A Review on Implementation of UART with Optimum Noise Immunity

Pallavi R. Upase* and Prof. P.S. Choudhari**
* M.E.Student, Department of EEE,
Prof. Ram Meghe College of Engineering Administration and Management, Amravati, Maharashtra, India
p.anil.deshpande@gmail.com
** HOD, Department of EXTC, Prof. Ram Meghe College of Engineering Administration and Management, Amravati, Maharashtra, India
pschoudhari@gmail.com

Abstract: For asynchronous transmission of serial data between a DTE (data terminal equipment) and a DCE (data communication equipment) a UART (universal asynchronous receiver/transmitter) is used. It is a serial communication protocol widely used for short-distance, low speed, low-cost full duplex data exchange between computer and an output device. In working environments employing multiprocessor communication such as industries, the received data may be erroneous due to noise. The consequences may be hazardous due to improper control. The present day UART are designed with an error detection logic which requests retransmission of data frame from the transmitter if any error is detected. This task demands extra time. In the proposed paper we have reviewed the various approaches made so far for detecting and correcting errors, and improving the noise immunity of the system in order to obtain an optimized error free reception of data.

Keywords: UART (Universal Asynchronous Receiver/ Transmitter), BIST (Built in Self Test), FEC (Forward Error Correction), Hamming Codes, SEC-DED (Single error correction- Double error detection).

Introduction

The UART simultaneously perform transmit and receive functions as the transmitter and receiver have separate clock signals and control signals. Also they share a bidirectional data bus, which allow them to operate virtually independently of one another. The primary functions performed by a UART are [1]:

- Parallel to serial data conversion in the transmitter and serial to parallel data conversion in the receiver
- Error detection by inserting parity bits in the transmitter and checking parity bits in the receiver
- Insert start and stop bits in the transmitter and detect and remove start and stop bits in the receiver
- Formatting data in the transmitter and receiver
- Provide transmit and receive status on CPU
- Voltage level conversion between the DTE and the serial interface and vice-versa
- Provide a means of achieving bit and character synchronism.

During the data communication, transmission errors may occur due to electrical interference by nearby sources such as fluorescent lights, motors, generators, power lines etc or by some system discrepancies. In this paper we have presented an overview of different methods to control these errors proposed by various authors so far.

Data frame format of UART

UART has a transmitter, a line control register (LCR), a baud rate generator (BRG), and a receiver. The nature of the data is to be specified before transmitting the data in either direction through an UART. This is done by programming an eight-bit control word into the UART line control register (Fig.1.) at the transmitter end.

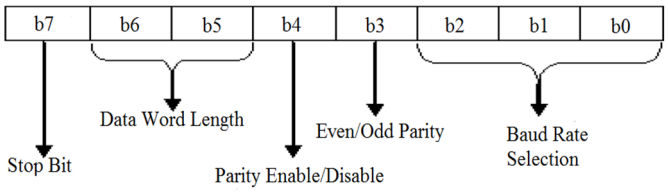

Fig.1. Line control register (control word register) bit description

Table 1 specifies exact logic levels of selection lines for selecting different baud rates, odd/even parity and data word length.

Table 1 : LCR Bit Description

BIT	Value			Description
0,1,2	b2	b1	b0	Baud rate selection
	0	0	0	57600
	0	0	1	38400
	0	1	0	19200
	0	1	1	9600
	1	0	0	4800
	1	0	1	2400
	1	1	0	1200
	1	1	1	600
3	0			Even Parity
	1			Odd Parity
4	0			Parity Disable
	1			Parity Enable
5,6	b6	b5		Data Word length
	0	0		5
	0	1		6
	1	0		7
	1	1		8
7	0			1 Stop bit
	1			2 Stop bits

An 11 bit frame format of UART can be as seen in Fig. 2. The signal line is assigned with two states- using logic high and logic low to distinguish a '1' and '0' respectively.

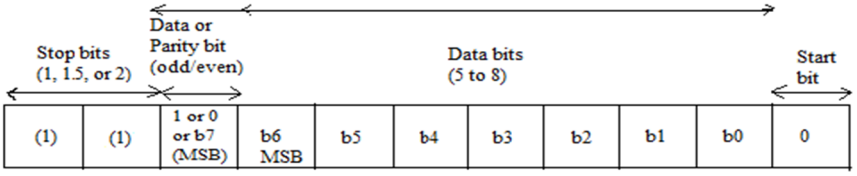

Fig. 2. UART data frame format [1]

UART frame format consist of a start bit, data bits, parity bit and stop bit. After the Start Bit, the Data Bits are sent, with the Least Significant Bit (LSB) sent first. The start bit is always low and the stop bit is always high. When the complete data word has been sent, it adds a Parity Bit .This parity bit may be used by the receiver to perform error checking. Then 1 or 1.5 or 2 Stop Bits is/are sent by the transmitter. Because asynchronous data are "self-synchronizing", if there is no data to transmit, the transmission line will be idle.

Sources of errors in UART

With asynchronous data, the transmit and receive clock may not be continuously synchronized. UART bit synchronization is achieved by establishing a timing reference at the centre of each start bit. Therefore, it is imperative that a UART detect the occurrence of a valid start bit early in the bit and establish a timing reference before it begins to accept data. The difference in time between when a sample is taken (i.e., when a data bit is clocked into the receive shift register) and the actual centre of data bit, is called the sampling error as shown in Fig. 3.

The difference in time between the beginning of the start bit and when it is detected is called the detection error. The higher the receive clock rate, the earlier a start bit would be detected. Because of the detection error, successive samples occur slightly off from the centre of the data bit. With asynchronous clocks, the magnitude of the sampling error for each successive sample would increase (the clock would slip over or slip under the data), eventually causing a data bit to be sampled twice or not sampled at all, depending on whether the receive clock is higher or lower than the transmit clock. [1] Hence the received data may be erroneous.

Errors can also occur if the transmission medium is not an ideal one. However internal defects, interference, channel noise, etc. keeps the transmission media far from ideal.

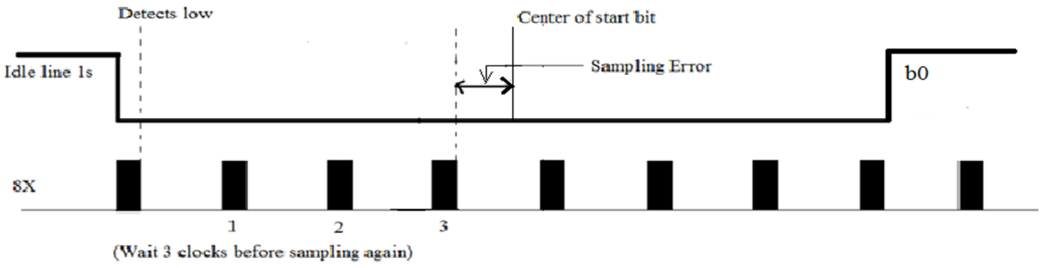

Fig.3. Sampling error

Data communication errors can be classified as single bit, multiple bits or burst. A single bit is when only one bit within the data string is erroneous. A multiple bit error occurs when two or more non-consecutive bits within the data string are in error. Whereas when two or more consecutive bits within the string are erroneous, it is a burst error. The length of burst errors is measured from the first corrupted bit to the last corrupted bit in the string. Some errors in between may not be corrupted ones. So far designers have developed two basic strategies for handling transmission errors; error detecting codes and error correcting codes. Parity bits, block and frame check characters and cyclic redundancy characters are examples of error detecting codes which include enough redundant information with each transmitted message to enable the receiver to determine when an error has occurred. Error correcting codes include sufficient extraneous information along with each message so that the receiver can determine in which bit the error has occurred and hence correct it.

Previous Work Done for Dealing with Errors

Mohd Yamani Idna Idris, Mashkuri Yaacob and Zaidi Razak [9] have implemented the BIST (Built IN Self Test) technique in the UART design which is a method of self testing on a system on chip. For testing sequential networks, we need to observe the state of flip-flops instead of the outputs only. The network output needs to be verified for each input combination and each state of flip-flop before going to the next correct state. This can be done by arranging the flip-flops to form a shift register. As seen in Fig.4, “Register A” and Register B” are configured by mode control bits B1 and B2. The state of the flip-flop is shifted out bit-by-bit using a single serial-output pin ’Q’ on the IC. This is called scan path testing.

A built-in-logic-block-analyser (BILBO) architecture is a scan register that can be modified to serve as a state register, a pattern generator, a signature register, or a shift register by setting the bits B1B2 of the BILBO scan register as shown in Table 2.

This technique checks if any logic errors present in the UART. Both transmitter as well as the receiver of the UART is checked by this method. The LFSR replaces the function of the external tester features such as a test pattern generator by automatically generating pseudo random patterns to give fault coverage to the UART module. The MISR acts as a compression tool, compressing the output result when automatic pseudo random pattern is fed to the UART. The shift register minimizes the input/output overhead by shifting the parallel signature produced by MISR into serial signature. Although time consuming (receiver has to wait for the signal from the transmitter) this method takes advantage of 100% fault coverage [10].

Table 2: BILBO Operating Modes

B1B2	Operating Mode
00	Shift Register
01	Linear feedback shift register(LFSR)/ Pseudo Random Pattern Generator (PRPG)
10	Normal
11	Multiple Input Signature Register (MISR)

Himanshu Patel, Sanjay Trivedi, R. Neelkanthan and V. R. Gujraty [3] have utilized recursive running sum filter to remove noisy samples from data samples at the receiver. Recursive Running Sum (RRS) is simple low pass filter. Input data signal is directly sampled with system clock and samples are accumulated over a window size. Serial receive data signal is directly sampled with system clock and samples are fed to RRS filter. The window size of the filter is user programmable and it decides baud rate. For correct data decoding, the selection of window size (down sampling factor-M) is important.

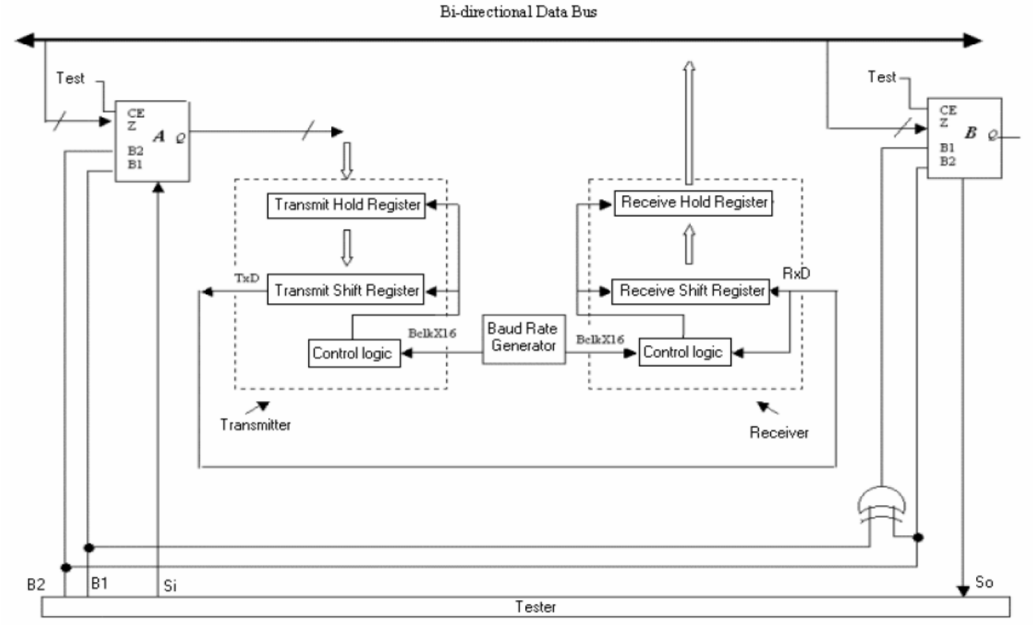

Fig.4. UART with BILBO register and Tester [9]

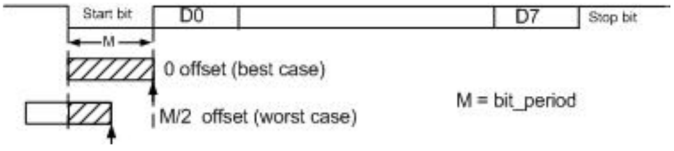

Fig 5.a Window offset when M = bit_period

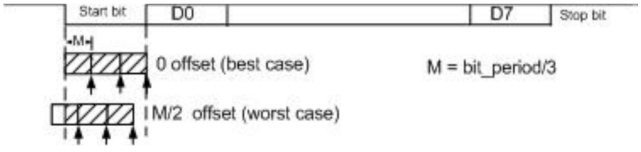

Fig 5.b Window offset when M = bit_period/3

For application of this filter in the asynchronous communication where data bits are transmitted asynchronously, the offset between window and start bit is important. When the window size (down sampling factor- M) is equal to bit period, different situations of offset between window and start bit is shown in the Fig-5.a. The window size is user programmable and it is set to one third of required bit period as shown in Fig.5.b. The baud rate equation is as follows.

$$Baud\_rate = \frac{clock\_freq}{M*3} \qquad (1)$$

Where, *clock_freq* is system clock frequency, M is window size, (1 to 2^{16}).
As the baud rate is decided by the window size so there is no need of any external "timer module" which is normally required for standard UARTs. The intermediate data bit is decoded using magnitude comparator as seen in Fig. 6.

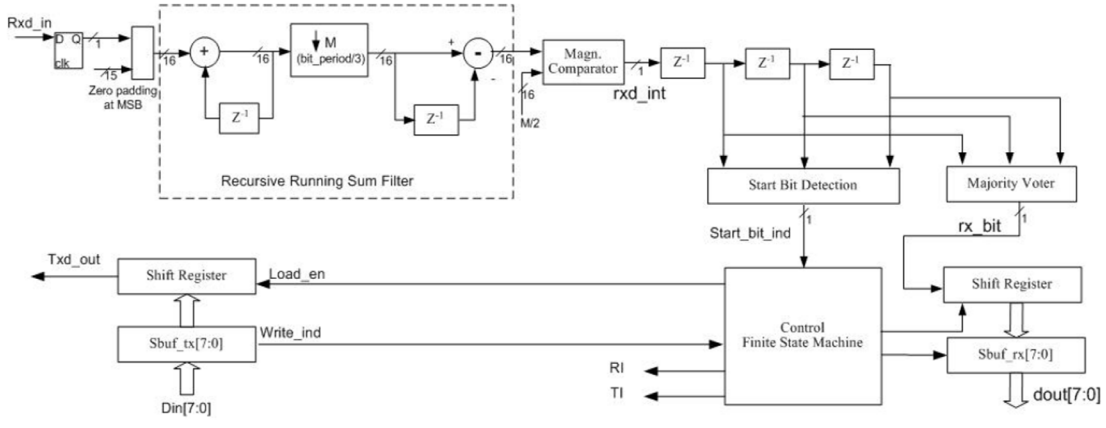

Fig.6. Robust UART architecture using running sum filter.[3]

A majority voter is used to decode actual data bit from three intermediate data bits. As three samples are available in a bit period, in the worst case too, two full windows are available in one bit period. Therefore, two out of three samples are correct. Hence the bit will be correctly decoded by the majority voter.
Comparison of simulation results at different noise level obtained by the authors, show that the robust UART described had far better performance than standard UART at higher noise levels as seen n Fig. 7. Percentage Ratio of correctly received bytes and total transmitted bytes is represented on the x-axis of graph. The y-axis of the graph indicates the percentage of average corrupted samples in a bit period. It should be noted that it is impossible to recover data bits if more than 50% of data samples in a bit period are corrupted by noise. The graph clearly indicate that the performance of standard UART deteriorates if more than 6% of data samples are corrupted in a bit period, while for the proposed robust UART the performance deteriorates only after 37%.

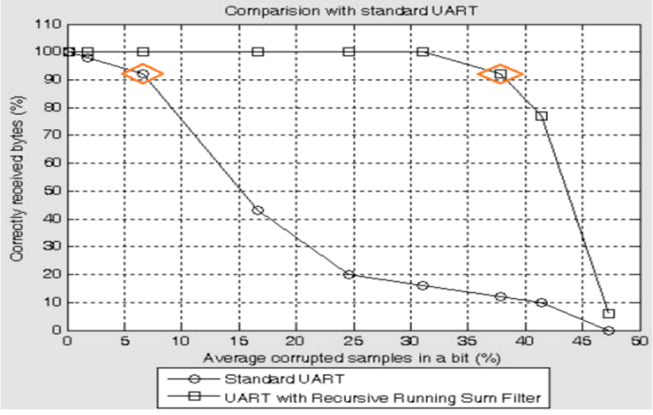

Fig. 7 Comparison with standard UART at different noise levels

Naresh Patel, Vatsalkumar Patel and Vikaskumar Patel [2] have proposed architecture of UART using the Status register which indicates parity error, framing error, overrun error and break error during the data reception.
In Fig.8, the error logic block handles four types of errors. PL bit will be set if the received parity does not match with the parity generated from data bits which indicates that parity error occurred. SL bit is set if frame error occurs when receiver fails to detect correct stop bit or when 4 samples do not match. OL bit is set when overrun error occurs if the receiver FIFO register is full and other data arrives at the RHR (receiver hold register). BL bit is set when there is a break in received data and break error occurs i.e. if the RXIN pin is held low for longer time than the frame time [2].

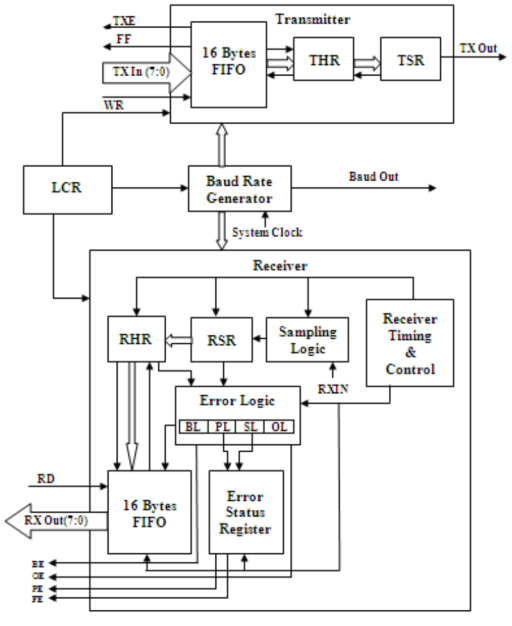

Fig.8. UART architecture with status register [2]

In the methods discussed so far, the major drawback of the logic used is, if an error is detected in received data, retransmission of corresponding data frames is required which take additional time for automatic repeat request (ARQ) and retransmission of data.

Sindhuaja Muppalla and Koteswara Rao Vaddempudi [5] have proposed a novel technique which improves the noise immunity of the UART system. They have implemented the extended hamming codes which have forward error correction (FEC) capability. These are also called Single Error Correction and Double Error Detection (SEC-DED) codes.

D3	D2	D1	D0	P0	P1	P2	P3

Fig. 9. a.-, Data format of (8,4) Hamming code

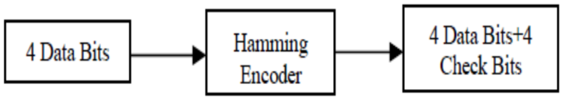

Fig. 9. b. Extended(8,4) Hamming code generation

In Fig.9.a- D0 to D3 are the data bits whereas P0 to P3 are the redundancy bits also called the Parity check bits formed by following equations [5]:

$$P3 = D0 \oplus D1 \oplus D2 \quad \ldots\ldots (2)$$
$$P2 = D0 \oplus D1 \oplus D3 \quad \ldots\ldots (3)$$
$$P1 = D0 \oplus D2 \oplus D3 \quad \ldots\ldots (4)$$
$$P0 = D1 \oplus D2 \oplus D3 \quad \ldots\ldots (5)$$

Hamming codes are used for error detection and correction. Hamming encoder (Fig. 9.b) generates the redundancy bits using a formula:

$$2^r \geq D+r+1 \quad \ldots\ldots (6)$$

In equation (6), r = number of redundancy bits and D = number of information data bits.

Redundancy bits are those extra bits which are required to detect and correct errors. The redundancy bits are added to the information bit at the transmitter and removed at the receiver. The receiver is able to detect the error and correct it because of these redundancy bits. [6]

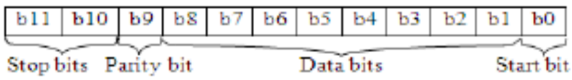

Fig. 10.a- Frame format for error detection mode

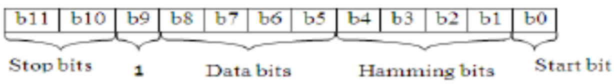

Fig. 10.b- Frame format for error correction mode

The frame formats used in this hamming code base architecture is as shown in Fig.10.a and Fig.10.b for the error detection and error correction modes respectively. The status register in the two modes are as shown in Fig.11.a and Fig.11.b. [6].

BE	PE	FE	OE

Fig.11.a. Error status register in error detection mode

BE	NE	SEC	DED	OE

Fig.11.b. Error status register in error correction mode

The different flags are set as follows:

BE : Break error – set when break in the received data string occurs.

PE: Parity Error- set when the received parity does not match the generated parity at the transmitter end.

FE: Frame error –set when the receiver fails to detect the correct stop bit

OE: Overrun error- set when data arrives at the receiver Hold register even though the FIFO register in receiver is full.

NE: No error- set when no errors occur (desired)

DED: Double error detected- set when two bit in the data string are erroneous

SEC- Single error corrected- set when only one erroneous bit is detected and consequently corrected by changing it.

At the receiver end, in the Hamming decoder, the syndrome is decoded to identify the error position if any, and the error is corrected if it is a single error and the DE flag is set if in case a double error is detected.

The decoding algorithm is as shown by following 4 equations:

$$S0 = D3 \oplus D2 \oplus D1 \oplus D0 \oplus P0 \oplus P1 \oplus P2 \oplus P3 \quad (7)$$
$$S1 = D3 \oplus D2 \oplus D1 \oplus P0 \quad (8)$$
$$S2 = D3 \oplus D2 \oplus D0 \oplus P1 \quad (9)$$
$$S3 = D3 \oplus D1 \oplus D0 \oplus P2 \quad (10)$$

All possible values of S0, S1, S2, S3 and their corresponding meanings are represented in Table 3.

Table3. Error Logic Register

S0	S1S2 S3	NE/SE/DE	Error Position	Flag set
0	000	No Error	-	NE
1	011	SE	D0	SEC
1	101	SE	D1	SEC
1	110	SE	D2	SEC
1	111	SE	D3	SEC
0	001	DE	Not predictable	DED
0	010	DE	Not predictable	DED
0	100	DE	Not predictable	DED
0	011	DE	Not predictable	DED
0	101	DE	Not predictable	DED
0	110	DE	Not predictable	DED
0	111	DE	Not predictable	DED

The test bench waveforms observed by the authors for the transmitter are as shown in Fig. 12.a . The waveforms for the RSR (receive shift register) and RHR (receive hold register) of the receiver part are given in Fig. 12.b and Fig.12.c.

Fig 12.a Test bench waveforms for transmitter

Fig. 12.b Test bench waveforms for Receive shift register

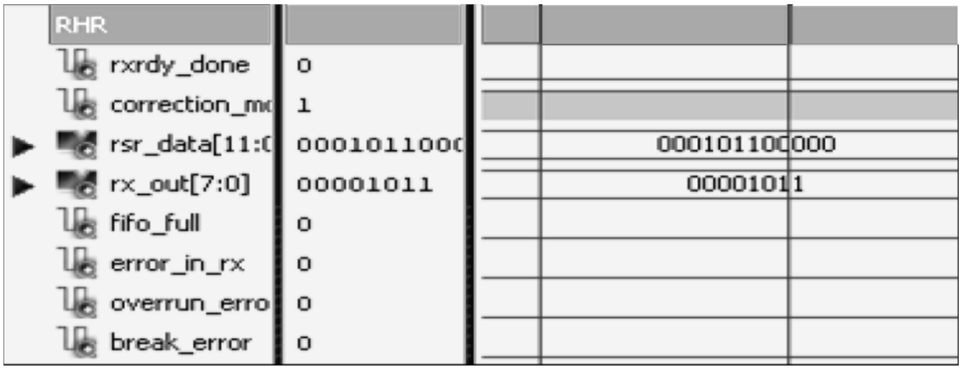

Fig 12.c Test bench waveforms for the receive hold register

As seen in Fig 12.a, only the 4 LSBs out of the 8 bit data "1011" (D3 to D0) are transmitted. The frame is formed starting with start bit '0', hamming bits "0100" (P0 to P3) and a '1' followed by two stop bits '11'. The RSR (receive shift register) received above data with an error in bit position D1. This error is corrected and the data output from RSR is given with all bits, except the data bits in the frame, replaced with zeros as "000101100000" shown in Fig12.b.The RHR (receive hold register) extracts the four data bits "1011" concatenated with "0000" in MSB position giving out a data byte as indicated in Fig.12.c.

Wilfried Elmenreich and Martin Delvai [7] have examined applicability of common UARTs in time-triggered systems and developed mathematical means to guarantee that a UART frame stays within its given time slot so that the communication is reliable.

A typical time-triggered protocol (TTP) defines a network cycle during which each node is given a dedicated time window during which it may broadcast a message. Each cycle begins with a reference message that contains the schedule for the upcoming cycle. Time-triggered messages are scheduled into exclusive windows, to ensure that there are no collisions. With no collisions occurring, the time-triggered approach is able to guarantee delivery time, and hence be deterministic. Today most microcontrollers already include a unit for serial transmission. TTP use a time triggered communication schedule to achieve predictable timing behavior and provide clock synchronization for imprecise cheap on-chip oscillators [11]. To achieve the intended goal of predictable communication, it must be ensured, that the UART communication of a microcontroller with an on-chip oscillator can hold the timing requirements imposed by the communication protocol, although UARTs were not originally designed for this application class.

The authors have studied the timing properties of a UART communication. The baud rate of a UART is usually configured by integer values - the arithmetic rounding error leads to baud rate deviations. The architecture of common UARTs furthermore leads to intrinsic delays at the sending UART. Moreover one has to regard the qualities of the clocks of the communication partners. Five timing deviations were observed [7]:

i. **Arithmetic error in baud rate setting**: If we want to set the UART to a given (ideal) baud rate, for every clock frequency *fclk* we can determine a UART Baud Rate Setting (UBRS) that best approximates the desired baud rate. Thus we can give an upper bound for the ratio between the fastest and the slowest approximated baud rate.

ii. **Send jitter problem**: At initialization of the UART, the baud rate generator is started. Running at a frequency corresponding to the configured baud rate, the baud rate generator periodically generates ticks which are possible start events of a message bit. When the UART receives a transmit signal, it starts the transmission at the next tick from the baud rate generator. Thus, depending on the internal state of the baud rate generator the transmission of a message may be delayed up to the interval time of two subsequent baud rate generator ticks. Usually the state of the baud rate generator cannot be read, leading to an indeterministic send delay jitter.

iii. **Clock drift**: The components performing a UART communication are set apart from each other and are clocked by different clock sources. To synchronize these clocks periodically, clock synchronization algorithms are used. Clocks can drift apart from each other after synchronization, depending on their drift rate. The drift of a physical clock is the ratio between the actual clock frequency *fclk* and the reference frequency *fref*. Good clocks have drifts that are very close to 1, so the drift rate ρ is introduced as in equation 11:

$$\rho = \frac{f_{clk}}{f_{ref}} - 1 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots .. (11)$$

A perfect clock will have a drift rate of 0. Real clocks have varying drift rates depending on temperature, voltage variations, or aging effects. The baud rate depends on the frequency of the UART clocking and is therefore influenced by the clock drift.

iv. **Clock offset**: This problem refers to the offset between the particular clocks of the communicating nodes. The offset of the local clock affects the instant when the communication partners send or expect to receive a message.

v. **Signal runtime**: Electric signals in a cable travel at approximately 2/3 rd of the speed of light. This results in a delay as given in equation 12:

$$t_{Signal} \approx \frac{l}{2 \times 10^8 \frac{m}{sec}} \quad \text{.....(12)}$$

Where, *l* is the length of cable between two transceivers.

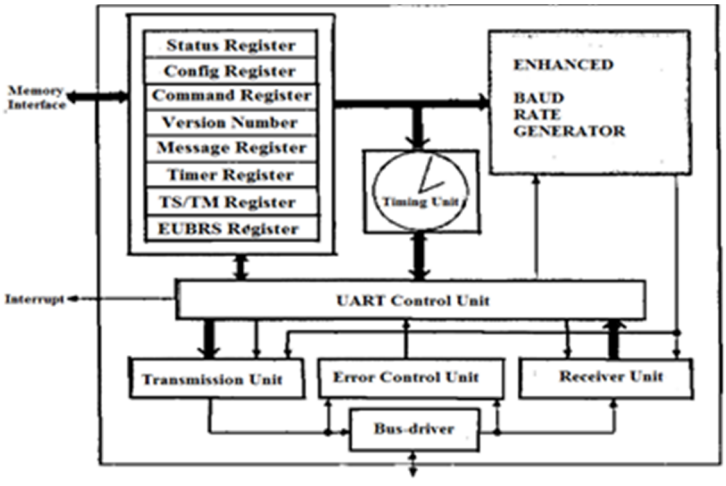

Fig. 13 Block diagram of UART I/O Module for Time Triggered Systems

When the send instant is a priori known to the sender and collisions between subsequent UART frames are obviated, the signal delay can be compensated by sending the time of minimal delay earlier. The condition for a successful and correct communication is given in Equation 13, where *n* is the number of bits in a frame including the start, stop and parity bits and BR is the Baud Rate:

$$\frac{BR_{fast}}{BR_{slow}} < \frac{n}{n - 1/2} \quad \text{..... (13)}$$

Fig.13 illustrates the block diagram of the UART I/O module proposed by Wilfried and Martin. This UART component overcomes the disadvantages of send jitter and arithmetic error in baud rate setting. It starts immediately after receiving the transmit signal with the message transmission. In this way the send jitter is completely eliminated.

Conclusion

We have analyzed that the received data may be erroneous due to three basic reasons:

(i) The noise induced by external nearby electric machines as dealt in [2] with the help of status register, in [3]with recursive running sum filter and in [5] using encoder and decoder;

(ii) The faults in the system on chip (gate level) as handled in [4] by implementing built-in self test technique; and

(iii) Timing problems due to imprecise on chip oscillators as tackled in [7] by adjusting the baud rate of transmitter and receiver.

We conclude that keeping the difference between the baud rate of the transmitter and receiver to minimum; keeping the receiver clock rate slightly higher than the transmitter; and, employing the extended Hamming code encoder and decoder at transmitter and the receiver end respectively can optimize error free reception of data, hence making the UART more immune to noise. Further rearranging the data bits as N columns and sending first bit of each followed by second bit of each, before implementing Hamming scheme may enable us to deal with burst errors; or modifying the Hamming matrices may facilitate to tackle adjacent errors, making the data more impervious to noise.

References

[1] Wayne Tomasi – "Electronic Communications systems - fundamentals through advanced"- fifth edition, Pearson Prentice Hall

[2] Naresh Patel, Vatsalkumar Patel, Vikaskumar Patel, "VHDL Implementation of UART with Status Register", 2012 IEEE International Conference on Communication Systems and Network Technologies, 2012.

[3] Himanshu Patel, Sanjay Trivedi, R. Neelkanthan, V. R. Gujraty , "A Robust UART Architecture Based on Recursive Running Sum Filter for Better Noise Performance", Held jointly with 6th International Conference on Embedded Systems, 20th International Conference on VLSI Design, pp. 819-823, Jan. 2007.

[4] Idris, M.Y.I, Yaacob M, "A VHDL implementation of BIST technique in UART design", 2003 Conference on Convergent technologies for Asia-Pacific Region (TENCON), pp. 1450-1454, 2003.

[5] Sindhuaja Muppalla, Koteswara Rao Vaddempudi," A Novel VHDL Implementation of UART with Single Error Correction and Double Error Detection Capability", SPACES-2015, Dept of ECE, KL UNIVERSITY

[6] Nutan Shep, Mrs. P.H. Bhagat," Implementation of Hamming code using VLSI", International Journal of Engineering Trends and Technology- Volume4Issue2- 2013

[7] Wilfried Elmenreich, Martin Delvai, "Time-Triggered Communication with UARTs", 4th IEEE International Workshop on Factory Communication Systems, Vasteris, Sweden, August 28-30, 2002

[8] Dipanjan Bhadra, Vikas S. Vij, Kenneth S. Stevens," A Low Power UART Design Based on Asynchronous Techniques", IEEE 978-1-4799-0066-4/2013

[9] Mohd Yamani Idna Idris, Mashkuri Yaacob, Zaidi Razak, "A VHDL Implementation of UART Design With BIST Capability", Malaysian Journal of Computer Science, Vol. 19 (1), 2006, pp 73-86

[10] Dr. T.V.S.P.Gupta,Y. Kumari, M. Ashok Kumar," UART realization with BIST architecture using VHDL " International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 3, Issue 1, January -February 2013, pp.636-640

[11] H. Kopetz, W. Elmenreich, and C. Mack, "A comparison of LIN and TTP/A", Proceedings of the 3rd IEEE International Workshop on Factory Communication Systems, Porto, Portugal, pages 99-107, September 2000.