Research and Design of Fast Charge Controller Based on USBPD Protocol of FPGA

Dewu Xia
Shao Xing Guang Da Xin Microelectronics Co., Ltd China. Zhejiang Hangzhou 310012
xiaidewu@symicro.com.cn

Abstract. With the advent of the intelligent era, the function of electronic devices such as large-screen mobile phones, laptops, and tablet computers have become increasingly abundant and intelligent. However, at the same time, it has brought the problem of a sharp rise in power consumption. In the case of no breakthrough progress in existing lithium battery technology, people have begun to explore more "electricity replenishment" in the field of fast charging. The fast charging formula is: energy (W) = time (T) × power (P). To increase the charging speed, the key is to increase the charging power. Therefore, USBPowerDelivery (USB_PD) was born as the most mainstream fast charging technology today.

1. Researches at home and abroad
In 2014, smart electronic devices entered the era of large screens, and the charging power of MicroUSB 5V2A reached its limit, which could no longer meet the demand for high-power charging. However, USB_IF, the creator of the USB protocol, did not develop a unified USB charging standard at that time. Therefore, Qualcomm Incorporated in the United States took advantage of its industry position and began to push QC 2.0 and QC 3.0 and to develop the 9V/2A, 12V/1.5A fast charging technology route. However, unlike USB_IF, which is a non-profit organization positioning, all Qualcomm's technical standards serve for its product ecology and business interests. OPPO, MTK, Huawei, TI and other companies have successively launched their own fast charging standards, trying to occupy a part of market.

USBPD is the mainstream fast charging technology currently on the market. It is an inevitable development trend to integrate USB Type-C interface and USBPD technology in embedded systems. In this paper, a USBPD controller based on PIC microprocessor is designed by top-down modularization method, and the design of USBPD controller transmitter, receiver and state machine is introduced in detail. Using FPGA emulator, USBPD protocol tester and oscilloscope as the verification platform, the actual measurement shows that the controller sends and receives data are in accordance with the USBPD data transmission standard. The USBPD controller reduces the hardware circuit area, has the advantages of low cost and easy expansion, and can be widely used in the design of the USBPD fast charging system.

2. Technology background
Most consumer electronic products such as mobile phones, MP3, MP4, and tablets use MicroUSB ports for charging, but they are limited by traditional USB charging specifications, and the power supply for charging is relatively small, whether it is 2.5W of USB2.0, 4.5W of USB3.0, or 7.5W of USBBC1.2, although they can meet the charging needs of relatively small electric devices such as...
mobile phones and MP3 batteries, it still takes a lot of time to fully charge the battery. As consumer electronic products such as mobile phones and tablet computers become increasingly powerful, the power consumption is also increasing, and the battery capacity is limited to increase, resulting in higher and higher charging frequency. For this reason, how to achieve the rapid speed of electronic products Charging has become a key research and development direction of major manufacturers. At present, more mature fast charging technologies include the QC (QuickCharge) standard proposed by Qualcomm Incorporated and the MaxCharge standard proposed by TI Incorporated. However, these technical standards are based on raising the charging voltage as a basic means. There are problems such as large power loss due to voltage conversion, large heat generation during charging of electrical products, and poor versatility.

Taking mobile phones as an example, the problems in traditional mobile phone fast charging are:

Firstly, the electrochemical characteristics of the battery itself have limitations. At present, the average battery capacity of mainstream mobile phones is around 2.5 AH. According to the design of high-energy density lithium-ion batteries, the charging current should not exceed 2.5 A. Under this premise, the theoretical limit of the time for the mobile phone to be fully charged is 1 hour, which is actually not less than 1.5 hours.

Secondly, the problems in the interface and connection impedance of the charging circuit.

Thirdly, the heat dissipation problem inside the mobile phone. The traditional mobile phone internal charging control circuit is to step down the external power supply through the BUCK voltage conversion circuit and then charge the mobile phone battery. The inductance and switching MOS in the BUCK voltage conversion circuit both consume energy, which in turn brings a higher temperature rise in the narrow space of the mobile phone.

3. USB_PD protocol
The USB_PD power transmission protocol is the latest USB power supply standard developed by the USB Association (USB_IF). This protocol standard can achieve a maximum power transmission of 100W (20V5A) between USB devices, which can meet the charging needs of almost all electronic devices at present. Due to the substantial increase in charging current and voltage, traditional Type_A, Type_B, Micro_USB and other interfaces and data lines will not be able to withstand the transmission of information under high current and voltage, therefore, this article introduces the Type_C interface. Type_C interface, as its special interface structure design (both positive and negative have 12 pins), which has 4 Vbus pins and 4 Gnd pins, can carry information transmission under high current and voltage. In other words, the USBType_C interface naturally supports fast charging. Under the current transmission of the same specification, the USBType_C interface has less energy loss during the charging and discharging process than the traditional interface.

Therefore, the research and design of USB_PD fast charging scheme based on Type_C interface were done. The VeriogHDL hardware description language is used to complete the design of the physical layer sending and receiving modules in the USB_PD power transmission protocol. The sending module uses 4b5b encoding, CRC32 calculation, and BMC encoding to complete the transmission protection of different types of energy data packets, and the receiving module completes the 5B4B decoding, CRC32 checking and BMC decoding of the energy data packet. The data transmission carrier is the Type_C interface, so by configuring the Type_C interface peripheral circuit on the FPGA development board, it provides a 3.3V working voltage for the CC channel that transmits USB_PD information. Then through the USB_PD logic analyzer and oscilloscope waveform analysis, we can verify the entire charge and discharge process, and finally through the Apple's new MacBook, we carried out charge and discharge test, which reached 29 W of charge and discharge power, and achieved the expected goal of high-power fast charge[1].

4. USBD controller architecture
The USBD controller mainly includes a PD transmitter, a PD receiver and a Baud rate control unit for sending and receiving. In our design, a top-down modular design method is used, and a model is
built using Verilog hardware description language. The USBPD controller is mainly responsible for data sending, receiving and protocol processing.

The task of the PD transmitter is to receive the initial data packet from the protocol layer, and then add the preamble, SOP, CR32 calculation and EOP to the data packet in the physical layer according to the message format specified by the USBPD protocol (except for the preamble External) to carried out 4B5B encoding, and finally BMC encoding the data packet to form a complete set of data packets sent to the external receiver through the CC channel; PD receiver's task is to receive data packets from the external sender on the CC channel; BMC decoding the data packet, testing SOP to decode the packets received by 5B4B (including CRC32), and finally testing EOP and performing CRC32 validation; if the CRC32 verification can validly send data to the protocol layer, the data package is received; if the CRC32 verification cannot validly sent, the data package is lost; In the PD Baud rate control unit, we set different Baud rates for sending and receiving data respectively [2].

5. Hardware design of USBPD controller

5.1. Design of Baud rate control unit

Data transmission and reception Baud rate, in our design, use MCT0 and MCPR0 as the transmission and reception Baud rate control, set different Baud rates for data transmission and reception respectively, the standard bit rate on the CC channel is 300kHz. The Baud rate is determined by MCPR0; if the system clock is 20 MHz, MCPR0 = 20 MHz/300 kHz -1 = 66, the sampling Baud rate is determined by MSAMPLE [5:0] in the MCCTON register, and the calculation formula is 20 MHz/300 kHz/4= 16; The sampling Baud rate is counted by MCT0S. When MCT0S and MSAMPLE[5:0] are equal, the flag bit SCNT_SYNC is set to 1 to indicate that the Firstly sampling is completed and BMCR_SCNT automatically increases by 1 to repeat this process, until MCT0 and MCPR0 are equal, BMCR_SCNT will be reset from 3 to 0, indicating that one bit sampling is completed [3].

5.2. Transmitter module design

The overall block diagram of the design of the transmitter is shown in Figure 1. PDTX_STETE and MC_STETE are the transmission state machine and the BMC encoding state machine, respectively. The former is mainly for grouping data and the latter is for BMC encoding of grouped data.

![General block diagram of transmitter](image)

Firstly, the physical layer receives the initial data packets (Header and Data) from the protocol layer. It adds the preamble, SOP, CR32 calculation, and EOP parts. The complete data packet is sent to the DATA_TRANS module, and the Header and Data in the data packet are passed in sequence. The CRC32 calculation is performed in the CRC32 module, and the calculated result is returned to the DATA_TRANS module. Secondly, the calculated values of SOP, Header, Data, CRC32, and EOP are
entered into the 4B5B module for encoding. At the same time, the PDTX_STATE state machine module selects the input data. The complete data packet is processed and output. Finally, all the bits in the data packet are sequentially passed into the MC_STATE state machine module for BMC encoding. The encoded data will be returned to the PDTX_STATE state machine module. The final data packet is output from PDTX_DATA to CC channel.

5.2.1. RC32 calculation module
In order to prevent the data packets (Header and Data) of the protocol layer from being damaged or lost during the transmission process, the CR32 calculation of the Header and Data parts of the data packet plays a role in protecting the data.

5.2.2. 4B5B encoding module
In this design, the entire data packet (except the preamble) is encoded by 4B5B, and the data packet is transmitted to the 4B5B encoding module as a group of 8 bits. The encoded data is stored in BMC_THBUF[4:0] and BMC_TLBUF[4:0] in the register.

5.2.3. Sending state machine module
As shown in Figure 2, the transmitter has been in the TX_IDLE state without being enabled; once enabled, it means that data is coming in the protocol layer. When LOADEN0 and LOADEN1 are both 1, the sending state machine is in the TX_WAIT state, and waiting for the data to come, TX_SYNC is the synchronization state, which is used to determine whether there is a delay of 0. If there is no delay of 0, it will enter the TX_OK state to indicate the completion of the transmission. If there is a delay of 0, the number will enter the TX_DLY state; if DLY_OK is 0, DLY_CNT will be decreased until the value of DLY_CNT is 0, the DLY_OK flag will be automatically set to 1, and then enter the TX_OK state to indicate that the data transmission is complete.

![Figure 2 Sending state module](image-url)
The data stored in the TXBUF[9:0] register has the high-order bit followed by the low-order bit and is sent bit by bit and the low-order bit is followed by the high-order bit. When TXBUF[9:0] is in the process of sending data, PDTXIF will be set to 0 and BMCTH[4:0] and BMCTL[4:0] registers are latched and data cannot be written. When TXBUF[9:0] data transmission is completed, PDTXIF will be automatically set to 1. TXBUF[9:0] is in TX_BUF state when sending data. When the data is being sent, TX_CNT will decrement and the TX_OK flag is 0, until TX_CNT is reduced to 1, indicating that the data transmission is completed, the TX_OK flag is automatically set to 1 and enters the TX_OK state.

When the data is being sent, TX_CNT will decrement and the TX_OK flag is 0, until TX_CNT is reduced to 1, indicating that the data transmission is completed, the TX_OK flag is automatically set to 1 and enters the TX_OK state.

5.2.4. BMC coding module

Biphasic Mark Coding (BMC) is a coding method used to transfer USBPD information. This method uses a dedicated DC connection and is identified by a CC line [6]. In BMC, there is a conversion at the beginning of each bit time (UI), and there will be a second conversion in the middle of UI when sending 1. BMC can perform effective DC balance (each 1 is DC balanced, and 2 consecutive 0 are also DC balanced, regardless of the inverse of the number of 1 revolutions), the schematic diagram of the BMC encoding method is shown in Figure 3.

As shown in Figure 4, MC_00 and MC_11 represent the BMC coding state corresponding to 0, and MC_01 and MC_10 represent the BMC coding state corresponding to 1. In this design, each bit of data under the TX_BUF state is BMC coded. The initial state of the state machine is MC_00 because the sending end is initially in low level, and the BMC encoded signal must meet the transmission requirements of the CC channel transmission. It will be detected once during transmission of each data. PDTX_DATA in the MC_00 state sends 0. PDTX_DA_TA in the MC_11 state sends 1. In MC_01 state, when and BMC_SCNT is 00, PDTX_DATA sends 0, when BCMCR_SCNT is 10, PDTX_DATA sends 1, when BMCR_SCNT is 10, PDTX_DA_TA sends 0. When MCT0 and MCPR0 are equal and BMCR_SCNT is 11, a data BMC encoding and sending is completed.

As shown in Figure 4, MC_00 and MC_11 represent the BMC coding state corresponding to 0, and MC_01 and MC_10 represent the BMC coding state corresponding to 1. In this design, each bit of data under the TX_BUF state is BMC coded. The initial state of the state machine is MC_00 because the sending end is initially in low level, and the BMC encoded signal must meet the transmission requirements of the CC channel transmission. It will be detected once during transmission of each data. PDTX_DATA in the MC_00 state sends 0. PDTX_DA_TA in the MC_11 state sends 1. In MC_01 state, when and BMC_SCNT is 00, PDTX_DATA sends 0, when BCMCR_SCNT is 10, PDTX_DATA sends 1, when BMCR_SCNT is 10, PDTX_DA_TA sends 0. When MCT0 and MCPR0 are equal and BMCR_SCNT is 11, a data BMC encoding and sending is completed.
5.3. Receiver module design
In the design of the receiver, PDRX_STETE is the receiving state machine, which mainly unpacks the received data packets. Firstly, the input data from the CC channel is BMC decoded, and the decoded data is reversed in order of the high bit in front of the low bit. At the same time, according to the state machine PDRX_STETE and PD receive flag status bits, the data of each part were received and decoded. Secondly, the data 5B4B was decoded. And finally, the decoded 5B4B data was sent to CRC32 to calculate the new CRC32 calculation value, then compared with the calculated values of the old and new CRC32. If they are the same, it means that the data packet is correct, and it is transmitted to the protocol layer of the receiver. If they are different, it means that the data packet is wrong, and discard it.[4]

5.3.1. BMC decoding module
The receiver receives the encoded data from the transmitter through the CC channel of the Type_C interface. Because the data transmitted on the CC channel requires BMC encoding, the receiver must perform BMC decoding on the received data. In this design, the data received on the channel is sampled by BMR_CCDAT. When BMR_SCNT is 01 and SCNT_SYNC is 1, the data of BMCR_CCDAT is transferred to BMCR_SDAT[0]. When BMCR_SCNT is 11 and SCNT_SYNC is 1, the BMCR_CCDAT data is passed to BMCR_SDAT. Two samples are taken from each bit received according to the sampling baud rate, and then the XOR operation is performed on the results of the two samples. The result of the XOR is the BMC decoded data. After each bit is received, we used MCTSYN_EN as the end flag. In this way, the preamble, SOP, Header, Data and EOP decoding are completed.

5.3.2. 5B4B decoding module
After BMC decoding, the Heder, Data and CRC32 data are stored in BMC_RLBUF[4:0] and BMC_RHBUF[4:0] for 5B4B decoding, and the final original data is output from the BMCR5T4 register in groups of 8 bits.

5.3.3. Receiving state machine module

![Figure 5 Receiver state diagram](image-url)
As shown in Figure 5, the receiver has been in the IDLE state without being enabled. Once it is enabled, it means that there is data coming in from the outside. It firstly enters the WAIT state and waits for data to be received. When BMR_SYNC[0] is 0 and BMCR_SYNC[1] is 1, it enters the SFirstly state, when BMCR_SYNC[0] is 1 and BMCR_SYNC[1] is 0, it enters the state of receiving the preamble, and at the same time, Preamble_cnt[5:0] is used as the preamble counter, in order to be accurate and error-free. After receiving the preamble, the BMC decoded preamble is detected. When the condition is met, the Preamble_cnt[5:0] will increase by one, and when the Preamble_cnt[5:0] count reaches 63, it indicates that the preamble reception is completed, and the SYNC64_FLAG flag is completed. The bit will be set to 1, and then enter the SYN1 state. When the Firstly SYNC in the SOP is detected and received, the SYN1_FLAG flag will be set to 1. The SYN1_FLAG must be cleared by software. In order to simplify the design of the receiver, the remaining three SYNC or the RST codes are in the DATA state, and use BMCRNT[2:0] as a counter, when the second SYNC in the SOP is detected, the SYN1_FLAG will be set to 1, and then cleared by software until the fourth in the SOP is detected and received. The detection of SOP is completed only when a SYNC is completed. After the reception of Header and Data is completed, the DATA_FLAG flag will be set to 1 and will be cleared by software. Finally, when the EOP_FLAG flag is 1, it indicates that the EOP reception is complete and jump to IDLE State, and when there is an error in the received data, it will also return to the IDLE state.

5.3.4. CRC32 calculation module
After the data is decoded by 5B4B, the obtained 4-bit output data is transmitted to the CRC32 calculation module for CRC32 calculation. The calculated result is compared with the received CRC32. If the two values are equal, the received data packet will be transmitted to the protocol layer, if the two values are not equal, the received data packet will be discarded.

6. Results verification
USBPD controller is embedded in the embedded system, AlterQuartusII tool is used to synthesize the door-level netlist circuit, AlterCylone IVEP4CE6E22C8NFPGA emulator and USBPD (POWER_Z) protocol tester, and oscilloscope as verification platform, USBPD controller module adopts PIC 20 MHZ system clock to ensure the correctness of the superior rate data from CC channel protocol layer. In this paper, the data is generated by the software and the HEX file is generated and burned into the emulator to verify the platform.

In order to verify whether the data transmission of the USBPD controller conforms to the USBPD protocol standard, a digital oscilloscope is used to sample the waveform of the CC transmission channel for verification. The PD protocol tester can be used as the user terminal (suction terminal) to set it to the Monitor mode, after the system is powered on, if the emulator (source end) detects that a device is plugged in, it will transmit its own power supply capability to the user end (suction end), and capture the CC line of TippeC through an oscilloscope.

The Firstly data packet is the 5V/3A power supply capability package (Source_Capability) command provided by the source to the sink. It is the data from the protocol layer (Header and Data) "111a00001912c". After it entering the physical layer, we added the preamble, SOP, CR32 Calculation and EOP to form a complete set of data packets. The second data packet is that the suction end successfully receives the data from the source end and determines that it conforms to the USBPD protocol specification and returns the GodCRC control command. After the source end successfully receives the GodCRC command, it will raise the level of PC0. This communication process illustrates the USBPD controller design meets the requirements.

This paper studies the message format, message type, and communication interaction mechanism in the USBPD protocol. Based on this design, a USBPD controller based on a PIC microprocessor is implemented. For each module design, VCS+Verdi function simulation is performed and verified by the FPGA platform. The results show that the loading of the USBPD driver by the PCI microprocessor can realize the correct transmission and reception of PD data. The hardware circuit of the USBPD
controller is simple to implement, which improves the system performance while reducing the cost and has great scalability\textsuperscript{[7]}. The software can be implemented according to the later requirements modify. For further research, the future work will realize the USBPD fast charging intelligent management system.

References
[1] Yu Dejun. Research on High Power Intelligent Power Supply Technology Based on USBTYPE-C Protocol [D]. Chengdu: University of Electronic Science and Technology, 2018.
[2] Ruan Yi, Song Qingliang, Wang Jia et al. Overview and Application of USBType-C and PD Technology [J]. IC Applications, 2017, 34(4): 31-36.
[3] Li Yong. USBPD Emerging Applications in Rapid Charging of Mobile Devices [J]. Electronics World, 2018, 25(9): 27-30.
[4] Hui Hui. Research and design of charging interface chip XD9523 supporting fast charging protocol [D]. Xi'an: Xi'an University of Electronic Science and Technology, 2017.
[5] Huang Yonghua. USBPD implementation of VDM messages in protocol [D]. Dalian: Dalian University of Technology, 2016.
[6] Li Shenghui. Design and implementation of USB B_PD protocol based on TYPE_C interface d. XI'AN: Xidian University, 2017.
[7] Zhang ye-chen, Liu you-yao, Cui pan. Design and implementation of USB PD controller based on FPGA. Information Technology, 20192: 116-120.