Towards a Formal Power Estimation Framework for Hardware Systems

Johanna Tuominen*†, Tero Säntti †, and Juha Plosila†
*Turku Centre of Computer Science, Finland
†Dept. of information Technology, University of Turku, Finland
{joeltu|teansa|juplos}@utu.fi

***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, exploit the possibilities to formally model power consumption in Action Systems context. The purpose is to develop formal power estimation flow, which can be used to monitor the power consumption from abstract level down to the gate level implementation.**

I. 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 [6], and asynchronous [5] systems.

Advances in VLSI technology over recent years have considerably increased both physical and functional size of single-chip systems. Thus, it is obvious that with the increased circuit integration, power dissipation aspects assumes greater importance [4]. Moreover, with the advent of wireless and mobile high performance computing platforms, and the limited operational life time of batteries, low power designs are required. To estimate the power consumption, there is a trade-off between the accuracy and the abstraction level of detail which the system is analyzed. The more detailed the description, the more accurate the simulation will be. But on the other hand, the more time consuming it will be. Moreover, the designer wants to make decisions as early as possible in the design flow to avoid design backtracking.

In this study, we exploit the possibilities to formally model physical quantity, power. We analyze the power consumption for the basic Action Systems compositions and system structures as an example. This paper lays foundation for the formal power estimation framework. In other words, the purpose is to develop a formal power estimation flow from initial specification down to implementation [7]. This would give us a possibility to formally estimate the power consumption of a hardware system during the different phases of the design process.

II. Action Systems

An action A is defined by (for example):

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

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.

A. 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. Using the items introduced in the interface and declarative parts the operation of the system is described in the iteration section; in the **do** $-$ **od** loop.

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. 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. In this study, we will discuss of $non-independent$ and $independent$ actions. Thus, independent actions may operate in parallel (no write-read or write-write conflicts between the actions executed in parallel), while the non-independent actions cannot.

III. Semantics of Formal Power Modeling

A. Timing and Activity

Consider an arbitrary action A, which consumes power during one execution by the amount of $P_A = \frac{e_A}{t_A}$. The e_A is an energy consumption, and the t_A is the execution time of the action A. Next, we assume that the action A is executed several times during discrete time period T, shown in Figure 1.

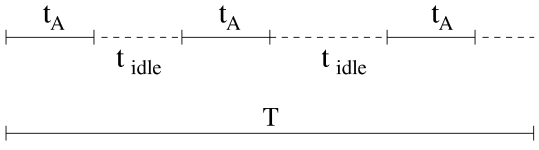

Fig. 1. Execution sequence for the action *A*

At first, we have to define the activity, noted by α, of the action A during the time period T.

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

where the n is a number of executions during the time period T. The time period T is then defined by: $n{\cdot}t_A{+}\sum t_{idle}$, which is the sum of the idle periods t_{idle} and the execution times t_A. By applying the definition of the time period T to the equation 1, we can estimate the activity $\alpha(A,T)$ to be:

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

Assuming that there is no fixed time period T under which the activity is estimated. Thus, the time is assumed to be continuous, and the equation 2 can be re-written into:

$$\alpha(A) = \frac{t_A}{t_A + \lim \frac{\sum t_{idle}}{n}} \tag{3}$$

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

$$P_A = \frac{e_A}{t_A} \cdot \alpha(A) \tag{4}$$

B. Power Analysis for Action Compositions

Consider two arbitrary actions A and B. We denote the energy consumption and execution times for the actions A and B by (e_A,t_A) for action A and (e_B,t_B) for action B. The activity α is defined for both actions according to the equation 3.

The actions are composed as follows: **do** $A \parallel B$ **od**. The result of the power estimation depends on how we physically interpret this composition. Moreover, how we are able to model these physical aspects into the formal power consumption model in the future. The operation of the actions A, and B can be either independent (no write-read or write-write conflict between the actions A and B) or non-independent. Notice that, the independent actions introduces a possibility for parallel behavior, which has influence on the power estimate. Therefore, we have four different execution sequence for the $non\text{-}deterministic\ choice$, shown in Figure 2.

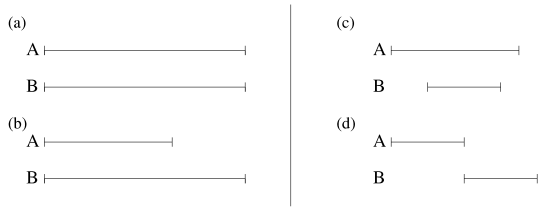

Fig. 2. Physical interpretations of the execution sequences for the composition **do** $A \parallel B$ **od**

We can divide the power estimation for the independent actions into two parts. At first, consider the situations shown in Figure 2 (a)-(c). Thus, in the figure 2 (a) the two actions are enabled simultaneously, and their execution times are assumed to be equal ($t_A = t_B$). On the contrary, in the figure 2 (b), we have simultaneous execution, but the execution times are not equal ($t_A \neq t_B$). Finally, in the figure 2 (c), (for instance), the action B is executed in parallel with action A at some time

during the execution time t_A. The average power consumption estimation for these three cases is defined by:

$$P_{avg} = \frac{e_A \cdot \alpha(A) + e_B \cdot \alpha(B)}{max(t_A, t_B)} \quad (5)$$

where $\alpha(A)$ and the $\alpha(B)$ are denoted as an activity for actions A and B, respectively. The $max(t_A, t_B)$ denotes the execution time of the slowest action. The transition activity is defined for both actions separately, because for instance, there might be a cycle where the action B is not executed at all.

Moreover, we can estimate the instantaneous power consumption for the execution sequences, shown in Figure 2 (a)-(c).

$$P_{inst} = \frac{e_A}{t_A} \cdot \alpha(A) + \frac{e_B}{t_B} \cdot \alpha(B) \quad (6)$$

Secondly, we consider the situation, shown in Figure 2 (d). The two actions A and B are executed sequentially. Therefore, the power consumption is estimated by:

$$P_{avg} = \frac{e_A \cdot \alpha(A) + e_B \cdot \alpha(B)}{t_A + t_B} \quad (7)$$

In conclusion, if the actions A and B are considered independent, the difference between formal notation and the physical interpretation may vary significantly. Thus, to determine the order of the events in the hardware system will have significant role in the formal power model. Moreover, another critical issue is to detect the idle periods between enabled actions. Therefore, we can conclude that timing issues plays a significant role in the formal power estimation model. However, if the actions A and B are considered to be non-independent, the actions are executed sequentially, and the power consumption is estimated according to the equation 7.

IV. System Level Power Estimation

A. Overview

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 of the initial specification. The first steps of the decomposition process is illustrated in Figure 3.

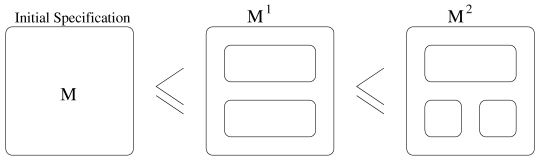

Fig. 3. Decomposition steps for the target system *M*

The initial specification of the module M is extracted into two submodules $M1$ and $M2$. Therefore, we can estimate the power consumption for each of the submodules separately, and then together. Notice that, the the division into submodules simplifies the estimation process.

B. Example: System level analysis

Consider the following Action System M, which includes three arbitrary actions: A, B, and C.

sys M ()
actions A, B, C
exec
do
$(A \; ; \; B) \; [\!] \; C$
od

From the action system composition **do** $(A \; ; \; B) \; [\!] \; C$ **od**, we can define three possible execution sequences: ABC, CAB, and ACB. However, the estimation procedure depends on how the system description is physically interpreted. The physical interpretations of the given execution sequence is shown in Figure 4.

At first, assume that the operation of the actions is non-independent. In other words, the execution times do not overlap between the actions A, B, and C. Thus, the operation of the actions is sequential, and the power estimate is defined by:

$$P_M = \frac{e_A + e_B + e_C}{t_A + t_B + t_C} \quad (8)$$

Secondly, we assume that the action C is independent with respect to actions A and B. Thus, the execution of the action A is followed by the execution of the action B, and the action C operates parallel with the actions A and B. Therefore, to estimate the power consumption of the system M, the problem is to determine the time t_x, which describes the amount of delay before the action C is executed, shown in Figure 4. The first execution sequence, shown in Figure 4, 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.

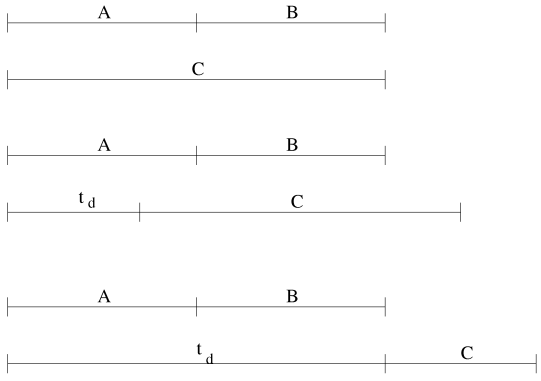

Fig. 4. Physical interpretation of the executions sequences for the system *M*

The time t_x is a variable delay between the execution of the action A and the action C, as shown in the Figure 4. Therefore, we estimate the power consumption of the system M by integrating over the difference of the starting times. The equation for the power estimate is shown in 9.

$$\frac{1}{t_{tot}} \int_{-t_c}^{t_a+t_b} P(t_x) dt_x \tag{9}$$

To evaluate the integral, the analysis is divided 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 A, and B are executed simultaneously with the action C. The first and the last case presents the situation where the execution of action C is interleaved by the amount of t_x. For simplicity, we define that $t_A + t_B = t_Z$, because the first and the last case results to a similar result, and therefore it is necessary to discuss only one of them. Therefore, the system composition is re-written into: **do** $Z \parallel C$ **od**. Moreover, we have only 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 5 (a) and (b), respectively.

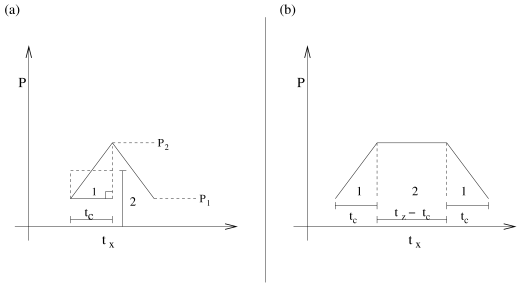

Fig. 5. Graphs for power estimation

The equation for the power estimate for the system M, shown in Equation 9, is solved with the aid of geometry. At first, consider the situation shown in Figure 5 (a), where the power estimate integral is solved by using the properties of the right triangle. Thus, we define the pivotal points for the triangle, shown in Figure 5 (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 5 (a). Thus, now we can determine the length of the 2, which is $P_1 + \frac{P_2 - P_1}{2}$. The average power consumption for the synchronous execution is shown in Equation 10.

$$P_{avg} = P_1 + \frac{P_2 - P_1}{2} = \frac{P_2 + P_1}{2} \tag{10}$$

Next, consider the situation shown in Figure 5 (b), where the $Z \geq C$. By applying the same pivotal points as in the previous example, we can define the average power consumption by:

$$P_{avg} = \frac{(P_2 + P_1) \cdot t_C + P_2(t_Z - t_C)}{t_C + t_Z} = \frac{P_1 \cdot t_C + P_2 \cdot t_Z}{t_C + t_Z} \tag{11}$$

By assuming that the $t_Z - t_C = 0$, the equation 11 recurs to the equation 10. In other words, the power estimate presented in Equation 10 is a special set from the estimate presented in Equation 11.

V. Conclusions and Future Work

In this paper, an early level specification for the formal power consumption model is introduced. We analyzed a basic Action Systems compositions and structures in terms of power consumption. At first, we discussed about timing and activity issues, which showed that the power consumption estimation is highly time dependent task. To determine the power consumption, we need to know the activity of the given action(s). For instance, in an asynchronous circuit there might be long idle period before the particular action is executed again. The problem rises how to model this behavior properly. Moreover the activity differs, whether we assume the time to be continuous or discrete. Therefore, we decided to analyze the timing and energy consumption separately.

Secondly, we analyzed basic Action Systems compositions. The form of the analysis depends on whether the actions under investigation are considered independent or not. Thus, independent actions introduces potential for parallel behavior, which complicates the power estimation. On the contrary, if the actions are non-independent, then the power estimation is straightforward. Therefore, the critical part of the estimation process is to determine the correct order of events, and detect the parallel behavior. Therefore, the meaning of timing issues are highlighted. Finally, the topics discussed above were analyzed using example Action Systems description.

Future Work: The experiences of this study lays the foundation for the formal power model. The formal power estimation framework is created in two phases. At first, we concentrate on energy estimation and its formal model. Then we will expand the existing energy estimation framework into the power estimation model. Finally, in the gate-level analysis, we will compare this model with existing power estimation methods and simulators. The technology dependent information is included as late as possible into the model. For timing solutions, we can apply a existing Timed Actions model [8], Petri Nets [4], etc.

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] A. K. Murugavel, and N. Ranganathan, *Petri Net Modeling of Gate and Interconnect Delays for Power Estimation*, in Proc. DAC 2002, June 10-14, 2002, New Orleans, LA, USA.

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

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

[7] 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.

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