RAPPORT MODELISATION TLM EN SYSTEMC

INTEGRATION DU LOGICIEL EMBARQUE

FAIT PAR : YASSMINE OUMOUSS -- YASSER KHARJ

Dans ce, on a étudié l'intégration d'un logiciel embarqué dans un SOC (system on-chip), il s'agit d'un affichage d'une image noir et blanc où une balle bondissante change de direction et de vitesse d'une façon pseudo-aléatoire. Dans ce TP on a étudié deux méthodes de simulation :

- **La simulation native :** Dans cette simulation, le logiciel embarqué est compilé avec le même compilateur que la plateforme, et traité comme un code C. Le code est encapsulé dans le composant TLM « NativeWrapper ».
- **La simulation via ISS :** Dans cette simulation, on utilise un ISS « Instructions Simulator Set ». Dans ce cas le processeur cible est RISC-V.

La simulation native :

Dans ce TP on a étudié l'emballage natif des interruptions en utilisant des méthodes pour gérer les interruptions :

- hal_wait_for_irq() : utilisé pour attendre les interruptions.
- Interrupt_handler_internal() : utilisé pour gérer les interruptions au moment de leur réception.
- hal_write32(unsigned int adrr, unsigned int data) : utilisé pour écrire la donnée « data ».
- hal_read32(unsigned int addr) : utilisé pour lire une donnée de type unsigned_int.
- get_instance() : utilisé pour la redirections des méthodes vers celles de la classe « NativeWrapper ».

Dans la classe NativeWrapper, on utilise variables suivantes :

- Irq : qui correspond au signal d'interruption de type est sc_in<bool>
- Socket : initié dans le fichier *native_wrapper.h.*

```
ensitlm::initiator_socket<NativeWrapper> socket;
```

- Interrupt : une variable booléenne.

P.S : Les méthodes sont implémentées dans le fichier native_wrapper.cpp

Résultat :

Pour afficher le résultat nous avon procéder comme suit :

- cd native-wrapper
- make
- ./run.x