THÈSE DE DOCTORAT DE L'UNIVERSITÉ PARIS VI

SPÉCIALITÉ INFORMATIQUE

Présentée par Karim DIOURY

Pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ PARIS VI

ANALYSE TEMPORELLE HIÉRARCHIQUE DES CIRCUITS VLSI À TRÈS HAUTE DENSITÉ D'INTÉGRATION

Présentée le 18 septembre 1998, devant le jury composé de

FRANÇOIS ANCEAU	Président
LUC CLAESEN	Rapporteur
PATRICE QUINTON	Rapporteur
ALAIN GREINER	Examinateur
MARIE-MINERVE LOUËRAT	Examinateur
MICHEL THILL	Examinateur

Résumé

L'analyse temporelle s'inscrit dans la phase de vérification d'un circuit VLSI numérique. Avec l'avènement des technologies fortement submicroniques, l'analyse temporelle statique s'impose comme la seule méthode possible pour la vérification temporelle de circuits pouvant atteindre aujourd'hui plusieurs dizaines de millions de transistors. Néanmoins, cette méthode génère des volumes de données trop importants.

Pour pallier ce problème, nous avons défini une méthode fondée sur le découpage hiérarchique de la phase de conception, afin de représenter les temps de propagation dans un circuit, grâce à une vue temporelle hiérarchique multi-niveaux. Les temps de propagations dus aux portes et aux réseaux RC sont représentés par un graphe de causalité dont les sommets sont les événements sur les signaux, et les arcs les temps de propagation entre deux événements sur deux signaux.

Chaque instance de l'arbre hiérarchique est représentée par une figure temporelle contenant l'information relative au graphe de causalité qui ne peut être décrite dans les figures temporelles associées aux sous-blocs instanciés par chacune de ces instances.

Une méthode de parcours du graphe de causalité a été définie afin de limiter la recherche des chemins entre les registres et les connecteurs à une petite partie du graphe.

L'approche intègre également un nouveau concept dans l'analyse temporelle, la factorisation des chemins critiques. Cela permet, par la résolution du problème de la croissance quadratique du nombre de ces chemins, d'accélérer la recherche de chemins critiques dans le graphe de causalité ainsi réduit.

La vue temporelle multi-niveaux que nous avons définie a permis de concevoir l'outil d'analyse temporelle hiérarchique ***HiTas*** ainsi que l'outil interactif de visualisation des chemins ***Xtas***. L'expérimentation de ces outils a montré que notre approche permet de traiter des circuits à très haute densité d'intégration.

Mots-clé :

Analyse temporelle

Vérification hiérarchique

Réduction de graphe

Abstract

Timing analysis is an important step in the verification of digital VLSI circuits. With the advent of deep submicronic technologies, static timing analysis has revealed itself as the only feasible method for the timing verification of circuits, which can nowadays reach up to several million transistors. Nevertheless, this method can generate an excessively large amount of data.

To alleviate this problem, we have defined a method based on the hierarchical partitioning of the design phase. The propagation times within a circuit are represented using a multi-level hierarchical timing view. Propagation times of gates and RC networks are represented by a causality graph in which the nodes correspond to events on the signals, and the arcs correspond to the propagation times between two events on two signals.

Each instance of the hierarchical tree is represented by a timing figure containing information relative to the causality graph which cannot be described in the timing figures associated with each sub-block instantiated by each of these instances.

A method for the traversal of the causality graph has been defined, which limits the search for paths between connectors and registers to a small part of the graph.

The approach also adopts a novel concept in timing analysis, that of critical path factorisation. This concept allows, thanks to its solution of the problem of quadratic growth in the number of paths, acceleration of the critical path search in a subsequently reduced causality graph. The multi-level timing view, which we have defined, has allowed us to produce the hierarchical timing analysis tool ***HiTas*** as well as the interactive path browser ***Xtas***. Experimentation with these tools has shown that our approach can deal with circuits of extremely high integration density.

Keywords:

Timing analysis

Hierarchical verification

Graph reduction

Remerciements

Cette thèse a été effectuée dans le Laboratoire ***LIP6*** de l'Université Pierre et Marie Curie sous la direction du Professeur Alain Greiner. Je le remercie vivement de m'avoir intégré à son équipe et permis de réaliser ce travail. Je lui suis également reconnaissant pour son aide à la rédaction de ce manuscrit.

Je remercie Madame Marie-Minerve Louërat pour avoir dirigé ce projet de recherche, pour les conseils qu'elle m'a donné et pour la relecture de ce manuscrit. J'ai été sensible au soutien et à la disponibilité dont elle a fait preuve pendant l'encadrement de ce travail.

Je remercie Messieurs Luc Claesen et Patrice Quinton d'avoir bien voulu juger cette thèse en acceptant d'en être rapporteurs, ainsi que Messieurs François Anceau et Michel Thill de l'intérêt qu'ils ont porté à ce travail en acceptant de l'examiner.

Je tiens à remercier la Compagnie BULL de valoriser ce travail en permettant une utilisation industrielle des outils développés dans le cadre de cette thèse et pour les échanges enrichissants que j'ai pu avoir avec Messieurs Michel Thill, Christian Masson, Alain Debreil, Bernard Marini, Jean-Claude Marcheteaux et Marcelo Duhalde.

Je remercie toute l'équipe ***ASIM*** du Laboratoire ***LIP6***. Cet environnement de travail chaleureux et enthousiaste a été appréciable au cours de ces années. Je remercie Anthony Lester et Grégoire Avot deux très bons amis pour les discutions fructueuses que j'ai pu avoir avec eux et avec qui j'ai eu grand plaisir à travailler.

Enfin, j'exprime toute ma reconnaissance à mes parents qui m'ont accompagné tout au long de mes études et à ma femme Fabienne qui m'a aidé et soutenu pendant toutes ces années de thèse.

À mes parents

Table des matières

Table des figures

Liste des tableaux

CHAPITRE 1

Introduction

L'analyse temporelle s'inscrit dans la phase de vérification des circuits VLSI. Elle consiste à s'assurer que les spécifications temporelles sont respectées, pour qu'à une fréquence donnée, un circuit numérique synchrone fonctionne sans erreur. La vérification se fait en deux étapes. La première consiste à calculer les temps de propagation, également appelés chemins, les plus longs et les plus courts entre les signaux caractéristiques du circuit, appelés dans ce mémoire « signaux de stabilité » : ce sont les connecteurs externes, les points mémoires, les commandes d'écriture dans les points mémoires et les bus préchargés d'un circuit. L'ensemble de ces chemins est sauvegardé dans un fichier qui est utilisé dans une deuxième étape, appelée analyse de stabilité, pour la vérification des règles temporelles.

Du fait de l'augmentation rapide de la complexité des circuits ces dernières années, l'analyse temporelle statique s'impose comme la seule méthode possible [Tiw94, SVG96, Nap96], contrairement à la simulation électrique ou logico-temporelle. En effet, les technologies CMOS submicroniques permettent de réaliser des circuits de plusieurs millions de transistors. La simulation ne permet plus de détecter à coup sûr les éventuelles chaînes longues ou chaînes courtes. Il faut une analyse indépendante des *stimuli*.

Les outils de synthèse et de génération des masques physiques d'un circuit s'appuient sur un découpage hiérarchique, qui facilite la conception. Inversement, la plupart des outils de vérification travaillent sur des représentations à plat du circuit. Par conséquent, la quantité de mémoire et la puissance des machines nécessaires pour ce type d'opérations sont très importantes.

L'analyse temporelle à plat au niveau transistor n'est plus possible à partir de quelques centaines de milliers de transistors. L'analyse au niveau portes logique ne permet guère de dépasser quelques millions de transistors. D'où l'idée d'utiliser le découpage hiérarchique [Pru95] de la phase de synthèse pour partitionner le travail. Certains analyseurs temporels utilisent ce partitionnement afin de traiter des circuits plus importants, mais gardent une représentation à plat des chemins [EPI97].

Avec l'avènement des technologies fortement submicroniques, la complexité des circuits va continuer à croître fortement. La fiabilité des circuits intégrés est bien supérieure à celles des systèmes à composants discrets, leur coût de production est plus faible et la diminution de la consommation pousse à l'intégration des systèmes embarqués. De plus, les progrès des outils de synthèse facilitent le travail des concepteurs. On peut aujourd'hui créer des circuits pouvant atteindre plusieurs dizaines de millions de transistors. L'analyse de stabilité grâce à un outil interactif de visualisation des chemins, devient essentielle pour de tels circuits.

Pour résoudre le problème de l'augmentation du volume des données générées par l'analyse temporelle de circuits aussi complexes, nous avons été amenés à définir une nouvelle méthode hiérarchique [DGR97]. Afin de démontrer l'intérêt de ce travail, le résultat de cette réflexion s'est traduit par le développement d'un outil logiciel. Cette thèse s'attache à présenter notre méthode et les résultats obtenus lors de son expérimentation.

1.1 L'analyseur temporel *Tas*

Les travaux sur l'analyse temporelle des circuits VLSI ont commencé au laboratoire ***MASI*** à la fin des années 80. La thèse de Amjad Hajjar [Haj92] a introduit une nouvelle modélisation des temps de propagation dans les circuits intégrés CMOS grâce au modèle de transistor à canal court. Un premier outil d'analyse temporelle ***Tas*** [Haj92] utilisant ce modèle a été développé par Amjad Hajjar et Payam Kiani. Cet outil prend en entrée une net-list de transistors et de capacités d'interconnexion. Couplé à l'outil de désassemblage ***Desb*** [Lau94, GLM92], il calcule tous les chemins les plus longs d'un circuit. ***Tas*** et ***Desb*** sont des outils de la chaîne de CAO ***ALLIANCE*** [GP92, BBG94] développée au laboratoire. Depuis plusieurs années, ***Tas*** est également utilisé avec succès dans l'industrie (compagnie BULL).

Le travail présenté dans cette thèse s'inscrit donc dans la continuité des travaux d'Amjad Hajjar et de Payam Kiani. Les améliorations méthodologiques qui y sont présentées ont été introduites dans l'outil ***HiTas***, qui est le successeur de ***Tas***.

1.2 Plan du mémoire

Le chapitre 2 présente l'analyse temporelle statique des circuits VLSI et expose les limites et les problèmes que pose l'approche à plat. Il précise les objectifs fixés en rappelant les fonctionnalités que doit offrir un outil d'analyse temporelle pour un concepteur de circuit.

Le chapitre 3 pose les définitions et les concepts de base. Il définit les notions de signal temporel, d'événement, de délai élémentaire, de chemin, de graphe de causalité et de graphe de chemins critiques.

Le chapitre 4 rappelle l'historique de l'analyse temporelle et présente les différentes approches universitaires et industrielles existantes. Il met en évidence leurs limites pour le traitement de circuits à très haute densité d'intégration. L'état de l'art permet d'apprécier les apports de notre approche dans le domaine de l'analyse temporelle.

La principale source d'augmentation non linéaire du volume des données manipulées par un outil d'analyse temporel, provient de l'explosion possible du nombre de chemins. Le chapitre 5 présente la solution que nous proposons pour réduire ce nombre et diminuer la taille des fichiers, afin d'accélérer l'exploitation et l'analyse des résultats.

Le chapitre 6 expose la représentation hiérarchique multi-niveaux ou vue temporelle d'un circuit. Celle-ci sert de base de données sur laquelle s'appuient tous les outils de la chaîne de vérification temporelle.

Le chapitre 7 décrit l'analyseur temporel hiérarchique ***HiTas*** ainsi que les étapes permettant de construire les figures temporelles à partir de figures feuilles. Il présente également l'outil interactif ***Xtas*** de visualisation des informations temporelles d'un circuit.

Afin de montrer l'intérêt de notre approche, des résultats chiffrés concernant les performances obtenues sur de gros circuits industriels sont exposés dans le chapitre 8.

CHAPITRE 2

Problématique et enjeux

Pour les circuits complexes de plusieurs millions de transistors, l'analyse temporelle statique est devenue la seule méthode pour vérifier le fonctionnement temporel d'un circuit.

Ce chapitre rappelle le rôle d'un outil de vérification formelle dans le processus de conception, ainsi que les objectifs qu'il doit remplir. Il précise les limites des approches de vérification du circuit « à plat » et expose les problèmes que pose la vérification des circuits utilisant une méthode basée sur le découpage hiérarchique de la phase de conception.

2.1 L'analyse temporelle statique

Nous nous intéressons uniquement à la vérification temporelle des circuits numériques CMOS. L'analyse temporelle intervient lors de la phase de validation finale, appelée généralement phase montante, par opposition à la phase de génération appelée phase descendante. L'objectif est de vérifier les spécifications temporelles. Il est donc nécessaire de calculer les temps de propagation à travers les portes logiques afin de vérifier que les contraintes temporelles d'écriture dans les registres sont respectées. Ces contraintes se traduisent par des règles concernant les temps de propagation les plus longs et les plus courts entre les connecteurs, les registres, les bus préchargés, les commandes d'écriture dans les registres et les bus préchargés d'un circuit.

2.1.1 L'extraction du schéma électrique

A partir des masques physiques, les outils d'extraction fournissent les schémas électriques au niveau transistors, résistances et capacités parasites. Ces schémas extraits de la description physique constituent le point d'entrée de l'analyse.

En règle générale, les portes logiques CMOS n'utilisent que des transistors. Par contre, les temps de propagation constatés dans les circuits sont non seulement dus aux capacités de diffusion et de grille et aux résistances de ces transistors, mais également aux interconnexions. Ces fils d'interconnexions mettent en jeu des effets résistifs et capacitifs parasites. Ces phénomènes sont d'autant plus importants, que la finesse de gravure augmente. Comme le précise Dai [Dai97], la part du temps de propagation due aux interconnexions peut atteindre 50 % pour une technologie $0,35\mu m$ et 70 % pour une technologie $0,25\mu m$.

Puisque la construction du circuit est hiérarchique, les net-lists peuvent aussi être extraites bloc par bloc. La net-list d'un bloc contient alors les instances d'autres blocs ainsi que les interconnexions décrites comme des réseaux de capacités et de résistances. Il existe cependant une contrainte : Le seul partitionnement possible des net-lists extraites est le découpage hiérarchique du circuit choisi par le concepteur.

2.1.2 Le calcul des temps de propagation

Le niveau d'abstraction auquel se situe l'analyse temporelle statique est le niveau porte. En effet, l'analyse statique s'appuie sur le concept de chemin, qui nécessite une « net-list » de portes orientées. Les délais des portes, appelés délais élémentaires, peuvent être calculés à partir de net-lists de transistors ou à partir de la caractérisation des cellules d'une bibliothèque.

Il existe plusieurs niveaux de modélisation du délai dans une porte CMOS. En effet, un transistor MOS possède un comportement non linéaire assez complexe. Un simulateur comme ***Spice*** [Nag75] possède plusieurs dizaines de modèles différents. Afin de réduire les temps de traitement, les outils d'analyse temporelle utilisent des modèles beaucoup plus simples. Cela peut aller d'une simple résistance à des modèles non linéaires. Mais la principale différence entre un analyseur temporel et un outil de simulation réside dans le découplage du comportement d'une porte par rapport au reste du circuit. Le calcul du temps de propagation utilise l'hypothèse que seul l'environnement immédiat d'une porte influe sur son comportement. La définition de cet environnement dépend de l'outil d'analyse. En général seuls le réseau RC vu par la sortie de la porte ainsi que le front appliqué sur l'entrée de la porte sont pris en compte.

Le niveau transistor

La net-list au niveau transistors est le niveau le plus bas de la description électrique d'un circuit. Peu d'outils sont capables de se placer à ce niveau. Cependant, cette approche à plat au niveau transistor laisse toute la latitude possible sur le choix des modèles de courant permettant de calculer les temps. De plus, les circuits performants utilisent des blocs optimisés où la net-list en portes n'est pas disponible et où il faut redescendre au niveau transistors [GDA97].

La difficulté principale de l'approche transistor provient du fait qu'il est au préalable nécessaire de retrouver le sens de propagation des signaux dans la net-list transistor. La transformation de la net-list de transistors non orientés en une net-list de portes orientées est appelée la phase de désassemblage. Il existe deux approches principales pour ce type d'outil : la reconnaissance de formes à partir d'une bibliothèques définie au préalable [Gui98] ou bien le désassemblage en cônes logiques utilisé dans ***Desb*** [GLM92, Lau94] et ***Yagle*** [Les94, LGB97] le nouveau désassembleur développé au Laboratoire ***LIP6***. La deuxième méthode est la plus générale surtout dans le cas de blocs optimisés utilisant des techniques de circuiterie distribuée (bus préchargés...).

L'analyseur temporel ***Tas*** développé au Laboratoire ***LIP6*** repose sur l'hypothèse d'une net-list au niveau transistors sans bibliothèque prédéfinie. Le modèle de transistor utilisé est le modèle ***MCC*** [HMG91, Haj92] (modèle canal court introduit par Amjad Hajjar). La phase de désassemblage est réalisée par ***Yagle***.

Les cellules précaractérisées

La plupart des outils existant utilisent des approches basées sur la caractérisation *a priori* de bibliothèques afin de s'affranchir des difficultés algorithmiques dues à la reconnaissance des portes et à la nécessité de caractériser de façon « dynamique » les portes ainsi reconstruites.

La net-list de portes qui instancie les cellules de bibliothèque est obtenue directement par extraction des masques physiques et rétro-annotation de la net-list pour introduire les capacités d'interconnexion. Aucune phase de désassemblage n'est nécessaire.

Toutes les cellules sont caractérisées, grâce à des simulateurs électriques comme ***Spice*** [Nag75, Met94] ou ***Eldo*** [Ana96]. En règle générales, le délai intrinsèque TPi_0 est obtenu avec un front $F0$ appliqué sur l'entrée i et une capacité $C0$ appliquée sur la sortie. Le délai effectif d'une porte est obtenu en additionnant à ce délai intrinsèque TPi_0, un temps dû à l'interconnexion. Ce dernier peut être calculé grâce à une résistance équivalente Ri. Une contribution due au front sur l'entrée i est déterminée à l'aide d'un troisième paramètre Si. Ceci définit donc trois paramètres pour chaque entrée i d'une porte. Le sens de commutation peut également être pris en compte. Le nombre de paramètres est alors multiplié par 2 ou 4. Le temps de propagation peut s'écrire :

$$TPi = TPi_0 + Ri \times (C - C0) + Si \times (F - F0) \tag{2.1}$$

C et F sont des valeurs de capacité et de front vues par la porte instanciée dans la net-list. On peut améliorer ce modèle linéaire en définissant plusieurs valeurs de Ri et Si pour plusieurs valeurs de $C0$ et $F0$. Ri et Si dépendent alors des valeurs de C et de F.

Néanmoins, cette approche n'est pas assez générale. Comme le précisent Grundmann et al [GDA97], elle a le défaut de ne pas pouvoir traiter les blocs optimisés que l'on rencontre souvent dans les circuits performants, à très haute densité d'intégration. De plus cette approche repose sur l'hypothèse que le délai intrinsèque constitue la contribution dominante au temps de propagation. Ceci n'est plus vrai pour les technologies submicroniques.

2.1.3 La représentation temporelle d'un circuit

Le temps de propagation dépend non seulement de l'entrée qui commute mais également du sens de commutation de l'entrée et de la sortie. Pour une entrée donnée, il existe quatre configurations possibles. Sur chaque signal, deux transitions sont possibles : passage à l'état haut ou à l'état bas. Cela peut produire à la sortie d'une porte les deux mêmes transitions. Chacune de ces transitions peut conduire à des temps de propagations très différents. Il faut donc définir un délai pour chacune des transitions entrée/sortie possibles pour une porte. Entre deux extrémités d'un chemin, les quatre configurations de transitions sont possibles. Par conséquent, un chemin n'est pas défini comme le temps de propagation entre deux signaux, mais comme le temps de propagation entre deux transitions (appelées événements dans la suite de ce manuscrit).

Le graphe de causalité

La recherche de chemins critiques (le plus long ou le plus court), peut être assimilée à l'analyse des chemins dans un graphe orienté. Un sommet de ce graphe représente une transition sur un signal. Il existe donc deux sommets pour chaque signal : un pour la transition montante et un autre pour la transition descendante. Un arc représente le temps de propagation entre deux transitions respectivement sur l'entrée et sur la sortie d'une porte. Le dédoublement en deux événements facilite les algorithmes de recherche de chemins. Ce graphe est appelé le graphe de causalité d'un circuit. La figure 2.1 montre les différentes étapes du passage de la net-list a), au graphe de chemin d), en passant par le graphe de dépendance des signaux fourni par le désassembleur b) puis par le graphe de causalité c).

Les chemins

Un chemin dans le graphe de causalité est caractérisé par un temps de propagation entre deux transitions sur des signaux particuliers ; c.-à-d. ceux qui sont assujettis à des contraintes de stabilité particulières :

- les connecteurs d'entrée et de sorties ;
- les registres : les latchs et les bascules d'un circuit ;
- les bus préchargés ;
- les commandes d'écriture dans les registres ou sur les bus préchargés.

Ces signaux sont appelés signaux de stabilité ou extrémités de chemin. Un chemin ne peut pas traverser des registres ou des signaux préchargés, qui constituent des barrières temporelles d'un circuit et donc des points d'arrêt pour les algorithmes de parcours.

La vérification des règles temporelles s'intéresse particulièrement aux chemins critiques. Ce sont les chemins les plus longs ou les plus courts entre deux transitions sur deux signaux de stabilité.

L'ensemble des chemins critiques d'un circuit peut-être assimilé à un graphe de causalité particulier où les événements sur les signaux intermédiaires ont été supprimés.

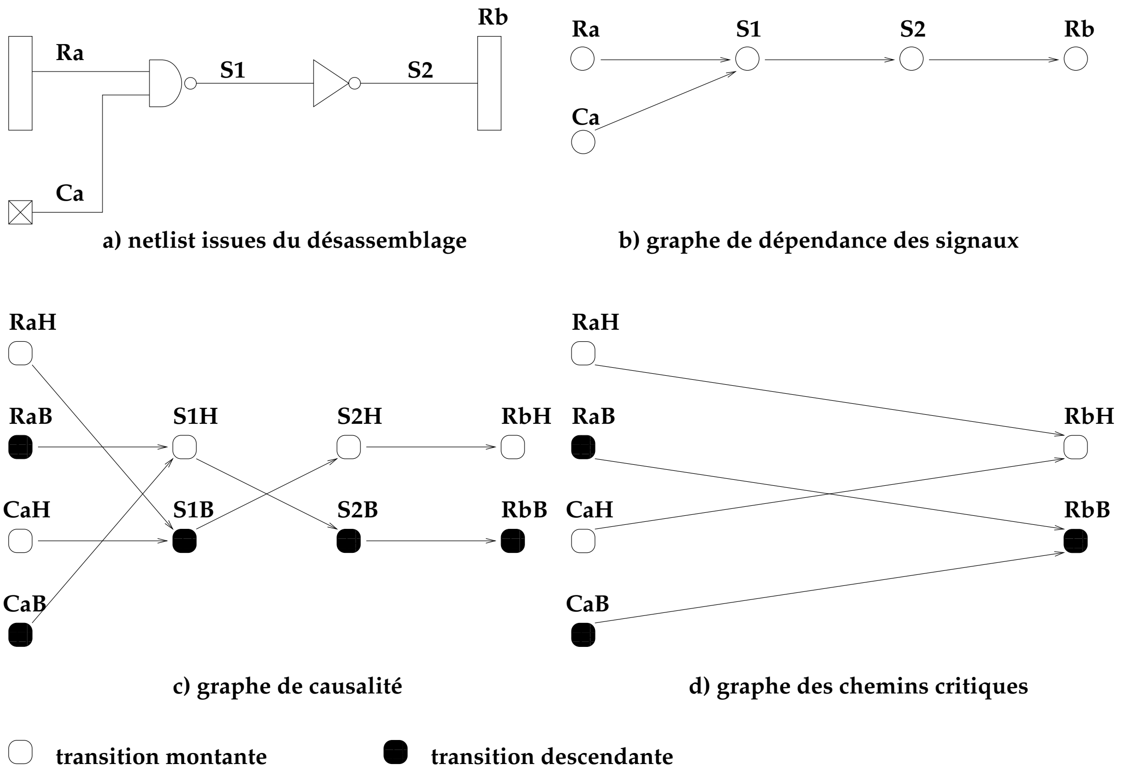


FIG. 2.1 – *La représentation temporelle d'un circuit*

La représentation du graphe de causalité et du graphe des chemins critiques

L'exploitation des résultats de l'analyse temporelle, nécessite généralement un outil d'analyse de la stabilité des signaux pour vérifier que les contraintes de prépositionnement et de maintien sont respectées. Il faut également un outil interactif de visualisation des temps de propagation permettant de comprendre par où passe un chemin critique. Ceci passe par une représentation du graphe de causalité et du graphe des chemins critiques du circuit, sous la forme de fichiers sur disque. Ces deux graphes sont nécessaires car leur utilité n'est pas la même. L'analyse de stabilité ne se sert que de la liste des chemins tandis que le graphe de causalité permet d'analyser de façon détaillée un chemin particulier.

2.1.4 Les faux chemins

Certains chemins ne peuvent être sensibilisés par aucune configuration fonctionnelle des entrées [BMC90]. Ces faux chemins devraient pouvoir être éliminés, mais ce problème est assez complexe. Il existe deux types de faux chemins : les faux chemins statiques et les faux chemins dynamiques [AKH94]. Un faux chemin dynamique ne peut jamais être sensibilisé. Un faux chemin statique peut être sensibilisé uniquement par une transition instable appelée « aléa ». La sortie retrouve son état initial à la fin de la transition. L'ensemble des faux chemins dynamiques est inclus dans l'ensemble des faux chemins statiques.

Or, seuls les faux chemins dynamiques doivent être éliminés. Les algorithmes de détection de faux chemins dynamiques sont extrêmement coûteux en temps d'exécution. Aucun outil d'analyse temporelle n'élimine ces faux chemins car le coût à payer en performance est trop important. Bien que les performances s'en trouvent sensiblement affectées, certains outils éliminent les faux chemins statiques. Par la même occasion, ils encourent le risque de passer à côté de vrais chemins dynamiques et donc de ne pas détecter certaines erreurs temporelles.

L'algorithme de parcours « en largeur d'abord » [KC66], que nous utiliserons et qui permet de retrouver les chemins les plus longs et les plus courts dans un graphe, ne permet pas de détecter les faux chemins mais garantit que les chemins critiques sont bien les plus longs ou les plus courts possibles.

Bien que nous ayons choisi dans cette thèse de ne pas aborder le problème de la détection des faux chemins, qui représente un domaine de recherche important, nous nous attacherons à préciser l'impact de l'implémentation d'algorithmes de sensibilisation dans notre méthodologie ainsi que la stratégie à adopter pour conserver des performances acceptables.

2.2 Les enjeux

2.2.1 L'analyse de stabilité

L'analyse de stabilité consiste à vérifier les contraintes de stabilité sur les entrées de tous les registres ou bus préchargés d'un circuit. Pour une fréquence de fonctionnement donnée et des caractéristiques de stabilité des connecteurs d'entrée fixées, les temps de prépositionnement et de maintien sur les registres internes ainsi que les contraintes imposées par le concepteur sur les connecteurs de sorties doivent être respectés. Cette vérification est possible grâce la connaissance de l'ensemble des chemins les plus longs et les plus courts d'un circuit.

Les algorithmes de vérification dépendent du type de registre utilisé. L'analyse de stabilité des circuits utilisant des latchs à échantillonnage sur état a été étudiée dans [Chi92]. C'est un problème de résolution de système couplé. En effet, les latchs permettent d'utiliser la phase de transparence. Les circuits à haute performance contiennent ce type de circuiterie car ils permettent d'utiliser des opérateurs combinatoires dont le temps de propagation est plus grand que la période de l'horloge.

Les algorithmes les plus couramment utilisés pour résoudre un tel système couplé sont basés sur la relaxation. Il est donc nécessaire de disposer de tous les chemins d'un circuit pour réaliser la vérification, puisque chaque chemin est consulté à tous les cycles de la relaxation. La taille de la mémoire nécessaire est donc directement proportionnelle au nombre de chemins dans le circuit. En première approximation, le nombre de chemins augmente de façon quadratique avec la complexité des circuits (mesurée en nombre de portes, ou en nombre de points mémoires). Pour des circuits complexes, le nombre de chemins dépasse la capacité mémoire des machines. Par ailleurs, le chargement dynamique des chemins à partir d'un fichier sur disque fait chuter les performances de façon dramatique. Le graphe de causalité qui décrit les délais au niveau porte a une complexité qui est elle linéaire avec le nombre de portes. On pourrait donc imaginer de recalculer dynamiquement les temps de chemins à partir du graphe de causalité. Malheureusement, la reconstitution des chemins à partir du graphe de causalité est très pénalisante en terme de temps d'exécution.

2.2.2 L'analyse interactive des résultats

La vérification automatique des règles temporelles par un outil d'analyse de la stabilité n'est pas suffisante du point de vue du concepteur. Il faut également lui donner la possibilité d'analyser les erreurs et d'étudier les possibilités de les corriger, grâce à un outil interactif lui permettant de visualiser de façon détaillée un chemin particulier.

La contribution au délai de chaque porte d'un chemin permet d'étudier les possibilités d'optimisation. Certaines portes mal dimensionnées peuvent alors être modifiées afin d'optimiser les performances du circuit.

L'optimisation du chemin le plus long ou le plus court n'apporte pas toujours la solution. Il peut exister d'autres chemins voisins dont le délai ne permet pas de respecter les spécifications. Il faut donc disposer de tous chemins possibles, ayant les mêmes extrémités que le chemin critique. Par conséquent, l'optimisation pourra se faire sur un ensemble de chemins trop longs ou trop courts. Il faut évidemment disposer du détail de ces chemins parallèles.

En résumé, le concepteur doit pouvoir disposer pour chaque chemin critique, de sa description détaillée au niveau porte ainsi que de ses chemins parallèles et de leur description détaillée. Afin d'obtenir ces informations, il faut pouvoir accéder de manière efficace à tous les délais élémentaires de toutes les portes d'un circuit et pouvoir les rechercher dans le graphe de causalité. Il est bien sûr inconcevable de conserver tous les détails de chaque chemin critique et encore moins ceux de chaque chemin parallèle.

La figure 2.2 montre les différentes étapes de la chaîne de vérification temporelle.

2.2.3 La complexité croissante des circuits

Les technologies CMOS submicroniques conjuguées aux progrès des outils de conception ont permis d'accroître de manière considérable le nombre de transistors intégrables dans un circuit.

Le tableau 2.1 [Fly97] montre l'évolution des possibilités d'intégration depuis 1992 et donne les prévisions jusqu'en 2007. En 1998 la limite d'intégration est d'environ vingt millions de transistors. Comme le montre la courbe de la figure 2.3, ce chiffre double

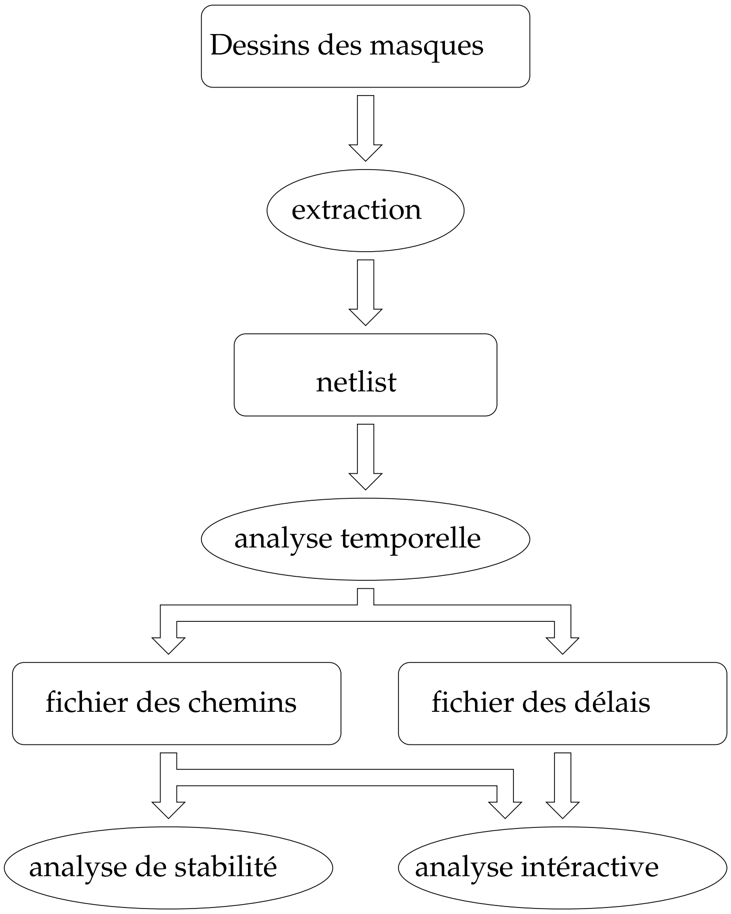


FIG. 2.2 – *Chaîne de vérification temporelle*

tous les trois ans et cette croissance devrait se poursuivre dans les années à venir. Par conséquent, les nouvelles possibilités technologiques vont favoriser l'apparition de circuits extrêmement complexes.

La fiabilité d'un circuit intégré est meilleure que celle d'une carte, son coût est moindre, sa taille et sa consommation sont réduites. L'ensemble de ces facteurs poussent donc au développement des systèmes embarqués sur une seule puce, donc très complexes.

Avec les nouvelles technologies, les fréquences de fonctionnement des circuits peuvent atteindre $500MHz$. De telles fréquences ne peuvent être obtenues qu'en interne. Les circuits imprimés ne permettent pas de véhiculer des signaux logiques à ces fréquences. Ces limitations poussent à utiliser les possibilités d'intégration offertes par les nouvelles technologies pour réaliser des systèmes sur une seule puce. L'horloge à haute fréquence est obtenue à partir d'une horloge externe à basse fréquence grâce à des systèmes de multiplication de fréquence d'horloge [CDG95, CDG96]

Ces progrès ont également permis d'augmenter la puissance des processeurs et la taille de la mémoire des ordinateurs. Mais la puissance supplémentaire ainsi obtenue n'est pas suffisante pour compenser l'accroissement du volume des données pour des

année	1992	1995	1998	2001	2004	2007
technologie (μm)	0,5	0,35	0,25	0,18	0,13	0,1
tension d'alimentation (V)	5,0	3,3	2,5	1,8	1,5	1,2
transistors par circuit (M)	5	10	20	50	110	260
surface (mm^2)	210	250	300	360	430	520
fréquence (MHz)	225	300	450	600	800	1000

TAB. 2.1 – *Prévision de l'évolution des possibilités d'intégration des circuits en technologie CMOS [Fly97]*

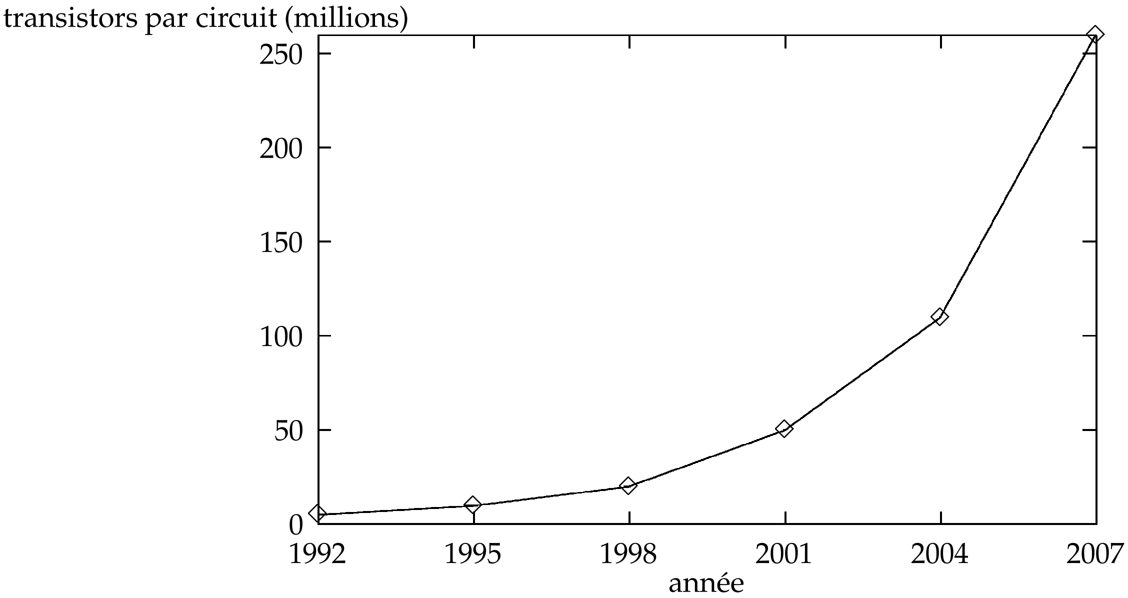


FIG. 2.3 – *Augmentation du nombre maximal de transistors par circuit entre 1992 et 2007*

algorithmes à complexité polynomiale. Le tableau 2.1 montre également qu'entre 1992 et 2007, la taille des circuit sera multipliée par 52 tandis que la fréquence ne le sera que par 4,5. Le traitement de très gros circuits impose donc de nouvelles méthodes hiérarchiques.

Les outils de conception et de génération des masques s'appuient sur des méthodes hiérarchiques, indispensables pour les très gros circuits. Une idée directrice de notre travail est d'utiliser la structure hiérarchique multi-niveaux qui a été définie dans la phase de conception pour maîtriser la complexité de la vérification temporelle.

2.2.4 L'explosion du nombre de chemins

Le calcul des délais élémentaires est local à chaque porte. La complexité de la caractérisation temporelle des portes est donc linéaire par rapport à la taille du circuit.

Les algorithmes de recherche de chemins ne sont pas nécessairement pénalisés par la complexité des circuits. En effet, le nombre de portes entre deux registres est limité car la fréquence de fonctionnement est liée à ce nombre. La complexité pratique des algorithmes de recherche de chemin est donc pratiquement linéaire par rapport à l'augmentation du nombre de registres dans un circuit.

Par contre, le nombre de chemins critiques peut croître de manière quadratique par rapport au nombre de registres. En effet, prenons l'exemple d'un banc de registres de n points mémoires qui écrivent dans un autre banc de registres de m points mémoires. Il existe potentiellement $n \times m$ chemins. Par conséquent, le graphe des chemins est la représentation la plus coûteuse en mémoire et en espace disque. Cette constatation est assez paradoxale car le graphe des chemins contient beaucoup moins d'informations que le graphe de causalité. Il n'est pas possible par exemple de retrouver le détail d'un chemin ou ses chemins parallèles alors que cette information est disponible dans le graphe de causalité. On ne peut malheureusement pas utiliser le graphe de causalité dans les outils d'analyse de stabilité, puisqu'il faudrait relancer autant de fois la recherche de tous les chemins qu'il y a de cycles de relaxation. L'explosion du nombre de chemins est ainsi l'une des principales limitations de l'analyse temporelle des circuits à très haute complexité.

2.3 L'approche hiérarchique

2.3.1 Pourquoi l'approche hiérarchique ?

La conception hiérarchique consiste à développer des opérateurs qui serviront de composants pour un opérateur plus complexe. De plus, un modèle ayant été défini, il est possible de l'instancier plusieurs fois. Cette méthodologie permet la réutilisabilité des composants. Mais ce n'est pas le seul avantage de la conception hiérarchique : elle permet également de diminuer la complexité des algorithmes de synthèse ou de placement/routage. Dans le cas de grands circuits, le découpage hiérarchique sert également à paralléliser la conception entre les différentes équipes de conception.

2.3.2 Les problèmes que pose la vérification hiérarchique

Les dépendances entre blocs

Le processus de décomposition hiérarchique respecte une approche descendante. En revanche, les outils de construction d'un circuit (synthèse, placement routage...) ont une approche ascendante : ils considèrent les instances comme des boîtes noires, car l'interface des blocs instanciés à un niveau, suffit à traiter ce niveau. Il suffit d'appliquer le traitement récursivement jusqu'au dernier niveau pour réaliser le circuit.

Il n'en est pas de même pour les outils de vérification (simulation, vérification de règles de dessin, calcul de consommation, analyse temporelle...), pour lesquels il n'est plus possible de considérer une instance comme une boîte noire, car les interactions entre instances ne sont pas limitées aux interfaces des blocs.

La vérification hiérarchique est cependant possible dans certains cas : Gilles-Eric Descamps [Des96] a montré que, dans le cadre de la vérification des règles de dessin, il est possible de s'appuyer sur la hiérarchie de conception pour réduire les dépendances entre instances à une couronne autour de chaque instance. Cette méthode a permis de réaliser un outil capable de paralléliser la vérification sur un réseau de stations de travail grâce à la bibliothèque PVM.

Dans le cadre de l'analyse temporelle, on pourrait envisager d'utiliser le découpage hiérarchique afin de générer les chemins d'un bloc à partir de ceux de ses instances. De la même manière que les cellules d'une bibliothèque, le modèle temporel du bloc ainsi créé serait alors décrit comme une cellule caractérisée par ses chemins critiques. Malheureusement, nous verrons que cette approche simple qui permet de considérer chaque instance comme une boîte noire, caractérisée par ses chemins ne répond pas à nos objectifs.

La recherche des informations détaillées sur un chemin

La liste des chemins critiques d'un circuit est suffisante pour l'analyse de stabilité. Mais elle est insuffisante pour un outil interactif de visualisation du détail de ces chemins au niveau portes ou pour la recherche de chemin parallèle. Ces informations ne peuvent être obtenues qu'à partir du graphe de causalité, c.-à-d. des délais élémentaires des portes. Dans ce cas, nous ne sommes plus dans la configuration où les instances sont des boîtes noires. Dans le circuit décrit hiérarchiquement de la figure 2.4, la question posée est :

« Comment retrouver le détail d'un chemin entre les registres Rc et Rd situés dans deux instances différentes $I11$ et $I12$? »

Le chemin peut éventuellement passer par l'instance $I22$. Il faut donc répondre à la question suivante :

« Comment éviter de construire le graphe de causalité à plat du circuit complet pour retrouver le détail d'un seul chemin ? »

La prise en compte des résistances d'interconnexion

Les technologies atteignent en 1998 des finesses de gravure inférieures à $0,25\mu m$. Par conséquent, les interconnexions en métal ne peuvent plus être assimilées à des équipotentielles traitées comme de pures capacités dans les modèles de calcul de temps de propagation. Dans les circuits à très haute densité d'intégration, la longueur d'une interconnexion peut dépasser $1cm$. La résistance de ces longs fils de métal dépasse celles des transistors. Les interconnexions doivent être considérées alors comme des réseaux de résistances et de capacités.

Dans la littérature, il existe de nombreux modèles d'interconnexion. Par exemple celui d'Elmore [Elm48, JR81], permet de calculer les temps de propagation de manière hiérarchique et incrémentale [RPH83]. Bien que ce modèle ait une précision limitée, elle peut être suffisante pour la majorité des signaux d'un circuit. Mais pour certains signaux contenant des réseaux complexes (distribution d'horloge, réseaux maillés ...), il n'est pas applicable. Il est nécessaire d'utiliser des modèles plus complexes tels que

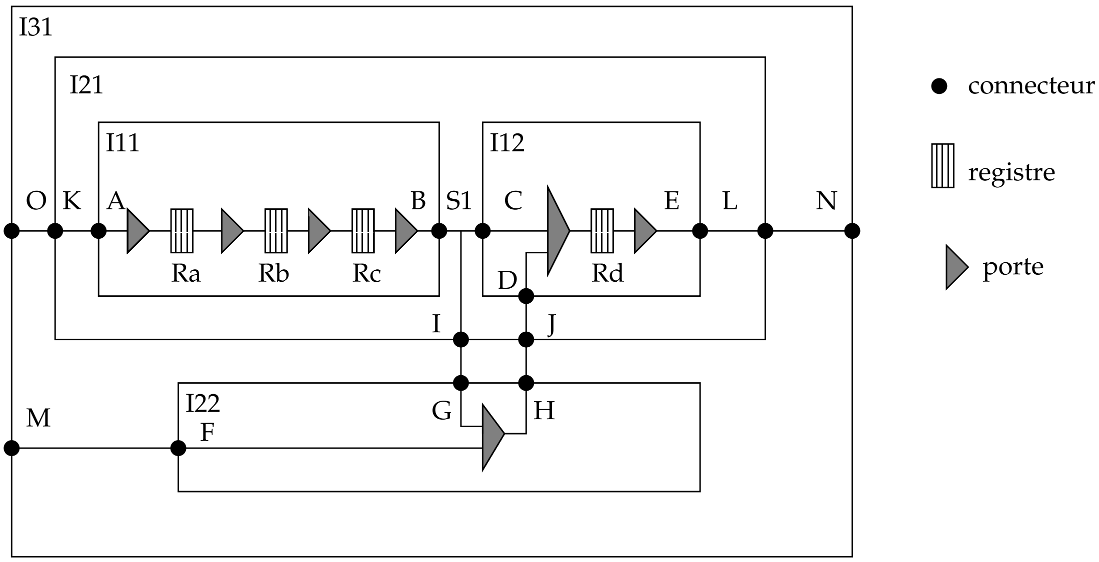


FIG. 2.4 – *Description hiérarchique d'un circuit*

« Asymptotic Waveform Evaluation » [PR90] voire même la simulation électrique de type ***Spice***. Les temps de propagation dans un tel réseau RC n'étant pas additifs, il n'est pas possible de découpler la contribution d'une partie du réseau décrite dans un bloc par rapport à la contribution d'une autre partie décrite dans un autre bloc. Il faut disposer de la totalité du réseau, ce qui n'est pas compatible avec une approche hiérarchique.

Dans cette thèse, nous ne nous intéresserons pas aux modèles de calcul des délais dans les réseaux d'interconnexion RC complexes, nous nous attacherons à définir une méthode hiérarchique multi-niveaux qui permet la prise en compte de ces réseaux RC. De plus, nous ferons l'hypothèse que les résistances d'interconnexion sont négligeables à l'intérieur des blocs de niveau hiérarchique le plus bas et que seules seront prises en compte les résistances des fils de routage entre blocs. La prise en compte des résistances entre les portes et internes aux portes ne sera pas abordée dans cette thèse. Nous nous intéresserons ici aux problèmes méthodologiques que pose la prise en compte des résistances d'interconnexion dans une approche hiérarchique.

2.4 Conclusion

Nous avons présenté dans ce chapitre l'analyse temporelle statique des circuits intégrés. Bien qu'elle reste la solution la plus satisfaisante pour la vérification formelle du fonctionnement temporel d'un circuit, nous avons mis en évidence les limites des méthodes existantes. L'augmentation de la puissance des machines ne permet pas de compenser l'explosion des volumes de données à traiter. Par conséquent, les méthodologies actuelles ne sont plus adaptées au traitement de circuits de plusieurs dizaines de millions de transistors, rendus possible par les technologies fortement submicroniques.

La seule approche envisageable passe par une méthode qui utilise le découpage hiérarchique de la phase de conception afin de maîtriser le volume de données à traiter. Il convient donc de définir une approche hiérarchique qui, dans le cas de l'analyse temporelle, est complexe sachant qu'il n'est pas possible de considérer les blocs instanciés comme des boites noires.

CHAPITRE 3

Concepts et algorithmes de base de l'analyse temporelle

Dans le chapitre 2, nous avons présenté les notions de graphe de causalité et de graphe des chemins critiques. Ce chapitre s'attache à définir précisément toutes les notions fondamentales qui permettent de construire ces deux représentations temporelles d'un circuit. Il décrit également les deux algorithmes : largeur d'abord et profondeur d'abord, appliqués à la recherche de chemin.

3.1 Les signaux temporels

3.1.1 Définition des signaux temporels

Nous cherchons à calculer des temps de propagation entre des points de référence dans un circuit. Dans le cas de l'analyse temporelle, les transitions sur les nœuds internes aux portes logiques ou sur les nœuds internes aux réseaux RC qui modélisent les interconnexions, ne présentent pas d'intérêt particulier pour la description d'un temps de propagation sur un chemin. Les points intéressants sont ceux dont les transitions entre l'état logique haut et l'état logique bas permettent de définir les temps de propagation significatifs. Nous les appelons signaux temporels.

Dans le cas où les résistances d'interconnexion sont négligeables, un fil d'interconnexion est une équipotentielle et les seuls temps de propagation à considérer sont ceux dus à la traversée des portes. Dans ce cas, le signal temporel défini à la sortie d'une porte est également celui de l'entrée de la porte suivante. C'est cette hypothèse qui est utilisée dans notre méthodologie pour le traitement à plat des blocs de plus bas niveau de la hiérarchie. Les signaux temporels sont les sorties des portes logiques.

Dans le cas où les résistances des fils d'interconnexion entre blocs doivent être prises en compte, un fil d'interconnexion n'est plus équipotentiel, il n'est donc plus possible de le représenter par un seul signal temporel. Ces interconnexions sont décrites avec des éléments RC. On considère le circuit comme une interconnexion d'instances reliées par des dipôles. Les signaux temporels sont les extrémités de ces éléments RC connectés aux entrées et aux sorties des portes. Les extrémités des dipôles sont les signaux temporels. Pour une interconnexion, il existe autant de dipôles que de couples entrée/sortie qu'elle permet de connecter. Chaque dipôle est constitué de tous les éléments RC de l'interconnexion.

La figure 3.1 schématise les deux cas : Si les résistances sont négligeables, les interconnexions A, B et C sont les signaux temporels, sinon les signaux temporels sont les nœuds des réseaux RC connectés aux entrées et aux sorties des portes.

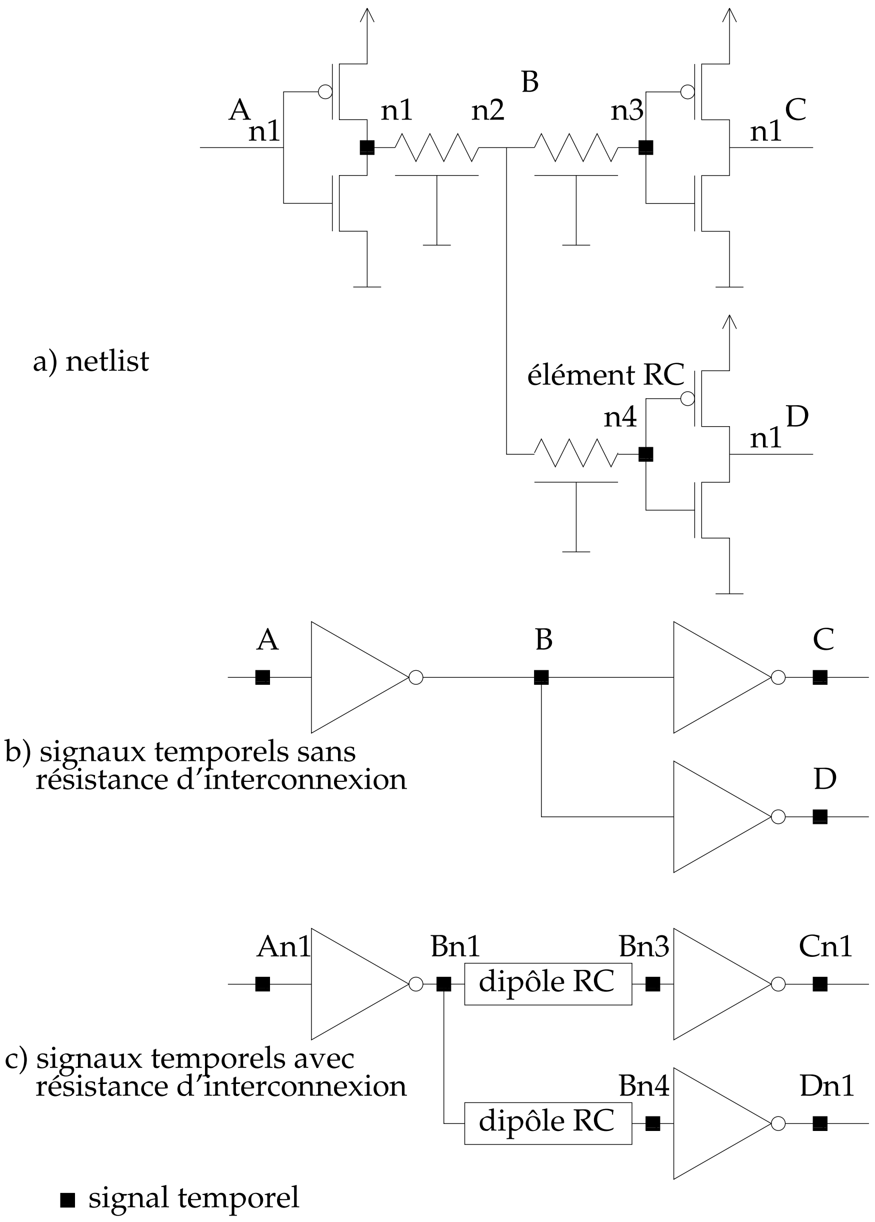


FIG. 3.1 – *Définition des signaux temporels*

3.1.2 Noms des signaux temporels

Un signal temporel est caractérisé par son nom. Ce nom est unique. Dans le cas d'une figure hiérarchique, le nom d'un signal est un « cheminom » construit à partir de son nom dans le plus bas niveau hiérarchique, préfixé par les noms des instances auxquelles il appartient.

Par exemple, dans une figure *grand-père*, où *S* est un signal de *fils*, *fils* une instance de *père* et *père* une instance de *grand-père*, le signal *S* a le nom : *père.fils.S*

3.1.3 Les signaux temporels de stabilité

Parmi l'ensemble des signaux temporels, les signaux temporels de stabilité sont ceux pour lesquels existent des contraintes de stabilité (temps de prépositionnement et de maintien). La vérification du fonctionnement temporel d'un circuit synchrone passe par la détermination des instants de stabilisation de ces signaux dans un cycle d'horloge. Il existe quatre types de signaux de stabilité : les connecteurs externes, les registres, les bus préchargés, les commandes d'écriture dans les registres et les bus préchargés.

Les connecteurs

Les connecteurs sont les signaux d'interface qui permettent à un bloc de communiquer avec les composants auxquels il est connecté. Ils sont soumis à des contraintes externes au circuit qui dépendent du type de connecteur : un temps de prépositionnement et un temps de maintien pour chaque connecteur d'entrée, un temps d'accès pour chaque connecteur de sortie. Tous ces temps sont définis par rapport à un ou plusieurs signaux d'horloge.

Il existe aussi des connecteurs bidirectionnels qui jouent à la fois le rôle d'entrée et de sortie. Il doivent donc respecter les deux types de contraintes.

Les registres

La notion de registre regroupe ici tous les dispositifs de mémorisation de données dans un circuit. Cela inclut les bascules à écriture sur front et les latchs à écriture sur état. La contrainte est de respecter les temps de prépositionnement et de maintien de la donnée par rapport aux instants de référence définis par le signal qui contrôle l'écriture dans le registre.

Les bus préchargés

De la même manière que pour les registres, les signaux qui interviennent dans l'évaluation d'un bus préchargé doivent respecter des règles de prépositionnement et de maintien.

Les commandes d'écriture dans les registres ou les bus préchargés

L'écriture d'une donnée dans un registre ou l'évaluation d'un bus préchargé se fait par l'intermédiaire d'un signal de commande, qui est généralement un signal d'horloge ou un signal construit à partir d'un signal d'horloge. Les transitions sur les signaux temporels de commande définissent les instants d'écriture dans les registres ou d'évaluation des bus préchargés et sont donc les instants de référence pour la vérification des contraintes de stabilité.

Pour chaque signal de stabilité, les règles temporelles sont définies plus précisément dans la thèse de [Chi92]. Dans ce qui suit, le terme « signal » sera utilisé pour désigner un signal temporel. De même, nous utiliserons les termes de connecteurs, registres, commandes et bus préchargés pour les signaux de stabilité.

3.2 Les événements

Un temps de propagation est défini non pas entre deux signaux mais entre deux événements sur deux signaux. Un événement est une transition d'un signal d'une valeur logique à l'autre. Il existe donc deux types d'événements : le passage de la valeur basse à la valeur haute ainsi que le passage inverse.

Un événement de stabilité est un événement sur un signal de stabilité.

Pour un signal S, nous désignerons par $S(U)$ l'événement correspondant au passage de la valeur basse à la valeur haute et par $S(D)$ l'événement inverse.

3.3 Les délais élémentaires

3.3.1 Définition d'un délai élémentaire

Un délai élémentaire est le temps de propagation entre deux événements sur deux signaux adjacents dans le graphe de causalité. Dans le cas d'une porte, il s'agit du délai entre un événement sur une entrée et l'événement résultant sur la sortie de la porte. Dans le cas d'un réseau RC, c'est le délai entre deux événements sur les extrémités des dipôles définis précédemment.

Un délai élémentaire peut être de type maximal ou minimal. En effet, dans le cadre d'une analyse indépendante des *stimuli*, le calcul des temps de propagation utilise des hypothèses sur les états des autres entrées. Ces hypothèses permettent de définir une borne supérieure et une borne inférieure du temps de propagation.

Par conséquent, entre deux signaux, il peut exister huit délais élémentaires : les quatre combinaisons entre les deux couples d'événements de chaque signal et pour chacune de ces combinaisons un délai maximal et un délai minimal sont calculés.

3.3.2 Caractéristiques d'un délai élémentaire

Un délai élémentaire est caractérisé par :

- sa valeur ;
- son type : maximal ou minimal ;
- son événement initial ;
- son événement terminal ;
- son événement commande dans le cas des registres et des bus préchargés.

Dans le cas où l'événement terminal concerne un registre ou un bus préchargé, il est nécessaire de préciser quel événement du signal de commande correspondant permet l'écriture dans ce registre ou ce bus préchargé. Pour un latch à écriture sur état, nous avons pris comme convention de choisir l'événement commande qui rend le latch transparent. Pour un bus préchargé, c'est l'événement qui autorise l'évaluation du bus. Pour une bascule, c'est l'événement qui correspond à la transition sur laquelle la donnée est écrite. Ces informations sont indispensables à la vérification ultérieure des règles temporelles. Dans le cas où l'événement terminal n'est pas sur un registre ou un bus préchargé, le délai élémentaire n'a pas d'événement commande.

Le latch L de la figure 3.2 possède deux commandes : $CK1$ et $CK2$. L'écriture dans le latch L d'une donnée arrivant par le signal A est écrite sur l'événement montant de $CK1$ alors qu'une donnée arrivant par le signal B est écrite sur l'événement montant de $CK2$.

3.4 Le graphe de causalité

La recherche des temps de propagation entre les signaux de stabilité nécessite la représentation des dépendances entre les différents événements. Ces dépendances sont modélisées sous la forme d'un graphe, appelé graphe de causalité qui définit les relations directes de cause à effet entre les événements.

3.4.1 Notations

Les éléments de la théorie des graphes présentés ci-dessous nous permettent de formaliser les concepts développés dans cette thèse. Ils sont tirés de l'ouvrage : Graphes et Algorithmes [GM79].

Notation

Un graphe $G = [X, U]$ est déterminé par l'ensemble de ses sommets X et l'ensemble de ses arcs U.

N est le nombre de sommets dans le graphe. Ils sont numérotés de $i = 1, .., N$.

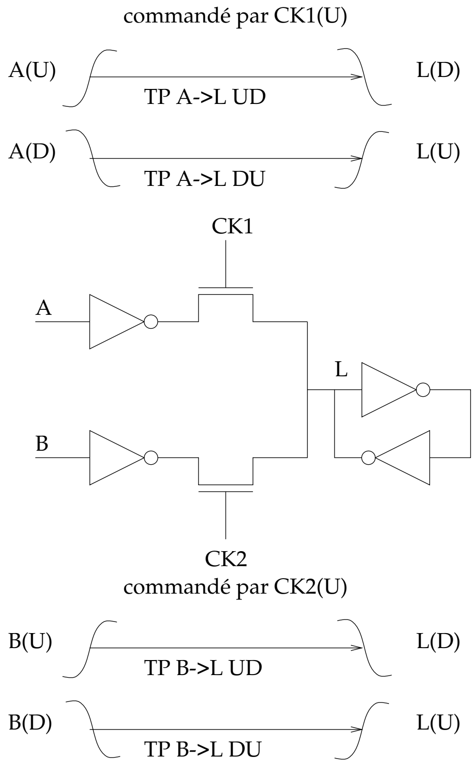


FIG. 3.2 – *Evénement commande d'un délai élémentaire*

M est le nombre d'arcs dans le graphe. Ils sont numérotés de $n = 1, .., M$. Un arc u_n est orienté; $u_n = (i, j)$ désigne un arc ayant comme sommet initial i et comme sommet terminal j.

Si i, j sont deux sommets du graphe et si $u = (i, j)$ est un arc de ce graphe alors i est un prédécesseur de j et j un successeur de i.

Chemin et cycle dans un graphe

Un chemin est un ensemble $P = \{u_1, u_2, ..., u_q\}$ de q arcs tel que chaque arc u_r, où $2 \leq r \leq q-1$, ait une extrémité commune avec l'arc $u_{r-1}(u_{r-1} \neq u_r)$ et l'autre extrémité commune avec l'arc $u_{r+1}(u_{r+1} \neq u_r)$ et tel que chaque arc soit orienté dans le même sens :

$u_1 = (i_0, i_1)$
$u_2 = (i_1, i_2)$
$u_q = (i_{q-1}, i_q)$

i_0 est l'extrémité initiale du chemin et i_q l'extrémité terminale de ce même chemin.

Un cycle du graphe est un chemin où les extrémités initiale et terminale sont confondues.

3.4.2 Définition du graphe de causalité

Les sommets du graphe de causalité représentent les événements sur les signaux. Les arcs du graphe de causalité représentent une relation de causalité directe entre deux événements. Deux sommets sont reliés par un arc si le premier sommet est un événement sur l'entrée d'une porte et le second sur la sortie de cette porte. Il existe également des arcs de type RC entre deux événements aux bornes d'un réseau RC connecté à la sortie d'une porte et à une entrée d'une autre porte.

Un arc est caractérisé par :

– son type : propagation à travers une porte ou un réseau RC ;
– son délai élémentaire minimal ;
– son délai élémentaire maximal ;
– son sommet initial ;
– son sommet terminal ;
– son sommet commande .

Le graphe de causalité est donc un graphe orienté. Il existe deux sens de parcours possibles : le sens direct, appelé causal, où chaque arc est parcouru de son événement initial vers son événement terminal ; le sens inverse, appelé anticausal, où chaque arc est parcouru de son événement terminal vers son événement initial.

La causalité d'un arc de type porte est défini de l'entrée vers la sortie de la porte. La causalité d'un arc de type RC est induite par les portes auxquelles il est connecté. Il existe cependant des cas (figure 3.3) où il existe deux orientations pour le dipôle. On crée dans ce cas deux arcs symétriques. Nous verrons dans le chapitre traitant de la hiérarchie que le cas des arcs symétriques ne peut se rencontrer que si l'on tient compte des résistances d'interconnexion entre deux connecteurs bidirectionnels.

Pour le délai élémentaire maximal et le délai élémentaire minimal entre deux événements, l'événement de commande est identique. En effet, à partir d'une transition sur une entrée d'un registre ou un bus préchargé, il n'existe en général qu'une seule commande possible pour écrire une transition sur le registre ou le bus préchargé. S'il existe, un seul sommet de commande est donc associé à un arc.

Définition 1

Deux arcs sont consécutifs s'il existe un et seulement un sommet qui soit terminal d'un des deux arcs et qui soit initial de l'autre et si au plus un arc est du type propagation à travers un réseau RC.

Les deux arcs $u_1 = (i_0, i_1)$ et $u_2 = (i_1, i_2)$ sont consécutifs si au plus un seul des deux modélise un réseau RC.

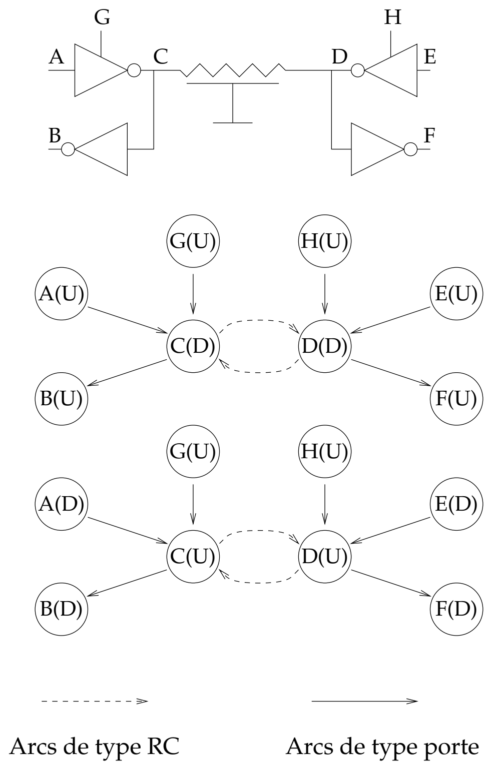


FIG. 3.3 – *Cas des arcs symétriques*

Définition 2

Deux arcs $u = (i, j)$ et $v = (k, l)$ sont symétriques si et seulement si $i = l$ et $j = k$. Deux arcs symétriques ne sont pas consécutifs.

3.4.3 Propriétés du graphe de causalité

Propriété 1

Le graphe de causalité est un 1-graphe, c.-à-d. qu'entre deux sommets $S1$ et $S2$, il existe au plus un arc orienté de $S1$ vers $S2$.

Cette propriété autorise la notation u_{ij} qui détermine l'arc entre le sommet i et le sommet j.

Propriété 2

Deux arcs symétriques sont forcément du type : propagation à travers un réseau RC.

Propriété 3

Il n'existe pas dans le graphe de causalité, de cycles composés d'arcs consécutifs et ne passant pas par un registre.

Cette propriété doit être vérifiée lors de la construction du graphe de causalité. Elle assure qu'il n'existe pas de boucles combinatoires. S'il existe des cycles combinatoires dans le graphe de causalité, ils doivent être éliminés par le concepteur.

Dans le cas des points de mémorisation construits avec des portes rebouclées, les portes de rebouclage sont marquées dans la phase de désassemblage et ne sont pas considérées comme fonctionnelles.

3.5 Les chemins temporels

3.5.1 Définition d'un chemin temporel

Un chemin temporel est un chemin du graphe de causalité tel que :

– L'extrémité initiale est un événement sur un signal de stabilité : soit un registre, soit un bus préchargé, soit un connecteur d'entrée ou un connecteur bidirectionnel ;
– L'extrémité terminale est un événement sur un signal qui est soit un registre, soit une commande, soit un bus préchargé, soit un connecteur de sortie ou un connecteur bidirectionnel ;
– Les paires d'arcs qui ont une extrémité commune sont consécutifs (définition 2). Cela signifie qu'un chemin ne peut pas contenir deux arcs successifs de type RC.

Nous appellerons $I \subset X$, l'ensemble des extrémités initiales et $T \subset X$ l'ensemble des extrémités terminales.

Nous ne nous intéresserons dorénavant qu'aux chemins temporels. Dans la suite, lorsque nous parlerons de chemin, ce sera pour désigner un chemin temporel.

Un chemin est caractérisé par :

– son type : maximal ou minimal ;
– son temps de propagation ;
– son événement extrémité initiale ;
– son événement extrémité terminale ;
– son événement commande ;
– sa liste de délais élémentaires.

Si un chemin est de type minimal (respectivement maximal), un temps de propagation minimal (respectivement maximal) lui est associé qui est la somme de tous les délais élémentaires minimaux (respectivement maximaux) de tous les arcs du chemin.

La commande associée au chemin est la commande associée au dernier délai élémentaire du chemin lorsque le dernier événement est sur un registre ou un bus préchargé. Sinon aucune commande n'est associée à ce chemin.

Tous les délais élémentaires associés à un chemin doivent avoir le même type (maximal ou minimal) que le chemin.

La figure 3.4 illustre les notions de délai élémentaire et de chemins entre les signaux temporels.

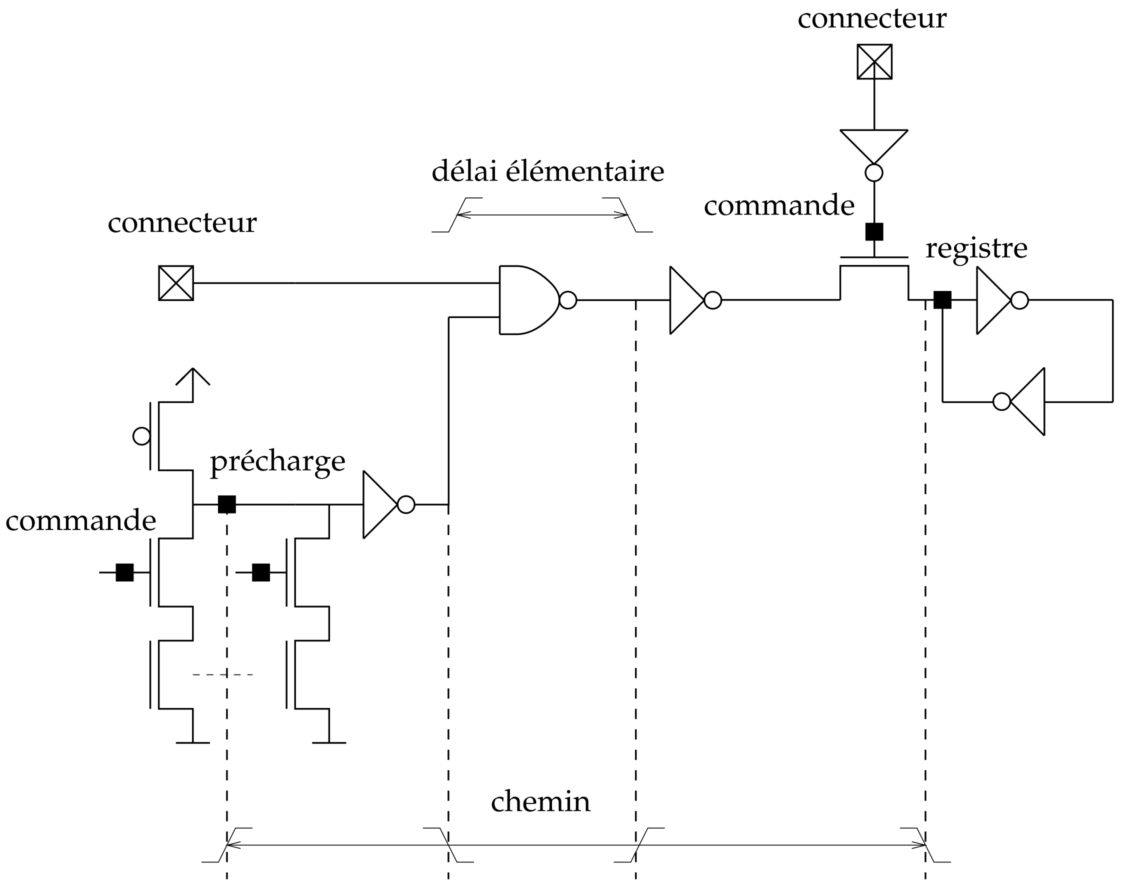


FIG. 3.4 – *Les chemins temporels*

3.5.2 Les chemins critiques

Un chemin minimal (respectivement maximal) est critique s'il n'existe pas de chemin avec la même extrémité initiale et la même extrémité terminale dont le temps de propagation est strictement plus petit (respectivement strictement plus grand).

Entre deux événements, il peut exister plusieurs chemins critiques qui ont les mêmes extrémités. Leur temps de propagation est identique, mais ils diffèrent par au moins deux délais élémentaires.

3.6 Le graphe des chemins critiques

3.6.1 Définition du graphe des chemins critiques

Pour l'analyse de stabilité, les informations intéressantes sont les temps de propagation les plus longs et les plus courts entre les événements de stabilité. Ces informations sont décrites sous la forme d'un graphe : le graphe des chemins critiques.

Les sommets du graphe des chemins critiques représentent les événements de stabilité. Il existe un arc entre deux sommets du graphe si et seulement si il existe au moins un chemin entre ces deux sommets dans le graphe de causalité.

Un arc est caractérisé par :

- son délai minimal ;
- son délai maximal ;
- son sommet initial ;
- son sommet terminal ;
- son sommet de commande.

3.6.2 Propriété du graphe des chemins critiques

Propriété 1

Le graphe des chemins critiques n'est pas un 1-graphe : il peut exister plusieurs arcs entre deux sommets : Ce cas peut se produire quand le sommet terminal de l'arc est soit un registre soit un bus préchargé et qu'il est contrôlé par plusieurs commandes.

Propriété 2

Le sommet initial et le sommet terminal d'un arc sont confondus si et seulement si ce sommet représente un événement sur un registre. Cette propriété découle de la propriété 3 du graphe de causalité.

3.7 La recherche des chemins

On s'intéresse à la recherche des chemins dans le graphe de causalité. La recherche d'un chemin dans un graphe est un problème connu. Il existe deux algorithmes généraux adaptés à notre problème : le parcours en largeur d'abord et le parcours en profondeur d'abord. Dans les sections suivantes, nous présentons l'adaptation de ces algorithmes à la recherche de chemins temporels dans le graphe de causalité.

Pour chaque algorithme, il faut préciser le type de chemin (maximal ou minimal) et le sens de parcours des arcs (causal ou anticausal) souhaité. Le choix du sens de parcours n'influe en aucun cas sur le résultat de la recherche, mais il influe sur le temps de calcul.

Les algorithmes sont décrits avec un sens de parcours anticausal, mais peuvent aisément être transposés au sens causal.

Deux variables sont associées à chaque sommet : un pointeur sur l'arc précédent qui a permis d'arriver sur ce sommet et le temps de propagation entre l'origine du parcours et le sommet considéré. Grâce à ces deux attributs, il est possible de reconstituer l'ensemble des délais élémentaires participant à un chemin. Pour un sommet S, on note $\pi(S)$ le temps de propagation entre le sommet origine de la recherche et le sommet S et $\omega(S)$ l'arc qui a permis d'arriver sur S (voir figure 3.6).

3.7.1 Le parcours en largeur d'abord

L'objectif

Le parcours en largeur d'abord permet de retrouver, pour un sommet de stabilité t du graphe de causalité, un chemin critique entre t et chacun des sommets de stabilité pour lesquels il existe au moins un chemin vers t..

L'algorithme est décrit dans le cas de la recherche de chemin critique maximal. La transposition à la recherche de chemin critique minimal est très simple.

La queue de traitement

La queue de traitement permet de définir un ensemble de sommets à traiter dans un ordre particulier.

La gestion de la queue est FIFO et réalisée par l'intermédiaire de deux fonctions :

– insérer un sommet ;

– enlever un sommet : le sommet enlevé est le plus ancien inséré dans la queue.

L'algorithme

Soit un graphe de causalité $G = \{X, U\}$.

Soit $I \subset X$ l'ensemble des extrémités initiales et $T \subset X$ l'ensemble des extrémités terminales.

Soit $C \subset X$ l'ensemble des sommets représentant des événements sur des connecteurs.

Les connecteurs sont des extrémités de chemins mais ne sont pas des sommets d'arrêt de la recherche, puisqu'un chemin peut passer à travers un événement sur un connecteur. Dans la figure 3.5 le chemin entre RI et RT passe par le connecteur de sortie C.

Soit $l(u)$, le délai élémentaire correspondant au type de recherche maximal (ou minimal) de l'arc $u \in U$.

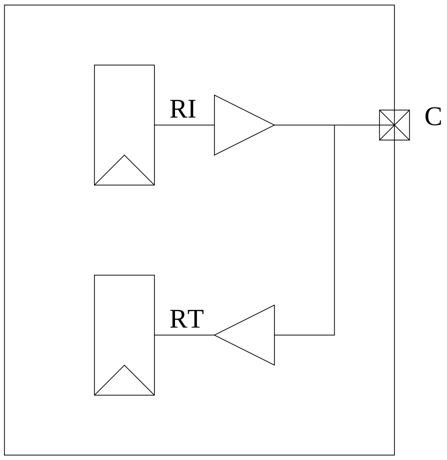


FIG. 3.5 – *Chemin à travers un connecteur*

La recherche part d'un sommet terminal de chemin $t \in T$.

(a) - initialisation

$\forall i \in X - \{t\} \quad \pi(i) = -\infty$ **et** $\pi(t) = 0$

$\forall i \in X \quad \omega(i) = \emptyset$

insérer t dans la queue

(b) - recherche des chemins

tant qu'il y a des sommets dans la queue

enlever un sommet i

si $i \in I \cap \bar{C}$ **et** $i \neq t$ **alors**

retour à **tant que**

sinon

pour tous les prédécesseurs j de i

si $(\pi(j) < \pi(i) + l(u_{ji}))$ **et**

$(((\omega(i)$ **et** u_{ji} sont consécutifs) **ou** $\omega(i) = \emptyset))$ **alors**

insérer j

$\pi(j) = \pi(i) + l(u_{ji})$

$\omega(j) = u_{ji}$

fin si

fin pour

fin si

fin tant que

Récupération des chemins critiques

Pour tous les sommets $i \in I$ avec $\pi(i) \neq -\infty$ il existe un chemin critique C_{it} entre les sommets i et t.

Le temps de propagation du chemin critique C_{it} est $\pi(i)$.

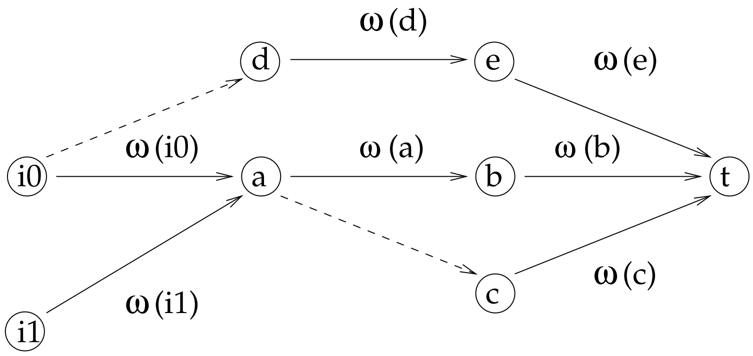
L'ensemble des sommets $i \in X$ avec $\pi(i) \neq -\infty$ et l'ensemble des arcs $\omega(i)$ forment un arbre avec t comme sommet racine.

La figure 3.6 montre l'état des $\omega(i)$ pour un graphe de causalité possédant trois sommets de stabilité : $i0$, $i1$ et t.

Pour retrouver l'ensemble des délais élémentaires du chemin C_{i0t}, il faut parcourir l'arbre entre $i0$ et t. Le premier arc est $\omega(i0)$, le deuxième arc est $\omega(a)$ et ainsi de suite jusqu'au sommet t : $C_{i0t} = \{\omega(i0), \omega(a), \omega(b)\}$.

La recherche des chemins critiques les plus courts est réalisée en changeant le sens de la comparaison $\pi(j) < \pi(i)+l(u_{ji})$ en $\pi(j) > \pi(i)+l(u_{ji})$ et l'initialisation $\pi(i) = -\infty$ en $\pi(i) = +\infty$.

La recherche dans le sens causal est possible en inversant les ensembles de départ



arcs appartenant à un chemin critique qui a t comme sommet terminal

arcs n'appartenant pas à un chemin critique qui a t comme sommet terminal

FIG. 3.6 – *Arbre des chemins critiques formé par les $\omega(i)$ ayant t comme racine*

et d'arrivée de la recherche et en prenant les successeurs et non pas les prédécesseurs des sommets.

La propriété 3 du graphe de causalité assure qu'il n'est pas possible de rentrer dans un cycle qui aurait comme effet de faire boucler l'algorithme.

3.7.2 Le parcours en profondeur d'abord

L'objectif

Le parcours en profondeur permet de retrouver pour un sommet de stabilité t du graphe de causalité, tous les chemins possibles entre t et les autres sommets de stabilité pour lesquels il existe un chemin vers t. A la différence de l'algorithme précédent, cet algorithme fournit tous les chemins et non pas seulement les chemins critiques.

L'algorithme

Cet algorithme est récursif. Le principe de ce parcours consiste à aller de sommet en sommet jusqu'au premier sommet initial, de revenir à la première bifurcation puis de repartir vers un autre sommet initial et ce, jusqu'à parcourir tous les chemins possibles.

Les ensembles I, T et C sont définis comme pour le parcours en largeur d'abord.

(a) - initialisation

$\forall i \in X - \{t\} \quad \pi(i) = -\infty \quad$ **et** $\quad \pi(t) = 0$

$\omega(i) = \emptyset$

$i = t$

(b) - parcours récursif de la fonction $profondeur(i)$

pour tous les prédécesseurs j de i

si ($\omega(i)$ **et** u_{ji} sont consécutifs **ou** $\omega(i) = \emptyset$) **alors**

$\pi(j) = \pi(i) + l(u_{ji})$

$\omega(j) = u_{ji}$

si $j \in I \cap \bar{C}$ **alors**

récupération du chemin C_{jt}

sinon

rappel de $profondeur(j)$

fin si

fin si

fin pour

La récupération des délais des chemins C_{jt} grâce aux $\omega(i)$ et des temps de propagation $\pi(j)$ est identique à l'algorithme en largeur d'abord mais elle est réalisée dès que l'on arrive sur un sommet initial et non pas à la fin de l'algorithme comme pour le parcours en largeur.

Il est possible de parcourir le graphe dans le sens causal avec les mêmes modifications que pour l'algorithme en largeur d'abord.

3.8 Conclusion

Le graphe de causalité et le graphe des chemins critiques sont les deux principales représentations temporelles d'un circuit. Le graphe de chemins critiques permet de vérifier les règles temporelles. Le graphe de causalité est utilisé par un outil interactif afin d'aider le concepteur à retrouver les informations nécessaires à la correction d'éventuelles erreurs.

Dans la problématique, nous avons mis en évidence que ces deux représentations peuvent générer un volume de données très important qui constitue la principale limitation du traitement de circuits à très haute complexité.

CHAPITRE 4

État de l'art

Dans le chapitre 2 nous avons défini l'objectif de ce travail qui consiste en la vérification temporelle des circuits à très haute densité d'intégration. Nous nous intéressons plus particulièrement à la vérification formelle des contraintes temporelles. La méthode qui permet ce type de vérification est l'analyse temporelle statique.

Nous considérons que pour un concepteur, il est indispensable de pouvoir retrouver facilement les temps de propagation de tous les chemins d'un circuit ainsi que le détail de chaque chemin décrit au niveau des portes logiques afin de pouvoir corriger les erreurs. Nous mettrons donc l'accent sur les méthodes utilisées par les outils existants et plus particulièrement sur leur capacité à traiter de gros circuits ainsi que leurs limitations en terme de performance et de possibilité d'exploitation des résultats.

Notre travail s'est appuyé sur des travaux antérieurs développés dans le Laboratoire ***LIP6***. Ces travaux sont analysés à la fin de ce chapitre.

4.1 Historique

Les premières recherches dans le domaine de l'analyse temporelle remontent aux années 60 avec l'outil PERT présenté dans [KC66]. Les circuits en technologie bipolaire étaient représentés par des portes interconnectées et caractérisées par des délais (maximal, minimal et typique). L'analyse dans PERT consiste à parcourir le circuit en largeur d'abord pour déterminer les chemins les plus longs.

Avec l'avènement des technologies MOS, une nouvelle génération d'outils, TV [Jou83b, Jou83a] et CRYSTAL [Ous83, Ous84, Ous85] utilisant une approche appelée « switch-level » fait sont apparition au milieu des années 80. Ce sont des approches qui se placent au niveau transistor. Les transistors sont modélisés comme des interrupteurs qui commutent instantanément entre un état bloqué et un état résistif. Cette modélisation est utilisée également dans la simulation logico-temporelle [Bry84] pour la modélisation des temps de propagation.

Les outils précédents ont posé les bases de l'analyse temporelle de l'analyse temporelle statique. Depuis, des améliorations significatives sont apparues. Des modèles de transistor plus complexes [SN90b, SN90a, SN91, HMG91, Haj92, MSN92], faisant intervenir des équations de courant non linéaires, ont permis d'améliorer la précision du calcul des temps de propagation. La forme du front d'entrée peut être également prise en compte [AAD90, HMG91, Haj92, CS96, DA97]. D'autres approches [FFG97] ont gardé un niveau d'abstraction au niveau porte en développant des modèles adaptés aux technologies submicroniques.

Avec les technologies submicroniques et l'apparition des effets résistifs des interconnexions, plusieurs modèles [JR81, RPH83, PR90, QPP94, DD95, GTP97, VD97] de temps de propagation dans les réseau RC ont été développés. Ces modèles sont basés, soit sur le délai d'Elmore [Elm48], soit sur des améliorations du délai d'Elmore [DD95], soit sur « Asymptotic Waveform Evaluation » [PR90] qui est une généralisation du délai d'Elmore à tous les types de réseaux.

De nombreux travaux [BMC90, AKH94, SCD89, CGM+97, CGA97, KB97] se sont penchés sur le problème des faux chemins qui est une source d'erreur dans l'analyse temporelle statique. Mais ce problème étant NP complet, toutes ces approches se heurtent à la trop grande complexité des algorithmes mis en œuvre qui les rend inutilisables pour des circuits de tailles importantes. Il existe également des approches hiérarchiques [JCd92, YH95] qui permettent de réduire la complexité des algorithmes de recherche de faux chemins dans l'analyse temporelle statique.

4.2 La gestion des volumes de données

Avec l'apparition des circuits de plusieurs millions de transistors, le problème principal posé par l'analyse temporelle statique est la gestion des volumes de données. Des travaux concernent soit la réduction de la taille des graphes, soit la définition de méthodes hiérarchiques.

4.2.1 La vectorisation

Comme nous l'avons vu dans le chapitre 2, l'explosion du nombre de chemins pose un problème important de modélisation des chemins critiques d'un circuit. Les travaux de l'équipe de l'université de Michigan [VM98] proposent une solution à ce problème grâce à la vectorisation qu'ils appellent : « Bus compression». Cette approche consiste à remplacer dans le graphe des chemins critiques, tous les signaux issus des registres d'un même vecteur de données par un unique signal. Tous les chemins critiques entre tous les registres sources de ce vecteur et un autre registre ou un autre vecteur de données sont représentés par un seul chemin critique qui est le plus critique de tous les chemins entre ces registres.

Dans la figure 4.1, les registres $a0$ et $a1$ sont remplacés par un seul registre a ; ceci a pour effet de réduire le nombre de chemins de 4 à 2.

Cette approche présente trois inconvénients principaux. La compression du graphe induit une modification de l'information car certains chemins critiques sont remplacés

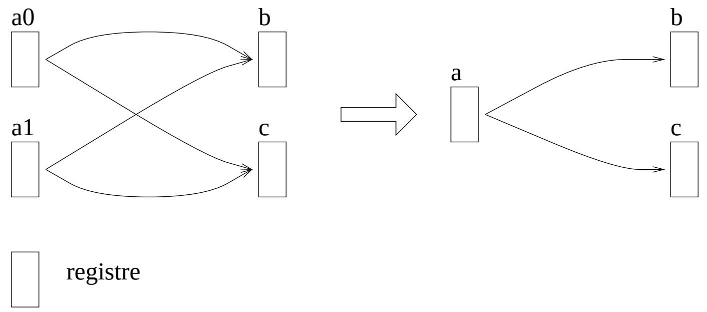


FIG. 4.1 – *La vectorisation*

par des chemins plus longs ou plus courts. Elle demande également que le concepteur de circuit intervienne en précisant quels sont les registres qui appartiennent à un même vecteur. De plus, cette approche n'est pas générale car la vectorisation n'existe pas toujours.

Il serait préférable de trouver une méthode de compression qui soit automatique, générale et qui ne modifie pas l'information contenue dans le graphe de chemins critiques.

4.2.2 Approche hiérarchique avec un modèle matriciel : le modèle CDM

Pour modéliser les chemins dans un circuit, l'outil CAT [YH95] utilise une représentation temporelle hiérarchique appelée CDM. Cette représentation temporelle se présente sous la forme d'une matrice D. Pour un circuit possédant n entrées et m sorties, cette matrice a une dimension égale à $n \times m$.

L'élément (i, j) de la matrice, noté D_{ij}, est l'ensemble des temps de propagation entre l'entrée i et la sortie j. Il y a autant de temps de propagation dans D_{ij} qu'il y a de chemins possibles entre i et j. Cette représentation permet de représenter les temps de propagation de tous les chemins possibles dans un bloc.

Cette représentation est étendue à la détection des faux chemins en associant, à chaque temps de propagation, une condition de propagation par rapport aux valeurs des entrées.

Bien qu'originale, cette méthode ne peut, en aucun cas, être exploitée pour traiter les circuits à très haute densité d'intégration. En effet, la matrice ayant une dimension égale au produit du nombre d'entrées et du nombre de sorties, elle conduit à des matrices de taille très importante. De plus, le fait de garder tous les chemins et non pas seulement les chemins critiques amplifie l'explosion. Son champ d'application se limite donc à des opérateurs combinatoires numériques tels que les additionneurs, multiplieurs, pour lesquels la dimension de la matrice est petite en comparaison du nombre de portes.

Une méthode de construction de l'ensemble des temps de propagation entre chaque entrée et chaque sortie d'un bloc constitué d'instances d'autres blocs, chacune d'elles modélisés par une matrice D est décrite dans [YH95]. Les entrées de ce bloc ont comme instant de transition un ensemble à un élément avec un temps nul. Supposons que l'on sache déterminer les instants de transition d'une entrée i de l'une des instances. On cherche à calculer les instants de transition d'une sortie j de cette instances. On ajoute à chaque temps de l'ensemble D_{ij} le premier instant de transition de i. Ceci définit un nouvel ensemble $D^{'}_{ij}$ pour les temps entre i et j. L'ensemble des temps de transitions de j à partir de l'entrée i : $D^{''}_{ij}$ est construit en faisant l'union de tous les ensemble D'_{ij} obtenus avec chacun des instants de transition de i. L'ensemble des instants de transitions est propagé à tous les blocs qui utilisent j comme entrée. lorsque tous les blocs sont traités, on dispose de tous les instants de transitions des sorties donc de tous les temps entre toutes les entrées et toutes les sorties.

Ce modèle possède donc une capacité de ré-entrance car à partir des matrices des instances d'un bloc, il est possible de déterminer la matrice de ce bloc. Cette propriété de ré-entrance est intéressante, cependant le problème de l'explosion quadratique du nombre de chemins reste entier.

4.2.3 Approche hiérarchique multi-mode : DHTA

DHTA [BDS93, BDS96] utilise une approche hiérarchique multi-mode. Chaque niveau hiérarchique possède un modèle temporel qui est le graphe des chemins critiques contenus dans ce niveau hiérarchique. DHTA propose une méthode pour retrouver les informations détaillées au niveau porte des chemins critiques : Plus le niveau hiérarchique est bas, plus le modèle de calcul des temps de propagation est précis. DHTA suppose que les modèles grossiers des blocs de niveau hiérarchique supérieur donnent des bornes supérieures des temps de propagation, c.-à-d. que le temps d'un chemin détaillé est forcement plus petit que celui calculé au niveau hiérarchique supérieur.

Les informations détaillées d'un bloc père relatives à un chemin se trouvent dans les modèles des instances des blocs fils sous la forme d'un graphe de chemins critiques de ces instances. Comme il n'est pas possible de charger en mémoire la totalité des modèles de toutes les instances à tous les niveaux hiérarchiques pour retrouver le détail au niveau porte d'un chemin critique, cette approche utilise des stratégies de sélection des instances ayant la plus grande probabilité de faire partie du chemin critique recherché. Il s'agit d'une mise à plat partielle de la hiérarchie : seules ces instances sont expansées. Tant que le détail du chemin n'est pas trouvé dans cette partie du circuit, de nouvelles instances sont expansées. L'algorithme s'arrête quand un chemin ayant un délai très proche du chemin critique et passant par des instances de plus bas niveau hiérarchique est trouvé.

Trois stratégies sont présentées pour choisir les instances critiques à un niveau hiérarchique, c.-à-d. celles qui ont le plus de chances de faire partie du chemin critique :

1. stratégie P : une instance est critique si elle est traversée par au moins une proportion P des K chemins les plus critiques. K et P doivent être définis par le concepteur ;
2. stratégie MCP : une instance est critique si elle est traversée par au moins une part P des K chemins les plus critiques ou par le plus long chemin critique ;

3. stratégie W : une instance est critique si son MF (facteur de mérite) est supérieur à un seuil W. Le facteur de mérite se calcule en associant un poids à chaque chemin des K chemins les plus critiques qui passent par cette instance. Ce poids dépend du délai de chemin.

L'approche de DHTA repose sur l'hypothèse rarement vérifiée que le graphe de chemins critiques d'un niveau hiérarchique est plus petit que la réunion des graphes de chemins critiques des instances. Dès qu'il y a explosion du nombre de chemins critiques à un niveau hiérarchique, cette méthode devient inapplicable.

L'idée de ne construire qu'une partie du circuit pour trouver le détail d'un chemin est intéressante ; cependant, la méthode utilisée ne tient ni compte du phénomène d'explosion du nombre de chemins ni du fait qu'il est nécessaire de pouvoir accéder à tous les chemins d'un circuit et non pas seulement aux chemins les plus longs.

4.3 Les outils industriels

Une comparaison des outils industriels utilisés pour l'analyse temporelle est donnée dans l'article [Sch95]. Pour la plupart de ces outils, la description du circuit est une « net-list » au niveau cellules précaractérisées.

L'outil PathMill d'EPIC [Nap96, EPI97] est l'outil industriel qui se rapproche le plus de nos spécifications. Il permet l'analyse temporelle à partir de net-lists transistor et utilise une méthode hiérarchique pour le traitement de très gros circuits.

Pour le traitement des blocs à plat, EPIC préconise le couplage de l'outil d'analyse temporelle statique PathMill avec l'outil d'analyse temporelle dynamique TimeMill qui est un simulateur. PathMill permet de rechercher les chemins qui posent des problèmes. Ces chemins et particulièrement les délais dans les réseaux d'interconnexion sont ensuite étudiés plus précisément par une simulation dynamique avec TimeMill.

L'analyse de circuits de taille importante est réalisée grâce à une approche hiérarchique. Le traitement des blocs à plat permet de générer des modèles temporels pour chaque bloc d'un circuit. Ces modèles servent à vérifier les contraintes temporelles au niveau du circuit et à générer de nouveaux modèles temporels pour un traitement hiérarchique multi-niveaux.

L'approche hiérarchique dans PathMill utilise deux types de modèles temporels pour les blocs :

Le premier modèle utilisé est le graphe des chemins critiques (défini dans le chapitre 3) qui est appelé boîte grise d'un circuit (figure 4.2). Le modèle boîte grise est utilisé pour l'analyse temporelle des circuits contenant des latchs (registre à échantillonnage sur état), car le gain de performance obtenu par l'utilisation de la transparence des latchs ne peut être vérifié que si l'on dispose de tous les chemins critiques. Cette première approche suppose la construction « à plat » du graphe des chemins critiques du circuit à analyser. Nous avons vu dans le chapitre 2 que ce modèle conduit à un volume de données important à cause du phénomène d'explosion du nombre de chemin.

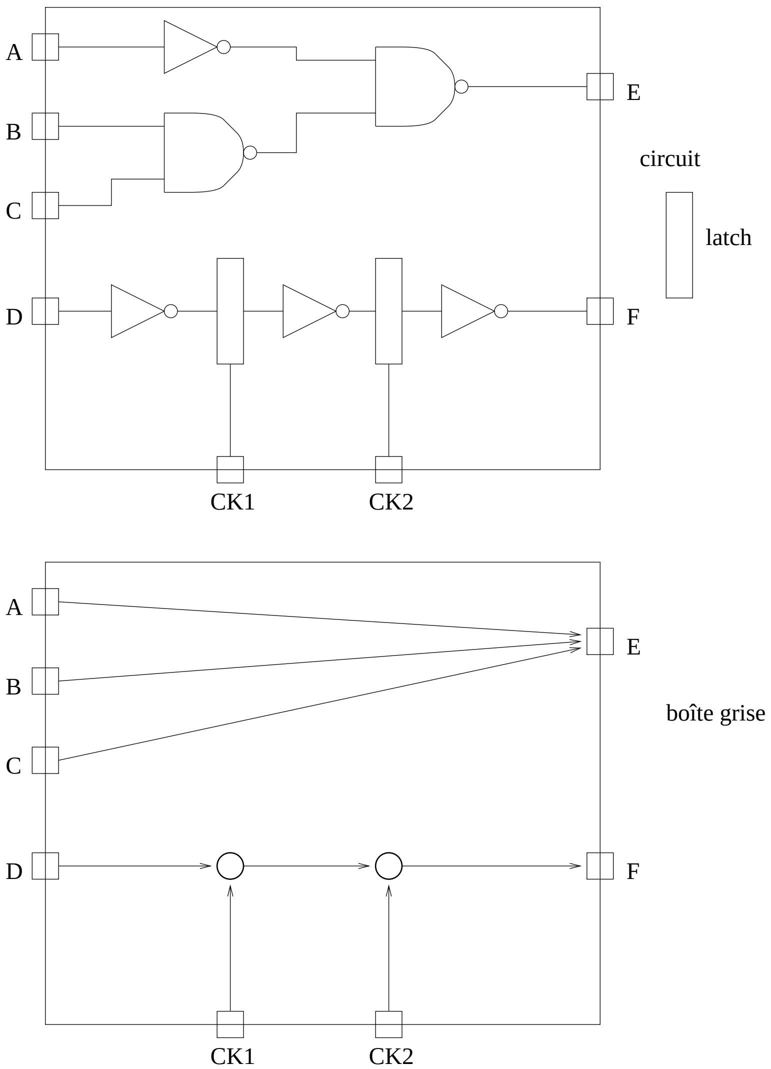


FIG. 4.2 – *La représentation temporelle d'un circuit*

Dans le cas de bascule à échantillonnage sur front, EPIC préconise plutôt l'utilisation d'un deuxième modèle beaucoup plus compact, appelé boîte noire (figure 4.3), qui ne contient que les chemins qui partent et qui arrivent sur un connecteur du bloc, ainsi que les temps de prépositionnement et de maintien sur les entrées, déterminés grâce aux chemins critiques se terminant sur des registres. Les temps d'accès aux données sur les connecteurs de sortie provenant des registres sont donnés à partir du front d'horloge.

Les boîtes noires ainsi définies permettent de s'affranchir de l'explosion du nombre de chemins puisqu'ils sont en général beaucoup plus nombreux de registre à registre que de connecteur à registre ou connecteur à connecteur. Le modèle boîte noire ne permet pas de réaliser une analyse de stabilité en utilisant la transparence des latchs. De plus, il ne peut être utilisé que si le déphasage entre les fronts d'horloges arrivant sur toutes les bascules est négligeable. Surtout, il ne fournit pas les chemins détaillés : Le dé-

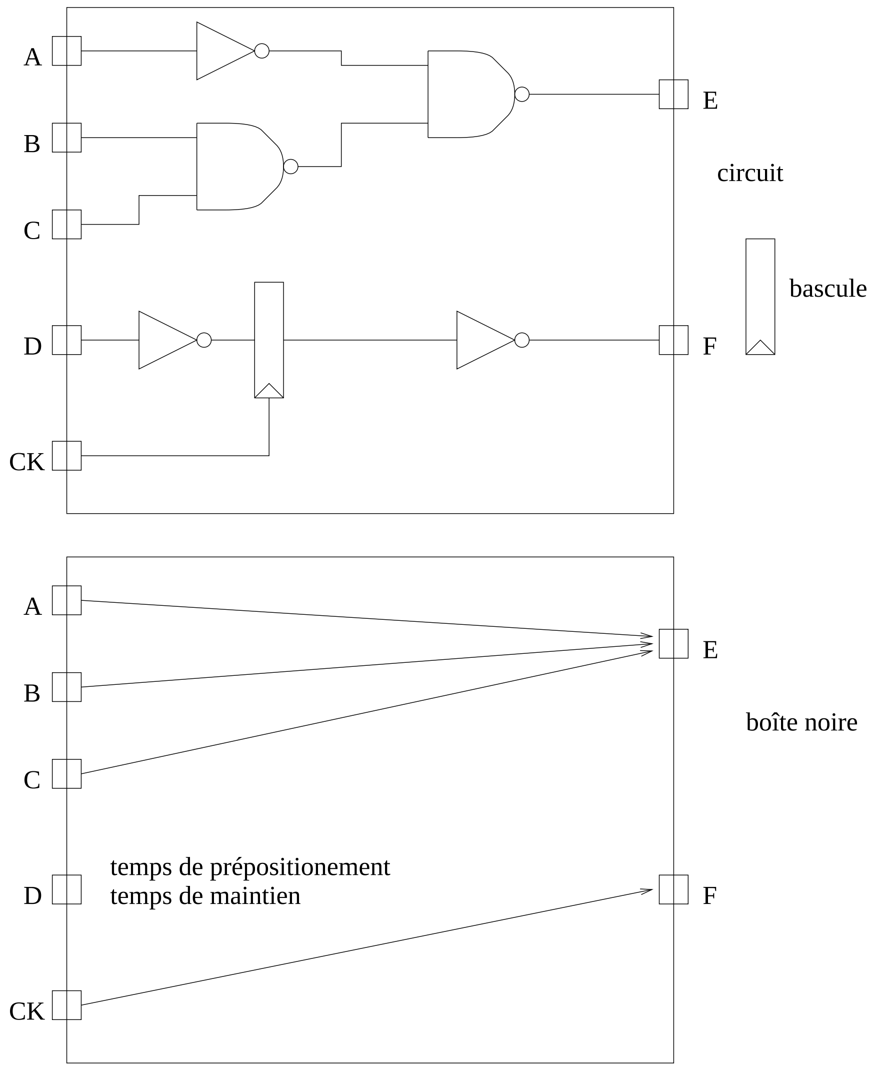


FIG. 4.3 – *La représentation temporelle d'un circuit*

tail au niveau porte est obtenu avec une simulation dynamique avec TimeMill, lors de l'analyse des blocs. Le détail des chemins critiques et les chemins parallèles ne peuvent être obtenus pour une analyse hiérarchique. Dans le cas des boîtes noires, la recherche des chemins entre les registres est nécessairement réalisée à partir des boîtes noires des blocs de niveau hiérarchique inférieur. La manipulation et la création de ces modèles est très coûteuse en terme de temps d'exécution et de consommation mémoire.

L'approche boîtes grises ne peut être utilisée dans le cas des circuits à très haute densité d'intégration, car elle contient tous les chemins critiques d'un circuit, trop nombreux pour un algorithme d'analyse de stabilité ou dans un outil interactif de visualisation des chemins.

Par contre, le principe des boites noires utilisé dans Pathmill présente un avantage intéressant : il n'y a pas de redondance entre l'information contenue dans la boîte noire d'un bloc et l'information contenue dans la boîte noire d'un niveau hiérarchique supérieur. L'ensemble des chemins d'un circuit est donc réparti dans plusieurs modèles. Cette propriété permet de définir une représentation réellement hiérarchique, où chaque niveau instancie des boites noires de niveaux hiérarchiques inférieurs.

4.4 L'analyse temporelle au Laboratoire *LIP6*

L'analyse temporelle représente un axe de recherche ancien au laboratoire ***MASI*** devenu le Laboratoire ***LIP6***. Les travaux ont commencé à la fin des années 80 et se sont concrétisés par deux thèses.

La thèse de Marc Laurentin [Lau94] a permis de définir une méthode de désassemblage de net-lists transistor. Elle a conduit au développement d'un outil logiciel ***Desb***.

La thèse d'Amjad Hajjar [Haj92] a défini un modèle de temps de propagation dans les portes CMOS à partir d'un modèle de courant ***MCC*** (MOS canal court) dans les transistors CMOS.

Elle a conduit au développement de l'outil d'analyse temporel non-hiérarchique ***Tas***.

4.4.1 Le désassemblage avec l'outil *Desb*

Les cônes logiques

Le principe du désassemblage consiste à construire à partir d'une net-list de transistors à plat, une net-list de portes orientées, appelées cônes logiques. A chaque signal contrôlant au moins une grille de transistor ou un connecteur de sortie est associé un cône. Ce cône contient l'ensemble des transistors susceptibles d'établir un chemin électrique entre le signal associé au cône et une alimentation ou un connecteur d'entrée (un cône est équivalent à la notion d'étage définie par [Ous83]). Ce chemin électrique est appelé une branche. Chaque branche contient un nombre variable de maillons qui sont des transistors. Un transistor peut être référencé par plusieurs maillons et peut appartenir à plusieurs cônes à la fois.

Pour chaque cône G, deux ensembles sont définis :

– l'ensemble des entrées G_e qui sont, soit des cônes qui contrôlent au moins une grille d'un transistor du cône G, soit des connecteurs d'entrée appartenant à au moins une branche du cône G ;
– l'ensemble des sorties G_s qui sont, soit des cônes qui ont au moins un transistor contrôlé par le cône G, soit des connecteurs qui sont connectés au signal associé au cône G.

La figure 4.4 représente le multiplexeur à deux entrées. La figure 4.5 illustre sa représentation en cône.

Le cône G possède :

– quatre branches composées respectivement des transistors $T1+T3$, $T1+T4$, $T2+T5$ et $T2+T6$;

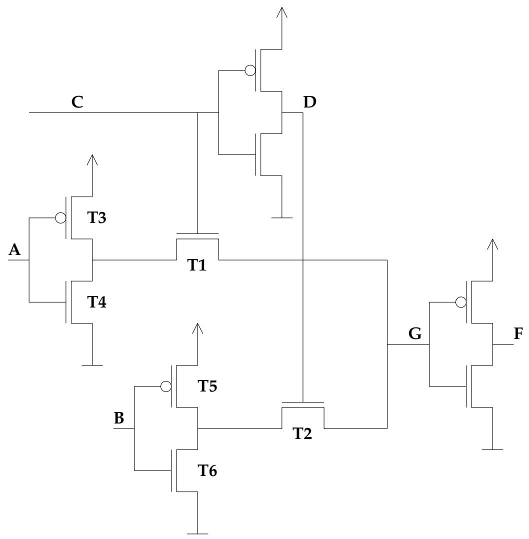


FIG. 4.4 – *Les cônes logiques*

- quatre entrées qui sont les cône A, B,C et D ;
- une sortie qui est le cône F.

Le cône G possède deux branches qui contiennent toutes les deux le même transistor $T1$ et deux branches qui contiennent toutes les deux le même transistor $T2$.

La représentation en cône et en branches est bien adaptée au calcul des délais dans les portes CMOS. En effet, elle permet de définir le sens du courant dans les transistors ainsi que les chemins électriques entre les signaux et les alimentations pour une commutation.

L'élimination des fausses branches

Une des difficultés du désassemblage est l'élimination des fausses branches qui sont celles dont au moins deux transistors ne peuvent pas être passants en même temps.

La figure 4.6 représente le multiplexeur à deux entrées où on a ajouté un inverseur contrôlé par le nœud G. Les branches du cône G passant par les transistors $T1$ et $T2$ sont des fausses branches car les signaux C et D qui les contrôlent sont mutuellement exclusifs.

L'élimination des fausses branches nécessite de vérifier que deux signaux qui contrôlent deux transistors d'une même branche sont mutuellement exclusifs. Il est nécessaire de réaliser une analyse fonctionnelle de la logique en amont de ces cônes.

Dans le cas de la figure 4.6, il suffit de remonter d'un cône pour vérifier la mutuelle exclusion de C et D. Ce n'est cependant pas toujours suffisant ; il faut alors remonter jusqu'aux variables primaires. Les variables primaires sont les signaux qui ne présentent aucune corrélation entre eux.

La phase d'analyse fonctionnelle qui permet l'élimination des fausses branches est une phase qui a une complexité non linéaire par rapport à la taille d'un circuit. Elle est

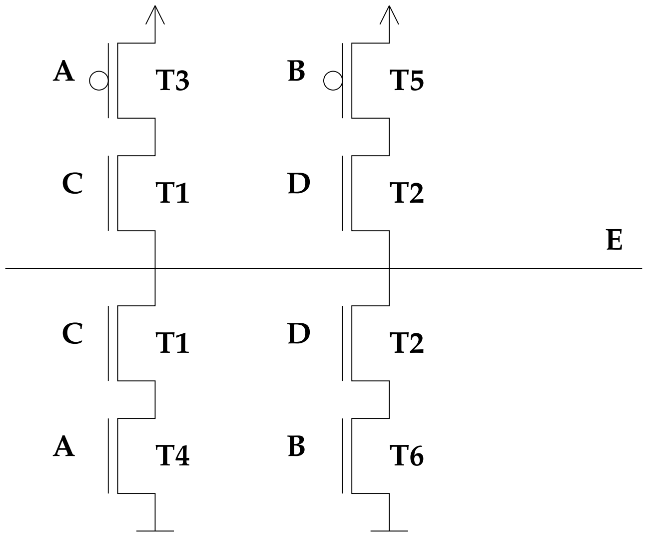


FIG. 4.5 – *Représentation en cône du multiplexeur de la figure 4.4*

donc très coûteuse en temps de traitement et en mémoire. Cette phase est le principal obstacle au traitement à plat des circuits à très haute densité d'intégration.

La reconnaissance des points mémoires

Une fois la net-list de cônes construite, intervient la phase de reconnaissance des points mémoires. Dans ***Desb*** cette phase est basée sur la reconnaissance de formes. Les points mémoires sont déterminés à partir d'un certain nombre de formes construites à partir de cônes rebloucIés entre eux.

Les apports de *Yagle*

Yagle [Les94, LGB97] est un nouveau désassembleur développé dans le cadre de la thèse d'Anthony Lester. Cet outil a permis d'améliorer la phase de désassemblage grâce à des concepts nouveaux comme la généralisation de la reconnaissance des points mémoires par une analyse fonctionnelle ou l'élimination dynamique des fausses branches (pendant la construction des branches).

4.4.2 L'analyse temporelle à plat avec *Tas*

Le modèle de courant *MCC*

Comme nous l'avons vu dans le chapitre 2, l'approche au niveau transistor impose un modèle complexe pour le calcul des délais dans les portes CMOS. Il faut tenir compte de phénomènes difficiles à modéliser comme les effets de front, les courants de conflits, les capacités internes aux portes...

Pour modéliser tous ces phénomènes dans ***Tas***, un modèle de courant dans les transistors MOS a été défini : le modèle ***MCC*** (Modèle Canal Court). Le modèle ***MCC*** donne

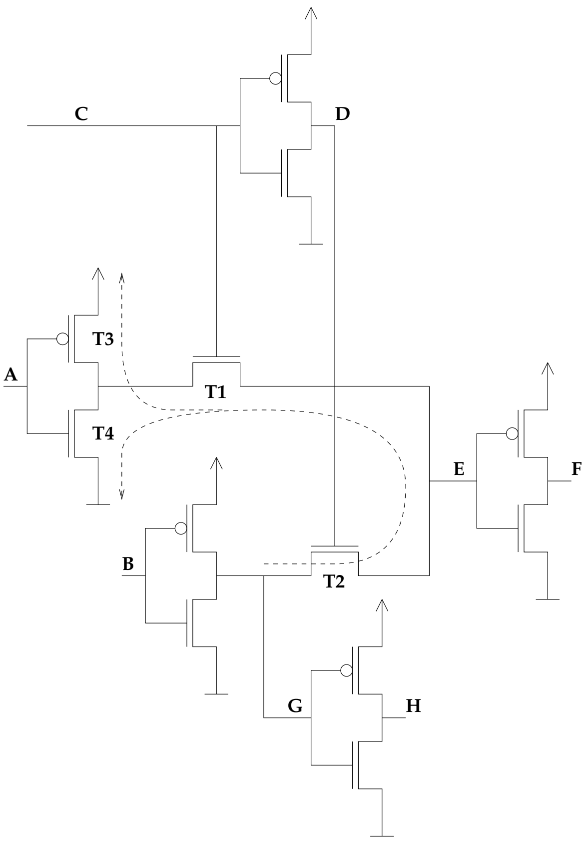


FIG. 4.6 – *Les fausses branches*

le courant de drain dans un transistor MOS en fonction des tensions de grille, de drain et de source de ce transistor.

Il existe trois équations régissant ce courant pour les trois régimes de fonctionnement d'un transistor MOS :

- le régime bloqué : $I_{ds} = 0$ pour $V_{gs} \leq V_t$
- le régime saturé : $I_{ds} = \frac{A \times W \times (V_{gs} - V_t)^2}{L \times (1 + B \times (V_{gs} - V_t))}$ pour $V_{ds} \leq V_{sat}$
- le régime linéaire : $I_{ds} = \frac{V_{ds}}{R_t}$ pour $V_{ds} > V_{sat}$

V_{sat} est la tension de saturation. Elle est égale à $K \times (V_{gs} - V_t)$ avec $0 < K \leq 1$.

R_t est la résistance d'un transistor en régime linéaire. Elle dépend de la tension V_{gs} et est égale à $\frac{V_{sat}}{I_{sat}}$. I_{sat} est le courant dans un transistor MOS en régime saturé pour la tension V_{gs} égale à $V_{sat} + V_t$.

A, B, V_t et K sont des paramètres technologiques déduits des paramètres ***Spice*** donnés par les fondeurs de circuits. Dans le fichier de paramètres de ***Tas*** « .elp », le paramètre K n'est pas donné directement. Il peut être calculé à partir du paramètre R_t donné pour $V_{gs} = V_{dd}$.

Il existe donc huit paramètres de courant dans le fichier de paramétrisation de ***Tas*** « .elp » parmi lesquels quatre sont relatifs au transistor de type N et quatre au transistor de type P. Ajoutées à ces paramètres de courant, se trouvent également les capacités surfaciques et périmétriques de grille et de diffusion ainsi que les capacités de couplage entre les grilles et les diffusions.

La méthode de détermination des paramètres MCC à partir des paramètres ***Spice*** est simple et rapide [DRL98, Duh94]. C'est l'un des principaux attraits de ce modèle. Cette méthode a été formalisée, et est donnée en annexe C.

Le calcul des délais

Nous ne donnerons pas toutes les équations permettant de calculer les temps de propagation dans les portes CMOS. Ces équations sont décrites en détail dans la thèse d'Amjad Hajjar [Haj92]. Nous nous attacherons uniquement à décrire les possibilités de ce modèle ainsi que les phénomènes pris en compte dans l'outil ***Tas***.

Tas permet de calculer les délais de toutes les portes logiques utilisées dans la conception de circuits en technologie CMOS :

- les portes CMOS duales ;
- les portes à transistors de passage N ;
- les portes à transistors de passage N et P ;
- les portes préchargées.

Le calcul de délais dans ***Tas*** tient compte des phénomènes suivants :

- du sens des fronts d'entrée et de sortie ;
- des données des fronts d'entrée ;
- de la position du transistor qui commute dans la branche ;
- du courant de court-circuit dû aux branches passives ;
- du courant de conflit dû aux bleeders et aux inverseurs de rebouclage dans les points mémoires ;
- des capacités internes aux portes (tension de charge et de décharge, courant de charge et de décharge en fonction de la position de la capacité dans la branche).

Les délais calculé par ***Tas*** sont des délais maximaux entre les entrées et la sortie d'un cône.

La construction du graphe de causalité

Les sommets du graphe de causalité représentent les événements sur les signaux associés aux cônes logiques. Pour chaque cône, il existe deux événements : la transition du niveau logique haut au niveau logique bas ainsi que la transition inverse. Cela conduit donc à deux sommets par cône dans le graphe de causalité.

Pour chaque entrée E appartenant à l'ensemble G_e d'un cône G, il peut exister quatre délais entre les deux événements sur E et les deux événements sur G. Chaque délai est modélisé par un arc dans le graphe de causalité.

La recherche des chemins les plus longs

En plus du graphe de causalité, ***Tas*** calcule tous les chemins les plus longs entre chaque couple de signaux de stabilité entre lesquels il existe au moins un chemin.

4.4.3 Les limites de *Tas*

La phase de désassemblage

Le traitement à plat de circuits à très haute densité d'intégration n'est pas possible à cause de la phase de désassemblage. En effet, sur les stations de travail disponibles en 1998, on ne peut désassembler que des blocs ne dépassant pas quelques centaines de milliers de transistors.

Les résistances d'interconnexion

Tas ne permet pas de prendre en compte les délais dus aux résistances des fils d'interconnexion. Or ces délais interviennent notamment dans le cas des interconnexions dues au routage inter-blocs qui ont la particularité d'être très longues donc très résistives.

Les chaînes courtes

Une des erreurs temporelles les plus fréquentes provient de l'existence de chaînes courtes. Cela se produit surtout dans le cas de la circuiterie monophase.

Afin d'éviter de telles erreurs, fatales au fonctionnement d'un circuit, il est impératif de calculer les chemins les plus courts et pas seulement les chemins les plus longs.

Le volume de données généré par *Tas*

Pour des circuits complexes, ***Tas*** génère un nombre énorme de chemins critiques. Comme nous l'avons expliqué dans le chapitre 2, les outils interactifs d'analyse et de visualisation des chemins critiques sont inutilisables avec de tels volumes de données.

4.5 Conclusion

Les différentes approches de vérification temporelle présentées dans ce chapitre mettent toutes en évidence une limitation dans le cas des circuits à très haute densité d'intégration puisque la gestion des volumes de données devient alors problématique. Il existe des méthodes hiérarchiques qui tentent de s'affranchir de ce problème en utilisant des représentations temporelles simplifiées des blocs du type « boites noires ».

Cependant, ces approches se font au détriment des possibilités offertes au concepteur de circuits qui ne peut plus retrouver facilement les informations temporelles détaillées. De plus, l'ensemble de ces approches ne présente pas de vue temporelle des circuits sur plusieurs niveaux hiérarchiques et ne résout pas le problème de la prise en compte des résistances d'interconnexions sur plusieurs niveaux hiérarchiques.

La vérification temporelle statique des circuits à très haute densité d'intégration se doit d'utiliser une méthodologie hiérarchique multi-niveaux avec une prise en compte des résistances d'interconnexion à tous les niveaux hiérarchiques. Elle doit également offrir au concepteur un outil interactif de visualisation de tous les chemins d'un circuit ainsi que du détail au niveau portes de ces chemins.

CHAPITRE 5

Factorisation de chemins

La représentation des chemins les plus longs et les plus courts d'un circuit sous la forme du graphe des chemins critiques est la principale source d'explosion du volume des données manipulées par la chaîne de vérification temporelle puisque le nombre de chemins augmente de façon quadratique avec la complexité des circuits. L'intérêt de cette représentation réside dans le fait de disposer directement des chemins critiques pour appliquer les algorithmes d'analyse des contraintes de stabilité.

Nous en avons déduit que le graphe des chemins critiques n'était pas la représentation adaptée au traitement des circuits complexes. Cela nous a conduit à proposer une autre manière de représenter les chemins critiques.

5.1 Origine de l'explosion du nombre de chemins

Le nombre de sommets N_c du graphe de causalité est égal à deux fois le nombre de signaux. Le nombre d'arcs dépend du nombre moyen N_i d'entrées d'une porte et, dans le cas où les résistances d'interconnexion sont prises en compte, du nombre moyen N_o de portes connectées à la sortie d'une porte. Le nombre d'arcs du graphe de causalité est donc proportionnel à :

$N_c \times (N_i + N_o)$.

N_i et N_o sont en général compris entre 2 et 4. Il est très rare de trouver des portes avec plus de 4 entrées. Le cas des portes avec un nombre d'entrées important se limite aux bus qui sont peu nombreux dans un circuit. La complexité du graphe de causalité est donc proportionnelle au nombre de sommets N_c.

En ce qui concerne le graphe des chemins critiques, le nombre de sommets se réduit au nombre de sommets de stabilité N_s. Le nombre d'arcs dépend du nombre moyen de chemins critiques par sommet de stabilité N_p. La complexité du graphe de chemin critique est donc proportionnelle à : $N_s \times N_p$.

N_p est un nombre très variable qui peut être très grand, surtout dans le cas d'architectures contenant des bancs de registres. En effet, pour un banc de 32 registres de

32 bits (ce qui fait 1024 registres), il peut exister des chemins entre tous les registres du banc (par exemple s'il existe un opérateur de décalage). Si l'on tient compte du fait qu'il y a deux événements par registre, on obtient plus de quatre millions de chemin.

5.2 La factorisation de chemins

La solution que nous avons adoptée pour réduire la complexité du graphe des chemins critiques consiste à augmenter légèrement le nombre de sommets du graphe des chemins critiques de manière à diminuer fortement le nombre moyen N_p de chemins critiques par sommet. Ces sommets supplémentaires sont choisis parmi les sommets du graphe de causalité et d'après leur capacité à réduire le nombre N_p.

Dans la figure 5.1, les ronds noirs sont les sommets de stabilité, les ronds blancs sont les autres sommets du graphe de causalité. Si I est un point extrémité de chemin, le nombre d'arcs du graphe passe de $4 \times 4 = 16$ à $4 + 4 = 8$.

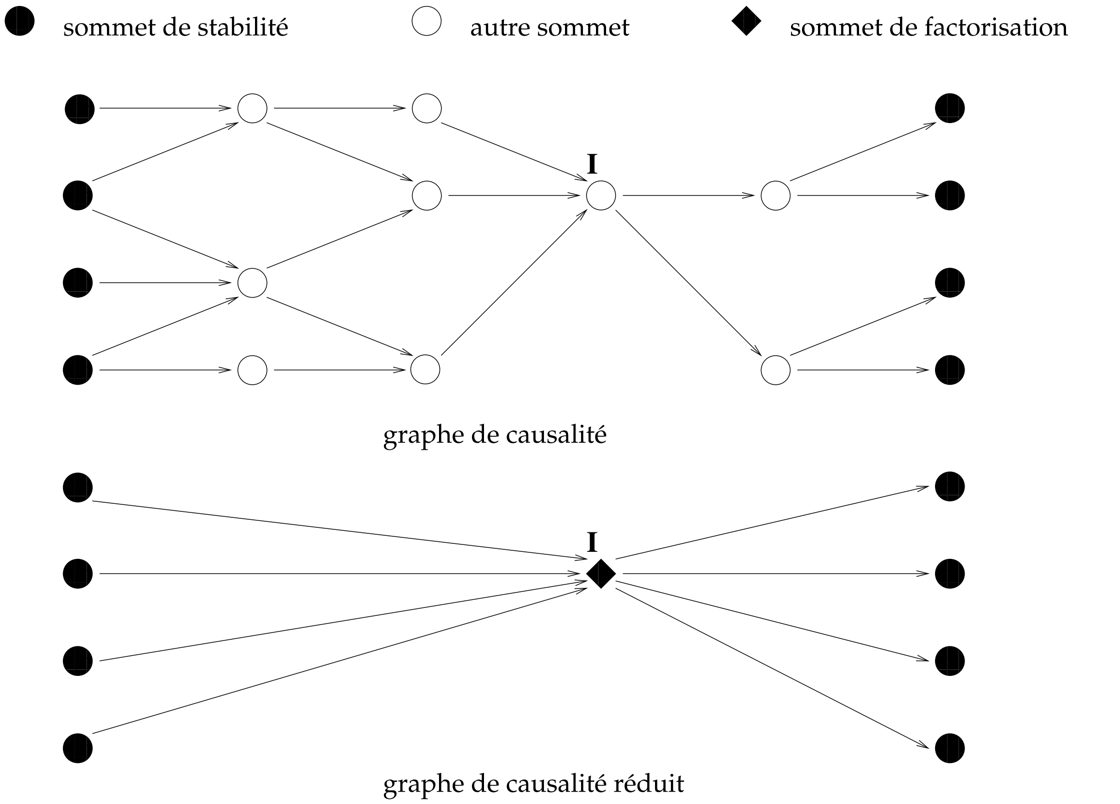


FIG. 5.1 – *Définition de la factorisation de chemins*

Les arcs de ce graphe modifié ne sont pas des chemins critiques (définis dans le chapitre 3), mais des « morceaux de chemins critiques » partagés par plusieurs chemins critiques. Nous parlerons de chemins factorisés. Les nouveaux sommets, extrémités des chemins factorisés, sont appelés sommets de factorisation.

5.3 Le graphe de causalité réduit

La représentation des chemins critiques passe donc par la définition d'un nouveau graphe intermédiaire entre le graphe de causalité et le graphe des chemins critiques : le graphe de causalité réduit. Il permet de représenter les temps de propagation des chemins critiques d'un circuit, par un graphe qui utilise un minimum de place en mémoire. Le passage du graphe de causalité au graphe de causalité réduit est réalisé grâce à une sélection des sommets de factorisation et à une méthode de construction des chemins factorisés. Dans cette section, nous présentons la méthode de construction des chemins factorisés en supposant que nous avons déjà choisi les sommets de factorisation. C'est grâce à cette méthode que l'algorithme de sélection automatique que nous présentons dans la section suivante, donne de bons résultats.

5.3.1 Les signaux de factorisation

Un signal de factorisation est tel qu'au moins un de ses événements est un sommet choisi comme point de factorisation de chemin critique. Cela conduit à définir un nouveau type de signal qui ne fait pas partie de la catégorie des signaux de stabilité.

Les deux événements d'un signal de factorisation ne sont pas forcément des événement de factorisation. De plus, un événement de factorisation peut être intéressant seulement dans le cas d'un type de chemin critique (maximal ou minimal), puisque les chemins critiques maximaux et minimaux ne passent pas forcément par les mêmes sommets du graphe de causalité. On définit donc deux types de sommets de factorisation : les sommets de factorisation de chemins critiques maximaux et les sommets de factorisation de chemins critiques minimaux.

5.3.2 Les chemins factorisés

Dans la figure 5.2, le chemin critique entre $E0$ et $S0$ passe par deux sommets de factorisation : $I0$ et $I1$. Dans ce cas, il existe trois chemins factorisés $c0$, $c1$ et $c2$ qui permettent de décrire le chemin critique entre $E0$ et $S0$. Autrement dit, si un chemin critique passe par un sommet de factorisation dans le graphe de causalité, il passe également par ce sommet dans le graphe de causalité réduit. Il existe donc un chemin factorisé entre deux sommets si :

– ces sommets sont soit des sommets choisis comme des sommets de factorisation, soit des sommets de stabilité ;
– s'il existe un chemin critique dans le graphe de causalité passant par ces deux sommets ;
– si ce chemin factorisé ne passe pas par des sommets de factorisation autres que ces deux sommets.

Comme pour les chemins critiques, un chemin factorisé est caractérisé par :

– son type : maximal ou minimal ;

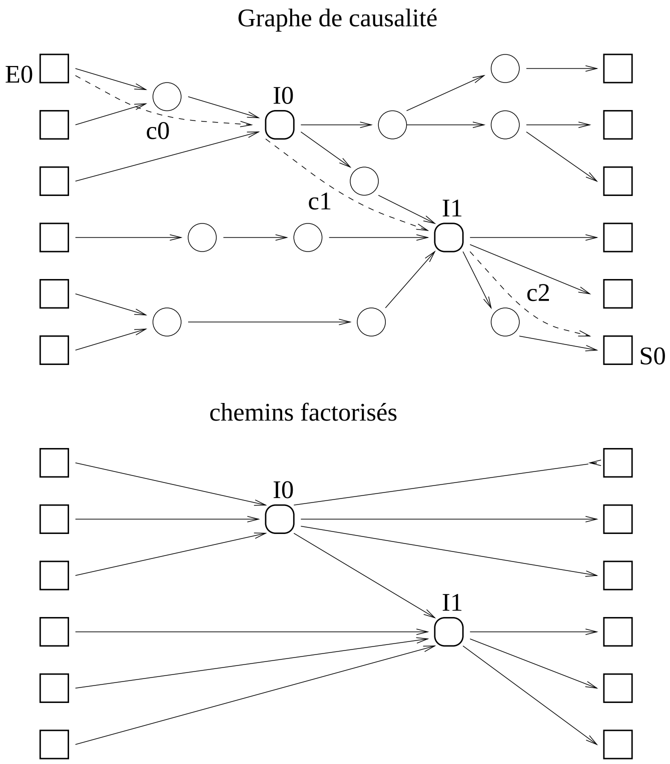


FIG. 5.2 – *Les chemins factorisés*

– la valeur du temps de propagation maximal ou minimal entre son événement initial et son événement terminal ;

– son événement initial ;

– son événement terminal ;

– son événement commande dans le cas des registres et des précharges ;

– ses délais élémentaires.

5.3.3 Construction du graphe de causalité réduit

Nous présentons ci-dessous la définition et la méthode de construction des arcs du graphe de causalité réduit en supposant que nous avons préalablement déterminé dans le graphe de causalité un ensemble de sommets de factorisation.

Définition du graphe

Nous cherchons à construire à partir d'un graphe de causalité $G = [X, U]$ un graphe de causalité réduit $G' = [S, V]$ tel que :

– $S \subset X$ est l'ensemble des sommets du graphe de causalité réduit qui représentent les événements sur les signaux de stabilité ou sur les signaux de factorisation.

– Il existe un arc $v \in V$ entre deux sommets du graphe de causalité réduit si et seulement si il existe un chemin factorisé (maximal ou minimal) entre ces deux sommets.

Un arc du graphe de causalité réduit est caractérisé par :

– son type : délai élémentaire à travers un réseau RC ou chemin passant par au moins un arc de type délai à travers une porte ;
– ses délais maximal et (ou) minimal ;
– son sommet initial ;
– son sommet terminal ;
– son sommet commande.

Un arc du graphe n'a pas forcément les deux délais maximal et minimal. Dans certains cas, un seul des deux lui est attribué car il peut s'avérer qu'aucun chemin critique d'un des deux types ne passe par cet arc.

De la même manière que pour le graphe de causalité, il peut exister des arcs symétriques dans le graphe de causalité réduit. Ce cas ne se produit que si le chemin factorisé correspond à un seul délai élémentaire à travers un réseau RC où les deux orientations sont possibles. Afin de définir la causalité entre deux arcs du graphe, un type est attribué à chaque arc : le type délai élémentaire à travers un arc de type RC et délai à travers au moins une porte. Le principe des arcs consécutifs est le même que pour le graphe de causalité (chapitre 3 section 3.4.2).

Construction des arcs du graphe de causalité réduit

La construction du graphe de causalité réduit à partir du graphe de causalité et des sommets choisis comme sommets de factorisation est réalisée grâce à un parcours en largeur d'abord dans le sens anticausal. L'ensemble des sommets de départ est l'ensemble T des extrémités terminales de chemins. On commence à construire les arcs avec le parcours de recherche des chemins critiques maximaux puis, quand tous les sommets de l'ensemble T ont été parcourus, on relance le parcours avec la recherche des chemins critiques minimaux.

La construction des arcs se fait lors de la phase de récupération de chemins (chapitre 3 section 3.7.1). La figure 5.3 représente un chemin critique C_{it} déterminé par le parcours en largeur d'abord. Il est décrit avec les arcs $\omega(i)$ qui forment un chemin dans l'arbre résultant de la recherche en largeur d'abord (voir le chapitre 3 et la figure 3.6).

Dans le cas où le chemin C_{it} passe par un ou plusieurs points de factorisation, on découpe le chemin en tronçons ayant comme extrémités i, t et les sommets de factorisation f.

Les délais affectés à chacun de ces chemins factorisés est la différence entre les fonctions π de chacune de ses extrémités :

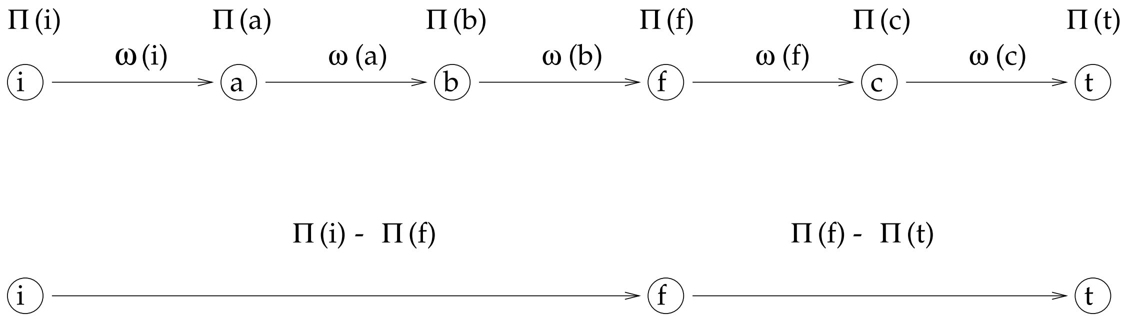


FIG. 5.3 – *Construction des arcs du graphe de causalité réduit*

$\pi(sommet \quad initial) - \pi(sommet \quad terminal)$

Pour respecter le principe de la factorisation, un arc n'est construit pour un chemin factorisé que si cela n'a pas été déjà fait lors d'une recherche précédente à partir d'un autre sommet. Dans ce cas, le chemin a forcément le même temps de propagation que celui trouvé précédemment car c'est le plus long possible (cette propriété est assurée par le parcours en largeur).

Après la recherche des chemins les plus longs, la recherche est relancée pour les plus courts. Lorsqu'un chemin factorisé est déterminé, soit l'arc n'existe pas et on le crée en lui associant le temps du chemin factorisé, soit l'arc existe et deux cas se présentent alors :

– il n'a qu'un chemin factorisé maximal ; on lui associe alors le chemin factorisé minimal ;
– il a déjà les deux et là, il n'y a rien à faire.

Cette méthode a pour avantage de construire uniquement les arcs et les chemins factorisés utiles à la description des chemins critiques. Certains arcs n'ont donc qu'un seul chemin factorisé maximal ou minimal.

La recherche des chemins critiques dans le graphe de causalité réduit se fait de la même manière que pour le graphe de causalité non réduit. Si un arc n'a pas de délai d'un des deux types (maximal ou minimal), c'est qu'il doit être ignoré dans le cas d'une recherche de ce type.

5.4 Choix des signaux de factorisation

Le problème qui reste à résoudre est le choix des événements de factorisation selon leur capacité à diminuer la taille du graphe de causalité réduit.

5.4.1 Sélection du concepteur

La première sélection envisageable est une sélection manuelle. Le concepteur choisit dans un circuit un ensemble de signaux qu'il juge intéressants pour réduire le nombre de chemins. Souvent, ces signaux sont facilement identifiables. Ce sont les bus, les sorties des unités arithmétiques et logiques... qui sont en général d'excellents points de factorisation. Dans la plupart des cas, le choix de ces signaux permet de diminuer considérablement la taille du graphe de causalité réduit. Cela peut être suffisant pour éviter l'explosion du nombre de chemins.

Le concepteur indique le nom des signaux qu'il souhaite sélectionner en tant que signaux de factorisation. Dans ce cas, les sommets représentant les deux événements sur ces signaux sont considérés comme étant des sommets de factorisation à la fois pour les chemins critiques maximaux et minimaux.

5.4.2 L'algorithme de factorisation automatique

Spécifications

Il peut cependant s'avérer difficile de désigner explicitement les signaux de factorisation. Ceci est le cas lorsque l'architecture du circuit n'est pas assez régulière pour que le concepteur puisse identifier tous les signaux de factorisation permettant d'éviter l'explosion de chemins. Il est donc nécessaire de faire appel à un algorithme de recherche automatique pour pouvoir les retrouver.

L'algorithme devra chercher dans le graphe de causalité les sommets de factorisation. Dans ce cas, il peut y avoir des signaux où un seul des deux événements serait un événement de factorisation. Les événements peuvent ne factoriser qu'un seul type de chemins (maximal ou minimal).

L'algorithme de factorisation doit satisfaire certaines contraintes :

– il doit être rapide pour ne pas devenir le point faible de la méthode ;

– il doit pouvoir être applicable à la méthodologie hiérarchique (voir chapitre 6);

– le graphe de causalité réduit qu'il engendre doit être le plus petit possible.

Principe de l'algorithme

La question qui se pose est : peut-on définir une solution optimale? On cherche à construire le graphe « le plus petit » possible en terme de nombre de sommets N_s et de nombre d'arcs N_a. Si l'on considère que l'espace mémoire occupé par un sommet est égal à celui occupé par un arc, on cherche à minimiser la somme $N_s + N_a$ sachant que, pour en arriver là, il faut augmenter N_s. La fonction de coût que nous cherchons à minimiser est donc $Fc = N_s + N_a$. On part du graphe de causalité. La fonction de coût vaut alors la somme du nombre de sommets et du nombre d'arcs du graphe de causalité. Nous allons supprimer certains sommets qui permettent de diminuer notre fonction de coût. Notre algorithme est « glouton » : on ne remet jamais en cause l'élimination d'un sommet. Dès qu'un sommet est supprimé, il ne pourra plus être choisi comme sommet de factorisation.

Le gain de factorisation

On appelle gain de factorisation d'un sommet i, le gain obtenu sur la fonction de coût Fc dans le cas où i est choisi comme point de factorisation.

Le gain de factorisation d'un sommet possède une valeur qui n'est pas intrinsèque. Elle dépend des autres sommets qui ont déjà été choisis comme sommet de factorisation. Autrement dit, dès qu'un sommet est choisi comme sommet de factorisation, tous les gains de factorisation de tous les sommets du graphe sont susceptible d'être modifiés. Nous avons donc un problème complexe car l'espace des solutions à explorer est gigantesque. Puisque le temps de calcul est une contraintes forte, on propose une heuristique qui conduit à un résultat proche de la solution optimale.

Gain de factorisation *a priori*

C'est le gain de factorisation d'un sommet, dans le cas où aucun des autres sommets du graphe de causalité n'est choisi comme sommet de factorisation.

Une entrée primaire est un sommet extrémité initiale d'un chemin temporel.

Une sortie primaire est un sommet extrémité terminale d'un chemin temporel.

On dit qu'une entrée primaire e est vue par un sommet i du graphe de causalité s'il existe un chemin critique ayant comme extrémité initiale e et passant par le sommet i.

On dit qu'une sortie primaire s est vue par un sommet i du graphe de causalité s'il existe un chemin critique ayant comme extrémité terminale s et passant par le sommet i.

Ne_i est le nombre d'entrées primaires vues par un sommet i. Le calcul de Ne_i pour tous les sommets du graphe de causalité est réalisé grâce au parcours en largeur d'abord dans le sens causal. Pour chaque entrée primaire e, on construit l'arbre des chemins critiques partant de e. Pour chaque sommet i appartenant à cet arbre, on incrémente Ne_i d'une unité. Quand toutes les entrées primaires ont été traitées, les nombres d'entrées primaires vues par chacun des sommets du graphe de causalité ont été calculés.

Ns_i est le nombre de sorties primaires vues par un sommet i. Le calcul de Ns_i est le même que pour Ne_i sauf que le parcours est effectué dans le sens anticausal en partant des sorties primaires.

Np_i est le nombre de chemins critiques qui passent par un sommet i. Le calcul de Np_i est également réalisé pendant l'un des deux parcours précédents. On incrémente d'une unité le nombre de chemins critiques de chaque sommet du graphe de causalité à chaque fois qu'un chemin critique passe par un sommet.

Si i est sélectionné comme point de factorisation, le nombre d'arcs du graphe de causalité réduit permettant de représenter les chemins critiques passant par i passe de Np_i (dans le cas où i n'est pas un point de factorisation) à $Ne_i + Ns_i$ (dans le cas où i est un point de factorisation). Ceci n'est exact que dans le cas où aucun de ces chemins critiques ne passe par aucun sommet de factorisation autre que i.

On définit donc le gain de factorisation *a priori* du sommet i par : $Np_i - (Ne_i + Ns_i + 1)$. Dans la figure 5.1, Np_I vaut 16, Ne_I et Ns_I valent 4. Le gain de factorisation du sommet I vaut 7.

Dans le cas des entrées et des sorties primaires, le gain de factorisation est égal à zéro. Si le gain de factorisation d'un sommet autre que les entrées et les sorties primaires vaut zéro, ce sommet n'est pas gardé comme un point de factorisation, car même si la fonction de coût reste constante la profondeur du graphe est augmentée, ce qui ralentit les algorithmes de parcours.

Dès qu'un sommet i a un gain de factorisation *a priori* supérieur à zéro, il permet de diminuer la taille du graphe de causalité réduit. Ceci n'est pas le cas si un autre point de factorisation possède un chemin critique en commun avec ce point de factorisation car dans ce cas, le gain exact n'est pas égal au gain de factorisation *a priori*. Malheureusement, le calcul du gain de factorisation d'un sommet i, sachant qu'il partage un chemin critique avec un autre sommet j, est très coûteux. De plus, il n'est pas simple de garder l'information qu'un sommet partage un chemin critique avec un autre sommet.

Dans un premier temps, on élimine de la liste des candidats tous ceux qui ont un gain de factorisation *a priori* inférieur ou égal à zéro. En règle générale, cette première étape permet d'éliminer la moitié des sommets du graphe de causalité. Le graphe de causalité réduit est construit avec les sommets restants.

Gain de factorisation local

Le gain de factorisation local d'un sommet est son gain de factorisation dans le cas où tous les sommets avec lesquels ce sommet partage un chemin critique sont choisis comme sommets de factorisation. C'est le problème inverse que de celui du gain *a priori*. On cherche à éliminer ceux qui ont un gain local inférieur à zéro.

Le calcul exact du gain de factorisation local n'est pas simple ; en revanche, pour certaines configurations, il est assez aisé de déterminer si le gain local est inférieur ou égal zéro. Si un signal est éliminé de la liste des sommets de factorisation, seuls les sommets prédécesseurs et successeurs de ce sommet ont un gain de factorisation local qui change, ce qui justifie le terme de gain « local ».

Il existe trois cas de sommets dont on est sûr que leur gain de factorisation local est inférieur ou égal à zéro.

- cas 1 : les sommets possédant un seul prédécesseur ou un successeur : le gain de factorisation local vaut -2 ;
- cas 2 : les sommets possédant deux prédécesseurs et deux successeurs : le gain de factorisation local est au plus égal à -1 ;
- cas 3 : les sommets possédant deux prédécesseurs et trois successeurs ou les sommets possédant trois prédécesseurs et deux successeurs : le gain de factorisation local est au plus égal à 0.

La figure 5.4 illustre les trois cas.

Dès qu'un sommet n'est plus considéré comme un sommet de factorisation, il est éliminé du graphe et ses voisins voient leurs nombres de successeurs et de prédéces-

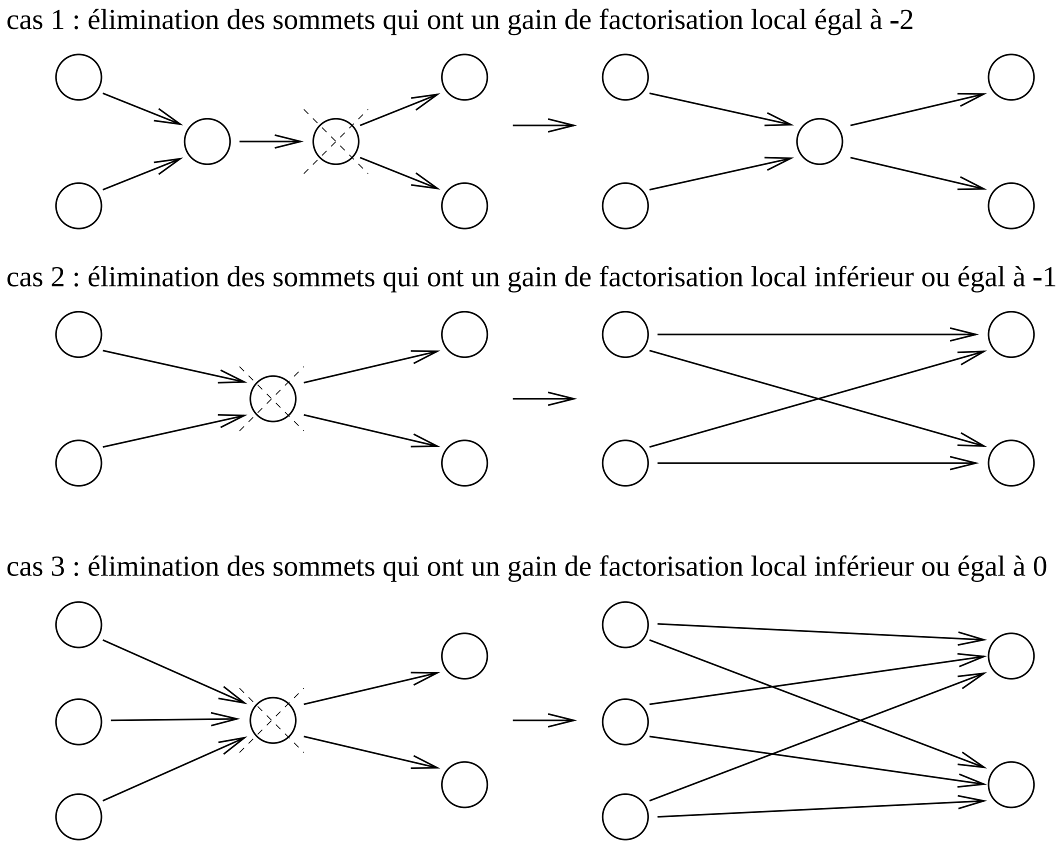


FIG. 5.4 – *Gain de factorisation local inférieur ou égal à zéro*

seurs augmenter. Dans ce cas, les voisins de ce sommet peuvent ne plus faire partie d'un des trois cas. Il est donc nécessaire d'ordonner l'élimination de ces sommets.

On commence par les sommets du premier cas puis du deuxième cas et enfin du troisième cas. Dans le premier cas, on élimine d'abord les sommets qui ont le plus petit gain de factorisation *a priori*. Dans les deuxième et troisième cas, on choisit d'éliminer en priorité les sommets qui ont le plus petit nombre de voisins dans le même cas qu'eux. Parmi les sommets qui ont le même nombre de voisins qui sont dans le même cas, on élimine d'abord les sommets qui ont le plus petit gain de factorisation *a priori*. On construit le graphe de causalité réduit avec les sommets restants.

Il se peut que des sommets qui sont dans un des trois cas précédents réapparaissent car, il est tout à fait possible d'avoir ajouté des liens qui ne font pas partie d'un chemin critique. Ce n'est pas parce que deux arcs consécutifs appartiennent à un chemin critique qu'ils appartiennent forcément à un même chemin critique. Dans ce cas, il faut éliminer ces sommets et reconstruire le graphe de causalité réduit avec les sommets restants. Même si le nombre de sommets restants dépend beaucoup du circuit traité, il ne reste à la fin de cette phase que 10 à 20 % des sommets du graphe de causalité.

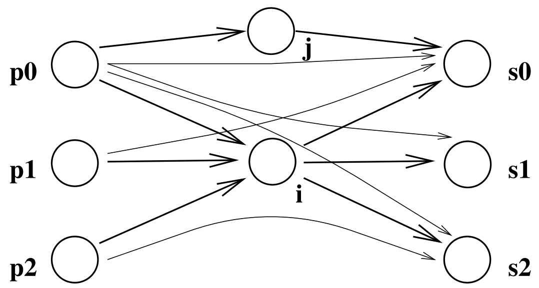
Calcul du gain de factorisation local

Après la phase précédente d'élimination de sommets, il peut rester des sommets qui ont un gain de factorisation local inférieur ou égal à zéro. Il faut donc continuer à éliminer des sommets.

On associe, à chaque sommet restant dans le graphe de causalité réduit, un ensemble de couples de sommets. Pour un sommet i, un couple (p, s) de sommets est composé d'un sommet prédécesseur p et d'un sommet successeur s.

Le couple (p, s) fait partie de l'ensemble des couples de i si et seulement si il existe un chemin critique qui passe par p, i, et s, autrement dit, s'il existe un chemin factorisé qui a comme extrémités p et s et qui passe par i. Pour calculer le gain de factorisation local d'un sommet, il faut calculer le nombre de couples de ce sommet. Si un sommet a P prédécesseurs et S successeurs, il n'y a pas forcément $(P \times S)$ couples possibles.

Dans la figure 5.5 le sommet i n'a que 4 couples (sur 9 possibles). Son gain de factorisation local est donc égal à -3. Il doit donc être éliminé. Les autres combinaisons possibles passent par d'autres sommets. Par exemple, le couple $(p0, s0)$ passe par le sommet j et non pas le sommet i. Pour des soucis de clarté, nous n'avons pas représenté les chemins factorisés des autres couples manquants.



le nombre de couples pour i est 4 : (p0,s1), (p0,s2), (p1,s0), (p2,s2)

FIG. 5.5 – *Nombre de couples d'un sommet*

Il est très coûteux de représenter explicitement tous les couples pour tous les sommets restants, puisque le nombre de couples peut atteindre $(P \times S)$ et que les sommets de factorisation ont un grand nombre de prédécesseurs et un grand nombre de successeurs. Par contre, dès que le nombre de couples dépasse $(P + S + 1)$, ce sommet a un gain de factorisation local qui est supérieur à zéro et doit être conservé. Il suffit de reconstruire le graphe de causalité réduit avec les sommets restants en ne gardant pour chaque sommet que les $(P + S + 2)$ premiers couples trouvés.

A la fin de la construction du graphe de causalité réduit, tous les sommets qui ont un nombre de couples inférieur ou égal à $(P + S + 1)$ peuvent être supprimés. Cependant, dès qu'un sommet est supprimé, le gain de factorisation local de ses voisins change. Dès qu'un sommet est supprimé ses voisins ne peuvent plus l'être. Les sommets sont éliminés dans le même ordre que pour l'élimination précédente. Il s'agit donc d'un processus itératif : à chaque itération, on élimine le maximum de sommets et on recommence la construction du graphe de causalité réduit avec les sommets restants ; ceci jusqu'à ce qu'il n'y ait plus de sommet de factorisation qui ait un gain de factorisation local inférieur ou égal à zéro. A la fin de l'algorithme, seuls quelques pourcents du nombre de sommets initiaux ont été conservés comme sommets de factorisation.

La figure 5.6 récapitule l'algorithme de factorisation automatique.

L'algorithme est rapide car le nombre de cycles dans les deux phases d'élimination de sommets est très petit. Dans le majorité des circuits testés l'algorithme s'effectue en une seule itération. De plus, à chaque étape l'algorithme s'applique sur un graphe de plus en plus réduit.

A la fin de l'algorithme, il n'est pas possible d'éliminer un sommet sans augmenter la taille du graphe de causalité réduit, mais rien ne prouve que l'algorithme conduit au graphe le plus petit possible. Tout dépend de l'ordre d'élimination des sommets. L'élimination prioritaire des sommets qui ont le plus petit gain de factorisation *a priori* conduit à de meilleurs résultats que dans le cas d'un ordre aléatoire. Cependant, les différences de taille du graphe de causalité réduit que nous avons obtenues avec différents ordres d'élimination des sommets restent assez faibles.

5.5 Avantages de la factorisation de chemins

5.5.1 L'analyse temporelle

La factorisation a deux avantages significatifs dans la phase de construction des chemins critiques.

Le premier avantage est la taille de la mémoire nécessaire à la représentation des chemins critiques ; elle est beaucoup plus petite ce qui permet le traitement de plus gros circuits.

Le deuxième avantage est la taille réduite du fichier sur disque qui représente les chemins critiques. Ceci a pour effet de diminuer considérablement le temps d'écriture des fichiers. La diminution de l'espace disque occupé par les fichiers peut atteindre plusieurs centaines de mega octets pour des gros circuits.

Il existe cependant une différence sémantique entre le graphe des chemins critiques et le graphe de causalité réduit. Dans le cas de l'analyse temporelle avec détection des faux chemins, le graphe des chemins critiques permet de ne représenter que les vrais chemins critiques. Par contre cela n'est pas toujours possible dans le cas du graphe de causalité réduit. En effet, un chemin critique formé de plusieurs chemins factorisés sensibilisables individuellement n'est pas forcément sensibilisable. Cette différence ne pose pas de problème car nous avons choisi de ne pas détecter les faux chemins durant la phase d'analyse temporelle.

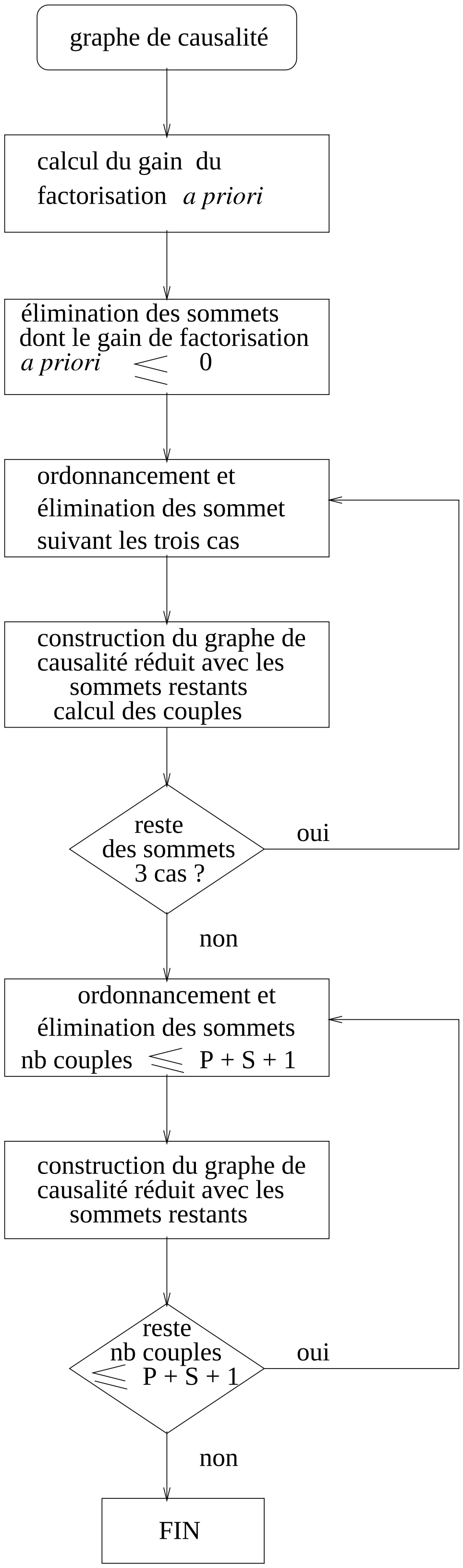


FIG. 5.6 – *Algorithme de factorisation automatique*

5.5.2 L'analyse interactive

Pour l'outil interactif de visualisation des chemins, les avantages sont les mêmes que pour l'analyse temporelle sauf que c'est la phase de lecture des fichiers qui est accélérée. Ceci est très important dans le cadre d'un outil interactif où le temps de réponse doit être réduit.

Lorsque les sommets de factorisation sont déterminés de façon automatique, ils ne sont pas significatifs pour le concepteur : Ce sont les chemins critiques entre les sommets de stabilité et non les chemins factorisés qui intéressent le concepteur de circuits. Les chemins critiques devront être recherchés dans le graphe de causalité réduit grâce au parcours en largeur d'abord. Le concepteur ne consulte que quelques chemins à la fois. Il peut préciser la (les) origine(s) et la (les) destination(s) de la recherche. Les chemins sont donc reconstruits dynamiquement sans qu'ils soient jamais tous présents en mémoire en même temps. Le coût en performance de cette recherche est négligeable dans le cadre d'un outil interactif. Elle ne prend en moyenne que quelques micro secondes par sommet de début de recherche.

5.5.3 L'analyse de stabilité

Le gain de performance dans la phase d'analyse de stabilité est double avec la factorisation. Non seulement la lecture du fichier est accélérée et la taille de la mémoire utilisée pour la représentation du graphe est réduite, mais une adaptation simple de l'algorithme de relaxation permet d'utiliser la factorisation de chemin afin d'obtenir une diminution du temps d'exécution de l'outil. Ceci peut être fait en calculant les variables de stabilité sur les sommets de factorisation à chaque cycle de relaxation. Le graphe de causalité réduit permet de ne parcourir que les chemins factorisés, beaucoup moins nombreux que les chemins critiques du graphe des chemins ou que les délais élémentaires du graphe de causalité.

5.6 Conclusion

Dans ce chapitre, nous avons présenté la factorisation de chemins qui est une méthode efficace pour réduire la taille des fichiers et la mémoire nécessaire à la représentation des chemins critiques d'un circuit. Nous verrons dans le chapitre traitant de l'expérimentation de notre méthode, que dans le cas de circuit où le nombre de chemins explose, nous obtenons des graphes de causalité réduits jusqu'à 100 fois plus petits que le graphe des chemins critiques. Cette méthode évite le phénomène d'explosion du nombre de chemins d'un circuit et permet de garder une dépendance linéaire du volume de données par rapport à la complexité du circuit. Dans le chapitre suivant, nous présentons la méthode de représentation hiérarchique des graphes de causalité et de causalité réduit qui permet de réduire encore la quantité de mémoire utilisée par les outils de la chaîne de vérification temporelle.

CHAPITRE 6

Méthode de vérification temporelle hiérarchique

Comme cela a été mis en évidence dans le chapitre 2, le volume de données générées par les outils d'analyse temporelle limite le traitement de circuits à très haute complexité. Bien que la factorisation de chemins, présentée dans le chapitre précédent, permet d'éviter le phénomène d'explosion du nombre de chemins critiques d'un circuit, les vues temporelles à plat qu'utilisent les outils actuels présentés dans le chapitre 4, ne sont plus adaptées à la taille des circuits réalisables avec les nouvelles technologies fortement submicroniques.

Cette constatation nous a amené à définir une nouvelle approche qui s'appuie explicitement sur la hiérarchie de conception afin de minimiser les ressources en mémoire nécessaires aux outils de la chaîne d'analyse temporelle.

6.1 Principe de la méthode hiérarchique

Nous avons vu que la recherche de chemins dans un graphe nécessite en principe une représentation complète du graphe en mémoire, c.-à-d. une mise à plat de la hiérarchie structurelle. Néanmoins, il existe une notion de localité dans un circuit complexe. En effet, un système intégré est constitué de plusieurs blocs fonctionnels interconnectés entre eux par un nombre limité de signaux de contrôle et à des bus de données. Par conséquent, les chemins d'un circuit ne sont en général concernés que par une petite partie du graphe de causalité et un chemin particulier ne passe que par un petit nombre de blocs. La difficulté est de déterminer la partie du graphe qui permet de retrouver un chemin particulier.

L'idée de base de notre approche consiste à utiliser le découpage hiérarchique de la phase de conception afin de partitionner le graphe de causalité ; ceci permet de ne disposer en mémoire que des parties du graphe de causalité nécessaires pour la construction des chemins recherchés. Ce principe étant appliqué à tous les outils de la chaîne de vérification temporelle, nous avons défini une structure de données bien adaptée à la représentation temporelle hiérarchique.

6.2 La vue temporelle hiérarchique multi-niveaux

6.2.1 La représentation hiérarchique

La représentation structurelle hiérarchique d'un circuit consiste à définir ce circuit sous la forme d'un modèle qui instancie d'autres modèles. Une instance d'un modèle est une boîte noire dans laquelle seule l'interface de ce modèle est décrite. L'ensemble du circuit est donc décrit sous la forme d'un arbre d'instanciation dont les nœuds sont des instances.

6.2.2 La figure temporelle

Une figure temporelle F correspond à un nœud de l'arbre d'instanciation. Une figure temporelle est donc une instance. Il peut donc exister plusieurs figures temporelles identiques en mémoire correspondant à différentes instanciations du même bloc. Une figure temporelle contient toute l'information concernant le graphe de causalité qui ne peut pas être décrite dans les figures temporelles associées aux sous blocs instanciés par F. Celle-ci consiste en la représentation de l'ensemble des signaux, des sommets (événements) et des arcs (délais) appartenant à ce graphe.

Il existe deux types de figures temporelles :

- les figures terminales : elles représentent des blocs du circuit dont la net-list est décrite au niveau transistor. ce sont donc les feuilles de l'arbre d'instanciation ;
- les figures non-terminales : elles représentent des figures dont la net-list est décrite comme une interconnexion d'instances d'autres figures.

Du point de vue de la représentation sur disque, le fichier « .dtx » associé à un bloc représente un modèle qui sera instancié en mémoire autant de fois que ce bloc apparaît dans la net-list hiérarchique.

Un but de la méthode est de ne conserver dans chaque fichier « .dtx » et par conséquent, dans chaque figure temporelle créée en mémoire, que l'information minimale tout en permettant un parcours aisé du graphe de causalité, comme si ce graphe était à plat.

Un circuit est représenté par une arborescence dont chaque nœud est une instance d'un bloc. Cette arborescence peut avoir plusieurs niveaux hiérarchiques.

La figure 6.1 décrit une figure temporelle *Grand-Père* à trois niveaux hiérarchiques. La figure *Grand-Père* n'a pas de nom d'instance car elle n'est pas instanciée dans une figure de plus haut niveau hiérarchique.

Père1 et *Père2* sont deux modèles instanciés dans « grand père ». $P1$ et $P2$ sont leur noms d'instance respectifs dans la figure *Grand-Père*.

Fils1, *Fils2* et *Fils3* sont des figures de plus bas niveau hiérarchique. Elles représentent des figures temporelles terminales correspondant à des blocs dont les net-lists sont décrites au niveau transistors.

Il existe deux figures temporelles identiques du même bloc *Fils1* instancié dans la figure *Père1*. Elles diffèrent par leur nom d'instance $F1$ et $F2$ dans *Père1*.

La notation *Fils1*$/F1$ désigne une figure temporelle instanciée avec comme nom d'instance $F1$.

On dit que F est une fille de P si F est une figure temporelle d'un bloc instancié dans P. P est appelé le père de F.

On appelle descendance de P l'ensemble des figures temporelles du sous-arbre qui a comme racine P.

On appelle les ascendants de F, les figures temporelles de la branche de l'arbre d'instanciation contenant F comprises entre la figure qui instancie F et la racine de l'arbre d'instanciation.

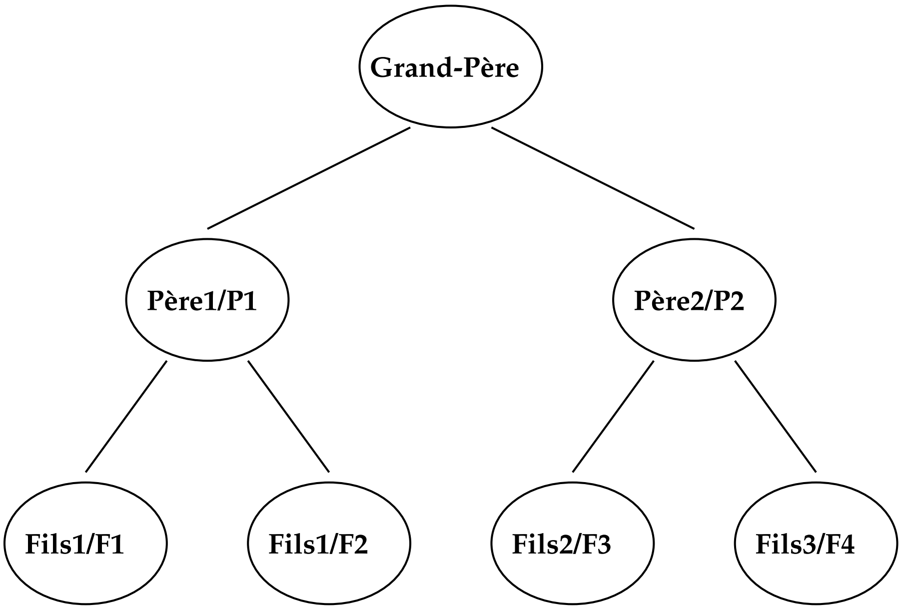


FIG. 6.1 – *L'arbre hiérarchique d'une figure temporelle*

6.2.3 La figure terminale

Une figure terminale est un graphe de causalité à plat. Les résistances des fils d'interconnexion internes au bloc sont négligées. Les seuls temps de propagation pris en compte sont les temps de traversée des portes.

Les signaux temporels

L'ensemble des signaux est divisé en trois sous-ensembles :

- l'ensemble Sc des connecteurs externes de la figure, c.-à-d. les signaux qui appartiennent à l'interface de la figure ;
- l'ensemble Sr des registres, des bus préchargés, des commandes, des signaux de factorisation (chapitre 5) ;
- l'ensemble So des signaux qui ne font partie d'aucun des sous-ensembles précédents.

Une commande peut également être un connecteur. Dans ce cas elle appartient à l'ensemble Sc.

Les événements sur les signaux correspondent aux sommets du graphe de causalité.

L'ensemble Sc définit les signaux qui n'ont un sens que pour la figure dont ils définissent l'interface. En effet, ce ne seront peut-être plus des signaux temporels (définis dans le chapitre 3) lorsque cette figure est instanciée. Nous montrerons dans la suite que cela dépend de la prise en compte ou non des résistances d'interconnexion.

Les arcs

La figure temporelle définit trois ensembles d'arcs :

- l'ensemble U_c des arcs attachés à au moins un événement sur un connecteur ;
- l'ensemble U_e des arcs n'ayant aucun événement sur un connecteur comme extrémité et faisant partie d'au moins un chemin ayant comme extrémité un connecteur. Ils appartiennent à un chemin externe au bloc ;
- l'ensemble U_i des arcs n'ayant aucun événement sur un connecteur comme extrémité et ne faisant partie d'aucun chemin ayant comme extrémité un connecteur, ils appartiennent seulement à des chemins internes au bloc.

Il existe donc trois ensembles disjoints. Les ensembles U_c et U_e permettent de construire l'ensemble des chemins temporels passant par au moins un connecteur. Ce sont les chemins externes. Les ensembles d'arcs U_i et U_e permettent de construire l'ensemble des chemins temporels ne passant par aucun connecteur. Ce sont les chemins internes. Ces derniers ont des délais entièrement définis dans la figure considérée contrairement aux chemins externes qui sont « incomplets ». Ils serviront à construire de nouveaux chemins dans les niveaux hiérarchiques supérieurs. En effet, ces chemins externes passent par les arcs U_c dont les délais seront modifiés aux niveaux hiérarchiques supérieurs car les capacités et les fronts avec lesquels ils sont calculés sont fonction du contexte d'instanciation.

La figure 6.2 illustre les différents ensembles de signaux et d'arcs d'une figure temporelle.

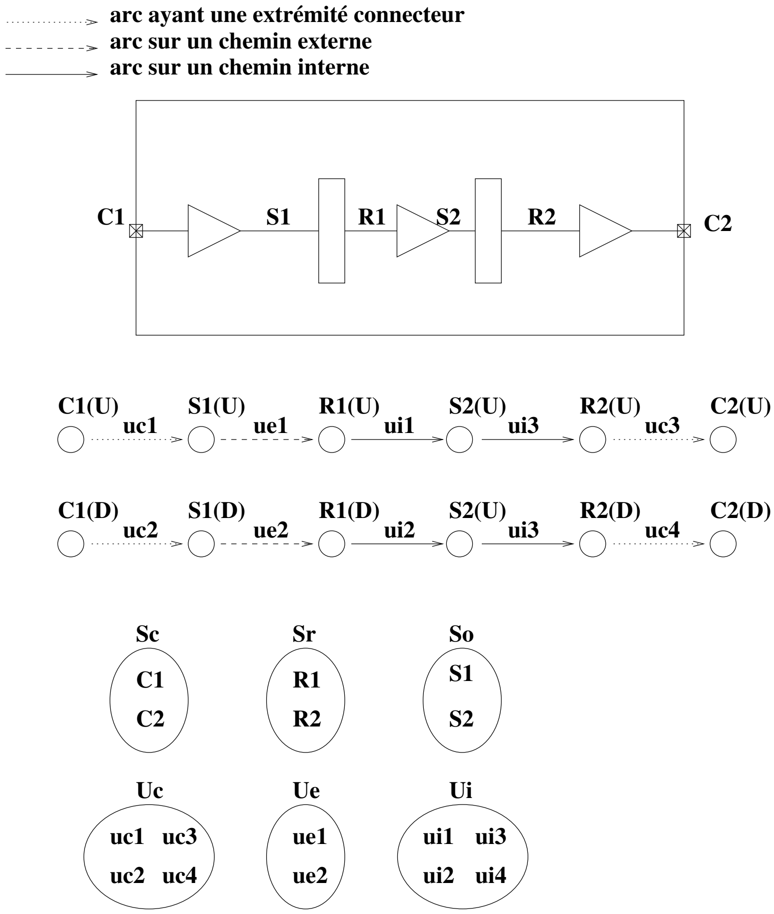


FIG. 6.2 – *Les ensembles de signaux et d'arcs d'une figure temporelle*

6.2.4 Définitions de sous-graphe et de graphe partiel

Afin de présenter notre approche hiérarchique, nous utilisons les notions de sous-graphe et de graphe partiel [GM79].

Soit un graphe $G = [X, U]$.

Sous-graphe

Étant donné $A \subset X$, le sous-graphe engendré par A est le graphe G_A dont les sommets sont des éléments de A et dont les arcs sont les arcs de G ayant leurs deux extrémités dans A.

Graphe partiel

Étant donné $V \subset U$, le graphe partiel engendré par V est le graphe ayant le même ensemble X de sommets que G et dont les arcs sont les arcs V.

Sous-graphe partiel

Étant donné $A \subset X$ et $V \subset U$, le sous-graphe partiel engendré par A et V est le graphe partiel de G_A engendré par V.

6.2.5 La figure non-terminale

Principe

Pour une figure non-terminale F, le principe consiste à ne représenter explicitement que les signaux et les arcs qui ne sont pas définis dans les figures temporelles des blocs instanciés dans F. Ils serviront à construire le graphe de causalité de F.

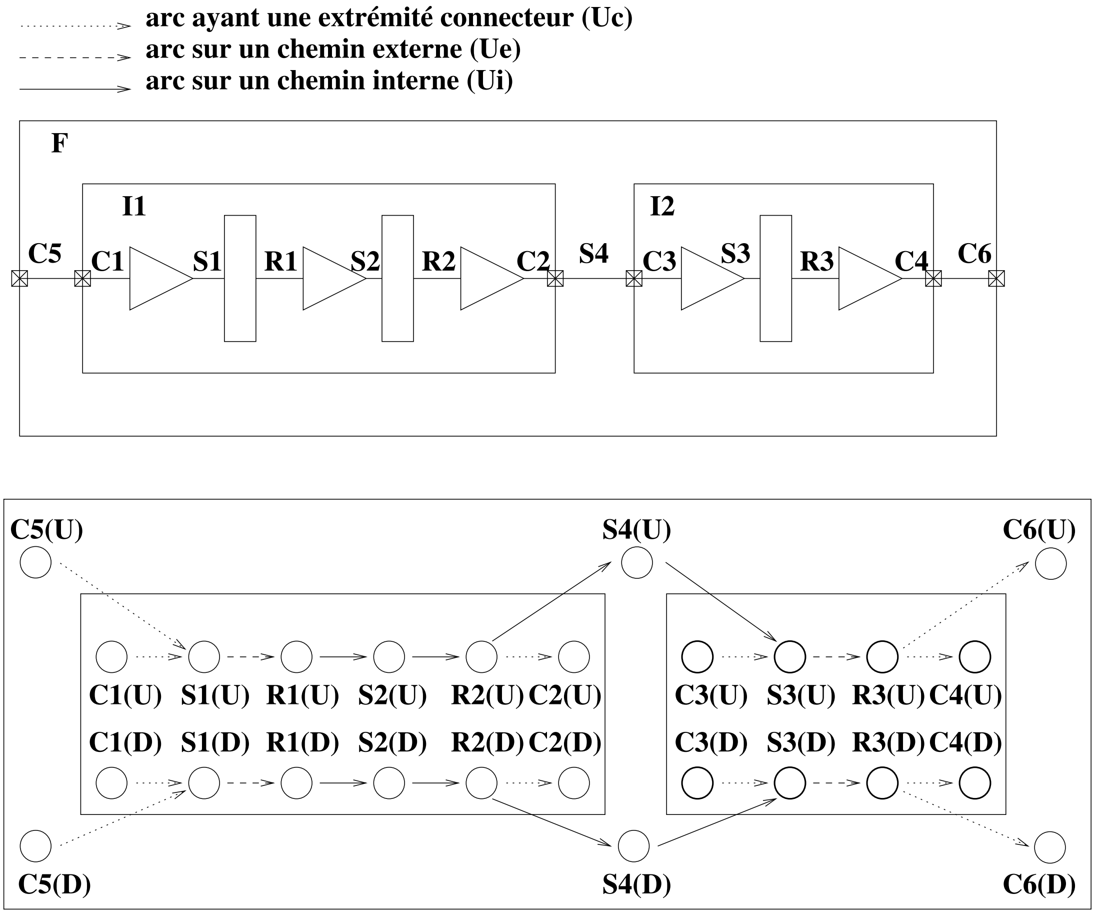


FIG. 6.3 – *Le graphe de causalité hiérarchique*

La figure 6.3 illustre la construction du graphe de causalité de F. Le graphe de causalité de F est constitué des sous-graphes partiels engendrés par les ensembles de signaux Sr, So et les ensembles d'arcs U_e, U_i de toutes les figures des blocs instanciés dans F jusqu'aux figures des blocs terminaux, ainsi que des nouveaux ensembles de signaux Sc, Sr et So et des nouveaux ensembles d'arcs U_c, U_e et U_i appartenant à la figure F.

On voit donc qu'un signal appartient à une seule figure, c.-à-d. à un seul nœud de l'arbre d'instanciation. Les ensembles de signaux Sc et d'arcs U_c et U_c des blocs instanciés dans F ne font pas partie du graphe de causalité de F. Ils ne font partie que des graphes des figures auxquelles ils appartiennent, mais ils n'appartiennent pas aux graphes des figures des niveaux hiérarchiques supérieurs.

La structure hiérarchique

Notre représentation hiérarchique n'est pas identique à la représentation hiérarchique des net-lists où les instances et les modèles sont modélisés par deux objets très différents. Dans notre approche, une instance et un modèle sont représentés par un même objet. Cela se justifie par la nature particulière de l'analyse temporelle pour laquelle les instances ne peuvent pas être considérées comme des boîtes noires. Dans notre cas, les informations nécessaires à la vérification temporelle consistent en l'ensemble des informations contenues dans le modèle.

Les niveaux hiérarchiques

On définit pour chaque figure d'une arborescence de figures temporelles un niveau hiérarchique. Les figures terminales ont un niveau égal à 1. Les autres figures non-terminales ont un niveau correspondant à la profondeur maximum du sous-arbre dont elles sont la racine. Dans la figure 6.1, *Grand-Père* a un niveau hiérarchique égal à 3.

Nous décrivons dans les deux paragraphes suivants, la méthode permettant de construire la figure temporelle d'une figure F à partir des figures temporelles de chacun des blocs instanciés par F et de la net-list décrivant comment les blocs sont interconnectés entre eux.

6.2.6 Les signaux d'une figure non-terminale

Les signaux d'une figure non-terminale F correspondent aux fils d'interconnexion entre les blocs instanciés par F.

La figure 6.4 illustre les deux cas possibles (prise en compte ou non des résistances d'interconnexion), de création de signaux temporels pour la figure temporelle *Père*.

Dans le premier cas, les résistances d'interconnexion sont négligées :

- pour chaque interconnexion non connectée à un connecteur externe de *Père*, un signal appartenant à l'ensemble So (Sr dans le cas où le signal est une commande) de la figure temporelle du père est créé (signal $SP1$) ;
- pour chaque interconnexion connectée à un connecteur externe de *Père*, un signal appartenant à l'ensemble Sc de la figure temporelle *Père* (signal $CP1$) est créé.

netlist hiérarchique

1er cas : les résistances sont négligeables

2ème cas : les résistances ne sont pas négligeables

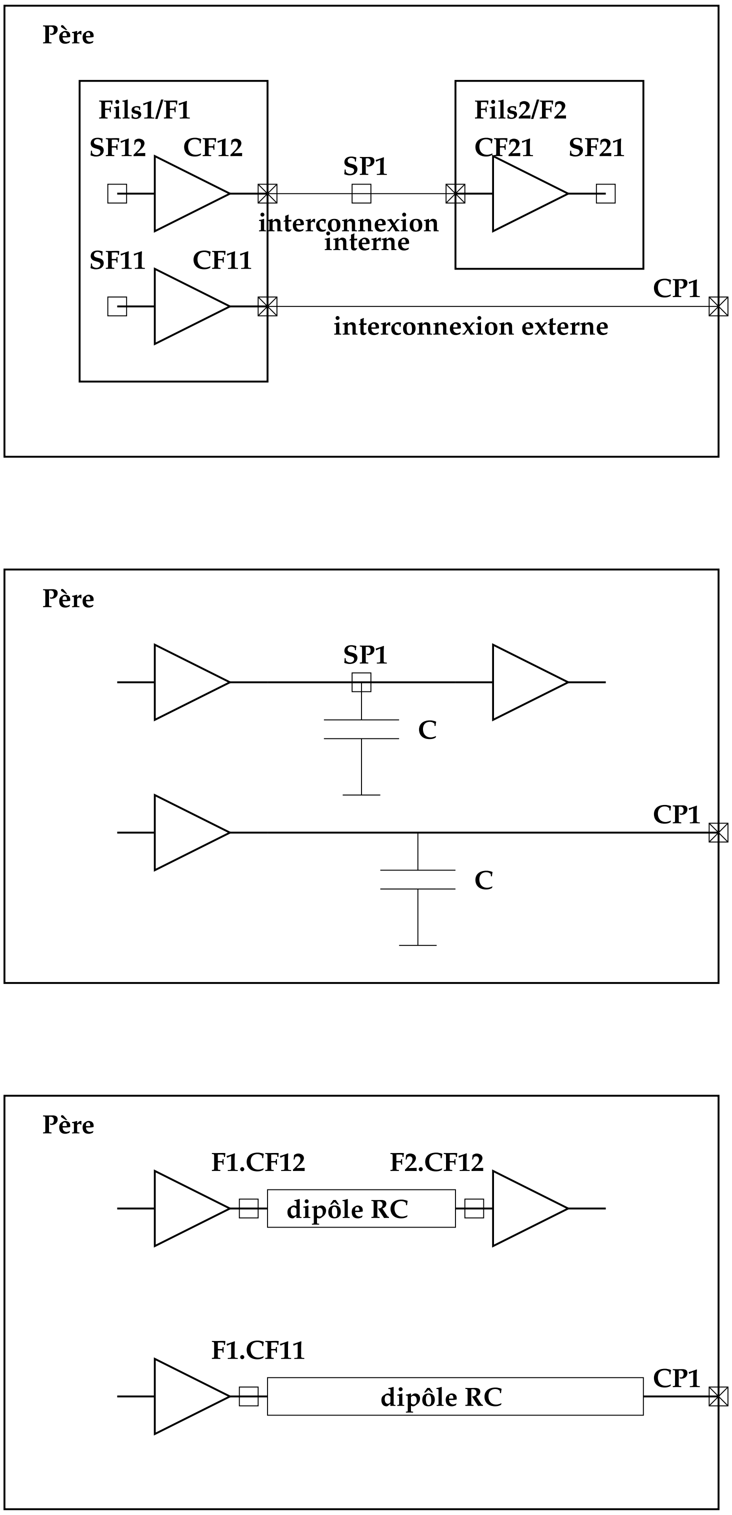


FIG. 6.4 – *Les signaux temporels appartenant à un niveau hiérarchique*

Dans le deuxième cas, les résistances d'interconnexion sont prises en compte, et il faut définir plusieurs signaux pour une même connexion :

– pour chaque interconnexion non connectée à un connecteur externe de *Père*, un signal temporel est créé pour chaque connecteur d'instance connecté à cette interconnexion. Ces signaux temporels appartiennent à l'ensemble So (Sr dans le cas où ces signaux sont des commandes) de la figure temporelle de *Père* ;
– pour chaque interconnexion connectée à un connecteur externe de *Père*, un signal est créé pour chaque connecteur d'instance connecté à cette interconnexion et un dernier pour représenter le connecteur externe de *Père*. Tous ces signaux appartiennent à l'ensemble Sc de la figure temporelle de *Père*.

Les connecteurs d'instances connectés à des interconnexions externes représentés par des signaux temporels appartenant à l'ensemble Sc d'une figure font partie d'une nouvelle catégorie de signaux : les connecteurs internes.

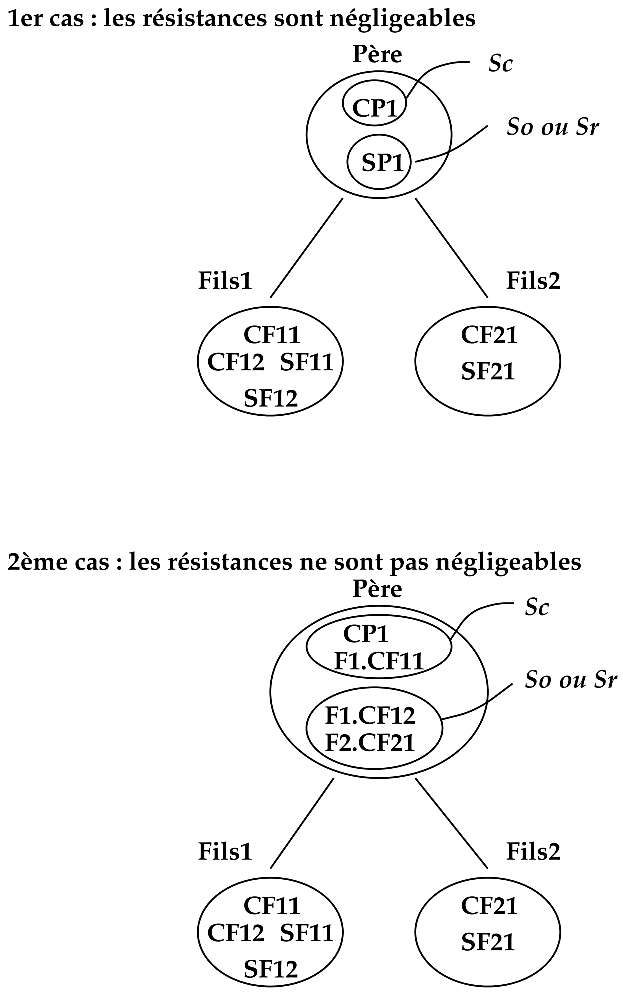


FIG. 6.5 – *Les signaux appartenant à l'ensemble des signaux d'un niveau hiérarchique*

Les signaux référencés

Le but recherché est de créer un arc du graphe de causalité d'une figure F pour chaque délai élémentaire calculé au niveau de la figure F. Or, l'ensemble des signaux appartenant au niveau d'une figure non-terminale n'est pas suffisant pour décrire les nouveaux arcs.

Dans la figure 6.7, afin de construire l'arc entre un événement sur le signal $SF11$ appartenant à la figure *Fils1*/$F1$ dans la figure *Père*/P et un événement sur le signal $F1.CF11$ appartenant à la figure *Père*, il faut avoir accès à ce signal au niveau de *Père* sans avoir à le rechercher dans la figure de plus bas niveau hiérarchique $Fils1$.

Tout signal d'un niveau hiérarchique inférieur *Fils* qui est connecté par une porte ou par un réseau RC à un signal d'un niveau supérieur *Père* est référencé dans la figure *Père*.

L'intérêt d'une telle représentation est que tout signal de stabilité, extrémité d'un chemin qui possède au moins un délai élémentaire calculé dans une figure temporelle particulière, est référencé dans cette figure temporelle.

Tout signal de factorisation, appartenant à un chemin temporel passant par au moins un délai élémentaire calculé dans une figure temporelle particulière, est référencé dans cette figure.

Le niveau hiérarchique le plus haut dans lequel un signal est référencé est appelé le niveau de référence maximal de ce signal.

Dans les ensembles de signaux d'une figure F non-terminale, il existe donc des signaux appartenant à la figure F et des signaux référencés par la figure F. Seuls les signaux des ensembles Sr et So peuvent être référencés car les signaux de l'ensemble Sc d'une figure n'ont un intérêt que pour la figure à laquelle ils appartiennent. Bien qu'un signal appartienne à une seul figure, les ensembles de signaux Sr et So des différentes figures temporelles d'une même sous-arborescence ne sont donc pas disjoints.

Les noms hiérarchiques

Un signal est identifié par son nom. Le nom d'un signal dans une figure non-terminale dépend du niveau hiérarchique. Le nom attribué à un signal est celui qu'il a dans le niveau hiérarchique auquel il appartient. Son nom dans les niveaux hiérarchiques supérieurs est donné par la concaténation du nom du signal et des noms d'instances des figures successives auxquelles il appartient. Le nom n'est donc pas représenté de façon explicite.

Lorsque l'on parle du nom d'un signal, il faut donc préciser dans quelle figure on se place puisque chaque figure de l'arborescence de la hiérarchie représente un modèle temporel d'un bloc.

Dans la figure 6.7, $SF11$ désigne le nom que ce signal a dans la figure *Fils1*, $F1.SF11$ dans la figure *Père* et $P.F1.SF11$ dans la figure *Grand-Père*.

Cette représentation des noms des signaux présente deux avantages importants :

– le préfixe représentant les noms des instances est implicite ce qui a pour effet de

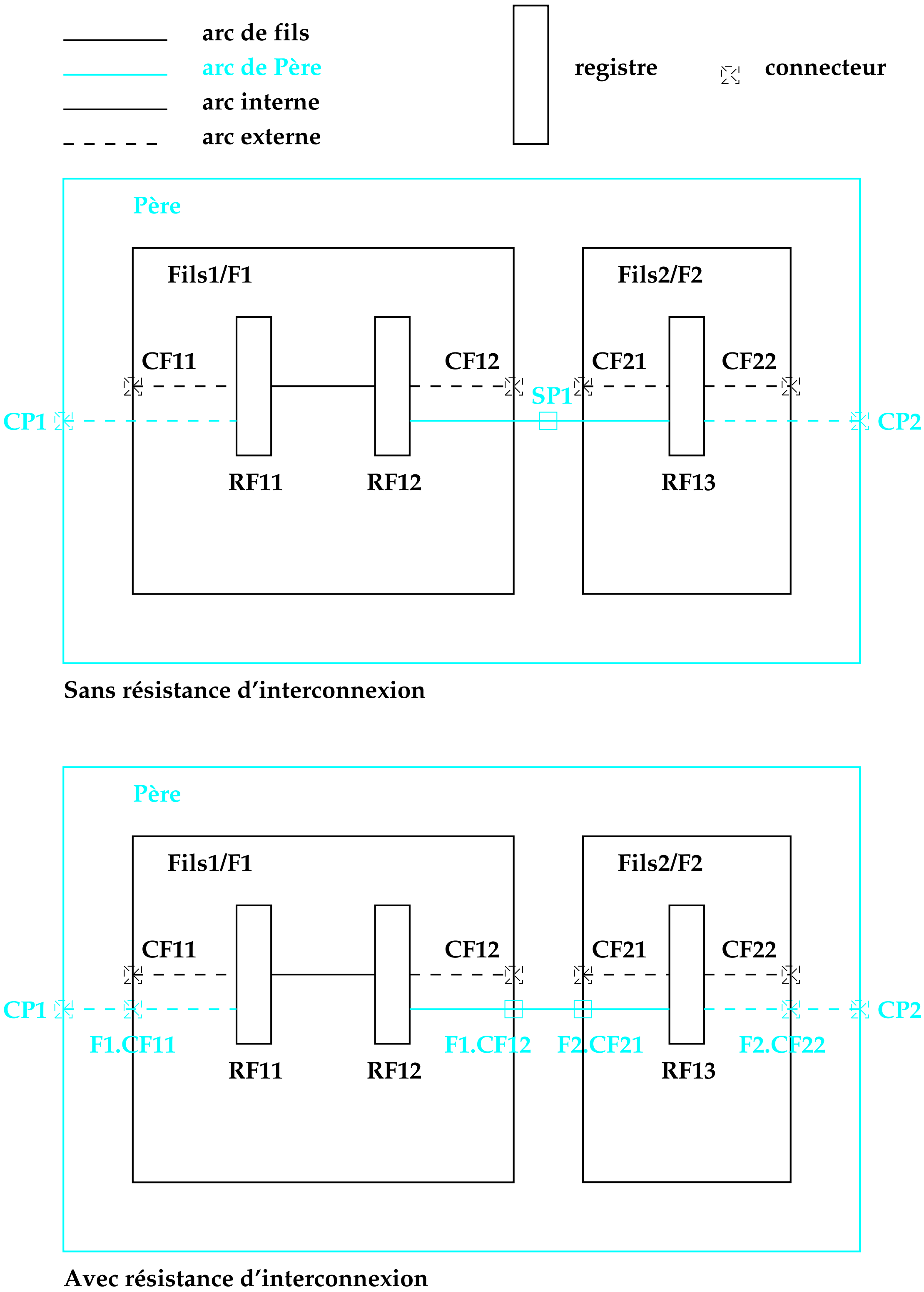


FIG. 6.6 – *Les arcs temporels appartenant à une figure non-terminale*

diminuer très fortement la taille que prennent les noms en mémoire ;

– il est possible de se placer sur toutes les figures de l'arborescence sans avoir à changer de nom de signal, le nom est déduit à partir de la figure où l'on se place.

6.2.7 Les arcs d'une figure non-terminal

Une figure temporelle Père hérite des délais internes aux figures instanciées. Ce sont tous les arcs des ensembles U_e et U_i qui n'ont pas d'extrémités représentant des événements sur des connecteurs des figures Filles. En effet, ces délais ne sont pas modifiés quand on instancie ces figures Filles. Ils sont représentés en trait plein dans la figure 6.6.

Les arcs créés lors de la construction de la figure Père sont les arcs qui représentent des délais passant par des signaux appartenant à la figure Père. Ces arcs sont construits selon les règles définies dans le chapitre 3 pour le graphe de causalité.

Les arcs représentés en pointillés dans la figure 6.6 sont les arcs des ensembles U_c. Comme les connecteurs, ils n'ont un sens que pour la figure à laquelle ils appartiennent. Ils sont remplacés dans les figures de niveau hiérarchique supérieur par de nouveaux signaux et de nouveaux arcs.

Les signaux et les arcs représentés en grisés sont ceux qui appartienent à la figure temporelle *Père*.

Les arcs appartenant à une figure F et les signaux appartenant ou référencés dans cette figure F définissent des sous-graphes partiels du graphe de causalité de F. Ces sous-graphes partiels représentent les délais calculés au niveau de F.

Contrairement aux ensembles de signaux, les ensembles d'arcs des figures d'une arborescence sont disjoints, puisque seuls sont présents dans une figure Père les arcs dont le délai est modifié dans Père.

6.2.8 Les interconnexions sur plusieurs niveaux

Pour les figures temporelles multi-niveaux où les résistances d'interconnexion sont prises en compte sur plusieurs niveaux hiérarchiques, les connecteurs d'instances des niveaux intermédiaires ne constituent pas des signaux temporels car ce ne sont pas des extrémités des réseaux RC connectées aux portes. Ils disparaissent aux niveaux hiérarchiques supérieurs. Les signaux temporels sont les connecteurs de plus bas niveau à partir duquel les résistances sont prises en compte. Ce sont donc les connecteurs internes des figures temporelles instanciées. Ils sont remplacés dans les niveaux hiérarchiques supérieurs par d'autres signaux qui appartiennent soit à l'ensemble So (Sr dans le cas de commande) si l'interconnexion devient interne, soit par d'autres connecteurs internes appartenant à Sc tant que l'interconnexion est toujours externe.

Dans la figure 6.7 $F1.CF11$ et $F2.CF21$ sont des connecteurs internes de la figure temporelle de *Père* et $CP1$ est un connecteur externe de *Père*.

Lorsque *Père* est instancié dans *Grand-Père*, le réseau RC est externe à *Grand-Père*. Les signaux $F1.CF11$ et $F2.CF21$ sont remplacés dans la figure temporelle de *Grand-Père*

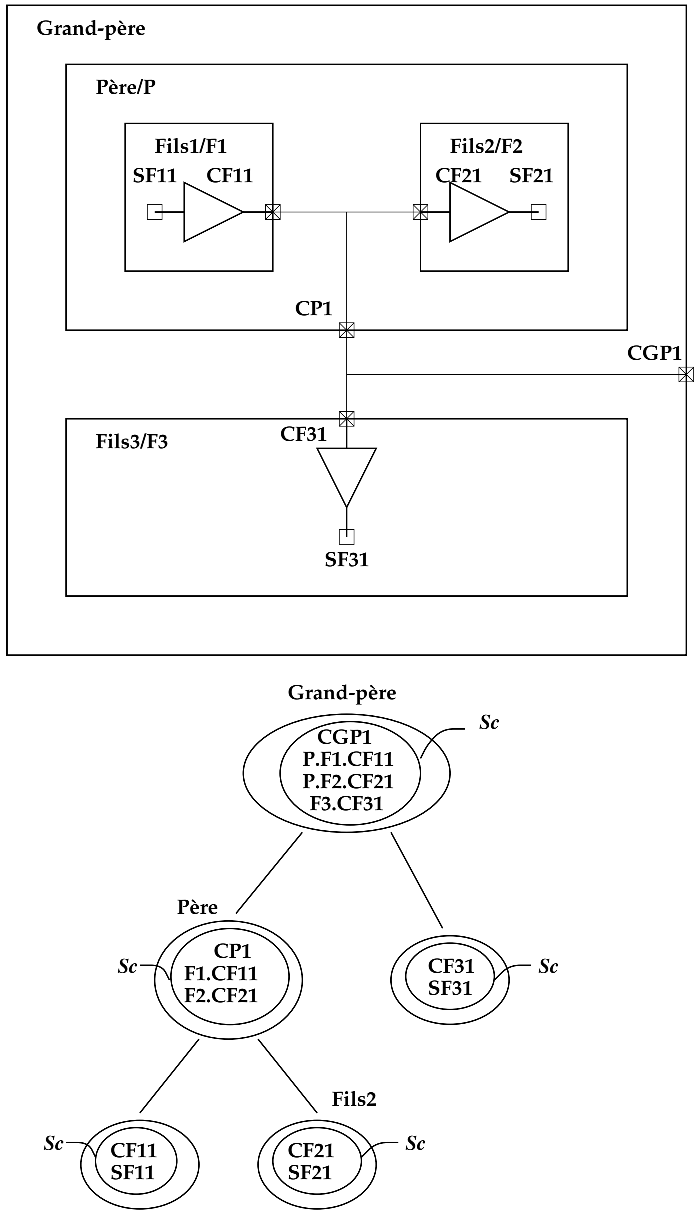


FIG. 6.7 – *Les signaux temporels multi-niveaux des interconnexions externes*

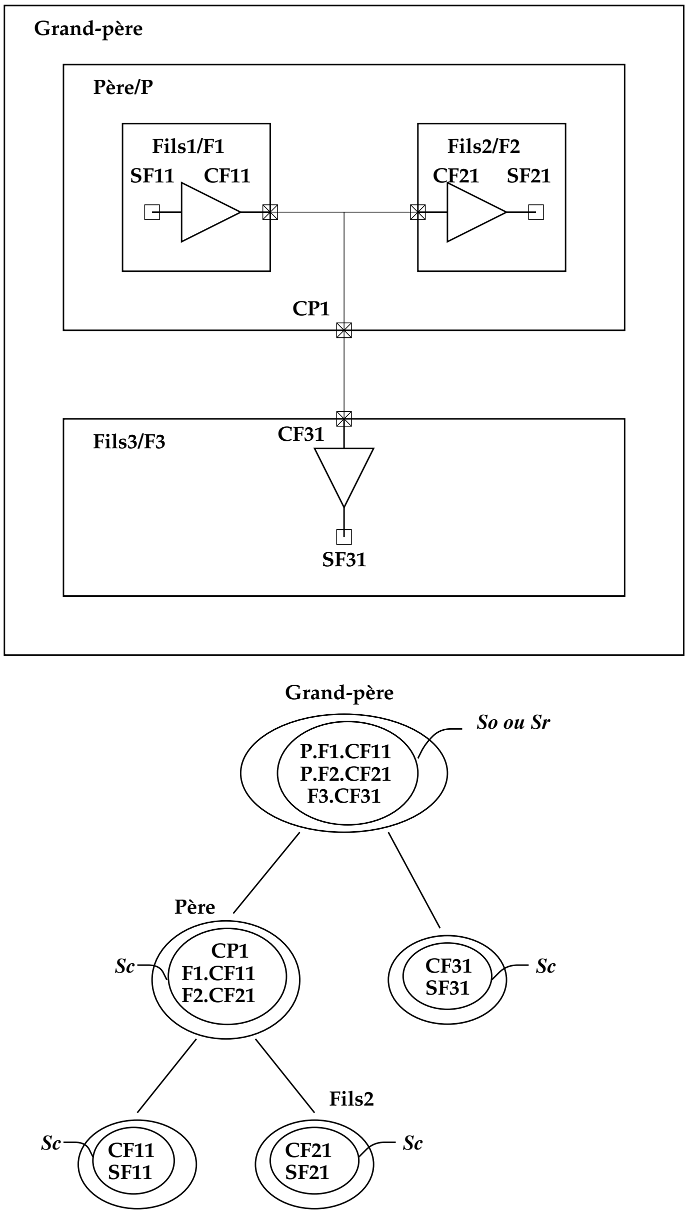


FIG. 6.8 – *Les signaux temporels multi-niveaux des interconnexions internes*

par de nouveaux connecteurs internes $P.F1.CF11$ et $P.F2.CF21$ différents de $F1.CF11$ et $F2.CF21$ de la figure temporelle de *Père*. Bien qu'ils représentent les mêmes points de référence, ce ne sont pas les mêmes signaux. Ceci résulte du principe que les signaux d'un ensemble Sc d'une figure ne peuvent pas faire partie des graphes d'une figure de niveau hiérarchique supérieur. De plus, pour calculer les temps de propagation des délais à un niveau hiérarchique donné, on attribue à chaque connecteur des informations électriques. Ces informations sont différentes d'un niveau à l'autre. Il faut donc créer de nouveaux connecteurs internes à chaque niveau de la hiérarchie.

Si l'interconnexion est interne (figure 6.8), les connecteurs internes $F1.CF11$ et $F2.CF21$ de *Père* sont remplacés dans la figure temporelle de *Grand-Père* par des signaux $P.F1.CF11$ et $P.F2.CF21$ appartenant à l'ensemble So (Sr dans le cas de commande). Ces signaux font partie des graphes de toutes les figures temporelles de plus haut niveau hiérarchique puisque leurs propriétés restent inchangées dans tous les niveaux hiérarchiques supérieurs.

La figure 6.9 donne l'exemple d'un circuit à deux niveaux hiérarchiques avec des réseaux RC à chaque niveau. Dans les graphes de causalité de *Fils1*, *Fils2*, *Père* et *Grand-Père*, les sommets en trait gras sont des sommets appartenant à la figure dans laquelle le graphe est décrit.

6.2.9 Le graphe de causalité hiérarchique

Le graphe de causalité d'une figure F est composé de tous les arcs des ensembles U_e et U_i et de tous les signaux de l'union des ensembles So et Sr des figures de la descendance de F ainsi que des ensembles d'arcs U_c, U_e et U_i et des signaux des ensembles Sc, So et Sr appartenant à la figure F. C'est un graphe à plat. Tous les algorithmes et toutes les propriétés présentées dans le chapitre 3 s'appliquent de la même manière que pour le graphe à plat d'une figure temporelle terminale si ce n'est qu'il existe des délais d'interconnexion.

La principale différence est que tous les signaux et les arcs du graphe de causalité appartiennent à des sous-ensembles qui définissent des sous-graphes partiels de ce graphe. Ces sous-graphes définissent un partitionnement local du graphe. Notre méthode consiste à exploiter ce partitionnement afin de ne construire que le sous-graphe partiel utile pour le traitement en cours.

6.2.10 Les signaux de commandes

Le cas des signaux de commande est assez particulier car ce sont les seuls signaux qui peuvent être également des connecteurs externes ou internes. Les signaux de commandes, qui appartiennent à deux catégories (connecteurs et commande), font partie de l'ensemble Sc. La conséquence de cette appartenance est que ces signaux n'ont un sens que dans la figure temporelle à laquelle il appartiennent.

Cette dernière conséquence pose un problème. Un arc possède un attribut commande, qui pour les arcs des ensembles d'arcs U_i et U_e, n'est pas le même en fonction

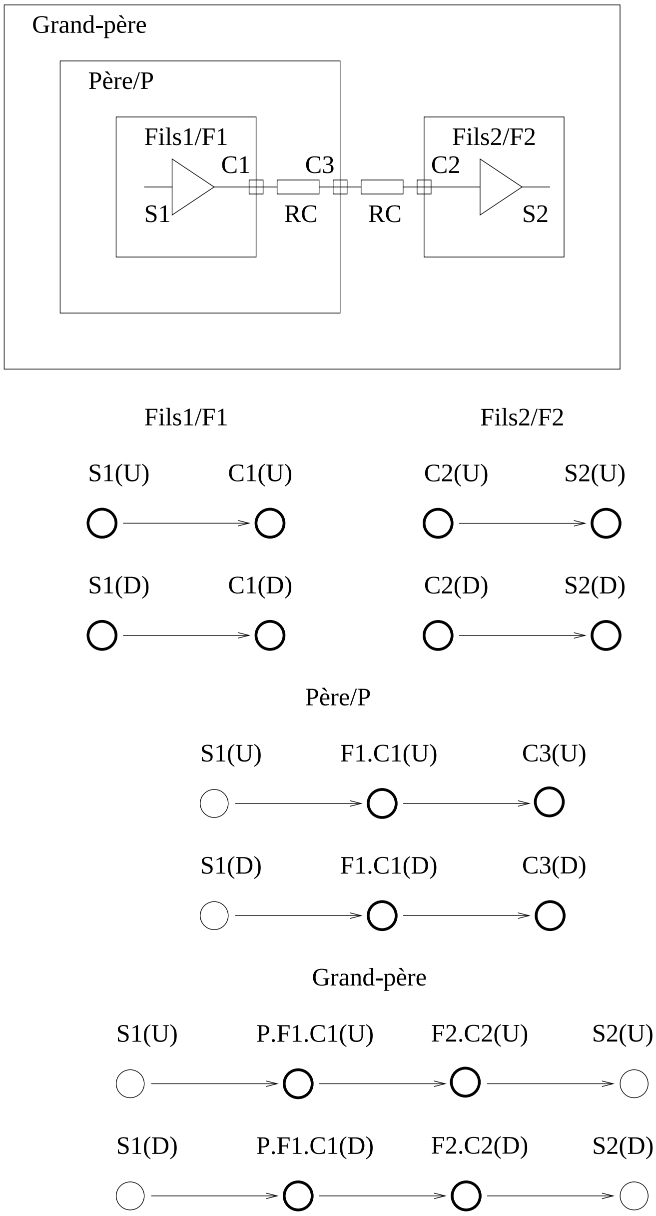


FIG. 6.9 – *Le graphe de causailté avec des interconnexions sur plusieurs niveaux*

du niveau hiérarchique dans lequel on se trouve. De plus, un registre doit connaître la liste de ses commandes. De la même manière, cette liste peut changer d'un niveau hiérarchique à l'autre. Cela impose de changer la notion de commande pour les attributs des arcs et pour la liste des commandes d'un registre.

Il faut créer une indirection entre le signal de commande et l'attribut commande. L'attribut a donc une commande équivalente en fonction de la figure dans laquelle on se trouve comme pour le nom d'un signal.

La construction de cette table d'indirection est très simple. Il suffit d'attribuer à chaque commande connecteur un lien sur la commande équivalente au niveau hiérarchique supérieur.

6.3 La représentation sur disque

La vue temporelle d'un circuit doit avoir une représentation sur disque. Cette représentation sert de passerelle entre les différents outils de la chaîne de vérification temporelle. Elle a également une très grande importance dans une approche hiérarchique. En effet, dans une méthode de vérification incrémentale, l'intérêt d'une approche hiérarchique consiste à utiliser la représentation sur disque comme une représentation intermédiaire entre le traitement de deux niveaux hiérarchiques consécutifs. La représentation sur disque d'une figure temporelle constitue la référence. La représentation en mémoire, elle, peut être partielle. Les signaux et les arcs d'une figure ne sont présents en mémoire que s'ils sont utilisés.

La représentation sur disque est hiérarchique. Pour chaque bloc, il existe un fichier « .dtx » pour représenter le graphe de causalité de ce bloc.

Il existe une représentation pour chaque bloc mais pas pour chaque figure. En effet, plusieurs instances d'un même bloc sont représentées par le même fichier « .dtx ». Le fichier « .dtx » est donc le modèle du bloc alors que la figure temporelle en mémoire est une instance de ce bloc.

En revanche, seules les informations contenues dans une seule instance d'un même modèle sont chargées à partir du fichier « .dtx ». Les informations contenues dans les autres instances de ce modèle sont dupliquées à partir de la première instance chargée en mémoire, ce qui est beaucoup plus rapide que de relire le fichier « .dtx ».

Le fichier ayant pour nom « $F.dtx$ » représente les ensembles de signaux Sc, Sr et So appartenants ou référencés dans la figure F et les ensembles d'arcs U_c, U_e et U_i appartenants à la figure F. L'ensemble des fichiers « .dtx » de tous les blocs et sous-blocs appartenant à l'arborescence décrivant un circuit permettent de construire en mémoire le graphe de causalité de ce circuit.

Ce point est très important : il signifie qu'il n'existe aucun fichier gigantesque contenant le graphe de causalité à plat. Le graphe de causalité à plat est construit de façon dynamique en mémoire et il s'agit en général d'une construction partielle où seule l'information utile est chargée en mémoire.

La syntaxe du fichiers « .dtx » est donnée dans l'annexe B.

6.4 Le cache logiciel

6.4.1 Pourquoi un cache logiciel?

Même si la recherche de chemins temporels dans les graphes de causalité nécessite une représentation à plat, il existe une propriété de localité qui doit être exploitée pour minimiser les ressources en mémoire.

Pour chaque opération de chaque phase de la vérification, ce ne sont pas les mêmes informations qui sont nécessaires et pas au même moment. D'où l'idée de ne conserver en mémoire que ce qui est utile pour une opération donnée.

La phase de construction des chemins d'une figure F n'a besoin que des chemins externes aux figures instanciées par F.

L'outil d'analyse de stabilité a besoin de tous les chemins ayant comme extrémités des latchs à mémorisation sur état à tous les cycles de relaxation.

Dans un outil d'analyse interactive, seuls quelques chemins sont consultés. Il n'y a donc aucune raison d'avoir la totalité du graphe de causalité en mémoire. L'utilisation d'un cache logiciel est donc indispensable.

6.4.2 Les états de la structure

La structure temporelle a été définie de manière à pouvoir être construite partiellement en mémoire. Le contenu des informations associées à chaque figure va pouvoir évoluer au cours des opérations effectuées sur cette structure. On appelle état d'une figure les ensembles de données temporelles contenues dans la figure à un instant donné. L'état d'une figure est une donnée explicite de la figure. Cet état définit les ensembles de signaux et d'arcs présents dans cette figure.

Nous noterons par exemple l'état $Sc.Sr.So.U_c.U_e.U_i$ l'état complet de la figure. L'état vide n'a pas d'intérêt. Sr est l'état minimum des figures blocs de la descendance d'une figure ; $Sc.Sr$ est l'état minimum de la figure de plus haut niveau hiérarchique. Ces états minimaux définissent les points de départ de la recherche de chemin.

L'état $Sr.So$ suffit à représenter tous les signaux d'une figure fille ; l'état $Sc.Sr.So$ suffit à représenter tous les signaux de la figure de plus haut niveau hiérarchique.

Pour les ensembles d'arcs, il est indispensable que les signaux correspondants soient présents dans la figure car un arc est défini entre deux sommets représentant des événements sur des signaux.

6.4.3 La construction des figures à partir des fichiers

La phase de récursion sur les instances

Quand on veut charger une figure en mémoire, la première étape de lecture des fichiers consiste à créer l'arborescence de la descendance de cette figure. Cette étape est récursive. La première partie d'un fichier « .dtx » contient la liste des instances d'une figure. Pour chacune des instances figurant dans cette liste, il faut créer la sous-arborescence des descendants de ces filles et mettre chacune des filles dans l'état Sr et ceci récursivement jusqu'aux figures terminales. Quand l'arbre est construit la figure de plus haut niveau est mise dans l'état $Sc.Sr$.

Dans le cas des signaux référencés dans plusieurs niveaux hiérarchiques, on construit une table de hachage afin de les retrouver rapidement et de les référencer de niveau en niveau.

Toutes les opérations effectuées durant cette phase font appel à des informations qui se trouvent en début de fichier. Il suffit donc de lire pour chaque figure les premières lignes des fichiers.

La chargement sélectif

Il ne reste plus qu'à changer les états de chacune des figures de l'arborescence en fonction des opérations à effectuer. La fonction de lecture du fichier « .dtx » possède donc deux paramètres : le pointeur sur la figure à compléter et l'état dans lequel elle doit se trouver à la fin de la lecture. La fonction de lecture se charge de rechercher les informations dans le fichier. Cette phase de changement d'état n'est pas récursive. Une seule figure de l'arborescence change d'état par appel à cette fonction. Le changement d'état est appelé expansion de la figure.

L'expansion ne détruit pas les ensembles déjà présents dans une figure. Ainsi, lorsqu'une figure contenant déjà des informations subit une expansion, cette figure peut se retrouver dans un état contenant plus d'informations que l'état qui est demandé.

La gestion de cohérence des fichiers

Le système de chargement partiel d'une figure, à partir de plusieurs fichiers représentant un circuit décrit hiérarchiquement, nécessite une gestion de la cohérence des fichiers. Ceci implique que figurent dans chaque fichier des informations permettant la vérification de règles de dépendance entre les fichiers de la hiérarchie. Ces informations sont présentes dans l'entête de chaque fichier. On y trouve :

- la date et l'heure de la création de la figure d'un bloc ;
- le nom et le numéro de version de l'outil qui a permis de générer ce bloc ;
- le nom et le numéro de version de la technologie dans laquelle est conçu le circuit.

La date définit un instant pour les figures temporelles en mémoire. Si cet instant n'est plus cohérent avec celui présent dans un fichier, ceci implique qu'il y a eu modification de ce fichier. En général, les équipes de conception partagent les fichiers temporels. Cette vérification permet de prévenir tout changement malheureux de fichier pendant son utilisation.

De plus, la méthode de construction hiérarchique ascendante impose que les fichiers des plus hauts niveaux hiérarchiques soient générés en dernier. Lors de la construction de l'arborescence, la cohérence des dates entre les figures des différents niveaux hiérarchiques est vérifiée.

Toutes les figures d'une même arborescence doivent être générées avec la même version d'outil et la même version de technologie. Dans le cas de la technologie, une différence conduit à un message de mise en garde et non pas un message d'erreur.

6.5 Les algorithmes de recherche de chemin

Le problème à résoudre est la recherche d'un ou plusieurs chemins dans la structure temporelle hiérarchique.

Nous avons vu dans le chapitre 2 que ce problème n'est pas simple car il n'est pas possible de savoir *a priori* par quelles instances passent ces chemins.

Prenons le cas de la recherche du chemin le plus long entre deux signaux de stabilité. Les seules informations dont nous disposons sont l'événement et la figure de départ ainsi que l'événement et la figure d'arrivée. La méthode consiste à construire le graphe de causalité pendant la recherche de ce chemin. Le graphe est chargé en mémoire au fur-et-à-mesure de la progression de la recherche. A la fin de cette recherche, le sous-graphe partiel permettant de décrire ce chemin est en mémoire.

6.5.1 Les parcours de graphes hiérarchiques

Les parcours de graphes hiérarchiques sont similaires à ceux décrits dans le chapitre 3. La seule modification consiste à sélectionner les arcs participant au parcours et à compléter le graphe à chaque sommet parcouru.

Supposons que la recherche porte sur le chemin le plus long entre deux registres. Ce processus de recherche s'effectue en deux phases : la phase d'initialisation suivie de la phase de recherche de chemins.

La phase d'initialisation

La phase d'initialisation se fait en trois étapes :

1. La première étape consiste à déterminer le niveau hiérarchique de recherche maximal. On a vu qu'un registre est référencé dans toutes les figures dans lesquelles il existe au moins un chemin ayant comme extrémité un événement sur ce registre et passant par un arc de cette figure. Parmi toutes les figures dans lesquelles est référencé ce registre celle qui a le niveau hiérarchique le plus grand définit le niveau de référence maximal de ce registre. Le niveau de recherche maximal est le plus petit des niveaux de référence maximal des deux registres.

2. La deuxième étape consiste à déterminer le niveau de recherche minimal entre les deux registres. C'est le niveau hiérarchique le plus bas à partir duquel les deux registres sont référencés dans la même figure. Autrement dit, c'est le premier niveau où il peut exister un chemin entre ces deux registres.

3. Dans le cas où le niveau de recherche maximal et le niveau de recherche minimal existent, il est possible de trouver un chemin entre ces deux registres. La troisième étape est la phase d'expansion des figures.

   Pour chacun des deux registres :

   – il y a expansion des figures comprises entre la figure à laquelle ils appartiennent et l'avant dernière figure avant le niveau de recherche minimal dans l'état :
   $Sr.So.U_e$;
   – il y a expansion des figures comprises entre le niveau de recherche minimal et l'avant-dernière figure avant le niveau de recherche maximal dans l'état :
   $Sr.So.U_e.U_i$;
   – Si le niveau de recherche maximal est le niveau de la figure de plus haut niveau alors il y expansion de la figure de plus haut niveau dans l'état :
   $Sc.Sr.So.U_c.U_e.U_i$. Sinon la figure du niveau de recherche maximal est mise dans l'état :
   $Sr.So.U_e.U_i$.

La figure 6.10 illustre deux phases d'initialisation correspondant à la recherche successive de deux chemins : le premier entre Ra et Rb qui correspond aux figures entourées d'un carré en trait plein et aux états en caractères gras ; le deuxième entre Ra et Rc qui correspond aux figures entourées d'un carré en trait pointillé et aux états en caractères normaux.

Dans le cas de l'initialisation correspondant à la recherche d'un chemin entre Ra et Rb, le niveau de recherche minimal correspond à la figure $FIab$ et le niveau de recherche maximal correspond à la figure $FRabc$ qui est la figure de plus haut niveau.

Dans le cas de l'initialisation entre Ra et Rc, le niveau de recherche minimal et le niveau de recherche maximal sont les mêmes et ils correspondent à la figure de plus haut niveau $FRabc$.

La phase de recherche de chemins

La recherche de chemins est réalisée avec les algorithmes du chapitre 3 qui fait l'hypothèse d'un graphe à plat. La seule différence consiste à détecter, pendant le parcours, les figures qui ne sont pas expansées dans l'état correspondant à la recherche en cours.

La détection des figures non expansées est très simple. A chaque fois qu'une figure est expansée, elle est marquée. Pendant le parcours, dès que l'on arrive sur un sommet représentant un événement sur un signal qui appartient à une figure qui n'est pas marquée, elle est expansée ainsi que toutes les figures de niveau hiérarchique supérieur dans l'état $Sr.So.U_e$ et ce, jusqu'à atteindre une figure déjà expansée. Toutes les figures expansées sont à leur tour marquées.

Dans la figure 6.11, les figures entourées d'un carré en traits pointillés dans un état en caractère normal sont les figures expansées à l'initialisation. Les figures $Fs1$ et $Fs2$

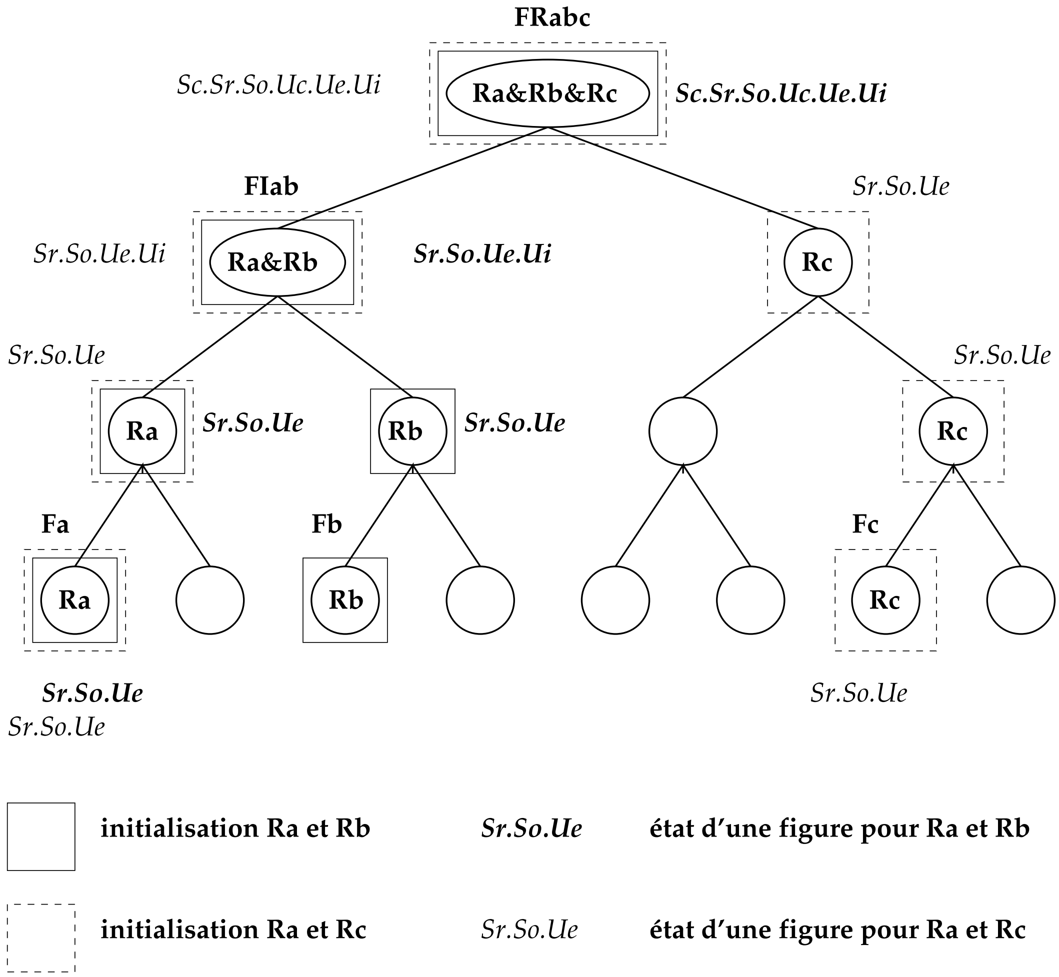


FIG. 6.10 – *Initialisation de la recherche de chemins*

sont expansées lors du parcours quand on passe de Ra vers Rb en passant par S grâce à l'arc uas. De même que l'arc uas, l'arc usb appartient forcément à une figure déjà expansée car il a S comme extrémité terminale.

La deuxième différence par rapport aux algorithmes du chapitre 3 est le principe d'évitement des arcs inutiles pour la recherche en cours. En effet, à cause d'une recherche précédente, les figures de l'arborescence peuvent être dans des états plus complets que ceux souhaités.

Pour éviter de parcourir des arcs inutiles, il faut vérifier que chaque arc parcouru appartient au sous-graphe partiel qui nous intéresse :

- tout arc appartenant à une figure de niveau supérieur à la figure de recherche maximal est ignoré ;
- tout arc appartenant à un ensemble U_c qui ne fait pas partie de la figure de plus haut niveau est ignoré ;
- tout arc appartenant à un ensemble U_i qui ne fait pas partie d'une figure dont le

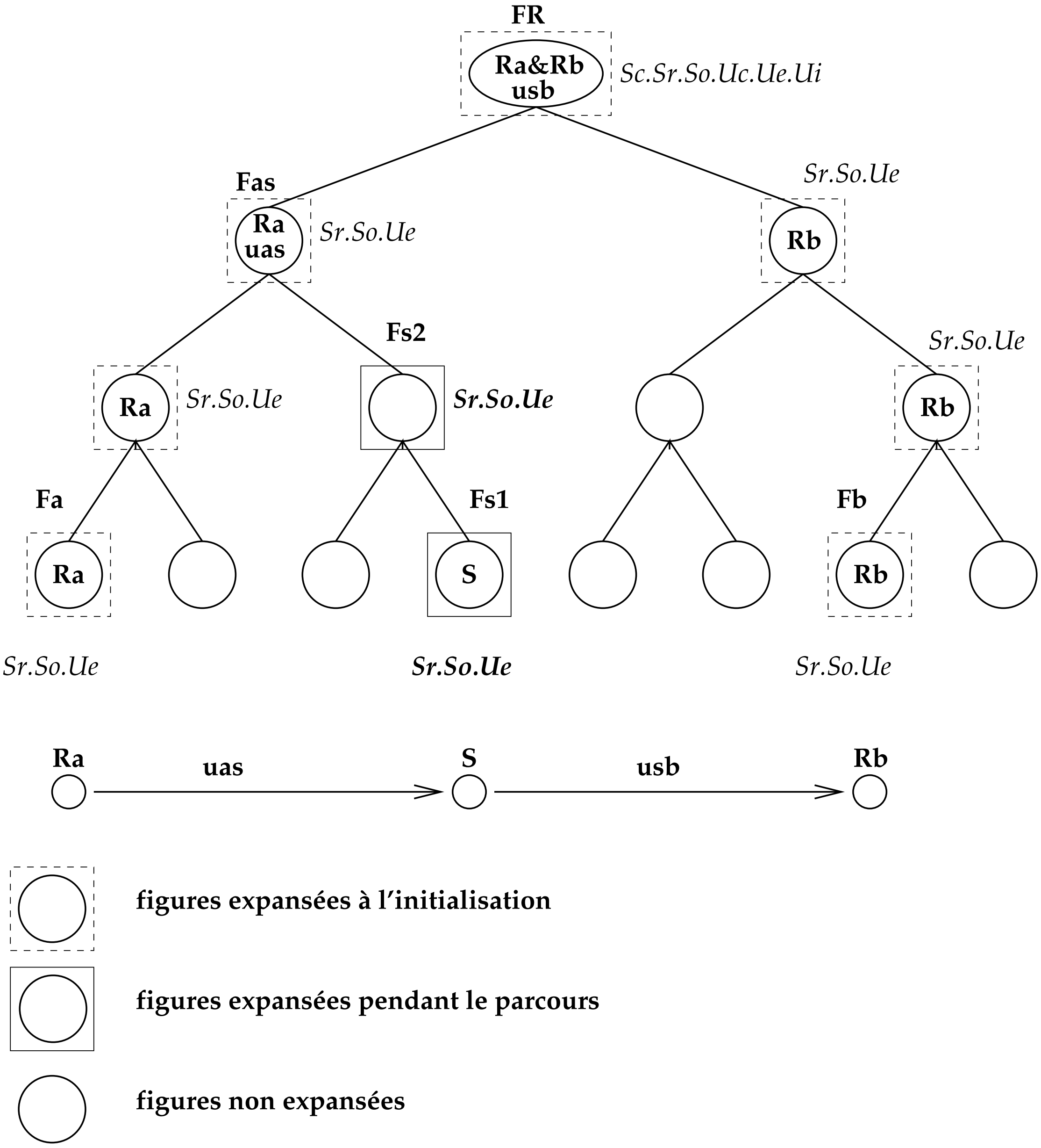


FIG. 6.11 – *Recherche de chemins*

niveau est compris entre le niveau de recherche minimal et le niveau de recherche maximal est ignoré ;

Cette vérification n'a pas seulement pour but d'éviter de parcourir des arcs mais permet surtout d'éviter l'expansion inutile de figures. Le temps de parcours est complètement négligeable à côté des phases d'expansion qui font appel à des lectures de fichiers.

6.5.2 Démonstration de l'algorithme d'expansion dynamique

Il convient de prouver que le sous-graphe partiel, construit pendant la recherche, permet bien de retrouver les bons chemins.

La validité de la phase d'initialisation est évidente. En effet, les ensembles de signaux et d'arcs des figures auxquelles appartiennent les extrémités d'un chemin ainsi que leurs ascendants sont indispensables à la recherche. Le fait que les ensembles U_i ne soient expansés qu'à partir de la figure de niveau de recherche minimal est justifié par la propriété de ces arcs qui ne font partie que des chemins internes à un niveau hiérarchique. Comme il ne peut exister de chemin entre deux sommets avant le niveau de recherche minimal, les arcs des ensembles U_i des figures d'un niveau hiérarchique inférieur à ce niveau ne peuvent pas participer à un chemin entre ces deux sommets. Ainsi, les ensembles U_e suffisent à construire le chemin entre ces deux sommets pour toutes les figures dont le niveau hiérarchique n'est pas compris entre le niveau de recherche maximal et le niveau de recherche minimal.

A tout moment du parcours où l'on se trouve sur un sommet du graphe de causalité, toutes les figures de niveau hiérarchique supérieur à la figure à laquelle appartient ce sommet sont expansées jusqu'à la figure de niveau de recherche maximal et ce avant de parcourir les sommets voisins de ce sommet. A la fin de l'expansion, tous les arcs permettant de passer à tous les voisins de ce sommet et conduisant à un chemin externe sont présents en mémoire. En effet, ils se trouvent nécessairement dans les ensembles U_e des ascendants de la figure à laquelle appartient le sommet considéré. Ainsi, à tout moment du parcours, un sommet connaît tous les arcs qui conduisent à ses successeurs ou à ses prédécesseurs qui permettent de retrouver le chemin recherché.

6.5.3 La recherche avec plusieurs extrémités

La recherche de chemins entre deux événements sur deux signaux est utilisée pour la recherche de chemins parallèles ou pour la recherche du détail au niveau porte d'un chemin particulier.

Avant d'en arriver à étudier les chemins entre deux points, il faut pouvoir identifier les chemins critiques. L'algorithme en largeur d'abord part d'un sommet et recherche dans le graphe de causalité tous les sommets qui ont un chemin critique avec ce sommet. Un seul signal extrémité de chemin est connu pendant la phase d'initialisation. Seule la figure à laquelle appartient ce signal ainsi que ses ascendants doivent être initialisées. Cela ne change rien à l'algorithme. Le niveau de recherche maximal sera le niveau le plus haut où ce signal est référencé. Le niveau de recherche minimal est bien évidemment la figure à laquelle appartient ce signal. L'algorithme d'expansion est le même que celui décrit précédemment.

6.5.4 La limitation de la recherche

Le niveau hiérarchique

Le mécanisme d'expansion dynamique permet de réduire la taille de la mémoire consommée ainsi que le temps de réponse d'un outil interactif.

Dans certains cas il est intéressant de limiter volontairement la recherche. En effet, dans la phase de construction des chemins, les chemins recherchés sont ceux qui passent par des délais du niveau le plus haut de la figure à analyser. Les niveaux de recherche maximal et minimal nécessaires et suffisants sont ceux de la figure de plus haut niveau. La construction des chemins factorisés dans la phase d'analyse temporelle se fait donc avec une limitation des niveaux de recherche maximal et minimal déterminés par la figure de plus haut niveau. Tous les arcs des ensembles U_i des figures de la descendance de la figure de plus haut niveau ne sont jamais présents pendant la phase de construction des chemins.

Dans le cadre de l'outil interactif, le concepteur peut limiter la recherche soit parce qu'il sait par où passe le chemin qu'il recherche soit parce qu'il considère que tous les blocs des niveaux inférieurs ont déjà été validés. Dans ce dernier cas, le concepteur ne s'intéresse qu'à l'optimisation des chemins passant par le niveau qu'il cherche à optimiser et il fixe donc les niveaux de recherche maximal et minimal à un niveau donné.

Toutes ces limitations conduisent à des économies de mémoire et des gains de temps importants, contraintes indispensables pour un outil interactif.

Le sens de parcours

Il existe un deuxième moyen de limiter le nombre d'expansions dans le cas de la recherche d'un chemin entre deux extrémités. C'est le choix du sens de parcours : le sens causal ou le sens anticausal. En effet, le sens de parcours a une grande importance. L'exemple le plus frappant est bien-sûr le cas du signal d'horloge. Le signal d'horloge possède des chemins vers tous les signaux de commande d'écriture dans tous les registres du circuit. La recherche d'un chemin dans le sens causal à partir du connecteur d'horloge conduit à l'expansion de toutes les figures du circuit dans l'état $Sr.So.U_e$ et de la figure de plus haut niveau dans l'état $Sc.Sr.So.U_c.U_e$. En revanche, seule la figure à laquelle appartient une commande d'un registre et ses ascendants sont expansées si la recherche est effectuée à partir de la commande dans le sens anticausal.

Grâce à sa connaissance du circuit, le concepteur peut choisir le sens de parcours afin de limiter le nombre d'expansions.

Le sens par défaut est le sens anticausal car, dans la majorité des cas, c'est celui qui limite le nombre d'expansions de figures.

6.6 La gestion de la mémoire

Dans le cas d'un outil interactif, au fur-et-à-mesure des requêtes, de plus en plus de figures sont expansées. La mémoire de la machine est donc de plus en plus utilisée. Il existe deux possibilités pour éviter le dépassement de capacité. Dans la première possibilité, le système d'exploitation utilise le mécanisme général de mémoire virtuelle en évinçant des pages sur le disque (« swap »). La deuxième possibilité consiste à changer l'état des figures qui ne sont pas impliquées dans la recherche courante et donc de libérer de la mémoire.

6.6.1 Pourquoi libérer de la mémoire

Les deux solutions présentent des avantages et des inconvénients différents. Les inconvénients liés à l'utilisation du « swap » sont connus :

- le système d'exploitation doit posséder un système de « swap » ;
- le « swap » doit être de taille suffisante pour que tous les signaux et tous les arcs soient présents en mémoire ;
- l'utilisation du « swap » conduit à une chute importante des performances.

De plus, l'outil interactif est souvent utilisé sur une station de travail ayant peu de mémoire et peu de « swap » contrairement à l'analyse temporelle et l'analyse de stabilité qui sont exécutées sur un serveur possédant en général beaucoup plus de mémoire. Dans le cadre d'un outil interactif, il est préférable de libérer la mémoire dès qu'elle n'est plus utilisée. Cela présente des avantages importants :

- il n'y a pas d'écriture sur le disque mais seulement des lectures ;
- l'application est portable sur un grand nombre de systèmes et de machines ;
- les algorithmes de libération sont optimisés pour notre application.

Toutes ces raisons nous ont poussé à ajouter un mécanisme de libération explicite de la mémoire.

6.6.2 Les algorithmes de libération

Le nombre maximum d'arcs et de sommets qui peuvent être présents en mémoire est fixé. Il existe une borne maximale pour le nombre de sommets et une borne maximale pour le nombre d'arcs. Tant que le nombre de sommets ou le nombre d'arcs est inférieur aux bornes, des sommets et des arcs sont alloués en fonction de la demande. Dès qu'une borne est atteinte, un mécanisme de sélection d'une figure pour la libération est mis en œuvre. L'objectif de ce mécanisme est de choisir une figure de manière judicieuse. Plusieurs critères sont pris en compte.

Le critère d'éloignement

Le premier critère pris en compte est le critère d'éloignement. Les figures sont classées en fonction de leur éloignement de la figure qui est en cours d'expansion. Les figures de plus basse priorité sont choisies en premier. Il existe cinq niveaux de priorité qui sont illustrés dans le figure 6.12. La figure $figex$ est la figure en cours d'expansion. $fig1$ est la figure de plus haut niveau de l'arborescence où se trouve la figure en cours expansion.

Il est possible d'avoir plusieurs arborescences $fig1$ et $fig2$ indépendantes et simultanément présentes en mémoire.

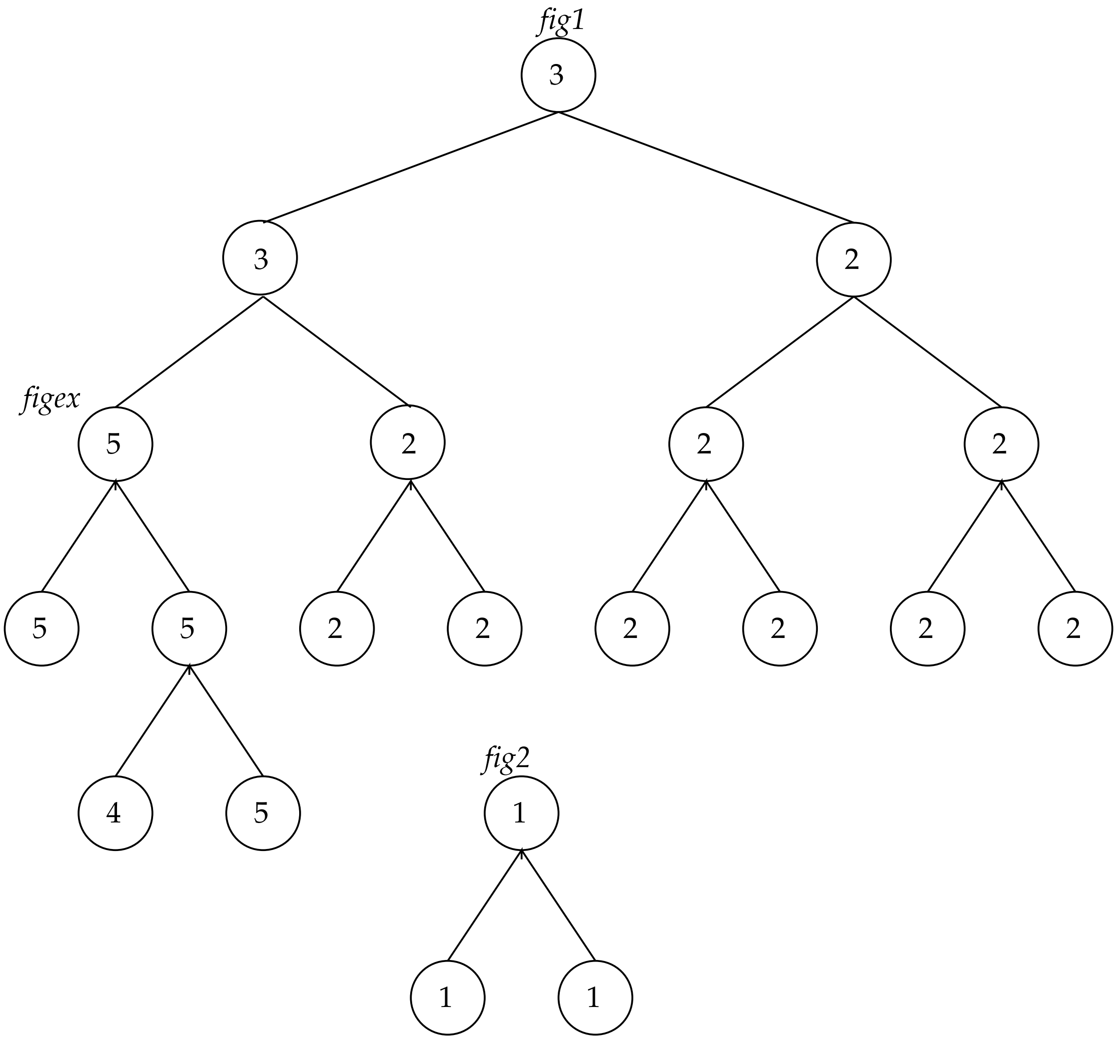


FIG. 6.12 – *Les niveaux de priorité de libération des figures*

– niveau 1 : c'est la priorité des figures appartenant à une autre arborescence ;
– niveau 2 : c'est la priorité des figures qui ne sont ni des ascendants ni des descendants de la figure en cours d'expansion ;

- niveau 3 : c'est la priorité des ascendants de la figure en cours d'expansion ;
- niveau 4 : c'est la priorité des descendants de la figure en cours d'expansion ;
- niveau 5 : c'est la priorité de toutes les figures expansées pendant la recherche en cours.

La victime est la figure de plus basse priorité qui possède des sommets ou des arcs, les figures des autres priorités sont ignorées. Le critère d'éloignement est le premier critère de comparaison entre deux figures. Ce choix est motivé par le principe de localité des chemins. Plus une figure est proche de la figure qui est en cours d'expansion, plus elle a de chance d'être elle même expansée à son tour dans un avenir proche.

Le critère de plus bas niveau

S'il y a plusieurs figures de même niveau de priorité, les figures sélectionnées seront les figures de plus bas niveau hiérarchique. Plus les figures ont un niveau hiérarchique élevé plus elle ont des chances d'être expansées. Elles font intervenir des chemins passant par un nombre de plus en plus grand de figures.

Les critères spécifiques aux arcs et aux sommets

Les seuls sommets libérables sont les sommets de l'ensemble So d'une figure car l'ensemble Sr est l'état minimal d'une figure et l'ensemble Sc doit être présent dans la figure de plus haut niveau hiérarchique. L'ensemble So est l'ensemble de signaux qui possède de loin le plus grand nombre d'éléments. Ce qui justifie que ce soit le seul qui soit assujetti à la libération de ses sommets.

Quand des sommets sont libérés, il faut également libérer les arcs associés, car un arc a besoin de ses extrémités. Il est donc préférable de choisir une figure qui n'a pas ou peu d'arcs à libérer.

De plus, pour la figure sélectionnée, il est préférable de libérer les arcs de l'ensemble U_i car ils ne font intervenir que des chemins internes à cette figure.

La date de la dernière expansion

A chaque fois qu'une figure est expansée, une date lui est attribuée. Le dernier critère prend en compte la figure qui a la date la plus ancienne.

L'échec

Dans le cas des signaux, les figures de niveau de priorité 5 ne peuvent pas être sélectionnées puisque ces figures ont été expansées pendant la recherche courante. Si aucune figure n'est sélectionnée, il y a échec. Aucun ensemble de signal ne peut-être libéré.

A chaque fois qu'une figure est expansée, elle est marquée. Ce marquage consiste à noter les ensembles d'arcs qui sont nécessaires à la recherche. Il correspond à l'état qui est demandé pour l'expansion. Si l'algorithme conduit à la sélection d'une figure expansée et d'un ensemble d'arcs correspondant au marquage de cette figure, c'est qu'il y a échec.

Dans le cas d'un échec, la recherche n'est pas bloquée pour autant. Il y a autorisation de dépassement des bornes maximales de sommets ou d'arcs. Un message de mise en garde prévient le concepteur que les bornes fixées ne sont pas suffisantes.

6.7 Le graphe de causalité réduit hiérarchique

Le graphe de causalité réduit hiérarchique est défini de la même manière que le graphe de causalité hiérarchique.

6.7.1 Les sommets du graphe de causalité réduit hiérarchique

L'ensemble $Sc \cup Sr$ constitue l'ensemble des signaux dont les événements sont les sommets du graphe de causalité réduit. Les signaux des ensembles Sc et Sr sont partagés par le graphe de causalité réduit et par le graphe de causalité. Par contre, l'ensemble So ne participe pas au graphe de causalité réduit.

6.7.2 Les arcs du graphe de causalité réduit hiérarchique

Les arcs du graphe de causalité réduit sont différents de ceux du graphe de causalité. Ils représentent des chemins factorisés et non pas des délais élémentaires. Nous noterons dans la suite :

- Uc l'ensemble des arcs du graphe de causalité ;
- Ur l'ensemble des arcs du graphe de causalité réduit.

Comme pour le graphe de causalité, il existe trois ensembles d'arcs dans le graphe de causalité :

- l'ensemble Ur_c des arcs attachés à au moins un événement sur un connecteur ;
- l'ensemble Ur_e des arcs n'ayant aucun événement sur un connecteur comme extrémité et faisant partie d'au moins un chemin ayant comme extrémité un connecteur. Ils appartiennent à un chemin externe ;
- l'ensemble Ur_i des arcs n'ayant aucun événement sur un connecteur comme extrémité et ne faisant partie d'aucun chemin ayant comme extrémité un connecteur, ils appartiennent seulement à des chemins internes.

Les arcs du graphe de causalité seront notés : Uc_c, Uc_e, Uc_i.

Le graphe de causalité réduit de F est constitué des sous-graphes partiels engendrés par les ensembles de signaux Sr et les ensembles d'arcs Ur_e, Ur_i de toutes les figures des blocs instanciés dans F jusqu'aux figures des blocs feuilles, ainsi que des nouveaux ensembles de signaux Sc et Sr et des nouveaux ensembles d'arcs Ur_c, Ur_e et Ur_i appartenant à la figure F.

6.7.3 Construction du graphe de causalité réduit

Le graphe de causalité réduit est également à plat. L'algorithme de factorisation défini au chapitre 5 s'applique donc de la même manière.

Il existe cependant une contrainte sur le choix des signaux de factorisation. Lors de la construction du graphe de causalité réduit d'une figure non-terminale, les points de factorisation sont choisis parmi les signaux de factorisation des instances de cette figure et les signaux de l'ensemble So appartenant à cette figure. Cette contrainte est due au fait que les signaux des ensembles So des instances ne changent pas de catégorie d'un niveau hiérarchique à l'autre.

Il existe une deuxième contrainte liée à la construction hiérarchique des chemins factorisés arrivant sur des connecteurs.

Propriétés des connecteurs internes et externes

Les connecteurs internes et externes d'une figure ne sont vus que par cette figure car contrairement aux autres signaux, leurs propriétés sont propres à la figure à laquelle ils appartiennent. Les arcs ayant pour extrémité un événement sur un connecteur interne appartiennent aux ensembles d'arcs Uc_c ou Ur_c puisque leur délai doit être recalculé aux niveaux hiérarchiques supérieurs.

Dans la figure 6.7, le délai entre $SF11$ et $SF21$ n'est pas le même dans la figure temporelle *Père* et la figure temporelle *Grand-Père*. Ce délai doit pouvoir être modifié lorsque la figure temporelle de *Grand-Père* est générée. Cela signifie qu'un arc du graphe de causalité réduit ne peut pas passer par un connecteur interne (sauf si ce connecteur en est l'extrémité) puisque son délai n'est défini qu'au niveau hiérarchique auquel appartient ce connecteur.

Les connecteurs internes et externes sont donc des extrémités de chemins factorisés dans le graphe de causalité réduit car les chemins qui passent par ces signaux seront remis en cause aux niveaux hiérarchiques supérieurs. Si un arc du graphe de causalité réduit passe par un signal de ce type sans en être extrémité, son délai ne pourra plus être modifié au niveau hiérarchique supérieur et sera donc faux. Afin de déterminer le délai des chemins qui passent par de tels signaux, il faut considérer ces signaux comme des points d'arrêt et de départ de la recherche de chemins pour la construction du graphe de causalité réduit. Ils ne faut pas construire d'arc dans le graphe de causalité réduit qui passe par un connecteur externe sans être une extrémité de l'arc considéré.

Par conséquent, nous sommes obligés de distinguer les ensembles de départ et d'arrivée de la recherche de chemins, pour le parcours en largeur d'abord, dans le cas de la construction du graphe de causalité réduit et dans le cas de la recherche de chemins temporels dans un outil interactif de visualisation de chemins, car dans ce cas, les connecteurs ne sont pas des points d'arrêt de la recherche puisqu'un chemin temporel peut passer à travers un connecteur (chapitre 3).

Il y a donc une nouveauté par rapport au graphe obtenu avec une figure à plat définie au chapitre 5 : la méthodologie hiérarchique impose que les connecteurs internes et les connecteurs externes soient des extrémités de chemins factorisés. Cependant, cela ne pose pas de problème particulier car, dans la notion de graphe de causalité réduit, les chemins ne sont pas complètement explicites. Ils doivent de toute façon être reconstruits. La seule conséquence est que le nombre d'arcs peut être légèrement plus important que celui qui serait obtenu avec une approche à plat. Cependant, cette augmentation est linéaire par rapport au nombre de connecteurs dans une figure. De plus, les connecteurs sont en général de bons points de factorisation.

Étant donné que, dans une approche hiérarchique, les chemins les plus courts et les plus longs d'un circuit ne peuvent pas être représentés explicitement, ces chemins sont forcément représentés par un graphe de causalité réduit et non pas par un graphe de chemins critiques.

6.7.4 Le fichier « .ttx »

Le fichier « .ttx » représente le graphe de causalité réduit. Le fichier ayant pour nom « $F.ttx$ » représente les ensembles de signaux Sc, Sr appartenant ou référencés dans la figure F et les ensembles d'arcs Ur_c, Ur_e et Ur_i appartenant à la figure F. L'ensemble des fichiers « .ttx » de tous les blocs et sous-blocs appartenant à l'arborescence décrivant un circuit permettent de construire en mémoire le graphe de causalité réduit de ce circuit.

Dans chacun des deux fichiers « $F.dtx$ » et « $F.ttx$ » on trouve la liste des filles de la figure F. Le fichier contient, pour chacune des filles, le nom de modèle et le nom d'instance de la fille considérée dans la figure représentée par ce fichier.

La date des fichiers doit être la même dans les deux fichiers « .dtx » et « .ttx ». Cela permet de vérifier que les deux fichiers ont été générés avec la même exécution de l'outil d'analyse temporelle, ce qui assure que les informations trouvées dans chacun des deux fichiers sont bien cohérentes entre elles.

Le choix de séparer la représentation des deux graphes dans deux fichiers différents se justifie par leur usage distinct. Le graphe de causalité sert pour l'analyse de stabilité et la recherche de chemins critiques, tandis que le graphe de causalité réduit sert à la recherche du détail d'un chemin ou des chemins parallèles.

Les ensembles de signaux Sc, Sr sont présentés dans les deux fichiers ce qui demeure peu coûteux puisque ces ensembles ont un petit nombre d'éléments.

La syntaxe des fichiers « .dtx » et « .ttx » est donnée dans l'annexe B.

6.7.5 Le recherche de chemins dans le graphe de causalité réduit

L'état d'une figure

L'état d'une figure différencie les ensembles d'arcs du graphe de causalité et ceux du graphe de causalité réduit. L'état complet de la figure est donc :

$Sc.Sr.So.Uc_c.Uc_e.Uc_i.Ur_c.Ur_e.Ur_i$

Les signaux de l'ensemble So d'une figure n'appartiennent pas au graphe de causalité réduit. Il n'y a donc expansion de l'état So que si la recherche est réalisée sur le graphe de causalité. Pour les autres états, le principe est identique que la recherche soit effectuée sur le graphe de causalité ou sur le graphe de causalité réduit. Dans le cas d'une recherche dans le graphe de causalité, ce sont les ensembles d'arcs Uc_c, Uc_e et Uc_i qui seront expansés. Dans le cas d'une recherche dans le graphe de causalité réduit, ce sont ceux des ensembles Ur_c, Ur_e et Ur_i.

La recherche de chemins

Le graphe de causalité réduit est utilisé pour la recherche de chemins critiques. Cette recherche peut être réalisée en spécifiant une seule extrémité de départ. Nous avons vu que cela augmente le nombre d'expansion par rapport à la recherche en fixant les deux extrémités. La recherche de chemins entre deux événements sur deux signaux est utilisée pour la recherche de chemins parallèles et la recherche du détail au niveau porte d'un chemin. Elle s'applique plus spécifiquement aux recherches de chemins dans le graphe de causalité. La connaissance pendant la phase d'initialisation des deux extrémités permet de limiter le nombre d'expansions. Le fait que cela s'applique à la recherche dans le graphe de causalité est d'autant plus intéressant que c'est le graphe qui prend le plus de place en mémoire et qui est donc le plus critique en temps de réponse pour un outil interactif.

De plus, l'outil interactif peut choisir automatiquement de limiter la recherche. En effet, lorsqu'un chemin critique est déterminé à partir du graphe de causalité réduit, les niveaux de recherche maximal et minimal sont déterminés puisqu'ils correspondent au niveau le plus haut de la figure à laquelle appartient au moins un arc du graphe de causalité réduit par où passe ce chemin. Ce niveau est utilisé par l'algorithme de recherche du détail de chemin dans le graphe de causalité comme niveau de recherche maximal et minimal. Ce qui a pour effet de limiter le nombre d'expansion du graphe de causalité réduit.

6.8 Conclusion

Bien que la recherche de chemins travaille sur une représentation à plat du graphe de causalité, nous avons défini une méthode de représentation hiérarchique multi-niveaux des graphes de causalité et de causalité réduit. Cette méthode permet d'utiliser le découpage structurel de la phase de conception afin de ne représenter explicitement en mémoire que des sous-graphes partiels. Le sous-graphe partiel nécessaire à la recherche d'un chemin est construit grâce à un mécanisme d'expansion dynamique des figures. Un système de cache permet la libération de mémoire en supprimant des sommets et des arcs des figures qui ne servent pas à la recherche. Toutes les phases de la vérification temporelle utilisent ce système de cache logiciel afin de limiter les ressources nécessaires à leur fonctionnement. Ainsi le temps de réponse de l'outil interactif de visualisation des chemins est beaucoup plus court que s'il fallait lire tous les fichiers pour rechercher un seul chemin. Cette méthode évite également l'utilisation du « swap » de la machine.

Notre méthode a l'avantage de conserver une vue à plat des différents graphes. De ce fait, tous les concepts de base de l'analyse temporelle ainsi que le principe de la factorisation de chemin restent inchangés.

A partir des principes qui ont été présentés, une structure de données a été définie. Elle sert de plate-forme commune à tous les outils de la chaîne de vérification temporelle.

Dans le chapitre suivant nous présenterons en détail la méthode de construction de cette structure ainsi que l'outil interactif permettant de visualiser les chemins.

CHAPITRE 7

Les outils de la chaîne de vérification temporelle

Ce chapitre présente les outils logiciels construits sur la structure temporelle hiérarchique multi-niveaux présentée dans le chapitre 6. La base de données ***Ttv*** est composée d'une structure de données et d'une bibliothèque de fonctions permettant l'accès et la modification de cette base de données, l'utilisation des algorithmes de recherche de chemins ainsi que la gestion du cache logiciel.

Cette base de données est utilisée par les outils ***HiTas*** et ***Xtas***. ***HiTas*** est l'outil d'analyse temporelle hiérarchique multi-niveaux tandis que ***Xtas*** est l'outil interactif de visualisation des chemins temporels au niveau porte.

7.1 L'environnement logiciel

La chaîne de CAO/VLSI ***ALLIANCE*** [BBG94, GP92] propose un ensemble de bibliothèques de fonctions et de représentations des vues structurelle, physique et comportementale d'un circuit. Tous les outils de la chaîne ***ALLIANCE*** utilisent ces plates-formes de développement communes.

L'analyse temporelle utilise la bibliothèque de représentation des net-lists hiérarchique [Pét94]. Pour les besoins de l'analyse temporelle, nous avons enrichi cette vue par la représentation des réseaux RC afin de modéliser les effets parasites introduits par les interconnexions.

Une net-list hiérarchique est un ensemble de composants. Chaque composant (ou modèle) représente un niveau hiérarchique ; un composant est décrit comme une interconnexion d'instances d'autres composants. Chaque composant possède un ensemble de connecteurs qui définissent son interface. Les instances sont représentées par des boîtes noires dont on décrit l'interface mais pas le contenu. Les connecteurs d'un composant sont appelés connecteurs externes. Les connecteurs des instances sont appelés connecteurs internes.

Les interconnexions entre les transistors ou les instances sont modélisées par des

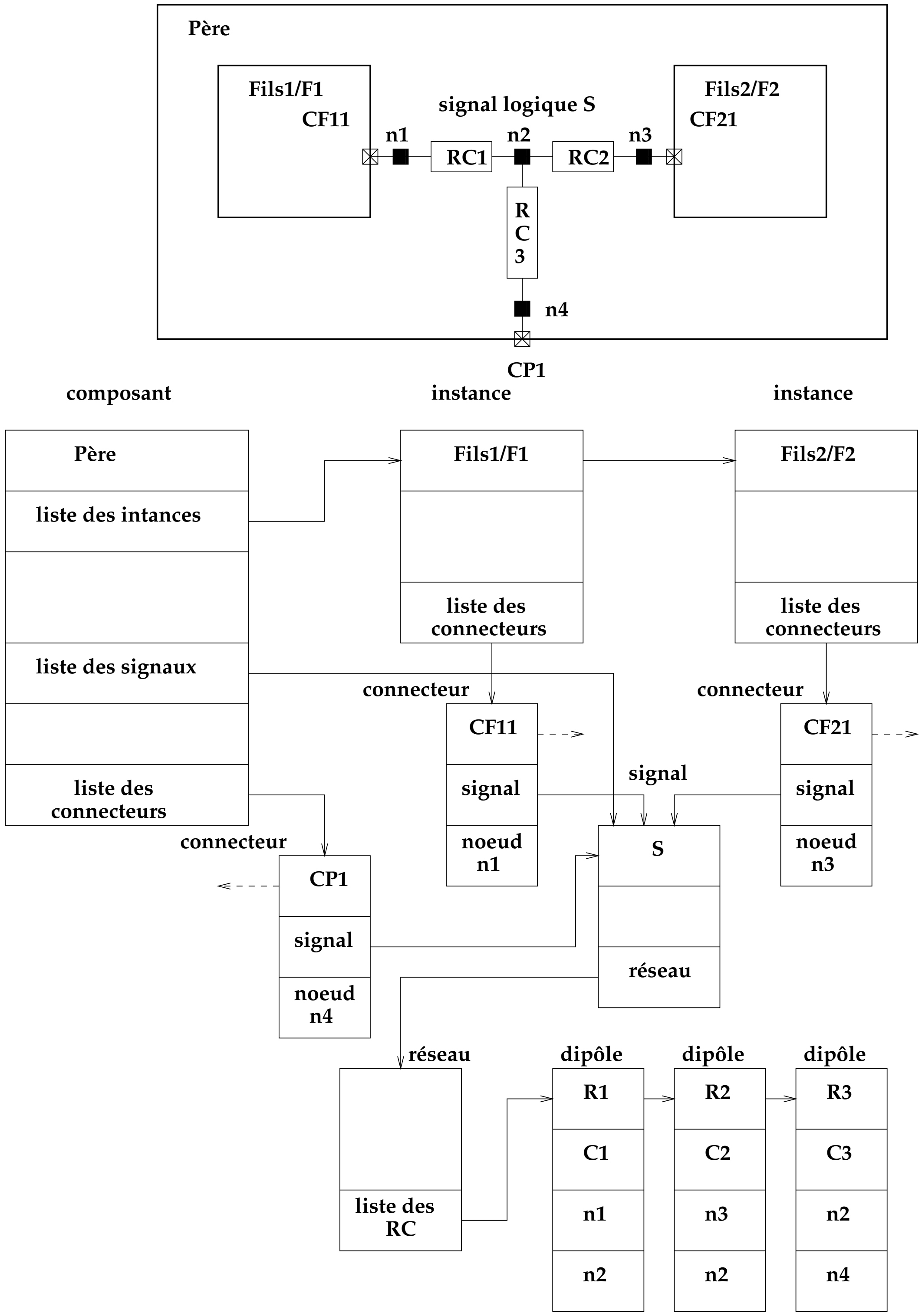


FIG. 7.1 – *Représentation des net-lists avec des réseaux d'interconnexion RC*

signaux logiques : À chaque connecteur est associé un signal logique qui représente l'interconnexion à laquelle il est connecté.

Dans notre représentation, les réseaux RC ne sont pas modélisés par des composants de la net-list. Les résistances et capacités sont des attributs des signaux logiques. A chaque signal logique peut être associé un réseau RC ou une simple capacité.

Un réseau RC est un ensemble de dipôles RC. Chaque extrémité d'un dipôle est connectée à un nœud logique.

Ainsi notre représentation en définissant les capacités et résistances d'interconnexions comme des attributs des signaux logiques facilite, la tâche des outils de vérification car la notion de signal logique est importante. En effet, elle définit la connectique entre les blocs alors que les réseaux RC ne sont que des éléments parasites.

Un exemple simplifié de représentation de net-list hiérarchique avec réseau RC est donné dans la figure 7.1.

La représentation logique d'un circuit est obtenue à partir de fichiers de net-lists décrits dans les formats standard : SPICE, COMPASS, EDIF, ainsi que dans le format propre de la chaîne ***ALLIANCE***. Les « parsers » de ces fichiers sont communs à tous les outils de la chaîne.

7.2 La structure de données hiérarchique multi-niveaux *Ttv*

La structure de donnée ***Ttv*** est un arbre qui reproduit la hiérarchie de conception du circuit. Il n'y a pas de distinction entre modèle et instance : toutes les *figures* présentes en mémoire représentent un nœud de l'arbre d'instanciation. Il y a donc un seul objet du type *figure*. Les trois autres types d'objets sont les *signaux*, les *événements* représentant les sommets du graphe de causalité et les *arcs* du graphe de causalité (caractérisés par les délais élémentaires à travers les portes) ou les *arcs* du graphe de causalité réduit (caractérisé par les délais des chemins factorisés). La figure 7.2 présente les structures de données représentant la vue temporelle d'un circuit. L'annexe A présente les structures de données en langage C.

7.2.1 La figure

Une figure correspond à une instance. Elle est identifiée par deux noms : son nom de modèle et son nom d'instance. Une figure contient la liste des figures qu'elle instancie ainsi qu'une liste de signaux et une liste de délais et de chemins factorisés. Elle contient de plus un descripteur d'état, le cas échéant un pointeur sur la figure de niveau hiérarchique supérieur et des informations diverses. Ces informations regroupent la date de création de la figure, la version de l'outil et de la technologie, le front d'entrée utilisé pour calculer les temps de propagation sur les connecteurs d'entrée, la capacité appliquée sur les connecteurs de sortie ainsi que le compteur d'activité de la figure pour la gestion du cache logiciel.

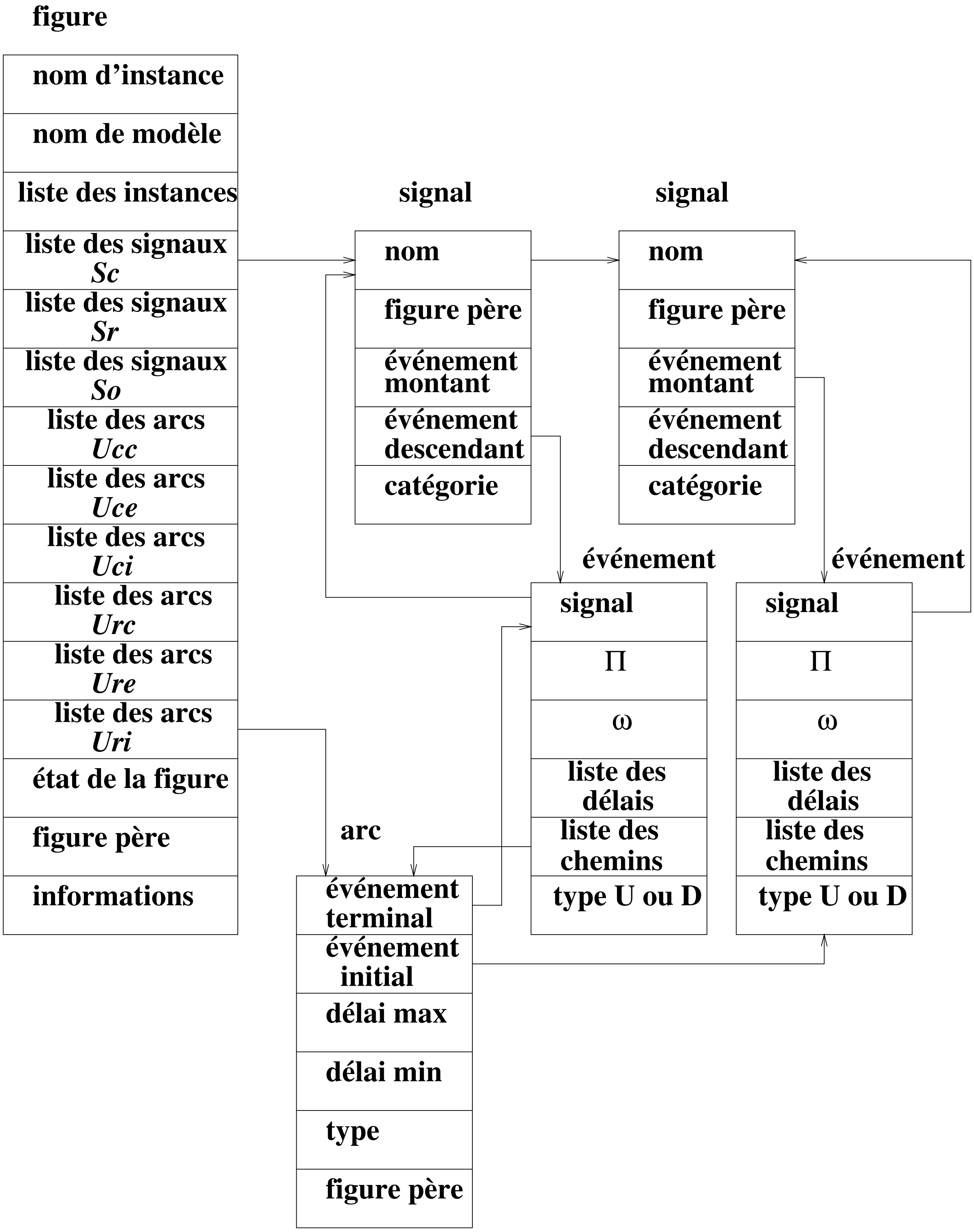


FIG. 7.2 – *Les objets d'une figure **Ttv***

7.2.2 Les signaux

Un même signal logique peut traverser plusieurs niveaux hiérarchiques (c.-à-d. plusieurs figures) et donc posséder plusieurs noms. Le nom d'un signal est défini localement dans la figure à laquelle il appartient, pointée par le champ « figure père » du signal.

Deux événements sont associés à chaque signal : l'événement correspondant à la transition montante et l'événement correspondant à la transition descendante.

Chaque signal appartient à une catégorie donnée. Il existe huit catégories différentes de signaux :

- les connecteurs externes ;
- les connecteurs internes ;
- les registres à écriture sur état ;
- les registres à écriture sur front ;
- les bus préchargés ;
- les commandes d'écriture sur les registres et les bus préchargés ;
- les signaux de factorisation ;
- les autres signaux.

Les connecteurs internes et externes appartiennent à l'ensemble Sc ; les registres, les bus préchargés, les commandes d'écriture et les signaux de factorisation appartiennent à l'ensemble Sr ; les autres signaux appartiennent à l'ensemble So de la figure temporelle.

7.2.3 Les arcs

Les chemins et les délais élémentaires sont représentés par la même structure. Un arc est orienté de son événement initial vers son événement terminal. Il est caractérisé par son délai maximal et son délai minimal représentant les deux bornes du temps de propagation entre l'événement terminal et l'événement initial. Le type de l'arc définit l'ensemble auquel appartient cet arc. Le pointeur sur la « figure père » définit la figure à laquelle appartient cet arc.

7.2.4 Les événements

La liste des chemins et des délais associés à un événement est donnée pour les prédécesseurs d'un événement. Les graphes de causalité et de causalité réduit sont donc représentés dans le sens anticausal. Si une recherche de chemin doit être réalisée dans le sens causal, la liste des successeurs est construite, pendant la phase d'expansion, pour tous les événements de toutes les figures concernées par cette recherche. La liste des arcs représentant les chemins factorisés n'existe que dans le cas d'événements sur des signaux de stabilité ou des signaux de factorisation.

π et ω sont les champs qui permettent de conserver les paramètres des algorithmes de parcours en largeur d'abord et profondeur d'abord (chapitre 3).

Le type définit le sens de transition de l'événement.

7.3 L'analyse temporelle à plat avec *Tas*

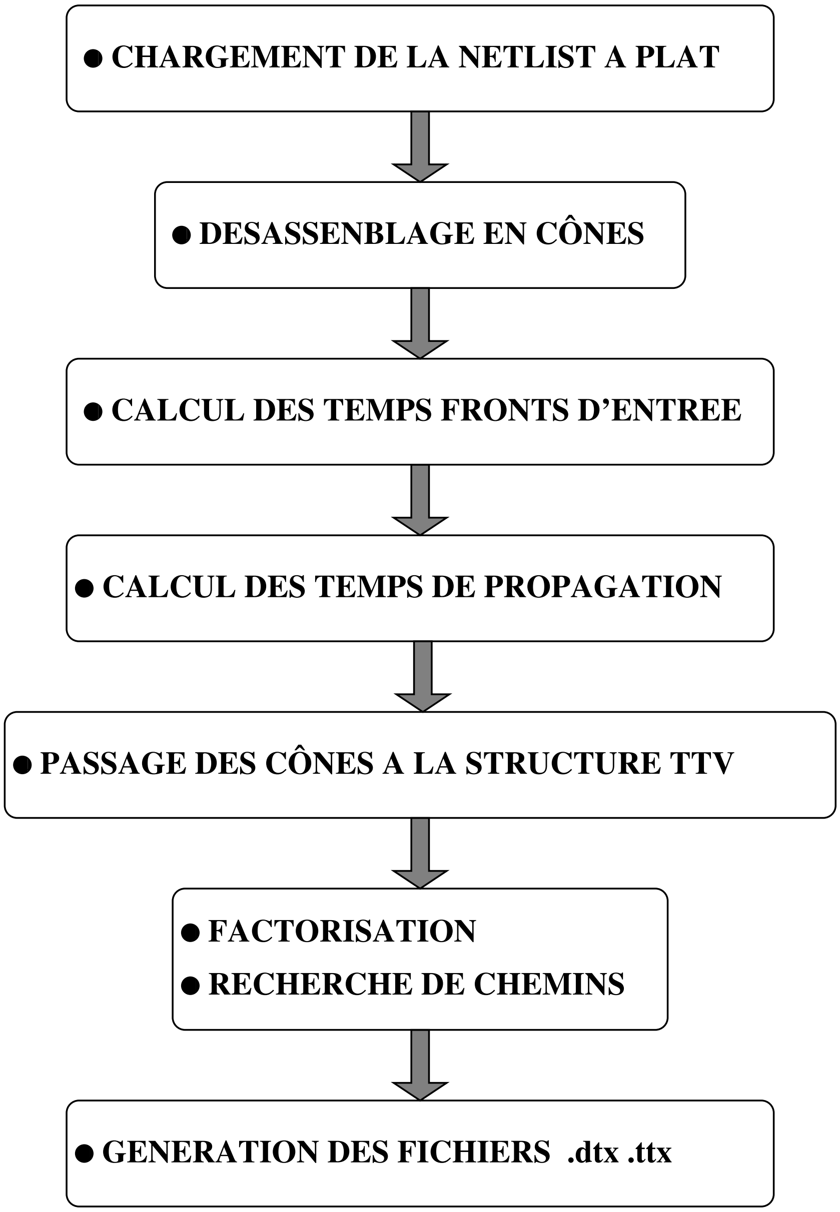


FIG. 7.3 – *Architecture de l'analyseur temporel à plat* ***Tas***

L'outil ***Tas*** d'analyse temporelle à plat au niveau transistor a été modifié pour qu'il s'adapte à l'approche hiérarchique. Il a tout d'abord été porté sur le nouvel outil de désassemblage ***Yagle*** et nous avons enrichi l'outil ***Tas*** en ajoutant le calcul des temps minimaux.

Le calcul des temps minimaux est similaire à celui des temps maximaux si ce n'est que les conditions sur les entrées des cônes sont évaluées dans le cas le plus favorable.

L'évolution la plus importante de ***Tas*** a consisté à convertir la représentation en cônes générée par l'outil de désassemblage ***Yagle*** vers la structure de données ***Ttv***.

L'utilisation de la structure ***Ttv*** a permis d'introduire dans ***Tas*** à plat la technique de factorisation de chemins.

7.4 L'analyse temporelle hiérarchique avec *HiTas*

Le problème posé ici consiste à construire la figure temporelle d'un bloc père, sachant qu'on dispose des figures temporelles associées à chacun des blocs fils instanciés dans le bloc père. On utilise en entrée une net-list dont les instances sont des blocs qui ont déjà subi une phase d'analyse temporelle soit avec ***Tas*** pour les blocs terminaux soit avec ***HiTas*** pour les blocs non-terminaux (c.-à-d. des blocs pour lesquels on dispose des fichiers .dtx et .ttx). Les net-lists décrivant la structure interne des blocs instanciés ne sont pas nécessaires.

La figure 7.4 présente la méthode d'utilisation des outils ***Tas*** et ***HiTas***.

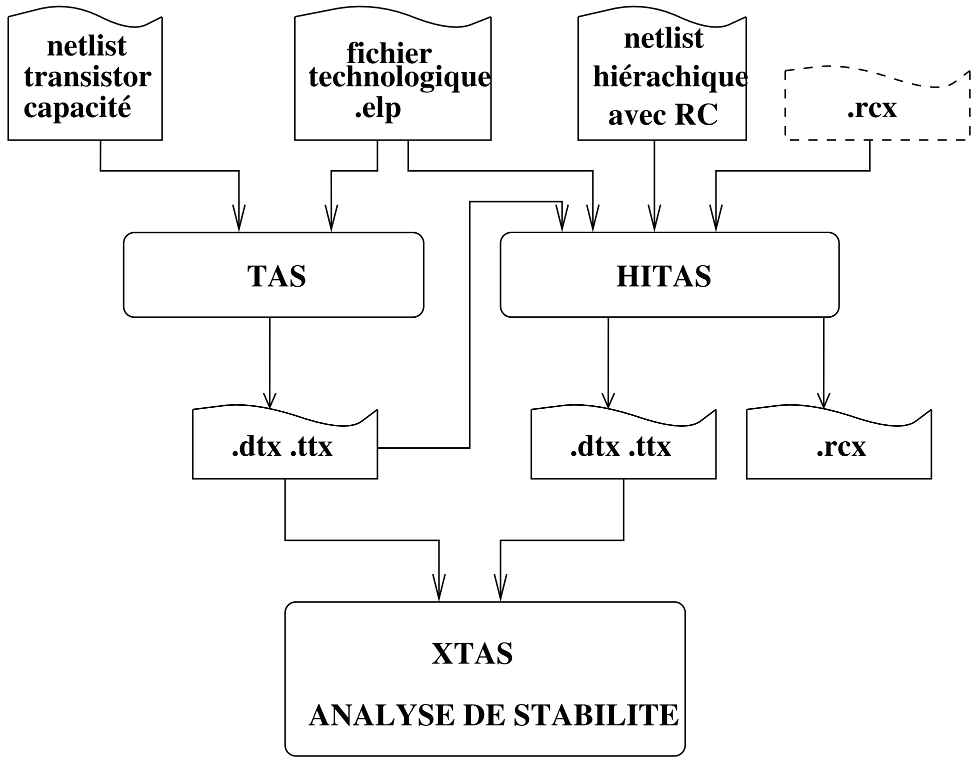


FIG. 7.4 – *Utilisation de* ***Tas*** *et* ***HiTas***

La figure 7.5 présente l'architecture interne de l'analyseur temporel hiérarchique

HiTas.

7.4.1 La construction de la figure

La première étape consiste à créer en mémoire la figure temporelle du bloc père. Puis, pour chacune des instances de la net-list, il convient de construire en mémoire une figure temporelle à partir des fichiers « .dtx » et « .ttx ».

Les figures représentant les blocs fils sont mises dans l'état $Sc.Sr.So.Uc_c.Ur_c$, qui représente explicitement l'ensemble des arcs dont les délais doivent être recalculés au niveau de la figure père. Les arcs concernés ont comme extrémité un connecteur dans les figures filles. Les figures de niveaux hiérarchiques inférieurs restent dans l'état minimum Sr. Elles ne contiennent alors que les signaux de stabilité.

7.4.2 L'orientation des connecteurs

Dans une net-list extraite, on ne dispose généralement pas des informations concernant l'orientation des connecteurs. La deuxième étape oriente les connecteurs afin de déterminer le sens de propagation de l'information pour la création des arcs. L'orientation des connecteurs d'une figure père est déduite de l'orientation des connecteurs des figures filles. Pour les figures feuilles de la hiérarchie construites grâce à ***Tas***, les connecteurs sont orientés par l'outil de désassemblage ***Yagle***.

7.4.3 La construction des signaux temporels

La création des signaux respecte les principes définis dans le chapitre 3 pour l'analyse à plat et le chapitre 6 pour l'analyse hiérarchique. Cette phase vérifie si un réseau RC est associé à un signal logique, puisque le nombre de signaux créés dépendent de l'existence de ce réseau RC.

7.4.4 Le calcul des délais des portes

L'interconnexion des blocs fils dans une figure père conduit à une modification des réseaux RC associés aux signaux auxquels sont connectés les connecteurs des blocs. Ces réseaux RC conduisent à une modification des délais associés aux portes attachées aux connecteurs. Ces délais doivent donc être recalculés.

La première étape consiste à calculer, pour chaque signal appartenant à la figure père, sa capacité totale et son front. Ce calcul s'effectue grâce à des paramètres électriques associés à chaque connecteur interne et externe. La capacité associée à un signal est bien sûr la somme de toutes les capacités de tous les connecteurs connectés à ce signal. Ces paramètres sont sauvegardés dans les fichiers « .dtx » et « .ttx ».

Le front est calculé grâce à un modèle RC présenté dans la thèse d'Amjad Hajjar [Haj92]. Dans le cas où il y a des réseaux RC, le front est calculé en ajoutant l'influence du réseau RC selon la méthode du délai d'Elmore décrite dans la littérature [RPH83].

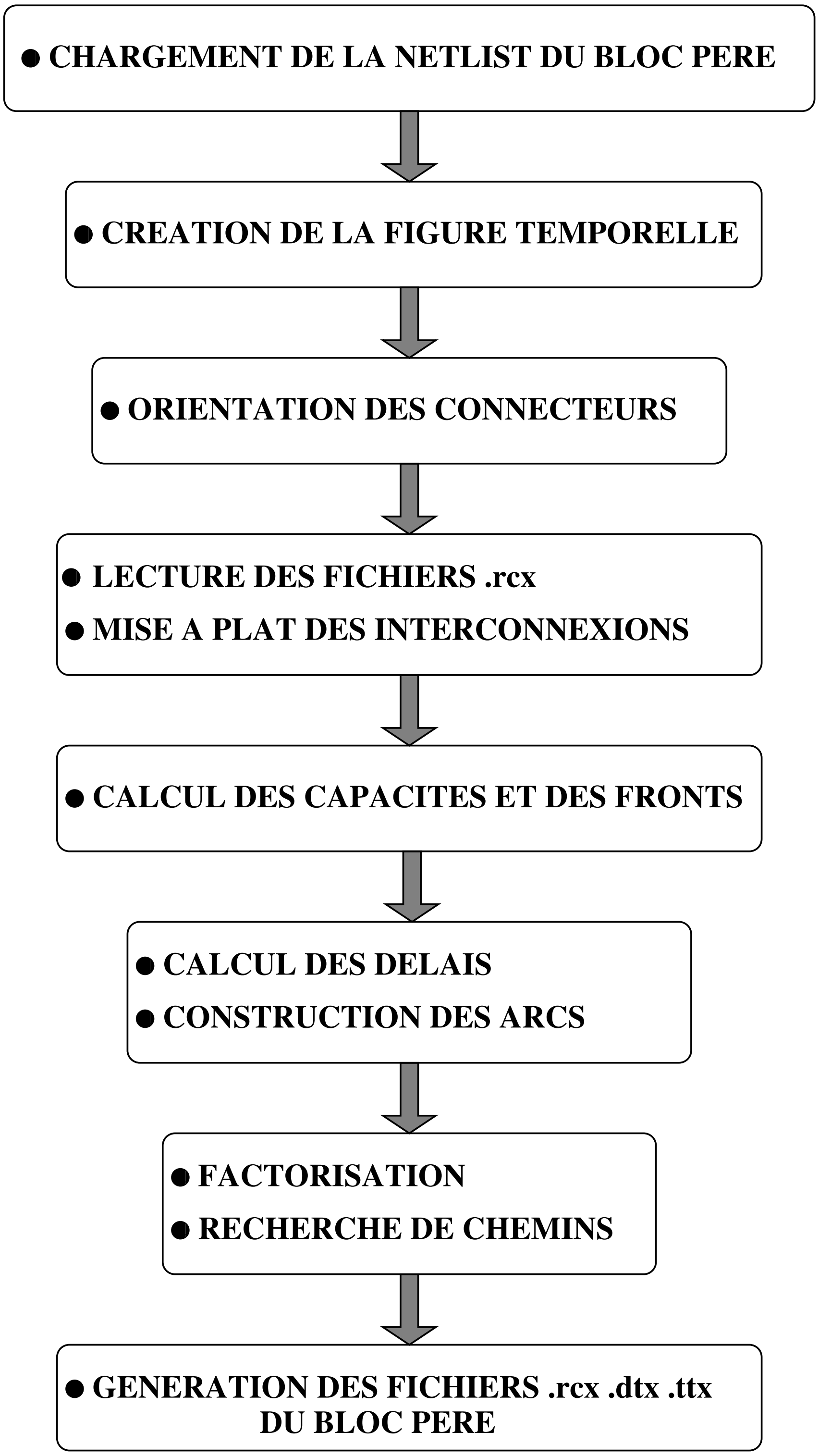


FIG. 7.5 – *Architecture de l'analyseur temporel hiérarchique* ***HiTas***

Les délais calculés durant l'analyse temporelle hiérarchique d'une figure sont obtenus à partir des délais des figures filles. Ce sont les délais représentés par les arcs des ensembles Uc_c et Ur_c des figures filles. Les délais de ces arcs sont caractérisés par un paramètre de front S et un paramètre de capacité R présentés dans le chapitre 2. Le paramètre de capacité R permet de corriger le délai en fonction de la variation de capacité de sortie ; le paramètre de front S permet quand à lui de corriger le délai en fonction de la variation du front d'entrée, grâce à l'équation :

$$TP = TP_0 + R \times (C - C0) + S \times (F - F0) \tag{7.1}$$

TP est donc le nouveau délai calculé au niveau hiérarchique supérieur lorsque la capacité attachée au connecteur de sortie passe de $C0$ à C et le front de $F0$ à F. Les délais sont calculés par un développement limité au premier ordre qui permet de corriger ces délais à tous les niveaux hiérarchiques.

Problème du chemin critique

La génération des chemins critiques d'un niveau hiérarchique à partir des chemins des instances ne garantit pas que les chemins trouvés soient forcément les chemins critiques.

Prenons l'exemple d'un chemin critique entre deux connecteurs i et j d'une instance F (figure 7.6).

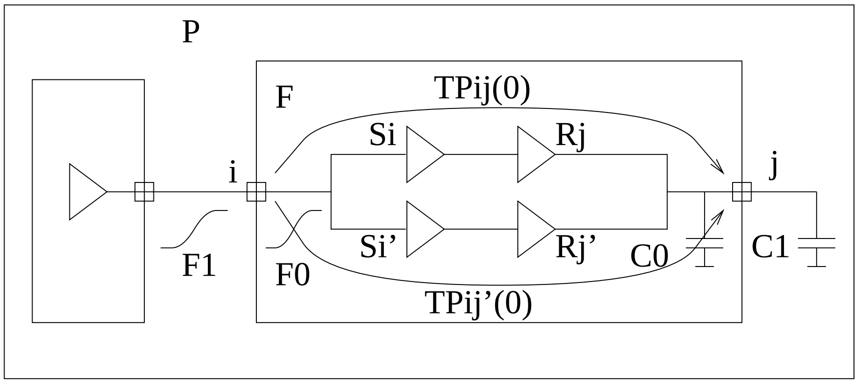


FIG. 7.6 – *Changement du chemin critique entre deux niveaux hiérarchiques*

Au niveau de l'instance, il a un temps égal à $TPij(0)$, calculé avec une capacité $C0$ présente sur le connecteur de sortie et un front $F0$ appliqué sur le connecteur d'entrée. Le temps de propagation au niveau hiérarchique de P s'écrit :

$$TPij = TPij(0) + Rj \times (C1 - C0) + Si \times (F1 - F0) \tag{7.2}$$

où, Si dépend de la porte attaquée par le connecteur d'entrée i et Rj dépend de la porte reliée au connecteur de sortie j ; $C1$ et $F1$ sont les valeurs de front et de capacité au niveau hiérarchique de P.

Supposons qu'il existe un deuxième chemin ayant les mêmes extrémités et un temps de propagation tel que $TPij'(0) < TPij(0)$ et que les deux paramètres Si' et Rj' diffèrent de Si et Rj avec $Si' > Si$ et $Rj' > Rj$. Ce cas peut apparaître lorsque ce deuxième chemin ne passe pas par la même première porte du premier chemin et passe par une entrée de la dernière porte différente de celle du premier chemin.

Le temps de propagation de ce deuxième chemin, au niveau hiérarchique supérieur, s'écrit alors :

$$TPij' = TPij'(0) + Rj' \times (C1 - C0) + Si' \times (F1 - F0) \tag{7.3}$$

Au niveau hiérarchique supérieur, la capacité $C1$ et le front $F1$ peuvent être très différents des valeurs $C0$ et $F0$. Avec les nouvelles valeurs $C1$ et $F1$, $TPij$ peut être inférieur à $TPij'$. Lorsque seuls les chemins critiques des instances sont conservés, le délai $TPij'$ ne peut plus être calculé alors qu'il constitue le chemin critique au niveau hiérarchique supérieur.

Une première solution consiste à considérer la plus grande résistance R et le plus grand paramètre S dans le cas ou les variations de capacité et de front sont positives et la plus petite résistance R et le plus petit paramètre S dans le cas ou les variations de capacité et de front sont négatives, indépendamment de la première porte et de l'entrée qui commute sur la dernière porte. Le chemin sera alors un majorant du chemin critique.

Pour retrouver les vrais chemins critiques, une seconde possibilité consiste à rechercher à partir du graphe de causalité et non pas du graphe de causalité réduit.

Les deux possibilités sont offertes par l'outil ***HiTas***, cependant la construction du graphe de causalité réduit à partir des graphes de causalité réduits de la descendance d'une figure est beaucoup plus rapide qu'à partir des graphes de causalité non réduits.

7.4.5 Le traitement hiérarchique des réseaux RC

Le calcul des délais à travers les interconnexions

Dans ***HiTas***, les délais dans les interconnexions sont calculés avec le délai d'Elmore. Un arc représentant un délai à travers un réseau RC est caractérisé par le délai d'Elmore [Elm48, JR81, RPH83] entre les deux nœuds correspondant aux signaux temporels des événements extrémités de cet arc.

Si le réseau RC présente des boucles, deux possibilités sont offertes au concepteur : soit ***HiTas*** ne calcule pas de délai et considère qu'il n'y pas de réseau ; soit les boucles sont cassées. Dans les réseaux RC représentant les interconnexions entre les blocs, les boucles se trouvent généralement sur les signaux d'horloge. Ce sont des signaux présentant une topologie maillée afin de rendre négligeable le déphasage du front d'horloge sur les commandes d'écriture dans les registres. L'influence des réseaux RC devient ainsi négligeable sur ces signaux. La vérification de la qualité du maillage du signal d'horloge doit être réalisée séparément grâce à une simulation électrique.

Les interconnexions sur plusieurs niveaux hiérarchiques

Dans le cas d'interconnexions réparties sur plusieurs niveaux hiérarchique, les délais à travers les réseaux RC doivent être recalculés sur toute l'interconnexion et non pas seulement sur la partie qui est ajoutée à chaque niveau hiérarchique. C'est pour cette raison que les réseaux RC des signaux connectés aux connecteurs externes de chaque bloc sont sauvegardés dans un fichier « .rcx ». ***HiTas*** utilise ces fichiers afin de reconstruire la totalité des réseaux RC du niveau en cours de traitement. Quand l'analyse temporelle est terminée, ***HiTas*** génère un nouveau fichier « .rcx » qui contient la description des interconnexions susceptibles d'être complétées à un niveau hiérarchique supérieur (celles qui sont connectées à un connecteur externe).

Ces réseaux peuvent être sauvegardés sous deux formes différentes : une forme complète qui n'est autre que l'ensemble des dipôles RC de chaque interconnexion ; une forme réduite qui est le délai d'Elmore. Cette deuxième forme réduite est possible car le délai d'Elmore peut être calculé de manière hiérarchique selon méthode une décrite dans la littérature [RPH83].

Cette méthode permet non seulement de traiter les interconnexions sur plusieurs niveaux mais laisse également la possibilité d'introduire des modèles plus complexes pour le calcul des délais à travers des réseaux bouclés ou complexes (avec beaucoup de branches) pour lesquels le délai d'Elmore ne serait pas assez précis.

7.4.6 La factorisation de chemins

L'algorithme de factorisation de chemins est identique à celui du graphe à plat. L'ensemble des signaux susceptibles d'être choisis comme points de factorisation sont les signaux de factorisation des figures de la descendance de la figure en cours de construction ainsi que tous les signaux nouvellement créés au niveau de la figure en cours de construction et appartenant à l'ensemble So de cette figure. En effet, dans une approche hiérarchique, le graphe de causalité réduit d'une figure père est obtenu avec les graphes de causalité réduit de ses figures filles.

7.4.7 Détermination des arcs et des signaux

La détermination des arcs du graphe de causalité réduit est réalisée durant la recherche des chemins critiques.

Pour les arcs du graphe de causalité, les ensembles d'arcs sont déterminés à partir d'un parcours en profondeur d'abord. Le parcours part des connecteurs. Tous les arcs parcourus sont classés soit dans l'ensemble Uc_c (s'ils ont un connecteur comme extrémité) soit dans l'ensemble Uc_e (si ce n'est pas le cas). Les arcs non parcourus appartiennent donc à l'ensemble Uc_i.

Les signaux de stabilité qui sont extrémités de chemins externes sont également séparés de ceux qui ne le sont pas, afin de les référencer dans les niveaux hiérarchiques supérieurs.

Pour chaque catégorie de signaux, il y a donc les signaux référencés comme étant externes à une figure et ceux comme étant internes à cette figure. Lors de la phase d'analyse temporelle, seuls sont référencés dans une figure les signaux référencés comme étant externes dans les filles de cette figure.

7.5 La visualisation interactive avec *Xtas*

Le problème de l'exploration interactive de la représentation temporelle hiérarchique d'un circuit n'est pas un simple problème d'interface graphique fournissant de beaux boutons et des menus déroulants. Il s'agit d'accélérer l'accès aux informations pertinentes dans une structure de données extrêmement complexe de façon à garantir des temps de réponse acceptables.

La recherche d'informations temporelles est réalisée grâce à des fonctions de requête. Ces fonctions prennent comme entrée des paramètres de recherche et renvoient en retour le résultat de la recherche. Ces fonctions jouent le rôle d'interface entre la structure ***Ttv*** et l'outil interactif et gèrent le cache logiciel. Les différents paramètres de ces fonctions de requête sont décrits dans l'annexe A.

7.5.1 La recherche d'une figure

Avant de commencer l'analyse interactive, il est nécessaire de construire l'arborescence de la figure temporelle à étudier.

Cette opération est réalisée par une fonction ayant comme paramètre de recherche le nom de modèle de la figure racine de l'arborescence.

Si cette figure est présente en mémoire, la fonction renvoie un pointeur sur cette figure. Si elle n'est pas présente, l'arborescence de cette figure est construite à partir des fichiers sur disque. Dans ce dernier cas, les figures de l'arborescence sont mises dans l'état minimal : Sr pour les instances, $Sc.Sr$ pour la figure de plus haut niveau. Elles ne contiennent donc que les signaux de stabilité.

7.5.2 La recherche des chemins critiques

Dans le cas des chemins critiques, il y a deux possibilités : la première concerne la recherche des n chemins critiques les plus longs ou les plus courts ; la deuxième concerne la recherche de chemins critiques dont une extrémité est fixée.

La recherche des n chemins critiques les plus longs et les plus courts

C'est une recherche très coûteuse en temps puisqu'elle s'effectue sur l'ensemble des chemins critiques.

Le nombre n est fixé par le concepteur et ne doit pas dépasser quelques dizaines de chemins pour éviter que la recherche ne prenne trop de temps et ne consomme trop de mémoire.

Par ailleurs, si la recherche nécessite de consulter un plus grand nombre de chemins critiques, le concepteur doit cette fois fixer les bornes de délai de recherche. S'il fixe comme borne maximale (respectivement minimale) le délai du énième chemin le plus long (respectivement le plus court), la recherche s'étendra aux $n - 1$ chemins suivants les plus longs (respectivement les plus courts).

Nous avons vu dans le chapitre 6 qu'il convient d'éviter que l'ensemble du graphe de causalité réduit se trouve en mémoire à la fin de la recherche. Notre algorithme prévoit de libérer les chemins au fur-et-à-mesure de la recherche.

La recherche est effectuée instance par instance en partant des figures de plus bas niveaux hiérarchiques, la multiplication de l'expansion d'une même figure avec les arcs de type Ur_i (arcs des chemins entre les registres d'un niveau hiérarchique dans le graphe de causalité réduit) est ainsi évitée. De plus, quand il existe plusieurs instances d'un même modèle, seuls sont recherchés les chemins d'une seule instance de ce modèle.

L'algorithme de libération a été optimisé pour cette recherche car c'est la plus coûteuse en mémoire donc en temps de réponse.

Dans le cas du graphe de chemins critiques à plat où tous les chemins sont explicites, il est possible de libérer les chemins au fur-et-à-mesure de la consultation et de ne garder que les n premiers en mémoire. Cependant la taille de ce graphe est tel que la phase de lecture du fichier, qui est la plus pénalisante, rend les temps de réponse trop importants. Le graphe de causalité réduit permet de diminuer la phase de lecture des fichiers tandis que l'approche hiérarchique permet de diminuer la mémoire nécessaire.

Le recherche est effectuée dans le sens anticausal car c'est celui qui limite en général le nombre d'expansions pour une même extrémité de chemin. Il est cependant possible de choisir le sens causal si le concepteur estime que c'est celui qui limite le nombre d'expansions.

Dans une méthode de vérification incrémentale ascendante, on considère que les blocs de niveau hiérarchique $n - 1$ ont été vérifiés quand on analyse le niveau n. Tous les chemins trop longs ou trop courts internes à ces blocs sont supprimés avant l'analyse temporelle du niveau hiérarchique supérieur. Seuls les chemins passant par des signaux du dernier niveau hiérarchique sont intéressants pour le concepteur. Il est possible de fixer le niveau de recherche maximal et minimal (chapitre 6) afin de limiter la recherche des chemins à un seul niveau hiérarchique. Cette fonctionnalité est rendue possible grâce à notre représentation hiérarchique qui permet non seulement de visualiser les chemins comme si le graphe était à plat, mais permet également de valider les blocs de manière ascendante. Notre représentation hiérarchique cumule donc les avantages de la représentation à plat et de la représentation hiérarchique.

La recherche des chemins critiques ayant une extrémité fixée

La fonction précédente est assez pénalisante en terme de temps. Elle permet d'identifier les chemins qui posent problème. A partir de là, il est nécessaire de disposer d'une fonction qui détermine tous les chemins critiques partant ou arrivant sur un signal particulier. Cette recherche est également réalisée sur le graphe de causalité réduit. L'utilisation de l'algorithme de recherche avec expansion de figure permet de ne charger en mémoire que la partie du graphe de causalité réduit qui nous intéresse.

L'utilisation de cette fonction est couplée avec une fonction de recherche de signaux. On commence par rechercher une liste de signaux en fonction de leur nom. Cela permet de rechercher les chemins critiques en sélectionnant un signal de départ ou d'arrivée parmi les signaux de cette liste. Le sens de la recherche détermine si le signal de recherche est considéré comme une extrémité initiale ou terminale de chemin. La recherche est réalisée de manière séquentielle (un signal à la fois) ce qui garantit un temps de réponse rapide et évite de construire trop de chemins à la fois. De ce fait, la fonction de recherche de chemins critiques à partir d'un signal autorise plus de possibilité d'interactivité que ne le fait la fonction de recherche de chemins critiques précédente. L'interactivité est également améliorée grâce à des paramètres de recherche qui permettent de limiter la recherche en fonction du délai du chemin ou du nom des extrémités de ces chemins. Il est également possible de limiter le niveau de recherche afin de ne consulter que les chemins du niveau hiérarchique qui nous intéresse.

7.5.3 La recherche du détail d'un chemin critique

Les fonctions de recherche de chemins critiques précédentes s'appliquent sur le graphe de causalité réduit. Après que les chemins critiques trop longs ou trop courts aient été identifiés, il est indispensable de connaître leur détail au niveau porte. Cette opération est quand à elle effectuée sur le graphe de causalité qui contient les délais élémentaires.

Pour cette recherche, les deux extrémités sont connues. Le niveau de recherche est également connu : c'est la figure où le chemin critique a été trouvé, c.-à-d. la figure de plus haut niveau hiérarchique à laquelle appartient au moins un arc du graphe de causalité réduit appartenant au chemin critique recherché. Cette optimisation est automatiquement utilisée pour limiter le nombre d'expansions en fixant le niveau de recherche. Le graphe de causalité étant beaucoup plus complexe que le graphe de causalité réduit, la limitation de la recherche à un niveau hiérarchique est donc très importante pour minimiser la mémoire et le temps de réponse.

Comme nous l'avons vu dans le chapitre 6, le sens de recherche peut également permettre de limiter de manière significative le nombre d'expansions. Il est nécessaire de choisir l'extrémité qui possède le moins de chemins vers des blocs extérieurs à la figure à laquelle elle appartient. Ceci est particulièrement vrai dans le cas des signaux d'horloge. Le résultat de la recherche demeure inchangé mais, si le sens est choisi à bon escient, il peut limiter considérablement le nombre d'expansions et donc le temps de réponse ainsi que la mémoire consommée.

Les deux optimisations précédentes ne sont possibles que grâce à notre représentation hiérarchique du graphe de causalité.

7.5.4 La recherche des chemins parallèles à un chemin critique

Les recherches des chemins critiques se font grâce à des parcours en largeur d'abord sur le graphes de causalité réduit. Cependant, la connaissance du seul chemin critique entre deux événements peut ne pas suffire à optimiser un circuit. Il peut exister d'autres chemins que le chemin critique qui ne respectent pas les spécifications temporelles. Il est alors nécessaire de connaître tous les chemins possibles entre deux événements. Comme nous l'avons vu dans le chapitre 3, cette opération s'effectue grâce à un parcours en profondeur sur le graphe de causalité.

Il n'est pas possible, comme dans le cas de la recherche du détail d'un chemin critique, de fixer le niveau de recherche avec celui du chemin critique. En effet, il peut exister à plusieurs niveaux hiérarchiques des chemins parallèles entre deux sommets du graphe. Cependant, nous avons vu dans le chapitre 6 que la connaissance des deux extrémités permet de fixer le niveau de recherche maximal et minimal et optimiser le temps de recherche.

Le sens de parcours peut être choisi de la même manière que pour la fonction de recherche du détail d'un chemin et permet donc de limiter la recherche.

Dans le cas de la recherche des chemins parallèles, il peut être intéressant de fixer des points de passage obligés. Cette fonctionnalité limite le nombre d'expansions. De plus, elle est très utilisée par les concepteurs afin d'optimiser des chemins particuliers.

Il est possible d'associer une condition aux points de passage :

- ET : le chemin doit passer par tous les signaux de la liste des points de passage ;
- OU : le chemin doit passer par au moins un signal de la liste des points de passage ;
- NON : le chemin ne doit passer par aucun signal de la liste des points de passage.

Comme cette recherche est effectuée sur le graphe de causalité, le concepteur peut demander le détail d'un chemin parallèle. Il suffit d'afficher la liste des délais élémentaires associés au chemin parallèle.

L'ensembles de ces possibilités offertes aux concepteurs ont été rendues possibles grâce à la représentation hiérarchique du graphe de causalité qui permet de ne charger en mémoire qu'une partie du graphe et limite ainsi la mémoire consommée ainsi que le temps de réponse de l'outil interactif.

7.6 L'analyse de stabilité

Un outil d'analyse de stabilité est en cours de réalisation dans le cadre d'un stage de DEA. Il utilise la représentation temporelle hiérarchique factorisée de la figure ***Ttv*** afin d'optimiser les algorithmes de relaxation et de limiter la taille de mémoire utilisée.

Nous avons vu dans le chapitre 2 qu'il faut parcourir plusieurs fois tous les arcs du graphe sur lequel on applique le relaxation, pour résoudre le système couplé qu'est l'analyse de stabilité des circuits conçu avec des registres à échantillonnage sur état.

Nous utilisons le graphe de causalité réduit car c'est celui qui contient le moins d'arcs. La complexité de l'algorithme étant proportionnel au nombre d'arc, les temps d'exécution seront ainsi réduits.

7.7 Conclusion

Grâce à la méthode hiérarchique et à la factorisation de chemins, nous avons conçu un outil d'analyse temporelle hiérarchique ***HiTas*** et un outil de visualisation et d'analyse interactive ***Xtas*** des informations temporelles d'un circuit. La chaîne de vérification temporelle sera complète dès que l'outil d'analyse de stabilité sera terminé. Tous ces outil s'appuient sur les structures et les fonctions de la bibliothèque ***Ttv*** qui représente la vue temporelle d'un circuit. Ainsi, notre approche rend possible le traitement de circuits complexes et ceci, à la différence des méthodes hiérarchiques existantes, sans réduire le nombre de fonctionnalités offertes au concepteur.

CHAPITRE 8

Résultats et expérimentation

Ce chapitre présente les résultats obtenus lors de l'expérimentation des outils ***HiTas*** et ***Xtas*** présentés dans le chapitre 7. Ces résultats ont pour objectif de montrer l'intérêt de notre approche hiérarchique et de notre représentation des chemins critiques sous la forme d'un graphe de causalité réduit.

Afin de vérifier la précision des temps de propagation obtenus avec le modèle ***MCC***, nous présentons tout d'abord une comparaison des temps de propagation obtenus avec ***HiTas*** et avec ***Spice*** sur des circuits utilisant les différentes circuiteries CMOS et pour des technologies allant de $0,35\mu m$ à $1,2\mu m$.

8.1 Le modèle *MCC* pour les technologies submicroniques

Les circuits à très haute densité d'intégration sont conçus en technologie submicronique. Le modèle ***MCC*** (MOS à canal court) ayant été défini à une époque où la finesse de gravure était à environ $1\mu m$, il convient de vérifier la validité de ce modèle pour ces technologies. Afin d'évaluer les performances du modèle ***MCC***, nous avons comparé les résultats donnés par ***HiTas*** et par ***Spice*** sur cinq circuits de test :

- alu [AMD77] : unité arithmétique et logique 4 bits de l'Amd2901 (CMOS dual), 434 transistors ;
- rsa [LGT93] : additionneur rapide 4 bits (xors avec bleeders et transistors de passage), 117 transistors ;
- grog [GP94b] : rom 64 mots de 1 bit (circuiterie à précharge domino), 368 transistors ;
- amg [RGA94] : multiplieur 6 bits (xor avec bleeder et portes CMOS complexes), 1114 transistors ;
- bsg : décaleur à barillet 4 bits (transistors de passage N), 270 transistors ;

TAB. 8.1 – *Comparaison entre le modèle **MCC** de **HiTas** et le simulateur **Spice***

Circuit	Technologie (μm)	Tension d'alimentation (V)	SPICE (ns)	TAS (ns)	Erreur
	0,35	3,0	3,12	3,11	-0,3
alu	0,5	3,0	7,35	7,59	+3,0
	0,8	4,5	8,58	8,58	0
	1,0	4,5	9,80	10,3	+4,5
	1,2	4,5	16,2	15,8	-2,0
	0,35	3,0	1,7	1,8	+5,5
rsa	0,5	3,0	3,76	4,02	+6,0
	0,8	4,5	4,33	4,41	+2,0
	1,0	4,5	5,02	5,50	+8,5
	1,2	4,5	8,15	8,29	+1,5
	0,35	3,0	3,45	3,78	+8,5
amg	0,5	3,0	7,75	8,48	+8,6
	0,8	4,5	8,83	9,20	+4,0
	1,0	4,5	9,96	11,1	+10,0
	1,2	4,5	17,0	18,0	+5,5
	0,35	3,0	0,99	1,01	+2
bsg	0,5	3,0	2,01	1,95	-3,0
	0,8	4,5	2,32	2,17	-7,0
	1,0	4,5	2,44	2,54	+4,0
	1,2	4,5	3,76	3,72	-1,0
	0,35	3,0	1,89	2,04	+7,0
grog	0,5	3,0	3,50	3,70	+5,0
	0,8	4,5	4,14	3,96	-4,5
	1,0	4,5	4,87	5,12	+5,0
	1,2	4,5	7,89	7,73	-2,0

Ces circuits sont conçus avec une technologie de dessin symbolique [AGL+93, GP94a] qui permet de porter chacun des circuits vers plusieurs technologies réelles [GPb94]. Nous avons choisi des circuits de petite taille pour qu'ils puissent être traités rapidement par ***Spice***. Afin de vérifier toutes les possibilités du modèle ***MCC***, tous les types de circuiteries qui permettent de concevoir des portes logiques CMOS sont utilisés.

Le simulateur ***Spice*** est considéré par les concepteurs et les fondeurs de circuits comme la référence pour la mesure des temps de propagation dans les portes CMOS. Les fondeurs délivrent donc les paramètres électriques des transistors pour ce simulateur. Nous avons mis au point une méthode automatique d'extraction des paramètres du modèle ***MCC*** à partir de ceux de ***Spice*** donnés par les fondeurs. Cette méthode est décrite dans l'annexe C

Le temps de propagation d'un chemin critique de chacun de ces circuits a été mesuré avec le simulateur ***Spice*** et l'analyseur statique ***HiTas*** pour cinq technologies auxquelles nous avons accès et qui ont des finesses de gravures différentes : $1,2\mu m$, $1\mu m$, $0,8\mu m$, $0,5\mu m$, $0,35\mu m$.

Les résultats obtenus sont donnés dans le tableau 8.1. Ces résultats ont donné lieu à une publication [DGR97].

Quelle que soit la technologie, ***HiTas*** donne des résultats très proches de ***Spice*** avec une erreur toujours inférieure à 10%. Compte tenu des hypothèses présentées dans le chapitre 3 et utilisées par l'analyse temporelle, ces résultats sont très satisfaisants. Ils montrent que le modèle ***MCC*** reste précis pour les technologies submicroniques qui permettent de concevoir les circuits à très haute densité d'intégration.

8.2 Intérêt du graphe de causalité réduit

Comme nous l'avons vu dans le chapitre 2, la représentation des chemins critiques d'un circuit est une des principales difficultés de l'analyse temporelle statique à cause de l'explosion du nombre de chemins. Pour résoudre ce problème nous avons adopté la factorisation de chemins présentée au chapitre 5. Elle permet de représenter les chemins critiques d'un circuit sous la forme d'un graphe de causalité réduit. Dans cette section, nous montrons l'intérêt du graphe de causalité réduit grâce à l'expérimentation de la factorisation de chemins sur plusieurs types de circuits.

8.2.1 Procédure de validation

HiTas permet de construire le graphe de causalité, le graphe de causalité réduit, et en désactivant la fonction de factorisation, le graphe des chemins critiques.

Pour montrer l'intérêt du graphe de causalité réduit par rapport aux deux autres représentations, nous allons comparer la taille des trois graphes obtenus sur différents circuits. Sur ces trois graphes, nous allons rechercher grâce à ***Xtas***, le chemin critique le plus long de chaque circuit et nous comparerons le temps pris par cette opération sur chacun des graphes.

Nous avons vu, dans le chapitre 7, que la recherche des n plus longs chemins d'un circuit est l'opération la plus coûteuse en temps et en mémoire pour un outil d'analyse interactive puisqu'il faut chercher parmi tous les chemins.

8.2.2 Les circuits de test

Les circuits utilisés pour la validation sont :

- Add32 [LGT93] : additionneur 32 bits, 1500 transistors ;
- Mul32 [RGA94] : multiplieur 32 bits, 24000 transistors ;
- Amd2901 [AMD77] : chemin de données 4 bits, 3800 transistors 4 bits ;
- Hadamard : calcul de la transformée d'Hadamard, 16000 transistors ;
- Dlxm : version micro-programmée du processeur 32 bits dlx, 38000 transistors ;
- Rcube [BVF+96] : cœur du routeur de message Rcube, 280000 transistors ;
- Pciddc [WDS+97] : circuit d'interface entre le routeur Rcube et le bus PCI, 200000 transistors.

Ces circuits ont été réalisés au Laboratoire ***LIP6***, fabriqués et testés. Les deux derniers sont utilisés dans des systèmes industriels.

8.2.3 Taille des graphes

Nous allons tout d'abord comparer les tailles $TGCR$ du graphe de causalité réduit, TGC du graphe de causalité et TCC du graphe de chemins critiques. Ces tailles sont définies comme étant la somme du nombre de sommets et du nombre d'arcs de chacun des graphes.

Ces résultats sont reportés dans le tableau 8.2.

TAB. 8.2 – *Comparaison entre les tailles des graphes de causalité réduit TGCR, de causalité TGC et de chemins critiques TCC (nombre de sommets + nombre d'arcs)*

circuit	TGCR	TGC	TCC	TGC/TGCR	TCC/TGCR
Add32	998	2844	4670	2,85	4,68
Mul32	2222	47401	12736	21,33	5,73
Amd2901	1783	5588	26936	3,13	15,11
Hadamard	8486	21332	13358	2,51	1,57
Dlxm	13903	49446	1013425	3,56	72,89
Rcube	150796	388154	421401	2,57	2,79
Pciddc	115421	276676	355437	2,40	3,08

Quel que soit le circuit testé, la taille du graphe de causalité réduit s'avère toujours être la plus petite. Mis à part le multiplieur Mul32 pour lequel le graphe de causalité a une taille beaucoup plus importante que les deux autre graphes, la taille du graphe de chemins critiques est souvent bien supérieure à celle des deux autres graphes. Cette constatation prouve que toutes les approches qui utilisent le graphe des chemins critiques comme représentation « compacte » font fausse route.

Dans les deux dernières colonnes du tableau 8.2, on remarque que les rapports entre le graphe de causalité et le graphe de causalité réduit est assez constant (compris entre $2,5$ et $3,5$) sauf dans le cas du multiplieur Mul32 où ce rapport est supérieur à 20. Ceci s'explique par la structure matricielle du multiplieur.

Par contre, le rapport entre le graphe de chemins critiques et le graphe de causalité réduit peut être énorme, puisqu'il vaut 73 pour le Dlxm et 15 pour l'Amd2901. Ces deux circuits possèdent un chemin de données avec un banc de registres et un décaleur, configuration présente dans la plupart des processeurs et qui conduit à des nombres de chemins critiques très importants par rapport au nombre de registre.

Afin d'analyser les rapports entre chacun des graphes, nous présentons dans le tableau 8.3, le rapport entre le nombre d'arcs NA et le nombre de sommets NS. Ce rapport indique la complexité de chacun des graphes par rapport à la taille d'un circuit, puisque la taille d'un circuit est directement liée au nombre de sommets.

TAB. 8.3 – *Complexité des graphes*

circuit	NA/NS GCR	NA/NS GC	NA/NS GCC	NBFACT/NBSOMMET
Add32	2,24	2,10	23,07	12,42 %
Mul32	5,24	3,06	48,75	0,86 %
Amd2901	1,79	1,52	48,88	5,14 %
Hadamard	2,69	1,69	5,04	1,08 %
Dlxm	2,83	2,20	303,88	2,19 %
Rcube	2,29	1,56	9,16	3,13 %
Pciddc	3,01	1,70	12,23	2,17 %

On remarque que pour le graphe de causalité le rapport entre le nombre de sommets et le nombre d'arcs est à peu près constant ; il dépend du nombre d'entrées par porte, qui en technologie CMOS et en moyenne égal à 2.

En revanche, dans le cas du graphe de chemins critiques, ce rapport est très variable puisqu'il va de 5 pour Hadamard à 304 pour le Dlxm, alors que pour le graphe de causalité réduit, il est à peu près constant et légèrement supérieur à celui du graphe de causalité.

Au vu de ces résultats, il apparaît que le graphe de causalité réduit permet de conserver une représentation des chemins critiques qui a une complexité linéaire par rapport à la taille du circuit. Autrement dit, plus l'explosion est importante, plus le gain est grand. Le rapport $NBFACT/NBSOMMET$ (nombre de points de factorisation par rapport au nombre de sommets dans le graphe de causalité) montre que ce résultat est obtenu en choisissant seulement quelques sommets comme sommets de factorisation.

Ce gain en consommation mémoire se répercute en temps de traitement dans l'analyse de stabilité des circuits. En effet, nous avons vu que cette phase nécessite que tous les arcs soient présents en mémoire en même temps. L'utilisation d'un graphe de taille la plus petite possible est donc essentielle. De plus, le temps d'exécution de l'algorithme de l'analyse de stabilité est proportionnel au nombre d'arcs.

8.2.4 La recherche du chemin critique le plus long

Il est important de vérifier que le gain de taille ne se fait pas au détriment de l'efficacité de la recherche interactive du chemin le plus long. En effet, les chemins sont explicites dans le graphe des chemins critiques alors qu'il faut les rechercher grâce à un parcours dans un graphe de causalité réduit.

Cette recherche est effectuée dans l'outil de visualisation des chemins ***Xtas*** exécutée sur un Pentium à $133MHz$ sous *Linux* et possédant 32 mega octets de mémoire vive. Les graphes sont représentés dans des fichiers sur un disque local à la machine.

TAB. 8.4 – *Temps de recherche du chemin le plus long sur le graphe de causalité réduit TRGCR, le graphe de causalité TRGC et de chemin critique TRGCC (temps en s)*

circuit	TRGCR (s)	TRGC (s)	TRGCC (s)	TRGC/TRGCR	TRGCC/TRGCR
Add32	0,02	0,08	0,12	4	6
Mul32	0,09	15,24	0,34	169,33	3,77
Amd2901	0,07	0,96	0,62	13,71	8,86
Hadamard	0,16	0,75	0,28	4,68	1,75
Dlxm	3,44	38,38	31,65	11,16	9,20
Rcube	3,81	12,22	9,61	3,21	2,43
Pciddc	3,11	12,66	8,83	4,07	2,73

Le tableau 8.4 montre que, la recherche du chemin le plus long est toujours plus rapide dans le graphe de causalité réduit que dans le graphe de chemins critiques ou que dans le graphe de causalité. En effet, les rapports $TRGC/TRGCR$ et $TRGCC/TRGCR$ sont significatifs. Le gain obtenu pendant la phase de lecture des fichiers contenant les graphes est beaucoup plus important que la perte liée au parcours du graphe.

Le graphe de causalité réduit qui a été défini pour réduire la consommation de mémoire de l'analyse de stabilité permet également de réduire de manière significative le temps de recherche des chemins critiques les plus longs d'un circuit. Ce gain est essentiel pour un outil interactif, pour lequel le temps de réponse est une contrainte forte.

8.3 Efficacité de l'approche hiérarchique

8.3.1 Les circuits de test

Nous ne disposons à l'heure actuelle, que de deux circuits de taille suffisamment importante pour justifier une approche hiérarchique. Ces circuits sont Rcube et Pciddc.

Rcube

Rcube [BVF+96] est un routeur de messages 8×8 configurable. Il dispose d'un « cross bar », de fifos asynchrones pour le contrôle de flux et d'automates de contrôle. L'interface du circuit contient sept liaisons séries à un giga bits par seconde, une sortie parallèle huit bits et des signaux de contrôle. Nous n'étudierons que le cœur du circuit car les liaisons séries contiennent de la circuiterie analogique et ne peuvent donc être traitées par ***HiTas***.

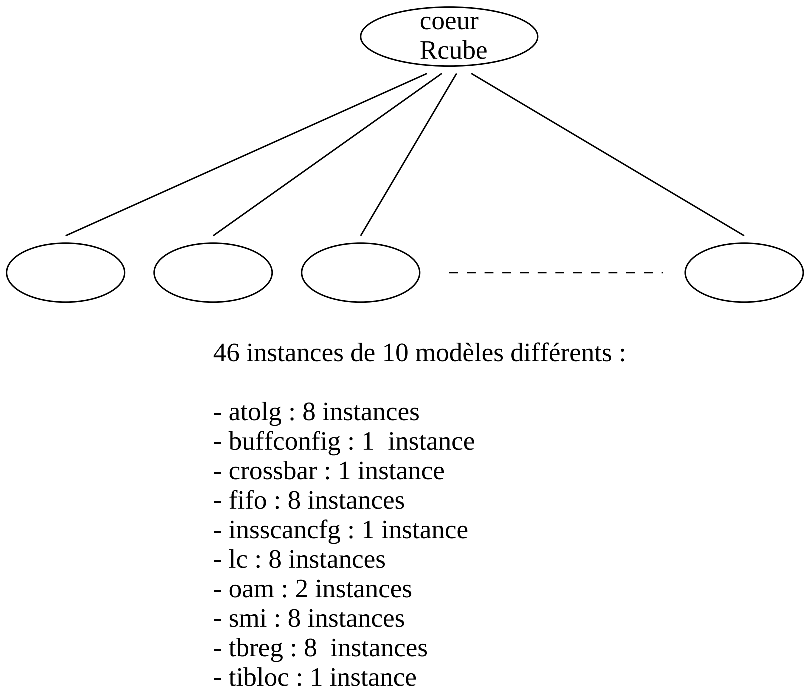


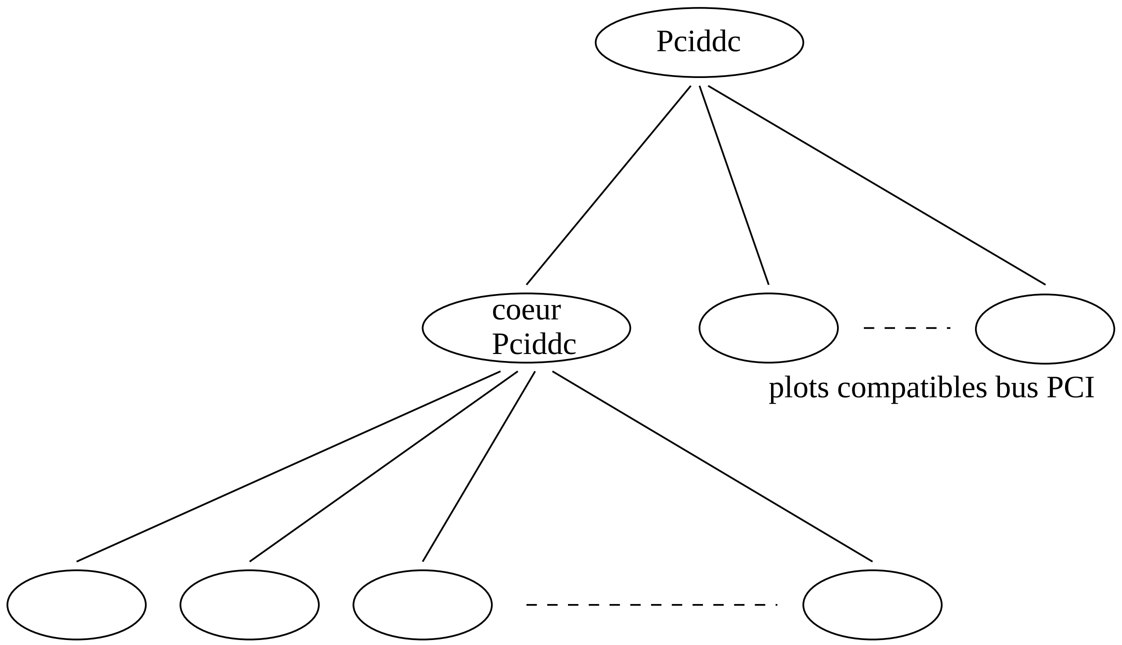
FIG. 8.1 – *La hiérarchie du cœur du circuit Rcube*

La description de la hiérarchie du circuit est représentée dans la figure 8.1. L'arbre des instances n'a qu'un seul niveau hiérarchique qui contient 46 instances de dix modèles différents. Le nombre total de transistors contenus dans le cœur est d'environ 280000.

Pciddc

Pciddc [WDS+97] joue le rôle d'interface entre un réseau de Rcube et le bus PCI. Il dispose de fifos pour le contrôle de flux des données et d'automates pour gérer les protocoles entre le bus PCI et Rcube.

La description de la hiérarchie du circuit est représentée dans la figure 8.2. L'arbre des instances possède deux niveaux hiérarchiques. Un premier niveau cœur qui a 17 instances de 14 modèles différents. Un deuxième niveau qui représente le circuit composé du cœur et des plots compatibles avec les caractéristiques électriques du bus PCI [WDP97, WPD98]. Le nombre total de transistors contenus dans Pciddc est d'environ 200000.



17 instances de 14 modèles différents :

- buffers : 1 instance
- bads : 1 instance
- r2wde : 1 instance
- r2wbc : 1 instance
- r2wab : 1 instance
- tx80 : 1 instance
- rx80 : 1 instance
- tx33 : 1 instance
- rx33 : 1 instance
- txreg : 1 instance
- rxreg : 1 instance
- mfsm : 1 instance
- fifo48m_33b_rw : 3 instances
- fifo16m_33b_rw : 2 instances

FIG. 8.2 – *La hiérarchie du circuit Pciddc*

8.3.2 La phase de construction des chemins

Construction à plat

Nous avons généré la net-list à plat en transistors de chacun des deux circuits. Nous avons exécuté l'outil ***HiTas*** sur ces deux net-list à plat sur la machine la plus puissante à laquelle nous avons accès. Cette machine est une station SPARC 10 dotée d'une mémoire vive de 420 Mégaoctets. La mémoire disponible étant insuffisante, aucun des circuits n'a pu être traité. C'est la phase de désassemblage qui limite le traitement de ces circuits.

La difficulté posée par ces circuits provient de la phase de désassemblage des fifos asynchrones. Leur conception fait appel à une circuiterie optimisée très complexe. Le désassemblage d'une fifo isolée est possible mais le traitement de plusieurs fifos dans un même bloc n'est pas possible avec les machines actuelles.

La construction hiérarchique

Étant donnée la limitation précédemment exposée, ces deux circuits nécessitent une analyse hiérarchique. Toutes les instances des figures 8.1 et 8.2 ont été traitées à plat. Les cœurs de Rcube et Pciddc ainsi que le dernier niveau de Pciddc ont été traités en mode hiérarchique.

Toutes ces analyses ont été effectuées sur une machine UltraSparc dotée de seulement 64 Mégaoctets de mémoire vive. Le traitement cumulé (chargement de la net-list, désassemblage, calcul des temps de propagation et construction des graphes) de toutes les instances et des niveaux hiérarchiques n'a pris que $3min56s$ pour le cœur de Rcube et $3min58s$ pour Pciddc.

Le désassemblage fait appel à des algorithmes complexes qui réduisent la taille des blocs pouvant être traités à quelques centaines de milliers de transistors et moins dans le cas de circuiteries optimisées. Cette limite est à peu près la même pour tous les outils qui ont une approche au niveau transistor.

C'est la phase de désassemblage qui impose une construction hiérarchique des graphes de causalité et de causalité réduit. Les temps obtenus en mode hiérarchique sont très cours sur une machine qui dispose de très peu de mémoire vive. Des temps très voisins ont été obtenus sur une machine Pentium sous Linux dotée de seulement 32 Mégaoctets de mémoire vive.

Les machines disponibles aujourd'hui sont beaucoup plus puissantes et disposent de plusieurs centaines de Mégaoctets de mémoire vive. Dans le cas des circuits contenant plusieurs millions de transistors, la complexité des blocs feuilles n'est pas très supérieure à celle de Rcube et Pciddc, seul leur nombre est plus grand. Le temps de traitement sera donc linéaire par rapport à la taille du circuit. Ceci a été confirmé par le circuit de 15 millions de transistors traité chez BULL (voir section 8.4).

8.3.3 La phase de recherche de chemins

Après la construction des graphes de causalité et de causalité réduit, nous allons tester les performances de l'outil de recherche interactive des chemins. Bien qu'il ne soit pas possible de traiter Rcube et Pciddc à plat, nous avons développé une fonction de mise à plat des graphes de causalité, de causalité réduit et de chemins critiques pour mettre en évidence les gains obtenus avec la représentation hiérarchique.

La recherche de chemins sur une structure hiérarchique

Nous avons vu dans le chapitre 7 que la recherche la plus coûteuse dans ***Xtas*** est la recherche du chemin critique le plus long puisqu'elle nécessite de consulter tous les chemins. Il est cependant possible de ne pas consulter les chemins de certaines instances. En effet, quand il existe plusieurs instances d'un même modèle les chemins internes à ces instances ne sont consultés que pour une seule des instances. Les arcs participant aux chemins internes des autres instances ne sont pas chargés en mémoire.

Ceci conduit à un gain en mémoire observé dans le tableau 8.5. Le gain est plus important dans Rcube que dans Pciddc où seules les fifos sont instanciées plusieurs fois.

TAB. 8.5 – *Consommation mémoire en Mégaoctets pour une recherche du chemin le plus long*

circuit	mémoire en MO à plat	mémoire en MO hiérarchique
Rcube	6,78	4,69
Pciddc	4,69	4,26

En ce qui concerne le gain en temps (tableau 8.6), deux facteurs favorables interviennent : le plus petit nombre de chemins à consulter et le fait de ne charger à partir des fichiers qu'une seule instance d'un même modèle. Les informations contenues dans les autres instances de ce modèle sont récupérées à partir de la figure déjà chargée, ce qui est beaucoup plus rapide que la lecture à partir du fichier.

TAB. 8.6 – *Temps mis par la recherche du chemin le plus long avec une structure hiérarchique*

circuit	temps en s à plat	temps en s hiérarchique
Rcube	3,81	2,63
Pciddc	3,11	2,51

Bien qu'ils soient appréciables, nous constatons que les gains en temps et en mémoire ne sont pas aussi importants que l'on pouvait l'espérer pour Rcube où de nombreux blocs sont instanciés plusieurs fois. Ceci s'explique par la répartition du nombre de chemins dans la hiérarchie structurelle. En effet, 50% des arcs du graphe de causalité réduit et 70% des chemins appartiennent au niveau hiérarchique du cœur de Rcube.

Efficacité du cache logiciel

Nous avons vu dans le chapitre 6 que les arcs du graphe de causalité réduit peuvent être libérés pendant la recherche des chemins, ce qui diminue la taille de mémoire nécessaire à la recherche du chemin le plus long.

TAB. 8.7 – *Gain en mémoire pour représenter les arcs en utilisant le cache logiciel*

circuit	mémoire en MO sans cache	mémoire en MO avec cache	rapport sans cache/avec cache
Rcube	3,11	2,58	1,2
Pciddc	3,26	1,23	2,65

Le tableau 8.7 montre que la mémoire nécessaire pour représenter les arcs du graphe de causalité peut être réduite d'un facteur $1,2$ pour Rcube et de $2,65$ pour Pciddc. Il est beaucoup plus important pour Pciddc car il dépend du nombre de modèles de blocs et du nombre de niveaux hiérarchiques. Plus le circuit possède de niveaux hiérarchiques, plus le cache logiciel permet de gagner de la place en mémoire. Ceci va dans le sens de l'analyse des circuits à très haute densité d'intégration qui disposent souvent d'au moins quatre niveaux hiérarchiques et plusieurs centaines de blocs feuilles.

L'utilisation du cache logiciel a un coût en temps de recherche du chemin critique le plus long. Les tableaux 8.8 et 8.9 donnent ce temps en fonction de la taille du cache.

TAB. 8.8 – *Perte de temps due au cache logiciel pour Rcube*

taille en MO	temps en s
2,58	4,49
2,7	2,73
2,8	2,65
3,11	2,51

TAB. 8.9 – *Perte de temps due au cache logiciel pour Pciddc*

taille en MO	temps en s
1,23	5,20
1,5	2,91
2	2,7
3,26	2,63

Nous remarquons dans les deux cas que, dès que la taille du cache est supérieure au minimum indispensable, le temps de recherche du chemin le plus long est très proche de celui obtenu avec la taille maximale nécessaire. Pour Pciddc, avec une occupation en mémoire divisée par 2, le temps de recherche n'est augmenté que de 10%.

8.3.4 Efficacité de la recherche à un seul niveau hiérarchique

Dans une méthode de vérification incrémentale hiérarchique, il suffit de rechercher parmi les chemins du dernier niveau hiérarchique puisque les blocs des niveaux hiérarchiques inférieurs ont déjà été vérifiés et optimisés.

Les résultats présentés dans le tableau 8.10 montrent que dans le cas de Pciddc, qui a deux niveaux hiérarchiques, le gain en temps et en mémoire pour la représentation des arcs est important.

Tous ces résultats sont à comparer avec ceux obtenus avec une représentation à plat du graphe des chemins critiques. La méthode hiérarchique avec factorisation de chemin et cache logiciel permet de gagner un facteur d'environ 4 en mémoire et en temps pour Rcube et un facteur $7,5$ en mémoire et $6,5$ en temps pour Pciddc.

TAB. 8.10 – *Consommation mémoire et temps de recherche du chemin le plus long en hiérarchique avec cache en se limitant aux chemins du dernier niveau hiérarchique*

circuit	mémoire totale GCR hiérarchique	temps de recherche GCR hiérarchique	mémoire totale GCC à plat	temps de recherche GCC à plat
Rcube	4,1	2,43	16,42	9,61
Pciddc	1,94	1,37	14,53	8,83

8.3.5 Tableau récapitulatif

Le tableau 8.11 récapitule les résultats des différentes étapes d'optimisation.

TAB. 8.11 – *Récapitulatif de la consommation mémoire et du temps de recherche du chemin le plus long grâce au graphe de causalité hiérarchique et au cache logiciel*

optimisation	Rcube mémoire	Pciddc mémoire	Rcube temps	Pciddc temps
GCC à plat	16,42	14,53	9,61	8,83
factorisation GCR à plat	6,48	4,69	3,81	3,11
GCR hiérarchique	4,69	4,26	2,63	2,51
GCR hiérarchique avec cache	4,28	2,43	2,73	2,91
limitation au dernier niveau hiérarchique	4,1	1,94	2,43	1,37

8.4 Les circuits à très haute densité d'intégration

L'objectif de cette thèse est l'analyse temporelle des circuits à très haute densité d'intégration. Les plus gros circuits actuellement développés possèdent entre 10 et 20 millions de transistors. Malheureusement, nous ne disposons pas au Laboratoire ***LIP6*** de circuits aussi importants.

Cependant, ***HiTas*** est développé dans le cadre d'un projet européen MEDEA en partenariat avec la compagnie BULL qui utilise, depuis 1992, la version à plat ***Tas*** pour le développement de ses circuits.

La compagnie BULL a choisi d'utiliser l'outil ***HiTas*** pour réaliser l'analyse temporelle de ses gros circuits dont la taille dépasse la dizaine de millions de transistors. L'expérimentation de ***HiTas*** chez BULL sur des circuits à très haute densité d'intégration a commencé au mois de juillet 1998.

8.5 Conclusion

L'analyse temporelle hiérarchique permet de traiter des circuits à très haute densité d'intégration ; par ailleurs la représentation de la structure ***Ttv*** permet grâce à la hiérarchie et à la factorisation de chemins, d'améliorer de manière significative les temps et la consommation mémoire de la recherche des chemins d'un circuit. Ces performances ont été constatées sur des circuits présentant très peu de blocs et seulement un et deux niveaux hiérarchiques. De plus, ce ne sont pas des circuits où le nombre de chemins par rapport au nombre de registres est important. Les gains sont d'autant plus significatifs que le nombre de niveaux hiérarchiques et le nombre de blocs sont grands, ce qui est le cas des circuits de plusieurs millions de transistors.

Chapitre 9

Conclusion

L'avènement des technologies fortement submicroniques permet de concevoir des circuits de plusieurs millions de transistors. Cependant, la validation des circuits à très haute densité d'intégration reste un problème délicat à résoudre puisque la définition de méthodes hiérarchiques n'est pas aussi aisée que pour les outils de synthèse ou de placement routage. En effet, dans le cas des outils de validation, les sous-blocs instanciés ne peuvent être assimilés à des boîtes noires.

Dans le cas de la vérification temporelle, il existe des méthodes hiérarchiques qui considèrent les instances comme des boîtes noires ou des boîtes grises et détectent différentes violations des contraintes de fonctionnement temporel. Cependant, un outil de validation doit également permettre au concepteur de retrouver les causes des erreurs afin de les corriger plus facilement.

Dans cette thèse, nous avons apporté une solution hiérarchique qui permet non seulement de vérifier le bon fonctionnement d'un circuit à très haute densité d'intégration, mais aussi de visualiser les informations temporelles relatives à ce circuits. Notre approche hiérarchique permet d'obtenir, avec la même facilité qu'une approche à plat, tous les chemins critiques ou non critiques, ainsi que leur détail au niveau porte.

Cette facilité d'utilisation a été rendue possible grâce à la définition d'une représentation hiérarchique temporelle multi-niveaux du circuit. Cette vue est générique et est utilisée par tous les outils de la chaîne de vérification temporelle : l'analyseur temporels ***HiTas*** pour le calcul des délais et des chemins critiques, l'outil interactif ***Xtas*** de visualisation des informations temporelles d'un circuit, et l'outil d'analyse de stabilité en cours de développement.

La représentation que nous avons définie a rendu possible la vérification de circuit de très haute complexité de plus de dix millions de transistors. Son système de cache logiciel limite la quantité de mémoire requise pour le traitement de gros circuits.

De plus, notre approche permet de prendre en compte les résistances d'interconnexion sur plusieurs niveaux hiérarchiques. Bien que les délais d'interconnexion soient pour l'instant calculés avec le délai d'Elmore, rien n'empêche l'utilisation de modèles plus complexes pour améliorer la précision du calcul des délais dans des réseaux RC contenant des boucles. Les réseaux incomplets à un niveau hiérarchique sont reportés au niveau hiérarchique supérieur soit sous une forme réduite, grâce à la propriété hiérarchique du délai d'Elmore, soit sous une forme complète pour l'utilisation de modèles plus complexes tel que « Asymptotic Waveform Evaluation » [PR90] ou la simulation électrique de type ***Spice*** ou ***Eldo***.

Notre approche hiérarchique intègre également un nouveau concept dans l'analyse temporelle, la factorisation des chemins critiques. En effet, l'un des problèmes fondamentaux de l'analyse temporelle est la croissance quadratique du nombre de chemins critiques par rapport au nombre de registres présents dans le circuit. La représentation temporelle qui conserve tous les délais élémentaires au niveau portes logiques a une complexité linéaire avec le nombre de transistors. Malheureusement, cette représentation explicite du graphe de causalité se prête mal à la recherche interactive des chemins à cause des temps de traitement. La représentation des chemins les plus longs et les plus courts sous la forme d'un graphe de chemins critiques n'est pas non plus une bonne solution puisqu'elle conduit à des volumes de données inacceptables. Afin de résoudre ce problème, nous avons mis au point une méthode automatique de factorisation de chemins qui permet de représenter les chemins critiques sous la forme d'un graphe de causalité réduit, beaucoup plus économique en mémoire que le graphe de chemins critiques et que le graphe de causalité. De plus, la recherche des chemins critiques dans ce graphe de causalité réduit est extrêmement plus rapide. Tous les outils de la chaîne de vérification temporelle utilisent le graphe de causalité réduit.

Perspectives

Outre le besoin d'amélioration de la précision du calcul des délais dans les réseaux RC, les technologies très fortement submicroniques font apparaître une nouvelle source d'imprécision dans les délais dus aux interconnexions. En effet, la diminution de la largeur entre les fils de métal fait que la capacité de couplage entre les fils métalliques devient beaucoup plus importante que leur capacité par rapport au substrat. Ce phénomène est amplifié par l'augmentation de l'épaisseur des fils métalliques afin qu'ils gardent des valeurs de résistances acceptables.

L'effet de la capacité de couplage entre deux fils est nul si les fils commutent dans le même sens en même temps ; il est double dans le cas où les deux fils commutent dans des sens opposés (effet Miller). Sachant que pour les technologies qui ont une finesse de gravure de l'ordre de $0,2\mu m$, les capacités de couplage peuvent atteindre 90% de la capacité totale, ce phénomène peut engendrer des erreurs significatives. En effet, l'analyse temporelle statique présentée dans cette thèse, fait l'hypothèse qu'il n'y a pas de transitions simultanées entre deux fils qui ont une capacité de couplage entre eux.

Il est donc indispensable de prendre en compte de façon exacte les capacités de couplage si un risque important de transitions simultanées existe. Dans tous les cas, il est nécessaire de s'appuyer sur une représentation des délais des portes, donc sur un graphe de causalité. De plus, le traitement de circuits à très haute densité d'intégration imposera le recours à une méthodologie hiérarchique. Notre vue temporelle permet de représenter le graphe de causalité de manière hiérarchique.

La difficulté principale de la prise en compte des capacités de couplage réside dans le fait qu'il est nécessaire de disposer des temps de propagation pour déterminer si deux signaux peuvent commuter en même temps. Malheureusement, la prise en compte des transitions simultanées conduit à la modification des temps de propagation qui ont permis de déterminer les transitions simultanées. Une thèse à débuté au Laboratoire ***LIP6*** afin de résoudre ce système couplé pour introduire l'effet des capacité de couplage dans l'analyse temporelle statique.

Annexes

Les annexes qui suivent présentent :

- la structure de données ***Ttv*** et les paramètres des fonctions de recherche de chemins ;
- la syntaxe des fichiers « .dtx » et « .ttx » ;
- la méthode de détermination des paramètres du modèle ***MCC***.

Annexe A

La figure TTV

Cette annexe présente la structure de données ***Ttv*** écrite en langage C ainsi que les fonctions qui permettent de rechercher les chemins dans cette structure. Ces fonctions permettent la recherche des chemins critiques, la recherche du détail au niveau porte des chemins critiques, la recherche des chemins parallèles à ces chemins critiques et la recherche du détail au niveau porte des chemins parallèles. Tous les paramètres de ces fonctions qui permettent de limiter ou d'orienter la recherche sont décrits en détail.

Les structures de données

La structure représentant les événements

```
typedef struct ttvevent
    {
     struct ttvsig       *ROOT ;
     long                 TYPE ;
     struct ttvline      *INLINE ;
     struct ttvline      *INPATH ;
     long                 DELAY ;
     struct ttvline      *OUTLINE ;
     struct ptype        *USER ;
    }
ttvevent_list ;
```

– ROOT : signal correspondant à l'événement ;
– TYPE : type de l'événement (montant ou descendant) ;
– INLINE : liste des délais élémentaires ;
– INPATH : liste des chemins factorisés ;

- DELAY : le délai π pour les parcours de graphes ;
- OUTLINE : l'arc ω représentant l'arc précédent pour les parcours de graphe ;
- USER : champs utilisateur.

La structure représentant les signaux

```
typedef struct ttvsig
    {
     char                *NAME ;
     struct ttvfig       *ROOT ;
     long                 TYPE ;
     struct ttvevent      NODE[2] ;
     struct ptype        *USER ;
    }
ttvsig_list ;
```

- NAME : nom du signal ;
- ROOT : figure à laquelle appartient le signal ;
- TYPE : catégorie du signal ;
- NODE : les deux événements associés au signal ;
- USER : champs utilisateur.

La structure suivante permet de définir un tableau de signal pour une allocation mémoire par blocs.

```
typedef struct ttvsbloc
    {
     struct ttvsbloc     *NEXT ;
     struct ttvsig        SIG[TTV_MAX_SBLOC] ;
    }
ttvsbloc_list ;
```

La structure représentant les arcs

```
typedef struct ttvline
    {
     struct ttvline      *NEXT ;
```

```
     struct ttvfig       *FIG ;
     struct ttvevent     *ROOT ;
     long                 TYPE ;
     struct ttvevent     *NODE ;
     long                 VALMAX ;
     long                 FMAX ;
     long                 VALMIN ;
     long                 FMIN ;
     struct ptype        *USER ;
    }
ttvline_list ;
```

- NEXT : arc suivant pour les listes d'arcs ;
- FIG : figure à laquelle appartient l'arc ;
- ROOT : événement terminal ;
- TYPE : type d'arc (ensemble auquel il appartient) ;
- NODE : événement initial ;
- VALMAX : délai maximal ;
- FMAX : front maximal ;
- VALMIN : délai minimal ;
- FMIN : front minimal ;
- USER : champs utilisateur.

La structure suivante permet de définir un tableau d'arcs pour une allocation mémoire par blocs.

```
typedef struct ttvlbloc
    {
     struct ttvlbloc     *NEXT ;
     struct ttvline       LINE[TTV_MAX_LBLOC] ;
    }
ttvlbloc_list ;
```

La structure représentant les figures temporelles

La structure contenant les informations d'une figure temporelle

```
typedef struct ttvinfo
```

```
    {
     char                   *TOOLNAME ;
     char                   *TOOLVERSION ;
     char                   *TECHNONAME ;
     char                   *TECHNOVERSION ;
     int                     TTVYEAR ;
     int                     TTVMONTH ;
     int                     TTVDAY ;
     int                     TTVHOUR ;
     int                     TTVMIN ;
     int                     TTVSEC ;
     long                    SLOPE ;
     long                    CAPAOUT ;
    }
ttvinfo_list ;
```

- TOOLNAME : nom de l'outil qui a permis de générer la figure temporelle ;
- TOOLVERSION : numéro de version de l'outil ;
- TECHNONAME : nom de la technologie ;
- TECHNOVERSION : numéro de version de la technologie ;
- TTVYEAR : année de génération de la figure temporelle ;
- TTVMONTH : mois de génération de la figure temporelle ;
- TTVDAY : jour de génération de la figure temporelle ;
- TTVHOUR : heure de génération de la figure temporelle ;
- TTVMIN : minute de génération de la figure temporelle ;
- TTVSEC : seconde de génération de la figure temporelle ;
- SLOPE : front appliqué sur les connecteurs d'entrée ;
- CAPAOUT : capacité appliquée sur les connecteurs de sortie.

```
\subsubsection*{La structure représentant une figure temporelle}

typedef struct ttvfig
    {
     struct ttvfig       *NEXT ;
     struct ttvfig       *ROOT ;
     long                 OLD ;
     long                 LEVEL ;
     char                *FIGNAME ;
```

```
        char                    *INSNAME ;
        long                     STATUS ;
        struct ttvinfo           INFO ;
        struct ttvsig          **CONSIG ;
        long                     NBCONSIG ;
        struct ttvsig          **NCSIG ;
        long                     NBNCSIG ;
        struct chain            *INS ;
        struct ttvsig          **ELCMDSIG ;
        long                     NBELCMDSIG ;
        struct ttvsig          **ILCMDSIG ;
        long                     NBILCMDSIG ;
        struct ttvsig          **ELATCHSIG ;
        long                     NBELATCHSIG ;
        struct ttvsig          **ILATCHSIG ;
        long                     NBILATCHSIG ;
        struct ttvsig          **EPRESIG ;
        long                     NBEPRESIG ;
        struct ttvsig          **IPRESIG ;
        long                     NBIPRESIG ;
        struct ttvsig          **EXTSIG ;
        long                     NBEXTSIG ;
        struct ttvsig          **INTSIG ;
        long                     NBINTSIG ;
        struct ttvsig          **ESIG ;
        long                     NBESIG ;
        struct ttvsbloc         *ISIG ;
        long                     NBISIG ;
        struct ttvlbloc         *PBLOC ;
        long                     NBPBLOC ;
        struct ttvlbloc         *JBLOC ;
        long                     NBJBLOC ;
        struct ttvlbloc         *TBLOC ;
        long                     NBTBLOC ;
        struct ttvlbloc         *FBLOC ;
        long                     NBFBLOC ;
        struct ttvlbloc         *EBLOC ;
        long                     NBEBLOC ;
        struct ttvlbloc         *DBLOC ;
        long                     NBDBLOC ;
        struct ptype            *USER ;
      }
ttvfig_list ;
```

- NEXT : figure suivante dans le liste de toutes les figures temporelles en mémoire ;
- ROOT : figure père ;
- OLD : date de la dernière expansion ;
- LEVEL : niveau hiérarchique ;
- FIGNAME : nom de la figure ;
- INSNAME : nom d'instance ;
- STATUS : état de la figure ;
- INFO : structure d'information ;
- CONSIG : tableau des connecteurs externes ;
- NBCONSIG : nombre de connecteurs externes ;
- NCSIG : tableau des connecteurs internes ;
- NBNCSIG : nombre de connecteurs internes ;
- INS : liste des instances ;
- ELCMDSIG : tableau des commandes externes ;
- NBELCMDSIG : nombre de commandes externes ;
- ILCMDSIG : tableau des commandes internes ;
- NBILCMDSIG : nombre de commandes internes ;
- ELATCHSIG : tableau des registres externes ;
- NBELATCHSIG : nombre de registres externes ;
- ILATCHSIG : tableau des registres internes ;
- NBILATCHSIG : nombre de registres internes ;
- EPRESIG : tableau des bus préchargés externes ;
- NBEPRESIG : nombre de bus préchargés externes ;
- IPRESIG : tableau des bus préchargés internes ;
- NBIPRESIG : nombre de bus préchargés internes ;
- EXTSIG : tableau des points de factorisation externes ;
- NBEXTSIG : nombre de points de factorisation externes ;
- INTSIG : tableau des points de factorisation internes ;
- NBINTSIG : nombre de points de factorisation internes ;
- ESIG : tableau des autres signaux externes ;
- NBESIG : nombre des autres signaux externes ;
- ISIG : tableau des autres signaux internes ;

- NBISIG : nombre des autres signaux internes ;
- PBLOC : chemins factorisés internes (ensemble Ur_i) ;
- NBPBLOC : nombre de chemins factorisés internes ;
- JBLOC : chemins factorisés externes (ensemble Ur_e) ;
- NBJBLOC : nombre de chemins factorisés externes ;
- TBLOC : chemins factorisés ayant un connecteur comme extrémité (ensemble Ur_c) ;
- NBTBLOC : nombre de chemins factorisés ayant un connecteur comme extrémité ;
- FBLOC : délais élémentaires internes (ensemble Uc_i) ;
- NBFBLOC : nombre de délais élémentaires internes ;
- EBLOC : délais élémentaires externes (ensemble Uc_e) ;
- NBEBLOC : nombre de délais élémentaires externes ;
- DBLOC : délais élémentaires ayant un connecteur comme extrémité (ensemble Uc_c) ;
- NBDBLOC : nombre de délais élémentaires ayant un connecteur comme extrémité ;
- USER : champs utilisateur.

Les paramètres des fonctions de recherche sous *Xtas*

Chaque fonction de parcours de la base de données possède des paramètres de recherche. Nous allons détailler les paramètres de recherche des principales fonctions de la base de données.

La recherche des signaux

Les paramètres de recherche des signaux sont :

- la figure de plus haut niveau hiérarchique : c'est la figure qui représente le circuit à analyser ;
- la figure de recherche : ce paramètre facultatif permet de limiter la recherche aux signaux d'une instance particulière ;
- les catégories de signaux à rechercher : elles permettent de limiter la recherche à une ou plusieurs catégories de signal ;
- les expressions régulières de recherche : elles constituent une liste d'expressions régulières pouvant contenir des caractères « * » pour limiter la recherche aux signaux dont le nom correspond à une des expressions régulières.

la fonction de recherche des signaux retourne la liste des signaux satisfaisant les paramètres de recherche.

La recherche des chemins critiques

La recherche des n chemins critiques les plus longs et les plus courts

Les paramètres de recherche sont :

- la figure de plus haut niveau hiérarchique ;
- la figure de recherche ;
- le nombre de chemins à rechercher ;
- le type de recherche : maximal ou minimal ;
- le sens de recherche : causal ou anticausal ;
- les bornes maximales et minimales de délai ;
- les expressions régulières de recherche initiales et terminales.

La figure de recherche permet de fixer le niveau de recherche maximal et minimal (chapitre 6) afin de limiter la recherche des chemins à un seul niveau hiérarchique.

Le sens de la recherche par défaut est le sens anticausal car c'est celui qui limite en général le nombre d'expansions. Il est cependant possible de changer le sens si ce n'est pas le cas.

Les bornes de délais et les expressions régulières permettent de limiter la recherche à un domaine de chemin plus réduit.

La recherche des chemins critiques à partir d'un signal

Les paramètres de recherches sont :

- la figure de plus haut niveau hiérarchique ;
- la figure de recherche ;
- le signal de recherche ;
- le type de recherche : maximal ou minimal ;
- le sens de recherche : causal ou anticausal ;
- les bornes maximales et minimales de délai ;
- la liste des expressions régulières de recherche (initiales ou terminales en fonction du sens).

Le sens de la recherche détermine si le signal de recherche est considéré comme une extrémité initiale ou terminale de chemin. Pour cette fonction, il n'y a qu'une liste d'expressions régulières. Elle concerne les extrémités initiales si le signal de recherche est considéré comme une extrémité terminale et vice-versa.

La recherche du détail d'un chemin critique

Les paramètres de recherche de cette fonction sont :

- la figure de plus haut niveau hiérarchique ;
- la figure de recherche : elle permet de limiter la recherche du détail d'un chemin au niveau dans lequel ce chemin a été trouvé dans le graphe de causalité réduit ;
- l'événement initial ;
- l'événement terminal ;
- le type de recherche : maximal ou minimal ;
- le sens de recherche : causal ou anticausal.

La recherche des chemins parallèles à un chemin critique

Les paramètres de recherche sont :

- la figure de plus haut niveau hiérarchique ;
- la figure de recherche ;
- l'événement initial ;
- l'événement terminal ;
- le type de recherche : maximal ou minimal ;
- le sens de recherche : causal ou anticausal ;
- les bornes maximales et minimales de délai ;
- une liste des expressions régulières pour les signaux de passage ;
- une condition sur la liste des expressions régulières.

La liste d'expressions régulières permet de choisir une liste de signaux intermédiaires parmi lesquels le concepteur fixe des points de passage des chemins qu'il recherche en associant une condition :

- ET : le chemin doit passer par tous les signaux de la liste ;
- OU : le chemin doit passer par au moins un signal de la liste ;
- NON : le chemin ne doit passer par aucun signal de la liste.

La possibilité d'imposer le passage d'un chemin par des signaux particuliers permet d'éliminer des faux chemins de la recherche.

La recherche du détail d'un chemin parallèle

Il est difficile de désigner un chemin parmi une liste de chemins parallèles puisqu'ils ont tous les mêmes extrémités. Nous avons choisi comme critère de sélection le délai du chemin. On recherche donc le détail des chemins parallèles qui ont un délai égal à une valeur fixé. Ceci peut conduire au choix de plusieurs chemins qui ont le même délai. Cette fonction retourne alors une liste de détails de chemins entre deux événements qui ont le même délai fixé.

Les paramètres de recherche sont :

– la figure de plus haut niveau hiérarchique ;
– la figure de recherche ;
– l'événement initial ;
– l'événement terminal ;
– le type de recherche : maximal ou minimal ;
– le sens de recherche : causal ou anticausal ;
– le délai du chemin parallèle ;
– une liste des expressions régulières pour les signaux de passage ;
– une condition sur la liste des expressions régulières.

Le résultat de la recherche

Les résultats de la recherche de chemins sont des structures de données qui représentent soit des listes de chemins, soit des listes de détails de chemins.

La structure chemin

```
typedef struct ttvpath
    {
     struct ttvpath      *NEXT ;
     struct ttvfig       *FIG ;
     struct ttvevent     *ROOT ;
     struct ttvevent     *NODE ;
     struct ttvevent     *CMD ;
     long                 TYPE ;
     long                 DELAY ;
     long                 SLOPE ;
    }
ttvpath_list ;
```

– NEXT : chemin suivant ;
– FIG : figure dans laquelle le chemin a été trouvé ;
– ROOT : événement terminal ;
– NODE : événement initial ;
– CMD : événement commande dans le cas où l'événement terminal est un registre ;
– TYPE : type de chemin maximal ou minimal ;
– DELAY : temps de propagation du chemin ;
– SLOPE : front sur l'événement terminal.

La structure détail

```
typedef struct ttvcritic
    {
     struct ttvcritic    *NEXT ;
     struct ttvfig       *FIG ;
     char                *NAME ;
     long                 TYPE ;
     long                 DELAY ;
     long                 SLOPE ;
     char                 SNODE ;
    }
ttvcritic_list ;
```

– NEXT : structure qui détermine l'événement suivant du détail de chemin ;
– FIG : figure à laquelle appartient l'arc qui a permis d'arriver sur ce sommet ;
– NAME : nom de l'événement ;
– TYPE : type de chemin maximal ou minimal ;
– DELAY : délai entre l'événement précédent et l'événement courant ;
– SLOPE : front sur l'événement courant ;
– SNODE : type de front montant ou descendant.

La figure A.1 donne les deux types de représentations.

Dans la représentation d'un chemin, la « figure du chemin » indique le niveau de recherche où le chemin a été retrouvé. Il est utilisé pour fixer le niveau de recherche du détail au niveau porte de ce chemin. La commande n'est documentée que si l'événement terminal est sur un registre ou sur une précharge.

Concernant la représentation du détail d'un chemin, il s'agit d'une liste chaînée de structures puisqu'un détail possède un nombre variable d'événements.

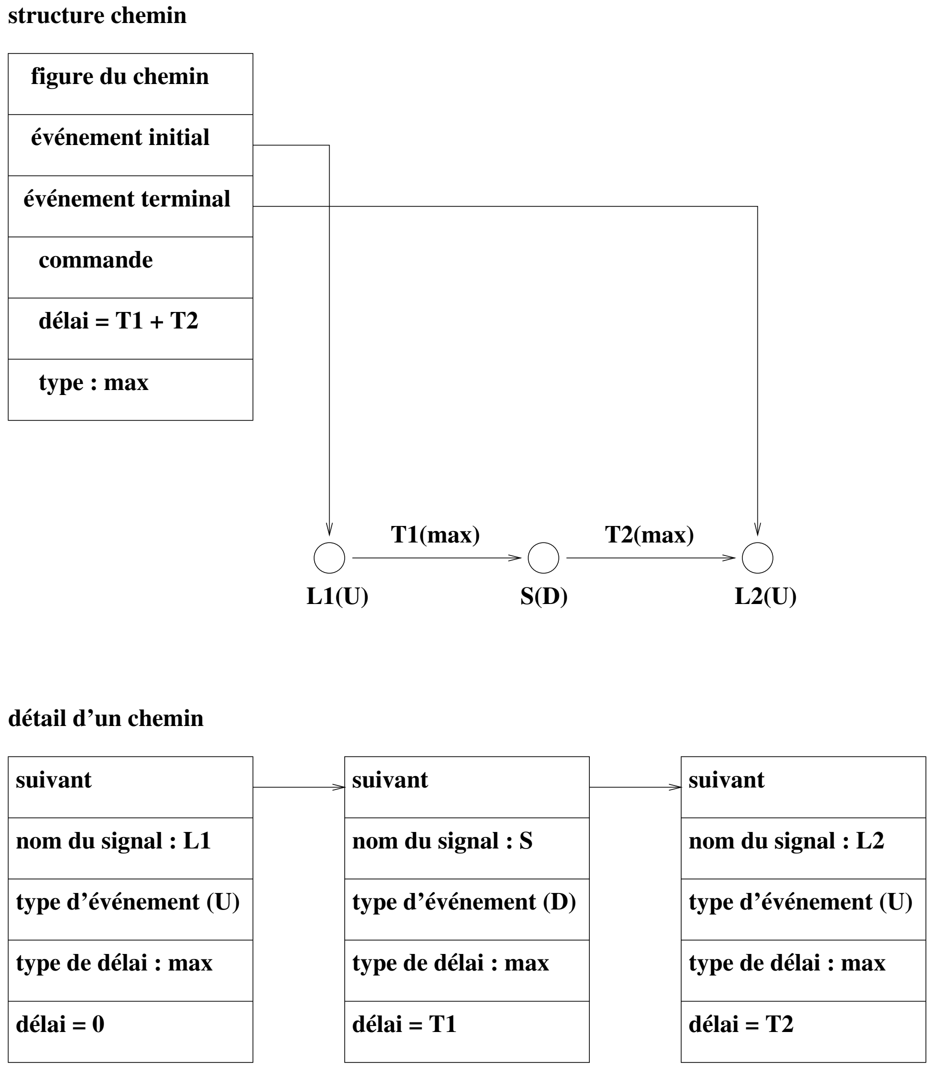


FIG. A.1 – *Structure résultat de la recherche de chemin*

Le champ délai correspond au délai entre l'événement courant et l'événement précédent. Le premier délai de la liste vaut zéro.

Contrairement à la structure chemin où les événements sont déterminés par des pointeurs sur le nœud événement de la structure ***Ttv***, dans le cas du détail d'un chemin, l'événement est déterminé par deux champs : son nom et son type. Ceci est dû au cache logiciel : les signaux intermédiaires des ensembles *So* peuvent, à un moment ou un autre, ne plus être en mémoire et, dans ce cas, le pointeur sur un événement peut être modifié.

ANNEXE B

La représentation sur disque

Les fichiers **.dtx** et **.ttx** sont la représentation sur disque d'une figure temporelle ***Ttv***. Le fichier **.dtx** permet de représenter le graphe de causalité tandis que le fichier **.ttx** permet de représenter le graphe de causalité réduit.

Cette annexe définit la syntaxe de ces fichiers et donne des exemples de représentation de deux figures terminales et d'une figure non-terminale.

Les commentaires

Les lignes commençant par **#** sont des commentaires.

Les informations générales

```
H outil version nom_de_figure technologie version_de_technologie
front_entrée capacité_de_sortie niveau_hiérarchique
(jour mois année heure minute second) ;
```

Liste des instances

```
X nom_de_figure nom_d_instances ;
```

Liste des connecteurs externes

```
C type index nom (capa rupmax rdmax fupmax fdownmax rupmin
rdownmin fupmin fdownmin) ;
```

Le type

Le type permet de définir le sens des connecteurs et éventuellement de préciser si ce connecteur est une commande.

- I : connecteur d'entrée ;
- O : connecteur de sortie ;
- B : connecteur de sortie réutilisé en entrée ;
- T : connecteur d'entrée et de sortie trois états ;
- Z : connecteur de sortie trois états ;
- X : connecteur non connecté ;
- IQ : connecteur d'entrée et commande de registre ou de bus préchargé ;
- BQ : connecteur de sortie réutilisé en entrée et commande de registre ou de bus préchargé.

Les paramètres électriques

Les paramètres électriques permettent de calculer les fronts et les délais pour les figures des niveaux hiérarchique supérieurs.

- capa : capacité attachée au connecteur ;
- rupmax : résistance maximale vers la tension d'alimentation haute ;
- rdownmax : résistance maximale vers la tension d'alimentation basse ;
- fupmax : front montant maximal ;
- fdownmax : front descendant maximal ;
- rupmin : résistance minimale vers la tension d'alimentation haute ;
- rdownmin : résistance minimale vers la tension d'alimentation basse ;
- fupmin : front montant minimal ;
- fdownmin : front descendant minimal ;

Liste des connecteurs internes

```
N type index nom (index_con capa rupmax rdmax fupmax fdownmax
rupmin rdownmin fupmin fdownmin) ;
```

Le type

Le type définit si le connecteur interne est une commande ou pas ;

- Q : connecteur interne commande ;
- I : connecteur interne autre que commande.

Les paramètres électriques

Les paramètres électriques sont les mêmes que pour les connecteurs externes. On spécifie également le connecteur externe associé grâce à son index : index_con.

Liste des commandes de registres ou de bus précharge

```
Q type index nom (nom_commande_0 nomcommande_1...) ;
```

Le type

- C : commande connecteur externe. Cette commande est également un connecteur externe ;
- N : commande connecteur interne. Cette commande est également un connecteur interne ;
- E : commande externe. Cette commande est une extrémité d'un chemin externe ;
- I : commande interne. Cette commande n'est pas une extrémité d'un chemin externe .

La liste des commandes des niveaux inférieurs

(nomcommand0 nomcommand1 ...) est la liste des commandes des niveaux hiérarchiques inférieurs qui sont des commandes connecteurs internes ou externes et qui ont été remplacées par cette commande à ce niveau hiérarchique. C'est ce qui permet de définir l'attribut commande d'un arc à chaque niveau hiérarchique.

Liste des registres

```
L type index nom (commande_index0 commande_type0
commande_index1 commande_type1 ...) ;
```

Le type

- LE : registre externe à écriture sur état ;
- LI : registre interne à écriture sur état ;
- FE : registre externe à écriture sur front ;
- FI : registre interne à écriture sur front.

Liste des événements commandes

(commande_index0 commande_type0 commande_index1 commande_type1 ...) définit la liste des événements permettant soit l'ouverture d'un registre à écriture sur état soit l'écriture dans le registre à écriture sur front. (commande_index) définit l'index du signal de commande. (commande_type) définit si c'est un événement montant (U) ou un événement descendant (D).

Liste des bus préchargés

```
R type index nom (commande_index0 commande_type0
commande_index1 commande_type1 ...) ;
```

Le type

- E : bus préchargé externe ;
- I : bus préchargé interne.

Liste des signaux de factorisation

```
I type index nom ;
```

Le type

- E : signal de factorisation externe ;
- I : signal de factorisation interne.

Liste des autres signaux pour le fichier « .dtx » seulement

```
S type index nom ;
```

Le type

- E : signal externe ayant un délai élémentaire avec un connecteur interne ou externe ;
- I : signal interne.

Les délais élémentaires et les chemins factorisés

```
type_délai début_type début_index fin_type fin_index (
        (com_index com_type(type_trans type_délai délai front inf_élec))
        (..(...))
        (..(...))
        .....
        (..(...)));
```

Le type : type_délai

Les types de délai du fichier « .dtx » sont :

- F : délai élémentaire sur des chemins internes à la figure ;
- E : délai élémentaire sur au moins un chemin externe à la figure ;
- D : délai élémentaire ayant comme extrémité un connecteur.

Les types de chemins factorisés du fichier « .ttx » sont :

- P : chemin factorisé sur des chemins internes à la figure ;
- J : chemin factorisé sur au moins un chemin externe à la figure ;
- T : chemin factorisé ayant comme extrémité un connecteur.

Les paramètres

- début_type : type de signal de l'extrémité initiale (C,N,Q,L,R,I,S) ;
- début_index : index du signal de l'extrémité initiale ;
- fin_type : type de signal de l'extrémité terminale (C,N,Q,L,R,I,S) ;
- fin_index : index du signal de l'extrémité terminale ;
- com_index : index de la commande dans le cas où l'extrémité terminale est un registre ou un bus préchargé rien sinon ;
- com_type : type de l'événement sur la commande, U si actif a l'état haut D sinon ;
- type_trans : type de transitions sur les extrémités (UU DD UD DU) ;
- type_délai : type de délai MAX ou MIN pour les portes ; IMAX ou IMIN pour les interconnexions ;
- délai : temps de propagation ;
- front : front sur l'extrémité terminale ;
- inf_elec : un ou deux paramètres de correction des délais de type T ou D par rapport au front d'entrée si l'extrémité initiale est un connecteur et par rapport à la capacité de sortie si l'extrémité terminale est un connecteur.

Marque de fin de fichier

```
G;
```

Exemple de net-list

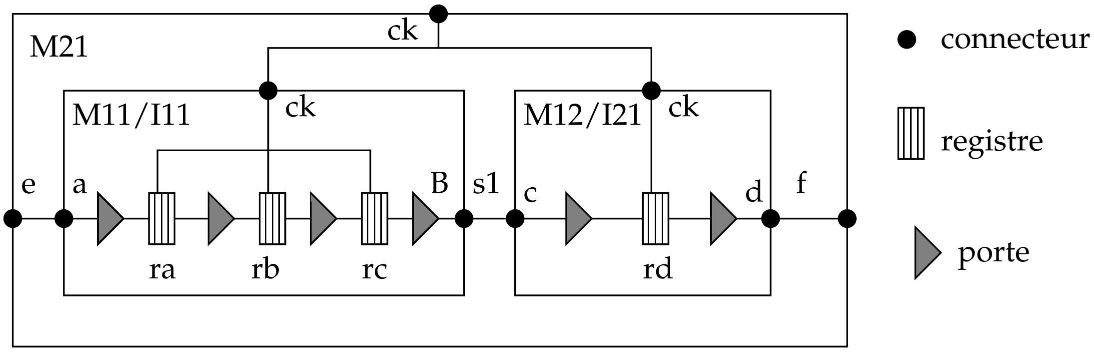


FIG. B.1 – *Exemple de net-list hiérarchique*

Fichiers « .dtx » et « .ttx » correspondants

L'analyse temporelle a été limitée aux délais et aux chemins les plus longs.

Il n'existe pas de point de factorisation dans cet exemple ; les graphes de causalité réduits sont donc équivalents aux graphes des chemins.

Les délais des réseaux d'interconnexion n'ont pas été pris en compte pour le modèle $m21$.

Fichier m11.dtx

```
H tas 5.34 m11 prol10 2.00 1000 0 1 (25 8 1998 10 59 7);

C IQ 1 ck (77 0 0 1000 1000);
C I 2 a (51 0 0 1000 1000);
C O 3 b (65 2875 3473 186 225);

Q C 1 ck;

L LE 1 ra (1 U);
L LE 2 rc (1 U);
L LI 3 rb (1 U);

S E 1 s3;
S I 2 src;
S I 3 srb;
S I 4 s2;
S I 5 sra;
S I 6 s1;

E S 2 S 1 (
```

```
((UD MAX 379 165))
((DU MAX 444 156)));

E L 2 S 2 (
((UD MAX 798 646))
((DU MAX 1188 771)));

F S 5 S 6 (
((UD MAX 379 165))
((DU MAX 444 156)));

F L 1 S 5 (
((UD MAX 798 646))
((DU MAX 1188 771)));

F S 3 S 4 (
((UD MAX 379 165))
((DU MAX 444 156)));

F S 6 L 3 (
(1 U(UD MAX 451 218))
(1 U(DU MAX 609 362)));

F L 3 S 3 (
((UD MAX 798 646))
((DU MAX 1188 771)));

F S 4 L 2 (
(1 U(UD MAX 451 218))
(1 U(DU MAX 609 362)));

D C 2 L 1 (
(1 U(UD MAX 556 218 394))
(1 U(DU MAX 851 362 426)));

D S 1 C 3 (
((UD MAX 222 133 2048))
((DU MAX 261 120 1848)));


G;
```

Fichier m11.ttx

```
H tas 5.34 m11 prol10 2.00 1000 0 1 (25 8 1998 10 59 7);

C IQ 1 ck (77 0 0 1000 1000);
C I 2 a (51 0 0 1000 1000);
C O 3 b (65 2875 3473 186 225);

Q C 1 ck;

L LE 1 ra (1 U);
L LE 2 rc (1 U);
L LI 3 rb (1 U);

P L 1 L 3 (
(1 U(DU MAX 2176 362))
(1 U(UD MAX 1693 218)));

P L 3 L 2 (
(1 U(DU MAX 2176 362))
(1 U(UD MAX 1693 218)));

T C 2 L 1 (
(1 U(DU MAX 851 362 426))
(1 U(UD MAX 556 218 394)));

T L 2 C 3 (
((DU MAX 1828 120 1848))
((UD MAX 1464 133 2048)));


G;
```

Fichier m12.dtx

```
H tas 5.34 m12 prol10 2.00 1000 0 1 (25 8 1998 10 59 9);

C IQ 1 ck (25 0 0 1000 1000);
C I 2 c (51 0 0 1000 1000);
C O 3 d (65 2875 3473 186 225);

Q C 1 ck;
```

```
L LE 1 rd (1 U);

S E 1 s1;
S I 2 srd;

E S 2 S 1 (
((UD MAX 379 165))
((DU MAX 444 156)));

E L 1 S 2 (
((UD MAX 798 646))
((DU MAX 1188 771)));

D C 2 L 1 (
(1 U(UD MAX 556 218 394))
(1 U(DU MAX 851 362 426)));

D S 1 C 3 (
((UD MAX 222 133 2048))
((DU MAX 261 120 1848)));


G;
```

Fichier m12.ttx

```
H tas 5.34 m12 prol10 2.00 1000 0 1 (25 8 1998 10 59 9);

C IQ 1 ck (25 0 0 1000 1000);
C I 2 c (51 0 0 1000 1000);
C O 3 d (65 2875 3473 186 225);

Q C 1 ck;

L LE 1 rd (1 U);

T C 2 L 1 (
(1 U(DU MAX 851 362 426))
(1 U(UD MAX 556 218 394)));

T L 1 C 3 (
((DU MAX 1828 120 1848))
((UD MAX 1464 133 2048)));
```

```
G;
```

Fichier m21.dtx

```
H tas 5.34 m21 prol10 2.00 1000 0 2 (25 8 1998 10 59 31);

X m12 i12;
X m11 i11;

C O 1 f (65 2875 3473 186 225);
C I 2 e (51 0 0 1000 1000);
C IQ 3 ck (102 0 0 1000 1000);

Q C 1 ck (i11.ck i12.ck);

L LE 1 i11.ra (1 U);
L LE 2 i12.rd (1 U);
L LI 3 i11.rc (1 U);

S E 1 i12.s1;
S E 2 i11.s3;
S I 3 s1;

F S 2 S 3 (
((UD MAX 326 237))
((DU MAX 355 214)));

F S 3 L 2 (
(1 U(UD MAX 294 218))
(1 U(DU MAX 597 362)));

D C 2 L 1 (
(1 U(UD MAX 556 218 394))
(1 U(DU MAX 851 362 426)));

D S 1 C 1 (
((UD MAX 222 133 2048))
((DU MAX 261 120 1848)));


G;
```

Fichier m21.ttx

```
H tas 5.34 m21 prol10 2.00 1000 0 2 (25 8 1998 10 59 31);

X m12 i12;
X m11 i11;

C O 1 f (65 2875 3473 186 225);
C I 2 e (51 0 0 1000 1000);
C IQ 3 ck (102 0 0 1000 1000);

Q C 1 ck (i11.ck i12.ck);

L LE 1 i11.ra (1 U);
L LE 2 i12.rd (1 U);
L LI 3 i11.rc (1 U);

P L 3 L 2 (
(1 U(UU MAX 2165 362))
(1 U(DD MAX 2216 218)));

T C 2 L 1 (
(1 U(DU MAX 851 362 426))
(1 U(UD MAX 556 218 394)));

T L 2 C 1 (
((DU MAX 1828 120 1848))
((UD MAX 1464 133 2048)));


G;
```

ANNEXE C

La détermination des paramètres technologiques du modèle MCC

Cette annexe présente la méthode de détermination des paramètres du modèle ***MCC***. Cette méthode a été mise au point en collaboration avec la compagnie BULL. Elle a été automatisée grâce à un programme écrit en langage C. Dans ce document, le modèle ***MCC*** est appelé ***TSCM*** (transistor short channel model). Ce document est extrait d'un rapport de recherche interne au Laboratoire ***LIP6*** rédigé par Mme Marie-Minerve Louërat.

The short channel transit or model *TSCM*

In ***TSCM*** model the drain current is expressed as follows :

$$\begin{array}{lr} V_{GS} < V_T \rightarrow I = 0; & \text{sub threshold mode} \\ V_{DS} < V_{SAT} \rightarrow I = V_{DS}/R_T; & \text{linear mode} \\ V_{DS} > V_{SAT} \rightarrow I_{SAT} = A(V_{GS} - V_T)^2/(1 + B(V_{GS} - V_T)); & \text{saturation mode} \\ V_{SAT} = K(V_{GS} - V_T) \text{ with } 0 < K < 1\ . & \text{saturation voltage} \end{array} \tag{C.1}$$

where V_T is the threshold voltage, V_{SAT} the saturation voltage, A and B are the transconductance parameters in the saturation region, R_T the resistance in the linear mode.
The following figures a show the comparison between Spice level 3 model, for a 0.8 μ technology and the ***TSCM*** model obtained from the this technology.

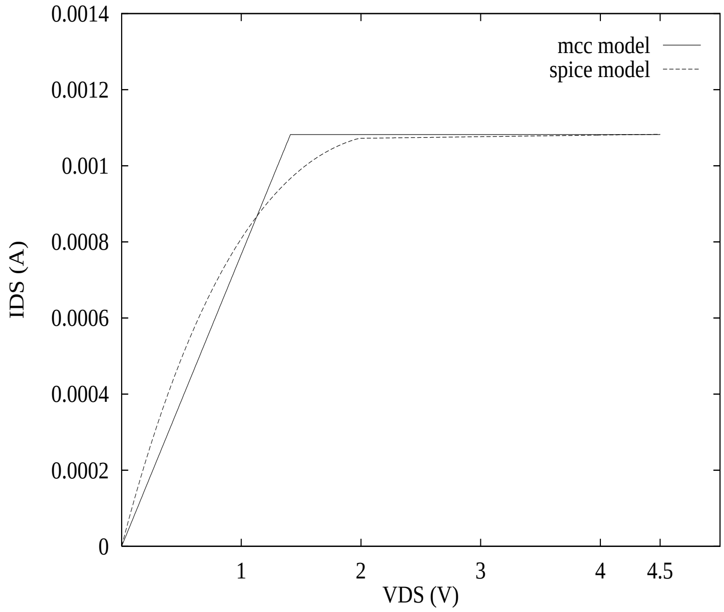


Figure C.1: Drain current versus voltage $I(V_{DS})$ for $V_{GS} = 4.5V$ of the ***TSCM*** model (NMOS) compared with the Spice level 3 model, for a 0.8 μ technology

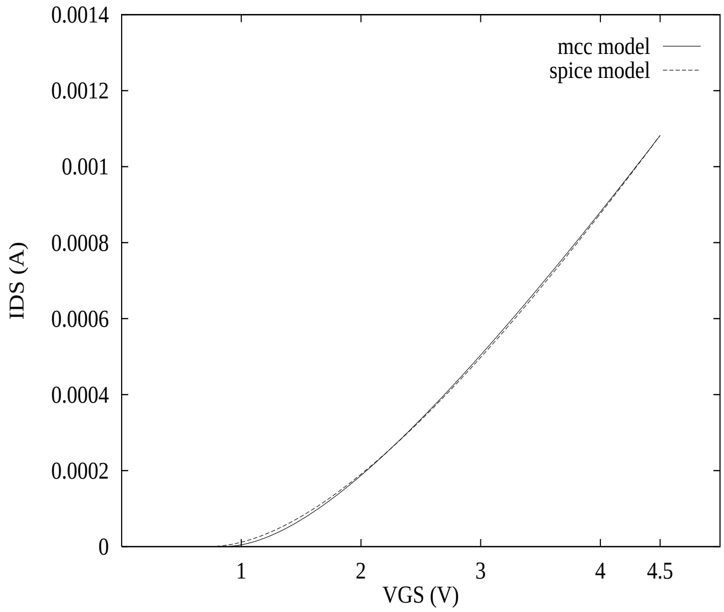


Figure C.2: Saturation current versus voltage $I(V_{GS})$ of the ***TSCM*** model (NMOS) compared with the Spice level 3 model, for a 0.8 μ technology

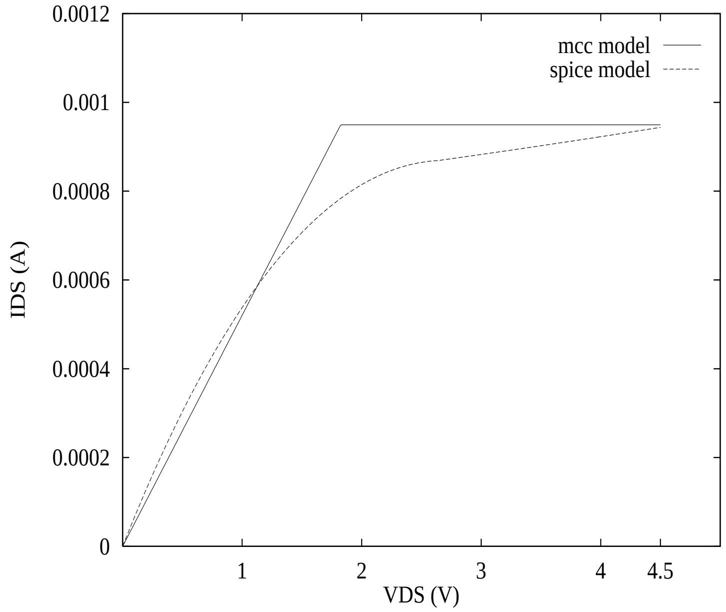


Figure C.3: Drain current versus voltage $I(V_{DS})$ for $V_{GS} = 4.5V$ of the ***TSCM*** model (PMOS) compared with the Spice level 3 model, for a 0.8 μ technology

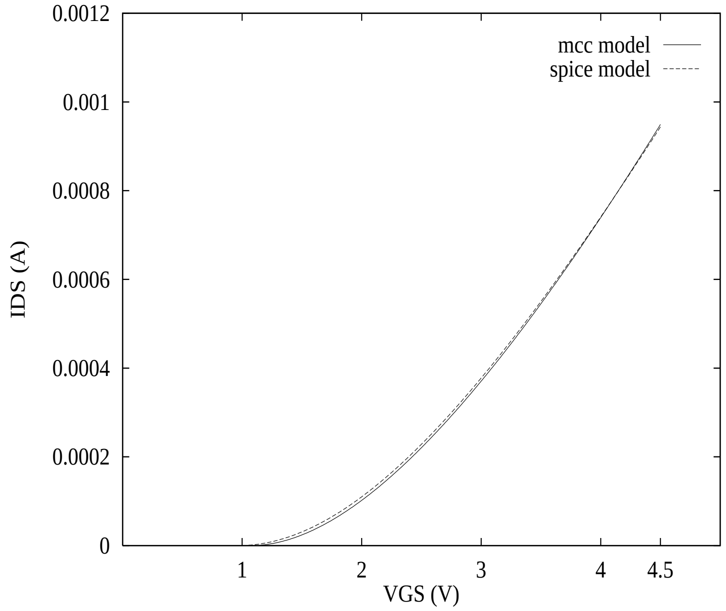


Figure C.4: Saturation current versus voltage $I(V_{GS})$ of the ***TSCM*** model (PMOS) compared with the Spice level 3 model, for a 0.8 μ technology

TAS technology File

Here we want to show how TAS is able to handle a large variety of technologies. In order to do this, TAS, as most of timing verification tools, uses SPICE-like simulations to compute its own technology file. The methodology used to make this technology file has improved a lot since the first development of TAS. Many people have contributed to the parameterization of TAS. Amjad Hajjar gave the first file [Haj92] for the technology he studied, Payam Kiani made the first attempt to define a general procedure of parameterization [Kia93] with Bull and Marcelo Duhalde at Bull explored the possibility of making the file in two steps : rough computation first and then accurate adjustment of sensitive parameters [Duh94]. We improved the methodology of parameterization by specifying fitting circuits and thus correcting bugs in the timing analyzer.

One example is given below “prol10.elp” for a fake technology “prol10”.

```
#TAS technology file, PROL10, LEVEL x, HSPICE
#shrinks are taken into account
#by DIOURY Karim, on february 20th 1997

Technology : PROL10 Version : 2.0

#Reference Simulator
ESIM  = HSPICE(META SOFTWARE)
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)
# surface effects
CGSN = 1500.0e-6
CGSP = 1500.0e-6
# perimeter effects
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
```

The detailed way to compute the parameters of TAS technology file is given in the following sections. The computation is worked out in two phases insuring the robustness of the parameters. The first one comes quite straightforwardly from the foundry electrical parameters (see *First determination of parameters*). The second one adjusts the more sensitive parameters of the TCSM model (see *Second determination of parameters*), considering the values of parameters resulting from the first phase as initial values.

First determination of parameters

One possible way to make a technology file for TAS (*.elp file) is to make a convenient simulation file, to run HSPICE [Met94], ELDO [Ana96] or some other electrical simulation tool and to use the output file of the simulation.

The simulation file contains :

- model of the transistors given by the foundry;
- nets of transistors chosen to allow measurements of TAS parameters. SPICE transient simulations are performed to measure :
  - *circuit 1* – transistor NMOS : IDS=f(VDS) with VGS=VDD
  - *circuit 2* – transistor NMOS : IDS=f(VGS) with VDS=VDD
  - *circuit 3* – transistor PMOS : IDS=f(VDS) with VGS=VDD
  - *circuit 4* – transistor PMOS : IDS=f(VGS) with VDS=VDD
  - *circuit 5* – pass transistor NMOS to VDD
  - *circuit 6* – pass transistor PMOS to VSS
- stimuli and control lines for measurements of TAS parameters.

The output file of the simulations is used to make the TAS technology file as follows :

- Reference Simulator : the actual simulator used for parameter computations
  - ESIM : name of the electrical simulator used;
  - MODEL : name of the model of transistor;
  - LEVEL : name of the model of transistor used.
- shrink parameters (microns) : from the foundry Process Parameters and written in the output file of the simulator in section concerning *parameters of the transistors*. The important matter is that in the following computations, the effective width of the transistor $Weff$ and the effective length of the transistor $Leff$, based on the shrink parameters are used;

$$D(LorW)(NorP) = (LorW)eff - (LorW)(designed) \tag{C.2}$$

- transistor characteristics : the parameters of the ***TSCM*** model (Eq C.1). They are given for the NMOS transistor first and then for the PMOS transistor :
  - VTN (or VTP) : Zero-bias threshold voltage of the transistor (volts), written in the output file of the simulator in section concerning *element parameters* of the transistors.
  - AN (or AP) : consider the parameter "beta" (HSPICE) or "K0" (ELDO) written in the output file of the simulator in section concerning *element parameters* of the transistors. Then :
    - HSPICE

$$A(NorP) = \beta(NorP)/2.(L/W) \tag{C.3}$$

∗ ELDO

$$AN = K0 \tag{C.4}$$

– BN (or BP) : depends on the maximum current between drain and source for each type of transistor. Results of *circuit 2* (see also Fig C.2) give the maximum current $I_{SAT} = IDS$ for $VGS = VDD$. Then :

$$B = (beta/2)((VDD - VT)/IDS) - (1/(VDD - VT)) \tag{C.5}$$

Same for PMOS transistor (*circuit 4*, Fig C.4).

– Linear resistance RNT (or RPT) : models the transistor in the linear mode. Results of *circuit1* illustrated Fig. C.1 give the drain current IDS for $VDS = VDDD/4$. Then :

$$RNT = ((VDD/4)/IDS(forVDS = VDD/4))(W/L) \tag{C.6}$$

same for PMOS transistor (RPT, *circuit 3*, Fig C.3).

- general parameters
  - VDDmax : power supply (volts), worse case;
  - VTHR : switching voltage (volts), often $VDD/2$;
  - VSSdeg : when the ground supply is accessed through a PMOS transistor, the resulting voltage at the transistor source is upper than VSS, it is called the degraded ground supply (*circuit 6*).
  - VDDdeg : when the power supply is accessed through a NMOS transistor, the resulting voltage at the transistor source is lower than VDD, it is called the degraded power supply (*circuit 5*).
  - TEMP : temperature needed for simulation (Celsius). Take the worse case one.
- dynamic capacitance : grid capacitance (in pF/u and pF/u2)
  - CGSN : oxide capacitance per unit area for NMOS transistor (pF/μ^2). The COX value, given in the foundry models of the transistors, is used.
  - CGSP : oxide capacitance per unit area for PMOS transistor (pF/μ^2), same as CGSN.
  - CGPN : gate-drain or gate-source overlap capacitance (pF/μ) for the NMOS transistor. The $CGDO$ value, given in the NMOS model of the transistor, is used.
  - CGPP : gate-drain or gate-source overlap capacitance (pF/μ) for the PMOS transistor. The $CGDO$ value, given in the PMOS model of the transistor, is used.

- dynamic capacitance : drain capacitance (in pF/u and pF/u2)
  - CDSN : junction capacitance (pF/μ^2) for the NMOS transistor. The CJ value, given in the NMOS model of the transistor, is used.
  - CDSP : junction capacitance (pF/μ^2) for the PMOS transistor. The CJ value, given in the PMOS model of the transistor, is used.
  - CDPN : sidewall junction capacitance (pF/μ) for the NMOS transistor. The $CJSW$ value, given in the NMOS model of the transistor, is used.
  - CDPP : sidewall junction capacitance (pF/μ) for the PMOS transistor. The $CJSW$ value, given in the PMOS model of the transistor, is used.
  - CDWN : sidewall junction facing gate capacitance (pF/μ) for the NMOS transistor. The $CGDO$ value, given in the NMOS model of the transistor, is used.
  - CDWP : sidewall junction facing gate capacitance (pF/μ) for the PMOS transistor. The $CGDO$ value, given in the PMOS model of the transistor, is used.
- dynamic capacitance : source capacitance (in pF/u and pF/u2) The values are the same as for the drain capacitances.
  - CSSN : same as CDSN;
  - CSSP : same as CDSP;
  - CSPN : same as CDPN;
  - CSPP : same as CDPP;
  - CSWN same as CDWN;
  - CSWP : same as CDWP.

Second determination of parameters

With this first determination of TAS technology parameters, running TAS and comparing with SPICE simulations shows that TAS can usually estimate the delay through a path to within 20 percent of SPICE. This lack of precision has two main sources :

- the determination of the conductance parameter A has been performed thanks to a static simulation;
- the values of the capacitances come directly from other models that the ***TSCM*** model, which is simpler and considers, in particular, that the capacitance values do not depend on the voltage and that the capacitances are set in between a node of the transistor (gate, drain or source) and the ground node.

An efficient way to improve the value of TAS parameters would be to find circuits that are sensitive to one single parameter. This is of course impossible. Yet it is possible to get close to this ideal state by designing small circuits that are much more sensitive to one particular ***TSCM*** model parameter than to the other parameters. To get rid of special effects due to the shape of the input rise, the circuits are always driven by six series connected inverters, the first of which is driven with a piece-wise linear function .

Figures C.5, C.6, C.7 show the fitting circuits.

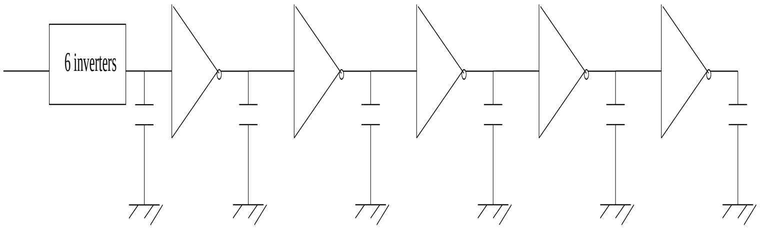


Figure C.5: Fitting the conductance parameter AN and AP

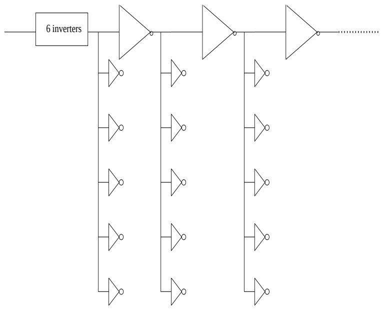


Figure C.6: Fitting gate capacitances CGXX

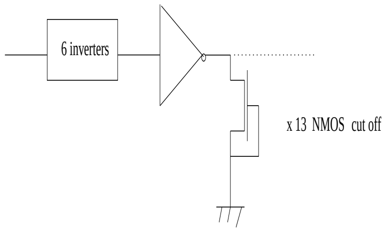


Figure C.7: Fitting the diffusion capacitance

For each special circuit designed to adjust one particular ***TSCM*** parameter, two delays are generated. One by running SPICE transient simulations the second by running TAS. The time delay given by SPICE $dspi$ is divided by the time delay given by TAS $dtas$ to produce the *correcting ratio* cr. The parameter under study $pfirst$ is modified in proportion to the correcting ratio to get his new value $pfit$:

$$cr = dspi/dtas\,; \qquad pfit = pfirst.cr; \tag{C.7}$$

Values resulting from the *First determination of parameters* are taken as initial values. Iteration is performed till the ***TSCM*** model fits the electrical simulator. Convergence is achieved generally in 3 steps.

- conductance parameter : AN (or AP). Five inverters, each one driving a large load, are connected in series (fig. C.5). The large load is chosen to get rid of the lack of precision on the gate and diffusion capacitances. The delays resulting from SPICE run are averaged over the first four inverters.
  - AN : take the delays resulting from rising input and falling output of inverter;
  - AP : take the delays resulting from falling input and rising output of inverter;
- Gate capacitances : consider the same five inverters, but replace the capacitive load by 5 inverters in parallel to increase the effect of gate capacitance versus diffusion capacitance (fig. C.6). All the gate capacitances (namely $CGSN, CGSP, CGPN, CGPP$) are modified by the same correcting ratio.
- Diffusion capacitances : their values are particularly important in the case of circuits with pass-transistors.
  - CDPN and CDSN of the NMOS transistor : one inverter is used, loaded with 13 NMOS transistors turned off to emulate such precharged circuits where one transistor in turned on in parallel with many transistors turned off (fig C.7). The average of the delay resulting from rising output and falling output is considered;
  - CDPP and CDSP of the PMOS transistor : one inverter is used, loaded with 13 PMOS transistors turned off to emulate such precharged circuits where one transistor in turned on in parallel with many transistors turned off . The average of the delay resulting from rising output and falling output is considered;

CDWN and CDWP are not modified.

Bibliographie

[AAD90] D. Auvergne, N. Azemard, D. Deschacht, and M. Robert. Input Waveform Slope Effects in CMOS Delays. *IEEE J. of Solid-State Circuits*, 25(6):1588–1590, December 1990.

[AGL+93] Amara Amara, Alain Greiner, Luis Lucas, Frédéric Pétrot, Franck Wajsbürt, and Laurent Winckel. The Portable Libraries of the Alliance CAD System. In *Proc. VIII Congresso da SB Micro*, pages III.14–III.18, Sau Paulo, Brasil, September 1993.

[AKH94] Nizar Abdallah, Payam Kiani, and Amjad Hajjar. New Algorithms for Static and Dynamic Sensitization of Paths in Timing Analysis. In *Proc. of the 26th Southeastern Symposium on System Theory*, pages 385–389, Athens, Ohio, March 1994. IEEE.

[AMD77] AMD. *Data sheet of the amd2901*, 1977.

[Ana96] Anacad - Mentor Graphics, Inc. *Eldo User's Manual*, 1996.

[BBG94] Pirouz Bazargan Sabet, Luc Burgun, Alain Greiner, and Frédéric Pétrot. Methodology and Development of a Complete CAD System for Digital VLSI Design. In *Proc. of the VIII Simposio Brasileiro de Concepcao de Circuitos Integrados*, pages 11–15, Gramado, Brasil, November 1994. *Invited paper*.

[BDS93] Yves Blaquière, Michel Dagenais, and Yvon Savaria. A New Accurate and Hierarchical Timing Analysis Approach. In *European Design Automation Conference*, pages 449–454, Hamburg, Germany, September 1993.

[BDS96] Yves Blaquière, Michel Dagenais, and Yvon Savaria. Timing Analysis Speed-up Using a Hierarchical and a Multimode Appproach. *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, 15(2):244–255, February 1996.

[BMC90] J. Benkoski, E. Vanden Meersch, L. J. M. Claesen, and H. De Man. Timing Verification Using Statically Sensitizable Paths. *IEEE Trans. on Computer-Aided Design*, 9(10):1073–1084, October 1990.

[Bry84] Randal E. Bryant. A Switch Level Model and Simulator for MOS Digital System. *IEEE Trans. on Computers*, 33(2):160–177, February 1984.

[BVF+96] Zerrouk Belkacem, Reibaldi Vincent, Potter Frédéric, Greiner Alain, and Derieux Anne. A gigabit serial links low latency adaptive router. In *In the Records of the IEEE Hot Interconnects IV*, Palo Alto CA, U.S.A, August 1996.

[CDG95] Michel Combes, Karim Dioury, and Alain Greiner. A Portable Clock Multiplier Generator using Digital CMOS Standard Cells. In *European Solid-State Circuits Conference*, pages 66–69, September 1995.

[CDG96] Michel Combes, Karim Dioury, and Alain Greiner. A Portable Clock Multiplier Generator using Digital CMOS Standard Cells. *IEEE Journal of Solid-State Circuits*, 31:958–965, July 1996.

[CGA97] S. Cremoux, J.L. Guntzel, N. Azemard, and D. Auvergne. Path Sensitization for Performance Optimization. In *Proc. Mixed Design of Integrated Circuits and Systems*, pages 195–200, Poznan, Pologne, June 1997.

[CGM+97] S. Cremoux, J.L. Guntzel, T. Monnier, N. Azemard, and D. Auvergne. Algorithme de sélection de chemins pour l'optimisation de performances. In *Proc. du Colloque CAO de circuits intégrés et systèmes*, pages 188–191, Grenoble, France, January 1997.

[Chi92] Messaoud Chiheb. *Analyse de la performance de systèmes électroniques complexes composés de circuits VLSI ASIC utilisant des latchs level-sensitive*. PhD thesis, Université Pierre et Marie Curie, Laboratoire MASI, 1992.

[CS96] V. Chandramouli and Karem A. Sakallah. Modeling the Effects of Temporal Proximity of Input Transitions on Gate Propagation Delay and Transition Time. In *Proc. of the 33rd ACM/IEEE Design Automation Conference*, Las Vegas, USA, June 1996.

[DA97] J.-M. Daga and D. Auvergne. Signal Transition Modeling in Submicronic CMOS Structures. In *Proc. PATMOS'97*, Louvain-La-Neuve, Belgium, September 1997.

[Dai97] Wayne W.-M. Dai. Chip Parasitic Extraction and Signal Integrity Verification. In *Proc. of the 34th ACM/IEEE Design Automation Conference*, pages 720–722, Anaheim, USA, June 1997.

[DD95] D. Deschacht and C. Dabrin. A New and Accurate Interconnection Delay Time Evaluation in a general Tree-Type Network. In *Proc. of the ASP-DAC'95*, 1995.

[Des96] Gilles-Eric Descamps. *Méthode de distribution hiérarchique d'outils de vérification de circuits intégrés VLSI sur un réseau de stations de travail, Application* *à un vérificateur de règles de dessin*. PhD thesis, Université Pierre et Marie Curie, Laboratoire MASI, France, June 1996.

[DGR97] Karim Dioury, Alain Greiner, and Marie-Minerve Rosset-Louërat. Analyse temporelle des circuits VLSI à haute densité d'intégration utilisant des technologies submicroniques. In *Proc. du Colloque CAO de circuits intégrés et systèmes*, pages 184–187, Grenoble, France, January 1997.

[DGR97] K. Dioury, A. Greiner, and M-M. Rosset-Louërat. Accurate Static timing Analysis for Deep Submicronic CMOS Circuits. In *Proc. of the VLSI'97*, Gramado, Brasil, August 1997. Chapman & Hall.

[DRL98] K. Dioury and M-M. Rosset-Louërat. Fitting the technology file for the static timing timing analyzer TAS. Technical report, Laboratoire LIP6, UPMC 55-65,bureau 213, 4,place Jussieu, 75252 Paris Cedex05, France, 1998.

[Duh94] Marcelo Duhalde. TAS: Manuel de paramétrisation. Technical report, Bull SA, Les Clayes sous Bois, France, July 1994.

[Elm48] W.C Elmore. The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers. *Journal of Applied Physics*, 19(1):55–63, January 1948. 1.

[EPI97] EPIC Design Technology. *PathMill*, 1997.

[FFG97] C. Forzan, B. Franzini, and C. Guardiani. Accurate and Efficient Macromodel for Submicron Digital Standard Cells. In *Proc. of the 34th ACM/IEEE Design Automation Conference*, pages 633–637, Anaheim, USA, June 1997.

[Fly97] M. J. Flynn. What's Ahead in Computer Design? In *Proc. of the 23rd Euro-Micro Conference*, pages 4–9, Budapest, September 1997.

[GDA97] W. J. Grundmann, D. Dobberpuhl, R. L. Allmon, and N. L. Rethman. Designing High Performance CMOS Microprocessors Using Full Custom Techniques. In *Proc. of the 34th ACM/IEEE Design Automation Conference*, pages 722–727, Anaheim, USA, June 1997.

[GLM92] Alain Greiner, Marc Laurentin, and Roland Marbot. DESB, a Functional Abstractor for CMOS VLSI Circuits. In *Proc. of the European Design Automation Conference*, pages 22–27, Hamburg, Germany, September 1992.

[GM79] Michel Gontron and Michel Minoux. *Graphes et Algorithmes*. Eyrolles, Paris, first edition, 1979.

[GP92] Alain Greiner and François Pêcheux. Alliance: A Complete Set of CAD Tools for Teaching Digital VLSI design. In *Proc. of the 3rd Eurochip Workshop on VLSI Design Training*, pages 230–237, Toledo, Spain, September 1992.

[GP94a] Alain Greiner and Frédéric Pétrot. Designing Portable Module Generator Using The Alliance CAD System. In *Proc. of the 26th Southeastern Symposium on System Theory*, pages 395–399, Athens, USA, March 1994. IEEE.

[GP94b] Alain Greiner and Frédéric Pétrot. A public domain high performances portable ROM generator. In *Proc. of the 20th Euromicro*, pages 414–419, Liverpool, September 1994. IEEE.

[GPb94] Alain Greiner, Frédéric Pétrot, and Franck Wajs bürt. The symbolic to real mask translation approach in the Alliance CAD system. Technical report, Laboratoire MASI - URA CNRS 818, UPMC 55-65,bureau 213, 4,place Jussieu, 75252 Paris Cedex05, France, April 1994.

[GTP97] Rohni Gupta, Bogdan Tutuianu, and Lawrence T. Pileggi. The Elmore Delay as a Bound for RC Trees with Generalized Input Signals. *IEEE Trans. on Computer-Aided Design*, 16(1):95–104, January 1997.

[Gui98] Jean-Bruce Guignet. *Abstraction Fonctionnelle des Composants VLSI*. PhD thesis, Université Pierre et Marie Curie, Laboratoire LIP6, March 1998.

[Haj92] Amjad Hajjar. *Modélisation des temps de propagation et analyse temporelle statique des circuits intégrés CMOS*. PhD thesis, Université Pierre et Marie Curie, Laboratoire MASI, July 1992.

[HMG91] A. Hajjar, R. Marbot, A. Greiner, and P. Kiani. TAS : An Accurate Timing Analyser for CMOS VLSI. In *Proc. of the The 2nd European Conference on Design Automation*, pages 261–265, Amsterdam, The Netherlands, February 1991.

[JCd92] P. Johannes, L. Claesen, and H. de Man. Performance through hierarchy in static timing verification. In J. van Leeuwen, editor, *Proc. 12th World Computer Congress*, pages 703–709, Madrid, Spain, September 1992. Elsevier Science Publishers B.V.

[Jou83a] N. Jouppi. Timing Analysis for nMOS VLSI. In *Proc. of the 20th ACM/IEEE Design Automation Conference*, pages 118–123, June 1983.

[Jou83b] N. Jouppi. TV: an nMOS Timing Analyser. In *Proc. of the IEEE 3rd Caltech Conference onVLSI*, pages 71–85, Pasadena, USA, March 1983.

[JR81] Paul Penfield Jr. and Jorge Rubinstein. Signal Delay In RC Tree Networks. In *Proc. of the 18thACM/IEEE Design Automation Conference*, pages 613–617, June 1981.

[KB97] Yuji Kukimoto and Robert K. Brayton. Exact Required Time Analysis via False Path Detection. In *Proc. of the 34th ACM/IEEE Design Automation Conference*, pages 220–225, Anaheim, USA, June 1997.

[KC66] T. I. Kirpatrick and N. R. Clark. PERT as an aid to logic design. *IBM J. Res. Develop.*, 10(2):135–141, March 1966.

[Kia93] Payam Kiani. TAS - Manuel de paramétrisation. Technical report, Laboratoire MASI - URA CNRS 818, UPMC 55-65,bureau 213, 4,place Jussieu, 75252 Paris Cedex05, July 1993.

[Lau94] Marc Laurentin. *Abstraction fonctionnelle de circuits intégrés CMOS*. PhD thesis, Université Pierre et Marie Curie, Laboratoire MASI, December 1994.

[Les94] Anthony Lester. Yagle : Yet another gate level extractor. Technical report, Laboratoire MASI - URA CNRS 818, UPMC 55-65,bureau 213, 4,place Jussieu, 75252 Paris Cedex05, September 1994.

[LGB97] Anthony Lester, Alain Greiner, and Pirouz Bazargan. Un Outil d'Evaluation de la Consommation Basé sur l'Extraction d'un Réseau de Portes Caractérisées. In *Proc. du Colloque CAO de circuits intégrés et systèmes*, pages 128–131, Grenoble, France, January 1997.

[LGT93] Luis Lucas, Alain Greiner, and Michel Thill. High speed adder generator. In *proc. of the 5th International Conference on Microelectronic*, pages 136–139, Dhahran, Saoudian Arabia, December 1993.

[Met94] Meta-Software, Inc. *HSPICE Users Manual*, 1994.

[MSN92] M.R.Dagenais, S.Gaiotti, and N.C.Rumin. Transistors-Level Estimation of Worst-Case Delays in MOS VLSI Circuits. *IEEE Trans. on Computer-Aided Design*, 11(3):384–394, March 1992.

[Nag75] L. W. Nagel. Spice2 : A computer program to simulate semiconductor circuits. Technical report, University of California, Berkekey, CA, May 1975.

[Nap96] Simon Napper. Static timing analysis – a demanding solution. *Electronic Engineering*, pages 35–42, January 1996.

[Ous83] John K. Ousterhout. Crystal : a Timing Analyzer for nMOS VLSI Circuits. In R. Bryant, editor, *Proc. 3rd Caltech Conference on VLSI*, pages 57–70, Pasadena, USA, March 1983. Computer Science Press.

[Ous84] John K. Ousterhout. Switch-Level Delay Models for Digital MOS VLSI. In *proc. of the 21st ACM/IEEE Design Automation Conference*, pages 542–548, Albuquerque, USA, June 1984.

[Ous85] John K. Ousterhout. A Switch-Level Timing Verifier for Digital MOS VLSI. *IEEE Trans. on Computer-Aided Design*, CAD-4(3):336–349, July 1985.

[Pét94] Frédéric Pétrot. *Outils d'aide au développement de Bibliothèques VLSI portables*. PhD thesis, Université Pierre et Marie Curie, Laboratoire MASI, July 1994.

[PR90] Lawrence T. Pillage and Ronald A. Rohrer. Asymptotic Waveform Evaluation for Timing Analysis. *IEEE Trans. on Computer-Aided Design*, 9(4):352–366, April 1990.

[Pru95] Linda Pruitt. Verifying Mega-Gate IC Designs. *Electronic Engineering*, pages 67–68, May 1995.

[QPP94] Jessica Qian, Satyamurthy Pullela, and Lawrence Pillage. Modeling the "Effective Capacitance" for the RC Interconnect of CMOS Gates. *IEEE Trans. on Computer-Aided Design*, 13(12):1526–1535, December 1994.

[RGA94] Philippe Royannez, Alain Greiner, and Amara Amara. Booth multi-pipelined multiplier generator. In *Proc. of the IX Congressoda Sociedade Brasileira de Microelectronica (SBMICRO)*, pages 686–689, Rio de Janeiro, Brasil, August 1994.

[RPH83] Jorge Rubinstein, Paul Penfield JR., and Mark A. Horowitz. Signal Delay in RC Tree Networks. *IEEE Trans. on Computer-Aided Design*, 2(3):202–211, July 1983.

[SCD89] S.Perremans, L. Claesen, and H. De Man. Static Timing Analysis of Dynamically Sensitizable Paths. In *Proc. of the 26th ACM/IEEE Design Automation Conference*, pages 568–573, June 1989.

[Sch95] Steven E. Schulz. Timing Analysis Tools and Trends. *Integrated System Design Magazine*, November 1995.

[SN90a] Takayasu Sakurai and A. Richard Newton. Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas. *IEEE J. of Solid-State Circuits*, 25(2):584–594, April 1990.

[SN90b] Takayasu Sakurai and A. Richard Newton. A simple short-channel MOSFET model and its application to delay analysis of inverters and series-connected MOSFETs. In *Proc. of the ISCAS*, pages 105–108, May 1990.

[SN91] Takayasu Sakurai and A. Richard Newton. Delay Analysis of Series-Connected MOSFET Circuits. *IEEE J. of Solid-State Circuits*, 26(2):122–131, February 1991.

[SVG96] A. L. Sangiovanni-Vincentelli, P. C. Mc Geer, and A. Saldanha. Verification of Electronic Systems. In *Proc. of the 33rd ACM/IEEE Design Automation Conference*, Las Vegas, USA, June 1996.

[Tiw94] Gyanendra Tiwary. Below the Half-Micron Mark. *Spectrum*, 31(11), November 1994.

[VD97] E. Vanier and D. Deschacht. Delay Propagation Term for Submicron Interconnect. In *Proc. of the PATMOS'97*, Louvain-La-Neuve, Belgium, September 1997.

[VM98] David Van Campenhout and Trevor Mudge. Faster static timing analysis via bus compression. Technical Report CSE-TR-285-96, The University of Michigan, Dept. of Electrical Engineering and Computer Science Ann Arbor, Michigan, USA, 1998.

[WDP97] Franck Wajsbürt, Karim Dioury, and Frédéric Pétrot. Low Power, Process Independent, Full Transistor Controlled Slew Rate, PCI Compliant I/O Pads. In *Proc. of the 21 st International Conference on Microelectronics*, volume 2, pages 811–814, Niš, Yougosliavia, September 1997. IEEE.

[WDS+97] F. Wajsbürt, J-L. Desbarbieux, C. Spasevski, S. Penain, and A. Greiner. An Integrated PCI component for IEEE 1355 Networks. In *Proc. of the European, Multimedia Microprocessor Systems and Electronic Commerce Conference and Exhibition*, Florence, Italy, November 1997.

[WPD98] Franck Wajsbürt, Frédéric Pétrot, and Karim Dioury. Transistor Controlled Slew Rate Process Independent PCI Compliant I/O Buffer with Possible Power/Delay Trade-off. *Microelectronics journal*, 1998. Accepted for publication.

[YH95] Hakan Yalcin and John P. Hayes. Hierarchical Timing Analysis Using Conditional Delays. In *ICCAD-95 Digest of Technical Papers*, pages 371–377, San Jose, Californie, November 1995.