

SISTEMI DIGITALI INTEGRATI

LABORATORIO 1

UART

Antonio Tudisco Lorenzo Lagostina Maria Elena D'Agostino

22 giugno 2022



Indice

1	UART	2
1.1	Divisione in blocchi e schema generale	2
2	TX	3
2.1	Datapath	3
2.2	ASM	6
2.3	Timing diagram	7
2.4	Simulazioni Modelsim	15
3	RX	19
3.1	Datapath	19
3.2	Timing diagram	21
3.3	ASM e CU	27
3.4	Simulazioni Modelsim	35
4	Bus Interface	41
4.1	Datapath	41
4.2	ASM	43
4.3	Timing diagram	44
4.4	Testbench	51
5	Risultati delle simulazioni e sintesi	52
5.1	Modelsim	52
5.2	Quartus	52

1 UART

1.1 Divisione in blocchi e schema generale

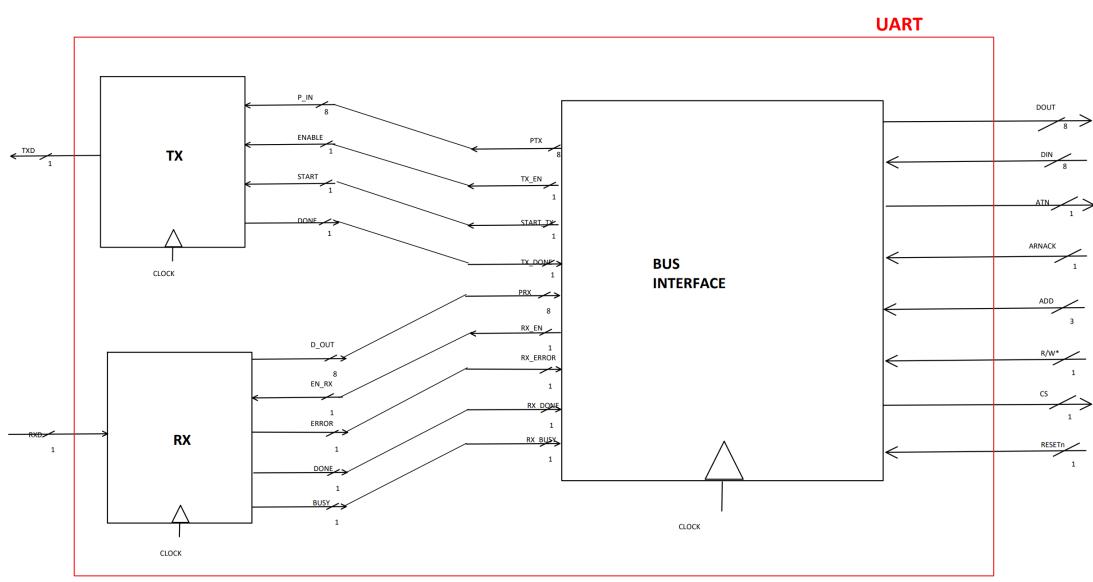


Figura 1: Schema generale

2 TX

2.1 Datapath

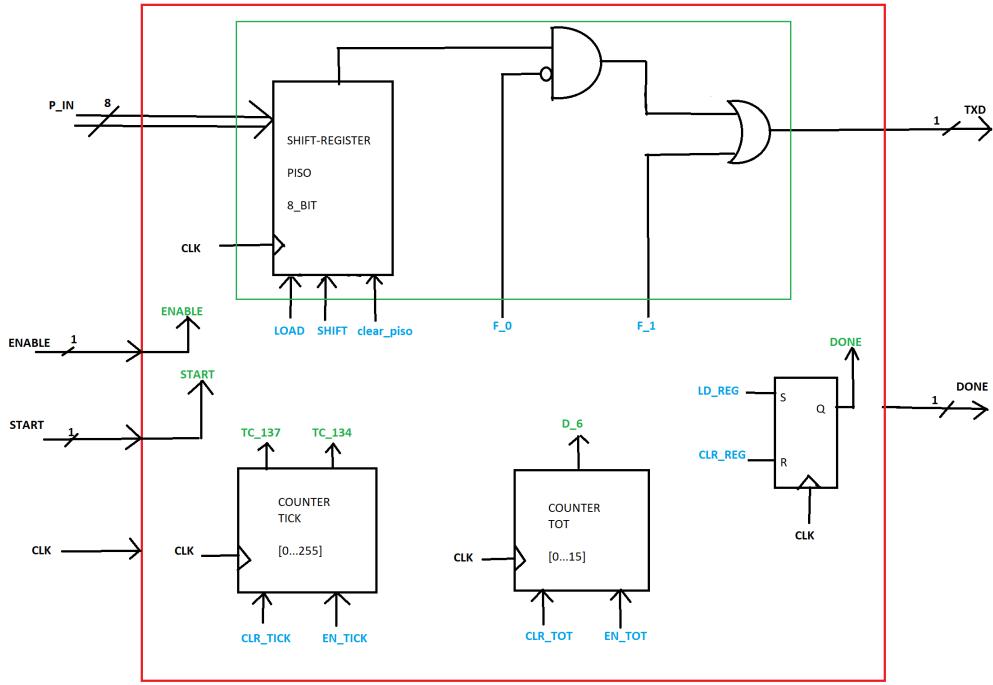


Figura 2: *Datapath del blocco di trasmissione*

Il blocco sotto esame trasmette verso l'esterno un dato di 8 bit contenuto nel registro di dato TxDATA.

L'interfaccia del blocco trasmettitore definita in fase preliminare è stata così strutturata:

- Segnale di enable: da ricevere e inviare alla Control Unit per abilitare il blocco trasmettitore.
- PIN: dato in ingresso al blocco trasmettitore che deve essere trasmesso serialmente in uscita.
- Segnale di start: da ricevere e inviare alla Control Unit del trasmettitore per avviare una nuova trasmissione.
- CLOCK
- TXD: uscita del blocco trasmettitore che invia un bit trasmesso per volta.
- Segnale di DONE: da inviare alla fine di ogni trasmissione.

Il trasmettitore è stato realizzato utilizzando i seguenti blocchi:

- Shift register PISO da 8 bit:

In esso viene caricato il dato da trasmettere che è formato da otto simboli. Questi saranno caricati in modo parallelo nel registro e trasmessi in modo seriale in uscita. Il blocco shift register è stato implementato in modo tale che quando è ricevuto il segnale di LOAD, viene abilitato il caricamento dei dati; grazie al segnale di controllo proveniente dalla Control Unit SHIFT i dati in ingresso saranno shiftati e dunque trasmessi in uscita. Il meccanismo di trasmissione è realizzato in modo tale che la prima cifra ad essere inviata è l'LSB del dato in ingresso. Infine, per il seguente blocco è stato previsto un segnale di clear, che permette di azzerare il contenuto del blocchetto quando necessario.

- Blocco start-stop:

il seguente blocco è puramente combinatorio e stato realizzato mediante l'utilizzo di due porte logiche elementari; una AND e una OR. È noto il fatto che in un protocollo asincrono per trasmettere correttamente un dato, è necessario costruire un frame, formato da: il bit di start, n bit di pay load da mandare serialmente in uscita ed il bit di stop. Per la trasmissione del bit di start, dalla Control Unit viene forzato il segnale F_1 al valore logico basso e F_0 al valore logico alto, in modo tale da avere in uscita il valore atteso corretto, lo zero logico, e mantenerlo per un periodo di simbolo. A fine trasmissione del dato è necessario inviare il bit di stop. Al contrario del caso descritto in precedenza, è necessario forzare il bit in uscita a '1', imponendo il segnale F_1 dalla Control Unit al valore logico alto.

- Counter tick:

è stato necessario inserire questo blocco poiché, per trasmettere ogni simbolo in uscita in modo corretto bisogna garantire che questo deve essere mantenuto per un tempo opportuno.

$$T_{\text{sym}} = (16 \text{ MHz}) / (115200 \text{ baud rate}) = 139 \text{ T-ck}$$

Il contatore è controllato dai segnali di clear (CLR_TICK) ed enable (EN_TICK), opportunamente gestiti dalla Control Unit, ed è munito di due terminal count (TC-137, utile per il conteggio del numero di cicli di clock necessari per trasmettere lo start-bit e tutti gli otto bit del dato e TC-134 necessario per il conteggio del tempo di simbolo dello stop_bit).

- Counter tot:

è stato inserito un secondo contatore utile per avvisare quando è terminata la trasmissione del simbolo D_6 e lo shift register piso ha effettuato per la settima e ultima volta lo shift, necessario per l'inizio della trasmissione dell'ottavo simbolo. Il contatore invierà alla Control Unit il segnale di stato D_6 grazie al quale si riuscirà a comprendere che la macchina si sta avviando verso la fine della trasmissione del frame. Anche in questo caso il contatore ha due segnali di controllo (CLR_TOT ed EN_TOT).

- FF-SR:

l'inserimento del flip flop di tipo S-R è stato necessario poiché utile per mandare in

uscita l'avviso del termine della trasmissione di un frame. L'invio del segnale di DONE sarà utile anche per indicare che il blocco è pronto per iniziare una nuova trasmissione. Il registro ha in ingresso i segnali di controllo CLR_REG e LD_REG.

2.2 ASM

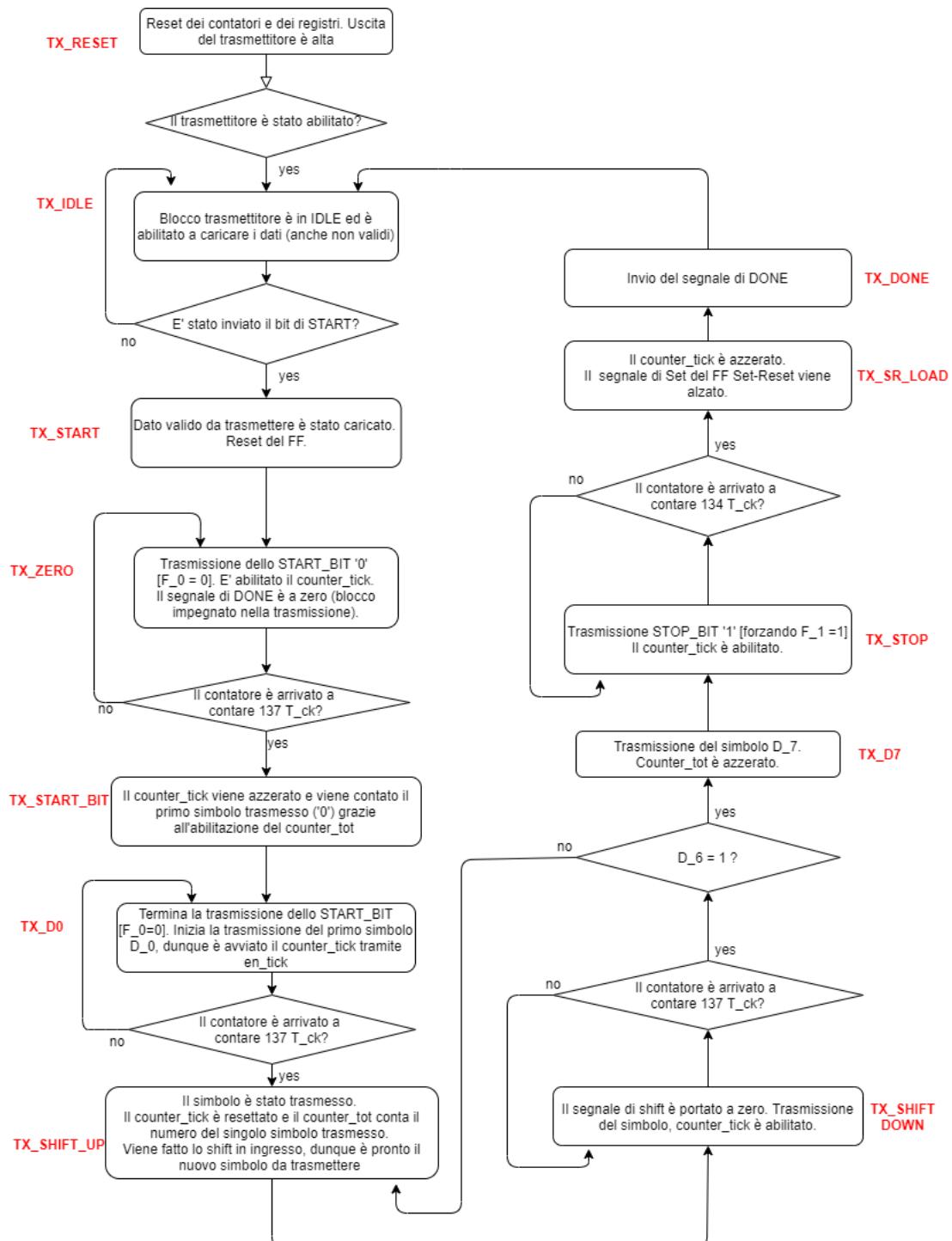


Figura 3: *ASM del blocco TX*

2.3 Timing diagram

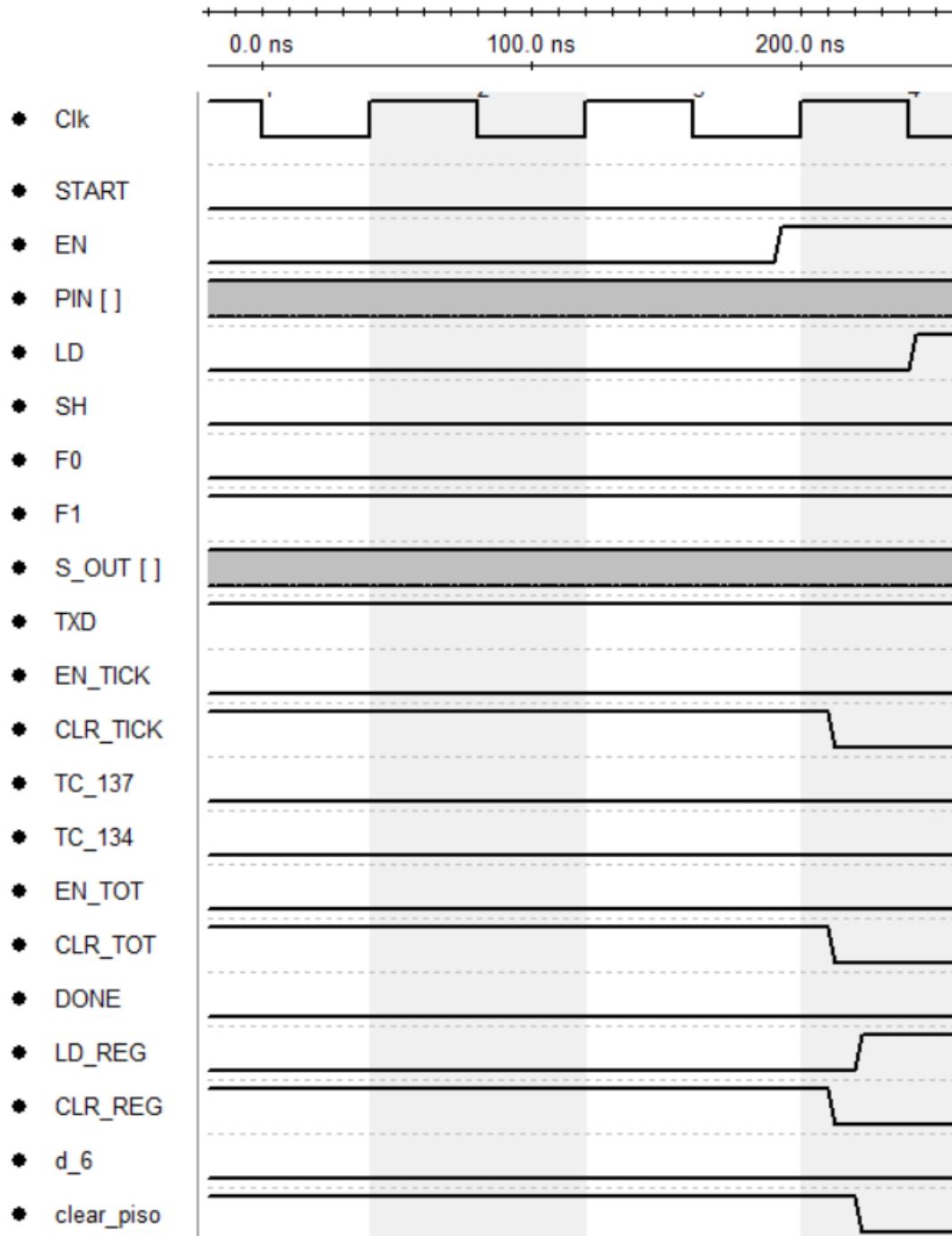
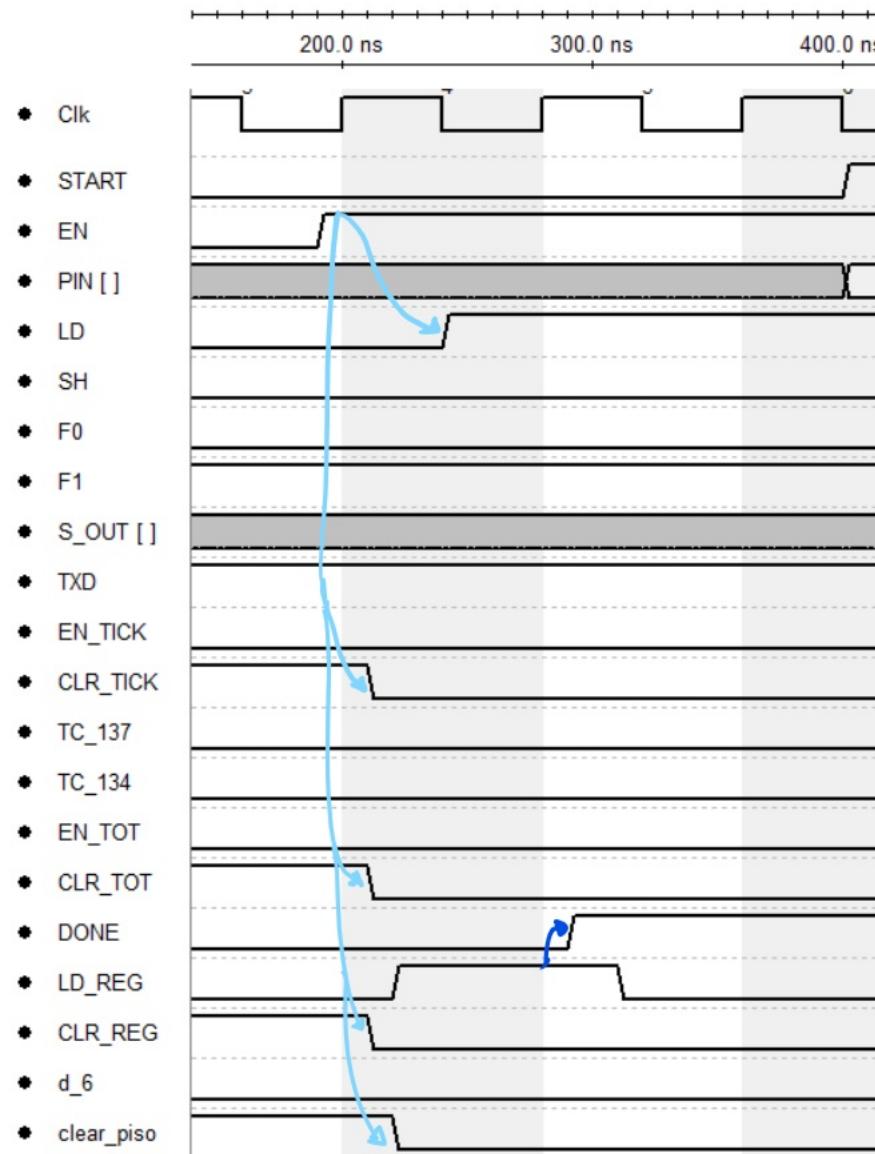
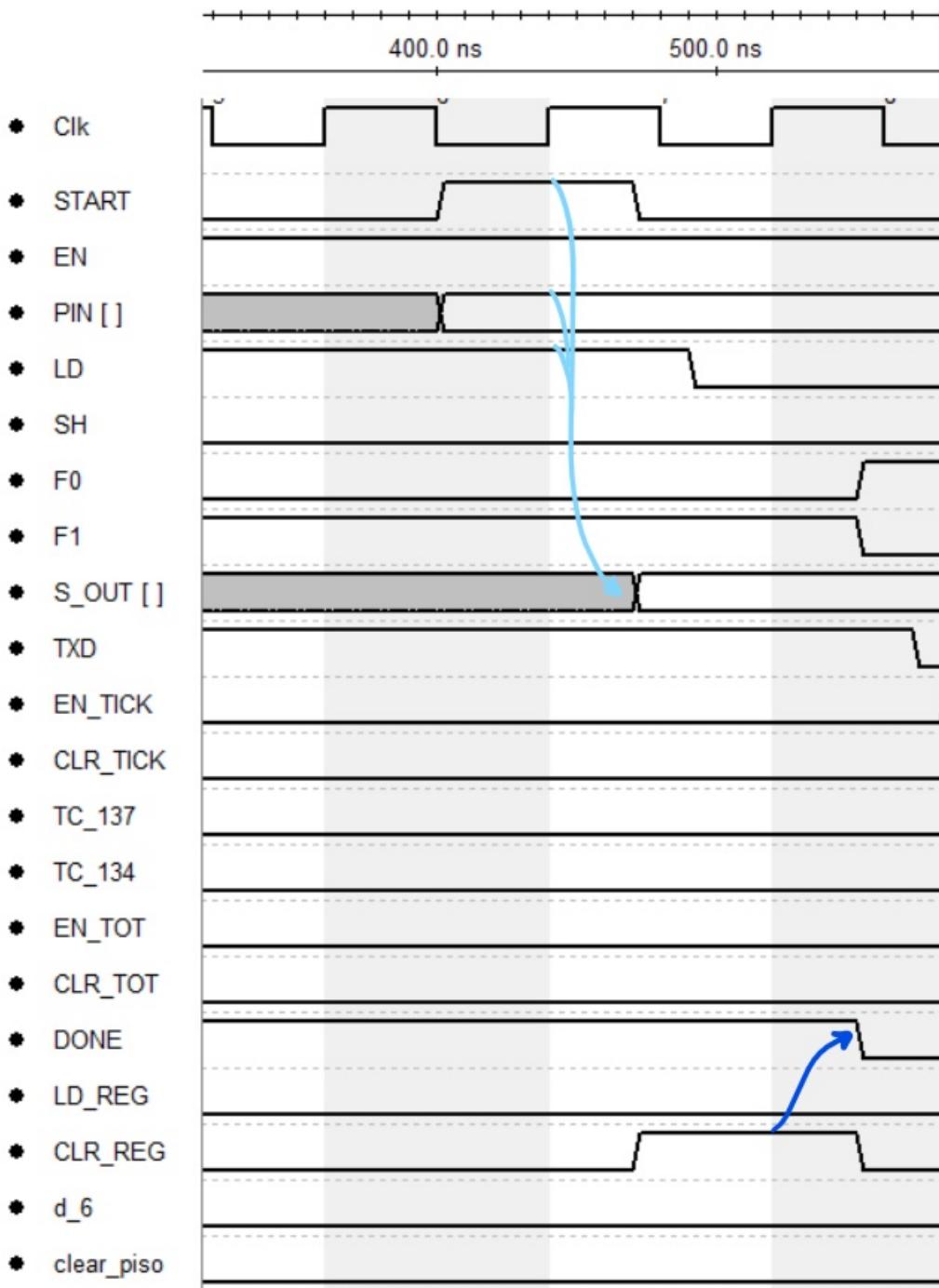


Figura 4: *stato di reset*

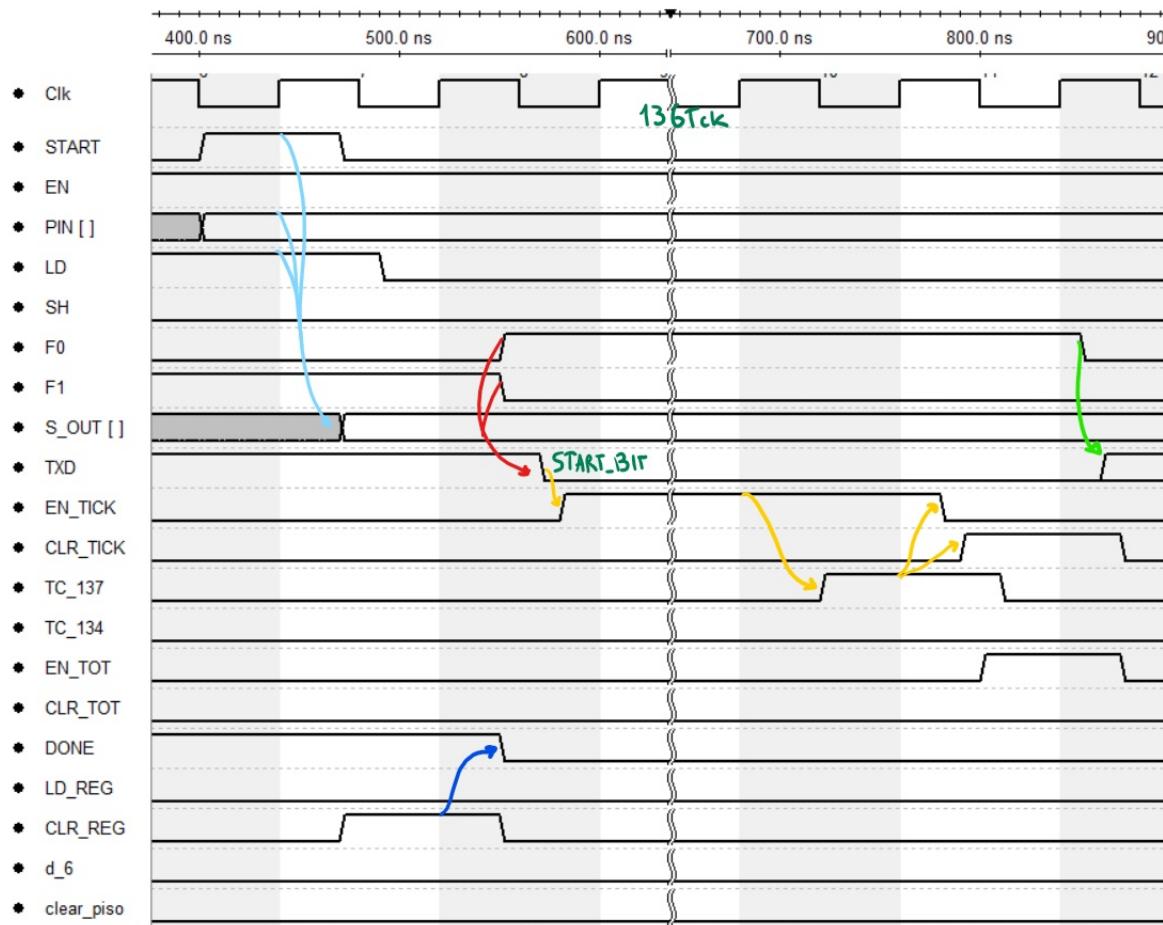
Nello stato di reset l'uscita del trasmettitore è a 1 [anche $F_1 = 1$]. Intanto vi è il clear dei contatori e dei registri.

Figura 5: *idle*

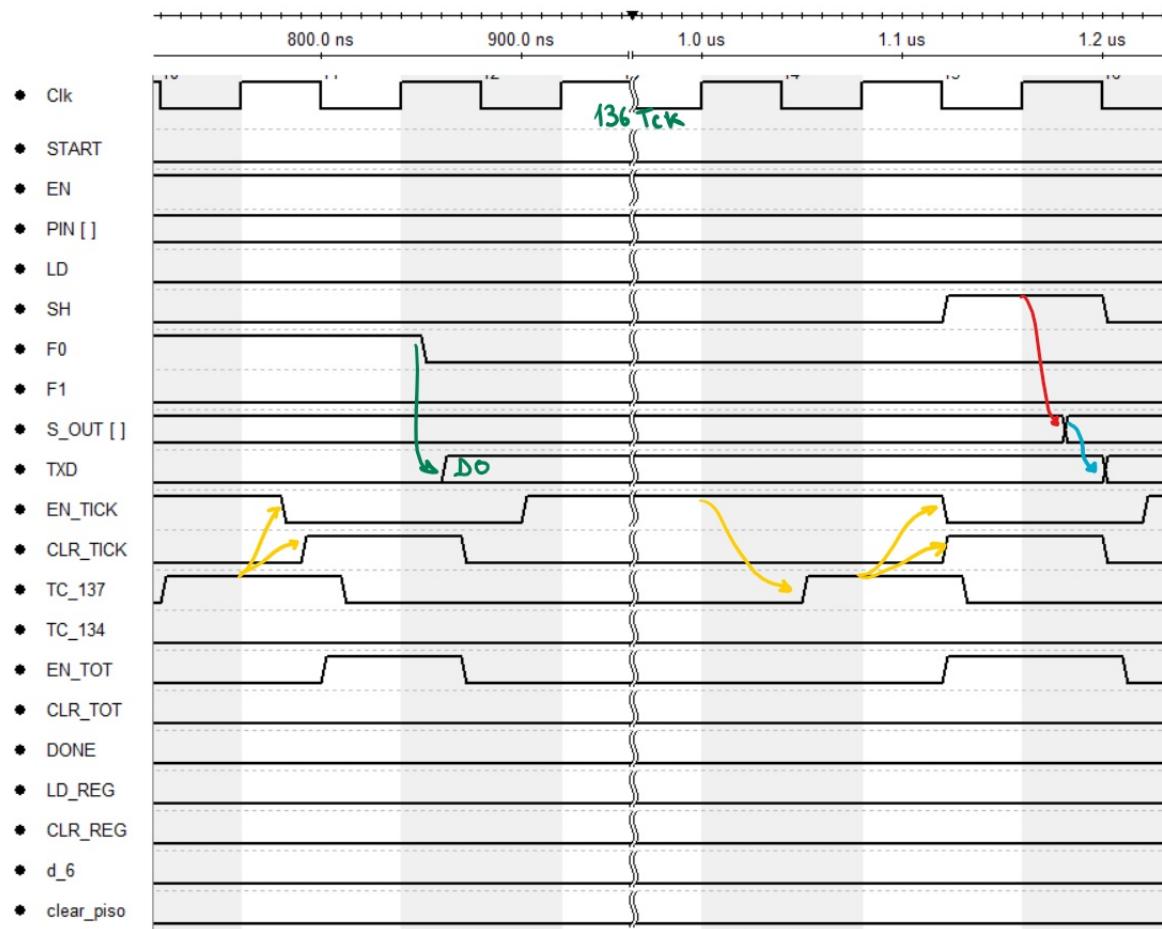
Durante lo stato di IDLE viene abilitato il trasmettitore tramite segnale di ENABLE; verrà inviato in uscita il segnale di DONE ad indicare il fatto che il blocco non è impegnato in nessuna operazione di trasmissione. Con LOAD=1 inizia il caricamento dello shift-register (possono essere caricati anche dati non validi).

Figura 6: *start*

Con la ricezione del segnale di $\text{START} = 1$, è avviata la trasmissione di un dato valido. Dopo un ritardo combinatorio, l'uscita dello shift-register (S_{out}) è definita. Si noti che START e LOAD sono due segnali allo stato logico '1' e PIN è valido nello stesso istante di clock, per rispettare la specifica di timing di lettura di un dato inviato dall'interfaccia sincrona.

Figura 7: *start_bit*

Dalla Control Unit il segnale F_1 è portato al valore logico basso; dunque, dopo un ritardo combinatorio, l'uscita del trasmettitore è a zero, iniziando la trasmissione del bit di start ('0'). La condizione per l'invio dello start bit viene mantenuta per 139 T_{ck} .

Figura 8: *D_0*

La trasmissione del simbolo *D_0* inizia quando viene posto *F_0* allo stato logico basso.

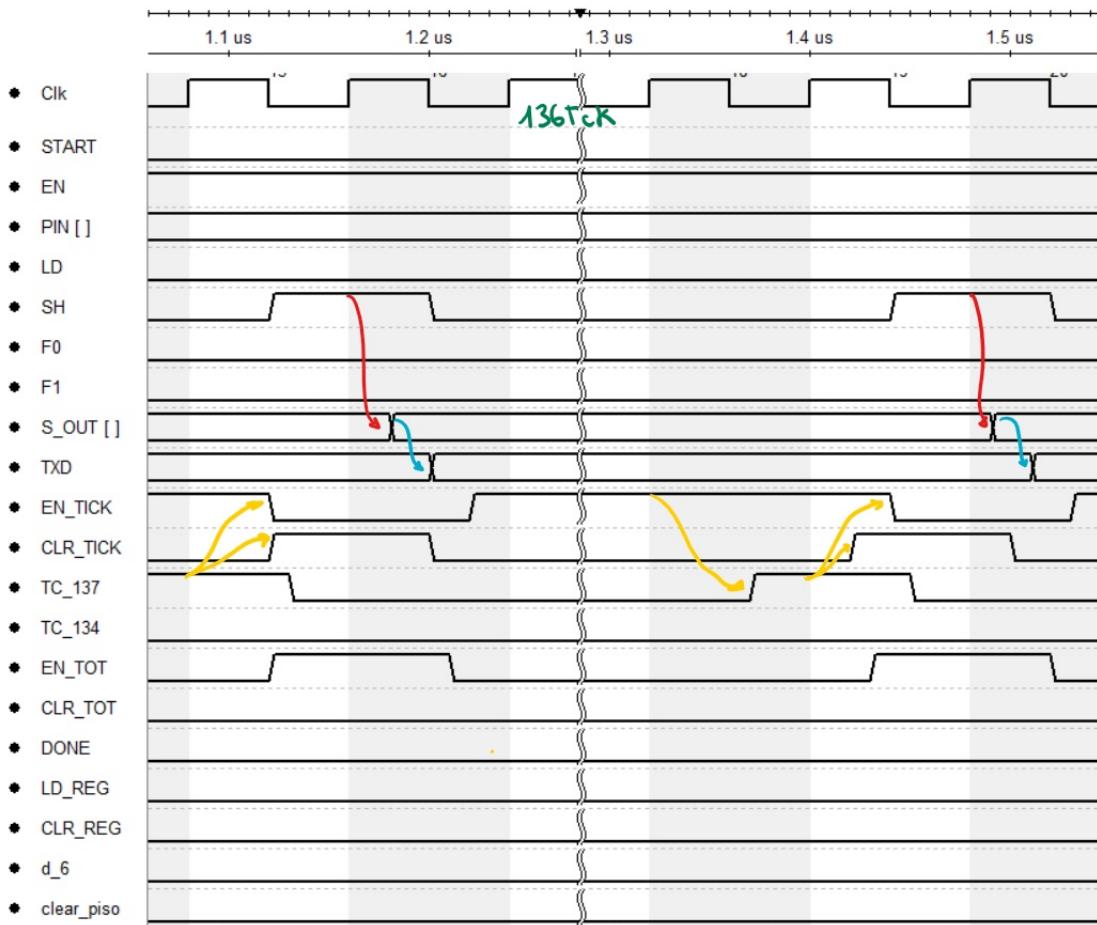


Figura 9: dato generico

$$T_{\text{sym}} = (16 \text{ MHz}) / (115200 \text{ baud rate}) = 139 T_{\text{ck}}$$

La trasmissione di ogni simbolo dura $139 T_{\text{ck}}$ [contatore conta fino a 137]. Da D_0 a D_6 i dati vengono trasmessi nel modo rappresentato dal timing diagram riportato. Il tempo di simbolo viene garantito grazie all'utilizzo del counter_tick. Appena termina la trasmissione di un simbolo, inizia quella del successivo grazie all'abilitazione del segnale di controllo SHIFT. Al termine della trasmissione di ogni simbolo si attiva il segnale EN_TOT del counter_tot, in modo da contare il numero dei simboli trasmessi in uscita, dallo start-bit a D_6.

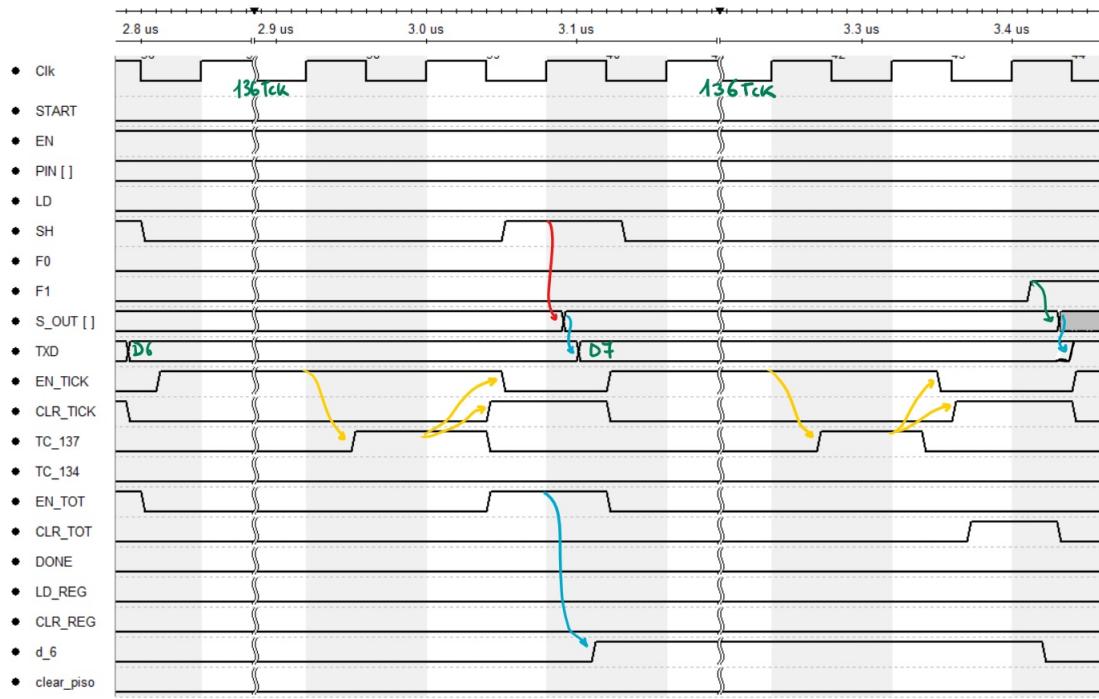
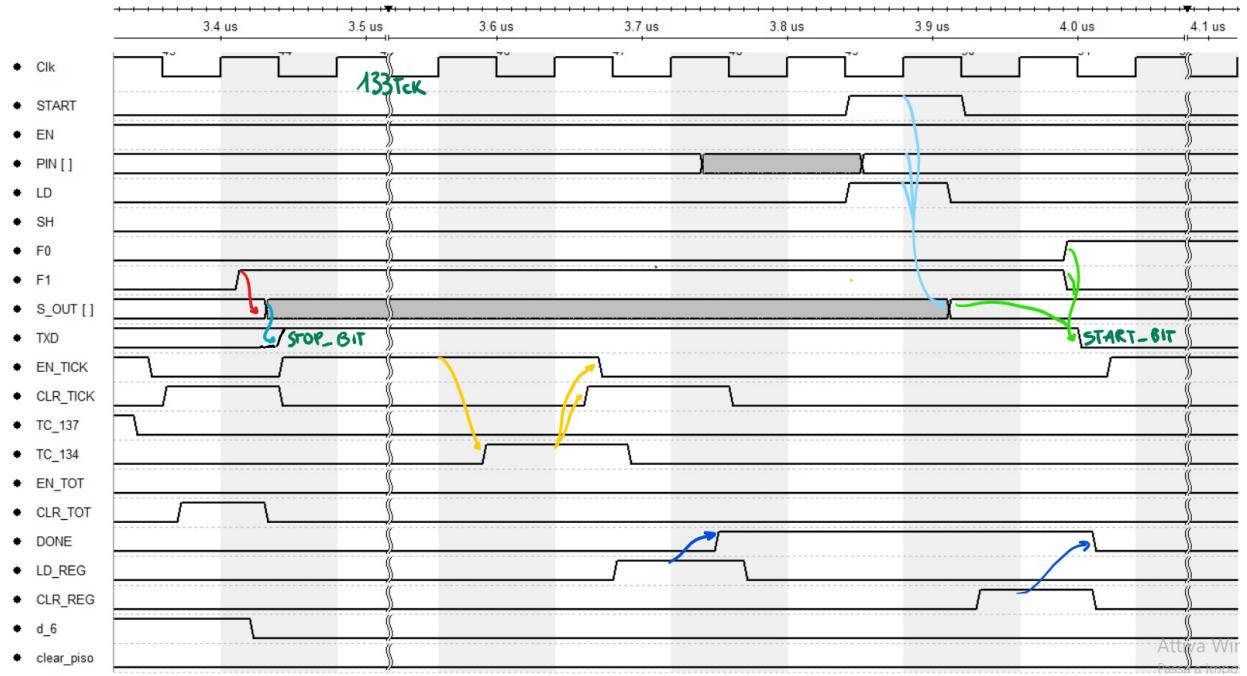


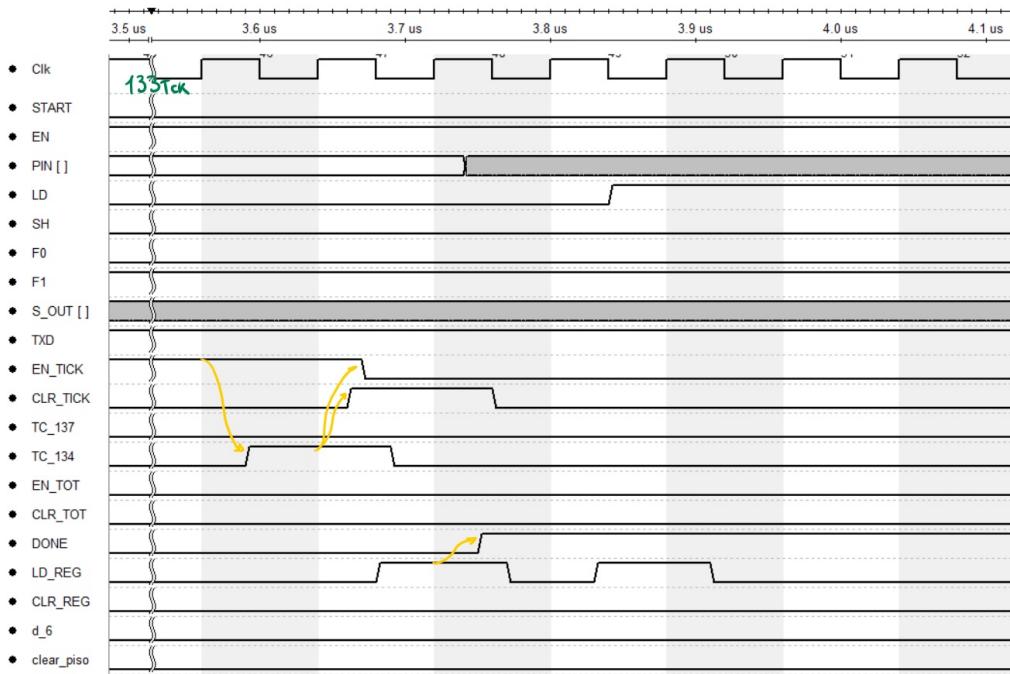
Figura 10: *ultimo shift (avviso con d_6)*

La trasmissione del bit D_7 avviene dopo l'abilitazione del segnale di controllo SHIFT dello shift-register-piso-8-bit per la settima volta [counter_tot conta fino a 8, poiché nel conteggio è compreso anche lo start-bit]. Al colpo di clock successivo il segnale d_6 (terminal count del counter_tot) raggiunge lo stato logico alto per avvisare che sta iniziando la trasmissione dell'ultimo degli otto simboli da trasmettere.

Figura 11: *stop_bit + nuovo dato*

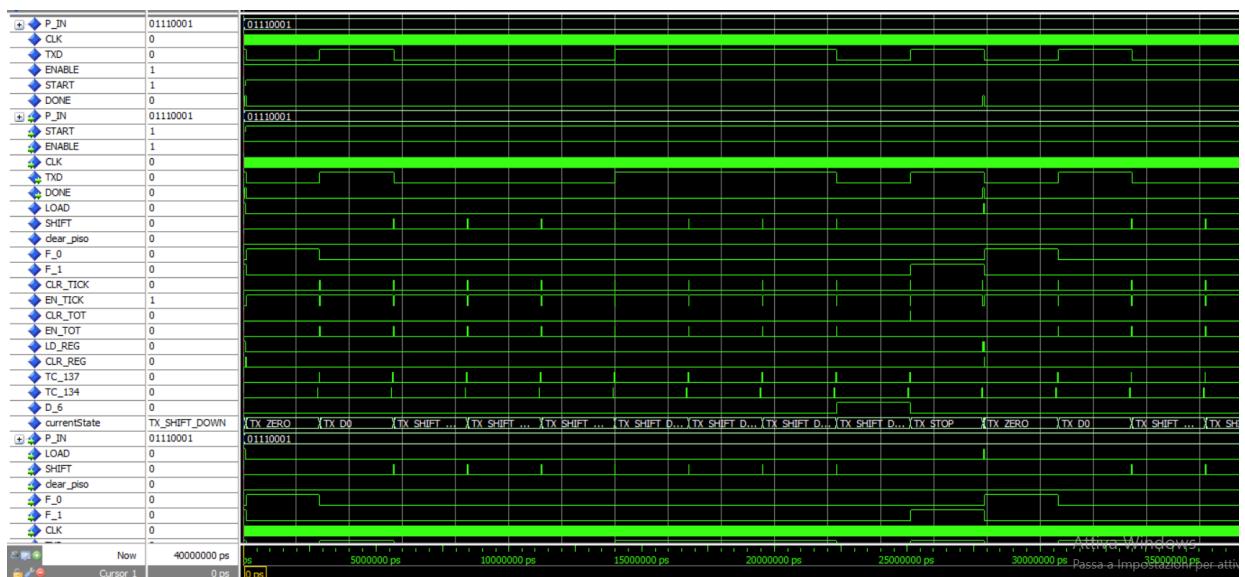
Lo STOP_BIT viene trasmesso imponendo il segnale $F_1 = 1$. In figura 11 è riportata la trasmissione di un nuovo frame, successiva alla trasmissione dello stop bit.

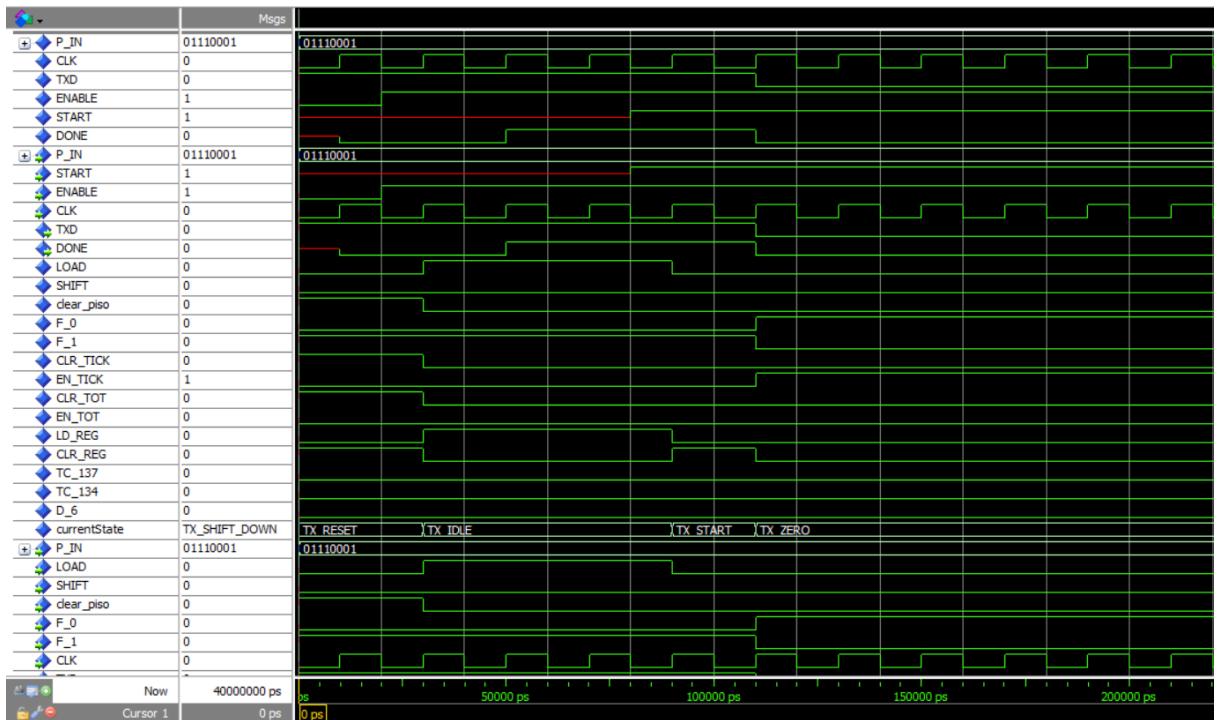
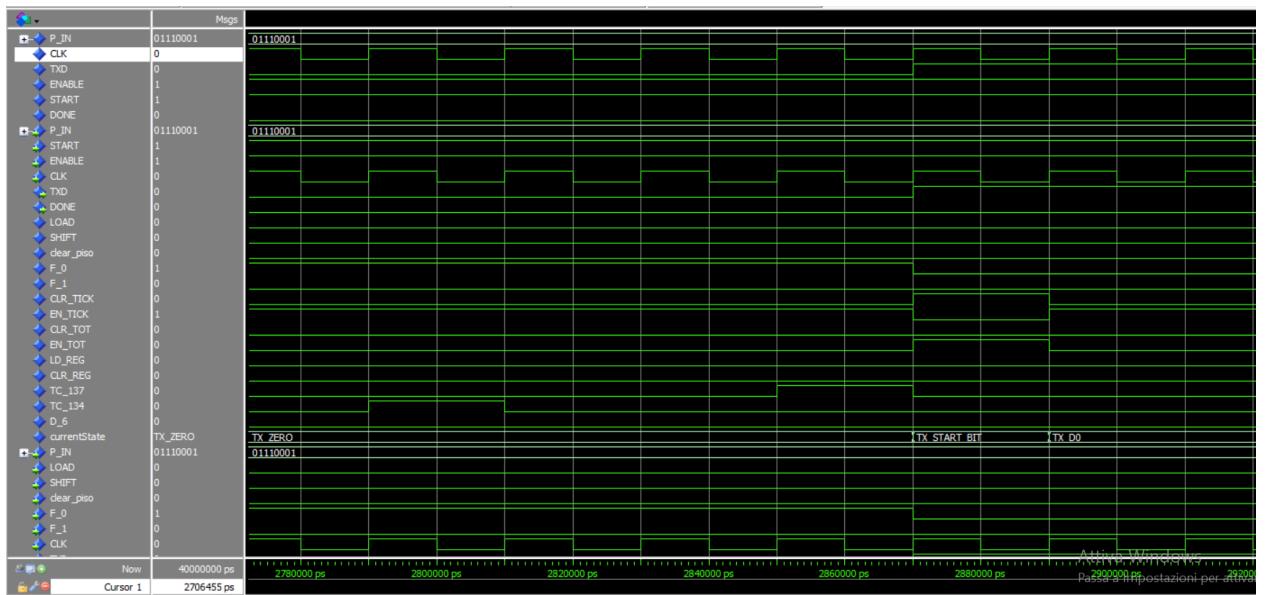
In figura 12 è rappresentata la condizione in cui il blocco trasmettitore ha terminato la trasmissione di un frame e, essendo ancora abilitato, attende un nuovo segnale di START, nello stato di idle, per trasmettere un nuovo dato.

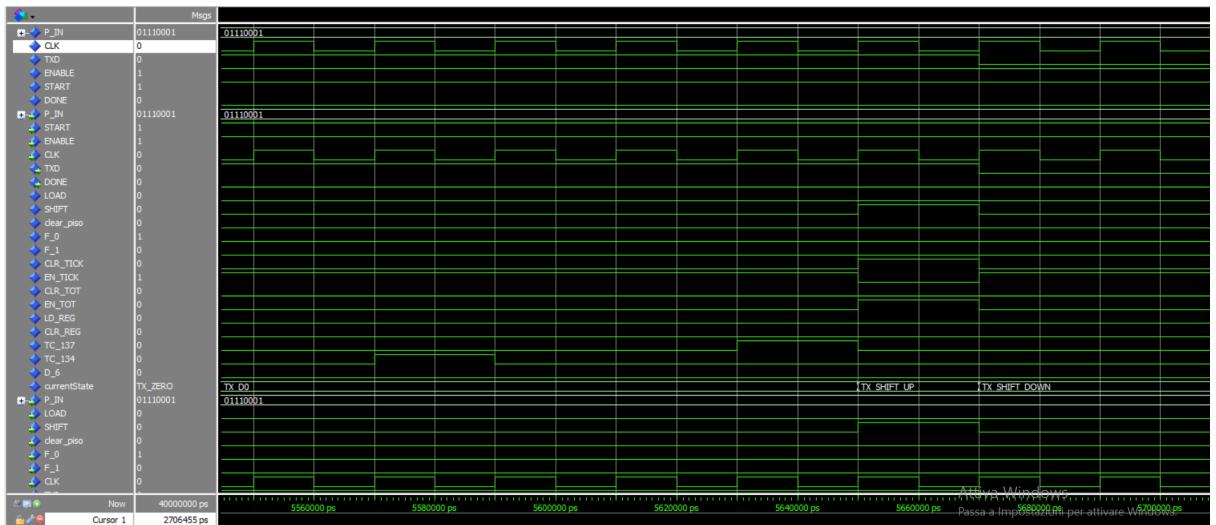
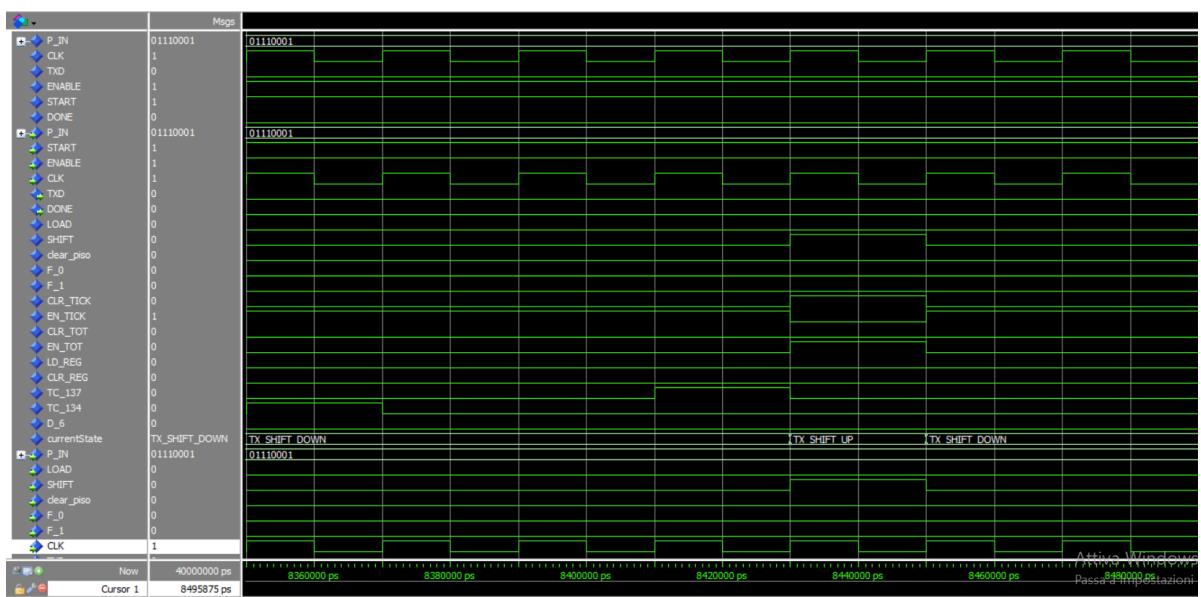
Figura 12: *stop_bit + idle*

2.4 Simulazioni Modelsim

Per testare il blocco trasmettitore progettato, l'hardware è stato descritto in VHDL. Di seguito vengono riportati i risultati delle simulazioni ottenute, congruenti con i risultati attesi.

Figura 13: *esempio di trasmissione di un dato generico*

Figura 14: *inizio della trasmissione*Figura 15: *inizio della trasmissione di D0*

Figura 16: *inizio della trasmissione di D1*Figura 17: *inizio della trasmissione di un dato generico*

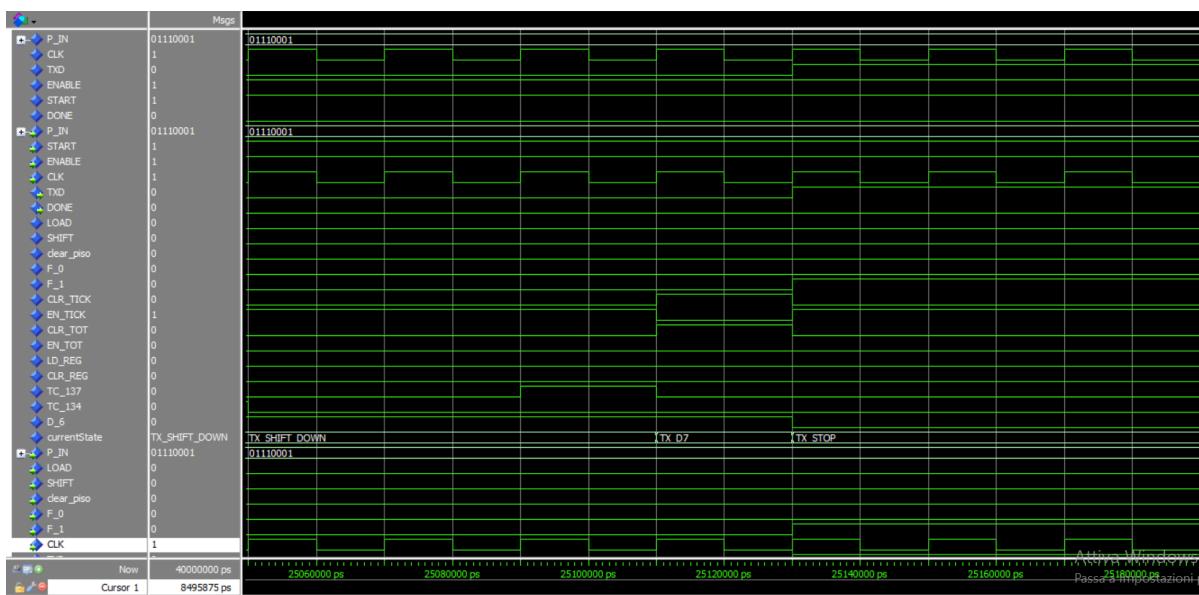
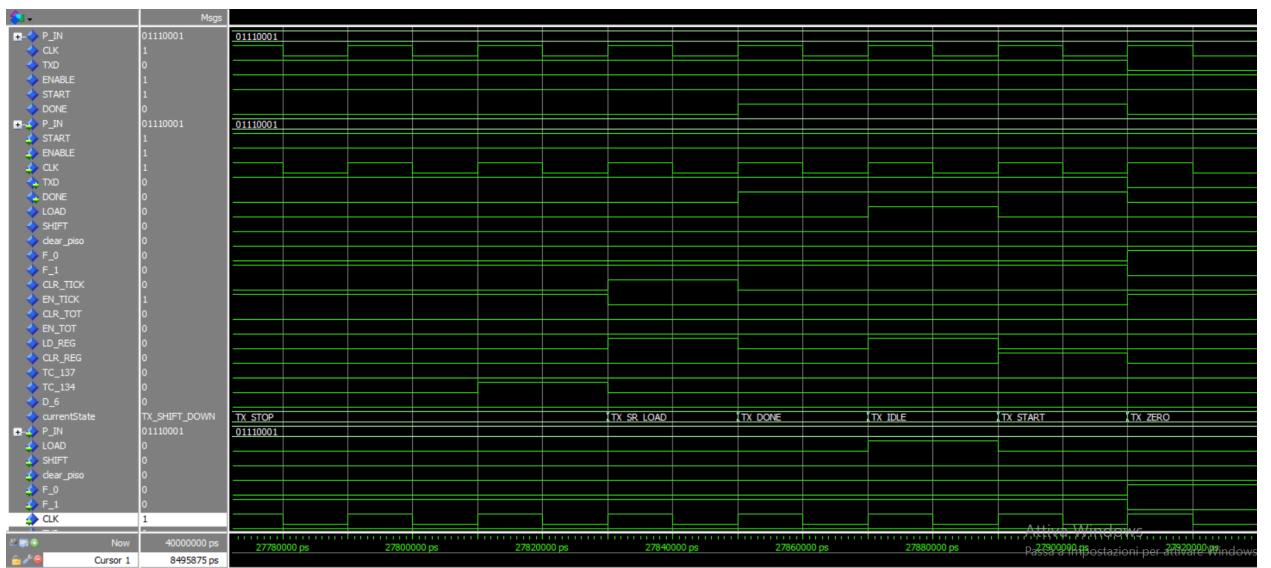
Figura 18: avviso con D_6 

Figura 19: fine trasmissione e inizio di una nuova

3 RX

3.1 Datapath

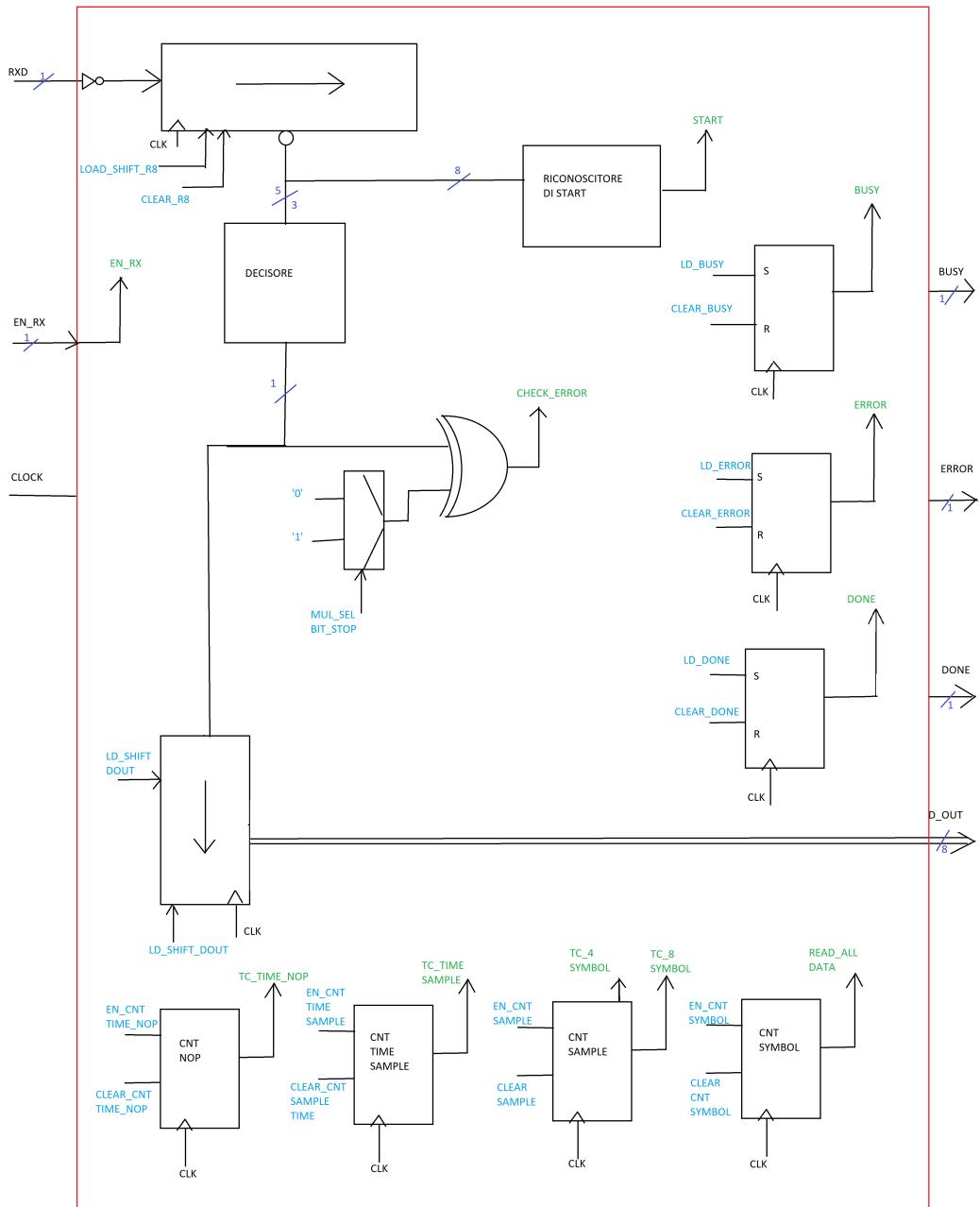


Figura 20: *Datapath del blocco di ricezione*

Il ricevitore è costituito da un primo shift register che serve a mantenere gli 8 campioni del simbolo che si sta analizzando. Il problema del classico shift register è il fatto che qualora si dovesse fare il clear per ritornare allo stato iniziale, lo shift register sarebbe costituito da soli 0, mentre nel nostro caso specifico sarebbe necessario avere tutti 1. Per risolvere questo problema abbiamo pensato di negare l'ingresso di dato e l'uscita in modo tale da mantenere nello shift register i valori negati e negarli all'uscita e di conseguenza, qualora fosse necessario fare un clear si avrebbe tutti 1 in uscita.

Successivamente vi è un blocco riconoscitore di start. Relativamente a questo blocco abbiamo pensato che il segnale di start andasse a 1 quando gli ultimi 4 campioni arrivati sono 0, mentre dei primi 4 almeno 3 sono 1, questo per risentire meno degli eventuali glitch. Il riconoscitore di start è un blocco combinatorio che abbiamo implementato usando la mappa di Karnaugh per quanto concerne i 4 campioni che sono arrivati per primi.

Il blocco DECISORE è stato implementato mediante l'uso della mappa di KARNAUGH, era necessario controllare che se vi fossero stati almeno 2 ingressi a 1, l'uscita del blocco sarebbe stata 1. Come campioni di ingresso abbiamo preferito usare i campioni i campioni il più possibili centrali all'occhio ovvero i campioni 5 4 3, in cui consideriamo il dato 7 l'ultimo dato salvato sullo shift register. Questo perché noi campioniamo ogni 17 colpi di clock anziché 17.36. Ciò potrebbe portare a errori di campionamento nel caso in cui ogni campione fosse preso ogni 17 colpi di clock, ma questo problema viene risolto aggiungendo degli stati di NOP all'interno della macchina a stati

Per il controllo dell'errore viene usata una porta XOR che abbia come ingresso sia l'uscita del DECISORE sia l'uscita di un multiplexer con ingressi 0 e 1, che viene selezionato a seconda del simbolo che siamo utilizzando, ovvero se dobbiamo controllare il simbolo di START o di STOP. In funzione della presenza o assenza di errore, sarà poi presente un set e reset che darà segnalazione di eventuale errore.

Sono poi presenti due ulteriori set e reset, che danno segnalazione di DONE e BUSY. Apparentemente i 2 segnali possono sembrare l'uno il negato dell'altro, ma qualora si dovesse accendere la macchina, il dato presente nel registro non è un dato valido e dobbiamo dare segnalazione di questo. Abbiamo poi uno shift register che custodisca il dato ricevuto una volta che questo sia passato mediante il decisore che ne ha deciso il valore da salvare. È anche questo uno shift register con uscita parallela.

Sono poi presenti 4 contatori, costituiti da una parte che aggiorni il valore di uscita ogni qual volta viene campionato l'enable a 1, e una parte che verifichi se si sia arrivati al valore desiderato.

I tre contatori sono:

–un contatore chiamato CNT_TIME_SAMPLE

- un contatore chiamato CNT_SAMPLES
- un contatore chiamato CNT_SYMBOLS
- un contatore chiamato CNT_TIME_NOP

Il contatore CNT_TIME_SAMPLE è un contatore che avvisa quando campionare il dato successivo.

Il contatore chiamato CNT_SAMPLES che conta il numero di campioni in modo tale da avvisare quando si debba passare a campionare il prossimo simbolo e quindi fare le operazioni di valutazione del simbolo analizzato, questo è costituito da 2 terminal counter, uno che viene utilizzato dopo che avviene lo start e un altro per tutti gli altri simboli.

Vi è poi il contatore CNT_SYMBOLS che conta il numero di simboli che sono stati analizzati in modo tale da capire quando si sta valutando il simbolo di STOP.

Infine, l'ultimo contatore si ritiene necessario per capire quanto tempo bisogni aspettare prima di ricominciare a campionare. Si ritiene che questo contatore sia utile per recuperare quei colpi di clock di anticipo che abbiamo precedentemente descritto

3.2 Timing diagram

Per quanto concerne il timing abbiamo preferito suddividerlo in diverse parti in quanto sarebbe risultato illeggibile se fosse stato caricato interamente.

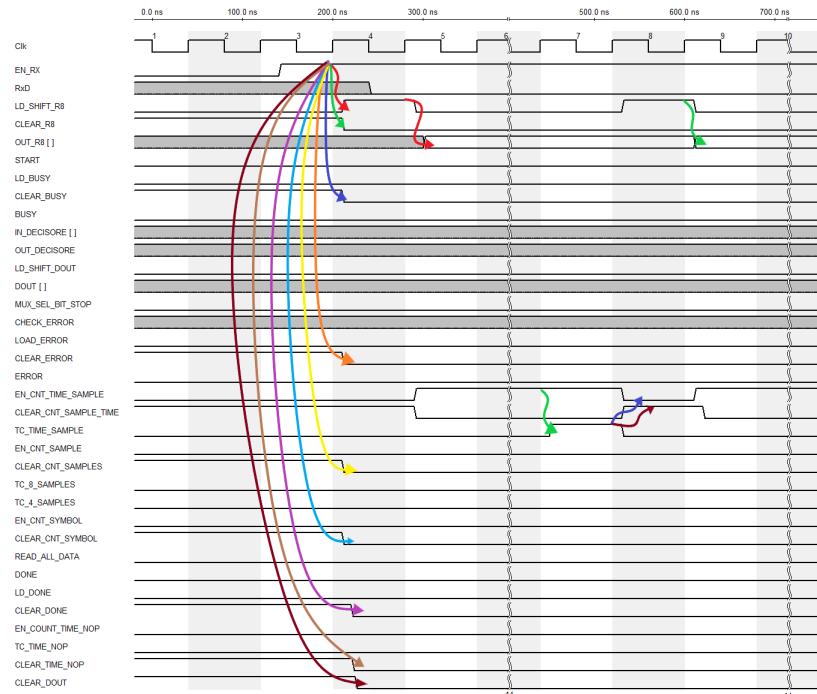


Figura 21: *stato di reset e idle*

Questa immagine mostra il passaggio del segnale di enable da 0 a 1, e il conseguente comportamento della macchina. Come si può osservare, qualora il segnale di enable fosse 0, tutti i registri vengono resettati.

Una volta che viene sentito il segnale di enable a 1, viene caricato il dato che viene dall'esterno mediante il segnale di LOAD. Successivamente viene alzato il segnale di enable del contatore CNT_TIME_SAMPLE necessario per aspettare 17 colpi di clock per poter campionare il prossimo dato.

Una volta sentito il termina counter, viene fatto un clear del contatore e poi un nuovo campionamento. Tutto ciò si ripete in modo ciclico fino a quanto il segnale di start non va a 1.

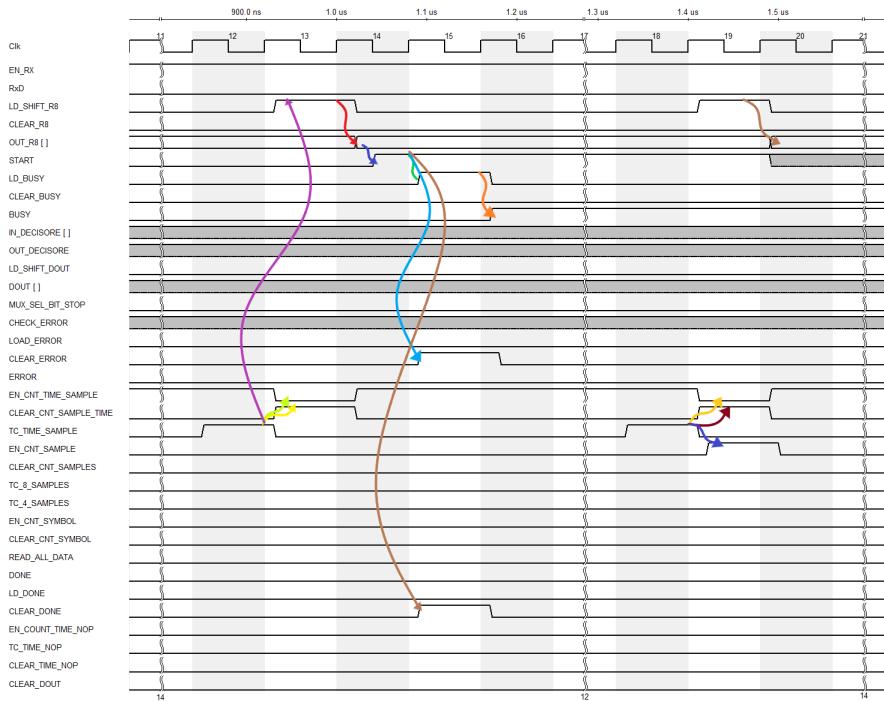


Figura 22: rilevato start

Una volta verificatosi lo START, viene effettuato un clear del set reset relativo al segnale di done, e un set del segnale di BUSY. Una volta verificatosi il prossimo terminal counter, viene alzato anche il segnale di enable relativo al contatore CNT_SAMPLE.

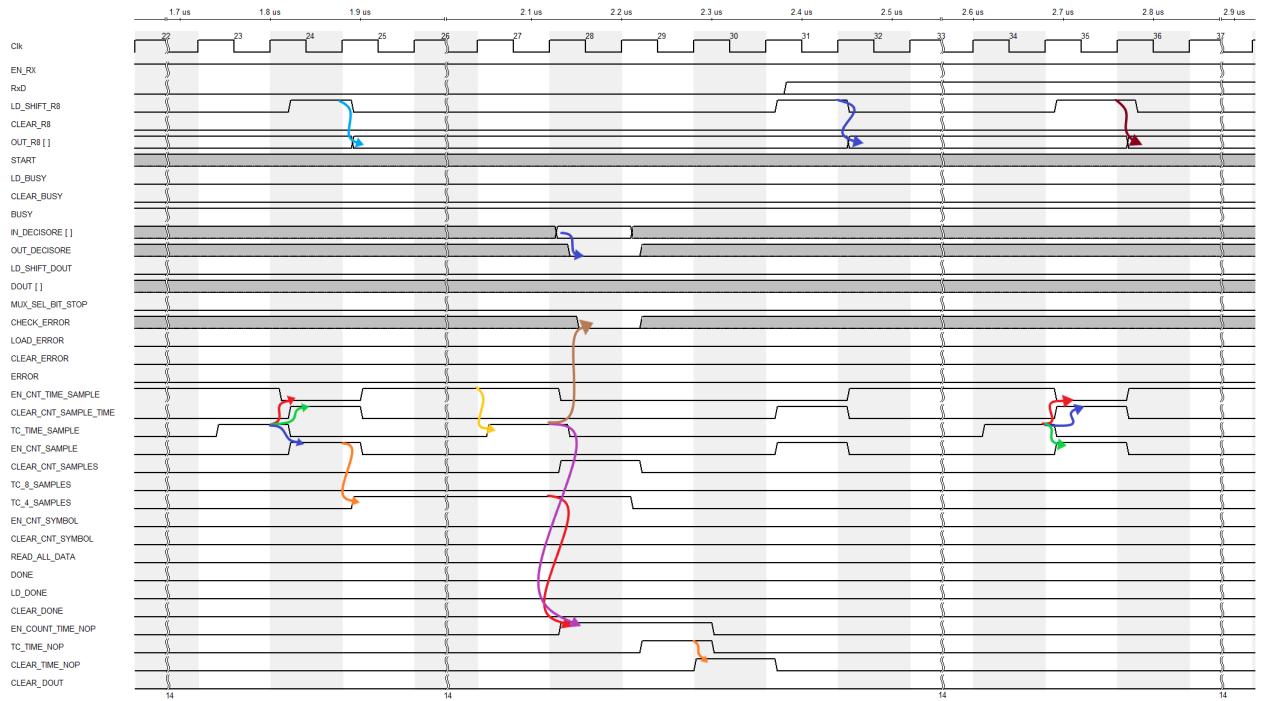


Figura 23: Inizio campionamento del primo bit del messaggio

Qualora il segnale qualora il segnale di terminal counter relativo al contatore CNT_TIME_SAMPLES fosse alto così come il segnale TC_4_SAMPLES, si riterrà necessario controllare il la presenza di eventuali errori nella ricezione del bit di start e viene alzato l'enable del contatore CNT_TIME_NOP, una volta che il terminal counter risulta essere alto e viene sentito dalla CU, si riterrà necessario fare un clear e al colpo successivo campionare e alzare il segnale di enable del contatore CNT_SAMPLES

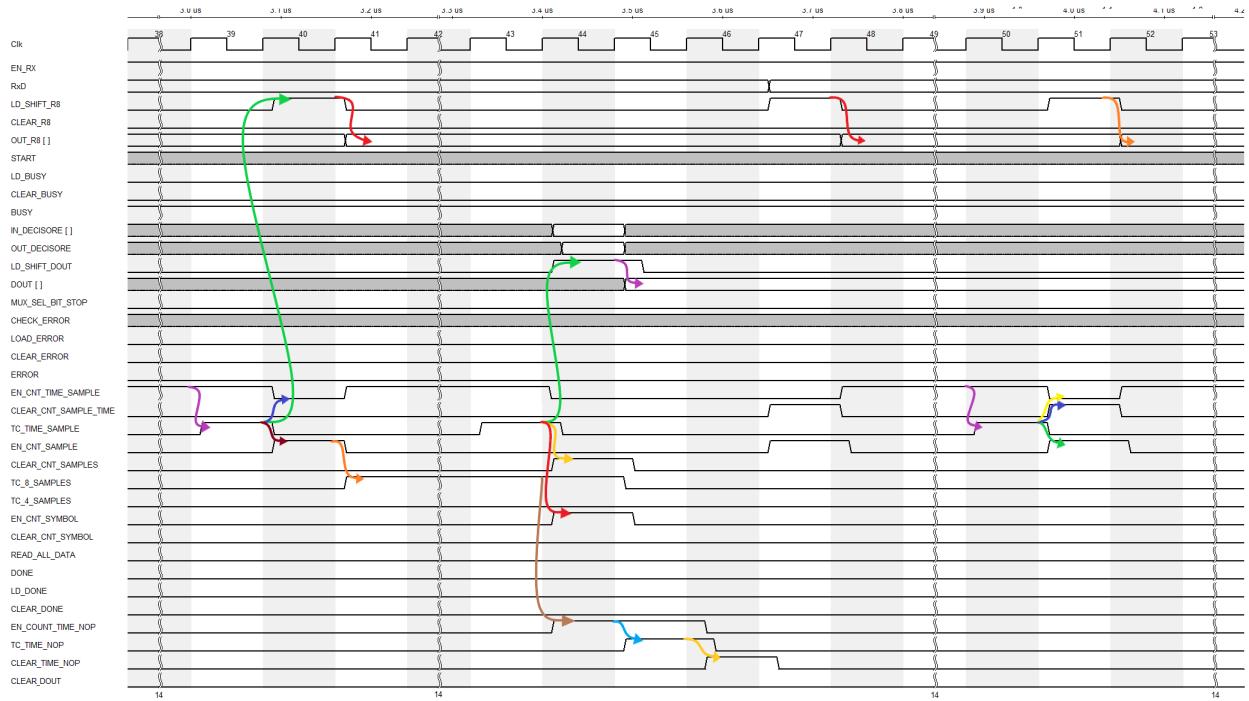


Figura 24: Fine campionamento del simbolo e inizio del nuovo

Come si può osservare in questa immagine, qualora si fossero campionati tutti e 8 i simboli, sarà poi necessario campionare fare uno shift del registro DOUT, e sarà necessario aggiornare il contatore CNT_SYMBOLS

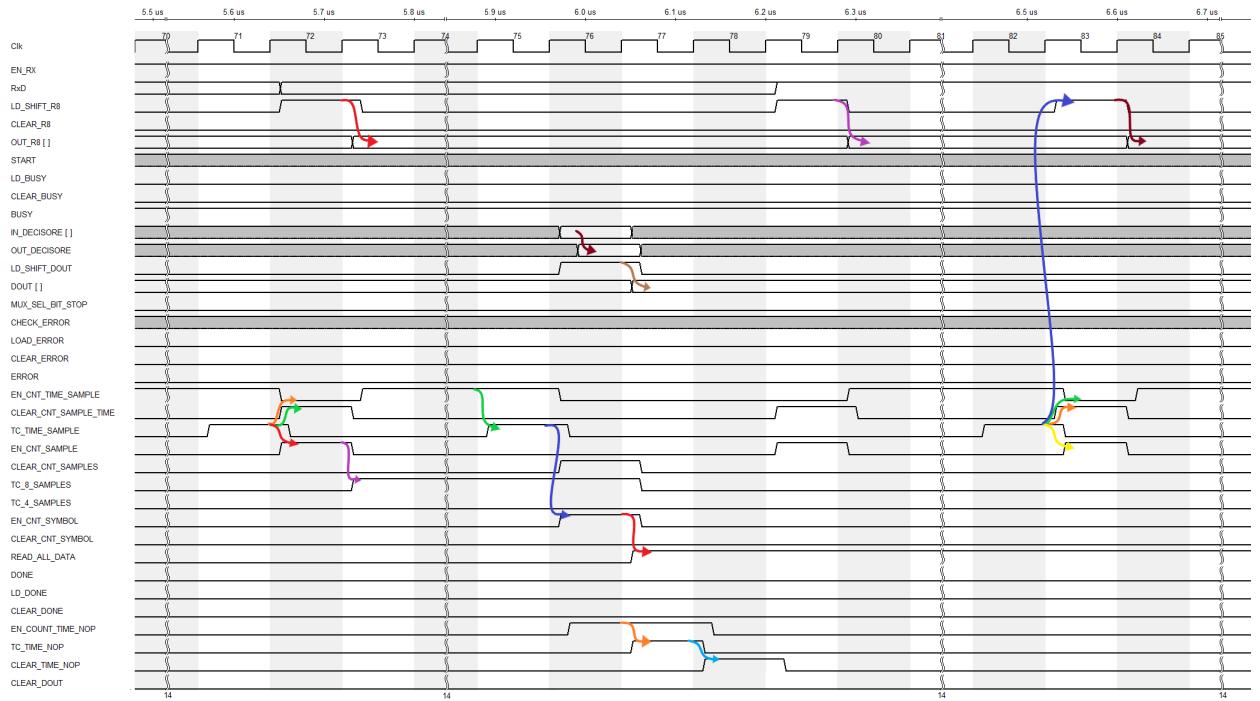


Figura 25: Inizio a campionare il bit di stop

Questa immagine mostra cosa succede una volta che si è effettuato l'ottavo campionamento dell'8 bit del messaggio. Si può osservare come una volta alzato il segnale di enable del contatore CNT_SYMBOL, si alzi il segnale TERMINAL_COUNT. Il campionamento successivo procede come i precedenti.

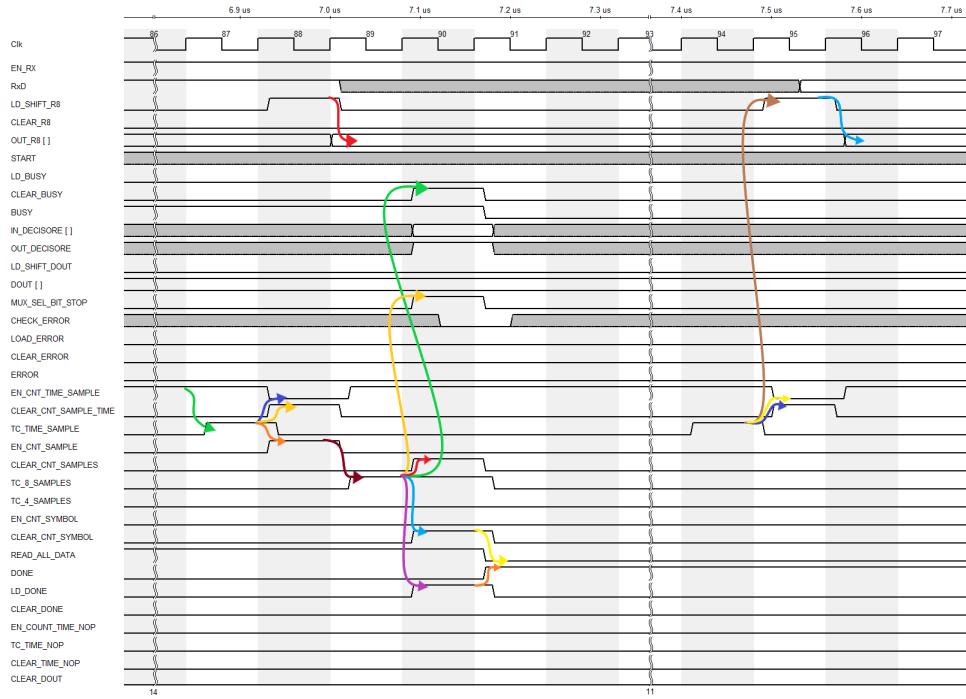


Figura 26: Fine della ricezione del messaggio

Una volta campionato l'intero messaggio, viene verificato che non vi siano stati errori con lo stop bit e qualora non vi fossero problemi, il sistema ritorna agli stati di IDLE.

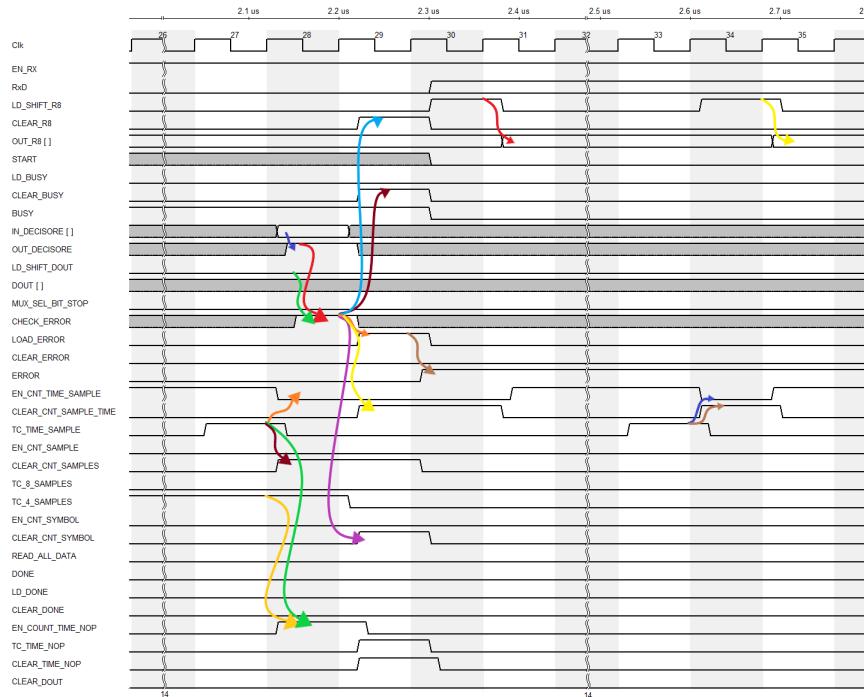


Figura 27: Errore ricezione bit di start

Abbiamo tenuto anche conto di eventuali errori nella ricezione dei dati, ovvero qualora il bit di start o il bit di stop non vengano correttamente ricevuti. Nel caso del bit di start, oltre a segnalare l'errore mediante il segnale di load error del flip flop SR corrispondente, si rende necessario fare in clear dei vari registri in cui viene mantenuto il dato.

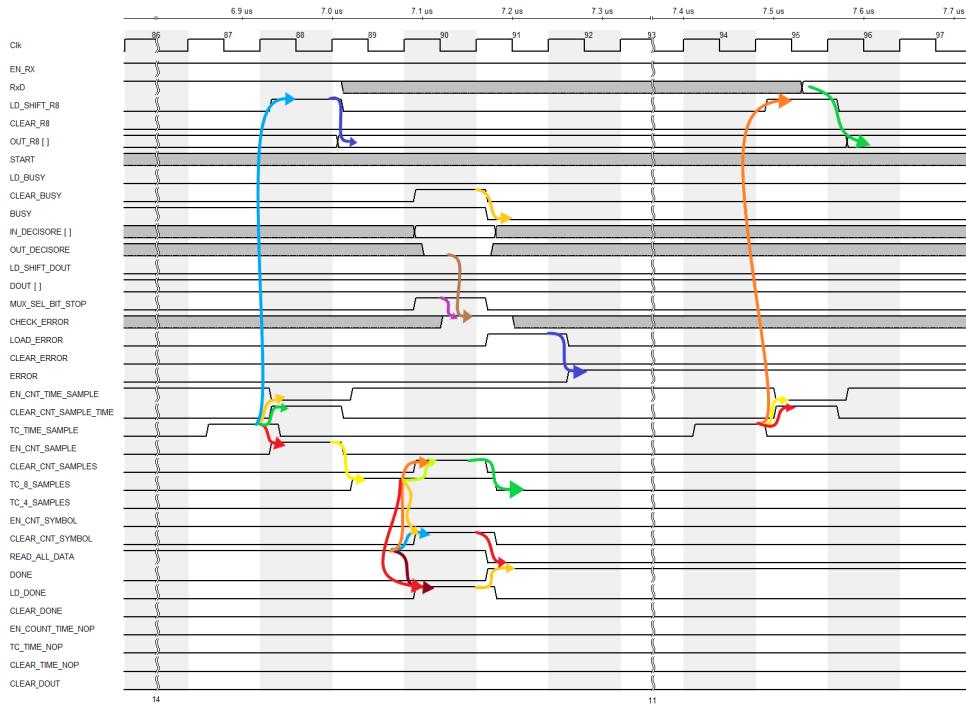


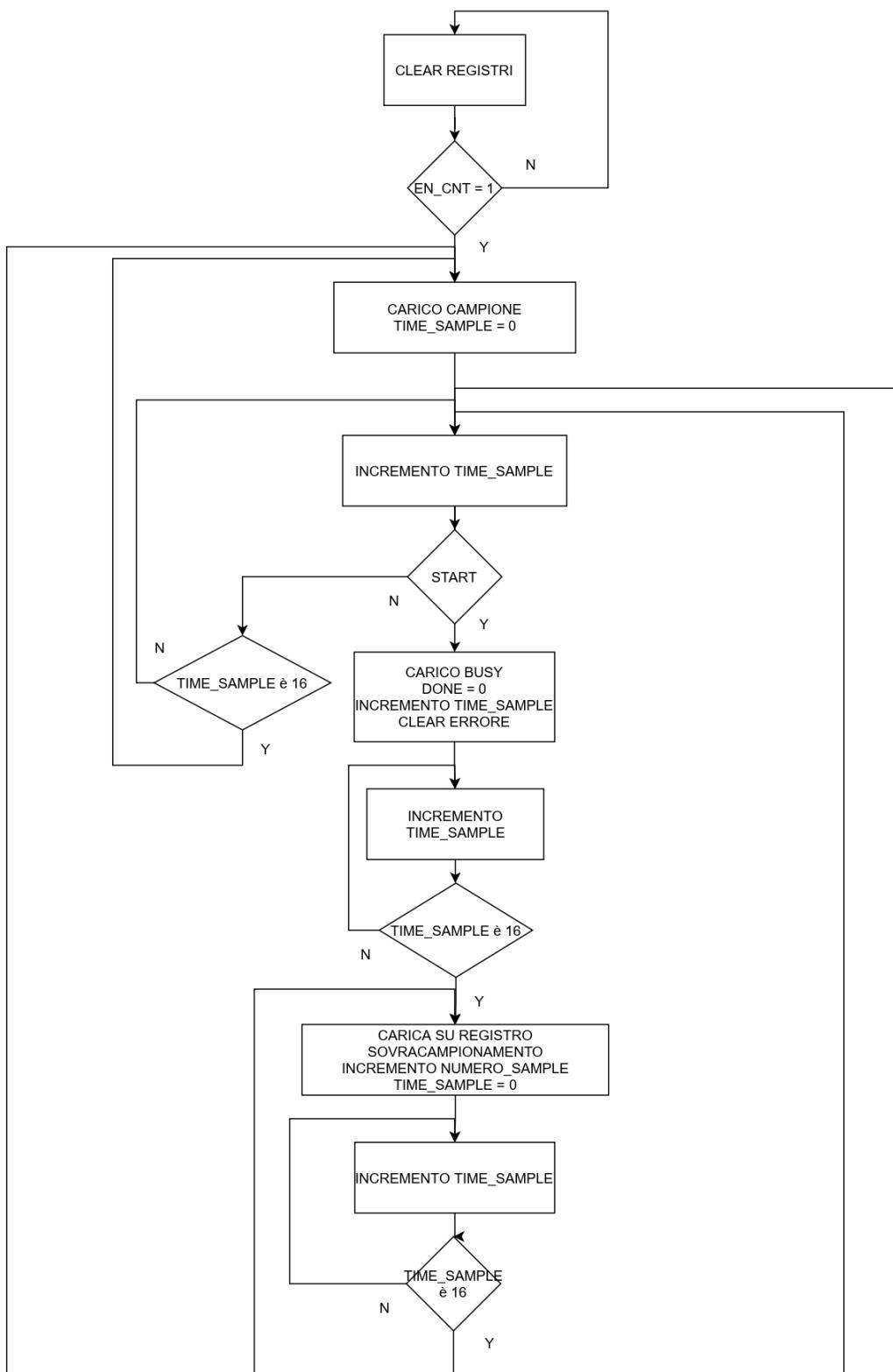
Figura 28: *Errore ricezione bit di stop*

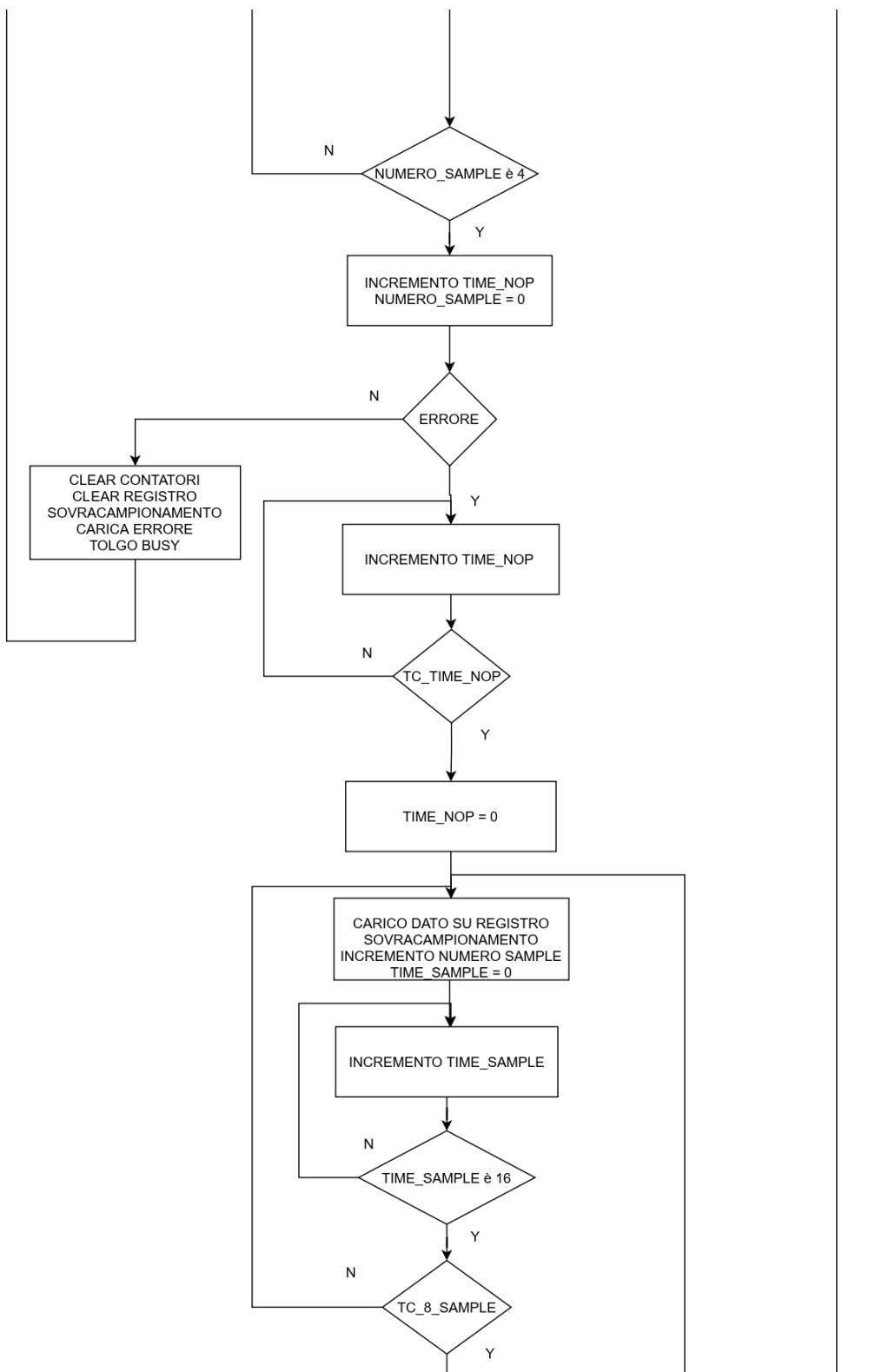
In questo caso invece, il reset sarebbe in qualsiasi modo avvenuto, è però necessario anche in questo caso dare segnalazione dell'errore mediante il load del medesimo flip flop SR, successivamente il circuito va in uno stato di attesa per poi ritornare a campionare.

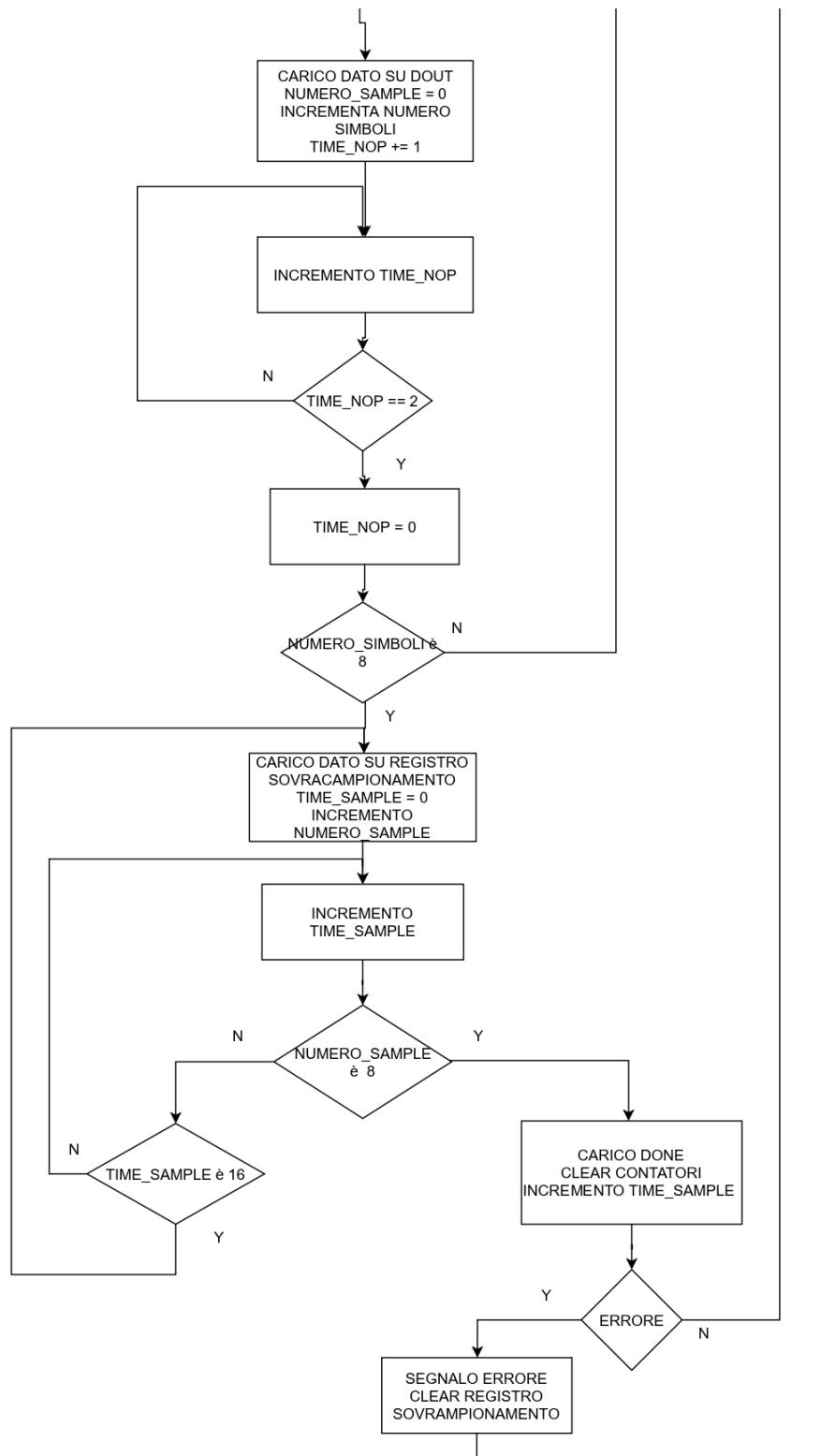
3.3 ASM e CU

Vengono qui riportati gli schemi relativi all'ASM e lo schema dei segnali prodotti dalla CONTROL UNIT. Questi 2 schemi sono conseguenza del timing prodotto precedentemente.

Viene di seguito mostrato lo schema dell'ASM ricavato dal TIMING.

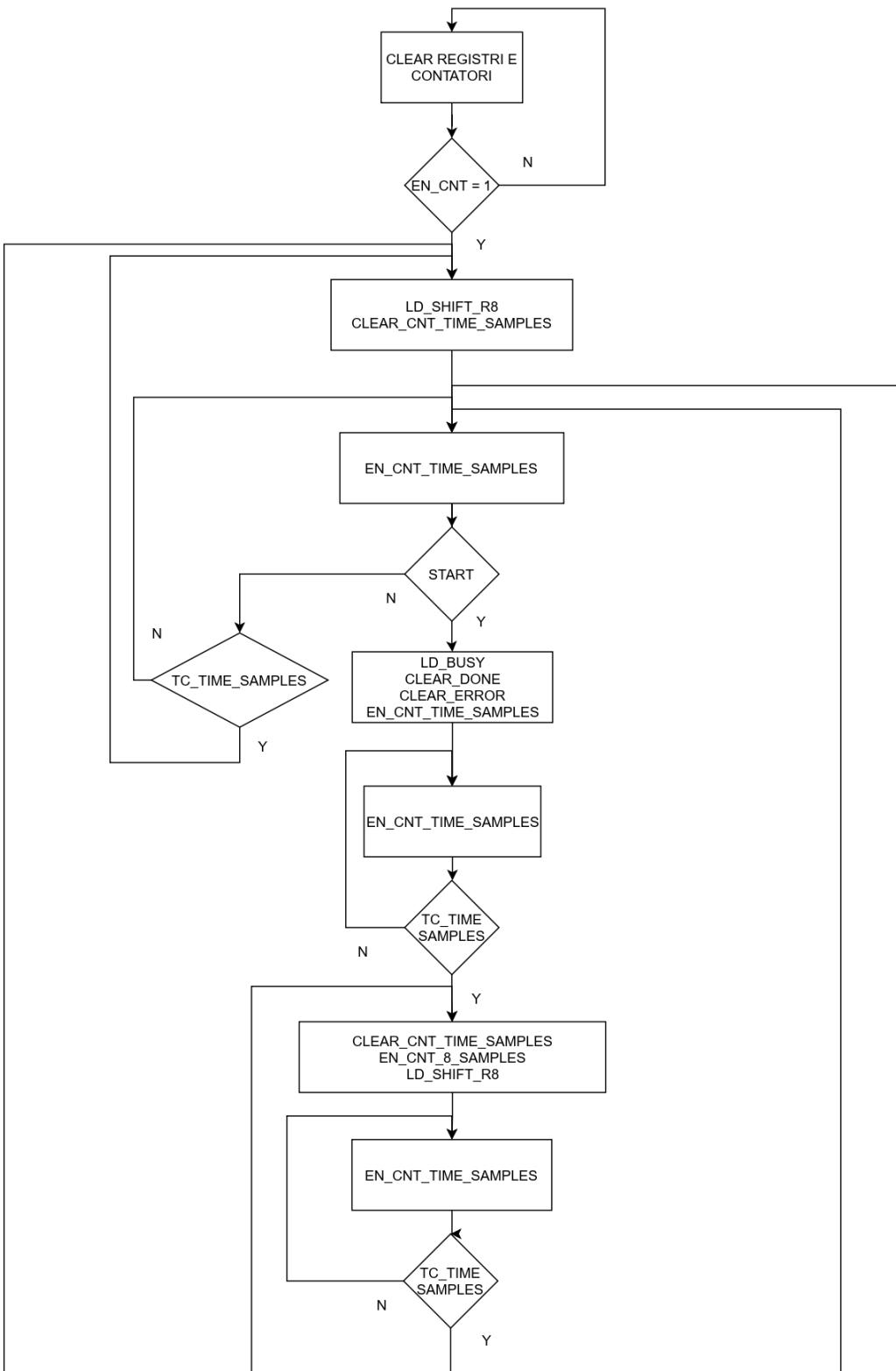
Figura 29: *ASM (parte A)*

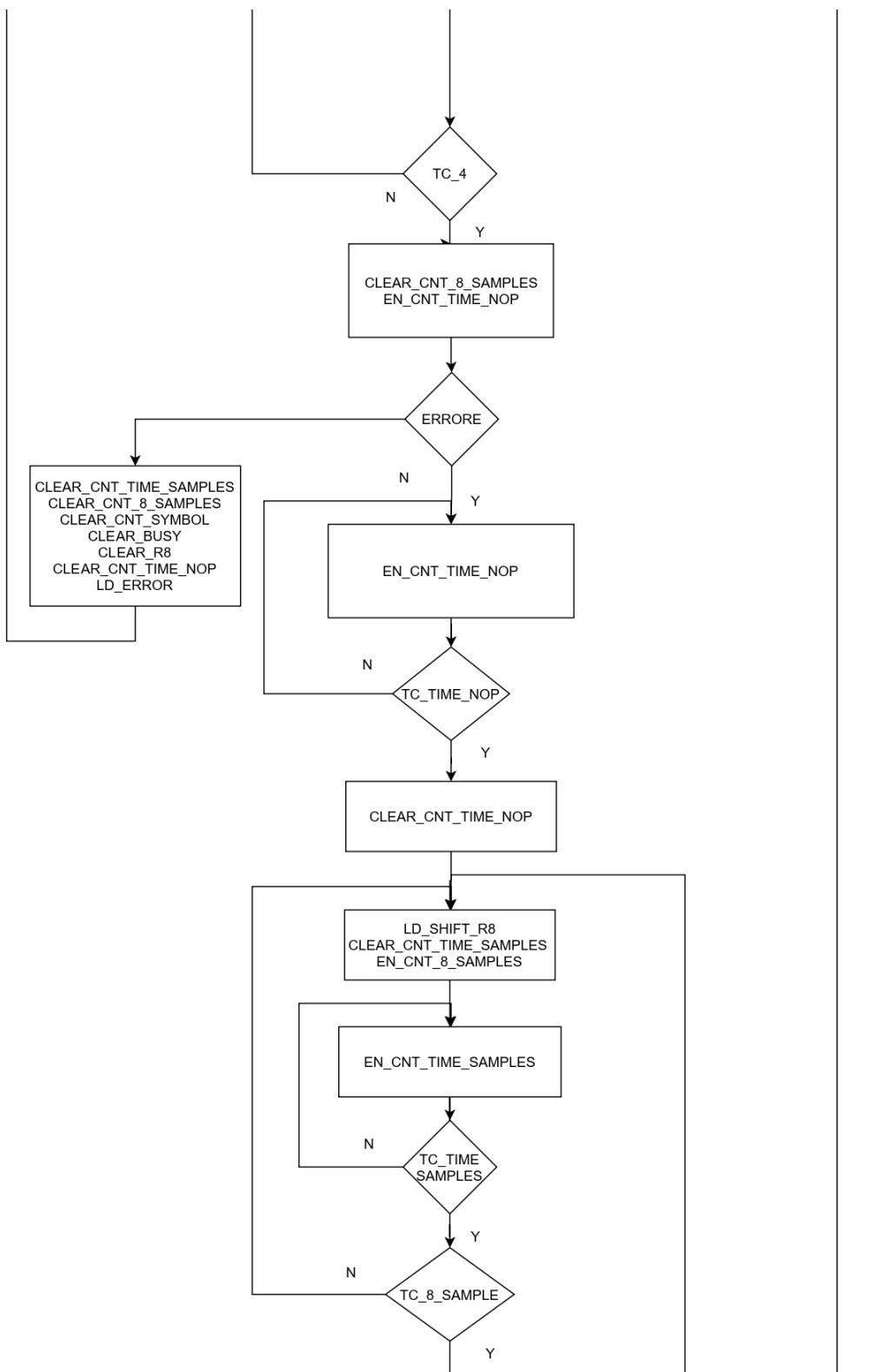
Figura 30: *ASM (parte B)*

Figura 31: *ASM (parte C)*

Per quanto concerne la CONTROL UNIT, abbiamo pensato per ragioni di spazio e

di maggiore chiarezza oltre al fatto di aiutarci nella descrizione dei file VHDL, di scrivere come valori negli stati i segnali che diventano alti. Consideriamo pertanto tutti i segnali di default a 0.

Figura 32: *CONTROL UNIT (parte A)*

Figura 33: *CONTROL UNIT (parte B)*

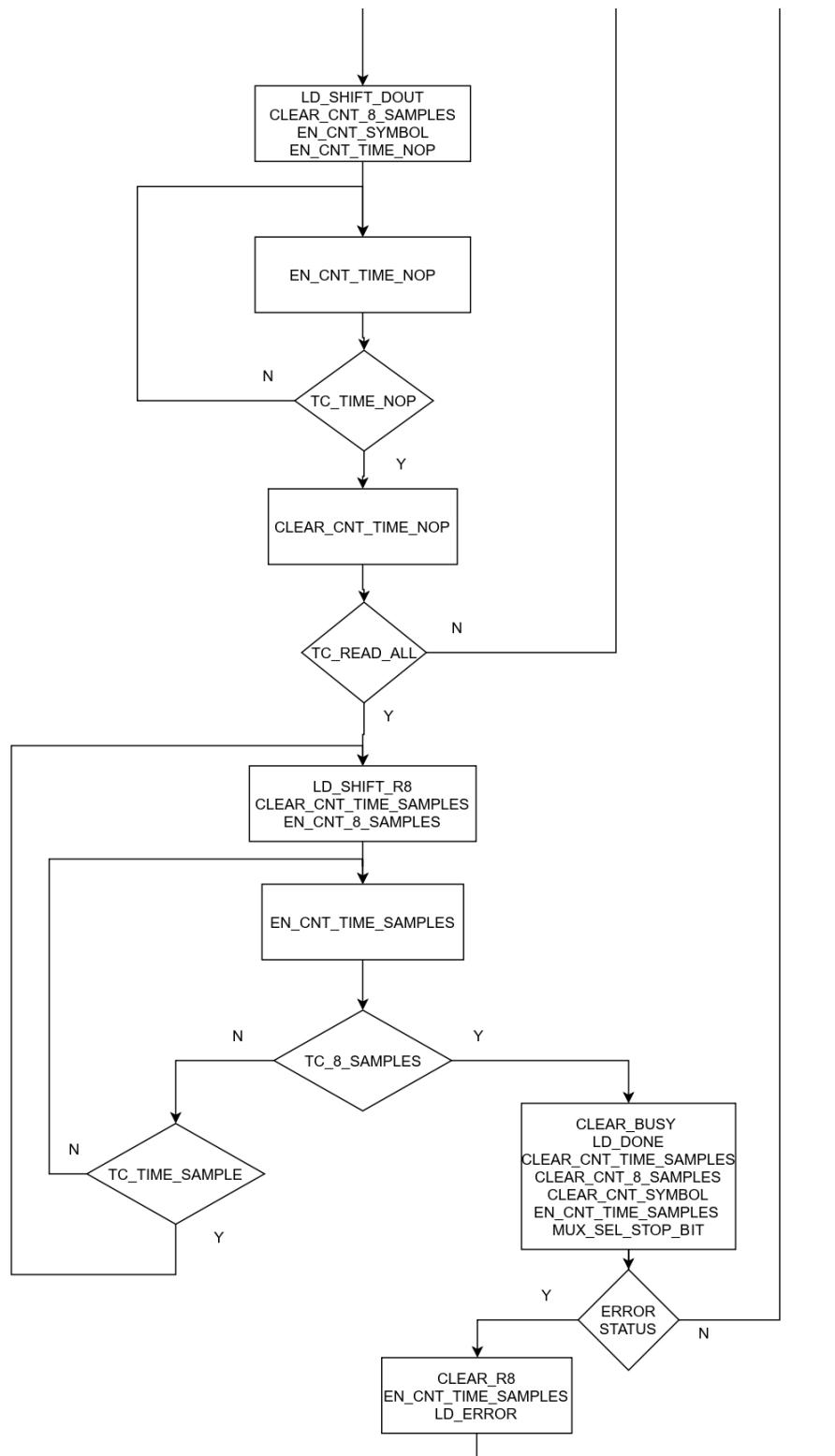


Figura 34: CONTROL UNIT (parte C)

3.4 Simulazioni Modelsim

Una volta descritto in VHDL i vari componenti del Datapath e la Control Unit, abbiamo testato il comportamento del ricevitore. I risultati trovati coincidono con quelli progettati.

Viene dapprima mostrato il comportamento del ricevitore nei casi in cui vi sia un'errata ricezione del bit di start, una corretta ricezione del messaggio e una errata ricezione del bit di stop

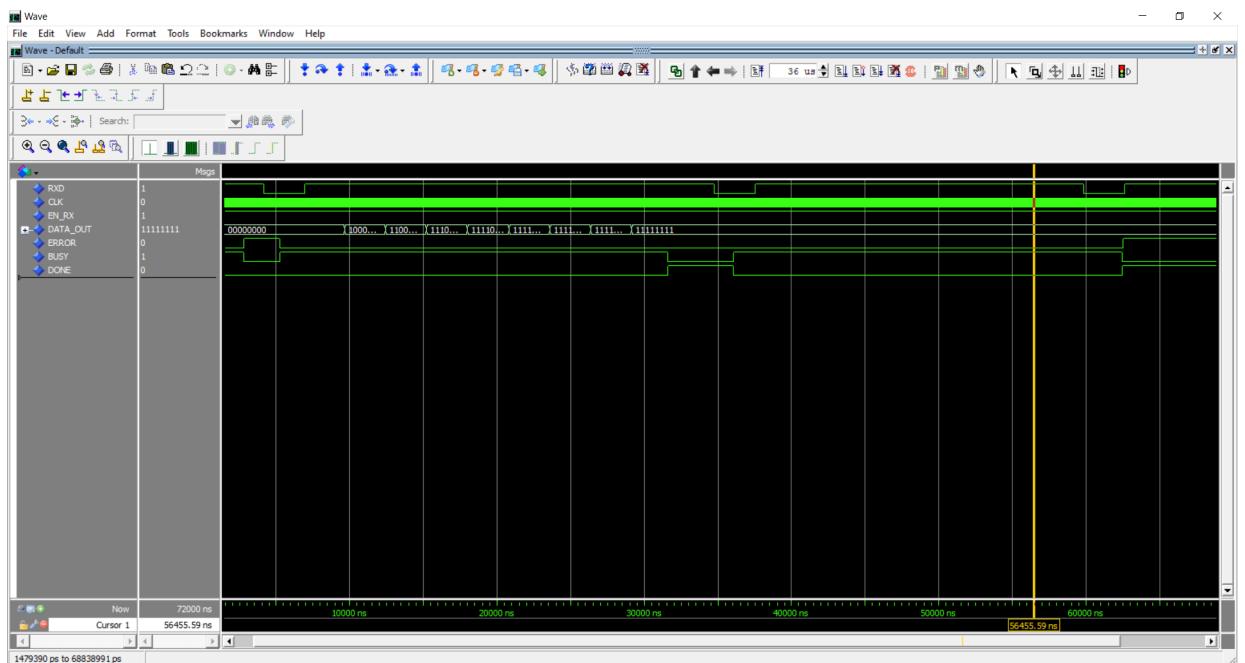


Figura 35: *Simulazione Modelsim del comportamento del ricevitore*

Di seguito sono riportati i soli segnali in ingresso e in uscita dalla CONTROL UNIT, in quanto per ragioni di spazio non era possibile mostrare l'andamento di tutti i segnali

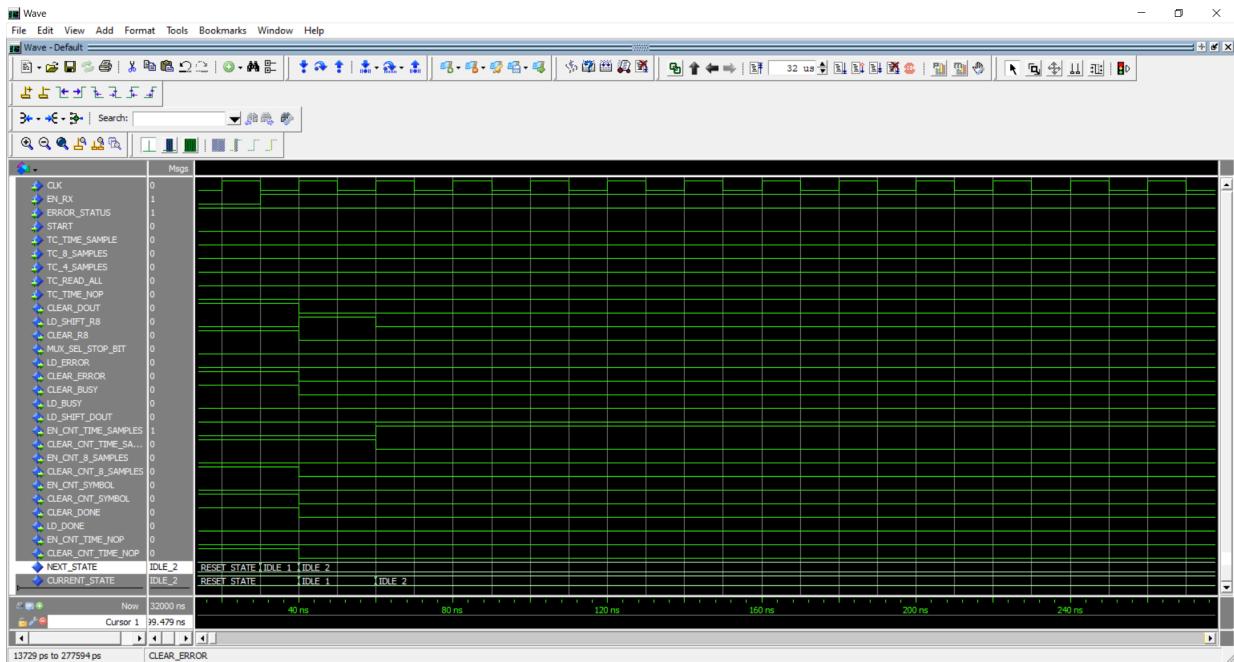


Figura 36: Simulazione Modelsim su transizione RESET IDLE

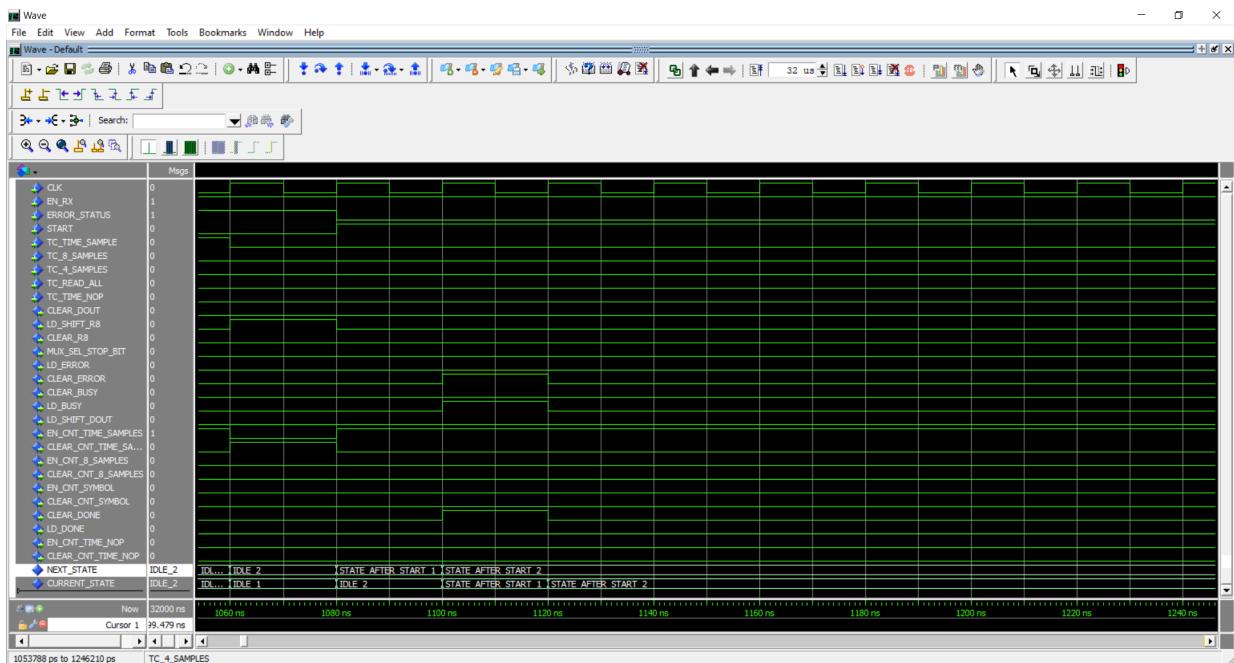


Figura 37: Simulazione Modelsim su transizione basso alto del segnale di START

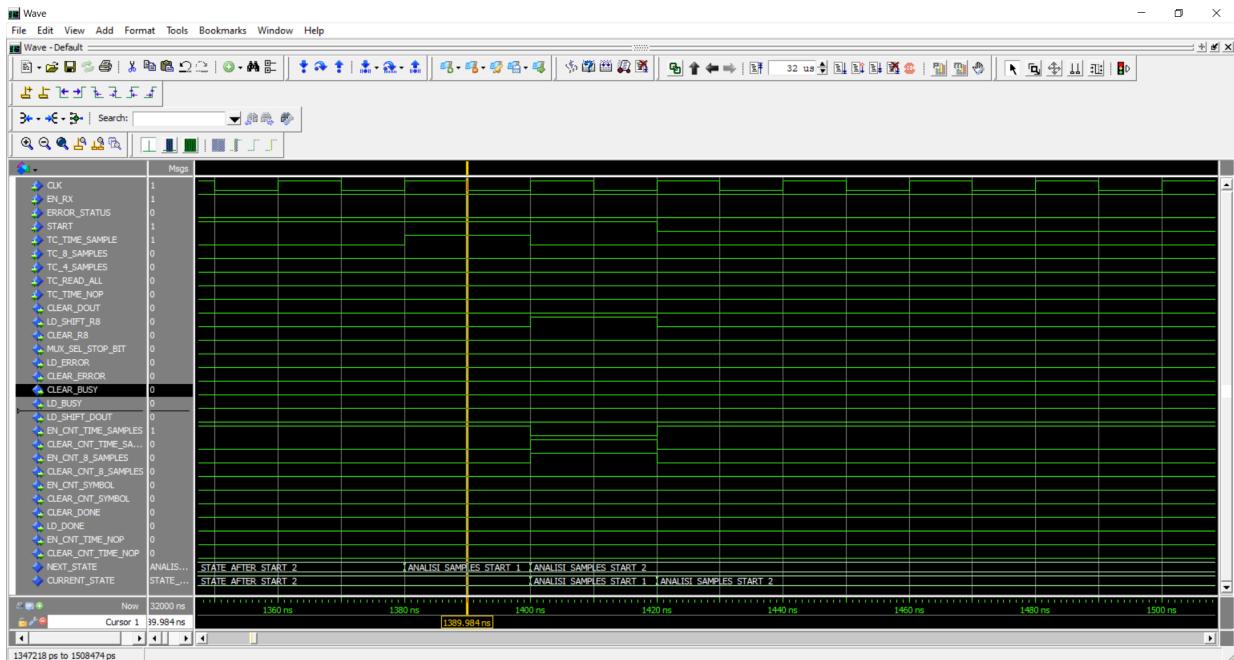


Figura 38: *Simulazione Modelsim relativa agli stati successivi lo START*

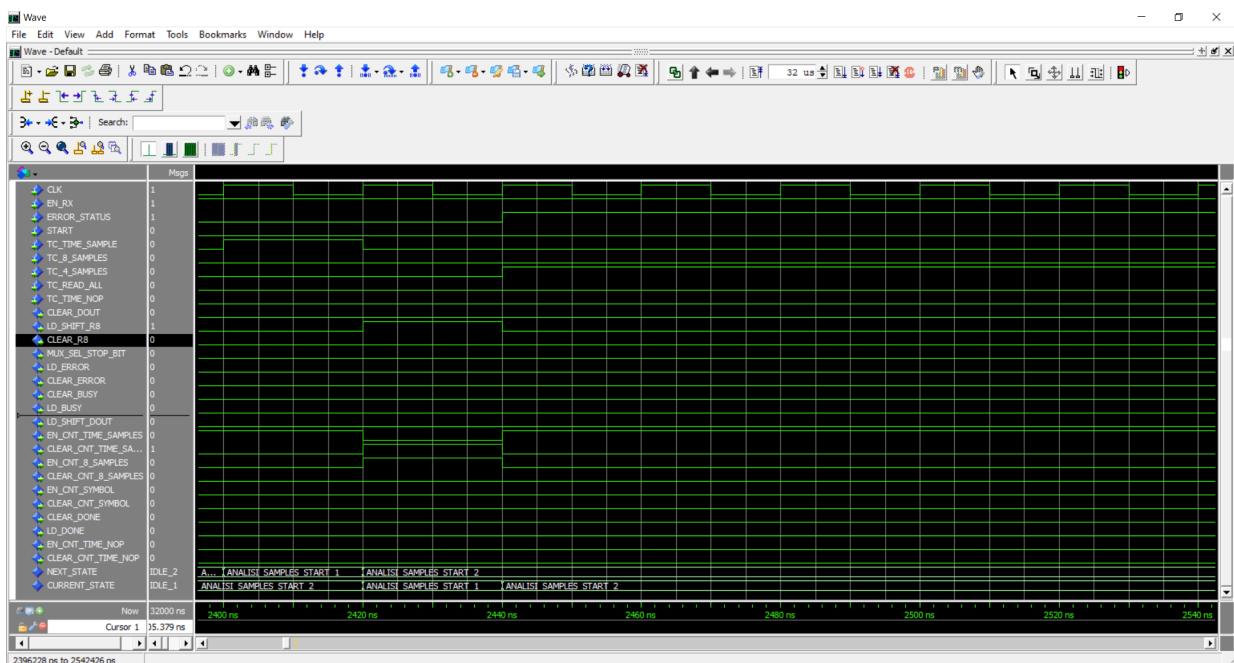


Figura 39: *Simulazione Modelsim in cui viene mostrata transizione basso alto del segnale TC_4_SAMPLES*

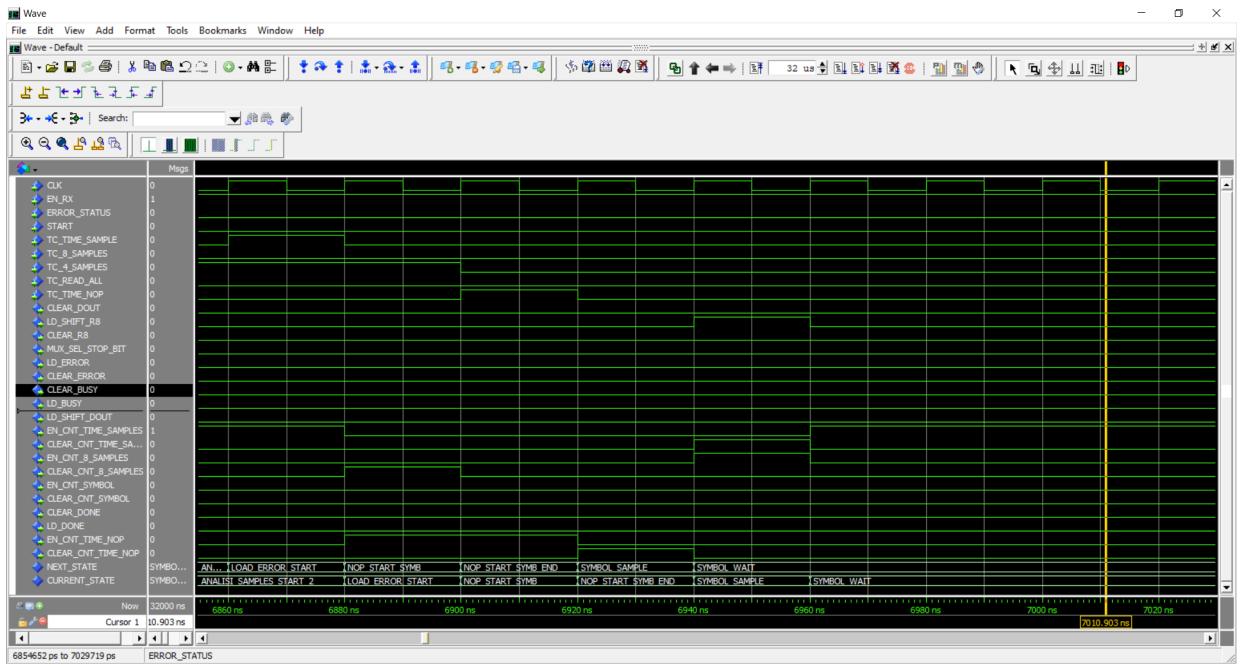


Figura 40: Simulazione Modelsim relativo alla corretta ricezione del bit di start

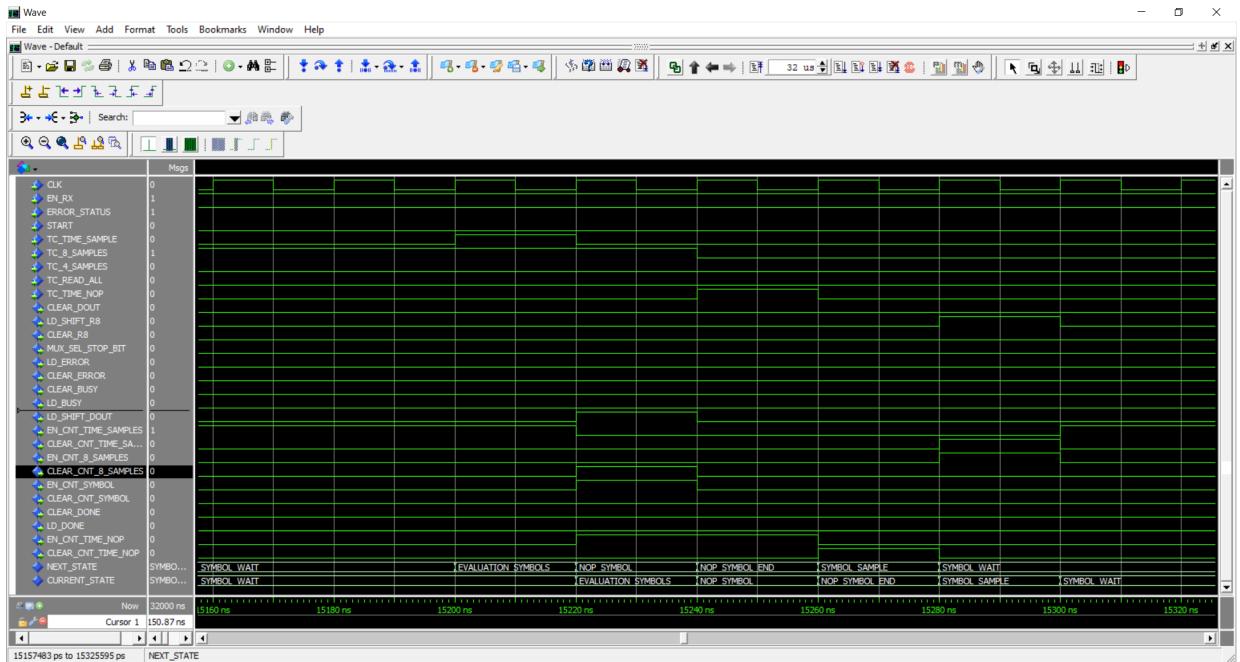


Figura 41: Simulazione Modelsim relativo alla fine della ricezione di un intero simbolo.

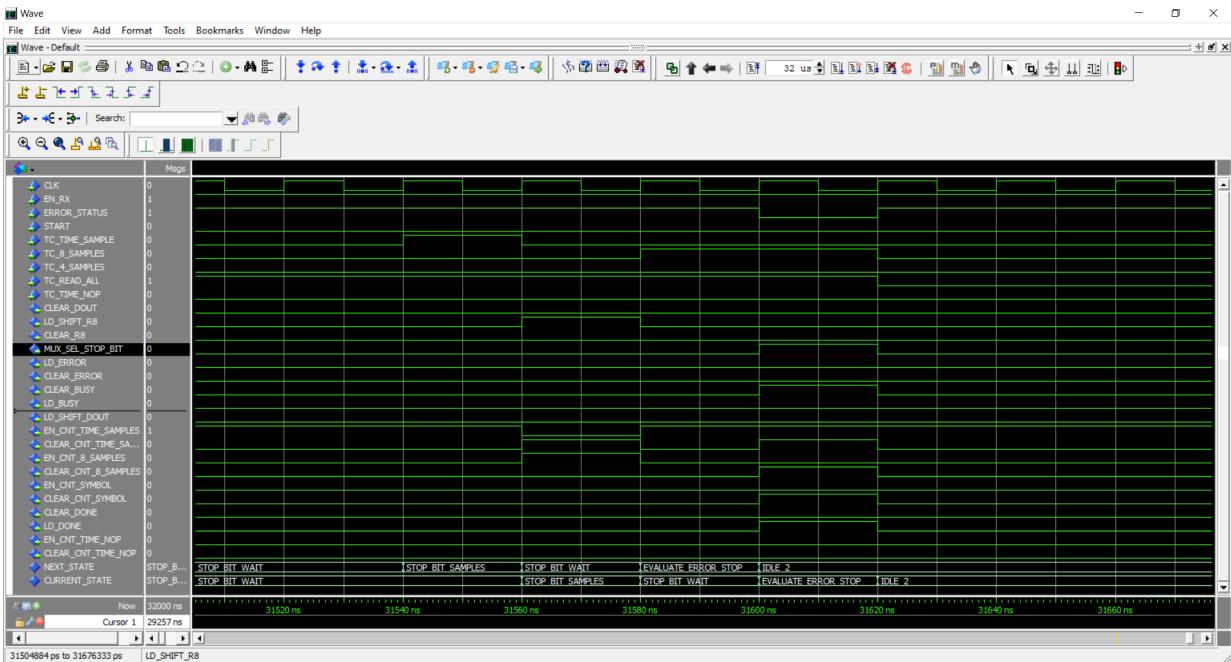


Figura 42: Simulazione Modelsim relativo alla corretta ricezione del messaggio.

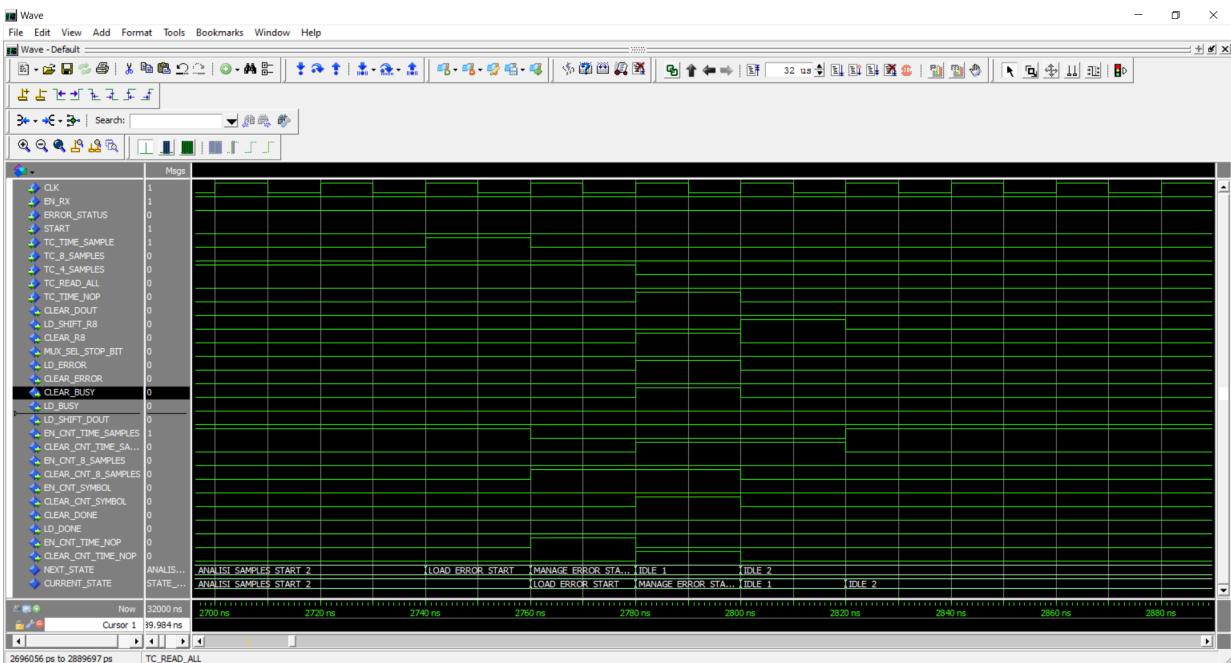


Figura 43: Simulazione Modelsim relativo alla errata ricezione del bit di start.

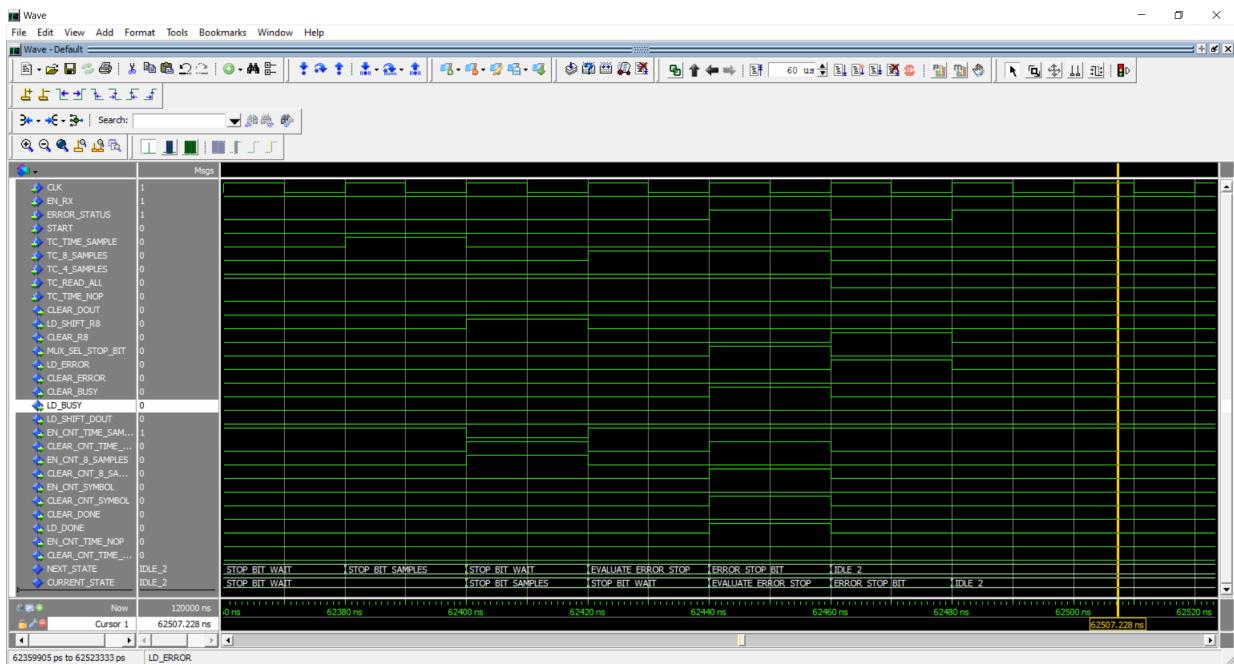


Figura 44: Simulazione Modelsim relativo alla errata ricezione del bit di stop.

4 Bus Interface

4.1 Datapath

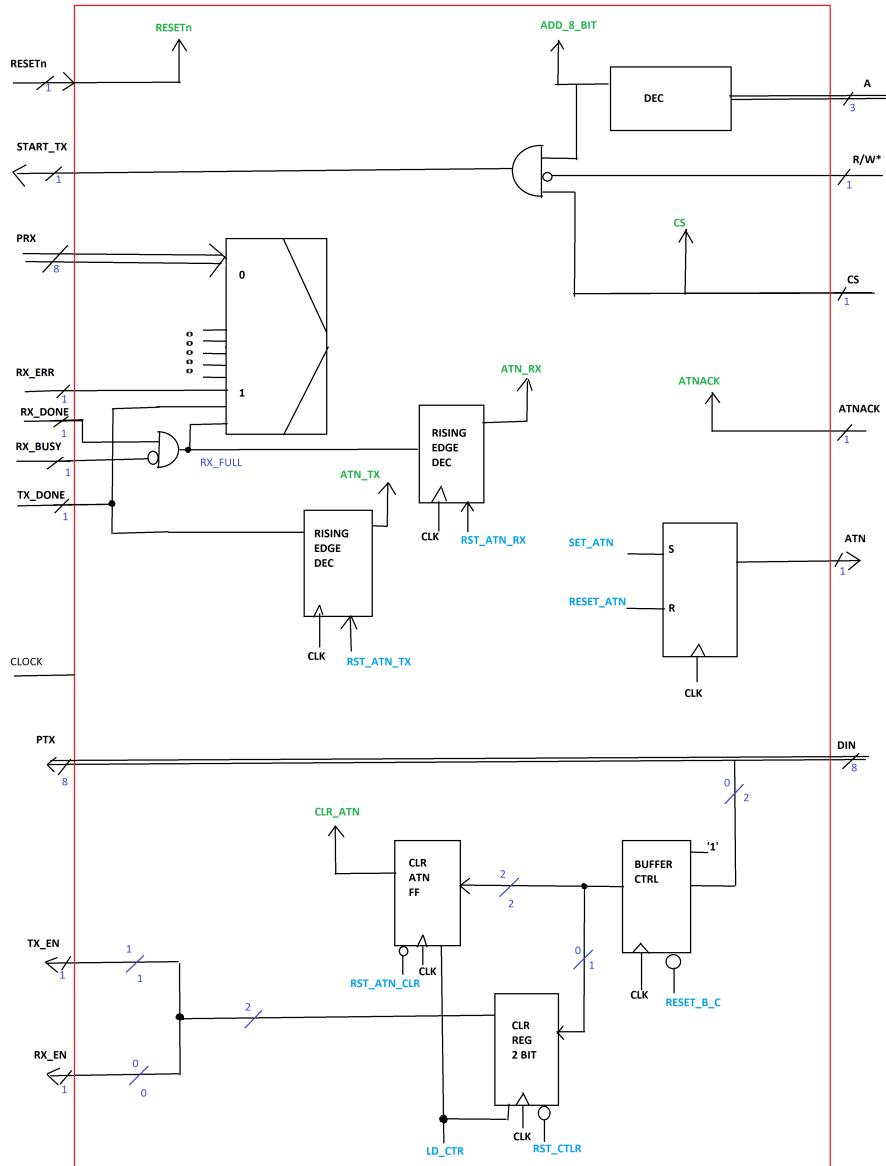


Figura 45: *Datapath della Bus Interface*

Il segnale di START_Tx viene generato tramite un'operazione di AND tra il CS, il R/W* e l'LSB dell'uscita del decoder dell'address. Il dato corrispondente viene inviato direttamente al Tx.

Il bit RX_FULL viene ottenuto con l'AND di RX_ERR*, RX_DONE, e RX_BUSY*;

il multiplexer viene pilotato tramite il segnale OUT_MUX.

La generazione dell'ATN avviene a seguito di un riconoscimento di una transizione in salita o del segnale RX_FULL o di TX_EMPTY; questo riconoscimento è effettuato da dei blocchi appositi, composti da un FF e due porte logiche; una volta che uno dei due avrà la propria uscita su l'uno logico, la CU invierà il segnale di ATN tramite un Flip Flop Set Reset. Il registro contenente il bit ATN verrà resettato nel caso si riceva l'ATNACK o il CLR_ATN.

La scrittura nel registro di controllo avviene tramite un registro, BUFFER_CTRL, il quale ha il segnale di load sempre attivo. L'uscita di questo registro viene inviata al FF del CLR_ATN ed il restante al registro di controllo. Nel caso si rilevi una richiesta di scrittura nel registro di controllo la CU andrà in uno stato di NOP, per permettere al BUFFER_CTRL di propagare il dato ricevuto in uscita, e successivamente abiliterà il load del FF e del registro.

Indirizzi e registri associati:

- “000” -> TXDATA
- “001” -> RXDATA
- “010” -> STATUS REG
- “011” -> CONTROL REG

Bit del registro di controllo:

- DIN(2) -> CLEARATN
- DIN(1) -> TXEN
- DIN(0) -> RXEN

4.2 ASM

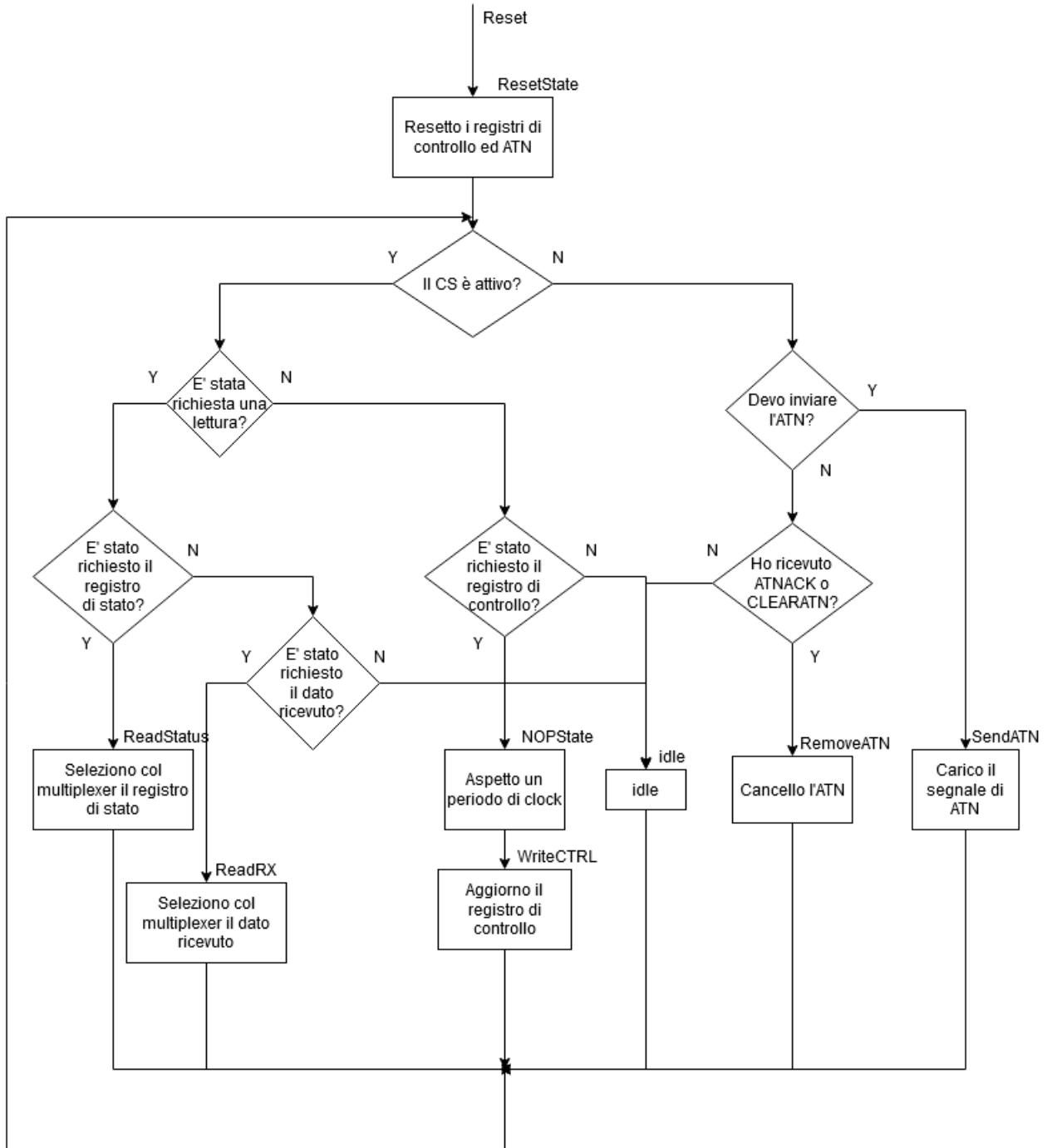


Figura 46: *FSM della Bus Interface*

4.3 Timing diagram

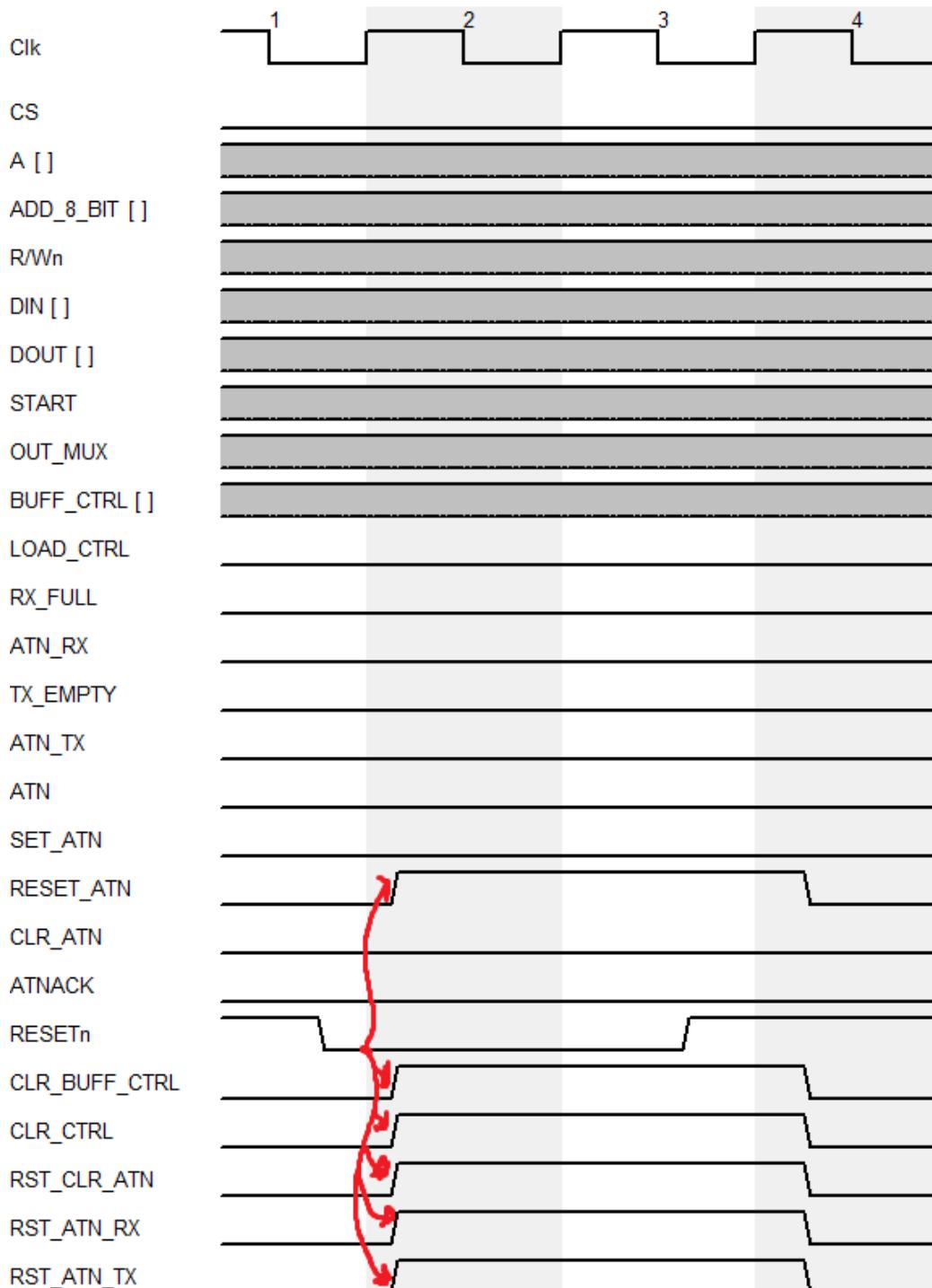
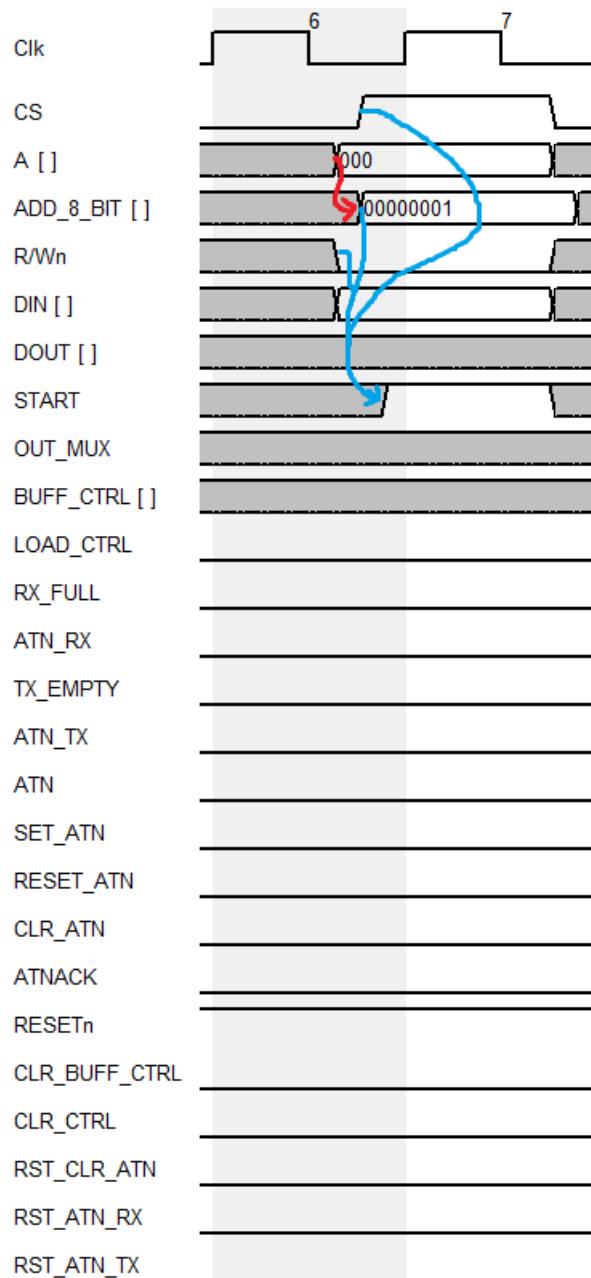


Figura 47: Reset della macchina

Figura 48: *Scrittura in TXDATA*

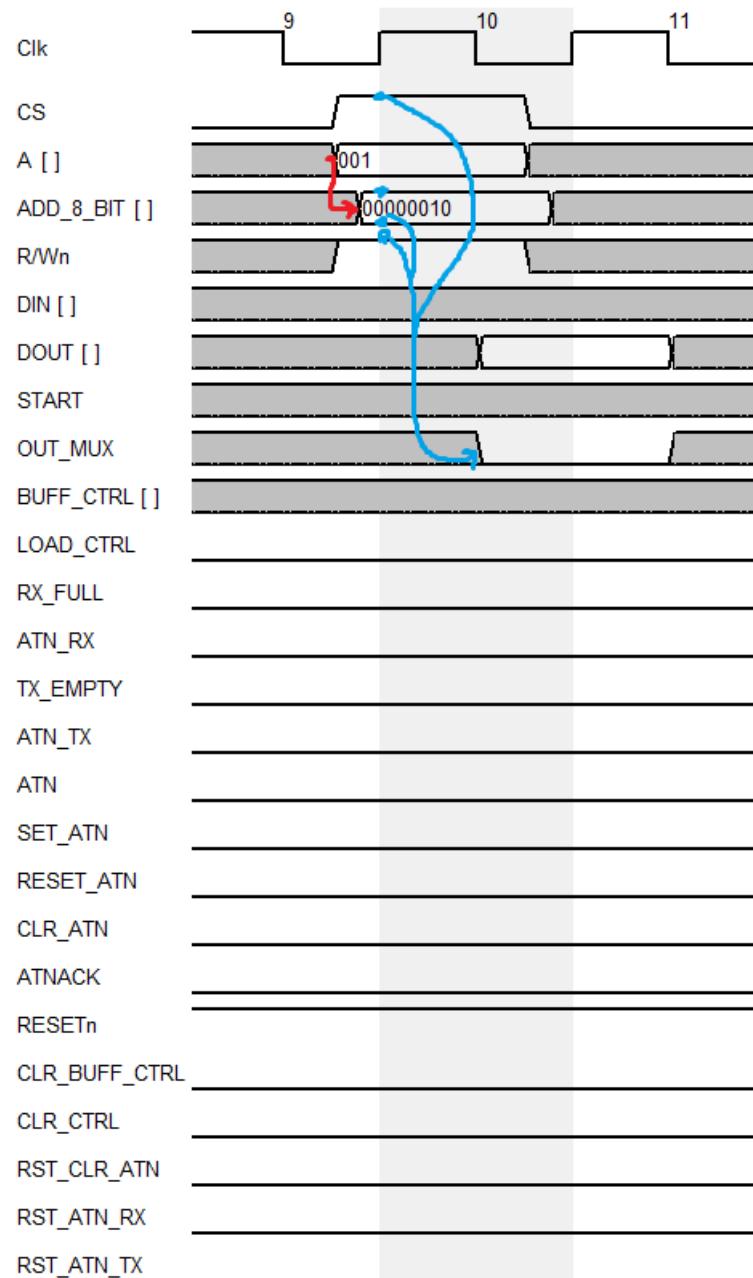
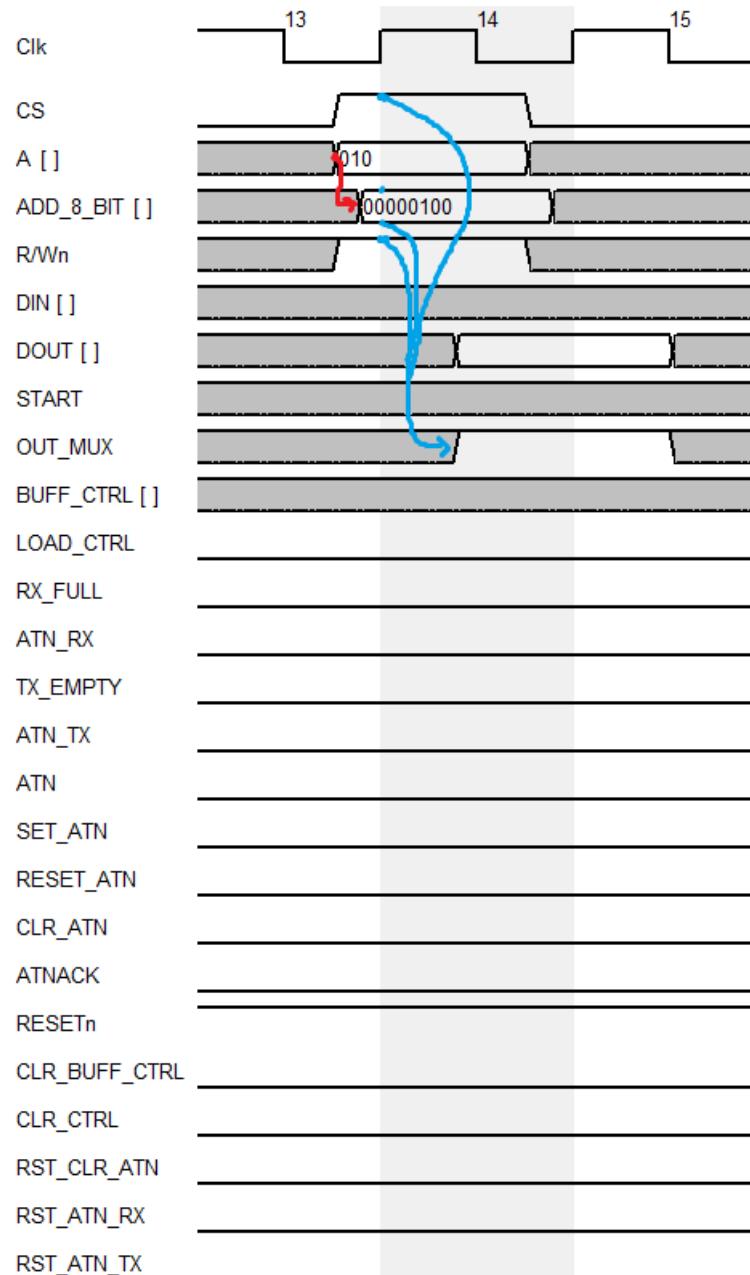
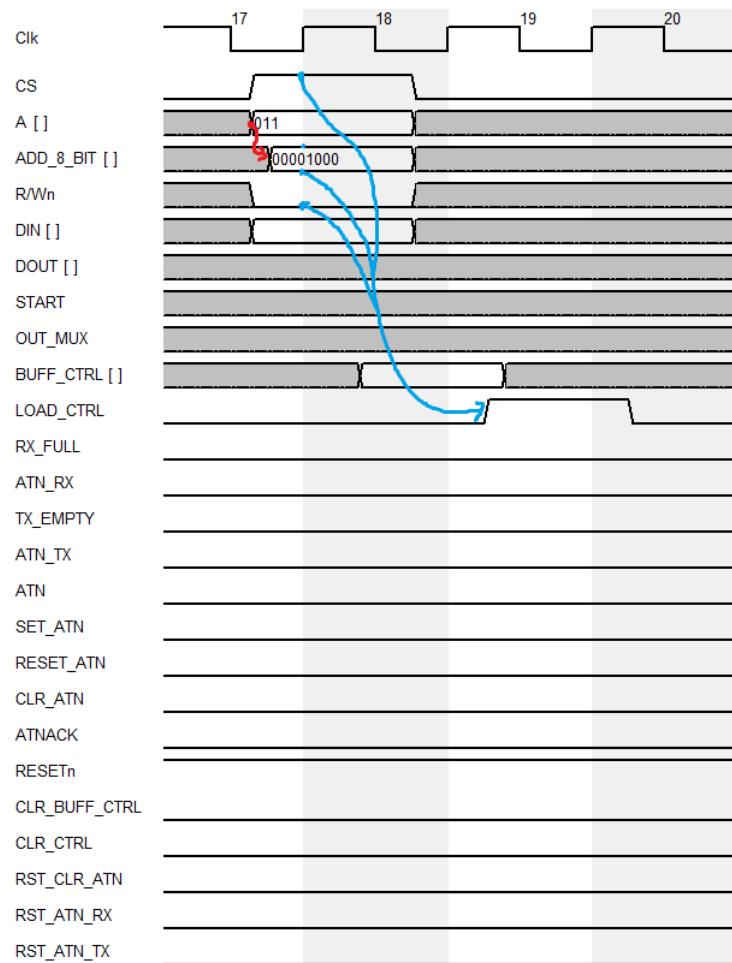
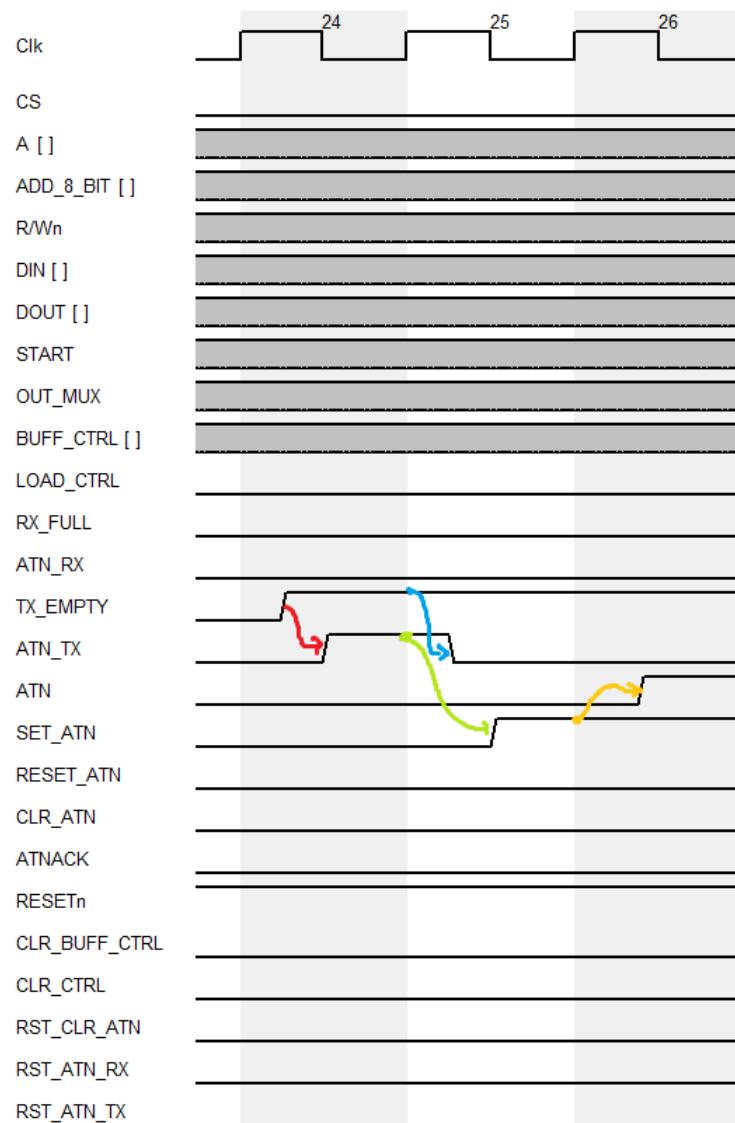


Figura 49: Lettura di RXDATA

Figura 50: *Lettura di STATUS REG*

Figura 51: *Scrittura in CONTROL REG*

Figura 52: *Invio di ATN*

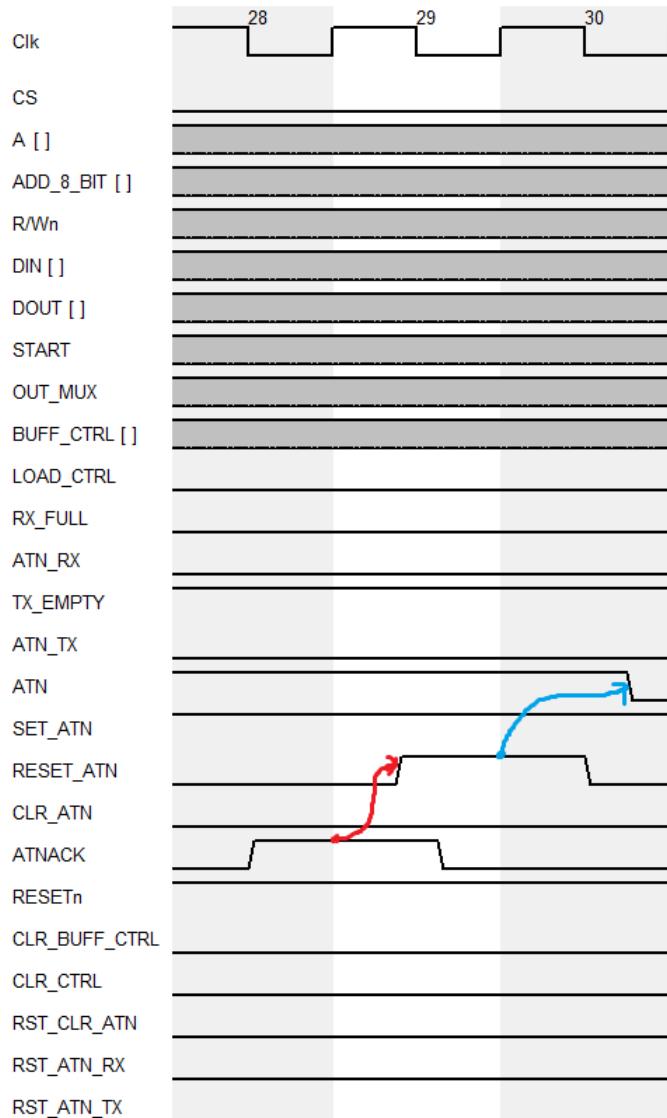


Figura 53: Ricezione di ATNACK

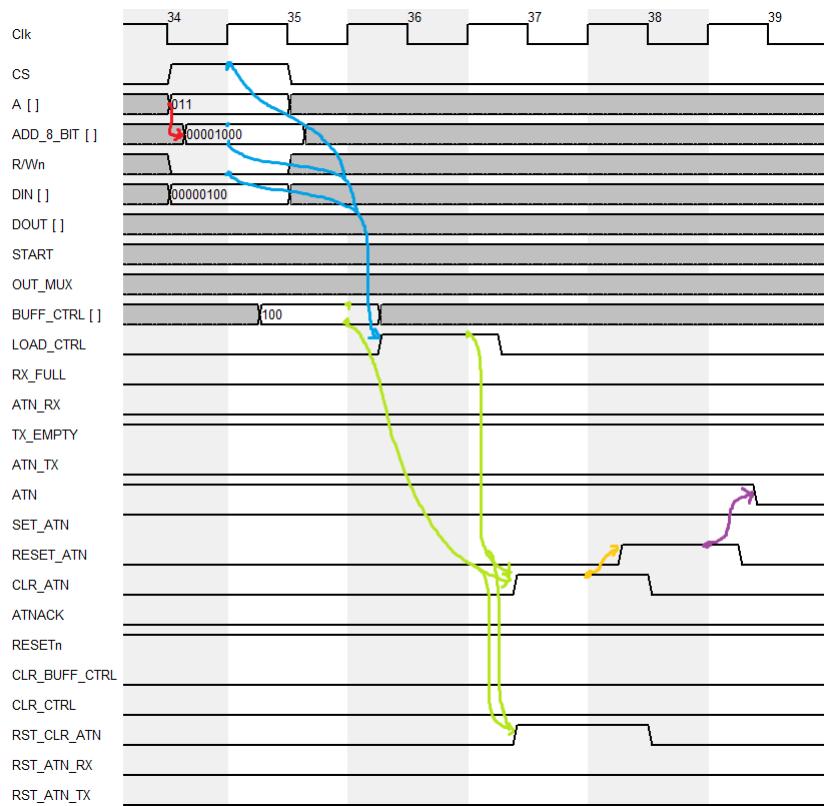


Figura 54: Ricezione di CLEARATN

4.4 Testbench

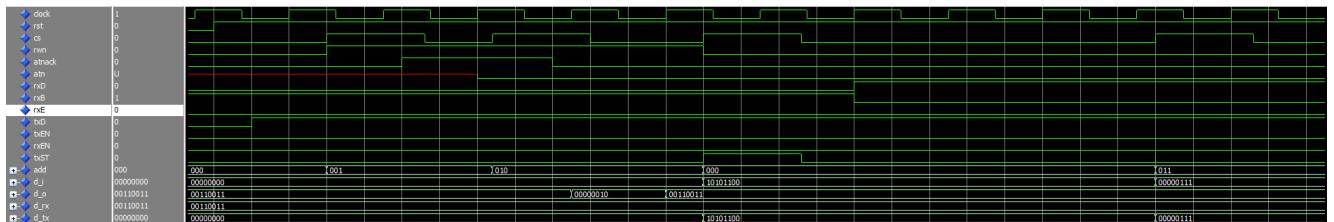


Figura 55: Simulazione su modelsim del comportamento della bus interface

5 Risultati delle simulazioni e sintesi

5.1 Modelsim

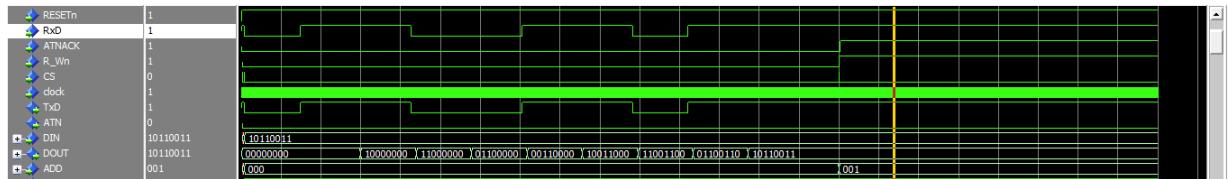


Figura 56: *Simulazione del timing della UART su modelsim*

5.2 Quartus

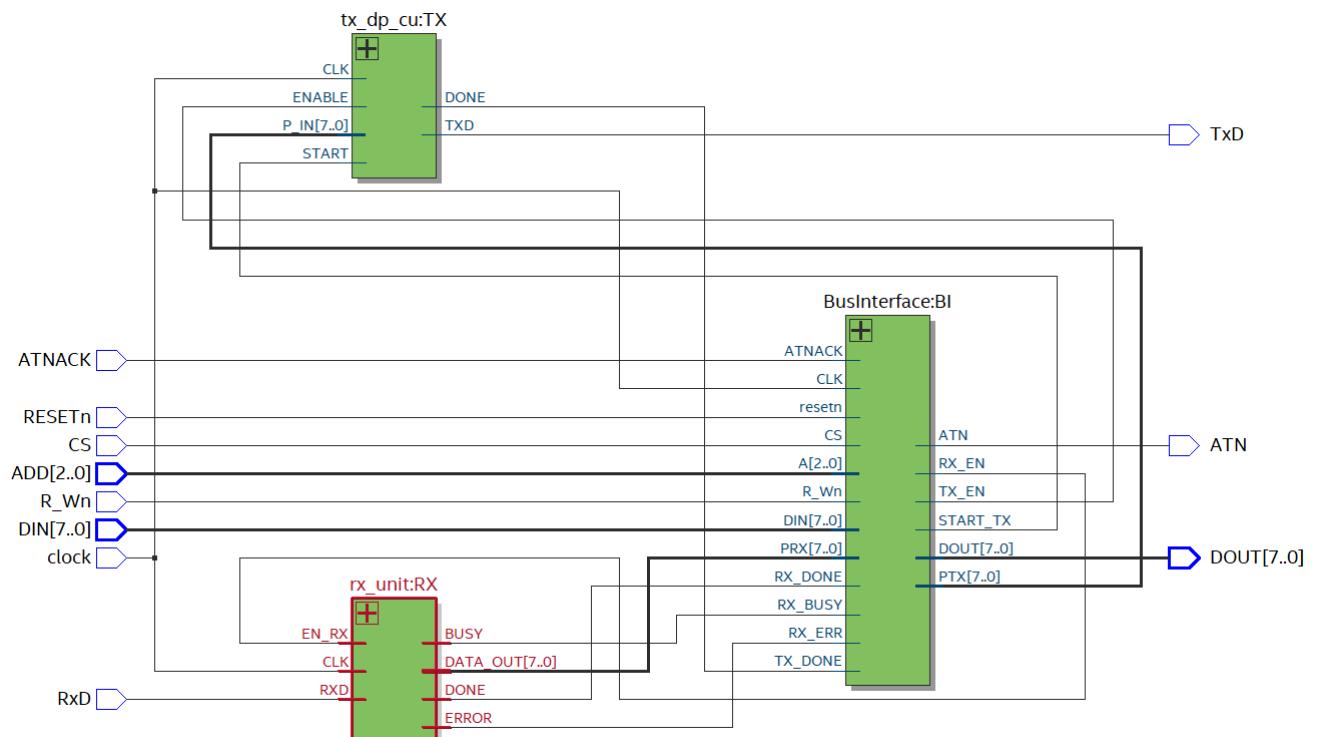


Figura 57: *Sintesi dei blocchi della UART*

5 RISULTATI DELLE SIMULAZIONI E SINTESI

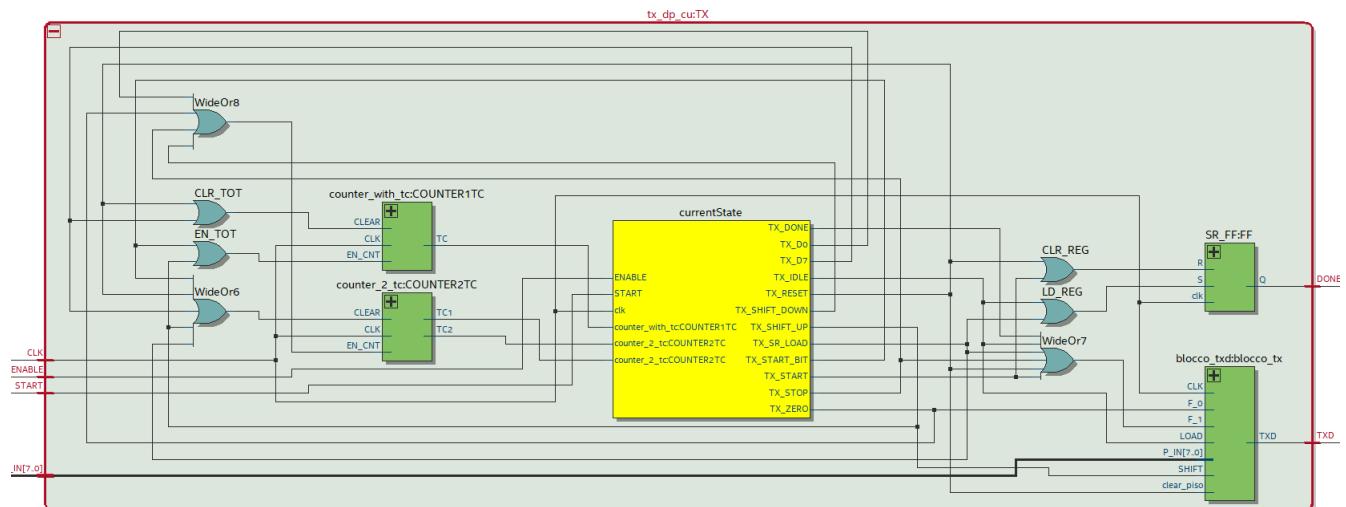


Figura 58: Sintesi del blocco di trasmissione

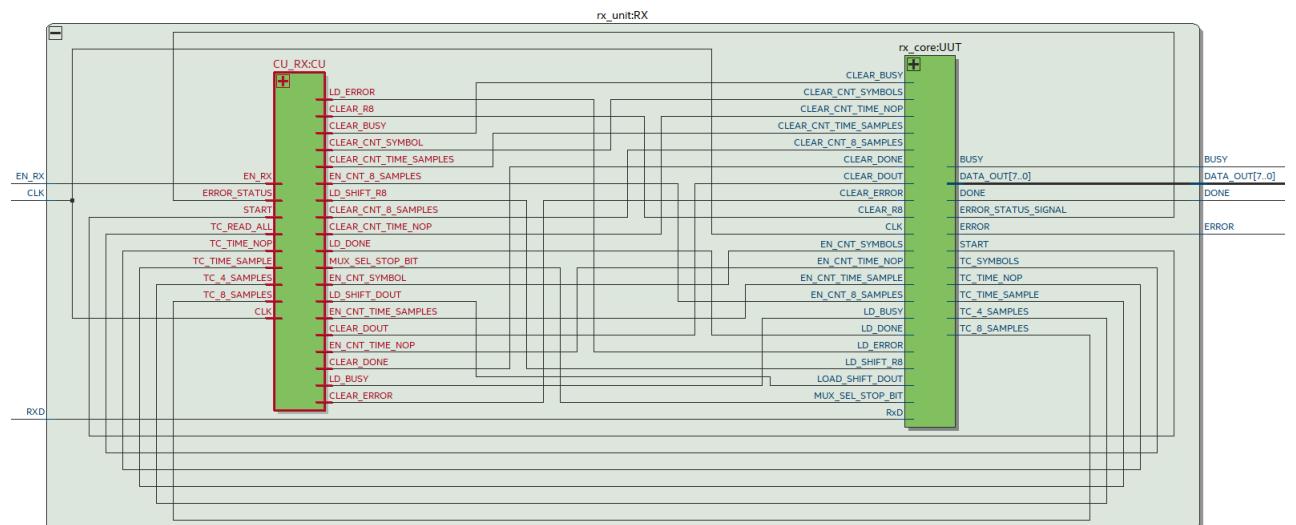


Figura 59: Sintesi del blocco di Ricezione

5 RISULTATI DELLE SIMULAZIONI E SINTESI

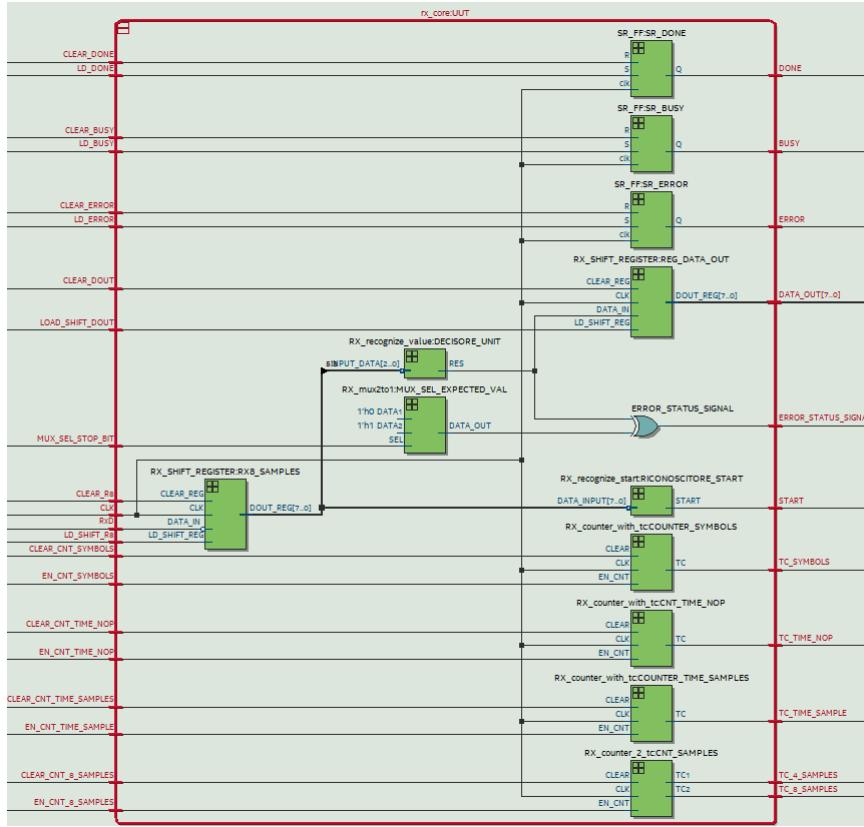


Figura 60: *Sintesi del datapath del blocco ricevitore*

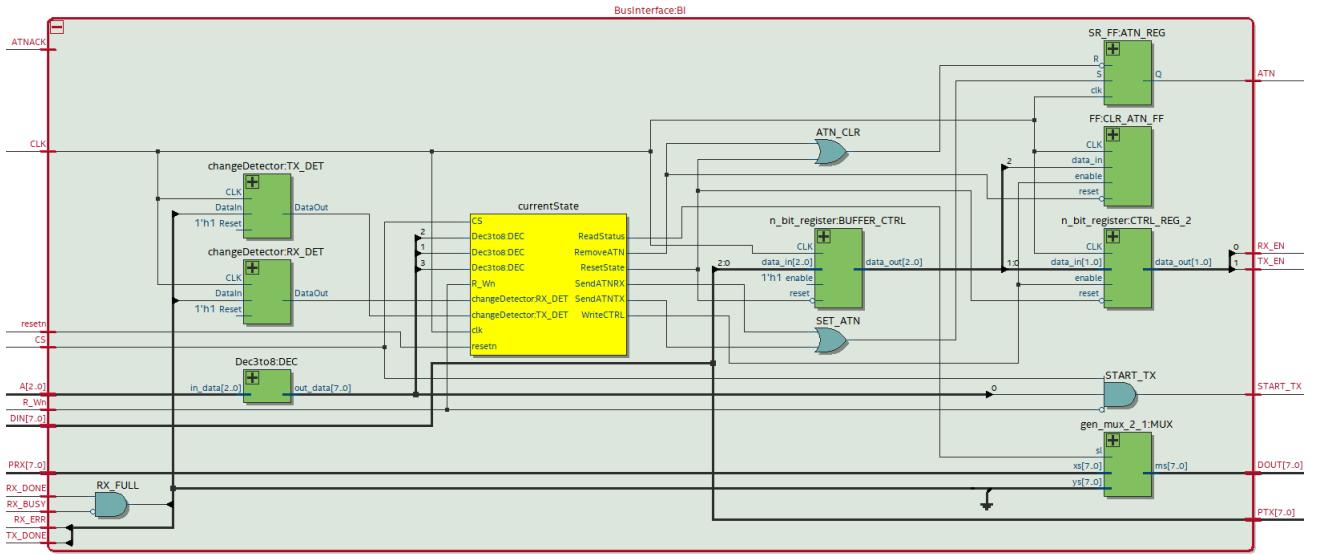


Figura 61: *Sintesi del blocco di bus interface*