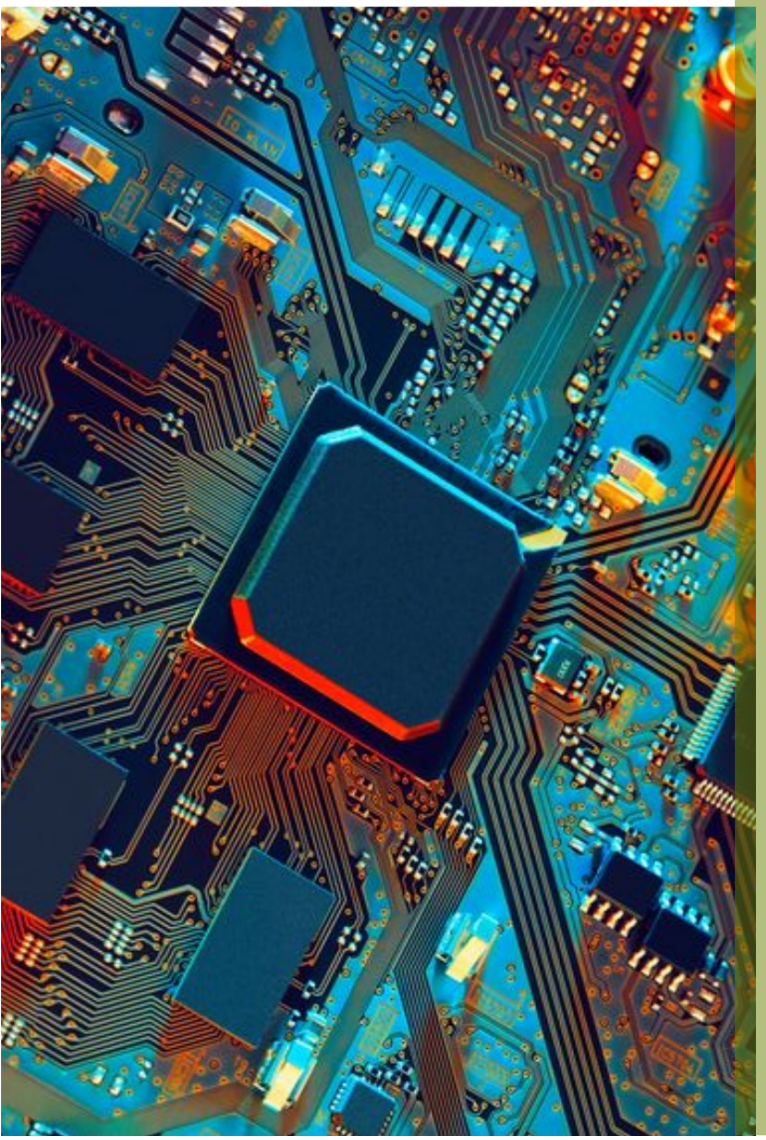
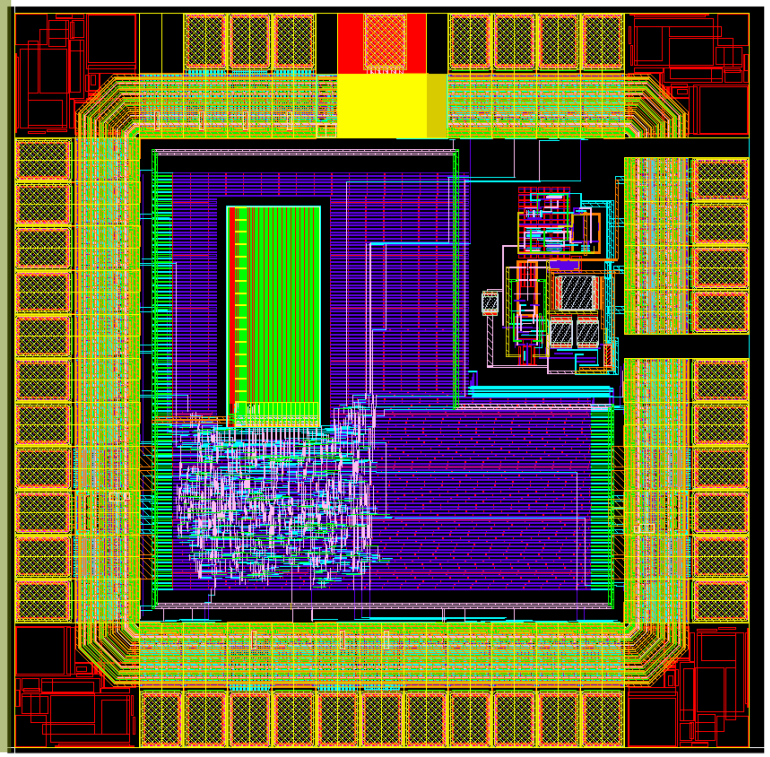


Techno des Circuits Numériques

EN208

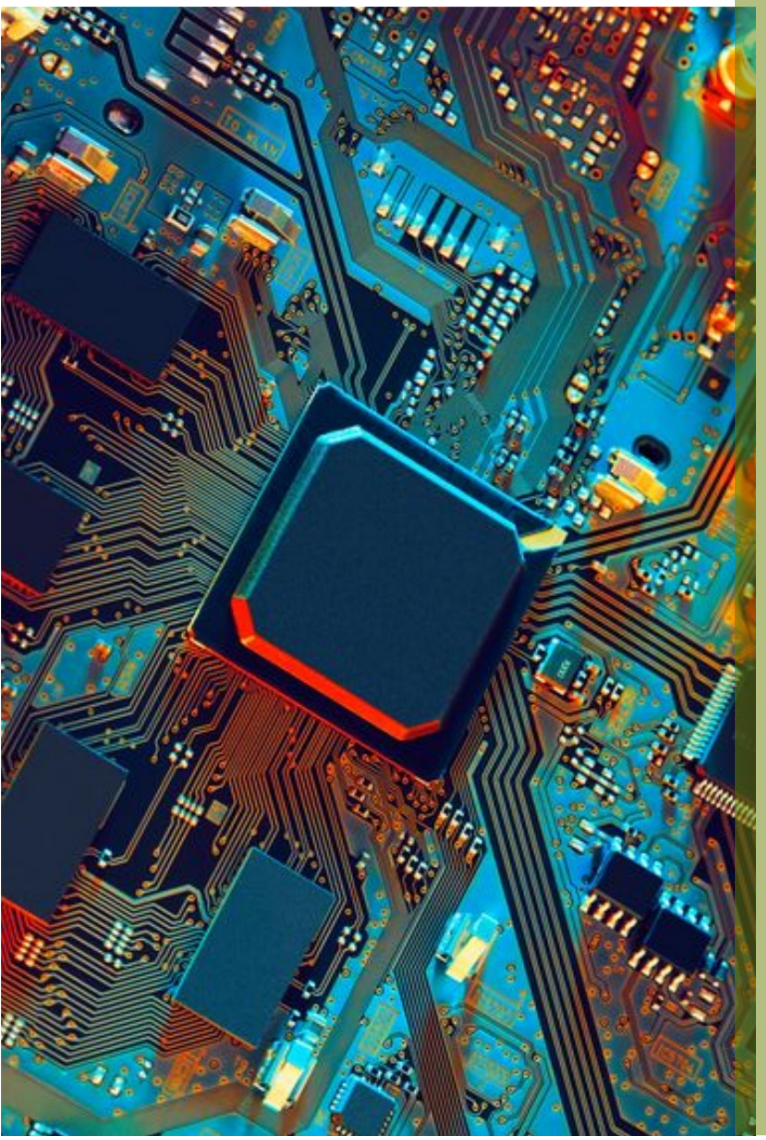
Yannick Bornat



Plan du cours

8 séances de CM :

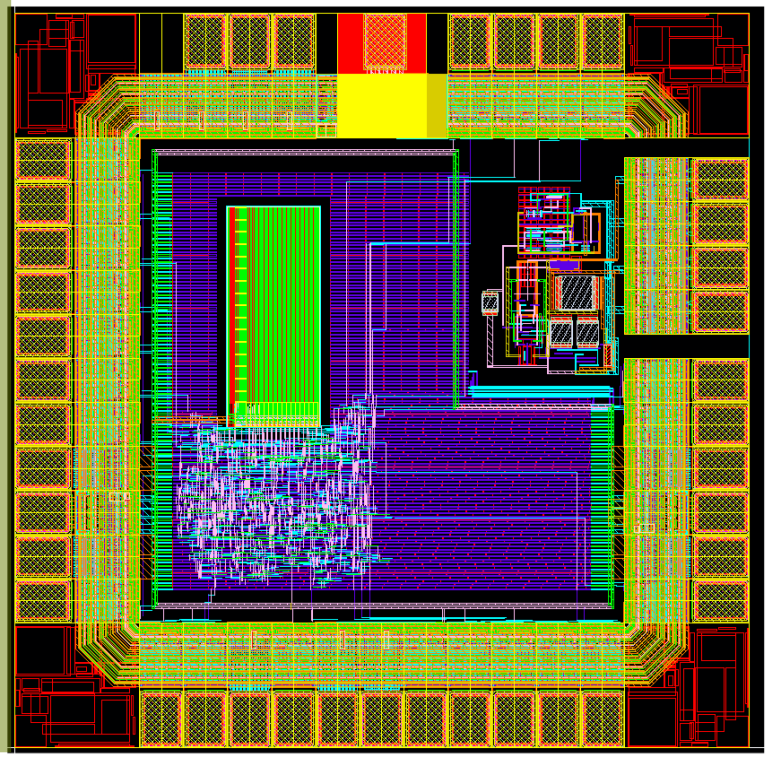
1	- Rappels sur le VHDL	Y. Bornat
2	- Intro : culture de l'industrie	S. Renaud (mardi prochain)
3,4	- FPGA : Architecture, placement/routage	Y. Bornat
5,6,7	- familles de CI numériques : caractéristiques performances …	S. Renaud
8	- Test	S. Renaud



Le FPGA
sa vie, son œuvre…

EN208

Yannick Bornat



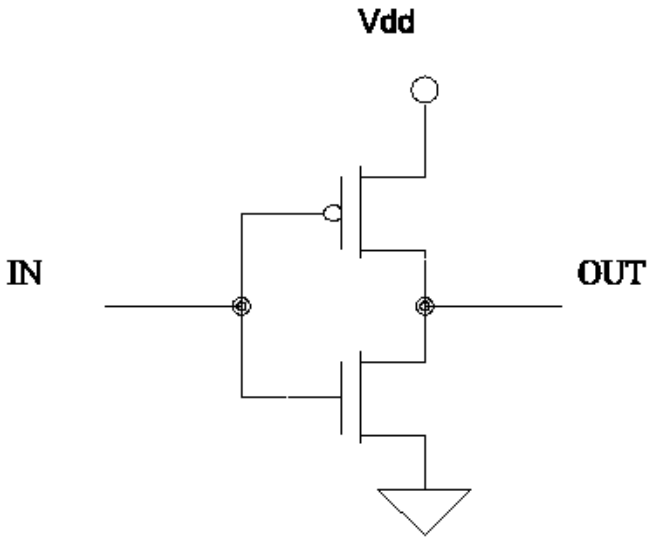
Faire du calcul avec des portes logiques …

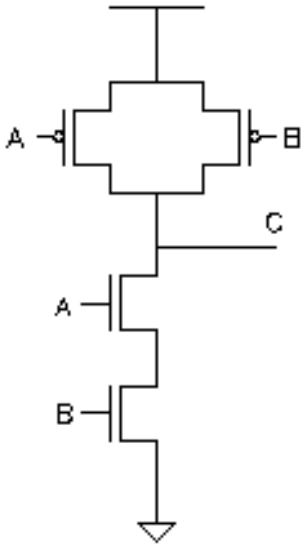
Pour l'instant, on n'a vu que les fonctions de base :

- **NOT**
- **NAND**
- **NOR**

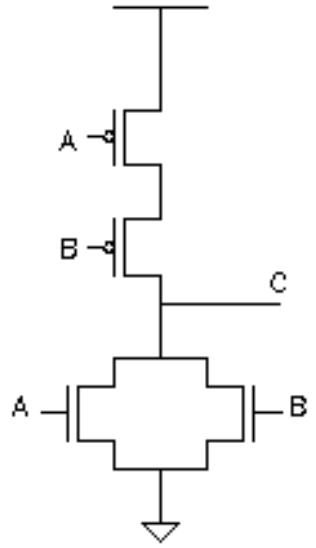
(Version CMOS)

- **Possibilité de créer des versions à 3, 4 +++ entrées**





NAND

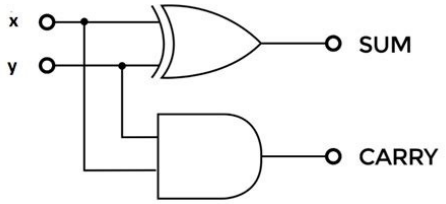


NOR

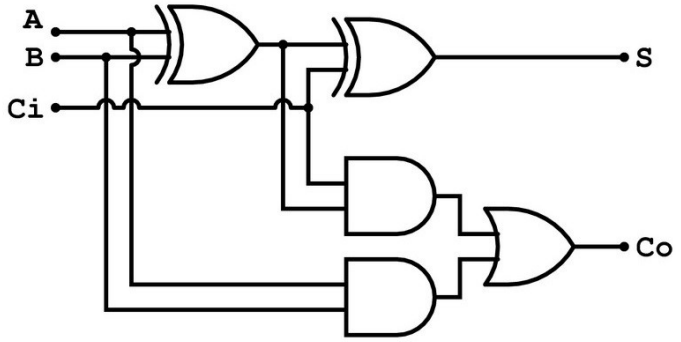
Faire du calcul avec des portes logiques …

L'addition :

- **2 valeurs sur 1 bit : facile…**

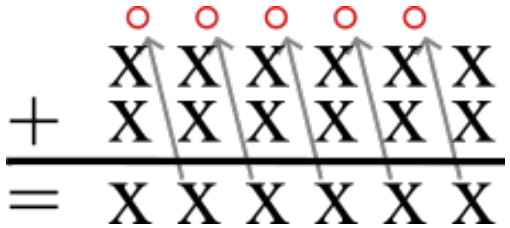


- **3 valeurs sur 1 bit : un peu moins facile :**



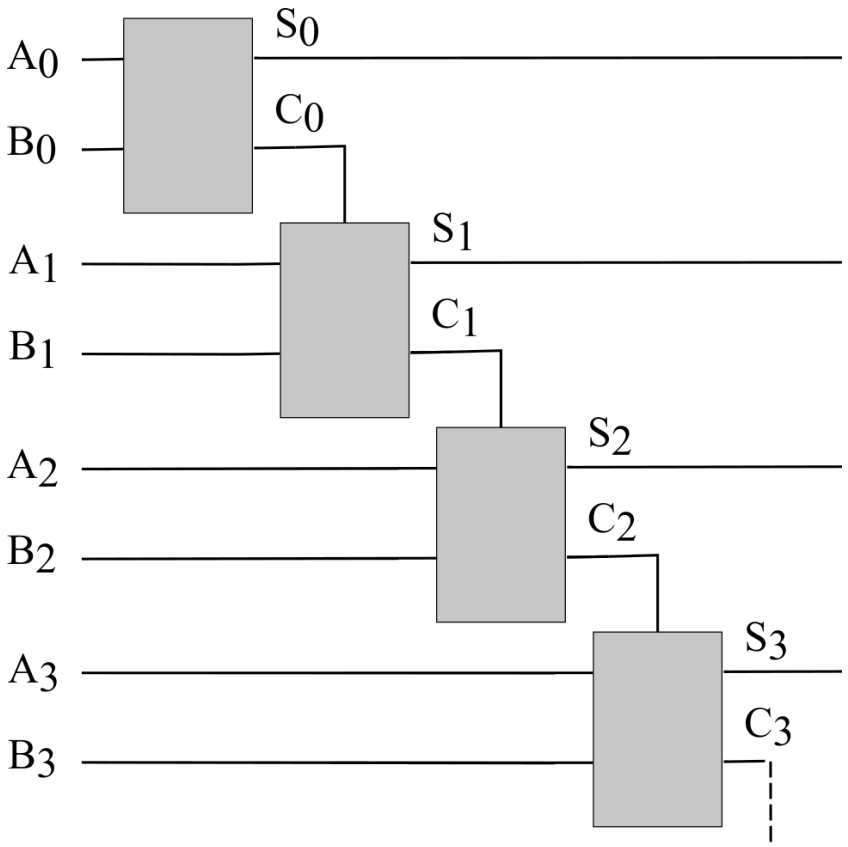
- **2 valeurs sur n bits …**

$$\begin{array}{r} X\ X\ X\ X\ X\ X \\ +\ X\ X\ X\ X\ X\ X \\ \hline =\ X\ X\ X\ X\ X\ X \end{array} \Rightarrow$$



Faire du calcul avec des portes logiques …

L'addition : Schéma final :



Faire du calcul avec des portes logiques …

Cas particulier : le compteur

Faire du calcul avec des portes logiques …

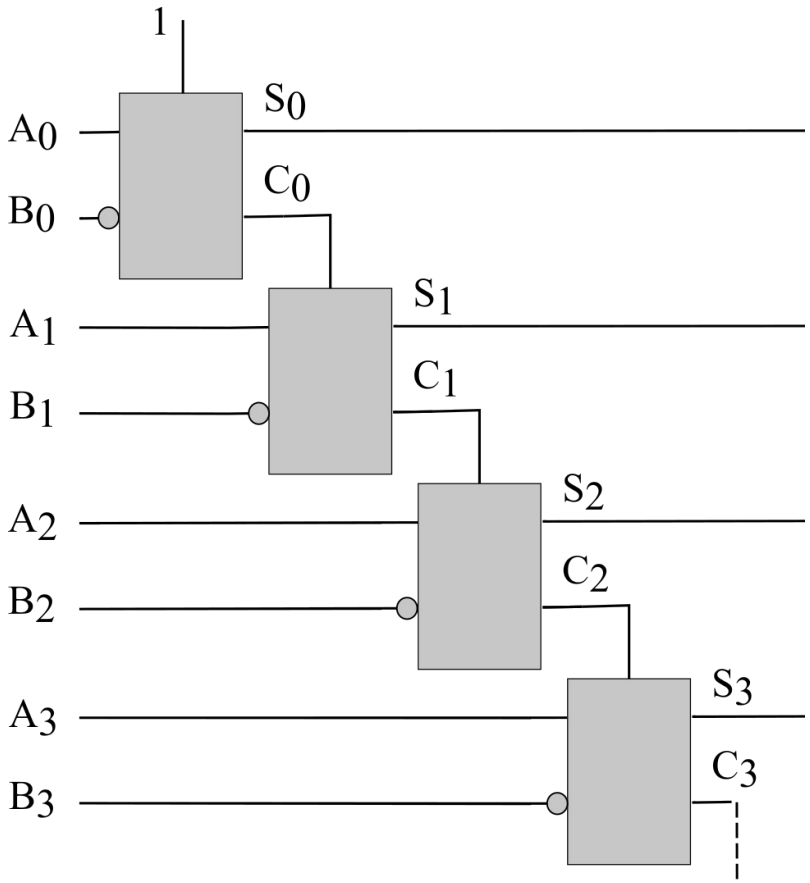
Cas particulier : la soustraction / le décompteur

- **Le truc important :**

  not(A) = -A-1 ou -A = not(A) + 1

Pour faire les deux :

- **Bit sub en entrée**
  - **Connecté au premier carry**
  - **En XOR avec l'entrée B**



Faire du calcul avec des portes logiques …

Focus : le chemin critique …

- **Définition :**
  - **Le temps de propagation le plus long entre les entrées et la/les sorties**
- **Détermine la fréquence max de fonctionnement**
- **Permet d'identifier la partie à optimiser pour monter en fréquence**

Faire du calcul avec des portes logiques …

La multiplication :

- **2 valeurs sur 1 bit : facile…**
  - **Table de multiplication à connaître par cœur :**

- **2 valeurs sur n bits :**

Faire du calcul avec des portes logiques …

La multiplication : schéma final

Faire du calcul avec des portes logiques …

La division/modulo :

- **Pareil : on fait comme à la main**
  - **Difficulté : les prises de décision**
  - **Pour chaque nouveau bit, deux possibilités**
    - **Plusieurs chemins critiques**

- **Solution lente: méthode itérative**
  - **On traite un nouveau bit à chaque front d'holorge**
- **Solution intermédiaire :**
  - **On traite n bits à chaque front d'horloge (radix n):**
    - **N fois plus rapide**
    - **Moins complexe que tout en une fois**
- **Possibilité de faire un pipeline pour augmenter les perfs en fréquence**

Faire du calcul avec des portes logiques …

Le reste :

- **On se démerde au cas par cas**
- **souvent : suite mathématique qui converge vers la solution**
  - **Le plus rapide : dichotomie par fonction réciproque**
- **Cas particulier : en trigo : CORDIC**

Supports de Calcul

Processeur
DSP
GPU

FPGA

ASIC

Supports de Calcul

Le processeur

Circuits logiques reconfigurables

Types de circuits :

- **PLD**
- **CPLD**
- **FPGA**

(re)configuration :

- **One-time (fusibles)**
- **EEPROM**
- **SRAM (volatile)**

Techno :

- **Low cost**
- **Low power**
- **High Freq**

Circuits logiques reconfigurables

PAL / GAL / PLD : le bourrin !

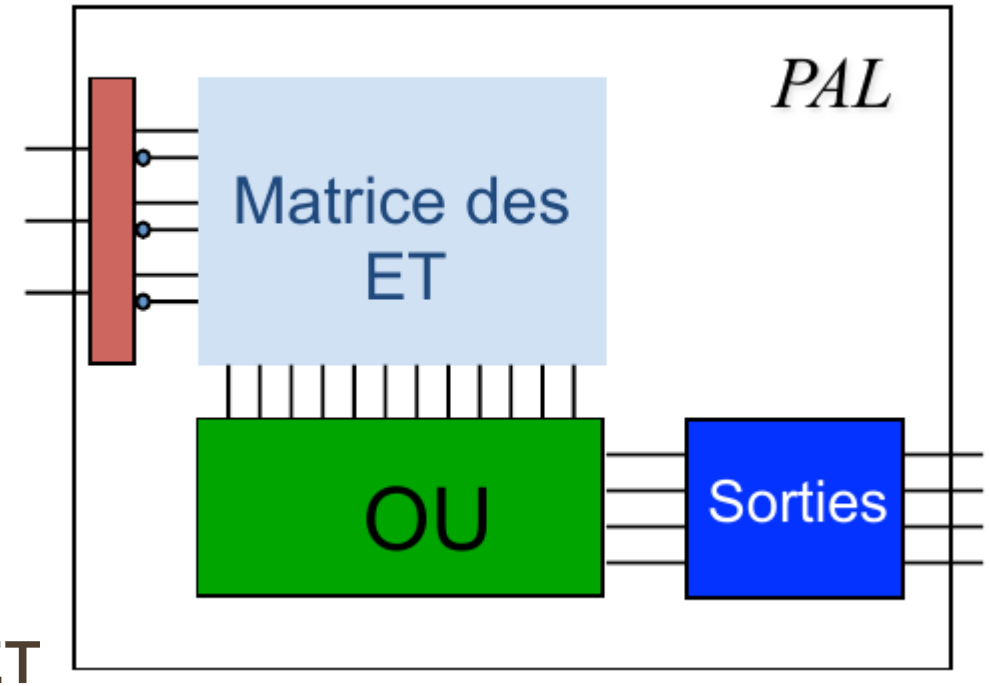
- **Programmable Array Logic**
- **Generic Array Logic**
- **Programmable Logic Device**

Techno / architecture / mémoire …

Circuits logiques reconfigurables

PAL / GAL / PLD : le bourrin !

- **Programmable Array Logic**
- **Generic Array Logic**
- **Programmable Logic Device**



Toutes les entrées sont complémentées
Une matrice de connexion vers des opérateurs ET
Une matrice de connexion des opérateurs ET vers les opérateurs OU
Un (éventuel) étage de sortie

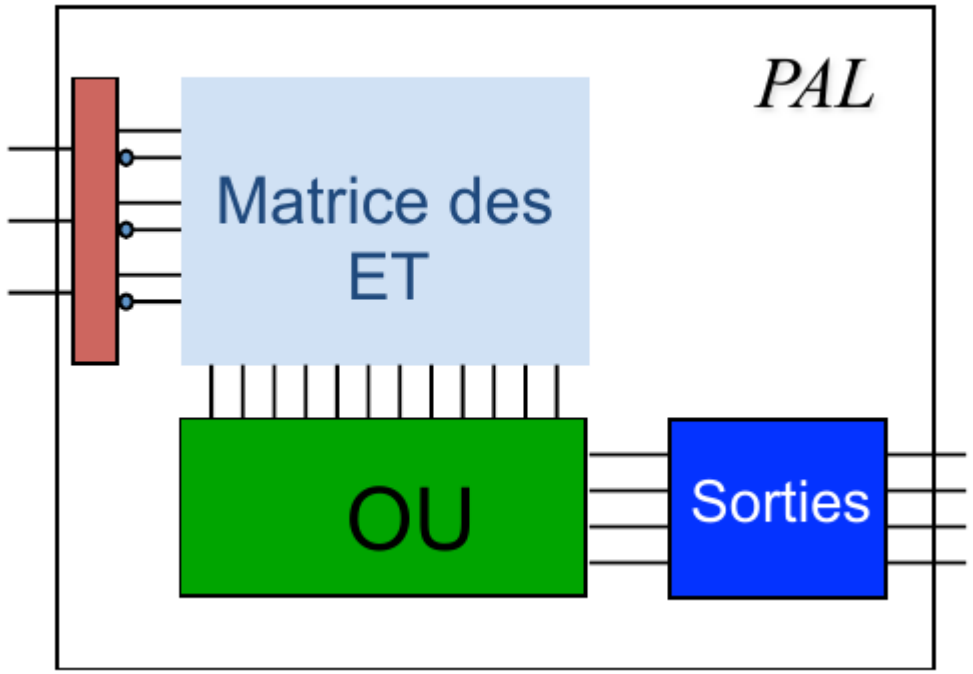
- **Inverseurs**
- **Logique 3 états**
- **Regsitres**
- **…**

Circuits logiques reconfigurables

PAL / GAL / PLD : le bourrin !

Une connexion, c'est ….
… un transistor controlé par un bit mémoire

Passant => connecté
Bloqué => déconnecté



Reconfigurer le composant, c'est mettre à jour l'ensemble de ces bits de contrôle. Et puis c'est tout.

Circuits logiques reconfigurables

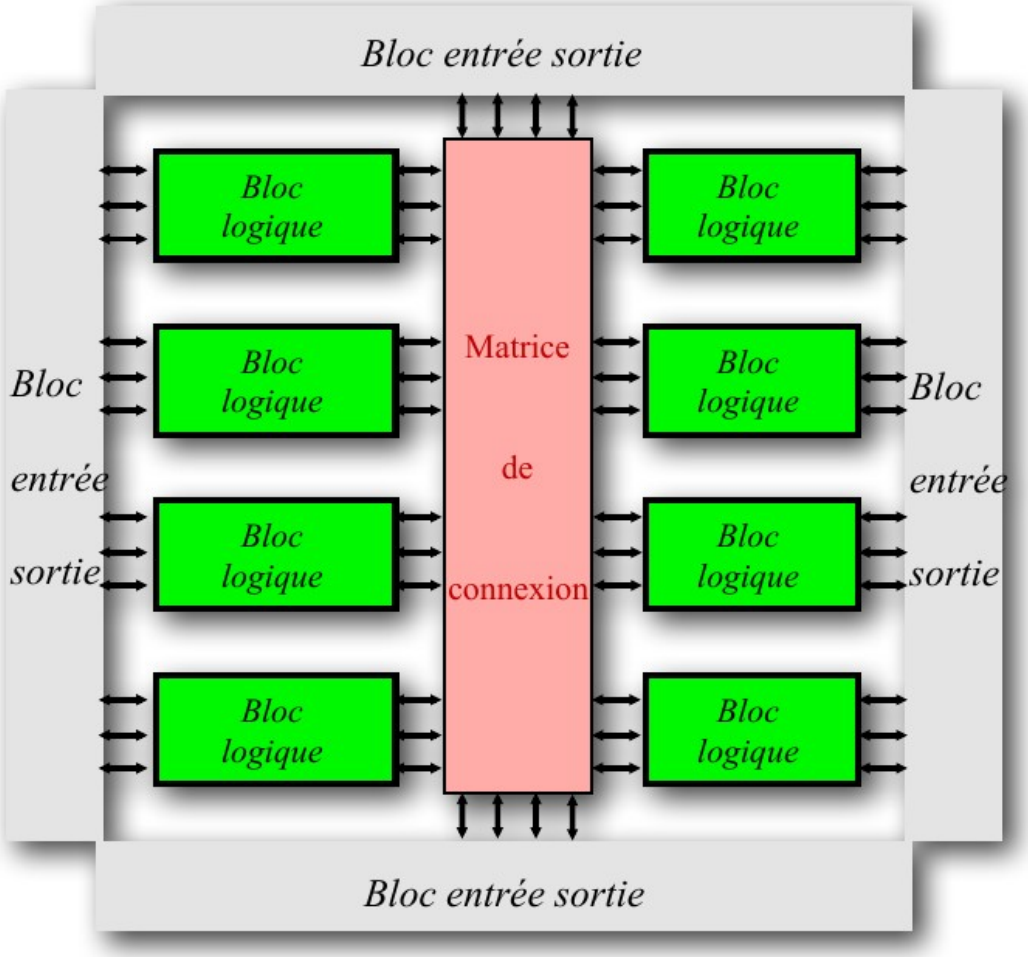
CPLD : le subtil !

Constat :

- **On n'utilise jamais TOUTES les entrées dans une équation logiques**
- **Pas besoin de garantir qu'il soit possible de tout combiner**

Architecture :

- **Similaire à plusieurs PLD reliés à l'aide d'une matrice (configurable) de connexions**

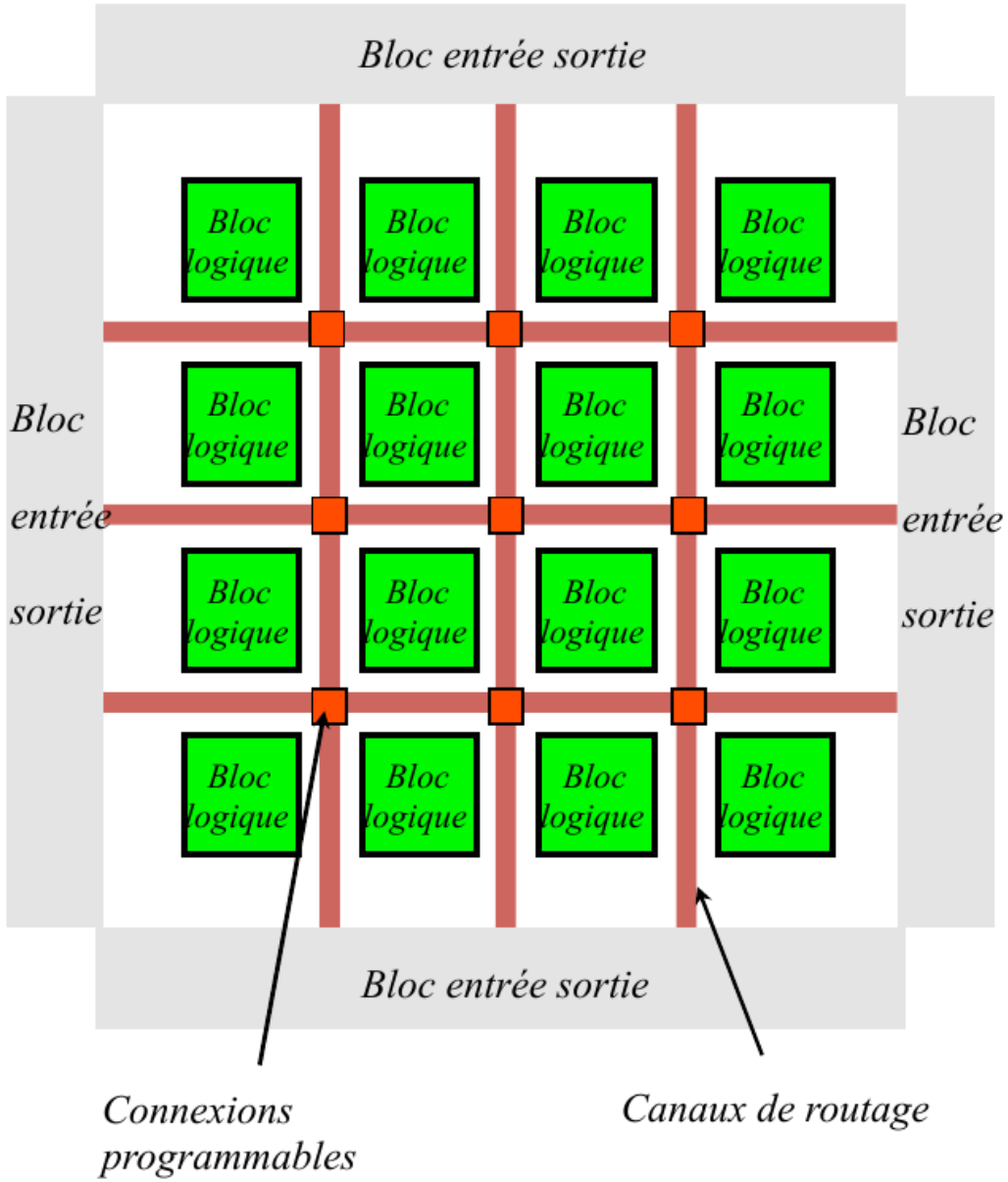


Circuits logiques reconfigurables

FPGA : le tricheur !

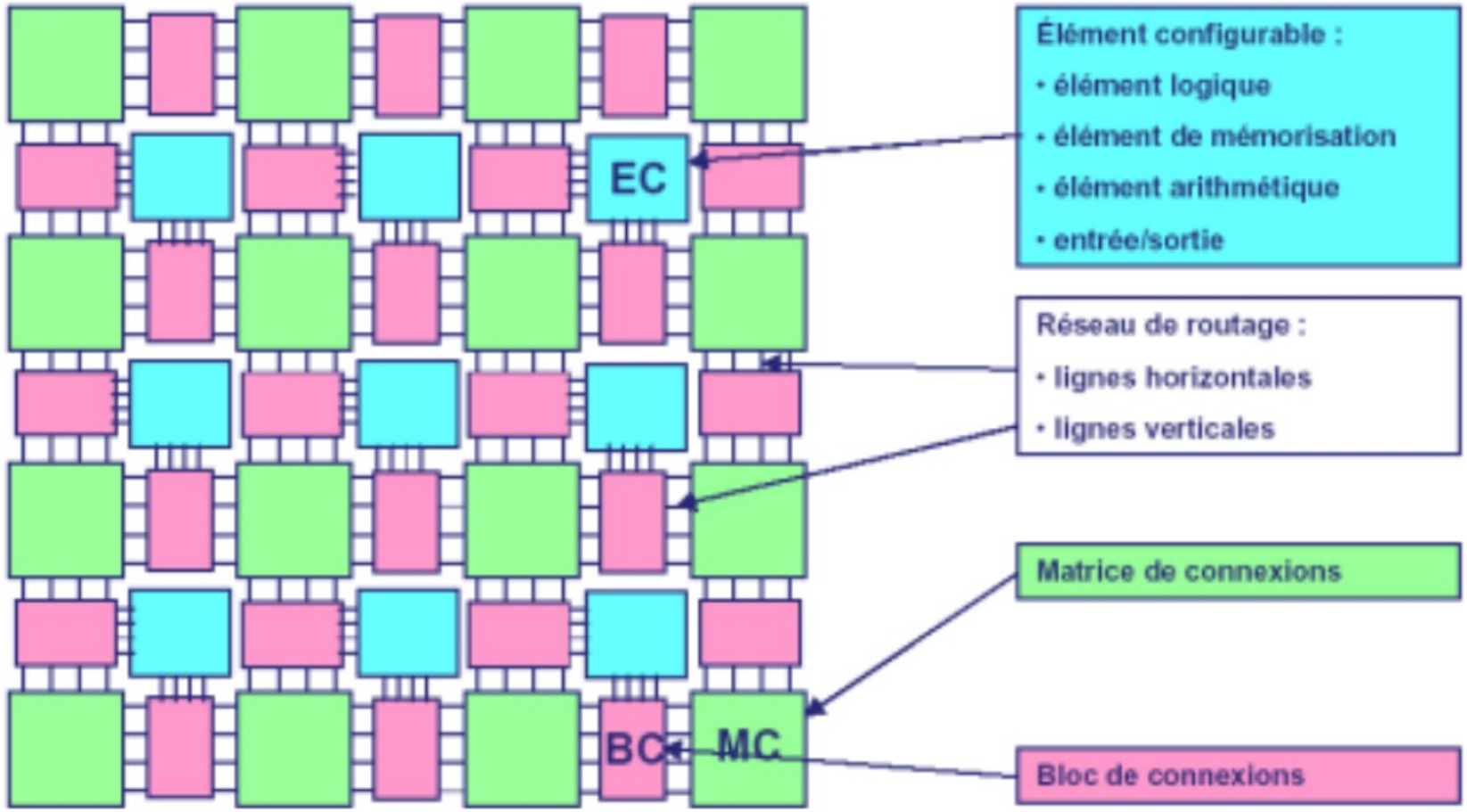
Constat :

- **On n'a pas besoin de portes logiques, juste de quelquechose qui y ressemble "suffisamment"**
  - **Augmentation de la densité**
- **Au-delà d'une certaine taille, pas besoin de centraliser les interconnexions**
  - **Généralement peu d'interconnexions lointaines**



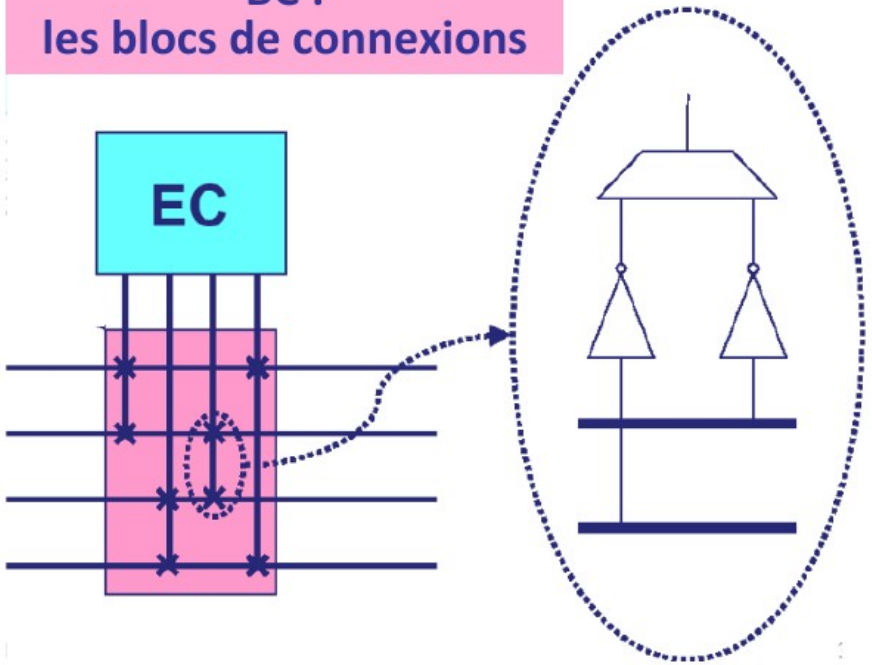
Structure interne d'un FPGA : principe

Explications techniquement simplistes,
mais ça donne une bonne idée…

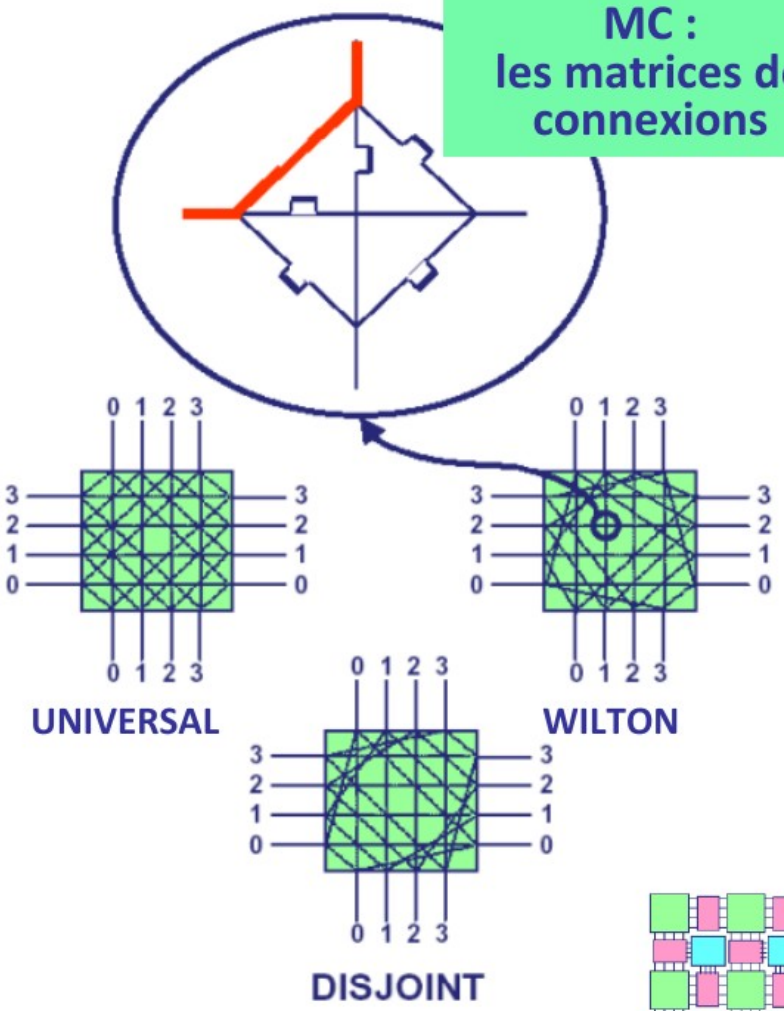


Structure interne d'un FPGA : principe

BC :
les blocs de connexions

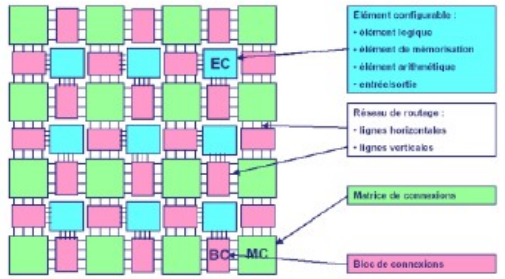


MC :
les matrices de connexions



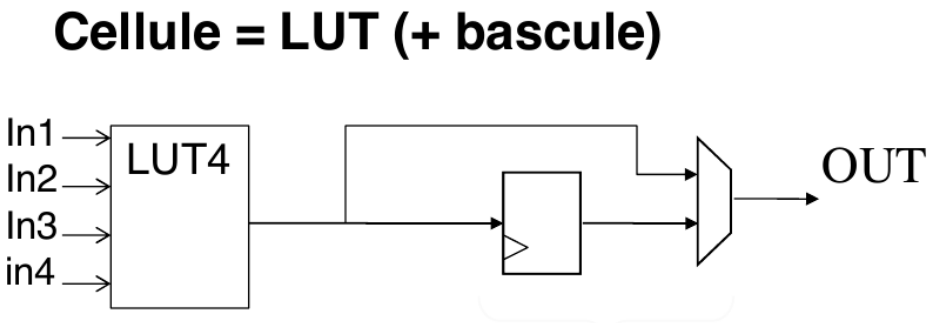
Les **blocs de connexions** assurent la connexion des éléments configurables

Les **matrices de connexions** assurent la connexion des blocs de connexions



Structure interne d'un FPGA : principe

Element logique de base …
(de base = techno d'il y a 30 ans…)



Une partie "LUT" s'occupe de l'équation logique/combinatoire

- **Ici à 4 entrées**

On utilise (ou pas) un bascule en sortie grâce à un multiplexeur

- **Contrôle du multiplexeur : bit de configuration**

Si pas besoin d'équation logique devant, on en met quand même une :

- **F(in1, in2, in3, in4) = in1**

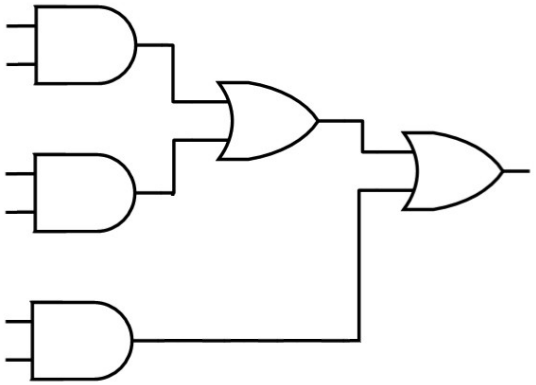
Structure interne d'un FPGA : principe

Lien entre LUT et equation logique …

On aimerait : **out = F(in1, in2, in3, in4)**

On formalise : **out = (in1 ET in2) OU (in1 ET in3) OU (in2 ET in4)**

On imagine :



Mais on a uniquement besoin de ….

Structure interne d'un FPGA : principe

Lien entre LUT et equation logique …

Ça :

In1	In2	In3	In4	out
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

In	out
0000	0
0001	0
0010	0
0011	0
0100	0
0101	1
0110	0
0111	1
1000	0
1001	0
1010	1
1011	1
1100	1
1101	1
1110	1
1111	1

ADDR	out
0	0
1	0
2	0
3	0
4	0
5	1
6	0
7	1
8	0
9	0
A	1
B	1
C	1
D	1
E	1
F	1

Structure interne d'un FPGA : principe

Lien entre LUT et equation logique …

Une LUT4 est une mémoire de 16x1bit

- **Chacun des 4 bits d'addresse représente une entrée de l'équation combinatoire**
- **La mémoire est remplie à la configuration**
- **Le contenue de la mémoire est la table de vérité recherchée**

Look Up Table (EN) : Table de correspondance (FR)

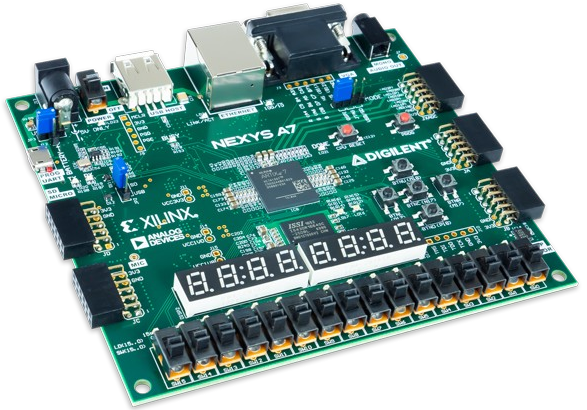
Aujourd'hui :

- **selon technos et vendeurs : LUT6 → LUT8**

Pour les plus grandes équations :

- **On combine les LUTs comme on le ferait avec des portes classiques …**

Pause : ordres de grandeur



Carte Nexys 4 / Nexys A7

- Sortie en 2010
- FPGA :Artix 7 – 100
- LUT6 : 63 400
- Bascules (Flip Flops) : 128 800
- Frequ typique : 100MHz easy / 200MHz atteignable
- ~ $350

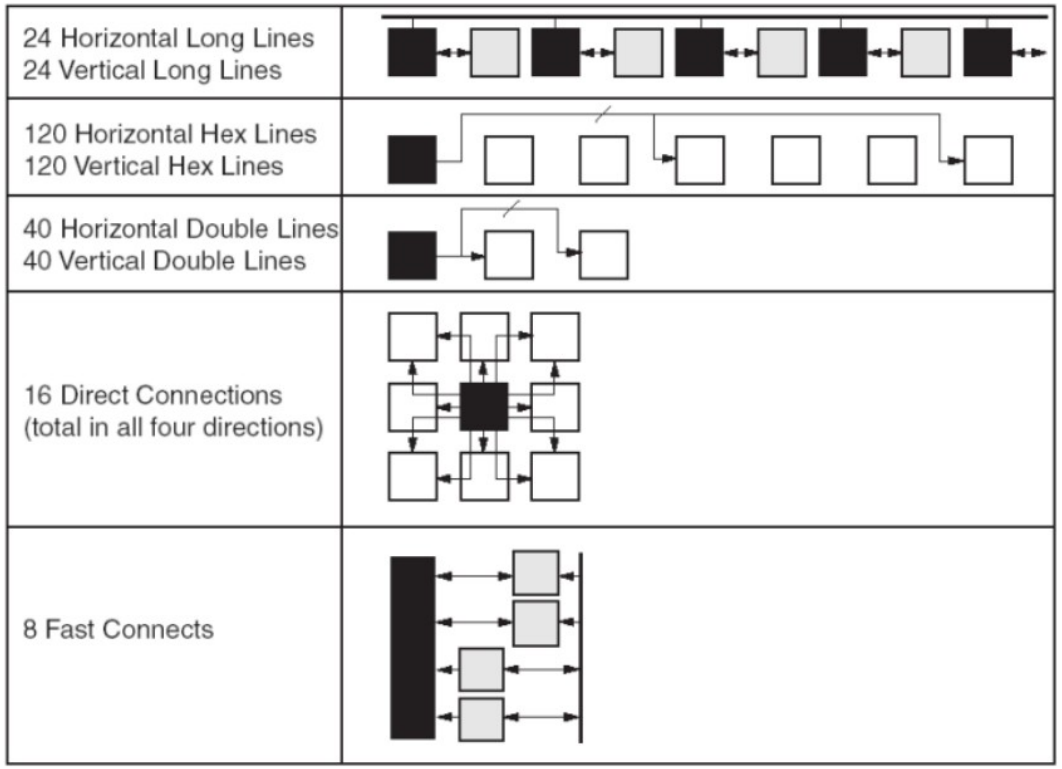
Virtex Ultrascale+ 13

- Sortie en 2019
- LUT6 : ~ 1 728 000
- Bascules (Flip Flops) : ~ 3 456 000
- Freq typique : 300 MHz easy / 500MHz atteignable
- > $10000

Structure interne d'un FPGA : Optimisations

Interconnexions :

- **Système de routage hiérarchique**
  - **Selon la portée géographique du signal**
- **Lignes dédiées aux horloges (peu)**
- **Lignes dédiées aux retenues**
  - **Locales et rapides**



Compromis de l'architecture :

beaucoup de connections longues
routage facilité
design plus rapide
MAIS : puce moins dense en éléments logiques
placement plus difficile
=> routage complexe …

Structure interne d'un FPGA : Optimisations

Elements logiques :

- **Plusieurs LUTs et bascules par elément**
  - **Artix 7 : 2 LUT6 + 4 bascules**
  - **Meilleures interconnexions locales**
- **Possibilité de dissocier les LUTs et les bascules**
  - **Meilleure densité**

Structure interne d'un FPGA : Optimisations

LUT4 ? LUT6 ? LUT8 ?

On retrouve le même compromis :

- **Grandes LUTs => moins de problèmes de routage**

Mais :

- **Grandes LUTs => moins de LUTs au total (à surface constante)**
  => occupation moins efficace (présence de petites equations)
  => placement moins optimal
  => difficultés de routage

Structure interne d'un FPGA : Optimisations

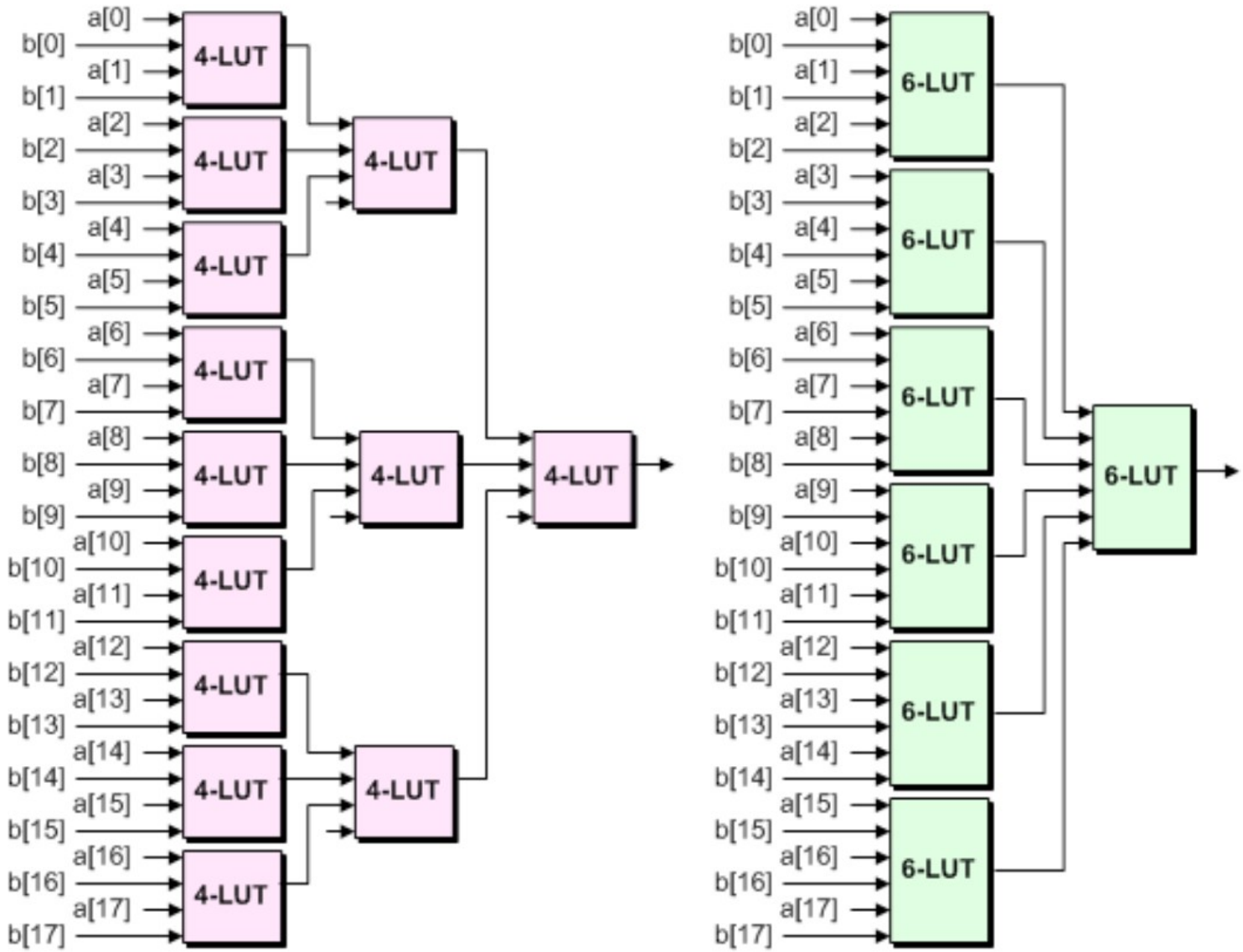
LUT4 ? LUT6 ? LUT8 ?

Autre aspect : la vitesse

Opération sur 2 registres 18 bits

- **LUT4 => 3 elts traversés**
- **LUT6 => 2 elts traversés**

Gain en vitesse 50 %



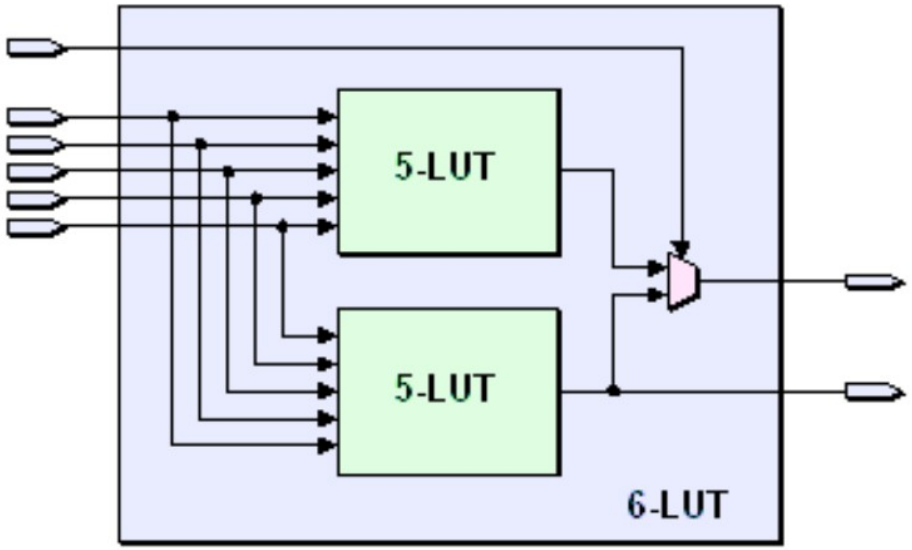
Structure interne d'un FPGA : Optimisations

Maximisation du rendement des LUTs

- 6 entrées … mais 2 sorties
- Basé sur 2 LUT5

Fonctions implémentées :

- 1 equation à 6 entrées
- 2 equations
  - 5 entrées identiques
  - 4 entrées chacune dont 3 communes
  - 3 entrées chacune dont 1 commune
  - …



Structure interne d'un FPGA : Optimisations

Maximisation du rendement des LUTs

Autres usages détournés des LUTs :

- **Registre à décalage**
  - **Config = chargement série ; le circuit existe déjà**
- **Mémoire (après tout… c'en est une)**
  - **Référencé comme LUTRAM dans le rapport de synthèse**
  - **Utilisable en lecture asynchrone (fait pour ça à la base…)**
- **Qqes transistors associés pour les additionneurs/soustracteurs**
  - **Propagation rapide de la retenue**
  - **Utilisation d'un maximum d'entrées**
- **…**

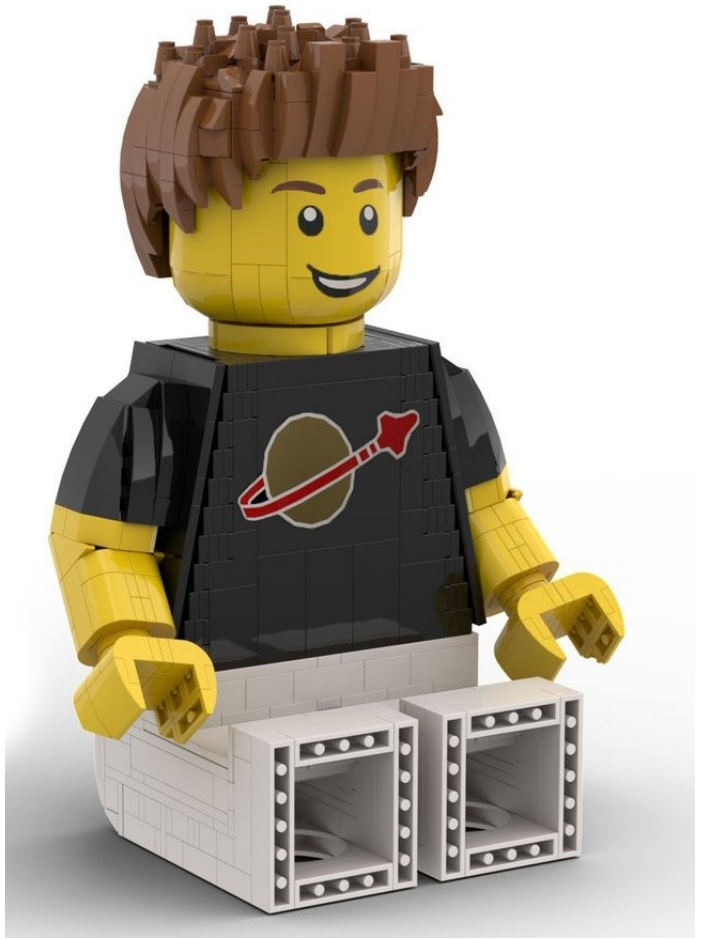
Structure interne d'un FPGA : Optimisations

On peut TOUT faire avec les blocs logiques …
Les blocs sont assez optimaux pour avoir une bonne densité

MAIS

Certaines structures de calcul se retrouve TRÈS souvent dans les designs…

- **On augmente encore les perfs en mettant des versions précablées de ces structures**
- **Degré de configurabilité faible, mais suffisant pour être très générique**



Structure interne d'un FPGA : Blocs matériels

Besoin d'optimiser la multiplication

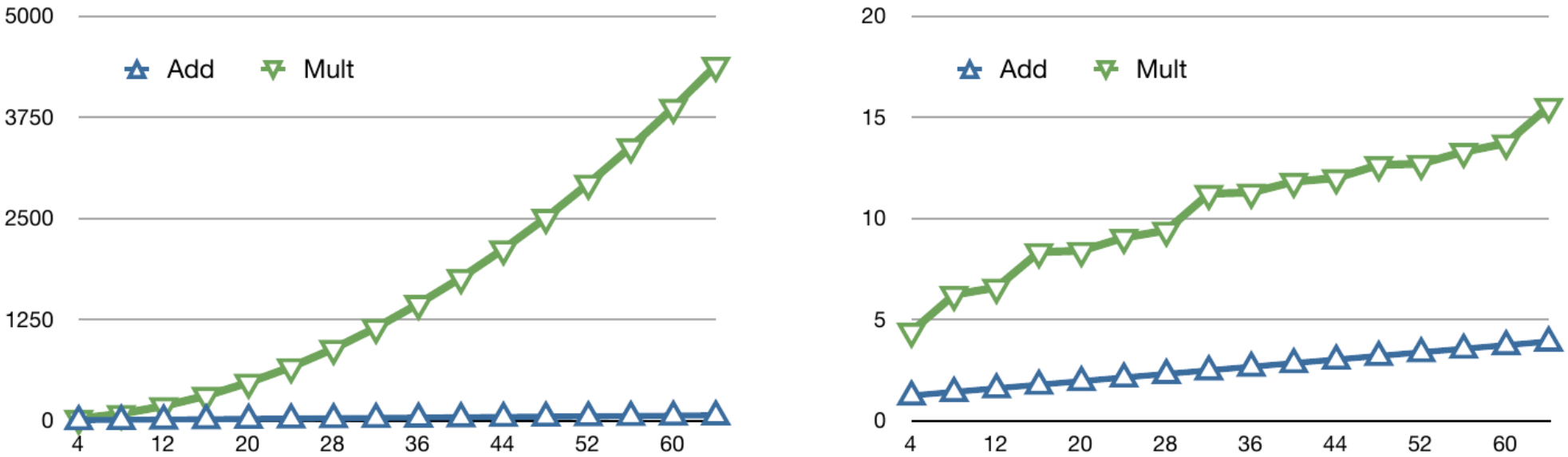


Figure 2 : *Évolution de caractéristiques des opérateurs (Add/Mult) entiers sur FPGA*

(a) Occupation en nombre de LUTs (b) Evolution du chemin critique en nanosecondes

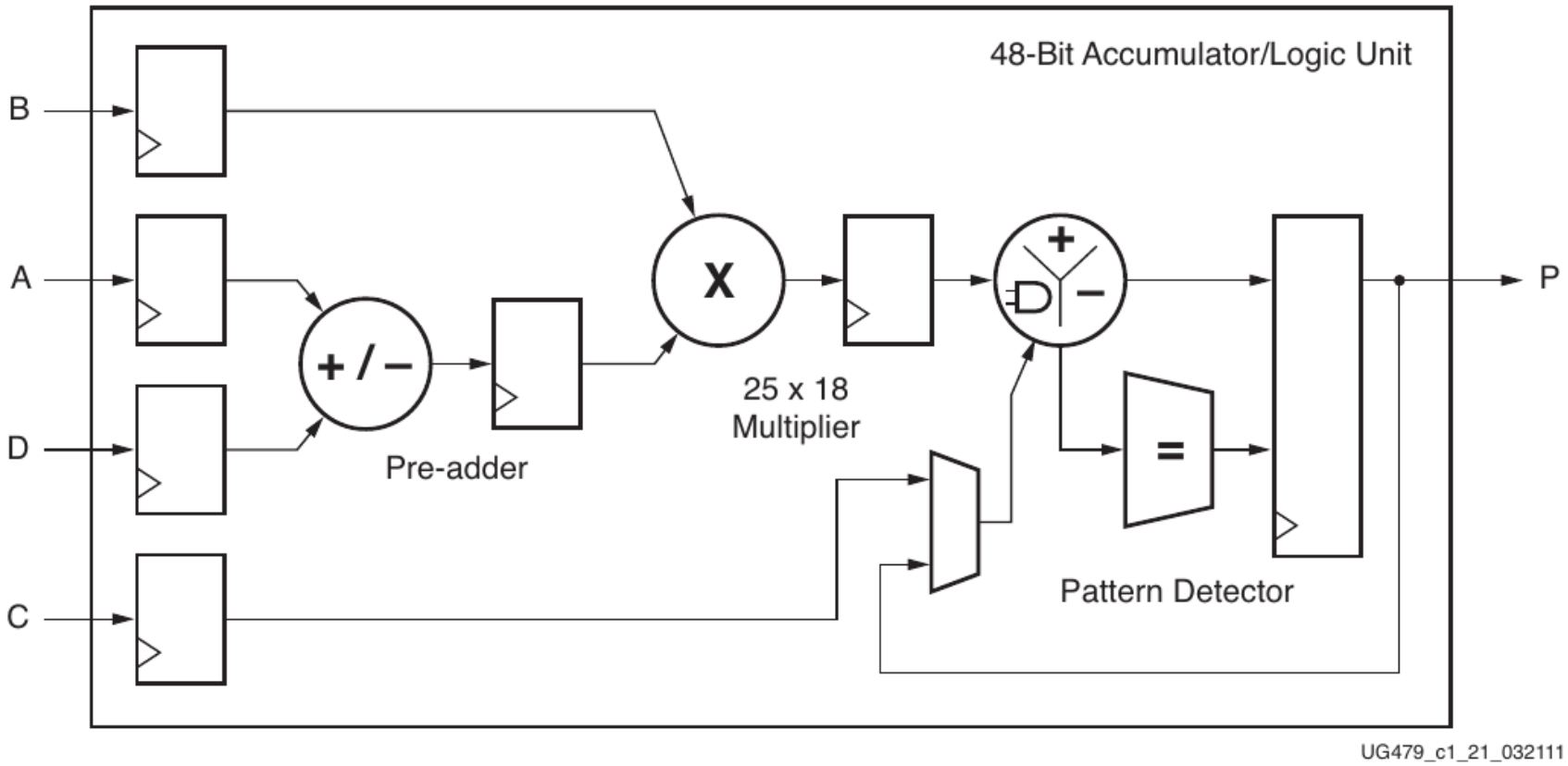
Utilisation d'un bloc dédié : le bloc DSP

Structure interne d'un FPGA : Blocs matériels

DSP : pour optimiser la multiplication

- **Mult 25 x 18**
- **Preadditionneur**
- **Accumulateur**

A7-100 : 240
VU13P : 12 288



Structure interne d'un FPGA : Blocs matériels

Comment utiliser un bloc DSP ?

=> faire confiance aux outils de synthèse…

… sans oublier qu'ils ne font pas de miracle

```
    signal a : unsigned(15 downto 0) ;
    signal b : unsigned(15 downto 0) ;
    signal c : unsigned(15 downto 0) ;
…
    …
process(clk)
  begin
        if rising_edge(clk) then
              c <= a*b ;
        end if ;
  end process ;
```

Structure interne d'un FPGA : Blocs matériels

Et la mémoire ???

- **Mémoire distribuée (utilisant les bascules)**
- **LUTRAMs : peu efficaces si fortement combinées**

BlockRAM (ref Xilinx AMD : BRAM36)

- **Jonction de 2 RAMs 18kb**
- **Ajout de logique supplémentaire pour macrofonctions (FIFO, …)**

Inconvénient : accès limités !

- **2 ports => 2 opérations max (R ou W) par bloc et par front d'horloge**
- **Chaque port a sa propre horloge**
  - **Utilisable pour passer des données d'un domaine à l'autre**

Structure interne d'un FPGA : Blocs matériels

BlockRAM : géometrie configurable :

- 16k x 1 bit
- 8k x 2 bits
- 4k x 4 bits
- 2k x 9 bits
- 1k x 18 bits
- 512 x 36 bits

Données synchrones

Registre intégré pour données / adresse

A7-100 : 135
VU13P : 2688

Structure interne d'un FPGA : Blocs matériels

Utiliser un Bloc RAM :

"inférence"

- **Déclarer un type vecteur de std_logic_vector / unsigned / …**
- **Déclarer un signal de ce type**

- **Restrictions d'usage :**
  - **Utilisation d'un seul index de lecture/écriture (simple accès)**
  - **Utilisation de deux index maximum (mémoire double accès)**

"instanciation"

- **Sous forme d'IP**
- **Macro *constructeur***

Structure interne d'un FPGA : Blocs matériels

Pour aller encore plus loin :

- **Blocs de gestion d'horloges**
  - **Génération d'horloges de fréquences différentes**
  - **Génération de phases supplémentaires**
  - **Synchro sur horloge externe**
- **Blocs de communication haute vitesse**
  - **Sur Artix 7 : débit max : 800MBps / 1200MBps (selon le "speed grade")**
  - **Sur Virtex : débit max 32GBps (sur un sous-ensemble de broches)**
- **Interfaces standard (PCIe / Ethernet)**

Structure interne d'un FPGA : Blocs matériels

Pour aller encore encore plus loin :

- **Processeur complet**
  - **Jusqu'à 4 coeurs**
  - **C'est le domaine du SoPC (system on programmable chip)**
  - **Pour la 3A / Systèmes embarqués**
- **On fait fonctionner ENSEMBLE, sur la même puce**
  - **Des éléments d'archi matérielle**
  - **Du logiciel**

Et là ? On fume quoi ?

Autres trucs complètement barrés mais possibles :

- **Des modules spécifiques permettent de manipuler la config du FPGA depuis l'archi qui tourne dessus !!!**
  - **Contrôle d'intégrité / chiffrement**
  - **Test de robustesse**
  - **Sauvegarde de l'état matériel du système**
- **Il est possible de modifier la configuration d'une partie pendant que l'autre partie fonctionne : reconfiguration dynamique partielle**
  - **Si des fonctions ne sont jamais nécessaires simultanément, pas besoin qu'elles soient disponibles simultanément : on utilise les mêmes ressources matérielles**
- **Autoreconfiguration dynamique partielle :**
  - **Toujours utile dans les satellites …**

Tous les FPGAs sont-ils faits de la même manière ?

Pour un même fondeur :

- **Même bilbiothèque de fonctions sur tous les FPGAs**
  - **Évolutions d'une génération à l'autre**
  - **Variation sur le nombre et la proportion des ressources**
- **Même stratégie de routage**
  - **Variations possibles selon la taille du circuit**
- **Possibles variations sur la techno silicium**
  - **Densité**
  - **Consommation**
  - **Timing**

=> objectif : limiter la complexité du synthétiseur

Tous les FPGAs sont-ils faits de la même manière ?

D'un fondeur à l'autre :

- **Principes fondamentaux identiques**
- **Architectures différentes**
  - **Stratégie de routage local / global différente**
- **Éléments aux optimisations différentes :**
  - **Exemples (non exhaustif)**

	Altera / Intel	Xilinx / AMD
Élément de base	Adaptive Logic Module (ALM) - 2 ou 4 LUT8 - 4 FF	Slice - 2 LUT6 - 4 FF
LUT	- 8 entrées - dissociables	- 6 entrées - 2 sorties
BRAM	- 9kbits* - clk par fonction - quad-half-ports*	- 2 x 18kbits - clk par port - dual-ports
DSP*	- 27 x 27 ou 2 x (18 x 19) - acc 64 bits - Floating point	- 27* x 18 - acc 48 bits

Tous les FPGAs sont-ils faits de la même manière ?

Optimiser son code / architecture

- Les langages (VHDL, verilog, …) sont synthétisables sur tous les FPGAs
- Optimisations génériques :
  - Construction de pipeline / machine d'état
    - Réduction du chemin critique
  - Mutualisation de données
    - Stockage en mémoires "bloc"
- Optimisations spécifiques à la cible
  - Gestion des ports d'accès à la RAM
  - Choix de géometrie pour réduire le nombre de BRAMs
  - Choix de précision pour réduire le nombre de blocs DSP

En écrivant du VHDL, il faut penser "composants"

- si on pense "ressource FPGA" le design devient optimisé pour une cible donnée

Logiciels de synthèse …

Constructeurs :

- **Altera / Intel : Quartus**
- **Xilinx / AMD : VIVADO / VITIS**
- **Lattice : Diamond**
- **ATMEL / Microchip : Libero / SmartHLS**
- **…**

Editeurs logiciel CAO :

- **Mentor Graphics**
- **CADENCE**
- **Synopsys**
- **…**

Logiciels de synthèse …

Passerelles depuis langages "classiques"

HDL coder : MATLAB / simulink → verilog / VHDL

- **Brutal, mais efficace en temps de développement**
- **Peu de contrôle (pour un elec habitué au codage VHDL/Verilog)**

Synthèse à partir de C / C++ :

- **High Level Synthesis (HLS)**
- **Complexité au niveau des I/Os**
- **Il faut guider le synthétiseur pour obtenir un résultat intéressant**
  - **Pas de salut sans une idée de l'architecture recherchée**

Tentatives de passerelle : Python → verilog / VHDL

- **Faux amis : codage de style HDL avec syntaxe et objets python**

Concept d'IP

IP : Intellectual Property

- **Module déjà codé (par quelqu'un d'autre)**

Inutile de réinventer la roue (Design For Reuse)

- **Conçu par des spécialistes**
- **Performances très prédictibles**
- **Temps conception (très) raccourci**
- **Prix variable :**
  - **En $ / € / ¥**
  - **Si c'est gratuit, c'est vous le produit !**
    - **"fidélisation" du client par un constructeur**
    - **Tentative d'éviction d'un concurent**
    - **Démonstration technologique**

Concept d'IP

IP : Intellectual Property

- **Module déjà codé (par quelqu'un d'autre)**

Inutile de réinventer la roue (Design For Reuse)

- **Conçu par des spécialistes**
- **Performances très prédictibles**
- **Temps conception (très) raccourci**
- **Prix variable**
- **Parfois inévitable**
  - **I/Os spéciales**
  - **Utilisation de ressources matérielles spécifiques**

Concept d'IP

Exemples :

- **Interface :**
  - **UART / PCI / CAN / IEEE1394 / USB / SDRAM / HDMI / …**
- **TSI :**
  - **Conversion de couleur / FFT / filtre / entrelacement / …**
- **Communication :**
  - **Ethernet MAC / Wi-Fi / modulation / …**
  - **chiffrement / compression / correction d'erreur …**
- **Système :**
  - **Processeur / Contrôleur de bus / Périphérique / …**
- **Calcul :**
  - **Diviseur / cordic / FPU / …**

Les SoPCs

Système sur puce programmable

- **On utilise le FPGA pour implanter (au moins) un processeur**
  - **Calcul violent / bas niveau => partie logique du FPGA**
  - **Calcul itératif / haut niveau => logiciel sur le processeur**
  - **Dans le FPGA, les éléments sont au plus proche**
    - **Grande efficacité dans les échanges**

Les SoPCs :

Processeur "softcore" :

- **Codé en verilog ou VHDL**

On peut dimensionner le processeur aux besoins logiciels

- **8, 16, 32, 64 bits ?**
- **Mode FSM / pipeline**
- **Complexité de ALU / FPU**

Possible aussi d'ajouter des instructions/fonctions spécifiques

- **Nombre et latence des Interruptions**
- **MACC en 1 instruction**
- **…**

Multiples compromis possibles

Les SoPCs

Processeur "hardcore"

- **Module précâblé**
- **Très puissant / compact (non limité par la techno FPGA)**
- **Économie de silicium pour les besoins standards**
- **Architecture courante et reconnue**
  - **Gérés par GCC**
  - **OS disponibles**

Les SoPCs

Inconvénients :

- **Fini le parallèle / concurent**
  - **Interconnexion par bus partagé**
- **Dépendance TOTALE du fournisseur / des outils**
  - **Xilinx / AMD : migration sans préavis de PowerPC vers ARM (2011)**
  - **Softcore : pas d'autre mainteneurs**
    - **Xilinx / AMD : PicoBlaze / MicroBlaze**
    - **Altera / Intel : Nios / Nios II / Nios V**
- **Complexité de la chaîne de développement**
  - **Codesign / Cosimulation**