Petri Net Modeling of Gate and Interconnect Delays for Power Estimation

Ashok K. Murugavel and N. Ranganathan
Dept. of CSEE and Center for Microelectronics Research
University of South Florida
Tampa, Florida 33620

ABSTRACT

In this paper, a new type of Petri net called Hierarchical Colored Hardware Petri net, to model real-delay switching activity for power estimation is proposed. The logic circuit is converted into a HCHPN and simulated as a Petri net to get the switching activity estimate and thus the power values. The method is accurate and is significantly faster than other simulative methods. The HCHPN yields an average error of 4.9% with respect to Hspice for the ISCAS '85 benchmark circuits. The per-pattern simulation time is about 46 times lesser than PowerMill.

1. INTRODUCTION

As the integrated circuit design moves into deep sub-micron technology, it is possible to pack several million transistors in a small die area. Besides the traditional goals of area and time, power has become an important and critical parameter in the design of circuits and systems. Power estimation techniques, in general, represent a trade-off between accuracy and speed. With the advent of wireless and mobile high performance computing platforms and the limited operational lifetime of a battery, low power designs are required. With higher power consumption, packaging and cooling costs increase and these issues need to be addressed. Thus, it is obvious that with increased circuit integration, power dissipation assumes greater importance. An important open problem in the area of power estimation at the gate level, is the modeling of real delays (both intrinsic gate and interconnect delays). Until, the advent of deep sub-micron (DSM) technologies, the overall path delay between gates was dominated by gate delays, and the bus/interconnect delays could be ignored. In deep sub-micron IC's the interconnects introduce critical problems on the signal integrity and the interconnect delays dominate over the gate delays and so cannot be ignored. It has been observed that the existing power estimation methods are fast and accurate, but, do not consider real delay models that include interconnect effects [3].

Numerous works have appeared in the literature on average power estimation at the gate level under a zero delay model [9, 1]. The zero delay models ignore glitches completely, and can thus lead to underestimation of power. In [4, 12], techniques for real delay power estimation considering only the gate delays has been proposed. In deep sub-micron IC's the interconnects introduce critical problems on the signal integrity and the interconnect delays dominate over the gate delays and so cannot be ignored. It has been observed that the existing power estimation methods are fast and accurate, but, do not consider real delay models that include interconnect effects [3].

In [11], a fast simulation algorithm based on a standard logic simulator to estimate the gate-level power has been presented. In [2], a new accurate and efficient method to estimate the switching activity of a circuit has been proposed. The method involves a novel event driven delay simulation using a time parallel simulator. A fast and accurate glitch modeling and estimation technique has been proposed in [6], the average power estimation of the circuit is based on a toggle count mechanism has been presented. The toggle information obtained is not exact for circuits with high depth. This leads to discrepancies in the power estimate. The methodology has been shown to be faster than Hspice but with the loss of accuracy. The methodology has also been compared with PowerMill. The first RT-Level power estimation technique that includes the interconnect effects was presented in [3]. The model for switching activity estimation is based on the relationship between switching activity and entropy. The RT-level technique has an estimation error of about 25.8%.

In this paper, a new technique based on simulation is presented that models switching activity based on a complete delay model (considering both gate and interconnect delays). In order to model switching activity, we propose a new type of Petri net called the Hierarchical Colored Hardware Petri net (HCHPN) which incorporates accurately the spatial and temporal correlations with real delays. The logic circuit is first modeled as a Gate Signal Directed Acyclic Graph (GSDAG), which is then converted into the corresponding Hierarchical Colored Hardware Petri net (HCHPN) and simulated as a Petri net to get power values. While providing the same accuracy levels compared to existing simulators, the proposed strategy requires significantly less per-pattern simulation time.

2. PRELIMINARIES

In this section, we provide some definitions and terminology related to Petri nets, required for describing the proposed approach. A Petri net model consists of a set of *places* (P) and a set of *transitions* (T) [5], in which the places correspond to the states of the model and the transitions represent the actions related to the states of the model. The places and transitions are connected by a set of directed *arcs* (A). An arc can only connect a transition with a place or vice-versa. An arc directed from a place to a transition is called an input-arc and an arc from a transition to a place is called an output-arc of the transition. A transition is said to be *enabled*

DAC 2002, June 10-14, 2002, New Orleans, Louisiana, USA

if there are enough tokens in each of the input places as specified by the arcs connecting the input places to the transition. An enabled transition can *fire* if the other conditions associated with the transition are satisfied. The initial state of the Petri net is called its *initial-marking*. The current state of a Petri net is the distribution of tokens to the places in the Petri net.

The three basic elements of CPNs are places, transitions and arcs. The color set (Σ) determines the types, operations and functions that can be used in the net. It is assumed that the color sets have at least one element each. The arcs have expressions attached to them called *arc-expressions* (E). A transition in a CPN is enabled if it is possible to bind the variables in such a way that the arc-expressions of all the input arcs evaluate when tokens are present at the corresponding input places. The transition can, in addition, have Boolean expressions called *Guard-expressions* (G), that are needed to evaluate to "true" in-order that the transition fires. The *node function N* maps each arc into a tuple where the first element is the source node and the second element is the destination node, the *color function C* maps the place p to a color set, the *guard function* (G) maps the transition t to the boolean function and the *arc expression function* (E) maps the arc, a to an expression. The *initialization function* (I) specifies the initial state of the Petri net. A CPN, can be defined mathematically as in [5],

$$CPN = (\Sigma, P, T, A, N, C, G, E, I) \tag{1}$$

where, (i) Σ is the color set, (ii) P is a finite set of places, (iii) T is a finite set of transitions, (iv) A is a finite set of arcs such that, $P \cap T = P \cap A = T \cap A = \oslash$, (v) N is a node function, defined from A into PxT $\cup$ TxP, (vi) C is the color function defined from P into Σ, (vii) G the guard function, (viii) E is an arc expression function, (ix) I is the initialization function.

It is difficult to model large systems as a single large CPN. To alleviate this problem the concept of Hierarchical Colored Petri nets [5] is used. The following definitions and terminology from [5], are required for defining the HCHPN.

Definition 1 [5]: A *multi-set s*, over a set S, is a function from S to N, where N is the set of Natural numbers. If $a \in S$ then $s(a)$ is the number of appearances of element a in S. A_{MS} is the set of all multi-sets over S, defined as,

$$\sum_{a \in S} s(a)a \tag{2}$$

Definition 2 [5]: A Time set (TS) is defined as the set of all non-negative real numbers, $TS = x \in R$, where x represents the time instant and R is the set of all real numbers.

Based on the above, we propose a new type of Petri net called the HCHPN and develop some definitions and terminology required for modeling switching activity.

3. HCHPN BASICS

The concepts of the Hardware Petri net (HPN) [10] and Hierarchical Colored Petri net (HCPN) [5] can be modified to define a new kind of Petri net which we call as the Hierarchical Colored Hardware Petri net (HCHPN). In order to obtain the switching activity of a circuit under a real delay model, we need to add the factor of time into the Petri net. The features of HCHPNs are: (i) certain restricted places can store only one token at any time and a newly arriving token always overwrites the existing token, (ii) when a transition fires, the associated gate function is evaluated, (iii) besides the weights on the arcs, functions can be defined on the arcs as additional firing conditions; delays for the gates and the interconnects can be specified on the transitions as firing conditions, (iv) each token has a time stamp and is called a timed token, which is available only after the clock time of the simulator is greater than or equal to the time stamp on the token.

The HCHPN has a set of *substitution transitions*. The substitution transition allows the embedding of a small or sub-Petri net within a larger Petri net for higher level modeling. When the substitution transition needs to be fired, the underlying Petri net needs to be evaluated in order to determine the outcome of the substitution transition. We relate the individual CPN to a substitution transition, the individual non-hierarchical CPN in a HCHPN is called a *page*. The input and output places of the page are called *port nodes*. The port nodes specify the type and number of tokens that can be introduced into the page. Each of the substitution transitions is called a *super node* and the page that contains these super nodes are called a *super page*. In a super page, the places that are connected to the super nodes (substitution transitions) are called *socket nodes*. The socket nodes of a super node (substitution transition) are related to the port nodes of the page through the port assignment function. The HCHPN has in addition to the substitution transitions, a set of transitions called *gate transitions*. The gate transitions are present only in the sub-nets/pages of the HCHPN. An important function added to the HCHPNs is time. With the introduction of time to each of the tokens we can model highly complex real time systems. The token in a HCHPN is called a colored token. The colored token is a 3-tuple (p,d,t) where p is the current position (place) of the token $p \in P$, d is the color set of the token and t is the time of the firing of the token. The HCHPN consists of a set of places called *restricted places*, than can hold only a finite number of tokens. To impart additional conditions during the firing of a transition, functions can be added to the transitions, which are called *code segment functions*. The HCHPN can be defined mathematically as follows,

Definition 3: A HCHPN is defined as a 10-tuple,

$$HCHPN = (PG, S, GT, PO, PA, PR, TI, TO, TC, RP) \tag{3}$$

where,

(i) PG is a finite set of pages such that (a) each page $pg \in PG$ is a non-hierarchical CPN and, (b) none of the pages have any net element in common,

(ii) $S \subseteq T$ where S is a set of substitution transitions (super nodes),

(iii) ($GT \subseteq T$ and $GT \cap S = \emptyset$), where GT is the set of gate transitions,

(iv) $PO \subseteq P$ is the set of port nodes, the port nodes are the input and output places of a page,

(v) PA is a port assignment function, which defines the relation between the socket nodes of the substitution transition and the port nodes of the page,

(vi) $PR \in PG_{MS}$ is a multi-set of prime pages; it determines the number of instances each individual pages has,

(vii) TI is the time set defined in **Definition 2**,

(viii) TO is the set of all possible colored tokens, i.e., all 3-tuples (p,d,t), where $p \in P$, d is the value of the token and $t \in TI$,

(ix) TC is the code segment function. It is defined from T into the set of functions,

(x) $RP \subset P$ is a set of restricted places, which can hold only a finite number of tokens in them.

The 10-tuple (PG, S, GT, PO, PA, PR, TI, TO, TC, RP) specifies the structure of the HCHPN. The following definitions represent the various characteristics of the HCHPN.

Definition 4: A *binding B(t)* of a transition t is a function defined on Var(t) as, $(i)\forall v \in Var(t) : s(v) \in Type(v)$, $(ii)G(t) < s >$, where, B(t) denotes the set of all bindings for t as defined by the function s. $G(t) < s >$ denotes the evaluation of the guard expression G(t) in the binding b.

Definition 5: A *token element* is a tuple (p,c,t) where $p \in P$, $c \in C(p)$ and $t \in TI$, while a binding element is a pair (t,b), where $t \in T$ and $b \in B(t)$. The set of all token elements is denoted by TE while the set of all binding elements is denoted by BE.

Definition 6: A *state* is defined as a multi-set of colored tokens with a time-stamp. S is the set of all possible states. A *marking* is a multi-set over TE without the time-stamp. The initial marking M_0 is the marking which is obtained by evaluating the initialization expressions:

$$\forall (p,c,t) \in TE : M_0(p,c,t) = (I(p))(c)(t) \tag{4}$$

The set of all markings is denoted by M.

Definition 7: *Event* (t,a,b) specifies the possibility of the firing of a transition t by removing tokens from the input places a and adding tokens to the output places b. The set of all events is the Event set ES.

Definition 8: A step S is *enabled* in a marking M iff

$$\forall p \in P : \sum_{(t,b) \in Y} E(p,t) < b > \leq M(p) \tag{5}$$

Let the step S be enabled in the marking M. When (t,b) $\in$ S, we say that t is enabled in M for binding b. When $(t_1,b_1,),(t_2,b_2) \in Y$ and $(t_1,b_1) \neq (t_2,b_2)$ we then say that (t_1,b_1) and (t_2,b_2) are concurrently enabled.

After the transition is enabled, it is not necessary that the transition must fire, this is because of the time factor involved, which leads to the definition of enabling time.

Definition 9: The *enabling time* E of an event (t,a,b) is the maximum of the time-stamps of the tokens consumed by transition t from the input place set a.

$$E(t,a,b) = max_{(p,d,t) \in TO^x} \tag{6}$$

Definition 10: If a transition t is enabled and the global clock is past the enabling time, it can *fire*. If more than one transition can fire at the same time, the priority of the transition firing is given such that the transition that was enabled first gets to fire first. The firing of transition t changes the marking **M** to marking **M'** such that,

$$M'(p) = \begin{cases} M(p) \pm aw & if(p,t) \in A, \\ M(p) & otherwise \end{cases}$$

where $M(p)$ and $M`(p)$ are the marking of place p before and after the firing of transition t. The value aw is the weight of the arc from place p to transition t.

Definition 11: The *firing delay* is defined as the difference between the firing time and the time-stamp of the token being produced as an output from a transition.

4. REAL DELAY MODELING

To model switching activity estimation using Petri nets, we first transform the gate-level combinational circuit into a GSDAG. We illustrate this using the example circuit c17, an ISCAS circuit with 6 NAND gates, shown in Figure 1.

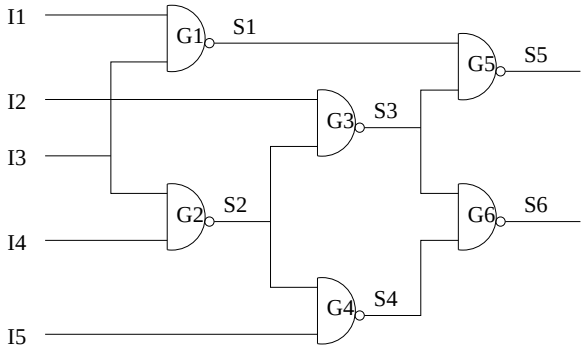

Figure 1: c17 ISCAS Combinational Circuit

The circuit has 6 gates marked {G1, G2, ..., G6} and we are interested in capturing the switching activity at the outputs of the gates marked as { S1, S2, ..., S6}. We need to determine the number of times the output of a gate switches from either 0 to 1 or 1 to 0. First, we represent the circuit as a DAG where each gate in the circuit is represented as a node and each signal in the circuit as an arc. The DAG thus obtained is converted into a GSDAG structure, by introducing additional nodes corresponding to the signals, {S1, S2, ..., S6} while maintaining connectivity by introducing arcs. The GSDAG for the circuit shown in Figure 1 is given in Figure 2. To represent multiple fan-outs, we introduce a node for each output fanning out from the gate. For instance, the output {S2} from gate {G2} is represented by two fan-out nodes {S21} and {S22} in the GSDAG. The GSDAG thus obtained can be represented as a Petri net by replacing the gates in the GSDAG by transitions and the signals by places. This Petri net does not contain the delay information of the circuit. The Petri net equivalent for the c17 circuit in Figure 1 is given in Figure 3.

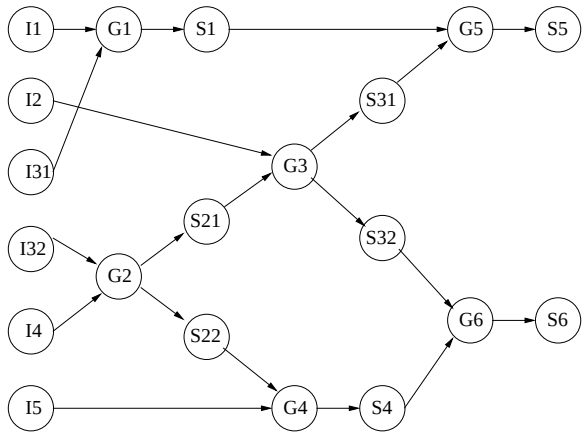

Figure 2: GSDAG structure of the c17 circuit

The important challenge in representing a synchronous system as a Petri net is in obtaining an exact correspondence between the propagation of signals in the circuit and the shifting of tokens in the Petri net. In a Petri net, when a transition fires, a token from each input place to the transition is consumed and a token is introduced into each place originating from that transition. However, in a logic circuit, the logic signal changes value and does not disappear. The propagation of signals in the circuit can be modeled by using a Colored Petri net (CPN) and defining two different tokens labeled {one} and {zero}, to specify the two different states of a

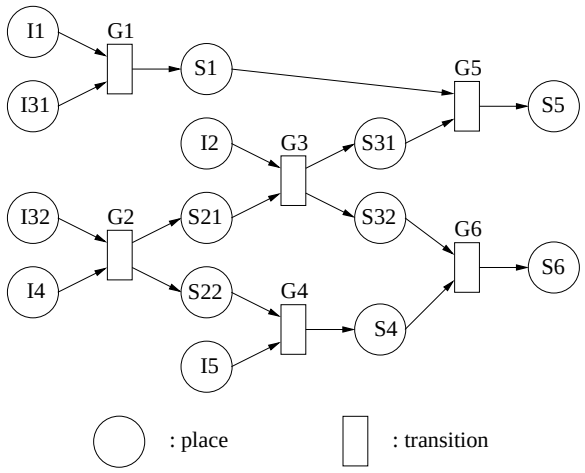

Figure 3: Petri net structure of the c17 circuit

logic signal. However, the CPNs do not allow the modeling of inertial gate delays and interconnect delays in a circuit. It will be seen that the HCHPN introduced in the last section enables the accurate modeling of the various aspects of a digital circuit.

The Petri net obtained from the GSDAG is transformed into a HCHPN by adding type, color, hierarchy and delay information. A place in a HCHPN has a type function specifying the type of tokens the place is associated with. While in a Petri net, all places are of the same type which can store any number of tokens, in a HCHPN there are two kinds of places: (i) unrestricted places and (ii) restricted places. The restricted places can store only one token at any point in time, which is useful in modeling interconnects. Fanning out inputs are replaced by an input place and a transition, to get the required number of tokens. Each transition in the Petri net structure is replaced with a substitution transition to perform the action corresponding to the gate that is modeled by the transition. The places in the Petri net representing signals are replaced by restricted places in the HCHPN. The output places of the Petri net are not converted into restricted places. The interconnect delay values are added to the arcs of the HCHPN to incorporate the delay values into the net. Figure 4 corresponds to the HCHPN equivalent of the circuit in Figure 1. If the Petri net obtained from the GSDAG is large, it needs to be segmented.

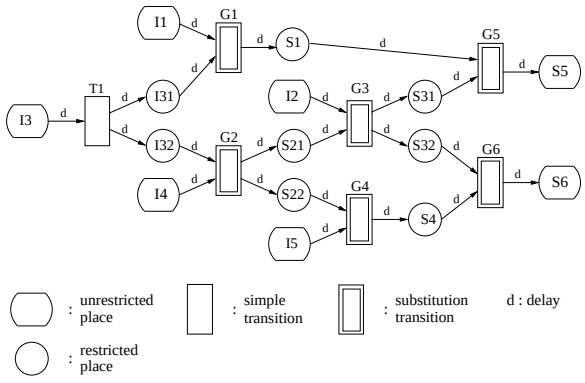

Figure 4: HCHPN structure of the c17 circuit

To model the gate of a circuit accurately, we need to (i) capture any glitches in the inputs, (ii) model the working of the gate, (iii) capture the switching activity of the gate and (iv) handle temporal and spatial correlations. The modeling of a 2-input NAND gate is shown in Figure 5. One can use this example to similarly model other types of gates. The inputs and outputs of a HCHPN must be places. In Figure 5, places {PI1, PI2} are the input places and the place {PO} is the output place of the NAND gate. Glitches are of two types: (a) generated glitches, (b) propagated glitches. The generated glitches are due to the different arrival times of the input signals. The glitches already present in the input signals are further propagated by the gate and these are called propagated glitches. The input glitches are accurately captured in our model, by using the structure consisting of the set of places {P0, P1, P2} and the transitions {T0, T1}. The places, {P0, P1, P2} are restricted places that can store only one token at any time. A newly arriving token always overwrites the existing token. The *gate transition T2* models the gate function. The delay associated with the gate is incorporated in the form of a special data structure associated with the transition {T2}. An assumption made in this model is that the delay of the gate is independent of the input pattern and is always a constant. The gate function is given as a logical expression attached to the transition {T2}, as a function of the input tokens to the transition. The logical function is a code segment function (TC) defined in Definition 3 in Section 2.

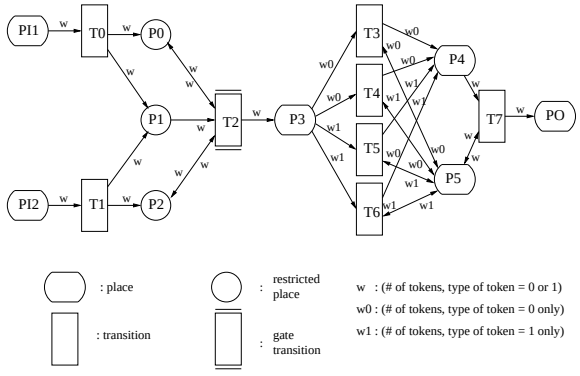

Figure 5: HCHPN structure of the NAND gate

The switching activity can be captured based on the previous and the current states of the gate. The current state of the gate is specified by the output token from the transition {T2}, stored in place {P3}. The previous state of the gate is the token stored in place {P5}, this token is initialized to 'zero' at the start of the simulation. The transitions {T3, T4, T5, T6} are used to distinguish the power consuming/dissipating transitions from the other transitions: (i) transition {T3} fires if the output of the gate remains at 0; (ii) transition {T4} fires if there is a switch from a 1 to a 0 (power dissipating transition); (iii) transition {T5} fires if there is a switch from a 0 to a 1 (power consuming transition); (iv) transition {T6} fires if the output of the gate remains at 1. The output tokens from the transitions {T3, T4, T5, T6}, are given to places {P4} and {P5}. It should be noted that in the HCHPN, *the transitions fire in the order in which they were enabled.* The temporal correlations are modeled using places {P4, P5} and transition {T7}. The arcs that connect the transitions {T3, T4, T5, T6} with places {P3, P4, P5} can only carry specific tokens. The other arcs in a HCHPN have arc weights specifying the type and the number of tokens that can be carried by the arcs. Thus, we have modeled the complete working of the NAND gate. The other gates such as *NOR, AND, OR, NOT, BUFFER and XOR* were also modeled similarly.

Now that we have seen the modeling of c17 as a HCHP-net, the working of the HCHP-net is shown here as a flow of tokens. The super page of the c17 HCHP-net is specified by Figure 4. The super page passes the tokens between the substitution transitions and handles the fanout from a gate. The output token from the substitution node is given to the places that are connected to the substitution transition. The tokens from the places are given to the sockets of other substitution transitions or the places themselves are port nodes of the super page. The sockets in the super page are of the same type as the port nodes of the page. The tokens are thus passed between the super page and the page. Since c17 consists of only NAND gates, the NAND gate in Figure 5 is used to explain the working of the HCHP-net. The port nodes of the NAND gate (page) are {PI1, PI2, PO}, {T2} is the gate transition, {P0, P1, P2} are restricted places and {P3, P4, P5} are unrestricted places of the HCHP-net. The tokens in the port nodes are fired by transition {T0, T1} into the restricted places {P0, P1, P2}. The restricted place {P1} ensures that the glitches that are smaller than the propagation time of the gate are not propagated through to the output. The gate transition {T2} has a code segment attached to it that specifies the working of the gate based on the token types that are input into the transition from the restricted places {P0, P2}. The transitions {T3, T4, T5, T6} are used to determine the type of transition on the output based on the current output from the gate transition and the previous output from the gate transition. The firing of these transitions are noted and the firing of transition {T4, T5} signify a switch in the output. The switching activity of all the gates are added to determine the average switching activity of the circuit. The previous state of the gate is stored in the unrestricted place {P5}. The transition {T7} puts the current token in the output place {PO} that is passed on to the super page through the socket node of the super page.

The HCHPN thus modeled is simulated in order to obtain the switching activity of the circuit. To calculate the toggle information, we need to calculate the various instances at which the gate switches from either 0 to 1 or from 1 to 0. A transition fires in a HCHPN only when an event occurs, i.e., the gate switches. The information obtained from the transition firing specifies the number of times the transition occurs at the output of a gate. Thus, we have a direct relationship between the transition firing in a Petri net and the switching activity of the circuit.

5. DELAY MODELING

The gates in a circuit can switch multiple times before settling to the correct output value, which leads to more switching activity. The main contributor of this unnecessary switching is the signal delays due to the different circuit paths. To obtain a generalized delay model for deep sub-micron (DSM) circuits, the delays of both the gates and the interconnects have to be taken into consideration as the interconnect delays dominate over the gate delays in DSM circuits. Thus, in the real delay modeling of a circuit, the total delay of the gate is calculated as the sum of (i) intrinsic gate delay (T_{GD}) and (ii) gate load delay (T_{RD})as $t_d = T_{GD} + T_{RD}$ [7]. The *intrinsic gate delay* is the delay due to the physical devices like transistors that constitute the gate. We calculate the intrinsic gate delay assuming an infinite load at the output of the gate using Hspice. The *gate load delay* is the delay due to the RC network connected to the gates output.

There are different methods to calculate the delay of the gate load due to the RC network at the output of a gate. In this work, we use a Thevenin equivalent circuit as shown in Figure 6, to model the interconnect and thus, the delay of the signal in the circuit. The capacitance at node A and node B are known from the layout and so is the resistance. In [7], a one segment Π model has been proposed to model the gate output while still considering the resistance shielding effects. The model approximates the load by considering only the first three moments of the driving point admittance. This model approximates the entire interconnect load with a Π circuit. The delay formula [7] is given as,

$$T_{RD} \approx \frac{1}{|s_1|} \left| ln \left(\frac{(1+s_1R_1C_2)(e^{T_R|s_1|}-1)}{b_2 s_1^2 (s_1-s_2)(1-v_{th})T_R} \right) \right| \quad (7)$$

where $b_1 = R_S(C_1+C_2)+R_1C_2$, $b_2 = R_SR_1C_1C_2$, and $s_{1,2} = \frac{-b_1 \pm \sqrt{b_1^2-4b_2}}{2b_2}$, R_S is the series source resistance as seen in Figure 6, T_R is the source ramp input rise time, v_{th} is the voltage, R_1 is the resistance, C_1, C_2 the capacitances in the Π model used for modeling the interconnects as shown in Figure 6. This model is incorporated into our work.

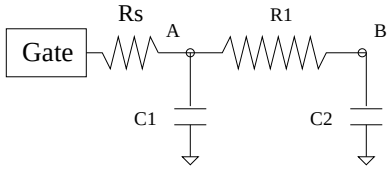

Figure 6: Thevenin equivalent circuit

Figure 7 gives the block diagram for the power estimation tool using HCHPNs. The input to the tool is the gate-level net-list, and the RC estimates (layout-level) at each node, the *intrinsic gate delay* estimates for each of the gates and the input vectors to be simulated. For, a gate delay model, we need only the gate-level net-list and the gate delay as inputs to the tool. The first step in the tool is the conversion of the input net-list into a GSDAG. The GSDAG is then converted into a HCHPN with the necessary code segments, substitution transitions, color functions, hierarchy and delay elements. The delay elements can be calculated based on the capacitance and resistance estimates obtained from the layout-level for each of the nodes in the circuit and equation 7. We have modified the Design/CPN tool [8], to perform the simulation. The toggle information thus generated is the switched capacitance of the circuit, it is used to calculate the power estimate as shown in Figure 7.

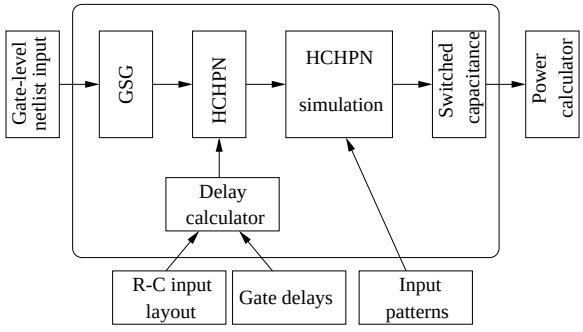

Figure 7: Block diagram of power estimation tool using HCHPNs

6. EXPERIMENTAL RESULTS

The experimental results for the ISCAS '85 combinational benchmark circuits. The ISCAS circuits consist of the basic gates like

AND, OR, NAND, NOR, NOT, EXOR, BUFF. The gates with larger number of inputs have been modified to include gates with only two, three and four inputs. The circuits were synthesized using MOSIS 0.4μ standard cell CMOS technology.

Since, there are no standard input vector sets available for these benchmark circuits, we generated a random pattern of 10,000 vectors for use with PowerMill and HCHPNs. Due to the time constraint placed by Hspice, it was not possible to simulate a large sequence of 10,000 vectors. Hence, a smaller sequence of 100 vectors was used to calculate the time taken per vector for all the three models in addition to the 10,000 vector sequence for HCHPNs and PowerMill. The rise and the fall times of the ramp input given to Hspice and PowerMill was 0.125*ns*. The intrinsic gate delays for the HCHPNs were calculated using Hspice. The HCHPNs for large circuits need to be segmented, such that the number of arcs between the segments is kept to a minimum. The simulations were performed on a Sun Ultra 10 with 128 M RAM. The switching activity at each gate is used to calculate the power estimate at that gate, thus, an accurate power estimate of the circuit can be obtained. We used Design/CPN [8], a Colored Petri net simulation and analysis tool for performing the simulation of the HCHPNs. It supports the use of complex color sets and arc expressions for CPNs as explained in Section 3 through the use of a functional programming language *Standard ML (SML)*. The SML language is used to specify the delays of the gates in the gate transition and the interconnect delays in the arc functions. For a more detailed information about Design/CPN the reader is directed to [8]. The delay values can also be estimated based on a wire length estimation technique as proposed in [3]. Experiments were performed for a gate interconnect delay model, where in both the intrinsic gate delay and the interconnect effects are taken into account to calculate the power estimate.

Table 1: Accuracy of the HCHPN modeling

Circuit	Hspice (in uW)	PowerMill (in uW)	HCHPN (in uW)	% error over Hspice	
				PowerMill	HCHPN
C432	918.0	986.4	992.7	7.4	8.1
C449	1385.9	1233.2	1298.5	11.0	6.3
C880	608.3	569.1	634.8	6.4	4.4
C1355	630.0	651.4	652.3	3.3	3.5
C1908	422.0	449.5	441.5	6.5	4.6
C2670	448.0	492.1	458.4	9.8	2.3
C3540	719.7	735.6	759.0	2.2	5.5
C5315	1098.0	1126.3	1040.6	2.5	5.2
C6288	4509.0	4898.9	4329.3	8.6	4.0
C7552	7902.7	7491.0	7528.6	5.2	4.7
Average				6.3	4.9

Tables 1 and 2 compare the accuracy and speedup for modeling both gate and interconnect delays. It can be seen that the HCHPNs have an average speed-up of 46 times over PowerMill. The power estimates are within 4.9% of that of Hspice, while PowerMill results are within 6.3% of that of Hspice. A comparison of the performance of the real-delay models reported in [6] was not possible since the results were presented there in terms of charge with Joule as unit.

7. CONCLUSIONS

In this paper, we have introduced a new fast and accurate real delay simulator for power estimation at the gate-level using the proposed Hierarchical Colored Hardware Petri nets. The modeling is exact and captures all the correlations in the circuit. We have presented a model for real delay based power estimation considering both the intrinsic gate delays and the interconnect delays. The model considers only complete transitions, all partial or incomplete transitions are ignored. The state explosion problem of Petri nets also exists with this modeling and will need to be addressed when modeling huge circuits. Currently, we are investigating the modeling of sequential circuits using HCHPNs.

Table 2: Simulation times

Circuit	Hspice (in ms)	PowerMill (in ms)	HCHPN (in ms)	Speed-up of HCHPN	
				Hspice	PowerMill
C432	19180	138.5	2.8	6850	49
C449	31806	197.0	4.6	6914	42
C880	49258	316.2	7.3	6747	43
C1355	85286	423.6	11.7	7289	36
C1908	105286	558.0	13.6	7741	41
C2670	139720	991.7	19.8	7056	50
C3540	205751	1409.7	25.9	7944	54
C5315	486910	3019.4	68.0	7160	44
C6288	794164	5380.0	106.1	7485	50
C7552	929651	7753.7	143.5	6478	54
Average				7166.4	46.3

8. REFERENCES

[1] S. Bhanja and N. Ranganathan. Dependency preserving probabilistic modeling of switching activity using bayesian networks. In *Proc. of Design Automation Conf.*, pages 209–214, 2001.

[2] M. Buhler, M. Papesch, K. Kapp, and U. G. Baitinger. Efficient switching activity simulation under a real delay model using a bit-parallel approach. In *Proc. of the EURO-DAC*, pages 459–463, 1999.

[3] K. M. Buyuksahin and F. N. Najm. High-level power estimation with interconnect effects. In *Proc. of the Intl. Symp. on Low Power Electronic Devices*, pages 197–202, 2000.

[4] T.-L. Chou, K. Roy, and S. Prasad. Estimation of circuit activity for static and domino cmos circuits considering signal correlations and simultaneous switching. *IEEE Trans. on Computer Aided-Design*, 15(10):1257–1265, Oct. 1996.

[5] K. Jensen. *Colored Petri Nets: Basic Concepts*, volume 1. Springer, second edition, 1996.

[6] G. Jochens, L. Kruse, and W. Nebel. Application of toggle-based power estimation to module characterization. In *Proc. of Power and Timing Modeling of Integrated Circuits*, pages 161–170, 1997.

[7] A. B. Kahng and S. Muddu. Gate load delay computation using analytical models. In *Proc. of Asia-Pacific Conference on Circuits and Systems*, pages 433–436, 1996.

[8] Meta Software Corporation. *Design/CPN Reference Manual*, 2 edition, 1993.

[9] F. N. Najm. A survey of power estimation techniques in vlsi circuits. *IEEE Trans. on VLSI Systems*, 2(4):644–649, Dec. 1994.

[10] P. Rokyta, W. Fengler, and T. Hummel. Electronic system design automation using high level petri nets. In *Hardware Design and Petri Nets, ed. A. Yakovlev, L. Gomes and L. Lavagno*, pages 193–204. Kluwer Academic Publishers, 2000.

[11] T H. Krodel. Power play: Fast dynamic power estimation based on logic simulation. In *Proc. of the Intl. Conf. on Computer Design*, pages 96–100, 1991.

[12] C.-Y. Tsui, M. Pedram, and A. M. Despain. Efficient estimation of dynamic power consumption under a real delay model. In *Proc. of the Intl. Conf. on Computer Aided Design*, pages 224–228, 1993.