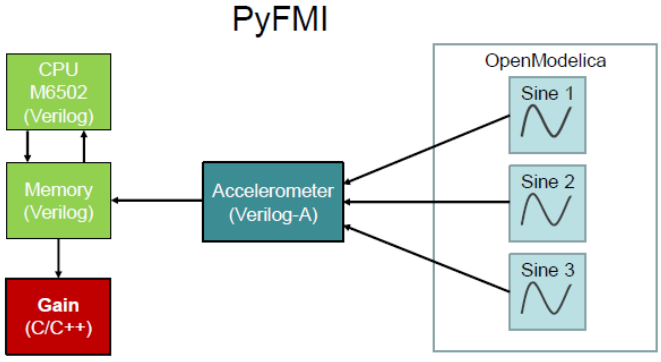
Embedded System Design - Third Assignment

Fabio Fiorio - VR422016

***Sommario*—L'elaborato si prefigge come obiettivo il completamento del tutorial su una macchina virtuale nel quale si desidera implementare ed integrare ai modelli già forniti, un componente gain che, sfruttando lo standard FMI, amplia di 10 volte il segnale di ingresso.**
Il segnale finale risulterà essere l'ingresso sinusoidale discretizzato.

I. Introduzione

Si desidera completare il tutorial che sfrutta lo standard FMI su macchina virtuale per la creazione di VPs (Virtual Platforms) da IP eterogenee. Vengono già forniti 3 modelli che implementano rispettivamente: un'accelerometro realizzato in Verilog-A, una memoria ed un processore m6502 realizzati in Verlog.
L'obiettivo finale è terminare il quarto componente, un gain, che riceve un valore e lo restituisce 10 volte maggiore, ed interfacciare il tutto in una piattaforma virtuale.



PyFMI

Dopo l'aggiunta del gain, il segnale sinusoidale, che in precedenza risultava continuo, risulta essere ora discretizzato.

II. Background

L'FMI (Functional Mock-up Interface) è uno standard che mi permette di far interagire tra loro componenti di natura eterogenea.
Un componente che implementa l'interfaccia prende il nome di FMU (Functional Mock-up Unit) ed è essenzialmente la combinazione di un file .xml che mi descrive l'interfaccia del sistema (porte, tipi di dato, ecc...) ed un sorgente .c che implementa le funzionalità del componente stesso.

III. Metodologia applicata

Una volta analizzate le specifiche ed effettuato tutto il setup dell'ambiente della macchina virtuale, si è lanciato il coordinator, ovvero il programma Python che sfrutta la libreria PyFMI per il caricamento e interazione con l'FMU, in cui sono stati realizzati tutti i binding tra i diversi moduli (già implementati).
Il programma mostra inizialmente il semplice grafico di una sinusoide.
Si è proceduto poi con il completamento del file .xml in cui sono state aggiunte le porte del componente gain:

```
<ScalarVariable causality="input"
description="data flag" name="data_rdy"
valueReference="0" variability="discrete">
    <Boolean start="false"/>
</ScalarVariable>
<ScalarVariable causality="output"
description="result flag" name="result_rdy"
valueReference="1" variability="discrete">
    <Boolean />
</ScalarVariable>
<ScalarVariable causality="input"
description="data from the memory"
name="data" valueReference="0"
variability="discrete">
        <Integer start="0"/>
</ScalarVariable>
<ScalarVariable name="result"
valueReference="1" description="integer"
causality="output" variability="discrete"
initial="approx">
    <Integer start="0"/>
</ScalarVariable>
```

- data_rdy: rappresenta il valore letto, un intero, inizializato a 0. Il suo valore (valueReference) sarà utile in seguito quando verranno implementati i metodi per il getting e il setting del valore delle variabili nel file fmuInterface.c.
- data_rdy: booleano che indica se il dato è pronto o meno.
- result: output intero che rappresenta il valore di data ampliato.
- result_rdy: booleano che indica se il risultato è pronto.

Successivamente si è passati all'interfaccia dell'FMU, con il completamento dei metodi setter e getter relativo ai tipi integer e boolean, per leggere/scrivere variabili di input/output.
Tutti i metodi utilizzati nel file .c sono funzioni dello standard FMI e sono implementate nelle 3 librerie contenute nella cartella fmi_headers.
Si è completato, infine, il metodo fmi2DoStep(), che si occupa di portare avanti l'intera simulazione ed il quale esegue un prodotto * 10 del valore dell'input.
Si è potuto così creare il file FMU contenente il tutto.
Il passo finale è stato completare il coordinator, ovvero aggiungere il nuovo modello alla lista di modelli già presenti.
Il gain comunica direttamente con la memoria, la quale è dotata di apposite porte che prendono e restituiscono dati al gain.
Tutti i dati sono stati registrati in un vettore, per poi essere messi in un grafico.

IV. Risultati

Le due schermate mostrano la situazione senza gain e con il gain:

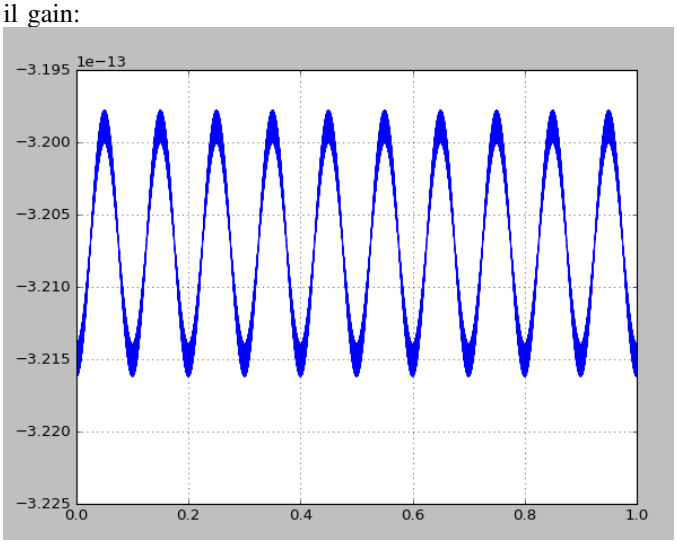


Grafico senza gain

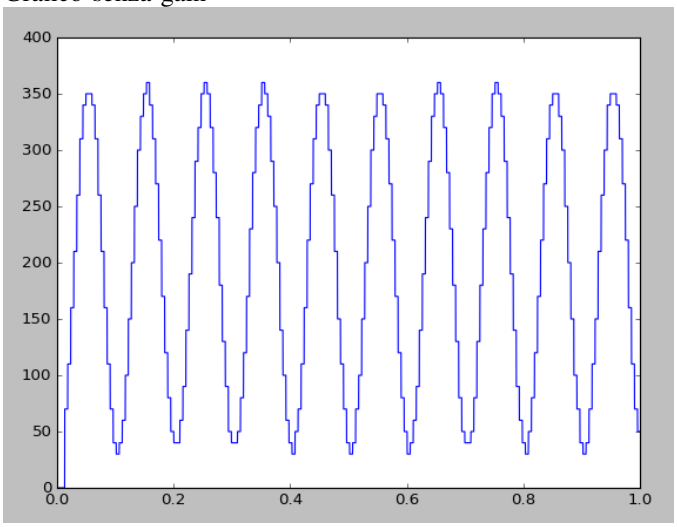


Grafico con gain

Si vede chiarmente come il risultato non è altro che una discretizzazione del segnale sinusoidale.

V. Conclusioni

Possiamo concludere dicendo di essere riusciti ad implementare lo standard ed aver ottenuto i dati attesi.