Conception à haut niveau pour FPGA et (r)évolution des composants programmables

LLR Ecole Polytechnique
F - 91128 PALAISEAU Cedex

Laboratoire Leprince Ringuet
LLR Polytechnique IN2P3/CNRS

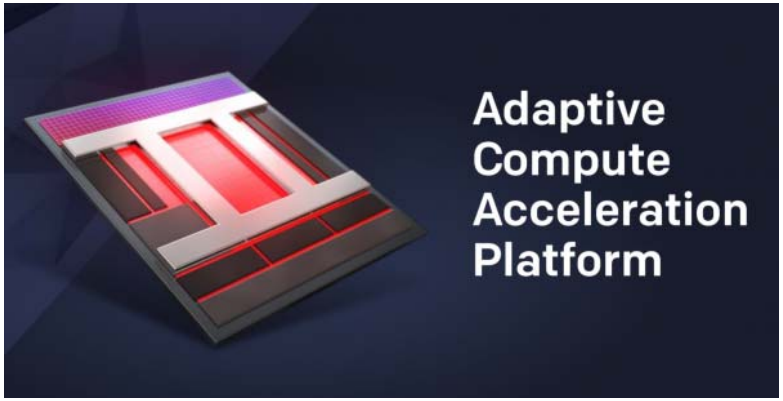
L. Pacheco, T. Romanteau,

12/11/2018

Introduction

- Les composants programmables évoluent beaucoup en capacité et complexité
- Les méthodologies de développement des systèmes s'adaptent :
  - Avec des nouveaux outils de synthèse et des flux de conception qui se standardisent
  - Pour supporter de nouvelles architectures de composants programmables

Vivado™ HLS



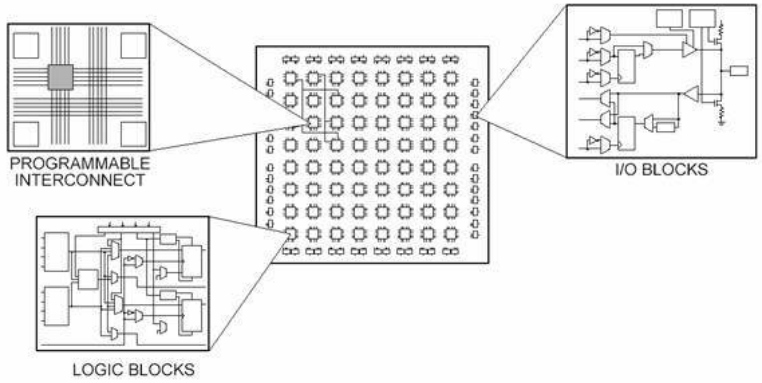
"This is a major technology disruption for the industry and our most significant engineering accomplishment since the invention of the FPGA"
– Victor Peng, president & CEO of Xilinx

Circuits programmables

- Composant électronique reconfigurable par l'utilisateur
  - Circuiterie modifiable
- Pour faire du traitement logique
  - Parallèle, haute vitesse

FPGA

- Blocs logiques:
  - Look-up table (LUT) = opérations logiques
  - Bascules = stockent les résultats des LUTs
- Interconnexion programmable. Relie les bloc logiques
- Blocs d'entré/sortie. Pour communiquer avec le monde extérieur
- D'autres blocs intégrés: RAM, Transceivers, DSP, PLL…



Conception FPGA

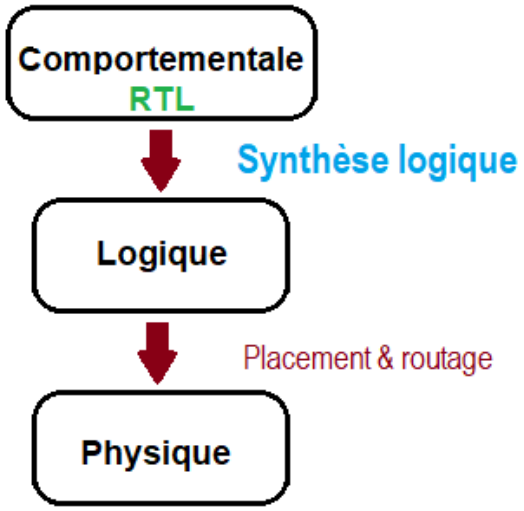
Etapes:

1. **Spécification** fonctionnelle appelé **description RTL** - *Register Transfer Level.*

   -Codage à travers les **langages de description matérielle:** VHDL ou Verilog
   -On définit: interfaces, opérations, séquencement, horloges,…

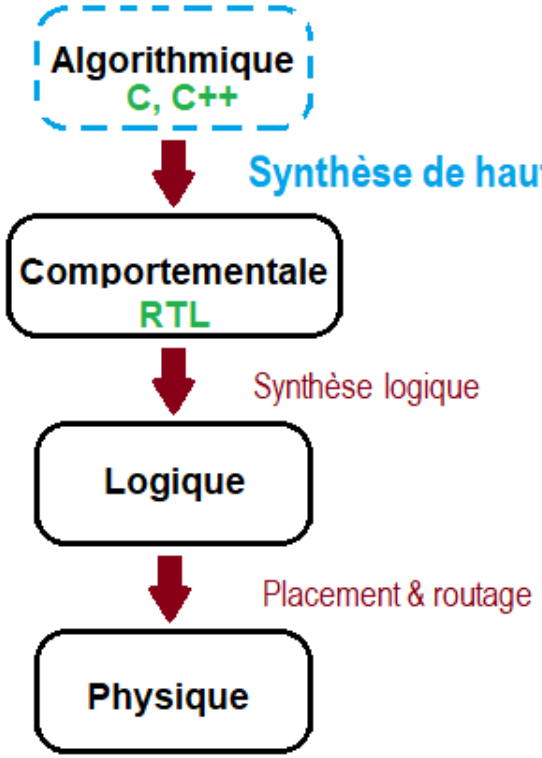
2. **Synthèse logique**. A partir de la description RTL génère automatiquement des représentations à plus bas niveau (en portes logiques)

3. **Implémentation**. Placement & routage, création du fichier à charger dans le composant



Niveaux d'abstraction de la conception FPGA/ASIC

Conception de haut niveau



Niveaux d'abstraction de la conception FPGA/ASIC

- Spécification algorithmique appelé ***description de haut niveau***
  - Codage à travers des langages de software C, C++
- **Transformation haut niveau → RTL automatisée** par des *outils de synthèse de haut niveau* comme **Vivado® HLS**, technologies assez matures pour remplacer le traditionnel codage RTL

La méthodologie de conception FPGA évolue

Depuis 2001 cette méthodologie reçoit le nom de

Electronic System Level (ESL) design

ou simplement *System Level design*

Conception de haut niveau

Pourquoi est intéressante la conception ESL?

- **On réduit les temps de conception** d'autant plus que l'algorithme à implémenter est complexe ou de grand taille.
  - ✓ Génération automatique RTL → sec/min.
  - ✓ Codage RTL à la main → heures/jours
- Les circuits programmables augmentent exponentiellement leur capacité
- **On améliore le développement** de systèmes complexes:
  - ✓ Permet de faire des analyses de performance algorithmique
  - ✓ Permet d'explorer plusieurs solutions matérielles

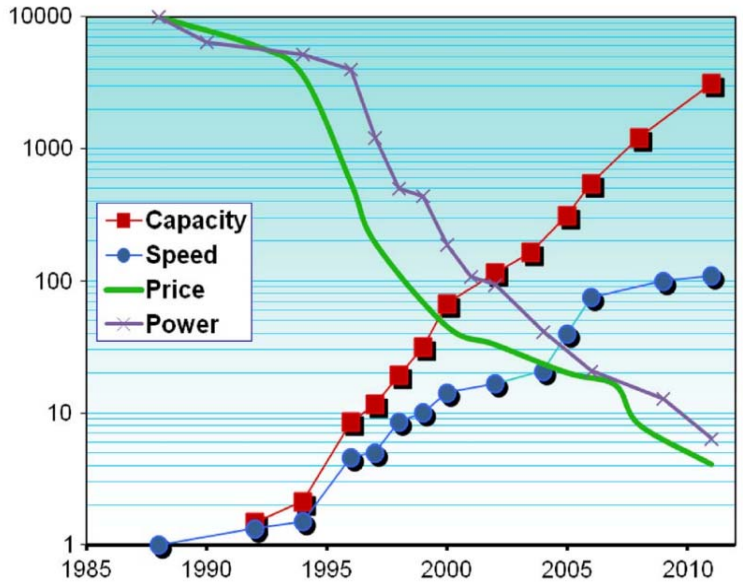
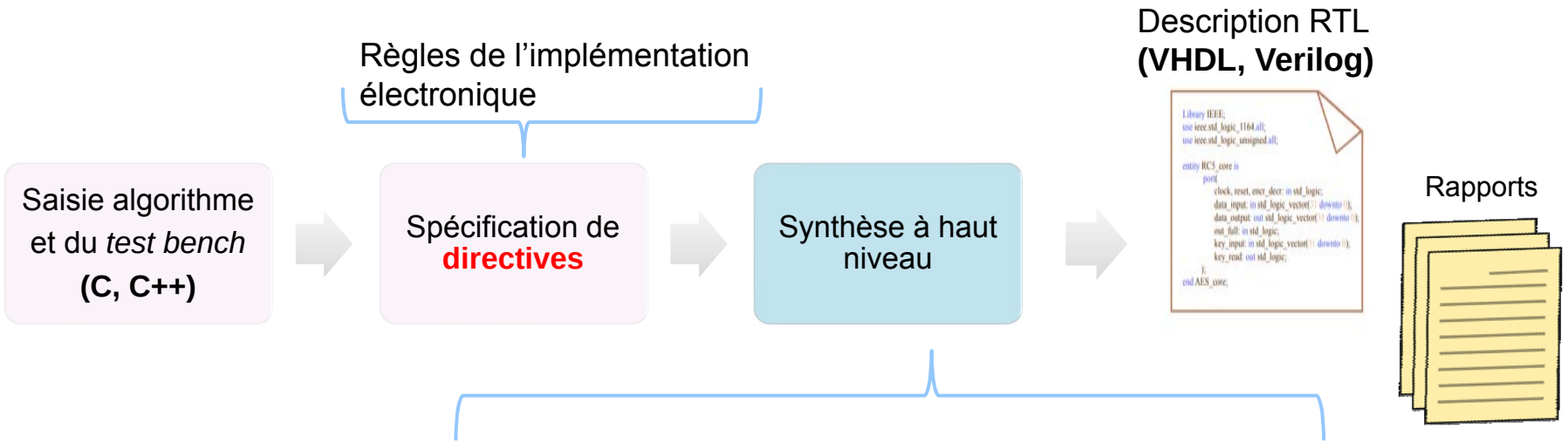


Fig. 2 Progress technique des **FPGA** Xilinx® depuis 1988. ● Nombre de cellules logiques (CLB), ● Vitesse ● Prix par CLB, ● Puissance par CLB [Réf Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology].

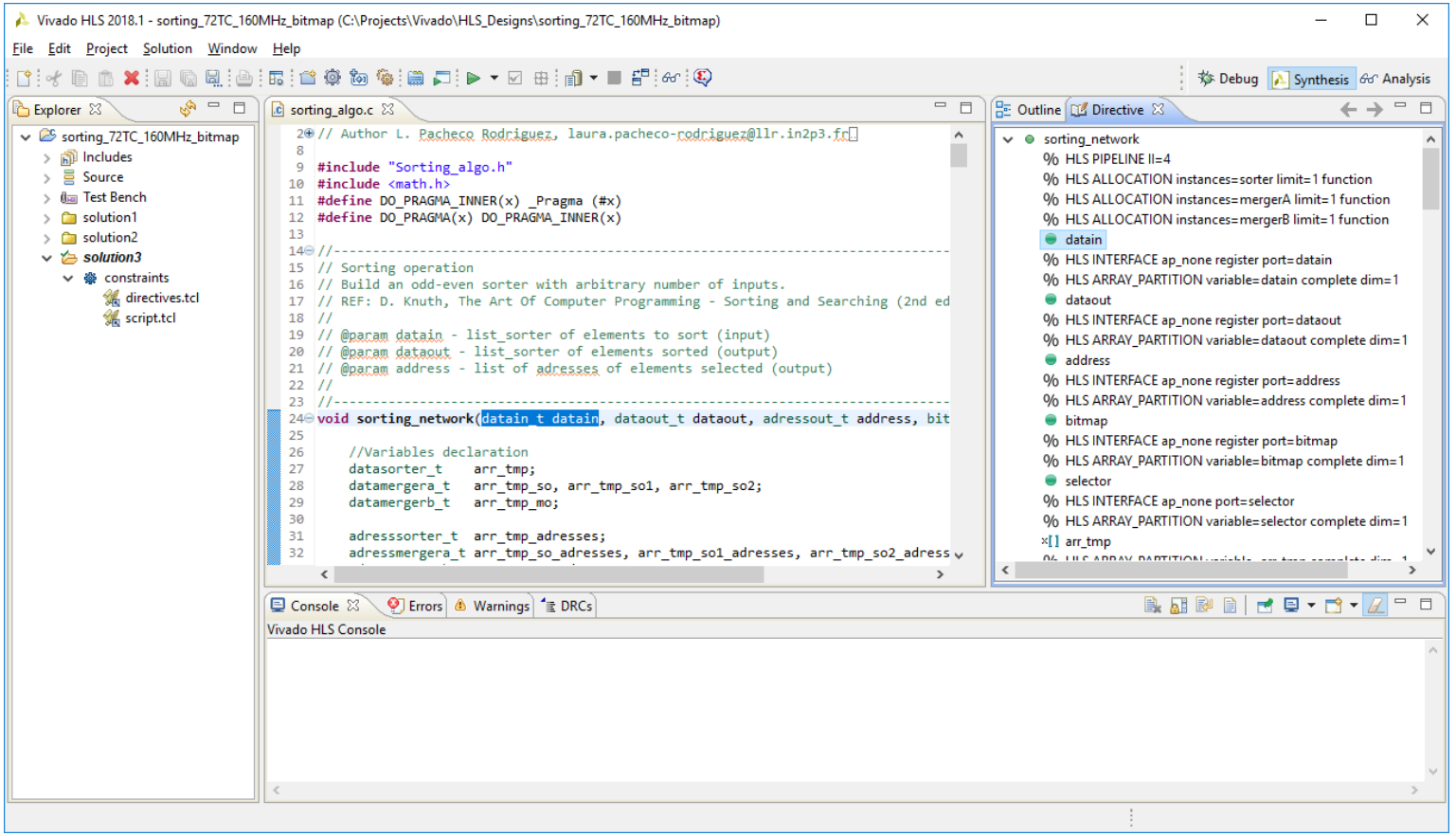
Synthèse de haut niveau



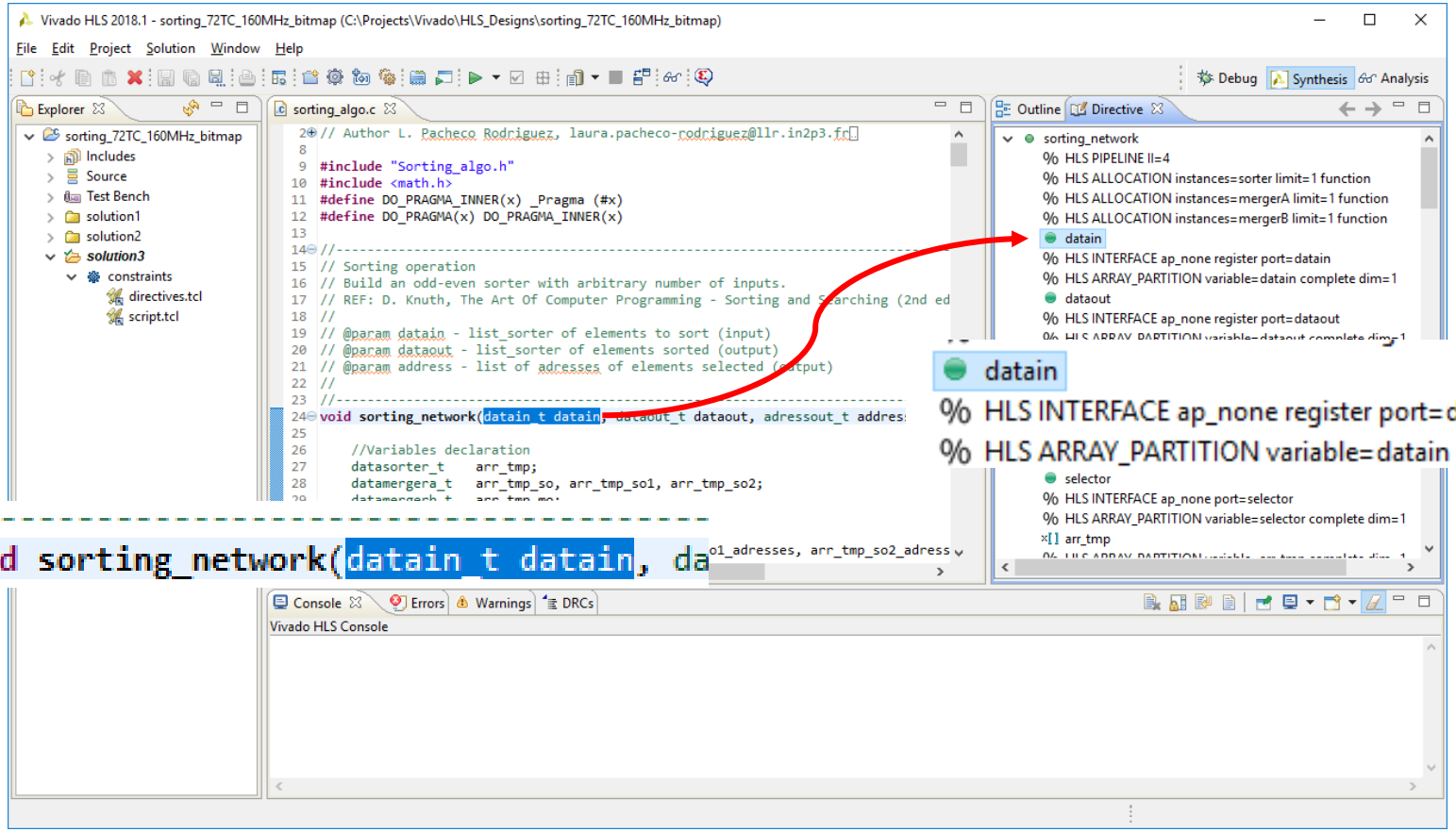
1. **Planification**
   Quelles tâches sont exécutés dans chaque cycle d'horloge
2. **Binding**
   Assignation de blocs matérielles à chaque tâche.
3. **Extraction logique de control**
   Création de la machine à états finis (FSM) pour séquencer les tâches

A partir d'une même *description de haut niveau* plusieurs *implémentations matérielles* sont possibles.
Le résultat dépend des directives!

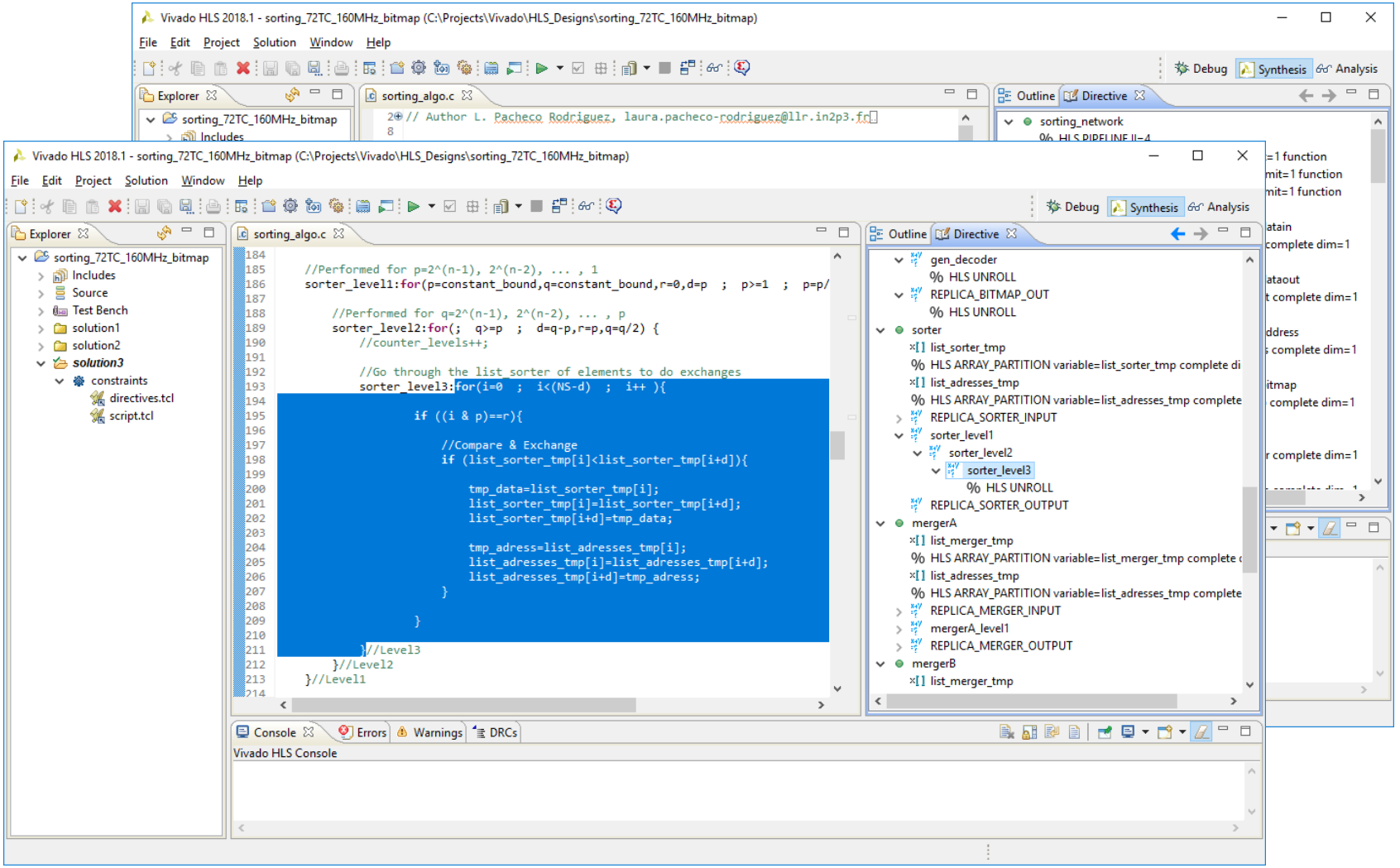
Directives sur Vivado® HLS



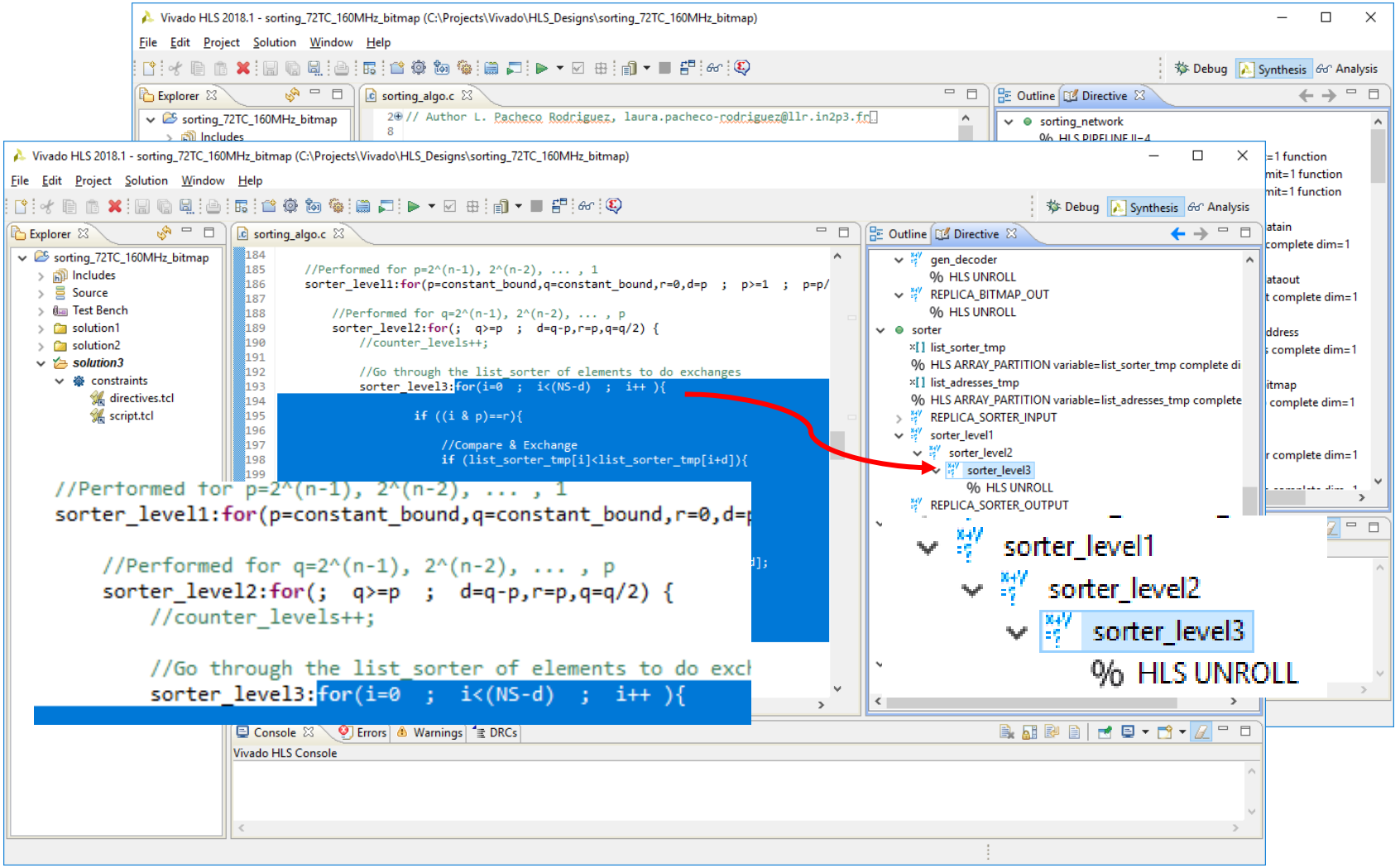
Directives sur Vivado® HLS



Directives sur Vivado® HLS



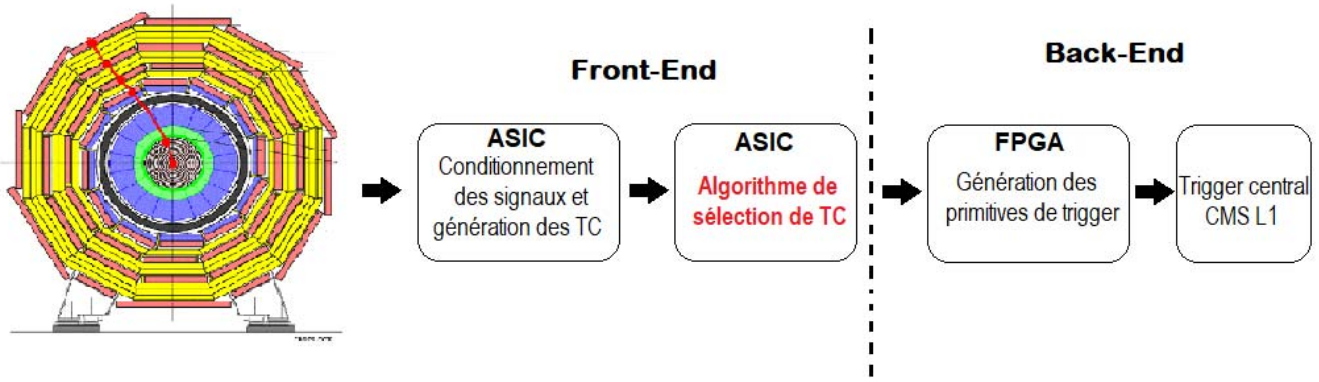
Directives sur Vivado® HLS



Exemple de conception à haut niveau

- Développé au sein du LLR
- **Objectif**. Tester des solutions matérielles répondant à la problématique de génération des primitives trigger dans l'expérience CMS

  Implémentation des algorithmes pour sélectionner les cellules trigger dans la partie *Front End* du calorimètre à haute granularité (HGCAL) du sous détecteur ECAL de CMS.



HGCAL trigger flow path

Etude préalable

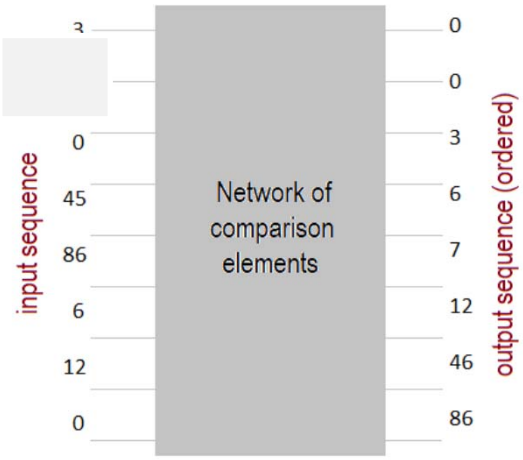
- Spécifications du système.
  - **Besoins de la physique**. Sélectionner les N cellules de trigger (ou régions d'intérêt) les plus énergétiques à chaque collision LHC, et respecter une certaine précision du codage des énergies.
  - **Contraintes électroniques**. Bande passante, vitesse de traitement, latence, interface de réception et transmission de données, capacité et puissance dissipée…

- Recherche documentaire & pré-étude de possibles solutions algorithmiques

Algorithmes de tri
- *Bitonic sorter*
- *Odd-even merge sort*

Critères de performance matérielle
- Nombre de comparateurs → *utilisation matérielle*
- Nombre de étapes de comparaison → *latence*

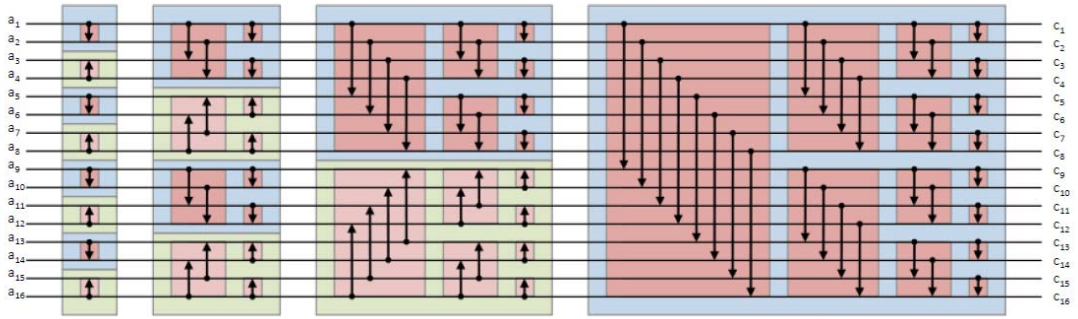
Versions adaptées à notre application
- Nombre d'entrées arbitraire ($N \neq 2^n$)
- sélection des M valeurs plus grandes (algorithme décomposable en blocs)



Estimation analytique des performances matérielles pour guider l'exploration de solutions

Etude préalable

- Spécifications du système.
  - **Besoin** chaque
  - **Contrai** transmi



us énergétiques à

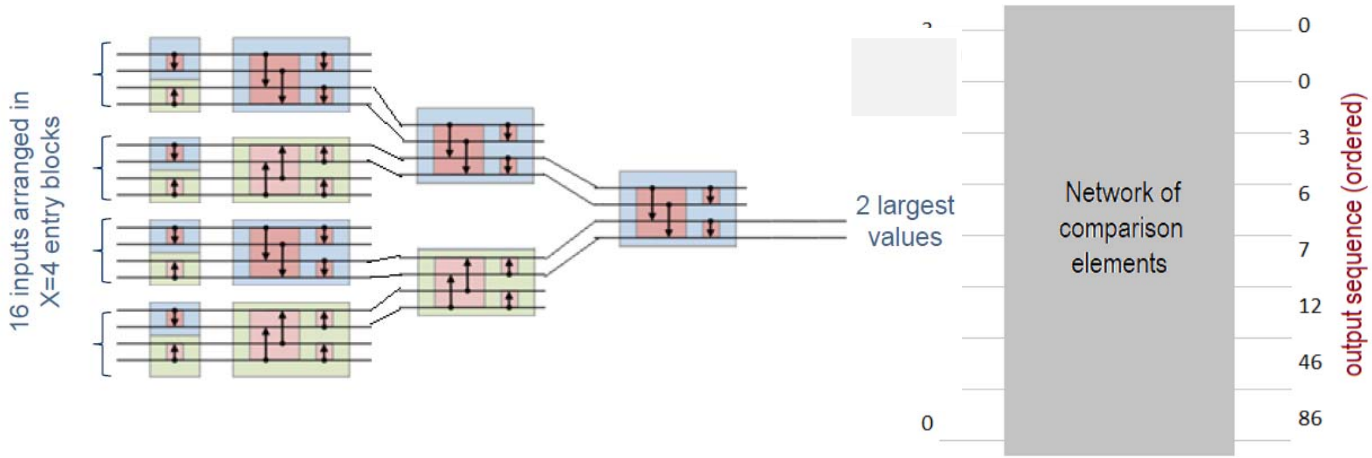
réception et

- Recherche c

Algorithmes de tri
- *Bitonic sorter*
- *Odd-even me*

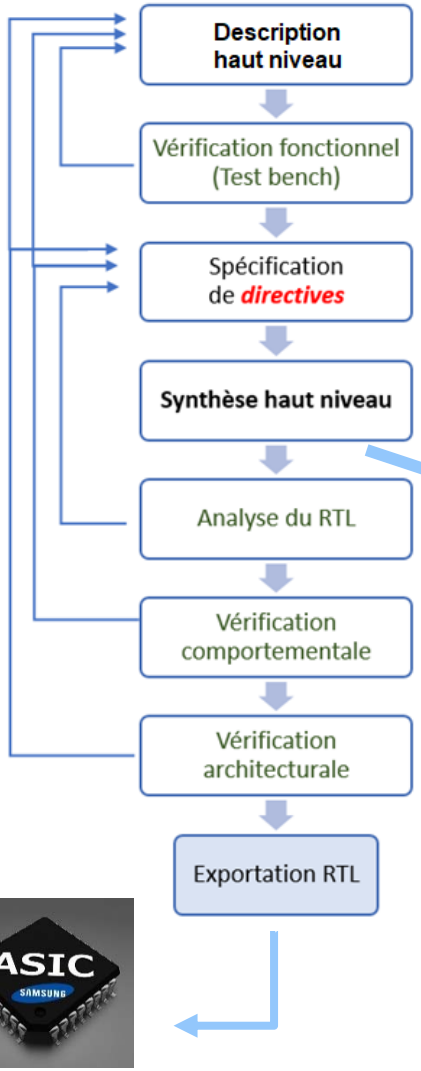
Critères de perfo
- Nombre de co
- Nombre de ét

Versions adaptée
- Nombre d'ent
- sélection des

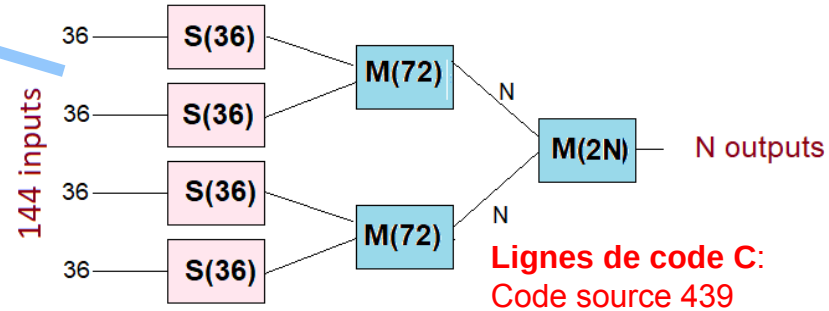


Estimation analytique des performances matérielles pour guider l'exploration de solutions

Workflow

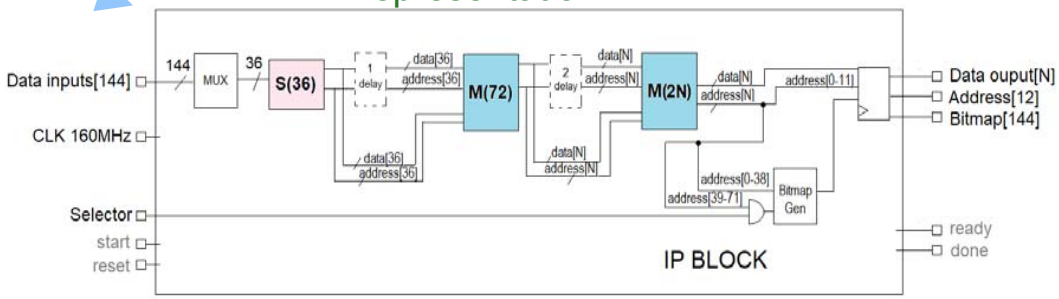


Représentation algorithmique



Lignes de code C:
Code source 439
Test bench 151

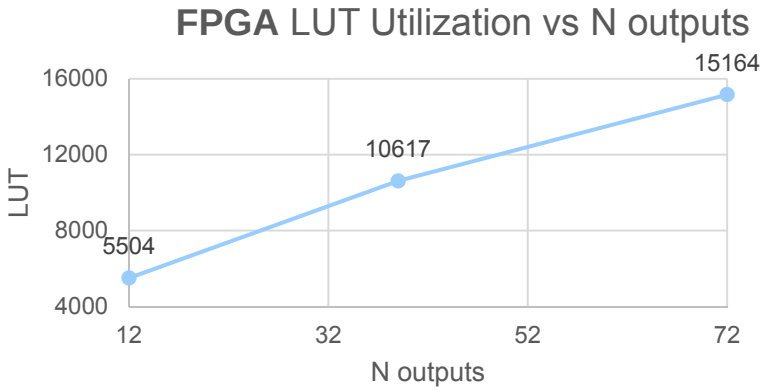
Représentation RTL



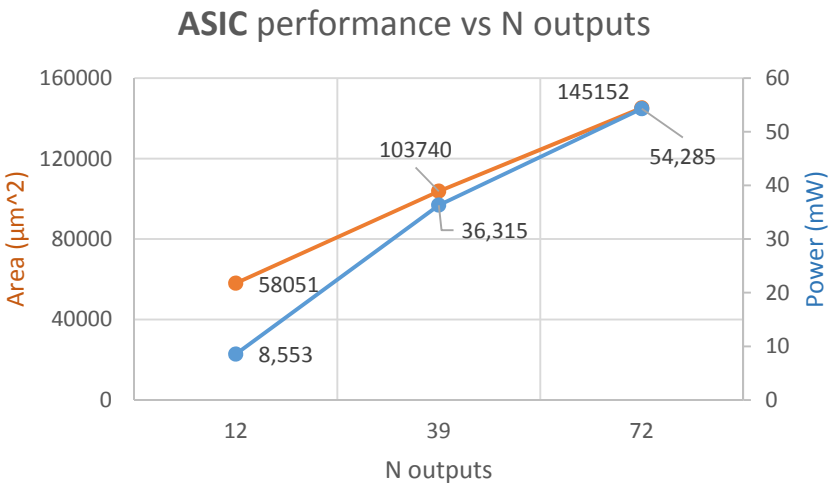
Lignes de code VHDL
18 296

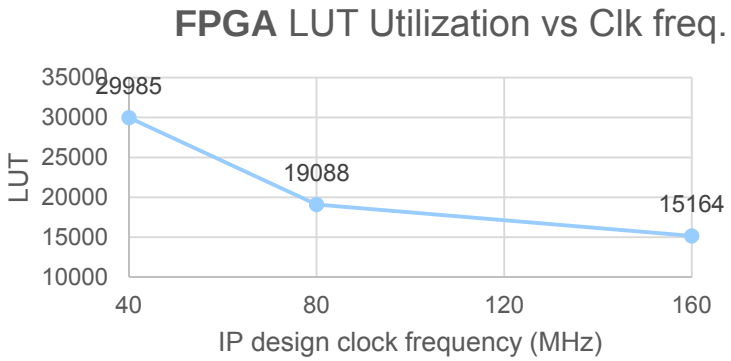
Analyse de solutions matérielles

- Synthèse logique FPGA et ASIC du RTL issue de la synthèse de haut niveau
- Performances vs:
  - # entrées
  - # sorties
  - # bits codage
  - Étapes de pipeline (latence)
  - Fréquence de traitement



~20 solutions ont été produites et analysées en 3 mois.





Conclusion Partie 1

La conception à haut niveau (ESL) :

- Permet un temps de conception plus rapide (moins de ligne de code), nécessaire pour des algorithmes plus complexes qui sont à présent implantable dans des composants programmables grâce à la forte augmentation de leur capacité
- Facilite beaucoup l'exploration de plusieurs solutions → Designs de meilleur qualité

Les 20 solutions produites par ESL dans l'exemple présenté auraient été très difficilement achevées dans un même délai par une conception traditionnel au niveau RTL

Il est néanmoins nécessaire d'avoir une vision électronique pendant l'exploration algorithmique pour pouvoir guider le développement et analyser de manière critique l'architecture produite par l'outil de synthèse

Introduction Partie 2

- Les semi-conducteurs ont beaucoup évolué depuis 30 ans:
  - Processeur CISC, RISC, MultiCore, DSP, GPU
  - Technologies aujourd'hui disponibles : 7nm, chip 3D

- La pseudo loi de Moore (2x / 1,5 yrs) n'est plus vérifiée
  - Les fréquences d'horloge stagnent, taille limite des transistors (The Wall), recours au multi-cœur

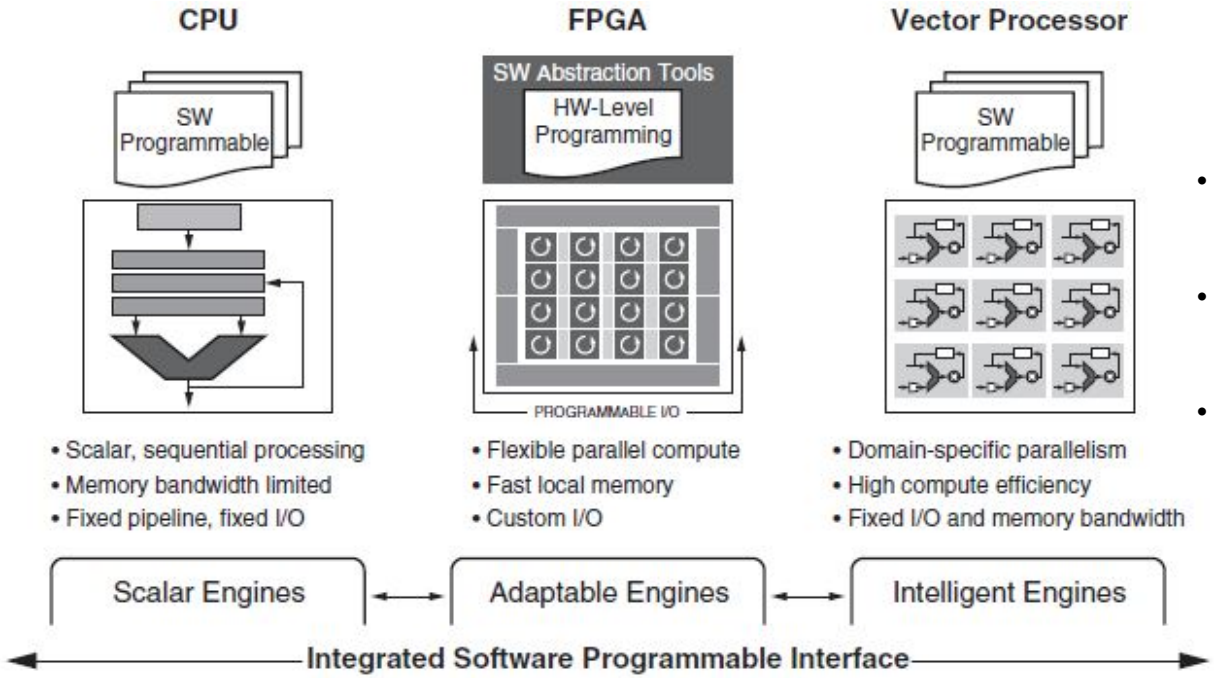
- Les évolutions en performances sont liées à de nouvelles architectures
  - Processeurs scalaires (CPU) : efficient pour des algorithmes complexes, en multi-cœur avec des bibliothèques Multithread, mais leur évolution en performance unitaire marque le pas
  - Processeur vectoriel (GPU, DSP) : efficient pour des fonctions de calcul parallélisables, pénalisé par la latence et la hiérarchisation mémoire
  - FPGA : efficient pour des fonctions spécifiques (ASIC like) temps réel et à faible latence et des structures de données complexes, néanmoins des changements algorithmiques requièrent des temps de compilation exprimés en heure versus des minutes

Et si une (r)évolution majeure de l'architecture de composants programmables permettait de bénéficier du meilleur des trois mondes !!

Xilinx a annoncé une nouvelle architecture hétérogène de calcul nommée :

« Adapative Compute Acceleration Platform » ACAP

ACAP un nouveau Paradigme



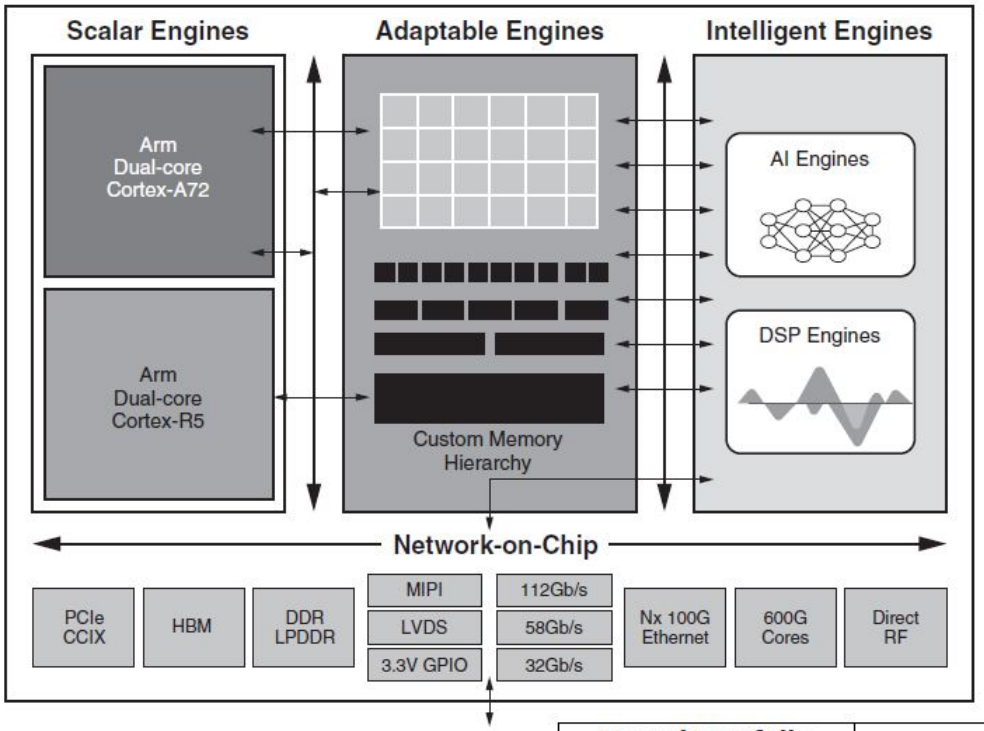
Avantages majeurs

- ***Programmable par software*** avec des chaines d'outils logiciels standard et/ou spécifiques
- ***Accélération pour des types variés d'application*** : réseau, stockage, sans fil, embarqué, IA, etc..
- ***Adaptabilité par reconfiguration dynamique partielle*** en quelques ms, depuis un interface plus abstrait que ce qui existe aujourd'hui

Architecture hétérogène composée de processeurs scalaires et vectoriel associés à une matrice programmable de nouvelle génération, le tout fortement couplé par un réseau sur silicium (NOC 1Tb/s) qui assure un accès par adressage mémoire aux trois types de d'unité de traitement.

Programmables par software via un environnement à haut niveau d'abstraction

Xilinx Versal ACAP



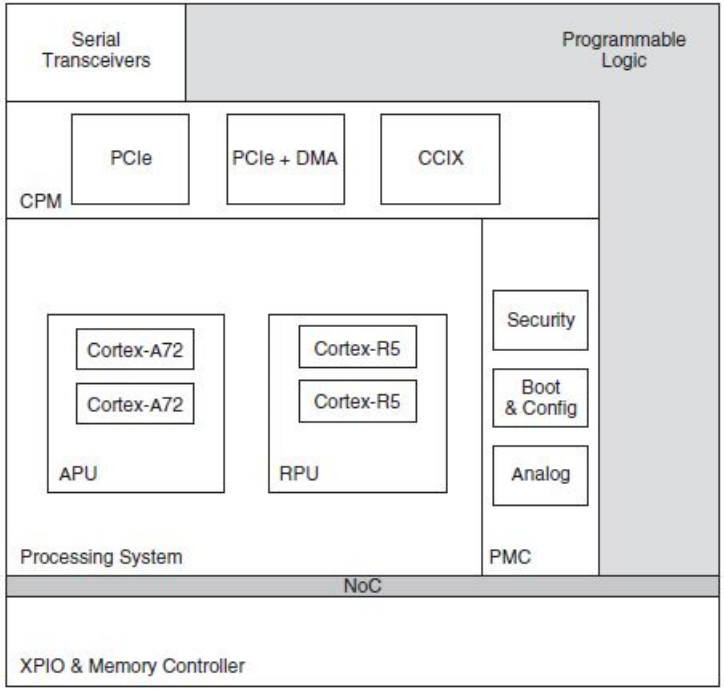
Caractéristiques des Moteurs

- *Moteurs scalaire;* cœurs ARM standard et temps réel à consommation optimisée (7nm). Perf: 15.980 DMIPs
- *Moteurs adaptable;* matrice programmable avec possibilités de créer des hiérarchies mémoires spécifiques à des applications. Perf: 2.505 DMIPs
- *Moteurs intelligent;* matrice de moteurs de calcul et mémoires tous interconnectés à quelques 100s de Tb/s.

Ces différents moteurs de calcul sont déclinés avec différents rapport de densités pour créer un portefeuille de composants Versal d'architecture ACAP.

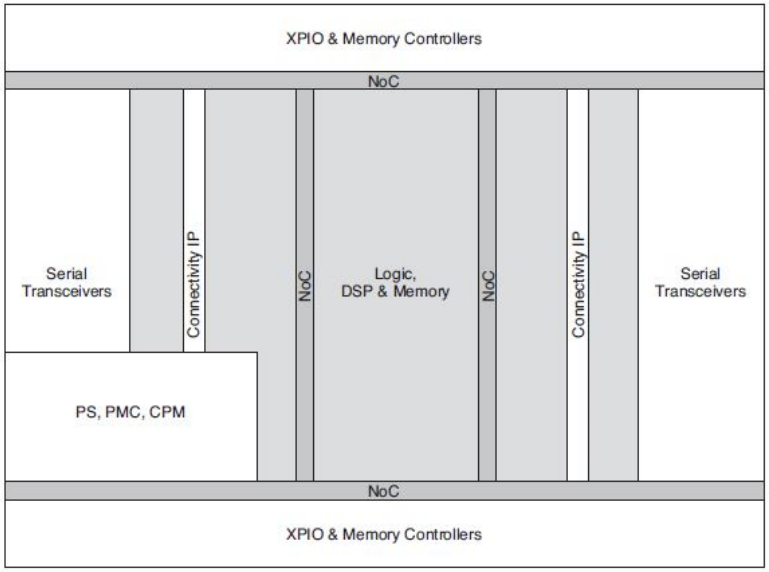
	Versal Portfolio	Primary Markets	Key Features
2Half 2019	Versal AI Core	Data Center, Wireless	Maximum intelligent engine compute
1Half 2020	Versal AI Edge	Automotive, Wireless, Broadcast, A&D	Efficient intelligent engine count within tight thermal envelopes down to 5W
2Half 2020	Versal AI RF	Wireless, A&D, Wired	Direct RF converters and SD-FEC
	Versal Prime	Data Center, Wired	Baseline platform with integrated shell
	Versal Premium	Wired, Test and Measurement	Premium platform with maximum adaptable engines, 112G SerDes and 600G Integrated IP
2Half 2021	Versal HBM	Data Center, Wired, Test and Measurement	Premium platform with HBM

Versal « Processing System »



- *Moteurs scalaire (APU);* ARM Cortex-R72 64 bits; 48KB Inst, 32KB data L1 w/ parity and ECC, 1MB L2 w/ ECC avec gestion de cohérence, FP simple et double précision. Super-scalaire 3 inst/s.
- *Moteurs temps réel (RPU);* ARM Cortex-R5 32 bits avec 32KB Inst et data L1 cache, 128KB OMC accès en un seul cycle en temps réel. Les deux RPU peuvent fonctionner indépendamment ou couplés avec unité de comparaison et dans ce cas 256KB de OMC. FP optimisé pour simple précision.
- *PCIe/CCIX:* basé sur la norme PCIeGen4, CCIX assure le maintient de cohérence de cache entre les APU, RPU et les divers autres accélérateurs du système. Supporte les modes 20 ou 25 GT/s x8.
- PCIe/DMA: interface PCIeGen4x16, peut être configuré comme le port « Root » avec accès DMA Gen4x16. De 1 à 4 Gen4x8 supplémentaire sans DMA.
- Platform Management Controller (PMC): support de mode crypté ou non (AES-GCM, SHA3-384), mémoire de boot interne, externe Flash (SD, eMMC) SPI8 bit/200MHz DDR, JTAG/SelectMap 8-32 bit/200 Mhz ou via PCIe / Ethernet pour une reconfiguration partielle. Surveillance interne de l'alimentation et de la température via un ADC 10 bit 200kSPS accessible via I2C, JTAG ou PowerManagementBus.

Versal « Programmable Logic »

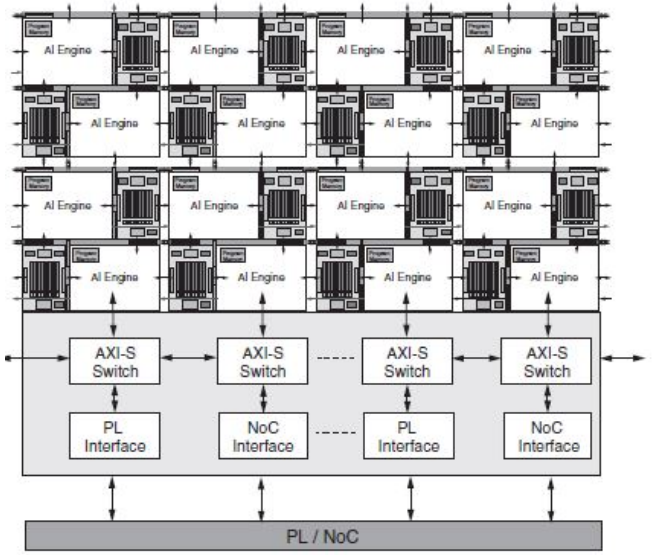


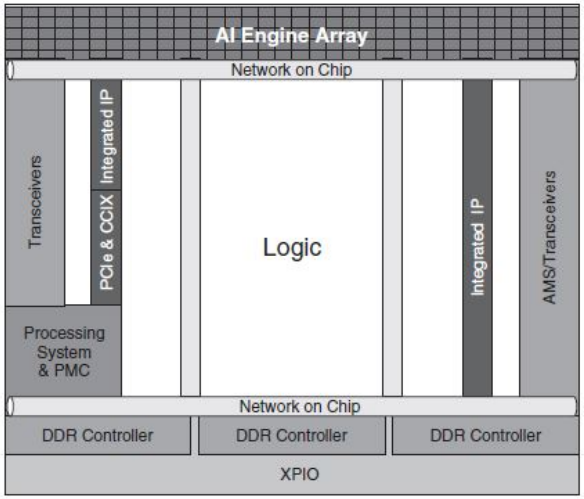
Logique programmable (PL):

- CLB contient 32 LUT de 6 entrées et 64 flips-flops
- Block RAM de 36Kb, de 4K x 9 à 512 x 72
- Ultra RAM de 288Kb, de 32K x 9 à 4K x 72
- Block DSP, multiplicateur 27 x 54 et accumulateur 58 bit
  dual 24-bit or quad12-bit add/subtract/accumulate (SIMD)
  Nouveau mode complexe 18b x 18b
  Nouveau mode flottant simple précision (FP32/FP16)
  Nouveau mode produit scalaire vectoriel tridimensionnel

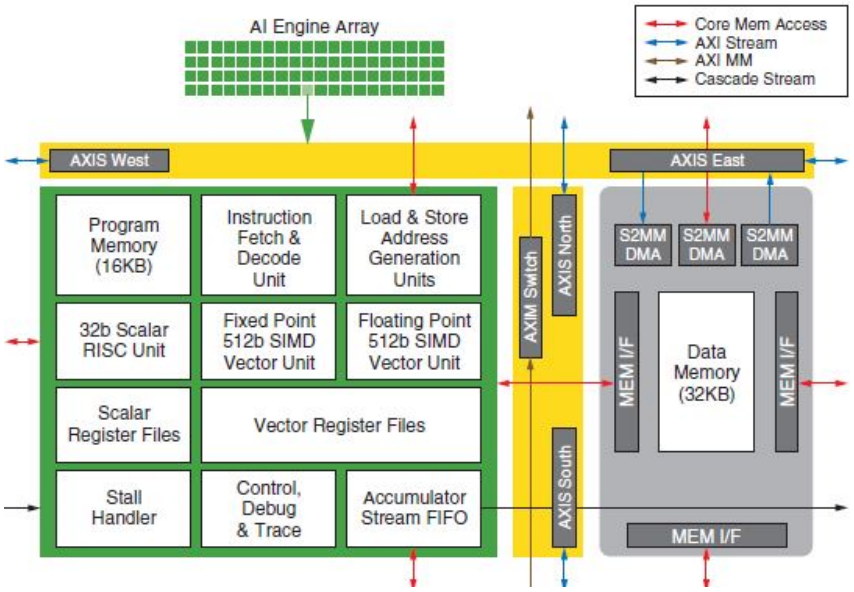
- *On Chip Memory (OMC);* 256KB OMC avec ECC pour les RPU, accessible via 1 interface AXI-128 bits, un autre interface AXI-128 est utilisé pour les APU et autres maître du bus.
- 4 MB OMC (Accelerator RAM) accédé depuis RPU via 1 x AXI-128 et depuis PL via 2 x AXI-256.
- *Connectivity IP;* USB 2.0 jusqu' 12 points, Ethernet MAC jusqu'à 1Gb/s, 2 x gigabit Ethernet controller, 2 x SPI controller, 2 I2C controller, 2 x CAN/CAN FD controller (communication automobile), 2x UART, GPIO.
- *IP dédié:* Soft-Decision Forward Error Correction (SD-FEC), utile pour la fiabilisation des transmissions série à très haute vitesse, 100Gb/s+. Supporte les codes LDPC, Polar et Turbo.
- *Serial Transceivers;* type GTY-NRZ 32,75 Gb/s, type GTM-PAM4 19–30Gb/s et 38–58Gb/s versus GTM-NRZ 9.5–15Gb/s et19–30Gb/s.
- Network on Chip (NOC): basé sur un bus « Advanced eXtensible Interface »(AXI-4, 32-1024 bits), utilisé pour la connexion à haute bande passante de donnée et à longue distance. Ex: PS / DDR, PS / PL, accès mémoire vers processeurs vectoriel, etc..
- *XPIO:* I/O optimisé pour les communications de hautes performances incluant l'interface aux mémoires externes DDR4, LPDDR4. Configuré en banques de 54 I/O de 9 x 6 bits.

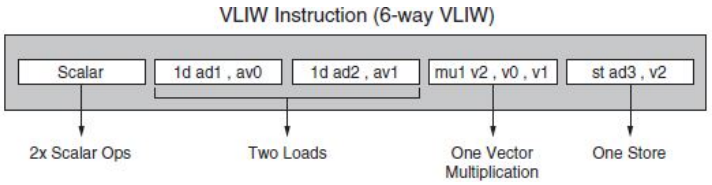
Versal « AI Engine »





- Arrangé en tableau à 2 dimensions constitué de moteur d'AI avec interfaçage nécessaire pour la connexion aux autres ressources PL, PS et NOC.
- De 128 à 400 « AI Engine » par composant Versal de la famille « AI core »





Moteur d'IA (AI Engine):

- Processeur scalaire RISC 32 bits
- Processeurs vectoriel flottant et fixe 512 bits avec registres vectoriels associés
- Mémoire (statique)16kB inst. (1k x 128) VLIW
- Mémoire (statique) 32kB data, multi accès
- Gestionnaires de synchronisation, de trace et debug

ACAP Plateforme Accélératrice

Cartes Accélératrice

- Famille de carte ***ALVEO*** double slot interfacé PCIe Gen3x16, version serveur ou station
- Basé sur FPGA UltraScale+ aujourd'hui, planifié pour supporter des composants Versal ACAP en 2019
- Supporté par le logiciel SDAccel version Alveo
- Coût licence SDAccel minimal (100$)

Alveo s'appuie sur un écosystème de partenaires et de fabricants OEM qui ont développés et qualifiés des applications clés, en IA / ML, transcodage vidéo, analyse de données, modélisation des risques financiers, sécurité et génomique. Quatorze partenaires ont développés des applications pour un déploiement immédiat sur Alveo.

Les cartes accélératrices Alveo, sont qualifiées sur des serveurs de Dell EMC, Fujitsu Limited et IBM, et bientôt d'autres.

Développement Software

Détails fonctionnels

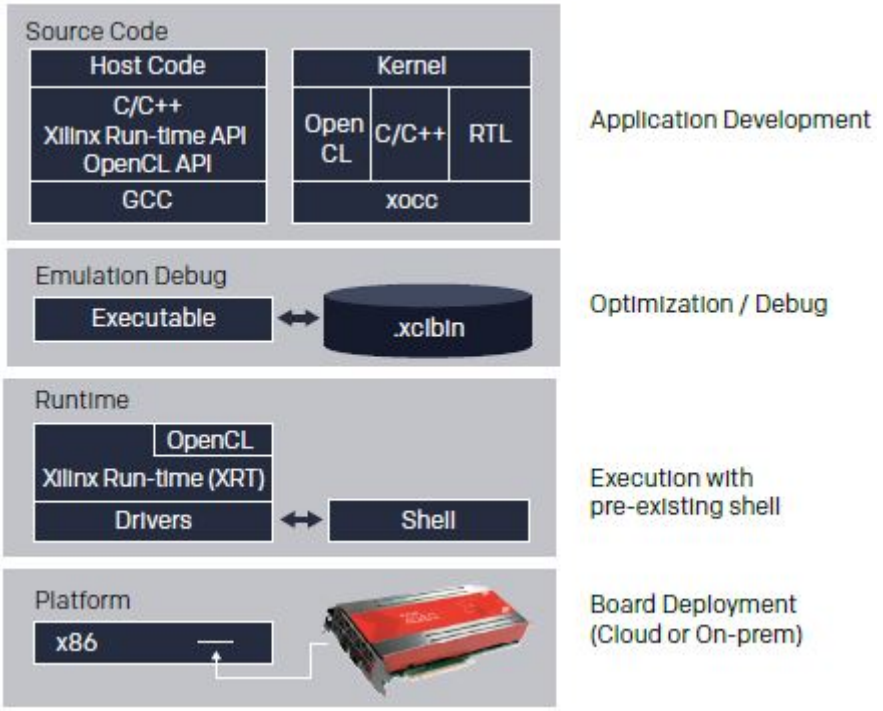
- L'application hôte est développée en C/C++ et utilise les appels standard de l'API OpenCL pour interagir avec la carte accélératrice Alveo et les fonctions accélérées. Les appels d'API d'exécution Xilinx (XRT) peuvent également contrôler des fonctions matérielles de bas niveau.

- Les fonctions de carte accélérée Alveo, également appelées « kernel », peuvent être modélisées au format RTL, C/C++ ou OpenCL Ceci fournit des points d'entrée aux concepteurs de matériel et aux ingénieurs logiciels. Le compilateur XOCC permet de relier plusieurs « kernel », multi-langage, pour créer des applications hautes performances.

- Les « kernel » personnalisés s'exécutent dans un FPGA Xilinx® via le moteur d'exécution SDAccel qui gère les interactions entre l'application hôte et l'accélérateur par le bus PCIe.

- Génération de rapports et de données de profilage lors de la compilation et de l'exécution. Outils de visualisation dédiés, le calendrier d'application et le rapport d'orientation fournissent des informations précieuses sur les goulots d'étranglement des performances et des commentaires exploitables sur l'optimisation.

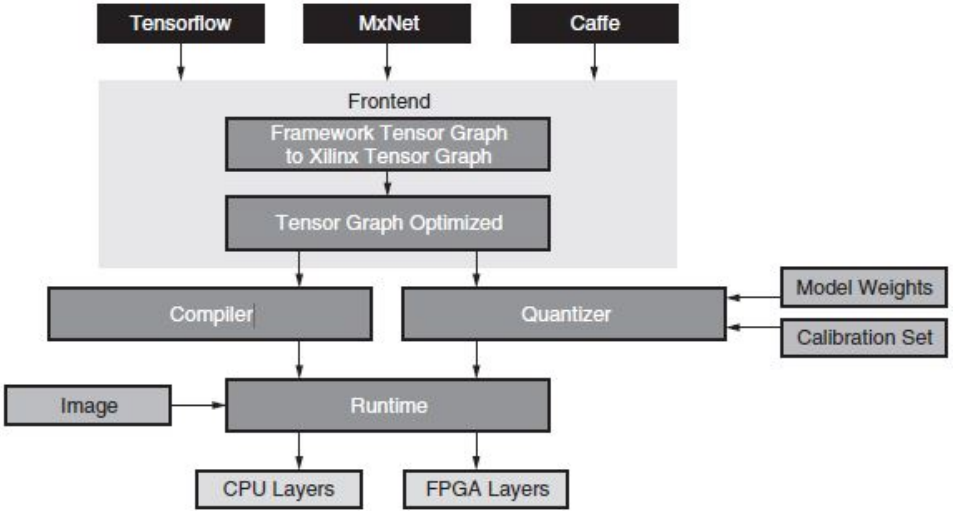
- SDAccel permet le débogage de type logiciel pour l'application hôte et le code du « kernel ». Les développeurs peuvent utiliser le flux de débogage matériel de ChipScope pour suivre l'activité.



L'outil SDAccel est l'environnement de développement intégré (IDE) pour les applications destinées à être exécutées sur des cartes accélératrices Xilinx Alveo Data Center.

L'application hôte est construite à l'aide du compilateur GCC standard et le binaire à l'aide du compilateur Xilinx XOCC.

Flux de Développement DNN



xfDNN comprends:

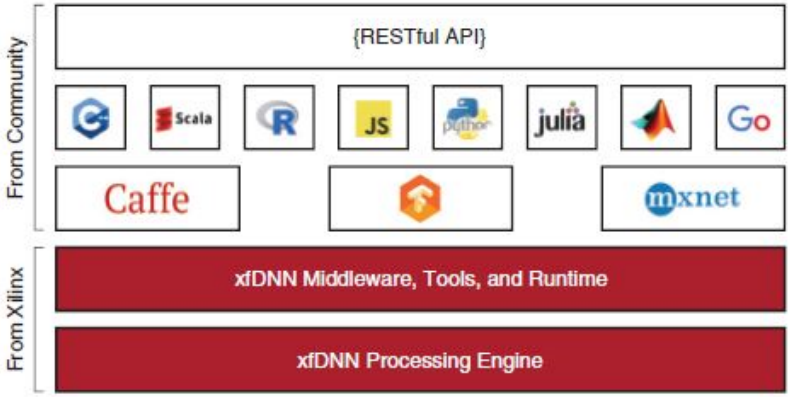
- *Compilateur réseau et optimiseur:* consiste à analyser les réseaux CNN et à produire un ensemble optimisé d'instructions à exécuter sur xDNN. L'optimisation est réalisé par fusion des couches et en gérant les dépendances mémoire du réseau.

- *Quantificateur:* il produit une quantification cible (INT8 ou INT16) à partir d'un modèle de réseau CNN entrainé sans nécessiter des heures de réapprentissage ou un jeu de données étiqueté.

- *Runtime et planificateur:* xfDNN simplifie la communication et la programmation du moteur de traitement xDNN et utilise un environnement d'exécution et une plate-forme compatibles SDx.

Xilinx propose pour ces cartes Alveo, le moteur de traitement « Xilinx Deep Neural Network » (xDNN) qui fournit une accélération DNN hautes performances et faible latence.
xDNN a été développés pour exécuter de manière générique des réseaux de neurones convolutifs (CNN) tels que ResNet50, GoogLeNet v1, Inception v4 et aussi les CNN contenant des couches personnalisées.

La pile logicielle xfDNN est une combinaison d'outils logiciels et d'API qui permettent une intégration et un contrôle transparents du moteur de traitement xDNN avec une variété de frameworks ML courants.
Une fois le réseau et les modèles compilés et quantifiés, l'utilisateur peut s'interfacer avec le moteur de traitement xDNN via une sélection d'API Python ou C++ simples à utiliser.



Expériences et perceptives dans la communauté HEP

- Le développement de réseaux neuronaux et/ou l'accélération d'application sur GPU / FPGA est concrète au sein de notre communauté
  - Analyse offline sur GPU, TTH / élément de matrice déployé au CC-Lyon (développement LLR)
  - Présentations au CERN de développements électroniques basés sur des réseaux de neurones
  - Test accélération sur plateforme FPGA avec SDAccel, L1-PF sur CMSSW présenté au CERN
  - Test d'utilisation de hls4ml avec SDAccel

- Le recourt aux outils de développement dédiés via le Cloud est effective, alternatives?
  - AWS utilisé dans les labos aux USA
  - Machine ACP pour ML utilisé par différent labos de Saclay (LLR, IPNO, LAL)

- Pourra t'on donner accès à des ressources informatiques lourdement configurées pour le Machine Learning (ML), via le Cloud? Peu envisageable dans un contexte CNRS
  - Les codes développés deviennent libre d'accès (confidentialité)
  - Cloud surtout hébergé aux USA

- *L'intérêt pour le DNN / ML est manifeste, débute pour l'accélération. Le domaine d'applications pour cette dernière technologie est prometteur (CERN)*

Conclusion Partie 2

- L'industrie électronique propose aujourd'hui une (r)évolution majeure dans les composants programmables, basée sur une architecture originale mais complexe qui tire partie du meilleur de ce que les standards technologiques offrent:
  - Processeur RISC, MultiCore, DSP, AI, FPGA sur le même chip
  - Liaisons séries hautes vitesse, IP diverses, mémoire on chip
  - Bus interne à haut débit avec maintien de cohérence de cache
  - Technologie 7nm, chip 3D (technologies clés pour ces nouveau chip)

- L'électronique permet d'accélérer l'exécution de tâches software et de se reconfigurer
  - Les premières cartes accélératrices FPGA pour data center existent à présent
  - Un écosystème applicatif est disponible, avec un système de développement basé sur le Cloud

- Seule une méthodologie de conception à haut niveau permet de prendre en compte un niveau de complexité architecturale élevé. Les outils s'interfacent aux framework de développement DNN en forte croissance et peut-être applicable à nos problématiques
  - Le développement est à présent « full software » et bien plus abstrait en terme hardware

Notre communauté a toujours naturellement veiller à ce que la technologie peut nous permettre de faire ce que nous espérions un jour faire!

Un pas important est sûrement franchi dans ce sens et Xilinx peut nous y aider aussi grâce à l'architecture **« Adapative Compute Acceleration Platform » ACAP**

BACK UP

Scheduling report