Applying Pay-Burst-Only-Once Principle for Periodic Power Management in Hard Real-Time Pipelined Multiprocessor Systems

GANG CHEN, Technische Universität Muenchen
KAI HUANG, Technische Universität Muenchen and Sun Yat-sen University
CHRISTIAN BUCKL, Fortiss GmbH
ALOIS KNOLL, Technische Universität Muenchen

Pipelined computing is a promising paradigm for embedded system design. Designing a power management policy to reduce the power consumption of a pipelined system with nondeterministic workload is, however, nontrivial. In this article, we study the problem of energy minimization for coarse-grained pipelined systems under hard real-time constraints and propose new approaches based on an inverse use of the pay-burst-only-once principle. We formulate the problem by means of the resource demands of individual pipeline stages and propose two new approaches, a quadratic programming-based approach and fast heuristic, to solve the problem. In the quadratic programming approach, the problem is transformed into a standard quadratic programming with box constraint and then solved by a standard quadratic programming solver. Observing the problem is NP-hard, the fast heuristic is designed to solve the problem more efficiently. Our approach is scalable with respect to the numbers of pipeline stages. Simulation results using real-life applications are presented to demonstrate the effectiveness of our methods.

Categories and Subject Descriptors: C.3 [**Special-Purpose and Application-based System**]—*Real-Time and Embedded Systems*

General Terms: Algorithms

Additional Key Words and Phrases: Scheduling, energy, pay-burst-only-once, periodic power management, real-time system

ACM Reference Format:
Gang Chen, Kai Huang, Christian Buckl, and Alois Knoll. 2015. Applying pay-burst-only-once principle for periodic power management in hard real-time pipelined multiprocessor systems. ACM Trans. Des. Autom. Electron. Syst. 20, 2, Article 26 (February 2015), 27 pages.
DOI: http://dx.doi.org/10.1145/2699865

1. INTRODUCTION

With increasing requirements for high performance, multicore architectures are believed to be the major solution for future embedded systems. Many real-time applications, especially streaming applications, can be executed on multiple processors simultaneously to achieve parallel processing. When real-time applications are executed

A preliminary version of a portion of this article appears in *Proceedings of the Conference on Design Automation and Test in Europe* 2013.
This work has been partly supported by China Scholarship Council, German BMBF project ECU (grant no. 13N11936) and Car2X (grant no. 13N11933).
The authors would like to thank their sponsors for their support.
Authors' addresses: G. Chen, Department of Informatics, Technische Universität Muenchen (TUM), Boltzmannstraße 3, 85748, Garching bei München, Germany; K. Huang (corresponding author), School of Mobile Information Engineering, Sun Yat-sen University, Zhu Hai, China; email: huangk36@mail.sysu.edu.cn; C. Buckl, Fortiss GmbH, Germany; A. Knoll, Department of Informatics, Technische Universität Muenchen (TUM), Boltzmannstraße 3, 85748, Garching bei München, Germany.

DOI: http://dx.doi.org/10.1145/2699865

on multicore architectures powered by batteries, minimizing the energy consumption is one of the major design goals, because an energy-efficient design will increase the lifetime, increase the reliability, and decrease the heat dissipation of the system.

Pipelined computing is a promising paradigm for embedded system design, which can, in principle, provide high throughput and low energy consumption [Carta et al. 2007]. For instance, a streaming application can be split into a sequence of functional blocks that are computed by a pipeline of processors where power-gating techniques can be applied to achieve energy efficiency.

Performance constraints of a streaming application are usually imposed on two principle metrics, that is, throughput and latency. The latency is the main concern for applications such as video/telephone conferencing and automatic pattern recognition applications, where the latency beyond a certain boundary is not tolerated. In the case of pipelined real-time systems, the latency of a streaming application can be expressed as the end-to-end deadline requirement that the application is processed through the pipeline.

Designing the scheduling policy for the pipeline stages under the requirements of both energy efficiency and timing guarantee is, however, nontrivial. In general, energy efficiency and timing guarantee are conflict objectives, that is, techniques that reduce the energy consumption of the system will usually pay the price of longer execution time, and vice versa. Previous work on this topic either requires precise timing information of the system [Yu and Prasanna 2002; Xu et al. 2007] or tackles only soft real-time requirements [Javaid et al. 2011b; Carta et al. 2007]. However, this precise timing of task arrivals might not be guaranteed in practice. Thus, the previous approaches cannot guarantee the worst-case deadline and cannot be applied to those embedded systems where violating deadlines could be disastrous. Compared to the preceding work, our work tackles a pipelined event stream with nondeterministic workloads in hard real-time systems by an inverted use of the pay-burst-only-once principle for energy efficiency.

This article studies the energy minimization problem of coarse-grained pipelined systems under hard real-time requirements. We consider a streaming application that is split into a sequence of coarse-grained functional blocks which are mapped to a pipeline architecture for processing. The workload of the streaming application is abstracted as an event stream and the event arrivals of the stream are modeled as the arrival curves in the interval domain [Le Boudec and Thiran 2001]. The event stream has an end-to-end deadline requirement, that is, the time by which any event in the stream travels through the pipeline should be no longer than this required deadline. The objective is thereby to find those optimal scheduling policies for individual stages of the pipeline with minimal energy consumption while the deadline requirement of the event stream is guaranteed.

Intuitively, the problem can be solved by partitioning the end-to-end deadline into sub-deadlines for individual pipeline stages and optimizing the energy consumption based on the partitioned sub-deadlines. However, any partition strategy based on the end-to-end deadline and the follow-up optimization method will suffer counting multiple times the burst of the event stream, which will inevitably overestimate the needed resource for every pipeline stage and lead to poor energy saving. A motivation example in Section 4 will demonstrate this drawback in detail. Therefore, a more sophisticated method is needed to tackle this problem.

In this article, we develop a new approach to solve the energy minimization problem for pipelined multiprocessor embedded systems while guaranteeing the worst-case end-to-end delay. This article summarizes and extends the results built in Chen et al. [2013]. Our idea to solve this problem lies in an inverse use of the well-known pay-burst-only-once principle [Le Boudec and Thiran 2001]. Rather than directly partitioning the end-to-end deadline, we compute for the entire pipeline one service curve that serves as a constraint for the minimal resource demand. The energy minimization problem is then formulated with respect to the individual resource demands of pipeline stages. To solve this problem, we propose two heuristics, that is, a quadratic programming heuristic and a fast heuristic. In the quadratic programming heuristic, the minimization problem is transformed to a standard quadratic programming problem with box constraint and then solved by a standard solver. Observing that the formulated problem is NP-hard, we present a fast heuristic to find a suboptimal solution by analyzing the properties of the optimal solution, running with the complexity $O(mn)$ (where m and n are the stage number and sample step number, respectively). For simplicity, we consider power-gating energy minimization and use periodic dynamic power management in Huang et al. [2009b, 2011a] to reduce the leakage power, that is, to periodically turn on and off the processors of the pipeline. In this work we compute period power management schemes offline and the fixed T_{on}/T_{off} for processors of every pipeline stage are applied during runtime. With this approach, we can not only guarantee the overall end-to-end deadline requirement but also retrieve the pay-burst-only-once phenomena, achieving a significant reduction of the energy consumption. In addition, our methods are scalable with respect to the number of pipeline stages. The contributions of this article are summarized as follows.

—A new method is developed to solve the energy minimization problem for pipelined multiprocessor embedded systems by inversely using the pay-burst-only-once principle.
—A minimization problem is formulated based on the needed resource of individual stages of the pipeline architecture and a transformation of the formulation to a standard quadratic programming problem with box constraints. The formulated problem is proved NP-hard.
—A quadratic programming heuristic is developed to solve the formulated problem and a formal proof is provided to show the correctness of our approach, that is, guarantee on the end-to-end deadline requirement.
—A fast heuristic is developed to solve the formulated problem, running with the complexity $O(mn)$.

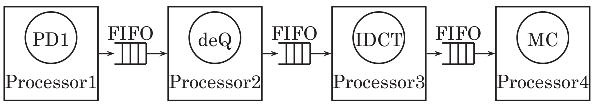
The rest of the article is organized as follows. Section 2 reviews related work in the literature. Section 3 presents basic models and the definition of the studied problem. Section 4 presents the motivation example and Section 5 describes the proposed approach. Experimental evaluation is presented in Section 6, and Section 7 concludes.

2. RELATED WORK

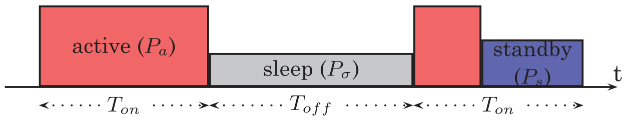
Pipelined computing is a promising paradigm for embedded system design, which can in principle provide high performance and low energy consumption. Pipelined multiprocessor systems are widely applied as a viable platform for high performance implementation of multimedia applications [Shee and Parameswaran 2007; Javaid and Parameswaran 2009; Shee et al. 2006; Karkowski and Corporaal 1997]. Energy optimization for pipelined multiprocessor systems is an interesting topic where a number of techniques have been proposed in the literature. Carta et al. [2007] and Alimonda et al. [2009] proposed a feedback control technique for dynamic voltage/frequency scaling (DVFS) in a pipelined MPSoC architecture with soft real-time constraints, aimed at minimizing energy consumption with throughput guarantees. Each pipelined processor is associated with a dedicated controller that monitors the occupancy level of the queues to determine when to increase or decrease the voltage-frequency levels of the processor. Javaid et al. [2011b] proposed an adaptive pipelined MPSoC architecture and a runtime balancing approach based on workload prediction to achieve energy efficiency. The authors in Javaid et al. [2011a] proposed a dynamic power management scheme for adaptive pipelined MPSoCs. In this work, the duration of idle periods is determined based on future workload prediction and used to select an appropriate power state for the idle processor. However, the prior approaches are under soft real-time constraints. Regarding hard real-time systems, these approaches cannot be applied.

There are also methods [Davare et al. 2007; Hong et al. 2011; de Langen and Juurlink 2006, 2009; Liu et al. 2014; Yu and Prasanna 2002] for hard real-time systems. To guarantee the end-to-end delay, the anthers in Liu et al. [2014] studied the problem of minimizing the number of processors required for scheduling end-to-end deadline-constrained streaming applications modeled as CSDF graphs, where the actors of a CSDF are executed as strictly periodic tasks. In Davare et al. [2007], the authors optimized periods for dependent tasks on hard real-time distributed automotive systems in order to meet the end-to-end constraints. In Hong et al. [2011], the authors proposed a distributed approach to assign local deadlines for periodical tasks on distributed systems to meet the end-to-end deadline constraints. To reduce the energy consumption, Yu and Prasanna [2002] presented an integer linear programming (ILP) formulation for the problem of frequency assignment of a set of periodic independent tasks on a heterogeneous multiprocessor system. The authors in de Langen and Juurlink [2006, 2009] proposed leakage-aware scheduling heuristics to reduce the energy consumption by translating real-time applications with periodic tasks to DAGs using the frame-based scheduling paradigm and considering the trade-offs among DVFS, DPM, and the number of the processors. But these methods require precise timing information such as periodical real-time events. However, in practice, this precise timing information of task arrivals might not be determined in advance. The nondeterminism in the timing of event arrivals results from two main causes: (a) An event may be triggered by the physical environment, which, in general, is not able to be accurately predicted. (b) When a distributed system is considered, an event might be triggered by other events on different processing components in which variable execution workloads would make the prediction of precise information on event arrivals extremely complicated. In the aforesaid research, there is no guarantee that an event will arrive in time. Therefore, these approaches cannot be applied to guarantee the worst-case deadline in embedded systems where violating deadlines could be disastrous. Unlike previous work, we focus on improving energy efficiency in hard real-time embedded systems while guaranteeing the system satisfies the worst-case deadline constraint.

To model irregular event arrivals, Real-Time Caculus (RTC) [Thiele et al. 2000], which is based on network calculus [Le Boudec and Thiran 2001], can be applied. Specifically, the arrival curve in the RTC models an upper bound and a lower bound of the number of event arrivals or the demand of computation under a specified time interval domain. Considering the DVFS system, Maxiaguine et al. [2005] computed a safe frequency at periodic intervals to prevent buffer overflow of a system. By adopting RTC models, Chen et al. [2009] explored the schedulability for the online DVFS scheduling algorithms proposed in Yao et al. [1995]. Combining optimistic and pessimistic DVFS scheduling, Perathoner et al. [2010] presented an adaptive scheme for the scheduling of arbitrary event streams. When only considering dynamic power management (DPM), Huang et al. [2009b, 2011a] presented an algorithm to find periodic time-driven patterns to turn on/off the processor for energy saving. Online algorithms are proposed in Huang et al. [2009a, 2011b] and Lampka et al. [2011] to adaptively control the power mode of a system, procrastinating the processing of arrived events as late as possible. In one algorithm in Huang et al. [2009a, 2011b], a tight bound of event arrivals is computed based on historical information of event arrivals in the recent past. Instead of using historical information, a dynamic counter technique [Lampka et al. 2011] is used to predict the future workload. Compared to preceding work, the



(a) H.263 decoder on pipeline hardware architecture



(b) power model of a processor

Fig. 1. System model.

distinct difference of ours is that we can tackle the correlation of a pipelined event stream by an inverted use of the pay-burst-only-once principle. With this new method, retrieving this correlation of the same event stream between different pipeline stages, we can compute longer deadlines for each pipeline stage and reduce the overall power consumption of the system.

3. MODELS AND PROBLEM DEFINITION

3.1. Hardware Model

The hardware architecture we have chosen is a simplified one with no shared cache and shared bus among different processing cores. The processing cores are connected in a pipelined fashion via dedicated FIFOs. We consider the system with pipeline architecture shown in Figure 1(a). Subtasks of a partitioned application are mapped and executed in different processors. The processors communicate data only through distributed memory units. Each memory unit can be organized as one or several FIFOs. The data communication and synchronization among processors are realized by blocking read and write SW primitives. This kind of hardware architecture has been realized in Nikolov et al. [2008]. As the service curve of each stage can be computed for energy efficiency by our proposed approaches offline, the worst-case FIFO size of each stage can be determined by applying the analysis approach in Wandeler et al. [2006].

Each processor in the pipelined system has three power consumption modes, namely active, standby, and sleep modes, as shown in Figure 1(b). To serve events, the processor must be in the active mode with power consumption P_a. When there is no event to process, the processor can switch to sleep mode with lower power consumption P_σ. However, mode switching from sleep mode to active mode will cause additional energy and latency penalty, respectively denoted as $E_{sw,on}$ and $t_{sw,on}$. To prevent the processor from frequent mode switches, the processor can stay at standby mode with power consumption P_s, which is less than P_a but more than P_σ, that is, $P_a > P_s > P_\sigma$. Moreover, the mode switch from active (standby) mode to sleep mode will cause energy and time overhead, respectively denoted by $E_{sw,sleep}$ and $t_{sw,sleep}$.

Consider the overhead of switching the system from active mode to sleep mode, the system break-even time T_{BET} denotes the minimum time length that the system stays at sleep mode. If the interval that the system can stay at sleep mode is smaller than T_{BET}, the mode-switch mode overheads are larger than the energy saving, therefore switching mode is not worthwhile. And break-even time T_{BET} can be defined as follows:

$$T_{BET} = \max\left(t_{sw}, \frac{E_{sw}}{P_s - P_\sigma}\right), \tag{1}$$

where $t_{sw} = t_{sw,on} + t_{sw,sleep}$ and $E_{sw} = E_{sw,on} + E_{sw,sleep}$.

3.2. Energy Model

The analytical processor energy model in Martin et al. [2002], Wang and Mishra [2010], Jejurikar et al. [2004], and de Langen and Juurlink [2009] is adopted in this article, whose accuracy has been verified with SPICE simulation [Martin et al. 2002; Wang and Mishra 2010; de Langen and Juurlink 2009]. The dynamic power consumption of the core on one voltage/frequency level (V_{dd}, f) can be given by

$$P_{dyn} = C_{eff} \cdot V_{dd}^2 \cdot f, \tag{2}$$

where V_{dd} is the supply voltage, f the operating frequency, and C_{eff} the effective switching capacitance. The cycle length t_{cycle} is given by a modified alpha power model

$$t_{cycle} = \frac{L_d \cdot K_6}{(V_{dd} - V_{th})^{\alpha}}, \tag{3}$$

where K_6 is technology constant and L_d is estimated by the average logic depth of all instructions' critical path in the processor. The threshold voltage V_{th} is given as

$$V_{th} = V_{th1} - K_1 \cdot V_{dd} - K_2 \cdot V_{bs}, \tag{4}$$

where V_{th1}, K_1, K_2 are technology constants and V_{bs} is the body bias voltage.

The static power is mainly contributed by the subthreshold leakage current I_{subn}, the reverse bias junction current I_j, and the number of devices in the circuit L_g. It can be presented by

$$P_{sta} = L_g \cdot (V_{dd} \cdot I_{subn} + |V_{bs}| \cdot I_j), \tag{5}$$

where the reverse bias junction current I_j is approximated as a constant and the subthreshold leakage current I_{subn} can be determined as

$$I_{subn} = K_3 \cdot e^{K_4 V_{dd}} \cdot e^{K_5 V_{bs}}, \tag{6}$$

where K_3, K_4, and K_5 are technology constants. To avoid junction leakage power overriding the gain in lowering I_{subn}, V_{bs} should be constrained between 0 and -1V. Thus, the power consumption at active mode and at standby mode, that is, P_a and P_s, under one voltage/frequency (V_{dd}, f) can be respectively computed as

$$P_a = P_{dyn} + P_{sta} + P_{on}, \tag{7}$$

$$P_s = P_{sta} + P_{on}, \tag{8}$$

where P_{on} is an inherent power needed for keeping the processor on.

3.3. Task Model

This article considers streaming applications that can be split into a sequence of tasks. As shown in Figure 1(a), an H.263 decoder is represented as four tasks (i.e., PD1, deQ, IDCT, MC) implemented in a pipelined fashion [Oh and Ha 2002]. To model the workload of the application, the concept of arrival curve $\alpha(\Delta) = [\alpha^u(\Delta), \alpha^l(\Delta)]$, originating from network calculus [Le Boudec and Thiran 2001], is adopted. $\alpha^u(\Delta)$ and $\alpha^l(\Delta)$ provide the upper and lower bounds on the number of arrival events for the stream S in any time interval Δ. Many other traditional timing models of event streams can be unified in the concept of arrival curves. For example, a periodic event stream can be modeled by a set of step functions, where $\bar{\alpha}^u(\Delta) = \lfloor \frac{\Delta}{p} \rfloor + 1$ and $\bar{\alpha}^l(\Delta) = \lfloor \frac{\Delta}{p} \rfloor$. For a sporadic event stream with minimal interarrival distance p and maximal interarrival distance p', the upper and lower arrival curves are $\bar{\alpha}^u(\Delta) = \lfloor \frac{\Delta}{p} \rfloor + 1$, $\bar{\alpha}^l(\Delta) = \lfloor \frac{\Delta}{p'} \rfloor$, respectively. Moreover, a widely used model to specify an arrival curve is the PJD model, where the arrival curve is characterized by period p, jitter j, and minimal

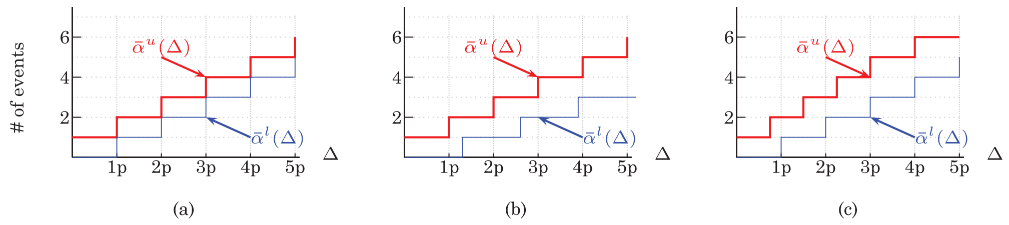


Fig. 2. Examples for arrival curves: (a) periodic events with period p; (b) events with minimal interarrival distance p and maximal interarrival distance $p' = 1.3p$; (c) events with period p, jitter $j = p$, and minimal interarrival distance $d = 0.75p$.

interarrival distance d. In the PJD model, the upper arrival curve can be determined as $\bar{\alpha}^u(\Delta) = \min\{\lceil\frac{\Delta+j}{p}\rceil, \lceil\frac{\Delta}{d}\rceil\}$. Figure 2 depicts arrival curves for the previous cases.

Analogous to arrival curves that provide an abstract event stream model, a tuple $\beta(\Delta) = [\beta^u(\Delta), \beta^l(\Delta)]$ defines an abstract resource model which provides upper and lower bounds on the available resources in any time interval Δ. Further details are referred to Thiele et al. [2000]. Note that arrival curves are event based, meaning they specify the number of events of the steam in one interval of time, while service curves are based on the amount of computation time. Therefore, service curve β has to be transformed to $\bar{\beta}$ to indicate the number of events of the stream that the processor can process in a specified interval time. Suppose that the execution time of an event is c, the transformation of the service curves can be done by $\bar{\beta}^l = \lfloor\frac{\beta^l}{c}\rfloor$ and $\bar{\beta}^u = \lfloor\frac{\beta^u}{c}\rfloor$. With these definitions, a processor with lower service curve $\bar{\beta}^{Gl}(\Delta)$ is said to satisfy the deadline D for the event stream specified by $\alpha^u(\Delta)$ if the following condition holds.

$$\bar{\beta}^{Gl}(\Delta) \geq \alpha^u(\Delta - D),\ \forall\Delta \geq 0 \tag{9}$$

Note that we adopt the same assumption as Maxiaguine et al. [2005], Huang et al. [2009a, 2009b], Lampka et al. [2011], and Chen et al. [2009] and assume the worst-case execution time (WCET) of each task can be predefined and considered as system input in the article. As mentioned in the previous section, the hardware architecture that we have chosen is a simplified one with no shared cache and shared bus among different processing cores. In this sense, we can safely assume the WCET of the running tasks as system inputs.

3.4. Problem Statement

This article considers periodic power management [Huang et al. 2009b] that periodically turns on and off a processor. In each period $T = T_{on} + T_{off}$, it switches the processor to active (standby) mode for T_{on} time units, followed by T_{off} time units in sleep mode, as shown in Figure 1(b). Given a time interval L, where $L \gg T$ and $\frac{L}{T}$ is an integer, suppose that $\gamma(L)$ is the number of events of event stream S served in L. If all the served events finish within L, the energy consumption $E(L, T_{on}, T_{off})$ by applying this periodic scheme is

$$\begin{aligned} E(L, T_{on}, T_{off}) = & \frac{L}{T_{on} + T_{off}}(E_{sw,on} + E_{sw,sleep}) \\ & + \frac{L \cdot T_{on}}{T_{on} + T_{off}} P_s + \frac{L \cdot T_{off}}{T_{on} + T_{off}} P_\sigma \\ & + c \cdot \gamma(L)(P_a - P_s) \end{aligned}$$

$$= \frac{L \cdot E_{sw}}{T_{on} + T_{off}} + \frac{L \cdot T_{on}(P_s - P_\sigma)}{T_{on} + T_{off}} + L \cdot P_\sigma + c \cdot \gamma(L)(P_a - P_s),$$

where E_{sw} is $E_{sw,on} + E_{sw,sleep}$ for brevity. Given a sufficiently large L, without changing the scheduling policy, the minimization of energy consumption $E(L, T_{on}, T_{off})$ of a single processor is to find T_{off} and T_{on} such that the average idle power consumption $P(T_{on}, T_{off})$ is minimized.

$$P(T_{on}, T_{off}) \overset{\text{def}}{=} \frac{\frac{L \cdot E_{sw}}{T_{on}+T_{off}} + \frac{L \cdot T_{on} \cdot (P_s - P_\sigma)}{T_{on}+T_{off}}}{L} = \frac{E_{sw} + T_{on} \cdot (P_s - P_\sigma)}{T_{on} + T_{off}}. \tag{10}$$

By defining $K = \frac{T_{on}}{T_{on}+T_{off}}$, the average idle power consumption P in (10) can be defined by T_{off} and $K (0 \leq K \leq 1)$ as follows:

$$P(K, T_{off}) \overset{\text{def}}{=} \frac{E_{sw}}{T_{off}} + \left((P_s - P_\sigma) - \frac{E_{sw}}{T_{off}} \right) \cdot K. \tag{11}$$

By analyzing (11), it is obvious that the following properties hold.

Property 1. $\forall T_{off}, T_{off} \geq \frac{E_{sw}}{P_s - P_\sigma}$, $P(K, T_{off})$ gets its minimum when K gets its minimum.

Property 2. $\forall T_{off}, T_{off} < \frac{E_{sw}}{P_s - P_\sigma}$, $P(K, T_{off})$ gets its minimum as $P_s - P_\sigma$ when $K = 1$.

According to Properties 1 and 2, when $T_{off} > \frac{E_{sw}}{P_s - P_\sigma}$ holds, the processing unit should turn on as briefly as possible in one period. When $T_{off} \leq \frac{E_{sw}}{P_s - P_\sigma}$ holds, the processing unit should turn on all the time with $T_{off} = 0$. In this context, $\frac{E_{sw}}{P_s - P_\sigma}$ can be seen as the break-even time of the processing unit.

Based on (10), the energy minimization problem of an m-stage pipeline can be formulated as minimizing the function

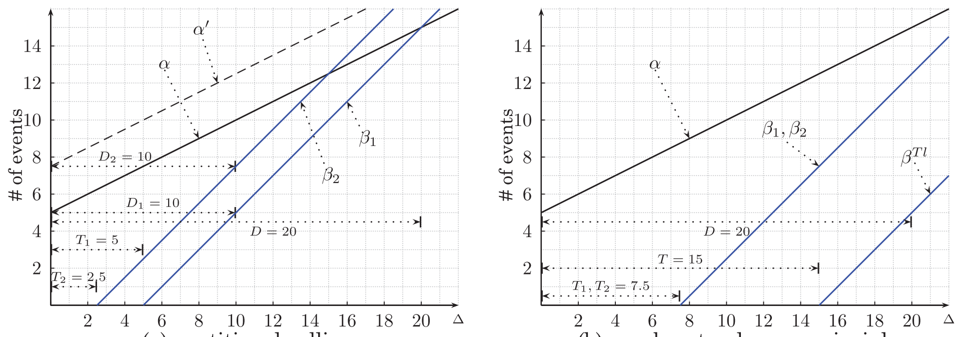
$$P(\vec{T}_{on}, \vec{T}_{off}) = \sum_{i}^{m} \frac{E^i_{sw} + T^i_{on} \cdot \left(P^i_s - P^i_\sigma\right)}{T^i_{on} + T^i_{off}}, \tag{12}$$

where $\vec{T}_{on} = [T^1_{on}\ T^2_{on}\ \dots\ T^m_{on}]$ and $\vec{T}_{off} = [T^1_{off}\ T^2_{off}\ \dots\ T^m_{off}]$. Now we can define the problem that we studied as follows.

> Given pipelined platform with m stages, an event stream $\mathcal{S}$ processed by this pipeline, and an end-to-end deadline requirement D, we are to find a set of periodic power managements characterized by $\vec{T}_{on}$ and $\vec{T}_{off}$ that minimize the average idle power consumption P defined in (12) while guaranteeing that the worst-case end-to-end delay does not exceed D.

4. MOTIVATION EXAMPLE

A phenomenon called pay-burst-only-once is well known and can give a closer upper estimate on the delay when an end-to-end service curve is derived prior to delay computations [Fidler 2003]. When a workload flow with a burst traverses a number of stages in sequence, the effect of the burst of the flow on the end-to-end delay bound is



(a) partition deadline (b) pay-burst-only-once principle

Fig. 3. Motivation example.

the same as if the flow traversed only one node. The end-to-end delay bound computed with this property can be tighter than the sum of delay bounds of each node.

This section presents a motivation example where an event stream passes through a two-stage pipeline with a deadline requirement D. For simplicity, arrival curves in the leaky-bucket form and service curves in rate-latency form [Le Boudec and Thiran 2001] are used. In this representation, an arrival curve is modeled as $\alpha(\Delta) = b + r \cdot \Delta$, where b is the burst and r the leaky rate. Correspondingly, a service curve is modeled as $\beta(\Delta) = R \cdot (\Delta - T)$, where R is service rate and T the delay. A graphical illustration of the example is shown in Figure 3, where $D = 20$, $b = 5$, $r = 0.5$, and $R_1 = R_2 = 1$.

We first inspect the strategy of partitioning the end-to-end deadline and using the partitioned sub-deadlines for the two pipeline stages. For simplicity, we split the D equally, that is, $D/2$ for each stage. As shown in Figure 3(a), given $D/2$ deadline requirement for the first pipeline stage, we obtain the maximal $T_1 = \frac{D}{2} - \frac{b}{R_1} = 5$, corresponding to the minimal service demand $\beta_1 = \Delta - 5$. To derive the minimal β_2 for the second stage of the pipeline is more involved. We need the output arrival curve α' from the first stage. According to Le Boudec and Thiran [2001], $\alpha'(\Delta) = b + r \cdot T_1 + r \cdot \Delta$. Now again, with a deadline requirement $D/2$ for α', we have $T_2 = \frac{D}{2} - \frac{b + r \cdot T_1}{R_1} = 2.5$.

Let's take a close look at this solution. According to the concatenation theorem $\beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{\min(R_1,R_2),T_1+T_2}$, we get a concatenated service curve $\beta = \Delta - (T_1 + T_2) = \Delta - 7.5$. With this concatenated service curve, the maximal overall end-to-end deadline for β_1 and β_2 is 12.5, which is far more strict than D. This example indicates that the obtained β_1 and β_2 based on partitioning the end-to-end deadline are too pessimistic.

The reason for the pessimism comes from paying the burst b/R_1 for the second stage of the pipeline as well as the additional delay $\frac{r \cdot T_1}{R_2}$ from the first stage, as the pay-burst-only-once principle points out. These effects will be accumulated for every stage of the pipeline, leading to even more pessimistic results as the number of pipeline stages increases. In addition, computing the resource demand of each stage requires the lower bound of the output arrival curve from the previous stage. Computing this output curve requires numerical min-plus convolution that will incur considerable computational and memory overheads. In conclusion, the strategy based on partitioning the end-to-end deadline is not a viable approach, in particular for those cases of pipelined systems with many stages.

On the other hand, one can first derive the total concatenated server demand β^{Tl}, in this case $T = 15$ as shown in Figure 3(b). Any partition based on this T will result in smaller but valid service curves for each pipeline stage, as we can always retrieve the original end-to-end deadline by means of the pay-burst-only-once principle. For example, by an equal partition of T, both T_1 and T_2 are 7.5 and D is still preserved. This brings the basic idea of our approach that will be presented in the next section.

5. PROPOSED APPROACH

Our approach lies in an inverse use of the pay-burst-only-once principle, as mentioned in the previous section. Rather than directly partitioning the end-to-end deadline, we compute one service curve for the entire pipeline, which serves as a constraint for the minimal resource demand. The energy minimization problem is then formulated with respect to the resource demands for individual pipeline stages. To solve this minimization problem, the formulation is transformed into a quadratic programming form and solved by a 2-phase heuristic.

Without loss of generality, a pipelined system with m heterogeneous stages ($m \geq 2$) is considered. The processor of the i stage can provide minimal β_i^{Gl} service. Since periodic power management is considered, the minimal service β_i^{Gl} can be modeled as a T_{on}^i and T_{off}^i pair:

$$\beta_i^{Gl}(\Delta) = \left(T_{on}^i \left\lceil \frac{\Delta - T_{off}^i}{T_{on}^i + T_{off}^i} \right\rceil \right) \otimes \Delta. \tag{13}$$

The derivation of Eq. (13) is presented in Lemma A.1 in the appendix section. In addition, to obtain a tight lower bound of the service curve of the entire pipeline, we restrict T_{on}^i as a multiple of the worst-case execution time c_i, that is, $T_{on}^i = n_i c_i, n_i \in N^+$.

5.1. Problem Formulation

Regarding the problem formulation, we first present an approximation approach (see Lemma 5.1) to derive a lower bound of the PPM service curve. By using this approximated curve, we derive the concatenated service curve directly (see Lemma 5.2), which can be used to guarantee the real-time properties (see Theorem 5.3). Then, the energy minimization problem is formulated with respect to the resource demands for individual pipeline stages. Before presenting the formulation, we first state a few basics. By defining $K_i = \frac{T_{on}^i}{T_{on}^i + T_{off}^i}$, we have the following two lemmas.

LEMMA 5.1. $\bar{\beta}_i^{Gl}(\Delta) \geq \frac{K_i}{c_i}(\Delta - T_{off}^i - c_i)$.

PROOF. According to the definition of the min-plus convolution operation $\bigotimes$, the inequality $\lfloor a + b \rfloor \geq \lfloor a \rfloor + \lfloor b \rfloor$, and the inequality Eq. (13), we have

$$\bar{\beta}_i^{Gl}(\Delta) \geq \left\lfloor \frac{T_{on}^i \left\lceil \frac{\Delta - T_{off}^i}{T_{on}^i + T_{off}^i} \right\rceil}{c_i} \right\rfloor \otimes \left\lfloor \frac{\Delta}{c_i} \right\rfloor.$$

With the restriction $T_{on}^i = n_i c_i, n_i \in N^+$ and $\lceil a \rceil \geq a$, we have

$$\begin{aligned} \left\lfloor \frac{T_{on}^i \left\lceil \frac{\Delta - T_{off}^i}{T_{on}^i + T_{off}^i} \right\rceil}{c_i} \right\rfloor &= n_i \cdot \left\lceil \frac{\Delta - T_{off}^i}{T_{on}^i + T_{off}^i} \right\rceil \\ &\geq \frac{K_i}{c_i}\left(\Delta - T_{off}^i\right). \end{aligned}$$

According to $\lfloor a \rfloor \geq a - 1$, we have $\lfloor \frac{\Delta}{c_i} \rfloor \geq \frac{1}{c_i}(\Delta - c_i)$.

According to the rule of min-plus convolution of rate-latency service curve $\beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{min(R_1,R2),T_1+T_2}$ in Le Boudec and Thiran [2001] and $K_i \leq 1$, we have

$$\frac{K_i}{c_i}(\Delta - T^i_{off}) \otimes \frac{1}{c_i}(\Delta - c_i) = \min\left(\frac{K_i}{c_i}, \frac{1}{c_i}\right)(\Delta - T^i_{off} - c_i) = \frac{K_i}{c_i}(\Delta - T^i_{off} - c_i).$$

Then, we get the right side of the inequality. □

Lemma 5.2. $\bigotimes_{i=1}^m \bar{\beta}_i^{Gl} \geq \min_{i=1}^m(\frac{K_i}{c_i})(\Delta - \sum_{i=1}^m(T^i_{off} + c_i))$.

Proof. According to the rule of min-plus convolution of rate-latency service curve $\beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{min(R_1,R2),T_1+T_2}$ in Le Boudec and Thiran [2001] and Lemma 5.1, we have

$$\bigotimes_{i=1}^m \bar{\beta}_i^{Gl} \geq \bigotimes_{i=1}^m \frac{K_i}{c_i}(\Delta - T^i_{off} - c_i) = \min_{i=1}^m\left(\frac{K_i}{c_i}\right)\left(\Delta - \sum_{i=1}^m(T^i_{off} + c_i)\right). \quad \square$$

With Lemma 5.2, we state the next theorem.

Theorem 5.3. *Assuming an event stream modeled with arrival curve α is processed by an m-stage pipeline and the lower service curve of each pipeline stage is defined by a T^i_{on} and T^i_{off} pair, the pipelined system satisfies an end-to-end deadline D if the following condition holds.*

$$\min_{i=1}^m\left(\frac{K_i}{c_i}\right)\left(\Delta - \sum_{i=1}^m(T^i_{off} + c_i)\right) \geq \alpha^u(\Delta - D) \tag{14}$$

Proof. In Lemma 5.2, the right-hand side of the inequality is a lower bound of $\bigotimes_{i=1}^m \bar{\beta}_i^{Gl}$ that is the concatenated service curve of the pipeline. With $\bigotimes_{i=1}^m \bar{\beta}_i^{Gl} \geq \alpha^u(\Delta - D)$, the end-to-end delay of the pipeline is no more than D according to the pay-burst-only-once principle. Therefore, the theorem holds. □

The left-hand side of inequality Eq. (14) can be considered as a bounded delay function $bdf(\Delta, \rho_0, b_0) = max(0, \rho_0(\Delta - b_0))$ with slope $\rho_0 = \min_{i=1}^m(\frac{K_i}{c_i})$ and bounded delay $b_0 = \sum_{i=1}^m(T^i_{off} + c_i)$. For the stream S with deadline D, a set of minimum bounded delay functions $bdf_{min}(\Delta, \rho, b)$ can be derived by varying b (see Section 5.2). Therefore, we should find a solution of $[\vec{K}, \vec{T}_{off}]$ such that the resulting bounded delay function $bdf(\Delta, \rho_0, b_0)$ is no less than the minimum bounded delay functions $bdf_{min}(\Delta, \rho, b)$. Therefore we can formulate our optimization problem as following:

$$\begin{aligned}
\underset{\vec{K},\, \vec{T}_{off}}{\text{minimize}} \quad & P(\vec{K}, \vec{T}_{off}) \\
\text{subject to} \quad & \min_{i=1}^m\left(\frac{K_i}{c_i}\right) \geq \rho \\
& \sum_{i=1}^m(T^i_{off} + c_i) \leq b \\
& 0 \leq K_i \leq 1,\ i = 1, \ldots, m \\
& T^i_{off} \geq 0,\ i = 1, \ldots, m,
\end{aligned} \tag{15}$$

where $\vec{K} = [K_1, \ldots, K_n]$. $P(\vec{K}, \vec{T}_{off})$ is obtained as follows by conducting a transformation $K_i = \frac{T^i_{on}}{T^i_{on}+T^i_{off}}$ to the average power consumption (10) of each stage.

$$P(\vec{K}, \vec{T}_{off}) = \sum_i^m \left(\frac{E^i_{sw}(1-K_i)}{T^i_{off}} + \left(P^i_s - P^i_\sigma\right) K_i \right).$$

The advantage of formulation (15) is twofold. First of all, the service curves of individual pipeline stages are the variables of the optimization problem, which, on the one hand, overcomes the problem of paying the burst multiple times while, on the other hand, avoiding the costly $\otimes$ computation during the optimization. Second, this formulation allows us to use a more efficient method to analyze the problem as presented in the following sections.

5.2. Quadratic Programming Transformation

How to solve the minimization problem (15) is not obvious. The constraints b and ρ, indeed, are not fixed values and in addition these two constraints are correlated. For a fixed b, the minimum bounded delay function $bdf_{min}(\Delta, \rho, b)$ can be determined by computing ρ.

$$\rho = \inf\{\rho : bdf(\Delta, \rho, b) \geq \alpha^u(\Delta - D), \forall \Delta \geq 0\}. \tag{16}$$

In this article, we conduct the optimization by varying b and computing ρ for every possible b. For a fixed b, we can transform (15) into a quadratic programming problem with box constraints (QPB), as stated in the following lemma.

LEMMA 5.4. *The minimization problem in (15) can be transformed as the following quadratic programming problem with box constraints:*

$$\begin{aligned} &\underset{\vec{x}=[x_1 \ldots x_m]}{\text{minimize}} && \vec{x}^T Q \vec{x} \\ &\text{subject to} && 0 \leq x_i \leq \sqrt{E^i_{sw}(1-\rho c_i)},\ i = 1, \ldots, m, \end{aligned} \tag{17}$$

where $Q = A - B$, A is an $m \times m$ matrix of ones and B is an $m \times m$ diagonal matrix with i^{th} diagonal element $\frac{(b-\sum_{j=1}^m c_j)(P^i_s - P^i_\sigma)}{E^i_{sw}}$.

Denote $\vec{x}^*$ as the optimal solution for the QPB *problem in (17), then the optimization solution for (15) can be obtained with $K_i = 1 - \frac{(x^*_i)^2}{E^i_{sw}}$ and $T^i_{off} = \frac{x^*_i}{\sum_{j=1}^m x^*_j}(b - \sum_{j=1}^m c_j)$.*

PROOF. With Cauchy-Buniakowski-Schwartz's inequality, we can get that

$$\sum_{i=1}^m T^i_{off} \cdot \sum_{i=1}^m \frac{E^i_{sw}(1-K_i)}{T^i_{off}} \geq \left(\sum_{i=1}^m \sqrt{E^i_{sw}(1-k_i)} \right)^2.$$

The minimum value of $\sum_{i=1}^m \frac{E^i_{sw}(1-K_i)}{T^i_{off}}$ can be obtained at $\frac{(\sum_{i=1}^m \sqrt{E^i_{sw}(1-k_i)})^2}{b-\sum_{j=1}^m c_j}$ when the following equation holds.

$$T^i_{off} = \frac{\sqrt{E^i_{sw}(1-K_i)}}{\sum_{j=1}^m \sqrt{E^j_{sw}(1-K_j)}} \left(b - \sum_{j=1}^m c_j \right).$$

Then optimization formulation in (15) can be formulated as

$$\begin{array}{ll} \underset{K_1,K_2,\ldots,K_m}{\text{minimize}} & \dfrac{(\sum_{i=1}^{m}\sqrt{E_{sw}^i(1-K_i)})^2}{b-\sum_{j=1}^{m}c_j}+\sum_{i=1}^{m}(P_s^i-P_\sigma^i)K_i \\ \text{subject to} & \rho\, c_i \le K_i \le 1,\ i=1,\ldots,m. \end{array}$$

By defining $x_i = \sqrt{E_{sw}^i(1-K_i)}$, formulation (15) can be transformed as the QPB problem in (17). □

Note that there is a feasible region for b. To guarantee all the resulting $T_{off}^i \ge 0$, the bounded delay b should not be less than $\sum_{i=1}^{m} c_i$. According to (14), the maximum slope ρ of the bounded delay function will not exceed $\frac{1}{\max_{i=1}^{m} c_i}$. Correspondingly, we derive the minimum bounded delay function $bdf_{min}(\Delta, \frac{1}{\max_{i=1}^{m} c_i}, b)$. By inverting (16), we can derive the maximum delay b^u by (18), which can guarantee that all the resulting K_i will not exceed 1. In summary, the feasible region of $b \in [b^l,\ b^u]$ can be bounded as follows:

$$\begin{aligned} b^u &= \sup\left\{d : bdf\left(\Delta, \frac{1}{\max_{i=1}^{m} c_i}, d\right) \ge \alpha^u(\Delta - D), \forall\Delta \ge 0\right\} \\ b^l &= \sum_{i=1}^{m} c_i. \end{aligned} \tag{18}$$

5.3. Quadratic Programming Heuristic

With the preceding information, we can now present the overall algorithm to the energy minimization problem defined in Section 3.4. Basically, bounded delay b is scanned by step ϵ within the range $[b^l,\ b^h]$. For each b, we first solve the subproblem (17) with a QPB solver, and then the obtained solution is repaired to fulfill further constraints (this will be explained later on). The pseudocode of the algorithm is depicted in Algorithm 1.

ALGORITHM 1: Quadratic Programming Heuristic

```
Input: $\alpha^u$, $b^l$, $b^h$, $\epsilon$, and $P_{min} = \infty$
Output: $\vec{K}_{opt}$, $\vec{T}_{off,opt}$
1: for $b = b^l$ to $b^h$ with step $\epsilon$ do
2:     compute $\rho$ by Eqn. 16;
3:     obtain $\vec{K}$ and $\vec{T}_{off}$ by solving (17);
4:     repair $\vec{K}$ and $\vec{T}_{off}$;
5:     if $P(\vec{K}, \vec{T}_{off}) < P_{min}$ then
6:         $\vec{K}_{opt} \leftarrow \vec{K}$; $\vec{T}_{off,opt} \leftarrow \vec{T}_{off}$;
7:         $P_{min} \leftarrow P(\vec{K}_{opt}, \vec{T}_{off,opt})$;
8:     end if
9: end for
```

THEOREM 5.5. *$\exists i \in \{1,2,\ldots,m\}$ and $\frac{E_{sw}^i}{P_s^i - P_\sigma^i} < b - \sum_{j=1}^{m} c_j$, then the problem is NP-hard.*

PROOF. If there exists a stage p_i where the condition $\frac{E_{sw}^i}{P_s^i - P_\sigma^i} < b - \sum_{j=1}^{m} c_j$ holds, the matrix Q in Lemma 5.4 is not positive semi-definite. Thus, QPB is the nonconvex quadratic programming problem which is NP-hard [Jeyakumar et al. 2006]. □

To solve the subproblem (line 3 in Algorithm 1), we apply the existing QPB solver. According to Theorem 5.5, QPB is NP-hard when the scanned bounded delay b is big enough (i.e., $\frac{E^i_{sw}}{P^i_s - P^i_\sigma} < b - \sum_{j=1}^{m} c_j$). It is in general difficult to solve the problem optimally. Nevertheless, there are approximation schemes [Fu et al. 1998] that can efficiently solve the nonconvex QPB and there are many excellent off-the-shelf software packages [Chen and Burer 2012] available. In this article, the state-of-the-art finite B&B algorithm [Chen and Burer 2012] is applied to solve our QPB problem.

After obtaining a pair of $\vec{K}$ and $\vec{T}_{off}$, the repair phase (line 4 in Algorithm 1) is conducted to fulfill further constraints. This repair scheme is represented in Algorithm 2. First of all, the resulting T^i_{off} of pipeline stage i may be smaller than t^i_{sw}. In the case where $T^i_{off} < t^i_{sw}$, turning off the processor of stage i is not possible, therefore the solution for stage i is repaired by $[K'_i, T^{i'}_{off}] = [1, 0]$, stage i is on all the time (line 2 in Algorithm 2). However, this repair step will lead to the loss of sleep time Q_i for each stage (line 21 in Algorithm 2). We record this loss and try to reassign the loss to each stage at the end of the algorithm (lines 21–32 in Algorithm 2) to minimize the power consumption further. Second, the resulting T^i_{on} may not be a multiple of c_i, which is one of our basic requirements. The repair steps are conducted to make T^i_{on} a multiple of c_i (lines 6–20 in Algorithm 2). To guarantee the resulting K'_i is constant with respect to K_i, $T^{i'}_{off}$ should be adjusted to $\frac{T^{i'}_{on}}{K_i} - T^{i'}_{on}$, and ΔT_i indicates how much sleep time of the stage i should be adjusted comparing to the original T^i_{off} (line 14 in Algorithm 2). If $\Delta T_i > 0$ holds, it means that $T^{i'}_{on}$ decreases and the stage i should decrease sleep time T^i_{off} to make K'_i constant (Line 16 in Algorithm 2), which will result in the loss Q_i and this part can be reassigned to prolong the sleep time of other stages. $\Delta T_i \le 0$ indicates that $T^{i'}_{on}$ increases and the stage should increase sleep time T^i_{off} to make K'_i constant. For this case, we make $T^{i'}_{off}$ constant with respect to T^i_{off}, which results in a K'_i increase and power consumption increase ΔE_i (line 18 in Algorithm 2). In the end, the total loss Q should be reassigned to the stage with $\Delta T_i < 0$ to reduce the power consumption further (lines 21–32 in Algorithm 2). The reassignment heuristic uses power increase ΔE_i as a metric to decide which stage should be assigned first. Specifically, the heuristic iterates through all stages that need to compensate and, in each iteration, picks the stage with maximum power increase ΔE_i and increase T^i_{off} without causing $K'_i < K_i$. The reassignment heuristic terminates when there is no loss to reassign or no stage needs to compensate. It is worth noting that the repair phase we conduct can still guarantee the repaired solution to satisfy the constraints, as stated in Lemma 5.6.

LEMMA 5.6. *The solution repaired by Algorithm 2 satisfies the constrains in (15).*

PROOF. The operation in lines 2–20 will not increase the term $\sum_{i=1}^{m} T^i_{off}$ without causing $K'_i < K_i$, which satisfies the constrains in (15). The reassignment heuristic in (lines 21–32) reassigns the total loss Q to each stage that needs to compensate and increase its sleep time T^i_{off} without increasing the total sleep time $\sum_{i=1}^{m} T^i_{off}$ and causing $K'_i < K_i$. Thus, the solution repaired by Algorithm 2 satisfies the constrains in (15). □

5.4. Fast Heuristic

In Section 5.3, we presented a quadratic programming heuristic with QPB transformation. According to Theorem 5.5, QPB is NP-hard when the scanned bounded delay b is big enough. Assume that bounded delay b is scanned by n steps, then the heuristic in Section 5.3 needs to solve this NP-hard problem several times, which is time consuming.

ALGORITHM 2: Repair Scheme

```
Input: solution of QPB problem:$[\vec{K}, \vec{T}_{off}]$
Output: $[\vec{K}', \vec{T}'_{off}]$
 1: compute the stage set: $S_1 = \{p_i | T^i_{off} < t^i_{sw}\}$;
 2: repair $[K', T'_{off}]$ of the stage $p \in S_1$ as $[1, 0]$;
 3: update budget $\Delta T_i \leftarrow T^i_{off}$ and power increase $\Delta E_i \leftarrow 0$ for stages $p \in S_1$;
 4: compute $T_{on}$ and the stage set: $S_2 = \{p_i | T^i_{off} \geq t^i_{sw}\}$;
 5: for each stage $p_i \in S_2$ do
 6:     if $T^i_{on} < c_i$ then
 7:         $T^{i'}_{on} \leftarrow c_i$ ;
 8:     else
 9:         $T^{i'}_{on} \leftarrow \lfloor \frac{T^i_{on}}{c_i} \rfloor c_i$ ;
10:         if $\frac{T^{i'}_{on}}{K_i} - T^{i'}_{on} < t^i_{sw}$ then
11:             $T^{i'}_{on} \leftarrow \lceil \frac{T^i_{on}}{c_i} \rceil c_i$ ;
12:         end if
13:     end if
14:     compute budget $\Delta T_i = T^i_{off} - (\frac{T^{i'}_{on}}{K_i} - T^{i'}_{on})$ ;
15:     if $\Delta T_i >= 0$ then
16:         $T^{i'}_{off} \leftarrow \frac{T^{i'}_{on}}{K_i} - T^{i'}_{on}$ ; $\Delta E_i \leftarrow 0$ ;
17:     else
18:         $T^{i'}_{off} \leftarrow T^i_{off}$; $\Delta E_i \leftarrow P(T^i_{on}, T^i_{off}) - P(T^{i'}_{on}, T^{i'}_{off})$ ;
19:     end if
20: end for
21: compute total budget $Q = \sum_{\Delta T_i > 0} \Delta T_i$;
22: while $Q > 0$ do
23:     find stage $p_i$ with maximum power increase $\Delta E_i$;
24:     if $\Delta T_i < 0$ then
25:         compute available allocation $allo = \min(Q, |\Delta T_i|)$;
26:         $T^{i'}_{off} \leftarrow T^{i'}_{off} + allo$ ; $\Delta T_i \leftarrow \Delta T_i + allo$;
27:         $\Delta E_i \leftarrow P(T^{i'}_{on}, T^{i'}_{off}) - P(T^i_{on}, T^i_{off})$ ;
28:         $Q \leftarrow Q - allo$ ;
29:     else
30:         break ;
31:     end if
32: end while
33: update $[\vec{K}', \vec{T}'_{off}]$;
```

Besides, in the first optimization step, the quadratic programming heuristic does not consider the break-even time constraint (i.e., T^i_{off} of pipeline stage i is not smaller than T^i_{BET}), which could also make the result pessimistic. To overcome these drawbacks, we present a fast heuristic to find a suboptimal solution, running with $O(mn)$ time complexity (m is the stage number). Different from the heuristic in Section 5.3, we consider the break-even time constraint in the optimization phase and partition stage set P as two stage sets according to this constraint, rather than decoupling the break-even time constraint and optimization. Based on this stage-set partition, we can derive a suboptimal solution as stated in Lemma 5.7.

LEMMA 5.7. *Give a fixed bounded delay b and denote $[\vec{K}, \vec{T}_{off}]$ as the optimal solution for the problem. Partition the stage set P into two subsets S_1 and S_2, where $S_1 =$*

$\{p_i | T^i_{off} < T^i_{BET}\}$ and $S_2 = \{p_i | T^i_{off} \geq T^i_{BET}\}$. Then, the optimal solution $[\vec{K}, \vec{T}_{off}]$ can be determined as follows

(*1*) *For the stage $p_i \in S1$, $[K_i, T^i_{off}] = [1, 0]$.*
(*2*) *For the stage $p_i \in S2$, $[K_i, T^i_{off}] = [\rho c_i, x_i]$, where $x_i = \frac{w_i}{\sum_{p_i \in S2} w_i}(b - \sum_{i=1}^m c_i)$ and $w_i = \sqrt{E^i_{sw}(1 - \rho \cdot c_i)}$.*

Proof. For the stage subset S_2, $T^i_{off} \geq T^i_{BET} \geq \frac{E^i_{sw}}{P^i_s - P^i_\sigma}$ holds. The average power consumption $P(K_i, T^i_{off})$ gets its minimum at $K_i = \rho c_i$ according to Property 1. Thus, the average power consumption of the stage subset S_2 can be transformed as $\sum_{p_i \in S2} \frac{w_i^2}{T^i_{off}} + \sum_{p_i \in S2} \rho c_i (P^i_s - P^i_\sigma)$ with constraint $\sum_{p_i \in S2} T^i_{off} \leq b - \sum_{i=1}^m c_i - \sum_{p_i \in S1} T^i_{off}$. According to Cauchy-Buniakowski-Schwartz's inequality, the optimal average power consumption of the stage subset S_2 can be determined as (19) when $[K_i, T^i_{off}] = [\rho c_i, \frac{w_i}{\sum_{p_i \in S2} w_i}(b - \sum_{i=1}^m c_i - \sum_{p_i \in S1} T^i_{off})]$ holds.

$$\sum_{p_i \in S2} P(K_i, T^i_{off}) = \frac{\left(\sum_{p_i \in S2} w_i\right)^2}{b - \sum_{i=1}^m c_i - \sum_{p_i \in S1} T^i_{off}} + \sum_{p_i \in S2} \rho c_i (P^i_s - P^i_\sigma). \tag{19}$$

According to (19), the average power consumption of the stage subset S_2 gets the minimum when $\sum_{p_i \in S1} T^i_{off}$ gets the minimum.

For the stage set S_1, there are two cases: (a) $T^i_{BET} = t^i_{sw}$, where for this case of $T^i_{off} < t^i_{sw}$, turning off the processor of stage i is not possible, as we stated in repair scheme, due to the hardware requirement that the sleep time T^i_{off} of the processor should not be smaller than the overhead t^i_{sw}. Thus, the solution for stage i is forced as $[K_i, T^i_{off}]_{p_i \in S1} = [1, 0]$; (b) $T^i_{BET} = \frac{E^i_{sw}}{P^i_s - P^i_\sigma}$ where, with Property 2, the optimal average power consumption of the stage subset S_1 gets its minimum at $[K_i, T^i_{off}]_{p_i \in S1} = [1, 0]$.

At this point, $\sum_{p_i \in S1} T^i_{off}$ gets the minimum as 0, thus the average power consumption of the stage subset S_2 gets its minimum. □

According to Lemma 5.7, the optimal solution can be derived directly if the stage partition $P = \{S1, S2\}$ is determined. Thus, optimal solution can be derived by exhaustively exploring all possible stage partitions with the complexity $\emptyset(2^n)$. When the stage number increases, the complexity will increase exponentially. To reduce its complexity, a fast stage partition scheme is proposed in this article. In this scheme, we first greedily put all stages into the stage set $S_2 = \{p_i | T^i_{off} \geq T^i_{BET}\}$ (i.e., we assume all the stages can enter sleep mode). Under this greedy partitioning, we compute the optimal T_{off} according to Lemma 5.7 as described in lines 1 and 2 in Algorithm 3. Then, we can assign the stages by checking whether the resulting optimal T_{off} under the greedy partition is greater than T^i_{BET} (see lines 3–9 in Algorithm 3). The feasibility of this partition scheme can be guaranteed by Lemma 5.8.

Lemma 5.8. *Stage partition $P = \{S1, S2\}$ generated by Algorithm 3 is feasible.*

Proof. In Algorithm 3, $\frac{w_i}{\sum_{p_i \in P} w_i}(b - \sum_{i=1}^m c_i) \geq T^i_{BET}$ holds for $S2$. According to Lemma 5.7, T^i_{off} in $S2$ can be determined as $\frac{w_i}{\sum_{p_i \in S2} w_i}(b - \sum_{i=1}^m c_i)$. As $S2 \subseteq P$, we

ALGORITHM 3: Greedy Partition Scheme

```
Input: $\rho, b, P$
Output: $S1$,$S2$
1: Compute $w_i = \sqrt{E^i_{sw}(1 - \rho \cdot c_i)}$ for each stage $p_i$;
2: Compute $x_i = \frac{w_i}{\sum_{p_i \in P} w_i}(b - \sum_{i=1}^{m} c_i)$ for each stage $p_i$;
3: for $p_i \in P$ do
4:     if $x_i < T^i_{BET}$ then
5:         Insert stage $p_i$ into set $S1$;
6:     else
7:         Insert stage $p_i$ into set $S2$;
8:     end if
9: end for
```

can get $T^i_{off} \geq \frac{w_i}{\sum_{p_i \in P} w_i}(b - \sum_{i=1}^{m} c_i) \geq T^i_{BET}$ holds for $S2$, thus the stage partition generated by Algorithm 3 is feasible. □

For each b, we can first obtain a suboptimal partition by the greedy partition scheme depicted in Algorithm 3, and then the optimal solution under the obtained partition can be determined. The pseudocode of the algorithm is depicted in Algorithm 4.

ALGORITHM 4: Fast Heuristic

```
Input: $\alpha^u$, $b^l$, $b^h$ $\epsilon$, $P_{min} = \infty$
Output: $[K_i, T^i_{off}]_{i=1}^m$
1: for $b = b^l$ to $b^h$ with step $\epsilon$ do
2:     compute $\rho$ by Eqn. (16);
3:     generate the feasible partition $S1$ and $S2$ by Algorithm 3;
4:     obtain $\vec{K}$ and $\vec{T}_{off}$ according to Lemma 5.7;
5:     repair $\vec{K}$ and $\vec{T}_{off}$ by Algorithm 2;
6:     if $P(\vec{K}, \vec{T}_{off}) < P_{min}$ then
7:         $\vec{K}_{opt} \leftarrow \vec{K}$; $\vec{T}_{off,opt} \leftarrow \vec{T}_{off}$;
8:         $P_{min} \leftarrow P(\vec{K}_{opt}, \vec{T}_{off,opt})$;
9:     end if
10: end for
```

6. PERFORMANCE EVALUATIONS

In this section, we demonstrate the effectiveness of our approach. We compare three approaches in this section: (1) the pay-burst-only-once algorithm based on quadratic programming (PBOOA-QP) presented in Section 5.3; (2) pay-burst-only-once algorithm based on the fast heuristic (PBOOA-FH) presented in Section 5.4; and (3) the deadline partition algorithm (DPA). DPA partitions the end-to-end deadline into sub-deadlines for individual pipeline stages and explores all the possible deadline partition combinations to find that deadline partition with the minimum energy consumption. For each deadline partition combination, DPA uses the scheme in Huang et al. [2009b] to minimize the energy consumption of individual pipeline stages to optimize the overall energy consumption. To show the effects of our scheme, we report the average idle power computed as Eq. (10) as well as the computation time of all the schemes. The simulation is implemented in Matlab using the RTC toolbox [Wandeler and Thiele 2006] and the finite B&B algorithm [Chen and Burer 2012] is used to solve QPB. All results are obtained from a 2.83 GHz processor with 4GB memory.

Table I. Constants for 70nm Technology [Martin et al. 2002; Wang and Mishra 2010]

Const	Value	Const	Value	Const	Value
K_1	0.063	K_6	5.26×10^{-12}	V_{th1}	0.244
K_2	0.153	K_7	-0.144	I_j	4.8×10^{-10}
K_3	5.38×10^{-7}	V_{dd}	[0.5,1]	C_{eff}	0.43×10^{-9}
K_4	1.83	V_{bs}	[−1,0]	L_d	37
K_5	4.19	α	1.5	L_g	4×10^6

Table II. Power Parameters

V_{dd}	P_a	P_s	P_σ	E_{sw}	t_{sw}
$0.7V$	$656mW$	$390mW$	$50\mu W$	$483\mu J$	$10ms$

6.1. Simulation Setup

The experiments are conducted based on the classical energy model of a 70nm technology processor in Martin et al. [2002], Wang and Mishra [2010], Jejurikar et al. [2004], and Chen et al. [2014], whose accuracy has been verified with SPICE simulation. Table I lists the energy parameter under 70nm technology [Martin et al. 2002; Wang and Mishra 2010; Jejurikar et al. 2004; Chen et al. 2014]. According to Jejurikar et al. [2004], executing at $V_{dd} = 0.7V$ is more energy efficient than executing at lower voltage levels. To achieve the minimization of the overall energy consumption of the system, we assume that the processor runs at this critical frequency level when the processor is in the active state. From Wang and Mishra [2010] and Jejurikar et al. [2004], body bias voltage V_{bs} is obtained as $-0.7V$. From Jejurikar et al. [2004], P_{on} related to idle power can be obtained as $100mW$ and the power consumption in sleep mode P_σ is set as $50\mu W$. In Jejurikar et al. [2004], we can obtain energy overhead E_{sw} of the state transition as $483\mu J$. We set time overhead t_{sw} of the state transition as 10ms. According to the energy parameter in Table I and the energy model in Section 3.2, we can calculate the corresponding active power P_a and standby power P_s under voltage level $V_{dd} = 0.7V$. Table II lists all the power parameters used in the experiment.

An event stream is specified by the PJD model with period p, jitter j, and minimal interarrival distance d. It is worth noting that a worst-case execution time c is associated with the service curve of different stages, as stated in Section 3.3. The jitter j and the relative deadline D of the stream are respectively defined as $j = \varphi \cdot p$ and $D = \gamma \cdot p$ and vary according to the corresponding factors.

To evaluate the effectiveness of our approach, we conduct the experiments with three applications. We collected results for these three applications with deadline and jitter varying with the corresponding factors γ and φ. In the following, we give a brief overview of the three applications. The H.263 decoder application [Oh and Ha 2002] was modeled by four tasks consisting of packet decoding (PD1), an inverse quantization operation (deQ), an inverse DCT operation (IDCT), and motion compensation (MC). The execution time of each subtask in the H.263 decoder application can be found in Oh and Ha [2002]. The activation period of the H.263 decoder application is 100ms with varying the jitter and the end-to-end deadline. The MP3 decoder application is implemented in a pipelined fashion [Oh and Ha 2002] that can be split into five tasks, including packet decoding (PD2), Huffman decoding (HD), the inverse quantization operation (deQ), an inverse DCT operation (IDCT), and antialiasing (FB). The execution time of each subtask in the H.263 decoder application can be found in Oh and Ha [2002]. The activation period of the MP3 decoder application is 100ms with varying the jitter and the end-to-end deadline. Time Delay Equalization (TDE) comes from the GMTI (Ground Moving Target Indiciator) application obtained from StreamIt benchmarks [Thies and

Table III. Average Power Savings with Respect to DPA

	H.263 2-stages	MP3 2-stages	TDE 2-stages	H.263 3-stages	MP3 3-stages	TDE 3-stages
PBOOA-QP	10.46%	11.57%	39.62%	23.59%	26.37%	30.60%
PBOOA-FH	10.46%	11.57%	39.65%	23.31%	25.69%	34.09%

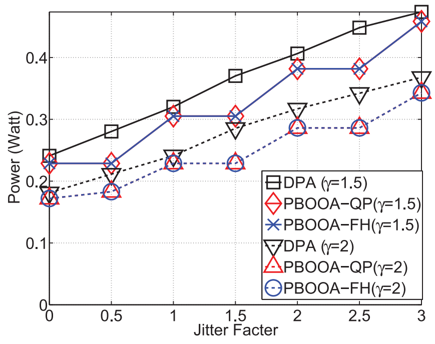
Amarasinghe 2010]. The Time Delay Equalization (TDE) application contains 4 tasks, including tasks like FFT reorder, combined DFT, FFT reorder, and combined IDFT. We set the activation period of the consumer application as 30ms.

6.2. Simulation Result

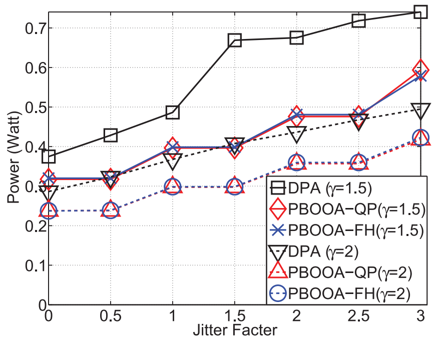
We first evaluate how the power consumptions of the compared approaches change as the jitter and deadline vary. Cases of 2-stage and 3-stage pipeline architectures with homogeneous 70nm processors are evaluated. We vary the jitter factor φ from 0–3 with step 0.5 and the deadline factor γ from 1.5–2 with step 0.5. The simulation results of the three approaches are shown in Figure 4. In the figure, each line represents the average energy consumption under the varied jitter factor settings with the fixed deadline factor and task mapping. From these, we can make the following observations: (1) pay-bust-only-once-based approaches always outperform the deadline partition approach for all settings on both pipeline architectures. We list average normailized power savings of PBOOA-QP and PBOOA-FH with respect to DPA in Table III; (2) the average idle power consumptions of the three approaches increase as jitter increases, since bigger jitter requires longer T_{on} to gurantee the worst-case end-to-end deadline; (3) the average idle power consumptions of the three approaches decrease as end-to-end deadline increases. This is expected becasue the loose end-to-end deadline requirement could result in smaller execution time T_{on} and longer sleep time T_{off}; (4) one interesting obervation is that pay-bust-only-once-based approaches can achieve more power savings on 3-stage pipeline rather than 2-stage pipeline for different jitter and deadline settings. This is because DPA on 3-stage pipeline pay burst more times than 2-stage pipeline, which leads PBOOA-QP and PBOOA-FH to achieve more power savings on 3-satge pipeline.

Next, we conduct the experiment to show the impact of time overhead of state transition t_{sw} on the effectiveness of our approaches. An H.263 application with jitter factor $\varphi = 0.5$ and deadline factor $\gamma = 1$ runs in 3-stage pipeline architectures with homogeneous 70nm processors. We vary the time overhead of state transition t_{sw} from 5ms–15ms with fixed step size 1ms. Figure 5 illustrates the average power consumptions for the three compared approaches. In this figure, we can observe that our approaches can find efficient solutions and outperform DPA in all of t_{sw} settings. Besides, when t_{sw} increases, the average power consumptions of DPA increase faster compared to pay-burst-only-once-based approaches. This is because DPA generates the less idle time due to suffering from a paying burst many times compared to pay-burst-only-once-based approaches, as we show in Section 4. The increase of t_{sw} will reduce the opportunities for turnning off the processor, which means that entering sleep mode should be more difficult for DPA.

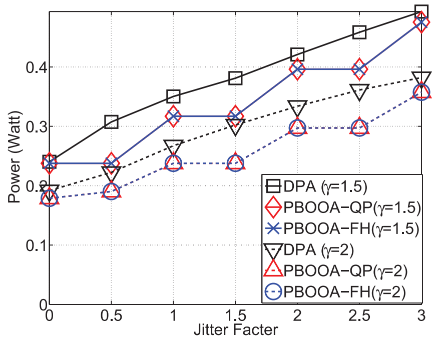
Then, we discuss the impact of the period setting on the effectiveness of the approaches. The MP3 application with jitter factor $\varphi = 1$ and deadline factor $\gamma = 1.5$ runs in two-stage pipeline architectures with homogeneous 70nm processors, where we vary period settings from 70–130ms with fixed step size 10ms. Figure 6 illustrates the average power consumptions for the three compared approaches under different period settings. From Figure 6, we can see that the pay-burst-only-once-based approaches outperform DPA at all period settings. Furthermore, the average power consumption of all approaches decreases when the period increases. This is expected because a bigger period of the application can prolong the idle intervals.



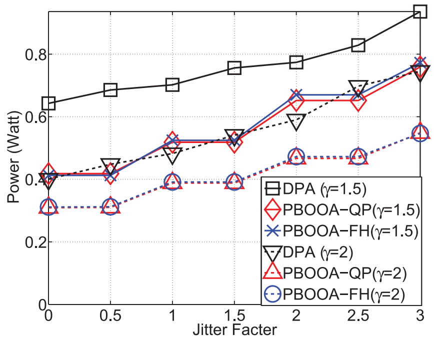
(a) H.263 on 2-stages pipeline (PD1, deQ $\rightarrow$ Core 1, IDCT, MC $\rightarrow$ Core 2)



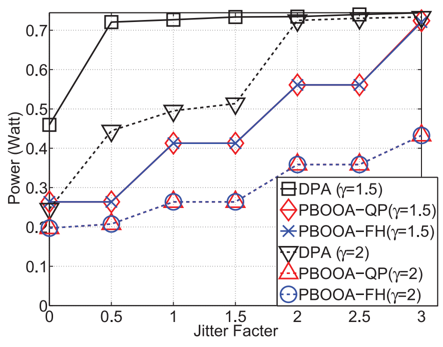
(b) H.263 on 3-stages pipeline (PD1, deQ $\rightarrow$ Core 1, IDCT $\rightarrow$ Core 2, MC $\rightarrow$ Core 3)



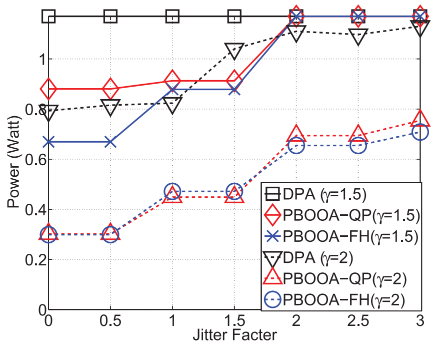
(c) MP3 on 2-stages pipeline (PD2, HD, deQ $\rightarrow$ Core 1, IDCT, FB $\rightarrow$ Core 2)



(d) MP3 on 3-stages pipeline (PD2, HD, deQ $\rightarrow$ Core 1, IDCT $\rightarrow$ Core 2, FB $\rightarrow$ Core 3)



(e) TDE on 2-stages pipeline (FFT, DFT $\rightarrow$ Core 1, FFT, IDFT $\rightarrow$ Core 2, FB $\rightarrow$ Core 3)



(f) TDE on 3-stages pipeline (FFT $\rightarrow$ Core 1, DFT $\rightarrow$ Core 2, FFT, IDFT $\rightarrow$ Core 3)

Fig. 4. Average idle power consumption for three applications on 2-stage and 3-stage pipeline architectures.

In the end, we demonstrate the scalability of our approaches. We test them by up to a 20-stage heterogeneous pipeline. The execution time of subtasks mapped on each stage are randomly generated between 5–15ms. According to the power model presented in Section 3.2, the power profile of each stage can be generated by randomly selecting voltage V_{dd} between $0.5V$–$0.8V$. The activation period of the event stream is 40ms with jitter factor $\varphi = 1$. The end-to-end deadline for the test case

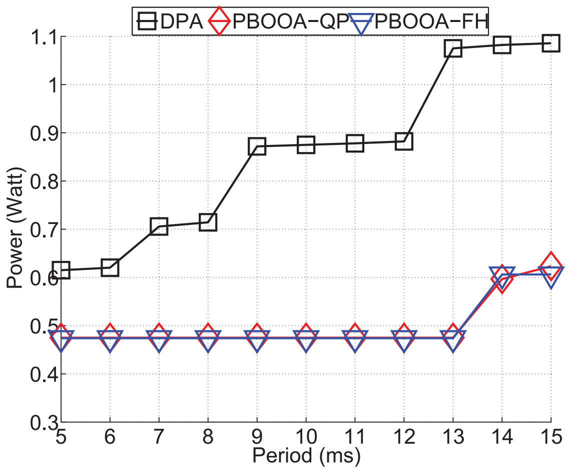


Fig. 5. Average power consumption with varying t_{sw}.

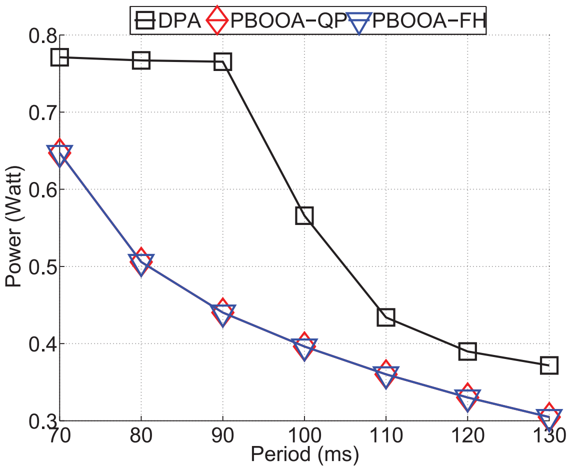
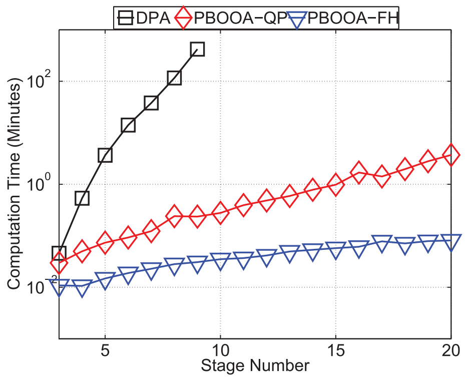
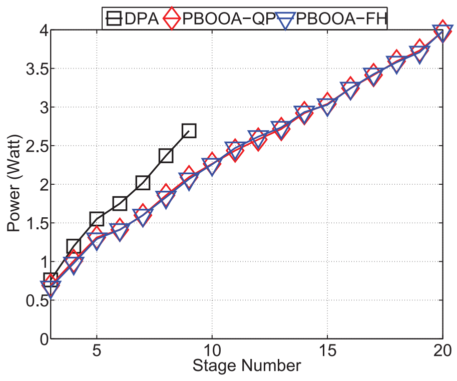


Fig. 6. Average power consumption with varying period.

with different stage number is determined by $n \cdot 20$, where n is the stage number. The overhead values of state transition t_{sw} and E_{sw} of different stages are randomly selected in $[1ms,\ 5ms]$ and $[400uJ,\ 800uJ]$, respectively. Based on the observation that the deadline partition algorithm may suffer deadline combination explosion and the costly $\bigotimes$ computation, we set the search step as 5 for the three compared approaches. Figure 7 shows the power consumption and computation overhead on different pipeline architectures. From this figure, we have the following observations: (1) as shown in Figure 7(a), the computation overhead of the deadline partition algorithm increases exponentially. When the stage number exceeds 10, the Deadline Partition Algorithm (DPA) fails to generate the results due to the expiration of time budget of 8 hours. For the case of 9-stage pipeline, DPA takes almost 420 minutes, which is 9182× longer than the 3-stage pipeline case. This is expected because the deadline combinations will increase exponentially as the stage number increases. In addition, as the stage number increases, the time for computing the resource demand of each following stage, which requires the lower bound of the output arrival curve from the previous stage, increases.



(a) time consumption



(b) power consumption

Fig. 7. Computation time and power computation for heterogeneous pipelined system.

Computing this output curve requires numerical min-plus convolution that will incur considerable computational and memory overheads; (2) compared to the deadline partition algorithm, pay-burst-only-once-based approaches are fast and the computation time increases slowly with respect to the stage number, especially for PBOOA-FH. With the case of 20-stage pipeline, the PBOOA-QP approach takes 3.7 minutes, 124× more computing time than the 3-stage case. PBOOA-FH takes only 0.08 minutes to generate the result, only 7.5× than the 3-stage case; (3) in the context of average idle power consumption, pay-burst-only-once-based approaches are more energy efficient than the deadline partition algorithm. In Figure 7(b), we can see PBOOA-QP and PBOOA-FH approaches always outperform DPA for all pipeline architectures, indicating that our approaches are not only faster but also more energy efficient than the DPA approach. Besides, as observed in prior experiments, the gap in power consumption between the deadline partition algorithm and pay-burst-only-once-based algorithm increase as the stage number increases. This is expected because, as the stage number increases, the times when DPA should pay burst also increase. In contrast, the proposed approaches only need to pay burst only once, which leads the tighter end-to-end delay bound and prolongs the idle intervals of the stages for energy efficiency; (4) the PBOOA-FH approach can achieve almost identical average idle power consumption with respect to the PBOOA-QP approach with almost 10× speedup. In some cases, PBOOA-FH can even achieve more energy savings than PBOOA-QP. This is because that, in contrast to the PBOOA-QP approach, PBOOA-FH integrates break-even time constraints into the optimization phase, which leads the PBOOA-FH approach to find better solutions than the PBOOA-QP approach.

7. CONCLUSION

This article presents new approaches to minimize energy consumption for pipelined systems. Targeting the streaming application with nondeterministic workload arrivals under hard real-time constraints, our approaches can not only guarantee the original end-to-end deadline requirement but also retrieve the pay-burst-only-once phenomena, resulting in a significant reduction in both the energy consumption and computing overhead. Moreover, our approaches are scalable with respect to the number of pipelined stages. Simulation results demonstrate the effectiveness of our approaches. In the future, we intend to extend our approaches to dynamic voltage-frequency scaling (DVFS) to reduce dynamic power for pipelined systems. Another interesting future work would be to target multidimensional issues such as energy and thermal constraints simultaneously. In addition, how to combine our approaches with consideration of the mapping of the application is also deemed worthy for our future work.

APPENDIX

LEMMA A.1. *The service curve of period power management specified by T_{on} and T_{off} can be represented as follows.*

$$\beta_i^{Gl}(\Delta) = \left(T_{on}^i \left\lceil \frac{\Delta - T_{off}^i}{T_{on}^i + T_{off}^i} \right\rceil\right) \otimes \Delta. \tag{20}$$

PROOF. According to Huang et al. [2009b], the service curve of period power management specified by T_{on} and T_{off} can be represented as Eq. (21).

$$\beta^{Gl}(\Delta) = \max\left(\left\lfloor \frac{\Delta}{T_{on} + T_{off}} \right\rfloor \cdot T_{on},\ \Delta - \left\lceil \frac{\Delta}{T_{on} + T_{off}} \right\rceil \cdot T_{off}\right). \tag{21}$$

This proof presents the derivation of Eq. (20), which is used to represent the service curve of period power management, to indicate that Eqs. (21) and (20) are equivalent.
According to the definition of the min-plus convolution,

$$\begin{aligned}\beta^{Gl}(\Delta) &= \left(T_{on} \left\lceil \frac{\Delta - T_{off}}{T_{on} + T_{off}} \right\rceil\right) \otimes \Delta \\ &= \inf_{0 \le s \le \Delta} \left(\Delta - s + T_{on} \cdot \left\lceil \frac{s - T_{off}}{T_{on} + T_{off}} \right\rceil\right).\end{aligned}$$

We make some transformations as follows.

$$\begin{aligned} T &= T_{on} + T_{off} \\ \Delta &= k_\Delta \cdot T + r_\Delta, \quad k_\Delta \in N^+, 0 \le r_\Delta < T, \\ s &= k_s \cdot T + r_s, \quad k_s \in N^+, 0 \le r_s < T. \end{aligned}$$

Then, we have

$$\beta^{Gl}(\Delta) = \inf_{0 \le s \le \Delta} \left((k_\Delta - k_s) \cdot T + (r_\Delta - r_s) + T_{on} \cdot k_s + T_{on} \cdot \left\lceil \frac{r_s - T_{off}}{T} \right\rceil\right). \tag{22}$$

As $s \le \Delta$, there are two possibilities between the parameters r_Δ and r_s: (1) when $k_s = k_\Delta$, $r_s \le r_\Delta$ should be held for $s \le \Delta$; (2) when $k_s \le k_\Delta - 1$, there is no constraint between r_Δ and r_s because $k_s \le k_\Delta - 1$ is sufficient to guarantee $s \le \Delta$.

Case 1: $k_s \le k_\Delta - 1$. For this case, there are no constraints between r_Δ and r_s, thus we can have Eq. (23) by calling Eq. (22).

$$\begin{aligned}\beta^{Gl}(\Delta) &= \inf_{0 \le s \le \Delta} \left((k_\Delta - k_s) \cdot T + (r_\Delta - r_s) + k_s \cdot T_{on} + T_{on} \cdot \left\lceil \frac{r_s - T_{off}}{T} \right\rceil\right) \\ &= \inf_{0 \le s \le \Delta} \left(k_\Delta \cdot T + r_\Delta - k_s \cdot (T - T_{on}) - r_s + T_{on} \cdot \left\lceil \frac{r_s - T_{off}}{T} \right\rceil\right) \\ &= \inf_{0 \le s \le \Delta} \left(k_\Delta \cdot T + r_\Delta - k_s \cdot T_{off} - r_s + T_{on} \cdot \left\lceil \frac{r_s - T_{off}}{T} \right\rceil\right). \end{aligned} \tag{23}$$

—When $T_{off} < r_s < T$ holds, we have Eq. (24) by calling Eq. (23).

$$\begin{aligned}\beta^{Gl}(\Delta) &= \inf_{0\le s\le\Delta}(k_\Delta\cdot T + r_\Delta - k_s\cdot T_{off} - r_s + T_{on})\\ &> (k_\Delta\cdot T + r_\Delta - k_s\cdot T_{off} - T + T_{on})\\ &= (k_\Delta\cdot T + r_\Delta - k_s\cdot T_{off} - T_{off})\\ &\ge (k_\Delta\cdot T + r_\Delta - k_\Delta\cdot T_{off}).\end{aligned} \tag{24}$$

—When $0 \le r_s \le T_{off}$ holds, we have Eq. (25) by calling Eq. (23).

$$\beta^{Gl}(\Delta) = \inf_{0\le s\le\Delta}(k_\Delta\cdot T + r_\Delta - k_s\cdot T_{off} - r_s) \xRightarrow[r_s=T_{off},k_s=k_\Delta-1]{inf} k_\Delta\cdot T + r_\Delta - k_\Delta\cdot T_{off}. \tag{25}$$

For the preceding two cases, the infimum of $\beta^{Gl}_{k_s\le k_\Delta-1}(\Delta)$ for the case $k_s \le k_\Delta - 1$ can be obtained as Eq. (26) by calling Eqs. (24) and (25).

$$\beta^{Gl}_{k_s\le k_\Delta-1}(\Delta) = k_\Delta\cdot T + r_\Delta - k_\Delta\cdot T_{off} = \Delta - k_\Delta\cdot T_{off}. \tag{26}$$

Case 2: $k_s = k_\Delta$. For this case, $r_s \le r_\Delta$ should be held for $s \le \Delta$, thus we have Eq. (27) by calling Eq. (22).

$$\beta^{Gl}(\Delta) = \inf_{0\le s\le\Delta}\left((r_\Delta - r_s) + k_\Delta\cdot T_{on} + T_{on}\cdot\left\lceil\frac{r_s - T_{off}}{T}\right\rceil\right). \tag{27}$$

As r_s should be constrained by r_Δ, there are two cases for r_Δ.

—$r_\Delta \le T_{off}$. For this case, we have $0 \le r_s \le r_\Delta \le T_{off}$, thus we have Eq. (28) by calling Eq. (27).

$$\beta^{Gl}_{k_s=k_\Delta,r_\Delta\le T_{off}}(\Delta) = \inf_{0\le s\le\Delta}((r_\Delta - r_s) + k_\Delta\cdot T_{on}) \xRightarrow[r_s=r_\Delta]{inf} k_\Delta\cdot T_{on}. \tag{28}$$

By integrating the cases of $k_s = k_\Delta$ and $k_s \le k_\Delta - 1$, we have Eq. (29) by calling Eqs. (26) and (28).

$$\beta^{Gl}_{r_\Delta\le T_{off}}(\Delta) = \min(\Delta - k_\Delta\cdot T_{off}, k_\Delta\cdot T_{on}) = k_\Delta\cdot T_{on}. \tag{29}$$

—$r_\Delta > T_{off}$. For this case, there are two subcases for r_s, that is, $0 \le r_s \le T_{off} < r_\Delta < T$ and $T_{off} < r_s \le r_\Delta < T$.

—$0 \le r_s \le T_{off} < r_\Delta < T$. For this case, we have Eq. (30) by calling Eq. (27).

$$\beta^{Gl}(\Delta) = \inf_{0\le s\le\Delta}((r_\Delta - r_s) + k_\Delta\cdot T_{on}) \xRightarrow[r_s=T_{off}]{inf} r_\Delta - T_{off} + k_\Delta\cdot T_{on} < T_{on} + k_\Delta\cdot T_{on}. \tag{30}$$

—$T_{off} < r_s \le r_\Delta < T$. For this case, we have Eq. (31) by calling Eq. (27).

$$\beta^{Gl}(\Delta) = \inf_{0\le s\le\Delta}((r_\Delta - r_s) + k_\Delta\cdot T_{on} + T_{on}) \xRightarrow[r_s=r_\Delta]{inf} T_{on} + k_\Delta\cdot T_{on}. \tag{31}$$

For the prior two cases, the infimum of $\beta^{Gl}_{k_s=k_\Delta,r_\Delta>T_{off}}(\Delta)$ can be obtained as Eq. (32) by calling Eqs. (30) and (31).

$$\beta^{Gl}_{k_s=k_\Delta,r_\Delta>T_{off}}(\Delta) = r_\Delta - T_{off} + k_\Delta\cdot T_{on} = \Delta - (k_\Delta + 1)\cdot T_{off}. \tag{32}$$

By integrating the cases of $k_s = k_\Delta$ and $k_s \leq k_\Delta - 1$, we have Eq. (33) by calling Eqs. (32) and (26).

$$\beta^{Gl}_{r_\Delta > T_{off}}(\Delta) = \min(\Delta - k_\Delta \cdot T_{off}, \Delta - (k_\Delta + 1) \cdot T_{off}) = \Delta - (k_\Delta + 1) \cdot T_{off}. \tag{33}$$

With Eqs. (29) and (33), we can obtain the service curve as Eq. (34).

$$\beta^{Gl} = \begin{cases} k_\Delta \cdot T_{on} & r_\Delta \leq T_{off} \\ \Delta - (k_\Delta + 1) \cdot T_{off} & r_\Delta > T_{off} \end{cases}. \tag{34}$$

When $0 \leq r_\Delta \leq T_{off}$ holds, we have $k_\Delta \cdot T_{on} \geq \Delta - (k_\Delta + 1) \cdot T_{off}$ and $k_\Delta = \lfloor \frac{\Delta}{T_{on}+T_{off}} \rfloor$. When $r_\Delta > T_{off}$ holds, we have $k_\Delta \cdot T_{on} < \Delta - (k_\Delta + 1) \cdot T_{off}$ and $k_\Delta + 1 = \lceil \frac{\Delta}{T_{on}+T_{off}} \rceil$.

Then, we can obtain the service curve as Eq. (35).

$$\beta^{Gl} = \max(k_\Delta \cdot T_{on}, \Delta - (k_\Delta + 1) \cdot T_{off}). \tag{35}$$

By transforming Eq. (35), we can obtain Eq. (21), thus the service curve of period power management can be represented as Eq. (20). □

REFERENCES

A. Alimonda, S. Carta, A. Acquaviva, A. Pisano, and L. Benini. 2009. A feedback-based approach to DVFS in data-flow applications. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst.* 28, 11, 1691–1704.

S. Carta, A. Alimonda, A. Pisano, A. Acquaviva, and L. Benini. 2007. A control theoretic approach to energy-efficient pipelined computation in mpsocs. *ACM Trans. Embedd. Comput. Syst.* 6, 4.

G. Chen, K. Huang, C. Buckl, and A. Knoll. 2013. Energy optimization with worst-case deadline guarantee for pipelined multiprocessor systems. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'13).*

G. Chen, K. Huang, and A. Knoll. 2014. Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination. *ACM Trans. Embedd. Comput. Syst.* 13, 3.

J. Chen and S. Burer. 2012. Globally solving nonconvex quadratic programming problems via completely positive programming. *Math. Program. Comput.* 4, 1, 33–52.

J. J. Chen, N. Stoimenov, and L. Thiele. 2009. Feasibility analysis of on-line DVS algorithms for scheduling arbitrary event streams. In *Proceedings of the 30th IEEE Real-Time Systems Symposium (RTSS'09).*

A. Davare, Q. Zhu, M. Di Natale, C. Pinello, S. Kanajan, and A. Sangiovanni-Vincentelli. 2007. Period optimization for hard real-time distributed automotive systems. In *Proceedings of 44th ACM/IEEE Design Automation Conference (DAC'07).*

P. de Langen and B. Juurlink. 2006. Leakage-aware multiprocessor scheduling for low power. In *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS'06).*

P. de Langen and B. Juurlink. 2009. Leakage-aware multiprocessor scheduling. *J. Signal Process. Syst.* 57, 1, 73–88.

M. Fidler. 2003. Extending the network calculus pay bursts only once principle to aggregate scheduling. In *Proceedings of the 2nd International Workshop on Quality of Service in Multiservice IP Networks (QoS-IP'03).* 19–34.

M. Fu, Z. Luo, and Y. Ye. 1998. Approximation algorithms for quadratic programming. *J. Combinat. Optim.* 2, 1, 29–50.

S. Y. Hong, T. Chantem, and X. S. Hu. 2011. Meeting end-to-end deadlines through distributed local deadline assignments. In *Proceedings of the 32nd IEEE Real-Time Systems Symposium (RTSS'11).*

K. Huang, J. J. Chen, and L. Thiele. 2011a. Energy-efficient scheduling algorithms for periodic power management for real-time event streams. In *Proceedings of the 17th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'11).*

K. Huang, L. Santinelli, J. J. Chen, L. Thiele, and G. C. Buttazzo. 2009a. Adaptive dynamic power management for hard real-time systems. In *Proceedings of the 30th IEEE Real-Time Systems Symposium (RTSS'09).*

K. Huang, L. Santinelli, J. J. Chen, L. Thiele, and G. C. Buttazzo. 2009b. Periodic power management schemes for real-time event streams. In *Proceedings of the 48th IEEE International Conference on Decision and Control (CDC'09).*

K. Huang, L. Santinelli, J. J. Chen, L. Thiele, and G. C. Buttazzo. 2011b. Applying real-time interface and calculus for dynamic power management in hard real-time systems. *Real-Time Syst.* 47, 2, 163–193.

H. Javaid and S. Parameswaran. 2009. A design flow for application specific heterogeneous pipelined multiprocessor systems. In *Proceedings of the 46th ACM/IEEE Annual Design Automation Conference (DAC'09).*

H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran. 2011a. System-level application-aware dynamic power management in adaptive pipelined MPSoCs for multimedia. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'11).*

H. Javaid, M. Shafique, S. Parameswaran, and J. Henkel. 2011b. Low-power adaptive pipelined MPSoCs for multimedia: An h.264 video encoder case study. In *Proceedings of the 48th ACM/EDAC/IEEE Design Automation Conference (DAC'11).*

R. Jejurikar, C. Pereira, and R. Gupta. 2004. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the 41st ACM/IEEE Design Automation Conference (DAC'04).*

V. Jeyakumar, A. M. Rubinov, and Z. Y. Wu. 2006. Sufficient global optimality conditions for non-convex quadratic minimization problems with box constraints. *J. Global Optim.* 36, 3, 471–481.

I. Karkowski and H. Corporaal. 1997. Design of heterogeneous multi-processor embedded systems: Applying functional pipelining. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT'97).* 156.

K. Lampka, K. Huang, and J. J. Chen. 2011. Dynamic counters and the efficient and effective online power management of embedded real-time systems. In *Proceedings of the 7th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'11).*

J. Y. Le Boudec and P. Thiran. 2001. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet.* Springer.

D. Liu, J. Spasic, J. T. Zhai, T. Stefanov, and G. Chen. 2014. Resource optimization of CSDF-modeled streaming applications with latency constraints. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'14).*

S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw. 2002. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD'02).*

S. Maxiaguine, A. Chakraborty, and L. Thiele. 2005. DVS for buffer-constrained architectures with predictable QoS-energy tradeoffs. In *Proceedings of the IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'05).*

H. Nikolov, T. Stefanov, and E. Deprettere. 2008. Systematic and automated multiprocessor system design, programming, and implementation. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst.* 27, 3, 542–555.

H. Oh and S. Ha. 2002. Hardware-software cosynthesis of multi-mode multi-task embedded systems with real-time constraints. In *Proceedings of the 10th International Symposium on Hardware/Software Codesign (CODES+ISSS'02).*

S. Perathoner, K. Lampka, N. Stoimenov, L. Thiele, and J. J. Chen. 2010. Combining optimistic and pessimistic DVS scheduling: An adaptive scheme and analysis. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'10).*

S. L. Shee, A. Erdos, and S. Parameswaran. 2006. Heterogeneous multiprocessor implementations for jpeg: A case study. In *Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'06).*

S. L. Shee and S. Parameswaran. 2007. Design methodology for pipelined heterogeneous multiprocessor system. In *Proceedings of the 44th Annual Design Automation Conference (DAC'07).*

L. Thiele, S. Chakraborty, and M. Naedele. 2000. Real-time calculus for scheduling hard real-time systems. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'00).*

W. Thies and S. Amarasinghe. 2010. An empirical characterization of stream programs and its implications for language and compiler design. In *Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques (PACT'10).*

E. Wandeler and L. Thiele. 2006. Real-time calculus (rtc) toolbox. http://www.mpa.ethz.ch/Rtctoolbox.

E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse. 2006. System architecture evaluation using modular performance analysis - A case study. *Int. J. Softw. Tools Technol. Transfer* 8, 6, 649–667.

W. X. Wang and P. Mishra. 2010. Leakage-aware energy minimization using dynamic voltage scaling and cache reconfiguration in real-time systems. In *Proceedings of the 23rd International Conference on VLSI Design (VLSID'10).*

R. B. Xu, R. Melhem, and D. Mosse. 2007. Energy-aware scheduling for streaming applications on chip multiprocessors. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS'07).*

F. Yao, A. Demers, and S. Shenker. 1995. A scheduling model for reduced CPU energy. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*.

Y. Yu and V. K. Prasanna. 2002. Power-aware resource allocation for independent tasks in heterogeneous real-time systems. In *Proceedings of the 9th IEEE International Conference on Parallel and Distributed Systems (ICPADS'02)*.

Received November 2013; revised July 2014; accepted November 2014