Implementing Serial Communication for the Instructional Processor

Ronald J. Hayne
The Citadel

An Instructional Processor has been developed as a design example in an Advanced Digital Systems course. The architecture is modelled in VHDL and can be simulated using Xilinx design tools. A basic microcontroller is created by adding memory-mapped input/output (I/O). The hardware system can be synthesized and implemented on a field programmable gate array (FPGA). The goal of this project was to add serial communication capabilities via software and a hardware UART (universal asynchronous receiver transmitter). The design 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 verify the communication protocol and timing via VHDL simulation. The FPGA microcontroller was able to communicate with serial devices at various baud rates. The UART gives students an in-depth look at both the internal details and external interfacing of a real-life system.

Keywords: VHDL, FPGA, UART

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 (Hayne, 2018). 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).

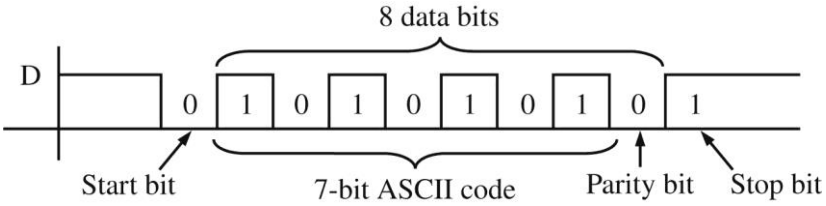
Serial communication is widely used to connect external devices to computer systems. The digital interface, which receives and transmits serial data, is commonly known as a UART (universal asynchronous receiver transmitter). The goal of this project was to develop design examples which add serial communication capabilities to the Instructional Processor. Students had the opportunity to explore both software and hardware UARTs.

The serial data format uses standard bit timing and framing. One possible implementation of the protocol is in software using timing loops and basic bit shifting. Unfortunately, the software UART will be limited to half-duplex (only one direction at a time). A full-duplex hardware UART requires interfacing with internal processor registers and memory. Available implementation options are also discussed.

SOFTWARE UART

The standard format for asynchronous serial data transmission is shown in Figure 1 (Roth & John, 2008). 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 with 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 data rate of 9600 baud, the required bit time is 104 µs. The FPGA implementation of the Instructional Processor on a BASYS 3 board uses a 100 MHz system clock (Digilent, 2019) 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.

FIGURE 2
ASSEMBLY LANGUAGE DELAY SUBROUTINE

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

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.

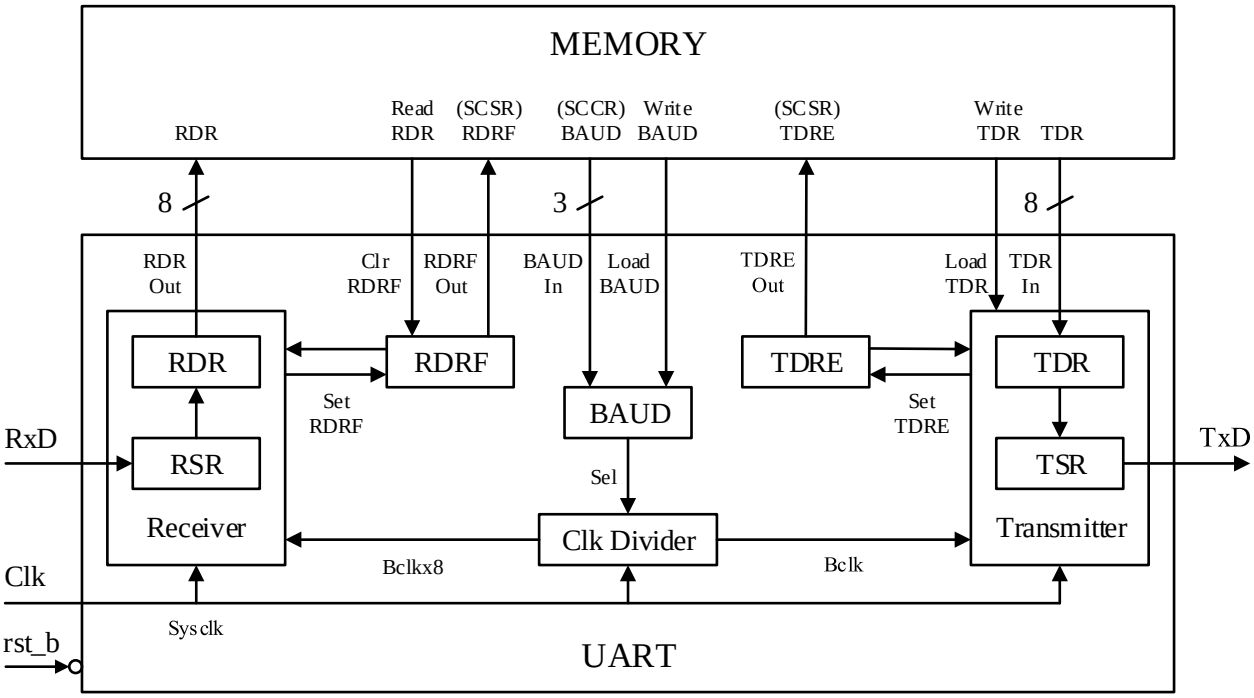
The software UART was tested using a main program to receive a serial data message and save the ASCII characters to a memory table. The end of the message was indicated by the carriage return character (0x0D). The message was then retransmitted to the sender to verify its contents. A PC based serial 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 (Digi-Key Electronics, 2021) (Unsalan & Tar, 2017) or intellectual property cores that connect to proprietary bus architectures (Xilinx, 2017). The approach for this project was to adapt an existing UART model, based on the MC6811 (Roth & John, 2008), to the memory-mapped I/O interface already 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 RxD and TxD pins were also mapped to the FPGA and BASYS 3 board. The processor VHDL 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

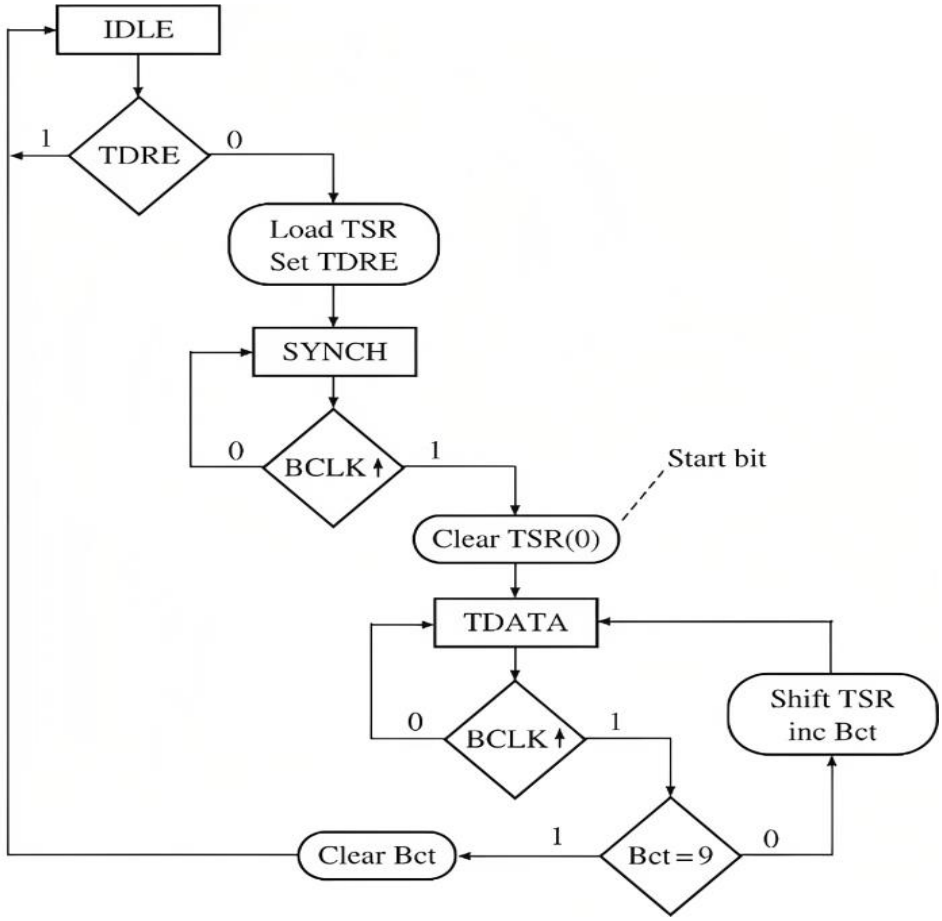


VHDL models for the receiver, transmitter, and clock divider were then adapted to use the new memory-mapped registers and control signals. As an example, control logic for the transmitter is represented by the state machine chart in Figure 4 (Roth & John, 2008). Operation of the transmitter can be described 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')

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.

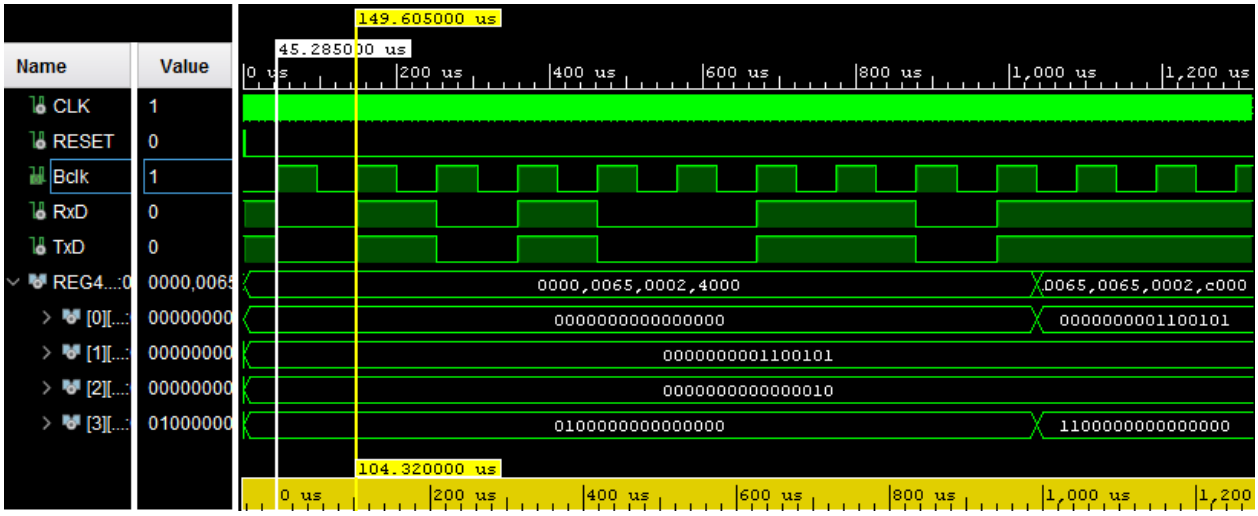
FIGURE 4
TRANSMITTER STATE MACHINE CHART



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 Vivado (Xilinx, 2019), are shown in Figure 5. The bit time of 104 μs 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.

The failed echo test provided an opportunity for students to observe the troubleshooting process required to identify and correct this design error. 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. Further 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.

FIGURE 6
CORRECTED VHDL MODEL FOR TRANSMITTER

```
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;
```

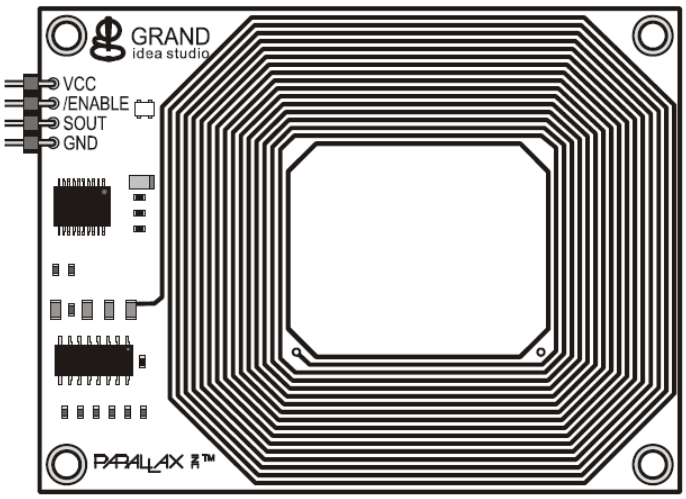
EXAMPLE MICROCONTROLLER APPLICATION

An example microcontroller application was chosen to demonstrate the newly added serial communication capability. An RFID (radio frequency identification) Card Reader (Parallax, 2016) 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 also part of the memory-mapped I/O. The UART capable microcontroller successfully read and displayed all tag IDs and made a great demonstration of interfacing a commercial product for 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.

Future work on the Instructional Processor includes development and integration of an interrupt system for timing applications and the UART. The guiding principle will be keeping the implementation simple enough to use as a teaching example, thus providing students with insights into capabilities available in modern commercial microcontrollers.

ACKNOWLEDGEMENT

This paper is a revised version of a paper presented at the 2020 American Society for Engineering Education Virtual Conference (Hayne, 2020).

REFERENCES

Digi-Key Electronics. (2021). *UART.vhd*. Retrieved from https://forum.digikey.com/t/uart-vhdl/12670

Digilent. (2019). *BASYS 3 FPGA Board Reference Manual*. Retrieved from https://digilent.com/reference/_media/reference/programmable-logic/basys-3/basys3_rm.pdf

Hayne, R.J. (2018). Design of an Instructional Processor [Supplemental material]. In Roth, C.H., Jr. & John, L.K. *Digital Systems Design Using VHDL* (3rd Ed.). Boston, MA: Cengage Learning**.** Retrieved from http://academic.cengage.com/resource_uploads/downloads/1305635140_559956.pdf

Hayne, R.J. (2020). Implementing Serial Communication for the Instructional Processor. *Proceedings ASEE Virtual Conference*.

Parallax. (2016). *RFID Card Reader, Serial #28140*. Retrieved from https://www.parallax.com/package/rfid-card-reader-documentation/

Roth, C.H., Jr., & John, L.K. (2008). *Digital Systems Design Using VHDL* (2nd Ed.). Toronto, Canada: Thompson.

Unsalan, C., & Tar, B. (2017). *Digital System Design with FPGA*. New York, NY: McGraw-Hill Education.

Xilinx. (2017). *AXI UART Lite v2.0*. Retrieved from https://docs.xilinx.com/v/u/en-US/pg142-axi-uartlite

Xilinx. (2019). *Vivado Design Suite User Guide*. Retrieved from https://docs.xilinx.com/v/u/2019.1-English/ug893-vivado-ide