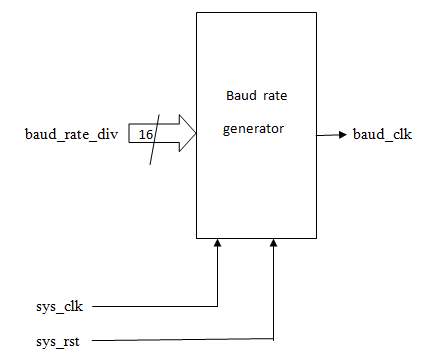
**BAUD RATE GENERATOR**

Generates the baud clock required by the UART transmitter and receiver module.This parameter specifies the desired baud rate of the UART. Most typical standard baud rates are: 300, 1200, 2400, 9600, 19200, etc. However, any baud rate can be used. This parameter affects both the receiver and the transmitter. The default is 2400(bauds).

****

**Fig 3.2 Schematic of the baud rate generator**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.NO** | **SIGNAL** | **TYPE** | **SIGNAL DESCRIPTION** |
| 1 | Sys\_clk | input | This is the system clock which must be equal to 16 times the baud rate |
| 2 | Sys\_rst | input | Asynchronous reset signal of the design |
| 3 | Baud\_rate\_div | input | signal by which the baud clock has to be divided |
| 4 | Baud\_clk | output | Frequency Divided clock signal |

**UART RECEIVER**

Since no common clock is shared, a known data transfer rate (baud rate) must be agreed upon *prior* to data transmission. That is, the receiving UART needs to know the transmitting UART’s baud rate (and conversely the transmitter needs to know the receiver’s baud rate, if any). In almost all cases the receiving and transmitting baud rates are the same. The transmitter shifts out the data starting with the LSB first. Once the baud rate has been established (prior to initial communication), both the transmitter and the receiver’s internal clock is set to the same frequency (though not the same phase). The receiver “synchronizes” its internal clock to that of the transmitter’s at the beginning of every data packet received. This allows the receiver to sample the data bit at the bit-cell center. A key concept in UART design is that UART’s internal clock runs at much faster rate than the baud rate. For example, the popular 16450 UART controller runs its internal clock at 16 times the baud rate. This allows the UART receiver to sample the incoming data with granularity of 1/16 the baud-rate period. This “oversampling” is critical since the receiver adds about 2 clock-ticks in the input data synchronizer uncertainty. The incoming data is not sampled directly by the receiver, but goes through a synchronizer which translates the clock domain from the transmitter’s to that of the receiver. Additionally, the greater the granularity, the receiver has greater immunity with the baud rate error. The receiver detects the start bit by detecting the transition from logic 1 to logic 0 (note that while the data line is idle, the logic level is high). In the case of 16450 UART, once the start-bit is detected, the next data bit’s “center” can be assured to be 24 ticks minus 2 (worse case synchronizer uncertainty) later. From then on, every next data bit center is 16 clock ticks later. Figure 2 illustrates this point. Once the start bit is detected, the subsequent data bits are assembled in a de-serializer. Error condition maybe generated if the parity/stop bits are incorrect or missing.

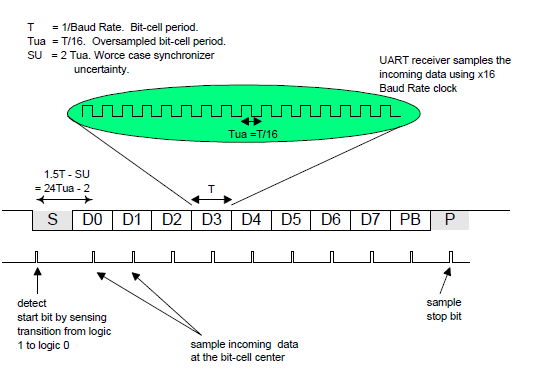
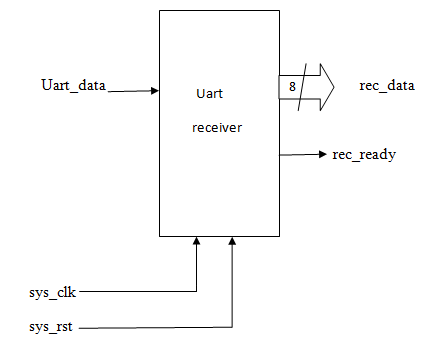
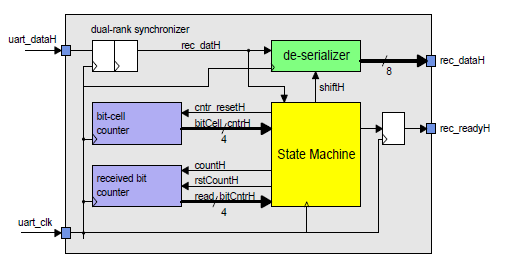


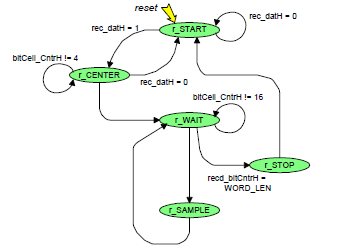
Fig 3.3: Data sampling points by the UART receiver

****

**Fig 3.4 Schematic of the UART receiver**

****

**Fig 3.5: UART Receiver Internal Block Diagram**

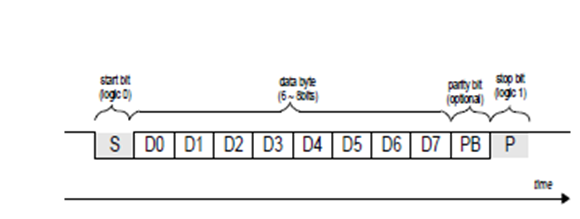
****

**Fig 3.6: Control state machine controlling the UART Receiver**

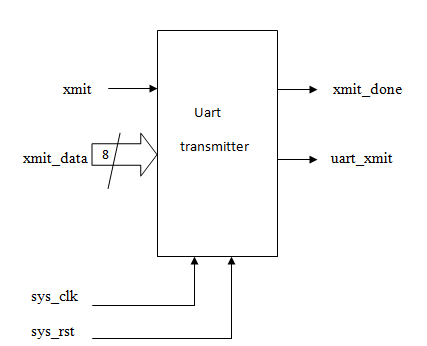
|  |  |  |  |
| --- | --- | --- | --- |
| **SL.NO** | **SIGNAL** | **TYPE** | **SIGNAL DESCRIPTION** |
| 1 | Sys\_clk | input | This is the system clock which must be equal to 16 times the baud rate |
| 2 | Sys\_rst | input | Asynchronous reset signal of the design |
| 3 | Uart\_data | input | This signal goes to the UART pin |
| 4 | Rec\_data | output | Parallely received data |
| 5 | Rec\_ready | output | This signal becoming high indicates that new set of data can be read |

**UART TRANSMITTER**

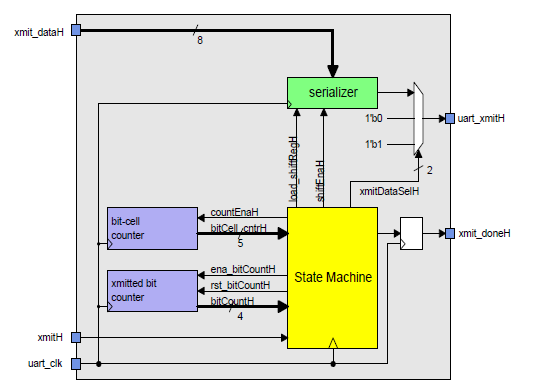
The UART transmitter, while no data is being transmitted, places a logic 1 in the XMIT line. A data packet is composed of 1 start bit, which is always a logic 0, followed by a programmable number of data bits (typically between 6 to 8), an optional parity bit, and a programmable number of stop bits (typically 1). The stop bit must always be logic 1. Most UART uses 8bits for data, no parity and 1 stop bit. Thus, it takes 10 bits to transmit a byte of data. In the UART protocol, the transmitter and the receiver do not share a clock signal. In our case the parity bit is not transmitted.



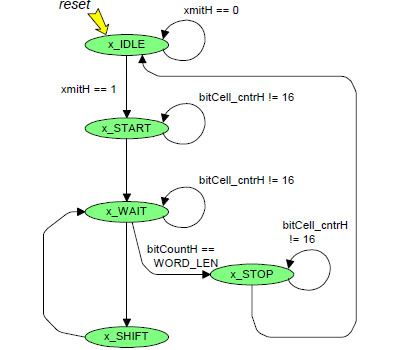
**Fig 3.7 Basic UART packet format: 1 start bit, 8data bits,1 parity bit, 1 stop bit.**

****

**Fig 3.8 Schematic of the UART transmitter**

****

**Fig 3.9: Blocks in the UART transmitter**

****

**Fig 3.10 States in the UART transmitter control state machine**

|  |  |  |  |
| --- | --- | --- | --- |
| SL.NO | SIGNAL | TYPE | SIGNAL DESCRIPTION |
| 1 | sys\_clk | Input | This is the system clock which must be equal to 16 times the baud rate |
| 2 | sys\_rst | Input | Asynchronous reset signal of the design |
| 3 | xmit\_data | Input | The data that needs to be transmitted serially |
| 4 | xmit\_done | output | Indicates wheather 8 bit of data is transmitted or not |
| 5 | uart\_xmit | output | This pin goes to the connector |