

Architecture matérielle

CE312/CE318

Module « Systèmes matériels et logiciels »

Crédit : 2

Vincent Berouille

Bureau : B206

vincent.berouille@esisar.grenoble-inp.fr

I Introduction

Informations préliminaires

CE312

- Volumes horaires :
 - 6 séances de cours (+5 CM/TD soutien Prépa)
 - 3 séances de TD (Florian Delclaux – Sagem)
 - 3 séances de TP (François Achard – SAFRAN, Nikos Polychronou CEA Leti) : distance et présentiel
- Méthode d'évaluation :
 - 1 examen écrit de 1h30, 40% de la note, un seul document autorisé « **Syntaxe VHDL** », sans calculatrice
 - note de 2^{ème} session remplace uniquement la note examen de 1^{er} session
 - **Devoir surveillé écrit de 1h pendant le semestre**, 20% de la note
 - **Contrôle Continu**, 40% de la note : moyenne des notes comptes rendus de TP, QCM et travaux à la maison

Informations préliminaires

CE318

- Volumes horaires :
 - 10 séances de cours/TD au total : 3 CM/TD harmonisation 7 séances de cours/TD
 - 3 séances de TP (Afef Kchaou – LCIS)
- Méthode d'évaluation :
 - 1 examen écrit de 1h30, 40% de la note, un seul document autorisé « Syntaxe VHDL », sans calculatrice
 - note de 2^{ème} session remplace uniquement la note examen de 1^{er} session
 - **Devoir surveillé écrit de 1h pendant le semestre**, 20% de la note
 - **Contrôle Continu**, 40% de la note : moyenne des notes comptes rendus de TP, QCM et travaux à la maison

Informations préliminaires

- Objectif du cours :

Savoir utiliser les outils de synthèses logiques et les composants reconfigurables afin de concevoir des systèmes numériques simples

Pré-requis : conception numérique (logique booléenne, portes logiques, fonctions combinatoires et séquentielles, simplification de fonctions booléenne, machine à états finis...), représentation des nombres en binaire...

Informations préliminaires

- Chamilo :
 - Polycopiés du cours (ce document en pdf)
 - *Syntaxe VHDL*
 - Sujets de TD et de TP
 - Corrections des examens des années précédents
 - QCM
- Autre :
 - Chaîne Youtube (VHDL) : LBEbooks

Informations préliminaires

Références

- VHDL, du langage au circuit, du circuit au langage, J. Weber, M. Meaudre, Masson
- Initiation au langage VHDL, Michel Aumiaux, Dunod
- VHDL, Introduction à la synthèse logique, P. Larcher, Eyrolles
- Circuits numériques et synthèse logique, un outil: VHDL, J. Weber, M. Meaudre, Masson
- VHDL, langage, modélisation, synthèse, Airiau, Bergé, Olive, Rouillard, P. P. Romandes

Informations préliminaires

- Version d'évaluation du simulateur (Modelsim) et de l'outil de synthèse VHDL (Vivado WebPACK) disponible
 - sur le site Xilinx :

[http://www.xilinx.com/ise/logic_design_prod/web
pack.htm](http://www.xilinx.com/ise/logic_design_prod/webpack.htm)

I Introduction

Plan global du cours

- **I Introduction**
 - **Pourquoi utiliser les composants reconfigurables?**
 - Architecture des FPGA
 - VHDL flot de conception
- **II Les bases du VHDL pour la synthèse**

I Introduction

Contexte « systèmes embarqués »

- Développement sous de **fortes contraintes** de miniaturisation, coût, performances temporelles, consommation, etc....
- Réduction du "**Time-To-Market**" (TTM)
- Adoption de techniques de conception à base de composants reconfigurables :
 - **VHDL** : « **V**ery high speed integrated circuit **H**ardware **D**escription **L**anguage »
 - Simulation et synthèse automatique

I Introduction

Pourquoi utiliser la logique reconfigurable?

- La logique reconfigurable offre **plus de performances** qu'un processeur
- Mais **moins de flexibilité**

I Introduction

Pourquoi utiliser la logique reconfigurable?

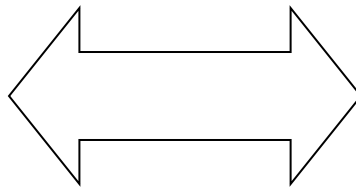
Pourquoi un composant reconfigurable peut-il être plus rapide qu'un processeur?

Processeurs

Exécution temporelle :

- Séquentiel
- ✓ Flexibilité
- ✗ Performance

PLD



Logique reconfigurable

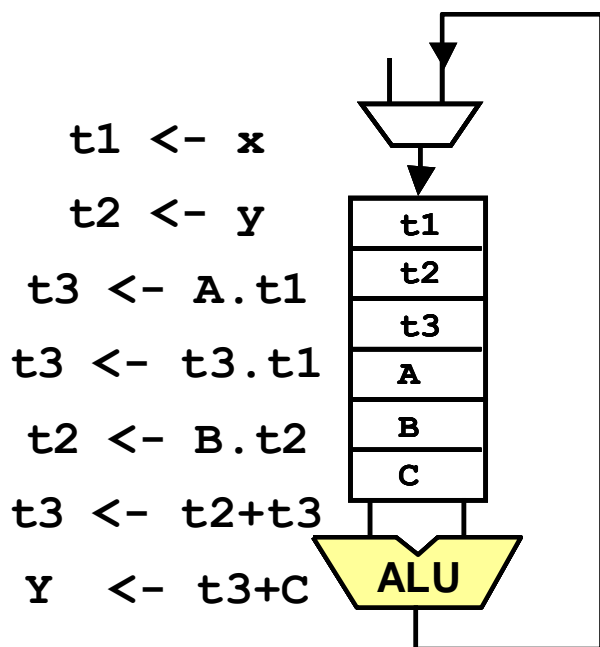
Exécution spatiale :

- Parallélisme
- ✓ Performance
- ✗ Flexibilité

I Introduction

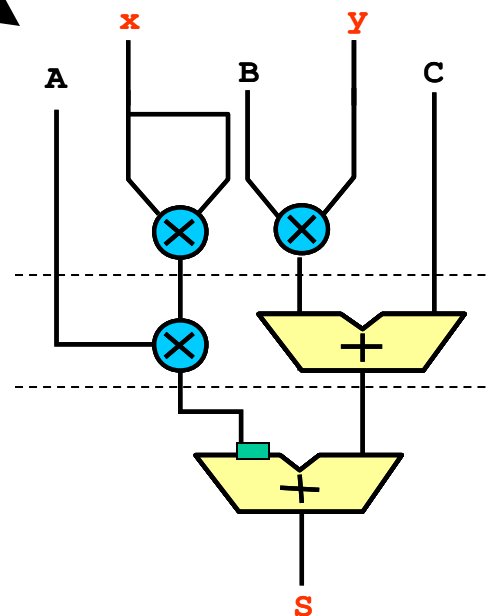
Pourquoi utiliser la logique reconfigurable?

$$S = Ax^2 + By + C$$



#CYCLES: 7

PLD



#CYCLES: 3

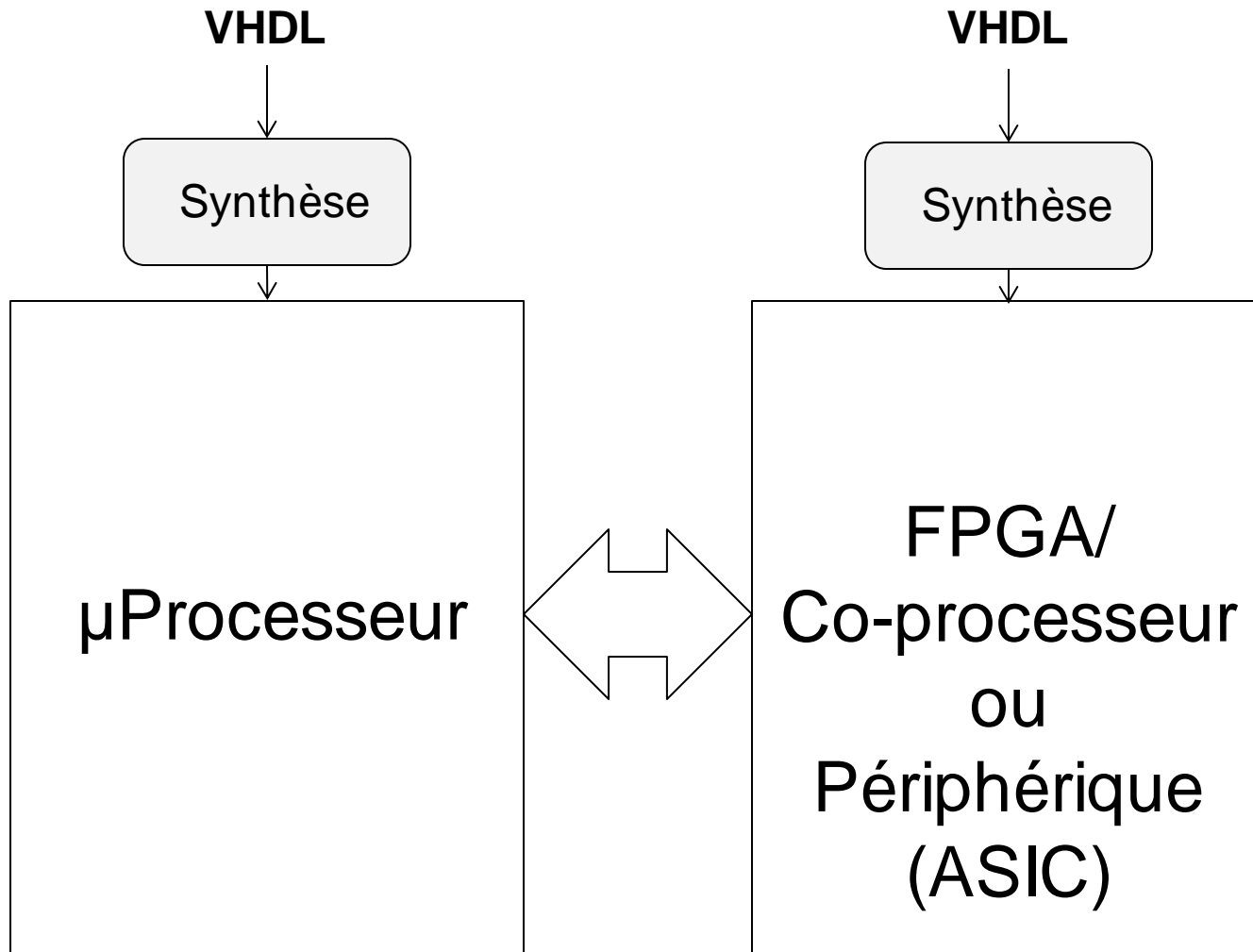
I Introduction

Quand utiliser la logique reconfigurable?

- Choix entre processeur (logique programmée, logiciel) et composant reconfigurable (logique câblée) en fonction de la périodicité des événements :
 - ps – ns : matériel (FPGA)
 - μ s : matériel ou logiciel
 - ms : logiciel (par exemple, lecture de la position d'interrupteurs, écriture sur les diodes d'afficheurs 7 segments)

I Introduction

Quand utiliser le VHDL?



Le VHDL permet de concevoir toutes les architectures numériques

I Introduction

Cibles des outils de synthèse logique

- Le VHDL permet de concevoir des systèmes sur des cibles matérielles différentes
- Concevoir sur :
 - Circuit reconfigurable (CPLD/FPGA) : de l'ordre de 10M à 100M de transistors
 - Circuit non reconfigurable (ASIC/SoC : processeurs, multi-cores) : de 100M à 10Md de transistors
 - Circuit mixte (SoPC ou eFPGA) mélangeant parties reconfigurables et non reconfigurables (10Md à 50Md de transistors)

I Introduction

Cibles des outils de synthèse logique

Glossaire :

- CPLD : Complex Programmable Logic Device
- FPGA : Field Programmable Gate Array
- ASIC : Application Specific Integrated Circuit
- SoC : System-on-Chip
- SoPC : System-on-Programmable-Chip
- eFPGA : embedded FPGA

I Introduction

Plan global du cours

- **I Introduction**

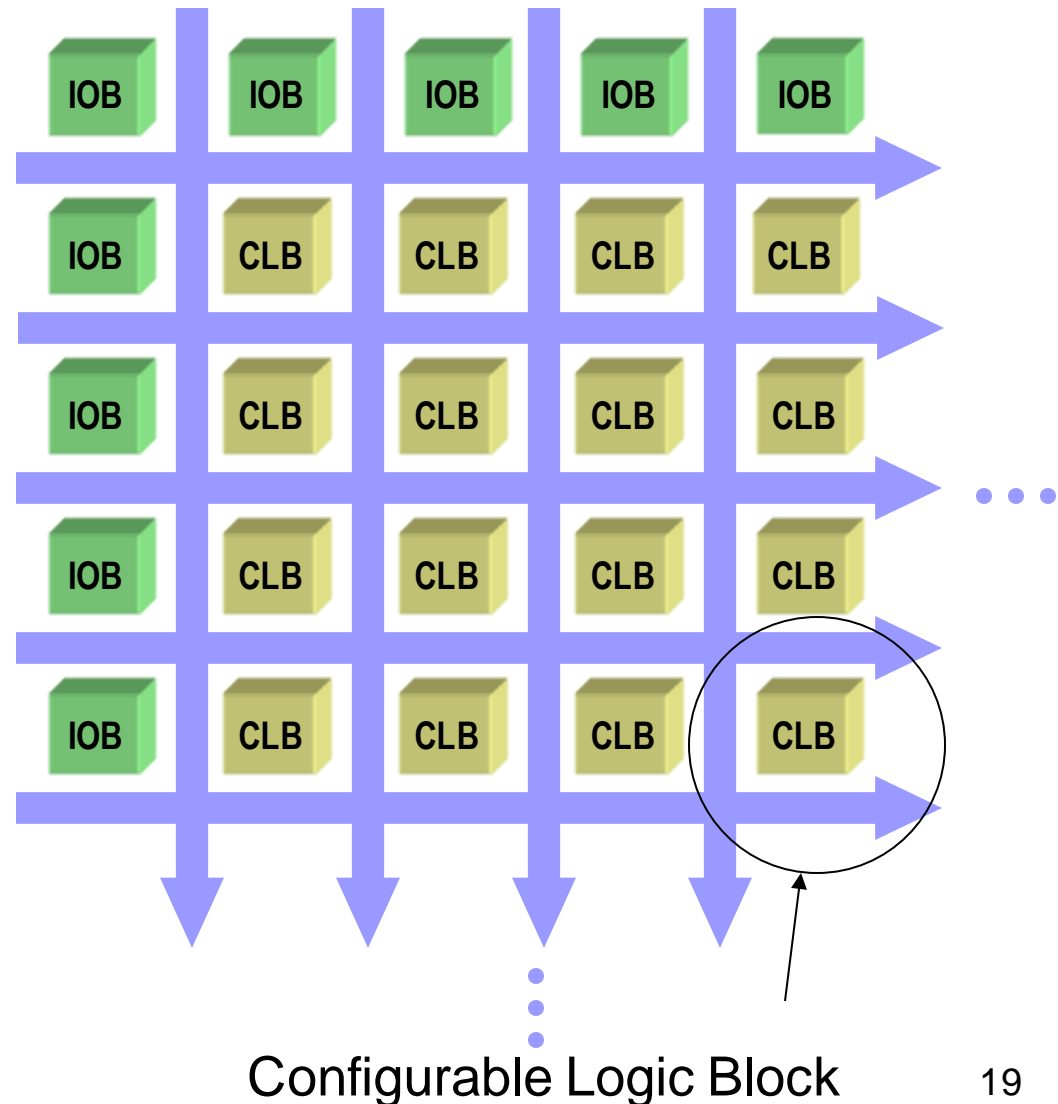
- Pourquoi utiliser les composants reconfigurables?
- **Architecture des FPGA SRAM**
- VHDL flot de conception

- **II Les bases du VHDL pour la synthèse**

I Introduction

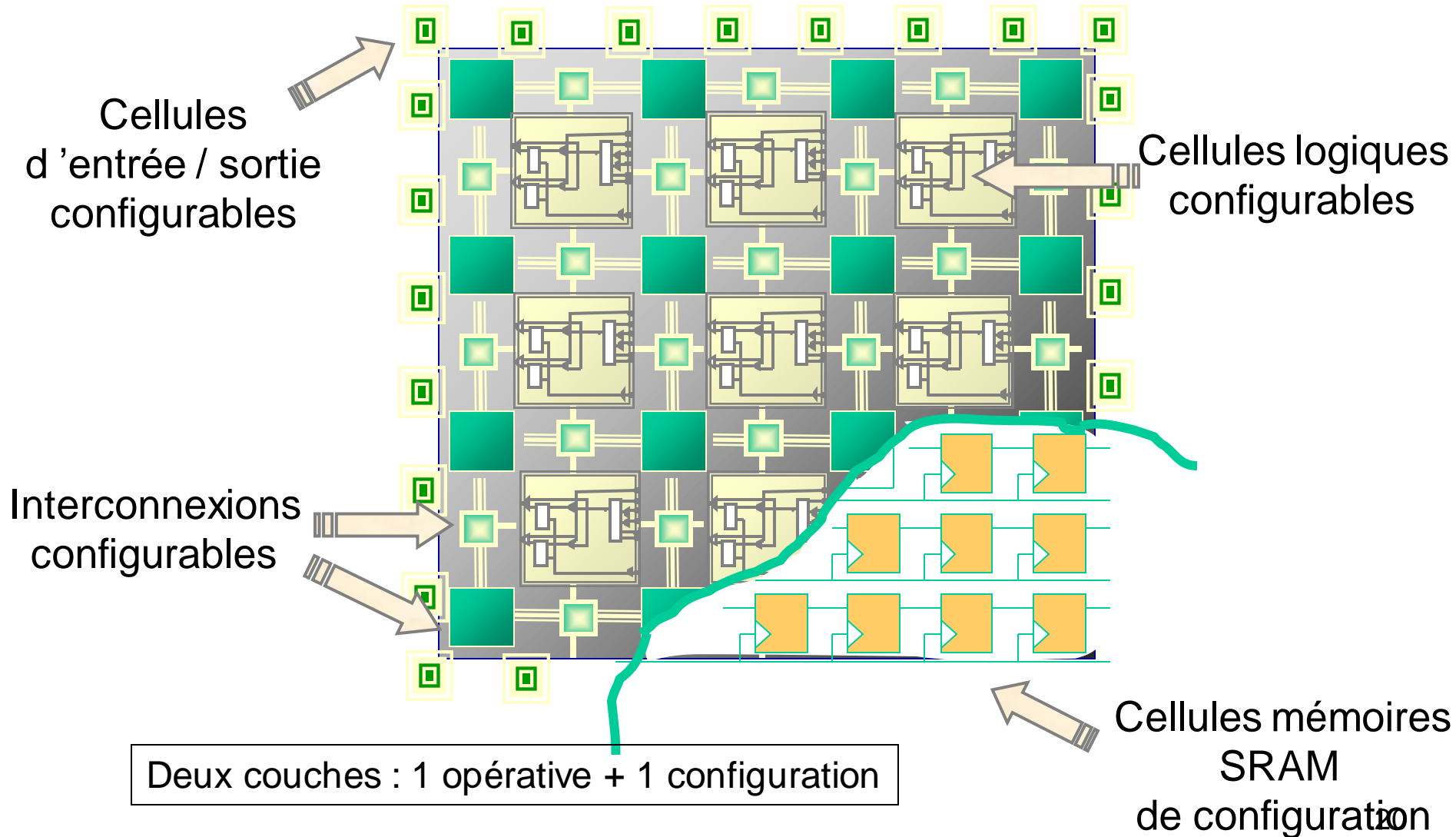
Architecture FPGA SRAM

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



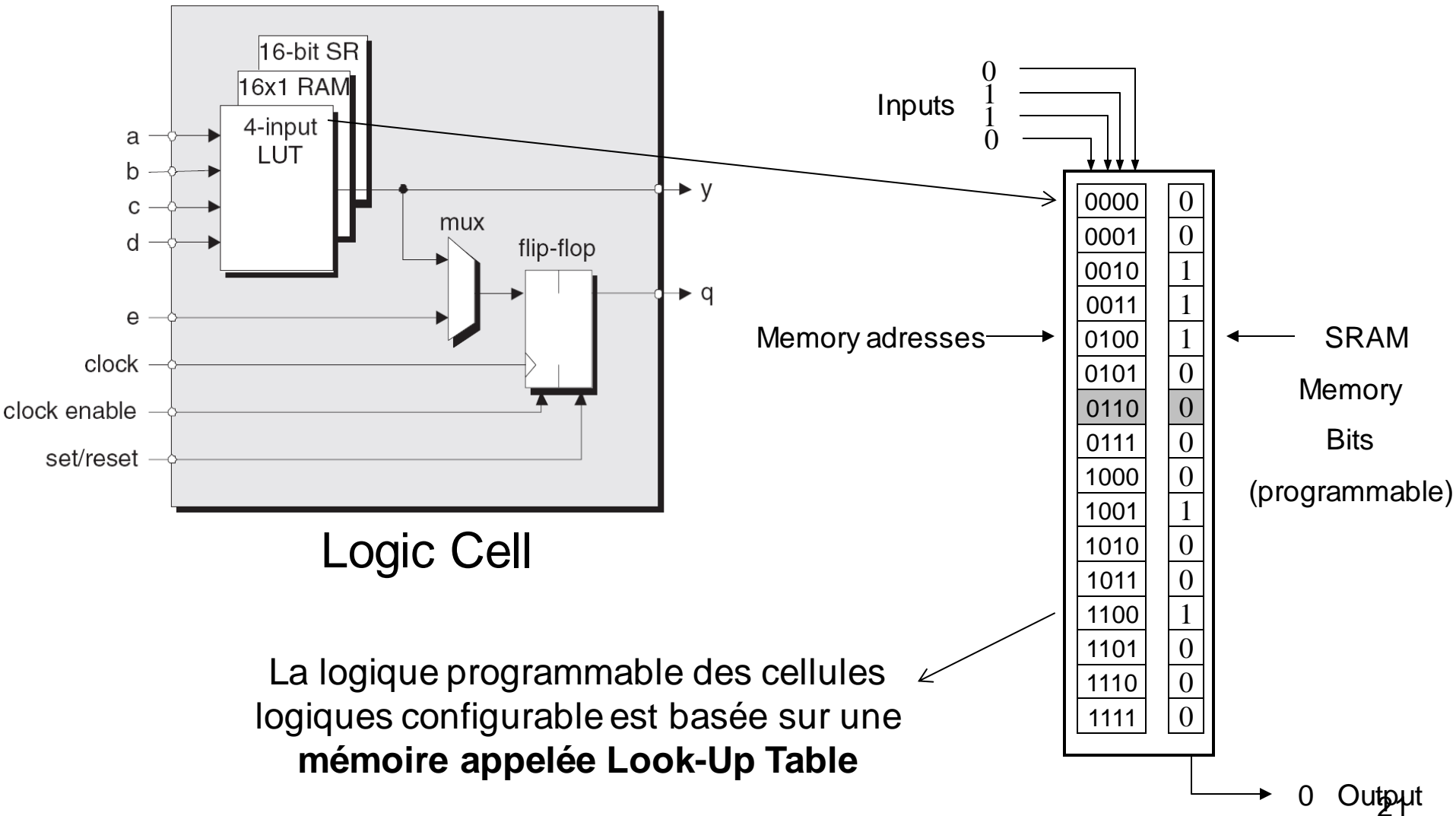
I Introduction

Architecture FPGA SRAM



I Introduction

Architecture FPGA SRAM



I Introduction

Plan global du cours

- **I Introduction**

- Pourquoi utiliser les composants reconfigurables?
- Architecture des FPGA
- **VHDL et flot de conception**

- **II Les bases du VHDL pour la synthèse**

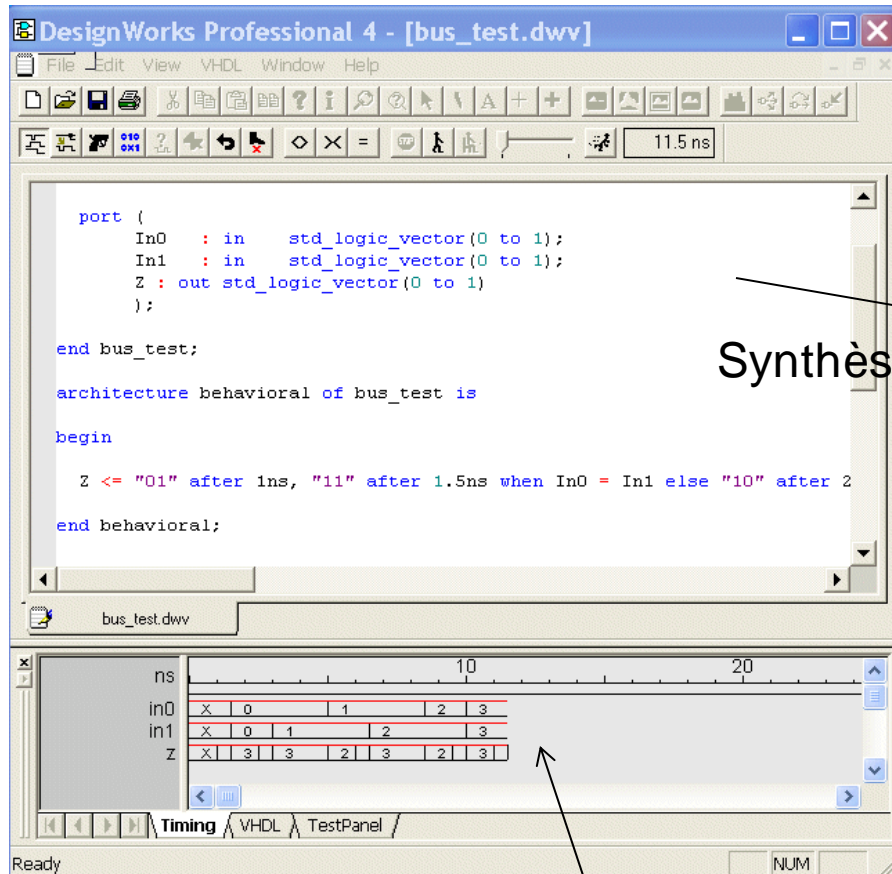
I Introduction

VHDL

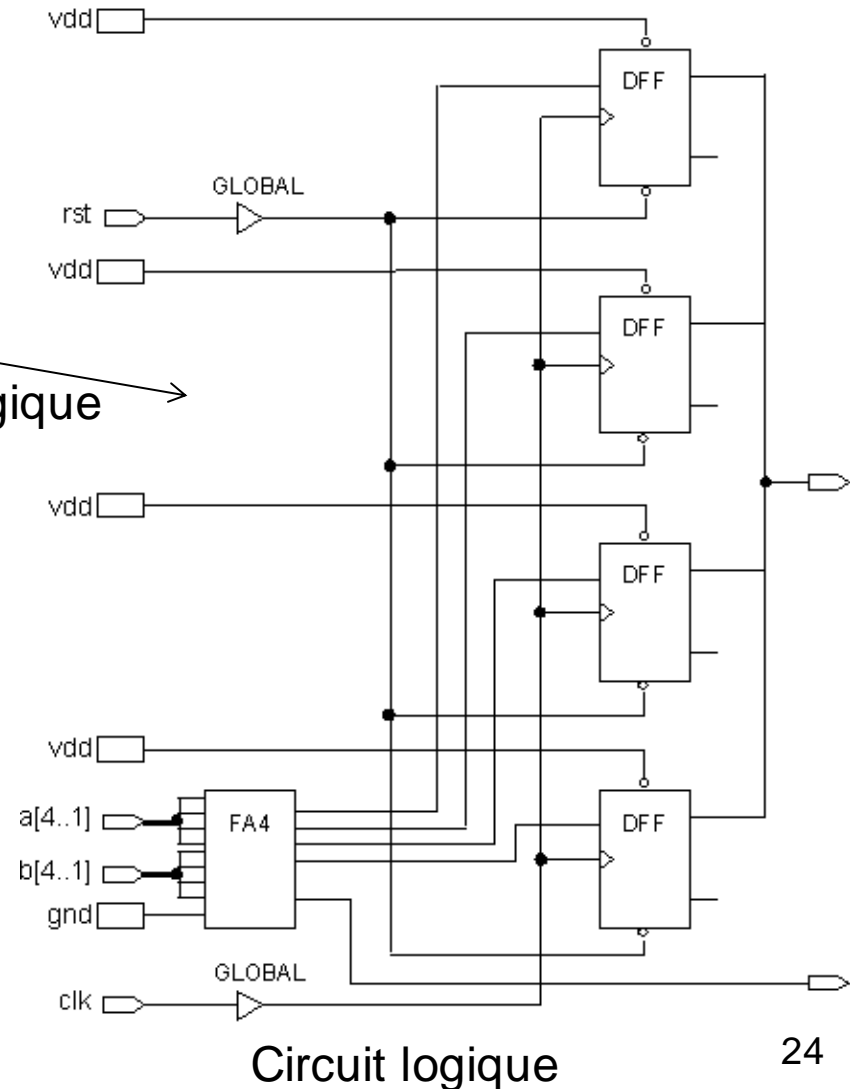
- Le VHDL permet de **ne plus manipuler de schéma** :
 - Constructions d'un langage de haut-niveau
 - Bibliothèques (pour la réutilisation de composants/fonctions)
- Le VHDL est **compatible avec les outils de synthèse et de simulation**
- C'est un **standard**

I Introduction

Environnement de conception VHDL



Synthèse logique



Chronogramme pour la simulation

Circuit logique

I Introduction

Synthèse

- Synthèse logique : **passage du VHDL niveau RTL au VHDL niveau porte**
 - RTL : *Register Transfer Level*
 - *Description des opérateurs et de l'ordonnancement des opérations dans le temps*
 - Niveau porte : schéma de portes logiques

C'est l'étape majeure du flot de conception


I Introduction

Défauts de la synthèse

- Perte du contrôle de l'implémentation bas-niveau à cause du niveau d'abstraction utilisé
 - **Il faut maîtriser l'outil!**
- Qualité des outils de synthèse variable
 - Chaque outil donne des implantations différentes (car ils utilisent des algorithmes spécifiques, ou des contraintes par défaut différentes)

I Introduction

Flot de conception

- 
- 1 Décrire les spécifications
 - 2 Coder le circuit en VHDL
 - 3 Simuler le code source
 - 4 Synthétiser, optimiser, placer et router le circuit dans une cible
 - 5 Simuler le circuit « après-synthèse »
 - 6 Programmer le composant
 - 7 Tester le composant

I Introduction

1 Décrire les spécifications

- Les spécifications permettent de fixer :
 - La fonction du circuit
 - Son initialisation
 - La fréquence maximale d'utilisation
 - ...
- VHDL comportemental :
 - Description des relations liant entrées et sorties

I Introduction

2 Coder le circuit en VHDL

- Coder le circuit :
 - Penser votre code en terme de matériel =
« **Se placer à la place du logiciel de synthèse** »

I Introduction

2 Coder le circuit en VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
entity synth is
port(    A, B, C, D : in integer;
        sel : in std_logic_vector(1 downto 0);
        Z : out integer);
end entity synth;

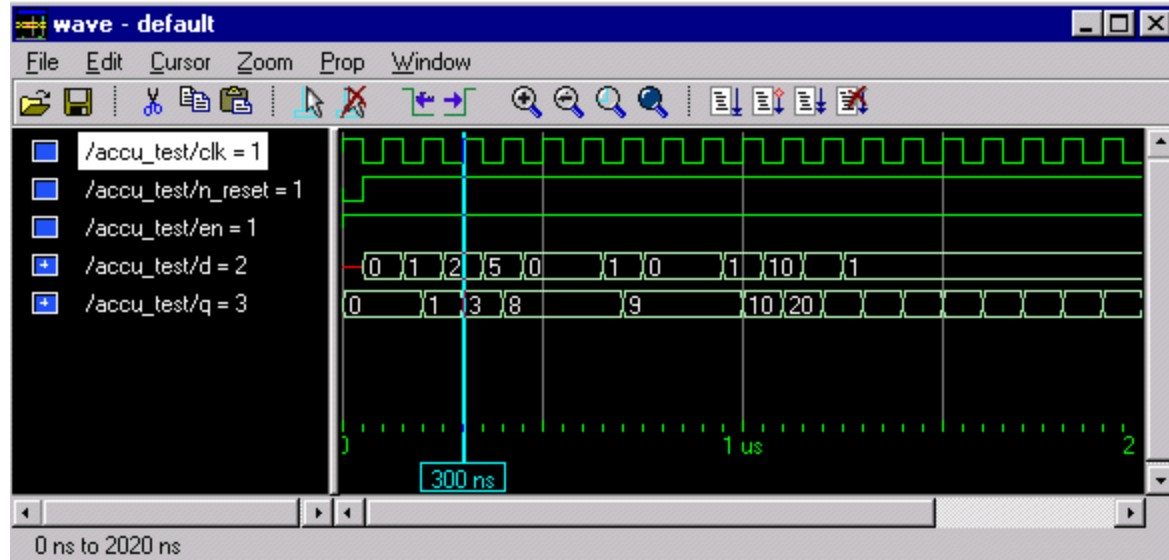
architecture behavioral of synth is
begin
with sel select
    Z <=    A+B when "00",
           C+D when "10",
           0 when others;
end architecture behavioral;
```

- Taille de(s) additionneur(s)?
- Plusieurs additionneurs?
- Quand plusieurs implémentations sont possibles **votre code doit guider les choix de l'outil de synthèse**

I Introduction

3 Simuler le code source (avant synthèse)

- Simuler son code avant de faire la synthèse permet d'éviter des erreurs de conception



I Introduction

4 Synthétiser, optimiser et ajuster

- Les étapes de la synthèse :
 - « *Inférence* » d'opérateurs
 - Inférence = reconnaissance de structures connues par l'outil de synthèse (additionneur, mémoire...)
 - Optimisation logique
 - « Placement - Routage »

I Introduction

4 Synthétiser, optimiser et ajuster

- De nombreuses éditeurs de logiciel ont créé des outils de synthèse :
 - Synopsys, Cadence , Mentor Graphics, ...
- Les fabricants de PLD (*fondeurs*) proposent leurs outils gratuitement
- Xilinx, Cypress, Altera...

I Introduction

5 Simuler le circuit « après-synthèse »

- Vérifier, les fonctionnalités mais aussi les **temps de propagation**
- Si échecs alors :
 - Modifier son code VHDL
 - Changer les contraintes de l'outil de synthèse
 - Changer de circuits cibles (et donc de fréquence maximale d'horloge)

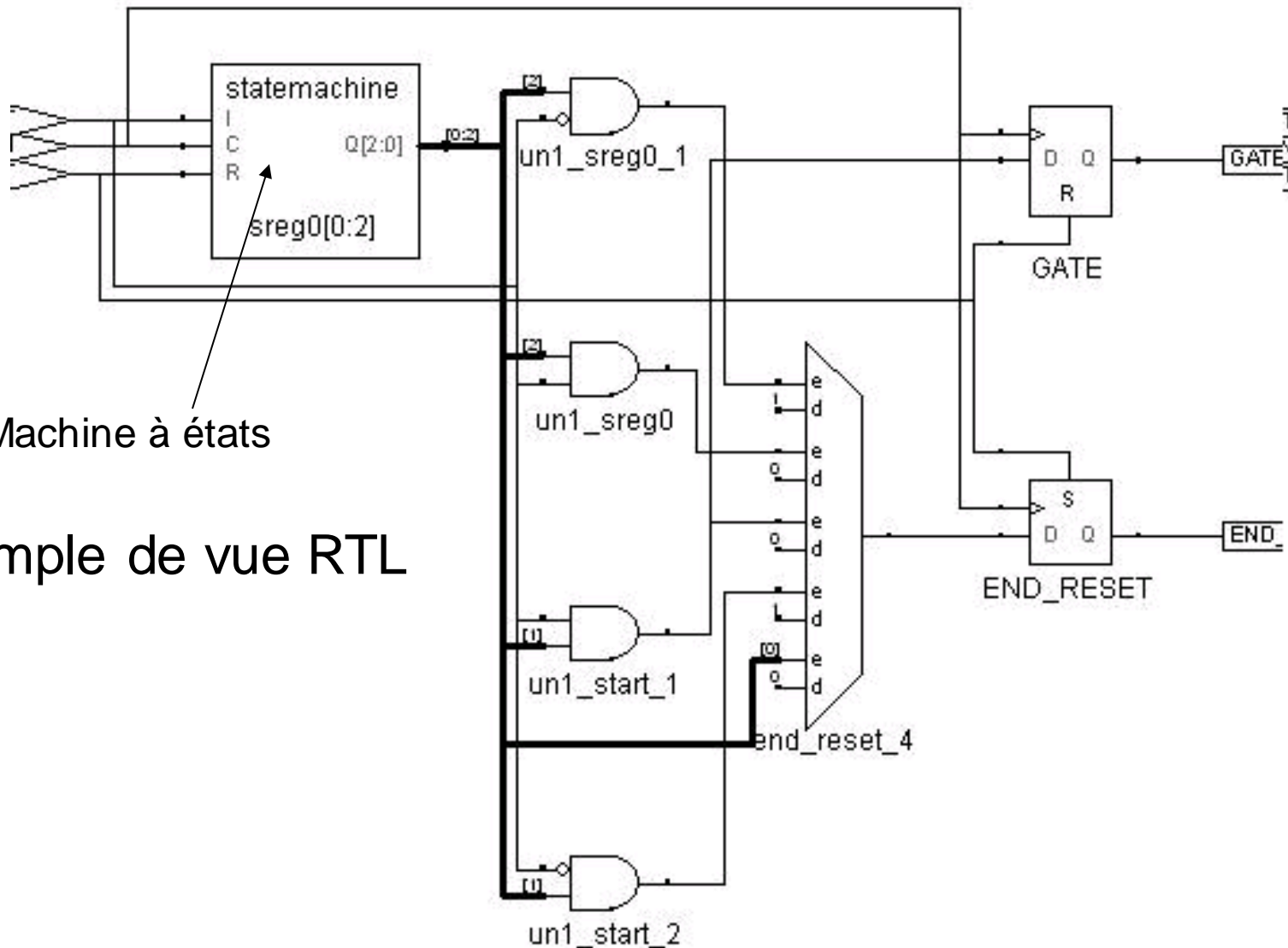
I Introduction

5 Simuler le circuit « après-synthèse »

- Simulation post-synthèse possible mais dans la majorité des cas :
 - Pas de simulation fonctionnelle après synthèse, car :
 1. *"il n'y a pas de raison pour que la synthèse soit fausse" ...*
 2. *"les temps de simulations seraient alors trop longs"*
- => analyse directe des rapports de synthèse :
Vue RTL, nombre et type des composants
utilisés, fréquence maximale...**

I Introduction

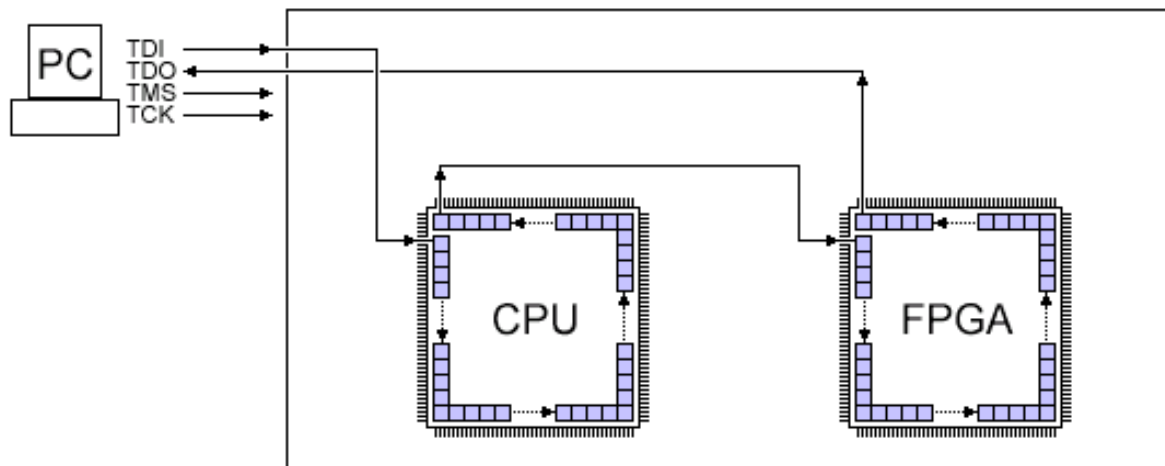
5 Simuler le circuit « après-synthèse »



I Introduction

6 Programmer le composant

- La synthèse produit un fichier binaire (Bitstream)
- « *In-System Programmability (ISP)* » : programmation du composant sur la carte
- Un PC permet de programmer le composant reconfigurable



I Introduction

Synthèse du flot de conception

