NVIDIA Tegra X1

CPU - ARM big.LITTLE

Bascunan - Di Terlizzi - Esmanhotto - Rios

SOMMAIRE

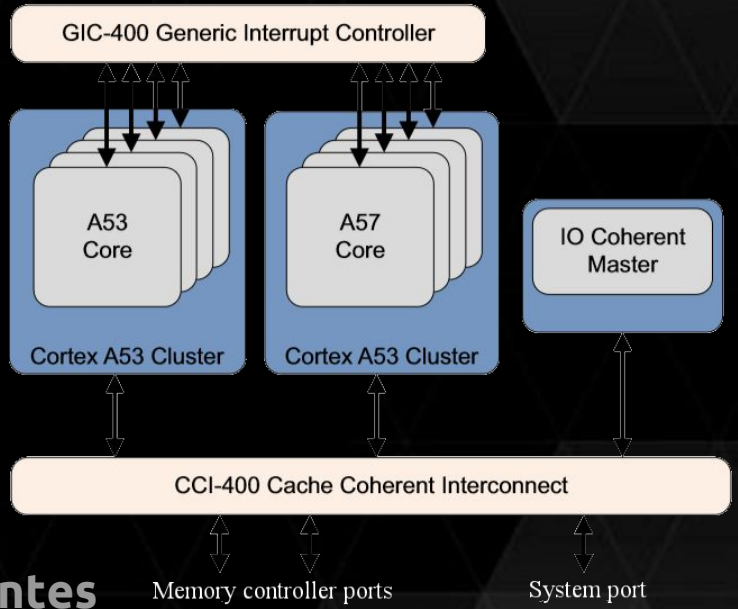
Technologie ARM big.LITTLE

Technologie ARM big.LITTLE - Présentation

- Architecture de Calcul Hétérogène (multi-cœur) développée par **ARM**
  - "**big**" de haut performance
  - "**LITTLE**" de faible puissance

- Les noyaux doivent être compatibles avec l'architecture

- Les noyaux peuvent être :
  - De 2 architectures différentes
  - **De la même architecture mais avec différentes**
    - Fréquences d'operations
    - Tailles de cache

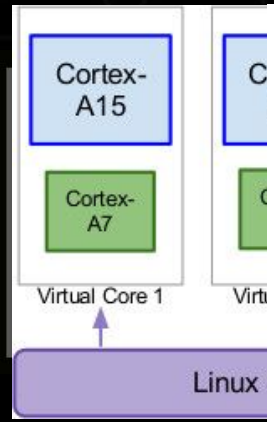


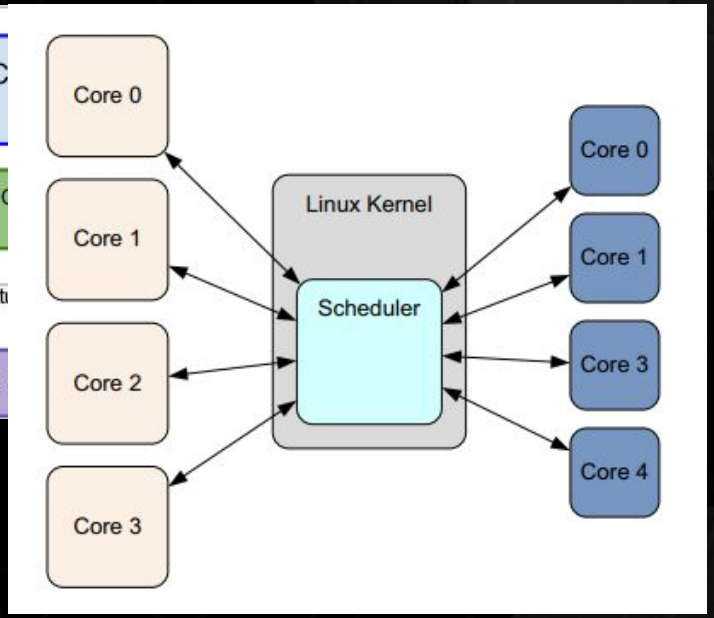
Technologie ARM big.LITTLE - Avantages

- L'équilibre optimal entre l'économie d'énergie et la performance
- Performances maximales uniquement lorsque cela est nécessaire
- Les noyaux optimisés en puissance fonctionnent la plupart du temps

Technologie ARM big.LITTLE - Cluster Scheduler

- Clustered switching
  - Un groupe de gros cœurs et un groupe de petits.
  - Le système d'exploitation ne peut utiliser qu'un cluster à la fois.





- In-kernel switching (CPU migration)
  - Les petits et grands noyaux sont divisés en paires.
  - Un seul noyau dans une paire peut être actif.
- ARM Global Task Scheduling (GTS)
  - Tous les cœurs peuvent être utilisés simultanément.

Technologie ARM big.LITTLE - Cohérence

- L'architecture des noyaux "**big**" et "**LITTLE**" sont complètement cohérent.
  - **Même jeux d'instruction**
  - **Cohérence de cache**
    - Tous les noyaux et bus ont la même vue de la mémoire cache.
    - Snooping: Bus commun, ce qui permet à chaque processeur de surveiller les accès à la mémoire des autres.
    - La cohérence entre les clusters est assurée par une cache-coherent bus telle que l'ARM CoreLink CCI-400.

Technologie ARM big.LITTLE - Cache L1 et L2

La gestion en niveaux de la mémoire cache :

- ***Niveau 1 (L1)*** : extrêmement rapide, relativement petite;
- ***Niveau 2 (L2)*** : plus grande capacité que la L1;

Configurations de la mémoire cache :

- ***Correspondance directe*** : Chaque ligne de la mémoire principale ne peut être enregistrée qu'à une seule adresse de la mémoire cache
- ***Correspondance pleinement associative*** : Chaque ligne de la mémoire principale peut être écrite à n'importe quelle adresse de la mémoire cache
- ***Correspondance associative par ensemble*** : chaque bloc est mis en correspondance avec un sous-ensemble d'adresses du cache

Architecture ARMv8-A

Architecture ARMv8-A - Comparaison ARM

Famille	Mot-cle	Caracteristique
A	Architecture	Efficacite
R	Temps reel	Fiabilite
M	Microcontroleur	Prix

Tableau comparatif des familles de processeurs ARM.

Architecture ARMv8-A - Nouveautés de l'ARM V8

Premiere version à exécuter des instructions à 64 bits

Grande adresse physique - Permet d'accéder plus que 4 GO de mémoire interne

Arch64 : Registres à 64 bits - 31 registres qui augmentent la performance et diminuent l'utilisation de la pile

NEON+SVE

NEON paquet d'operations du type SIMD (Single Instruction, Multiple Data)

SVE - Scalable Vector Extension - Complement des instructions NEON

Architecture ARMv8-A - AArch64

- Compatibilité ARMv7 avec l'architecture AArch32 et les jeux d'instructions A32, T32
  => Mode AArch32 et mode AArch64
  - 31 registres polyvalents 64 bits X0-X30, dont les moitiés inférieures sont accessibles sous forme de W0-W30 (X => 64 bits, W => 32 bits)
  - Quatre registres de pointeurs de pile SP_EL0, SP_EL1, SP_EL2, SP_EL3
  - Trois registres de liens d'exception ELR_EL1, ELR_EL2, ELR_EL3
  - Trois registres de statut de programme sauvegardés SPSR_EL1, SPSR_EL2, SPSR_EL3
  - Un compteur de programmes

- Nouveau jeu d'instructions A64 sur l'architecture AArch64 (64 bits)

```
ADD W0, W1, W2 // add 32-bit register
ADD X0, X1, X2 // add 64-bit register
ADD X0, X1, W2, SXTW // add 64-bit extending register
ADD X0, X1, #42 // add 64-bit immediate
```

Architecture ARMv8-A - SVE

Scalable Vector Extension : instructions vectorielles permettant d'effectuer une même opération sur plusieurs données à la fois

- Avantages : VLA (Vector Length Agnostic)
  - jeu d'instruction indépendant de la taille des vecteurs
  - VLA indique directement quelles instructions appliquer aux vecteurs sans s'occuper d'un quelconque découpage
  - Permet de traiter 4 données de 32 bits simultanément => instruction 128 bits

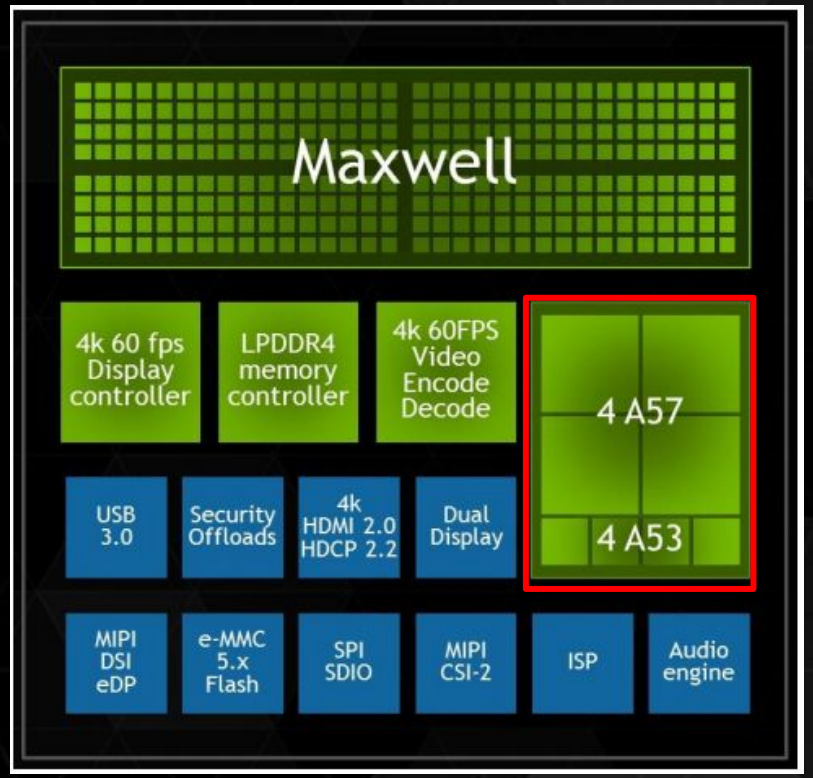
    Instruction entre 128 et 2048 bits sans recompilation ou réécriture
- Inconvénient : Fonctionnement assez flou (découpage)
  - ARM indique simplement que l'encodage de la taille du vecteur n'est pas nécessaire et qu'elle est déterminée par les mécanismes de prédiction des puces

Architecture ARMv8-A - A57 et A53

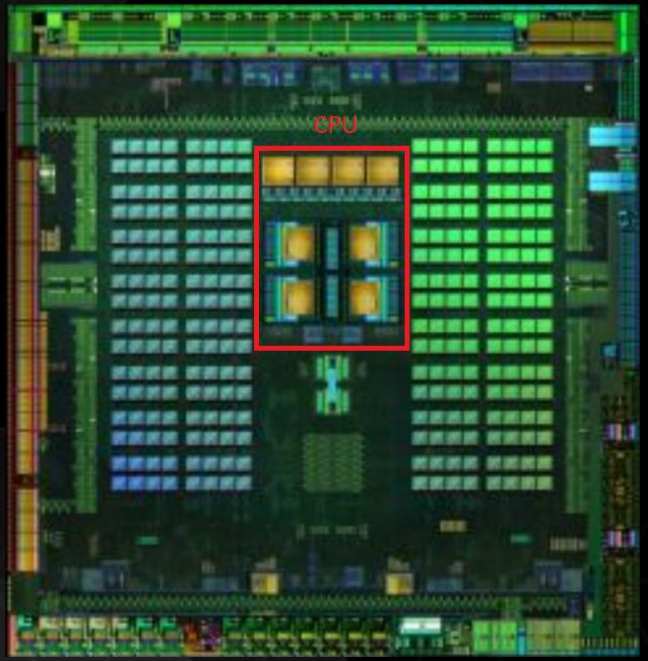
	Processeurs	
	A53 [2014, Juillet]	A57 [2015, Janvier]
Clock	2 GHz on 28 nm	1,5 to 2,5 GHz on 20 nm
Execution	In-order	Out of order
Integer Peak throughput	2,3 MIPS/MHz	4,1 to 4,76 MIPS/MHz
Pipeline Stages	8	15

Tegra X1 - Overview

GRAPHICS	Maxwell GeForce - World's Fastest GPU *2 x SMM units, DX-next, OpenGL 4.4*
CPU	Octo-Core 64b ARM v8 CPU Complex *4xCA57 Atlas/2MB L2; 4xCA53 Apollo/512KB L2*
MEMORY	64b / Dual-Quad Channel Memory Interface *LPDDR4-3200, LPDDR3E-1866, DDR3L-1866*
VIDEO	4K x 2K Encode and Decode *H.264, H.265, VP8, VP9 (dec-only)*
POWER	Low Power *20nm, HW Offloads, Isolated Pwr Rails, PRISM*
DISPLAY	4K x 2K 24b @60Hz, 1080p @120Hz *DSI 2x4, eDP, High Speed HDMI 2.0, DP*
IMAGING	Full Quad Camera imaging, Dual ISP 650Mp/s *Maxwell 16fp imaging GPGPU, HW AO-HDR*
Mobile I/O	Designed for mobile *e.MMC5.x, USB3.0/2.0/HSIC, SD/SDIO 3.0, CSI-2*

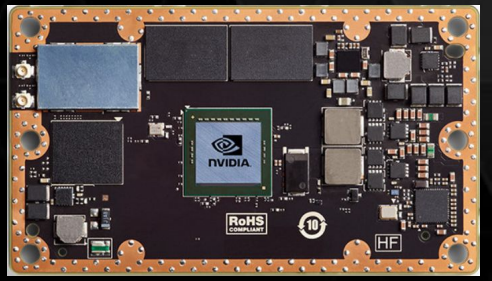


Tegra X1 - CPU



- 4 performance A57 “big” cores
  - 2MB L2 cache
  - 48KB L1 instruction cache
  - 32KB L1 data cache
- 4 high efficiency A53 “little” cores
  - 512KB L2 cache
  - 32KB L1 instruction cache
  - 32KB L1 data cache
- Les caches sont à **correspondance directe**
- Custom NVIDIA cluster switching
- Débit de la mémoire LPDDR4:
  2*32bits * 3200MT/s / 8bits = 25.6GB/s
- CPU clock speed : 2GHz

Tegra X1 - Jetson TX1

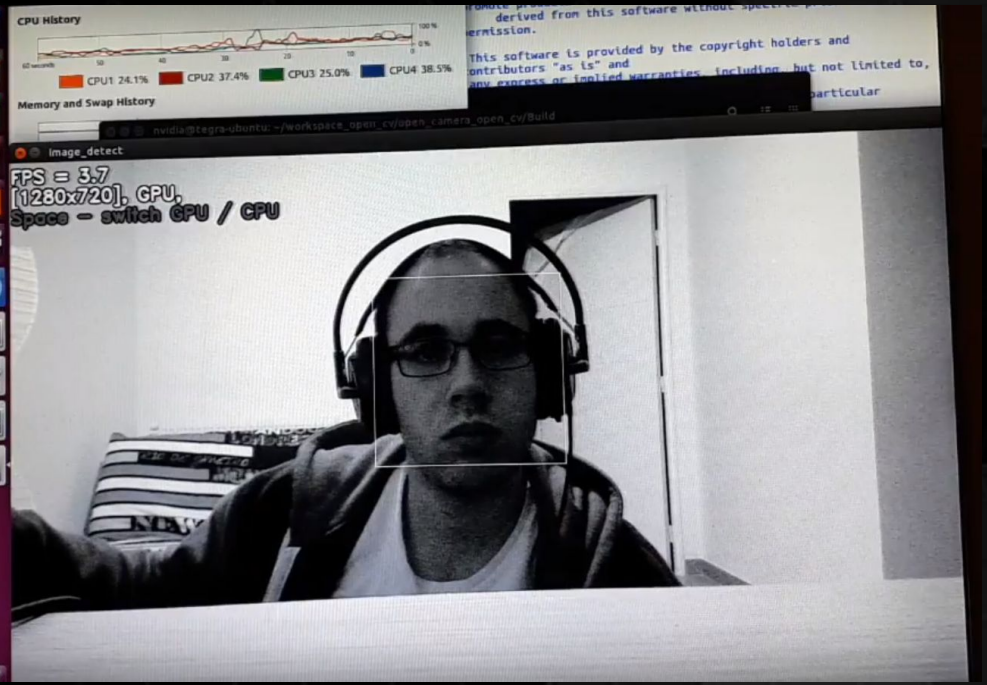


- Module Jetson pour les systèmes embarqués
- Module de développement avec port HDMI, USB 3.0, lecteur SD…
- Linux Kernel installable via L4T, bootable
- Cross-compilation via gcc
- Programmation en C++

Tegra X1 - Exemple d'application

main_cuda.cpp | test_cuda.cpp | main_vision.cpp | iomanip | *main.cpp

```
86          frame_gpu.upload(grayframe);
87
88          TickMeter tm;
89          tm.start();
90
91          if(useGPU)
92          {
93              cascade_gpu->detectMultiScale(frame_gpu, facesBuf_gpu);
94              cascade_gpu->convert(facesBuf_gpu, faces);
95          }
96          else
97          {
98              cascade_cpu.detectMultiScale(grayframe, faces);
99          }
100
101         for(size_t i=0; i < faces.size(); i++)
102         {
103             rectangle(grayframe, faces[i], Scalar(255));
104         }
105         tm.stop;
106
107         double detectionTime = tm.getTimeMilli();
108         double fps = 1000/detectionTime;
109
110         displayState(grayframe, useGPU, fps);
111         imshow("image_detect", grayframe);
112
113         char key = (char)waitKey(1);
```



Conclusion

- big.LITTLE : technologie work/sleep
  LITTLE : faible consommation, big : hautes performances
- Armv8 : adaptaté à une grande quantité de données => fait pour les besoins applicatifs d'aujourd'hui, technologie innovante (SVE)
- Tegra X1 : processeur mobile avec le Jetson TX1 : module pour systèmes embarqués
  Hautes performances, se rapproche des systèmes fixes

  Ouverture aux serveurs HCP, supercalculateurs