Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, 2005 (pp520-525)

Parameterized Soft IP Core of High Performance ARINC 429 and UART Interfaces

USMAN YOUSAF, MUHAMMAD ASIF, *IMRAN QADEER
National Engineering and Scientific Commission, *Imagination Technologies
Islamabad, *London
PAKISTAN, *UK

Key-words: - System on chip, Intellectual Proprietary, Universal Asynchronous Receiver Transmitter.

Abstract: - ARINC 429 [1] is a commercial avionics standard for serial communication and the UART [2] is the most commonly used asynchronous serial communication interface. In existing interface modules read and write cycle time is high enough that today's high speed processors have to insert wait states between data read and write to or from these modules which results in decrease of overall performance of the system. Secondly, in most of the systems, frame size for data reception and transmission varies with the application, but the interrupt can be programmed for fixed frame sizes of 8, 16 and 32 words only. The achievement to be presented is the design of a high speed ARINC 429 module along with a UART, having a high speed processor interface with read and write cycles much shorter than the existing modules resulting in an improved performance of the system. Secondly, for varying frame sizes, system's interrupt can be programmed to interrupt the processor after receiving full frame which further enhances the performance. Furthermore, the ARINC transceiver is customized for point to point communication that reduced the complexity of design and hence made it more easy to use. While on the other side the UART offers a wide range of baud rates covering all the standard ones with programmable interrupt according to the receive frame size, from one byte to maximum of receive FIFO. This design is tested up to the clock speed of 24MHz with 16 bytes of transmit and receive FIFO. The hardware testing is performed by interfacing it with DSP TMS320C32. The soft IP core of the interface is fully parameterized and it can also be customized according to the application and may be embedded into a complete SoC solution.

1 Introduction

Integration density and performance of integrated circuits have gone through an astounding revolution in the last two decades. In 1960s, Gordon Moore, then with the Fairchild Corporation and later the confounder of Intel, predicted that the number of transistors that can be integrated on a single die would grow exponentially with time. This prediction, later called Moore's law, has proven to be amazingly visionary [3]. From the early 1970s the microprocessor has grown in performance and complexity at a steady predictable pace. The million-transistor chip barrier was crossed in the late 1980s. Clock frequencies have doubled every three years in the past decade and have reached into the GHz range. An important observation is that, as of now, these trends have not shown any signs of slowdown.

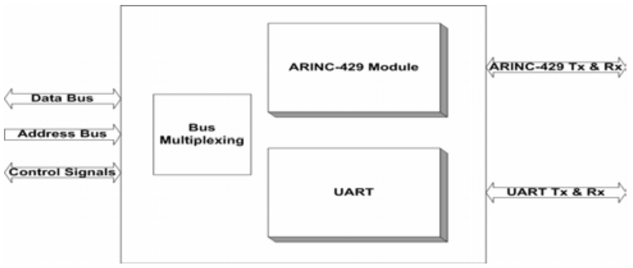

Fig.1: Block diagram of the interface

In ARINC-429, independent of speed and type of processor, serial communication can be carried out at a data rate of 12 – 14.5 or 100 Kbps. While asynchronous serial communication standards like RS-232 and RS-422 offer even greater data rates up to several Mbps. However, in the presently available interface modules for ARINC-429 and UART, parallel interface with the processor demands large read and write cycle times forcing today's high performance processors to insert wait states in software. In time critical applications this time can be utilized for other useful computations.
In this research project, parallel interface of the UART and the ARINC module to the microprocessor has been

enhanced by decreasing read and write cycle times. Another useful feature of this interface is that it is interrupt based and its interrupt is configurable by the central processing unit, according to the frame size to be received.

This design basically consists of a UART and an ARINC module as shown in Fig. 1. Transmit and receive lines of ARINC-429 and UART coming out from this interface are fed into the line drivers and receivers for level translation. A more detailed inside look shows a controller, FIFOs and an ARINC transceiver in ARINC module as shown in Fig. 2. On the other hand the UART has address decoding and register logic, FIFOs and control circuitry in it as shown in Fig. 5.

The controller of ARINC module is built around RISC architecture. ARINC transceiver is designed just like any other commercial one leaving some of the features untouched to reduce over head and unnecessary operational complexity. FIFOs in this design use DPRAM for storage while a control circuitry is designed to keep track of the data. Address decoding and register logic of the UART contains control registers and their address decoding. Transmit and receive shift registers are located in the control unit of the UART.

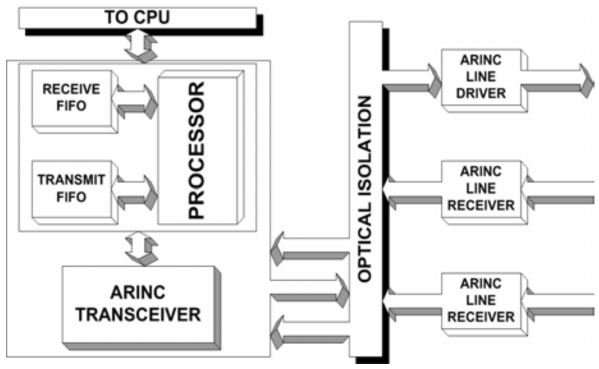

Fig.2: Block diagram of ARINC module

The interface contains two ARINC 429 receivers and one transmitter. Everything other than 429 line drivers and receivers was designed indigenously using Verilog HDL [4][5], totally independent of technology. As it is clear from its name that it is a soft core, anyone can use it in his custom design on FPGA or ASIC just as a library component without using separate ICs that's why it is best suitable for SoC designs. In this way soft core considerably reduces the overall cost, area and the hardware in board level design. The implementation was carried out on XCS40-3PQ208I FPGA from Xilinx Inc.

2 Arinc-429 Overview

ARINC 429 is a serial communication standard specially designed for commercial avionics equipment. In this standard the data is transmitted serially in the form of thirty two or twenty five bit words. Electronically data is represented in bipolar return to zero (RZ) format and the differential voltage levels used for that are +5V and -5V. Data rates supported by ARINC-429 standard are 12.5 – 14 kbps and 100 kbps.

This standard is preferred in avionics on other serial communication standards due to intelligence embedded into it and secure communication that it provides. Using this standard one transmitter can communicate with up to twenty receivers without any conflict on the same bus. Using source/destination identification feature receivers can check if the data received is meant for it or not. Each transmitted word has a label with it that enables the receiver to differentiate between different types of data. On the other hand differential signaling is more secure in a noisy environment.

3 Design and Implementation of ARINC Controller

The operation of ARINC interface is controlled by a microcontroller that is specifically designed for this sort of applications where event based control is required. In this one bit controller multi cycle instruction execution was employed rather than using typical fixed cycle instruction execution [6]. Using this technique a lot of time has been saved because when fixed cycle execution is used all the instructions take the same time for execution even if the objective of the instruction has been achieved in half or even less time. But in multi cycle execution, each instruction takes as much time as it needs without wasting time. The controller has an instruction set of just ten instructions which were chosen from 8051 microcontroller instruction set but, however, hardware implementation of these instructions is done in a different way. Instruction set is described in Table 1 and the block diagram of controller is given in Fig. 3.

The controller has eight bit wide input port and sixteen bit wide output port all of which are bit addressable. The software written for specific application is stored into the ROM inside processor. The ROM size employed in this design was chosen to be one hundred and twenty eight bytes as it was enough for this specific application. However, the size of ROM can be adjusted very easily as the design is fully parameterized. The software embedded into the ROM was written in the assembly language of the controller and then converted into machine language manually. The 32 byte RAM is being used as stack memory to store the program counter.

Since ROM size for this design was one hundred and twenty eight, seven bits were enough for program counter and hence for RAM locations. As the architecture is register intensive, there are quite a few registers in the controller. These registers are contained inside a register file. All components of the controller are controlled by a control unit. A Mealy state machine [7] is used in this design for building the control unit.

Sr. No.	Instruction	Meanings
1	Clrb Addr	Clear bit at Addr
2	Setb Addr	Set bit at Addr
3	Clr Cry	Clear carry
4	Set Cry	Set carry
5	Ret	Return from subroutine
6	Jmp Addr	Jump to Addr
7	Jnb Bit, Addr	Jump to Addr if bit not set
8	Jb Bit, Addr	Jump to Addr if bit set
9	Jsr	Jump to subroutine
10	Nop	No operation

Table 1: Instruction set summary of the controller

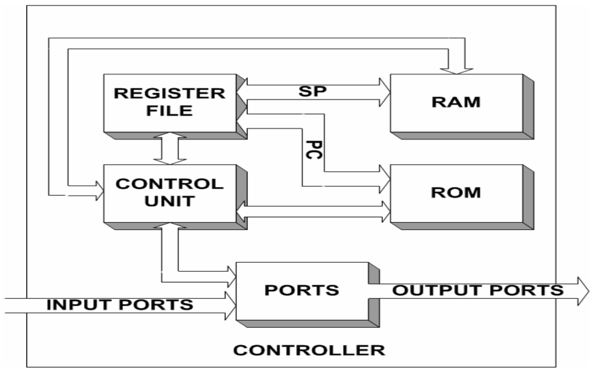

Fig.3: Architectural design of controller

4 Design and Implementation of ARINC Transceiver

The ARINC transceiver implemented in the design is also customized according to the requirements of the design leaving some of the features available in commercial integrated circuits. It reduced the unnecessary complexities involved in the design. Block diagram of transceiver is shown in Fig.4.

The control word written for this transceiver is stored in control register. Through this control word parity, word size and transmitter and receiver baud rates can be adjusted. The control unit generates control signals for other modules after examining this control word. For the generation of baud rate according to the control word, a baud rate generator is designed. This baud rate generator takes the clock coming from outside and then divides it by an appropriate number for the generation of required baud rate. The output clock from baud rate generator is then provided to the transmitter and receiver circuits separately. Internal operation of the transmitter and receiver circuits is carried out at a clock rate ten times faster than the selected baud rate in order to ensure that no data loss occurs and the skew generation is minimized. In the transmitter, data sent from the transmit FIFO is first stored in a data register and then it is shifted into a shift register from where data is transmitted bit by bit at the adjusted baud rate. Similarly in the receiver, data is received bit by bit in a shift register and when complete word is received, data is shifted into a data register from where it is sent to receive FIFO.

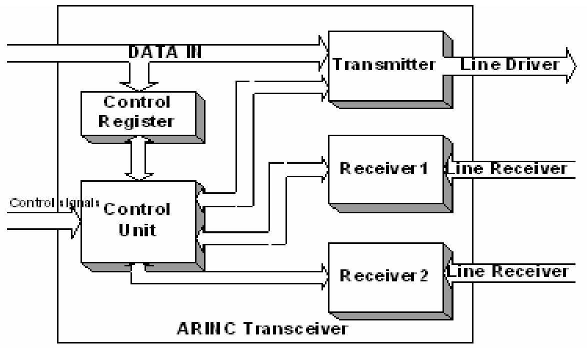

Fig.4: Block diagram of ARINC transceiver

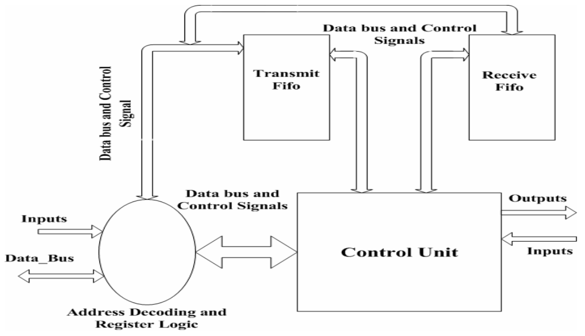
Data bus and Control Signals
Transmit Fifo
Receive Fifo
Data bus and Control Signal
Data bus and Control Signals
Outputs
Inputs
Data_Bus
Control Unit
Inputs
Address Decoding and Register Logic

Fig.5: Block diagram of UART

5 Address Decoding and Register Logic of UART

This section generates the control signals for transmit and receive sections of UART and performs the address decoding for the UART registers and FIFOs. This block contains writeable command and baud rate control registers and one writeable and readable FIFO size control register to configure the receiver interrupt. These registers are designed in such a way that reading and writing to these registers is very fast. This block is also responsible for multiplexing on bi-directional data bus.

6 UART Control Unit

Control unit is the major and important part of the UART. This section is responsible for handling all the control functions of the UART. It has many small portions which are mutually working and designed in such a way to keep the delay as minimum as possible. This unit can be partitioned into the following subsections.

1. Baud generator.
2. Reset controller.
3. Interrupt controller.
4. Transmit shifter.
5. Receive shifter.
6. Data bus controller.
7. Error checker.

Baud generator is responsible for generating all the baud rates according to the baud word written in the baud register. It has first stage divisor of 16/3 and second stage has multiple divisors of 2. Baud register bits are used as signals to select the required baud rate from the Baud generator. Baud generator is also parameterized whose counter and register size can be changed. It is designed in such a way that it not only covers all the baud rates available in other UARTs but also higher ones.

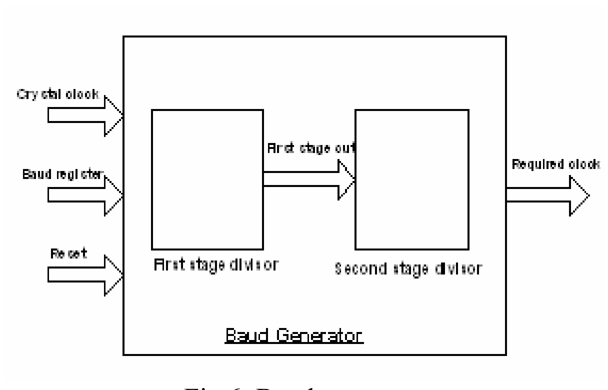

Fig.6: Baud generator

B8	B7	B6	B5	B4	B3	B2	B1

Table 2: Baud register format

Reset controller has reset circuitry which utilizes external reset and software reset as input to reset the UART. Both resets initialize the device. Software reset provides us the flexibility to reset it whenever required without resetting all the hardware. If internal reset bit of command register is set it become automatically turn to zero after resetting the device.
Interrupt controller handles the transmitter and receiver interrupts of the UART. Both interrupts can be enabled and disabled independently. Receiver interrupt is configurable and user can select the limit where he wants the receiver interrupt. It utilizes the read and write pointers of receive FIFO and packet register. Interrupt is active low and duration of that interrupt is equal to the three crystal clocks.
Transmitter and receiver shifters are same in architecture but one is used for transmitting and other for receiving purposes. These are the 11-bit shift registers which shifts data using transmit and receive clocks from the baud generator. Transmit register contains start, data, parity and stop bit. First it transmits start bit then LSB of data byte and in last stop bit. Receive shifter receive the data in the same way.
Receiver is designed in such a way that it samples the serial data in the middle, minimizing chances of error. Receiver goes in idle state after receiving each frame and starts receiving data on detection of start bit.
Data bus controller handles the selection of data out at data bus of the UART core. Error checker checks all the errors in data communication whether it is parity error, framing error or overrun error. These errors are checked as complete frame reaches in the receive shifter. Error reset is also provided to reset the error bits of the status when error occurs.

7 FIFO Design and Implementation

FIFO on ARINC module's transmit side is 16x18 and on the receive side it is 64x18. While in UART it is 16x8 on both the sides. Size of the FIFOs can be changed according to the user requirement.
In ARINC module the complete ARINC word is stored in two parts as they come from the processor or from receiver. Along these parts of the word, two extra bits are stored for sequencing and recognition of the words. For example, on the transmit side data must be distinguished as the control word, part one of the word or part two of the word. Similarly, on the receive side one FIFO is employed for both the receivers, therefore some flags are required to distinguish between data from both the receivers and part one and part two of the words from separate receivers. These flags stored along with data are showed up in a separate status register so that

the central processing unit can recognize the data and its sequencing by reading the status register.

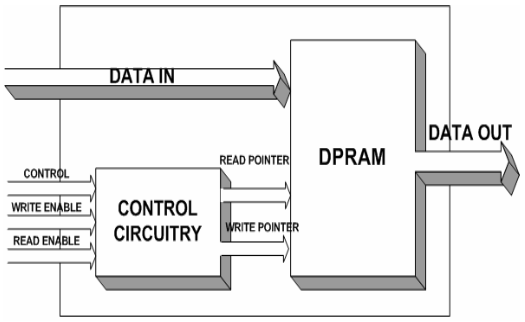

Fig.7: Block diagram of FIFO

In UART a 16x8 FIFO is on transmit side and another 16x8 FIFO on receive side. FIFO empty and full flags with transmitter ready and receiver ready flags are present in the status register of the UART for error free transmission and reception.
The architectural design of FIFO is shown in Fig. 7. For the storage of data a dual port RAM is used. Dual port rams are created using “LogiBlox” module generator from Xilinx. The reason for using a DPRAM is to ensure faster operation of FIFO since it has two ports, read and write operations can occur simultaneously. Dual Port rams created using this tool cover minimum chip area as compared to the rams created using Verilog construct and reading and writing to these dual port rams are very fast. The control circuitry consists of read and write pointers and their comparisons generate FIFO full and empty flags and prevent it from overwriting. All FIFOs are parameterized and their sizes can be customized according to user requirement.

8 Configurable Interrupt

There are two separate registers in ARINC module and UART to determine the frame size. Central processing unit writes these register to describe the frame length, otherwise, the interface would use the default frame length of one for ARINC module and eight for UART. However, like all other features this can also be customized according to the application. The frame size is basically used in reception, the number of words stored in FIFO are compared with this register and when number of words in FIFO equals the stored value an interrupt is sent to the central processing unit to indicate that complete frame is received and it can read its data now. This feature is very useful because using this, the central processing unit needs not to poll the status of FIFO again and again, and hence time is saved.

9 Verification and Testing

The design entry tool used is HDL Designer while simulation was carried out using Modelsim. For the purpose of testing first the functional simulation was carried out for each individual design unit after that all the design units were integrated and integrated functional simulation was performed.
After functional simulation the design was synthesized using Leonardo spectrum level-3 for XCS40-3PQ208I. Leonardo Spectrum generated an EDIF file for the design. This file was used as an input to the Xilinx foundation series for place, route and implementation of the design on the device stated earlier. It generated a device downloadable bit file along with a gate level Verilog file and SDF file that contains the delay information of the technology dependent modules used in that gate level Verilog code. With the help of these files post layout timing simulation was performed.
After the design was fully verified in timing simulation, the design was tested in hardware. The bit file generated by Xilinx ISE was downloaded into an EEPROM from where it is uploaded in the FPGA every time it powers up. The UART was independently tested with a PIC microcontroller based testing jig. The data written by PIC16F877 to the UART in parallel was transmitted serially. This serial data was received in PC and the same data was transmitted back to the UART. Then this data was read by the controller and compared with the original data.
For integrated testing of all the modules, Texas Instrument’s TMS320C32 digital signal processor (DSP) based kit was used. The software for the DSP was written in C [8] programming language using Code Composer Studio. The data written from the DSP emulator in parallel was transmitted in ARINC-429 standard from the interface. This data was received in a standard ISA based ARINC card by DDC corporation. From there the same data was sent back to the interface on both the receiving channels. These receivers received the data, stored it into FIFO and after receiving the required number of words interrupted the DSP. The data written to the UART was serially transmitted on RS-232. It was received in a PC and from there the same data was transmitted back to the UART receiver which stored this data in receive FIFO. After receiving the complete packet an interrupt was sent to the DSP. On receiving interrupt, the DSP read the data stored in the receive FIFO and compared it with the original data. In this way the functionality of the interface was fully tested.

10 Conclusion

The performance of this interface proved that it can serve the purpose it was designed for. It can survive the day by day increasing demand for speed alongside ensuring a safe communication. Its performance, if compared with commercially available ARINC-429 interfaces proves that it is almost three hundred and thirteen percent (313%) faster. This is shown in Fig.8 clearly in a comparison between times taken by read cycles in both the interfaces. Normally the commercially available interfaces take round about 470ns for one read cycle while this interface takes just 150 ns at the most. It can be even lesser than this if an FPGA with improved speed grade is used.

In addition to it the interface also contains the UART. The parallel interface of the UART is also enhanced in terms of read and write cycles. UART core is 60% faster than commercially available UART. Fig. 9 undoubtedly shows the comparison between both UARTS. Commercial UART takes almost 100ns of time for its cycles while UART core takes 40ns for its read and write cycle. Same is the case for read cycles of UART core. This time can be further reduced by increasing the clock speed or by using an improved speed grade FPGA.

These figures are fully tested and verified according to the procedure described under the heading of "Verification and Testing".

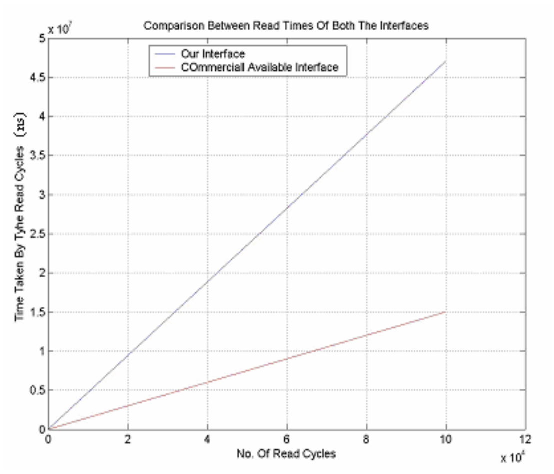

Fig.8: A comparison between the times taken by the read cycles in both the ARINC interfaces

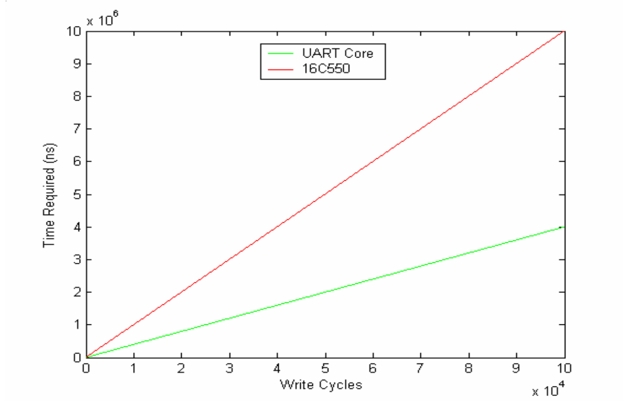

Fig.9: A comparison between the time taken by the write cycles of both the UARTs

References:

[1] www.aims-online.com

[2] www.beyondlogic.org/serial.pdf

[3] Jan M. Rabaey, Anantha Chandrakasan, Brivoje Nikloic, Digital Integrated Circuits: A Design Perspective, Prentice Hall, 2003.

[4] Samir Palnitkar, Verilog HDL: A Guide to Digital Design and Synthesis, Prentice Hall, 2000.

[5] Zainalabedin Navabi, Verilog Digital System Design, McGraw-Hill Inc, USA, 1999.

[6] David A. Patterson, John L. Hennessy, Computer Organization & Design: A Hardware/Software Interface, Morgan Kaufmann Publishers Inc., 2000.

[7] John F Wakerly, Digital Design Principles and Practices, Prentice Hall, Third Edition, 2000.

[8] Robert Lafore, The Wait Group's Turbo C Programming for PC and Turbo C++, SAMS, Third Edition, 1999.