

## Introduction au microprocesseur

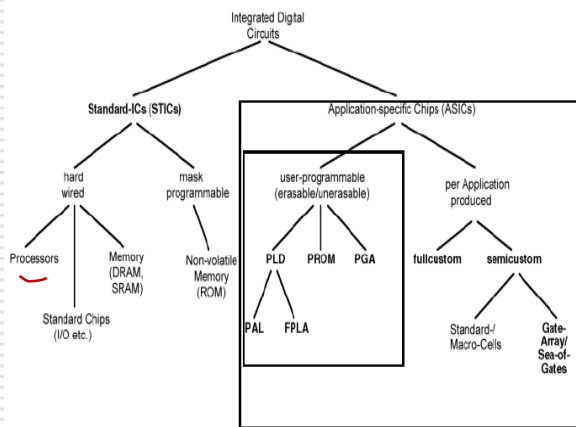
Du câblé au programmé

1

## Solutions numériques

- 1- Définition du problème
- 2- Conception d'une solution
  - VHDL
  - Schéma ?
  - Autres :
- 3- Réalisation
- 4- Vérification
- 5- etc.
- 6- etc.

2



## Logique câblée

- Toute réalisation d'un circuit spécifique à une fonction
  - Circuit intégré spécifique
  - Circuit programmable
  - Etc.
    - => logique câblée
    - « hardwired »
    - Câblée en 'dur'

4

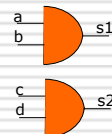
## Limites de la logique câblée

- Modification fonction => recommencer conception-réalisation
  - Fonctions ne peut être modifiée « instantanément »
- Réutilisation de ressources
  - Peu, sinon conception trop compliquée
  - Extension en surface silicium
    - N'est pas infinie !

5

## Réutiliser la logique

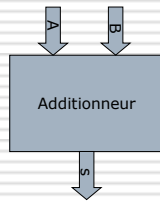
- Entity...
  - end
- Architecture
  - Begin
    - S1 = a and b;
    - S2 = c and d;
  - End titi;
- 2 and en VHDL => 2 portes et réalisées
- Comment réutiliser les ressources ?



6

## Réutilisation de Ressources

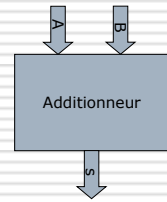
- ❑ Réaliser un additionneur n bits
- ❑ Bien l'optimiser



7

## A + B

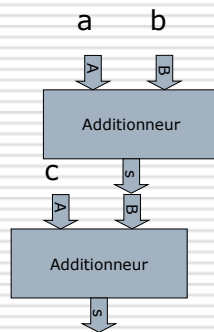
- ❑ Positionner A et B fil par fil !
- ❑ Positionner F !
- ❑ Recueillir Sortie S



8

## a + b + c ?!

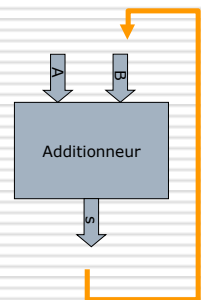
- ❑ Soit utiliser un deuxième additionneur



9

## a + b + c ?!

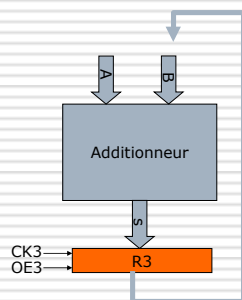
- ❑ Soit utiliser un seul additionneur, et appliquer successivement :
  - A et B, et récupérer S,
  - puis appliquer C et s, et récupérer S finale
- ❑ Mais il faut mémoriser S entre temps



10

## a + b + c ?!

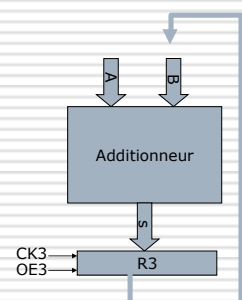
- ❑ Soit utiliser un seul additionneur, et appliquer successivement :
  - A et B, et récupérer S,
  - puis appliquer c et s, et récupérer s finale
- ❑ Mais il faut mémoriser s entre temps



11

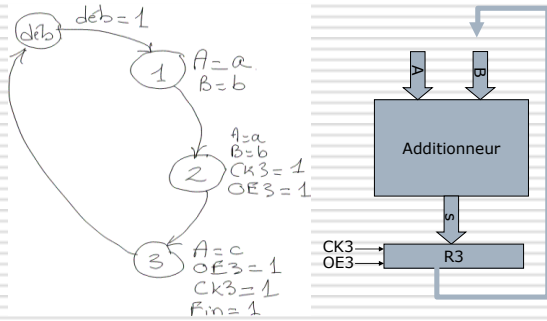
## Séquence

- ❑ 1ere étape
  - A + B et S dans un registre
- ❑ 2e étape
  - S dans R3
- ❑ 3e étape
  - R3 sur R2
- ❑ 4e étape
  - S dans R3
- ❑ 5e étape
  - Fin calcul



12

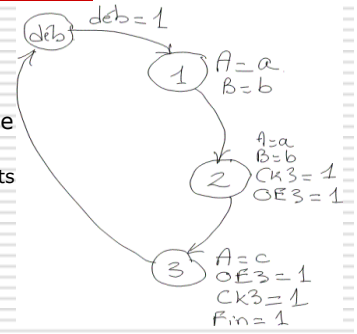
## Diagramme d'états



13

## Machine à états

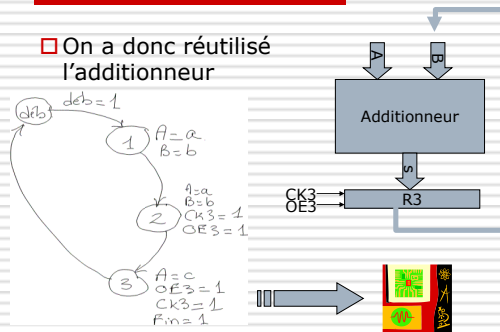
- Modélisation en VHDL
- Réalisation directe en silicium
  - Pas de composants programmables



14

## Réutilisation de ressources

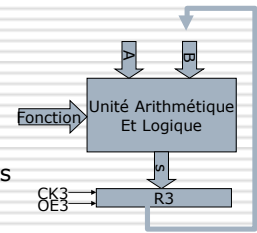
- On a donc réutilisé l'additionneur



15

## Bloc logique multifonction

- Non seulement additionneur
  - Mais additionneur, soustracteur, décalage, etc.
- Sélection de la fonction par des lignes de fonction



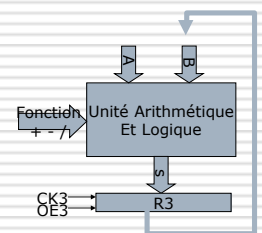
16

## Multiplication\*

- Difficile à concevoir
  - Surtout pour 16 bits \* 16 bits = 32 bits !
  - Ou 32 \* 32 = 64 bits !
- 2 solutions :
  - 1- Table de vérité géante
    - 16 bits \* 16 bits
    - $2^{16} * 2^{16}$  lignes dans la table !!
  - 2-  $a * b = a + a, b \text{ fois}$ 
    - $6 * 4 = 6 + 6 + 6 + 6$
- \*remarque TP multiplieur

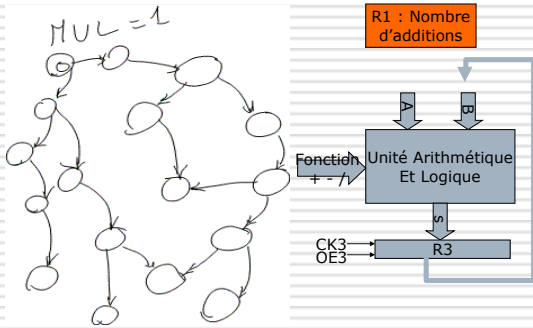
17

## Réalisation d'une multiplication



18

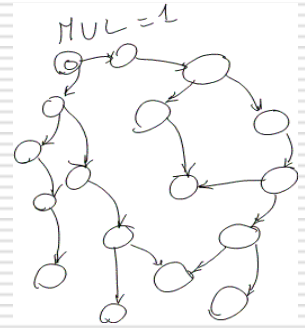
## Réalisation d'une multiplication



19

## Réutilisation de ressources

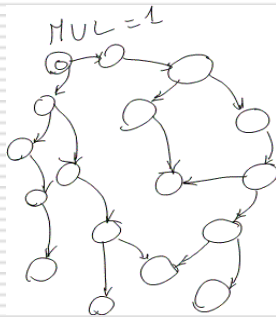
- On a donc réutilisé l'additionneur
- Mais on doit fournir :
  - Mul = 1
  - Signal qui démarre la séquence
  - Les *opérandes* a et b
- On appellera ces éléments
  - une *instruction*



20

## Terminologie

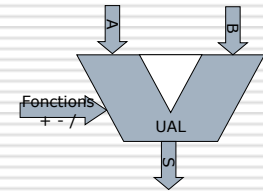
- La machine à états qui réalise l'instruction
  - Micro-séquence



21

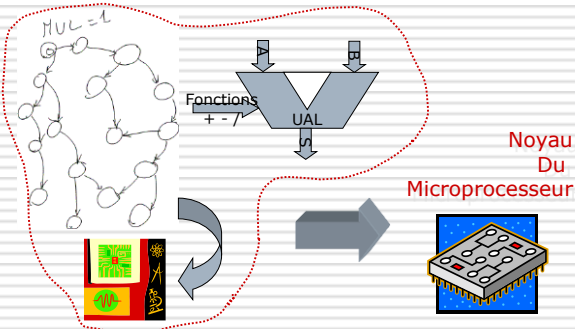
## Bloc logique multifonction

- Symbole exacte d'une UAL



22

## Microprocesseur



23

## Résumons

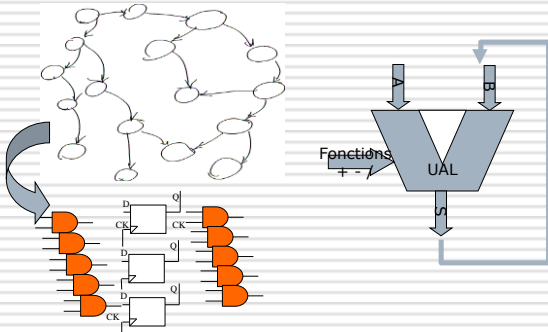
- On a réalisé une multiplication  $a * b$ 
  - En utilisant un seul additionneur
  - En multipliant
    - Le plus grand (a) par lui-même, b fois
    - $6 * 4 = 6 + 6 + 6 + 6$ 

4 fois
    - $6 * 4 = 4 + 4 + 4 + 4 + 4 + 4$ 

6 fois

24

## Micro-séquence en Machine à états



25

## Format instruction

### Code binaire

■ 1100100100100100100101010 = \$4E1F ?

Adresse  
Micro-  
séquence

Registres

### Mnémonique

□ MUL R1,R2 (R1 \* R2 -> R2)

■ R1 : source, R2 : destination

□ MUL #12,R1 (12 \* R1 -> R1)

■ 12 source = valeur directe, R1 destination

26

## Séquence déclenchée par une instruction

- Micro-code
- Micro-séquence
- Micro-programme

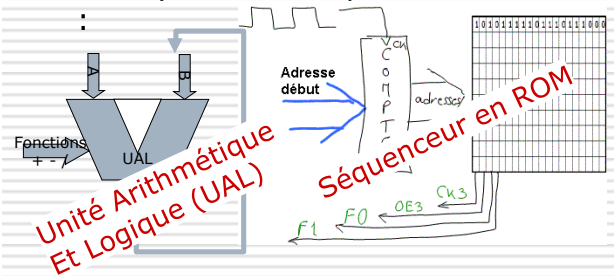
### La séquence est soit :

- Micro-codée
  - tableau en ROM
- Câblée
  - machine à états en logique (bascule D, logique transitions, logiques sorties)

27

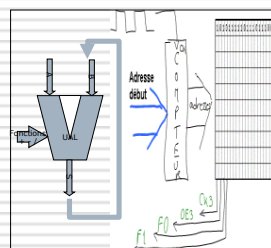
## Résumons

### Le noyau du microprocesseur est



28

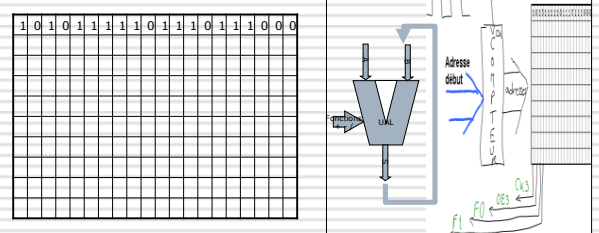
## Comment présenter les instructions



29

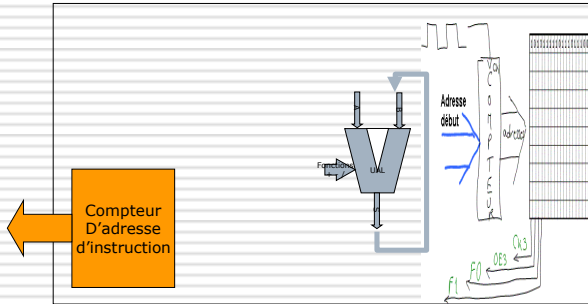
## Comment présenter les instructions

### Elles seront stockées en mémoire RAM externe



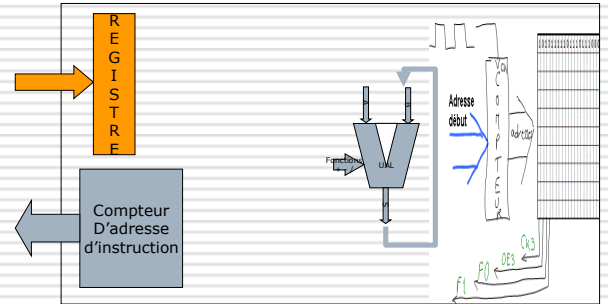
30

### Rajouter compteur d'adresse d'instruction



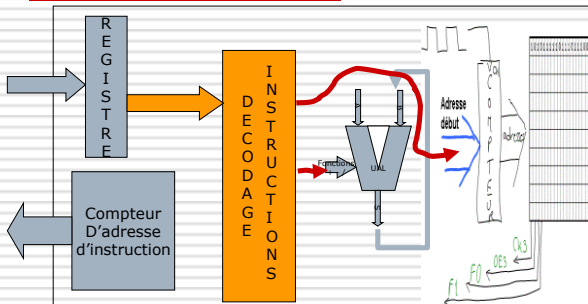
31

### Rajouter registre instruction



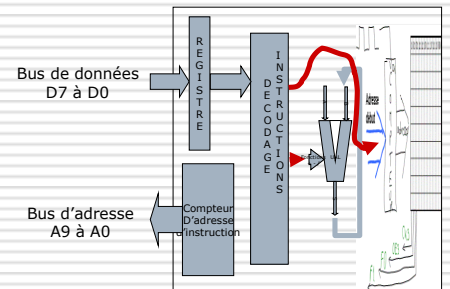
32

### Rajouter décodeur d'instructions



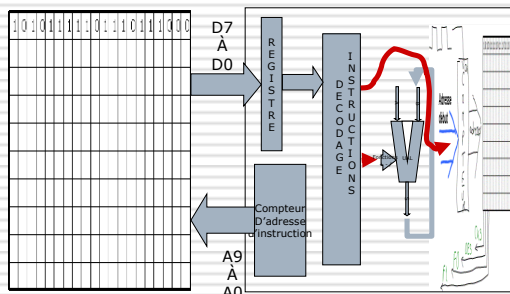
33

### Interface externe



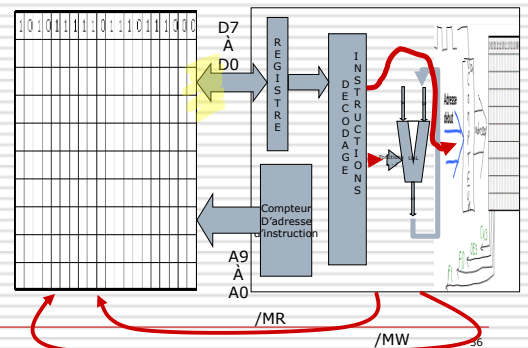
34

### Interface mémoire processeur



35

### Interface mémoire processeur



36

The diagram shows the ALU stage. On the left, a vertical bar contains 'REGISTRE' and 'INSTRUCTIONS DECODAGE'. Arrows point from these to the ALU. Below the ALU is a box labeled 'Logique de contrôle'. To the right of the ALU is a vertical bar labeled 'ALU' with 'Adresse d'entrée' and 'Adresse de sortie' labels. Arrows point from the ALU to the 'ALU' block. Below the 'ALU' block is a box labeled 'Fi' with 'F0', 'F1', 'F2', 'F3' labels. Arrows point from the 'ALU' block to the 'Fi' box.

Diagrama de um sistema de controle de temperatura baseado em lógica difusa. O sistema recebe "Temperatura ambiente" e "Temperatura do ambiente" como entradas. Essas entradas passam por "Fuzzificação" para serem processadas por um "Banco de regras fuzzy". O resultado da inferência fuzzy é então submetido a "Defuzzificação" para produzir o "Liquor de controle". Este liquor atua sobre um "Sistema de controle", que é representado por um bloco contendo uma "Região de controle" e um "Sistema de controle" propriamente dito. O sistema de controle gera uma saída que é convertida em "Temperatura ambiente" e "Temperatura do ambiente" novamente, fechando o ciclo de controle.

The diagram illustrates the flow of instructions and data in a computer system. On the left, a large memory block labeled "Mémoire d'instructions et de données" is shown with a 16x16 grid. Above it, a 16-bit address bus is labeled "1 0 0 1 0 1 1 1 1 1 1 0 1 0 1 0 0". Data flows from memory to a "Registres de données" block (8 registers) and a "Contrôleur d'adresses" block. The "Registres de données" block contains "Registres temporaires" and "Registres internes". The "Contrôleur d'adresses" block contains a "Logique de contrôle". On the right, a "Bus" connects to a "CPU" block, which is further connected to a "RAM" block.

```

graph TD
    A[Solutions numériques] --> B[Solutions câblées]
    A --> C[Solutions programmées]
    B --> D[Circuit spécifique]
    B --> E[Circuit programmable]
    C --> F[microprocesseur]
    C --> G[microcontrôleur]
    D --> H[Modélisation VHDL]
    E --> I[Synthèse]
    F --> J[Programmation  
Assembleur, langage info]
    G --> K[Compilation]
  
```

7