Dynamic Frequency Scaling for MPSoCs based on Chaotic Workload Analysis

Nikolaos Zompakis¹, Vasilios Tsoutsouras², Alexandros Bartzas¹, Dimitrios Soudris¹, Georgios Pavlos²

¹ ECE School, National Technical University of Athens, 15780 Zografou, Greece
² ECE Department, Democritus University of Thrace, 67100 Xanthi, Greece

Abstract

Modern applications running on Multi-Processor Systems-on-Chip vary their workload dynamically. This gives the opportunity to the designers to perform dynamic frequency scaling (DFS). Thus the system adapts to the workload requirements of the application and increases its power efficiency. In this work, we present a DFS technique based on the workload trend of a dynamic application. The designer can adjust the frequency of the system by analyzing the workload fluctuations without degrading the final performance or violating any deadlines. To achieve this we employ an abstract model of workload analysis that combines advanced mathematical tools from the Chaos Theory domain, allowing us to handle dynamic data streams with complex behavior. To evaluate the efficiency of the proposed approach we applied it using a workload from a real application on a cycle-accurate Network-on-Chip simulation framework. The simulation results showed that the proposed technique can achieve remarkable improvements at the final power consumption, between 17.5% and 37.8%, depending on the system constraints.

1 Introduction and Related Work

In recent years, the mobile technology has developed so dramatically that has opened new challenges in the embedded system design domain. Different from the traditional desktop systems, embedded devices demand not only high performance but also low power consumption. Thus, the development of proper methods in power consumption management, which will extend the battery life, is without doubt, an imperative need. Furthermore, embedded applications often are time-constrained, which means that their tasks must be completed before specific deadlines.

In addition, embedded systems sometimes experience transient overloads due to hardware malfunctions or workload bursts. For that reason such systems have to be designed to take timely reactions to the occurrences of unexpected usage scenarios. The development of smart techniques that focus the available computing power on these urgent events and, at the same time, to slow down the processing during inactive periods could be the key for preserving energy at mobile devices.

The majority of processors are able to support multiple power modes that enable trading off execution time for energy savings. For example, some mobile processors can alter their frequency and energy consumption at run-time. In this way, they are able to achieve significant power savings by scaling their voltage levels and frequency values. Scaling the supply voltage means scaling the operating frequency accordingly. From another perspective, to every working frequency, there should be a corresponding lowest supply voltage to support this frequency properly while minimizing the power consumption. In this paper, we perform only frequency scaling.

Dynamic frequency scaling (DFS) is used to adjust the working frequency according to the system workload in order to save the power consumption without degrading the system performance significantly beyond the application tolerance. The main problem of DFS solutions is how to compute the system workload trend. Generally, workload analysis is a research issue that has occupied the literature since the early days of the computer science [1].

As a matter of fact, while in running mode, the system workload varies from time to time. If the power consumption is reduced when the workload is low, then the overall energy consumption of the embedded system can be further decreased. Correspondingly, when the system workload becomes higher, the CPU voltage will be increased to satisfy the necessary deadlines. In recent years, few researchers have proposed some new workload modeling methodologies [2]. These methodologies are based on the construction of novel frameworks, which can monitor and predict the progress of the workload traffic. An analysis of system performance degradation induced by workload fluctuations is presented in [3]. That analysis takes into account the interaction between workload fluctuations and the nonlinearity of the system. The work described in [4] presents a performance case study of parallel jobs capable of predicting the completion time distribution of the executing jobs under real workloads and the effects of system...
design changes on application performance. Furthermore, advanced mathematical approaches like statistical clustering, advanced Markovian models and time series analysis [5] are used. Some studies go even further by proposing adaptive techniques related with dynamic power management [6] and dynamic voltage and frequency scaling [7] exploiting the information about the future state of the system by the workload prediction analysis.

The main contribution of our work is that we propose a systematic way to analyze real dynamic application workloads with ostensibly chaotic and non-ascertainable behavior at the system-level. The goal of our approach is to predict the “critical points” that represent the unstable states of a computing system applying power efficient frequency scaling policies. The proposed methodology employs advanced mathematical tools from the Chaos Theory domain integrates them in a complete frequency scaling technique. These tools specialize at the analysis of non-linear flows with very high complexity.

The rest of the paper is organized as follows. An overview of the proposed methodology is presented in Section 2. The workload analysis techniques based on Chaos Theory are presented in Section 3, whereas the DFS methodology is presented in Section 4. We evaluate the proposed technique using a complex dynamic application on an NoC cycle-accurate simulator and the results are presented in Section 5. Finally, conclusions are drawn in Section 6.

2 Methodology Overview

An overview of the proposed methodology, divided in three separate stages, is presented in Figure 1. In the first stage (Step (1) of Figure 1) the workload analysis, which in turn is divided in ten separate steps, is performed. This allows the designer to gain an insight of the workload behavior (e.g., predict the trend of the time series). At the second stage, the system-level DFS strategy is defined. In order to do this the designer takes into consideration the frequency ranges supported by the platform, the application constraints and the results of the workload analysis (such as the critical points) (Steps (2) and (3) of Figure 1). At the third stage, the predictions and the DFS policies are evaluated using an NoC simulation environment [8] (Step (4) of 1). NoC systems represent an emerging paradigm for on-chip communications within large multi-core VLSI systems [9].

3 Workload Analysis Methodology

The aim of the proposed methodology is to extract the application’s critical points. We define as critical points the points where the dynamics of the system’s workload change. The goal of this analysis is to isolate and predict these regions giving the ability to the system to utilize the available resources while reducing the power consumption. To achieve this, analysis tools that belong to the Chaos Theory domain are employed. Today chaos is synonymous to complexity. Low dimensional chaos is one of the central properties of complex (non-linear) systems, the appearance of which is accompanied by self organization properties and correlations in space and time [10]. The external driving (tuning) of the complex system (change of control parameters) causes bifurcation at critical points from one kind of attractor to the other. Bifurcation points or critical points are the limit between different dynamical profiles such as stability (limit point), periodicity or randomness (chaoticity). Spatially distributed dissipative nonlinear systems, discrete or continuous (in space or/and time) can reveal spatiotemporal chaos and criticality. The spatiotemporal chaos can be manifested with a wealth of spatiotemporal structures and self organized processes such as: long range spatial order, first or second order phase transitions critical dynamics and Self Organized Criticality, directed percolations structures, local instabilities (bursts) and intermittency chaos which are some of the most significant possible processes of distributed complex systems.

3.1 Steps of the Workload Analysis

The proposed workload analysis is based on modern non-linear time series analysis proposed by Pavlos et al. [11]. The main purpose of time series analysis is to extract significant information for the underlying dynamics of the observed application/system, as well as to dynamics and the
effectiveness of the observed methods for modeling and prediction.
In the following we suppose that the workload bursts are caused by the internal dynamics of the entire system according to the general mathematical form:

$$\frac{d\vec{X}}{dt} = \vec{F}(\vec{X}, \lambda, w)$$  \hspace{1cm} (1)

where $\vec{X}(t)$ is the state vector of the underlying workload system, $\lambda$ is an external parameter of the dynamics and $w(t)$ is a noise component. For different regimes of $\lambda$ we can observe different dynamics: stable, unstable, periodic or chaotic. $\vec{F}$ describes the flow of the dynamics in the state space. Every observed signal $y(t_i)$ depends on the state vector $y(t_i) = f(\vec{X}(t_i))$.

The steps of the workload analysis applied in the context of the proposed methodology are:

1. **Extract time series** Initially extract the time series of the input that triggers the application behavior.

The time series under study is the size of packages that are sent over a period of time $T$. More specifically, the time series used in the context of this work consists of 868,000 measurements. This time series is split in 217 sections, where each section contains 4000 measurements. For each of these sections we have implemented the tools that perform the: a) autocorrelation function; b) Shannon entropy; c) signal power; d) Hurst exponent; e) correlation dimension; and f) forecast.

2. **Analysis in the time and frequency domain** The power spectrum $P(\omega)$ and the autocorrelation function $C(\tau)$ of the signal are related according to the Wiener-Khintchine theorem:

$$P(\omega) = \int_{-\infty}^{+\infty} C(\tau) e^{-i\omega\tau} d\tau.$$  \hspace{1cm} (2)

The power spectrum $P(\omega)$ of a scalar signal can show whether a system is periodic or quasi-periodic, but it cannot provide any information for broadband spectrum. Also, classical time series analysis cannot discriminate between the two significant cases: a) high dimensional linear or non-linear stochastic dynamics and b) low-dimensional chaotic dynamics, while in both cases we can observe broadband power spectrum.

3. **Calculation of the Hurst exponent** The Hurst exponent may be defined by the scaling properties of the time series according to the equation:

$$< |y(t + \tau) - y(t)|^q >_{T} \sim \tau^{qH(q)}$$  \hspace{1cm} (3)

where $q > 0$ and $T$ is the time lag. The averaging is over $T$ with $T >> \tau$. The values of Hurst exponent range between 0 and 1 corresponding to time series derived by fractional Brownian motion. The case $H = 1/2$ is the ordinary Brownian motion (white noise), which has independent increments. For $H > 1/2$, there is positive correlation between temporal time series increments. For $H < 1/2$, there is a negative correlation between temporal time series increments. The positive autocorrelation corresponding to $1/2 < H < 1$ indicates persistent behavior of the time series, while for $H$ values between $0 < H < 1/2$ the time series reveals anti-persistent behavior (negative autocorrelation) [12].

4. **Flatness coefficient $F$** The flatness coefficient $F$ of the observed time series shows the deviation from the Gaussian distribution corresponding to purely random data. It can be estimated by using the two-points differential time series according to the equation:

$$F = \frac{\delta y_{\tau}(t)^4}{\delta y_{\tau}(t)^2}$$  \hspace{1cm} (4)

where $\delta y_{\tau}(t) = y(t + \tau) - y(t)$. For random Gaussian process the coefficient $F = 3$. Chaotic or intermittent characteristics appear in the heavy tails of the distribution functions. In this case $F > 3$.

5. **De-correlation time** In each section, out of the 217, in the time series the autocorrelation function was applied. This gives the designer information on whether there are linear correlations and periodicities in the signal. The autocorrelation function gives similar information to the power spectrum Fourier, but on the time-domain. From the autocorrelation function defined the de-correlation time of the signal. The de-correlation time of time series is the time for which the time series exhibits no periodicity. The autocorrelation function, provide only the linear correlation of the signal with itself therefore does not control the non-linear relationship.

6. **Correlation Dimension** The correlation dimension $D$ reveals the dynamical degrees of freedom of the underlying dynamics and can be estimated in the reconstructed state space according to the equation:

$$D = \lim_{\tau \to 0, m \to \infty} \frac{d \ln C^m(r)}{d \ln(r)}$$  \hspace{1cm} (5)

where $C^m(r)$ is the so-called correlation integral for a radius $r$ in the reconstructed phase space. When an attracting set exists, then $C^m(r)$ reveals a scaling profile ($C^m(r) \sim r^{d(m)}$). The correlation integral depends on the embedding dimension $m$ of the reconstructed phase space and is given by the following equation:

$$C(r, m) = \frac{2}{N(N - 1)} \sum_{i=1}^{N} \sum_{j:i+1}^{N} \Theta(r - ||\xi(i) - \xi(j)||)$$  \hspace{1cm} (6)
where \( \Theta(\alpha) = 1 \) if \( \alpha > 0 \), \( \Theta(\alpha) = 0 \) if \( \alpha \leq 1 \) and \( N \) is the length of the time series. The scaling exponent \( d(m) \) increases as we increase the embedding dimension \( m \). When the time series is related to a low dimensional dynamical system then \( d(m) \) saturates at a final value \( D \) for a sufficiently large embedding dimension.

7. Estimation of the Lyapunov exponents spectrum

Chaotic dynamics is caused by positive Lyapunov exponents. The spectrum of Lyapunov exponents measures the rate of convergence or divergence of close trajectories in all \( d \) directions of the phase space [13]. The spectrum of the Lyapunov exponents can be estimated from a time series by following the evolution of small perturbations of the reconstructed orbit, making use of a linearized approximation. The evolution of the displacement vector between the neighboring points \( y(i) \) and \( y(i) + w(i) \) in the reconstructed phase space is given by the equation:

\[
w(i + 1) = D\tilde{F}(x(i))w(i) \tag{7}
\]

where \( D\tilde{F} \) denotes the derivative matrix of \( \tilde{F} \).

8. Estimation of Shannon Entropy

Estimation of mutual information and Shannon entropy in chaotic or stochastic dynamical systems can be described by using the concept of information. For this scope, we suppose that the random behaviour of the system is a realization of Shannon’s concept of an ergodic information source. If \( S \) is some property of the dynamical system and \( s_i, i = 1, 2, \ldots \) possible values of \( S \), then the average amount of information gained from a measurement that specifies \( S \) is given by the entropy \( H(S) \).

\[
H(S) = - \sum_i P(s_i) \log P(s_i) \tag{8}
\]

where \( P(s_i) \) is the probability that \( S \) equals \( s_i \) and the logarithm is taken with respect to base 2. An estimate of \( P(s_i) \) is given by \( n(s_i)/nT \), where \( n(s_i) \) is the number of times that the value \( s_i \) is observed, and \( nT \) is the total number of measurements.

9. Modeling and prediction

The observable information \( \tilde{X}(t_i) \) on the temporal evolution of the orbit in the reconstructed state space can be used for prediction or modeling purposes. This can be achieved by building the predictor map \( F(\tilde{X}(t_i)) \) for \( T \) time steps ahead according to the equation:

\[
\hat{y}(t_i + T) = F^T(\tilde{X}(t_i)) \tag{9}
\]

where \( \hat{y}(t_i + T) \) corresponds to the predicted values of the observed time series \( y(t_i) \). The predictor map \( F \) constructed in the mirror state space may be approximated with different functional forms of the global-, local- or semi-local time.

10. Surrogate data and statistics construction

According to equation 8, the scaling properties of the correlation integral as \( r!0 \) and the saturation of the scaling exponent \( d(m)/D \) as \( m \) increases are necessary conditions for the existence of low dimensional dynamics underlying the experimental time series. Moreover, the concept of low correlation dimension (fractal or integer) can be applied to time series in two distinct ways [14]. The first one indicates the number of degrees of freedom in the underlying dynamics, and the second quantifies the self-affinity or “crinkliness” of the trajectory through the phase space.

The method of “surrogate” data includes the generation of an ensemble of data sets which are consistent to a null hypothesis. According to [14], the first type of null hypothesis is the linearly correlated noise which mimics the original time series in terms of the autocorrelation function, variance and mean. The second and more general null hypothesis takes into account that the observed time series may be a nonlinear monotonic static distortion of a stochastic signal. Therefore, it is a statistical problem to distinguish between a nonlinear deterministic process and a linear stochastic process. For this purpose, we use as discriminating statistic a quantity \( Q \) derived by a method sensitive to nonlinearity, as the correlation dimension estimation. The discriminating statistic \( Q \) is calculated for the original and the surrogate data, and the null hypothesis is verified or rejected according to the value of “sigmas” \( S \)

\[
S = \frac{\mu_{obs} - \mu_{sur}}{\sigma_{sur}} \tag{10}
\]

where \( \mu_{sur} \) and \( \sigma_{sur} \) is the mean and the standard deviation of \( Q \) on the surrogate data, and \( \mu_{obs} \) is the mean of \( Q \) on the original data. For a single time series, \( \mu_{obs} \) is the single \( Q \) value [14].

4 Dynamic Frequency Scaling Methodology

The workload analysis approach is the basis of the proposed run-time frequency scaling technique. Using the proposed technique the designer can analyze the trend of the workload without knowledge of the origin of the input data, such as application tasks or dependencies. This offers the necessary flexibility that is required to build a high-sensitivity run-time mechanism, which traces the fluctuation of the workload flow.

The key point is the prediction of the data transfer size and the “gap time” between the data transfers. This prediction is a very demanding task for the classical mathematics especially for dynamic applications. However, the complexity becomes much lower when we use advanced chaotic
mathematical tools. Such mathematical tools provide reliable forecast models for ostensibly accidentally events transfers. This knowledge can be extracted either at design time, as a library of our DFS module analyzing the application of our interest, or at run-time giving feedback for the behavior of the running application improving the response of our DFS mechanism. In both cases, we extract the critical points (which represent the points where the behavior of our workload is remarkable change) based on which we can scale the operational frequency of the processors according to the needs of our system.

For the optimum exploitation of the workload analysis we follow simple scaling rules which satisfy the applications requirements with the minimum resources consumption. As it is showed in Figure 1, after the workload analysis and the extraction of the critical points the DFS strategy is specified, defining the frequency modes, which will represent the system responses at the workload fluctuation. To define these configurations, we initially choose the lowest frequency, from the operational frequencies supported by the system, which satisfied all timing constraints of the application workload. It is obvious that this frequency represents the low-power consumption configuration for a system without DFS capabilities. However, in a DFS-capable system such a frequency is the high-performance one, representing the high-frequency threshold and will be used only for the high-demanding parts of the workload (thus leading to high power consumption). Correspondingly, for the less-demanding parts of the workload lower operational frequencies are chosen, which satisfy the corresponding deadlines, and thus achieving lower power consumption.

The goal of the proposed frequency scaling strategy is to adapt dynamically the frequency at the lowest level, following the workload fluctuations, without violating any timing constraints. The proposed workload analysis offers the ability to implement the goal of detecting, predicting the density of the traffic, splitting the workload in high- and low-demand regions, and finally choose the appropriate frequency for each one of them.

5 Case Study

In order to validate our approach, we have modeled the following concurrent threads, which are triggered by wireless streams in a Linux multi-threaded environment [16], relevant for embedded systems (each application kernel is executed on its own independent thread and communicates asynchronously with the other threads. All the communication queues have locking mechanisms to ensure proper synchronization between threads):

- Simulated VoIP, FTP and Web browsing activity (as measured in [18]). This thread feeds into the test-bench each packet of the input trace.
- TCP/IP packet formation (it builds the complete TCP/IP packet filling in the header fields). Reflecting the entry point to the operating system like a write() system call, this thread builds the complete packet filling in the header fields that are available (source and destination IP addresses and port numbers).
- Encryption (packets that belong to an encrypted connection are processed with the DES algorithm).
- TCP checksum. Calculated applying the 16 bit one’s complement sum to the whole TCP packet and the so-called “IP pseudo header” as described in [17].
- The QoS manager builds a prioritized list of destinations. When a packet arrives to this subsystem it is queued in one of the priority classes. Packets are extracted from them and forwarded to the network adaptor according to a simplified Deficit Round Robin (DRR) algorithm.

More information about this application and the network traces can be found in [15]. The case study was evaluated on a cycle-accurate Network-on-Chip (NoC) simulation environment [8]. The system under test is composed of four processing elements on a mesh topology. The platform supports four frequencies (200, 400, 600 and 800 MHz).

5.1 Workload Analysis

Here we present indicatively the realization of steps 1, 5, 6 and 9 described in Section 3. Figure 2 presents the data (a), their spectrogram (b), the slopes of the correlation dimension for emending dimensions $m = 6 \rightarrow 10$ compared with the slope of the surrogate data (c), the autocorrelation function (d) and the probability distribution function (PDF) of the package size time series (e).

Figure 3, is similar to Figure 2 but for another period during which the correlation dimension is different. During the period corresponding to Figure 2 the correlation dimension is high ($> 8$). The distribution function for this period reveals small probability for large size package. In contrast with these characteristics corresponding to Figure 2 the period presented in Figure 3 reveals low value ($\sim 2$) saturation of the slopes (indicating low value of the correlation dimension) and equal probabilities for small and large size packages.

Figure 4 presents the slope of the correlation dimension estimated for successive elements of a long size package time series consisted of 868,000 measurements (black line) and its predicted data (red line). Figure 4b presents the estimated ratio of the distribution functions of small to large data transfer sizes.

5.2 Power Consumption Evaluation

The first step that the designer has to perform is to define the initial application timing constraints. To study the in-
Figure 2: (a) Section of data transfer size time series (4000 values), (b) Short Term Fourier Transformation â STFT, (c) Slopes of the Correlation Integral estimated for the packet size series for parameters $m = 6 - 10$, $\tau = 10$, $w = 10$ and its Surrogate (red line) using $m = 10$, $\tau = 10$, $w = 10$, (d) autocorrelation function, (e) Probability distribution functions (PDF).

Figure 3: (a) Section of data transfer size time series (4000 values), (b) STFT, (c) Slopes of the Correlation Integral estimated for the packet size series for parameters $m = 6 - 10$, $\tau = 13$, $w = 13$ and its Surrogate (red line) using $m = 10$, $\tau = 13$, $w = 13$, (d) autocorrelation function, (e) PDF.
fluence of timing constrains of the proposed methodology we fixed some deadlines. Initially, we defined a hard deadline calculated as the maximum latency time (30 ms) for a transfer at the high performance mode, which in our case was running at 800 MHz. Figure 5 presents the power consumption evaluation results, with and without DFS. The values are normalized based on the power consumption of the system without using DFS (first column). The second column shows the power consumption of the system when the DFS strategy is used, achieving gains of 17.5%, without violating any timing constraints. The power consumption of the system when we relax the timing constraints by 5%, 10%, 15% and 20% is shown in the other columns (light grey) of Figure 2. By relaxing the constraints (whenever is that possible) we can achieve gains of 21.1%, 25.5%, 29.2% and 37.8% respectively.

6 Conclusions

In this paper, we propose a novel system-level DFS strategy predicting the workload trend based on advanced mathematical tools from the Chaos Theory domain. The benefit of our approach is that we can analyze workload with high complexity without requiring knowledge of the specifications of the target application. To evaluate the effectiveness of our system-level approach we applied our methodology simulating a complex dynamic multi-threaded application in a cycle-accurate NoC simulator. Using the proposed DFS strategy the designer can achieve remarkable power consumption improvements, ranging from 17.5% (hard timing constraints) up to 37.8% (relaxed timing constraints), depending on the timing constraints of the application.

References

Figure 5: Power Consumption with and without DFS


