The Design of a USB Device Controller IYOYOYO

Tomoaki Kouyama† Hibiki Nano†† Chiaki Kon† Naohiko Shimizu††

†Graduate School of Engineering, Tokai University
††Dept. Communications Engineering, Tokai University
Hiratsuka, Kanagawa 259-1292
E-mail: {2aepm025, 9jet1239, 1aepm024, nshimizu}@keyaki.cc.u-tokai.ac.jp

Abstract—
USB(Universal Serial Bus) is a very popular interface in recent computer systems. USB replaces legacy interface, such as serial port, pararel port, and so on. We designed USB device controller IYOYOYO, and USB Gamepad device with it. This Gamepad device was implemented on an FPGA, with IYOYOYO.

I. Design Policy

IYOYOYO supports Low Speed mode, which is defined in *Universal serial bus specification revision 1.1*[1].

IYOYOYO pursues USB protocol. The device which uses IYOYOYO can transfer data to EndPoint without knowledge of USB. A USB device is configured as Figure.I. We used **SFL**[8] for the description language. We designed a USB Gamepad device with IYOYOYO.

II. USB Device Controller IYOYOYO

Many of existing USB controllers pursue up to Transfer level. These controllers can not handle non-supported Transfers.

We intended IYOYOYO USB Controller to pursue only packet level. All the high level jobs are left for firmware. Hence, IYOYOYO can process any type of Transfers.

Figure.II is a block diagram of USB device with IYOYOYO. IYOYOYO consists of USB interface module and MPU module SNXU. The interface module is memory mapped in the MPU's address space. MIC2550[7] is a level converter between differential bus signal and digital signal.

Table.I show the protocol stack and the role of module.

A. USB Interface

Figure.III is the block diagram of USB Interface. It pursues subordinate layer protocols.

USB Interface provides following services.

- Auto synchronize.
- NRZI Encoding and Decoding.
- 16bit CRC Encode, 16bit 5bit CRC Check.
- Stuff bit detect and remove, insert.
- Detect EOP(End Of Packet) Bus state.
- Detect Reset Bus state, and system Reset.
- Auto device address check. Auto reject illegal device address token packet.
- Auto split and byte align packet elements. Endpoint, Packet ID, Address, CRC, DATA.
- Receive Packet ID to PID register. Send Packet ID from PID register.

i. Packet Receive Operation

On sync pattern of a packet, the interface makes the synchronization of the device. It decodes NRZI and removes stuff bits of receive packets. When it receives packet ID, it sets the ID to packet ID register.

If Packet ID designates a DATA, the interface sets next data to Recv FIFO. And it checks the DATA by CRC16. If packet ID designates a TOKEN, next 7bits is address, and 4bits is endpoint, and 5bits is CRC code, respectively. The interface also checks device address in the packet. The interface extends the received endpoint to 8bit and set it into Recv FIFO. And, it checks the data by CRC5 and sets status flag.

TABLE I
USB Protocol stack

Protocol	Contents	Cover Modules
Application	HID GamePad	SNXU, Firmware
Transfer Level	Control, Interrupt	
Transaction Level	IN, OUT, SETUP	
Packet Level	Packet distinction, analysis. Token, Data, Handshake	USB Interface
Error check	CRC5, CRC16	
Signal Encode	NRZI, stuff bit, sync	
Electrical Level	differential signal	MIC2550 & Pull-up resister

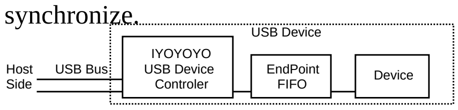


Fig. I. The configuration of USB Device with IYOYOYO

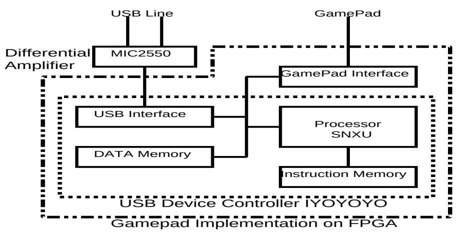


Fig. II. The block diagram of USB gamepad device with IYOYOYO

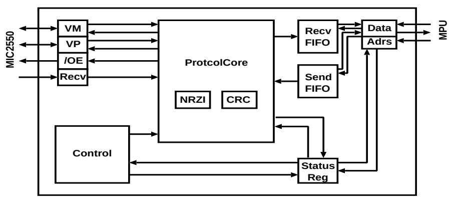
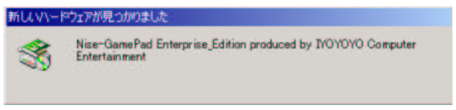


Fig. III. The block diagram of USB Interface module


新しいハードウェアが見つかりました

Nise-GamePad Enterprise_Edition produced by IYOYOYO Computer Entertainment

Fig. IV. Detect Dialog on MS Windows

TABLE II
USB INTERFACE REGISTER MAP

Name	Address	Type(from MPU)
Recv FIFO	0x8000	ReadOnly
Recv STAT	0x8002	ReadOnly
Send FIFO	0x8003	WriteOnly
Send STAT	0x8006	WriteOnly
DeviceAddress	0x8008	WriteOnly
Packet ID	0x8009	ReadWrite

TABLE III
USED MEMORY

Memory Type	Used Word
Instruction	242
Data	300

TABLE IV
SNXU SPECIFICATION

16Bit RISC like Harvard architecture Processor	
General Register 16Bit×4	
Non-Pipeline, Average Instruction Cycle 2clk	
Instruction set	ADD AND SLT NOT SR LD ST LDA BZ BAL TAG

Packet receive operation is terminated when it receives End of packet(EOP).

ii. Packet Send Operation

MPU initiates the packet send operation. The interface sends SYNC pattern and packet ID with the send request from MPU. If also carries our stuff bit insertion and NRZI encoding. For HANDSHAKE packet, the interface send SYNC and packet ID. For DATA packet, it sends SYNC, packet ID, and the DATA. The interface adds 16bit CRC following the DATA. At the end of the packet, it sends EOP.

B. MPU SNXU

Table.IV shows the specification of SNXU. SNXU is a RISC processor which have only 11 instructions.

We added a special instruction TAG for the USB interface. It read 1byte from Recv FIFO of USB interface. If the FIFO is empty, SNXU will wait for the arrival of the data in Recv FIFO.

C. Firmware

The firmware uses TAG instruction for the packet data from FIFO. It uses Load(LD) instruction for Packet ID.

In Control Transfer, the device sends Descriptors or configuration data depending on the receiving data packet. Those operations use EndPoint 0.

In Interrupt Transfer, the device sends EndPoint data. EndPoint is designated by the received TOKEN packet. Our USB device has EndPoint 1 for interrupt transfer. EndPoint1 is used for the GamePad data in the USB device.

Table.III shows the used memory words for the device. We use the internal memory of FPGA.

III. CONCLUSION

We designed a Low Speed HID Gamepad with IYOYOYO.

The Gamepad interface module has a memory mapped FIFO for the pad data. It detects Gamepad state change and pushes Pad state on FIFO.

We also designed firmware and descriptors(Device, Configuration, EndPoint, HID, Report, and String) compliant to USB specification 1.1[1].

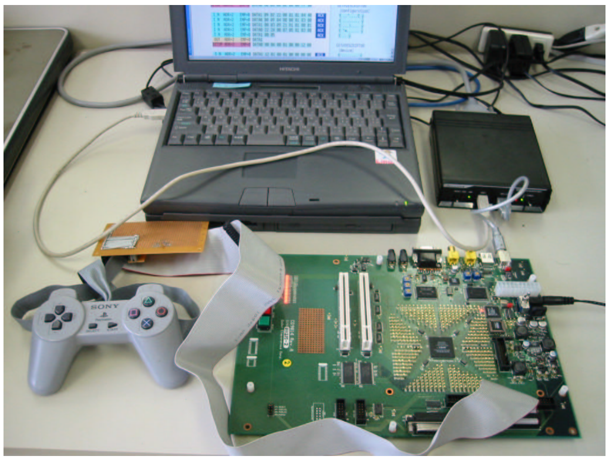
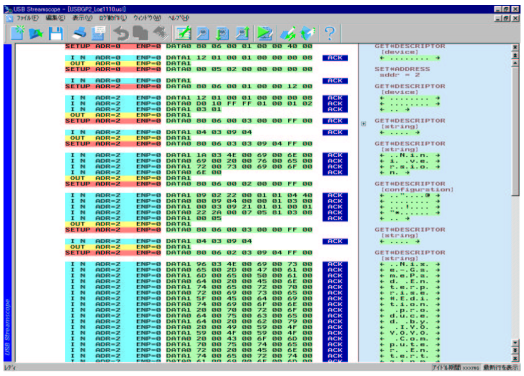


Fig. V. Implemented USB Gamepad on FPGA Board



```
SETUP ADR=0  ENP=0 DATA0 80 06 00 01 00 00 40 00         GET#DESCRIPTOR
                                                           [device]
 I N  ADR=0  ENP=0 DATA1 12 01 00 01 00 00 00 08  ACK      ← ........ →
 OUT  ADR=0  ENP=0 DATA1
SETUP ADR=0  ENP=0 DATA0 00 05 02 00 00 00 00 00         SET#ADDRESS
                                                           addr = 2
 I N  ADR=0  ENP=0 DATA1                          ACK
SETUP ADR=2  ENP=0 DATA0 80 06 00 01 00 00 12 00         GET#DESCRIPTOR
                                                           [device]
 I N  ADR=2  ENP=0 DATA1 12 01 00 01 00 00 00 08  ACK      ← ........ →
 I N  ADR=2  ENP=0 DATA0 D0 10 FF FF 01 00 01 02  ACK      ← ........ →
 I N  ADR=2  ENP=0 DATA1 03 01                    ACK      ← .. →
 OUT  ADR=2  ENP=0 DATA1
SETUP ADR=2  ENP=0 DATA0 80 06 00 03 00 00 FF 00  ACK    GET#DESCRIPTOR
                                                           [string]
 I N  ADR=2  ENP=0 DATA1 04 03 09 04              ACK      ← .... →
 OUT  ADR=2  ENP=0 DATA1
SETUP ADR=2  ENP=0 DATA0 80 06 03 03 09 04 FF 00         GET#DESCRIPTOR
                                                           [string]
 I N  ADR=2  ENP=0 DATA1 1A 03 4E 00 69 00 6E 00  ACK      ← ..N.i.n. →
 I N  ADR=2  ENP=0 DATA0 69 00 20 00 76 00 65 00  ACK      ← i. .v.e. →
 I N  ADR=2  ENP=0 DATA1 72 00 73 00 69 00 6F 00  ACK      ← r.s.i.o. →
 I N  ADR=2  ENP=0 DATA0 6E 00                    ACK      ← n. →
 OUT  ADR=2  ENP=0 DATA1
SETUP ADR=2  ENP=0 DATA0 80 06 00 02 00 00 FF 00         GET#DESCRIPTOR
                                                           [configuration]
 I N  ADR=2  ENP=0 DATA1 09 02 22 00 01 01 04 40  ACK      ← .."....@ →
 I N  ADR=2  ENP=0 DATA0 00 09 04 00 00 01 03 00  ACK      ← ........ →
 I N  ADR=2  ENP=0 DATA1 00 03 09 21 01 01 00 01  ACK      ← ...!.... →
 I N  ADR=2  ENP=0 DATA0 22 2A 00 07 05 81 03 08  ACK      ← "*...... →
 I N  ADR=2  ENP=0 DATA1 00 05                    ACK      ← .. →
 OUT  ADR=2  ENP=0 DATA1
SETUP ADR=2  ENP=0 DATA0 80 06 00 03 00 00 FF 00         GET#DESCRIPTOR
                                                           [string]
 I N  ADR=2  ENP=0 DATA1 04 03 09 04              ACK      ← .... →
 OUT  ADR=2  ENP=0 DATA1
SETUP ADR=2  ENP=0 DATA0 80 06 02 03 09 04 FF 00         GET#DESCRIPTOR
                                                           [string]
 I N  ADR=2  ENP=0 DATA1 96 03 4E 00 69 00 73 00  ACK      ← ..N.i.s. →
 I N  ADR=2  ENP=0 DATA0 65 00 2D 00 47 00 61 00  ACK      ← e.-.G.a. →
 I N  ADR=2  ENP=0 DATA1 6D 00 65 00 50 00 61 00  ACK      ← m.e.P.a. →
 I N  ADR=2  ENP=0 DATA0 64 00 20 00 45 00 6E 00  ACK      ← d. .E.n. →
 I N  ADR=2  ENP=0 DATA1 74 00 65 00 72 00 70 00  ACK      ← t.e.r.p. →
 I N  ADR=2  ENP=0 DATA0 72 00 69 00 73 00 65 00  ACK      ← r.i.s.e. →
 I N  ADR=2  ENP=0 DATA1 5F 00 45 00 64 00 69 00  ACK      ← #.E.d.i. →
 I N  ADR=2  ENP=0 DATA0 74 00 69 00 6F 00 6E 00  ACK      ← t.i.o.n. →
 I N  ADR=2  ENP=0 DATA1 20 00 70 00 72 00 6F 00  ACK      ←  .p.r.o. →
 I N  ADR=2  ENP=0 DATA0 64 00 75 00 63 00 65 00  ACK      ← d.u.c.e. →
 I N  ADR=2  ENP=0 DATA1 64 00 20 00 62 00 79 00  ACK      ← d. .b.y. →
 I N  ADR=2  ENP=0 DATA0 20 00 49 00 59 00 4F 00  ACK      ←  .I.Y.O. →
 I N  ADR=2  ENP=0 DATA1 59 00 4F 00 59 00 4F 00  ACK      ← Y.O.Y.O. →
 I N  ADR=2  ENP=0 DATA0 20 00 43 00 6F 00 6D 00  ACK      ←  .C.o.m. →
 I N  ADR=2  ENP=0 DATA1 70 00 75 00 74 00 65 00  ACK      ← p.u.t.e. →
 I N  ADR=2  ENP=0 DATA0 72 00 20 00 45 00 6E 00  ACK      ← r. .E.n. →
 I N  ADR=2  ENP=0 DATA1 74 00 65 00 72 00 74 00  ACK      ← t.e.r.t. →
```

Fig. VI. USB Gamepad configuration packet log

TABLE V
LOGIC SYNTHESIS SUMMARY

Device	Gates(LE)	Max Frequency
ALTERA EP20K160EFC484-3	1645/6400 (25%)	28.48MHz
NEC cmos9	14747gates	44.62MHz

We used an ALTERA APEX 20KE, and Quartus II1.1[9] for fitting. Table.V shows the Logic Synthesis statistics. The picture of the board is shown in the Figure.V.

We successfully use our device as a Gamepad with the control panel and DirectX Games on MS-Windows. The packet log is shown in the Figure.VI, and the detecting dialog is shown in the Figure.IV.

IYOYOYO is a compact and flexible USB core. We will use it for many projects in our laboratory.

REFERENCES

[1] USB ORG : *Universal Serial Bus Specification Revision 1.1*
[2] Triceps : *The Design Method of USB Interface*
His Publisher, 1998.
[3] CQ Publishing Co.,Ltd.
TECHI Vol.8 The All of USB Hard & Soft Development
His Publisher, 2001.
[4] Moroboshi Ramu : *The Trial Development of USB Host System, Dos/v danyo*
His Publisher, 2001 C61.
[5] *USB Acceleration Sensor Mouse*
http://www.yellowsoft.com/contents/usersamp/user5/
[6] *USB Communication Programing Technique*
http://www.picfun.com/usbframe.html
[7] MICREL
MIC2550 Universal Serial Bus Transceiver Specification
[8] Parthenon : http://www.kecl.ntt.co.jp/parthenon/
[9] ALTERA : http://www.altera.co.jp/