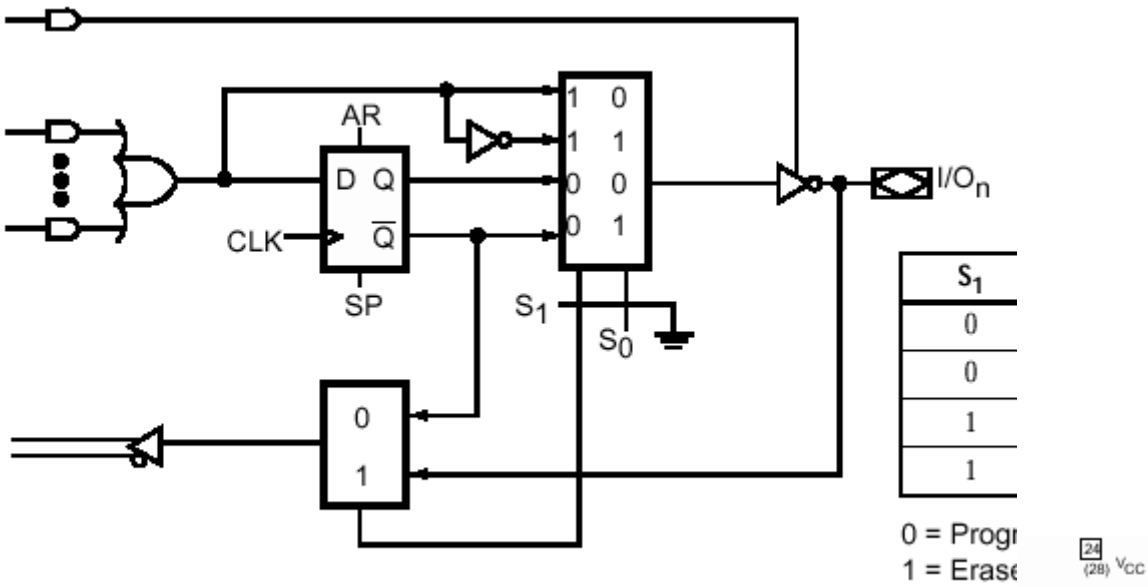
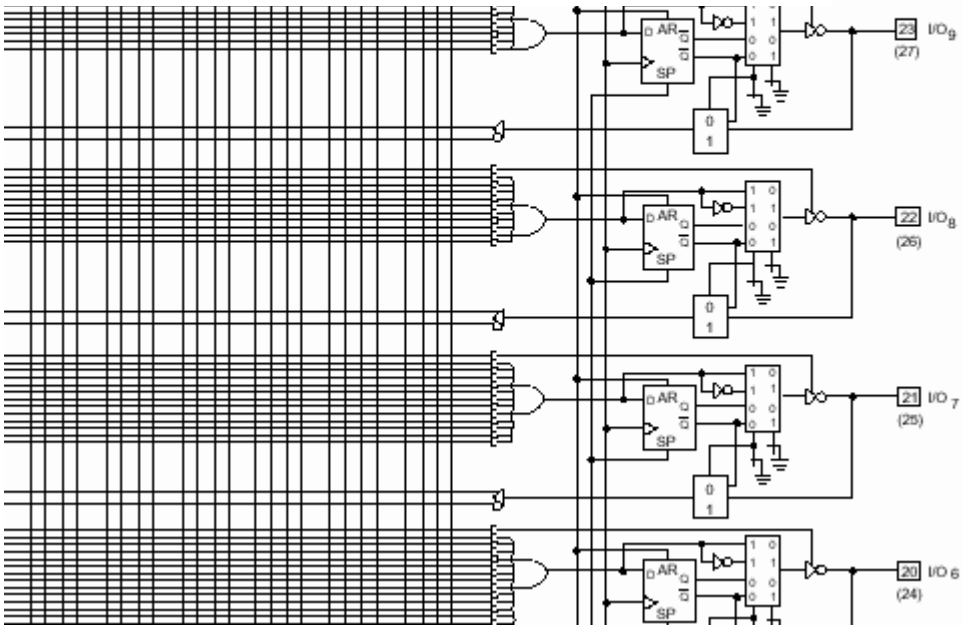
CACHAN
Département d'électronique

Circuits logiques programmables





Les circuits logiques programmables

L'histoire des circuits programmables commence, à peu de chose près, avec la décennie quatre-vingt. A la fin des années soixante-dix le monde des circuits numériques se répartit schématiquement en quatre grands groupes :

- Les fonctions standard sont utilisées pour les applications réalisées en logique câblée. Les catalogues TTL et CMOS présentent plusieurs centaines de fonctions d'usage général, sous forme de circuits intégrés à grande échelle.
- Les microprocesseurs, désormais d'usage courant, et sont omniprésents dans les applications industrielles.
- Dans des applications trop complexes pour être raisonnablement traitées en logique câblée traditionnelle, et trop rapides pour avoir une solution à base de microprocesseurs, on utilise des séquenceurs micro programmés.
- Quand les volumes de production importants le justifient, les circuits intégrés spécifiques (ASICS) offrent une alternative aux cartes câblées classiques.

Le développement des microprocesseurs stimule une évolution rapide des technologies de réalisation des mémoires à semi-conducteurs ; les circuits logiques programmables ont hérité directement des mémoires pour ce qui concerne les aspects technologiques. Leurs architectures internes sont, en revanche, très différentes. Il n'est donc pas surprenant que le premier fabriquant de circuits programmables ait été un fabriquant de mémoires (MMI, *monolithic memories inc*.). Notons, pour la petite histoire, que cette société a « fusionné[1] » avec l'un des leaders des fabriquants de processeurs rapides, processeurs micro programmés, processeurs spécialisés dans le traitement de signal, processeurs RISCs, AMD (*advanced micro devices*).

Ce chapitre est consacrée à un tour d'horizon général au cours duquel nous tenterons de donner au lecteur une vision d'ensemble de ce que sont les membres de la famille (nombreuse) des circuits programmables.

Les aspects structurels internes ne seront abordés que dans la mesure où ils éclairent la compréhension du fonctionnement. L'utilisateur d'un circuit a principalement besoin de bien dominer son architecture, les technologies de fabrication appartiennent au domaine du fondeur.

1. Les grandes familles

Indépendamment de sa structure interne et des détails de la technologie concernée, une mémoire est caractérisée par son mode de programmation et sa faculté de retenir l'information quand l'alimentation est interrompue. Les catégories de mémoires qui ont donné naissance aux circuits programmables sont :

- Les mémoires de type PROM (*programmable memory*) sont programmables une seule fois au moyen d'un appareil spécial, le programmateur[2]. Les données qui y sont inscrites ne sont pas modifiables. Elles conservent les informations quand l'alimentation est interrompue.
  Leur inconvénient majeur est l'impossibilité de modifier les informations qu'elles contiennent.
- Les mémoires de type EPROM (*erasable programmable memory*) sont programmables par l'utilisateur au moyen d'un programmateur, effaçables par une exposition aux rayons ultraviolets et reprogrammables après avoir été effacées. Elles aussi conservent les informations quand l'alimentation est interrompue.

[1]Fusion entre un géant et un nain.

[2]Quelle que soit la technologie, la programmation d'un circuit consiste à lui faire subir des niveaux de tension et de courant qui sortent du cadre de son utilisation normale. Typiquement, les tensions mises en jeu vont de 10 (CMOS) à 20 volts (bipolaires), les courants correspondants de quelques dizaines à quelques centaines de milliampères, pour des circuits alimentés sous 5 volts en fonctionnement ordinaire.

Leur boîtier doit être équipé d'une fenêtre transparente, ce qui en augmente le coût. La modification de leur contenu est une opération longue qui nécessite des manipulations : plusieurs minutes pour l'effacement des données anciennes, sur un premier appareil, et transfert de nouvelles informations sur un second appareil.

– Les mémoires de type EEPROM (*electricaly erasable programmable memory*), ou FLASH, sont effaçables et reprogrammables électriquement. Non alimentées, elles conservent les informations mémorisées.
  La diminution des tensions à appliquer pour programmer les mémoires FLASH permet même de s'affranchir du programmateur : il est intégré dans le circuit. On parle alors de mémoires programmables *in situ* (ISP, pour *in situ programming*), c'est à dire sans démonter la mémoire de la carte sur laquelle elle est implantée.
  Même programmable in situ, une mémoire FLASH fonctionne dans un mode tout à fait différent du mode « utilisation » quand elle est en cours de programmation. Les technologies FLASH sont de loin les plus séduisantes pour les circuits programmables pas trop complexes.
– Les mémoires RAM (*random access memory*) statiques[3], ou SRAM, sont constituées de cellules accessibles, en mode normal, en lecture et en écriture. Elles sont utilisées dans certains circuits programmables complexes pour conserver la configuration (qui définit la fonction réalisée) du circuit.
  Ces mémoires perdent leur information quand l'alimentation est supprimée. Les circuits qui les utilisent doivent donc suivre un cycle d'initialisation à chaque mise sous tension. Ces circuits peuvent être reconfigurés dynamiquement, changeant ainsi de fonction à la demande, en cours d'utilisation.

Comme tout domaine spécialisé, le monde des circuits programmables comporte une terminologie, d'origine anglo-saxonne le plus souvent, difficile à éviter. Pour compliquer les choses, de nombreux termes sont à l'origine des noms propres, propriétés des sociétés qui sont à la source des produits concernés. Qui se souvient encore que frigidaire est un nom déposé par la société General Motors ?

Les sigles utilisés dans la suite semblent communément admis par la majorité des fabricants :

– PLD (*programmable logic device*) est un terme générique qui recouvre l'ensemble des circuits logiques programmables. Il est le plus souvent employé pour désigner les plus simples d'entre eux (équivalent de quelques centaines de portes logiques).
– CPLD (*complex programmable logic device*) désigne évidemment un circuit relativement complexe (jusqu'à une ou deux dizaines de milliers de portes), mais dont l'architecture dérive directement de celle des PLDs simples.
– FPGA (*field programmable gate array*) marque un saut dans l'architecture et la technologie, il désigne un circuit qui peut être très complexe (jusqu'à cent mille portes équivalentes) ; la complexité des FPGAs rejoint celle des ASICs (*application specific integrated circuits*).

Notons que la complexité d'un circuit n'est pas mesurable simplement : il ne suffit pas qu'un circuit contienne un grand nombre de portes pour être puissant ; encore faut-il que ces portes soient utilisables dans une grande proportion. Ce dernier point est à la fois un problème d'architecture et de logiciels d'aide à la conception.

2. Qu'est-ce qu'un circuit programmable ?

Un circuit programmable est un assemblage d'opérateurs logiques combinatoires et de bascules dans lequel la fonction réalisée n'est pas fixée lors de la fabrication. Il contient potentiellement la

[3]L'adjectif statique s'oppose à dynamique. Les mémoires dynamiques stockent les informations dans des capacités - intrinsèquement liées aux structures MOS - qui nécessitent un processus dynamique de rafraîchissement. Les cellules des mémoires statiques sont des bistables, qui conservent leur état tant que l'alimentation est présente.

possibilité de réaliser toute une classe de fonctions, plus ou moins large suivant son architecture. La programmation du circuit consiste à définir une fonction parmi toutes celles qui sont potentiellement réalisables.

Comme dans toute réalisation en logique câblée, une fonction logique est définie par les interconnexions entre des opérateurs combinatoires et des bascules (synchrones, cela va presque sans dire), et par les équations des opérateurs combinatoires. Ce qui est programmable dans un circuit concerne donc les interconnexions et les opérateurs combinatoires. Les bascules sont le plus souvent de simples bascules D, ou des bascules configurables en bascules D ou T.

La réalisation d'opérateurs combinatoires utilise des opérateurs génériques, c'est à eux que nous allons nous intéresser dans la suite.

2.1 Des opérateurs génériques

Les opérateurs combinatoires génériques qui interviennent dans les circuits programmables proviennent soit des mémoires (réseaux logiques) soit des fonctions standard (multiplexeurs et ou exclusif).

Réseaux logiques programmables

Un réseau logique programmable (PLA, pour *programmable logic array*[4]) utilise le fait que toute fonction logique combinatoire peut se mettre sous forme d'une somme (OU logique) de produits (ET logique), c'est ce que l'on appelle classiquement la première forme normale, ou forme disjonctive.

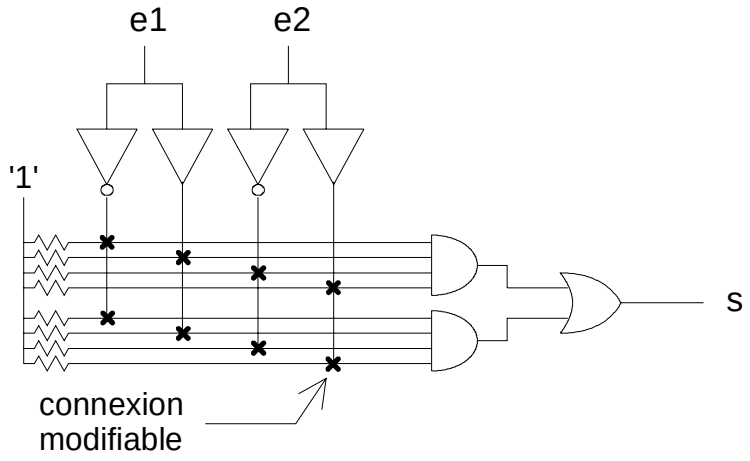


Figure1 : Un PLA simple

Le schéma de la figure III-1 représente une structure de PLA simple. La programmation du circuit consiste à supprimer certaines des connexions marquées d'une croix. Si une connexion est supprimée, une valeur constante '1' est appliquée à l'entrée correspondante de la porte ET, c'est ce que symbolise le réseau de résistances relié à cette valeur constante.

[4]Que l'on ne confondra pas avec PAL (*programmable array logic*), qui désigne les PLDs historiques de MMI...Ni avec GAL (*gate array logic*) nom déposé par la société Lattice, etc.

Un tel schéma permet de réaliser n'importe quelle fonction booléenne s(e1,e2), de deux variables binaires e1 et e2[5], pourvu qu'elle ne dépasse pas deux termes.

En effet, si toutes les connexions sont présentes, en notant par + et ∗ les opérateurs OU et ET, respectivement, s s'écrit :

$$s(e1,e2) = \overline{e1} * e1 * \overline{e2} * e2 + \overline{e1} * e1 * \overline{e2} * e2$$

qui vaut trivialement '0'.

Un opérateur ou exclusif, par exemple, obéit à l'équation :

$$e1 \oplus e2 = \overline{e1} * e2 + e1 * \overline{e2}$$

d'où la programmation du PLA de la figure III-2.

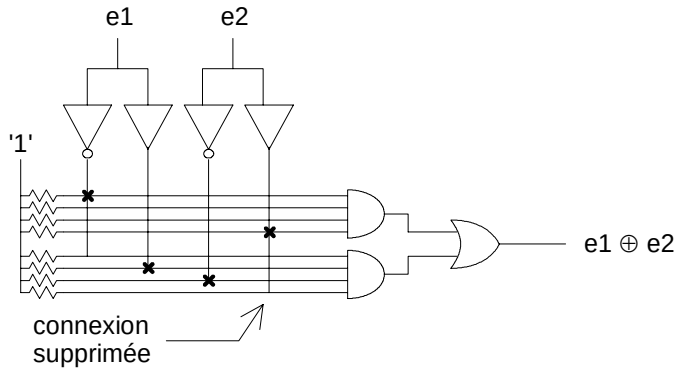


Figure 2 : PLA réalisant un ou exclusif

Le plus simple des PLDs, un 16L8 par exemple, utilise des opérateurs ET à 32 entrées et des opérateurs OU à 8 entrées. Un schéma tel que celui des figures précédentes deviendrait, dans de telles conditions, illisible. Pour éviter cet écueil, les notices de circuits utilisent des symboles simplifiés, pour représenter les réseaux logiques programmables.

La figure III-3 représente le PLA précédent avec ces symboles.

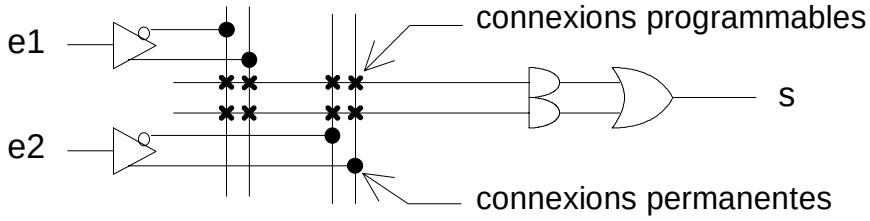


Figure 3 : symbole d'un PLA 2×4[6]

[5]Dans tout cet ouvrage, sauf précision contraire, nous utiliserons les valeurs 0 et 1 pour représenter les états possibles d'une variable binaire. Les opérateurs ET et OU sont définis avec la convention 0↔FAUX et 1↔VRAI. Les circuits associent bien sûr les valeurs logiques à des niveaux électriques ; sauf précision contraire, nous prendrons une convention logique positive qui associe 1 à un niveau haut (H pour *high*) et 0 à un niveau bas (L pour *low*).

[6]Le nombre 2×4 indique la dimension de la matrice de fusibles : deux lignes et quatre colonnes.

Dans un tel schéma, toutes les entrées (et leurs compléments) peuvent être connectées à tous les opérateurs ET par programmation. Par référence à la première technologie utilisée, ces connexions programmables portent le nom de fusibles, même quand leur réalisation n'en comporte aucun. Quand il s'agit uniquement d'illustrer la structure d'un circuit programmable, et non la réalisation d'une fonction particulière, les croix qui symbolisent les fusibles ne sont même pas représentées.

Dans cette évocation simplifiée, le schéma de l'opérateur ou exclusif devient celui de la figure III-4, dans laquelle une croix représente une connexion programmable maintenue, l'absence de croix une connexion supprimée.

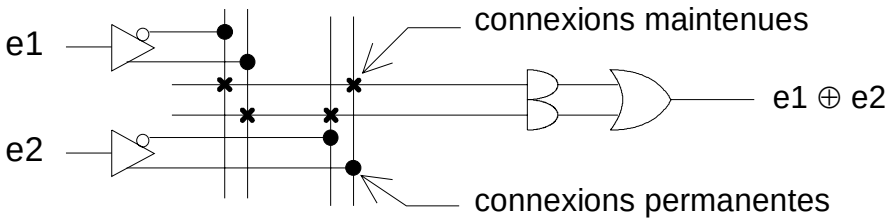


Figure 4 : PLA 2×4 réalisant un ou exclusif

Multiplexeurs

Un multiplexeur est un aiguillage d'informations. Dans sa forme la plus simple, il comporte deux entrées de données, une sortie et une entrée de sélection, conformément au symbole de la figure III-5.

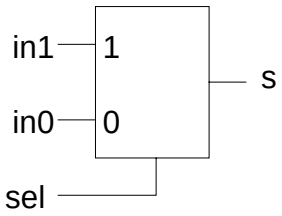


Figure 5 : un multiplexeur élémentaire

Le fonctionnement de cet opérateur se décrit très simplement sous une forme algorithmique :

si sel = '0' s ← in0 ;
si non (sel = '1') s ← in1 ;

Certains constructeurs notent sur le symbole, comme nous l'avons fait, la valeur de l'entrée de sélection en regard de l'entrée correspondante.

La première utilisation des multiplexeurs dans les circuits programmables est, évidemment, de créer des chemins de données. La programmation consiste alors à fixer des valeurs aux entrées de sélection.

Une autre utilisation de la même fonction consiste à remarquer qu'un multiplexeur est, en soi, un opérateur générique. Reprenant l'exemple précédent du *ou exclusif*, on peut le décrire sous forme algorithmique :

si e1= ‘0’ $\quad e1 \oplus e2 \leftarrow e2$;
si non (e1 = ‘1’) $\quad e1 \oplus e2 \leftarrow \overline{e2}$;

D’où une réalisation possible de l’opérateur *ou exclusif* au moyen d’un multiplexeur dans le schéma de la figure III-6.

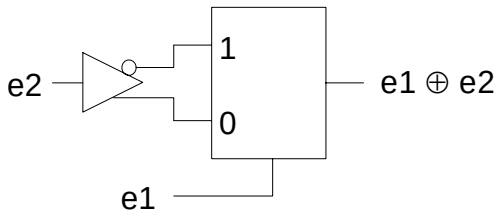


Figure 6 : ou exclusif réalisé par un multiplexeur

L’exemple précédent peut être généralisé sans peine : un multiplexeur à n entrées de sélection, soit 2^n entrées de données, permet de réaliser n’importe quelle fonction combinatoire de n + 1 entrées, pourvu que l’une, au moins, de ces entrées existe sous forme directe et sous forme complémentée.

Dans un circuit programmable dont les « briques » de base sont des multiplexeurs (c’est le cas de beaucoup de FPGAs) la programmation consiste à fixer des chemins de données, c’est à dire à établir des interconnexions entre des cellules de calcul et des signaux d’entrée et de sortie. Cette opération de création d’interconnexions entre des cellules génériques s’appelle le *routage* d’un circuit ; l’affectation des cellules à des fonctions souhaitées par l’utilisateur s’appelle le *placement.*

Ou exclusif

L’opérateur élémentaire *ou exclusif*, ou somme modulo 2, dont nous avons rappelé l’expression algébrique précédemment, est disponible en tant que tel dans certains circuits.

Cet opérateur intervient naturellement dans de nombreuses fonctions combinatoires reliées de près ou de loin à l’arithmétique : additions et soustractions, contrôles d’erreurs, cryptages en tout genre, etc. Or ces fonctions se prêtent mal à une représentation en somme de produits, car elles ne conduisent à aucune minimisation de leurs équations[7]. Un simple générateur de parité sur 8 bits, qui rajoute un bit de parité à un octet de données, ne nécessite pas moins de 128 (2^{8-1}) produits, quand il est « mis à plat », pour être réalisé au moyen d’une couche de ETs et d’une couche de OUs. De nombreuses familles de circuits programmables disposent, en plus des PLAs, d’opérateurs *ou exclusifs* pour faciliter la réalisation de ces fonctions arithmétiques.

Une autre application de l’opérateur *ou exclusif* est la programmation de la polarité d’une expression. Quand on calcule une fonction combinatoire quelconque sous forme disjonctive, il peut arriver qu’il soit plus économique, en nombre de produits nécessaires, de calculer le complément de la fonction et de complémenter le résultat obtenu.

Par exemple, si $(cba)_2$ représente l’écriture en base 2 d’un nombre N, compris entre 0 et 7, on peut exprimer par une équation logique que N est premier avec 3 :

$$prem3 = c * \overline{b} + \overline{b} * a + c * a + \overline{c} * b * \overline{a}$$

[7]Au moyen de tableaux de Karnaugh, ou, de façon plus réaliste, par des programmes de minimisation qui utilisent des algorithmes comme celui de Queene et Mc Cluskey ou ESPRESSO.

On peut également remarquer que si N est premier avec 3 c'est qu'il n'est pas multiple de 3, soit :

$$prem3 = \overline{mul3} = \overline{\overline{c} * \overline{b} * \overline{a} + \overline{c} * b * a + c * b * \overline{a}}$$

La deuxième forme, apparemment plus complexe, nécessite un produit de moins que la première pour sa réalisation. Dans des circuits où le nombre de produits disponibles est limité, cela peut présenter un avantage.

Un opérateur *ou exclusif* permet de passer, par programmation, d'une expression à son complément, comme l'indique la figure III-7. Comme cet opérateur peut être réalisé avec un multiplexeur, l'une ou l'autre de ces formes peut se trouver dans les notices !

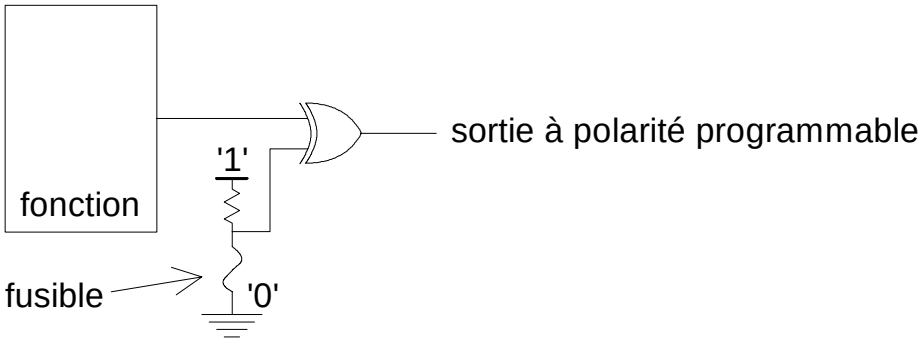


Figure 7 : polarité programmée par un ou exclusif

Bascules

Qui dit logique dit logique séquentielle. Les circuits programmables actuels offrent tous la possibilité de créer des fonctions séquentielles, synchrones dans leur immense majorité. La brique de base de toute fonction séquentielle est la bascule, cellule mémoire élémentaire susceptible de changer d'état quand survient un front actif de son signal d'horloge.

Bascule D, T ou J-K ? La première est toujours présente. Comme certaines fonctions se réalisent plus simplement avec la seconde, les compteurs par exemple, de nombreux circuits permettent, toujours par programmation, de choisir entre bascule D et bascule T[8], voire entre l'un des trois types de base. La figure III-8 rappelle, par un diagramme de transitions, le fonctionnement de ces trois types de bascules.

[8]C'est un (bon) exercice de logique séquentielle élémentaire que de trouver le schéma de n'importe quel type de bascule en utilisant n'importe quel autre type.

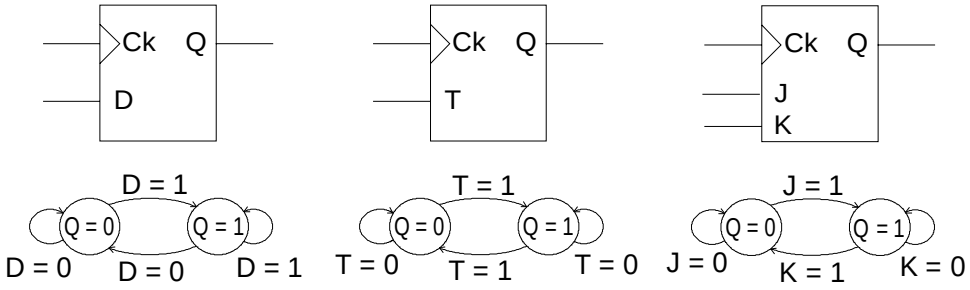


Figure 8 : les trois types fondamentaux de bascules

Le programmeur n'a, en réalité, que rarement à se préoccuper de ce genre de choix, les optimiseurs déterminent automatiquement le type de bascule le mieux adapté à l'application.

2.2 Des technologies

Premier critère de choix d'un circuit programmable, la technologie utilisée pour matérialiser les interconnexions détermine les aspects électriques de la programmation : maintien (ou non) de la fonction programmée en l'absence d'alimentation, possibilité (ou non) de modifier la fonction programmée, nécessité (ou non) d'utiliser un appareil spécial (un programmateur, bien sûr).

Fusibles

Première méthode employée, la connexion par fusibles, est en voie de disparition. On ne la rencontre plus que dans quelques circuits de faible densité, de conception ancienne.

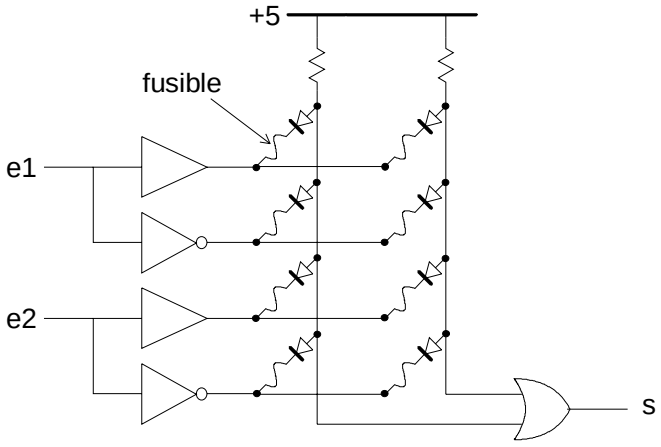


Figure 9 : Pld élémentaire à fusibles

La figure III-9 en illustre le principe ; toutes les connexions sont établies à la fabrication. Lors de la programmation le circuit est placé dans un mode particulier par le programmateur[9], mode dans lequel des impulsions de courant sont aiguillées successivement vers les fusibles à détruire.

[9]Ce mode est activé par une tension supérieure à la normale, appliquée sur une broche particulière du circuit. Dans ce mode, les autres broches servent à fournir au circuit le numéro du fusible à détruire et à appliquer une impulsion qui

Pour programmer un circuit, il faut transférer dans le programmateur une table qui indique par un chiffre binaire l'état de chaque fusible : la table des fusibles. Cette table est généralement transférée entre le système de CAO et le programmateur sous forme d'un fichier au format normalisé : le format JEDEC.

L'exemple ci-dessous est un extrait du fichier JEDEC, généré par un compilateur VHDL, qui implémente un compteur binaire 10 bits dans un circuit de type 22V10 :

```
 C22V10*
QP24*                Number of Pins*
QF5828*              Number of Fuses*
F0*                  Note: Default fuse setting 0*
G0*                  Note: Security bit Unprogrammed*
NOTE DEVICE C22V10*
NOTE PACKAGE PAL22V10G-5PC*
NOTE PINS  hor:1 oe:2 en:3 raz:4 compte_6:14 compte_8:15 compte_9:16
compte_3:17 *
NOTE PINS  compte_1:18 compte_0:19 compte_2:20 compte_4:21 compte_7:22 *
NOTE PINS  compte_5:23 *
NOTE NODES *
L00000
00000000000000000000000000000000000000000000
* Node hor[1] => BANK : 1 *

L00044
11110111111111111111111111111111111111111111
11011111011010101110111011101111111111111111
11101111110110111111111111111111111111111111
11101111111110111111111111011111111111111111
11101111111111001111111111111111111111111111
11101111111110111111110111111111111111111111
11101111111110111101111111111111111111111111
11101111101110111111111111111111111111111111
00000000000000000000000000000000000000000000
* Node compte_5[23] => OE : 1 ,LOGIC : 8 *

L00440
11110111111111111111111111111111111111111111
...
C72EB*               Note: Fuse Checksum*
 9604
```

Pour chacun des 5828 fusibles de ce circuit, un '0' indique un fusible intact, un '1' un fusible programmé.

L'examen du début de la table précédente met en évidence un défaut majeur de cette technologie : la programmation détruit plus de fusibles qu'elle n'en conserve, et de loin. Cela se traduit par une mauvaise utilisation du silicium, un temps de programmation important (quelques secondes) et des contraintes thermiques sévères imposées au circuit lors de l'opération. Cette technologie n'est donc pas généralisable à des circuits dépassant quelques centaines de portes équivalentes.

---

provoque une surintensité dans ce fusible. Les caractéristiques détaillées des signaux à appliquer lors de la programmation sont consignées, pour chaque circuit et pour chaque fabricant, dans une base de données d'algorithmes du programmateur.

Le lecteur averti aura peut-être remarqué, à la lecture de l'en-tête du fichier JEDEC, qu'en réalité le circuit précédent ne contient aucun fusible. Il s'agit en vérité d'un circuit CMOS à grille flottante, mais l'ancienne terminologie est restée.

MOS à grille flottante

Les transistors MOS sont des interrupteurs[10], commandés par une charge électrique stockée sur leur électrode de grille. Si, en fonctionnement normal, cette grille est isolée, elle conserve sa charge éventuelle éternellement[11]. Il reste au fondeur à trouver un moyen de modifier cette charge, pour programmer l'état du transistor. Le dépôt d'une charge électrique sur la grille isolée d'un transistor fait appel à un phénomène connu sous le nom d'effet tunnel : un isolant très mince (une cinquantaine d'angströms, 1 Å = 10^{-10} m) soumis à une différence de potentiel suffisamment grande (une dizaine de volts, supérieure aux 3,3 ou 5 volts des alimentations classiques) est parcouru par un courant de faible valeur, qui permet de déposer une charge électrique sur une électrode normalement isolée. Ce phénomène, réversible, permet de programmer et d'effacer une mémoire.

La figure III-10 montre la structure du PLD élémentaire précédent, dans lequel les fusibles sont remplacés par des transistors à grille isolée (technologie FLASH).

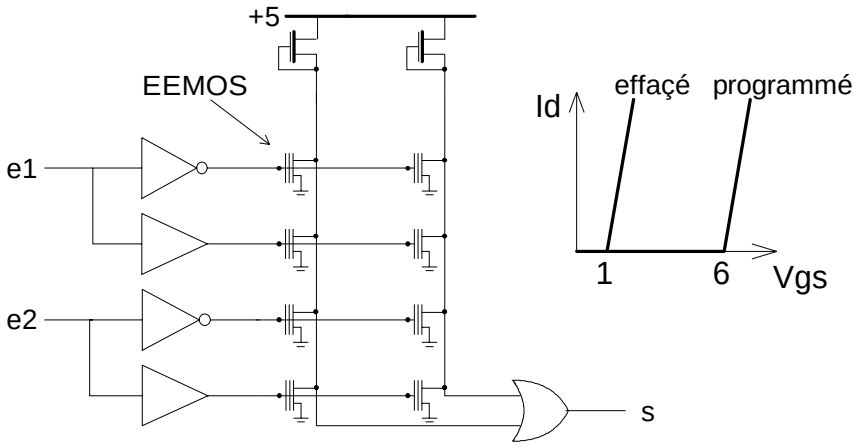


Figure 10 : Pld simple à MOS

Les transistors disposent de deux grilles, dont l'une est isolée. Une charge négative (des électrons) déposée sur cette dernière, modifie la tension de seuil du transistor commandé par la grille non isolée. Quand cette tension de seuil dépasse la tension d'alimentation, le transistor est toujours bloqué (interrupteur ouvert). Une variante (plus ancienne) de cette structure consiste à mettre deux transistor en série, l'un à grille isolée, l'autre normal. Le transistor à grille isolée est programmé pour être toujours conducteur ou toujours bloqué ; on retrouve exactement la fonction du fusible, la réversibilité en plus.

Le contrôle des dimensions géométriques des transistors permet actuellement d'obtenir des circuits fiables, programmables sous une dizaine de volts, reprogrammables à volonté (plusieurs centaines de fois), le tout électriquement.

Les puissances mises en jeu lors de la programmation sont suffisamment faibles pour que les surtensions nécessaires puissent être générées par les circuits eux-mêmes. Vu par l'utilisateur, le circuit devient alors programmable *in situ*, c'est à dire sans appareillage accessoire. Dans ces

[10]Quand on les utilise en tout ou rien, le régime source de courant contrôlée relève du monde des fonctions analogiques.
[11]L'éternité en question est garantie durer plus de 20 ans.

circuits, un automate auxiliaire gère les algorithmes de programmation et le dialogue avec le système de développement, via une liaison série.

Mémoires statiques

Dans les circuits précédents, la programmation de l'état des interrupteurs, conservée en l'absence de tension d'alimentation, fait appel à un mode de fonctionnement électrique particulier. Dans les technologies à mémoire statique, l'état de chaque interrupteur est commandé par une cellule mémoire classique à quatre transistors (plus un transistor de programmation), dont le schéma de principe est celui de la figure III-11.

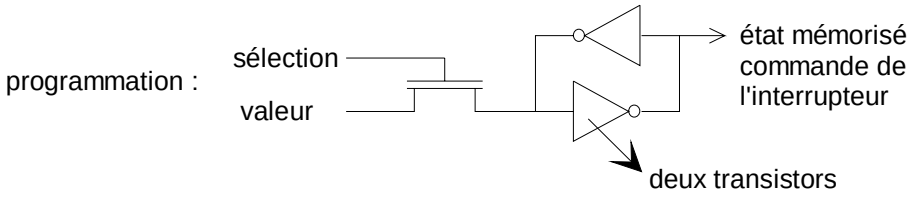


Figure 11 : Cellule SRAM

La modification de la configuration d'un circuit devient alors une opération logique quasi ordinaire, qui ne nécessite pas d'opération électrique spéciale. Ces circuits permettent des reconfigurations, partielles ou totales, en nombre illimité. Il est même envisageable de créer des fonctions dont certains paramètres sont modifiables en cours de fonctionnement, des filtres adaptatifs par exemple.

Le prix à payer pour cette souplesse est que les cellules SRAM doivent être rechargées à chaque mise sous tension et que chaque interrupteur occupe plusieurs transistors : l'interrupteur lui-même et les transistors de la cellule mémoire.

Antifusibles

L'inverse d'un fusible est un anti-fusible. Le principe est, à l'échelle microscopique, celui de la soudure électrique par points. Un point d'interconnexion est réalisé au croisement de deux pistes conductrices (métal ou semi-conducteur selon les procédés de fabrication), séparées par un isolant de faible épaisseur. Une surtension appliquée entre les deux pistes provoque un perçage définitif du diélectrique, ce qui établit la connexion.

Les points d'interconnexions ont un diamètre de l'ordre de la largeur d'une piste, c'est à dire de l'ordre du micron ; il est donc possible de prévoir un très grand nombre d'interconnexions programmables. La résistance du contact créé est très faible, de l'ordre d'une cinquantaine d'ohms (dix fois moins que celle d'un transistor MOS), d'où des retards liés aux interconnexions très faibles également.

Les circuits à anti-fusibles partagent, avec ceux à SRAM, le sommet de la gamme des circuits programmables en vitesse et en densité d'intégration.

Il est clair que ces circuits ne sont programmables qu'une fois.

2.3 Des architectures

Les différences de technologies se doublent de différences d'architectures. Nous tenterons ici de mettre en lumière des grands points de repère, sachant que toute classification a un côté un peu réducteur. La plupart des circuits complexes panachent les architectures.

Somme de produits

Toute fonction logique combinatoire peut être écrite comme somme de produits, nous avons évoqué ce point à propos des PLAs. La partie combinatoire d'un circuit programmable peut donc être construite en suivant cette démarche : chaque sortie est une fonction de toutes les entrées. Si la sortie se rapporte à un opérateur séquentiel, les équations programmables calculent la valeur de la commande d'une bascule en fonction des entrées et des états de toutes les bascules du circuit : nous retrouvons l'architecture matérielle d'une machine d'états générique. Les PLDs de première génération suivaient ce principe.

La capacité de calcul de cette architecture est limitée par le nombre maximum de produits réunis dans la somme logique, et, dans une moindre mesure, par le nombre de facteurs de chaque produit. Les valeurs typiques sont respectivement de 16 et 44 pour un 22V10.

Très efficace pour la réalisation d'opérateurs relativement simples, cette architecture n'est pas directement généralisable à des circuits complexes : pour augmenter la capacité potentielle de calcul du circuit, il faut augmenter les dimensions des produits et des sommes logiques. Or même dans une fonction complexe, de nombreux sous-ensembles sont simples ; ces sous-ensembles monopoliseront inutilement une grande partie des potentialités du circuit[12].

Cellules universelles interconnectées

L'autre approche, radicalement opposée, est de renoncer à la réduction en première forme normale des équations logiques. On divise le circuit en blocs logiques indépendants, interconnectés par des chemins de routage. Une fonction logique est récursivement décomposée en opérations plus simples, jusqu'à ce que les opérations élémentaires rentrent dans une cellule. La figure III-12 en fournit un exemple.

[12]L'architecture du 22V10 contourne cette difficulté en utilisant des sommes de dimensions différentes (de 8 à 16). Mais cette solution impose des contraintes sur l'affectation des broches aux sorties d'une fonction : les broches centrales sont plus « puissantes » que les broches situées aux extrémités d'un boitier DIL. Cela peut, par exemple, interdire la modification d'une fonction en conservant le câblage extérieur.

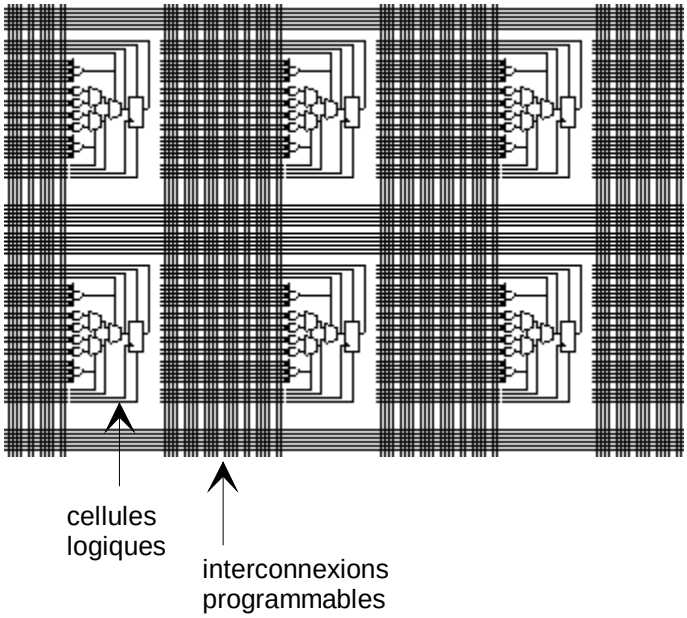


Figure 12 : Cellules logiques interconnectées

Les différences d'architectures entre les circuits concernent le compromis fait entre capacité de calcul de chaque cellule et possibilités d'interconnexions :

- Cellules de grande taille, à la limite l'équivalent d'un PLD classique, et interconnexions limitées. C'est schématiquement le choix fait pour les circuits CPLDs, en technologie FLASH.
- A l'autre extrême, cellules très petites (une bascule et un multiplexeur de commande), avec des ressources de routage importantes. C'est typiquement le choix fait dans les FPGAs à anti-fusibles, dont la figure III-12 est un exemple.
- La solution intermédiaire est, sans doute, la plus répandue : les cellules comportent une ou deux bascules et des blocs de calcul combinatoires qui traitent de 6 à 10 entrées. Ces cellules sont optimisées pour accroître l'efficacité de traitement d'opérations courantes, comme le comptage ou l'arithmétique. Les circuits FPGAs à SRAM sont généralement associés à de telles cellules de taille moyenne.

Cellules d'entrée-sortie

Dans les circuits programmables de première génération, les sorties étaient associées de façon rigide à des noeuds internes du circuit : résultat combinatoire, état d'une bascule.

Très vite est apparu l'intérêt de créer des macrocellules d'entrée-sortie pourvues d'une certaine capacité de reconfiguration. La figure III-13 reprend le schéma de principe des cellules d'un PLD 22V10.

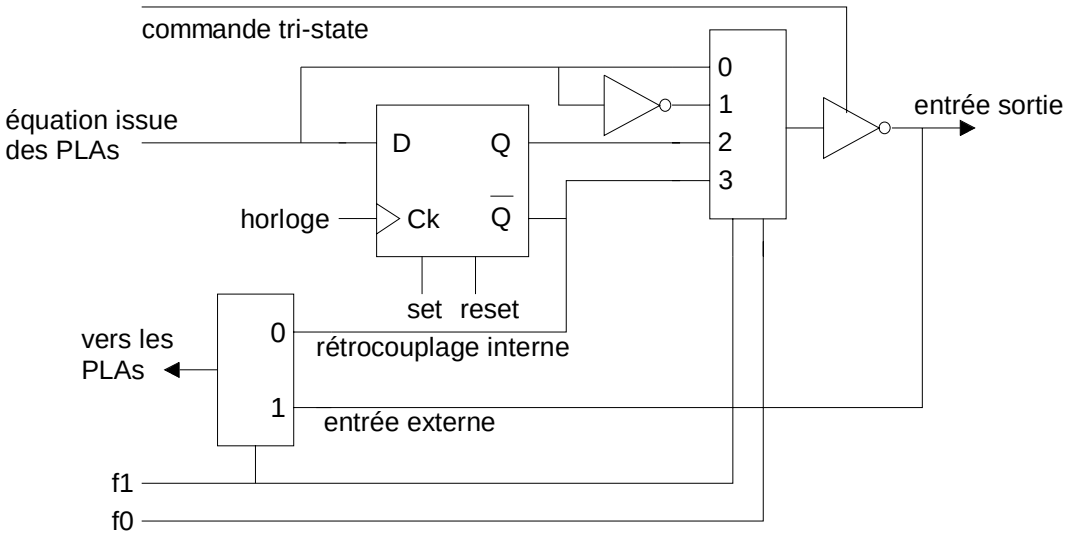


Figure 13 : Macrocellule configurable

Les deux fusibles f1 et f0 permettent de configurer la macrocellule en entrée-sortie combinatoire bidirectionnelle, complémentée ou non[13], ou en sortie registre trois états. Chaque sortie du circuit peut disposer de son propre mode, grâce aux vingt fusibles de configuration.

Les évolutions ultérieures rendent indépendantes les macrocellules et les broches du circuit, autorisant ainsi la création de bascules enterrées (*burried flip flops*) et d'entrées-sorties bidirectionnelles, quel que soit le mode, registre ou non, attaché à la sortie. Dans les architectures à cellules universelles interconnectées des FPGAs, les cellules d'entrée-sortie sont entièrement configurables et routables, au même titre que les cellules de calcul. Elles disposent de leurs propres bascules de synchronisation, en entrée et en sortie, indépendantes de celles des blocs logiques qui interviennent dans la fonction programmée.

Placement et routage

Le placement consiste à attacher des blocs de calcul aux opérateurs logiques d'une fonction et à choisir les broches d'entrée-sorties. Le routage consiste à créer les interconnexions nécessaires.

Pour les PLDs simples, le placement est relativement trivial et le routage inexistant. Les compilateurs génériques (i.e. indépendants du fondeur) effectuent très bien ces deux opérations.

Dès les CPLDs, et plus encore pour les FPGAs, ces deux opérations deviennent plus complexes et nécessitent un outil spécifique du fondeur, qui seul a les compétences nécessaires[14]. Le compilateur VHDL sert, dans ces cas, de frontal homogène qui traduit, après une première optimisation, la description VHDL dans un langage structurel adapté au logiciel spécifique[15]. Nous avons vu, à propos de la rétroannotation, que les outils des fondeurs fournissent en retour un modèle, VHDL ou VERILOG, du circuit généré qui prend en compte les temps de propagation internes.

[13]Complémenter une sortie permet, dans certains cas, de simplifier les équations logiques.
[14]Pour la simple raison qu'il est seul à connaître ses circuits dans leurs moindres détails.
[15]Une certaine portabilité demeure, même à ce niveau. Il existe des formats de fichiers communs à plusieurs fondeurs : les fichiers PLA ou, plus souvent, des fichiers dans un langage symbolique, EDIF pour *electronic data interchange format for net-lists*. Il s'agit d'un langage de description structurelle, qui ressemble un peu à LISP, compris par la majorité des systèmes de CAO.

2.4 Des techniques de programmation

Le placeur-routeur transforme la description structurelle du circuit en une table des fusibles consignée dans un fichier (JEDEC dans les cas simples, LOF, POF, etc. autrement). Pour la petite histoire, signalons que cette table peut contenir plusieurs centaines de milliers de bits, un par « fusible ».

Traditionnellement, la programmation du circuit, opération qui consiste à traduire la table des fusibles en une configuration matérielle, se faisait au moyen d'un programmateur, appareil capable de générer les séquences et les surtensions nécessaires. La tendance actuelle est de supprimer cette étape de manipulation intermédiaire, manipulation d'autant plus malaisée que l'augmentation de la complexité des boîtiers va de pair avec celle des circuits. Autant il était simple de concevoir des supports à force d'insertion nulle pour des boîtiers DIL (*dual in line*) de 20 à 40 broches espacées de 2,54 mm, autant il est difficile et coûteux de réaliser l'équivalent pour des PGA (*pin grid array*) et autres BGA (*ball grid array*), de 200 à plus de 300 broches réparties sur toute la surface du boîtier, sans parler des boîtiers miniaturisés, au pas de 0,65 mm, destinés au montage en surface.

Une difficulté du même ordre se rencontre pour le test : il est devenu quasi impossible d'accéder, par des moyens traditionnels tels que les pointes de contact d'une « planche à clous », aux équipotentielles d'une carte. De toute façon, les équipotentielles du circuit imprimé ne représentent plus qu'une faible proportion des noeuds du schéma global : un circuit de 250 broches peut contenir 2500 bascules.

Trois modes : fonctionnement normal, programmation et test

Fonctionnement normal, programmation et test : l'idée s'est imposée d'incorporer ces trois modes de fonctionnement dans les circuits eux-mêmes, comme partie intégrante de leur architecture. Pour le test de cartes, une norme existe : le standard IEEE 1149.1, plus connu sous le nom de *boundary scan* du consortium JTAG (*join test action group*). Face à la quasi impossibilité de tester de l'extérieur les cartes multicouches avec des composants montés en surface, un mode de test a été défini, pour les VLSI numériques. Ce mode de test fait appel à une machine d'états, intégrée dans tous les circuits compatibles JTAG, qui utilise cinq broches dédiées :

- `Tck`, une entrée d'horloge dédiée au test, différente de l'horloge du reste du circuit.
- `Tms`, une entrée de mode qui pilote l'automate de test.
- `Tdi`, une entrée série.
- `Tdo`, une sortie série.
- `Trst` (optionnelle), une entrée de réinitialisation asynchrone de l'automate.

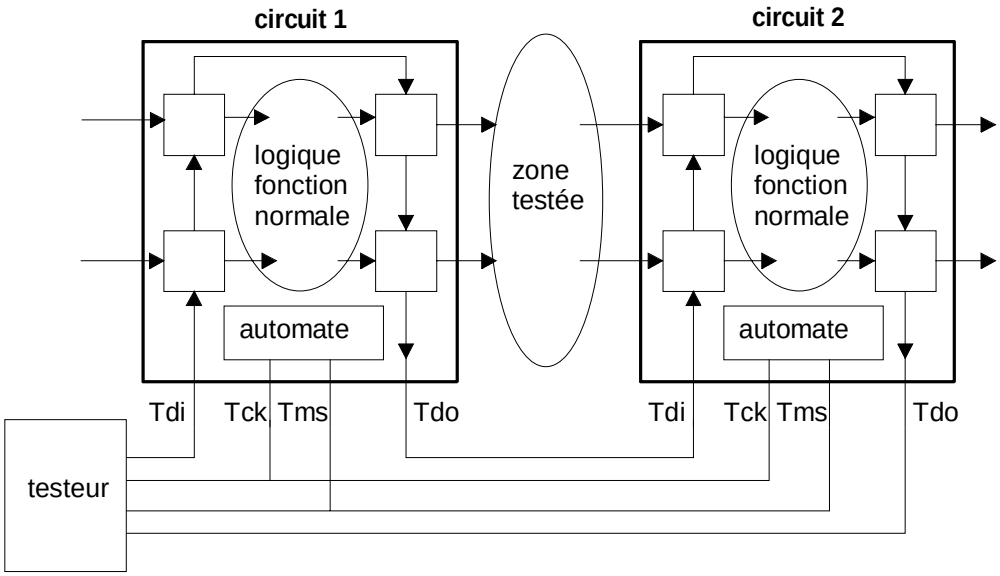


Figure 14 : Boudary scan

L'utilisation première de ce sous-ensemble de test est la vérification des connexions d'une carte. Quand le mode de test est activé, via des commandes ad-hoc sur les entrées `Tms` et `Trst`, le fonctionnement normal du circuit est inhibé. Les broches du circuit sont connectées à des cellules d'entrée-sortie dédiées au test[16], chaque cellule est capable de piloter une broche en sortie et de capturer les données d'entrée, conformément au schéma de principe de la figure III-14.

Toutes les cellules de test sont connectées en un registre à décalage, tant à l'intérieur d'un circuit qu'entre les circuits, constituant ainsi une chaîne de données, accessible en série, qui parcourt l'ensemble des broches de tous les circuits compatibles JTAG d'une carte. Les opérations de test sont programmées via des commandes passées aux automates et des données entrées en série. Les résultats des tests sont récupérables par la dernière sortie série.

Les automates de test permettent d'autres vérifications que celles des connexions : il est possible de les utiliser pour appliquer des vecteurs de test internes aux circuits, par exemple. C'est souvent de cette façon que sont effectués certains des tests à la fabrication. L'idée était séduisante d'utiliser la même structure pour configurer les circuits programmables. C'est ce qui est en train de se faire : la plupart des fabricants proposent, ou annoncent (en 1997), des solutions plus ou moins dérivées de JTAG pour éviter à l'utilisateur d'avoir recours à un appareillage extérieur[17].

Programmables in-situ

Les circuits programmables *in situ* se développent dans le monde des PLDs et CPLDs en technologie FLASH. Du simple 22V10, à des composants de plus de 10000 portes équivalentes et 400 bascules (LATTICE, par exemple), il est possible de programmer (et de modifier) l'ensemble d'une carte, sans démontage, à partir d'un port parallèle de PC.

Les technologies FLASH conservent leur configuration en l'absence d'alimentation.

[16]Typiquement, une broche d'entrée-sortie bidirectionnelle est pilotée par 6 bascules : un couple en entrée, un couple en sortie et un couple en commande de trois états. Les bascules par paires permettent de décaler les données tout en mémorisant la configuration précédente de chaque broche.

[17]Les pionniers en la matière furent les sociétés XILINX pour les technologies SRAM et LATTICE pour les technologies FLASH.

Reconfigurables dynamiquement

Les FPGAs à cellules SRAM offrent des possibilités multiples de chargement de la mémoire de configuration :

- Chargement automatique, à chaque mise sous tension, des données stockées dans une mémoire PROM. Les données peuvent être transmises en série, en utilisant peu de broches du circuit, ou en parallèle octet par octet, ce qui accélère la phase de configuration mais utilise, temporairement du moins, plus de broches du circuit. Plusieurs circuits d'une même carte peuvent être configurés en coopération, leurs automates de chargement assurent un passage en mode normal coordonné, ce qui est évidemment souhaitable.
- Chargement, en série ou en parallèle, à partir d'un processeur maître. Ce type de structure autorise la modification rapide des configurations en cours de fonctionnement. Cette possibilité est intéressante, par exemple, en traitement de signal.

3. PLDs, CPLDs, FPGAs : Quel circuit choisir ?

Dans le monde des circuits numériques les chiffres évoluent très vite, beaucoup plus vite que les concepts. Cette impression de mouvement permanent est accentuée par les effets d'annonce des fabricants et par l'usage systématique de la publicité comparative, très en vogue dans ce domaine.

Il semble que doivent se maintenir trois grandes familles :

- Les PLDs et CPLDs en technologie FLASH, utilisant une architecture somme de produits. La tendance est à la généralisation de la programmation in-situ, rendant inutiles les programmateurs sophistiqués. Réservés à des fonctions simples ou moyennement complexes, ces circuits sont rapides (jusqu'à environ 200 MHz) et leurs caractéristiques temporelles sont pratiquement indépendantes de la fonction réalisée. Les valeurs de fréquence maximum de fonctionnement de la notice sont directement applicables.
- Les FPGAs à SRAM, utilisant une architecture cellulaire. Proposés pratiquement par tous les fabricants, ils couvrent une gamme extrêmement large de produits, tant en densités qu'en vitesses. Reprogrammables indéfiniment, ils sont devenus reconfigurables rapidement (200 ns par cellule), en totalité ou partiellement.
- Les FPGAs à antifusibles, utilisant une architecture cellulaire à granularité fine. Ces circuits tendent à remplacer une bonne partie des ASICs prédiffusés. Programmables une fois, ils présentent l'avantage d'une très grande routabilité, d'où une bonne occupation de la surface du circuit. Leur configuration est absolument immuable et disponible sans aucun délai après la mise sous tension ; c'est un avantage parfois incontournable.

3.1 Critères de performances

Outre la technologie de programmation, capacité et vitesse sont les maîtres mots pour comparer deux circuits. Mais quelle capacité, et quelle vitesse ?

Puissance de calcul

Les premiers chiffres accessibles concernent les nombres d'opérateurs utilisables.

Nombre de portes équivalentes

Le nombre de portes est sans doute l'argument le plus utilisé dans les effets d'annonce. En 1997 la barrière des 100000 portes est largement franchie. Plus délicate est l'estimation du nombre de portes qui seront inutilisées dans une application, donc le nombre réellement utile de portes.

Nombre de cellules

Le nombre de cellules est un chiffre plus facilement interprétable : le constructeur du circuit a optimisé son architecture, pour rendre chaque cellule capable de traiter à peu près tout calcul dont la complexité est en relation avec le nombre de bascules qu'elle contient (une ou deux suivant les architectures). Trois repères chiffrés : un 22V10 contient 10 bascules, la famille des CPLDs va de 32 bascules à quelques centaines et celle des FPGAs s'étend d'une centaine à quelques milliers.

Dans les circuits à architectures cellulaires, il est souvent très rentable d'augmenter le nombre de bascules si cela permet d'alléger les blocs combinatoires (*pipe line*, codages *one hot*, etc.).

Nombre d'entrée-sorties

Le nombre de ports de communication entre l'intérieur et l'extérieur d'un circuit peut varier dans un rapport deux, pour la même architecture interne, en fonction du boîtier choisi. Les chiffres vont de quelques dizaines à quelques centaines de broches d'entrée-sorties..

Capacité mémoire

Les FPGAs à SRAM contiennent des mémoires pour stocker leur configuration. La plupart des familles récentes offrent à l'utilisateur la possibilité d'utiliser certaines de ces mémoires en tant que telles. Par exemple, la famille 4000 de XILINX permet d'utiliser les mémoires de configuration d'une cellule pour stocker 32 bits de données ; la cellule correspondante n'est évidemment plus disponible comme opérateur logique. Les capacités de mémorisation atteignent quelques dizaines de kilobits.

Routabilité

Placement et routage sont intimement liés, et le souhait évident de l'utilisateur est que ces opérations soient aussi automatiques que possible. Le critère premier de routabilité est l'indépendance entre la fonction et le brochage. Certains circuits (mais pas tous) garantissent une routabilité complète : toute fonction intégrable dans le circuit pourra être modifiée sans modification du câblage externe.

Le routage influe sur les performances dynamiques de la fonction finale. La politique généralement adoptée est de prévoir des interconnexions hiérarchisées : les cellules sont regroupées en grappes (d'une ou quelques dizaines) fortement interconnectées, des pistes de communication reliant les grappes entre elles. Les interconnexions locales n'ont que peu d'influence sur les temps de calcul, contrairement aux interconnexions distantes dont l'effet est notable.

A priori c'est au placeur-routeur que revient la gestion de ces interconnexions ; à condition que le programmeur ne lui complique pas inutilement la tâche. Un optimiseur a toujours du mal à découper des blocs de grandes tailles, il lui est beaucoup plus simple de placer des objets de petites dimensions. En VHDL cela s'appelle construction hiérarchique ; un ensemble complexe doit être conçu comme l'assemblage d'unités de conceptions aussi simples que possibles.

Vitesse de fonctionnement

Nous avons vu, à propos de la rétroannotation, que les comportements dynamiques des FPGAs et des PLDs simples présentent des différences marquantes. Les premiers ont un comportement

prévisible, indépendamment de la fonction programmée ; les limites des seconds dépendent de la fonction, du placement et du routage. Une difficulté de jeunesse des FPGAs a été la non reproductibilité des performances dynamiques en cas de modification, même mineure, du contenu d'un circuit. Les logiciels d'optimisation et les progrès des architectures internes ont pratiquement supprimé ce défaut ; mais il reste que seule une analyse et une simulation post synthèse, qui prend en compte les paramètres dynamiques des cellules, permet réellement de prévoir les limites de fonctionnement d'un circuit.

Modèle général de détermination de fmax

Le modèle général de détermination de la fréquence maximum d'un opérateur séquentiel prend en compte les retards dans les circuits et les règles concernant les instants de changement des entrées vis à vis des fronts actifs de l'horloge. La figure III-15 définit les temps les plus importants : t_{Pi} pour des temps de propagation et t_{SU} pour le temps de prépositionnement d'une bascule.

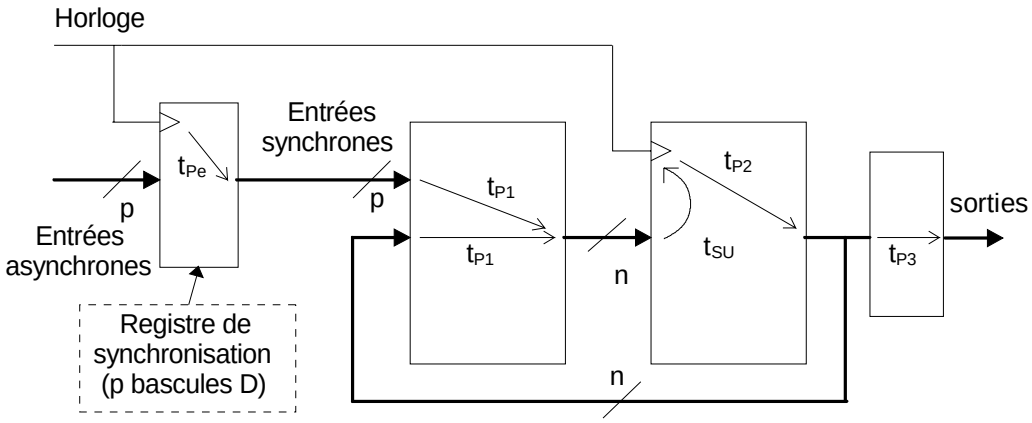


Figure 15 : Modèle de calcul de la fréquence maximum

La fréquence maximum de fonctionnement interne est donnée par :

$$Fint = 1/(t_{P2} + t_{P1} + t_{SU})$$

Pour le calcul de la fréquence maximum externe, il convient de rajouter, dans la formule précédente, le temps de propagation des cellules de sortie :

$$Fext = 1/(t_{P3} + t_{P2} + t_{P1} + t_{SU})$$

Dans un FPGA le routeur analyse le schéma généré et en déduit les différents temps de propagation, à partir d'un modèle des cellules élémentaires du circuit.

Dans le cas des PLDs simples et de beaucoup (pas tous) des CPLDs, les notices fournissent directement les valeurs des fréquences maximum et/ou des temps de retard et de prépositionnement entre les signaux appliqués aux broches du circuit et les fronts de l'horloge.

Style de programmation et performances

Pour l'auteur d'un programme VHDL, quelques guides de programmation sont utiles :

– Réfléchir au codage des états, dans la conception des machines d'états. Les sorties directes du registre d'état sont préférables ; les codes *one hot* sont très efficaces dans les FPGAs. Nous avons évoqué ces points précédemment.

- Subdiviser les blocs de calcul combinatoires en tranches séparées par des registres ; autrement dit, penser aux architectures *pipe line*. Ces architectures génèrent un retard global de plusieurs périodes d'horloge, mais permettent d'obtenir des flots de données rapides[18].
- Savoir que les bibliothèques des fondeurs sont riches en modules structurels optimisés en fonction du circuit cible : les modules LPM (*Library of Parameterized Modules*).
- Les synthétiseurs infèrent automatiquement des modules LPM, à partir de descriptions comportementales de haut niveau, sous réserve que le programmeur respecte certaines règles d'écriture ou indique explicitement qu'il souhaite utiliser les librairies correspondantes. La notice de tous les compilateurs explique la démarche à suivre.

Consommation

Les premiers circuits programmables avaient plutôt mauvaise réputation sur ce point. Tous les circuits actuels ont fait d'importants progrès en direction de consommations plus faibles.

Le compromis vitesse consommation

Règle générale des circuits numériques, encore plus vraie dans le monde des technologies MOS que dans celui des technologies bipolaires : pour aller vite il faut de la puissance. Les notices fournissent communément des courbes de consommation pour des éléments classiques, comme un compteur synchrone 16 bits, en fonction de la fréquence d'horloge. Le passage à 3,3 V des tensions d'alimentation permet une économie non négligeable de puissance, pour les mêmes valeurs de courant.

Les cellules de certains circuits sont programmables en deux modes : faible consommation ou vitesse maximum (bit *turbo*). Le gain de vitesse se paye par une consommation nettement plus élevée (pratiquement un facteur 2 pour un EPM7032 cadencé à 60Mhz, par exemple).

De façon générale, le courant moyen consommé par un circuit est de la forme :

$$I_{CC} = I_{CC0} + k \times n_L \times F + n_S \times C_S \times \Delta V \times F/2$$

Où I_{CC0} représente le courant statique consommé au repos, n_L le nombre moyen de cellules logiques qui commutent simultanément à chaque front d'horloge, n_S le nombre moyen de sorties qui commutent à chaque front d'horloge, C_S la capacité de charge moyenne des sorties, ΔV l'excursion de la tension de sortie, F la fréquence d'horloge et k un coefficient de consommation par cellule par hertz.

Quelques chiffres

Un ordre de grandeur du paramètre k précédent est, pour un FPGA de la famille FLEX 8000 d'ALTERA, de 150 µA par MHz et par cellule.

Un circuit cadencé à 50 MHz, dans lequel 100 cellules commutent, en moyenne, à chaque front d'horloge, consomme, sans charge extérieure, un courant moyen de l'ordre de 750 mA. Ce qui est loin d'être négligeable.

A titre de confrontation, la valeur précédente doit être comparée à la consommation de 50 compteurs binaires[19]. Un compteur binaire de la famille TTL-AS (il faut prendre des circuits de vitesses comparables) consomme 35 mA. Les chiffres parlent d'eux-mêmes.

---

[18]L'image classique est le principe de la fabrication des voitures à la chaîne : même si une voiture sort toutes les dix minutes, il faut plus de dix minutes pour fabriquer une voiture prise isolément. Le débit est très supérieur à l'inverse du temps de fabrication d'une seule voiture.

[19]Dans un compteur binaire deux cellules commutent, en moyenne, à chaque période d'horloge. Nous laissons au lecteur le soin de le démontrer.

Toujours dans le même ordre, un PLD 22V10 rapide, 10 cellules, consomme un courant de l'ordre de 100 mA. Dans ce dernier cas, la fonction programmée et la fréquence d'horloge n'ont qu'une incidence faible sur le courant consommé.

L'organisation PREP

Nombre de portes, de cellules, de bascules, fréquence maximum, dans quelle condition ? Avec quel logiciel ? Les comparaisons ne sont pas simples.

Le consortium PREP (*programmable electronics performance corporation*) regroupait jusqu'en 1996 la plupart des fabricants de circuits programmables. Cet organisme a défini un ensemble de neuf applications, typiques de l'usage courant des circuits programmables, qui servent de test à la fois pour les circuits et le système de développement associé. Les fruits de la confrontation à ce *benchmark* sont fournis pour la plupart des CPLDs et FPGAs. Ces résultats contiennent des informations de vitesse, fréquences maximums interne et externe pour chaque test, et de capacité, nombre moyen d'exemplaire de chaque test que l'on peut instancier dans un circuit.

Des applications « types »

Les 9 épreuves de test sont :

- *Datapath* : un chemin de données, sur un octet, franchit dans l'ordre un multiplexeur 4 vers 1, un registre tampon et un registre à décalage arithmétique (avec extension de signe). Le schéma ne comporte pratiquement pas de calcul entre les bascules des registres ; les fréquences maximum obtenues sont à peu de choses près celles des circuits en boucle ouverte. Le nombre de vecteurs d'entrée sollicite beaucoup les ressources de routage.
- *Counter timer* : Un compteur 8 bits à chargement parallèle parcourt un cycle défini par une valeur de chargement et une valeur finale. Un comparateur provoque le rechargement du compteur quand il a atteint la valeur finale. Le schéma comporte, outre le compteur, deux registres, un comparateur et un multiplexeur, le tout sur un octet. Le fonctionnement place le comparateur dans la boucle de commande du compteur, limitant par là sa fréquence maximum de fonctionnement.
- *Small state machine* : petite machine d'états, 8 états, 8 entrées, 8 sorties. Beaucoup de CPLDs arrivent à la faire fonctionner à leur fréquence maximum, les résultats sont plus variables pour les FPGAs.
- *Large state machine* : machine à 16 états, 8 entrées et 8 sorties. La plupart des CPLDs doivent abandonner leur fréquence de fonctionnement maximum : le nombre de variables est trop grand pour autoriser les calculs en une seule passe dans la logique combinatoire.
- *Arithmetic* : un multiplieur de deux nombres de 4 bits, résultat sur 8, suivi par un additionneur accumulateur sur 8 bits. La structure en « somme de produits » des CPLDs les rend très inefficaces dans les problèmes d'arithmétique. Les FPGAs montrent une supériorité architecturale nette face à ces problèmes.
- *Accumulateur* : accumulateur-additionneur sur 16 bits. Un additionneur de deux nombres de 16bits est suivi par un registre dont le contenu est pris comme l'un des opérandes de l'addition. Ce test génère un schéma moins complexe que le précédent.
- *16-bit counter* : un classique compteur 16 bits, à chargement parallèle synchrone et remise à zéro asynchrone. C'est un peu un test de vitesse pure dans une application standard.
- *16-bit prescaled counter* : compteur 16 bits à prédiviseur synchrone. Pour accélérer le comptage, une technique consiste à traiter à part l'étage de poids faible d'un compteur, quitte à perdre la possibilité d'effectuer le chargement parallèle en un seul cycle. Pour les CPLDs il n'y a aucune différence avec l'épreuve précédente ; pour les FPGAs l'architecture en petites cellules conduit à une accélération nette du fonctionnement.

– *Décodeur d'adresses* : un décodeur d'adresse génère 8 signaux de décodage mémorisés dans un registre à partir d'un signal d'entrée sur 16 bits ; il découpe ainsi l'espace d'adresses en huit pages. Le traitement de ce schéma favorise les circuits qui disposent de portes ET à grand nombre d'entrées. L'architecture CPLD se prête mieux à cette épreuve que celle des FPGAs.

À chacun son interprétation

Les résultats à ces épreuves sont généralement présentés sous forme de tableaux comparatifs. Chaque constructeur veillant, évidemment, à citer les résultats de la concurrence qui illustrent sa propre supériorité.

Une analyse générale, proposée par beaucoup de fabricants, consiste à calculer pour chaque circuit les moyennes des fréquences maximums de fonctionnement, et des nombres d'instances de chaque épreuve implantable dans le circuit. Ces deux chiffres fournissent une information globale de vitesse et une information globale de capacité. Sans entrer dans des comparaisons chiffrées qui ne valent qu'à un instant donné, il est intéressant de délimiter dans le plan fréquence/capacité les zones de prédilection des différentes catégories de circuits programmables. C'est le sens de la figure III-16.

Les petits circuits sont incapables de contenir les épreuves PREP, ils se concentrent à une capacité moyenne nulle. Nous les avons malgré tout placé dans ce plan, bien que la comparaison entre des moyennes d'un côté et une information ponctuelle de l'autre soit un peu trompeuse[20].

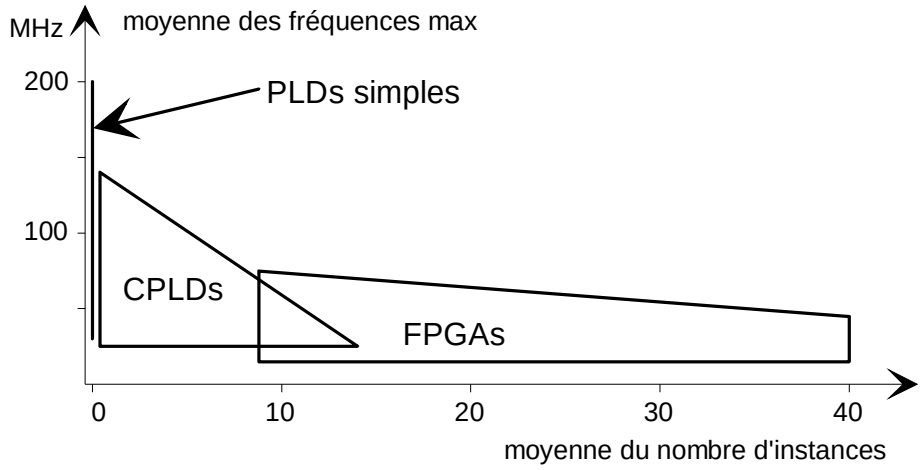


Figure 16 : Moyennes des tests PREP

3.2 Le rôle du « fitter »

Nous avons un peu exploré les aspects matériels des circuits programmables. Terminons en retrouvant, à travers quelques remarques, l'aspect logiciel des choses. Un circuit n'est rien sans son logiciel de développement ; un résultat surprenant aux épreuves PREP nous en fournit un exemple.

[20]Il n'est pas difficile de trouver dans un *data book* un exemple particulier de montage avec lequel un FPGA dépasse très largement les 200 MHz.

« Mise à plat » ou logique multi-couches ?

L'une de ces épreuves concerne l'arithmétique : pour cette épreuve un CPLD qui n'est ni moins performant que son concurrent direct, ni plus petit, s'est vu attribuer, en 1995, la note zéro en fréquence maximum de fonctionnement. Un zéro dans une moyenne, cela pèse lourd. La règle est la synthèse automatique, optimisation comprise. Le *fitter* du fondeur concerné s'est vraissemblablement fourvoyé dans la mise à plat, sous forme somme de produits logiques, des opérateurs arithmétiques. Nous avons eu l'occasion d'évoquer le caractère explosif de cette démarche.

La bonne approche était, pour cet exemple, de conserver plus de couches logiques, au détriment de la vitesse.

Les performances des FPGAs ne se dégradent que très progressivement quand la complexité d'un schéma augmente ; cette faculté est liée à leur architecture à granularité fine, qui impose de toute façon de passer à des structures multi-couches, même pour des fonctions combinatoires de complexité moyenne. Les constructeurs de circuits ont optimisé les passages de retenues d'une cellule à l'autre, qui autorisent des structures de propagation des retenues sans trop ralentir le système.

Certains logiciels donnent à l'utilisateur le loisir de régler manuellement le seuil de dédoublement d'équations logiques trop larges. Il est possible de spécifier le nombre maximum de facteurs dans un produit et le nombre maximum de termes dans une somme, par exemple. Ce genre de réglages manuels peut, bien qu'un peu délicat à manipuler, donner de bons résultats quand les choix automatiques ne conviennent plus.

Surface ou vitesse

Les options de réglage standard d'un *fitter* permettent de privilégier la surface (de silicium) ou la vitesse. Les deux choix sont, en effet, souvent contradictoires : pour diminuer la surface il faut augmenter le nombre de couches, ce qui pénalise la vitesse[21].

Librairies de macro fonctions

Rappelons l'importance des librairies de modules optimisés en fonction du circuit cible. L'idéal est qu'elles soient explorées automatiquement par l'analyseur de code VHDL, mais cette recherche automatique suppose que le code source ne brouille pas les cartes.

Le meilleur des compilateurs ne peut donner que ce que le circuit possède

Ultime remarque : le meilleur des compilateurs ne peut qu'organiser ce qui préexiste dans un circuit, il ne crée rien. Trivialement, tenter d'implanter un compteur 16 bits dans un 22V10 est un objectif inaccessible. Cet exemple en fera sourire plus d'un ; Transposé à un circuit de compression de la parole, à implanter dans un circuit plus conséquent, le problème reste le même ; même si l'analyse de faisabilité est plus ardue, elle doit pourtant être poursuivie.

[21]La situation réelle est un peu plus complexe : quand on diminue le nombre de couches logiques, la sortance imposée aux opérateurs augmente. Dans les technologies MOS les temps de propagation de ces opérateurs dépendent beaucoup de leurs capacités de charge, donc du nombre d'entrées qu'ils doivent commander. Les *fitter* contrôlent également ce type de contraintes.

Index