

La conception de FPGA mise en œuvre avec le standard OpenCL

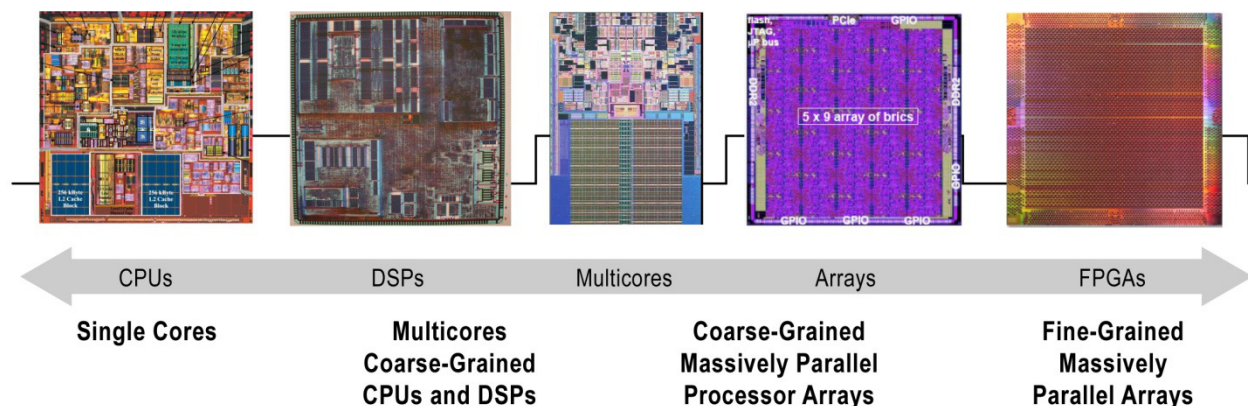
Deshanand Singh, ingénieur en chef pour la supervision des logiciels et de l'IP chez Altera Corporation

Durant des décennies, les technologies programmables se classaient suivant deux concepts opposés de l'aspect « programmabilité ». Le premier de ces concepts faisait appel aux CPU à un seul cœur et aux unités DSP. Ces composants étaient programmables grâce à des logiciels contenant la liste des instructions à exécuter. Ces instructions étaient créées suivant un modèle séquentiel par le programmeur. Cependant, un processeur avancé pouvait modifier l'ordonnancement des instructions pour en faire ressortir le parallélisme sous-jacent au moment de l'exécution. Le second concept intervenant dans les technologies programmables était représenté par les FPGA. La capacité de programmation de ces composants repose entièrement sur le fait qu'il s'agit de circuits matériels configurables sur lesquels l'exécution est intrinsèquement parallèle. Un concepteur utilisant un FPGA crée, par essence même, une application massivement parallèle à granularité fine. Pendant des années, ces deux concepts extrêmes ont cohabité, et chacun des styles de programmabilité qui en découlent intervenait dans des domaines d'application différents. Cependant, la tendance vers une diminution toujours plus poussée des géométries des circuits intégrés a favorisé l'avènement des technologies actuelles qui sont à la fois programmables et parallèles.

Par ailleurs, les composants programmables par logiciel ont profité de l'émergence d'un matériel complexe capable de faire apparaître le parallélisme au niveau des instructions issues de programmes séquentiels. Ainsi, une architecture à simple cœur pouvait recevoir un flot d'instructions et l'exécution était assurée par un circuit doté de plusieurs unités fonctionnelles parallèles. Au niveau hardware, une partie importante du processeur était dédiée à l'extraction dynamique du parallélisme à partir du code séquentiel. De plus, cette partie matérielle du CPU avait également pour tâche de compenser la latence de la mémoire. En général, les concepteurs écrivent leurs

programmes sans se soucier de la hiérarchie régissant la mémoire du processeur, considérant que cette mémoire est étendue, de nature unique et de vitesse d'accès uniforme. Au contraire, le processeur est soumis aux réalités physiques imposées par la forte latence et la limitation de la bande passante des connexions à de la mémoire externe. Pour assurer un apport continu vers les unités fonctionnelles, le processeur doit également pré-charger spéculativement les données de la mémoire externe sur les caches de sa puce, afin que ces données soient beaucoup plus proches de l'endroit où l'exécution s'effectue. Après plusieurs décennies d'amélioration des performances en utilisant ces techniques, les gains apportés par ce type d'architecture sont cependant de moins en moins probants.

Figure 1 : les tendances récentes des technologies programmables et parallèles.



L'impact de ces deux tendances sur l'architecture conventionnelle des processeurs étant de moins en moins bénéfique, nous constatons que ces composants programmables par logiciel évoluent de manière significative, comme illustré sur la [figure 1](#). Plutôt que de porter sur l'extraction automatique du parallélisme des instructions au moment de l'exécution, l'attention se concentre maintenant sur l'identification du parallélisme au niveau des « threads » (mailles ou petites routines d'instruction) lors du codage. Des composants multi-cœurs massivement parallèles font leur entrée sur le marché avec une tendance marquée pour des architectures

réunissant plusieurs processeurs de moindre complexité où la majorité des transistors est dédiée au calcul plutôt qu'au cache et à l'extraction du parallélisme. Ces composants vont des processeurs à cœur multiple, qui ont communément 2, 4 ou 8 cœurs, à des GPU constitués de centaines de cœurs simples mais optimisés pour le traitement parallèle des données. Afin d'atteindre les hautes performances offertes par ses composants multi-cœurs, les programmeurs doivent explicitement coder leurs applications de manière parallèle. A chaque cœur doit être assignée une tâche de façon à ce que tous les cœurs coopèrent ensemble à l'exécution d'un calcul particulier. C'est en fait exactement ce que les concepteurs de FPGA font quand ils créent leurs architectures de système haut de gamme.

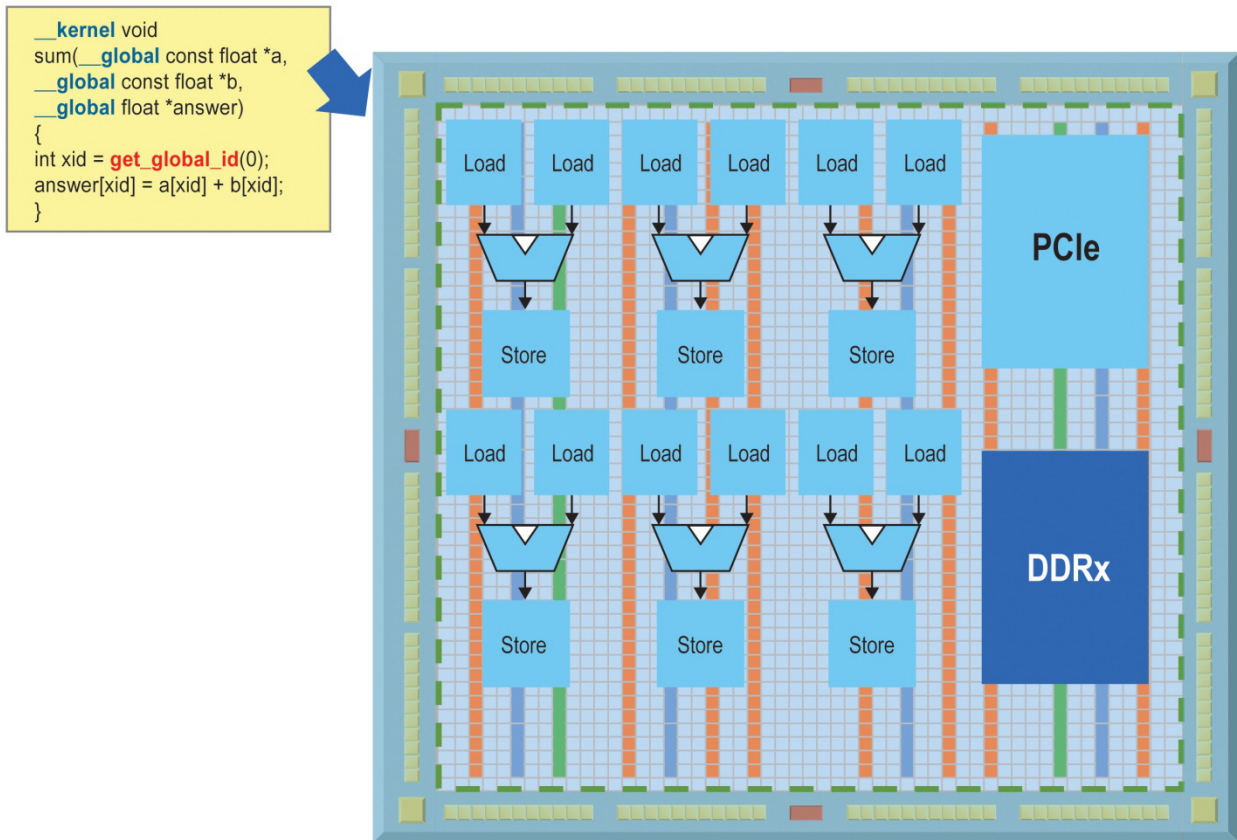
Face à cet impératif d'écrire des programmes parallèles pour la nouvelle génération du multi-cœur, le langage OpenCL (Open Computing Language) a été conçu dans le but de définir un standard de programmation parallèle quelle que soit la plate-forme ciblée. Intrinsèquement, le standard OpenCL offre la capacité de décrire des algorithmes parallèles applicables aux FPGA, à un niveau d'abstraction beaucoup plus élevé que celui des langages de description hardware (HDL) tels que VHDL ou Verilog. Bien que plusieurs outils de synthèse de haut-niveau existent pour atteindre ce plus haut degré d'abstraction, ils ont tous été confrontés au même problème fondamental. En effet, ces outils s'efforçaient de mettre en œuvre une implémentation HDL parallèle à partir d'un programme C séquentiel. La difficulté n'était pas tant la création d'une implémentation HDL parallèle, mais plutôt l'extraction du parallélisme au niveau des « threads » qui permettrait au FPGA d'atteindre une performance maximale. Du fait du parallélisme extrême de la structure FPGA, tout échec dans l'extraction optimale du parallélisme est plus invalidant que sur d'autres composants. Le standard OpenCL élimine beaucoup de ces problèmes en permettant au programmeur de spécifier et contrôler explicitement le parallélisme. Par nature même, il convient beaucoup mieux à la structure hautement parallèle des FPGA que le font les programmes séquentiels écrits en pur C.

Les applications OpenCL comportent deux parties. Le programme hôte OpenCL est un logiciel entièrement écrit en C/C++ standard qui peut s'exécuter sur n'importe quel

microprocesseur. Celui-ci peut être un processeur « soft » embarqué dans un FPGA, un processeur « hard » ARM, ou un processeur x86 externe.

Au cours de l'exécution du programme logiciel hôte, il arrive qu'une fonction exige des calculs intensifs. Elle peut alors bénéficier d'un accélérateur matériel sous la forme d'un circuit hautement parallèle : un CPU, un GPU, un FPGA, etc. On appelle alors noyau OpenCL (OpenCL kernel) cette fonction destinée à être accélérée. Ces noyaux sont écrits en C standard. Cependant, ils comprennent des annotations sous forme d'instructions spécifiant le parallélisme et la hiérarchie de la mémoire. L'exemple présenté sur la figure 2 réalise l'addition vectorielle de deux tableaux, a et b, et écrit le résultat dans un tableau de sortie. Des « threads » parallèles s'appliquent à chaque élément du vecteur. Résultat : le calcul s'effectue beaucoup plus rapidement quand il est accéléré par un composant offrant un parallélisme massif à grain fin comme un FPGA. Le programme hôte a accès aux API du standard OpenCL qui permettent le transfert des données vers le FPGA, l'invocation du noyau sur le FPGA et le retour des données traitées.

Figure 2 : exemple d'une implémentation OpenCL sur un FPGA



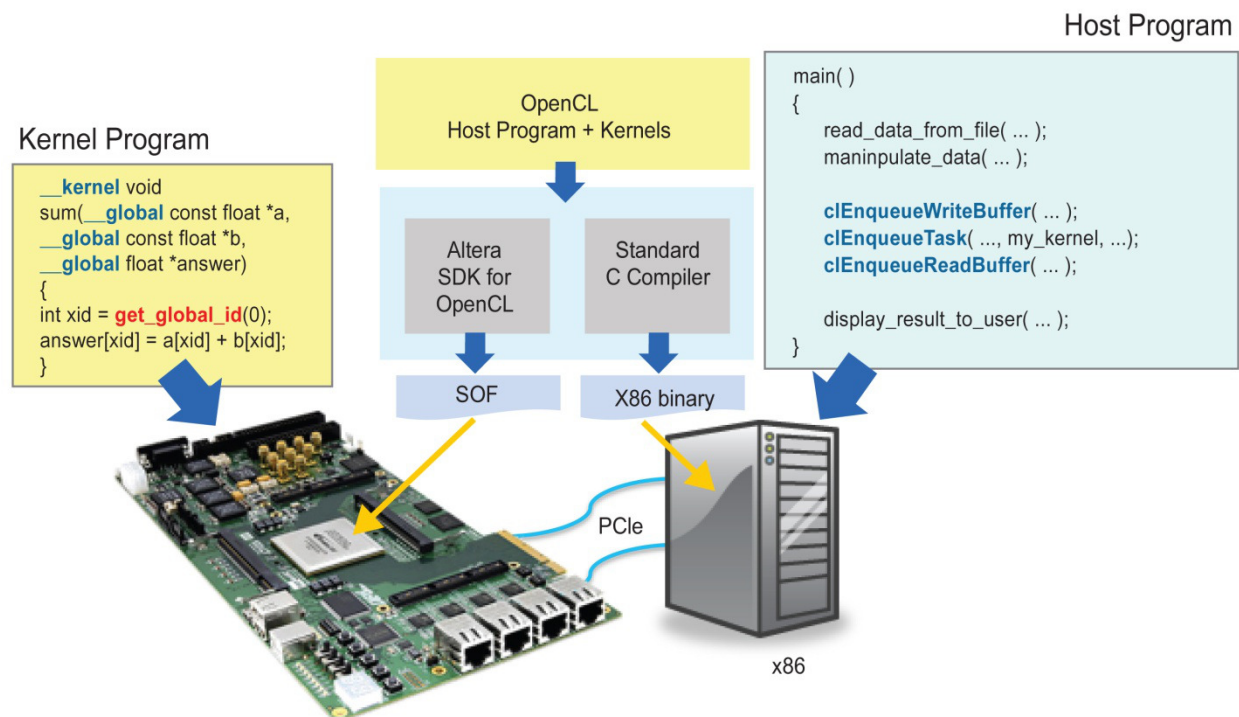
Au sein des FPGA, les fonctions noyaux sont transformées en circuits hardware dédiés, structurés en profond pipeline, dont la nature inhérente de « multi-threads » est obtenue grâce à l'exploitation du concept de parallélisme des pipelines. Chacun de ces pipelines peut être dupliqué plusieurs fois afin de fournir encore plus de parallélisme qu'avec un pipeline simple.

Les avantages de la mise en œuvre du standard OpenCL sur un FPGA

Concevoir pour des FPGA en utilisant une description basée sur le standard OpenCL offre plusieurs avantages par rapport aux méthodologies traditionnelles basées sur une conception en HDL. Le développement de composants programmables par logiciel suit typiquement les étapes suivantes : détermination d'une idée, codage de l'algorithme

dans un langage de haut-niveau tel que le C, puis utilisation d'un compilateur automatique pour obtenir le flot d'exécution. Le SDK d'Altera pour OpenCL fournit un environnement de conception idéal pour réaliser facilement des applications OpenCL implémentées sur des FPGA. Ceci est représenté sur la figure 3.

Figure 3 : une vue d'ensemble du SDK d'Altera



Cette approche peut être comparée aux méthodologies traditionnelles de conception sur FPGA qui demandent au concepteur de décrire cycle par cycle le matériel utilisé pour implémenter leur algorithme. Le déroulement classique fait intervenir la création de chemins de données, et de machines d'état pour le contrôle de ces chemins, la connexion des cœurs d'IP de bas-niveau en utilisant des outils systèmes, ainsi que la gestion des problèmes de synchronisation dus aux contraintes temporelles imposées par les interfaces externes. Le SDK d'Altera pour OpenCL effectue toutes ces étapes automatiquement pour les concepteurs, leur permettant de se concentrer sur la définition de leur algorithme plutôt que sur les détails fastidieux de la conception

hardware. En concevant ainsi, les concepteurs peuvent migrer facilement vers de nouveaux FPGA offrant de meilleures performances et des densités plus élevées. En effet, le compilateur OpenCL transformera la même description de haut niveau en des pipelines qui profiteront des avantages offerts par ces nouveaux FPGA.

L'utilisation du standard OpenCL sur FPGA peut assurer des gains significatifs en performances pour une consommation bien moindre qu'avec d'autres architectures matérielles actuelles (CPU, GPU, etc). Enfin, un système hétérogène (CPU + FPGA) utilisant le standard OpenCL permet une mise sur le marché beaucoup plus rapide comparé à un développement FPGA classique avec des langages de description hardware (HDL) tel que Verilog ou VHDL.