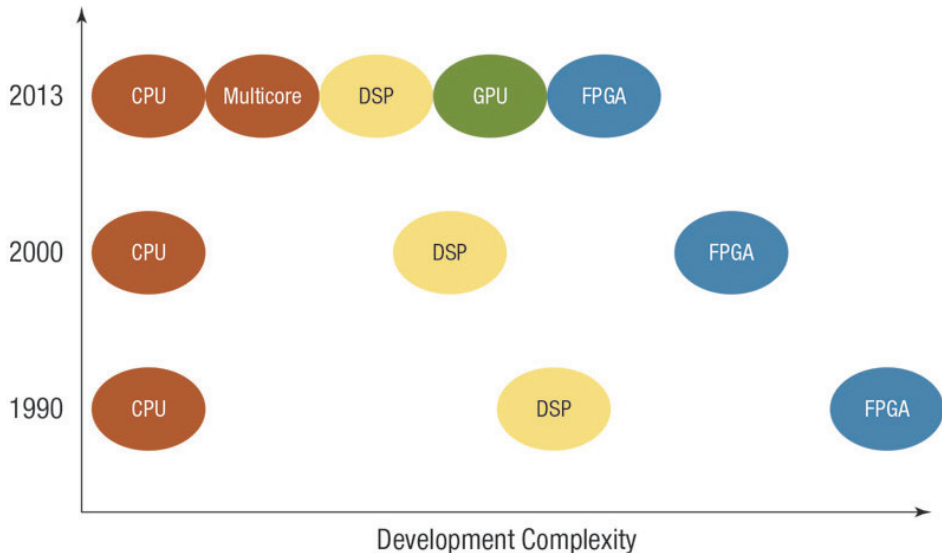


# Syst de Télécom Embarqués

## Chapitre II: Technologie des circuits FPGA



# Introduction

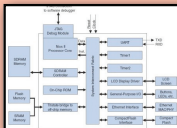
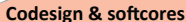


# Introduction

## Implantation logicielle

```

1 my $showname, $showname_bname := (0,0);
2 my $formatstr = $format{"show"} . "\n";
3
4 return if ( !check_title_werr($title));
5
6 my $i = 0;
7 while ($show) {
8     my $dir = $show;
9     my $sarg = $title_bname;
10    my $showname_bname = $showname;
11    if ($show) {
12        if ($sarg =~ /^-b$/i) {
13            $showname_bname = $showname;
14        }
15        elsif ($sarg =~ /^-n$/i) {
16            $showname_bname = $showname;
17        }
18        elsif ($sarg =~ /^-i$/i) {
19            $showname_bname = $showname;
20        }
21        $showname_bname = $showname;
22    }
23    my $dir = $show;
24    my $sarg = $title_bname;
25    my $showname_bname = $showname;
26    if ($show) {
27        if ($sarg =~ /^-b$/i) {
28            $showname_bname = $showname;
29        }
30        elsif ($sarg =~ /^-n$/i) {
31            $showname_bname = $showname;
32        }
33        elsif ($sarg =~ /^-i$/i) {
34            $showname_bname = $showname;
35        }
36        $showname_bname = $showname;
37    }
38    my $dir = $show;
39    my $sarg = $title_bname;
40    my $showname_bname = $showname;
41    if ($show) {
42        if ($sarg =~ /^-b$/i) {
43            $showname_bname = $showname;
44        }
45        elsif ($sarg =~ /^-n$/i) {
46            $showname_bname = $showname;
47        }
48        elsif ($sarg =~ /^-i$/i) {
49            $showname_bname = $showname;
50        }
51        $showname_bname = $showname;
52    }
53    my $dir = $show;
54    my $sarg = $title_bname;
55    my $showname_bname = $showname;
56    if ($show) {
57        if ($sarg =~ /^-b$/i) {
58            $showname_bname = $showname;
59        }
60        elsif ($sarg =~ /^-n$/i) {
61            $showname_bname = $showname;
62        }
63        elsif ($sarg =~ /^-i$/i) {
64            $showname_bname = $showname;
65        }
66        $showname_bname = $showname;
67    }
68    my $dir = $show;
69    my $sarg = $title_bname;
70    my $showname_bname = $showname;
71    if ($show) {
72        if ($sarg =~ /^-b$/i) {
73            $showname_bname = $showname;
74        }
75        elsif ($sarg =~ /^-n$/i) {
76            $showname_bname = $showname;
77        }
78        elsif ($sarg =~ /^-i$/i) {
79            $showname_bname = $showname;
80        }
81        $showname_bname = $showname;
82    }
83    my $dir = $show;
84    my $sarg = $title_bname;
85    my $showname_bname = $showname;
86    if ($show) {
87        if ($sarg =~ /^-b$/i) {
88            $showname_bname = $showname;
89        }
90        elsif ($sarg =~ /^-n$/i) {
91            $showname_bname = $showname;
92        }
93        elsif ($sarg =~ /^-i$/i) {
94            $showname_bname = $showname;
95        }
96        $showname_bname = $showname;
97    }
98    my $dir = $show;
99    my $sarg = $title_bname;
100    my $showname_bname = $showname;
101    if ($show) {
102        if ($sarg =~ /^-b$/i) {
103            $showname_bname = $showname;
104        }
105        elsif ($sarg =~ /^-n$/i) {
106            $showname_bname = $showname;
107        }
108        elsif ($sarg =~ /^-i$/i) {
109            $showname_bname = $showname;
110        }
111        $showname_bname = $showname;
112    }
113    my $dir = $show;
114    my $sarg = $title_bname;
115    my $showname_bname = $showname;
116    if ($show) {
117        if ($sarg =~ /^-b$/i) {
118            $showname_bname = $showname;
119        }
120        elsif ($sarg =~ /^-n$/i) {
121            $showname_bname = $showname;
122        }
123        elsif ($sarg =~ /^-i$/i) {
124            $showname_bname = $showname;
125        }
126        $showname_bname = $showname;
127    }
128    my $dir = $show;
129    my $sarg = $title_bname;
130    my $showname_bname = $showname;
131    if ($show) {
132        if ($sarg =~ /^-b$/i) {
133            $showname_bname = $showname;
134        }
135        elsif ($sarg =~ /^-n$/i) {
136            $showname_bname = $showname;
137        }
138        elsif ($sarg =~ /^-i$/i) {
139            $showname_bname = $showname;
140        }
141        $showname_bname = $showname;
142    }
143    my $dir = $show;
144    my $sarg = $title_bname;
145    my $showname_bname = $showname;
146    if ($show) {
147        if ($sarg =~ /^-b$/i) {
148            $showname_bname = $showname;
149        }
150        elsif ($sarg =~ /^-n$/i) {
151            $showname_bname = $showname;
152        }
153        elsif ($sarg =~ /^-i$/i) {
154            $showname_bname = $showname;
155        }
156        $showname_bname = $showname;
157    }
158    my $dir = $show;
159    my $sarg = $title_bname;
160    my $showname_bname = $showname;
161    if ($show) {
162        if ($sarg =~ /^-b$/i) {
163            $showname_bname = $showname;
164        }
165        elsif ($sarg =~ /^-n$/i) {
166            $showname_bname = $showname;
167        }
168        elsif ($sarg =~ /^-i$/i) {
169            $showname_bname = $showname;
170        }
171        $showname_bname = $showname;
172    }
173    my $dir = $show;
174    my $sarg = $title_bname;
175    my $showname_bname = $showname;
176    if ($show) {
177        if ($sarg =~ /^-b$/i) {
178            $showname_bname = $showname;
179        }
180        elsif ($sarg =~ /^-n$/i) {
181            $showname_bname = $showname;
182        }
183        elsif ($sarg =~ /^-i$/i) {
184            $showname_bname = $showname;
185        }
186        $showname_bname = $showname;
187    }
188    my $dir = $show;
189    my $sarg = $title_bname;
190    my $showname_bname = $showname;
191    if ($show) {
192        if ($sarg =~ /^-b$/i) {
193            $showname_bname = $showname;
194        }
195        elsif ($sarg =~ /^-n$/i) {
196            $showname_bname = $showname;
197        }
198        elsif ($sarg =~ /^-i$/i) {
199            $showname_bname = $showname;
200        }
201        $showname_bname = $showname;
202    }
203    my $dir = $show;
204    my $sarg = $title_bname;
205    my $showname_bname = $showname;
206    if ($show) {
207        if ($sarg =~ /^-b$/i) {
208            $showname_bname = $showname;
209        }
210        elsif ($sarg =~ /^-n$/i) {
211            $showname_bname = $showname;
212        }
213        elsif ($sarg =~ /^-i$/i) {
214            $showname_bname = $showname;
215        }
216        $showname_bname = $showname;
217    }
218    my $dir = $show;
219    my $sarg = $title_bname;
220    my $showname_bname = $showname;
221    if ($show) {
222        if ($sarg =~ /^-b$/i) {
223            $showname_bname = $showname;
224        }
225        elsif ($sarg =~ /^-n$/i) {
226            $showname_bname = $showname;
227        }
228        elsif ($sarg =~ /^-i$/i) {
229            $showname_bname = $showname;
230        }
231        $showname_bname = $showname;
232    }
233    my $dir = $show;
234    my $sarg = $title_bname;
235    my $showname_bname = $showname;
236    if ($show) {
237        if ($sarg =~ /^-b$/i) {
238            $showname_bname = $showname;
239        }
240        elsif ($sarg =~ /^-n$/i) {
241            $showname_bname = $showname;
242        }
243        elsif ($sarg =~ /^-i$/i) {
244            $showname_bname = $showname;
245        }
246        $showname_bname = $showname;
247    }
248    my $dir = $show;
249    my $sarg = $title_bname;
250    my $showname_bname = $showname;
251    if ($show) {
252        if ($sarg =~ /^-b$/i) {
253            $showname_bname = $showname;
254        }
255        elsif ($sarg =~ /^-n$/i) {
256            $showname_bname = $showname;
257        }
258        elsif ($sarg =~ /^-i$/i) {
259            $showname_bname = $showname;
260        }
261        $showname_bname = $showname;
262    }
263    my $dir = $show;
264    my $sarg = $title_bname;
265    my $showname_bname = $showname;
266    if ($show) {
267        if ($sarg =~ /^-b$/i) {
268            $showname_bname = $showname;
269        }
270        elsif ($sarg =~ /^-n$/i) {
271            $showname_bname = $showname;
272        }
273        elsif ($sarg =~ /^-i$/i) {
274            $showname_bname = $showname;
275        }
276        $showname_bname = $showname;
277    }
278    my $dir = $show;
279    my $sarg = $title_bname;
280    my $showname_bname = $showname;
281    if ($show) {
282        if ($sarg =~ /^-b$/i) {
283            $showname_bname = $showname;
284        }
285        elsif ($sarg =~ /^-n$/i) {
286            $showname_bname = $showname;
287        }
288        elsif ($sarg =~ /^-i$/i) {
289            $showname_bname = $showname;
290        }
291        $showname_bname = $showname;
292    }
293    my $dir = $show;
294    my $sarg = $title_bname;
295    my $showname_bname = $showname;
296    if ($show) {
297        if ($sarg =~ /^-b$/i) {
298            $showname_bname = $showname;
299        }
300        elsif ($sarg =~ /^-n$/i) {
301            $showname_bname = $showname;
302        }
303        elsif ($sarg =~ /^-i$/i) {
304            $showname_bname = $showname;
305        }
306        $showname_bname = $showname;
307    }
308    my $dir = $show;
309    my $sarg = $title_bname;
310    my $showname_bname = $showname;
311    if ($show) {
312        if ($sarg =~ /^-b$/i) {
313            $showname_bname = $showname;
314        }
315        elsif ($sarg =~ /^-n$/i) {
316            $showname_bname = $showname;
317        }
318        elsif ($sarg =~ /^-i$/i) {
319            $showname_bname = $showname;
320        }
321        $showname_bname = $showname;
322    }
323    my $dir = $show;
324    my $sarg = $title_bname;
325    my $showname_bname = $showname;
326    if ($show) {
327        if ($sarg =~ /^-b$/i) {
328            $showname_bname = $showname;
329        }
330        elsif ($sarg =~ /^-n$/i) {
331            $showname_bname = $showname;
332        }
333        elsif ($sarg =~ /^-i$/i) {
334            $showname_bname = $showname;
335        }
336        $
```



## Architecture dédiée



DSP

TMS320C674x



### Circuit ASIC préconçu et fondu par un tiers



ASIC: Application Specific Integrated Circuit

# Classification des circuits reconfigurables

## Différentes familles de circuit (re)programmable

- PLD (Programmable Logic Device): circuits de 100-200 portes
- CPLD (Complex Programmable Logic Device): circuits de 5-100 K portes
- FPGA (Field Programmable Gate Array): circuits atteignant  $+10^6$  portes

## Différentes techniques de programmation

- Cellules à fusible ou à antifusible
- Mémoire effaçable électriquement (EEPROM ou Flash EPROM)
- Mémoire SRAM



# Classification des circuits reconfigurables

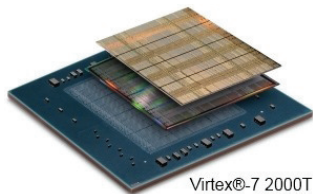
## Différentes familles de circuit (re)programmable

- PLD (Programmable Logic Device): circuits de 100-200 portes
- CPLD (Complex Programmable Logic Device): circuits de 5-100 K portes
- FPGA (Field Programmable Gate Array): circuits atteignant  $+10^6$  portes

## Différentes techniques de programmation

- Cellules à fusible ou à antifusible
- Mémoire effaçable électriquement (EEPROM ou Flash EPROM)
- Mémoire SRAM

# Classification des circuits reconfigurables



Virtex®-7 2000T

## Différentes familles de circuit (re)programmable

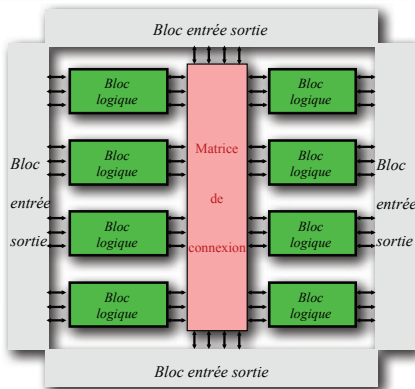
	Antifusible	EPROM	SRAM
Facilité de programmation	faible	importante	importante
Densité	faible	élevé	très élevé
Reprogrammabilité	non	oui	oui

# Les circuits configurables de type CPLD

- Un CPLD se présente comme un ensemble de fonctions de type PLD reliées à l'aide d'une matrice de connexion
- Intérêt: Développement de circuits numériques de faible complexité (5000 à 100000 portes)

## Exemple

- Altera MAXII: 240 à 2200 éléments logiques
- Xilinx CoolRunner2: 32 à 512 macrocells

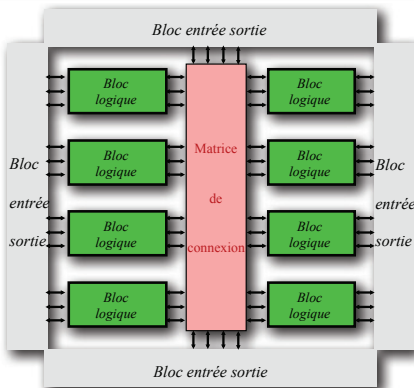


# Les circuits configurables de type CPLD

- Un CPLD se présente comme un ensemble de fonctions de type PLD reliées à l'aide d'une matrice de connexion
- Intérêt: Développement de circuits numériques de faible complexité (5000 à 100000 portes)

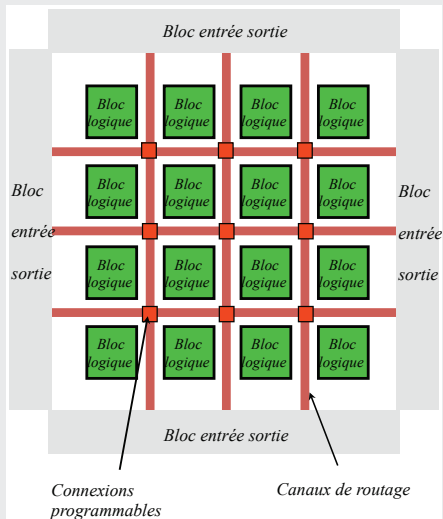
## Exemple

- Altera MAXII: 240 à 2200 éléments logiques
- Xilinx CoolRunner2: 32 à 512 macrocells

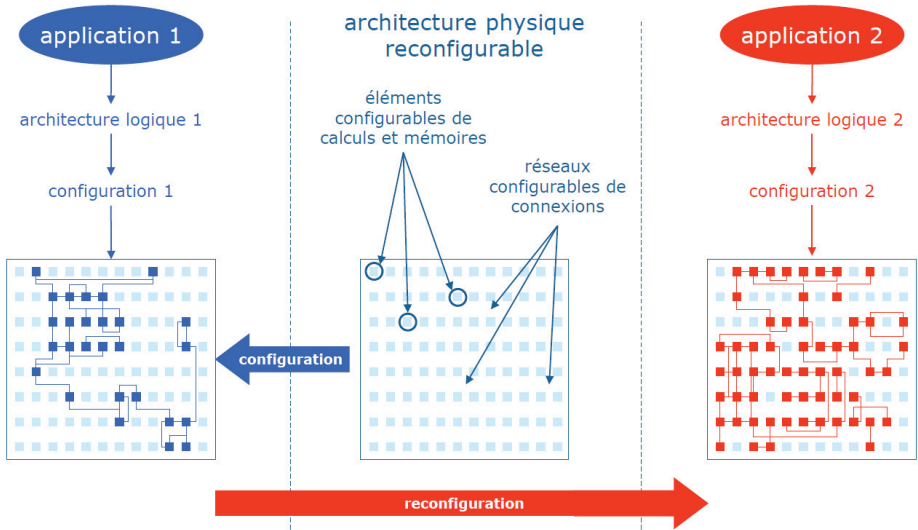


# Les circuits configurables de type FPGA

- Les FPGAs comprennent:
  - De nombreux modules logiques
  - Un réseau d'interconnexions entre les modules logiques
- Deux familles de FPGA :
  - FPGA de type SRAM (reprogrammable)
  - FPGA de type antifusible (non reprogrammable)
- Les FPGAs ont des capacités d'intégration élevées



# Intérêt des circuits (re)configurables de type FPGA



# Circuits (re)configurables VS ASICs

Caractéristiques	Circuits Configurables		ASIC	
	FPGA SRAM	CPLD	Standard Cell	Full Custom
Densité (portes/m <sup>2</sup> )	1500 à 6500 0,13µm 90 nm	Faible 0,18 µm	Grande 90 nm	Grande 90 nm
Performance (Hz, Bits/s)	500 MHz 10 Gbits/s	200 MHz	qq Ghz 12 Gbits/s	qq Ghz 20 Gbits/s
Consommation (Watt)	Grande	Grande	Faible	Faible
Flexibilité	Grande	Grande	Aucune	Aucune
Temps de conception	Court	Court	Long	Très Long
Utilisation des outils	Facile	Facile	Complexe	Complexe
Cout de conception	Faible	Faible	Elevé	Très élevé
Volume de production	Faible et moyen	Faible et moyen	> 1 000 000 de pièces	> 1 000 000 de pièces

# Evolution

## Évolution de la technologie FPGA



0.6  $\mu$

trois couches  
de métal

Technologie :

Taille relative :

100%

Année :

1994



0.5  $\mu$

trois couches  
de métal

78%

1995



0.35  $\mu$

quatre couches  
de métal

32%

1996



0.25  $\mu$

cinq couches  
de métal

19%

1998



0.18  $\mu$

six couches  
de métal

12%

1999

En 2012

Technologie 0.020  $\mu$ m avec 11 couches de cuivre



# Classement des vendeurs de FPGA

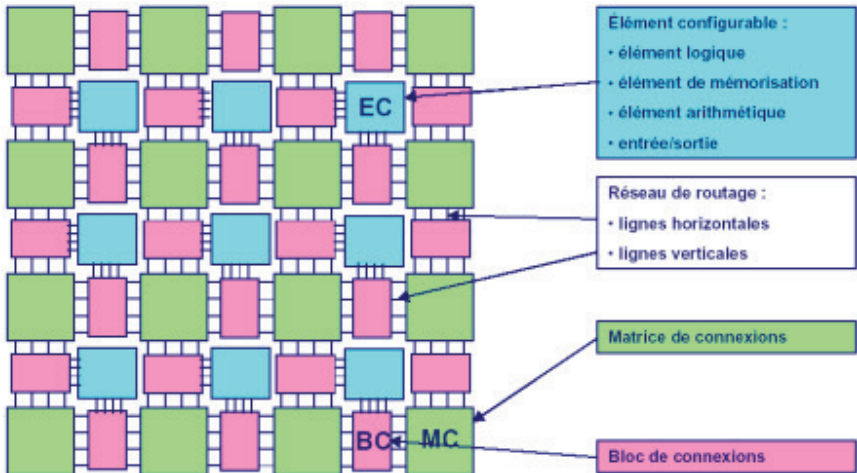
Rank 2007	Rank 2008	Company
1	1	Xilinx
2	2	Altera
3	3	Lattice Semiconductor
4	4	Actel
6	5	QuickLogic
5	6	Cypress Semiconductor
7	7	Atmel
8	8	Chengdu Sino Microelectronics System

Source: Gartner

# Architecture interne des circuits FPGA

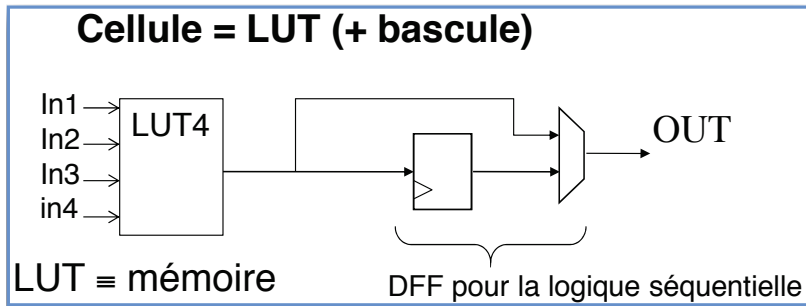
# FPGA de chez Xilinx

# Type d'architecture employé chez Xilinx



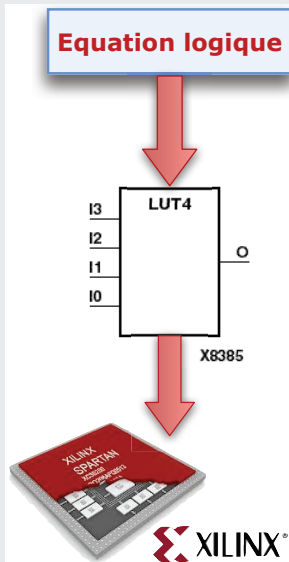
# Élément logique de base (cellule élémentaire)

- Cellule de base, composée d'une LUT4 à 4 entrées et d'une Flip-Flop
- Une LUT4 à 4 entrées qui correspond à une mémoire de 16 bits

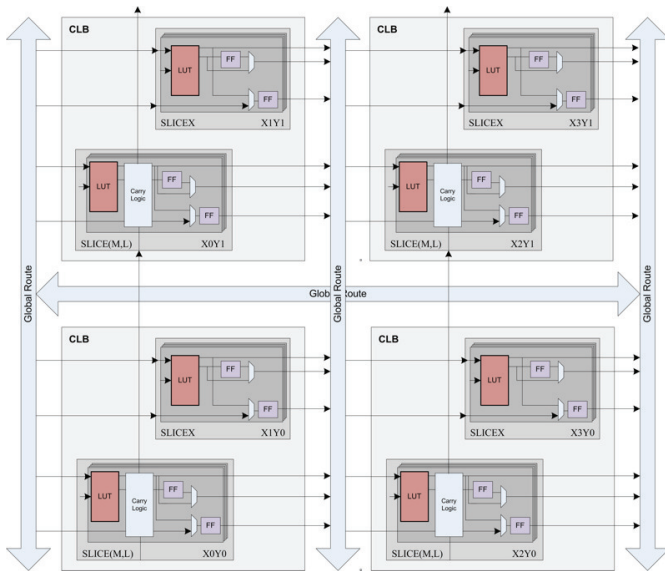


# Élément logique de base (cellule élémentaire)

- Il faut précalculer le contenu de la mémoire à partir de la fonction logique à intégrer
- Peu importe la complexité de la fonction logique (4 entrées max.)  $\Rightarrow$  1 LUT4



# Utilité de la cellule de base



## Limitation ?

- Les équations logiques à  $n > 4$  entrées

## Solution

- Nécessaire d'utiliser plusieurs LUT4 interconnectées les unes avec les autres



## Limitation ?

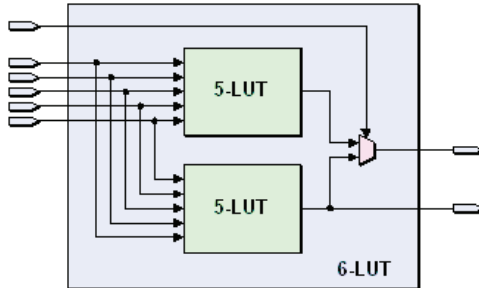
- Les équations logiques à  $n > 4$  entrées

## Solution

- Nécessaire d'utiliser plusieurs LUT4 interconnectées les unes avec les autres

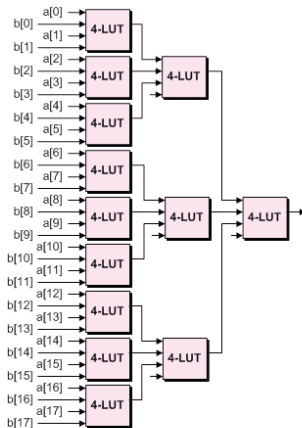
# Evolution de la taille des LUTs chez Xilinx

- Le passage des LUT4 (Virtex-4) aux LUT6 (Virtex-5,6,7) a été réalisée en factorisant 2 LUTs de tailles inférieures

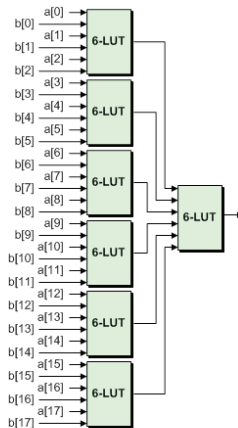


# Evolution de la taille des LUTs chez Xilinx

- Réduction du nombre d'étages de logique (LUT) à traverser pour implanter des fonctions complexes



(a) Virtex-4: 13 LUTs, 3 levels

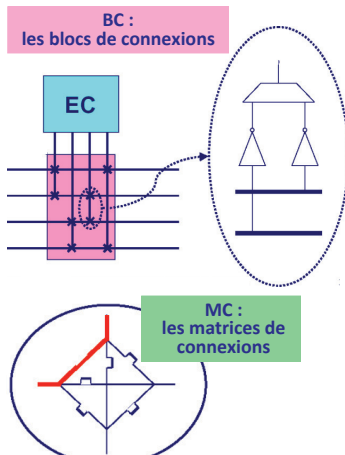


(b) Virtex-5: 7 LUTs, 2 levels

# Interconnexion des éléments configurables



- Les blocs de connexions assurent la connexion des éléments configurables
- Les matrices de connexions assurent la connexion des blocs de connexions



# Interconnexion des éléments configurables

## Ressources de routage

24 Horizontal Long Lines 24 Vertical Long Lines	
120 Horizontal Hex Lines 120 Vertical Hex Lines	
40 Horizontal Double Lines 40 Vertical Double Lines	
16 Direct Connections (total in all four directions)	
8 Fast Connects	

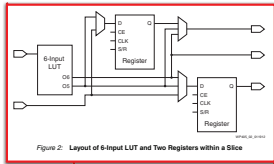
# Interconnexion des éléments configurables

Il existe également des ressources de routage dédiées aux horloges et aux retenues

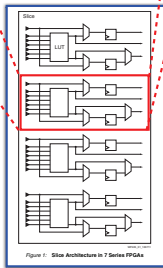
La structure interne des FPGA est plus complexe

- Ajout de ressources spécialisées dans le FPGA
- Bloc mémoire (RAM 18/36kbits)
- Blocs DSP
- D'autres blocs dédiés peuvent être disponibles: PLL, Ethernet ...etc

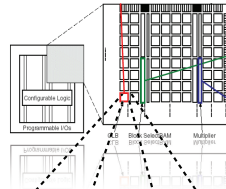
# Terminologie dans la structure des FPGA de chez Xilinx



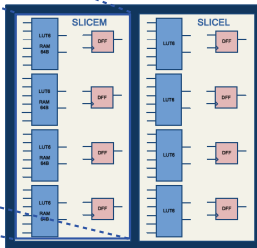
*Element  
de base*



*Slice  
(M ou L)*



*Xilinx FPGA  
structure*



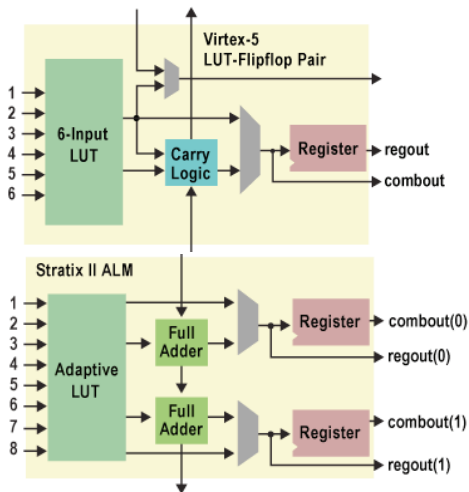
*Configurable  
Logic Block  
(CLB)*

# FPGA de Altera



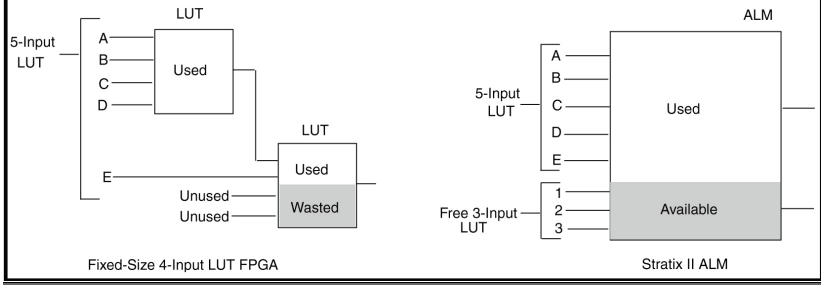
# Décomposition de l'élément logique configurable chez Altera

- L'association chez Xilinx d'une LUT et d'une flip-flop constitue une cellule de base
- Chez Altera, la cellule de base est plus complexe: Adaptive LUT + 2 registers + 2 FA
- Adaptive LUT: Une LUT8 à plusieurs sorties qui peut être découpée en LUTs de tailles inférieures pour maximiser l'occupation



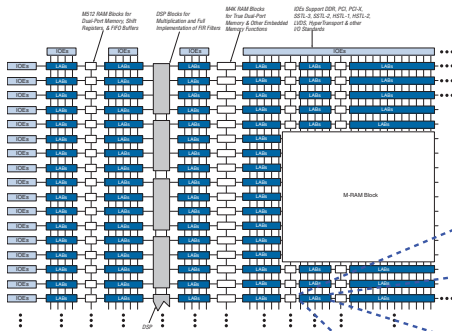
# Intérêt de la structure modulaire de la LUT8 de chez Altera

Figure 5. Five-Variable Function Implementation Comparison Between a Fixed-Size Four-Input LUT FPGA & Stratix II Devices

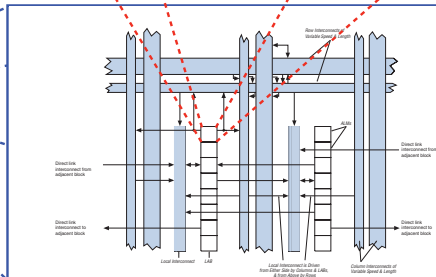
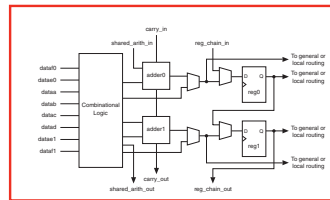


La LUT8 de chez Altera permet un meilleur taux d'utilisation des ressources versus les LUT4 et LUT6 (en théorie)

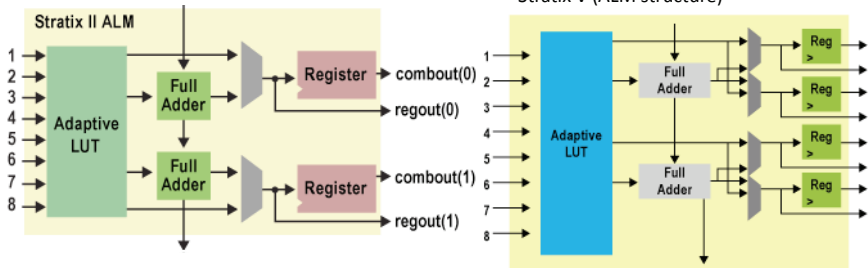
# Architecture du Stratix II de chez Altera



1 LAB = 8 ALM



# Evolution de l'ALM (Stratix II vs Stratix V)

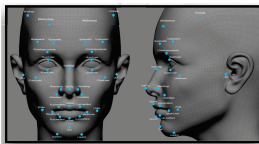


Augmentation du nombre de registre au sein de l'ALM ( $2 \Rightarrow 4$ )

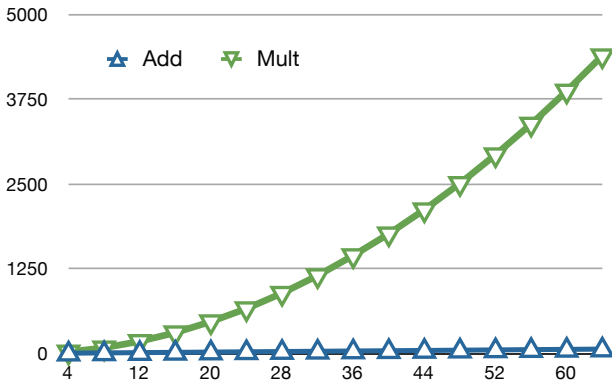
## Les blocs dédiés: Etude des DSP blocs

# Contexte général

- Les FPGA sont utilisés pour réaliser des calculs intensifs
- Calcul scientifique, traitement vidéo, communication numérique, traitement du signal, etc
- Ces applications sont consommatrices d'opérations arithmétiques (mult, mac, etc.)
- Une forte complexité calculatoire implique beaucoup de ressources de calcul
- Les structures à base de LUTs sont inefficaces

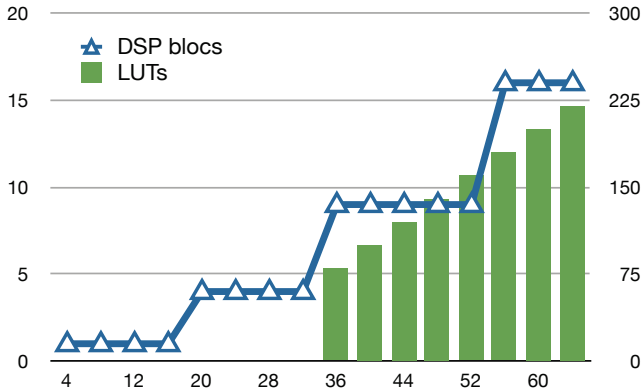


# Virtex-4



Implanter des opérations arithmétiques à l'aide de LUTs devient vite inefficace. Cela est dû au nombre des LUTs utilisées et aux délais introduits par le routage

# Evaluation des performances des blocs DSP48 (multiplication)





## Les blocs DSP48 de Xilinx (Virtex)

- Développé pour optimiser les applications fortement calculatoires
- Fonctionne jusqu'à 600MHz. Multiplieur 25x18 et accumulateur 48bits

## Les blocs DSP de chez Altera

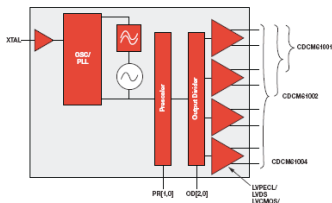
- Les blocs DSP sont basés sur des multiplieurs 18x18
- 4 multiplieurs par bloc DSP

# Les blocs mémoires Spartan-6

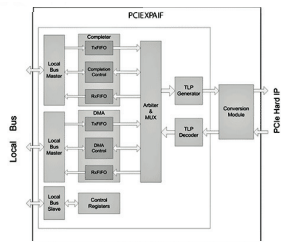
- Les DSP48 ont été introduits pour augmenter l'efficacité calculatoire
- Les blocs RAM ont été ajoutés pour permettre une mémorisation efficace des informations
- Ils sont répartis en colonne dans la matrice du FPGA
- Leur taille varie en fonction des FPGAs ainsi que la capacité totale
- La taille des blocs mémoire du Virtex-6 est 2 fois supérieure à celle du Spartan-6

# Autres ressources

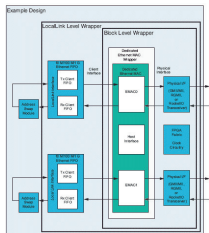
## Horloge (PLL)



## Interface PCIe



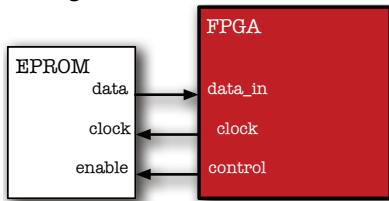
## Interface Ethernet



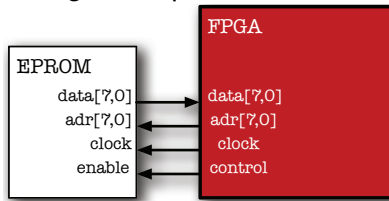
# Les modes de configuration d'un FPGA

- Le FPGA en mode Master
- Le FPGA est maitre de sa configuration
- Au démarrage, il vient charger les données à partir d'une mémoire externe
- La mémoire (EPROM) contient le fichier bitstream
- Le chargement du bitstream peut être série ou parallèle
- Mode de configuration utilisé en production

## Chargement série des données



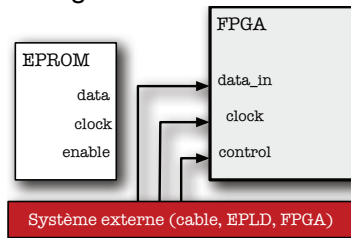
## Chargement parallèle des données



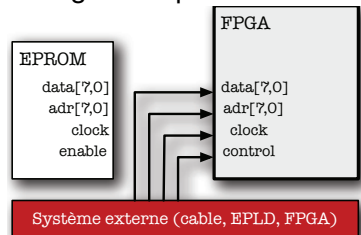
# Les modes de configuration d'un FPGA

- Le FPGA en mode Slave
- Le FPGA subit la configuration
- Au cours du temps un acteur extérieur configure le FPGA
- Le bitstream est transmis depuis l'acteur extérieur vers le FPGA
- Mode de configuration utilisé en phase de développement

## Chargement série des données



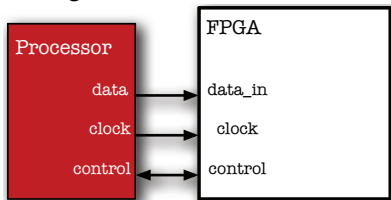
## Chargement parallèle des données



# Les modes de configuration d'un FPGA

- Le FPGA en mode Périphérique
- Le FPGA est vu comme un périphérique du système embarqué dans lequel il est intégré
- Au cours du temps le processeur configure le FPGA
- Le bitstream est transmis du processeur vers le FPGA

## Chargement série des données



## Chargement parallèle des données

