

A decorative graphic on the left side of the slide consists of a network of thin, light blue lines. These lines are arranged in a way that resembles a circuit board or a neural network, with several small circles at the end of the lines, suggesting nodes or connections. The lines and circles are more densely packed on the left and become sparser towards the right.

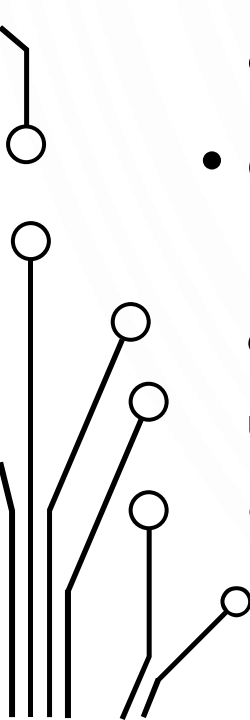
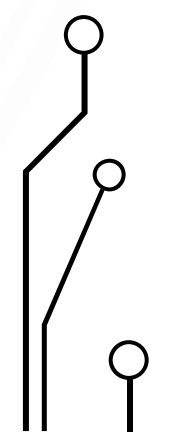
COMUNICAZIONE TRA DISPOSITIVI TRAMITE INTERFACCIA SERIALE

GIOVANNI COZZOLINO



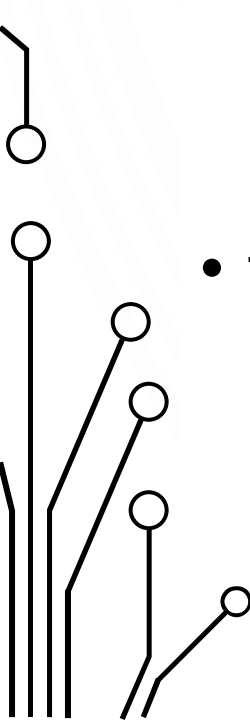
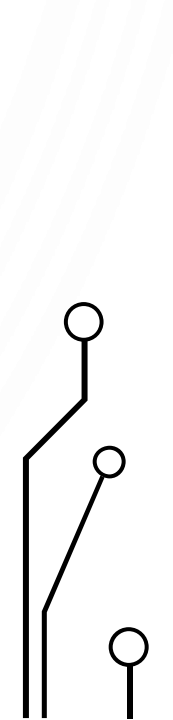
UART - USART



- Lo UART o Universal Asynchronous Receiver-Transmitter (ricevitore-trasmettitore asincrono universale) è un dispositivo hardware, di uso generale o dedicato, che converte flussi di bit di dati da un formato parallelo a un formato seriale asincrono o viceversa.
 - Gli USART (Universal Synchronous-Asynchronous Receiver/Transmitter), hanno il compito di gestire le comunicazioni delle interfacce seriali RS-232. Costituiscono un'evoluzione degli UART, come si evince dal nome stesso, in grado di gestire anche trasmissioni seriali sincrone.
- 
- 

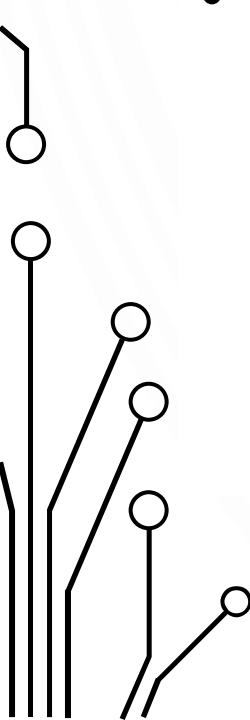
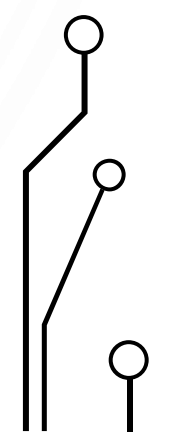


CARATTERISTICHE

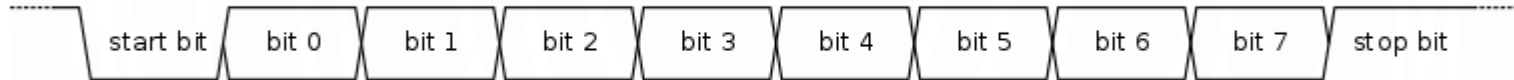
- Un UART è di solito un singolo (o parte di) un circuito integrato (IC) utilizzato per le comunicazioni seriali su una porta seriale di un computer o di una periferica.
 - Una o più periferiche UART sono comunemente integrate nei chip del microcontrollore
 - La trasmissione seriale attraverso un singolo filo è meno costosa della trasmissione parallela attraverso più fili.
 - Tipicamente un UART è un dispositivo in cui:
 - sono configurabili il formato dei dati e le velocità di trasmissione
 - i livelli e i metodi di segnalazione elettrica sono gestiti da un circuito di pilotaggio esterno all'UART.
- 
- 



TRASMISSIONE E RICEZIONE DEI DATI

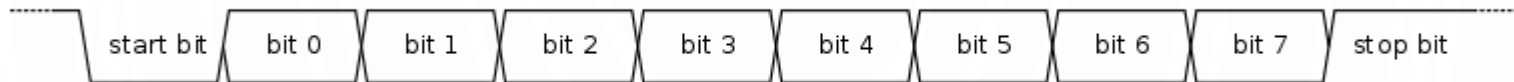
- Ogni UART contiene un registro a scorrimento, che è il metodo fondamentale di conversione tra forme seriali e parallele.
 - La comunicazione può essere
 - **simplex** (in una sola direzione, senza che il dispositivo ricevente possa inviare informazioni al dispositivo trasmittente),
 - **full duplex** (entrambi i dispositivi inviano e ricevono contemporaneamente)
 - **half duplex** (i dispositivi a turno trasmettono e ricevono).
- 
- 

FRAME DEI DATI



- Lo stato **inattivo** ha il livello logico alto.
 - Questo è un retaggio storico della telegrafia, in cui la linea è tenuta in alto per mostrare che la linea e il trasmettitore non sono danneggiati.
- Ogni carattere è racchiuso tra
 - Uno **start bit**, con livello logico basso, segnala al ricevitore che sta arrivando un nuovo carattere
 - 5-9 **bit dati**, a seconda del set di codici utilizzato, rappresentano il carattere (dato) da trasmettere.
 - (eventualmente) un **bit di parità**
 - uno o due **stop bit**, sempre nella condizione mark (logico alto), segnalano al destinatario che il carattere è completo.

DETTAGLI SUL FRAME DATI

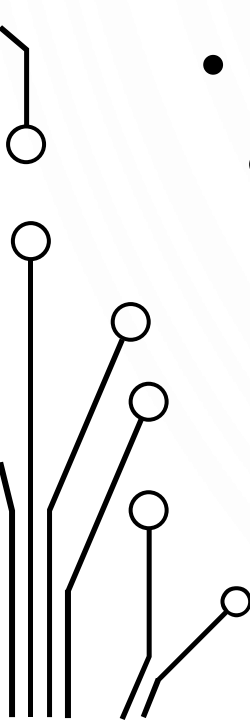
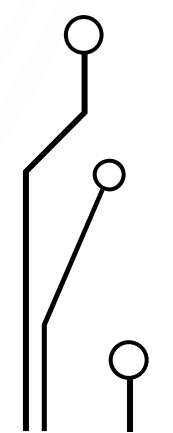


- Nella maggior parte delle applicazioni viene trasmesso per primo il bit di dati meno significativo (quello a nel data frame), ma ci sono eccezioni.
- Poiché il bit di avvio è logico basso (0) e il bit di arresto è logico alto (1), ci sono sempre almeno due cambi di segnale garantiti tra i caratteri.
- Se la linea viene mantenuta nella condizione logica bassa per un periodo di tempo superiore a un carattere, questa è una condizione di interruzione che può essere rilevata dall'UART.



PROTOCOLLO DI RICEZIONE (1 / 2)

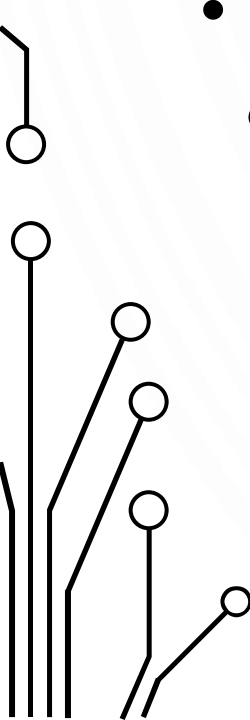
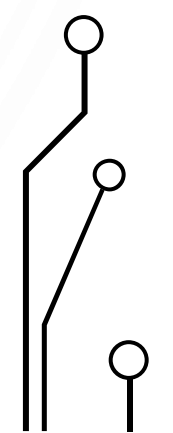


- Tutte le operazioni dell'UART sono controllate da un segnale di clock interno che oscilla ad una frequenza multipla della velocità di trasmissione dei dati,
 - in genere 8 o 16 volte la velocità di trasmissione.
 - Il ricevitore verifica lo stato del segnale in ingresso ad ogni colpo di clock, in attesa dello start bit.
 - Se il potenziale start bit dura almeno la metà del periodo di campionamento, esso è valido e segnala l'inizio di un nuovo carattere.
 - In caso contrario è considerato un impulso spurio e viene ignorato.
- 
- 



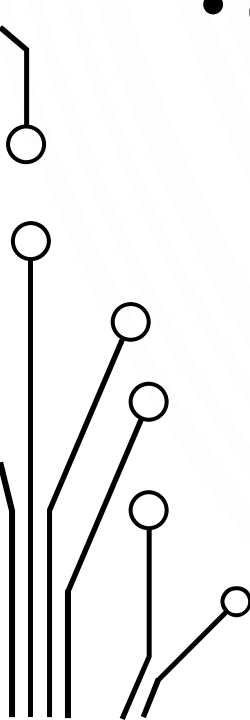
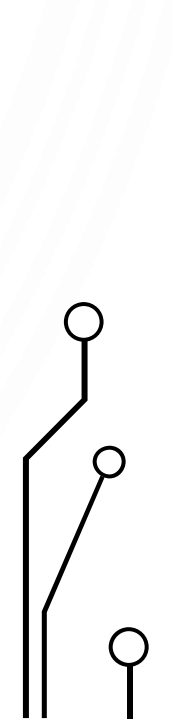
PROTOCOLLO DI RICEZIONE (2/2)



- La linea viene quindi campionata periodicamente a centro periodo e il livello risultante viene registrato in un registro a scorrimento.
 - Dopo un numero di periodi pari alla lunghezza del carattere (da 5 a 8 bit, in genere), il contenuto del registro a scorrimento viene reso disponibile (in modo parallelo) in output.
 - L'UART imposterà un flag indicante la disponibilità di nuovi dati e potrebbe anche generare un interrupt al processore per richiedere il trasferimento dei dati ricevuti.
- 
- 

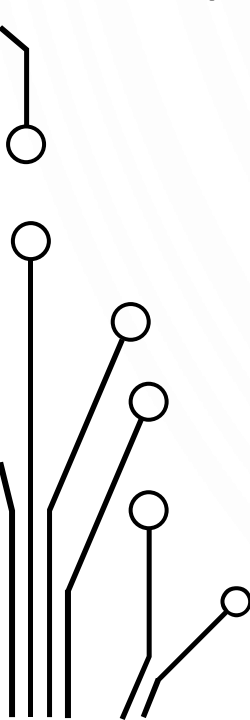
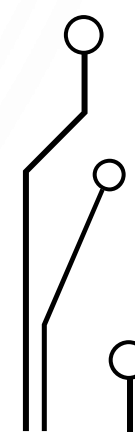


SINCRONIZZAZIONE DEGLI UART

- Gli UART comunicanti di solito non hanno un sistema di temporizzazione condiviso, a parte il segnale di comunicazione.
 - Gli UART risincronizzano i loro clock interni
 - ad ogni variazione della linea di dati che non sia un impulso spurio, oppure
 - sul fronte di discesa dello start bit, e campionando il centro di ciascun bit di dati
 - questo sistema funziona se la velocità di trasmissione dei dati è sufficientemente accurata da consentire ai bit di arresto di essere campionati in modo affidabile.
- 
- 

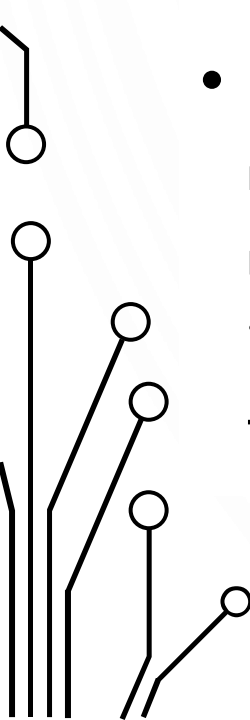
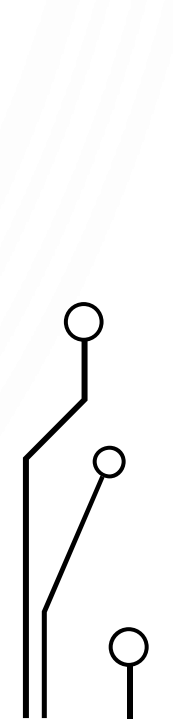


PROTOCOLLO DI TRASMISSIONE

- L'operazione di trasmissione è più semplice in quanto non è necessario determinare la temporizzazione dallo stato della linea, né è vincolata a intervalli di temporizzazione fissi.
 - Non appena il sistema trasmittente deposita un carattere nel registro a scorrimento (dopo il completamento del carattere precedente), l'UART
 - genera uno start bit,
 - sposta il numero richiesto di bit di dati sulla linea,
 - genera e invia il bit di parità (se utilizzato),
 - invia i gli stop bit.
- 
- 

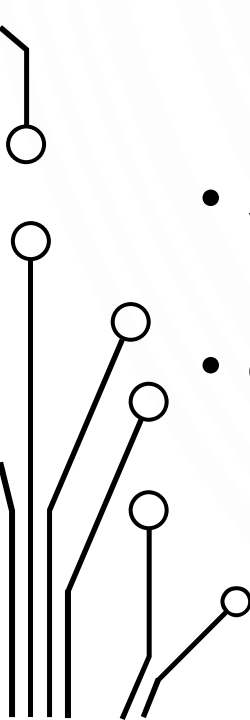
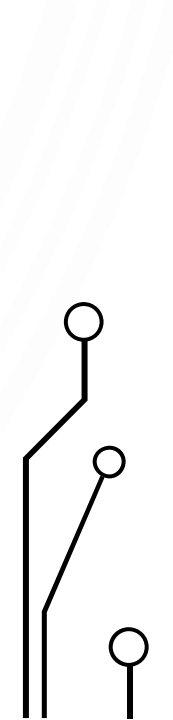


DISPONIBILITÀ DEI DATI

- Poiché la modalità full duplex richiede che i caratteri vengano inviati e ricevuti contemporaneamente, gli UART utilizzano due diversi registri a scorrimento per i caratteri trasmessi e ricevuti.
 - Poiché la trasmissione di uno o più caratteri può richiedere molto tempo rispetto alla velocità della CPU, un UART mantiene un flag che mostra lo stato di occupato in modo che il sistema host sappia se c'è almeno un carattere nel buffer di trasmissione o nel registro a scorrimento
- 
- 

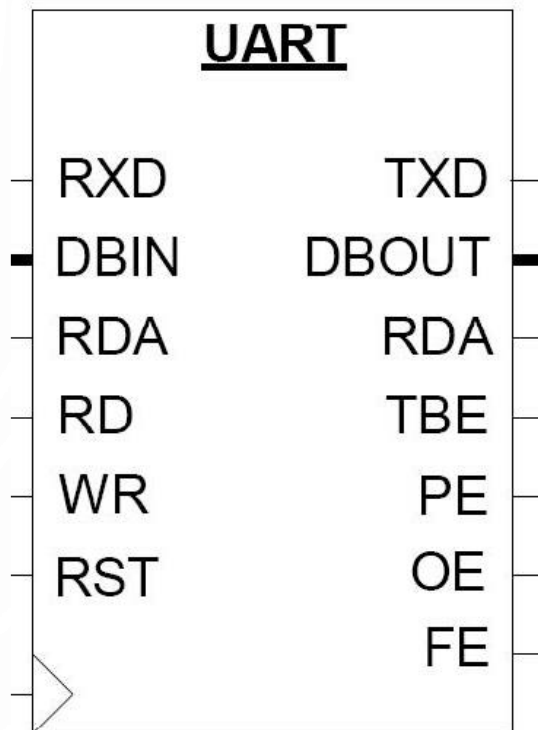


CONFIGURAZIONE DELLA COMUNICAZIONE

- Per garantirne il corretto funzionamento i dispositivi di trasmissione e ricezione devono accordarsi su:
 - velocità di bit (**baud rate**),
 - lunghezza dei caratteri,
 - numero di bit di parità
 - numero di stop bit.
 - Se l'UART ricevente rileva una configurazione errata può impostare un bit di flag "errore di framing."
 - **Configurazione tipica:** otto bit di dati, nessuna parità, un bit di stop.
 - per questa configurazione il numero di caratteri ASCII al secondo è uguale alla bit rate divisa per 10.
- 
- 

DIGILENT RS232 COMPONENT

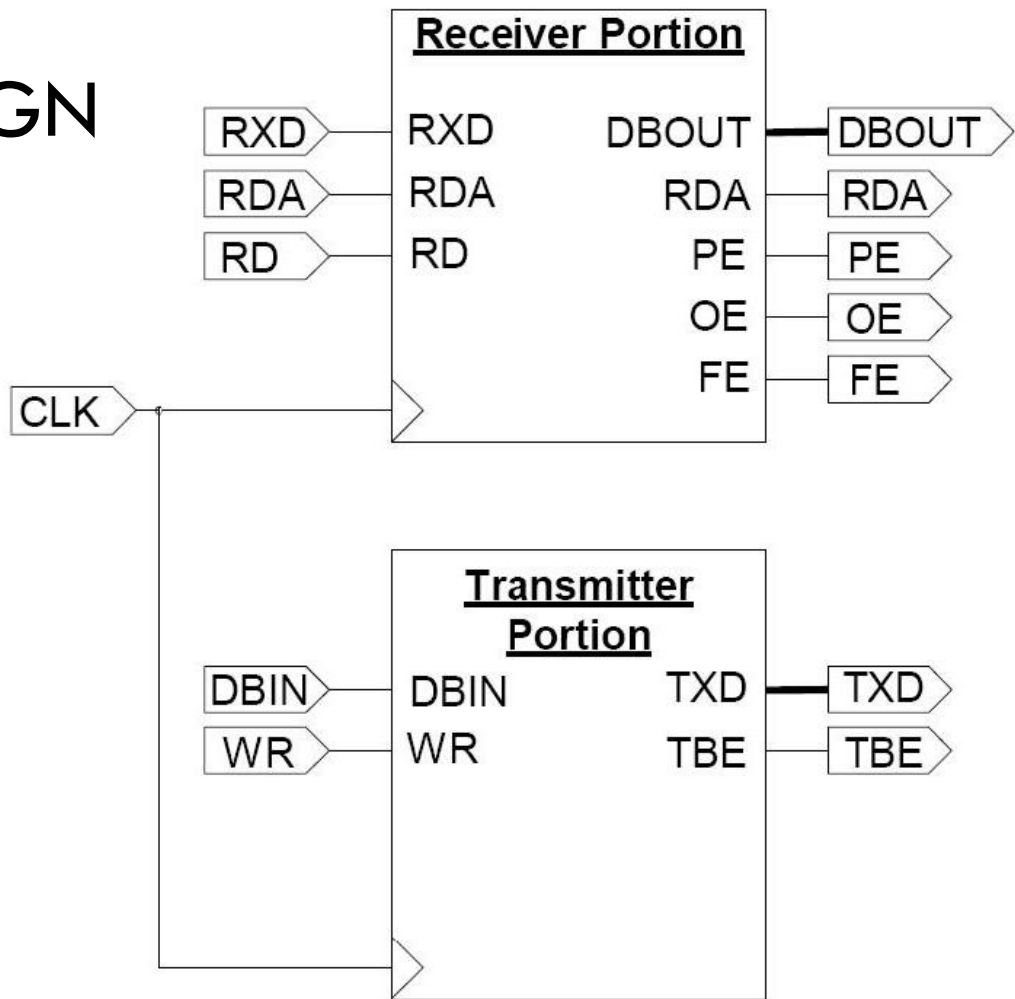
- **Underrun error (TBE)**, quando il trasmettitore UART ha completato l'invio di un carattere e il buffer di trasmissione è vuoto. Nelle modalità asincrone questo viene trattato come un'indicazione che non rimane alcun dato da trasmettere
- **Errore di parità (PE)**, si verifica quando la parità dei bit dati non corrisponde col bit di parità.
- **Overrun error (OE)**, quando il destinatario non è in grado di elaborare il carattere appena arrivato prima che arrivi quello successivo.
- **Framing error (FE)**, se la linea di dati non si trova nello stato previsto (alto) quando è previsto il bit di stop (in base al numero di bit di dati e di parità per i quali è impostato l'UART)



INTERNAL DESIGN

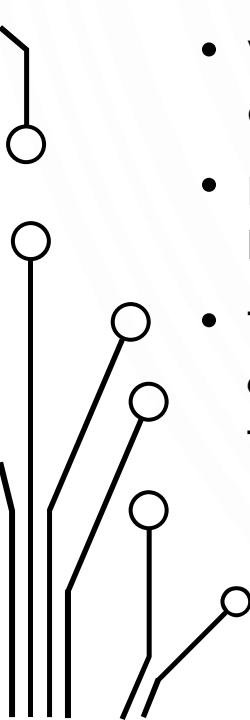
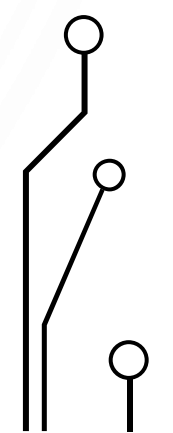
The **receiver** takes in a byte of serial data transmitted to the RXD port, and converts it into one byte of parallel information. This byte is then placed on the DBOUT port.

The **transmitter** takes a byte of parallel information found on the DBIN port, and converts it into a byte of serially transmitted data. This data is transmitted on the TXD port.





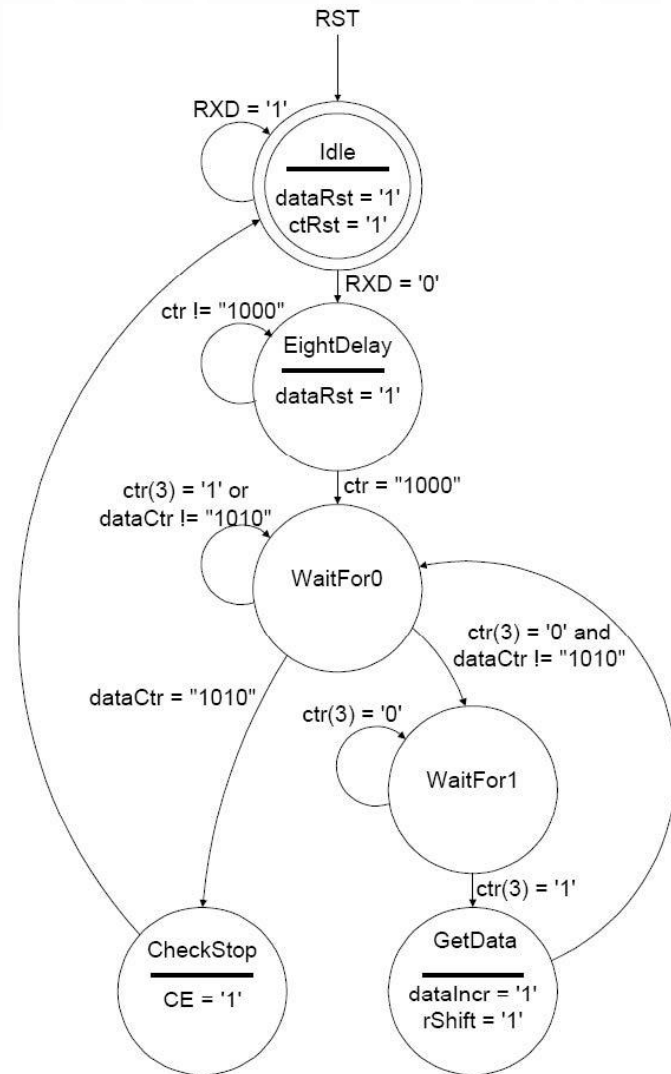
IMPLEMENTAZIONE DEL RICEVITORE

- The receiver portion of the UART accepts serial data input, and converts it into parallel data. This portion includes:
 - a serial data controller,
 - two counters used for synchronization,
 - a shift register, used to store the incoming data from the RXD port
 - an error bit controller.
 - When a read cycle is over, the shift register is ready to acquire the received data byte.
 - Because the data being transferred to the RXD port is coming in at a specific baud rate, a controller is needed to synchronize the data acquisition phase.
 - The serial data controller, which uses a state machine and the two synchronization counters, is used for this synchronization. The state machine is calibrated to read the RXD port in the middle of each bit transmission.
- 
- 

RECEIVER FSM

When idle, the serial data pin (RXD in this case) is held high, so the idle state remains unchanged until the RXD port goes low. Once the RXD is asserted low, the EightDelay state is entered. This state is used to assure that the RXD port is read in the middle of each bit transmission. The counter, ctr, increments at a rate that is 16 times faster than the baud rate. When the EightDelay state is entered, the ctr counter is allowed to count up to eight.

When this occurs, the WaitFor0 state is entered, followed by the WaitFor1 state. These two states rely on the most significant bit of the ctr counter for their next state logic. These states precede the GetData state, which shifts in the value of RXD at the time that the state is entered. The two wait states ensure that the state machine is delayed exactly long enough to read the RXD signal in the middle of its next transmission. The GetData state also increments the dataCtr counter, which is used to keep track of how many data bits have been shifted into the shift register. Once this counter is equal to 10 (8 data bits, 1 parity bit, and 1 stop bit), the CheckStop state is entered. This state enables the error bit controller. The idle state is loaded immediately after the CheckStop state.



CAMPIONAMENTO DEI BIT RICEVUTI

- The rShift signal enables the receiving shift register to shift in the value of RXD at the moment that rShift goes high. Notice how the rShift signal is asserted in the middle of each data bit on RXD.

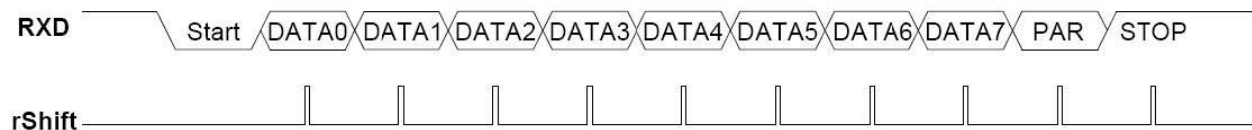
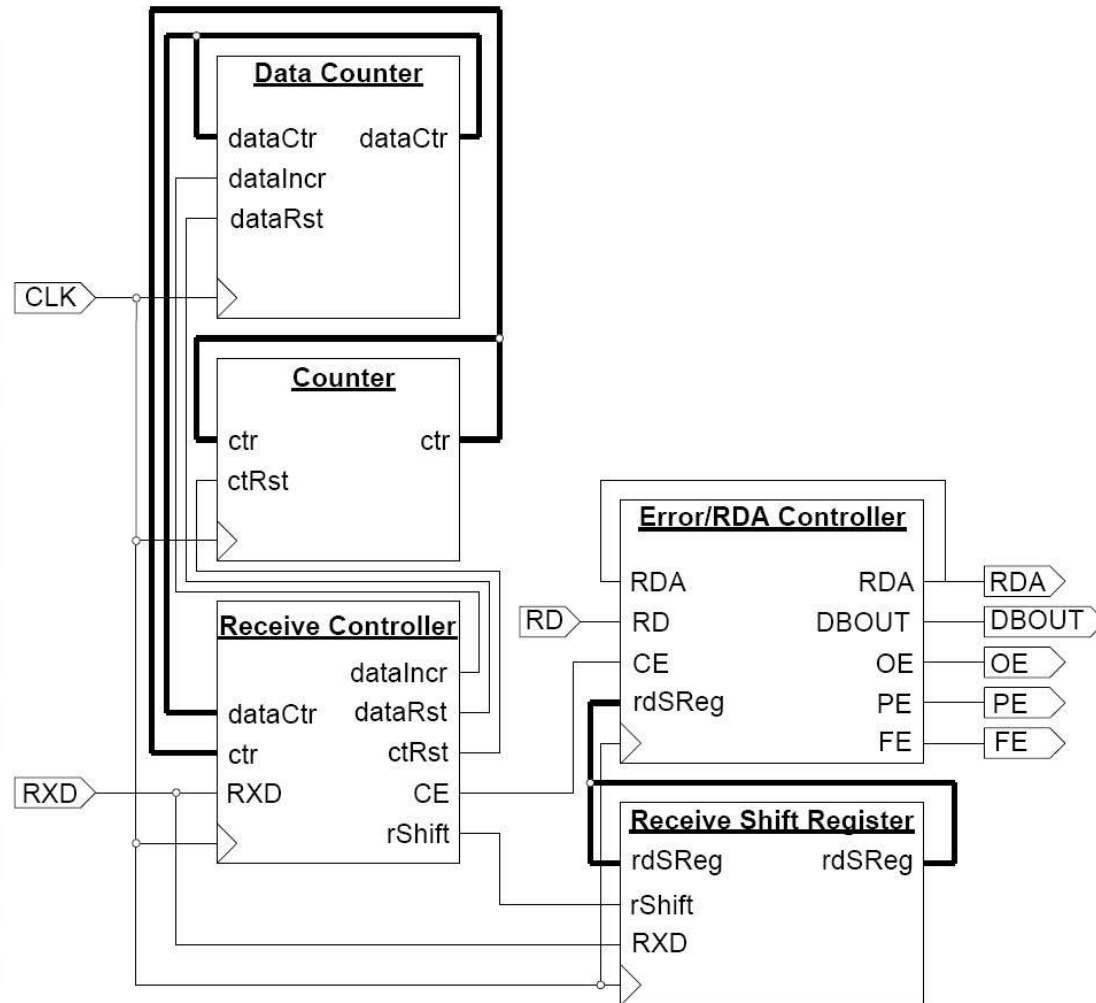


DIAGRAMMA A BLOCCHI DEL RICEVITORE



IMPLEMENTAZIONE DEL TRASMETTITORE

- The transmitter portion of the UART accepts a byte of data from the DBIN port and transmits it as serial data on the TXD port. The baud rate for the transmission is the same as the baud rate for the receiving portion of the UART
- To transmit the byte stored in the DBIN port, the transfer portion of the UART must contain a transfer controller, two counters for synchronization, and a transfer shift register.
- The transfer controller uses the two synchronization counters to control the rate at which the data byte must be transmitted across the TXD port, and the number of bits transmitted.
 - One counter is used to delay the transfer controller between transmissions,
 - while the other counter is used to keep track of how many transmissions have been sent out.
- The TXD port is set equal to the least significant bit of the transfer shift register, allowing data to be transmitted by simply shifting the register to the right once at the time of each transmission.

TRANSMITTER FSM

The transfer portion of the UART stays idle until the WR port goes high. When this happens, the UART is told to write whatever data is in the DBIN port, and the next state, Transfer, is loaded in the transfer state machine.

The Transfer state prepares the transfer shift register to transmit data. By setting load = '1', the shift register is loaded with a start bit, the data byte in DBIN, a proper parity bit, and a stop bit. The two reset signals, tClkRst and tDelayRst, are also set to '1' to reset the two synchronization counters. The next state loaded is the Shift state.

The Shift state sets the shift signal to '1' in order to shift the shift register right one. The tflncr signal is also set to '1' to increment the data counter. If the data counter is not equal to nine, all of the transfer shift register has not been shifted yet, and the Delay state is gone too. If all of the transfer shift register has been shifted, the WaitWrite state is loaded.

The Delay state is entered so that the transmission process sends out the data at the appropriate baud rate. Once the tDelayCtr is equal to the baudRate constant, the delay state is left, and the shift state is reloaded.

Once the WaitWrite state has been entered, the transmission is complete. This state is needed to make sure that the WR port that was held high to start the transfer process has been unasserted. Without this state, if the WR port is held high for too long, a garbage transmission will occur.

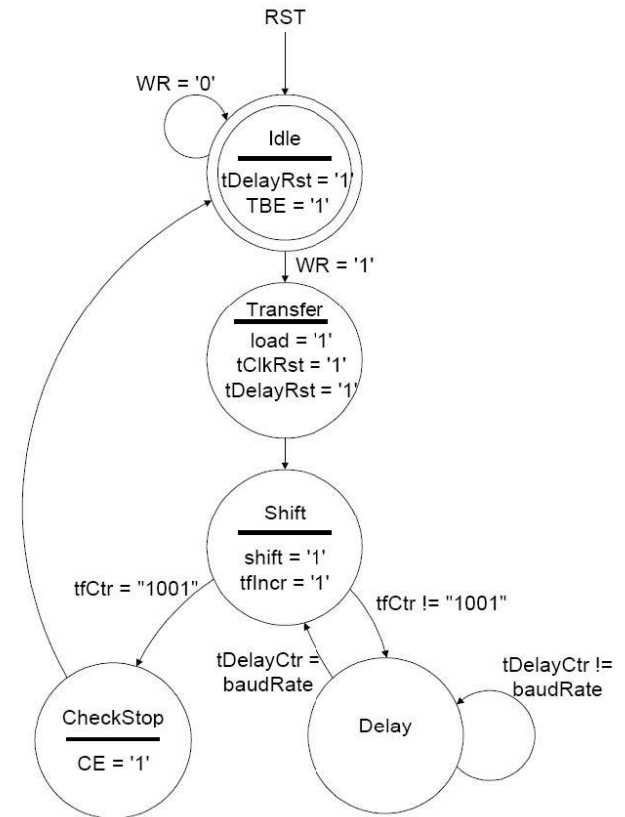
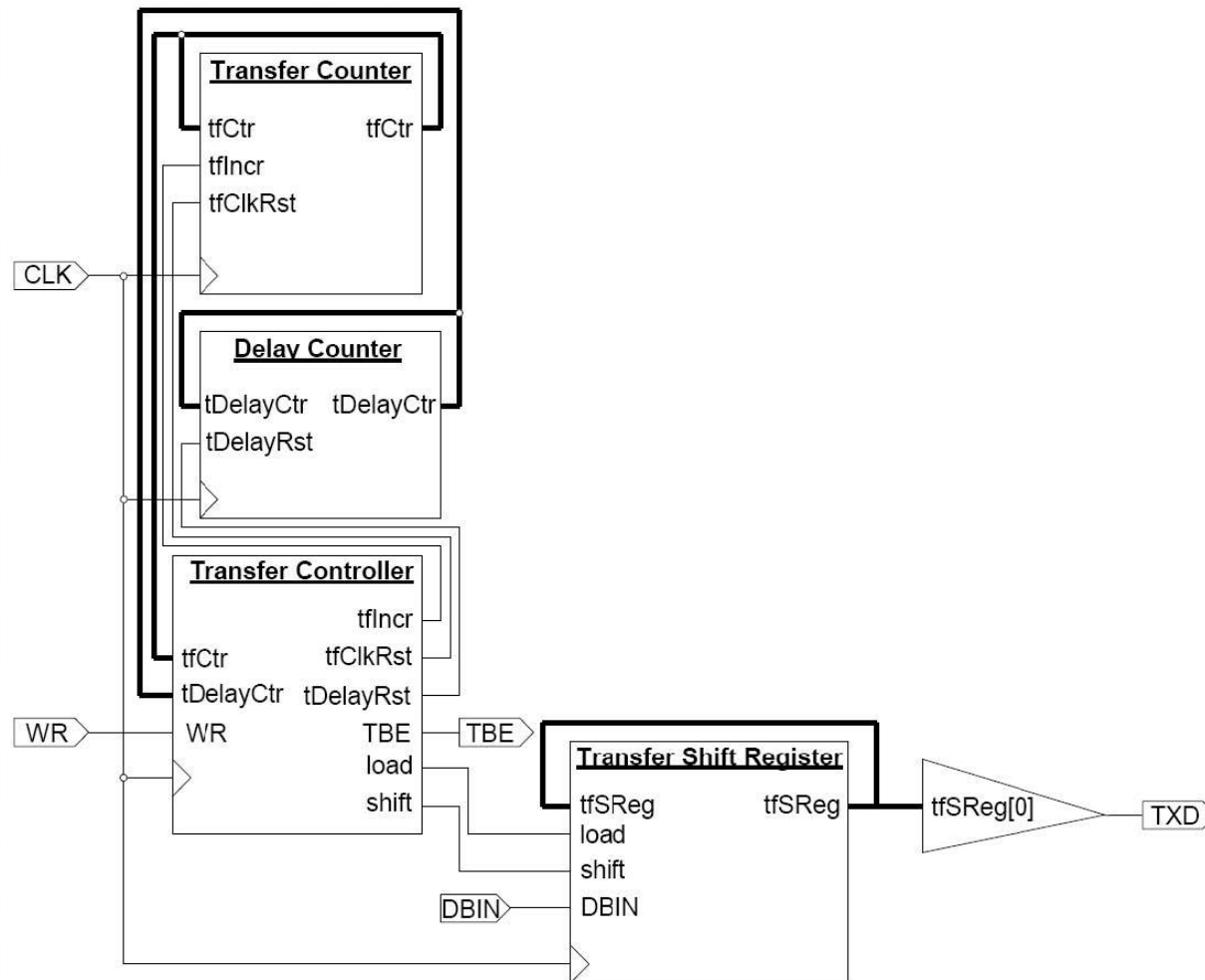


DIAGRAMMA A BLOCCHI DEL TRASMETTITORE



ESERCITAZIONE

- Sfruttando l'implementazione fornita dalla Digilent di un dispositivo UART (componente RS232RefComp.vhd), progettare ed implementare in VHDL i seguenti componenti:
 - a) UART_TAPPO: il componente acquisisce una stringa di 8 bit (fornita attraverso gli switch della board di sviluppo) e la serializza tramite la sezione di trasmissione del dispositivo UART; l'output seriale della UART viene re-inviato in ingresso alla sezione di ricezione dello stesso dispositivo (configurazione a tappo), e il dato deserializzato viene visualizzato sui led della board di sviluppo.
 - b) 2_UART: il componente acquisisce una stringa di 8 bit (fornita dall'utente tramite gli switch della board di sviluppo), la serializza tramite la sezione di trasmissione di un primo dispositivo UART, la deserializza tramite la sezione di ricezione di un secondo dispositivo UART collegato a valle del primo, e mostra le stringa led della board di sviluppo.

ESERCITAZIONE

- Sfruttando l'implementazione fornita dalla Digilent di un dispositivo UART (componente RS232RefComp.vhd), progettare ed implementare in VHDL i seguenti componenti:
 - c) UART_PC (facoltativo): il componente realizza la comunicazione fra la board di sviluppo e un terminale seriale in esecuzione su PC (es. *Termite*), previa connessione di PC e board tramite dispositivo fisico RS232 (uno degli endpoint di comunicazione è rappresentato dal PC). Il componente deve poter acquisire una stringa di 8 bit che rappresenta un carattere in codifica ASCII (fornita attraverso gli switch della board di sviluppo), ed inviarla tramite il dispositivo UART al terminale in esecuzione sul PC, in cui il carattere viene visualizzato. Allo stesso modo, il componente deve essere in grado di ricevere attraverso lo stesso dispositivo UART (oppure una seconda UART) un carattere trasmesso dal terminale e mostrarlo sui led.