Johanna Tuominen | Tero Säntti | Juha Plosila

Towards a Formal Power Estimation Framework

Turku Centre *for* Computer Science

TUCS Technical Report
No 672, March 2005

Towards a Formal Power Estimation Framework

Johanna Tuominen
Turku Centre for Computer Science
Lemminkäisenkatu 14 A, 20520 Turku, Finland
joeltu@utu.fi
Tero Säntti
University of Turku, Dept. of Information Technology
Lemminkäisenkatu 14 - 18, 20520 Turku, Finland
teansa@utu.fi
Juha Plosila
University of Turku, Dept. of Information Technology
Lemminkäisenkatu 14 - 18, 20520 Turku, Finland
juplos@utu.fi

TUCS Technical Report
No 672, March 2005

Abstract

Conventionally, the correctness of functional and non-functional properties of hardware components is ensured during design process by simulation. Moreover, different description languages are needed during development phases. Thus, by adopting the Action Systems, we are able to use the same formalism from specification down to implementation. In this study, we introduce a formal approach for an abstract level power estimation in Action Systems context. The purpose is to develop formal power estimation flow, which can be used to monitor the power consumption from an abstract level down to the gate level implementation.

Keywords: Power estimation, formalism, abstract-level

TUCS Laboratory
Communication Systems

1 Introduction

Formal methods provides an environment to design, analyze, and verify digital hardware with the benefits of rigorous mathematical basis. In this study, the Action Systems formalism is applied [2]. It is a framework for specification and correctness preserving development of concurrent systems and it is based on an extended version of Dijkstra's language of guarded commands [3]. Development of the action system is done in a stepwise manner within the refinement calculus [1]. The specification of a hardware system is transformed into an implementation using correctness preserving transformations. In conventional Action Systems, only the logical correctness of the system is verified, while non-functional properties, like time, power and area, are not validated. The Action Systems formalism has been proved to be suitable for designing both synchronous [5], and asynchronous [4] systems.

In this study, we introduce a formal approach for power estimation in an Action Systems context. The power analysis is carried out in an abstract level using basic Action System compositions and system structures as an example. For the power consumption estimate, we have to specify an approximation for energy consumption and an execution time for the action under analysis. Moreover, we specify an activity factor for the given action, where we can assume either continuous or discrete time. The purpose is to develop a formal power estimation flow from initial specification down to implementation [6]. This would give us a possibility to formally estimate and verify the power consumption of a hardware system during the design process.

Overview of the paper; We proceed as follows. In Section 2 we shortly describe the properties of the Action Systems formalism. Section 3 concentrates on the semantics of the abstract level power estimation, and show how these are applied to the basic composition structures. Section 4 discusses system level issues, and gives a more detailed example on system level power estimation. Finally in Section 5, we draw some conclusions and describe future efforts in a field of formal power analysis.

2 Action Systems

Action Systems [2, 4] is a state-based formalism for concurrent system specification and correctness-preserving development. The basic building blocks of the formalism are called $actions$. An action A is defined (for example) by

$$
\begin{array}{ll}
A ::= abort & (abortion, non-termination) \\
\quad | \; skip & (empty\ statement) \\
\quad | \; A_1 [\!] \ldots [\!] A_n & (non-deterministic\ choice) \\
\quad | \; A_1; \ldots ; A_n & (sequential\ composition) \\
\quad | \; x := e & ((multiple)\ assignement) \\
\quad | \; g \rightarrow A & (guarded\ command)
\end{array}
$$

where A_i, $i = 0, ..., n$, are actions; x is a variable or a list of variables; x_0 is a value(s) of the variable(s); e is an expression or a list of expressions; g is a predicate.

The actions are defined using weakest precondition for predicate transformers [3]. For instance, the correctness of an action A with respect to predicates P and Q (precondition and postcondition) is denoted by:

$$\{P\}A\{Q\} = P \Rightarrow wp(A, Q)$$

Here $wp(A, Q)$ is the weakest precondition for the action A to establish the postcondition Q.

Action is considered to be atomic, which means that only the initial and final states are observed by the system. Furthermore, when action is selected for execution, it is completed without any interference from other actions.

The $guard\ gA$ of an action A is defined by $gA = \neg wp(A, false)$. An action is enabled when its guard evaluates to $true$, otherwise disabled.

2.1 Action System

An action system has a form:

sys *Name* (g) $[par]$
$|[$
type t
const c
var v
actions A
init "$initialization\ of\ the\ variables\ g\ and\ v$"
exec
do "$composition\ of\ actions\ A''$ **od**
$]|$

Three different parts can be identified from the action system description: $interface$, $declarations$, and $iteration$.

The interface part specifies global variables g, that is, variables that are visible outside the action system. In other words, global variables are accessible by other action systems. If an action system does not have any interface variables, it is a $closed$ action system otherwise it is an $open$ action system. The declaration part consists of type (t), variable (v), constant (c), and action (A) declarations. Furthermore, type definitions and initializations are described in the declaration part.

In this paper, we will use a $non\text{-}atomic\ composition$ structures in the system models. Non-atomicity means that an action outside the composition can execute between two component actions of the construct, which is not possible in the $atomic$ composition structures. For instance, we will use the bold semicolon '**;**' as the operator symbol for the non-atomic sequential composition.

The operation of an action system is started by initialization in which the variables are set to predefined values. Actions are selected for execution based on the composition operators and the enabledness of the actions. Thus, non-independent actions cannot operate operate in parallel. The operation is continued until there are no actions to enable, which temporarily aborts the system. Thus, the operation continues if some action enables it.

3 Semantics of Abstract Power Modeling

This section discusses about the semantics of abstract power modeling. At first, we define the activity factor, which is used to estimate the number of execution per action during some predefined time period. Then we continue the discussion by analyzing the power estimation procedure for action compositions.

3.1 Activity Factor

Consider an example action A. We can approximate the energy consumption and the execution time of the action A as e_A and t_A, respectively. Therefore, we can estimate that one execution consumes power by the amount of $P_A = \frac{e_A}{t_A}$. In general, we should be able to estimate the power consumption during some time period T, which includes several execution cycles. Therefore, we define an activity factor α which describes the number of execution times during some time period T. For the action, A we define the activity factor by adopting the execution sequence, shown in Figure 1.

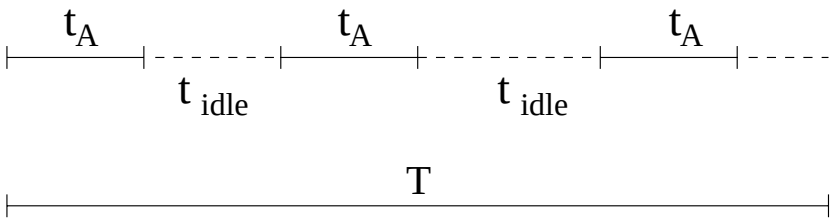

Figure 1: Execution sequence for the action *A*

In general, we can define an estimate for the activity factor of the action A during a certain time period T by:

$$\alpha(A, T) = \frac{n \cdot t_A}{T}$$

where the n is a number of executions during time period T. However, there may be idle period of variable length between the executions, as shown in Figure 1. Therefore, we define time period T by:

$$T = n \cdot t_A + \sum t_{idle}$$

which is a sum of the execution times $n \cdot t_A$ and the sum of the idle periods t_{idle}. Next, by using the definition for the T, we can estimate the activity factor $\alpha(A, T)$ as shown in Equation 1.

$$\alpha(A, T) = \frac{n \cdot t_A}{n \cdot t_A + \sum t_{idle}} = \frac{t_A}{t_A + \frac{\sum t_{idle}}{n}} \tag{1}$$

Next, we assume that there is no fixed time period T under which we estimate the activity factor. Thus, the activity factor is estimated without any time periods, i.e. by adopting the concept of continuous time. Therefore, we re-define the equation 1 by:

$$\alpha(A) = \frac{t_A}{t_A + \lim \frac{\sum t_{idle}}{n}} \tag{2}$$

where evaluation process depends on the limit value of $\frac{\sum t_{idle}}{n}$. For instance, if we assume that $\lim_{n\to\infty}$, the value of $\alpha(A)$ is approaching to long term average. Thus, in general we can define the power estimate for the action A as shown in Equation 3.

$$P_A = \frac{e_A}{t_A} \cdot \alpha(A) \tag{3}$$

3.2 Power Estimation for Action Compositions

Consider two arbitrary actions A and B. We assume that the actions are composed as follows:

do $A \parallel B$ **od**

We define the unit energy values and execution times for the actions A and B as (e_A, t_A) and (e_B, t_B), respectively. The activity factor α for actions A and B is defined according to Equation 2.

At first, we assume that the actions are independent (no write-read or write-write conflicts between the actions A and B). Thus, from the Action System description we can define two execution sequences: (AB) and (BA). Thus, independent actions have potential for parallel behavior. The physical interpretation of the execution sequence AB is illustrated in Figure 2 (a)-(b).

The first two sequences 2(a)-(b) illustrates the simultaneous execution of the actions A and B. The third one 2(c) models parallel behavior as well, but the actions are not enabled simultaneously. Finally the last sequence 2(d) describes sequential execution.

Consider the situation, shown in Figure 2(b), where the two actions are executed at the same time, but the other one is completed earlier. We can estimate the average power consumption by using the execution time of the slowest action, noted as $max(t_A, t_B)$. The transition activity is defined for both actions separately

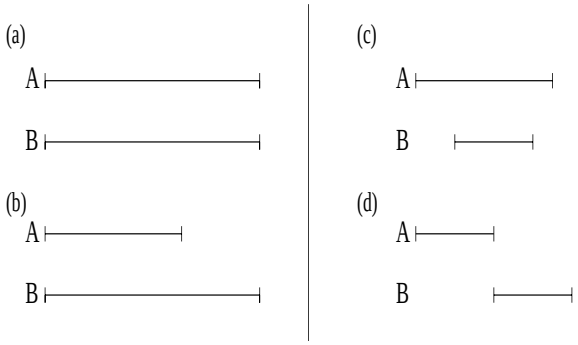

Figure 2: Physical interpretations of the execution sequences for the actions *A* and *B*

because, for instance, there might be cycle where the action B is not executed at all. The average power estimate is shown in Equation 4.

$$P_{avg} = \frac{e_A \cdot \alpha(A) + e_B \cdot \alpha(B)}{max(t_A, t_B)} \tag{4}$$

Moreover, we can estimate the instantaneous power consumption as shown in Equation 5.

$$P_{inst} = \frac{e_A}{t_A} \cdot \alpha(A) + \frac{e_B}{t_B} \cdot \alpha(B) \tag{5}$$

In conclusion, we can estimate the average power consumption and instantaneous power consumption for the execution sequences 2(a)-(c) by using the equations 4 and 5, respectively.

By adopting the asynchronous execution sequence from Figure 2(d) the power consumption estimate is calculated as shown in Equation 6.

$$P = \frac{e_A \cdot \alpha(A) + e_B \cdot \alpha(B)}{t_A + t_B} \tag{6}$$

Moreover, if the operation of the actions A and B would not considered to be independent, this would be the only possible execution sequence.

To illustrate the effect of timing in power estimation, we assumed that the composition presented in Figure 2(a) is synchronous. Therefore, we can roughly estimate the power consumption by assuming that the execution time is half of the clock period, $(t_A, t_B) = t_{clk}/2$.

Moreover, we assume that the actions A and B have completed their tasks during one clock cycle. Thus, we can define the the power consumption estimate for the synchronous execution as shown in Equation 7.

$$P = \frac{e_A \cdot \alpha(A) + e_B \cdot \alpha(B)}{t_{clk}/2} \tag{7}$$

The $\alpha(A)$ and $\alpha(B)$ are denoted as an activity factors for actions A and B, respectively.

Comparison were made between the synchronous and the asynchronous sequences, shown in the Figures 2(a) and 2(d), respectively. The purpose was to analyze the effect of timing to the power consumption estimate. Therefore, we can simplify the equation (7) for synchronous execution to $P = \frac{1}{T_{clk}/2}$, which is noted as a relative power consumption. For instance, we can assume that the two actions A and B are executed using a clock frequency of $f_{clk} = 2\,GHz$. Thus, the relative power consumption is 4. Similarly, the equation for asynchronous execution 6 can be simplified to $P = \frac{1}{T_A+T_B}$. By adopting the 0.35 ns execution time per action the relative power consumption for sequential execution is decreased $31\ \%$ with the expense of 0.15 ns increment in delay. Therefore, comparisons between these two equations shows that there is a trade off between speed and power consumption.

4 Abstract Level Power Consumption Estimation

In this section the power estimation techniques are applied to the action system structures. At first, we give an overview of the system level estimation approach. Then we concentrate on defining the power estimate for a one subsystem description. The scope is at abstract level, which give us possibility to analyze problem areas, that has to be taken care as we move towards the gate level analysis.

4.1 Overview of the System Level Estimation

Consider the following example, an abstract level description of the target module M and its environment Env, shown in Figure 3.

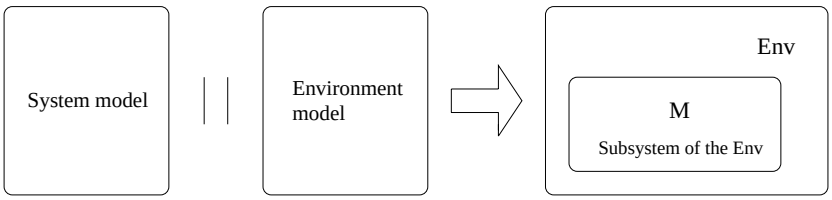

Figure 3: Target system module within its environment

The computation is carried out in the target module M, but communication is assumed between the module M and its environment Env. To analyze the power consumption, we roughly divide the system into three power consuming parts: the targeted system M, communication channel Com, and the environment Env. We can estimate the power consumption during time period t, if we have some information or approximation on the modules energy consumption and their execution times. The activity factor is defined according to the equations 1 and

2. For instance, if we assume that the operations of the given system construct is sequential, and that the execution times of the different parts do not overlap, the power estimate can be described as:

$$P_M = \frac{e_C \cdot \alpha(E) + e_M \cdot \alpha(E) + e_E \cdot \alpha(E)}{t_C + t_M + t_E}$$

where the e_C, e_M, and e_E are the energy consumption of one computation cycle, and the t_C, t_M, and t_E are the corresponding execution times. The activity factor α is described under continuous time domain. Depending on the timing issues the power consumption estimate varies, which is discussed more detailed in the next section.

The initial specification of the target system module M can be decomposed into an architecture of dedicated subsystem modules. Typically lots of new variables, procedures, invariants and protocols are introduced during the refinement process. This combined with several atomicity refinement steps introduces new actions into the system. Thus, result of the decomposition is a correct architecture model from the initial specification. The first steps of the decomposition process is illustrated in Figure 4.

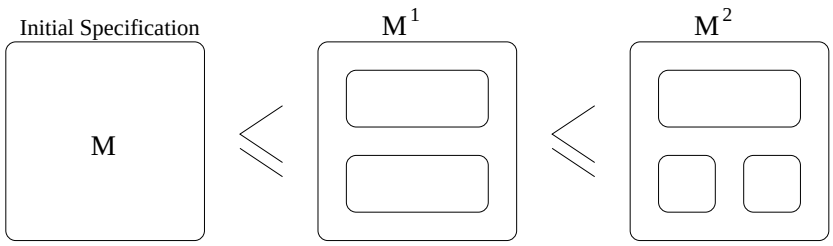

Figure 4: Decomposition steps for target system

The initial specification of the module M is extracted into two submodules $M1$ and $M2$. The power estimation techniques can be applied to estimate the power consumption for each of the submodules together and separately. The purpose is to develop a flow that can monitor the power consumption from the initial specification to the final architecture model. The next case study presents the power estimation procedure for a one submodule.

4.2 Detailed Subsystem Power Estimation

The system $sysMod$ specification includes three arbitrary actions: A, B, and C.

sys *subMod* ()
actions A, B, C
exec
do
$(A$ **;** $B) \parallel C$
od

The estimation procedure depends on how the abstract level description is interpreted. From the Action Systems we can define three possible execution sequences: ABC, CAB, and ACB. At first, consider the case when the operation of the actions is not independent. In other words, the execution times of the given actions do not overlap, and therefore the operation is sequential. Thus, the power estimate can be constructed according to the Equation 6.

$$P_{subMod} = \frac{e_A \cdot \alpha(A) + e_B \cdot \alpha(B) + e_C \cdot \alpha(C)}{t_A + t_B + t_C}$$

The activity factor α is defined according to the equation 2. Furthermore, we estimate the energy consumption and execution times for each action A, B, and C to be (e_A, e_B,e_C) and (t_A, t_B, t_C), respectively.

In the second case, we assume that the operation of the action C is independent with respect to the operation of the actions A and B. Thus, the execution of the actions A is followed by the execution of the action B, and the action C operates parallel with the actions A and B. This situation is illustrated in Figure 5. Therefore, the problem is to determine the time t_x, which describes the amount of delay before the action C is executed. The first execution sequence, shown in Figure 5, presents the worst case situation in terms of power consumption. Thus, the action C is executed simultaneously along with the actions A and B. The last one presents the best case where the actions are executed sequentially.

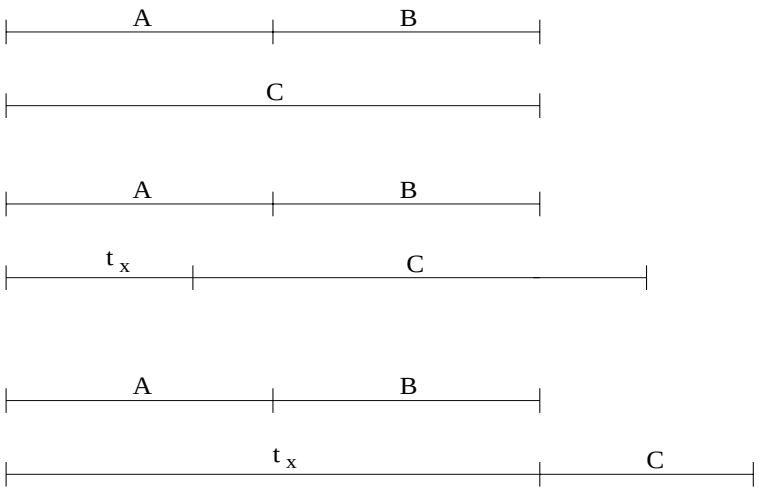

Figure 5: Physical interpretation of the executions sequences for the system *subMod*

In order to determine the power estimate in each case we have to be able to estimate the value of t_x properly. The t_x is variable delay between the execution of the action A and the action C, as shown in the Figure 5. Therefore, we estimate the power consumption of the system $subMod$ by integrating over the difference of the starting times. The equation for the power estimate is shown in 8.

$$\frac{1}{t_{tot}} \int_{-t_c}^{t_a+t_b} P(t_x) dt_x \tag{8}$$

The equation, shown in 8, does not take into account the possible idle periods between two executions. To evaluate the integral, we divided the analysis into three cases:

$$t_A + t_B < t_C$$
$$t_A + t_B = t_C$$
$$t_A + t_B > t_C$$

where the second one represents the situation when the actions are executed simultaneously. The first and the last case presents the situation where the execution of action C is interleaved by the amount of t_x. Furthermore, the first and the last case returns to a similar result, and therefore it is necessary to discuss only one of them. For simplicity, we define that $t_A + t_B = t_Z$. In conclusion, we have two cases under evaluation: $t_Z = t_C$ and $t_Z \geq t_C$. The graphical representations of these two cases are shown in Figure 6 (a) and (b), respectively.

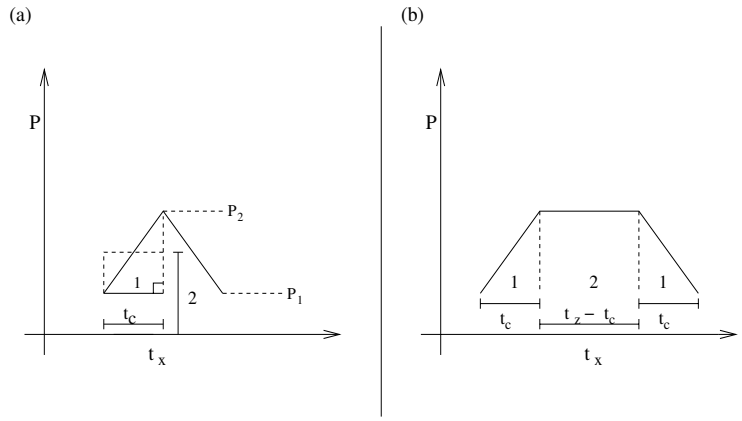

Figure 6: Graphs for power estimation

We solved the equations for the power estimates with the aid of geometry. At first, consider the situation where $t_Z = t_C$, shown in Figure 6 (a). The power estimate integral is solved by using the properties of the right triangle. Therefore, we define the pivotal points for the triangle, shown in Figure 6 (a). The points are defined as: $P_1 = \frac{e_{tot}}{t_Z + t_C}$ and $P_2 = \frac{e_{tot}}{t_Z}$. Next, we form a square from the right triangle, marked as 1 in the Figure 6 (a). Thus, now we can determine the length of the 2, which is $P_1 + \frac{P_2 - P_1}{2}$. In other words the average power consumption for the simultaneous execution is shown in Equation 9.

$$P_{avg} = P_1 + \frac{P_2 - P_1}{2} = \frac{P_2 + P_1}{2} \tag{9}$$

Next, consider the situation shown in Figure 6 (b), where the $t_Z \geq t_C$. By applying the same pivotal points as in the previous example, we can define the average power consumption by:

$$P_{avg} = \frac{2 \cdot (P_1 + \frac{P_2 - P_1}{2}) \cdot t_C + P_2(t_Z - t_C)}{t_C + t_Z} = \frac{(P_2 + P_1) \cdot t_C + P_2(t_Z - t_C)}{t_C + t_Z} \tag{10}$$

By assuming that the $t_Z - t_C = 0$ the equation 10 recurs to the equation 9. In other words, the power estimate presented in Equation 9 is a special set from the estimate presented in Equation 10.

5 Conclusions and Future Work

In this paper, we introduced an approach to estimate power consumption in an Action Systems context. The work carried out so far showed that the power consumption procedure is highly time dependent process, and therefore we analyzed timing and energy consumption separately. For instance, if we consider the activity factor specification we have to know whether the activity factor is calculated under discrete time period or continuous one. Furthermore, we have to take into account the possible idle periods between the execution of an action. Secondly, as we moved into system level implementations, the timing issues depends on whether the actions under investigation are independent or not. Thus, independent actions introduces parallel behavior, which complicates the estimation procedure. The issues presented above were analyzed using example compositions and systems.

Future Work: The experiences of this study showed the possibilities and the problem areas to formally investigate and verify the power estimation from an abstract level system description. The next step is to expand the current work into a complete power estimation framework for the Action System formalism. This includes the definition of update action, which should monitor the energy consumption and timing from the abstract level system specification to the final architecture model. Moreover, the update action should include possibility to insert technology dependent information as we move towards gate level analysis. For timing, we have a framework for timed actions [7], which we will use as a guideline. In conclusion, the purpose is to create a power estimation flow that would be usable for both synchronous and asynchronous systems.

References

[1] R. J. R. Back, *On the Correctness of Refinement Steps in Program Development*, Ph.D Thesis, University of Helsinki, 1978.

[2] R. J. R. Back and K. Sere, *From Modular Systems to Action Systems*, in Proc. of Formal Methods Europe' 94, Spain, October 1994. Lecture notes on computer science, Springer-Verlag.

[3] E. W. Dijkstra, *A Discipline of Programming*, Prentice-Hall International, 1976.

[4] J. Plosila, *Self-Timed Circuit Design - The Action Systems Approach*, Ph.D Thesis, University of Turku, 1999.

[5] T. Seceleanu, *Systematic Design of Synchronous Digital Circuits*, Ph.D Thesis, Turku Centre for Computer Science, 2001.

[6] J. Tuominen and J. Plosila, *High Level Power Estimation*, Turku Center for Computer Science Technical Report Series, Number 623, September 2004, ISBN 952-12-1416-3.

[7] T. Westerlund and J. Plosila, *Formal Timing Model for Hardware Components*, in Proc. IEEE Norchip 2004, November 8-9, Oslo, Norway.

Turku
Centre *for*
Computer
Science

Lemminkäisenkatu 14 A, 20520 Turku, Finland | www.tucs.fi

University of Turku
- Department of Information Technology
- Department of Mathematics

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration
- Institute of Information Systems Sciences

ISBN 952-12-1517-8
ISSN 1239-1891