AUTODDM: AUTOmatic characterization tool for the Delay Degradation Model

J. Juan-Chico*, M. J. Bellido*, P. Ruiz-de-Clavijo*, C. Baena*, M. Valencia*
Instituto de Microelectrónica de Sevilla-CNM, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
e-mail: jjchico@imse.cnm.es
*Also with the Departamento de Tecnología Electrónica. Universidad de Sevilla. Spain.

Abstract

As delay models used in logic timing simulation become more and more complex, the problem of model parameter values extraction arise as an important issue, which is necessary to face in order to achieve a practical implementation of the model. In this way, this communication describes the characterization process associated to the previously developed Delay Degradation Model for CMOS logic gates (DDM) and the implementation of an automatic characterization tool that automates the process and allows an easy and fast model parameters extraction.

In the field of logic simulation of digital CMOS circuits, delay models exist that take into account most issues affecting accuracy [1,2,3,4]: low voltage, submicron and deep submicron devices, transition waveform, etc. There are also dynamic effects, the most important being the so-called *input collisions* [5], which happens when two or more input signals change almost simultaneously. The type of input collision that more notably affects the behaviour of digital circuits are the *glitch collisions*, or those that may cause narrow pulses or glitches. In previous papers [6, 7, 8] we have presented a very accurate model that handles the generation and propagation of glitches, which makes an important headway in logic timing simulation. This model is called *Delay Degradation Model* (DDM).

One important point in any delay model (including the DDM) is the definition of the model parameters and the set up of an useful characterization process that describes how the model parameter values are obtained. This information is necessary to be able to reproduce simulation results by others and also to check the viability of the approach: a model that is very hard or expensive to characterize may be useless.

In this paper we describe the parameter characterization process associated to the DDM and we present a tool that automates the process. We will show that the automation of the process is necessary in order to be able to use the model in a practical way.

The paper is organized as follows: in section 2 the Delay Degradation Model is presented, in section 3 we describe the characterization process and its complexity, section 4 presents the characterization tool *autoddm* from the point of view of use and implementation; performance results are shown in section 5 and we will finish with the main conclusions of this work.

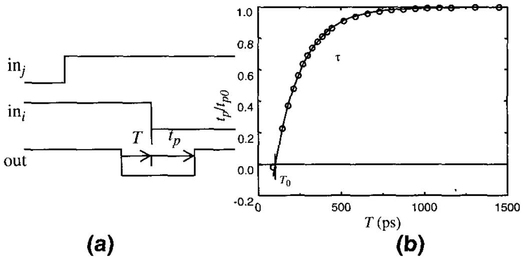

Figure 1. Modelling delay degradation. a) T parameter, b) degradation curve.

2. Degradation Delay Model (DDM)

Narrow pulses or glitches propagating through CMOS gates are affected by the so-called *degradation effect*. In previous papers, we have developed a model that handles the degradation effect and is able, then, to accurately simulate the propagation of arbitrarily narrow pulses. Despite the model has been presented in previous papers [6, 7, 8], we will summarize its main points in this section.

A suited way to describe and quantify the degradation effect is as a reduction in the delay (t_p) with respect to the *normal propagation delay* as a function of the inputs and output timing characteristics. Here, *normal* means the delay when no degradation effect takes place, which is the one calculated by conventional delay models [1,2]. It has been demonstrated in [6] that this reduction in the delay depends on the time elapsed since the last output transition (T) as shown in Figure 1a. The t_p dependence on T of Figure 1b very well fits to the following expression:

$$t_p = t_{p0}\left(1 - e^{-\frac{T - T_0}{\tau}}\right) \qquad \text{(eq. 1)}$$

where t_{p0} is the normal propagation delay and T_0 and τ are fitting parameters.

For a given gate, there is a set of input triggering conditions that may produce degradation effect, called *glitch collisions*. Some of them correspond to narrow pulses in a given input while others correspond to almost simultaneous changes of a pair of input signals. A classification of

0-7803-7057-0/01/$10.00 ©2001 IEEE.

the types of input collisions that produce degradation effect can be found in [8].

For each particular gate, τ 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 in [8] that this dependence can be expressed as:

$$\tau_x V_{DD} = A_{xi} + B_{xi} C_L \qquad \text{(eq. 2)}$$

$$T_{0x} = \left(\frac{1}{2} - \frac{C_{xi}}{V_{DD}}\right)\tau_{in} \qquad \text{(eq. 3)}$$

where "x" stands for "r" or "f" depending on the sense of the output transition (rise or fall respectively). In this way, a CMOS gate is fully characterized with respect to the degradation effect when the set of A, B and C parameters (the *degradation parameters*) are obtained for each gate input. Therefore, the number of degradation parameter for a n-input gate is:

$$n_p = 6n \qquad \text{(eq. 4)}$$

The degradation parameters are obtained by fitting (eq. 2) and (eq. 3) to HSPICE simulation data.

3. Characterization process

The objective of the characterization process is to obtain the values of the set of degradation parameters of (eq. 2) and (eq. 3) for a particular gate, i.e.

$$\{A_{xi}, B_{xi}, C_{xi}\} \qquad x = r, f \qquad i = 1\ldots n \quad \text{(eq. 5)}$$

3.1 Characterization process description

The characterization process is composed of three main tasks:

Task 1: obtain t_p vs. T curves corresponding to (eq. 1) and Figure 1b.

Task 2: obtain τ vs. C_L curves corresponding to (eq. 2).

Task 3: obtain T_0 vs. τ_{in} curves corresponding to (eq. 3).

The first task is the source to obtain τ, T_0 pairs, which are needed by tasks 2 and 3. τ, T_0 pairs are extracted from applying a regression fitting to data obtained from electrical simulations (SPICE [9] or HSPICE [10]) like in Figure 1b. Each point in the curve requires a transient analysis for a given value of T. The number of transient analysis that are necessary to represent a t_p vs. T curve is noted as n_{curve} and a typical value is 20.

By repeating task 1 with different values of C_L, a set of τ, C_L pairs is obtained, allowing for the calculation of parameters A and B by linear regression (eq. 2). The number of points used in a τ vs. C_L curve is noted as n_{AB} and a typical value is 10. Task 3 is carried out in a similar way, by performing task 1 a number of times (n_C) with different values of τ_{in} to obtain parameter C by linear regression (eq. 3). n_C is also typically equal to 10.

3.2 Characterization process complexity

A good manner to measure the complexity of the characterization process is by evaluating the total number of transient analysis that are needed to characterize a gate. Considering that $2n$ is the number of input collisions that produce degradation, the number of transient analysis can be calculated as:

$$n_{tran} = 2(n_{AB} + n_C) n_{curve} n \qquad \text{(eq. 6)}$$

where n is the number of inputs of the gate.

Another interesting parameter measuring the characterization process performance is the *characterization time* (t_{car}) which is necessary to complete the characterization of a gate. It is proportional to the number of transient analysis and can be expressed like:

$$t_{car} = t_f n_{tran} \qquad \text{(eq. 7)}$$

where t_f is the *characterization time factor* which measures the average time needed to set up and run each transient analysis. It is useful to split this time factor in two:

$$t_f = t_{fop} + t_{fsim} \qquad \text{(eq. 8)}$$

where t_{fsim} is the *simulator time factor*, which measures the average time used by the electrical simulator in each transient analysis; and t_{fop} is the *operator time factor* which includes the average time spent in any other tasks done to prepare each transient analysis: file editing, simulation launching, data storing, problem resolution and data analysis (regression, fitting, etc.).

In a conventional laboratory setup, t_{fop} would correspond to the time spent by a human operator in performing the mentioned tasks using a set of computer tools, while t_{fsim} is the CPU time consumed by an electrical simulator in each transient run. In this case, t_{fsim} is negligible with respect to t_{fop}.

As an example, the time factor t_f can be optimistically estimated in 10sec., assuming that an experimented operator is driving the characterization process. If we also consider typical values like $n_{curve} = 20$, $n_{AB} = n_C = 10$ and a 4-input gate, the characterization time can be calculated from (eq. 6) and (eq. 7):

$$\begin{aligned} t_{car} &= 10 \times 2(10 + 10)20 \times 4 = 3200\text{sec} \\ &= 8\text{h, } 53\text{min, } 20\text{sec} \end{aligned} \qquad \text{(eq. 9)}$$

This means that, in the best case, a well trained human operator would spent around 9 hours in the characterization of a single gate. This cost is excessive in most cases, specially when the objective is to characterize a whole library of gates or when exploring different gate configurations.

4. Characterization tool description

A tool called *autoddm* has been implemented in order to automate and speed up the characterization process described in the previous section, saving "human" time. The tool is easy to use and is able to provide with the whole set of degradation parameters of a gate, from a basic information specified by the user.

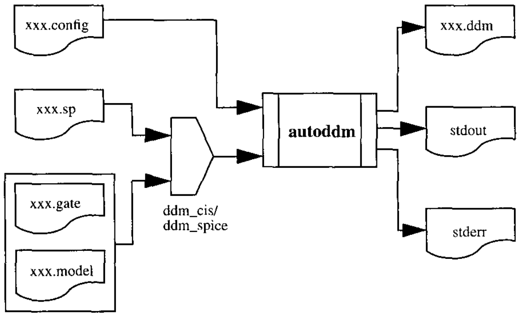

Figure 2. *Autoddm* input/output data flow.

Two versions of the program are currently implemented: *autoddm_cis* and *autoddm_spice*, which only difference is the electrical simulator used to perform the electrical simulations. The *autoddm_cis* uses CIS, which is a library designed to control the HSPICE simulator, while *autoddm_spice* uses Berkeley's SPICE3f5.

Figure 2 shows the information flow around the *autoddm* tools. The designer has to specify a configuration file (*xxx.config*) with the information about the characterization process parameters and basic information about the gate to be analysed (type and number of inputs). Additionally, a netlist of the gate under analysis need to be provided, as well as the MOS models. In the case of the HSPICE version (*autoddm_cis*), this information is included in a file called *xxx.sp*, while in the case of the SPICE3f5 version (*autoddm_spice*), the netlist is placed in file *xxx.gate* and the MOS transistors model card is placed in file *xxx.model*.

Once the input files are created, the tool runs in batch mode and produces a characterization result which is stored in the *xxx.ddm* output file. The progress of the characterization process is printed to the standard output (*stdout*) and error messages are directed to the standard error output (*stderr*).

For the sake of clarity, we will use the *autoddm_spice* version in the following examples, the other version being similar.

The configuration file just defines a set of process parameters, and is composed of five sections:

Technology: oxide capacitance (C_{ox}) and minimum channel length of the technology (l_{min}). This parameters are not mandatory, but can simplify the use of other characterization process parameters.

Geometry: channel widths of the N-MOS and P-MOS trees in the gate. Again, these are not mandatory but convenient when using gates with homogeneous trees.

Files: name of input and output files. In the example, the configuration file is set to work either with the SPICE and HSPICE versions. Lines corresponding to the HSPICE version are commented out to select the SPICE version.

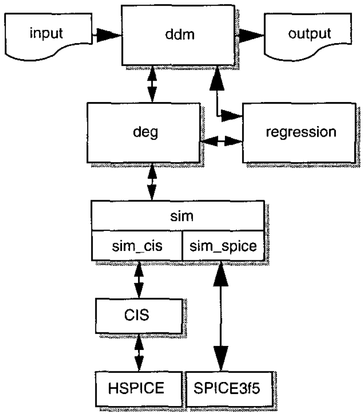

Figure 3. *Autoddm* block diagram.

Gate: gate type, number of inputs and gate's input capacitance (or reference capacitance). If technological and geometrical parameters are provided, C_{in} is calculated automatically if not present in the configuration file.

Simulation: various simulation parameters like the power supply value (V_{DD}) and the ranges of interest for the curves corresponding to the different tasks of the characterization process mentioned in the previous section.

The input to the tool is completed with the netlist and model files mentioned above. The last is usually one provided by the foundry, while the former is just a spice netlist of the gate.

With respect to the program output, a lot of useful information about the characterization process is generated besides the value of the degradation model parameters. For example, the program gives information about each degradation curve (task 1 of the process) allowing the monitoring of the characterization process.

5. Characterization tool implementation

Internally, *autoddm* is organized in four separate modules that interact following the diagram in Figure 3. These modules are: *ddm*, *deg*, *sim* and *regression*. We will describe them going from a lower to a higher level:

regression: is an utility module which consist on just one function to calculate the linear regression parameters that corresponds to a set of points. This is used by other parts of the program which need this facility.

sim: this module is in charge of running transient analysis on an electrical simulator in order to obtain delay values (t_p) for a given set of simulation parameters (C_L, τ_{in}, etc.). The sim module incorporates two submodules to allow interaction with either HSPICE or SPICE electrical simulators (*sim_cis* and *sim_spice*).

deg: this module uses the sim module functionality to obtain degradation curves like the one in Figure 1b, and calculates the corresponding τ and T_0 parameters from it (using the *regression* module) thus implementing task 1 of the characterization process.

ddm: corresponds to the main function of the program and implements the rest of the characterization algorithm (tasks 2 and 3). This function does the following: after reading the configuration from the input files, the program initiates some variables and starts the main loop that runs through every input collision that produces degradation, in order to obtain a set of parameters for each input. In each pass, the main loop executes two new loops: the first for different values of C_L, implementing task 2, and the second for different values of τ_{in}, implementing task 3. In both cases, the corresponding degradation parameters are obtained at the end of the loop by linear regression. Characterization data are printed as they are obtained. At the end of the main loop, all inputs have been characterized and the program ends.

6. Autoddm performance results

An example of a gate characterized using *autoddm* has been presented in Table 1. The value of the parameters obtained by *autoddm* are at least as accurate as those obtained by a human operator. In fact, *autoddm* actually achieves better results than a human operator since it makes more elaborated calculations to obtain a more homogeneous distribution of points in each degradation curve. Nevertheless, the main result is related to the time gain of *autoddm* with respect to a human operator. As an example, we will compare three representative cases: two automatic systems using both versions of *autoddm* and a human operator. System 1 corresponds to *autoddm_cis* (HSPICE) running on a Sun Ultra-SPARC-2 400MHz work station with Solaris 7, while system 2 is *autoddm_spice* (SPICE3f5) running on a PC Pentium II 266MHz with Linux 2.2.14. The characterization time factors for system 1 is 2.34, for system 2 is 0.179 and for the human operator is 10.

It is worth to remark that system 2 is much faster than system 1. We have discovered that the main reason is the delay added at the beginning of each HSPICE simulation due to the licence management tasks that are run by this tool during the simulator's set up. This is an important portion of the total simulation time.

Table 1 compares the performance of the three examples as a function of the number of inputs to characterize. The number of transient analysis is calculated using (eq. 6) with $n_{AB} = n_C = 10$ and $n_{curve} = 20$. The characterization time is calculated from (eq. 7). Four characterization tasks are studied: an inverter, a gate, a small library of gates and a big library of gates.Attending to these results, it is easy to conclude that the characterization process automation greatly improves the characterization time with respect to a manual characterization. Comparing systems 1 and 2, it is clear that the long HSPICE set up time represents a bottleneck in the characterization time, which does not appears when using SPICE3f5. Furthermore, characterization of a whole library is not affordable using a traditional method but is viable using an automatic

Table 1. Characterization times for the three compared cases in days, hours and minutes.

n	n_{tran}	characterization time (D:H:M)		
		Human	System 1	System 2
1 (inverter)	800	0:2:13	0:0:31	0:0:2.4
4 (gate)	3200	0:8:53	0:3:7	0:0:9.5
100 (small lib.)	80000	9:6:13	2:4:0	0:3:59
500 (big lib.)	400000	46:7:7	10:20:0	0:19:53

7. Conclusions

To provide a characterization process for a delay model may be as important as the model itself. In this way, we have described the tasks involved in the characterization process of the Delay Degradation Model developed previously and showed the need for an automatic characterization tool. This tool has been implemented following an easy to understand and reusable modular design. Performance results of the automatic characterization tool running on different platforms are presented, showing an improvement in the characterization time up to 50 times better than a characterization driven by a human operator, making the characterization process affordable.

8. References

[1] L. Bisdounis, S. Nikolaidis, O. Koufopavlou. "Analytical Transient Response and Propagation Delay Evaluation of the CMOS Inverter for Short-Channel Devices". IEEE J. of Solid-State Circ. pp. 302-306. Vol. 33, no. 2, Feb. 1998.

[2] J.M. Daga, D. Auvergne. "A Comprehensive Delay Macro Modeling for Submicrometer CMOS Logics". IEEE J. of Solid State Circuits. Vol. 34, No. 1, Jan. 1999.

[3] A.I. Kayssi, K.A. Sakallah, T.N. Mudge. "The Impact of Signal Transition Time on Path Delay Computation". IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 40, No. 5, pp. 302-309, May 1993.

[4] D. Auvergne, N. Azemard, D. Deschacht, M. Robert. "Input Waveform Slope Effects in CMOS Delays". IEEE J. of Solid-State Circ., Vol. 25, No. 6, pp. 1588-1590. Dec. 1990

[5] E. Melcher, W. Röthig, M. Dana. "Multiple Input Transitions in CMOS Gates". Microprocessing and Microprogramming 35 (1992) pp. 683-690. North Holland.

[6] M.J. Bellido-Díaz, J. Juan-Chico, A.J. Acosta, M. Valencia, J.L. Huertas. "Logical modelling of delay degradation effect in static CMOS gates". IEE Proc. Circuits Devices and Systems, Vol. 147, No. 2, pp. 107-117, April 2000.

[7] J. Juan-Chico, P. Ruiz-de-Clavijo, M.J. Bellido, A.J. Acosta, M. Valencia. "Inertial and degradation delay model for CMOS logic gates". In Proc. IEEE International Symposium on Circuits and Systems (ISCAS) 2000, pp. I-459-462, Geneva, May 2000.

[8] J. Juan-Chico, P. Ruiz-de-Clavijo, M.J. Bellido, A.J. Acosta, M. Valencia: "Degradation delay model extension to CMOS gates". In Proc. Power and Timing Modelling, Optimization and Simulation (PATMOS) 2000, pp. 149-158, Sept. 2000.

[9] A.R. Newton, D.O. Pederson, A. Sangiovanni-Vicentelli: "SPICE3 Version 3f3 User's Manual". Department of Electrical Engineering and Computer Sciences, Universidad de California, Berkeley. Distributed with SPICE 3F.

[10] "HSPICE User's Manual". Meta-software, 1999.