



Circuits FPGA et Synthèse logique

Principes et flot de développement

Tarik Graba
Année scolaire 2024/2025



Plan

Les FPGAs

- La Look Up Table (LUT)
- Les blocs logiques
- L'interconnect
- Les hardblocks

Flot de développement FPGA

- La synthèse
- L'inférence

Spécificité des FPGAs

- Modélisation et inférence des mémoires
- Logique synchrone et FPGA

Annexes : Le Cyclonell

les FPGAs

Field Programmable Gate Array

- Matrice de bloc logiques programmables après la fabrication.
- Xilinx (maintenant division d'AMD) produit le premier FPGA moderne en 1984
 - 1984 XC2064 64 blocs logiques
 - 2023 Virtex Ultrascale+ autour de 10 millions de blocs logiques

Lectures :

- Les premiers chapitres de : *The Design Warrior's Guide to FPGAs*, Clive Maxfield, 2004
- *Reverse-engineering the First FPGA Chip Xilinx XC2064*

les FPGAs

Qui les produits ?

Dans l'ordre :

- Les deux grands :
 - **Xilinx**/AMD et **Altera**/Intel FPGA
- Les historiques
 - Lattice, QuickLogic, Microsemi/Actel, Microchip/Atmel
- Les autres
 - Achronix, Cowin, Cologne Chip...

Pour quelles applications ?

- de l'embarqué au data-center...

Plan

Les FPGAs

- La Look Up Table (LUT)

- Les blocs logiques

- L'interconnect

- Les hardblocks

Flot de développement FPGA

- La synthèse

- L'inférence

Spécificité des FPGAs

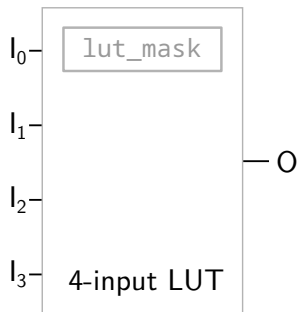
- Modélisation et inférence des mémoires

- Logique synchrone et FPGA

Annexes : Le Cyclonell

La LUT

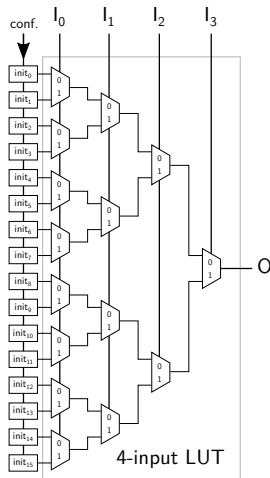
Le principe



- Un bloc matériel permettant d'implémenter toutes les fonctions combinatoires à n entrées.
 - configurable en chargeant la table de vérité (le masque de la LUT)
- Implémenter comme une mémoire dans laquelle l'entrée est l'adresse
- Plusieurs technologies
 - SRAM pour les FPGA standards
 - FLASH, anti-fusibles...
- configuration en série pour réduire la complexité
 - d'où le terme *bitstream* pour désigner le fichier de configuration

La LUT

Le principe



- Un bloc matériel permettant d'implémenter toutes les fonctions combinatoires à n entrées.
 - configurable en chargeant la table de vérité (le maque de la LUT)
- Implémenter comme une mémoire dans laquelle l'entrée est l'adresse
- Plusieurs technologies
 - SRAM pour les FPGA standards
 - FLASH, anti-fusibles...
- configuration en série pour réduire la complexité
 - d'où le terme *bitstream* pour désigner le fichier de configuration

La LUT

La taille idéale (nombre d'entrées) ?

- Maximiser la densité (fonctions/surface)
 - plus grande est la lut, plus compacte elle sera
- Permettre d'implémenter le maximum de fonctions logiques
 - combien de fonctions à plus de quelques entrées (2,3,4 ?)
- Longtemps, l'optimum était des lut à 4 entrées.
- depuis quelques générations on a des structures plus complexes avec 5 ou 6 entrées
 - des lut-5 ou 6 partielles

Voir les travaux de Jonathan Rose :

- J.S. Rose, R.J. Francis, D. Lewis, and P. Chow, "Architecture of Field-Programmable Gate Arrays : The Effect of Logic Block Functionality on Area Efficiency," IEEE JSSC, Vol. 25 No. 5, October 1990, pp. 1217-1225
- E. Ahmed and J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density," in IEEE Trans. on VLSI, Vol 12, No. 3, March 2004, pp. 288-298

Plan

Les FPGAs

La Look Up Table (LUT)

Les blocs logiques

L'interconnect

Les hardblocks

Flot de développement FPGA

La synthèse

L'inférence

Spécificité des FPGAs

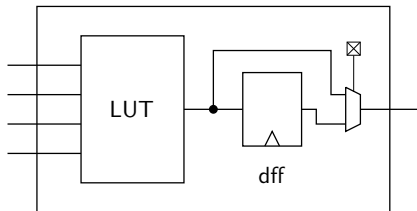
Modélisation et inférence des mémoires

Logique synchrone et FPGA

Annexes : Le Cyclonell

La blocs logiques

Plus que la logique combinatoire

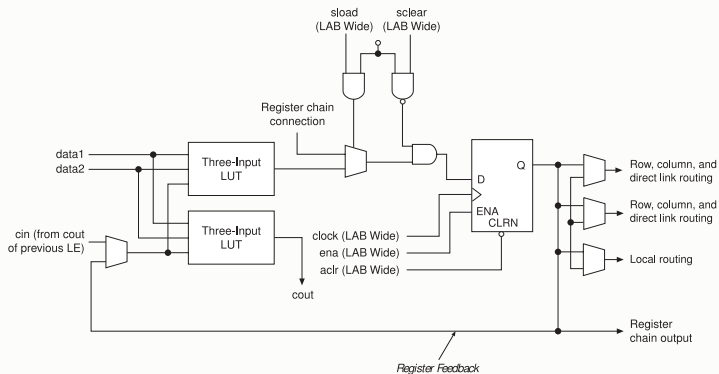


- Des registres en plus de la LUT
 - permettre d'implémenter de la logique séquentielle
- Configurable pour permettre de fonctionner en mode combinatoire ou séquentiel
 - la configuration est chaînée avec celle des LUTs (même *bitstream*)

La blocs logiques

Les modes de configuration

Configuration en mode Arithmétique



- Mode arithmétique
 - pour l'addition
 - calcul somme et retenue
 - propagation de proche en proche

Plan

Les FPGAs

La Look Up Table (LUT)

Les blocs logiques

L'interconnect

Les hardblocks

Flot de développement FPGA

La synthèse

L'inférence

Spécificité des FPGAs

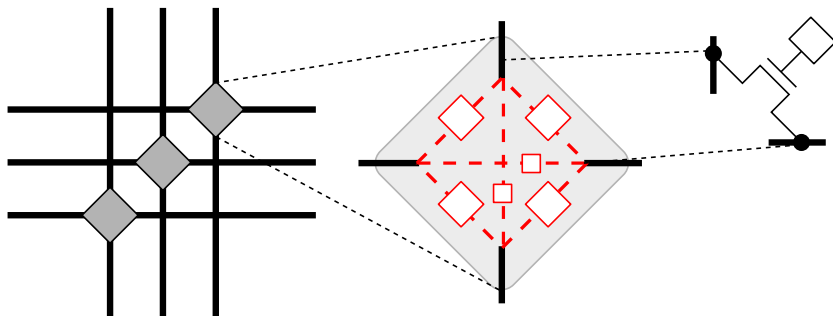
Modélisation et inférence des mémoires

Logique synchrone et FPGA

Annexes : Le CyclonII

L'interconnect

Les switch-boxes

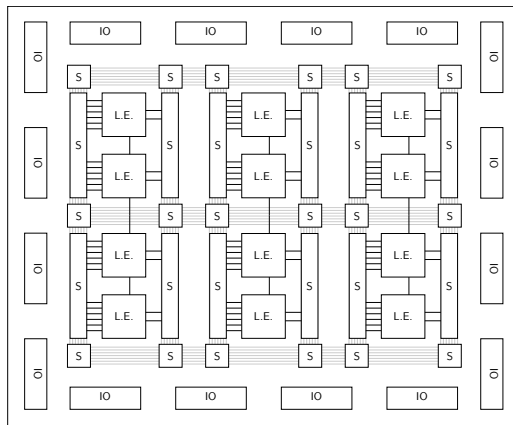


- Aiguillages configurables
- Un point mémoire par point contrôlable
 - partie du *bitstream*

L'interconnect

La hiérarchisation

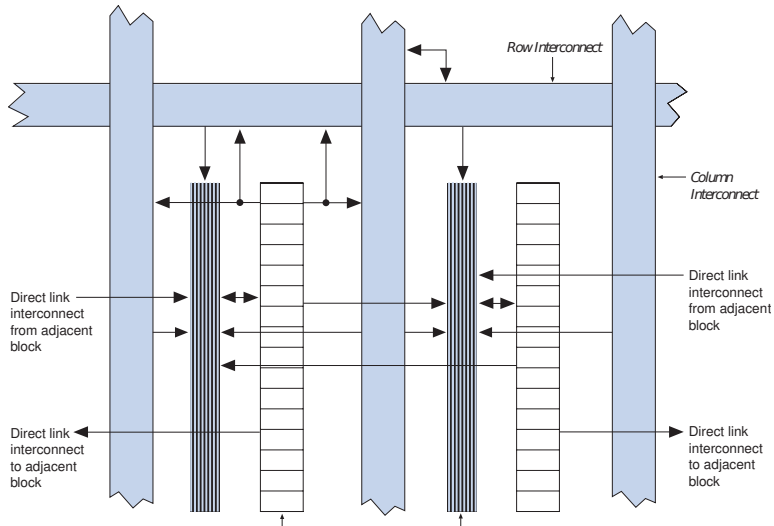
- Routage local vs routage global
- Clusters (groupements)
- Routes favorisées
 - Arithmétique, horloges...



L'interconnect

Exemple : Logic Area Block (LAB)

■ Organisation hiérarchique des cellules logiques du Cyclonell



Plan

Les FPGAs

La Look Up Table (LUT)

Les blocs logiques

L'interconnect

Les hardblocks

Flot de développement FPGA

La synthèse

L'inférence

Spécificité des FPGAs

Modélisation et inférence des mémoires

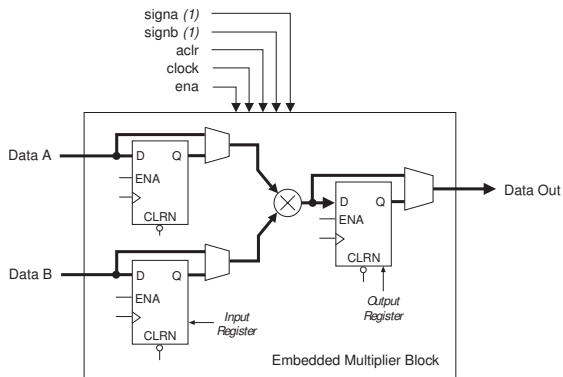
Logique synchrone et FPGA

Annexes : Le CyclonII

Les hardblocks

Les multiplieurs/DSP

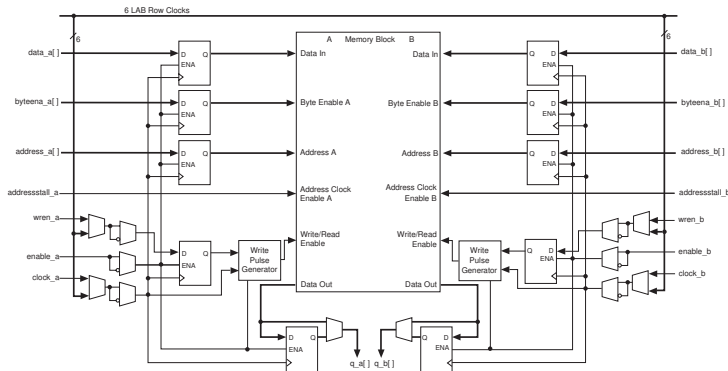
■ Le multiplieur 18x18 du Cyclone II



Les hardblocks

Les mémoires

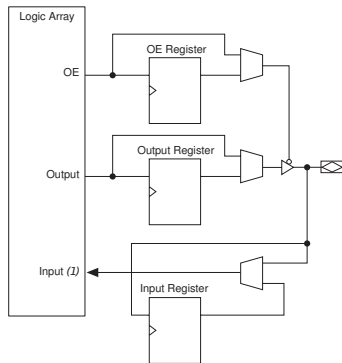
■ Les RAM synchrones du Cyclone II



Les hardblocks

Les IOs

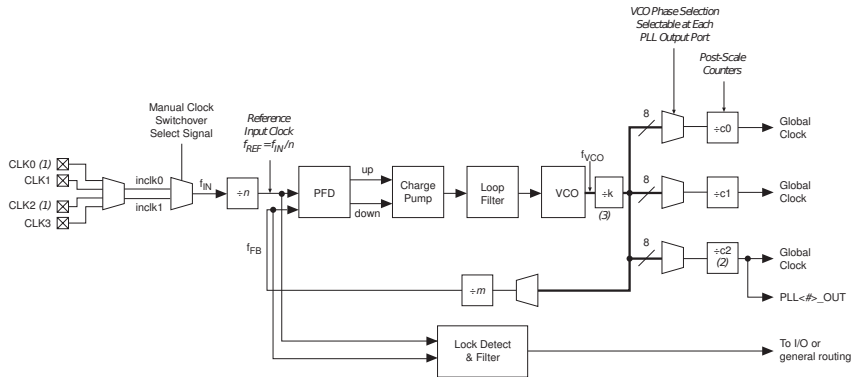
■ Les IOs du Cyclone II



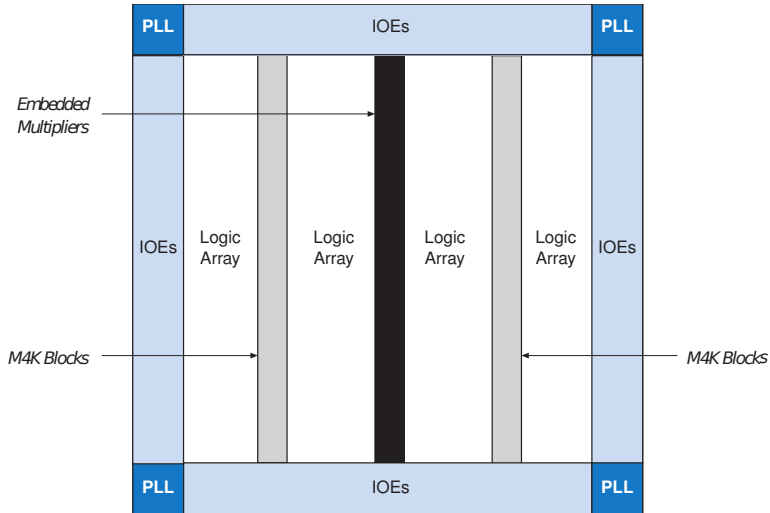
Les hardblocks

Les PLLs

■ Les PLLs du Cyclone II



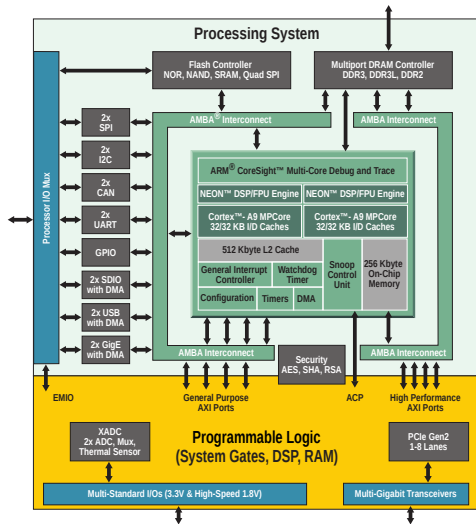
Structure du Cyclone II



Les hardblocks

Plus : Les SoC

- Tout un système avec un processeur et ses périphériques
- Exemples :
 - Xilinx Zynq
 - Altera SoC-FPGA



Plan

Les FPGAs

- La Look Up Table (LUT)
- Les blocs logiques
- L'interconnect
- Les hardblocks

Flot de développement FPGA

- La synthèse
- L'inférence

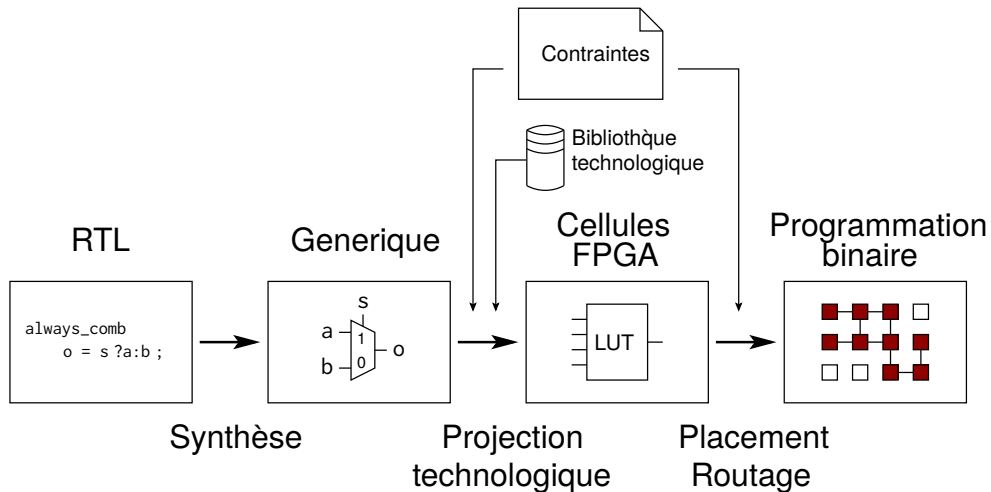
Spécificité des FPGAs

- Modélisation et inférence des mémoires
- Logique synchrone et FPGA

Annexes : Le Cyclonell

Synthèse FPGA

Flot simplifié



1. Une synthèse générique

- A partir d'une description RTL (votre code)
- Générer une structure de primitives génériques (**Netlist**)

2. Projection technologique (mapping)

- Générer une structure de primitives spécifique à la cible

3. Placement et routage (P&R)

- Placer les cellules dans la matrice
- Choisir les canaux de routage
- Générer un fichier binaire pour programmer le FPGA

■ Le RTL

- Ce que vous fournissez
- Utilise un sous-ensemble **synthétisable** du langage HDL (SystemVerilog)
- Peut être comportemental ou structurel

■ Les bibliothèques technologiques

- Décrit les cibles FPGA
- La nature et nombre de cellules disponibles

■ Les contraintes

- La position des entrées et sorties
- La fréquence de fonctionnement
- Modèle et taille du FPGA, cellules particulières, ...
- Un standard (**sdc** : Synopsys Design Constraints File)

■ Les « netlists »

- Forcement structurel (ensemble de primitives)
- En HDL (verilog, vhd1) pour permettent la simulation
- Des langages spécifiques par exemple edif
- Des formats propriétaires (chacun a le sien)
- Des informations sur les timings (**sdf** Standard Delay Format)
 - On parle de netlists rétro-annotées (back annotated)

■ Le fichier de programmation

- Fichier binaire
- Format propriétaire (généralement non documenté)
- Diffèrent d'un modèle de FPGA à l'autre

Plan

Les FPGAs

- La Look Up Table (LUT)
- Les blocs logiques
- L'interconnect
- Les hardblocks

Flot de développement FPGA

- La synthèse
- L'inférence

Spécificité des FPGAs

- Modélisation et inférence des mémoires
- Logique synchrone et FPGA

Annexes : Le CyclonII



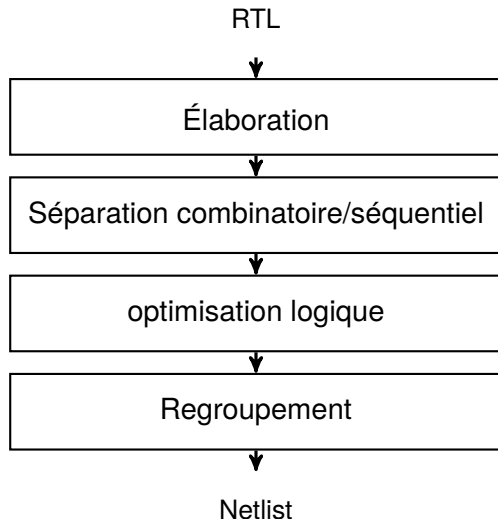
La synthèse

Les principes

- Comment passer d'une description **RTL** à une liste de primitives (pour l'instant indépendantes de la technologie cible).

La synthèse

Les principes





La synthèse

Les principes

- **Élaboration**
 - Instancier tous les sous-modules
 - Remplacer les constantes/paramètres, dérouler les boucles
 - ...
- **Séparer les éléments séquentiels de la logique combinatoire**
 - Détection de patterns
 - ...
- **Optimisation de la logique**
 - Technique d'optimisations
 - Arbres binaires...
- **Regroupement (Clustering)**
 - Dégrouper en blocs de base
 - Dépend de la cible

Exemple de synthèse

Que donnerait ceci pour $A \in [0 : 16[$

```
always_ff@(posedge clk)
  if(A/2) B <= 0;
  else   B <= 1;
```

Exemple de synthèse

Séparation logique séquentielle/combinatoire

```
always_ff@(posedge clk)
  B <= B_c;
```

```
always_comb
  if(A[3:1] != 0) B_c = 0;
  else           B_c = 1;
```

Exemple de synthèse

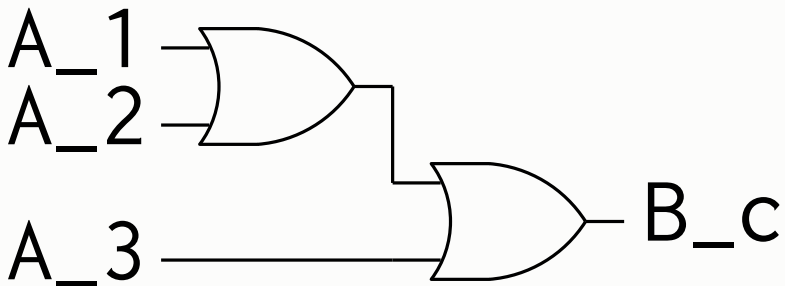
Optimisation de la partie combinatoire

A	A_3	A_2	A_1	A_0	B_r
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
11	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

$$B_c = A_3 \mid A_2 \mid A_1;$$

Exemple de synthèse

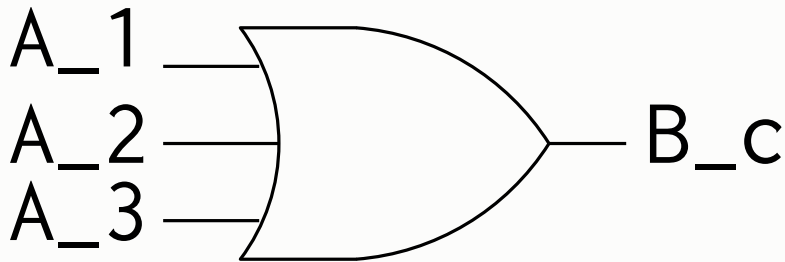
Regroupement



Dépendra de la technologie cible

Exemple de synthèse

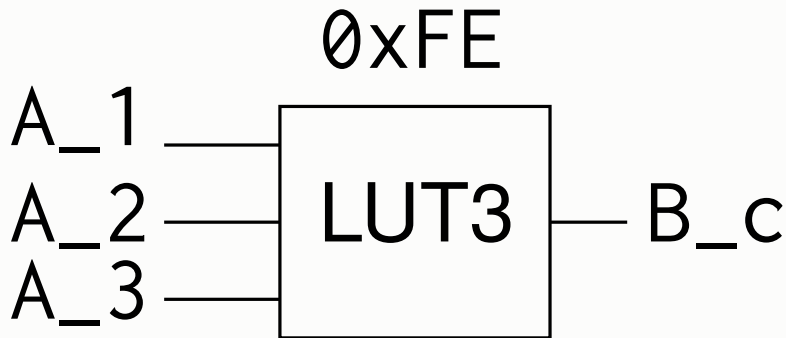
Regroupement



Dépendra de la technologie cible

Exemple de synthèse

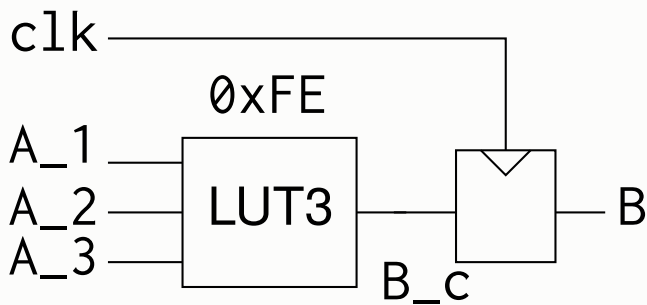
Regroupement



Pour les FPGA il faut regrouper en Luts

Exemple de synthèse

Netlist



Exemple de synthèse

Netlist

```
module foo( input clk, input[3:0]A, output B);  
  
  wire A_1 = A[1];  
  wire A_2 = A[2];  
  wire A_3 = A[3];  
  wire B_c;  
  
  LUT3 #(.conf(8'hFE)) lut3_i (.i0(A_1),.i1(A_1),.i2(A_2), .o(B_c))  
  REG1 reg_i (.clk(clk), .d(B_c), .Q(B));  
  
endmodule
```


Plan

Les FPGAs

- La Look Up Table (LUT)
- Les blocs logiques
- L'interconnect
- Les hardblocks

Flot de développement FPGA

- La synthèse
- L'inférence

Spécificité des FPGAs

- Modélisation et inférence des mémoires
- Logique synchrone et FPGA

Annexes : Le Cyclonell

Problème

Cette méthode n'est pas forcément la plus efficace.

- Il y a des structures connues qui sont efficaces pour implémenter certains opérateurs arithmétiques.
- Dans les FPGAs, il existe de blocs en « dur » qui ne sont pas un assemblage de LUTs

Deux possibilités :

- Instancier explicitement des primitives (faire du structurel).
 - le code n'est plus générique !
- L'outil **détecte** ces opérateurs et blocs spéciaux dans le code et les remplace automatiquement.

Cette 2nd option est appelée **l'inférence**.

Inférence/Instanciation

Exemple

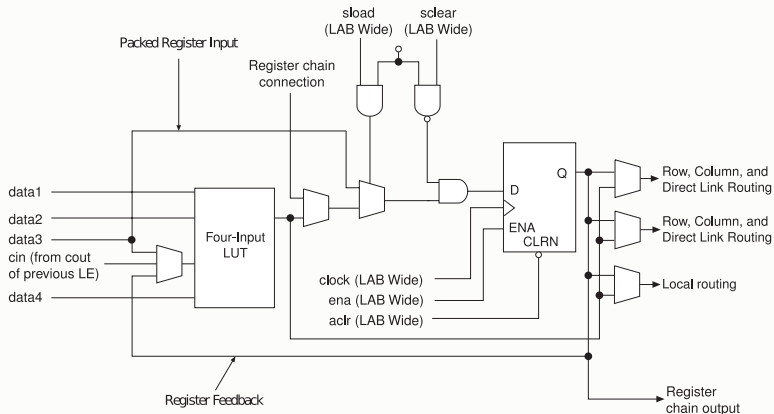
L'addition

```
assign {Co,S} = A + B + Ci
```

Inference/Instanciation

Exemple

Cellule logique du Cyclone II

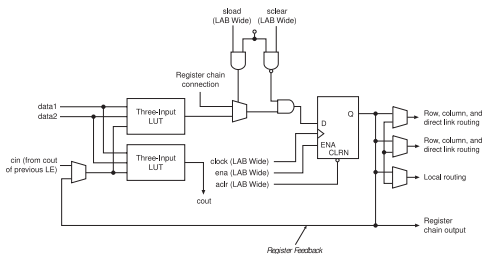


Inférence/Instanciation

Que peut-on inférer ?

les opérateurs arithmétiques

- L'addition : en utilisant les chaines de retenue.

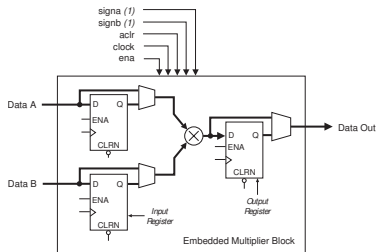


Inférence/Instanciation

Que peut-on inférer ?

les opérateurs arithmétiques

- La multiplication : en utilisant les DSPs.



multiplieur 18 × 18 du CycloneII



Inférence/Instanciation

Que peut-on inférer ?

les opérateurs arithmétiques

- La division ?

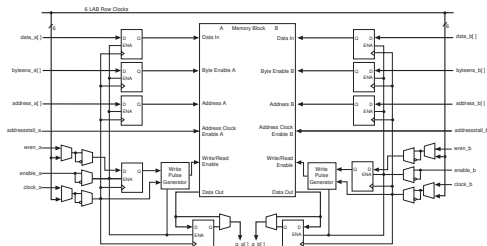
En utilisant beaucoup de LUTs.

Inférence/Instanciation

Que peut-on inférer ?

les mémoires

- Généralement les FPGA ont des blocs de mémoires **synchrones**.



Bram 4Kbit du CycloneII



Inférence/Instanciation

Contrainte

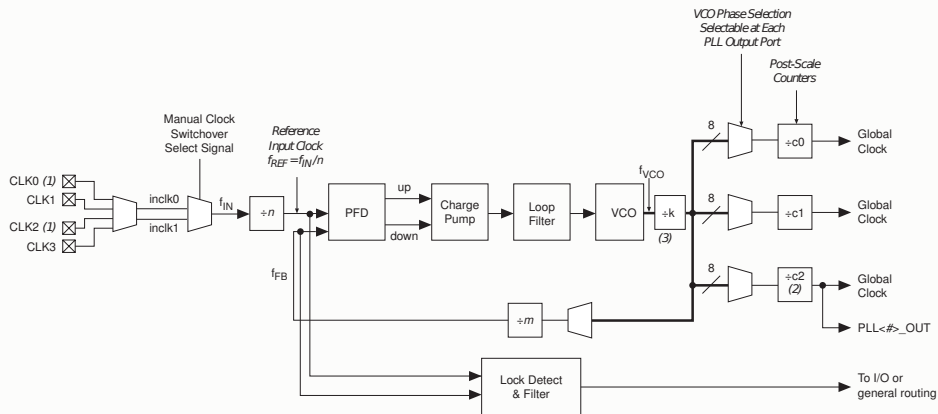
- Il faut que le comportement RTL corresponde au comportement du bloc que l'on veut inférer.
- Il faut que l'outil de synthèse sache le faire
- Il faut que le style de codage corresponde à celui attendu par l'outil !

Inférence/Instanciation

Que ne peut-on pas inférer ?

- Tous ce qu'on ne sait pas exprimer en RTL

Par exemple les PLLs



Plan

Les FPGAs

- La Look Up Table (LUT)
- Les blocs logiques
- L'interconnect
- Les hardblocks

Flot de développement FPGA

- La synthèse
- L'inférence

Spécificité des FPGAs

- Modélisation et inférence des mémoires
- Logique synchrone et FPGA

Annexes : Le Cyclonell

Plan

Les FPGAs

- La Look Up Table (LUT)
- Les blocs logiques
- L'interconnect
- Les hardblocks

Flot de développement FPGA

- La synthèse
- L'inférence

Spécificité des FPGAs

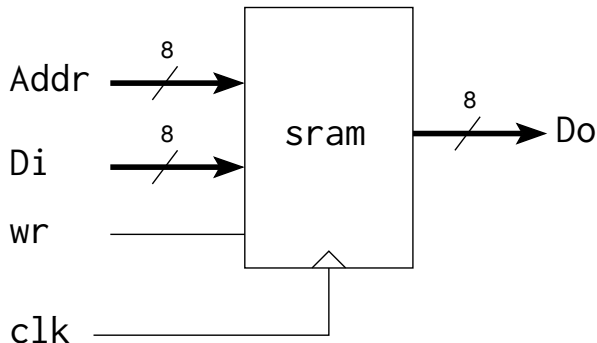
- Modélisation et inférence des mémoires
- Logique synchrone et FPGA

Annexes : Le CyclonII

Mémoire synchrone

mémoire simple port

- un bus d'adresse
- 2 bus pour les données :
 - écriture
 - lecture
- des signaux de contrôle
- une **horloge**
- **PAS DE RESET**
- On ne peut accéder qu'à un seul élément dans le même cycle !



Mémoire synchrone

mémoire simple port

- un bus d'adresse
- 2 bus pour les données :
 - écriture
 - lecture
- des signaux de contrôle
- une horloge
- **PAS DE RESET**
- On ne peut accéder qu'à un seul élément dans le même cycle !

```
module sram(input clk, wr,
            input [7:0] Addr,
            input [7:0] Di,
            output logic [7:0] Do );

logic[7:0] mem [0:255];

always_ff @(posedge clk)
begin
    if (wr)
        mem[Addr] <= Di;
    Do <= mem[Addr];
end

endmodule
```

Mémoire synchrone

mémoire simple port

- un bus d'adresse
- 2 bus pour les données :
 - écriture
 - lecture
- des signaux de contrôle
- une horloge
- **PAS DE RESET**
- On ne peut accéder qu'à un seul élément dans le même cycle !

```
module sram(input clk, wr,
            input [7:0] Addr,
            input [7:0] Di,
            output [7:0] Do );

    logic[7:0] mem [0:255];
    logic[7:0] Addr_r;

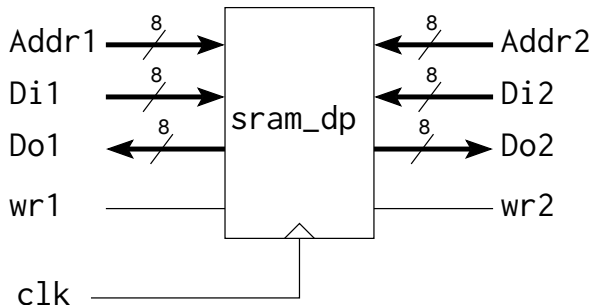
    always_ff @(posedge clk)
    begin
        if (wr)
            mem[Addr] <= Di;
            Addr_r <= Addr;
        end

    assign Do = mem[Addr_r];

endmodule
```


Mémoire synchrone mémoire double ports

- permet un accès double
- l'écriture et la lecture à la même adresse dans le même cycle n'est pas prédictible
- pourrait avoir deux horloges



Mémoire synchrone

mémoire double ports

- permet un accès double
- l'écriture et la lecture à la même adresse dans le même cycle n'est pas prédictible
- pourrait avoir deux horloges

```
module sram_dp(input clk, wr1, wr2,
               input [7:0] Addr1, Addr2,
               input [7:0] Di1, Di2,
               output logic [7:0] Do1, Do2 );

logic[7:0] mem [0:255];

always_ff @(posedge clk)
begin
    if (wr1)
        mem[Addr1] <= Di1;
    Do1 <= mem[Addr1];
end

always_ff @(posedge clk)
begin
    if (wr2)
        mem[Addr2] <= Di2;
    Do2 <= mem[Addr2];
end

endmodule
```

Mémoire synchrone

Initialisation du contenu

- Possible seulement pour les FPGA
- **initial** est normalement exclusivement réservé à la simulation
- **\$readmemh** (ou **\$readmemb**) permet d'initialiser une table à partir d'un fichier

```
module sram(input clk, wr,
            input [7:0] Addr,
            input [7:0] Di,
            output logic [7:0] Do );

logic[7:0] mem [0:255];

initial
    $readmemh("init.txt", mem);

always_ff @(posedge clk)
begin
    if (wr)
        mem[Addr] <= Di;
    Do <= mem[Addr];
end

endmodule
```

Mémoire synchrone

ROM synchrone

- Possible seulement pour les FPGA
- il suffit d'enlever la possibilité d'écrire

```
module rom (input clk,  
            input [7:0] Addr,  
            output logic [7:0] Do );  
  
logic[7:0] mem [0:255];  
  
initial  
    $readmemh("init.txt", mem);  
  
always_ff @(posedge clk)  
    Do <= mem[Addr];  
  
endmodule
```

Plan

Les FPGAs

- La Look Up Table (LUT)
- Les blocs logiques
- L'interconnect
- Les hardblocks

Flot de développement FPGA

- La synthèse
- L'inférence

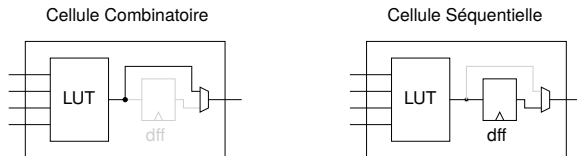
Spécificité des FPGAs

- Modélisation et inférence des mémoires
- Logique synchrone et FPGA

Annexes : Le Cyclonell

Particularité des FPGA

La logique synchrone

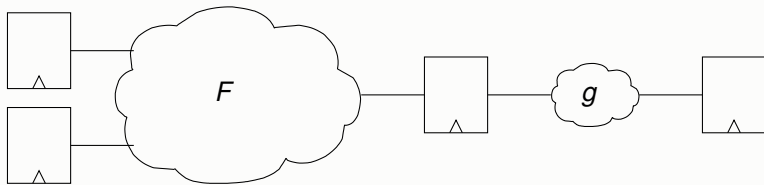


- Les bascules sont une ressource "gratuite" dans les FPGA.
- On doit essayer de les utiliser au maximum.
 - Faire du pipeline.
 - Encodage des états OneHot

Exemple : Le retiming

Pipeline automatique

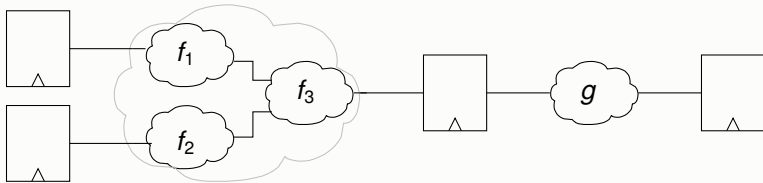
Le pipeline c'est quoi ?



Exemple : Le retiming

Pipeline automatique

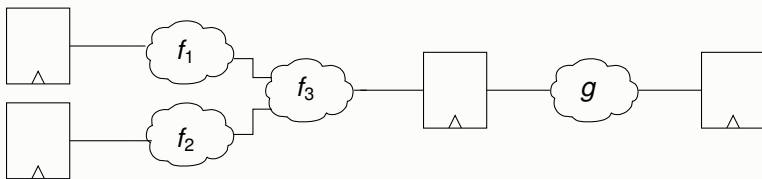
Le pipeline c'est quoi ?



Exemple : Le retiming

Pipeline automatique

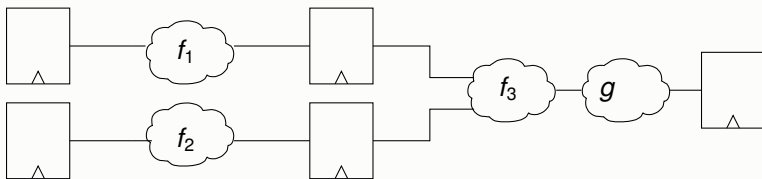
Le pipeline c'est quoi ?



Exemple : Le retiming

Pipeline automatique

Le pipeline c'est quoi ?

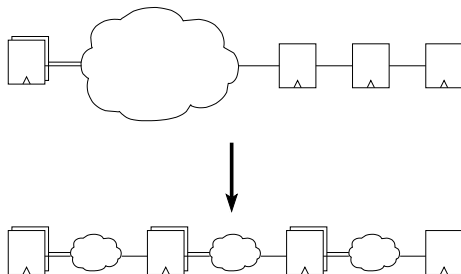


Exemple : Le retiming Pipeline automatique

```
always_ff @(posedge clk)
begin
    // Les registres d'entrées
    R0 <= In0;
    R1 <= In1;
    // La fonction combinatoire
    S0 <= f(R0,R1);
    // registres de sortie
    S1 <= S0;
    S <= S1;
end
```

Exemple : Le retiming Pipeline automatique

```
always_ff @(posedge clk)
begin
    // Les registres d'entrées
    R0 <= In0;
    R1 <= In1;
    // La fonction combinatoire
    S0 <= f(R0,R1);
    // registres de sortie
    S1 <= S0;
    S <= S1;
end
```



Plan

Les FPGAs

- La Look Up Table (LUT)
- Les blocs logiques
- L'interconnect
- Les hardblocks

Flot de développement FPGA

- La synthèse
- L'inférence

Spécificité des FPGAs

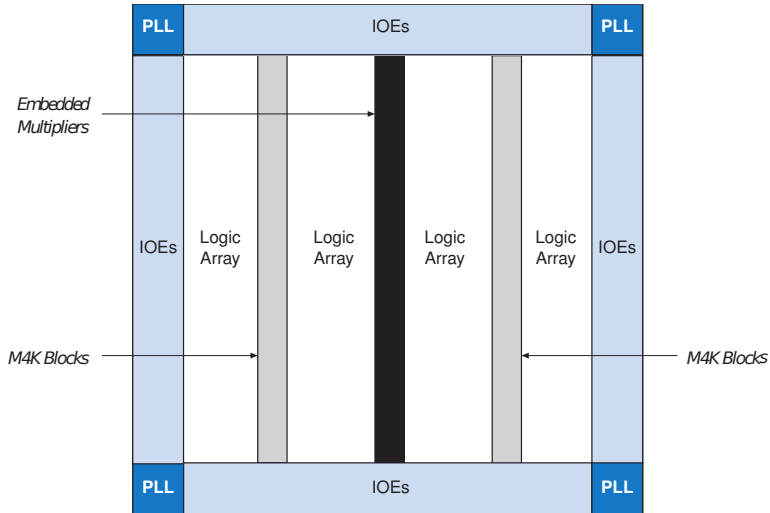
- Modélisation et inférence des mémoires
- Logique synchrone et FPGA

Annexes : Le Cyclonell

Ressources Disponibles Dans un FPGA

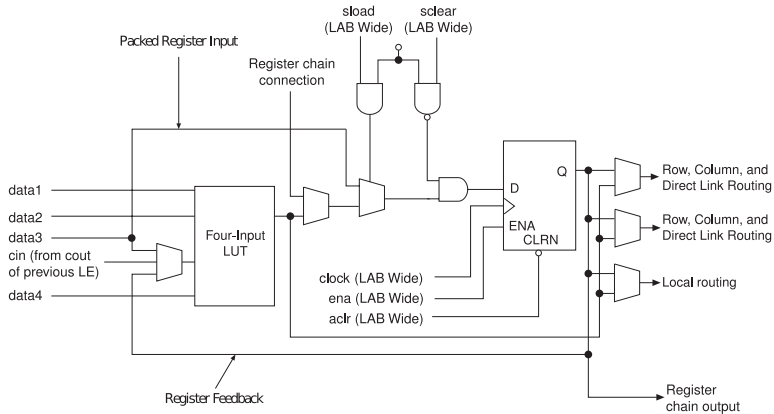
- La principale ressource disponible dans un FPGA est la
 - Cellule Logique.
 - Une Cellule logique contient une au moins 1 LUT + 1 DFF
- Des blocs embarqués :
 - des mémoires
 - des multiplieurs/DSP
 - des PLLs/ circuits d'horloges
 - des I/O

Structure du Cyclone II



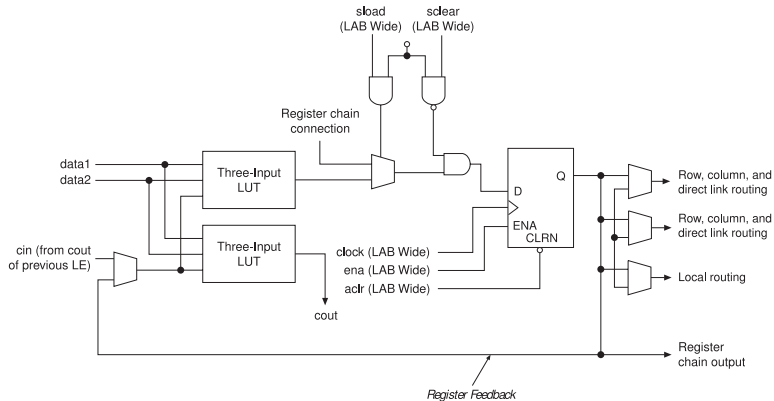
Logic Element

■ Une LUT à 4 entrées + Une bascule



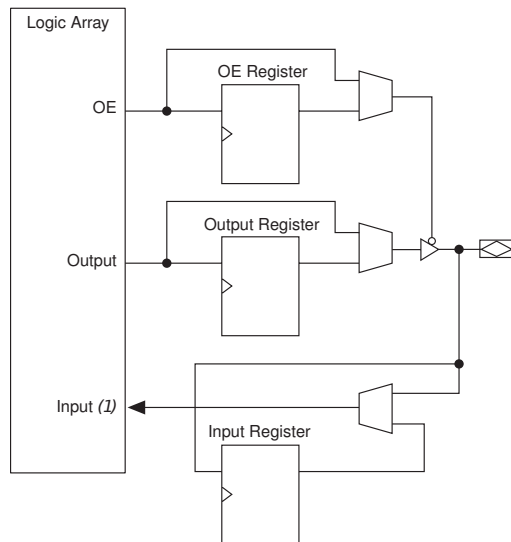
Logic Element en mode arithmétique

- Spécialement pour faire des additions/soustractions



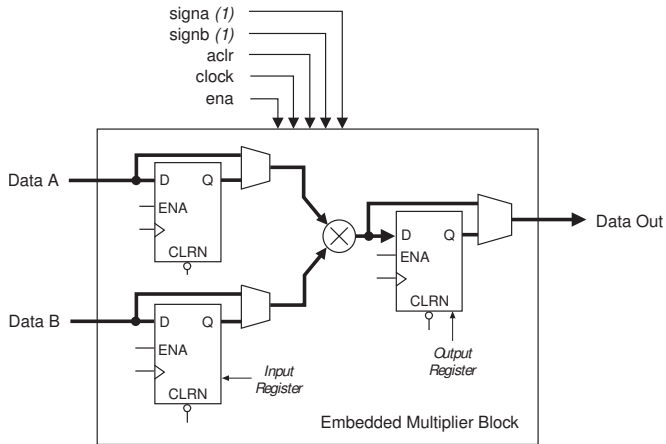
Les Entrées/Sorties (IO)

- Configurables
- Contiennent des registres
- Bidirectionnelles (3 états)

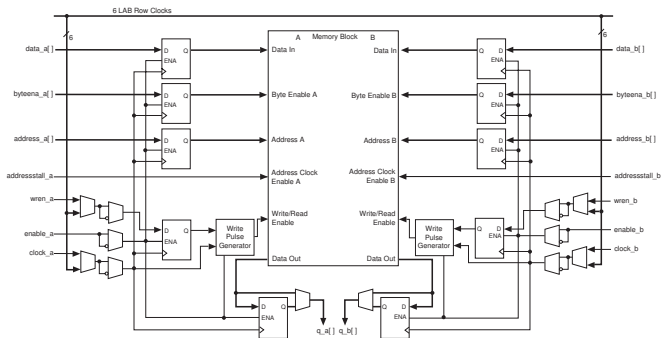


Multiplieur

- Multiplieur 18×18
- 2 multiplieurs 9×9



Mémoire embarquée



PLL

