Chapitre 2

De la configuration des circuits électroniques

Seul l'éphémère dure.

Eugène IONESCO

LES CIRCUITS électroniques envahissent le quotidien de manière impressionnante. Ils sont présents dans les machines à laver, les montres, les voitures, et évidemment dans les ordinateurs. Qui aurait pensé, le 23 décembre 1947, date de la création du premier transistor par William Shockley, Walter Braittain et John Bardeen, physiciens des laboratoires Bell, qu'en 55 ans l'électronique allait révolutionner le style de vie de l'humanité ?

En 1952, G. W. A. Dummer, un Anglais expert en radar, publia un article proposant d'utiliser un bloc de matériel solide pour connecter des composants électroniques, sans fils de connexions. Le concept de circuit intégré était né, et six ans plus tard, en 1958, Jack Kilby, travaillant pour Texas Instruments, en réalisa le premier spécimen, sous la forme d'un oscillateur à décalage de phase contenant cinq éléments sur une seule pièce de semi-conducteur. Depuis lors, la complexité des circuits intégrés n'a cessé de croître, pour arriver à des systèmes jusqu'à 1.7 milliard de transistors pour le dernier processeur d'Intel.

La fabrication d'un circuit intégré spécifique à une application (ASIC) est une tâche ardue et coûteuse en temps et en argent. En effet, la réalisation d'un prototype nécessite la création d'un ou plusieurs masques [1], qui sont très coûteux s'il ne sont destinés qu'à une petite quantité. De plus, une simple erreur dans le design du système implique la création d'un ou plusieurs nouveaux masques. Et finalement, l'investissement en temps de développement est important, et n'est pas forcément tolérable si un produit doit très rapidement être mis sur le marché.

Parallèlement à l'avancée des ASICs, divers circuits programmables ont fait leur apparition, afin de réduire le temps et le coût de développement des circuits électroniques, tout en restant relativement compétitifs sur le plan de la rapidité d'exécution.

[1] Un masque sert à apposer des couches de métal sur le silicium, dans l'optique de relier entre eux les divers éléments tels que les transistors (ce type de technique est également exploitée pour le dopage du silicium).

La question est alors de pouvoir concevoir des systèmes contenant des circuits intégrés de la manière la moins coûteuse en terme de temps et d'argent. En fonction de différents paramètres tels que la complexité du système, le nombre de pièces à fabriquer ou le temps imparti, un type d'implémentation est choisi parmi la gamme des possibilités.

Concernant les systèmes bio-inspirés, qui seront introduits dans le chapitre suivant, un nombre non négligeable d'applications ont mis en jeu des circuits programmables : à titre d'exemples, le large parallélisme des réseaux de neurones s'applique parfaitement à une implémentation matérielle [56], et certains chercheurs travaillent à un nouveau type de systèmes informatiques tolérants aux pannes [142] dont les prototypes nécessitent la possibilité de se reprogrammer.

Dans ce chapitre, nous allons évoquer les différents types de circuits intégrés non-programmables, programmables et reprogrammables, allant de la simple mémoire programmable au circuit FPGA (Field Programmable Gate Array) [33] de plus de 10 millions de portes logiques reprogrammables, en passant par les circuits logiques programmables simples et les circuits spécifiques à une application. Nous allons tout d'abord présenter brièvement les processeurs puis les différents types de circuits ASICs. Ensuite nous présenterons les technologies de programmation matérielle existantes et l'évolution des circuits programmables PLD (Programmable Logic Device), en suivant un ordre chronologique, jusqu'à arriver aux FPGAs, qui feront l'objet d'une attention toute particulière. L'optique de cette thèse étant de proposer une nouvelle FPGA possédant des possibilités de routage et d'auto-configuration non présentes dans les circuits commerciaux, la description de tous les circuits se fera en mettant un accent particulier sur leurs possibilités et limitations en terme de routage.

2.1 Processeur

Un processeur [185] est un circuit intégré permettant d'implémenter n'importe quelle fonction en exécutant de manière séquentielle un code compilé. Le grand avantage du processeur est sa généricité, puisque n'importe quel programme peut y être exécuté. De plus, il est très facile de le programmer, c'est-à-dire de transcrire dans le langage du processeur un problème particulier.

Malheureusement, cette généricité s'accompagne d'un coût, à savoir le temps d'exécution d'une tâche. En effet, les instructions sont exécutées une à une (bien que certains processeurs présentent un certain parallélisme), l'opération $a + 2 + b + d$ nécessitant au minimum trois pas de temps. Un système matériel peut, lui, effectuer cette opération en un seul pas. De plus, le processeur étant une unité chargée d'un calcul, une seule erreur compromet l'ensemble de son exécution. Les systèmes parallèles laissent quant à eux la porte ouverte à l'autoréparation, une unité pouvant potentiellement prendre la place d'une autre se trouvant dans un état défectueux.

Les applications intrinsèquement parallèles tels que les réseaux de neurones ou les automates cellulaires sont donc exécutées de manière séquentielle par un processeur, alors qu'un système matériel a la possibilité d'exploiter tout le parallélisme possible. Les systèmes cellulaires composés de plusieurs cellules effectuant des tâches en parallèle sont donc bien plus efficacement réalisés en matériel qu'en logiciel. C'est pourquoi le reste de cette thèse traite de matériel, en enchaînant directement sur les ASICs, une des solutions d'implémentation de tels systèmes.

2.2 ASIC

Le Circuit Intégré Spécifique à une Application (ASIC) [219] est une des manières de réaliser un système de calcul matériel. Pour chaque application, un circuit différent est créé, soit en le construisant entièrement, soit en configurant une grille d'éléments préconstruits. L'avantage des ASICs sur les circuits reconfigurables, pour une même technologie, est évidemment leur rapidité, puisque les connexions sont créées physiquement plutôt que programmées. Toutefois, leur force est également leur faiblesse, puisque leur conception nécessite plus de temps, étant donné qu'en tout cas une couche de métal doit être apposée. De plus, ils ne sont en aucun cas programmables, et une erreur dans le design implique un changement complet du circuit.

Quatre grandes classes d'ASICs sont présentées ici, le graphique 2.1 les exposant avec un niveau de complexité croissant de gauche à droite. Nous allons les détailler dans l'ordre chronologique de leur apparition.

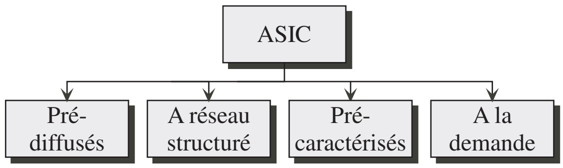


Figure 2.1 : *Les différents types de circuits ASIC.*

2.2.1 A la demande

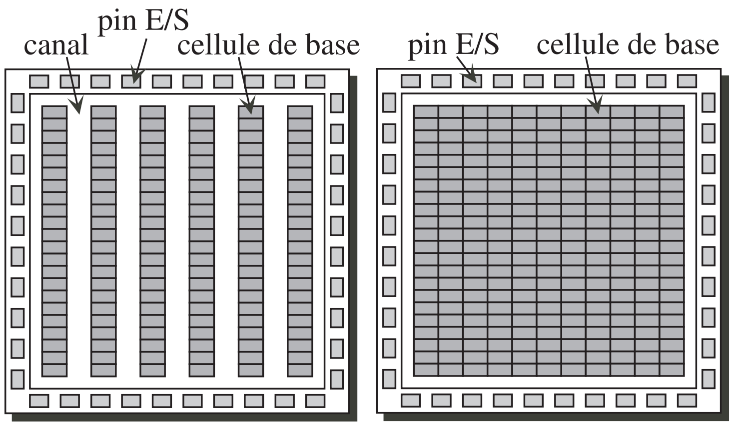
Pour la création d'un ASIC à la demande, communément appelé Full-Custom [48], les ingénieurs contrôlent chaque élément du circuit fabriqué. Ils dessinent les transistors, les placent et les connectent, sans qu'aucune contrainte prédéfinie n'existe concernant le placement des éléments. La taille de chaque transistor peut être changée afin d'optimiser le fonctionnement du système. L'avantage de cette approche est l'efficacité du circuit final en terme de quantité de silicium et de vitesse. En effet, il est possible de placer les éléments de manière optimale, sans aucune perte de place, et de le faire en tenant compte des contraintes de temps d'exécution. Par contre, si une erreur s'est subrepticement glissée dans la conception, le dessin du circuit entier doit être revu, dans un long et coûteux processus de correction. Ce type d'ASIC n'est donc pas du tout adapté au prototypage, étant donné le temps et l'investissement nécessaire à chaque prototype.

Les possibilités de routage dans un tel circuit sont quasiment infinies, étant donné que le développeur a total contrôle sur toute la conception du circuit. La seule limitation intervient par le nombre de couches de métal apposées sur le silicium. Toutefois, ce routage est fixé une fois pour toutes, ne laissant aucune possibilité de modification ultérieure.

2.2.2 Prédiffusés

Dans le milieu des années 60, Fairchild Semiconductor créa le précurseur des tableaux de portes, sous la forme d'un circuit appelé Micromatrix, composé d'une centaine de transistors. Les ingénieurs devaient dessiner à la main les interconnexions sur des feuilles de plastique, pour réaliser un circuit spécifique. Ensuite, en 1967, un nouveau circuit, appelé Micromosaic, et contenant quelques centaines de transistors, fut proposé par la même société, avec cette fois-ci un programme capable de générer automatiquement les interconnexions en fonction des spécifications des ingénieurs. Ce sont toutefois les compagnies telles qu'IBM et Fujitsu qui développèrent le concept d'ASIC prédiffusé (ou à tableau de portes) [2].

Dans le cas des mers de portes [99], les vendeurs fabriquent un circuit composé d'un tableau de cellules de base tels que des transistors. Une deuxième variante, le tableau de portes avec canaux de routage, propose plusieurs rangées de ces cellules séparées par des canaux de routage pouvant être connectés à ces éléments (Figure 2.2). La connectique interne à une rangée permet de créer des modules simples du type multiplexeurs ou bascules, et les canaux de routage offrent la possibilité de connecter ces modules entre eux. Le fabricant fournit généralement une librairie de modules, que le développeur peut utiliser, et qui sont ensuite placés sur le tableau en connectant les éléments de base grâce a des couches de métal supplémentaires.



(a) Un tableau de portes à canal (b) Une mer de portes

Figure 2.2 : *Deux types de circuits prédiffusés.*

L'un des avantages des circuits prédiffusés est que le vendeur peut fabriquer une grande quantité de pré-circuits identiques, et qu'il n'est nécessaire, pour réaliser un circuit final, que d'ajouter des couches de métal, sans se soucier de la préparation du silicium. En contrepartie, il y a souvent un nombre non négligeable d'éléments de base inutilisés, le placement des transistors étant contraint dans le sens où ils sont déjà incrustés dans le silicium, et le routage n'est donc pas optimal. Ces différents facteurs font que le tableau de portes est nettement moins efficace qu'une implémentation full-custom en terme de consommation et performance. Il offre cependant un temps de réalisation nettement plus faible, et également un moindre coût.

[2]En anglais *gate array*, et *sea of gates* pour mer de portes. Connu également sous l'appelation MPGA pour Metal Programmable Gate Array ou Mask Programmable Gate Array.

2.2.3 Pré-caractérisés

Les ASIC pré-caractérisés (en anglais Standard Cell)[220] ont vu le jour au début des années 80, dans le but de combler les lacunes des circuits prédiffusés. Contrairement à ces derniers, le silicium n'est pas prédiffusé avec des éléments de base. Le vendeur propose ici une librairie de cellules de base, ainsi que des modules plus complexes tels que des RAMs, des microprocesseurs, et bien d'autres. Le développeur utilise alors ces divers éléments pour créer une liste de modules et de leurs interconnexions décrivant son système, et le vendeur se charge ensuite de créer le circuit intégré.

L'avantage de cette approche réside dans l'optimisation des modules fournis par le fabriquant. Le placement de ces modules est réalisé de manière à également optimiser les performances, tout en minimisant la place nécessaire sur le silicium. Dès lors, la solution pré-caractérisée est très proche de l'optimal obtenu grâce à l'approche full-custom, et nécessite un temps de développement nettement moindre.

2.2.4 A réseau structuré

Comme nous le verrons à la section suivante, les circuits reconfigurables ont l'avantage d'être d'excellents candidats pour les applications industrielles de petit volume ainsi que pour le prototypage, un design pouvant être immédiatement testé. Toutefois, ils ne peuvent, en terme de vitesse, rivaliser avec un circuit ASIC spécialement conçu pour une application. C'est pourquoi en 2003 le dernier-né des circuits programmable a fait son apparition [3] : l'ASIC à réseau structuré [257].

Un ASIC à réseau structuré consiste en un circuit composé de différents éléments semblables à ceux que l'on peut trouver dans un FPGA standard, tels que look-up tables [4], inverseurs, flip-flops, et bien d'autres. Toutefois, à l'inverse du FPGA, ces différents blocs ne sont pas connectés entre eux. Par analogie avec les mers de portes, on pourrait les appeler mers de macros, des modules de complexité supérieure étant prédiffusés. Le circuit de base créé, il reste donc des couches de métal disponibles qui permettent de réaliser la connectique nécessaire à la réalisation d'un design particulier.

Certains fabricants proposent des solutions mettant en jeu des look-up tables et différentes portes logiques, ainsi que des blocs de mémoire, des DLLs (Delay-Locked Loops) et des PLLs (Phase-Locked Loops). Les outils de conception, responsables de la synthèse et du placement-routage, sont petit à petit adaptés à cette nouvelle technologie. Altera, un des leaders du marché du FPGA, offre quant à lui le concept de HardCopy, un circuit contenant exactement les mêmes éléments de base que les FPGAs qu'il propose sans toutefois qu'ils soient interconnectés. A l'heure actuelle, les circuits Stratix, APEX 20Ke et APEX 20KC supportent cette solution. Un utilisateur ayant réalisé et testé un système sur un de ces FPGAs peut alors le transférer sur un ASIC structuré correspondant, en étant sûr de garder la même fonctionnalité tout en augmentant potentiellement la fréquence de fonctionnement.

A ce jour peu de tests ont été effectués dans le but de comparer cette méthodologie avec d'autres, mais les résultats préliminaires tendent à prouver que les ASICs à réseau structuré occupent trois fois plus de place que les ASICs pré-caractérisés, et consomment deux à trois fois plus [146].

[3] Notons toutefois que le concept est apparu au début des années 90, sans connaître grand succès.

[4] Une look-up table, ou table de correspondance en français, fait correspondre une sortie à chaque combinaison d'entrées. Dans nos systèmes digitaux, nous parlerons de k-LUT pour une look-up table à k entrées. Dans ce cas, 2^k bits définissent la sortie pour chacune des combinaisons possibles des entrées.

2.2.5 Limitations

Les quatre types d'ASIC présentés sont tous des circuits fixes, qui une fois réalisés n'offrent aucune souplesse, leur fonctionnalité étant définitivement figée. L'avantage de cette approche est la rapidité du circuit final, ainsi que la place nécessaire à l'implémentation d'un design particulier. Toutefois, la non-reprogrammabilité de tels circuits est un important obstacle au prototypage, une réalisation physique pouvant demander jusqu'à 4 mois. De plus, la fixité de tels circuits ne permet pas la conception de systèmes adaptables, capables d'évoluer en fonction d'une tâche à résoudre dans un environnement qui peut être changeant. C'est pourquoi nous allons maintenant présenter les circuits programmables, qui se proposent de pallier au manque de flexibilité des ASICs. Nous allons tout d'abord présenter les différentes technologies de programmation avant d'aborder les circuits programmables dans l'ordre chronologique de leur apparition sur le marché, correspondant également à l'ordre de complexité.

2.3 Technologies de programmation

N'ayant pas peur des truismes, commençons par constater qu'un circuit programmable doit pouvoir être programmé, de même qu'un circuit reprogrammable doit pouvoir être reprogrammé. Nous allons donc présenter ici différentes technologies servant à la réalisation de circuits programmables ou reprogrammables. Nous commencerons par les masques, les fusibles et les antifusibles, non reprogrammables, puis nous enchaînerons sur les technologies reprogrammables, à savoir l'EPROM, l'EEPROM, la FLASH et la SRAM.

2.3.1 Masque

Comme nous l'avons vu précédemment, certains circuits ASIC (prédiffusés et à réseau structuré) peuvent être programmés grâce à un masque. Ce procédé, qui est également utilisé pour la création de mémoires à lecture seule (ROM), consiste à fabriquer un circuit intégré en y plaçant les composants et quelques ou aucune couche de métal. Ensuite, l'ajout d'une ou plusieurs couches de métal supplémentaires suffit à définir exactement la fonctionnalité du circuit. Un ou des masques sont donc générés pour chaque circuit différent, et servent à imposer les dernières couches de métal.

Dans le cas d'une mémoire ROM, qui n'est accessible qu'en lecture, après sa réalisation, toutes ses connexions sont figées. Il est possible de créer une ROM spécifique à partir de rien, ou de la préconstruire et d'utiliser la technique de masquage. Dans ce deuxième cas, une cellule mémoire est semblable à celle de la figure 2.3, où l'ensemble de la mémoire est répartie en ligne/colonne, la ligne servant d'adresse, et la colonne de donnée, une seule ligne pouvant être active à un instant donné. La programmation se fait par l'ajout d'une couche de métal, connectant certains transistors à leur colonne, une résistance en pull-up forçant la valeur à '1' si la connexion n'est pas établie, ou si le transistor est ouvert. Dans le cas de l'activation d'une ligne, chaque cellule de la ligne active fournit à sa colonne la valeur mémorisée, qui dans notre exemple, est un '1' si la connexion n'est pas programmée, et '0' sinon.

Notons que la réalisation d'un circuit par masquage offre un avantage en terme de délais, ces circuits étant les plus rapides des circuits programmables. Cependant

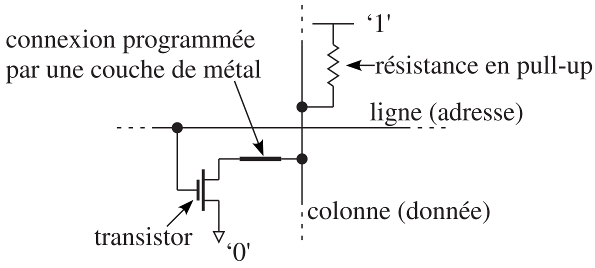


Figure 2.3 : *Une cellule de mémoire ROM programmée par masque, basée sur un transistor.*

le fait de devoir appliquer des couches de métal est un processus long et coûteux en comparaison des autres techniques.

2.3.2 Fusible

Basé sur le même principe que les fusibles domestiques, qui, pour éviter la destruction de nos appareils électroménagers, sont détruits par un trop fort courant, la technologie basée sur des fusibles fut une des premières à être utilisée. Le principe est simplement d'avoir des fusibles sur certains fils, et d'en faire brûler certains en leur appliquant un courant trop important. Sur chacun de ces fils, une résistance en pull-up ou pull-down force la valeur à 1 ou à 0 si le fusible a été détruit, comme illustré à la figure 2.4.

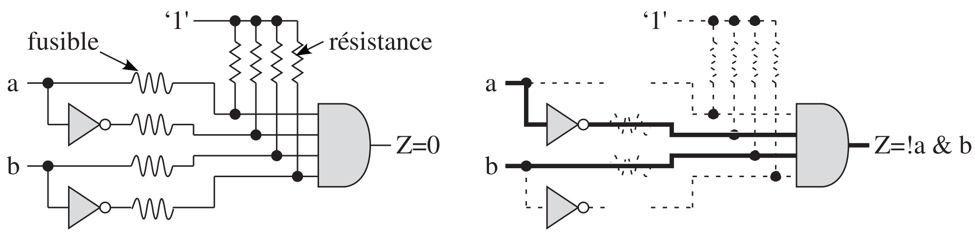


Figure 2.4 : *Un circuit contenant 4 fusibles non programmés, puis le circuit résultant après avoir brûlé le premier et le quatrième fusible.*

2.3.3 Antifusible

A ce que l'on peut considérer comme l'opposé des fusibles, se trouvent les antifusibles. De la même manière, sur certains fils du circuit se trouvent des antifusibles, mais contrairement aux fusibles, lorsqu'ils ne sont pas programmés, ils agissent comme une résistance infinie, comme si le fil était coupé. En appliquant un fort courant de grand voltage, l'antifusible est programmé, et laisse dès lors passer le courant. Une résistance est également placée après chaque antifusible, de manière à forcer la ligne à 1 ou 0 dans le cas où il n'est pas programmé (Figure 2.5).

Notons que les circuits à base de fusibles et d'antifusibles ne peuvent être programmées qu'une seule fois. Chaque nouvelle implémentation, en cas d'erreur de design par exemple, implique dès lors la programmation d'un nouveau composant et l'élimination de l'ancien. Cependant, cette programmation étant électrique, un simple appareil

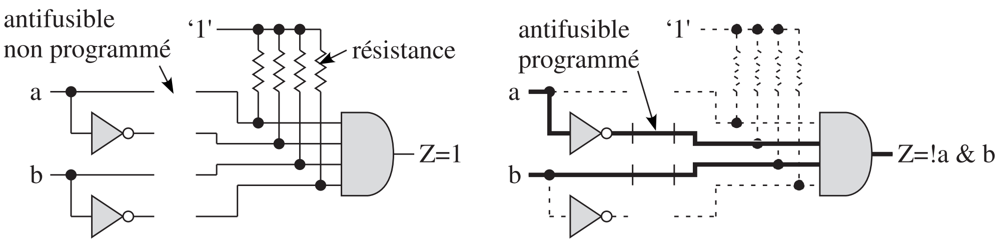


Figure 2.5 : *Un circuit contenant 4 antifusibles non programmés, puis le circuit résultant d'une programmation.*

est nécessaire, rendant le processus nettement plus simple et rapide que le masquage.

2.3.4 EPROM

Avant de décrire la technologie EPROM, passons quelques instants sur le concept de mémoire PROM. Le fonctionnement d'une mémoire PROM est identique que celui d'une ROM, si ce n'est le mode programmation. Au lieu de devoir physiquement créer des connexions grâce à une couche de métal, un fusible peut être brûlé ou non, ce qui est nettement plus rapide et moins coûteux (Figure 2.6). En effet, lors de la réalisation d'un système informatique, il n'est pas rare que des erreurs se glissent dans un design, et la destruction d'une PROM à chaque erreur n'est pas la panacée.

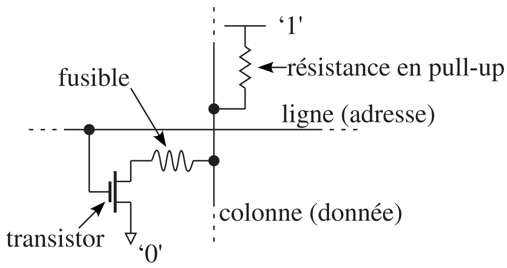


Figure 2.6 : *Une cellule de mémoire PROM, basée sur un transistor et un fusible.*

La technologie de type Erasable Programmable Read-Only Memory (EPROM), introduite en 1971 par Intel est dès lors une nette amélioration de la PROM sur le plan de la flexibilité, puisqu'elle permet une reprogrammation du circuit. Il s'agit d'un nouveau type de transistor, basé sur un transistor MOS, auquel une couche de silicium polycristallin, appelé porte flottante, a été ajoutée, isolée par des couches d'oxyde (Figure 2.7). Dans son état initial, le transistor agit normalement, tel un MOS standard. En appliquant un courant de haut voltage (typiquement 12V) entre la grille et le drain, un effet tunnel charge la porte flottante en électrons, ce qui a pour effet de bloquer le transistor en état ouvert. A la suite de la programmation, quelle que soit la tension appliquée au contrôle, aucun courant ne peut passer entre la source et le drain. La charge créée par la programmation perdure, même lorsque le circuit est hors tension, et ce n'est qu'avec une exposition du circuit à des rayons UV que la programmation est effacée. Lié à l'effacement, un des désavantages de l'EPROM est qu'une fenêtre en quartz doit être apposée sur le circuit, afin de pouvoir laisser passer les rayons UV, ce qui augmente grandement le prix du boîtier.

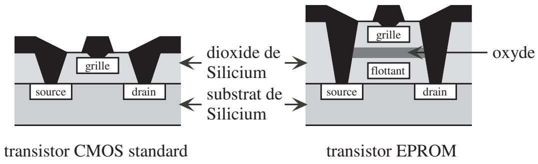


Figure 2.7 : *Un transistor CMOS standard et un transistor EPROM.*

2.3.5 EEPROM/Flash

Le principal inconvénient des EPROMs, la déprogrammation par rayons UV, est oublié dans les EEPROMs, qui sont effaçables électriquement. Il n'est dès lors plus nécessaire d'intervenir physiquement avec de la lumière UV, un système électronique pouvant reprogrammer l'EEPROM de manière autonome. Sur le plan de l'espace pris sur le silicium, une cellule mémoire d'EEPROM occupe toutefois environ 2,5 fois plus de place qu'une cellule d'EPROM, car elle est composée de deux transistors, ainsi que de l'espace nécessaire entre eux deux (Figure 2.8).

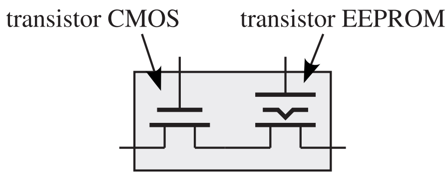


Figure 2.8 : *Une cellule mémoire EEPROM.*

Les mémoires FLASH [24] sont basées sur le même principe d'effacement électronique et de non-volatilité. De nombreuses expériences et implémentations en ont fait des composants très largement utilisés à l'heure actuelle. Vues de l'extérieur, les FLASH sont quasiment identiques aux EEPROMs, si ce n'est que l'effacement se fait par secteur, et non octet par octet. Elles sont toutefois souvent préférées, de par leur vitesse et leur taille, qui peut être plus imposante que celle des EEPROMs.

2.3.6 SRAM

La technologie SRAM, pour Static Random Access Memory, est, contrairement à celles déjà présentées, volatile, et doit donc être reprogrammée à chaque remise en marche du système. Leur second désavantage réside dans la place nécessaire à une cellule, qui est composée de quatre à six transistors formant un latch (Figure 2.9).

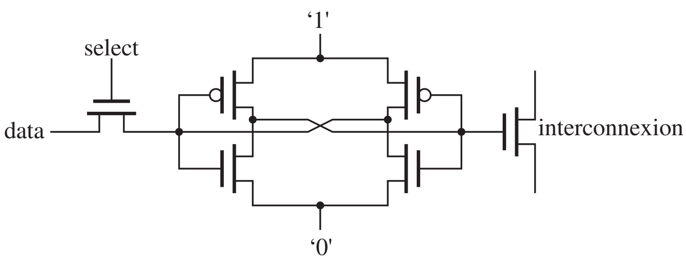
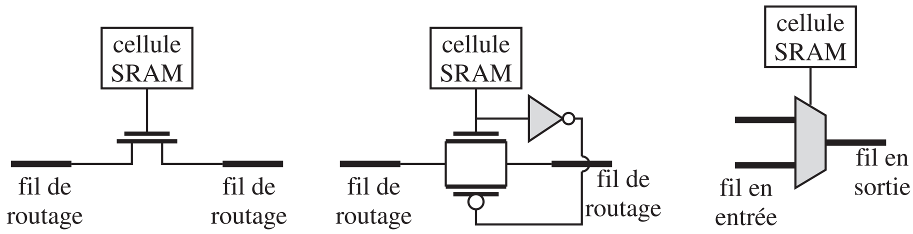


Figure 2.9 : *Une cellule mémoire SRAM.*

Toutefois, la grande facilité de programmation des circuits en technologie SRAM en a fait le choix privilégié des deux plus grands fabricants de FPGAs, à savoir Xilinx et Altera. La figure 2.10 illustre les différentes manières d'utiliser des cellules de mémoire SRAM pour la réalisation de circuits programmables. Les FPGAs commerciaux basés sur cette technologie sont le plus souvent implémentés avec des pass-transistors ou des portes de transmission. Seul le fameux XC6200, que nous décrivons à la page 23 est réalisé avec des multiplexeurs. Cette dernière technique offre l'avantage d'interdire tout court-circuit, contrairement aux deux autres. Toutefois, cet avantage se paie par une baisse de performances en terme de rapidité, un signal mettant plus de temps à passer un multiplexeur qu'un simple transistor.



(a) Pass transistor (b) Transmission gate (c) Multiplexeur

Figure 2.10 : *Trois technologies de programmation associées à la RAM statique.*

2.3.7 Résumé

Nous avons présenté différentes technologies utilisées dans le cadre des circuits programmables. Le tableau 2.1 résume les caractéristiques des trois principales techniques utilisées dans les circuits programmables. Les antifusibles sont d'excellents candidats pour les systèmes nécessitant une très haute fréquence d'horloge, mais ont le très net problème de ne pouvoir être reprogrammés. Les fusibles ne sont plus utilisés, étant donné qu'ils sont couplés à une technologie bipolaire, qui n'est plus d'actualité. Les technologies EPROM et EEPROM permettent cette reprogrammation, mais au coût d'une intervention humaine et un temps d'effacement de une à quinze minutes pour les premières, et d'un temps d'effacement non négligeable pour les deuxièmes, ainsi que d'une forte tension, de l'ordre de 12V pour la reprogrammation. Les cellules SRAM, quant à elles, sont très rapidement reprogrammables, au coût d'une plus grande place occupée et de leur volatilité.

Type	**EPROM**	**Antifusible**	**SRAM**
Rapidité	–	+	–
Densité	–	+	– –
Facilité	+	–	+
Reprogrammabilité	+	–	++

Tableau 2.1 : *Comparaison des caractéristiques des différentes technologies.*

2.4 Circuits logiques programmables

Les circuits programmables sont apparus en 1970, et depuis se sont complexifiés de manière spectaculaire. Il existe dans la littérature plusieurs manières de les classer, certains considérant que les CPLDs (Complex Programmable Logic Device) ne sont pas un sous-ensemble des PLDs (Programmable Logic Device). Nous choisirons toutefois l'approche illustrée à la figure 2.11 consistant à séparer les PLDs en deux sous-classes, à savoir les SPLDs (Simple Programmable Logic Device) et les CPLDs.

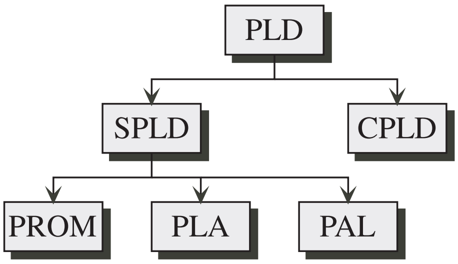


Figure 2.11 : *Une classification des circuits logiques programmables.*

Nous allons décrire les différents types de circuits programmables, dans l'ordre chronologique de leur apparition, correspondant également à leur complexité, en commençant par les SPLD, les CPLDs, puis les FPGAs.

2.4.1 SPLD

Les SPLDs (Simple Programmable Logic Device)[36], dans une description haut niveau, sont composés d'une grille de portes ET et d'une grille de portes OU, les deux étant reliées. Les entrées du système peuvent être connectées aux portes ET, et le résultat des portes OU correspond à la sortie du système (Figure 2.12). Dans ces circuits, les connexions sont préexistantes, les différentes lignes étant reliées par des fusibles, des transistors EPROM, ou des transistors EEPROM. En brûlant certains de ces fusibles, ou en programmant les transistors, il est alors possible de réaliser différentes fonctions logiques.

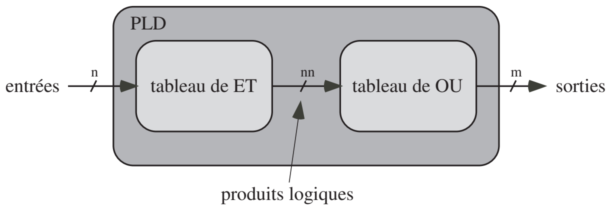


Figure 2.12 : *L'architecture d'un SPLD.*

PROM

Les premiers circuits programmable à avoir vu le jour sont les PROMs (Programmable Read-Only Memory) (Figure 2.13), le terme PROM étant introduit en 1970. Ces circuits sont des mémoires accessibles en lecture, qui contrairement aux ROMs, sont programmables. En effet, une ROM est livrées déjà configurée, et ne peut être accédée

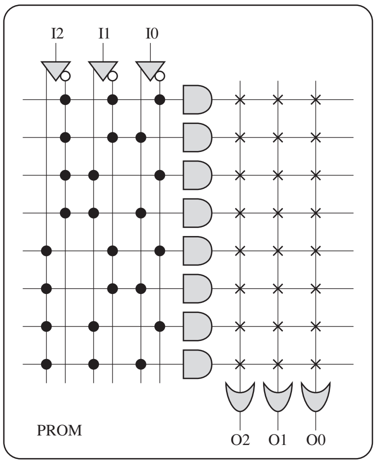


Figure 2.13 : *L'architecture fonctionnelle d'une PROM.*

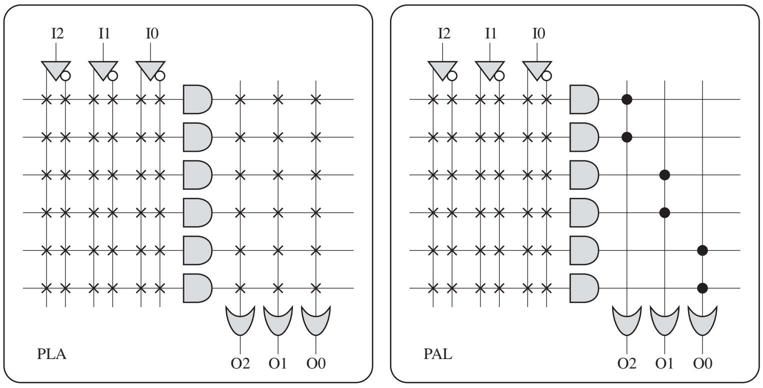
qu'en écriture, alors qu'une PROM est une mémoire vide, qui peut être écrite, une seule fois, par l'utilisateur. Après cette unique écriture, une PROM se comporte exactement comme une ROM. Ces circuits, qui ne sont pas reprogrammables, sont répartis en deux sous-classes, les Mask-Programmable Chips, programmés par le fabricant, et les Field-Programmable Chips, qui sont programmés par l'utilisateur.

Notons également qu'à l'origine les PROMs étaient vouées à faire office de mémoire d'instruction pour les ordinateurs. Leur emploi s'est toutefois généralisé puisqu'on les a utilisées pour l'implémentation de fonctions logiques simples tels que des look-up tables ou des machines d'état. Les PROMs se sont très vite imposées dans ce domaine, de par le fait qu'elles étaient plus petites, moins lourdes, moins chères, et moins sujettes aux erreurs que les systèmes composés de plusieurs circuits comportant des portes logiques. De plus, une erreur de design pouvait être rapidement modifiée en programmant une nouvelle PROM, ce qui était nettement plus rapide et simple que de modifier un circuit imprimé.

L'avantage des PROMs sur les autres PLDs est leur efficacité pour l'implémentation de fonctions logiques nécessitant un grand nombre de produits, mais elles ont le désavantage de n'accepter qu'un nombre limité d'entrées, étant donné que toutes les combinaisons possibles des entrées sont décodées (la figure 2.13 illustre ce décodage par la partie ET et la programmation de la partie OU).

PLA

Les PLAs (Programmable Logic Array) [21] apparurent aux environs de l'année 1975 dans le but de pallier les limitations des PROMs. En effet, dans un PLA (Figure 2.14(a)), contrairement à une PROM, toutes les interconnexions peuvent être programmées, ce qui en fait le PLD le plus général. Il est alors possible de définir les produits et les sommes, les rendant particulièrement efficaces lorsque plusieurs sorties utilisent les même produits. Une amélioration s'accompagnant souvent de désagréments, citons un des désavantages du PLA comparé à la PROM : étant donné que les deux tableaux (ET/OU) sont programmés, le temps de propagation d'un signal de l'entrée à la sortie est nettement plus important que lorsqu'un seul tableau l'est.



(a) *L'architecture d'un PLA* (b) *L'architecture d'un PAL*

Figure 2.14 : *Architectures d'un PLA et d'un PAL.*

PAL

Vers la fin des années 1970, les PALs (Programmable Array Logic) [133] furent introduites afin de contrer le problème de vitesse de propagation des PLAs. Dans un PAL (Figure 2.14(b)), les connexions entres les portes ET et OU sont fixes, et les connexions entre les entrées et les portes ET peuvent être programmées. L'avantage des PALs sur les PLAs est donc leur rapidité, mais elles présentent l'inconvénient de n'avoir qu'un nombre limité de produits pour chaque porte OU.

Résumé

Nous venons de présenter l'architecture générale des PLDs. Avant de continuer notre exploration des circuits programmables, notons que cette architecture a vu bon nombre de variantes voir le jour. De nombreux circuits possèdent la capacité d'inverser les sorties, de disposer de portes tri-state, ou même de faire passer les sorties par des registres. Certains autres ont même la capacité d'utiliser leurs pins comme sorties ou comme entrées supplémentaires.

Sur le plan du routage, nous pouvons noter qu'il est très limité dans le sens où il n'y a pas réellement de connectique à définir. En effet, la programmation d'un PLD agit sur la réalisation d'une fonction plutôt que sur la connexion de divers blocs de fonctions.

2.4.2 CPLD

Les SPLDs présentent deux limitations majeures, à savoir l'impossibilité de réaliser des fonctions à plusieurs niveaux et celle de ne pouvoir partager les produits de différentes fonctions. Les CPLDs (Complex Programmable Logic Device), apparus au début des années 80, sont donc le résultat de l'évolution des PLDs. Ils permettent l'implémentation de systèmes nettement plus complexes, et sont composés d'éléments de base programmables, connectés entre eux par un réseau d'interconnexions relativement simple. Ces éléments de base sont du type SPLD, et peuvent, comme dans le cas de la famille MAX3000 d'Altera [8], être composés d'un tableau de portes ET programmables et de portes OU fixes (une sorte de PAL), ainsi que d'un registre. La technologie de programmation des CPLDs dépend évidemment du constructeur, et peut être de type EPROM, EEPROM, FLASH ou SRAM.

Notons qu'un des avantages des CPLDs sur les FPGAs, que nous présenterons plus loin, est la rapidité. En effet, le réseau d'interconnexions, en étant nettement plus rudimentaire, est plus rapide que celui d'un FPGA. De plus, les connexions se font toujours avec une destination pour une source, et le temps de propagation est donc toujours le même. Le placement d'un design dans un CPLD n'est donc pas critique, et le routage peut être systématisé, sans avoir besoin de tenir compte de contraintes de temps.

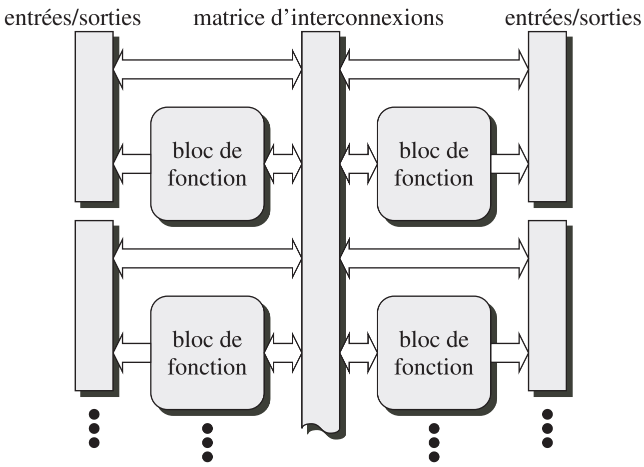


Figure 2.15 : *L'architecture d'un CPLD*

Le routage des CPLDs commence à être intéressant, puisqu'en plus de définir la fonctionnalité de blocs simples comme les SPLDs, il devient possible de les interconnecter.

2.5 FPIC

Classés hors catégorie figurent les Field Programmable Interconnect Chips ou Devices (FPIC ou FPID) [12, 39, 86]. Il s'agit de circuits d'interconnexions programmables qui font office de switchbox. Leur structure est semblable à celle du réseau de routage des FPGAs, la principale différence étant le fait qu'ils ne possèdent aucun élément logique capable d'effectuer du calcul. Placés sur un circuit imprimé, ils permettent de définir au démarrage les connexions entre différentes parties du circuit. Leur avantage est donc de permettre une plus grande souplesse dans la mise au point d'une plateforme comprenant plusieurs circuits intégrés. Par exemple, la réalisation d'un système multi-fpga peut tirer avantage des FPICs dans le but de connecter ensemble les FPGAs avec un circuit également reprogrammable.

2.6 FPGAs

Nous en arrivons au point central de ce chapitre, à savoir les circuits de type FPGA (Field Programmable Gate Array). Au début des années 80, les développeurs disposaient des circuits de type PLD, facilement configurables, mais ne pouvant contenir des designs trop complexes. Les ASICs, quant à eux, supportaient des systèmes de grande complexité, mais n'avaient pas les propriétés de configuration des PLDs. Il manquait donc un type de circuits permettant la réalisation de systèmes complexes, tout en offrant une reconfiguration rapide et peu onéreuse. C'est pourquoi en 1984, Ross Freeman, Bernie Vonderschmitt, et Jim Barnett fondent la compagnie Xilinx. En 1985, ils introduisent sur le marché le premier FPGA, le XC2064, et offrent ainsi une alternative aux précédentes approches.

Les circuits FPGAs [32, 203, 244] permettent d'implémenter des systèmes numériques aussi complexes que ceux réalisés jusqu'alors grâce aux ASICs, tout en ayant le grand avantage de pouvoir être programmés électriquement. Ils sont principalement composés d'un tableau d'éléments plus ou moins complexes pouvant être configurés, ainsi que d'un réseau complexe de connexions également configurables (Figure 2.16).

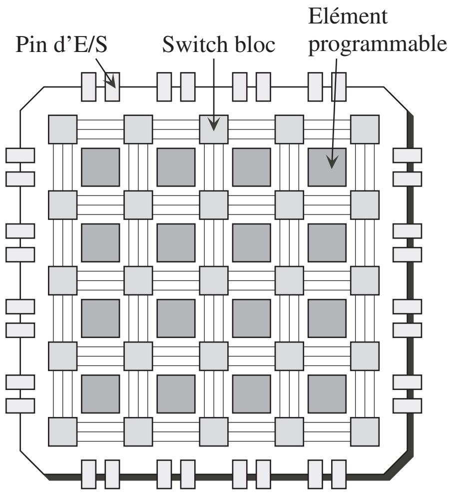


Figure 2.16 : *L'architecture générale du FPGA.*

Nous allons présenter ici deux circuits de Xilinx, les circuits d'Altera présentant le même type d'architecture. Nous aborderons le premier FPGA, le XC2064, puis nous accorderons une attention particulière à la famille XC6200, qui fut la plus utilisée dans les applications de matériel évolutif, décrites à la page 49. Nous verrons ensuite la manière dont les circuits se complexifient en embarquant de plus en plus de composants tels que RAM, multiplicateurs, processeurs, et d'autres. Afin de ne pas surcharger ce chapitre, nous ne présenterons pas les derniers-nés des FPGAs en détail, leur architecture n'étant qu'une variation évoluée du XC2064, mais nous proposerons une comparaison des circuits existant proposés par les six fabricants de FPGAs.

2.6.1 XC2000

Nous allons nous attarder quelque peu sur le premier FPGA commercial, le XC2000 de Xilinx [37]. Son architecture est relativement simple en comparaison des énormes circuits actuels, mais sensiblement identique à l'ensemble des FPGAs. Il est basé sur une technologie SRAM, de la même manière que tous les FPGAs de Xilinx, et est donc reprogrammable un nombre illimité de fois, et ce de manière très rapide.

Son élément de base, le CLB, pour Configurable Logic Bloc (Figure 2.17), est composé d'une bascule et de deux look-up tables de trois entrées qui ont également la possibilité de réaliser une look-up table à quatre entrées (suivant la dénomination introduite par Hill et Woo dans [96], il s'agit d'une look-up table à 4 entrées et 2 sorties [5]). Il est intéressant de noter que les deux sorties, X et Y, sont interchangeables, puisque leurs multiplexeurs ont exactement les mêmes entrées. Ce détail sera intéressant sur le plan du routage, comme nous le verrons plus loin.

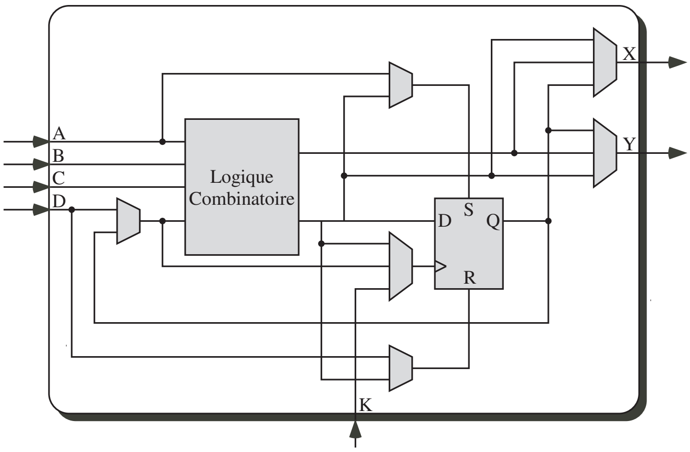


Figure 2.17 : *Le CLB d'un XC2000.*

Sur le plan des interconnexions, le XC2000 contient des switchboxs, chacun étant connecté à quatre autres switchboxs, comme présenté sur la figure 2.18. Ils sont reliés verticalement par cinq fils, et horizontalement par quatre. Ces switchboxs permettent de relier des CLBs sur de longues distances, au travers du FPGA, les problèmes liés au délai RC des noeuds routés étant évités en plaçant des buffers sur certains fils. Le choix a été fait de diviser le FPGA en neuf parties, et de placer des buffers sur toutes les frontières de ces parties. Cette structure de routage est encore présente dans les FPGAs actuels, dans lesquels nous trouvons des canaux de routage accessibles par les unités fonctionnelles, et qui sont reliés entre eux par des switchboxs.

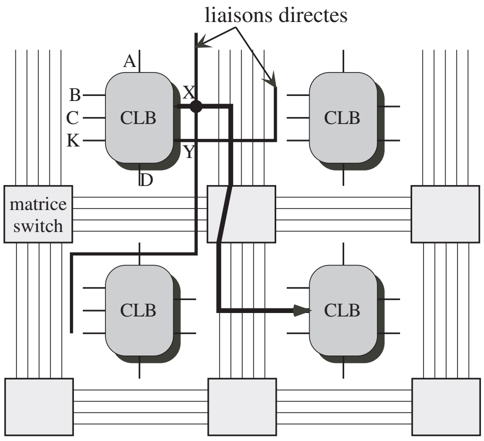


Figure 2.18 : *Le schéma d'interconnexions d'un XC2000.*

[5] Nous pouvons d'ores et déjà noter que l'implémentation du tissu POEtic utilisera la même base.

Afin de pouvoir implémenter des designs encore plus efficacement, des liaisons directes sont également disponibles (Figure 2.18). Elles permettent de connecter un CLB à ses quatre voisins, sans accéder aux liaisons des switchboxs, et réduisent dès lors le temps de propagation du signal, ainsi que les problèmes de congestion du routage.

La figure 2.19 montre les différents bits de configuration liés à l'interconnexion des blocs. Chaque carré correspond à un élément de mémoire SRAM, relié à un transistor qui connecte ou non les lignes verticales et horizontales. Nous pouvons noter que chacune des deux sorties X et Y est connectée à un nombre réduit de lignes. Si le routeur rencontre des difficultés à acheminer un signal, il peut simplement interchanger les deux sorties, pour avoir de nouvelles possibilités de routage. De même pour les entrées, qui ne sont pas connectées aux mêmes lignes, le routeur [6] peut les interchanger, en modifiant toutefois le contenu de la look-up table de manière à garder la même fonctionnalité.

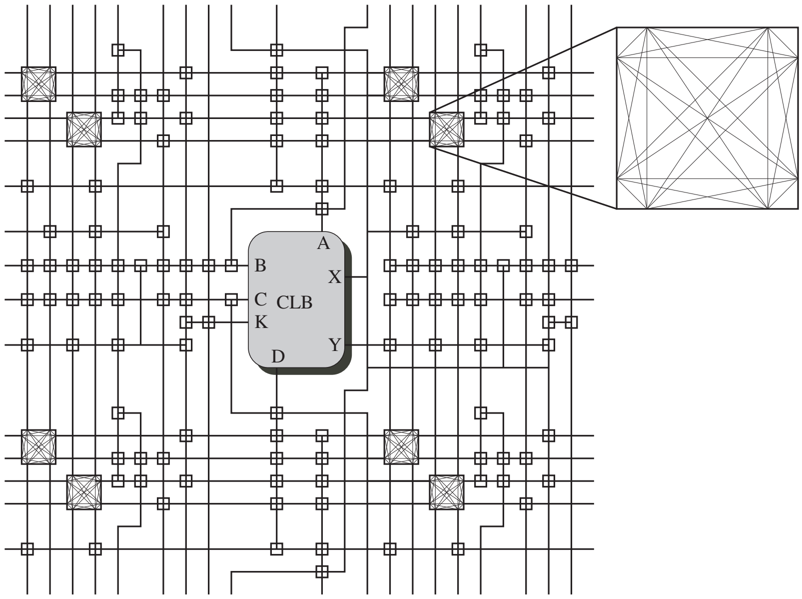


Figure 2.19 : *Les connexions d'un bloc d'un XC2000.*

Pour terminer, il est important de constater que la configuration du FPGA dans un état correct est crucial. En effet, il est très facile de créer des courts-circuits en connectant plusieurs sorties de CLBs ensemble, ce qui peut être très dommageable pour le circuit.

2.6.2 XC6200

La famille XC6200 de Xilinx [258] fut très importante pour l'ensemble de la communauté intéressée par le matériel évolutif. En effet, son architecture est encore plus simple que celle des XC2000, autant en ce qui concerne les blocs logiques que le réseau de routage. De plus, le point central de son attrait est le fait qu'elle est basée sur une technologie SRAM couplée à des multiplexeurs. Tout court-circuit est alors impossible, rendant aisée l'évolution de systèmes au niveau des portes, concept sur lequel nous reviendrons dans la section 3.3.1.

[6] Le concept de routeur est décrit à la section 2.6.7, page 32.

Son bloc de base, présenté par la figure 2.20, ne possède que trois entrées, et sa fonctionnalité n'est pas réalisée par une look-up table, mais par des multiplexeurs, ce qui, dans cette configuration, ne permet pas d'implémenter n'importe quelle fonction à trois variables. Une bascule est placée de la même manière que dans un XC2000, et l'unique sortie peut donc être combinatoire ou séquentielle.

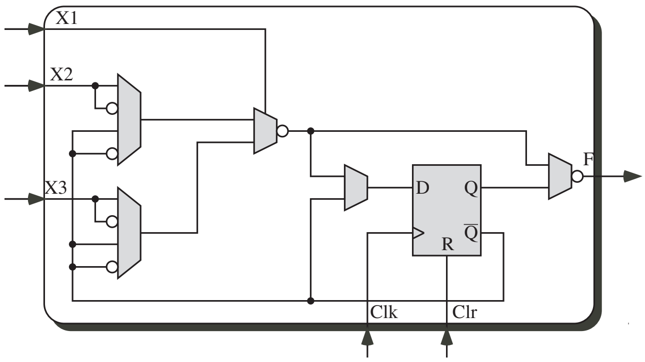


Figure 2.20 : *L'unité fonctionnelle d'un XC6200.*

Concernant les interconnexions, la figure 2.21 détaille la manière dont les entrées et la sortie de l'unité fonctionnelle sont connectées aux voisines. Nous pouvons observer que les valeurs des quatre voisines immédiates peuvent directement être récupérées par l'unité fonctionnelle. Outre ces liaisons courte-distance, il existe des lignes d'une longueur de quatre cellules, seize cellules, ou parcourant le circuit entier. Toutefois, leur nombre est très réduit en comparaison de tous les autres types de FPGAs. Un circuit de la famille XC6200 est donc efficace pour des systèmes ne demandant que peu de ressources de routage de distance plus importante que le simple voisinage.

2.6.3 Architecture MUX versus LUT

Deux types d'architectures sont utilisées par les fabricants, certains basant les éléments logiques sur des multiplexeurs (MUX), à la manière de la XC6200, et d'autres sur des look-up tables (LUT), comme dans la XC2000.

Dans l'approche MUX, la fonctionnalité est réalisée en connectant les entrées et le signal de sélection des multiplexeurs. De la logique peut également être ajoutée, les multiplexeurs sélectionnant alors une parmi plusieurs fonctions.

L'implémentation des éléments logiques grâce à une ou des LUTs permet quant à elle de réaliser n'importe quelle fonction, en programmant correctement les bits de configuration de la LUT. Cette approche a été choisie par Altera et Xilinx, les deux plus gros fabricants. Cette alternative permet d'implémenter des registres à décalage et des mémoires, en tirant parti des bits de configuration des LUTs, ce que les multiplexeurs ne peuvent faire. Son désavantage est toutefois de n'être que peu efficace pour les applications réalisées par un grand nombre de fonctions logiques simples et indépendantes. En effet, une fonction à 2 entrées nécessite la réquisition d'une LUT entière, alors qu'elle pourrait n'être implémentée que sur un multiplexeur. Concernant la taille optimale d'une LUT, les recherches ont montré que l'optimum se situait à 4 entrées [23, 96, 196, 216].

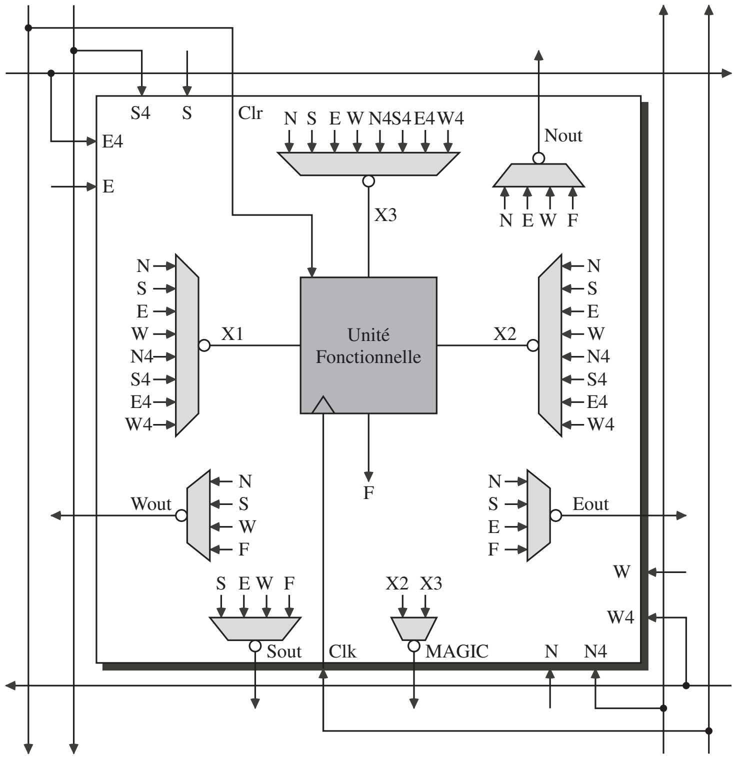


Figure 2.21 : *La cellule de base du XC6200.*

2.6.4 Technologies de programmation

Nous avons déjà présenté en détail les différentes technologies de programmation. Les FPGAs actuels ne tirent parti que des technologies anti-fusibles, EEPROM/FLASH et SRAM, le tableau 2.2, tiré de [146], résumant les caractéristiques de ces approches.

Il est intéressant de noter que les cellules anti-fusibles sont nettement plus petites que les EEPROM, elles-mêmes plus petites que les SRAM. On pourrait donc préférer utiliser des EEPROMs plutôt que des SRAM, ces premières permettant de placer plus de composants sur le même circuit. Cependant les deux plus petites technologies nécessitent l'ajout de plusieurs processus au-dessus de la technologie CMOS standard, et ont donc toujours une ou plusieurs générations de technologie de retard.

Finalement, une nouvelle technologie est peut-être en train de voir le jour. Le 12 juin 2002, Fujitsu annonçait la réalisation d'un nouveau FPGA utilisant une technologie RAM ferroélectrique (FRAM) [180]. L'avantage de cette nouvelle approche réside dans le fait que la mémoire ferroélectrique est non-volatile, et que donc le circuit est directement opérationnel au démarrage. De plus le temps de configuration du FPGA est vingt fois plus rapide qu'une technologie EEPROM ou FLASH, et ne nécessite qu'un courant de 3,3V. L'avenir nous dira si cette technologie saura s'imposer face à celles existantes.

Caractéristique	**SRAM**	**Antifusible**	**EEPROM/FLASH**
Technologie	état de l'art	une ou deux générations de retard	une ou deux générations de retard
Reprogrammable	Oui	Non	Oui
Vitesse de reprogrammation	Rapide	-	3x plus lent que SRAM
Volatile	Oui	Non	Non
Prototypage	Excellent	Non	Bon
Sécurité IP	Non	Oui	Oui
Taille des cellules	Grande (6 transistors)	Petit	Moyen (2 transistors)
Consommation	Moyenne	Basse	Moyenne
Sensible aux radiations	Oui	Non	Plutôt oui

Tableau 2.2 : *Comparaison des caractéristiques des différentes technologies de programmation appliquées aux FPGAs.*

2.6.5 Accroissement de complexité

Les deux FPGAs présentés font bien pâle figure à côté des circuits actuels. En moins de vingt ans d'existence, les circuits se sont complexifiés de manière impressionnante. A l'heure actuelle, la plupart des FPGAs comprennent, outre les blocs de logique, des blocs de mémoire RAM configurables. Dans les Virtex II Pro, un ou deux processeurs sont ajoutés, et dans les Stratix II d'Altera, ce sont des blocs multiplicateurs qui le sont.

Blocs de mémoire

Des applications toujours plus complexes en terme de ressources nécessaires sont implémentées dans les FPGAs actuels, et pour une partie d'entre elles, l'utilisation de mémoires RAM, ROM, CAM, ou FIFO est nécessaire. Les FPGAs fournissent donc la possibilité d'implémenter efficacement de tels éléments, et ceci de deux manières, en utilisant les look-up tables des éléments logiques, ou grâce à des blocs de mémoire spécialement conçus et placés dans le circuit.

Les deux plus grands fabricants de FPGAs, Xilinx et Altera, ont des architectures basées sur des LUTs. Outre la fonction standard consistant à fournir une sortie en fonction des entrées, ces LUTs peuvent servir à la réalisation de registres à décalage ou de blocs de mémoires RAM. En effet, 2^k bits de configuration sont nécessaires à l'implémentation d'une LUT à k entrées, et ces bits peuvent être utilisées à ces deux autres fins. La configuration des FPGAs se faisant en général de manière sériel, la LUT est vue comme un registre à décalage qui peut être exploité par une application. La transformer en bloc de mémoire est un peu plus compliqué, mais permet d'exploiter au maximum les cellules RAMs de configuration (dans le cas de Xilinx et Altera). Typiquement, une LUT à 4 entrées peut être vu comme une mémoire 16x1 bit et de larges mémoires peuvent être réalisées en combinant plusieurs LUTs.

Cette manière d'implémenter des mémoires n'est toutefois pas excellente en terme d'utilisation des ressources pour des mémoires de taille non négligeables. C'est pourquoi la plupart des FPGAs actuels contiennent des blocs de mémoire RAM paramétrables en ce qui concerne la profondeur et la largeur. Dans la série APEX d'Altera par exemple, ces blocs contiennent 2048 bits pouvant réaliser des mémoires de taille 2048×1, 1024×2, 512×4, 256×8, 128×16 ou 64×32. Les blocs sont soit répartis autour des éléments de logique, soit en colonnes, de manière à pouvoir minimiser la longueur des chemins de données.

Reconfiguration partielle

Pour la grande majorité des FPGAs, la configuration ne peut se faire qu'entièrement. Seuls les Virtex de Xilinx et les FPSLIC d'Atmel offrent des possibilités de reconfiguration partielle [154], où certaines parties du circuit peuvent être modifiées sans influer sur le fonctionnement du reste du circuit. La famille Virtex proposant des FPGAs imposants (jusqu'à 8 million de portes équivalentes), il est possible d'y placer de gros systèmes composés de plusieurs modules distincts. Dès lors, un nouvel axe de recherche a vu le jour concernant des systèmes d'ordonnancement de tâches matérielles [225] afin de pouvoir implémenter une application nécessitant plus de place que celle disponible sur le FPGA. Il s'agit donc d'y construire une application composée de modules n'étant pas forcément utilisés au même moment. Un ordonnancement correct permet de ne charger les modules qu'au moment de leur utilisation, grâce à la reconfiguration partielle. Nous sommes persuadés que le pourcentage de FPGAs possédant cette capacité va croître, les fabricants étant toujours à la recherche de plus de flexibilité. Notons toutefois que la reconfiguration partielle d'un circuit de type Virtex n'est pas sans risque, un court-circuit pouvant facilement être généré, avec la potentielle brûlure du FPGA.

Sur le plan des systèmes bio-inspirés, nous verrons plus loin que la reconfiguration partielle est un avantage décisif. Certaines applications mettant en jeu des systèmes évolutionnistes ne nécessitent, pour leur bon fonctionnement, de ne modifier qu'une partie du circuit. Dès lors un gain de temps est obtenu durant la configuration. De plus, alors que les circuits existants sont partiellement reconfigurés par un contrôleur, il serait utile de posséder un tel FPGA capable de s'auto-reconfigurer et où les cellules logiques auraient la possibilité de reconfigurer partiellement d'autres cellules, sans l'intervention d'un agent externe.

Blocs IP

La quantité de logique pouvant être incorporée à un circuit électronique étant à l'heure actuelle très importante, les fabricants en profitent pour intégrer à leurs FPGAs des blocs fonctionnels tels que des multiplicateurs, des MACs (multiplie et accumule), ou même des processeurs. En effet, beaucoup d'applications, notamment en traitement du signal, nécessitent des opérations particulières qui sont difficiles à implémenter grâce aux éléments logiques des FPGAs. Les multiplicateurs intégrés augmentent donc sensiblement la rapidité d'exécution de nombreuses applications.

Les FPGAs contenant un ou des processeurs permettent, quant à eux de conjuguer la flexibilité des processeurs avec la rapidité du matériel. Xilinx et Altera proposent respectivement la série Virtex II Pro et la série Excalibur, contenant chacune un processeur matériel, pouvant communiquer avec les éléments logiques. De même, ils proposent respectivement le MicroBlaze et le Nios, qui ne sont pas matériels, mais logiciels, dans le sens où il s'agit de modules pouvant être intégrés à un design qui sera ensuite programmé sur le FPGA. Ces blocs IP (Intellectual Property) sont loin d'être les seuls, chaque fabricant proposant divers modules pouvant être utilisés par les développeurs, tels qu'interface PCI, UART, et bien d'autres.

2.6.6 Fabricants

Ayant présenté les caractéristiques globales des FPGAS, nous allons rapidement passer en revue les produits proposés par les différents fabricants, à savoir, dans l'ordre alphabétique, Actel, Altera, Atmel, Lattice Semiconductor, QuickLogic et Xilinx [7].

Actel

Les principaux circuits d'Actel sont basés sur des anti-fusibles, les rendant non-volatiles, rapides, et très sûrs sur le plan de la propriété intellectuelle. De plus, les anti-fusibles sont insérés entre les couches de métal, afin d'économiser la surface de silicium. Toutefois, la taille des ces circuits n'est pas exceptionnelle, la série Axcelerator [1] contenant au plus 10'752 cellules composées d'un registre, 21'504 cellules combinatoires (quelques multiplexeurs et portes logiques), et 294'912 bits de mémoire dédiés, pour un total de 2 millions de portes équivalentes [8].

Afin de proposer une alternative à leurs précédents produits tout en maintenant la non-volatilité, Actel a récemment introduit la série ProASIC Plus [2], basée sur une technologie FLASH. Elle est également sûre, les bits de configuration étant cryptés, et va jusqu'à une densité de 1 million de portes équivalentes.

Altera

A l'heure de la rédaction de ces lignes, Altera, le deuxième plus gros fabricant, se concentre sur deux séries de base à technologie SRAM, Cyclone [7] et Stratix [10]. L'optique de la série Cyclone est de disposer d'un FPGA à bas prix. Bien que les éléments de base (LE, pour Logic Element) des deux séries soient semblables, la version Cyclone a été optimisée, et requiert 30% de moins d'espace. Il est principalement composé d'une LUT à 4 entrées, d'une bascule, et d'un système de propagation de retenue, afin d'accélérer les opérations arithmétiques. Le plus imposant des Cyclones contient 20'060 LEs et 288Kbits de RAM, pour un total de 1 million de portes équivalentes.

La série Stratix est nettement plus imposante, pouvant contenir jusqu'à 79'040 LEs, 7'427Kbits de RAM et 22 blocs DSP (88 multiplicateurs 18×18, 178 de 9×9 ou 22 de 36×36). En ajoutant jusqu'à 20 transcievers sériel à 3.125 Gbps et moyennant une baisse de densité (41'250 LEs et 3'423Kbits de RAM), nous obtenons la série Stratix GX [11]. Enfin, la dernière née des séries est la Stratix II [9]. Elle est deux fois plus dense que la première, notamment grâce à une redéfinition de son élément de base. Le LE a été remplacé par l'ALM (Adaptive Logic Module), qui contient deux bascules, deux reports de retenue, et un bloc combinatoire composé principalement de deux 4-LUTs et quatre 3-LUTs. A titre de comparaison, le plus gros Stratix II correspond à 179'400 LEs, 9Mbits de RAM et 384 multiplicateurs 18×18.

Sur le plan des processeurs, la série Excalibur propose un FPGA de type APEX20KE (avec un maximum de 38'400 LEs, soit 1.7 millions de portes équivalentes), auquel a été ajouté un processeur ARM922T tournant à 200MHz. Il

[7] Notez l'attrait du A comme première lettre de nom de Compagnie.

[8] Une porte équivalente correspond à la fonctionnalité d'une porte NAND, soit quatre transistors.

semblerait toutefois qu'il n'ait pas obtenu le succès escompté, et c'est plutôt sur son processeur Nios qu'Altera mise. Il s'agit d'un processeur soft, qui peut être inséré dans n'importe quel design, puis programmé sur le FPGA cible. Sa version 32 bits n'occupe que 1400 LEs, et permet donc d'en placer plusieurs sur un FPGA.

Finalement, Altera propose la solution HardCopy, qui offre à l'utilisateur la possibilité de transformer un design pour FPGA Stratix ou Apex en un ASIC. Basé sur le concept d'ASIC à réseau structuré, le circuit est prédiffusé avec les mêmes composants que ceux du FPGA, et seul l'apposition de deux couches de métal supplémentaires est nécessaire à la réalisation physique du design. L'avantage de cette approche est que la consommation est plus faible et la rapidité plus grande en comparaison de l'implémentation sur FPGA, et que le temps de réalisation est moins élevé que pour un ASIC standard.

Atmel

Atmel, fabricant de semi-conducteurs, propose deux architectures de FPGA, la série AT6000 [15] et la série AT40 [16], toutes deux de technologie SRAM. Le bloc de base de la première est composé de deux multiplexeurs à quatre entrées, et de portes logiques simples, ainsi que d'une bascule, alors que celui de l'AT40, plus conventionnel, est composé de deux 3-LUT, d'une porte ET et d'une bascule. Bien que d'architecture relativement simple, l'originalité de ces deux circuits réside dans le routage. En effet, outre un réseau de routage longue distance simple, chaque cellule est directement reliée à ces 8 voisines, alors que chez la plupart des autres fabricants les liaisons directes ne sont que 4. Cette spécificité permet entre autre d'implémenter efficacement des multiplications de matrice. Concernant la taille, nous pouvons noter que ces circuits sont plutôt petits en comparaison de leur concurrents, la série AT6000 possédant au maximum 75'000 portes équivalentes et la série AT40 1 million, dont 18Kbits de RAM.

Finalement, la série FPSLIC [17] propose un système composé d'un microcontrôleur 8-bit AVR associé à un tableau reconfigurable identique à celui de l'AT40. Ce contrôleur a priori plus simple que l'ARM d'Altera ou le PowerPC de Xilinx, a toutefois la capacité de reprogrammer le FPGA dynamiquement, avantage certain quant à de potentiels systèmes adaptatifs. Notons également que la série Secure FPSLIC offre une EEPROM de 1Mbits, permettant une programmation immédiate au démarrage, sans la nécessité d'une mémoire externe stockant la configuration du FPGA.

Lattice

Les circuits Lattice peuvent être regroupés en 3 familles de FPGA, ORCA, ispXPGA, ECP, et une famille FPSC. La première, ORCA, consiste en trois séries, à savoir les séries 2, 3 et 4. La série 2 est composée d'éléments de base appelés PFU (Programmable Functional Unit) contenant quatre 4-LUT et quatre bascules. Ces PFUs peuvent servir à implémenter des blocs de mémoire RAM et ROM, synchrone, asynchrone ou dual-port, et offrent un maximum de 99'400 portes équivalentes. Les PFUs de la série 3 et des suivantes ont une taille doublée par rapport à la série 2. La série 3 possède également une interface processeur facilitant la configuration et propose des circuits allant jusqu'à 340K portes équivalentes. Enfin, la série 4 [45] offre encore plus de complexité avec des blocs de mémoire additionnels pour un total maximal de 148Kbits, pouvant être utilisés comme RAM, ROM, FIFO, CAM ou multiplicateur, et un maximum de 899K portes équivalentes. Une interface processeur y est également proposée, qui sera exploitée dans les circuits ECP.

Alors que la famille ORCA fut récupérée lors de l'achat de Lucent Technologies, la famille ispXPGA [46] fut entièrement développée par Lattice Semiconductors. Son élément de base est composé de quatre 4-LUT et huit bascules, pour l'implémentation efficace de pipelines. Chaque PFU peut y être utilisé pour la réalisation de 6-LUT, d'une fonction logique jusqu'à 20 entrées, d'un multiplexeur à huit entrées, d'un bloc de 61 bits de RAM, ou d'un registre à décalage de 8 bits. Jusqu'à 246Kbits de RAM sont en outre disponibles, et le nombre maximal de portes équivalentes proposé est de 1'250K. La grande particularité de cette famille concerne sa programmation basée sur des cellules de mémoire E^2CMOS, offrant la non-volatilité à ces circuits. De ce fait aucune mémoire externe n'est nécessaire, et le circuit est automatiquement configuré lors du démarrage.

Suivant la série 4, Lattice propose des Field Programmable System Chip (FPSC) composés d'un tableau reconfigurable identique à celui trouvé dans la série ORCA4 ainsi que d'éléments réalisés en technologie ASIC, pouvant être des interfaces 10Gbits/s, ou backplane transceivers.

Finalement, deux familles, LatticeECP [47], pour EConomy Plus, et LatticeECP-DSP, offrent une solution à bas coûts. Leurs éléments de base sont relativement semblables à ceux de la série ORCA4, mais elles offrent plus de modules utiles à l'implémentation d'applications DSP, tels que des multiplicateurs (jusqu'à quarante 18×18), et des blocs de mémoire. La famille ECP-DSP propose en plus jusqu'à 10 blocs DSP permettant chacun l'implémentation de huit multiplicateurs 9 bits ou quatre Multiply-Accumulate sur 9 bits, par exemple. Le plus grand de ces circuits possède 40 multiplicateurs, 5120 PFUs et 645Kbits de RAM.

QuickLogic

Tous les circuits QuickLogic sont de type anti-fusibles (la technologie ViaLink leur permet d'intercaler le fusible entre deux couches de métal, afin de sauver de l'espace sur le silicium), basés sur un même élément de base, avec une légère différence pour la série Eclipse [188]. Cet élément comprend 2 portes ET à 6 entrées, 4 portes ET à 2 entrées, 6 multiplexeurs à 2 entrées, et une bascule. Cette architecture est particulière en comparaison des concurrents. En effet, elle semble moins intuitive que l'approche LUT, mais permet de créer des fonctions à grand nombre de variables d'entrées comme plusieurs fonctions à moins de variables, grâce à ses 5 sorties. Dans la série Eclipse, la principale différence réside en la présence de deux bascules au lieu d'une. Dans tous les circuits, le réseau de routage est composé de lignes et colonnes connectés par des switchboxs.

La série pASIC consiste en un simple tableau d'éléments logiques et du routage correspondant, pour un maximum de 1584 cellules logiques, soit 75'000 portes équivalentes. Augmentée de blocs de RAM, la série QuickRAM propose jusqu'à 25'344 bits, pour un total de 176'608 portes équivalentes. La série QuickPCI, toujours sur la même base, contient quant à elle un contrôleur PCI, alors que la série QuickMIPS, de loin la plus complexe, possède, outre 1152 cellules, 82'944 bits de RAM et 18 ECUs, un processeur MIPS 32 bits, un contrôleur PCI, 2 UARTs, etc. La série Eclipse, quant à elle, pour un maximum de 662'208 portes équivalentes, peut contenir jusqu'à 82'900 bits de RAM et 18 blocs ECU (Embedded Computational Units), chacun étant composé d'un multiplicateur 8×8, d'un additionneur 16 bits et d'un registre.

Xilinx

A l'heure actuelle, Xilinx, le premier fabricant de FPGAs, propose principalement deux familles, Virtex et Spartan, toutes deux de type SRAM, basées sur une architecture LUT. La différence entre les deux familles est minime, et tient principalement du nombre d'éléments proposés ainsi que du type de process utilisé, les Spartan étant positionnées bas coût en comparaison des Virtex. L'élément de base, le CLB (Configurable Logic Block), est composé de deux Slices, eux-mêmes comprenant deux 4-LUT et deux bascules. Les versions Virtex-II, Virtex-4 et Spartan-3 diffèrent toutefois, le CLB y contenant quatre Slices. Les deux familles contiennent des blocs de RAM, pouvant être utilisés en single ou dual port.

Alors que les séries Spartan-3, Virtex-II et Virtex-4 sont presque identiques en terme d'architecture, la série Virtex-II Pro introduit un ou deux processeurs de type PowerPC. Contrairement à l'échec de la solution d'Altera faisant intervenir un ARM, Xilinx a su imposer son produit, et la série Virtex-II Pro se trouve très bien positionnée dans la gamme de ses produits.

Le plus imposant de la série Spartan-3 [259] propose 5M de portes équivalentes, dont 104 multiplicateur de 18×18 bits et 1'872K bits de RAM. En comparaison, les plus gros Virtex sont le XC4VLX200, de la famille Virtex-4LX [260] et le XC4VFX140, un Virtex-FX. Le premier contient 178'176 éléments logiques, un élément logique étant une LUT à 4 entrées et une bascule. A ceci s'ajoutent 6'048 Kbits de RAM configurables, ainsi que 96 multiplicateurs 18x18 bits, pour un total de 15M de portes équivalentes (chiffre calculé). Le deuxième, le XC4VFX140, est composé de 126'336 éléments logiques, de 9'936 KBits de RAM, 192 multiplicateurs, et 2 processeurs de type PowerPC.

Nous pouvons finalement noter que Xilinx, à l'instar d'Altera, propose une solution appelée EasyPath, permettant la réalisation en ASIC d'un design implémenté sur un Virtex-II, Virtex-II Pro ou un Spartan-3.

Résumé

Le tableau 2.3, en page 34, résume les principales caractéristiques des FPGAs disponibles sur le marché. Nous pouvons observer une complexification constante de ces circuits qui se voient ajouter de plus en plus de nouvelles caractéristiques. Il devient en effet quasiment impossible d'en trouver un ne possédant pas de blocs de mémoire RAM, et la présence de multiplicateurs devient presque systématique.

Sur le plan des structures de routage, la tendance est à l'optimisation. Les technologies utilisées devenant de plus en plus petites, le délai des portes se réduit d'autant, alors que celui des fils ne diminue pas. Dès lors une nouvelle attention est donnée à cette partie du circuit, de manière à assurer un fonctionnement correct à haute fréquence. Concernant la complexité du routage, nous pouvons noter que les réseaux de routage des gros FPGAs ne se bornent pas à une simple grille de switchboxs, mais qu'ils font intervenir plusieurs niveaux. La série Startix d'Altera, par exemple, est composés de Logic Array Blocs (LABs) contenant 10 Logic Elements (LEs), chaque LE étant une LUT, une bascule et quelques portes logiques. Le LAB y a une connectique interne, et une connectique externe pour des chemins de longue distance entre les LABs. Altera voit ce réseau comme une structure à plusieurs dimensions, l'ensemble du réseau étant vu comme une grille de sous-réseaux interconnectés, eux-mêmes étant une grille de sous-réseaux, etc.

2.6.7 Placement/Routage

Le réseau de connexions des FPGAs est des plus complexes, de par le nombre d'éléments à connecter (jusqu'à plus de 150'000). Le placement-routage d'un design pour un circuit donné est donc une tâche laborieuse. Un développeur commence par décrire son système, soit dans un langage de description matériel de type VHDL, Verilog ou SystemC, soit grâce à un éditeur de schéma, tel ViewLogic ou HDLDesigner. Un logiciel s'occupe ensuite de la synthèse, transformant la création du développeur en une liste de composants de base et de leurs interconnexions. Un deuxième logiciel, est alors responsable de placer ces composants dans les éléments de base du FPGA, puis de créer le routage nécessaire au transfert des signaux entre les composants. Cette tâche est relativement simple pour de petits designs, mais peut laisser un processeur dernier cri travailler pendant quelques heures lorsque le design occupe la presque entièreté du circuit. En effet, des modules interconnectés doivent être placés les plus proches possibles, afin de limiter les délais sur les fils, et le logiciel doit trouver le moyen de tous les connecter avec l'ensemble des fils de routage existants.

Afin de réduire le temps de calcul effectué par le PC lors du routage, il serait donc intéressant de disposer d'un FPGA capable de router lui-même certains signaux. De plus, de tous les circuits actuellement existants, pas un seul n'a la capacité de modifier son routage pendant son exécution. Cette caractéristique serait pourtant bien utile à la réalisation de systèmes adaptatifs nécessitant une modification de leur fonctionnement en fonction d'un environnement changeant.

2.6.8 FPGA versus ASIC

Terminons notre survol par un bref aperçu des avantages et désavantages de l'approche FPGA en comparaison des ASICs, en commençant par les désavantages :

- Une fréquence d'horloge moins élevée, pour une même application.
- Une plus grande place nécessaire sur le silicium, beaucoup de logique additionnel étant nécessaire au bon fonctionnement de la programmation du FPGA.
- Dans la même lignée, l'ensemble des éléments programmables ne sont jamais entièrement utilisés. Leur nombre dépend de la taille du design, mais ne peut pas atteindre 100%, le routage devenant quasiment impossible lorsque le circuit est presque plein.
- Pour certains, la grande facilité de tests pousse les développeurs à appliquer une méthodologie de design “essayer-et-voir-ce-qui-se-passe”, ce qui n'encourage pas les méthodes formelles de vérification des circuits.

Et enfin les avantages des FPGAs :

- Un faible coût de développement, le prototype ne nécessitant pas de réalisation matériel, mais seulement des tests successifs sur un FPGA.
- Peu de risques, dans le sens où une erreur de design est très vite corrigée et n'implique pas la création d'un nouveau circuit.
- Une grande rapidité lors de la réalisation d'un prototype.
- La possibilité de pouvoir réaliser des applications de matériel évolutif, laissant un algorithme génétique trouver la solution à un problème donné.

2.7 Conclusion

Nous venons de présenter un survol des circuits reprogrammables, prêtant une attention particulière aux FPGAs, qui sont les circuits reconfigurables les plus flexibles et offrant le plus de fonctionnalités. Ils sont les meilleurs candidats pour la réalisation rapide de prototype, de par leur capacité à être reconfigurés un très grand nombre de fois.

Dans le chapitre suivant nous verrons qu'ils ont été largement utilisés par la communauté bio-inspirée dans la réalisation de réseaux de neurones artificiels notamment, ou dans le développement de matériel évolutif. En effet, certaines applications possédant un parallélisme inhérent peuvent tirer profit d'une implémentation matérielle efficace.

Nous avons également vu que leur configuration est lancée par un agent externe, typiquement un petit contrôleur accédant une mémoire qui contient les bits de configuration du FPGA. De plus le placement et le routage des éléments est calculé par un ordinateur, et ne peut être modifié durant l'exécution d'une application que sur peu de circuits (Virtex et FPSLIC), et de façon peu aisée, avec le risque de détruire le circuit.

Dès lors, nous allons proposer, dans le chapitre 6 un nouveau FPGA possédant des capacités d'auto-reconfiguration et d'auto-routage. Les cellules de base y auront le pouvoir de reconfigurer partiellement d'autres parties du circuit ainsi que de créer des chemins de données durant le fonctionnement du circuit.

Vendeur	**Famille**	**Technologie**	**Portes Equivalentes**	**Nb Flip-flops**	**Mémoire Bloc (Kbits)**	**Multiplicateur 18×18**	**Processeur Processeur**
Actel	Axcelerator	anti-fusible	2M	21'504	294	-	-
Actel	ProASIC Plus	FLASH	1M	56'320	198	-	-
Altera	Cyclone	SRAM	1M	20'060	295	-	-
Altera	Stratix	SRAM	4M (calcul)	79'040	7'427	176	-
Altera	Stratix II	SRAM	9M (calcul)	143'520	9'383	384	-
Altera	Excalibur	SRAM	1.7M	38'400	256	-	1 ARM922T
Atmel	AT40	SRAM	50K (Usable gates)	3'048	18.4	-	-
Atmel	AT6000	SRAM	30K (Usable gates)	6'400	-	-	-
Atmel	FPSLIC	SRAM	50K (déduit)	2'962	18.4	-	8-bit AVR
Lattice	ORCA4	SRAM	899K	18'216	148	-	-
Lattice	ispXPGA	E^2CMOS	1.25M	30'700	414	-	-
Lattice	ECP	SRAM	1M (calcul)	46'080	645	40	-
QuickLogic	Eclipse II	anti-fusible	320K	4'002	55	-	-
QuickLogic	pASIC	anti-fusible	75K	2'692	-	-	-
QuickLogic	QuickRAM	anti-fusible	176K (90K Usable PLD gates)	2'692	25	-	-
QuickLogic	QuickMIPS	anti-fusible	457K (115K Usable PLD gates)	4'032	83	18	MIPS32 4Kc
Xilinx	Virtex-II	SRAM	8M	93'184	3'024	168	-
Xilinx	Virtex-II Pro	SRAM	8M	88'192	7'992	444	2 PowerPC
Xilinx	Virtex-4LX	SRAM	15M (calcul)	178'176	6'048	96	-
Xilinx	Virtex-4FX	SRAM	11M (calcul)	126'336	9'936	192	2 PowerPC
Xilinx	Spartan-3	SRAM	5M	66'560	1'971	104	-

Tableau 2.3 : *Comparaison des caractéristiques des différentes FPGAs.*