Labo 7 sur QuartusII

VHDL : Synthèse et utilisation d'un processeur

Le but de la séance est de vous permettre d'intégrer un processeur à l'intérieur d'une application complète. L'application choisie est la simulation de la trajectoire d'objets dans l'espace en fonction de diverses forces qui agissent dessus.

Vous trouverez sur le site les fichiers suivants :

Processeur.vhd : le code VHDL du petit processeur vu au cours (presque terminé)
Physic.src : un petit programme en assembleur qui tourne sur le processeur en question
Physic.mif : Le fichier de configuration de la mémoire du processeur avec le programme.

Procasm.def : Le fichier de définition de l'assembleur de notre processeur.
Wintim32.exe : un méta-assembleur très buggé.
Wintim32.hlp : le fichier d'aide du méta-assembleur très buggé.

Test_demodisp : Un fichier graphique d'implémentation incomplet.

Pour simplifier les aspects algorithmiques, on suppose qu'à chaque nouvelle image, les opérations suivantes doivent être réalisées pour chaque objet:

1) Calcul des forces (Fx,Fy) qui agissent sur l'objet (selon l'application)
2) Calcul des vitesses : Vx=Vx+Fx; Vy=Vy+Fy;
3) Calcul des positions : Px=Px+Vx; Py=Py+Vy;

De plus, si l'objet arrive sur un bord de l'écran, il doit être réfléchi. Il faut donc inverser sa vitesse (Vx=-Vx ou Vy=-Vy);

Le fichier d'exemple « Physic.asm » illustre le cas d'une seule balle qui se déplace sous l'action d'une force constante en X et Y. Le système fonctionne en virgule fixe avec 4 bits après la virgule.

On demande :

1) Réalisez une simulation du système et assurez-vous que vous comprenez bien le rôle de chacune des instructions assembleur et la manière dont elles sont exécutées par le processeur. Vous remarquerez que le processeur s'arrête après avoir réalisé un seul calcul.
2) Modifiez le fichier assembleur pour que le processeur boucle sans fin de manière à observer l'évolution du système sur une plus grande échelle de temps. Utilisez le méta-assembleur pour produire le code assemblé. L'évolution des positions et des vitesses vous semble-t-elle correcte ?

3) Intégrez le processeur dans une application qui visualise la trajectoire de la balle. Ceci implique plusieurs choses :

    a. Le processeur doit pouvoir « sortir » les valeurs des positions. *Ajoutez des signaux PosX et PosY en sortie.*
    b. Le processeur doit pouvoir prendre connaissance de la venue d'une nouvelle trame. *Ajoutez un signal d'entrée NewFrame qui, dans un premier temps, sera utilisé comme une interruption « reset ».*
    c. Le processeur doit travailler à une fréquence compatible avec sa complexité (environ 7Mhz). *Utilisez un diviseur d'horloge (un simple compteur 2 bits suffit et la sortie du bit de poids fort devient maintenant votre seule et unique horloge globale !).*
    d. Vous pouvez alors intégrer votre ressource dans l'environnement de test « test_demodisp.gdf ».

4) Ce processeur ne permet pas d'adressage immédiat, ce qui serait bien pratique pour l'utilisation de constantes.
    a. Spécialisez l'instruction de chargement « LDR » en utilisant le champ de bits IR[5..3] qui était inutilisé pour cette instruction :
        i. LOW (000) permet de charger le poids faible du registre avec la valeur contenue dans le champs d'adressage (IR[15..8]);
        ii. HIGH (001) permet de charger le poids fort du registre avec la valeur contenue dans le champs d'adressage (IR[15..8]);
        iii. MEM (111) : réalise l'opération classique LDR
    b. Modifiez le fichier de spécifications du méta-assembleur et le programme assembleur lui-même et recompilez pour tester vos résultats.

5) Utilisez la souris pour faire varier la force qui agit sur la balle de sorte que :
    Fx=(MouseX-320)/32
    Fy=(MouseY-240)/32

    Ou encore :
    Fx=(MouseX-PosX)/32
    Fy=(MouseY-PosY)/32

    Et pour les motives … simulez le comportement de trios balles qui s'attirent l'une l'autre selon une force :
    Fx= (AutrePosX1+AutrePosX2-2*PosX)/32
    Fy= (AutrePosY1+AutrePosY2-2*PosY)/32

Bon travail !

P.S : Qui trouvera (et corrigera) le petit bug caché du processeur ?