Modellazione di un sistema hardware per il controllo del livello d'acqua in una cisterna tramite SystemC

Mori Samue - VR439256

***Sommario*—In questo documento vengono riportate le scelte progettuali adottate per la realizzazione di un sistema hardware per il controllo del livello d'acqua all'interno di una cisterna utilizzando diversi livelli di astrazione del SystemC.**

I. Introduzione

Il progetto ha come obiettivo l'implementazione di un sistema per la stabilizzazione del livello dell'acqua all'interno di una cisterna attraverso il controllo dell'apertura della valvola ad essa collegata (Figura 1). Il controllore legge il valore del livello dell'acqua ed in base ad esso decide il comando da inviare: IDLE (valore compreso nell'intervallo), OPEN (valore sotto la soglia) o CLOSE (valore sopra la soglia). Inoltre viene inviata l'apertura massima oltre la quale la valvola non deve aprirsi. Essendo cifrati tramite l'algoritmo Xtea i dati devono essere decifrati tramite un modulo intermedio.
Il progetto si basa sui concetti di modularità e di riuso. Si è partiti da singoli moduli astratti, raffinati più volte, fino ad arrivare ad un impianto, formato dai vari moduli, che fornisce risultati ottimali entro i limiti delle specifiche date.
Per la realizzazione del progetto sono state utilizzate le librerie SystemC e SystemC-ams attraverso le quali è stato possibile sviluppare i moduli in stili di progettazione e livelli di astrazione diversi.

II. Background

Sono state utilizzate le seguenti librerie:

- SystemC[1]: Libreria standard di classi C++ per la progettazione di sistemi HW o per sistemi ibridi hardware/software a diversi livelli di astrazione,
- SystemC-AMS[2]: Libreria standard di classi C++ basata su systemC per la progettazione di sistemi con funzionalità analog/mixed-signals.

Le componenti digitali sono state descritte utilizzando i seguenti livelli di astrazione:

- RTL (Register Trasfer Level): è il livello di astrazione più basso fornito dal SystemC, si basa sui concetti di porte, segnali e registri,
- TLM (Transaction-Level Modelling): livello di astrazione più alto rispetto a RTL, si concentra sulle funzionalità, su come i moduli interagiscono tra loro.

Mentre per le componenti analogiche sono stati utilizzati i seguenti formalismi di modellazione:

- LSF (Linear Signal Flow): modello a eventi continui a tempo continuo, non conservativo,
- TDF (Timed Data Flow): modello a eventi discreti in tempo discreto, non conservativo.

III. Metodologia applicata

Seguendo l'approccio modulare si è iniziato dallo sviluppo delle singole componenti, partendo dal modulo per la cifratura e decifratura Xtea.

A. Xtea TLM

Il primo passo è stata la rappresentazione in TLM del modulo Xtea. Sono state svilupate tre varianti:

- TLM UT (Untimed model),
- TLM LT (Loosely timed model),
- TLM AT4 (Approximately timed model).

Si è deciso di sviluppare il modulo Xtea completo nonostante nel modulo finale non sia necessaria la parte di cifratura.
Fin da subito è stato chiaro che il payload TLM non fosse sufficiente e quindi è stato necessarario definire una struttura di supporto. La struttura contiene le seguenti informazioni:

- due parole da cifrare/decifrare,
- quattro chiavi,
- un booleano per selezionare la modalità,
- due parole per il risultato.

Nella struttura sono stati inseriti i risultati per semplificare la restituzione dei calcoli.
Inoltre ci si è resi conto che non erano necessari sottomoduli, entrambe le funzioni vengono svolte da un unico modulo. Per controllare la sua correttezza è stato collegato ad un testbench, come rappresentato in figura:

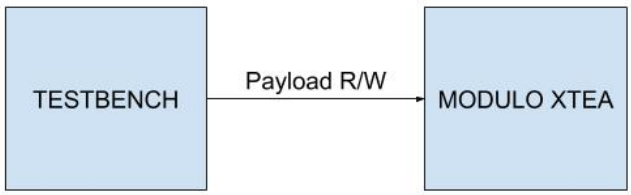

Figura 2: Schema riassuntivo dei moduli TLM UT e TLM LT.

Il testbench agisce come initiator comunicando con il modulo Xtea, il target. L'initiator chiama la funzione *b_transport* del target e si mette in attesa della risposta. Il target, ricevuta la richiesta, la elabora ed invia la risposta al richiedente. Terminato ciò, il testbench può verificare il risultato della computazione analizzando i campi presenti nella struttura

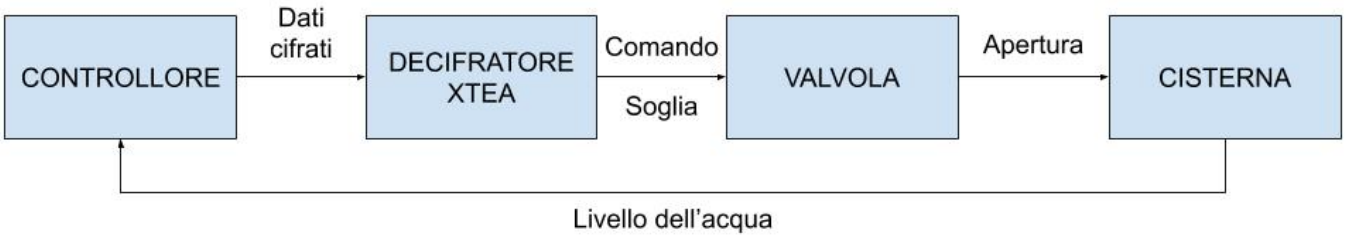

Figura 1: Schema generale dell'impianto

precedentemente descritta. Dato l'utilizzo di TLM, il quale si concentra sulle transazioni piuttosto che sull'effettiva implementazione, l'algoritmo Xtea è stato implementato in C++. Si è poi passati alla rappresentazione in TLM LT. Il passaggio dalla versione UT non è stato complicato, infatti è stata necessaria solo l'aggiunta della nozione di tempo, la quale serve a simulare il tempo speso nella computazione.
Infine si è passati alla rappresentazione in TLM AT4. Il passaggio dalla versione LT è stato più complicato rispetto al precedente. La comunicazione si differenzia dalle due varianti precedenti, nel protocollo AT4 è prevista una comunicazione bidirezionale tra initiator e target che ha portato ad una modifica significativa del testbench.

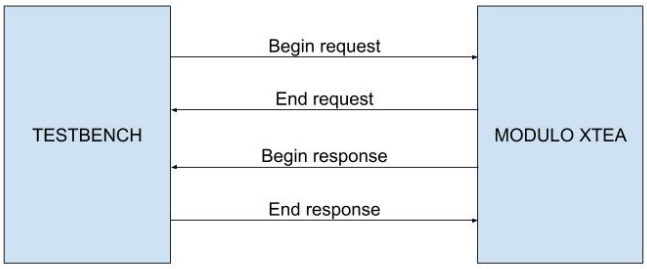

Figura 3: Schema riassuntivo della comunicazione dei moduli in TLM AT4.

B. Xtea RTL

Il modulo Xtea è stato poi modellato in RTL. Grazie allo sviluppo in TLM è stato più facile rappresentare l'interfaccia del modulo: le componenti della struttura TLM sono ora porte I/O. Oltre le quali sono state aggiunte altre due porte in input: clock e reset.

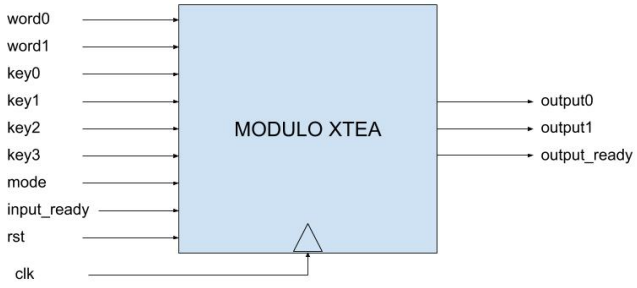

Figura 4: Schema dell'interfaccia del modulo RTL.

Inoltre si è dovuto tradurre le funzioni Xtea scritte in C++ per renderle compatibili con la descrizone RT. Si è quindi definito una EFSM, formata da una FSM (mostrata in figura 5) ed un datapath.
La macchina a stati finiti presenta i seguenti 8 stati:

- START: stato iniziale;
- INITIAL: fase di inizializzazione, rimane nello stato finché non sono pronti nuovi input;
- ASSIGN: vengono assegnati i valori ai segnali w0, w1 ed anche a sum se è stata selezionata la fase di decifratura;
- ENCRYPT_1, ENCRYPT_2: fasi di cifratura;
- DECRYPT_1, DECRYPT_2: fasi di decifratura;
- TERM: vengono scritti i risultati su output0 e output1 e viene messo output_ready a 0.

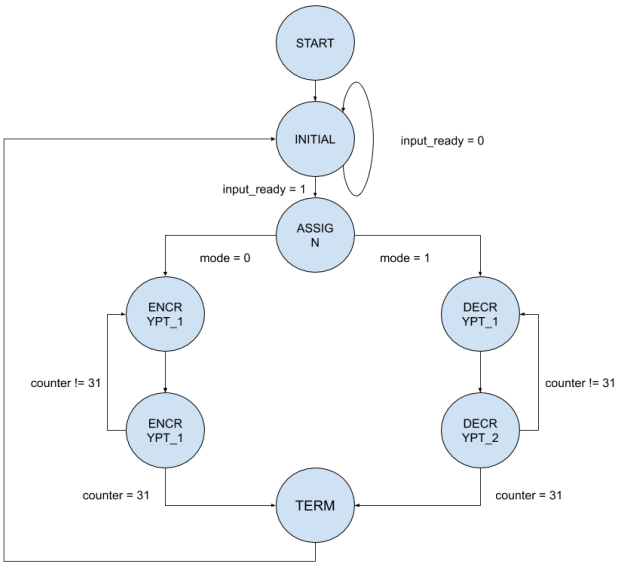

Figura 5: FSM.

Una volta finita la EFSM, è stato definito un SC_MODULE con porte input e output identiche a quelle specificate in precedenza e con lo stesso tipo dei campi utilizzati in TLM. Inoltre FSM e datapath sono stati definiti utilizzando due SC_METHOD separati. La FSM, sensibile alle porte input_ready e mode, si occupata del calcolo dello stato successivo. Il datapath, sensibile a clock e reset, si occupa dell'aggiornamento dello stato corrente e dell'esecuzione delle operazioni previste del nuovo stato. Il tutto è stato poi verificato tramite un testbench collegato al modulo.

C. Cisterna LSF

Si è poi passati allo sviluppo della parte AMS. Siccome la cisterna dell'acqua e la valvola sono espresse tramite valori reali a tempo continuo è stato necessario progettarli tramite la libreria SystemC-ams.
Si è partiti dalla cisterna dove l'evoluzione dell'acqua è specificata dall'equazione differenziale:

$$x' = 0,6a - 0,03x \tag{1}$$

Tale modulo è stato sviluppato utilizzando la rappresentazione LSF. Si è convertito l'equazione differenziale in un diagramma a blocchi, che è stato poi implementato nel SC_MODULE.

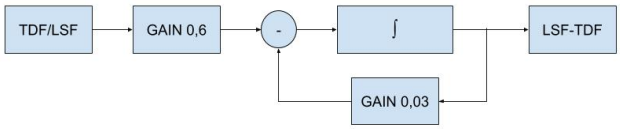

Figura 6: Diagramma a blocchi dell'equazione differenziale della cisterna.

D. Valvola TDF

Dopo aver modellato la cisterna si è passati alla valvola. In questo caso è stata scelta la rappresentazione TDF. È stato creato un modulo SCA_TDF:MODULE all'interno del quale sono stati definiti gli input, gli output e le variabili interne. Il modulo possiede due metodi: set_attributes() e processing(). Il primo serve per inizializzare time step ed eventuali ritardi. Il secondo, che viene invocato ad ogni time step, svolge la funzione di controllo dell'apertura.
Il comportamento della valvola può essere espresso come segue:

- IDLE, non modifica l'apertura della valvola
- OPEN, $a = a + 0.25t$
  dove a è l'apertura e t il time step
- CLOSE, $a = a + 0.25t$
  dove a è l'apertura e t il time step

E. Controllore TLM

Dopo aver implementato la cisterna e la valvola si è passati alla fase di testing, effettuata tramite un controllore in TLM LT separato dal testbench. Inoltre è stato necessario costruire dei transattori che rendessero possibile la comunicazione tra il controllore e i moduli ams.

F. Piattaforma eterogenea

Dopo aver controllato che tutto si comporti correntamente si è passati all'unione dei vari componenti che ha portato alla creazione della piattaforma eterogenea.
Affinché i moduli comunichino tra loro sono necessari dei transattori, componente che permette la comunicazioni tra moduli differenti. Il loro utilizzo comporta ad un aumento del tempo di simulazione e di conseguenza si è dovuto modificare il tempo di attesa del controllore.
Sono stati implementati i seguenti transattori:

- Transattore TLM-RTL: modulo SystemC TLM con numero di porte output pari al numero delle porte input del modulo RTL. Alla chiamata della funzione *b_transport* i valori del payload vengono scritti sulle rispettive porte.
- Transattore RTL-TLM: modulo SystemC con all'interno una SC_THREAD in loop.
- Transattore AMS-RTL: mudulo SystemC-ams che utilizza le porte sca_tdf::sca_de::sca_in per convertire i dati passati da AMS a RTL.
- Transattore RTL-AMS: mudulo SystemC-ams che utilizza le porte sca_tdf::sca_de::sca_in per convertire i dati passati da RTL a AMS.

La figura 7 mostra la struttura della piattaforma eterogenea.

IV. Risultati

Durante la fase di sviluppo del sistema, una volta ultimato, ogni componente è stato testato singolarmente per verificarne il corretto funzionamento. Ciò è stato possibile grazie ai testbench, capaci di fornire vari input controllando la correttezza dei risultati e misurando il tempo di simulazione. Il primo modulo testato è stato lo Xtea. I livelli TLM (UT, LT e AT4) hanno un tempo di esecuzione basso e i tempi sono molto vicini tra loro, differenze causate dalla schedulazione, mentre il livello RT ha una simulazione più lenta rispetto ai precedenti livelli.
Poi si è passati alla parte AMS, testata inizialmente come sistema AMS puro e poi assieme al controllore TLM e i transattori. In figura 8 è mostrata la curva di stabilizzazione dell'acqua.
Ed infine si è testata la piattaforma eterogenea. In figura 9 è mostrata la curva di stabilizzazione dell'acqua.

V. Conclusioni

Al termine del progetto ci si è resi conto dell'importanza del riuso e dell'utilizzo di diversi livelli di astrazione, hanno avuto un ruolo fondamentale nella riduzione del tempo di modellazione e di verifica. Inoltre ci si è resi conto dell'utilità del SystemC che ha reso il tutto più semplice grazie all'utilizzo di classi e moduli già implementati nelle librerie. Questo progetto ha messo quindi in evidenza che la combinazione di questi fattori permette lo sviluppo in tempi ristretti di un sistema funzionante e con risultati soddisfacenti.

Riferimenti bibliografici

[1] IEEE Computer Society, "IEEE Standard SystemC® Language Reference Manual," 2013.
[2] Accellera Systems Initiative, "Standard SystemC® AMS extensions 2.0 Language Reference Manual," 2016.

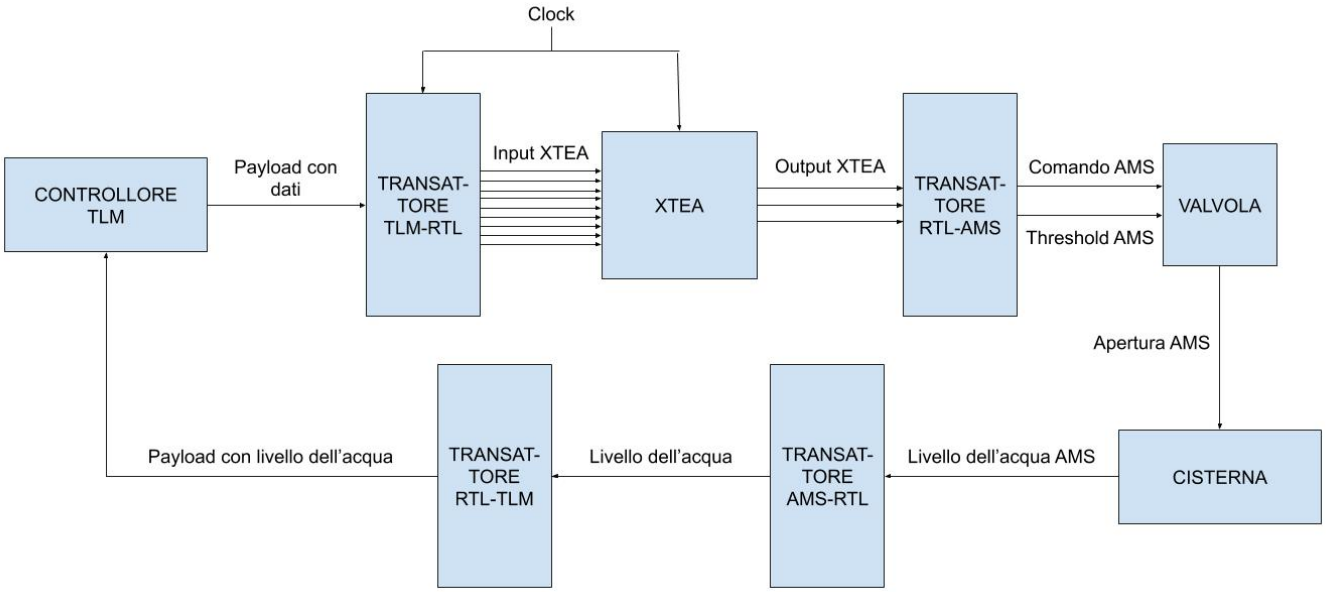

Figura 7: Schema dell'impianto finale (per leggibilità i nomi dei I/O del modulo Xtea sono stati omessi).

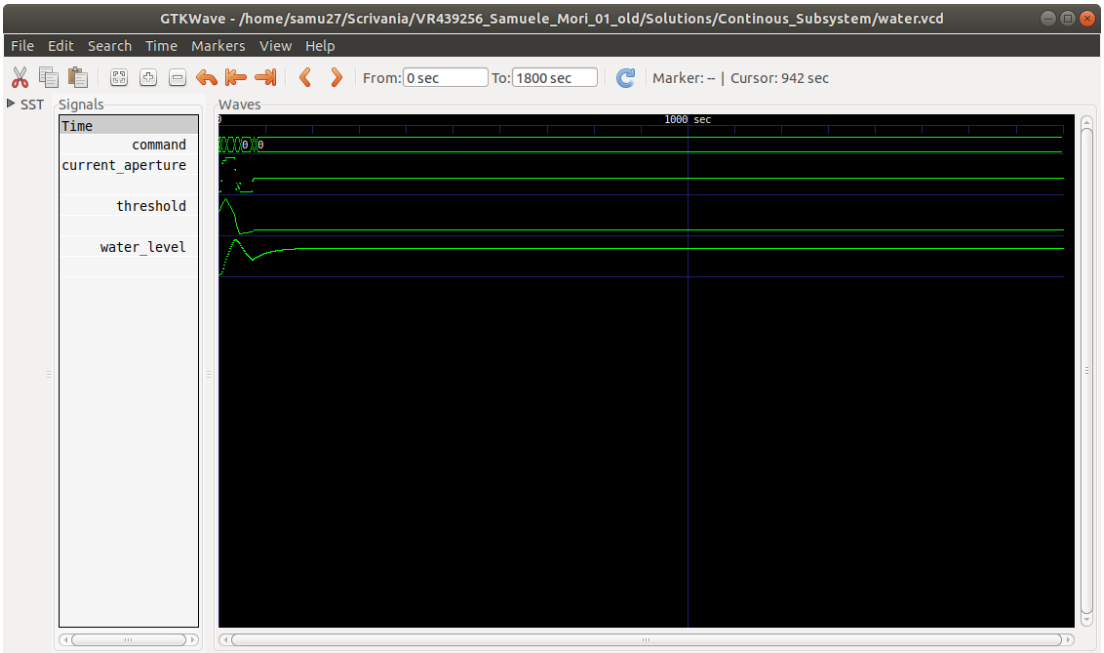

Figura 8: Continous Subsystem: livello dell'acqua.

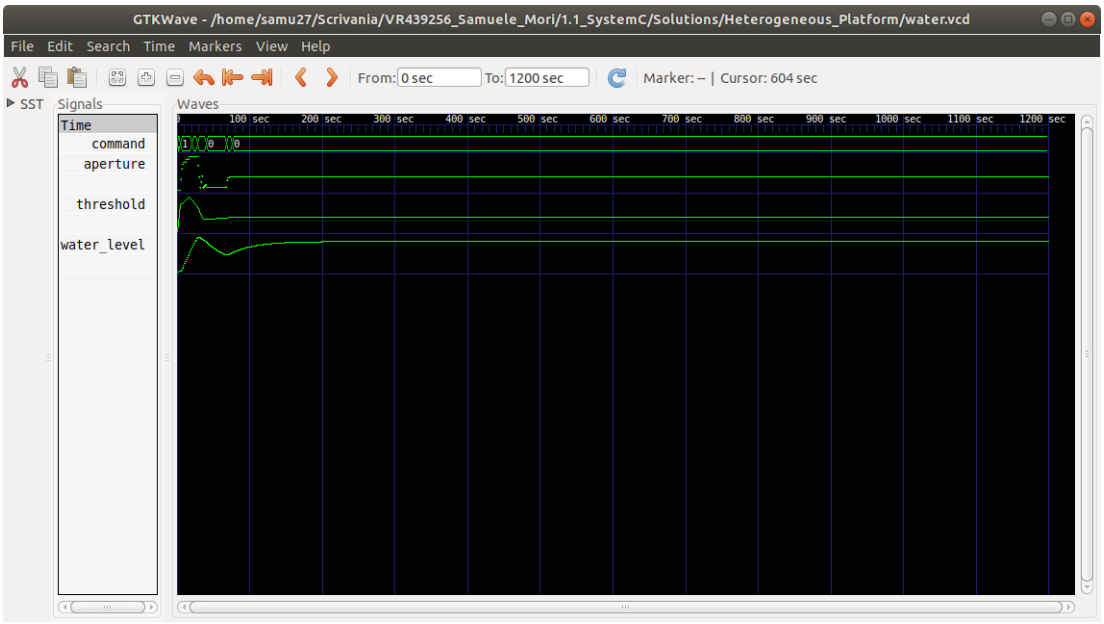

Figura 9: Heterogeneous Platform: livello dell'acqua.