Battery Discharge Aware Energy Feasibility Analysis *

Henrik Lipskoch
University Oldenburg
henrik.lipskoch@informatik.uni-oldenburg.de

Karsten Albers
University Oldenburg
karsten.albers@informatik.uni-oldenburg.de

Frank Slomka
University Oldenburg
frank.slomka@informatik.uni-oldenburg.de

ABSTRACT

It is observed that pulsed discharge currents allow to drain the battery with a higher specific power. Thus they improve the batteries durability and discharge performance. The question is how can the allowed discharge of a battery be modeled. Embedded real-time systems often rely on batteries. For these systems it is necessary to prove both, real-time feasibility and the constraint to not exceed the allowed discharge currents. This paper presents a unified approach for both objectives using the flexible model of event streams, because it allows both to model a complex allowed-discharge curve and the real-time requirements for complex task stimuli. We present the model and the calculation of the allowed and requested discharge curves. Together with the known event-stream based real-time feasibility analysis this allows a unified analysis of both aspects of the system. This work enables the modeling of complex discharge requirements of batteries for real-time systems.

Categories and Subject Descriptors

B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids – verification; C.3 [**Special-Purpose and Application--Based Systems**]: Real-Time and embedded systems; I.6.5 [**Model Development**]: Model methodologies

General Terms

Performance, Reliability, Theory, Verification

Keywords

Event stream, demand bound function, energy bound function, battery modeling, power consumption, spice

*This work is supported by the German Research Foundation (DFG), grant GRK 1076/1 and SL 47/2-1.

CODES+ISSS'06, October 22–25, 2006, Seoul, Korea.

1. INTRODUCTION

Analyzing an embedded system normally turns out to give an answer if the system works within its specification, more precisely the timing behavior is checked. That is if all deadlines are met in every case. A lot of work exists regarding the analysis and methods of how to guarantee hard deadlines, for example [2], [3], [4], [6]. Usually an embedded system is considered to work within a mobile environment, i.e. its energy sources are mostly batteries. To lower the energy consumption is the focus of works regarding techniques like adaptive body biasing [8], [13] or dynamic voltage scaling, [14], [15], [16]. Those works focus on using available slack time for slowing down tasks and thus saving energy. But what is if there exists a point or a period of time while the system can not operate because there is not enough energy to support it or the requested power exceeds the reasonable battery discharge power? Then the system can not guarantee its deadline and this malfunction can not be detected. This question is not regarded by real-time analysis.

This work proposes a method to answer the question of energy feasibility. It uses event streams [7] to model a worst-case energy behavior of the task system and an energy limiting function gathered out of the properties of the battery (SPICE model) describing at what time what amount of energy will be available.

Another objective the proposed method covers is to guarantee an upper limit of energy demand in a certain time for an optimal battery lifespan. There is no other general solution available for this problem. In [12] the authors focus on the distribution of tasks over time to improve the battery lifespan. However the work is based on static scheduling only. Our method can be applied to dynamic scheduled task systems using Earliest Deadline First or priority based scheduling.

For the task model we abstract from the real timing behavior of the systems stimuli by using event streams. These can easily handle tasks with periodic behavior, with jitter or with sporadic triggering, see [1].

The focus of this work lies on how to use the theory of event streams and its methods of analysis to obtain an energy limiting function and an energy demand function, which are necessary for an efficient real-time and energy bound analysis. The complete tool flow is shown in fig. 1. A discharge power function (dcp) is obtained out of the SPICE battery model given by the manufacturer of the battery. With our energy interval transformation (EIT) a discharge energy function (dce) is generated and compared with the systems energy function (W).

The paper is structured as follows. The next section first briefly describes event streams and the idea behind them. The following subsection shows how to apply the idea of event streams to power curves yielding an energy curve. Energy streams, the energy equivalent of event streams, are the focus of the last subsection. We show

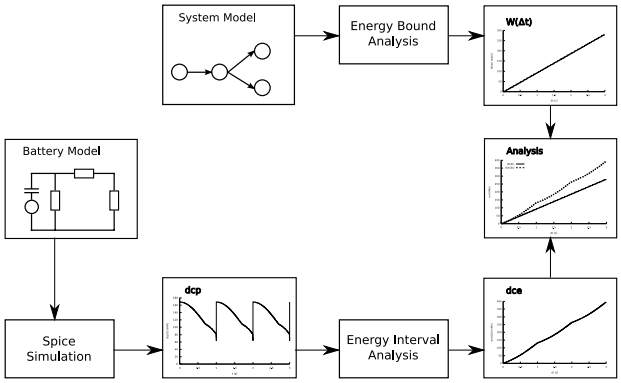

Figure 1: Tool flow

there how to annotate energy properties of tasks on a given event stream to get a stream of energy demands, resulting in an energy bound function. The paper closes with a concluding section.
Throughout the paper we make the notation $\mathbb{N} = 1, 2, 3, 4, \ldots$ and $\mathbb{N}_0 = 0, 1, 2, 3, 4, \ldots$.

2. MODEL

The setup of this work is a task system running on one resource. We call each invocation of a task a job and the point in time t when a task τ is triggered event. We write $e = (t, \tau)$, where τ is said to be the type of e. Together with the assumption that the tasks can be triggered simultaneously, we further consider the system to be scheduled by the Earliest Deadline First scheduling policy, which is optimal for timing [11].
Each task of our task systems is assumed to have a

- relative deadline d, measured from the time when the task is triggered,
- a worst-case execution time c,
- a worst-case execution energy w, which is an upper bound for the real energies needed to execute each possible job of τ,
- and an event stream denoting the worst-case triggering of that task.

The latter one is described in the following subsection. Here, we do not focus on how to gather that information. Instead this information is assumed to be known.

2.1 Event Streams

In [1] the theory of event streams is used to provide real-time analysis for embedded systems. Gresser introduced it [7], motivated by the idea to determine processing bottlenecks in the execution of the task set on a specific resource.
A bottleneck is understood as the shortest time span in which the most tasks have to be completed. Event streams are a way to model this. They list for each time length Δt the maximal number of tasks triggered within that amount of time. Thus the stream abstracts from the start and end points of the intervals on the systems time-line and regards only the distance between start and end of an interval.

Definition 1. Let τ be a task. An event stream $E(\tau)$ is a sequence of real numbers $a_1, a_2, \ldots$, where for each $i \in \mathbb{N}$ a_i denotes the length of a shortest interval in which i number of events of type τ can happen. (See [1] for a more detailed definition)

Albers and Slomka explain how to gather that information for periodic, periodic with jitter, and sporadic triggered tasks [1].

Example 1. Consider the following three tasks. First, let τ_1 be triggered with a period of 100 ms. Then the shortest time span to trigger two tasks is 100 ms. For three it is 200 ms and so on, thus the resulting event stream is $E(\tau_1) : a_1 = 0\,s, a_n = (n-1) \cdot 100\,ms$. Then, let τ_2 be triggered sporadically with a minimal distance between two events of 150 ms. Then the shortest time span to trigger two tasks is 150 ms. For one task more it is 300 ms. The resulting event stream is $E(\tau_2) : a_1 = 0\,s, a_n = (n-1) \cdot 150\,ms$. Finally, let τ_3 be triggered periodically every 60 ms but can be triggered 5 ms before and after its period. Thus the shortest time span to trigger two tasks is 50 ms, which corresponds to one triggering 5 ms after one period and the next triggering 5 ms before the next period. The then earliest task after both can not be triggered shorter than 60 ms later, which is 5 ms before the over-next period, and this corresponds to a time length of 110 ms to trigger 3 tasks. Following this argumentation the resulting event stream is $E(\tau_3) : a_1 = 0\,s, a_2 = 50\,ms, a_n = 50\,ms + (n-2) \cdot 60\,ms$.

To guarantee the deadline of a task one has to consider the current workload of the resource the task runs on. The demand bound function (see [3] and [1]) provides a way to describe this, and the np-hard feasibility test using this function can be approximated in polynomial time [1].
For the workload we now calculate the maximal amount of needed processing time within an interval of length Δt. If we allow the simultaneous triggering of different tasks, which was our assumption, this leads to synchronizing the event streams, and that is to assume all the intervals, out of which we obtained the time lengths for our event streams, have a common start. Thus, the sum of the worst-case execution times of all events in all event streams happening during a time of length Δt, having their deadline within that time, gives us an upper bound of the execution demand for any interval of length Δt. Note, that we only have to process jobs with deadline within the time. Formulated with the notion of definition 1 the demand bound function turns out as follows.

Definition 2. Let $\tau_1, \ldots, \tau_n$ be tasks running on the same resource, each with worst-case execution time c_i and relative deadline d_i, $i = 1, \ldots, n$. And let $E_1, \ldots, E_n$ be their event streams. Define $a_0 := -\infty$. The demand bound function is then

$$D(\Delta t) = \sum_{i=1}^{n} \max\{j \in \mathbb{N}_0 : a_j \in E_i \cup \{a_0\}, a_j \leq \Delta t - d_i\} c_i.$$

(see [1] and [3])

Example 2. Consider the task set of example 1. Let the deadlines be 30 ms for the first, 20 ms for the second, and 10 ms for the third task; let the worst-case execution times for each task be 25 ms, 15 ms, and 5 ms, respectively. Out of these properties, we obtain the demand bound function, which is

$$D(\Delta t) = \left\lfloor \frac{\Delta t + 70\,ms}{100\,ms} \right\rfloor \cdot 25\,ms + \left\lfloor \frac{\Delta t + 130\,ms}{150\,ms} \right\rfloor \cdot 15\,ms + \left(\max\left\{0, \frac{\Delta t - 10\,ms}{|\Delta t - 10\,ms|}\right\} + \left\lfloor \frac{\Delta t}{60\,ms} \right\rfloor \right) \cdot 5\,ms.$$

The next step proceeds with a match of the needed processing time below the available processing time. This is the feasibility test of the real-time system.
Since one can process exactly t s processing time within an interval of t s, if every consumer of processing time is modeled within the task set, and thus the feasibility test results in proving

$$D(\Delta t) \leq \Delta t \quad \forall \Delta t \in \tilde{E}, \tag{1}$$

where $\tilde{E}$ denotes the union of all event streams of tasks on the resource to analyze (see [1]).
Furthermore, there exists a maximal interval length which is an upper bound for the feasibility test, see [3].
Is the test done for every resource in the embedded system, it is certain that the embedded system will work within its specification for timings.

2.2 Battery Modelling

A batteries ability to support with electrical energy depends on the temperature, the discharge current, the history of the battery and other properties. A theoretical and a nominal capacity are distinguished. The former is based on the amount of electro-chemical material the battery consists of. The latter is the product of a nominal discharge current by the time needed to discharge the battery to the cutoff voltage. The nominal capacity is always much less than the theoretical value, see [10].
If the cutoff voltage is reached the cell is said to be discharged. A battery must not be discharged below the cutoff voltage, because it would possibly take damage then. The discharge performance is the energy that can be obtained from the battery within a certain amount of time, see [5].

Definition 3. The discharge power function dcp denotes for each point in time t the highest power to discharge the battery, without violating any constraints of the battery, e.g. heat.

A discharge power function can be generated via a SPICE model of the battery. For analyzing the energy consumption of the task system the knowledge of the available energy for every point in time, like a histogram, is not important. This is because the system is infeasible if there exists one interval of time within which the battery is not able to support the task system with enough power.
Thus we abstract from specific models for electro-chemical cells. Instead we assume a discharge energy function describing for each time length the least available energy within that time length.

Definition 4. Let dcp be a batteries discharge power function. The discharge energy function dce denotes for each amount of time Δt the minimal available energy portion, that is

$$\text{dce}(\Delta t) = \inf_{t \in (0\ s, \infty\ s)} \int_t^{t+\Delta t} \text{dcp}(\theta) d\theta$$

This function models the lower bound of the available battery discharge. Its calculation can turn out to be very complex as shown in figure 2, but it will help us answering the question, if the task system can always run, even in the worst-case, which is when the maximal demand for energy comes along with the lowest available energy. And this is the topic of the third section of this article.
Energy limiting functions can turn out to be very complex to describe, even if they were derived out of approximated discharge performance curves from measuring experiments for a specific battery. To handle them we need an approximation. Our need is for a lower bound of the batteries discharge power. Hence it is suitable to calculate under sums and add them to get the integral. For less approximation errors it is more likely to allow varying intercepts

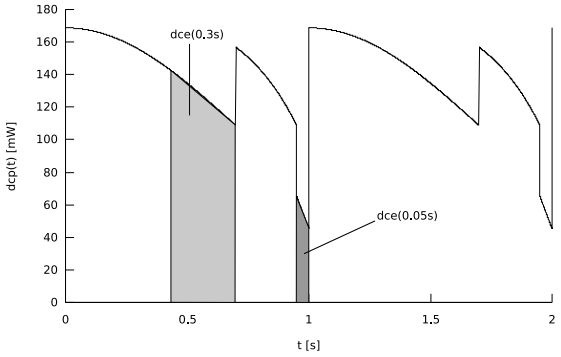

Figure 2: Complex discharge power function

and express the approximation as a sequence of pairs consisting of a slope and an intercept.

Definition 5. Let dce be a discharge energy function. Let $S := \{(m_0, x_0), (m_1, x_1), \dots\}$ be a sequence of pairs of slopes m and intercepts x. An approximating discharge energy function dcb, defined by S, is

$$\text{dcb}(\Delta t) = \sum_{i=0}^{j_{\min}(\Delta t)-1} m_i \cdot x_i + m_{j_{\min}(\Delta t)} \cdot t(\Delta t),$$

where $j_{\min}(\Delta t) := \min\{j \in \mathbb{N} : \sum_{i=0}^{j} x_i \geq \Delta t\}$ and $t(\Delta t) := \Delta t - \sum_{i=0}^{j_{\min}(\Delta t)-1} x_i$,
if for all $\Delta t \geq 0$ the following holds

$$\text{dcb}(\Delta t) \leq \text{dce}(\Delta t).$$

Of course if more is known about the sequence the calculation of the function becomes much easier. The following example shows what changes if equidistant intercepts are assumed.

Example 3. Let $S = \{m_0, m_1, m_2, \dots\}$ with equidistant intercepts of t_S. Then $j_{\min}(\Delta t) = \lceil \frac{\Delta t}{t_S} \rceil$ and the resulting approximating function is

$$\begin{aligned}\text{dcb}(\Delta t) &= \sum_{i=0}^{j_{\min}(\Delta t)-1} m_i \cdot t_S \\ &\quad + \left(\Delta t - \sum_{i=0}^{j_{\min}(\Delta t)-1} t_S\right) \cdot m_{j_{\min}(\Delta t)} \\ &= \sum_{i=0}^{j_{\min}(\Delta t)-1} t_S(m_i - m_{j_{\min}(\Delta t)}) + m_{j_{\min}(\Delta t)} \cdot \Delta t.\end{aligned}$$

In the next example we assume a battery that rests periodically, i.e. the discharge current has to be low. This is common practise to free the surface of the electrodes within the electrolyte from consumed active material. The time is needed to allow diffusion of the material into the electrolyte solution. The discharge current needs not to be equal to zero. In our examples we assume it is high enough to run the processor in an idle state during the time needed for resting.

Example 4. In this example we assume a battery that shall be discharged with a pulse, that is the battery must have resting periods with a duration of 5 ms every 1 s. We assume the available power during the resting period as least as high the power the processor

consumes in idle mode. Within the period the following function shall limit the batteries discharge performance from above.

$$\mathrm{dcp}(t) = \frac{1}{25}\,W \cdot \begin{cases} -1.5\sin(-\frac{\pi}{2} + t(\frac{\pi}{1.4s})) + e & 0s \le t < 0.7s \\ -0.4e^{\frac{t-0.7s}{0.295s}} + e + 0.4 & 0.7s \le t < 0.995s \\ -\frac{100}{1s}t + 99.9 + \frac{e}{6} & 0.995s \le t < 1s \end{cases}$$

The first 0.7 s the discharge performance is limited above with a sine behavior, between the 0.7 s and the 0.995 s it is bounded above with exponential behavior and during the last 5 ms of the period its linearly bounded. In figure 3 we draw the function for the first 3 s.
Now we have to calculate the minimal available discharge performance for every amount of time. For a time span Δt the definition says that we have to calculate the energy for every interval having a length of Δt and afterwords take the infimum over all. However, looking at the function, we see the lowest performance is periodically every one second. And we see the performance increases from the end of a period to the beginning of a period, which means an interval of length 5 ms has the lowest cumulated discharge performance if it ends at the period. The other way around, every interval with the same length ending anywhere within the period must have a higher cumulated discharge performance.
This is the reason why we get an energy limiting curve out of the above discharge specification by integrating the function $-f$ from $t = 1\,s$ down to $t = 1\,s - l$, if $0\,s < l < 1\,s$ is the interval length. For intervals with length $l > 1\,s$ the least energy available will be $n \cdot \int_{1\,s}^{0\,s} -\mathrm{dcp}(t)dt + \int_{1s}^{1\,s+n-l} -\mathrm{dcp}(t)dt$ with $n \le l < (n+1)1\,s$. It is $-\int_{1\,s}^{0\,s} \mathrm{dcp}(t)dt = 0.1319\,W$
To approximate the integral we use the function f for calculating the slopes. An approximation is given by the sequence

$$S := \{(\mathrm{dcp}(0.9999s), 0.005s), (\mathrm{dcp}(0.995s), 0.045s), (\mathrm{dcp}(0.95s), 0.05s), (\mathrm{dcp}(0.90s), 0.05s), \ldots, (\mathrm{dcp}(0.05s), 0.05s)\},$$

which contains 20 segments. Let Δt be the interval length to calculate the energy for. Let Δn be the largest whole numbered time span less than or equal to Δt. Then

$$j_{\min}(\Delta t) = \begin{cases} 0 & \Delta t - n \le 0.005\,s \\ 1 & 0.005\,s < \Delta t - n \le 0.1\,s \\ \lfloor \frac{\Delta t - n}{0.1\,s} \rfloor & 0.1\,s < \Delta t - n \le 1\,s \end{cases}$$

and the approximating function is

$$\mathrm{dcb}(\Delta t) = \Delta n \cdot 0.1319\,W + \sum_{i=0}^{j_{\min}(\Delta t)-1} m_i \cdot x_i + m_{j_{\min}(\Delta t)} \left(\Delta t - n - \sum_{i=0}^{j_{\min}(\Delta t)-1} x_i \right).$$

The mean error over the period from 0 to 1 of this approximation is 1%.

The abstraction also generalizes from certain parameters of a battery like temperature or memory effects and thus allows to consider yet more complex power supplies like batteries that are charged during operation or have rest periods while other batteries support the embedded system.

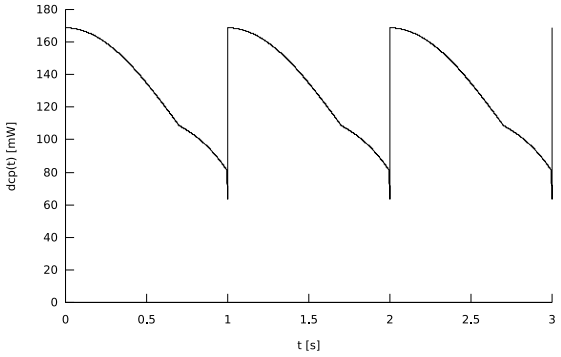

Figure 3: Discharge curve of example 4

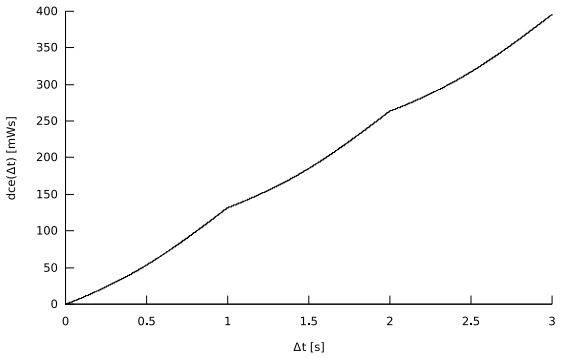

Figure 4: Available energy per interval length for example 4

2.3 Energy Streams

We now use the event stream model to get a worst-case triggering of our task set. This is similar with the real-time analysis. But here, to calculate an upper bound for the energy demand for each interval length, we sum up the worst-case energy portions, instead of summing up the worst-case execution time needed to complete each job.
First of all, we give a proper reason to consider event streams in real-time and energy analysis.

Lemma 1. An event stream represents the worst-case scenario for time and for energy modelling.

PROOF. Let τ be a task with relative deadline d, worst-case execution time c, and worst-case energy w. Let $E = \{a_1, a_2, \ldots\}$ its event stream, let $a_0 := -\infty$. For each time length Δt the amount of triggered tasks to finish within is

$$D := \max\{j \in \mathbb{N}_0 : a_j \in E \cup \{a_0\}, a_j \le \Delta t - d\}.$$

Thus the processing demand for the task τ is $D \cdot c$, and the energy demand $D \cdot w$, respectively.
Clearly the first way to increase both values is by increasing the number of tasks triggered per time unit. And this is assured by the event stream, definition 1. □

If we would have a single-task system we could stop here. For a system with more than one task we again assume that all tasks can be triggered simultaneously. This is superposition and works fine for the real-time analysis (see [1] and [2]).
The demand bound function for energy then can be calculated in an analogous way as in subsection 2.1.

Definition 6. Let $\tau_1, \ldots, \tau_n$ be tasks running on the same resource, each with worst-case execution energy w_i and relative deadline d_i, $i = 1, \ldots, n$. And let $E_1, \ldots, E_n$ be their event streams. Define $a_0 := -\infty$. The demand bound function on energy for the task set is then

$$W(\Delta t) = \sum_{i=1}^{n} \max\{j \in \mathbb{N}_0 : a_j \in E_i \cup \{a_0\} \leq \Delta t - d_i\} \cdot w_i \tag{2}$$

From the view of the task set, we have our upper bound for every interval length, but from the batteries point of view we are not finished yet, because the processor still consumes energy when it becomes idle. To model this case, we introduce the maximal power p_{idle} of the idle task. We can assume that whenever the processor is not idle, the running job consumes more energy per time unit than the idle one. Therefore we first calculate the remainder of each interval length, which is the amount of processing time left after all jobs are done even in their worst case, and multiply it with p_{idle}.
The remainder R is given by the demand bound function

$$R(\Delta t) := \Delta t - D(\Delta t).$$

It is not necessary to apply a check if R is negative, because then the task system is infeasible in the sense of timings, and then no test on energy behavior is needed.
Hence the energy to add, the idle job consumes within an interval of length Δt, is then $p_{\text{idle}} \cdot R(\Delta t)$ in the case all other jobs use their worst-case execution time. The modified energy demand bound function is the result of the following theorem.

Theorem 1. Let $\tau_1, \ldots, \tau_n$ be tasks running on the same resource, each with worst-case execution energy w, worst-case execution time c, and relative deadline d_i, $i = 1, \ldots, n$. And let $E_1, \ldots, E_n$ be their event streams. Let p_{idle} be the power the processor consumes when it becomes idle. Define $a_0 := -\infty$.
The upper bound for the energy consumed within an interval of length Δt is

$$\tilde{W}(\Delta t) = p_{\text{idle}} \cdot \Delta t + \sum_{i=1}^{n} \max\{j \in \mathbb{N}_0 : a_j \in E_i \cup \{a_0\}, a_j \leq \Delta t - d_j\} \cdot (w_i - p_{\text{idle}} c_i)$$

PROOF. Define $j_{\max}(\Delta t, i) := \max\{j \in \mathbb{N}_0 : a_j \in E_i \cup \{a_0\}, a_j \leq \Delta t - d_i\}$, $i = 1, \ldots, n$. The energy consumed by the task system during an interval of length Δt is $W(\Delta t)$. The energy for the idle job is $R(\Delta t) \cdot p_{\text{idle}}$. Thus the total energy consumed during an amount of time Δt is bounded above by

$$\begin{aligned} W(\Delta t) + p_{\text{idle}} R(\Delta t) &= W(\Delta t) + p_{\text{idle}} (\Delta t - D(\Delta t)) \\ &= p_{\text{idle}} \cdot \Delta t \\ &\quad + \sum_{i=1}^{n} j_{\max}(\Delta t, i) w_i - p_{\text{idle}} \sum_{i=1}^{n} j_{\max}(\Delta t, i) c_i \\ &= p_{\text{idle}} \cdot \Delta t + \sum_{i=1}^{n} j_{\max}(\Delta t, i)(w_i - p_{\text{idle}} c_i) \end{aligned}$$

□

Example 5. Lee, Henkel, and Wolf give an example of a GPS-application running on a Palm-Pilot with 7 Tasks with 5 having different implementations [9]. To present a worst-case of energy consumption we have taken those implementations, which afford the highest energy. The properties of the tasks are shown in table 1, where c denotes the worst-case execution time, w denotes the worst-case execution energy, T the period of the task, and d its relative deadline. The periods make it easy to compute the event-streams for every task.

Task	c [ms]	w [mWs]	T [ms]	d [ms]
τ_1	5	0.45	100	100
τ_2	7	0.42	40	40
τ_3	12	1.5	100	100
τ_4	6	1.02	30	30
τ_5	6	0.75	50	50
τ_6	3	0.375	20	20
τ_7	10	0.4	150	150

Table 1: Task set of example 5

$$E(\tau_i) = \{a_j : j \in \mathbb{N},\ a_j = (j-1) \cdot T_i\} \quad \forall i \in \mathbb{N}.$$

3. ENERGY ANALYSIS

With knowledge of a battery energy curve and the systems energy curve, we are able to state and prove the constraint for an energy feasible task system. Let dce be the batteries energy function, let E be the union of the event streams of all tasks within the system and let W be the energy function (Theorem 1) constructed thereof, then the task system is energy feasible if for all intervals of time the following is true:

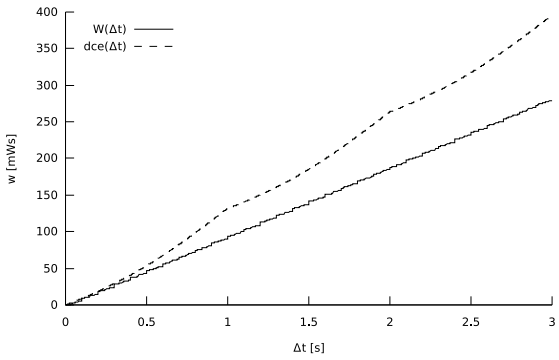

Figure 5: Match of battery and task curve

$$\text{dce}(\Delta t) \geq W(\Delta t) \quad \forall \Delta t \in E. \tag{3}$$

To follow our examples above we have put the energy limiting curve (fig. 4) and the resulting energy demand curve of example 5 together. As the curves do not intersect (see fig. 5), we can state that the task system is feasible for the given battery. There are, however, time durations where both curves lie very close together. As we can see in figure 6, the power of the energy is just enough for intervals of length 100 ms and 200 ms.

If we modify our task set, for example we reduce the deadline of the 7th task to 100 ms, then the curves do intersect at the afore mentioned interval length of 100 ms. This is shown in figure 7. The intersection happens, because the system with the reduced deadline has less time to deliver the required energy and thus the needed energy is added for a smaller interval.

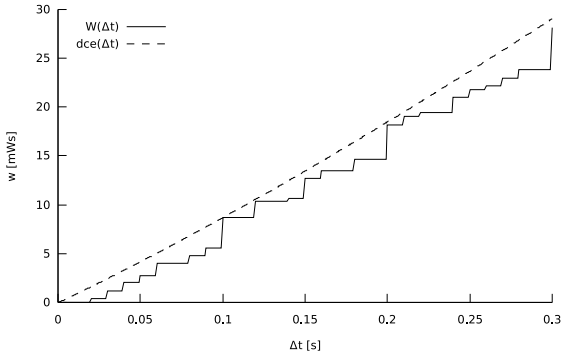

Figure 6: A closer look

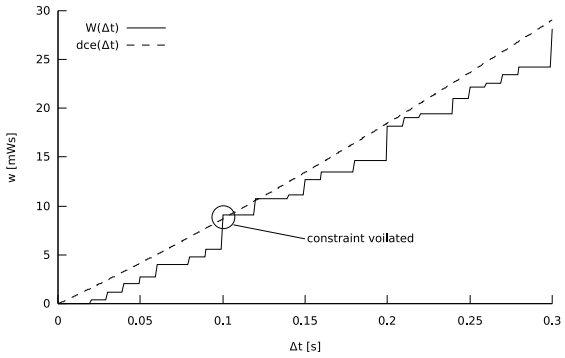

Figure 7: A closer look with the modified task set

4. CONCLUSIONS

In this paper we have proposed a new method for energy feasibility analysis. The method is on system level and takes complex models of batteries into consideration. Furthermore, it can extract the information needed for the analysis out of SPICE models provided by battery manufacturers. We have shown how to use these extracted models to proof that a given load on a processor does not exceed a certain battery discharge curve.

In the previous section we saw that our example is an energy feasible task system, because the energy curve of the task system is below the energy limiting curve of the battery.The other way around we can state that the battery fits our task system. This is important: We can use the limiting curve obtained from the battery's properties to analyze the task system and modify it and it the other way around, that is we can use the energy curve obtained from the event stream of the task system and match a battery on it.

5. REFERENCES

[1] K. Albers and F. Slomka. An event stream driven approximation for the analysis of real-time systems. In *Proceedings of the Euromicro Conference on Real-Time Systems*, 2004.

[2] K. Albers and F. Slomka. Efficient feasibility analysis for real-time systems with EDF-scheduling. In *Proceedings of the Design, Automation and Test in Europe Conference*, 2005.

[3] S. Baruah, D. Chen, S. Gorinsky, and A. Mok. Generalized multiframe tasks. *Real-Time Systems*, 17(1):5–22, 7 1999.

[4] S. Chakraborty, S. Künzli, and L. Thiele. Approximate schedulability analysis. IEEE International Real-Time Systems Symposium, 12 2002.

[5] C.-F. Chiasserini and R. R. Rao. Energy efficient battery management. *IEEE Journal on Selected Areas in Communications*, 19(7), 7 2001.

[6] U. Devi. An improved schedulability test for uniprocessor periodic task systems. In *Proceedings of the 15th Euromicro conference on Real-Time Systems*, pages 23–30, 7 2003.

[7] K. Gresser. An event model for deadline verification of hard real-time systems. In *Proceedings of the Fifth Euromicro Workshop on Real Time Systems*, pages 118–123, 6 1993.

[8] T. Kuroda, T. Fujita, S. Mita, T. Nagamatsu, S. Yoshioka, K. Suzuki, F. Sano, M. Norishima, M. Murota, M. Kako, M. Kinugawa, M. Kakumu, and T. Sakurai. A 0.9-V, 150-MHz, 10-mw, 4 mm, 2-D discrete cosine transform core processor with variable threshold-voltage (VT) scheme. *IEEE Journal of Solid-State Circuits*, 31(11):1770–1779, 11 1996.

[9] T.-M. Lee, J. Henkel, and W. Wolf. Dynamic runtime re-scheduling allowing multiple implementations of a task for platform-based designs. In *Proceedings of the Design, Automation and Test in Europe Conference*, 2002.

[10] D. Linden and T. B. Reddy. *Handbook of Batteries*. McGraw-Hill, New York, 3 edition, 2002.

[11] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1 1973.

[12] J. Luo and N. K. Jha. Battery-aware static scheduling for distributed real-time embedded systems. In *IEEE Design Automation Conference*, 2001.

[13] M. Miyazaki, H. Mizuno, and K. Ishibashi. A delay distribution squeezing scheme with speed-adaptive threshold-voltage CMOS (SA-Vt CMOS) for low voltage LSIs. In *Proceedings of the International Symposium of Low Power Electronics and Design*, 8 1998.

[14] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles. Energy-efficient mapping and scheduling for DVS enabled distributed embedded systems. In *Proceedings of the Design, Automation and Test in Europe Conference*, 2002.

[15] J. Seo, T. Kim, and J. Lee. Optimal intratask dynamic voltage-scaling technique and its practical extensions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(1), 1 2006.

[16] L. Yuan and G. Qu. Analysis of energy reduction on dynamic voltage scaling-enabled systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(12), 12 2005.