Design of Universal Serial Bus for Low Cost Field Programmable Gate Array

V.Arun , ASSISTANT PROFESSOR,MLRIT , DUNDIGAL , HYDERABAD
D.Laxma Reddy, ASSISTANT PROFESSOR, MLRIT, DUNDIGAL, HYDERABAD

ABSTRACT:

The objective of the paper is to develop a USB device core, comprising of a Transmitter and Receiver on FPGA integrated circuits using VHDL as the coding language. Higher speed and quality requests for communications between digital systems and devices introduced the need of developing a new standard to allow replacement of the traditional series and parallel interfaces, thus allowing not only an increased speed in data transfer, but also a chance to connect different data receivers and transmitters over one common structure, bus type. This would allow the development of a data communication system that would be simple to manage but also very efficient. The answer to this challenge, the well known USB standard has become, during the last years, the basic communication module for digital systems, replacing, with many advantages, the previous and traditional series and parallel communication interfaces. USB characteristics include low cost, easiness of use, and simple construction. Thus, USB standard must be considered as something to be used in any kind of device requiring data communication with other systems or devices.

1. INTRODUCTION

The main reason why new interfaces don□t appear very often is that existing interfaces have the advantage of all of the peripherals that users don□t want to scrap. By choosing compatibility with the existing Centronics parallel interface and RS-232 serial-port interface, the developers of the original IBM PC sped up the design process and enabled users to connect to printers and modems already in the market. These interfaces proved serviceable for close to two decades. But as computers became more powerful and the number and kinds of peripherals Increased, the older interfaces became a bottleneck of slow communications with limited options for expansion. This is the situation that prompted the development of USB. There has been two versions of USB standards previously USB1.0 and USB1.1 (September 1998) these versions supported only two speeds Low and Full speed. April 2000 saw the release of USB 2.0 which added the option to use high speed.

2. BENEFITS OF USB

Ease of Use: Ease of use was a major design goal for USB, and the result is an interface that's a pleasure to use for many reasons:

One interface for many devices USB is versatile enough to be usable with a variety of peripheral types. Instead of having a different connector type and supporting hardware for each peripheral, one interface serves many.

Automatic configuration When a user connects a USB peripheral to a PC, Windows detects the peripheral and loads the appropriate software driver. The first time the peripheral connects, Windows may prompt the user to insert a disk with driver software, but other than that, installation is automatic There□s no need to restart the system before using the peripheral.

Easy to connect With USB, there□s no need to open the computer□s enclosure to add an expansion card for each peripheral. A typical PC has four or more USB ports. You can expand the number of ports by adding hubs with additional ports.

Easy cables USB cable connectors are keyed so you can□t plug them in wrong. A cable segment can be as long as 5 meters. With hubs, a peripheral can be as far as 30 meters from its host PC. USB connectors are small and compact in contrast to typical RS-232 and parallel connectors. To ensure reliable operation, the USB specification includes detailed requirements that all cables and connectors must meet.

Hot pluggable. You can connect and disconnect a USB peripheral whenever you want, whether or not the system and peripheral are powered, without damaging the PC or device. The operating system detects when a peripheral is attached and readies it for use.

No user settings. USB peripherals don□t have user-selectable settings such as port addresses and interrupt-request (IRQ) lines so there are no jumpers to set or configuration utilities to run.

No power supply required (sometimes): The USB interface includes power-supply and ground lines that provide a nominal +5V from the computer's or hub's power supply. A peripheral that requires up to 500 mill amperes can draw all of its power from the bus instead of having to provide a power supply. In contrast, peripherals that use other interfaces may have to choose between including a power

3. DATA ENCODING

All data on USB is encoded. The encoding format is called Non-Return to Zero Inverted (NRZI).

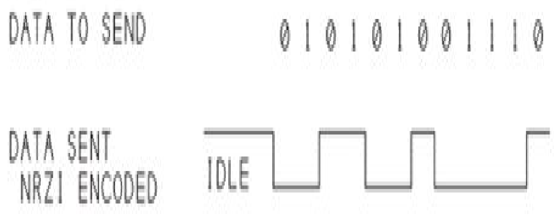


Figure: Example of NRZI Encoding

Instead of defining logic 0s and 1s as voltages, NRZI encoding defines logic 0 as a voltage change, and logic 1 as a voltage that remains the same. Figure 2.6 shows a example. Each logic 0 results in a change from the previous state. Each logic 1 result in no change in the voltages. The bits transmit least-significant-bit (LSB) first.

4.USB PROTOCOLS

Unlike RS-232 or similar serial interfaces where the format of data being sent is not defined, USB is made up of several layers of protocols.

Each USB transaction consists of a Token Packet (Header defining what it expects to follow), Optional Data Packet, (Containing the payload) Status Packet (Used to acknowledge transactions and to provide a means of error correction) USB is a host centric bus. The host initiates all transactions. The first packet, also called a token is generated by the host to describe what is to follow and whether the data transaction will be a read or write and what the device's address and designated endpoint is. The next packet is generally a data packet carrying the payload and is followed by a handshaking packet, reporting if the data or token was received successfully, or if the endpoint is stalled or not available to accept data.

5. COMMON USB PACKET FIELDS

Data on the USB bus is transmitted LSB bit first. USB packets consist of the following fields [7, 3]

Sync: All packets must start with a sync field. The sync field is 8 bits long, which is used to synchronize the clock of the receiver with the transmitter. The last two bits indicate where the PID fields starts.

PID: PID stands for Packet ID. This field is used to identify the type of packet that is being sent. The following table shows the possible values.

ADDR: The address field specifies which device the packet is designated for. Being 7 bits in length allows for 127 devices to be supported. Address 0 is not valid, as any device which is not yet assigned an address must respond to packets sent to address zero

ENDP: The endpoint field is made up of 4 bits, allowing 16 possible endpoints. Low speed devices, however can only have 2 endpoint additional addresses on top of the default pipe. (4 Endpoints Max)

CRC: Cyclic Redundancy Checks are performed on the data within the packet payload. All token packets have a 5 bit CRC while data packets have a 16 bit CRC.

EOP: End of packet is signaled by a Single Ended Zero (SE0) for approximately 2 bit times followed by a J for 1 bit time.

6. TYPES OF TRANSFERS

USB is designed to handle many types of peripherals with varying requirements for transfer rate, response time, and error correcting. There are four types of data transfers each handling different needs and a device can support the transfer types that is best suited for its purpose.

The four transfer types are

CONTROL TRANSFERS are the only type that has functions defined by the USB specification. Control transfers enable the host to read information about a device, set a device□s address, and select configurations and other settings. Control transfers may also send vendor-specific requests that send and receive data for any purpose. All USB devices must support control transfers.

BULK TRANSFERS are intended for situations where the rate of transfer isn□t critical, such as sending a file to a printer, receiving data from a scanner, or accessing files on a drive. For these applications, quick transfers are nice but the data can wait if necessary. If the bus is very busy, bulk transfers are delayed, but if the bus is otherwise idle, bulk transfers are very fast. Only full- and high-speed devices can do bulk transfers.

INTERRUPT TRANSFERS are for devices that must receive the host□s or device□s attention periodically. Other than control transfers, interrupt transfers are the only way that low-speed devices can transfer data. Keyboards and mice use interrupt transfers to send key press and mouse-movement data. Interrupt transfers can use any speed.

ISOCHRONOUS TRANSFERS have guaranteed delivery time but no error correcting. Data that might use isochronous transfers includes audio or video to be played in real time. Isochronous is the only transfer type that doesn't support automatic re-transmitting of data received with errors, so occasional errors must be acceptable. Only full- and high-speed devices can do isochronous transfers.

7. USB TRANSMITTER

USB TRANSMITTER STATE MACHINE

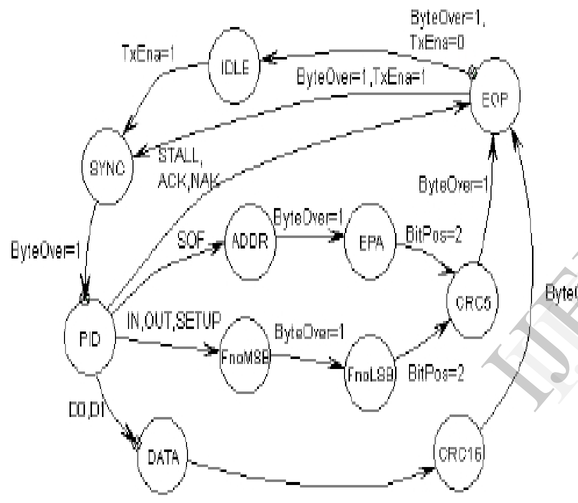


Figure: State Machine Diagram of USB Transmitter

The state diagram shown above depicts the operation of the State Machine Controller. The Transmitter State Machine Controller in the USB is responsible for the transmission of the appropriate packets, with their appropriate CRCs. The USB Transmitter transmits four kinds of packets. Each of the packets has different formats. The SYNC and PID fields are common in all the packets

USB Transmitter:

Transmitter Block Diagram:

The block diagram shown below is that of the USB Transmitter. The inputs of the transmitter are Data, TxEna, Addr and PID Type. All the inputs go to the State Machine Controller, and the Register Array. TxEna signal initiates the state machine to start the transmission. The outputs of the state machine enable the respective fields of the packets to be transmitted. For example, if Sync Ena is the output, the SYNC field is transmitted from the Register Array. When the PID is to be transmitted, the PID type is checked, and is transmitted with its compliment. The PID determines the type of packet being transmitted. Thus, the packets are transmitted accordingly from the Register array.

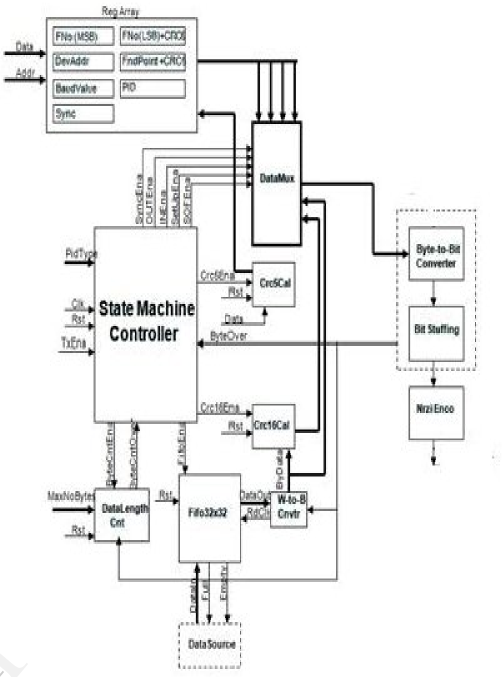


Figure : Transmitter Block Diagram

8.USB RECEIVER

STATE DIAGRAM OF TOKEN PACKET

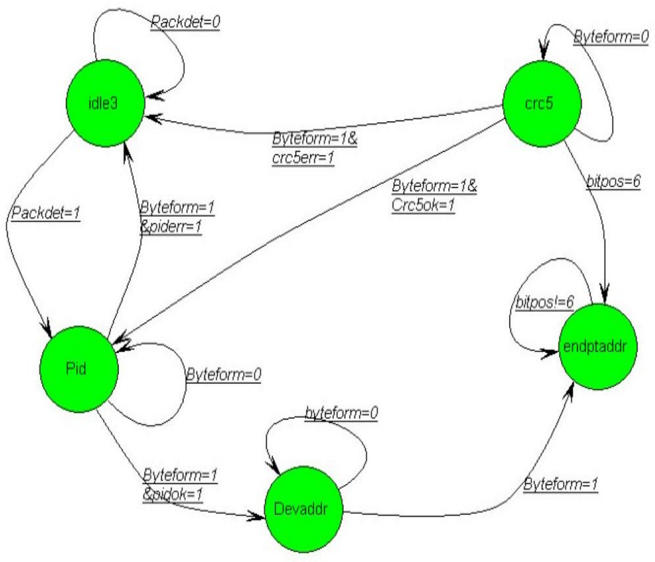


Figure 5.1 State Diagram of Token Packet

USB –Rx receives a Token Packet which has 5 states:

- Idle
- PID
- DevAddr
- EndPtAddr
- CRC-5

State Diagram of Data Packet

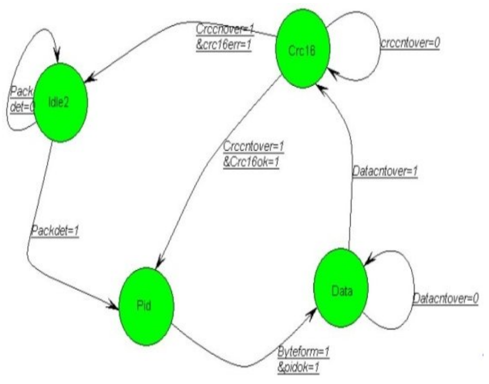
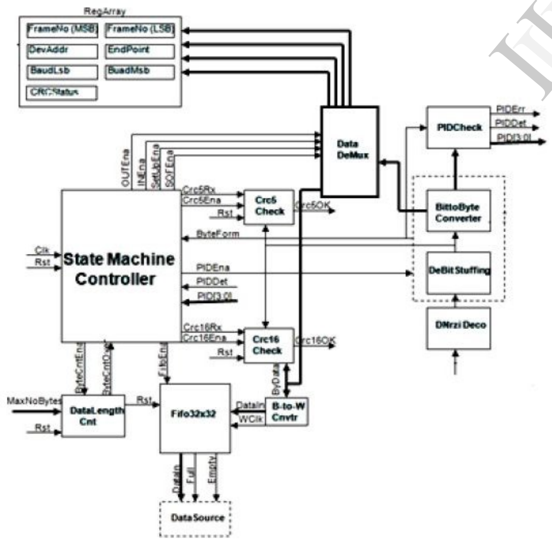


Figure: State Diagram of Data Packet

USB –Rx receives a data packet which has 4 states:

- Idle
- PID
- DATA
- CRC-16

USB Receiver:



-

Figure: USB Receiver Block Diagram

9. INTRODUCTION TO PROGRAMMABLE LOGIC DEVICES

Programmable Devices are integrated circuits which can be programmed "in house" or on the field. The design and implementation of an application on these devices can be achieved with the help of software tools. Hardware descriptive languages such as Verilog and VHDL are widely used for this purpose. The codes written in these languages can also be synthesized using a third party Electronic Design and Automation tool (EDA) tool or the software tool provided by the vendor. With the help of these tools it is also possible to optimize the design for speed or space.

FIELD PROGRAMMABLE GATE ARRAY (FPGA)

Traditional gate arrays contain a number of building blocks or primitive cells etched on a single silicon substrate. The connections between cells are permanent and made later. These are non reprogrammable high-density devices containing about 5 millions gates. The FPGAs have similar structure to gate arrays however they have programmable elements.

The programmable cell is called Logic Element (LE) in case of Altera device and Configurable Logic Block (CLB) in Xilinx devices. FPGA use the Complementary Metal Oxide Semiconductor SRAM technology and are thus reset at power.

10. RESULTS

SIMULATION RESULTS:

The results of the USB Transmitter and Receiver explained before are discussed here.

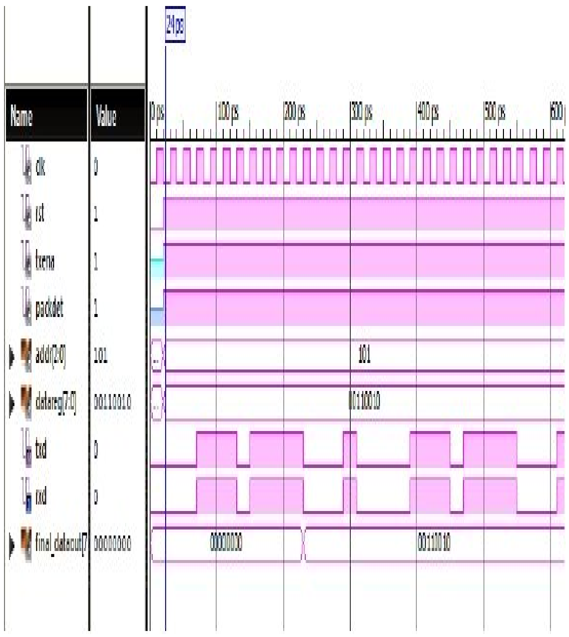


Figure: Simulation Result of Transmitter.

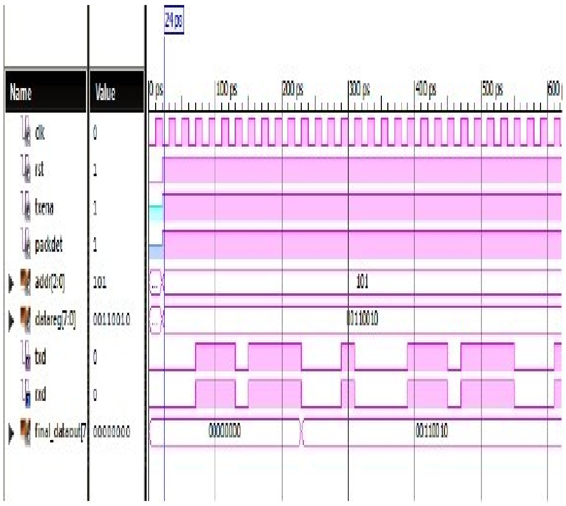


Figure: Simulation Result of Receiver.

11. CONCLUSION

In this paper a USB device core i.e., a Transmitter and Receiver performing the USB communication is designed. The design was done using VHDL as the coding language. The tool used was Xilinx ISE 10.1. The FPGA board used was Spartan 3E. The transmitter module could only be implemented on the FPGA.The receiver module was done up to synthesis level.

12.REFERENCES

BOOKS:

[1] Craig. Peacock "USB in a Nutshell. Making Sense of the USB Standard"
[2] Don Anderson, "Universal Serial Bus System Architecture", 2nd. Edition, Mindshare Inc. 2001. ISBN- 13: 978-0201309751.
[3]Douglas L. Perry "VHDL: Programming by Example" 4th edition, McGraw- Hill Professional, ISBN 978-0071400701.
[4]Elio A. A. De Maria, Edgardo Gho, Carlos E. Maidana, Fernando I Szklanny, Hugo R. Tantignone: A LOW COST FPGA BASED USB DEVICE CORE in programmable logic, 2008, 4th southern conference on 26 – 28 March 2008 pages: 149 – 154, 2008.
[5]Jan Axelson "USB Complete – The Developers Guide", Fourth Edition, Lakeview Research LLC Madison, WI 53704.
[6]John Hyde "USB Design by Example A Practical Guide to Building I/O Devices", 2nd Edition. Intel University press. ISBN 0–471–37048-7.
[7]\J Bhasker "A VHDL Primer" Third Edition, Pearson Education, ISBN 81- 7808-016-8.
[8]Pavel Kubalik and Jiri Bucek: FPGA IMPLEMENTATION OF USB 1.1 DEVICE CORE 2003.

WEBSITES:

www.beyondlogic.org/usbnutshell
www.usb.org.
www.xilinx.com

AUTHOR'S BIOGRAPHY

Mr.ARUN did B.Tech in Electronics and communication in JNTU Hyderabad , M.Tech in Embedded systems from, JNTU Hyderabad. His interested areas in Embedded systems and image processing.

Mr. Laxma Reddy did B.Tech in Electronics and communication Engineering from Mother Theresa College of Engineering ,Affiliated to jntuh, Hyderabad and M.Tech in computer communications from RRS college of engineering and Technology, JNTU Hyderabad. His interested areas in communications and image processing.