Low Power Network Processor Design Using Clock Gating

Yan Luo, Jia Yu, Jun Yang, Laxmi Bhuyan
University of California Riverside
Riverside, CA 92521
{yluo, jiayu, junyang, bhuyan}@cs.ucr.edu

ABSTRACT

Network processors (NPs) have emerged as successful platforms to providing both high performance and flexibility in building powerful routers. Typical NPs incorporate multiprocessing and multi-threading to achieve maximum parallel processing capabilities. We observed that under low incoming traffic rates, most processing elements (PEs) in NPs are nearly idle and yet still consume dynamic power. This paper develops a low power technique to reduce the activities of PEs according to the varying traffic volume. We propose to monitor the average number of idle threads in a time window, and gate off the clock network of unused PEs when a subset of PEs is enough to handle the network traffic. We show that our technique brings significant reduction in power consumption (up to 30%) of NPs with no packet loss and little impact to the overall throughput.

Categories and Subject Descriptors: B.9.1: Low-power design **General Terms:** Design **Keywords:** Network Processors, Low Power.

1. INTRODUCTION

Network processors (NPs) have emerged as a new class of programmable processors for packet processing. New generation of NPs offer high performance through parallel processing architecture, which incorporates multiple processing elements (PEs) configured as either parallel or pipelined units. Being programmable, NPs support new applications with improved time to market, product's life time and lower cost.

A number of challenges for NP implementation are already evident, and power dissipation is one of them. Typical routers mount a few racks containing groups of line cards (e.g. 8, 16) each of which contains one or two NPs. Such routers are extremely dense in power dissipation, which results in high operating temperature. Moreover, the clock frequency of many commercial NPs keeps increasing and more PEs are added to an NP to deliver high performance and throughput. This trend implies that the NP's power consumption will keep rising. For example, Intel IXP2850 contains 16 PEs operating at 1.6GHz with 19∼ 25W power consumption [11], while it's predecessor IXP1200 contains 6 PEs operating at 232MHz with 4.5W power consumption.

This paper develops a low power technique by exploiting the variation of the network traffic load. Routers, especially edge routers, experience different volumes of traffic during different times of a day. From some of the NLANR [9] router traces, we can observe that traffic volume varies, for example, from 5 to 50 Mbps in a 24-hour period with low rates at the nighttime. This implies that much less processing power is required at nighttime compared to the daytime. The low traffic load leaves the NP underutilized, which brings challenges and opportunities for low power NP design.

We propose a low power technique to save active power of NPs without sacrificing the throughput. Our approach is to use the clock gating technique on PEs when the packet processing requirement is low, and reopen the clocks when the need is high. The motivation of using clock gating is to effectively "turn off" PEs but not actually power them down completely considering the high cost of powering them up. The cost mainly comes from the long latency of loading the program code into the PE memory. The decision of turning on/off PEs is made dynamically according to the utilization of PEs. A good indication is the number of idle threads that are present in the system. If some of them are idle, it means that there are more processing power than the amount required by the incoming packets. Therefore, we propose to use the number of idle threads to determine how many active PEs are necessary. To determine when to turn on a PE, we observe the pressure arising from the packet incoming buffer. A full buffer indicates low processing capability from NP, and packet drops may happen. Our goal here is not to introduce *extra* packet loss due to clock gating the PEs, but to guarantee enough processing capability at low power consumption.

We investigate the potential problems after turning off the PEs and give solutions to overcome them. We design techniques to avoid possible extra packet loss due to turning off of PEs. To accurately measure and test the effectiveness of our technique, we implemented our scheme in an NP simulator [6]. We added clock power modeling to the simulator, and also studied the proper timing to apply clock gating in NP. We measured the power savings and throughput using real world router traces from NLANR [9]. Our experiments show that up to 30% of power savings can be achieved when the traffic is non-saturated.

The remainder of the paper is organized as follows. In section 2, we introduce the network processor model that is used in our design. Section 3 to 5 discusses in detail about our clock gating algorithms, the problems encountered and their solutions, and the modeling and the justification of using clock gating. We show the results of our clock gating technique in section 6 and discuss related work in section 7. Finally, section 8 concludes this paper.

2. NETWORK PROCESSOR MODEL

A network processor usually contains multiple processing cores, dedicated hardware for common networking operations, memory interfaces, high-speed I/O interfaces etc. Here we use NePSim simulator [6] to model the NP architecture. NePSim is based on Intel IXP1200 NP and includes a cycle-accurate architecture simulator and a power estimator. IXP1200 consists of a StrongARM processor, six multi-threaded PEs called microengines (MEs), memory interfaces and high-speed buses. The StrongARM processor is used for management functions such as initializing MEs, running routing protocols, exception handling. The MEs can be programmed to perform high-speed packet inspection and data manipulation. The memory modules outside IXP1200 are used to store control data information and packets. The bus interface, IX bus unit, transfers packets between MEs and network interfaces. The usage of each component is highly dependent on the application and workload.

3. CLOCK GATING POLICIES

In this section, we discuss the policies of turning off and on PEs, and the parameters with their thresholds we use in the policies.

DAC 2005, June 13–17, 2005, Anaheim, California, USA.
Copyright 2005 ACM 1-59593-058-2/05/0006 ...$5.00.

3.1 Selecting the parameters

The idea of turning off PEs originates from the observation of the network traffic variation over time. Such variation is usually specified in terms of packet arrival rate in unit of megabits per second (Mbps). It is natural to use this information as a guide to making decisions. However, it is very difficult to set common thresholds on arrival rate because different applications support different line speeds.

When the NP is overloaded, incoming packets start to be dropped at network interfaces, which indicates that current NP processing power is not enough and more PEs are needed. Thus, packet loss is a good indication of the saturation of an NP. Although it can serve the purpose of waking up inactive PEs, this parameter implies that a packet has been lost, whereas one of the design goals of our scheme is to avoid extra packet losses that are introduced due to the reduced number of PEs for power savings.

Alternatively, a PE should be turned off(on) when the required workload for the entire NP is low(high). Such a status can be indicated by (1) the idle time of a PE during which the PE does no effective work; (2) the length of the thread queue in which a thread waits for incoming packets; and (3) the fullness of an internal packet buffer where packets come in and wait to be processed. We will analyze each of these parameters next.

The idle time of a PE is a measure of the PEs being put into the *sleep mode*. A multi-threaded PE "sleeps" when all of its threads go to "sleep". A thread is put into sleep mode when it needs to wait for certain events. Two major events are memory access and new packet arrival, as they are both time consuming. Thus, the idle time of a PE is a mixture of different events, not a direct suggestion of low workload. Therefore, we will not use this parameter in guiding the decision of turning off the PEs.

The thread queue holds threads that are waiting for the arrival of new packets. When a new packet arrives, the head of the queue wakes up to service the packet. The lower the packet arrival rate the longer the thread queue. In fact, the number of threads waiting in the queue is the number of excessive threads for the current traffic load. In addition, the length of the waiting queue can be easily monitored with little hardware overhead and is not application-specific. Thus, we will use it as the main parameter to determine when to clock gate a PE.

The internal packet buffer is a place to hold temporarily the incoming packets before a thread fetches them for processing. In IXP1200, the RFIFO is a buffer of this kind [10]. When the RFIFO starts to saturate, it implies that current PEs are almost inadequate, and more processing power is required. When partial PEs are operating, a full RFIFO implies that more PEs should be brought up to clear off the buffer. Otherwise, packets will be dropped. Thus we use the fullness of the internal buffers as an indicator to activate more PEs.

In summary, the parameters we will use are the length of the thread queue and the fullness of the internal packet buffer. Both of them are monitored on-chip. The queue length is compared with certain thresholds at fixed time intervals. If a pre-defined condition is satisfied, the PEs will be turned off or on. Next, we will discuss how to determine the thresholds for the parameters.

3.2 Determining the thresholds

As explained earlier, the length of the thread queue, $\boldsymbol{l}$, indicates the excessive processing power of the NP. If the number of threads each PE supports is $\boldsymbol{T}$, then we could use one fewer PE if $l > T$ to sustain the network traffic. However, l is a varying number since the packet arrival rate is varying due to the network conditions. Therefore, we need to estimate the averaged l during a period of observation time, $\boldsymbol{P}$, to decide the excessive processing power in the NP.

We monitor the thread queue length l for a period of P cycles. During this time, l may exceed T for a certain number of cycles, $\boldsymbol{C}$. If C accounts for the majority of cycles in P, then we have high confidence of l being greater than T. Hence, we use a threshold $\boldsymbol{th}$ ($th < P$) in unit of cycles that, when C is greater than th, a PE will be clock gated. Note that the value of th reflects how aggressively we shutdown PEs. The smaller the th the more aggressive our scheme is. We initially set th as half of P. This indicates a medium aggressiveness. After that, we allow th to be updated dynamically so that different applications and traffic condition can find their own appropriate thresholds. This is because the th is very application and network condition dependent. The updating algorithm is as follows. The th is increased (up to the value of P) if negative impacts, such as internal packet buffer full, have been observed. On the other hand, th is lowered if it has not introduced negative impact to the NP. This means that there still might be excessive processing power left on average. The advantage of allowing th to adjust itself is that we can let th converge for an application without the need to find a common value across all applications.

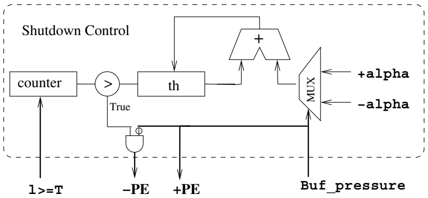

Figure 1: Threshold self-adjusting logic.

We then develop a control logic, as illustrated in Figure 1, for choosing PE configurations using the thresholds. *counter* stores the number of cycles when $l >= T$. We turn off a PE if *counter* is greater than th. When buffers experience pressure, i.e., the buffer is full, we turn a PE back on. During every period P, th is increased by *alpha* if the internal packet buffer has been full once, or is decreased by *alpha* otherwise.

4. CHALLENGES IN GATING OFF PE'S

4.1 Terminating threads safely

To turn off a PE completely, the hardware needs to terminate all the threads first. However, the threads are either in the middle of processing a packet or just finished processing a packet. For the latter case, it is safe to kill the thread immediately. For the former case, the thread should finish processing the current packet and then terminate. Otherwise, the packet occupying the spaces in the packet buffer (or memory) cannot be removed from the system, creating "leakage" in resources which would be drained out eventually.

When a decision is made on turning off a PE, we set an "off" flag in that PE informing it to prepare for shutting down. In IXP1200, there is a "kill" instruction that a thread can use to terminate itself. For the threads that are responsible for receiving packets and processing them, they need to check the "off" flag right after finishing processing a packet. If the flag is set, it executes the "kill" instruction and relinquish the pipeline resources. Therefore, implementing this part requires a flag bit per PE and an extra conditional branch and "kill" instruction in the program.

4.2 Reschedule packets for orphan ports

In some NPs such as IXP1200, the packet incoming ports are statically assigned to the PEs. Hence, when a PE is turned off, the ports from which the PE reads packets become "orphans", i.e., there will be no threads that receive and process packets from those ports. As a result, the packets coming from those ports would be dropped. To address this problem, we develop a dynamic mapping scheme, i.e., every thread can take packets from every port as long as there is an incoming packet. Such a dynamic mapping can be found in IXP2400/2800 implemented in software as well [11], which demonstrates the readiness of applying our technique in up-to-date NPs. The main advantage is to provide flexible packets dispatching to threads as explained next.

The dynamic mapping is accomplished by adding very simple hardware, shown in Figure 2, in the interface controller. Basically, it determines which port has new packets and assign them to the first awaiting thread. This simple scheduler scans an existing register, the "port_rdy_status", that indicates the readiness of each port. The first available port number P_s is then assigned to the thread T_i at the head of the thread queue. Thread "T_i" is then woken up to read a packet from "P_s". The scheduler scans through the register in a round-robin fashion.

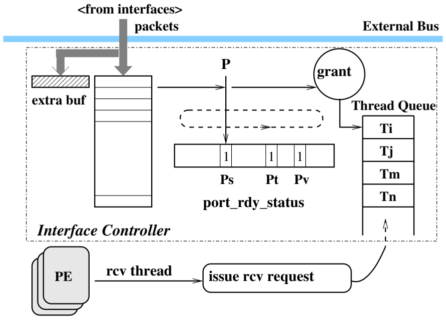

Figure 2: Dynamic thread-port mapping design.

The dynamic scheduler will take some extra time to perform mapping between the ports and the threads. This is because the scheduler needs to read and test the bit one by one to find the first bit that is set. Also, the thread's request needs to be enqueued and dequeued which are both extra operations compared to the static scheme. Though reading and testing the status register bit can be done very quickly, we conservatively charge 1 clock cycle of the 232MHz NP to every operation, i.e., if m bits are scanned, m cycles are charged. We also charge 1 cycle to thread enqueue and dequeue respectively. In addition, the controller will also take up some power which will be included in calculating the net power savings of the NP.

4.3 Avoid extra packet loss

In general, packet loss may happen when the incoming traffic load exceeds the maximum processing capacity of the NP. We cannot avoid this kind of packet loss since it is the nature of the NP even if all the PEs are running. However, when we clock gate some PEs, packets may be lost when they come in burst but the NP has not responded to such a burst. Specifically, the packets will quickly fill up the internal packet buffer so that newer packets are dropped. We have discussed earlier that when the buffer is full, we immediately wake up a PE to drain the packet buffer.

A clock-gated PE can be woken up very quickly in several cycles [5]. However, it still takes some more cycles before an entry in the packet buffer can be cleared. This time includes some initialization of a thread upon execution and the time to put a thread into the thread queue. In the NP we modeled, this time is within 50 cycles. If a new packet arrives in this period, it cannot be moved into the already saturated internal buffers.

Therefore, we need to use extra buffer space to hold the packets that arrive before a thread comes to fetch packets. The extra buffer space is calculated as follows. The delay before a thread is ready to receive packets is about 50 cycle. The maximal packet throughput we observed in NePSim is about 1Gbps (higher than OC12 supported by IXP1200 typically due to the smaller routing table we used). Thus we need about 30 bytes (1Gbps*50cylces/232MHz) extra buffer space. That is, there are at most 30 bytes coming into the NP during the initiation of a new PE. Since the IXP1200 fragments packets into 64-byte "mpackets", only one additional "mpacket" entry is needed to the packet buffer (RFIFO) as shown in Figure 2. Thus, the extra buffer space needed to avoid packet loss is very minimal.

5. CLOCK GATING

There are many circuit-level approaches proposed to save dynamic or static power. Examples are dynamic voltage scaling(DVS), power gating, clock gating etc. When applying DVS to NPs [6], the adjustment of voltage/frequency requires long latency which is critical for Giga-bit NPs. Similarly, the power gating technology has notable latency considering the huge capacitance on the power supply nodes in a unit. Compared with the above two techniques, clock gating is suitable for NPs because it's simple to implement and quick to take effect. We choose to use *coarse-grained* scheme that deactivates PEs for substantially long periods. It requires less hardware overhead in terms of enabling gates, gating controllers and wires. It also involves less cycle-to-cycle current variation which could introduce large transient power on chip.

5.1 Clock power model

We followed the clock models in [1, 2] and made modifications according to physical features of the IXP1200. The major sources of clock power we considered include: (1) the clock distribution tree; (2) the clock generator (PLL); (3) the clock buffers; (4) pipeline latches; (5) bitline precharging in memory structures; and (6) the clock capacitance in execution units such as the ALU and the worldline decoders. We use TSMC 0.25 μm technology parameters to estimate the power of the clock loading. The above clock load in PEs consumes 0.21W, and Figure 3 captures how the different components contribute to the clock power. Here PLL's power is not included in the pie chart, because it's located outside the PEs.

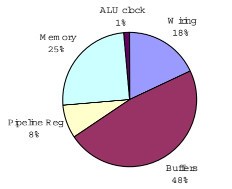

Figure 3: Clock power breakdown for six PEs.

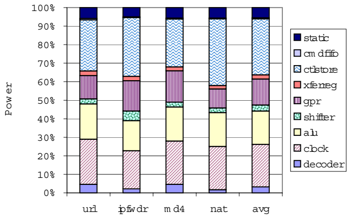

Figure 4: Total power breakdown in one PE.

Figure 4 shows the power consumed by individual component inside a PE, obtained through the NePSim simulations with unlimited uniform traffic input. We observe that the clock load on average consumes 22% of the total PEs' power, while the dynamic logic modules (i.e. ALU, decoders) consume 21%. During clock gating, we gate off the circuitry in the clock network, as well as the activity in the dynamic logic modules. Taking these two factors, we find the total clock related power can achieve 43% of total PEs' power. Additionally, no new accesses to the remaining components will happen, so some of the functional units accesses, which consume the rest 50% dynamic power, can also be saved.

6. EXPERIMENT EVALUATION

6.1 Benchmarks and traffic inputs

There are four benchmarks that are currently ported to NePSim: *ipfwdr*, *url*, *nat*, and *md4*. The *ipfwdr* implements an IPv4 router which forwards IP packets between networks. The *url* is a content-aware routing program that routes packets based on their contained URL request. The *nat* is a network address translation program. The *md4* is a cryptography algorithm used in SSL or firewall to generate 128-bit digital signature on an arbitrary length message. We will use all these four benchmarks in our design and experiments.

We evaluate our design with real network packet traces (e.g. Leipzig-I) from NLANR[9]. The advantage of using real network traces is that they represent typical Internet traffic, in terms of packet size and arrival rate, seen by a router. However, due to the limited simulation speed of NePSim, it is too expensive to simulate the entire traces of dozens of hours. We therefore sample a few seconds of real traffic with different arrival rates as individual inputs to the simulator.

6.2 Power overhead of the control logic

We extended the NePSim simulator with clock power modeling (discussed in section 5), so that we can measure the power savings of clock gating. For the execution units, pipeline latches, memory wordline decoders, the dynamic and static power is included if it is not clock-gated. If the circuit is clock-gated in a cycle, zero power is added.

We also take into account the power overhead associated with the additional control logic in Figure 1. We included the power overhead of the counters, threshold registers, thread queue and comparators measured both with Cadence and the Wattch model using 0.25μm technology, and found they consume negligible power, i.e. 0.00038 Watts for a 20-bit counter or register, 0.002 Watts for a comparator, 0.0065 Watts for 24 entry thread queue. The extra buffer we added increase the buffer access energy by 3.4% which is very small, considering the fact that the original receive buffer only contributes less than 2% of NP power ([6]). Additionally, the controller includes a finite state machine (FSM) which decides the PE on/off decision and an adder. Both FSM and the adder are used just once in each time window, so their contribution to the overall power consumption is very small. We conservatively charge 2% of total PE power as the overhead of the controller.

6.3 Experiment results

We experimented with several traces and present the results of Leipzig-I trace because its link speed falls in the capacity of the IXP1200 NP system. Using other traces have similar results. We scan the trace and extract four segments with different packet arrival rates. For each segment of traces, We feed it to the 16 input ports of NePSim. In this way we formed four traces with the overall arrival rates of, from low to high, ∼90Mbps, ∼180Mbps, ∼360Mbps and ∼480Mbps respectively. We tested different length of shutdown period P ranging from 125K to 8M cycles and found that it affects little to the results. We thus choose 1M cycles as the shutdown period since it can hide well the longest PE shutdown latency observed (60K cycles for url). The initial th is set to 500K cycles (half of 1M cycles) to represent medium aggressiveness. The threshold adjustment amount $alpha$ is set to 2% of P. The metrics we evaluate are power consumption (in Watts) and throughput (in Mbps).

Figure 5 shows the power saving of four benchmarks at different input traffic loads. The power savings are significant in all cases we tested. At the lowest traffic load, up to 30% of the power can be saved for $ipfwdr$ and nat. $md4$ and url saved about 15% and 14% respectively. As the traffic load increases, the power saving amount decreases because less power saving opportunity can be exploited. At the highest traffic load, power reduction numbers are the lowest, but still there are 17%, 15%, 12%, 6% of the total power saved for nat, $ipfwdr$, $md4$, url respectively. Among the four benchmarks, nat has the most power savings while url has the least. This is because the per-packet processing time of nat is the shortest, so on average the thread queue is the longest. This implies that more power saving opportunities can be exploited by our scheme. On the other hand, url has the longest processing time, resulting in the shortest thread queue and the least PE shutdown opportunity.

Our clock gating scheme has very little impact on the system throughput as shown in Figure 6. Deactivating PEs reduced throughput by at most 4% (url with high traffic load). When less number of PEs are active, the packets tend to stay longer in the internal buffer before they are processed and drained out of the NP system. As a result, the system throughput decreases. Note that lower throughput does not imply packet losses; there is no packet loss with the help of the extra one-entry buffer. In addition, if the traffic load continues to increase towards the NP system capacity, the internal buffer will become saturated and clock gating to PEs will not be applied.

Note that the power saving data we present here is in one second period. The overall power saving on a daily basis is tremendous considering that low traffic period contributes to a large portion of a day.

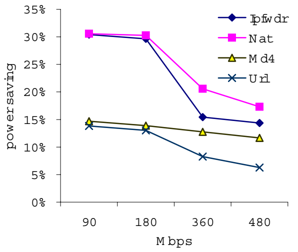

Figure 5: Power saving vs packet arrival rate.

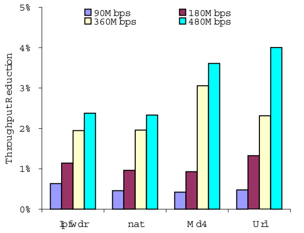

Figure 6: Throughput reduction vs packet arrival rate.

7. RELATED WORK

Recently, power reduction techniques for NPs have appeared at various levels. Our earlier work used dynamic voltage scaling (DVS) to reduce NP's power [6]. It exploited the PE idleness when the PE and the memory speed gap is significant. Our clock gating technique in this paper proceeds further to exploit the variation of the network traffic, achieving higher power conservation. Kaxiras *et al.* proposed IPStash memory architecture to act as a TCAM (used in packet classification and routing) replacement, which significantly reduces the memory set associativity and thus power [4]. Franklin and Wolf developed an analytic performance-power model for typical NPs. They explored the design space of NPs and showed performance-power impact of different systems [3]. Mallik and Memik investigated the relation between transient errors and lowering the voltage for the cache memories of an NP to save power [7].

8. SUMMARY

We investigated mechanisms to lower the power consumption of NPs under non-saturated incoming traffic rates in routers. We studied thresholds parameters that can be used to turn on/off PEs. We presented scheduling policies to address the problems that occur to PE shutdown. In addition, we described the clock power and clock-gating technique. Our experiments show a significant reduction in power consumption of an NP taking network traffic traces of real-world routers.

9. REFERENCES

[1] D. Brooks, V. Tiwari, M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," *ISCA-27,* pp. 83-94, 2000.

[2] D. E. Duarte, N. Vijaykrishnan, M. J. Irwin, "A Clock Power Model to Evaluate Impact of Architectural and Technology Optimizations," *IEEE TVLSI,* pp. 844-855, Vol. 10, Iss. 6, 2002.

[3] M. Franklin and Tilman Wolf, "Power Considerations in Network Processor Design," *Workshop on Network Processors – NP2, in conjunction with HPCA9,* pp. 10–22, 2003.

[4] S. Kaxiras and G. Keramindas, "IPStash: a Power-Efficient Memory Architecture for IP-lookup," *MICRO-36,* pp. 361-372, 2003.

[5] H. Li, S. Bhunia, Y. Chen, T.N. Vijaykumar, K. Roy, "Deterministic Clock Gating for Microprocessor Power Reduction," *HPCA-9,* pp. 113-122, 2003.

[6] Y. Luo, J. Yang, L. Bhuyan, L. Zhao, "NePSim: A Network Processor Simulator with Power Evaluation Framework," *IEEE Micro Special Issue on Network Processors for Future High-End Systems and Applications,* Sept/Oct 2004.

[7] A. Mallik and G. Memik, "A Case for Clumsy Packet Processors," *MICRO-37,* pp. 147-156, 2004.

[8] M. D. Powell, S. Yang, B. Falsafi, K. Roy, T. N. Vijaykumar, "Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories," *ISLPED,* pp. 90-95, 2000.

[9] The NLANR Measurement and Network Analysis, *http://www.nlanr.net/*

[10] Intel Corporation, "IXP1200 Network Processor Family Hardware Reference Manual," http://developer.intel.com/design/netwrok/ixa.html, 2001.

[11] Intel IXP2XXX Product Line of Network Processors, http://www.intel.com/design/network/products/npfamily/ixp2xxx.htm