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

Vladislav Bragoi - VR436747

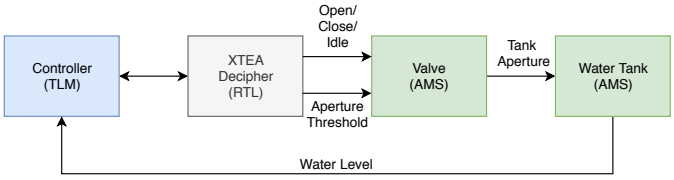


Figura 1: Organizzazione in moduli del sistema.

***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 (Water Tank), e di inviare sulla base di questo livello un comando al modulo Valve, attraverso l'XTEA Decipher, in modo da attuare il controllo sull'apertura della valvola, regolando di conseguenza il livello dell'acqua nel 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 svolge la funzione di attuatore, regolando i gradi di apertura della valvola.
- Valve, applica i comandi che gli vengono inviati dal Controller attraverso l'Xtea Decipher, e modifica 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.

Il sistema da progettare segue diverse implementazioni prima del suo completamento nella versione finale chiamata "Heterogeneous Platform". In particolare, occorre sviluppare i vari sottomoduli seguendo diversi stili di progettazione, con un'implementazione dei singoli a diversi livelli di astrazione.

Nella subdirectory HW_Subsystem vengono raccolte le diverse implementazioni del modulo XTEA, che hanno permesso di estrarre le principali caratteristiche della progettazione digitale e che vengono presentate nella sezione III-A, mentre nella subdirectory Continous_Subsystem viene presentata un'implementazione delle componenti analogiche (a tempo continuo e a tempo discreto) del sistema, e che verranno descritte nella sezione III-B.

Come sarà possibile verificare in seguito, l'implementazione modulare dei vari componenti ne permette sia una progettazione in SystemC (linguaggio di descrizione hw/sw di riferimento per questo progetto), sia la possibilità di simulare tutti i componenti eterogenei nel loro insieme.

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 ai due stili, 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:

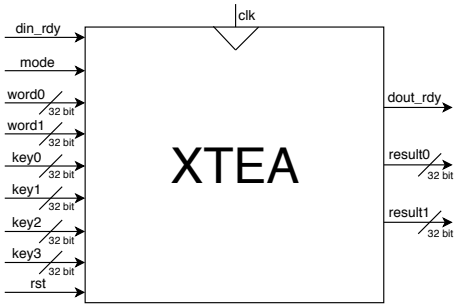


Figura 2: Struttura generale del modulo XTEA

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 (in base a 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;
  - *sum*, segnale a 64 bit utilizzato per i calcoli ad ogni iterazione;

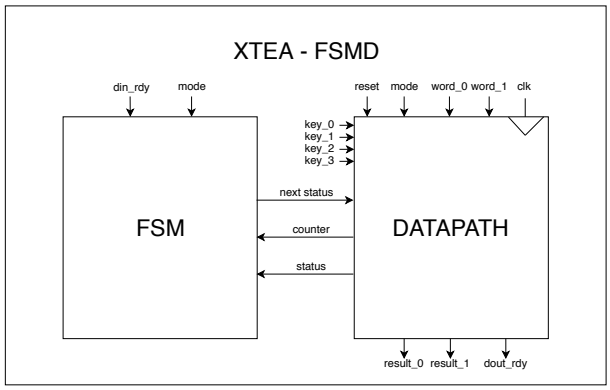


Figura 3: FSMD del modulo XTEA RTL

  - *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 9 (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 transazione viene inviato un payload esteso, ovvero un payload con allegate le informazioni definite nella struttura

[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

`iostruct`, che racchiude gli input e gli output definiti nella struttura generica del modulo in figura 2.
Di seguito vengono presentate le diverse versioni, dalla più specifica alla più astratta:

- 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.

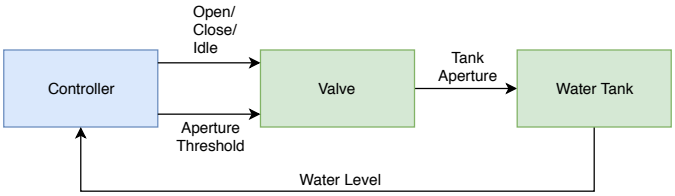


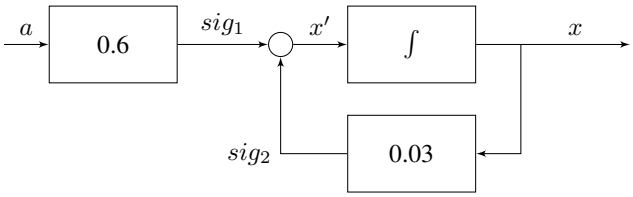
Figura 4: Struttura del watertank system

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

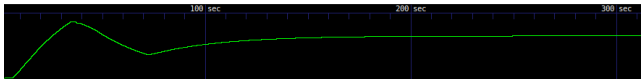


Figura 5: Stabilizzazione dell'acqua relativamente al Water Tank System (AMS) - Continous Subsystem

attorno ai 200 secondi.

C. Heterogeneous Platform

L'Heterogeneous platform, ovvero l'unione in un unico sistema di tutti i moduli visti fin'ora (descritti ognuno secondo un proprio livello di astrazione) e già presentato in parte nella sezione I, è costituito da ulteriori 4 componenti, necessari per poterne simulare tutto l'insieme: i transattori. In figura 6 è possibile vedere come questi nuovi componenti trasformano un segnale di interconnessione tra i singoli moduli:

- 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 (descritto in TLM);

Da notare che i segnali di colore nero in figura sono segnali RTL, mentre quelli di colore arancio sono segnali AMS. Invece i segnali rappresentati dalle frecce tratteggiate rappresentano transazioni TLM. Inoltre, i componenti Valve Interface e Tank Interface sono interfacce RTL necessarie per poter far comunicare tra loro moduli scritti in TLM, con moduli scritti in AMS e viceversa, mentre il modulo Valve Interface contiene al suo interno una parte di logica per riconoscere quando l'Xtea Decipher gli invia un comando oppure una threshold. Queste informazioni infatti, sono inviate in due momenti separati e servono per poter utilizzare lo stesso componente in modo da decifrare due tipologie di informazione diverse che gli vengono passate. Infine, il Controller essendo descritto in SystemC TLM, è l'initiator sia del modulo Tank Transactor, sia del modulo Xtea Transactor (che sono a loro volta di tipo target), poichè tale configurazione permette di gestire meglio la sincronizzazione del loop di controllo direttamente nel modulo Controller.

IV. Risultati

La corretta funzionalità del sistema è stata testata a partire dai singoli moduli. Questi infatti sono stati sottoposti a diversi testbench per verificarne il comportamento individuale prima di poter essere integrato con gli altri moduli. Inoltre, in alcuni casi è stato necessario utilizzare anche il debugger, come ad esempio per la versione RTL di XTEA che ha permesso di risolvere alcuni dei problemi legati alla sincronizzazione e scrittura sui vari segnali. Successivamente è stata testata la comunicazione tramite transazioni TLM, ad esempio per la comunicazione tra Controller e Xtea Transactor.

Infine, il sistema eterogeneo è stato sviluppato seguendone il flusso di esecuzione utilizzando i file di tracing dei segnali, che permettono di tracciare l'intero flusso rispettando però i reali tempi di esecuzione dell'intero sistema. La figura 8 riporta infatti i tempi relativi all'esecuzione di un ciclo di decifratura del modulo XTEA RTL, che impiega - come ben visibile in figura - poco più di 2 millisecondi. Mettendo in relazione questo risultato con quello ottenuto dalla stabilizzazione del livello dell'acqua in figura 7, è possibile vedere infatti, oltre all'evidente stabilizzazione del livello ad un valore pari a 8.5 c.a., che l'esecuzione modulo RTL è molto più veloce rispetto agli altri moduli del sistema (questo perché il modulo lavora alla frequenza del clock essendo il componente descritto a più basso livello di astrazione, quindi più vicino all'hardware).

In riferimento ai tempi di simulazione invece, poiché per una co-simulazione di diversi modelli descritti a diversi livelli di astrazione è richiesta una maggiore complessità di sincronizzazione, l'esecuzione totale dell'intero sistema risulta essere piuttosto lenta. Questo perché occorre sincronizzare i moduli TLM che sono meno accurati e che permettono però una simulazione più veloce, con il modulo XTEA RTL, che, come già visto in tabella I richiede più tempo di simulazione dovuto alla sua maggior precisione (poiché viene simulato il comportamento del modulo ad ogni ciclo di clock).

V. Conclusioni

Il riuso dei vari componenti per ridurre il cosidetto TTM (Time to Market) permette di costruire in breve tempo un prototipo da simulare per poter studiare la fattibilità del progetto nel suo complesso. In particolare, grazie all'utilizzo di SystemC è stato possibile modellare sia componenti analogiche (a tempo continuo e a tempo discreto), sia componenti digitali ed eseguirne una simulazione dell'intero sistema. Questo vantaggio si è mostrato rilevante considerando un'eventuale progettazione classica del sistema, in cui si sarebbe dovuto suddividere la parte di progettazione hw da quella sw (che ovviamente non sarebbe stata altrettanto fattibile).

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.

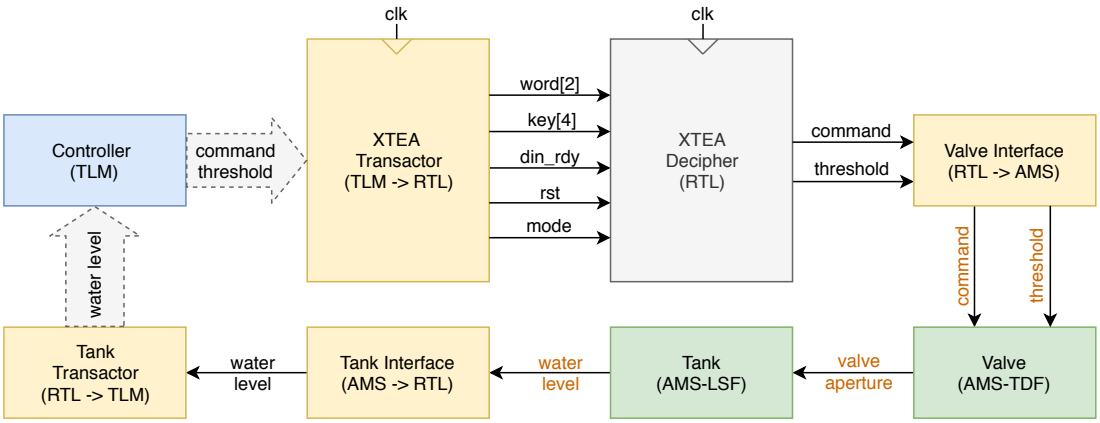


Figura 6: Struttura del heterogeneous platform

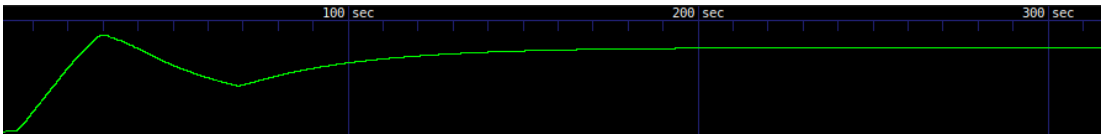


Figura 7: Stabilizzazione dell'acqua del sistema eterogeneo

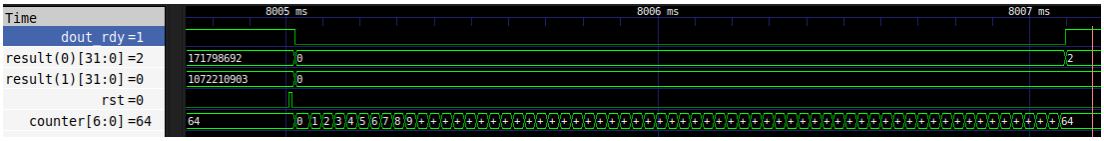


Figura 8: Valori dei segnali nel modulo XTEA RTL

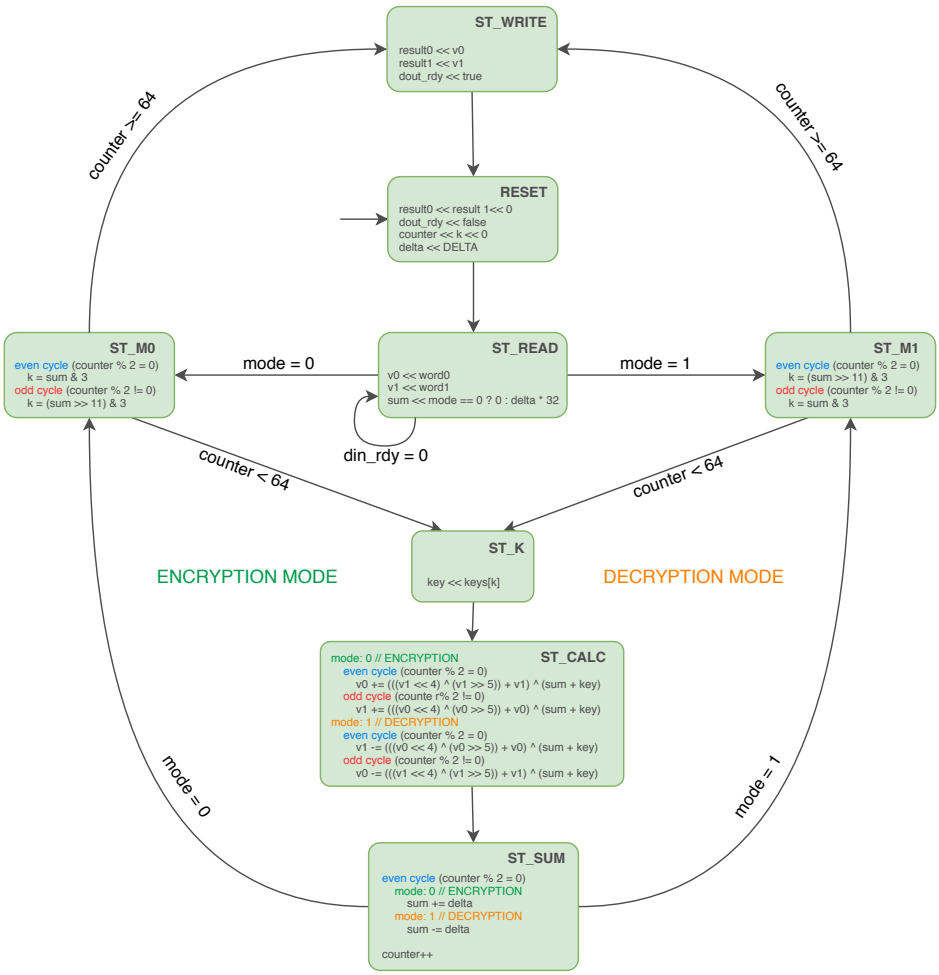


Figura 9: EFSM del modulo XTEA RTL