

# Optimum Buffer Size for Dynamic Voltage Processors

Ali Manzak<sup>1</sup> and Chaitali Chakrabarti<sup>2</sup>

<sup>1</sup> Suleyman Demirel University, Isparta 32260, Turkey,  
manzak@mmf.sdu.edu.tr,

<sup>2</sup> Arizona State University, Tempe, AZ 85287 USA,  
chaitali@asu.edu

**Abstract.** This paper addresses the problem of calculating optimum buffer size for a dynamic voltage scaling processor. We determine the minimum required buffer size giving minimum energy solution for periodic (single, multiple) or aperiodic tasks. The calculations are based on information about data size (maximum, minimum), execution time (best case, worst case), and deadlines.

## 1 Introduction

Dynamic voltage scaling is one of the most effective ways of reducing energy in a system with varying computational loads [1]-[4]. In this paper we consider the problem of optimum buffer size calculation for a dynamic voltage scaling processor. Inclusion of buffer is particularly important in multimedia applications since buffering stabilizes the fluctuating inter-arrival of input frames and allows for a constant processor speed. Since buffer is an additional hardware component that occupies real estate, finding the minimum buffer size that reduces the overall energy consumption is clearly an important problem. This problem is made harder by the fact that future data size and/or the execution time of the task is not known until the task is actually executed.

Use of buffers to *average* workloads was first presented in [5]. The assumption was that the exact workloads of all buffered tasks are known priori. More recently, [6] proposed a rate selection algorithm to distribute the workload under similar assumptions. They consider a practical scenario where the dynamic voltage system provides quantized voltage values instead of continuous values. The proposed algorithm evaluates the workload of all the tasks in the buffer and selects the appropriate quantized rate value. While task configurations with deadlines are not considered in [5] and [6], [7] considers multimedia applications where the task deadlines are relaxed. The scheme in [7] relies on the best case and worst case execution times (instead of exact workloads) to calculate the minimum buffer size to achieve maximum energy savings for single task, multiple sub-task and multi-task configurations. The calculations are based on the fact that maximum energy savings is obtained when the processor is not idle. While this fact is true for soft tasks, it is not valid for tasks with hard deadlines.

If latency can be tolerated (as in multimedia applications), a method to adjust task deadlines for further reducing energy consumption has also been presented.

In this paper we address the problem of finding the optimum (minimum) buffer size that minimizes energy. The minimum energy configuration corresponds to the case when the processor is not idle, the deadlines are not violated and all the task instances are assigned the same voltage. While for tasks with soft timing constraints, the idling condition is sufficient, for tasks with hard timing constraints, it is the task deadlines that determine the buffer size.

The rest of the paper is organized as follows. Section 2 describes the problem and the condition for energy minimization. Section 3 describes how to calculate the buffer sizes for periodic and periodic tasks. Section 4 concludes the paper.

## 2 Preliminaries

### 2.1 System Configuration

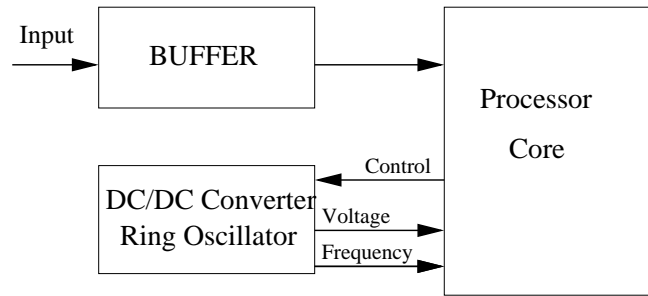


Fig. 1. System architecture

The basic architecture for a dynamic voltage scaling system is shown in Fig 1. The input data is first stored in the buffer and then sent to the processor core. The task scheduling algorithm specifies the operating voltage and frequency based on the task load.

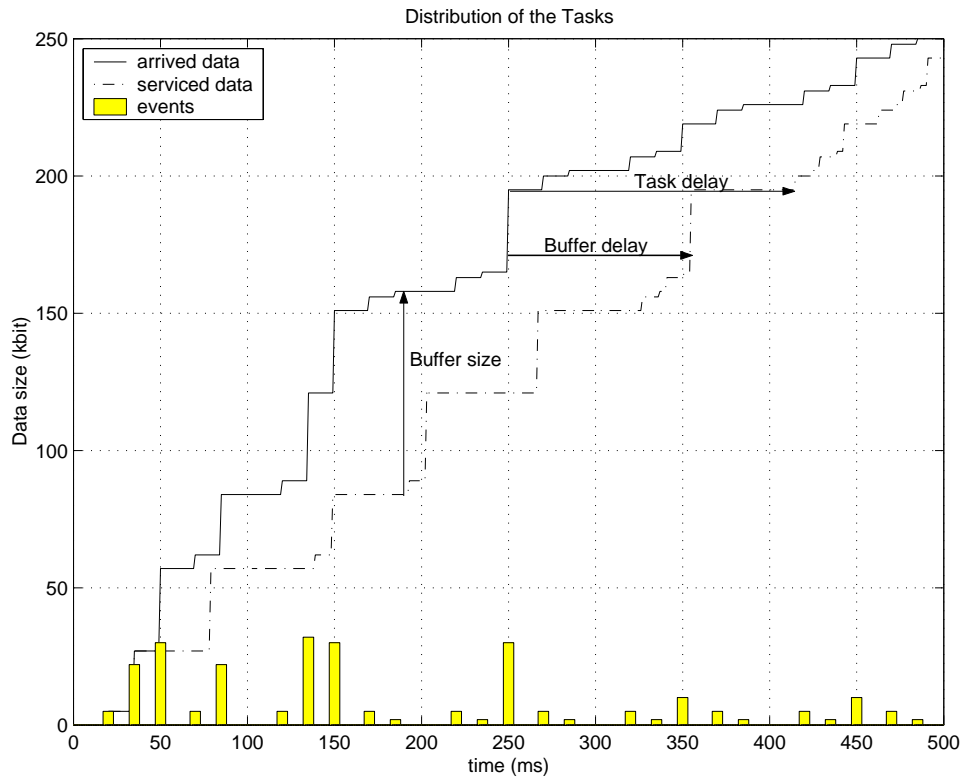
The DC/DC converter has a sensitivity of  $\Delta V$  and can provide voltages in the range  $V_{min}$  to  $V_{max}$ . The oscillator provides a clock frequency that is related to voltage  $V$  by  $1/f = k' C_L \frac{V}{(V - V_t)^2}$ , where  $V_t$  is the threshold voltage,  $C_L$  is the load capacitance and  $k'$  is a device parameter (which depends on the transconductance and the width to length ratio).

### 2.2 Definitions

The following parameters has been used in the rest of the paper. Task  $j$  has arrival time  $a_j$ , finish time  $f_j$ , deadline  $d_j = f_j - a_j$ , service time  $s_j$ , period  $P_j$ ,

worst case execution time of  $WCET_j$ , best case execution time of  $BCET_j$ , average case execution time of  $ACET_j$ , maximum data size of  $maxdata_j$ , minimum data size of  $mindata_j$ , and average data size of  $avedata_j$ . The service time is defined as the execution time + maximum{ service time of the previous task, its arrival time}. In addition,  $Data_j(k)$  is the data size of the  $k^{th}$  instance of task  $j$ .

Next we explain the concept of arrival time, service time, arrival curve, service curve, buffer delay and buffer size with the help of the following simple example.



**Fig. 2.** Arrival and service curve for a simple example

**Simple example:** Figure 2 shows the arrival times and sizes of individual events which are input to the system. For instance, data  $j$  is input at time  $a_j = 200ms$  and is of size 30 kbits. Furthermore, there is a delay associated with the buffer. This is defined as the difference between the time when data is input and when the data gets serviced. In Figure 2, data  $j$  which arrives at  $250ms$  is serviced at time  $s_j = 354ms$ . Thus the buffer delay is  $354 - 250 = 104ms$  for data  $j$ . If it takes 60ms to execute data  $j$ , then the finish time of task  $j$ ,  $f_j \geq 354 + 60 = 414ms$  for feasible assignment ( $f_j$  is the time where execution of task  $j$  is completed).

Figure 2 also describes the arrival and service curve for this example. The distribution of the event data is used to generate the *arrival curve*. Associated with each point in the arrival curve is arrival data,  $a(t)$ , defined as the total data that is input to the system at time  $t$ . Associated with each point in the *service curve* is serviced data,  $s(t)$ , defined as the total data that is sent to the processor for execution at time  $t$ . In the example in Figure 2, at time  $t = 180ms$ ,  $a(t) = 157kbit$  and  $s(t) = 84kbit$ . Thus at time  $t$ , the system has to be able to store data of size  $a(t) - s(t) = 73kbit$ . For feasibility, the buffer size  $B \geq \max\{a(t) - s(t)\}$ , for all  $t$ .

In order to generate the service curve in Figure 2, all the tasks are executed at 2V (reference voltage  $V_{ref} = 3.3V$ ). This voltage assignment corresponds to the minimum energy solution. However this comes at the expense of a buffer size  $B$  of 80kbit and a buffer delay (maximum) of 170ms. If the given constraints (the buffer size and delay) are less than these values, it would not be possible to execute the tasks at 2V and the resulting energy reduction would be smaller.

### 2.3 Problem Statement

In this paper we address the following problem.

- Given a set of tasks,  $\gamma_1, \gamma_2, \dots, \gamma_n$ , each with its arrival time, deadline, period, data size (maximum and minimum data sizes), and execution time (WCET, BCET in cycles), find the optimum (minimum) buffer size such that energy is minimum.

The minimum energy configuration corresponds to the case when the tasks are assigned equal voltages [8] (provided the other constraints are satisfied). Thus energy saving increases as buffer size increases, since tasks have a better chance of being assigned similar voltages. This statement is only valid when the tasks are soft. If the tasks have hard deadlines, there is an optimum buffer size and if the given buffer size is larger than the optimum value, the energy savings do not necessarily increase since deadline constraints still need to be satisfied. If the buffer size is smaller than the optimum value, the processor can be forced to be idle resulting in the task voltages not being optimally assigned.

## 3 Calculating the Optimum Buffer Size

In this section we describe a procedure to calculate the 'optimum' buffer size. The optimum buffer size is defined as the *smallest* buffer required for a minimum energy assignment. Recall that the minimum energy assignment corresponds to the case where all the tasks are being executed using the same voltage. Thus larger the buffer larger is the potential of obtaining the minimum energy assignment (ignoring the effect of additional energy required to access larger buffers). In this section we describe how to calculate the smallest or optimum buffer size for single periodic tasks (section 3.1), multiple periodic tasks (section 3.2) and

aperiodic tasks (section 3.3). We consider two cases – one in which the execution time can be derived from the data size and one in which the execution time is independent of the data size. While the first case is more common, the second case occurs in video applications where the execution time is a function of the frame type or the extent of compression, and does not depend on the datasize.

In the derivation of the optimum buffer size, we consider two factors

1. *The processor should never be idle.* This is because energy savings are larger when the processor operates at lower voltage instead of idling.
2. *Task deadlines should be taken into account.* When deadline constraints are given, tasks should first satisfy the deadline constraints. While large deadline constraints give more opportunities for the tasks to be assigned closer voltages, they also require a larger buffer. Smaller deadline constraints require tasks to be executed earlier, thereby decreasing the number of tasks inside the buffer resulting in smaller buffer size.

**Optimum buffer size:** Minimum energy is obtained when the processor is never idle, and all the task instances are assigned the same voltages. In a system with *soft* deadline constraints such as in multimedia applications, preventing idling is a necessary condition for energy minimization. However, in a system with *hard* deadline constraints, the deadline constraints determine the buffer size. Let  $B_{idle}$  be the minimum buffer size that prevents idling and let  $B_d$  be the minimum buffer size that satisfies the deadline constraints. Choosing buffer size bigger than  $B_d$  is a waste, since all the tasks have to be executed before their deadlines and there won't be any situation where there is more than  $B_d$  data in the buffer. These results conclude that for a system with hard deadline constraints, the optimum buffer size  $B_{opt}=B_d$ .

On the other hand, in a system with soft deadline constraint there is no optimum buffer size. Bigger the buffer, greater chance of tasks in the buffer being assigned equal voltages. Preventing idling can be considered as a major factor for determination of buffer size for a soft task system.

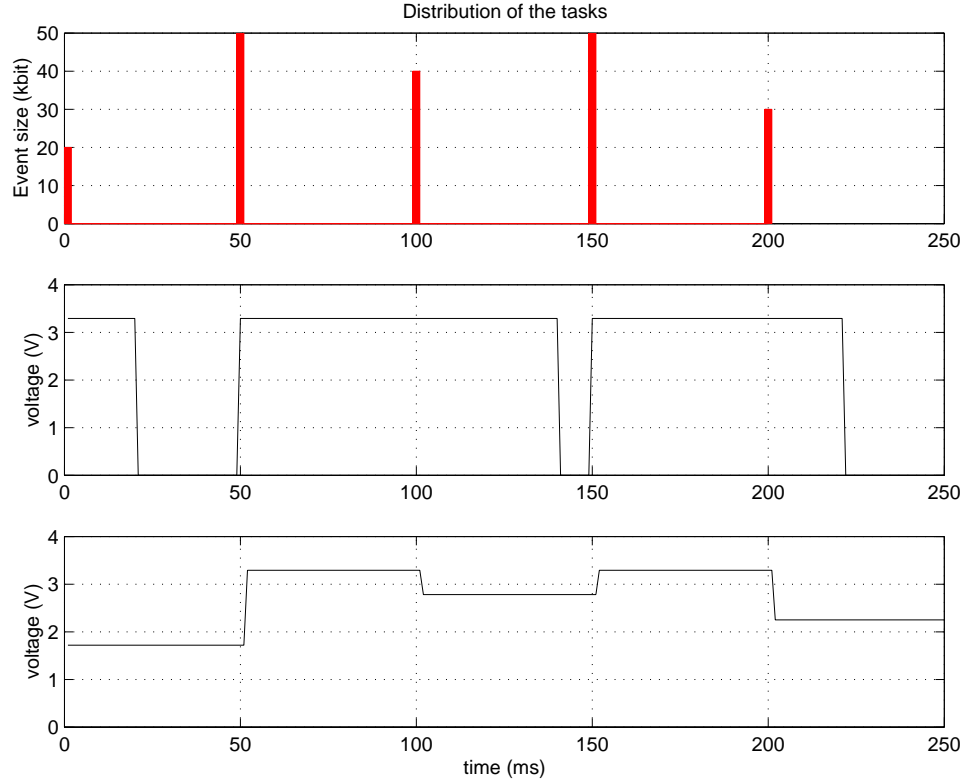
### 3.1 Single periodic task

We consider two cases, one in which the execution time can be derived from the data size and one in which the execution time cannot be derived from the data size.

**Execution time is proportional to data size** We assume that maximum data size (maxdata) and minimum data size (mindata) are known and that the execution time can be determined from the data size.

**Prevent Idling:** Fig.3 describes a single periodic task with period equal to deadline and a period of 50ms (Fig.3).

The data sizes for different instants of the task vary from 20kbit to 50kbit (Fig.3a) and the corresponding execution times vary from 20ms to 50ms. When the processor executes all instances of the task at the reference voltage of 3.3V,



**Fig. 3.** Example for single periodic task assignment where period=deadline: a) task size and arrival times b) task execution profile when processor operates at  $V_{ref} = 3.3V$  c) task execution profile when slack is utilized.

we find that the processor is idle when the task size is small (Fig.3b). The slack can be exploited and the supply voltage of the processor can be reduced such that the processor is never idle as in Fig.3c. This implies that the current task should be completed before the next data arrives in the buffer. The minimum buffer size that prevents idling,  $B_{idle}$ , is then given by  $B_{idle} = 0$ . If, however, we require all tasks to be stored before their execution, the required buffer size is

$$B_{idle} = \max data.$$

Thus, in this example, if  $B = 50kbit$ , the processor is never idle. The energy saving for the assignment in Fig.3c is 32% compared to the case when the processor runs at 3.3V and is idle at times (for the assignment in Fig.3b).

**Deadline constraint:** Now let us consider the case when the deadline of a task is equal to  $nP$ . For the minimum energy assignment, the most pessimistic case occurs when the execution of a task instance takes time  $nP$ . The buffer should be capable of storing the  $(n-1)$  data received during  $n$  periods. Then the required buffer size is  $(n - 1) * \max data$ .

In general, if  $d'$  is the deadline, then the minimum buffer size that minimizes energy satisfying deadline constraints is given by

$$B_d = \text{maxdata} * (\lceil \frac{d}{P} - 1 \rceil)$$

**Optimal buffer size:** The optimal buffer size for hard real-time tasks is

$$B_{opt} = B_d = \text{maxdata} * (\lceil \frac{d}{P} - 1 \rceil) \quad (1)$$

**Execution time is not proportional to data size:** The parameters  $\text{maxdata}$ ,  $P$ , BCET and WCET are known; the execution time of the data varies between BCET and WCET and is not known.

**Prevent Idling:** The processor can be idle when a task completes its execution early (BCET) and there is no task ready in the buffer to be executed. This situation can be easily prevented by arranging the supply voltage of the processor such that even the best case execution time takes  $P$  (period) sec. However, if the task is executed in WCET, then the above voltage arrangement will result in the task finishing computation at  $P * WCET/BCET$  sec. Then the buffer size should be large enough to store data coming in time  $P * WCET/BCET$ . Since only one task arrives every  $P$  sec, the required minimum buffer size is

$$B_{idle} = \text{maxdata} * \lceil \frac{WCET}{BCET} - 1 \rceil$$

A similar expression has been derived in [7].

**Effect of deadline constraint:** Let the voltage of the  $i^{th}$  instance of the task be such that BCET takes  $P$  sec. If the task is executed at  $WCET$ , the task finishes its computation at time  $P(WCET/BCET)$  sec. which should be earlier than its deadline. Thus

$$d \geq P(\frac{WCET}{BCET})$$

If  $d < P(\frac{WCET}{BCET})$ , we can not guarantee that the processor will not be idle and the buffer size is then given by

$$B_d = \text{maxdata} * (\lceil \frac{d}{P} - 1 \rceil)$$

But if  $d = P$  and  $P \geq WCET$ , the required buffer size is  $B = \text{maxdata}$  (theoretically there is no need for buffer). Note that no feasible solution can be guaranteed if  $P < WCET$ .

When deadline constraint is higher than  $P(\frac{WCET}{BCET})$ , the buffer size should be large enough to accept all the tasks received before the deadline. Note that if the current task execution is completed earlier than WCET, extra slack can be used for the remaining instances.

**Optimal buffer size:** The optimal buffer size for a single task with variable execution time is the same as that of a single task with execution time proportional to data size.

$$B_{opt} = B_d = \text{maxdata} * (\lceil \frac{d}{P} - 1 \rceil) \quad (2)$$

However there is a difference during scheduling of these two cases as we will see in Section 4.1. Since the exact execution time of the tasks are not known, the voltages of the tasks are assigned according to WCET. While this guarantees a feasible solution, it overestimates the voltages and decreases the energy savings compared to the case when execution time is proportional to data size.

### 3.2 Multiple periodic tasks

**Execution time is proportional to data size** We assume that  $P$ ,  $maxdata$ , and  $mindata$  values of the tasks are known. The execution time can be derived from the data size.

**Prevent idling:** When there are multiple tasks with different periods and execution times, each task requires a different buffer size to prevent idling. The minimum calculated buffer size ensures that the processor is never idle. So, the task with the smallest buffer requirement determines the required buffer size. Ideally,  $B_{idle} = 0$  since the arrival time of the future data is known. However since the buffer can be overloaded when the different tasks arrive at the same time, the required buffer size to prevent idling is

$$B_{idle} = \sum_{j=1}^n maxdata_j$$

**Effect of deadline constraint:** The maximum data that can be stored in the buffer when the deadlines are satisfied is

$$B_d = \sum_{j=1}^n maxdata_j * (\lceil \frac{d_j}{P_j} \rceil)$$

This is because there can be maximum of  $\lceil \frac{d_j}{P_j} \rceil$  instances in the buffer.

**Optimal buffer size:** The optimal buffer size for a system with hard deadline constraints is given by

$$B_{opt} = B_d = \sum_{j=1}^n maxdata_j * (\lceil \frac{d_j}{P_j} \rceil) \quad (3)$$

**Execution time is not proportional to data size:** We assume that  $P$ ,  $maxdata$ ,  $mindata$ , WCET and BCET values of the tasks are known. The exact execution time is not known.

**Prevent idling:** Let  $P_{min}$  be the minimum period among all the tasks. This implies that the processor can be idle for atmost  $P_{min}$ . Idling can be prevented when the task with BCET takes  $P_{min}$  by operating at a lowered voltage. However, if any task is executed at WCET instead of BCET, the buffer should be large enough to hold all the new coming data during the execution of that task. The following equation gives the total data that can arrive at the buffer if task execution takes WCET (BCET is  $P_{min}$ ).

$$B_{idle} = \sum_{j=1}^n maxdata_j * (\lceil \frac{(WCET/BCET)_{max} P_{min}}{P_j} \rceil)$$

**Effect of deadline constraint:** The minimum buffer size that allows tasks to use all available deadlines is the same as the case where the data sizes are proportional to execution times.

$$B_d = \sum_{j=1}^n maxdata_j * (\lceil \frac{d_j}{P_j} \rceil)$$

**Optimal buffer size:** The optimal buffer size is for a system with hard deadlines is then

$$B_{opt} = B_d = \sum_{j=1}^n maxdata_j * (\lceil \frac{d_j}{P_j} \rceil) \quad (4)$$

### 3.3 Aperiodic tasks:

For aperiodic tasks, if no information on task data size, arrival and deadline time is available, the buffer size can not be calculated. If, on the other hand, maximum data size, arrival and deadline time are known, buffer size can be calculated for minimum energy.

#### Prevent idling when execution time is proportional to data size:

This is similar to the case of single periodic task. If  $maxdata$  is the maximum data size of all tasks, then

$$B_{idle} = maxdata$$

#### Prevent idling when execution time is not proportional to data size:

The maximum execution time that ensures that the processor is not idle for task  $j$  is given by

$$e_j = \left(\frac{WCET}{BCET}\right) * t_r$$

where  $t_r$  ( $= a_{j+1} - a_j$ ) is the time the next data arrives. The buffer should essentially be able to store all the data that arrived during  $e_j$ .

$$B_{idle} = maxdata * [\max\{\text{number of data arrived during } e_j\} - 1].$$

#### Effect of deadline constraint:

Deadline constraint determines the optimal buffer size in both cases (execution time is or is not proportional to data size). Basically, the buffer should be capable of storing all the data that arrives during the execution of task  $j$  in the processor. Since this might take upto  $d_j$ , the deadline of task  $j$ ,  $B_d$  is given by

$$B_{opt} = B_d = maxdata * [\max\{\text{number of data arrived during } d_j\} - 1].$$

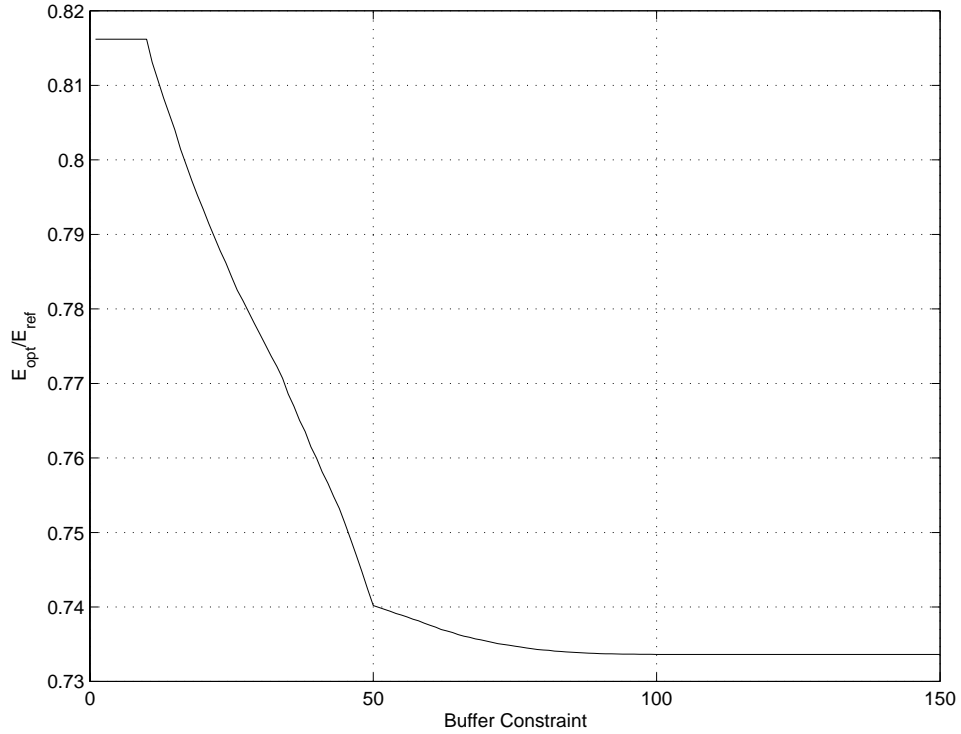
### 3.4 Summary and Results

Table 1 summarizes the buffer sizes for single and multiple periodic tasks and aperiodic tasks.

Tasks	$e$ prop datasize	Soft deadline, $B_{idle}$	Hard deadline, $B_d$
Single periodic	yes	$maxdata$	$maxdata * (\lceil \frac{d}{P} - 1 \rceil)$
	no	$maxdata * \lceil \frac{WCET}{BCET} - 1 \rceil$	
Multiple periodic	yes	$\sum_{j=1}^n maxdata_j$	$\sum_{j=1}^n maxdata_j * (\lceil \frac{d_j}{P_j} \rceil)$
	no	$\sum_{j=1}^n maxdata_j * (\lceil \frac{(WCET/BCET)_{max} P_{min}}{P_j} \rceil)$	
Aperiodic	yes	$maxdata$	$maxdata * [\max\{\# \text{ of data arr. in } d_j\} - 1]$
	no	$maxdata * [\max\{\# \text{ of data arr. in } e_j\} - 1]$	

**Table 1.** Summary of buffer size calculation

To verify the correctness of the table an experiment for single periodic tasks with task execution time  $e$  is proportional to data size has been done. Simulation parameters are chosen as following:  $period = p = 50$ ,  $maxdata = 50$  and  $deadline = 3 * p$ . Task voltages optimally assigned using the optimal offline scheduling algorithm in [8].



**Fig. 4.** Normalized energy versus buffer size constraint curve for 10 task assignment problem for deadline=3P.

We experiment with 1000 different task configurations, where each configuration consists of 10 different tasks with task execution times generated randomly between  $0.1maxdata$  and  $maxdata$ . Normalized energy versus buffer constraint curve is shown in Fig. 4, where  $E_{opt}$  is the optimum energy and  $E_{ref}$  is the energy when tasks are executed at reference voltage (3.3V). Buffer size is changed from 0 to 150 and average optimum normalized energy is calculated for each buffer size. As a result of theoretical calculation the optimum buffer size is 100. Figure 4 also confirms that no further energy savings is possible if buffer size is bigger than 100.

When buffer is less than 10, each task is scheduled in  $p = period$ . No task will be stored in buffer, since the smallest task is 10. Task starts scheduling whenever task arrives and finishes its scheduling before the next task arrives. When buffer size is bigger than 10, we start storing next data in buffer and more savings can be done arranging voltages of tasks optimally. When buffer is bigger than 50, next two tasks can be stored in buffer and more savings can be done. However energy savings do not improve much. As expected no more energy savings is possible when buffer is bigger than 100.

## 4 Conclusions

In this paper we addressed the problem of optimum buffer size selection for a dynamic voltage scaling processor. Our contribution is:

Determination of the optimum buffer size for periodic (single and multiple) and aperiodic tasks.

Our study shows that the energy savings increase with larger buffer sizes and relaxed deadline constraints. This is because in both cases, the chances of the tasks being assigned the same voltage are higher. However this improvement is very limited when certain buffer size has been reached. Also large buffers result in larger access times and larger energy consumption. These factors have not been considered in our analysis.

The task configurations that we have considered assume hard deadlines. However, in a real-time environment, there may be missed deadlines, missed tasks etc. that can be part of the QoS specification. We plan to extend our current minimum energy scheduling algorithms to maximise QoS.

## References

1. Yao, F., Demers A., Shenker, S.: A scheduling model for reduced CPU energy. IEEE Annual Foundations of Computer Science. (1995) 374–382
2. Ishihara T., H. Yasuura, H.: Voltage scheduling problem for dynamically variable voltage processor. Proc. of the Int. Symp. on Low Power Design. (1998)
3. Pering, T., Burd, T., Brodersen, R.: The simulation and evaluation of dynamic voltage scheduling algorithms. Int. Symp. on Low Power Electronics and Design. (1998) 76–81
4. Manzak, A., Chakrabarti, A.: Variable voltage task scheduling for minimizing energy. Proc. of the Int. Symp. on Low Power Design. (2001) 279-282
5. Gutnik, V.: Variable supply voltage for low power DSP. Master's thesis, Massachusetts Institute of Technology. (1996)
6. Chandrasena, L. H., Liebelt, M. J.: A rate selection algorithm for quantized undithered dynamic voltage scaling. Int. Symp. on Low Power Electronics and Design. (2000) 213–215
7. Im, C., Kim, H., Ha, S.: Dynamic voltage scheduling technique for low-power multimedia applications using buffers. Proc. of the Int. Symp. on Low Power Design. (2001) 34-39
8. Manzak, A., Chakrabarti, C.: Voltage Scaling for Energy Minimization with QoS Constraints. Proc of Int. Conf. on Comp. Design. (2001) 438-446