edu

Abstract— Power consumption is a key issue in high-performance interconnection network design. Communication links, already a significant consumer of power now, will take up an ever larger portion of the power budget as demand for network bandwidth increases. In this paper, we motivate the use of dynamic voltage scaling (DVS) for links, where the frequency and voltage of links are dynamically adjusted to minimize power consumption. We propose a history-based DVS algorithm that judiciously adjusts DVS policies based on past link utilization. Despite very conservative assumptions about DVS link characteristics, our approach realizes up to 4.3X power savings (3.2X average), with just an average 27.4% latency increase and 2.5% throughput reduction. To the best of our knowledge, this is the first study that targets dynamic power optimization of interconnection networks.

Keywords— Dynamic voltage scaling, interconnection network, power optimization.

I. INTRODUCTION

IN recent years, interconnection network fabrics, historically used in high-end multiprocessor systems [1], have been deployed and proposed for a wide range of communication systems – clusters [2], terabit Internet routers [3], server blades [4], and on-chip networks [5]. These myriad applications place new demands on the network fabric. In the past, a low-latency and high-throughput fabric was the ultimate goal. This is no longer sufficient. Applications of interconnection networks are becoming power-limited.

Interconnection networks are a significant consumer of power in a system. The integrated router and links of the Alpha 21364 processor [1] consumes 25W, about 20% of maximum system power of 125W [6]. Since high-speed links consume approximately the same amount of power on average and at peak, this percentage will be even more significant when compared to average system power. In a terabit Internet router, the power consumed by the interconnection network circuitry is about a third that dissipated by the entire line card [3]. This will only worsen as the demand for network bandwidth increases. Off-chip memory bandwidth has been increasing fast in multiprocessor systems – While Alpha 21164 had just 1.2GB/s bus bandwidth to off-chip memory, the newest Alpha 21364 router provides a whopping 22.4GB/s raw network bandwidth. In Internet routers, a similar increasing trend of network bandwidth is observed, with port bandwidths scaling from OC48 (2.5Gb/s) to OC768 (40Gb/s).

Dynamic voltage scaling (DVS) has been proposed and deployed for microprocessors [7], [8], uncovering significant power savings [9]. Recently, variable-frequency links have been proposed [10], [11]. These links track and adjust their voltage level to the minimum supply voltage as link frequency is changed, lowering power dissipation by 10X potentially. Variable-frequency links demonstrate the feasibility of DVS links. This opens up exciting avenues for interconnection network designers.

Like microprocessors, there is a wide variance in link utilization in interconnection networks, depending on applications' communication patterns. This can lead to huge power savings if network bandwidth is tuned accurately to track usage. In this paper, we first discuss the components of a DVS link, and link characteristics that are critical to DVS (Section II). Next, we propose a simple history-based DVS method (Section III). Its performance is then evaluated and presented in Section IV.

Researchers have investigated the power consumption of interconnection networks, but the focus has been on power modeling – for chip-to-chip interconnection networks [12] and on-chip switchbox networks [13].

II. DVS LINK ARCHITECTURE

Wei and Kim looked into voltage scaling for parallel [10] and serial links [11], where the link automatically tunes to the minimal supply voltage as link frequency varies. The main components of a DVS link are shown in Fig. 1. It consists of components of a typical high-speed link: a transmitter to convert digital binary signals into electrical signals; a signaling channel usually modeled as a transmission line; a receiver to convert electrical signals back to digital data; and a clock recovery block to compensate for delay through the signaling channel. In addition, a DVS link needs an adaptive power-supply regulator that tracks link frequency and regulates voltage to the minimum level required, and feeds regulated supply voltage to multiple links (say, the links of a network channel), amortizing its area and power costs. Finally, a frequency synthesizer feeds the user-controlled frequency to the power-supply regulator.

The DVS serial link was fabricated in 0.25- μ m CMOS technology. The total power dissipated by each link varies from 21mW at 1Gb/s to 197mW at 3.5Gb/s. Hence, a 10X power savings can potentially be achieved.

The important characteristics of a DVS link are (i) transition time - how long it takes for a link's voltage level to change from voltage level V_1 to V_2 , (ii) transition energy -

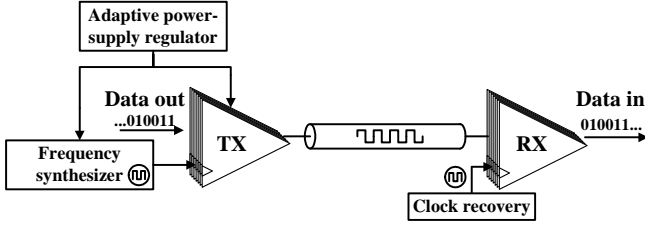


Fig. 1. Components of a DVS link

the energy consumed for a transition from V_1 to V_2 , (iii) transition status - whether the link functions during a transition, and (iv) transition step - whether the link supports a continuous range of voltages, or if it only supports a fixed number of voltage levels.

We are in the process of exploring realistic assumptions for the above parameters with the link designers. Initial findings confirm that a DVS link will be able to send bits during voltage transition, within a range, and that very fast transition times coupled with low transition energy can be realized by replacing the adaptive power-supply regulator with a multi-voltage supply (multiple levels of voltages instead of a continuous range). However, as a first study, we wish to explore the potential of DVS links by making very conservative assumptions that are confirmed feasible for current link circuitry. Hence, we assume continuous voltage transition, at a slow rate of $0.1V/\mu s$ (as compared to the $2V/\mu s$ transition time of Berkeley’s lpArm micro-processor [7]), and that the link does not function during a transition. Transition energy is derived based on Stratakos’s analysis [14], where dynamic energy overhead when voltage transitions from V_1 to V_2 is calculated with the following first-order estimation equation.

$$Energy_{overhead} = C * (1 - u) * |V_2^2 - V_1^2| \quad (1)$$

where C is the filter capacitance of the power-supply regulator and u is the power efficiency. In our experiments, we assume $5\mu F$ capacitance and 90% power efficiency [11].

III. HISTORY-BASED DVS

We propose a history-based DVS algorithm, where each router predicts future communication traffic based on past link utilization, then dynamically adjusts the link frequency (and corresponding voltage) to track the predicted usage. We characterize link utilization, U_{link_i} , as follows.

$$U_{link_i} = \frac{\sum_{t=1}^H A(t)}{H} \quad (2)$$

where $A(t) = \begin{cases} 1 & \text{if traffic passes link } i \text{ in cycle } t \\ 0 & \text{if no traffic passes link } i \text{ in cycle } t \end{cases}$ and H is the history window size.

In predicting future link utilization, our history-based DVS algorithm uses exponential weighted average utilization to combine both short-term ($U_{short-term}$) and long-term ($U_{long-term}$) utilization history. This allows the scheduler to filter out transient fluctuations and adapt to real traffic trends.

Given the predicted link utilization, $U_{predicted}$, the DVS algorithm judiciously adjusts the voltage scaling

policies to achieve power savings with minimal impact on performance. First, it prescribes whether to increase link speed ($VS_{trend}=increase$), decrease link speed ($VS_{trend}=decrease$), or do nothing ($VS_{trend}=static$). It also prescribes how often voltage scaling should be carried out, whether frequently (VS_{fast}) or less frequently (VS_{slow}). For instance, if VS_{slow} is 32, we carry out voltage scaling once every 32 history intervals. The prescribed action depends on four thresholds ($T_{highest}$, T_{high} , T_{low} , T_{lowest}). Intuitively, when a link is going to be highly utilized, voltage scaling is enabled more frequently so that link frequency can be quickly increased to handle the load. Similarly, if a link will be mostly idle, voltage scaling is carried out more often so that link frequency can drop rapidly to save power. Otherwise, voltage scaling is conservatively carried out to minimize impact on performance. The pseudo-code of our proposed DVS algorithm is shown in Algorithm 1.

Algorithm 1 History-based Dynamic Voltage Scaling

```

while ( $Signal_{VS_{fast}}$  or  $Signal_{VS_{slow}}$ ) do
   $U_{predicted} = (W * U_{short-term} + U_{long-term}) / (W + 1)$ 
   $U_{long-term} = U_{predicted}$ 
  if ( $U_{predicted} < T_{low}$ ) then
     $VS_{trend} = decrease$ 
    if ( $U_{predicted} < T_{lowest}$ ) then
      trigger voltage scaling by  $Signal_{VS_{fast}}$ 
    else
      trigger voltage scaling by  $Signal_{VS_{slow}}$ 
    end if
  else if ( $U_{predicted} > T_{high}$ ) then
     $VS_{trend} = increase$ 
    if ( $U_{predicted} > T_{highest}$ ) then
      trigger voltage scaling by  $Signal_{VS_{fast}}$ 
    else
      trigger voltage scaling by  $Signal_{VS_{slow}}$ 
    end if
  else
     $VS_{trend} = static$ 
  end if
end while

```

The proposed DVS algorithm relies only on local link information. This avoids communication overhead in relaying global information, and permits simple hardware implementation of the algorithm. At each output port, two counters gather link utilization information, one for $U_{short-term}$, another for $U_{long-term}$. The $\log_2 H$ -bit counters are triggered by the link clock. We set W to $2^k - 1$, where k is a constant, so that division can be implemented as a shift operation. A special adder is designed for the $W * U_{short-term} + U_{long-term}$ operation and some combinational logic for threshold comparisons. Logic synthesis of the proposed circuitry shows that it consumes area equivalent to that of 250 logic gates. Since it does not lie on the critical path of the router, the delay can be ignored. We have yet to quantify the power overhead of the circuitry, but believe it to be negligible given its simplicity.

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

We implemented a flit-level interconnection network simulator for evaluating the proposed history-based DVS algorithm. The simulator supports k -ary n -cube network

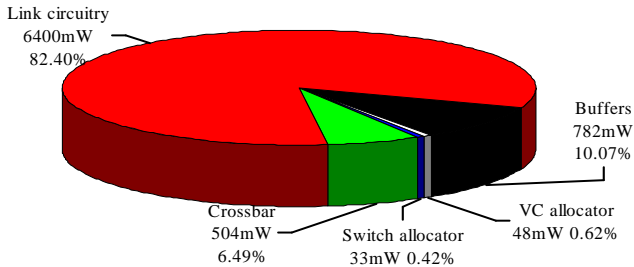


Fig. 2. Router power consumption distribution

topologies consisting of pipelined virtual-channel (VC) routers [15] and channels comprising DVS links. Different routing algorithms, both deterministic and adaptive, are supported, with credit-based flow control. The simulator has been implemented in C++ (5,200 lines of code) based on standard template libraries (STL), running under Linux OS.

In our experiments, we assumed a two-dimensional 8x8 mesh network topology, consisting of 64 1GHz routers, each with two virtual channels and 128 flit buffers per input port. The router is pipelined according to [16], with four pipeline stages: routing, virtual channel allocation, switch allocation, crossbar traversal. Fixed-length packets of five flits are assumed, with a head flit leading four body flits, and each flit being 32-bits wide.

Each port of the router has 32Gb/s channel bandwidth so that a 32-bit flit can be transported across the channel every cycle¹. The channel consists of eight serial links, each link's frequency (voltage, power) can be scaled from 125MHz (0.9V, 23.6mW) to 1GHz (2.5V, 200mW), with a corresponding bandwidth of 0.5Gb/s to 4Gb/s with 4:1 multiplexing.

We synthesized the Verilog description of our router to a gate-level netlist in 0.25 μ m technology to characterize its power consumption. Combined with real-delay timing simulation, we used Synopsys Power Compiler, a gate-level power estimator, to estimate the power dissipated by various parts of the router. The profile is shown in Fig. 2, based on maximum power consumed by a channel of eight links [11]. In our router, 82.4% of the total power is consumed by the link circuitry. In Alpha 21364, the ratio is about 63.6%. The percentage of power consumed by the links versus nodes depends largely on the amount of buffering in a router, relative to channel bandwidth. The main purpose of our power characterization is to determine the additional power consumed as a result of a flit staying longer in a router due to slower links. A flit that stays in a router can potentially trigger more arbitrations. However, it does not increase buffer read/write power, nor crossbar power. Since the allocators consume minimal power (81mW), we can ignore router power consumption when evaluating our history-based DVS algorithm.

The experimental parameters of our history-based DVS algorithm are given in Table I. Latency, throughput and

¹We ignore the encoding overhead which will require much higher channel bandwidth, say 40Gb/s for 8b/10b encoding.

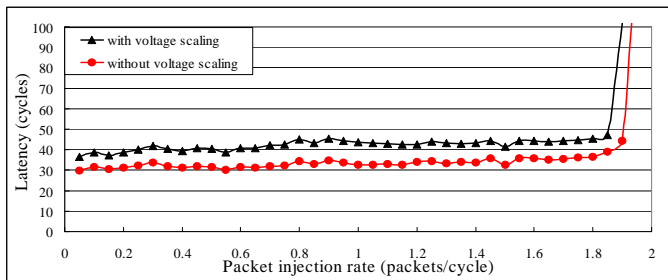


Fig. 3. Latency-throughput performance with and without history-based DVS for 100 tasks

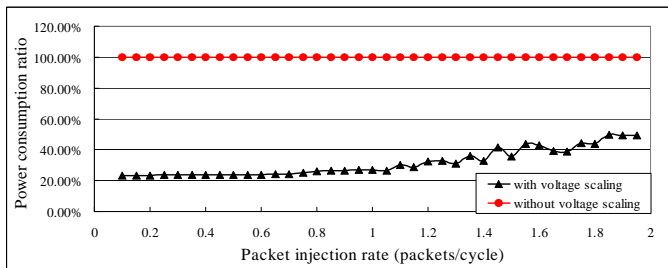


Fig. 4. Power consumption with and without history-based DVS for 100 tasks

power consumption are the metrics we use to evaluate our technique. Latency spans the creation of the first flit of the packet to ejection of its last flit at the destination router, including source queuing time and assuming immediate ejection. Each simulation is run for 10 million cycles, and the latency of all packets averaged. The saturation throughput of the network is where average packet latency worsens to more than twice the zero-load latency (*i.e.*, the average packet latency when there is no network congestion). Power consumed by the network is derived based on the frequency and voltage levels set for all the channels in the network. Hence, without DVS links, the power consumed by a network is 64 routers * 4 ports * 8 links * 0.2W = 409.6W. Router power is ignored since it stays relatively constant with or without DVS links.

TABLE I
PARAMETERS IN THE HISTORY-BASED DVS ALGORITHM

$V_{S_{fast}}$	$V_{S_{slow}}$	W	H	T_{lowest}	T_{low}	T_{high}	$T_{highest}$
16	32	3	50	0.1	0.3	0.4	0.9

B. Performance of History-based DVS Algorithm

We model network communication traffic with a task workload, where communication tasks are generated at random nodes, with an average length of 1ms. Each task injects packets at a rate governed by Poisson distribution, with random uniformly distributed destinations. This task workload allows us to approximate parallel task communication through an interconnection network, and is widely used to model workloads in distributed embedded systems. Prior workloads commonly used in interconnection networks are unsuitable. Random uniformly distributed traffic does not exhibit any spatial or temporal variance, other than that brought about by the topology; and while permutation traffic stresses routing algorithms with spatial vari-

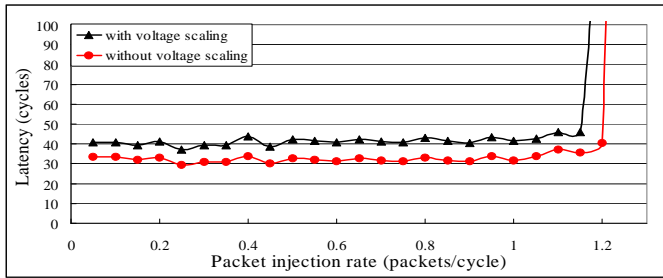


Fig. 5. Latency-throughput performance with and without history-based DVS for 50 tasks

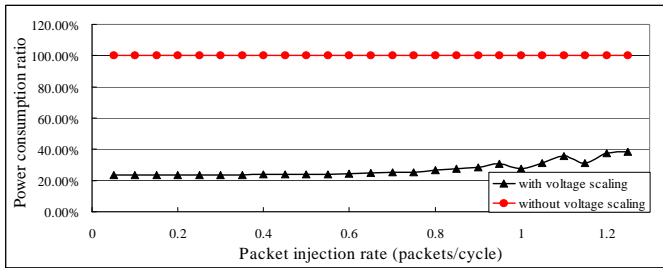


Fig. 6. Power consumption with and without history-based DVS for 50 tasks

ance in load, it does not capture any temporal variance.

We evaluated the latency-throughput performance of our history-based DVS algorithm with 50 and 100 tasks running concurrently in every cycle. Fig. 3 shows the performance at increasing packet injection rates while Fig. 4 plots the power consumed by the network with and without DVS. With 100 tasks, history-based DVS increases zero-load latency by 22.8% (and 27.4% on average before congestion), while decreasing throughput by 2.5%. This moderate impact on performance is accompanied by a large power saving of up to 4.3X (3.2X average). Even when the network is congested, history-based DVS can still realize a 2.0X power saving.

Similarly, with 50 tasks, history-based DVS has a moderate impact on performance - increasing latency by 27.5% on average before congestion, and lowering throughput by 4.7%, while realizing up to 4.3X power saving (see Figs. 5 and 6). The lower throughput as compared to the 100-task workload is due to a higher imbalance in traffic.

A snapshot of a DVS link in the network is shown in Fig. 7. It shows history-based DVS successfully adjusting link frequency to track actual link utilization over time. By dynamically adjusting its policies to maximize power savings when the network is lightly loaded, and minimizing performance impact when the network is congested, history-based DVS is able to realize substantial power savings without significant impact on performance. It should be noted that part of the impact on performance is due to our conservative assumptions for DVS links, i.e., that the link is down during voltage scaling, and that voltage transitions very slowly.

V. CONCLUSIONS

We presented a history-based DVS algorithm to minimize power consumption in high-performance interconnection networks. Our algorithm judiciously adjusts the DVS

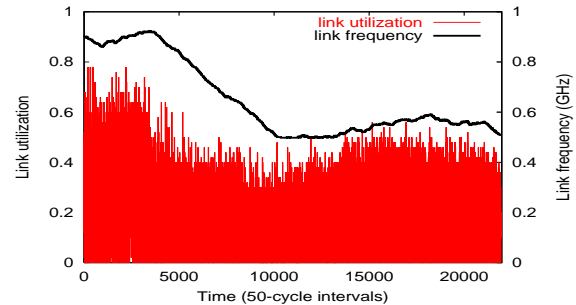


Fig. 7. A link in the network, showing scaled link frequency tracking actual utilization.

policies to strike a good balance between power saving and latency-throughput penalty. This algorithm is distributed, which avoids communication overhead in relaying global information, and permits a simple hardware implementation.

ACKNOWLEDGMENTS

The authors would like to thank Jaeha Kim of Stanford for his help in our understanding of his variable-frequency link, and for validating and extracting the link characteristics for dynamic voltage scaling. We also wish to thank Edward Lee of Velio for insightful technical discussions. This work was supported in part by DARPA under contract no. DAAB07-00-C-L516.

REFERENCES

- [1] S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb, "The Alpha 21364 network architecture," in *Proc. Hot Interconnects 9*, Aug. 2001.
- [2] N. Boden *et al.*, "Myrinet - a Gigabit-per-second local-area network," *IEEE Micro*, vol. 15, no. 1, pp. 29-36, Feb. 1995.
- [3] W. Dally, P. Carvey, and L. Dennison, "The Avici Terabit switch/router," in *Proc. Hot Interconnects 6*, Aug. 1998.
- [4] The Infiniband architecture, <http://www.infinibanda.org>.
- [5] W. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. Design Automation Conf.*, pp. 684-689, June 2001.
- [6] A. Jain *et al.*, "A 1.2 GHz Alpha microprocessor with 44.8GB/s chip pin bandwidth," in *Proc. Int. Solid-State Circuits Conf.*, pp. 240-241, Feb. 2001.
- [7] T. Burd and R. Brodersen, "Design issues for dynamic voltage scaling," in *Proc. Int. Symp. Low-Power Electronic Design*, pp. 9-14, July 2000.
- [8] Intel XScale microarchitecture, <http://developer.intel.com/design/intelxscale/>.
- [9] N. Jha, "Low power system scheduling and synthesis," embedded tutorial, in *Proc. Int. Conf. Computer-Aided Design*, pp. 259-263, Nov. 2001.
- [10] G. Wei, J. Kim, D. Liu, S. Sidiropoulos, and M. Horowitz, "A variable-frequency parallel I/O interface with adaptive power-supply regulation," *J. Solid-State Circuits*, vol. 35, no. 11, pp. 1600-1610, Nov. 2000.
- [11] J. Kim and M. Horowitz, "Adaptive supply serial links with sub-1V operation and per-pin clock recovery," in *Proc. Int. Solid-State Circuits Conf.*, Feb. 2002.
- [12] C. Patel, S. Chai, S. Yalamanchili, and D. Schimmel, "Power-constrained design of multiprocessor interconnection networks," in *Proc. Int. Conf. Computer Design*, pp. 408-416, Oct. 1997.
- [13] H. Zhang, M. Wan, V. George and J. Rabaey, "Interconnect architecture exploration for low-energy reconfigurable single-chip DSPs," in *Proc. Workshop VLSI*, pp. 2-8, Apr. 1999.
- [14] A. Stratakos, "High-efficiency low-voltage DC-DC conversion for portable applications," Ph.D. Thesis, Univ. of California, Berkeley, June 1998.
- [15] W. Dally, "Virtual-channel flow control," *IEEE Trans. Parallel & Distributed Systems*, vol. 3, no. 2, pp. 194-205, Mar. 1992.
- [16] L. Peh and W. Dally, "A delay model and speculative architecture for pipelined routers," in *Proc. High-Performance Computer Architecture*, pp. 255-266, Jan. 2001.