Department of Electronics and Telecommunications
Master degree in Electronic Engineering

Progetto di Sistemi Digitali Integrati
CRC-16/CCITT-FALSE

Autore:

Roberta Leo, Manuel Cataldo, Lorenzo Leli

5 febbraio 2024

Indice

1 CRC

1.1 Descrizione

La trattazione seguente descrive come è stato progettato e verificato un blocco hardware in grado di calcolare il CRC. In particolare, il blocco è in grado di calcolare il CRC di una sequenza arbitraria di parole di 16 bit tramite lo standard **CRC-16/CCITT-FALSE**. La comprensione delle operazioni da svolgere per effettuare il calcolatore del CRC e delle caratteristiche peculiari dello standard adottato sono state ricavate tramite ricerche in rete di cui nella sezione Fonti sono presenti i link degli articoli utilizzati. Da essi è emerso che il polinomio caratteristico da utilizzare è $x^{16} + x^{12} + x^5 + 1$ che in numero esadecimale corrisponde a $0x1021$ e che il valore di inizializzazione è $0xFFFF$.

1.2 Descrizine Datapath

Il blocco CRC si interfaccia con l'esterno con sei segnali: DIN (in input su 16 bit), DOUT (in output su 16 bit), ADDRESS (in input su 8 bit) e i segnali RD, WR e RST_SW (tutti su 1 bit in ingresso). I prima cinque elencati sono i segnali con cui il blocco CRC comunica con il blocco SPI che a sua volta manderàriceverà i dati aldal master. Il segnale RST_SW, invece, è un segnale di reset asincrono presente direttamente sullo slave. Nel nostro progetto è stato associato allo SWITCH 0 della scheda VirtLab in modo che sia attivo quando lo switch è sul 1 logico. Il blocco è stato realizzato utilizzando sei registri sparsi da 16 bit. E' possibile effettuare la lettura dai registri 0, 1, 2 e 3 in qualsiasi momento, quindi anche mentre si sta effettuando il calcolo del CRC. Questo è possibile dal momento che la lettura viene richiesta direttamente dal blocco SPI senza dover passare dalla control unit del blocco CRC. Non è invece possibile leggere il valore del registro 4 e 5 tramite il blocco SPI. Per quando riguarda la scrittura, è sempre il modulo SPI che scrive direttamente nei registri ma in questo caso solo due di essi sono accessibili. Si tratta del registro 0 in cui va inserito il nuovo dato su cui va effettuato il CRC e del registro 2 che serve per poter forzare il reset del CRC. In tutti gli altri registri non è possibile scrivere dall'esterno.
Il blocco CRC è composto dai seguenti blocchi:

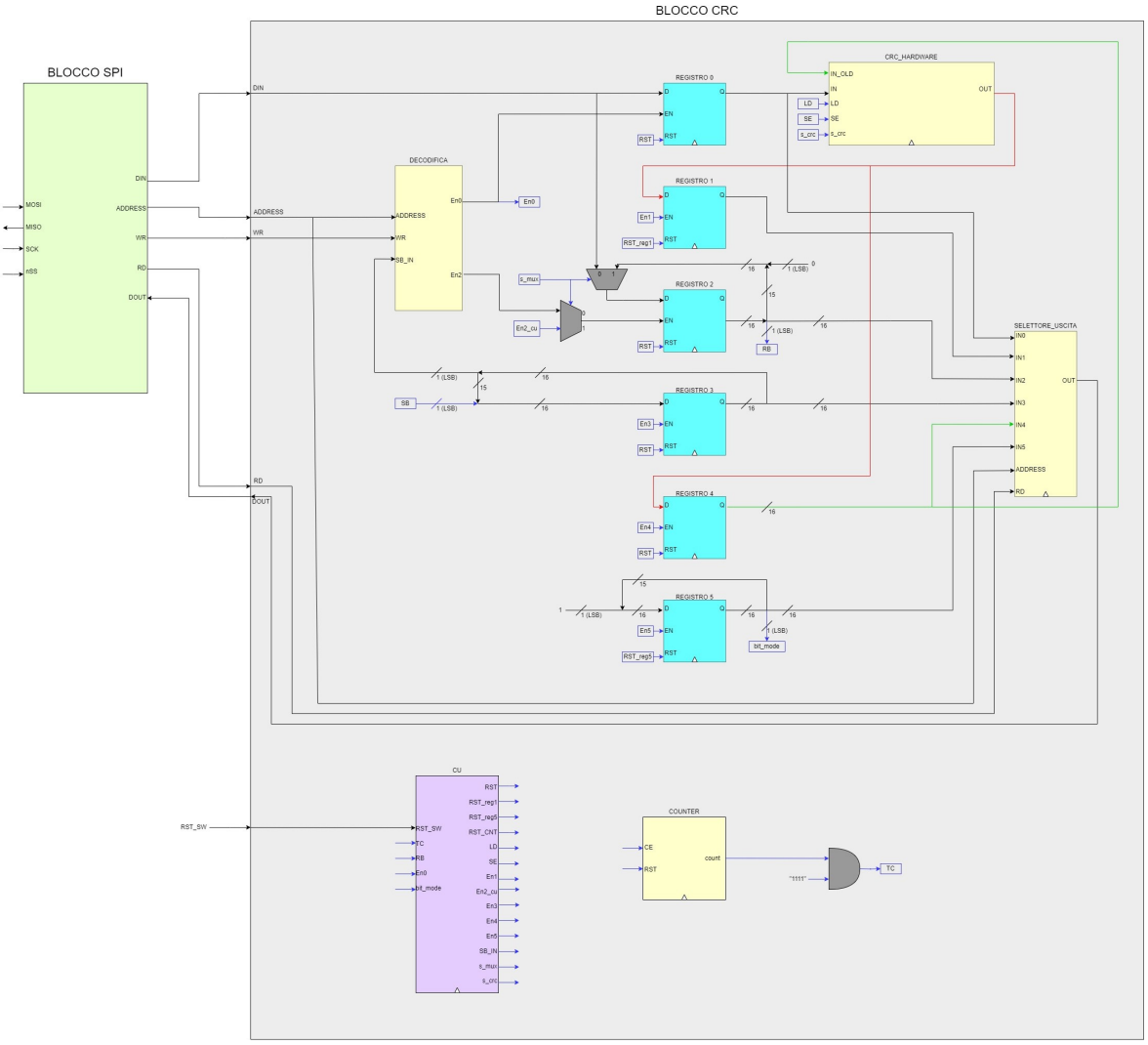


Figura 1: Datapath

1. DECODIFICA: questo blocco combinatorio serve per fare in modo che il blocco SPI possa scrivere direttamente nel registro 0 e nel registro 2 nel caso siano soddisfatte alune specifiche condizioni. Per entrambi i registri è necessario che i segnali WR e SB_IN siano attivi e nel caso in cui ADDRESS ha il valore di 0x00 il segnale DIN viene scritto nel registro 0 mentre se ADDRESS ha il valore di 0x02 il segnale DIN viene scritto nel registro 2. Il significato del segnale SB_IN è descritto nella spiegazione del registro 3.

2. REGISTRO 0: questo registro da 16 bit è accessibile all'indirizzo 0x00 dell'interfaccia SPI. Esso contiene il dato in inviato al blocco CRC e su cui andrà effettuato il calcolo del CRC. Quando si effettua il reset del registro, esso viene inizializzato al valore di 0x0000. La scrittura in questo registro è comandata unicamente dal blocco SPI.

3. REGISTRO 1: questo registro da 16 bit è accessibile all'indirizzo 0x01 dell'interfaccia SPI. Esso contiene il risultato del calcolo del CRC corrente. Quando si effettua il reset del registro, esso viene inizializzato al valore di 0xFFFF. La scrittura in questo registro è comandata unicamente dalla control unit del blocco CRC.

4. REGISTRO 2: questo registro da 16 bit è accessibile all'indirizzo 0x02 dell'interfaccia SPI. La scrittura di un '1' nel bit meno significativo causa il reset del CRC. Questo significa che nel registro 1 sarà contenuto il valore di inizializzazione del CRC (ovvero 0xFFFF). Quando si effettua il reset del registro, esso viene inizializzato al valore di 0x0000. La scrittura può essere effettuata sia dalla control unit del blocco CRC sia dal blocco SPI.

5. REGISTRO 3:questo registro da 16 bit è accessibile all'indirizzo 0x03 dell'interfaccia SPI. Il bit meno significativo viene definito SB_IN (status bit ingresso) ed è un flag che permette di sapere quando il blocco CRC ha terminato il calcolo di un dato e quindi è pronto a riceverne un nuovo dato dall'SPI. In questo caso il bit va posto a '1' mentre se il calcolo del CRC è ancora in svolgimento il bit deve essere posto a '0'. Quando si effettua il reset del registro, esso viene inizializzato al valore di 0x0001. La scrittura in questo registro è comandata unicamente dalla control unit del blocco CRC.

6. REGISTRO 4: questo registro da 16 bit serve come registro di appoggio per salvare il valore intermedio del calcolo del CRC che verrà poi utilizzato in caso di calcolo del CRC di sequenze di parole. La scrittura in questo registro è comandata unicamente dalla control unit del blocco CRC.

7. REGISTRO 5: il bit meno significativo di questo registro da 16 bit viene utilizzato come flag per distinguere il caso in cui si sta calcolando il CRC della prima parola della sequenza (LSB = '0') oppure se il nuovo dato inviato dal blocco SPI fa parte di una sequenza già iniziata del messaggio complessivo su cui viene calcolato il CRC (LSB = '1'). Il valore del LSB di questo registro viene portato come segnale di ingresso alla CU e prende il nome di "bitmode". La scrittura in questo registro è comandata unicamente dalla control unit del blocco CRC.

8. SELETTORE USCITA: questo blocco combinatorio serve per trasmettere sul segnale di uscita DOUT il contenuto dei registri 0, 1, 2 o 3 in base al segnale di indirizzo ADDRESS ricevuto. Nello specifico il segnale DOUT assume il valore di uno dei quattro registri elencati il colpo di CLK successivo rispetto a quando il segnale RD diventa attivo. Questo è dovuto al fatto che l'interfaccia SPI a cui è collegato il blocco CRC campiona il segnale DOUT il colpo di CLK successivo rispetto a quando diventa attivo il segnale di RD. In tutti gli altri casi e se l'indirizzo specificato dal segnale ADDRESS non corrisponde ad un numero da 0 a 3 in decimale (indirizzo dei registri che si vuole poter leggere), il segnale DOUT rimane uguale a 0x0000.

9. CRC HARDWARE: è il blocco che si occupa di svolgere le operazioni per il calcolo del CRC data una parola di 16 bit o una sequenza di parole. Il funzionamento di questo blocco viene spiegato in seguito.

10. COUNTER: è un contatore su 4 bit.

11. TC: questo blocco (terminal counter) riceve in ingresso un segnale su 4 bit che corrisponde al valore del blocco COUNTER e restituisce in uscita un segnale su 1 bit. Esso è posto al valore logico '1' se il valore del blocco COUNTER è 0xF, altrimenti è posto a '0'.

12. CU: blocco che permette di gestire i comandi di controllo di tutti i blocchi di cui è costituito il blocco CRC.

1.3 Funzionamento blocco CRC HARDWARE

L'idea di base dietro al calcolo del CRC-16 consiste nel partire da un messaggio composto da una o più parole su 16 bit, aggiungere al messaggio tanti zeri pari al numero di bit su cui ci aspettiamo di ricevere il risultato del CRC (nel nostro caso 16 zeri) e infine, eseguendo la divisione tra il messaggio

e il polinomio caratteristico, si ottiene il risultato del calcolo del CRC. La divisione, per semplicità, viene eseguita senza riporto e quindi può essere implementata con le porte XOR. Un modo semplice per svolgere la divisione senza riporto in hardware è quello di utilizzare uno shift register con delle porte XOR poste in punti particolari determinate dal divisore. Lo shift register deve poter contenere un numero di bit pari al numero di bit del divisore meno uno (nel nostro caso 16 bit) e l'algoritmo inizia ponendo il MSB del messaggio nel MSB dello shift register. Si effettua poi l'operazione di shift del registro fino a quando LSB del messaggio (ovvero il dato ricevuto dall'SPI aumentato da 16 bit posti a 0) non si trova nella posizione del LSB dello shift register. A questo punto il calcolo del CRC è terminato e il contenuto dello shift register corrisponde al risultato del calcolo del CRC del messaggio iniziale.
Nella nostra architettura, l'interfaccia SPI è in grado di gestire solo parole su 16 bit perciò se il messaggio su cui va calcolato il CRC è composto da più parole, esse arriveranno al blocco CRC in momenti diversi. E' necessario quindi continuare il calcolo del CRC ogni volta che l'interfaccia SPI fornisce al blocco CRC una nuova parola della sequenza del messaggio da calcolare. Per effettuare quanto appena detto il blocco CRC HARDWARE è composto da due shift register da 16 bit che attraverso l'attivazione del segnale LD (load) permettono il caricamento in parallelo. Come mostrato in Figura 2, i due shift register sono posti in serie per creare un unico shift register da 32 bit. Prendendo come riferimento la Figura 2, il LSB è il bit più a destra mentre il MSB è il bit più a sinistra tra tutti quelli contenuti in ciascun shift register. Quando il segnale SE (shift enable) diventa attivo, viene caricato il bit '0' nel LSB del shift register 2. Lo shift register 1 è lo shift register che si occupa di svolgere attivamente il calcolo del CRC mentre lo shift register 2 è un registro di appoggio. Nello shift register 1, il cui schema di massima è riportato in Figura 3, sono presenti tre gate XOR tra i flip-flop del registro. Quando il segnale SE diventa attivo, viene caricato nel flip-flop a sinistra di ciascuna porta XOR (sempre in riferimento allo schema riportato) il risultato dell'operazione XOR tra il bit contenuto nel flip-flop a destra rispetto alla porta logica e il MSB dello shift register 1. Le porte XOR sono state collocate all'interno dello shift register 1 in base ai termini non nulli del polinomio caratteristico del CRC da eseguire.

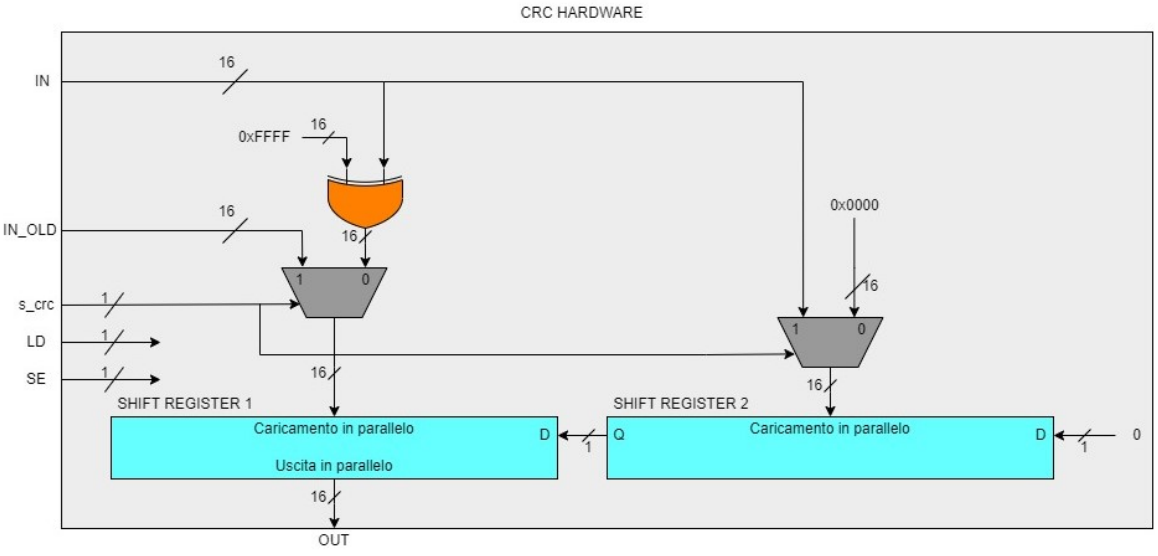


Figura 2: Schema blocco CRC HARDWARE

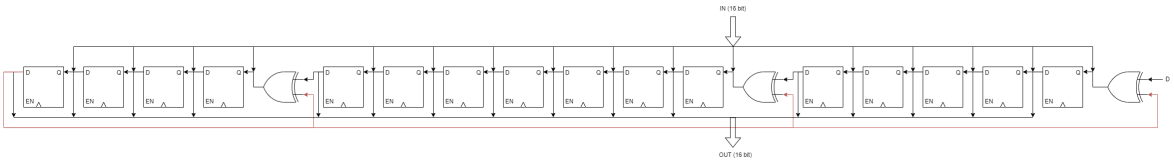


Figura 3: Schema shift register 1

Nello svolgiemento delle operazioni vanno considerati due casi distinti: il caso in cui la nuova parola trasmessa dall'interfaccia SPI sia la prima del messaggio su cui va calcolato il CRC oppure se è una parola all'interno o alla fine del messaggio.
Trovandosi nel primo caso, la nuova parola prima di essere caricata in parallelo nello shift register 1 deve essere posta in xor bit a bit con il valore di inizializzazione. Quanto appena detto viene spiegato nel paragrafo 8.6 dell'articolo posto al seguente link.
Nello shift register 2, invece, viene caricato un vettore da 16 bit posti tutti a '0'. A questo punto, non sapendo a priori se la parola data è l'unica del messeggio o se arriveranno altre parole, è necessario tenere valide entrambe le strade. In particola viene salvato il contenuto dello shift register 1 nel REGISTRO 4 usato come registro di appoggio. In esso infatti viene salvato il valore intermedio del calcolo del CRC prima che esso venga "contaminato" dallo shift con il vettore di 16 bit posti a '0' che serve per terminare l'operazione del calcolo del CRC. Successivamente si effettua 16 volte lo shift in entrambi gli shift register del blocco CRC HARDWARE in modo da concludere il calcolo del CRC e ottenere nello shift register 1 il risultato corretto del calcolo.
Trovandosi nel secondo caso, va aggiunto al calcolo parziale del CRC la nuova parola. Di consequenza viene caricato in parallelo nello shift register 1 il valore del calcolo parziale del CRC contenuto nel REGISTRO 4 e nello shift register 2 viene caricata la nuova parola. Dopo aver effettuato lo shift per 16 volte si ha che lo shift register 1 contiene il valore del calcolo parziale del CRC che quindi come prima viene salvato nel REGISTRO 4 mentre lo shift register 2 ha tutti i bit posti a '0'. Effettuando ulteriormente 16 shift, il contenuto dello shift register 1 corrisponde con il calcolo corretto del CRC di tutto il messaggio (la sequenza di tutte le parole).

1.4 Timing diagram

In Figura 4 è mostrato il timing diagram di due scritture effettuate all'indirizzo 0x000, precedute da un reset della macchina. Quando il segnale esterno asincrono RST_SW passo allo stato attivo, immediatamente si effettua il reset di tutto il blocco CRC. Successivamente essa rimane in attesa (stato di IDLE) di uno o più segnali per procedere con il prossimo stato. Il blocco SPI, per come è stato progettato, tiene allo stato attivo il segnale WR per un tempo pari al periodo del CLK e nel mentre mantiene invariati i dati presenti sugli ingressi DIN e ADDRESS. Dal momento che la frequenza del CLK del blocco SPI è la medesima del blocco CRC, è garantito che il segnale En0 (attivato in modo combinatorio tramite il blocco DECODIFICA) viene campionato una sola volta dal fronte di salita del CLK. In quel momento il segnale DIN viene salvato nel REGISTRO 0 e si passa allo stato LOAD_0 in cui inizia il calcolo per ottenere il risultato del CRC. Una volta completato si torna nello stato di IDLE. Dal momento che prima di eseguire qualsiasi operazione è stato effettuato il reset della macchina, il bit_mode era pari al livello logico 0. Al termine del calcolo del CRC della prima parola, esso avrà il valore pari al livello logico 1. Successivamente il blocco SPI scrive una nuova parola sempre all'indirizzo 0x0000 e quindi si procede con il calcolo del CRC in modo sequenziale.

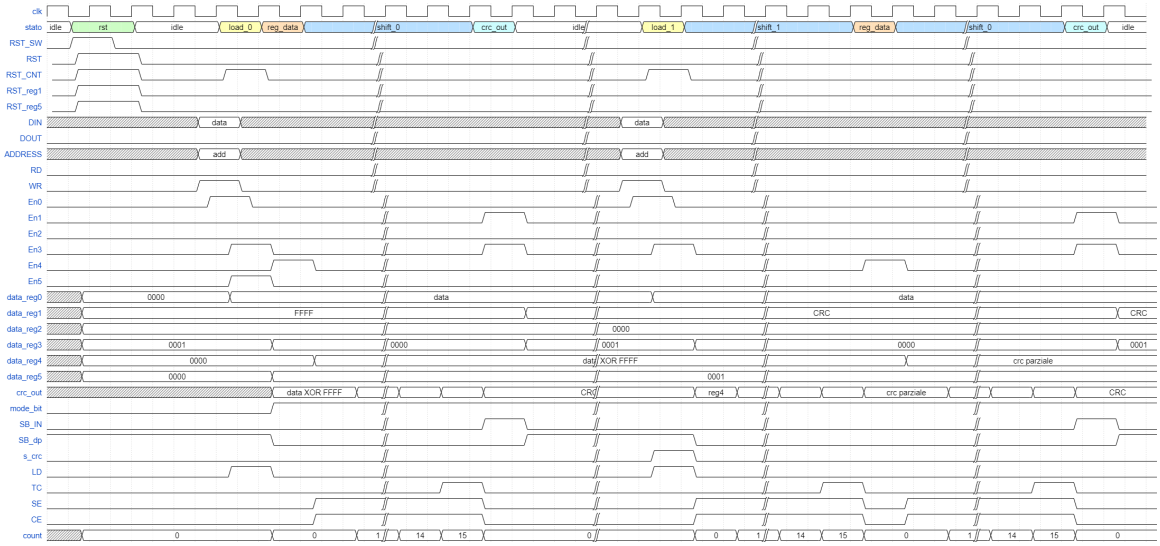
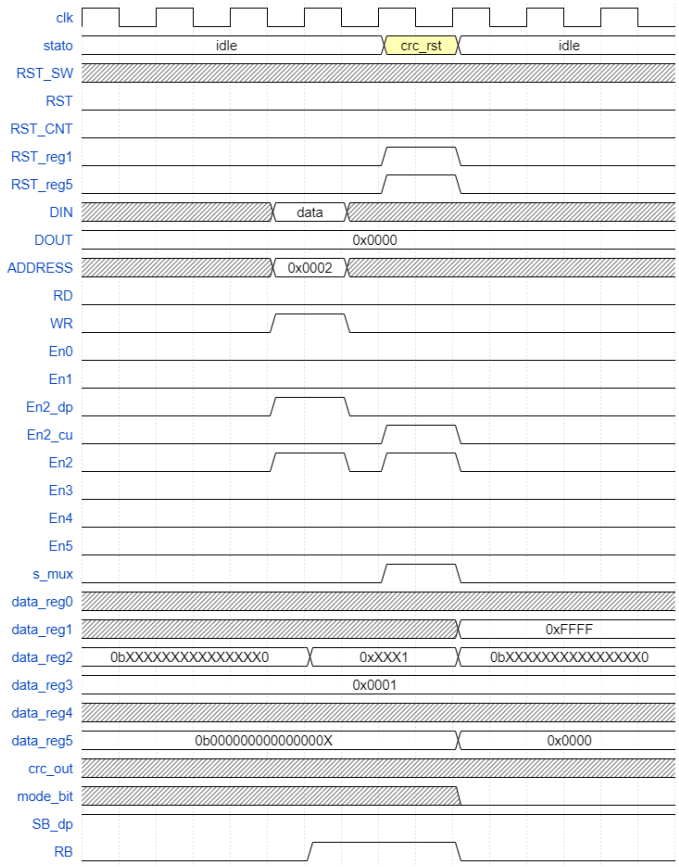
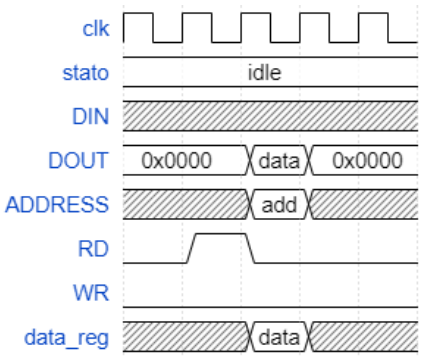


Figura 4: Timing diagram di 2 scritture effettuate all'indirizzo 0x0000 con conseguente calcolo del CRC

In Figura 5a è stato effettuato il timing diagram nel caso di una scrittura all'indirizzo 0x0002 avete l'LSB del dato scritto posto a 1. Questo causa il reset del calcolo del CRC. In particolare, viene effettuato il reset del REGISTRO 1 in modo da contenere all'interno di esso il valore di inizializzazione del CRC. Inoltre viene effettuato anche il reset del REGISTRO 5 per fare in modo che il segnale bit_mode sia uguale a 0. Infine si porta al livello logico 0 l'LSB del REGISTRO 2 (in modo da non bloccare la macchina in uno stato di reset del CRC perpetuo) senza modificare gli altri bit del registro. Infine in Figura 5b è rappresentato il timing diagram di una lettura generica da un qualsiasi registro. E' necessario che il blocco SPI interfacciato con il blocco CRC mantenga invariato il segnale ADDRESS per l'intera durata del colpo di clock successivo a quello in cui asserisce il segnale RD. Il segnale DOUT in uscita dal blocco CRC contiene il valore del dato contenuto nel registro che si vuole leggere solamente il colpo di CLK successivo a quello in cui il segnale RD rimane attivo. Questo è necessario dal momento che il blocco SPI, per come è concepito, campiona il segnale DOUT in ingresso al blocco SPI il colpo di CLK successivo rispetto all'asserimento di RD. Quando appena detto funziona solo nel momento in cui il blocco CRC e il blocco SPI lavorino alla stessa frequenza di CLK. La fase tra i due CLK, invece, non è rilevante per il corretto funzionamento.



(a) Timing diagram di una scrittura effettuata all'indirizzo 0x0002



(b) Timing diagram di una lettura

Figura 5

1.5 Descrizione FSM

Dal timing diagram è stata ricavata la Finite State Machine per la realizzazione del blocco CRC come riportato in Figura 6.

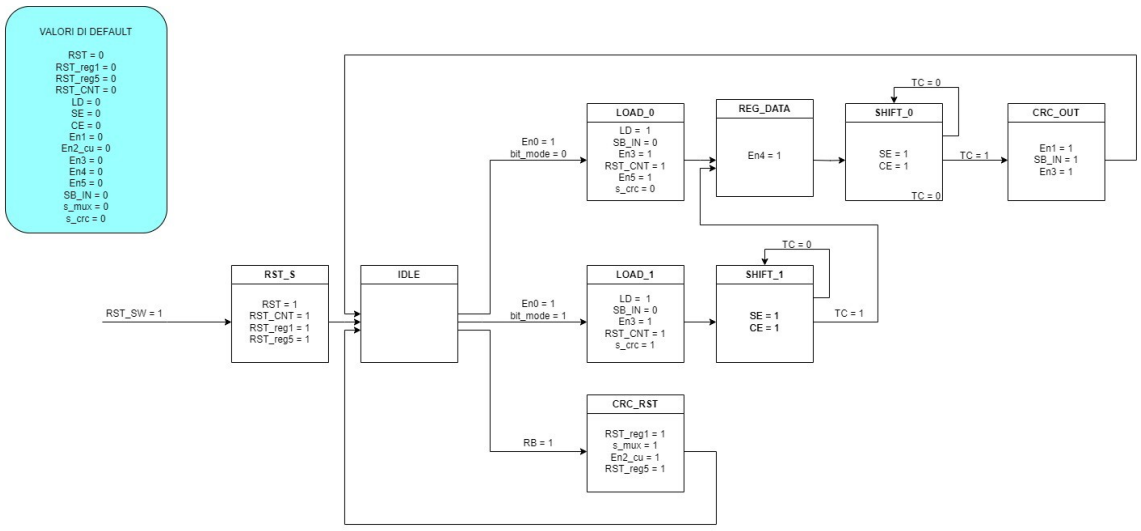


Figura 6: FSM

Come per l'interfaccia SPI è stato utilizzato un clock alla frequenza di 10 MHz. La descrizione di ognuno degli stati è riportata di seguito:

1. RST S: si entra in questo stato quando il segnale asincrono RST_SW diventa attivo. Quando si disasserisce, al colpo di clock successivo si passa allo stato di IDLE. In questo stato si forniscono i comandi per il reset di tutto il blocco CRC.

2. IDLE: è lo stato di attesa in cui tutti i segnali di controllo rimangono al loro valore di default.

3. LOAD 0: si entra in questo stato se viene inserita una nuova parola nel REGISTRO 0 (comando di En0 attivo) e il segnale "bitmode" è posto a '0'. In questo stato si fornisce il comando di reset al COUNTER, si fornisce il comando di impostare il LSB del REGISTRO 3 (ovvero lo SB_IN) a '0' per segnalare che il blocco CRC sta effettuando il calcolo del CRC, si fornisce il comando di LD al blocco CRC HARDWARE per caricare i dati negli shift register e infine si fornisce il comando per impostare il "bitmode" a '1'.

4. REG DATA: in questo stato si fornisce il comando per salvare il valore del calcolo parziale del CRC nel REGISTRO 4.

5. SHIFT 0: in questo stato si fornisce il comando di shift (SE) agli shift register del blocco CRC HARDWARE. Si rimane in questo stato fino a quando il segnale TC non diventa attivo.

6. CRC OUT: in questo stato si fornisce il comando per caricare nel REGISTRO 1 il risultato del calcolo del CRC e si forniscono i comandi per impostare il segnale SB_IN a '1' per segnalare che l'operazione del calcolo del CRC è terminata.

7. LOAD 1: si entra in questo stato se viene inserita una nuova parola nel REGISTRO 0 (comando di En0 attivo) e il segnale "bitmode" è posto a '1'. In questo stato si fornisce il comando di reset al COUNTER, si fornisce il comando di impostare il LSB del REGISTRO 3 (ovvero lo SB_IN) a '0' per segnalare che il blocco CRC sta effettuando il calcolo del CRC e si fornisce il comando di LD al blocco CRC HARDWARE per caricare i dati negli shift register.

8. SHIFT 1: in questo stato si fornisce il comando di shift (SE) agli shift register del blocco CRC HARDWARE. Si rimane in questo stato fino a quando il segnale TC non diventa attivo.

9. CRC RST: si entra in questo stato quando LSB del REGISTRO 2 viene posto a '1'. In questo stato si danno i comandi per effettuare il reset del REGISTRO 1 e portare al valore '0' il LSB dei REGISTRI 2 e 5.

2 Simulazioni

2.1 Simulazione blocco CRC

Tramite il file *tb_CRC.vhd* è possibile eseguire una simulazione per verificare il corretto funzionamento del blocco CRC. In particolare in questa simulazione viene effettuato il reset della macchina, in seguito viene eseguita la lettura in ordine dei REGISTRI da 0 a 3. Successivamente si effettuano due scritture nel REGISTRO 0 per verificare che il blocco calcoli correttamente il CRC sia della singola parola sia di sequenze di parole. Successivamente viene scritto il numero 0x0001 nel REGISTRO 2 per forzare il reset del CRC. Infine viene effettuata un'ulteriore scrittura nel REGISTRO 0 per verificare che il reset del CRC sia andato a buon fine e quindi il calcolo del CRC sia effettuato solo sull'ultima parola mandata.
Il timing diagram effettuato dalle simulazioni modelsim è riportato in Figura 7.

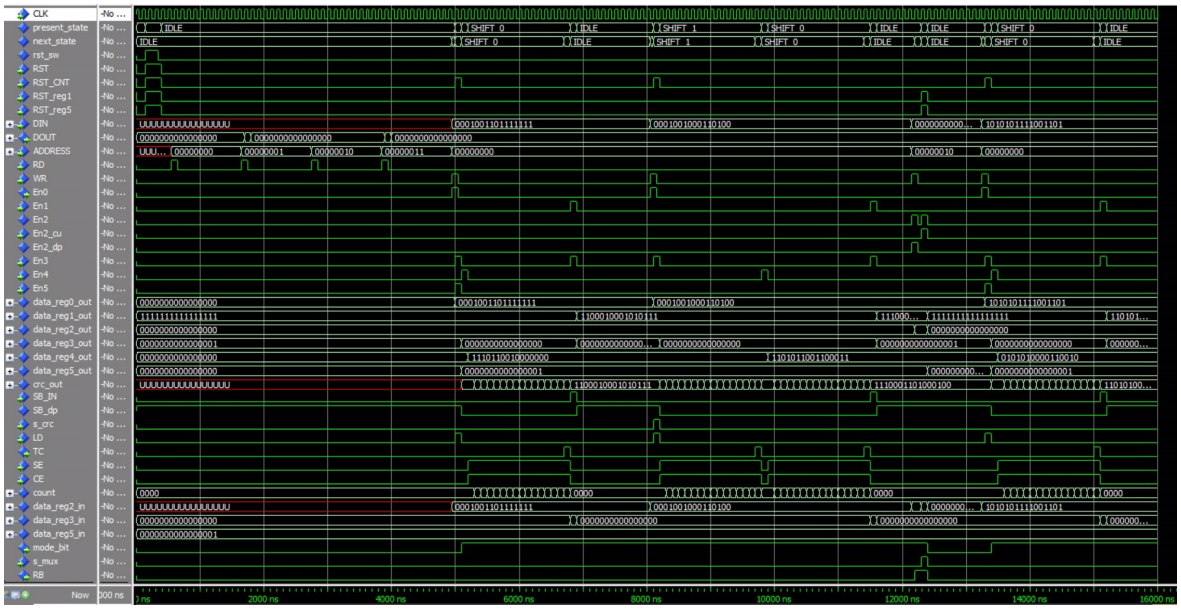


Figura 7: Timing diagram del blocco CRC effettuato con la simulazione modelsim

I vari zoom delle sezioni della simulazione modelsim riportata in Figura 7 sono riportati in Figura 8, 9, 10 e 11.

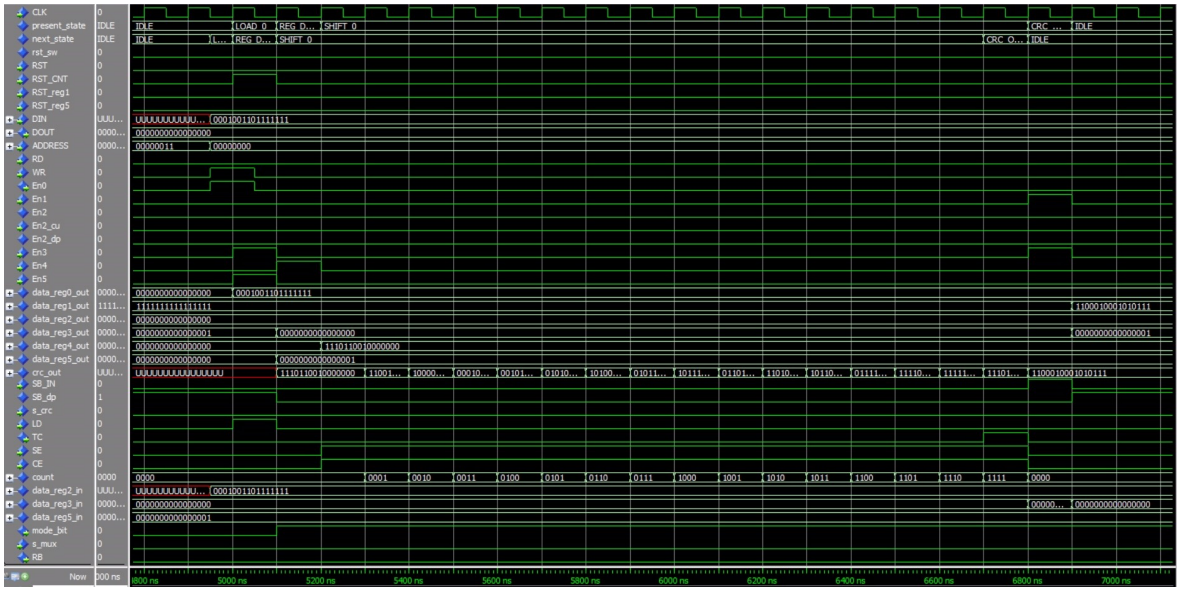


Figura 8: Scrittura del REGISTRO 0 e calcolo CRC con bit_mode=0

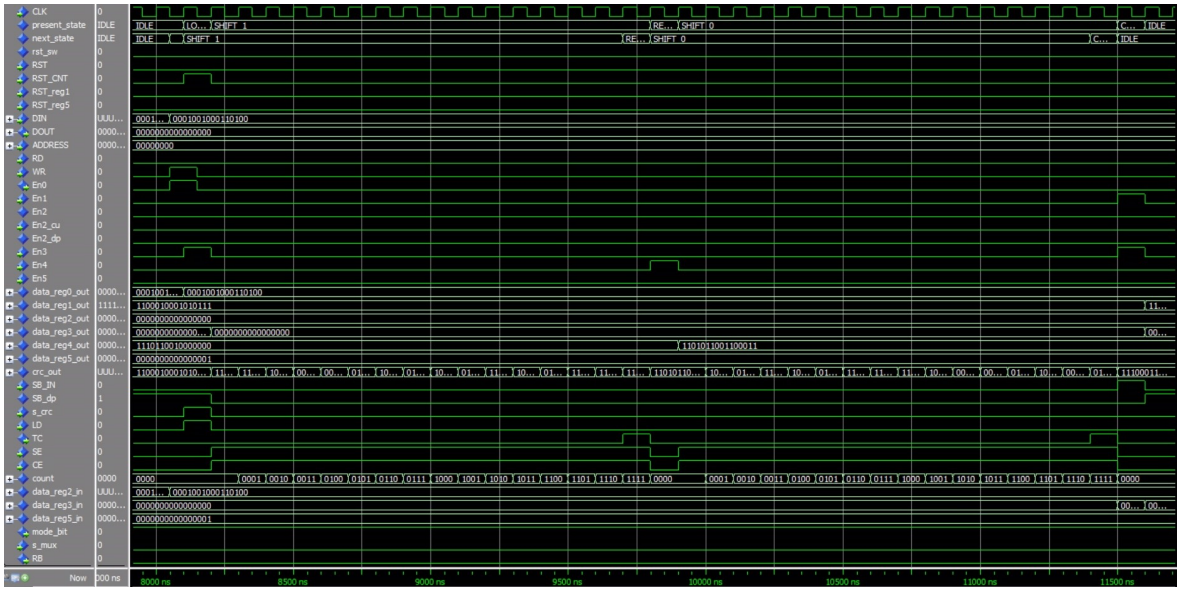


Figura 9: Scrittura del REGISTRO 0 e calcolo CRC con bit_mode=1

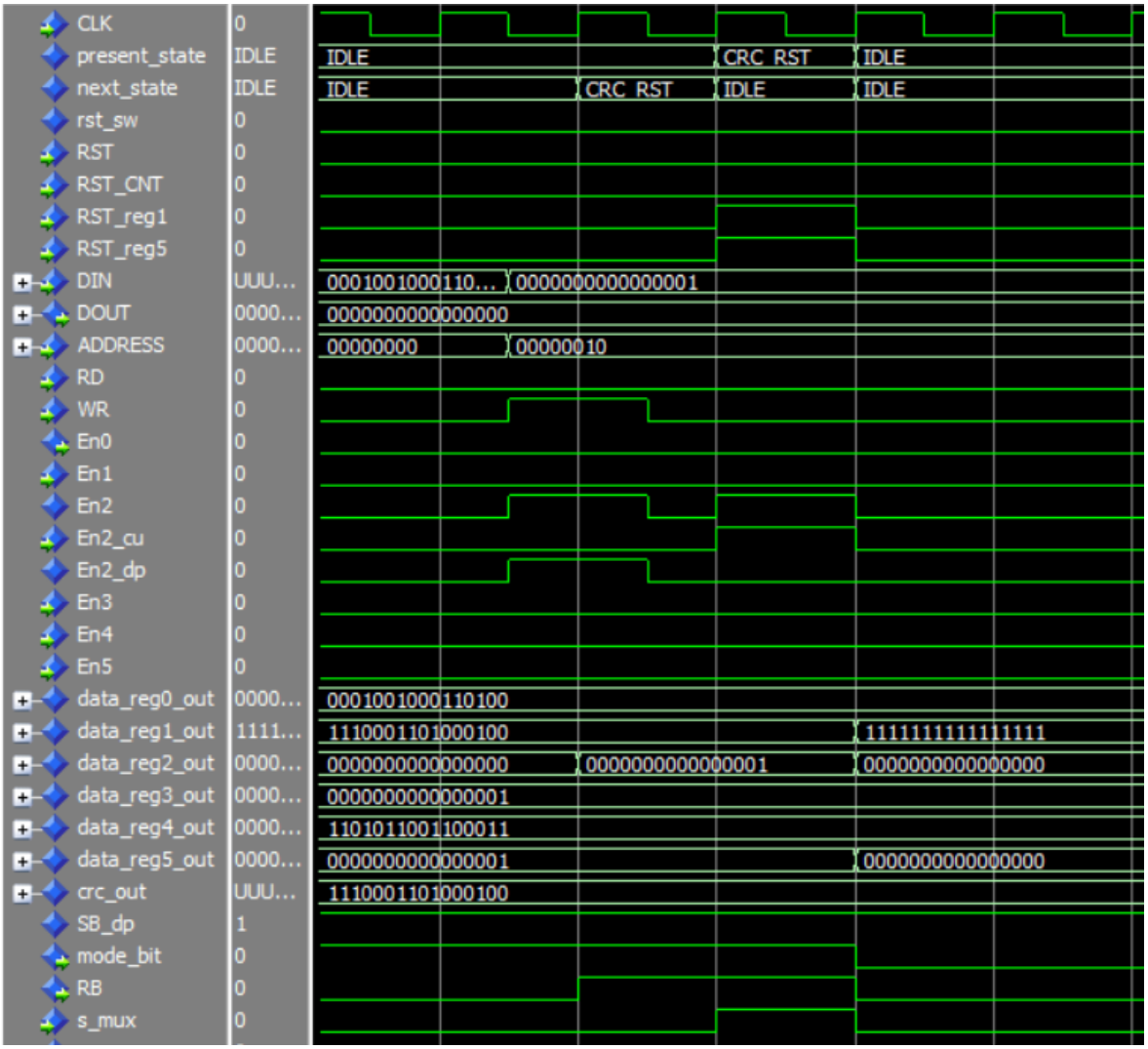


Figura 10: Scrittura del REGISTRO 2 e reset blocco CRC

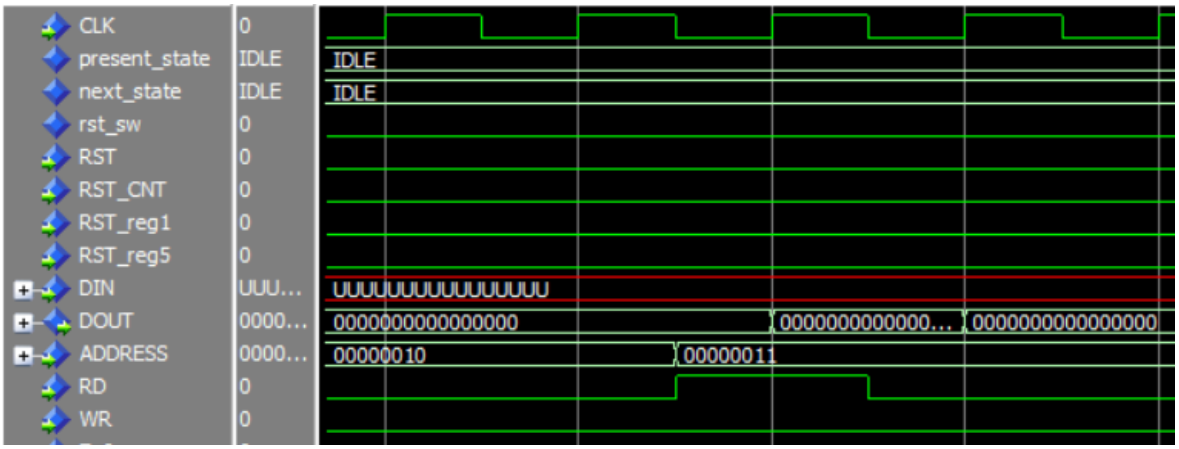


Figura 11: Lettura

Per verificare che non siano stati commessi errori, ogni sezione del timing diagram ottenuto per mezzo della simulazione modelsim è stato confrontato con le rispettive sezioni del timing diagram atteso. Inoltre sono stati verificati che i valori contenuti dentro ai REGISTRI sia sempre quelli attesi in basi ai valori impostati nella simulazione. Il valore del calcolo del CRC che è stato usato per verificare la correttezza del blocco CRC è stato calcolato per mezzo del simulatore online al seguente link.

2.2 Simulazione blocco CRC + blocco SPI

Tramite il file *tb_CRCconSPI.vhd* è possibile eseguire una simulazione per verificare il corretto funzionamento del blocco CRC unito all'interfaccia SPI. Per poter lanciare la simulazione, oltre a tutti i file .vhd necessari a descrivere il blocco CRC e il blocco SPI, è necessario anche il file "CRCconSPI.vhd" che unisce i due blocchi detti precedentemente in un unico blocco. La simulazione ha lo scopo di verificare che sia rispettato il timing tra il blocco SPI e il blocco CRC e che quindi sia possibile utilizzare correttamente il blocco CRC tramite l'interfaccia SPI. Essa, dopo aver effettuato il reset di entrambi i blocchi, effettua una scrittura e una lettura sul REGISTRO 0 del blocco CRC. Per effettuare la scrittura tramite il protocollo SPI è necessario trasmettere il numero 32 su 8 bit, seguito dall'indirizzo sempre su 8 bit e infine il dato su 16 bit. Per la lettura, invece, è necessario trasmettere il numero 33 su 8 bit seguito dall'indirizzo su cui si vuole leggere sempre su 8 bit. Verrà restituito subito dopo sul segnale MISO il dato su 16 bit. La velocità di trasmissione del protocollo SPI deve essere di 1 MHz. Tra la scrittura e la lettura il segnale nSS (slave select) viene disasserito, si aspetta per $1\mu s$ e infine si riasserisce il segnale di nSS per effettuare la lettura.
Il timing diagram effettuato dalle simulazioni modelsim è riportato in Figura 12.

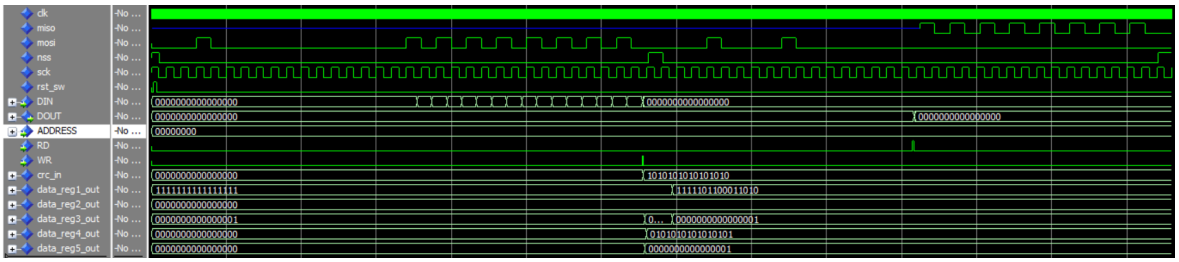


Figura 12: Simulazione blocco CRC con blocco SPI

2.3 Test finale

Una volta verificato il corretto funzionamento del blocco CRC con le due simulazioni precedenti, è stato caricato il progetto sulla FPGA della scheda VirtLab. In primo luogo, sono state testate manualmente le principali funzioni del blocco CRC, ovvero la lettura dai registri 0, 1, 2 e 3 e la scrittura nei registri 0 e 2. Ovviamente per ognuna di queste operazioni è stato verificato che la scheda eseguisse le operazioni attese senza commettere errori. Successivamente è stato eseguito un test su larga scala per verificare il corretto calcolo del CRC per mezzo del codice python *test_CRC.py*. Come nella trattazione precedente viene definito "parola" il dato da 16 bit inviato ogni volta che si fa una scrittura sul blocco CRC e viene definito "messaggio" la successione di più parole su cui andrà calcolato il CRC finale. Prima di lanciare la simulazione è possibile impostare a piacimento due parametri:

1. NUMERO_PROVE: ovvero il numero di messaggi che viene inviato alla scheda VirtLab.
2. LUNGHEZZA_MASSIMA_MESSAGGIO: corrisponde al numero massimo di parole di cui è composto ogni messaggio.

La simulazione consiste nel mandare alla scheda un numero di messaggi specificato dal parametro NUMERO_PROVE la cui lunghezza è un numero di parole casuale tra 1 e LUNGHEZZA_MASSIMA_MESSAGGIO. Ogni parola, a sua volta, è un numero casuale tra il valore minimo (0x0000) e il valore massimo (0xFFFF). Si va infine a verificare che, ad ogni nuova parola mandata alla scheda, il CRC calcolato sia corretto. Per calcolare il CRC teorico è stata utilizzata la libreria gratuita *crcmod* impostata opportunamente per calcolare lo standard CRC-16/CCITT-FALSE. Nel caso in cui il programma dovesse riscontrare errori, essi vengono segnalati all'utente. In caso contrario viene restituito un messaggio di corretta esecuzione del blocco CRC.

3 Specifiche di utilizzo

Per comunicare il blocco CRC è necessario utilizzare il protocollo SPI alla velocità di 1 MHz. Dal momento in cui il segnale nSS (slave select) si asserisce, inizia la comunicazione. Sul segnale MOSI, i primi 8 bit sono per lo *stato* e i successivi 8 sono per specificare l'*indirizzo* su cui si vuole scrivere o leggere. Se si vuole scrive i bit di stato devono avere un valore di 32 in decimale. Se si vuole leggere devono avere un valore di 33 in decimale. In caso di scrittura, va mandato sempre sul segnale MOSI il dato su 16 bit. In caso di lettura verrà ricevuto sul segnale MISO il dato presente all'indirizzo specificato sempre su 16 bit. Una volta terminata la comunicazione il segnale nSS si disasserisce. Questo segnale è attivo basso. Per quanto riguarda il blocco CRC, se si vuole mandare un dato su cui andrà calcolato il CRC esso deve essere su 16 bit e deve essere mandato all'indirizzo 0x00. Ad ogni dato su 16 bit ricevuto dal blocco all'indirizzo 0x00, viene calcolato il CRC. Per ottenere il risultato del calcolo è necessario effettuare una lettura all'indirizzo 0x01. Se si vuole calcolare il CRC di un messaggio, quindi la successione di più dati, è necessario inviare uno per volta i dati a partire dal più significativo (sempre all'indirizzo 0x00) e infine si effettua una lettura all'indirizzo 1 per avere il valore del CRC calcolato sull'intero messaggio. In ogni caso, ad ogni nuovo dato mandato, all'indirizzo 1 è presente il calcolo del CRC corretto effettuato sui dati mandati fino a quel momento. Per indicare al blocco CRC che il messaggio precedente è terminato e quindi che il prossimo dato scritto all'indirizzo 0 faccia parte di un nuovo messaggio, è necessario scrivere il bit '1' nel LSB del registro all'indirizzo 2.
E' possibile effettuare la scrittura unicamente sui registri all'indirizzo 0 e 2. Ogni tentativo di scrittura su un registro con indirizzo diverso da quelli specificati verrà ignorato.
E' possibile effettuare la lettura dai registri con numero di indirizzo compreso tra 0 e 3. La lettura da un registro con indirizzo diverso da quelli specificati restituirà il valore di 0x0000.

4 Documentazione

La documentazione necessaria per comprendere l'algoritmo e le specifiche dello standard CRC-16/CCITT-FALSE sono state trovate consultando i seguenti link:

1. articolo 1: funzionamento CRC e implementazione
2. articolo 2: parametri dello standard
3. articolo 3: funzionamento CRC
4. articolo 4: funzionamento CRC
5. articolo 5: utilizzo libreria python per calcolare il CRC
6. Calcolatore online CRC