International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064

A Robust UART Implementation for Industrial Applications on FPGA

Nagaraju. A[1], S. Nagi Reddy[2]

[1]M. Tech Student, Department of ECE, T.K.R College of Engineering, Meerpet, Andhra Pradesh, India

[2]Assistant Professor, Department of ECE, T.K.R. College of Engineering, Meerpet, Andhra Pradesh, India

Abstract: ***This paper describes a novel architecture based on Recursive Running Sum (RRS) filter implementation for wire and Wireless data processing. UARTs are used for asynchronous serial data communication between remote embedded systems. The universal asynchronous receiver/transmitter i.e. UART which is the kind of serial communication protocol which allows the full duplex communication in serial link. This paper presents the hardware implementation of a high speed and efficient UART using FPGA. If physical channel is noisy then, serial data bits get corrupted during transmission. The UART core described here, utilizes recursive running sum filter to remove noisy samples. Input data signal is directly sampled with system clock and samples are accumulated over a window size. The window size is user programmable and it should be set to one tenth of required bit period. The intermediate data bit is decoded using magnitude comparator. The advantage of this architecture is that 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 Recursive Running Sum (RRS) filter architecture with programmable window size of M is designed and modules are implemented with VHDL language. This project implementation includes many applications in wireless data communication Systems like RF, Blue tooth, WIFI, ZigBee wireless sensor applications. Total coding written in VHDL language. Simulation in Modelsim Simulator, Synthesis done by XILINX ISE 9.2i. Synthesis result is verified by the Chipscope. Input signal given from the keyboard and output is seen by the help of HyperTerminal.***

Keywords: Robust UART, RRS filer, data processing, communication, noise.

1. Introduction

Universal Asynchronous Receiver Transmitter (UART) is a kind of serial communication protocol [1], [2], [3], [4]. Mostly used for short-distance, low speed, low-cost data exchange between computer and peripherals. UARTs are used for asynchronous serial data communication by converting data from parallel to serial at transmitter with some extra overhead bits using shift register and vice versa at receiver. It transmit 9600 to 38400 bps for transmitting data bit. But if the physical channel is noisy then data bits get corrupted during transmission and it leads to wrong data decoding at receiver.

To overcome the noise problem a digital low pass filter based architecture is proposed in this paper. Recursive Running Sum (RRS) is simple low pass filter [5]; it can be used to remove noise samples from data samples at receiver. 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. RRS filter hardware design and implementation is described in section-2, Window size selection criteria are described in section-3, The UART Architecture and block diagram described in section-4, while section-5 gives simulation results and comparison with standard UART core. The robust UART core described here is designed using VHDL and implemented on Xilinx Vertex FP.

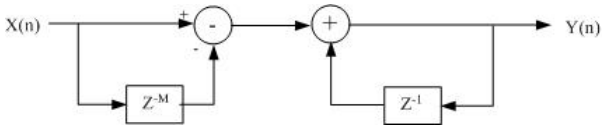
2. RRS Filter Design

To decode the correct data signal from corrupted noisy signal at the UART receiver input, we use digital low pass filter based architecture. The architecture name called recursive running sum filter, is a simple low pass filter it can be used to remove noise samples from data samples at receiver. The Recursive Running Sum (RRS) filter with window size of M is described by following equations.

$$H(z) = \frac{1-z^{-M}}{1-z^{-1}}$$

$$y(n) = x(n) + y(n-1) - x(n-M)$$

The hardware realization of the above equation is as shown in the Figure -1. It requires an Adder, subtracted, a unit delay and an M samples delay element. The window size (M) is related to baud rate which is user programmable. So M is variable, if a 16 bit register is used to hold value of M, it can have values from 0 to 65535. The hardware implementation of variable delay with above range would require 65535 D flip-flops and large number of combinatorial logic for MUX and selection logic implementation. So this implementation is not feasible for FPGA or ASIC platform.



A. Figure 1: Hardware realization of RRS filter

The Other approach for hardware realization of RRS filter using a factor of M decimator is shown in Figure -2. It requires an Adder, subtracter, two unit delays and a down sampler of factor M. The implementation of down Sampler requires a 16 bit counter and a magnitude comparator which is much simpler than previous approach. So this approach is selected for robust UART implementation.

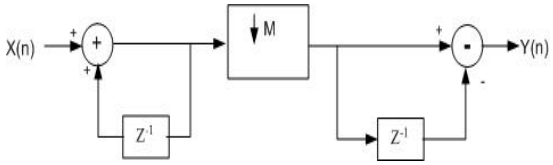


Figure 2: Implementation of RRS using down sampler

3. Window Size Determination

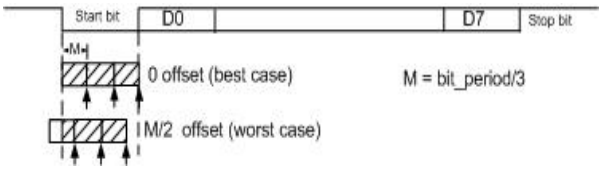
The selection of window size (down sampling factor-M) is important for correct data decoding. The input data signal (rxd_in) is sampled at system clock and fed to RRS filter as input x (n), output data samples y (n) are available at the interval of M samples because of the down sampler. For application of the above 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 scenarios of offset between window and start bit is shown in the figure 3.

Figure 3: window size determination

The offset between window and start bit can vary from 0 to M/2. In the worst case offset of M/2, half of the samples in current window are from previous bit and remaining half samples are from current bit, so this will lead to wrong decoding of the data bits.

To overcome above problem the window size should be chosen smaller than bit period so that multiple windows are available in one bit period. Odd number of windows should be chosen in one bit period so that a majority voter can be used to decode a bit from the odd number of the samples

.



.

Figure 4: Window offset when M = bit_period/3

For this case, different scenarios of offset between window and start bit is shown in the figure -4. So, three samples are available in a bit period. In the worst case also two full windows are available in one bit period so two Out of three samples are correct, so bit will be correctly decoded by the majority vote. Thus window size equal to one third of bit period is optimum selection.

4. UART Architecture

The architecture of UART receiver is shown in the figure, the incoming rxd_in signal is sampled at system clock.

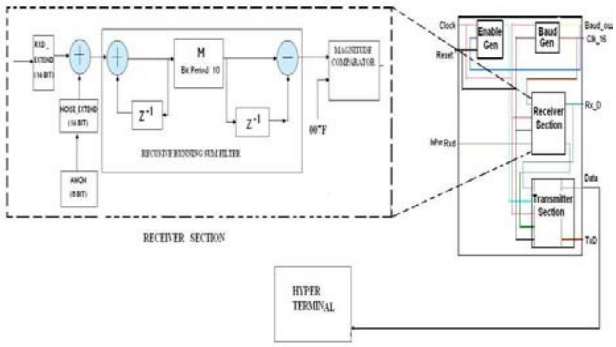


Figure 4: The UART block diagram

The 1 bit input given to rxd_extended, based on the input if it is '1' then the value of output is converted to +127 in 16 bit signed magnitude format else -127 16 bit signed magnitude format. Next stage is RSS filter; the two basic building blocks of a RSS filter are an integrator and a comb. An integrator is simply a single-pole IIR filter with a unity feedback coefficient Y [n] =Y [n-1] + X [n]. This system is also known as an accumulator. The transfer functions for an Accumulator. The transfer function for an integrator on the z- plane is H_I (Z) =1 / 1-Z $^{-1.}$ The power response is basically a low-pass filter with a 20 dB per decade (- 6 dB per octave) roll off, but with infinite gain at DC. This is due to the single pole at z = 1; the output can grow without bound for a bounded input. In other words, a single integrator by itself is unstable.

After that passed through a decimator block i.e. The digital clock manager (Decimator) module implemented in our project divides input clk signal by 10 times. The input signal x is the input data of 16 bit and y is the output of the given data input. Then given to a comb filter which is running at the low sampling rate, f_s / R, for a rate change of R is an odd-symmetric FIR filter described by Y [n] = X [n] -- X [n-RM] In this equation, M is a design parameter and is called the differential delay. M can be any positive integer, but it is usually limited to 1 or 2. The corresponding transfer at f_s / R=> H_C (Z) =1 / 1-Z $^{-RM.}$

When we build a RRS filter, we cascade, or chain output to input, N integrator sections together with N comb sections. This filter would be fine, but we can simplify it by combining it with the rate changer. Then given to magnitude comparator, this compares the input value with a reference level and gives the output to the UART module.

5. Results

The UART core described here has been designed using VHDL. Noise insertion is done by additive White Gaussian Noise. The core has been simulated using ModelSim Simulator while hardware implementation is done on Xilinx Spartan 3e FPGA and has been checked hardware with On

Chip Debugging tool Chip Scope pro with the help of icon and ila component

5.1 Simulation Results

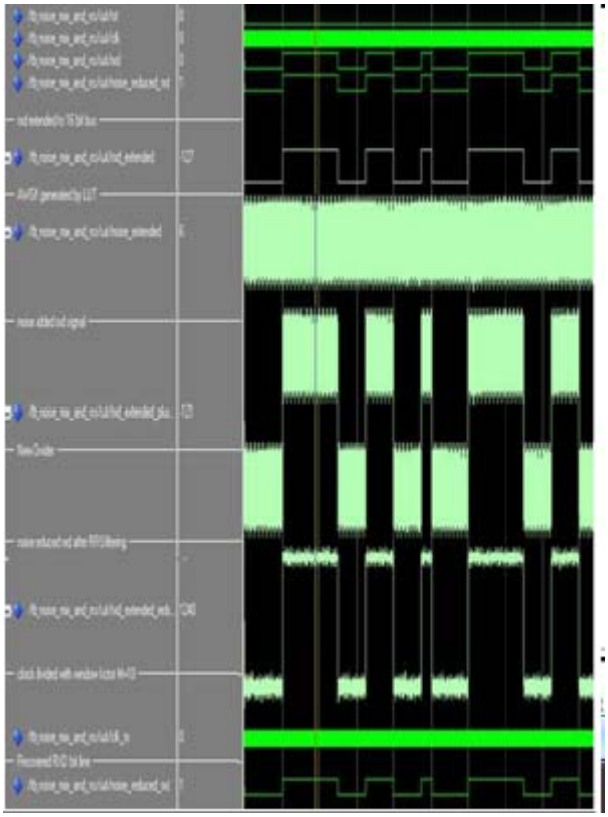


Figure 5: op Module Simulation Wave Form

5.2 RTL Schematic

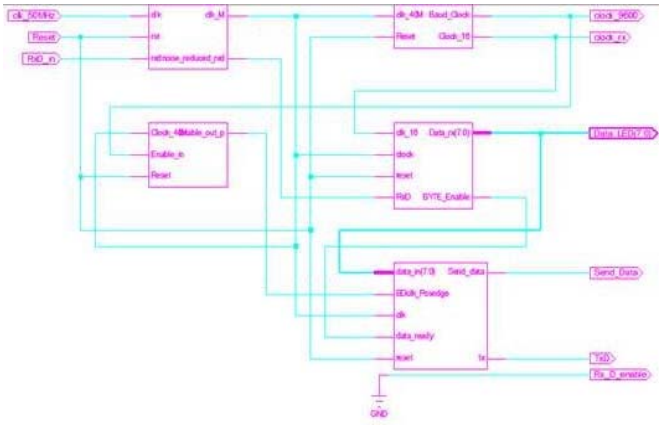
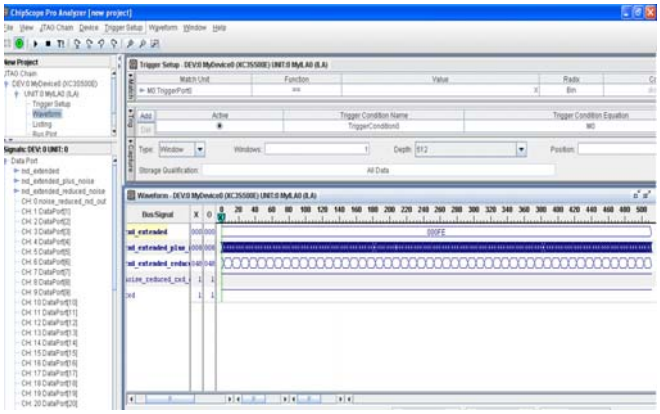


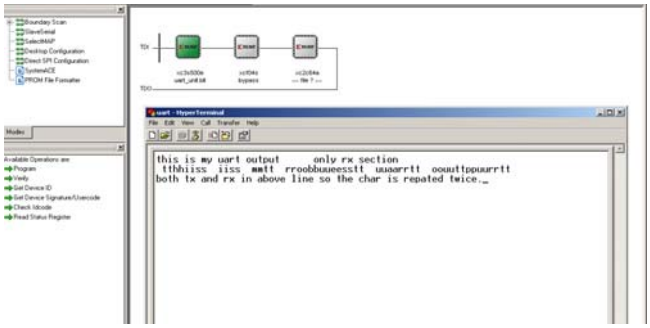
Figure 6: Top Module RTL schematic diagram

5.3 Chipscope Result

Synthesis result is verified by the Chipscope. Input signal given from the keyboard and output is seen by the help of HyperTerminal.



5.4 Out Put for UART



6. FPGA Implementation

The robust UART described in this paper has been implemented on Xilinx® Spartan XC3S500E-5FG320. The synthesis report is as shown below, it consumes only 655 Spartan slices, thus UART core occupies very small area.

Selected Device: 3S500E-5
Number of Slices: 655 of 4656 14%
Number of Slice Flip Flops: 458 out of 9312 4%
Number of 4 input LUTs: 773 out of 9312 8%
Number of bonded IOBs: 16 out of 232 6%
Number of GCLKs: 4 out of 24 16%

7. Conclusion

This paper describes a robust UART architecture based on recursive running sum filter for removal of channel induced noise. The baud rate of serial communication is decided by the window size of the filter, which is user programmable and should be set to one tenth of required bit period. Thus it does not require any external module for baud rate generation. Comparison of simulation results shows that the performance of standard UART deteriorates if more than 6% of data samples are corrupted in a bit period while for robust UART the performance deteriorates only after 37% thus the robust UART has far better performance than standard UART at higher noise levels.

8. Future Scope

In existing system we have one baud rate (i.e. 9600) and One data width. Variable width data and variable baud rate Robust UART can be implemented in future Implementation models.

References

[1] Dr. Garima Bandhawarkar Wakhle, Iti Aggarwal And Shweta Gaba "Synthesis and Implementation Of UART using VHDL Codes" 2012 International Symposium on Computer, Consumer and Control
[2] Himanshu Patel, Sanjay Trivedi, R. Neelkanthan, V. R. Gujraty "robust uart architecture based on Recursive running sum filter for better noise Performance"
[3] Intel® MCS-51 microcontroller family user's Manual
[4] Synopsys DesignWare® DW8051 MacroCell Databook
[5] Liakot Ali, Roslina Sidek, Ishak Aris, Alauddin Mohd. Ali and Bambang Sunaryo Suparjo "Design of a micro-UART for SoC application" Computers & ElectricalEngineering Journal, Elsevier Publ. Volume 30, Issue 4 , June 2004, Pages 257-268
[6] Norhuzaimin, J. Maimun, H.H. "The design of high Speed UART" Asia-Pacific Conference on Applied electromagnetics (APACE 2005). Dec. 2005
[7] Yong Lian Poh Choo Ho "ECG noise reduction Using multiplier-free FIR digital filters" 7th International Conference on Signal Processing (ICSP '04.) 2004. Volume: pages: 2198 - 2201
[8] M. Delvai, U. Eisenmann, W. Elmenreich, "Intelligent UART Module for Real-Time Applications" First Workshop on Intelligent Solutions in Embedded Systems (WISES), pages 177-185, Vienna, Austria, June 2002
[9] P.J. Ashenden, "The Designer's guide to VHDL", Morgan Kaufmann Publ., 2000[10] D. Ravi1, K. S. R Murthy2." Noise Immune and Area Optimized Serial Interface for FPGA based Industrial Interfaces"
[10] Mitra, S. K., "Digital Signal Processing – A Computer based approach," Tata McGraw-Hill, 2001
[11] Xilinx® Virtex FPGA datasheets
[12] Spartan-3 FPGA Family: Complete Data Sheet