Progetto di reti Logiche

Interfaccia seriale asincrona UART

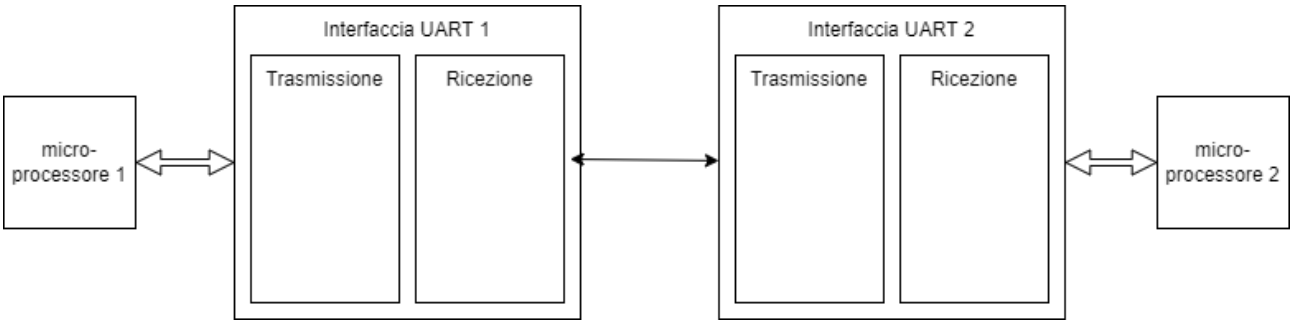
Francesco Micucci e Davide Frati

INTRODUZIONE

Il progetto riguarda la realizzazione di una interfaccia seriale asincrona UART. L'obiettivo è quello di progettare un dispositivo in grado di:

- Ricevere una parola di 8 bit in parallelo dall'esterno (ad esempio da un microprocessore) e trasmetterla in serie (un bit alla volta) ad un'altra interfaccia UART.
- Ricevere una parola di 8 bit in serie da un'altra interfaccia UART e trasmetterla in parallelo verso l'esterno (ad esempio un microprocessore).

Ogni interfaccia può trasmettere e ricevere in maniera contemporanea e indipendente.



La comunicazione in una certa direzione tra due interfacce UART deve essere possibile solo nelle situazioni in cui il ricevitore è disponibile.
Il ricevitore fornisce quindi un segnale RTS (Request To Send) in uscita, che indica la sua predisposizione a ricevere nuovi dati. Lato trasmettitore, questo segnale deve essere gestito tramite un ingresso CTS (Clear To Send), il quale indica che la trasmissione seriale può avvenire.

La comunicazione avviene nel modo seguente:

- L'interfaccia in trasmissione si prepara a trasmettere inviando uno "start bit", successivamente trasmette la parola di 8 bit e infine invia uno "stop bit".
- L'interfaccia in ricezione legge i dati in serie, e una volta ricevuto lo "stop bit" mostra in parallelo la parola ricevuta. In questa fase è anche necessario controllare che l'ultimo bit ricevuto sia effettivamente uno "stop bit".

Il dispositivo esterno ha anche la possibilità di scegliere la modalità di comunicazione, la quale può essere:

- 8N1 (8 bit, no bit di parità, 1 stop bit) ⇒ in questo caso l'intera parola contiene dati e la codifica è in binario naturale.
- 7E1 (7 bit, 1 bit di parità, 1 stop bit) ⇒ in questo caso, i primi 7 bit della parola rappresentano i dati, mentre l'ottavo è un campo calcolato (parità pari, vale 1 se il numero di 1 nella parola è dispari). L'interfaccia in ricezione deve verificare la correttezza della parola ricevuta, inviando un segnale nel caso in cui dovesse rilevare un errore tramite il bit di parità.

Due interfacce UART devono essere in grado di comunicare tra loro anche nelle seguenti situazioni:

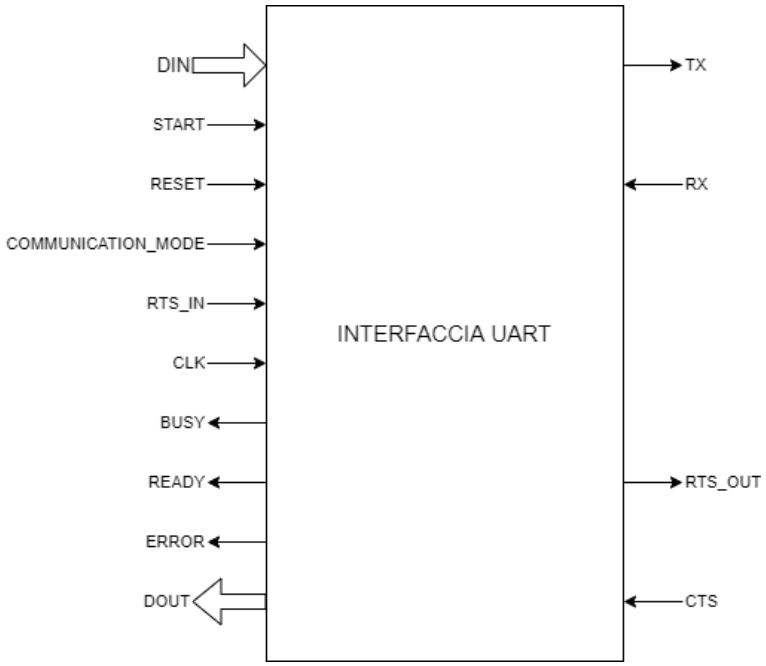
- I loro segnali di clock non sono in fase.
- Per motivi fisici di varia natura, i loro clock non hanno esattamente la stessa frequenza.

È dunque necessaria una sincronizzazione tra i due dispositivi prima dell'inizio effettivo della trasmissione della parola.
Individuando l'inizio trasmissione, il ricevente deve selezionare l'istante di tempo più consono alla rilevazione dei dati trasmessi (in altre parole, il dato deve essere campionato in un istante di tempo quanto più centrale possibile rispetto all'intervallo di tempo in cui tale informazione rimane disponibile).

La frequenza massima di funzionamento dei dispositivi è di 921600 bit/s: l'assunzione fatta è che qualunque segnale proveniente dall'esterno abbia la durata di almeno un ciclo di clock.
Si suppone inoltre che la modalità di trasmissione (8N1 o 7E1) nelle due direzioni sia preventivamente accordata tra i due dispositivi che vogliono scambiarsi dati.

SPECIFICA

Interfaccia del sistema



Il dispositivo UART è caratterizzato dai seguenti segnali:
La figura mostra i segnali (ingresso e uscita) del modulo "interfaccia UART".

Sul lato sinistro troviamo i segnali che la UART scambia con il dispositivo esterno. In particolare:

- DIN ⇒ di tipo std_logic_vector, è una parola di 8 bit codificata in binario naturale che viene presa in input dal dispositivo e che deve essere serializzata;
- START ⇒ di tipo std_logic, è un segnale della durata minima di un ciclo di clock che indica che i dati presenti in DIN sono pronti per essere trasmessi e quindi che è possibile proseguire con la serializzazione;
- RESET ⇒ di tipo std_logic, segnale che rimane a 1 per la durata di un ciclo di clock e che viene utilizzato per la configurazione iniziale del dispositivo;
- COMMUNICATION_MODE ⇒ di tipo std_logic, è un bit che indica quale dovrà essere la codifica applicata alle parole scambiate (8N1 oppure 7E1). In particolare:
    - 1 indica la codifica 7E1;

    - 0 indica la codifica 8N1;
- RTS_IN ⇒ di tipo std_logic, è un bit che indica se il buffer in ricezione possiede sufficiente spazio per lo stoccaggio di nuove informazioni;
- CLK ⇒ di tipo std_logic, è il "clock veloce del sistema". La sua frequenza è 8 volte quella reale di funzionamento del sistema;
- BUSY ⇒ di tipo std_logic, è un bit che indica che il dispositivo sta già trasmettendo un'altra parola e quindi che non è momentaneamente disponibile ad altre trasmissioni. In particolare:
    - 1 indica che, nel caso venisse asserito il segnale di start, il cambiamento di DIN verrà ignorato;
    - 0 indica che, nel caso venisse asserito il segnale di start, inizierà la trasmissione di DIN qualora il ricevente fosse disponibile;
- READY ⇒ di tipo std_logic, è il segnale avente durata pari a un ciclo di clock che attesta la ricezione di una nuova parola e quindi la necessità di prelevare il valore di DOUT;
- ERROR ⇒ di tipo std_logic, è un bit che indica se, nel caso di codifica 7E1, il bit di parità è corretto o meno. In particolare:
    - 1 indica che la parità non è corretta;
    - 0 indica che la parità è corretta;
- DOUT ⇒ di tipo std_logic_vector, è una parola di 8 bit codificata in binario naturale che deriva dalla de-serializzazione della parola ricevuta in ingresso dall'altra UART.

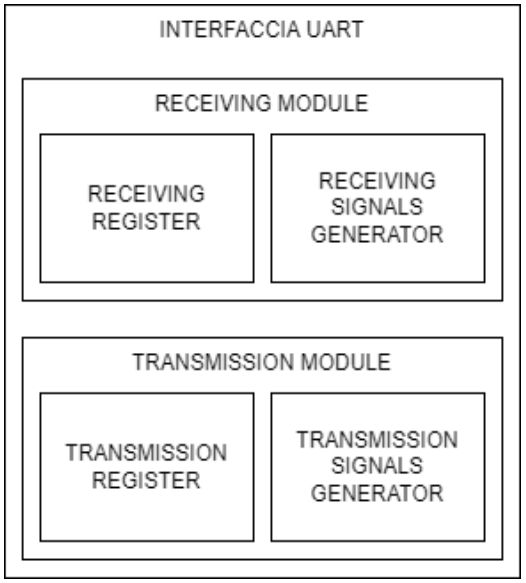
Sul lato destro, invece, troviamo i segnali che le due UART si scambiano tra di loro:

- TX ⇒ di tipo std_logic, è un segnale che rappresenta la linea di trasmissione seriale verso un'altra UART. Quando inizia la trasmissione, inizia sempre con uno '0' (bit di start); i successivi 8 bit contengono la parola contenuta in DIN codificata in modo appropriato (8N1 oppure 7E1); infine l'ultimo bit è sempre un '1' (bit di stop). Quando non vi è nulla da trasmettere, TX mantiene un valore fisso pari a '1'.
- RX ⇒ di tipo std_logic, è un segnale che rappresenta la linea di ricezione seriale;
- RTS_OUT ⇒ di tipo std_logic, è un segnale che informa la UART collegata se può procedere con la trasmissione di nuove parole oppure se è impossibilitata in quanto non si ha abbastanza spazio nel buffer di ricezione;
- CTS ⇒ di tipo std_logic, è un segnale che indica se il destinatario è incline a ricevere una nuova parola: nel caso in cui valesse '0', un'eventuale asserzione del segnale di start verrebbe ignorata;

Esistono diverse relazioni tra i segnali dell'interfaccia UART:

- Se CTS = '1', alla ricezione dello START avremo che il segnale di BUSY effettuerà una transizione verso il valore '1' sul primo fronte di clock disponibile;
- Il segnale RTS_OUT in uscita dalla UART è il valore campionato, sui fronti di clock, del segnale RTS_IN fornito dall'esterno;
- Se CTS = '0' avremo che TX = '1' a meno che il segnale di BUSY non risulti essere asserito, in questo caso si terminerà la trasmissione della parola corrente e solo successivamente il segnale TX verrà lasciato al valore '1';

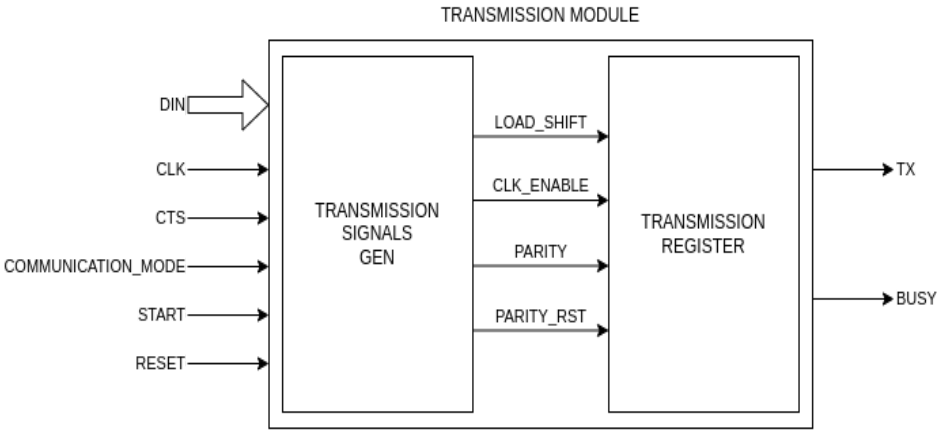
Architettura del sistema



Possiamo schematizzare un interfaccia UART effettuando una divisione interna basata sulle funzionalità che tale strumento deve garantire.
In particolare nella figura di fianco viene evidenziato come tale interfaccia sia costituita da 2 macro-moduli totalmente indipendenti l'uno dall'altro (non esiste alcuna connessione tra di loro). Rispettivamente il primo è destinato alla ricezione (RECEIVING_MODULE) mentre il secondo è adibito alla trasmissione (TRANSMISSION_MODULE).

Entrambi questi moduli sono ulteriormente suddivisi in un complesso formato da uno storage di informazioni realizzato attraverso l'utilizzo di FLIP-FLOP (RECEIVING REGISTER e TRANSMISSION REGISTER) e da un generatore di segnali.
Il generatore di segnali è realizzato sfruttando un contatore a 7 bit (basato su un sommatore RCA) che permette di ricoprire e scandagliare il lasso di tempo che intercorre dalla ricezione/trasmissione dello start bit fino alla ricezione/trasmissione dello stop bit. In questo modo a seconda del valore assunto dall'uscita del nostro contatore, possiamo, attraverso una rete combinatoria, generare i segnali che agiranno attivamente sui registri per farne mutare lo stato.

Modulo 1: TRANSMISSION_MODULE

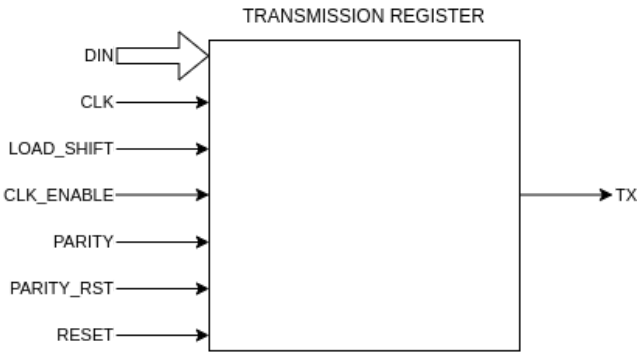


La figura mostra il funzionamento del modulo di trasmissione.
I segnali di ingresso e di uscita coincidono con quelli descritti nella parte di specifica relativa all'interfaccia del sistema.
Come accennato poc'anzi, ogni macromodulo ha una componente destinata alla generazione di segnali atti a garantire determinati comportamenti all'interno dei registri.
Per questa motivazione gli unici segnali interni sono diretti verso il sistema di memorizzazione dell'informazione.
Nel caso della trasmissione consideriamo per quanto concerne i registri una struttura che legge in parallelo i dati e li trasmette in modo seriale.

Analisi dei segnali interni:

- LOAD_SHIFT ⇒ di tipo std_logic, è il segnale che ci permette di caricare una determinata parola dall'esterno all'interno del registro di trasmissione. In particolare:
    - 1 indica che il registro deve caricare DIN al suo interno;
    - 0 indica che il registro può procedere normalmente con lo scorrimento dei dati.
- CLK_ENABLE ⇒ di tipo std_logic, è il segnale di enable per i flip flop utilizzati nel TRANSMISSION_REGISTER;
- PARITY ⇒ di tipo std_logic, è il segnale che individua l'istante di tempo di invio del bit di parità qualora la codifica delle parole inviate fosse 7E1;
- PARITY_RST ⇒ di tipo std_logic, è il segnale utilizzato per resettare il flip flop adibito al calcolo della parità ogni qual volta termina la trasmissione di una parola.

Modulo 1.1: TRANSMISSION_REGISTER



È una prima componente del TRANSMISSION_MODULE.

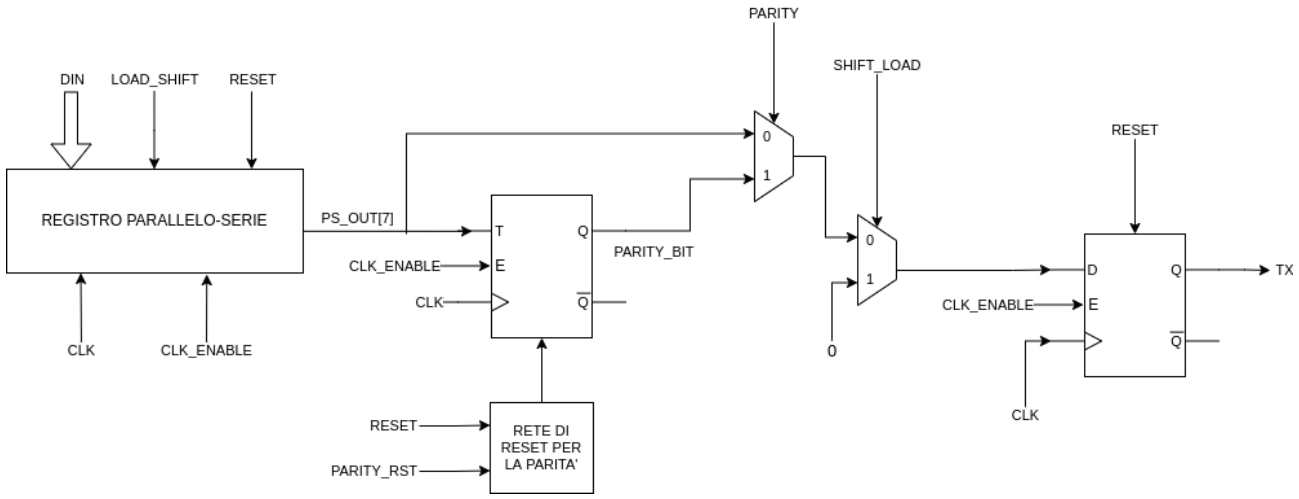
Essa è incaricata di 3 operazioni fondamentali: Calcolare la parità associata ad una parola fornita dall'utente, caricare in un registro locale tale parola e fornire una corretta uscita durante la trasmissione seriale.

I segnali forniti in input sono i seguenti:

- DIN ⇒ di tipo std_logic_vector, è la parola fornita dall'esterno;
- CLK ⇒ di tipo std_logic, è il clock di base del nostro strumento;
- LOAD_SHIFT ⇒ di tipo std_logic, è il segnale che indica se bisogna prelevare DIN dall'esterno o procedere con lo scorrimento dei valori all'interno dei registri;
- CLK_ENABLE ⇒ di tipo std_logic, è il segnale che comunica ai flip flop costituenti il registro se essere trasparenti e quindi essere soggetti alla mutazione del loro stato in corrispondenza dei fronti di clock;
- PARITY ⇒ di tipo std_logic, è il segnale che individua l'istante di tempo di invio del bit di parità qualora la codifica delle parole inviate fosse 7E1; esso viene utilizzato per la selezione del successivo valore da trasmettere tra l'uscita del flip flop adibito al calcolo della parità oppure il valore del flip flop precedente;
- PARITY_RST ⇒ di tipo std_logic, è il segnale utilizzato per resettare il flip flop adibito al calcolo della parità ogni qual volta termina la trasmissione di una parola;
- RESET ⇒ di tipo std_logic, è il segnale fornito all'accensione del dispositivo per inizializzare i campi dei flip flop interni.

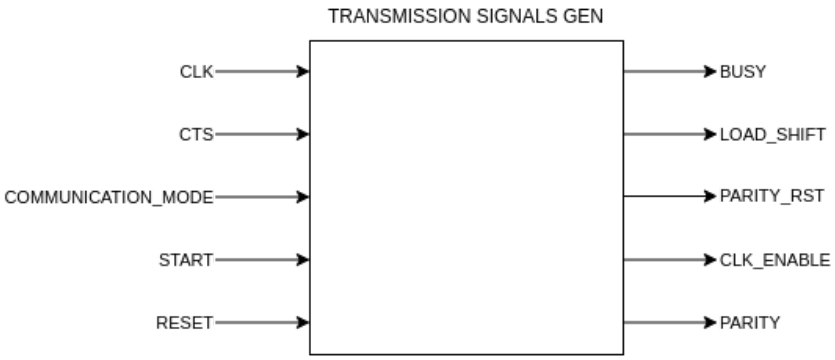
I segnali forniti in output sono i seguenti:

- TX ⇒ di tipo std_logic, è il segnale che rappresenta il valore che stiamo trasmettendo.



Dalla figura riportata osserviamo com'è organizzata la struttura del nostro modulo.
Il tutto è basato su un classico registro parallelo-serie al quale viene aggiunta una struttura per l'invio del corretto valore TX. Particolarità di tale registro è che quando si procede con lo scorrimento dei valori al suo interno, nel flip flop più lontano da quello contenente l'uscita seriale viene inserito sempre il valore '1'. L'uscita seriale di tale registro viene indicata con PS_OUT[7] in quanto corrisponde all'ottavo flip flop (partendo a contare da 0).
Tale uscita viene utilizzata in modo duplice, è entrata di un flip flop T per il calcolo della parità man mano che si trasmette la parola ma anche uno dei candidati ad essere il successivo valore ad essere trasmesso. Quando stiamo effettuando lo scorrimento dei registri (LOAD_SHIFT = '0') , usufruiamo del segnale PARITY per scegliere tra il PARITY_BIT calcolato e PS_OUT[7] per individuare il prossimo valore TX. Se invece stiamo caricando i registri con la parola DIN (LOAD_SHIFT = '1'), il valore inserito nel flip flop con uscita TX sarà esattamente '0'.

Modulo 1.2: TRANSMISSION_SIGNALS_GEN



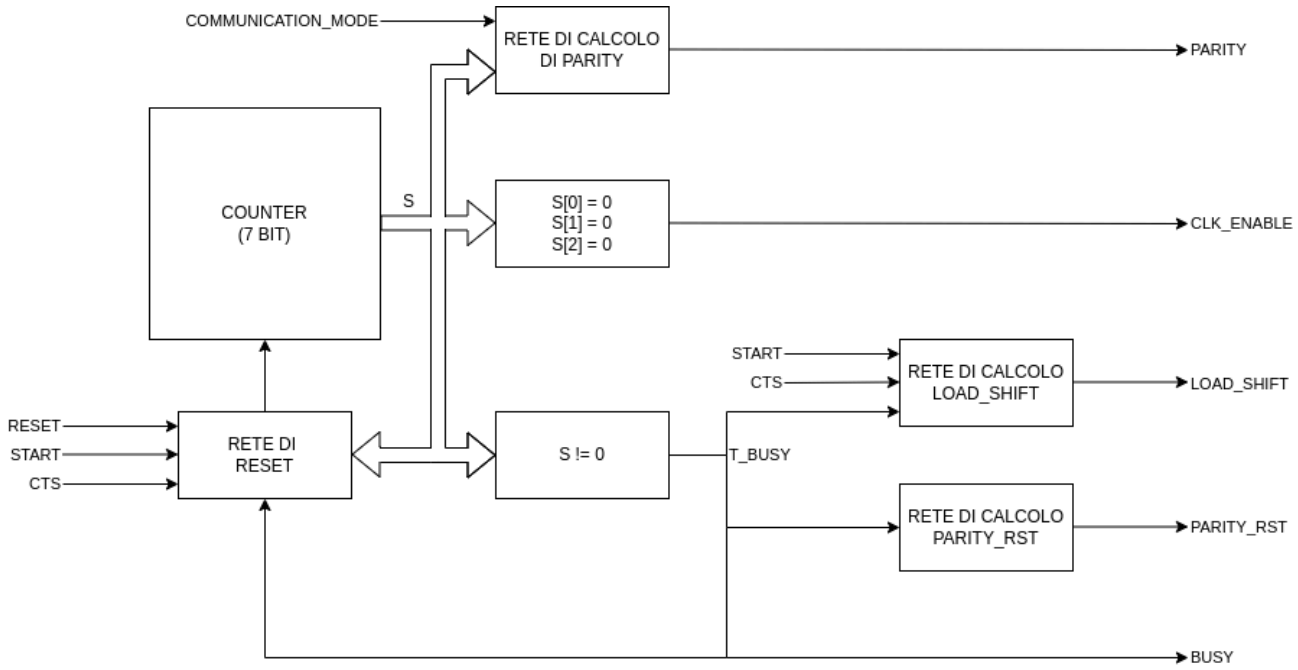
È il secondo componente del TRANSMISSION_MODULE.
Esso è incaricato della generazione dei segnali atti al caricamento della parola dall'esterno all'interno dei registri di trasmissione e alla sua successiva trasmissione.

I segnali forniti in input sono i seguenti:

- CLK ⇒ di tipo std_logic, è il clock di base del nostro strumento;
- CTS ⇒ di tipo std_logic, è un segnale che indica se il destinatario è incline a ricevere una nuova parola;
- COMMUNICATION_MODE ⇒ di tipo std_logic, è un bit che indica quale dovrà essere la codifica applicata alle parole scambiate (8N1 oppure 7E1);
- START ⇒ di tipo std_logic, è il segnale che attesta che i valori forniti in ingresso sono corretti e che quindi si può procedere con la trasmissione;
- RESET ⇒ di tipo std_logic, è il segnale fornito dall'esterno che garantisce l'azzeramento del contatore interno all'accensione del dispositivo.

I segnali forniti in output sono i seguenti:

- BUSY ⇒ di tipo std_logic, è il segnale che attesta che si sta procedendo con una trasmissione;
- LOAD_SHIFT ⇒ di tipo std_logic, è il segnale che indica se bisogna procedere con il caricamento di una nuova parola o con lo scorrimento;
- PARITY_RST ⇒ di tipo std_logic, è il segnale utilizzato per resettare il flip flop adibito al calcolo della parità ogni qual volta termina la trasmissione di una parola;
- CLK_ENABLE ⇒ di tipo std_logic, è il segnale di enable da fornire ai flip flop del TRANSMISSION_REGISTER;
- PARITY ⇒ di tipo std_logic, è il segnale che individua l'istante di tempo di invio del bit di parità qualora la codifica delle parole inviate fosse 7E1.

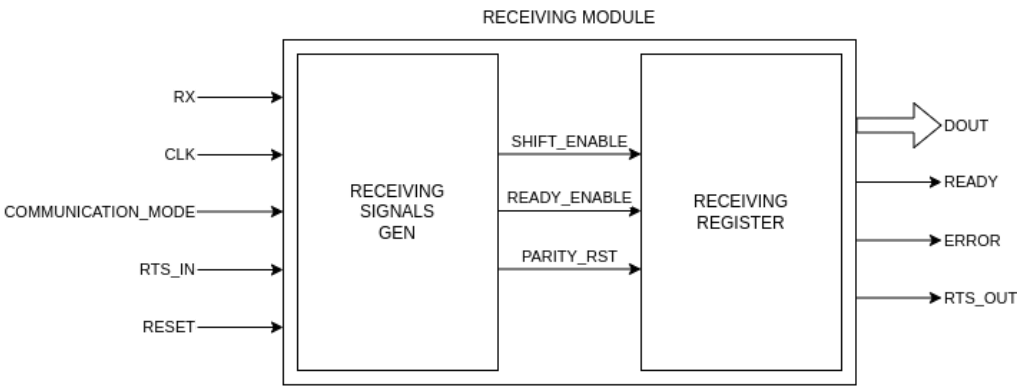


Alla base del nostro generatore di segnale c'è un contatore a 7 bit che permette di scandagliare il lasso di tempo in cui procediamo con la trasmissione.
Questo contatore avrà la sua unica uscita S fissata al valore "0000000" dal segnale di RESET iniziale, dopo di che il suo valore rimarrà fisso fintanto che esso risulta non occupato nel conteggio e il segnale START o il segnale CTS sono pari a '0'. Quando invece è impegnato nel conteggio abbiamo che esso viene riportato al valore "0000000" solo quando l'uscita assume un valore tale da garantire la completa trasmissione della parola.

L'uscita del contatore viene anche utilizzata per generare i segnali che si occuperanno di manipolare lo stato dei nostri registri o segnali che saranno visibili all'utilizzatore della UART.
In particolare abbiamo i seguenti segnali:

- BUSY: segnale che risulta asserito quando l'uscita del contatore è diversa da "0000000"; viene generato da una porta OR a 7 bit;
- CLK_ENABLE: segnale che risulta asserito quando l'uscita del contatore modulo 8 è uguale a 0 (S mod 8 = 0);
- PARITY: segnale calcolato tramite una porta AND a 8 ingressi avente in entrata il COMMUNICATION_MODE e un determinato valore dell'uscita del contatore;
- PARITY_RST: segnale ottenuto negando BUSY;
- LOAD_SHIFT: segnale che risulta asserito quando BUSY = '0', START = '1' e CTS = '1'.

Modulo 2: RECEIVING_MODULE

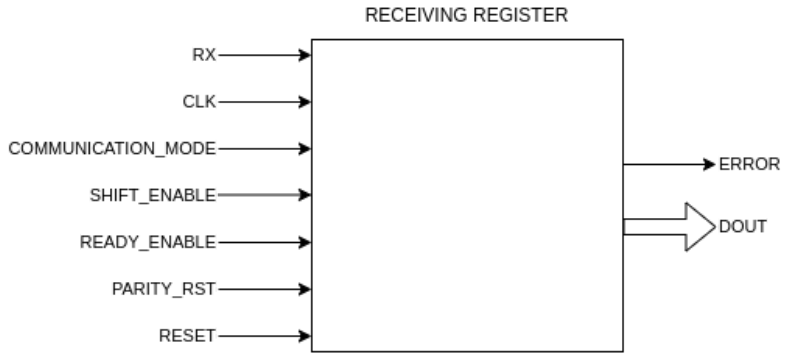


La figura mostra il funzionamento del modulo di ricezione.
I segnali di ingresso e uscita coincidono con quelli descritti nella parte di specifica relativa all'interfaccia del sistema.
Per il modulo di ricezione abbiamo una situazione analoga a quella riscontrata per il modulo di trasmissione.
Avremo infatti che le operazioni di ricezione sono totalmente contenute nel sottomodulo dei registri e che il coordinamento del tutto viene garantito dal sottomodulo di generazione dei segnali.
Anche in questo caso avremo che tutti i segnali interni di interazione tra i moduli saranno diretti dal generatore di segnali verso i registri.

Analisi dei segnali interni:

- SHIFT_ENABLE ⇒ di tipo std_logic, è il segnale di enable utilizzato dal RECEIVING REGISTER per permettere a determinati flip flop di cambiare stato;
- READY_ENABLE ⇒ di tipo std_logic, è il segnale di enable utilizzato dal RECEIVING REGISTER per permettere ai flip flop, che mostrano i dati all'utilizzatore, di cambiare stato;
- PARITY_RST ⇒ di tipo std_logic, è il segnale utilizzato per resettare il flip flop T adibito al controllo della parità della parola ricevuta.

Modulo 2.1: RECEIVING_REGISTER



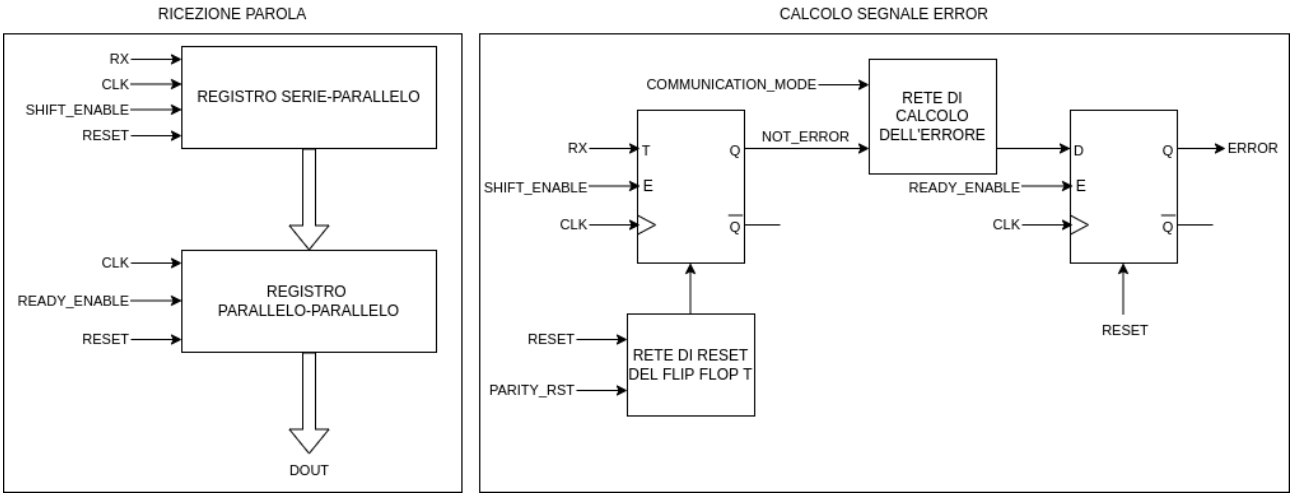
È il modulo che consente all'utente finale la fruizione dei dati ricevuti.
Esso è basato su 2 diversi registri: uno serie-parallelo e l'altro parallelo-parallelo.
Le principali operazioni di cui è incaricato sono: la lettura dei segnali, il controllo di parità e caricamento dei dati in un buffer di uscita.
I segnali forniti in input sono i seguenti:

- RX ⇒ di tipo std_logic, è il segnale che rappresenta i valori che vengono ricevuti dal nostro strumento istante per istante;
- CLK ⇒ di tipo std_logic, è il clock di funzionamento del dispositivo;

- COMMUNICATION_MODE ⇒ di tipo std_logic, è un bit che indica quale dovrà essere la codifica applicata alle parole scambiate (8N1 oppure 7E1);
- SHIFT_ENABLE ⇒ di tipo std_logic, è il segnale di enable del registro serie-parallelo;
- READY_ENABLE ⇒ di tipo std_logic, è il segnale di enable del registro parallelo-parallelo;
- PARITY_RST ⇒ di tipo std_logic, è il segnale utilizzato per resettare il flip flop T adibito al controllo della parità della parola ricevuta;
- RESET⇒ di tipo std_logic, è il segnale fornito all'accensione del dispositivo per inizializzare i campi dei flip flop interni.

I segnali forniti in output sono i seguenti:

- ERROR ⇒ di tipo std_logic è il segnale che evidenzia se la parità associata alla parola ricevuta risulta rispettata o meno nel caso di COMMUNICATION_MODE = '1';
- DOUT ⇒ di tipo std_logic_vector è il vettore che rappresenta la parola ricevuta.



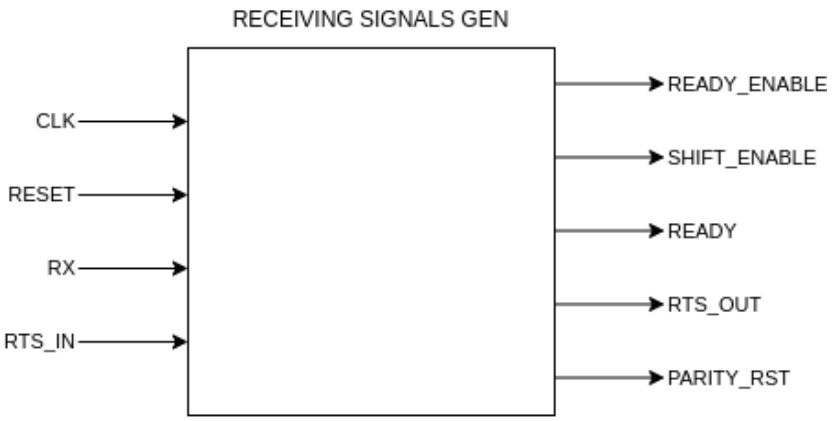
La struttura per la de-serializzazione delle parole ricevute si basa, come accennato poc'anzi, su 2 tipologie di registri: un registro serie-parallelo e un registro parallelo-parallelo.

Il registro serie-parallelo viene utilizzato per la lettura sequenziale dei bit in ricezione. Lo scorrimento dei dati al suo interno è eseguito in corrispondenza dei fronti di salita del clock quando il segnale SHIFT_ENABLE risulta essere asserito.

Il registro parallelo-parallelo invece viene utilizzato in corrispondenza della completa ricezione della parola e serve a renderla disponibile all'utente finale sui fronti di salita del clock quando il segnale READY_ENABLE risulta essere asserito.

Per quanto concerne il calcolo del segnale ERROR consideriamo una struttura basata su 2 flip flop. Utilizziamo un flip flop T che calcola la parità associata alla parola ricevuta considerando anche lo stop bit (tale segnale è chiamato NOT_ERROR poiché se stessimo considerando solo la parola di 8 bit esso risulterebbe asserito qualora la parità calcolata fosse corretta). Dopo di che l'uscita del flip flop t negata in AND logico con la COMMUNICATION_MODE garantirà il corretto valore del segnale ERROR, il quale verrà salvato nel flip flop d sul fronte di salita del clock quando READY_ENABLE è asserito.

Modulo 2.2: RECEIVING_SIGNALS_GEN



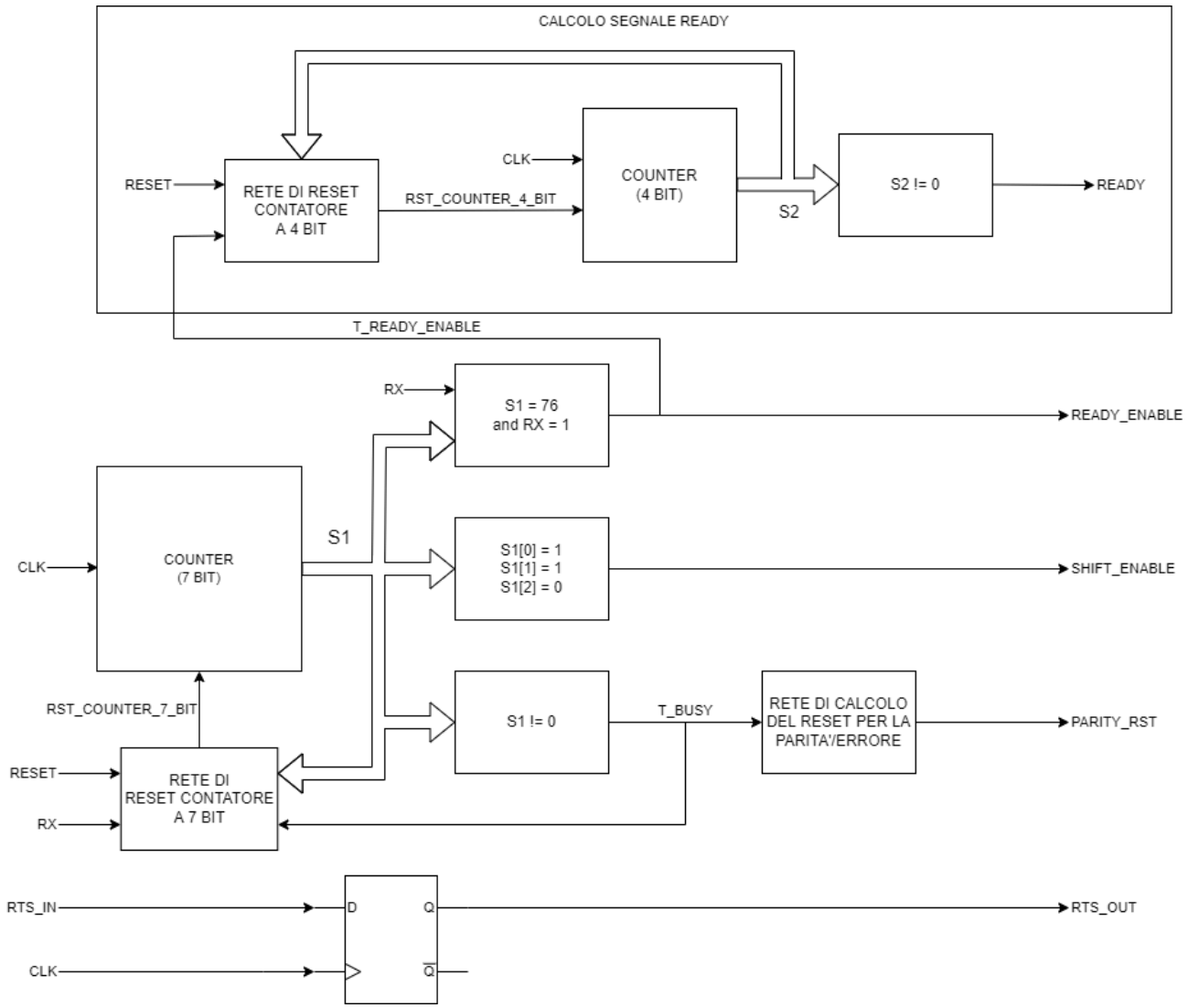
È il modulo incaricato di fornire i segnali per effettuare l'operazione di sampling della linea di ricezione e indicare gli istanti di tempo in cui la parola risulta essere completamente consegnata.

I segnali forniti in input sono i seguenti:

- CLK ⇒ di tipo std_logic è il clock di base del nostro strumento;
- RESET ⇒ di tipo std_logic, è il segnale fornito all'accensione del dispositivo per inizializzare i campi dei flip flop interni;
- RX ⇒ di tipo std_logic, rappresenta i valori che vengono ricevuti dal nostro strumento istante per istante;
- RTS_IN ⇒ di tipo std_logic, è un bit che indica se il buffer in ricezione possiede sufficiente spazio per lo stoccaggio di nuove informazioni;

I segnali forniti in output sono i seguenti:

- READY_ENABLE ⇒ di tipo std_logic, è il segnale di enable che permette il cambio di stato ai flip flop contenenti le informazioni visibili dall'utente finale. È asserito solo dopo aver ricevuto interamente la parola;
- SHIFT_ENABLE ⇒ di tipo std_logic, è il segnale di enable che permette il cambio di stato dei flip flop costituenti il registro serie-parallelo nel RECEIVING_REGISTER;
- READY ⇒ di tipo std_logic, è il segnale avente durata pari a un ciclo di clock che attesta la ricezione di una nuova parola e quindi la necessità di prelevare il valore di DOUT;
- RTS_OUT ⇒ di tipo std_logic, è un segnale che informa la UART collegata se può procedere con la trasmissione di nuove parole oppure se è impossibilitata in quanto non si ha abbastanza spazio nel buffer di ricezione;
- PARITY_RST ⇒ di tipo std_logic, è il segnale utilizzato per resettare il flip flop adibito al calcolo della parità ogni qual volta termina la ricezione di una parola.



Alla base del nostro generatore di segnale c'è un contatore a 7 bit che permette di scandagliare il lasso di tempo in cui procediamo con la ricezione.
Questo contatore avrà la sua unica uscita S1 fissata al valore "0000000" dal segnale di RESET iniziale, dopo di che il suo valore rimarrà fisso fintanto che esso risulta non occupato nel conteggio (ovvero la sua uscita corrisponde a "0000000") e il segnale RX è pari a '1'. Quando invece è impegnato nel conteggio abbiamo che esso viene riportato al valore "0000000" solo quando l'uscita assume un valore tale da garantire la completa ricezione della parola.

L'uscita del contatore viene anche utilizzata per generare i segnali che si occuperanno di manipolare lo stato dei nostri registri o segnali che saranno visibili all'utilizzatore della UART.
In particolare abbiamo i seguenti segnali:

- BUSY: segnale che risulta asserito quando l'uscita del contatore è diversa da "0000000"; viene generato da una porta OR a 7 bit;
- READY_ENABLE: segnale che risulta asserito quando l'uscita del contatore è pari a 76 e RX = '1' (corrispondente allo stop bit);
- SHIFT_ENABLE: segnale che risulta asserito quando l'uscita del contatore modulo 8 è uguale a 3 (S1 mod 8 = 3);
- PARITY_RST: segnale ottenuto negando BUSY;

A questi segnali possiamo aggiungere RTS_OUT, ottenuto campionando RTS_IN in corrispondenza dei fronti di clock.

L'ultimo segnale calcolato è quello di READY il quale è ottenuto tramite un contatore a 4 bit.
Questo contatore è settato ad un'uscita iniziale pari a "0000" tramite il segnale di RESET. Dopo di che rimarrà tale valore fintanto che il segnale READY_ENABLE è pari a '0'. Non appena ci sarà la completa ricezione di una parola e quindi READY_ENABLE = '1', tale contatore avrà un incremento ad ogni ciclo di clock fino a raggiungere il valore "1000". Il segnale di READY sarà ottenuto da un OR logico tra tutti i bit costituenti l'uscita di questo contatore.

Modulo 3: RCA

È il modulo di base utilizzato per la realizzazione dei contatori.
Come lascia intendere il nome, si tratta di un sommatore basato su un'architettura Ripple-Carry.
In ingresso avrà i segnali:

- X e Y ⇒ di tipo std_logic_vector, sono i 2 valori da sommare;
- CIN ⇒ di tipo std_logic, è il riporto in ingresso.

In uscita avremo:

- S ⇒ di tipo std_logic_vector, è il risultato della somma.

In quanto Ripple-Carry utilizziamo n Full Adder per il calcolo della somma. Ogni Full Adder provvederà al riporto da fornire al Full Adder successivo e al valore del corrispondente bit di uscita che comporrà la somma finale.

Modulo 4: COUNTER

È il modulo che rappresenta un classico contatore a n bit soggetto ad un incremento pari a 1.
In ingresso avrà i segnali:

- CLK ⇒ di tipo std_logic, è il clock sul quale si aggiornerà l'uscita del contatore.
- RST ⇒ di tipo std_logic, è il segnale di reset del contatore.

In uscita avrà il segnale:

- S ⇒ di tipo std_logic_vector, è l'uscita corrente del contatore.

La sua realizzazione è basata sul modulo RCA e da una serie di flip-flop che permettono di salvare il valore corrente calcolato. In particolare avremo che l'ingresso X del RCA corrisponderà al valore salvato nei registri, l'ingresso Y corrisponde all'incremento a cui sarà soggetto il nostro contatore (in questo caso 1) e il CIN sarà pari a '0'.
L'uscita del nostro RCA verrà campionata e salvata nei registri in corrispondenza dei fronti di clock.
Per quanto riguarda invece il segnale RST, esso è sincrono e permette di settare il valore S del contatore a 0.

VERIFICA

Test-bench

Il test-bench realizzato prevede l'utilizzo di 2 unità UART, rispettivamente chiamate UART1 e UART2, interconnesse tra di loro. Le due unità verranno testate sulla ricezione e sulla trasmissione dimostrando che le parole trasmesse dal modulo UART1 vengono ricevute correttamente dal modulo UART2 e viceversa. Entrambi i moduli dovranno riuscire a trasmettere il corretto bit di parità qualora la codifica lo richieda.

Per una miglior visione del test-bench vengono introdotti 2 segnali aggiuntivi, uno per la trasmissione da UART1 a UART2 e l'altro per la trasmissione da UART2 a UART1. Essi risulteranno asseriti ogni qualvolta, in seguito ad una trasmissione, i valori nei registri in ricezione risulteranno coincidere con gli expected output. Qualora il valore in ricezione dovesse risultare diverso dagli expected output avremo che i relativi segnali di verifica saranno posti al valore '0'.

Casi d'uso

I casi d'uso trattati dal Test-bench realizzato sono i seguenti:

- Trasmissione/Ricezione tra 2 unità UART con frequenza di funzionamento leggermente diversa per verificare la corretta lettura dei valori trasmessi in situazioni in cui per motivazioni di carattere fisico abbiamo che uno dei 2 strumenti (nel test-bench l'unità UART2) non riproduce fedelmente la frequenza impostata;
- Verifica mancata trasmissione quando si riceve un segnale di START ma il segnale CTS che attesta che il ricevente è pronto ad assorbire nuove informazioni risulta non asserito;
- Verifica trasmissione contemporanea tra le 2 unità UART;
- Verifica possibilità di trasmettere bit in modo continuo tra le 2 unità (trasmissione start bit successivamente ad uno stop bit);
- Verifica comportamento delle unità UART in seguito al passaggio dell'RTS del ricevente al valore '0' durante una trasmissione;
- Verifica corretto calcolo del segnale ERROR;
- Verifica corretta trasmissione delle parole fornite in ingresso alle 2 unità e verifica del calcolo del bit di parità quando la codifica lo richiede. Verranno trasmesse 7 parole da UART 1 a UART 2 e 5 parole da UART2 a UART1;