l of Nonlinear Analysis and Optimization
0(2) (2019), February 2019
ɔs://ph03.tci-thaijjo.org/

1906-9685

J.Nonlinear Anal. Optim

UART-SPI INTERFACE WITH ERROR DETECTION AND CORRECTION

V.Pratapa Rao
Associate Professor, Department of EEE
Sri Sai Institute of Technology and Science, Rayachoti
Email: prataprao@gmail.com

Abstract- This paper proposes a UART-SPI controller with error correction and detection that can be implemented in SOC's (System On chip). This UART-SPI interface can be used for communication between multiple SPI slave devices from a UART. The frame error is detected and corrected using hamming code. The complete project is simulated and synthesized using Xilinx13.2.

Key Words:- Universal Asynchronous Receiver or Transmitter (UART), Serial Peripheral Interface (SPI), Hamming code, Check bits

I INTRODUCTION

An UART is a device allowing the transmission and reception of information, in a serial and asynchronous way. It is used for asynchronous serial data communication between remote embedded systems. The UART can be used to control the process of breaking parallel data from PC down into serial data that can be transmitted. It consists of receiver module and transmitter module and baud rate generator module. UART has been an important input/output tool for decades and still widely used. UART's are used for communication between two devices.

SPI is a synchronous protocol that allows master device to initiate communication with slave devices. It is a full duplex, serial bus commonly used because of its simple hardware interface requirements and protocol flexibility. SPI consists of two blocks. The SPI master and the SPI slave, the SPI Master which is being used in this design implements the master functionality of the SPI protocol. The master block generates the control signals to interface to external devices.

The main advantage is, the UART- SPI fit in any application where an SPI device has to be used. As the UART-SPI interface can be used to communicate to SPI slave devices from a PC with UART port it can be used for typical applications like interfacing of EEPROM, flash memories and sensors.

The paper is organized as follows. Section 2 describes working of UART. Section 3 describes implementation of SPI. Section 4 describes how we can detect and correct errors, Section 5 explains about the implementation of interface between UART and SPI; Section 6 shows the simulation results, RTL Schematics and conclusion.

II UART IMPLEMENTATION

The UART is Asynchronous Receiver and Transmitter, in which hardware is available for the translation of parallel data in serial form. The UART is commonly used with other serial communication standards such as RS232, RS485.

In asynchronous transmission, data to be transmitted without synchronization of the clock pulse between transmitter and receiver. Instead, the time parameters of the sender and receiver are matched in advance and special bits are added to each work. When a word (8 bits) of data is given to the UART, a start bit is added to the beginning of each word that is to be transmitted. The start bit is used to inform receiver that a word is sent and thus the clock in receiver is synchronized with the transmitter. After the start bit, the LSB is sent first and the individual bits of the word are sent. Irrespective of the data received is correct or not, the UART automatically discards the start, stop and parity bits. If the data is sent continuously the start bit of new word can be sent as soon as the stop bit of previous word is transmitted. As asynchronous data is self synchronous, if no data is available for transmission, the line can be idle.

Baud rate generator is used to provide the reference time of sending and receiving data for transmitter module and receiver module. The serial data is in the form of a stream of '1's and '0's. Only repetitive measurements of the smallest occurring interval

between data transitions can establish the incoming data baud rate. The method described here uses digital counters, edge detectors, comparators, and register storage of the time interval data from which the baud rate is easily computed. The fig. 2.1 shows 'n' represents falling edge while 'p' stands for rising edge followed by 'n'. 't1' and 't2' are the time intervals between two adjacent edges. The time interval is equal to one or several system clock cycles A stream of serial data input is passed to the edge detectors; the edge detectors begin to detect 'n' and 'p'. Counters start to sample counting at falling edge and rising edge, respectively, and reach at the values of 't1' and 't2', which are stored in a holding register respectively. Then continuously repeat sampling count, and compare with the value of holding registers, eventually reach at their minimum values 'min1' and 'min2'. The sum of 'min1' and 'min2' divided by 2 is the number of system clock cycle (MIN) occupied by one of RXD data. Because transmitter and receiver modules are based on 16 times of the baud rate clock to work, MIN divided by 16 is the frequency divisor (DIV). After obtaining DIV, it only needs a sample counter to implement DIV time's frequency division on system clock, and generate corresponding clock signal.

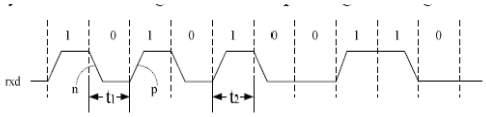

Fig. 2.1 Serial data wave form from input

III. SPI IMPLEMENTATION

SPI stands for Serial Peripheral Interface.SPI is asynchronous protocol that allows a master device to initiate communication with a slave device. Data is exchanged between these devices. SPI is implemented by a hardware module called the Synchronous Serial Port or the Master Synchronous Serial Port. This module is built into many different micro devices. It allows serial communication between two or more devices at a high speed and is reasonably easy to implement.

It can support up to speeds of 10 Mbps of data rate. The single master and multi master protocols are also implemented in SPI. The peripherals can be a RTC, ADC, DAC, EEPROM, FLASH etc. In addition to setting clock frequency, the master must also configure the CPOL and CPOH.

The SPI bus is a 4 wire serial communications interface used for micro controllers and processors for peripherals to communicate with each other. This SPI protocol can also be used for communication between two processors.

The 4 signals used in SPI protocol are:

1. MOSI – Master out Slave In is a output to master controller

2. MISO – Master in Slave Out is an input to master controller.

3. SCLK or SCK - Serial Clock. Used for synchronization between master and the slave.

4. SS - Slave Select. SS signal is used to select the slave device and it is an active low signal.

A full duplex data transmission can occur during each clock cycle. That means the master sends a bit on the MOSI line; the slave reads it from that same line and the slave sends a bit on the MISO line, the master reads it from that same line. Data transfer is organized by using Shift register with some given word size such as 8- bits (remember it's not limited to 8-bits), in both master and slave. They are connected in a ring. While master shifts register value out through MOSI line, the slave shifts data in to its shift register.

When CPOL =0, and CPHA =0 the data is captured for rising edge of clock and data is propagated for the falling edge of clock. When CPOL=0 and CPHA=1 the data is captured for falling edge of clock and propagated for the rising edge of clock.

When CPOL =1 and CPHA=0 the data is captured during the falling edge of clock and propagated during rising edge of clock. When CPOL=1 and CPHA=1 the data is captured during the rising edge of clock and during falling edge of the data is propagated.

IV. ERROR DETECTION AND CORRECTION

Messages that are transmitted over a transmission medium can be damaged; bits can be masked or inverted by noise. Hence detection and correction of these errors are important. One of the widely used techniques is Hamming code. It can be used to detect error and to correct single bit error. In this technique the check bits are appended to the data bits then transmits.

In the Hamming code, *k C*heck bits (parity bits) are added to an *n*-bit data word, forming a new word of "*n +k"* bits. The bit positions are numbered in sequence from "1 to *n+ k"* [5].

Hamming Code that appends four check bits C0, C1, C2 and C3 to an eight-bit value with data bits 'd0 . . . d7' as shown if fig 4.1.

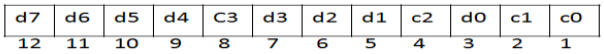

Fig. 4.1 Data appended with check bits

Number the position of bits starting form 1 and write all the bit numbers in binary. Check bits C0, C1, C2 and C3 are placed for all bit positions in Powers of two *i.e.* 2^0, 2^1, 2^2 and 2^3 respectively. In binary representation of all bit positions, C0 covers all bit position which have the LSB set (like 1, 3, 5,etc), C1 covers bits positions of 2nd LSB set (like 2, 3, 6, etc), C2 covers 3rd LSB set (like 4, 5, 6 etc) and C3 covers 4th LSB set (like 8, 9, etc). The check bits are calculated by XORing the data bits presented in covered bit positions as stated above are show in eq. 4.1.

The check bits are calculated as

C0= d0 ^ d1 ^ d3 ^ d4 ^ d6

C1= d0 ^ d2 ^ d3 ^ d5 ^ d6

C2= d1 ^ d2 ^ d3 ^ d7

C3= d4 ^ d5 ^ d6 ^ d7 ------- Eq. 4.1

The generated 12-bit code word is splitted as dataword1 and data word 2. Data word 1 consists of LSB of 8-bits; remaining 4-bits of codeword are appended with four 0's, considered as data word 2 as shown in fig. 4.2.

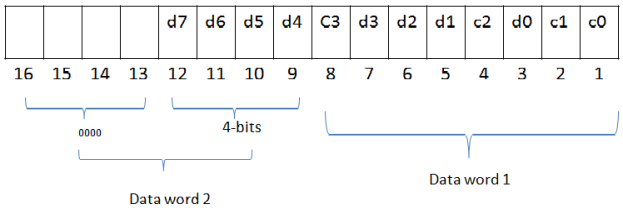

Fig. 4.2. Generating Datawords

Again check bits are calculated for received datawords and compared it with the check bits of transmitted code word. After comparison, the result is

C = C4C3C2C1 ------ Eq. 4.2

If *C* is equalts to '0' then there is no error in received data, when C is not equalts to '0', the decimal value of '(*C* –1)' gives the position of the bit in error. The error can then be corrected by complementing the corresponding bit. Note that an error can occur in the data or in one of the check bits.[5]

V. UART-SPI INTERFACE

Initially the data to be transmitted is given to code word generator, which provides the data with parity bits or check bits used for detecting the errors that may occur during the transmission of the data. The code word generator adds 4-bits in addition to the 8-bits of data; these 12-bits code word is divided into two data words and transmitted. The first 8-bits are grouped as a frame and the remaining 4 bits are appended with four zeros and form a frame. These frames are then given for transmission, in which one start bit (0), parity and a stop (1) bits are added. Once the data is received in the receiver, the start bit, parity and stop bits are discarded and actual data is recovered from the transmitted frame

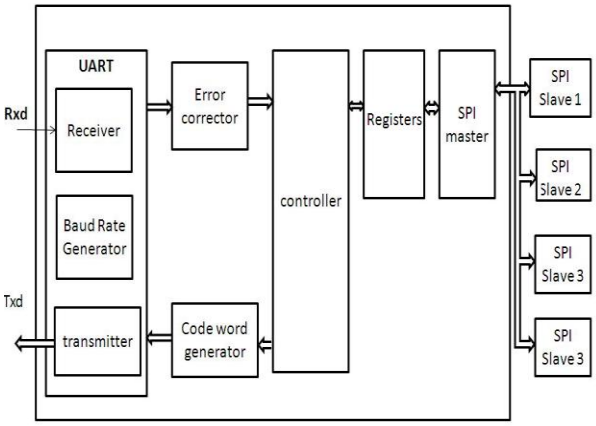

Fig. 5.1 Block diagram of UART-SPI interface with error detection and correction

Fig5.1. shows the received data is given to error corrector for detection and correction of errors if any. The error is detected by comparing the calculated check bits/parity bits with the received check bit. The error free data is stored in a SPI transmits data register and forward to the slave through MOSI. For transmission of data the controller should be initiated as per the requirement with write command (0x55) or read command (0xAA). Next the address of control register and slave address is to be transmitted followed by actual data. The control register consists of baud rate and other control signals.

The registers used are UART control register, SPI control register, SPI transmit data register, SPI receive data register. UART control register is of 8-bits, where first two bits are for selecting the baud rate, 3rd bit is for transmit parity enable, 4th bit is for transmit polarity phase, 5th bit is for receive parity enable, 6th bit is for receive polarity phase, 7th and 8th bits are reserved for future. SPI control register is of 8-bits, where 1st bits is for selecting clock phase, 2nd bit is for selecting clock polarity, 3rd and 4th bits are for selecting baud rate, 5th and 6th are for slave selection, 7th bit is for SPI transmission enable, 8th bit is for transmission or reception done. SPI transmit data register is of 8-bits and is used for transmission of data. SPI receive data register if of 8-bits and is used for reception of data. The selection of these registers is based upon the two bits of address. Where 00- used for UART control register, 01- used for SPI control register, 10- used for SPI transmit data register, 11- used for SPI receive data register.

VI RESULTS AND CONCLUSION

The results obtained from Xilinx ISE 13.2 implementation with the device xc3s500e-5fg320.

Fig. 6.1. Simulation results of code word generator

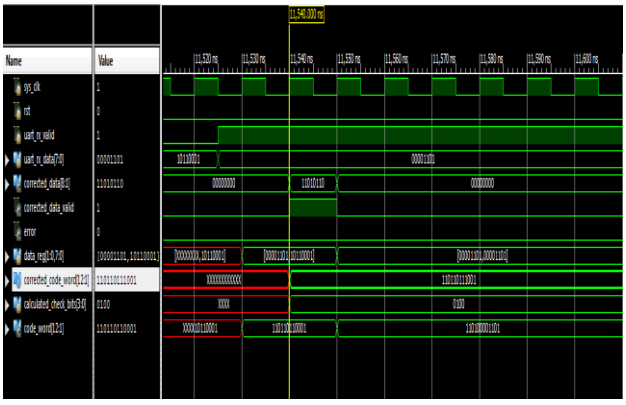

Fig. 6.2. Simulation results of Error corrector

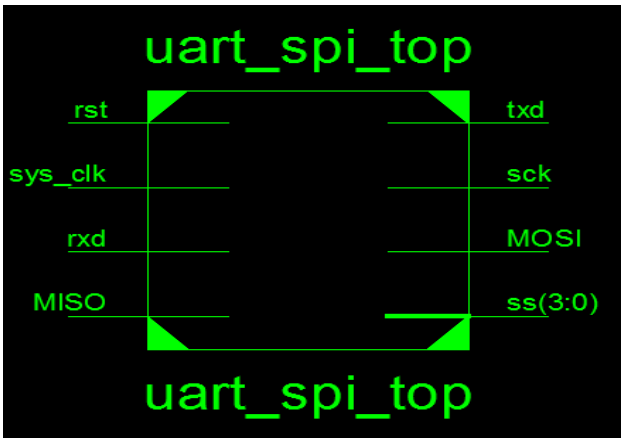

Fig. 6.3. RTL Schematic of top module uart_spi_top

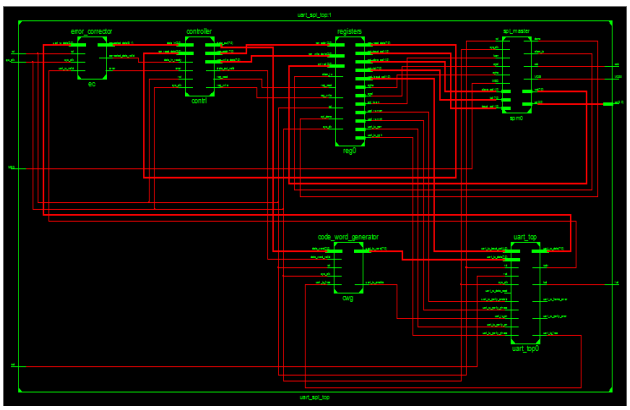

Fig.6.4 Internal structure of RTL Schematic of top module uart_spi_top

This paper proposed a new architecture of having UART-SPI interface along with error detection as well as correction. The technique used for error correction and detection is hamming code

REFERENCES:

[1] T. Praveen Blesington, B. Bhanu Murthy, G. V. Ganesh, T.S.R Prasad , " Optimal Implementation of UART-SPI Interface in SOC", International conference on Devices, Circuits and Systems (ICDCS), PP 673 – 677, Mar-2012

[2] Nutan Shep, Mrs. P.H. Bhagat, "Implementation of Hamming code using VLSI",International Journal of Engineering Trends and Technology, Vol.4Issue2- 2013

[3] Mohammad-Hamed Razmkhah, Seyed Ghassem Miremadi, and Alireza Ejlali, "A Micro-FT-UART for Safety-Critical SoC-Based Applications", International Conference on Availability, Reliability and Security, 2009

[4] M. Morris Mano, Charles Kime, "Logic and Computer Design Fundamentals" 3rd Ed. , Pearson Education 2004

[5] "Single Error Correction and Double ErrorDetection (SECDED) with CoolRunner-II™ CPLDs" published in Xilinx XAPP383 (v1.1), Aug -2003

[6] Hamming W.Richard. "Coding And Information Theory", Prentice-Hall Pub. ,2nd Edition

[7] Satyandra Sharad , Anjali Shobhna , Prof. Satender Sharma, "Designing Parallel Bus Using Universal Asynchronous Receiver Transmitter", International Journal of Electronics and Computer Science Engineering, Vol. 2 No. 2,PP. 472-476, Mar-2013

[8] M.Sandya, K.Rajasekhar, "Design and Verification of Serial Peripheral Interface"International Journal of Engineering Trends and Technology, Vol. 3 Issue4- 2012

[9] "LogiCORE IP XPS Serial Peripheral Interface (SPI) (v2.02a)", published in Xilinx DS570 June 22, 2011

[10] Frederic Leens, "An Introduction to I^2 C and SPI Protocols", IEEE Instrumentation & Measurement Magazine February 2009