Prova finale - Progetto di Reti Logiche

INTERFACCIA SERIALE ASINCRONA UART

SIMONE AMIGHINI
JUVRAJ SINGH BANWAIT

Introduzione

La specifica del progetto richiede di implementare un'interfaccia seriale asincrona UART (Universal Asynchronous Receiver/Transmitter), componente fondamentale nelle comunicazioni seriali ampiamente utilizzato in vari sistemi embedded e applicazioni di comunicazione.

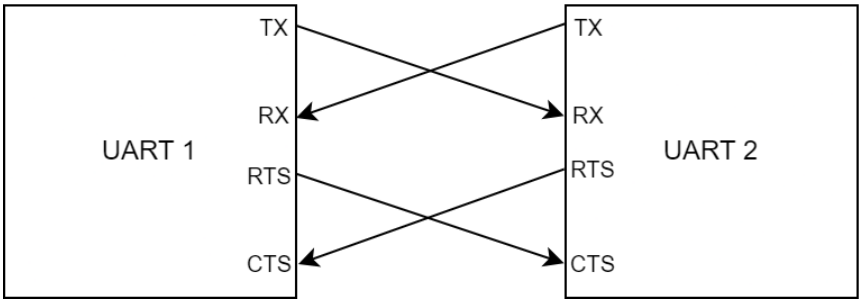
L'UART, opportunamente collegata ad un dispositivo utilizzatore (tipicamente un microprocessore) ha il compito di leggere dei dati forniti da quest'ultimo (8 bit, nella specifica richiesta), elaborarli secondo la modalità di trasmissione richiesta e infine, al ricevimento di un segnale di partenza, inviarli in serie ad un'altra interfaccia.

Dualmente, un'identica interfaccia campiona i bit ricevuti, li salva in modo da ricostruire l'intera parola trasmessa, effettua opportune verifiche (in base alla modalità di utilizzo impostata) e segnala al dispositivo utilizzatore l'avvenuta ricezione mediante un segnale. Si è deciso inoltre di aggiungere un segnale in uscita in modo da notificare l'utilizzatore in caso di errore durante la trasmissione.

Il principio di funzionamento dell'UART si basa dunque sul collegamento di due interfacce analoghe mediante quattro linee seriali e unidirezionali: due linee di trasmissione dei dati e due linee riservate al controllo di flusso. Più precisamente, ciascuna interfaccia è dotata di:

- un'uscita **TX** per la trasmissione dei dati verso un'altra interfaccia;
- un ingresso **RX** per la ricezione dei dati da un'altra interfaccia;
- un'uscita **RTS** (request to send*)* per la segnalazione del controllo di flusso a un'altra interfaccia;
- un ingresso **CTS** (clear to send) per la verifica dello stato del controllo di flusso segnalato da un'altra interfaccia.

Lo schema di collegamento è descritto dalla seguente figura:



La linea di trasmissione TX assume normalmente valore 1; quando è richiesto l'inizio della trasmissione, l'interfaccia legge il dato in ingresso e inizia il trasferimento dati: TX viene abbassato a 0 (start bit), quindi sono trasmessi in sequenza altri 9 bit in una delle tre modalità descritte di seguito:

- **8N1**: 8 bit di dati [D7:D0] + 1 bit di stop

D0	D1	D2	D3	D4	D5	D6	D7	STOP

- **7E1**: 7 bit di dati [D6:D0] + 1 bit di parità (pari) [PE] + 1 bit di stop

D0	D1	D2	D3	D4	D5	D6	PE	STOP

- **7O1**: 7 bit di dati [D6:D0] + 1 bit di parità (dispari) [PO] + 1 bit di stop

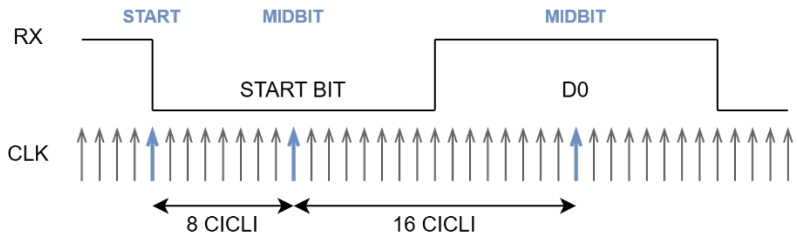
D0	D1	D2	D3	D4	D5	D6	PO	STOP

La modalità viene impostata in base ai segnali in ingresso **LEN** (lunghezza) e **PARITY** (parità):

LEN	PARITY	MODALITÀ
1	-	8N1
0	0	7E1
0	1	7O1

L'interfaccia trasmette a un baud rate compreso tra 9600 b/s e 921600 b/s; in particolare le tipiche frequenze di funzionamento (in b/s) sono: 9600, 19200, 38400, 57600, 115200, 230400, 460800 e 921600.

Adottando due clock differenti (aventi, in generale, fasi differenti), il ricevitore deve utilizzare un meccanismo di sincronizzazione per poter leggere correttamente i bit ricevuti: la soluzione adottata prevede di sovracampionare il segnale ricevuto utilizzando un clock di frequenza pari a 16 volte il baud rate e memorizzando solo il valore medio del bit (midbit) al fine di migliorare la reiezione di eventuali disturbi. Inoltre, sempre al fine di ridurre gli effetti dei disturbi sulla linea, è necessario che START si mantenga uguale a 0 per 8 cicli di clock per considerare tale evento come l'arrivo dello start bit.



L'interfaccia implementa un meccanismo di controllo di flusso hardware attraverso i segnali RTS e CTS descritti in precedenza.

1. Durante il normale funzionamento, il dispositivo ricevente mantiene il segnale RTS = 1, indicando che è pronto a ricevere dati. Questo segnale RTS è collegato all'ingresso CTS del dispositivo trasmittente.
2. Se il dispositivo ricevente non può più accettare dati (ad esempio perché il suo buffer è quasi pieno), pone STOP_RCV = 1 il quale causa RTS = 0.
3. Il dispositivo trasmittente, rilevando CTS = 0, termina la trasmissione corrente e blocca eventuali nuove trasmissioni fino a quando CTS non torna attivo.
4. Quando il dispositivo ricevente è di nuovo pronto ad accettare dati, pone STOP_RCV = 0, causando la riattivazione di RTS e permettendo così la ripresa delle trasmissioni.

Specifica

A livello di sistema, l'interfaccia UART è suddivisa in due macro-componenti:

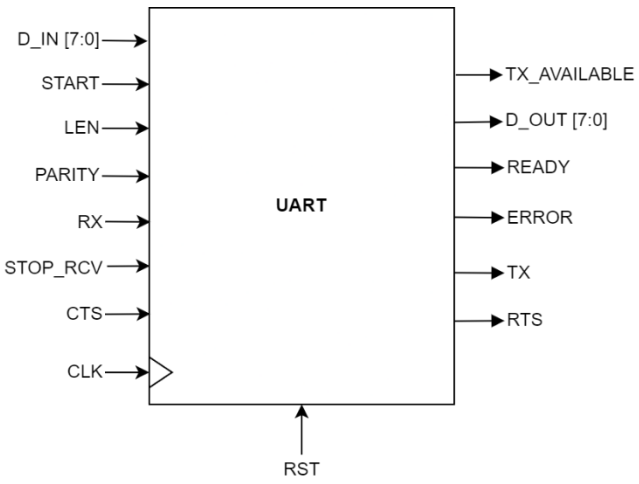
1. **Trasmettitore**
    - Opera ad una frequenza pari al baud rate desiderato, dunque pari a 1/16 della frequenza di clock.

- Riceve in ingresso D_IN (parola di 8 bit), calcola il bit di parità e lo sostituisce al bit più significativo di D_IN (se richiesto dalla configurazione di LEN e PARITY).
- Al ciclo di clock successivo al rilevamento di inizio trasmissione attraverso START, verifica se CTS è asserito e, in caso affermativo, carica la parola elaborata insieme seguita dallo start bit in un registro parallelo-serie a 9 bit.
- Invia in serie i bit caricati nel registro parallelo-serie dal bit di start fino al bit più significativo.
- Segnala all'utilizzatore quando è possibile inviare dei dati, cioè se CTS è asserito e non vi è alcuna trasmissione in corso, tramite TX_AVAILABLE. In caso di START non autorizzato, ovvero asserito quando TX_AVAILABLE = 0, ignora tale segnale.

2. **Ricevitore**
    - Sovracampiona RX a una frequenza pari a 16 volte il baud rate della trasmissione.
    - Tramite una macchina a stati finiti, rileva l'inizio della trasmissione e, in corrispondenza dei midbit, abilita il caricamento dei bit letti in un registro serie-parallelo.
    - Al termine della trasmissione fornisce la parola in uscita tramite D_OUT, segnala eventuali errori di frame o di parità tramite ERROR e asserisce un segnale READY per notificare l'utilizzatore dell'avvenuta ricezione.
    - Riceve STOP_RCV (che segnala la richiesta di interruzione di nuove trasmissioni) e, al ciclo di clock successivo, sovrascrive RTS con la sua negazione.

Interfaccia del sistema

Segnali di ingresso e uscita



Ove non specificato, i segnali sono da considerarsi da 1 bit.

INGRESSI

- **D_IN [7:0]** - byte che deve essere trasmesso; nel caso in cui LEN sia pari a 0 (cioè i dati trasmessi e ricevuti siano di soli 7 bit), vengono considerati solamente i 7 bit meno significativi [6:0];

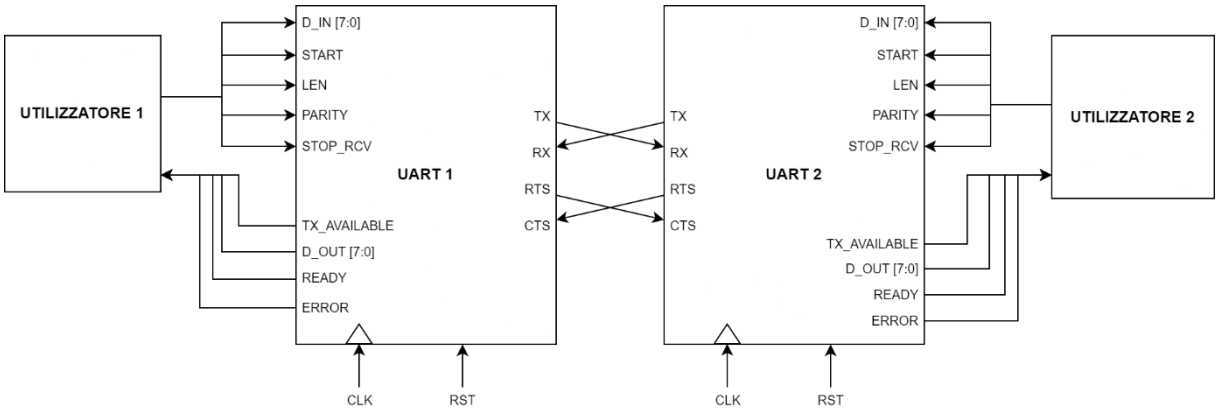
- **START** - segnale di inizio trasmissione (1 indica l'inizio);
- **LEN** - selezione della lunghezza della parola da trasmettere (0 per 7 bit, 1 per 8 bit);
- **PARITY** - selezione del tipo di parità (0 per pari, 1 per dispari);
- **RX** - ingresso seriale di ricezione di dati trasmessi da un'altra interfaccia;
- **STOP_RCV** - segnale inviato dal dispositivo utilizzatore che indica all'interfaccia di attivare il controllo di flusso e dunque di interrompere la trasmissione di dati verso la stessa (1 attiva tale richiesta);
- **CTS** - clear to send, segnale di ingresso per il controllo di flusso (1 permette il prosieguo delle trasmissioni, 0 causa la sospensione di quelle successive);
- **CLK** - clock di sistema, la cui frequenza è pari a 16 volte il baud rate a cui si vuole fare operare l'interfaccia;
- **RST** - segnale di reset sincrono (active high).

USCITE

- **TX_AVAILABLE** - segnale di notifica all'utilizzatore che l'interfaccia è pronta per trasmettere (1 indica la possibilità di trasmettere nuovi dati, 0 la l'impossibilità di farlo);
- **D_OUT [7:0]** - dati ricevuti da un'altra interfaccia;
- **READY** - segnale di dato ricevuto pronto che notifica all'utilizzatore l'aggiornamento di D_OUT (1 indica l'aggiornamento del dato);
- **ERROR** - segnale di rilevazione di un errore sul dato ricevuto (1 indica un errore);
- **TX** - uscita seriale di trasmissione dei dati verso un'altra interfaccia;
- **RTS** - request to send, segnale di uscita per il controllo di flusso (1 indica la disponibilità ad accettare dati, 0 l'indisponibilità): il segnale è calcolato dall'interfaccia come la negazione di STOP_RCV.

Modalità di utilizzo del sistema

Lo schema seguente mostra come collegare correttamente sia l'interfaccia UART al suo utilizzatore sia due interfacce tra di loro, in modo da costituire un collegamento utile alla trasmissione di dati.



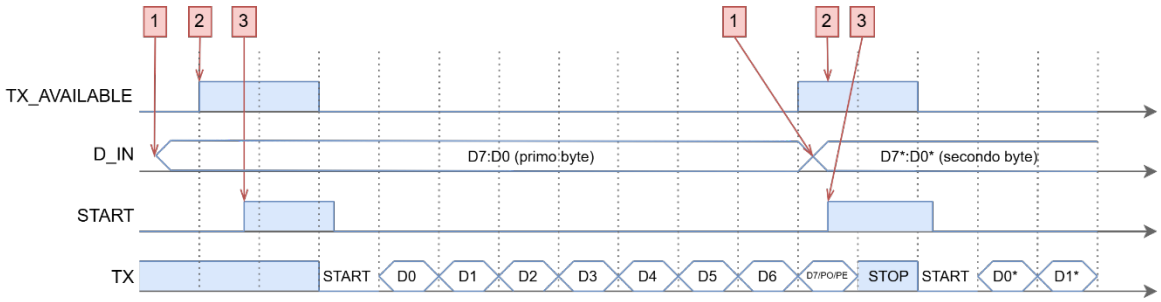
Nel garantire la correttezza della trasmissione dei dati e nelle indicazioni seguenti si assume che le interfacce collegate adottino la stessa configurazione della modalità di trasmissione (dunque valori analoghi per LEN e PARITY) nonché clock aventi la stessa frequenza.

Per trasmettere dati, l'utilizzatore deve seguire i seguenti passi:

1. indicare D_IN;

2. verificare che TX_AVAILABLE sia uguale 1;
3. porre START = 1 e mantenerlo alto fino a quando TX_AVAILABLE = 0: se il periodo in cui START rimane asserito è maggiore di 16 cicli di clock, ciò significa che la trasmissione ha avuto inizio.

I passaggi sono mostrati in questo diagramma temporale:



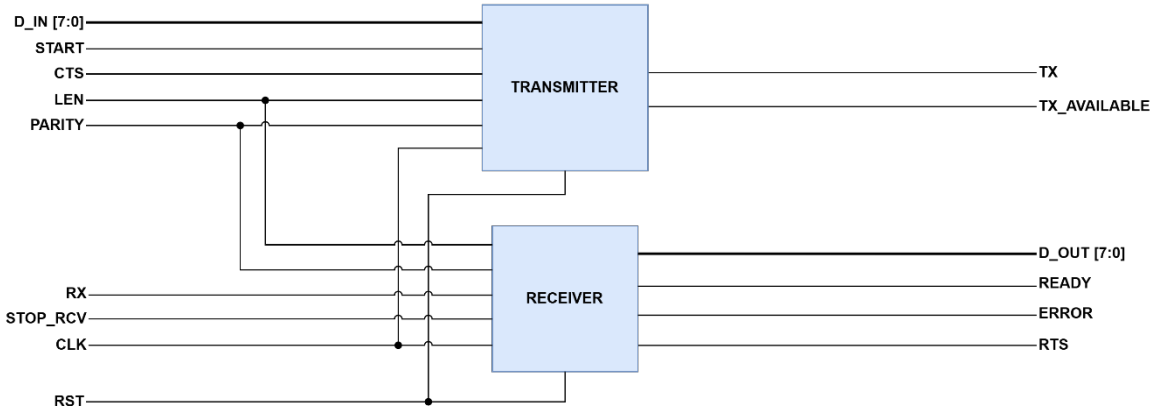
Per poter leggere i dati ricevuti, l'utilizzatore deve seguire i seguenti passi:

1. rilevare il passaggio di READY da 0 a 1;
2. verificare il valore di ERROR e prelevare D_OUT.

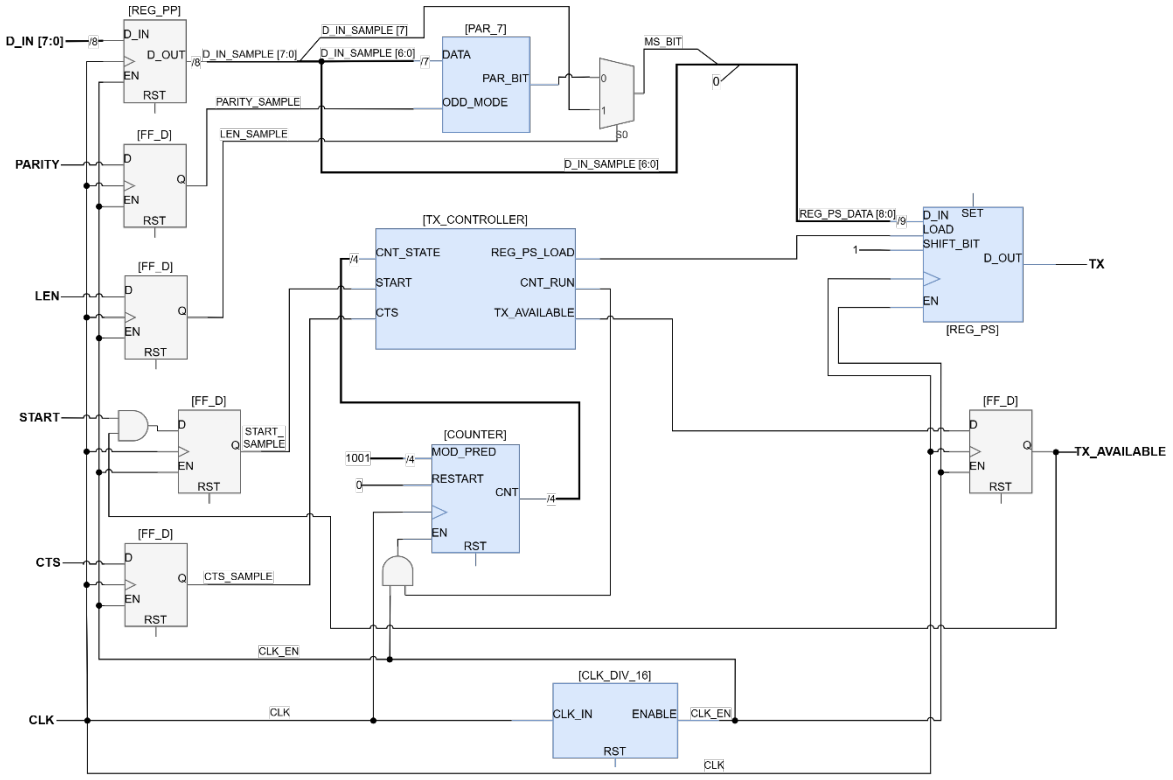
READY viene mantenuto a 1 per un solo periodo di CLK; D_OUT e ERROR, invece, rimangono disponibili fino alla conclusione della ricezione successiva dunque, nel caso pessimo, per 160 cicli di clock.

Architettura del sistema

L'architettura implementata prevede i due moduli indicati nella specifica: **TRANSMITTER** (trasmettitore) e **RECEIVER** (ricevitore). I segnali di ingresso e uscita per ciascun modulo sono quelli riportati nell'immagine seguente.



Trasmettitore (TRANSMITTER)



Tutti gli elementi del trasmettitore operano a una frequenza pari al baud rate desiderato, dunque pari a 1/16 della frequenza di clock: per ottenere tale comportamento è stato utilizzato un componente apposito, il divisore del clock (CLK_DIV_16), il cui clock enable in uscita (CLK_EN) è stato collegato in ingresso a tutti gli elementi sequenziali. Di seguito, dunque, per ciclo di elaborazione, si intende un periodo di elaborazione corrispondente al periodo di funzionamento del modulo (16 volte il periodo di clock).

Ad ogni ciclo, il registro D_IN viene aggiornato: sulla base del dato campionato il componente PAR_7 calcola il bit di parità (pari se PARITY_SAMPLE = 0, dispari viceversa) su D_IN [6:0]. A questo punto, in base a LEN, viene mantenuto o sostituito D_IN [7]: se LEN = 0 il bit più significativo è sostituito con PAR_BIT, se LEN = 1 rimane inalterato. Alla parola ottenuta viene aggiunto uno 0 (lo start bit) come bit meno significativo, ottenendo una parola di 9 bit, PS_REG_DATA [8:0], pronta entro il ciclo successivo.

Affinché la trasmissione abbia inizio è necessario che l'ingresso LOAD del registro parallelo-seriale sia uguale a 1 all'inizio del ciclo successivo: in questo caso, REG_PS_DATA [8:0] viene caricata nel registro; ad ogni ciclo successivo, il registro scorre verso destra di un bit e dunque espone in sequenza i bit da inviare sulla linea TX, inoltre, esso viene caricato a sinistra (dal bit più significativo) con degli 1 che causano, alla terminazione dello scorrimento, il ritorno di TX al valore di default (1). Se invece, LOAD = 0, REG_PS_DATA [8:0] non viene caricata nel registro e la trasmissione non ha luogo.

La gestione di LOAD (e dunque dell'inizio della trasmissione) è controllata dalla rete combinatoria TX_CONTROLLER: essa riceve in ingresso START_SAMPLE, CTS_SAMPLE e lo stato del trasmettitore, identificato tramite il valore di conteggio CNT proveniente da un contatore modulo 10 (COUNTER), e calcola i segnali REG_PS_LOAD, CNT_RUN e TX_AVAILABLE per il ciclo successivo. TX_CONTROLLER implementa dunque tutte le

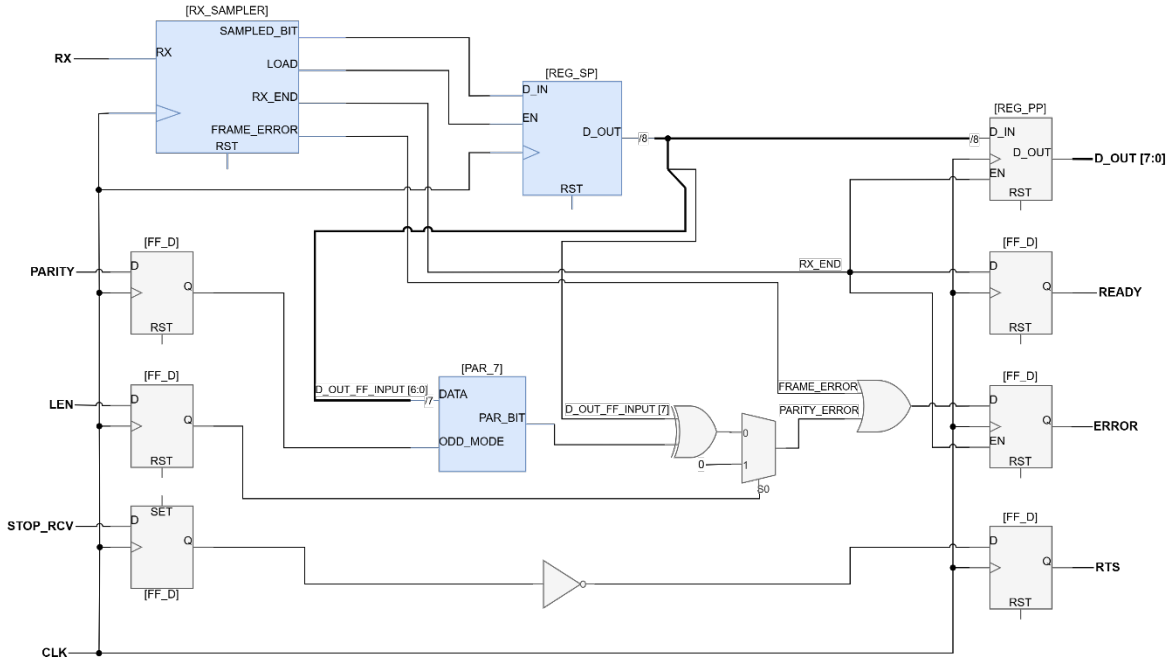
funzionalità di gestione della trasmissione e controllo di flusso (tramite REG_PS_LOAD), aggiornamento all'utilizzatore (tramite il segnale TX_AVAILABLE) e aggiornamento dello stato interno del trasmettitore (tramite il segnale CNT_RUN). I valori in uscita sono calcolati nel seguente modo:

CNT	REG_PS_LOAD	TX_AVAILABLE	CNT_RUN
0000	START_SAMPLE · CTS_SAMPLE	$\overline{\text{START\_SAMPLE}}$ · CTS_SAMPLE	START_SAMPLE · CTS_SAMPLE
0001	0	0	1
0010	0	0	1
0011	0	0	1
0100	0	0	1
0101	0	0	1
0110	0	0	1
0111	0	0	1
1000	0	0	1
1001	0	CTS_SAMPLE	1

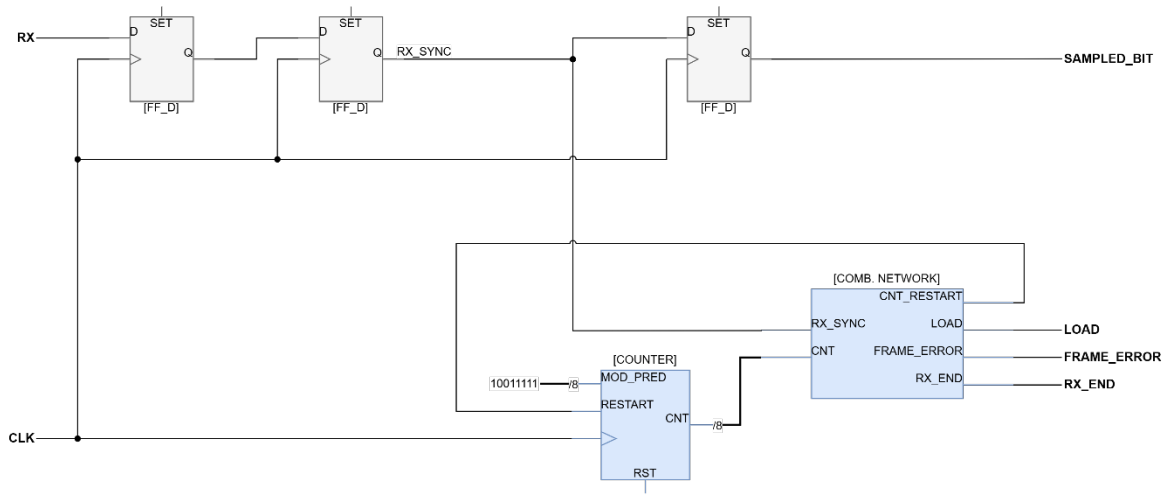
CNT identifica univocamente lo stato del trasmettitore:

CNT	SIGNIFICATO
0000	Trasmettitore in attesa di START. Eventuale invio in corso dello stop bit della trasmissione precedente che permette comunque di iniziare una nuova trasmissione.
0001	START ricevuto e parola appena caricata nel registro parallelo-serie. Invio in corso dello start bit.
0010	Invio in corso di D0.
0011	Invio in corso di D1.
0100	Invio in corso di D2.
0101	Invio in corso di D3.
0110	Invio in corso di D4.
0111	Invio in corso di D5.
1000	Invio in corso di D6.
1001	Invio in corso di D7 o del parity bit.

Ricevitore (RECEIVER)



Tramite RX_SAMPLER viene sovracampionato il segnale RX: data la complessità di tale componente si è preferito evidenziarlo come sottomodulo, riportando di seguito la sua architettura:



All'interno di RX_SAMPLER, RX viene sovracampionato a una frequenza pari alla frequenza di clock, ovvero 16 volte il baud rate di trasmissione: ciò è realizzato mediante due flip-flop posti in serie. Si è deciso di inserire un secondo flip-flop in modo da diminuire la probabilità che si verifichi un fenomeno di metastabilità: infatti, il segnale (potenzialmente metastabile) campionato nel primo flip-flop ha un ciclo di clock in più per stabilizzarsi a 0 o a 1 prima di essere memorizzato nel secondo flip-flop.

Il segnale memorizzato nel secondo flip-flop (RX_SYNC) viene poi valutato da una rete combinatoria (COMB. NETWORK) che, insieme a un contatore modulo 160 (COUNTER), costituisce una macchina a stati finiti che gestisce i segnali LOAD, FRAME_ERROR e RX_END; RX_SYNC viene inoltre inviato in ingresso a un terzo flip-flop il cui unico compito è quello di esporre in uscita a RX_SAMPLER un segnale SAMPLED_BIT in ritardo di un ciclo di clock

rispetto a RX_SYNC, in modo tale che i segnali elaborati dalla macchina a stati siano riferiti al dato che li ha prodotti.

- LOAD indica a un registro serie-parallelo esterno a RX_SAMPLER di caricare il dato SAMPLED_BIT.
- FRAME_ERROR segnala che lo stop bit rilevato non corrisponde a 1.
- RX_END indica la fine della trasmissione.

Il funzionamento del contatore modulo 160 e della rete combinatoria è riassunto nella tabella seguente:

RX_ SYNC	CNT	CNT_ RESTART	LOAD	FRAME_ ERROR	RX_ END	SIGNIFICATO
0	[0, 7]	0	0	0	0	Rilevamento start bit: in caso di 1 rilevato, il conteggio ricomincia.
1	[0, 7]	**1**	0	0	0	
-	8	0	0	0	0	Rilevato 0 otto volte consecutivamente. Inizio della ricezione di dati.
-	25	0	**1**	0	0	Caricamento midbit D0.
-	41	0	**1**	0	0	Caricamento midbit D1.
-	57	0	**1**	0	0	Caricamento midbit D2.
-	73	0	**1**	0	0	Caricamento midbit D3.
-	89	0	**1**	0	0	Caricamento midbit D4.
-	105	0	**1**	0	0	Caricamento midbit D5.
-	121	0	**1**	0	0	Caricamento midbit D6.
-	137	0	**1**	0	0	Caricamento midbit D7 o parity bit
0	153	0	0	**1**	**1**	Conclusione della ricezione con il rilevamento del midbit dello stop bit: se viene rilevato 0, è presente un errore di frame.
1	153	0	0	0	**1**	
-	[9, 24] ∪ [26, 40] ∪ [42, 56] ∪ [58, 72] ∪ [74, 88] ∪ [90, 104] ∪ [106, 120] ∪ [122, 136] ∪ [138, 152] ∪ [154, 159]	0	0	0	0	Periodi che intercorrono tra i midbit.

In uscita da RX_SAMPLER il registro serie-parallelo (REG_SP) carica i bit campionati dopo 3 cicli di clock rispetto alla loro ricezione sulla linea RX; l'uscita di REG_SP è collegata in ingresso al registro parallelo-parallelo D_OUT.

Al termine della ricezione, corrispondente con il valore CNT = 153 nella precedente tabella, RX_END = 1: essendo tale segnale collegato al clock enable del registro D_OUT e del flip-flop ERROR, esso causa l'aggiornamento (al fronte di clock successivo) di tali valori in uscita con i dati forniti rispettivamente da REG_SP e dalla rete di calcolo dell'errore. Nello stesso ciclo in cui D_OUT e ERROR sono forniti in uscita anche il flip-flop READY viene aggiornato, ponendo il suo valore a quello di RX_END, cioè 1. A differenza di D_OUT e ERROR, i quali sono aggiornati solo in corrispondenza della fine della trasmissione, READY rimane uguale a 1 solo per un periodo di clock.

La rete di calcolo dell'errore che fornisce il valore di ERROR prevede il calcolo della parità (in base alla configurazione di PARITY) sul segnale D_OUT_FF_INPUT [6:0], corrispondente ai 7 bit meno significativi dell'uscita del registro parallelo-serie. Se LEN = 0 il bit di parità calcolato viene confrontato con quello effettivamente ricevuto (D_OUT_FF_INPUT [7]): in caso di uguaglianza, PARITY_ERROR = 0, viceversa PARITY_ERROR = 1; se invece LEN = 1 non è necessario verificare la parità, perciò PARITY_ERROR = 0. Infine, ERROR è dato dalla disgiunzione di PARITY_ERROR e di FRAME_ERROR (fornito da RX_SAMPLER).

Il ricevitore si occupa anche di aggiornare RTS, ponendolo uguale alla negazione di STOP_RCV.

Componenti di base

Flip-flop D (FF_D)

INGRESSI

- **CLK** - clock;
- **EN** - clock enable;
- **SET** - segnale di set sincrono;
- **RST** - segnale di reset sincrono;
- **D** - bit da memorizzare.

USCITE

- **Q** - stato (bit memorizzato).

Al fronte di salita di CLK, se EN = 1, lo stato Q assume il valore di D. Se RST = 1, allora Q diventa 0 mentre, se SET = 1, allora Q diventa 1. Ad ogni fronte di clock, SET prevale su RST il quale a sua volta prevale su D.

Multiplexer a 2 ingressi (MUX2)

INGRESSI

- **IN_0** - segnale abilitato se S = 0;
- **IN_1** - segnale abilitato se S = 1;
- **S** - segnale di selezione dell'ingresso.

USCITE

- $\mathbf{OUT} = \overline{\mathrm{S}} \cdot \mathrm{IN\_0} + \mathrm{S} \cdot \mathrm{IN\_1}$

Registro parallelo-parallelo (REG_PP)

PARAMETRI

- $\mathbf{REG\_NUMBER} \in (\mathbb{N} - \{0\})$ - numero di bit della parola da memorizzare.

INGRESSI

- **CLK** - clock;
- **EN** - clock enable;
- **SET** - segnale di set sincrono;
- **RST** - segnale di reset sincrono;

USCITE

- **D_OUT [(REG_NUMBER-1):0]** - stato (parola memorizzata).

- **D_IN [(REG_NUMBER-1):0]** - parola da memorizzare.

Registro parallelo-serie (REG_PS)

PARAMETRI

- **REG_NUMBER** $\in (\mathbb{N} - \{0\})$ - numero di bit della parola da memorizzare.

INGRESSI

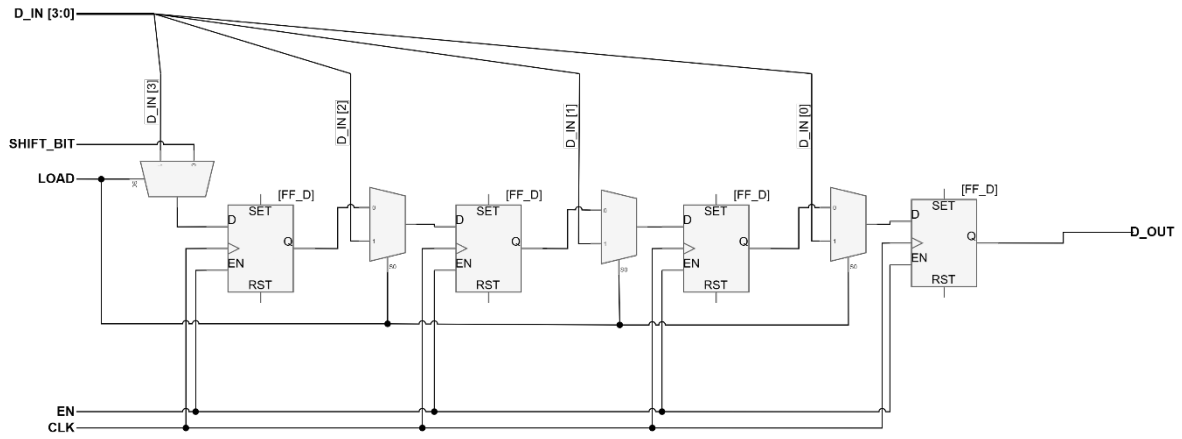
- **CLK** - clock;
- **EN** - clock enable;
- **SET** - segnale di set sincrono;
- **RST** - segnale di reset sincrono;
- **D_IN [(REG_NUMBER-1):0]** - parola da memorizzare;
- **LOAD** - segnale di caricamento di D_IN;
- **SHIFT_BIT** - bit inserito come cifra più significativa durante la fase in cui il registro sta operando lo scorrimento.

USCITE

- **D_OUT** - bit meno significativo della parola memorizzata nel registro.

Il registro opera in due modalità: se LOAD = 1 viene caricato D_IN invece, se LOAD = 0, ad ogni ciclo di clock (in cui EN = 1), viene effettuato uno scorrimento a destra con inserimento a sinistra di SHIFT_BIT.

È riportata l'architettura per REG_NUMBER = 4.



Registro serie-parallelo (REG_SP)

PARAMETRI

- **REG_NUMBER** $\in (\mathbb{N} - \{0\})$ - numero di bit della parola da memorizzare.

INGRESSI

- **CLK** - clock;
- **EN** - clock enable;

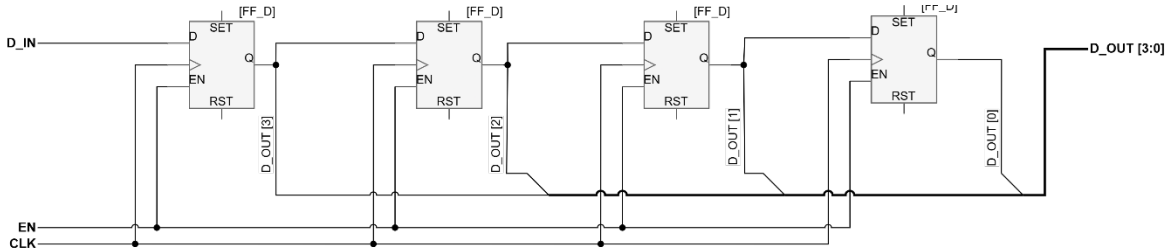
USCITE

- **D_OUT [(REG_NUMBER-1):0]** - parola memorizzata nel registro.

- **SET** - segnale di set sincrono;
- **RST** - segnale di reset sincrono;
- **D_IN** - bit da caricare.

Se EN = 1, il registro opera uno scorrimento a destra della parola inserendo a sinistra D_IN come bit più significativo.

È riportata l'architettura per REG_NUMBER = 4.



Contatore (COUNTER)

PARAMETRI

- **REQUIRED_BITS** $\in (\mathbb{N} - \{0\})$ - numero del massimo valore di conteggio.

INGRESSI

- **CLK** - clock;
- **EN** - clock enable;
- **RST** - segnale di reset sincrono;
- **RESTART** - richiesta di ricominciare il conteggio, cioè di porre CNT = 0;
- **MOD_PRED [(REQUIRED_BITS-1):0]** - predecessore del modulo che si vuole impostare per il contatore.

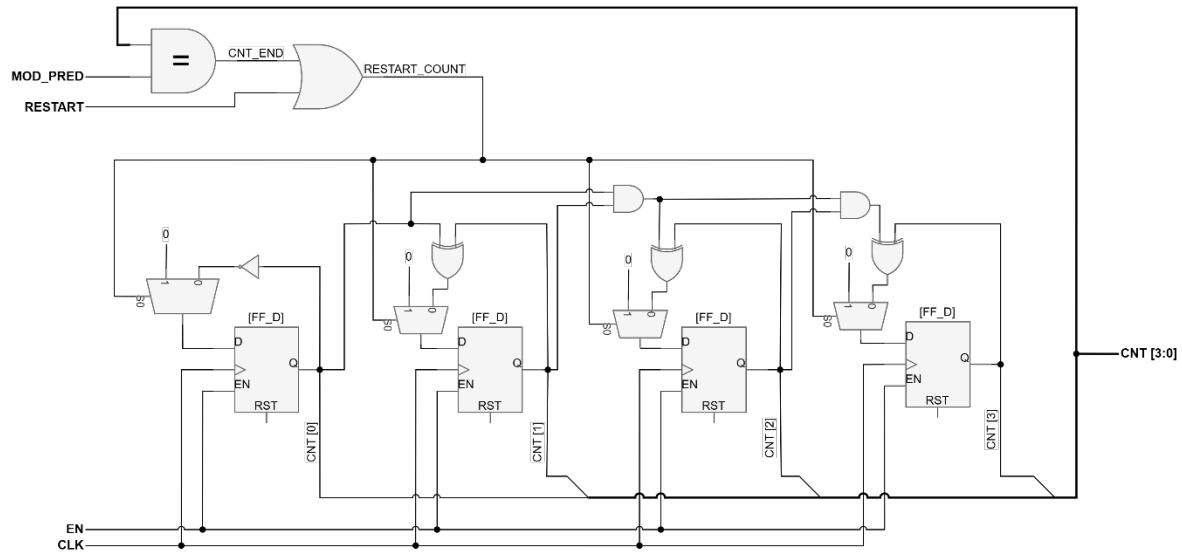
USCITE

- **CNT [(REQUIRED_BITS-1):0]** - valore di conteggio.

Ad ogni ciclo di clock (in cui EN = 1) il valore di CNT aumenta di 1; quando CNT è uguale a MOD_PRED oppure RESTART = 1, CNT è reimpostato a 0. Il contatore realizza dunque una sequenza di conteggio da 0 a MOD_PRED.

L'architettura si basa su quella di un contatore veloce realizzato con flip-flop T: essa è stata modificata per utilizzare dei flip-flop D in modo da consentire la funzionalità di RESTART senza utilizzare il reset.

È riportata l'architettura per REQUIRED_BITS = 4.



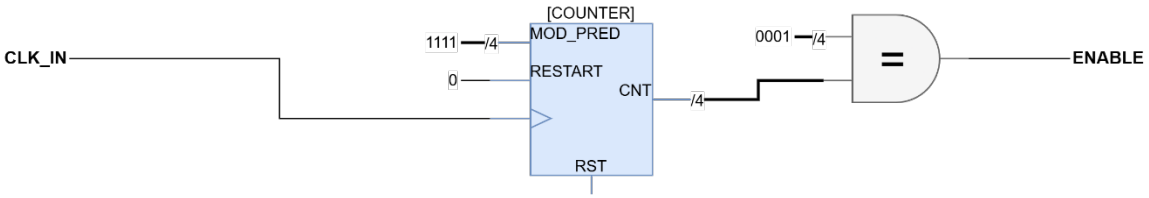
Divisore del clock (CLK_DIV_16)

INGRESSI

- **CLK_IN** - clock veloce;
- **RST** - segnale di reset sincrono.

USCITE

- **ENABLE** - segnale con frequenza pari a 1/16 della frequenza di CLK_IN e duty cycle pari al 6,25%.



Utilizza un modulo COUNTER per implementare un contatore modulo 16 (MOD_PRED = 15). ENABLE = 1 se e solo se CNT = 1: in questo modo, ENABLE è un segnale con una frequenza esattamente pari a 1/16 della frequenza di CLK_IN e duty cycle pari a 1/16 = 6,25%. Per i componenti che usano ENABLE come segnale di clock enable, il duty cycle al 6,25% fa sì che ENABLE = 1 solamente una volta ogni 16 fronti di salita del clock e quindi che tali componenti funzionino in maniera sincrona con il clock ma a una frequenza pari a 1/16 dello stesso.

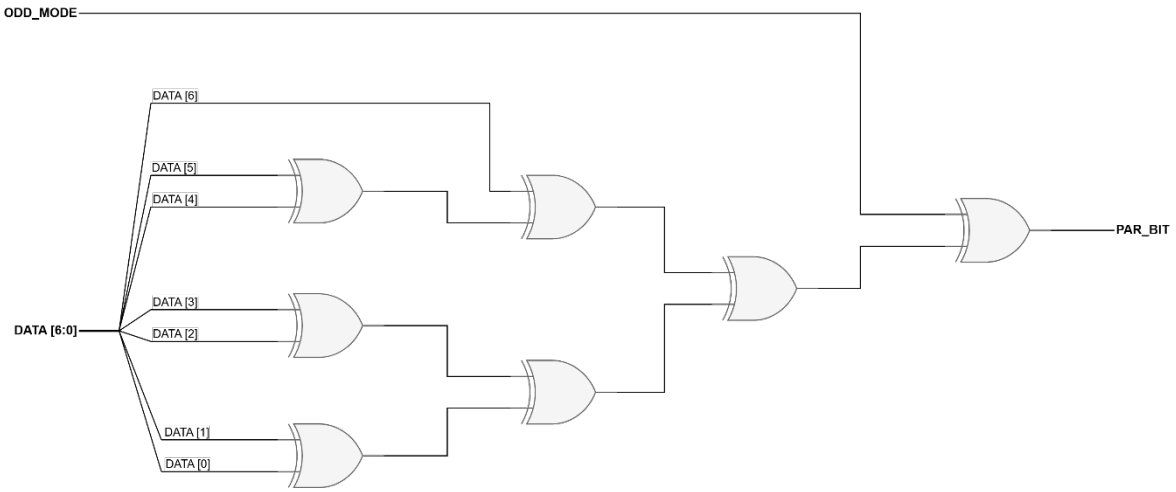
Calcolatore di parità su 7 bit (PAR_7)

INGRESSI

- **DATA [6:0]** - parola di 7 bit su cui calcolare la parità;
- **ODD_MODE** - selettore del tipo di parità (0 parità pari, 1 parità dispari).

USCITE

- **PAR_BIT** - bit di parità calcolato.



Verifica

Il test-bench progettato per verificare il corretto funzionamento dell'interfaccia realizzata prevede l'istanziazione di due dispositivi, un'interfaccia UART che opera solo in trasmissione (UART A) e un'interfaccia UART che opera solo in ricezione (UART B), nonché l'uso di una procedura automatica (SEND_BYTE) in grado di emulare le funzionalità di trasmissione di un'interfaccia UART con l'inserimento di errori di frame e di parità.

UART A e SEND_BYTE operano a una frequenza di clock pari alla frequenza massima ammissibile (16 volte il massimo baud rate) ovvero 14,7456 MHz, corrispondenti a un periodo di circa 67,817 ns. UART B, invece, è alimentata con un clock sfasato di metà periodo rispetto al clock di UART A e di frequenza inferiore del 2,5% rispetto allo stesso.

I segnali LEN e PARITY sono configurati in maniera identica per UART A, UART B e SEND_BYTE. All'interfaccia UART A vengono forniti i byte da trasmettere (D_IN_A) e il segnale di inizio trasmissione (START_A) mentre si agisce sull'interfaccia UART B attivando o disattivando il controllo di flusso tramite STOP_RCV_B; collegando UART A e UART B vengono dunque effettuate varie trasmissioni facendo variare LEN, PARITY e STOP_RCV_B e verificando, lato UART B, che i segnali D_OUT_B e ERROR_B siano coerenti con quelli attesi.

Per verificare se l'interfaccia ricevente (UART B) è in grado di rilevare errori di frame o di parità, si agisce su un flag ERROR_TEST: se asserito la linea RX di UART B viene scollegata dalla linea TX di UART A e collegata all'output di SEND_BYTE. Gli input forniti a SEND_BYTE ricalcano quelli forniti a UART A (DATA, LEN e PARITY) a cui si aggiungono due flag per selezionare il tipo di errore da generare (BAD_STOP_BIT e BAD_PARITY).

Sono stati implementati i seguenti casi di test:

1. Invio dato con configurazione 8N1
   CONFIGURAZIONE
   - **LEN** = 1
   - **PARITY** = 0
   - **D_IN_A** = 10101111
   - **STOP_RCV_B** = 0
   - **ERROR_TEST** = FALSE

RISPOSTA ATTESA

- **D_OUT_B** = 10101111
- **ERROR_B** = 0

2. Invio dato con configurazione 7E1
   CONFIGURAZIONE
   - **LEN** = 0
   - **PARITY** = 0
   - **D_IN_A** = 10101011
   - **STOP_RCV_B** = 0
   - **ERROR_TEST** = FALSE

   RISPOSTA ATTESA
   - **D_OUT_B** = 00101011
   - **ERROR_B** = 0

3. Invio dato con configurazione 7O1
   CONFIGURAZIONE
   - **LEN** = 0
   - **PARITY** = 1
   - **D_IN_A** = 00101011
   - **STOP_RCV_B** = 0
   - **ERROR_TEST** = FALSE

   RISPOSTA ATTESA
   - **D_OUT_B** = 10101011
   - **ERROR_B** = 0

4. Richiesta di interruzione della trasmissione mediante controllo di flusso
   CONFIGURAZIONE
   - **LEN** = 0
   - **PARITY** = 1
   - **D_IN_A** = 00101011
   - **STOP_RCV_B** = 1
   - **ERROR_TEST** = FALSE

   RISPOSTA ATTESA
   - Nessun dato ricevuto.

5. Invio dato con configurazione 8N1 mediante SEND_BYTE impostata per generare un errore di frame
   CONFIGURAZIONE
   - **LEN** = 1
   - **PARITY** = 0
   - **DATA** = 00000000
   - **BAD_STOP_BIT** = TRUE
   - **BAD_PARITY** = FALSE
   - **STOP_RCV_B** = 0
   - **ERROR_TEST** = TRUE

RISPOSTA ATTESA

- **D_OUT_B** = 00000000
- **ERROR_B** = 1

6. Invio dato con configurazione 7N1 mediante SEND_BYTE impostata per generare un errore di parità

CONFIGURAZIONE

- **LEN** = 0
- **PARITY** = 1
- **DATA** = 00000000
- **BAD_STOP_BIT** = FALSE
- **BAD_PARITY** = TRUE
- **STOP_RCV_B** = 0
- **ERROR_TEST** = TRUE

RISPOSTA ATTESA

- **D_OUT_B** = 00000000
- **ERROR_B** = 1

La simulazione è svolta sul modello Post Place & Route per verificare il comportamento del sistema nelle condizioni più vicine possibili a quelle reali.