On-board multi-lane parallel MLVDS bus design for in-orbit embedded satellite test device

Zhiyuan Chen¹, Xiuhai Cui¹, Ning Ma¹, Datong Liu¹

¹School of Electrical Engineering and Automation, Harbin Institute of Technology, Harbin, 150080, People's Republic of China

Abstract: The traditional satellite prognostic and health management (PHM) mostly relies on the satellite downlink telemetry data. Nowadays, the application of satellite becomes more diversification and its function becomes much more complicated. As a result, the telemetry data cannot meet the data requirements for satellite PHM due to the limited resources of telemetry channels and low sampling rate. Therefore, satellite in-orbit embedded testing was proposed. In order to satisfy the requirement of large volumes of data acquisition and transmission in satellite in-orbit embedded testing, this article proposes a new data agreement by the modification of the Modbus protocol to unified control 5-lane MLVDS bus. Therefore, a multi-channel parallel MLVDS data transmission bus is proposed to realise the data transmission from the data collection end to the information processing end of the PHM system on-board. The bus protocol designed here is implemented by using the Programmable Logic Part on the Xilinx XC7Z045.

1 Introduction

With the increasing complexity of satellite functions, satellite PHM technology has become more and more important and requires larger and more real-time data [1].

The traditional satellite PHM can only rely on the housekeeping computer in satellite-integrated electronic system and its telemetry data downlinked to the ground. The ground station completes the satellite PHM through telemetry data [2, 3].

Due to the limitation of the traditional telemetry transmission system, satellite downlink data channel can hardly carry out large-scale data information transmission, and the downlink data must be transmitted in low rate. In addition, the satellite cannot be telemonitored in the non-measurement and control area. If the errors cannot be detected timely by PHM, the satellite may fail [4, 5].

To solve these problems, scholars have proposed a satellite PHM framework consisting of on-board system and ground system such as the ITHM technology proposed by NASA [6, 7], the satellite PHM system whose feature is the Satellite and Earth three-level closed loop proposed by Sun Bo et al. from the Chinese Academy of Space Technology [8, 9], and Space and Ground Integration PHM system based on a satellite bus proposed by Yu Gongjing et al. from Beijing Aerospace TT & C Technology Co., Ltd. [10].

The on-board system is responsible for fault diagnosis and short-term fault prediction through data collected on the satellite directly which has a degree of intelligent diagnostic functions. The ground system is responsible for long-term or complex fault prediction through telemetry data. The two parts cooperate to complete the satellite's health management and failure prediction tasks [11].

In-orbit embedded satellite test device is responsible for satellite PHM on-board. It consists of data acquisition unit and information processing unit and the two units can communicate through the bus. Multiple data acquisition units can be connected to the system according to actual needs which ensure the flexibility of system.

The data acquisition unit monitors the temperature, voltage, and current system working status of each load by the data communication bus between subsystems of the satellite integrated electronic system. Second, it can also directly collect the measurement and control calculations results, daily management information, and satellite telemetry point parameters from housekeeping computer through multiple data acquisition channels.
can be only one master device and multiple slave devices on one bus.

More detailed diagram is shown in Fig. 2.

The data information collected from the housekeeping computer or the on-board bus is buffered in the FIFO of each channel. The MLVDS parallel bus consists of five MLVDS bus lanes, and each channel is responsible for one or more FIFOs.

2.1 Modbus protocol

Modbus protocol is the basis of the communication protocol of the on-board multi-lane parallel MLVDS bus.

Modbus, invented by Modicon in 1979, is a standard and open communications protocol widely used in industrial control [15]. Modbus development is simple, with perfect bus scheduling strategy and error checking strategy. Through this agreement, each control device can be connected into an industrial network to achieve distributed control [16]. The on-board multi-channel MLVDS parallel bus communication base on the mature Modbus protocol design. The scheduling strategy and verification method of the Modbus bus are used to ensure the reliability of the designed bus.

Information processing unit as a master, data acquisition units as slaves, can be all hung on the bus. Information processing unit can read the data collected by each data acquisition unit through the bus, to complete the data collection requirements of PHM on-board.

2.1.1 RTU mode byte description:

As shown in Fig. 3, each byte in the RTU mode is 11 bits, which are 1-bit start bit, 8-bit data bits (Send LSB first), 1-bit parity bit, and 1-bit stop bit, respectively.

2.1.2 Modbus frame description:

As shown in Fig. 4, the Modbus protocol on a specific bus defines an application data unit that is independent of the underlying communication layer. The maximum number of Modbus RTU frames is 256 bytes. Modbus CRC generator, its polynomial representations can be expressed as (1):

\[ g(x) = x^{16} + x^{15} + x^2 + 1 \]  

2.1.3 Modbus messages description:

As shown in Fig. 5, the sending device configures the Modbus message as a frame with a known start and end marker. This allows the device to receive new frames at the beginning of the message and know when the message ends. In the RTU mode, the message frame is distinguished by the idle interval of 3.5-character time.

Combined with the application scenarios and the application requirements of the parallel MLVDS data communication bus, this paper proposed a kind of multi-lane parallel MLVDS bus on the basis of the Modbus bus, designing the following data conventions, selecting the scheduling method of Modbus and modifying the data conventions and function codes of Modbus.

2.2 Frame format design

Based on the scheduling and verification strategy of Modbus protocol, we re-design the frame format of the on-board multi-lane parallel MLVDS bus. Thus, the information processing unit can accomplish to collect data from multiple data acquisition units. According to the function desired, the designed frame is divided into request frame, data response frame, and error response frame. The request frame is an information processing unit sending a data read request to the data acquisition unit. If no check error or other error occurs, the data acquisition unit sends a data response frame and returns the corresponding data. If there is a check error or other error.
error, the data acquisition unit sends an error frame indicating the type of error.

2.2.1 Request frame: Request frame format is shown in Fig. 7. The maximum number of request frame is 256 bytes. The meaning of the data fields is as follows:

Address: 8bits. The target address 0 is the broadcast address. All slave devices must identify the broadcast address and deal with the request. 1–255 is the target slave device's address.

Function: 8bits. It indicates the function. In this design, the function only reads the data acquisition channel data, which is 0x01.

Channel number \( N \): 8bits. The serial number of the data acquisition channel read.

Size \( N \): 8bits. The number of data bytes to read from the channel \( N \).

CRC: 16bits. The CRC of the request frame. The lower 8bits are in the front, and the higher 8bits are in the rear.

2.2.2 Response frame: Data frame: Data frame format is shown in Fig. 8. The maximum number of data frame is 256 bytes. The meaning of the data fields is as follows:

Address: 8bits. The slave device address.

Function: 8bits. Indicates the function. In the data frame, the area is 0x01.

Channel number \( N \): 8bits. The serial number of the data acquisition channel read.

Data \( N \): Size \( N \times 8 \) bits. Size \( N \) bytes of data collected from the Channel \( N \).

CRC: 16bits. The CRC of the data frame. The lower 8bits are in the front, and the higher 8bits are in the rear.

Error frame: Error frame format is shown in Fig. 9. The maximum number of error frame is 256 bytes. The meaning of the data fields is as follows:

Address: 8 bits. The slave device address.

Function: 8 bits. Indicates the function. In the error frame, the area is 0xFF.

Channel number \( N \): 8 bits. The serial number of the data acquisition channel read.

Data \( N \): Size \( N \times 8 \) bits. Size \( N \) bytes of data collected from the Channel \( N \).

Check Error: If it is a check error, this area is 0x01, otherwise it is 0x00.

Reserved: Reserved for other functional error.

CRC: 16 bits. The CRC of the data frame. The lower 8bits are in the front, and the higher 8bits are in the rear.

3 Logic implementation

Logical design for the protocol is introduced in this chapter.

3.1 Master logic design

The Master logic is implemented in the Information Processing Unit to obtain the data collected by the Data Acquisition Unit.

3.1.1 Logic module block diagram: Master Logic Module Block is shown in Fig. 10.

Master Send Module: The inner part contains FIFO with a depth of 256 and a width of 8 bits, which is used to cache a request frame. At the same time, The Master Send Module sends a request frame stored in FIFO to the slave device in RTU mode. The request frames will be sent only through the Lane 0.

Master Receive Module: The inner part contains FIFO with a depth of 256 and a width of 8 bits, which is used to cache the corresponding slave response frames. At the same time, the master receive module can perform parity check on the received response frame and send the check result to the master controller.

Transferring Monitor: It is used to determine whether data transmission is being performed on the \( m_{\text{rx}} \) data receiving port. It judges whether the transmission of one frame of data is completed by judging whether the time of no data transmission exceeds 3.5 characters transmission time.

Response Timer: When the request is issued, the timer starts. By setting the timeout period, it is determined whether a timeout error occurs. This ensures that the bus will not be occupied for a long time.

CRC Generator: It generates CRC of the received response frame of the corresponding lane, and the generated CRC is transmitted to master controller for comparison.

Master Controller: It can judge whether there is an error in the response frame through the comparison of CRC checksums. Through the control of other modules, it realises the bus scheduling in Modbus RTU mode; for the specific introduction, see the next section.

External Interface:

- Control signals: Master logic module necessary control signals, including mode settings (broadcast mode/unicast mode), transmission enable, clock, reset etc.;
- Send fifo control signals, send data: Send fifo control signals are used to send FIFO write control signal and write data channel; Send data are used to write request frame in the Master Send Module.
- Receive fifo control signals, receive data: Receive fifo control signals are used Receive FIFOs read control signal and read data channel; Receive data are used to read the response received from the Slaves.

\( m_{\text{tx}} \): It is master device send port.
\( m_{\text{rx}} \): It is master device receive port.
3.1.2 Master controller state machine: The state transition diagram of master controller is shown in Fig. 11.

IDLE: IDLE state. After the bytes of the request frame to be sent are sequentially stored in the FIFO, the transmit FIFO is not empty, the system start sending the request frame and jump to SEND status.

SEND: SEND state. Turn on the Response Timer. According to the external mode, select the next state to jump to BROADCAST or UNICAST;

BROADCAST: Broadcast status. Turn off the Response Timer. When counting is started, it is expected that the slave devices receive the broadcast information and the processing is completed. The counting stops and the broadcast is completed, then the IDLE state is returned.

UNICAST: Unicast status. Waiting for response, if there is data transmission on the m_rx data receiving port, jump to the RECEIVE state; if the timer's response timer expires in this state, it indicates that the waiting response timed out, then jumps to the ERROR state.

RECEIVE: Receive status. The response frame is received, and the parity check is performed. After the data reception of one frame is completed, that is, there is no data transmission after 3.5-characters time on the m_rx data receiving port, jump to CHECK state.

CHECK: If the parity check passes, check whether the CRC generated by comparing CRC generator and the received CRC are the same. If there is an error in the data transmission, go to the ERROR status; If there is no error in data transmission, go to DEAL state.

DEAL: After the master device receives the data, the user reads the data for processing. After the processing is completed, the state jumps to DELAY state.

ERROR: ERROR: According to different error types, the user is provided with error information through the external interface. After the error processing is completed, it jumps to the DELAY state.

DELAY: Delays 3.5 characters, jumps to the IDLE state and waits for sending the next request.

3.2 Slave logic design

The Slave logic is implemented in the data acquisition unit to receive requests from the information process unit and return the response.

3.2.1 Slave logic module block diagram: Slave logical module block diagram is shown in Fig. 12.

Slave Send Module: The inner part contains FIFO with a depth of 256 and a width of 8bits, which is used to cache a response frame. At the same time, the slave send module can send a response frame stored in FIFO to the master device in RTU mode through the corresponding lane.

Slave Receive Module: The inner part contains FIFO with a depth of 256 and a width of 8 bits, which is used to cache the corresponding master request frames. At the same time, the slave receive module can perform parity check on the received response frame and send the check result to the slave controller. In a slave device, only the lane corresponding to Lane 0 has this module, because the request frame is sent only through Lane 0 Transferring Monitor: It is used to determine whether data transmission is being performed on the s_rx data receiving port. It judges whether the transmission of one frame of data is completed by judging whether the time of no data transmission exceeds 3.5 characters transmission time.

CRC Generator: Generate CRC of the received request frame of the corresponding lane, and the generated CRC is transmitted to slave controller for comparison. It can also generate the response CRC.

Slave Controller: It can compare the CRC check code to determine whether there is an error in the response frame. It can parse the request frame and write the corresponding data acquisition channel data into the corresponding Modbus lane according to the response frame format. It realises the bus scheduling in the Modbus RTU mode through the control of other modules; for the specific introduction, see the next section.

External Interface:
3.2.2 Slave controller state machine: The state transition diagram of the slave controller is shown in Fig. 13.

- **IDLE**: Idle state. Wait for receiving request, if there is data transmission on the \( m_{rx} \) data receiving port, jump to **RECEIVE** state.
- **RECEIVE**: Receive status. The request frame is received, and the parity check is performed. After the data reception of one frame is completed, that is, there is no data transmission after 3.5 character times on the \( m_{rx} \) data receiving port, jump to **CHECK** state.
- **CHECK**: If the parity check passes, check whether the CRC generated by comparing CRC Generator and the received CRC are the same. If there is an error in the data transmission, go to the **ERROR** status; If there is no error in data transmission, go to **DEAL** state.
- **DEAL**: After the slave receives the data, the user parses the request, completes the framing of the response frame according to the request, and sequentially stores each byte of the response frame to be sent into the sending FIFO, and jumps to **SEND**.
- **SEND**: Start sending response RTU message frame, after sending is finished, jump to **DELAY** state.
- **DELAY**: Delays 3.5 characters, jumps to the **IDLE** state and waits for receiving the next request.

4 Verification

As shown in Fig. 14, to verify the correctness of the specific design, in order to verify the correctness of the design, verification was done on the two XC7Z045. The bus function is tested at 100 Mbps (20 Mbps/lane).

The on-board multi-lanes of parallel MLVDS data bus has total five lanes for data acquisition of 6 FIFOs. Lane 0–Lane 3 are in turn responsible for FIFO0–FIFO3, Lane 4 is responsible for FIFO4 and FIFO5.

4.1 BER test

Perform BER (bit error rate) test on the on-board multi-lane parallel MLVDS Bus with 100 Mbps (20 Mbps/lane). Test code is generally adopted m sequence, its characteristic polynomial is expressed as (2), and the period is 511 bits. The result is shown in Table 1. Let the bus continuously transmit 1Kbits to 1Gbits of data. The bus error rate is 0, at a rate of 100 Mbps.

\[ g(x) = x^9 + x^5 + 1 \]  \hspace{1cm} (2)

4.2 Function test

4.2.1 Overtime test: The master device starts the response timer after sending a request. If the request is not received within a certain period of time, a timeout alarm is issued. Verify this timing function by changing the timer duration.

As shown in Fig. 15, after the master device sends the data in the fifo, the empty signal is set to 1, and the timeout count is started.

Set the timer to 100, as shown in Fig. 16, if the \( rx\_time\_cnt \) == 100, if no data is still received, the timeout flag is set to 1 to inform the user. The master device enters the timeout error handling state.

Change the timer to 200, as shown in Fig. 17, if the master device receives a response within the specified time, the timer is turned off. The timer_en is set to 0.

---

**Table 1** BER test result

| The amount of data | Error count |
|--------------------|-------------|
| 1Kbits/lane        | 0           |
| 100Kbits/lane      | 0           |
| 1Mbits/lane        | 0           |
| 100Mbits/lane      | 0           |
| 1Gbits/lane        | 0           |
4.2.2 CRC check test: A device on the bus receives a frame of data and performs a CRC check on it. If the check fails, the CRC check flag is set.

In order to verify this function, the master device sends the correct and incorrect CRC check code.

As shown in Fig. 18, master device sends request frame ‘01 02 00 01 01 03 02 04 06 05 02 02 02’ through the Lane 0. ‘02 02’ is the wrong CRC. It can be seen from the figure that the CRC code calculated by the slave based on the received data is 16'h8298. The low eight bits are at the top and the high eight bits are at the end, then the 16'h8282 and 16'h0202 are different, so the crc_error is set to 1.

As shown in Fig. 19, if the master device sends the request frame, the last two bytes are 8'h98 and 8'h82, then the same as the CRC code calculated by the slave, so the crc_error is set to 0.

4.2.3 Data agreement test: To verify the data protocol, the master device sends the following request frames according to the previously designed frame format.

As shown in Fig. 20, master device sends request frame ‘01 01 00 01 01 03 02 04 06 05 02 36 C2’

Address: 0x01; Function: 0x01; Channel 0: 0x00; Channel 0 Size: 0x01; Channel 1: 0x01; Channel 1 Size: 0x03; Channel 2: 0x02; Channel 2 Size: 0x04; Channel 3: 0x03; Channel 3 Size: 0x02; Channel 4: 0x04; Channel 4 Size: 0x06; Channel 5: 0x05; Channel 5 Size: 0x02;

According to the request frame, the slave responds to the data frame as shown in Fig. 21.

According to the data agreement, the request frame data is parsed as shown in Table 2.

As shown in Fig. 21 and Table 1, in the data frame, the address is 0x01, and the function code is 0x01. The data field is based on the FIFO number and the number of bytes in the master-initiated request. They are:

1. FIFO 0, 1 byte, 00
2. FIFO 1, 3 bytes, 00 01 02;
3. FIFO 2, 4 bytes, 00 01 02 03;
4. FIFO 3, 2 bytes, 00 01;
5. FIFO 4, 6 bytes, 00 01 02 03 04 05;
6. FIFO 5, 2 bytes, 00 01;

So, logical design satisfies functional requirements.

Then, send a request frame with an error CRC to verify the error processing, you can get a response error frame as shown in Fig. 22.

According to the data agreement, the response error frame is parsed as shown in Table 3.

| Table 2 | Data frame |
|---------|------------|
| Lane    | Address  | Function | Data | CRC       |
| lane 0 01 | 01     | 00 00 | 50 18 |
| lane 1 01 | 01     | 01 00 01 02 | BD A7 |
| lane 2 01 | 01     | 02 00 01 02 03 | 63 70 |
| lane 3 01 | 01     | 03 00 01 | 29 FC |
| lane 4 01 | 01     | 04 00 01 02 03 04 05 05 00 01 | 1E 1A |

| Table 3 | Error frame |
|---------|-------------|
| Lane    | Address  | Function | Check Error | CRC       |
| lane 0 01 | 01     | FF     | 01         | A0 30     |
| lane 1 01 | 01     | FF     | 01         | A0 30     |
| lane 2 01 | 01     | FF     | 01         | A0 30     |
| lane 3 01 | 01     | FF     | 01         | A0 30     |
| lane 4 01 | 01     | FF     | 01         | A0 30     |

Fig. 16 Timeout waveform

Fig. 17 No timeout waveform

Fig. 18 Wrong CRC code waveform

Fig. 19 Correct CRC code waveform

Fig. 20 Request frame waveform

Fig. 21 Data frame waveform

Fig. 22 Error frame waveform

8964
As shown in Fig. 22 and Table 3 in the error frame, the address is 0x01, and the function code is 0xff. The check error field is 0x01, the CRC filed is 0xA0, 0x30.

So, logical design meets functional requirements.

5 Conclusion

This paper proposes an on-board multi-lane parallel MLVDS data bus for internal data acquisition and transmission problems in satellite in-orbit embedded test device.

Through the reference and modification of the Modbus protocol, multi-lane parallel MLVDS bus scheduling strategies and data conventions are implemented on the FPGA. The transmission rate can be up to 5 times that of single-lane MLVDS. It can provide bus standards for internal data transmission of satellite in-orbit embedded test device.

so as to achieve a single-channel MLVDS five times faster performance.so as to achieve a single-channel MLVDS five times faster performance. The bus speed will be faster with the faster data rate of lane-channel MLVDS.

6 Acknowledgments

The authors are grateful to the School of Electrical Engineering and Automation, Harbin Institute of Technology and Automatic Test and Control Institute for providing with technical guidance and equipment support.

7 References

[1] Xiyuan, P., Jingyue, P., Yu, P., et al.: ‘Review on anomaly detection of spacecraft telemetry data’, Chin. J. Sci. Instrum., 2016, 37, (9), pp. 1929–1945
[2] Narheim, B., Olsen, O., Helferen, O., et al.: ‘A Norwegian satellite for space-based observations of AIS in the high north’. Small Satellite Conf. (SSC), Logan, UT, USA, August 2008, vol. III, p. 3
[3] Xiaotong, L.: ‘Satellite keeping technology of small satellite’, Chin. Space Sci. Technol., 2001, 21, (1), pp. 29–36
[4] Kai, G.: ‘Research on the Status Quo and development trends of aerospace telemetry technology’, J. Telemetry Track Command, 2015, 36, (5), pp. 7–15
[5] Qinyi, L., Xin, X.: ‘Application of large-data-driven PHM technology in satellite test and on-orbit management’. Prognostics and System Health Management Conf. (PHM), Harbin, China, July 2017, pp. 1–5
[6] Baroth, E., Powers, W.T., Fox, J., et al.: ‘IVHM (integrated vehicle health management) techniques for future space vehicles’. Joint Propulsion Conf. & Exhibit, Salt Lake City, UT, USA, 2001, pp. 468–470
[7] Aaseng, G.B.: ‘Blueprint for an integrated vehicle health management system’. Digital Avionics Systems Conf. (DASC), Dayton, OH, USA, October 2001, pp. 3C1/1–3C1/11
[8] Bo, S., Lei, Z., Huaimao, W.: ‘Study on satellite PHM system design technology’, Comput. Meas. Control, 2013, 21, (3), pp. 554–559
[9] Rongzheng, L., Bo, S., Lei, Z.: ‘Analysis of PHM technology for spacecraft’, Spacecraft Eng., 2013, 22, (4), pp. 95–102
[10] Gongjing, Y., Yi, X., Hongzheng, F.: ‘Survey of prognostics and health management technology and application ideal for satellite’, J. Electron. Meas. Instrum., 2014, 28, (3), pp. 227–232
[11] Shaozhi, H., Gang, Z.: ‘Review on spaceflight safety and health management’, J. Shanghai Inst. Technol., 2015, 15, (3), pp. 286–292
[12] Jian, Z., Zheng, Y., Guofei, Z.: ‘Small satellite commonly used data bus technology’, Aerospace China, 2007, (2), pp. 35–37
[13] Verma, S., Chatterjee, A.: ‘A methodology for designing LVDS interface system’. Int. Symp. on Embedded Computing and System Design (ISED), Patna, India, December 2016, pp. 284–288
[14] Shujing, S., Yanjie, X., Wanfan, H.: ‘The design of high-speed half Duplex data transmission based on MLVDS’, Sci. Technol. Eng., 2016, 16, (28), pp. 226–229
[15] Shaozheng, D., Yuan, H., Chunliang, L.: ‘Design and implement of monitor and control function of CNC based on modbus’, Med. Mach. Tool Autom. Manuf. Tech., 2017, (3), pp. 87–90
[16] Shukla, P., Singh, S., Joshi, T., et al.: ‘Design and development of a MODBUS automation system for industrial applications’. Computer Applications In Electrical-Engineering-Recent Advances (CERA), Roorkee, India, October 2017, pp. 515–520