CIRCUITS LOGIQUES PROGRAMMABLES

Bien concevoir avec un *FPGA*

Réussir une application à base de FPGA, c'est respecter un certain nombre de principes comme une bonne connaissance de la topologie du réseau logique programmable et l'adoption d'une méthodologie synchrone.

Par Edgard Garcia,

MVD

Edgard Garcia a créé la société MVD, spécialisée dans le traitement d'images et référence française en matière de FPGA Xilinx. Depuis de nombreuses années, MVD conçoit des FPGA en langage VHDL pour tous types d'applications, y compris le prototypage d'Asic.

Dès l'origine, les FPGA, tels que Xilinx les a inventés, avaient la réputation de mettre à disposition de l'utilisateur une conception rapide, fiable et simple. Si cette réputation est totalement justifiée, les progrès technologiques ont permis, depuis, d'accéder à des matrices logiques programmables de plusieurs millions de portes. Cette complexité actuelle reste absolument gérable et permet la réalisation d'applications très performantes moyennant une bonne connaissance des ressources offertes et le respect d'une méthodologie de conception.

Malheureusement, certains concepteurs débutants, voyant dans la souplesse du programmable une possibilité infinie de corrections d'erreurs, dédaignent toute méthode et connaissent des échecs. Ces déboires, relevant de leur entière responsabilité, les amènent parfois à déclarer à tort que l'utilisation de certains FPGA haut de gamme est difficile. L'objet de cet article est de montrer que les adjectifs «simple» ou «difficile» sont hors de propos pour ce type de conception. C'est la réussite du projet qui est facile à obtenir si l'on suit avec sérieux et méthode un certain nombre d'étapes.

La haute densité des matrices de Xilinx n'est pas un obstacle car un jeu d'outils de développement accompagne efficacement l'utilisateur. Les quatre règles de base sont:

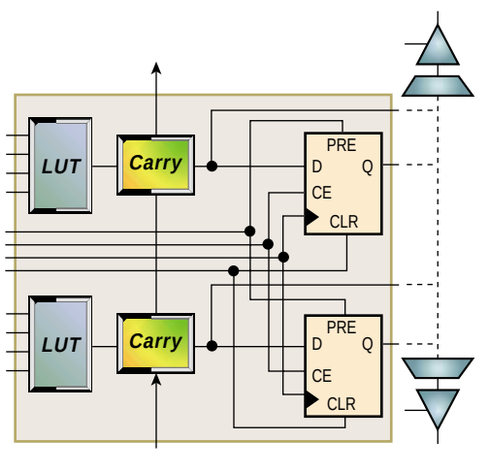
- bien connaître les caractéristiques du FPGA ciblé pour assurer son adéquation avec les besoins du projet;
- respecter une méthodologie de conception et un style d'écriture;
- opter pour des outils de synthèse de qualité;
- maîtriser les outils d'implantation.

Autrement dit, il faut bien comprendre les possibilités offertes par le réseau logique programmable choisi ainsi que les moyens à mettre en œuvre pour en tirer le meilleur profit.

En nous appuyant sur différentes architectures de FPGA, nous allons décrypter les principales possibilités mises à la disposition des utilisateurs et les points qu'ils doivent absolument connaître et maîtriser, pas seulement pour concevoir simplement et rapidement, mais aussi pour réussir leurs conceptions. La démonstration de cette méthode de

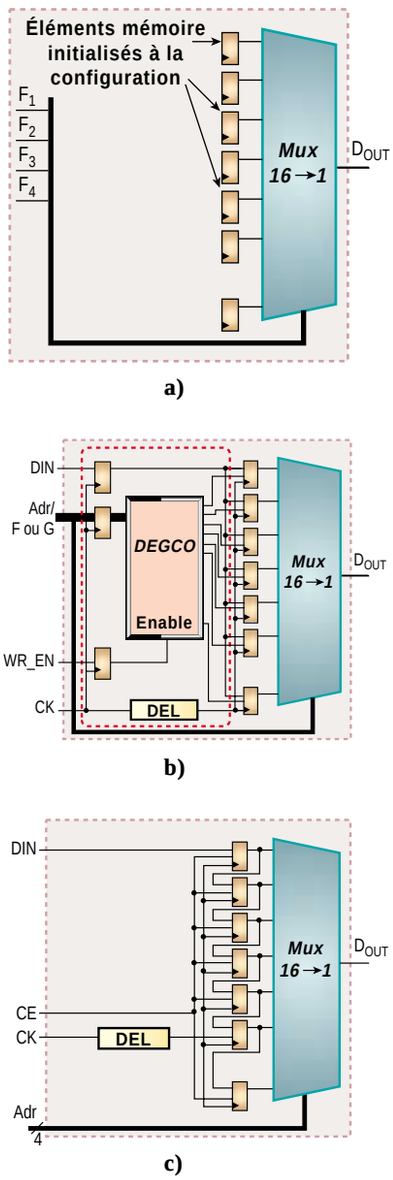
Architecture simplifiée d'un «slice» — FIGURE 1

Un bloc de logique combinatoire est constitué de deux ou quatre «slices».



Trois modes de configuration des LUT — FIGURE 2

En a, un générateur de fonctions combinatoires. En b, configuration en mémoire 16 bits à écriture synchrone. En c, configuration en registre à décalage de longueur programmable.



conception sera faite sur des FPGA Xilinx. Les familles proposées sont Spartan-II, Spartan-IIE, Virtex-E et Virtex-II.

Bien connaître son sujet

Les qualités et l'organisation générale des matrices de ces familles sont décrites dans l'encadré ci-dessous. Nous y voyons qu'un des modules de base est le bloc logique configurable (ou CLB), lui-même constitué de « slices ». La figure 1 illustre l'architecture simplifiée d'un slice. La logique combinatoire est implantée grâce aux LUT (look-up tables) contenues dans chaque slice. Ces LUT peuvent également être configurées comme éléments de mémoire synchrone, simple ou double-port de 16 bits, ou encore comme registres à décalage de 16 bits.

Il existe donc trois modes de configuration de ces LUT. Plus précisément, le fonctionnement en mode combinatoire est obtenu en lisant le contenu pointé par les signaux d'entrée (figure 2a). Autrement dit, les LUT sont des mémoires dont le contenu est initialisé lors de la configuration du FPGA. De ce fait, elles permettent à l'utilisateur d'en disposer en mode « élément mémoire » dans chacun des slices si nécessaire (figure 2b). La figure 2c décrit le mode de configuration particulier en registre à décalage de longueur programmable jusqu'à 16 bits.

Par ailleurs, une logique supplémentaire utile pour la réalisation de fonctions arithmétiques est disponible dans chaque slice. Grâce à ces éléments, et au style d'écriture adapté, des modules de type accumulateur chargeable en addition/soustraction pourront être implantés à raison de 2 bits par slice. Le niveau de performance est alors prédictible et il pourra dépasser les 200 MHz pour des opérateurs 32 bits.

A noter que l'appel à ces éléments de logique arithmétique, comme la propagation rapide de la retenue (figure 3), implique l'utilisation de ressources de routage dédiées. En effet, afin de préserver la performance de telles fonctions, la propagation de la retenue s'effectue du bas vers le haut via des connexions directes entre slices adjacents. Ainsi, les LSB sont vers le bas de la matrice et les MSB sont vers le haut, ceci par rapport à la vue du FPGA Xilinx dans les outils graphiques que sont le « Floorplanner » et le « FPGA Editor ».

Les bascules dans chaque slice ont aussi des caractéristiques importantes pour le concepteur. En particulier, elles sont initialisées systématiquement à la mise sous tension (par défaut à la valeur '0'), et sont utilisables indépendamment de la logique combinatoire disponible dans le même slice. En outre, chaque bascule bénéficie de broches de contrôle telles que : entrée dédiée de validation de l'horloge (Clock enable) permettant d'activer ou de suspendre le fonctionnement de chacune des bascules individuellement, et ceci sans avoir à insérer de la logique combinatoire sur le chemin de l'horloge ; entrées de « set » et de « reset » synchrones ou asynchrones. La polarité des signaux d'horloge, de Clock enable, de set et de reset est programmable pour chacune des bascules. Autrement dit, ces signaux peuvent être individuellement actifs au niveau haut ou au niveau bas.

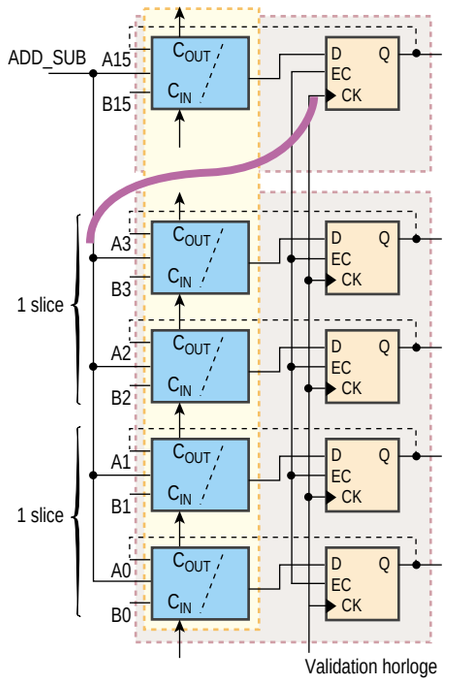
Les outils de synthèse appropriés permettront de tirer profit de ces détails d'architecture à partir du code source VHDL (ou Verilog).

Maîtriser la propagation des signaux

Une attention particulière doit être apportée à la distribution des horloges, élément crucial dans tout projet d'électronique numérique. La plupart des FPGA, et notamment ceux de Xilinx, disposent de ressources (buffers et routage associé) qui permettent une

Configuration de « slices » pour fonctions arithmétiques — FIGURE 3

Une logique supplémentaire (Fast Carry) permet l'implantation de fonctions de type accumulateur chargeables en addition/soustraction.



Performances et densités à la carte

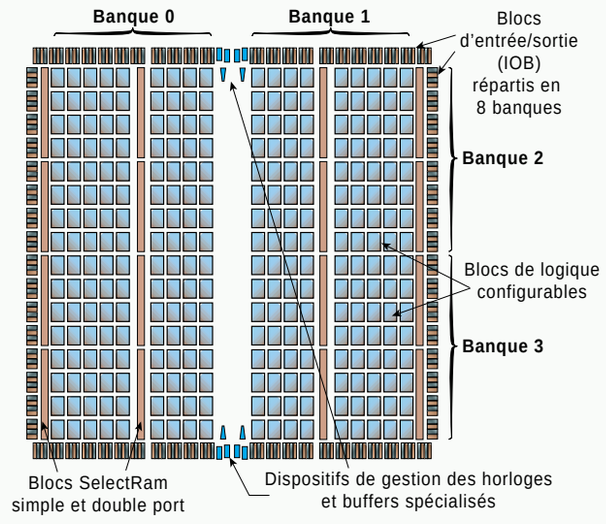
→ Les différentes architectures de FPGA Xilinx sont Spartan-II, Spartan-IIE, Virtex-E et Virtex-II. Leurs caractéristiques principales sont :
- Complexités allant de 15 000 à plus de 8 millions de portes.
- Faible consommation.
- Grande souplesse d'utilisation des entrées-sorties avec adaptation d'impédance (Virtex-II) et configuration en mode différentiel (Spartan-IIE, Virtex-E et Virtex-II).
- Fonctions mémoire (distribuée et blocs de Ram).
- Dispositifs de gestion des horloges (DLL et DCM).
- Multiplieurs câblés (Virtex-II).

Et bien d'autres possibilités permettant d'optimiser à la fois la performance et la densité des fonctions logiques et/ou arithmétiques.

Comme le montre la figure ci-après sur l'organisation générale des FPGA Xilinx, l'essentiel des fonctions logiques sera implanté en CLB (blocs de logique configurable), eux-mêmes constitués de « slices ». Un « slice » comprend deux bascules D, de la logique combinatoire et arithmétique et/ou des fonctions de mémoire distribuée ou de registre à décalage. Suivant les familles, un CLB peut être constitué de deux ou quatre slices, c'est-à-dire qu'il peut comprendre jusqu'à huit bascules D, et, par exemple, 128 bits de mémoire ou des fonctions arithmétiques et/ou logiques.

Organisation des FPGA de Xilinx — FIGURE

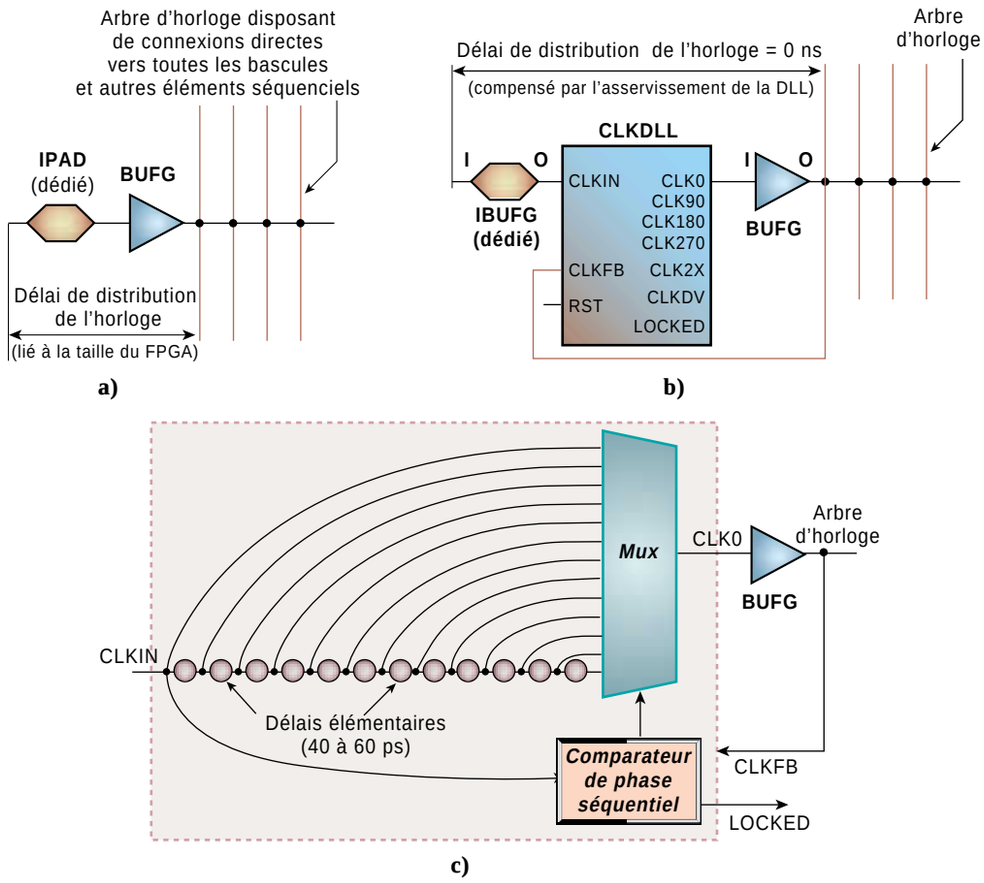
Les matrices de Xilinx comportent essentiellement des blocs logiques configurables.



Ressources de routage pour les horloges

FIGURE 4

En a, les buffers d'horloge et les ressources de routage associées. En b, des dispositifs de gestion des horloges permettent d'adapter la fréquence d'horloge. En c, diagramme simplifié d'un dispositif de gestion de la phase et de la fréquence de l'horloge.



répartition d'horloge parfaite, c'est-à-dire sans dérive (skew); un point particulièrement important pour les conceptions synchrones. Le déroulement des événements est le suivant: les outils de synthèse détectent automatiquement les sources d'horloge, et infèrent les buffers spécialisés et les ressources de routage associées (figure 4a). A la charge du concepteur de choisir une matrice offrant un nombre de tampons d'horloge suffisant pour son application.

Les dispositifs de gestion des horloges permettent l'implantation aisée et efficace de fonctions comme la multiplication et la division de la fréquence, l'ajustement statique ou dynamique de la phase ou la synthèse de fréquence. Ces deux dernières caractéristiques sont propres aux FPGA Virtex-II. Grâce à cette technologie entièrement numérique, le concepteur peut faire appel à toutes ces fonctions de gestion d'horloge, sans avoir à se soucier des filtres de boucles et autres critères habituellement rencontrés avec les technologies analogiques (figure 4b et 4c). La gestion de la phase et de la fréquence d'horloge s'effectue en utilisant un grand nombre de lignes à retard (50 ps environ), ce qui permet de garantir une gigue ne dépassant pas 100 ps dans le pire des cas.

Dans les FPGA Xilinx, les ressources de routage sont fortement structurées. Elles sont constituées de plusieurs types d'interconnexions (directes, générales…) et de longues lignes de métal horizontales et verticales. Ces dernières vont permettre le routage efficace de certains types de signaux.

Pour résumer, nous pourrions dire que le flot de données principal doit circuler en horizontal (par rapport à la vue graphique de la matrice dans les outils Xilinx), et que les signaux de contrôle doivent être distribués en vertical pour optimiser le routage. Cette optimisation engendrera une meilleure performance, de même qu'une réduction des temps de placement/routage. La consommation sera également réduite car les charges capacitives inhérentes au routage seront minimales.

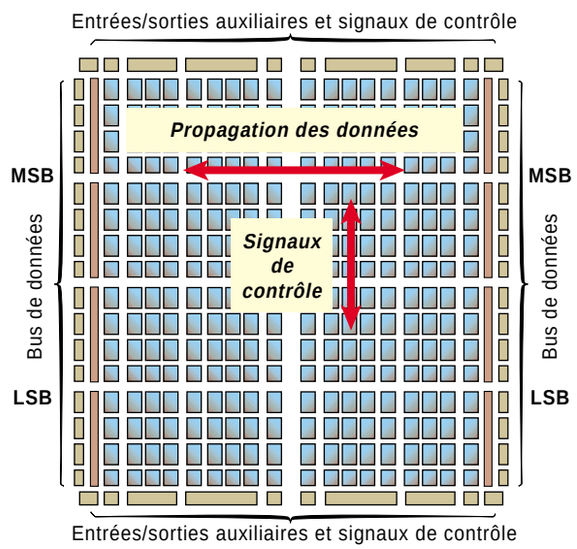
Cependant, pour garantir un routage efficace, rien de tel que de suivre quelques règles simples pour définir le brochage du FPGA. En effet, un bon positionnement des blocs d'entrées/sorties —généralement décidé avant l'implantation sur le FPGA pour paralléliser la réalisation des lignes de cuivre— donne au logiciel toutes facilités pour utiliser les ressources de routage optimales pour chacun des signaux. En outre, si la méthode d'implantation des fonctions arithmétiques et assimilées, telles que compteurs, comparateurs, etc., est correctement respectée (position verticale avec LSB vers le bas, MSB vers le haut du FPGA), les bus principaux d'entrées/sorties suivront la même loi de positionnement. Les entrées/sorties restantes seront placées de manière intuitive en tenant compte des connexions entre les divers modules. Si la figure 5 présente ces principes de positionnement pour les entrées/sorties, précisons les caractéristiques de ces blocs d'E/S. La règle est toujours la même: plus vous connaîtrez les possibilités offertes, meilleures seront les performances.

L'échange de données avec la circuiterie externe (microprocesseur, DSP,

Les règles de positionnement des E/S

FIGURE 5

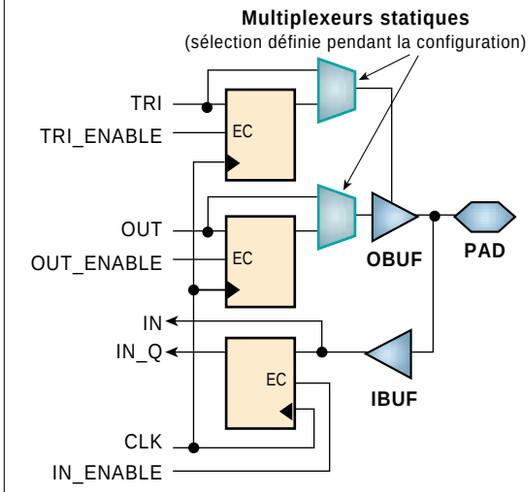
Pour obtenir des performances optimales, il est important de respecter les lois d'implantation des bus principaux et des entrées/sorties.



mémoires rapides, convertisseurs A/N et N/A) est souvent un critère important dans la perspective des performances à atteindre. Les blocs d'entrées/sorties disposent de bascules sur les chemins d'entrée, de sortie et de contrôle trois-états. Les outils de synthèse les plus sophistiqués permettent l'utilisation systématique (lorsque cela s'applique) de ces bascules, garantissant ainsi les temps d'établissement et de «clock-to-out» puisqu'ils deviennent indépendants du placement et donc du routage. Enfin, les FPGA de la famille Virtex-II comprennent des registres doubles sur chacun de leurs trois chemins, autorisant ainsi la réalisation d'interfaces DDR (Double Data Rate) pour la communication avec les mémoires SDram du même nom ou autres dispositifs du même type (figure 6).

Configuration des blocs d'E/S — FIGURE 6

Les blocs d'entrées/sorties (IOB) comportent des bascules sur les chemins d'entrée, de sortie et de contrôle trois-états, bascules utilisées, si nécessaire, par les outils de synthèse.



La configuration électrique des blocs d'E/S donne la possibilité d'ajuster la raideur des fronts sur les étages de sortie, la sortance, les seuils de commutation et les standards de communication (LVTTL, LVCmos, SSTL, PCI, LVDS…). Par ailleurs, la famille Virtex-II offre d'adapter en impédance aussi bien les entrées que les sorties, sans avoir à insérer les traditionnelles résistances d'adaptation. Résultat: un gain de place important sur les circuits imprimés et une grande souplesse.

Savoir tirer un bon parti des blocs de Ram

Les familles Spartan-II, Spartan-IIE et Virtex-E possèdent des blocs de mémoire (Ram double-port) à écriture et lecture synchrones de 4Kbits. Leur nombre va de 4 blocs pour la matrice 2S15, à 96 blocs pour le FPGA V1000E. De son côté, la série Virtex-II dispose de blocs de Ram d'une capacité de 18Kbits chacun. Ils sont entre 4 et 200 suivant le FPGA. Chacun de ces blocs de 18Kbits est «accompagné» d'un multiplieur 18x18 bits signés. Chaque bloc est configurable individuellement et ceci sur chacun des deux ports A et B. Pour les capacités de 4Kbits, les dispositions possibles sont: 4Kx1, 2Kx2, 1Kx4, 512x8 et 256x16. Ceux de la série Virtex-II acceptent: 16Kx1, 8Kx2, 4Kx4, 2Kx9, 1Kx18 et 512x36.

Outre l'efficacité d'implantation de fonctions mémoires et/ou Fifo, cette architecture permet également la conversion automatique et instantanée de la largeur des bus entrants et sortants. Ainsi, il est possible par exemple de recevoir (et d'envoyer) un flot de données série à 180MHz sur le port A, alors que les données seront exploitées à une fréquence moyenne de 5MHz et 36 bits sur le port B, sans que le concepteur ait à se soucier de la mise en parallèle ou de la sérialisation.

Enfin, dans certains cas, la souplesse de la configuration et des signaux de contrôle de ces blocs Ram autorise l'implantation de fonctions logiques complexes avec une efficacité surprenante. Pour plus d'informations, se référer aux articles sur l'implantation de machines d'états en blocs Ram et sur un convolueur bidimensionnel en matrice 9x9 fonctionnant à 200MHz (*Electronique* n°103, p.69 et n°119, p.58, et sur le site www.mvd-fpga.com).

Méthodologie de développement: vous avez dit synchrone?

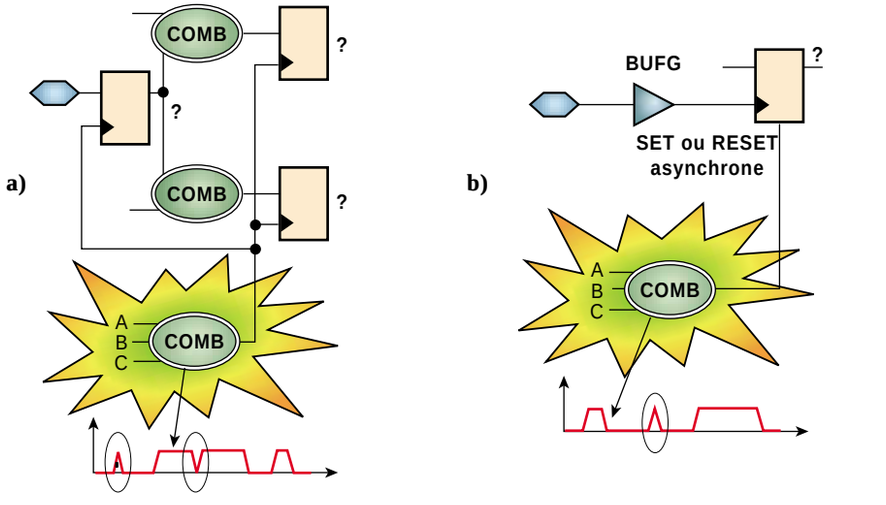
Il est important de rappeler quelques notions de base sur la méthodologie à adopter dans le cadre de conception d'électronique numérique. La citation suivante donne une idée assez précise des ennuis encourus si une certaine méthodologie de conception n'est pas prise en compte: *«Les développements asynchrones peuvent ruiner votre projet, votre carrière professionnelle et même votre santé. Refusez simplement de concevoir en asynchrone…»*

Les conceptions asynchrones présentent facilement des défauts de fonctionnement aléatoires. Certaines d'entre elles manifesteront des problèmes de fonctionnement en température, d'autres liés au vieillissement du composant ou autres paramètres externes.

La figure 7a montre un des montages source d'asynchronismes et donc de fonctionnement aléatoire communément appelé «Gated clock». Dans cet exemple, l'horloge est générée à partir d'une fonction combinatoire de plusieurs signaux. La sortie de cette fonction présentera des parasites qui pourront être interprétés par certaines bascules comme des fronts d'horloge; d'où un fonctionnement non prédictible des bascules. De plus, dans ce type de montage, le signal de sortie de la logique combinatoire sera normalement routé par des ressources d'interconnexions générales. Le temps de propagation jusqu'à l'entrée des bascules présentera un décalage (skew) difficilement contrôlable. Ceci augmentera encore le risque de fonctionnement aléatoire. Dans l'exemple illustré sur la figure 7b, une fonction combinatoire va effectuer un «set» (ou un «reset») asynchrone d'une ou plusieurs bascules. Comme dans toute

Quelques pièges de l'asynchronisme — FIGURE 7

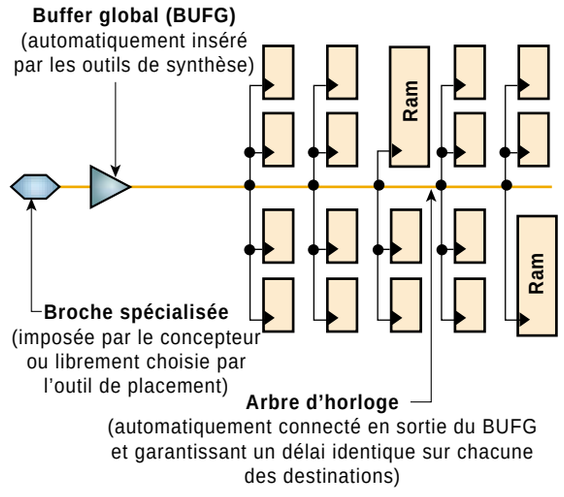
En a, ne pas générer une horloge à partir d'une fonction combinatoire. En b, ne pas générer un signal de contrôle à partir d'une fonction combinatoire.



fonction combinatoire, le signal de sortie ne sera généralement pas exempt de parasites, et des remises à «1» intempestives sont à prévoir.
Afin d'éviter ces problèmes très difficilement maîtrisables par des techniques d'implantation complexes et coûteuses, il est important d'adopter une méthodologie adéquate. Un développement fiable doit pouvoir être reproduit de manière totalement prédictible et indépendamment de la technologie utilisée qui évolue tous les jours. Il doit présenter un fonctionnement sans faille quelle que soit la fréquence de fonctionnement, du continu jusqu'à une fréquence limite, dans toute la plage active de l'application.
Il se trouve que cette reproductibilité est particulièrement aisée à obtenir: il suffit pour cela de concevoir avec une méthodologie synchrone. Un développement (ou une partie de développement) synchrone est entièrement séquencé par une horloge. Celle-ci est distribuée sur des ressources spécialisées (buffer global ou BUFG, et arbres d'horloges associés), de manière à garantir la simultanéité parfaite des fronts sur tous les éléments destinataires. La figure 8 illustre quelques recommandations utiles pour une conception synchrone. Les bascules et autres éléments séquencés reçoivent l'horloge précisément au même moment, car celle-ci est distribuée par un arbre spécialisé, grâce au buffer global, lui-même automatiquement inféré par l'outil de synthèse sur l'analyse du code. Le set (ou reset) asynchrone n'est pas utilisé en cours de fonctionnement, mais seulement pour l'initialisation lors de la mise sous tension.
Lorsque deux modules travaillant dans des domaines d'horloges différents échangent des données, le concepteur devra prendre les précautions nécessaires pour s'assurer de la validité de l'interprétation des données par l'élément récepteur. La figure 9 expose l'une des manières de faire communiquer entre elles deux portions de circuit séquencées par des horloges asynchrones. Dans cet exemple, les échanges de données entre les deux domaines d'horloge s'effectuent par l'intermédiaire d'une mémoire Fifo. L'horloge CK_2 est utilisée comme horloge d'écriture, alors que la lecture est séquencée par CK_1. Les signaux «autorisation d'écriture» (WR-ENA) et «autorisation de lecture» (RD-ENA) sont respectivement générés à partir des horloges CK_2 et CK_1. L'asynchronisme entre les deux horloges est absorbé dans la Fifo. Sa profondeur devra prendre en compte les différences de cadence sur chacun des deux ports.
De nombreux autres cas de figure peuvent être envisagés. En tout état de cause, le passage d'informations d'un domaine d'horloge vers un autre devra faire l'objet d'une attention particulière pour assurer l'intégrité du transfert. ■

Quelques conseils pour une conception synchrone — FIGURE 8

Le respect de ces quelques principes assure que tous les éléments reçoivent les signaux nécessaires (dont l'horloge) à l'exact moment prévu.



Echanges de données sur deux horloges (CK_1 et CK_2) — FIGURE 9

Dans cet exemple, l'échange des données entre les zones du circuit, contrôlées l'une par CK_1 et l'autre par CK_2, s'effectue par l'intermédiaire d'une mémoire Fifo.