Johanna Tuominen | Juha Plosila

High Level Power Estimation

Turku Centre *for* Computer Science

TUCS Technical Report
No 623, September 2004

High Level Power Estimation

Johanna Tuominen
Juha Plosila

TUCS Technical Report
No 623, September 2004

Abstract

In a synchronous circuits, the global clock signal couples the computation and the physical time. Therefore, the clock frequency measures directly the switching activity of the system which can be used directly in the power calculation. On the contrary, in a self-timed circuits there is no notion of physical time. Thus, the operation of such system is based on ordering events between circuit components. In addition to that, the duration of given computation depends strongly on the data, which makes it difficult to use the algorithms developed for synchronous circuits directly.

An overview of selected high-level power estimation techniques is given for both synchronous and asynchronous systems. In this report, a specification of a formal high level power estimation is presented. The purpose is to include this model into our existing formal framework for system specification.

Keywords: asynchronous, power consumption, switching activity, noise

TUCS Laboratory
Communication Systems Laboratory

1 Introduction

The objective of this report is to give a short overview of high level power estimation methods for ULSI circuits. At first, we discuss power estimation in a synchronous context, where the power consumption is not localized, i.e., a significant portion of the power is consumed by a circuit elements not performing useful computation, at a given time. For example, flip-flops and clock distribution network consumes power during each clock cycle whether they are involved in the useful circuit function or not. Techniques to measure average power per clock cycle in synchronous CMOS circuits, are based on an probabilistic estimate of the switching activity in the circuit [10, 18]

The operation of a self-timed system is based on ordering events between circuit components. The absence of the global clock signal gives several advantages in terms of modularity, robustness, lack of clock-generation and clock distribution problems etc. However, this introduces an interesting concern when it becomes to measuring power consumed by a circuit. Since, even though the global clock have a lot of drawbacks, they have one big advantage, i.e. they couple computation and physical time. On the contrary, in a self-timed circuits there is no notion of physical time [13]. Therefore, it is difficult to define power consumption in general for a self-timed circuit. Furthermore, it is not possible to directly apply the power estimation techniques developed for synchronous circuits.

This report will present an overview of two high level power estimation methods for asynchronous systems. The first one is a Petri net based solution, where the switching activity is captured from the Petri net graph by tracing the token flow [13]. The second estimation method is for quasi-delay-insensitive (QDI) circuit [20]. These circuits use no timing assumptions on the delays of the operators and wires, only isochronic forks are expected [22]. This means that delays on the different branches of the fork are assumed to be identical. Finally, the future work section presents a specification of a high level power estimation for the action systems formalism [3, 4]. The formal power estimation model is included as a part of our existing framework of formal system specification. Furthermore, the high level model is extend to determine a noise estimate for the system.

The overview to the high level power estimation techniques for synchronous circuits is presented in Section 2. The power estimation in asynchronous context is discussed in Section 3. Section 4 gives specification for the future work, and finally, conclusion of the work is presented in Section 5.

2 Power Estimation in Synchronous Circuits

The amount of energy dissipated by a CMOS logic gate each time its output changes is roughly equal to the change in energy stored in the gate's output capacitance. If the gate is a part of the synchronous digital system controlled by a global clock, it follows that the average power dissipated by the gate is given by [10]:

$$P_{avg} = 0.5 \times {V^2}_{dd} \times C_{load} \times f \times E(transitions) \quad (1)$$

In Equation 1, P_{avg} denotes the average power, C_{load} is the load capacitance, V_{dd} is the supply voltage, f is the frequency of the global clock, and $E(transitions)$ is the switching activity per clock cycle. In a synchronous circuit, a node might switch several times before settling to the steady state value during a clock cycle. These spurious transitions (glitches) complicates the energy estimation of synchronous circuits. The main reason for this is that glitches are highly delay dependent, and therefore any attempt to estimate the energy properly has to include an accurate timing model [20].

2.1 Estimation of Average Switching Activity

In CMOS logic circuits, the transition rate of a node is a good indicator of the circuits susceptibility to runtime failures [18]. However the quantification of the circuit activity is difficult, because logic signals are, in general, non-periodic, and thus, have no fixed switching activity.

2.1.1 Static probabilities

Consider the case of dynamic CMOS logic. At the beginning of each clock cycle, all the gates are precharged, and gates make transitions only if their associated Boolean functions are satisfied. For example, a three-input AND-OR gate's Boolean function might be

$$(i_1 \cdot i_2) \vee (i_2 \cdot i_3) \quad (2)$$

where i_1, i_2, and i_3 are primary inputs. In this case, the expected number of transitions at the gate's output is

$$E(transitions) = 2 \times P((i_1 \cdot i_2) \vee (i_2 \cdot i_3) = 1) \quad (3)$$

where the P(x) is defined as probability that x is true, and the factor of two in the equation accounts for the reset transition during precharge [10].

To evaluate the (3) it is necessary to determine the primary input probabilities. The primary inputs are assumed to be uncorrelated, and that each is a waveform in time whose value is either zero or one, changing instantaneously at global clock edges. Assuming ergodicity, the probability of a particular input i_j being one at a given point in time, denoted p_j^{one}, is given by

$$p_j^{one} = \lim_{N\to\infty} \frac{\sum_{k=1}^{N} i_j(k)}{N} \tag{4}$$

where N is the total number of global clock cycles and $i_j(k)$ is the value of input i_j during clock cycle k [10]. Hence, the probability that i_j is zero, denoted as p_j^{zero} is

$$p_j^{zero} = 1 - p_j^{one} \tag{5}$$

The probabilities p_j^{zero} and p_j^{one} are referred as static probabilities. Note that the equation (6) is false, since the first and the second product terms are not independent.

$$P((i_1 \cdot i_2) \vee (i_2 \cdot i_3) = 1) \neq p_1^{one} p_2^{one} + p_2^{one} p_3^{one} \tag{6}$$

Instead, the equation (6) should be write in a form [10]

$$P((i_1 \cdot i_2) \vee (i_2 \cdot i_3) = 1) = P((i_1 \cdot i_2) \vee (\overline{i_1} \cdot i_2 \cdot i_3) = 1) = p_1^{one} p_2^{one} + p_1^{zero} p_2^{one} p_3^{one} \tag{7}$$

where the second equality holds because $i_1 \cdot i_2$ is disjoint from $\overline{i_1} \cdot i_2 \cdot i_3$ [10]. In this example, $i_1 \cdot i_2 \vee \overline{i_1} \cdot i_2 \cdot i_3$ is a disjoint cover for the logic function, and the terms $i_1 \cdot i_2$ and $\overline{i_1} \cdot i_2 \cdot i_3$ are referred as *cubes* in the cover [9]. The equivalent logical expression $i_1 \cdot i_2 \vee i_2 \cdot i_3$, does not represent a disjoint cover because $i_1 \cdot i_2 \cdot i_3$ is contained in both cubes $i_1 \cdot i_2$ and $i_2 \cdot i_3$.

In general, given a disjoint cover for a Boolean function of uncorrelated inputs described by static probabilities, the probability of the function evaluating to '1' can be easily determined. The procedure follows the two theorems given below, whose proof follows directly from the elementary probability [19].

Theorem 2.1 *Given any disjoint cover for a Boolean function, the probability of the function P(f=1) is equal to the sum of the probabilities of each of the cubes in the cover evaluating to a '1'.*

Theorem 2.2 *Given a logical function of uncorrelated inputs in the form*

$$f = i_{\alpha 1} \cdot i_{\alpha 2} \cdots i_{\alpha M} \cdot \overline{i_{\beta 1}} \cdot \overline{i_{\beta 2}} \cdots \overline{i_{\beta N}}$$

where $i_{\alpha j}$ are the non-negated inputs and the $\overline{i_{\beta k}}$ are the negated inputs, then

$$P(f = 1) = p_{\alpha 1}^{one} \cdot p_{\alpha 2}^{one} \cdots p_{\alpha M}^{one} \cdot p_{\beta 1}^{zero} \cdot p_{\beta 2}^{zero} \cdots p_{\beta N}^{zero}$$

2.1.2 Transition probabilities

Consider the case of static CMOS logic, a gate output can only change when its inputs change, and then only if the boolean function describing the gate evaluates differently. For example, a 2-input AND gate's output, will change between clock cycle t and $t+1$ if

$$(i_1(t) \cdot i_2(t)) \oplus (i_1(t+1) \cdot i_2(t+1)) \tag{8}$$

evaluates to 1, where $i_1(t)$, $i_2(t)$ and $i_1(t+1)$, $i_2(t+1)$ are the inputs at clock cycle t and $t+1$, respectively. The disjoint cover for (8) is

$$\begin{aligned} &(i_1(t) \cdot i_2(t)) \cdot \overline{(i_1(t+1))} \vee \\ &(i_1(t) \cdot i_2(t)) \cdot (i_1(t+1) \cdot \overline{i_2(t+1)}) \vee \\ &\overline{i_1(t)}(i_1(t+1) \cdot i_2(t+1)) \vee (i_1(t) \cdot \overline{i_2(t)})(i_1(t+1) \cdot i_2(t+1)) \end{aligned} \tag{9}$$

In this case, the input at time $t+1$ is correlated to its behavior at time t. Therefore, it is not possible to use the theorem (2.2) to evaluate the probability of (9). Hence, the transition probabilities for the transitions $0 \to 0, 0 \to 1, 1 \to 0, 1 \to 1$ have to be used [10]. As an example, the Ghosh et al. defines transition probability for p_j^{10} as follows:

$$p_j^{one} = \lim_{N \to \infty} \frac{\sum_{k=1}^{N} i_j(k)\overline{i_j(k+1)}}{N} \tag{10}$$

where N is the total number of clock cycles and $i_j(k)$ is the value of input i_j during clock cycle k. The other transition probabilities follow similarly.

For the dynamic CMOS-logic, static probabilistic calculation is applied. These probabilities can calculated directly from the transition probabilities, as shown in Equations (11) and (12) [10].

$$p_j^{one} = p_j^{00} + p_j^{01} \tag{11}$$

$$p_j^{zero} = p_j^{10} + p_j^{11} \tag{12}$$

Both static and transition probabilities are used to calculate the $E(transitions)$ for static logic circuits, as shown in Equation (13).

$$p_1^{10} \cdot p_2^{one} + p_1^{11} \cdot p_2^{10} + p_1^{01} \cdot p_2^{one} + p_1^{11} \cdot p_2^{01} \tag{13}$$

For all primary inputs, it may be assumed that successive input vectors are uncorrelated and '1' and '0' are equally likely. Thus, all transition probabilities may be assumed to be 0.25, and all static probabilities to be 0.5.

To obtain the expected switching activity in the entire circuit over all time points corresponding to a clock cycle, the probabilities of all the gates are summed together. The general delay model [8] of combinational logic is used to compute the boolean conditions that cause glitching. Thus, in some cases glitching may account for a significant amount of the dissipated power or switching activity [20].

Najm in [18] presents transition density calculation for switching activity estimation. Transition densities corresponds to average switching rates for gates in the circuit. These densities are propagated through combinational logic modules without regard to their structure. Correlations between internal lines due to the re-convergence are ignored during propagation. It is possible to take into account correlations by lumping all the modules into one large module, but in this case the information regarding the delay of the individual modules are lost. Cirit in [5] gives methods to calculate dynamic power dissipation based on approximate signal probability evaluation procedures.

3 Power Estimation in the Asynchronous Context

In synchronous circuits many gates switch without having actual input to process, because they are connected to the clock. For instance, the clock buffer must switch regularly to provide a timing reference to the circuit. The operation of a self-timed system is based on ordering events between circuit components. Therefore in an asynchronous system the computational events are a direct measure of the energy consumption. However, it is difficult to quantify the power dissipation due to the lack of unit of operation, namely, something like the clock cycle. In addition to that, the duration of given computation depends strongly on the data, which makes it difficult to use the algorithms developed for synchronous circuits directly [13].

Kudva et. al presents a Petri-net based solution, which captures the control flow of the circuit [13]. The switching activity is estimated by tracing

the token flow in the Petri net, which is described in Section 3.1. Penzes et. al are developing an energy estimation technique for quasi-delay-insensitive (QDI) circuits [20]. QDI circuit use no timing assumptions on the delays of the operators and wires, only isochronic forks are expected [22]. This means that delays on the different branches of the fork are assumed to be identical. This method is described in Section 3.2.

3.1 Power Estimation Using Petri Nets

Petri nets can be used to model a self-timed circuit [1] [6, 17]. The power estimation method, presented by Kudva et. al, describes the circuit as a structural composition of the Petri nets corresponding to the individual components. The switching activity of the circuit is then estimated by tracing the token flow in the Petri net [13]. Notion of *invocation* of a module is defined to estimate the average input/output activity at the terminals of the module by adapting the equations developed by Ghosh et. al [10]. The various transitions in the Petri net are weighted with the energy consumed by the corresponding self-timed element per invocation [13].

The circuit is partitioned into three blocks, more precisely into control block (CB), data block (DB), and predicate block (PB), in order to estimate the amount of transitions accurately in Petri nets [13]. Examples of control blocks models are XOR, Muller C-element, and call-module [23]. Models for DB and PB are shown in Figure 1. The DB represents the data-path model of the circuit, and can be modeled as a combinational circuit with a parallel delay line. The PB implements a data-dependent control flow, ie. a conditional branch. The transition on req is sent through $T-ack$ or $F-ack$ depending on whether the *sel* signal is set to '1' or '0'. The logic block has data inputs and it produces the *set* signal as an output.

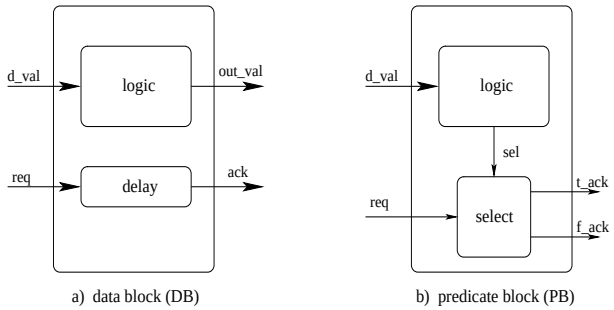

Figure 1: Models for DB and PB

[1] The reader is assumed to be familiar with the basic semantics of the Petri net.

The Petri nets have a special set of places called the *initial places* (IP) and *final places* (FP). These places correspond to the inputs and outputs of the circuit. The environment is responsible for placing the tokens in IP and removing the tokens from FP. In this method, [13] the environment is assumed to work properly, ie. without causing glitching or hazards.

The Petri net presentation of the Muller C-element is shown in Figure 2. The input wires (i1, i2) and the output wire (out) are denoted as a transitions. The places, marked with(I), denotes the IP set for the circuit. The FP set of the circuit is marked with (♯).

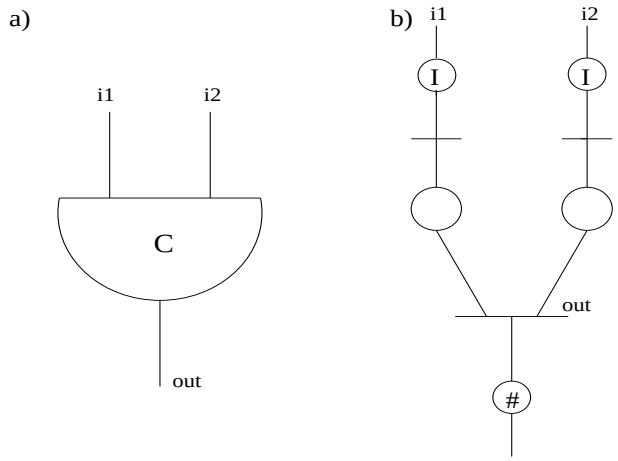

Figure 2: a) C-element b) Petri net description of a)

The *invocation* of a circuit module (DB,PB) can be defined as a pair of transitions (req, ack) in the Petri net corresponding to its initiation and completion [13]. The firing of these transitions in the Petri net indicates that the corresponding DB or PD have been activated.

3.1.1 Switching Energy in Control Blocks

Kudva et. al defines the switching activity of CBs with the help of pre-defined macro modules [13]. This method is illustrated by calculating the energy consumption of the Muller C-element. The energy consumed by the C-element on an output transition is given by:

$$E_c = E_k + E_v \tag{14}$$

where $E_v = 0.5 \times C_{load} \times V_{dd}^2$ and E_k is a constant which depends on the implementation (hence, can be obtained from technology libraries). The gate implementation of the C-element is used, that is three AND gates at

the input and OR-gate at the output. The E_v is obtained by taking account the fanout of the C-element to other modules and V_{dd} characteristics of the circuit. The E_k results as the energy consumed by the input layer of AND gates.

3.1.2 Energy Per Invocation in Data and Predicate Blocks

The average energy consumed by data and predicate blocks is shown in Equations (15) and (16), respectively [13]. The D(transitions) indicates the average number of transitions for the combinational circuit per invocation. The factor E_{dl} and E_{select} presents the energy consumed in the delay element of the DB and in the selector of the PB, respectively.

$$E_{invocation} = 0.5 \times V_{dd}^2 \times C_{load} \times D(transitions) + E_{dl} \tag{15}$$

$$E_{invocation} = 0.5 \times V_{dd}^2 \times C_{load} \times D(transitions) + E_{select} \tag{16}$$

The transition probabilities presented in Section 2 can now be applied. Hence, the key difference is the use of invocations to count the probabilities, instead of the clock cycle [13].

3.2 Energy Estimation for Quasi-Delay-Insensitive Circuit

Penzes et. al presented a energy estimation method for quasi-delay-insensitive (QDI) circuits [22]. According to these guidelines given for the energy simulation of QDI circuits, a simulator has been used to estimate the energy consumption in the different parts of the asynchronous processor (MiniMIPS) [15]. The *esim* simulator gives energy estimates within 10 % of a electrical (hspice) simulation [20]. The dynamic energy consumption is defined as shown in Equation (17). The C_i is the total capacitance at node i, n_i is the transition count of node i during measured period of time and m is the total number of circuit nodes.

$$E_{Dynamic} = \frac{1}{2} V_{dd}^2 \sum_{i=1}^{m} C_i n_i \tag{17}$$

The main difficulty of using this formula is to define n_i. In the following subsections, we will discuss on how these difficulties are solved for QDI circuits [20].

3.2.1 Production rules as a model for CMOS circuits

The operation of the simulator is based on a logical representation of the design, called the production rules (PR) [11]. A PR is a construct of the form $G \mapsto S$, where S is a simple boolean assignment and G is a boolean expression called guard of the PR [16]. For instance, a *nand* gate with inputs x and y and an output z is implemented using two PRs: $x \wedge y \mapsto c\downarrow$ and $\neg x \wedge \neg y \mapsto z\uparrow$. The PRs that set and reset the same variable, like $\neg g1 \mapsto z\uparrow$, $g2 \mapsto z\downarrow$ are implemented as one operator ($g1$ corresponds to the pull-up, while $g2$ to the pull-down circuitry of node z).

The PR set is used as a logical representation of QDI circuits. Furthermore, PRs closely correspond to the intended CMOS implementation of the circuit. The PR set of given QDI circuit could be either synthesized from a high level description or can be directly generated by the designer.

3.2.2 Energy estimation error due to glitching

In synchronous circuits the energy simulation is complicated by glitches. Hence, glitches are delay dependent and therefore any attempt to accurately model the energy due to them has to include timing model. The QDI design methodology avoids glitches by enforcing the monotonicity of signal transitions [20]. As a consequence the energy consumed by glitches is null and there is no need for timing model in this energy estimation method.

3.2.3 Energy estimation error due to *spatial dependence* (input correlation)

For all data channels, which in QDI circuits are usually encoded as dual-rail or 1-of-n signals, similar switching activity is expected. Therefore, the corresponding CMOS circuit is symmetric, in terms of transistor sizes, for each data rail within the channel. Furthermore, each activated node transitions exactly twice in 4-phase signaling. Hence, not all signals become active since some of them are mutually exclusive. For the above mentioned reasons, this type of circuit consumes the same amount of energy, independently of the data being processed [20]. One important exception are adders. For instance, a carry look-ahead adder has more load in the propagate rails than in the kill and generate rails, since for uniform input distribution a transition on a propagate rail is more likely. This generates some input pattern dependence on the consumed energy. However, when applied in a realistic context of the QDI design (input/output completions, data/control acknowledges, internal enable) this dependence is very weak [20].

For the control channels the dependence is stronger, since different values of the control might activate different parts of the circuit. For instance, in case of a regular add instruction, the two inputs are coming from the register file. However, in an add immediate instruction one input comes from the register file and the other one from the immediate bus. Depending on the implementation this could take a different amount of energy, and therefore the energy corresponding every distinct control signal of a given circuit block has to be estimated individually [20].

3.2.4 Energy estimation error due to *temporal dependence* (cycle-to-cycle dependency)

In a synchronous circuit, the switching of a signal depends not only on the data value on the current cycle, but also on the previous value of that node, causing cycle-to-cycle dependency. For QDI circuits, during operation each active node will go through two transitions, a charging and a discharging transition [22]. By the end of the cycle, each node will be at the same state as it was in the beginning of the cycle, independently of the performed operation. This property eliminates the cycle-to-cycle dependency altogether [20]. However, if state variables are implemented as a flip-flops, temporal dependency exists. Hence, if the current state and the next state has the same value, the bit will not flip. On the other hand if the two values are different, the bit will transition and as a result energy is consumed. One way to deal this is to simulate the system for the worst case, i.e., all bits changing and for the best case, i.e., no bits changing. Then the resulted energy estimate is the average value between the worst- and the best-case. All together the energy variability due to temporal or spatial dependence of input data is localized and relatively small; thus, in general it can be ignored [20].

3.2.5 Energy model - the *esim* simulator

The *esim* energy simulator estimates the dynamic energy consumption for QDI circuit using Equation 17 [20]. There are two unknown factors in the Equation 17, the node capacitance C_i and the number of transitions n_i. The transition count (n_i) is taken care of automatically by the simulator: for each actual transition, the corresponding weight is added to an energy counter. The capacitance model for PRs is shown in Figure 3. Each node in a PR set could have the following capacitive components: source / drain diffusion capacitance due to the PR's pull-ups and pull-downs (C_d), wiring capacitance (C_w), capacitance due to the keeper if the node is state holding (C_s), and

gate capacitance due to the transistors the node is driving (C_g).

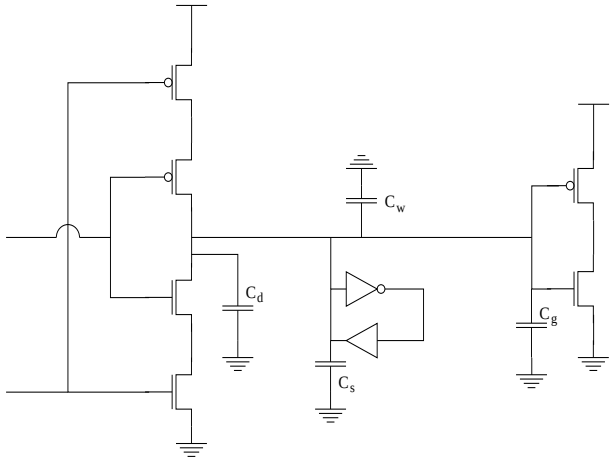

Figure 3: The *esim* capacitance model

If a sized but unwired PR set is available for simulation, all capacitive components, except C_w, can be computed with a good accuracy. The dynamic energy consumed by node i can be written as $E = E_{Cg} + E_{Cd} + E_{Cs} = K \times \sum(transistorwidth)$ [20]. Where the technology dependent constant K can be computed directly or calibrated using *hspice*. The $\sum(transistorwidth)$ is computed for each node and assigned to its corresponding *weight* variable in the data structure. If a wired up layout exists, the wire information can be added as explicit PRs into the original PR set.

The general operation is as follows [20]: Once the simulation is launched on a closed PR set, all nodes are initialized and an energy *weight* is assigned. Those PRs that are ready to be fired are collected into the ready queue, ie. those guards that are true. A step of a simulator consists of taking one PR out of the ready queue and firing it. The firing of the PR may cause other PRs to become enabled, in which case their PRs are added to the ready queue. When the PR is taken of from the ready queue, its assigned energy *weight* is added to a counter, which keeps track of the total energy consumed.

There are several energy dissipating sources that the *esim* is ignoring [20]. One of the most important one is the energy consumed due to the leakage currents. The importance of the leakage currents to the overall energy consumption is increasing due to technology scaling. Furthermore, there are two elements related to dynamic power consumption which are not captured. First, the energy due to charge-sharing, and second, the energy spent on charging and discharging the internal nodes of the transistor stacks are not accounted for.

4 Future Work

In this section, a specification of a formal high level power estimation is presented. The purpose is to include this model into our existing formal framework for system specification. The power consumption estimation process is divided into two phases, abstract and gate level power estimation. After the gate level estimation, the purpose is to develop model for a system noise levels, and include it as a part of the overall framework. Finally, a simulator for high level power and noise estimation will be implemented.

4.1 Action Systems Formalism

The action system formalism is a framework for specification and correctness preserving development of concurrent programs [3, 4]. It is based on an extended version of the guarded command language of Dijkstra [7]. The action system language includes assignment, sequential composition, nondeterministic choice, and iteration, and is defined using weakest precondition predicate transformers. The parallel behavior is modeled by interleaved actions, i.e., by two or more actions which can be executed in any order. The action systems are developed in a stepwise manner using the refinement calculus which guarantee the correctness of each transformation step [1, 2]. Furthermore, the action systems formalism is inherently well-suited for modeling asynchronous concurrent behavior. However, a set of concepts are introduced in [21], which are useful especially in circuit derivation.

4.2 Abstract Level Power Consumption Estimation

A coarse-grained power estimation is applied to the abstract level system description by the designer, i.e., the designer specifies the upper limit for the power consumption of the system. The first power consumption estimate is then defined as a constant P. After each refinement step, the value of the power consumed by the subsystems is added together. Thus, this value cannot exceed the value of the constant P. In other words, during refinement, the inequality shown in Figure 4 have to hold. Therefore, it will be necessary to add some guidelines of the amount of power consumed per particular system structures. The estimation process is illustrated in Figure 4.

4.3 Gate Level Power Consumption Estimation

The Petri net based power estimation analysis is used as a guideline for our work [13]. In order to determine the circuits switching activity a timing

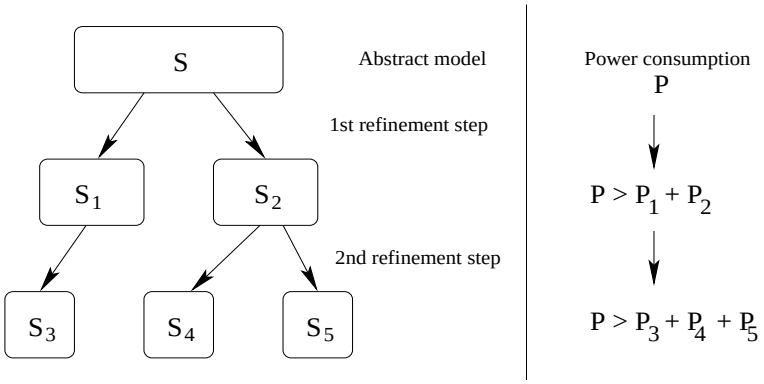

Figure 4: Abstract level power estimation flow

model will be included into the design description. The switching activity is captured from the action systems description by counting enabled events at a given time. The gate level power estimation flow is shown in Figure 5.

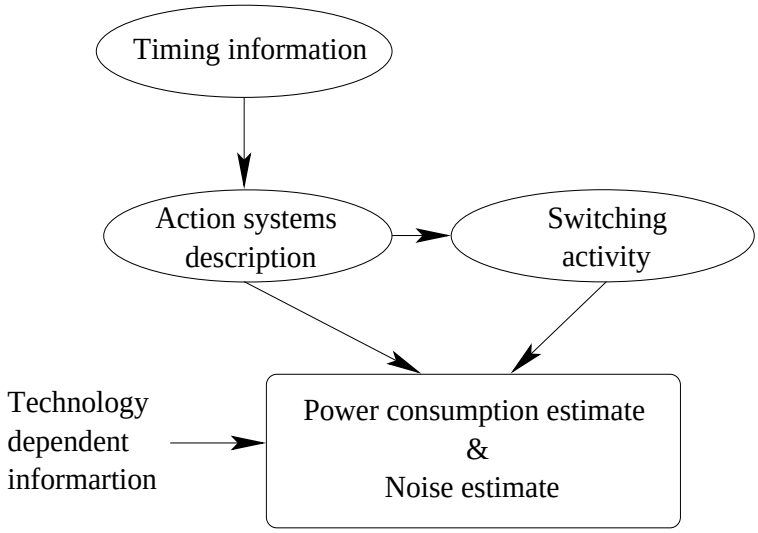

Figure 5: Gate level power estimation flow

The power dissipation per library component can be calculated from Equation 18.

$$P_{total} = P_{dynamic} + P_{static} \tag{18}$$

The dynamic energy consumption of a given node can be calculated from Equation 17, and the static power consumption per component can be obtained from technology data sheets. Furthermore, the power consumption

caused by leakage currents is added as a constant to the overall power consumption.

4.3.1 Noise estimation

Once the switching activity is determined form the action systems description, we can start focusing into the system noise levels. The purpose is to concentrate to estimate power supply noise and crosstalk. The power supply noise within a digital chip mainly originates from simultaneous switching of CMOS circuits which causes high peak current draws from the power source. Thus, high simultaneous switching activity increases the amount of power supply noise [14]. The crosstalk estimation is more complicated task. Since, the crosstalk noise is caused by the inductive and capacitive coupling between on-chip signal lines. Therefore, the crosstalk estimation have to be done using some crosstalk model [12, 24]. Furthermore, the amount leakage noise is included as a constant into the action system description.

4.3.2 Simulator

After the action systems framework for power dissipation and noise is ready, the purpose is to build simulator for high level power estimation. The programming language chosen for the implementation will be some object-oriented language, i.e., C++ or Java. The actions systems description can be turned into Petri nets description in order to make the software implementation easier.

5 Conclusion

An overview of selected high level power estimation techniques for ULSI circuits is presented. In a synchronous circuit, the power consumption is measured as an average power per clock cycle. Then the switching activity of the circuit is calculated using static or dynamic probabilities. On the contrary, in the self-timed circuit there is no notion of physical time. Therefore, it is difficult to define the power consumption for an asynchronous circuit. Furthermore, the techniques presented for the synchronous power estimation are not directly applicable.

A specification for high level power and noise estimation in action systems context is presented. The power estimation is divided into two phase, abstract and gate level power analysis. The formal framework for power estimation can be develop further to the system noise estimation. Finally, a high-level power and noise estimation will be implemented.

References

[1] R. J. R. Back. "On the Correctness of Refinement Steps in Program Development", *Ph.D Thesis* Department of Computer Science, University of Helsinki, Helsinki, Finland, 1978. Report A-1978-4.

[2] R. J. R. Back. "A Calculus of Refinements for Program Derivations", *Acta Informatica*, 25(6):593-624, 1988.

[3] R. J. R. Back and R. Kurki-Suonio. "Decentralization of Process Nets with Centralized Control", in *Proc. of the 2nd ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, 1983, pp. 131-142.

[4] R. J. R. Back and K. Sere. "From Modular Systems to Action Systems", in *Formal Methods Europe'94*, Spain, October 1994, *Lecture Notes in Computer Science* Springer - Verlag, 1994.

[5] M. A. Cirit. "Estimating Dynamic Power Consumption of CMOS Circuits", in *Proc. ICCAD* November 1987, Santa Clara, CA, USA, pp. 534 - 537.

[6] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. " Logic Synthesis of Asynchronous Controllers and Interfaces", *Springer*, 2002

[7] E. W. Dijkstra. " A Discipline of Programming", *Prentice Hall Series in Automatic Computation*, Prentice Hall, 1976.

[8] S. Devadas, K. Keutzer, and J. White. "Estimation of Power Dissipation in CMOS Combinational Circuits", in *Proc. Custom Integrated Circuit Conference*, May 1990, Boston, MA, USA, pp.19.7.1- 19.7.6.

[9] D. D. Gajski. " Principles of Digital Design", *Prentice-Hall*, 1997, NJ, USA.

[10] A. Ghosh, S. Devadas, K. Keutzer, and J. White. "Estimation of Switching Activity in Combinational and Sequential Circuits", in *Proc. ACM/IEEE 29th Design Automation Conference*, June 1992, pp. 253-259.

[11] M. Hanna, and E. Grinspun. " A Production Rule Simulation", *Caltech Computer Science Technical Report*, 2000.

[12] H. Kawaguchi and T. Sakurai. "Delay and Noise Formulas for Capacitively Coupled Distributed RC Lines" in *Proc. of Asia and South Pacific Design Automation Conference*, Yokohama, Japan, 1998.

[13] P. Kudva, and V. Akella. "A Technique for Estimating Power in Asynchronous Circuits", IEEE, 1994.

[14] P. Liljeberg, J. Tuominen, S. Tuuna, J. Plosila and J. Isoaho. "Self-Timed Approach for Noise Reduction in NoC Interconnects", Interconnect Centric Design for Advanced SoC and Noc, Kluwer Academic Publishers, Boston, April 2004, Chapter 11, pp. 285-313.

[15] A. J. Martin, et al. "The Design of an Asynchronous MIPS R3000 Microprocessor", in *Proc. of the 17th Conf. on Adv. Research in VLSI*, IEEE Computer Society Press, 1997, pp. 164-181.

[16] A. J. Martin. "Synthesis of Asynchronous VLSI Circuits", in *Formal Methods for VLSI Design*, ed. J. Staunstrup, North-Holland, 1990.

[17] T. Murata. "Petri Nets: Properties, Analysis and Applications", in *Proc. the IEEE*, vol. 77, no. 4, April 1989.

[18] F. N. Najm. "Transition Density: A New Measure of Activity in Digital Circuit", in *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 2, February 1993.

[19] A. Papoulis. " Probability, Random Variables and Stochastic Prosesses", *McGraw Hill*, 1984.

[20] P. I. Penzes, and A. J. Martin. "An Energy Estimation Method for Asynchronous Circuits with Application to an Asynchronous Microprocessor".

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

[22] J. Sparso, and S. Furber. "Principles of Asynchronous System Design - A Systems Perspective", editors, *Kluwer Academic Publishers*, Boston, 2001 .

[23] I. Sutherland. " Micropipelines", *Communications of the ACM*, June 1998, *The 1988 ACM Turing Award Lecture.*

[24] S. Tuuna and J. Isoaho. "Estimation of Crosstalk Noise for On-Chip Buses" in *Proc. of 13th International Workshop on Power and Timing Modeling, Optimization and Simulation*, Turin, Italy, September 2003.

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 952-12-1416-3
ISSN 1239-1891