Energy Management via PI Control for Data Parallel Applications with Throughput Constraints

Anca Molnos, Warody Lombardi, Diego Puschini, Julien Mottin, Suzanne Lesecq, Arnaud Tonda

Energy Management via PI Control for Data Parallel Applications with Throughput Constraints

Anca Molnos, Warody Lombardi, Diego Puschini, Julien Mottin, Suzanne Lesecq, Arnaud Tonda
Univ. Grenoble Alpes, Grenoble, France. CEA - LETI, MINATEC Campus, Grenoble, France
Email: anca.molnos@cea.fr

***Abstract*—This paper presents a new proportional-integral (PI) controller that sets the operating point of computing tiles in a system on chip (SoC). We address data-parallel applications with throughput constraints. The controller settings are investigated for application configurations with different QoS levels and different buffer sizes. The control method is evaluated on a test chip with four tiles executing a realistic HMAX object recognition application. Experimental results suggest that the proposed controller outperforms the state-of-the-art results: it attains, on average, 25% less number of frequency switches and has slightly higher energy savings. The reduction in number of frequency switches is important because it decreases the involved overhead. In addition, the PI controller meets the throughput constraint in cases where other approaches fail.**

I. Introduction

Reducing energy consumption in embedded computing systems is still a challenge, despite many recent advances on the subject. Recent hardware platforms integrate multiple processor cores on a chip and support operating point switching, e.g., dynamic frequency and/or voltage scaling (DVFS) at the level of processor cores [1], clusters [2] or entire chip [3], [4]. Taking scaling decisions is a complex issue because variable workload demands have to be taken into account, such that application performance is not negatively impacted. An overview of operating point scaling methods at system level is presented in [5].

Among these methods, some employ controllers with a feedback loop because of their abilities to adapt to workload dynamics. Operating points are tuned by looking either at the filling of hardware queues [6], [7], [8], [9] or software queues, e.g., at task scheduler [10], at inter-task communication [11], [12]. Existing work proposes low overhead, simple controllers, such as proportional integral derivative (PID) [10], [13], [12] and non-linear [11] ones, as well as complex custom controllers [7], [8], [9]. Significant energy reductions are reported. The experiments are mostly performed in simulation with the exception of two approaches that target high-performance chips such as Intel's SCC [14] and hexa-core Xeon platforms [15].

Nevertheless, most existing methods do not fully address applications that have throughput constraints, which are typical in domains like, computer vision, image and signal processing. Furthermore such applications may have different QoS demands and, depending on the available memory resources of the hardware platform, their buffers may have different sizes. Larger buffer sizes may accommodate larger variations of the workload while allowing the application to still meet the throughput constraint. All these aspects need to be considered when designing a controller that tunes the operating points of hardware blocks.

To this end, the first contribution of this paper is a PI-based controller that addresses data-parallel applications with throughput constraints. These applications execute on a tiled architecture, where each tile is placed in a voltage-frequency island that can be controlled independently. Here we control the speed of each tile, i.e., its clock frequency, assuming that the lowest voltage supply necessary for a given frequency is set by existing methods [16]. The controller has two distinct regions, an energy management one where the PI is applied, and a critical one where the tiles will run at their maximum frequency to avoid throughput degradations. The evaluation experiments are performed on a test chip following a tiled multi-VFI architecture [2], with a realistic HMAX object recognition application [17]. The evaluation metrics are: (1) energy reduction (we consider as baseline an execution at maximum frequency), (2) number of frequency switches (often frequency switches bring energy and performance overhead), (3) degree of throughput degradation (here, ideally the controller should always meet the constraint), and (4) the controller overhead. All timing-related numbers, e.g., throughput, number of frequency switches, are realistic and include overheads, e.g, of frequency switch. Energy reductions are estimated off-line using models constructed from physical chip measurements.

The second contribution of this paper is an in-depth experimental analysis of the sensitivity of the PI controller to different configurations. This analysis is meant to determine the best settings and led to the definition of three controller operating intervals, referred to as, the high-amplitude variation, the low-amplitude variation, and the saturation interval. The interval containing the best controller settings depends on the QoS levels and buffer sizes of the application.

The third contribution is a comparison against the performance of an existing non-linear controller with two variants. Experiments indicate that for our use-case, the PI controller attains on average 25% less number of frequency switches, leads to slightly better energy savings than the two non-linear ones. In addition, the PI controller meets the throughput constraint in cases where the other two approaches fail. All investigated controllers have low overhead.

In the rest of this paper we first present the background information in Section II, followed by the model of our system in Section III and the proposed PI controller. Section IV covers the experimental results, and Section V discusses the related work. Finally Section VI concludes the paper.

II. Application, architecture

This section covers preliminary information, namely the SoC architecture template and the application model, required to understand this work.

A. Tiled, GALS architecture template

We consider a tiled SoC that follows a general template of Globally Asynchronous Locally Synchronous (GALS) architecture, such as the existing chips [1], [2]. Figure 1 illustrates this template. Each tile is placed in a separate voltage-frequency island (VFI) and hence its speed and energy consumption can be tuned independently of other tiles. The tiles are connected via an asynchronous network on chip (NoC) that ensures communication. A tile comprises one or more cores, i.e., processor elements (PE), as well as level one memory blocks (L1) and hardware blocks for communication (DMA) and inter-tile synchronization. The tiles may share a level two on-chip memory (L2), accessible over the NoC.

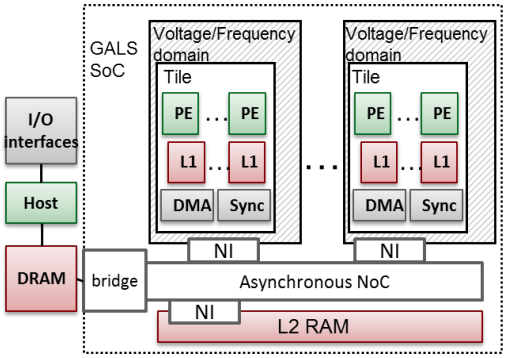


Fig. 1. Tiled SoC template

The SoC is connected with a host processor, which is responsible with starting and stopping the execution of the tiles and accessing the I/O interfaces. From this perspective, the multi-/many-core can be regarded as a accelerator for the host. The host and the SoC share a high-speed main memory.

B. Data parallel application model

In many domains, e.g., computer vision, image processing, robotics, the application model is data-flow, namely a set of tasks that process a continuous stream of incoming data. These tasks communicate within each other through buffers. Such applications are typically constrained by a throughput demand. This constraint is expressed in, e.g., samples, pixels, entire images, required to be available at each time unit.

Here we address data-flow applications following a common structure with a source task, multiple data parallel worker tasks, and a sink task, as described in Figure 2. The source and the sink tasks are not involved in heavy processing; the source distributes the work (and input data) to the worker tasks and the sink collects the results. We assume that the source task attempts a simple work balancing strategy by splitting the input data in pieces of similar size before sending them to the worker tasks. The worker tasks process input data and store the results in output buffers.

The main concern of the sink node is to respect the throughput constraint, i.e., to make available an output token

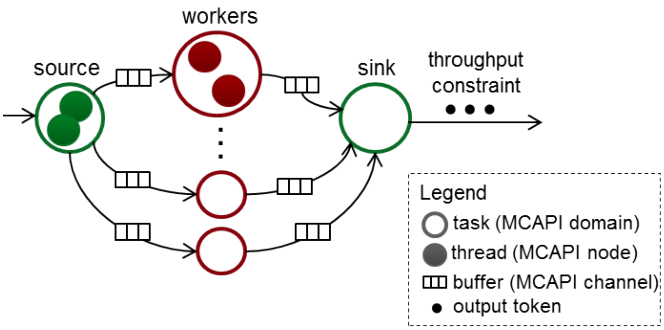


Fig. 2. Data parallel application model

each unit of time. The exact definition of an output token and its size is application dependent. We assume that the sink is not required to collect workers' results in any particular order, meaning it does not block if one of the workers is slower than the others. However, to complete an output token, the sink requires all the tasks that work on that token to have finished. Multiple tokens may be in processing at a given time moment at the worker nodes.

In case that the throughput constraint is not met, i.e., the workers produced insufficient data for the sink, the partially computed output token is discarded. Other scenarios can be implemented here, however we consider this as the most realistic for streaming applications. We denote this shortly as a skipped output. Note that we intentionally avoid the term deadline miss, as we do not address hard real-time applications where a deadline miss leads to severe consequences, but rather applications that have a soft throughput constraint.

C. Implementation

The application is implemented with the Multicore Association Application Programming Interface (MCAPI) [18]. The MCAPI specification defines an API and a semantic for communication and synchronization between processing cores in many-core embedded systems. Our work is not limited to MCAPI, as any other data-flow implementation model that supports processes that communicate through buffers or channels can be used.

MCAPI is based on three basic concepts: (1) a node, which is an independent thread of control that executes a program and communicates with other nodes, (2) an endpoint, which is a communication termination point attached to a node; two endpoints can be connected to form a channel, and (3) a domain which consists of a set of MCAPI nodes that are grouped together based on architectural or routing matters. Each node belongs to one single domain. We use the blocking communication in MCAPI. This means that, if a channel is empty/full its consuming/producing node blocks.

We implement each worker task as an MCAPI domain, which will be mapped on a computation tile. Inside a task, a worker may split in threads, each corresponding to a MCAPI node. These threads may execute in parallel on the PEs of the tile; no constraint is imposed on the type of parallel structures used inside a task, e.g., pipeline, data parallel. All the threads of a worker task are scaled together, when the DVFS technique is applied to their underlying tile.

The sink and the source task are mapped on a single MCAPI domain, executing on the host processor. Each of them may have several threads, implemented as nodes. For example, in our case we assign a source and a sink thread to each worker domain such that, even if some of the worker buffers are full/empty, the source/sink may send/collect data from other workers.

III. Control of the clock frequency of the tile

The clock frequency of the different tiles can be controlled in order to reduce energy consumption, however the throughput constraint should be met. Here, four similar tiles are considered, with the throughput equally split over the tiles. An independent, local controller is designed for each tile, following the method below. The system model (subsection III-A), together with the requirements of the closed-loop system, motivate the controller choice (subsection III-B).

A. System modelling

In order to meet the throughput constraint, the sink node should have available from the workers a token of size Y_{th} data units, each Δ_{th} time units. Here we modify only the tile clock frequency f. Therefore, the objective is to act on f so as to ensure the constraint is fulfilled.

The available data $y(k)$ at time k in the buffer is given by:

$$y(k) = y(k-1) + Q^{in}(k) - Q^{out}(k) \tag{1}$$

$Q^{in}(k)$ and $Q^{out}(k)$ are the quantities of data produced by the worker and consumed by the sink, respectively, at time k. Moreover, as we have no direct action on $Q^{out}(k)$, it is considered as a disturbance. Therefore, the z-transform of the system is:

$$\frac{Y(z)}{Q^{in}(z)} = \frac{1}{1-z^{-1}} \tag{2}$$

which represents an integrator. Moreover, the quantity of data that enters the buffer depends on the producer clock frequency, modeled here by a simple linear equation:

$$Q^{in}(z) = b \cdot F(z) \tag{3}$$

with b being an application-dependent parameter. We approximate b with an average value on typical execution traces.

B. Controller design

The system under study is nonlinear, as pointed out in the literature, e.g. [9]. However, a linearized model as presented in Section III-A is used in order to design the controller. Indeed, when the system can be modeled with a linear(ized) approximation, simple controllers can be obtained. Such controllers have minimal implementation overhead, unlike controllers relying on complex nonlinear control theory.

Notice that non-linearities, such as saturation and bounded control values exist in the system. Because buffers have limited size, they may become full or empty, leading, in this latter case to a "skipped output". The frequency is also bounded and has discrete values. The presence of saturation can affect the behavior of the closed-loop system, leading to controller performance loss, oscillations and even instabilities. However, the controller is designed such that the system stays far from saturation.

From a control viewpoint, the closed-loop system has to fulfill several requirements:

- *Settling time*. Because the quantity of data in the buffer should be large enough to guarantee the throughput, we do not need the controller to be highly reactive. This "low" reactive time will have beneficial side effects because the clock frequency will not be subject to fast/strong changes, the opposite leading to extra overhead (in energy and/or time). Hence the settling time, defined as the time required for the available data y to reach the setpoint y_{SP}, does not need to be short;
- *Overshoot*. Oscillations around y_{SP} are not desired, as they could increase the number of frequency switches;
- *Rise time*. Here again, the rise time must not be too short in order to avoid rapid (instantaneous) frequency changes as soon as there is some activity at the buffer's input/output;
- *Steady state error*. We do not have a strong constraint on a reduced steady state error, because, ultimately, the exact level of buffer filling is not important. However, if the steady state error is not brought to zero, the closed-loop system might present inappropriate behavior.

Finally, the controller must ensure the stability of the closed-loop system. Taking into account the system model (2) and the above requirements, the following PI controller is designed:

$$C(z) = \frac{F(z)}{E(z)} = K_p + K_i \frac{z}{z-1} \tag{4}$$

where $F(z)$ is the controlled frequency. The error $E(z)$ is equal to $Y_{SP}(z) - Y(z)$, where $Y(z)$ is the available data and $Y_{SP}(z)$ is the setpoint (in z-notations). K_p is the proportional gain and K_i is the integral gain. The associated recurrence equation is:

$$f(k) = f(k-1) + (K_p + K_i)e(k) - K_p e(k-1) \tag{5}$$

Note that the proposed controller being a PI one, during saturation the integral term may increase faster than expected, and the usage of an anti-windup mechanism is mandatory.

K_p and K_i influence the value of the closed-loop poles, as resulting from Equations (2) and (4). The closed-loop behavior is fixed via the *choice of the closed-loop poles*, leading to the computation of the controller parameters. The characteristic equation of the closed loop transfer function is given by:

$$z^2 + (-2 + K_p b + K_i b)z + (1 - K_p b) = 0 \tag{6}$$

where the z_1 and z_2 are the poles to be placed:

$$(z - z_1)(z - z_2) = 0 \tag{7}$$

By using (6) and (7), K_p and K_i can be found:

$$-z_1 - z_2 = -2 + K_p b + K_i b \tag{8}$$

$$z_1 z_2 = 1 - K_p b \tag{9}$$

The closed-loop poles must be in the unit circle to ensure the controlled system stability. Moreover, they must have real values to avoid overshoot. Section IV investigates the performance of the closed-loop system for various pole placements within the $(0; 1)$ interval. In addition, the controller is triggered periodically. Its activation period Δ_{ac} should be smaller than the time period at which the throughput is enforced, $\Delta_{ac} < \Delta_{th}$. This is quite similar to the Shannon condition that has to be fulfilled for sampled systems.

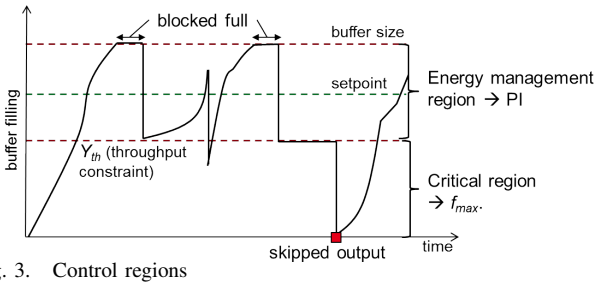


Fig. 3. Control regions

The PI control will be applied in the so-called "energy management region", as visible in Figure 3. When the buffer filling level becomes lower than the quantity of data that gives the throughput constraint, Y_{th}, the maximum clock frequency is applied to avoid a "skipped output". We consider the case where Y_{th} is a constant value, e.g., decoding applications that need to deliver a constant throughput to a rendering system.

C. Limitations

The present work is limited to the following cases: (1) Data-parallel application, with simple inter-task data-dependencies, e.g., no pipeline support; (2) Local, per-tile, control. Currently, the same controller is implemented for each tile, and no coordination between controllers is present to explicitly address inter-tile variations in workload; (3) The system modeling makes several simplifying assumptions. Equation (3) represents a crude approximation for applications behaviour. Furthermore, the work does not straightforwardly apply to cases where neither the output nor the input throughput are constant. Future work will relax these hypotheses by addressing more complex application dependencies, distributed control, and a wider system model.

IV. Experimental results

To validate our PI controller, we used HMAX, a complex recognition application. This application is a powerful hierarchical framework for object recognition which attempts to mimic the behavior of the human brain [17]. Hierarchical recognition typically involves the computation of a set of features at a given level, based on the features computed at the previous level. The input data of the first level is typically a raw image buffer, and the extracted features are the results of multiple filters applied to the input image such as Gabor filters [19].

For the scope of this paper, we consider the first level of the HMAX application, hence the sink node of this HMAX level needs to deliver to a next level a given throughput, expressed in blocks of pixels per second. The application is data parallel.

We investigate the HMAX application for different QoS demand level and buffer size, because these two parameters impact energy management. The QoS demand level indicates the amount of slack available in the application. When the QoS demand is high, the amount of slack available in the application is low, and the other way around. The size of the buffers influences the time window available to utilize the slack. Intuitively, if the buffers are small they will fill up quicker and even if the application has a lot of slack, the workers may block, hence energy would be waisted.

The main metrics considered for evaluation are: (1) energy reductions (here we chose a baseline with no energy management), (2) the number of clock frequency switches, or shortly, number of switches, and (3) the number of skipped outputs. We also present more detailed metrics, e.g., the time in which the workers are blocked on a full buffer, when necessary for the explanation of results. The observed controllers' overhead is low in all cases (under 1%), hence we do not detail it further.

This application executes on a test chip that follows the template described in Figure 1 and embeds four tiles, each with 16 processors [2]. We experimented with several input images over tens of seconds of application execution time. Because we utilize a fully-fledged multi-cluster test chip, the performance results, e.g., throughout, number of skipped output, are very realistic. As the test chip does not support dynamic supply voltage scaling, we obtain the energy numbers by utilising the off-line model presented in what follows.

A. Energy model

This work considers only dynamic power because the measured static leakage energy is very small ($< 3\%$) on our test chip. The dynamic energy consumption model is given by the classical equation: $E_{dyn} = A \cdot V^2$, where A is a coefficient that represent the circuit activity, and V is the supply voltage. We obtain the parameters of the energy model for the test chip by direct measurements. Static voltage scaling and dynamic frequency scaling are performed to obtained several (V, f) points. We assume a linear dependency between the clock frequency and the supply voltage, and from the measurement results we interpolate a model of the form:

$$E_{dyn} = \sum_{f \in f_{levels}} N_f \cdot \left(\alpha f^2 + \beta f\right). \tag{10}$$

where f_{levels} is the set of available frequencies and N_f are the number of clock cycles executed at frequency f. Here we consider 6 frequency levels between 200 and 400MHz, however any other configuration can be used.

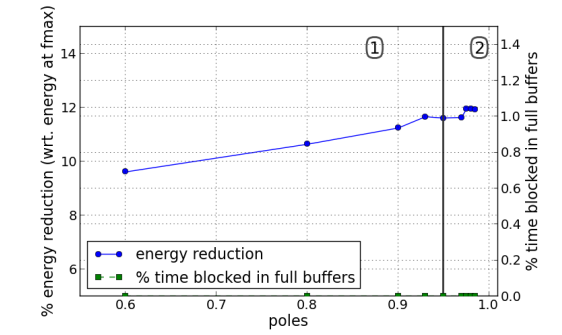
B. PI poles placement

This section presents the investigation of the controller behaviour function of the poles placement on the real axis, for all cases described in Table I.

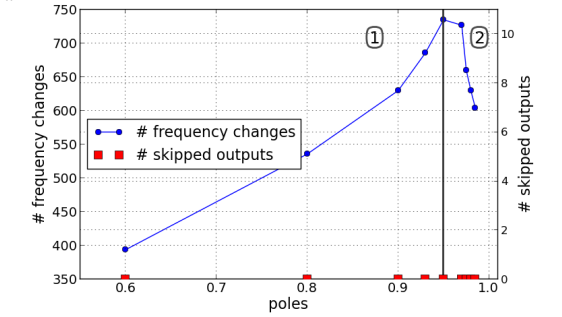
As we will see, the controller has three types of poles intervals, where its behaviour, e.g., number of frequency switches, energy reduction, is different. The first is denoted as the *high-amplitude variation* interval (marked with (1) in figures) and it represents the poles interval where, after largely scaling down the frequency, the controller needs to quickly go

TABLE I. INVESTIGATED COMBINATIONS OF DEMANDED QOS AND BUFFER SIZES

	High QoS demand (QoS_{max})	Low Qos demand ($0.5 \cdot QoS_{max}$)
Large buffer size	Low amount of slack Large slack utilization window	High amount of slack Large slack utilization window
Small buffer size	Low amount of slack Small slack utilization window	High amount of slack Small slack utilization window



(a) Energy reduction and percentage of time when the workers a



(b) Number of frequency switches

Fig. 4. Influence of poles placement for large buffer sizes with high QoS

back to higher frequencies for longer time. This case occurs for reactive controllers, when QoS demand is high, hence the application does not have a large amount of slack. The second is denoted as the *low-amplitude variation* interval (marked with (2) in figures) and it represents the poles interval where the frequency is scaled down less but for longer time periods. The third is the *saturation* interval (marked with (3) in figures), where the workers are blocked because they are either too fast and the output buffers are full, or too slow and the output buffers are empty (skipped outputs). Typically in the saturation zone energy is wasted.

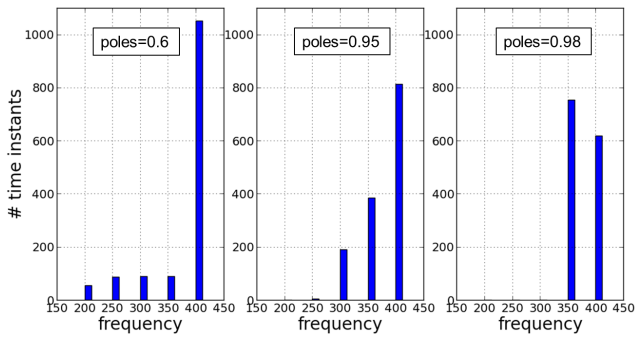
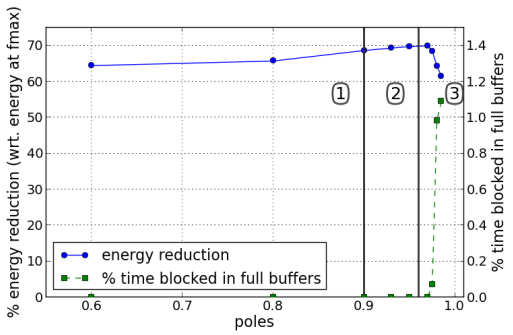
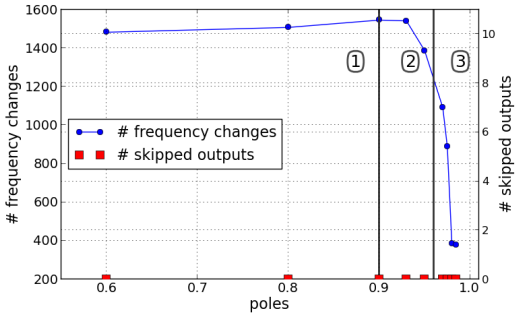


Fig. 5. Histogram of clock frequencies, large buffer sizes, high QoS

1. Large buffer sizes



(a) Energy reduction and percentage of time when the workers a



(b) Number of frequency switches

Fig. 6. Influence of poles placement for large buffer sizes with low QoS

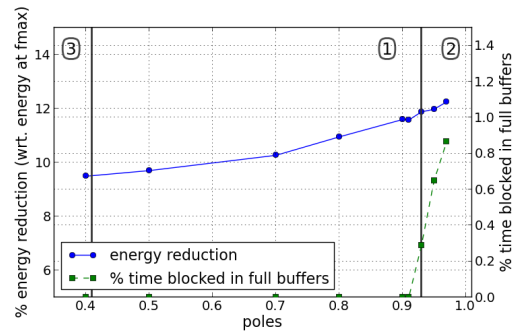
We present the results for poles' belonging to the interval [0.6; 0.985]. Outside of this interval the benefits in energy reduction or number of frequency switches diminish.

First, we would like to highlight that no output is skipped, for all QoS levels, regardless of the poles' location.

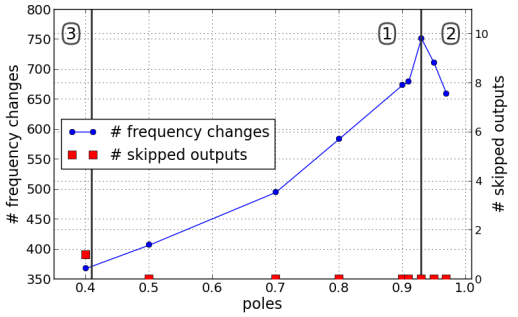
Second, the results for *high QoS demand* are presented in Figure 4a and Figure 4b, for the energy reductions and number of frequency switches, respectively. We observe that, when we move from a more reactive control (poles at 0.6) to a less reactive one (poles at 0.985), the energy reduction slightly increases, hence, from this point of view, the controller performs better when the closed-loop system has slower dynamics. This is because, when the controller is less reactive, the slack accumulates, permitting to run longer times at lower frequencies, which is beneficial for energy consumption. Figure 4a also indicates that the buffers never fill up, hence the workers never block, even when the controller is very slow. This is because the QoS demand is maximum.

Furthermore we observe an increase in number of switches, up an inflexion point at 0.95. This is somewhat counter-intuitive because one would expect that a less reactive controller would induce fewer frequency switches. However, this is not the case here because the QoS demand is high, hence we do not have a large amount of slack, and thus after scaling down the frequency, the controller needs to quickly go back to higher frequencies where is stays for longer time. In other words we are in the high-amplitude variation interval. This is also indicated by the frequency histogram in Figure 5, which presents the number of time instants when the application runs at given frequencies.

For larger poles (after 0.95), the controlled system becomes slow enough to so that the application accumulates slack and it can run longer times at a lower frequency. This results in

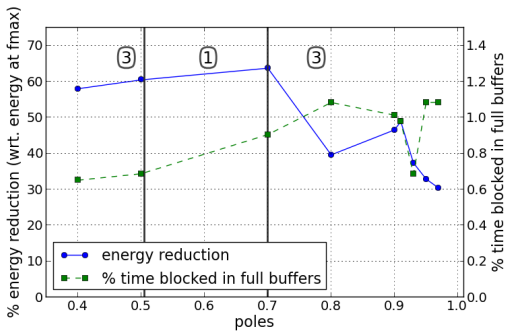


(a) Energy reduction and percentage of time when the workers a

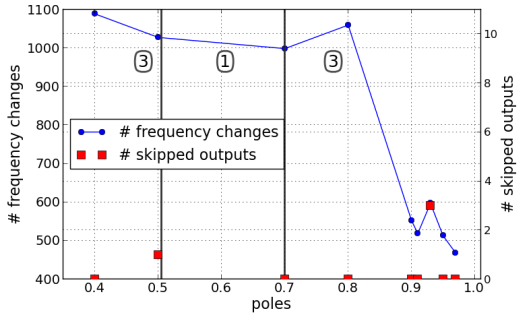


(b) Number of frequency switches, number of skipped outputs

Fig. 7. Influence of poles placement for small buffer sizes with high QoS



(a) Energy reduction and percentage of time when the workers a



(b) Number of frequency switches, number of skipped outputs

Fig. 8. Influence of poles placement for small buffer sizes with low QoS

a decreasing number of switches and the controller is in the low-amplitude variation interval, using mostly the two highest levels of frequency (350MHz and 400Mhz). This is indicated also by Figure 5: with large poles the clock frequency is less reduced in amplitude (but for longer time periods) and the application spends less time at f_{max}.

The "best" controller corresponds to large poles (at the right end of the low-amplitude interval). However, highly reactive controllers can be chosen from the beginning of the high-amplitude interval, when the desired number of frequency switches must be low, as this zone presents only a 2% lower energy reduction when compared to the low-variation one.

Third, the results for *low QoS demand* are presented in Figure 6a and Figure 6b, for the energy reductions and number of frequency switches, respectively. Here also the energy reductions slowly increase over the investigated range of poles. We observe a similar behaviour as in the previous case of high QoS demand, with two intervals, namely high- and low-amplitude. However the differences in number of switches in the first interval is not that pronounced because the QoS demand is lower and, hence there is less need to spend a lot of time at higher frequencies. This is why the number of switches has an inflection point earlier, at 0.93. After 0.93 the system enters in the low-amplitude interval where the slack can accumulate and allow large periods at low frequencies.

Moreover, for poles larger than 0.97, the energy reductions quickly diminish, because the controlled system is too slow and, as the sink node imposes a low QoS demand, the output buffers fill up. In other words, the controller is in the saturation interval.

In the case of low QoS demands, the "best" controller setting corresponds to closed-loop poles just before the saturation zone. Here, unlike the case of high QoS demand, controllers in the high-variation interval are not interesting because they do not bring a significant reduction in number of switches.

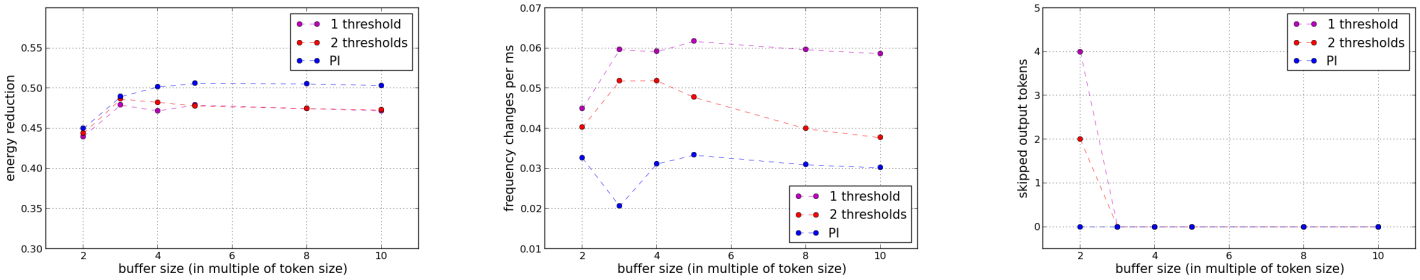
2. *Small buffer sizes*

Figures 7 and 8 present the results for high QoS and low QoS demand, respectively. In case of small buffer sizes we have similar controller behaviour, i.e., high-amplitude, low-amplitude, saturation, but for different poles' intervals. Furthermore, because the buffers are smaller, the controller saturates much earlier. For example, for low QoS demand, for closed-loop poles higher that 0.7, the fact that workers block for more that 1% of the time already causes a decrease in energy reduction, and if the controlled system is not sufficiently reactive, the application will even skip outputs (poles at 0.93). Outputs are skipped in the case the controller is either too reactive or too slow, as visible in Figures 7b and 8b.

C. *Controllers' comparison*

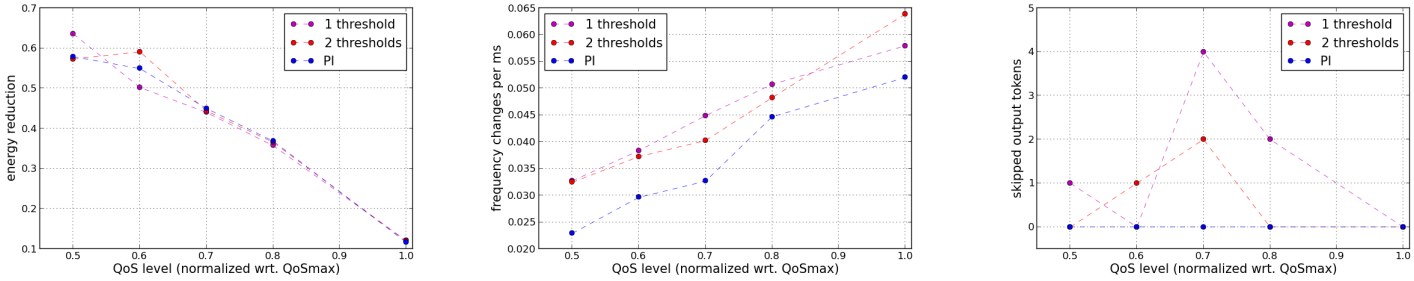
We compare the PI controller proposed in this paper with a state-of-the-art non-linear controller proposed in [11], which we denote as the 1-threshold controller. This controller increases or decreases the frequency with one step, if the workers' buffer filling is smaller or larger, respectively, than the setpoint. The controller is triggered at a variable time interval, when the difference in buffer filling is higher than a certain level, to reduce the number of frequency switches.

Furthermore, we also compare against an extension that we make to the 1-threshold controller, and that we refer to as the 2-thresholds controller. The 2-threshold controller is quite similar to the 1-threshold one, except that it has a low threshold (to trigger frequency switches when the buffer filling increases) and a high threshold (to trigger frequency switches when the



(a) Energy reduction (b) Number of frequency switches (c) Number of skipped output tokens

Fig. 9. Comparison of 3 controllers for medium QoS level ($0.7 \cdot QoS_{max}$), various buffer sizes



(a) Energy reduction (b) Number of frequency switches (c) Number of skipped output tokens

Fig. 10. Comparison of 3 controllers for small buffer size ($2 \cdot$ token size), various QoS levels

buffer filling decreases). This type of non-linear controller is typically used for processes that require a low number of actuating actions, e.g., temperature regulation, hence it has the potential to reduce the number of switches also in our case. The setpoint is the same for all controllers. The 2-thresholds controller thresholds are at 10% lower/higher that this setpoint.

The results for the case of medium QoS level ($0.7 \cdot QoS_{max}$) and different buffer sizes is presented in Figure 9. We can observe that for our use-case application, buffer sizes larger than 5 times the token size do not bring large improvements in number of frequency switches or energy consumption. We observe that the 3 controllers have similar energy reductions, up to 50%, as visible in Figure 9a. The PI controller offers less frequency changes, with an average of 35% and 20% decrease when compared to the 1-threshold controller and 2-thresholds controller, respectively, as indicated in Figure 9b. An important point is that the PI controller skips no output tokens, unlike the non-linear controllers in case of small buffer sizes, as covered by Figure 9c. This means that these controllers are too reactive for the highly constraint case of small buffer sizes.

The results for the case of small buffer sizes, with various QoS levels are presented in Figure 10. Again, we observe that the 3 controllers have similar energy reductions, up to 63% for low QoS levels, as visible in Figure 9a. In term of number of frequency changes, the 2-thresholds controller is close to the 1-threshold one. Furthermore, the PI controller offers less frequency changes than the non-linear ones, with an average reduction of 26% over the entire range of QoS levels, as indicated in Figure 9b. The PI controller skips no output tokens, unlike the non-linear controllers, as presented by Figure 9c.

The results for large buffer sizes are similar; they are not reported due to space limitation.

V. Related work

The management of operating points has been addressed in the literature for the purpose of reducing energy consumption or mitigate hot-spots. A recent survey of such methods is presented by Zhuravlev et al. [5].

Control theoretic strategies have the potential to provide good adaptation solutions for dynamic processes, as proved in many engineering domains. Therefore, they were applied to operating point tuning already in early DVFS approaches for single-core systems [20]. One of the first to study control-theoretic methods for DVFS in multi-processor systems are Juang et al.[10]. They propose a proportional-integral derivative (PID) controller, and the ideas originate from earlier work by the same authors on multiple clock domain microprocessors [6]. The PID controls the clock frequency of each processor by looking at the filling of software tasks queues on each core. Local and distributed methods are compared in simulation, proving a lot of energy saving potential, with a minor performance loss.

Another category of solutions looks at hardware queues in the processor [6] or multi-/many-core systems [7], [8]. A custom controller is designed to adapt the frequency depending on the filling of the queues of the network interface of the NoC [7], [8]. This controller is evaluated initially on an FPGA prototype with good results. Furthermore the method is applied on an Intel SSC platform [14] for a data-parallel application following the same model as in our work. As lessons learned, the need for fast voltage switching and for tailored control algorithms is advocated. Recently, the same authors proposed an optimal control approach base on fractal-state equations [9]. The controller assumes a platform where

operating frequency of PEs and routers can be individually actuated. The experimental workload consists of web-server and data-base applications. The solution seems well tailored for dynamic workload, however it involves on-line linear programming solving, which brings a non-negligible overhead.

The closest approaches to our work are the ones addressing data-flow applications [12], [11]. These bodies of work support a more general application model than our paper because they include pipeline tasks. The SHoP architecture successfully applies a distributed PI(D) controller to adapt the cores' clock frequency and balance multimedia workload over an SoC [12]. The results indicate that the systems can adapt to perturbations and reach throughput in most cases. In contrast to this work, we propose a controller that explicitly addresses the throughput constraint by having two regions, namely, a critical one and an energy management one. The PI is applied only in the energy management region, when the buffer filling is above the level that would guarantee the fulfilment of the throughput. Below that level the maximum frequency is applied. Furthermore, we extend existing work with an experimental analysis of the controller setting for various QoS levels and buffer sizes.

Alimonda et al. [11] propose a threshold-based non-linear controller. The controller is trigger with a variable rate to reduce the number of frequency switches. Energy savings are obtained in simulation for a software FM radio use-case. A comparison of this type of non-linear controller with a PI controller is presented by the same authors [13], indicating that the PI leads to a higher number of frequency switches. Contrary to these previous findings, we show that, for our use-case, the PI controller results in significantly less frequency switches and slightly less energy consumed than the threshold-based controller with variable triggering.

In summary, unlike most of the state-of-the-art, this paper addresses applications where no performance loss is desired, i.e., throughput constraint should be firmly respected. We extend state-of-the-art with an investigation the PI configurations that offer the "best" performance, in terms of energy, number of frequency switches, number of skipped outputs, for various demanded QoS levels and buffer sizes levels. Last but not least, we apply and investigate the control on an real GALS test chip, in contrast to most of the existing approaches.

VI. Conclusion

This paper addresses the problem of determining the clock frequency of tiles in GALS architectures that execute data-parallel applications with throughput constraints. The solution is a novel PI-based controller with two distinct regions, namely, an energy management one where the feedback control is applied and a critical one where the application runs at maximum frequency. These regions are designed to avoid throughput degradations. We evaluated our controller on a test-chip embedding dynamic frequency scaling and static voltage scaling per tile. All timing-related numbers, i.e., throughput, number of frequency switched, are realistic and include overheads. Energy reductions are estimated off-line using models constructed from physical measurements. We have observed that the best PI controller configuration depends on the application's QoS levels and buffer sizes. Furthermore, we compared the PI with two simple non-linear approaches, i.e., (1) a state-of-the-art controller with one threshold and (2) an improved version of it with two thresholds. All three controllers have low overhead. Experimental results suggest that the PI controller outperforms the state-of-the-art in number of frequency switches (25% less, on average) and energy savings. Furthermore the PI meets the throughput constraint in cases where the other approaches fail.

Acknowledgment

This work was partially supported by the Artemis JU, the Catrene JU, and the French Ministere de l'Economie, du Redressement productif et du Numrique, under Grant Agreement n°332913 (project COPCAMS) and Grant Agreement n°CA112 (project HARP).

References

[1] I. M. Panades and et al., "A fine-grain variation-aware dynamic Vdd-hopping AVFS architecture on a 32 nm GALS MPSoC," *J. Solid-State Circuits*, vol. 49, no. 7, pp. 1475–1486, 2014.

[2] D. Melpignano *et al.*, "Platform 2012, a many-core computing accelerator for embedded socs: performance evaluation of visual analytics applications," in *DAC*, 2012, pp. 1137–1142.

[3] A. Varghese *et al.*, "Programming the adapteva epiphany 64-core network-on-chip coprocessor," *IPDPS*, pp. 984–992, 2014.

[4] F. Conti *et al.*, "Energy-efficient vision on the pulp platform for ultra-low power parallel computing," *IEEE SiPS: Design and Implem.*, 2014.

[5] S. Zhuravlev *et al.*, "Survey of energy-cognizant scheduling techniques." *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1447–1464, 2013.

[6] Q. Wu *et al.*, "Formal online methods for voltage/frequency control in multiple clock domain microprocessors," *ASPLOS*, vol. 32, no. 5, pp. 248–259, Oct. 2004.

[7] Ü. Y. Ogras *et al.*, "Design and management of voltage-frequency island partitioned networks-on-chip," *IEEE Trans. VLSI*, vol. 17, 2009.

[8] S. Garg, D. Marculescu, and R. Marculescu, "Custom feedback control: Enabling truly scalable on-chip power management for MPSoCs," in *ISLPED*, 2010, pp. 425–430.

[9] P. Bogdan, R. Marculescu, and S. Jain, "Dynamic power management for multidomain system-on-chip platforms: An optimal control approach," *ACM TODAES*, vol. 18, no. 4, Oct. 2013.

[10] P. Juang *et al.*, "Coordinated, distributed, formal energy management of chip multiprocessors," in *ISLPED*, 2005.

[11] A. Alimonda *et al.*, "A feedback-based approach to dvfs in data-flow applications." *IEEE Trans. on CAD of IC and Syst.*, vol. 28, 2009.

[12] G. Almeida *et al.*, "PI and PID regulation approaches for performance-constrained adaptive multiprocessor system-on-chip," *Embedded Systems Letters, IEEE*, vol. 3, no. 3, pp. 77–80, Sept 2011.

[13] S. Carta *et al.*, "A control theoretic approach to energy-efficient pipelined computation in mpsocs," *ACM TODAES*, vol. 6, 2007.

[14] R. David, P. Bogdan, and R. Marculescu, "Dynamic power management for multicores: Case study using the intel SCC." in *VLSI-SoC*, 2012.

[15] R. Z. Ayoub *et al.*, "OS-level power minimization under tight performance constraints in general purpose systems." in *ISLPED*, 2011.

[16] M. Altieri *et al.*, "Coupled voltage and frequency control for dvfs management," in *PATMOS*, 2013.

[17] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *CVPR'05*, 2005, pp. 994–1000.

[18] www.multicore-association.org.

[19] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex." *Nature Neuroscience*, vol. 3, pp. 1199–1204, 1999.

[20] Z. Lu *et al.*, "Control-theoretic dynamic frequency and voltage scaling for multimedia workloads," in *CASES*, 2002, pp. 156–163.