Power Control of High Speed Network Interconnects in Data Centers

Krishna Kant
Intel Research

***Abstract*—With continuing speed increases of communication links found both inside and outside the servers in a data center, link power management is becoming an important issue. In this paper we examine width control of such links and show that it provides substantial benefits over other methods for latency sensitive applications. The paper also presents the design of a specific width control algorithm called DWCA (dynamic width control algorithm) that is effective and easily implemented in hardware at high data rates.**

Keywords: Serial links, power management, link width modulation, exponential smoothing, latency, performance

I. Introduction

The communication links deployed in data centers have continued their march towards higher and higher speeds in tune with the speed of other elements. This, coupled with the proliferation of links both inside and outside a server has made link power management an important issue. Although the topic of link power control has been dealt with in the literature (See related work), much of the work relates to shutting down inactive links at a relatively low rate. In this paper, we consider the problem of dynamic power control that attempts to squeeze out power savings for relatively active links without introducing substantial performance impact. In particular, we consider dynamic power control of emerging bit-serial ultra high speed links intended to carry server to server traffic in a data center. This traffic is likely to be generated by a variety of applications, some of which could be highly latency sensitive. Consequently, we examine not only the power savings but also the power-latency tradeoffs closely.

To put the communication link power consumption in perspective, note that while the power consumption of a 1 Gb/s Ethernet is only 1-2 watts, the power consumption of 10 Gb/s Ethernet can easily exceed 10 watts. In the next several years, Ethernet speeds are expected to scale up to 40 or 100 Gb/s, thereby making the power consumption of Ethernet interface (and associated switching infrastructure) very substantial. At the same time, the *average* link utilization – already quite low – is only expected to go down as the link speeds increase. This coupled with the fact that the link power consumption is relatively insensitive to traffic intensity implies that without aggressive power control links, will burn near 100% of their rated power while doing very little.

Although Ethernet remains the dominant networking fabric in data centers, it is by no means the only one to focus upon. Infiniband has made substantial strides in the data center and is already available at speeds of 48 Gb/s [5]. Since Infiniband was designed from ground up to as a data center networking fabric, it is better suited than the traditional TCP/IP/Ethernet for high performance applications [8]. PCI-Express (or PCI-E) was originally designed as a replacement for the aging "inside-the-box" PCI interconnect [14] and has become universal in that space. Extensions of PCI-E with capabilities such as multi-root addressing and peer-to-peer communications make it useful as an "outside-the-box" fabric as well.[1] Fiber Channel (FC) remains a dominant fabric for storage in data centers and has already surpassed 10 Gb/s [11]. There are also other niche fabrics such as Myrinet and QsNet that continue to scale up in speed and power.

In general, communication links can support three ways of doing power control: (1) use of one or more (non-operational) low power states, (2) link width control, where only a portion of the link is put into a low-power mode, and (3) multiple operational speeds. In this paper, we focus primarily on (2) and show that for emerging high-speed links, this technique can achieve the most power savings with the smallest performance impact. This focus is justified since all of these fabrics mentioned above have already moved towards the bit-serial technology for which width control comes naturally. To the best of our knowledge, this is the first comprehensive design and evaluation of dynamic link width control for emerging bit-serial links and its comparison against the traditional power-state control.

Although our focus in this paper is on networking fabrics, the power management techniques discussed here apply equally well to interconnects that proliferate *inside* a server, e.g., CPU cores interconnect, processor-memory interconnect, PCI-E links connecting NICs, graphics card, and SAS/SATA adapters, etc. Power reduction and careful attention to latency are even more critical for these internal links.

Section II of the paper introduces essential characteristics of power control algorithms and performance metrics. Section III discusses design issues for link width control. Section V discusses the evaluation methodology and shows detailed performances. Section VI discusses the related work.

II. Power Control Basics

A. Serial Links and Networking Fabrics

Until about a decade ago most multi-conductor interconnects (e.g., twisted pair Ethernet, PCI bus, processor-memory

[1] These and many other capabilities such as packet tunneling were originally promoted as PCI-E advanced switching. [12] but that did not get much traction due to inter-operability issues.

bus, etc.) used a parallel interface where multiple bits (e.g., 4, 8, 16, or 32) are transmitted at a time. As the link speeds increased, the parallel interface became untenable because of a host of problems including excessive cross-talk between bits, high capacitance, skew between bit timings, reflections, etc. As a result, links started a slow transition to bit-serial technologies with differential signaling. Differential signaling (where the signal and its complement are sent along a twisted pair of wires and the receiver detects their difference) can improve signal quality substantially. With this technology, link BW scaling is done by running multiple "lanes". Serial links also admit dynamic width changes (by putting certain lanes in low power mode), something that is not practical with parallel links.

Most of the link types mentioned above are already using bit-serial technology, which has moved rapidly from the original GEN1 (2.0 Gb/s) to GEN2 (4.0 or 5.0 Gb/s) to GEN3(8.0 or 10.0 Gb/s). Gen1 uses 8b/10b encoding which makes the raw speeds of 2.5 GHz deliver only 2.0 Gb/s, but later generations may or may not using this encoding. However, running 10 Gb/s over copper is extremely challenging, and further speed bumps w/o repeaters and signal conditioners may be questionable at least over long lengths. Therefore, the BW increases are more likely to be achieved via multi-laning. For example, the PCI-Express is routinely available in up to 16 lane configurations (denoted as x16). Infiniband has been available in x4 and x12 configurations. The 10 Gb/s copper implementations of Ethernet (CX-4 and KX-4) already use four 2.5 Gb/s lanes. Although the upcoming 40 Gb/s Ethernet can work with 4 lanes of Gen3 links in the optical domain, copper implementations may be forced to use 10 lanes of Gen2 links. The 100 Gb/s Ethernet almost surely will have to use 10 lanes of Gen3.

With all networking technologies adopting the serial interface, PHYs are increasingly becoming "re-purposable", i.e., an appropriate set of configurable parameters can make a link behave like a PCI-E, Infiniband, Ethernet, Fiber-Channel, etc. This unification implies that it should be possible to provide a standard set of link power management capabilities for all link types. In practice, however, the nature and availability of power management mechanisms will depend on the extent to which the lower level capabilities are exploited and enabled at the MAC and perhaps higher layers.

B. Nonoperational Link Power States

The most important aspect in terms of idle power control is the existence of non-operational, low-power states that can be entered whenever the resource in question is idle. Generally there are 3 link power states, usually denoted as L0, L0s and L1. In the following, we shall denote the power consumption in state Lx as $P(Lx)$.

1) L0: This is the normal operational state. For most links, $P(L0)$ has very little dependence on the utilization level, i.e., an idle link dissipates nearly as much power as one bursting data at full rate.
2) L0s: This is a non-operational state with $P(L0s) < P(L0)$ and fairly low latency to enter into or exit from L0s. We henceforth denote the entry+exit latency for L0s state as η_0. Typically, each direction of the link can be put in L0s state independently.
3) L1: This is an even lower power state (often much lower than L0s) but with a significantly higher entry+exit latency η_1. L1 requires a handshake between the two directions of a bi-directional link. If either side refuses to go into L1, L1 will not be entered.

Not all links provide both L0s and L1, but when both states are available, L0s is entered first and if no acitivity is observed for a period significantly longer than η_1, the link is *promoted* to L1. It is to be noted that with many links, even η_0 is quite large (e.g., at least 100's of ns). Thus, without a careful design, power management of links could result in significant latency penalty. Of course, the performance impact of the latency depends on myriad details including the type of link and the workload [7].

Link width control – or putting certain lanes in L0s state – comes naturally for multilane serial links. The main advantage of link width control is that so long some lanes are "up", the non-zero communication bandwidth significantly reduces the impact of high entry+exit latencies.

The PCI-E specification explicitly includes the definition of L0s and L1 states for each lane. With PHY unification, L0s and L1 states (and link width control) should be feasible even for other link types, however, the current availability is spotty at best. For example, IEEE is currently engaged in an effort called *Energy Efficient Ethernet* or EEE that has examined link state control [4]. However, the currently defined low power state corresponds to L1 (bidirectional control) and the focus is more towards slow changes so that the interfaces can be placed in low power model during long-idle periods. Consequently the issue of minimizing exit latencies has not been addressed adequately at this point.

C. Operational Link Power States

When the link is lightly loaded, a potential method for saving power is to run the link at a lower frequency. The link speed change impacts both directions of a bidirectional link and requires negotiation between the ends. Auto-negotiation of speeds is already available in some link types (e.g., Ethernet), however, speed bumps may not be simply a matter of changing clock rate. For example, in case of Ethernet, the original 10 Mb/s version evolved through 100 Mb/s, 1 Gb/s and then 10 Gb/s with significant changes in terms of collision detection/avoidance and retry. Thus Ethernet speed change really amounts to PHY switching, which can be very slow. The Energy Efficient Ethernet (EEE) project has considered a rapid PHY switching (RPS) scheme that is more suited for dynamic speed changes, but it too is designed for dealing with low traffic over longish intervals (at least a few seconds) [1].

In addition to the long switching latency, speed switching has two inherent problems that make it less attractive than link power state and link width control. First, it is generally better to transmit packets at a high speed and then go into a low power state instead of "trickling" at a low rate. Second,

running a link at very low speed simply because there isn't enough traffic could be disastrous. For example, down-rating a 10 Gb/s link to 100 Mb/s means a 100-fold increase in packet transmission latency. Such a huge increase in communication latency could result in significant performance impact irrespective of the bandwidth utilized.

D. Power Control Effectiveness

When evaluating power control, the two basic metrics to consider are *average efficiency* ($\mathcal{E}$) and *average additional latency* ($\mathcal{L}$) introduced by the power control. The efficiency $\mathcal{E}$ is defined as the fraction of idle period (or "gap") for which an algorithm is able to keep the resource in low power state (excluding entry and exit periods, which typically burn L0 level power). Efficiency directly translates into the power savings. That is, $\mathcal{P}'_{\rm idle}$, the average idle power with power control, can be related to L0 and L0s mode power consumption by the trivial equation:

$$\mathcal{P}'_{\rm idle} = [1 - \mathcal{E}(U_{\rm rsrc})]P_{L0} + \mathcal{E}(U_{\rm rsrc})P_{L0s} \quad (1)$$

The power control latency $\mathcal{L}$ can be anywhere between 0 and the entry + exit latency η. The upper bound is realized when the traffic arrives just when the resource begins to enter the low power state. Generally, there is no way to short-circuit the process – the resource must enter and then immediately exit low power state in this case.

For any well designed power control algorithm, a higher efficiency almost always comes at the cost of a higher latency. The additional latency is important only to the extent of its performance impact; thus an appropriate metric is performance per watt (PpW), where the performance can be defined as the appropriate measure of workload throughput.

Relating latency to workload throughput is well studied for certain types of links such as processor-memory links, but not necessarily for others (e.g., Ethernet links). In all cases, however, the mechanism is the same: additional latency results in CPU stall if none of the HW threads can make any progress due to data or control dependency. Following the standard approach used in architectural modeling, we can write the following equation for the workload throughput λ in terms of the overall data access latency L [6]:

$$\lambda = \frac{C\,U_{cpu}}{1 + \beta L} \quad (2)$$

where C and β are appropriate constants. Thus PpW can be estimated for specific systems and workloads. In this paper, we however stay at latency control itself, by assuming that *tolerable latencies* have been estimated from performance impact considerations.

E. Link State Control

A power state control algorithm has to make two crucial decisions: (a) when to go into low-power state, and (b) when to exit it [15]. Because of the finite entry/exit time to/from low-power mode, it is undesirable to go into low-power mode for small gaps (or idle periods). Since the gap duration is not known in advance, the general technique is to monitor it for some period, henceforth called *runway*, and if the resource is still idle, start transition into low-power mode. The mechanism to set runway is one aspect of designing power state control algorithms.

The exit from low power state could be either *reactive* or *proactive*. A reactive exit is directly driven by the arrival of packets to transmit, and is trivial to implement but results in full exit latency hit every time. A proactive exit is driven by a prediction model of future packet arrivals and initiation of exit in anticipation of arrival. The motivation for a proactive exit is to minimize exit latency; however, its success depends on how well future events can be predicted based on the past history. Reference [7] examines in detail the comparison of proactive and reactive power control, and presents a comprehensive evaluation of a simple but effective proactive algorithm called *exponential smoothing algorithm* or ESA. The algorithm is based on exponentially smoothed estimate of next "gap" based on the past history. The ESA algorithm can be easily designed to behave as either proactive or reactive algorithm and shows better behavior than pure reactive algorithms [7]. It can be implemented in HW with about 4000 gates.

It is important to note here that in order to achieve a fine granularity power control of multi-Gb/s links, any usable algorithms must be extremely simple and easily implementable in HW (e.g., no multiplications/divisions). Moreover, the algorithms should only use information that is readily available to the interface (e.g., number of waiting requests), instead of something that needs to be fetched from elsewhere (e.g., current CPU utilization). These considerations are also reflected in the width control algorithm described in the next section.

III. Link Width Control

A. Width Management Considerations

A multilane link usually allows for certain feasible widths, and the width control algorithm needs to decide how to transition between them. For example, for a x10 link, the possible widths are x1 through x10, but in reality the supported widths may be only x10, x4, x2 and x1. For convenience in description, we assume that successive steps are represented by multiplicative factors between adjacent widths. (In an actual implementation, additive factors may be more convenient.) We denote the decrease and increase factors by the vectors δ and Δ respectively. For example, for the set [x1, x2, x4, x10], we have $\Delta = \{2.0, 2.0, 2.5, 1.0\}$ and $\delta = \{1.0, 0.5, 0.5, 0.4\}$ where the 1.0 entry really means that no change is possible. Obviously, $\Delta_i = 1/\delta_{i+1}$. Often, it is convenient to talk about a delta value for a given width W, we shall denote this as $\delta(W)$ and $\Delta(W)$.

The fundamental decision in link width control is when to decrease or increase the width. These decisions are based on an estimate of idle and busy periods in the recent past. We estimate these via simple exponential smoothing. That is, if G_n is an estimate of the gap at nth step, g_n the actual gap, and $0 < \alpha < 1$ the smoothing constant, we have:

$$G_n = (1 - \alpha)\, G_{n-1} + \alpha g_n \quad (3)$$

Similarly, for the busy period B_n, we can write: $B_n = (1-\alpha)\,B_{n-1} + \alpha b_n$ where b_n is the latest busy period. The parameter α is chosen by considering a tradeoff between responsiveness and jitteriness of the estimate. A value of around 1/16 is usually reasonable.

One parameter of interest in width control is the recent link utilization, which is given by $U_n = B_n/(B_n + G_n)$. The other parameter of interest is the current *residual work*, i.e., time needed to transmit all accumulated packets. Depending on the link type, residual work may be measured either in terms of number of packets or number of bytes to be transmitted. For ease of reference, we shall call both as "queue length" (denoted Q_n at step n) with appropriate interpretation being implementation dependent.

Let W_n denote the width of the link at nth step, and W_{min}, $W_{\max}$ the minimum and maximum widths. Usually, $W_{min} = 1$. (Note that $W_{\min}$ excludes the case where the entire link will be shut down.) Then the condition for width decrease is given by:

$$W_n > W_{min} \;\&\; G_n > \gamma_1\, B_n \tag{4}$$

where γ_1 is a constant. The first part of the condition is obvious and the second part requires the current link utilization to be sufficiently small, i.e., $U_n < 1/(1+\gamma_1)$. Note that since queue length is an instantaneous measure, it is inappropriate to include it in the width decrease condition.

For link width increase, we need an emergency provision; if a lot of packets arrive suddenly, we must increase the width. This is done by having a high queue threshold (Q_{HT}). In normal cases, we require the transmit queue to have some minimum number of packets [defined by the low threshold (Q_{LT})] and a sufficiently high link utilization estimate. This leads to the following width increase condition:

$$W_n < W_{\max} \;\&\; \left[Q > \frac{Q_{HT} W_n}{W_{\max}} \;\middle|\; Q > \frac{Q_{LT} W_n}{W_{\max}} \;\&\; G_n < \gamma_2 B_n \right] \tag{5}$$

where γ_2 is another constant (to be addressed shortly). Note that the equation does not use the queue length thresholds Q_{HT} and Q_{LT} directly; instead it scales them by the current width. The effect of this scaling is to maintain the same latency threshold as the link width goes down (e.g., 1/2 width link means double the packet service time but then halving the queue length threshold means no net impact). Of course, this mechanism may not have much role to play for small queue length threshold choices.

An important question in width control is by how much should the width be changed at each step. For example, if the current width is x10 and the "right" width to transition to is x2, we can either jump to x2 directly or go through the intermediate width of x4. During the development of the algorithm we considered the following 3 strategies for changing the width:

1) Cut to $W_{\min}$ in one step, but increase gradually in steps given by the vector Δ.
2) Decrease gradually according to the vector δ but increase to $W_{\max}$ in one step.
3) Increase and decrease gradually according to the vectors Δ and δ respectively.

The first scheme is motivated by power consumption being prioritized over performance, i.e., we immediately cut the width to minimum value when conditions are ripe, and then "float up" to the correct width. The second scheme has the opposite motivation – minimize latency at the cost of power consumption by restoring the link to full width immediately and then letting the width "float down". Finally, the third scheme balances the two. Based on extensive experimentation we deemed the last scheme as most suitable and will be used henceforth (details omitted due to lack of space). Hence only this scheme is described in the following.

The parameters γ_2 and γ_1 in the equations above need to be related for the algorithm to work properly. Suppose that we are right at the point where $G_n/B_n = \gamma_1$. Now, if the prevailing utilization U_n dips ever so slightly, we decrease the width by the factor of $\delta(W)$ (i.e., the δ value at the current width W) thereby bumping up the utilization by $1/\delta(W)$. Thus if there is no further change, the ratio G_n/B_n – after both G_n and B_n have adjusted to the change – no longer equals γ_1, but something else, say γ_1'. It is easy to see:

$$\gamma_1' = \frac{1 - U/\delta(W)}{U/\delta(W)}, \text{ where } U = \frac{1}{1+\gamma_1} \tag{6}$$

Simplifying, we get $\gamma_1' = \delta(W)(1+\gamma_1) - 1$. Now, we can choose $\gamma_2 = h\gamma_1'$ where $h < 1$ is the hysterisis parameter that ensures that the link will not flip-flop. To simplify multiplication by h, we can assume it to be $1 - 2^{-k}$ for $k = 1$ or $k = 2$.

As an example, suppose that $\gamma_1 = 4$ and $\delta(W) = 0.5$. Thus when the link utilization hits $1/(1+\gamma_1) = 20\%$ on its way down, we halve its width. With no material change in traffic, the link utilization will now increase to 40%. In this case, $\gamma_1' = 1.5$ and with $h = 0.5$, the link width will increase only when $G_n > 0.75 B_n$, or $U_n > 1/(1+0.75) = 57\%$. If the width does increase, the resulting utilization would be 28.5% and the utilization would again have to go down to 20% for a downshift to occur. Thus no link flip-flop can occur. Although this safety margin may seem somewhat low, note that the condition $Q > Q_{LT} W_n / W_{\max}$ further protects against flip-flops.

IV. Complete Width Control Algorithm

In this section we use the equations above to craft a complete *dynamic width control algorithm* or DWCA. Width control, as described above, is inadequate by itself for a complete link power control algorithm for two reasons: (a) Even at extremely low traffic, it will keep one lane active, and (b) If the traffic stops suddenly for some long period of time, the link will get "stuck" at its current width W, i.e., W lanes will continue to consume full power. To handle both of these issues, we need a simple power-state control as well. In particular, whenever the interface goes idle, we start a *runway* timer. If the runway expires, the entire link is transitioned to the low power state. The exit from low power state will be triggered by packet arrival, at which time the link can go into its previous width state.

One issue in implementing width control is the conditions under which width control is invoked. Width decrease is best

invoked at the end of a busy period, since there is little reason to slow down while the packets are still queued up. On the other hand, the need for width increase should be checked each time some number, say N, new packets arrive at the interface. In a HW implementation, however, choosing $N > 1$ only complicates the implementation w/o offering much in efficiency. One other point note is that width decrease and increase behave slightly differently: width decrease can happen almost immediately after the decrease decision has been made, but the actual increase happens only after the sleeping lanes have exited the low power state. The actual width change does require reconfiguring the multiplexer which makes the entire link inoperational for a very brief period (1-2 ns).

In order to simplify implementation, multiple simultaneous increases and decreases may need to be disallowed. For example, consider a x4 link which can operate in x4, x2 and x1 modes. Because of long exit latencies, it is possible that while x1→x2 transition is in progress, the conditions signal the need for x2→x4 transition. While there is no conflict between two separate sets of lanes exiting the low power state concurrently, the HW needs to be either designed to manage these concurrent exits or concurrent events should be disallowed. In theory, the same could apply to downshifts as well, except that entry latencies are typically rather small. It is even possible that during a x2→x4 transition, the traffic drops fast enough to trigger a x2→x1 transition. Such a parallel transition should be disallowed to avoid link flip-flops.

In a HW implementation it is important to avoid expensive operations such as arbitrary multiplications and divisions. This requires a judicious choice of parameters such as γ_1, h, $Q_{HT}/W_{\max}$ and $Q_{LT}/W_{\max}$ for equations (4) and (5). Also, even though in the description we simply called δ as a "vector", it is not useful to try to implement it like an arbitrary array; instead, the implementation will be specific to the number of distinct δ's required. By making appropriate optimizations, it is possible to implement the entire algorithm in less than 10000 gates, which is reasonably small for most situations. In particular, a detailed design that we did for the special case where all delta values are 0.5 takes only 6000 gates and a power consumption in 30-60 milliwatt range with 65nm process.

V. Power Control Performance

For evaluating various power control algorithms, we designed a comprehensive simulation model called LMPOWER[9]. LMPOWER represents two platforms connected via a communication link with each system using an abstract model of CPU cores and a rather detailed model of the memory and the interconnect. The model does represent the impact of latencies on CPU stalls; however, this aspect is not important for the results shown here. The model could be driven both using analytically generated traffic as well as actual link traces. The results shown here focus on analytic traffic with a finite horizon Zipf($\alpha = 2.25$) inter-arrival time distribution (inter-arrivals times are integers). The actual link traces show varying degrees of correlational properties in the traffic; however, to keep the power saving results conservative, no correlation was dialed into the traffic used here. This is because positive correlation improves predictability and hence yields better results.

For the results, we emulated a 40 Gb/s Ethernet link assuming 10 lanes of serial copper link, each providing 5.0 Gb/s. (With Gen2 technology using 8b/10b encoding, this means that the link will run at 6.25 GHz rate). Since we are interested in latency sensitive situations, we consider transfer of small control packets of size 128B. Only the link power and link latency are considered here even though the model can produce system level numbers. The actual power numbers are somewhat uncertain since no real 40 Gb/s Ethernet exists today; however, the absolute numbers are really not of interest in the discussion that follows. In the plots, we use request interarrival time (IAT) on the x-axis (instead of link utilization). It is, of course, inversely related to link utilization and IAT=100 approximately corresponds to 47% link utilization.

Figs 1 and 2 show power consumption and Latency for the DWCA algorithm against ESA (the proactive power-state control algorithm). The two were parameterized so that they will have roughly the same and rather low latency at high link utilizations. For width control, we used 1x, 2x, 4x and 10x widths, $\gamma_1 = 6$, $Q_{LT} = 4$ and $Q_{HT} = 8$ (both in terms of packets). The runway to transition the entire link to the low power state was assumed to be 4x the entry+exit latency (as opposed to 1x for power state control). It is seen that width control easily outperforms ESA in *both* power consumption and latency in much of the range. *This is significant since in most situations, one can merely tradoff power against latency.*

We shall refer to the case above with 4 possible widths as *fine grain control*. Figs 3 and 4 compare this against the *coarse grain control* where the only possible widths are 1x, 2x, and 10x. It turns out that with the "nominal" γ_1 value of 6, the difference between the two is rather small; hence, we chose to show the situation with $\gamma_1 = 4$. A lower γ_1 means a more aggressive width control and hence larger latency. In particular, the starting latency is now 105 ns as opposed to 70 ns. In this case, a finer grain control is a definite plus in terms of power (and very marginally so in terms of latency). Of course, the finer grain control will be a bit more complex in gate count, but the additional complexity seems worthwhile.

Finally, Figs 5 and 6 show power and latency for 3 sets of DWCA parameters: (a) "Normal", by which we mean $\gamma_1 = 6$, $Q_{LT} = 4$, $Q_{HT} = 8$, (b) High utilization threshold where γ_1 is reduced to 4 (i.e., the first width cut happens at 20% utilization instead of 14%), and (c) High queue length threshold, i.e., $Q_{LT} = 16$, $Q_{HT} = 32$. Here we see the typical power-latency tradeoff – lower power means higher latency and vice-versa. Interestingly, however, at moderate utilizations, case (c) provides lower power than (b) but at about the same latency. Thus, choosing higher queue length thresholds is more desirable than a lower γ_1 value.

VI. Related Work

Although the link width control or simple algorithms to accomplish it are not new, we are not aware of a comprehensive treatment of link width control in the open literature.

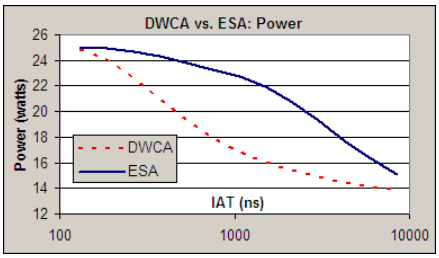

Fig 1. Power Consumption: Link Width & Power-state Controls

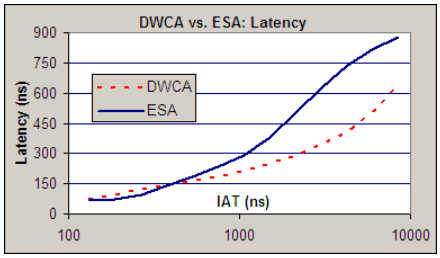

Fig 2. Transfer Latency: Link Width & Power-state Controls

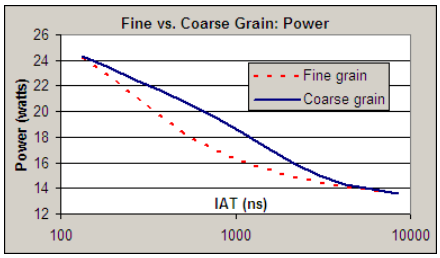

Fig 3. Power Consumption: Fine vs. Coarse Grain Width Control

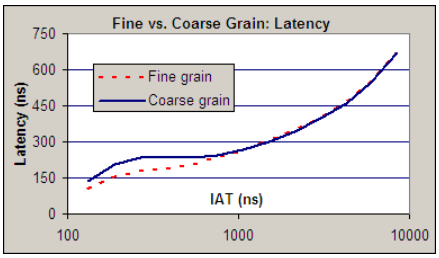

Fig 4. Transfer Latency: Fine vs. Coarse Grain Width Control

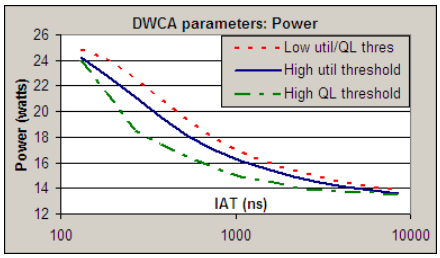

Fig 5. Power Consumption: Sensitivity to Utilization & QL Threshold

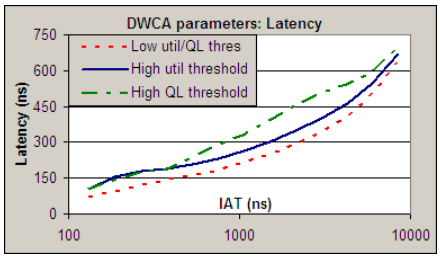

Fig 6. Transfer Latency: Sensitivity to Utilization & QL Threshold

There is work in the literature on other link power schemes, but it is directed primarily towards shutting down nearly idle links. For example, both [3] and [2] make the case for shutting down idle links and switch/router ports in order to save energy until certain number of packets have accumulated. Dynamic voltage frequency scaling (DVFS) – a popular scheme for CPU power state control – has been explored for links [16] and can be exploited in designing PHYs that switch between multiple modes. Simultaneous DVFS control for both processors and links is considered in [10]. Reference [13] proposes redesigning OS to increase the burstiness of the workload and thereby elongate low power periods. References [15] provides a survey of techniques for energy-efficient on-chip communication, and reference [17] is a survey of various power reduction techniques for microprocessors including putting components in low power mode.

VII. Conclusions and Future Work

In this paper we studied the power saving potential of dynamic link width management for emerging high speed communication links that are invariably based on the serial-link technology. We showed that link width control is highly desirable when available, especially in terms of its potential to mitigate power state switching latencies. Link width control can be combined with power-state and link speed control in more sophisticated ways so that it is possible to take advantage of whatever power control capabilities are available for a given link. It is also interesting to explore end-to-end traffic shaping (as in [13]) to increase the effectiveness of link power control.

Acknowledgements: The author would like to acknowledge numerous discussions with Sebastian Herbert and Alexander Jimbo in the design and evaluation of DWCA.

References

[1] F. Blanquicet & K. Christensen, “An Initial Performance Evaluation of Rapid PHY Selection (RPS) for Energy Efficient Ethernet”, IEEE Conf on local computer networks, Oct. 2007, pp223-225
[2] M. Gupta and S. Singh, ” Dynamic Ethernet Link Shutdown for Power Conservation on Ethernet Links”, Proc. of IEEE Intl Conf on Communications 2007, June 2007.
[3] M. Gupta, S. Grover and S. Singh, ” A Feasibility Study for Power Management in LAN Switches”, Proc of 12th IEEE ICNP, Oct 2004.
[4] IEEE task group 802.3.az, “Energy Efficienct Ethernet”, www.ieee802.org/3/az/public/nov07/hays_1_1107.pdf.
[5] Infiniband Trade Association, “Infiniband Architecture Specification 1.2.1”, Vols 1 & 2. Available at www.infinibandta.org/specs/
[6] K. Kant and Y. Won, “Server Capacity Planning for Web Traffic Workload”, IEEE trans. on knowledge and data engineering, Oct 1999, pp 731-747.
[7] K. Kant and J. Alexander, “Proactive vs. Reactive Idle Power Control”, Proc. of DTTC, Aug 2008.
[8] K. Kant, “Towards a Virtualized Data Center Transport Protocol”, Proc. of 2008 INFOCOM workshop on High Speed Networks, Phoenix, AZ, April 2008,
[9] K. Kant, “LMPOWER – A Comprehensive Link-Memory Power Management Simulator”, Unpublished report.
[10] J. Luo, N. Jha, Li-S Peh, “Simultaneous dynamic voltage scaling of processors & communication links in real-time distributed embedded systems”, IEEE Trans on VLSI, 15, 4, April 2007.
[11] Z. Meggyesi, “Fiber Channel Overview”, Available at hsi.web.cern.ch/HSI/fcs/spec/overview.htm.
[12] T. Miller, “PCI Express and Advanced Switching: Data Movement Protocols”, COTS Journal, Oct 2003, pp 76-79.
[13] A. Papathanasiou and M. Scott, ”Energy Efficiency through Burstiness”, Proc of the 5th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA’03), pp. 44-53, Oct 2003.
[14] PCI Special Interest Group, “PCI-Express Base 2.0 Specification”, available at www.pcisig.com/specifications/pciexpress/base2/
[15] V. Raghunathan, M. B. Srivastava, and R. K. Gupta. A survey of techniques for energy efficient on-chip communication. In Proc. the 40th Conference on Design Automation, 2003.
[16] L. Shang, Li-S Peh, N. Jha, “Dynamic voltage scaling with links for power optimization of interconnection networks”, Proc of HPCA 2003.
[17] V. Venkatachalam and M. Franz, “Power Reduction Techniques for Microprocessors”, ACM computing surveys, Vol 37, NO 3, Sept 2005, pp 195-237. (http://www.ics.uci.edu/ vvenkata/finalpaper.pdf)