Modellazione in SystemC del sistema crittografico XTEA

Mirko Albanese - VR431583

***Sommario*—Questo documento descrive la modellazione effettuata mediante SystemC a livello RTL, TLM(UT, LT, AT4), AMS e a sistema eterogeneo (AMS+TLM+RTL) del sistema crittografico XTEA.**

I. Introduzione

Il progetto consiste nella realizzazione a livello RTL, TLM, AMS di un sistema crittografico chiamato XTEA (*eXtended Tiny Encryption Algorithm*). Il cifrario dovrà essere implementato nelle versioni:

- RTL: Rapppresentato da due processi, FSM e il relativo datapath.
- TLM *untimed*: Implementazione attraverso le classi di TLM, senza informazioni temporalli di simulazione.
- TLM *loosely timed*: Simile a quello precedente, con la differenza che tiene parzialmente conto dell'informazione.
- TLM *approximately timed 4 phased*: In questa versione vi sono chiamate TLM non bloccanti, ogni transazione attraversa quattro fasi e richiede una sincronizzazione.

Inoltre viene richiesto di implementare un sistema composto da un controllore, una valvola e da un serbatoio d'acqua. Il controllore riceve il livello dell'acqua da parte del serbatoio e invia comandi alla valvola per aprire e chiudere il rubinetto. Questo sistema dovrà essere implementato in SystemC AMS. Infine viene modificato il sistema modellato in SystemC AMS implementando un sistema eterogeneo dove il controllore viene implementato in TLM mandando comandi criptati mediante XTEA alla valvola e riceve il livello dell'acqua dal serbatoio, tali moduli AMS vengono interfacciati da dei transattori, infine viene posto tra la valvola e il controllore un decifratore XTEA in RTL.

II. Modellazione RTL

La versione RTL del cifratore viene implementata a partire dal codice sorgente fornito. L'implementazione dell'interfaccia RTL viene illustrata nella figura 1.

Le porte sono le seguenti:

- Input
  - Clock
  - Mode: *Comando da eseguire sul modulo (valore alto = Decodifica, valore basso = codifica)*
  - Input ready: *Permette la lettura dei dati in input*
  - Reset: *Permette di portare il modulo allo stato iniziale*
  - Word 1, Word 2: *Dati da processare*
  - Key 0, Key 1, Key 2, Key 3: *Chiavi da utilizzare*
- Output
  - Out ready: *Computazione terminata*

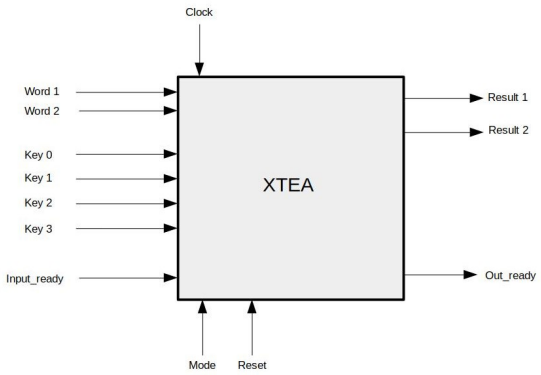

Figura 1. Interfaccia XTEA

  - Result 1, Result 2: *Risultato della computazione*

Viene diviso il modulo in due processi separati: *fsm* e *datapath*. Il primo aggiorna la variabile di stato e scandisce l'avanzamento della computazione; la seconda si occupa di elaborare espressioni aritmetiche, effettuare il reset del modulo e scrivere i valori nelle porte di output. La struttura del *FSM* e del *Datapth* è illustrata nella figura 2, dove i cerchi sono gli stati, le etichette sugli archi rappresentano le condizioni con i quali viene cambiato lo stato e i rettangoli sono i calcoli svolti dal datapath.

Il sistema viene testato da un *Testbench* dove inizializza il modulo fornendo i dati casuali in input testandolo un totale di 128 volte e confrontando i risultati in uscita. Il testbench asseconda il processo di elaborazione fornendo al modulo i valori delle porte di input come il comando di *encoding* e *deconding*, i dati da processare, il bit di reset.

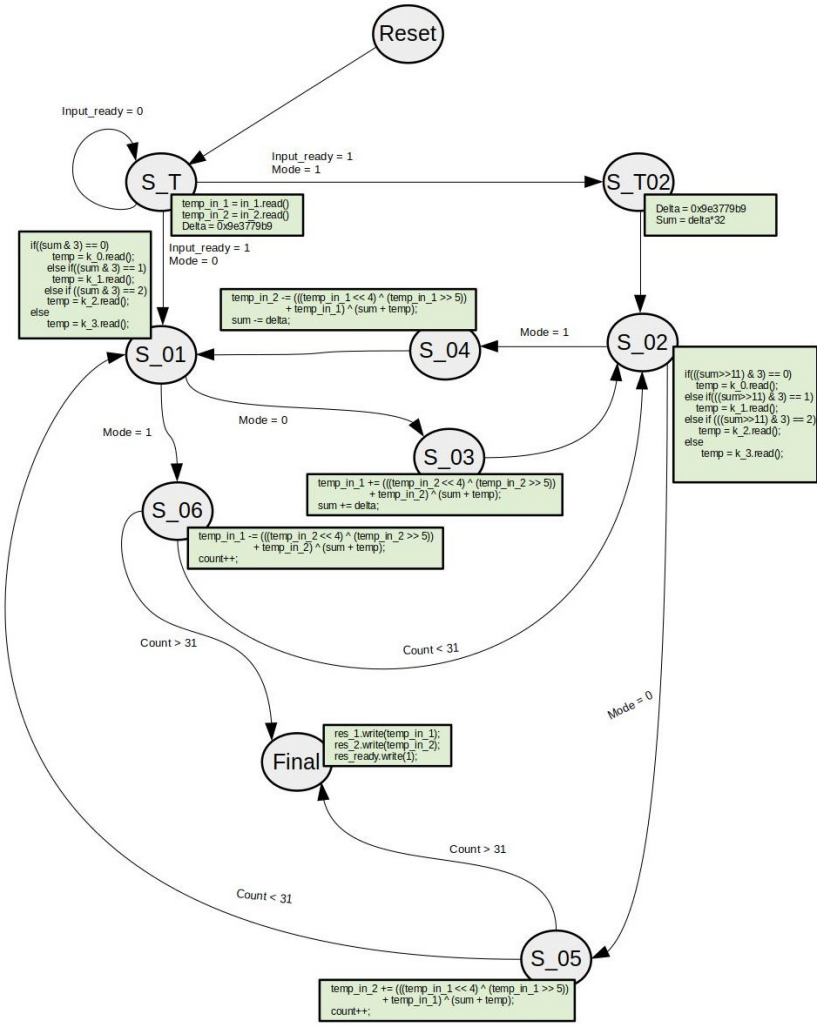

Figura 2. Diagramma degli stati. Le transizioni sono comandate dalla FSM. In verde sono presenti le elaborazioni fatte dal datapath.

III. Modellazione TLM

Di seguito le versioni del cifratore vengono implementate utilizzando il framework TLM, sviluppato al di sopra di SystemC, dove ogni modulo TLM è un modulo SystemC che eredita ulteriori interfacce, tali, possono essere di due tipi: *initiator* o *target*. I moduli comunicano tra loro mediante porte, dette *socket*, mediante transizioni, scambiano messaggi. Lo scambio dei messaggi si basa su una struttura dati `tlm : : tlm_generic_payload`, dove al suo interno contiene un puntatore ad un oggetto definito dall'utente che rappresenta il dato da scambiare. Nel nostro caso è un istanza di `struct payloadStruct`, ed ha al suo interno i dati da elaborare, le chiavi e il comando da analizzare.

A. Modellazione TLM Untimed

In questa modellazione il nucleo della comunicazione si basa sulla chiamata `be_transport` che prende in input il *carico*, ovvero il `tlm_generic_payload` e una variabile che rappresenta il tempo. Tale chiamata viene effettuata dall'interfaccia *initiator*, ed è bloccante.

Il payload, in questo tipo di modello, è solo di scrittura, ovvero il *testbench* (initiator) chiama la `b_transport` e il modulo XTEA (target) effettua l'operazione di codifica e di decodifica, infine, il *testbench* controlla il risultato.

B. Modellazione TLM Loosely Timed

In questa versione viene parzialmente introdotta la gestione dell'informazione temporale. Il *target* aggiornerà il tempo in esecuzione sommando il tempo dell'operazione richiesta dalla transazione.
Viene introdotta la sincronizzazione tra *initiator* e *target* mediante il *quantum keeper*, il metodo *sync* permette allo scheduler di effetuare il controllo della sincronia.

C. Modellazione TLM Approximatively Timed (AT4)

Questa modellazione viene implementata mediante una sequenza di chiamate non bloccanti `nb_transport` dove ognuna delle quali determina una *fase* della transazione. In questo approccio le fasi sono quattro: `BEGIN_REQ`, `END_REQ`, `BEGIN_RESP`, `END_RESP`. L'*initiator* inizia la comunicazione con la fase `BEGIN_REQ` e il *target* risponde alla transazione

mediante la fase `END_REQ`, nel frattepo vengono svolti i calcoli, una volta terminati il *target* chiama la procedura con la fase `BEGIN_RESP` e l'*initiator* chiude la transazione mediante la fase `END_RESP`.

IV. Confronto delle Prestazioni

Procediamo a fare un confronto delle prestazioni delle quattro implementazioni. Ogni volta le chiavi vengono inizializzate una volta sola, poi vengono criptate e decriptate 100'000 word generate casualmente. I risultati sono visibili nella tabella seguente.

	UT	LT	AT4	RTL
Real Time	7,923s	6,548s	4,039	60,648s
User Time	2,109	2,284s	1,735	53,181
System Time	2,319	1,738	0,993	4.098s

Possiamo concludere che le peggiori prestazioni le otteniamo con il modello RTL, essendo il più preciso e accurato.

V. Modellazione AMS

Il seguente sistema è composto da un serbatoio pieno d'acqua, una valvola e da un controllore. La valvolaregola l'apertura della tanica e il controllore monitora il livello dell'acqua all'interno del serbatoio e invia comandi alla valvola di conseguenza.

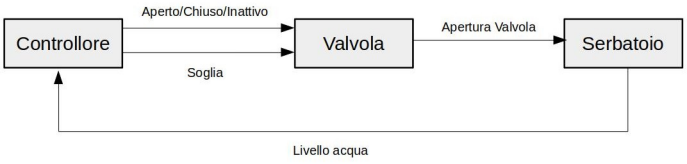

Figura 3. Schema a blocchi

La tanica mantiene al suo interno l'acqua, il suo livello di riempimento $x(t)$ è dato dalla seguente funzione:

$$\begin{cases} \dot{x} = 0.6a - 0.03x \\ x(0) = 0 \end{cases}$$

Occorre che il livello dell'acqua $x(t)$, da specifica sia compreso tra 5.0 e 8.8. La valvola aggiorna il livello di apertura a a seconda del comando che riceve dal controllore. Tale livello di apertura $a(t)$ deve avere un valore minimo pari a 0 e un valore massimo pari a T:

$$0 \leq a(t) \leq T$$

dove, T soglia in input dal controllore. Il valore iniziale si suppone $a(0) = 0$.
Il controllore moitora il livello dell'acqua e comanda la valvola di conseguenza. Il comandi generati sono: `APERTO`, `CHIUSO` e `INATTIVO`. Il valore di soglia viene aggiornato secondo i valori mostrati nella Tabella I. Il suo valore iniziale è $T(0) = 0.7$.

Segnale	Min lev	Max lev	Molt. threshold	Molt. a
`OPEN`	0.0	5.0	1.1x	0.25x
`IDLE`	5.0	8.8	1.0x	0.00x
`CLOSE`	8.8	∞	0.7x	-0.25x

Tabella I
Comandi, range di livello di acqua, moltiplicatore per l'aggiornamento della soglia, moltiplicatore per l'aggiornamento del livello di apertura

A. Valvova

La valvola è implementata tramite il modello discreto AMS-TDF, definita dalla classe `SCA_TDF_MODULE`. Il suo compito è quello di aggiornare il valore dell'apertura a seconda del comando ricevuto. All'interno del metodo `processing` vi è racchiusa la logica del componente.

B. Controllore

Il controllore è implementato tramite il modello discreto AMS-TDF, con il compito di ricevere il livello dell'acqua da parte del serbatoio e generare i comandi e la soglia da passare alla valvola. Viene definita mediante la classe `SCA_TDF_MODULE` e come per la valvola la logica del componente è racchiusa all'interno del metodo `processing`. Inoltre, viene impostato un ritardo sulla porta di input per poter definire un ordine di esecuzione, esso viene settato nel metodo `set_attributes`.

C. Serbatoio

Il serbatoio è implementato tramite il modello AMS-LSF. Il sistema è composto da vari elementi mostrati nello schema a blocchi in figura 4 e in figura 5.

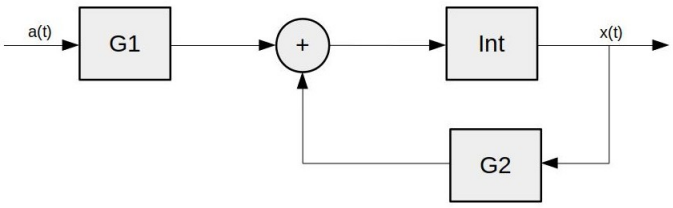

Figura 4. Serbatoio. Schema ad alto livello

Di seguito vengono illustrati i componenti utilizzati in LSF. I componenti utilizzati sono:

- `sca_tdf::sca_in` : Definisce la porta di input;
- `sca_lsf::sca_tdf::sca_source` : Converte il segnale TDF in LSF;
- `sca_lsf::sca_gain` : Guadagno di apertura;
- `sca_lsf::sca_add` : Sommatore;
- `sca_lsf::sca_gain` : Guadagno feedback;
- `sca_lsf::scal_integ` : Integratore;
- `sca_lsf::sca_tdf::sca_sink` : Converte il segnale LSF in TDF;
- `sca_tdf::sca_out` : Definisce la porta di output;

D. Risultato AMS

Si noti dalla figura 6 che dopo la prima sovraelongazione il livello dell'acqua si stabilizza.

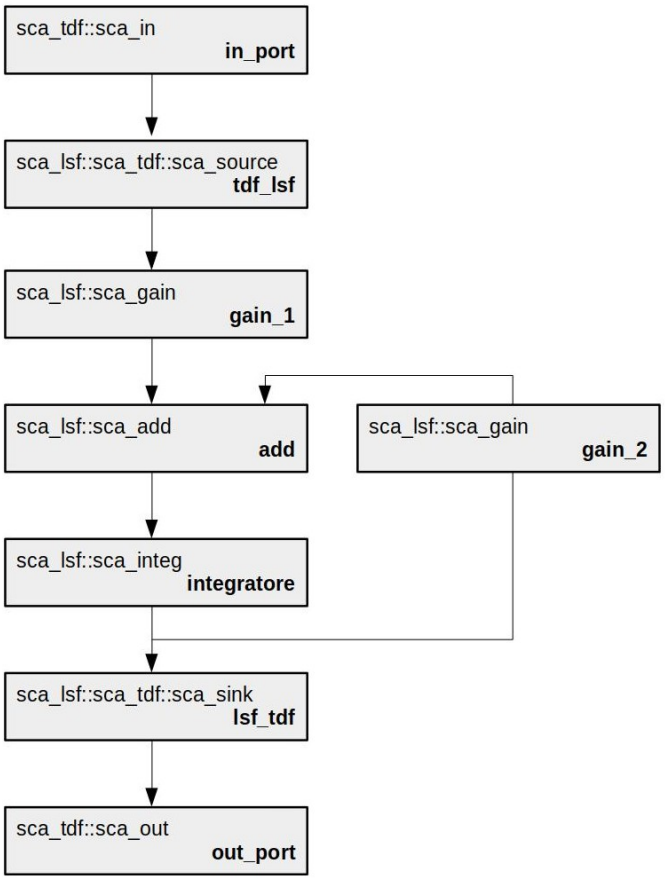

Figura 5. Serbatoio. Componenti LSF

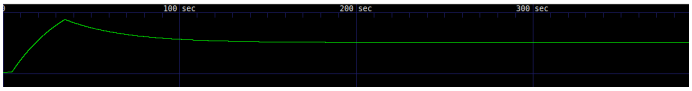

Figura 6. Livello del'acqua

E. Modellazione Heterogeneous Platform

La piattaforma eterogenea contiene componenti scritti con diverssi formalismi e connessi tutti insieme. Essi sono:

- Controllore: Implementato come TLM *untimed*;
- Valvola: Implementato dome modulo AMS-TDF;
- Serbatoio: Implementato come modulo AMS-LSF;
- Decrifratore XTEA: Implementato come modulo RTL;

Inoltre vengono implementati i vari transattori che permettono di far comunicare il sistema *valvola-serbatoio* con l'ambiente TLM e il transattore che permette di far comunicare *controllore*, *decifratore* e *valvola-serbatoio*.

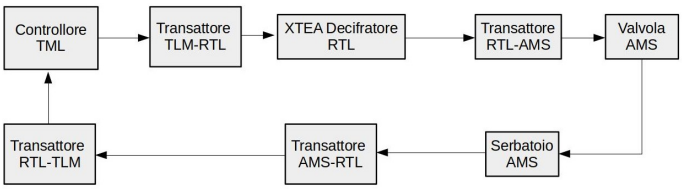

Figura 7. Diagramma a blocchi

Lo schema complessivo della piattaforma eterogenea è raffigurato in figura 7.

F. Risultato Heterogeneous Platform

Di seguito viene illustrato l'andamento del livello dell'acqua e la sua relativa stabilizzazione.

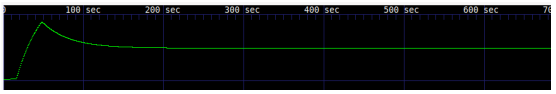

Figura 8. Livello dell'acqua