

Conception de circuits numériques et architecture des ordinateurs

Équipe pédagogique :

Marie Badaroux (TD), Julie Dumas (TD+TP), Florence Maraninchi (TD+TP),
Olivier Muller (TD+TP), **Frédéric Pétrot** (CM+TD), Pierre Ravenel (TD+TP)
Eduardo Tomasi Ribeiro (TP), Lionel Rieg (TD+TP),
Sébastien Viardot (TD+TP Apprentis)



Année universitaire 2024-2025

- C1 Codage des nombres en base 2, logique booléenne, portes logiques, circuits combinatoires
- C2 Circuits séquentiels
- C3 Construction de circuits complexes
- C4 Micro-architecture et fonctionnement des mémoires
- C5 Machines à état
- C6 **Synthèse de circuits PC/PO**
- C7 Optimisation de circuits PC/PO
- C8 Interprétation d'instructions - 1
- C9 Interprétation d'instructions - 2
- C10 Interprétation d'instructions - 3
- C11 Introduction aux caches

Intérêt

Mécanisme de base de construction des processeurs jusqu'au milieu des années 80.
Encore fort utilisé pour des objets à faible consommation et grande rentabilité (montre, calculatrice, ...)

Plan détaillé du cours d'aujourd'hui

1 Construction de circuits

- Introduction
- Approche microprogrammée

Plan

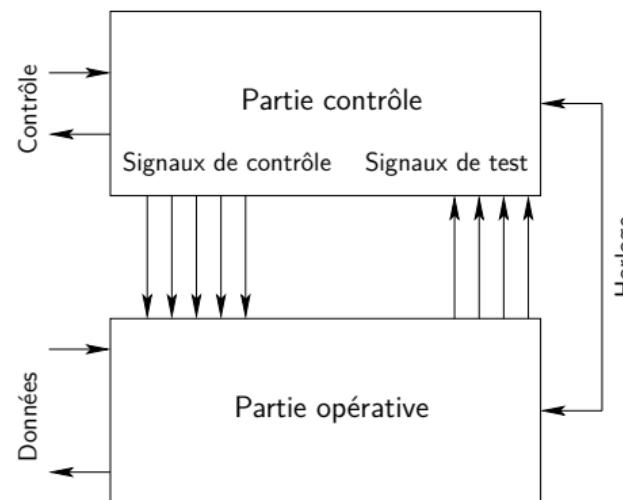
- 1 Construction de circuits
 - Introduction
 - Approche microprogrammée

Introduction

Problématique :

Implantation matériel d'un algorithme

Solution générale :



Introduction

- Partie contrôle :
génère des commandes en fonction de signaux de test
machine à états
- Partie opérative :
ensemble d'opérateurs effectuant des opérations en fonction des commandes
interconnexion de registres, additionneurs, etc, sur n bits

Plusieurs approches :

- Micro-programmées : notre cas
- Cablées : optimisation du cas précédent, également vu
- Pipelinée : plusieurs PC en actions simultanément, vu en 2A
- etc ...

Introduction

Partie opérative (*data-path*) :

- Entrées : données externes, autorisation d'écriture dans les registres, commandes de multiplexeurs, code opération pour les UAL
- Sorties : données externes, informations pour les prises de décision de la partie contrôle
- Contenu : registres, mémoires vives ou mortes, opérateurs arithmétiques, multiplexeurs, bus

Partie contrôle (*control part, FSM*) :

- Entrées : signaux de contrôle externes, signaux de test, horloge
- Sorties : signaux de contrôle externes, commandes vers le chemin de données
- Contenu : logique de génération de l'état et des sorties, doit *garantir* que les dépendances de données sont bien respectées

Principe

Principe

- partitionner l'algorithme entre contrôle, PC, et calcul, PO
- connecter PC et PO avec des fils de contrôle et de test

Pour cela il faut :

- 1 identifier les variables qui deviendront des *registres*
- 2 identifier les opérations à effectuer entre variables et y associer des *opérateurs*
- 3 construire les *chemins* de calculs, à l'aide de multiplexeurs
- 4 construire le *séquencement* de l'activation des chemins de calcul

Choix d'architecture

La conception influe principalement sur le degré de parallélisme :

- nombre d'opérateurs
- nombre de registres
- complexité des opérateurs : nombre de lectures simultanées dans un banc de registers, nombre d'entrées d'un multiplexeur, d'un opérateur arithmétique, etc
- surface de silicium, longueur des chemins combinatoires, puissance consommée, etc

Construction du chemin de données : registres

Identifier les registres

Registres : Variables « conservant leur valeur » :

Relation implicite avec le séquencement prévu :
variables lues et écrites dans le même cycle disparaissent

Exemple : $x := a + b; y := x - c$

- si le « ; » est une frontière de cycle :
 - x est un registre
 - temps de calcul : additionneur à 2 entrées
- si le « ; » n'est pas une frontière de cycle :
 - x est un fil
 - temps de calcul : $2 \times$ le temps de calcul précédent

Construction du chemin de données : opérateurs

Identifier les opérateurs

- quels sont les opérations à effectuer ?
- peut-on partager des opérateurs ?

Relation implicite avec le séquencement prévu :

2 opérations identiques dans 2 cycles peuvent partager un opérateur

S'il y a plusieurs opérateurs, il faut choisir lequel utiliser

Exemple : $x := a + b; y := x - c$

- si le « ; » n'est pas une frontière de cycle :
additionneur pour + et soustracteur pour –
2 × add/sub pour + et – (dépend du contexte)



valeur de x !

- si le « ; » est une frontière de cycle :
add/sub

Construction du chemin de données : chemins

Construire les chemins de calcul

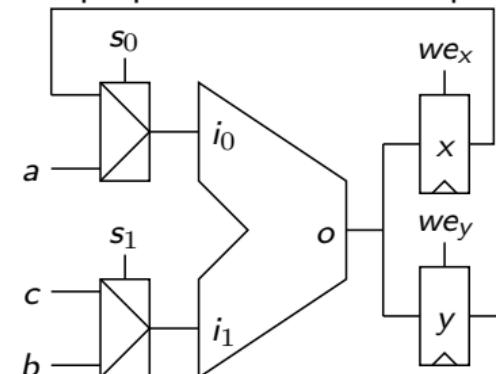
- enchaîner les opérateurs
- amener les registres et signaux sur les opérateurs à l'aide de multiplexeurs

Relation implicite avec le séquencement prévu :

quels opérateurs chaîner, quelles entrées doivent être proposées devant les opérateurs ?

Exemple : $x := a + b; y := x - c$

- si le « ; » est une frontière de cycle :
 - entrée i_0 de add/sub prend soit a , soit x
 - entrée i_1 de add/sub prend soit b , soit c



Construction du séquencement

Séquencement

Définition de l'ordre des commandes à appliquer sur la PO

Doit garantir le respect des dépendances de données

Allocation et séquencement sont interdépendant

Exemple : $x := a + b; y := x - c$

si le « ; » est une frontière de cycle

- 1 choisir la valeur des selecteurs pour présenter a et b , autoriser l'écriture dans x
- 2 choisir la valeur des selecteurs pour présenter x et c , autoriser l'écriture dans y

si le « ; » n'est pas une frontière de cycle

- 1 autoriser l'écriture dans y

Exemple

Algorithme *cordic*[†]

Data : trois variables x , y et z . On a de plus un tableau t de 30 constantes

Réultat : trois variables X , Y et Z donnant respectivement $X := \frac{(x\cos(z)-y\sin(z))}{K}$,

$$Y := \frac{(x\cos(z)+y\sin(z))}{K} \text{ et } Z := 0, \text{ avec } K = \prod_{k=0}^{\infty} \sqrt{1 + 2^{-2k}}$$

$X := x; Y := y; Z := z;$

for $i := 0$ **to** 29 **do**

$$a := \frac{X}{2^i}; b := \frac{Y}{2^i}; c := t[i];$$

if $Z > 0$ **then**

$$| X := X - b; Y := Y + a; Z := Z - c;$$

else

$$| X := X + b; Y := Y - a; Z := Z + c;$$

end if

end for

†. J.E. Volder, "The Cordic Trigonometric Computing Technique", IRE Transactions on Electronic Computers, sept. 1959, p. 330-334

Exemple : registres

Identification des registres

```

1   $X := a; Y := b; Z := c;$ 
2  for  $i := 0$  to 29 do
3       $a := \frac{X}{2^i}; b := \frac{Y}{2^i}; c := t[i];$ 
4      if  $Z > 0$  then
5           $X := X - b;$ 
6           $Y := Y + a;$ 
7           $Z := Z - c;$ 
8      else
9           $X := X + b;$ 
10          $Y := Y - a;$ 
11          $Z := Z + c;$ 
12     end if
13 end for

```

- si « ; » frontière de cycle ligne 3 :
 a, b, X, Y, Z (c inutile car i ne change pas dans boucle)
- si « ; » non frontière de cycle lignes 3, 5, 6, 7, 9, 10, 11 : X, Y, Z
mise en évidence du parallélisme potentiel des affectations
- note : i , variable de la boucle **for**, sera produite ici par l'automate

Exemple : opérateurs

Identification des opérateurs

```

1   $X := a; Y := b; Z := c;$ 
2  for  $i := 0$  to 29 do
3       $a := \frac{X}{2^i}; b := \frac{Y}{2^i}; c := t[i];$ 
4      if  $Z > 0$  then
5           $X := X - b;$ 
6           $Y := Y + a;$ 
7           $Z := Z - c;$ 
8      else
9           $X := X + b;$ 
10          $Y := Y - a;$ 
11          $Z := Z + c;$ 
12     end if
13 end for

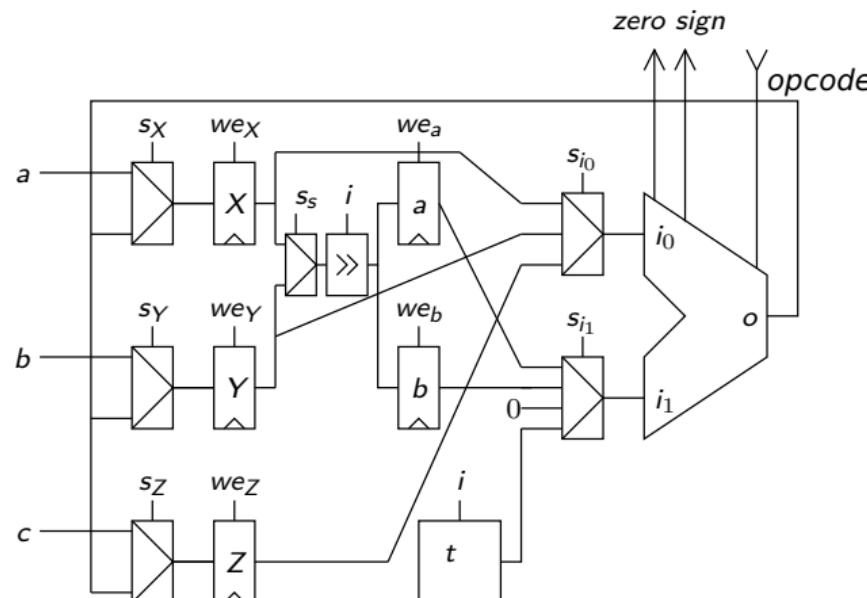
```

- ligne 3 : décaleur à droite de i positions \leadsto décaleur*
- ligne 4 : comparateur de supériorité à zéro \leadsto drapeau opérateur ou opérateur spécifique
- ligne 5 à 11 : max 2 additionneurs et 2 soustracteurs, min 1 add/sub
- note : pas d'incrémenteur de i , parcours de l'automate

*. *Barrel shifter.*

Exemples : chemins

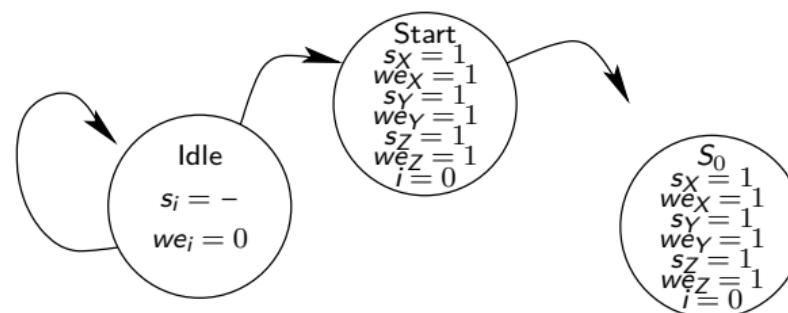
On prend une architecture très séquentielle :



Signaux verticaux issus de la partie contrôle

Exemples : instructions

On prend une architecture très séquentielle :



Exemples : séquencement

```

1  $X := a; Y := b; Z := c;$ 
2 for  $i := 0$  to 29 do
3    $a := \frac{X}{2^i}; b := \frac{Y}{2^i}; c := t[i];$ 
4   if  $Z > 0$  then
5      $X := X - b;$ 
6      $Y := Y + a;$ 
7      $Z := Z - c;$ 
8   else
9      $X := X + b;$ 
10     $Y := Y - a;$ 
11     $Z := Z + c;$ 
12   end if
13 end for

```

- automate non optimisé au tableau
- sorties automate précisées dans chaque état
- état Idle attente démarrage
- état Start ligne 1
- partie automate répétée 30 fois pour la boucle for
est-ce plus ou moins coûteux qu'un registre avec un incrémenteur et un comparateur ?
- état avec 2 successeurs ligne 4
- état avec 2 prédécesseurs ligne 12

Conclusion partielle

Synthèse PC/PO :

- technique systématique
- peut se faire sous contrainte
 - de ressources : surface de silicium liée aux opérateurs mais aussi aux multiplexeurs
 - de temps : choix entre séquentialité et parallélisme, influe fortement sur les ressources
- technique complexe : il faut faire des choix difficiles
- peut être automatisée [‡] :
synthèse dite d'architecture ou de haut niveau
(*Architectural Synthesis or High-Level Synthesis*[§])
tous les problèmes sont non polynomiaux, ...

[‡]. Premières tentatives autour de 1980

[§]. High-Level Synthesis : from Algorithm to Digital Circuit, P. Coussy and A. Morawiec, editors, Springer, 2008