



Implementing Serial Communication for the Instructional Processor

Dr. Ronald J. Hayne, The Citadel

Ronald J. Hayne is a Professor in the Department of Electrical and Computer Engineering at The Citadel. He received his B.S. in Computer Science from the United States Military Academy, his M.S. in Electrical Engineering from the University of Arizona, and his Ph.D. in Electrical Engineering from the University of Virginia. Dr. Hayne's professional areas of interest include digital systems design and hardware description languages. He is a retired Army Colonel with experience in academics and Defense laboratories.

Implementing Serial Communication for the Instructional Processor

Abstract

An Instructional Processor has been developed for use as a design example in an Advanced Digital Systems course. The architecture is modelled in VHDL and can be simulated using Xilinx design tools to demonstrate operation of the processor. A basic microcontroller is then created by adding memory-mapped input/output (I/O). The system can be synthesized and implemented in hardware on a field programmable gate array (FPGA). The goal of this project was to add serial communication capabilities to the Instructional Processor via software and hardware. The enhanced microcontroller can then be interfaced with multiple peripheral devices.

The approach for this project was to adapt a UART (universal asynchronous receiver transmitter), based on the MC6811, to the memory-mapped I/O interface developed for the Instructional Processor. This implementation allows direct access to the UART data registers (receive and transmit), status register (flags), and control register (baud rate). Test programs, written in assembly language, were used to test the communication protocol and timing via VHDL simulation. The FPGA microcontroller was also able to communicate with several serial devices at various baud rates.

This project successfully added serial communication capabilities to the Instructional Processor. Software and hardware implementations were developed and tested using VHDL and a Xilinx FPGA. The UART has now been added to the processor design example in the Advanced Digital Systems course, giving students an in-depth look at both the internal details and external interfacing of a real-life system. Feedback has been very positive that the simulations and microcontroller implementation help illustrate fundamental design concepts reinforced by actual functioning hardware.

Introduction

Teaching digital design involves use of many examples including counters, registers, arithmetic logic units, and memory. The design of a computer processor combines these components into an integrated digital system. An Instructional Processor has been developed as a design example in an Advanced Digital Systems course at The Citadel [1], [2]. The simple architecture provides sufficient complexity to demonstrate fundamental programming concepts. The entire system is modeled in VHDL and can be simulated to demonstrate operation of the processor. Memory-mapped input/output (I/O) provides the external interfaces necessary to demonstrate example microcontroller applications, when synthesized to a field programmable gate array (FPGA).

Serial communication is widely used to connect external devices to computer systems. The communication interface, which receives and transmits serial data, is commonly known as a UART (universal asynchronous receiver transmitter). The serial data format uses standard bit timing and framing. The protocol can be implemented in software using timing loops and basic

bit shifting, but will be limited to half-duplex (only one direction at a time). A full-duplex hardware implementation requires interfacing with internal processor registers and memory.

Software UART

The standard format for asynchronous serial data transmission is shown in Figure 1 [3]. The beginning of a data byte is indicated by a start bit and the end is framed by a stop bit. Data bits are sent least significant bit (LSB) first and the eighth bit is an optional parity bit. The number of bits transmitted per second is commonly referred to as the baud rate. This rate establishes the timing for each bit.

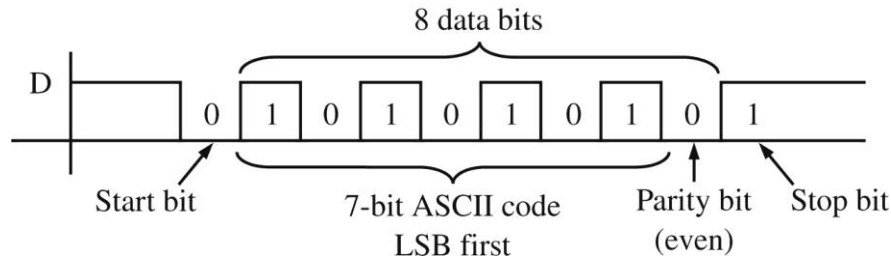


Figure 1. Standard Serial Data Format.

A software UART was created using delay loops to determine bit timing. For a communication rate of 9600 baud, the required bit time is 104 us. The FPGA implementation of the Instructional Processor on a BASYS 3 board uses a 100 MHz system clock [4] and a known number of clock cycles for each instruction. Calculating the timing for each nested loop results in the delay subroutine shown in Figure 2.

```

;Delay Subroutine
;104us
D104:  MOVE  26, R3
LOOP1: MOVE  50, R2
LOOP2: ADD   -1, R2
      BNZ  LOOP2
      ADD  -1, R3
      BNZ  LOOP1
      RTN

```

Figure 2. Assembly Language Delay Subroutine.

Implementing the software UART required using pins on two existing memory-mapped I/O ports for the received serial data (RxD) and transmitted serial data (TxD). Additional memory data registers were used for the receive shift register (RSR) and the transmit shift register (TSR). A receive subroutine was written to sample RxD and collect the data bits in the RSR. A transmit subroutine used the TSR to shift data bits out TxD.

The software UART was tested using a main program to receive a serial data message and save the ASCII characters to a memory table. The end of the message was indicated by the carriage return character. The message was then retransmitted to the sender to verify its contents. A PC

based serial communication terminal program was used to transmit the ASCII message at 9600 baud. The serial data was received by the FPGA microcontroller, running the software UART program, and successfully retransmitted back to the PC. The test results demonstrate half-duplex serial communication by the Instructional Processor.

Hardware UART

In order to achieve full-duplex serial communication, a hardware UART needs to be interfaced with internal processor registers and memory. While VHDL models of hardware UARTs exist, they are often presented as stand-alone devices [5], [6] or intellectual property cores that connect to proprietary bus architectures [7]. The approach for this project was to adapt a UART model, based on the MC6811 [3], to the memory-mapped I/O interface developed for the Instructional Processor.

The UART block diagram and memory interface are shown in Figure 3. This implementation allows direct access to the UART data registers for receive and transmit (RDR and TDR), serial communication status register (SCSR) for flags RDRF (receive data register full) and TDRE (transmit data register empty), and control register (SCCR) for baud rate, via memory-mapped I/O. Dedicated Rx/D and Tx/D pins were also mapped to the FPGA and BASYS 3 board. The processor memory model was modified to interface with the UART control signals (Read RDR, Write TDR, and Write BAUD) when accessing the associated memory locations.

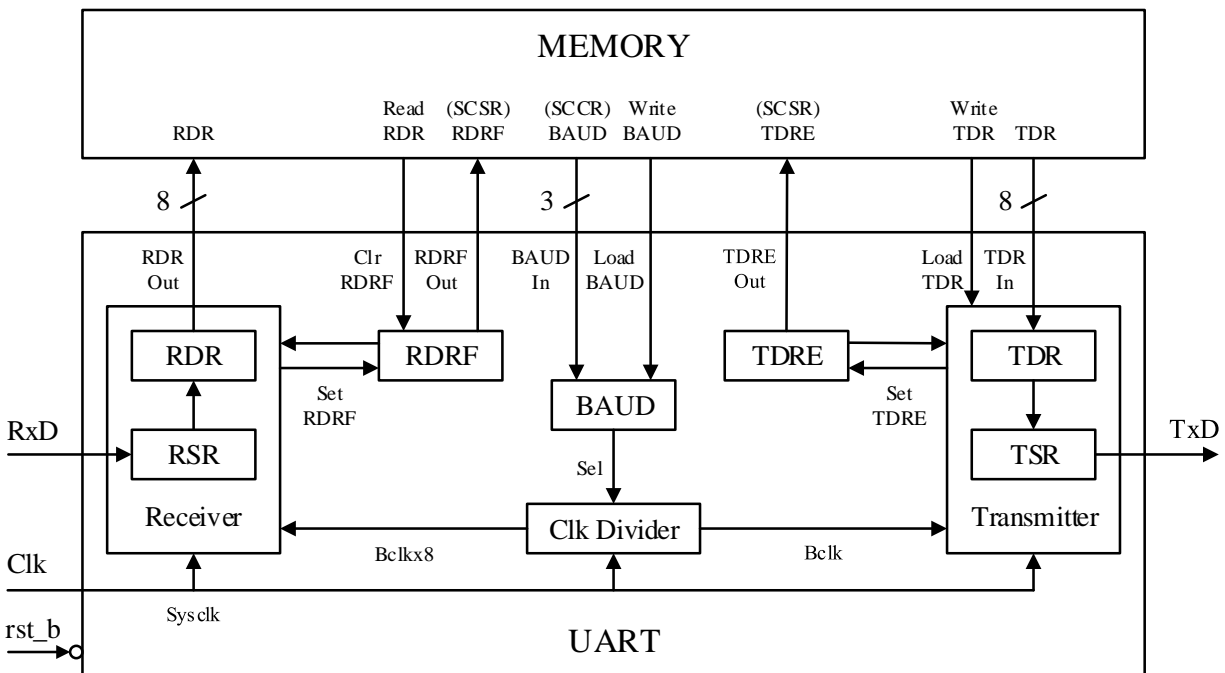


Figure 3. UART Block Diagram and Memory Interface.

VHDL models for the receiver, transmitter, and clock divider were adapted to use the memory-mapped registers and control signals. As an example, operation of the transmitter is described by the following sequence:

- Microcontroller waits until TDRE = '1'
 - Loads data into TDR
 - Clears TDRE
- UART transfers data from TDR to TSR
 - Sets TDRE
- UART outputs start bit ('0') then shifts TSR right eight times followed by a stop bit ('1')

Control logic is represented by the state machine chart shown in Figure 4 [3]. A behavioral VHDL model was developed for the transmitter component, which was then instantiated into the UART module using structural VHDL. The remaining modules were developed and integrated to complete the hardware UART.

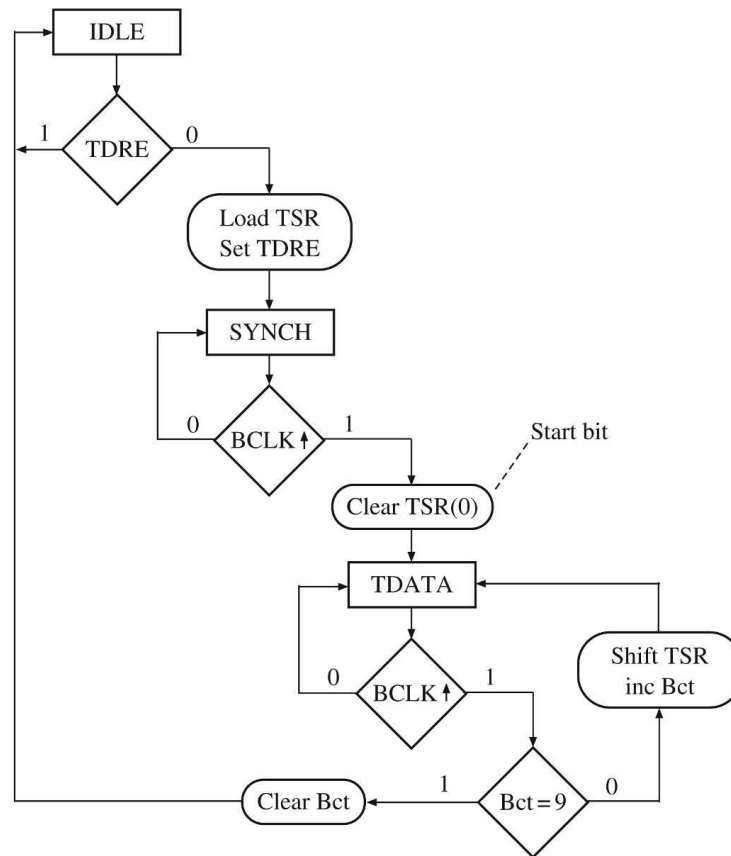


Figure 4. Transmitter State Machine Chart.

Testing the Hardware UART

The VHDL model for the integrated UART was first evaluated using a loopback test. The TxD pin is connected directly to the RxD pin and a single data byte (0x65) is transmitted (and received) at 9600 baud to verify operation and timing. The VHDL simulation results, using Xilinx Vivado [8], are shown in Figure 5. The bit time of 104 us can be verified by cursors on the bit clock (Bclk). The transmitted data is first loaded into REG4[1] and can be seen on TxD as it is sent LSB first (framed by start and stop bits). The correctly received data is stored in REG4[0], which can be seen at the end of the simulation.

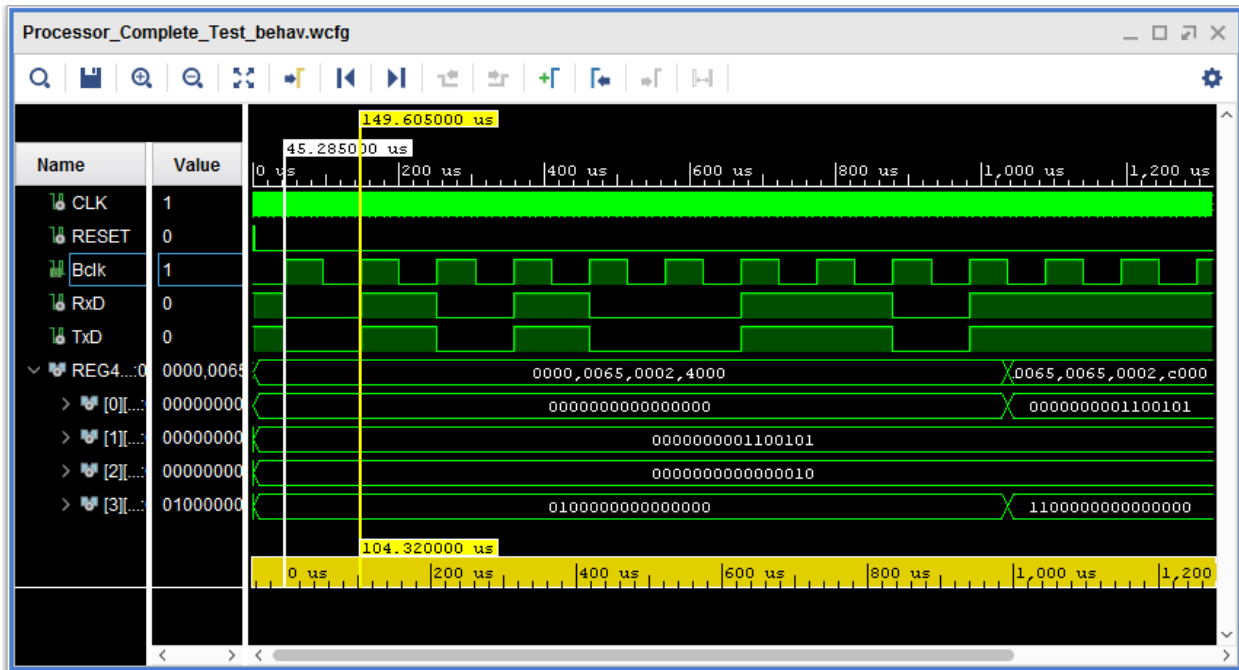


Figure 5. VHDL Simulation Results for Loopback Test.

Next, the FPGA microcontroller was tested by interfacing with the PC based serial communication terminal. First, the results of the software UART were replicated with half-duplex reception, storage, and retransmission of a character string. Finally, full-duplex communication was tested by simultaneous reception and retransmission (echo) of an entire text data file (containing thousands of characters). Unfortunately, this test failed and numerous characters were dropped from the outgoing message.

VHDL simulation of the full-duplex echo test revealed that the transmitter was generating two stop bits, instead of just one. This meant that the transmitter would fall behind the receiver and miss characters in the message. Analysis of the original transmitter state machine chart (Figure 4) revealed a logic flaw when two characters were transmitted back-to-back. This problem was corrected by skipping the (re-) SYNC state if another data byte was immediately ready to transmit (TDRE = 0), as shown in Figure 6. The updated VHDL model was synthesized to the FPGA and the echo test was demonstrated successfully.

```

when TDATA =>
  if (Bclk_rising = '0') then nextstate <= TDATA;
  elsif (Bct /= 9) then
    shftTSR <= '1'; inc <= '1'; nextstate <= TDATA;
  elsif (TDRE = '0') then -- eliminate second stop bit
    loadTSR <= '1'; start <= '1'; clr <= '1'; nextstate <= TDATA;
  else clr <= '1'; nextstate <= IDLE;
  end if;

```

Figure 6. Corrected VHDL Model for Transmitter.

Example Microcontroller Application

An example microcontroller application was chosen to demonstrate the newly added serial communication capability. A Parallax RFID (radio frequency identification) Card Reader [9] provides the capability to sense passive transponder tags used for access control and inventory tracking. The card reader transmits serial data at 2400 baud. Each tag has a unique ID consisting of a 10-byte ASCII string framed by start (0x0A) and stop (0x0D) bytes.

The card reader is connected to the FPGA microcontroller using the serial output (SOUT) and active low enable (/ENABLE) shown in Figure 7. An assembly language program was written to receive the 2400 baud serial data stream and store the tag ID in a memory table. The last character from the ID was then displayed on the BASYS 3 board 7-segment display, which is part of the memory-mapped I/O. The microcontroller successfully read and displayed all tag IDs and made a great demonstration of a real-life application.

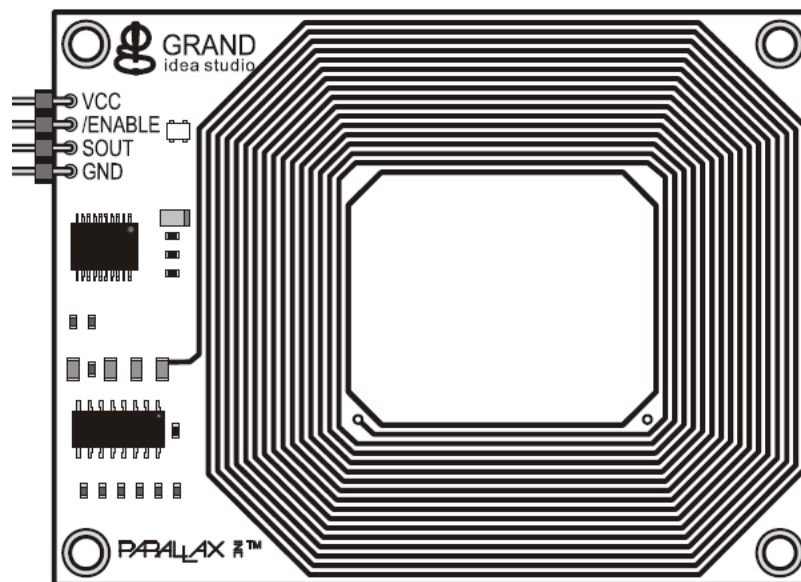


Figure 7. Parallax RFID Card Reader.

Results and Conclusions

This project successfully added serial communication capabilities to the Instructional Processor. First, a software UART was developed using timing loops and shift instructions to implement the standard asynchronous serial communication protocol. Testing, using a PC based serial communication terminal, verified half-duplex reception and retransmission of an ASCII text string. This implementation has been incorporated into the assembly language programming examples used as demonstration applications for the Instructional Processor.

The hardware UART started as a final, stand-alone, design example in the Advanced Digital Systems course. It has now been integrated into the memory-mapped I/O of the Instructional Processor. The VHDL models are simulated using Xilinx Vivado, providing functional

verification of the serial communication protocol. Students gain valuable insights into using simulation to identify and correct a logic flaw in the original UART design. The FPGA implementation then provides students actual hardware they can interface with, rather than just VHDL simulations.

An RFID Card Reader was chosen as a microcontroller application for the FPGA implementation of the Instructional Processor. The design example gives students experience with interfacing a commercial product with their hardware and software design. RFID seems to be a student favorite, because they see real-life uses for the technology. The end result of this project is an expanded processor design example that continues to achieve its goal as a valuable instructional tool.

References

- [1] R. Hayne and J. Moore, "Evolution of the Instructional Processor," *Computers in Education Journal*, ASEE, Vol. 6 No. 4, October - December 2015.
- [2] R. Hayne, "Design of an Instructional Processor," Supplement to: C. Roth and L. John, *Digital Systems Design Using VHDL*, Third Edition, Cengage Learning, Boston, MA, 2018. [Online]. Available: http://academic.cengage.com/resource_uploads/downloads/1305635140_559956.pdf.
- [3] C. Roth and L. John, *Digital Systems Design Using VHDL*, Second Edition, Thompson, Toronto, Canada, 2008.
- [4] BASYS 3 FPGA Board Reference Manual, Digilent, Inc., April 2016.
- [5] UART.vhd, Digi-Key Electronics, May 2017. [Online]. Available: <https://www.digikey.com/eewiki/download/attachments/59507062/uart.vhd?version=1&modificationDate=1495813046350&api=v2>.
- [6] C. Unsalan and B. Tar, *Digital System Design with FPGA*, McGraw-Hill Education, New York, NY, 2017.
- [7] AXI UART Lite, PG142, v2.0, Xilinx, Inc., April 2017.
- [8] Vivado Design Suite User Guide, UG892, v2017.2, Xilinx, Inc., June 2017.
- [9] RFID Card Reader, Serial (#28140), v2.2, Parallax, Inc., March 2010.