Dynamic Voltage and Frequency Scheduling for Embedded Processors Considering Power and Timing Constraints

M. E. Salehi, M. Samadi, M. Najibi, A. Afzali-Kusha, M. Pedram, S. M. Fakhraie

M. E. Salehi, M. Samadi, A. Afzali-Kusha, and S. M. Fakhraie are with Nanoelectronics Center of Excellence, School of Electrical and Computer Engineering, University of Tehran, Tehran 14395, Iran (email: {mersali, afzali, fakhraie}@ut.ac.ir)

M. Najibi is with Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran, (email: najibi@ce.aut.ac.ir)

M. Pedram is with Department of EE-Systems, University of Southern California, Los Angeles, CA, U.S.A (email: pedram@ceng.usc.edu).

***Abstract*— An adaptive method to perform dynamic voltage and frequency scheduling (DVFS) for minimizing the energy consumption of microprocessor chips is presented. Instead of using a fixed update interval, the proposed DVFS system makes use of adaptive update intervals for optimal frequency and voltage scheduling. The optimization enables the system to rapidly track the workload changes so as to meet soft real-time deadlines. The technique, which can be realized with very simple hardware, is completely transparent to the application. The results of applying the method to some real application workloads demonstrate considerable power savings and fewer frequency updates compared to DVFS systems based on fixed update intervals.**

Keywords— Dynamic Voltage Scheduling, Dynamic Frequency Scheduling, Dynamic Power Management.

I. INTRODUCTION

In recent years, researchers have proposed several static and dynamic techniques to scale the operating frequency and voltage of embedded processors. These techniques address power and performance tradeoffs at either hardware (fabricated chips) [1]-[4] or software levels [4]-[9]. Software-based scheduling techniques rely on pre-collected offline statistical information of different applications and adjust the voltage and frequency accordingly. In reality, different hardware conditions, such as temperature and process variations, may not be easily observed in software and also different workloads of the same task might need various computation requirements. Therefore, offline methods should use the worst-case execution time (WCET) for scheduling the voltage to cover possible runtime changes in the critical path. Execution times of real-world embedded tasks vary by as much as 87% relative to the measured WCET [10]. Therefore, budgeting for the WCET may result in excessive energy consumption. The researchers in [11] and [12] have proposed power management approaches which track the critical path changes and consider the impact of process variations.

There are feedback-based online DVFS solutions that dynamically control the clock frequency and supply voltage considering the real operating conditions of the underlying processing hardware [13][14][15]. Traditional feedback-based hardware modules for online voltage scaling are computationally expensive, and thus significantly hamper the possible energy savings. In this paper we propose an adaptive DVFS method with low area and power overhead to overcome the hardware complexity of online methods. The proposed DVFS not only scale the frequency and voltage to the minimum required value, but also reduce the frequency and voltage update rates considering both power consumption and system responsiveness.

The remainder of this paper is organized as follows. In Section II, we briefly review the related works while Section III explains the proposed frequency adjustment algorithm based on the effective deadline. The results are discussed in Section IV. Finally, the summary and conclusion are given in Section V.

II. RELATED WORKS

In this section, we briefly review some of the hardware-based DVFS systems which are most relevant to our proposed scheme. A DVFS method that dynamically controls the clock frequency and supply voltage with a fixed update interval is proposed in [13]. Fixed interval DVS scheme for multiple clock domain processors has also been presented in [14]. The online DVS method proposed in [15], exploits a scheduling algorithm based on fixed update intervals. This method is implemented with complex PID controllers that may compensate the power saving of the DVS scheme. The Razor DVS technique [4] uses a delay-error tolerant flip-flop for scaling the supply voltage to minimum allowed value for a given frequency. This method also works based on fixed update intervals. An online hardware-based DVFS scheme for dynamically selecting operating frequencies and voltages in multiprocessor GALS systems is proposed in [16]. This DVFS approach monitors the application workload at predefined times called T_{sample} and scales frequency and voltage values accordingly. The frequency prediction algorithm of this method exploits multipliers and dividers which complicate the hardware realization of this DVFS.

All of these work use fixed update intervals for scheduling voltage and frequency. The optimum value of the fixed update interval strictly depends on the application and patterns of the workloads. Therefore, the value of fixed interval should be carefully tuned for different applications. Fixed update interval DVFS methods can be used when the behavior of the application is predictable for various applications and input conditions and the worst-case behavior is not very different from the average-case behavior. Efficient adaptive techniques, however, are required because of the gap between worst-case and average-case processing demands of different applications.

III. Frequency Scheduling Methods

In DVFS algorithms, the supply voltage is adaptively adjusted based on the predicted frequency. Therefore, one of the major challenges of DVFS systems is how to determine the optimal working frequency without violating the deadlines. In hard real-time systems, there exists an explicitly specified deadline for each workload and if this deadline is not met, the system may not function properly. In some applications, the real-time requirement is not that strict, *i.e.*, it is simply required that the system be able to process the workload in an acceptable amount of time. This can be assured by following a simple conservative rule, which states that the processing of the current workload must be finished prior to the arrival of the next workload. In this way, we may consider the arrival time of the next workload as an *effective deadline* for the current workload in soft real time applications. Knowing the effective deadline for each workload, the frequency may be adjusted to the lowest required value. The importance of the effective deadline in the frequency adjustment is shown in [17] by considering greedy and deadline-aware frequency scheduling policies. As shown in [17], an accurate prediction of the effective deadline may yield a significant reduction in the energy consumption. In this section, we briefly discuss the frequency adjustment method exploited in [13] and then present our frequency scheduling method based on the concept of the effective deadline.

A. Fixed Update Intervals

The frequency scheduling circuitry presented in [13] is composed of an activity monitor and a frequency adjuster. By inspecting a subset of the system control signals, the activity monitor determines whether the system is in active (useful) or idle (useless) mode. By active we mean the period of time the processor is processing any workload. The processor is idle in the slack time when the processing of the current workload is finished and is waiting for the next workload.

Intuitively the output of the activity monitor can be considered as a signal with “1” indicating active and “0” showing idle cycles. When the number of idle cycles in an interval exceeds a threshold value, the frequency is lowered; otherwise, the frequency is increased or held steady. The frequency update rate in [13] directly depends on the value of the fixed interval, i.e., larger values for the interval lead to lower rates of the frequency changes, and hence, a weaker workload tracking ability. Choosing small values for the fixed interval leads to unnecessary frequency and voltage updates.

B. Variable Update Intervals

We have developed an adaptive frequency scheduling algorithm that decreases the frequency update interval in order to track abrupt workload changes while increasing this interval to minimize unwanted voltage fluctuations for slowly varying workloads. The algorithm works based on the concept of effective deadline introduced previously. It is thus desirable to develop a dynamic algorithm to calculate the effective deadline.

1) Effective Deadline Prediction

In the proposed deadline prediction, for each workload, a prediction of the arrival time of the next workload (effective deadline) is made based on an adaptive algorithm. We use effective deadline, *predicted adaptive interval length*, or simply *target interval length* (TIL) interchangeably in the remainder of this paper. During each workload, the system clock cycles are counted by a *cycle counter* whose value is considered as the *current interval length* (CIL). The counter saturates when its value reaches the TIL and resets to zero at the arrival of the next workload. If at the arrival of the next workload, CIL < TIL (see Figure 1(a)), we decrease TIL by an interval step. In this case, the value of the TIL is updated at the arrival of the next workload. If the next workload is not arrived until the CIL reaches the TIL (see Figure 1(b)), we increase TIL by the interval step. In this case, the TIL is updated when the cycle counter reaches the value of TIL (i.e., the counter is saturated.)

The value of the interval step (k_{step}) is updated adaptively. When ‘k’ consecutive interval increases (decreases) occur, it means that the TIL should reach a much larger (smaller) value. In both cases, the interval step will be multiplied by two to speed up the move to higher or lower TIL’s. If these cases do not occur, the interval step is divided by two to minimize the interval variations.

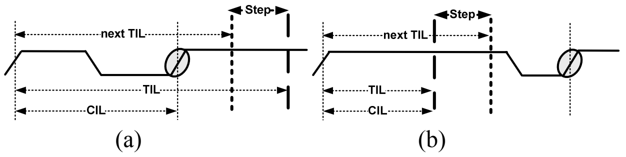


Figure 1. Effective deadline prediction concept for cases corresponding to (a) interval decrement and (b) interval increment.

2) Proposed Frequency Scheduling Method

The predetermined value of $k_{history}$ is used to determine the number of clock ticks that the workload history is examined. $k_{history}$ thus represents the maximum number of acceptable idle cycles between consequent workloads. At the update interval, if all of the workload history bits are '1' ('0'), the estimated frequency is lower (higher) than the required value, and hence, should be increased (decreased) for the next workload. Finally, when some of these history bits are '1', and some are '0', it shows that the system is working with a proper frequency and should not be changed.

A simple zero/one detector logic such as one introduced in [17] may be used for frequency scheduling. For large sizes of the $k_{history}$, the area overheads of the zero/one detection logic increase linearly with the size of the $k_{history}$. To overcome the problem, we propose a Count-Reset-Hold (CRH) circuit whose area overhead is proportional to $\log_2 k_{history}$. For this purpose, we need two types of the CRH counters denoted by CRH0 and CRH1. The proposed counter is an up counter that is controlled by *hold* and *reset* signals. At each clock cycle, CRH0 (CRH1) counts the subsequent '0' ('1') bits of the input signal. It counts up while the *reset* and *hold* signals are '0'. When the *reset* is high, the counter output is reset to 0, and when the *hold* signal is high, the counter output remains unchanged. Figure 2 (a) and (b) show both the CRH0 and CRH1 circuits.

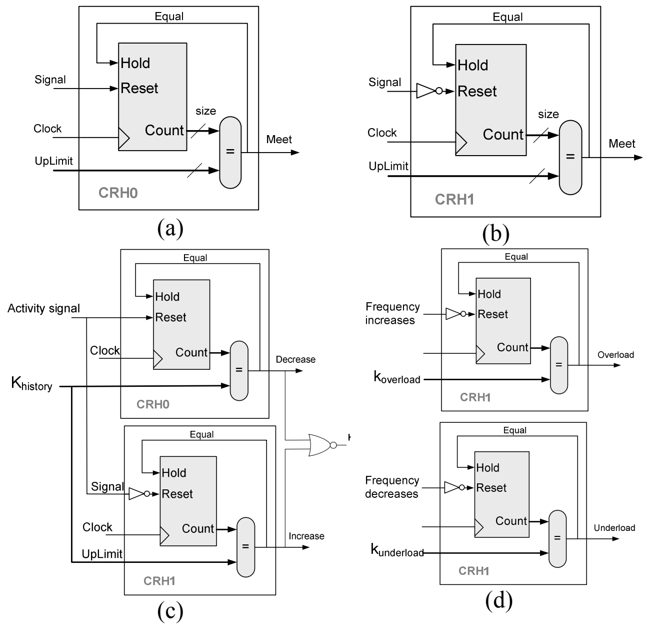


Figure 2. Circuits for (a) CRH0, (b) CRH1 (c) Frequency scheduling (d) overload/underload detection.

The frequency scheduling circuit based on the proposed counter is also depicted in Figure 2(c). When the number of consequent '0' ('1') in the activity signal reaches the $k_{history}$, it means that for the last $k_{history}$ clock cycles the CPU has been idle (active), and hence, the frequency *decrease* (*increase*) signals are activated. Finally, note that the perfect value for the frequency is the amount that provides some idle cycles before the arrival of the next workload [13][17].

3) *Overload and Underload States*

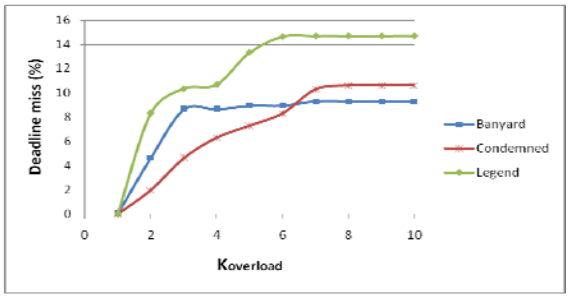
If the effective deadline is not determined correctly, the frequency adjustment may malfunction in some special cases. The two special cases called overload and underload that must be looked upon are suggested in [17]. We detect the overload/underload situation by counting the consequent frequency increases, *i.e.*, if the number of consequent frequency increases/decrease reaches a value denoted by $k_{overload}$/ $k_{underload}$, we conclude that an abrupt increase/decrease in the workload has occurred and the system is overloaded/underloaded. If the overload or underload condition is detected, TIL will be replaced by the minimum possible value and all the frequency updates will be done in the direction of frequency increase/decrease. This small update interval leads to the highest frequency update rate. The overload/ underload detection circuits are shown in Figure 2 (d).

IV. Results and Discussion

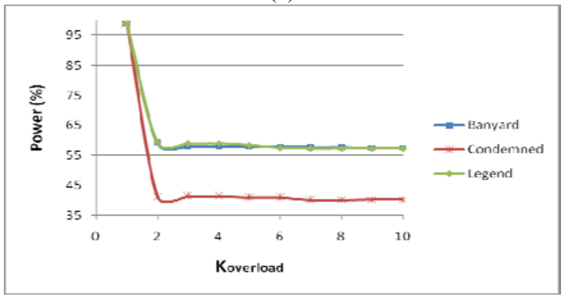
To assess the efficiency of the proposed frequency scheduling method, we applied the fixed update interval [13], adaptive update interval, and oracle DVFS methods to a workload set obtained from real workloads based on realistic computational load of the MPEG2 decoder and two packet-processing applications. We have used MIPS-based SimpleScalar [18] simulations to determine the computational load of the MPEG2 decoding in terms of instructions per frame. The computational complexities of the packet-processing applications were also extracted from [20].The packet-processing applications which were selected from the PacketBench [21] included an IPv4 look-up (IPv4-trie) and flow classification (Flow-class) algorithms that are widely used in network processing nodes. We have also exploited a method the same as Wattch [19] for estimating the power consumption based on the effective capacitance concept.

The DVFS algorithms and the power estimation methods were modeled in Verilog and simulated in Modelsim. The MPEG2 decoder workload is obtained from decoding three different MPEG2 movies. The computational load is related to the number of instructions required for decoding a frame.

In ADAPTIVE, there are four parameters whose values determine the power saving capability and deadline misses (responsiveness) of the system. The parameters are the size of the workload history ($k_{history}$), *overload*/*underload* history ($k_{overload}$/$k_{underload}$), and the *Interval step* (k_{step}). For MPEG decoding workloads, we used some simulations to obtain the best values for the parameters. The priority of finding the values of the parameter was based on their effects on the power and deadline misses. The results showed that the most effective parameter was $k_{overload}$. Figure 3 shows the deadline misses and power consumption of the selected movies as a function of the $k_{overload}$. In the MPEG2 decoder, frames should be processed in 33µs. Each frame-processing time violating this specified period causes a deadline miss. As shown in Figure 3, a good value for $k_{overload}$ is between 2 and 7, and hence, the overload detection circuit can be implemented with a 3-bit CRH1 counter leading to the deadline misses in the range of 5%-15%, and the power consumption reduction of 40%-60%.



(a)

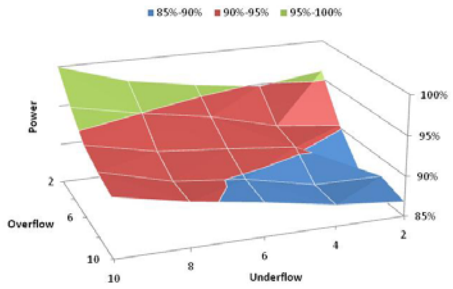


(b)

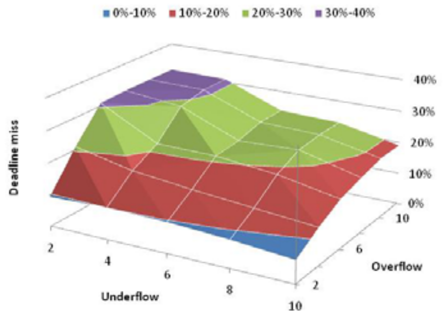
Figure 3. Power consumption and deadline miss of the selected movies versus $k_{overload}$ (a) deadline miss, (b) power.

The next parameter that is tuned is the $k_{underload}$. As the results reveal, when $k_{underload}$ is around 10, the deadline miss is about 6% and the power saving is nearly saturated. This value needs a 4-bit CRH1 counter for underload detection. Finally, we have found that setting $k_{history}$ to 3200 reduces the deadline miss to below 2% which requires a 12-bit CRH counter. The last parameter of the system is k_{step} which is used for properly tuning the interval value. We have found the power consumption and deadline miss of the selected movies as a function of k_{step}. Since the power consumption is a weak function of k_{step}, this parameter may be used for fine tuning of the deadline misses.

Our study showed that $k_{overload}$ and $k_{underload}$ were the most effective parameters. Figure 4 shows the effect of these parameters on the power consumption and deadline misses of one of the selected movies. The power values are normalized to the maximum values of the power consumption. We can observe that higher $k_{underload}$ and lower $k_{overload}$ values yield lower deadline misses (more responsiveness) while lower $k_{underload}$ and higher $k_{overload}$ values give rise to lower power consumptions.



(a)



(b)

Figure 4. Power consumption and deadline misses of the Legend movie for a wide range of $k_{overload}$ *and* $k_{underload}$ values. (a) power consumption, (b) deadline miss.

We have tuned the parameters of ADAPTIVE based on MPEG workloads and used these parameters for the packet-processing applications as well. Therefore, we have used the results of Figure 4 and set the values of $k_{overload}$ and $k_{underload}$ to 2 and 6, respectively. The other parameters including $k_{history}$ and $k_{step,}$ were also set to 1000 and 5, respectively. In Figure 5, we have compared the power consumptions and the frequency update rates of the FIXED, ADAPTIVE, and oracle (ideal) techniques for the MPEG2 decoding and two packet-processing applications.

With the oracle we mean an ideal DVFS algorithm in which the workload computational complexity is known at the beginning of processing and the frequency is set to the optimum value accordingly. Therefore, the update rate of oracle is one update per frame and the deadline miss is zero. Besides, FIXED and ADAPTIVE parameters were set such that the deadline misses of both systems were the same. As shown in Figure 5, on average, ADAPTIVE leads to about 12% more power savings, compared to that of the FIXED, while the update rate of ADAPTIVE is also about 2.6 times lower than that of the FIXED. According to the presented results in [13], exploiting the DVFS scheme does not increase the power consumption of subsystems such as memory. Therefore, excluding the memory power consumption does not affect the results.

V. Conclusion

In this work, an efficient and adaptive update interval method for dynamic voltage and frequency management was proposed. In this method, the frequency and voltage of the system for the periodic workloads were scheduled while maintaining soft real-time deadlines. The saving was achieved by introducing the concept of the effective deadline and taking advantage of the possible correlation between the consecutive workloads. To show the efficacy of the method, comparisons between oracle, adaptive, and fixed interval DVFS systems were performed using realistic workloads of the MPEG2 decoder and packet-processing applications. The results showed that the proposed adaptive interval DVFS technique could save power more with fewer frequency updates as compared to the fixed interval DVFS systems.

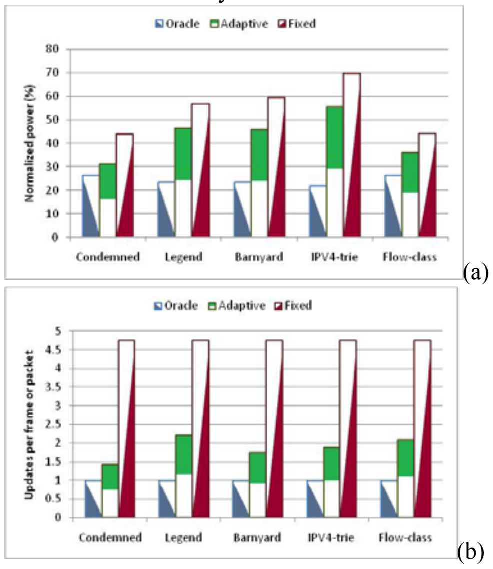


Figure 5. (a) Power consumption and (b) updates rates per frame or packet for the realistic workloads in FIXED, ADAPTIVE, and oracle techniques.

References

[1] B. C. Mochocki, X. S. Hu, and G. Quan, "A Unified approach to variable voltage scheduling for nonideal DVS processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 23, no. 9, Sept. 2004, pp. 1370-1377.

[2] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "A self-tuning DVS processor using delay-error detection and correction," *IEEE Journal of Solid-State Circuits,* vol. 41, no. 4, April 2006, pp. 792-804.

[3] K. Choi, R. Soma, and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 24, no. 1, January 2005, pp. 18-28.

[4] T. Burd, T. Pering, et al., "A Dynamic voltage scaled microprocessor system," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, Feb. 2000, pp.294-295.

[5] K. J. Nowka, G. D. Carpenter, et al., "A 32-bit PowerPC system-on-a-chip with support for dynamic voltage scaling and dynamic frequency scaling," *IEEE Journal of Solid-State Circuits,* vol. 37, no. 11, Nov. 2002, pp. 1441-1447.

[6] K. Flautner, D. Flynn, et al., "IEM926: an energy efficient SoC with dynamic voltage scaling," *in proc. of the Design, Automation and Test in Europe Conference and Exhibition Designers' Forum,* Feb. 2004, pp. 324-329.

[7] J.R. Lorch, A.J. Smith, "PACE: A new approach to dynamic voltage scaling," *IEEE Transactions on Computers,* vol. 53, no. 7, July 2004, pp. 856-869.

[8] M. Marinoni and G. Buttazzo "Elastic DVS management in processors with discrete voltage/frequency modes," *IEEE Transactions on Industrial Informatics,* vol. 3, no. 1, February 2007, pp. 51-56.

[9] S. Liu, Q. Qiu, and Q. Wu, "Energy aware dynamic voltage and frequency selection for real-time systems with energy harvesting," *in proc. of Design, Automation and Test in Europe,* 2008, pp. 236-241.

[10] J. Wegener and F. Mueller. "A comparison of static analysis and evolutionary testing for the verification of timing constraints," Real-Time Systems, 21(3):241–268, Nov. 2001.

[11] M. Elgebaly and M. Sachdev, "Variation-aware adaptive voltage scaling system," *IEEE Transactions on Very Large Scale Integration (VLSI) systems,* vol. 15, no. 5, May 2007, pp. 560-571.

[12] S. Chandra, K. Lahiri, A. Raghunathan, and S. Dey, "Variation-tolerant dynamic power management at the system-level," *IEEE Transactions on Very Large Scale Integration (VLSI) systems,* vol. 17, no. 9, September 2009, pp. 1220-1232.

[13] M. Nakai, S. Akui, et al., "Dynamic voltage and frequency management for a low-power embedded microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 1, Jan. 2005, pp 28-35.

[14] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," *in proc. of Int'l Symp. on Low Power Electronics and Design,* Aug. 2007, pp.38-43.

[15] Wu, Q., Juang, P., Martonosi, M., and Clark, D.W.: "Formal Online Methods for Voltage/Frequency Control in Multiple Clock Domain Microprocessors," *in proc. of ASPLOS-XI,* Oct. 2004, pp. 248-259.

[16] P. Choudhary and D. Marculescu, "Power management of voltage/frequency island-based systems using hardware-based methods," *IEEE Transactions on Very Large Scale Integration (VLSI) systems,* vol. 17, no. 3, March 2009, pp. 427-438.

[17] M. Najibi, M. Salehi, A. Afzali Kusha, M. Pedram, S. M. Fakhraie, and H. Pedram "Dynamic voltage and frequency management based on variable update intervals for frequency setting," *in proc. of IEEE/ACM International Conference on Computer-Aided Design,* Nov. 2006, pp. 775-760.

[18] D. Burger and T. Austin, The SimpleScalar tool set version 2.0, Computer Architecture News, 25, (3) (1997) 13–25, June.

[19] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," *In proc. of the 27th International Symposium on Computer Architecture*, June 2000, pp. 83-94.

[20] M. E. Salehi and S. M. Fakhraie, "Quantitative analysis of packet-processing applications regarding architectural guidelines for network-processing-engine development," *Journal of System Architecture,* vol. 55, no. 7-9, July-September 2009, pp. 373-386.

[21] R. Ramaswamy, N. Weng and T. Wolf "Analysis of network processing workloads," *Journal of System Architecture, (2009), 10.1016/j.sysarc.2009.09.001*.