Modellazione in SystemC di un sistema hardware/software per il controllo del livello dell'acqua in un serbatoio

Vladislav Bragoi - VR436747

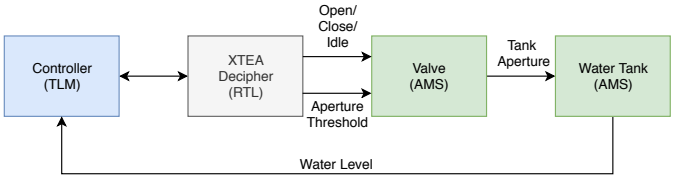


Figura 1: Organizzazione in moduli del sistema.

***Sommario*—Questo documento presenta l'implementazione ai fini didattici, di un sistema per il controllo del livello dell'acqua in un serbatoio, con sviluppo di moduli SystemC a vari livelli di descrizione hardware/software.**

I. Introduzione

Il sistema descritto in questo documento deve monitorare e controllare il livello dell'acqua in un serbatoio. La figura 1 ne mostra la struttura, organizzata nei seguenti moduli:

- Controller, si occupa di leggere il livello dell'acqua ricevuto dal serbatoio, e di inviare sulla base di questo un comando ai moduli successivi, in modo da controllare l'apertura della valvola sulla base di un limite massimo (anche questo calcolato e inviato dallo stesso controllore) per fare riempire il serbatoio. Da notare che questo modulo cifra tutte le informazioni prima di inviarle agli altri moduli.
- XTEA Decipher, si occupa di decifrare i messaggi ricevuti dal Controller secondo l'agoritmo eXtended TEA (interessante in questo caso poiché utilizza poche risorse hw, adattandosi perfettamente a questo tipo di sistema embedded), per poi inviarli direttamente al modulo che aziona la valvola.
- Valve, applica i comandi che gli sono stati inviati dal Controller, modificando l'apertura/chiusura della valvola per far riempire il serbatoio.
- Water Tank, è il sistema dinamico che si occupa del controllo dell'acqua sulla base della seguente funzione:

$$\dot{x} = 0.6 * a - 0.03 * x$$

dove a è l'apertura della valvola e x è il livello dell'acqua.

Nell'introduzione viene descritto in maniera astratta quello che poi viene dettagliato nel seguito del report. Una buona scaletta per l'introduzione può essere la seguente:

- Descrizione ad alto livello delle principali caratteristiche del sistema che si vuole modellare.
- Descrizione delle motivazioni principali per l'utilizzo delle tecnologie descritte nel corso. Qual'è il problema che si vuole risolvere?
- Descrizione dei passi utilizzati per arrivare all'implementazione finale. Descrivere la motivazione di ciascun passo. La descrizione dei passi dovrebbe formare la descrizione del flusso di lavoro svolto per completare l'assignment.
- Rapidissima descrizione dei risultati principali.

L'introduzione non dovrebbe andare oltre la metà della seconda colonna (nel caso a due colonne), o la prima pagina (nel caso a colonna singola): bisogna cercare di essere concisi (e chiari). Alla fine, l'introduzione è solo "chiacchiere": deve semplicemente rendere chiari quali sono gli obiettivi del lavoro (*e nel caso del corso, deve far capire a me che avete gli obiettivi chiari in testa*). Consiglio: l'introduzione (e spesso l'abstract) è l'ultima parte che viene completata.

II. Background

Il sistema è stato implementato interamente in C++, utilizzando la libreria SystemC, poiché una delle motivazioni principali all'utilizzo di questa libreria è quello di poter integrare e simulare i vari componenti assieme, sviluppati a diversi livelli di astrazione. In particolare, i livelli di astrazione utilizzati sono:

- SystemC RTL[1] (Register Transfer Level), per una modellazione a basso livello, e dunque più vicina all'hardware, con sviluppo dei moduli suddividendo la parte di logica di controllo, (*fsm*), dalla parte che esegue i calcoli e le operazioni aritmetiche/logiche (*datapath*).
- SystemC[2] TLM (Transaction-level Modeling), che permette di descrivere come i vari moduli interagiscono tra loro, grazie ai diversi stili messi a disposizione dalla libreria, ovvero TLM AT (approximately- timed) e TLM LT (loosely-timed) per una sincronizzazione basata su chiamate non bloccanti e/o basata sul concetto del temporal decoupling rispettivamente, per arrivare ad una sincronizzazione a chiamate bloccanti in cui non si tiene traccia del tempo di simulazione, e dunque molto più vicina ad una descrizione algoritmica, coincidente con lo stile TLM UT (untimed).
- SystemC AMS[3] (Analog/Mixed-Signal), che permette di modellare componenti analogiche e/o mixed-signal utilizzando diversi tipi di formalismi quali TDF (Timed Data Flow) per una modellazione non conservativa di sistemi a tempo discreto, LSF (Linear Signal FLow) per

una modellazione non conservativa di sistemi a tempo discreto e a tempo continuo (utilizzando solver automatici per risolvere le equazioni differenziali o alle differenze che descrivono il sistema), e ELN (Electrical Linear Network) per una modellazione conservativa di sistemi a tempo continuo.

III. Metodologia applicata

Le specifiche del progetto richiedevano di strutturare il sistema in moduli che possano essere implementati secondo i diversi livelli di astrazione messi a disposizione dalla libreria SystemC. Di seguito verranno illustrate nel dettaglio le caratteristiche di ciascun modulo:

A. Xtea [RTL/TLM]

Il modulo XTEA implementa l'agoritmo eXtended TEA, ideato da David Wheeler e Roger Needham al Cambridge Computer Laboratory. Il vantaggio principale di questo algoritmo è quello di avere una struttura semplice per poter essere impiegato in sistemi di limitate risorse hardware, e quindi il suo impiego si adatta perfettamente ad un sistema embedded semplice quale quello che stiamo modellando, per poter offrire un minimo livello di sicurezza. A partire dai sorgenti i n C/C++ è stato possibile ricavare la seguente struttura di base:

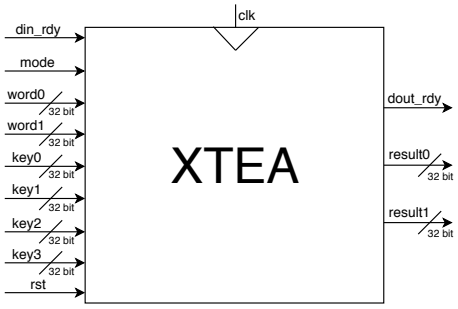


Figura 2: Rappresentazione del modulo XTEA

Di seguito vengono descritti i diversi livelli di astrazione in cui il modulo è stato implementato:

- XTEA RTL, descritto in SystemC RTL, è stato progettato suddividendo (secondo quanto detto precedentemente) la parte di controllo dalla parte relativa ai calcoli. In figura 3 vengono presentate le porte di ingresso e di uscita del modulo e i segnali che interconnettono l'FSM al Datapath. In particolare, l'FSM è sensibile alle variazioni sul segnale *din_rdy* e *status*, mentre utilizza *mode* e *counter* per decidere lo stato prossimo. Il Datapath invece è sensibile ai segnali di *reset* e *clock*, e utilizza *mode*, *word_0*, *word_1* per calcolare i valori di output *result_0* e *result_1*. Per quanto riguarda i segnali interni, nel datapath vengono definiti i seguenti:
  - *k*, segnale a 2 bit per memorizzare quale delle 4 chiavi in input utilizzare nel calcolo;
  - *key*, segnale a 32 bit per memorizzare temporaneamente la chiave da utilizzare;
  - *delta*, per memorizzare il valore del parametro delta;

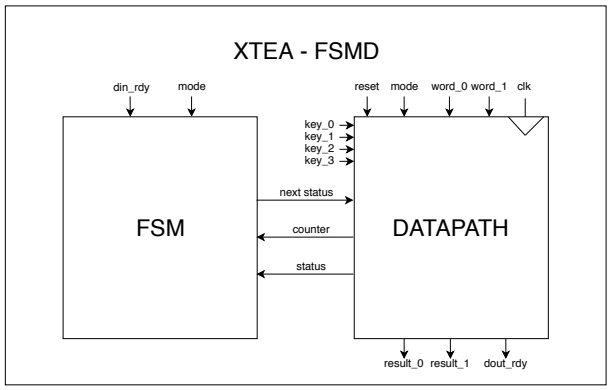


Figura 3: FSMD del modulo XTEA RTL

  - *sum*, segnale a 64 bit utilizzato per i calcoli ad ogni iterazione;
  - *counter*, segnale a 7 bit, utilizzato per le 64 iterazioni del ciclo per le due modalità (il motivo per cui è stato scelto di effettuare 64 iterazioni invece di 32 verrà chiarito più avanti). Sono sufficienti 6 bit per compiere 64 iterazioni ma poiché all'ultima iterazione il segnale viene ulteriormente incrementato di una unità, questo dev'essere a 7 bit per non perdere di informazione;
  - *v0* e *v1*, segnali di appoggio per i calcoli, a 32 bit.

  Nella descrizione del modulo secondo la EFSM[1] in figura 6 (in appendice), la parte di logica di controllo è rappresentata dalle transizioni e dalle condizioni sulle transizioni, mentre il Datapath è rappresentato dagli stati dell'automa. A partire dagli stati *ST_M0*, *ST_K*, *ST_CALC* e *ST_SUM* per la modalità di cifratura (e equivalentemente *ST_M1*, *ST_K*, *ST_CALC* e *ST_SUM* per la modalità di decifratura) è possibile vedere come sia stato scelto, a differenza della versione dell'algoritmo di C/C++, di effettuare un numero pari a 64 iterazioni per poter riutilizzare i blocchi che altrimenti sarebbero ripetuti per le due modalità. Inoltre, per quanto riguarda lo stato *ST_CALC* che tra tutti sembrerebbe essere quello a complessità più alta, è stato scelto di non spezzarlo in due stati in quanto le operazioni al suo interno risutano essere molto simili, e dunque un'eventuale ottimizzazione dell'area secondo gli algoritmi allo stato dell'arte ne ridurrebbe drasticamente sia la complessità, sia gli elementi hardware da impiegare.
- XTEA TLM è la versione di XTEA sviluppata in SystemC TLM (nelle relative varianti UT, LT e AT4). La struttura cha accomuna tutti e tre i moduli è la seguente: il *testbench* (che implementa l'interfaccia `tlm_bw_transport_if`) effettua la funzionalità dell'initiator, ovvero esegue le chiamate alla controparte *xtea* (che di conseguenza effettua la funzionalità del target, implementando l'interfaccia `tlm_fw_transport_if`), avviando transazioni per permettere lo scambio di messaggi, che in questo caso, corrispondono all'invio dei dati da cifrare/decifrare. In particolare, ad ogni transa-

[1] Extended Finite State Machine

	UT	LT	AT4	RTL
Real time	0,039s	0,045s	0,069s	1,025s
User time	0,009s	0,015s	0,016s	0,894s
System time	0,018s	0,018s	0,044s	0,052s

Tabella I: Statistiche di esecuzione di XTEA secondo i diversi livelli di astrazione

zione viene inviato un payload esteso, ovvero un payload con allegate le informazioni definite nella struttura `iostruct`, che racchiude gli input e gli output definiti nella rappresentazione in figura 2.

- TLM Aproximately Timed (4 phases), è la versione basata su chiamate non bloccanti. La sincronizzazione avviane secondo il classico protocollo di handshake a 4 fasi, in particolare il *testbench* invia i dati alla parte *xtea* mettendosi successivamente in attesa dell'acknowledgement (fasi `BEGIN_REQ`, `END_REQ`), *xtea* riceve la transazione, elabora i dati e ne salva il risultato, per poi successivamente ritornare l'acknowledgement alla controparte sbloccandone l'esecuzione (fasi `BEGIN_RESP`, `END_RESP`).
- Loosely Timed, è la versione basata su una sincronizzazione a chiamate bloccanti. Qui l'initiator (che, come detto sopra, è il *testbench*) chiama la funzione `b_transport` implementata nel target, aggiungendo alla transazione, oltre al payload per lo scambio dei messaggi, anche l'informazione relativa al tempo. Questa viene utilizzata per sfruttare il cosidetto Temporal Decoupling, ovvero per permettere ai processi (nel nostro caso il testbench) di continuare la propria esecuzione senza però incrementare il tempo di simulazione, per poi sincronizzarsi esplicitamente in punti definiti oppure allo scadere del quanto di tempo che gli è stato associato dallo scheduler, riducendone così il tempo totale della simulazione.
- TLM Untimed è una versione simile alla precedente, in cui non viene però specificata l'informazione relativa al tempo. Non richiede dunque una sincronizzazione esplicita, poichè questa viene garantita dalla chiamata bloccante alla `b_transport`, permettendo una classica sincronizzazione dei processi.

La differenza tra queste quattro versioni sta nel tempo di simulazione. In tabella I vengono presentate le statistiche di simulazione basate su un'esecuzione di 1000 iterazioni per modulo, le quali mettono in evidenza come la versione RTL richieda più tempo di simulazione rispetto alle altre, poichè è la versione di XTEA più precisa, ovvero quella che garantisce una sincronizzazione tra le parti più accurata. Gli altri moduli (TLM), essendo meno accurati in termini di sincronizzazione, impiegano meno tempo di simulazione, con una differenza che cala gradualmente con l'aumento del livello di astrazione.

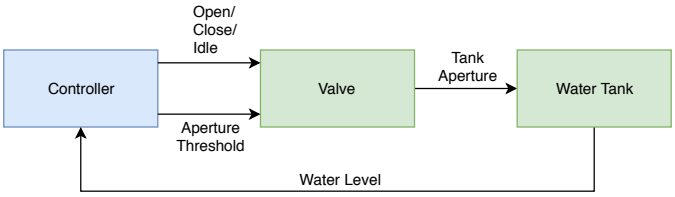


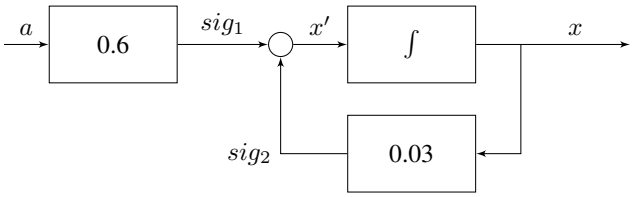
Figura 4: Struttura del watertank system

Water level	Command	Aperture threshold
$range(5, 8.8)$	IDLE	-
> 8.8	CLOSE	$t = t * 0.7$
< 5	OPEN	$t = t * 1.1$

Tabella II: Parametri che influenzano il controllo sulla valvola

B. Wwatertank System [AMS]

Water Tank System è un sistema per il controllo e il monitoraggio del livello dell'acqua di un serbatoio. La modellazione, che segue la struttura presentata in figura 4, ha richiesto l'utilizzo di SystemC AMS, poiché i componenti da modellare presentano tutti comportamenti a tempo discreto e/o a tempo continuo. In particolare, il Controller, descritto in SystemC AMS utilizzando il formalismo TDF per una modellazione a tempo discreto, deve leggere il livello dell'acqua fornitogli in input e deve poi verificare che stia dentro il range $[5 \dots 8.8]$. Inoltre, sulla base delle condizioni riportate in tabella II, il modulo deve periodicamente inviare al suo immediato vicino un comando di apertura o chiusura della valvola per fare riempire il serbatoio, e una threshold per indicare il livello di apertura massima che la valvola deve raggiungere. Dell'attuazione del comando si occuperà il modulo Valve, descritto anche questo secondo il formalismo TDF non conservativo per una modellazione a tempo discreto, mentre il modulo Water Tank è il componente che rileva il livello dell'acqua. Il suo comportamento rispecchia il seguente modello matematico, rappresentato sotto forma di schema a blocchi, e dunque si è scelto di descriverlo utilizzando il formalismo LSF non conservativo a tempo continuo, che aiuta nella risoluzione dell'equazione differenziale associata al modello grazie ai solver automatici che la libreria mette a disposizione:



Ai capi di questo modello sono stati posizionati (in input e in output) dei convertitori da TDF a LSF e da LSF a TDF per potersi interfacciare con gli altri moduli (descritti in SystemC AMS-TDF appunto) a tempo discreto. Inoltre, per evitare problemi dovuti al loop creato, è stato inserito un delay sulla porta water_level del Controller.

Per quanto riguarda la stabilizzazione dell'acqua, questa, come mostrato in figura 5, si stabilizza approsimativamente

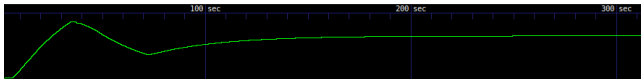


Figura 5: Stabilizzazione dell'acqua relativamente al Water Tank System (AMS) - Continous Subsystem

attorno ai 200 secondi.

C. Heterogeneous Platform

L'Heterogeneous platform, ovvero l'unione in un unico sistema di tutti i moduli (descritti ognuno secondo un proprio livello di astrazione) visti fin'ora e già presentato in parte nella sezione I, è caratterizzato, oltre che da questi moduli appunto, anche da ulteriori 4 componenti, necessari per poter simulare l'intero sistema: i transattori. Nella figura 7 è possibile vedere come questi componenti trasformano un segnale di interconnessione:

- da TLM a RTL, per i segnali che partono dal Controller (in TLM) e che vengono inviati al modulo Xtea Decipher (in RTL);
- da RTL a AMS, per i segnali che partono dal modulo Xtea Decipher e che entrano nel modulo Valve (in AMS);
- da AMS a RTL e nuovamente a TLM, per i segnali in uscita dal modulo Tank (in AMS) ed entranti nel modulo Controller (TLM);

Da notare che i componenti Valve Interface e Tank Interface mostrati in figura, sono interfacce RTL necessarie per poter far comunicare tra loro moduli scritti in TLM, con moduli scritti in AMS e viceversa. Il modulo Valve Interface inoltre, contiene al suo interno una parte di logica per riconoscere quando il modulo Xtea Decipher gli manda un comando oppure una threshold. Queste informazioni infatti, sono inviate in due momenti separati e servono per utilizzare lo stesso componente per decifrare due tipologie di informazione diverse che gli vengono passate dal Controller (tramite l'Xtea Transactor).

IV. Risultati

Qua vanno "messi i numeri". Questa sezione dovrebbe contenere i risultati della simulazione. La simulazione mostra che il sistema funziona correttamente? Come è stato provato? Che tipo di testbench sono stati utilizzati? Come è stato scomposto il sistema per verificarne la correttezza?

Per quanto riguarda le performance:

- cosa si può dire in merito ai tempi di simulazione dei modelli a diversi livelli di astrazione? Quali dati supportano questa affermazione? Come sono stati ottenuti i dati?
- Come cambia la velocità di simulazione per la parte digitale e la parte a tempo continuo? Come influiscono le frequenze scelte? E i livelli di astrazione? *etc.*

Questa sezione può contenere anche riflessioni personali sui risultati ottenuti. Importante: tutte le affermazioni devono essere supportate da numeri[2].

[2]Richard Feynman on Scientific Method (1964) - https://www.youtube.com/watch?v=OL6-x0modwY

V. Conclusioni

Le conclusioni dovrebbero riassumere in poche righe tutto ciò che è stato fatto. Un paio di righe descrivono i risultati osservati, in modo da introdurre poi la conclusione "vera e propria". Nel caso del corso, la "lezione da portare a casa" sarà quello che si è imparato svolgendo l'elaborato.

Riferimenti bibliografici

[1] Synopsys, "Describing synthesizable rtl in systemc."
[2] "Ieee standard for standard systemc language reference manual," *IEEE Std 1666-2011 (Revision of IEEE Std 1666-2005)*, pp. 1–638, Jan 2012.
[3] "Ieee standard for standard systemc(r) analog/mixed-signal extensions language reference manual," *IEEE Std 1666.1-2016*, pp. 1–236, April 2016.

Appendice

Se non avete abbastanza spazio, potete inserire le figure delle EFSM in una pagina extra, appendice. Un esempio di come potete fare solo le Figure **??**, **??**, **??**.

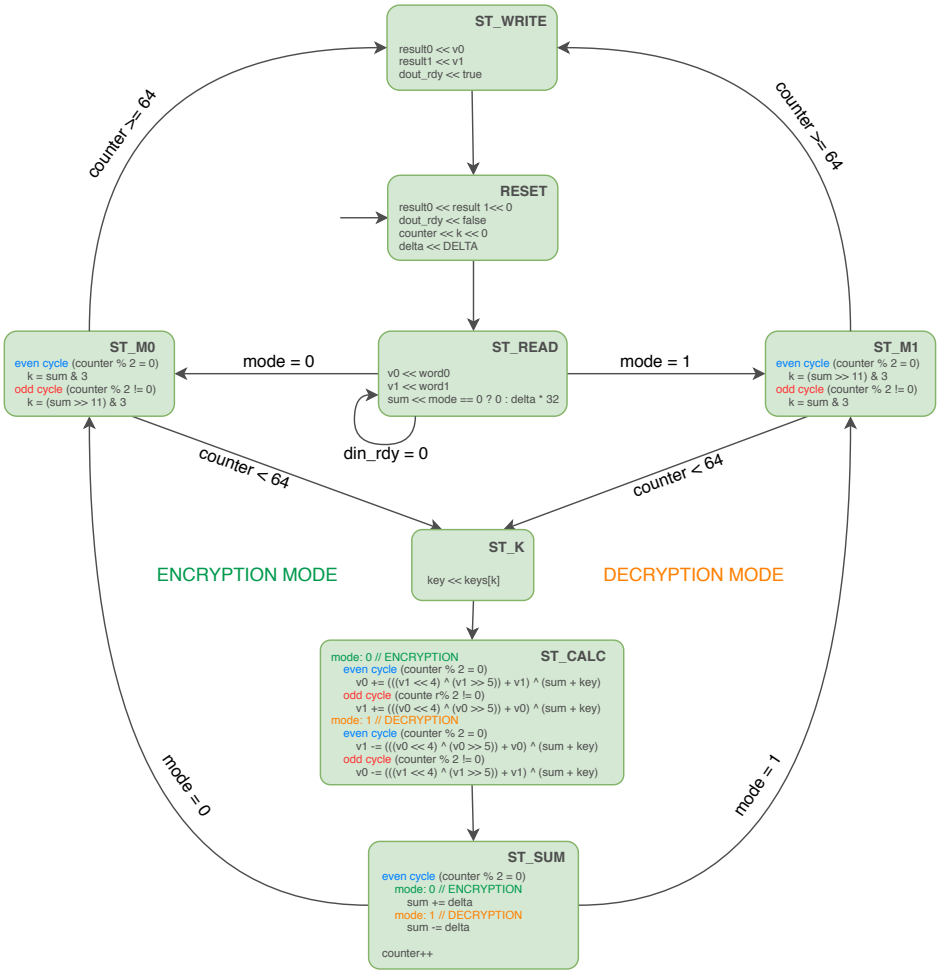


Figura 6: EFSM del modulo XTEA RTL

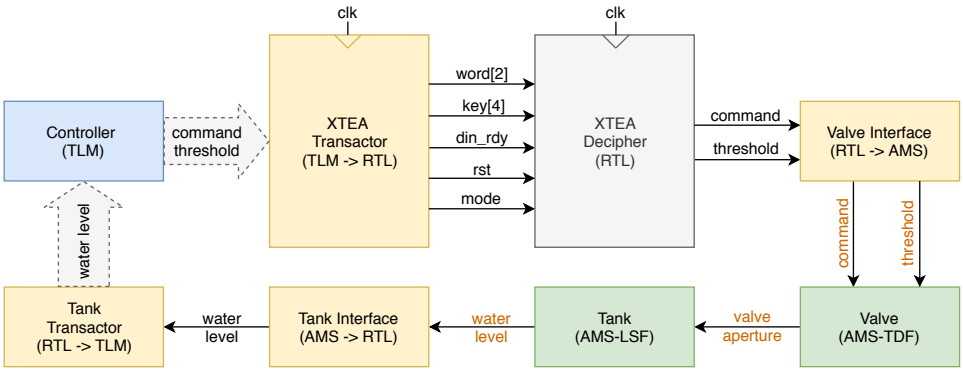


Figura 7: Struttura del heterogeneous platform