International Journal of Advances in Electronics Engineering
Vol:1 Issue:1 ISSN 2278 - 215X

FPGA IMPLEMENTATION OF USB 2.0 RECEIVER PROTOCOL

Mr.N.S.Panchbudhe[1], Prof.S.S.Shriramwar[2], Mrs.G.A.Jichkar3
1Lecturer, Dept of E.C.E, *S.V.S.S. College of Engineering, Nagpur, India*
nilesh.panchbudhe@ gmail.com, 9325552026
2 *Assi.Professor, Dept of E.C.E, Priydarshani College of Engineering, Nagpur, India*
sshriramwar@yahoo.com
3 Lecturers, Dept of E.C.E. S.R.P.*College of Engineering, Nagpur, India*
geeta_0275@yahoo.com

Abstract: - The Universal Serial Bus (USB2.0) is support data exchange between a host computer and a wide range of simultaneously accessible peripheral like Mouse, keyboard, Digital camera, Printer, scanner etc..The USB2.0 supported three types of data transmission rates, those are operated Low speed (1.5MHZ), High speed (12MHZ) & Full speed (480MHZ). In this paper our coverage is up to implementation of* USB 2.0 Receiver protocol *on FPGA kit [SPARTAN-II XC2S200]. The FPGA design of* USB 2.0 Receiver protocol *provides an interface for 'system on chip 'designer to connect UBS bus. This will save the design time and the time required for debugging and testing a USB controller. The* USB 2.0 Receiver protocol *has been developed in VHDL which is a widely used hardware description language and is supported by the major FPGA designer such as Xilinx and Altera. The VHDL code for the* USB 2.0 Receiver protocol *is synthesizable onto a Xilinx Spartan II FPGA and is synthesizable onto other types of FPGAs with minimal modification.

Key words – FPGA, USB Protocol, Transreceiver, PID, SOF, CRC, Bit Stuffer etc.

I. INTRODUCTION

The trend in electronics is to make devices as small as possible and to get them to market quickly. This trend has caused designers to focus on FPGAs rather than the traditional PCBs. This trend to make devices smaller and cheaper combined with the rising costs of labor to solder components onto PCBs is resulting in developments of "Systems on a Chip" such as FPGAs. The USB standard was developed to overcome the shortcomings of older interfaces to peripheral devices for PCs. The standard makes interfacing to the PC extremely easy for the end user and life more complicated for the peripheral designer. The USB 2.0 understands the USB protocol and is capable of carrying out transactions on behalf of the device. The Universal Serial Bus (USB) solves the inadequacies of traditional interfaces for peripheral devices to PCs. Some of the problems it is trying to solve are:

- The limited number of ports on PCs. Before USB most computers came with one printer port and two serial ports.
- Several modern peripherals require high-speed connections to the PC.
- Attaching new peripheral for the first time involved manual configuring.

The Universal Serial Bus gives a single, standardized, easy-to-use way to connect up to 127 devices to a computer. Each device can consume up to a maximum of 6 Mb per second of bandwidth, which is fast enough for the vast majority of peripheral devices that most people want to connect to their machines.

The USB supports the three speeds of operation.

a. Low speed =>1.5 Mbps=>Keyboard, mouse
b. Full speed=>12Mbps=> pots broadband, audio
c. High speed=>480Mbps=>video,storage,image

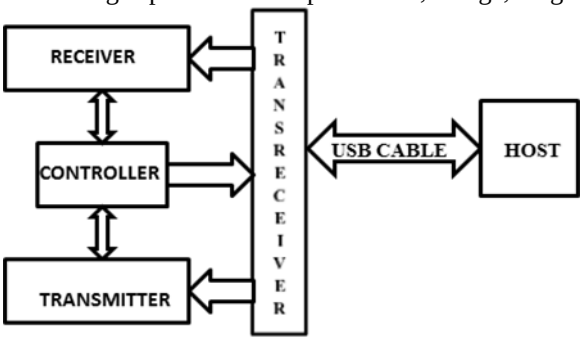

Fig.1 block diagram of USB 2.0

II. USB PROTOCOL

There are five types of packet & they are completely responsible for communication between Host & device, they are

A.TOKEN PACKETS

Figure 2 shows the field formats for a token packet. A token consists of a PID, specifying either IN, OUT, or SETUP packet type; and ADDR and ENDP fields. For OUT and SETUP transactions, the address and endpoint fields uniquely identify the endpoint that will receive the subsequent Data packet. For IN transactions, these fields uniquely identify which endpoint should transmit a Data packet. Only the host can issue token packets. IN PIDs define a Data transaction from a function to the host. OUT and SETUP PIDs define Data transactions from the host to a function.

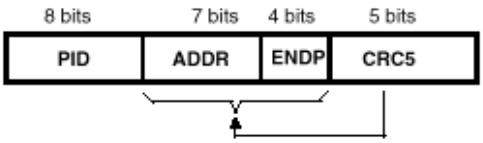

Fig. 2 Token packets

Token packets have a five-bit CRC that covers the address and endpoint fields as shown above. The CRC does not cover the PID, which has its own check field. Token and SOF packets are delimited by an EOP after three bytes of packet field data.

Vol: 1 Issue 1 ISSN 2278 – 215X

If a packet decodes as an otherwise valid token or SOF but does not terminate with an EOP after three bytes, it must be considered invalid and ignored by the receiver.

B. START-OF-FRAME PACKETS

Start-of-Frame (SOF) packets are issued by the host at a nominal rate of once every 1.00ms _ 005ms.SOF packets consist of a PID indicating packet type followed by an 11-bit frame number field as illustrated in fig..3

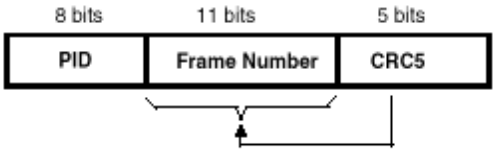

Fig.3Start-of-Frame (SOF) packets

The SOF token comprises the token-only transaction that distributes an SOF marker and accompanying frame number at precisely timed intervals corresponding to the start of each frame. All full-speed functions, including hubs, receive the SOF packet. The SOF token does not cause any receiving function to generate a return packet; therefore, SOF delivery to any given function cannot be guaranteed. The SOF packet delivers two pieces of timing information. A function is informed that an SOF has occurred when it detects the SOF PID. Frame timing sensitive functions, which do not need to keep track of frame number (e.g., a hub), need only decode the SOF PID; they can ignore the frame number and its CRC. If a function needs to track frame number, it must comprehend both the PID and the time stamp. Full-speed devices that have no particular need for bus timing information may ignore the SOF packet.

C. DATA PACKETS

A data packet consists of a PID, a data field containing zero or more bytes of data, and a CRC as shown in Figure 4. There are two types of data packets, identified by differing PIDs: DATA0 and DATA1. Two data packet PIDs are defined to support data toggle synchronization.

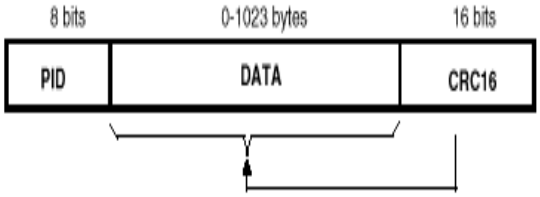

Fig. 4 Data packet

Data must always be sent in integral numbers of bytes. The data CRC is computed over only the data field in the packet and does not include the PID, which has its own check field.

D. HANDSHAKE PACKETS

Handshake packets, as shown in Figure 5, consist of only a PID. Handshake packets are used to report the status of a data transaction and can return values indicating successful reception of data, command acceptance or rejection, flow control, and halt conditions. Only transaction types that support flow control can return handshakes. Handshakes are always returned in the handshake phase of a transaction and may be returned, instead of data, in the data phase. Handshake packets are delimited by an EOP after one byte of packet field. If a packet decodes as an otherwise valid handshake but does not terminate with an EOP after one byte, it must be considered invalid and ignored by the receiver.

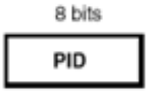

Fig.5 Handshake packet

There are three types of handshake packets:

I. . ACK indicates that the data packet was received without bit stuff or CRC errors over the data field and that the data PID was received correctly. ACK may be issued either when sequence bits match and the receiver can accept data or when sequence bits mismatch and the sender and receiver must resynchronize only in transactions in which data has been transmitted and where a handshake is expected. ACK can be returned by the host for IN transactions and by a function for OUT or SETUP transactions.

II. NAK indicates that a function was unable to accept data from the host (OUT) or that a function has no data to transmit to the host (IN). NAK can only be returned by functions in the data phase of IN transactions or the handshake phase of OUT transactions. The host can never issue NAK. NAK is used for flow control purposes to indicate that a function is temporarily unable to transmit or receive data, but will eventually be able to do so without need of host intervention.

III. STALL is returned by a function in response to an IN token or after the data phase of an OUT transaction. STALL indicates that a function is unable to transmit or receive data, or that a control pipe request is not supported. The host is not permitted to return a STALL under any condition.

IV. DESIGN ASPECTS

The present USB 2.0 has been designed by using embedded system and it is on mother board of computer, In this project our aim is to design USB 2.0 using FPGA kit .

A.RECEIVER

The Bloch diagram of the USB 2.0 Receivers is shown in Figure 6.The Receiver module has been implemented by considering the following specifications**.**

- Enable the Receiver when start of packet (sync) sequence" 01010100' is received.
- The data should be decoded using NON Return to Zero Inverted decoding technique.
- Stuffed bit is removed after checking the six consecutive '1's occur in the data stream then a zero to be removed.
- Serial to Parallel conversion.
- By checking the correct PID controller will decide the type of data is received.
- Receiver checks the 7-bit address of device.

Vol:1 Issue:1 ISSN 2278 - 215X

- Receiver check the 4-bit end point address of host controller

- By checking 5-bit CRC & 16-bit CRC of packet, host can do error checking.
- Allowing controller to decide what type of byte to received

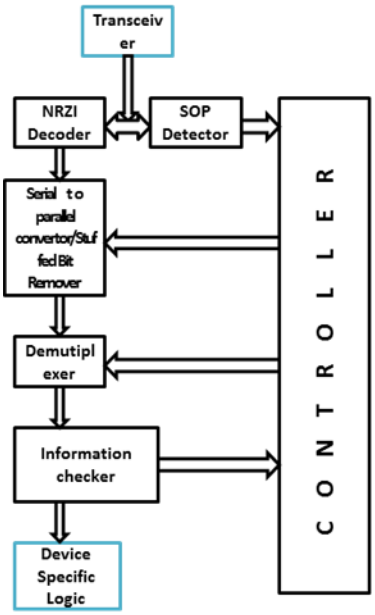

Fig 6 Receiver block diagram of USB 2.0

. IV. SIMULATED RESULTS OF RECEIVER

The Fig.7 shows the simulation result of USB 2.0 Receiver, which is designed using VHDL as stated above and simulated within the Xlink 9.1 environment. At the start of data transfer the rst signal is goes to high and it will reset the Receiver. After resetting the Receiver the controller give enable signal to Receiver to enable ,then it start normal operation by sending SYNC, PID, ,ENDP,DATA,CRC.

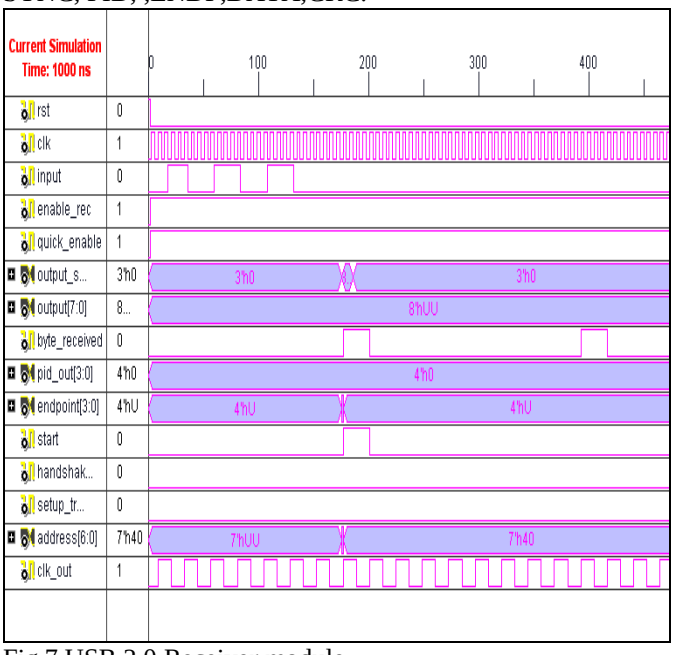

Fig.7 USB 2.0 Receiver module

V. FPGA IMPLEMENTATION

The top order module of USB 2.0 Receiver is synthesized within the Xilinx 9.1 ISE software tool and it is programmed to the targeted SPARTAN 2 family of FPGA Device. The various levels Implementation such as Synthesis report, RTL View, Place and Route Report and Device Programming has been explained and visualized in the following sub sections.

A. SYNTHESIS REPORT

The Table 1 shows the synthesis Summary of top order module of USB 2.0 Receiver. It is observed from the Table 1 that the total equivalent gate count required for this design is 848 gates. From the same table we can get the information about the target FPGA device utilization.

W Project Status			
Project File:	w.ise	Current State:	Synthesized
Module Name:	Receiver	• Errors:	No Errors
Target Device:	xc2s15-6cs144	• Warnings:	62 Warnings (62 new, 0 filtered)
Product Version:	ISE 9.2i	• Updated:	Thu Dec 30 17:42:17 2010

W Partition Summary
No partition information was found.

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	154	192	80%
Number of Slice Flip Flops	128	384	33%
Number of 4 input LUTs	284	384	73%
Number of bonded IOBs	36	86	41%
Number of GCLKs	2	4	50%

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Thu Dec 30 17:33:33 2010	0	62 Warnings (62 new, 0 filtered)	5 Infos (5 new, 0 filtered)
Translation					

Table 1 Synthesis Summery

B. RTL VIEW

This section gives the visualization of Resister Transistor Logic (RTL) views in the form of schematic and
Net list diagrams which are shown in Figure 8 and Figure 9 respectively. Figure 8 which gives RTL schematic diagram reveals the pin diagram of Top order module with the required specified notes whereas Figure 9 reveals the Gate level logic diagram of Top order module with the required
Input and output ports (Net list view).

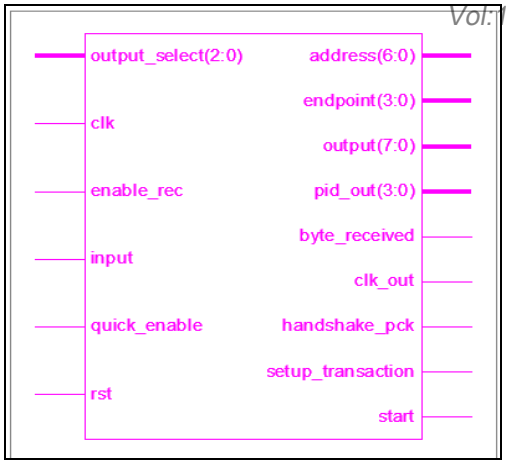

Fig.8 RTL schematic diagram

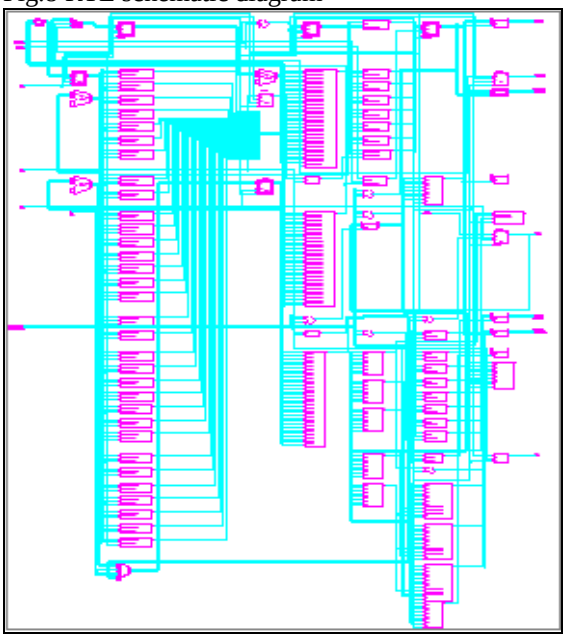

Figure 9: RTL netlist view

C. PLACE ROUT REPORT

This section concentrates on target FPGA device utilization summary which reveals the information required for proper layout at the level of manufacturing in the form of Place and Route report. Further it gives the timing synchronization of CPU with the REAL time environment.

Device utilization summary:

Selected Device: 2s15cs144-6

Number of Slices:	154 out of	192	80%
Number of Slice Flip Flops:	128 out of	384	33%
Number of 4 input LUTs:	284 out of	384	73%
Number of IOs:	36		
Number of bonded IOBs:	36 out of	86	41%
IOB Flip Flops:	25		
Number of GCLKs:	2 out of	4	50%.

DEVICE PROGRAMMING

After successful process of synthesis the Target device xc2s15 of Spartan2 is connected to the system through printer port. The pin assignment is specified in the User Constraint File (UCF). The functional Verification is carried out by using a pattern generator.

VI. CONCLUSION

The goal of the project is FPGA Implementation of USB 2.0 Receiver Protocol is achieved & the target device used for this project is SPARTAN II, where our Receiver has required only 192Slices, 128Slice Flip-Flop, 284 input LUTs, 36 IOs, 36 bounded IOs & 2 GCLKs. The resulting system works in simulation.

VII. FUTURE SCOPE

The USB 2.0 Receiver has been implemented for 8-bit, but it can also be extended to 16- bit USB 2.0 Receiver.

VIII. APPLICATIONS

The USB2.0 has been developed into a common code which can be used for developing the complete USB device stack. Some of the Low speed and High speed USB devices, which are presently available in the market, are:

1. Optical Mouse
2. Key Board
3. Printer
4. Scanner
5. Joy Stick
6. Memory Stick
7. Flash Memory
8. Mobiles
9. Video cameras.

IX. REFERENCES

[1]Universal serial bus specification-2000 Compaq,hewlett-packard,intel,Lucent,microsoft, nec, Philips

[2] FPGA Implementation of USB Transceiver Macrocell Interface with USB2.0 Specifications.Babulu, K. Rajan, K.S. Coll. of Eng., Dept. of E.C.E, Jawaharlal Nehru Technol. Univ., Kakinada;Emerging Trends in Engineering and Technology, 2008. ICETET'08. First International Conference

[3] Design of reusable software for USB host driver in embedded system

2010 International Conference on Computing, Control and Industrial Engineering

Gaohua Liao, Quanguo Lu, Weizhong Zhang Nanchang Institute of Technology Nanchang, China

[4] Zainalabedin Navabi "Vhdl Analysis and Modeling of Digital Systems" 2nd edition, McGraw- Hill, Hardcover,Published January 1998

[5] William Stallings, "Data and Computer Communications" ,McGraw-Hill Publications.

[6]Andrew S.Tannenbaum,"Computer Networks", Pearson publications.

[7] Charles H Roth "Digital system using VHDL".2nd edition, Thomson publication

Vol:1 Issue:1 ISSN 2278 - 215X
[8] Stephen Brown, Zvonko Vranesic "Fundamentals Digital logic with VHDL design". 2nd edition, McGraw-Hill, Hardcover, Published July 2004
[9] [www.usb.org/developers]