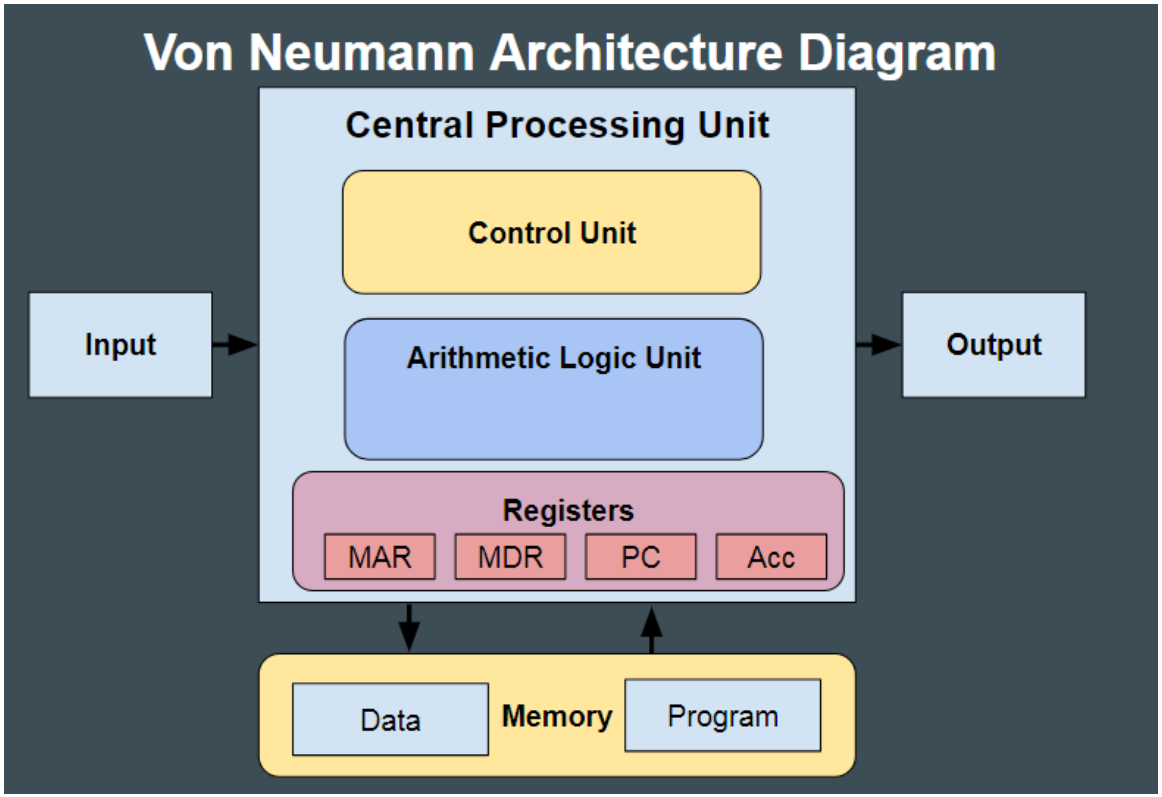
Calcul sur FPGA

21.06.2018

Bogdan Vulpescu
LPC+

Un changement du paradigme de calcul

- MAR = Memory Address Register
- MDR = Memory Data Register
- PC = Program Counter
- Acc = Accumulator
- ALU = Arithmetic Logic Unit
- CU = Control Unit



learnlearn.uk/gcsecs/von-neumann-achitecture

Journée calcul du LIP sur les FPGA

21 juin 2018

Thème

Le but de la journée est de dresser un petit panorama de l'utilisation des FPGA comme accélérateurs de calcul. Nous souhaitons en particulier avoir un aperçu des techniques et réalisations existantes et notamment aborder les sujets suivants :

- Qu'est-ce qu'un FPGA, quelles ressources de calcul sont disponibles, comment s'intègrent-ils aux plateformes de calcul traditionnelles ?
- Quels sont les outils de programmation existants, de bas niveau (VHDL, Verilog) ainsi que les langages de synthèse de haut niveau.
- Quelques exemples de mise en oeuvre pratique.

Localisation et inscription

Pour des raisons d'organisation, merci d'indiquer votre participation sur ce doodle avant le **lundi 18 juin**:

https://evento.renater.fr/survey/participation-a-la-journee-calcul-du-lip-du-21-juin-2018-mw7fgzs4

La journée aura lieu à l'ENS de Lyon, site Monod, dans l'amphithéatre B au 3e étage du batiment M1

Programme

- Matin :
  - 9h30-10h15: Florent De Dinechin (INSA de Lyon), *Introduction au calcul sur FPGA*
  - 10h45-11h30: Florent De Dinechin (INSA de Lyon), *Les FPGA pour calculer au plus juste*
- Après midi :
  - 13h-13h45: Yohann Uguen (INSA de Lyon), *Aperçu des outils de synthèse de haut-niveau*
  - 13h45-14h45: Bodgan Pasca (Intel), *Matériels et outils proposés par Intel*
  - 15h15-16h15: Steven Derrien (Université de Rennes)
  - 16h15-17h: Matthieu Haefele (Maison de la simulation), *Portage sur FPGA de Metalwalls, un code de dynamique moléculaire, avec les technologies proposées par la société Maxeler*

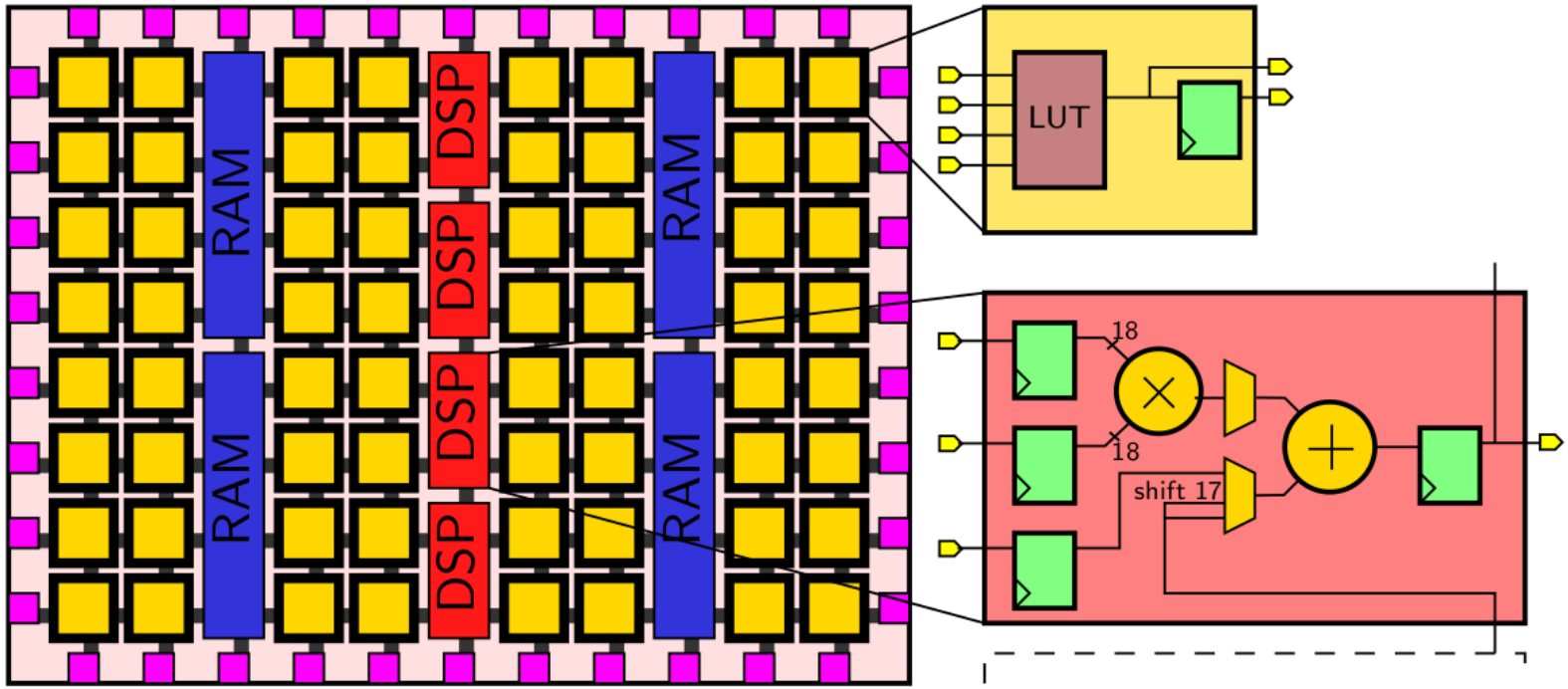
4 diapos de la présentation « FPGA Multipliers » par Bogdan PASCA à la 11ème Rencontres Arithmétiques de l'Informatique Mathématique (RAIM'11)

What's an FPGA?

Field **P**rogrammable **G**ate **A**rray

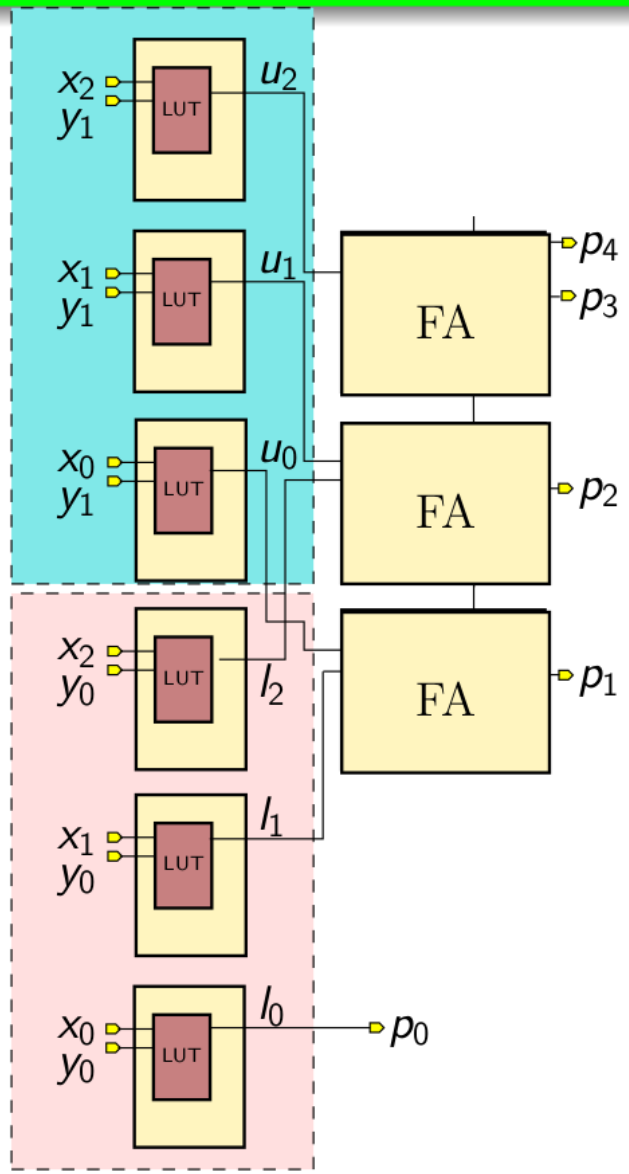
- integrated circuit
- has a regular architecture (hence **array**)
- logic elements can be programmed to perform various functions

Modern FPGA Architecture



- a set of **configurable** logic elements
- on chip memory blocks
- digital signal processing (DSP) blocks (including multipliers)
- connected by a **configurable** wire network
- all connected to outside world by I/O pins

What can we compute?



$$
\begin{array}{r}
x_2\,x_1\,x_0\times \\
\hline
y_1\,y_0 \\
\hline
l_2\;\;l_1\;\;l_0+ \\
u_2\,u_1\,u_0 \\
\hline
p_4\,p_3\,p_2\,p_1\,p_0
\end{array}
$$

$$
\begin{aligned}
l_0 &= y_0 \wedge x_0 \\
l_1 &= y_0 \wedge x_1 \\
l_2 &= y_0 \wedge x_2
\end{aligned}
$$

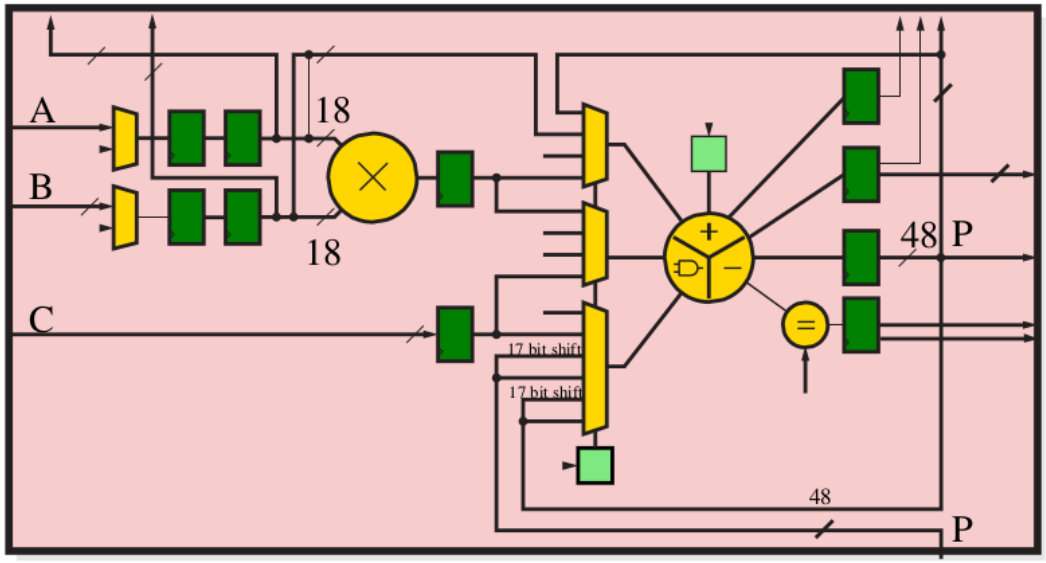
$$
\begin{aligned}
u_0 &= y_1 \wedge x_0 \\
u_1 &= y_1 \wedge x_1 \\
u_2 &= y_1 \wedge x_2
\end{aligned}
$$

Need of DSP blocks

Multiplication in logic is expensive

- $n \times n\ bit \approx \underbrace{n^2}_{\text{partial products}} + \underbrace{n(n-1)}_{\text{adder tree}} LUTs$
- $18 \times 18\ bit \approx 324LUT + 306LUT = 630LUTs$
- 1 DSP block = 8 LEs (size on FPGA layout)

DSP blocks are a need in modern FPGAs



Démo


Download DE1-SoC CD from
http://de1-soc.terasic.com
DE1-SoC
ALTERA
PREFERRED BOARD FOR
OpenCL™

Projet FPGA en langage Verilog HDL (Quartus II 16.0)

allumer et éteindre un LED

Projet FPGA avec Verilog HDL et blocs IP (Quartus II 16.0)

faire défiler les LED's et ralentir/accélérer la vitesse

Projet FPGA + ARM (Quartus II 16.0)

comptage avec les LEDs piloté par le code hôte (ARM)

Créer et utiliser des bibliothèques OpenCL pour le FPGA (Quartus II 16.0)

racine carrée et son inverse

Noyau OpenCL pour le calcul sur FPGA (Quartus II 16.0)

réduction complète : addition d'une série de 1048576 floats

Modèle de calcul

école	*device* OpenCL (CPU, GPU, FPGA, ...)
problème à résoudre	programme *kernel*
salle de classe	composante de calcul (coeur CPU, ...)
classe d'élèves	groupe de travail (*work-group*)
élève dans une classe	unité de calcul (*work-item*)

mémoire = globale (table noire centrale), locale (table noire dans la classe), privée (notebook)

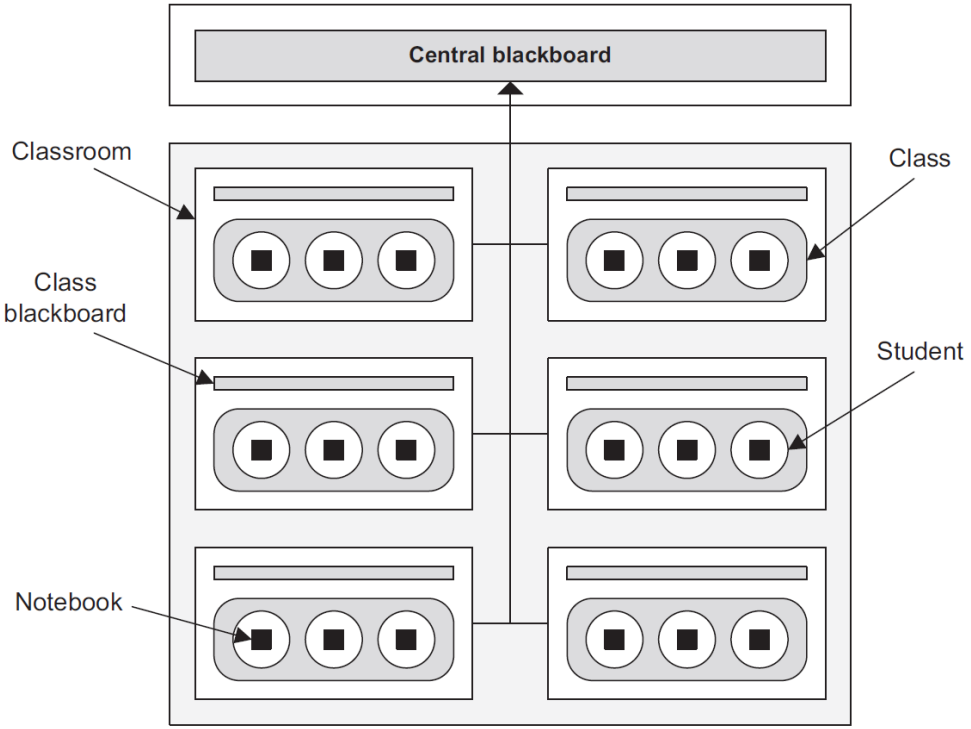


Figure 4.6 **School of math students in OpenCL device analogy**

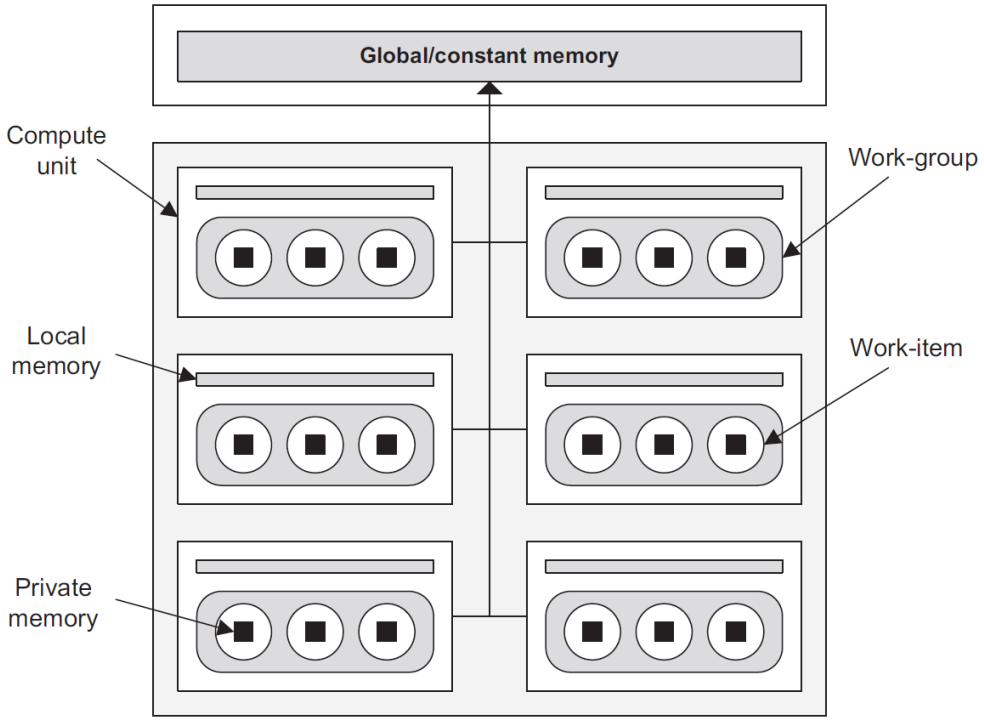


Figure 4.7 **OpenCL device model**

Exemple avec 16 nombres :

1) lancée du 1er noyau avec 4 groupes de 4 unités

```
groupe 0:
  lid = 0,1,2,3
  partial_sums[lid] = data[get_global_id]
                              0,1,2,3
    I = group_size/2 , /2 , …
    lid = 0
      i = 2
      partial_sums[0] += partial_sums[0+2]
      i = 1
      partial_sums[0] += partial_sums[0+1]

      out[0] = partial_sums[0]

    lid = 1
      i = 2
      partial_sums[1] += partial_sums[0+3]

group 1:
  lid = 0,1,2,3 group_size = 4
  partial_sums[lid] = data[get_global_id]
                              4,5,6,7
    lid = 0
      ...

      out[1] = partial_sums[0]

    lid = 1
      ...
```

```
group 2:
  lid = 0,1,2,3 group_size = 4
  partial_sums[lid] = data[get_global_id]
                              8,9,10,11

    lid = 0
      ...

      out[2] = partial_sums[0]

    lid = 1
      ...

group 3:
  lid = 0,1,2,3 group_size = 4
  partial_sums[lid] = data[get_global_id]
                              12,13,14,15

    lid = 0
      ...

      out[3] = partial_sums[0]

    lid = 1
      ...
```

barrière d'accès mémoire

2) lancer le 1er noyau avec *global_size*=16/4 > *local_size*=4 ? Non.

3) lancer le 2ème noyau avec *global_size*=16/4=4 et *local_size*=4
---> 1 seul groupe

```
group 0:
  lid = 0, 1, 2, 3 group_size = 4
  partial_sums[lid] = data[get_global_id]    avec data[] = out[] du 1er noyau
                                0,1,2,3
    lid = 0
      i = 2
      partial_sums[0] += partial_sums[0+2]
      i = 1
      partial_sums[0] += partial_sums[0+1]

      out[0] = partial_sums[0]
```

Fin !

Démo


ALTERA
Cyclone V
FPGA • SoC
Cyclone V
FPGA DEVELOPMENT KIT
PCI EXPRESS

Transformation Fourier Discrète sur FPGA (Quartus II 17.0)