Prototypage virtuel de système sur puce pour une simulation rapide et fidèle

Matthieu Moy

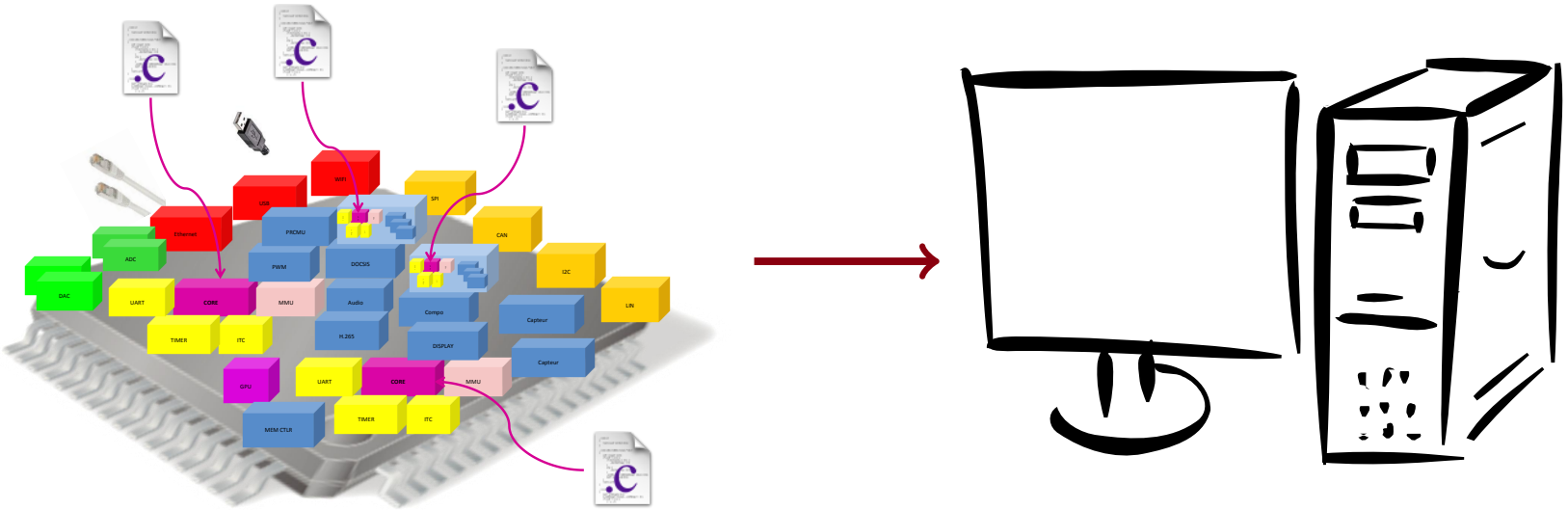
Verimag (Grenoble INP)
Grenoble, France

29 janvier 2014

Plan

Plan

Parallélisation des simulations



Problèmes et solutions pour l'exécution parallèle de SystemC/TLM

(1) Ordre d'exécution imposé par la sémantique de SystemC

(2) Accès concurrent aux ressources partagées
(comme `x++` sur une variable globale)

Problèmes et solutions pour l'exécution parallèle de SystemC/TLM

(1) Ordre d'exécution imposé par la sémantique de SystemC

(2) Accès concurrent aux ressources partagées
(comme `x++` sur une variable globale)

$\rightsquigarrow$ Pas de solution efficace 100% automatique pour TLM

Problèmes et solutions pour l'exécution parallèle de SystemC/TLM

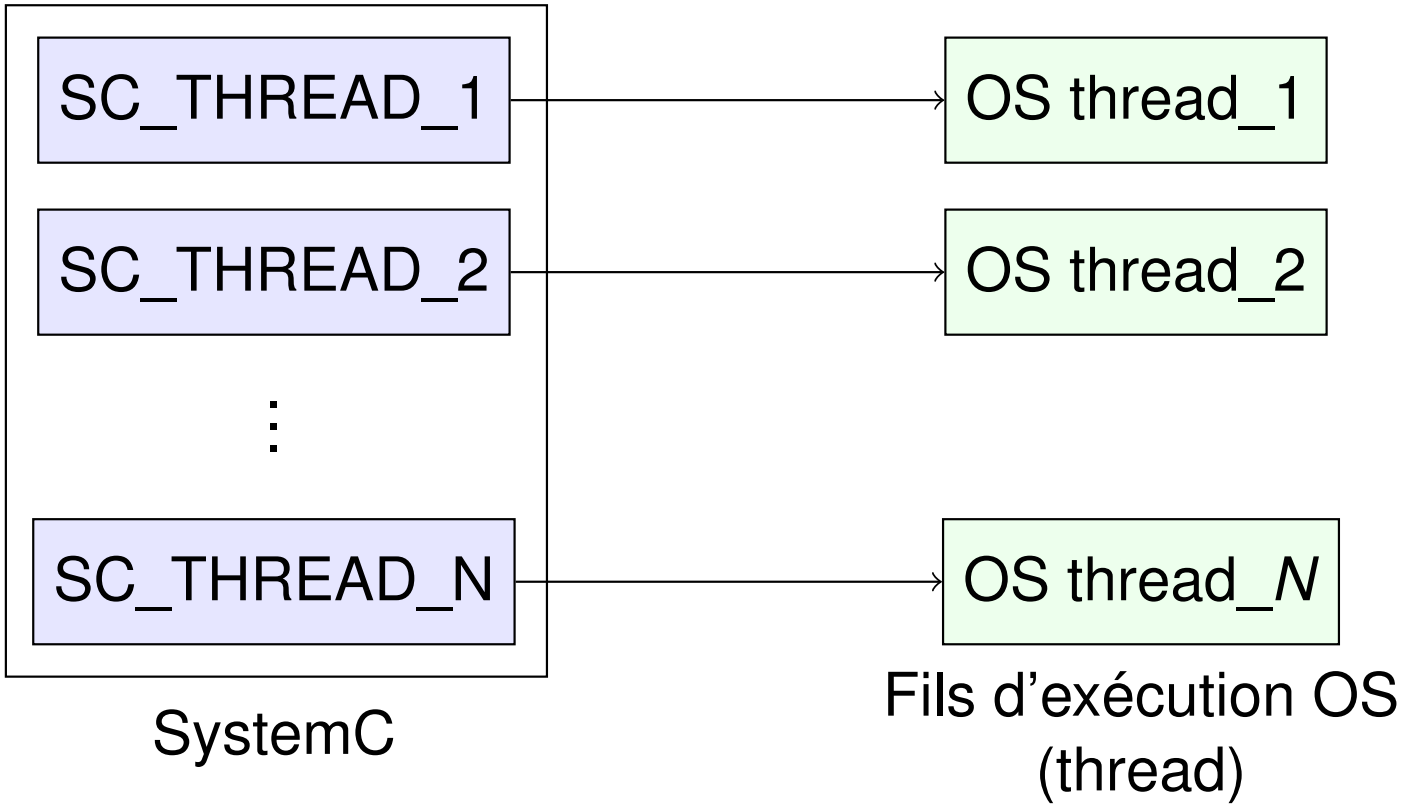
(1) Ordre d'exécution imposé par la sémantique de SystemC

(2) Accès concurrent aux ressources partagées
(comme `x++` sur une variable globale)

$\rightsquigarrow$ Pas de solution efficace 100% automatique pour TLM

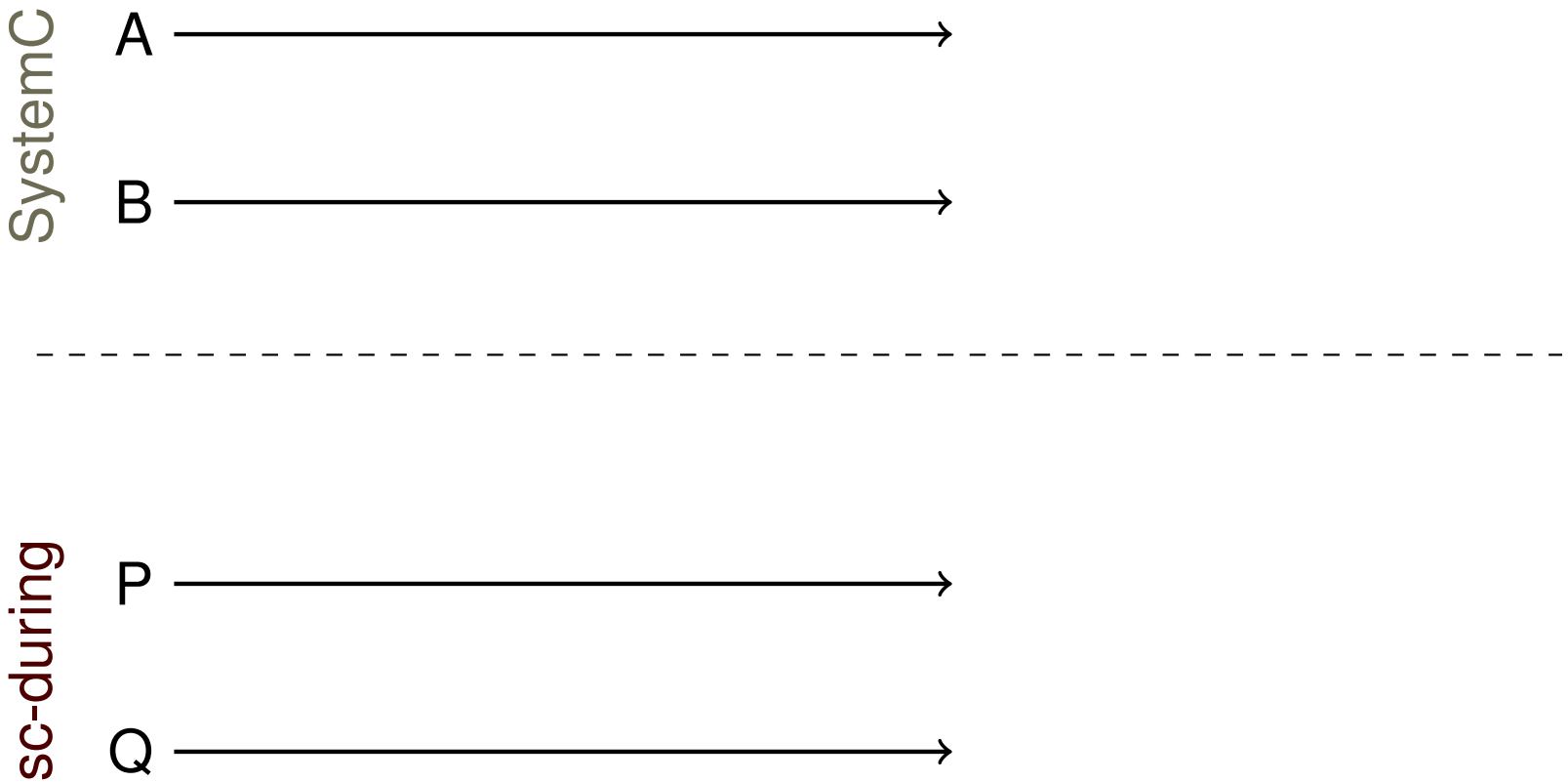
Notre proposition = des constructions en plus :
Désynchronisation (1) / Synchronisation (2)

SC-DURING: l'idée

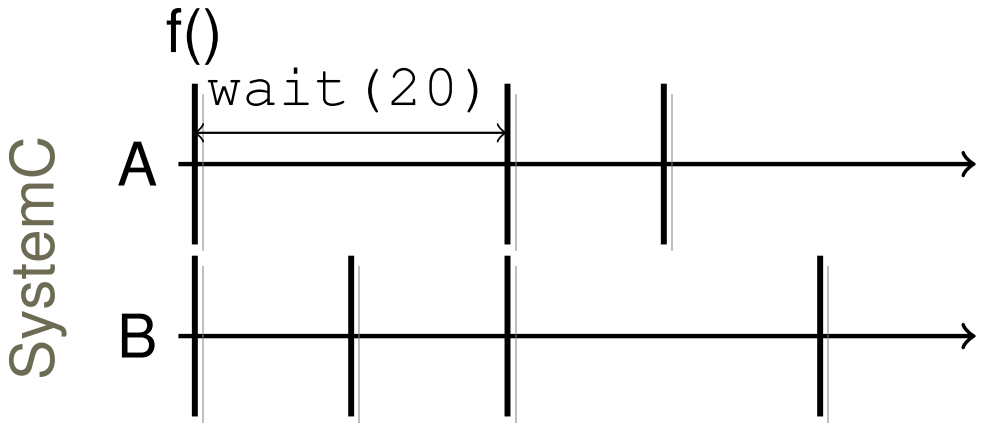


- SystemC non-modifié
- Certains calculs délégués à d'autres threads
- Synchronisation faible entre SystemC et les threads grâce aux tâches avec durée

Temps simulé en SystemC et sc-during

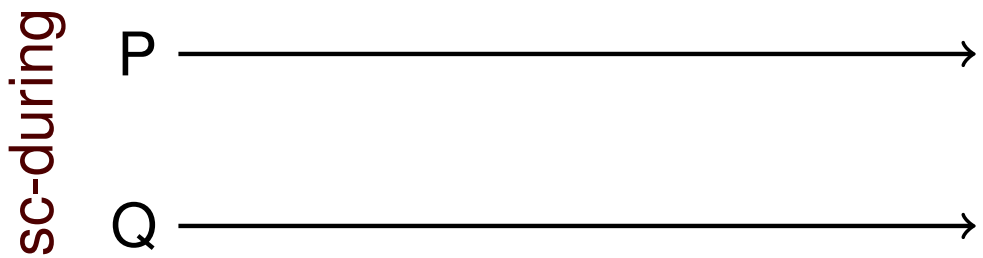


Temps simulé en SystemC et sc-during

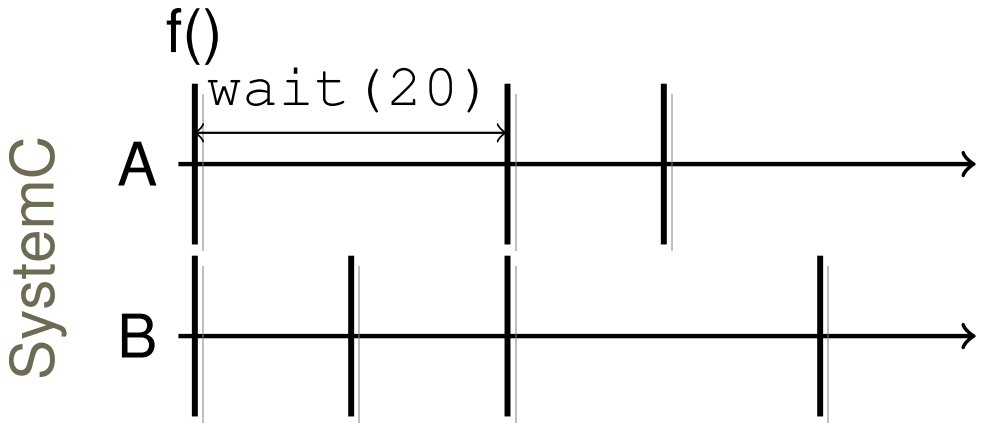


Processus A:

```
//Calcul
f();
//Temps pris par f
wait(20);
```

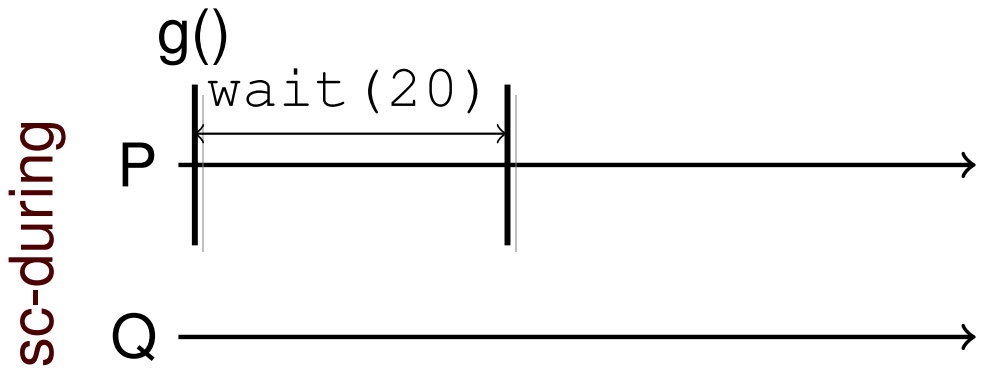


Temps simulé en SystemC et sc-during



Processus A:

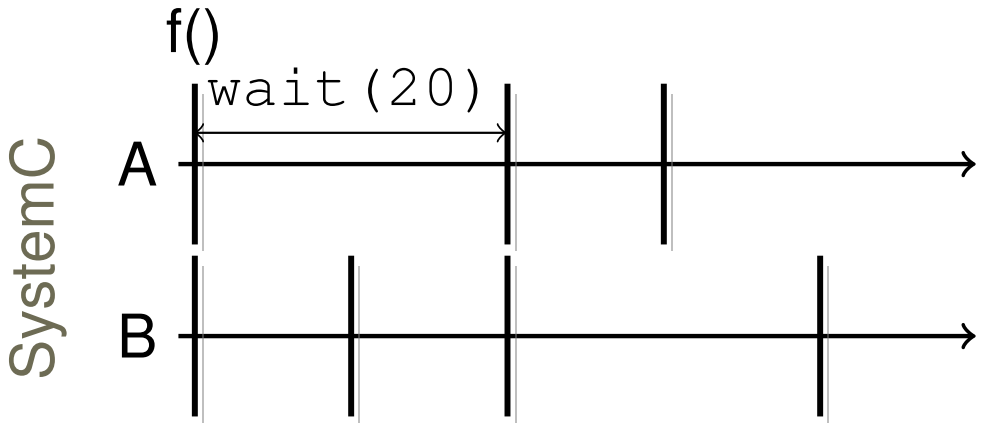
```
//Calcul
f();
//Temps pris par f
wait(20);
```



Processus P:

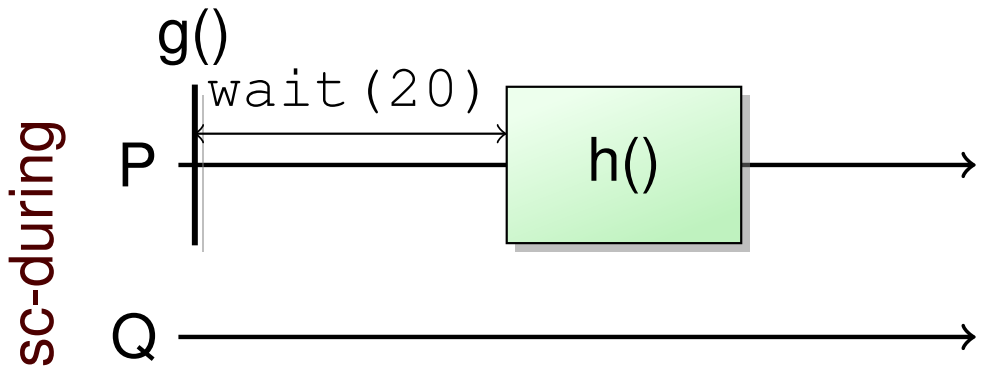
```
g();
wait(20);
```

Temps simulé en SystemC et sc-during



Processus A:

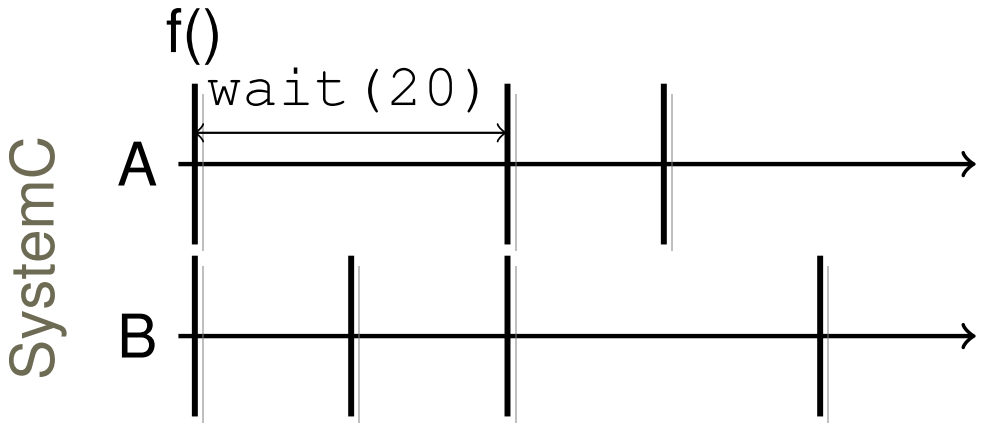
```
//Calcul
f();
//Temps pris par f
wait(20);
```



Processus P:

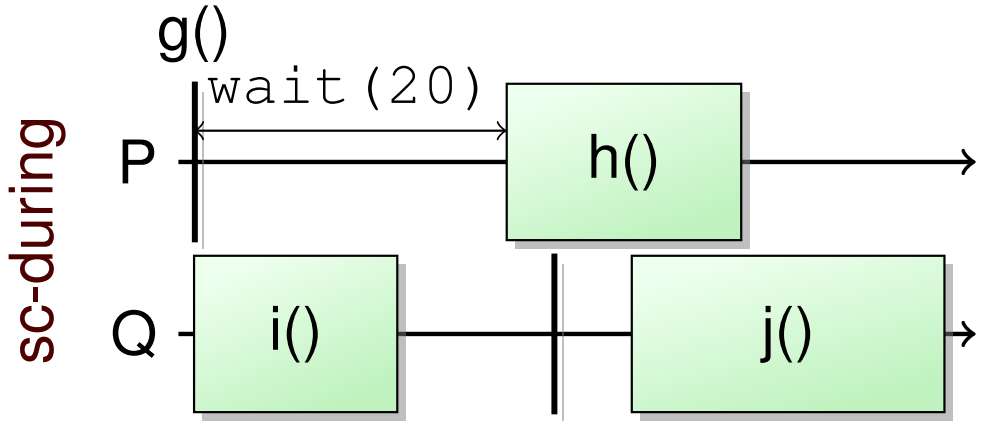
```
g();
wait(20);
during(15, h);
```

Temps simulé en SystemC et sc-during



Processus A:

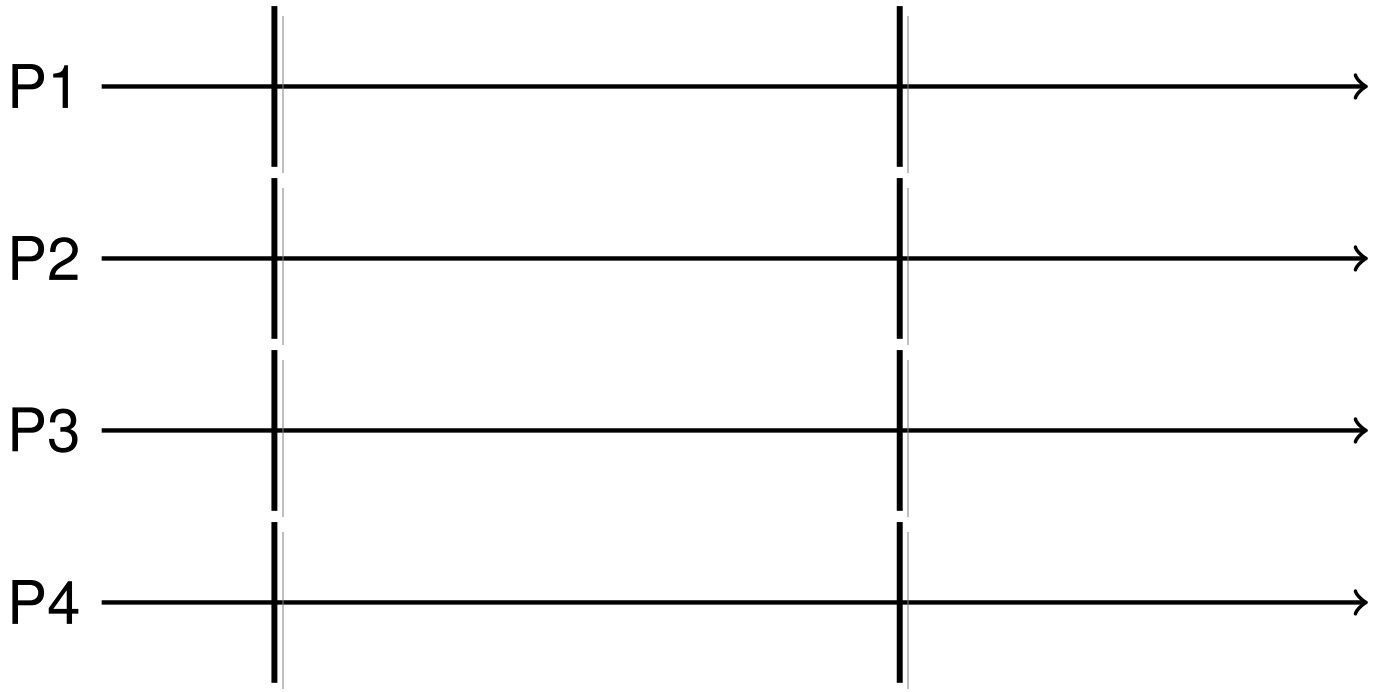
```
//Calcul
f();
//Temps pris par f
wait(20);
```



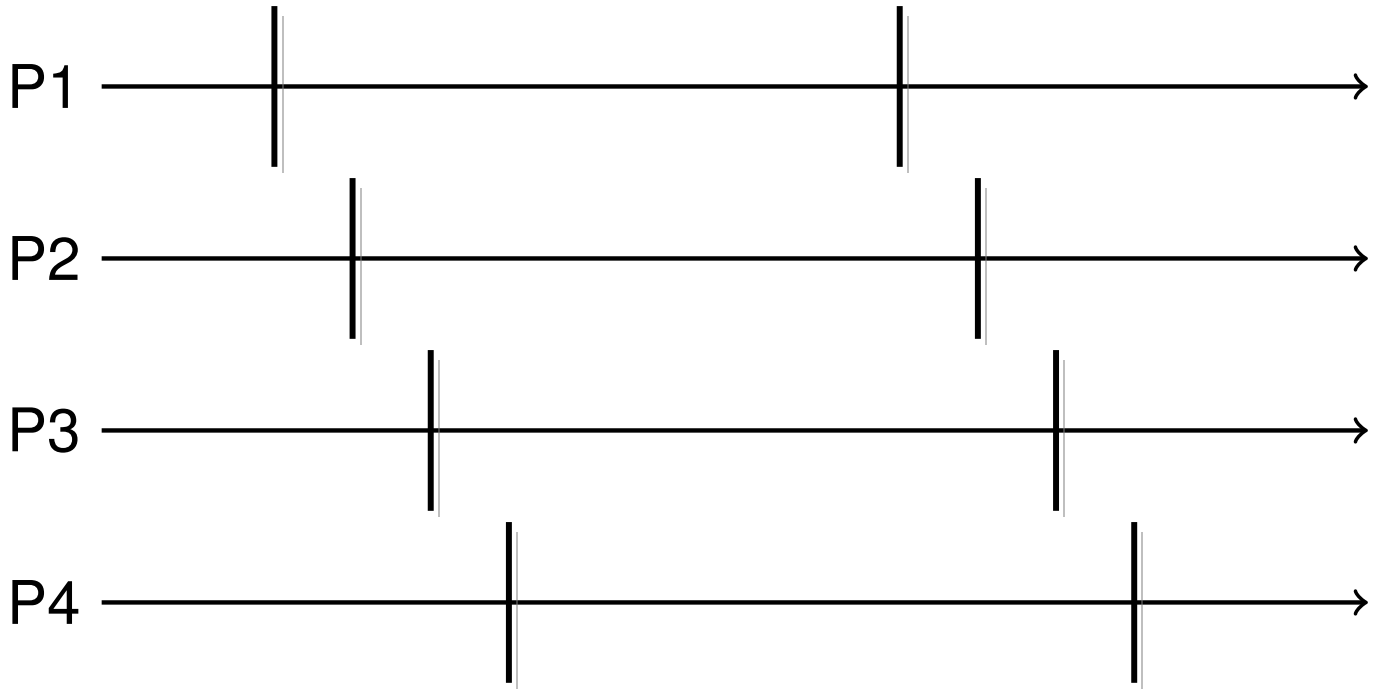
Processus P:

```
g();
wait(20);
during(15, h);
```

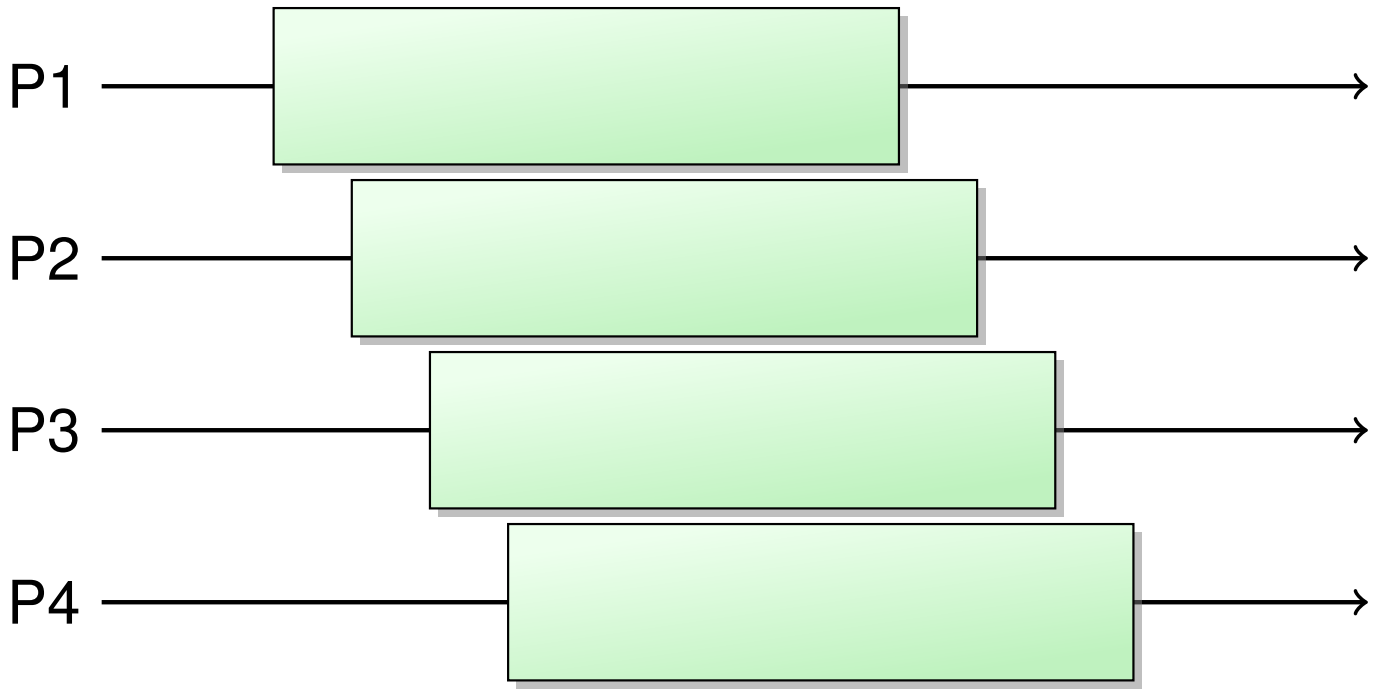
Impact sur le parallélisme



Impact sur le parallélisme



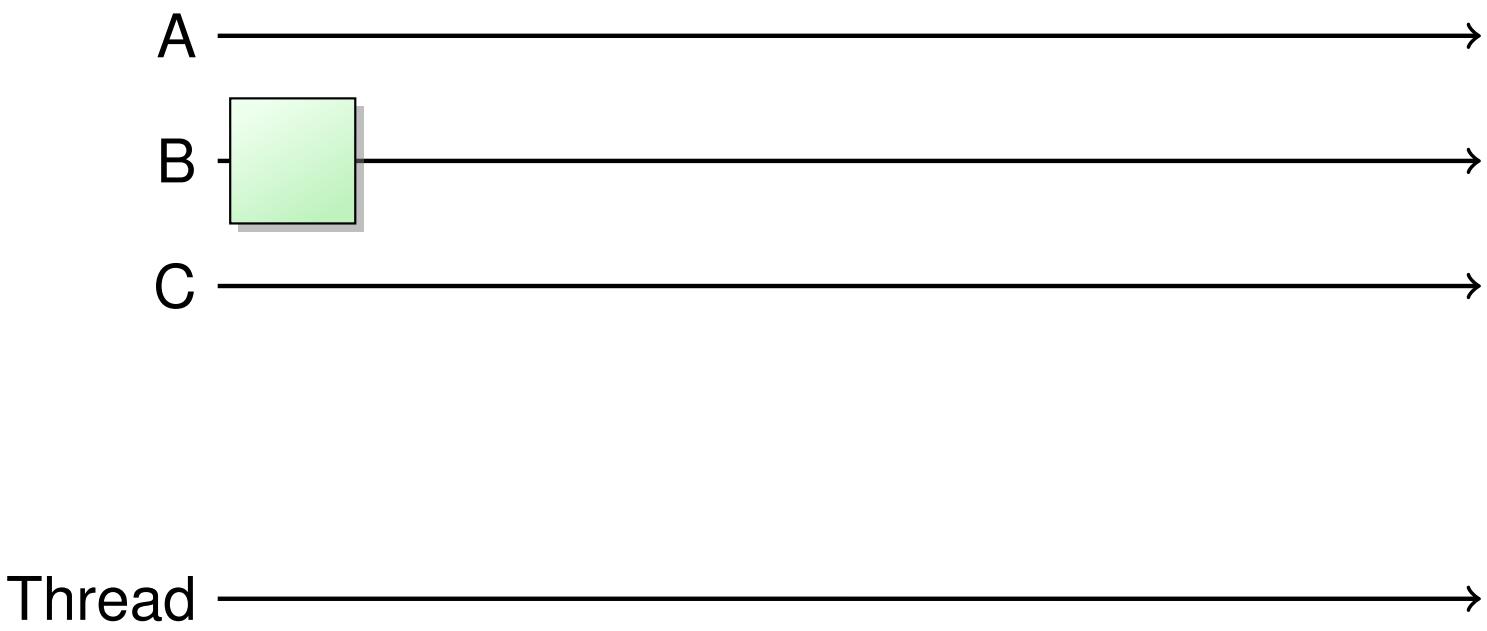
Impact sur le parallélisme



Recouvrement entre tâches $\rightsquigarrow$ exécution parallèle

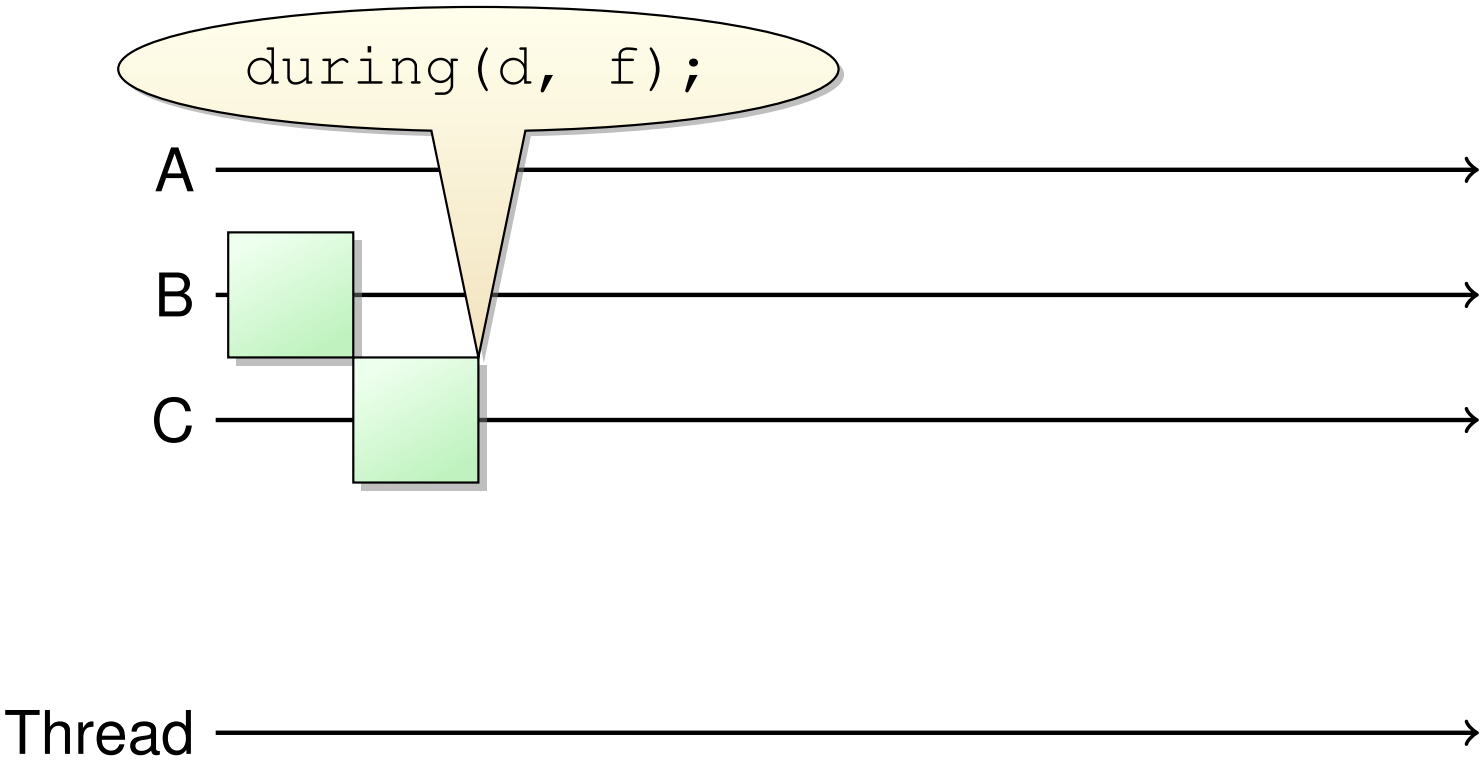
SC-DURING: première implémentation

```
void during(sc_core::sc_time d,
            std::function<void()> f) {
1     std::thread t(f); // Création du thread
2     sc_core::wait(d); // SystemC s'exécute
3     t.join(); // Attente de terminaison
}
```



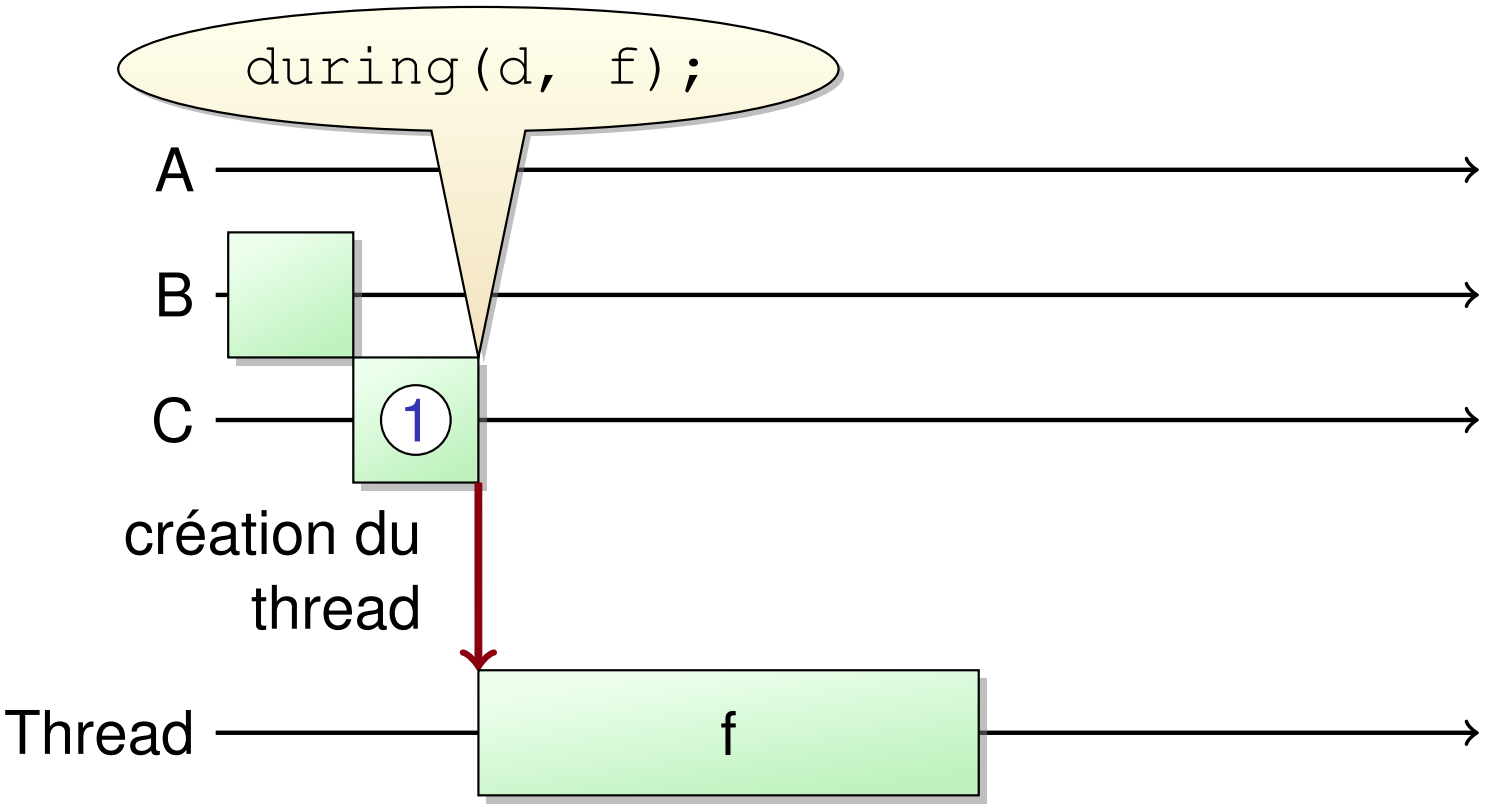
SC-DURING: première implémentation

```
void during(sc_core::sc_time d,
            std::function<void()> f) {
1     std::thread t(f); // Création du thread
2     sc_core::wait(d); // SystemC s'exécute
3     t.join(); // Attente de terminaison
}
```



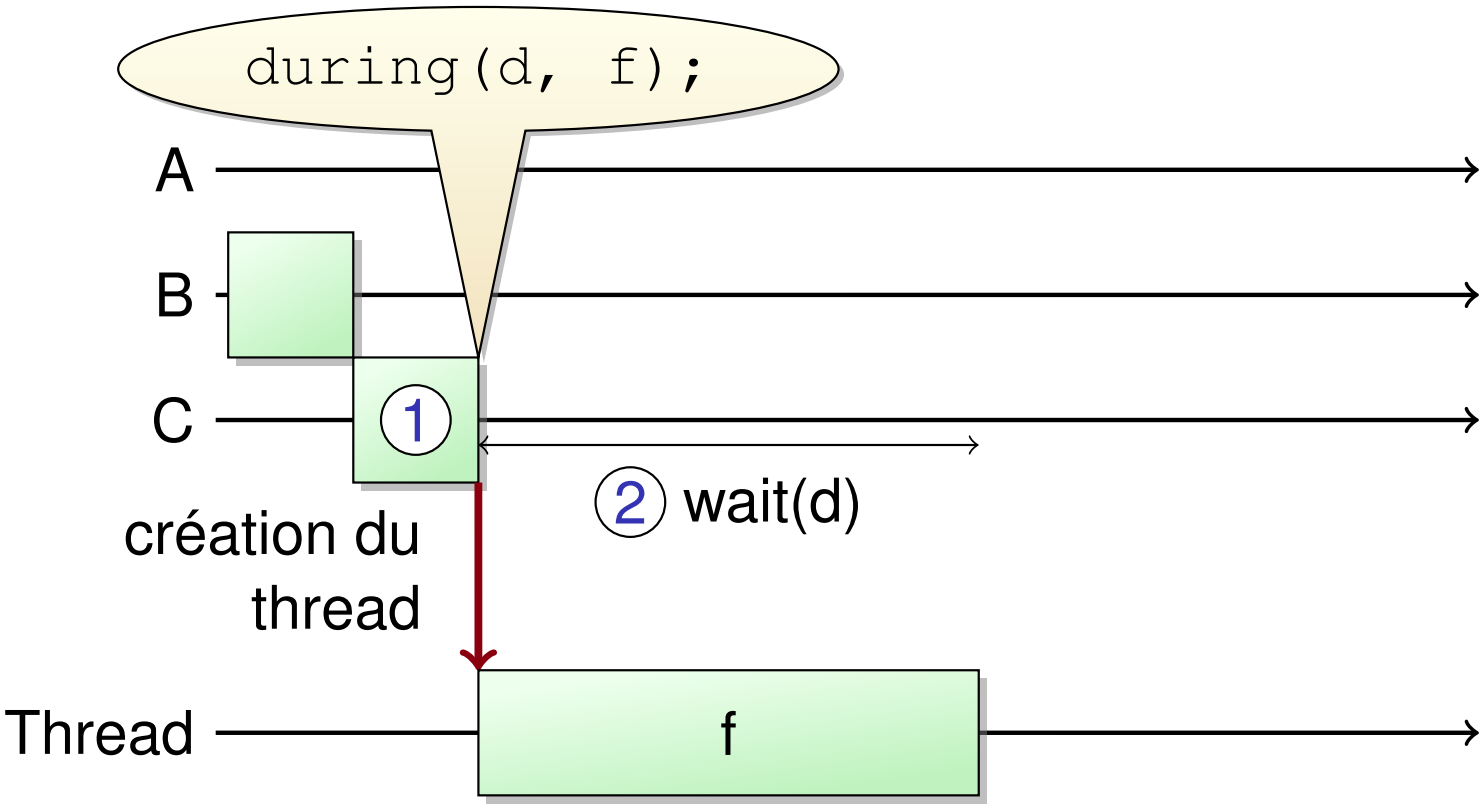
SC-DURING: première implémentation

```
void during(sc_core::sc_time d,
            std::function<void()> f) {
  1  std::thread t(f); // Création du thread
  2  sc_core::wait(d); // SystemC s'exécute
  3  t.join(); // Attente de terminaison
}
```



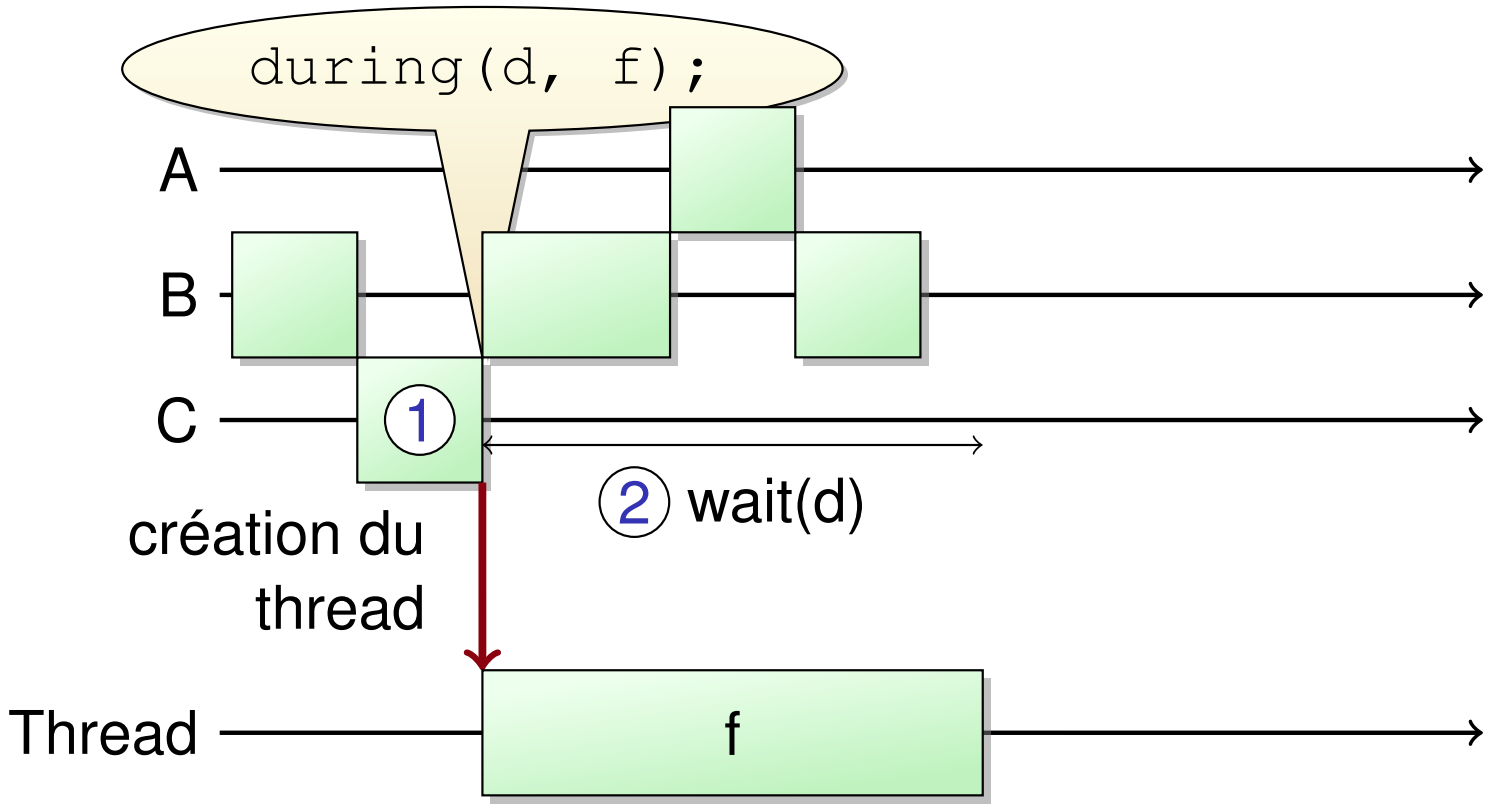
SC-DURING: première implémentation

```
void during(sc_core::sc_time d,
            std::function<void()> f) {
  1  std::thread t(f); // Création du thread
  2  sc_core::wait(d); // SystemC s'exécute
  3  t.join(); // Attente de terminaison
}
```



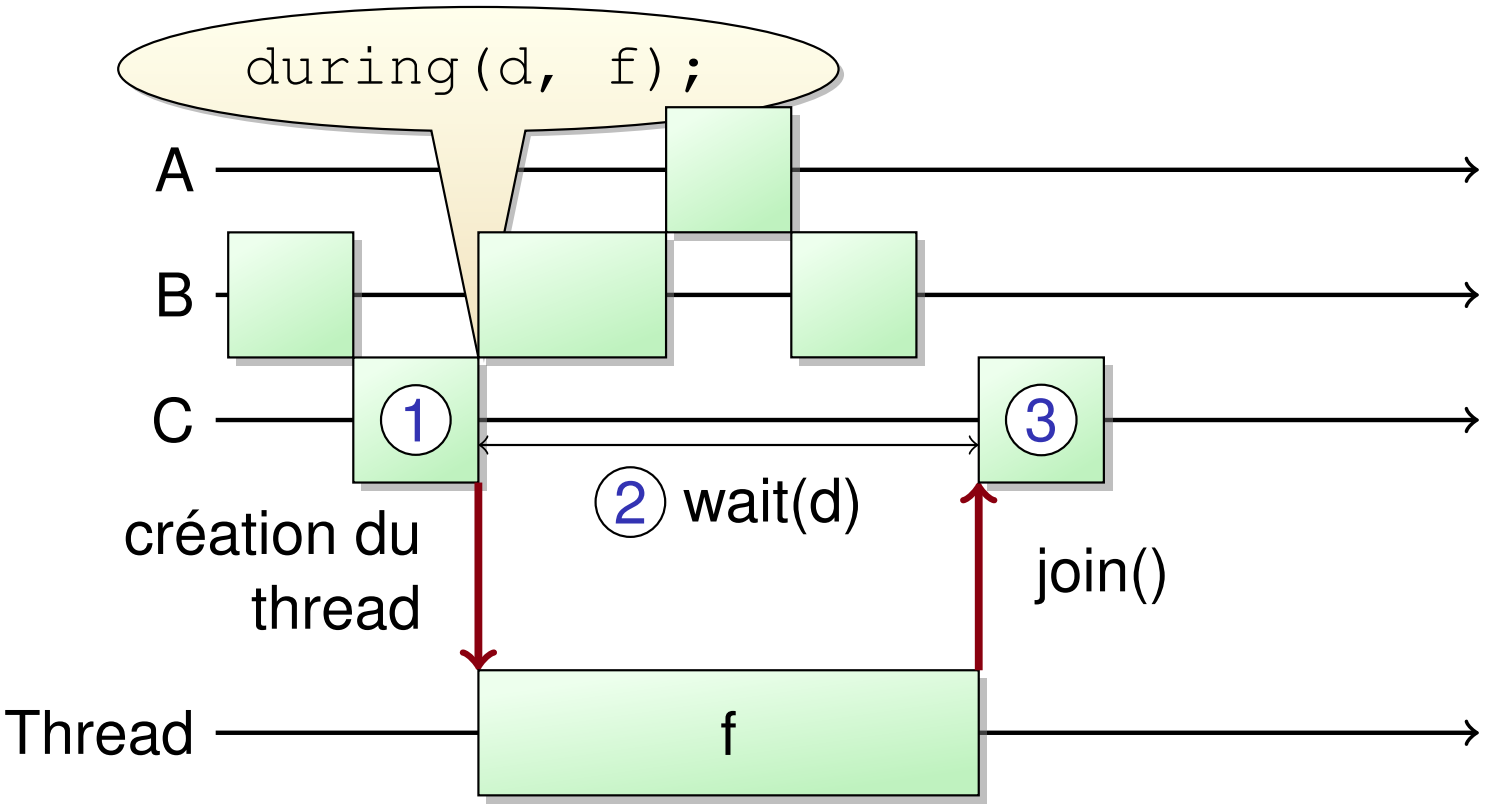
SC-DURING: première implémentation

```
void during(sc_core::sc_time d,
            std::function<void()> f) {
  1  std::thread t(f); // Création du thread
  2  sc_core::wait(d); // SystemC s'exécute
  3  t.join(); // Attente de terminaison
}
```



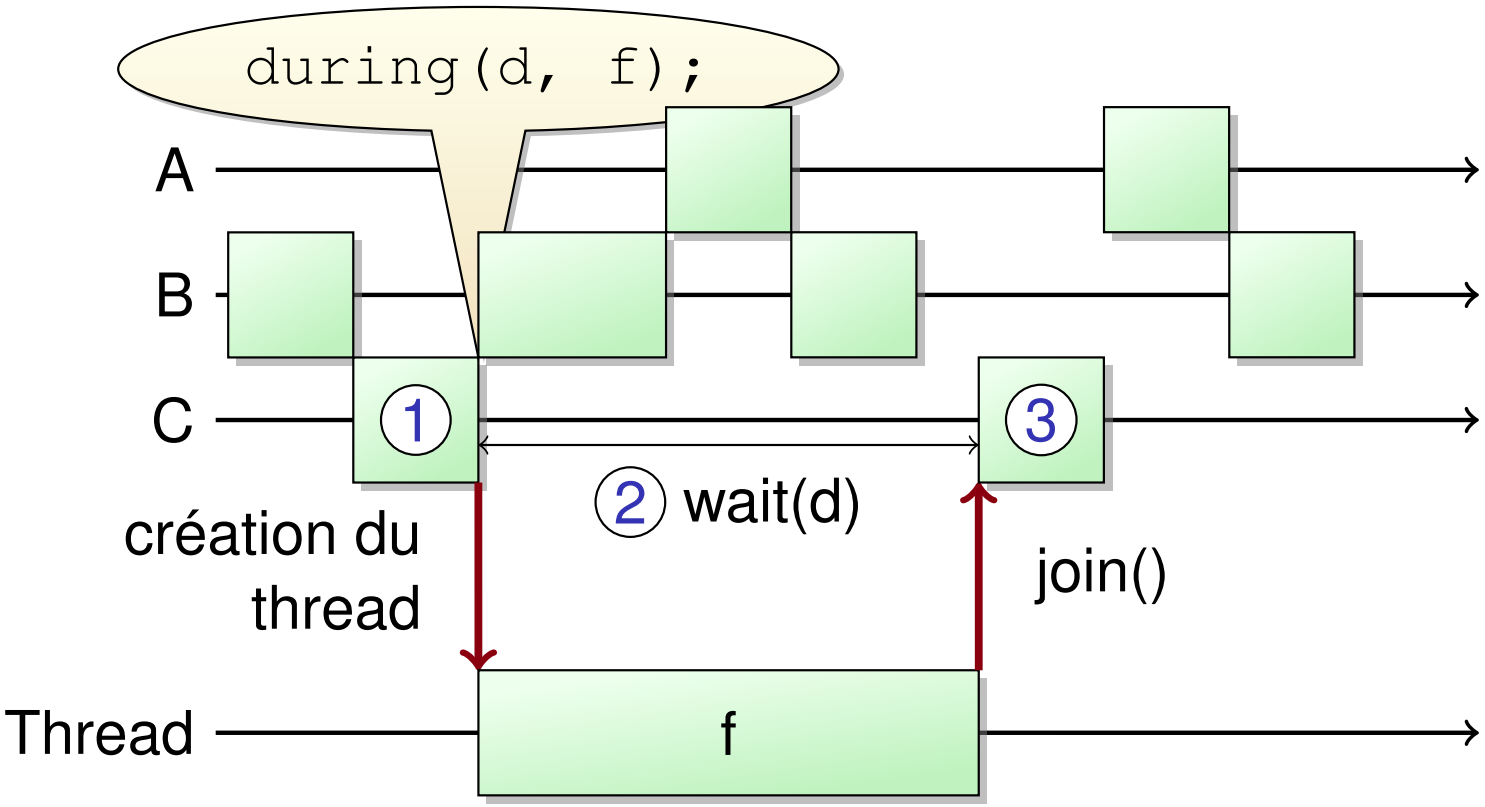
SC-DURING: première implémentation

```
void during(sc_core::sc_time d,
            std::function<void()> f) {
  1  std::thread t(f); // Création du thread
  2  sc_core::wait(d); // SystemC s'exécute
  3  t.join(); // Attente de terminaison
}
```



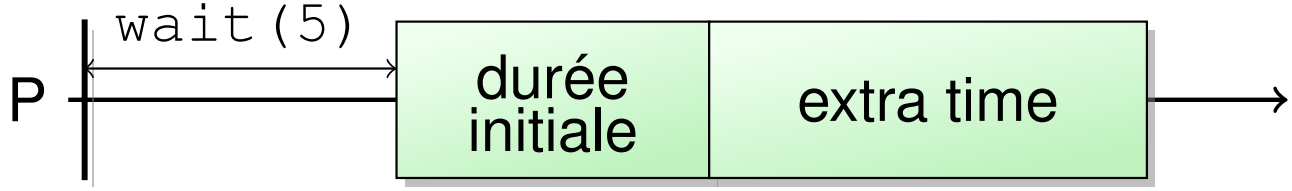
SC-DURING: première implémentation

```
void during(sc_core::sc_time d,
            std::function<void()> f) {
  1  std::thread t(f); // Création du thread
  2  sc_core::wait(d); // SystemC s'exécute
  3  t.join(); // Attente de terminaison
}
```



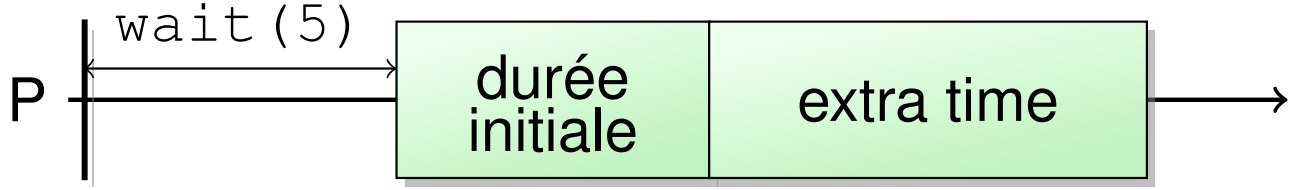
SC-DURING: nouvelles primitives de synchronisation

extra_time(t): Augmenter la durée de la tâche en cours



SC-DURING: nouvelles primitives de synchronisation

extra_time(t): Augmenter la durée de la tâche en cours

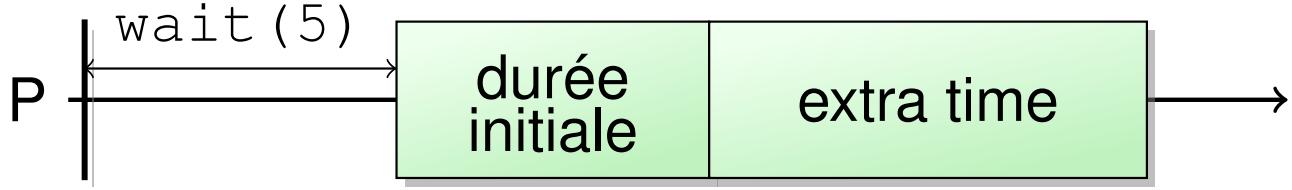


catch_up(): Attendre que SystemC ait atteint la fin de la tâche

```
while (!c) {
    extra_time(10);
    catch_up(); // Assure l'équité
}
```

SC-DURING: nouvelles primitives de synchronisation

extra_time(t): Augmenter la durée de la tâche en cours



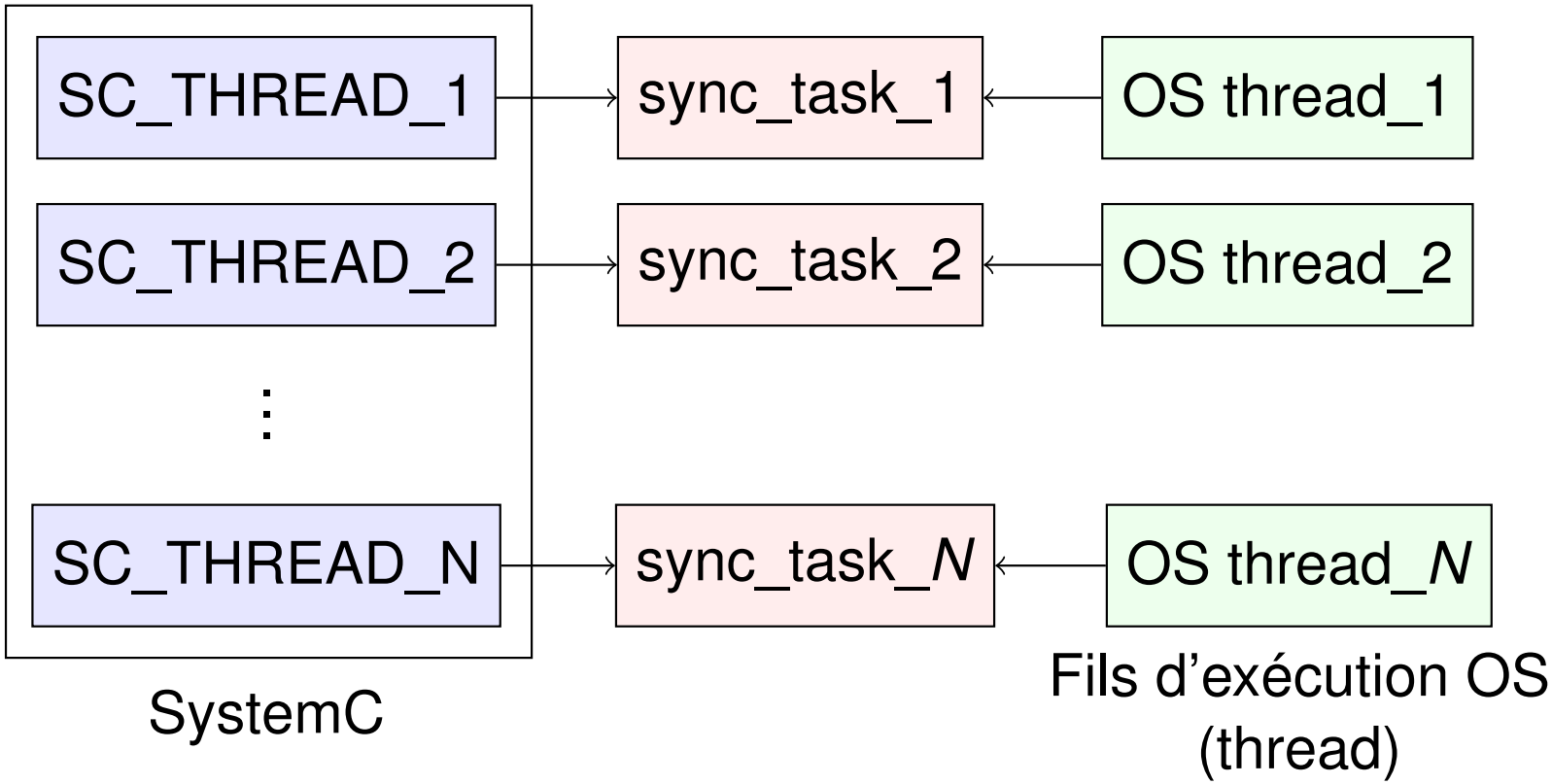
catch_up(): Attendre que SystemC ait atteint la fin de la tâche

```
while (!c) {
    extra_time(10);
    catch_up(); // Assure l'équité
}
```

sc_call(f): Appel de `f` dans le contexte de SystemC

```
x++; // Interdit dans une
     // tâche avec durée
sc_call([]{  x++;  }); // OK
```

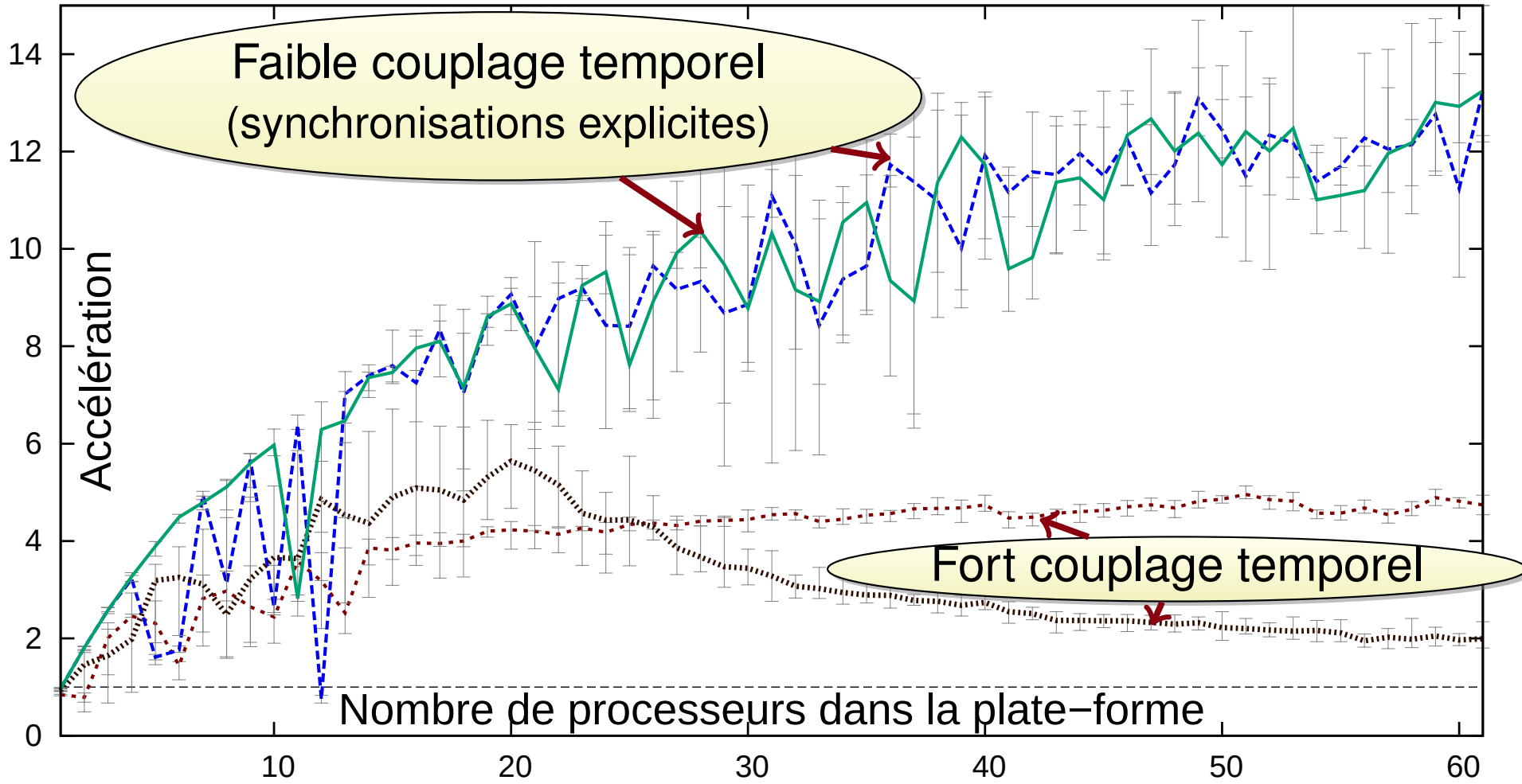
SC-DURING: implémentations



Stratégies:

SEQ	Séquentiel (= référence)
THREAD	Création + destruction de thread pour chaque tâche
POOL	Ensemble de threads pré-alloués
ONDEMAND	Thread créé sur demande et réutilisé

SC-DURING: résultats et conclusions



Machine de test : $4 \times 12 = 48$ cœurs

Plan

Estimation de consommation et de température

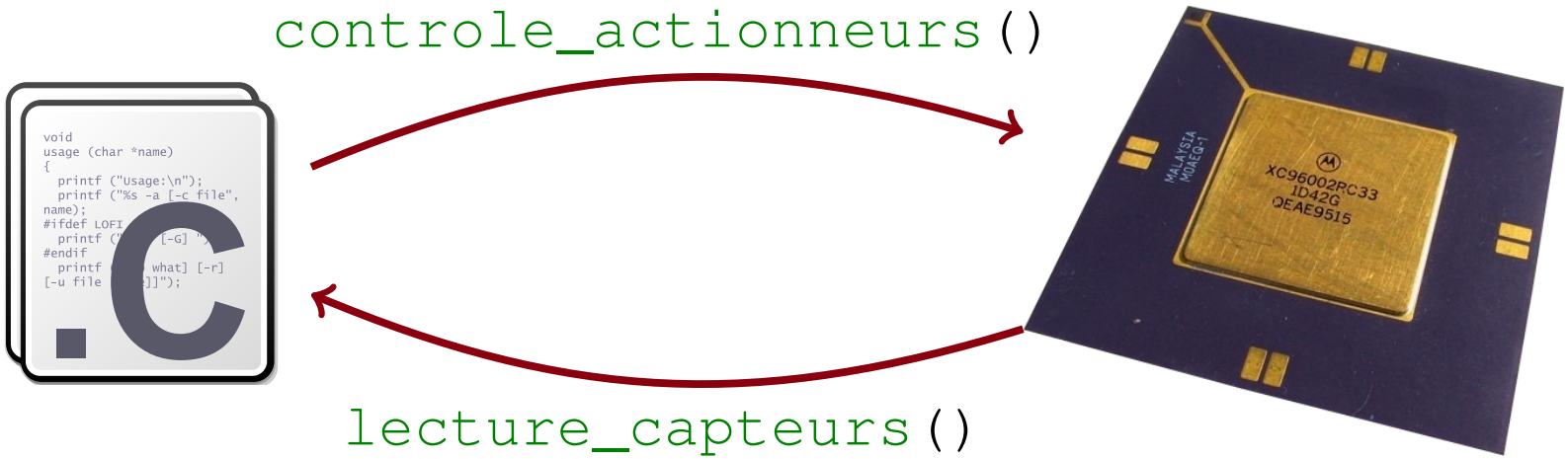
Un exemple

« Comment valider un logiciel embarqué qui régule la température de la puce ? »

```
while (true) {
    // Température d'un ou
    // plusieurs points de la puce
    lecture_capteurs();

    calcul_decision();

    // Réduction de fréquence/tension,
    // extinction d'urgence, ...
    controle_actionneurs();
}
```

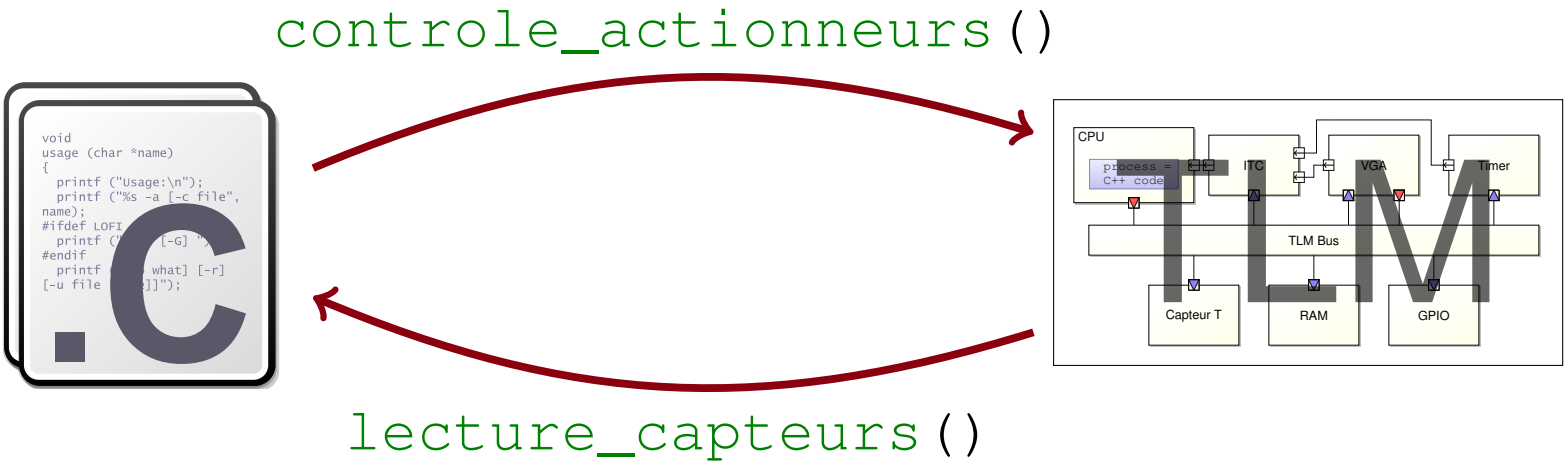
Estimation de consommation et de température

Quelle précision ? Quelles applications ?



Estimation de consommation et de température

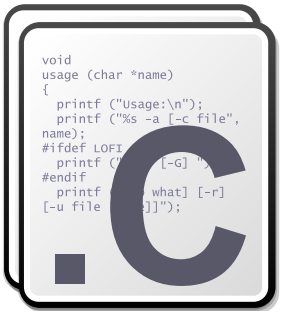
Quelle précision ? Quelles applications ?

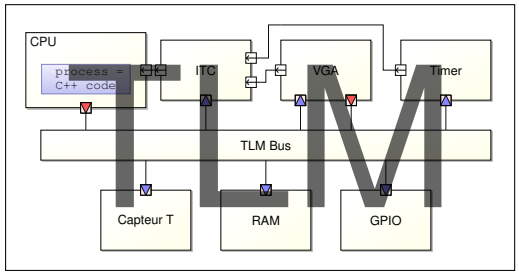


Estimation de consommation et de température

Quelle précision ? Quelles applications ?

`controle_actionneurs()`



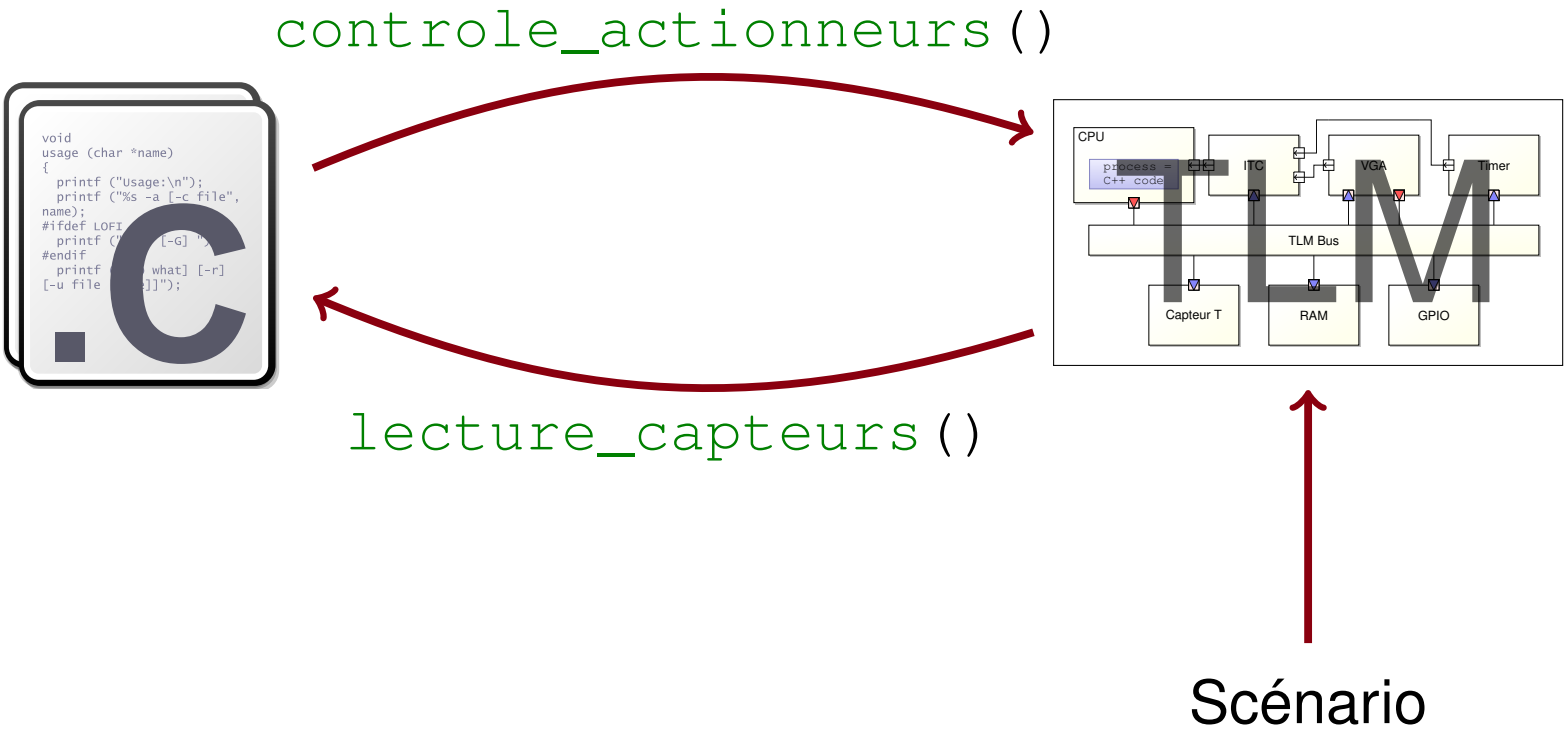


`lecture_capteurs()`

Température arbitraire

Estimation de consommation et de température

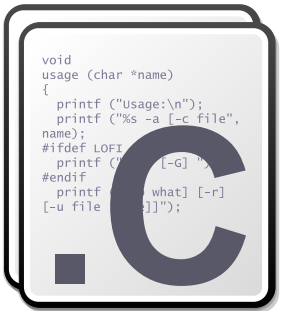
Quelle précision ? Quelles applications ?

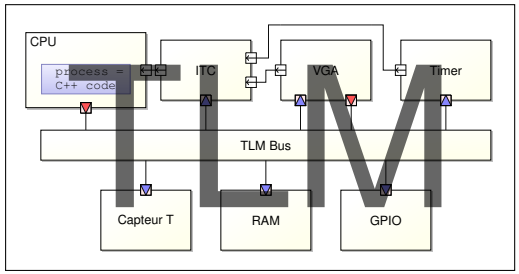


Estimation de consommation et de température

Quelle précision ? Quelles applications ?

`controle_actionneurs()`





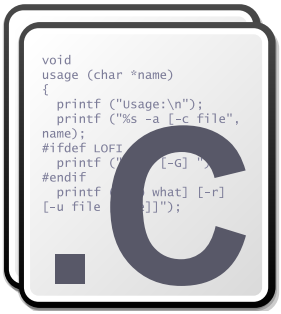
`lecture_capteurs()`

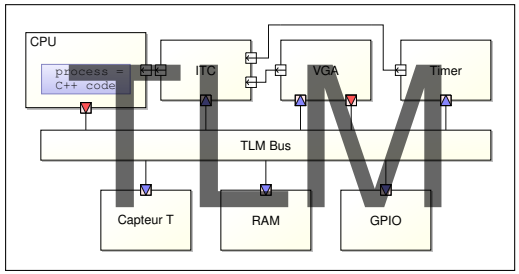
Calculs sur un modèle

Estimation de consommation et de température

Quelle précision ? Quelles applications ?

`controle_actionneurs()`

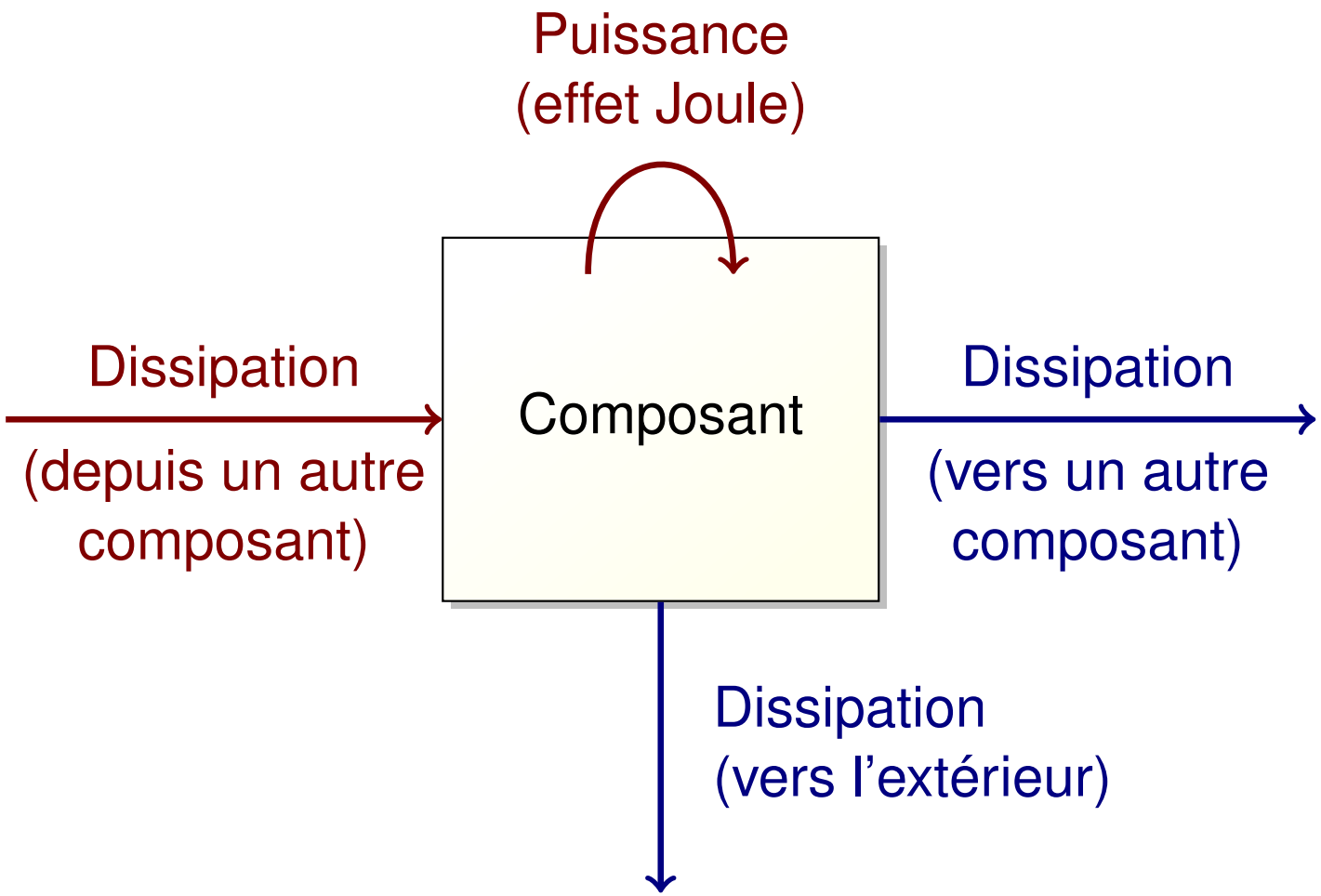




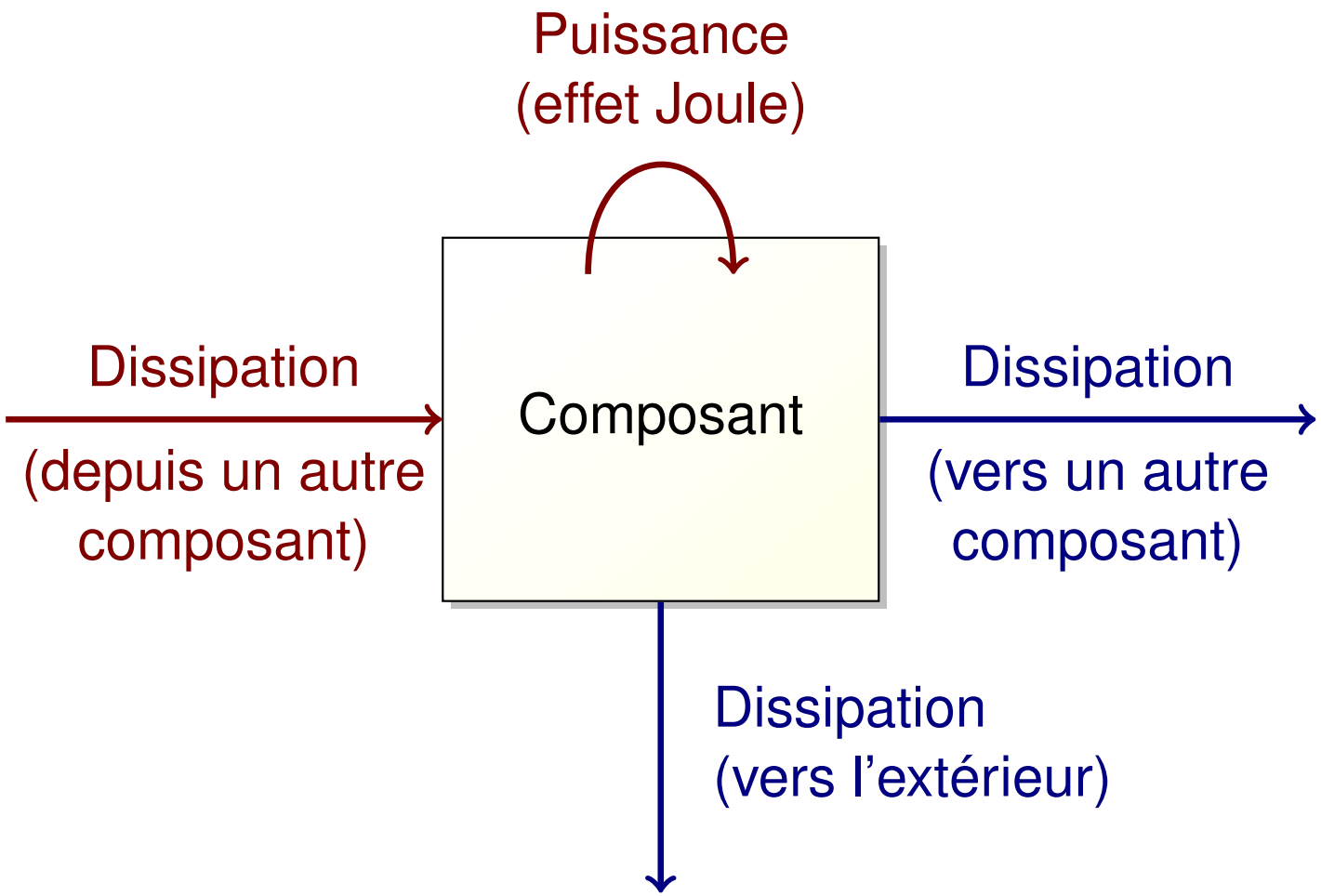
`lecture_capteurs()`

Calculs sur un modèle

Puissance consommée, température, et dissipation de chaleur

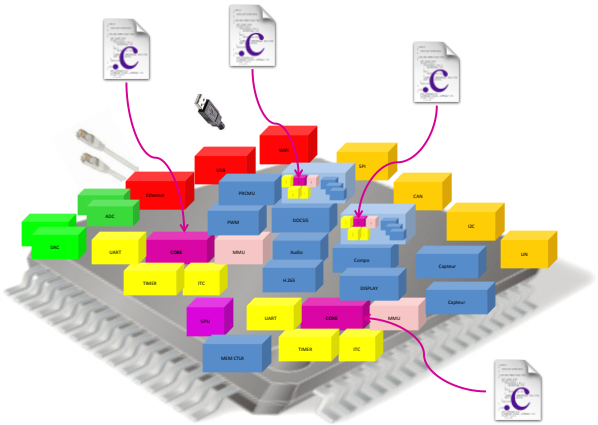


Puissance consommée, température, et dissipation de chaleur



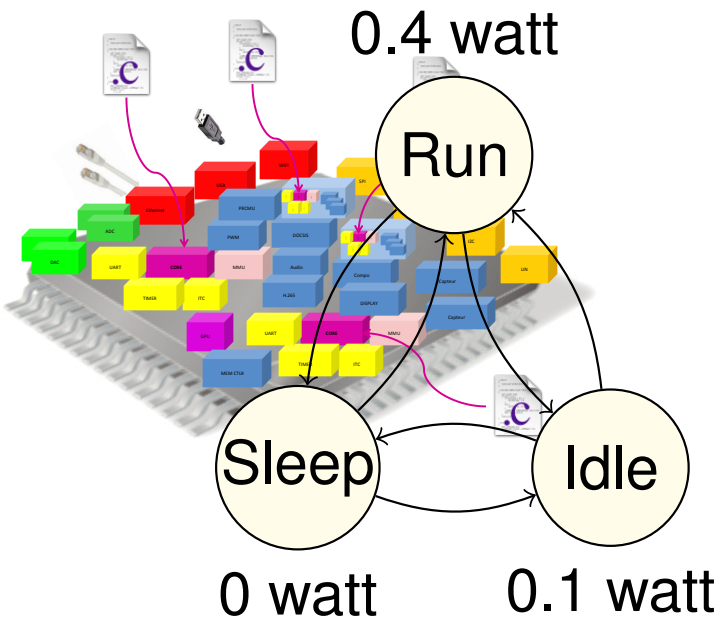
$\rightsquigarrow$ système d'équation différentielles, résolu par des solveurs dédiés

Estimation de consommation avec des modèles à état



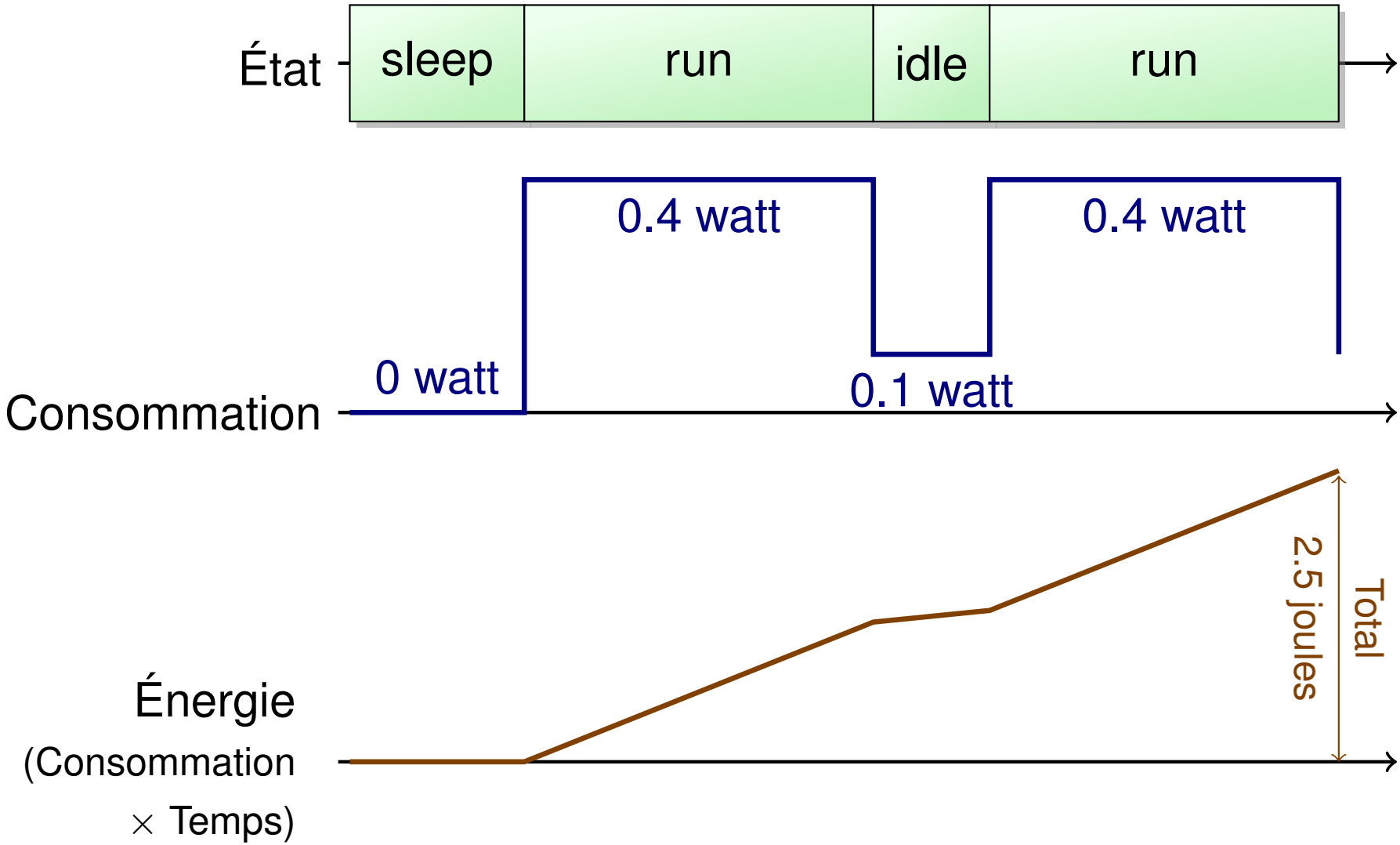
```
// Processus SystemC
void compute() {
    while (true) {

        f();
        wait(10);

        wait();
    }
}
```

Estimation de consommation avec des modèles à état

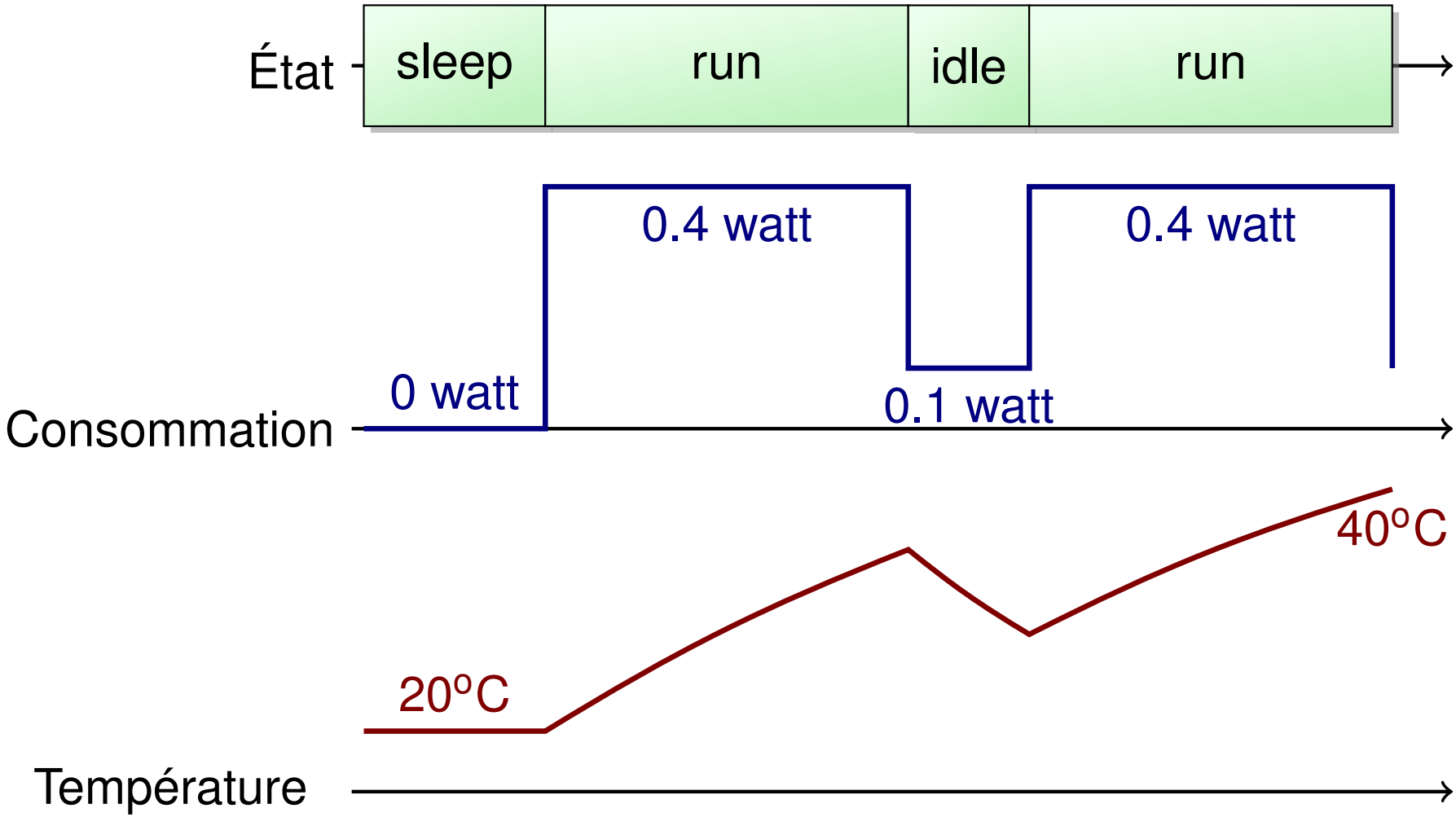



```
// Processus SystemC
void compute() {
    while (true) {
        set_state("run");
        f();
        wait(10);
        set_state("idle");
        wait();
    }
}
```

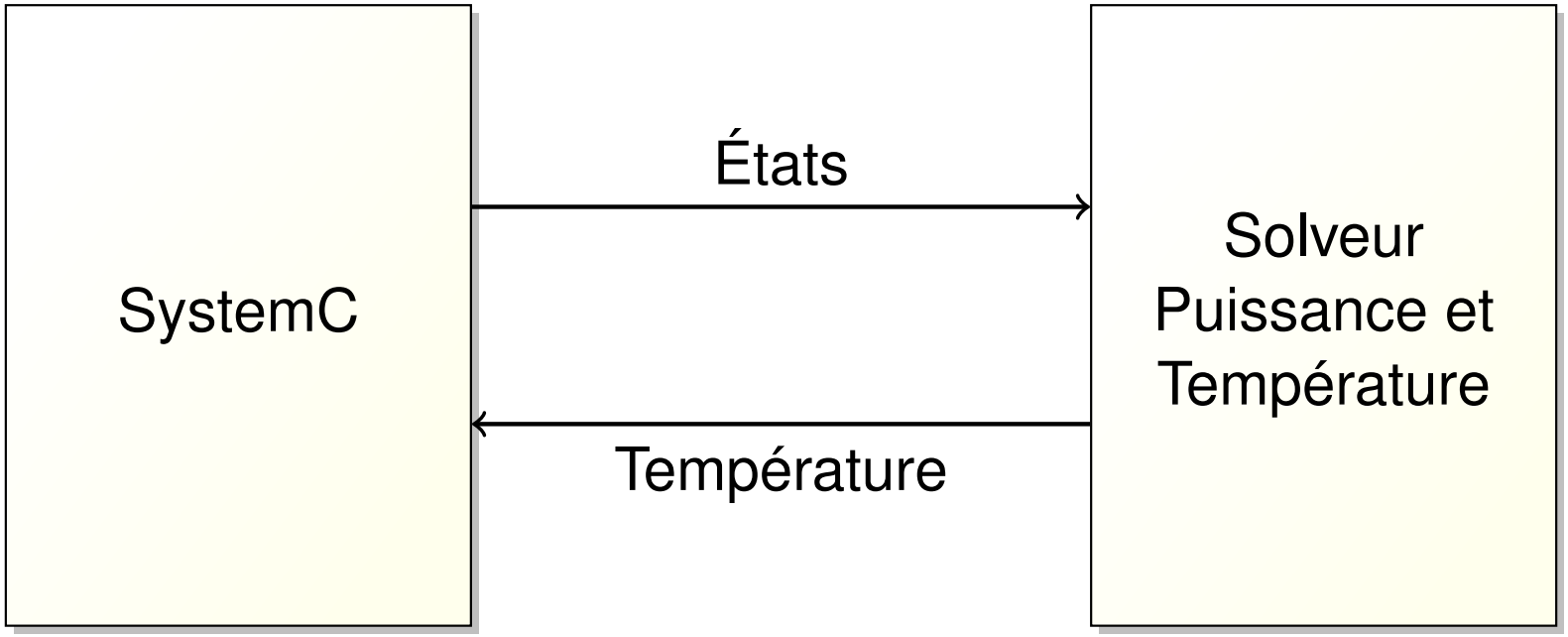
Des états à la consommation



De la consommation à la température

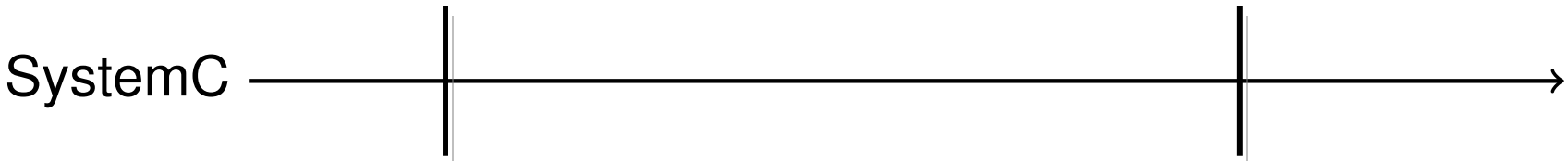


Cosimulation SystemC et solveur extra-fonctionnel

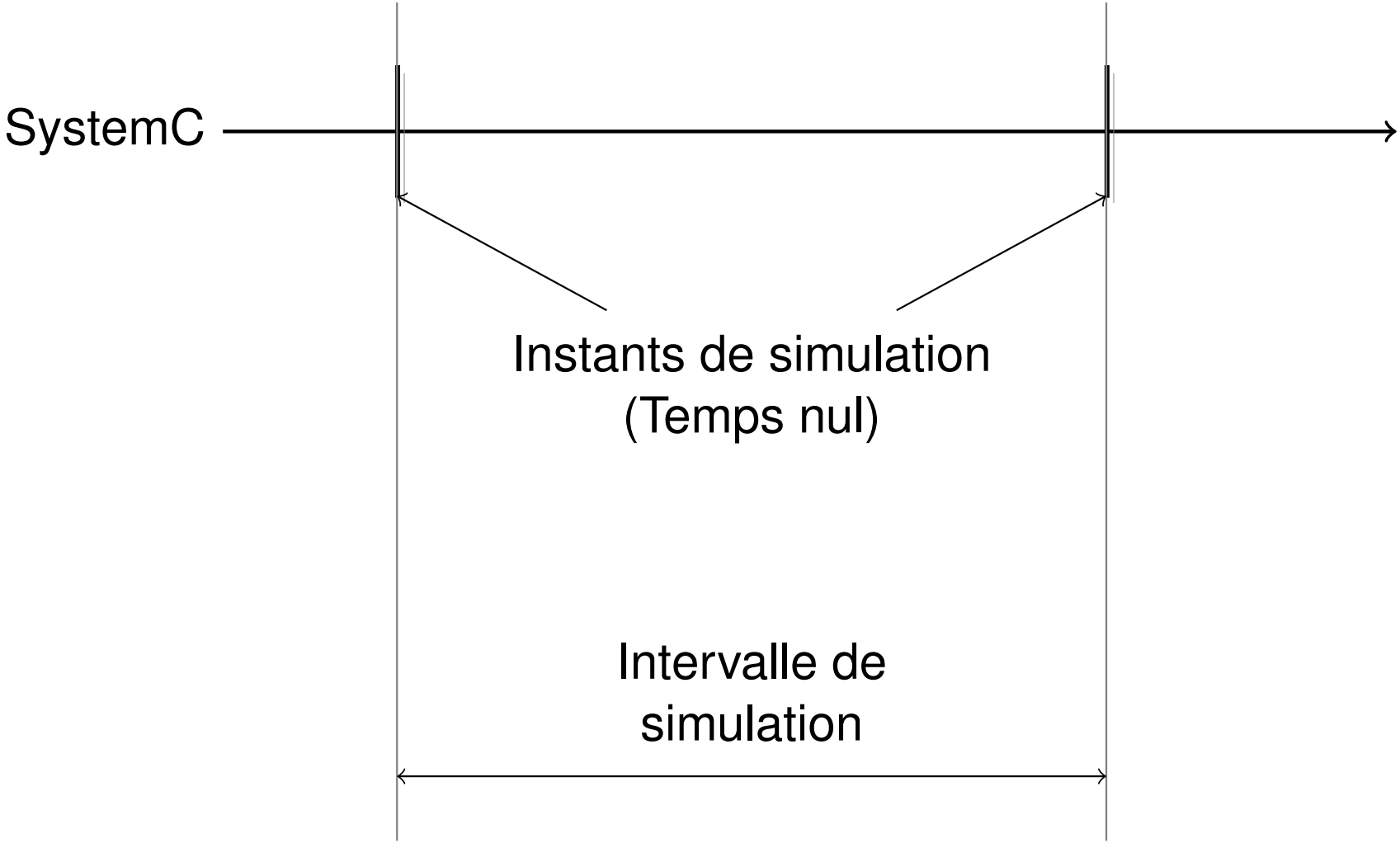


La fonctionnalité peut dépendre des données extra-fonctionnelles
(exemple : capteur de température)

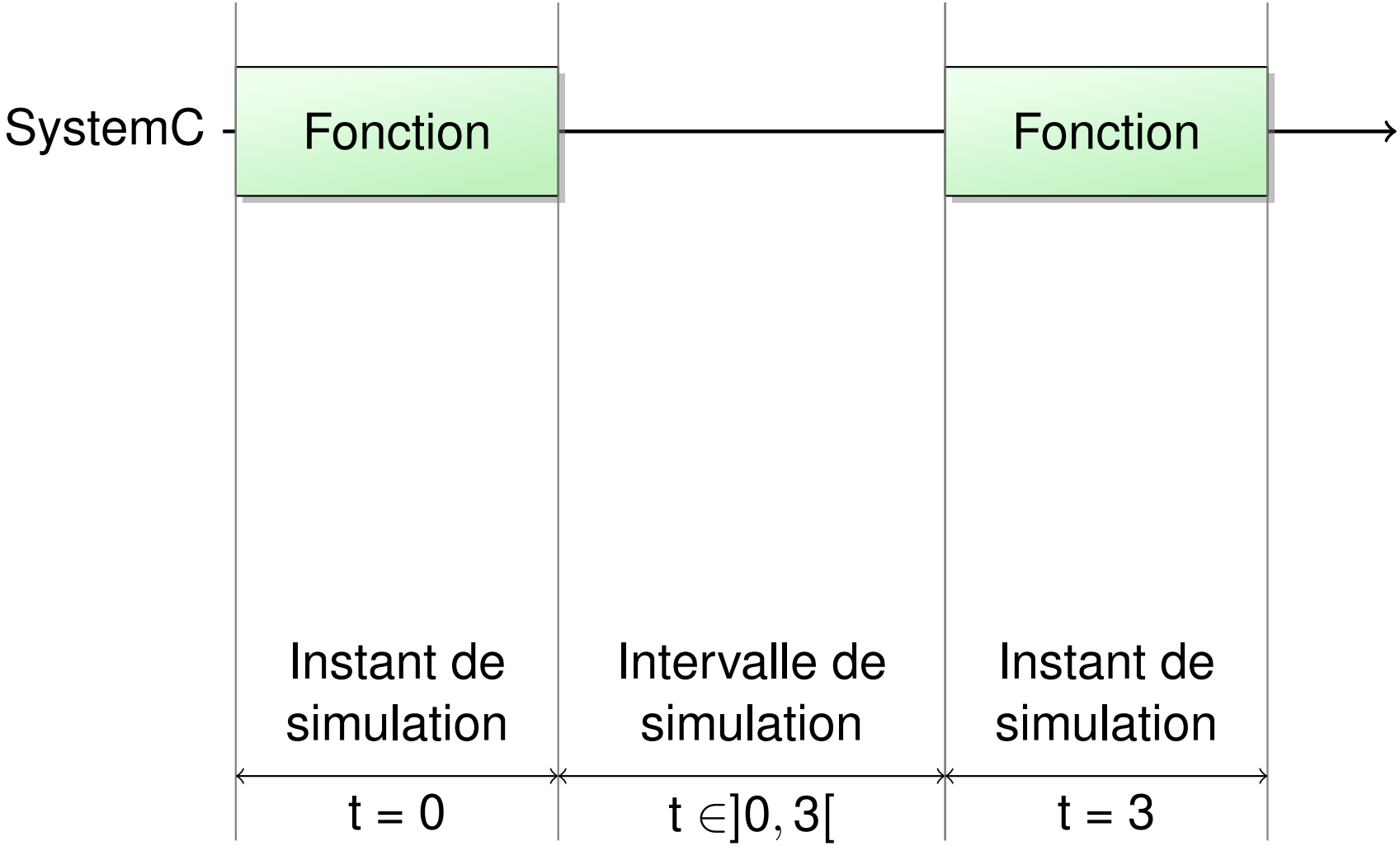
Cosimulation SystemC et solveur extra-fonctionnel



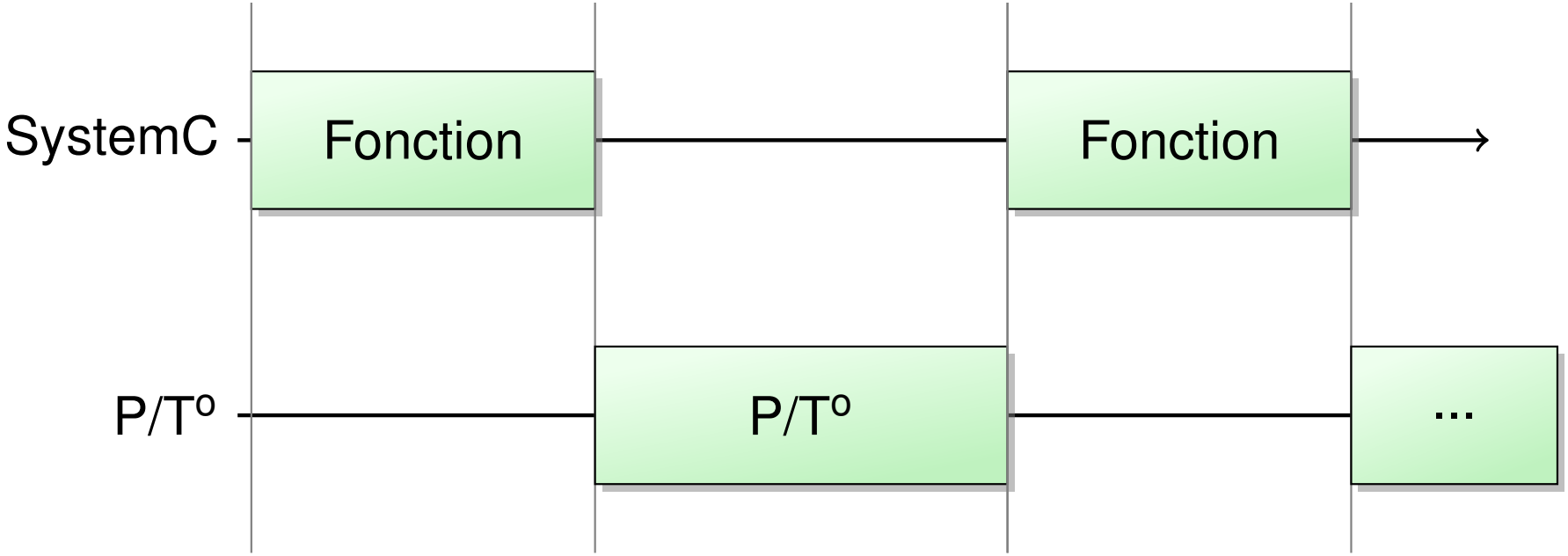
Cosimulation SystemC et solveur extra-fonctionnel



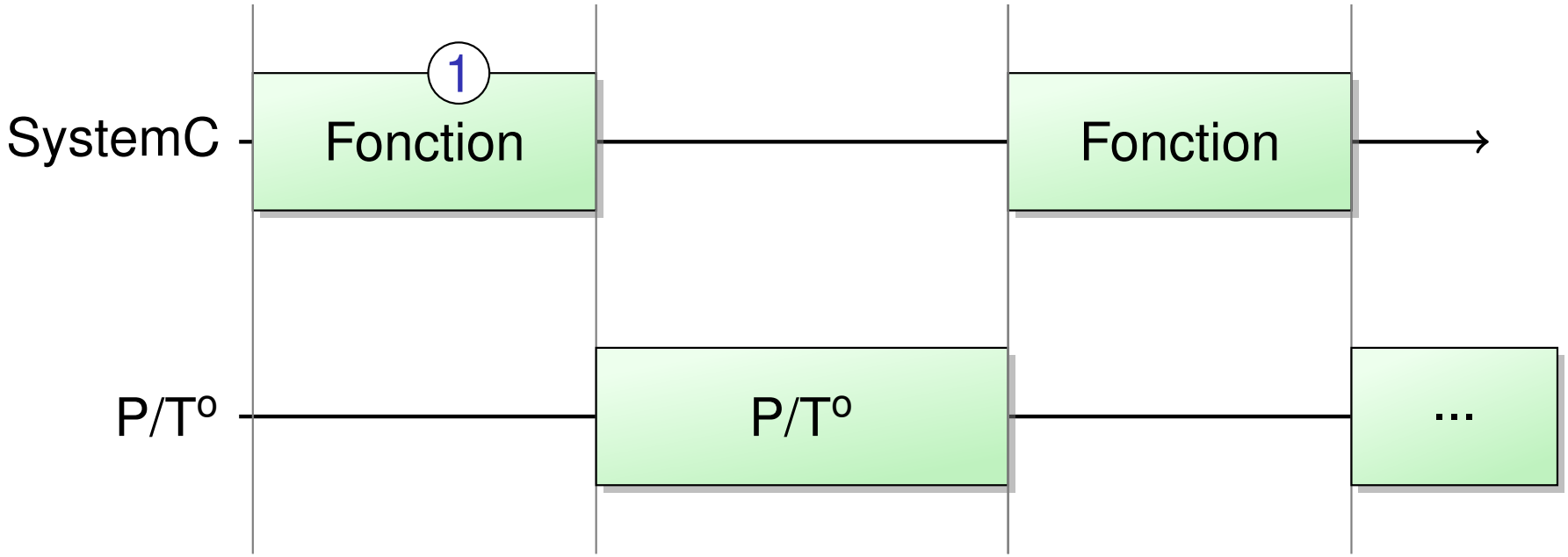
Cosimulation SystemC et solveur extra-fonctionnel



Cosimulation SystemC et solveur extra-fonctionnel

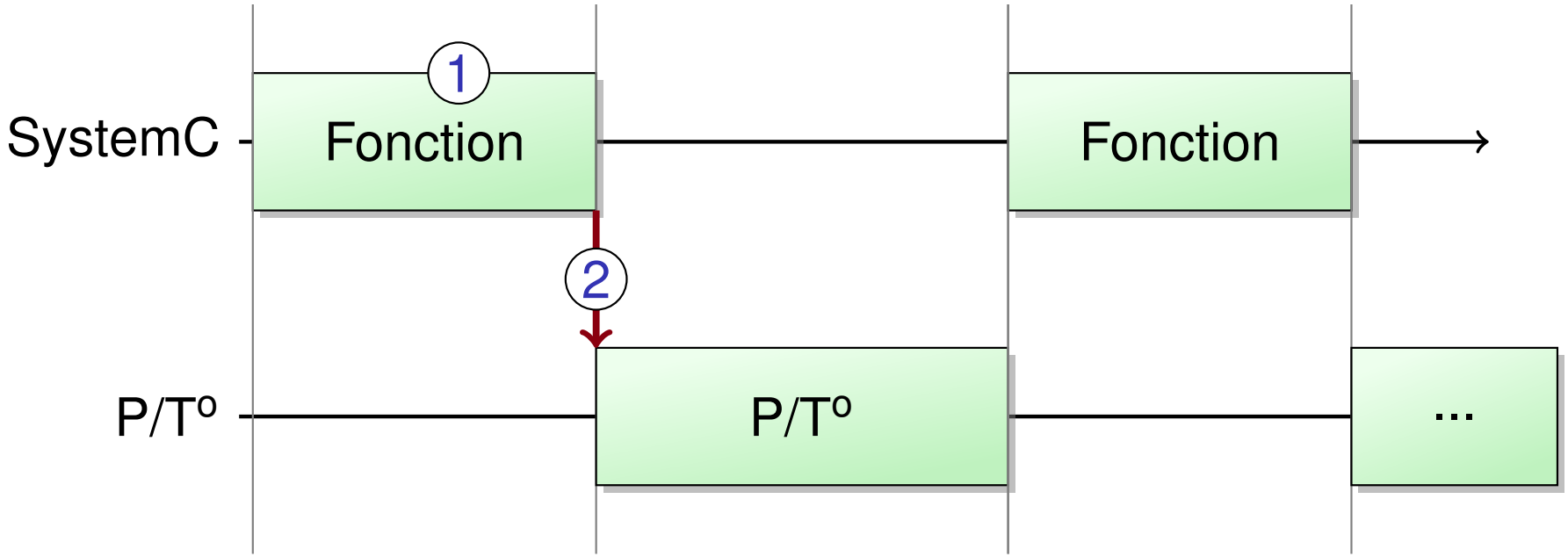


Cosimulation SystemC et solveur extra-fonctionnel



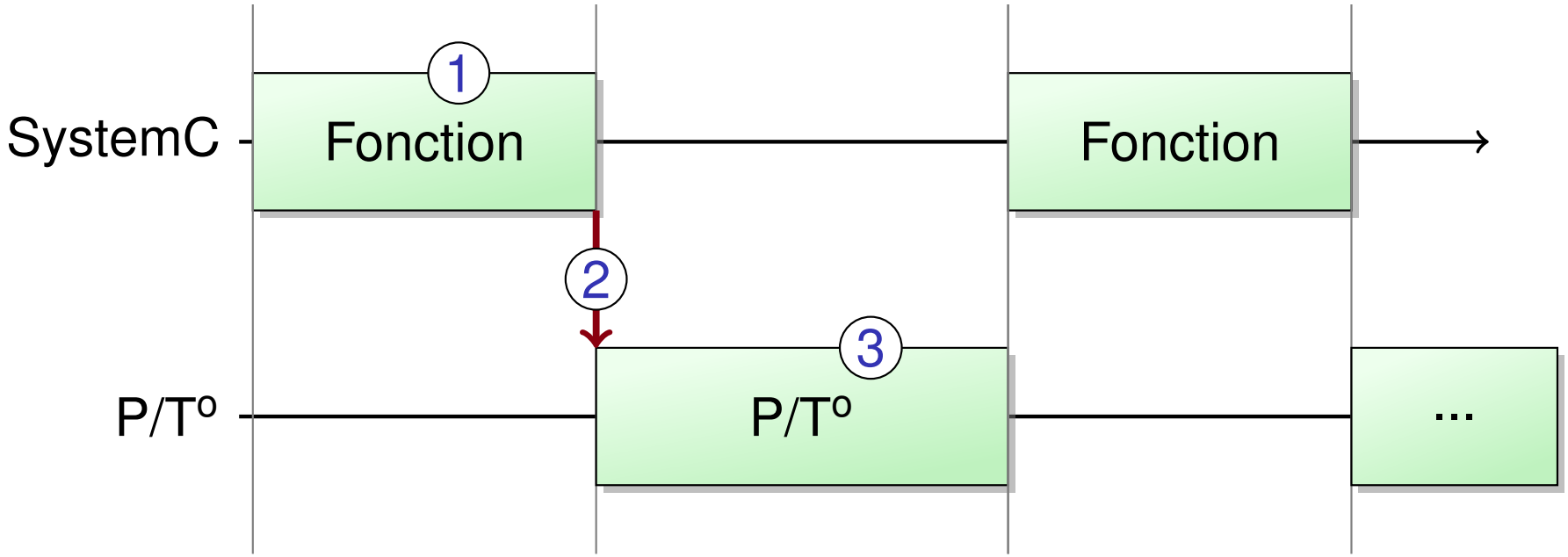
1. SystemC avance la simulation jusqu'à l'instant *t*

Cosimulation SystemC et solveur extra-fonctionnel



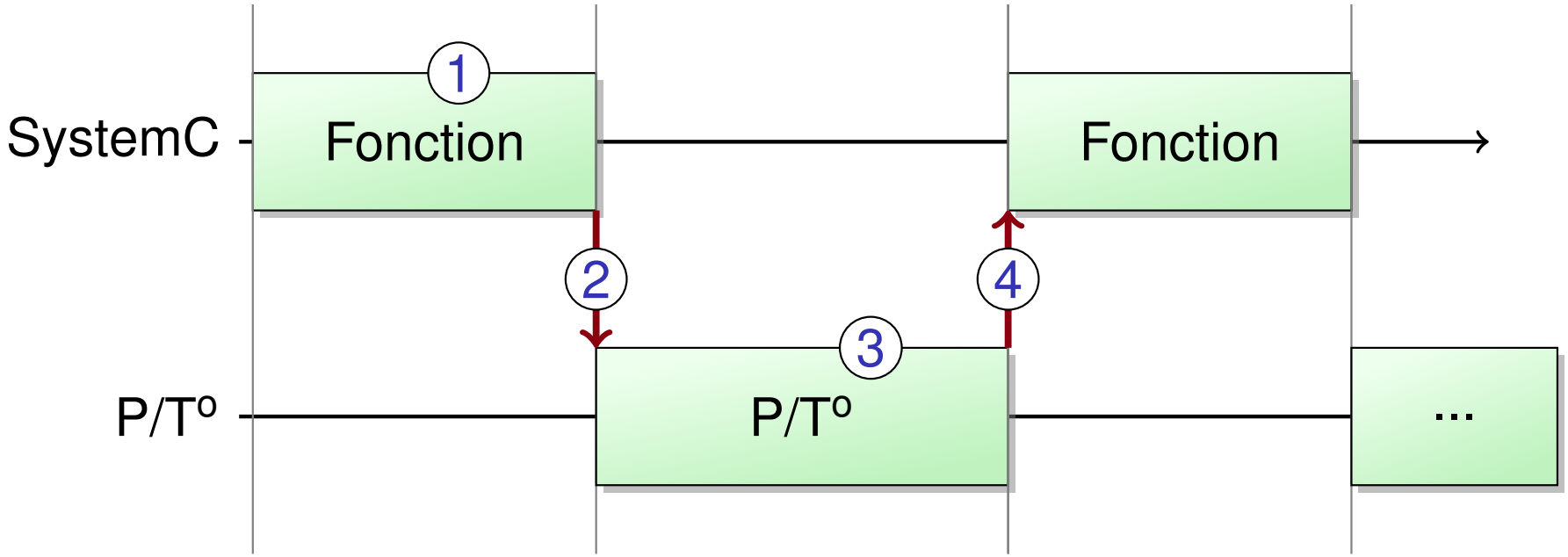
1. SystemC avance la simulation jusqu'à l'instant t
2. SystemC envoie une requête de simulation extra-fonctionnelle sur $[t, t + d]$

Cosimulation SystemC et solveur extra-fonctionnel



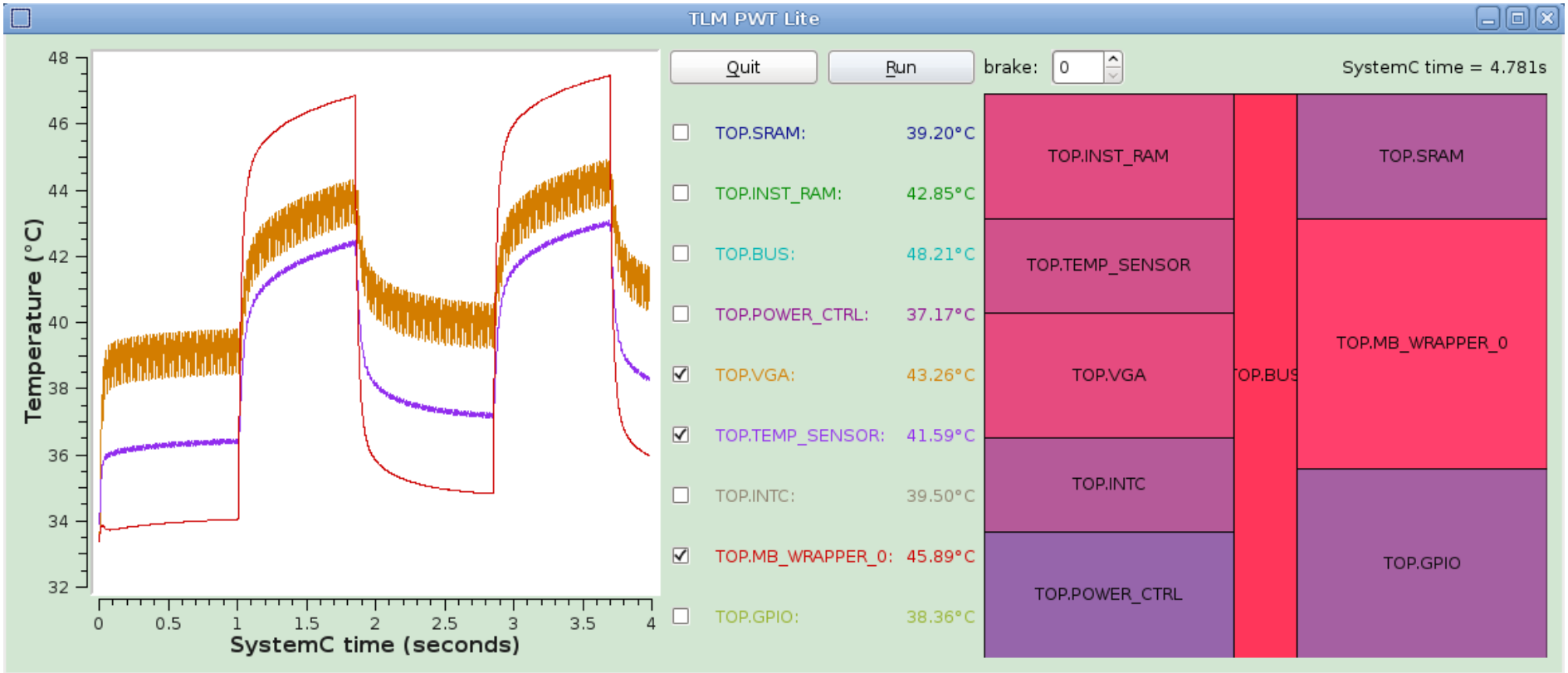
1. SystemC avance la simulation jusqu'à l'instant t
2. SystemC envoie une requête de simulation extra-fonctionnelle sur $[t, t + d]$
3. Le solveur extra-fonctionnel fait le calcul sur l'intervalle

Cosimulation SystemC et solveur extra-fonctionnel

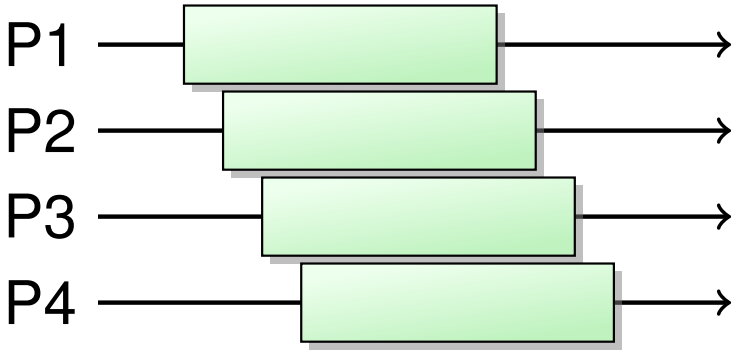


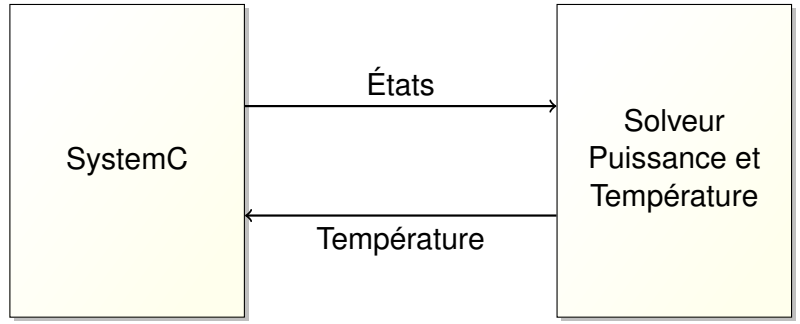
1. SystemC avance la simulation jusqu'à l'instant t
2. SystemC envoie une requête de simulation extra-fonctionnelle sur $[t, t + d]$
3. Le solveur extra-fonctionnel fait le calcul sur l'intervalle
4. SystemC reprend la main

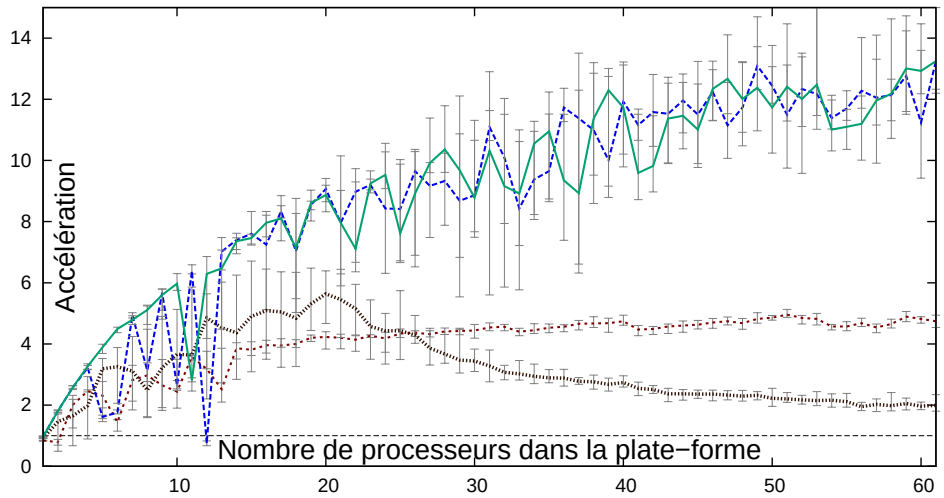
Résultats

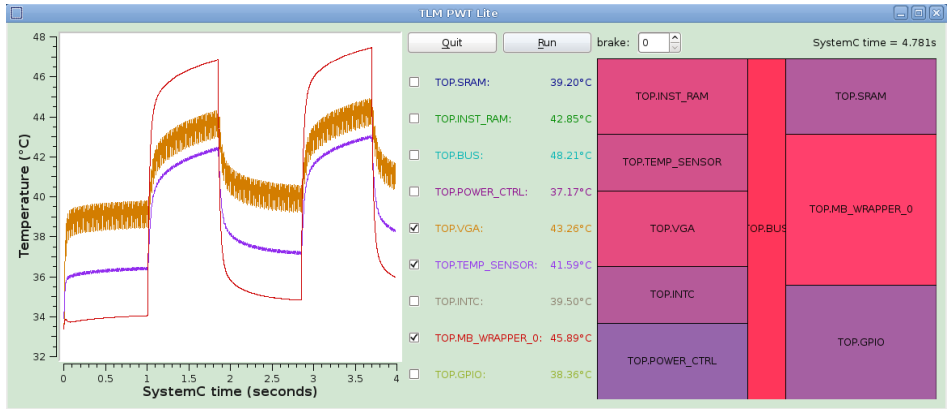

TLM PWT Lite
Quit
Run
brake:
0
SystemC time = 4.781s
TOP.SRAM: 39.20°C
TOP.INST_RAM: 42.85°C
TOP.BUS: 48.21°C
TOP.POWER_CTRL: 37.17°C
TOP.VGA: 43.26°C
TOP.TEMP_SENSOR: 41.59°C
TOP.INTC: 39.50°C
TOP.MB_WRAPPER_0: 45.89°C
TOP.GPIO: 38.36°C
Temperature (°C)
SystemC time (seconds)
TOP.INST_RAM
TOP.SRAM
TOP.TEMP_SENSOR
TOP.MB_WRAPPER_0
TOP.VGA
TOP.INTC
TOP.GPIO
TOP.POWER_CTRL

Conclusion et perspectives

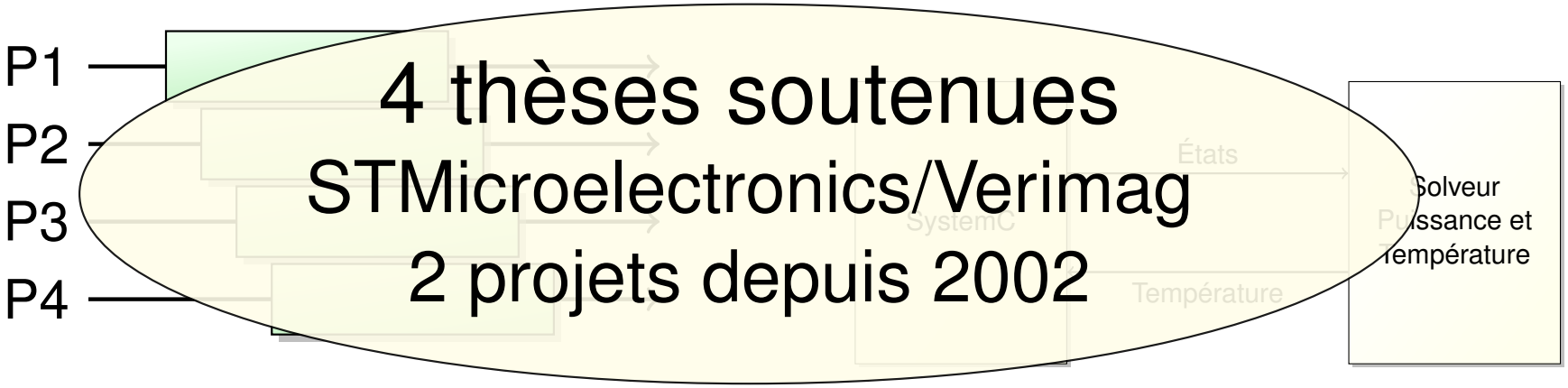


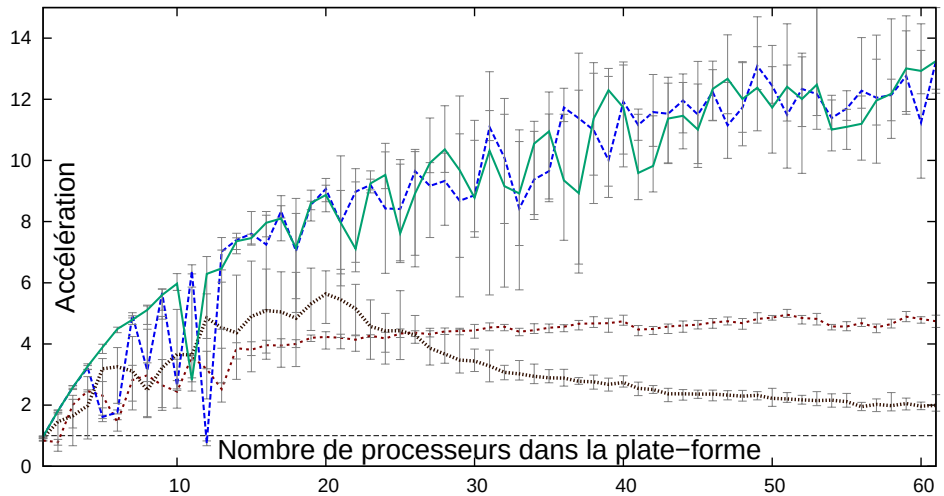


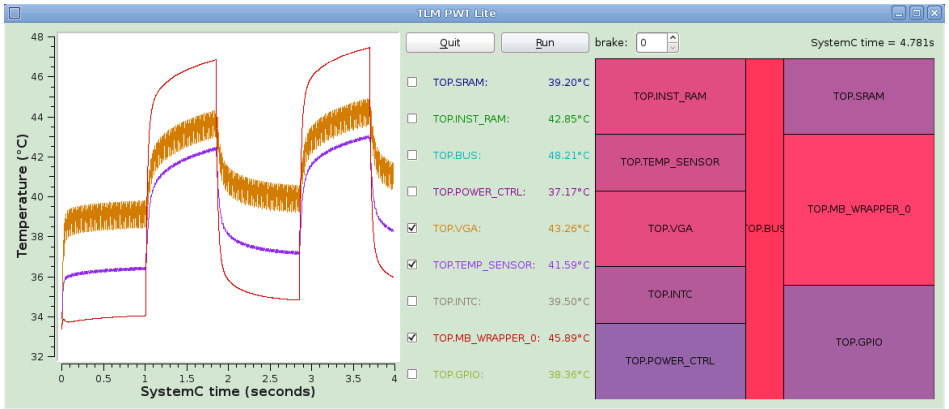




Conclusion et perspectives

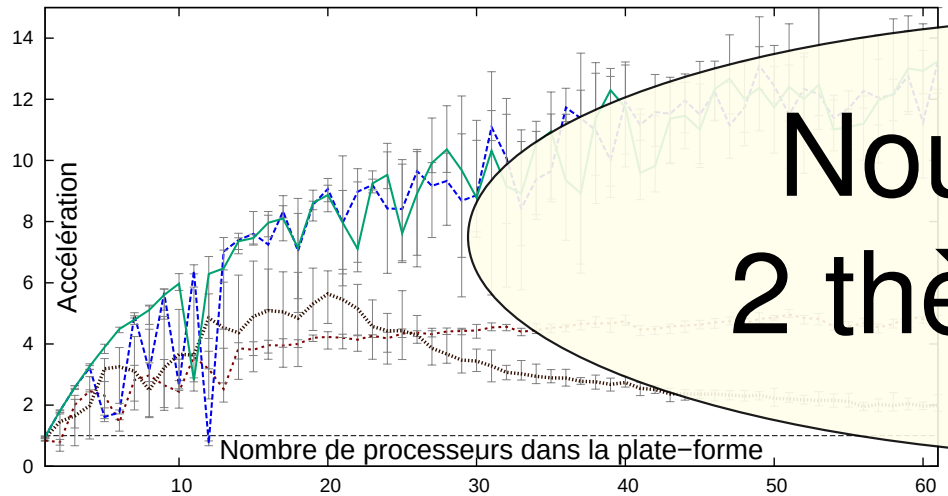






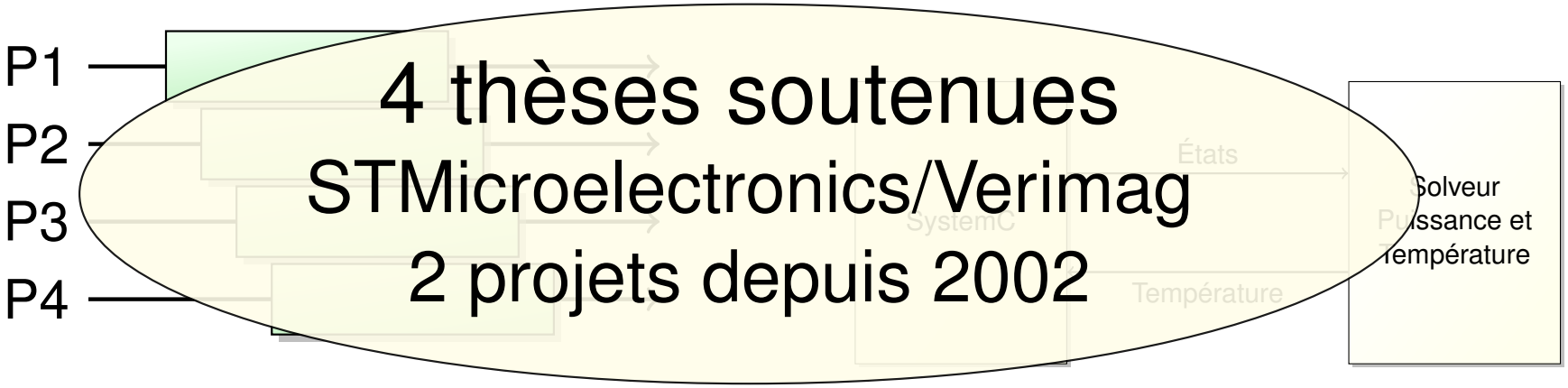
Conclusion et perspectives

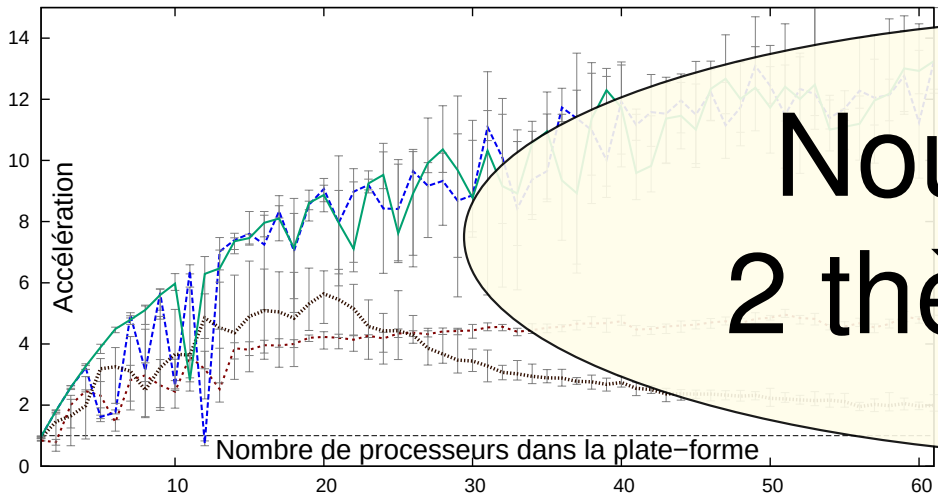
4 thèses soutenues
STMicroelectronics/Verimag
2 projets depuis 2002



Nouveau projet,
2 thèses en cours

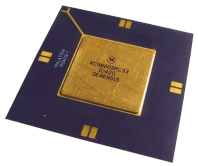
Questions ?





Merci !

Sources



http://www.fotopedia.com/items/flickr-367843750
(oskay@fotopedia, CC Attribution 2.0 Generic)