Simulation coopérative et parallèle :

Expérimentations avec le scheduler SystemC

Mohamed Zaim Wadghiri

Encadré par :

Matthieu Moy

Claire Maiza

Travaux d'Etudes et de Recherches - Verimag

Contexte du travail

Laboratoire et tuteurs

- Laboratoire Verimag. Equipe Synchrone.
- Tuteurs : Matthieu Moy et Claire Maiza.

Présentation du sujet

- SystemC est un langage de description de matériel.
- Mode de simulation purement coopératif : l'ordonnanceur n'est pas préemptif (les processus doivent rendre la main eux-mêmes)
- **But du TER :** Paralléliser la simulation pour exploiter les nouvelles machines multiprocesseurs en donnant aux processus une notion de durée.

Généralités sur le flot de conception des SoC

- Différentes étapes et niveaux d'abstraction pour la conception d'un SoC
- Plusieurs contraintes à respecter
  - Taille minuscule de la puce
  - Puissance de calcul croissante
  - Consommation énergétique garantissant une bonne autonomie du système
  - Time To Market (TTM)

Simulation des SoC avec SystemC

- SystemC est un HDL (Hardware Description Language)
- Une bibliothèque open-source définie au-dessus de C++
- Permet une modélisation au niveau transactionnel des SoC
- Un bloc = Un module SystemC
- Le comportement du bloc est modélisé par des processus SystemC

Simulation des SoC avec SystemC

- SystemC est un HDL (Hardware Description Language)
- Une bibliothèque open-source définie au-dessus de C++
- Permet une modélisation au niveau transactionnel des SoC
- Un bloc = Un module SystemC
- Le comportement du bloc est modélisé par des processus SystemC

Simulation des SoC avec SystemC

- SystemC est un HDL (Hardware Description Language)
- Une bibliothèque open-source définie au-dessus de C++
- Permet une modélisation au niveau transactionnel des SoC
- Un bloc = Un module SystemC
- Le comportement du bloc est modélisé par des processus SystemC

Simulation des SoC avec SystemC

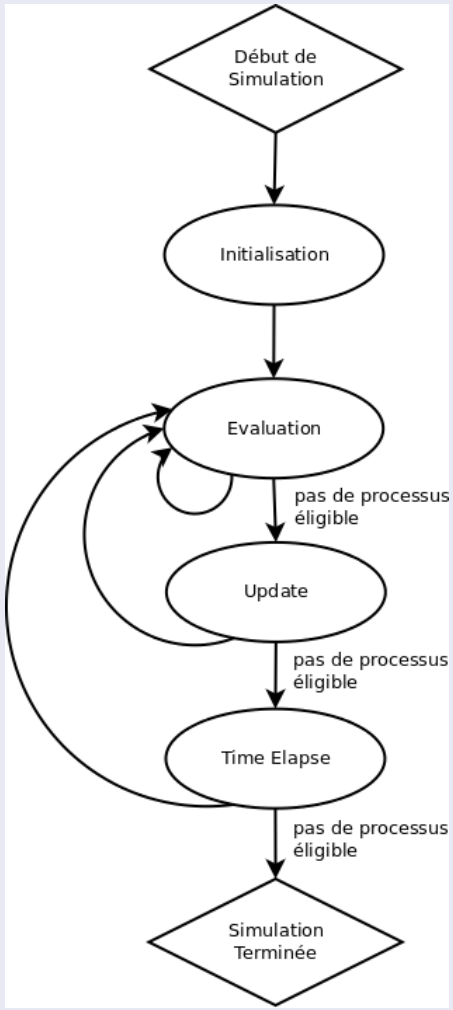
- SystemC est un HDL (Hardware Description Language)
- Une bibliothèque open-source définie au-dessus de C++
- Permet une modélisation au niveau transactionnel des SoC
- Un bloc = Un module SystemC
- Le comportement du bloc est modélisé par des processus SystemC

Simulation des SoC avec SystemC

- SystemC est un HDL (Hardware Description Language)
- Une bibliothèque open-source définie au-dessus de C++
- Permet une modélisation au niveau transactionnel des SoC
- Un bloc = Un module SystemC
- Le comportement du bloc est modélisé par des processus SystemC

Scheduler SystemC

- Scheduler multitâche coopératif
- Nécessité d'ajouter des points de synchronisation ou une liste de sensibilité



Simulation coopérative et parallèle

Simulation coopérative

- Scheduler non préemptif $\Rightarrow$ exécution des processus SystemC de façon coopérative.
- Une simulation exploitant un seul processeur avec une machine qui peut en avoir des dizaines ...

Simulation parallèle

- Exécution des différents processus de façon parallèle.
- Exploitation des processeurs de la machine.
- Simulation beaucoup plus rapide.

La bibliothèque pThread

- Thread = Processus léger. Les threads partagent la mémoire virtuelle.
- Bibliothèque `pThread` : implémentation des threads (Norme `POSIX`).
- Idée de la solution de parallélisation de SystemC : déléguer les fonctions exécutées par les processus SystemC (ou une partie) aux `pThreads` $\Rightarrow$ Exploitation du parallélisme de l'OS.

Idée générale de la parallélisation

Version séquentielle

```
void Module::Proc_A() {
  f();
  wait(10, SC_NS);
}
```

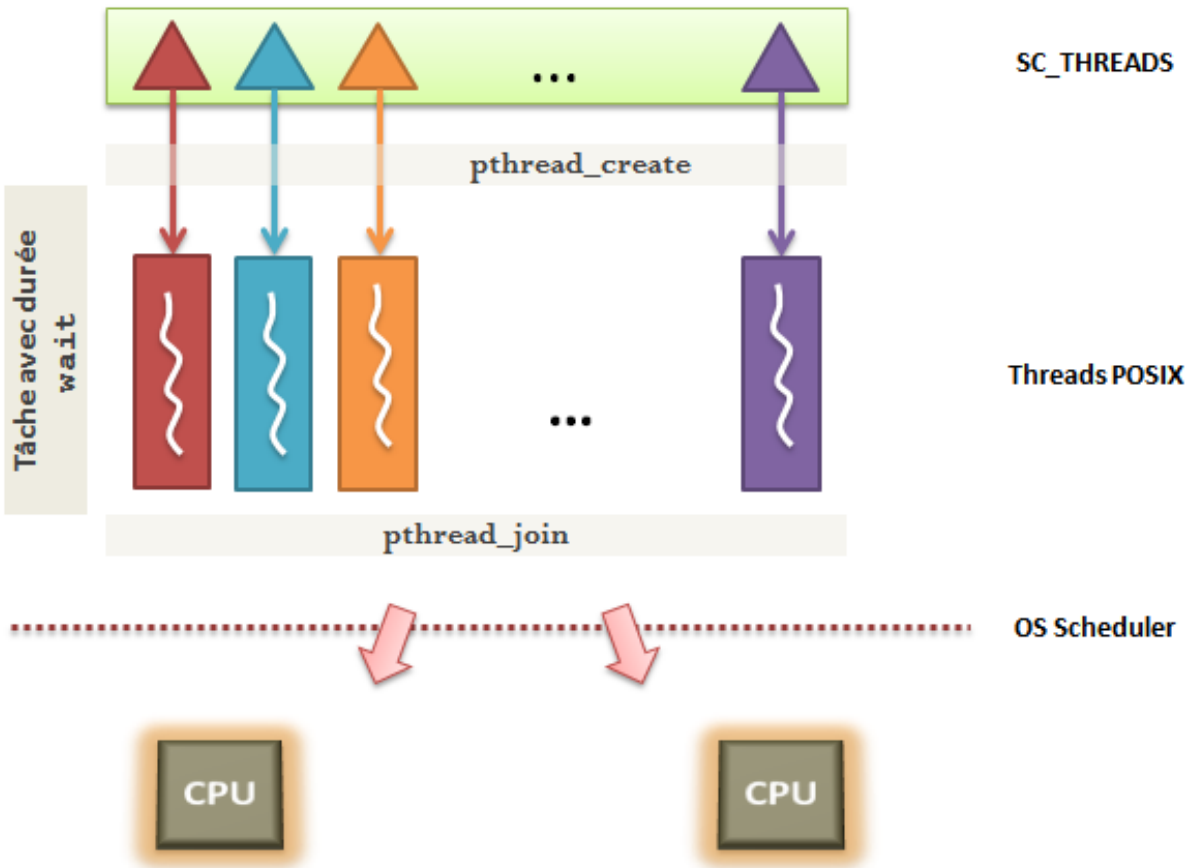
```
void Module::Proc_B() {
  g();
  wait(5, SC_NS);
}
```

Version parallèle

```
Module::Proc_A() {
  during(10, SC_NS, f, this);
}
```

```
 Module::Proc_B() {
   during(5, SC_NS, g, this);
}
```

Solution naïve : One-To-One SC_THREAD - pThread



Solution naïve : One-To-One SC_THREAD - pThread

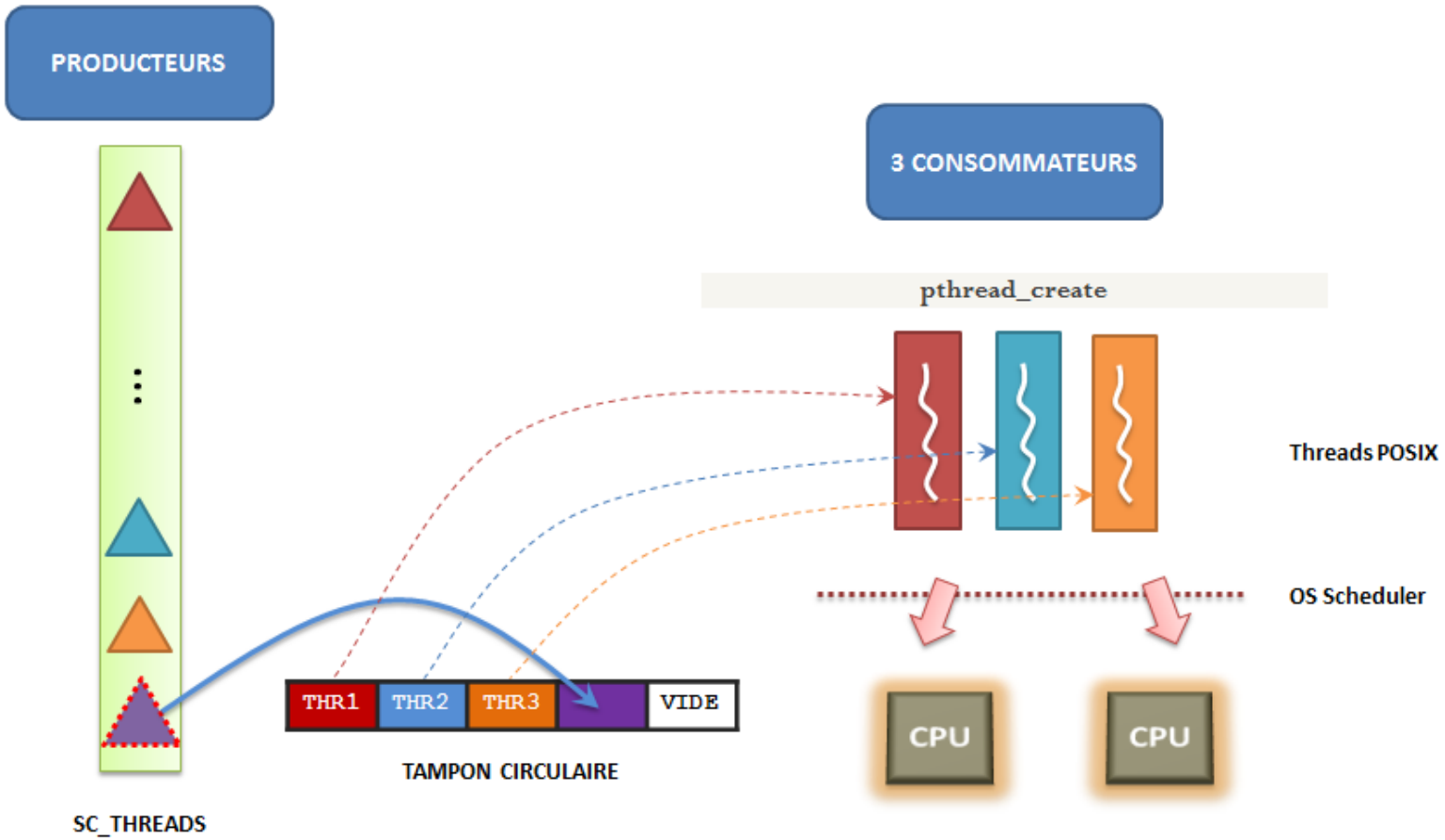
Implémentation

```
void during_threads(double time_ns,
                    sc_time_unit unit,
                    void *(*routine) (void *),
                    sc_core::sc_module *m)
{

    pthread_create(&thr, NULL, routine, m);
    wait(time_ns, unit);
    pthread_join(thr, &st);

}
```

Deuxième approche basée sur le modèle producteur/consommateur



Deuxième approche basée sur le modèle producteur/consommateur

Implémentation

```
void during_prod_cons(double time, void *(*routine) (void *))
{
    ecrire_tampon(routine);
    wait(time, SC_NS);
}

void init_prod_cons()
{
    //initialisation du tampon et des différents sémaphores
    [...]
    for (int i = 0; i < 3; i++)
        pthread_create(&tabpthr[i], NULL, lire_tampon, NULL);
}
```

Test d'une affectation de variable

- Chaque module dispose d'une variable et d'un `SC_THREAD` l'affectant à une valeur plusieurs fois
- Instanciation de plusieurs modules

	Version séquentielle	**Sol.** `1 SC_Thread↔1 pThread`	**Sol. prod/cons**
Machine Dual Core	`0.65 s`	`14.50 s`	`4.83 s`
Machine Quad Xeon	0.22 s	6.36 s	1.58 s
Machine 32 coeurs	0.18 s	21.72 s	7.25 s

Test avec la fractale de Mandelbrot

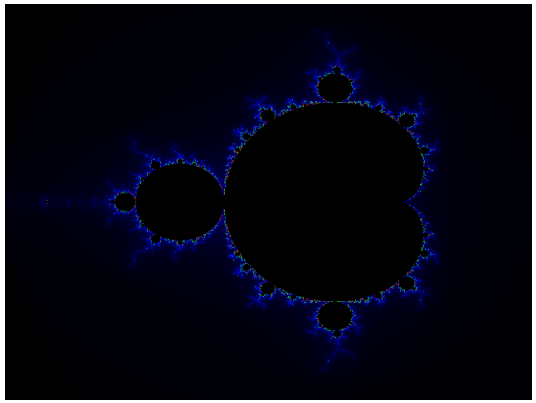
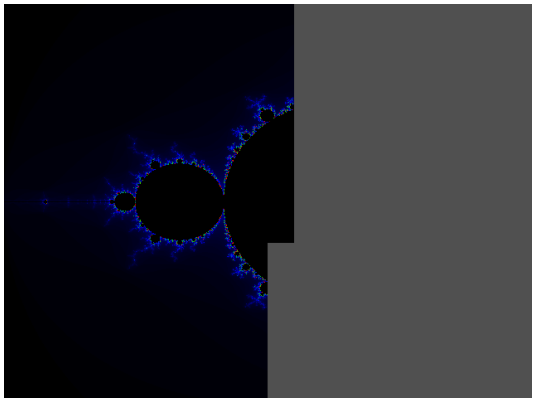
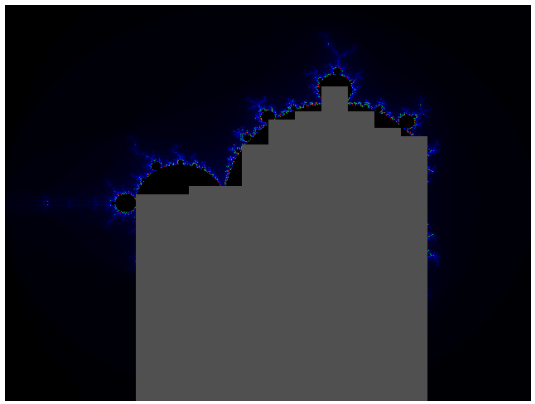


Figure: Fractale de Mandelbrot



(a)



(b)

Test avec la fractale de Mandelbrot

- Fractale de Mandelbrot demande un calcul coûteux pour choisir la couleur de chaque pixel.
- Chaque processus SystemC dessine le prochain slice libre jusqu'à la fin

	Version séquentielle	**Sol. 1 SC_Thread↔1 pThread**	**Sol. prod/cons**
Machine Dual Core	13.71 s	7.78 s	7.08 s
Machine Quad Xeon	9.91 s	3.78 s	3.31 s
Machine 32 coeurs	8.77 s	3.29 s	2.83 s

Conclusion

Notre parallélisation de SystemC a permis de :

- gagner un temps significatif en simulation
- exploiter la robustesse des machines multiprocesseurs actuelles : le ratio d'accélération est presque proportionnel au nombre de coeurs de la machine
- travailler avec les programmes déjà existants : pas de modification de la bibliothèque SystemC

Merci !

Questions ?