Design and Verification of MIL-STD-1553B Remote Terminal Modules

L. Karthik, K.V. Ramana Reddy, Dr. Siva Yellampalli

***Abstract*— MIL-STD-1553B is the military specification defining a Digital Time Division Command/Response multiplexed data bus. The 1553 data bus is a dual-redundant, bidirectional, Manchester II bi-phase encoded data bus with high bit error reliability. All bus communications are controlled and initiated by a main Bus Controller. Remote Terminal devices attached to the bus respond to the controller's commands. This protocol is used for effective communication in military and aerospace electronic systems. A remote terminal typically consists of a Manchester encoder, Manchester decoder, Command decoder, Command legalization and Control logic blocks. A Remote Terminal must be capable of receiving, decoding valid legal commands from the bus controller and respond within a well defined time of 2 to 12 µs. In this paper design and verification of MIL-STD-1553B Remote Terminal modules namely Encoder, Decoder, Command decoder and Command legalization is described. The design has been implemented using Verilog HDL and has been taken through front end of ASIC flow (viz., functional verification, lint check, etc.).**

***Index Terms*—MIL-STD-1553B, Digital Time Division Multiplexing, Manchester II bi-phase, Remote Terminal modules, Encoder, Decoder, Command Decoder, Command Legalisation, Verilog HDL, ASIC design, Functional Verification, Lint Check**

I. INTRODUCTION

Digital Time Division Command/Response Multiplexed Data Bus is defined in MIL-STD-1553B [7]. It is a bidirectional, dual redundant, Manchester II encoded data bus with low bit error rate and a high reliability. MIL-STD-1553B can be effectively used for aerospace applications and in avionic systems because of its serial, 1 Mbps data rate, very low error rate of 1 word fault per 10 million words, dual redundant and high reliability. As a result of its highly robust architecture [9], MIL-STD-1553B is used as a means of efficient communication network in surface based launch vehicles, submarines, surface based targets and target drones, satellites and space systems including the present International Space Station,. This paper discusses a method for designing remote terminal modules of MIL-STD-1553B namely Manchester encoder, Manchester decoder, command decoder and command legalisation based on NRZ, manchester II bi-phase encoding and decoding schemes. The design blocks are simulated and verified for their functionality. Linting analyzes the HDL code and reports warnings and errors in the design as per DRC.

L.Karthik, Student M.Tech Digital Electronics, VTU Extension Centre UTL Technologies Ltd, Bangalore, Karnataka, India, 9483428998,

K.V.Ramana Reddy, Assistant Professor, VTU Extension Centre UTL Technologies Ltd, Bangalore, Karnataka, India

Dr. Siva Yellampalli, Professor and Head of the dept, VLSI & Embedded Systems, VTU Extension Centre UTL Technologies Ltd, Bangalore, Karnataka, India

II. MATERIALS AND METHODS

1. BLOCK DIAGRAM

a) ENCODER

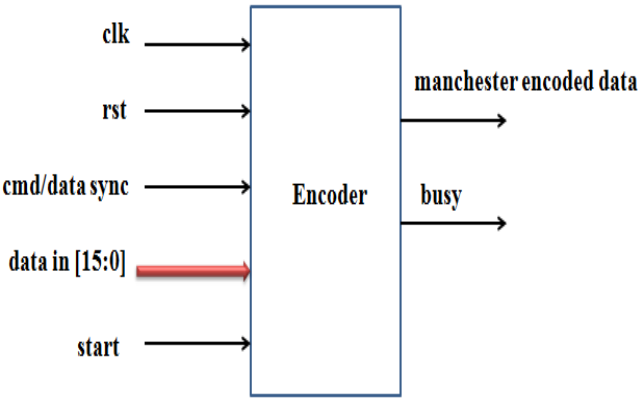

Fig 1: Block diagram of Encoder

Fig 1 shows the block diagram of a Manchester encoder. The protocol requires the 16-bit data to be transmitted in following format: command or data sync occupying 3 bit times and Manchester II bi-phase encoded occupying 17 bit times (16 bit data and one bit of odd parity) in a 20 bit word frame as shown in Fig 2.

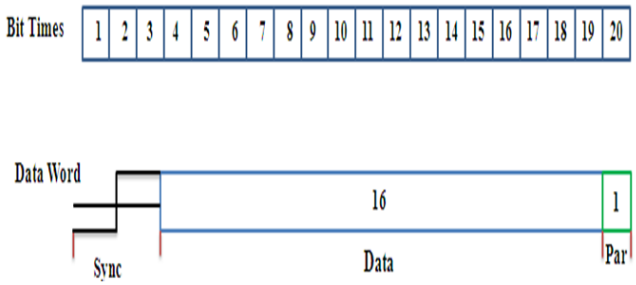

Fig 2: Word format of MIL-STD-1553B Encoder

Manchester encoding transitions at the center of the bit time and provide a self clocking waveform with equal positive and negative values. Logic '1' is a signal that transits from a positive level to a negative level. Logic'0' is a signal that transits from a negative level to a positive level.

Manchester encoding is shown in Fig 3. Encoder uses a 2 MHz clock. At power on, reset signal is asserted for two clock cycles. When logic '1' is received on the start input of encoder, it begins the manchester encoding of the 16 bit data.

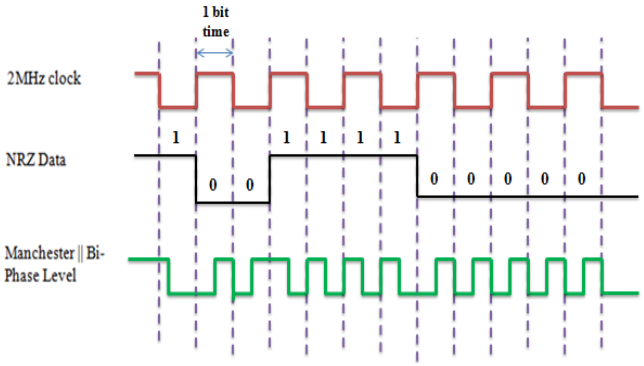

Fig 3: Manchester II Bi-Phase Level Encoding

Table I gives the description of inputs and outputs of Encoder.

Table I Input/output description of Encoder

SL NO	Parameter	Input/ output	Description
1	clk	input	2 MHz with a period of 250 ns
2	rst	input	1 clock cycle
3	start	input	2 clock cycles initially
4	cmd_data	input	Cmd or data sync
5	din [15:0]	input	16 bit hex data
6	man_enc	output	Serial manchester encoded data
7	Busy	output	Indicates the end of manchester encoding

b) DECODER

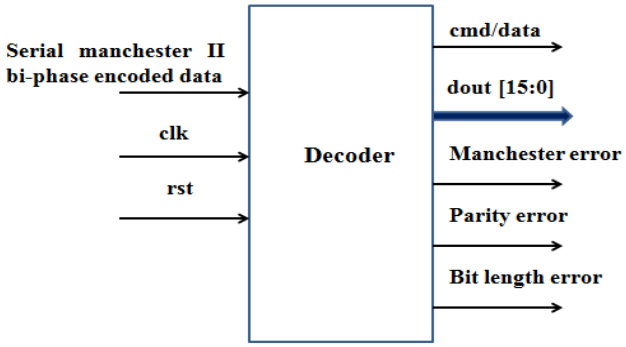

Fig 4: Block diagram of decoder

Fig 4 shows the block diagram of decoder. It reads the serial data from the 1553B data bus decodes it and verifies for valid sync and data bits. Manchester encoded data is de-serialized and the 16 bit word is decoded. Error detection viz., Manchester error, parity error and bit length error are also implemented in this module. Table II gives the description of inputs and outputs of Decoder.

Table II Input/output description of Decoder

SL NO	Parameter	Input/ output	Description
1	clk	input	2 MHz with a period of 250 ns
2	rst	input	1 clock cycle
3	man_enc	input	Serial manchester encoded data from encoder
4	dout [15:0]	output	16 bit hex decoded data
5	valid_data_out	output	Indicates whether decoded data is valid
6	cmd_data_out	output	Indicates cmd sync or data sync pattern
7	man_error	output	Indicates manchester error in the data bus
8	parity_error	output	Indicates parity error in the data bus
9	bit_length_error	output	Indicates bit length error in the data bus

Fig 5 shows the proposed finite state machine diagram for decoder based on the state transition tabulated in table III.

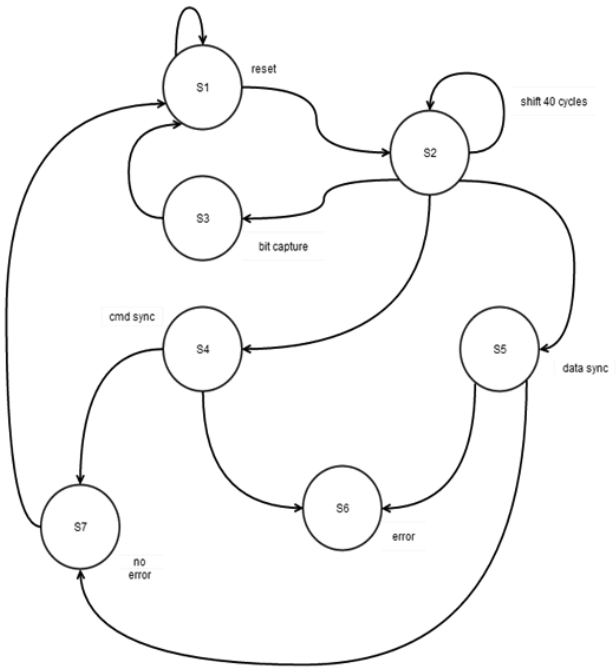

Fig 5: Proposed FSM for decoder

where, S1 is rst condition

S2 is shifting state

S3 is bit capture state

S4 is cmd sync detection state

S5 is data sync detection state

S6 is error detection state

S7 is normal decoding state with no errors introduced

Table III State transition for decoder

SL. NO	PS	NS	Input
1	rst	rst	rst = 1
2	rst	shift	rst = 1
3	shift	shift	-
4	shift	cmd sync	data_bus [47:40] == 'b11100001 || 'b11100010
5	shift	data sync	data_bus [47:40] == 'b00011101 || 'b00011110
6	shift	bit capture	-
7	cmd sync	error	man error = dout[16:0]==~dout[16:0] bit length error = (count>message)&(count-message<39) parity error = ~(^(dout[17:2])==dout[1]
8	cmd sync	no error	-
9	data sync	error	man error = dout[16:0]==~dout[16:0] bit length error = (count>message)&(count-message<39) parity error = ~(^(dout[17:2])==dout[1]
10	data sync	no error	-
11	bit capture	rst	-
12	no error	rst	-

c) COMMAND DECODER

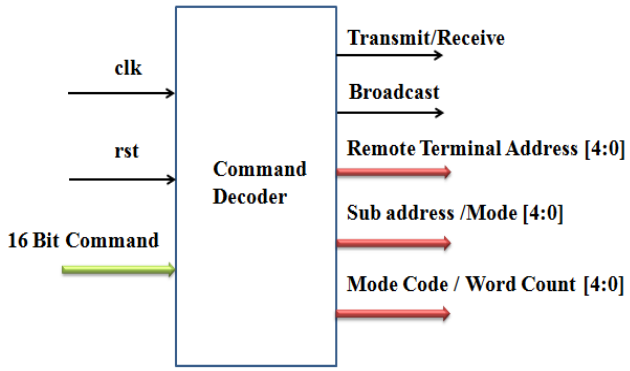

Fig 6:Block diagram of command decoder

The 16 bit command consists of RT Address, Transmit or Receive, RT Sub address and Word count or Mode command as shown in Fig 6.

Command decoder decodes the command words and also determines whether the received command is a broadcast command. Fig 7 shows the format of a MIL-STD-1553B command word.

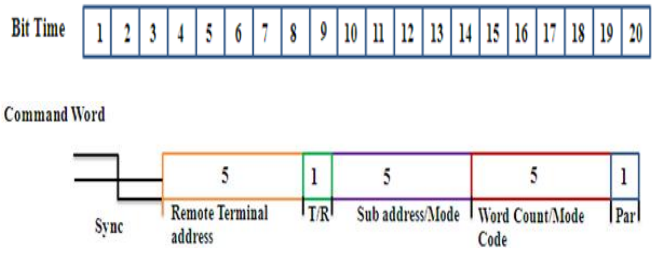

Fig 7: Format of MIL-STD-1553B Command Word

Table IV illustrates the various command word combinations along with their respective data transfer schemes.

Table IV Command Decoding

SL. NO.	Command Word (CW)	Description
1	CW[10] = 0	Normal Data Transfer-Receive
2	CW[10] = 1	Normal Data Transfer-Transmit
3	CW[15:11] = 11111 and CW[10] = 0	Broadcast Command-Receive
4	CW[15:11] = 11111 and CW[10] = 1	Broadcast Command-Transmit
5	CW[10] = 0 and CW[9:5] = 11111/00000	Mode Code-Receive
6	CW[10] = 1 and CW[9:5] = 11111/00000	Mode Code-Transmit
7	CW[15:11] = 11111 and CW[10] = 0 and CW[9:5] = 11111/00000	Broadcast Mode Code-Receive
8	CW[15:11] = 11111 and CW[10] = 1 and CW[9:5] = 11111/00000	Broadcast Mode Code-Transmit

d) COMMAND LEGALIZATION

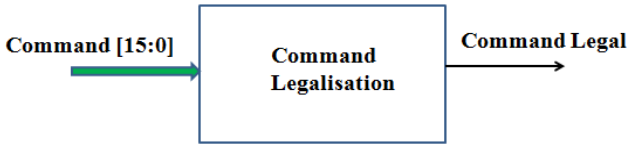

Fig 8:Block diagram of command legalization

Fig 8 shows the block diagram of command legalization module. Command Legalization block contains the logic for validating all the 1553B command words. Command word is provided as an input and the logic generates the valid or invalid output indicating the legality of the received command.

Table V shows a sample register map of command illegalisation block where all Broadcast Transmit commands are illegalised.

Table V Command Illegalisation register map

	Reg No	15/ 31	14/ 30	13/ 29	12/ 28	11/ 27	10/ 26	9/ 25	8/ 24	7/ 23	6/ 22	5/ 21	4/ 20	3/ 19	2/ 18	1/ 17	0/ 16
Receive Subaddress (0 to 31)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Transmit Subaddress (0 to 31)	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Broadcast Receive Subaddress (0 to 31)	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Broadcast Transmit Subaddress (0 to 31)	6	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Receive Mode Code (0 to 31)	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Transmit Mode Code (0 to 31)	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Broadcast Receive Mode Code (0 to 31)	12	0	0	0	0	0	0	0	0	0	0	0	0	0	I	0	I
	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Broadcast Transmit Mode Code (0 to 31)	14	0	0	0	0	0	0	0	0	0	0	0	0	0	I	0	I
	15	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I

where I-illegal command

III. RESULTS AND DISCUSSIONS

1. SIMULATION

Simulation of remote terminal modules of MIL-STD-1553B was carried out using a RTL simulator. The simulation results of the various remote terminal modules of MIL-STD-1553B are described in detail in the following sections. Verification was carried out through testbenches.

a) ENCODER

Fig 9 illustrates the working of an encoder through a simple flowchart. Fig 10 shows the simulation result of encoder which consists of cmd sync cycle occupying 20 bit times from 0.75 µs to 20.75 µs and data sync cycle occupying another 20 bit times from 20.75 µs to 40.75 µs.

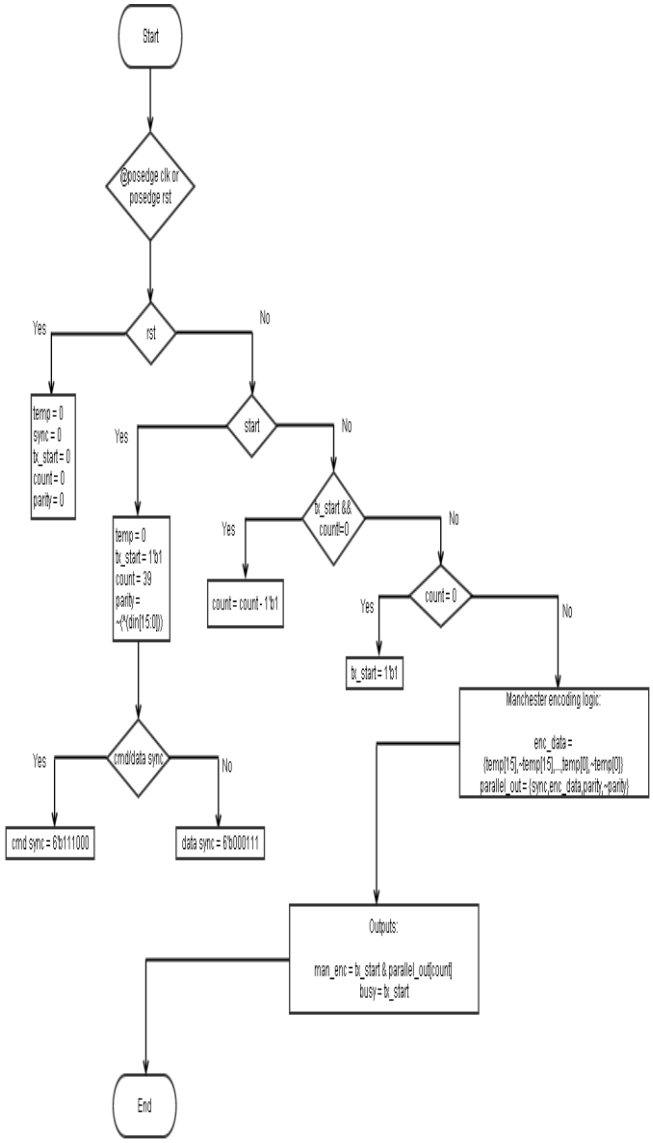

Fig 9 Proposed flowchart for Encoder

For example A5A5h was taken as the input data. Output of the encoder consists of either cmd/data sync occupying 3 bit times, serial manchester encoded II biphase data occupying 16 bit times and odd parity bits occupying the last bit time out of a 20 bit word frame as shown in Fig 10.

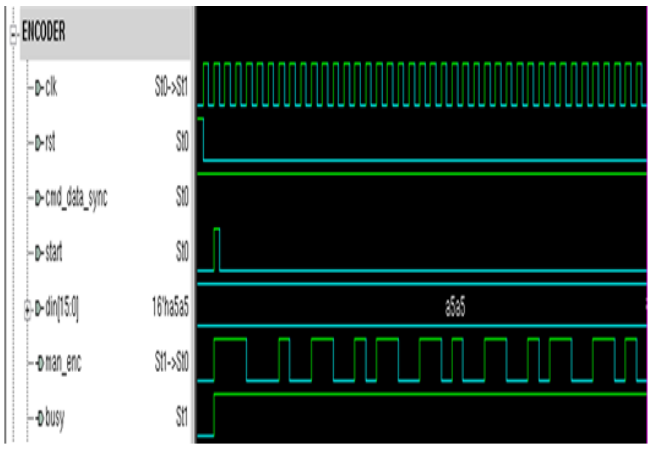

Fig 10: Encoder with sync, manchester encoded data and parity

Tables VI and VII illustrate an example of manchester encoding with corresponding bit times in µs wherein 16 bit input A5A5h is encoded.

Table VI Example for cmd sync cycle of encoding

din [15:0]	binary equivalent of din [15:0]	manchester encoded data	sync (cmd)	odd parity
A5A5	1010 0101 1010 0101	10011001011001101001100101100110 (16 µs)	111000 (3 µs)	10 (1 µs)

Table VII Example for data sync cycle of encoding

din [15:0]	binary equivalent of din [15:0]	manchester encoded data	sync (cmd)	odd parity
A5A5	1010 0101 1010 0101	10011001011001101001100101100110 (16 µs)	000111 (3 µs)	10 (1 µs)

b) DECODER

Decoder reads the serial manchester encoded data from the 1553B data bus and checks for valid sync and data bits as shown in Fig 11. When the valid signal goes high decoded data is available at the output. manchester error, parity error or bit length error, if any are also detected.

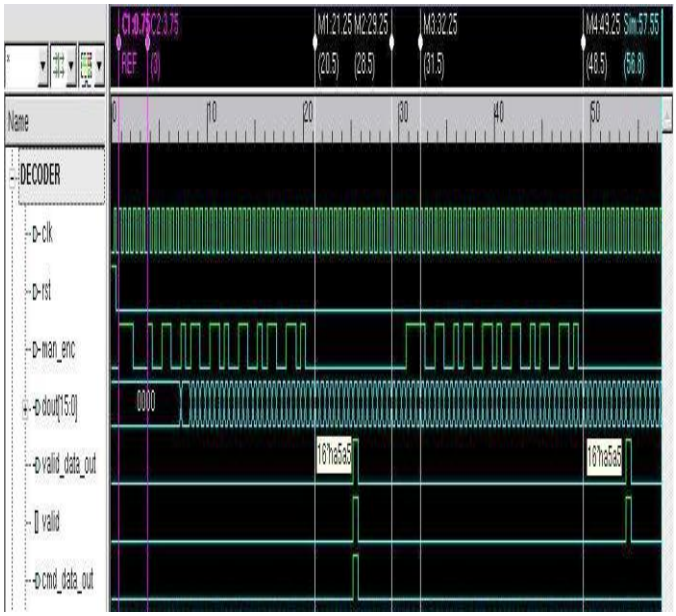

Fig 11: Decoder with decoded data and valid sync

i. Manchester error

Manchester error occurs when there is no transition present in the encoding of '1' or '0'. Error is introduced through testbench by modifying the manchester encoded data 01(for data bit '0') to 00/11 or 10(for data bit '1') to 00/11 in the serial Manchester encoded data stream. The decoder detects this as manchester error as shown in Fig 12.

For example, if the 18th bit of serial encoded data 3A84 is flipped from 0 to 1, invalid manchester encoding 00 is obtained which is detected as manchester error as shown in Fig 12.

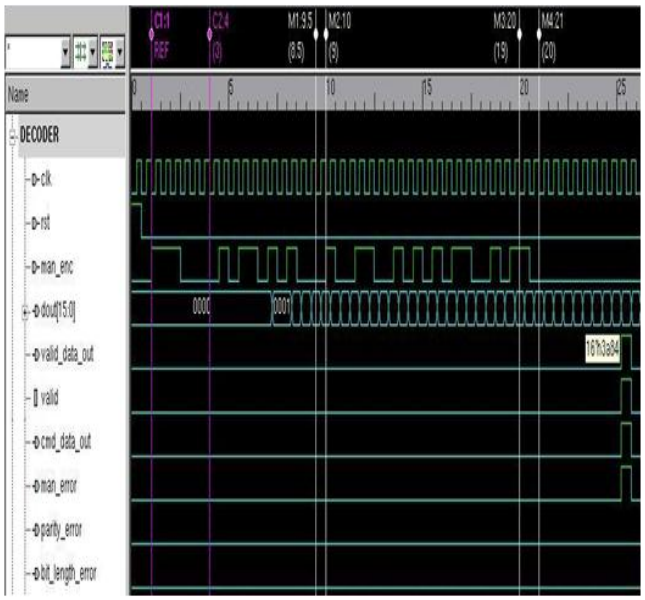

Fig 12: Decoder with manchester error

Table VIII illustrates the introduction and detection of manchester error in the serial manchester encoded data stream.

Table VIII Detection of manchester error

SL NO	Manchester encoded data	Description
1	111000 01011010 10***00***1001 10010101 01100101 10 cmd sync manchester error parity	18th bit is flipped from 1 to 0 to get an invalid manchester encoding ***00***
2	dout [15:0] = 3A84h	Manchester decoded data

ii. Parity error

Parity error is recognized by the decoder when the expected parity for the 16-bit data is not same as the parity bit received in the serial data stream. Parity error detection is tested using testbench by flipping the expected parity bit in the serial manchester encoded data stream.

For example, the expected parity of input 7C12H is '0' and the received parity is '1', then parity error output is set high as illustrated in table IX. It is shown in Fig 13.

Table IX Detection of parity error

SL NO	Manchester encoded data	Description
1	111000 01101010 10100101 01010110 01011001 ***10*** cmd sync manchester encoding parity error	Last two odd parity bits are flipped to ***10*** giving rise to parity error.
2	Expected parity = ~(^(7C12h)) = ~(1) = 0	Corresponds to 01 in manchester encoding
3	dout [15:0] = 7C12h	Manchester decoded data

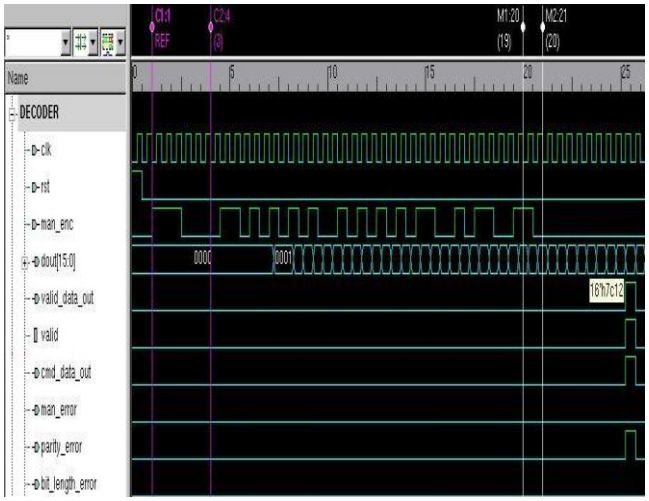

Fig 13:Decoder with parity error

iii. Bit length error

Bit length error is detected when there are missing bits or additional bits in the serial data on the bus. For example, if the last two odd parity bits of the input 95A4H after encoding are missing then bit length error and manchester error are detected as illustrated in table X and is shown in Fig 14.

Table X Detection of bit length error

SL NO	Manchester encoded data	Description
1	111000 01101010 10100101 01010110 01011001 -- cmd sync manchester encoding bit length error	Last two odd parity bits are missing giving rise to bit length error.
2	dout [15:0] = 95A4h	Manchester decoded data

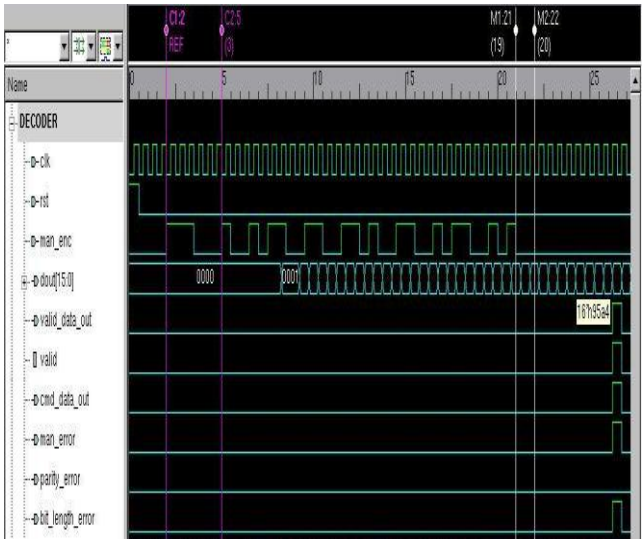

Fig 14:Decoder with bit length error

c) COMMAND DECODER

Fig 15 illustrates a flowchart for the working of command decoder. A 16 bit data input is decoded into various fields as per the word format of command decoder as shown in fig 16 and 17.

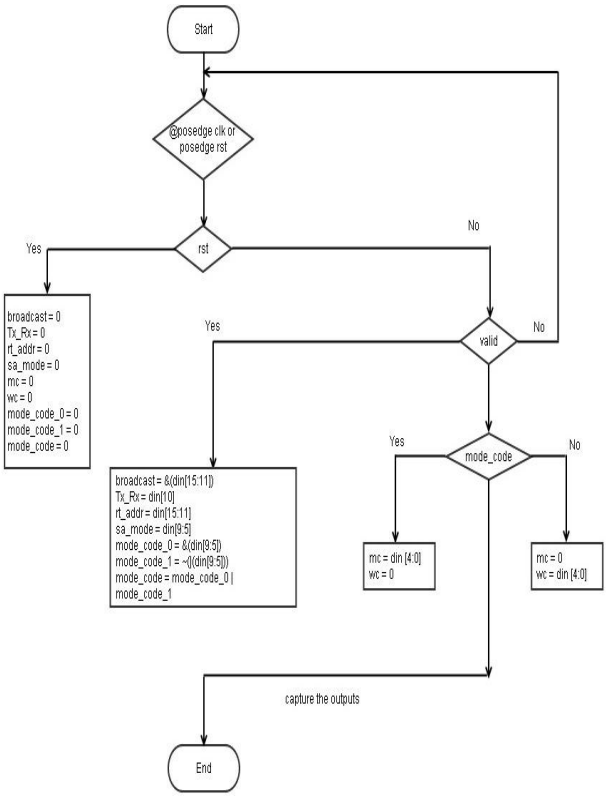

Fig 15: Proposed flowchart for Command Decoder

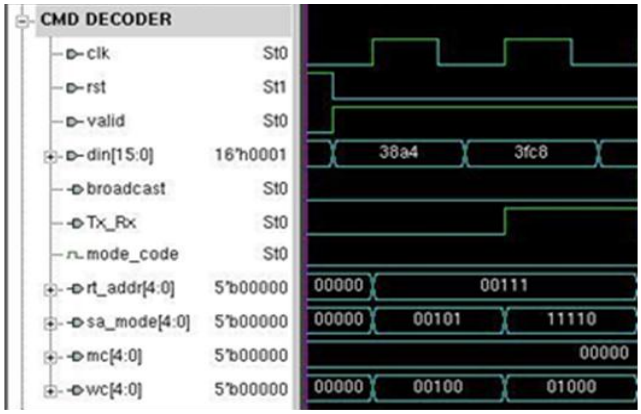

Fig 16:Decoding the normal data transfer commands

For example, normal data transfer command 38A4h is decoded into the following fields as shown in Fig 16:
RT address – 00111, Broadcast – 0, Tx/Rx – 0,
mode code – 0, Subaddress – 00101, word count – 00100

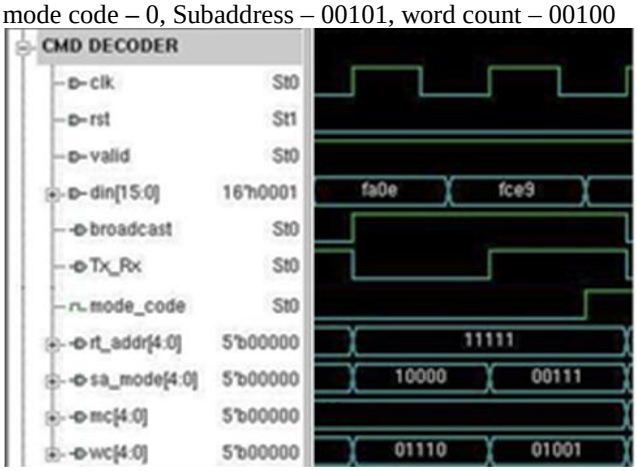

Fig 17: Decoding Broadcast commands

For example broadcast command FA0Eh is decoded into the following fields as shown in Fig 17:
RTAddress – 11111, Broadcast – 1, TX/RX – 0,
mode code – 0, Subaddress – 10000, word count – 01110

d) COMMAND LEGALIZATION

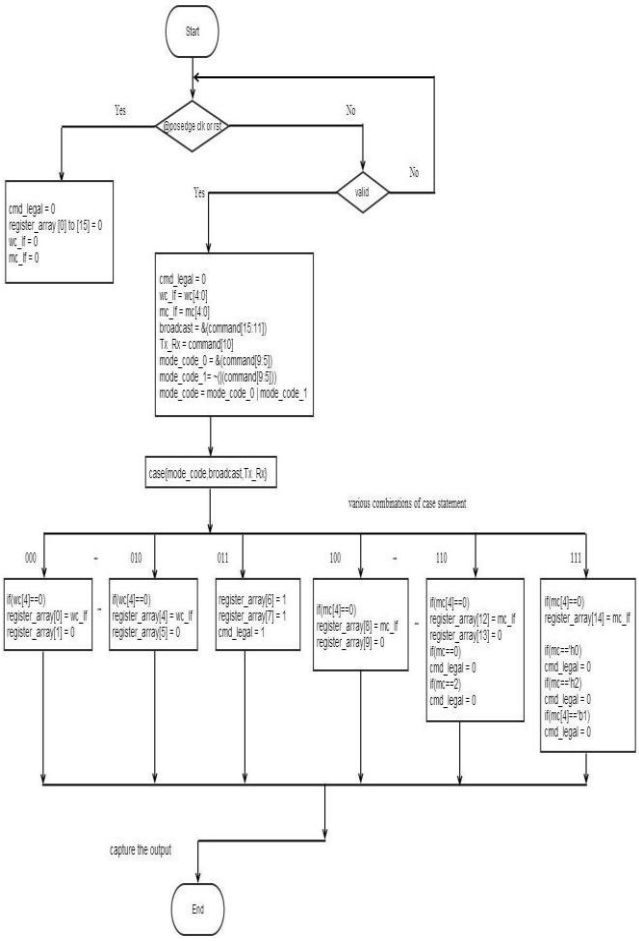

Fig 18: Proposed flowchart for Command Legalisation

Fig 18 illustrates a flowchart for the working of command legalisation. A 16 bit command is given as an input and checked for its legality, based on the command illegalization register map according to table V as shown in Fig 19 and 20.

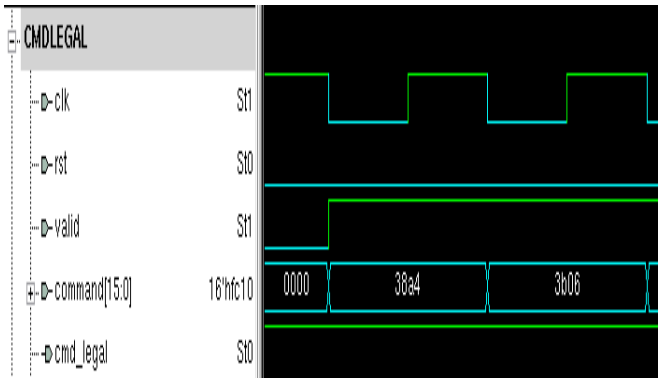

Fig 19:Command legalisation simulation window

For example commands 3A84h and 3B06h are legal as shown in Fig 19 by the cmd legal signal which is set to 1.

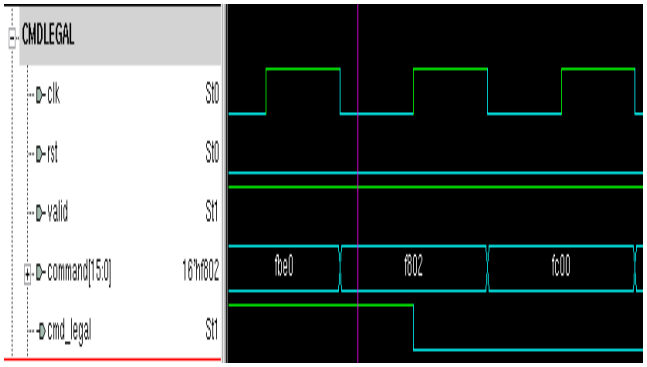

Fig 20:Detecting an Illegal command

For example commands FB82h and FC00h are illegal as shown in Fig 20 by the cmd legal signal which is set to 0.

IV. LINTING RESULTS

Linting check was carried out to detect any semantic errors in the design. Table X1 mentions the error report summary obtained using the original design.

Table XI Summary for lint check with original design

SL. NO.	RT module	Fatal error	Error	Warning	Total errors/warnings
1	Encoder	0	1	0	1
2	Decoder	0 0 0	2 2 1	0 0 0	5
3	Command decoder	0	10 1 7	0 0 0	18
4	Command legalisation	0	0	2	2

Necessary changes were made in the design to minimise the errors and they have been described in detail as follows:

a) Non-blocking statements in a combinatorial block: This error is eliminated by making use of blocking assignments.

b) Internally generated resets: This error is resolved by properly defining reset conditions in the design.

c) Unregistered outputs: This error is removed by registering the outputs with respect to clock.

Table XII Summary for lint check with changed design

SL. NO	RT module	Fatal error	Error	Warning
1	Encoder	0	1	0
2	Decoder	0	1	0
3	Command decoder	0	0	0
4	Command legalisation	0	0	0

V. CONCLUSION

In this paper we have discussed a method for designing the remote terminal modules of MIL-STD-1553B namely Encoder, Decoder, Command decoder and Command legalization based on Manchester II bi-phase encoding and decoding schemes. The design blocks have been simulated and verified for their functionality. Linting analyses the HDL code and reports all the potential design rule check violation, warnings and errors which may interfere during synthesis, post synthesis simulation, static timing analysis and other stages of ASIC design flow.

ACKNOWLEDGMENT

L.Karthik would like to express his gratitude to the director of ISRO/ISAC for providing an opportunity to carry out this project work.

REFERENCES

[1] A S Ancy Das, A Aravindhan and K R Lekshmi, "MIL-STD-1553 bus controller terminal", International Journal of Computer Engineering & Science, vol.4, issue 2, pp. 1-12, March 2014.
[2] Jemti Jose, "An FPGA implementation of 1553 protocol controller", International Journal of Computer Information Systems and Industrial Management Applications, vol.6, pp 66-76, 2014.
[3] Jemti Jose, "Design of manchester || bi-phase encoder for MIL-STD-1553 protocol", IEEE 12th International Conference on Intelligent Systems Design and Applications, pp 240-245, 2013.
[4] Suchitra Suresh, "VHDL implementation of manchester encoder and decoder", International Journal of Electrical, Electronics and Data Communication, vol.1, issue 2, April 2013.
[5] Tobias Schneider, "Civil certification of MIL-STD-1553B", IEEE/AIAA 31st Conference on Digital Avionics Systems, pp 6A4-1– 6A4-9, October 2012.
[6] Junling Tian, Kai Hu, Huiying Zhang, Jianwei Niu and Hong Jiang, "Design of MIL-STD-1553B protocol simulation system", IEEE 3rd International Conference on Advanced Computer Theory and Engineering, vol.6, pp 389-392, August 2010.
[7] MIL-STD-1553 Tutorial, AIM GmbH, Avionics data bus solutions, v 2.3, November 2010.
[8] Core 1553BRT-EBR Enhanced Bit Rate 1553 Remote Terminal, Actel Data Sheet, advanced v 1.1, pp 1-26, February 2006.
[9] MIL-STD-1553 Tutorial, Condor Engineering, v 3.41, June 2000.

ACRONYMS AND ABBREVIATIONS

The following section provides the abbreviations of the various acronyms used in this paper.

ASIC-Application Specific Integrated Circuit
BC-Bus Controller
DRC-Design Rule Check
FSM-Finite State Machine
HDL-Hardware Description Language
Mbps-Megabit per second
NRZ-Non Return to Zero
NS-Next State
PS-Present State
RT-Remote Terminal

L.Karthik, M.Tech Digital Electronics student, UTL Technologies Ltd, VTU Extension Centre, Bangalore, Karnataka, India.
K.V.Ramana Reddy, Assistant Professor and Internal guide, UTL Technologies Ltd, VTU Extension Centre, Bangalore, Karnataka, India.
Dr. Siva Yellampalli, Professor and Head of the dept, VLSI & Embedded Systems, UTL Technologies Ltd, VTU Extension Centre, Bangalore, Karnataka, India.