

Cours VHDL - II

L3-S6 - Université de Cergy-Pontoise

Laurent Rodriguez – Benoît Miramond

Plan du cours

I – Historique de conception des circuits intégrés

- HDL

- Modèles de conceptions

- VHDL

- Les modèles de conceptions en VHDL
- Les 5 briques de base

II – VHDL et FPGA

- VHDL

- Syntaxe et typage
- Retour sur les briques de bases
- Retour sur la conception structurelle et comportementale en VHDL
Port map, Equations logiques, Tables de vérités (With ... Select)

- FPGA

- Qu'est ce qu'un FPGA
- Flot de conception
- Carte de développement et environnement de TP

A thick, blue, hand-drawn border surrounds the entire page, giving it a sketchy, artistic appearance.

I - Rappels

I – Rappels : Hardware Description Language – Modèles de descriptions

On distingue donc plusieurs niveaux de modélisation (modèles ou encore vues) :

- **Physiques**

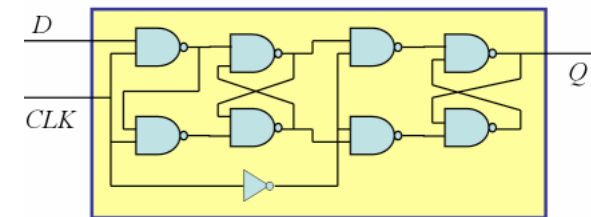
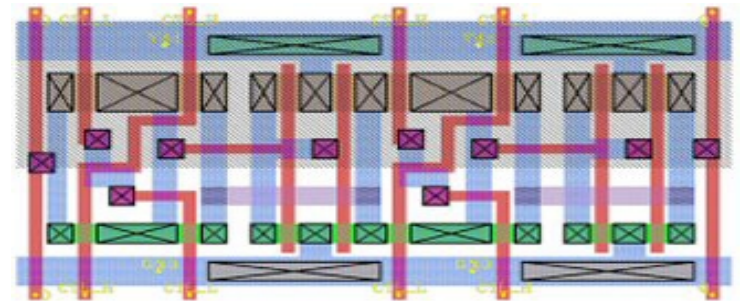
- Dimensions
- Matériaux
- Transistors
- Masques, ...

- **Structurelles (textuelle ou schématique)**

- Assemblage de composants
- Hiérarchie d'interconnexions de différents sous-ensemble dont le plus bas niveau est le transistor

- **Comportementales**

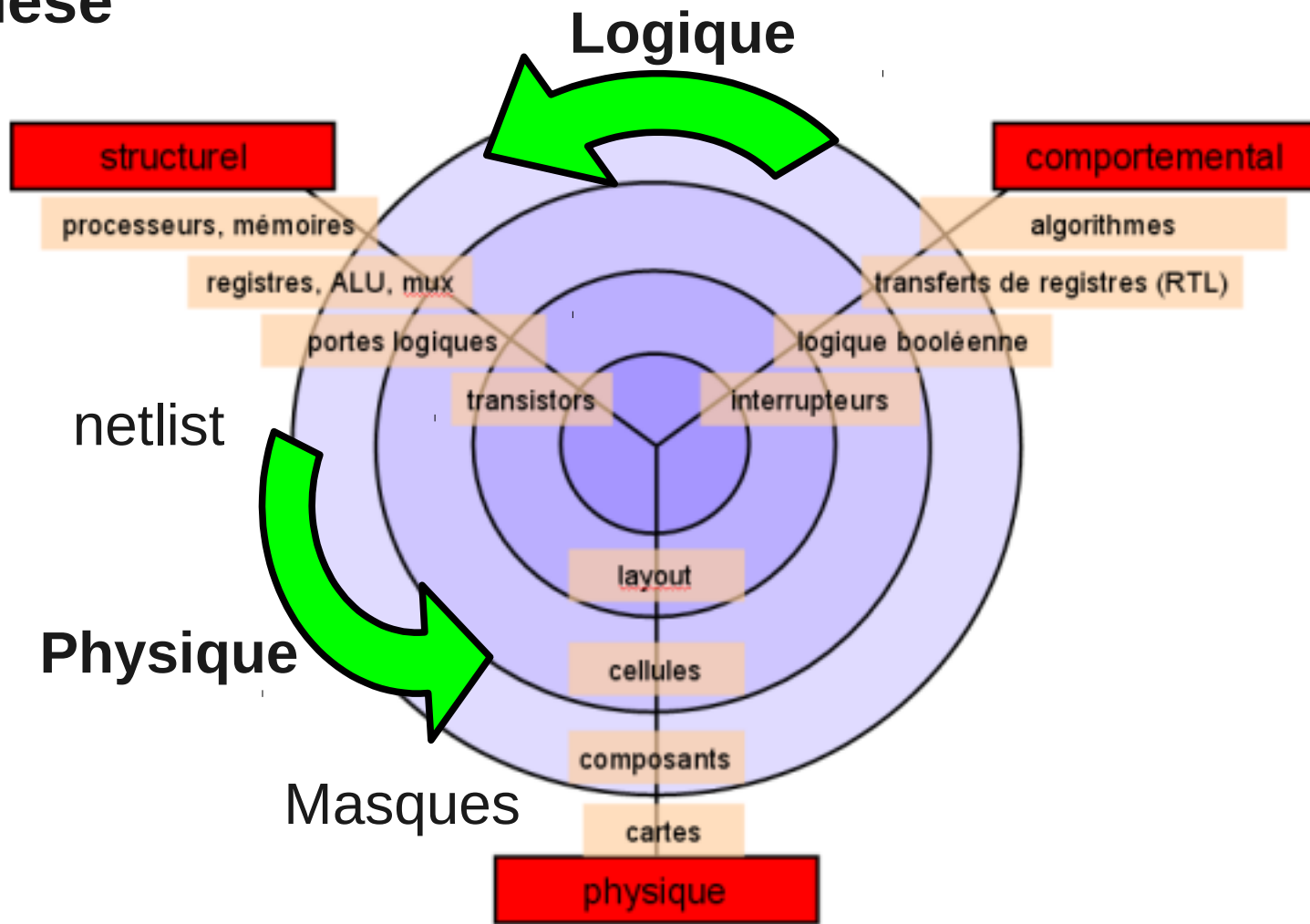
- Fonction réalisée
- Le « Quoi » et non le « Comment »



```
bloc BK(I0, I1, I2 : in;  
        S0, S1 : out)  
  
{  
  S0 = non(I0 ou (I1 et I2));  
  S1 = non(I1) ou non(I2);  
}
```

I – HDL : Hardware Description Language – Modèles de descriptions et Synthèse

Synthèse

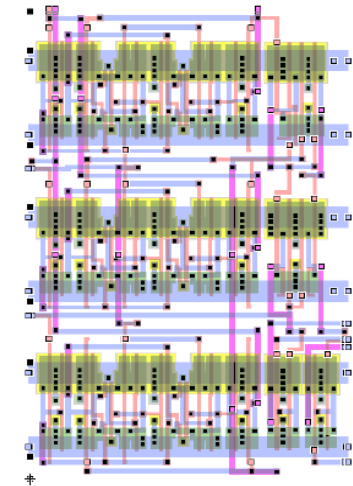


I – HDL : Hardware Description Language – Synthèse

La synthèse logique

Transforme le code en une représentation structurelle de bas niveau (netList) utilisant les cellules de la bibliothèque de la technologie visée (fondeur ou FPGA).

Permet la **simulation** sans connaissance du matériel cible

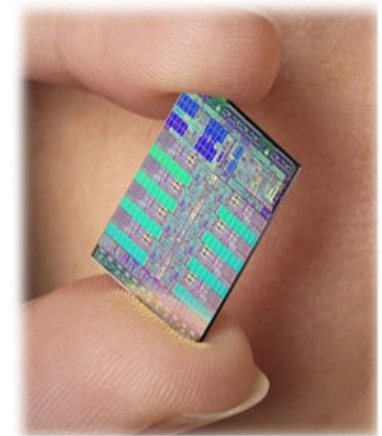


La synthèse physique

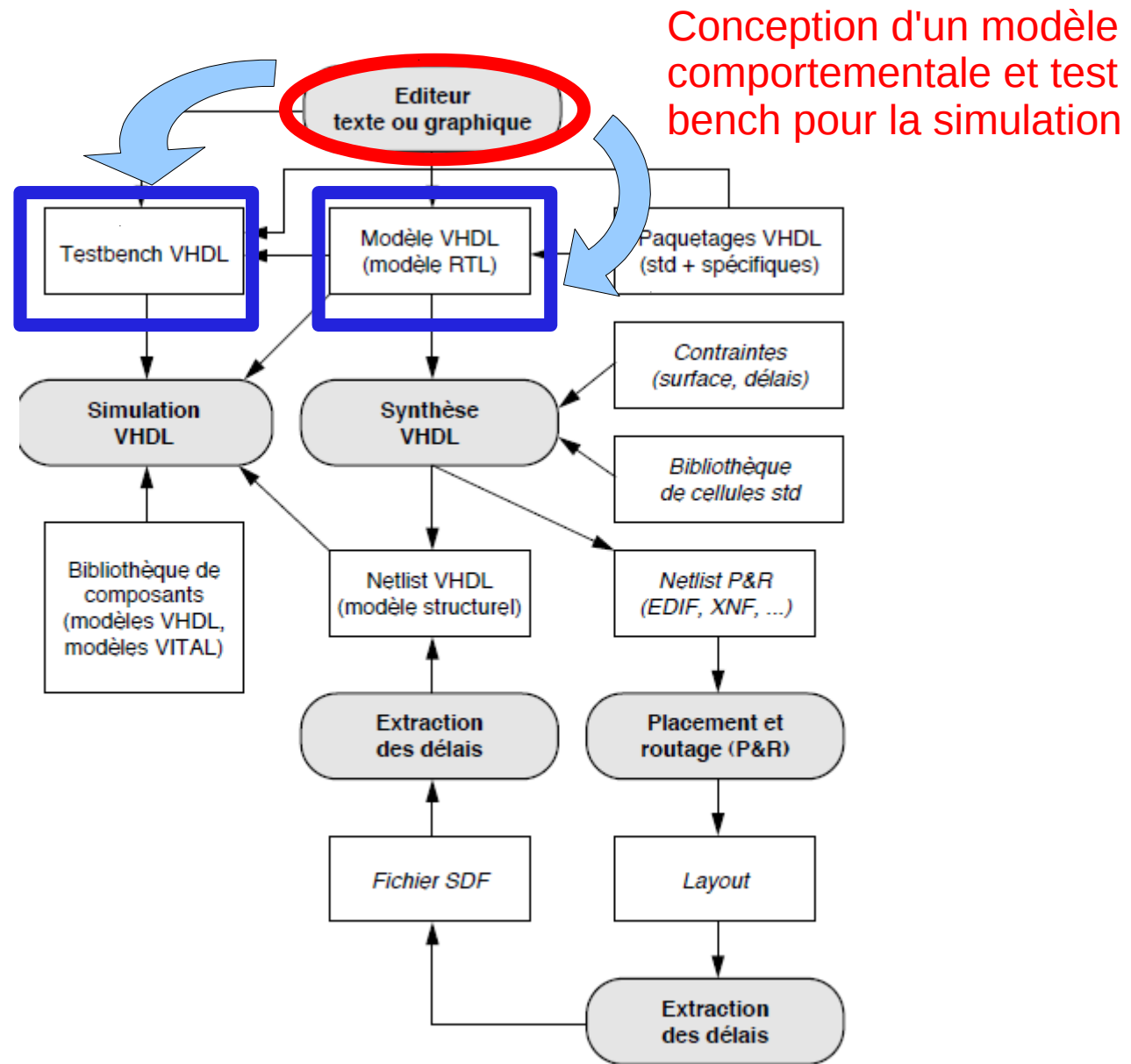
Transforme une représentation structurelle de bas niveau en une description physique du circuit (layout). Elle nécessite une étape supplémentaire de placement et de routage des portes.

Permet une **simulation** plus riche tenant compte du matériel cible (délais)

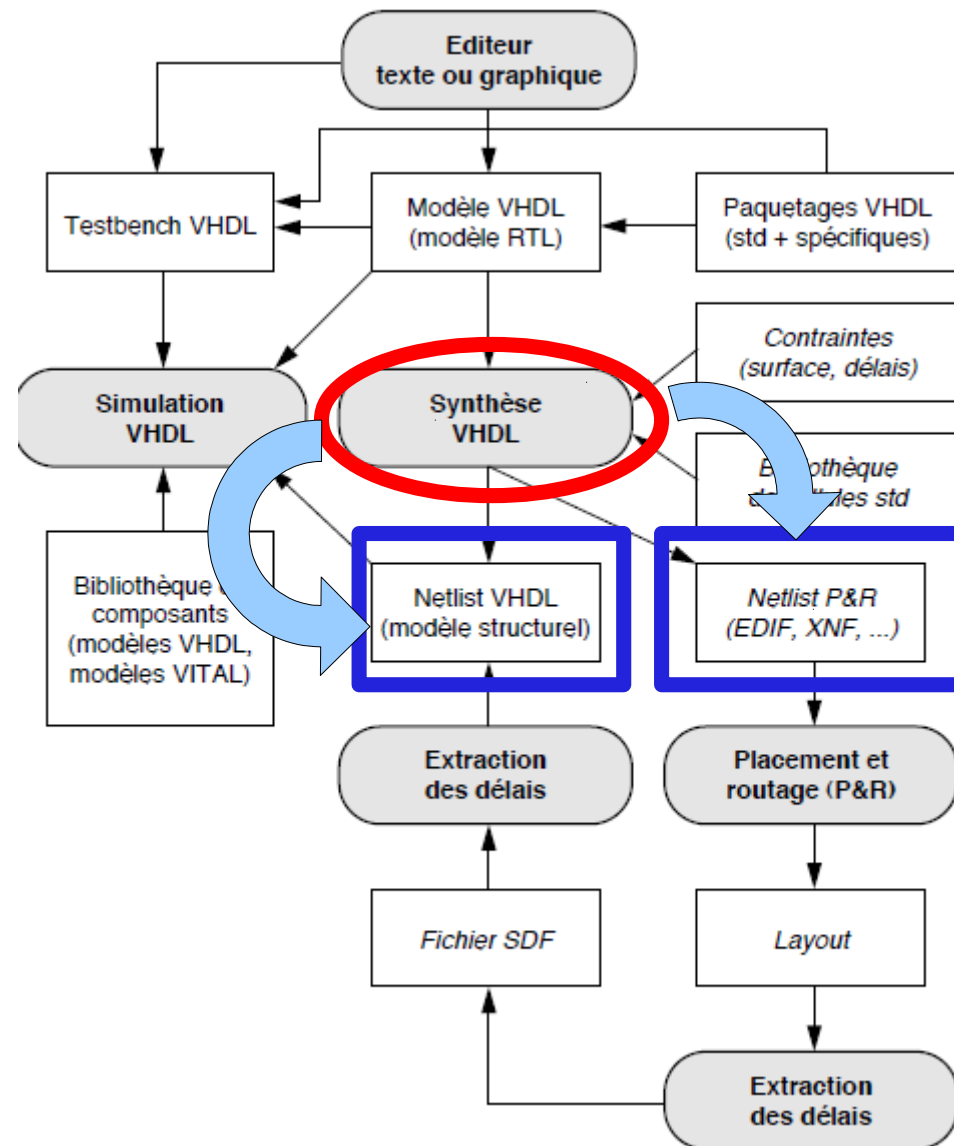
Permet la **configuration** sur **FPGA** ou la réalisation d'un circuit dédié (**ASIC**)



I – VHDL : Flot de conception V(HDL)

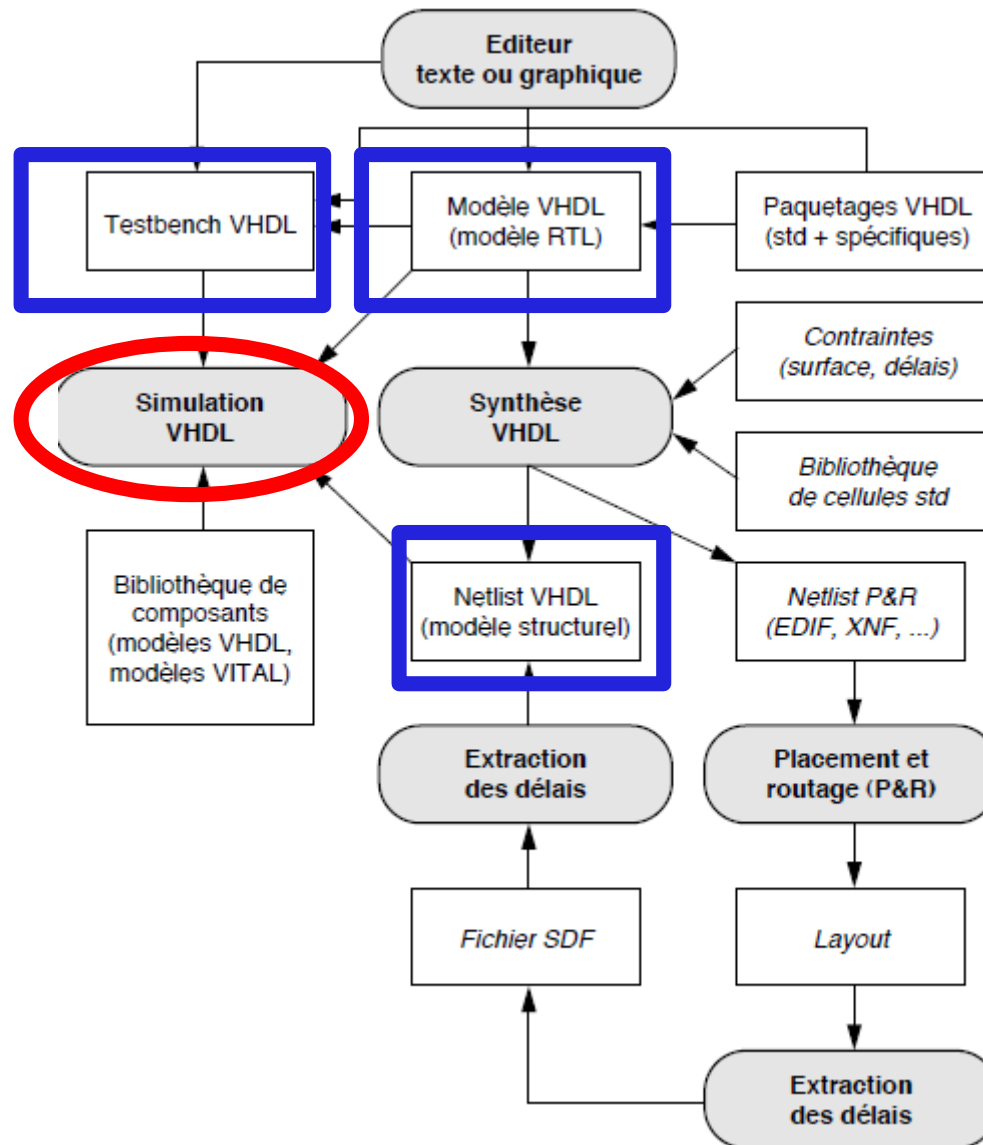


I – VHDL : Flot de conception V(HDL)

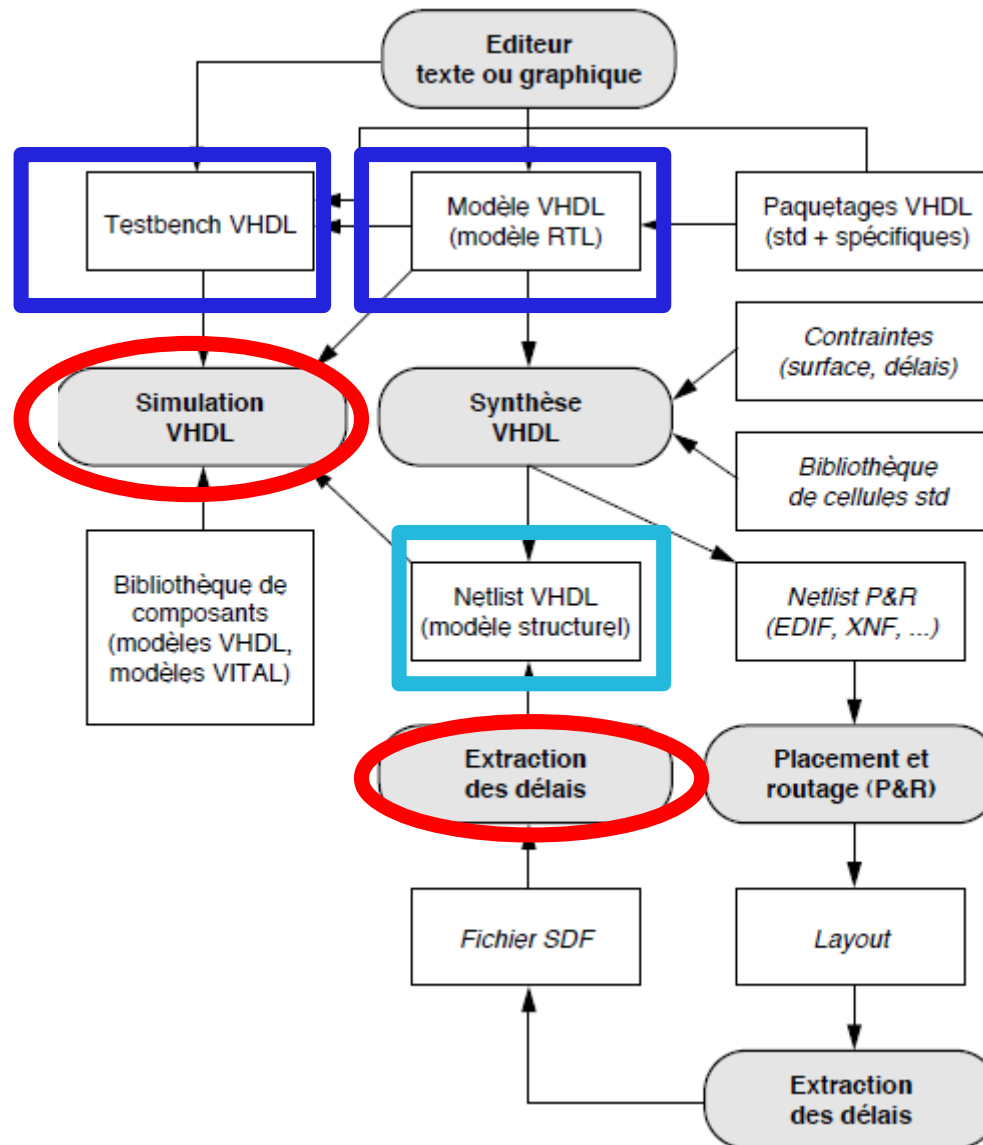


I – VHDL : Flot de conception V(HDL)

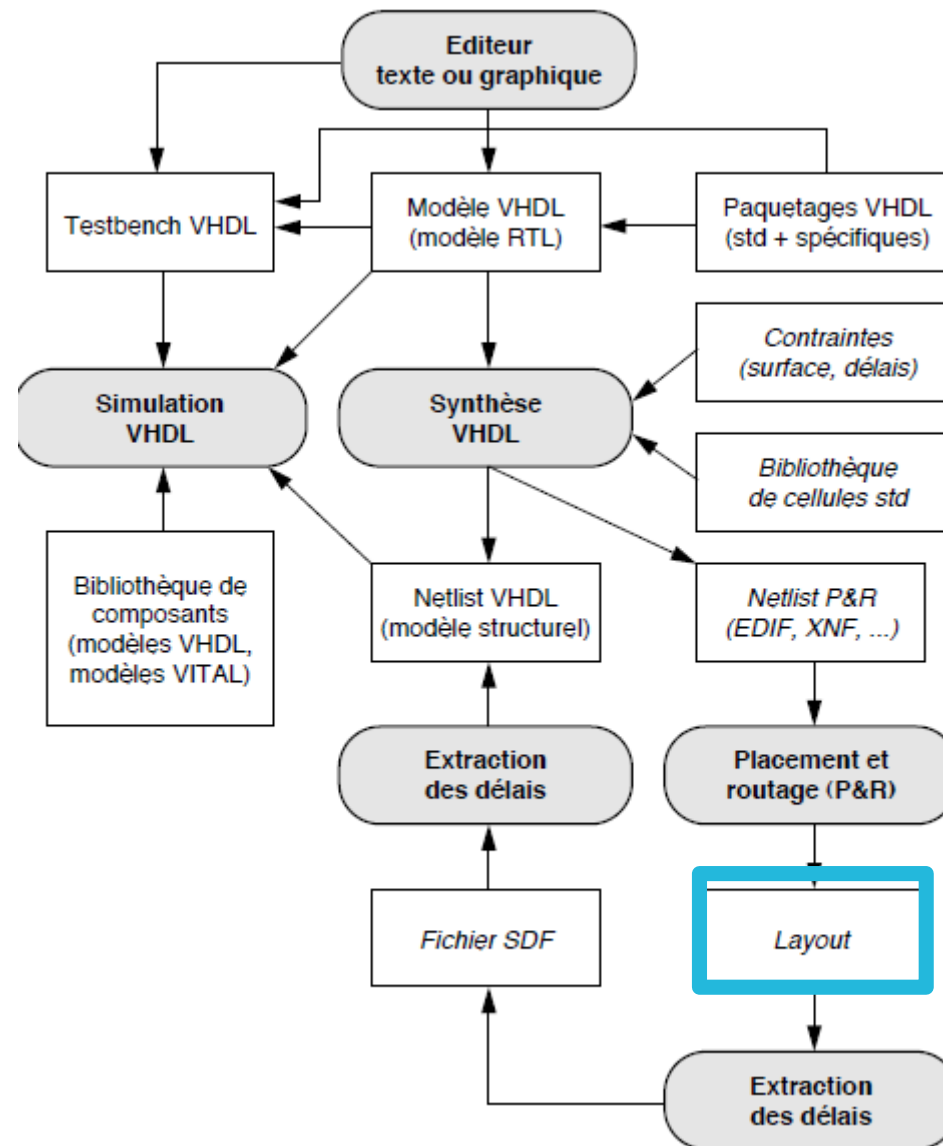
Simulation





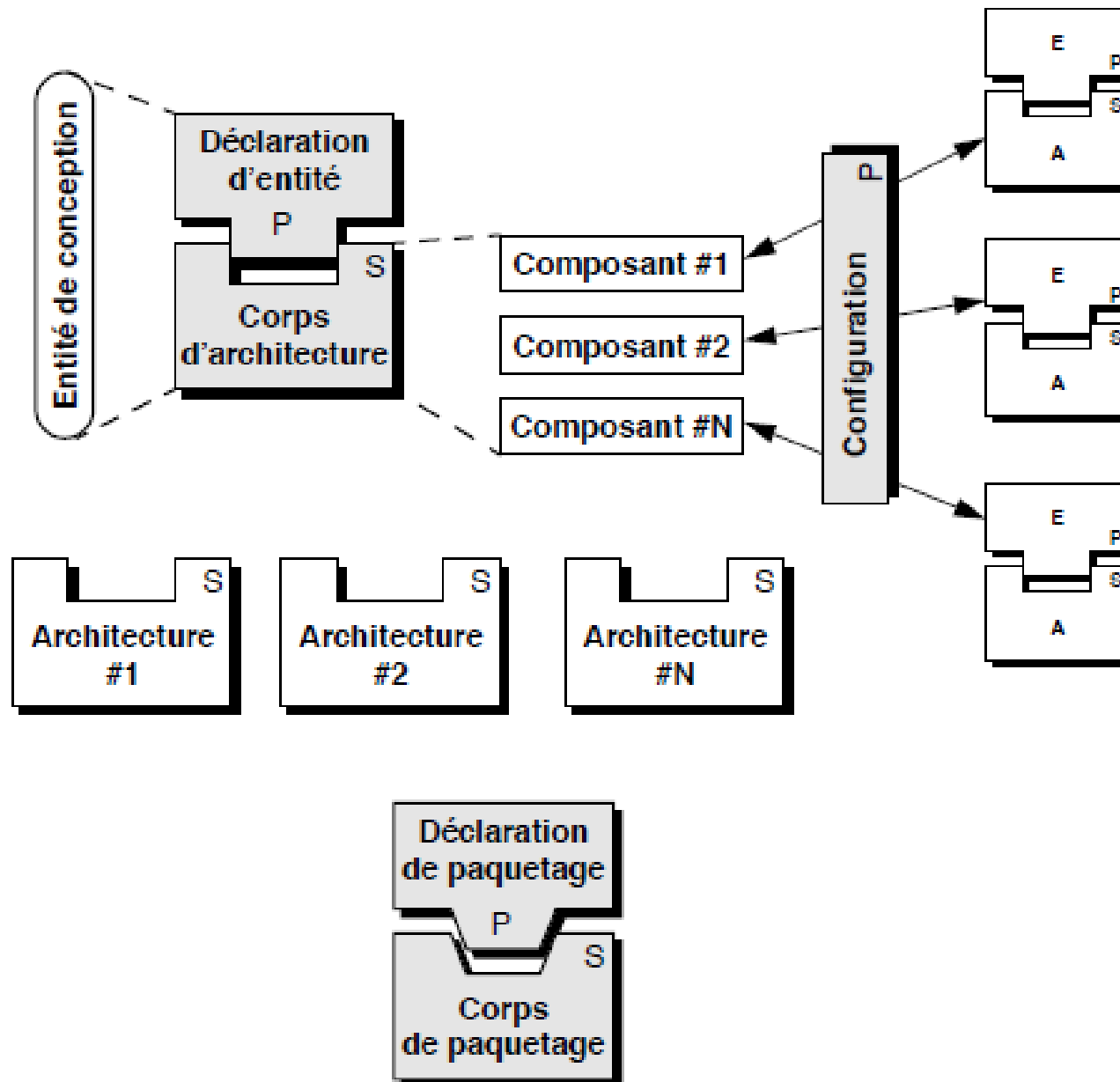


I – VHDL : Flot de conception V(HDL)



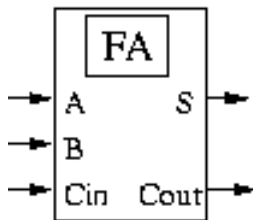
**Création du
circuit /
utilisation sur
FPGA**

I – VHDL : Unités de conception



I – VHDL : L'entité

L'entité est la description de l'interface du circuit . Elle correspond au symbole dans les représentations schématiques :



L'entité précise :

- ★ le nom du circuit
- ★ Les ports d'entrée-sortie :
 - ★ Leur nom
 - ★ Leur direction (in, out, inout,...)
 - ★ Leur type (bit, bit_vector, integer, std_logic,...)
- ★ Les paramètres éventuels pour les modèles génériques

```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;
```

```
entity fa is  
  port (  
    a, b, cin : in std_logic;  
    s, cout : out std_logic  
  );  
end entity;
```

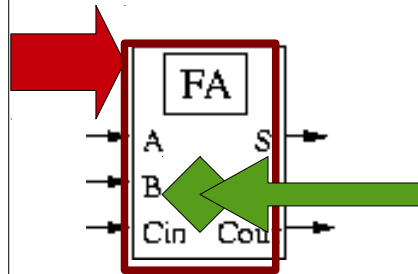
I – VHDL : Architecture – description des modules

L'architecture est la description interne du circuit.

- ★ Elle est toujours associée à une entité.
- ★ Une même entité peut avoir plusieurs architecture.

Le mécanisme de **configuration** (décrit dans le VHDL structurel) permet d'indiquer l'architecture rattachée à une entité.

```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
  
entity fa is  
  port (  
    a, b, cin : in std_logic;  
    s, cout : out std_logic  
  );  
end entity;
```



```
architecture arc2 of fa is  
  begin  
    s <= a xor b xor cin;  
    cout <= (a and b) or ((a xor b) and cin);  
  end arc1;
```

II – VHDL plus détaillé

II – VHDL : Architecture –

Type de descriptions - Comportementale

Comportementale

Ce type correspond à expliciter le comportement d'un modèle par ses équations

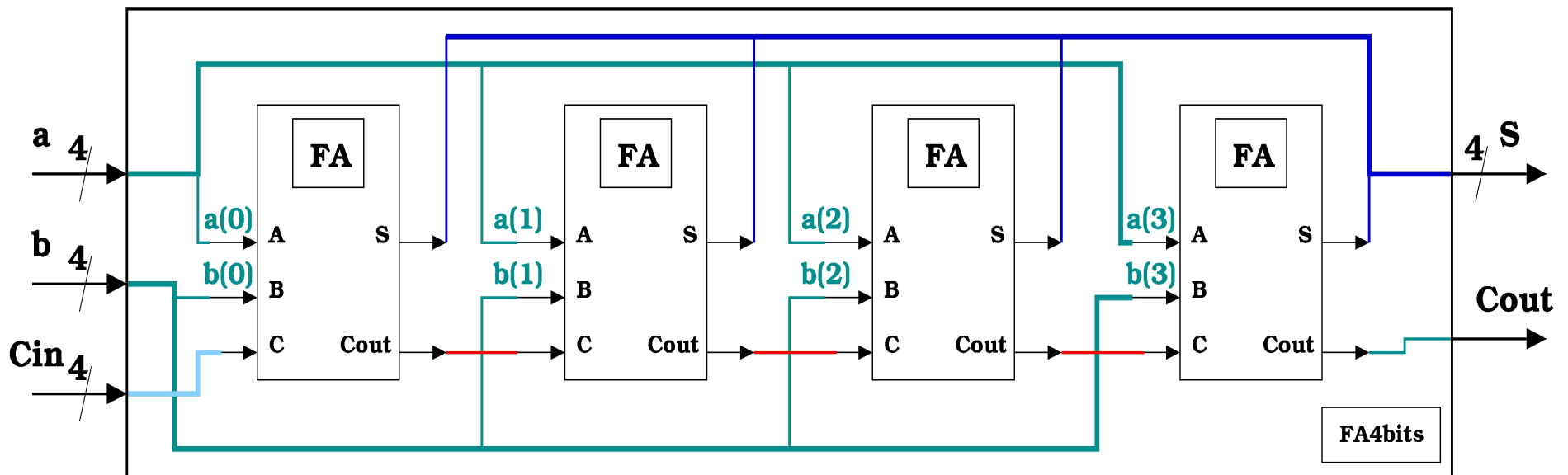
```
architecture arCL of fa is
begin
  s <= a xor b xor cin;
  cout <= (a and b) or ((a xor b) and cin);
end arCL;
```

```
architecture arcTV of fa is
  Signal tmp : std_logic_vector(1 down to 0) ;
begin
  tmp <= a&b;
  with tmp select
    S <= '0' when '00',
    '1' when '01',
    '1' when '10',
    '0' when '11';
end architecture bhv;
```

II – VHDL : Architecture – Type de descriptions - Structurelle

Structurelle

Ce type correspond à l'instanciation hiérarchique d'autres composants



— Signaux internes

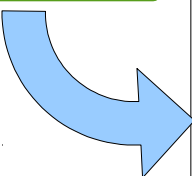
II – VHDL : Architecture – Type de descriptions - Structurelle

```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
  
entity fa is  
  port ( a, b, cin : in std_logic;  
         s, cout : out std_logic);  
end entity;
```

```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
  
entity fa4b is  
  port ( a, b, cin : in std_logic_vector(3 downto 0);  
         s, cout : out std_logic_vector(3 downto 0));  
end entity;  
  
Architecture fa4b_bhv of fa4b is  
  
  component fa is  
    port ( a, b, cin : in std_logic;  
          s, cout : out std_logic);  
  end component;  
  
  Signal cout_b : std_logic_vector(2 downto 0) ;  
  Begin  
    Add0 : fa port map (a => a(0), b => b(0), cin => cin,  
                        S => s(0), cout => cout_b(0)) ;  
    Add1 : fa port map (a(1), b(1), cout_b(0), s(1), cout_b(1)) ;  
    Add2 : fa port map (a(2), b(2), cout_b(1), s(2), cout_b(2)) ;  
    Add3 : fa port map (a(3), b(3), cout_b(2), s(3), cout) ;  
  End architecture ;
```

II – VHDL : Architecture – Type de descriptions - Structurelle

```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
  
entity fa is  
  port ( a, b, cin : in std_logic;  
         s, cout : out std_logic);  
end entity;
```



```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
  
entity fa4b is  
  port ( a, b, cin : in std_logic_vector(3 downto 0);  
         s, cout : out std_logic_vector(3 downto 0));  
end entity;
```

Architecture fa4b_bhv of fa4b is

```
component fa is  
  port ( a, b, cin : in std_logic;  
         s, cout : out std_logic);  
end component;
```

```
Signal cout_b : std_logic_vector(2 downto 0) ;  
Begin
```

```
  Add0 : fa port map (a => a(0), b => b(0), cin => cin,  
                      S => s(0), cout => cout_b(0)) ;  
  Add1 : fa port map (a(1), b(1), cout_b(0), s(1), cout_b(1)) ;  
  Add2 : fa port map (a(2), b(2), cout_b(1), s(2), cout_b(2)) ;  
  Add3 : fa port map (a(3), b(3), cout_b(2), s(3), cout) ;
```

```
End architecture ;
```

II – VHDL : Architecture – Type de descriptions - Structurelle

```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
  
entity fa is  
  port ( a, b, cin : in std_logic;  
         s, cout : out std_logic);  
end entity;
```

```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
  
entity fa4b is  
  port ( a, b, cin : in std_logic_vector(3 downto 0);  
         s, cout : out std_logic_vector(3 downto 0));  
end entity;
```

Architecture fa4b_bhv of fa4b is

```
  component fa is  
    port ( a, b, cin : in std_logic;  
           s, cout : out std_logic);  
  end component;
```

```
  Signal cout_b : std_logic_vector(2 downto 0) ;
```

Begin

```
  Add0 : fa port map (a => a(0), b => b(0), cin => cin,  
                      S => s(0), cout => cout_b(0)) ;
```

```
  Add1 : fa port map (a(1), b(1), cout_b(0), s(1), cout_b(1)) ;
```

```
  Add2 : fa port map (a(2), b(2), cout_b(1), s(2), cout_b(2)) ;
```

```
  Add3 : fa port map (a(3), b(3), cout_b(2), s(3), cout) ;
```

End **architecture** ;

II – VHDL : Architecture – Type de descriptions - Structurelle

```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
  
entity fa is  
  port ( a, b, cin : in std_logic;  
         s, cout : out std_logic);  
end entity;
```

```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
  
entity fa4b is  
  port ( a, b, cin : in std_logic_vector(3 downto 0);  
         s, cout : out std_logic_vector(3 downto 0));  
end entity;
```

Architecture fa4b_bhv of fa4b is

```
  component fa is  
    port ( a, b, cin : in std_logic;  
           s, cout : out std_logic);  
  end component;
```

Signal cout_b : std_logic_vector(2 downto 0) ;

Begin

```
  Add0 : fa port map (a => a(0), b => b(0), cin => cin,  
                      s => s(0), cout => cout_b(0)) ;
```

```
  Add1 : fa port map (a(1), b(1), cout_b(0), s(1), cout_b(1)) ;
```

```
  Add2 : fa port map (a(2), b(2), cout_b(1), s(2), cout_b(2)) ;
```

```
  Add3 : fa port map (a(3), b(3), cout_b(2), s(3), cout) ;
```

End **architecture** ,

II – VHDL : éléments de langage - Conventions lexicales

Casse : VHDL est **insensible** à la casse. Il est cependant conseillé d'avoir des règles d'écriture cohérentes. Par exemple, les mots réservés du langage peuvent être en majuscule et les autres mots en minuscule.

Commentaires : Les commentaires doivent être inclus dans le code, pour augmenter la lisibilité et la documentation. Ils commencent par 2 tirets (--) en se terminent en fin de ligne ;

Identificateurs : Ce sont les noms de variables, de signaux, de fonctions, ...

- Ils ne peuvent contenir que des lettres, des chiffres et le "underscore" _ .
- Ils doivent commencer par une lettre.
- Ils ne peuvent pas contenir d'espace.
- Les mots-clefs du langage ne peuvent pas être utilisés comme identificateurs.

Expressions : Elles se terminent par un point virgule ;

Ex : l'affectation: a <= b ; -- pour connecter des ports, signaux, affecter une valeur, ...

Littéraux : Ce sont des valeurs explicites :

- 67 est un littéral pour le type entier
- '0' est un littéral pour un bit
- "001" O"562" X"FF1" sont des littéraux pour les vecteurs de bits
- "chaine" est un littéral de type chaîne de caractères
- null est un littéral pointeur

III - FPGA

III – FPGA : Qu'est ce que c'est ?

Field-Programmable Gate Array - réseau de portes programmables

Ce sont des circuits intégrés programmables (ou plutôt **reconfigurables**) **plusieurs fois** et de plus en plus dynamiquement!

Ils permettent

- Soit d'émuler un circuit afin de le valider avant la gravure silicium
- Soit d'implanter un système complet, System on Chip (SoC)
 - En utilisant la reconfiguration dynamique, ce système peut devenir adaptatif en fonction des besoins (accélération de fonctionnalités choisies).

III – FPGA : Qu'est ce que c'est ?

FPGA vs ASIC

Field Programmable Gate Arrays (FPGAs) et Application Specific Integrated Circuits (ASICs)

D'après XILINX

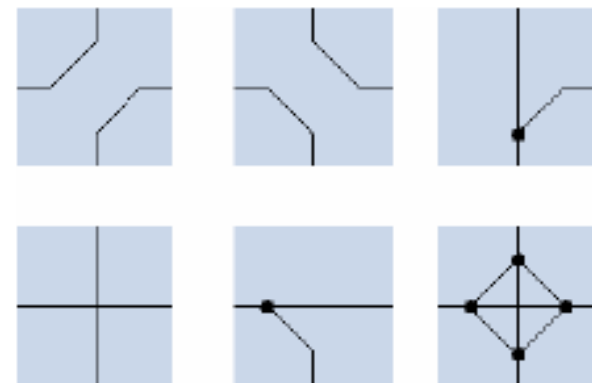
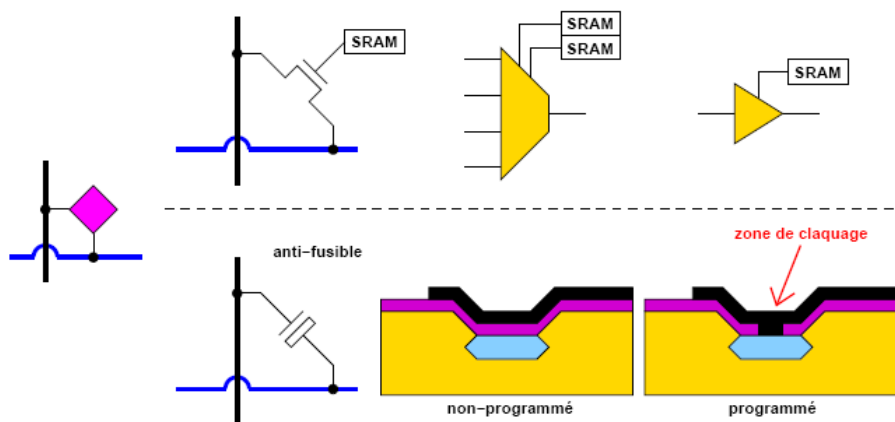
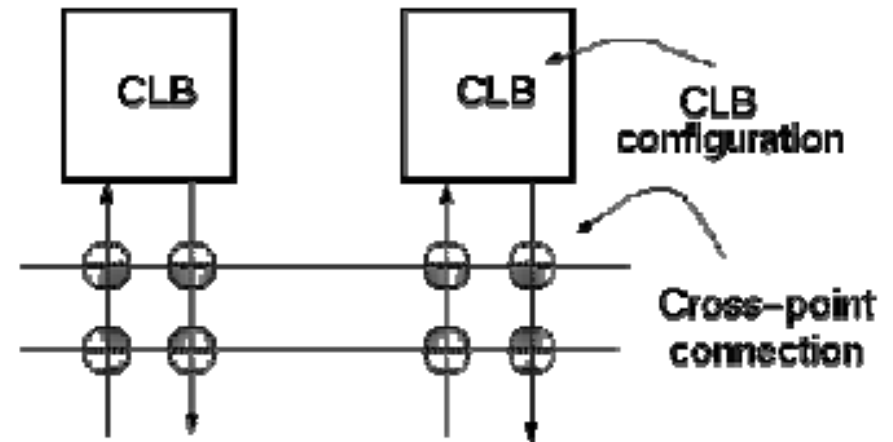
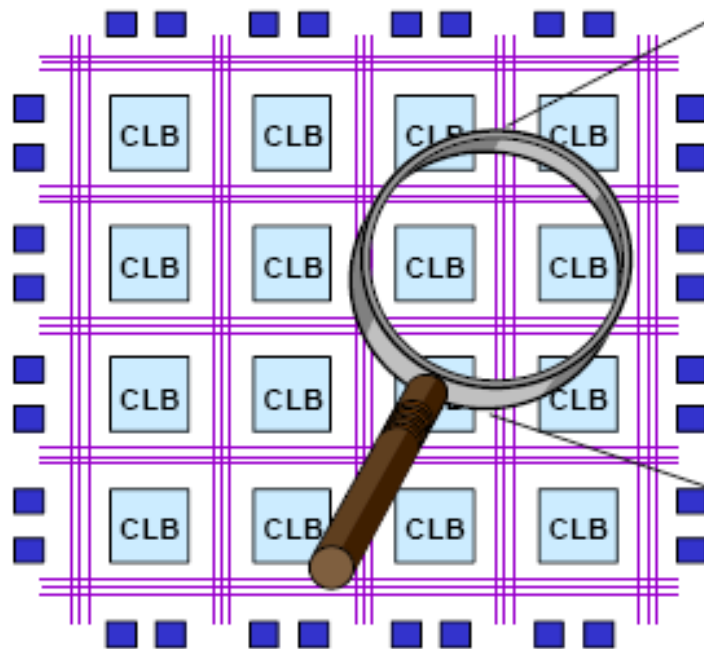
FPGA :

Coût de développement (Delta simulation/réalité), temps d'accès au marché, temps de design réduit, programmable, ...

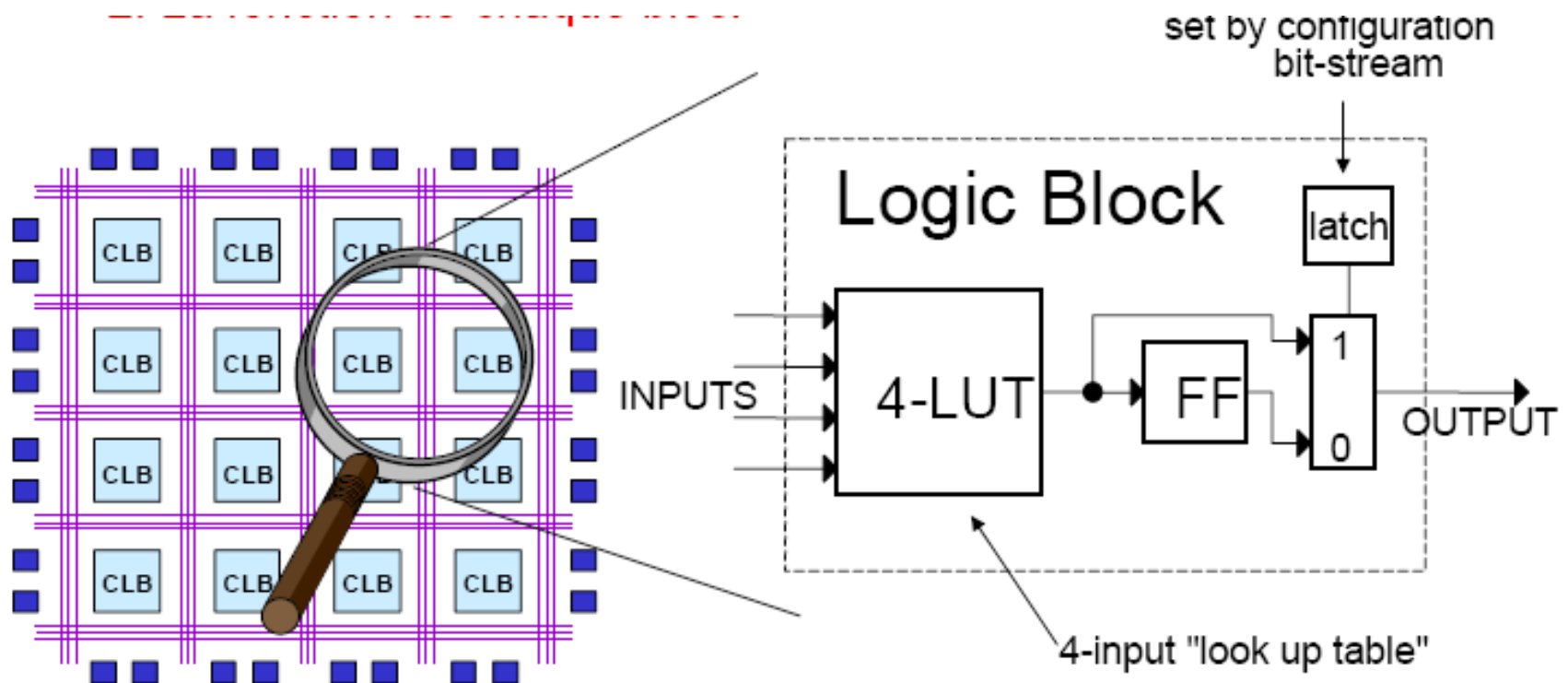
ASIC :

Circuit correspondant complètement au design, plus petit facteur d'échelle (ne contient que le nécessaire), moins cher sur de gros volumes, ...

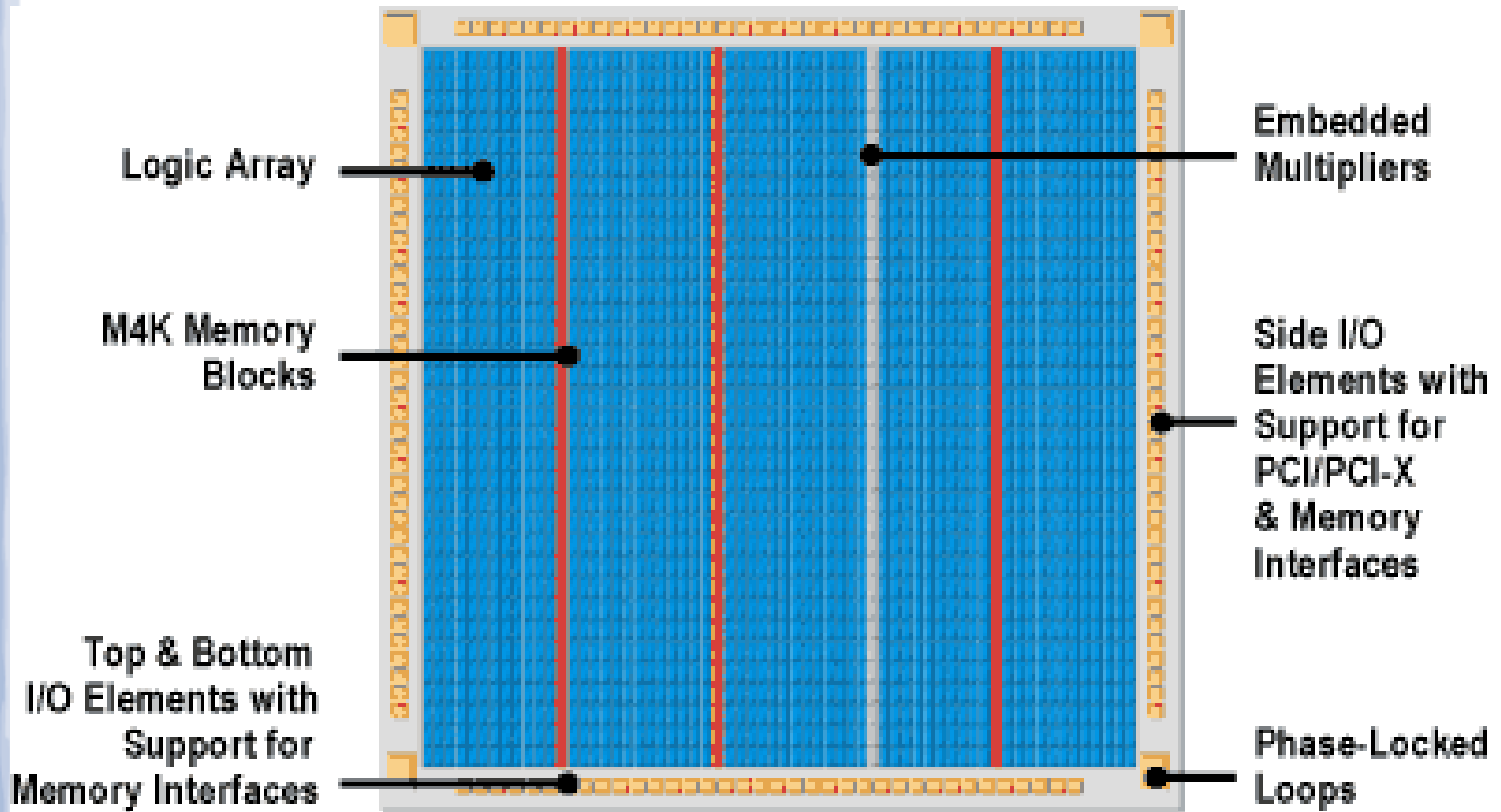
III – FPGA : des CLB – Blocs Logiques (Re)Configurables



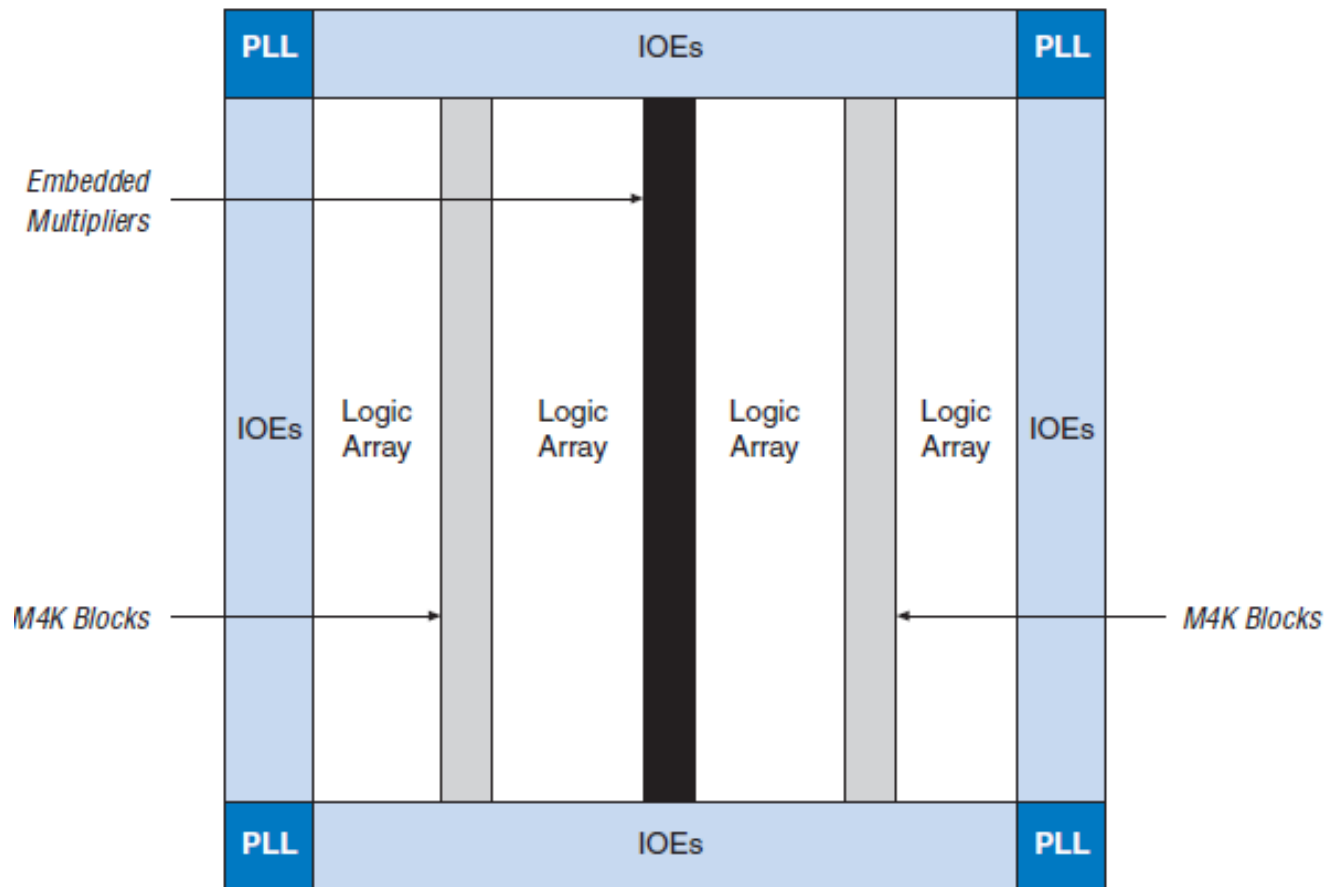
III – FPGA : des CLB – Blocs Logiques (Re)Configurables



III – FPGA : des CLB's – Blocs Logiques (Re)Configurables – mais pas seulement !



III – FPGA : Altera - Cyclone II



III – FPGA : Altera - Cyclone II : Caractéristiques

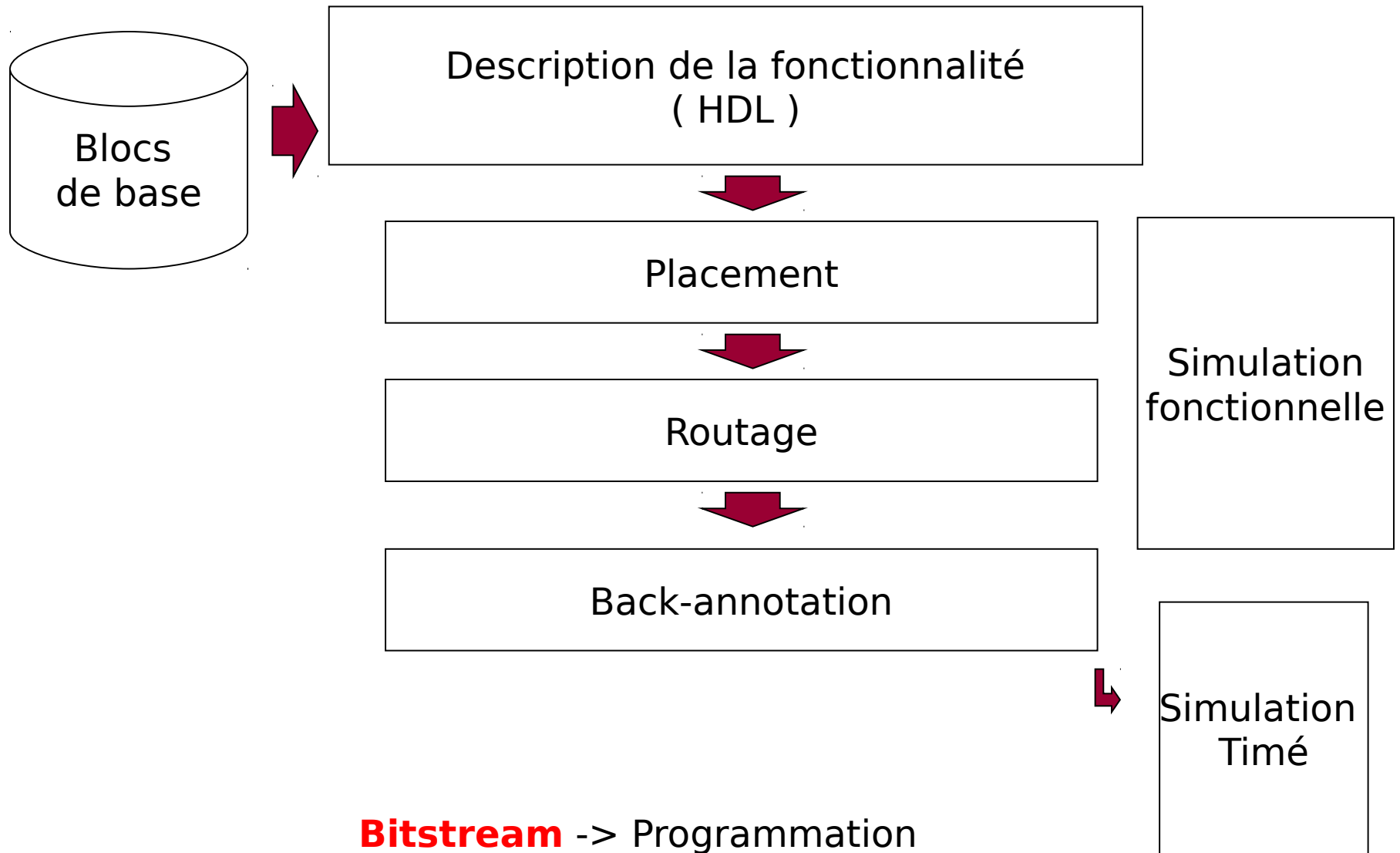
Table 1–1. Cyclone II FPGA Family Features (Part 1 of 2)

Feature	EP2C5 (2)	EP2C8 (2)	EP2C15 (1)	EP2C20 (2)	EP2C35	EP2C50	EP2C70
LEs	4,608	8,256	14,448	18,752	33,216	50,528	68,416
M4K RAM blocks (4 Kbits plus 512 parity bits)	26	36	52	52	105	129	250
Total RAM bits	119,808	165,888	239,616	239,616	483,840	594,432	1,152,000
Embedded multipliers (3)	13	18	26	26	35	86	150
PLLs	2	2	4	4	4	4	4

III – FPGA : Altera - Cyclone II : multiplieurs

<i>Table 2–10. Number of Embedded Multipliers in Cyclone II Devices</i> <i>Note (1)</i>				
Device	Embedded Multiplier Columns	Embedded Multipliers	9 × 9 Multipliers	18 × 18 Multipliers
EP2C5	1	13	26	13
EP2C8	1	18	36	18
EP2C15	1	26	52	26
EP2C20	1	26	52	26
EP2C35	1	35	70	35
EP2C50	2	86	172	86
EP2C70	3	150	300	150

III – FPGA : le flot de conception



III – FPGA : Pin assignment

Il faut préciser sur quelles pattes du circuit les signaux fonctionnels du design vont être branchés

Cela dépend

- Du nombre de pin du FPGA choisi

- De la façon dont il est connecté aux autres éléments de la carte (PCB)

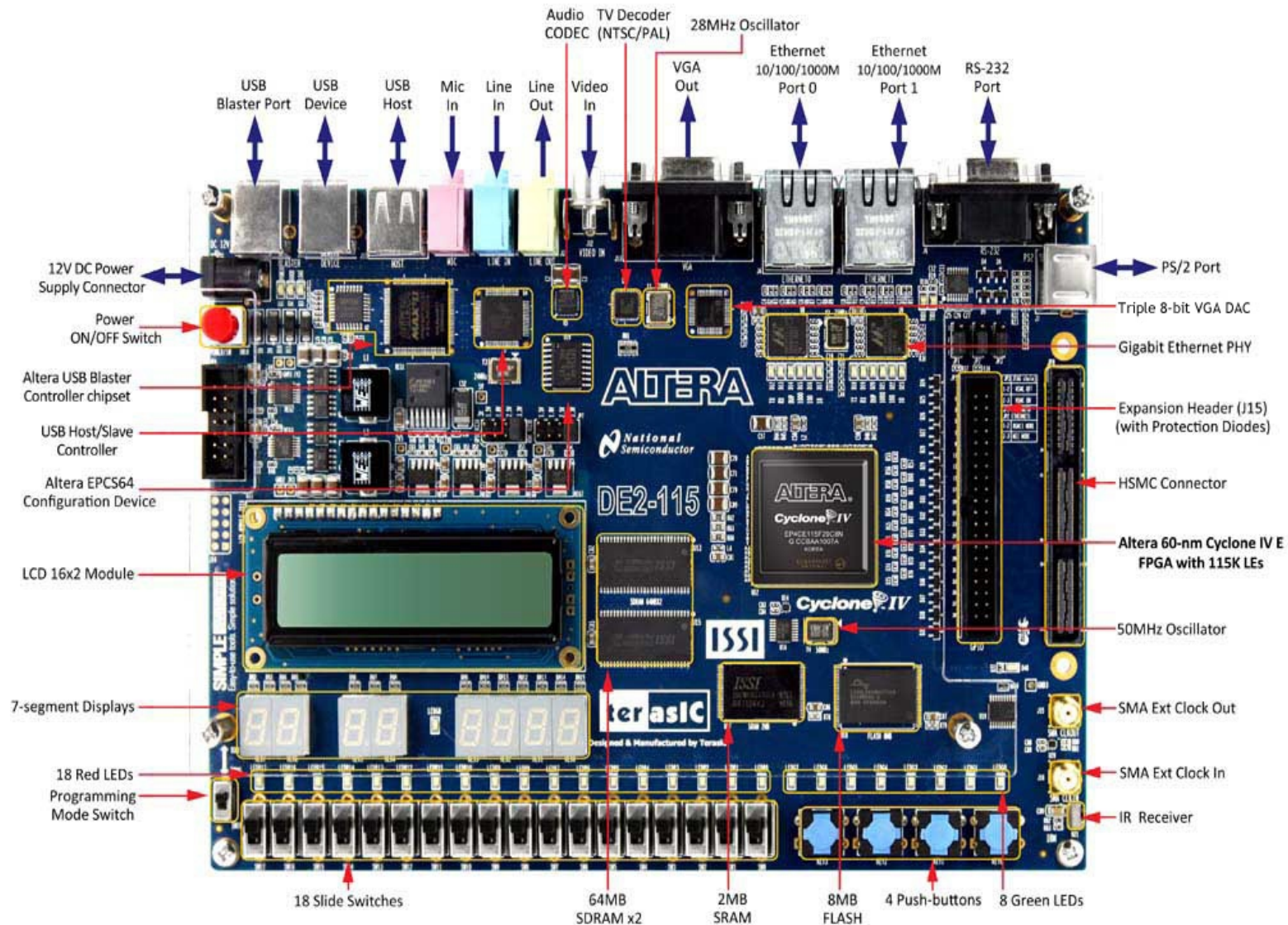
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

-- Simple module that connects the SW switches to the LEDR lights
ENTITY part1 IS
    PORT ( SW    : IN    STD_LOGIC_VECTOR(17 DOWNTO 0);
          LEDR   : OUT   STD_LOGIC_VECTOR(17 DOWNTO 0)); -- red LEDs
END part1;

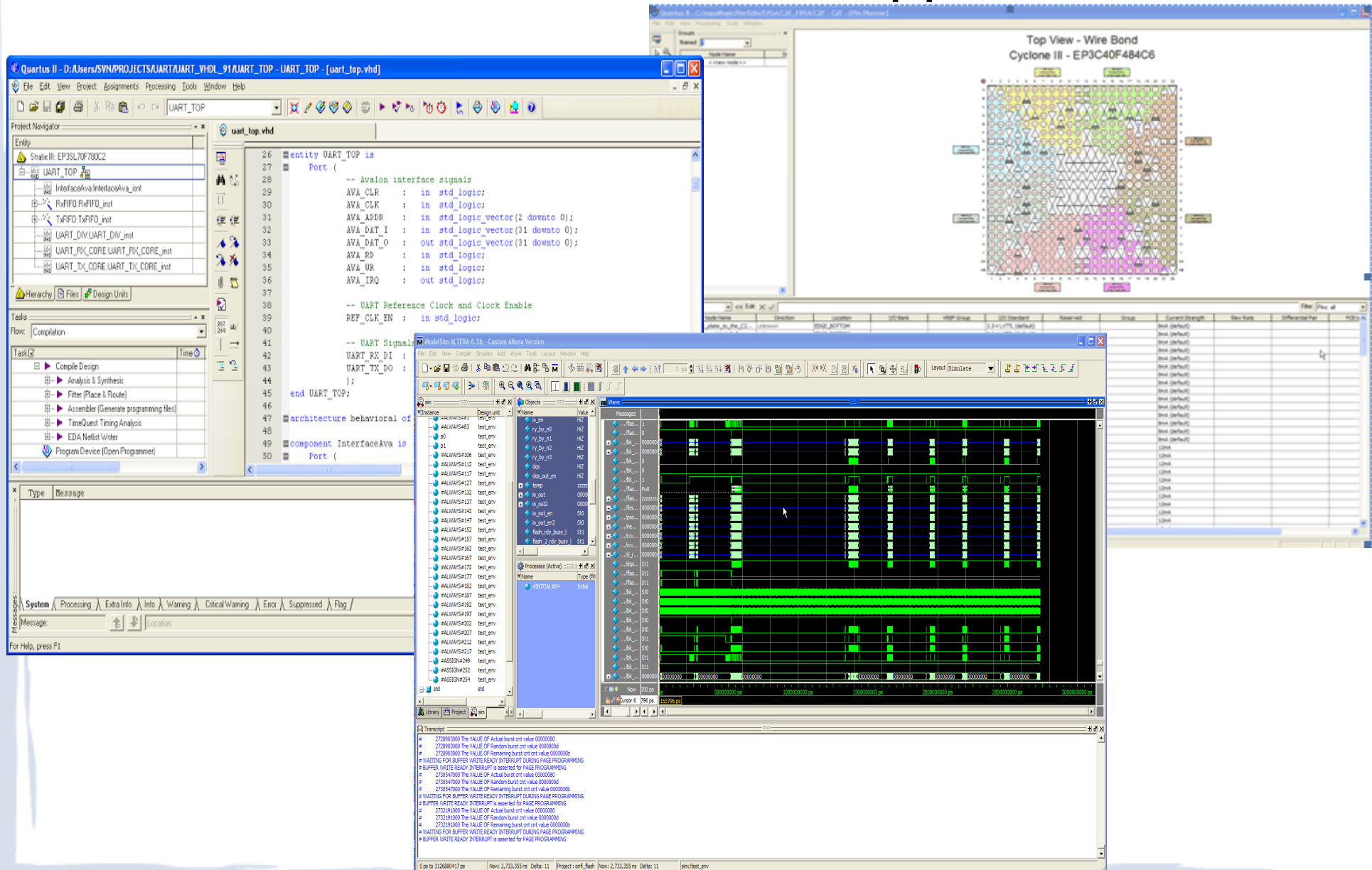
ARCHITECTURE Behavior OF part1 IS
BEGIN
    LEDR <= SW;
END Behavior
```

```
LEDR(17) <= SW(17);
LEDR(16) <= SW(16);
...
LEDR(0) <= SW(0);
```

III – FPGA : Environnement de développement - Carte de développement DEII et DEII-115



III – FPGA : Environnement de développement - IDE et outils de développement



Organisation Cours/TPs/Projet

- ★ 8 séances de CM – dont 1-2 séances finales dédiées au Projet et préparation de l'Exam
- ★ 8 ou 7 séances de Tps
 - ★ Préparation des différents blocs logiques (ALU, Registres, ...)
 - ★ Contrôleurs – FSM
 - ★ Projet (assemblage et extension des blocs précédents) + définition d'un ISA + instanciation mémoire

Biblio

Documents de cours

- Ce cours (bientôt) en ligne à :

[Http://perso-etis.ensea.fr/rodriguez/](http://perso-etis.ensea.fr/rodriguez/)

- Un très bon support de cours

<http://hdl.telecom-paristech.fr/index.html>

- Le cours de Licence 2 (en particulier pour ceux qui ne l'ont pas suivi):

http://perso-etis.ensea.fr/miramond/Enseignement/L2/circuits_numeriques.html

- Le cours de Licence 3 de 2013 :

http://perso-etis.ensea.fr/miramond/Enseignement/L3/Cours_VHDL_2011.html

Documents de développement

- Quartus

<http://quartushelp.altera.com/current/>

- Documentation Altera sur les Cyclones II et IV (entre autre...)

<http://www.altera.com/literature/lit-index.html>