VHDL IMPLEMENTATION OF 1553 PROTOCOL USING ACTEL IP CORE

[1]Roopesh N.M, [2]Archana & [3]Lakshmi Narasimhan

[1&2]CIT, Gubbi, Karnataka, India; [3]Scientist/Engineer SD, DSG, ISAC, Bangalore, Karnataka, India
Email: [1]roops.nm@gmail.com, [2]arc_telecom@yahoo.co.in, [3]lakshmin@isac.gov.in

Abstract- To implement a MIL-STD-1553 compatible system, the options that exist are to use a stand-alone integrated circuit or an IP core. Today, with most digital logic being implemented on FPGAs, the choice of an IP Core is advantageous. One such IP core is Core1553BRM from Actel. Core1553BRM provides a complete, MIL-STD-1553A and B Bus Controller (BC), Remote Terminal (RT) and Monitor Terminal (MT) and is compatible with legacy 1553 solutions.

This paper involves study of 1553 architecture and study of various features of Core1553BRM IP core from Actel and implementation on FPGA. A Bus Controller(BC) and Remote Terminal (RT) are implemented on Actel FPGA and data transfer between the BC and RT is demonstrated.

Keywords: Bus Controller (BC); Remote Terminal(RT); 1553B; Encode; Decode

I. INTRODUCTION

Today the use of digital techniques in spacecraft equipment has greatly increased, as have the number of subsystems and the volume of data processed by them.

Because analog point-to-point wire bundles are inefficient and cumbersome means of interconnecting the sensors, computers, actuators, indicators, and other equipment onboard the spacecraft, a serial digital multiplexed data bus was developed. MIL-STD-1553 defines all aspects of the bus which is widely used in aircraft systems also.

The 1553 multiplexed data bus provides integrated, centralized system control and a standard interface for all equipment connected to the bus. The bus concept provides a means by which all bus traffic is available to be accessed with a single connection for testing and interfacing with the system. The standard defines operation of a serial data bus that interconnects multiple devices via a twisted, shielded pair of wires. The system implements a command-response format.

MIL-STD-1553 gives an effective solution for implementation of Telemetry and Telecomm and in a spacecraft system. The main onboard computer of a spacecraft can act as a Bus Controller and all other subsystems can implement a 1553 remote terminal. This paper involves the study of 1553B Protocol and implementation of a BC and RT using IP cores in Actel FPGA.

II. EXISTING & PROPOSED SYSTEM

The MIL-STD-1553 can be implemented in two ways, they are, stand alone integrated circuit and using IP Core. If it is implemented by using stand alone integrated circuit then the overall circuit size in the satellite is going to increase.

Implementations based on stand-alone integrated circuit along with discrete components increase circuit size. An IP core implementation shall be useful in reducing the overall circuit size. So in order to decrease overall circuit size, a single chip solution i.e. implementation using IP Core is done.

Since the most digital logic in the satellite is being implemented on FPGAs, the 1553 protocol can be implemented on the same FPGA by using an IP Core on the same which utilizes the FPGA effectively & decreases the overall circuit size in satellite, hence the choice of an IP Core is advantageous.

III. 1553B PROTOCOL

A. Hardware Characteristics

The MIL-STD-1553B bus has four main elements.

- ❑ A bus controller that manages the information flow.
- ❑ Remote terminals that interface one or more simple subsystems to the data bus and respond to commands from the bus controller.
- ❑ Bus monitor that is used for data bus testing.
- ❑ Data bus components (bus couplers, cabling, terminators and connectors). Data is sequentially transmitted and received in a multiplexing scheme over two copper wires from computer to computer at a rate of 1 megabit per second.

B. Word Types

Three distinct word types are defined by the standard. These are:

- ❑ Command words.
- ❑ Data words.
- ❑ Status words.

Each word type has a unique format, yet all three maintain a common structure. Each word is twenty bits in length. The first three bits are used as a synchronization field, thereby allowing the decode lock to re-sync at the beginning of each new word.

The next sixteen bits are the information field and are different between the three word types. The last bit is the parity bit. Parity is based on odd parity for the single word. The three word types are shown in Fig. 1.

A command word shall be comprised of a sync waveform, remote terminal address field, transmit/receive (T/R) bit, sub address mode field, word count/mode code field, and a parity (P) bit. A data word shall be comprised of a sync waveform, data bits, and a parity bit. A status word shall be comprised of a sync waveform, RT address, message error bit, instrumentation bit, service request bit, three reserved bits, broadcast command received bit, busy bit, subsystem flag bit, dynamic bus control acceptance bit, terminal flag bit, and a parity bit.

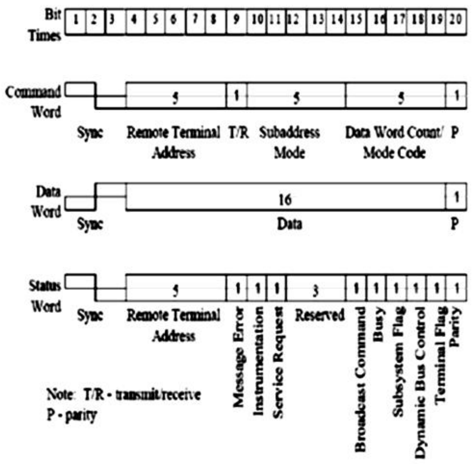


Fig 1: Word types

C. Encoding

The data encode shall be Manchester II bi-phase level. A logic one shall be transmitted as a bipolar coded signal 1/0 (i.e. a positive pulse followed by a negative pulse). A logic zero shall be a bipolar coded signal 0/l (i.e. a negative pulse followed by a positive pulse). A transition through zero occurs at the midpoint of each bit time. The transmission bit rate on the bus shall be 1.0 megabit per second. The command sync waveform shall be an invalid Manchester waveform. The width shall be three bit times, with the sync waveform being positive for the first one and one-half bit times, and then negative for the following one and one-half bit times. If the next bit following the sync waveform is a logic zero, then the last half of the sync waveform will have an apparent width of two clock periods due to the Manchester encoding.

D. Message Format

The primary purpose of the data bus is to provide a common media for the exchange of data between systems. The exchange of data is based on message transmissions. The standard defines ten types of message transmission formats. All of these formats are based on the three word types.

The messages transmitted on the data bus includes bus controller to remote terminal transfers, remote terminal to bus controller transfers, remote terminal to remote terminal transfers, mode command without data word, mode command with data word (transmit), mode command with data word (receive), optional broadcast command as shown in Figures 2 & 3.

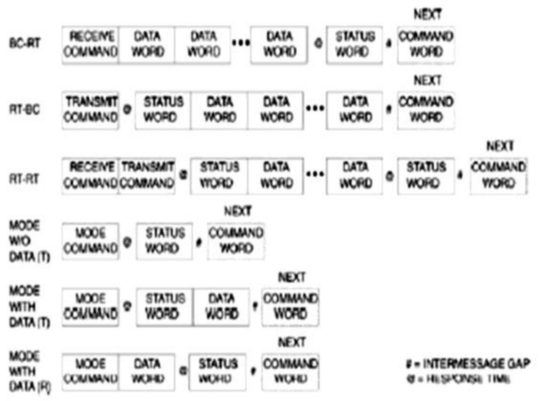

BC-RT RECEIVE COMMAND DATA WORD DATA WORD ••• DATA WORD @ STATUS WORD # NEXT COMMAND WORD
RT-BC TRANSMIT COMMAND @ STATUS WORD DATA WORD DATA WORD ••• DATA WORD # NEXT COMMAND WORD
RT-RT RECEIVE COMMAND TRANSMIT COMMAND @ STATUS WORD DATA WORD ••• DATA WORD @ STATUS WORD # NEXT COMMAND WORD
MODE W/O DATA (T) MODE COMMAND @ STATUS WORD # NEXT COMMAND WORD
MODE WITH DATA (T) MODE COMMAND @ STATUS WORD DATA WORD # NEXT COMMAND WORD
MODE WITH DATA (R) MODE COMMAND DATA WORD @ STATUS WORD # NEXT COMMAND WORD
# = INTERMESSAGE GAP
@ = RESPONSE TIME

Fig 2: Information transfer format

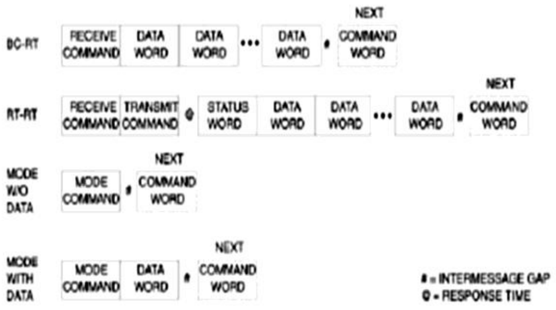

BC-RT RECEIVE COMMAND DATA WORD DATA WORD ••• DATA WORD # NEXT COMMAND WORD
RT-RT RECEIVE COMMAND TRANSMIT COMMAND @ STATUS WORD DATA WORD DATA WORD ••• DATA WORD # NEXT COMMAND WORD
MODE W/O DATA MODE COMMAND # NEXT COMMAND WORD
MODE WITH DATA MODE COMMAND DATA WORD # NEXT COMMAND WORD
# = INTERMESSAGE GAP
@ = RESPONSE TIME

Fig 3: Information transfer format (Broadcast)

E. Mode codes

Mode Codes are defined by the standard to provide the bus controller with data bus management and error handling/recovery capability. The mode codes are divided into two groups: those with, and those without, a data word.

The data word that is associated with the mode codes (only one word per mode code is allowed) contains information pertinent to the control of the bus and does not generally contain information required by the subsystem (the exception may be the Synchronize with Data Word Mode Code). The mode codes are defined by bit times 15-19 of the command word. The most significant bit (bit 15) can be used to differentiate between the two-mode code groups.

IV. CORE 1553BRM

The core consists of six main blocks: a 1553 encoder, 1553 decoders, a protocol controller block, a CPU interface, a command word legality interface, and a backend interface as shown in Figure 4.

The core can be configured to provide all three functions BC, RT, and MT or any combination of the three. All core variations use all six blocks except for the command legalization interface, which is only required in RT functions that implement the RT legalization function externally.

A single 1553 encoder takes each word to be transmitted and serializes it using Manchester encoding.

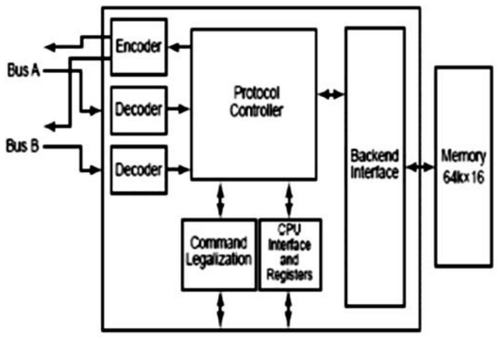


Figure 4: Core1553BRM Block Diagram

The encoder also includes loopback fail logic and independent logic to prevent Core1553BRM from transmitting for longer than the allowed period. The loopback logic monitors the received data and verifies that the core has correctly received every word that it transmits. The output of the encoder is gated with the bus enable signals to select which busses the core should be transmitting on.

Two decoders take the serial Manchester received data from each bus and extract the received data words. The decoder requires a 12, 16, 20, or 24 MHz clock to extract the data and clock from the serial stream.

The decoder contains a digital phase-locked loop (PLL) that generates a recovery clock used to sample the incoming serial data. The data is then deserialized and the 16-bit word decoded. The decoder detects whether a command, status, or data word has been received and checks that no Manchester encoding or parity errors have occurred in the word.

The protocol controller block handles all the message sequencing and error recovery for all three operating modes BC, RT, and BM. This is a complex state machine that processes messages based on the message tables set up in memory, or reacts to incoming command words. The protocol controller implementation varies depending on which functions are implemented.

The CPU interface allows the system CPU to access the control registers within the core. It also allows the CPU to directly access the memory connected to the backend interface; this can simplify the system design. The core includes thirty-three 16-bit registers. Of the 33 registers, 17 are used for control functions and 16 for RT command legalization. The RT command legalization registers can be omitted from the core, reducing device utilization.

For 1553 receive commands (BC transmits data), the data pointer determines the location of the data words to be retrieved. The core will retrieve data words sequentially from the address specified by the data pointer. Conversely, for a transmit command (BC receives data), the data pointer determines the memory location for data storage.

The core stores data words sequentially starting from this memory location. After transmission or reception, the core will begin command post processing.

Control of the core operating as an RT is accomplished through the use of control words stored in descriptor blocks, and mode codes and sub addresses sent in 1553 messages. Control word information allows the core to generate interrupts, buffer messages, and control message processing. Moreover, the descriptor block contains pointers to data buffers where mode codes and sub addresses to be used by the host or subsystem in further message processing are stored.

For receive commands, the core processes each incoming message for correct format, word count, and contiguous data. If a message error is detected, the core will stop processing the remainder of the message, suppress status word transmission, and set the message error bit (ME, bit 9) of the status word.

The core will track the message until the end of the message is detected.

V. DESIGN AND IMPLEMENTATION

The BC is designed and implemented as shown in Figure 5.

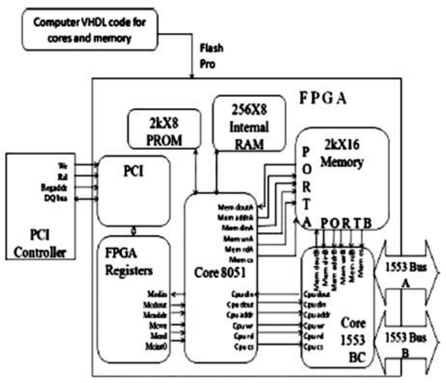


Figure 5: BC configuration block diagram

To write values to 1553core registers and to monitor command blocks we use Core8051, and a 2kX16 external memory is used to store command blocks and data block. Since the same memory is used between 8051 and 1553 it is possible that Core1553 and Core8051 read and write in the same location at the same time. Such conflicts should be avoided by using arbitration or multiplexer logic. PCI interface is used to write/read TC/TM commands to FPGA TC/TM registers.

For telecomm and, 8051 programs 1553 for BC by programming the CPU interface registers and operation starts by writing start execution bit. PCI writes command words into FPGA TC registers.

On user request FPGA sends an interrupt to 8051. 8051 writes this data to shared memory and Core1553 transmits these words on the 1553 bus.

For telemetry, a periodic scheduler(counter) is run on the FPGA once in a second. This interrupts 8051. 8051 programs Core1553BC to send TM request to RT. Appropriate locks are put in the BC FPGA to see to it TM and TC requests do not affect each other when they occur simultaneously and are scheduled one after the other. On completion, stops BC, writes status to FPGA status register.

The RT is designed and implemented as shown in Figure 6.

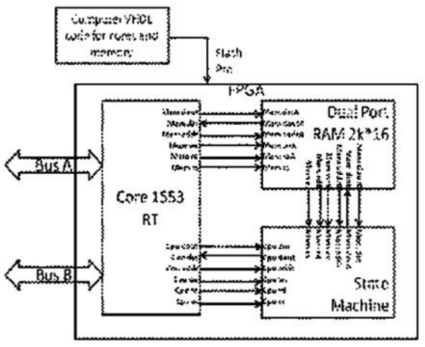


Figure 6 : RT Configuration Block Diagram

On RT side we will not use Core 8051 for initialization and controlling instead we used state machine. Core 1553 RT receives requests on bus and if the request is TC, it receives all data words and saves in memory location allocated for particular sub address/mode code. If the request word is TM, then RT takes TM data saved in memory and sends to BC through the required bus.

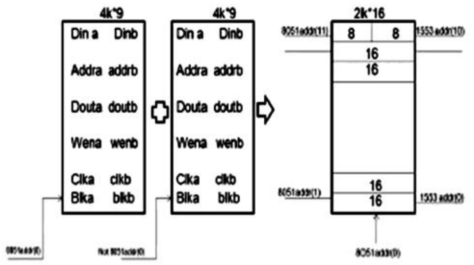
BC and RT are implemented on Actel A3P1000 Proasic3 FPGA. BC is implemented on a PCI card. RT is implemented on an FPGA kit.

An application software is designed to handle simultaneous telemetry and telecomm and requirements, using C++ on MS Visual Studio.

This software periodically runs a thread to access PCI registers(TM related) and read and display Telemetry data. A user driven thread takes command inputs from user for Telecomm and and sends them to PCI registers (TC related) which are then transmitted by BC to RT.

Memory Formation

Since in BC set up, Core8051 processes 8-bit data and Core 1553 processes 16-bit data, we formed a common memory where we can access 8-bit and 16-bit data. Actel's A3P1000 Proasic3 Evaluation kit supports 4k*9 memory block hence we used two 4k*9 memory blocks to form single 2k*16 memory block as shown in Figure 7, where one block is used to access lower byte and other block is used to access higher byte data. These two blocks are differentiated by 8051addr(0) bit on microcontroller side.



VI. ALGORITHM

Step1: Start

Step2: Initialize all required signals, 18-bit

Counter, Memory, flip flops, Multiplexer etc.

Step3: Design 18-bit Counter, required amount of Memory, flip flop, Multiplexer etc.

Step 4: Divide the board clock by 4 to 8051 clock and 8051 clock by 32 to 18-bit counter using Flip Flops.

Step 5: Using counter generate a pulse for every second for TM request.

Step 6: Write 8051 code to configure Core1553 BC registers and to manage TM and TC operation of BC.

Step 7: On BC side write VHDL code for arbiter, to provide signals for INT0 and INT1.

Step 8: If INT0 occurs 8051 makes 1553 BC to send telecomm and by accessing data stored in common memory on BC side, on receiving RT saves TC data in its memory and generates appropriate pulses.

Step 9: If INT1 occurs 8051 makes 1553 BC to send telemetry request to RT then RT sends a data words stored in memory on RT side.

Step 10: Stop

VII. RESULT

BC and RT designed using Core1553 are interfaced and tested for TM and TC. The results of simulation are recorded. The BC sends a TC request (command word) and 12 data words, RT receives those 12 TC words and saves in it memory and generates command pulses on the RT subsystem, and sends status word back to BC. This is shown in Figure 8-Figure 9.

The BC sends a TM request for every 1 sec, upon receiving this request RT sends status word and required amount of data words as shown in Figure 10.

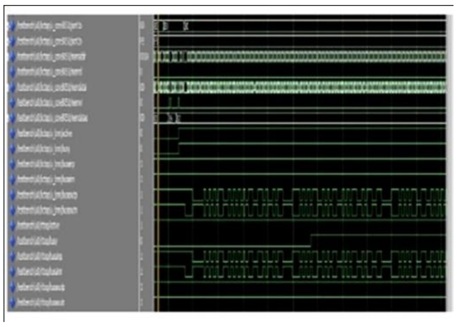


Figure 8 :BC sending command & data words

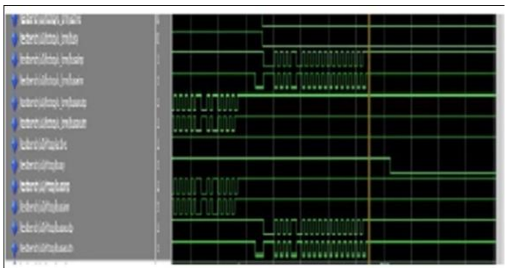


Figure 9 :RT sending command & data words

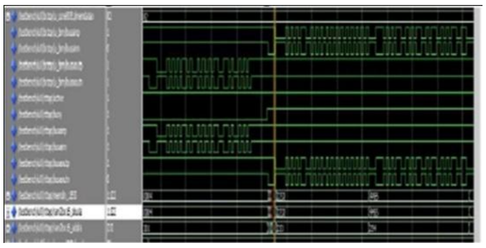
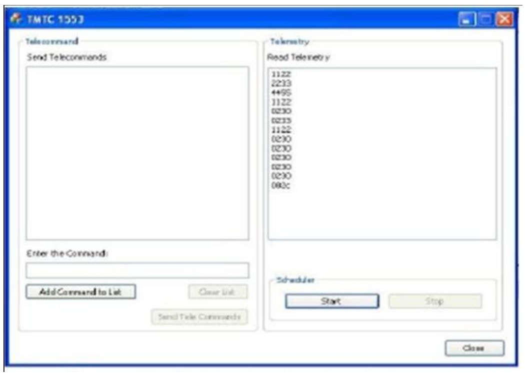


Figure 10 :BC requesting TM & RT replying

Next the BC and RT is implemented on FPGA kits, and using application software TM reception and TC sending is verified as shown in Figure 11-Figure 12. The TC reception on RT is verified by pulses and also we made loopback logic for verification and it is shown in Figure 12.


TMTC 1553

Telecommand

Send Telecommands

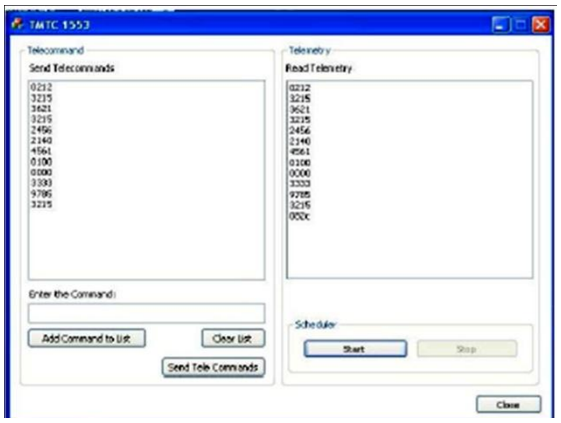
Enter the Command:

Telemetry

Read Telemetry

```
1122
2233
4495
1122
0230
0233
1122
0230
0230
0230
0230
0230
080c
```

Figure 11 : TM in Application Software


TMTC 1553

Telecommand

Send Telecommands

```
0212
3215
3621
3215
2456
2140
4561
0100
0000
3333
9785
3215
```

Enter the Command:

Telemetry

Read Telemetry

```
0212
3215
3621
3215
2456
2140
4561
0100
0000
3333
9785
3215
052c
```

Figure 12 : TC & TM in Application Software

VIII. CONCLUSION

In this paper the study of 1553 architecture and study of various features of Core1553BRM IP core from Actel are carried out successfully. A Bus Controller(BC) and Remote Terminal (RT) are implemented on an Actel FPGA and data transfer between the BC and RT is demonstrated successfully.

IX. ACKNOWLEDGEMENT

On the completion of my paper, I would like to express my deepest gratitude to all those whose kindness and advice have made this work possible.

I am greatly indebted to my guides N Lakshmi Narasimhan, scientist/engineer SD, DSG, ISAC, Bangalore and Archana sompur, Senior Lecturer, ECE Department, CIT, Gubbi who gave me valuable suggestions. Their effective advice, shrewd comments have kept the paper in the right direction. I would like to thank my friends for their constructive suggestions.

X. REFERENCES

[1] www.aim-online.com, MIL-STD-1553 TUTORIAL, AIM Gmbh

[2] www.actel.com, Core 1553BRT-EBR Enhanced Bit Rate 1553 Remote terminal

[3] Ing. Cristian Pérez, Ing. Mauricio Principi, Lic. Ariel Principi, Ing. Diego Fusari, Sr. Diego Badino, "Bus Monitor Implementation for MIL-STD 1553B protocol with FPGA device and PC connectivity"

[4] Interfaces and Standards MIL-STD-1553/1773 Basics, NASA Office of Logic Design

[5] www.actel.com, Core 1553 BRM

[6] www.actel.com, Designing a MIL-STD-1553 System Using Core1553 and Core8051

[7] Junling Tian, Kai Hu, Huiying Zhang, Jianwei Niu, Hong Jiang, "Design of MILSTD-1553B

Protocol Simulation System" 2010 3rd International Conference on Advanced Computer Theory and Engineering. L

[8] Iang Zhijian, "Research and Design of 1553B Protocol Bus Control Unit" 2010 International Conference on Educational and Network Technology.

[9] Myung-Jin Baek, Jong-In Lee, Eun-Sup Sim, Hak-Jung Kim, Joo-Jin Lee, "On-Board Management of Multiple Processor Spacecraft System" International Conference, 2002 IEEE.

[10] www.actel.com, Core 8051 Data Sheet.

[11] www.actel.com, Core 1553 Development Kit User Guide