Energy Optimization with Worst-Case Deadline Guarantee for Pipelined Multiprocessor Systems

Gang Chen
TU Munich, Germany
cheng@in.tum.de

Kai Huang
TU Munich, Germany
huangk@in.tum.de

Christian Buckl
fortiss GmbH, Germany
buckl@fortiss.org

Alois Knoll
TU Munich, Germany
knoll@in.tum.de

***Abstract*—Pipelined computing is a promising paradigm for embedded system design. Designing the scheduling policy for a pipelined system is however more involved. In this paper, we study the problem of the energy minimization for coarse-grained pipelined systems under hard real-time constraints and propose a method 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 solve it by quadratic programming. Our approach is scalable w.r.t the number of the pipeline stages. Simulation results using real-life applications as well as commercialized processors are presented to demonstrate the effectiveness of our method.**

I. Introduction

Pipelined computing is a promising paradigm for embedded system design, which can in principle provide high performance and low energy consumption [1]. For instance, a streaming application can be split into a sequence of functional blocks that are computed by a pipeline of processors where clock/power-gating techniques can be applied to achieve energy efficiency.

Designing the scheduling policy for the pipeline stages under the requirements of both energy efficiency and timing guarantee is however non-trivial. In general, energy efficiency and timing guarantee are conflict objectives, i.e., 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 [15], [14], [12] or tackles only soft real-time requirements [6], [1]. In the context of hard real-time systems, seldom work has been published that can handle non-deterministic workloads.

This paper 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 interval domain [7]. The event stream has an end-to-end deadline requirement, i.e., 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 the 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 overall 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 from counting multiple times of the burst of the event stream, which will inevitably over-estimate the needed resource for each pipeline stage and lead to poor energy saving. A motivation example in Section IV will demonstrate this drawback in details. Therefore, more sophisticated method is needed to tackle this problem.

Our idea to solve this problem lies in an inverse use of the known pay-burst-only-once principle [7]. Rather than directly partitioning the end-to-end deadline, we compute for the entire pipeline one service curve which 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 and is solved with standard quadratic programming. For simplicity, we consider power-gating energy minimization and use periodic dynamic power management to reduce the leakage power, i.e., to periodically turn on and off the processors of the pipeline. Note that the basic idea can also be applied to clock-gating energy reduction. With this approach, we can not only guarantee the overall end-to-end deadline requirement but also retrieve the pay-burst-only-once phenomena, resulting in a significant reduction of the energy consumption. In addition, our method is scalable with respect to the number of the pipeline stages. The contributions of this paper are summarized as follows:

- A new method is developed to solve the energy-minimization problem for pipelined multi-processor embedded systems by inversely using the pay-burst-only-once principle.
- We derive a formulation of the minimization problem 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.
- A two-phase heuristic is developed to solve the formulated problem and a formal proof is provided to show the correctness of our approach, i.e., guarantee on the end-to-end deadline requirement.
- We conduct simulation using real-life applications as

978-3-9815370-0-0/DATE13/©2013 EDAA

well as commercialized processors to demonstrate the effectiveness of our method.

The rest of the paper is organized as follows: Section II reviews related work in the literature. Section III presents basic models and the definition of the studied problem. Section IV presents the motivation example and Section V describes the proposed approach. Experimental evaluation is presented in Section VI and Section VII concludes the paper.

II. Related Work

Energy optimization for pipelined multiprocessor systems is an interesting topic where numbers of techniques have been proposed in the literature. For instance, approaches based on control theory [1] and runtime workload prediction [6] are proposed, targeting energy minimization under soft real-time constraints. There are also methods [15], [14], [12] for hard real-time systems. But these methods require precise timing information of task arrivals, e.g., periodic arrivals. However, in practice, this precise timing information of task arrivals might not be known in advance, since arrival time of tasks depends on many nonfunctional factors, e.g., environmental impacts.

There are also many works on hard real-time systems but allowing non-deterministic task arrivals. By using the arrival curve model [7] to abstract task arrivals into time interval domain, techniques based on dynamic frequency scaling [8], [10] and dynamic power management [5], [4] have been recently proposed for uni-processor systems. Nevertheless, how to cope with multiple processors is not yet clear. In this paper, we present an approach to derive energy-efficient scheduling with hard real-time constraints for pipelined multiprocessor systems using the arrival curve model.

III. Models and Problem Definition

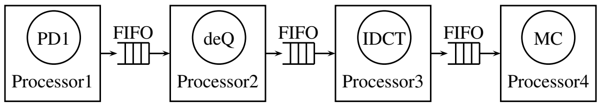
A. Hardware Model

We consider the system with pipeline architecture showed in Fig. 1(a). Each processor in the pipelined system has three power consumption modes, namely active, standby, and sleep modes, as shown in Fig. 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, i.e. $P_a > P_s > P_\sigma$. Moreover, the mode-switch from active (standby) mode to sleep mode will cause energy and time overhead, respectively denoted by $E_{sw,sleep}$ and $t_{sw,sleep}$.

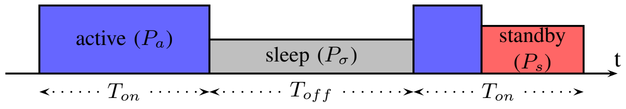
B. Task Model

This paper considers streaming applications that can be split into a sequence of tasks. As shown in Fig. 1(a), a H.263 decoder is represented as four tasks (i.e., PD1, deQ, IDCT, MC) implemented in a pipeline fashion [9]. To model the workload of the application, the concept of arrival curve $\alpha(\Delta) = [\alpha^u(\Delta), \alpha^l(\Delta)]$, originated from Network Calculus [7], is adopted. $\alpha^u(\Delta)$ and $\alpha^l(\Delta)$ provides the upper and lower bounds on the number of arrival events for the stream S in any time interval Δ. 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 an upper and lower bounds on the available resources in any time interval Δ. Note that arrival curves are event-based and service curves are based on amount of computation 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.

$$\bar{\beta}^{Gl}(\Delta) \geq \alpha^u(\Delta - D), \forall \Delta \geq 0 \quad (1)$$



(a) H.263 decoder on pipeline hardware architecture



(b) Power model of a processor

Fig. 1. System model

C. Problem Statement

This paper considers periodic power management [4] that periodically turns on and off a processor. In each period $T = T_{on} + T_{off}$, switch the processor to active (standby) mode for T_{on} time units, following by T_{off} time units in sleep mode, as shown in Fig. 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

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

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.

$$\begin{aligned} P(T_{on}, T_{off}) &\overset{\text{def}}{=} \frac{\frac{L\cdot E_{sw}}{T_{on}+T_{off}} + \frac{L\cdot T_{on}\cdot(P_s-P_\sigma)}{T_{on}+T_{off}}}{L} \\ &= \frac{E_{sw} + T_{on}\cdot(P_s - P_\sigma)}{T_{on}+T_{off}} \end{aligned} \quad (2)$$

Based on (2), the energy minimization problem of a m-stage pipeline can be formulated as minimizing following function:

$$P(\vec{T}_{on}, \vec{T}_{off}) = \sum_{i}^{m} \frac{E_{sw}^i + T_{on}^i \cdot (P_s^i - P_\sigma^i)}{T_{on}^i + T_{off}^i} \quad (3)$$

where $\vec{T}_{on} = [T_{on}^1\ T_{on}^2\ \dots\ T_{on}^m]$ and $\vec{T}_{off} = [T_{off}^1\ T_{off}^2\ \dots\ T_{off}^m]$. 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 Eqn. (3), while guaranteeing that the worst-case end-to-end delay does not exceed D. ■

IV. Motivation Example

This section presents a motivation example, where an event stream passes through a 2-stage pipeline with a deadline requirement D. For simplicity, arrival curves in the leaky-bucket form and service curves in rate-latency form [7] are used. In this representation, an arrival curve is modeled as $\alpha(\Delta) = b + r \cdot \Delta$, where b is the burst and r is the leaky rate. Correspondingly, a service curve is modeled as $\beta(\Delta) = R \cdot (\Delta - T)$, where R is service rate and T is the delay. A graphical illustration of the example is shown in Fig. 2, 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, i.e., $D/2$ for each stage. As shown in Fig. 2, 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 [7], $\alpha'(\Delta) = b + r \cdot T_1 + r \cdot \Delta$. Now again with a deadline requirement $D/2$ for α', we have $T_2 = \frac{D}{2} - \frac{b + r \cdot T_1}{R_1} = 2.5$.

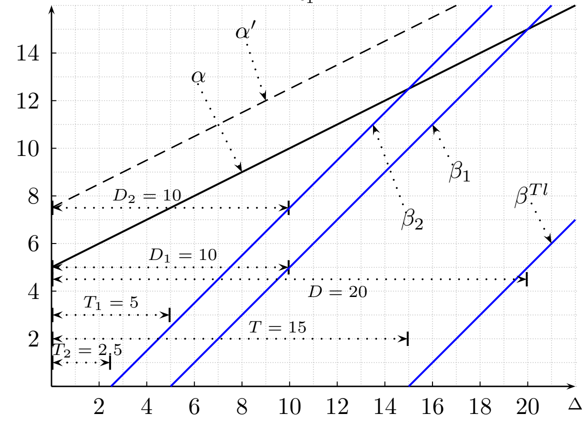


Fig. 2. Motivation example.

Lets 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 too stricter than D. This example indicates that the obtained β_1 and β_2 based on partitioning the end-to-end deadline is 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 the 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 which 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 the cases of pipelined systems with many stages.

On the other hand, one can first derive the total server demand β^{Tl}, in this case $T = 15$. 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.

V. 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 an T_{on}^i and T_{off}^i pair:

$$\beta_i^{Gl}(\Delta) = (T_{on}^i \left\lceil \frac{\Delta - T_{off}^i}{T_{on}^i + T_{off}^i} \right\rceil) \otimes \Delta \quad (4)$$

In addition, to obtain a tightened lower bound of service curve of the entire pipeline, we restrict T_{on}^i as a multiple of the worst case execution time c_i, i.e., $T_{on}^i = n_i\, c_i, n_i \in N^+$.

A. Problem Formulation

Before presenting the formulation, we first state a few bases. By defining $K_i = \frac{T_{on}^i}{T_{on}^i + T_{off}^i}$, we have the following two lemmas.

Lem. 1: $\bar{\beta}_i^{Gl}(\Delta) \geq \frac{K_i}{c_i}(\Delta - T_{off}^i - c_i)$

Proof:

$$\bar{\beta}_i^{Gl}(\Delta) \geq \left\lfloor \frac{T_{on}^i \left\lceil \frac{\Delta - T_{off}^i}{T_{on}^i + T_{off}^i} \right\rceil}{c_i} \right\rfloor \otimes \left\lfloor \frac{\Delta}{c_i} \right\rfloor$$
$$\geq n_i\left(\frac{\Delta - T_{off}^i}{T_{on}^i + T_{off}^i}\right) \otimes \frac{1}{c_i}(\Delta - c_i)$$
$$\geq \frac{K_i}{c_i}(\Delta - T_{off}^i - c_i)$$

■

Lem. 2: $\bigotimes_{i=1}^{m} \bar{\beta}_i^{Gl} \geq \min_{i=1}^{m}(\frac{K_i}{c_i})\big(\Delta - \sum_{i=1}^{m}(T_{off}^i + c_i)\big)$

Proof: It can be directly derived from the definition of min-plus convolution [7] and Lem. 1. ■

With Lem. 2, we state below theorem.

Thm. 1: 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_{on}^i and T_{off}^i pair, the pipelined system satisfies an end-to-end deadline D, if the following condition holds:

$$\min_{i=1}^{m}(\frac{K_i}{c_i})\Big(\Delta - \sum_{i=1}^{m}(T_{off}^i + c_i)\Big) \geq \alpha^u(\Delta - D) \tag{5}$$

Proof: In Lem. 2, the right hand side of inequality is a lower bound of $\bigotimes_{i=1}^{m} \bar{\beta}_i^{Gl}$ which 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 the inequality Eqn. (5) 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 $b_0 = \sum_{i=1}^{m}(T_{off}^i + c_i)$. 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 V-B). 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 minimum bounded-delay functions $bdf_{min}(\Delta, \rho, b)$. Therefore, we can formulate our optimization problem as following:

$$\begin{aligned} \underset{\vec{K}, \vec{T}_{off}}{\text{minimize}} \quad & P(\vec{K}, \vec{T}_{off}) \\ \text{subject to} \quad & \min_{i=1}^{m}(\frac{K_i}{c_i}) \geq \rho \\ & \sum_{i=1}^{m}(T_{off}^i + c_i) \leq b \\ & 0 \leq K_i \leq 1,\ i = 1, \dots, m \\ & T_{off}^i \geq 0,\ i = 1, \dots, m \end{aligned} \tag{6}$$

where $\vec{K} = [K_1 \dots K_n]$. $P(\vec{K}, \vec{T}_{off})$ is obtained as follows by conducting a transformation $K_i = \frac{T_{on}^i}{T_{on}^i + T_{off}^i}$ to (3).

$$P(\vec{K}, \vec{T}_{off}) = \sum_{i}^{m}\left(\frac{E_{sw}^i\,(1 - K_i)}{T_{off}^i} + (P_s^i - P_\sigma^i)\,K_i\right)$$

The advantage of the formulation (6) is two-fold. 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 burst multiple times, on the other hand avoids the costly $\bigotimes$ computation during the optimization. Second, this formulation allows us to use more efficient method to analyze the problem, which will be present in the following sections.

B. Quadratic Programming Transformation

How to solve the minimization problem (6) is not obvious. The constraints b and ρ indeed are not fixed values. 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:

$$\rho = \inf\{\rho : bdf(\Delta, \rho, b) \geq \alpha^u(\Delta - D), \forall \Delta \geq 0\} \tag{7}$$

In this paper, we conduct the optimization by varying b and computing ρ for every possible b. For a fixed b, we can transform (6) into a quadratic programming problem with box constraints(QPB), as stated in the following lemma.

Lem. 3: The minimization problem in (6) can be transformed as the following quadratic programming problem with box constraints:

$$\begin{aligned} \underset{\vec{x} = [x_1\ \dots\ x_m]}{\text{minimize}} \quad & \vec{x}^T Q \vec{x} \\ \text{subject to} \quad & 0 \leq x_i \leq \sqrt{E_{sw}^i(1 - \rho\, c_i)},\ i = 1, \dots, m. \end{aligned} \tag{8}$$

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

Denote $\vec{x}^*$ as the optimal solution for the QPB problem in (8), then the optimization solution for (6) can be obtained with $K_i = 1 - \frac{(x_i^*)^2}{E_{sw}^i}$ and $T_{off}^i = \frac{x_i^*}{\sum_{j=1}^{m} x_j^*}(b - \sum_{j=1}^{m} c_j)$

Proof: With Cauchy-Buniakowski-Schwartz's inequality, we can get that:

$$\sum_{i=1}^{m} T_{off}^i \cdot \sum_{i=1}^{m} \frac{E_{sw}^i(1 - K_i)}{T_{off}^i} \geq \Big(\sum_{i=1}^{m} \sqrt{E_{sw}^i(1 - k_i)}\Big)^2$$

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

$$T_{off}^i = \frac{\sqrt{E_{sw}^i(1 - K_i)}}{\sum_{j=1}^{m} \sqrt{E_{sw}^j(1 - K_j)}}(b - \sum_{j=1}^{m} c_j)$$

Then optimization formulation in (6) can be formulated as:

$$\begin{aligned} \underset{K_1, K_2, \dots, K_m}{\text{minimize}} \quad & \frac{(\sum_{i=1}^{m} \sqrt{E_{sw}^i(1 - K_i)})^2}{b - \sum_{j=1}^{m} c_j} + \sum_{i=1}^{m}(P_s^i - P_\sigma^i)K_i \\ \text{subject to} \quad & \rho\, c_i \leq K_i \leq 1,\ i = 1, \dots, m \end{aligned}$$

By defining $x_i = \sqrt{E_{sw}^i(1 - K_i)}$, formulation (6) can be transformed as the QPB problem in (8). ■

Note that there is a feasible region for b. To guarantee all the resulting $T_{off}^i \geq 0$, the bound-delay b should not be less than $\sum_{i=1}^{m} c_i$. According to (5), the maximum slope ρ of bound-delay function will not exceed $\frac{1}{\max_{i=1}^{m} c_i}$. Correspondingly, we derive the minimum bound-delay function $bdf_{min}(\Delta, \frac{1}{\max_{i=1}^{m} c_i}, b)$. By inverting (7), we can derive the maximum delay b^u by (9), 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:

$$b^u = \sup \{d : bdf(\Delta, \frac{1}{\max_{i=1}^m c_i}, d) \geq \alpha^u(\Delta - D), \forall \Delta \geq 0\}$$

$$b^l = \sum_{i=1}^{m} c_i \tag{9}$$

C. Two-Phase Heuristic

With above information, we can now present the overall algorithm to the energy minimization problem defined in Section III-C. Basically, bounded-delay b is scanned by step ϵ within the range $[b^l, b^h]$. For each b, we first solve the sub-problem (8) with a QPB solver. Then, the obtained solution is repaired to fulfill further constraints (will explain later on). The pseudo code of the algorithm is depicted in Algo. 1.

Algorithm 1 PBOOA

```
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. (7);
3:     obtain $\vec{K}$ and $\vec{T}_{off}$ by solving (8);
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
```

To solve the sub-problem (Line 3 in Algo. 1), we apply existing QPB solver. According to [2], when Q is positive semi-definite, QPB is solvable in polynomial time. Otherwise, QPB can be seen as the non-convex quadratic programming problem which is NP-Hard. Nevertheless, there are approximation schemes [3] that can efficiently solve the non-convex QPB and there are many excellent off-the-shelf software packages [2] available. In this paper, state-of-the-art finite B&B algorithm [2] is applied to solve our QPB problem.

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: compute the loss $Q = \sum_{p_i \in S_1} T^i_{off}$;
4: reassign $Q$ to stage $p$ with maximum power savings;
5: compute $T_{on}$ and the stage set: $S_2 = \{p_i | T^i_{off} \geq t^i_{sw}\}$;
6: for each stage $p \in S_2$ do
7:     if $T_{on} < c$ then
8:         $T'_{on} \leftarrow c\,; T'_{off} \leftarrow T_{off}$;
9:     else
10:        $T'_{on} \leftarrow \lfloor \frac{T_{on}}{c} \rfloor c\,; T'_{off} \leftarrow \frac{T'_{on}}{K} - T'_{on}$;
11:        if $T'_{off} < t_{sw}$ then
12:            $T'_{on} \leftarrow \lceil \frac{T_{on}}{c} \rceil c\,; T'_{off} \leftarrow T_{off}$;
13:        end if
14:    end if
15: end for
```

After obtaining a pair of $\vec{K}$ and $\vec{T}_{off}$, the repair phase (Line 4 in Algo. 1) is conducted to fulfill further constraints. This repair scheme is represented in Algo. 2. First of all, the resulting T^i_{off} of pipeline stage i may be smaller than t^i_{sw}. In the case that $T^i_{off} < t^i_{sw}$, 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 Algo. 2). However, this repair step will lead to the loss of sleep time Q (Line 3 in Algo. 2). We try to assign the loss Q to each stage by $T_{off} = T_{off} + Q$ and compute their power savings by comparing with the previous solution. Then assign Q to the stage with maximum power saving (Line 4 in Algo. 2). Second, the resulting T^i_{on} may not be a multiple of c_i, which is one of our basic requirement. The repair steps are conducted to make T^i_{on} to be a multiple of c_i (Line 5–Line 15 in Algo. 2). It is worth noting that the repair phase we conduct can still guarantee the repaired solution to satisfy the constraints.

VI. Performance Evaluations

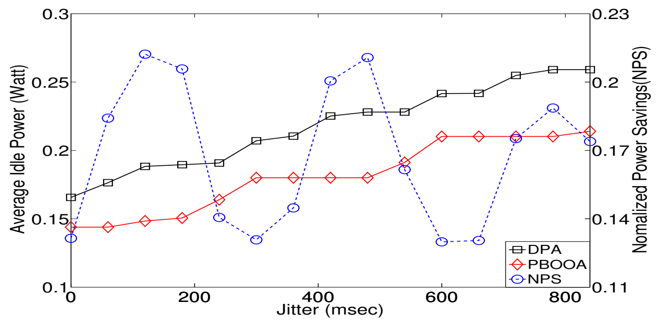
In this section, we demonstrate the effectiveness of our approach. We compare our approach (PBOOA) with deadline partition approach (DPA), where DPA partitions the end-to-end deadline into sub-deadlines for individual pipeline stages and optimizes the overall energy consumption by using the scheme in [4] to minimize the energy consumption of individual pipeline stages. The simulation is implemented in Matlab using RTC-toolbox [13] and the finite B&B algorithm [2] is used to solve QPB. All results are obtained from a 2.83GHz processor with 4GB memory.

A. Simulation Setup

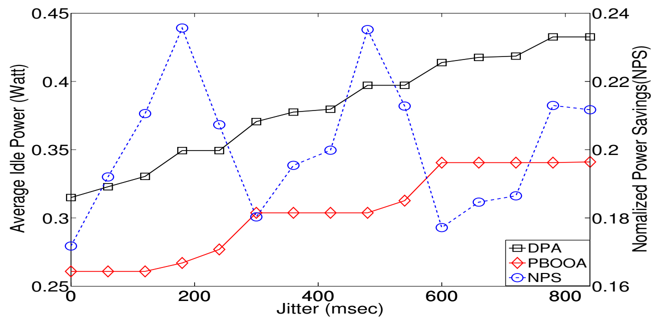
The H.263 decoder shown in Fig. 1(a) is used as the test application. The execution time of each subtask in H.263 decoder application can be found in [9]. The event stream is specified by the PJD model. The activation period of the application is 300 ms with a end-to-end constraint of 600 ms. Regarding to processors for the pipeline architecture, we consider Marvell PXA270 processor and source its power profile from [11]. Standby power P_s and sleep power P_σ are respectively 0.260 Watt and 0.0154 Watt with considering switching time overhead t_{sw} and energy overhead E_{sw} as 0.067 sec and 10.19 mJ, respectively.

B. Simulation Result

We first evaluate how the power consumptions of the two approaches change as the jitter varies. Cases of 2-stage and 3-stage pipeline architectures with homogeneous PXA270 processors are evaluated. We vary the jitter of the stream from 0 to 840 ms. The simulation results are shown in Fig. 3. From figures, we can make the following observations: (1) PBOOA always outperforms DPA for both pipeline architectures. PBOOA on average can achieve 16.8% and 20.09% normalized power savings w.r.t DPA on 2-stage and 3-stage pipeline architectures, respectively; (2) PBOOA can achieve more power savings on 3-stage pipeline than 2-stage pipeline



(a) 2-stage:(PD1,deQ)→$P1$,(IDCT,MC)→$P2$



(b) 3-stage:(PD1,deQ)→$P1$,IDCT→$P2$,MC→$P3$

Fig. 3. Power consumption of PBOOA and DPA on 2-stage and 3-stage homogeneous pipelined system with varying jitter

for different jitter setting. The reason is that DPA on 3-stage pipeline pays burst for more times than 2-stage platform, which leads to PBOOA can achieve more power savings on 3-stage pipeline; (3) The power consumptions of both approaches increase as the jitter increases, since the bigger jitter requires the longer T_{on} to guarantee the worst-case end-to-end deadline.

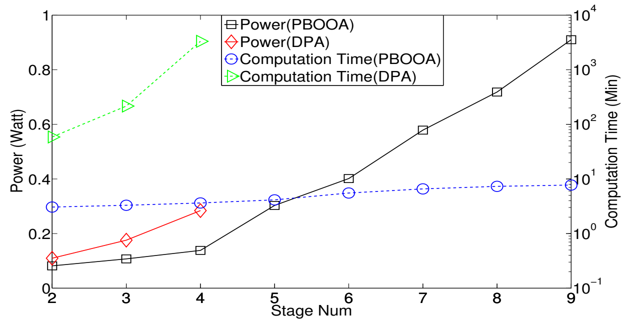


Fig. 4. Computation time and power computation for heterogeneous pipelined system

Second, we demonstrate the scalability of our approach. We test our approach by up to 9-stage heterogeneous pipeline with jitter of 300 ms. Power profile of processors are randomly generated, while the range is set according to PXA270 processor. Fig. 4 shows the power consumption and computation overhead on different pipelines. From this figure, we can have below observations: (1) The DPA approach is time consuming. For the case of 3-stage pipeline, DPA takes almost four hours, which is 65 times longer than PBOOA on the same pipeline. In addition, the 4-stage case needs 15 times more computing time than the 3-stage case. When core number exceeds 4, deadline partition approach fails to provide a result due to expiration of time budget. (2) PBOOA is considerably fast. The 2-stage takes about three minutes. Even with the case of 9-stage pipeline, PBOOA needs 2.5 times more computing time than the 2-stage case.

VII. Conclusion

This paper presents a new approach to minimize the energy consumption of pipelined systems. Our approach can tackle streaming applications with non-deterministic workload arrivals under hard real-time constraints. This approach 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 approach is scalable with respect to the number of pipelined stages. Regarding to future work, it is an interesting problem to combine our approach with the consideration of the mapping of the application.

Acknowledgment

This work has been partly funded by German BMBF projects ECU (grant number: 13N11936) and Car2X (grant number: 13N11933).

References

[1] S. Carta, A. Alimonda, A. Pisano, A. Acquaviva, and L. Benini. A control theoretic approach to energy-efficient pipelined computation in mpsocs. *ACM Transactions on Embedded Computing Systems*, 2007.

[2] J. Chen and S. Burer. Globally solving nonconvex quadratic programming problems via completely positive programming. *Mathematical Programming Computation*, 2012.

[3] M. Fu, Z.-Q. Luo, and Y. Ye. Approximation algorithms for quadratic programming. *Journal of Combinatorial Optimization*, 1998.

[4] K. Huang, L. Santinelli, J.-J. Chen, L. Thiele, and G. Buttazzo. Periodic power management schemes for real-time event streams. In *CDC*, 2009.

[5] K. Huang, L. Santinelli, J.-J. Chen, L. Thiele, and G. Buttazzo. Applying real-time interface and calculus for dynamic power management in hard real-time systems. *Real-Time Systems*, 2011.

[6] H. Javaid, M. Shafique, S. Parameswaran, and J. Henkel. Low-power adaptive pipelined mpsocs for multimedia: An h.264 video encoder case study. In *DAC*, 2011.

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

[8] S. Maxiaguine, A. Chakraborty and L. Thiele. Dvs for buffer-constrained architectures with predictable qos-energy tradeoffs. In *CODES+ISSS*, 2005.

[9] H. Oh and S. Ha. Hardware-software cosynthesis of multi-mode multi-task embedded systems with real-time constraints. In *CODES+ISSS*, 2002.

[10] S. Perathoner, K. Lampka, N. Stoimenov, L. Thiele, and J.-J. Chen. Combining optimistic and pessimistic dvs scheduling: An adaptive scheme and analysis. In *ICCAD*, 2010.

[11] Marvell PXA270. `http://www.marvell.com/application-processors`.

[12] K. Srinivasan and K. S. Chatha. Integer linear programming and heuristic techniques for system-level low power scheduling on multiprocessor architectures under throughput constraints. *Integration,the VLSI Journal*, 2007.

[13] E. Wandeler and L. Thiele. Real-Time Calculus (RTC) Toolbox. `http://www.mpa.ethz.ch/Rtctoolbox`, 2006.

[14] R. Xu, R. Melhem, and D. Mosse. Energy-aware scheduling for streaming applications on chip multiprocessors. In *RTSS*, 2007.

[15] Y. Yu and V. Prasanna. Power-aware resource allocation for independent tasks in heterogeneous real-time systems. In *ICPADS*, 2002.