[Won86] K. F. Wong, M. A. Franklin, R. D. Chamberlain, and B. L. Shing, "Statistics on Logic Simulation", *IEEE 23rd Design Automation Conference*, pp. 13-19, 1986.

[Yoe64] M. Yoeli, S. Rinon, "Application of Ternary Algebra to the Study of Static Hazards", *Journal of the ACM*, pp. 84-97, Janvier 1964.

[Szy76] S. A. Szygenda, E. W. Thompson, "Modeling and Digital Simulation for Design Verification and Diagnosis", *IEEE Transactions on Computers*, pp. 1242-1253, Décembre 1976.

[Ter83] C. J. Terman, "RSIM - A Logic Level Timing Simulator", *IEEE International Conference on Computer Design*, pp. 308-331, Octobre 1983.

[Tha92] T. J. Thatcher, R. A. Saleh, "Automatic Partitioning and Dynamic Mixed-Mode Simulation", *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 12.7.1-12.7.4, 1992.

[Tho91] D. E. Thomas, P. R. Moorby, "The Verilog Hardware Description Language", Kluwer Academic Publishers, Boston, 1991.

[Tod96] A. R. W. Todesco, T. H.-Y. Meng, "Symphony : A Simulation Backplane for Parallel Mixed-Mode Co-Simulation of VLSI Systems", *ACM/IEEE Design Automation Conference*, Juin 1996.

[Ulr69] E. G. Ulrich, "Exclusive Simulation of Activity in Digital Networks", *Communications of the ACM*, pp. 102-110, Février 1969.

[Ulr78] E. G. Ulrich, "Event Manipulation for Discrete Simulations Requiring Large Numbers of Events", *Communications of the ACM*, pp. 777-785, Septembre 1978.

[Vau75] J. G. Vaucher, and P. Duval, "A Comparison of Simulation Event List Algorithms", *Communications of the ACM*, pp. 223-230, Avril 1975.

[Vis91] C. Visweswariah, R. A. Rohrer, "Piecewise Approximate Circuit Simulation", *IEEE Transactions on Computer-Aided Design*, pp. 861-870, Juillet 1991.

[Vis93] C. Visweswariah, J. A. Wehbeh, "Incremental Event-Driven Simulation of Digital FET Circuits", *ACM/IEEE Design Automation Conference*, pp. 737-741, Juin 1993.

[Vuo97] H. N. Vuong, "Une Nouvelle Méthode de Simulation par évaluation Directe des expressions logiques représentées par des graphes : Application à des Circuits Modélisés par un Sous-Ensemble du Langage VHDL", Thèse de Doctorat, Université de Paris VI, Décembre 1997.

[Wee73] W.-T. Weeks, A. J. Dumenez, G. W. Mahoney, D. Mehta, H. Qassemzadeh, T. R. Scott, "Algorithms for ASTAP -- A Network Analysis Program", *IEEE Transactions on Circuit Theory*, pp. 628-634, Novembre 1973.

[Whi83] J. White, A. L. Sangiovanni-Vincentelli, "RELAX2 : A New Waveform Relaxation Approach for the Analysis of LSI MOS Circuits", *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 756-759, Mai 1983.

[Whi84] J. White, A. L. Sangiovanni-Vincentelli, "RELAX2.1 : A Waveform Relaxation Based Circuit Simulation Program", *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 232-236, Mai 1984.

[Rab96] D. Rabe, W. Nebel, "Short Circuit Power Consumption of Glitches", *IEEE/ACM International Symposium on Low Power Electronics Design*, pp. 125-128, 1996.

[Rao82] V. B. Rao, T. N. Trick, M. R. Lightner, "Hazard Detection in a Multiple Delay Logic Simulator", *IEEE International Symposium on Circuits and Systems*, pp. 72-75, Mai 1982.

[Rao83] V. B. Rao, T. N. Trick, I. N. Hajj, "A Table-Driven Delay-Operator Approach to Timing Simulation of MOS VLSI Circuits", *IEEE International Conference on Computer Design*, pp. 445-448, Novembre 1983.

[Rao89] V. B. Rao, D. V. Overhauser, T. N. Trick, I. N. Hajj, **Switch-Level Timing Simulation of MOS VLSI Circuits**, Kluwer Academic Publisher, 1989.

[Sak90] T. Sakurai, A. R. Newton, "Alpha-Power Law MOSFET model and its Applications to CMOS Inverter Delay and Other Formulas", *IEEE Journal of Solid-State Circuits*, Avril 1990.

[Sak91] T. Sakurai, A. R. Newton, "Delay Analysis of Series-Connected MOSFET Circuits", *IEEE Journal of Solid-State Circuits*, pp. 122-131, Février 1991.

[Sal83] R. A. Saleh, J. E. Kleckner, A. R. Newton, "Iterated Timing Analysis and SPLICE1", *IEEE International Conference on Computer-Aided Design*, pp. 139-140, 1983.

[Sal89] R. A. Saleh, A. R. Newton, "The Exploitation of Latency and Multirate Behavior in Circuit Simulation Using Nonlinear Relaxation", *IEEE Transactions on Computer-Aided Design*, pp. 1286-1298, Décembre 1989.

[Sal94] R. A. Saleh, S.-J. Jou, A. R. Newton, **Mixed-Mode Simulation and Analog Multilevel Simulation**, Kluwer Academic Publisher, 1994.

[Shi68] H. Shichman, and D. A. Hodges, "Modeling and Simulation of Insulated Gate Field Effect Transistor Switching Circuits", *IEEE Journal of Solid State Circuits*, Septembre 1968.

[Sir95] I. Sirot, "Modélisation et Caractérisation des Signaux Logiques CMOS", Thèse de Doctorat, Télécom Paris, Décembre 1995.

[Smi87] S. P. Smith, M. R. Mercer, B. Brock, "Demand Driven Simulation : BACKSIM", *ACM/IEEE Design Automation Conference*, pp. 181-187, 1987.

[Ste92] R. Stewart, V. Anjubault, P. Garcin, J. Benkoski, "Automatic Import of Custom Designs into Cell-Based Environment using Switch-Level and Circuit Simulation", *IEEE Proceedings of the European Design Automation Conference (EURO-DAC)*, 1992.

[Sub86] P. Subramaniam, "Modeling MOS VLSI Circuits for Transient Analysis", *IEEE Journal of Solid-State Circuits*, Avril 1986.

[Lel82] E. Lelarasmee, A. E. Ruehli, A. L. Sangiovanni-Vincentelli, "The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* pp. 131-145, Juillet 1982.

[Mel92] E. Melcher, W. Röthig, M. Dana, "Multiple Input Transitions in CMOS Gates", *EUROMICRO*, vol. 35, Septembre 1992.

[Mel93] E. Melcher, "Analyse temporelle de Circuits Combinatoires", Thèse de Doctorat, Télécom Paris, Avril 1993.

[Mur88] J. Y. Murzin, J. B. Guignet, "DESA, a Netlist Desassembler", *BULL Internal Report*, Les Clayes sous Bois, 1988.

[Nab94] A. Nabvi-Lishi, N. C. Rumin, "Inverter Models of CMOS Gates for Supply Currents and Delay Evaluation", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1271-1279, 1994.

[Nag75] L. W. Nagel, "SPICE2 : A Computer Program to Simulate Semiconductor Circuits", University of California, Berkeley, Memo No. ERL-M520, Mai 1975.

[Naj90] F. N. Najm, R. Burch, P. Yang, I. N. Hajj, "Probalistic simulation for reliability analysis of CMOS VLSI circuits", *IEEE Transactions on Computer-Aided Design*, pp. 439-450, Avril 1990.

[Naj91] F. N. Najm, "Transition density, a stochastic measure of activity in digital circuits", *ACM/IEEE Design Automation Conference,* pp. 644-649, Juin 1991.

[Neb86] W. Nebel, "Automatic Extraction of RT-Level Description from Integrated Circuit Layout Data", Thèse de Doctorat, Université de Kaiserlauten, 1986.

[New78] A. R. Newton, "The Simulation of Large Scale Integrated Circuits", Thèse de Doctorat, Université de Californie, Berkeley, ERL Memo. ERL-M78/52, Juillet 1978.

[New83] A. R. Newton, A. L. Sangiovanni-Vincentelli, "Relaxation-Based Electrical Simulation", *IEEE Transactions on Computer-Aided Design*, pp. 308-331, Octobre 1983.

[Ove86] D. Overhauser, I. N. Hajj, V. B. Rao, "Switch-Level Timing Analysis of VLSI MOS Circuits", *IEEE International Symposium on Circuits and Systems*, pp. 761-764, Mai 1986.

[Pen81] P. Penfield, J. Rubinstein, "Signal Delay in RC tree networks", *ACM/IEEE Design Automation Conference*, pp. 613-617, Juin 1981.

[Pig94] C. Piguet, J.-M. Masgonty, S. Cserveny, E. Dijkstra, "Low-Power Low-Voltage Digital CMOS Cell Design", *PATMOS'94*, 1994.

[Haj92] A. Hajjar, "Modélisation des Temps de Propagation et Analyse Temporelle Statique des Circuits Intégrés CMOS", Thèse de Doctorat, Université de Paris VI, Juillet 1992.

[Ho75] C. W. Ho, A. E. Ruehli, P. A. Brennan, "The Modified Nodal Approach to Network Analysis", *IEEE Transactions on Circuits and Systems*, pp. 504-509, Juin 1975.

[HSP92] HSPICE User's Manual, volume 2, chapitre 7, Meta–Software, Inc. 1992.

[IEE88] IEEE Computer Society, **IEEE Standard VHDL Language Reference Manual**, The Institute of Electrical and Electronic Engineering, New York, Publ. No : IEEE Standards Coordinating committee 20, 1988.

[Jep69] J. Jephson, R. McQuarrie, R. Vogelserg, "A Three-Value Computer Design Verification System", *IBM Systems Journal*, pp. 178-188, 1969.

[Jun89] Y.-H. Jun, R. Jun, S.-B. Park, "An Accurate and Efficient Delay Time Modeling for MOS Logic Circuits Using Polynomial Approximation", *IEEE Transactions on Computer-Aided Design*, pp. 1027-1032, Septembre 1989.

[Kah96] A. B. Kahng, S. Muddu, "Analysis of RC Interconnections Under Ramp Input", *ACM/IEEE Design Automation Conference*, 1996.

[Kay92] A. I. Kayssi, K. A. Sakallah, T. M. Burks, "Analytical Transient Response of CMOS Inverters", *IEEE Transactions on Circuits and Systems*, pp. 1347-1350, Janvier 1992.

[Kay93] A. I. Kayssi, K. A. Sakallah, T. Mudge, "The Impact of Signal Transition Time on Path Delay Computation", *IEEE Transactions on Circuits and Systems–II*, pp. 302-309, 1993.

[Kia93] P. Kiani, "TAS – Manuel de Paramétrisation", Rapport Technique, Laboratoire MASI–URA CNRS 818, Université de Paris VI, Juillet 1993.

[Kon95] J. T. Kong, K. A. Sakallah, "Methods to Improve Digital MOS Macromodel Accuracy", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 868-881, 1995.

[Kos91] T. Kostelijk, B. De Loore, "Automatic Verification of Library-Based IC Designs", *IEEE Journal of Solid-State Circuits*, 1991.

[Lau92] M. Laurentin, A. Greiner, R. Marbot, "DESB, a Functional Abstractor for CMOS VLSI", *IEEE proceedings of the European Design Automation Conference*, 1992.

[Lau94] M. Laurentin, "Abstraction Fonctionnelle de Circuits Intégrés CMOS", Thèse de Doctorat, Université de Paris VI, Décembre 1994.

[Bry83] R. E. Bryant, "Race Detection in MOS Circuits by Ternary Simulation", F. Anceau Editor. North-Holland Publishing Company, 1983.

[Bry84] R. E. Bryant, "A Switch-Level Model and Simulator for MOS Digital Systems", *Transactions on Computers*, pp. 160-177, Février 1984.

[Bry87] R. E. Bryant, "Boolean Analysis of MOS Circuits", *IEEE Transactions on Computer Aided Design*, 1987.

[Cha75] B. R. Chawla, H. K. Gummel, P. Kozak, "MOTIS An MOS Timing Simulator", *IEEE Transactions on Circuits and Systems*, pp. 901-910, Décembre 1975.

[Cha88] R. Chadha, C.-F. Chen, "Extension of a transistor level digital timing simulator to include first order analog behavior", *IEEE International Conference on Computer Design*, pp. 116-119, 1988.

[Cha96] V. Chandramouli, K. A. Sakallah, "Modeling the Effects of Temporal Proximity of Input Transitions on Gate Propagation Delay and Transition Time", *Design Automation Conference*, 1996.

[Des69] C. A. Desoer, E. S. Kuh, **Basic Circuit Theory**, McGraw Hill, 1969.

[Dev89] P. Deverchère, "The Layout Extractor DAX", Rapport Technique, BULL S.A., Les Clayes sous Bois, 1989.

[Dio97] K. Dioury, A. Greiner, and M.-M. Rosset-Louërat "Accurate Static Timing Analysis for Deep Submicronic CMOS Circuits", *IEEE VLSI'97 Conference*, Août 1997.

[Duh94] M. Duhalde, "TAS – Manuel de Paramétrisation", Rapport Technique, BULL S.A., Les Clayes sous Bois, Juillet 1994.

[Dun96] J. Dunoyer, N. Abdallah, P. Bazargan-Sabet, "A symbolic simulation approach in resolving signals' correlation", *ACM/IEEE Annual Simulation Symposium*, pp. 203-211, Avril 1996.

[Fro94] C. Froidevaux, M.-C. Gaudel, M. Soria, **Types de Données et Algorithmes**, Ediscience International, 1994.

[Gre92] A. Greiner, F. Pêcheux, "ALLIANCE, a Complete Set of CAD Tools for Teaching VLSI Design", *Proceedings of the Eurochip Workshop on VLSI Design Training*, pp. 230-237, 1992.

[Hac71] G. D. Hachtel, R. K. Brayton, F. G. Gustavson, "The Sparse Tableau Approach to Network Analysis and Design", *IEEE Transactions on Circuits Theory*, pp. 101-113, Janvier 1971.

[Haj91] A. Hajjar, R. Marbot, A. Greiner, P. Kiani, "TAS : An Accurate Timing Analyzer for CMOS VLSI", *IEEE Proceedings of the European Design Automation Conference (EDAC)*, pp. 261-265, Février 1991.

BIBLIOGRAPHIE

[Abd94] N. Abdallah, P. Kiani, A. Hajjar, "New Algorithms for Static and Dynamic Sensitization of Paths in Timing Analysis", *IEEE Southeastern Symposium on System Theory*, pp. 385-389, Mars 1994.

[Abd95a] N. Abdallah, J. Dunoyer, P. B. Sabet, "A New Generation of VLSI CAD Tools Based on Probability", *IEEE Southeastern Symposium on System Theory*, pp. 348-352, Mars 1995.

[Abd95b] N. Abdallah, P. Bazargan-Sabet, A. Greiner, "On the Design of Mixed-Mode Simulators for Modern VLSI Circuits", *IEEE Midwest Symposium on Circuits and Systems*, pp. 1168-1171, Août 1995.

[Abd97a] N. Abdallah, P. Bazargan-Sabet, "Technique de Simulation Event-Driven en Vue d'une Simulation Mixte Efficace", *Colloque CAO de Circuits Intégrés et Systèmes*, pp. 22-25, Janvier 1997.

[Abd97b] N. Abdallah, P. Bazargan-Sabet, J. Dunoyer, "SWISSE : A Fast Switch-Level Timing Simulator with Slope Effects for Large Digital MOS Circuits", *IEEE International Conference on Electronics, Circuits, and Systems*, pp. 875-879 Décembre 1997.

[Abd97c] N. Abdallah, P. Bazargan-Sabet, J. Dunoyer, "Application de la Simulation Logico-Temporelle à l'Évaluation de la Consommation", *Journées d'Études sur la Faible Tension – Faible Consommation*, Novembre 1997.

[Acu89] E. Acuna, J. Dervenis, A. Pagones, R. Saleh, "iSPLICE3 : A New Simulator for Mixed Analog/Digital Circuits", *Custom Integrated Circuits Conference Digest of paper*, pp. 13.1.1-13.1.4, Mai 1989.

[Acu90] E. Acuna, J. Dervenis, A. Pagones, F. L. Yang, R. Saleh, "Simulation Techniques for Mixed Analog/Digital Circuits", *IEEE Journal of Solid-State Circuits*, pp. 353-362, Avril 1990.

[Agr96] V. D. Agrawal, "Low-Power Design by Hazard filtering", *PATMOS'96*, 1996.

[Auv90] D. Auvergne, N. Azemard, D. Deschacht, M. Robert, "Input Waveform Slope Effects in CMOS Delays", *IEEE Journal of Solid-State Circuits*, pp. 1588-1589, Avril 1990.

[Baz93] P. Bazargan Sabet, A. Greiner et H. N. Vuong "ASIMUT : A Public Domain VHDL Simulation Tool", *4th Eurochip Workshop*, pp. 62-65, Septembre 1993.

[Boe88] M. Boehner, "LOGEX : An Automatic Logic Extractor from Transistor Level for CMOS Technology", *ACM/IEEE Design Automation Conference*, 1988.

```
#Fichier de parametrisation de TAS & SWISSE, PROL10, LEVEL X, HSPICE
#Les parametres sont calcules en tenant compte des shrinks
#Par DIOURY karim, le 15/06/1995

Technologie : prol10 Version : 2.0

#Reference Simulator
ESIM  = HSPICE
MODEL = MOS
LEVEL = X.X

#shrink parameters (micron)
DLN = 0.0
DLP = 0.0
DWN = 0.0
DWP = 0.0

#transistor characteristics
#NMOS
VTN = 0.8
BN  = 0.7
AN  = 4.8e-05
RNT = 11000.0

#PMOS
VTP = 1.2
BP  = 0.25
AP  = 1.7e-05
RPT = 26000.0

#general parameters
VDDmax = 4.50
VTHR   = 2.250
VSSdeg = 1.6
VDDdeg = 3.5
TEMP   = 70.0

#dynamic capacitance: grid capacitance (in pF/u and pF/u2)
CGSN = 1500.0e-6
CGSP = 1500.0e-6
CGPN =  200.0e-6
CGPP =  200.0e-6

#dynamic capacitance: drain capacitance (in pF/u and pF/u2)
CDSN = 400.0e-6
CDSP = 600.0e-6
CDPN = 500.0e-6
CDPP = 800.0e-6
CDWN = 200.0e-6
CDWP = 200.0e-6

#dynamic capacitance: source capacitance (in pF/u and pF/u2)
CSSN = 400.0e-6
CSSP = 600.0e-6
CSPN = 500.0e-6
CSPP = 800.0e-6
CSWN = 200.0e-6
CSWP = 200.0e-6
```

Annexe 4

EXEMPLE DE FICHIER TECHNOLOGIQUE UTILISÉ PAR TAS & SWISSE

Ces registres sont composés de deux latchs préalablement détectés par DESB formant une bascule maître–esclave.

Pour le premier type de registre, l'inversion entre la commande du maître M et celle de l'esclave S doit être générée par un nombre impair d'inverseurs. La donnée peut être générée par n'importe quel réseau dont le comportement est dual.

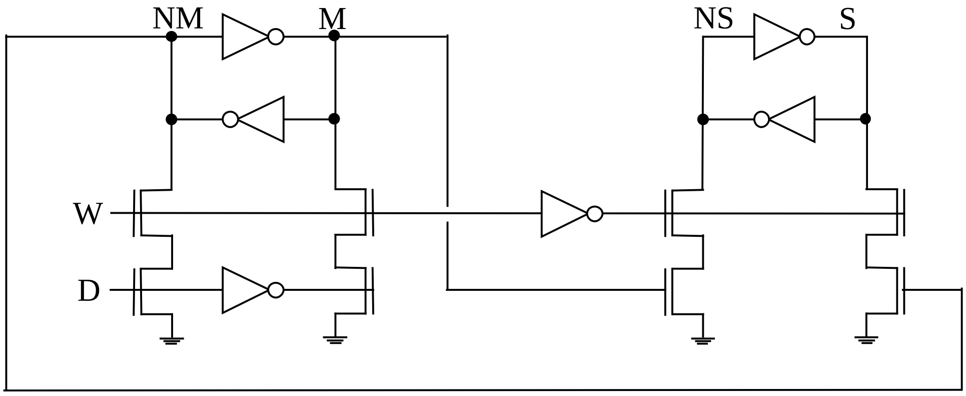


Figure A3.4 : Registre reconnu par DESB

Dans le cas des registres, c'est toujours le latch esclave S qui est ensuite considéré comme l'élément mémorisant. C'est donc lui qui fait l'objet d'une description comportementale.

3. Les bleeders

Les bleeders peuvent être utilisés pour améliorer le niveau d'un signal électriquement dégradé. Ils sont eux aussi constitués du rebouclage de deux cônes et doivent être distingués des latchs. Les configurations reconnues sont présentées sur la Figure A3.5.

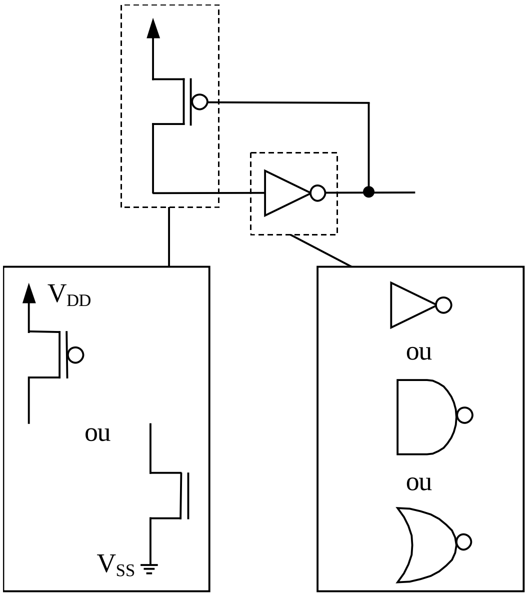
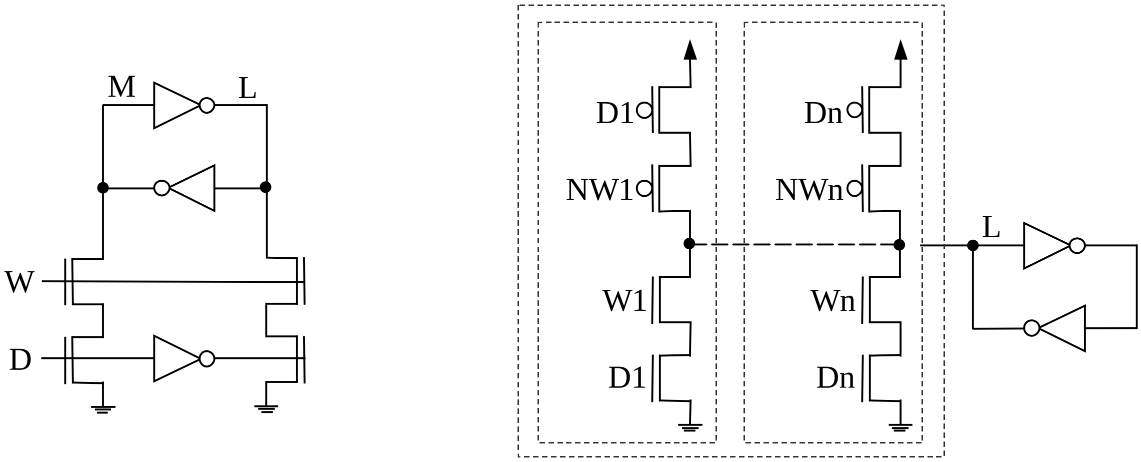


Figure A3.5 : Bleeders reconnus par DESB ou une porte de transfert (d). La donnée est générée par n'importe quel réseau dont le comportement est dual (e); ce n'est pas nécessairement une porte CMOS duale.

Deux types de latchs supplémentaires sont reconnus et sont illustrés sur la Figure A3.2.



a) b)

Figure A3.2 : Un autre latch reconnu par DESB

La Figure A3.2–a présente un latch systémique. Les nœuds M et L ne peuvent contrôler que des grilles de transistors. C'est le nœud L qui est considéré comme le latch. Le nœud M est simplement décrit comme l'inverse du nœud L.

Pour le latch de la Figure A3.2–b, la donnée et la commande attaquent un inverseur trois états. Le nombre d'entrées du latch correspond au nombre d'inverseurs trois états qui écrivent sur le nœud L.

3. Les registres

Les deux modèles de registres reconnus par DESB sont présentés Figure A3.3 et Figure A3.4.

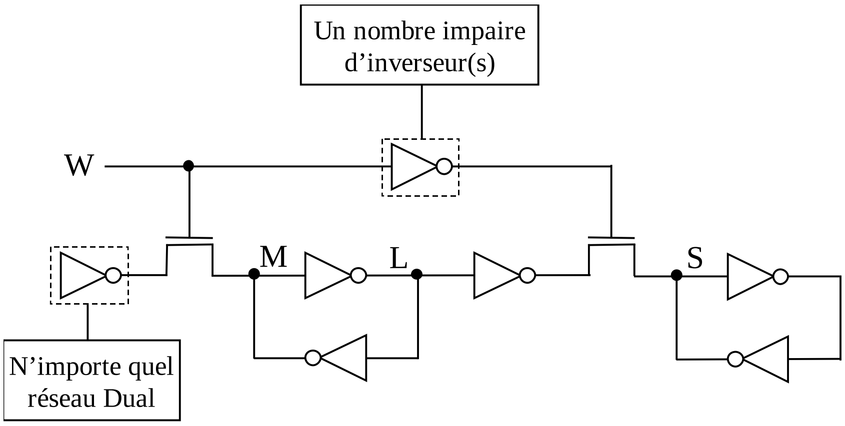


Figure A3.3 : Registre général reconnu par DESB

1. Introduction

Cette annexe est une reproduction de l'Annexe 3 de la thèse de M. LAURENTIN [Lau94]. Toutefois, il faut signaler que depuis, beaucoup d'améliorations ont été apportées à l'outil d'abstraction fonctionnelle DESB. Une de ces améliorations concerne justement la reconnaissance des points mémorisants.

En effet, outre les latchs et registres présentés dans cette annexe, DESB est capable dans sa version actuelle de reconnaître n'importe quel point mémorisant préalablement décrit par l'utilisateur sous la forme d'un réseau de transistors.

2. Les latchs

La Figure A3.1 présente le modèle général des latchs reconnus par DESB.

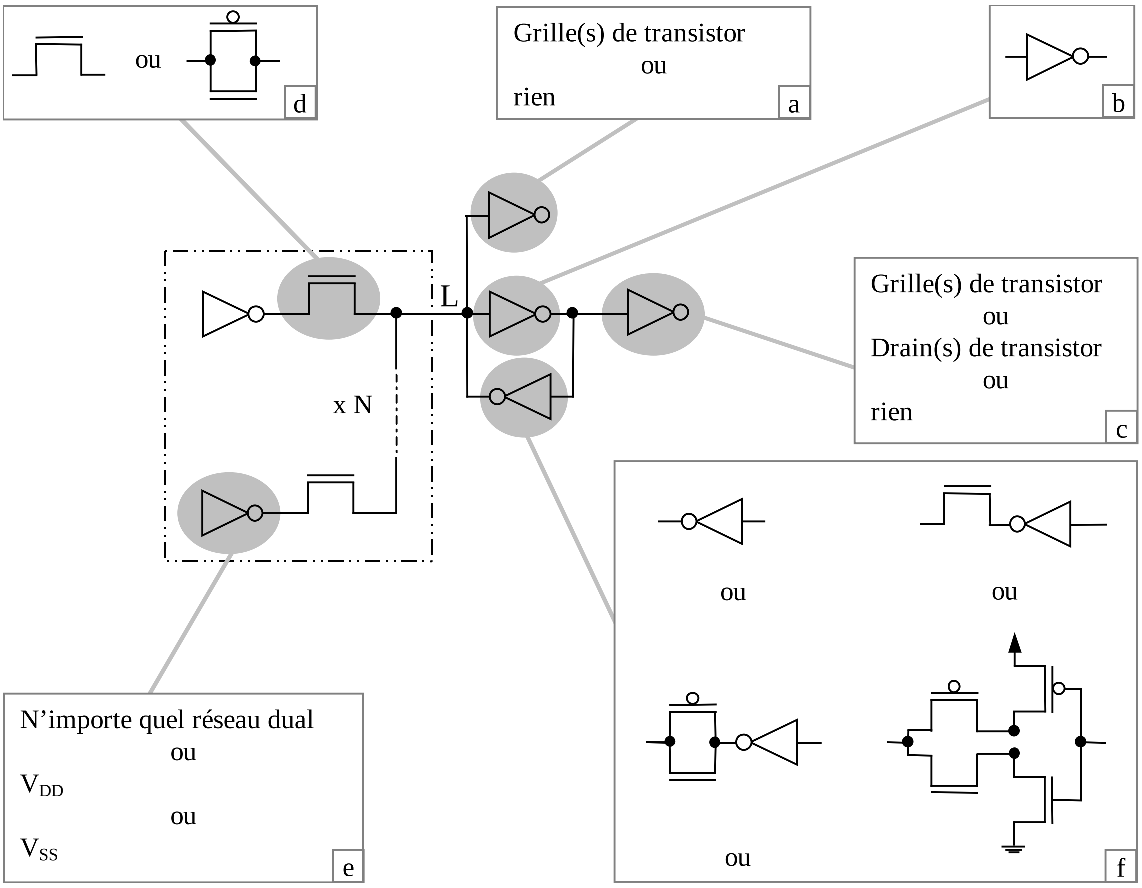


Figure A3.1 : Latch général reconnu par DESB

Le latch est constitué d'un inverseur (b) et d'éléments de rebouclage. La circuiterie de rebouclage peut être constituée par un des quatre motifs représentés sur la figure (f). Le nœud mémorisant est le signal L. Ce nœud peut contrôler une ou plusieurs grilles de transistors (a). Le latch peut avoir plusieurs entrées. On écrit dans le latch à travers un transistor de passage

Annexe 3

LATCHS ET REGISTRES RECONNUS PAR DESB

1. Introduction
2. Les latchs
3. Les registres
4. Les bleeders

minimiser les écarts entre les deux fonctions elles mêmes ou encore chercher à minimiser les écarts entre les deux fonctions résultant de la réponse d'une porte logique aux deux fonctions d'origine V(t) et $\tilde{V}(t)$.

Bien que moins rigoureuse mathématiquement, nous avons choisi la deuxième approche car elle tient mieux compte de l'objectif recherché, à savoir, une simulation la plus précise possible. Notons que ce choix se justifie également par le fait que l'importance du front n'est pas la même à tout instant de la commutation. Des simulations SPICE intensives indiquent que le meilleur résultat est obtenu quand la tangente hyperbolique est traversée par la droite aux instants de passage du signal par $V(t) = V_{TN}$, et $V(t) = \mathrm{th}(1)$ (Figure A2.1). Plus formellement, les paramètres de la tangente hyperbolique $\tilde{V}(t) = V_{TN} + (V_{DD} - V_{TN})\,\mathrm{th}\left((t - t_{VTN})/F\right)$ se calculent de la manière suivante :

$$F = \frac{V_{DD}\,(3V_{DD} - 4V_{TN})}{4|a|\,(V_{DD} - V_{TN})} \tag{A2.6}$$

$$t_{VTN} = t_{vss} - 0{,}5V_{DD}\ln\left((V_{DD} - 2V_{TN}) / V_{DD}\right) \tag{A2.7}$$

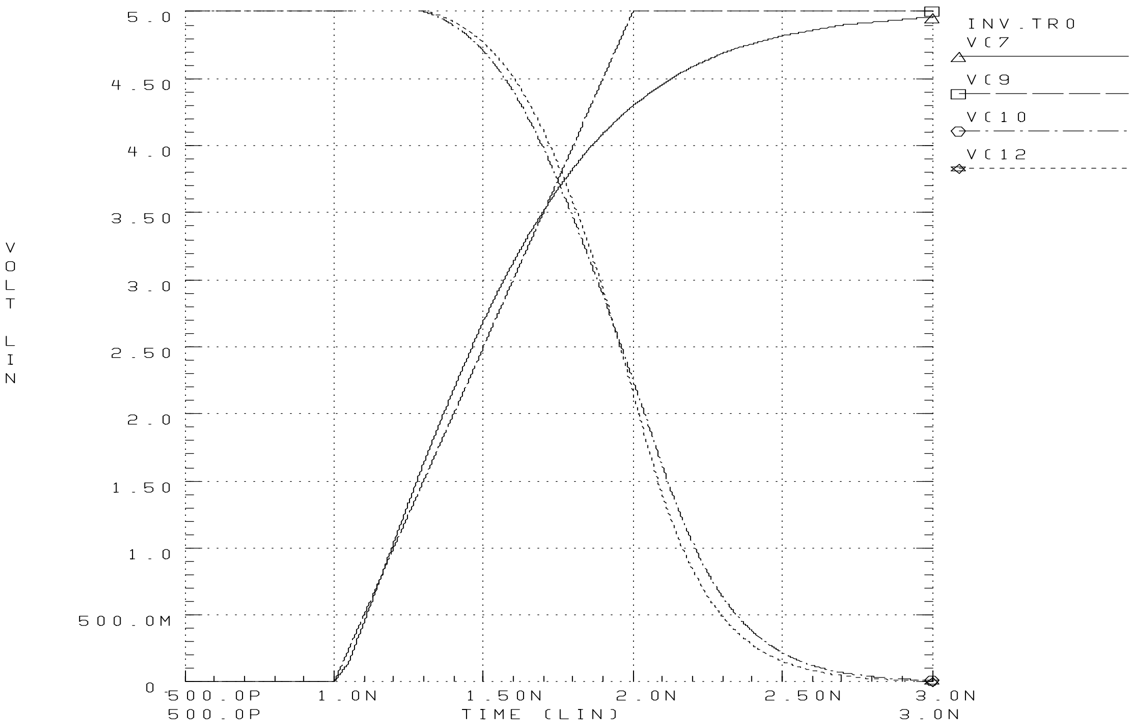


Figure A2.1 : Simulation d'une rampe et de sa fonction approchante en tangente hyperbolique

de t_{fin}, une fonction $Q_n'(t)$ dans laquelle $(V_{gs} - V_T) = V_{fin}$. Ainsi, nous sommes obligés, dans ce cas, d'intégrer le courant par morceaux.

La deuxième remarque est qu'une telle approche brise l'homogénéité de notre modèle et ouvre la voie à des ajouts ultérieurs qui risquent de compliquer l'ensemble.

C'est pourquoi nous optons pour une deuxième solution dans laquelle la rampe est approchée par une tangente hyperbolique avant d'être injectée dans le modèle général de simulation.

3. Approximation par une tangente hyperbolique

Considérons un signal qui commute de V_{SS}, à l'instant t_{vss}, vers V_{DD}, suivant une rampe de pente A. L'équation du signal s'écrit

$$V(t) = \begin{cases} V_{SS} & \text{pour} \quad t < t_{vss} \\ A(t - t_{vss}) + V_{SS} & \text{pour} \quad t_{vss} \le t \le t_{vdd} = t_{vss} + (V_{DD} - V_{SS})/A \\ V_{DD} & \text{pour} \quad t > t_{vdd} \end{cases} \qquad (A2.5)$$

Le but est d'approcher ce morceau de droite par une fonction tangente hyperbolique $\tilde{V}(t)$, telle qu'elle est définie par l'équation 4.5. Deux approches sont possibles : chercher à

1. Introduction

Tous les signaux au sein du simulateur sont représentés par une fonction tangente hyperbolique. Or, il est possible parfois d'avoir sur les connecteurs d'entrée des signaux représentés par des modèles différents. Ceci est d'autant plus important que nous cherchons à faire traiter par notre simulateur des stimuli au format de simulateurs industriels connus. Typiquement, dans SPICE, on peut modéliser les transitions des entrées par différentes fonctions.

Deux méthodes sont alors possibles. La première consiste à faire traiter par le simulateur tout nouveau modèle de signal en l'intégrant à l'équation du courant pour générer un modèle temporel spécifique. La deuxième consiste à approcher chacune de ces fonctions par une tangente hyperbolique. A titre d'exemple, nous allons étudier le cas d'une rampe qui est la forme la plus couramment rencontrée.

2. Modèle spécifique à une rampe

Dans cette approche, Le courant est toujours exprimé par les formules du modèle MCC, et la tension d'entrée par l'équation d'une droite :

$$I_n(t) = (W/L)\,A\,(V_{gs} - V_T)^2 \Big/ \left[1 + B\,(V_{gs} - V_T)\right] \qquad \text{(A2.1)}$$

$$V_{gs} - V_T = a\,(t - t_T) \qquad \text{(A2.2)}$$

$$I_n(t) = I_{max}\,\frac{(1+BU)\,a^2\,(t-t_T)^2}{U^2\,\big(1 + B\,a\,(t-t_T)\big)} \qquad \text{(A2.3)}$$

A l'instar du modèle utilisant une tangente hyperbolique, nous trouvons la charge perdue par la capacité à travers le transistor en intégrant l'Équation A2.3, soit :

$$Q_n(t) = \int I_n(t)dt = I_{max}\,\frac{(1+BU)a^2}{U^2}\left(-\frac{1}{B^2a^2}(t-t_T) + \frac{1}{2Ba}(t-t_T)^2 + \frac{1+B^2a^2}{B^3a^3}\ln(t-t_T)\right) \text{(A2.4)}$$

Ensuite, une inversion par des méthodes numériques permet de calculer l'instant qui nous intéresse, à savoir, l'instant où la charge a atteint C $V_{DD}/2$.

Nous pouvons faire deux reproches à ce modèle. Tout d'abord, le domaine de définition de (V_{gs} - V_T) dans l'Équation A2.2 est compris entre l'instant de début de la transition, $t_{début}$, et l'instant de fin, t_{fin}. Par conséquent $I_n(t)$ n'est pas continue et présente un point anguleux au temps t_{fin}. Si, $Q_n(t)$ n'a pas atteint la valeur C $V_{DD}/2$ avant t_{fin}, il est nécessaire d'utiliser, au delà

Annexe 2

MODÉLISATION D’UNE RAMPE EN ENTRÉE

.TITLE [titre]

titre	remplace la première ligne qui représente le titre par défaut.

Analyse transitoire

Forme générale

.TRAN [pas_d_analyse] [instant_d_arrêt] [START=instant_de_début]

pas_d_analyse, instant_d_arrêt et START indiquent respectivement le pas d'analyse, l'instant d'arrêt de l'analyse utilisant le pas précédent et le temps à partir duquel le simulateur effectue la sauvegarde des résultats.

Conditions initiales pour analyse transitoire

Forme générale

.IC V(nœud1)=valeur1 [V(nœud2)=valeur2...]

nœud$_i$	correspond au nom d'un nœud qui peut être hiérarchique.
valeur$_i$	indique la tension à appliquer à ce nœud.

Cette carte permet d'indiquer les valeurs initiales d'une analyse transitoire.

Édition des résultats

Forme générale

.PRINT TRAN V(nœud1) [V(nœud 2)...]
.PROBE TRAN V(nœud1) [V(nœud 2)...]

TRAN	indique que l’analyse sera transitoire, donc temporelle.
nœud$_i$	correspond au nom d'un nœud qui peut être hiérarchique.

Dans notre cas, ces deux cartes sont équivalentes. Elles permettent d'indiquer au simulateur les signaux qui sont à surveiller.

Pour la bonne interprétation de ce signal, il est nécessaire que les paramètres qui suivent le mot clé PULSE soient mentionnés de façon explicite.

Exemple

Vi1 3 0 PULSE (0v 5v 22Ns 10Ps 10Ps 50Ns 100Ns)

Signal linéaire par segment

Forme générale

Vxxx n+ n– PWL [(] t1 v1 [t2 v2] [R[=repeat]] [TD=delay] [)]

Les t_i doivent absolument être ordonnés par ordre croissant.

R	permet d'indiquer le point de répétition. Sa valeur associée doit nécessairement appartenir à l'ensemble formé par tous les t_i excepté le dernier. Si cette valeur n'est pas mentionnée de façon explicite alors le point de répétition par défaut est le premier.
TD	permet d'indiquer la valeur du décalage temporel que l'on souhaite voir appliquer au signal.

Exemple

Vtoto 3 0 PWL 1n 1v, 2n 5v, 2.25n 5.65v, 10n 1v, 11n 3v R 10n 1v, 11n 3v R 10n TD=1n

Options

forme générale

.OPT POST=[Numéro] ACCT=[niveau] PROBE

POST	active l'interface de sortie HSPLOT. Cette sortie est en ASCII pour "numéro" égal à 1 et en binaire pour 2.
ACCT	permet la génération d'un rapport contenant les temps d'exécution de la simulation ainsi que les différences statistiques. Sa valeur associée permet d'indiquer le niveau d'information à générer.
PROBE	limite la sortie aux variables explicitement demandées.

Titre

Forme générale

paire de [ ]. Dans ce cas, la paire de [ ] se doit d'encadrer un nombre au format entier et n'être suivie d'aucun symbole pouvant être considéré comme appartenant au nom.

Exemple

∗ net 3 horloge

∗ net 7 = commande[4]

Sources indépendantes

Forme générale

Vxxx n+ n– [[DC[=]] dcval] [fonction_transitoire]

n+	Nœud par lequel entre le courant positif. Cette quantité doit impérativement être un entier strictement positif.
n–	Nœud par lequel sort le courant. Pour le simulateur SWISSE, cette quantité doit dans tous les cas être nulle. Ceci est dû au fait que nous supposons toutes les différences de potentiels référencées par rapport à la masse qui correspond au nœud zéro.
DC	permet d'indiquer la valeur de la source continue. Si cette valeur est suivie de la déclaration d'une source transitoire, alors seule cette dernière est prise en compte.

Si elle est présente, la fonction_transitoire indique une des fonctions suivantes :

Signal impulsionnel périodique

Forme générale

Vxxx n+ n– PULSE [(] v1 v2 td tr tf pw per [)]

v1, v2	indiquent respectivement la valeur initiale (plateau inférieur) et finale (plateau supérieur) du signal.
td	indique le retard précédant la première pente montante.
tr	indique le temps de montée.
tf	indique le temps de descente.
pw	indique la durée du plateau supérieur.
per	indique la valeur de la période.

Nous présentons ici un résumé des directives de simulation SPICE acceptées par le simulateur SWISSE. On trouvera une description détaillée de ces commandes dans le guide de l'utilisateur de SPICE de Berkeley ou des différentes versions commerciales de SPICE.

Commentaire

Forme générale

∗ *n'importe quel texte sauf "net"*

Une instruction commençant avec * est une ligne de commentaire et ne produit aucun effet. Toutefois, le simulateur SWISSE utilise cette directive afin de nommer des nœuds dans le circuit comme ceci est expliqué un peu plus loin dans l'annexe. Dans ce cas, le caractère * est suivi du mot clé "net".

Exemple

```
∗ net    Voici un exemple de commentaire
```

Commentaire en ligne

Forme générale

texte du fichier circuit ; n'importe quel texte

Un ; est considéré comme une fin de ligne, de sorte que le simulateur poursuit son analyse à la ligne suivante du fichier. Le texte après le ; est un commentaire et ne produit aucun effet.

Exemple

```
Vi0    3    0    4.5V                    ; Source de tension sur i0
```

Affectation de noms aux nœuds

Forme générale

∗ *net*   *n*   *[=] nom*

n   est le numéro identifiant le nœud.

nom   est le nom associé au nœud.

Cette instruction est propre au simulateur SWISSE. Elle permet d'affecter un nom à un nœud afin de rendre plus conviviale l'interface utilisateur. Un nom peut contenir au plus une

Annexe 1

SOUS-ENSEMBLE DES STIMULI SPICE SUPPORTÉ PAR SWISSE

que la grande majorité des portes dans un circuit sont des portes simples et la majorité des transitions sont sans collisions, il y a de fortes chances pour que les événements successifs sur un même signal d'entrée aient des caractéristiques temporelles (des pentes) très voisines. Un cache distribué pourrait préserver le résultat du calcul de délai associé aux paramètres des événements de l'entrée et à la capacité de sortie. Ainsi, le délai pourra être retrouvé sans calcul supplémentaire, si cette même configuration se reproduit.

Enfin, une des retombées possibles du travail présenté ici concerne un domaine d'actualité que représente l'évaluation de la consommation [Abd97c]. Le calcul de la consommation présente les mêmes contraintes que la simulation temporelle, à savoir, il est difficile d'avoir à la fois des évaluations précises et rapides.

Il existe actuellement des méthodes d'évaluation probabilistes basées sur le nombre probable de commutations des nœuds du composant [Naj90] [Naj91] [Abd95a] [Dun96]. Les résultats de ce mode de calcul ne peuvent, par essence même, être qu'une approximation de premier ordre de la consommation.

La rapidité du simulateur SWISSE, due à notre approche événementielle, lui permet de faire une analyse des circuits, quelle qu'en soit la taille. Étant donné que les délais sont calculés en utilisant les caractéristiques courant–tension, on pourrait aisément en déduire la puissance dissipée tout en tenant compte des collisions et des effets qu'elles génèrent comme les "glitches" [Agr96] [Rab96].

une fonction à un seul paramètre, représentant la pente du signal. Cette observation est confortée par le fait qu'un concepteur est surtout intéressé par la forme générale du signal lors d'une commutation.

Toutefois, la configuration à l'entrée d'une porte ne se limite pas toujours à un simple événement sur une des entrées. Il arrive en effet que plusieurs événements interviennent dans une certaine proximité temporelle telle qu'ils modifient la configuration électrique de la porte, perturbant ainsi la commutation de la sortie initialement prévue. Ce phénomène de collisions peut produire des effets d'accélération, de retard ou de transitions incomplètes. Son ignorance est susceptible de générer des erreurs qui peuvent, dans certains cas, dépasser 100%. Dans cette thèse, nous avons donné une définition claire du phénomène de collisions entre les événements. Ensuite, pour chaque type de collisions, nous avons proposé un modèle en adéquation avec l'importance des effets qu'il produit. En résumé, le traitement proposé est ramené à celui d'un événement simple tout en tenant compte des changements apportés aux courants dans la porte. Le nouvel instant de passage de la sortie par la tension $V_{DD}/2$ est calculé pour ensuite en déduire un événement traduisant le changement dans la forme d'onde en cours de transition. Nous avons de la sorte réussi à garder une précision qui satisfasse nos objectifs initiaux.

Un outil logiciel s'appuyant sur cette méthode, SWISSE, a été développé. Ce programme a permis de montrer la pertinence des modèles temporels choisis ainsi que l'efficacité de la méthode de simulation proposée.

Aujourd'hui, ce travail peut évoluer dans les directions suivantes :

Premièrement, la course à l'infiniment petit a un impact significatif sur les outils de vérification temporelle. A 0,5 micron et en dessous, les résistances d'interconnexion ne peuvent plus être négligées. Plus encore, les interconnexions sont devenues la source principale des délais dans les circuits intégrés fortement submicroniques, et il est nécessaire de les intégrer dans un simulateur temporel afin de garder un niveau de précision équivalent. La méthode PÉCF, ainsi que l'architecture de SWISSE permettent en principe d'étendre le module de calcul de délais pour prendre en compte un délai supplémentaire d'interconnexions de type RC [Kah96].

Deuxièmement, une accélération logicielle significative du simulateur est possible par la mise en œuvre d'un mécanisme de cache pour le calcul des délais. Partant de la constatation

L'évolution de la technologie des circuits intégrés CMOS permet depuis quelques années d'intégrer des petites fonctions analogiques à un cœur de circuit numérique de plusieurs millions de transistors. La vérification fonctionnelle de ce type de circuits pose des problèmes aux simulateurs actuels qui sont, soit trop lents pour valider des circuits entiers, soit peu précis sur la partie numérique pour répondre à l'exigence imposée par la partie analogique. Les concepteurs sont en quête d'applications capables de garantir un fonctionnement temporel correct, pourvu que la rapidité de l'outil permette la vérification du circuit dans son intégralité, et ce en un temps acceptable. Ce type de simulateurs trouve également sa place dans le processus de vérification des circuits intégrés numériques de hautes performances.

Dans cette thèse, nous avons cherché à apporter une solution à ce problème : développer une méthode de simulation logico–temporelle pour des circuits numériques qui, tout en conservant une vitesse de simulation proche de celle des simulateurs logiques à pilotage événementiel, permette d'obtenir dans le domaine temporel une précision proche de celle des simulateurs électriques de type SPICE.

Nous partons d'un réseau de portes orientées obtenu à la suite d'une phase d'abstraction fonctionnelle appliquée à une représentation du circuit sous la forme d'un réseau de transistors et capacités, lui même extrait à partir du dessin des masques. Nous avons pu par conséquent élaborer une approche de simulation de type événementiel et tirer ainsi profit du faible taux d'activité des circuits.

Ensuite, nous avons proposé une méthode originale de simulation à pilotage événementiel sur changement de front (PÉCF). Par contraste avec les approches classiques, dans notre méthode un événement correspond à un changement dans la valeur du front du signal. Une telle définition de la notion d'événement permet de réduire considérablement le nombre de calculs temporels effectués dans les simulateurs temporels existants. En effet, pour un événement intervenant sur une des entrées, la détermination des paramètres de l'événement sur la sortie passe par le calcul de l'instant de passage de cette sortie par la tension $V_{DD}/2$, à partir de l'expression des courants dans le réseau des transistors constituant la porte.

Les modèles temporels utilisés s'appuient sur les caractéristiques courant–tension des transistors MOS submicroniques ainsi que sur la forme d'onde du signal qui commande la grille des transistors. Une bonne modélisation des signaux qui permet d'inclure l'effet de la pente est un facteur qui ne peut être négligé dans les modèles actuels. Or, l'expérience nous a montré que, dans les parties numériques CMOS, la forme d'onde des signaux peut être modélisée par

Chapitre 9

CONCLUSION

8.4. Conclusion

Nous avons montré dans ce chapitre qu'il est possible de réaliser un outil logiciel effectuant une simulation telle que nous l'avons décrite aux chapitres 5 et 6.

Un taux d'activité observé de moins de 10% quel que soit le circuit simulé confirme l'intérêt d'une méthode de simulation à pilotage événementiel. La méthode que nous avons présentée exploite au maximum le faible taux d'activité des circuits en utilisant, dans la majorité des cas, un événement pour représenter une commutation et un traitement pour déduire l'événement en sortie. Le résultat est une vitesse d'exécution 1000 fois plus rapide que celle de SPICE, capable de traiter des circuits très complexes hors de portée des simulateurs électriques. La précision est maintenue à un niveau acceptable grâce à un traitement adéquat des collisions entre les événements, qui permet de rendre compte de la dynamique des circuits.

Une synthèse de la performance de SWISSE sur ce circuit est donnée dans le Tableau 8.6.

Tableau 8.5 : Temps de simulation du DLX–m par SWISSE

Chargement du circuit	04,8s
Désassemblage du circuit	19m 12,6s
Chargement des stimuli	00,5s
Initialisation du circuit à t = 0	04,5s
Simulation du circuit jusqu'à t = 102700000 pS	05m 04,3s

Tableau 8.6 : Performance de SWISSE sur le DLX–m

# Cycles	**# Événements**	**# Cycles/Sec**	**Temps total**
340	148386	1,12	24m 27s

8.3.3.3. Un circuit complexe : RCUBE

RCUBE est un routeur de paquets à hautes performances réalisé au laboratoire LIP6 [Rei97]. Il est conçu autour d'un crossbar non bloquant 8x8 et offre une latence de routage de 150 nanosecondes, ainsi qu'un débit global de 640 Mégaoctets/seconde à 80 Mégahertz. La complexité du cœur du routeur (sans ses 8 liaisons série) atteint 275832 transistors.

De même que pour le DLX–m, nous utilisons comme stimuli d'entrée du simulateur SWISSE, le fichier de stimuli qui a servi à la validation fonctionnelle du circuit. Ceci représente 4433 cycles de 12,5 nanosecondes.

Les temps des différentes phases de simulation sont donnés dans le Tableau 8.7 et une synthèse de la performance de SWISSE sur ce circuit est donnée dans le Tableau 8.8.

Tableau 8.7 : Temps de simulation de RCUBE par SWISSE

Chargement du circuit	01m 32,0s
Désassemblage du circuit	67m 36,1s
Chargement des stimuli	42m 59,3s
Initialisation du circuit à t = 0	19,8s
Simulation du circuit jusqu'à t = 102700000 pS	21h 23m 38,0s

Tableau 8.8 : Performance de SWISSE sur le circuit RCUBE

# Cycles	**# Événements**	**# Cycles/Sec**	**Temps total**
4433	105246786	0,06	23hr 23mn 38sec

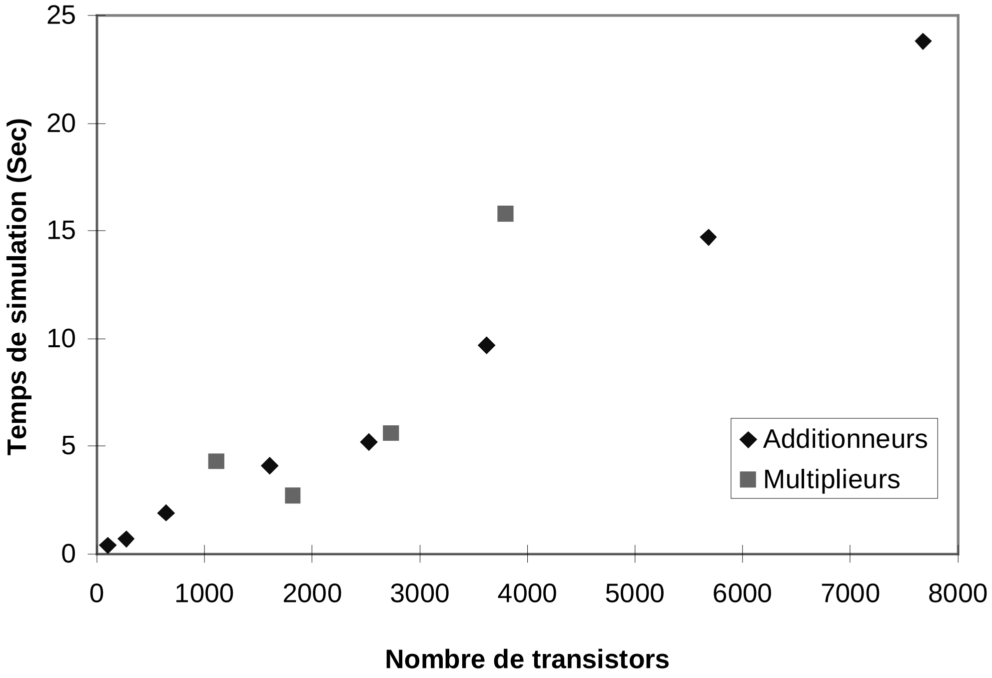


Figure 8.17 : Temps de simulation des additionneurs et multiplieurs

La Figure 8.16 montre que le rapport entre les temps d'exécution va en croissant et dépasse le facteur 1000 pour les circuits de plus de 10.000 transistors. Le facteur 850 obtenu pour l'additionneur 16 bits est dû à des problèmes de convergence de HSPICE constatés lors de la simulation de ce circuit. La Figure 8.17 montre quant à elle que la progression des temps de simulation est linéaire avec l'augmentation du nombre de transistors. Dans le cas du multiplieur, la linéarité est plus faible, mais ceci est dû surtout à la phase de désassemblage et non à la simulation elle même.

8.3.3.2. Un circuit complexe : le DLX–m

Le DLX–m est un micro–processeur 32 bits conçu par [Hen92]. Nous utilisons ici une version microprogrammée (DLX–m) réalisée au laboratoire LIP6 et dont la complexité atteint 37794 transistors.

Les stimuli utilisés pour la simulation du DLX–m sont obtenus à partir du même fichier de patterns que celui utilisé pour la validation fonctionnelle du circuit. Ces stimuli représentent 340 cycles de 50 nanosecondes. Un front par défaut de 0,5 nanoseconde est affecté aux transitions sur les entrées.

Les temps des différentes phases de simulation sont donnés dans le Tableau 8.5. Nous remarquons que la phase de désassemblage a pris nettement plus de temps que la phase de simulation.

8.3.2.6. Taux de répartition des collisions par type

Le Tableau 8.4 donne la répartition des collisions comptabilisées par type de collisions sur une multitude de circuits différents. Le résultat de la colonne "Total" confirme l'importance du phénomène et la nécessité de le prendre en considération dans les simulateurs temporels.

Tableau 8.4 : Répartition par type de collisions

	Type de collisions					
Circuit	**Même entrée**	**Branches parallèles**	**Même branche**	**"Glitch"**	**Conflit**	**Total**
Add. 32	9,0 %	10,2 %	7,6 %	1,3 %	1,4 %	**29,5 %**
Multip. 6x6	2,9 %	6,5 %	4,3 %	6,6 %	3,0 %	**23,3 %**
BShift 32	1,9 %	0,3 %	1,8 %	9,9 %	2,6 %	**16,5 %**
AMD2901	0,9 %	1,3 %	0,9 %	3,7 %	0,8 %	**7,6 %**
DLX	1,0 %	2,4 %	1,4 %	3,8 %	0,6%	**9,2 %**
Routeur	0,1 %	0,9 %	0,1 %	0,5 %	0,1 %	**1,7 %**

8.3.3. Les performances en temps CPU

8.3.3.1. Les additionneurs

Des simulations ont été effectuées sur une série d'additionneurs **n** bits, obtenus à partir d'un générateur, afin de montrer le gain en performance du simulateur SWISSE par rapport à HSPICE en termes de temps CPU. Chaque circuit est simulé pendant 1 microseconde et le pas utilisé dans HSPICE est de 0,1 nanoseconde.

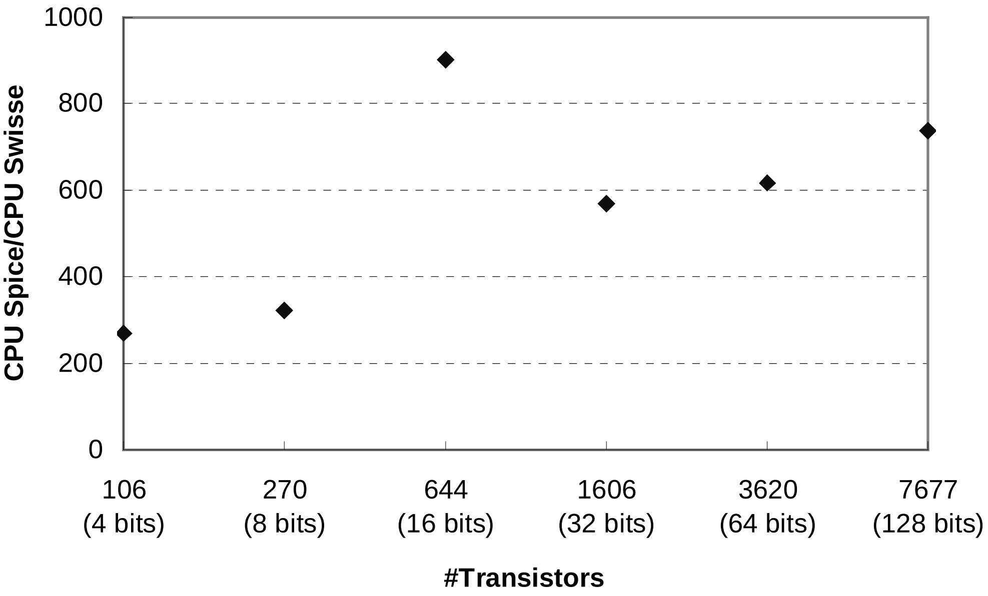


Figure 8.16 : Performance de SWISSE par rapport à HSPICE sur une série d'additionneurs

La modélisation de l'effet de retard nous permet de ramener l'erreur à 5% par rapport à HSPICE (Tableau 8.3).

Tableau 8.3 : Erreurs sur les temps de propagation suite à une collision sur une même branche

Erreur relative sans collision sur une même branche	**4,4 %**
Erreur relative si la collision est ignorée	**27 %**
Erreur relative après traitement de la collision	**5 %**

8.3.2.5. Traitement de l'ensemble des collisions sur un circuit entier

Nous avons comparé les résultats de HSPICE et de SWISSE sur un circuit présentant un fort taux de collisions de tout type (> 30% de l'ensemble des événements). C'est un additionneur 4 bits utilisant des portes XOR à base de transistors de passage.

La Figure 8.15 montre les 4 signaux de sortie de l'additionneur ainsi que le signal "CarryOut". L'erreur la plus significative est observée sur le front descendant du signal "CarryOut" et s'élève à 13,5%. Les autres signaux de sortie gardent une erreur inférieure à 8%.

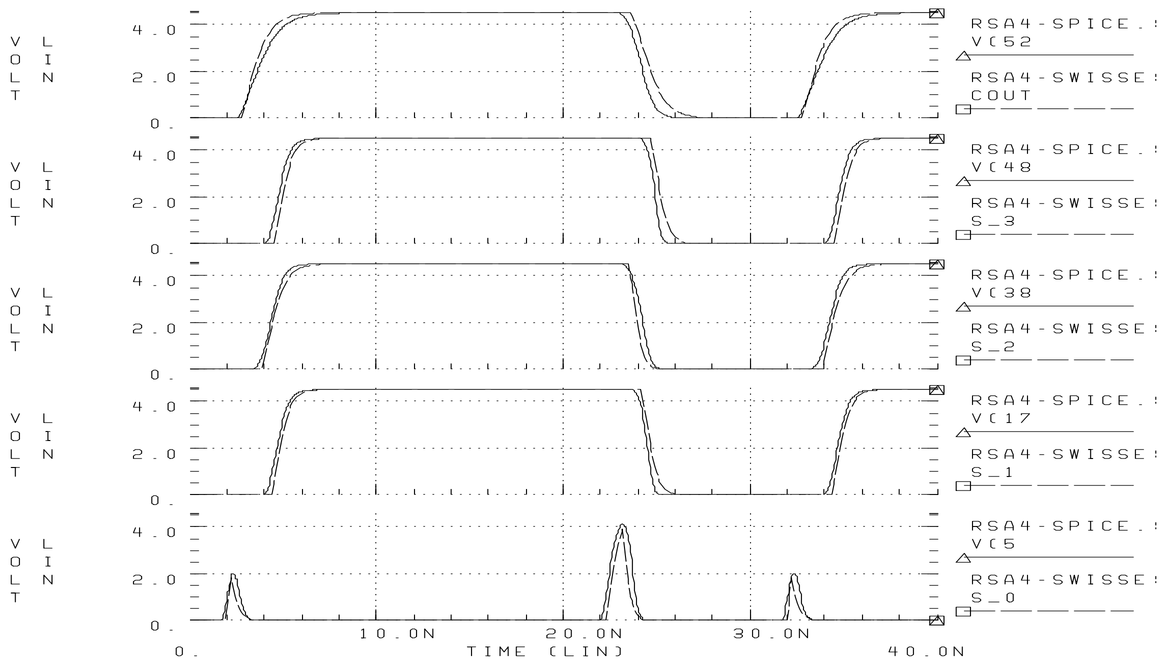


Figure 8.15 : Simulation d'un additionneur 4 bits avec un taux de collisions élevé

Le Tableau 8.2 montre qu'une erreur de plus de 85% est possible si l'effet de la collision est négligé, alors qu'elle est ramenée entre 4% et 6% par l'utilisation du modèle mis en œuvre dans SWISSE.

Tableau 8.2 : Erreurs sur les temps de propagation suite à une accélération

Erreur relative sans collision de type parallèle	**3,6 %**
Erreur relative si la collision est ignorée	**85,7 %**
Erreur relative après traitement de la collision	**5,4 %**

8.3.2.4. Collision entre deux événements sur une même branche

La Figure 8.14 montre le résultat de la modélisation d'une collision entre deux événements et qui rend passante la branche V_{SS} d'une porte Nand à deux entrées. Nous constatons que le traitement, basé sur une détermination du front dominant, rend compte de l'effet de retard quand les transitions sont décalées entre elles et possèdent des fronts très différents.

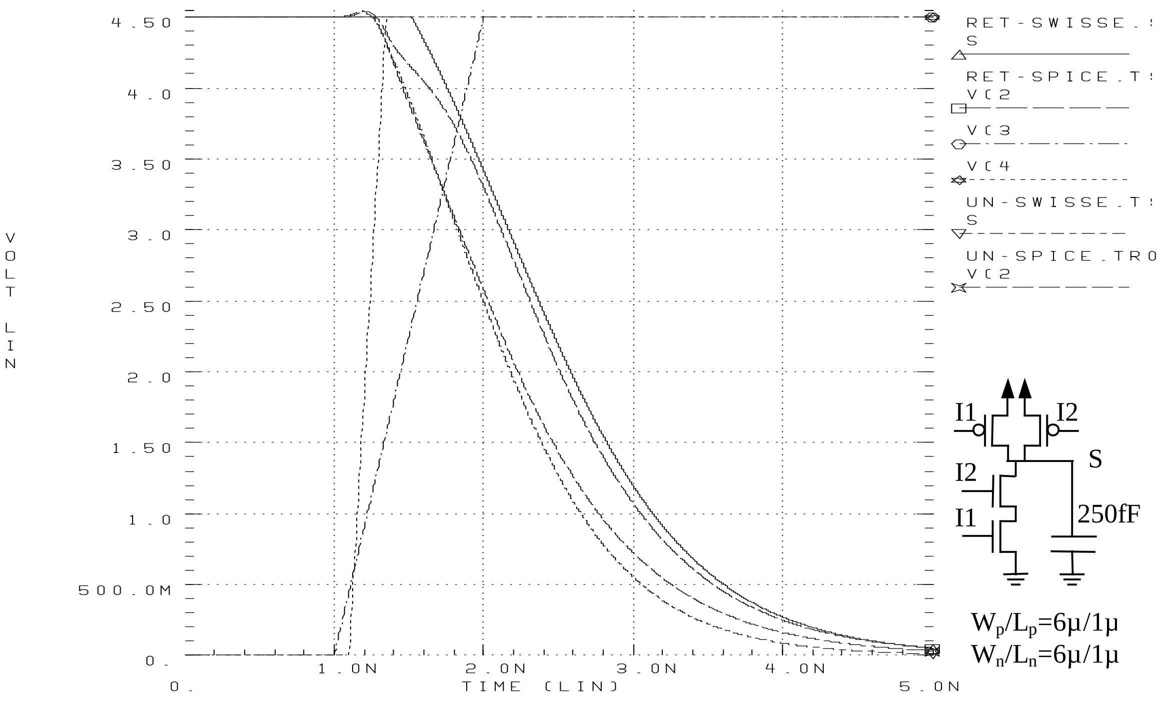


Figure 8.14 : Collision entre des événements sur une même branche

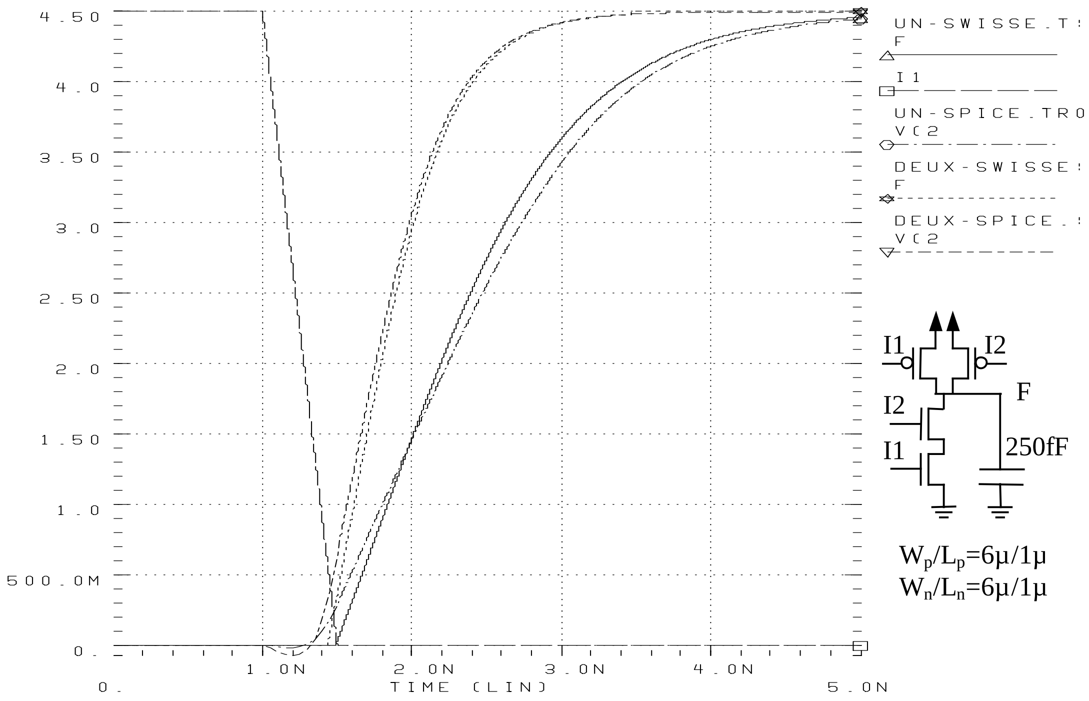


Figure 8.12 : Traitement d'une collision sur des branches parallèles

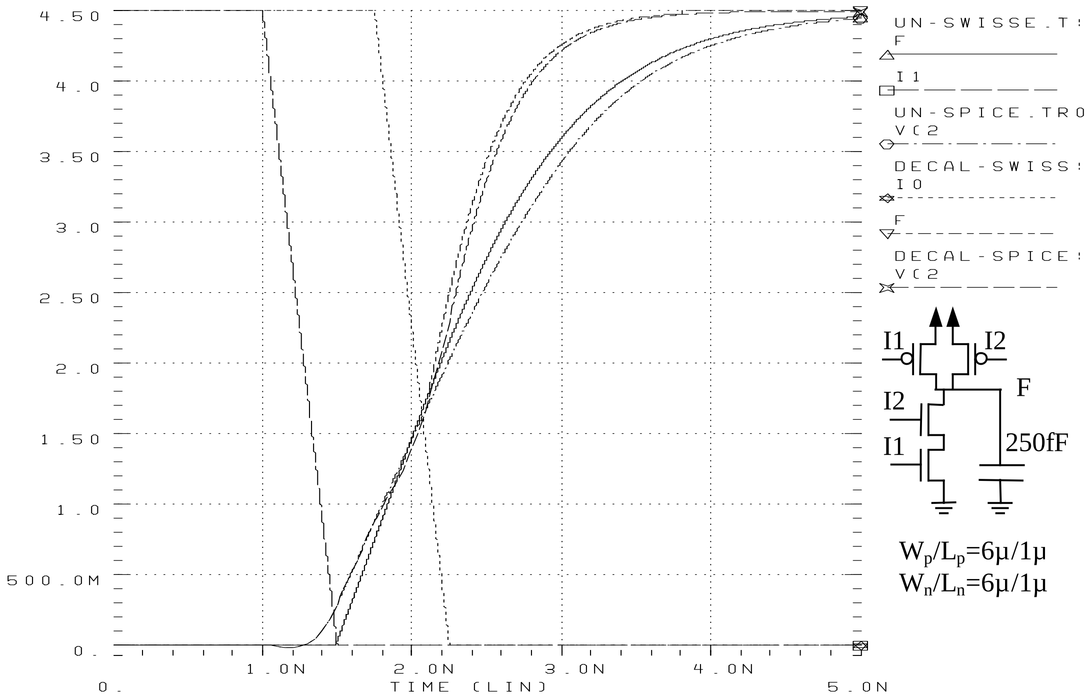


Figure 8.13 : Traitement d'une collision sur des branches parallèles
avec décalage entre les événements

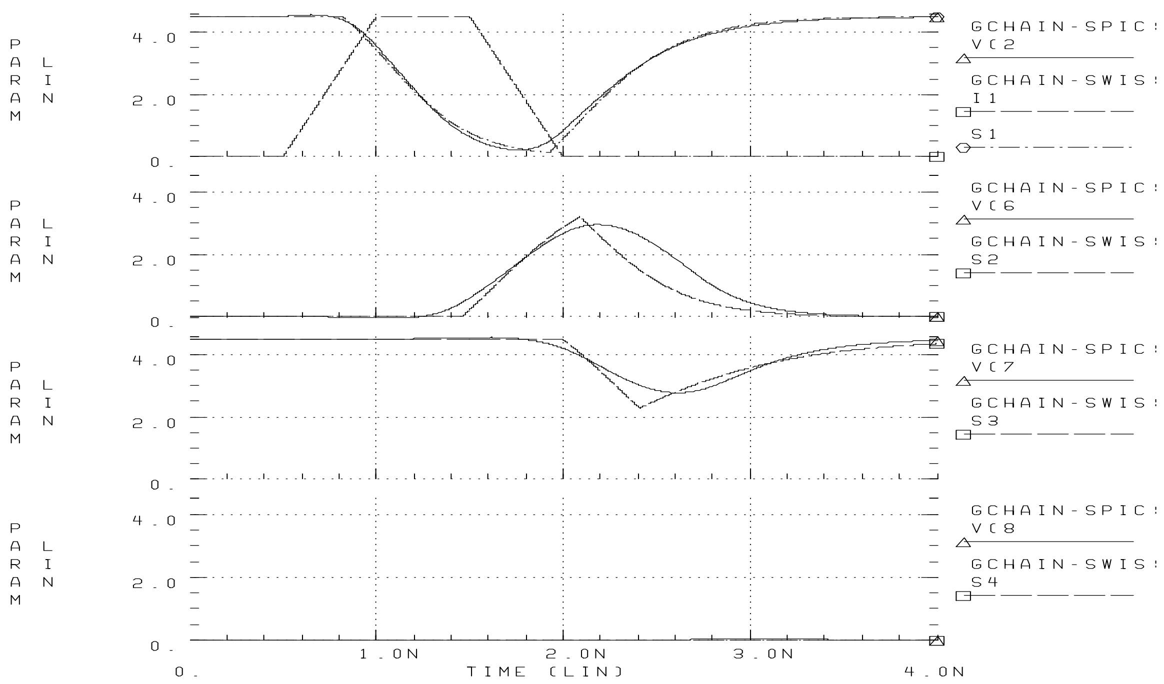


Figure 8.11 : Propagation d'un glitch

La Figure 8.11 montre la propagation du "glitch" à travers plusieurs étages. Nous constatons que le phénomène disparaît bien en même temps pour les deux simulations SWISSE et HSPICE à la sortie de la quatrième porte.

8.3.2.3. Collision entre deux événements générant une accélération

Dans ce cas, les deux branches V_{DD} d'un Nand à deux entrées sont rendues passantes, simultanément dans un premier temps, puis avec un décalage temporel entre elles dans un deuxième temps.

La Figure 8.12 traduit bien l'importance de ce type de collisions. Les deux courbes avec un front lent sont le résultat d'une simulation sans collision par HSPICE (UN-SPICE) et par SWISSE (UN-SWISSE) (une seule branche devient passante), et les deux autres courbes (DEUX-SPICE et DEUX-SWISSE) sont obtenues après simulation de l'ouverture simultanée des deux branches de la porte.

Nous constatons également sur cette figure que le modèle utilisé par SWISSE rend compte avec une très bonne précision de l'effet d'accélération observé dans un tel cas.

Dans la Figure 8.13, l'effet observé est identique, excepté qu'il intervient après que la sortie n'ait commencé sa commutation suite au premier événement. Le deuxième événement vient corriger la transition en cours suivant la méthode présentée au Chapitre 6.

Le tableau suivant montre que l'erreur sur le temps de propagation suite au traitement du changement de front est ramenée à celle du cas d'un événement simple, alors qu'elle aurait atteint 11,5% si le simulateur ne traitait pas ce type de collisions.

Tableau 8.1 : Erreurs de temps de propagation suite au changement de front

Erreur relative sans changement de front	**4,2 %**
Erreur relative si le changement de front est ignoré	**11,5 %**
Erreur relative après traitement du changement de front	**4,9 %**

8.3.2.2. Collision entre deux événements générant une transition incomplète

Dans cette expérimentation, nous avons provoqué une transition incomplète ("glitch") en effectuant deux commutations simples très proches l'une de l'autre à l'entrée d'un inverseur.

Le résultat obtenu par SWISSE montre une bonne précision par rapport à HSPICE (Figure 8.10), avec cependant une surévaluation de l'amplitude du "glitch". Ceci est dû à la méthode de simulation événementielle sur changement de front (PÉCF) qui utilise dans ce cas un seul événement. Quand les fronts des deux transitions représentant le "glitch" sont rapides, l'erreur sur l'amplitude peut atteindre 100%.

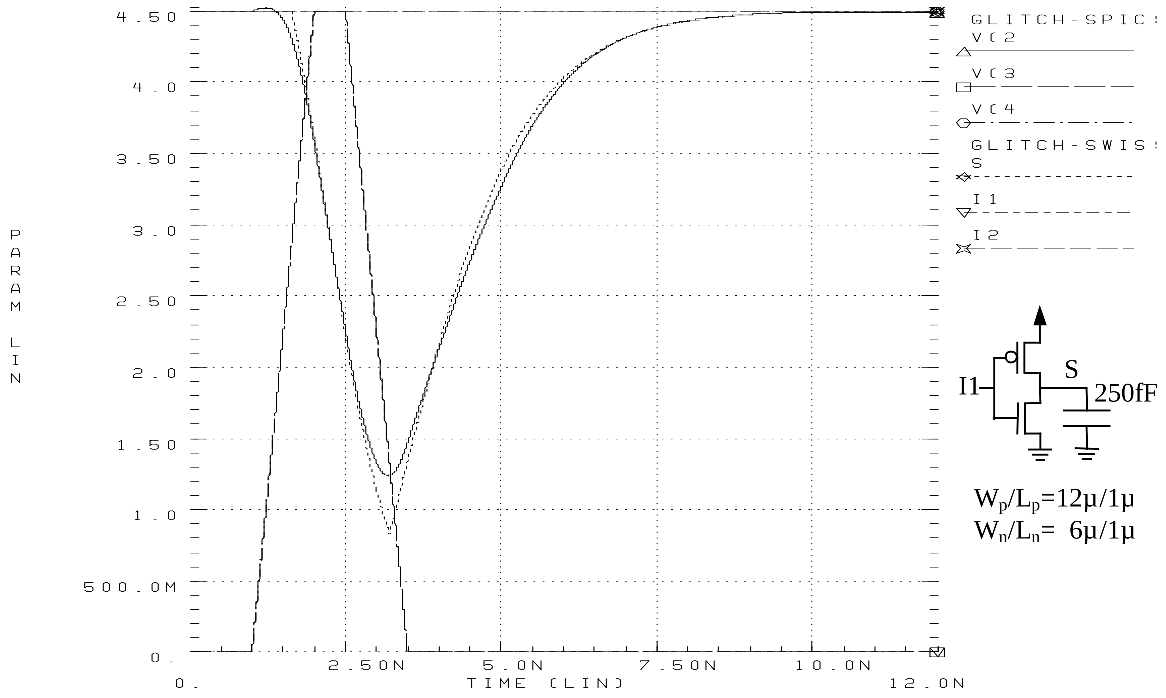


Figure 8.10 : Traitement d'un "glitch"

HSPICE (CHF-SPICE). Dans le deuxième cas, la sortie modifie sa trajectoire comme prévu en cours de commutation.

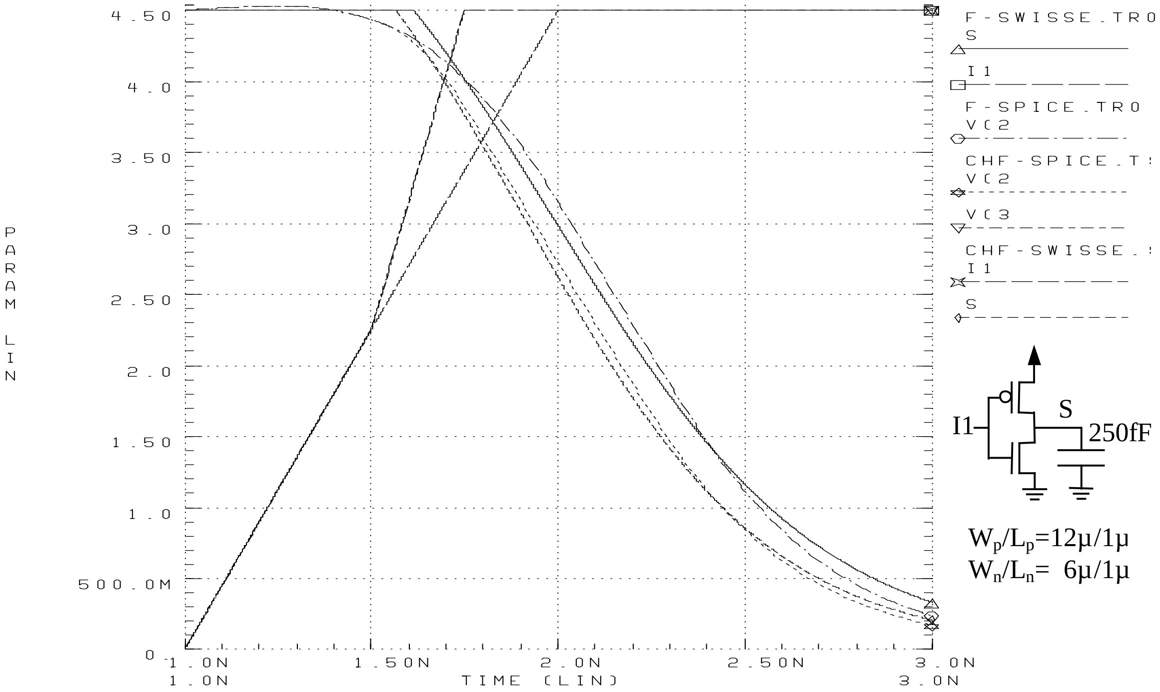


Figure 8.8 : Changement de front à l'entrée d'un inverseur

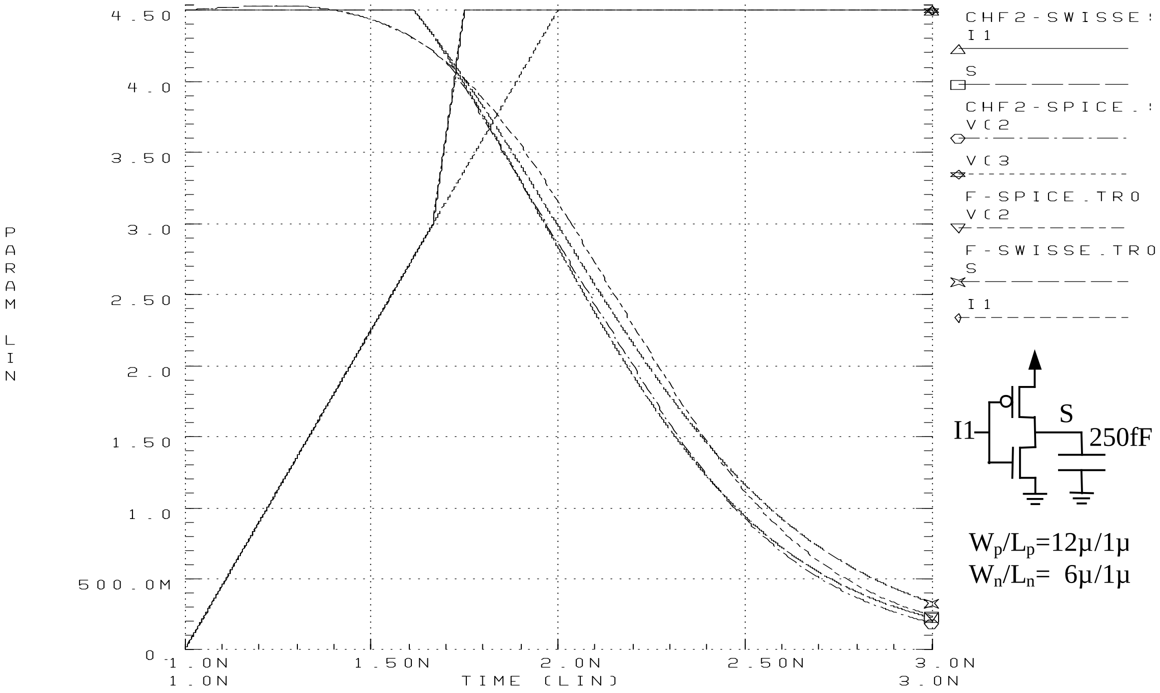


Figure 8.9 : Changement de front tardif à l'entrée d'un inverseur

La Figure 8.7 montre les résultats de simulation, tous les 10 inverseurs, d'une chaîne de 50 inverseurs. Dans ce cas, l'erreur est cumulée à travers la chaîne. Toutefois, au bout du cinquantième inverseur, les erreurs relatives sur les temps de propagation et de commutation restent inférieures à 1%. Nous pouvons ainsi conclure que la précision est maintenue à travers les étages des portes logiques, et ce toujours dans le cas des transitions simples.

8.3.2. Les collisions

Pour chaque type de collisions, nous comparons les courbes produites par SWISSE à celles produites par HSPICE. Ensuite, nous donnons l'erreur générée dans le cas où l'effet de la collision est ignoré par le simulateur et celle obtenue suite au traitement adéquat.

8.3.2.1. Collision entre deux événements sur une même entrée

Le circuit utilisé est toujours un inverseur. Nous avons expérimenté deux cas : le cas où le changement de front intervient avant que la sortie ne commence sa commutation provoquée par le premier événement (Figure 8.8) et le cas où ce changement intervient après (Figure 8.9). Les courbes F-SWISSE et F-SPICE sur les deux figures correspondent respectivement aux résultats d'une simulation par SWISSE et par HSPICE sans changement de front. Pour les deux configurations, nous observons que la courbe SWISSE (CHF-SWISSE) rejoint celle de

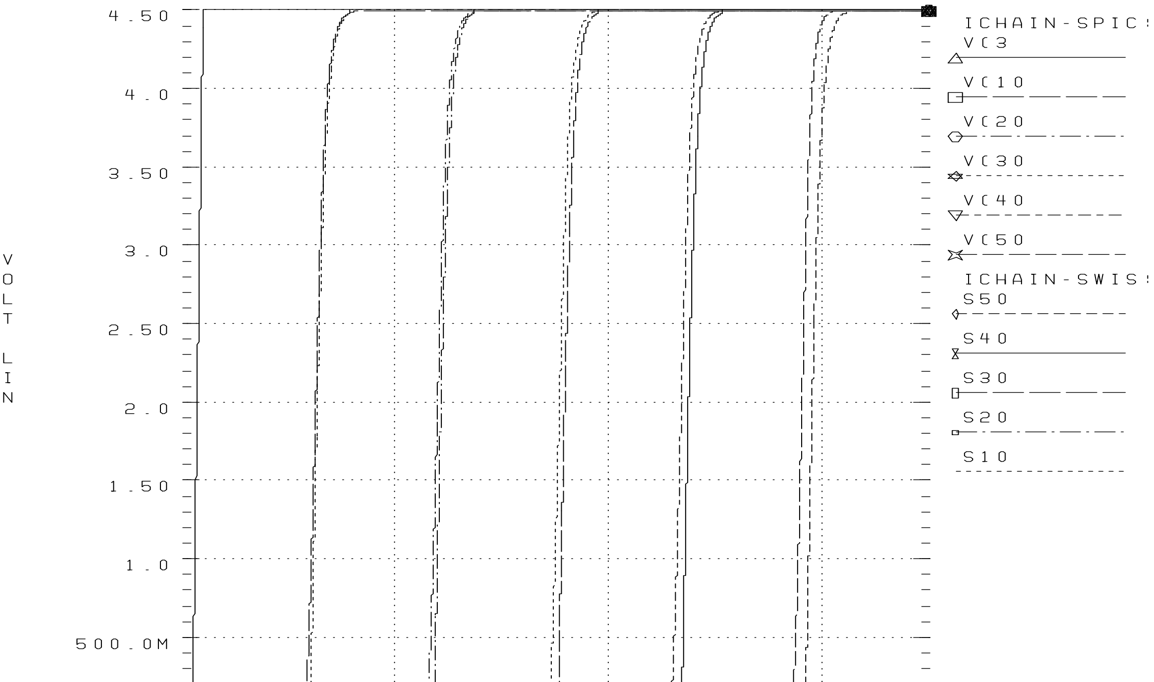


Figure 8.7 : Simulation d'une chaîne de 50 inverseurs

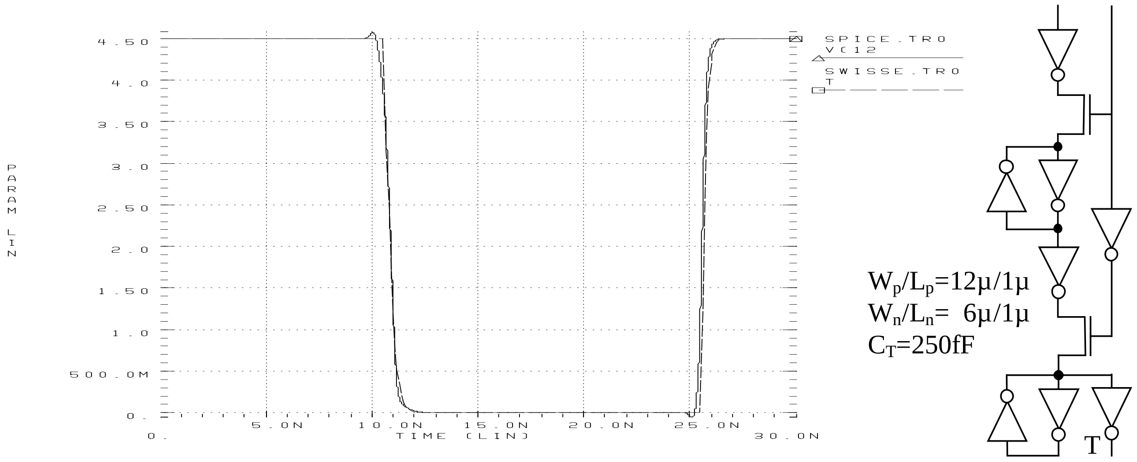


Figure 8.4 : Simulation d'une bascule maître–esclave

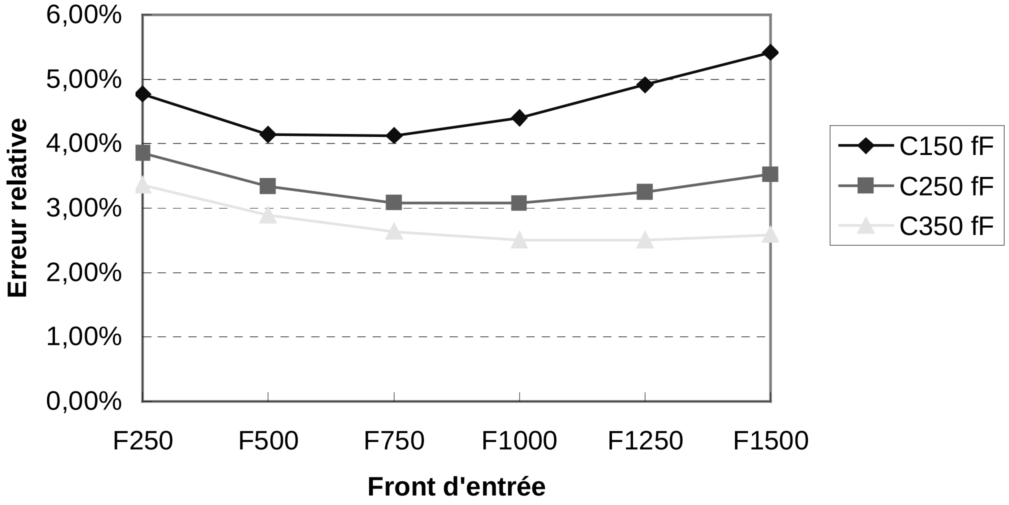


Figure 8.5 : Erreur sur le temps de propagation d'un inverseur en fonction du front en entrée (W_p/L_p=12μ/1μ; W_n/L_n= 6μ/1μ)

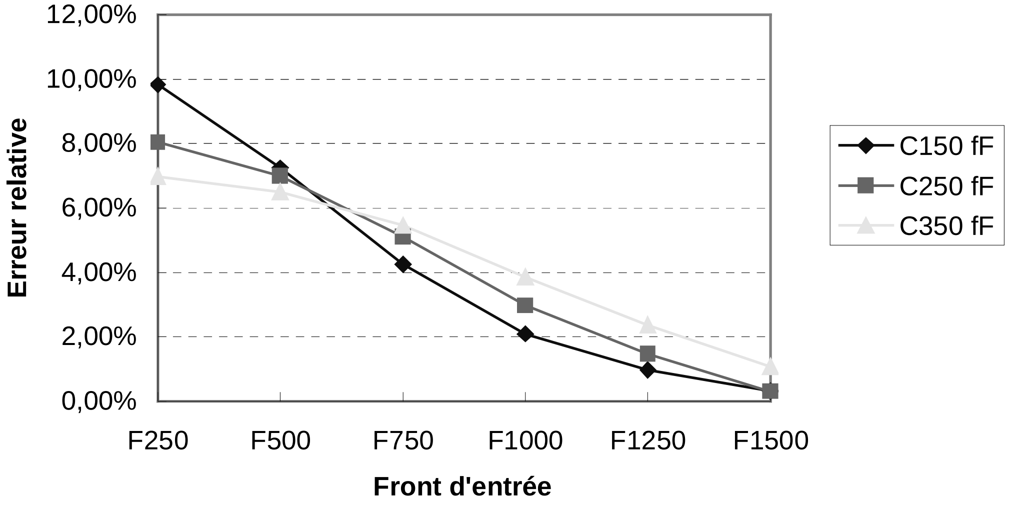


Figure 8.6 : Erreur sur le front en sortie d'un inverseur en fonction du front en entrée (W_p/L_p=12μ/1μ; W_n/L_n= 6μ/1μ)

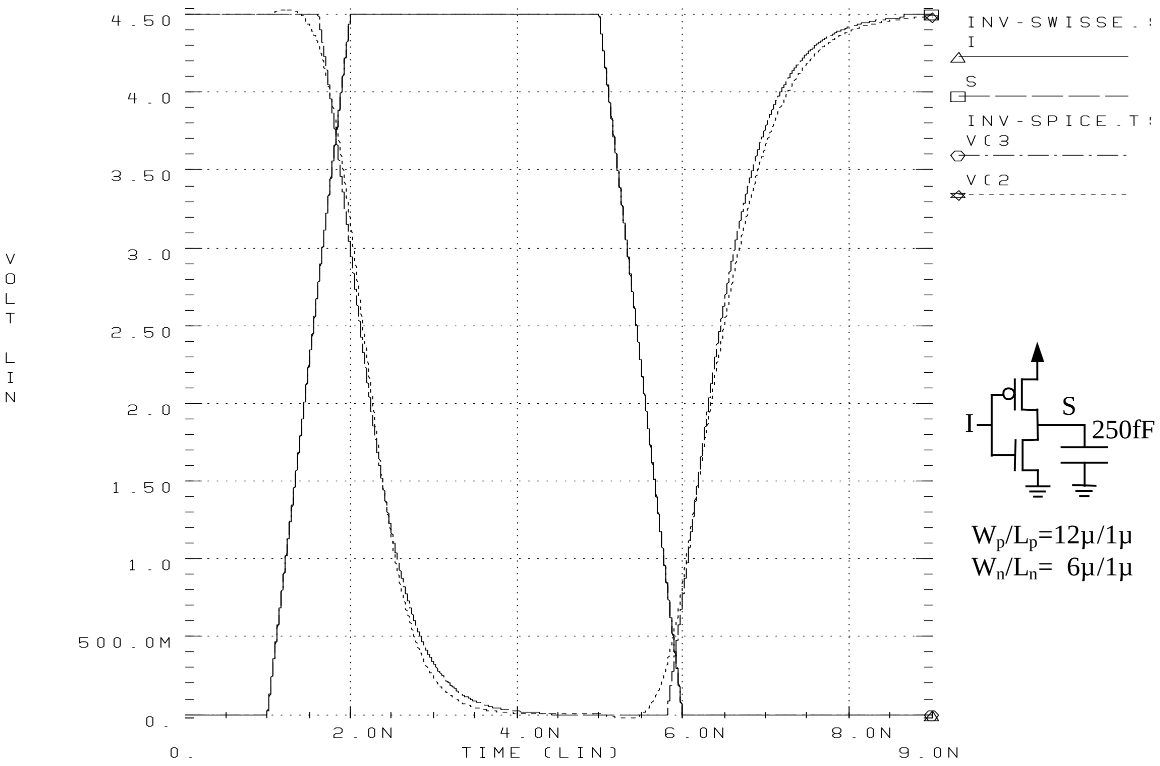


Figure 8.2 : Simulation d’un inverseur

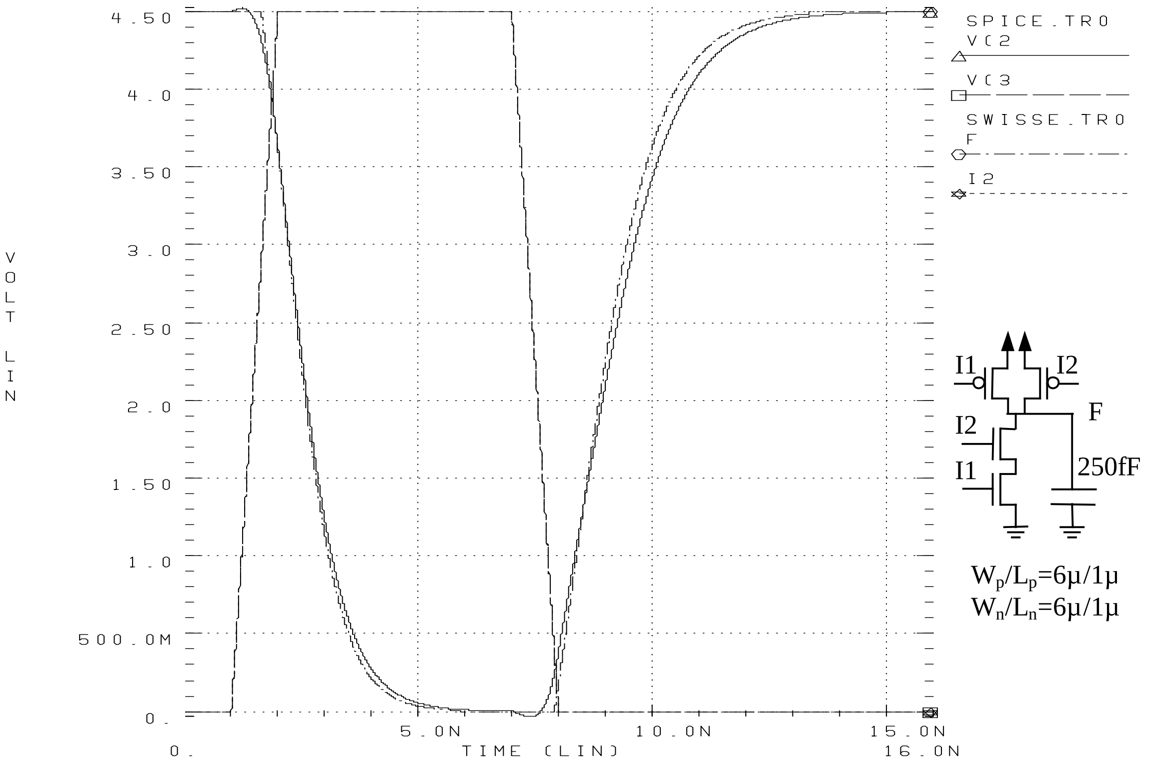


Figure 8.3 : Simulation d'une porte Nand à deux entrées

commutation; le temps de commutation étant la valeur absolue de la différence entre les instants de passage de la sortie observée par les tensions 0,2 V_{DD} et 0,8 V_{DD}.

La rapidité de la simulation est évaluée en comparant les temps CPU de la simulation HSPICE (quand la complexité du circuit le permet) à celle d'une simulation SWISSE, en variant le nombre de transistors des circuits simulés.

Nous analysons ensuite, pour deux circuits complexes, le temps CPU des tâches significatives exécutées en appliquant l'ensemble des patterns utilisés lors de la validation fonctionnelle de ces circuits, à l'aide d'un simulateur logique :

- Le temps de chargement du circuit
- Le temps de désassemblage en cônes
- Le temps d'initialisation du circuit
- Le temps de chargement des stimuli
- Le temps de simulation du circuit

8.3. Expérimentations

8.3.1. Les transitions simples

Nous présentons ici les résultats de SWISSE comparés à ceux de HSPICE sur un inverseur (Figure 8.2), sur une porte Nand à deux entrées (Figure 8.3) et sur une bascule maître–esclave à transistors de passage (Figure 8.4).

Ces résultats montrent que le modèle de signal et le modèle de courant sont réalistes dans le cas d'une transition simple intervenant à l'entrée d'une porte CMOS à une ou plusieurs entrées, ou encore d'une porte mettant en jeu des transistors de passage.

La Figure 8.5 et la Figure 8.6 montrent l'erreur relative observée, par rapport à HSPICE, sur les temps de propagation et de commutation en fonction du front d'entrée, et ce pour différentes valeurs de la capacité en sortie. Ces erreurs sont acceptables par rapport aux objectifs que nous nous sommes fixés au début de cette thèse. Le but maintenant est de vérifier que cette marge d'erreur se maintient dans le cas des collisions et lors de la simulation de circuits complexes.

8.1. Introduction

Nous présentons dans ce chapitre une analyse des performances du simulateur SWISSE. L'objectif est de valider la pertinence des compromis qui fondent la méthode de simulation PÉCF décrite dans cette thèse.

Les différentes simulations ont été réalisées sur une machine HyperSparc à 180 Mégahertz (200 SPECint92) et dotée d'une mémoire vive de 420 Mégaoctets. Nous avons utilisé une description des circuits au format SPICE pour permettre des comparaisons avec le simulateur HSPICE de Meta–Software. Tous les circuits sont simulés en technologie 1 micron avec un modèle MOSFET de niveau 2 pour HSPICE [HSP92] et le fichier technologique correspondant pour le simulateur SWISSE.

8.2. La méthode de validation

L'évaluation des performances de SWISSE passe par une évaluation de sa rapidité d'exécution et de la précision de ses estimations temporelles. La méthode s'articule autour de trois axes : une validation de la précision du modèle de transition simple, une validation de la précision des différents modèles de collisions et une validation de la capacité de la méthode à traiter des circuits complexes en un temps acceptable.

La précision de la simulation est estimée en comparant la forme d'onde obtenue par une simulation HSPICE à celle obtenue par une simulation SWISSE. Deux valeurs significatives sont calculées (Figure 8.1). La première représente l'erreur relative en temps de propagation; le temps de propagation étant la différence entre les instants de passage de l'entrée et de la sortie du circuit par la tension $V_{DD}/2$. La deuxième valeur est l'erreur relative en temps de

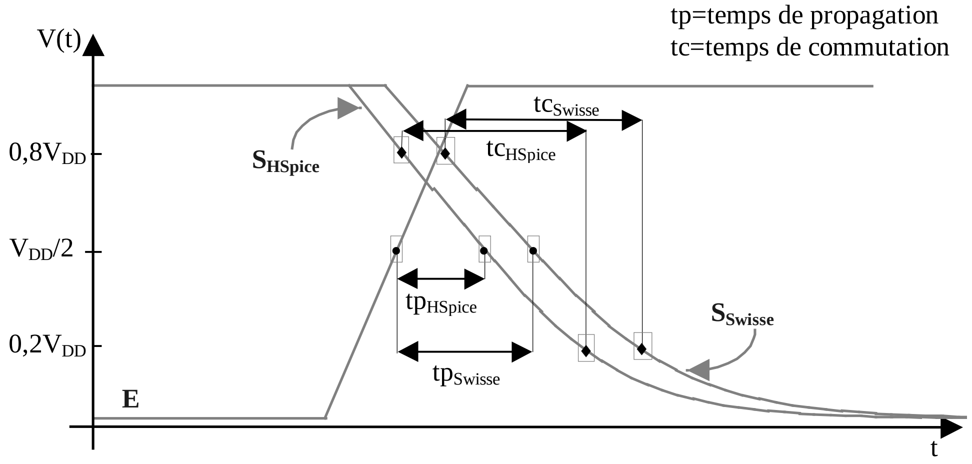
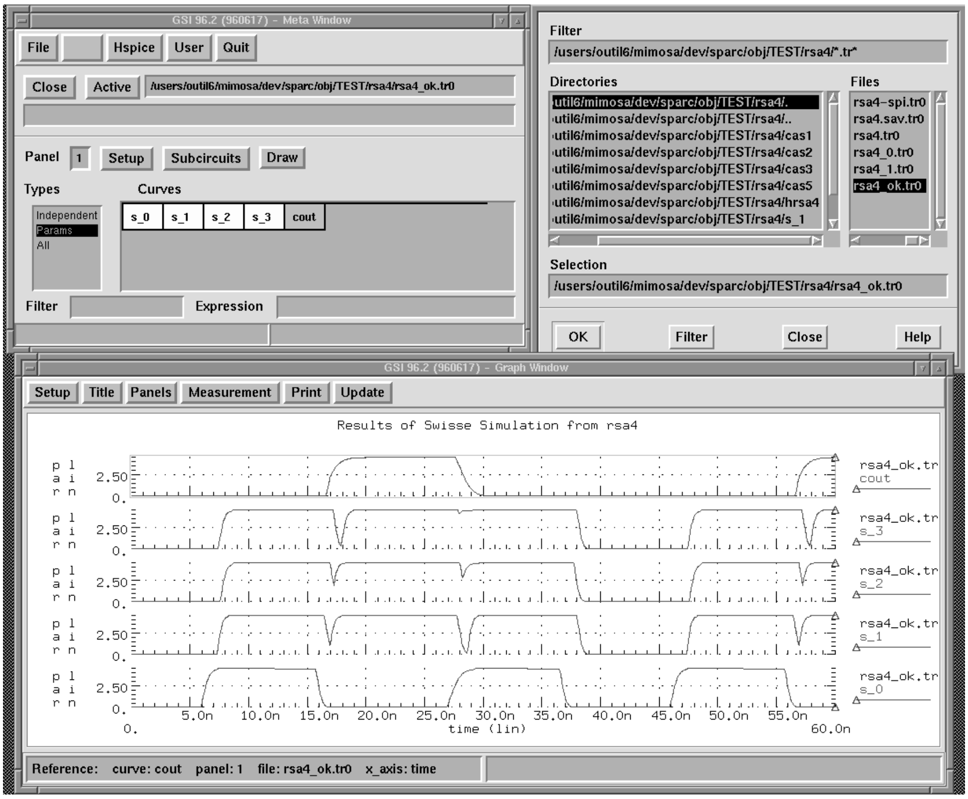


Figure 8.1 : évaluation de la précision des modèles

Chapitre 8

RÉSULTATS ET PERFORMANCES



Active: /users/outil6/mimosa/dev/sparc/obj/TEST/rsa4/rsa4_ok.tr0

Filter: /users/outil6/mimosa/dev/sparc/obj/TEST/rsa4/*.tr*

Directories:
- utíl6/mimosa/dev/sparc/obj/TEST/rsa4/.
- util6/mimosa/dev/sparc/obj/TEST/rsa4/..
- util6/mimosa/dev/sparc/obj/TEST/rsa4/cas1
- util6/mimosa/dev/sparc/obj/TEST/rsa4/cas2
- util6/mimosa/dev/sparc/obj/TEST/rsa4/cas3
- util6/mimosa/dev/sparc/obj/TEST/rsa4/cas5
- util6/mimosa/dev/sparc/obj/TEST/rsa4/hrsa4
- util6/mimosa/dev/sparc/obj/TEST/rsa4/s_1

Files:
- rsa4-spi.tr0
- rsa4.sav.tr0
- rsa4.tr0
- rsa4_0.tr0
- rsa4_1.tr0
- rsa4_ok.tr0

Selection: /users/outil6/mimosa/dev/sparc/obj/TEST/rsa4/rsa4_ok.tr0

Reference: curve: cout panel: 1 file: rsa4_ok.tr0 x_axis: time

Figure 7.6 : Sortie de SWISSE au format GSI de Meta–Software

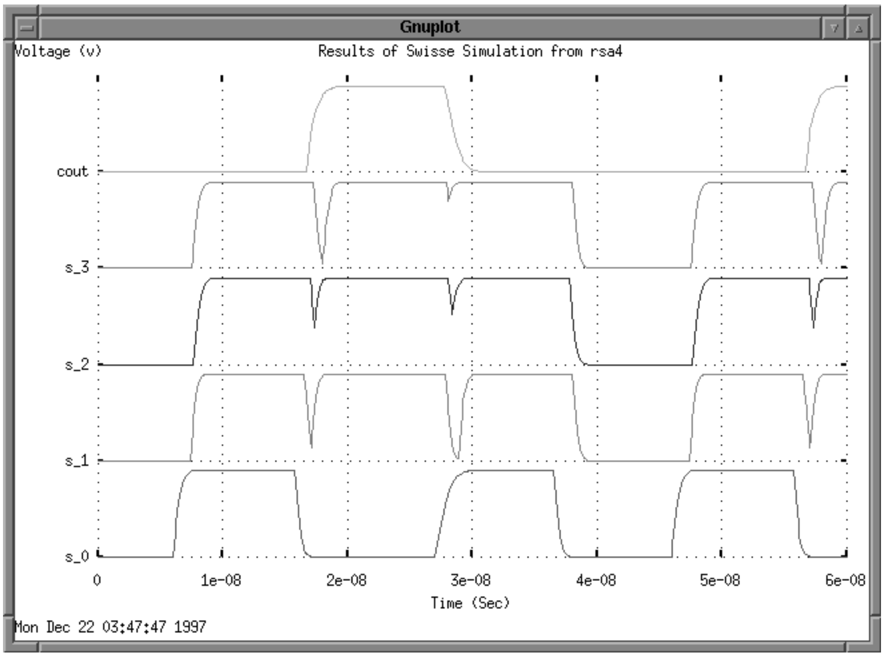


Figure 7.7 : Sortie de SWISSE au format GNUPLOT

7.6). Ainsi, nous tirons profit de la puissance d'un outil industriel et surtout, nous pouvons aisément effectuer des comparaisons et des analyses entre les résultats d'une simulation SWISSE et ceux obtenus suite à une simulation HSPICE.

7.6.5. Le générateur de résultats au format GNUPLOT

GNUPLOT est un utilitaire interactif permettant de tracer des fonctions sur différentes plates-formes telles que UNIX et MSDOS. L'avantage de cet outil est d'être disponible gratuitement sur le réseau Internet. Le simulateur SWISSE est également capable de traduire ses résultats sous format GNUPLOT (Figure 7.7).

7.7. Paramétrage technologique de SWISSE

Les paramètres technologiques de SWISSE traduisent les caractéristiques courant–tension des transistors selon le modèle MCC. Un fichier technologique qui contient l'ensemble de ces paramètres pour une technologie cible est lu en entrée par le simulateur. Nous utilisons en réalité le même fichier que celui utilisé par l'analyseur temporel statique TAS [Haj91]. La détermination de ces paramètres s'appuie sur une méthode proposée par A. HAJJAR dans sa thèse [Haj92]. Elle est basée sur la simulation des transistors MOS pour des valeurs spécifiques de V_{ds} et de V_{gs} (voir aussi la section 4.3.1). Les simulations sont effectuées à l'aide d'un simulateur de référence (par exemple HSPICE de Meta–Software [HSP92]) et utilisant un modèle de référence. Depuis, la méthode a sensiblement évolué. P. KIANI [Kia93] et M. DUHALDE [Duh94] ont étendu cette méthode en présentant une approche plus générale de calcul des paramètres. Enfin, K. DIOURY [Dio97] a spécifié un ensemble de circuits permettant d'ajuster avec plus de précision la valeur des paramètres sensibles. Un exemple de fichier technologique TAS et SWISSE est donné en Annexe 4 pour une technologie fictive de 1micron.

7.8. Conclusion

Nous avons montré dans ce chapitre la mise en œuvre d'un outil logiciel effectuant la simulation suivant la méthode PÉCF décrite aux chapitres 5 et 6.

L'interface utilisateur que nous avons développée permet l'utilisation de l'outil avec la chaîne ALLIANCE mais aussi avec d'autres chaînes de CAO. De même, les formats de sortie adoptés rendent possible l'exploitation des résultats par une variété d'outils, y compris des outils industriels, permettant ainsi d'effectuer des comparaisons avec des simulateurs tels que SPICE.

7.6. L’interface utilisateur de SWISSE

7.6.1. Le compilateur de stimuli au format SPICE

SPICE est un des deux formats accepté par SWISSE. La popularité d'un tel format ainsi que l'utilisation de SPICE comme simulateur de référence pour mesurer le degré de précision de notre méthode ont motivé le développement de ce compilateur.

Si la description du circuit n'est pas au format SPICE, il est tout de même possible de donner à SWISSE les stimuli au format SPICE dans un fichier séparé.

L'Annexe 1 donne une description du sous-ensemble des commandes SPICE accepté par le simulateur SWISSE.

L'Annexe 2 explique comment sont traduits les stimuli en forme de rampe donnés par SPICE en une tangente hyperbolique qui est le modèle de signal interne à SWISSE.

7.6.2. Le compilateur de stimuli au format PAT

La chaîne ALLIANCE comporte un simulateur logique sans aspect temporel ASIMUT [Baz93] [Vuo97], qui utilise en entrée un fichier de stimuli au format PAT.

Un compilateur de ce format vers la structure de stimuli interne au simulateur SWISSE a ainsi été développé. L'aspect temporel étant manquant dans cette description (puisque dans ASIMUT, les transitions sont instantanées), une valeur du front et un délai séparant les stimuli sont fournis en paramètre par l'utilisateur. Par ce moyen, nous avons pu récupérer les stimuli fonctionnels écrits pour les circuits conçus au laboratoire, et nous avons également pu comparer la validité de l'aspect logique du simulateur sur des circuits très complexes.

7.6.3. Le générateur de résultats sous la forme de stimuli SPICE

Outre les formats classiques utilisés par différents outils d'affichage, SWISSE est capable de générer ses résultats sous la forme de stimuli SPICE. Cette fonctionnalité permet d'injecter ces stimuli en entrée du simulateur SPICE pour la simulation d'une partie analogique dans le cas où l'interface est uniquement dans le sens numérique vers analogique.

7.6.4. Le générateur de résultats au format GSI de Meta–Software

La société Meta–Software distribue avec son simulateur HSPICE [HSP92] un outil de visualisation graphique et interactif appelé GSI, permettant d'analyser les résultats de simulation et d'effectuer différentes mesures à partir des courbes de courants ou de tensions. SWISSE est capable de générer ses résultats dans un format compréhensible par GSI (Figure

Le résultat de cette deuxième phase est couple d'événements intervenant sur la sortie (un événement indiquant un changement de front et un événement indiquant la fin de commutation) et qui sont programmés dans l'échéancier.

7.5.2.3. Détermination de la valeur finale en sortie

Comme expliqué précédemment, un événement inclut dans ses paramètres la valeur de la tension finale stable vers laquelle tend le signal à la fin de sa commutation. Cette valeur est susceptible de changer si un nouvel événement intervient avant que le signal n'atteigne sa valeur finale. Ainsi, chaque fois qu'un événement est programmé, la valeur de la tension finale doit être déterminée pour cet événement en fonction de l'ensemble des branches passantes de la porte.

Ainsi, la sortie tend vers un niveau dégradé, si et seulement si, toutes les branches passantes de la porte contiennent un transistor dégradant le niveau du signal.

C'est dans la phase de désassemblage que chaque branche est analysée au moment de sa génération pour savoir si elle est de nature à dégrader le niveau et est marquée en conséquence. Ainsi, lors de la simulation, il suffit d'examiner ces marqueurs ainsi que le type des branches passantes pour déterminer la valeur finale de la sortie.

7.5.3. Algorithme général de simulation

Les étapes principales de l'algorithme général de simulation sont les suivantes :

- Initialisation du simulateur
- Enregistrement dans l'échéancier des événements sur les entrées
- Stabilisation du circuit dans un état initial cohérent par une simulation logique zéro–délai
- Simulation du circuit tant que l'échéancier est non vide
  - Avancer le temps jusqu'à la date du premier événement programmé
  - Extraire les événements
  - Mise à jour de l'état des signaux
  - Si nouveau front non nul, marquer les branches et les portes comme étant à évaluer
  - évaluer les portes puis éventuellement appliquer les fonctions de résolution
  - Programmer les nouveaux événements dans l'échéancier
- Génération des résultats

7.5.1.6. Les connecteurs externes

Ils constituent l'interface externe du circuit. Chaque connecteur externe de type "OUT" a un pointeur sur le cône auquel il est relié. Lorsque le connecteur est de type "IN", il a un pointeur sur la liste des cônes auxquels il appartient.

7.5.2. Algorithme d'évaluation d'une porte

Le modèle événementiel que nous avons choisi inclut la possibilité d'avoir des événements significatifs représentant une modification de la configuration électrique d’une porte, sans pour autant qu'il y ait de changement de la valeur logique de sa sortie.

Le modèle Booléen étant insuffisant à rendre compte de ce type de configurations, le modèle d’évaluation par analyse des branches va nous permettre de détecter si un calcul temporel est nécessaire quand un événement se présente aux entrées d’une porte.

Ainsi, quand un événement intervient sur un signal **S**, toutes les portes ayant ce signal comme entrée sont planifiées pour subir un traitement, mais aussi, toutes les branches dont un des transistors est commandé par **S** sont marquées comme étant à évaluer.

Lors du traitement de chaque porte, ce sont uniquement les branches marquées et appartenant à cette porte qui sont analysées, les autres n'ayant subi aucun changement. L'évaluation d'une porte s'effectue en deux phases : une première phase d'analyse logique des branches, suivie, selon les résultats, d'une phase éventuelle de résolution de la porte pour déduire les caractéristiques de l'événement en sortie. Nous utilisons le terme résolution par analogie avec la résolution d'un bus à émetteurs multiples dans le cadre d'une simulation logique.

7.5.2.1. Phase 1 : Analyse logique des branches

l'analyse logique d'une branche consiste à parcourir les transistors un à un depuis le signal de sortie vers l'alimentation, afin d'évaluer son état.

7.5.2.2. Phase 2 : Fonction de résolution de la sortie

L'appel à la fonction de résolution est nécessaire, si, au moins une des branches analysées appartenant à la porte a subi un changement dans son état logique (passante ou bloquée), ou si, déjà passante, elles a subi un changement de ses caractéristiques électriques.

Ensuite, si plusieurs branches sont passantes, cela indique une collision de type branches parallèles. Autrement, nous sommes en présence d'une commutation simple dont le traitement a été décrit au chapitre précédent.

du désassemblage, la sortie d'un cône n'attaque que des grilles de transistors. C'est grâce à ce champ que l'on retrouve le cône qui conditionne l'état de ce transistor.

7.5.1.3. Le cône

Cette structure est la plus riche obtenue après la phase de désassemblage. Elle comprend principalement un nom (le nom du signal correspondant à la sortie du cône), un type constitué de la concaténation d'un ensemble d'attributs (conflictuel, latch, etc.), un pointeur sur la liste des cônes récepteurs (cônes en aval), un pointeur sur la liste des cônes émetteurs (cônes en amont) et un pointeur sur une liste de branches.

Lors de la simulation, le signal représentant la sortie du cône sera également associé aux informations suivantes :

- $t_{évt}$: Date du dernier événement sur ce signal.
- $V_{évt}$: Valeur de la tension à l'instant $t_{évt}$.
- F : Valeur de la constante de temps (zéro si le signal est stationnaire).
- V_{final} : Valeur finale prévue si le signal est en cours de commutation ($F \neq 0$).

Cette définition traduit le fait que le jeu de paramètres décrivant le signal permet de calculer la valeur réelle du signal à tout instant.

7.5.1.4. Les branches

Les branches sont constituées par listes typées (branches V_{DD}, V_{SS}, Ext, etc.). La structure d'une branche est principalement constituée d'une liste de maillons.

Lors de la simulation, la branche est caractérisée par les informations suivantes :

- *État* : État logique de la branche (passante ou bloquée).
- *ÀÉvaluer* : Indique que la branche doit être évaluée.
- *ÉvtsMultiples* : Indique que des événement sont intervenus sur plus d'un transistor appartenant à la branche.
- *ModifFront* : Indique qu'un changement de la constante de temps, ou front, est intervenu sur la branche.

7.5.1.5. Les maillons

Un maillon est le constituant élémentaire d'une branche. Dans le cas général, c'est un pointeur sur un transistor. Cependant, si l'extrémité de la branche est un connecteur externe, alors le maillon pointe sur ce dernier. En outre, le maillon comporte un champ contenant la capacité visible sur le drain du transistor.

Tableau 7.2 : Les fonctions de statistiques du gestionnaire d'événements

Code retour	Nom de la fonction	Description
unsigned long	sch_GetCurrentEventsNumber	Création d'un échéancier et renvoi d'un identificateur à utiliser lors des futurs appels de fonctions concernant cet échéancier.
unsigned long	sch_GetMaxEventsNumber	Créer puis insérer un événement dans un échéancier. Renvoie un pointeur sur une zone de données utilisateur.
Unsigned long	sch_GetTotalEventsNumber	Dupliquer un événement avec une nouvelle date de programmation.
Void	sch_GetSchedulerConfig	Renvoi des caractéristiques de l'échéancier.

Tableau 7.3 : Les fonctions de trace du gestionnaire d'événements

Code retour	Nom de la fonction	Description
int	sch_GetDebugLevel	Renvoi du niveau de précision courant du mode de trace lors de l'utilisation de l'échéancier.
void	sch_SetDebugLevel	Positionnement d'un niveau de précision du mode de trace.
Void	sch_PrintScheduler	Affichage sur la sortie standard des caractéristiques de l'échéancier.
void	sch_PrintEvent	Affichage sur la sortie standard des caractéristiques d'un événement.

7.5. Module d'évaluation

Cette partie présente les structures de données et les principaux algorithmes du module d'évaluation des portes logiques.

7.5.1. Structures de données

7.5.1.1. La figure

Une figure représente la structure de données de plus haut niveau dans le simulateur. Avant désassemblage, cette structure représente un réseau de transistors non orienté. Après désassemblage, elle contient principalement trois listes chaînées : la liste des transistors, la liste des connecteurs externes et la liste des cônes.

7.5.1.2. Le transistor

Outre ses dimensions, la structure d'un transistor contient un champ incône qui est une référence vers le cône dont la sortie commande la grille de ce transistor. Rappelons qu'à l'issue

Tableau 7.1 : Les fonctions standards d'accès au gestionnaire d'événements

Code retour	Nom de la fonction	Description
SCH_ID	sch_InitScheduler	Création d'un échéancier et renvoi d'un identificateur à utiliser lors des futurs appels de fonctions concernant cet échéancier.
SCH_EVENT_ID	sch_AddEvent	Création puis insertion d'un événement dans un échéancier. Renvoi d'un pointeur sur une zone de données utilisateur.
SCH_EVENT_ID	sch_DupEvent	Duplication d'un événement avec une nouvelle date de programmation.
Void	sch_RemoveEvent	Suppression d'un événement et libération de sa zone de données utilisateur correspondante.
Void	sch_ClearEventOnKey	Suppression de tous les événements sur un signal et libération de leurs zones de données utilisateur correspondantes.
Short	sch_Go2NextTime	Libération des zones de données utilisateur correspondant aux événements au temps courant, puis avancement du temps courant jusqu'à la date de l'événement le plus proche.
SCH_TIME	sch_GetCurrentTime	Renvoi de la valeur du temps courant.
SCH_EVENT_ID	sch_ExtractFirstEvent	Avancement du temps courant jusqu'à la date de l'événement le plus proche et extraction du premier événement à cette date.
SCH_EVENT_ID	sch_ExtractAtCurrent	Extraction du premier événement à l'instant courant et renvoi d'un pointeur sur la zone de données utilisateur.
SCH_EVENT_ID	sch_GetFirstEvent	Renvoi d'un pointeur sur la zone de données de l'événement le plus proche sans l'extraire.
SCH_EVENT_ID	sch_GetNextEvent	Renvoi d'un pointeur sur la zone de données de l'événement suivant sans l'extraire.
SCH_EVENT_ID	sch_GetFirstFromTime	Renvoi d'un pointeur sur la zone de données du premier événement à un instant donné sans l'extraire.
SCH_EVENT_ID	sch_GetNextFromTime	Renvoi d'un pointeur sur la zone de donnée de l'événement suivant à un instant donné sans l'extraire.
SCH_EVENT_ID	sch_GetFirstFromKey	Renvoi d'un pointeur sur la zone de données du premier événement sur un signal donné sans l'extraire.
SCH_EVENT_ID	sch_GetNextFromKey	Renvoi d'un pointeur sur la zone de donnée de l'événement suivant sur un signal donné sans l'extraire.

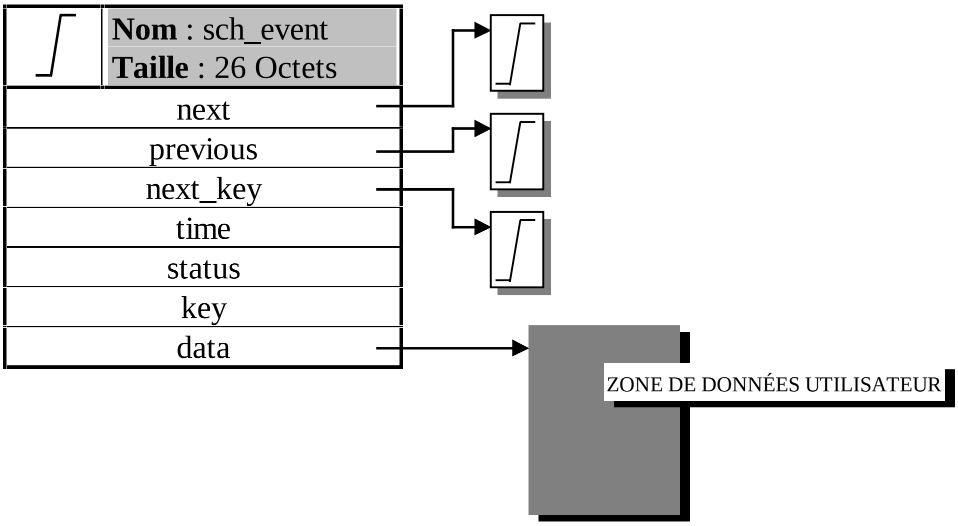


Figure 7.5 : Structure de données événement

événement, le gestionnaire d'événements s'alloue une zone permettant l'intégration de l'événement au sein de l'échéancier et une zone de données qui est renvoyée au simulateur afin d'y placer les données correspondant à l'événement lui même (front, tension finale, etc..). Cette distinction donne au gestionnaire d'événements son aspect générique et permet son utilisation par tout simulateur indépendamment de la notion d'événement qu'il manipule.

7.4.3. Les fonctions d'accès aux services du gestionnaire d'événements

Nous pouvons distinguer trois classes de fonctions d'accès aux services du gestionnaire d'événements. Il y a les fonctions standards (Tableau 7.1) permettant la création d'un échéancier, l'extraction d'événements, l'adjonction ou la suppression d'événements, ou encore la simple consultation.

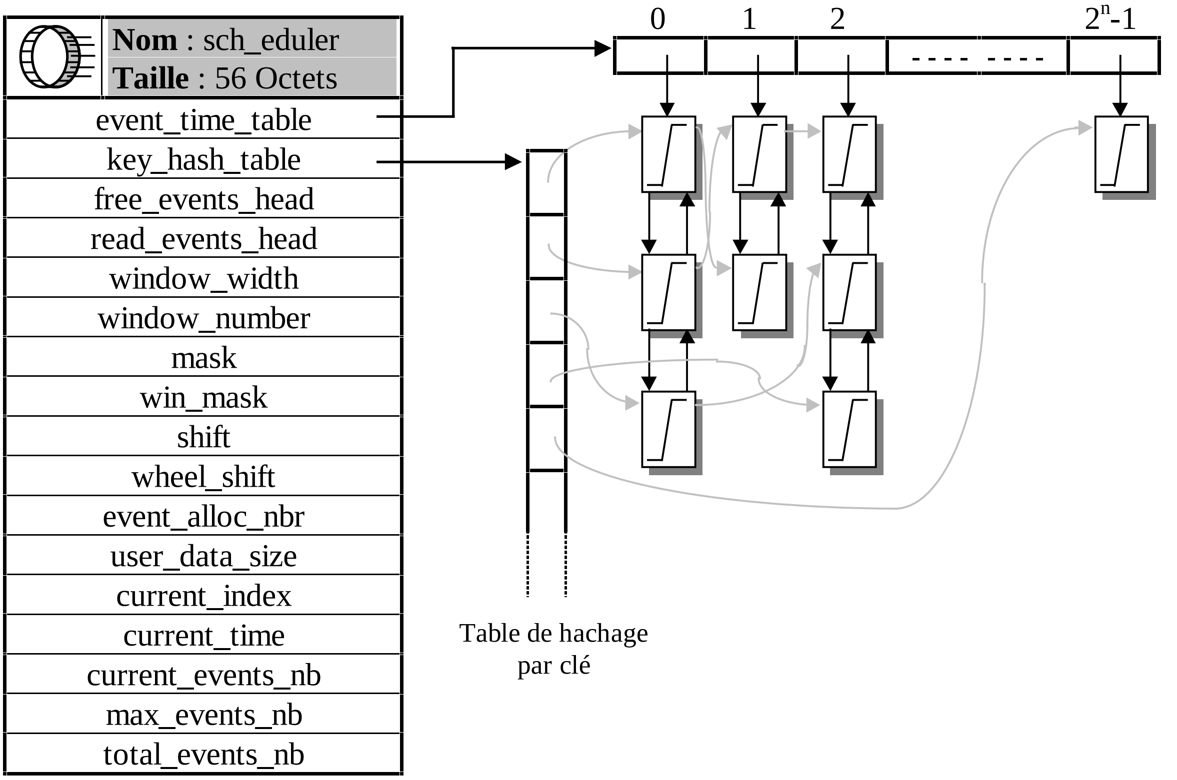
Il y a également une classe de fonctions qui permet l'obtention de statistiques concernant l'utilisation de l'échéancier (Tableau 7.2). Enfin, le gestionnaire d'événements offre un ensemble de fonctions permettant d'avoir des traces de l'exécution des fonctions et ce suivant plusieurs niveaux de précision (Tableau 7.3). Les deux dernières classes de fonctions peuvent être débrayées du code par une directive de pré–compilation afin de ne pas pénaliser les performances du simulateur.

La Figure 7.4 montre la structure de données privée représentant un échéancier. Outre le pointeur *event_time_table* qui pointe sur le tableau des fenêtres temporelles décrit précédemment, nous trouvons dans cette structure un pointeur sur une table de hachage, *key_hash_table*, permettant de retrouver les événements correspondant à une clé donnée. Dans le cas du simulateur SWISSE, cette clé est le nom du signal.

Nous trouvons également dans cette structure un pointeur, *read_events_head*, sur la liste des événements extraits de l'échéancier qui sont en cours de traitement par le simulateur. L'espace occupé par ces événements devient libre et est récupéré dans la liste pointée par *free_events_head* dès que le temps courant est mis à jour.

Les autres éléments de la structure de données représentent les différents paramètres de l'échéancier tels que les masques permettant d'obtenir l'indice de la fenêtre temporelle ou des statistiques quant à l'utilisation de l'échéancier suite à une simulation.

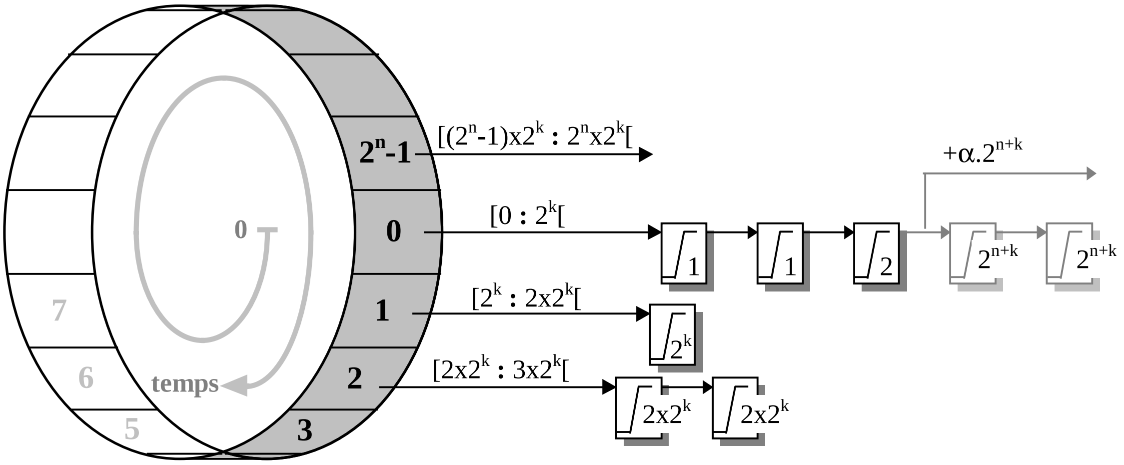
La Figure 7.5 montre la structure de données représentant un événement générique. L'aspect le plus intéressant dans cette structure tient dans la séparation faite entre les informations utilisées par l'échéancier et les informations spécifiques au simulateur utilisant l'échéancier (dans notre cas, c'est le simulateur SWISSE). En effet, lors de la création d'un



Nom : sch_eduler
Taille : 56 Octets

event_time_table
key_hash_table
free_events_head
read_events_head
window_width
window_number
mask
win_mask
shift
wheel_shift
event_alloc_nbr
user_data_size
current_index
current_time
current_events_nb
max_events_nb
total_events_nb

Figure 7.4 : La structure de données échéancier



2^n fenêtres avec un intervalle de 2^k par fenêtre

Figure 7.2 : Principe de l'échéancier

seconde, il y a peu de chances d'avoir des événements qui se suivent à une pico–seconde près. Ainsi, nous risquons d'avoir à parcourir plusieurs éléments vides du tableau avant d'atteindre le temps de simulation suivant.

D'autre part, le choix des tailles en termes de puissance de deux permet l'application de masques sur la valeur du temps, afin d'extraire l'indice de la fenêtre temporelle où l'événement doit être rangé (Figure 7.3).

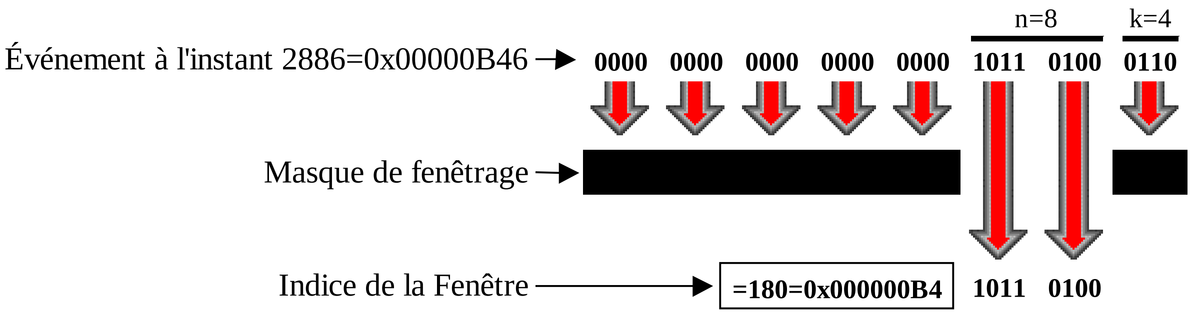


Figure 7.3 : Exemple de détermination de la fenêtre d'accueil d'un événement dans un échéancier à 2^8 fenêtres temporelles, de largeur 2^4 pico–secondes chacune

7.4.2. Les structures de données

Les différentes structures de données composant le module échéancier sont privées et accessibles uniquement à travers les fonctions de l'interface d'accès. Le point d'entrée au module est la création d'un échéancier. Cette opération renvoie à l'application appelante un identificateur SCH_ID représentant en réalité le pointeur sur la structure de données. Toute requête ultérieure concernant cet échéancier fournit l'identificateur SCH_ID en paramètre. Ce mécanisme permet pour une même application la gestion simultanée de plusieurs échéanciers.

7.4. Le gestionnaire d'événements (échéancier)

7.4.1. Présentation générale

L'objectif est d'obtenir un gestionnaire d'événements générique utilisable par différents simulateurs à pilotage événementiel, d'où une approche de développement orientée objet avec une interface à accès normalisé.

Les opérations d'insertion et de suppression d'événements dans un échéancier doivent être rapides. Ceci nécessite une bonne stratégie d'allocation mémoire afin de solliciter au minimum le système d'exploitation.

De nombreuses études ont été réalisées concernant la distribution des événements dans un échéancier lors d'une simulation [Ulr69] [Vau75] [Ulr78] [Won86]. En résumé, il apparaît que le nombre maximum d'événements dans l'échéancier à un instant donné dépasse rarement 10% la taille du circuit en nombre de transistors et qu'il varie de façon linéaire avec la complexité du circuit. D'autre part, la distribution des événements indique une concentration de ces derniers sur un petit nombre de points temporels proches de l'instant courant. Enfin, une comparaison des différentes méthodes de hachage [Fro94] permettant de mettre en œuvre un échéancier indique que la méthode la plus efficace est *le hachage avec chaînage séparé* qui supporte le mieux l'adjonction et la suppression d'éléments.

Dans cette thèse, nous nous appuyons sur ces études et sur notre propre expérience pour proposer une mise en œuvre adaptée à nos besoins.

L'idée est d'utiliser un tableau de 2^n éléments. Ce tableau est vu par le gestionnaire comme une roue temporelle sans limite (Figure 7.2). Chaque élément du tableau est en réalité une fenêtre temporelle d'une largeur égale à 2^k unités temporelles. Ainsi, un tour complet de l'échéancier s'effectue en 2^{n+k} unités temporelles. Au delà de cette taille, les événements continuent à être rangés dans les fenêtres correspondant à leur instant de programmation modulo 2^{n+k}. Les valeurs de **n** et de **k** sont fournies par l'outil utilisateur du gestionnaire lors de l'appel de création de l'échéancier (le simulateur SWISSE est un utilisateur du gestionnaire d'événements). Enfin, chaque fenêtre maintient la liste de ses événements triée par ordre croissant.

Le fait que chaque élément du tableau contienne 2^k unités temporelles permet d'éviter la présence de fenêtres temporelles vides, surtout quand l'unité de temps utilisée est très fine. Par exemple, dans le cas où l'unité de temps de base manipulée par le simulateur est la pico–

7.3.4. Mise sous tension du circuit

Cette phase est sollicitée avant la phase de simulation, et consiste à mettre le circuit dans un état cohérent et stable avant de pouvoir appliquer les stimuli. La particularité de la mise sous tension du circuit est qu'elle peut s'effectuer à l'aide d'une simulation zéro–délai.

7.3.5. Échéancier

L'échéancier est un module critique en terme de temps CPU dans un simulateur à pilotage événementiel. Comme il était possible d'utiliser plusieurs modèles d'événements, j'ai été amené à développer un échéancier générique, indépendant du modèle d'événement. Les deux seules caractéristiques obligatoires d'un événement étant de posséder une date et une clé. Une approche orientée objet a permis son exploitation par d'autres simulateurs de la chaîne ALLIANCE (un simulateur logique et un simulateur symbolique). Son architecture est explicitée plus loin dans ce chapitre.

7.3.6. Évaluation et résolution des cônes

C'est dans ce module que s'effectue le traitement des événements intervenant à l'instant courant de simulation. Chaque cône ayant un événement sur une de ses entrées est évalué ce qui peut générer un événement sur sa sortie.

7.3.7. Traitement des résultats

Les événements d'une simulation sur les signaux internes ou sur les sorties ont la même forme que les stimuli sur les entrées. C'est pourquoi, le simulateur range ses résultats dans une structure STM. Par la suite, des traducteurs vont transformer cette structure en un fichier exploitable par différents outils de visualisation.

7.3.8. Statistiques

Ce module permet d'obtenir des statistiques quant aux taux d'activités des différents signaux ou encore concernant les pourcentages des différents types de collisions lors d'une simulation.

7.3.9. Traces et mise au point (débogueur)

Un prototype de débogueur de simulation a été mis en œuvre à partir du squelette du débogueur de programmes du domaine public GDB. Ce module permet de piloter la simulation événement par événement. Il permet également d'ajouter, de supprimer ou de modifier des événements. Enfin, il permet de consulter l'état des signaux, de poser des points d'arrêt ou encore d'afficher des traces aidant dans la mise au point du simulateur.

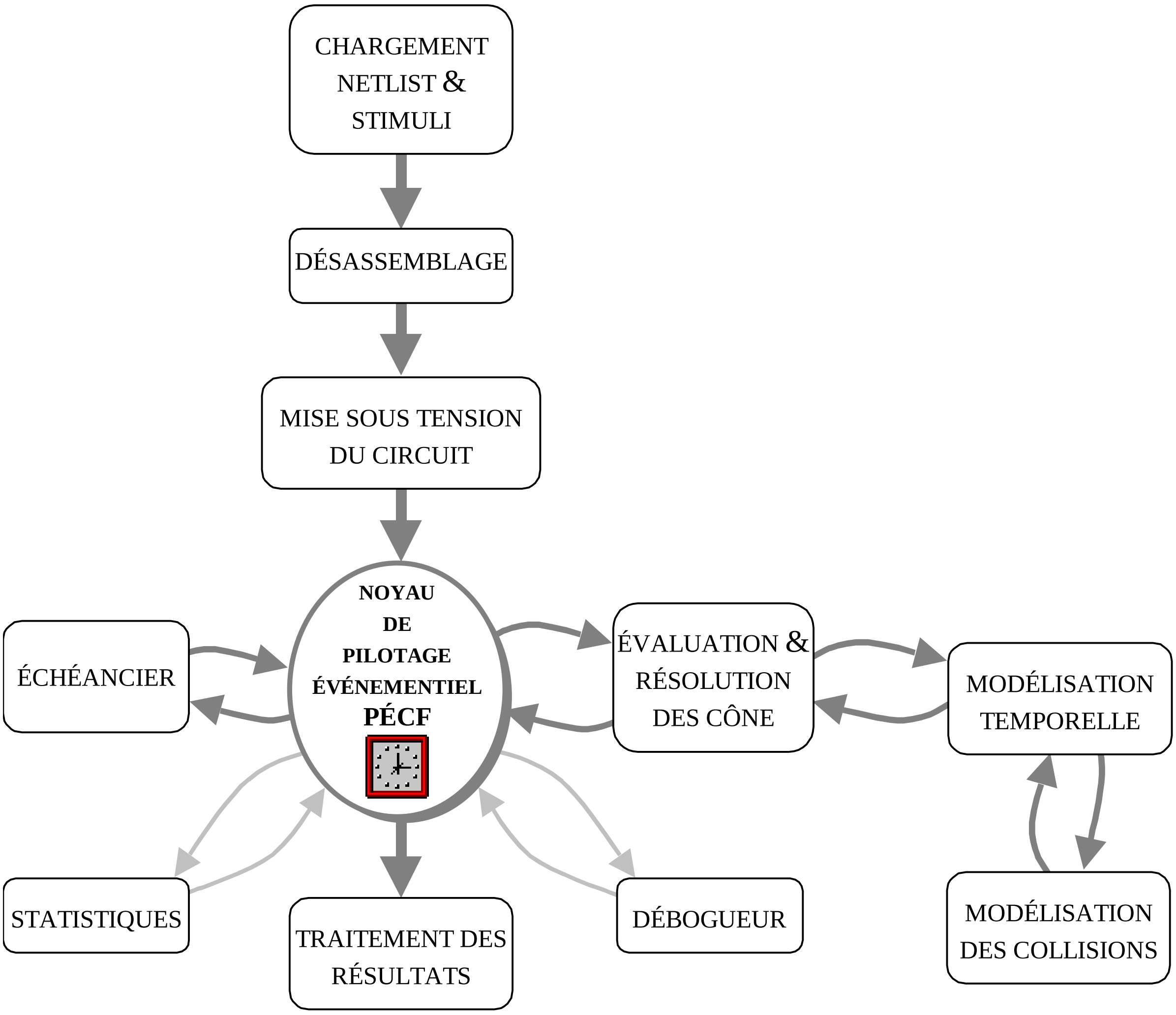


Figure 7.1 : Architecture du simulateur SWISSE

En ce qui concerne les stimuli, le même principe d'indépendance vis à vis du simulateur est appliqué. Le simulateur travaille sur une structure normalisée appelée STM qui décrit le concept de stimuli. Des compilateurs permettent de charger différents formats de stimuli vers cette structure. Pour le moment, deux formats de stimuli sont supportés : le format SPICE et le format PAT du simulateur logique d'ALLIANCE.

7.3.3. Désassemblage

La première tâche du simulateur SWISSE est de désassembler le circuit afin de construire une description en cônes. Il utilise à cet effet la bibliothèque de désassemblage mise à disposition par l'outil d'abstraction fonctionnel DESB [Lau94]. Les différents algorithmes mis en œuvre par ce module ont été exposés au chapitre 4.

7.1. Introduction

Ce chapitre présente une description d'un prototype de simulation appelé SWISSE [Abd97b], dans le but de valider la méthode de simulation exposée précédemment.

Nous décrivons ensuite l'architecture du simulateur avec ses différentes composantes. Dans cette description, nous insistons sur la conception du gestionnaire d'événements qui constitue le pivot de tout simulateur à pilotage événementiel.

7.2. Environnement logiciel

La chaîne de conception assistée par ordinateur, ALLIANCE [Gre92], propose un ensemble d'outils permettant la réalisation de la totalité des étapes de conception et de vérification d'un circuit intégré. La cohérence entre les différents outils est rendu possible grâce à une représentation standardisée des différentes vues d'un circuit intégré CMOS. Chaque outil accède à une vue donnée à travers une bibliothèque de fonctions, ce qui lui permet de s'affranchir des problèmes liés à la réalisation de certaines interfaces.

7.3. Architecture logicielle

La Figure 7.1 présente l'architecture interne du simulateur SWISSE. Chaque fonctionnalité est développée dans un module séparé ayant sa propre structure de données interne et privée. L'accès aux services de chaque module s'effectue exclusivement à travers un ensemble de fonctions représentant l'interface du module.

7.3.1. Noyau de pilotage événementiel PÉCF

Ce module est le noyau central du simulateur responsable de l'agencement des différentes étapes de simulation. C'est également dans ce module qu'est géré le temps courant qui détermine où en est la progression de la simulation.

7.3.2. Chargement

Le simulateur SWISSE supporte les mêmes formats pour la description d'entrée que ceux acceptés par l'analyseur TAS (COMPASS, SPICE, ALLIANCE), car ils interviennent tous deux au même niveau de description. Il s'agit d'une "net–list" au niveau transistors qui peut être hiérarchique. Dans ce cas, elle est automatiquement mise à plat au niveau transistors avant l'opération de désassemblage.

Chapitre 7

MISE EN ŒUVRE LOGICIELLE : LE SIMULATEUR SWISSE

6.6. Conclusion

Dans ce chapitre, nous avons commencé par donner une définition claire du phénomène de collision. Ensuite nous avons effectué une énumération des différents cas de collision, ainsi que des effets observés qui leurs sont associés. Enfin, nous avons proposé une modélisation adaptée à chaque cas de figure.

Nous avons défini ces modèles en gardant à l'esprit les contraintes de précision et de complexité, mais aussi, la nécessité de pouvoir les combiner pour traiter des cas encore plus complexes.

En résumé, le traitement proposé se ramène au traitement d'un événement simple, et permet de traiter plusieurs conflits successifs de différents types au fur et à mesure de leur occurrence.

Le chapitre suivant décrit une mise en œuvre de la méthode PÉCF dans un outil de simulation appelé SWISSE. Le but est de valider les principes exposés dans ces deux derniers chapitres et de montrer la pertinence de l'approche que nous proposons.

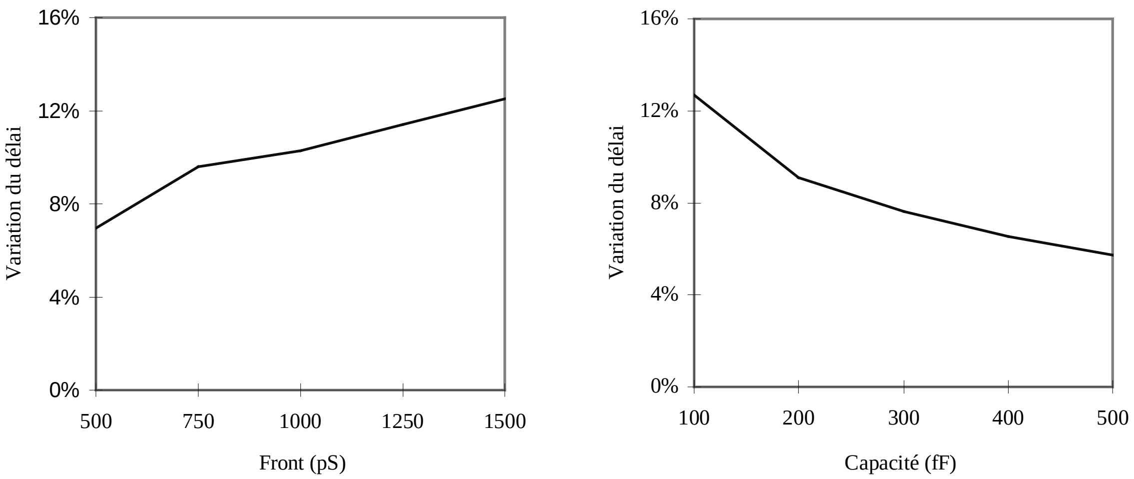


Figure 6.14 : Variation du délai pour des transitions montantes simultanées sur un NAND2

Heureusement, l'effet de ce type de collisions est nettement plus faible que celui des collisions intervenant sur des branches parallèles.

La Figure 6.14 montre l'erreur obtenue si l'effet de collision est totalement négligé, et ce dans une configuration pire cas; c'est à dire pour des entrées commutant ensemble sans décalage dans le temps. Nous constatons que l'erreur ne dépasse pas les 12%.

Toutes ces constatations ont motivé l'utilisation d'une approche pragmatique pour traiter ce type de conflits. Cela consiste simplement à choisir parmi l'ensemble des **N** entrées, qui sont en commutation à l'instant en cours $t_{\text{présent}}$, celle qui sera considérée comme l'*entrée dominante* **E_D** avec son *front dominant* **F_D**. C'est à dire, l'entrée dont le signal traverse $V_{DD}/2$ en dernier. Il permet de tenir compte de configurations particulières telle que celle montrée dans la Figure 6.15.

Ainsi, nous ramenons le problème à celui d'une commutation simple, en remplaçant l'ensemble des événements par l'événement dominant associé à une seule des entrées.

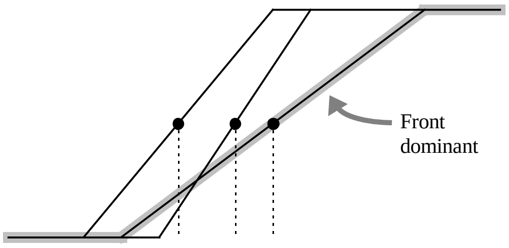
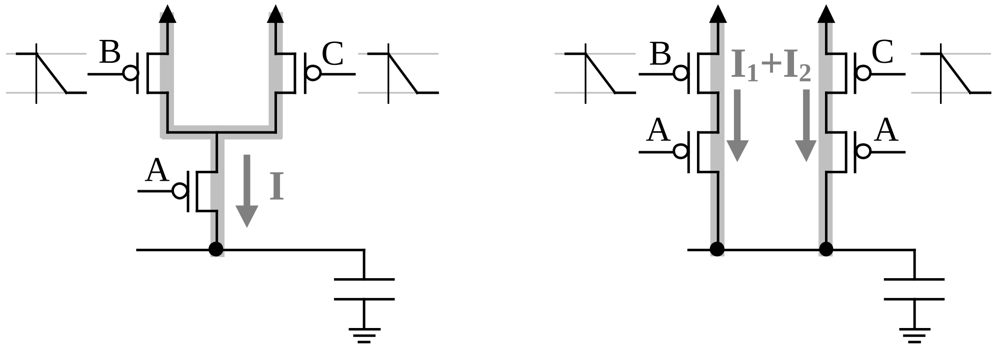


Figure 6.15 : Le front dominant n'est pas forcément celui imposé par le dernier événement



(a) Structure réelle (b) Représentation vue par le simulateur

Figure 6.13 : Collision de type série-parallèle

Si par la suite, aucun événement ne vient supprimer l'état conflictuel du signal dans un temps "raisonnable" (paramètre donné par l'utilisateur), le conflit est alors déclaré comme étant persistant et la simulation s'arrête.

6.4.5. Cas particuliers

Comme indiqué au chapitre 4, la représentation des cônes sous la forme d'un ensemble de branches parallèles ne reflète pas toujours la topologie réelle du cône. Dans certains cas (transistor commun à plusieurs branches), la représentation en branches indépendantes est susceptible de fausser les calculs.

La Figure 6.13 montre une configuration où l'on détecte une collision de type parallèle entre les signaux **B** et **C**. Si l'on se fie à la représentation du cône après désassemblage (Figure 6.13–b), nous sommes en présence de deux courants indépendants, I_1 et I_2, alors qu'en réalité (Figure 6.13–a), les deux branches ne sont pas indépendantes. Utiliser ces deux courants de manière cumulative serait trop optimiste. En effet, le transistor commandé par le signal **A** joue dans ce cas le rôle de limiteur de courant.

Dans notre méthode, nous tenons compte de ce type de configurations, en recherchant les branches qui possèdent un transistor en commun. Dans ce cas, seule la branche qui présente le courant maximal est conservée lors du calcul des temps de propagation et de commutation.

6.5. Modélisation des collisions entre des événements sur une même branche

Ce type de collisions met en jeu des événements concurrents sur plusieurs signaux pilotant des transistors appartenant à une même branche. Les phénomènes électriques qui entrent en jeu dans ce type de configurations sont très complexes et font intervenir la valeur des nœuds internes. Il est donc difficile de les prendre en considération sans recourir à des modèles complexes qui dégradent les performances de simulation.

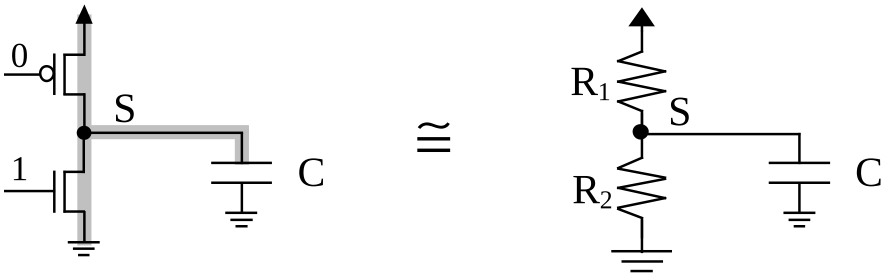


Figure 6.12 : Modèle utilisé dans le calcul des paramètres d'un conflit

ou encore

$$C\frac{dV_S}{dt}+\left(\frac{1}{R_1}+\frac{1}{R_2}\right)V_S=\frac{1}{R_1}V_{DD} \tag{6.13}$$

Soit t_0 l'instant de passage de la transition sur l'entrée provoquant le conflit par la tension de seuil V_T; c'est à dire, l'instant où le conflit se déclenche. Soit, V_{t0}, la valeur de la tension en S à l'instant t_0. Dans ce cas, la solution de l'Équation 6.13 s'écrit :

$$V_S(t)=\left(V_{t0}-\frac{R}{R_1}V_{DD}\right)e^{-\frac{(t-t_0)}{RC}}+\frac{R}{R_1}V_{DD}\,,\ \text{avec}\ R=\frac{R_1\,R_2}{R_1+R_2} \tag{6.14}$$

Ainsi, la tension V_S tend, suivant une fonction exponentielle, vers une valeur finale V_{fin}, telle que

$$V_{fin}=V_{DD}\frac{R_2}{(R_1+R_2)} \tag{6.15}$$

Or, nous avons dans cette thèse adopté un modèle de signal en tangente hyperbolique. Nous utilisons alors une approximation de la fonction exponentielle, valable pour les petites valeurs de **t** :

$$e^{-\frac{t}{RC}}\cong 1-\mathrm{th}(t/RC) \tag{6.16}$$

ce qui nous donne :

$$V_S(t)=\left(V_{t0}-\frac{R}{R_1}V_{DD}\right)\left(1-\mathrm{th}((t-t_0)/RC)\right)+\frac{R}{R_1}V_{DD} \tag{6.17}$$

Nous obtenons ainsi les caractéristiques de l'événement sur **S** : l'instant de programmation t_0, la valeur finale V_{fin} et le front F = RC.

que l'on résout de façon itérative pour calculer t_{sn} en appliquant le théorème du point fixe.

La dérivée en ce point est obtenue par

$$a = \frac{dv}{dt} = \frac{\sum_{i=1,n} I_i(t - t_{vti})}{C} \tag{6.11}$$

à partir de laquelle on calcule la nouvelle constante de temps F_{sn} à l'aide de l'Équation 5.1.

6.4.2. Calcul de l'instant de prise en compte de l'accélération sur la sortie

Le but final du traitement est de trouver l'instant de départ $t_{évt}$ à partir duquel les nouveaux paramètres deviennent opérationnels. Le calcul de cet instant revient à trouver le point d'intersection de la courbe définie par les nouveaux paramètres avec l'ancienne forme de la sortie. La solution est donnée par l'Équation 6.4 avec les conditions d'existence définies par l'Équation 6.5. Cet instant est utilisé comme instant de l'événement à programmer dans l'échéancier.

6.4.3. Cas produisant une décélération

Le calcul présenté dans cette section s'applique également quand l'événement en cours de traitement fait passer une des branches passantes dans l'état bloqué. Dans ce cas, nous observons une décélération de la commutation en sortie due à la réduction du courant global.

Néanmoins, le calcul des paramètres de l'événement généré en sortie s'effectue d'une manière identique à celui provoquant une accélération.

6.4.4. Cas des branches conflictuelles

Un conflit de courants se traduit par la présence simultanée de branches passantes vers les deux alimentations V_{SS} et V_{DD}. Il y a deux types de conflits : les *conflits transitoires* et les *conflits persistants*. Les conflits transitoires sont dus à des temps de propagations différents à travers le circuit. Quand un conflit est de type persistant, nous pouvons considérer que c'est une erreur qui doit être corrigée par le concepteur.

Pour les conflits transitoires, nous proposons un traitement qui assimile l'ensemble à un pont diviseur en remplaçant les branches par des résistances linéaires équivalentes (Figure 6.12). Dans ce cas, l'application de la loi de Kirchhoff sur les courants nous permet d'écrire l'équation différentielle suivante au nœud **S** :

$$C\frac{dV_S}{dt} + \frac{V_S - V_{DD}}{R_1} + \frac{V_S}{R_2} = 0 \tag{6.12}$$

t_{sn} ainsi que la nouvelle constante de temps. Ensuite, il s'agit d'en déduire l'instant $t_{évt}$ à partir duquel ces caractéristiques prennent effet et de programmer cet événement dans l'échéancier.

Le calcul de t_{sn} revient à calculer l'instant **t** tel que (Figure 6.11)

$$\int_{t_{vt1}}^{t} I_1(t - t_{vt1})dt + \int_{t_{vt2}}^{t} I_2(t - t_{vt2})dt + \ldots + \int_{t_{vtn}}^{t} I_n(t - t_{vtn})dt = Q_S = C\,V_{DD} / 2 \qquad (6.6)$$

ou encore en terme de fonction de charge

$$Q_1(t - t_{vt1}) + Q_2(t - t_{vt2}) + \ldots + Q_n(t - t_{vtn}) = Q_S = C\,V_{DD} / 2 \qquad (6.7)$$

Or, d'après le modèle MCC, $Q_k(t)$ peut s'écrire sous la forme :

$$Q_k(t) = \mathrm{Imax}_k + F_k\,\mathrm{Imax}_k f_k(t) \qquad (6.8)$$

où $f_k(t)$ est une fonction qui dépend du temps et des paramètres technologiques du modèle.

Ainsi, l'Équation 6.7 peut s'écrire :

$$\left(\sum_{i=1,n} \mathrm{Imax}_i\right) t - \sum_{i=1,n} \mathrm{Imax}_i\left(t_i - F_i f_i(t - t_i)\right) = CV_{DD} / 2 \qquad (6.9)$$

Mise sous la forme x = f(x), nous avons :

$$t = \frac{C\frac{V_{DD}}{2} + \sum_{i=1,n} \mathrm{Imax}_i\left(t_{vti} - F_i f_i(t - t_{vti})\right)}{\sum_{i=1,n} \mathrm{Imax}_i} \qquad (6.10)$$

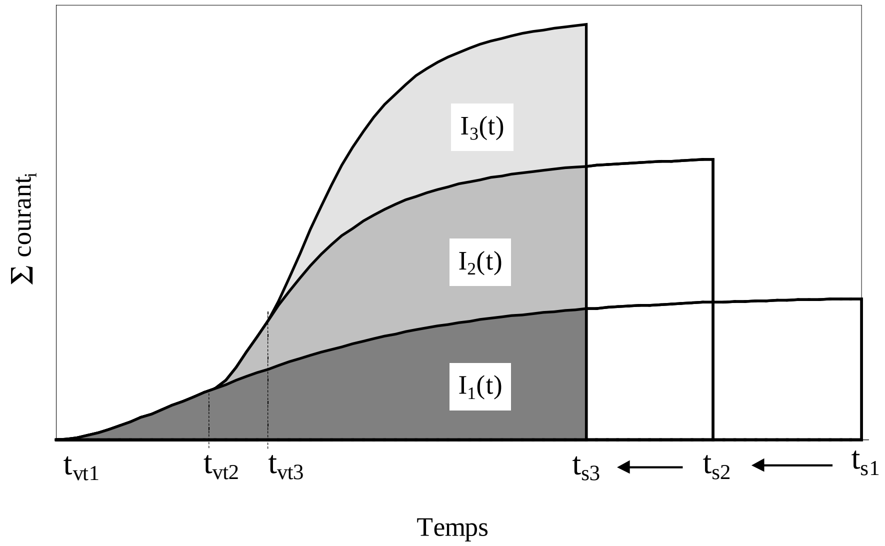


Figure 6.11 : Calcul du nouvel instant de basculement suite à une collision de type parallèle

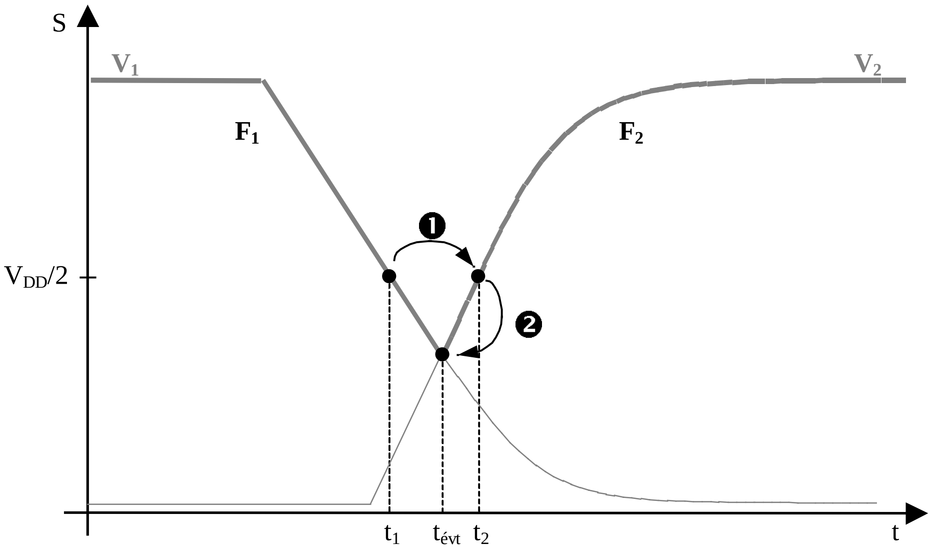


Figure 6.10 : Différentes étapes dans le traitement d'un glitch

6.4. Modélisation des collisions entre des événements sur des branches parallèles

Ce type de collision met en jeu plusieurs branches conduisant en parallèle, ce qui se traduit par un effet d'accélération ou de décélération de la commutation en sortie. Dans un premier temps, nous allons nous intéresser au cas de l'accélération, c'est à dire, au cas où plusieurs branches deviennent conductrices. C'est de loin le cas qui, s'il est ignoré, génère le taux d'erreur le plus élevé dans le calcul des temps de propagation. L'explication en est très simple : charger ou décharger une capacité avec deux sources de courant identiques est deux fois plus rapide. L'effet d'une telle configuration est cumulatif. Chaque fois qu'une nouvelle branche commence à conduire, il est nécessaire d'effectuer une nouvelle résolution du cône si la sortie n'a toujours pas atteint sa valeur finale.

Fort heureusement, les phénomènes électriques sont faciles à comprendre. La charge ou la décharge de la capacité de sortie s'effectue à l'aide de plusieurs sources de courant qui s'additionnent. Il faut bien évidemment tenir compte du décalage temporel qu'il peut y avoir entre les instants où les différentes branches commencent à conduire. Une modélisation adéquate consisterait à intégrer le courant apporté par la nouvelle branche dans les calculs effectués pour en déduire le point de basculement.

6.4.1. Calcul du nouvel instant de commutation sur la sortie

Considérons un cône où n-1 branches sont en train de faire commuter la sortie d'un état stable vers l'état stable opposé. Supposons qu'à un instant t_c un événement rende passante une $n^{ième}$ branche. L'objectif est alors de calculer le nouvel instant de passage de la sortie par $V_{DD}/2$

Cette dernière équation nous ramène au cas d'une commutation simple mettant en jeu un courant I' supposé se déclencher à l'instant t'_{vt} avec la constante de temps F'. La seule différence apparaît dans le terme de droite, auquel il faut ajouter le terme correctif $[Q(t_c-t_{vt}) - Q'(t_c-t'_{vt})]$.

Ainsi, nous pouvons calculer les paramètres t'_s et F'_s de l'événement en sortie, en utilisant les mêmes fonctions que pour une transition simple, pourvu que l'on ait pris soin, au préalable, d'apporter une correction à la charge initiale.

6.3.2. Calcul de l'instant de prise en compte du changement sur la sortie

Une fois les nouveaux paramètres de la commutation de la sortie calculés, nous avons besoin de trouver l'instant $t_{évt}$ à partir duquel ils deviennent opérationnels. $t_{évt}$ définit l'instant auquel l'événement sur la sortie doit être programmé dans l'échéancier. Il s'agit en fait de calculer le point d'intersection entre la courbe courante sur la sortie et la nouvelle courbe calculée suite au changement de front. La solution de cette équation s'écrit

$$t_{évt} = \frac{F'_s t_{vt} - F_s t'_{vt}}{F'_s - F_s} \tag{6.4}$$

Les conditions pour qu'une telle solution existe sont :

$$\begin{cases} F_s \neq F'_s \\ \dfrac{t_{vt} - t'_{vt}}{F_s - F'_s} \leq 0 \end{cases} \tag{6.5}$$

Si cette solution n'est pas vérifiée, cela signifie que les deux courbes ne se croisent pas entre V_{DD} et V_{SS}. Dans ce cas, c'est le second événement qui sera programmé dans l'échéancier, en invalidant le premier événement.

6.3.3. Cas de deux événements de sens opposés

Quand les deux événements sont de sens opposés, la sortie effectue une commutation incomplète puis revient à son état initial (glitch). Le calcul du nouvel instant de commutation sur la sortie est semblable au cas précédent (Figure 6.10). Toutefois, dans le cas présent, contrairement au cas de deux événements de même sens, il s'agit de traiter à nouveau la partie de la charge déjà traitée suite à la première commutation. Dans le cas présenté Figure 6.10, c'est la charge perdue par la capacité suite au premier événement qu'il nous faut charger à nouveau.

Le calcul de l'instant de prise en compte du changement sur la sortie consiste ici aussi à rechercher l'intersection de deux courbes mais de sens opposés.

propre à notre méthode. Elle est dûe au fait que le traitement de la collision est basé sur un nouveau calcul du point de passage de la sortie par $V_{DD}/2$. Or, il est possible que la sortie ait déjà traversé ce point à l'instant où le second événement intervient. Si tel est le cas, nous ignorons l'événement courant car son effet sur la transition en sortie est négligeable.

6.3.1. Calcul du nouvel instant de commutation sur la sortie

Il s'agit dans cette première étape de calculer le nouvel instant t'_s de passage de la sortie par le seuil logique $V_{DD}/2$. A cet effet, nous tenons compte du nouveau courant I'(t) dû à F'_E, ainsi que de l'instant courant t_c, à partir duquel le nouveau courant prend effet. Il suffit pour cela d'intégrer par morceaux l'équation du courant comme le montre la Figure 6.9 : une intégrale sur I(t) (en gris clair sur la figure) et une intégrale sur I'(t) (en gris foncé sur la figure). Plus formellement, ceci revient à résoudre l'équation :

$$\int_{t_{vt}}^{t_c} I(t - t_{vt})\,dt + \int_{t_c}^{t'_s} I'(t - t'_{vt})\,dt = Q_S = C\,V_{DD} / 2 \qquad (6.1)$$

En ajoutant de chaque côté le terme $\int_{t'_{vt}}^{t_c} I'(t - t'_{vt})\,dt$, l'équation devient

$$\int_{t'_{vt}}^{t'_s} I'(t - t'_{vt})\,dt = C\frac{V_{DD}}{2} - \int_{t_{vt}}^{t_c} I(t - t_{vt})\,dt + \int_{t'_{vt}}^{t_c} I'(t - t'_{vt})\,dt \qquad (6.2)$$

Soit finalement, en remplaçant par la fonction de charge

$$Q'(t - t'_{vt}) = C\frac{V_{DD}}{2} - \left[Q(t_c - t_{vt}) - Q'(t_c - t'_{vt})\right] \qquad (6.3)$$

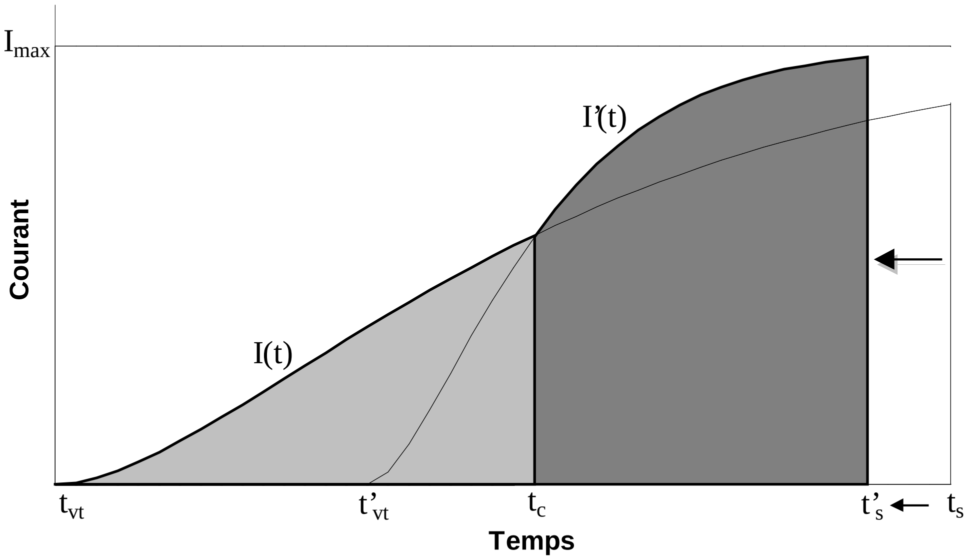


Figure 6.9 : Calcul du nouvel instant de basculement par une analyse du courant suite à un changement de la constante de temps

événement peut intervenir sur ce même signal alors qu'il n'a pas atteint son état stable final. Ce cas est une conséquence du modèle événementiel choisi. Supposons, dans un premier temps, que le nouvel événement est du même signe que l'ancien. La courbe de la Figure 6.7, obtenue par des simulations SPICE, montre les effets de la collision sur le temps de propagation, en fonction du décalage entre les deux événements.

C'est en effectuant une analyse de l'effet de changement de la constante de temps sur le courant, que nous allons déduire le nouvel instant de basculement de la sortie. Un tel traitement s'effectue en deux étapes (Figure 6.8) : une première étape pour déterminer le nouvel instant de passage de la sortie par la tension $V_{DD}/2$ du cône (Figure 6.8–❶) et une deuxième étape pour calculer la date de l'événement à programmer dans l'échéancier pour la prise en compte de ce changement au niveau de la sortie (Figure 6.8–❷).

Considérons une entrée **E**, qui commute à un instant donné suivant une constante de temps F_E, et faisant commuter la sortie **S** avec un courant **I**. Supposons qu'un nouvel événement, possédant une constante de temps F'_E, intervienne sur cette entrée à l'instant t_c, tel que, $t_{vt} < t_c < t_s$; t_s est l'instant de passage de la sortie par $V_{DD}/2$ en l'absence de collision, et t_{vt} est l'instant de passage de l'entrée par la tension de seuil V_T. Une conséquence du changement de front est un changement dans le courant qui traverse la branche, impliquant un retard ou une accélération de la commutation en cours sur la sortie.

La condition $t_{vt} < t_c$ traduit le fait qu'il est inutile de tenir compte de la constante de temps F_E tant que $t_c \leq t_{vt}$, car le courant est nul avant l'instant t_{vt}. La condition $t_c < t_s$ est

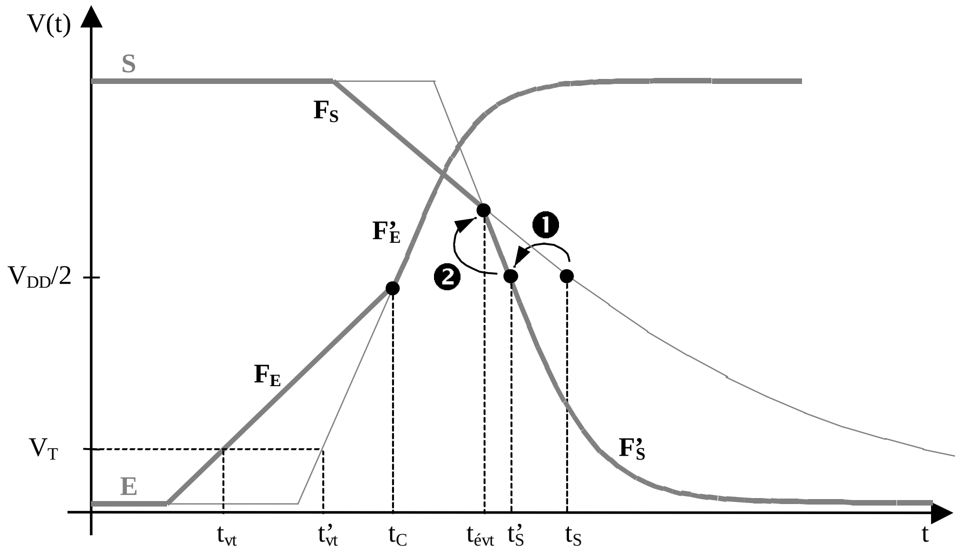


Figure 6.8 : Différentes étapes dans le traitement d'un changement de la constante de temps

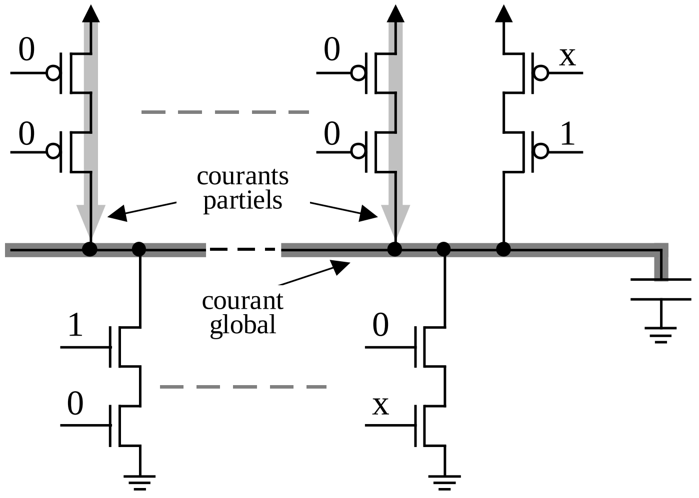


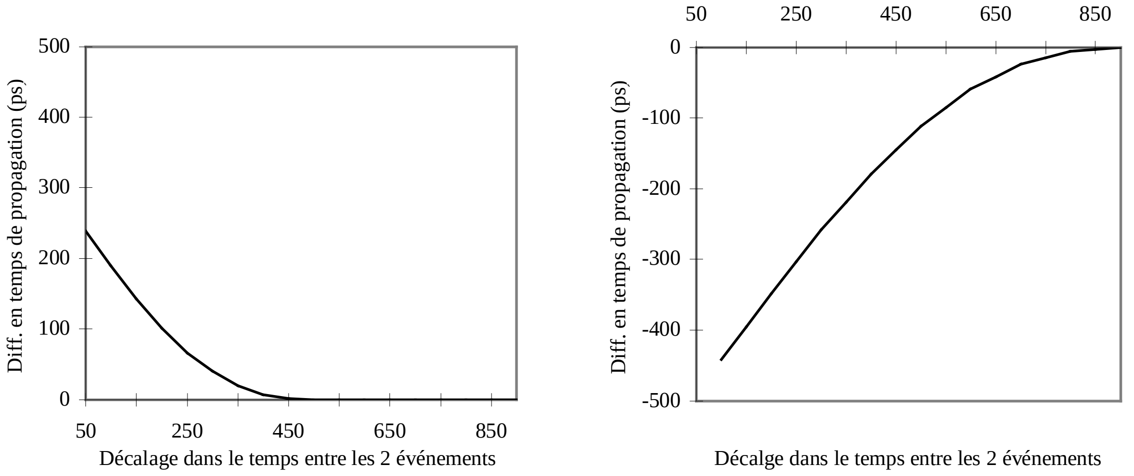
Figure 6.6 : La sortie d'un cône vue comme un bus à émetteurs multiples

La mise en œuvre de cette technique est favorisée par la stratégie de désassemblage présentée au chapitre 4, et qui représente chaque cône sous la forme d'un ensemble de branches parallèles reliant la sortie aux alimentations. Cette représentation s'intègre bien dans l'algorithme événementiel PÉCF, et permet une évaluation aisée des courants qui sont à la base de notre méthode de calcul des délais, et ce, dans tous les cas de figure.

Définition 6.3 : Une branche **B** est dans un des deux états suivants : *bloqué* ou *passant*. Une branche est dans un état passant, si tous les transistors appartenant à la branche laissent passer le courant, établissant ainsi un chemin électrique entre la sortie et une des deux alimentations. □

6.3. Modélisation des collisions entre deux événements sur une même entrée

Un signal est considéré dans un état instable si sa constante de temps $F \neq 0$. Or un



(a) Changement de front dans le sens accélération (b) Changement de front dans le sens décélération

Figure 6.7 : Perturbation du temps de propagation d'un inverseur suite à un changement de la constante de temps du signal d'entrée

6.2.3.3. Collision d'événements sur des entrées différentes dans la même branche

Ce type de collisions se produit quand plusieurs événements surviennent sur la grille de transistors appartenant à la même branche. Si tous les événements sont de nature à rendre la branche passante, il en résulte un retard de la commutation de la sortie par rapport à une commutation sans collisions (Figure 6.5). Ce retard dépend des caractéristiques des transistors, des fronts sur les entrées et du décalage temporel entre les événements. Mais comme le montre l'exemple de la Figure 6.1–c, la variation du délai reste dans ce cas assez faible.

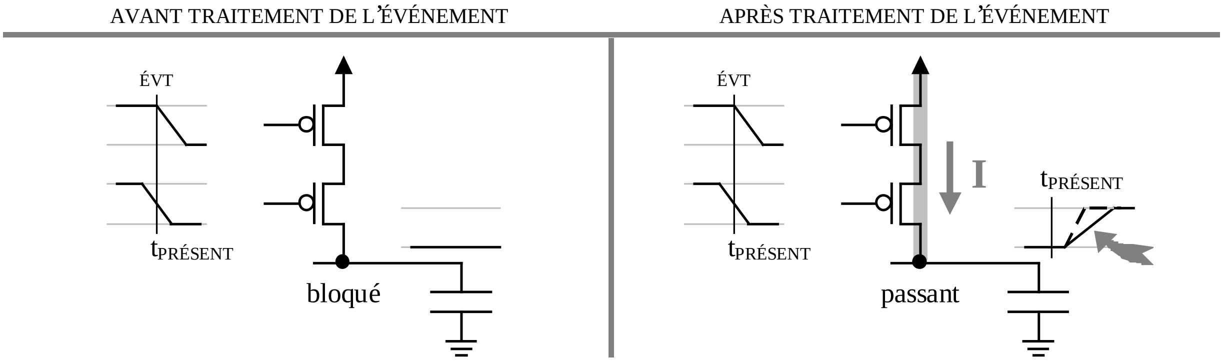


Figure 6.5 : Impact d'une collision impliquant deux entrées sur une même branche

6.2.4. Objectif

Notre objectif est de définir un modèle temporel pour chaque type de collisions. Dans la définition de chacun de ces modèles, nous sommes en quête d'un compromis entre la précision recherchée et la complexité du calcul.

Par ailleurs, il faut garder à l'esprit que plusieurs collisions de même type ou de types différents peuvent se produire sur les entrées d'un cône. Pour éviter la multiplication des modèles, nous sommes amenés à définir des modèles de calcul de telle manière qu'ils puissent être combinés les uns aux autres.

La solution à ces problèmes repose sur une analyse des courants responsables de la commutation du signal de sortie.

6.2.5. Modèle des courants

L'idée directrice de l'approche que nous proposons est de considérer la sortie comme un bus piloté par plusieurs émetteurs, chaque branche représentant un émetteur potentiel d'un courant partiel participant au courant global (Figure 6.6). Un changement dans les paramètres d'un de ces courants partiels affecte le courant global et provoque une résolution du cône pour déterminer l'impact de ce changement sur la sortie. Nous appellerons cette approche, *la technique d’évaluation par analyse des courants dans les branches.*

6.2.3.1. Collision de deux événements sur une même entrée

Il s'agit d'un événement intervenant sur une entrée qui est déjà en cours de commutation. Le nouvel événement indique un changement dans la constante de temps représentant le front du signal. Si la branche à laquelle appartient cette entrée est non passante, cette collision reste sans effet. Si la branche est passante, il y a deux possibilités : si le nouvel événement est du même signe que l'ancien, il s'ensuit soit une accélération (Figure 6.3), soit une décélération de la commutation en cours sur la sortie. En revanche, s'il est de signe opposé, alors c'est un "glitch" qui est propagé vers la sortie.

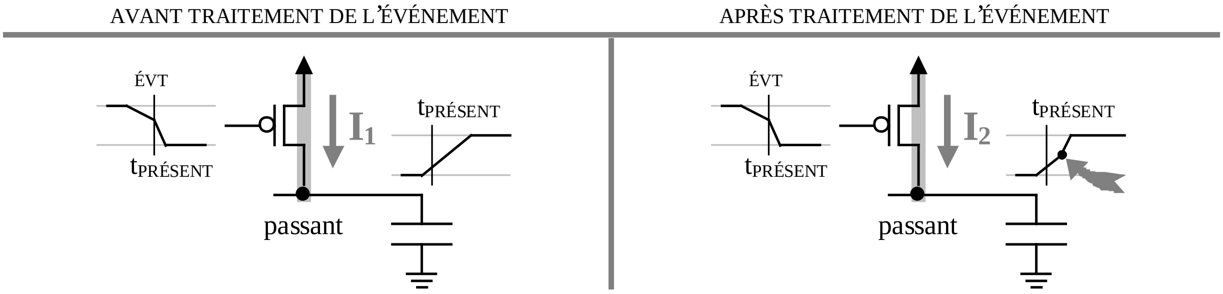


Figure 6.3 : Impact d'une collision entre événements sur une même entrée

6.2.3.2. Collisions d'événements sur des entrées dans des branches parallèles

Ce type de collisions se produit sur des entrées qui contrôlent les transistors appartenant à des branches différentes. Dans ce cas, le dernier événement rend passantes ou non passantes des branches, modifiant ainsi le courant global qui charge ou décharge la capacité de sortie.

Si les branches en parallèle conduisent vers la même alimentation, il en résulte une accélération (Figure 6.4) ou une décélération de la commutation en sortie. Cet effet dépend des courants dans les branches parallèles, ainsi que du décalage dans le temps entre les événements. Dans le cas contraire, il s'agit d'un conflit provoquant la commutation de la sortie vers un état instable qui dépend du rapport entre les courants conflictuels.

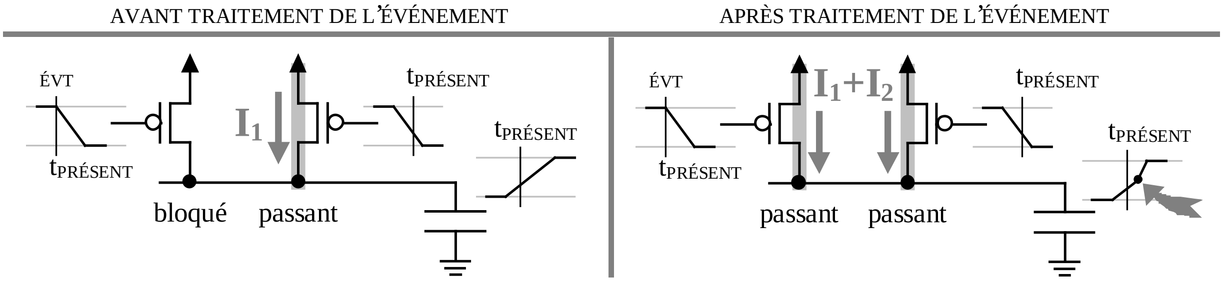


Figure 6.4 : Impact d'une collision entre événements impliquant des branches différentes

Définition 6.1 : Il y a un phénomène de collision directe, si et seulement si :
sur une entrée E' de C, non forcément différente de E, il existe
un événement antérieur Evt'(t', F', V'_{fin}, E'), tel que, $t' \leq t_c$ et $V_{E'}(t_c) \neq V'_{fin}$,
□

Définition 6.2 : Il y a un phénomène de collision indirecte, si et seulement si :
sur la sortie S de C, il existe déjà
un événement de fin de commutation, programmé à l'instant t", tel que, $t'' > t_c$
□

La Définition 6.1 regroupe les configurations où les commutations de plusieurs entrées d'un même cône se chevauchent, ainsi que le cas d'un événement de changement de la constante de temps sur une même entrée.

La Définition 6.2 regroupe les configurations où un événement intervient sur une des entrées sans chevauchement avec d'autres commutations sur les entrées alors que la sortie n'est pas dans un état stable. Ce type de configurations intervient pour des entrées commutant très rapidement. La Figure 6.2 montre un exemple typique d'une collision indirecte. Dans cet exemple, les deux commutations complètes successives à l'entrée de l'inverseur sont suffisamment proches l'une de l'autre pour que la deuxième perturbe le courant, non encore nul, dû à la première, empêchant ainsi la sortie de compléter sa première commutation.

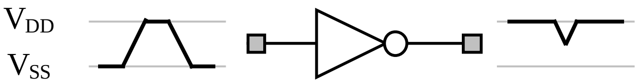
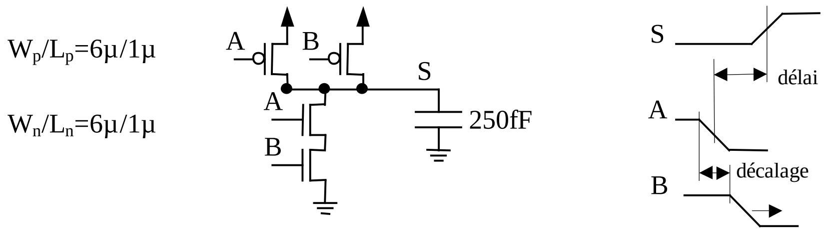


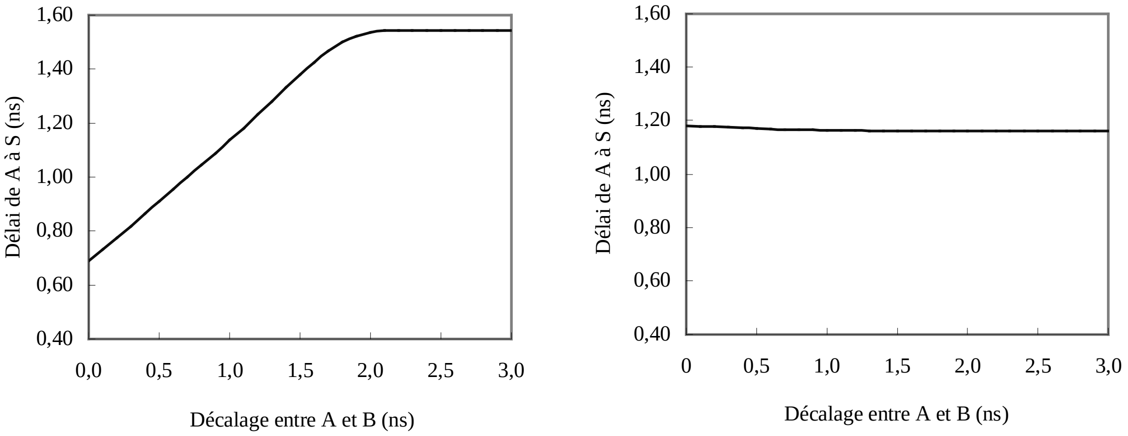
Figure 6.2 : Une collision générée par deux commutations complètes en entrée

Ces deux définitions donnent une description formelle et complète de la notion de proximité temporelle des événements. Elles s'intéressent non seulement aux configurations présentes sur les entrées d'un cône, mais également à l'effet que peut avoir un événement sur les courants qui entrent en jeu dans la commutation de la sortie. L'exemple de la Figure 6.2 montre bien qu'il est impossible de détecter la présence d'une collision uniquement à partir de l'analyse des commutations en entrée.

Nous pouvons distinguer trois configurations de base, présentant un phénomène de collision, directe ou indirecte, que nous allons présenter succinctement :



(a) Porte Nand à deux entrées



(b) Variation du délai pour des transitions descendantes des entrées

(c) Variation du délai pour des transitions montantes des entrées

Figure 6.1 : Effet de la proximité des commutations sur une porte Nand

branches parallèles sont mises en jeu. Ce type de phénomènes doit être modélisé si nous sommes à la recherche de précision dans notre calcul temporel.

Considérons maintenant le cas où les deux transitions sont montantes. Comme le montre la Figure 6.1–c, le temps de propagation devient alors une fonction très légèrement décroissante du décalage entre les deux transitions. La variation du temps de propagation reste inférieure à 10%. Des simulations en 0,8 et 0,7 micron font apparaître que cette faible variation de délai reste relativement identique. Dans ce cas (des collisions sur une même branche), une modélisation complexe ayant pour but la prise en compte de ce phénomène n'est pas justifiée.

6.2.3. Définition et taxinomie

Soit un événement Evt(t_c, F, V_{fin}, E) à l'instant courant t_c, avec un front F ≠ 0, intervenant sur une entrée E d'un cône C (le signal de sortie est S). Nous constatons deux types de collisions possibles : les collisions, dites *collisions directes,* qui peuvent être détectées à partir des événements sur les entrées de P, et les collisions, dites *collisions indirectes,* qui ne peuvent être détectées qu'en observant la sortie de C; plus formellement :

6.1. Introduction

Nous avons présenté au chapitre précédent une nouvelle approche de la simulation à pilotage événementiel, où un événement est défini comme un changement de la valeur de la constante de temps du signal. La méthode générale de propagation, ainsi que la technique d'évaluation locale ont été explicitées dans le cas d'une commutation simple à l'entrée d'un cône.

Or, au cours d'une simulation, il peut se produire des commutations multiples sur les différentes entrées d'un cône. En règle générale, nous sommes en présence d'une configuration complexe quand plusieurs commutations interviennent sur les entrées d'un cône, avec une certaine proximité temporelle. Ce phénomène est appelé "collision" dans la suite de ce chapitre.

Ce phénomène ne peut être négligé : dans des circuits où les chemins combinatoires ont été équilibrés, le taux de collision peut atteindre 20%.

6.2. Modèle d'évaluation par analyse des courants dans les branches

6.2.1. Position du problème

Alors que la modélisation des commutations simples a reçu beaucoup d'attention par le passé [Pen81] [Sub86] [Auv90] [Sak91], celle du phénomène de collision est plus récente [Jun89] [Mel92] [Nab94] [Cha96]. Malheureusement, les modèles proposés, qui tiennent compte de l'effet de la proximité temporelle des événements sur les entrées qui commutent, restent incomplets et, surtout, difficilement intégrables au sein d'un algorithme de simulation.

6.2.2. Étude quantitative

Afin d'illustrer l'effet de la proximité des transitions sur les temps de propagation d'une porte, considérons une porte NAND (Figure 6.1–a), et supposons que les deux entrées A et B effectuent toutes deux des transitions descendantes. La Figure 6.1–b illustre la variation du temps de propagation en fonction du déphasage entre les transitions de A et de B. Quand ce déphasage est assez grand, la transition de B reste sans effet sur le temps de propagation. La transition de S ne dépend alors que de la transition de A. Cependant, quand ce déphasage diminue, la branche pilotée par B commence à apporter sa participation à la commutation de la sortie. Il en résulte un temps de propagation deux fois plus court.

Cet exemple indique clairement que les variations dans les temps de propagation dues à la proximité temporelle des commutations en entrée peuvent être importantes dans le cas où des

Chapitre 6

MODÉLISATION DES COLLISIONS

- L'abstraction du circuit au niveau cônes (portes) permet l'utilisation d'un algorithme à pilotage événementiel, tirant ainsi un maximum de profit de la faible activité des circuits. De plus, l'absence de bibliothèque de référence confère à cette méthode une grande généralité, puisque la description initiale est un réseau de transistors extrait du dessin des masques.
- L'introduction d'un nouveau concept d'événement (changement de la constante de temps dans la fonction représentant le signal) permet de réduire considérablement le nombre d'événements à traiter. Là où plusieurs événements sont nécessaires pour traiter une commutation dans des algorithmes logico-temporels classiques, un seul est suffisant dans l'approche que nous préconisons. Il en résulte naturellement une plus grande rapidité d'exécution, confirmée par les résultats d'expérimentations du Chapitre 8.
- Les délais sont calculés suivant les caractéristiques courant–tension des transistors avec un modèle de courant adapté aux technologies submicroniques d'aujourd'hui. Il s'ensuit des temps de propagation et de commutation d'une précision de l'ordre de 5% par rapport à SPICE.
- L'utilisation d'une représentation du signal par une tangente hyperbolique permet de rendre compte de manière réaliste de l'effet de front dans le calcul des temps de propagation. De plus, cette modélisation facilite les échanges avec le simulateur électrique chargé de simuler les parties analogiques du circuit.

Toutefois, à ce stade de la description de la méthode PÉCF, nous avons uniquement présenté l'évaluation de la sortie d'un cône dans un cas simple. Bien que la configuration traitée précédemment constitue le cas le plus fréquent, il nous faut également traiter les configurations mettant en jeu des commutations multiples sur les entrées d'un cône.

et

$$t_S = t_{vdd/2} + \frac{F_S}{2} \ln\left(\frac{(V_{fin} + V_S - 2V_T)(V_{fin} - V_{DD}/2)}{(V_{fin} - 2V_T + V_{DD}/2)(V_{fin} - V_S)}\right) \tag{5.2}$$

Enfin, l'événement est programmé dans l'échéancier pour être traité à l'instant t_S.

□

D'ores et déjà, il est utile de signaler que l'étape ❷ constitue le cœur du calcul des temps de propagation. En effet, c'est la seule étape qui nécessite une résolution numérique itérative servant à déduire l'instant de basculement logique du signal.

Ainsi, en partant d'un événement simple en entrée d'un cône, nous pouvons déduire l'événement en sortie suite à une seule itération du traitement décrit ci-dessus.

5.5. Les événements de fin de commutation

Jusqu'à présent, nous avons défini un événement comme étant un changement de valeur du paramètre F. Or, le modèle analytique du signal est une fonction qui tend vers une asymptote qui n'est jamais atteinte par la fonction V(t). Pour définir la fin de la transition, nous utilisons la notion d'*événement de fin de commutation*. Cet événement traduit le fait que le signal a atteint une valeur stable. Par conséquent, son paramètre F vaut zéro.

La date de l'événement de fin de commutation, t_{fin}, est calculée en même temps que celle de début de commutation, $t_{début}$, de la manière suivante :

$$t_{fin} = t_{début} + K\,(t_{vdd/2} - t_{début}) \tag{5.3}$$

K étant une constante de simulation d'une valeur typique égale à 4.

Si, par la suite, un événement de changement du paramètre temporel est programmé à un instant $t'_{début} < t_{fin}$, l'événement de fin de commutation programmé à t_{fin} est supprimé par application du Principe 5.2. Un autre événement de fin de commutation viendra le supplanter à un instant, t'_{fin}.

5.6. Conclusion

Dans ce chapitre, nous avons présenté une méthode de simulation logico-temporelle qui s'applique à une description du circuit numérique obtenue après désassemblage. Les avantages de cette approche sont les suivants :

Les étapes principales dans le traitement d'un événement simple, à l'instant t_E sur une entrée **E**, et provoquant la commutation de la sortie **S** sont les suivantes (Figure 5.3) :

❶ L'état et les paramètres de l'entrée **E** sont mis à jour selon les informations contenues dans l'événement.

❷ L'instant de basculement logique de la sortie **S**, $t_{vdd/2}$, est calculé par une résolution numérique de l'équation $Q(t_{vdd/2}) = C\ V_{DD}/2$, traduisant le temps mis par le cône à charger ou décharger la moitié de la capacité C.

❸ La dérivée au point de basculement, $a = dv/dt$, est calculée à l'aide de l'Équation 4.9. En se basant sur le Principe 5.3, la forme d'onde de la sortie est une tangente hyperbolique décrite par l'Équation 4.5. Elle est totalement définie par le point $(t_{vdd/2}, V_{DD}/2)$ et la dérivée en ce point.

❹ Il reste à trouver le point t_S tel que, $V(t_S) = V_S$. V_S étant la valeur de la tension sur la sortie **S**, avant commutation. Les paramètres de l'événement sur la sortie **S**, (t_S, F_S), sont calculés à partir de $t_{vdd/2}$ et de la dérivée en ce point, $a = dv/dt$. Ceci passe par une détermination préalable de la valeur finale stationnaire, V_{fin}, que prend le signal à la fin de la commutation.

Ainsi, nous obtenons

$$F_S = \frac{(2V_{fin} + V_{DD} - 4V_T)(2V_{fin} - V_{DD})}{4|a|(V_{fin} - V_T)} \tag{5.1}$$

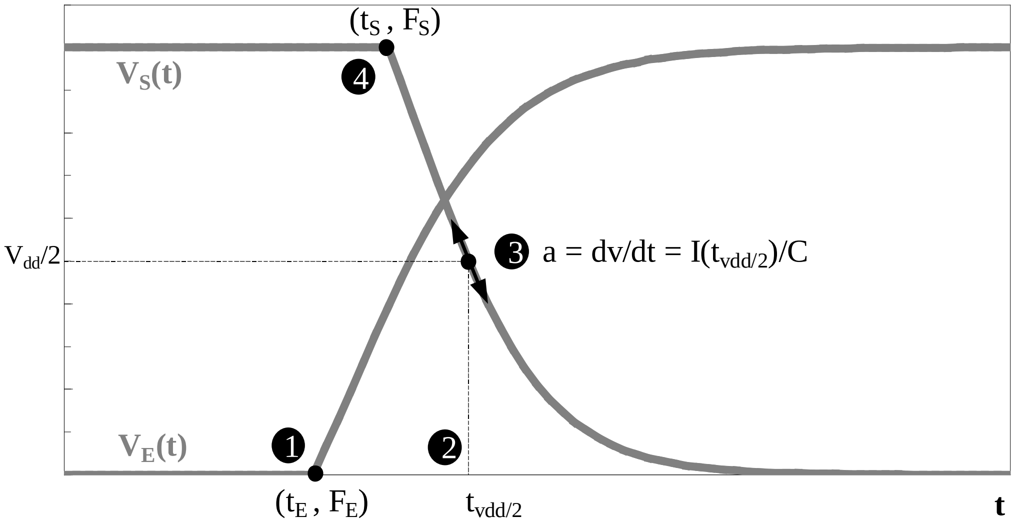
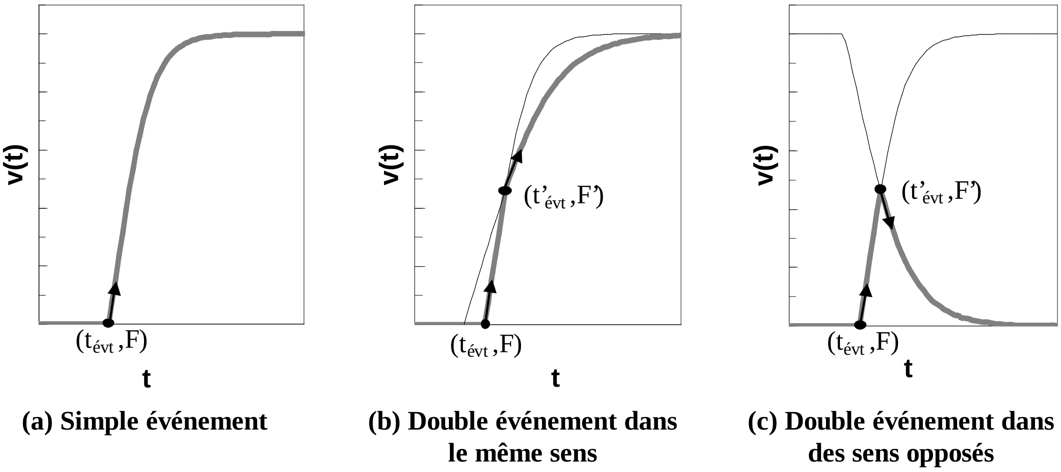


Figure 5.3 : Phases de traitement d'un événement simple à l'entrée d'un cône



(a) Simple événement

(b) Double événement dans le même sens

(c) Double événement dans des sens opposés

Figure 5.2 : Commutation et événements

d'une même porte), d'avoir en cours de commutation un changement de valeur de la constante de temps (Figure 5.2–b et c). Les aspects temporels d'une commutation à événements multiples mettent en jeu des phénomènes plus complexes et feront l'objet du chapitre suivant qui leur est entièrement dédié.

5.4. Évaluation locale

Dans cette section, nous allons aborder la méthode d'évaluation d'un événement généré en sortie d'un cône lorsqu'un événement se produit à l'entrée.

Soit un signal **S** associé à un cône **C**, dont toutes les entrées ainsi que la sortie sont dans un état stable et dont tous les courants à l'intérieur des branches sont nuls. Considérons maintenant un événement, suivant la Définition 5.2. Cet événement représente une transition sur une des entrées, de son état stable courant vers l'état stable opposé, suivant l'Équation 4.5. Nous supposerons que les autres entrées sont stables et que la sortie est sensible à l'entrée qui commute. Par la suite, nous appellerons transition simple toute transition de ce type.

L'objectif est de déterminer les paramètres de l'événement en sortie qui sera programmé dans l'échéancier, étant donné que l'on impose à cette sortie d'effectuer sa transition suivant l'Équation 4.5 (Principe 5.3).

L'approximation de la forme d'onde à la sortie d'un cône, suite à une commutation d'une de ses entrées, passe par un calcul fin du point de basculement logique de cette sortie. Ce point correspond à l'instant où le signal de sortie traverse $V_{DD}/2$. Ce choix est motivé par le fait que la grande majorité des portes CMOS ont un seuil de basculement logique proche de cette valeur.

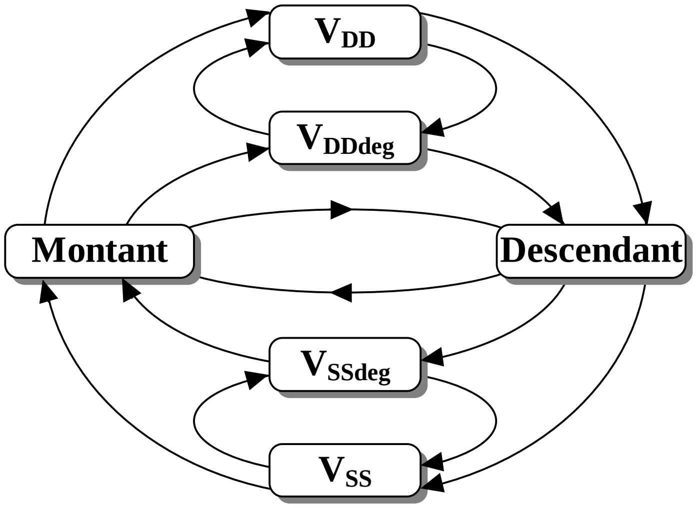


Figure 5.1 : États logiques d'un signal

signal en cours de commutation effectue une suite de transitions en visitant un ensemble, plus ou moins grand, de valeurs intermédiaires pré-définies. Chaque transition représente un événement.

Aucune de ces deux définitions ne nous semble satisfaisante pour répondre à nos objectifs. La première présente l'inconvénient de ne pas contenir assez d'information pour pouvoir effectuer une simulation précise. La seconde est beaucoup trop gourmande en temps de calcul. Nous proposons un principe de simulation et une définition de l'événement qui, tout en incluant l'effet du front pour la précision, permettent un traitement simple et rapide des commutations dans un circuit.

Principe 5.4 : Dans PÉCF, un événement intervient sur un signal **S** décrit par l'Équation 4.5, chaque fois que la constante de temps **F** de ce signal change de valeur. Quand le signal se trouve dans un état stable, la valeur de **F** est considérée comme étant égale à zéro. □

Définition 5.2 : Dans PÉCF, un événement **E** est représenté par le quadruplé ($\mathbf{t_{évt}}$, **F**, $\mathbf{V_{fin}}$, **S**). Cela traduit un changement de valeur de la constante de temps du signal **S** commençant à l'instant $\mathbf{t_{évt}}$, à partir de la valeur courante de la tension, en direction de la valeur finale $\mathbf{V_{fin}}$ et suivant la constante de temps **F**. □

Ce principe de simulation est suffisamment général pour inclure tous les cas de figures possibles. Le cas le plus simple est celui d'une commutation complète entre deux états stables, suivant un seul paramètre F, nécessitant par conséquent un seul événement (Figure 5.2–a). Outre ce cas simple, il est possible en cas de collision (plusieurs événements sur les entrées

Aussi, la difficulté principale de cette méthode passe par une définition claire de la notion d'événement et une description, non moins claire, de la technique de traitement de cet événement.

5.3.2. Les états stables

Avant de définir la notion d'événement, nous commençons par présenter l'ensemble des états logiques que peut prendre un signal dans PÉCF. Ces valeurs sont représentées par le paramètre V_{fin} de l'Équation 4.5 dans le modèle du signal que nous avons adopté.

Ce paramètre peut, en théorie, prendre n'importe quelle valeur comprise entre V_{SS} et V_{DD}. Cependant, nous allons nous limiter à quatre valeurs discrètes, qui représenteront les états stationnaires possibles d'un signal appartenant au réseau de cônes. Les deux premières valeurs que prennent généralement ces signaux sont, bien évidemment, celles des alimentations V_{SS} et V_{DD}. Toutefois, il y a des exceptions qui correspondent aux cas où un signal est connecté à l'alimentation V_{SS} (resp. V_{DD}) à travers des transistors de type P (resp. N). Dans ces cas, la valeur finale du signal est dégradée d'un seuil. Pour modéliser la dégradation, nous ajoutons à notre ensemble de valeurs stationnaires deux autres valeurs, V_{SSdeg} (pour un signal connecté à V_{SS} à travers un transistor P) et V_{DDdeg}, (pour un signal connecté à V_{DD} à travers un transistor N). Ces deux valeurs sont des paramètres technologiques, au même titre que les paramètres du courant des transistors.

Il existe bien évidemment des montages de circuits où un signal peut être dégradé de plusieurs niveaux. Ce cas de figure est susceptible de se présenter quand, dans un cône, un transistor de passage est lui même commandé par un signal dégradé. Nous avons choisi de ne pas inclure ces cas, car notre méthode de simulation vise des circuits numériques qui interdisent ce type de montages.

Définition 5.1 : Un signal S, représenté par un nœud dans le graphe de causalité G, peut être dans l'un des états suivants : V_{SS}, V_{DD}, V_{SSdeg}, V_{DDdeg}, montant ou descendant (Figure 5.1). Les quatre premiers états sont des états stationnaires. □

Quand le signal est à l'état montant ou descendant, il est possible, dans PÉCF, de calculer la valeur réelle de sa tension à partir de l'Équation 4.5.

5.3.3. Les événements dans PÉCF

Dans les simulateurs logiques, un événement est un changement instantané entre deux états logiques. Dans les simulateurs électriques qui utilisent les techniques de relaxation, un de manière complète. Autrement dit, les signaux sont en général dans un état stable (V_{DD} ou V_{SS}) et lorsqu'ils quittent cet état stable, ils effectuent une transition complète vers l'état stable opposé suivant une forme prévisible. De plus, la transition d'un signal de sortie d'un cône est généralement due à une commutation d'un des signaux d'entrée du cône.

Considérons une représentation du signal suivant l'Équation 4.5, telle que :

$$V(t) = V_T + (V_{fin} - V_T)\,\mathrm{th}\left((t - t_T)/F\right)$$

où F constitue la constante de temps représentant le front du signal, V_T est un paramètre technologique indiquant la tension de seuil du transistor et t_T est l'instant de passage de V(t) par la tension V_T. La méthode de simulation PÉCF, dans son approche générale, s'appuie sur les trois principes suivants :

Principe 5.1 : Le temps de simulation avance toujours dans le sens positif et n'effectue jamais de retour dans le temps. Autrement dit, on ne reconsidère jamais un événement déjà traité. □

Principe 5.2 : En programmant dans l'échéancier un événement sur un signal S à la date $\mathbf{t_S}$, tous les événements sur ce signal dont la date est supérieure ou égale à $\mathbf{t_S}$ sont supprimés. □

Principe 5.3 : Soit une commutation de la forme **V(t)**, avec les paramètres $\mathbf{(V_{fin}, t_T, F)_E}$, entre deux états stationnaires sur un nœud $\mathbf{n_E}$ du graphe **G**. Dans PÉCF, quand cette commutation est propagée vers un nœud $\mathbf{n_i \in Fanout(n_E)}$, elle provoque une commutation qui est également de la forme **V(t)**, avec les paramètres $\mathbf{(V_{fin}, t_T, F)_i}$. □

Les deux premiers principes sont liés à l'utilisation d'une méthode de simulation à pilotage événementiel. Le troisième principe est propre à notre méthode de simulation PÉCF. Il suggère que l'objectif d'une fonction d'évaluation est de déterminer les paramètres d'une transition en sortie, à partir des paramètres d'une transition en entrée, grâce au modèle de courant présenté au chapitre précédent. Ce dernier principe est primordial si l'on veut obtenir un système de simulation fermé, c'est à dire, un système dans lequel le modèle d'évaluation peut être appliqué au résultat de l'évaluation précédente.