Power Reduction in Embedded Systems Using a Design Methodology Based on Synchronous Finite State Machines

Douglas P.B. Renaux and Fabiana Pöttker

Federal Technological University of Paraná, Department of Electronic Engineering, Curitiba, Brazil
{douglasrenaux,fpottker}@utfpr.edu.br

Abstract. To achieve the highest levels of power reduction, embedded systems must be conceived as low-power devices, since the early stages of the design process. The proposed Model-Based-Development process uses Synchronous Finite State Machines (SFSM) to model the behavior of low-power devices. This methodology is aimed at devices at the lower-end of the complexity spectrum, as long as the device behavior can be modeled as SFSM. The implementation requires a single timer to provide the SFSM clock. The energy reduction is obtained by changing the state of the processor to a low-power state, such as deep-sleep.

The main contribution is the use of a methodology where energy consumption awareness is a concern from the early stages of the design cycle, and not an afterthought to the implementation phase.

Keywords: Synchronous finite state machine, energy consumption, ARM microcontroller, Tankless water heater, Model-Based-Development.

1 Introduction

The reduction of power consumption by embedded devices has become a major concern for designers for three main reasons: (1) Global electrical energy consumption is growing at a rate close to 40% per decade [1] and embedded devices, in the order of 100 billion, contribute to this consumption; (2) many embedded devices, mainly due to mobility requirements, are battery powered (using both rechargeable and non-rechargeable batteries) and lower consumption results in a longer usage time and less pollution to the environment; (3) in a futuristic view, embedded devices will harvest energy from the environment, thus disposing of a very limited energy budget.

Bohn [2] presents a futuristic scenario of Ambiental Intelligence where a large number of embedded devices will be present in our surroundings, including clothes, appliances, and most of the equipment that we use daily. This ubiquitous network of devices, many being of very small dimensions, will benefit from the approach of Energy Harvesting, i.e. the use of the energy sources in our surroundings, including moving objects, vibrating machine parts, changes in temperature, light and other

G. Schirner et al. (Eds.): IESS 2013, IFIP AICT 403, pp. 61–72, 2013.

electromagnetic waves. Since the amount of energy obtainable by Energy Harvesting is usually very small, such devices will be required to have extremely low power consumption levels, in the order of tens to hundreds of μW [3].

To achieve the lowest possible levels of energy consumption, the low-power requirements must be a major concern since the early stages of development. The proposed methodology addresses this concern by the use of Synchronous Finite State Machines (SFSM) as it is a modeling approach that is prone to low-power implementation. A straightforward implementation of putting the processor in deep-sleep mode between clock ticks of the SFSM results in very low consumption levels combined with low resources usage: just a timer is required.

2 Problem Domain

2.1 Approaches for the Reduction of Energy Consumption in Embedded Systems

There are many approaches to achieve energy consumption reduction in Embedded Systems [4]. These can be categorized in four classes:

- **Hardware**: these are hardware design techniques including lower supply voltage, lower processor clock frequencies, low power functional blocks design and low power operating modes.
- **Computer Architecture and Compilers**: concerns the instruction-set design techniques and the appropriate use of the instruction-set by the compilers [5],[6].
- **RTOS**: the kernel can manage the system's energy consumption by changing to low-power modes or reducing the processing power [7],[10].
- **Application**: the applications can change state to low-power requirement modes whenever their processing needs allow so [4].

The methodology proposed in this paper falls into the Application class.

Our approach uses a very straightforward mechanism of changing the state of the processor to sleep-state, where power consumption is minimal and the CPU is not operating. Another approach is used in Chameleon [13], an application level power management system that controls DVFS settings.

2.2 Synchronous Finite State Machines

A Synchronous Finite State Machine performs state transitions only when its clock ticks. Hence, its transitions are synchronous, as opposed to Asynchronous Finite State Machines where transitions may occur as soon as an external event triggers them.

One important difference between SFSM and AFSM is depicted in Fig. 1 where a SFMS is presented. The initial state is A. The occurrence of the event *ev1* enables a transition to state B. This transition occurs only on the next clock tick. Between two consecutive clock ticks, more than one event may occur. This situation is illustrated by the transition from state B to state A. This transition is enabled if both events *ev1* and *ev2* occurred between two consecutive clock ticks.

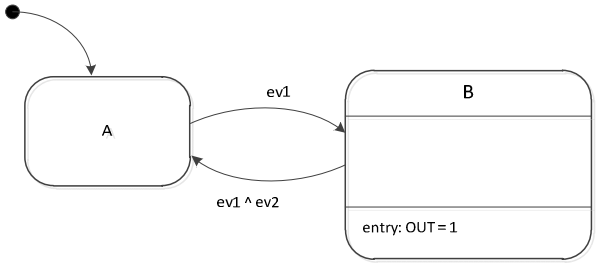

Fig. 1. Example of a State Diagram of a SFSM

3 Related Work

The reduction of energy consumption in embedded systems has been the research subject for several decades already. All the approaches listed in Section 2.1 have been analyzed. Ishihara [8] presents techniques for analysis and measurements in embedded systems, so that energy consumption reduction techniques can be evaluated. Tiwari [9] presents a methodology for embedded software energy consumption analysis.

Venkatachalam [5] presents a survey of the available techniques for energy consumption reduction in embedded systems. These techniques can be applied at several levels: circuit, systems, architecture and applications.

At circuit level, most of the power consumption is due to charging the intrinsic capacitances in digital circuits. Power is a function of the capacitance, clock frequency and the square of the supply voltage. The Dynamic Voltage Scaling (DVS) technique is frequently used to reduce power consumption. It consists of the reduction in the supply voltage and clock frequency whenever the system does not require its full processing speed.

Other circuit level techniques comprise the reduction of transistor sizes in the IC fabrication process, thus, diminishing the intrinsic capacitances, as well as logic gates restructuring to reduce the amount of switching needed. Concerning the buses, some of the applied techniques are the reduction of switching frequency, crosstalk, and signal amplitude, as well as bus segmentation and bus precharging.

At compiler level, energy consumption reduction is obtained by code optimization (less instruction to produce the same logical result) and by reducing the number of memory accesses.

At the applications level, the application and its runtime environment (e.g. RTOS) exchange information concerning opportunities for energy reduction, such as availability of resources and processing power requirements.

The proposed methodology is concerned with design techniques at the applications level. Future improvements in the proposed methodology will include the interactions with the RTOS and the required support to be implemented in the RTOS.

4 Proposed Methodology

The proposed methodology is applicable at the applications level (Section 2.1), therefore, it follows the usual application software development process. In each phase, decisions are taken so that the system can be modeled by SFSM and implemented as such.

1. **Operational Concept**: the product must be conceived to be prone to a SFSM modeling approach, as such, it must deal with discrete time events and actions. Inputs may represent continuous values, but they will be sampled synchronously to the SFSM clock. Outputs will be generated on every SFSM clock period, hence, with delays when compared to a system that operates continuously.
2. **Software Requirements**: the requirements must consider the discrete time operation of the system, hence, delays in input signal detection and output signal generation should be allowed. The larger the allowed delays, the higher the possibilities for reduction of energy consumption. At this phase, if functional and temporal requirements are too stringent then the solution space is reduced, as well as the attainable low-power requirements.
3. **Software Architecture**: The current version of our proposed methodology concerns lower-complexity devices that can be modeled as a single SFSM or as a collection of SFSM. If several SFSM are used, from an energy consumption point of view, it is preferable if all the SFSM clocks have the same period or are multiples of the same master period, thereby reducing the number of times that the processor has to switch from deep-sleep mode to operating mode.
   The selection of the SFSM clock period is of significance both for the reduction of consumption and also for the effect on the control algorithm of the embedded device. In the remainder of this section, several considerations regarding the determination of the SFSM clock period are presented.
4. **Software Design and Implementation**: to improve flexibility, reusability, and portability, the control algorithms design should be parameterized with respect to the SFSM clock periods such that changes to the clock period do not required code modifications except for a possible change in a defined constant.
   For the implementation either a hardware timer or an RTOS timer is required to tick at the end of each SFSM clock period. On each tick the processor is awakened from the deep-sleep mode and executes a simple procedure: (1) reading inputs, (2) identifying if a transition is enabled and firing it, (3) generating new outputs, and (4) returning to deep-sleep mode.
5. **Software Testing**: special care must be taken in the testing phase with respect to input detection delay and output delays due to the SFSM operation.

4.1 Considerations on the Proposed Technique

Applicability
The proposed technique is aimed at lower complexity embedded systems, particularly to reactive systems. It is feasible to apply this technique to systems composed of

several SFSM, however, more significant reduction of energy consumption is achieved when all SFSM operate with the same clock period or on multiples of the same clock period.

SFSM Clock Period Selection

In general, most embedded applications allow for a range of the SFSM clock period. The higher the SFSM clock period, the higher the reduction in energy consumption, at the expense of a larger output delay.

The energy consumption (in Wh) over a given time of operation (much larger than the SFSM clock period) is given by (1). The same type of equation (2) is used to calculate the power consumption (in W). Equation 2 can be simplified to (3) when the computation time is considerably smaller than the SFSM clock period. Figure 2 shows the asymptotes of (3) and their crossing point T_m. As may be noticed, there is no benefit in increasing the SFSM clock period above T_m because the maximum consumption reduction has already been achieved. Hence, the optimal SFSM clock period to achieve the maximum reduction in power consumption is given by (4).

$$EC = EC_{ON}\frac{t_{ON}}{T_{clk}} + EC_{Ds}\frac{(T_{clk}-t_{ON})}{T_{clk}} \quad (1)$$

$$C = C_{ON}\frac{t_{ON}}{T_{clk}} + C_{Ds}\frac{(T_{clk}-t_{ON})}{T_{clk}} \quad (2)$$

$$C \cong C_{ON}\frac{t_{ON}}{T_{clk}} + C_{DS} \quad (3)$$

$$T_m = \frac{C_{ON}}{C_{DS}} \times t_{ON} \quad (4)$$

Where:

EC – energy consumption (in Wh)
C – power consumption (in W)
C_{ON} – power consumption with the processor operating
C_{DS} – power consumption with the processor in the deep sleep mode
t_{ON} – computation time
T_{clk} – SFSM clock period
T_m – optimal SFSM clock period

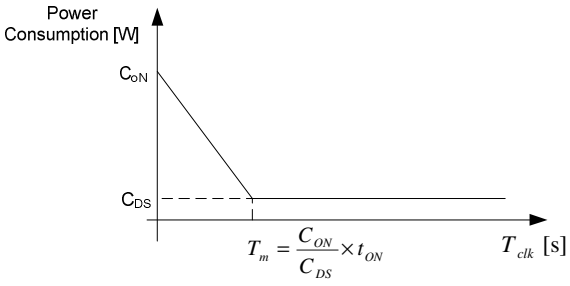

Fig. 2. Asymptotic behavior of equation (3) – power consumption versus SFSM clock period

Figure 3 presents the effect of the SFSM clock period increase on the power consumption of two embedded systems with different power consumptions when the processor is operating. The embedded system 1 (Consumption 1) has a power consumption (C_{ON}) five times lower than embedded system 2 (Consumption 2), while both have the same power consumption in deep sleep mode (C_{DS}). In both cases the reduction in power consumption is significant with the increase of the clock period, up to their T_m (0.021s for the system 1 and 0.1s for the system 2 – obtained from Equation 4). Higher SFSM clock periods than these would only lead to an increased delay in the response time of the system without significant reduction in power consumption.

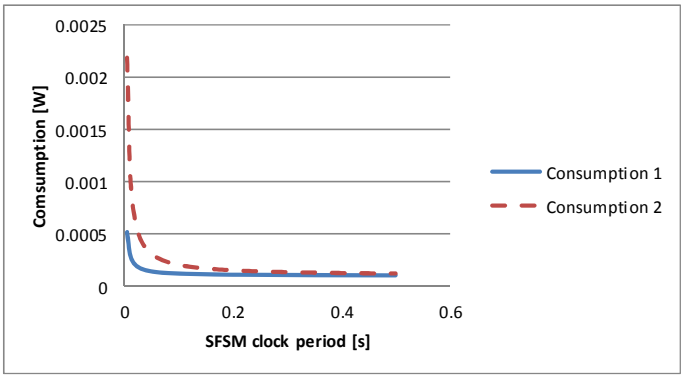

Fig. 3. Effect of the SFSM clock period on the energy consumption

Effects on Real-Time Systems
When a system is modeled as a SFSM a computational delay is added and this delay is dependent on the SFSM clock. The effect of the delay must be taken into account in the design process of the system as well as the maximum possible clock period.

5 Experimental Results

To illustrate the proposed methodology and validate the results the control unit of a tankless gas water heater was developed and measured. In the literature there are other experiments performed with tankless gas water heaters, such as the fuzzy logic control unit presented by Vieira [11].

5.1 Tankless Gas Water Heater Description and Requirements

A tankless gas water heater is composed of a heat exchanger, a gas burner, a control unit (gas heater controller), several sensors (water flow, water outlet temperature, and pilot flame detector) and one actuator (gas valve). As soon as water is flowing through the heat exchanger the gas valve opens and the gas is ignited by the pilot flame. Water is heated while it flows through the heat exchanger. There is no storage

of hot water in this system. As soon as the usage of hot water stops the gas valve is closed and the flame extinguish.

The gas heater controller is the electronic module responsible for the control of the temperature at the water outlet by controlling the gas flow valve. The controller needs to sensor the water temperature (T_w), the water flow (W) and the presence of the pilot flame (P) as well as the desired temperature (T_s) selected by the user.

The functional requirements concern the opening of the gas valve to allow for the adequate gas flow to maintain the water temperature within 2 degrees Celsius from the desired temperature. The safety requirements are that the gas flow valve must be closed if the water flow is below the safety limit (0.5 liters per minute) or the pilot flame is off.

Figure 4 depicts the tankless gas water heater block diagram. The controller receives signals from the water flow sensor (W), the pilot light sensor (P) and the water temperature sensor (Tw). The first two sensors are digital (on/off) while the water temperature sensor is analog. The desired temperature (Ts) is given by a 10 position selector. Table 1 presents the desired temperature corresponding to each position of the selector switch. The water outlet temperature sensor is connected to a 4 bit analog-to-digital converter. The temperatures corresponding to each value of the ADC are also presented in Table 1. The controller output V controls the gas flow valve in the range from 0% (valve totally closed) up to 100% (valve totally open).

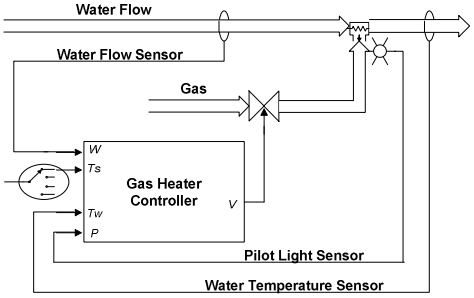

Fig. 4. Tankless gas water heater block diagram

Table 1. Desired and Measured Temperature Ranges

	Desired Temperature (^oC) - T_s	Measured Temperature (^oC) - T_w
0		below 38°
1	40°	38° to 42°
2	44°	42° to 46°
3	48°	46° to 50°
4	52°	50° to 54°
5	56°	54° to 58°
6	60°	58° to 62°
7	64°	62° to 66°
8	68°	66° to 70°
9	72°	70° to 74°
10	76°	74° to 78°
11		78° to 82°
12		82° to 86°
13		86° to 90°
14		90° to 94°
15		94° to 98°

5.2 Control Algorithm Design

The control algorithm is implemented by the SFSM presented in Fig 5. In the *Idle* state, no water is flowing and the gas flow valve is closed. When the water flow is above the minimum level of 0.5 liters/min and the pilot flame is on, then the SFSM transitions to the *Init Delay* state and, after 3 seconds, it transitions to the T state with the gas flow valve at 50%. The control algorithm consists of making no corrections to the gas flow valve output whenever the measured temperature is within 2 degrees of the desired temperature; to do small corrections (2% per 350 ms control cycle) when the temperature difference is small (up to one setting of the temperature selector) and doing larger corrections (10% per 350 ms control cycle) when the temperature difference is larger.

Due to the safety requirement, whenever the water flow is below the minimum level or the pilot flame is off, the SFSM transitions to the *Idle* state.

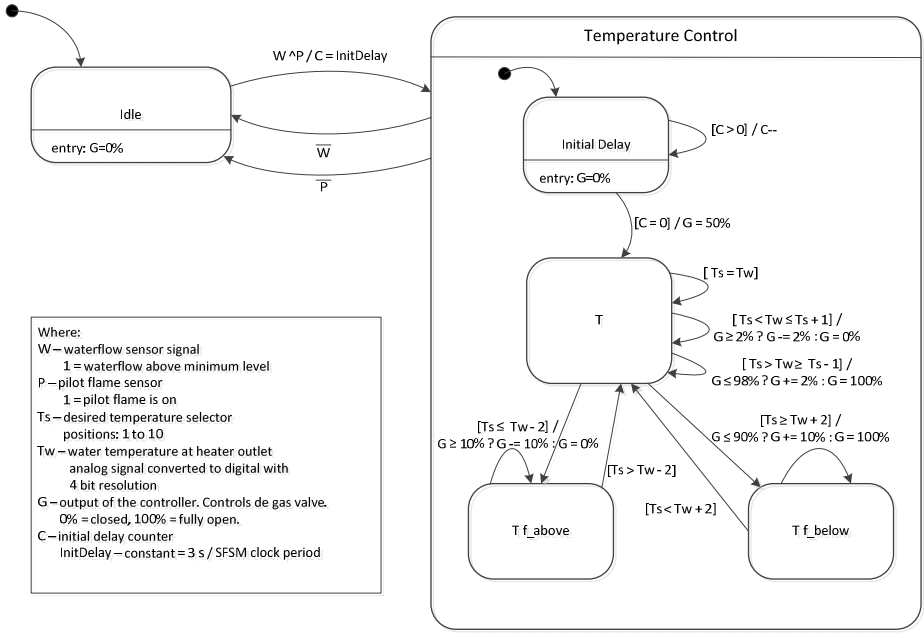

Fig. 5. Tankless gas water heater controller state diagram

5.3 Control Algorithm Simulation

The tankless gas heater was simulated in Matlab/Simulink. The block diagram of the simulation model is depicted in Fig 6. The thermal transfer is modeled as a second order system with a time constant of 1 second. This block models the thermal transfer from the flame to the heat exchanger followed by the thermal transfer from the heat exchanger to the water flow. The change in temperature is given by (derived from the formulae presented by Henze [12]):

$$dT = P\ [kW] \times \frac{1}{flow\ [\frac{kg}{s}]} \times \frac{1}{c_{pw}} \tag{5}$$

where:

P is the power (in kW) generated due to gas burning
flow is the water flow (in kg/s) through the heat exchanger
c_{pw} is the specific heat capacity of water (4.18 kJ/kg.K)

The gas heater controller has an input for the timer tick and two parameterized inputs to accommodate for the changes in the SFSM clock period. The simulation evaluates the desired temperature step-response (desired temperature changed to 40 degrees). Fig 7 shows the response of the simulation: after 12 seconds the system stabilizes within ± 1 degree of the desired temperature.

The simulations were performed with SFSM clock periods ranging from 25 ms to 350 ms. Due to the parameterization of the correction values the responses are nearly identical for all values of the SFSM clock periods.

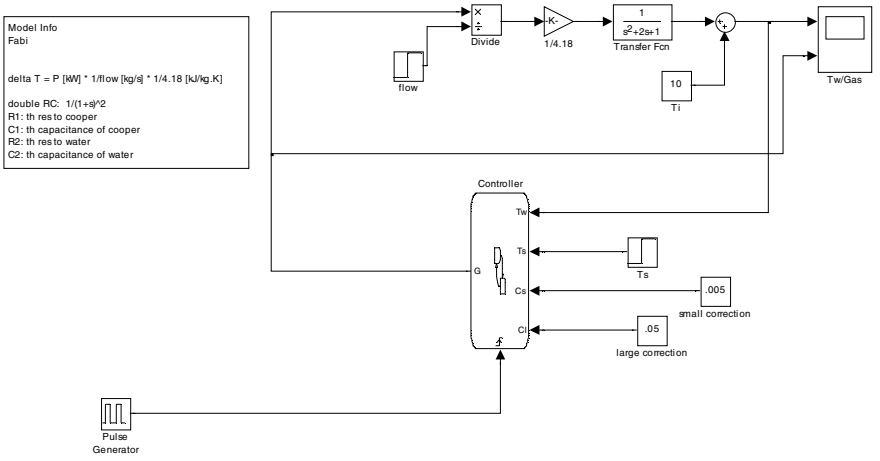

Fig. 6. Tankless gas heater simulation model

5.4 Implementation

For the implementation an ARM Cortex-M processor was used: the NXP LPC 1343, a Cortex-M3 processor running at 72 MHz (Fig 8). The SFSM was implemented in C using the EWARM compiler from IAR. The SFSM clock ticks were generated by the SYSTICK timer, a standard timer available in all Cortex-M3 processors. On every tick the processor wakes up, executes the SFSM and returns to deep-sleep mode (Section 4). On this processor, the required supply current is of 21 mA in the operating mode and of 100 uA in deep-sleep mode.

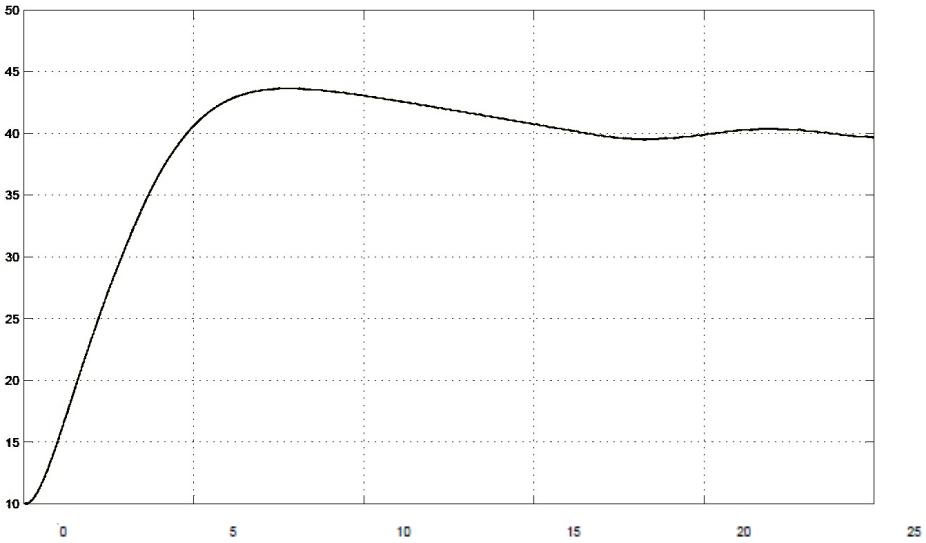

Fig. 7. Step-response obtained by simulation

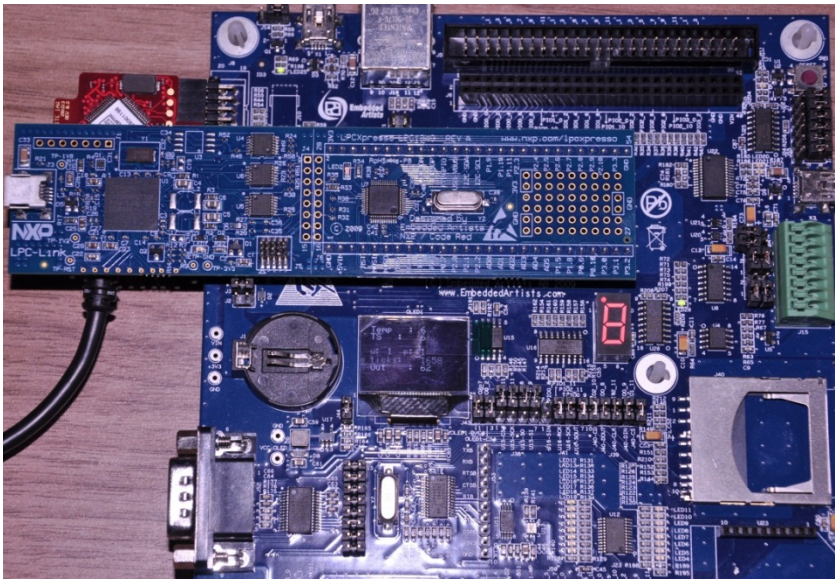

Fig. 8. Photograph of the gas heater controller prototype

Energy consumption was evaluated at three scenarios: (1) processor operating continuously; (2) SFSM clock period of 25 ms and (3) SFSM clock period of 350 ms. For scenarios (2) and (3), deep-sleep mode is used. The measured values of the supply current and supply power are presented in Table 2. The supply voltage is of 3.3 V.

The optimal SFSM clock period (Section 4.1) for this implementation is 21 ms. One can observe that changing the SFSM clock period from 25 ms to 350 ms (a 14 fold increase) results in a power reduction from 606 uW to 350 uW, a reduction of only 42%.

The effects of the changes in the SFSM clock period on the water temperature are not noticeable, as predicted by the simulation results. This is mainly due to the time constants of the physical system (heat transfer from flame to heat exchanger and then to water flow) being larger than the maximum SFSM clock periods used in this evaluation.

Table 2. Measured Power Consumption

Scenario	Supply Current	Power Consumption
Continuously in operating mode	21 mA	69 mW
SFSM clock period of 25 ms.	184 uA	606 uW
SFSM clock period of 350 ms	106 uA	350 uW

6 Conclusion

The proposed methodology advocates the use of Synchronous Finite State Machine models since the early phases of development as well as a concern, during requirements elicitation, for delay tolerances. In this way, a straightforward implementation technique of short execution bursts followed by periods of low-power deep-sleep results in significant reduction of energy consumption.

A tankless gas water heater controller was implemented with the proposed methodology. This controller is powered from a 4000 mAh non-rechargeable battery whose life is extended from 190 hours (if the processor is in continuous operating mode) to 2.25 years (using deep-sleep mode).

One could argue that the total amount of energy consumed by a gas water heater renders useless the small amount of energy that is saved in the given example. However, there are two important considerations: (1) currently the gas water heater has two energy sources – gas and electricity; the potential for significant reduction in energy consumption from the battery will extend battery life, reducing cost and environmental waste. (2) Since gas water heaters produce waste heat, one could apply energy harvesting to power the electronics, provided a start-up mechanism is available.

The proposed methodology is currently aimed at devices of lower complexity. A future research direction is broadening the scope to include more complex devices; these are likely to require the use of an RTOS that should manage the entry into deep-sleep mode. Another research direction is the application of the technique to Asynchronous Finite State Machines, aiming at the devices whose functionality is not adequately modeled by SFSM.

References

1. International Energy Agency – 2012 Key World Energy Statistics (2012), http://www.iea.org/publications/freepublications/publication/name,31287,en.html
2. Bohn, J., Coroama, V., Langheinrich, M., Mattern, F., Rohs, M.: Social, Economic and Ethical Implications of Ambient Intelligence and Ubiquitous Computing. In: Weber, W., Rabaey, J.M., Aarts, E. (eds.) Ambient Intelligence, pp. 5–29. Springer (2005)
3. Strba, A.: Embedded Systems with Limited Power Resources, Enocean (2009), http://www.enocean.com/fileadmin/redaktion/pdf/white_paper/wp_embedded_systems_en.pdf
4. Inführ, J., Jahrmann, P.: Hard- and Software Strategies for Reducing Energy Consumption in Embedded Systems. Seminar-Thesis, Vienna University of Technology (2009)
5. Venkatachalam, V., Franz, M.: Power Reduction Techniques for Microprocessor Systems. ACM Computing Surveys 37(3), 195–237 (2005)
6. Ortiz, D.A., Santiago, N.G.: Highl Level Optimization for Low Power Consumption on Microprocessor-Based Systems. In: 50th Midwest Symposium on Circuits and Systems, Montreal, pp. 1265–1268 (2007)
7. Wiedenhoft, G.R., Hoeller Jr., A., Fröhlich, A.A.: Um Gerente de Energia para Sistemas Profundamente Embarcados. In: Workshop de Sistemas Operacionais, Rio de Janeiro, pp. 796–804 (2007)
8. Ishihara, T., Goudarzi, M.: System-Level Techniques for Estimating and Reducing Energy Consumption in Real-Time Embedded Systems. In: International SoC Design Conference, Seoul, pp. 67–72 (2007)
9. Tiwari, V., Malik, S., Wolfe, A.: Power Analysis of Embedded Software: A First Step Towards Software Power Minimization. IEEE Transactions on VLSI Systems, 437–445 (1994)
10. Huang, K., Santinelli, L., Chen, J., Thiele, L., Buttazzo, G.C.: Adaptive Dynamic Power Management for Hard Real-Time Systems. In: 30th IEEE Real-Time Systems Symposium, Washington, pp. 23–32 (2009)
11. Vieira, J.A.B., Mota, A.M.: Modeling and Control of a Water Gas Heater with Neuro Fuzzy Techniques. In: 3rd WSEAS International Conference, Switzerland, pp. 3571–3576 (2002)
12. Henze, G.P., Yuill, D.P., Coward, A.H.: Development of a Model Predictive Controller for Tankless Water Heaters. HVAC&R Research 15(1), 3–23 (2009)
13. Liu, X., Shenoy, P., Corner, M.D.: Chameleon: Application-Level Power Management. IEEE Transactions on Mobile Computing 7(8) (August 2008)