Internode: Internal Node Logic Computational Model

Alejandro Millan, Manuel J. Bellido, Jorge Juan, David Guerrero, Paulino Ruiz-de-Clavijo, Enrique Ostua
Departamento de Tecnologia Electronica - Universidad de Sevilla
Av. Reina Mercedes, s/n (E. T. S. Ingenieria Informatica) - 41012 Sevilla (Spain)
Tel.: +34 954552764 - Fax: +34 954552764. http://www.dte.us.es
Instituto de Microelectronica de Sevilla - Centro Nacional de Microelectronica
Av. Reina Mercedes, s/n (Edificio CICA) - 41012 Sevilla (Spain)
Tel.: +34 955056666 - Fax: +34 955056686. http://www.imse.cnm.es
{amillan, bellido, jjchico, guerre, paulino, ostua}@dte.us.es

Abstract

In this work, we present a computational behavioral model for logic gates called Internode (Internal Node Logic Computational Model) that considers the functionality of the gate as well as all the different internal states the gate can reach. This computational model can be used in logic-level tools and is valid for any dynamic behavioral model (delay models, power models, switching noise models, etc.). Also, we show a very efficient implementation of the model, in C language, for N-inputs SCMOS NOR/NAND gates. Finally, we demonstrate the functionality of the model showing three different examples of modeling: (a) a propagation delay model, (b) the degradation delay model (DDM), and (c) a simple power model.[1]

1. Introduction

In order to achieve the advantages of the evolution in integrated circuits technology, it is necessary not only to fabricate the same circuits with higher performance but to include full systems inside the chip. That is, the complexity of the digital designs is increasing significantly. This fact has two main consequences: (a) it is much more difficult to design high-speed circuits and (b) the power consumption must be reduced in order to avoid chip damage.

Nowadays, tools for verification and analysis have a higher importance because they assist the designer in obtaining the best design for a specific system. In this sense, it is necessary to make an effort and improve the accuracy of current logic-verification tools in order to provide the designer with good enough results allowing a correct evaluation of the circuit behavior. An example of this idea is the logic tools for power consumption estimation. These tools are based on the switching activity measurement. Recently, we have shown that accuracy of switching activity in current logic tools is very low [2, 3]. This low accuracy is the reason why tools for power consumption estimation are not precise enough for designers to rely on.

In order to improve this precision it is necessary to develop new and more accurate models of the gate's dynamic behavior. In this sense, we have observed that a mandatory way for improving accuracy is to consider different models in the dynamic behavior, depending on the different states associated to the logic behavior [11, 12]. For example, the propagation delay of a gate can vary according to what input changes. This fact is due to gate's internal structure, which is not symmetrical from the inputs point of view.

Including different models of the dynamic behavior in a logic-level tool leads to change the functional model of the logic components into other one. This new model must include not only the functionality but also the gate's internal state allowing us to choose the correct dynamic model for each situation.

So, in this work, we present a computational behavioral model for logic gates that considers the functionality of the gate as well as all the different internal states the gate can reach. This computational model can be used in logic-level tools and is valid for any dynamic behavioral model (delay models, power models, switching noise models, etc.) that may be developed for the different gates.

The model is based on a Finite State Machine (FSM) with a number of states adequate to represent all possible

[1]This work has been partially supported by the MCYT MODEL project TIC 2000-1350, the MCYT VERDI project TIC 2002-2283, and the MECD/SEEU/DGU project PHB2002-0018-PC of the Spanish Government.

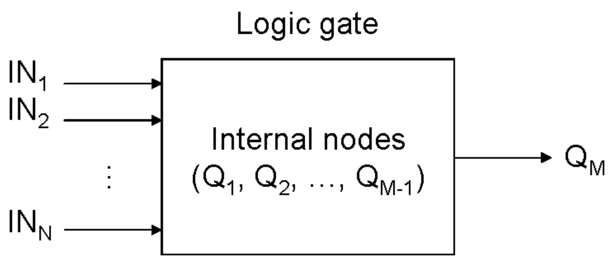


Figure 1. Input-Output structure of the Internode model for a logic gate with N inputs and M internal nodes.

internal states of the gate (Fig. 1). The internal states are related to the internal nodes of the logic gate in the sense that there is one state variable for each internal state.

In the rest of the paper we are going to present the full model for Standard CMOS (SCMOS) gates. With this idea, the organization of the paper is as follows: in Sect. 2 we show a detailed description of the Internode model for the cases of 1-input and 2-inputs SCMOS gates; in Sect. 3 the extension to N-inputs SCMOS gates is presented; in Sect. 4 we demonstrate the functionality of the model showing three different examples of modeling (a propagation delay model, the degradation delay model, and a simple power model); finally, in Sect. 5 we will finish with the main conclusions of this work.

2. Model description for 1-input and 2-inputs SCMOS gates

The Internode (Internal Node Logic Computational) model considers a FSM for the behavior of the gate. The specific FSM depends on the gate structure and the number of states of the internal and output nodes. The model is based on the notation used in the Moore automata [10]. In this section we are going to present the model for 1-input and 2-inputs SCMOS gates.

The simplest case is the one for the inverter gate (Fig. 2). In this gate we have only one transistor in each MOS-tree and there are no internal nodes. In this case, the corresponding Internode model has only two states depending on the output value, and transitions are determined by present input values (Fig. 3). Note that, in the case of the inverter, the corresponding Internode model is equivalent to the functional model of the gate.

If we consider gates with two or more inputs, we always have two or more transistors in each MOS-tree with one or more internal nodes. So, the corresponding Internode model must consider the state of these internal nodes (charged or discharged) as well as the output value.

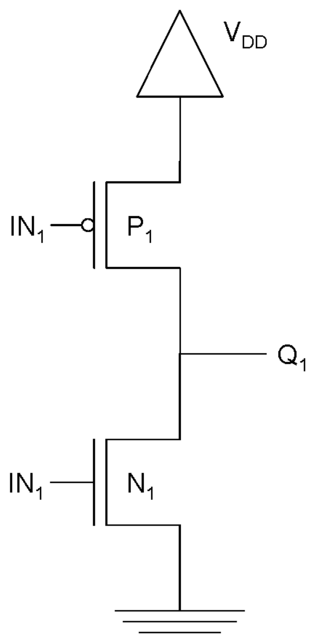


Figure 2. SCMOS structure for an INV gate.

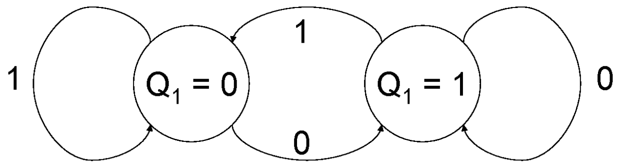


Figure 3. Internode model corresponding to an INV gate.

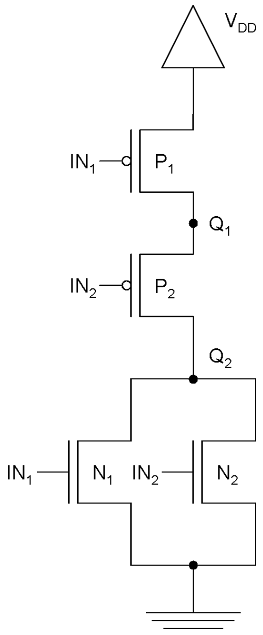


Figure 4. SCMOS structure for a NOR-2 gate.

Let us consider, for example, the case of the NOR-2 gate (Fig. 4). This gate has two transistors in serial mode in the PMOS-tree (P_1 and P_2) and one internal node. On the one hand, considering the output value (Q_2) and the internal node state (Q_1), we have four possible cases producing four states in our Internode model. On the other hand, in order to consider the behavior of the gate, it is necessary to establish transitions between states due to input changes. In Fig. 5 we show the Internode model for a 2-inputs NOR gate (NOR-2).

Due to the gate structure, the state $Q_1Q_2 = 01$ is impossible to reach because, if Q_2 is charged ($Q_2 = 1$), then

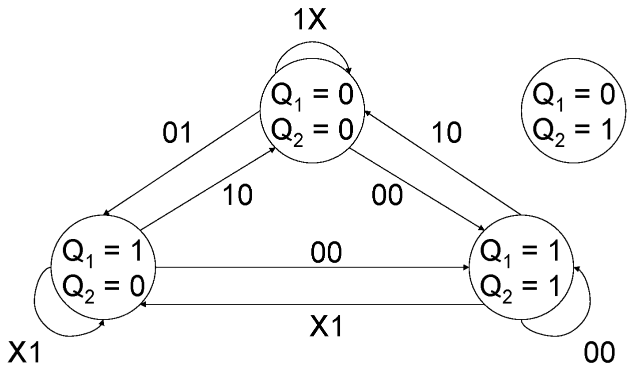


Figure 5. Internode model corresponding to a SCMOS NOR-2 gate.

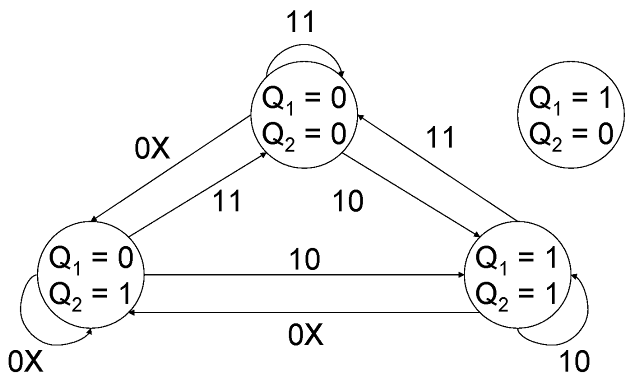


Figure 6. Internode model corresponding to a SCMOS NAND-2 gate.

it must be connected to V_{DD} and it would imply Q_1 to be connected to V_{DD} too (producing $Q_1 = 1$ instead of $Q_1 = 0$). Also, the state $Q_1Q_2 = 11$ is reached only when the gate receives $IN_1IN_2 = 00$. The input case $IN_1IN_2 = 10$ leads to state $Q_1Q_2 = 00$ always, and the input case $IN_1IN_2 = 01$ leads to state $Q_1Q_2 = 10$.

With this operation method we can consider that each state has a characteristic input value. However, it is possible to stay in states $Q_1Q_2 = 00$ or $Q_1Q_2 = 10$ receiving other input value (for example, $IN_1IN_2 = 11$).

In a similar and dual way it is possible to get the Internode model for the SCMOS NAND-2 gate (Fig. 6).

When we compare the Internode model for 2-inputs SCMOS gates (Fig. 5 and Fig. 6) with the functional model for the same gates (Fig. 7 and Fig. 8), we can observe that the Internode model deals with the internal state of the gate much better than the functional model. If we intend to apply a dynamic behavioral model, we can reach a higher accuracy using the Internode model because it allows to consider different situations that are considered to be the same in the functional model.

Let us consider, for example, a delay model for the case of a NOR-2 gate. In the functional model we have only one situation in which output raises: from state $Q_2 = 0$ to state $Q_2 = 1$. However, this raise can be reached from two different real states: internal node charged (state $Q_1Q_2 = 10$) or internal node discharged (state $Q_1Q_2 = 00$). That is, in the Internode model we can consider different delay models for these different situations, while in the functional model we have to use the same delay model for them both.

3. Extension to N-inputs SCMOS gates

The presented model can be easily extended to SCMOS gates with more than two inputs. In this section, we will

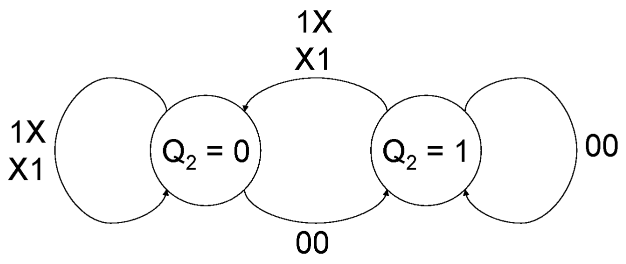


Figure 7. Functional model corresponding to a NOR-2 gate.

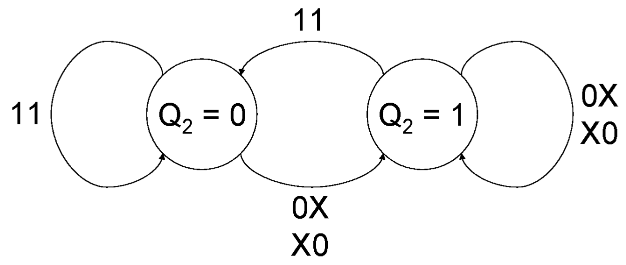


Figure 8. Functional model corresponding to a NAND-2 gate.

extend it to N-inputs SCMOS gates. In order to do this, in the easiest way, we are going to establish several behavioral rules that we can apply to the implementation of the model. The rules are the next:

1. A node is only charged if and just if connected to V_{DD}.
2. A node is only discharged if and just if connected to ground. Note that it is impossible for a node to charge and discharge at the same time due to the standard CMOS structure (a node can not be connected to ground and V_{DD} simultaneously).
3. If there is a node (Q_A) disconnected from V_{DD}, that previously has been charged, and due to a new input configuration this node is connected to a second one (Q_B), then we consider that the charge remains at the first node (Q_A).

Note that in the case described by rule 3 actually the charge would be distributed in both nodes. However, the model performs correctly making the supposition of rule 3, and this rule is necessary in order to maintain the model in a logic level.

The two first rules imply that a new input in a gate (that is, to model a gate with $(N + 1)$-inputs instead of an N-inputs one) only means a new state in the FSM of the Internode model respect of the N-inputs case.

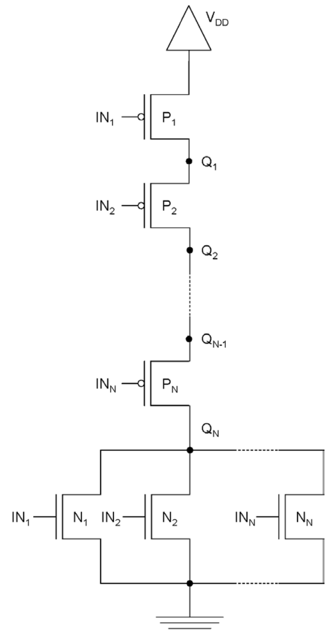


Figure 9. Structure of a SCMOS NOR-N gate.

Keeping these rules in mind, we are going to study the N-inputs NOR and NAND gates (NOR-N and NAND-N). As we are going to see, it is possible to build an algorithm for the Internode model in order to establish the specific model for each NOR/NAND gate. The algorithm is a more suitable representation of the model than the state diagram we have used for the 1-input and 2-inputs gates because, for the N-inputs case, the N-states diagram is more difficult to manipulate and understand. Also, the algorithm shows a possible implementation of the Internode model in a logic-level tool.

3.1 Extension to NOR-N gates

For the NOR-N case, we can establish the next algorithm in order to estimate the new state for a given gate (Fig. 9):

1. Consider $Q = (Q_1, Q_2, ..., Q_N)$ as the present state of the gate having Q_1, Q_2, etc. as the internal nodes charge indicators and Q_N as the output charge indicator (Q_1 is the internal node nearest to V_{DD}).
2. Consider $IN = (IN_1, IN_2, ..., IN_N)$ as the present input configuration (IN_1 is the input nearest to V_{DD}).
3. Consider $QP = (QP_1, QP_2, ..., QP_N)$ as the future state of the gate.
4. Initially, assume $QP = Q$.

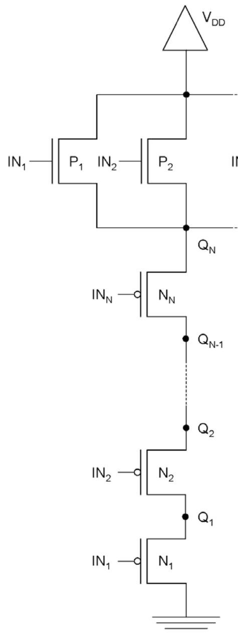


Figure 10. Structure of a SCMOS NAND-N gate.

5. Analyze internal nodes from V_{DD} to output node. If IN_1 is 0 then transistor P_1 is in on-state and internal node Q_1 is charged. So, if P_1 is in on-state, we study Q_2: if IN_2 is 0 then transistor P_2 is in on-state and internal node Q_2 is charged (because Q_2 is connected to V_{DD} through P_1 and P_2). While we are charging nodes, we must continue until output node (Q_N) is reached.

6. Analyze transistors in NMOS tree. If there is any transistor in on-state (name it N_J) then output node (Q_N) is discharged. So, if Q_N is discharged, we study Q_{N-1}: if IN_N is 0 then transistor P_N is in on-state and internal node Q_{N-1} is discharged (because Q_{N-1} is connected to ground through P_N and N_J). While we are discharging nodes, we must continue until Q_1 is reached.

3.2 Extension to NAND-N gates

For the NAND-N case, we can establish a similar algorithm in order to estimate the new state for a given gate (Fig. 10):

1. Consider $Q = (Q_1, Q_2, ..., Q_N)$ as the present state of the gate having Q_1, Q_2, etc. as the internal nodes charge indicators and Q_N as the output charge indicator (Q_1 is the internal node nearest to ground).

2. Consider $IN = (IN_1, IN_2, ..., IN_N)$ as the present input configuration (IN_1 is the input nearest to ground).

3. Consider $QP = (QP_1, QP_2, ..., QP_N)$ as the future state of the gate.

4. Initially, assume $QP = Q$.

5. Analyze internal nodes from ground to output node. If IN_1 is 1 then transistor N_1 is in on-state and internal node Q_1 is discharged. So, if N_1 is in on-state, we study Q_2: if IN_2 is 1 then transistor N_2 is in on-state and internal node Q_2 is discharged (because Q_2 is connected to ground through N_1 and N_2). While we are discharging nodes, we must continue until output node (Q_N) is reached.

6. Analyze transistors in PMOS tree. If there is any transistor in on-state (name it P_J) then output node (Q_N) is charged. So, if Q_N is charged, we study Q_{N-1}: if IN_N is 1 then transistor N_N is in on-state and internal node Q_{N-1} is charged (because Q_{N-1} is connected to V_{DD} through N_N and P_J). While we are charging nodes, we must continue until Q_1 is reached.

3.3 Implementation function for N-inputs gates

The presented algorithms can be easily implemented in a single function. Now, we show the corresponding function in C language. Note that due to C syntax the internal node further to the output node is Q_0 and the output node is Q_{N-1}. The input vector is changed in the same way: the first input is now IN_0 and the last one is IN_{N-1}. The code would be the next:

```
void qpgate(int G, int N, int Q[], int
  IN[], int QP[])
// G:    Gate type (0 = NOR, 1 = NAND)
// N:    Number of inputs of the gate
// Q[]:  Vector containing present
//       state.
// IN[]: Vector containing present
//       inputs.
// QP[]: Vector containing future state
// This function assumes that a NAND-N
// gate is in state Q receiving IN as
// present input configuration and ret-
// urns the future state QP of the gate
{
  int b; // Flag
  int n; // Counter
  int ser_on; // Value that causes a
  // serial ttor. to enter in on-state
  int par_on; // Value that causes a
// paral. ttor. to enter in on-state
int A; // New Q[j] in serial process
int B; // New Q[j] in paral. process

if (G == 0) { // NOR gate
  ser_on = 0;
  par_on = 1;
  A  = 1;
  B  = 0;
}
else { // NAND gate
  ser_on = 1;
  par_on = 0;
  A  = 0;
  B  = 1;
}

// Initially we assume QP = Q:
for (n = 0; n < N; n = n + 1)
  QP[n] = Q[n];

// We analyze internal nodes
// (dis)charge:
b = 1;
n = 0;
while (b && (n < N)) {
  if (IN[n] == ser_on) {
    QP[n] = A;
    n = n + 1;}
  else b = 0;
}

// We analyze if any parallel ttor.
// is in on-state:
b = 0;
for (n=0; (n<N) && (b==0); n=n+1)
  if (IN[n] == par_on) b = 1;

// If there is any parallel ttor. in
// on-state we must analyze internal
// nodes (dis)charge:
if (b == 1) {
  QP[N - 1] = B; // The output node
                 // is (dis)charged.
  b = 1;
  n = N - 1;
  while (b & (n >= 1)) {
    if (IN[n] == ser_on) {
      QP[n - 1] = B;
      n = n - 1;}
    else b = 0;
  }
}
}
```

Observe that the order of the function is N. For this reason, the inclusion of a new transistor in the gate (a gate with one more input) produces only one more iteration in the estimation of the future state of the gate.

So, from the computational point of view, the usage of the Internode model in a logic-level tool has the same performance than the functional model. Also, as the domain of the presented function is finite, we can use a look-up table to store the precalculated future states for all the situations the gate can reach. In this case, the order will be reduced to a unit at simulation time.

4. Example applications of the Internode model

In order to demonstrate the functionality of the Internode model, we want to show how to include a specific dynamic behavioral model into it. In this way, we will present three different examples of modeling: (a) a propagation delay model, (b) the degradation delay model (DDM), and (c) a simple power model. To simplify, we will show the application of the models in the case of a NAND-2 gate.

4.1 Application of the Internode model to a propagation delay model

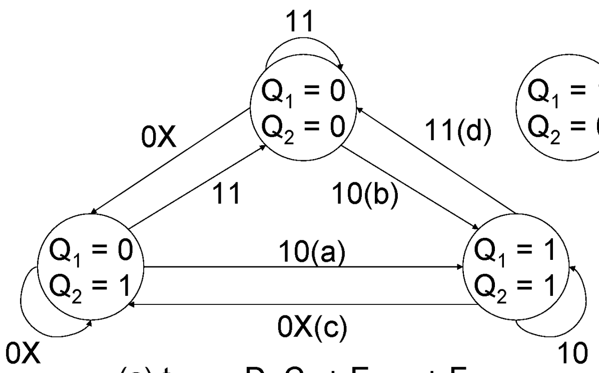
The value of the normal propagation delay (t_{p0}) depends on both the output load, C_L, and the input transition time, τ_{in} [5, 1]. In previous papers [8, 9] we have applied this dependence for obtaining a simple model for the normal propagation delay.

The model is based in a characterization process in which, for each gate, we measure the delay from each input to the output for both falling and raising output transitions. This results in a simple and well-known heuristic model that fits the normal propagation delay:

$$t_{p0} = D_{xi}C_L + E_{xi}\tau_{in} + F_{xi} \qquad (1)$$

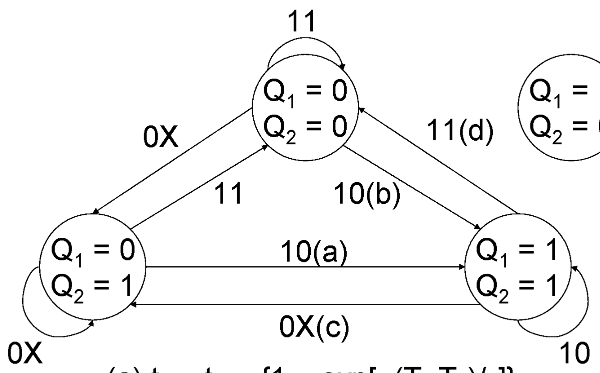
where D_{xi}, E_{xi}, and F_{xi} are the model parameters. An individual parameter value is obtained for each type of output transition (r or f, noted by x) and each input of the gate (noted by i). This model relies on the mentioned set of parameters $\{D_{xi}, E_{xi}, F_{xi}\}$ which have to be characterized for each gate and transition type.

Once the model has been roughly explained, we need to include it into the corresponding Internode model. This process consists of assigning an equation for t_{p0} in those transitions in which output changes. The resulting diagram can be observed in Fig. 11.



(a) $t_{p0r1} = D_{r1}C_L + E_{r1}\tau_{in} + F_{r1}$
(b) $t_{p0r2} = D_{r2}C_L + E_{r2}\tau_{in} + F_{r2}$
(c) $t_{p0f1} = D_{f1}C_L + E_{f1}\tau_{in} + F_{f1}$
(d) $t_{p0f2} = D_{f2}C_L + E_{f2}\tau_{in} + F_{f2}$

Figure 11. Internode model corresponding to a SCMOS NAND-2 gate with a normal propagation delay model included.



(a) $t_p = t_{p0r1}\,\{1 - \exp[-(T-T_0)/\tau]\}$
(b) $t_p = t_{p0r2}\,\{1 - \exp[-(T-T_0)/\tau]\}$
(c) $t_p = t_{p0f1}\,\{1 - \exp[-(T-T_0)/\tau]\}$
(d) $t_p = t_{p0f2}\,\{1 - \exp[-(T-T_0)/\tau]\}$

Figure 12. Internode model corresponding to a SCMOS NAND-2 gate with the degradation delay model included.

4.2 Application of the Internode model to the Degradation Delay Model (DDM)

The DDM is a very accurate model that handles the generation and propagation of glitches [4, 7, 6]. The equation to evaluate the propagation delay according to the DDM is:

$$t_p = t_{p0}\left\{1 - \exp\left[\frac{-(T-T_0)}{\tau}\right]\right\} \qquad (2)$$

where T is the time elapsed since the last output transition, t_{p0} is the normal propagation delay and T_0 and (are the degradation parameters.

For each gate, $(\tau$ and T_0 depend on the output load (C_L), the supply voltage (V_{DD}), the input transition time (τ_{in}), and the position of the input that is changing state (i). It has been obtained [6] that this dependence can be expressed as:

$$\tau_x V_{DD} = A_{xi} + B_{xi}C_L \qquad (3)$$

$$T_{0x} = \tau_{in}\left(\frac{1}{2} - \frac{C_{xi}}{V_{DD}}\right) \qquad (4)$$

where x stands for r or f depending on the sense of the output transition (rise or fall respectively). So, a CMOS gate is fully characterized with respect to the degradation effect when the set $\{A_{xi}, B_{xi}, C_{xi}\}$ is obtained for each gate input.

Once the model has been roughly explained, we can include it into the corresponding Internode model. The process consists of assigning an equation for t_p in those transitions in which output changes (the corresponding t_{p0} can be calculated as in Fig. 11). The resulting diagram can be observed in Fig. 12.

4.3 Application of the Internode model to a simple power model

Finally, we will show how to model power consumption. In this way, we consider a simple model: only dynamic power consumption is taken into account. So, if we represent the dissipation energy in the gate, we can model this effect as: $E = (1/2)C_L V_{DD}^2$, in both cases of output falling and output rising. This equation must be included in those transitions in which output changes, obtaining the diagram showed in Fig. 13 (we assume an energy dissipation of zero in the rest of transitions).

5. Conclusions

A mandatory way for improving accuracy is to consider different models in the dynamic behavior, depending on the different states associated to the logic behavior. So, in this work, we have presented a computational behavioral model for logic gates that considers the functionality of the gate as well as all the different internal states the gate can reach. This computational model can be used in logic-level tools and is valid for any dynamic behavioral model (delay models, power models, switching noise models, etc.) that may be developed for the different gates.

Firstly, we have presented the Internode model corresponding to 1-input and 2-inputs gates. Secondly, we have extended this explanation to a generic N-inputs NOR/NAND gate showing how to implement the model in a standard programming language like C. We have mentioned

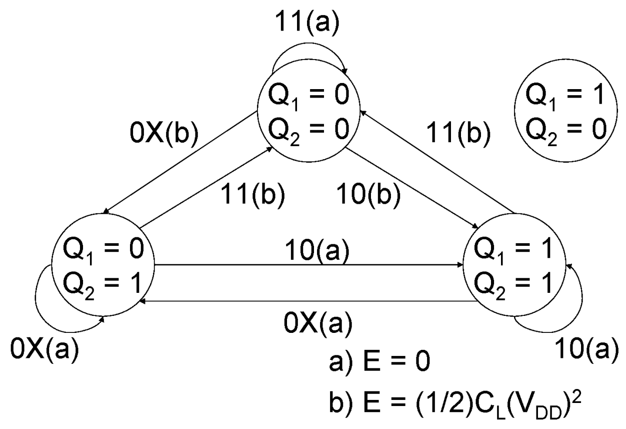


Figure 13. Internode model corresponding to a SCMOS NAND-2 gate with a simple power model included.

that the order of the implemented function is N (that is, the amount of inputs). Also, it is possible to use a look-up table due to the finite nature of the function domain, reducing this order to a unit at simulation time.

Finally, we have demonstrated the functionality of the Internode model showing how to include a specific dynamic behavioral model into it. In this way, we have presented three different examples of modeling: (a) a propagation delay model, (b) the degradation delay model (DDM), and (c) a simple power model.

References

[1] D. Auvergne, N. Azemard, D. Deschacht, and M. Robert. Input waveform slope effects in CMOS delays. *IEEE Journal of Solid-State Circuits*, 25(6):1588–1590, December 1990.

[2] C. Baena, J. Juan, M. J. Bellido, P. Ruiz-de Clavijo, C. J. Jimenez, and M. Valencia. Simulation-driven switching activity evaluation of CMOS digital circuits. In *Proc. 16th Conference on Design of Circuits and Integrated Systems (DCIS)*, Porto (Portugal), November 2001.

[3] C. Baena, J. Juan, M. J. Bellido, P. Ruiz-de Clavijo, C. J. Jimenez, and M. Valencia. Measurement of the switching activity of CMOS digital circuits at the gate level. In *Proc. 12th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 353–362, Seville (Spain), September 2002.

[4] M. J. Bellido-Diaz, J. Juan-Chico, A. J. Acosta, M. Valencia, and J. L. Huertas. Logical modelling of delay degradation effect in static CMOS gates. *IEE Proc. Circuits Devices and Systems*, 147(2):107–117, April 2000.

[5] L. Bisdounis, S. Nikolaidis, and O. Koufopavlou. Analytical transient response and propagation delay evaluation of the CMOS inverter for short-channel devices. *IEEE Journal of Solid-State Circuits*, 33(2):302–306, February 1998.

[6] J. Juan-Chico, P. Ruiz-de Clavijo, M. J. Bellido, A. J. Acosta, and M. Valencia. Degradation Delay Model extension to CMOS gates. In *Proc. Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 149–158, September 2000.

[7] J. Juan-Chico, P. Ruiz-de Clavijo, M. J. Bellido, A. J. Acosta, and M. Valencia. Inertial and Degradation Delay Model for CMOS logic gates. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pages I–459–462, Geneva, May 2000.

[8] A. Millan, J. Juan, M. J. Bellido, R. de Clavijo, and D. Guerrero. Characterization of normal propagation delay for Delay Degradation Model (DDM). In *Proc. 12th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 477–486, Seville (Spain), September 2002.

[9] A. Millan, J. Juan, M. J. Bellido, R. de Clavijo, and D. Guerrero. Integration of a characterization method for normal propagation delay into AUTODDM. In *Proc. 17th Conference on Design of Circuits and Integrated Systems (DCIS)*, pages 93–97, Santander (Spain), November 2002.

[10] E. F. Moore. *Gedanken experiments on sequential machines*, pages 129–153. Princeton University Press, Princeton, NJ, 1956.

[11] P. Ruiz-de Clavijo, J. Juan, M. J. Bellido, A. J. Acosta, and M. Valencia. Halotis: High Accuracy LOgic TIming Simulator with Inertial and Degradation Delay Model. In *Proc. Design, Automation and Test in Europe (DATE) Conference and Exhibition*, Munich (Germany), March 2001.

[12] P. Ruiz-de Clavijo, J. Juan, M. J. Bellido, A. Millan, and D. Guerrero. Efficient and fast current curve estimation of CMOS digital circuits at the logic level. In *Proc. 12th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 400–408, Seville (Spain), September 2002.