Smart Energy Management and Low-Power Design of Sensor and Actuator Nodes on Algorithmic Level for Self-Powered Sensorial Materials and Robotics

Stefan Bosse[1,3], Thomas Behrmann[2,3]

University of Bremen, Department Computer Science, Workgroup Robotics, Germany[1], BIMAQ Bremen Institute for Metrology, Automation and Quality Science[2], University of Bremen, ISIS Sensorial Materials Scientific Centre, Germany[3]

Abstract

We propose and demonstrate a design methodology for embedded systems satisfying low power requirements suitable for self-powered sensor and actuator nodes. This design methodology focuses on 1. smart energy management at runtime and 2. application-specific System-On-Chip (SoC) design at design time, contributing to low-power systems on both algorithmic and technology level.

Smart energy management is performed spatially at runtime by a behaviour-based or state-action-driven selection from a set of different (implemented) algorithms classified by their demand of computation power, and temporally by varying data processing rates. It can be shown that power/energy consumption of an application-specific SoC design depends strongly on computation complexity.

Signal and control processing is modelled on abstract level using signal flow diagrams. These signal flow graphs are mapped to Petri Nets to enable direct high-level synthesis of digital SoC circuits using a multi-process architecture with the Communicating-Sequential-Process model on execution level. Power analysis using simulation techniques on gate-level provides input for the algorithmic selection during runtime of the system, leading to a closed-loop design flow. Additionally, the signal-flow approach enables power management by varying the signal flow and data processing rates depending on actual energy consumption, estimated energy deposit, and required Quality-of-Service.

1. Introduction and Overview

Today there is an increasing demand for miniaturized smart sensors embedded in sensorial materials and smart actuators. Each sensor and actuator node provides some kind of sensor, electronics, data processing, and communication. With increasing miniaturization and sensor-actuator density, decentralized network and data processing architectures are preferred, but energy supply is still centralized. Using local energy-harvesting technologies, a **decentralized energy supply** can be provided, too. **Energy harvesting**, for example using solar cells, photo diodes sourced by optical fibers, or thermo-electrical sources actually delivers only low electrical power (due to technology or size constraints).

We propose and demonstrate a design methodology for embedded systems satis-

fying low-power requirements suitable for self-powered sensor and actuator nodes. This design methodology focuses on

1. **smart energy management** at runtime using advanced computer science algorithms (artifical intelligence) and
2. application-specific **System-On-Chip** (SoC) design using high-level synthesis at design time. Low-power systems are designed on algorithmic rather than on technological level.

In contrast to various other approaches targeting algorithms and architectures with high computational effort, for example **[5]**, the proposed **smart energy management** is performed spatially at runtime by a selection from a set of different (implemented) algorithms classified by their demand of computation power, and temporally by varying data processing rates. It can be shown that power/energy consumption of an application-specific SoC design strongly depends on computation complexity.
For example, a classical Proportional-Integral-Differential (PID) controller used for feedback position control of an actuator requires basically only the P-part; the I- and D-parts only increase position accuracy and response dynamics which are selectable. Depending on the actual state of the system and the actual and estimated future energy deposit, suitable algorithms can be selected and executed optimizing the Quality-of-Service (QoS) and the trade-off between accuracy and economy.
Signal and control processing is modelled on abstract algorithmic level using signal flow diagrams. These signal flow graphs are mapped to Petri Nets to enable direct high-level synthesis of digital SoC circuits using a multi-process architecture with the Communicating-Sequential-Process model on execution level and the high-level synthesis framework ConPro **[1]**.
Power analysis using simulation techniques on gate-level provides input for the algorithmic selection during runtime of the system leading to a closed-loop design flow. Additionally, the signal-flow approach enables power management by varying the signal flow rate which will be discussed later.

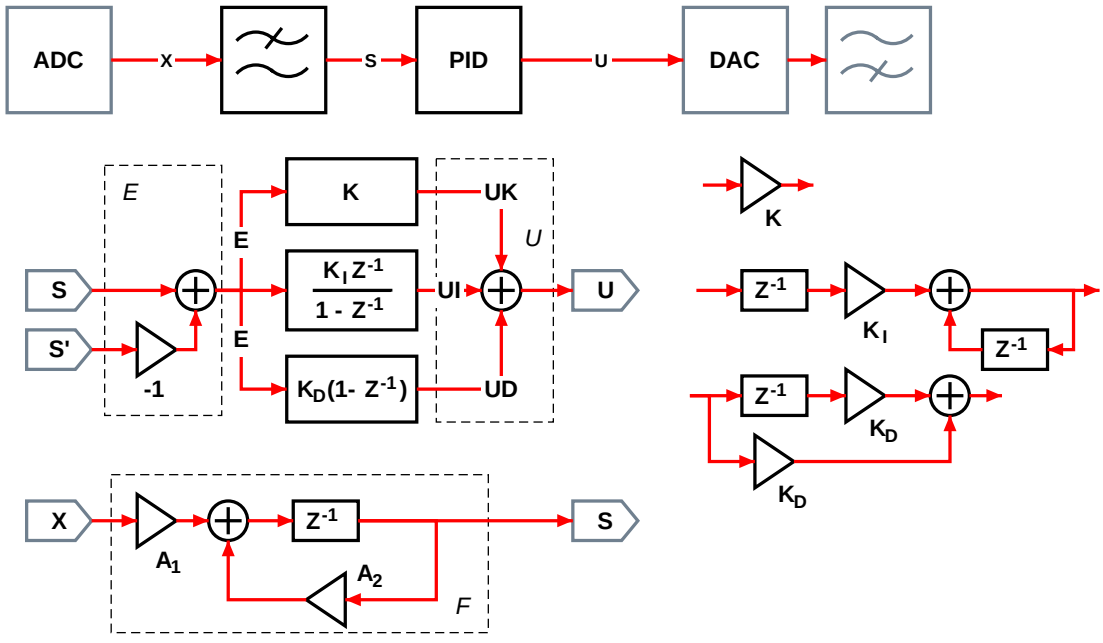
2. Design Flow

The design flow for low-power embedded data processing and control systems should be demonstrated using a concrete example. The system is modelled on an abstract level using signal flow diagrams **[3]**.

Figure **1** shows a composition of a complete feedback-controlled system consisting of sensor signal acquisition (ADC), filtering, an error controller with a proportional, integral, and differential sub-controller **[4]**, and finally a signal generator (DAC) driving an actuator. The controller is used to control the position of an actuator. The control error is defined by the difference of the acquired position signal X(S) and the setable position parameter S'.
This initial specification is used to derive 1. a multi-process programming model, and 2. a hardware model for a SoC design on Register-Transfer level. Furthermore,

the signal flow diagram provides input for energy optimization at synthesis and run-time.

Figure 1. Composition and modelling of a digital control system with signal flow diagrams



The signal flow diagram is first transformed into a S/T Petri Net representation which is shown in figure **2**. Functional blocks are mapped to transitions, and states represent data which is exchanged between those functional blocks. The partitioning of functional blocks to transitions of the net can be performed at different composition and complexity levels. The signal flow diagram from figure **1** was partitioned using complex blocks (merging low-level blocks like multipliers and adders) to reduce communication complexity (and data processing latency).

Sensor data (X) is acquired periodically and passed to the data processing system. A token of the net is equal to a data set of one computation processed by the functional blocks. The functional blocks P,I, and D are placed in concurrent paths of the net.

The Petri Net is then used 1. to derive the communication architecture, and 2. to determine an initial configuration for the communication network. Functional blocks with a feedback path require the injection of initial tokens in the appropiate states (not required in the example).

States of the net are mapped to buffered communication channels and transitions are mapped to concurrently executing processes - each with sequential instruction processing - using the ConPro programming language **[1]**, shown in figure **2**, too.

Forked states indicate concurrency in the Petri Net flow. Exploring concurrency in signal flow diagrams using Petri Nets reduces latency for the computation of one data set. Also pipelining can decrease latency of a data set stream significantly, de-

rived again from the Petri Net representation.

Figure 2. Mapping of the signal flow diagram to a Petri Net and mapping of Petri Net to communication channels and sequential processes using the ConPro programming language.

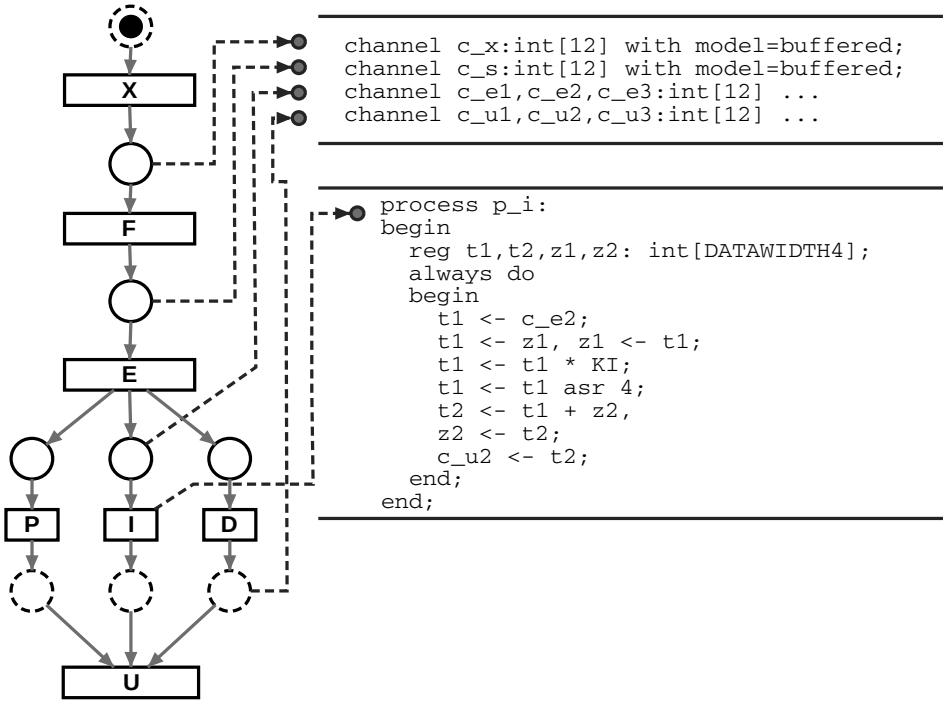
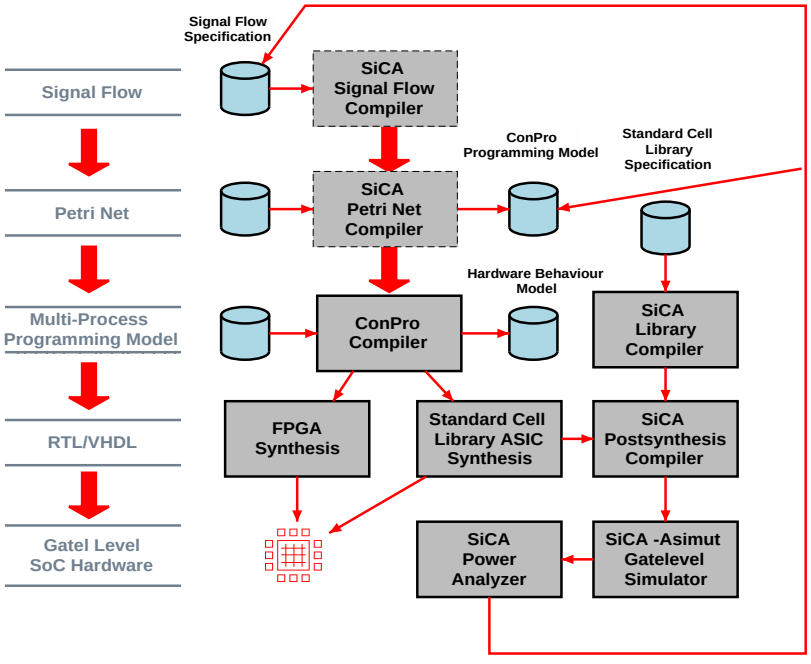


Figure 3. Overall design flow for low power embedded systems using the SiCA and ConPro Synthesis and Analysis framework (Dotted boxes are actually under development). The SiCA design flow uses a graph-based virtual database for advanced data management.



The complete design flow is shown in figure **3**. It is a closed-loop design flow with feedback from power analysis. Results from power analysis are used to make modifications and optimizations on algorithmic level (signal flow and programming model level). The SiCA design flow consists of several analyzer and compiler modules, and uses a graph-based virtual database for advanced data management and inter-module data exchange.

3. Energy Analysis

The derived multi-process programming model was synthesized to a digital logic SoC using high-level synthesis. For simulation, gate-level synthesis was performed with a standard logic cell technology library. The resulting net-list was analyzed with an event-driven simulator, calculating the overall cell activity for each time unit, defined by terms of cell output changes. Synthesis and analysis were performed using the Concurrent Programming (ConPro) compiler **[1]** and the Silicium-Compiler-and-Analyzer framework (SiCA).

The SoC circuit activity correlates strongly with a computation of a new data set (with sensor data input sampled periodically) and the computation complexity, shown in figure **4**.

The logic cell activity of the circuit has strong peaks around the computation of a new output value U. About every 140 clock cycles a new input value X is generated, triggering the calculation of a new output value U.

The first five data sets are computated with an enabled P-part of the controller only. After the fifth computation, the I and D parts were enabled, too. This results in an increase of circuit activity of about 50%.

But power dissipation cannot be estimated directly from this cell activity. Logic cells consist of a network of (paired) transistors. Power dissipation of a CMOS circuit depends proportionally from the transistor switching activity. Simulation results for the controller are shown in figure **5** (using SiCA, too). There is only weak correlation between data processing activity (and computation complexity) and power dissipation due to clocking activity of registers.

Power dissipation can only be estimated from the above circuit cell activity if clock-gated registers are assumed **[2]**. The principle-architecture of gated registers is shown in figure **6**. The clock gating prevents switching activity of the register cell if there is no change of input data. Fine-grained inherent clock gating is a requirement for the proposed low-power design method and enables a strong correlation between computation activity (and hence algorithmic complexity) with the power dissipation of the data processing system.

Results of such a modified control system with clock-gated registers are shown in figure **7**. There is again a significant increase of transistor switching activity of about

30% if the two different computation levels (P, PID) are compared.

Figure 4. Averaged SoC cell activity correlates strongly with computation and signal/data flow. After obtaining the fifth result value U, the I and D computational blocks are switched on.

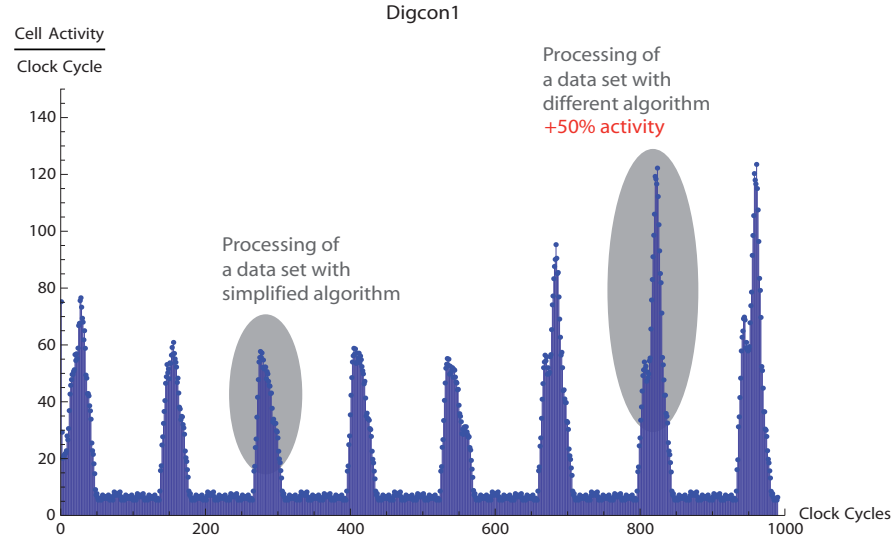


Figure 5. Averaged SoC transistor switching activity of the circuit retrieved from simulation. Power dissipation is proportional to transistor activity.

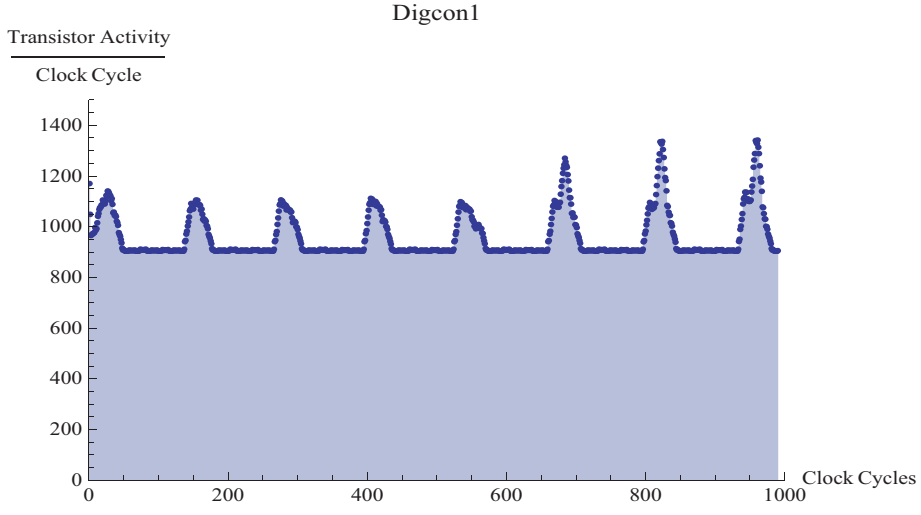


Figure 6. Principle architecture for clock gating of registers required for the implementation of low power embedded systems. The clock gate prevents exaustive transistor switching activity (and hence power dissipation) inside the register cell in the case of no data change (D=Q).

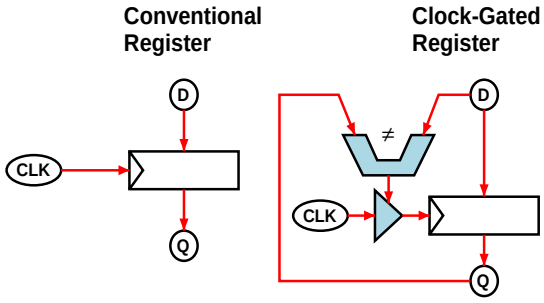
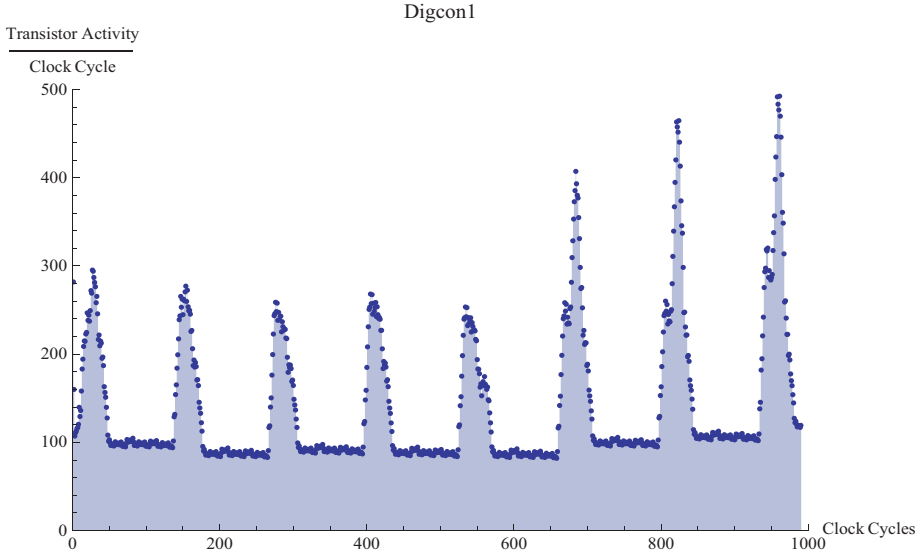


Figure 7. Averaged SoC transistor switching activity of the circuit retrieved from simulation. Using clock-gated registers results again in strong correlation between data processing activity (and computation complexity) and power dissipation.



The power dissipation of the controller can be changed during runtime 1. by selecting different computation levels, and 2. by varying the data processing rate.

4. Smart Energy Management at Runtime

Self-powered systems must deal with a limited amount of energy during runtime. The energy charge in future is uncertain. Smart energy management should handle the conjunction of energy demand and energy conversation.

Methods from artificial intelligence (AI) can be used to manage energy at runtime with dynamic parameter adaption and algorithmic selection. AI methods differ in complexity, thus only few are suitable to be embedded in microchips. Suitable methods are for example constraints nets and decision trees in conjunction with machine

learning approaches.
A simulation of a complex sensor-actuator system implementing the PID controller from figure **1** should demonstrate the benefits of using a decision tree method for dynamic parameter adaption which can be retrieved from machine learning. Parameters to be controlled are data processing rate R and the algorithmic level L (1: only P, 2: P+I+D controller). The controller performs minimization of the position error of the actuator, that means the difference between a desired and a measured (angular) position.
The system is charged with a stochastic energy source and discharged by computation and actuator activity. The power and energy required for computation can be calculated from the results of the power analysis, the power and energy required for actuator activity can be calculated by simplified physical simulation.
The decision tree used in the simulated system is shown in figure **8**. A decision tree is system and environment specific and must be derived for each different system.
Parameters are modelled with a discrete set of values related with a discrete set of cost values, shown in definition **1**.The cost values are used to calculate the overall runtime costs of the system. The goal is to minimize the overal costs and to maximize the energy conversation, but still serving the quality of the service to be provided (in this case the acurate position control of the actuator).
The smart energy algorithms used for the system simulation are shown in algorithm **1**. The `estimate` procedure calculates actual computation and quality-of-service-costs for the system, and the `choose` procedure calculates optimized values for the data processing rate and algorithmic level based on actual system state. The costs are derived from the previous power analysis results. The `costs` function returns a linear interpolated cost value of the particular parameter.
The basic concepts of the simulation are shown in algorithm **2**. The `charge` procedure stores energy in the system from a random source. The `controller` procedure implements the selectable P/PI(D) controller, and finally the `stimuli` procedure simulates the simplified mechanical actor behaviour due to a drive signal calculated by the controller.

Definition 1. System parameter value sets and related cost values.

```
Data processing rate      R={1,5,10,50,100} [milli seconds]
                          R_C={100,50,10,5,1}
Algorithmic level         L={1,2}
                          L_C={100,150}  derived from power analysis
Actuator position error   E={0,5,10,100} [arb. units]
                          E_C={0,250,500,5000}
```

Algorithm 1. Smart energy management algorithms

```
1  VAR
2    level,rate: actual algorithmic level and data processing rate
3    error: actual position control error
4    runtime: passed runtime in time units
```

```
5    energy: energy storage in arb. units
6    cost: quality of service costs
7  PROCEDURE Estimate:
8    for each time unit do
9      delta := Costs(level)*Costs(rate)+Costs(error);
10     energy := energy - delta;
11     quality := Average(runtime)/Average(cost);
12     runtime := runtime + 1;
13     cost := cost + Costs(error);
14     Choose(level,rate);
15 PROCEDURE Choose:
16   Use decision tree to choose optimal {level,rate} values based on
17   averaged quality and actual error
```

Algorithm 2. System simulation algorithms

```
1  VAR
2    pos_act,pos_set: actual and desired actuator position
3  PROCEDURE Stimuli:
4    each 500th time unit do: pos_set := -pos_set;
5    pos_act := pos_act + drive/4;
6    delta := pos_act - pos_set;
7    error := min 100 (abs delta);
8  PROCEDURE Charge:
9    energy := energy + Random(CHARGE_MAX)
10 PROCEDURE Controller:
11   case l is
12     when 1:  (* P-controller *)
13       for each rate time unit do:
14         delta := pos_act-pos_set;
15         S := delta*KD/100;
16         drive := S;
17     when 2:  (* PI controller *)
18       for each rate time unit do:
19         delta := pos_act-pos_set;
20         S := delta*KD/100;
21         Z1' := Z1, Z2' := Z2, Z1 := delta;
22         T := (Z1'*KI)/100+Z2';
23         Z2 := T;
24         drive := S+T;
```

Figure 8. Decision tree used for energy optimization at runtime based on parameters. Input parameters: Error, Quality=Runtime/Error. Output parameters: Data processing rate R={1,5,10,20,100}, Algorithmic level L={P:1,PID:2}

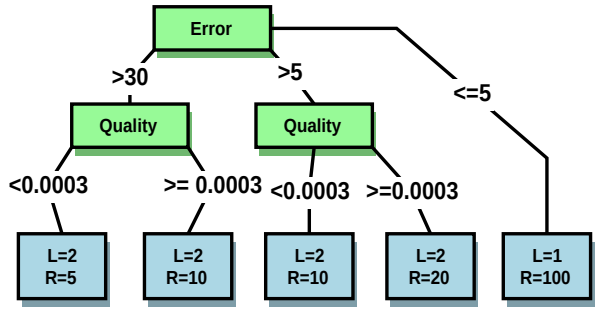
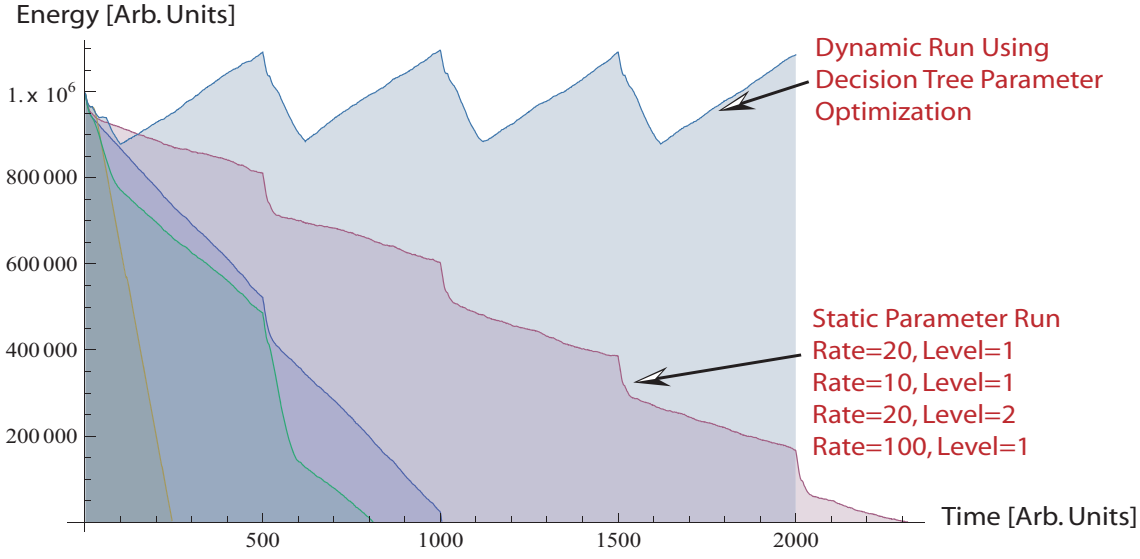


Figure **9** shows simulation results of the complex sensor-actuator system implementing the PID controller. The system runs always out of energy if fixed parameter settings {Rate,Level} are used, regardless of the parameter values. But if dynamic parameter adaption based on the decision tree method and system feedback is used, the system can reach a stable state balancing energy charging and discharging.

Figure 9. System simulation with different runtime behaviours using a decision tree which can be retrieved by machine learning methods. Parameters: Data processing rate={1,5,10,20,100}, Algorithmic level={P:1,PID:2}



5. Summary and Outlook

A new design methodology and architecture was presented for the design of low power embedded systemes preferred for self-powering with energy harvesting technologies. Energy aware design is mainly done on algorithmic level. The data processing of the embedded system is modelled on a high abstract level using signal

flow diagrams. The signal flow diagram is partitioned and mapped to a Petri Net to explore concurrency and to derive a suitable communication architecture. This communication architecture is used in conjunction with a derived concurrent multi-process model, finally mapped to a application specific System-On-Chip design using high-level synthesis, which can be implemented in ASIC and FPGA technologies.
Additionally a smart energy method was demonstrated using decision tree and machine learning methods to improve the runtime behaviour of the implemented system under various conditions. Smart energy management relies on the results of power analysis and optimizes data processing rates and the selection of different algorithms with different complexity, overall optimizing the quality-of-service of the system.
Future work must investigate suitable machine learning methodologies to derive the decision trees.

6. Bibliography

[1] S. Bosse, *Hardware Synthesis of Complex System-on-Chip-Designs for Embedded Systems Using a Behavioural Programming and Multi-Process Model*, Proceedings of the 55th IWK - Internationales Wissenschaftliches Kolloquium, Session C4, Ilmenau, 13 - 17 Sept. 2010

[2] Y. Xia and A.E.A. Almaini., *Differential CMOS edge-triggered flip-flop with clock-gating*, ELECTRONICS LETTERS, Vol. 3rd January2002 Vol. 38 No. IJ

[3] T. Behrmann, C. Zschippig, M. Lemmel, S. Bosse, *Toolbox for Energy Analysis and Simulation of self-powered Sensor Nodes*, Proceedings of the 55th IWK - Internationales Wissenschaftliches Kolloquium, Session A3, Ilmenau, 13 - 17 Sept. 2010

[4] R. Isermann, *Digital Control Systems*, Springer, 1989

[5] D.Y.R. Nagesh, J.V.V Krishna,. S.S Tulasiram, *A real-time architecture for smart energy management*, IEEE Innovative Smart Grid Technologies (ISGT), 2010 Conference