Une école de l'IMT

Simulation évenmentielle

Principes

Tarik Graba
tarik.graba@telecom-paristech.fr
Année scolaire 2018/2019

Plan

La simulation événementielle

Simuler le comportement du matériel

Simuler efficacement une représentation RTL

- D'un côté, le matériel est intrinsèquement parallèle :
  - tous les composants d'un circuit sont actifs et fonctionnent en même temps.
- D'un autre côté,
  - les simulations sont exécutées sur des machines séquentielles (le CPU de votre PC) ;
  - il est plus simple pour nous de décrire des séquences.

Simuler le parallélisme

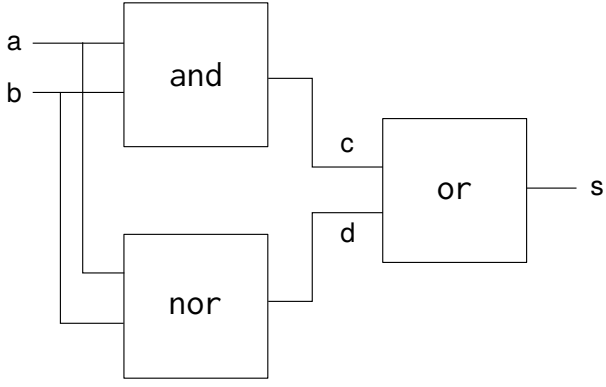
- Comment simuler efficacement du matériel ?
- On ne veut pas simuler ce qui se passe physiquement ! Uniquement le comportement (logique/arithmétique).

Simuler le parallélisme

Exemple : de la logique combinatoire

Initialement :

- a = 1
- b = 0
- c = x
- d = x
- s = x



Simuler le parallélisme

Pour de la logique combinatoire

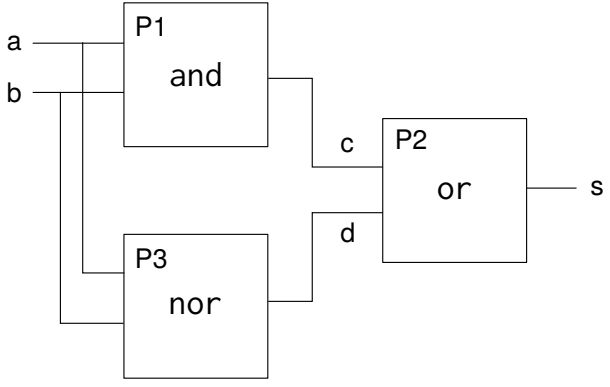
- Une fonction par bloc combinatoire
  - qu'on appellera **processus**
  - les instructions dans un processus sont exécutées séquentiellement
- On agit sur des variables globales vues par tous les processus
- On exécute les processus dans un ordre quelconque
- On re-exécuter tant qu'il y a des changements
- Ne fonctionne plus s'il y a une boucle

Simuler le parallélisme

Exemple : de la logique combinatoire

Initialement :

- a = 1
- b = 0
- c = x
- d = x
- s = x

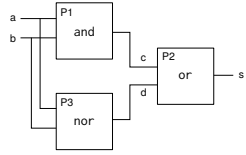


Simuler le parallélisme

Pour de la logique combinatoire

Initialement :

- a = 1
- b = 0
- c = x
- d = x
- s = x



		0				1				2				3
		P1	P2	P3		P1	P2	P3		P1	P2	P3		
c	x	0	–	–	0	0	–	–	0	0	–	–	0	
d	x	–	–	0	0	–	–	0	0	–	–	0	0	
s	x	–	x	–	x	–	0	–	0	–	0	–	0	

Simuler le parallélisme

Le temps symbolique

- Chaque cycle d'exécution du simulateur est appelé **temps symbolique** (ou delta Δ).
- Ça ne représene pas un temps "physique' !
- Tant qu'un processus a besoin d'être exécuté, on est dans le même delta.

Simuler le parallélisme

Le temps symbolique

					fin				fin				fin
	0ns	0ns			0ns	0ns			0ns	0ns			0ns
	init	Δ_1	$\rightarrow$		Δ_1	Δ_2	$\rightarrow$		Δ_2	Δ_3	$\rightarrow$		Δ_3
		P1	P2	P3		P1	P2	P3		P1	P2	P3	
c	x	0	-	-	0	0	-	-	0	0	-	-	0
d	x	-	-	0	0	-	-	0	0	-	-	0	0
s	x	-	x	-	x	-	0	-	0	-	0	-	0

Simulation événementielle

évènements et liste de sensibilité

Pour accélérer la simulation, il ne faut exécuter que les processus dont les entrées ont changé.

On ajoute alors deux notions :

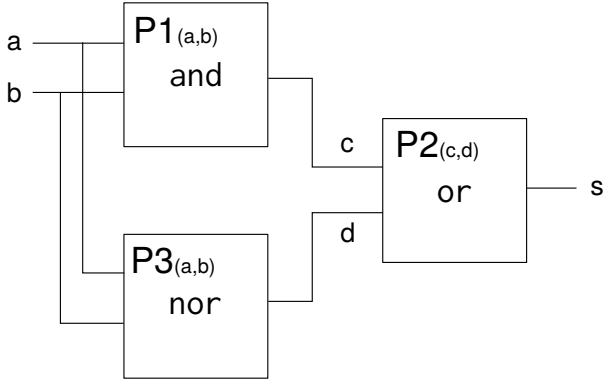
- Les évènements sur les entrées
  - Si une entrée change
- La liste de sensibilité
  - La liste des évènements qui nécessitent l'exécution d'un processus

Simulation événementielle

évènements et liste de sensibilité

Initialement :

- a = 1
- b = 0
- c = x
- d = x
- s = x

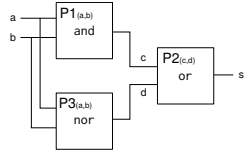


Simulation événementielle

évènements et liste de sensibilité

Initialement :

- a = 1
- b = 0
- c = x
- d = x
- s = x



				fin		fin
	0ns	0ns		0ns	0ns	0ns
	init	Δ_1	$\rightarrow$	Δ_1	Δ_2	Δ_2
		P1	P3		P2	
c	x	0	-	0	-	0
d	x	-	0	0	-	0
s	x	-	-	x	0	0

On a réduit le nombre d'itérations de simulation et de fonctions exécutées.

Simulation événementielle

La gestion du temps

Le simulateur maintient un compteur pour le **temps physique** indépendant du temps symboliques.
À un évènement sont attachés 2 informations de temps :

- Le temps symbolique Δ
- Le temps physique

Ceci permet d'ordonner les événements dans **l'échéancier** du simulateur ou de préciser le temps physique auquel il doit être pris en compte.

On parle de **notification**.

Simulation événementielle

Exemple

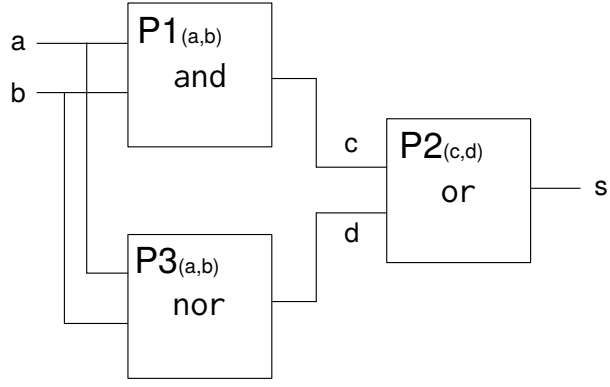
- a = x
- b = x
- c = x
- d = x
- s = x

@t=0ns

- a = 1
- b = 0

@t=10ns

- a = 1
- b = 1



Simulation événementielle

Exemple

- a = x
- b = x
- c = x
- d = x
- s = x

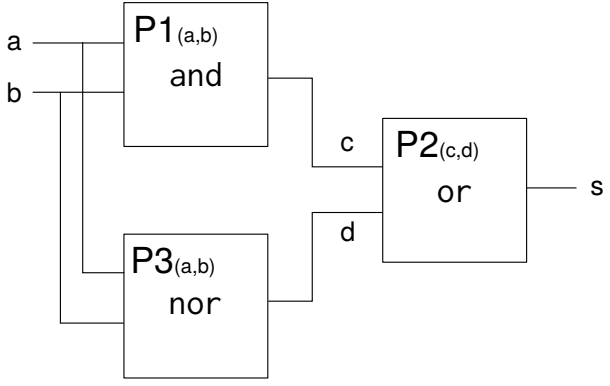
@t=0ns — P0

- a = 1
- b = 0

@t=10ns

- a = 1
- b = 1

@



Simulation événementielle

le temps physique

		fin			fin		fin		fin			fin		fin
		0ns			0ns	0ns	0ns	→	10ns	10ns		10ns	10ns	10ns
	init	Δ_0	Δ_1	→	Δ_1	Δ_2	Δ_2		Δ_0	Δ_1	→	Δ_1	Δ_2	Δ_2
		P0	P1	P3		P2			P0	P1	P3		P2	
a	x	1	–	–	1	–	1		1	–	–	1	–	1
b	x	0	–	–	0	–	0		1	–	–	1	–	1
c	x	x	0	–	0	–	0		–	1	–	1	–	1
d	x	x	–	0	0	–	0		–	–	0	0	–	0
s	x	x	–	–	x	0	0		–	–	–	0	1	1

Le temps physique avance quand il n'y a plus d'évènements déclenchant un processus.

Les processus sont en pratique des **boucles infinies**.

Ils sont en attente d'un événement :

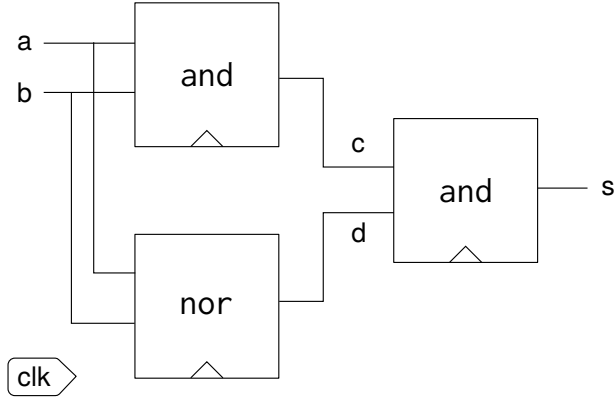
- Implicite due à la liste de sensibilité
- Explicite à l'aide d'instructions de synchronisation

Simulation événementielle

et logique séquentielle ?

Initialement :

- a = 1
- b = 0
- c = x
- d = x
- s = x



Simulation événementielle

et logique séquentielle ?

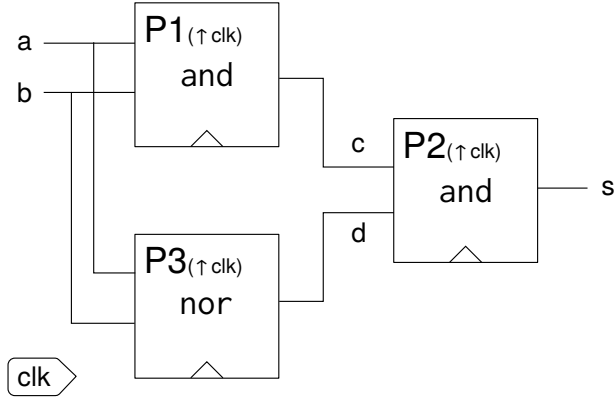
- Un évènement unique sur une entrée particulière, l'horloge.

Simulation événementielle

et logique séquentielle ?

Initialement :

- a = 1
- b = 0
- c = x
- d = x
- s = x

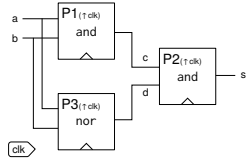


Simulation événementielle

et logique séquentielle ?

Initialement :

- a = 1
- b = 0
- c = x
- d = x
- s = x



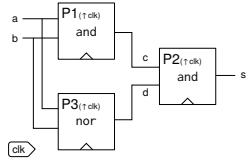
						fin						fin
		0ns	→	↑ clk			→	↑ clk				
	init	Δ_0		Δ_1	→	→	Δ_1		Δ_1	→	→	Δ_1
		P0		P1	P2	P3			P1	P2	P3	
a	x	1		-	-	-	-		-	-	-	-
b	x	0		-	-	-	-		-	-	-	-
c	x	x		0	-	-	0		0	-	-	0
d	x	x		-	-	0	0		-	-	0	0
s	x	x		-	0	-	0		-	0	-	0

Simulation événementielle

et logique séquentielle ?

Initialement :

- a = 1
- b = 0
- c = x
- d = x
- s = x



						fin						fin
		0ns	→	↑ clk			→	↑ clk				
	init	Δ_0		Δ_1	→	→	Δ_1	Δ_1	→	→	Δ_1	
		P0		P2	P3	P1		P2	P3	P1		
a	x	1		–	–	–	1	–	–	–	1	
b	x	0		–	–	–	0	–	–	–	0	
c	x	x		–	–	0	0	–	–	0	0	
d	x	x		–	0	–	0	–	0	–	0	
s	x	x		x	–	–	x	0	–	–	0	

Simulation événementielle

et logique séquentielle ?

Problème

- Tous les processus sont déclenchés en même temps.
- En modifiant instantanément les sorties, le résultat n'est pas tout le temps le même, il dépend de l'ordre d'exécution.

Simulation événementielle

Les signaux

- Un **signal** est une structure de données contenant :
  - la valeur **courante** avant le Δ
  - la valeur **future** qu'il aura après le Δ
- Une affectation sur un signal ne modifie que sa valeur future
- Lire un signal renvoie sa valeur courante
- À la fin du Δ la valeur du signal est mise à jour
  - ie. la valeur future remplace la valeur courante

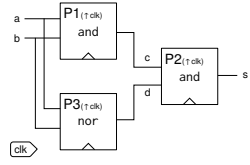
La valeur d'signal est maintenue tant que tous les processus actifs à un "Δ" n'ont pas été exécutés. On parle d'**affectations différées**.

Simulation événementielle

et logique séquentielle ?

Initialement :

- a = 1
- b = 0
- c = x
- d = x
- s = x



		m.à.j				m.à.j				m.à.j
	0ns	→	↑ clk			→	↑ clk			
init	Δ_0		Δ_1	→	→	Δ_1	Δ_1	→	→	Δ_1
	P0		P1	P2	P3		P1	P2	P3	
a (x,x)	(x,1)	(1,1)	-	-	-	-	-	-	-	-
b (x,x)	(x,0)	(0,0)	-	-	-	-	-	-	-	-
c (x,x)	(x,x)	(x,x)	(x,0)	-	-	(0,0)	(0,0)	-	-	(0,0)
d (x,x)	(x,x)	(x,x)	-	-	(x,0)	(0,0)	-	-	(0,0)	(0,0)
s (x,x)	(x,x)	(x,x)	-	(x,x)	-	(x,x)	-	(x,0)	-	(0,0)

Simulation événementielle

Les affectations en Verilog/SystemVerilog

En Verilog/SystemVerilog, il n'y pas de différence de déclaration entre une variable et un signal. C'est le symbole utilisé pour l'affectation qui permet de faire la différence :

- `a <= b` : affectation différée (donc signal)
- `a = b` : affectation immédiate (donc variable)

`a <= b` : affectation différée

Doivent être utilisées pour modéliser de la logique séquentielle de façon deterministe.

`a = b` : affectation immédiate

Peuvent être utilisées pour modéliser de la logique combinatoire.

Simuler le parallélisme

Résumons

- Décrire sous la forme d'un ensemble de fonctions : **les processus**
- La communication entre ces processus se faisant en modifiant des variables globales de type **signal** pour garantir le déterminisme.
- On définit la liste des **évènements** sur les **signaux** qui nécessitent de relancer des processus.

Simuler le parallélisme

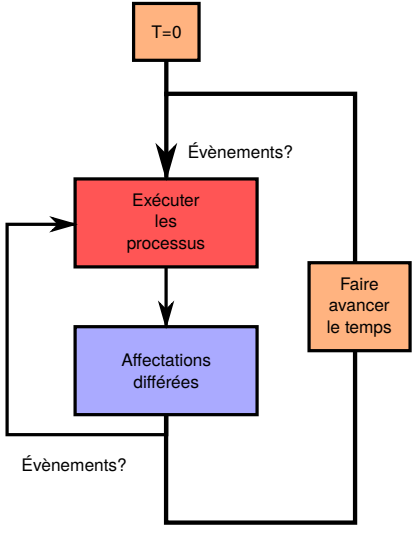
principes

1. Exécuter les processus dans un ordre arbitraire.
2. Si on demande à modifier la valeur d'un signal, mémoriser sa valeur **future**.
   Les variables sont modifiées immédiatement.
3. Quand tous les processus ont été exécutés (à la fin du Δ), modifier vraiment la valeur des signaux.
4. Refaire les étapes 1,2,3 si on a déclenché un nouvel évènement.
5. S'il n'y a aucun évenement, alors, faire avaner le temps physique

Simulation évènementielle

Algorithme

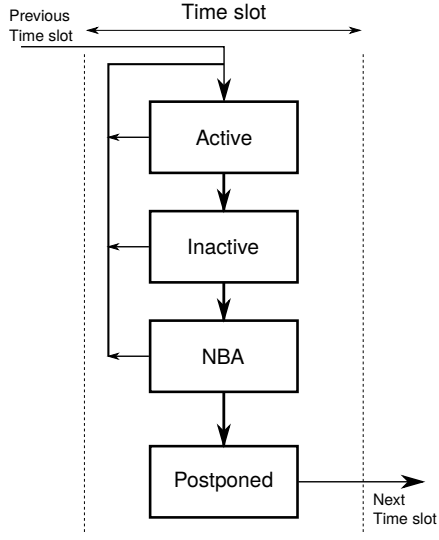
```
T = 0;
While( Event ) {
   While( Event at t=T ) {
      RunProcess
      ApplyDelayedAssignment
   }
   T = AdvanceToNextTime
}
End
```



Simulation évènementielle

En Verilog 2001

Pour un temps physique donné («*time slot*»), l'ordonnancement est divisé en plusieurs régions.

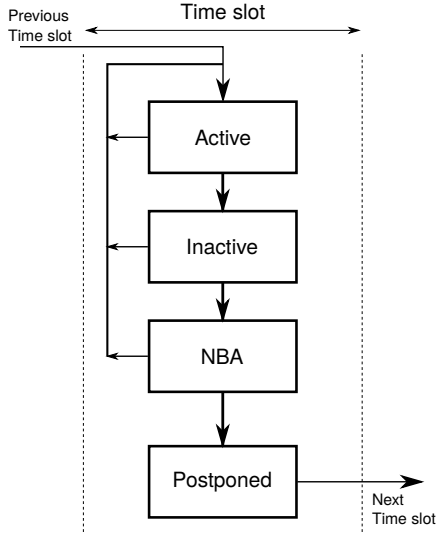


Simulation évènementielle

En Verilog 2001

Active

- les affectations immédiates des processus **`always`** ou **`initial`** ,
- les affectations concurrentes **`assign`** ,
- propagation des I/O (**`in`** , **`out`** ...)
- l'évaluation de ce qui se trouve à droite des affectations

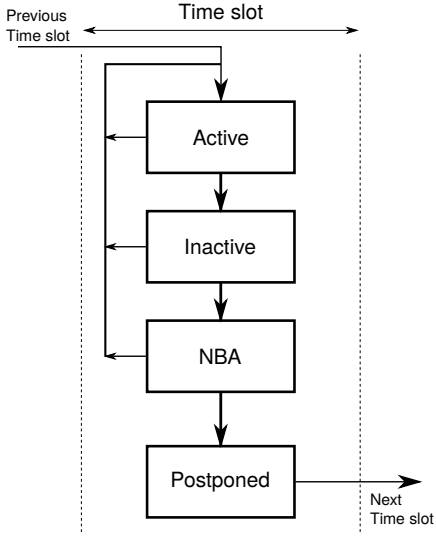


Simulation évènementielle

En Verilog 2001

Inactive

- les affectations à retard nul (`#0`)

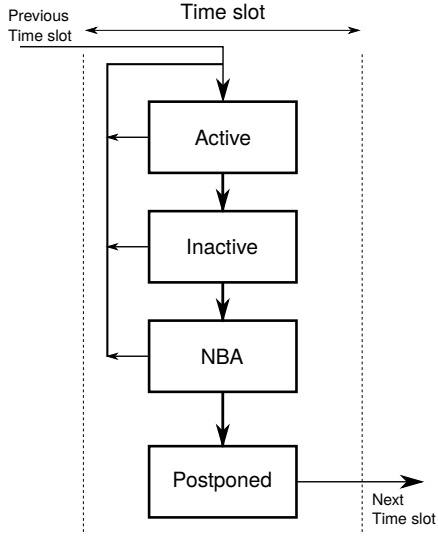


Simulation évènementielle

En Verilog 2001

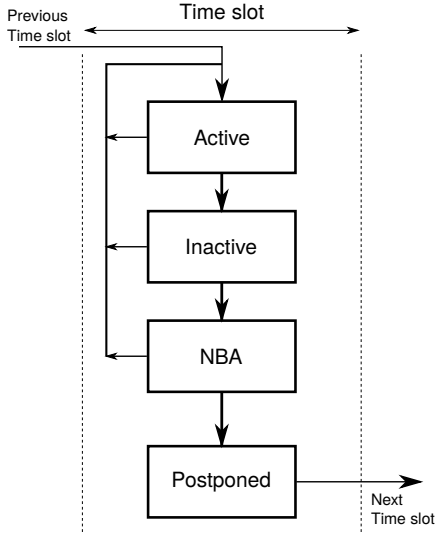
NBA : Non blocking assignment

- les affectations différées sont appliquées (<=)



Simulation évènementielle

En Verilog 2001



Postponed

- pour la fonction `$monitor`