Conception & Simulation Vhdl

Pascal BENOIT

Contact:

Pascal.Benoit@univ-montp2.fr

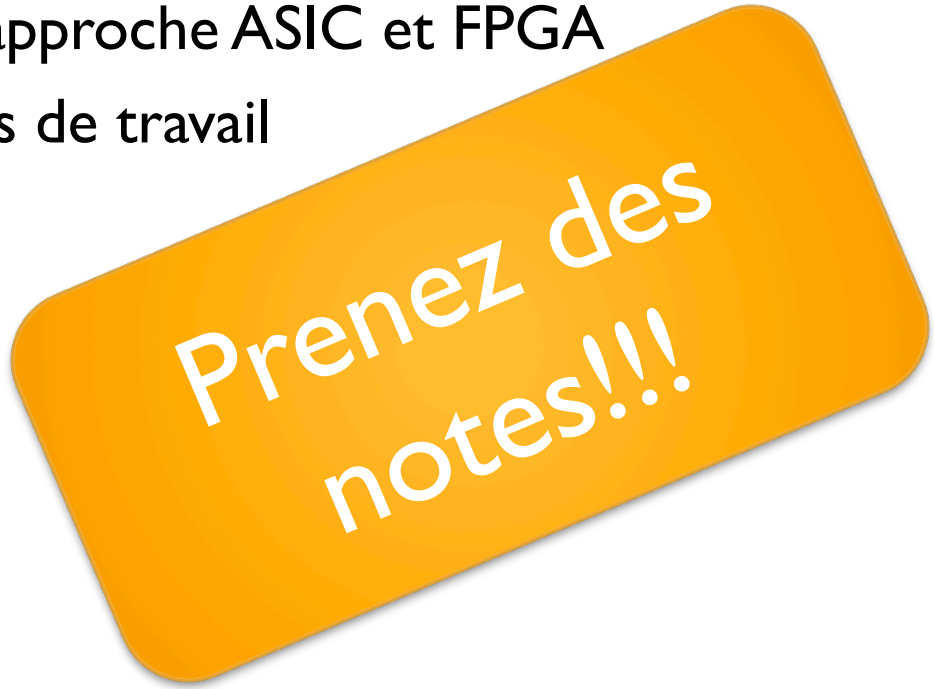
UNIVERSITÉ MONTPELLIER 2
SCIENCES ET TECHNIQUES

Organisation de l'enseignement du VHDL

- Cours magistral (1 séance de 1h30)
  - Introduction
- Cours / TD (7 séances de 1h30)
  - Conception de composants simples
  - Simulation
- TP (5 séances de 3h)
  - Conception de composants complexes
  - Synthèse sur cible FPGA
- MINI-PROJET (7 séances de 4h)
  - P2: Circuit intégré sur cible Silicium
  - P9: Système embarqué sur cible FPGA

Objectif du cours aujourd'hui

- Comprendre
  - la finalité
  - la différence entre Fabrication/Conception
  - ce qu'est un flot de conception
  - le rôle des HDL
  - les différences entre l'approche ASIC et FPGA
  - la méthode et les outils de travail



La finalité?


A13 – Paris
VOIE GEORGES POMPIDOU
AVENUE DES CHAMPSS-ELYSÉES
Options
NAVIGON


Sprint
10 08
Seattle
Messages
Voicemail
People
Mail
Market
Camera
Phone

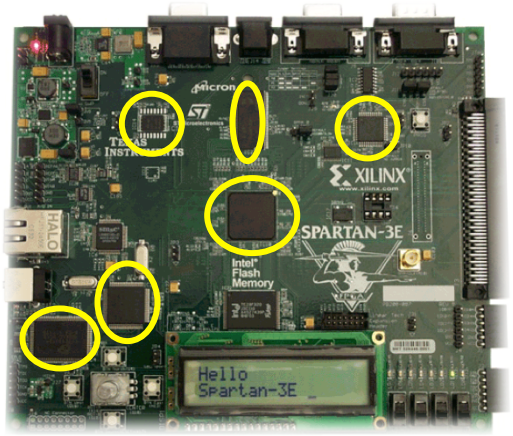

SONY
Cyber-shot
G


The New York Times

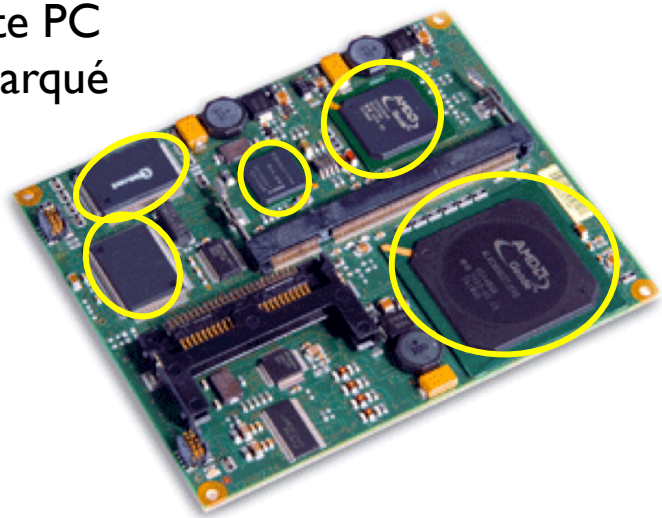

91.7

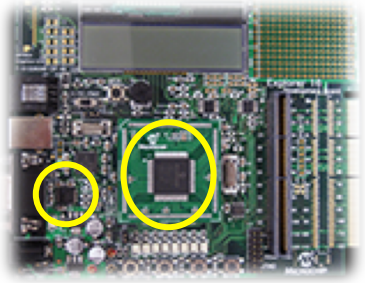
La finalité?

Carte FPGA

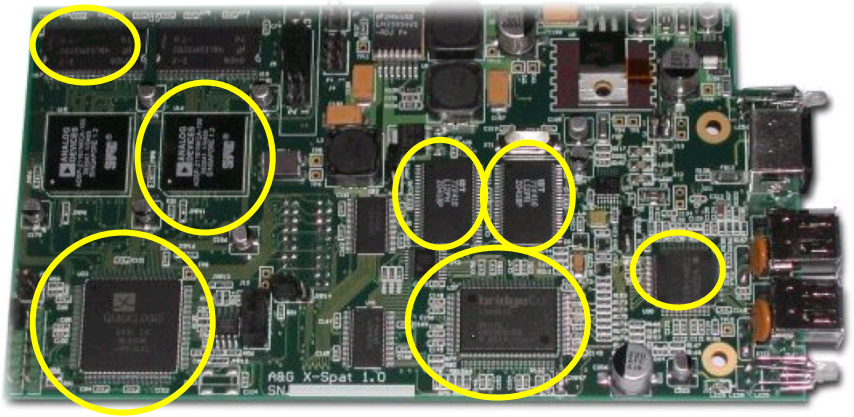


Carte PC embarqué





Ordinateur de bord

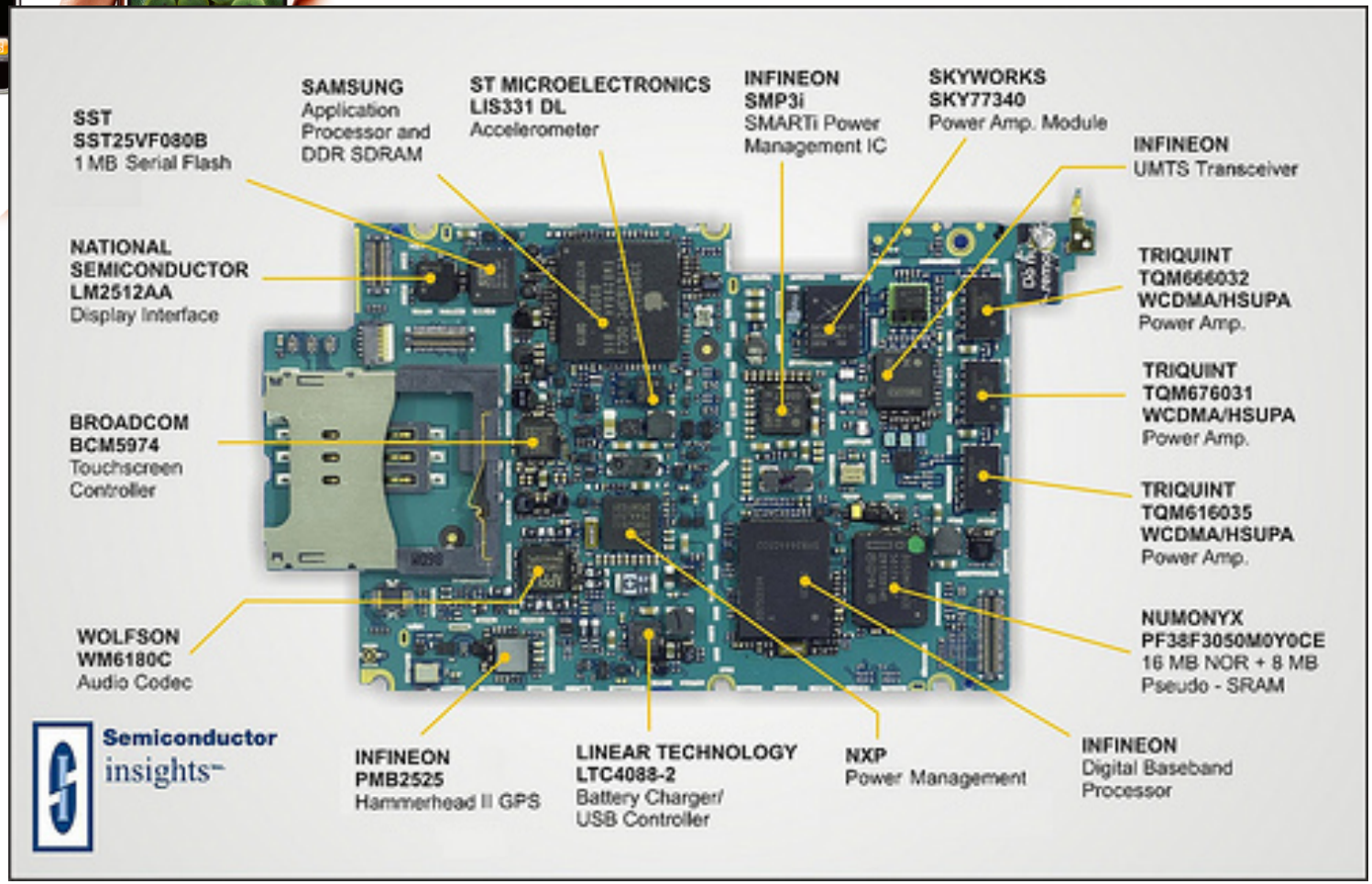


Carte DSP

Circuits intégrés numériques

La finalité?

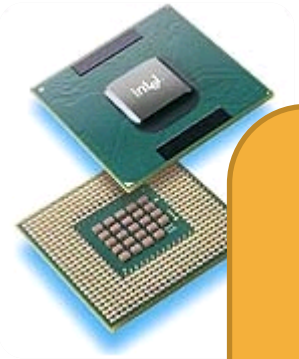
Un exemple: iPhone 3G

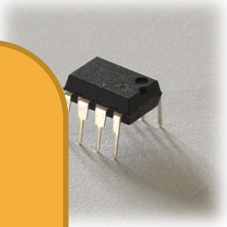


Qu'ont-ils en commun?

Microprocesseurs

Interfaces USB,
Ethernet, SPI, …





Hardware Description Language

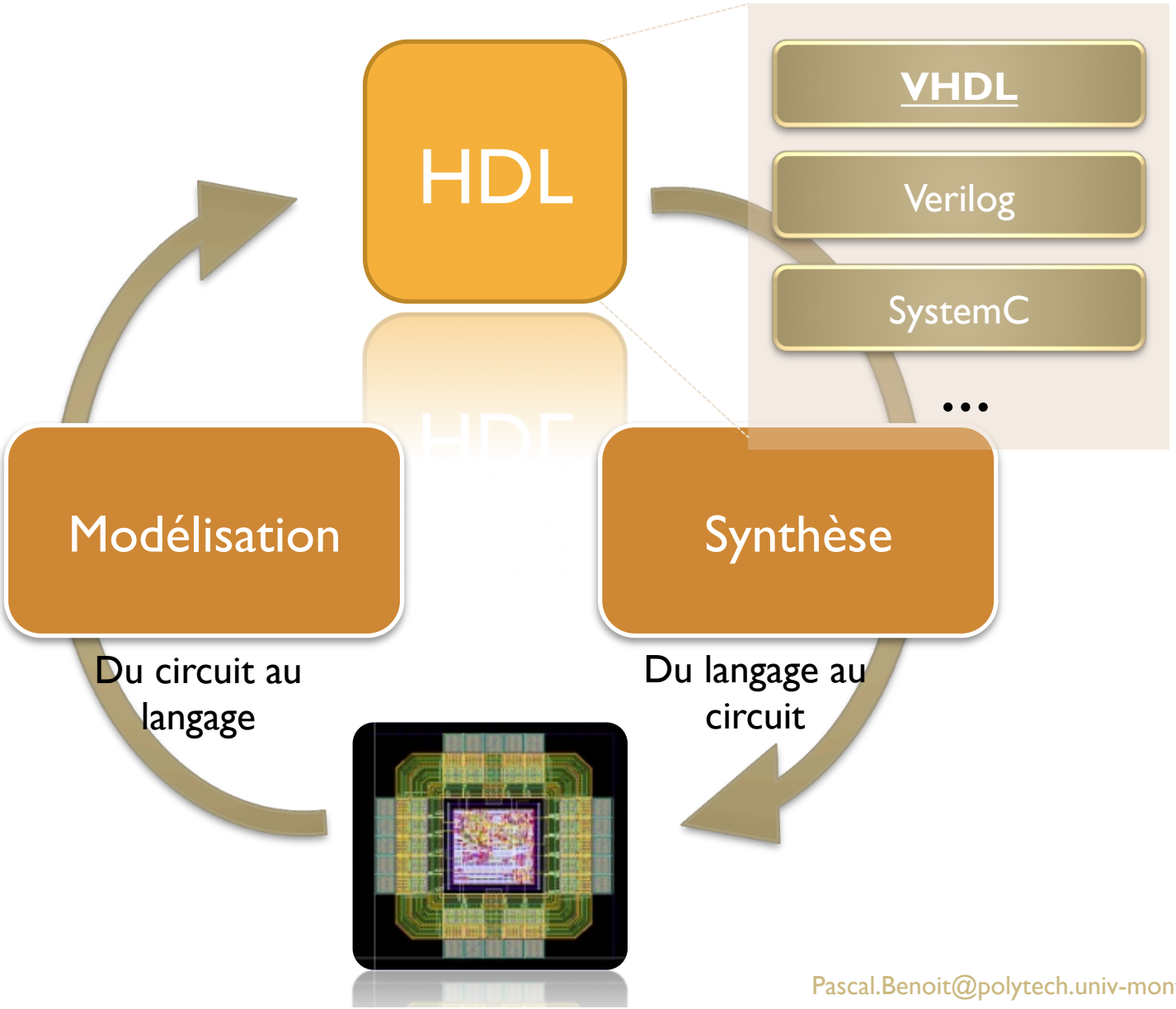
HDL

Langage de description matériel

FPGA

Périphériques

Les « Hardware Description Languages »

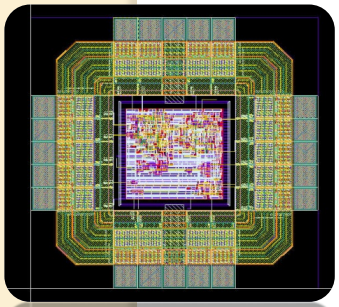


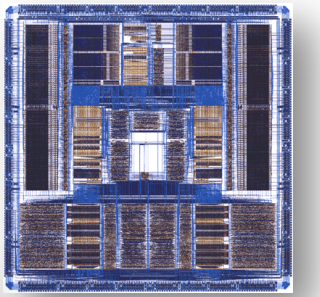
HDL, des langages pour…

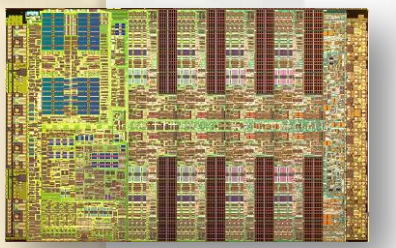
HDL

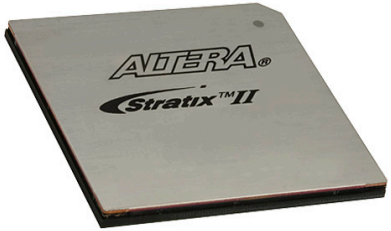
Concevoir des circuits intégrés

« Programmer » des circuits FPGA



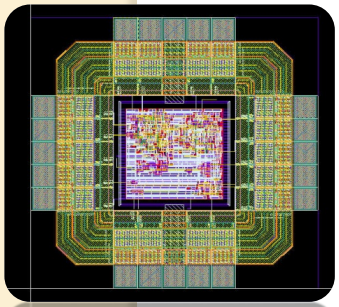


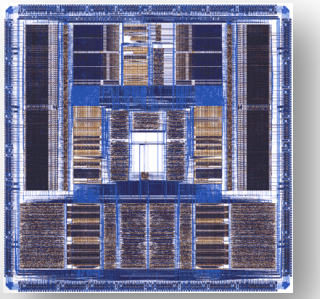


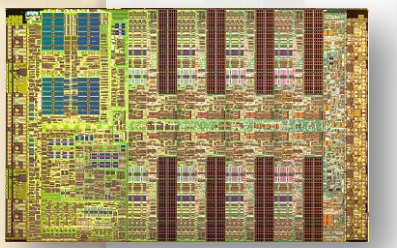


HDL, des langages pour…

HDL







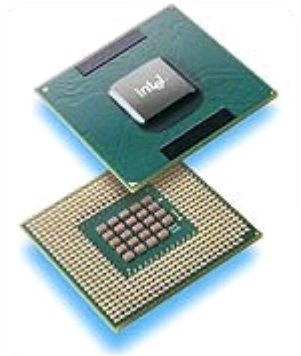
Quoi ?

Pourquoi ?

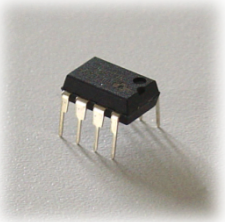
Comment ?

Concevoir des circuits intégrés

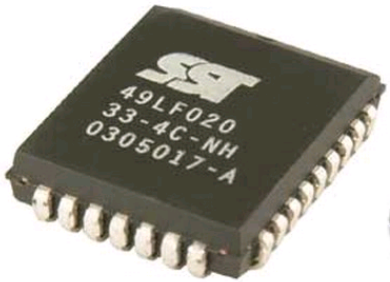
Microprocesseurs



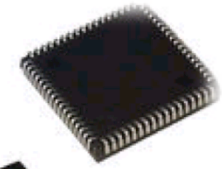
Interfaces USB,
Ethernet, SPI, …



Processeurs spécialisés
(Télécom, CoDec, etc.)



DSP

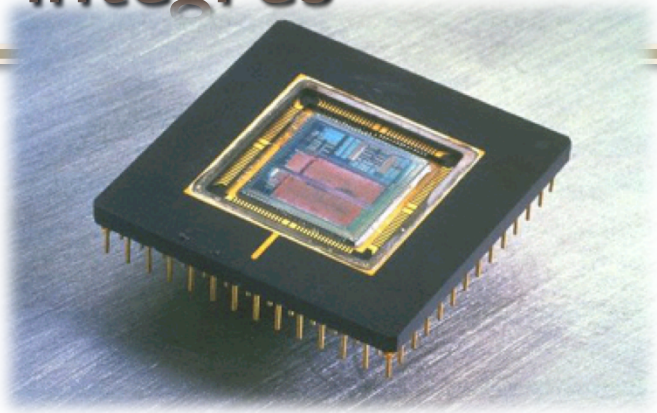


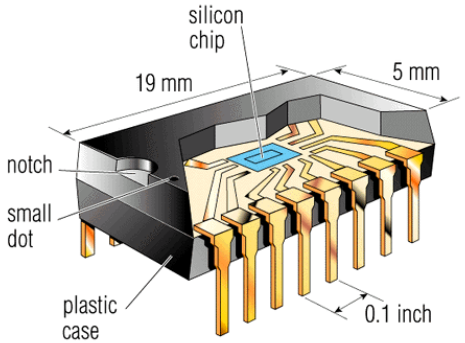
FPGA

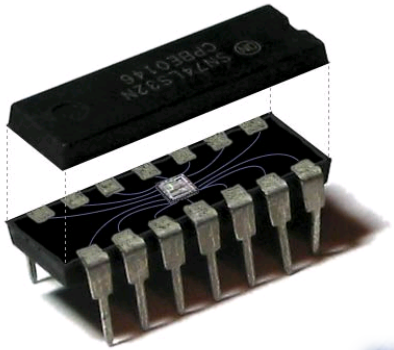
Périphériques

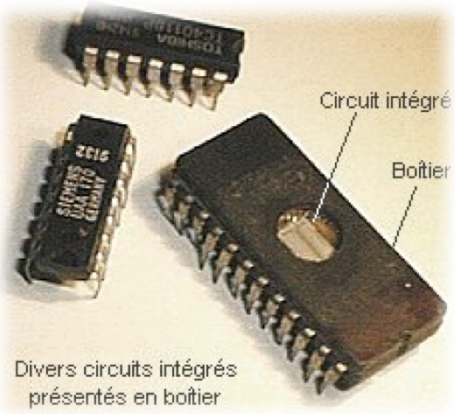
Concevoir des circuits intégrés

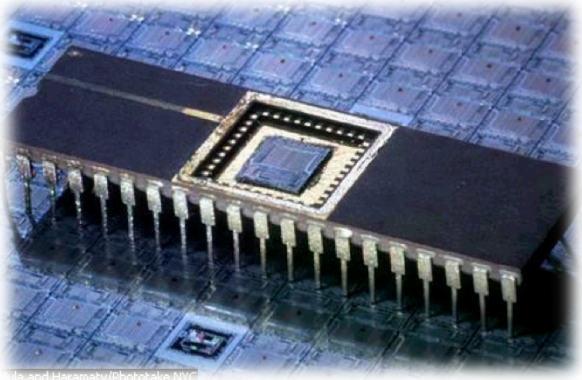
Zoom sur le cœur des circuits intégrés…





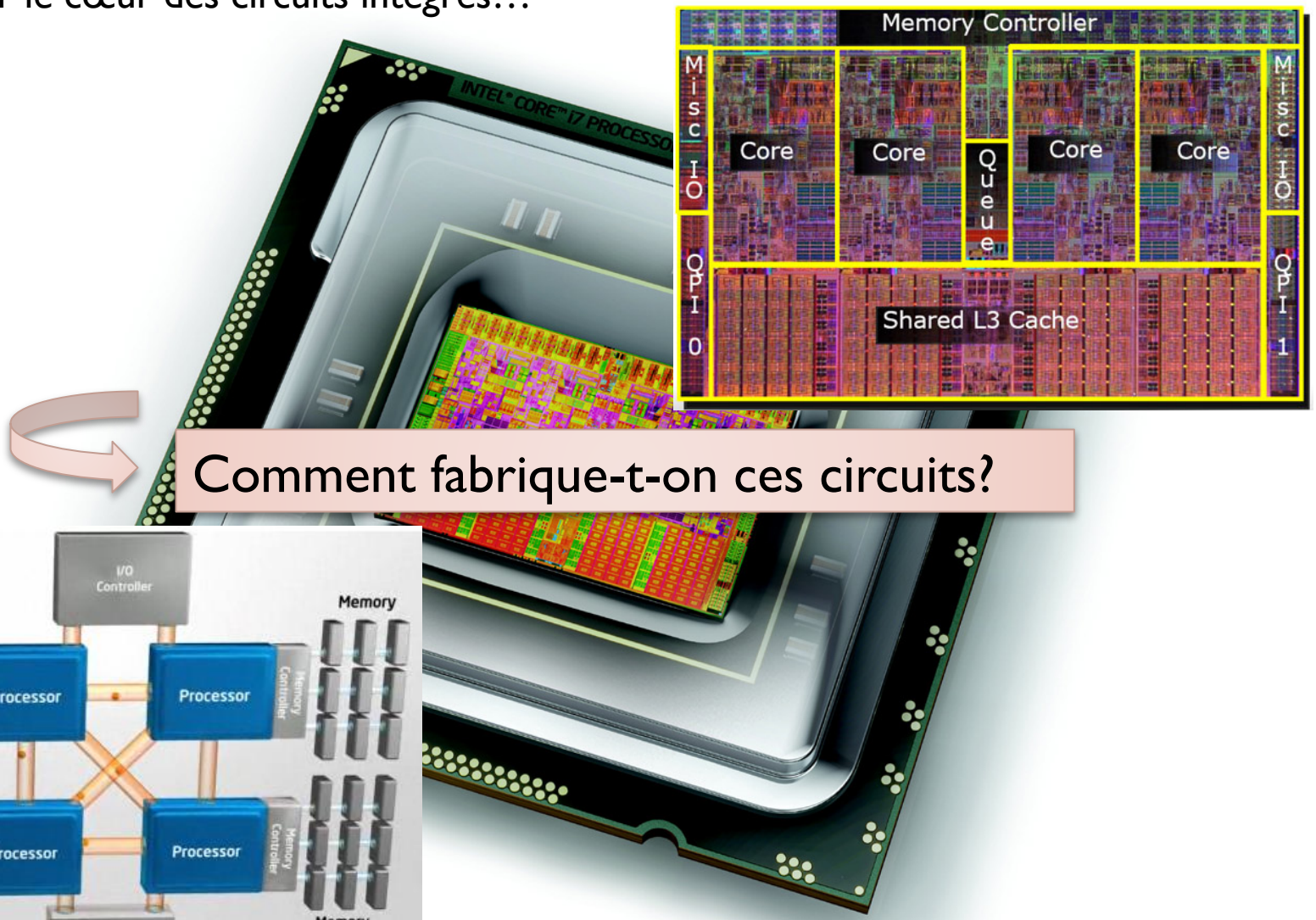






Concevoir des circuits intégrés

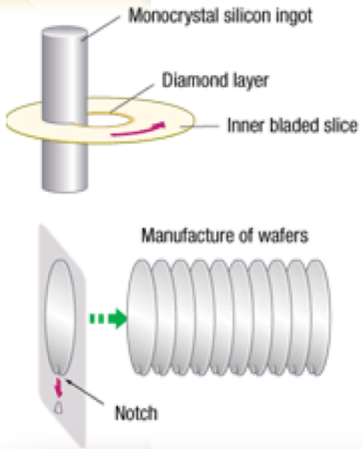
Zoom sur le cœur des circuits intégrés…

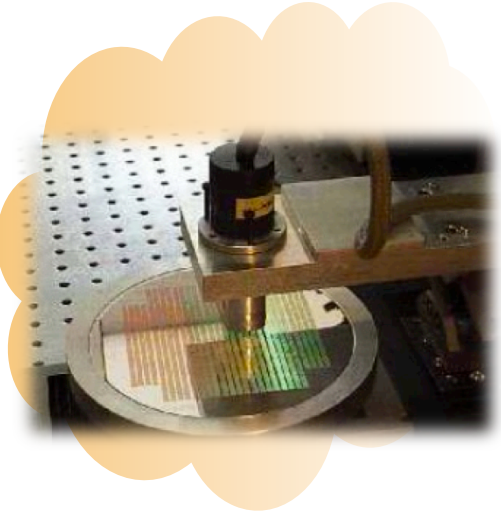


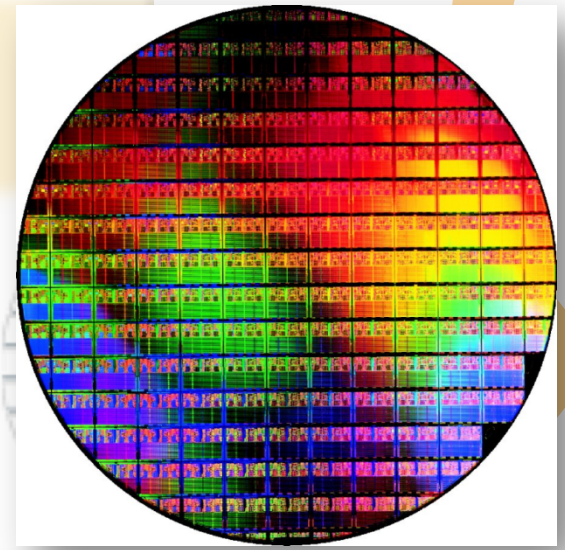
Comment fabrique-t-on ces circuits?

Processus de fabrication

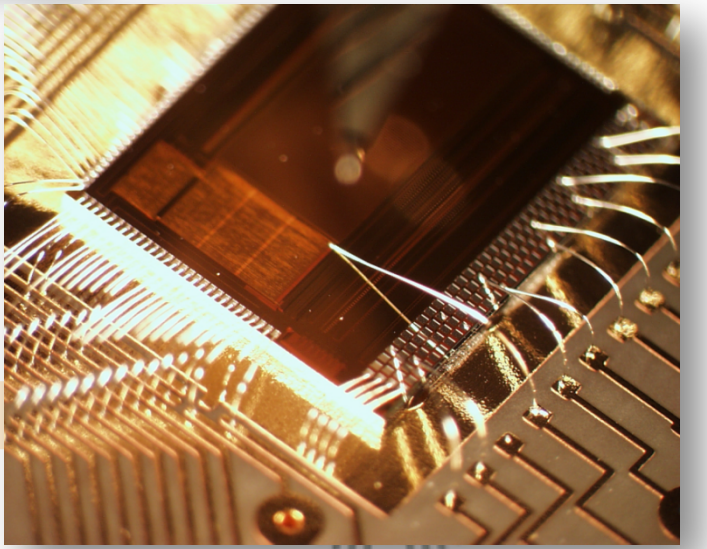
Découpage du Wafer



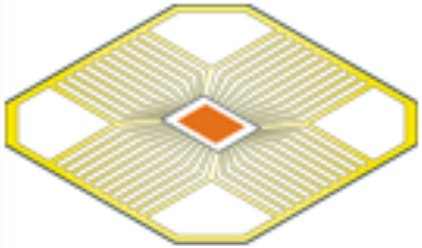




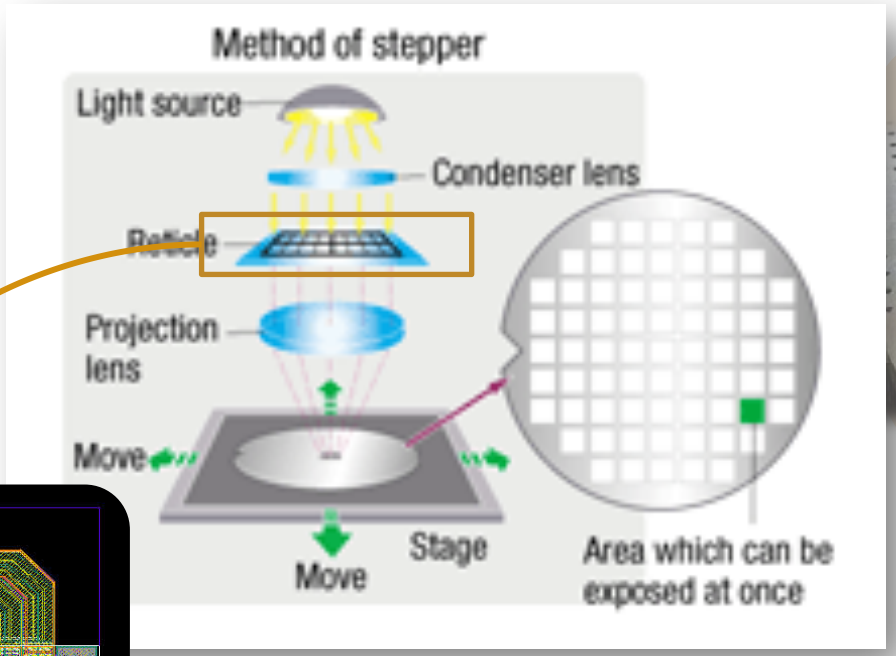
Photolithographie

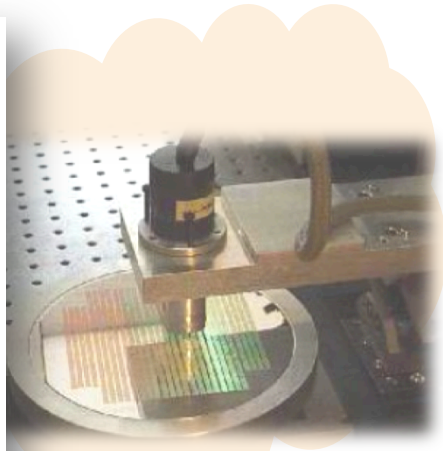


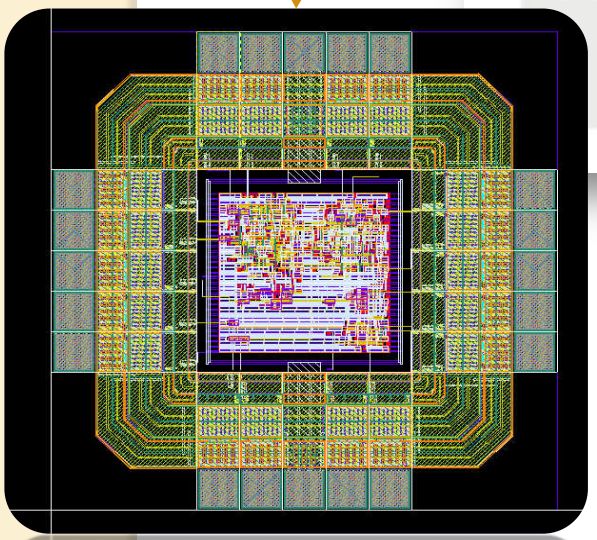
Bonding



Processus de fabrication







Dessin des masques

Ingénieur ERII!

Conception & Fabrication

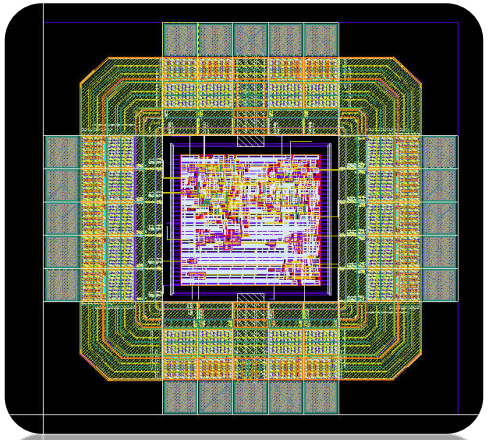
Cahier des charges /
Spécifications du circuit

Processeur
Multimédia
…
F = 500 MHz
P = 1 Watt
…

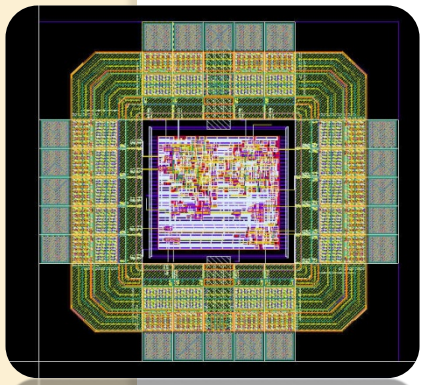
Processus de
Conception

Ingénieurs

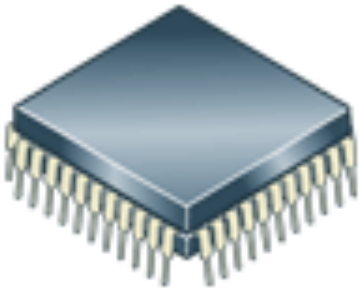
Dessin des masques



Fondeur



Processus de
Fabrication



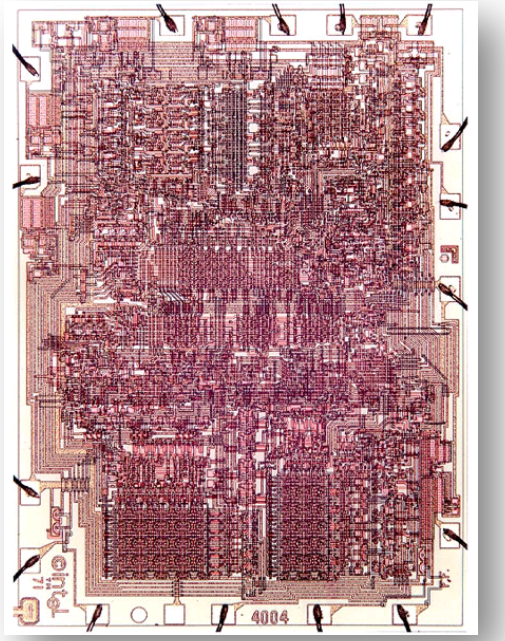
Circuit

Test de
production

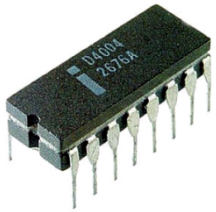
Produit Final

Evolution des méthodes de Conception

Intel 4004 10um (10000nm)



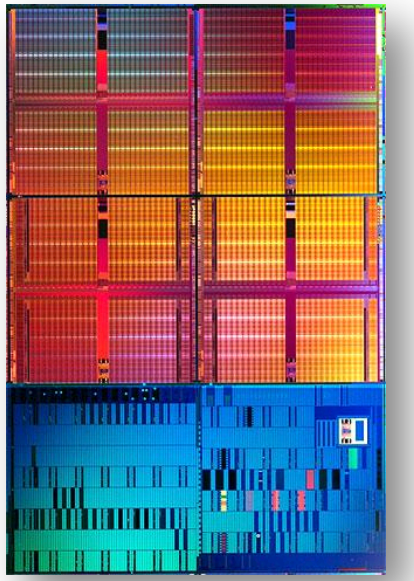
2300 transistors



1971

37 ans plus tard…

Intel Penryn 45nm

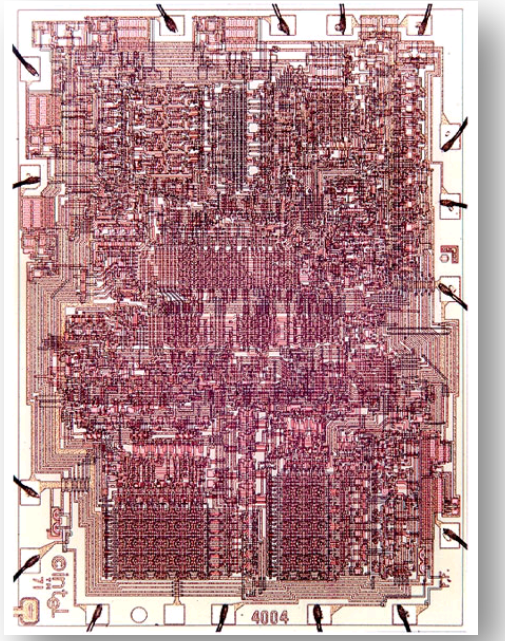


410 millions de transistors!

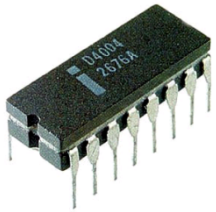
2008

Evolution des méthodes de Conception

Intel 4004 10um (10000nm)



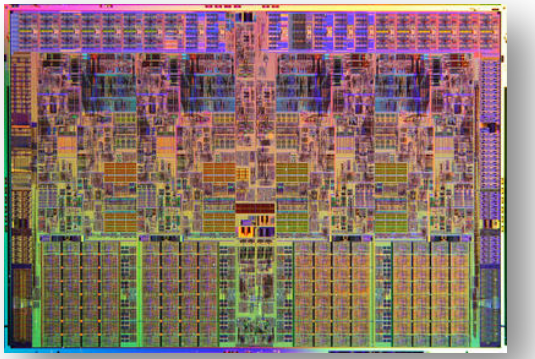
2300 transistors



1971

38 ans plus tard…

Intel Core i7 45nm



731millions de transistors!!!

2009

Evolution des méthodes de Conception

Intel 4004 10um (10000nm)

nsistors

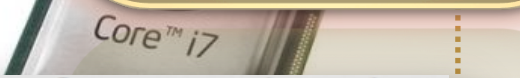
39 ans plus tard…

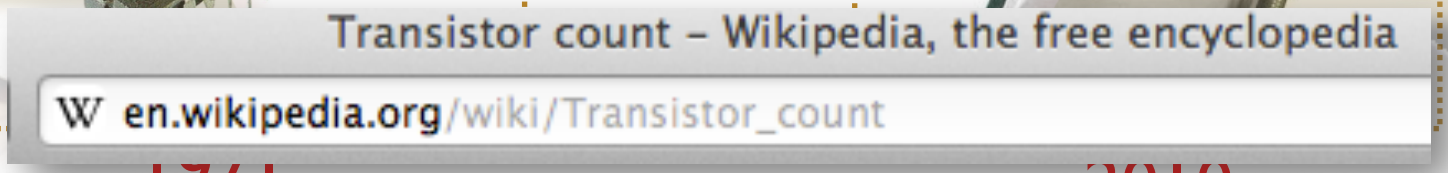
Intel Core i7 32nm

1170 mill

Dessin des masques « à la main »

Conception Assistée par Ordinateur

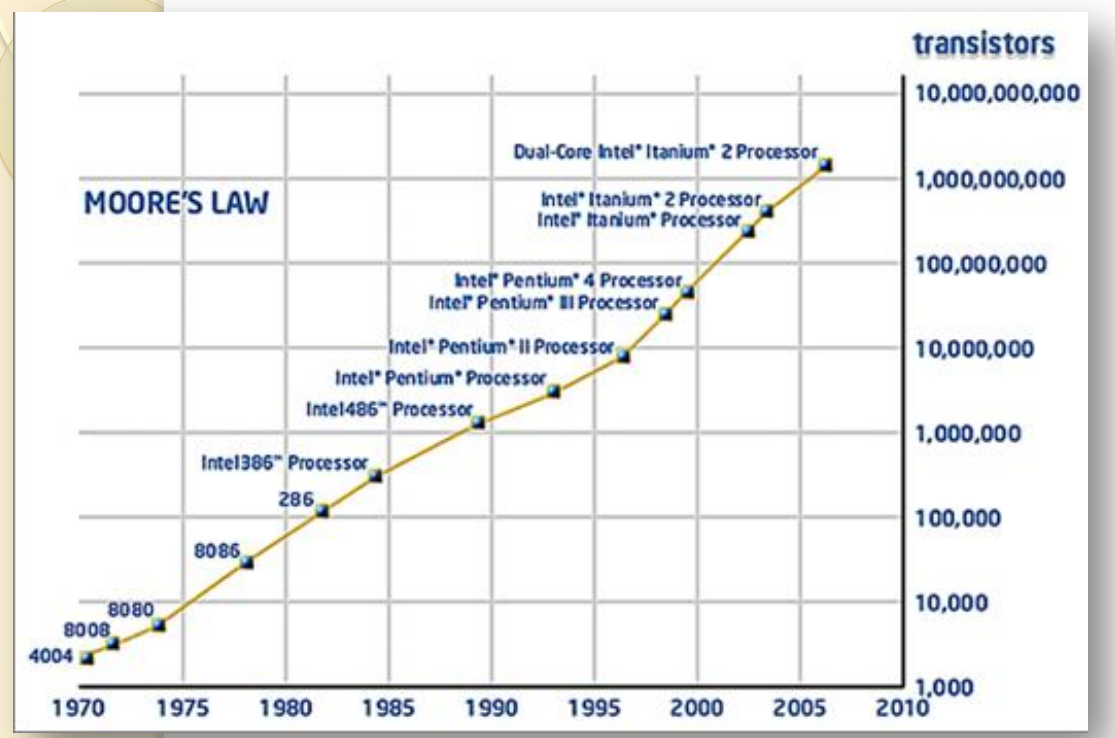




1971

2010

Loi de Moore et conséquences…



Petit calcul…

Le masque d'1 transistor, c'est grossièrement l'intersection de 2 segments de droites perpendiculaires. Si on suppose qu'il faut 10s pour tracer à la règle ces 2 segments, combien de temps faudrait-t-il pour dessiner 1 milliard de transistors?

Réponse

115740 jours, soit 317 ans à temps plein, rien que pour le dessin des masques, sans compter le temps qu'il faudrait pour définir les fonctionnalités du circuit, puis la manière de connecter les transistors, et vérifier qu'il n'y a pas d'erreurs sur le dessin!

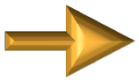
Comment faire pour concevoir plus rapidement un circuit intégré?

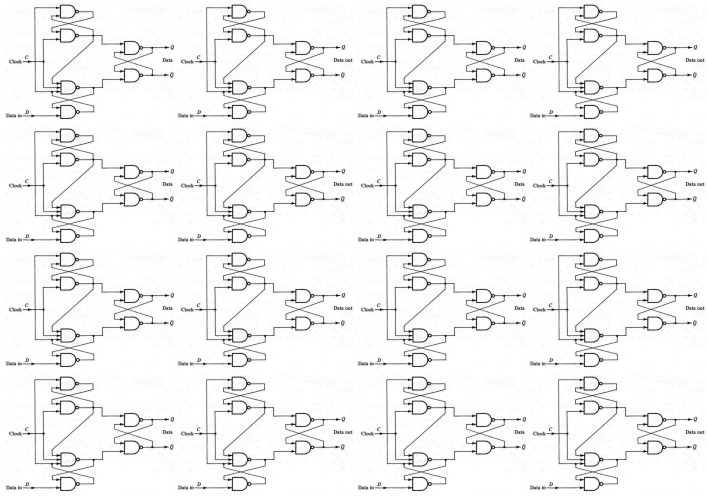
ABSTRACTION ***AUTOMATISATION***

Exemple: registre 16 bits

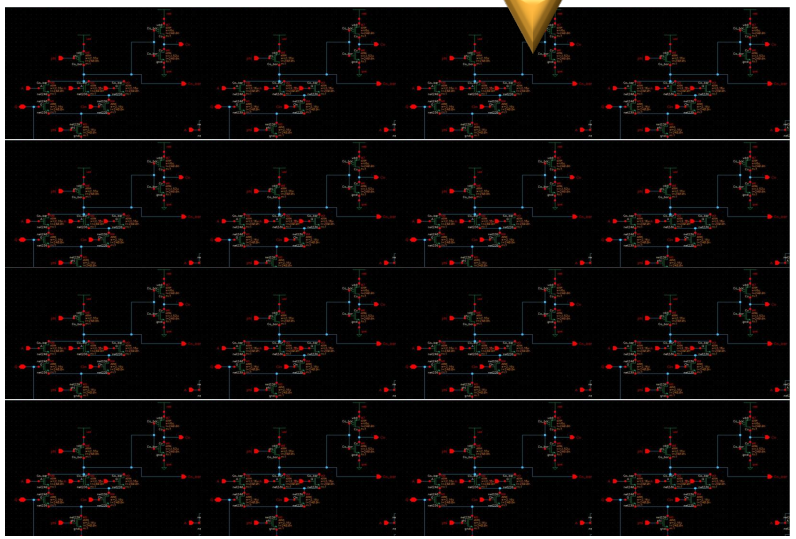
```
library ieee ;
use ieee.std_logic_1164.all;
entity dff isport(data_in:in std_logic_vector(15 downto 0);
clock:in std_logic;
data_out:out std_logic_vector(15 downto 0));
end dff;
 architecture behv of dff is
begin
process(data_in, clock)
begin
if (clock='1' and clock'event) then
data_out <= data_in;
end if;
end process;
end behv;
```

15 lignes de VHDL



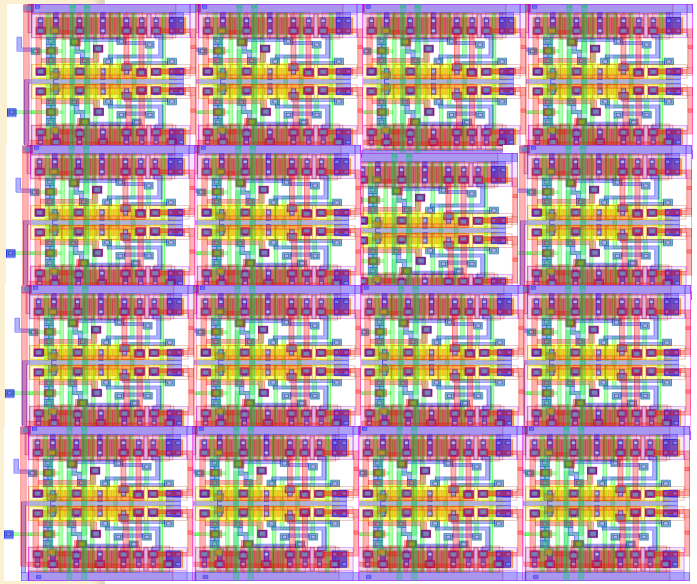


~ 96 portes logiques



~ 384 transistors

~ 768 segments



Un exemple d'abstraction…

- Registre simple16 bits…
  - Description VHDL ➔ 15 lignes de code
  - Portes logiques ➔ 96 portes logiques
  - Circuit électrique ➔ 384 transistors
  - Dessin des masques ➔ 768 « segments »
- En résumé, l'abstraction permet de décrire plus simplement des choses compliquées…
- Outils CAO pour automatiser le passage d'un niveau d'abstraction à un autre
  - Outils de **synthèse**
  - Attention: ce n'est pas magique…
  - Vérification: outils de **simulation**, etc.

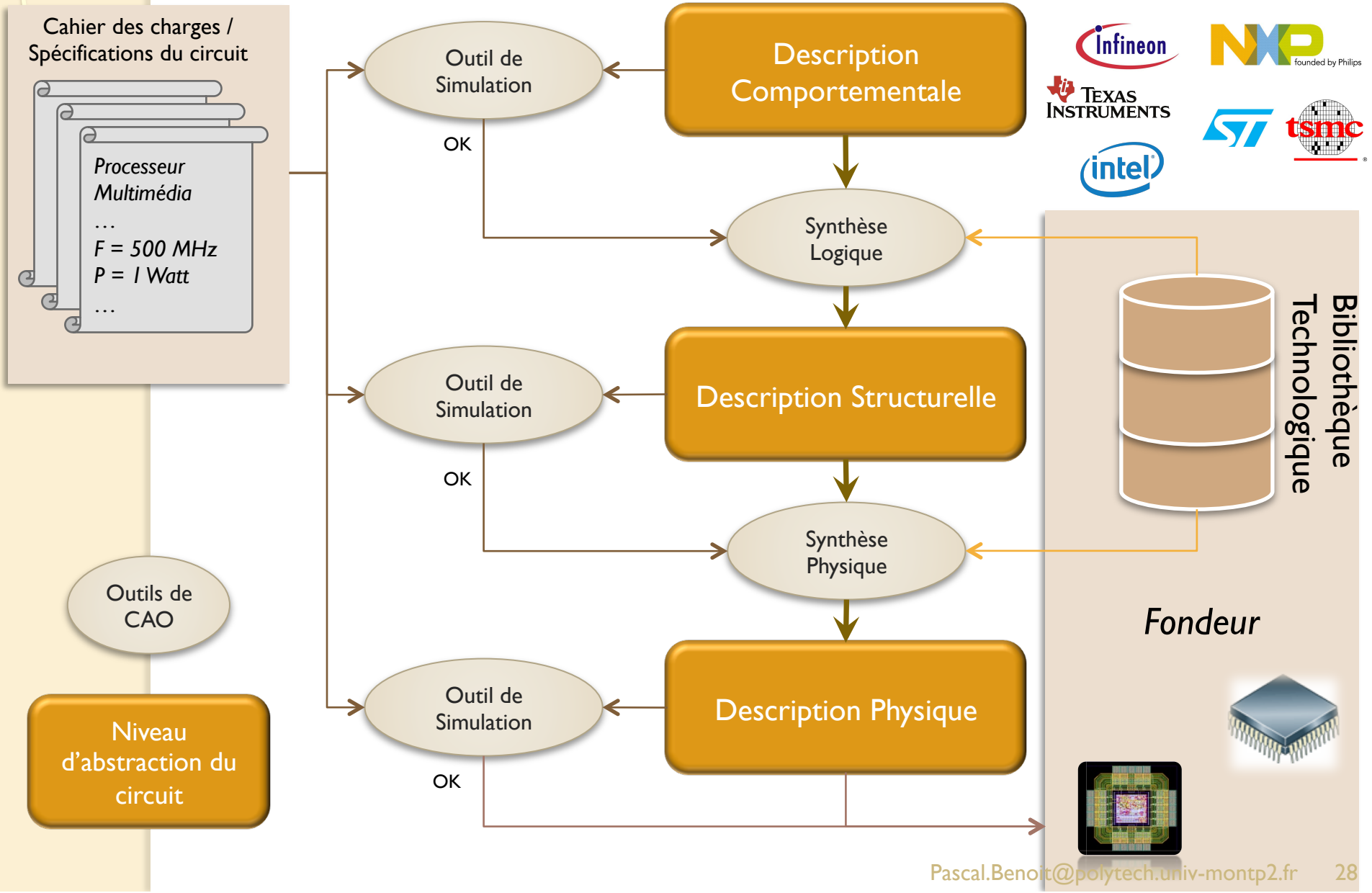
Conséquences…

iPod

Téléphones Mobiles

Gameboy

Flot de Conception Silicium

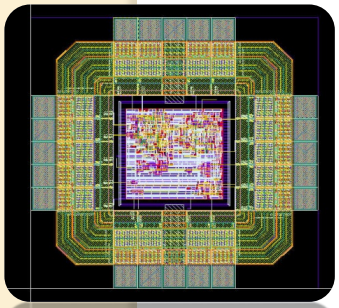


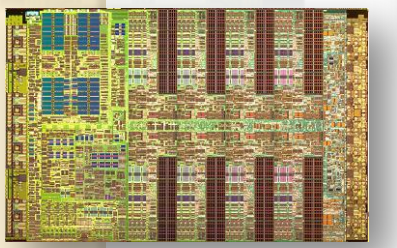
HDL, des langages pour…

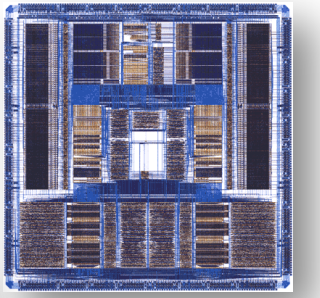
HDL

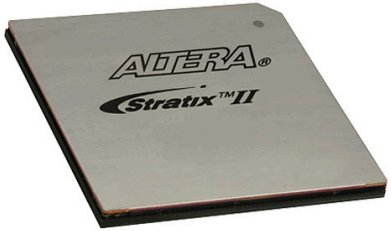
Concevoir des circuits intégrés

« Programmer » des circuits FPGA









HDL, des langages pour…

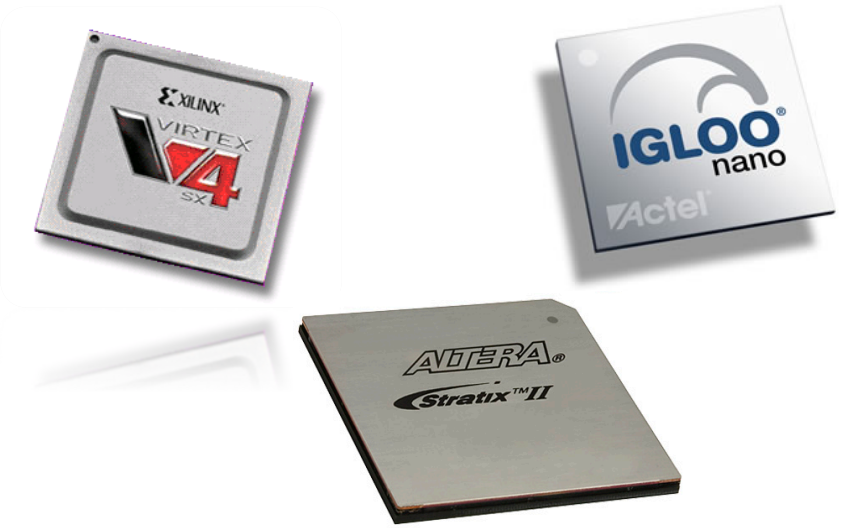
HDL

« Programmer » des circuits FPGA

Quoi ?

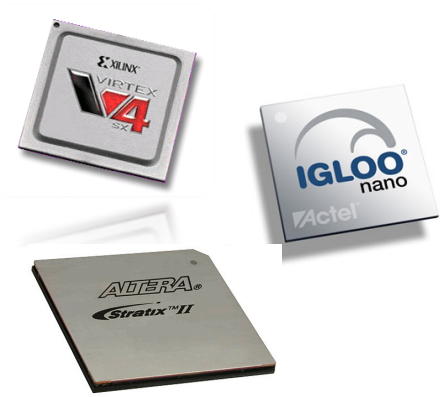
Pourquoi ?

Comment ?



Composants logiques (re-)programmables

- FPGA
  - **FPGA** : de l'ordre de 1000M de transistors
  - Pour les plus complexes, > 3G transistors…

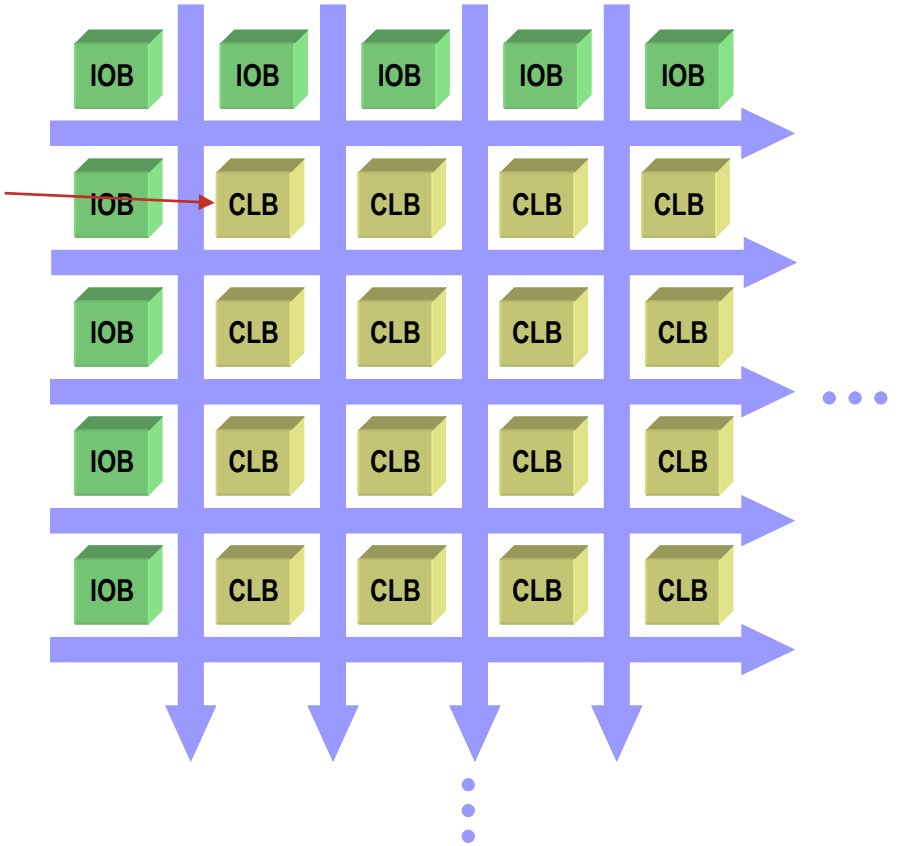


- Programmation de fonctions logiques complexes pour
  - **Gagner du temps** lors de la conception
  - De **faibles volumes** de productions
  - La réalisation du **premier prototype**

Architecture des FPGA

- Field Programmable Gate Array
  - Tableau de blocs logiques programmables interconnectés entre eux (et vers les entrées/sorties) par des canaux de routage

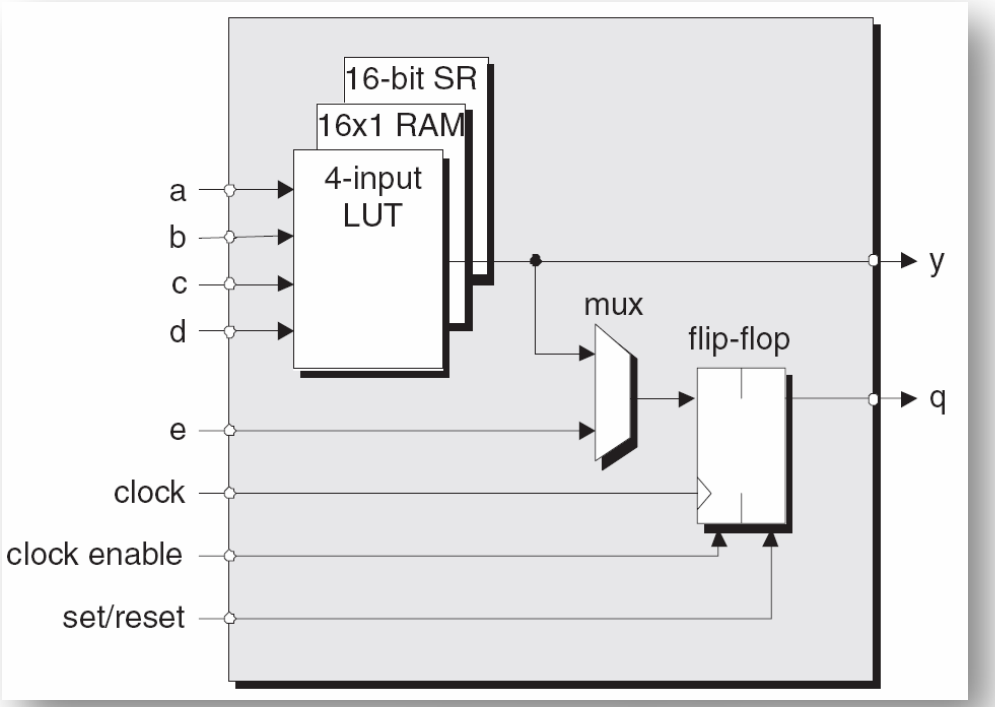
Cellule élémentaire: Configurable Logic Block



Principaux fabricants: *Xilinx, Altera, Actel, Lattice, …*

Architecture des FPGA

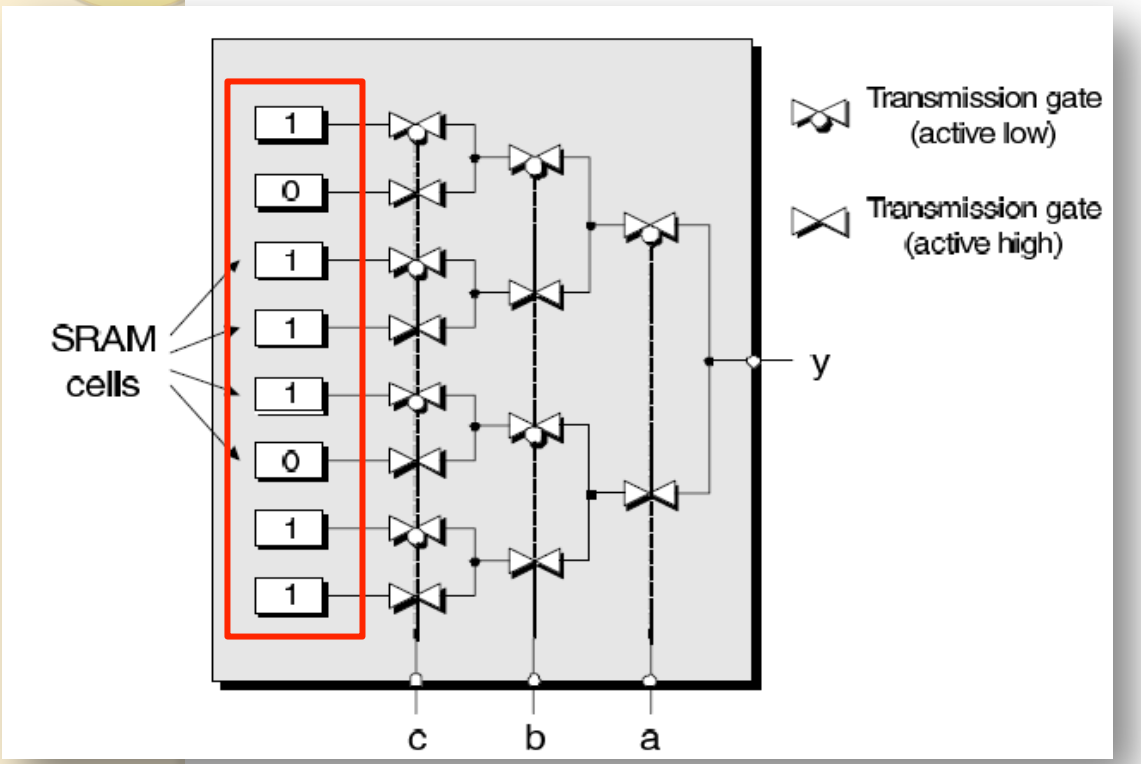
- Architecture Xilinx
  - Élément de base : Logic Cell
  - Slice = 2 Logic Cell
  - Configurable Logic Bloc
    - CLB = 2 ou 4 Slices



Logic Cell

Architecture des FPGA

- LUT ⇒ Look-Up Table
  - Une mémoire de taille 2^n peut implanter n'importe quelle fonction d'au plus *n* variables



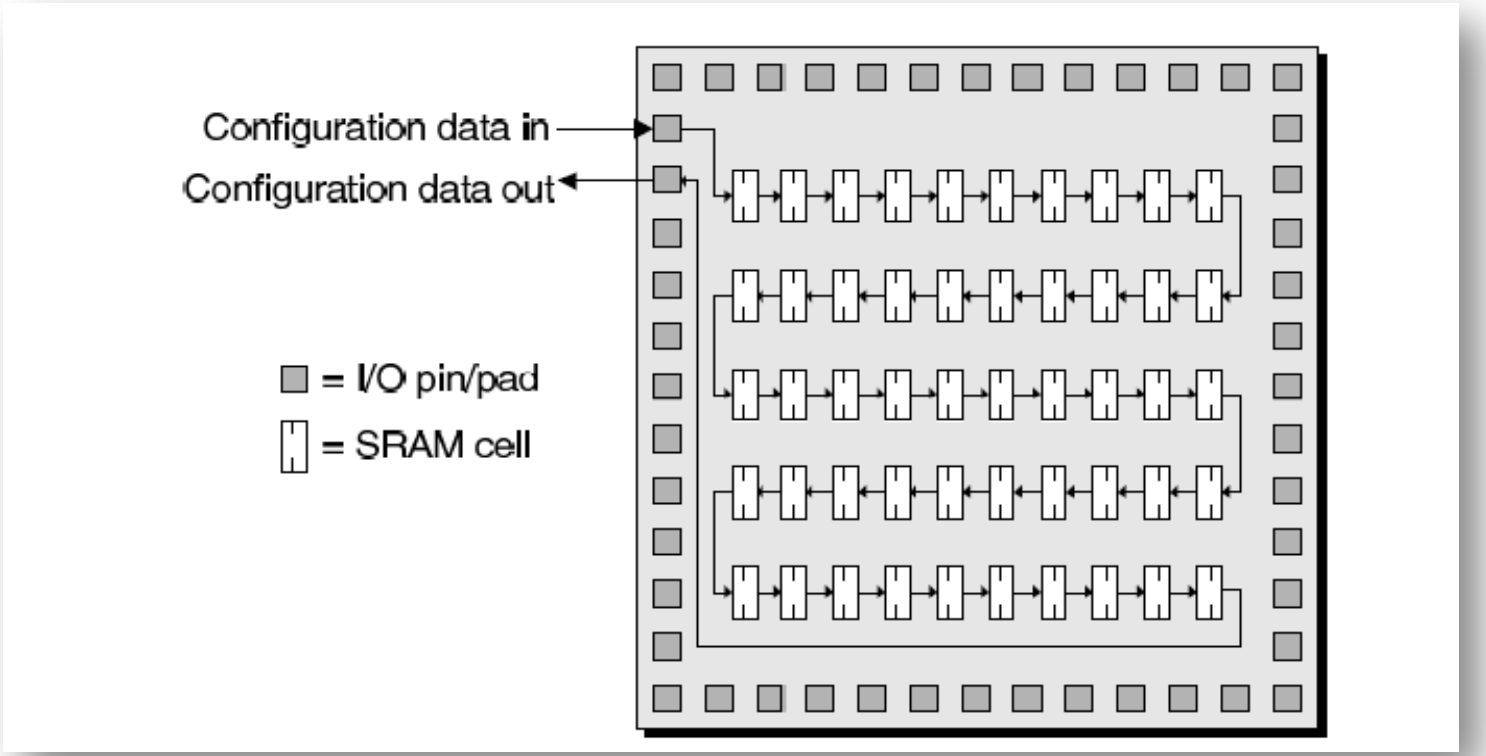
a	b	c	y
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**

…

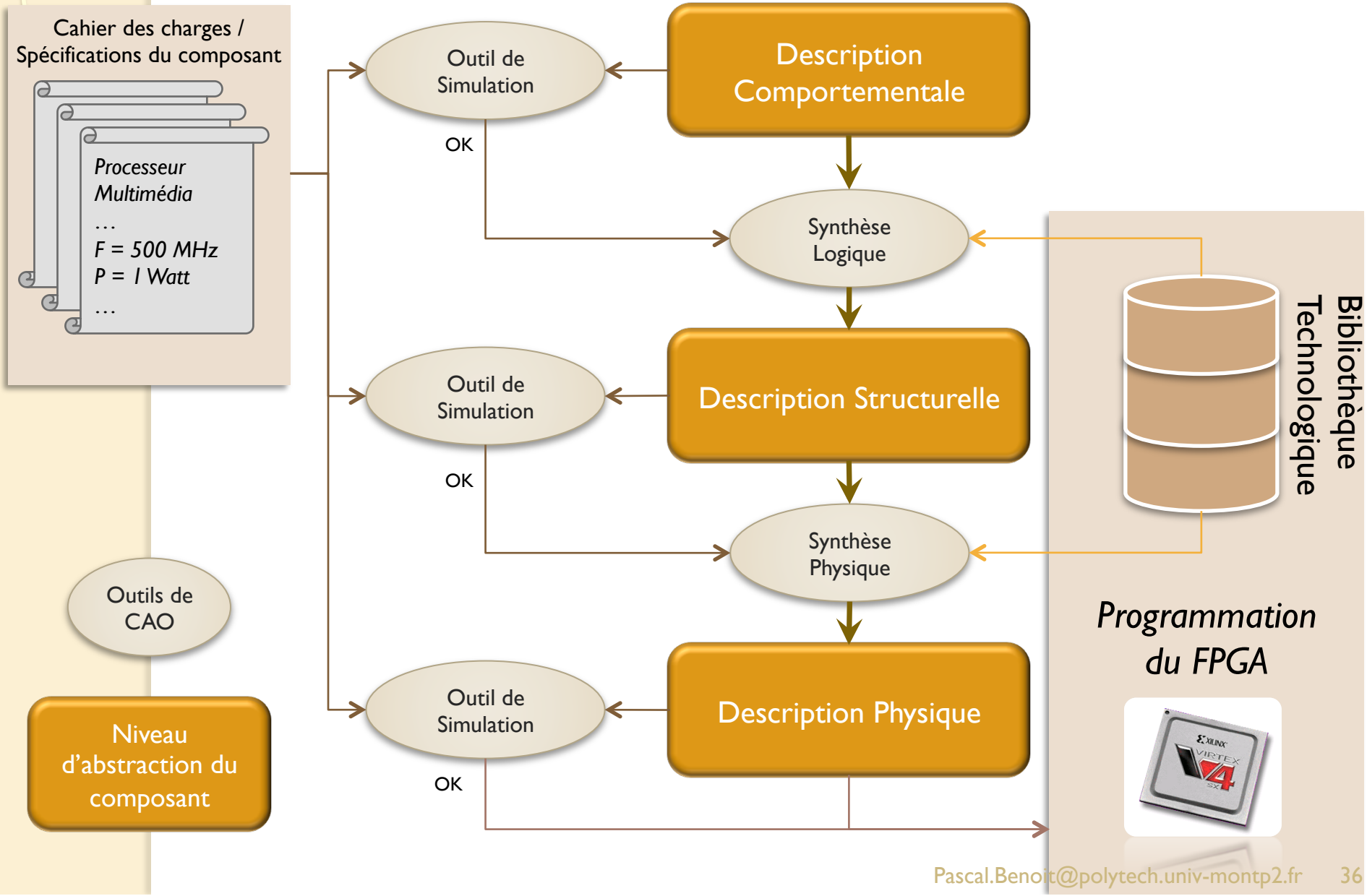
$y = \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}bc + \ldots$

Architecture des FPGA

- **Technologie SRAM**
  - La programmation des FPGA SRAM est réalisée par l'intermédiaire d'une interface série et d'un Bitstream (flux de bit)



Flot de Conception FPGA

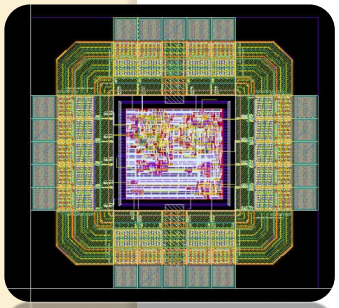


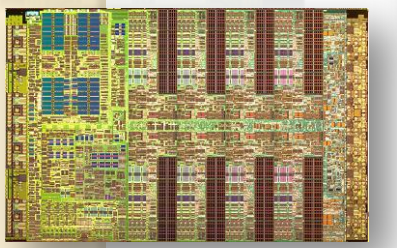
HDL, des langages pour…

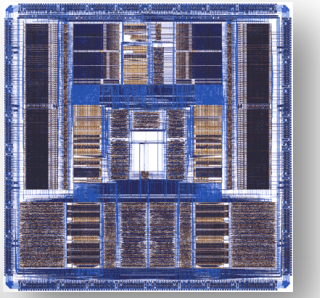
HDL

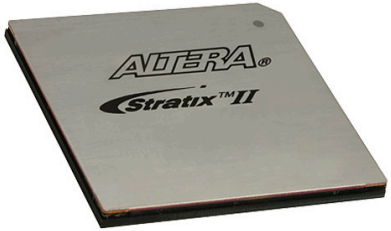
Concevoir des circuits intégrés

« Programmer » des circuits FPGA

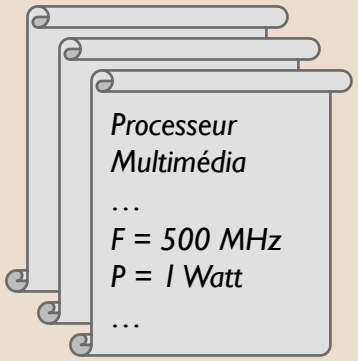








Flot de conception



Outils de CAO

Niveau d'abstraction du circuit

Description Comportementale

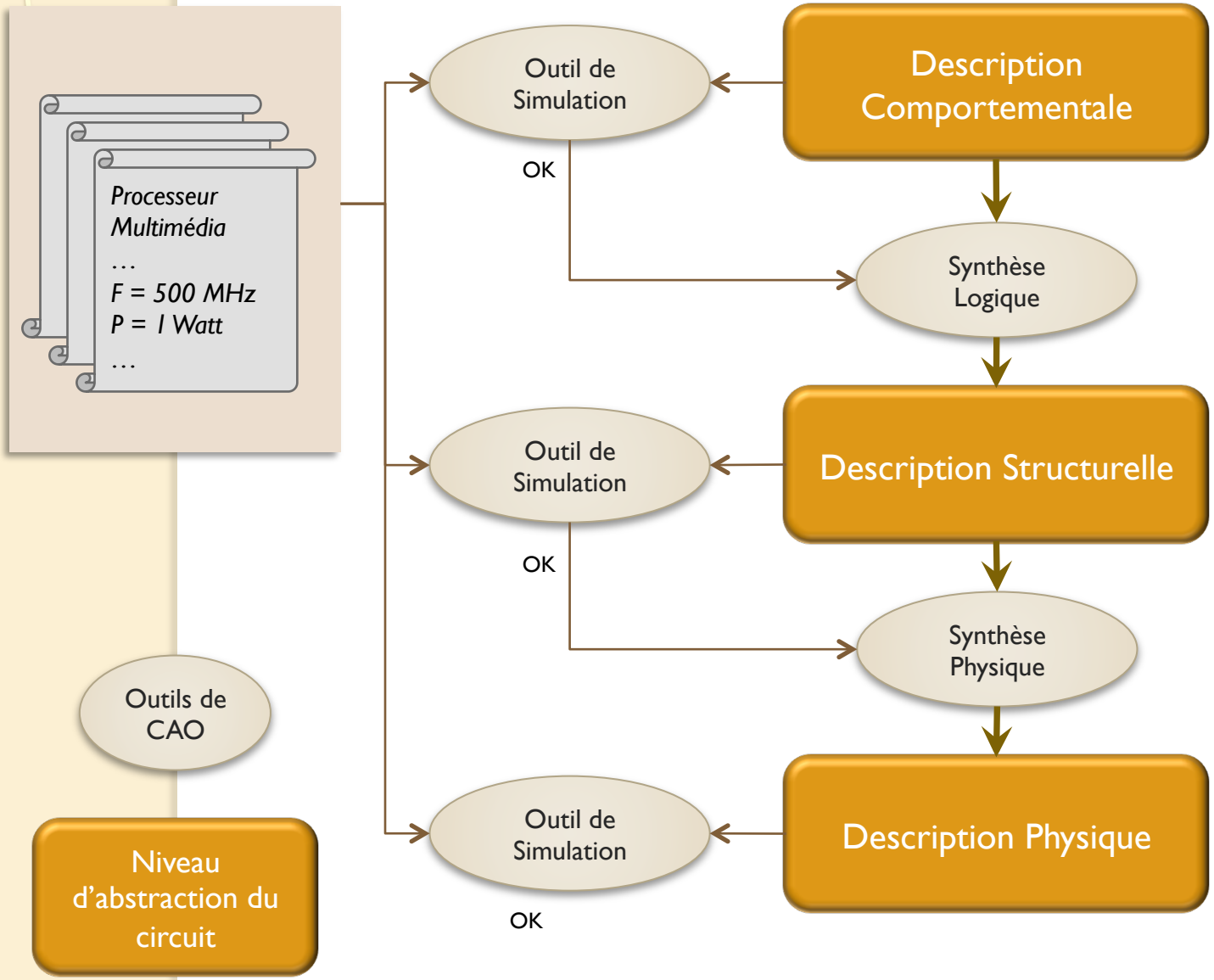
Synthèse Logique

Description Structurelle

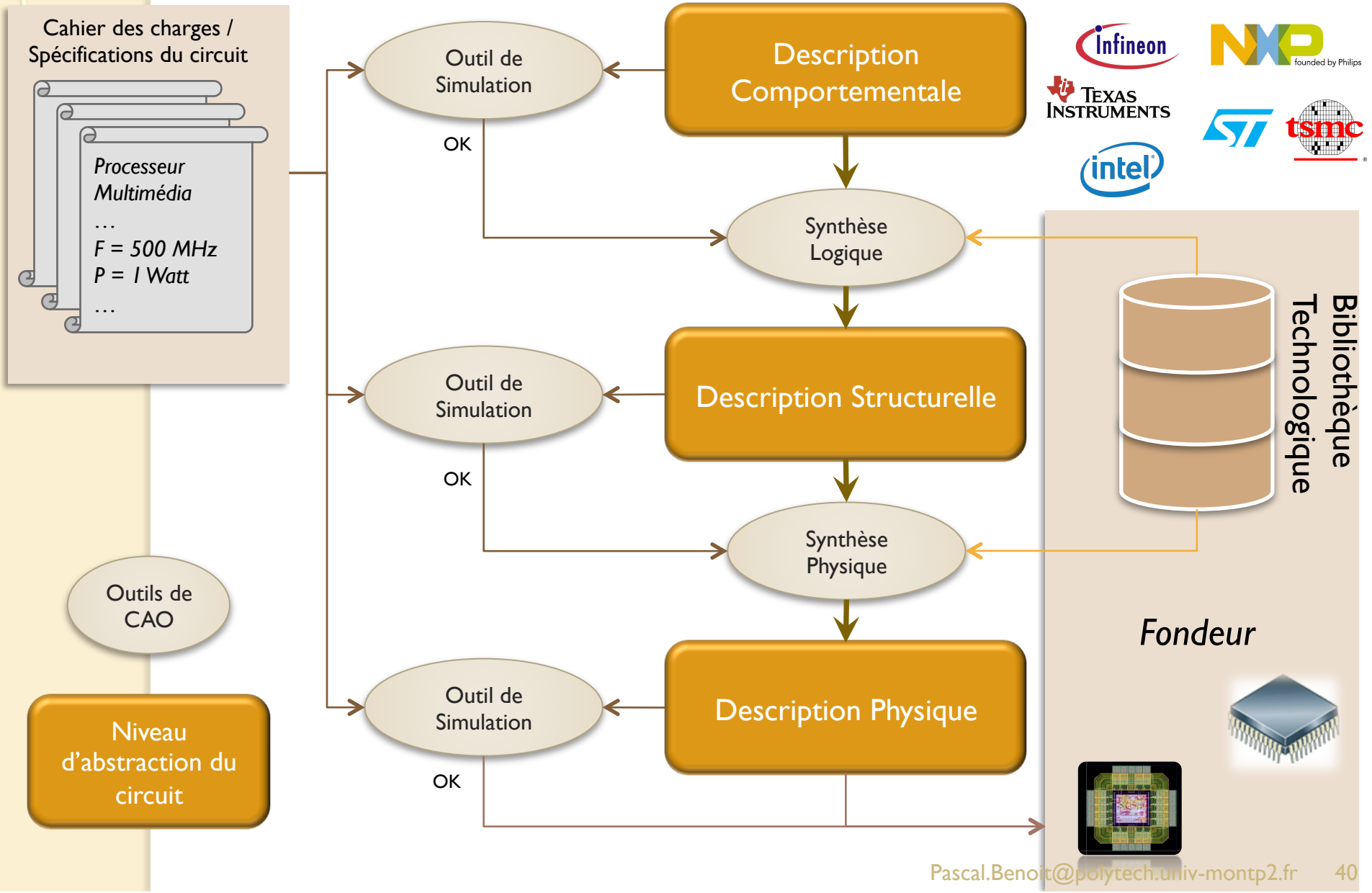
Synthèse Physique

Description Physique

Flot de conception



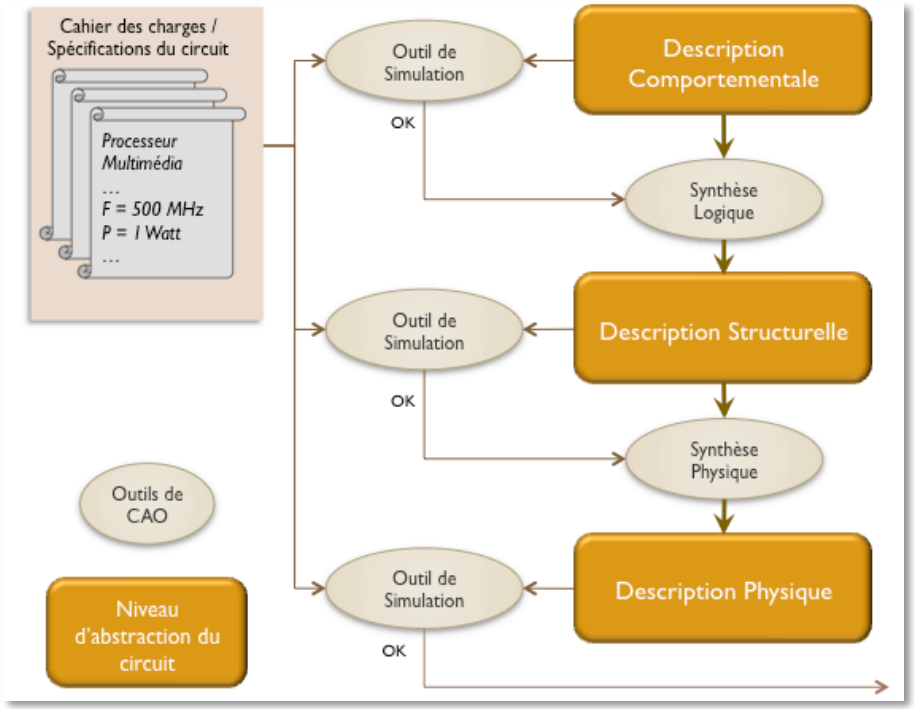
Flot de Conception Silicium



Flot de Conception Silicium

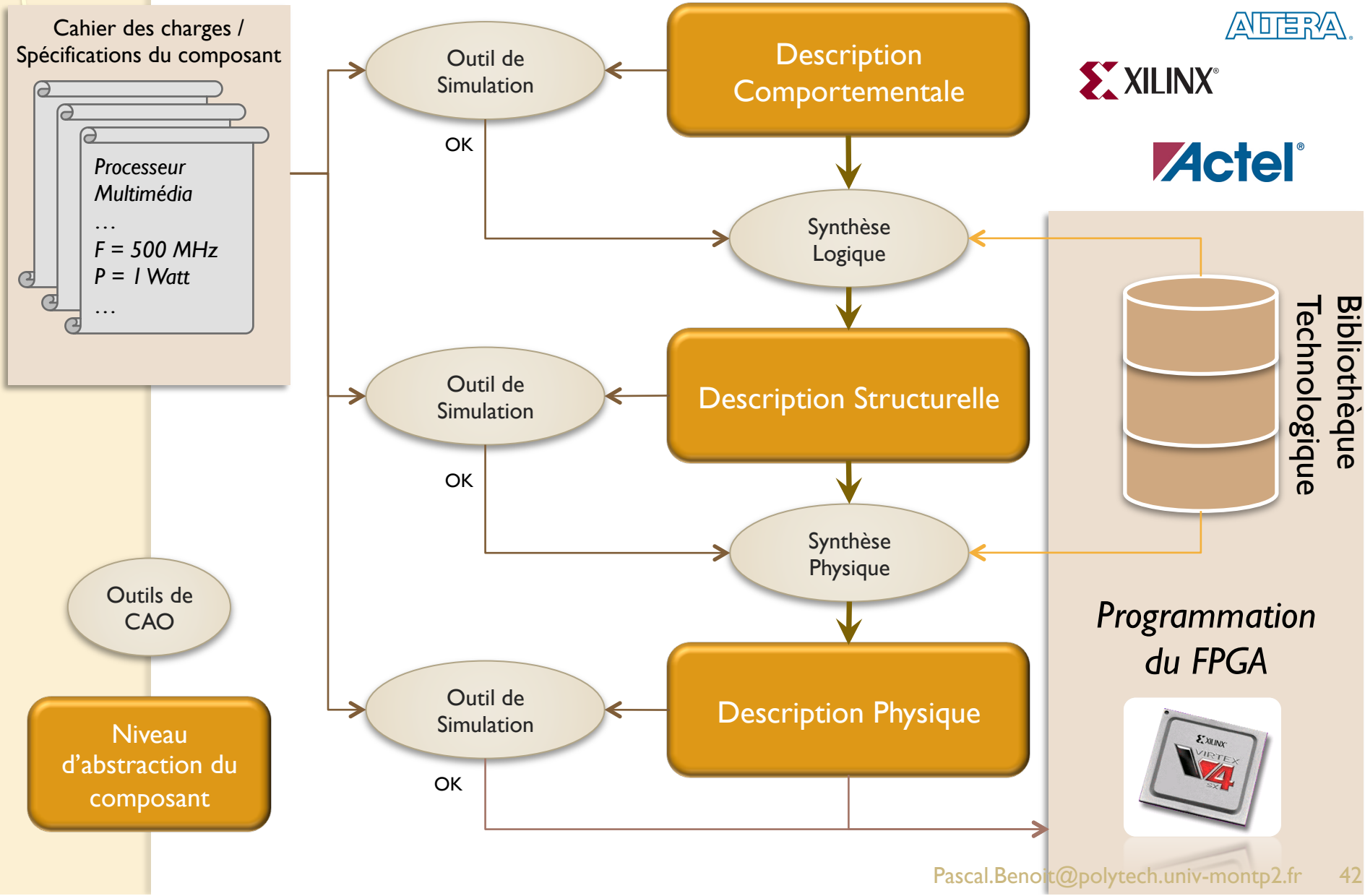
Principaux acteurs de la CAO

cadence™ SYNOPSYS® Mentor Graphics®



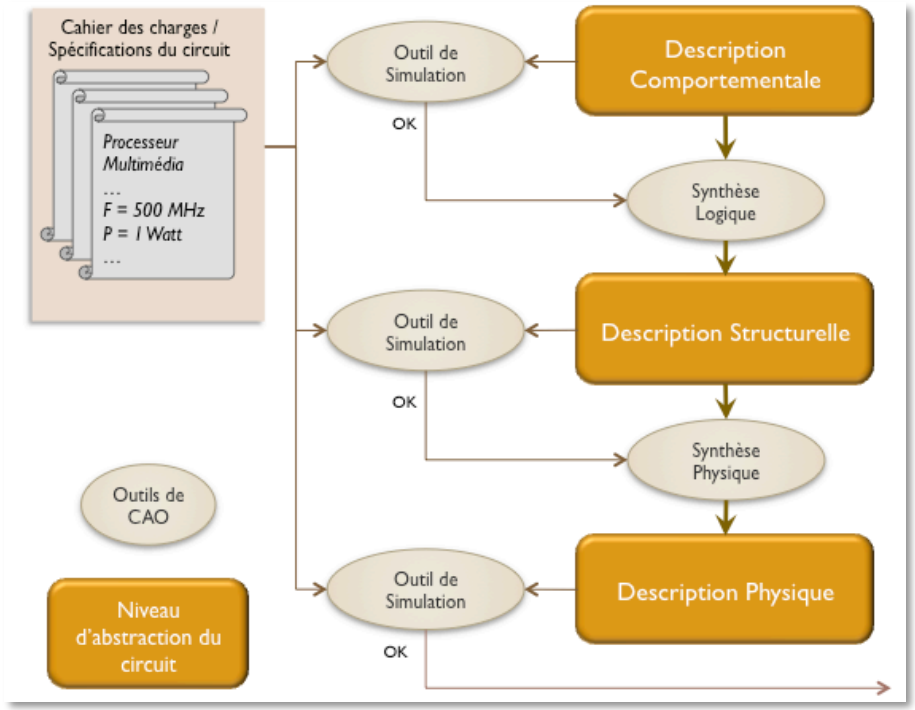
En réalité, c'est une trentaine d'outils nécessaires…

Flot de Conception FPGA

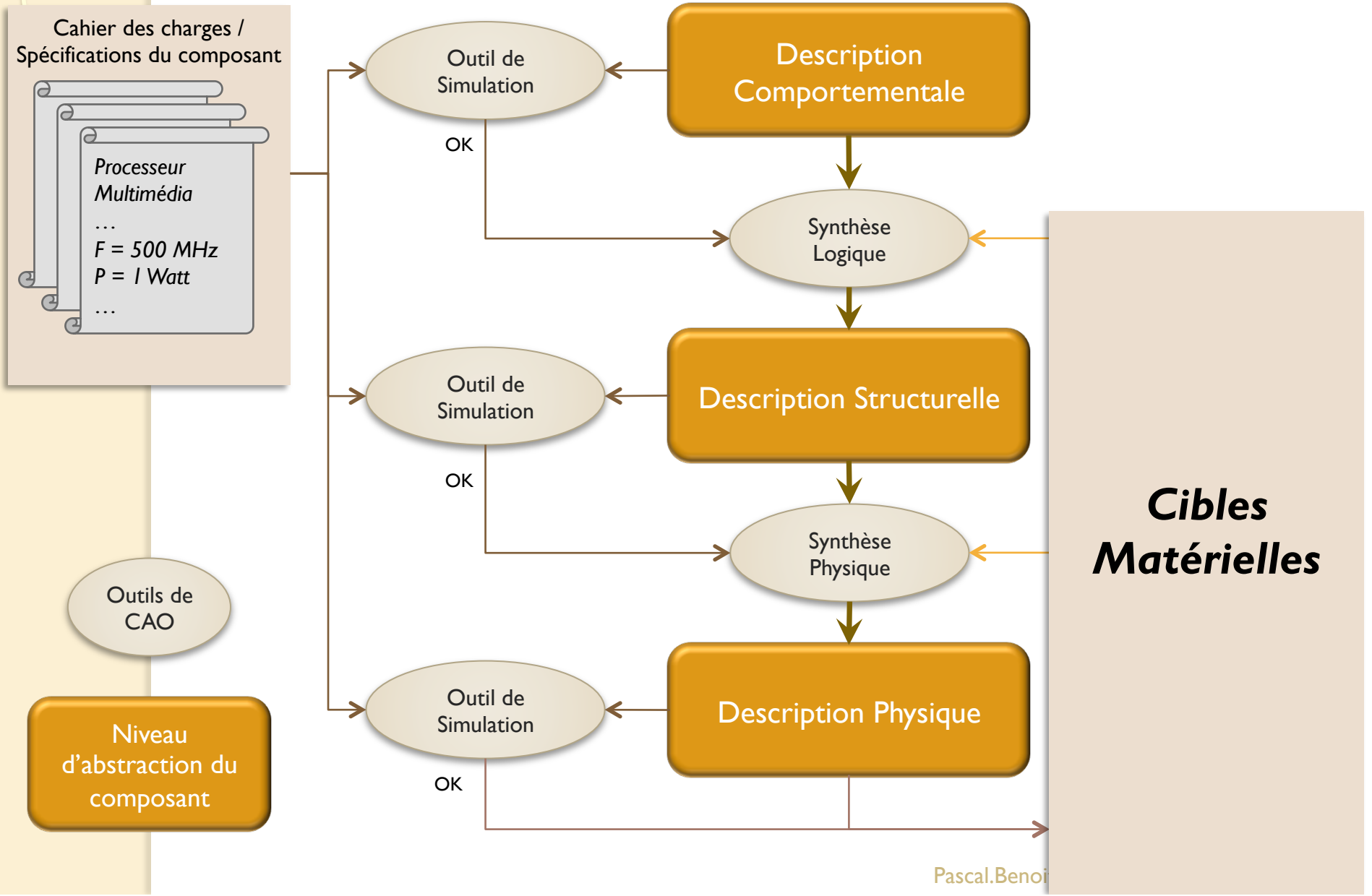


Flot de Conception FPGA

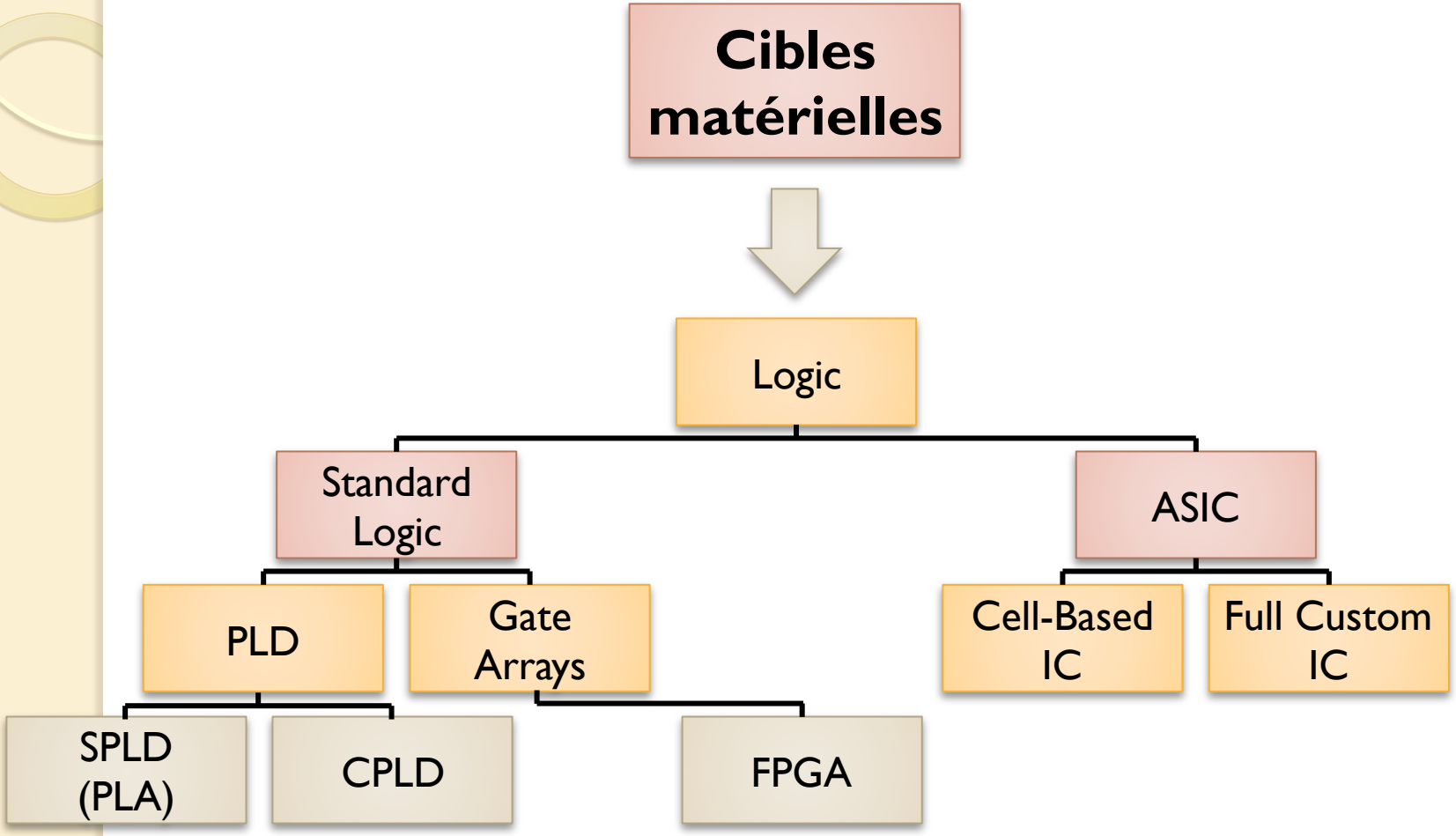
Les fabricants de FPGA fournissent leurs propres outils CAO…



Flot de Conception générique

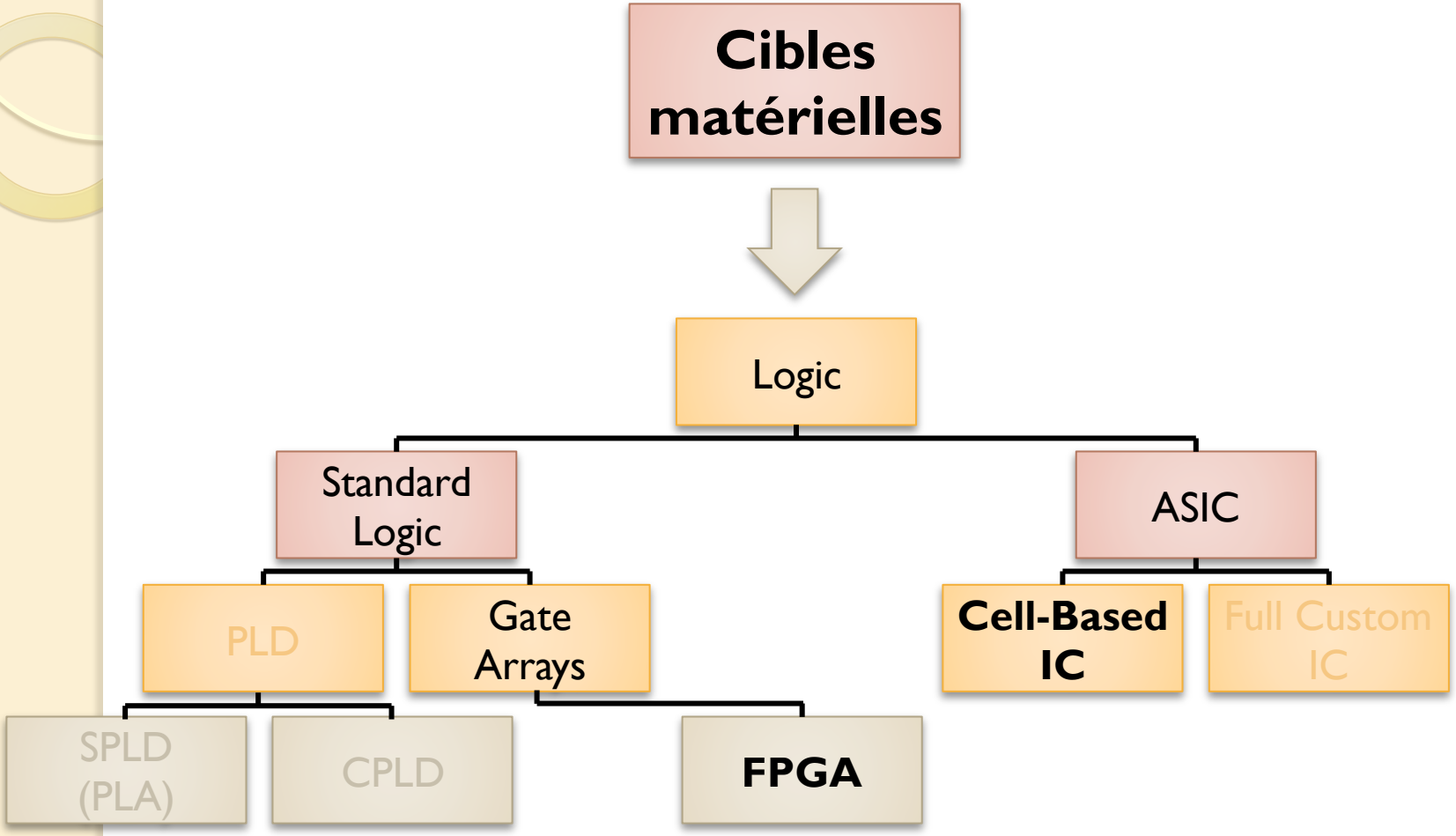


Cibles matérielles



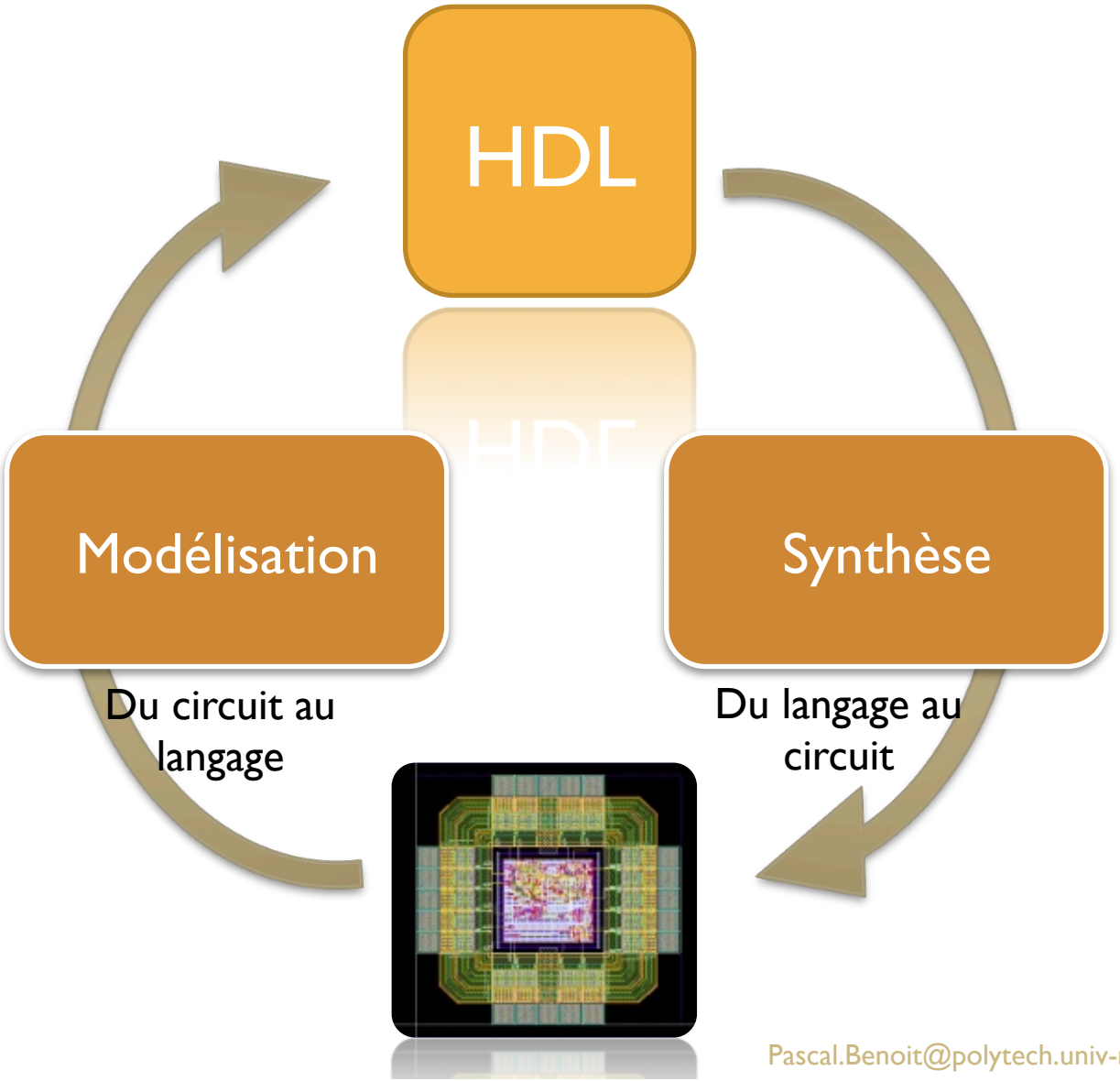
SPLD Simple Programmable Logic Device
CPLD Complex PLD (Programmable Logic Device)
FPGA Field Programmable Gate Array

HDL: Cibles matérielles



SPLD Simple Programmable Logic Device
CPLD Complex PLD (Programmable Logic Device)
FPGA Field Programmable Gate Array

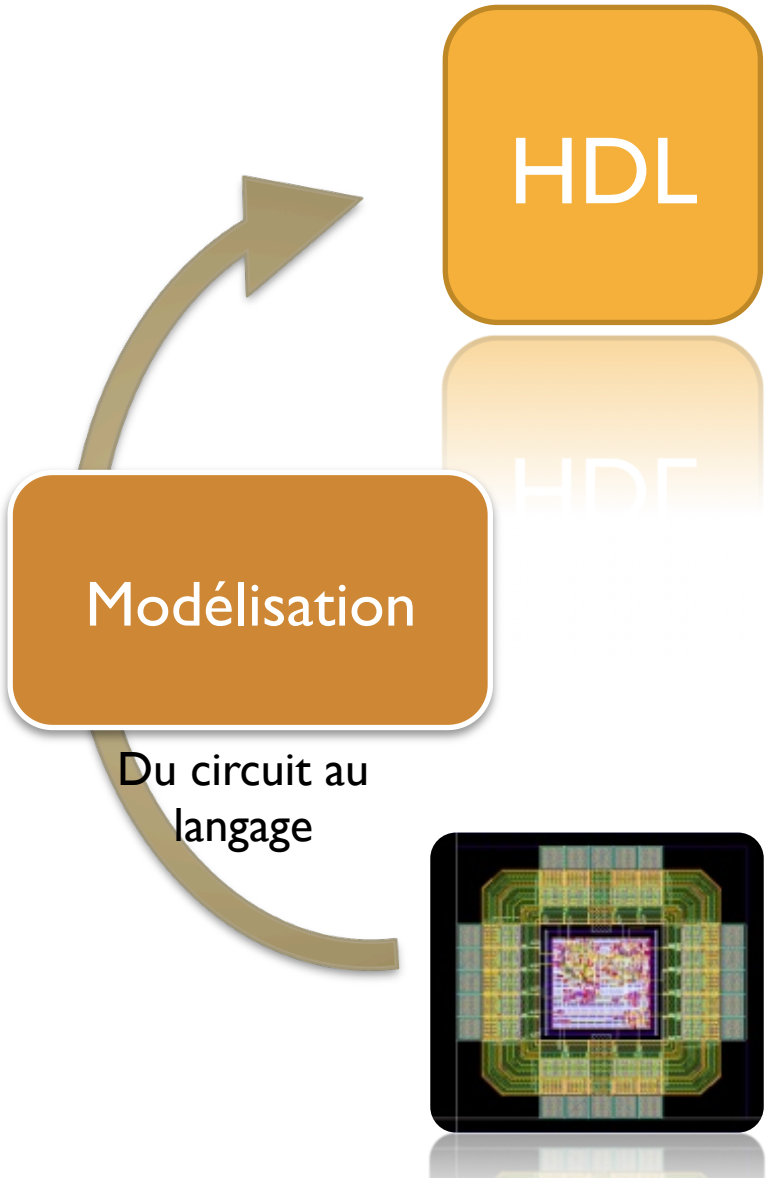
Les « Hardware Description Languages »



Histoire des HDL

- 1977
  - ISP (Instruction Set Processor)
    - Carnegie Mellon University
  - KARL
    - University of Kaiserslautern
- 1980
  - ABL
    - Descendant de KARL, interface graphique
- 1983
  - ABEL
    - Composants logiques programmables
    - Machines à Etats Finis
- 1985
  - Verilog
    - Gateway Design Automation (racheté par Cadence)
- 1987
  - VHDL
    - Département de la défense USA
- Jusque là, HDL utilisés « bottom-up » Modélisation
- Progressivement, « top-down » Synthèse

Les « Hardware Description Languages »

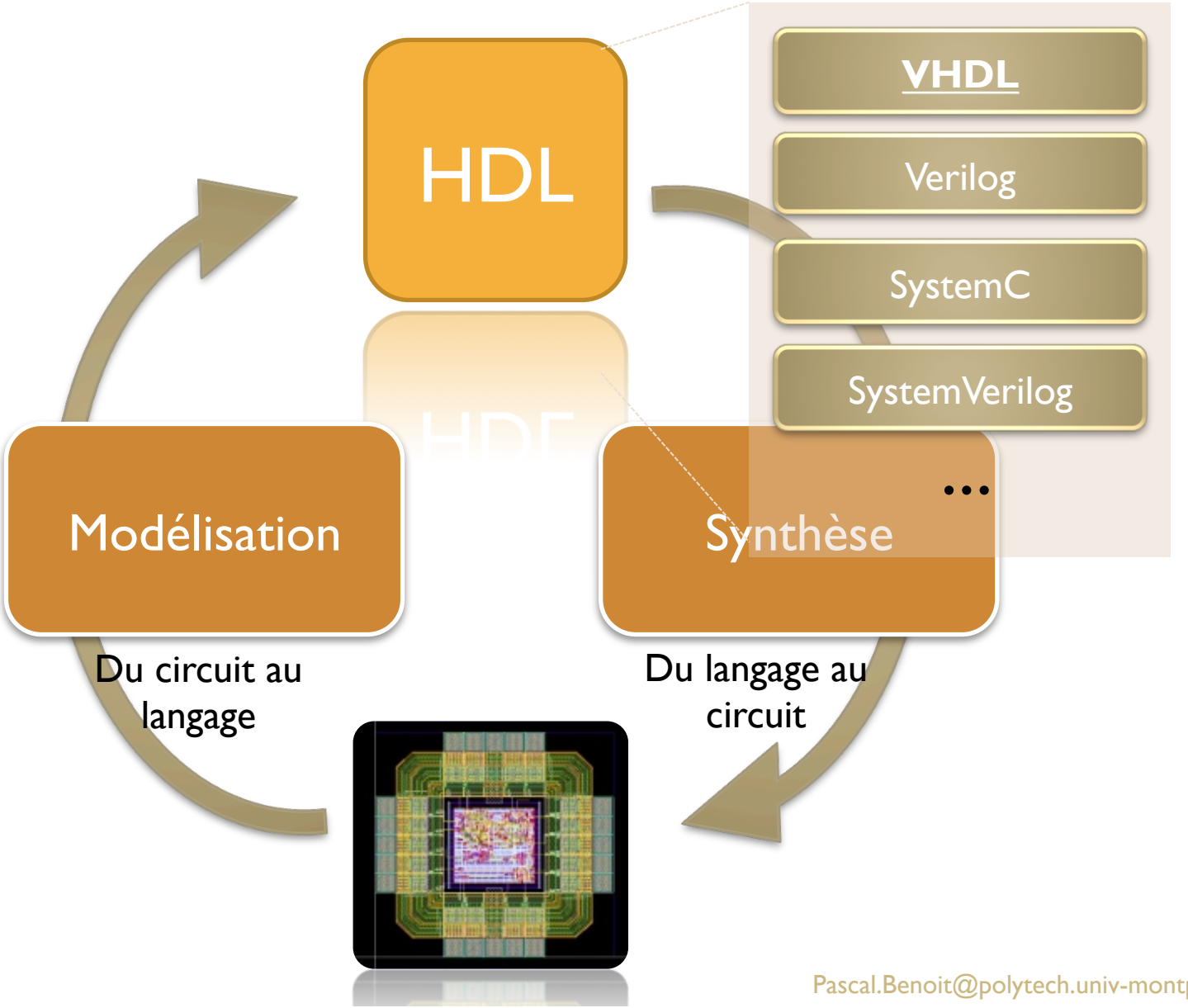


Histoire des HDL

- 1990 - 2000
  - Verilog & VHDL standards IEEE
  - Quelques évolutions
- 2000
  - SystemC
    - Niveau « système »
    - Classes & macros C++
- 2002
  - SystemVerilog
    - Extension « système » de Verilog

- Langages « systèmes » « bottom-up » Modélisation
- VHDL/Verilog « top-down » Synthèse

Les « Hardware Description Languages »



VHDL ou Verilog?

- Verilog utilisé le plus souvent après la synthèse logique
- le passage Verilog / VHDL se fait assez facilement sur les outils CAO
- Il existe des traducteurs « automatiques »…

Conception & Simulation VHDL

CONCEPTION

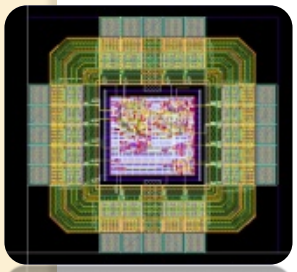
VHSIC – HDL: Very High Speed Integrated Circuits HDL

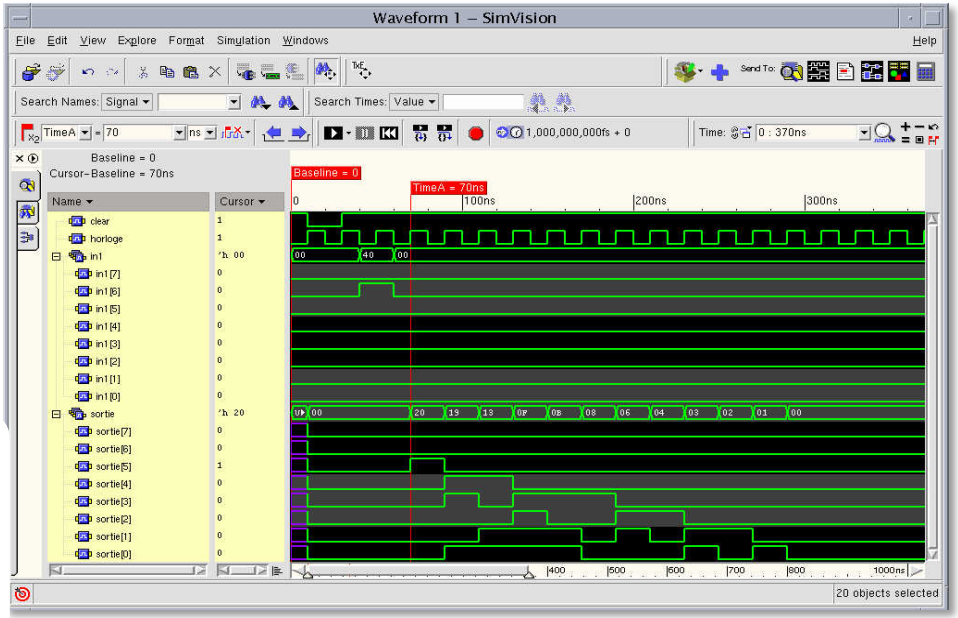
VHDL

Description du circuit ⇐ Stimuli

Du langage au circuit

Synthèses





SIMULATION

Principes généraux

- **Programmation « classique » (exemple: le langage C)**
  - Consiste à décrire une succession d'instructions qui se déroulent dans un ordre préétabli
- **Matériel ou Circuits intégrés**
  - Par essence, parallèle: tous les composants sont actifs en même temps
- **(V)HDL**
  - Sert à décrire et simuler du matériel
  - Comporte des aspects parallèles (structure du circuit) et des aspects séquentiels (processus qui décrivent les comportements des blocs)
- **Testbench**
  - C'est un programme de « test », qui doit servir à appliquer des stimuli en entrée du circuit, permettre d'observer les signaux de sorties, et ainsi vérifier la fonctionnalité du circuit simulé
  - Le testbench est écrit en VHDL
- **Simulation**
  - Émulation du matériel → parallélisme

Interro!

- Qu'est-ce qu'un circuit intégré numérique?
- Comment est conçu un CIN?
- Comment est fabriqué un CIN?
- Qu'est-ce qu'un FPGA?
- Comment est « programmé » un FPGA?
- Quel est l'intérêt des langages HDL?
- Quelle est la différence majeure entre un langage « type C » et un langage HDL?
- Qu'est-ce qu'un testbench?
- A quoi sert la simulation?

(V)HDL: langage(s) et métiers…

- Ingénieur Systèmes Embarqués
  - Spécifications, modélisation système
- Ingénieur Microélectronique Numérique « Front-end »
  - Conception de blocs, simulation, synthèse logique
- Ingénieur Microélectronique Numérique « Back-end »
  - Synthèse physique, placement / routage
- Ingénieur Test & Vérification
  - Qualité, test, simulation, etc.
- Ingénieur Electronicien FPGA
  - « Programmation » de FPGA
- Ingénieur R&D Microélectronique
  - Doctorat microélectronique, électronique, traitement du signal, etc.
  - Départements Recherche des grands groupes
- …

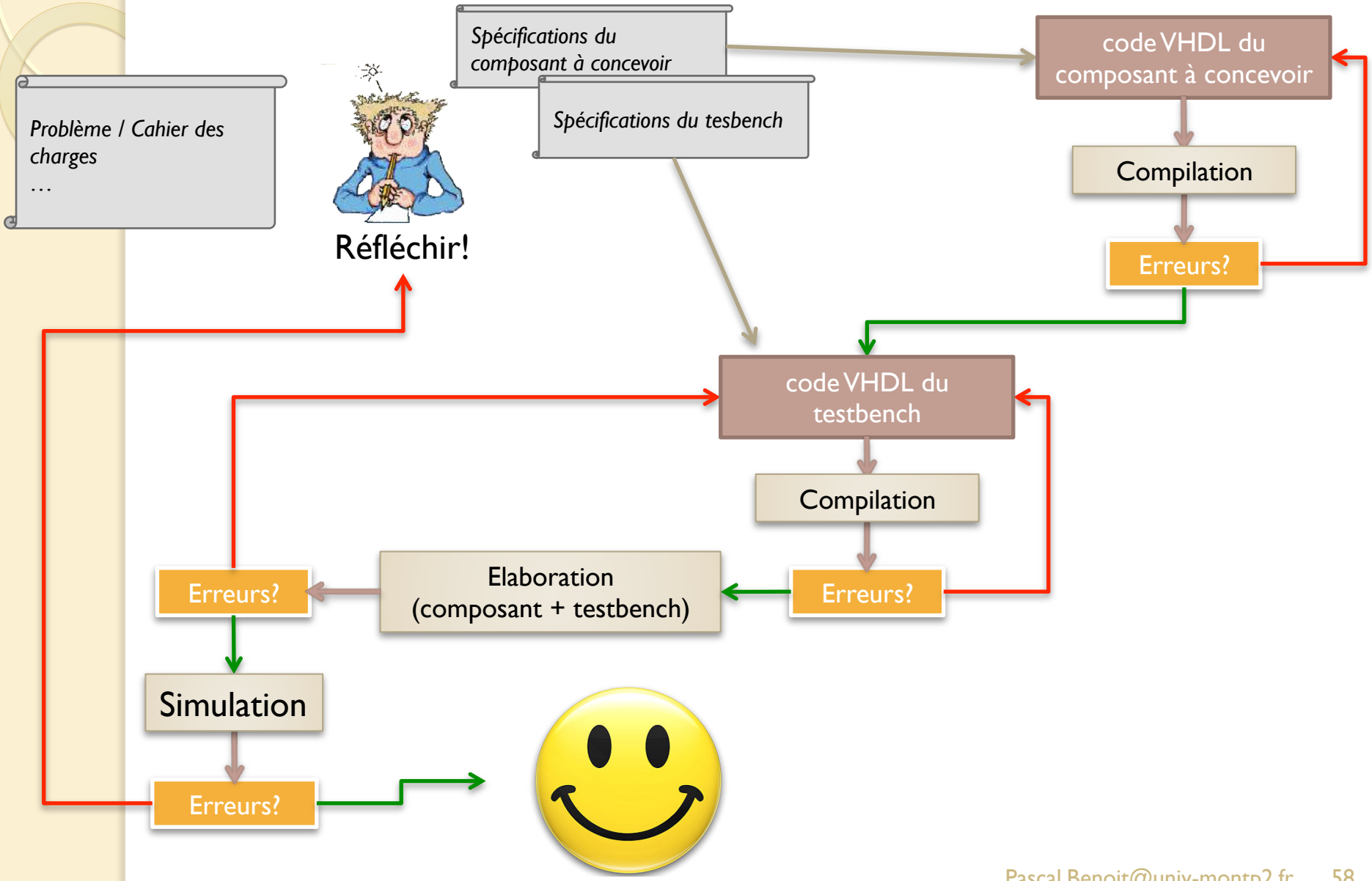
Regardez les offres d'emploi sur le Net!

Tous les métiers autour du Circuit Intégré Numérique

Terminologie

- Analyse (Compilation)
  - C'est ce qu'on appelle la compilation en langage C
  - Tous les fichiers VHDL d'un projet sont compilés et le résultat est stocké dans la bibliothèque de travail (appelée «library work »)
- Elaboration
  - Cela correspond à l'édition des liens en C
  - Elle s'effectue sur l'entité de plus haut niveau hiérarchique
- Simulation
  - s'effectue sur le résultat de l'élaboration

Algorithme de l'étudiant modèle



Mise en pratique!

Cahier des charges
Concevoir un circuit dont la sortie est le résultat d'un ET logique des 2 entrées sur 1 bit

Réfléchir!

Composant
2 entrées sur 1 bit
1 sortie sur 1 bit
Fonction: ET logique

Testbench
Appliquer sur les 2 entrées toutes les combinaisons possibles « 00 », « 01 », « 10 », « 11 »

code VHDL du composant à concevoir

Code VHDL du circuit à concevoir

```
--Bibliothèques
library ieee;
use ieee.std_logic_1164.all

--Entité
entity circuit is
        port (  a,b: in std_logic;
                s: out std_logic
        );
end circuit ;

--Architecture
architecture archi of circuit is

begin
    s <= a and b;
end;
```

Code VHDL du circuit à concevoir

```
--Bibliothèques
library ieee;
use ieee.std_logic_1164.all

--Entité
entity circuit is
        port (  a,b: in std_logic;
                s: out std_logic
        );
end circuit ;

--Architecture
architecture archi of circuit is

begin
     s <= a and b;
end;
```

COMMENTAIRES

Les commentaires doivent être inclus dans le code, pour augmenter la lisibilité et la documentation.

Ils commencent par 2 tirets (--) et se terminent en fin de ligne

Code VHDL du circuit à concevoir

```
--Bibliothèques
library ieee;
use ieee.std_logic_1164.all

--Entité
entity circuit is
        port (  a,b: in std_logic;
                s: out std_logic
        );
end circuit;

--Architecture
architecture archi of circuit is

begin
     s <= a and b;
end;
```

Entity

C'est une « boîte noire » ou encore l'interface d'un module matériel.
Elle précise:

- Le nom du circuit
- Les ports d'entrées/sorties (nom, direction, type)
- *Les paramètres éventuels pour les modèles génériques*

Code VHDL du circuit à concevoir

```
--Bibliothèques
library ieee;
use ieee.std_logic_1164.all

--Entité
entity circuit is
        port ( a,b: in std_logic;
               s: out std_logic
        );
end circuit;

--Architecture
architecture archi of circuit is

begin
    s <= a and b;
end;
```

in
a et b sont des entrées
Leur type est **std logic**

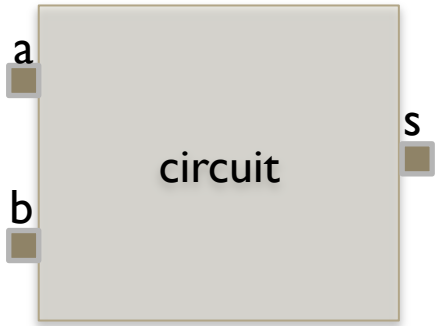
Code VHDL du circuit à concevoir

```
--Bibliothèques
library ieee;
use ieee.std_logic_1164.all

--Entité
entity circuit is
        port (  a,b: in std_logic;
                s: out std_logic
        );
end circuit;

--Architecture
architecture archi of circuit is

begin
     s <= a and b;
end;
```

out
s est une sortie de type **std logic**

Code VHDL du circuit à concevoir

```
--Bibliothèques
library ieee;
use ieee.std_logic_1164.all

--Entité
entity circuit is
        port (  a,b: in std_logic;
                s: out std_logic
        );
end circuit;

--Architecture
architecture archi of circuit is

begin
     s <= a and b;
end;
```



Code VHDL du circuit à concevoir

```
--Bibliothèques
library ieee;
use ieee.std_logic_1164.all

--Entité
entity circuit is
        port (  a,b: in std_logic;
                s: out std_logic
        );
end circuit;

--Architecture
architecture archi of circuit is

begin
     s <= a and b;
end;
```

architecture <nom_architecture> **of** <nom_entité> **is**
C'est la description interne du circuit.
Elle est toujours associée à une entité.

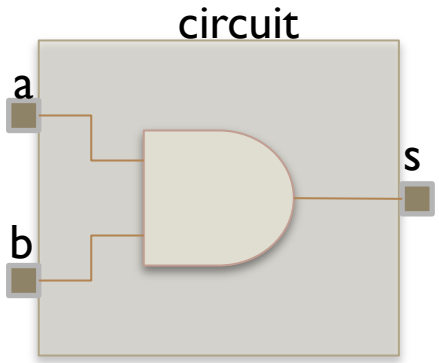
Code VHDL du circuit à concevoir

```
--Bibliothèques
library ieee;
use ieee.std_logic_1164.all

--Entité
entity circuit is
        port (  a,b: in std_logic;
                s: out std_logic
        );
end circuit;

--Architecture
architecture archi of circuit is

begin
     s <= a and b;
end;
```

« **s** prend la valeur du résultat de l'expression logique **a ET b** »

Code VHDL du circuit à concevoir

```
--Bibliothèques
library ieee;
use ieee.std_logic_1164.all

--Entité
entity circuit is
        port (  a,b: in std_logic;
                s: out std_logic
        );
end circuit;

--Architecture
architecture archi of circuit is

begin
     s <= a and b;
end;
```



Mise en pratique!

Cahier des charges
Concevoir un circuit dont la sortie est le résultat d'un ET logique des 2 entrées sur 1 bit

Réfléchir!

Composant
2 entrées sur 1
1 sortie sur 1 bit
Fonction: ET logique

Testbench
Appliquer sur les 2 entrées toutes les combinaisons possibles « 00 », « 01 », « 10 », « 11 »

code VHDL du composant à concevoir

Compilation

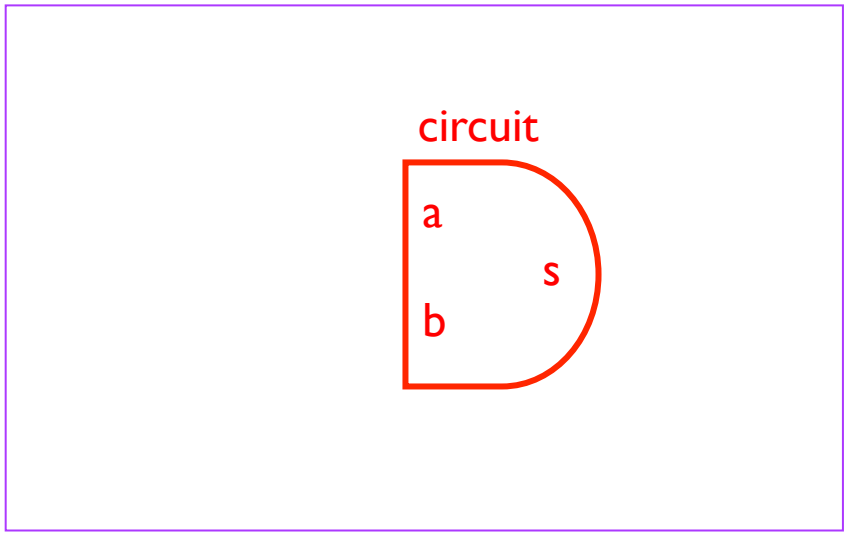
Erreurs?

code VHDL du testbench

Code VHDL du testbench

```
-- simulation du modèle and2
library ieee;
use ieee.std_logic_1164.all;
entity tb_circuit is
end tb_circuit;
architecture archi of tb_circuit is

component circuit
        port (a,b:in std_logic; s: out std_logic);
end component;
begin
  top: circuit
        port map( a=> entree1, b=> entree2,
                  s=>sortie);




end;
```

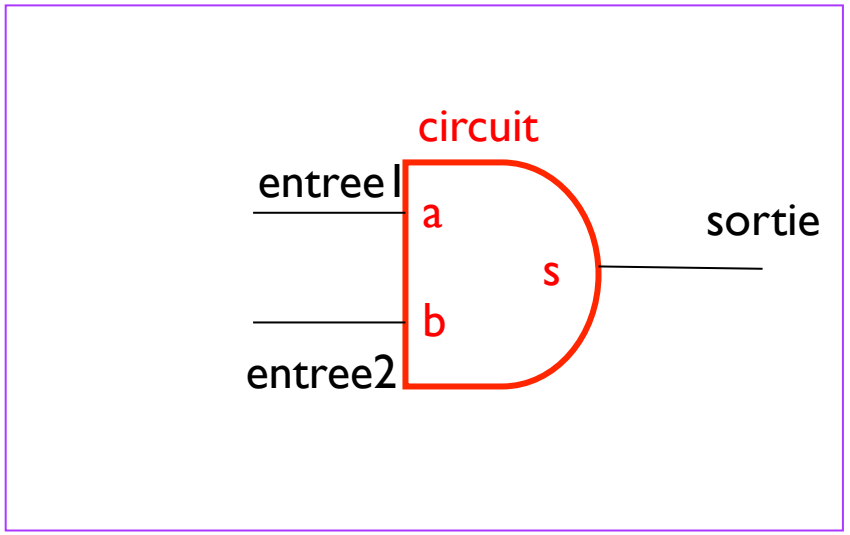
entity tb_circuit



Code VHDL du testbench

```
-- simulation du modèle and2
library ieee;
use ieee.std_logic_1164.all;
entity tb_circuit is
end tb_circuit;
architecture archi of tb_circuit is
signal entree1, entree2, sortie: std_logic;
component circuit
            port (a,b:in std_logic; s: out std_logic);
end component;
begin
  top: circuit
            port map( a=> entree1, b=> entree2,
                                          s=>sortie);




end;
```

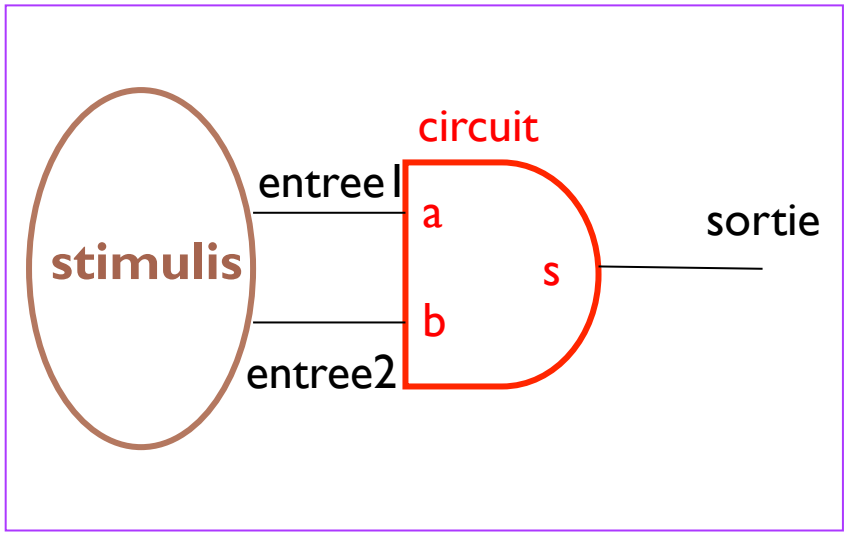
entity tb_circuit



Code VHDL du testbench

```
-- simulation du modèle and2
library ieee;
use ieee.std_logic_1164.all;
entity tb_circuit is
end tb_circuit;
architecture archi of tb_circuit is
signal entree1, entree2, sortie: std_logic;
component circuit
            port (a,b:in std_logic; s: out std_logic);
end component;
begin
  top: circuit
            port map( a=> entree1, b=> entree2,
                                          s=>sortie);
 stimulis: process
     begin
      entree1<='0'; entree2<='0';
      wait for 30 ns; entree1<='1';
      wait for 30 ns; entree2<='1';
      wait for 30 ns; entree1<='0'; entree2 <='0';
      wait for 30 ns;
    end process;
end;
```

entity tb_circuit



Mise en pratique!

Cahier des charges
Concevoir un circuit dont la sortie est le résultat d'un ET logique des 2 entrées sur 1 bit

Réfléchir!

Composant
2 entrées sur 1
1 sortie sur 1 bit
Fonction: ET logique

Testbench
Appliquer sur les 2 entrées toutes les combinaisons possibles « 00 », « 01 », « 10 », « 11 »

code VHDL du composant à concevoir

Compilation

Erreurs?

code VHDL du testbench

Compilation

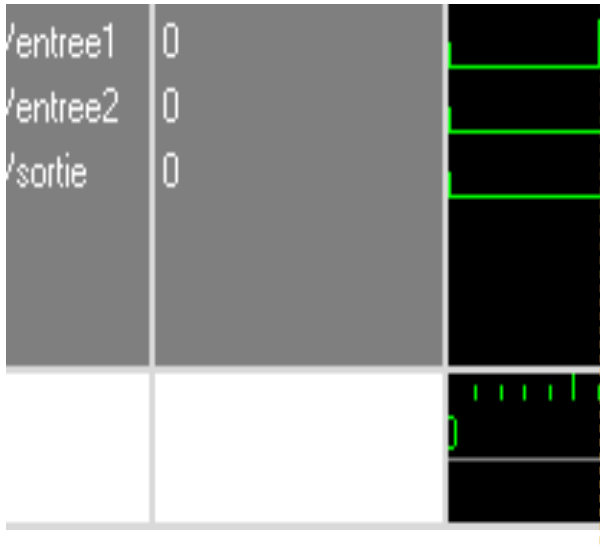
Erreurs?

Elaboration (composant + testbench)

Erreurs?

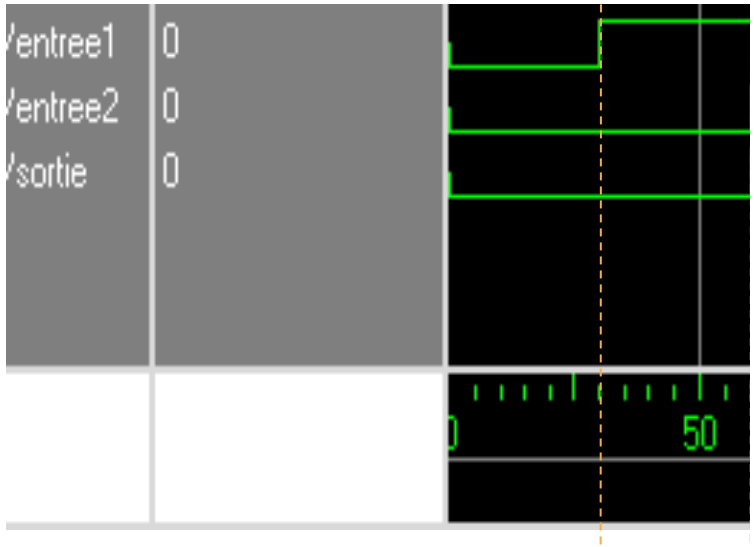
Simulation

Simulation



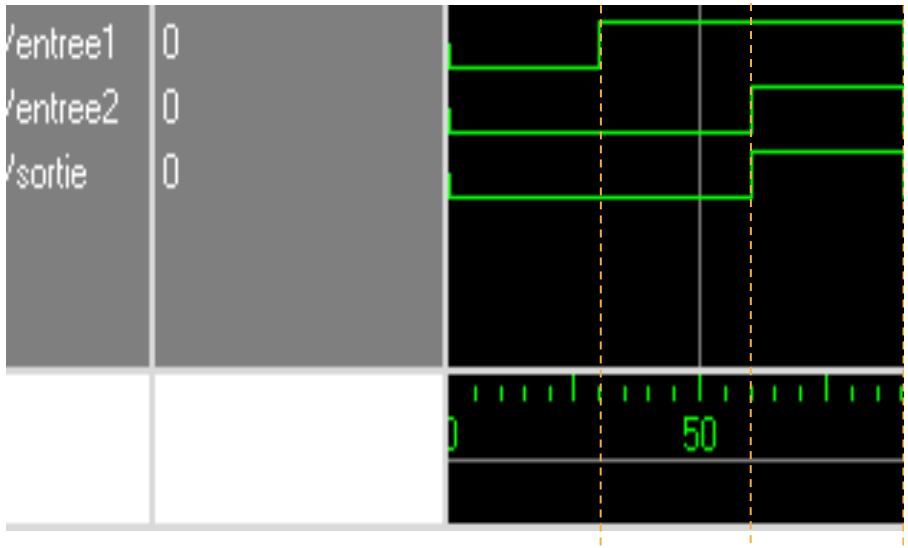

```
stimulis: process
      begin
       entree1<='0';  entree2<='0';
      wait for 30 ns;









     end process;
```

Simulation



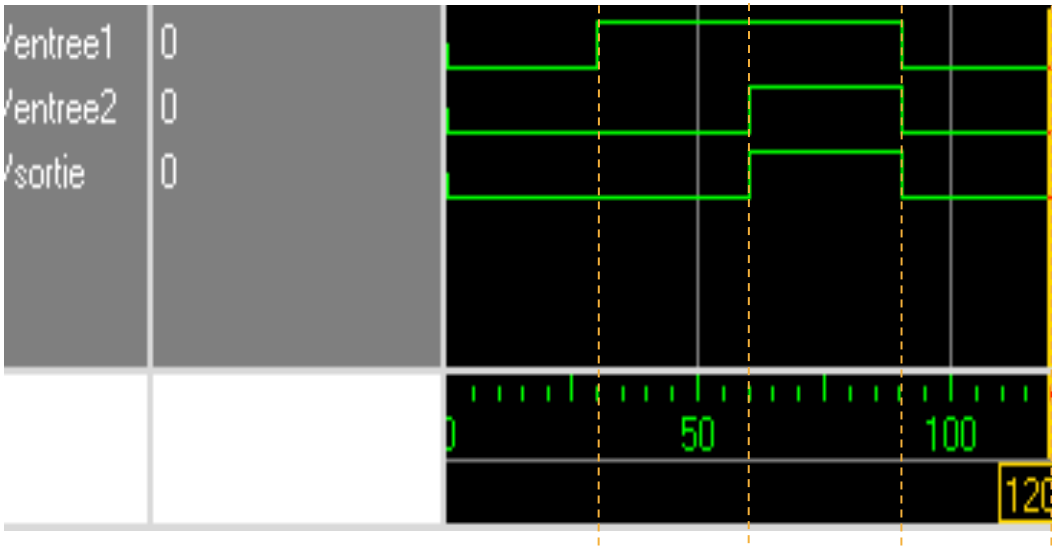

```
stimulis: process
      begin
       entree1<='0';  entree2<='0';
      wait for 30 ns;
       entree1<='1';
       wait for 30 ns;




     end process;
```

Simulation



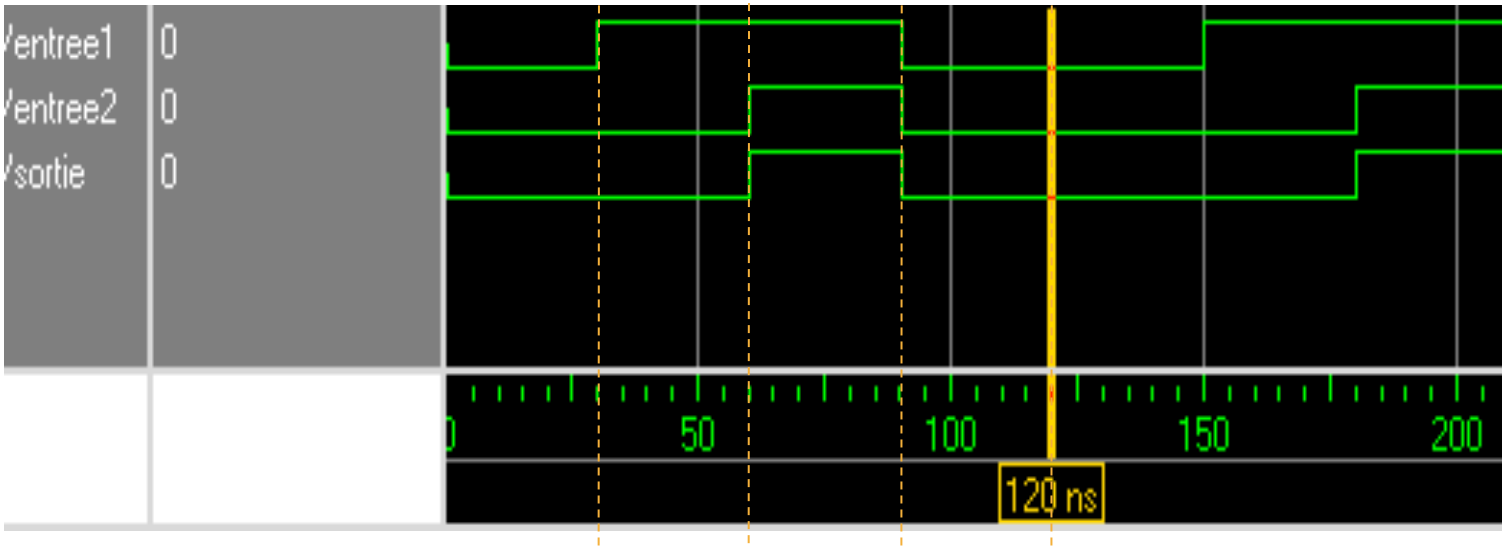

```
stimulis: process
      begin
       entree1<='0';  entree2<='0';
      wait for 30 ns;
       entree1<='1';
       wait for 30 ns;
       entree2<='1';
       wait for 30 ns;


     end process;
```

Simulation




```
stimulis: process
      begin
       entree1<='0';  entree2<='0';
      wait for 30 ns;
       entree1<='1';
       wait for 30 ns;
       entree2<='1';
       wait for 30 ns;
       entree1<='0';  entree2 <='0';
       wait for 30 ns;
     end process;
```

Simulation




```
stimulis: process
      begin
       entree1<='0';  entree2<='0';
      wait for 30 ns;
       entree1<='1';
       wait for 30 ns;
       entree2<='1';
       wait for 30 ns;
       entree1<='0';  entree2 <='0';
       wait for 30 ns;
     end process;
```

Mise en pratique!

Cahier des charges
Concevoir un circuit dont la sortie est le résultat d'un ET logique des 2 entrées sur 1 bit

Réfléchir!

Composant
2 entrées sur 1
1 sortie sur 1 bit
Fonction: ET logique

Testbench
Appliquer sur les 2 entrées toutes les combinaisons possibles « 00 », « 01 », « 10 », « 11 »

code VHDL du composant à concevoir

Compilation

Erreurs?

code VHDL du testbench

Compilation

Erreurs?

Elaboration (composant + testbench)

Erreurs?

Simulation

Erreurs?

Organisation de l'enseignement du VHDL

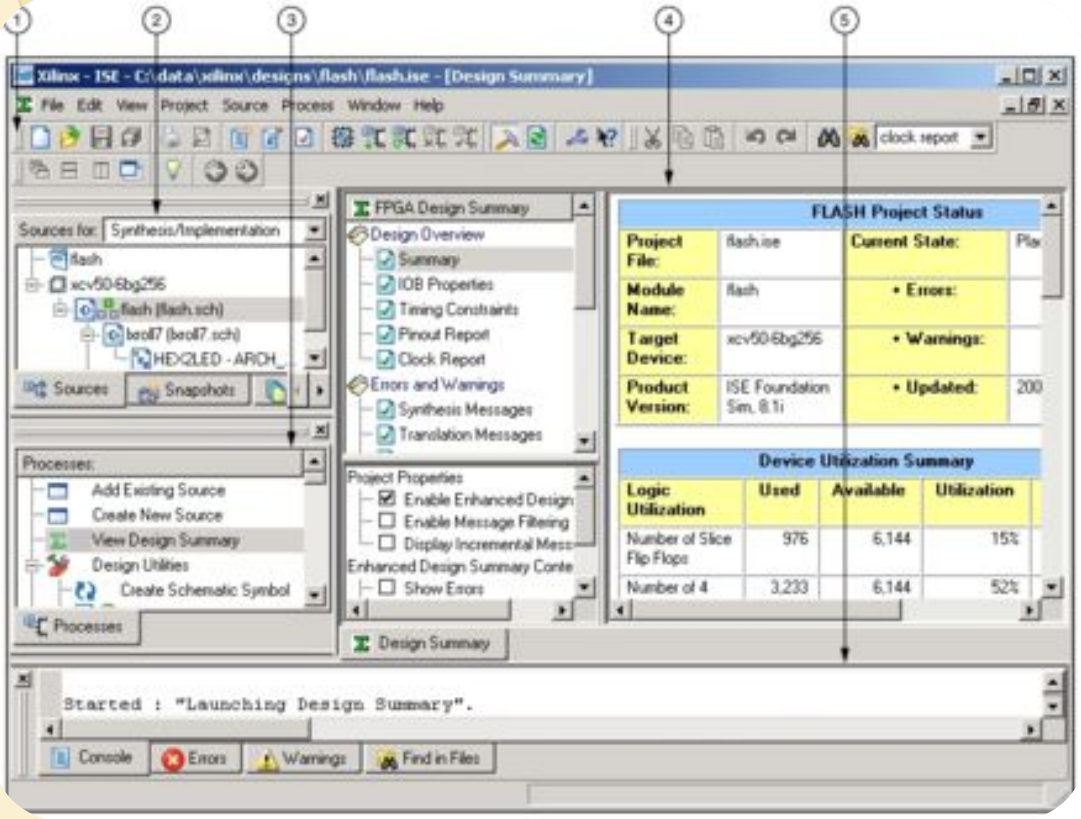
- Cours magistral (1 séance de 1h30)
  - Introduction
- Cours / TD (7 séances de 1h30)
  - Conception de composants simples
  - Simulation
- TP (5 séances de 3h)
  - Conception de composants complexes
  - Synthèse sur cible FPGA
- MINI-PROJET (7 séances de 4h)
  - P2: Circuit intégré sur cible Silicium
  - P9: Système embarqué sur cible FPGA

Outils de travail

- Fascicule de Cours/TD
- Simulateur HDL
- Site web
- Cartes FPGA
- Outils CAO
- Liens Internet
- Livres

Flot de conception FPGA (TP & Projet P9)

Xilinx ISE

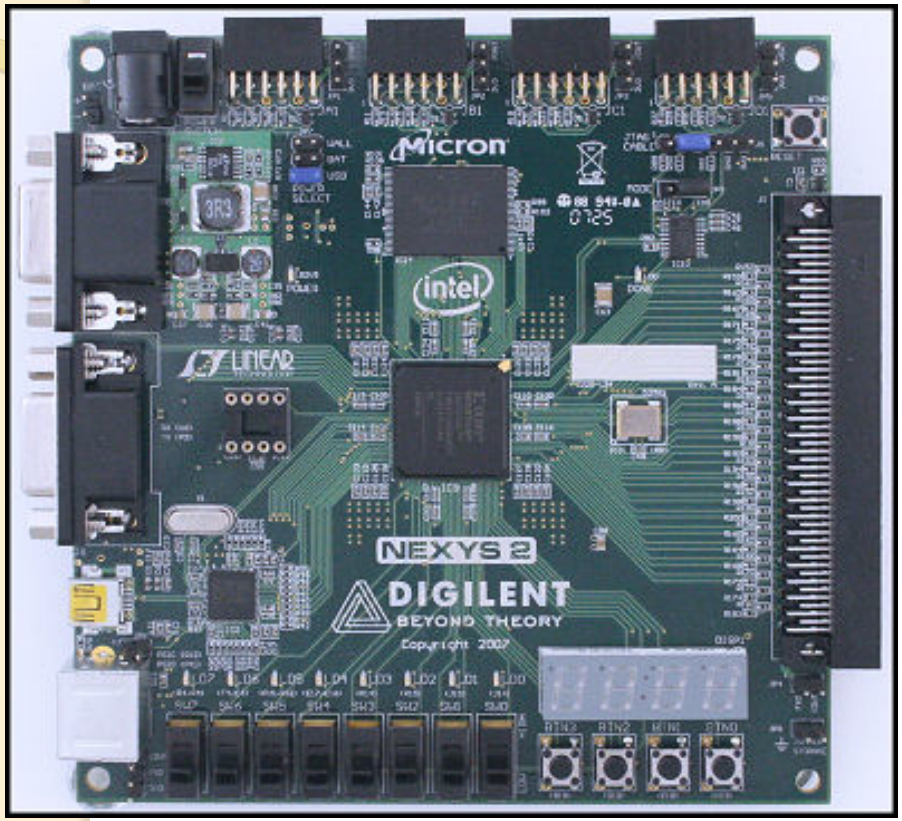


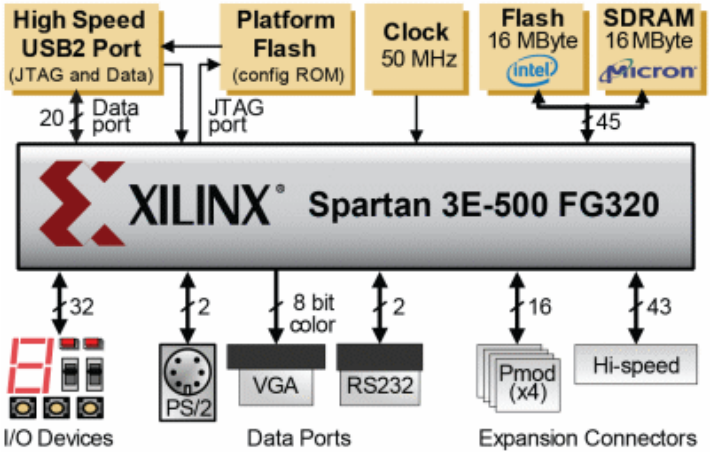
- Edition
- Simulation
- Synthèse
- Placement / Routage
- Programmation du bitstream
- …

http://www.xilinx.com/tools/webpack.htm

Carte FPGA (TD/TP & Maison ☺)

Nexys2 FPGA Board

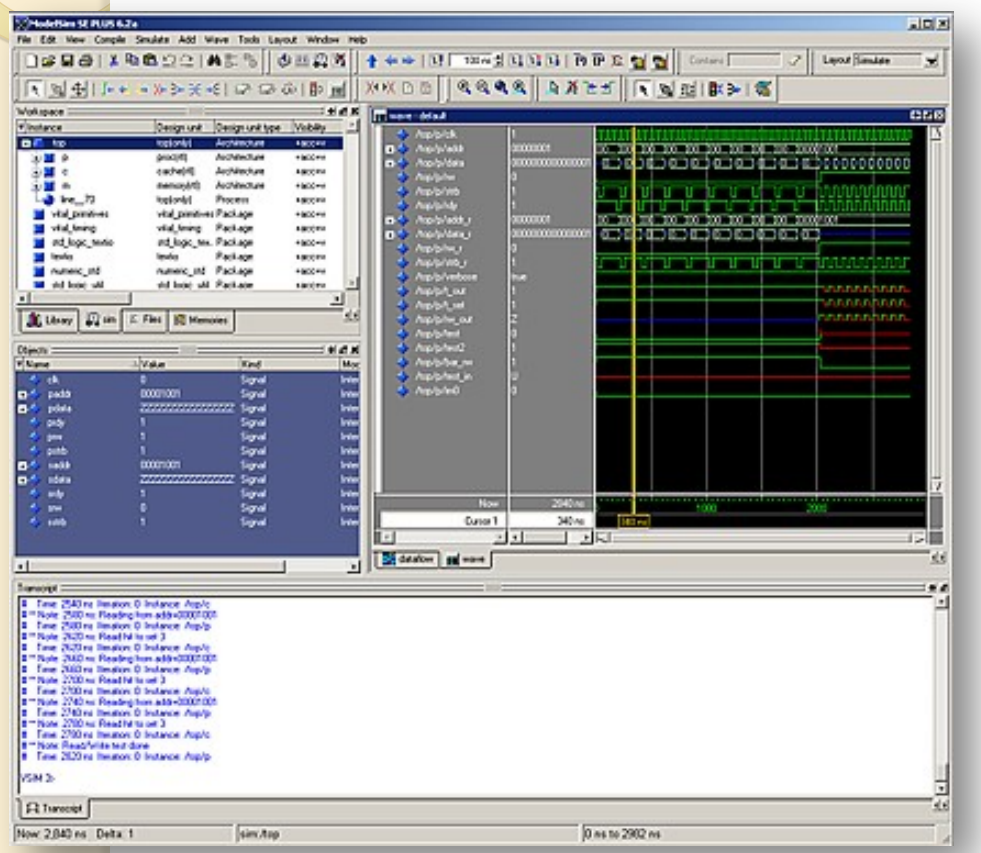




http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,451&Prod=NEXYS2

Simulateur HDL

- **ModelSim** (en TD/TP)
  - le simulateur le plus utilisé dans le milieu industriel et académique



- Multi-language simulation engine
- Verilog, VHDL, SystemVerilog
- Design Code Coverage
- SystemVerilog for Design
- Integrated debug
- Mixed HDL
- Simulation option SystemC
- Option TCL/tk

Autres simulateurs: *ActiveHDL (Aldec), NcSim (Cadence), Vcs (Synopsys), …*

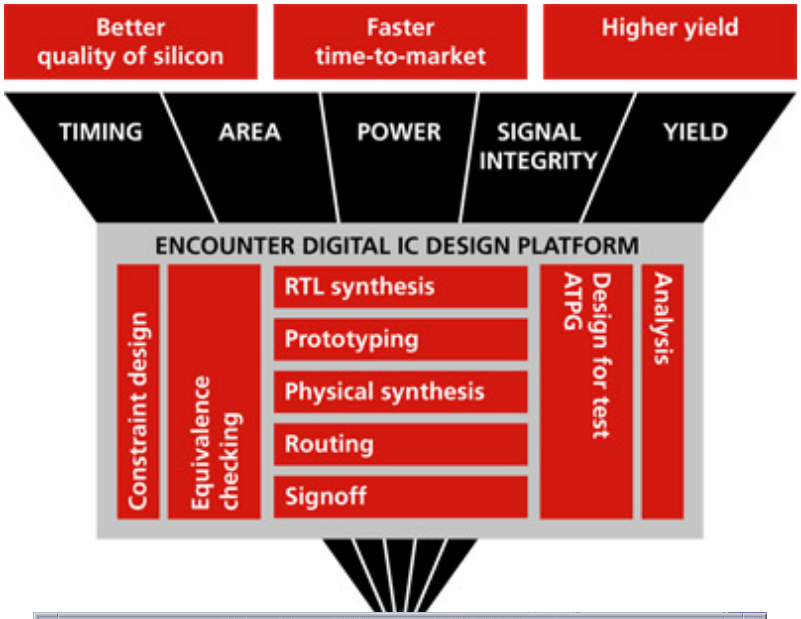
http://www.model.com/resources/student_edition/download.asp

Carte FPGA (Projet P9)

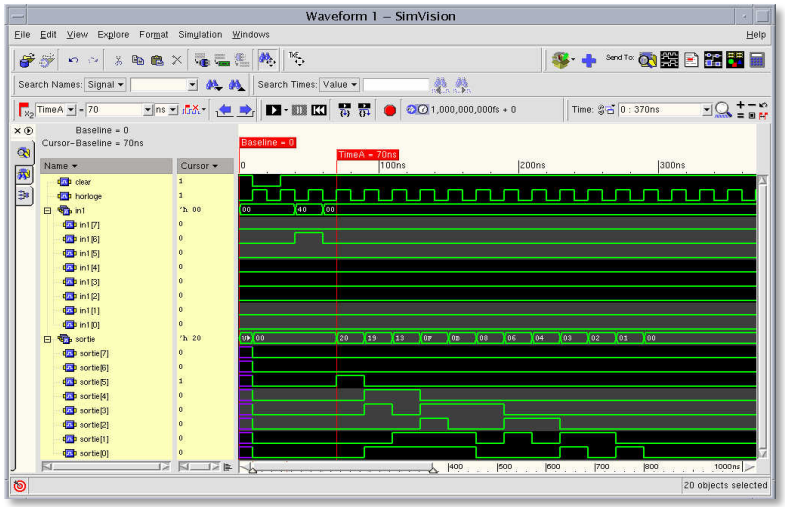
Carte Digilent Spartan 3E Starter Kit

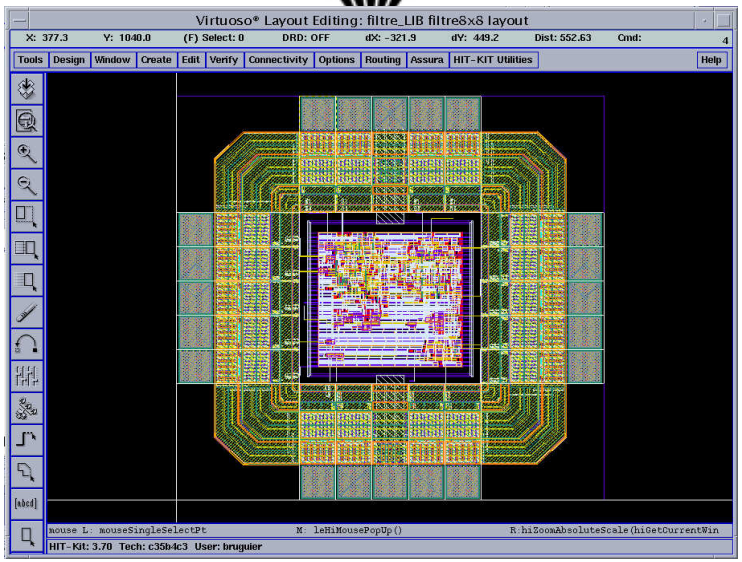
- Xilinx XC3S500E FPGA
- Xilinx XCF04 Platform Flash for storing FPGA configurations
- St Microelectronics M25P16 16Mbit Serial Flash
- Intel TE28F128 (or JS28F128) 128Mbit StrataFlash
- Linear Technologies Power Supplies
- Texas Instruments TPS75003 Triple-Supply Power Management IC
- SMSC LAN83C185 Ethernet PHY
- Micron 256Mbit DDR SDRAM

Flot de conception Silicium (Projet P2)

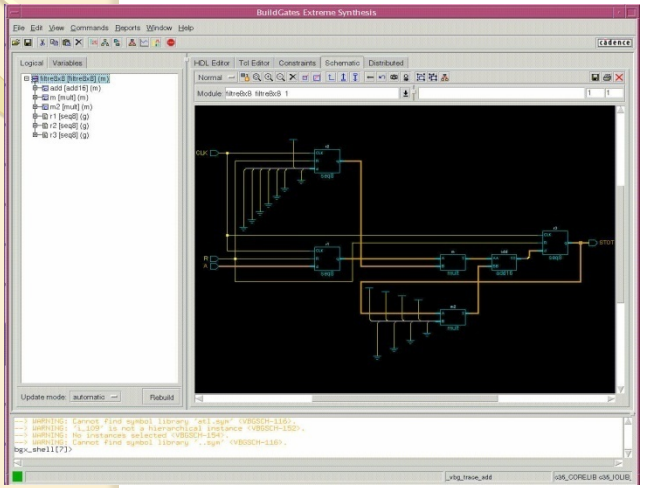


cadence



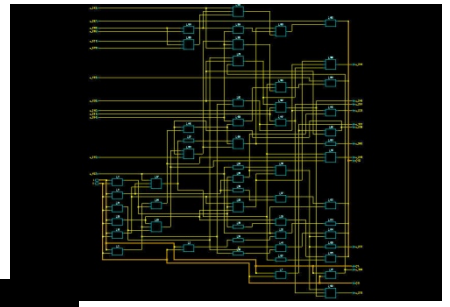


Flot de conception Silicium (Projet P2)

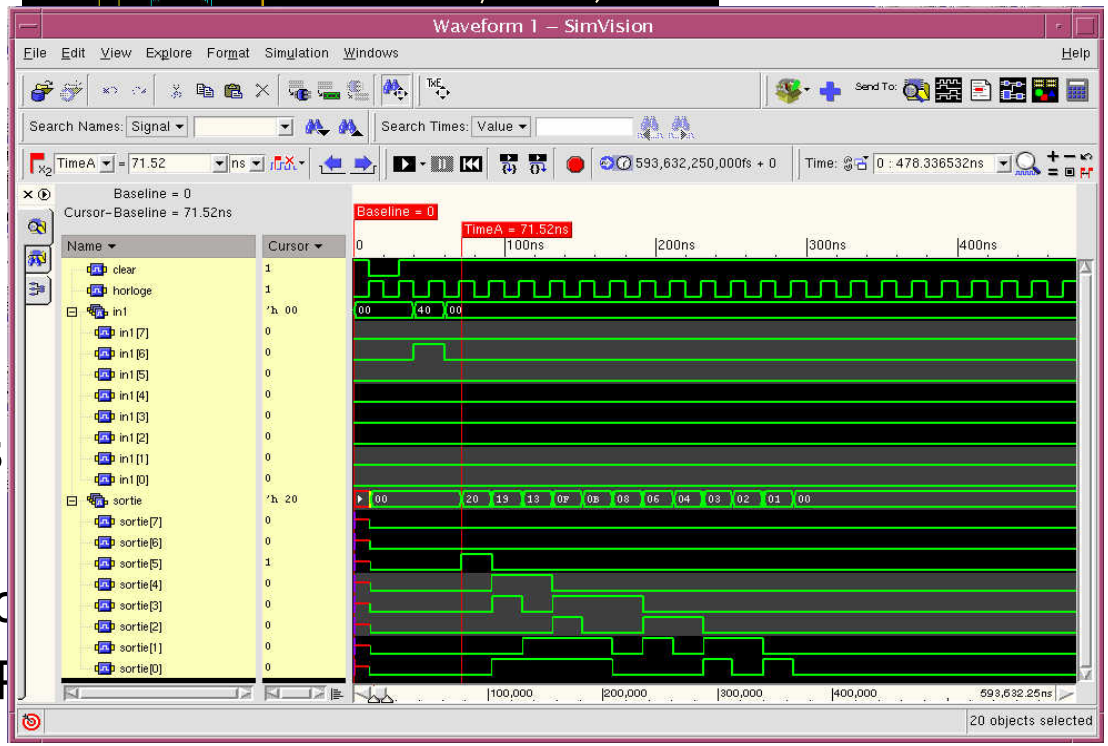


(a)

Additionneur



(b)



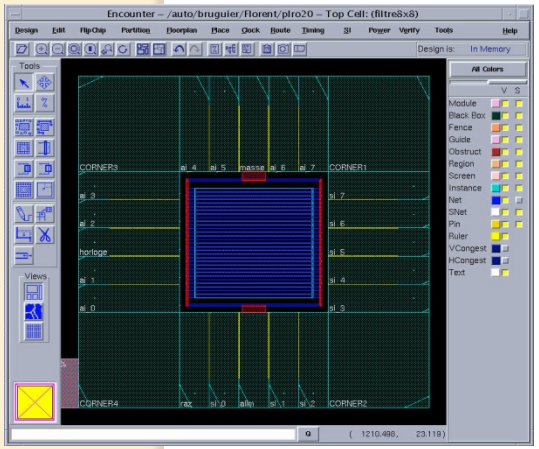
Simulation Post-Synthèse

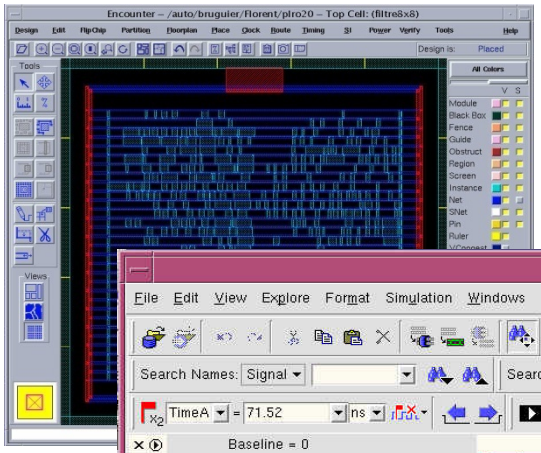
Méthodologie suivie

- Lecture et analyse des fichiers
- Synthèse générique (a)
- Synthèse spécifique sur techno
- Extraction de métriques (S, f, P

Flot de conception Silicium (Projet P2)

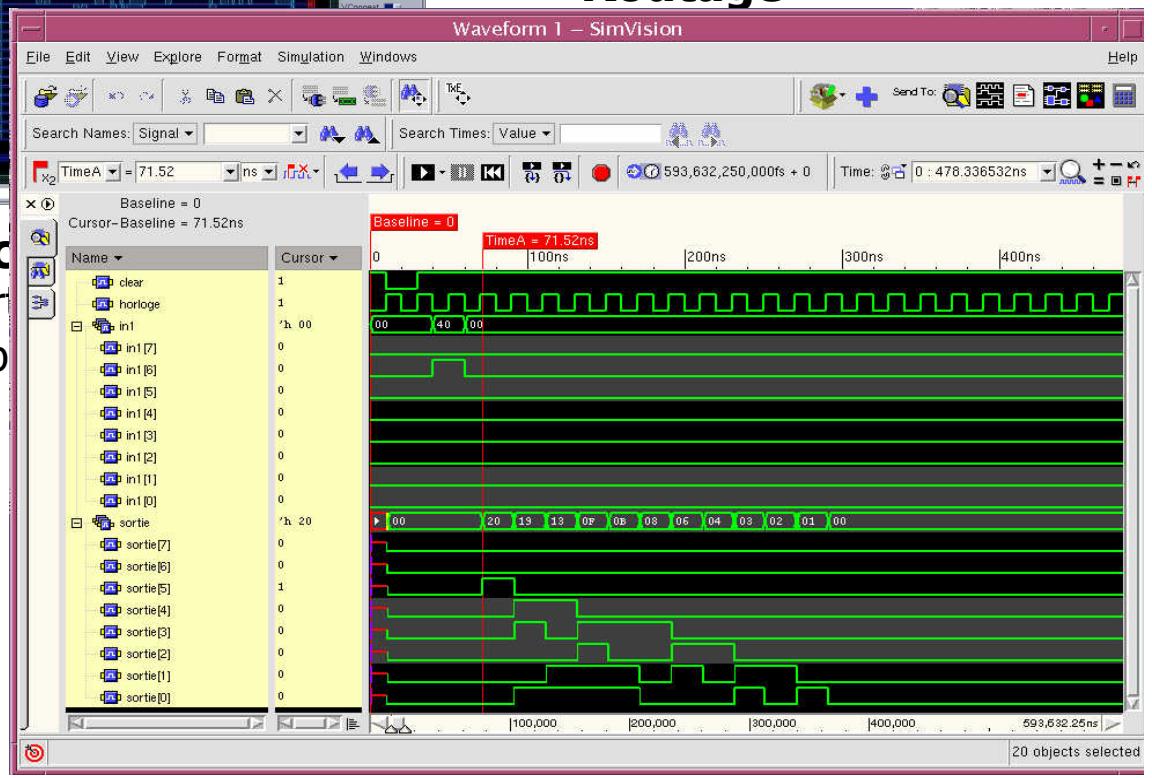
Floorplanning
- Insertion des I/O Pads
- Géométrie
- Rails d'alimentation



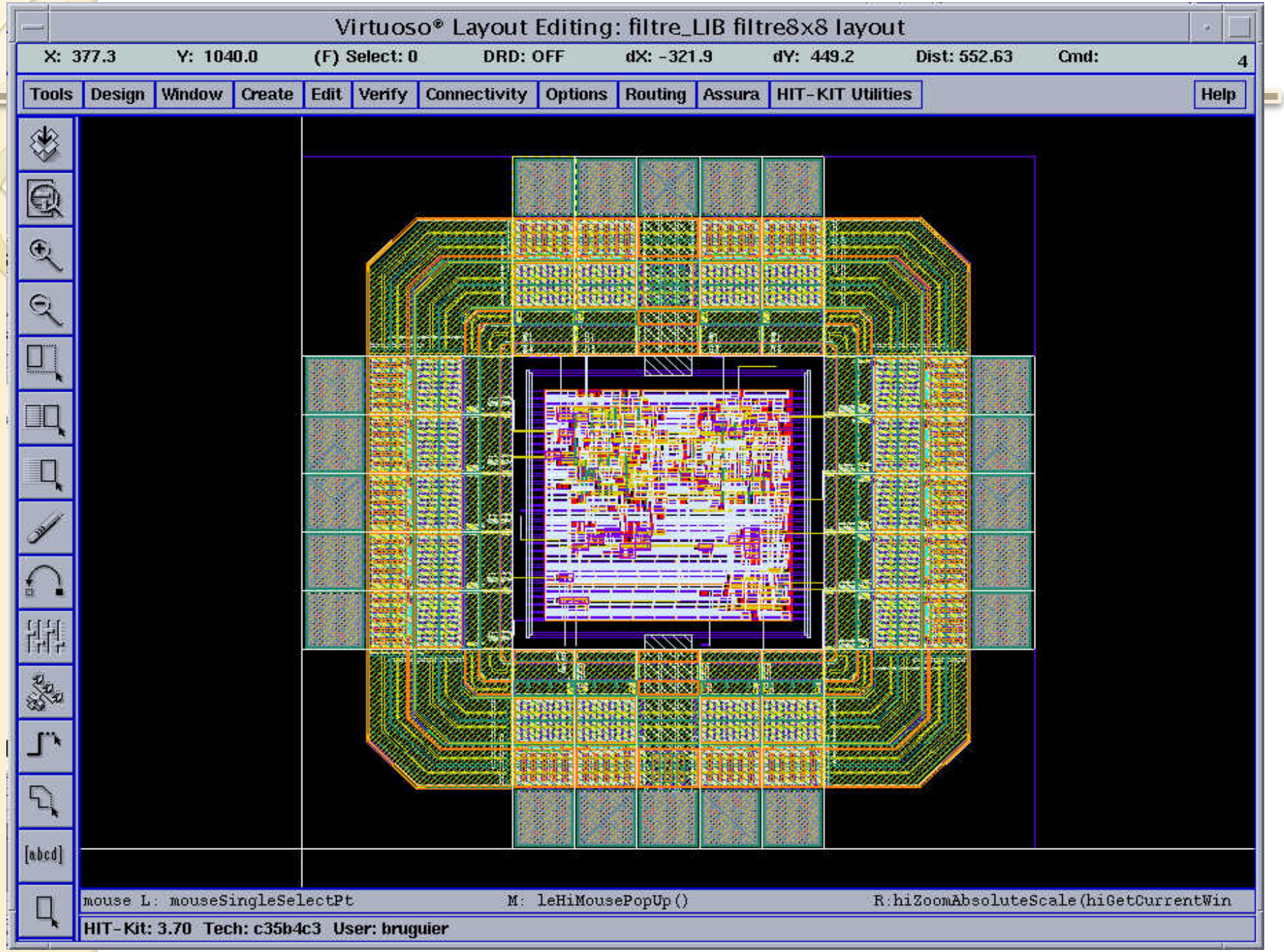


Routage

Plac
- Por
- Arb



Simulation Post-P&R


Virtuoso® Layout Editing: filtre_LIB filtre8x8 layout
X: 377.3
Y: 1040.0
(F) Select: 0
DRD: OFF
dX: -321.9
dY: 449.2
Dist: 552.63
Cmd:
4
Tools
Design
Window
Create
Edit
Verify
Connectivity
Options
Routing
Assura
HIT-KIT Utilities
Help
[abcd]
mouse L: mouseSingleSelectPt
M: leHiMousePopUp()
R:hiZoomAbsoluteScale(hiGetCurrentWin
HIT-Kit: 3.70 Tech: c35b4c3 User: bruguier