Green Computing: Power Optimisation of VFI-based Real-time Multiprocessor Dataflow Applications

Waheed Ahmad*, Philip K.F. Hölzenspies†, Mariëlle Stoelinga*, Jaco van de Pol*
*University of Twente, Netherlands
Email: {w.ahmad, m.i.a.stoelinga, j.c.vandepol}@utwente.nl
†Facebook Inc. Email: drphil@fb.com

***Abstract*—Execution time is no longer the only performance metric for computer systems. In fact, a trend is emerging to trade raw performance for energy savings. Techniques like Dynamic Power Management (DPM, switching to low power state) and Dynamic Voltage and Frequency Scaling (DVFS, throttling processor frequency) help modern systems to reduce their power consumption while adhering to performance requirements. To balance flexibility and design complexity, the concept of Voltage and Frequency Islands (VFIs) was recently introduced for power optimisation. It achieves fine-grained system-level power management, by operating all processors in the same VFI at a common frequency/voltage.**

This paper presents a novel approach to compute a power management strategy combining DPM and DVFS. In our approach, applications (modelled in full synchronous dataflow, SDF) are mapped on heterogeneous multiprocessor platforms (partitioned in voltage and frequency islands). We compute an energy-optimal schedule, meeting minimal throughput requirements. We demonstrate that the combination of DPM and DVFS provides an energy reduction beyond considering DVFS or DMP separately. Moreover, we show that by clustering processors in VFIs, DPM can be combined with any granularity of DVFS. Our approach uses model checking, by encoding the optimisation problem as a query over priced timed automata. The model-checker UPPAAL Cora extracts a cost minimal trace, representing a power minimal schedule. We illustrate our approach with several case studies on commercially available hardware.

I. INTRODUCTION

The power consumption of computing systems has increased exponentially [11]. Techniques like Dynamic Power Management (switching to low power state) (DPM) [5] and Dynamic Voltage and Frequency Scaling (throttling processor frequency) (DVFS) [19] help modern systems to reduce their power consumption while adhering to the performance requirements. However, in case of DVFS, distributing a single global frequency to all processors does not prove to be energy efficient. Similarly, assigning single local frequency per processor adds up to the design complexity.

To balance energy optimisation and design complexity, the concept of voltage-frequency islands (VFIs) [10] was introduced to achieve fine-grain system-level power management. A VFI consist of a group of processors clustered together, and each VFI runs on a common clock frequency/voltage. The clock frequencies/voltage supplies of a VFI may differ from other VFIs. Furthermore, different VFI partitions represent DVFS *policies* of different granularity.

While DPM and DVFS are popular power minimisation techniques, the earlier state-of-the-art work [13][15][20] focusses on DVFS only, neglecting static power completely. On the contrary, modern processors have significant static power, and must be addressed. Hence, optimal power minimisation cannot be guaranteed without considering both DVFS and DPM. More advanced approaches that combine DPM and DVFS are presented in [6][18]. Unlike our method, these approaches discuss a specific power reduction policy where DPM and DVFS can be applied to each processor independently only. In contrast, we consider VFI-based hardware platforms where DPM and DVFS can be applied with any DVFS policy.

This paper is the first to compute *energy schedules* for combined DPM and DVFS, on a heterogeneous multiprocessor platform, partitioned in voltage-frequency islands (VFIs). With the help of VFIs, we combine DPM with a DVFS policy with *any* granularity, generalising local and global DVFS. This achieves fine-grained system-level power management.

We use Synchronous Dataflow (SDF) [12] as a computational model. SDF provides a natural representation of real-time streaming and DSP applications. Contemporary SDF analysis tools, e.g., SDF3 [16] lack support for energy optimisation, while power reduction techniques based on mathematical optimisation [6][13][18] do not support model-checking of user-defined properties. Therefore, we propose a novel approach based on Priced Timed Automata (PTA). These extend timed automata [3] (for the modelling of time-critical systems and time constraints) with costs, which we use to model energy consumption. PTA can be analysed by the tool UPPAAL Cora [4] (Cost Optimal Reachability Analysis). PTA and UPPAAL's model checker also extend the analysable properties to include for instance the absence of deadlocks and unboundedness, safety, liveness and reachability. Finally, PTA provide straightforward compositional and extensible modelling capabilities to system engineers, as opposed to mathematical optimisation approaches.

Our approach takes three inputs: an SDF graph that models the application; a platform model that describes the specifics of the hardware such as VFI partitions, frequency levels and power usage per processor; and a throughput constraint. We compute an energy optimal schedule that meets the constraint, utilising the dynamic and static slack in the application.

The main contribution of this paper is a fully automated technique to compute power-optimal schedules. In particular, we demonstrate the following: (1) We apply a combination of DPM and DVFS, confirming earlier results [7][8] that DPM and DVFS together result in lower energy consumption than considering only DVFS; (2) Our method considers processors partitioned into VFIs; thus allowing to combine DPM, and DVFS policy with any granularity. (3) We consider the transition overhead between different frequencies. (4) Our approach

978-1-4673-8035-5/15 $31.00 © 2015 IEEE
DOI 10.1109/DSD.2015.59

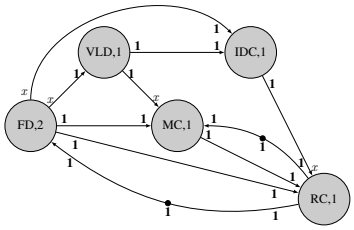

Fig. 1: MPEG-4 Decoder

is able to handle heterogeneous platforms, in which only specific processors can run a particular task. Moreover, our technique is based on the solid semantic framework of Priced Timed Automata, enabling the verification of functional system correctness.

II. Problem Formalisation

A. SDF Graphs

Typically, real-time streaming applications execute a set of periodic tasks, which consume and produce a fixed amount of data. Such applications are naturally modelled as SDF graphs: a directed, connected graph in which tasks are represented by *actors*. Actors communicate with each other via streams of data elements, represented by $tokens$.

Definition 1. An *SDF graph* is a tuple $G = (A, D, \mathsf{Tok}_0, \tau)$ where: A is a finite set of actors, $D \subseteq A^2 \times \mathbb{N}^2$ is a finite set of dependency edges, $\mathsf{Tok}_0 : D \to \mathbb{N}$ denotes distribution of initial tokens in each edge, and the execution time of each actor is given by $\tau : A \to \mathbb{N}_{\geq 1}$.

An actor $a \in A$ can fire if each input edge $(a', a, p, q) \in In(a)$ of a contains at least q tokens; firing actor a removes q tokens from the input edge (a', a, p, q). Firing lasts for $\tau(a)$ time units and ends by producing p' tokens on each output edges $(a, b, p', q') \in Out(a)$. Each actor $a \in A$ fires according to the $repetition\ vector$ $\gamma(a)$ in an SDF graph $iteration$ [9].

Example 1. Figure 1 shows the SDF graph of an MPEG-4 decoder [17]. The SDF graph contains five actors A={FD, VLD, IDC,RC, MC}, representing the tasks performed in MPEG-4 decoding. For example, the frame detector (FD) determines the number of macro blocks to decode. To decode a single frame, FD must process between 0 and 99 macroblocks, i.e., $x \in \{0, 1, \ldots, 99\}$ in Figure 1.

B. Platform Application Model

The Platform Application Model (PAM) models the multi-processor platform where the application, modelled as SDF graph, is mapped on. Our PAM models supports several features, including (1) heterogeneity, i.e., actors can run on certain type of processors only, (2) a partitioning of the processors in voltage and frequency islands, (3) different frequency levels each processor can run on (4) power consumed by a processor in a certain frequency, both when in use and when idle, (5) power-overhead required to switch between frequency levels.

Definition 2. A *platform application model* (PAM) is a tuple $\mathcal{P} = (\Pi, \zeta, F, P_{idle}, P_{occ}, P_{tr}, \tau_{act})$ consisting of a finite set of processors Π assuming that $\Pi = \{\pi_1, \ldots, \pi_n\}$ is partitioned into disjoint blocks of voltage/frequency islands (VFIs) such that $\bigcup \Pi_i = \Pi$, and $\Pi_i \cap \Pi_j = \emptyset$ for $i \neq j$, a function $\zeta : \Pi \to 2^A$ indicating which processors can handle which actors, a finite set of discrete frequency levels available to all processors denoted by $F = \{f_1, \ldots, f_m\}$ such that $f_1 < f_2 < \ldots < f_m$, a function $P_{occ} : \Pi \times F \to \mathbb{N}$ denoting the power consumption (static plus dynamic) of a processor operating at a certain frequency level $f \in F$ in the operating state, a function $P_{idle} : \Pi \times F \to \mathbb{N}$ assigning the power consumption (static) of a processor operating at a certain frequency level $f \in F$ in the idle state, a function $P_{tr} : \Pi \times F^2 \to \mathbb{N}$ expressing the transition overhead from one frequency level $f \in F$ to next frequency level $f \in F$ for each processor $\pi \in \Pi$, and the valuation $\tau_{act} : A \times F \to \mathbb{N}_{\geq 1}$ defining the actual execution time τ_{act} of each actor $a \in A$ mapped on a processor at a certain frequency level $f \in F$.

Level	Voltage	Frequency	Level	Voltage	Frequency
1	1.2	1400	4	1.05	1128.7
2	1.15	1312.2	5	1.00	1032.7
3	1.10	1221.8			

TABLE I: DVFS levels of Samsung Exynos 4210

Example 2. Exynos 4210 is a state-of-the-art processor used in high-end platforms such as Samsung Galaxy Note, SII etc. Table I shows its different DVFS levels, and corresponding CPU voltage (V) and clock frequency (MHz) [14].

Definition 3. The *throughput* of an SDF graph mapped on a PAM is the average number of iterations that are executed per unit time, measured over a sufficiently long period.

III. Power Optimisation Techniques

This section illustrates the importance of considering DPM with DVFS, and VFIs with the help of a non-trivial observation. Let us consider a real-time periodic application mapped on a single processor. Figure 2 shows the behaviour of static (E_S) and dynamic (E_D) energy consumption of the processor as a function of processor frequency for the execution of an entire iteration. Note that E_S also includes power-overheads. The minimum frequency at which the task can meet its deadline is denoted by f_a. Similarly, f^* denotes the minimum frequency at which there is enough slack for the processor to move to the low power state. Thus, the processor can only move to the low power state, if its frequency is no less than f^*. Otherwise, it will not be able to meet the deadline.

As explained earlier, E_D increases cubically with the increase of frequency. However, E_S shows varying patterns. In Region A where $f_a \leq f < f^*$, idle period of the processor is too short to allow it to move to the low power state where static energy consumption is lower. Therefore, E_S is higher and constant in Region A. Nevertheless, as frequency reaches f^*, slack, i.e., idle period of the processor increases, allowing the transition to less static power consuming states. Thus, E_S drops down at $f = f^*$. As frequency increases beyond f^* in Region B, idle period of the processor increases further in linear fashion, leading to switching to deeper sleep states by the processor. Without loss of generality, if we assume that power-overhead of switching to deeper low power states

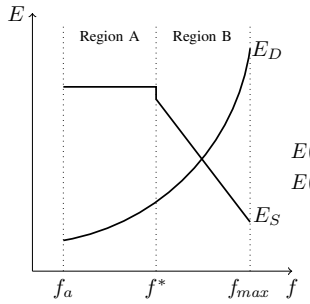

Fig. 2: E_S and E_D

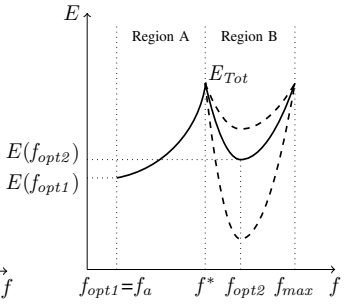

Fig. 3: E_{Tot}

also increases linearly, we get linear decrease of E_S with the increase of frequency in Region B, as shown in Figure 2.

Figure 3 shows $E_{Tot} = E_S + E_D$, as a function of processor frequency. We can see in Figure 3 that local optimal frequencies to minimise E_{Tot} in both regions are well defined. However, *there is no a priori reason that global minimum of E_{Tot} should lie in Region A or Region B*. Depending on the power consumption values of the processor and deadline of the application, the global optimal frequency can be either in Region A or B.

Alternatively, if we do not consider DPM, E_S in Region B remains same as A, and consequently, E_{Tot} increases in Region B as well. Therefore, we can safely conclude that we must consider *both* DPM and DVFS to determine optimal power consumption. We can generalise this result for multi-processors also. Moreover, partitioning processors into VFIs enable us to assign frequency per group of processors, rather than running all processors at the same frequency. The different power minimisation techniques discussed in this section are evaluated experimentally in Section V.

IV. Power Optimisation using Priced Timed Automata

A. From SDF Graphs to Priced Timed Automata

Timed automata (TA) are a popular and powerful formalism to model and analyse real-time systems [3]. TA are state-transition diagrams augmented with real-valued clocks, which can be used in enabling conditions for transitions and in state invariants that enforce deadlines.

Price timed automata (PTA) extend TA with costs. Costs can either be accumulated in states, proportionally to the residence time, or by taking a transition. In particular, the model-checker UPPAAL Cora [4] has a support for finding cost-optimal schedules, i.e., it can provide a trace to a given goal state with the lowest accumulated value costs.

Our framework consists of separate models of an SDF graph and the platform application model. In this way, we split the problem of optimal power management in terms of tasks and resources. Given an an SDF graph $G = (A, D, \mathsf{Tok}_0, \tau)$ mapped on a platform application model $(\Pi, \zeta, F, P_{idle}, P_{occ}, P_{tr}, \tau_{act})$, we generate a parallel composition of PTA:

$$A_G \| Processor_1 \| , \ldots , \| Processor_n \| Scheduler.$$

Here, the automaton A_G models the SDF graph. The PTA $Processor_1, \ldots, Processor_n$ model the processors $\Pi = \{\pi_1, \ldots, \pi_n\}$, and the automaton $Scheduler$ decides when to switch the frequency level of all processors in the same VFI. Due to lack of space, we refer to [2] for details on translation of SDF graphs to PTA and analysis in UPPAAL Cora.

Voltage(V)	Frequency(MHz)	GDVFS		GDVFS+DPM	
		P_{idle}(W)	P_{occ}(W)	P_{idle}(W)	P_{occ}(W)
1.2	1400	0.4	4.6	0.1	4.6
1.00	1032.7	0.4	1.8	0.4	1.8

TABLE II: Platform power consumption of different scenarios

B. Power Optimisation utilising UPPAAL Cora

This subsection illustrates how we use UPPAAL Cora to obtain power optimal schedules. As explained earlier, each actor fires according the repetition vector γ in an iteration. For each actor $a \in A$ in the SDF graph, we define its corresponding entry in the repetition vector as $\gamma(a)$. We also define the number of iterations per period as m.

A technique of calculating the maximum throughput of an SDF graph mapped on a given number of processors via timed automata (TA), using the model-checker UPPAAL is proposed in [1]. This work demonstrates that the *fastest execution* of every consistent and strongly connected SDF graph, mapped on a platform application model, repeats the periodic phase n times if each actor $a \in A$ fires equal to $(nm + k)\gamma(a)$ for some constants n and k. The maximal throughput of the SDF graph is determined from the periodic phase.

Power optimisation using UPPAAL Cora is a two-fold method. In the first step, we obtain the *completion time* of the *fastest execution* of an SDF graph using the method explained above. The second step is to incorporate *completion time*, in our PTA model discussed in IV-A. Then, using the `Best trace` option in UPPAAL Cora, we derive an execution which minimises the power consumption, while maintaining the *completion time*.

V. Experimental Evaluation via MPEG-4

We analyse power optimisation, by means of an MPEG-4 decoder example in Figure 1. We evaluate energy consumption with respect to (1) fixed number of processors (2) varying number of processors.

A. Fixed Number of Processors

We consider an MPEG-4 decoder mapped on the platform containing four Exynos 4210 processors, i.e., $\Pi = \{\pi_1, \pi_2, \pi_3, \pi_4\}$. For a constraint of completing three iterations within varying deadlines, we consider the following scenarios.

- Global DVFS only (`GDVFS`): In the first scenario, the processors employ DVFS only, without DPM. We consider two frequency levels (MHz), i.e., $f_1 = 1032.7$ and $f_2 = 1400$. Table II shows the power consumption (W) at both frequencies. Note that the idle power consumption of all processors $\pi \in \Pi$ is constant at both frequencies. Recall that `GDVFS` assumes one VFI $\Pi_1 = \{\pi_1, \pi_2, \pi_3, \pi_4\}$.
- Global DVFS + DPM (`GDVFS + DPM`): To allow the processors to benefit from both DPM and DVFS, we introduce a low power state, i.e., idle power consumption of all processors $\pi \in \Pi$ at frequency level $f_2 = 1400$

MHz is changed to $P_{idle}(\pi, f_2)$=0.1 W because more idle time allows DPM. However, $P_{occ}(\pi, f_2)$ and $P_{occ}(\pi, f_1)$ remains same as GDVFS, as given in Table II. The power-overhead (W) of all processors $\pi \in \Pi$ is, $P_{tr}(\pi, f_2, f_1) = 0.2$ and $P_{tr}(\pi, f_1, f_2) = 0.1$.

- DVFS + DPM with 2 VFIs (DVFS + DPM − 2): The processors are partitioned into two VFIs such that $\Pi_1 = \{\pi_1, \pi_2\}$ and $\Pi_2 = \{\pi_3, \pi_4\}$, while utilising both DVFS and DPM. The power consumption values remain the same as in GDVFS + DPM.
- DVFS + DPM with 3 VFIs (DVFS + DPM − 3): In this case, the processors are partitioned into three VFIs such that $\Pi_1 = \{\pi_1\}$, $\Pi_2 = \{\pi_2\}$ and $\Pi_3 = \{\pi_3, \pi_4\}$.

Figure 4 shows the energy consumption calculated for each scenario. The first two scenarios are compared as follows.

1) GDVFS vs GDVFS+DPM:

- In almost all cases, GDVFS results in higher energy consumption, as compared to GDVFS + DPM.
- At tighter deadlines when idle time of processors is insufficient to move to the low power state, the difference between GDVFS and GDVFS + DPM is not significant. Thus, E_{Tot} lies in Region A. However, as deadline is relaxed, the processors spend more time in the low power state and E_{Tot} moves to Region B. Consequently, GDVFS + DPM gets more promising, implying the benefits of DPM. For instance, at 50 ms, GDVFS + DPM saves considerable energy consumption equal to 10.3%, compared to GDVFS.

Therefore, the results explained above prove our earlier claim that static power is non-negligible in order to guarantee optimality, and both Region A and B must be analysed to determine minimum energy consumption.

Now we have seen the benefits of DPM, the effect of varying the number of VFI partitions, i.e., GDVFS + DPM, DVFS + DPM − 2 and DVFS + DPM − 3 is explained below.

2) DVFS+DPM with VFIs:

- At tighter deadlines, for the reason that system is at maximum capacity all the time, having higher number of VFIs does not result in major energy reduction.
- But, as deadline is relaxed, we see that increasing the number of VFIs prove to be more effective, and produce considerable reduction in energy consumption. For example, for the deadline of 50 ms, DVFS + DPM − 2 and DVFS + DPM − 3 save 4.9% and 8.3% energy consumption respectively, as compared to GDVFS + DPM. The reason is that in GDVFS + DPM where we have one VFI only, all processors have to run at the same frequency, even though fewer might be required. By partitioning into more VFIs, we can cluster processors in such a way that only required processors run at the specific frequency, and others may run at the different frequency; thus, trading system's complexity for energy minimisation.

Hence, VFIs provide better control over energy optimisation and design complexity. Without VFIs, system designers are left with two options only, i.e., either local or global DVFS. However, with the help of VFIs, it is possible to achieve fine-grain power reduction by employing any DVFS policy, ranging from local to global. Therefore, the use of VFIs enables system designers with the larger range of design choices.

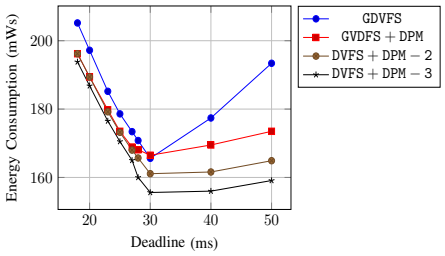

Fig. 4: Comparison of power optimisation techniques

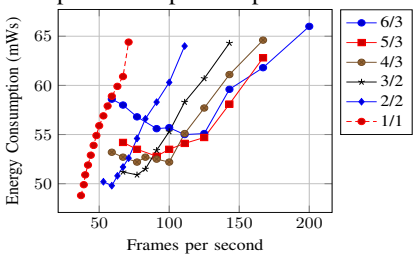

Fig. 5: Energy usage per frame against frames per second. The legend shows the number of processors/ number of VFIs.

B. Varying Number of Processors

We also evaluated the performance of the MPEG-4 decoder on a varying number of processors. The maximum number of processors required to accomodate maximum parallelism of this example is 6, calculated by SDF3. We obtain a Pareto front by sweeping the throughput constraint, as shown in Figure 5. We get three majors results from Figure 5, as explained below.

- Achieving higher frames per second at fewer processors increases the energy consumption. The reason is the smaller slack at the tighter frames per second constraint. Therefore, more work is done on fewer processors to attain the same frames per second.
- As we relax the frames per second constraint, slack increases, and the same frames per second can be achieved by consuming less energy on fewer processors. For instance, in Figure 5, we can reach 100 frames per second on four processors with 2.4% less energy consumption, as compared to five processors. For higher slack in the application, this difference gets bigger. Thus, we may not require more processors in our platform, and reach a certain throughput at a considerably lower energy consumption, contributing to prolonged battery life.
- Relaxing the frames per second beyond a certain limit on a fixed number of processors increases the energy consumption, as static energy surpasses the dynamic energy. For instance, the energy consumption of three processors increases by 1.9%, when moving from 77 to 59 frames per second.

VI. Other Case Studies

Apart from the MPEG-4 decoder example, we present other real-life case studies taken from [1], namely a bipartite graph, an MP3 playback application, a MP3 decoder and an Audio Echo Canceller. We assume that these case studies are mapped on a multiprocessor platform containing Samsung Exynos 4210 processors $\Pi = \{\pi_1, \ldots, \pi_n\}$. The execution times of these case studies are given in ms.

TABLE III: Experimental Results of Case Studies

Processor	VFIs	Time per Iteration	Energy Consumption
Bipartite Graph			
4	$\Pi_1 = \{\pi_1, \pi_2\}, \Pi_2 = \{\pi_3, \pi_4\}$	42	345·3
3	$\Pi_1 = \{\pi_1\}, \Pi_2 = \{\pi_2, \pi_3\}$	44	338·5
2	$\Pi_1 = \{\pi_1\}, \Pi_2 = \{\pi_2\}$	51	333·1
1	$\Pi_1 = \{\pi_1\}$	73	335·8
MP3 Playback Application			
2	$\Pi_1 = \{\pi_1, \pi_2\}$	1880	9907
1	$\Pi_1 = \{\pi_1\}$	2118	9742·8
MP3 Decoder			
2	$\Pi_1 = \{\pi_1, \pi_2\}$	8	64·6
1	$\Pi_1 = \{\pi_1\}$	14	64·4
Audio Echo Canceller			
4	$\Pi_1 = \{\pi_1, \pi_2\}, \Pi_2 = \{\pi_3, \pi_4\}$	23	324·2
3	$\Pi_1 = \{\pi_1\}, \Pi_2 = \{\pi_2, \pi_3\}$	24	322·3
2	$\Pi_1 = \{\pi_1, \pi_2\}$	35	322
1	$\Pi_1 = \{\pi_1\}$	73	335·8

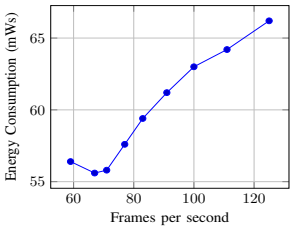

Fig. 6: Power consumption of the heterogeneous system

Table III shows the results of the experiments to find out the least power consumption. The first column displays the given number of processors, and the second column represents division of processors into VFIs. Columns 3-4 show per iteration, minimum achievable time (ms) and minimum energy consumption (mWs) respectively, on the given processors.

So far, we have considered a homogeneous system, in which we can assign any actor to any processor. However, this freedom is limited in a heterogeneous system by which processors could execute a particular actor. In UPPAAL Cora, we can utilise the same models described earlier in a heterogeneous system. Let us consider an MPEG-4 decoder mapped on a heterogeneous system having two Exynos 4210 processors $\Pi' = \{\pi'_1, \pi'_2\}$ and two Exynos 4212 processors $\Pi'' = \{\pi''_1, \pi''_2\}$. Let us consider that actor $\{\text{FD}\} \subseteq A$ can be mapped only on $\{\pi'_1\} \subseteq \Pi'$, actors $\{\text{VLD, IDC}\} \subseteq A$ can be executed only on $\{\pi'_2, \pi''_2\} \subseteq \Pi' \cup \Pi''$, and $\{\pi''_1\} \subseteq \Pi''$ is assigned to execute actors $\{\text{RC, MC}\} \subseteq A$ only. We have two VFIs, i.e., $\Pi_1 = \{\pi'_1, \pi''_1\}$ and $\Pi_2 = \{\pi'_2, \pi''_2\}$. Figure 6 shows the Pareto front of energy dissipation for the varying throughput constraints.

VII. CONCLUSIONS

Despite the remarkable progress in power optimisation using DVFS, compact methods for optimal power management by combining DVFS and DPM, with VFIs are still needed. We demonstrate a novel power reduction technique for SDF-modelled streaming applications, which combines the benefits of DVFS, DPM and VFIs using model-checking.

Future research directions are to carry on from the results achieved in this paper and explore the possibilities of battery-aware scheduling of SDF graphs. Another exciting prospect is to add a third dimension, taking also reliability relaxations and constraints into account.

ACKNOWLEDGEMENT

This research is supported by the EU FP7 project SENSATION (318490).

REFERENCES

[1] W. Ahmad, R. de Groote, P. K. F. Hölzenspies, M. Stoelinga, and J. van de Pol. Resource-constrained optimal scheduling of synchronous dataflow graphs via timed automata. In *ACSD '14*, pages 72–81, 2014.

[2] W. Ahmad, P. K. Hölzenspies, M. Stoelinga, and J. van de Pol. Green computing: Power optimisation of VFI-based real-time multiprocessor dataflow applications (extended version). Technical Report TR-CTIT-15-04, University of Twente, 2015.

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[4] G. Behrmann, K. G. Larsen, and J. I. Rasmussen. Optimal scheduling using priced timed automata. *SIGMETRICS Perform. Eval. Rev.*, 32(4):34–40, Mar. 2005.

[5] L. Benini, A. Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Trans. on VLSI Systems*, 8(3):299–316, June 2000.

[6] G. Chen, K. Huang, and A. Knoll. Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination. *TCES*, 2014.

[7] V. Devadas and H. Aydin. On the interplay of voltage/frequency scaling and device power management for frame-based real-time embedded applications. *IEEE Trans. on Computers*, 61(1):31–44, Jan 2012.

[8] M. E. T. Gerards and J. Kuper. Optimal DPM and DVFS for frame-based real-time systems. *TACO*, 9(4):41:1–41:23, Jan. 2013.

[9] A.-H. Ghamarian, M. C. W. Geilen, S. Stuijk, T. Basten, A. J. M. Moonen, M. Bekooij, B. Theelen, and M. Mousavi. Throughput analysis of synchronous data flow graphs. In *ACSD'06*, pages 25–36, June 2006.

[10] S. Herbert and D. Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *ISLPED'07*, pages 38–43, Aug 2007.

[11] S. Irani and K. R. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, June 2005.

[12] E. A. Lee and D. G. Messerschmitt. Synchronous data flow: Describing signal processing algorithm for parallel computation. In *COMPCON '87*, pages 310–315, 1987.

[13] A. Nelson, O. Moreira, A. Molnos, S. Stuijk, B. Nguyen, and K. Goossens. Power minimisation for real-time dataflow applications. In *DSD'11*, pages 117–124, Aug 2011.

[14] S. Park, J. Park, D. Shin, Y. Wang, Q. Xie, M. Pedram, and N. Chang. Accurate modeling of the delay and energy overhead of dynamic voltage and frequency scaling in modern microprocessors. *TCAD*, May 2013.

[15] A. K. Singh, A. Das, and A. Kumar. Energy optimization by exploiting execution slacks in streaming applications on multiprocessor systems. In *DAC'13*, pages 115:1–115:7. ACM, 2013.

[16] S. Stuijk, M. Geilen, and T. Basten. SDF^3: SDF For Free. In *ACSD'06*, pages 276–278, June 2006.

[17] B. Theelen, M. C. W. Geilen, T. Basten, J. P. M. Voeten, S. V. Gheorghita, and S. Stuijk. A scenario-aware data flow model for combined long-run average and worst-case performance analysis. In *MEMOCODE'06*, pages 185–194, July 2006.

[18] Y. Wang, H. Liu, D. Liu, Z. Qin, Z. Shao, and E. H.-M. Sha. Overhead-aware energy optimization for real-time streaming applications on multiprocessor system-on-chip. *TODAES*, 16(2):14:1–14:32, Apr. 2011.

[19] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced CPU energy. In *OSDI'94*. USENIX Association, 1994.

[20] J. Zhu, I. Sander, and A. Jantsch. Energy efficient streaming applications with guaranteed throughput on MPSoCs. In *EMSOFT'08*, pages 119–128, 2008.