International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2020): 7.803

Interfacing BLE Module on FPGA using Nios II

Shama. S. Naik[1], Pritam Thomke[2]

[1]Student of M. E (Microelectronics, Goa College of Engineering), Farmagudi, Goa, India
shamanaik10[at]gmail.com

Hardware Developer, Siemens Limited, Verna Goa
pritam. thomke[at]siemens. com

Abstract: ***This paper presents the implementation of Bluetooth Low Energy (BLE) on Field Programmable Gate Array (FPGA) using System on Programmable chip (SOPC). The design is implemented using the soft intellectual property (IPs) of the Nios II processor. The test results are verified on the serial terminal. This implementation has applications in the designing of wireless gateway on FPGA.***

Keywords: Nios II Processor, BLE, UART

1. Introduction

Bluetooth Low Energy (BLE) is an upcoming technology designed as both, a complementary technology to classic bluetooth as well as the lowest possible power wireless technology that can be designed and built. BLE is deployed in high volumes, in devices that do not have wireless technology today. Bluetooth incorporated basic rate (BR) with a maximum physical data rate of 1 megabit per second. Version 2.0 of Bluetooth had enhanced data rate to increase the physical layer data rate of 3MBps [1].

This design implements dual UART in the Intel FPGA board embedded with the Nios II soft core processor as shown in fig 1. Universal asynchronous Receiver Transmitter is a component used for communication between serial input and serial output devices where the parallel data is converted to serial data. It is generally used for short distance communication and for low - cost data transfer between a computer and its peripherals. The design is implemented on the Altera Development board. This design implements dual UART for communication within the FPGA.

Nios II processor is a soft embedded core in the Altera Development boards. This processor adopts Reduced Instruction Set Computer (RISC) architecture with length of 32 - bits. Nios II architecture uses separate instruction and data buses which is referred to as Harvard architecture. The Hardware Abstraction Layer (HAL) provides a simple device driver interface for programs to connect to the underlying hardware. It serves as a device driver package for the Nios II processor, providing a consistent interface to the peripherals in the system. The library drivers present in the HAL system enables the designer to access the UART core using the ANSI C standard library functions. [2]

The BLE module SI MGM series is interfaced with the FPGA. A free source Serial Terminal app is used over the Android device to send the data to the PC using UART communication

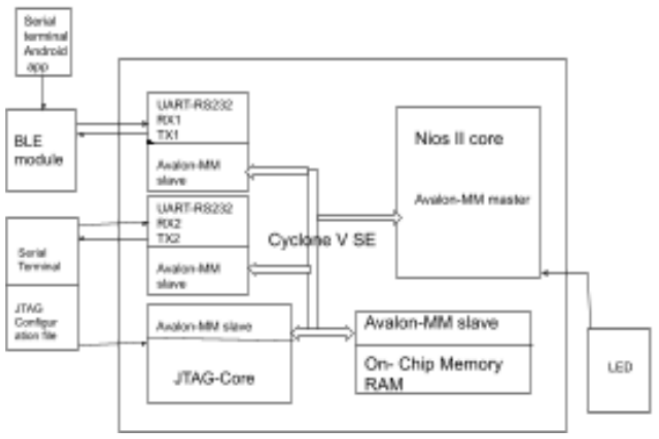


Figure 1: Block diagram of interfacing BLE on the FPGA.

2. Implementation

The dual UART RS - 232 is implemented on the FPGA using the Platform Based Design (PBD) from the IPs available in the Quartus Prime Lite software. The IPs used in the design include UART RS - 232 (UART_0, UART_1), JTAG_UART, On - chip memory, Parallel I/O (LED).

a) UART RS - 232 Core

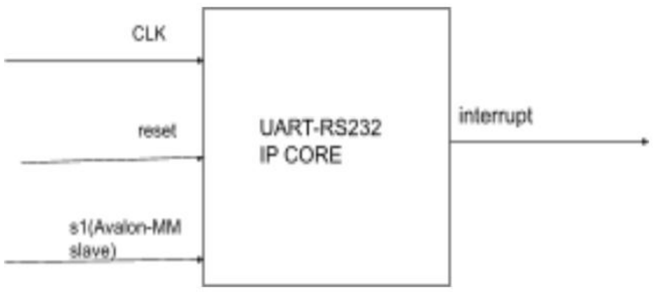


Figure 2: UART - RS232 IP design

The UART core provides a method of implementation for serial communication which allows streaming of characters between an embedded system to an Intel FPGA or the external device. The core implements RS - 232 protocol and allows modification of the baud rate, parity and data bits. The core provides Avalon Memory - mapped (MM) slave interface that communicates with the Avalon Memory -

Mapped master peripherals by reading and writing control and data registers.

The implementation of the UART core is done using the RS - 232 asynchronous transmit and receive logic. The data is sent and received via RXD and TXD ports. The UART consists of an 8 - bit data shift receiver and transmitted register corresponding to the 8 - bit holding register. These two registers provide double buffering. The master peripheral can write new data into the holding registers while previously written characters can be shifted out to the register.

The internal baud rate of the UART core is derived from the Avalon - MM clock input. The internal baud rate can be generated by using a clock divider. The reconfigurable parameters of the UART core are set to as follows

Table 1: Values of the Parameters used in UART - RS232

Parameter	Value
Data bit	8 - bits
Parity	None
Stop bit	1 bit
Flow Control	None
Baud Rate	115200

Two UART - RS232 cores are used in the design to implement dual UART. UART_0 occupies the address of 0x0004_1020 to 0x0004_103f while UART_1 occupies the address of 0x0004_1000 to 0x0004_101f for the Avalon - MM slave on the Nios II processor [3].

b) JTAG UART Core

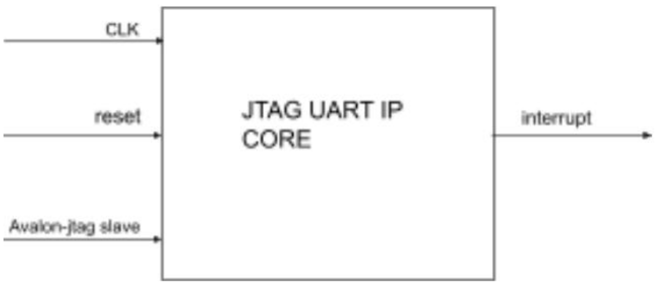


Figure 3: JTAG UART IP design

The JTAG UART core with Avalon interface implements a method of communication which allows streaming of characters between host PC and Platform Designer System of the FPGA. The JTAG UART uses the JTAG circuitry built in the Intel FPGA through any Intel FPGA download cable. Software support in the Nios II processor is provided in the Hardware Abstraction Layer (HAL) system library, allowing the software to access the core using ANSI C standard library stdio. h routines.

The JTAG UART core also provides an Avalon slave interface to the JTAG circuitry on the Intel FPGA. The interface of the JTAG UART consists of 32 - bits data and control registers that are accessed via Avalon slave port. The Nios II processor (Avalon Master) accesses the registers, controls the core and transfers the data over the JTAG connection. This core provides an active - high interrupt output. This allows the core to request an interrupt when read data is available or when the write FIFO is available to write the data. The core provides bidirectional FIFOs to improve bandwidth over JTAG communication. The FIFO depth is parameterizable to accommodate the available on - chip memory.

Intel FPGAs contain in - built JTAG circuitry between the JTAG pins and the logic inside the device. The Intel Quartus Prime software generates the JTAG UART logic. Manual design is not necessary to connect the JTAG circuitry inside the device.

The Avalon - MM slave of the JTAG UART uses the address 0x0004_1050 to 0x0004_1057 from the Nios II processor [3].

c) On - Chip Memory

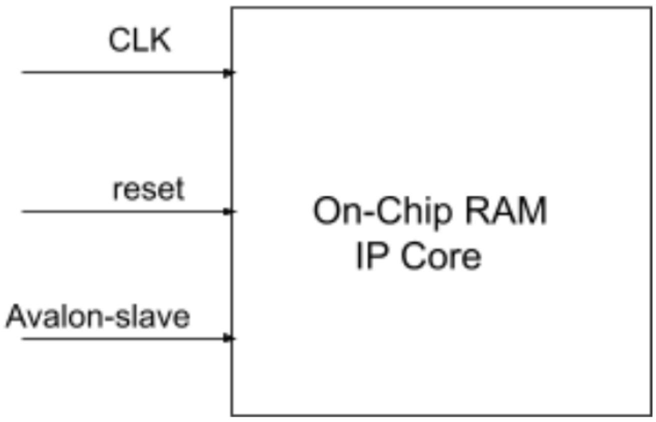


Figure 4: On - chip memory RAM

Intel FPGAs contain inbuilt On - chip memory that can be used as RAM or ROM. This memory has fast access time in comparison to the off - chip memory. On - chip memory can be automatically instantiated within the Platform based design and certain memory blocks have initialized content when the block powers up. This memory is used to store the data constants and also to store the processor boot code. Nios II /e core processor does not include instruction cache and data cache. The OCRAM provides the memory required by the processor to store the data constants [3].

d) Nios II core

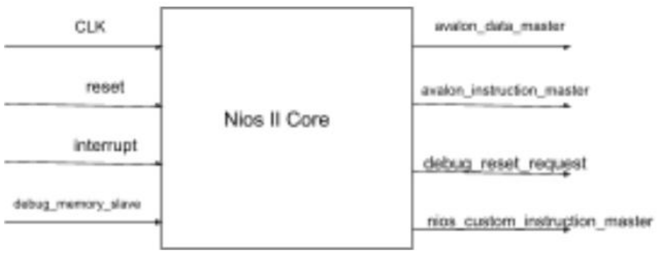


Figure 5: Nios II IP design

Nios II/e version is used in the above implementation. This is the economical version which aims at using minimum FPGA resources. The Nios II along Avalon interface provides a connection of the master to the other embedded IPs. The processor controls the data transfer via 8 - bit data bus of the Avalon - MM master to UART - RS232, JTAG -

UART and PIO. The 8 - bit instruction_master signal (Avalon - master) communicates with the Avalon - MM slave of the on - chip memory RAM. The core provides an interrupt receiver which receives the interrupt requests from other cores [4].

e) Parallel I/O

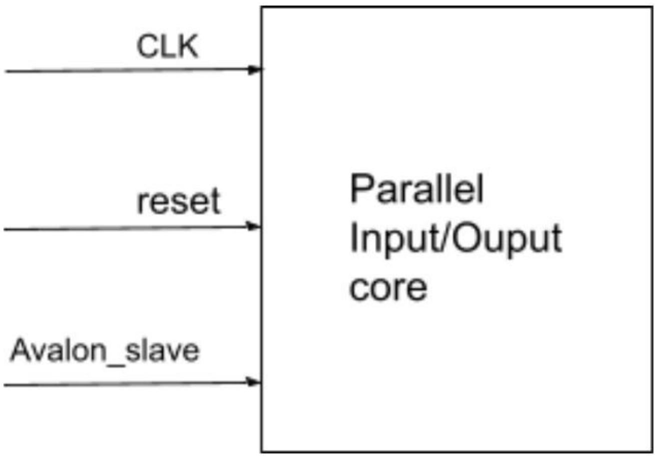


Figure 6: PIO IP design

The parallel input/output core along with Avalon interface provides connection between Avalon Memory - mapped slave port and general purpose input output ports. The input output pins connect to the pins that are connected to the devices external to the FPGA. The PIO core allows display LEDs for the verilog code which is executed [3].

A. Hardware Design

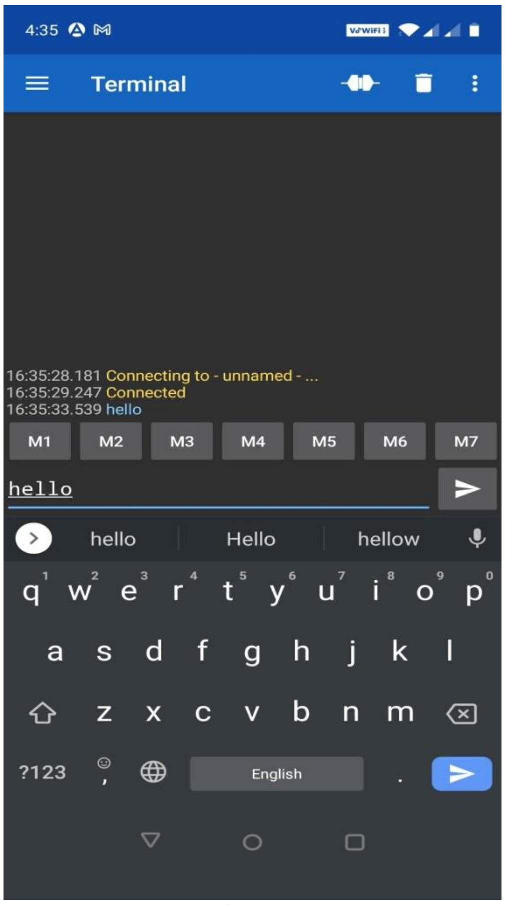
The Platform design is programmed on the Development board in Active Serial mode. The SRAM object file (. sof) created after the compilation process is converted to JTAG Indirect Configuration file (.jic). The MSEL [4:0] switch on the board is set to "10010" to use flash as the configuration device. The configuration file (.jic) is downloaded into the serial configuration device (EPCS128). The configuration file in EPCS128 is retained even when the Development board is restarted. A Serial Flash Loader (SFL) Megafunction is used to program the serial configuration via JTAG interface.

B. Software Design

The UART design is implemented using the Nios II Software Build Tools (SBT). The UART .c code implements the main UART function for read/write operations. The file contains the receiver (RX) and transmitted (TX) buffers, data transmission and collection and registers check procedures. The Main .c file contains the main C function.

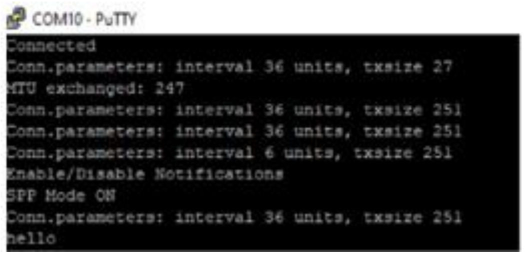
3. Result and Conclusion

The test results were verified on the Nios II console and on the Putty serial terminal software. The BLE module was connected to the RX and TX pins of UART_0 and the PC was connected to the RX and TX pins of UART_1 via TTL to UART converter. Serial Bluetooth terminal App is used on the Android device to send data to the BLE module which is received on the Putty serial terminal software on the PC via the two UARTs.



```
16:35:28.181 Connecting to - unnamed - ...
16:35:29.247 Connected
16:35:33.539 hello
```

Figure 7 (a): Data sent from the Serial Terminal App



```
Connected
Conn.parameters: interval 36 units, txsize 27
MTU exchanged: 247
Conn.parameters: interval 36 units, txsize 251
Conn.parameters: interval 36 units, txsize 251
Conn.parameters: interval 6 units, txsize 251
Enable/Disable Notifications
SPP Mode ON
Conn.parameters: interval 36 units, txsize 251
hello
```

Figure 7 (b): Data at the Serial Terminal (Putty) on the PC

References

[1] Kang Eun Jeon, James She, Perm Soonsawad, and Pai Chet N"BLE Beacons for Internet of Things Applications: Survey, Challenges and Opportunities"
[2] Nios® II Software Developer

Handbook
[3] Embedded Peripherals IP User Guide
[4] Nios II Classic Processor Reference Guide