EPJ Web of Conferences **70**, 00081 (2014)
DOI: 10.1051/epjconf/20147000081
© Owned by the authors, published by EDP Sciences, 2014

Realization of dynamical electronic systems

Elena Hammari[1,a], Francky Catthoor[2], Leonidas Iasemidis[3], Per Gunnar Kjeldsberg[1], Jos Huisken[4], and Konstantinos Tsakalis[3]

[1]*Dept. of Electronics and Telecommunications, Norwegian University of Science and Technology, 7491 Trondheim, Norway*
[2]*imec, 3001 Heverlee, Belgium and University of Leuven, 3000 Leuven, Belgium*
[3]*Dept. of Bioengineering and Dept. of Electrical Engineering, Arizona State University, Tempe, AZ 85287-9709, USA*
[4]*imec/Holst Center, The Netherlands*

Abstract. This article gives an overview of a methodology for building dynamical electronic systems. As an example a part of a system for epileptic seizure prediction is used, which monitors EEG signals and continuously calculates the largest short-term Lyapunov exponents. In dynamical electronic systems, the cost of exploitation, for instance energy consumption, may vary substantially with the values of input signals. In addition, the functions describing the variations are not known at the time the system is designed. As a result, the architecture of the system must accommodate for the worst case exploitation costs, which rapidly exceed the available resources (for instance battery life) when accumulated over time. The presented system scenario methodology solves these challenges by identifying at design time groups of possible exploitation costs, called system scenarios, and implementing a mechanism to detect system scenarios at run time and reconfigure the system to cost-efficiently accommodate them. During reconfiguration, the optimized system architecture settings for the active system scenario are selected and the total exploitation cost is reduced. When the dynamic behavior is due to input data variables with a large number of possible values, current techniques for bottom-up scenario identification and detection becomes too complex. A new top-down technique, based on polygonal regions, is presented in this paper. The results for the example system indicate that with 10 system scenarios the average energy consumption of the system can be reduced by 28% and brought within 5% of the theoretically best solution.

1 Introduction

Modern electronic systems are becoming increasingly more complex and ubiquitous. We are surrounded by wireless mobile systems that need to react to changing channel and user demands and utilize different communication protocols. The upcoming multimedia systems like audio and video players are designed to adjust their output to different context and quality-of-service requirements. Other typical examples are portable health-care devices for continuous monitoring of a patient's conditions and for seizure warnings from such complex chronic diseases as epilepsy.

[a]e-mail: elena.hammari@iet.ntnu.no

These systems are expected to perform a lot of functionality under tight constraints of available energy resources, timing deadlines and required quality-of-service. Current implementations of such systems are based on worst case conditions and requirements. This results in costly systems consuming much power, hence having short battery lifetimes. Much research is being performed to overcome these challenges. Scenario based design methodologies do this by identifying a set of most common working situations and optimizing the system for each of them separately. In addition, a switching mechanism is developed and added to the system. The mechanism identifies at each time point what situation the system is entering and configures the system to the corresponding preoptimized instantiation. In this way, the system runs efficiently in all encountered situations, allowing several diverse operations to be performed on the same device without violating the design constraints. Such systems, capable of adapting their behavior at run-time under the influence of changing environmental stimuli, are called dynamical electronic systems.

In design of electronic systems quite different methods are used for general purpose systems and for systems performing a (set of) specific dedicated function(s). The latter are referred to as embedded electronic systems and will be the target for the discussion here. They are specialized devices that are typically included or embedded in a larger system, e.g., mobile communication and health care devices, and must satisfy stringent requirements.

The existing scenario based design methodologies for dynamic embedded systems can be divided into multiple classes, depending on the way the common working situations are identified and predicted and by the type of the utilized switching mechanism. In one class of design methodologies the working situations are extracted by analyzing the different usage episodes of the system in terms of user actions and the system operations. For instance, this is how the 'use-case scenarios' are identified in [13] and 'workload scenarios' in [3]. These working situations do not take into consideration the resources required by the system to meet its constraints and may therefore produce suboptimal designs with higher total exploitation costs.

On the other hand, there exists a class of design methodologies that perform direct selection of the system reconfiguration by observing the values of a set of system parameters when the system is run in the field without explicit identification of common working situations in advance. This is the case for most of the methodologies based on the Dynamic Voltage and Frequency Scaling (DVFS) techniques surveyed in [4]. However, these techniques cannot be generalized for costs depending on the parameters in a complex and nonuniform way: the decision making process itself would require excessive resources deteriorating the expected cost savings.

Finally, there is a class of design methodologies that are based on design-time identification of a set of groups of possible exploitation costs, called system scenarios. These methodologies also include a mechanism for detection of system scenarios at run time, and for reconfiguration of the system accordingly. This class of scenario based methodologies can be further subdivided depending on whether control or data variables are used as parameters for scenario identification and detection. This paper provides an overview of a system scenario based design methodology [1]. It has been successfully applied on a number of wireless and multimedia designs [8][9][10][7]. Recently, this methodology has been extended to data-dependent system scenarios [6] enabling its use on new types of systems like an epileptic seizure predictor [5]. Bottom-up multi-valued decision diagram techniques developed for control variable scenario identification and detection then becomes too complex because of the huge set of possible input values. The new extension based on top-down identification of polygonal regions is required to handle data variables, as shown in this paper. Even though this paper only considers design of a system with a single application, previous work has demonstrated system scenario based designs of systems with multiple applications sharing the same hardware device [15][11].

After an introduction to background on the embedded systems design process in Sec. 2, the systems-scenario based design approach is presented in Sec. 3. This is followed by demonstrations of its benefits, using an epileptic seizure predictor as an application example in Sec. 4. We draw our conclusions in Sec. 5.

2 Background: The embedded system design process

The standard embedded system design process includes the following steps [14]. It starts with identification of requirements, both functional - what function the system will perform, inputs and outputs, i.e., its purpose; and nonfunctional - constraints imposed on the system from its environment, other parts in the system or the system user, for example the speed of the system, physical size and weight, power consumption, manufacturing cost. The requirements are in the next step, called specification, converted to a formal description of the system that can be verified when the system is designed. A special formal language called UML (unified modeling language) is typically used. In the next step, the overall architecture of the system is developed, consisting of blocks performing major operations and the data flows between them. At this step it is also decided which parts of the system will be implemented as hardware electronic circuits and which as software programs, and also the hardware and software architecture are decided. Next, the hardware and the software engineers design all the components of the architecture. Finally, the components are put together into a complete system in the integration step where it is also verified against the specification. If the requirements are not fulfilled, redesigning may be needed.

When designing a dynamical embedded system, the designer is faced with a design that has varying exploitation costs depending on the input signals of the system. In addition, the functions describing the variations are not known at the time the system is designed. As a result, the architecture of the system is designed to accommodate for the worst case exploitation costs which, however, rapidly exceed the available resources (for instance battery life) when accumulated over time.

System scenario based design is developed to overcome these difficulties by including in the standard embedded system design process the consideration of the dynamic properties of the system. This is done through the introduction of system scenarios which are defined as follows:

System scenarios group system behaviors that are similar from the multidimensional cost perspective, such as resource requirements, delay and energy consumption, in such a way that the system can be configured to exploit this cost similarity.[1]

The next section describes in details the concept of system scenario based design.

3 System scenario based design methodology

Fig. 1 illustrates the steps in system scenario based design. Given the system software architecture, a graph G of Tasks T and Dependences between them D, where each task consists of a set of variables V and a set of operations on them F. By running a simulation of this software on the model of the selected hardware architecture using a set of all possible inputs I, we identify a set of system behaviors P. Each behavior represents one unique path through the software task graph which is activated by specific values of the variables inside the tasks along the path. We call the set of variables that activate a path V_p and the cost to perform all calculations along the path C_p.

For a dynamic electronic system we select flexible software and hardware architectures that can be tuned with different settings at the time the system is running. Examples of possible settings include different versions of the software code, different compiler optimizations being applied to the software

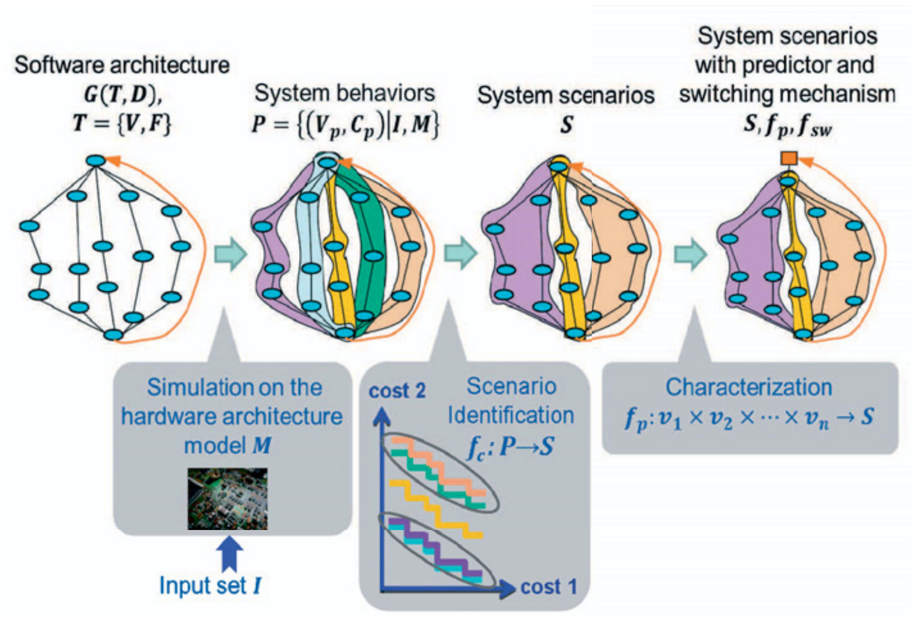

Figure 1. System scenario based design

code, changing the voltage and frequency of processing elements on the run, and using different number and type of available hardware components for a given calculation. Through the exploration of these possibilities for each system behavior the designer finds an optimal system configuration that executes this system behavior with minimal cost C_p. When the system must simultaneously satisfy several design constraints, for example consuming a limited amount of energy while performing its system function within the given timing deadline, each system behavior will have several corresponding costs (time and energy in the example above). It is important that the exploration process is automatic as the number of system behaviors is typically very large.

In fact, the number of system behaviors is so big that it is too costly to switch the system configuration for each system behavior when the system is running. Therefore, in the next step system behaviors with similar costs are identified and divided into a manageable set of groups, which are called system scenarios S. System scenarios are stored in the system and are used to make the decisions about switching. They represent the common working situations for this designed system.

Finally, the system architecture is extended with components f_p and f_{sw} to predict system scenarios from existing variable values and the input signals and to switch the system configurations at run-time. Since each particular system configuration optimizes the use of system resources for the given system scenario, the overall exploitation costs of the system are reduced.

For more complex systems, a calibration component f_c (not shown in Fig. 1) is also added to the system. The calibration allows adjusting scenarios and predictor/switching components at run-time if the system detects that any of these components do not work properly anymore due to the change in the statistical properties of the input signals.

Fig. 2 shows the operation of the implemented system at run-time. Based on received input signals the system predicts what scenario it is about to enter. If it is the same scenario currently in use, it starts directly executing its system function (denoted the Exploitation step). If the system is entering a new

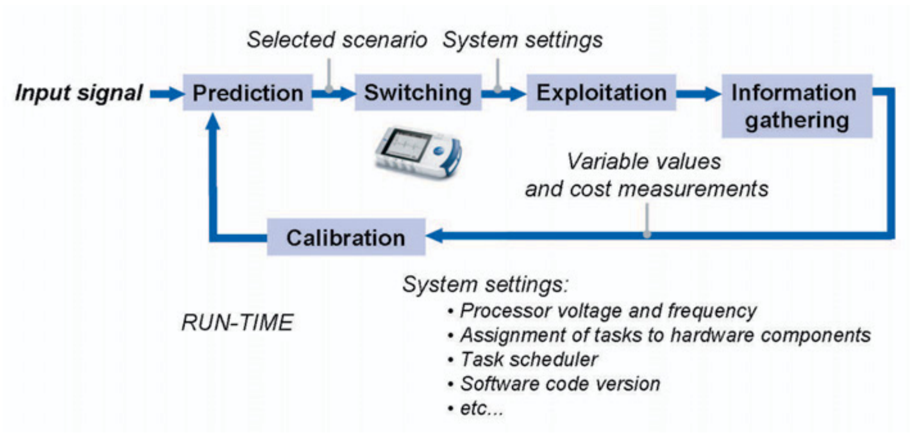

Figure 2. Operation of completely designed system

scenario, for which a more cost-efficient system configuration exists, a system switch is performed to adjust the system settings. After the switch, the system executes its system function. Finally, for more complex systems, information is gathered to evaluate whether the existing scenarios still provide a cost-efficient exploitation. If not, a calibration step is performed. Finally, the whole cycle is repeated from the start.

The work in our group has concentrated on efficient identification of system scenarios and on proving that the additional components for prediction of scenarios and system switching do not introduce unreasonably high overhead costs. The work on the second part is still ongoing, while the next sections will describe our results for the scenario identification part.

3.1 Bottom-up versus top-down system scenario identification

As mentioned in the introduction, scenario identification can be performed bottom-up or top-down. In this section we show that a top-down technique is needed for systems with scenarios based on data input variables. The problem of system scenario identification can be formulated as follows. Given a user specified number of scenarios k and the system behavior set $P = (V_p, C_p)$, where each system behavior is described by the set of variable values V_p and the exploitation cost C_p, divide the set P into k groups, such that system behaviors in one group are similar to each other, both in terms of variable values V_p and the exploitation costs C_p, and dissimilar to system behaviors of other groups. The groups formed in this way are called *system scenarios*.

Fig. 3 shows two different approaches to group system behaviors into system scenarios. On the left-hand side the previous approach [1] is presented. This approach dealt with few system behaviors (small N) and performed identification through bottom-up clustering. It compared system behaviors pairwise with each other and clustered those having a similar cost. Then it compared the pair-clusters pairwise with each other and clustered those with similar costs and so on until the acceptable number of clusters was achieved. The resulting tree, identical to a multi-valued decision diagram (MDD), was also used to find a predictor for the system. For a given input, it was known what system behavior it activates and following the tree up - which scenario it belongs to.

Unfortunately this approach does not work for systems having thousands of different system behaviors (large N), as it would require an unreasonably large predictor that would severely impact the

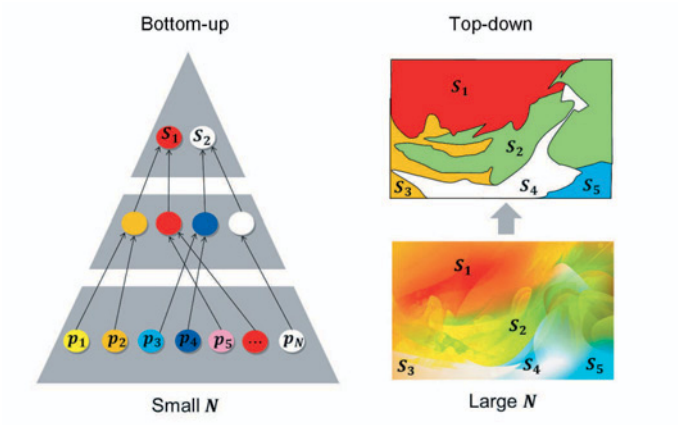

Figure 3. Scenario identification and characterization approaches

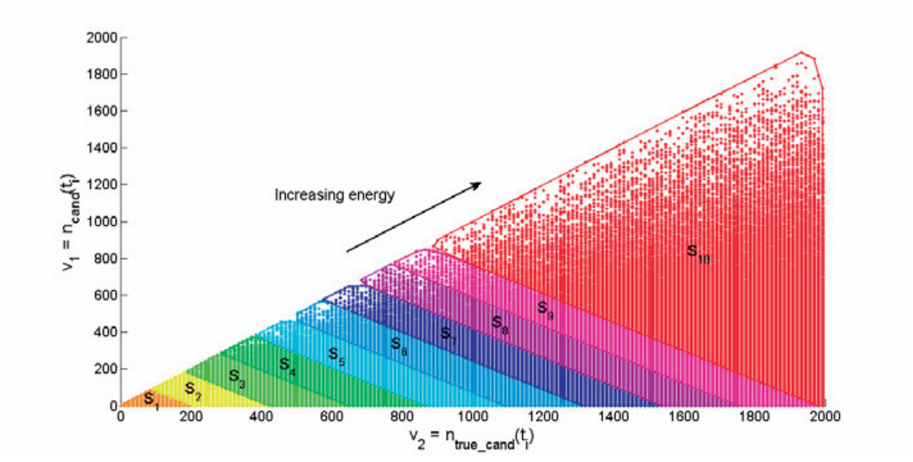

Figure 4. System scenarios in STLmax calculation

exploitation costs of the system. This is the case for many modern systems where the behavior of the system is strongly dependent on the values of input signals, especially if these signals have a continuous nature. We have investigated one such system from the biomedical domain - an epileptic seizure predictor [5]. The system monitors EEG signals from the patient brain and continuously calculates the largest short-term Lyapunov exponents (STLmax). From the STLmax values, the system estimates the probability that the patient will experience a seizure within the next three hours and issues a seizure warning if the probability is high. See Sec. 4 for details about the EEG algorithm. We have observed considerable amount of dynamicity in processing time and energy consumption of the STLmax calculation, and over 150,000 different system behaviors, see Fig. 4. In the figure the system behaviors are plotted in the V_p plane as colored dots. The very large number of system behaviors is caused by deeply nested for-loops with iteration intervals dynamically decided by input data. This is typical for the kind of applications we are targeting. To enable use of the system-scenario based design methodology on such systems, we have developed an alternative system scenario identification approach, which performs identification in a top-down way.

3.2 Top-down system scenario identification

The new top-down approach is presented on the right-hand side of Fig. 3. The bottom figure is the initial projection of system behaviors onto the V_p plane. The color of each dot indicates the exploitation cost of its system behavior. Due to the large number of system behaviors N, the dots are perceived as continuous colors. We first sort all system behaviors by their exploitation cost, such that those having similar cost are placed near each other in the data structure storing the projection. Then we split the behaviors into approximately equally sized groups of similar costs. Finally, we find polygonal borders of the groups in the variable-value plane V_p and represent each group by its highest exploitation cost as shown on the top figure. The obtained groups are the system scenarios that can now be used to design efficient system configurations. In each configuration the system should be able to execute all system behaviors of the corresponding system scenario at the exploitation cost no higher than the cost of the system scenario. More detailed description of our identification technique can be found in [6].

The borders of the scenarios are used to design the system scenario predictor. The predictor checks which polygon the newly started system behavior belongs to and thus finds the upcoming scenario. Checking if a point lies inside a polygon is the classical point location problem from the computational geometry domain [16], and the advantage of using it for prediction instead of MDD is that it operates on/stores only the vertices of polygonal regions, not all possible system behaviors. Balancing of scenarios is also important. It ensures that the probability of occurrence of each scenario is approximately equal and rare system scenarios are avoided.

It should be noted that a scenario based implementation of a dynamic system cannot have the theoretically minimal exploitation cost. This minimum can only be reached if the system switches to an optimal configuration for each behavior. This cannot be done in practice, because the switching would increase the exploitation cost beyond what is reasonable. Thus, the system-scenario based design approach provides good tradeoffs for the dynamical embedded systems. In the next section we show that a limited set of scenarios can come close to the theoretical limit.

The technique described here still has limitations that require further work. The identification process currently does not take the cost of switching between system configurations into account assuming that they happen rarely and have no measurable impact on the exploitation cost of the system. This is, however, a reasonable assumption for applications where system scenarios are beneficial to use. The identification is also restricted to a two-dimensional V_p plane and the corresponding geometric algorithms. It should be noted that generalization to higher dimensions is not trivial, as the complexity of these algorithms increases non-linearly and in some cases efficient algorithms do not exist [16]. This means in practice that the current technique can only be applied to systems where no more than two variables are the main cause of the dynamic behavior. However, our investigation of the STLmax calculation shows that such systems do exist in practice. It is also possible to use worst case values for the less important of three or more variables.

4 Results from epileptic seizure prediction

We have performed system scenario identification for STLmax calculation, a part of the epileptic seizure predictor described in [5], run on the CoolBio embedded processor [2]. Six hours continuous EEG recording from the Arizona State University is used in the experiments. The Short-Term maximum Lyapunov exponent (STLmax) is an estimated maximum Lyapunov exponent from the finite

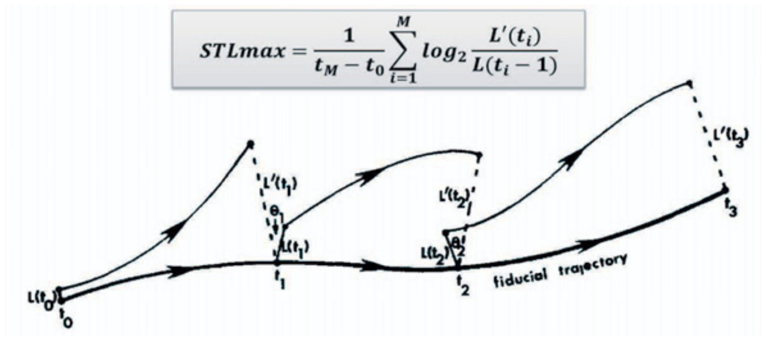

Figure 5. Estimation of STLmax from experimental time series

length experimental time series [12]. It is used in many practical applications dealing with phenomena that are modeled as physical dynamical systems[1], when the system equations are not explicitly known. Lyapunov exponents are defined as the average exponential rates of divergence or convergence of nearby orbits in phase space representations of physical dynamical systems.

In epileptic seizure prediction, the brain is modeled as a chaotic nonlinear system (a type of physical dynamical system) and the epileptic seizures are assumed to result from spontaneous state transitions in this system, followed by a progressive convergence or divergence of STLmax at specific anatomical areas in the brain. STLmax is calculated from the EEG signals by the algorithm presented graphically in Fig. 5. A single pair of nearby orbits is followed in time. One starts with an initial point at t_0 and locates the nearest neighbor to the initial point. The distance between the two points is denoted $L(t_0)$. At a later time t_1, the initial length will have evolved to length $L'(t_1)$. One now looks for a new data point that satisfies two criteria; its distance $L(t_1)$ from the evolved fiducial point is small, and the angular separation between the evolved and replacement elements is small. If an adequate replacement point cannot be found, the existing points are retained. This procedure is repeated until the fiducial trajectory has traversed the EEG data segment under consideration, at which point the maximum short-term Lyapunov exponent is estimated according to the formula in Fig. 5. What is important to notice here is that the algorithm spends most of the time and energy doing the replacement point search and checking whether candidates for the new replacement point satisfy the length and angle criteria. The time needed for STLmax calculation and thus the energy it requires on the embedded processor varies with the input EEG signal as shown in Fig. 6. This is a clear indication that this is a dynamical electronic system.

We applied the system scenario based design methodology described in Sec. 3 with our identification technique in Sec. 3.2 and obtained system scenarios presented in Fig. 4. The points show the different system behaviors that are activated by two internal variables in the algorithm: 1) the number of candidate replacement points n_{cand} at each time step t_i, and 2) the number of true candidates at each time step, n_{true_cand}, i.e., candidates having distance $L > L_{noise}$ to the evolved fiducial point. In the algorithm the nearest search is done concurrently on two regions of the phase space, and the number of candidates in each region changes with time. Also, the number of true candidates changes with the input data. Whenever a seizure is approaching, it becomes harder to find a replacement point and the number of true candidates decreases. Since each true candidate requires an extra calculation step

[1]Beware a terminology confusion here, a physical dynamical system is a completely different concept unrelated to the electronic dynamical system described in this paper.

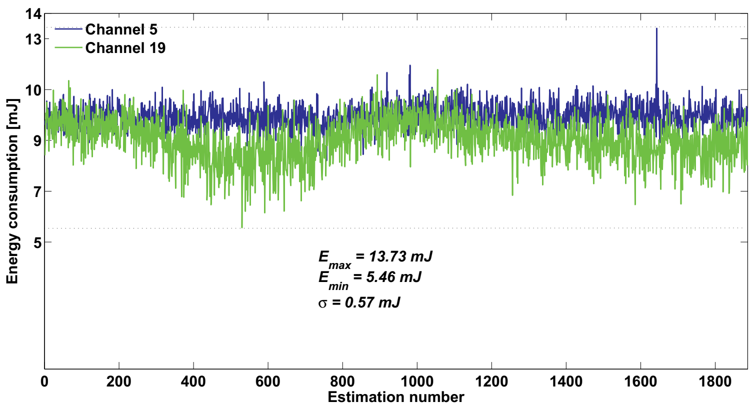

Figure 6. Estimation of STLmax from experimental time series

Table 1. Characteristics of system scenarios for STLmax calculation

Scenario	1	2	3	4	5	6	7	8	9	10
Energy consumption (uJ)	1.3	2.7	4.0	5.3	6.6	8.0	9.3	10.6	12.1	27.3
Probability of occurrence	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10

(checking the angle change), the system behaviors consume different amounts of energy (represented by different colors). The system behaviors have been grouped into 10 scenarios and their borders in the V_p plane have been identified. The energy consumption and the frequency of occurrence for each scenario are presented in Tab. 1. The energy consumption is presented per replacement point search at a given distance/angle scale. The energy consumption for the STLmax calculation on Fig. 6 is the product of the energy numbers in Tab. 1 and the number of searches the algorithm performs for each window of the considered EEG recording (6 hrs 32-channel continuous recording using standard 10-20 electrode setup). It can be seen that the scenarios are well-balanced.

For six hours patient monitoring on the CoolBio DSP processor, the total energy consumption of the system built based on the worst-case conditions, i.e., with only one (10th) scenario, requires 829 J of energy for each EEG channel. The system with 10 system scenarios requires 594 J of energy per channel, which is a 28% improvement. The theoretically best dynamic system, where the number of scenarios equals to the number of system behaviors and the switching costs are assumed to be zero, 567 J of energy per channel are required. Thus, our result is within 5% of this theoretically best solution.

5 Conclusions

We have presented a methodology for design of dynamic embedded electronic systems, where the required exploitation costs vary substantially with the input signals. Instead of using the worst case conditions to design the system, this methodology utilizes system scenarios in order to reduce the final exploitation costs.

Our evaluation has shown that the scenario-based design methodology with top-down identification of scenarios is efficient in reducing the energy in the system for STLmax calculation. Already with few scenarios, an energy reduction of 28 % can be achieved as long as the scenario predictor and switching mechanism do not introduce too high additional costs. This result is within 5% of the theoretical limit for the energy reduction using this methodology.

References

[1] S. V. Gheorghita et al., ACM Transactions on Design Automation of Electronic Systems (TODAES) **14**, (2009), Issue 1, Article 3, 1-45.

[2] M. Ashouei et al., Proceedings of International Solid-State Circuits Conference, (2011) 332-334.

[3] M Otoom and J. M. Paul, International Journal of Parallel Programming **40**, Number 2, (2012) 184-224.

[4] Y. Liu and H. Zhu, Software - Practice & Experience **40**, (2010) pp. 943-964.

[5] L. Iasemidis, Neurosurgery clinics of North America **22**, (2011) pp. 489-506.

[6] E. Hammari et al., Proceedings of The International Conference on Engineering of Reconfigurable Systems and Algorithms, (2012) in print.

[7] D. Stroobandt and K. Bruneel, Proceedings of The International Conference on Engineering of Reconfigurable Systems and Algorithms, (2011) pp. 184-194.

[8] S. Mamagkakis, D. Soudris, and F. Catthoor, Proceedings of Design, Automation, and Test in Europe Conference, (2007), pp. 1-6.

[9] N. R. Miniskar et al., Proceedings of International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, (2009) pp. 48-57.

[10] M. Steine et al., Proceedings of The 15th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, (2011) pp. 31-40.

[11] M. Koedam, S. Stujik and H. Corporaal, Proceedings of the 14th Euromicro Conference on Digital System Design, (2011) pp 708-715

[12] A. Wolf, J. B. Swift, H. L. Swinney and J. A. Vastano,Physica 16D, (1985) pp. 285-317

[13] John M. Caroll, *Scenario-Based Design: Envisioning Work and Technology in System Development* (John Wiley & Sons, Inc., New York, NY, USA, 1995)

[14] W. Wolf, *Computers as Components, Principles of Embedded Computer System Design* (Morgan Kaufmann Publishers, San Francisco, CA, USA, 2005) pp. 10-21.

[15] P. v. Stralen, *PhD thesis, Scenario Based Design Space Exploration*, (University of Amsterdam, Amsterdam, The Netherlands, 2009)

[16] M. J. Atallah and M. Blanton, *Algorithms and theory of computation handbook: special topics and techniques*, (Chapman & Hall/CRC, New York, NY, USA, 2010)